CHAPTER 5

Frequently Asked Questions

CONTENTS

INTRODUCTION

At the beginning, it can be very exciting to move into an Agile process. You did your homework, memorized the Agile Manifesto, gone to a few training sessions, and read everything ever written by Jeff Patton and Jeff Gothelf; you are pumped up and ready to go. As you start to do the real work of putting all

Agile User Experience Design. http://dx.doi.org/10.1016/B978-0-12-415953-2.00005-4

that theory into practice, you realize that you still have many, many questions. It is always a challenge to try to figure out how to fit your UX teams' efforts into what Agile looks like at your company. You may have come to realize that something about your team is less than ideal for a textbook application of the chosen Agile method, and you are not sure if you are making the right trade-offs or just compromising the method. As it turns out, every UX person faces this same struggle when beginning the Agile journey and trying to customize the philosophy to a specific situation. Here are some of the common questions that people have, and some solutions that might work for you.

SHOULD WE EVEN BE AGILE?

This is a valid, interesting question to ask, but the answer almost never belongs to the UX team. Realistically, most of us learn about the decision to switch to Agile once it has already been made, although a lucky few may actually be part of that decision-making process. However, since the UX team will be left to define how it fits into the chosen Agile method, thinking about this issue is worthwhile, so you can understand what you might be up against and where your challenges lie. Most projects can benefit from some adoption of Agile methods. In my most optimistic moments, I like to think that any type of project could be Agile, if an Agile method is applied in the right way and the project work was scoped appropriately. However, in a few varieties of projects, the team should proceed with extreme caution and know that it may face more difficulties than the average team.

The first type of project that requires extra care in the application of Agile is those large-scope projects that cannot easily be broken into smaller pieces. Large infrastructure projects often require a sufficiently large project team to make the constant levels of communication required by Agile unwieldy and potentially distracting. The conventional wisdom is that the best size for a scrum team is between five and nine members. Kanban teams seem to work most effectively at similar sizes. That is not to say that these methods cannot be used with a larger team, but at a certain point, maintaining the high level of communication, transparency, and collaboration that is the intention of engaging in Agile will be a challenge. No matter how cohesive a team is, how much pair programming occurs, or how aggressively colocated the team is, a team larger than 12 people will struggle with the process. Not only will the kind of overhead that the ceremonies of a process like Scrum entails seem to be a burden and not necessarily helpful for a large scale project, a Scrum-style standup meeting with 20 team members is completely infeasible. Now, if you are a part of a large team and there is a genuine commitment to going Agile, there may be some ways to mitigate the size. Have representatives from the functional areas rather than the whole group attend daily standups and make sure that everyone takes a turn in that role. Also, engage not only in pair programming, but pair designing and pair testing. Partnering within and between functional areas makes sure that at least a few members in the team are engaged in the kind of communication that occurs on a small Agile team.

Working incrementally on large-scale infrastructure efforts is difficult, since they are generally made up of very large pieces of work that need to fit together, rather than many small pieces that make up the whole. Defining the smaller units of work, such that they can be completed within a sprint, may require sprint lengths so long that the release cycle resembles staged development more than anything Agile. If your sprints are eight weeks long and there are only six of them, is that still Agile? The answer is "It depends" or "Kind of," although I am sure many people would be happy to shout "No!" It would certainly be a step forward in the Agile direction if the previous cycle was just one long 12- to 18-month stretch. However, it might be hard to have demonstrations of completed work or show evidence of progress and do many of the things typical of Agile. Some infrastructure projects cannot even be broken down to this level; and if that is the case, the decision makers should think about what they are trying to achieve in using an Agile methodology on that work. It might make more sense to take inspiration from some Agile techniques around collaboration and the tracking of tasks to realize something that is more efficient than waterfall but less than Agile.

