CHAPTER 2

Agile Methods + UX = Agile UX

CONTENTS

INTRODUCTION

Just as Agile is a really big term that can refer to the values, the manifesto, and all of the process that emerged from the movement, *user experience* is also a broad and sometimes controversial term. The formal definition that the Nielsen Norman Group uses ("User Experience: Our Definition," n.d.) is

> "'User experience' encompasses all aspects of the end-user's interaction with the company, its services, and its products. The first requirement for an exemplary user experience is to meet the exact needs of the customer, without fuss or bother. The next is the simplicity and elegance that produce products that are a joy to own, a joy to use. True user experience goes far beyond giving customers what they say they want, or providing checklist features. In order to achieve high-quality user experience in a company's offerings there must be a seamless merging of the services of multiple disciplines, including engineering, marketing, graphical and industrial design, and interface design."

Agile User Experience Design. http://dx.doi.org/10.1016/B978-0-12-415953-2.00002-9

(This definition can be found on the Nielsen Norman website at www.nngroup.com) Since Donald Norman is one of the earliest and most influential users of the term, the definition is the foundation of how many people understand the concept of user experience.

In my career, the words *usability*, *interaction design*, *product design*, and *user experience design* have appeared at different times in my title, even though my role in each of these positions was fundamentally the same. This is why, for some people, *user experience* feels like just another of the many buzz words used to describe user-centered design. For me, it means something specific or rather something that is very inclusive. I respectfully disagree slightly with the Nielsen Norman definition, which I think encompasses what many people now commonly refer to as *customer experience* (a term that evolved recently). I consider *user experience* to be slightly narrower than what the words describe, something that is more focused on the interactions and use of the product itself than speaking to the entire organization that produces, sells, and markets the software. Working with a sales representative to buy a license or a customer's ability to access helpful marketing materials seems to fall more appropriately under the broader term of *customer experience*, although it would be within the boundaries of *user experience* as defined by Norman. Part of this is because the customer and the user may not be the same person, especially in the case of enterprise software. When using these terms in this way, it is possible that the customer who negotiates the license or decides to upgrade to a new version may not be the same person as the user who actually touches the product. Of course, in many cases, they are the same, especially in the case of websites and consumer software.

I specifically see the idea and practice of user experience as an umbrella that describes only the end users' interaction with the product; including their environments, motivations, thoughts, and reactions to their interaction with the product. This means that UX considers not just the customers' ability to execute a task using an application but takes into account where they use it, why they use it, what other tasks and tools that they might need to use. It also speaks to their level of delight, frustration, comfort, and trust evoked as part of performing their tasks using that product. For example, if you design an application for use by business travelers and think of using audio and voice commands, you need to be aware that they will often use it at busy airports with lots of background noise, which might make it difficult to hear audio and challenging for the system to process voice commands. This could lead to a frustrating user experience, even if the user accomplishes the tasks after great effort and even though the product design might be very attractive. From a process perspective, UX also refers to the spectrum of activities that contribute to the resulting product—research, design and usability testing.

I refer to *UX practitioners* throughout this book for the sake of brevity. However, it is worth pointing out that t many people may not actually think of themselves as a UX practitioners even if it appears in their job title. People in many domains, collectively, could be considered as contributing to the user experience of a product—researchers, usability specialists, information architects, product/interaction/user experience/interface designers, and visual designers. So many job descriptions are used to classify those of us in roles that support the creation of a product's user experience, but in the end, all of us are focused on contributing to the creation of something that meets or exceeds the customer's wants and needs. A user researcher might study the target audience for a banking application and learn that trust is an important part of the user experience for a certain group and certain things influence that group's ability to trust in the application. A company may then have a product designer, an interaction designer, or a visual designer work on the design of the product to ensure that the elements establishing or influencing trust are incorporated in to the resulting application because of the information provided by the researcher. A usability specialist might facilitate usability tests to validate the design, and one of the many dimensions that he or she explores will be the issue of trust. The researcher then feeds the information back to the team, possibly with recommendations for improvements. In some companies, some or all of those roles might be filled by a single person. Because the theoretical "UX team" might be an actual team or teams or simply one person, I choose to use a single umbrella term to represent all of such efforts.

In describing the banking application, it is clear that there are many places for potential hand-offs, customer touch points, and activities just within the formation and refinement of the user experience. Looking at the list of tasks, it can be easy to imagine that these things require a lot of time to do thoroughly and properly. In good conscience, I have to acknowledge that, to do a truly thorough and proper research, design, and validation requires a certain amount of time, effort, and resources. Regardless of whether or not an organization is Agile, there are limits to how thorough a job one can do, due to normal constraints on time or resources. As a result, the various UX disciplines have not often had the luxury of time to conduct their efforts. Since it might be possible to spend an infinite amount of time creating a design or conducting research, often we make do with filling the time allocated to us. The crux of the challenge of fitting UX efforts into Agile methods is that we still need to do the same activities we have always done but do them even more quickly than before, because less time probably is allotted to the release in general. This is where we need to look at what we and how we are doing while thinking about what Agile values. It may not be a case of shrinking to fit as much as working differently and more efficiently

to achieve and identify the tasks that contribute the most to creating a fantastic user experience.

FITTING A UX PEG INTO AN AGILE-SHAPED HOLE

In the previous chapter, we explored Agile values and principles and saw their connection to the goals and values of user-centered design. It is clear that many of the core values of Agile resonate quite well with those employed by your average user experience designer/researcher. It would be hard for a UX practitioner not to be excited about a process that emphasizes collaboration, communication, and consideration of the customer's needs and accommodates changes in requirements whenever they occur. That is exactly what most UX teams wish for. The core values that are the foundation of Agile methods can provide significant benefit to any organization that is genuinely committed to creating a culture that embodies those values. However, Agile methods are not a magic bullet and may not be an ideal fit for all projects or organizations. In fact, an organization that has existing challenges and simply trains one or two people in Scrum, then expects to "be Agile" will find that using an Agile process only shines a spotlight on those issues. The difficulty is because the obstacles that tend to undermine Agile processes are generally around communication or other cultural issues and typically the kind of problems with which organizations prefer not to deal. The reluctance to address these things is because they are tough to resolve and require a lot of soft skills that may not be available on the staff. The benefit of engaging fully in a detailed Agile process is that it can help to encourage better behavior on these fronts.

This especially affects the UX teams, because we are often in the role of supporting the work of development teams and tend to be directly affected by organizational issues and communication breakdowns. Just "going Agile" without laying the proper foundation does not result in an effortless transition to faster production of more user-centered products. Doing so really requires a team or organization to truly engage in fostering an environment of collaboration and communication. This means that if you are struggling to get usability feedback listened to by other functional areas or design specifications are considered to be more of a suggestion than an instruction with the development process currently in place, you are likely to find that is still the case if your company goes Agile. In fact, moving to an Agile methodology might serve only to make these issues more obvious to everyone on the team and bring things to a head quicker in the cycle. If there is a lack of communication between different team members or some functional areas are considered less important than others, these dysfunctions become more obvious when Agile methods are in place. However, if the move to Agile is thoughtful process and

existing issues are acknowledged and dealt with properly, it certainly can be an effective change agent.

Instead of the organization trying, from the top down, to change the way that teams work, in an Agile environment, each team can contribute to a more grassroots level of fomenting change. In companies where only a few teams are Agile or all the teams are Agile but not necessarily embodying Agile values, the deck may seem stacked against you. However, ultimately, the team controls its own culture; and no matter how friendly or unfriendly the environment that the team is in, it is up to it to define its own values and set the tone for its behavior. Of course, this requires the teams to be very conscious of their adaptation of Agile, they need to be committed to the spirit of Agile and not focused exclusively on the details of the process. As part of the production team, the UX staff is in a position to influence the attitude of the team and model good behavior. Just as teams that model Agile values and have successful projects can persuade other teams to follow their example, UX team members can help set the tone of the production team by embodying Agile values themselves. The project team can be the agent of change within the organization and the UX person can be the agent of change, if necessary, within the team.

