JUS
JOURNAL OF USABILITY STUDIES

Vol. 2, Issue 3, May 2007, pp. 112-132

Adapting Usability Investigations for Agile User-centered Design

Desirée Sy
Autodesk, Inc. (formerly Alias)

210 King Street East

Toronto, ON, M5A 1J7 Canada

desiree.sy@autodesk.com

Abstract

When our company chose to adopt an Agile development process for new products, our User Experience Team took the opportunity to adjust, and consequently improve, our user-centered design (UCD) practices.

Our interface design work required data from contextual investigations to guide rapid iterations of prototypes, validated by formative usability testing. This meant that we needed to find a way to conduct usability tests, interviews, and contextual inquiry—both in the lab and the field—within an Agile framework.

To achieve this, we adjusted the timing and granularity of these investigations, and the way that we reported our usability findings. This paper describes our main adaptations.

We have found that the new Agile UCD methods produce better-designed products than the "waterfall" versions of the same techniques. Agile communication modes have allowed us to narrow the gap between uncovering usability issues and acting on those issues by incorporating changes into the product.

Keywords

usability method, Agile, XP, iterative development, software, case study, field study, contextual inquiry, ethnography, formative usability testing, user-centered design, iterative design.

Context

As more organizations adopt Agile development practices, usability practitioners want to ensure that the resulting products are still designed with users in mind [1,2,3,4,5,6,7]. This was the situation of our User Experience Team at Alias (now Autodesk) in 2002. Our group's goal-directed rapid design process was successful, and we wanted to continue to design usable, engaging and innovative user experiences while "working Agile."

Members of our team, along with developers, were trained in Agile software development practices by the Cutter Consortium [8]. The method for gathering user feedback taught Agile team members to conduct a focus group after a feature was implemented, demonstrate the product, and ask for users' opinions. This method was insufficient for our design process, which relies heavily on observing detailed user behavior with interactive usability investigation methods, such as formative usability testing [9,10] and contextual inquiry [11] before implementation.

Our initial approaches to conducting Agile usability investigations had similarities to methods described by Holtzblatt, Wendell, and Wood [12] and some of the case studies reported by McInerney and Maurer [5], but with modifications. The differences were previously described by Miller [13] (but for an audience of developers unfamiliar with usability methods), and by Sy [14] (but with emphasis on the design iterations rather than the investigation methodology).

This paper describes our adaptations to the timing, granularity, and reporting used for Agile interactive usability investigations, with an intended audience of usability practitioners.

Agile development: a brief primer for usability practitioners

Agile programming methods are a family of software development processes [15,16,17] that share a common philosophy (more fully described by the Agile Manifesto at http://agilemanifesto.org/). Examples of Agile principles include valuing the following:

- individuals and interactions over processes and tools
- working software over comprehensive documentation
- customer collaboration over contract negotiation
- responding to change over following a plan.

In a traditional development lifecycle (sometimes called *waterfall* lifecycle), requirements gathering for all features in a release leads to a design phase, which is then followed by coding and quality assurance testing. The entire process for a release is measured in months, if not years.

In contrast, the Agile development lifecycle is characterized as a series of incremental mini-releases. (See Figure 1.) Each mini-release, with a subset of the features for the whole release, has its own requirements analysis, design, implementation, and quality assurance phases, and is called a *working version*. Each working version must be complete and stable, which makes it possible for the product release date to coincide with that of any working version. Working versions are created at regular intervals, called *iteration cycles* or *sprints*, which are generally two to four weeks long. Cycle end dates are fixed; features that cannot be completed are moved to the next working version.

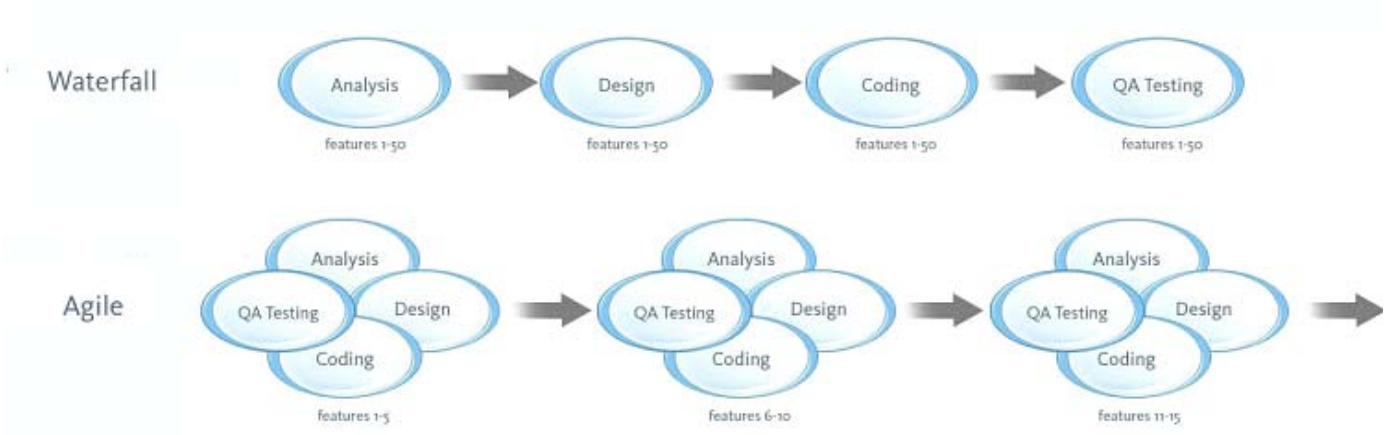

Figure 1. In a waterfall development cycle, analysis, design, coding, and quality assurance testing are separate stages of a software release that spans months or years. In Agile development, each of a set of incremental mini-releases (each created in 2-4 weeks) has these stages. Adapted from *Cutter Consortium* [8].

At the beginning of each iteration cycle, the full, cross-functional Agile team meets to do *cycle planning*. They determine the theme, or *user story*, of the next working version, and the features to put in it. Future cycles remain more loosely planned, since each cycle planning session is based on the most current information.