It is also typical for these large-scale efforts to require significant architectural changes, which may not easily be broken into small pieces and may make for very awkward user stories. In Scrum terms, an architectural epic that cannot be broken in to sprint-sized tasks can derail the process. An infrastructure or architectural change that is not being done for any performance or customer-facing reason ise difficult to express in a user story. Obviously, work that is to clean up a code base, modernize the architecture, or any of the other reasons for engaging in infrastructure work has value. However, these reasons cannot always be translated into customer needs; and while I have seen user stories such as "As a system, I want to…," those are really not good Agile practice. This is not to say that these types of projects cannot be made to work in Agile but that the project and the team need to be defined and chunked to fit well into the rhythm and needs of the Agile cycle. Also, some of the out-of-the-box methods may not be a good fit for the work and the team will have to be very conscientious in its application of Agile. If you find yourself working too hard to make the fit happen, it is worth taking a step back and considering whether or not Agile is that right answer.

At a minimum, this kind of project would definitely not be a good pilot for an organization with no experience in Agile projects and really should be tackled (if at all) only by an experienced team. A middle ground might be more reasonable for a novice team looking to leverage Agile principles. When you add up the potential for trouble with team size and the type of work, making these kinds of projects fit into Agile clearly is such an uphill battle that the value in doing so is very questionable. Working in a phased approach (with three phases instead of one), creating transparency around the project, increasing communication, and empowering the team can all be incredibly powerful for a large project. While it is true that a mini-waterfall is still a waterfall, it is also true that an expensive and abject failure using Agile could cause a company to write it off indefinitely. Taking baby steps on projects that are not an ideal fit for Agile methods and embracing the values rather than trying to fully engage in an Agile process (or remain completely committed to waterfall) could be a good first step.

Agile can also be complicated for projects with upstream or downstream dependencies. In these cases, it can take longer to be "done" with a piece of work if the project team completes its task but it cannot truly be completed until that work is integrated with the work of another team. Additional challenges are introduced depending on whether the team is upstream or downstream of the dependencies. If the team is downstream and not in control of ground that can change beneath them, the rework can be unpredictable and difficult to manage in an Agile manner. As with the large-scope projects, it is certainly possible to use Agile values and processes partially or entirely. However, it is probably best to have some organizational experience and expertise with Agile before taking on such an effort.

HOW LONG SHOULD SPRINTS BE?

There is no magical answer to this question, unfortunately, but there seems to be a bit of a sweet spot around three weeks. In true Goldilocks fashion, three weeks is somehow not too long and not too short, although it would be hard to say that it is just right for everyone. This length cycle seems to allow for a good balance between getting tasks done, working collaboratively, and engaging in all the meetings that might be required for the sprint. For sprints that are two weeks or less, the ratio of work to ceremonies can often be a weighted a little too much in the direction of the ceremonies, and some of the efficiencies that could be gained with a longer sprint are lost. In determining the right length for a project, consider the team size and location, the amount of scheduled meetings, and the smallest average unit of work. If the smallest unit of work is typically around three weeks, then your team will likely want to start with four week sprints to allow for the majority of the tasks to fit in, with some cushion for meetings and collaboration. One factor might constrain the solution space, but all these elements need to be considered together to properly identify if the team should work with a two-, three-, or four-week sprint.

If a team is very small, and nimble, then a two-week sprint could be a very good fit. If the communication levels are high, the scope of work is clear, and the overhead of meetings is streamlined, then working in two-week iterations could be ideal. Conversely, a team that is very large may be able to tackle and complete more tasks in a shorter amount of time and fit comfortably within two weeks. In both cases, it would probably work best if the teams were located in the same geographic location. Any diversity in the location can lead to a lag in communication, which can slow down the work effort. Having any kind of significant delays introduces a challenge in working within the shortest of sprint cycles but could be mitigated by adding additional time to the cycle. The most limiting factor in selecting a timeframe are the tasks themselves. If the smallest, or most typical, work unit is less than a week's worth of effort, then a short cycle is feasible. For something like a large architectural project in which the typical task takes a week and a half, it might make more sense to go with a three- or four-week cycle to achieve a more coherent piece of work within the cycle. Maybe the most important thing to bear in mind is that these numbers are not written in stone. If the team decides to work in two-week increments and, after a few cycles, finds that this is not working, then by all means adjust. Let the team decide what a better rhythm might look like and if adding a few days would make the difference.