The best indicator of a project team's dedication to Agile values is the likelihood that it will be successful in Agile, the best opportunity to facilitate healthy communication and the most influential time to sway the culture is during the retrospectives. Regardless of which Agile method is practiced, there will be some form of re-evaluation and discussion. In Scrum, this activity is an explicit ceremony, which ensures that even teams not comfortable having this kind of discussion will at least make an attempt to do so. When these meetings allow for candid conversation and team member's feedback is taken seriously and acted on, there is a good chance of success for the project. Since this is the best time to raise issues and seek a resolution with the involvement of the entire team, it is critical that everyone feel comfortable doing so. If the team members feel like their suggestions will be dismissed, then this meeting will end up being a missed opportunity. The UX person will not likely be in the role of scrum master and so will not run the meeting or directly facilitate its tone. However, the UX person can help to set or change the dynamic of the meeting if he or she is willing to identify areas where the team could have done better during the sprint, is open to feedback on that topic, and acts on that feedback. Not only will the UX person benefit from the discussion but it allows the individual to be an active participant in driving the tone of the conversation and sets an example for the broader team. If a team member sees that the UX person is willing to do this and the outcome is positive, then it might inspire the member to behave the same way.

Another factor that can create challenges for an organization trying to embrace a culture of Agile are the logistics of the team and scale of the project. If a team is large, more than 10 people, or geographically dispersed, then scaling the method effectively is difficult but not impossible. The team should be aware of this and that it might need to modify its tactics to achieve the same outcome as a smaller, more colocated team. These particular obstacles can often affect UX people disproportionately, because they require extra effort and creativity to both collaborate on and communicate designs and to share research findings. Often, the project team responds to such a situation by asking the UX person for more documentation, a request that needs to be examined closely before agreeing to accommodate it. It is important for the UX team to use some of its resourcefulness in finding solutions to some of these challenges and the best answer to solve the problem rather than falling back on what might be more familiar, comfortable, or easy but does not really move the team forward in their Agile evolution. In this case, consider whether a document is really what is needed—Would an interactive prototype do the trick as effectively and with less effort? Would a video conference give you the most bang for the buck?

Working in an Agile environment can free you up to try new things; and if you approach the process from that angle, it will broaden your solution set. Moving to this process certainly is an adjustment, but it can also be a good excuse try new techniques and take more ownership of defining the UX role. If you were working with more traditional methods or the UX team had been in place for some time at your organization, you may not have had direct influence over how your role would integrate with other functional areas. This can be an opportunity to change that redefines how the UX team works. One thing to consider about Agile, however, is that, even if the UX role is broadly defined for the organization, each Agile team works in a slightly different way to fit its team members. Even if you do not get to define a new role for UX across the company, there is certainly room within Agile for your project team(s) to leverage your unique skills to great effect and with your direct input.

Probably the most important issue for the UX person to bear in mind is that out-of-the-box Agile methods tend to focus exclusively on the process of creating code and pay little attention to other functional areas. This is because, with the exception of Lean UX, they are intended to be development methods and not necessarily address the whole of what it takes to release a product. While this is such an obvious thing to point out, sometimes the reminder is necessary. I heard people dismiss Agile methods because they do not consider the UX team, the quality team, the documentation team, or any other team in the company. Well, it is true—and it is on purpose. Nor is it a bad thing. Most Agile methods are trying to tackle one problem—how to create high-quality, customer-centric code. If the development team elects to use this process, the burden is on the surrounding teams to define how they fit into the new way of

working, keeping the Agile values and principles in mind. There tends to be little or no guidance for how a UX team can or should expect to fit into the cycle. This is an opportunity and should be embraced as such. We may not be the decision makers who control whether or not to use an Agile method or even which one to use, but we are completely in the driver's seat when it comes to defining how UX will fit into the Agile world. While a development team may have the luxury of picking a prepackaged Agile methodology and consuming it whole, the UX team has the responsibility (and freedom) to take a more thoughtful and targeted approach to writing the rules of engagement for working in an Agile way.

When I first started working on Agile projects and needed to figure out how my UX work fit into that picture, the common wisdom was for the UX team to work a sprint ahead of the development team on either design tasks or gathering user feedback. In fact, this perspective was so prevalent at that time and had so many enthusiastic adherents, that it did not even occur to me to consider working any other way. It also seemed very logical, given the pace of the sprints and my novice Agile status. While the development team worked out its infrastructure issues in the planning stages, I could conduct customer interviews and flesh out some of the user stories. Once we got underway, all of us could collaborate on design solutions before they reached the implementation stage; and once they got there, the developers would know what to do based on our discussions and the artifacts I put into a shared location. Transitioning away from traditional waterfall UX was daunting enough for me; and the sprint-ahead framework felt like a manageable solution for my first Agile outing. When working in advance of the development team, the UX team's deliverable of a design, lightweight spec, or customer feedback would be delivered to the entire team in the end of the sprint demonstration; then the design would be implemented by the development team in a future sprint. We chose to include the UX work in the demo partly because the spirit of that meeting for us was to show all the work that had been completed and it was the best way to reach the entire team without needing another meeting. The best practice for this style of Agile UX is to also to have a Stage 0 for the entire team to engage in planning and during which the UX team could conduct user research and do some high-level design to help guide future work, although that was not something I was able to do in this particular case, due to the timing of when UX was brought onto the project. Lynn Miller (n.d.) reported one of the earliest and most comprehensive examples of working in this way and describes her positive experience with this technique in "Case Study of Customer Input for a Successful Product," which can be found online at http://agileproductdesign.com/useful_papers/millller_customer_input_in_agile_projects.pdf. (This is just one of her papers, seek out more of her work if you are considering this approach for you or your team.) Quite a few examples are published on UX teams working in parallel

sprints and achieving collaborative design results that provide great user experiences to their customers.

However, others express concern that working this way is tantamount to mini-waterfall. While this is not always an accurate description of what is happening for the teams who have been conscious, collaborative, and purposeful in their methods, it is a legitimate comment and a fair criticism. An inherent risk is that working this way might compromise the cooperative spirit of Agile. Since a typical sprint moves at a very fast pace, it may be challenging to get the time from developers to work collaboratively with the designers. This can be mitigated by having dedicated time from key team members or assigning tasks to the right developers to make sure they are given some slack in their schedule to work with the design team. But, if the development team needs to both support the work and its sprint as well as the work in the design sprint, what is the real value in parallelizing the work? The answer for your team may well be that there is a benefit to doing so, but some conversation around this question is needed to determine whether this is the right approach for your team. If the benefit is only for the UX staff and causes a burden for the developers, then the team needs to agree that it is the right trade-off. If everyone stands to gain, then work in parallel and reap the benefits.

When doing so, you should be aware of the danger in working in separate sprints, since there is less visibility and insight in to what the design team does. The design people end up being a bit distanced from the rest of the team and focused more on their own work. While one or two developers might be involved in the UX efforts, for other team members the design work certainly might become a black box. They might hear about it in daily scrum meetings (or not, depending how the parallel sprints are handled) and see the results in the demonstration, but beyond that, it is a bit of a mystery. To prevent this, create more opportunities to discuss and advertise the UX work throughout the course of the sprint. One of the benefits of working within a sprint can be a heightened awareness of what each team member is working on and a real sense that the team is working as a cohesive unit to achieve a common goal. Working in different sprints can affect some of the camaraderie and allow a design silo to be created. However, many teams happily and successfully used this technique and gave presentations on their experience—Lynn Miller has frequently reported on this topic, and her work is a great resource for insight into how to make this style work for a team while avoiding any of the dangers. As with any Agile method, it is important to keep reflecting on your process and seeing how you can become more efficient.

In interviewing different UX team members and managers, it is clear that as some UX teams spend more time in Agile environments, they evolve new ways

of working with the process that do not necessarily match the parallel sprint approach. This evolution is a very healthy thing and can look different for each team. While working within a sprint might sound like the UX team works at breakneck speed, it can sometimes actually result in a more manageable pace. This is because it is easier to work collaboratively with the development team members, as they will be tasked to the same user story that UX supports. Since the work is being done in tight coordination, it is not likely to require much in the way documentation or the production of deliverables. The UX person will hash out the design issues with the development team in real time as the code is developed. It will be a very concentrated effort, but since it requires less time spent on manufacturing an artifact, it may be a more efficient use of UX time. It also relieves the UX team of trying to get it all done ahead of time, a smaller-scale version of the big up-front design burden that occurs in more traditional methods. One challenge in working ahead in an Agile environment is that things are much more unpredictable than in the traditional methods, where priorities and scope are fairly static. It is entirely possible to be working ahead on an item whose priority has changed before it gets in the hands of development, and it ends up being scoped out or deferred. The UX team cannot afford to waste that kind of time very often in an Agile environment. When working within a sprint, the collaboration tends to be more seamless and the design person works with the developer(s) to come up with a design solution as the developer is writing the code, and tweaks will be made along the way during the course of the sprint as issues are identified. This can actually be more manageable than trying to fill every moment of a sprint, front-loading design work so that UX is not a bottleneck to development, having the design ready for implementation, then finding out, a few sprints down the road, that the design has difficulties and needs to be reworked.

