CHAPTER 6

Using Agile Concepts for UX Teams

CONTENTS

INTRODUCTION

User Experience teams can borrow Agile practices, even if they support projects and development teams that do not use Agile methods. A process focused on iteration, integrating user feedback and customer needs has something to offer any UX team. Most UX teams do not even consider how Agile could be integrated into the user-centered design process until they learn their organization or team adopting an Agile process. If your team is part of a more-traditional process, you might think that there is no point in figuring how to practice Agile UX, because it cannot be possible to do so if the development team is not doing the same. However, the core values and principles of Agile are very well suited to any team practicing user-centered design. If you are looking to evolve the practice of your

Agile User Experience Design. http://dx.doi.org/10.1016/B978-0-12-415953-2.00006-6

team or just your own practice, plenty of inspiration is to be found in Agile practices and you need not wait for your development teams to lead the way.

CREATING A USER EXPERIENCE BACKLOG

In most Agile environments, the backlog of user stories and tasks is local to the project team. However, after hearing Catherine Robson's description of having a centralized UX backlog, clearly, this idea could be helpful for many UX teams regardless of the development methodology. One issue UX teams face time and again is the return on investment for the design effort, and it is often necessary to prove to management that the design team adds value to the production process. Even if the UX team is in an organization where these requests are not explicitly made, this does not mean the issue will not come up in the future. After all, with changes in leadership, the new executives have no prior experience with UX or your team, so it can be very helpful to be able to illustrate the contributions of the UX team and mitigate the need for conversations around the value of UX. While some groups can come up with benchmarks and usability testing metrics that show the effect of their efforts, most are unable to show such concrete measurements and oftenare at a disadvantage when trying to justify head count or budget.

Having a process by which requests for UX support are created by the development team and submitted to some sort system adds an element of thoughtfulness to the act of seeking UX support. Instead of just wandering in to a UX person's cube and suggesting he or she look at something or viewing the UX person as an on-call resource whose tasks require no advance planning, having to formally request the UX person's times gives the sense that the UX work requires time and effort and needs to be planned for. That is not to say that teams that do not request UX support do not value the work, but when the client teams have to put the requests in to writing and think about what they need, it imparts a certain sense of worth to the work. Having the tasks come to UX in writing from the team also provides some clarity about what kind of help sought and an opportunity to discuss what activities are most appropriate or beneficial. Many times, a team thinks it needs one thing but could really benefit more from a different activity, which can turn the discussion around the request into a powerful teaching moment.

Having a backlog of requests can not only help a UX team balance its load balance more effectively, it can be a valuable artifact to illustrate both the supply and demand for the team. The UX group can also gain insights into where the members spend most of their time and determine if they need to adjust. The team need not be centralized for this to be effective. Even if designers are dedicated to specific projects or products, the backlog can be structured to support individual efforts but roll up into a summary that gives an overview of the entire group. When UX teams are not centralized, having such

a reporting mechanism can also identify potential efficiencies that might be realized. Regardless of the UX team organization, having a backlog can increase visibility, awareness, and understanding of what the team does.

RECURRING USER TESTING

In a waterfall cycle, it is easy to schedule user testing to occur after a specific milestone. That milestone often slips out of the schedule, taking the usability testing with it and possibly pushing it to the point where customer feedback cannot be incorporated into the shipping release. It can be even easier to continually put off scheduling user testing because the product is not ready or there is so much design work to do and specifications to write that it seems like there is not enough time to run the sessions. The intentions are good—wanting to wait until the design is "done enough" before showing it to users, the desire for a sufficiently robust prototype, or resolving bandwidth issues. We know that the more "done" a product is, the less likely the user feedback is to be incorporated in to the product, but there is often internal pressure not to show customers a rough prototype that may give the wrong impression. Looking at Agile teams that embrace a "test what you have" mentality can inspire you to find some ways to get around this pressure. Instead of trying to show a prototype that is so rough the feedback may not be as meaningful as it could, show the customers sketches or wireframes or have a conversation about workflow. None of these concepts is unique to Agile, but seeing how user research is incorporated into such tight timeframes can certainly undermine any excuses not to do research in a release cycle that is a year or longer in duration. If someone with a three-week or three-month release cycle can make time for weekly feedback sessions, then surely someone with more time can do so as well.

Agile encourages frequent user feedback sessions and the ideal state is to have a predictable, recurring event happen weekly, biweekly, or monthly. There is no reason not to adopt this in a waterfall cycle. It might mean testing with rough digital or paper sketches, but the same holds true in Agile environments. It may even mean to have nothing to show the customer but simply use the opportunity to do collaborative design or engage in a conversation for feedback. The sprint cycle provides an incentive for testing, because there is an opportunity to get the customer issues into the backlog and influence the direction of the next sprint. There is no reason the outcome of a testing event in a non-Agile environment could not simply influence the next week of design or implementation.