The UX person also needs to consider, and have the team take into consideration, what the interaction with UX will look like as the team moves through the release cycle. Are you going to work a sprint ahead and need some dedicated time from the developers to support working collaboratively? If that is the case, the development team needs to accommodate that time. Or, are you going to work within the sprint and need a considerable amount of time from several team members? In this case, more time is needed to work together but overall the tasks may close out more quickly and result in a net gain in velocity for the team. All these things affect the velocity of the team and its ability to fit into a sprint cycle. This is not a problem, just something that the team needs to take into consideration, preferably before the cycle kicks off.

WHAT DELIVERABLES SHOULD UX PRODUCE?

This is a good place to invoke the classic usability practitioner answer for everything: "It depends." It would also to be fair to mention that this is another question that many Agile folks, especially those of the Lean UX persuasion, would quickly answer, "None!" Agile strongly encourages teams to move away from specifications to focus on more immediate and interactive forms of communication. Jeff Gothelf (n.d.), "Getting out of the Deliverables Business," is a very insightful article that does an eloquent job of expressing the rationale behind this idea. The proposal is to eliminate traditional deliverables and instead work off a rapid prototype that can be used both for testing and iterating with customers and becomes the artifact shared with the development team. It is the antithesis of the old-school specification document that is thrown over the wall and lands with a thud. Instead, this more interactive method of communicating design intent is many steps closer to resembling the actual implementation. And, if you are in a situation where you can do this, I wholeheartedly encourage you to do so. a certain amount of truth lies in the idea that creating a deliverable also creates unnecessary boundaries between the functional areas on Agile project. There are better and more efficient ways to share information about the design. If the level of communication and dynamics of the team allow for a seamless workflow that can survive and thrive without deliverables, then throw away the idea of specifications and do not look back.

However, in some environments, this ideal may not be entirely realizable. Sometimes, groups are spread across countries and time zones or encounter language differences that make nonverbal communication more effective than conference calls. In situations where some sort of deliverable needs to be created; the goal should be to produce the least possible amount of documentation. This cannot be emphasized enough. It really is okay to produce something if it supports the team's work, but there is a reason that Agile recommends eliminating it, so proceed with caution if you choose to ignore this advice. In generating any kind of document, the UX staff should also work with the team to determine the necessary amount of information and what it looks like. This not only sets expectations and goals but provides an opportunity to look for another solution to the problem that does not involve the creation of a document. The team should examine exactly why documentation of any kind is being produced and make sure that the needs are being met. It might help to think of these as communication-support items rather than deliverables, because that is really the purpose they serve. If a team is geographically diverse, smooth communication is not achievable through face-to-face discussion. Sometimes, posting a sketch or interactive prototype can facilitate communication immensely. The issue is more about sharing ideas effectively than handing off documentation. When large time zone gaps and language barriers are at play, such items can bring a common understanding more quickly than a meeting.

Long software releases, where the current version does not always represent the front-end design, can also lead to the need to create "speclettes." It is very easy for the Agile team to be aware of where they are "now" and the work going on in the current cycle. It tends to be a bit more challenging to remember where the team has been, how certain things have changed along the way, or where the team is headed. This information can be especially important for quality and documentation teams, as they try to stay on top of a moving target. To support these team, and generally remind everyone of the overall state of the design, it can be helpful to have some sort of document to which team members can refer when needed. I prefer to call them *speclettes*, because they are much more lightweight versions of their heavy duty cousin, the specification, and bear a much stronger relationship to a design brief. Usually, these documents contain a screenshot and a few words about an interaction for clarity. They may be grouped together by area or sprint for which they were designed or implemented. They can be attached to a design task, posted to a Wiki page, or printed out and displayed on the wall. Both the content and delivery mechanism are determined by the needs of the team, but minimal effort to deliver and consume should be the hallmark of these speclettes.

