JUS
JOURNAL OF USABILITY STUDIES

Vol. 8, Issue 3, May 2013 pp. 55-60

Five Agile UX Myths

Diana DeMarco Brown
Principal UX Designer
Nuance, Healthcare
Burlington, MA
United States
diana.demarco@gmail.com

Introduction

As more and more user experience practitioners dip their toes into Agile waters, the practice of Agile user experience (UX) continues to evolve and mature. While researching my book on Agile UX (*Agile User Experience Design: A Practitioner's Guide to Making It Work*), I talked with designers, developers, and consultants about their experiences with Agile UX (the practice of designing the user experience in an Agile development environment, which is a flexible, incremental, iterative design process), and I continue to hear stories from the field when I give presentations and workshops in my local area. Some misconceptions evolved along with our practice and continue to persist and color how we approach design when working with a development team that engages in Agile software development. In my conversations with UX friends and colleagues who are in various stages of Agile adoption, I hear the same fears consistently expressed—for example, how will we fit usability testing in or how will we document our designs? Even the question of whether we can produce quality designs while working with the many iterations and frequent deliverables that are the hallmark of Agile. And yet, when I look at UX teams who are successfully implementing Agile UX, these concerns seem unfounded. But where there's smoke there's fire, and each of these issues merit closer examination so we can see what the real story is.

Usability Professionals' Association
Promoting usability concepts and techniques worldwide

Myth 1. There Is Only One Way to "Do" Agile UX

There are people who preach that Agile UX should be conducted this way or that, and they all have valid methods and legitimate approaches to creating usable designs in an Agile environment. Once a team has found a formula that works for them, they will want to shout it from the rooftops and present it at every conference that they can. Because they are only presenting their own viewpoint, it can sound as if they are sharing the ultimate solution with you. For example, some people say that you must have your UX team working a sprint ahead of the development team. But you'll hear other people swearing that working a sprint ahead is just a mini-waterfall (or micro version of the traditional—design it, throw it over the wall, and then build it—style of production and completely non-Agile). And both camps would be justified.

It can be easy to lose sight of the fact that these ideas are not one-size-fits-all solutions. Listen to what your fellow practitioners have to say, and take inspiration from their examples. However, if you find that the approach used by a small co-located team working on a website is not a fit for your complex enterprise software project with a geographically diverse team—that's okay. Adapting someone else's sure-fire method to your own situation may require some tweaks, because different challenges require different solutions.

When you're evaluating a given method, consider what your situation has in common with theirs.

- Is the organization of their UX team and the larger company similar to yours?
- Are they dealing with new functionality in an isolated code base or complex legacy code with upstream and downstream dependencies?
- What is their relative maturity with Agile compared to your team's?
- Does it sound like they are in a naturally collaborative environment or one that is very formal? Listen for cultural cues.

All of these considerations, especially culture, can affect the way software is designed and developed in an Agile environment. Ignoring these factors and trying to fit your square peg into someone else's round hole will only lead to frustration.

For example, when I embarked on my first Agile project, I attended several presentations given by colleagues who had gone down the path before me with great success. They presented strategies that had worked for them, including that a parallel sprint approach was the best way for the UX team to be effective in an Agile environment. It sounded like a great approach, so I engaged my team, fully expecting to replicate what others had done. It took less than a week for me to realize that I was going to have to adjust—their model had worked well with a small, co-located team, the UX folks had been involved with the project from day one, and were dedicated to that project. But we were starting on our project after the engineers had been working for several weeks, our large team was in three locations, and I was juggling another non-Agile project. While I kept the parallel sprints because it was the best way to have a separate pace from the development team, I deviated in many other ways from their model. I spent quite a bit of time feeling guilty and "non-Agile," and I really shouldn't have. I worked with my team in the best way for our situation, and we adjusted as needed when we had trouble finding time to collaborate or getting the designs done on time. We may have had a less than ideal situation and our process might not have been a textbook example, but we were collaborative and communicative, so it worked. If I'd tried to make the team work the way I thought an ideal Agile team operated, I would have spent all my time fussing over process and not enough time just getting things done. Instead, I took what I needed from the techniques my colleagues had shared and made the rest my own.

Upon reflection, if you see that there's a mismatch between your situation and someone else's, don't give up hope. Ideas can be found in the most unlikely places, and even if you can't fully adopt a particular convention, you might be able to borrow an idea or two from it. Or simply use their story to spark inspiration to try something new in your organization. Just as you don't need to feel like you should practice Agile UX a certain way, you shouldn't dismiss an idea out of hand. The idea is to understand why a given tactic was so successful in another setting and consider how you might influence your team in a different way.

Myth 2. You Can't Conduct User Research

When you move from a waterfall environment to an Agile one, it will feel like you flipped a switch and are moving at breakneck speed. This feeling is partly due to the fact that you've left behind a comfortable and familiar rhythm and exchanged it for one in which design activities occur more frequently. You may or may not have the luxury of adjusting to and mastering this new cadence before you are asked to fit user research activities into the mix. Figuring out how you are going to hit your design deadlines can be daunting enough, but where are you going to find the time to squeeze in usability testing? If those thoughts have run through your head, you are in good company, because that is one of the questions I hear most often. The truth of the matter is that most Agile teams are actually able to fit in more customer feedback more frequently than is typical with other processes. Both design validation and long-term, deeper research projects can be fit in to an Agile cycle.