Cycle planning is guided by an overall vision or plan for the release. At Alias, Agile teams did release-level planning during a brief 4- to 6-week phase called *Cycle Zero*. The first iteration immediately follows Cycle Zero.

Each *feature* (the smallest development component, as defined by developers) is described on an index card called a *feature card*. Feature cards are grouped into iteration cycle clusters, and displayed in a public space for the whole Agile team as a communication artifact, in lieu of more traditional planning documents. Each feature card describes the *acceptance criteria* that determine when that feature is complete, and also includes a time estimate for completion.

The Agile team meets daily at a short, stand-up meeting (sometimes called a *scrum*), where team members each describe what they are working on, and any blocking issues. Scrums, through face-to-face communication, take the place of detailed documents to guide project planning.

Working versions are periodically delivered to users or to *customers* to validate the acceptance criteria for the feature cards. Their feedback influences current cycle implementation, and directs future cycle planning.

Note that in Agile terminology, a *customer* is not a person external to the product team who purchases or uses the product, but a role filled by one or more members of the product team. The duties of the Agile customer include acting as the voice of the end-user on the development team, and helping to prioritize and

plan cycles and releases. Miller [18] suggests that interaction designers who are willing to understand and accept Agile development concepts are well-suited to take on the Agile customer role.

This is particularly relevant to Agile usability investigations because Agile projects are highly feedback-driven, yet product teams often rely on user opinion in situations where observation is more appropriate (such as the focus group elicitation strategy described earlier). Usability practitioners can be the best-suited members of an Agile team to prevent this type of data bias because of their skills in gathering and analyzing user experience data. On the Agile projects that our User Experience Team works on, interaction designers assume the role of the Agile customer.

The iterative and incremental lifecycle of Agile development methods described in Figure 1 are similar to those of other iterative development processes (such as the Rational Unified Process). They differ mainly in the length of the iteration *timeboxes* (in Agile, measured in weeks, rather than months), the fixed nature of the cycle end dates, and the highly collaborative and document-light form of project planning and implementation. (There are other differences as well, but these affect developers rather than usability practitioners.)

Because of the similarities in their development lifecycles, the adaptations to usability investigations described in this paper may also benefit usability practitioners working on projects using an iterative development process.

Changes to the timing of usability investigations

Problems with the timing of waterfall UCD

Previously, in waterfall development projects, the User Experience Team conducted usability investigations as early as we could during the development cycle.

We performed contextual inquiry (sometimes combined with field usability testing of the prior release) before or at the onset of a project, often during a market validation for a release. During the design phase, we would rapidly iterate on key designs for a release, using formative in-house usability testing to direct the re-design of prototypes [19,20]. We would then describe the validated designs as feature specifications, and pass them on to development to be incorporated into the implemented code.

In theory, the analysis and design phases preceded the implementation (see Figure 2). However, in practice, developers would begin coding at the onset of a project without waiting for feature specifications. The result was that the implementation of some features would begin before they were designed. To combat this tendency, we investigated, designed, and validated well in advance, often conducting usability investigations almost a full release ahead. However, this led to writing many unused or out-of-date feature specifications, since we could not anticipate all planning issues (such as business goals).

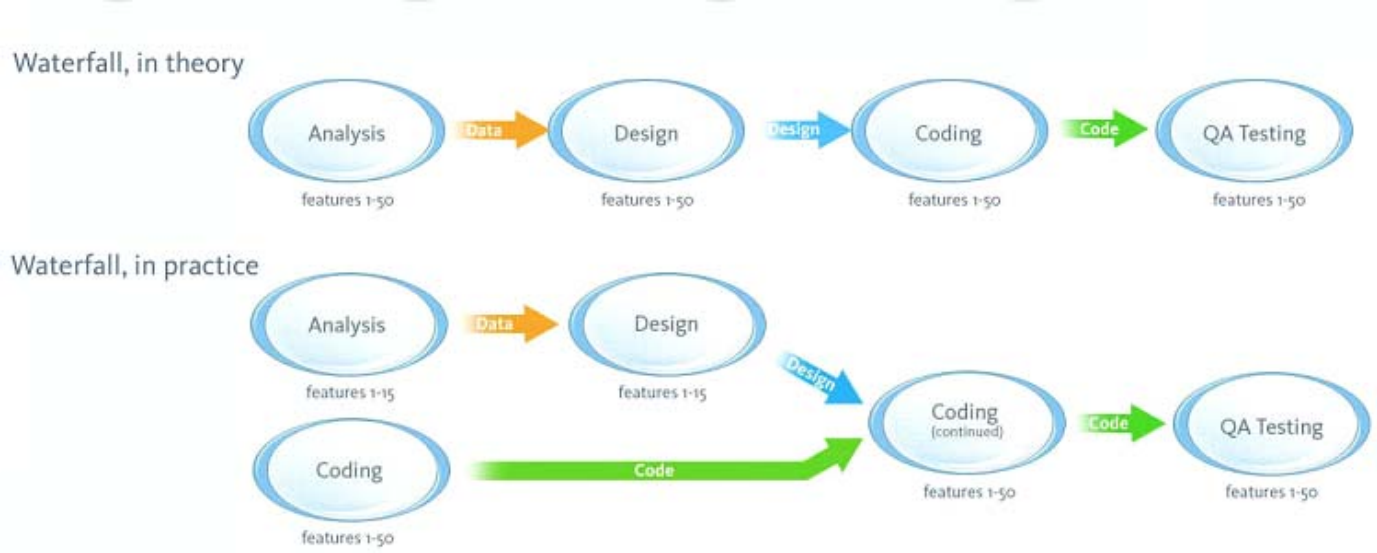

Figure 2. In the perfect theoretical version of waterfall development, usability investigations contributing to the analysis and design phases were supposed to precede coding, but in reality developers would begin coding immediately.

Also, since months often would pass between when we specified a design and the time it was coded, implementation would sometimes drift from the design intent.

Finally, development for all features going into a release would begin simultaneously, with one developer working on each feature. Because there were more developers than interaction designers, the result was that some features in a release were designed while other features were not. Furthermore, because work on all features was partially implemented, waterfall products often shipped with incomplete features, despite delaying the release date to try and accommodate feature completion.