However, it is important to be very conscious of why any sort of documents are being produced. Sometimes, it will just be more comfortable for a UX person to produce something and publicly share it. We are so used to providing documentation, often to prove that what was designed is different than what was implemented and to have a point of reference for the subsequent bugs and change requests. It may feel as though, if we do not post such artifacts, then there is no way that resulting product will match the design intentions. In an Agile environment, this can end up being a self-fulfilling prophecy. Expecting a failure in communication will likely lead to such a failure because it means that there is no trust. If the request for documentation comes from outside the UX team, take a look at what is motivating the request. If the issue is that a functional area does not feel it has a sufficient understanding of what is going on in a given sprint or the overall release, simply posting a few annotated sketches may not address the root cause and may distract the team from solving the real problem. The design team is always creating sketches and it takes very little effort to share these on a Wiki page or attach them to a user story, but it important to make sure that doing so is not preventing or inhibiting a richer form of communication.

HOW SHOULD THE UX TEAM FIT IN WITH THE DEVELOPMENT SPRINTS?

A certain amount of debate centers around whether the UX team should work ahead of the development team or within the same sprint, on the same tasks. There are pros and cons to both approaches. The one thing we can all agree on enthusiastically is that the UX team should, at all costs, avoid working a sprint (or two or more) behind the development team. This can happen if the project was kicked off before there was a UX resource, often in companies where the few UX resources are distributed among many projects. Working behind the implementation leads to confusion among the entire team—no one can remember what the software is “really” supposed to look and act like, since some UI coding work will have been submitted prior to the design work then will need to be updated to reflect the design. The developers know that designs were discussed and reviewed and they remember seeing development tasks on that item occurring within the sprint, but it is a challenge to know whether the implementation included the design or not. This keeps the team from being able to trust that what they see is actually "done." It can also be much harder to affect change or appropriately influence the design.

While Agile environments are more open to rework or "refactoring" than traditional environments, there is a limit. The development team can cover the same ground only so many times before it is counterproductive to the schedule. And, as in waterfall, a design that requires an architectural change or a substantial rewrite of code will be a harder sell to get the development team to do it. Of course, that the UX team is behind the rest of the team is a red flag in and of itself; and while it may not be possible to change the behavior during a single sprint, the team needs to be aware that this is not a viable way to work. Bringing UX into the process late, regardless of the reason, might be a sign that other functional areas are not being completely integrated into the team, which is not a recipe for success.

Having the UX team work a sprint or two ahead of the development team, on its own velocity, is a conservative approach and can be great strategy for UX teams that are just finding their way with Agile. It creates the space for the UX team to really work on a design and iterate it, without affecting the time needed for implementation. This can also allow the UX team to dedicate a sprint to high-level design, usability testing, or another type of user research. Having a separate stream of work also creates a certain amount of visibility in to what the UX team is doing, which can be quite beneficial in an organization where the value of the UX team is not always clear. It is also an ideal approach for organizations where members of the UX team support multiple projects and need to build time into their schedule for that. One potential pitfall to this approach is that it can easily turn into a mini-waterfall. UX creates a design in Sprint 1, documents and hands it off to the development team to code in Sprint 2. If you take away the word *sprint*, you have a good, old-fashioned waterfall method on your hands. The best way to avoid this is communication. The UX team needs to make sure that it is working iteratively with customers, stakeholders, and the development team. In the fast-paced world of Agile, make sure that there is time for formal and informal communication among team members, even if it means assigning "support the UX team" tasks to critical team members to ensure their availability for and engagement in the UX sprint.

Working within a sprint on the same task as the development team, so that a given item is designed and implemented in a single sprint cycle, can be a challenge for a UX person new to Agile. However, it isa delightfully collaborative way of working, if a bit intense, and many teams are quite successful working this way. It is probably best taken on by teams that already have a healthy communication style, good working relationship, and are located in the same office. It may also be best if the UX resource supports only this project and does not try to balance several projects, as this type of approach is very fast paced and may not allow for load balancing on multiple projects. At a minimum, the UX person should take on only one major project when working in this style.

HOW DO YOU GET DEVELOPERS TO TALK ABOUT THE DESIGN OF ONE THING WHILE THEY ARE BUSY IMPLEMENTING ANOTHER?

