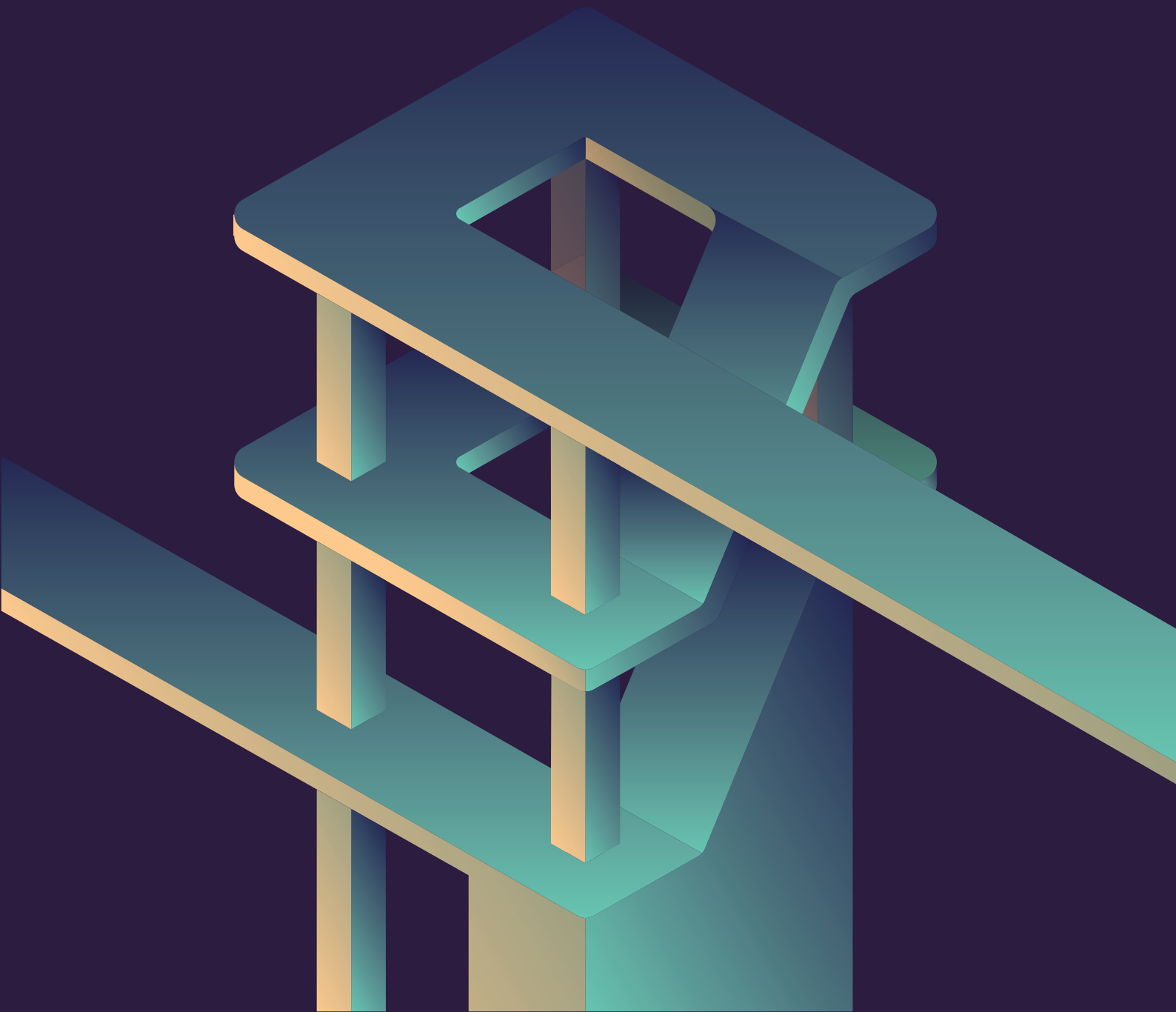
UXPin

UXmatters

The Definitive Guide to Integrating UX & Agile



The Definitive Guide to Integrating UX & Agile

Index

Authors

Richard F. Cecil

Rick has nearly a decade of experience envisioning and designing innovative solutions for a variety of companies, including startups, non-profits, universities, and Fortune 100 companies. During his tenure at Motricity, he worked with Cingular, BET, Ask, Alltel, and other clients and was the Design Lead for their core product offering. Rick was a co-founder of both the Interaction Design Group – now the Interaction Design Association (IxDA) – and the Triangle Usability Professionals' Association (UPA). Active in the UPA, he also serves on the organizing committee for World Usability Day. Rick is also the UXnet Local Ambassador for Research Triangle Park.