Working within the sprint does not necessarily mean that an earlier sprint was not dedicated to planning, high-level design, and design thinking or user research. Nor does it mean that time cannot be set aside during the release to do those things. It may, however, mean that the UX people have to estimate their efforts and track their velocity to fit their efforts properly into the sprint. Both methods have their pros and cons and may work better in some situations than others. If you are not sure which method suits your team or you can see the value in both, you can try to mix and match the two. They are not necessarily mutually exclusive; if you can find a way to strike a balance between the two approaches and it works well for your team, then that could be a good solution for you.

The pace of Agile development can be very fast and it can be easy to get lost in the work of the current iteration and lose sight of the big picture. Taking an iteration to conduct research and design or working a sprint ahead of development allows the UX team to maintain more control over the pace of its work

and create the opportunity to take a breath. After a few projects or a few years, the UX team might want to work differently and feel more comfortable working within a sprint with the development team, once the UX team better understands the rhythm of the Agile cycle and establishes a good rapport with the project team. Or working so separately might feel awkward to you, and you would prefer to be in the same sprint as the development team from the get-go. Even among UX teams practicing Agile and effectively integrated in to the process, there is a wide variety of approaches and styles. Figure out what works best for your team and know that the only wrong answer is one that does not facilitate the production of a good user experience.

THE UX WORK

Regardless of how the UX team chooses to integrate with the development events, it can expect that some of the activities it is used to doing will need to be refined. The timing and the rhythm of a waterfall cycle is very different from the timing and the rhythm of the Agile cycle, even if the work that needs to be done remains the same. No matter how much Agile and UX values have in common, real work still must be done by the UX team. Simply understanding that people are more important than process does not provide enough guidance for the UX team to know which techniques should be used and when, which ones need to be modified (and how), and which tactics need to be dropped altogether. Development teams have a much clearer path for moving from one methodology to the other, because the processes are development centric. Since UX team does not have the same guidance, moving to Agile is a good time to re-examine the techniques the team uses to see where it can grow. Just like moving from Alaska to Florida, you need to get rid of your favorite snow boots and your heavy sweaters and start shopping for shorts and tank tops (but maybe you can hang on to your jeans and T-shirts). We start the inventory process by looking at the activities the UX team engages in and discuss what it means for the team to fit into Agile.

Resourcing and staffing

For the sake of brevity and simplicity, I have been generalizing as if the UX team is a part of traditional development environment moving wholesale to an Agile process and taking the UX team with it. I also speak in general terms as if one UX person supports a single Agile team. While this does happen, it is an oversimplification for the purpose of illustration. Realistically, it probably does not match the situation that you are in. I know that it does not describe any position in which I have been. It is certainly important to acknowledge that UX practitioners might find themselves in many other possible configurations, and each of these presents unique challenges.

Some organizations do not dictate specific development methods but have certain criteria in place, and as long as those are satisfied, a development team can elect to use whatever development method it wants. Other companies subscribe to a single development process but allow incubation teams to explore Agile and apply it to their release cycles as an experiment. In both situations, the UX team, or a single UX resource, might find itself straddling both Agile and non-Agile projects at the same time. This can be difficult, confusing, and a little bit exhausting. Instead of being able to mentally prepare for the switch from one method to another and really immerse yourself in an Agile state of mind, you may find yourself with one foot in both worlds. It can be daunting enough for a UX person to learn about Agile, sort out how to fit into that method, and get ready to work at a faster speed. To have your focus split between the old world and the new makes adapting to the pace of Agile even more challenging.

For those UX people who support Agile and non-Agile projects, I can say from experience that it is not easy, but it can be done without too much trauma if you have a good plan of attack going into it. The main reason you need a strategy in place before you tackle the work is that everything in Agile is very urgent and demands your immediate and undivided attention. Without a course of action laid out, you will get overwhelmed very quickly. Even if you have not yet experienced Scrum and are not sure what it will mean for you day to day, you can think about the practicalities of supporting two demanding projects, irrespective of the methodology they use. If you plan on supporting your Agile team on Mondays, Wednesdays, and Thursdays, how will you handle daily standups and requests for collaboration that come in on "off" days? Consider whether or not this is a team that you have worked with before and whether you have established a trusting relationship with them. If you need to invest in a certain amount of relationship building, then plan on being available for more (if not all) team events and meetings. Think about the level of experience the team has with UX and if it has a good understanding of what you do and how you do it. No matter what ther level of familiarity is with UX activities, you need to devote time to working with the team to define your role with it and understand what kind of support it needs. If the team is not familiar with UX activities or some of the techniques you are planning to use, you may also need to do some evangelizing and education to manage their expectations.

Also think about how you will handle unexpected requests for advice. If you ask a developer with a tight schedule to wait a day to talk to you because your Agile day is Tuesday, it is pretty likely that he or she will just move forward without your advice and not come looking for your input in the future. If you can at least hear the developer out with a commitment to thinking about the problem and getting back with an answer by lunch tomorrow, it will go a long way to establishing a healthy dynamic. It might also be necessary to make sure hours

are dedicated to the Agile team every day, since it is not very collaborative to tell a team member that you do have no time until next week because you used up your Agile project time allotment by Wednesday. Being flexible and allocating hours of the week to each project could work well, as long as you are very strict about it. Since it is always very challenging for a UX person to say no to providing help and the Agile project always is in need of urgent help, the reality is that the non-Agile project that might suffer when time becomes tight. It is easy to convince yourself that there will be more time later on to do the necessary work for the non-Agile project, but if you trade off in favor of the Agile project every time, you will absolutely run out of time. Keeping a balance between the two projects requires more discipline than supporting multiple traditional projects. It can be very effective to allocate hours during the week, as is the idea of managing your level of activity in a given sprint. Some sprints are design heavy relative to others, and it may make sense to choose to focus on only the Agile project for that sprint. Similarly, if the non-Agile project has a spike in its need for the UX person, then it could make sense to take that sprint off from the Agile project, except for some maintenance, consulting hours, or time to attend team meetings. Perhaps, the non-Agile project needs intensive design activities, usability testing, or a solid chunk of time generating your specification. You could commit to not taking on any tasks for that sprint cycle and, aside from the agreed upon support activities, focus on your non-Agile project. This requires discussion with the team, but it is likely to be very understanding. After all, it probably feels overwhelmed with its one Agile project and very sympathetic to the fact that you are supporting multiple projects.

The same factors that you analyze for staffing Agile and traditional projects apply if you support multiple Agile projects. However, in this situation, you are less likely to have one project with more urgency than another, as both will be operating at a high level of urgency. Unfortunately, you may not be able to attend all the events for both projects, since balancing multiple daily standups, regularly occurring planning meetings, and retrospectives can leave you with little time to do any work. You need to be more strategic in how you spend your time and decide which meetings are valuable both for information exchange and team building. You do not want to be viewed as less of team member because of your split focus, as that could inhibit some of trust and collaboration you are trying to engender. Perhaps, attend only a few of the standups each week, but make sure that your status is communicated to the team even for the meetings that you cannot attend in person and work with the scrum master to be made of aware issues that arise in your absence.

If you do find yourself only supporting one Agile project, you may still want to do dedicate time to longer-term research projects in addition to your design work. In that case, treat the research as a separate project and manage your time

and effort that way. The main difference is that, instead of it being an entirely separate piece of work, you may occasionally bring the results of your research back into your Agile project. Of course, the entire load balancing effort requires that your teams understanding what your commitments are, where you are spending your time, and why. It really requires that you keep the doors of communication open, so that your team can let you know if your strategy is causing them any concern. It is also possible that your first attempts to balance the load may not work very well in the beginning and require a bit of refinement. That is why communication is so critical to maintain, since the team may notice problems in your strategy even before you do.