Design validation is the easiest to work into an Agile release. The key is planning. Set up a regularly occurring time to have customer feedback sessions and conduct them no matter what. Too often, we wait until we get to a milestone before showing the product to customers, but those milestones often occur too late to make necessary adjustments to the product. Let go of the idea that you need a high-fidelity prototype or working code before you can get feedback. Schedule a session and bring what you have—hand drawn sketches, wireframes, or concept maps—and have a conversation with your users. Or use a version of the Rapid Iterative Testing and Evaluation (RITE) technique (learn about it here: http://uxmag.com/articles/the-rite-way-to-prototype). Or involve your customer in participatory design. Conducting predictably occurring sessions once a week, once a sprint, or once a month will allow you to have constant customer engagement throughout the production process. This kind of testing can be conducted by UX teams working in parallel sprints and with teams that are working in the same sprint with their developers. For teams working within sprint, it will be important to find the right day in the cycle to have the testing and then provide the feedback. If you expect the development team to act on the feedback immediately, you need to plan the testing event to occur early on in the sprint. If your team is comfortable reworking the design in the next sprint or you will be testing a design for implementation in the next sprint, then the session needs to happen early enough to inform the planning session that will occur to finalize the scope of the sprint.

Regularly scheduled events also have the added benefit of making it routine for the product team to include customer feedback into the process, rather than having usability testing be a special occurrence that requires an out of the norm response. When Suzanne O'Kelley at AppNexus used this approach, she found that having such routine feedback sessions also provided the additional benefit of increasing customer confidence and trust. Customers who returned for subsequent sessions could see that their input was taken seriously and acted upon because they were able to see the changes in the product as the design evolved. (To learn more about this case, read Suzanne's blog post at http://techblog.appnexus.com/2011/autotagger-a-case-study-for-lean-ux.)

Longer term research projects need to be treated a little bit differently. Ideally, research would be conducted ahead of a kickoff in order to provide context and data that will inform the creation of user stories. If there is a need for this type of deeper research for a project that has already kicked off, it would be good for the researcher(s) to be plugged into the cycle and functioning as a part of the team. However, because a single piece of research may not fit within an iteration, the researcher may need to function a bit outside of the day-to-day rhythm of the team. Then when they have findings that they want to share with the team, they should try to do so in a way that fits in with how the team typically handles their rework or refactoring. It's critical to be conscious of how to integrate the results into the process and help the team react to the findings—sure, you can deliver a presentation or a report, but it's more important that you sit down with the team and work with them to define (or refine) user stories or participate in discussions about how to adjust their priorities.

Myth 3. No "Upfront" Design Is Allowed

In traditional waterfall environments, the design of the software is often defined in minute detail before development even begins. The design can be nailed down months, if not years, before the software ships to the customer. This practice makes it difficult to respond to changes along the way and inevitably unanticipated technical challenges or shifting requirements arise that necessitate changes. It can be difficult to accommodate such changes without reworking the original design significantly, which may not be feasible. There's also the risk that designers will get emotionally attached to their work, having invested so much time and energy in it, that they are reluctant to make adjustments. Agile environments are intended to create a culture where such course corrections are anticipated and addressed as a part of the process. This means a move away from such detailed definition and its accompanying documentation. But it doesn't mean that you should just try to wing it and expect the design to reveal itself to you as you move through each iteration. Take a look around at the other functional areas on an Agile team, and you'll see that they're all working with some sort of high-level plan. The development team doesn't figure it out as they go along and hope that the code fits together coherently, because that's a recipe for disaster for a project of any significant size. We should take a lesson from our development peers and recognize the importance of creating a design roadmap, which we revisit and update as necessary, that will allow us to create a cohesive user experience.

This roadmap could be a simple sketch, a workflow diagram, a handful of wireframes, or a bunch of colorful Post-its stuck on the wall. Do the least amount of work necessary to give you and your team a sense of where the design is heading and be willing to change it often. Not only will this practice help you keep your sanity as you get down in the weeds of designing one chunk at a time, but it'll help manage the expectations of the team around you. The design roadmap will provide a rough idea of what you are considering for a solution and keep that idea in the back of the other functional areas' minds as they do their work. This foresight will allow them to be proactive about identifying potential technical roadblocks for the design as they do their own work. Even if the roadmap is the most basic of sketches, it also provides a "face" for the product. Often having a visual of some kind gives people a much more concrete sense of what's coming and allows them to raise issues or provide ideas that they might not have had otherwise.

Myth 4. UX Deliverables or Documentation Should NOT Be Created (or MUST Be Created)

There's a new school of thought that promotes the idea that UX teams should not be producing deliverables. On the other hand, those of us who have spent our careers producing deliverables are ready to create documents at the drop of the hat. Neither approach is necessarily wrong, but before going in either direction it's important to think through your motivation.