Jacob Harris

At CDK Global, Jacob leads UX strategy and design for a suite of mobile and SaaS communications applications and communications-intelligence products. He is particularly interested in business-model validation through UX research. As opinionated as Jacob is, he should have a blog, and he promises to get around to writing one soon. He is working on a graphic novel that he expects to publish sometime in mid-2016.

Atul Handa

Atul is a seasoned UX leader with 16 years of industry experience. He has a Master's in Information Technology and is also a certified user experience designer and researcher. His greatest strengths are in product strategy and UX best practices and methods within an agile world. He helps implement design thinking in tune with the latest standards and industry benchmarks. He also lends his expertise to bridging business goals and user aspirations, translating complex business ideas into actionable blueprints that he's based on exhaustive user research. He has successfully led the design and development of Web and mobile products in a variety of domains. Beyond the digital world, Atul enjoys spending time with his frisky, two-year-old son. When time allows, he likes to get creative with his woodworking skills in his garage.

Kanupriya Vashisht

A seasoned content strategist, editor, and writer with more than 15 years of experience, Kanupriya provides communications-strategy leadership that, supporting a user-centric approach, to a diverse array of clients. She oversees projects throughout their entire lifecycle – from research, analysis, content strategy, and design to content creation and testing. She also regularly writes and edits features and columns for leading magazines, newspapers, and Web sites. Kanupriya has a Master's in Journalism and Mass Communications from Arizona State University.

Todd Zazelenchuk

For the past 12 years, Todd has led UX projects and teams in both industry and academic environments, including Plantronics, Intuit, Whirlpool Corporation, and Indiana University, where he earned his PhD in Instructional Technology and Human-Computer Interaction. He has authored peer-reviewed articles, contributed to Tullis and Albert's Measuring the User Experience, presented at international conferences, and received design and utility patents for his work across multiple industries.

Jeff Larson

As an Interaction Designer at Plantronics, Jeff participates in all phases of the user interface development process—from ideation and opportunity identification to the design and implementation of both hardware and software. He excels in communicating concepts to stakeholders and has a fervent desire to expand and refine his UX skillset. To continue his professional growth, Jeff will be pursuing a master's degree in UX design.

Jeff Veen

As a Design Partner at True Ventures, Jeff helps companies create better products. Jeff advises companies like about.me, Medium and WordPress. Previously, Jeff was VP of Design at Adobe after they acquired Typekit, which he co-founded and ran as CEO. Jeff also co-founded Adaptive Path. While there, he lead Measure Map, which was acquired by Google. At Google, he redesigned Google Analytics and lead their apps UX team.

Adrian Howard

Adrian Howard is passionate about building effective teams and great products. He co-founded Quietstars to help organisations do that using Agile, Lean and User Experience practices. You'll find Adrian working with startup and product development teams — doing everything from coaching & teaching to hands on user experience & development work. With more than fifteen years experience working with startups, established businesses and agencies Adrian is an active member of the Agile, Balanced Team & Lean UX communities.

Laura Klein

Laura is a product management and UX expert with over 20 years of experience in tech. Prior to Usersknow, Laura's roles included the VP Product at Hint Health, Mentor at Tradecraft, and Director at Sliced Bread Design. She is also the author of UX for Lean Startups (O'Reilly) and Build Better Products (Rosenfeld Media).

Daniel Castro

Daniel is a seasoned UX professional of over 15 years in strategic, operational, and product design leadership. Currently, he serves as a Design Director at Sumo Logic. His past UX experience includes design and leadership roles at TIBCO Software, Sephora, Walmart eCommerce, and Verizon. His designs have been featured in Parenting Magazine, winning the 2015 CES Innovation Award in "Better World", and being presented in keynote speeches as a market differentiator in ease of use.

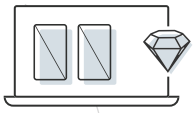
Design Systems in UXPin

UXPin

One platform for consistent design and development.