Tactics

- **Communication**. This is a recurring theme for all UX and Agile activities. Agile environments can be pressure cookers, especially for teams that are not as experienced. Failing to manage expectations about the UX role and the amount of time you will spend with the team can result in tension and compromise your ability to be seen as a fully vested team member. This can compromise your ability to work collaboratively and really hinder your efforts. It is also important to provide the team with insight into your work and why it is important for you to spend time on a particular design issue or run a round of usability testing. While everyone might be content for the UX team to simply do its thing in a waterfall environment, there is a greater team investment in individual activities in Agile. Part of developing the desired transparency is not to just let the rest of the team know what you are doing but to provide them with an understanding of why it is important that you do it.
- **Planning.** The process is very fluid and there will certainly be a need for replanning, but it is critical to have a rough plan, so you can see when you start to deviate and need to look at a course correction or simply adapt your plan. Sorting out what your teams will need from you and when helps anticipate your peak demands times and whether you need to streamline any of your methods.

Specifications

In many companies, specifications is *the* deliverable created by the design team. These can range from a short brief describing a feature to a novel-length tome that describes every bit of the interactions in a product (Figure 2.1). The document may be created as an expression of the design intent, to support communication, as the design statement of record, or as the single point of contact between design and development. Once a team goes Agile, the need for heavy-handed documentation is replaced by frequent communication and collaboration. For the UX practitioners comfortable and skilled at creating

FIGURE 2.1
Specifications.

heavy documentation of their design, it may be a challenge to let go of doing so. Some practitioners may find it freeing to no longer have to labor long hours over documenting the minutia of their design intent, but others may be reluctant to relinquish the illusion of control that such a process imparts. It can feels as if the act of putting every detail into writing results in a process endorsed document of record, which of course means that the resulting software has to match the specifications. Unfortunately, this is rarely, if ever, the case. Despite the benefit of no longer having to spend days and weeks working on such a large effort of questionable value, it can be hard to move away from what might be the UX team's largest and most concrete contribution to the release. For managers who are transitioning an existing UX team to Agile, this is an area that may lead to skill-set mismatches. A staff member who prefers to plod quietly along on a design solution and spend a significant amount of time working through every design detail and documenting it, may be very challenged to work as quickly, collaboratively, and with more uncertainty as he or she must in an Agile environment. Replacing paperwork with real communication absolutely takes getting used to and may come more easily for some people than others. Some team members may also just not like the change in working style, which is fair—Agile is not for everyone.

For a colocated, highly collaborative team, where all the functional areas are well represented, there is no real need to create any kind of specification. Daily communication and intense collaboration should replace the need for any written document. If the team is working well together and focused on a given piece of work, then all the members should be on the same page and have a common understanding and shared vision, which is realized in the implementation and the documentation. Since Agile is so focused on the creation of working code, the software should be produced relatively quickly and can be reviewed in short order to verify that this assumption is true. With a faster turnaround of the design, the team finds out sooner if there are any misunderstandings, and if there are, these can be corrected in fairly short order.

In practice, teams are often spread across time zones and countries, and some resources may straddle multiple projects and be less able to participate in the necessary conversations to achieve mind share. When practical issues such as these arise to slow down the speed of communication, it is not uncommon to produce some artifact to help speed things back up. Wireframes, .html prototypes, and screenshots are all fair game and likely being produced by the UX team anyway as part of the design process. Even producing lightweight specs that contain sketches or screenshots with callouts and a small amount of text (I like to call them *speclettes* to reflect their diminutive nature) can be a great solution in many situations. The benefit to these, if they are needed, is that they are no burden to produce, can be done as needed, and can pay off huge dividends by ensuring that everyone on the team is on the same page, even if they cannot be in the same room together. Any artifacts that are created should live in a known shared location, team members should be aware of when they are posted, and finding them should be effortless. The system used to create and track tasks, if there is one, can be easily leveraged for this purpose. If the team does not use such a system, a commonly accessible location like a Wiki page or a SharePoint folder works well, too. One caveat to bear in mind is that, before producing any kind of documents, even speclettes, the UX team should examine why it is being done and if it is the only or the best solution. If one part of the team is located in Boston, another part in Shanghai, and yet another in San Francisco, then it is probably reasonable to have a Wiki page or shared documents to gain a common understanding, since face-to-face communication is limited. International teams often appreciate the ability to view some kind of screenshot or sketch before having a conference call, to give them the opportunity to digest the information and formulate their questions. This is a best practice for any kind of remote collaboration, as it can help make a conference call more efficient. However, if the UX team posts documents because the colocated documentation staff or quality team is not brought on to the project until late in the process, it would be better to consider options such as bringing them on earlier, even if only part time, or having on-boarding meetings to bring them up to speed with the state of the project and the design. Direct communication is always more effective and usually more efficient than even the most interactive prototypes. If the team is completely colocated and all the functional areas support the project, but there is a request for documentation, this should be considered a red flag. It is natural to look for a document to replace or supplement communication, because documents served that role in traditional methods. But, if a team is looking to lean on documents when there is no logistical reason for doing so, this can be a sign of a process or communication breakdown and the team should most definitely look for a solution beyond just creating a document, since that will likely only be a quick fix that may not really address the root problem.

When replacing this particular deliverable, the UX team should think about communication and how to facilitate it better and without creating the overhead a large, single-purpose document requires. Before it became a bloated item on a checklist for release criteria, the specifications were intended to let all the functional areas know what the design should look like and how it should behave. Anything that replaces the specifications should serve that purpose while addressing the needs of the team and being lightweight to produce and to consume.

Tactics

- **Conversation and collaboration.** Engaging the entire team in design exercises greatly reduces the need for any design document. The team members know what has been designed because they were a part of the process of creating it. Having white board sketching sessions, conducting card sort activities, and engaging the team in participatory design sessions can help the UX person generate a lot of ideas very quickly, serve as team building exercises, and educate everyone about what design is.
- **Coding.** It is not unheard of for UX teams to have someone on the team who can write code that defines the front end and pass that to the developers to use. This is a more common occurrence for website or web applications than desktop applications, where the code base may not lend itself as well to doing this. But, what better way to influence the design than to code the presentation layer? Doing this not only ensures that the front-end interactions are implemented as intended but is an effective way to increase consistency. If styles and standards are built-in and the development team simply has to code the back end, there will be significantly less low-hanging fruit to deal with later.
- **Prototypes.** While it does not always make sense to have the UX team writing production code, it can still produce reasonably high-fidelity .html prototypes with very little investment. This allows the interactions of a screen to be shown, instead of the flat layout captured in screenshots and sketches. Illustrating the interactions provides clarity and eliminates the need for interpretation that often happens with a static picture. Not only is this a great communication tool to use internally, it can be reused for a usability testing session. The prototypes can be done witn a tool like Axure or simply be a set of screenshots pasted into PowerPoint with a few hotspots to simulate interactions. Regardless of how robust these items are, they may be worth the investment if they help remove communication obstacles.
- **Wireframes and sketches.** Certainly, these are the most lightweight kind of documentation and, since they are often created as part of the design process, might not require additional effort to produce. The sketches can be

paper and pencil drawings, whiteboard sketches photographed by a cell phone, Balsamiq sketches, or Photoshop masterpieces, but the important thing is that something is captured in a visual way. It very easy for a group of people to discuss a UI and come to agreement without realizing that all of them had very different visions in their head. Sharing an image grounds the conversation and gets everyone on the same page, so the proposed design can be discussed productively. It is even better if producing these items is done with members of the team, so it becomes not just a document but a collaboration exercise. When that is the case, sharing the item reminds everyone of the agreed-upon solution. If a few words are added to it to provide some explanation, then it can inform members of the team who were not present for the discussion.

- **Speclettes.** These should be used only as a last resort; and even then, it is best that they be used only with teams that are spread across time zones and have a limited amount of direct interaction. A speclette is a document that might contain a few screenshots, some callouts describing elements of the UI, and a few explanatory sentences. They should require no significant effort to produce and be used when the recommended methods of communication are impossible due to logistics. They should be no more than a page or two in length and specific to a particular task or user story. They should have as little as possible in common with their more formal relative, the specifications. They should contain the least amount of information necessary to convey the appropriate information to the intended audience and no more than that. They need not be polished or fancy, but they must be easy to find and access and clearly mapped to a user story or task.