We were caught between wanting to push our requirements gathering as late as possible, so that the most timely information would inform the product direction, and yet not leaving it so late that too many undesigned features started being implemented [13].

Just-in-time design

In contrast, on Agile projects the Agile team only focuses on a few new features at a time. This means that the User Experience Team does not have to work on all the designs in a release at the same time. Instead, we can focus on the most important designs, a few at a time.

At any given time during the Agile development cycle for a release, we conduct usability activities for only those key designs. We then work closely with

developers to ensure that implementations of the designs do not drift from the validated design intent.

Because developers are working on only a subset of features at one time, and interaction designers are designing the same subset, this also means that any features that require careful UCD work receive it. Since everything in the product must be done, traces of half-complete features don't impede the user experience.

In waterfall UCD, field investigation data would usually not have a visible product impact until the next release. Just-in-time design spreads out contextual inquiry and field work through the whole development process, instead of concentrating those activities at the beginning of the lifecycle (or in the lifecycle of the previous release). Consequently, the data that we bring to Agile projects during cycle planning is up-to-the-minute. This allows product improvements to be implemented within the current release—sometimes even in the next working version.

Two parallel tracks: iterating the design and implementation separately, but simultaneously

A key principle of our User Experience Team's UCD process is design iteration; we need to be able to catch design failures early, change designs as many times as needed, and then incorporate the design fixes [19].

Therefore, we not only need to conduct formative usability tests to check our prototypes, but we need to do so before coding begins, while the design is still malleable. Because coding begins immediately in Agile development, we needed to find a way to separate design iterations from implementation iterations.

To do this, the UCD work was done in an *Interaction Designer Track* while developers worked in a separate and parallel *Developer Track* [13,14]. The Agile UCD parallel tracks for development and interaction design are illustrated in Figure 3.

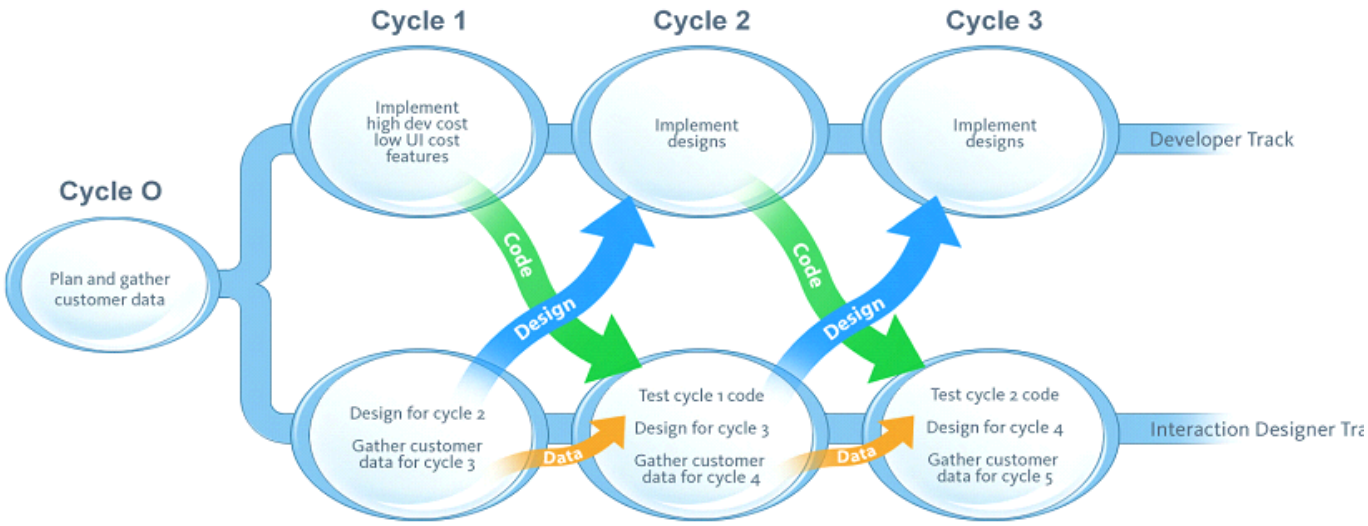

Figure 3. To allow the User Experience Team to iterate on designs, we usability tested prototypes at least one cycle ahead of developers, and then passed on the validated designs to be implemented. We would also conduct contextual inquiry for workflows at least two cycles ahead, and usability test the implemented working version to check for design drift.

Usability investigation activities in Cycle Zero

Cycle Zero is the brief requirements-gathering phase at the start of the project. Usability investigation activities in Cycle Zero depend on whether the product is the next release of an existing product or completely new. They can include the following activities:

- Gathering data to refine or hone product- and release-level goals. Facilitating the alignment of all team members' understanding of these goals, so they constitute a shared vision.
- (For a completely new product) Interviewing or conducting contextual inquiry during customer site visits for market validation. Preparing high-level exploratory designs for market validation. Based on these data, deriving the design principles that inform and guide design decisions for the product.
- (For an ongoing release) Analyzing and summarizing prior contextual inquiry and usability test data. Based on these data, elucidating release-level design goals to inform and guide design decisions through all iterations.
- (For a completely new market or capability) Developing brief and vivid descriptions of target users and workflows (light personas and scenarios) from investigations.

For example, we conducted an investigation during Cycle Zero for SketchBook Pro v2.0, a digital sketching application developed specifically for use with tablet input devices. Version 1.0 of the product had a free trial version. The release goal of v2.0 was “to improve the rate of conversion of trial users to paid users by removing barriers to purchase.” The User Experience Team helped to focus the feature set by conducting a survey targeted at people who had downloaded SketchBook v1.0, but who had not purchased it. These data helped refine the feature set for v2.0 from over 100 potential features to five major workflows. The release goal also informed design prioritization during cycle planning throughout the release lifecycle.

Usability activities for Cycle Zero of the first release of Autodesk Showcase (a real-time automotive 3D visualization product) were different. We helped the team prepare for a market validation trip to Europe, and also traveled there with the project manager and subject matter expert. We interviewed potential purchasers about their work activities for the areas that the new product would support. We then reported these back to the larger team. Also, these data were the foundation for the design principles we wrote for Autodesk Showcase that allowed us to make prioritization and design decisions as development progressed.