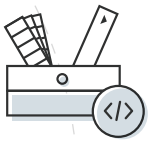
PayPal Sapient adidas HBO Microsoft

DESIGN SYSTEMS



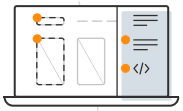
Design Language

Sync Sketch with UXPin for a consistent design language: fonts, colors, icons, assets, and more.



UI Patterns

Scale designs consistently with Symbols and interactive components.



Automated Documentation

Documentation syncs everywhere and travels with library elements.

Modular design and development

Scale quickly with design system libraries.

One source of truth for everyone

Close your knowledge gaps. Formalize your design and code conventions.

Painless documentation and developer handoff

Eliminate busywork. Generate style guides, specs, and documentation.

Tracy Dendy
HBO

"My productivity and developer productivity have both increased. They love that they can collaborate and move quickly to a powerful experience.

To book a demo, call +1 (855) 223-9114 or email us at sales@uxpin.com

Introduction

Agile and UX don’t always play well together.

After all, Agile development never originally accounted for UX. As the popularity of the design profession grew in recent years, many companies still struggle to fit the square peg of UX into the round hole of Agile.

It doesn’t need to be like that. As a collaborative design platform, we’ve seen customers from small agencies to 10,000+ employee corporations evolve their processes.

The first step is understanding that Agile and UX share similar goals. Both aim to deliver business value more quickly. Both rely on customer feedback to drive tighter iteration loops. They become incompatible only when we obsess over the letter rather than the spirit of collaborative product development.

In the first half of this e-book, you'll get best practices from designers published on UXmatters. In the second half, you'll get expert perspectives from hours of interviews with:

- **Jeff Veen** – Design Partner at True Ventures
- **Daniel Castro** – Design Director at Sumo Logic
- **Laura Klein** – Principal at Users Know
- **Adrian Howard** – UX Designer at Quietstars

You'll learn not just about how different designers adapt to the realities of product development, but also the common mistakes they've seen along the way.

Marcin Treder, CEO of UXPin

Clash of the Titans: Agile and UCD

By Richard F. Cecil

Agile software development[1] has become fairly popular in the last few years, leaving many UX professionals wondering how user-centered design (UCD) can fit into an extremely fast-paced development process that uses little documentation. User-centered design can involve a variety of techniques that provide insights into users' wants, needs, and goals, including ethnography, contextual inquiry, contextual interviewing, usability testing, task analysis, and others. But all of these take *time* – time that an agile development process might not allow. There is hope, though. Agile and UCD methods are not completely at odds with each other – and in some cases, agile development can even enable a *more* user-centered approach. By taking the time to understand the differences and similarities between agile development and UCD, it's possible to devise a process that is both user-centered and agile.

Similarities Between Agile and UCD Methods

Let's start by exploring the similarities between the two approaches.

I particularly like Alistair Cockburn's comparison of an agile development process to a cooperative game: "Software development is a (resource-limited) cooperative game of invention and communication. The primary goal of the game is to deliver useful, working software. The secondary goal, the residue of the game, is to set up for the next game." Thus, according to Cockburn, an agile development method, at its core, is about delivering useful software. According to Rassa Katz-Haas, user-centered design is about understanding people's needs – so we can provide useful software. She writes: "[User-centered design] places the person (as opposed to the thing) at the center… UCD seeks to answer questions about users and their tasks and goals, then use the findings to drive development and design."

A human-centered design approach allows us to better understand the people who use our products, while agile development techniques let us build, test, deliver, and revise our products faster. This is what software design and development is all about: delivering meaningful products to people. So, if these two methods seem to complement each other so well, why is there so much friction and frustration when it comes time to integrate them? A surface examination of the issues can't answer this question. We must dig into the details of these methods, where integration gets more complicated.

Differences Between Agile and UCD Methods

Defining a development process is a tricky thing. The most challenging aspect is that of defining how people are going to work together. People who have egos and opinions. People whose skills may be undervalued or who may not be fully committed to the process. Define a process too strictly and it becomes unbearable and unadaptable; define it too loosely and there is a risk of not including the right people at the right time.