User research

We consider two main types of user research (Figure 2.2), each of which fits into an Agile process in a slightly different way. Design research is done during the release to validate the design and general product direction. The solicitation and use of user feedback during the design cycle tends to fit fairly easily into an Agile rhythm. With its focus on iteration and support of refactoring, tweaking the design based on customer reaction is a natural part of the cycle and well accommodated by the process. Longer term research is used to determine user needs, define personas, and may contain other more significant efforts that take a bit more time and effort. These larger, more extensive or more formal pieces of user research can still work in an Agile process but may happen in a different way and with different timing than in a more traditional environment.

Integrating customer input regarding the design into an Agile cycle is very doable. It is advisable to come up with a strategy around this before engaging in the project life cycle, just as when working on a more traditional project. Many

FIGURE 2.2
User research.

teams have adapted to the rhythm of Agile can accommodate predictably recurring testing events, often weekly. A certain day of the week (or month) may be set aside for gathering user feedback. When that is the case, users are constantly being scheduled and whatever is ready for feedback is shown. Anything can be put in front of the customer—a concept, a sketch, a paper or digital prototype, or working software. One benefit of Agile, though, is that you might very well have working code to put in front of the customers and are more likely to have access to a development prototype early on than in a more traditional process. Bear in mind that such frequent testing is an ideal state, as it represents the (literally) continuous feedback and customer interaction valued by Agile, but it is not necessarily achievable for every situation. If you have easy access to customers, then it is worth building that into the schedule and seeing if it can be achieved. Not planning for it in on the schedule ahead of time makes it highly unlikely that weekly or even biweekly feedback sessions will occur. Having such a constant flow of customer insight might transform the designs in a way that results in a more powerful user experience. However, if this does not feel like a reasonable option for your situation, another choice would be to identify key milestones and perform usability testing around those dates, gather user research ahead of the sprint cycle to drive early design, or simply squeeze in the testing when it makes sense. While being a little more spontaneous about scheduling the usability testing is unlikely to result in weekly test

sessions, looking for opportunities and sprints where time can be set aside for them is not entirely unrealistic.

Many techniques can fit easily into a more streamlined timeframe, so you may not be as limited in your choices as you might think, even for more high-level research. It might mean thinking beyond the methods and techniques you are most comfortable using and exploring different options. For defining user stories or influencing the design, it might be worthwhile to investigate using a technique like Indi Young's mental models, which she discusses in *Mental Models: Aligning Design Strategy with Human Behavior* (2008). This technique allows a researcher to collect information about user behavior and visualize it in a way that it makes very accessible for team members. Card sorting, especially if it is done using an online tool, can quickly yield results that assist with design decisions relating to information organization and terminology. The online version can even be done with remote users and, since it happens without moderation, have a very low overhead in terms of the UX team time required. Surveys also have an extremely low cost to the UX team in terms of the time required and can quickly yield actionable information about the users or the features from a large number of participants. Information from web analytics can be brought to bear on the design or feature definition without extra work by the UX team beyond reading a report. Unmoderated remote testing can fit in at any point, since it requires no administration. While it does not yield as much rich data as classic, face-to-face usability testing, it might help supplement efforts during the release cycle. Kyle Soucy's "Unmoderated, Remote Usability Testing: Good or Evil?" (n.d.) takes a look at the pros and cons of this type of effort and some of the vendors of this service. If you consider using remote testing, it would be worthwhile to review her findings. Her article can be found online at www.uxmatters.com. Moderated usability testing, either remote or in person, can be made to fit within the cycle as well. It might simply require working with a smaller sample size than if there was more time. Another tweak might be to run shorter sessions that have a more narrow focus than you typically do. It is also possible to make the testing more efficient by simply running the sessions over a shorter period of time and turning around the results more quickly. Usability testing can certainly be done within the sprint cycles with minimal refinements, and the information gained from these sessions are well worth the effort.

For iterative design and customer collaboration consider RITE (Rapid Iterative Testing and Evaluation), a method that applies user feedback to the product immediately and proceeds to test it again, change the design, and test it again. This technique is gaining popularity in Agile circles, because it can drive significant change in the design very quickly and really uses the customer as a collaborative design partner. A case study written by Jeff Patton about salesforce.com's use of the technique can be found in the article "Getting

Software RITE" (Patton, n.d.). Another, more detailed source for understanding the technique can be found on the Microsoft website in t "Using the RITE Method to Improve Products: a Definition and a Case Study" (Mollock et al., n.d), which fully explains the method.

If this is your fist time working with Agile, be prepared that the pace of Agile can be overwhelming at first. Because of this, if you have no plan for what kind of techniques you will use and how you will use them, you will find at the end of the release cycle you have done little or no research. Taking a proactive approach to ensure that user feedback is part of the design cycle by looking at the schedule and finding where it can fit comfortably makes your life much easier in the long run. The reality is that the need for research does not change simply because the schedule moves faster, it still needs to happen and it might take a little more creativity to do it. However, including user research somehow in the sprint cycles helps keep the user perspective at the forefront of the product design, and that is what Agile UX is about.The more significant research efforts need to be handled more carefully, since they require more time and effort, neither of which is in large supply in an Agile release. It is not impossible, but it will require even more planning and work to incorporate into the release than the design research. This is one of the few places where Agile might make you long for the waterfall method, where the slower pace seems to allow an easier fit for this kind of research. Unfortunately, this is also the area about which there is the least amount of guidance for UX practitioners. This is not because research is not happening but because so much of this work is happening outside the sprint cycles and does not always make it into the discussion about Agile UX. Just as the importance of design research exists regardless of whether a team is Agile or waterfall, the need for customer interviews and quantitative usability tests does not go away just because a team changes its development practice In fact, it might be fair to say that this type of research is even more necessary when engaging in Agile. Since the teams move so quickly once a release is kicked off, it is very important to have a clear vision and direction before the work starts. When testing is done in a sprint, the work is treated, from a process perspective, like any other task the UX person might do. The UX person reports on it in standups, tracks the effort, and fits the event within the boundaries of the sprint.

This kind of work is challenging to fit into a sprint cycle, not just because it can take longer than a single sprint but because the type of information it yields may not be easily fed back into the process. With design feedback, it is easy to turn issues into backlog items, prioritize them, and get the changes into the product. Larger research efforts yield bigger results, and these findings lend themselves better to being incorporated into the planning stages, even in traditional environments. Many teams have some kind of planning period, it may be called *Sprint 0*, it may be a time when the development team is working

out infrastructure issues and is not ready for design feedback, or it may be a window of opportunity before the project is formally kicked off, where some research can be done. All these moments can be used as opportunities to do research and allow the UX team to deliver findings that will influence the release cycle.

It is also possible to work outside of the sprint cycle altogether and treat research as if it is a completely separate project, in the same way you would carve out time to support multiple products. Another option, which may not be available to everyone, would be to resource it to a different person, who will manage the research work based on the project's needs. In both cases, these tasks are not tracked by the scrum master, they are not necessarily reported on in daily standups, and they need not be constrained to occur within a sprint-based timeframe. However, even when gaining a little more flexibility by working outside the sprint cycle, it is likely that the time for research will be shorter than in a non-Agile world, since the findings still need to be turned around quickly enough to be used to inform the release cycle. Less time does not mean that the quality of the research needs to be compromised. To work with shorter deadlines, and in keeping with the lean spirit of Agile, the UX team needs to look for opportunities to reduce waste and streamline the process as much as possible. This might mean spending less time on the recruiting phase, working with more participants in a shorter time span, and producing analysis more quickly. There may be no need to change the actual methods or techniques but rather fit the entire event into a smaller time frame. Also consider which methods to use when trying to decide on how to handle this work. Both usability testing and customer interviews can be accommodated within the sprint cycle, if necessary. However, working with larger user groups or investigating broader topics is best saved for a separate effort. Contextual inquires, diary studies, and any kind of ethnographic research requires more time and attention and should not be attempted to be fit into sprint cycles.