Cycle Zero usability activities are those that most closely resemble their waterfall UCD counterparts. However, they occur in weeks rather than months.

Usability investigation activities in iteration cycles
In Cycle 1, usability investigation activities can include:

- Designing prototypes for Cycle 2, and conducting rapid formative usability testing to refine their design.
- Conducting contextual inquiry and interviews to investigate designs for Cycle 3.

During the first few early cycles, to give interaction designers time to do these usability investigations, developers work on coding software architecture (which requires no user interface design) or important features that need only minor design.

For example, during Cycle 1 for SketchBook Pro v2.0 developers worked on adding Adobe Photoshop export. This was identified as a key issue that affected users’ purchasing decisions. It required significant development effort, but had a very simple UI (adding an item to the File Save As type list).

In Cycle 2, the designs from Cycle 1 are presented to developers, who begin coding them. Interaction designers work closely with developers to answer questions about the design as it is implemented.

Cycle 2 usability investigation activities can include:

- Prototyping and usability testing for Cycle 3 designs, using the requirements information gathered in Cycle 1.
- Contextual inquiry to investigate designs for Cycle 4.
- Usability testing the implemented working version from Cycle 1.

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

In cycles later in the release, while we continue to focus on checking the implementation of designs, we can also begin some contextual investigations to prepare for the Cycle Zero of the next release [13,14].

Changes to the granularity of usability investigations

Problems with the size of the problems to investigate
The parallel tracks allowed the User Experience Team to iterate designs before they were implemented. However, we still had to deal with the reality of cycles that were only two to four weeks long. We could complete small designs in this timeframe, but complex designs required more than four weeks to finish. We needed to figure out how to do usability investigations for designs spanning more than one Agile cycle.

Furthermore, the overall speed of Agile UCD was much faster than when we were doing waterfall UCD. We had to move much more quickly toward design solutions with a fewer number of usability tests within a release.

Design chunking: Breaking designs apart into cycle-sized pieces
We looked to the Agile model of implementation for hints about how to approach this dilemma. Working versions are implemented mini-releases that incrementally build on each other. Based on the same principles, we decided to create mini-designs that incrementally build on each other.

We break large designs into small, cycle-sized pieces called *design chunks* that incrementally add elements to the overall design. We investigate, prototype, and usability test design chunks in the Interaction Designer Track, carrying the progressively built design forward in this track until it is complete. Then, we pass the finished design to the Developer Track for implementation [14].

Interaction designers are trained to consider experiences holistically, so breaking designs into pieces—especially into pieces that do not initially support workflows—can be difficult at first, but it is a skill that comes with practice. Design chunking yields many benefits in Agile UCD, which we will describe in later sections.

To deconstruct a large design into smaller pieces, it is essential to start with well-defined design goals and to understand the high-level design intent. Our design goals are derived from observation, which is why contextual inquiry plays a critical role in our Agile UCD process. Each design chunk lets us progressively achieve a subset of the design goals.

The priority and sequence of the design chunks is determined by what we can validate at any given time in the product lifecycle. We examine the full list of design goals, and decide which we can attain with the current resources within a cycle's length. There is also an ordering dependency. In Agile projects, components build on one another, so early design chunks must be low-level and fundamental—design attributes that will not change as more design chunks are added on top of them. For example, for SketchBook Pro v2.0, we needed to design the ability to move, rotate, and scale a selected area in a canvas. Contextual investigation during usability testing of the prior release told us that these functions were experienced as one single high-level activity (positioning and fitting a selection).

Other observations were that users didn't tend to need this functionality in the first few hours of use in the application, and that it was often used with 2D layers.

A few of the design goals that we derived from these investigations included:

- Cursor travel should be minimized when swapping between move, rotate, and scale.
- The interaction should feel natural and smooth.
- Users should be able to figure out how to position a selected area with out-of-box materials, but not necessarily in the first hour of use.
- The function should work seamlessly with layers.

These goals suggested an order of approach. We could not validate the interaction of positioning a selection with layers, for example, until users could work with a high-fidelity prototype that incorporated the new functionality with the existing layer feature. This clearly was a much later stage of design for this feature, so we designated it as a late design chunk.

It was possible, however, to design and validate the kinesthetic qualities for the move, rotate, and scale modes right away, within a 2-week period using only internal resources. This was the earliest design chunk that we worked on for this functionality.

For example, we examined two possible dragging algorithms for the Rotate interaction. In one, dragging the cursor along the virtual canvas was like putting your hand on it and spinning it. In the other, dragging the cursor along an axis would rotate the canvas—this was like pulling along an invisible Rotate slider. (Figure 4)

To evaluate these design alternatives, we asked internal users (quality assurance, training, support, or subject matter experts) to use high-fidelity prototypes to rotate images, and observed the interaction. The usability acceptance criteria for the design chunk included checking whether users could rotate the canvas accurately without instruction, and whether the interaction felt smooth to them. We could get feedback from each tester in less than two minutes—we did not even have to set up informal usability test sessions to achieve this. We could just drop by someone's desk with a tablet PC that had the prototypes installed, and ask for a few minutes of their time.

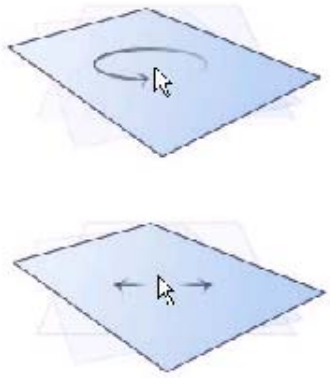

Figure 4. An early design chunk for Move/Rotate/Scale selection investigated two ways that dragging the cursor could work for the Rotate mode. In one, dragging acted like a virtual hand on the canvas. In the other, dragging along an axis rotated the canvas.