Agile's iterative development cycle is one of the method's strengths, but it also makes for some tight deadlines. As the now infamous interview between Alan Cooper and Kent Beck[2] shows, the timeline is perhaps the most controversial aspect of agile methods. In such a high-speed development cycle, do we have time to fully understand users' needs? The short answer is *no*. The long answer: If we're defining users' needs *during* development – even in an agile development process – something has gone *horribly* wrong.

Case in point: One of the engineering teams I've worked with used six-week cycles and two-week iterations. My original plan was to stay one iteration ahead of them, but that proved problematic – especially when I got to the second iteration. By that time, I was supporting revisions to iteration 1, supporting development on iteration 2, while designing for iteration 3. Needless to say, I drank a lot of coffee and burned the midnight oil for several weeks.

Redesigning a Development Process

What to do? I started out by taking a deep dive into agile development methods. I needed to better understand the theory behind these methods before I could even begin to resolve the friction between the two processes. In doing this study, the first thing that became apparent to me was that agile is a method of development. It's certainly not a research process, and it's only loosely a design process. Despite what many proponents of agile development methods would have you believe, they cannot replace the need for some up-front user research or design. Agile proponents may cringe at that statement, but I stand by it. That said, though, while agile methods can significantly reduce the amount of up-front design that's required, it does not, in any way, reduce the time user research requires.

1. **User Research**

   User research and agile do *not* play well together. The time to conduct field research is *not* during development. Research should occur *before* any design or development work begins. This may seem obvious, but is an extremely important point – especially when you consider that agile development is about writing code as early as possible and delivering working software as often as possible. Conducting user research slows things down. However – and I'm probably preaching to the choir here – user research provides insights into customers and their needs that will help a product team to identify useful new features and products as well as to prioritize those features and products for development.

The outcome of user research is documentation that describes users and their needs and goals – for example, personas that represent a product's primary and secondary audiences. Many agile development methods employ user stories – which are somewhat similar to scenarios – and personas can become the main actors in these user stories. By creating behavior and goal-based personas, you can make your user stories much more effective. At this stage – before design has begun – user stories should be fairly high level and detail only how people will generally interact with your product.

I recommend involving a representative from engineering in your field research. By seeing users in the wild, engineers will develop some empathy toward them. Of course, doing this is helpful no matter what type of development method your team is using. Also, include this engineer in the process of developing personas and user stories. Getting buy-in from engineering is critical to the successful use of personas and user stories during the development cycle.

2. **Feature Prioritization**

   Once user research is complete, and you've created your personas and high-level user stories, it's time to define and prioritize product features.

   Feature prioritization is important in agile, because it lets engineering focus first on the features that make up basic working

software. There are many different prioritization techniques – some belonging to the different agile methods. Of all the techniques for ranking features I've encountered, I prefer Janice Fraser's approach[3], because it's simple, objective, and accounts for three important perspectives on product definition: the business, users, and technology. Using her prioritization technique, stakeholders rank the importance of each feature to the business, UX professionals rank the importance of each feature to the users, and technical analysts rank the feasibility of implementing the feature. Once they have all ranked the features, the product team combines the different rankings and compares them to identify the most important features.

3. Design

Once you've defined the feature priorities for your product, it's time for design. Most agile development cycles comprise a time-boxed development period, or iteration, of 2–4 weeks, and the expectation is that a developer will have produced working software by the end of that cycle. As a designer, you should focus your efforts on designing the features in the next one or two iterations. If you get too far ahead of the engineers, you run the risk of requirements changing, necessitating a lot of rework. However, you need to stay far enough ahead so you have time to do usability testing and remedy any usability problems in the previous iteration, support development on the current iteration, and design for future iterations. If your development team's iterations are brief, I recommend designing at least two iterations ahead. But with four-week iterations, designing one iteration ahead will probably be sufficient.

All design should start with the creation of detailed user stories. Be sure to involve the engineers who are developing related features in creating the user stories. You must have their buy-in for this technique to be effective. Once your user stories are complete, create your design documentation of choice. Keep in mind that there is a delicate balancing act you must manage. First, you must consider that you don't have much time for producing a lot of design documentation, so the documentation you do create needs to be as concise, effective, and useful as possible. I've found wireframes and workflows to be most useful for engineers.