BREAKING THE WORK IN TO SMALLER PIECES

A problem with more traditional methods is that they deal with very large pieces of work, such as "create a mobile application for product *x*" and ask only

that you get it done by a given point in time. Not breaking the work down into smaller units makes it harder to track progress against the goal. For a designer, it can also make for an overwhelming task. Identifying the pieces that make up the whole and tackling those incrementally allows for a more focused effort. It also supports the designer's ability to seek feedback throughout the cycle. It is clear that the designer who focuses on the entire mobile app as one task and waits to have a design review until the whole thing is designed will take considerably longer than one who narrows the first review session to address the high-level architecture and a few key screens.

CONSTANT FEEDBACK AND ITERATION

The motivation to have so many user feedback sessions comes from the Agile intention to keep the customer needs at the forefront of the team's consciousness, but it is also encourages a more accepting approach to reworking and refining the product in a direction that will create the best user experience. Since refactoring is an accepted and even encouraged concept in Agile, the team is mentally prepared to refine the product and revisit different aspects of the design throughout the release cycle.

If you are in a more traditional environment, the values around keeping the customer in mind and refactoring along the way may not be as dominant. However, the earlier in the cycle design changes are proposed, the less expensive they are and the more likely the changes will make it into the product. Regardless of whether you are validating designs with usability testing or having design review sessions with the team, you should solicit feedback as often as possible. Waiting until you have handed of a huge design specifications makes you more reluctant or able to make corrections. This does not mean that you should consider assaulting your team with weekly change requests. Instead, use regularly occurring sessions with internal and external stakeholders to guide and refine your designs before you hand them off to the development team. Doing this creates a stronger design that requires less rework for everyone and can free you up to focus on more valuable design validation later in the cycle. as you have managed to resolve the most glaring issues.

RECURRING EVENTS AND RITUALS

Some of the value in the scrum events is that they are predictable and repeatable. It is more of a break in the rhythm of the team to meet occasionally than to meet briefly every day. Similarly, scheduling a weekly customer feedback session can make it more natural to interact with customers throughout the cycle, and working their comments into the design becomes second nature. Take inspiration from this tactic and consider scheduling routine design

reviews with your product team. Keep the meetings short, efficient, and relevant, since attendance is not mandatory or part of the official process. If you can keep the discussion interesting and everyone knows that there will be a design session at lunchtime every other Wednesday, it might just become a habit for the team to attend to see what you have to share and work collaboratively with you to improve the design.

NO DESIGN DIVAS OR HEROES

It is always fun to be the genius designer who swoops in with a fantastic solution at the last minute that makes the design amazing. The reality is that depending on one star performer to always independently create the best design solution is not very healthy. It is not collaborative and it creates a bottleneck if one (or two) people are the only ones allowed or expected to generate designs. Many people have gotten used to being in this role and enjoy it, but it is a good idea to move away from encouraging this kind of culture. Someone who is very attached to that dynamic will not last long in Agile environment, but that person should not be allowed to engage in such behavior in any environment. Part of valuing the individual and his or her skills in Agile means recognizing each person's unique contributions and skills, regardless of title. While the UX person is responsible for the design, that is not the same as "owning" it. It means that the person is the shepherd who guides the design to completion and is on the hook to do so. It does not mean that the UX person is the only one who can, or should, contribute to the design. Not only will team members have better insight into feasible implementation solutions, they often have domain expertise that the designer may lack.

This does not mean that design should be a chaotic activity, where everyone's design suggestion needs to find its way into the design or the team must vote on design solutions. However, team members should be viewed as design partners and contributors to work as collaboratively as possible. Working collaboratively also allows the designers to come up with ideas they might never have identified on their own, while discarding truly infeasible ideas quickly before investing too much time and energy in them. Although Agile values support this, the principles are equally valid in more traditional environments. Just because they work in a waterfall environment does not mean that development resources cannot contribute to the design in a meaningful way. Divalike behaviors are based on the assumption that the solution could not be made better with additional input, and it is hard to believe that this is true, regardless of what development process is in place.

Additionally, a good designer teaches design skills, in a "teach a man to fish" strategy. Not only does this enable more people to recognize and implement

a better user experience, it allows the designers to focus on areas that require their attention instead of trying to be everywhere at once. When everyone on the team has a working understanding of good design principles, the resulting user experience is improved. The UX person cannot define every widget and label, especially in an Agile environment; and many, many implementation decisions are made by the developers as a matter of course and necessity. This happens just as frequently in more traditional environments, but when the finest detail is written in a specification document, there is an illusion that the outcome will match the specification, which is very rarely the case. This is why empowering all the team members to wear the hat of "designer" while giving them the tools to do so effectively makes the UX staff infinitely more effective.

FOCUSING ON COMMUNICATION OVER DOCUMENTATION

If your software development life cycle requires epic novel-length specifications, you will not be able to get them off the team's plate, but you can consider how to support a more effective way to share its content. If there is no hard requirement for such deliverables, consider examining your UX deliverables to see if they are geared toward supporting communication or simply documenting your intentions. Relieving your team of the burden of producing something that does not serve its purpose might be worthwhile. If the designers focus on engaging in collaboration with other functional areas and achieving common understanding through conversations, there is less to communicate in writing. In many cases, a simple sketch that reminds the different team members of the general design and direction might be enough. For functional areas that are less involved in day-to-day design activities or tend to come into the process later in the cycle, having a meeting to review the design might prove more effective, and quicker, than posting a long document.