The interaction for the Move, Rotate, and Scale modes were good choices as early design chunks for the following reasons:

- We could prototype them very quickly. Our student intern coded more than a dozen variations for this design chunk within a cycle.
- It was easy to usability test them. In-house testers were sufficient to evaluate the usability criteria, and the prototypes for all modes took less than 15 minutes to test.
- We anticipated that some set-up or explanation would be needed to run the prototypes, and we knew that the testing tasks would be highly artificial, and operation-level, rather than workflow-level. For example, we told our testers that we had two different algorithms for rotating (without explaining how each one worked), and we had to intervene to switch between the two prototypes. This was fine for internal users, but it would be inappropriate and confusing for an external end-user to evaluate these prototypes.
- The interaction was a fundamental, low-level design component. There could be many different ways that we would approach how to position a selected area, but no matter what happened for later design chunks, we would need to decide on the drag interaction.
- The design problem could be validated in isolation. For the design goal we were looking at (a natural and smooth interaction), we didn't have to evaluate the three modes in combination, so prototypes for each mode could be quickly built separately. Once all the modes were combined in a later design chunk, we could address other more interdependent design problems.

In contrast, the types of design chunks that we usually complete in later cycles include the following:

- Prototypes that require an implementation or technology that hasn't been completed yet.
- Design chunks that provide workflow-level, rather than operation-level, functionality.
- Design chunks to support any investigation of a discoverability or learning goal, such as the design of how a user will access a new function. Since these are designs that depend on the first experience of a user, you need to replicate that experience to test them. These prototypes should be incorporated into a copy of a working version to avoid task bias.
- Design chunks that are hubs for other designs. For example, many different designs converged in the Brush Palette, which is why it was one of the last designs that we completed for SketchBook Pro v2.0.

At the same time that we break big designs into these small chunks, we are still completing small designs to pass to development for the next iteration cycle. Key to our success as interaction designers on Agile teams is that we keep ahead of development, feeding a steady stream of designs into the Developer Track. For this reason, we only use design chunking for a few key large designs per interaction designer in a release.

All of the illustrating examples in this article are slightly simplified for clarity. It is possible to chunk more complex designs than the one described. This is because design chunks are not complete designs. They are simply design components that can be prototyped,

iterated, and validated within Agile timeframes. By design chunking, we do not ignore high-level design considerations; instead, we work toward them in cycle-sized steps.

Progressive refinement of protocols: breaking usability testing, contextual inquiry, and recruiting into cycle-sized pieces

Agile UCD presents particular challenges in protocol design for usability investigations, because of two considerations:

- **The progressively incremental character of both implementation and design**. It is one thing to decide to design in chunks that build incrementally, but how is it possible to validate and investigate many small pieces for different designs simultaneously? It is impossible to usability test early-release design chunks with external users, and seemingly impossible to conduct meaningful contextual investigations to understand the work users might do with them. Yet we needed to explore our users' work domains to derive relevant activities, both to design later design chunks and also to provide real-world (and hence, unbiased) validation activities.
- **The fixed number of usability investigations that fit within the timeframe of a cycle**. Because Agile development is faster than waterfall, the time to create a release is briefer. Thus, we have fewer usability tests per design that we did in waterfall UCD, and in particular, we have fewer opportunities to test full workflows before they are implemented. Yet, we need to uncover more information during each investigation, since we need to collect contextual information as we progress. In essence, we need to mine more ore while drilling fewer holes.

To overcome these hurdles, we took the Agile idea of progressive mini-iterations one step further. In addition to design chunking, we also progressively performed the following activities:

- defined test protocols
- recruited test participants
- conducted contextual investigations.

Just as working versions are mini-implementations that get progressively closer to the final release, and design chunks are mini-designs that get progressively closer to the final designs, we use usability testers who get progressively closer to our end-users, ask them to let us observe them doing work that gets progressively closer to their real-world activities, and then ask them to do those activities as usability test tasks [21].

Because usability investigations are a limited resource, we need to both maximize the information we collect per session, and to hoard the validation commodity of our external users. We reserve external users to test only mid- to late-stage design chunks, and the focus of those usability tests is on validating design goals that can *only* be determined by an actual user.

For the earliest design chunks, as described in the previous section, we use in-house users who share some characteristics with our end-users (that is, people who are not developers, and with the same domain knowledge as our users). We ask them to do operation-level tasks that would probably not occur in isolation in real-world work (such as arbitrarily rotating an image to different angles, for no reason).

In later design chunks that have prototypes capable of evaluating more holistic activities, we refine usability test activities. Beginning with our internal testers, we prepare very artificial tasks. For example, as shown in Figure 5, we asked SketchBook Pro testers to position and resize the shapes to fit within the appropriate boundary lines. This was a completely arbitrary and unnatural task. We admitted this to internal usability testers, and then asked for a more realistic example of when they might use the function. We used their examples as our next iteration of the usability test task.

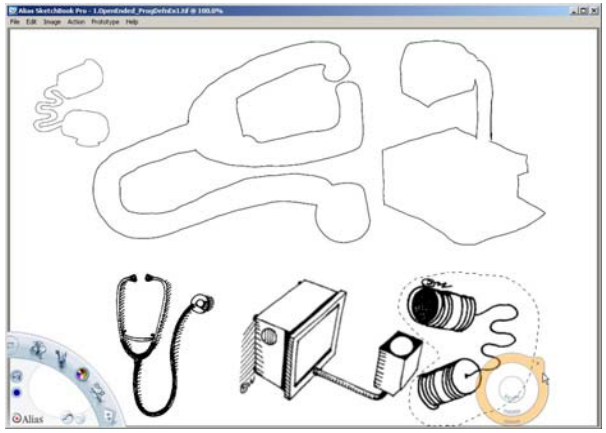

Figure 5. We asked usability testers to fit the shapes within the boundary lines to test a mid-release design chunk.

With these newer test tasks, we then recruit usability testers external to our company, and bring them in-house to our usability lab. Because the actual users of our products who are accessible to us are in limited supply, and we want to reserve them for late-stage design chunks, we find people who could be (but are not) users. For example, these testers sometimes include people studying in our users' work domains. For SketchBook Pro, we recruited computer animation students, or students with digital sketching experience from industrial design programs or art colleges.