Second, in many agile development methods, documentation is nearly considered taboo. Some would say that the attitude "don't document, do" is a core strength of agile, but where does that leave designers, whose main deliverables are information architecture diagrams, usability reports, wireframes, workflows, specifications, prototypes, and a myriad of other documents? This documentation helps designers model user interactions with a technology product, so it's important – especially when you consider that it can actually reduce development time by identifying common patterns and eliminating unnecessary steps before any code has actually been written. The challenge, then, is to find a balance between providing too little documentation and too much documentation. Determining how much documentation is enough is not easy.

I have found that defining the 20 percent of a feature that includes its most important aspects gets it to a relatively good place. You can generally define and design the most important user interactions

for a feature. Then, whiteboard sessions throughout an iteration let you define the remaining 80 percent.

4. **Develop and Test**

   One of the core principles of agile development is its focus on delivering working software frequently. In some agile methods, a customer review of the working product even caps each biweekly or monthly iteration. In addition to or, sometimes, instead of these reviews, you can run iterative usability tests. According to Jakob Nielsen, if you do three rounds of usability testing, testing five people in each round, you can discover the majority of usability problems[4]. With agile development, you can test and revise a feature over three iterations, allowing you to discover and correct most usability problems before launching a product.

Conclusion: Integration of Agile and UCD Requires Collaboration

Regardless of whether your development team adopts agile techniques, they hold plenty of valuable lessons that make it worthwhile for UCD practitioners to learn more about them.

In this article, I've described some of my experiences and shared my thoughts on how to integrate agile software development and UCD. Of course, the goal for any process is to create something that will work for both the designers and the engineers working on a project, not to impose a canned process on a team. What's actually pretty

amazing about agile is that, at its core, it's simply a manifesto that emphasizes the importance of collaboration to delivering useful, working software[5] – a manifesto that nearly anyone can get behind. Regardless of whether your development team adopts agile techniques, they hold plenty of valuable lessons that make it worthwhile for UCD practitioners to learn more about them.

Originally published on UX matters.

Notes

[1] There are many different agile development methods, including Scrum, Crystal, and XP. In this article, I focus on only the concept of agile software development, not any particular agile method.

[2] In Extreme Programming vs. Interaction Design Elden Nelson interviewed Alan Cooper and Kent Beck. By the end of their discussion, neither seemed to understand the value of the processes the other had defined.

[3] In Setting Priorities, Janice Fraser outlines a simple and effective way of prioritizing features. The only thing I would consider adding is a column for weighting the feasibility of assigning a UCD resource to define a specific feature, which would be especially helpful on product teams where one designer is supporting several engineers across several features or products.

[4] Jakob Nielsen makes a strong case for limiting testing to only five users in Why You Only Need to Test With 5 Users.

[5] If you are interested in learning more about agile, you can read the full Manifesto for Agile Software Development.

Agile UX

By Jacob Harris

When a UX designer must work across product teams and juggle multiple sprints and priorities, an agile development process can seem both chaotic and rigid at the same time.

I work in a very strict agile environment where the small size of the UX group means we have to straddle product teams and be creative in allocating our time and effort. For User Experience, working within an agile environment is often fraught with challenges. When priorities align and a UX designer is fully embedded in a product team, agile can be that designer's best friend – helping in prioritizing deliverables and organizing cross-functional efforts. However, when a UX designer must work across product teams and juggle multiple sprints and priorities, an agile development process can seem both chaotic and rigid at the same time.

My UX team has been developing and testing a model in which UX designers work *across* agile teams within our organization. So far, we are seeing some fairly positive results. In this article, I'll share

how we do it, so you can try out our approach or modify it to suit your needs.

The Problem

We've encountered one major roadblock time and time again: how to rank the priorities of one agile team against another's to determine what stories to complete first.

Since the problem in my organization is that User Experience is a small group and UX designers must straddle multiple projects and balance multiple priorities, we've encountered one major roadblock time and time again: how to rank the priorities of one agile team against another's to determine what stories to complete first. Because there was no good answer out in the world, we realized in fairly short order that we would need to design a method internally to resolve the problem.