The reality is that, even when long documents are required by the process, not everyone who should read them does so. Just as often, team members read an earlier version as part of a review process and never take the time to sort through the changes made with the final draft. If your team is in the business of writing such documents and your process does not allow for a different way of communicating the design, take a cue from Agile and think about additional delivery methods, such as a walkthrough of the design or a review and discussion of the critical design elements. This provides the other functional areas with a much richer opportunity to understand what the design is trying to accomplish and gives them a chance to ask questions and learn more about what the design is doing. This significantly increases the chances of the key elements of the design being implemented correctly.

THINKING AND COMMUNICATING IN TERMS OF USER STORIES

UX practitioners are accustomed to thinking in terms of personas, either formally developed and robust personas based on user research or more informal constructs that live in a designer's head based on customer interactions. Personas are valuable when thinking about intended audience and segmenting different user types and provide excellent guidance when working through the high level design. However, even personas can get a little abstract for people from all functional areas when they start to work on the detailed design. For some people, it is a challenge to move from the description of the person and his or her needs and apply that to a single interaction or set of controls. Thinking in terms of "What would 'Kevin' want?" still leaves plenty of room for interpretation on what exactly "Kevin" would want from a given element of the design. User stories can make a wonderful complement to personas, if you are already using them, or can be a technique that can change the language of the discussion. User stories are framed in terms of customer use instead of product requirements, which can be helpful as well. Despite anyone's best efforts, if the work is constantly defined by and spoken about in terms of features, the conversation can be constrained. As UX professionals, we always try to stay focused on the user, but when the dialog is always centered on the presupposed solution and not the problem, this can be a challenge.

DEFINING ACCEPTANCE CRITERIA

Many organizations have implicit or explicit release criteria that serve as gatekeepers for releasing a product. This can be quality criteria, performance criteria, or simply sign-off criteria. Depending on how seriously these criteria are taken by the organization, they can be effective at preventing products from shipping with major flaws. More local quality criteria might be applied to features or code submissions, and those can be very helpful in improving the overall health and robustness of the product. Release criteria, and even local quality criteria, are necessarily generalized and intended to enforce a certain set of standards across products.

The Agile concept of "acceptance criteria" can be helpful in any environment. The idea behind the acceptance criteria is that the team agrees on what needs to be a part of that user story in order for it to ship. For example, if the user story is, "As a student traveler, I want to book a cheap flight to Amsterdam," then the accompanying acceptance criteria might be "Show available flights to Amsterdam," "Allow the arrival and departure dates to be changed," and "Highlight low fares." The agreement is established ahead of the stress and the heat of an impending deadline, where the natural tendency is to ship what you have done and there can be pressure to compromise to support that goal. These

criteria are tailored to the specific user story, with the unique goal of making sure that the user story is implemented in a way that satisfies the user's needs. This specificity gives these conditions greater impact on a specific feature and can drive feature completeness and a broader clarity about what the user story means. Without the framework of an Agile process and its many tools to support the creation of these items, it can be a little more difficult to work these into a more traditional process. If you choose to employee use stories as a technique, then discussing and agreeing on the acceptance criteria can be a natural part of the story creation. If that seems like it might require too much effort or you did not choose to use user stories as a method of communication, there is a still an opportunity to apply this technique. It works best to capture the acceptance criteria in writing, to make it easier to refer to and remember what the actual agreement was. The criteria can be stored in something as simple a publicly available Excel spreadsheet or attached somehow to an existing artifact in your process.

USING LESS UP-FRONT DESIGN

Before you panic, even in an Agile environment, "less up-front design" is not the same as "*no* up-front design." It really means that the design should not, and really cannot, be fully designed before being turned over to development. There is no handoff period in Agile, and things move so quickly you cannot do a complete design ahead of the implementation work, even if you want to try. If you are in a more traditional environment, you might wonder why you should even think about using this technique. After all, you have a lot of design work to do and really big specifications to write. If you think back to the last time there was a review of those specifications, someone probably caught an error or a design change needed to be made. It probably took you hours or days to update the sketches, screenshots, and wireframes and bring the document up to date. If, instead of laboring over the document alone in your cube, you conducted periodic reviews with the team and collaborated on a variety of design solutions and proceeded to document those efforts, you might save yourself a lot of time and effort. You might also find that you have not only produced a better design, but the implementation more closely matches the specification.

SUMMARY

It is not necessary to move an entire organization to an Agile development process to get some of the benefits from its values and tactics. Since there is so much overlap with Agile and user-centered design values, it is fairly easy for UX teams to borrow from Agile methods. Inspiration is to be found in treating the design as a deliverable that can be produced through a series of smaller design

efforts and incorporates continuous feedback. Specific elements can also be translated into UX activities by using the language of user stories to describe customer needs, having recurring events to increase communication, and creating a collaborative culture. These ideas can help create a more high-functioning team, even when still working with a traditional process, and the UX team should consider how it can take advantage of using these methods.