We continue to refine our usability test tasks with these participants, asking them whether the activities we ask them to do represent how they might really use the tool, and adjust the protocols for later testers.

Finally, when late-stage design chunks are available that can emulate partial in-product workflows, we bring these *workflow prototypes* to actual end-user sites for usability testing. We call these users *Design Partners*. They commit to evaluating a longitudinal series of workflow prototypes, and sometimes also working versions. They act as expert usability testers, and also as observable beta testers. (Users who cannot commit to a series of visits are used to test mid-stage design chunks at our lab.)

For usability testing of workflow prototypes by Design Partners, we start with the more realistic test tasks derived from external usability testers. We often progress to using real work files and true work activities.

There are several progressive stages of contextual investigations that we use for Agile UCD work, but all are based on the idea of examining a user's work context for a specific activity in retrospect. These methods are described more fully elsewhere [21].

Stages of contextual investigation include the following activities:

- Interviewing and watching internal users. This provides us with the relevant vocabulary and framework to structure our interviews with external users.

- Asking external users to bring work artifacts (identified by a remote interview) to our company. This does not yield any environmental data about our users, but can still provide us workflow context.
- Asking Design Partners to set up work artifacts at their site. We visit them to watch both a high-level walkthrough of the artifacts, and a compressed workflow demonstration to view detailed interactions.
- Installing a working version at a Design Partner's site, and then watching an artifact walkthrough and compressed workflow demonstration (on- or off-site).

This last stage—the ability to observe how using the actual implemented product changes the work behavior of users—is unique to Agile contextual investigations. This contextual data can inform future designs within the timeframe of a release. Comparable waterfall contextual inquiry sessions could only guide designs for the following release.

Mixing design chunks: studying more than one design problem at a time

There seem to be an overwhelming number of usability investigation activities for any given cycle in the Interaction Designer Track described in Figure 3.

Design chunking is what gives us the freedom to solve the problem of how to do so much at the same time in fewer sessions. We combine smaller-scaled investigations for different design chunks into single usability investigation sessions.

Here are three hypothetical examples of usability investigation sessions for different design chunks of several complex designs:

- **(Early design cycle) In-house usability test, with internal users**. A 15-minute session where a QA person evaluates 6-8 very low-level prototypes for two designs by performing operation-level tasks. During the session, we ask for a better activity-level task for each tool.
- **(Mid-release design cycle) In-house usability test, with external users**. A one-hour session. Before the test, we interview the tester by telephone about a workflow for a future design (two cycles later), and ask her to bring some relevant files on a USB drive. We begin the session with the contextual investigation, watching her walk us through the files and demonstrate key interactions. We also usability test four prototypes exploring different stages for two design chunks we are designing in the current cycle. During the session, we check that our test activities are representative. If needed, we will adjust the tasks for the tester coming in the next day.
- **(Late design cycle) Usability investigation, at a Design Partner's site**. A 2-hour session, the second in a series of longitudinal visits to the same user. Before our visit, we ask the tester to set up some files to show us what type of work he was doing with the working version that we left with him on our last visit. We also interview him concerning a workflow for a future design (for the next release), and ask if he can show us the relevant files. When we arrive, we observe as he walks us through the files and demonstrates the interactions. We note

that he is using an implemented feature in an unexpected way, and he gives us permission to bring back the files so that we can demonstrate this new workflow to the Agile team. We have brought two workflow prototypes on a laptop, and he usability tests them for us. We install the new working version on his system, and watch a usability test of the two new functions that have been added since our last visit.

Breaking designs down into cycle-sized chunks gives us the freedom to mix and match different types of usability investigations into the same session, which enables us to juggle more than one design, and more than one type of usability investigation at the same time. Design chunks are what allow us to elicit more data from fewer users.

Changes to the reporting of usability investigations

Oral storytelling: showing versus writing
Agile processes value "working software over comprehensive documentation" [22], and yet traditional deliverables from usability investigations (such as formal UI specifications, use case scenarios, or usability test reports) are formal documents. To deliver on the prized Agile quality of rapid response, our User Experience Team had to figure out how to reduce the number of detailed documents that we wrote. We re-evaluated the content and form of our communication with the project team to re-align it with Agile principles.

We need to communicate the following information to the Agile team:

- which designs we are working on, and approximately when developers should expect to receive them
- usability test results and high-level progress for late-stage design chunks
- recommendations and fixes for working versions
- user and task information from external users, especially from field visits
- the user interface design to be implemented.

Examining this list, we realized that this information was only of value to the Agile team once it was in the product, not when we wrote it down. In a sense, it was interim data and did not need to be captured for posterity. In fact, since its purpose was to instigate action, written text was not the best medium to achieve this. We needed to convey it to the team in a more timely, vivid, and specific manner.

We patterned our solution on how other Agile team members communicate. For developers and project managers on an Agile team, documents such as marketing and product requirements documents are largely replaced by daily scrums and cycle planning sessions, with the displayed feature cards acting as a reference artifact for these discussions, supplemented by brief documents. We do the same.

Design Cards: communicating to plan
We give high-level progress reports on upcoming designs at the scrum. Upcoming designs are represented on the planning boards as *design cards*,

which are blue to differentiate them from feature cards, and have no implementation time estimates [14].

In addition to keeping in touch with the whole Agile team through the daily scrum, we work with developers very closely throughout design and development. Although the dual tracks depicted in Figure 3 seem separate, in reality, interaction designers need to communicate every day with developers. This is not only to ensure that designs are being implemented correctly, but also so that we have a thorough understanding of technical constraints that affect design decisions [13].

Issue Cards: communicating to persuade

We report information gathered from our Design Partners the day after the visit in a review meeting that happens after the scrum. Any interested Agile team members can stay for the presentation, and it's usually the full team. This is far more people than those who read our reports in the waterfall UCD days, and a wider cross-section of the team than those who used to listen to our debrief presentations. It often includes, for example, technical writers and QA people.

We present the following kinds of information in the form of verbal stories, supplemented where necessary by a demonstration of the prototypes or working versions to represent the observed interactions:

- successes with design prototypes or working versions, and examples of our users' work in the application
- contextual information about users' workflows, especially unexpected uses of the product
- feature requests
- major usability problems with design prototypes
- bugs in working versions.