The Solution: A Kanban Board

While discussing this issue during one of our backlog grooming sessions, an engineer suggested that we might be able to use multiple Kanban boards as part of a solution. *Kanban* is a workflow-management system that uses a board and sticky notes to prioritize and manage workflow debt. In its most basic form, a Kanban board comprises three columns: Backlog, In Progress, and Completed. A team writes

tasks on sticky notes, then places them in the appropriate column. Generally, there is a limit to the number tasks that can be in the In Progress column at any given time, so you can add new tasks to it only as you complete other tasks.

Overall, this is a simple and elegant way to get team members on the same page and make them accountable to each other, because it enables people collaborate and internalize priorities as a team.

Many Kanbans, One Board

Getting back to how we use Kanban boards at my workplace… Since User Experience is a separate department within my organization, using other teams' Kanbans would not work. After much scribbling on whiteboards, we came up with a simple solution: Let's create one big Kanban board that consists of other teams' Kanbans, as shown in Figure 1. That way, their backlog becomes our collective UX debt.

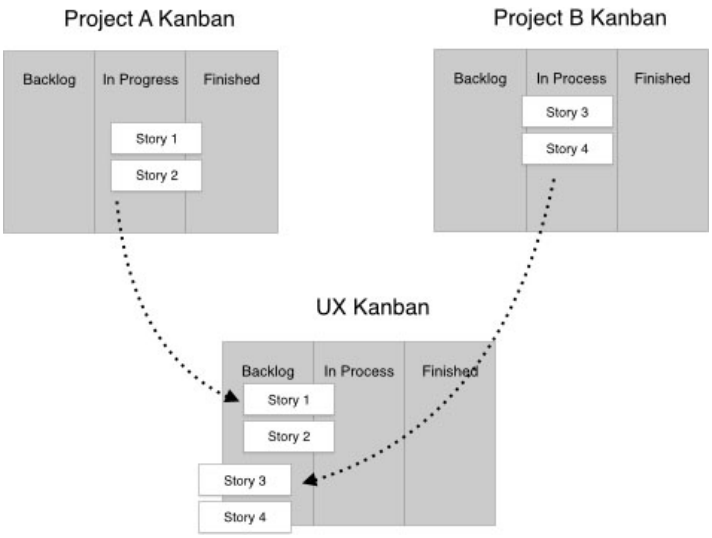


Figure 1 – UX Kanban board consisting of UX stories from other projects' Kanbans

To accomplish this, it made more sense to use a digital Kanban rather than the traditional analog method, so we could tag other teams' stories for *UX* in our Kanban. The tool we decided use for this is Jira's Kanban board. Because we were already using Atlassian collaboration tools in our shop, we could easily import projects' existing boards into our own. In truth, though, any system that can import stories from one board to another would do – even something as simple as an Excel spreadsheet.

Grooming and Prioritizing the Backlog

All the awesome tools in the world don't mean a thing if you don't plan and groom your backlog. To do this, we follow a pretty traditional grooming approach, meeting before each sprint to capture the stories for which we need to create UX design solutions.

Where things diverge, however, is in the cadence of the sprint itself: in my shop, we do two-week sprints. However, no two project timelines ever align cleanly, so the answer was *not* to shoehorn our workflow into other team's sprints, but to create and follow our own sprints. We came up with this rule, an example of which is shown in Figure 2:

Rule 1: If the duration of a story exceeds the number of days remaining in a sprint, it has to wait until the next one.