Even in more traditional environments, this can often happen; it is just less explicit. However, it is a legitimate concern given that the pace of the Agile work might make it harder for the developers to be able to spend time on any activity that takes them away from accomplishing their work for the sprint. While bribing the team with candy and pizza is always an option, and one that I highly recommend, it gets you only so far, and a more sustainable solution needs to be found. Start by taking a look at what the issue is. Consider whether the developers are strapped for time or are just having a tough time switching focus, since either or both could be at play. If it is just an issue of time, try building in "UX-collaboration" time into the schedule. Have a task with a set amount of time assigned to the necessary resources and try to keep your collaboration work within those time boundaries to show respect for the developers' schedules. If the problem is more along the lines that you are asking them to multiprocess and it just is not working, then maybe the problem is that the UX team is working too separately. Consider an approach that has the UX team and the developer working together on the same problem, at the same time. This can take the form of in-sprint UX work or be as extreme as pair designing. The reality is this: If it is not working to ask a team member to help you out with one thing while his or her head is somewhere else, you should probably find another way to work together.

WHAT IF UX TEAM MEMBERS HAVE TO SUPPORT MORE THAN ONE PROJECT?

This is obviously a less-than-ideal situation but fairly common. Most UX teams have to face this reality and have found ways to deal with it. One approach is to have a UX person support one larger effort and one (or more, if it is feasible) smaller effort, so that the workload is balanced. This allows the UX person to spend time operating as a team member on the project that requires most of their effort and supports the ability to engage in as many of the team's event as possible. Giving the UX person the space to establish and foster a strong relationship with his or her product team is as important as having sufficient time to generate designs. Another way to tackle the work is to have the UX staff members work as consultants and integrate themselves with the teams only to the degree necessary. This approach can be tricky if the UX team or staff member does not already have a strong trust relationship with the team. Choosing to not attend certain meetings in order to spend more time working on the design may not go over very well with an Agile team unless they already feel comfortable with the designer. Additional care should be taken to make sure that the UX person engages in and encourages as much communication with the other functional areas as possible, informally or formally. Maintaining transparency is critical to building trust and keeping the collaborative spirit healthy.

HOW DO WE FIT USER RESEARCH INTO THE SPRINT CYCLE?

It is a common Agile UX practice to schedule regularly recurring usability sessions and get feedback on whatever is available at that moment. Sessions can be scheduled to occur on the same day every week or every month. Feedback can be solicited on design concepts, sketches, wireframes, prototypes, or working software. The user comments can be incorporated into new user stories and given priority by the team, no formal report necessary. When done routinely like this, customer reactions can be incorporated as seamlessly as observations made by team members and can function as a natural extension of the team. This is usability in the most Agile form.

If this is not possible in your situation, it is still possible to work standard research practices into an Agile framework. User research can be a part of the planning cycle and can include surveys, persona definition, usability testing, card sorting activities, or any other research technique used at this stage to help with the definition and clarification of requirements for the release. These activities can also be done within the cycle, as tasks assigned to the UX team in support of a specific user story or stories. As with any process, incorporating some user research is better than none, so create as many opportunities to do so as bandwidth allows.

Practically speaking, larger research efforts have to occur outside the sprint cycle, either using different resources internally or leveraging outside consultants. To a degree, Agile methods assume a good understanding of the customers and that in-cycle feedback will be sufficient to drive the product in the right direction. While this may often be true, it is not always enough. If your organization is entering a new market space, using of a new platform or other transformative technology, or expanding the product line in a direction that is more unknown, then up-front user research needs to occur. Ideally, this research occurs well ahead of the product cycle kickoff. Since we are often in fairly nonideal situations, this may not be an option. If the product cycle has kicked off and there is a need for research, coordinate with the team and conduct the research in as streamlined a way as possible while the developers work on architecture issues or other efforts that do not require design decisions to already be in place.

WHAT IF THE TEAM CLAIMS TO BE AGILE, BUT AGILE VALUES ARE NOWHERE TO BE SEEN?