With these stories and demos, we replace personas with people, and scenarios with workflows and sample work files.

We use index cards as a reporting artifact for these data, so the team is reminded of the presentation during later cycles. To prepare for this, on the way back from a Design Partner site, the interaction designers write out individual usability issues, feature requests, and bugs on *issue cards* [14].

After we present the information to the Agile team, team members decide what to do about feature requests or unexpected uses of the product. Sometimes an issue card is moved immediately into the cycle planning board, and becomes a feature or design card. Any bugs are logged in our bug-tracking database.

The remaining issue cards are tracked on a User Experience board in the same public space as the cycle planning board. (See Figure 6.) On each issue card, the interaction designers note any fixes for a design issue, the usability testers who found it, and the iteration progress.

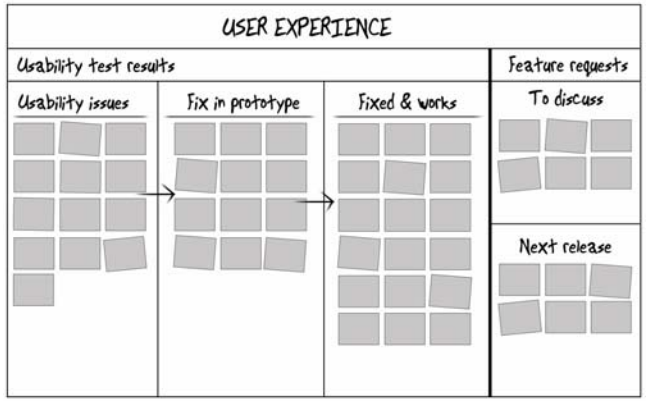

Figure 6. Issue cards are posted on a User Experience board. As prototypes are fixed, the cards move from the *Fix in prototype* area to the *Fixed & works* area of the board.

Issue cards move on the board from *Usability issues* to *Fix in prototype* when the prototype design changes. If the change solves the problem, then the card moves to the *Fixed and works* area of the board.

The User Experience board also tracks feature requests from Design Partners that are not moved to the cycle planning boards as feature cards.

Moving to Feature Cards: communicating to implement
Finally, when designs are completed, the interaction designers pass them to developers. Our Agile design deliverables are also different than the comprehensive feature specifications that we used to write for waterfall development [14].

We present the design in person to the developer(s) who will implement it. We demonstrate a typical workflow on the last available prototype using data from a Design Partner. We explain in detail any additions or changes to the design from this prototype.

If the final prototype is a high-fidelity one, we also pass the code to the developer as part of the specification.

With the developers, we then work out what feature cards will represent the implementation of the design. For a simple design, this may be only one or two cards. We ensure that usability criteria are defined as part of the acceptance criteria for the feature card before it is added to the cycle planning board.

For a complex design, we help ease the transition to feature cards by breaking the whole design into suggested implementation chunks. Whereas the order of design chunks depends on what can be validated with the current resources, the order of implementation chunks depends on creating working versions that enable a user to complete a workflow. The interim workflows will not be ideal, or even allow users to do what they want to, but working versions should enable users to complete actions that they can evaluate.

We work with developers to finalize the implementation order, and they compose feature cards based on this. Since they are responsible for completing the work on feature cards, and for creating the time estimates, developers own them. Once again, we ensure that usability criteria are part of the acceptance criteria.

A factor that has allowed our User Experience Team to modify our deliverables in this manner is that we were in the same building as the rest of the Agile team. A future challenge we may face is how to capture the immediacy and vividness of our storytelling with team members in other locations.

Documents are for the design team
Although the User Experience Team writes far fewer documents working on Agile projects than we did for waterfall projects, we have not eradicated the written

word entirely. However, we had a key insight: we ourselves are the primary readers of design documents. We can, therefore, write the shortest possible documents that still capture the information that we need to reference [14].

This information is the design history for a feature or set of features. The purpose of the record is principally to avoid “design thrash,” where design decisions are accidentally re-visited, particularly between versions, or if a new User Experience Team member is added to a project.

We generally write one document for each implementation chunk. The document is written in a medium that is easy to update (such as a Wiki page).

Information in a design history document can include the following:

- Design goals and a brief description of the problems that the design addresses.
- A high-level description of the design, including rough sketches, and a pointer to the last design prototype.
- Links to related design history documents.
- A reverse chronology of the design iterations, including the reasons for the design changes, and design limitations and constraints defined by usability investigation data as design work proceeds. Relevant usability investigation data are recorded in this chronology. This section of the document is written as design cycles progress. The oldest entry in this design chronology describes technology constraints.
- The names of the associated feature cards.
- As working versions are completed, the design chronology is extended to include additional workflow information and links to bugs related to changes in design, or unexpected uses.

To give an idea of the length of this type of light specification, the Move/Rotate/Scale selection feature for SketchBook Pro was written as two documents (one for each implementation chunk). One described the look and interaction behavior for the Move/Rotate/Scale UI widget, and the other described the interaction of this widget within a selected area of the canvas. The description sections of these documents (excluding the design chronology and feature cards) were, respectively, 1,215 and 735 words long.

Design history documents are available to the whole Agile team, but we have found that few team members read them, preferring the immediacy of face-to-face conversation to clarify issues.

Reflections

Five years ago, our User Experience Team faced the challenge of adjusting our practices. We didn’t anticipate it then, but now we prefer Agile user-centered design for the following reasons:

- More of the product is designed than before.
- Usability investigations are conducted throughout the entire product release lifecycle, rather than clustered at the front end of a release, or in the prior release.
- The most important designs are worked on first, and there is no effort wasted writing unused designs.

- Product changes suggested by usability testing and contextual inquiry investigations of the actual product can be implemented in the current release.
- Just as formative usability test results allow us to iterate on the *design* of a product, now the Agile team's responsiveness to contextual inquiry results allow us to iterate on the *requirements* for a product.
- Implementations of designs are more true to the design intent. The overall quality of the user experience in our Agile products is higher than in the waterfall products we worked on.