When the designer and the researcher are the same person or people, much more out-of-the-box thinking is required to find opportunities and make time for research. This is a challenge because the UX person hase to be heavily involved in the day-to-day operations and rhythm of the sprint cycle as well as produce the designs and accommodate user research when it can fit. If the heavy lifting is done by a researcher or consultant, then making it a part of the cycle without overloading an already maxed out staff member can be a little easier. While the designers have to be involved in sprints, because they are responsible for the production of the user experience for the product, researchers or consultants can work more independently because they are provide information that the designer or product team then feeds back into the cycle. Dedicated researchers still need to move more quickly to get their analysis

into the process in a timely manner, but it will likely be easier for them to carve out time to focus on more significant user research.

Tactics

- **Right method, right time.** Discount usability methods that require less effort can fit easily within sprint cycles. Surveys, web analytics, unmoderated remote tests, and unmoderated online card sorts can happen without much investment or effort on the part of the UX staff. Usability testing, customer interviews, mental model development, moderate card sorting, and RITE sessions require more dedicated time and may need an entire sprint to support but can be incorporated with a just little more effort. Contextual inquiries, ethnographic research, broader usability test, and customer interviews are challenging to handle in a sprint, unless Sprint 0 is long enough to accommodate the work. Otherwise, these are probably best done outside the sprint cycle and managed as an independent piece of work.
- **Plan ahead to execute successfully.** Of course, we always try to plan our time; but with much less slack in the Agile schedule, it is critical to have a good, realizable plan in place. It might be easier to be more flexible when it comes to in-cycle usability testing, but larger research efforts require careful forethought and execution. A single, central research resource is one way to have someone doing the planning and the investigative work outside of the Agile production cycle. This allows the team to worry about syncing up the timing of only that resource's efforts with sprint planning activities. If no central resource is available, as is often the case, research work needs to be coordinated so that it occurs during lulls in the production cycle. Since the planning cycle, if there is one, might be too short to do the necessary research, it might be helpful to use the endgame stage of the current cycle to begin research efforts. There is likely to be less of a need for design or usability testing in the final sprint(s), so that might be the best opportunity to focus on research in support of future work. It that does not allow enough time, and it very well may not, then it might be necessary to carve out more time to do research. If it is too much of a challenge to set large pieces of time aside for research, consider setting smaller bits of time aside throughout the release cycle as you work on day-to-day design and testing efforts. Consider dedicating a set amount of time each week to ongoing research and consider that a separate project you are supporting.
- **Target user stories.** When any kind of research is done, it should be used to inform and define the user stories. This is one of the most powerful ways to get the user feedback into the cycle and influence the direction of the product. Since the user stories are the basic building blocks of the release, this is your best opportunity to affect the entire team by describing the work

that they will be doing in a way that accurately reflects the user's needs. Influencing the language used in and the content of these stories influences the entire direction of the release.

Usability testing reports

Regardless of how user research is integrated into the process, it is highly unlikely that the research will culminate in the generation of any kind of formal report. The purpose of a report in more traditional methods is to serve as a record of what was discovered. It can also be used to stamp the findings as being "official," because they appeared in a document of record (Figure 2.3). It also serves as a mechanism for communication, although this is often the least of the motivations for a formal report, since the key findings are often distilled into a presentation to communicate them to a broad audience that may not be motivated enough to wade through a more structured document. In an Agile environment, any kind of formal documentation is meant to be replaced by communication and human interaction. There is no need to record findings simply for the sake of posterity on behalf of the project team, since the findings and necessary action items are immediately discussed by the team and their disposition resolved. In the case of frequently recurring testing, gathering the data and feeding it back into the product is a seamless activity that requires no documentation beyond the backlog items created as a result. Since sprint tests tend to be fairly narrow and focused in their scope, conducted with a small number of users, and qualitative in their analysis, analyzing the results and presenting them to the team should require little time. Ideally, representatives from each functional area observes the test sessions and the test team debriefings over the findings at the end of the day or on completion of the sessions. They could then bring back the key findings to the team and work together to identify feasible solutions and prioritize these efforts to fit within the sprint schedule and defer items that cannot be addressed in the time available. In the real world, the UX person might be on his or her own participating in the sessions, since other functional areas may not be able to find the bandwidth to attend all (or any) of the sessions. Even without full team

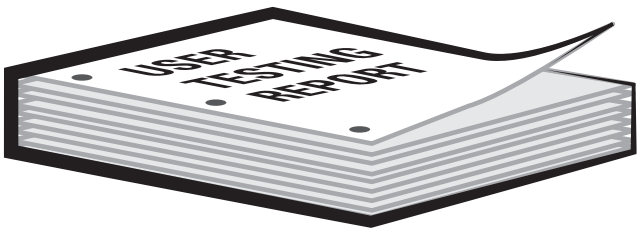

FIGURE 2.3
Usability testing reports.

participation in the sessions, the group can still follow the same workflow; it just might require a little more context given to describing the results. Stopping to create a deliverable around the testing results could interrupt the immediacy of the activity. The speed of the testing and quick delivery of results to the team should not mean sloppy research. While it is okay to test whatever is available at the time of the testing, the sessions should still have clear goals and objectives and the feedback from the users must be analyzed and presented in such a way that it can be translated into actionable tasks. While the research may be a little more ad hoc and evolve a bit organically, it is still research and needs to be treated as such.

It is also important to consider what will happen with the results when the testing is done. Producing a set of slides or a high-level summary report, even if they are fairly lightweight, serve sas communication tools but does not guarantee that the information will influence the design. It might be a great way to communicate the findings to remote team members to make them feel more involved with research activities or as a method of distributing information to internal stakeholders, but a simple presentation is not really a consumable format for the project team. Bear in mind that any document created should be done with the intention of supporting communication and distributing information rather than serving as a report of record. Since the broader product team will not necessarily be a part of the research effort but will be affected by the findings, it is important to share the testing process with the team so that it can appreciate the value of the feedback and become more engaged in a user-centered design process. If this means writing key findings on a whiteboard and working together to write out user stories or create personas, that kind of processing of the results is much more powerful and "Agile" than posting a 20-page document online and hoping that team members read it. The intention of this is not to minimize the importance of user research but to encourage a new way of thinking about how to more effectively work those findings back into the process and influence the product. Instead of leaning on the old way of working up a set of slides or producing a document, we should be thinking of how to make the feedback more tangible and the delivery something that results in concrete actions. At a minimum, the information should be presented to the product team in a meeting that allows for questions and not through an email informing the team members that the results have been posted. If the researcher is a separate resource than the UX person supporting the team, then these two people need to work together to translate the results into user stories, UX tasks, and preliminary sketches to put the information into a format that the Agile process can consume. The only time when a more formal piece of documentation or reporting should be produced is if a larger research effort is occurring outside of the sprint cycle and the intended audience is broader that a single project team.

Tactics

- **Create personas.** Personas are a great way to feed complex information back to the team, especially if it is the result of larger research efforts and poised to drive the direction of the release. If you used some of the usability test sessions to flesh out or validate personas drafted earlier in the cycle, update them, and recirculate the personas. Considering using the end of sprint demonstration for delivering the updates or turn the delivery of the updates into an activity for the team. If you uncovered new personas during the course of testing, write them up and share them. Personas are very helpful in reminding the team for whom you are designing and making the users less theoretical and more real. If the word *persona* is too loaded of a term to use in your organization, call them *profiles*, *customer snapshots*, or whatever helps get them accepted into the culture. Do not just leave them as something that is discussed at the beginning of the process; find ways to weave them throughout the release to make them most effective. Have the user stories reference them, put them up on the walls, incorporate them into the sketches, frame research findings in terms of the personas, and do what you need to make them a living, breathing part of the development process.
- **Inform user stories.** Yes, user stories again. They are so central to the Scrum process that any opportunity to create or contribute to user stories should be used to its fullest advantage. Leveraging this mechanism to also communicate findings eliminates a barrier to getting the research results back into the process. Since user stories are the basis of the development effort, translating feedback into this format makes it that much easier to settle on priority, identify related tasks, and assign the work to a sprint.
- **Present findings in person.** It is always good to give the team an overview of how the testing went, and the act of creating or discussing the resulting user stories serves to drill down into more specific issues. When team members are unable to attend the test sessions, an interactive meeting gives them a chance to hear the big picture of what happened during the sessions and a sense of the general reaction (both the positives and the negatives). It would be great to also have a separate brainstorming session or design discussion to resolve the issues that came up, but delivering an overall report verbally to the team should be the minimum done following the testing. Delivering the results verbally removes the barrier to common understanding that can sometimes be created by a written document, which can be interpreted differently by each reader. While listening to a presentation on research findings is no substitute for attending the actual sessions, it is the next best thing for teams that are sure to be very strapped for time.