I describe this as having the scent of Agile on it. Those involved call themselves *Agile* and give lip service to a method, but they does not practice Agile methods in any real way. Unfortunately, this is a fairly common phenomenon and can be one of the hardest challenges to overcome. It can happen for many reasons and sorting out the root cause is the best way to know whether or not the issue is intractable and, if it is not, identify potential solutions. It is entirely possible that it might just be the result of an overly enthusiastic scrum master, fresh off of Scrum training, who is dazzled by all the new things to do and fixates on them at the expense of Agile values. In this case, a candid team discussion might be warranted. If enough team members recognize and speak about the problem, they may be able to effect change and influence the culture of the team.

It is also very common, oddly enough, that no one on the team has any kind of training in any Agile beyond reading the Agile Manifesto (and maybe some teammates have not even done that) but are nonetheless speeding down the highway toward delivering the product using an Agile process. Conquering this dynamic requires members of the team who care about having a better understanding of the process. If there is an interest in genuinely applying the method, a champion who educates him- or herself and shares this knowledge can inform the team and create an Agile value-driven environment. Plenty of online training classes, articles, and blogs can be leveraged in place of more formal training or coaching. Doing some of these activities as a team not only increases the team's knowledge of Agile and allows it to be more effective, it can also serve as a team building activity that brings everyone closer together. However, if the team is happy enough to continue as it is, perhaps, doing so to simply avoid a more rigid process by calling themselves *Agile*, then the options for influencing cultural change are a bit more limited. In that situation, modeling good behavior as a team member and encouraging similar behavior might be the most effective option to shift the attitude of the team.

Then, there are the times when the problem is at a higher level of the organization. Executives or stakeholders who do not know or understand some of the basic details of Agile may simply view it is an interesting trend that should allow them to move faster. They want to do some of "that," but do not necessarily know what that means from a day-to-day perspective or realize that a certain amount of training and organizational support is needed for it to really succeed. When this is the case, these people may not understand the benefits in moving to an Agile process and may have unrealistic expectations of what to expect as a result of moving to an Agile process. In this environment, a successful project is the best way to get attention and support. Success would mean not just an effective adoption of Agile, but also an enthusiastic customer response and great revenue numbers. Showing that a team can not only deliver with Agile but t deliver products even better, more effectively, quicker, or more fully featured if the organization with support, training, coaching, or whatever is needed can be a very powerful message that gets the attention of people who are more removed from the process.

WHAT IF THE TEAM IS NOT COLOCATED?

Rarely is an ideal Agile team described in a way that includes teams in multiple time zones, but for most of us, this is becoming a common reality. Many of the teams interviewed for the case studies where in this situation, and more than a few of them were quite successfully Agile. Distance certainly represents communication challenges but nothing that cannot be overcome with a little bit of technology and creativity. The key is to identify which elements of the process need to be modified to accommodate the situation and where the communication gaps are. Wiki pages are a great way to post daily happenings and key bits of information. These can serve as a repository for different things that need to be shared and accessed asynchronously. Easily editable and informal types of communication like this can help fill the gap of daily standups when there is no convenient, common time zone in which to have a daily conversation with the entire team. The greater the distance between time zones, the more helpful an online communication tool can be. For remote meetings, tools that allow for both conversation and screen sharing can help replicate the dynamics of in-person demonstrations and discussions. If people are simply in different offices, without drastic geographical differences, IM chat tools can support the immediacy of communication that is more common when everyone sits together in a common area.

WHAT DO I DO WHEN SOMEONE USES “THAT’S NOT AGILE” AS A REASON NOT TO DO SOMETHING?

The first thing is to do is to consider if the statement is actually true. It is entirely possible that someone may be suggesting a course of action that completely undermines Agile values and violates it principles. However, do not be too surprised if this is not the case, and definitely do not take the comment at face value and assume that it is true. While individuals and teams do many things to undermine the Agile process, if someone is tossing around this particular phrase, it should raise a red flag. Agile methods are not particularly dogmatic in their approach, and it is entirely possible and very common to alter the method a bit to fit a particular situation while still achieving an Agile environment. If something is not working for the team, it should be discussed by the team and resolved in a way the team is comfortable with. Unfortunately, a person who responds to any kind of idea with that phrase is effectively shutting down the conversation and failing to address whatever it is to be resolved. Even if the solution would absolutely not work in an Agile environment, a problem lies underneath that should be investigated to see if there is a more-Agile solution. A more-Agile response would be to ask, "Why?" or "What are we trying to achieve with that?" and sort out what is actually going on and identify the best solution.