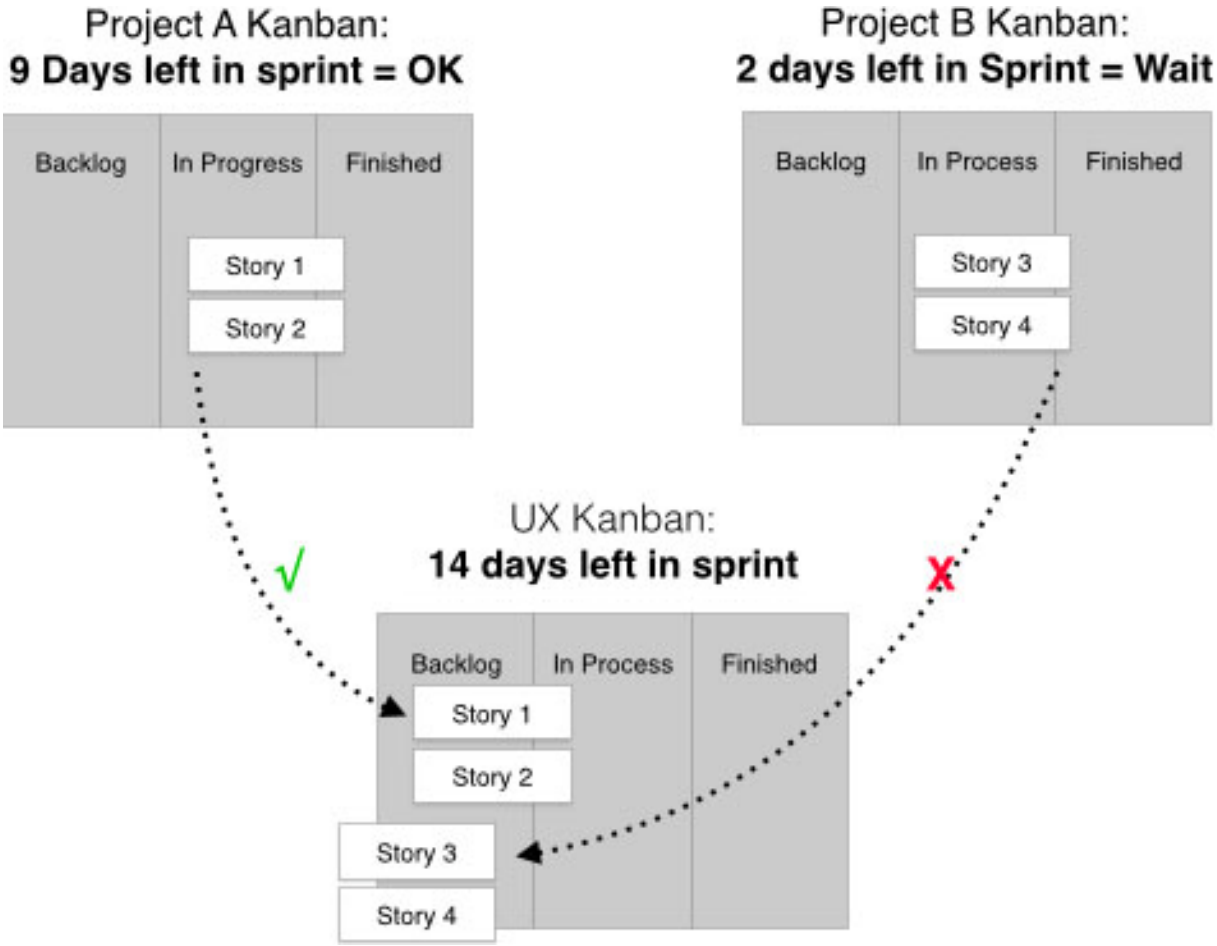


Figure 2 – Project A's stories fit in the current sprint, but Project B's do not

The reason we did this is fairly obvious: we wanted to control the workflow – *not* force ourselves to prioritize existing stories arbitrarily.

Planning It Out

Planning is also a standard part of the agile process. What is different about our planning sessions is that we invite product owners (POs) from *all* of the projects that are currently utilizing UX design resources to attend planning meetings. We hold these meetings monthly – covering two sprint cycles – to accommodate all of the stories they've given to us.

In the meetings, we review the stories from all of our backlog grooming sessions and place them onto the UX Kanban board. This brings us to our second rule, which is illustrated in Figure 3.

Rule 2: Only other Kanbans' in-progress stories can go into our Kanban's backlog.