Design activities

In some organizations, the design is "owned" by the user experience staff. However, in Agile, only the product owner role is recognized as having any kind of ownership and ultimate authority, while the rest of the team members are contributors. The team is intended to be empowered to make its own decisions, so the product owner should need to exercise power only on the rare occasions when the team cannot come to an agreement without intervention. It is also important to remember that team members are meant to work to their skills and not their titles. In some cases, this might mean a team member who does not have a UX title might execute some design work. The idea of ceding territory might be frightening, but it also has potential for increasing the general awareness of design issues across the team. Design divas or UX rock stars will prefer to capture the spotlight and the glory by swooping in with a genius design solution, and they feel very uncomfortable with this idea. There is no room in Agile for people that are not interested in working as a team, so if you are struggling with this concept, then you might want to consider whether you are concerned with moving the product forward or about not being recognized for your genius. When small design tasks are supported by people with good design skills, regardless of their functional area, then the user experience staff can be freed up to focus on the larger design issues and user research. Schedules tend to be fairly tight on Agile projects, so a given UX person is unlikely to be able to address every detail of everything that appears in the user interface. Being open to allowing other team members to contribute in a substantive way to the design can improve the overall design and result in a better product. It also gives the broader team a greater sense of ownership of the design and acknowledges the contributions that they make to the user experience.

The idea of pair programming that evolved from Agile methods can also be co-opted to pair a designer with a specific development person as a partner. This kind of real-time collaboration can speed up the process of design and implementation considerably, just as pairing two coders increases the quality of the resulting code and the time it takes to produce it. Leis Reichelt explores the application of this technique in "Pair Design Pays Dividends" (2006), in which she discusses her experience with this method and the value that it added to her work as a consultant (Reichelts's article can be found online at uxmag.com). It may not be necessary to engage in a fully paired approach to reap the rewards of this technique, but having a specific development person responsible for large portions of the UI and working in concert with the UX designer could be a very transformative experience for everyone. This may not be a good fit for every project; especially if multiple people work on the UI elements, pairing closely with one person might be counterproductive.

In addition to relying on the team members to be collaborators, using customers as design partners is well supported by the Agile values and principles, which encourage working with customers collaboratively. Leveraging the research sessions to actively work out design issues with the customers using techniques like RITE, card sorting, brainstorming, cognitive walkthroughs, in addition to usability testing can provide the users with a chance to directly influence the design process and express their needs. This might also be your chance to explore participatory design methods, if you have not done so in the past. Participatory design encourages customers to completely immerse themselves in the current context, imagine an ideal future, and create a representation of the ideal. A great introduction to the technique can be found online at boxesandarrows.com in "Making Emotional Connections through Participatory Design," written by Marty Gage and Preetham Kolari (2002). Participatory design can yield really interesting results and is great technique to get the customers to think in terms of their experiences and needs and not be constrained by their understanding or expectations of the software. Another benefit to this technique is that it requires no working software and so can easily be done at the beginning of a cycle to inspire the design.

The goal for any design activity in Agile is to identify the best solution as quickly as possible and refine it so that it meets the customer's needs and creates the best user experience possible. The most efficient way to do that is to involve as many internal and external stakeholders as possible in the design process, so they can contribute their unique perspectives and help move the design forward. This should be done without creating a chaotic environment or a design by committee mentality, however. Using techniques that allow this the collaboration to occur within a specific framework, so that the process does not spin out of control, can help to keep the conversations on track and moving forward in a productive way.

Tactics

- **Teammates are codesigners.** This can be made explicit by using pair designing and collaborative design techniques or assigning certain design tasks to capable (and interested) teammates. It can also be more implicit by simply involving the team more frequently in conversations about the user stories. It is also powerful to solicit team members' opinions in design reviews, especially if you show that you are genuinely listening to their ideas and incorporate the best comments into the design.
- **Customers are codesigners.** In addition to user research activities, consider working some of the participatory design methods into your repertoire, especially if you are having regularly occurring customer sessions. Getting their feedback is great; getting their design perspective can be even more powerful. Use one of the techniques meant to support this, since unless

your customers are software designers or UX professionals, they are unlikely to sit in front of a blank piece of paper and generate anything terribly meaningful.

CASE STUDY—CATHERINE ROBSON, SEACHANGE INTERNATIONAL

While there is often trepidation around the change from traditional to Agile methods, the reality is that it can actually be very beneficial to the UX team. Not only is it an opportunity to be in control of defining the team's process, possibly for the first time, it is an opportunity to become a more cohesive team. Catherine Robson's story is that of a team that not only excelled in an Agile environment but has been innovative in their approach. The result is that they are able to be very effective and now have a way of proving their value to the organization.

Catherine is the manager of user experience at Seachange International, a software company that specializes in multi-screen video. She was a part of the original adoption of Agile at the company. Overall, the company weathered the transition fairly easily but the UX team had to change its approach along the way to better support the production efforts.

A few years ago at Catherine's company, a team on a separate code base was working on a forward-thinking project and picked Agile as their process. The team was successful in its project, and the use of Agile was credited with its ability to react quickly to changes and improved communication. As a result, the director of engineering chose to roll the methodology out to the entire organization. As a part of the new initiative, the managers were sent to Agile and scrum master training. Additionally, the initial team members that piloted the process were spread across the company to other teams to distribute their knowledge. At the time of the interview, they were two years in to their adoption and Catherine is really enjoying it. She feels it has created a more organized environment for UX. Historically, team member had their heads down and were focused on their individual projects, but Agile has allowed them to become a closer team. It is interesting to hear her make this comment, because many UX teams feel that the pace of Agile increases their individual focus and reduces the sense of UX community. The fact that Catherine's team had the opposite experience is a sign that the UX efforts to become Agile were done with great forethought and in a way that created a more cohesive team.

Catherine and her team were asked to guide how UX would fit in to the process. One of the pilot team members was a UX person, and she was able to share her experience. This helped the team as they tried to sort out how to come up with a generalized approach. Catherine did some research. She invited folks from IBM to share their experiences with the team. She also attended some user interface engineering seminars, which really informed her opinion of how the sprint process should work. Additionally, she participated in Scrum training and found it helpful to understanding the point of the Scrum methodology and where it was okay to bend the rules. This type of training and the confidence can provide is invaluable for someone just starting out with Agile. You have to make many adjustments along the wa, and it is important to know when you are making an adaptation and when you are compromising the Agile method that you support. For her, the point of Agile is to *be* Agile. She suggests being comfortable with the idea that, if the first sprint does not work, we will fix it; it does not have to be perfect the first time around. Refining the process is fine and healthy, rigidly sticking to something that is not working is not.

Retrospectives were uncomfortable at first. Seeing change happen as a result of the meeting—even just a small change—built up confidence in the process and made it more comfortable. It is really important that the team members see that their feedback is being taken seriously. Sometimes, being responsive to the small things is the most important action, because this builds the trust and creates an environment where the team feels safe raising bigger issues. She feels that, if you can get the team to buy into the idea that you are creating your own environment and are not just mindlessly implementing a process, you can be successful. The executives may have mandated that the organization go Agile, there may have been formal training activities, but ultimately the team members decide what Agile means for them, and the retrospectives are the main vehicle for this. She cautions that, if you have a scrum master who is closed off, then this really shuts down the discussion. This can cripple the process and inhibit growth.

At first, the adoption of Agile was messy. Sprints were three weeks long and each project team had its own starting point. The UX team did not have its own sprints, and its work was not centralized. One UX person might have two Agile projects to support, and this was hard for everyone. Engineering did not want to do story-point estimations on the UX stories, and the UX people did not want to sit through detailed architecture discussions. They changed things up so that the UX team had its own separate schedule. Catherine gathered requirements from each team. She is the UX product owner and scrum master. All UX requests are in one backlog. This is very helpful for load balancing, and she was much better able to resource the requests based on skill sets and availability. This is one of the changes that helped contribute to a stronger sense of team for the UX people.