If a team member proposes to do away with all the daily scrum meetings, which goes against what the method recommends, the right answer is never to simply dismiss the proposal by saying, "That's not Agile." The more appropriate response would be to find out why the person made the suggestion in the first place and see if there is an issue that needs resolving. Maybe this is a knee jerk reaction representing a deep-seated mistrust of daily meetings of any kind, which might not require any more corrective action than to have a discussion about why the team has that activity. Or, that person could feel completely overwhelmed by the workload, and a daily event, even if it is only 15 minutes long, feels like one stress too many. This situation would obviously merit a different kind of discussion and potential solution. Really, the "Agile" thing to do is make sure that the team is communicating and feeling empowered.

HOW DOES THE UX TEAM PLAN AND RESEARCH FOR THE NEXT RELEASE?

This can be a difficult question to answer from an Agile perspective, since Agile methods really are focused on the "now" of the current release and the immediacy of the development work. The UX team can do a few things to facilitate this. If there is a central UX resource, it can be used to support the research efforts and coordinate with the product team to deliver the information in a timely way. It will also be important for the resource to think about making its findings fit easily into an Agile process, so he or she might want to look at how to write good user stories as a method of communicating research results.

If there is no central resource and the same person that supports the product and its design also needs to work on the research, then a little more finesse must be applied to the timing of these efforts. Consider whether or not the team uses a Sprint 0 for planning and if the sprint provides enough time to complete the necessary research. Depending on the length of the release and the sprints, maybe just using a sprint during the cycle for the research could provide enough time. This most likely will work best at the beginning or end of the release cycle, where there is often fewer design activities, but it is certainly possible for the UX person to pick a sprint at any point and dedicate time during that cycle for research. Depending on your team's schedule, the lengths of the sprints may or may not lend themselves to being used a period of dedicated focus. Perhaps, the timeframes are too short to do all the necessary research activities and analyze the results. If this is the case, the UX person could have a set of tasks dedicated to research activities that occur throughout the sprint. If the efforts are forward looking, it may be a little awkward to make these tasks fit in as part of the Agile method t being used. If this is the case, then simply do not make it part of the Agile method. Ideally, everyone on an Agile team is a dedicated resource, but this often is the case only for the development team, and many of the other functional areas support other projects. It could make sense to treat the research work as another project, which it is. The UX person simply adjusts the available time for the project team and carves out the time the need to work on the research. The time spent on research can vary throughout the release, and the UX person simply takes on more or less project work to accommodate it.

HOW DO YOU MANAGE INTERNAL STAKEHOLDERS?

Some touch points with internal stakeholders are built into the process as a part of the planning activities and in the demonstration of the work done during the sprint. This is mostly intended to communicate with immediate stakeholders that are somewhat involved in the project and have a good sense of the work that is occurring. However, there might also be a need to work with stakeholders who have less contact with the team, especially if it is a pilot project to explore the methodology. This may require an occasional meeting with these people to let them know what the project is tackling and the kind of progress it is making. It is probably worth doing these meetings as dedicated events, separate from the day-to-day work and involving only necessary team members. There is a natural tendency to want to invite these interested parties to attend the demonstrations at the end of the sprint, but try to resist that temptation. Meeting as a team to discuss what was achieved during a specific sprint is a different conversation than talking to management about the project, and trying to do both at once will likely prevent the right level of communication on either front.

SUMMARY

There are many questions about how to get started with Agile UX, and the answer to most of them is "It depends." As design professionals we need to be comfortable with a certain amount of ambiguity, and it might help to think of fitting into Agile as a design problem. Consider the project team to be the users, identify their needs, and create an elegant solution. There are many best practices but few definitive answers on to how to practice Agile UX.

Reference

Gothelf, J., n.d. Lean UX: Getting out of the Deliverables Business, Retrieved April 2, 2012, from http://uxdesign.smashingmagazine.com/2011/03/07/lean-ux-getting-out-of-the-deliverables-business/.