Much of the power that drives user-centered design is generated from observational data gathered in usability investigations. By adjusting the way that we derived and reported these data, we could harness its power to the Agile characteristics of speed, responsiveness, and high implementation quality to better fuel the UCD engine.

Acknowledgements

When working Agile, it's best to make haste slowly and take a moment for reflection. I thank Autodesk for giving me some of that time. The other key contributors on our User Experience Team to the described process innovations were John Schrag and our fearless leader, Lynn Miller. They, Veronica Meuris, Stephen Gaebel, Amy Kidd, Peter Liepa, and Catherine Courage read early drafts, and helped me to clarify the article. However, all errors and omissions are my own. Illustrations in the article are by Marsha Leverock.

Practitioners' take-away

Here are some practical implications of the concepts discussed in this paper:

- For our User Experience Team, Agile user-centered design resulted in better-designed software than waterfall user-centered design. Agile communication modes narrowed the gap between gathering usability data and acting on it.
- Because Agile development is highly feedback-driven, product teams may rely on user opinion in situations where user observation is more appropriate. Usability practitioners can be the best-suited members of an Agile team to mitigate this bias because of their skills in gathering and analyzing user experience data.
- It is possible to use the familiar arsenal of usability investigation methods on Agile (and other iterative development) projects, including formative usability testing, user and task analysis, interviews, and even field-based work like contextual inquiry. This is achieved by changing the timing and granularity of the investigations, and how results are reported.
- Just as Agile (and iterative) development implement software as incremental mini-releases, usability and design activities can be scoped as incremental mini-designs. Different validation and elicitation activities can be blended within single sessions conducted at a usability lab or in the field. Design activities occur at least one Agile *cycle* or *sprint* ahead of the development team in an *Interaction Designer Track* separate from the *Developer Track.* Developers receive validated designs.

- Prototype demonstrations and daily conversation have largely replaced detailed documents, such as usability test reports and UI specifications, when communicating with the product team. Documents are now written for interaction designers, to record a history of design decisions.

References

[1] Constantine, L. L. (2002). Process Agility and Software Usability. *Information Age*, August 2002.

[2] Beck, K. and Cooper, A. (2002). *Extreme Programming vs. Interaction Design*. Retrieved on December 8, 2006 from www.fawcette.com/interviews/beck_cooper/

[3] Patton, J. (2004). *Interaction Design Meets Agility*. Retrieved on December 8, 2006 from www.agilealliance.org/system/article/file/1368/file.pdf

[4] Pearson, G., and Pearsall, S. (2005). Becoming Agile: Usability and Short Project Cycles. *User Experience*, 4(4), 2005.

[5] McInerney, P., and Maurer, F. (2005). UCD in Agile Projects: Dream Team or Odd Couple?, *interactions*, 12(6), November + December 2005, 19-23.

[6] Lee, J.C. (2006). Embracing Agile Development of Usable Software Systems. *Proceedings of CHI 2006*. Montréal: ACM.

[7] Lievesley, M.A., and Yee, J.S.R. (2006). The Role of the Interaction Designer in an Agile Software Development Process. *Proceedings of CHI 2006*. Montréal: ACM.

[8] Highsmith, J. A. III (2002). *Agile Software Development* (course notes). Arlington, MA: Cutter Consortium.

[9] Dumas, J., and Redish, J. (1999). *A Practical Guide to Usability Testing* (revised edition). Bristol, U.K.: Intellect.

[10] Rubin, J. (2001). *Handbook of Usability Testing: How to Plan, Design, and Conduct Effective Tests*. New York, NY: John Wiley & Sons.

[11] Holtzblatt, K., and Beyer, H. (1996). *Contextual design: defining customer-centered systems*. San Francisco, CA: Morgan Kaufmann Publishers.

[12] Holtzblatt, K., Wendell, J.B., and Wood, S. (2005). *Rapid Contextual Design*. San Francisco, CA: Morgan Kaufman/Elsevier.

[13] Miller, L. (2005). Case Study of Customer Input For a Successful Product. *Proceedings of Agile 2005*. Denver: Agile Alliance.

[14] Sy, D. (2005). Strategy & tactics for Agile design: a design case study. *Proceedings of UPA 2005*. Montréal: Usability Professionals' Association.

[15] Highsmith, J. A. III (2000). *Adaptive Software Development: A Collaborative Approach to Managing Complex Systems*. New York, NY: Dorset House Publishing Co., Inc.

[16] Beck, K. (2000). *Extreme Programming Explained: Embrace Change.* Boston, MA: Addison-Wesley Professional.

[17] Schwaber, K., and Beedle, M. (2002). *Agile Software Development with Scrum*. Upper Saddle River, NJ: Prentice Hall.

[18] Miller, L. (2006). Interaction Designers and Agile Development: A Partnership. *Proceedings of UPA 2006*. Denver/Broomfield: Usability Professionals' Association.

[19] Schrag, J. (2006). Using Formative Usability Testing as a Fast UI Design Tool. *Proceedings of UPA 2006*. Denver/Broomfield: Usability Professionals' Association.

[20] Medlock, M., Terrano, M., Wixon, D. (2002) Using the RITE Method to Improve Products: A Definition and a Case Study. *Proceedings of UPA 20*02. Orlando: Usability Professionals' Association.

[21] Sy, D. (2006). Formative usability investigations for open-ended tasks. *Proceedings of UPA 2006*. Denver/Broomfield: Usability Professionals' Association.

[22] Beck, K. *et al.* (2001). *Manifesto for Agile Software Development*. Retrieved on December 7, 2006 from agilemanifesto.org

[23] Highsmith, J. A. III (2004). *Agile Project Management*. Boston, MA: Addison-Wesley/Pearson Education.

Desirée Sy has been crafting useful and enjoyable user experiences for over 16 years, the last 11 of them at Autodesk (formerly Alias). Customers and trade reviewers have described the user interfaces she's designed as both innovative and a pleasure to use. She has presented and taught on topics, such as Agile user-centered design, gathering good usability data, usability investigations for open-ended tasks, and low-cost usability investigations at UPA and ACM SIGDOC.