User testing is done as a story that occurs within a given sprint. When the product is ready for tech support and customers, the UX team solicits feedback on it. Two people work on the story within a sprint and do the planning, testing, writing the report, and meeting with the team to discuss the results. Then, they negotiate to get any additional stories resulting from the testing into the backlog. They also work to get informal feedback as often as possible, but those activities do not necessarily require an entire sprint dedicated to them.

They use Rally for tracking tasks and monitoring work. They schedule the UX sprints a couple of months ahead of time. They make changes along the way, but work off a soft plan. They assign UX stories as a requirement for the engineering story, so development depends on the UX work being done first. Timing of the UX work is based on the complexity of the design effort, the task dependencies, or where it is planned for implementation in the engineering schedule.

The most important thing in moving to an Agile environment is a focus on communication. There are improvements on this front within the project teams, mostly supported by the daily standups. Having an opportunity to discuss problems daily and not wait until something blows up to raise an issue has made it easier to manage issues. However, since Scrum focuses on the communication within a team and not across teams, projects, or products, there is still a struggle to improve the sharing of information more broadly.

A UX person who supports multiple teams still invests in meeting time, because it fosters discussion that might not otherwise happen or might not include the UX person. It is expected that not every piece of information discussed is relevant to the UX team, but being present ensures that if something comes up the UX team is there to speak to the issue. The presence at these meetings also signals the commitment to the team and an interest in being fully engaged in the process. The team has four UX people spread across 5—10 big projects. Each UX person takes on two projects per sprint, at the most. If the people try to take on more than that, it is not possible to support all the meetings. They work together as a team to review each other's work. This increases efficiency by introducing multiple perspectives, increasing awareness of other projects, and making it easier for team members to swap projects, if necessary. The team has also made an investment in spending a lot of time speaking to engineering management and product management to see where the UX team should spend its time. This allows the members to be as strategic as possible in their efforts and to make sure that these efforts are in line with what the other teams consider to be a high priority.

The team's deliverables are the Axure prototypes. It is working toward a unified UI and has a prototype that shows all the teams' contributions to the UI. It uses Wiki pages (Confluence) to post screenshots with a brief description to provide additional detail on things that might not be clear. It is definitely not a specification. The pages are there more to support and complement the existing team interactions and discussion and provide only what is necessary for clarity. The QA/QE (Quality Assurance/Quality Engineering) is integrated enough that the team can rely on Wiki and know who to ask if it needs any additional information. It is harder for the documentation team, but the UX team mitigates that by meeting with documentation team members and walking them through the prototype. Since this tends to come late in the cycle, it is easy for the UX team to accommodate the other team's members because it is not as entrenched in design activities. This is allows the UX team to produce only what is necessary; an interactive prototype that not only illustrates design intention but also promotes consistency across applications. For the team members who need more information because they are not as heavily involved in the design cycle, the team does not produce a more heavyweight document but meets face-to-face with them to help bring them up to speed. This shows that the team is not just doing things the way it used to do them but is trying to be as Agile as possible.

The UX team is located in Acton, Massachusetts, but many of the functional areas are around the world. A development team is in the Shanghai office, and for that group,

it does more documentation than it might for a team that is colocated. For this group, the UX team creates Illustrator pictures, fully detailed, in their design documents. For the teams in Manila and Europe, it tends to communicate in the same way as with local teams. This reduces some of the burden of working with geographical distributed teams, since only some of them require more written communication than is typical.

The hardest part of the transition was figuring out the time management at the beginning. Once you have experienced everything, you can better identify what is valuable and what you can trade off. But, you have to go through the initial period first to figure that out. This is very true for teams that have been working in waterfall and cannot really wrap their minds around what it means to work in a three-week sprint. You can plan for how you might need to adjust to accommodate the new work flow, but until you have actually done it, you do not know exactly where the pain points or the easy parts lie.

The most important piece of advice that she would pass along to other UX practitioners is to get the company bought in to having a UX backlog. It is a way of positioning the UX team and establishing that teams come to UX for help and the UX team provides value. It was possible to make that statement and prove it, because the team established its value and can use the product backlog to prove it. She acknowledges that this tactic might not work in every organization or for every team. Still, she finds having a backlog of requests useful to prove that engineering has a need that the UX can satisfy. There have been management changeovers during the past two years, but having a body of evidence of what the UX team has done and what the engineering teams would like it to do helped prove the worth of the team.

In this story, we can see a UX team that not only thrived in an Agile environment but even benefited as a team from making the transition. The move to a new process allowed the UX team to become a more tightly knit unit and work in a more organized and efficient way. It also inspired Catherine to build UX backlog that not only organizes the efforts of the team and assists in load balancing but provides high-level visibility of the UX effort and a clear record of its value add to the product.

Key points

- A culture of open and honest communication can grow along the way. People who are initially inhibited during retrospectives can work through the problem if they see that feedback is well received and taken seriously. As long as somebody is modeling that behavior, change will occur. The more people who model it, especially if they are in leadership positions, the faster that change will occur.
- Showing up to meetings is a drain on time, but it can be worth it. Often, the discussions may not directly bear on the design, but showing up and being present goes a long way to reassuring the team that, no matter how busy or hectic things are, you are going to be an active and collaborative team member. If this means supporting fewer projects to be able to support these events, then that might be the right trade-off to make.
- Consider a common UX backlog of tasks. This is not a common tactic, but it can provide very clear evidence of where and how the UX team adds value without trying to tackle complicated metrics. It will require that the team is somewhat centralized and that one person is responsible for the distribution of work, but it can provide benefits beyond simply dovetailing into the Agile process.

SUMMARY

These recommendations and modifications to UX techniques assume that an organization or team has fully embraced the spirit and values of Agile and the whole process has been undertaken in a thoughtful and supportive manner. If this is not the case, these activities are still worthwhile but may meet with less support or enthusiasm than if the organization had genuinely adopted Agile

values. For those situations, your priority might be on simply fitting your work into the time it has been given rather than doing so in a way that also embodies Agile values. This does not mean that you should not try to be as Agile as possible, but you should be realistic in the expectation of how much traction that your Agile UX efforts will have. However, do what you can to stay on the Agile path, because you never know when the culture might change to be more receptive. In the next chapter, I share some case studies of teams that fall on both ends of the spectrum and one or two that land somewhere in the middle.

References

Gage, M., Kolari, P., 2002, March 11. Making emotional connections through participatory design. In: Boxes and Arrows: The Design behind the Design Retrieved April 2, 2012, from. http://www.boxesandarrows.com/view//making_emotional_connections_through_participatory_design.

Medlock, M.C., Wixon, D., Terrano, M., Romero, R., Fulton, B., n.d. Using the RITE method to improve products: A definition and a case study (Playtest Research). Retrieved April 2, 2012, from http://www.microsoft.com/en-us/download/details.aspx?id=20940.

Miller, L., n.d. Case study of customer input for a successful product. Agile Product Design. Retrieved April 2, 2012, from http://agileproductdesign.com/useful_papers/miller_customer_input_in_agile_projects.pdf.

Patton, J., n.d. Getting software RITE. Agile Product Design. Retrieved April 2, 2012, from www.agileproductdesign.com/writing/ieee/patton_getting_software_rite.pdf.

Reichelt, L., 2006, November 20. Pair design pays dividends UX Magazine: Defining and Informing the Complex Field of User Experience (UX) Retrieved April 2, 2012, from. http://uxmag.com/articles/pair-design-pays-dividends.

Soucy, K., n.d. Unmoderated, remote usability testing: Good or evil? UXmatters: Insights and Inspiration for the User Experience Community. Retrieved April 2, 2012, from www.uxmatters.com/mt/archives/2010/01/unmoderated-remote-usability-testing-good-or-evil.php.

User Experience: Our Definition, n.d. Retrieved April 2, 2012, from www.nielsennormangroup.com/about/userexperience.html.

Young, I., 2008. Mental Models: Aligning Design Strategy with Human Behavior. Rosenfeld Media, Brooklyn, NY.