Documents are a bit of an anathema in Agile because they don't necessarily support or engender collaboration or real communication. No matter how many collaborative sessions you conducted to get to a design, the minute you sit down to document the design, you are performing a solo activity. And because a design might dynamically change over time, you are, at best, just documenting a moment in time. The act of handing off a document for review also creates distance between team members. If you are lucky, folks might review the document and comment on it, but that is not a real discussion. And the likelihood of anyone reading the document again, after the comments are responded to, is pretty slim. So, if you're looking to get away from that entirely, I applaud you. You're approaching communication in a new way, and that is fantastic. But, I'll also throw the caution flag. If you have an overseas team and English is not their first language, there can be a very strong case for producing a brief document on occasion. Sending written information and pictures ahead of a meeting is a good idea to allow people on the other end to prepare for a discussion. It can also be a good way to summarize the collaborative discussions that they miss out on because they are at a different location. I've had good luck using one-page documents with screenshots and callouts to seed discussions with teams based in China. Just bear in mind that using a document in this case is to support and enable collaboration, not replace it.

Small documents can probably serve a similar purpose in other situations as well, but if you are reflexively turning to document creation to answer other situations you really need to take a

pause. It can be easy to produce a sketch or a specification or a prototype, and in fact you may already have some sketches lying around as artifacts from the design process. So what's the harm in adding a few words to them and distributing them? Well, you could be missing an opportunity to engage in a more collaborative activity. I recall whipping up simple one-page specs for the benefit of a documentation colleague who was based in a faraway office. It didn't take me very long to put them together, and it was really helpful for her, so I thought I was doing a good deed. Until I talked with a colleague who told me that their documentation people usually ramped up on the project at the time the UX workload was decreasing, so the UX folks would sit down with them and have a meeting to review the designs and bring them up to speed. I slapped myself in the forehead and realized that I had missed out on a chance to sit down, albeit virtually, with an experienced colleague and possibly improve the design. You may be creating similar documents, with the best of intentions, but take a minute and consider whether or not the problem you are trying to solve could be addressed in a way that supports live, interactive communication.

Myth 5. You Don't Need To Have Any Agile Training

There seem to be many teams operating in Agile environments with little or no formal training. Most UX folks receive no Agile training at all, even if their company has provided some for the development team. However, it is critically important that the UX team members have a well-informed understanding of the methods that their team is using and what their Agile UX colleagues are doing. The UX team is rarely a part of the decision to adopt Agile or even what form of Agile to practice. However, because most methods don't address UX directly, we practitioners are responsible for defining how UX fits into Agile practices. And how many Agile development environments are exactly the same? How will you define your relationship to a process that you have no understanding of? Don't spend your time re-inventing the wheel. Learn from your UX colleagues who have spent years wrestling with Agile—they have a lot to tell you.

There are so many resources out there to support you, even if you don't have a training budget. Start with the Agile Manifesto (www.agilemanifesto.org) and read through the twelve principles. This is the foundation of everything Agile, so go to the source and read it for yourself. After that, educate yourself on the type of Agile methods your team will be practicing so you can understand how it works. Your company may do its own homegrown version of things, but it's still beneficial to learn how a process like Scrum (an Agile process that defines events to support increased communication and transparency) works and understand its framework, because it gives structure to the intentions of Agile and is a great reference point. Then, start looking around to see who's talking about what in terms of Agile UX. There are tons of blogs and groups and presentations and books out there to help you learn more about what your UX friends are doing in the world of Agile.

Conclusion

When it comes to Agile UX and finding the approach that works best for you, I highly encourage the approach of "Trust, but verify." As the practice continues to mature and evolve, more of our colleagues are presenting and publishing on the topic. New tactics and techniques are being introduced, and there is a wonderful body of work out there for us to use as inspiration. However, it is important to take all of these with a grain of salt. Every Agile team and project is different and your mileage may vary. Trust that the advice that they are giving you is solid, but verify that their approach will work for you in your situation. Similarly, if you hear a team member describe an Agile concept as if it is an immutable truth—take a breath and a closer look. There really are few absolutes in the Agile world and "can'ts" and "musts" generally don't apply. It doesn't mean there isn't a grain of truth in there, but dig deeply and get to that core truth and see what it really means to you and your work. You will probably find that the "can't" is really more of a "shouldn't" and then learning why you shouldn't and when you should. That exploration is where the real learning and inspiration occurs, making you well-informed and creative when defining your Agile UX approach.

Agile UX can certainly make different demands on you, but making the adjustments is well worth it. Ultimately, it may lead to a more collaborative way of working. The other functional

areas will become your co-designers and provide you with more direct support and interaction than ever before. It will also provide an opportunity for you to step back and redefine your approach to UX in order to fit into a fast-paced, collaborative, communicative environment.

About the Author

Diana DeMarco Brown
Ms. Brown, author of the recently released book, *Agile User Experience Design: A Practitioner's Guide to Making It Work*, currently works as a Principal User Experience Designer at Nuance in Burlington, MA. She holds an MS in Engineering Management and a BS in Engineering Psychology, both from Tufts University.