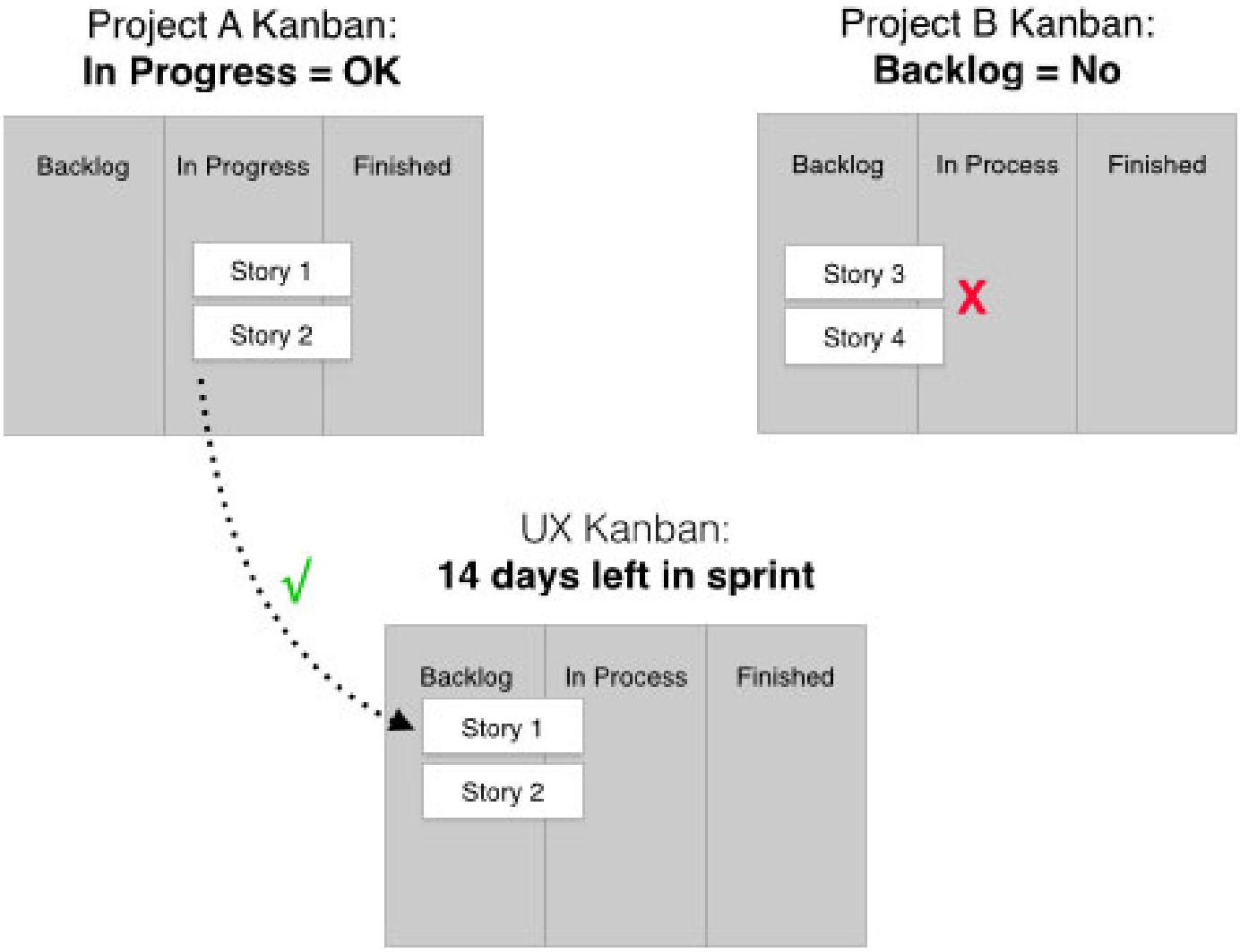


Figure 3 – Project A's stories are part of the current sprint, but Project B's are not

This may seem somewhat counterintuitive on the surface, because it might appear that there is no backlog, so *everything* is a priority. However, this is not the case.

Instead, we groom priorities during a planning session in terms of their absolute value relative to one another, placing the top three to five stories in the In Progress column. Then, we place the rest of the stories that we intend to complete before the next four-week planning meeting into our backlog.

So how do we determine the absolute priority of something? Two words: *horse trading*. This may not sound simple or elegant, but the reason we hold these large planning sessions with all POs present is because it lets us set priorities as a group. The UX team could easily do this in isolation, locking ourselves away for four weeks and deeming what stories are most worthy on our own, but this would contradict some very fundamental agile principles. So, instead, we opt for transparency and discourse and prioritize stories as a group This accomplishes two goals:

1. It lets us share UX resources fairly.
2. We can manage expectations for each sprint as a group.

Despite the rather messy appearance of this process, it has actually made us more efficient because we have to make the tough decisions together. But what if you hit an impasse?

Rule 3: You need a tie-breaker when you hit a prioritization impasse.

Yes, we do need a referee from time to time. This person should be someone who has some sort of role that straddles all of the projects in

question because they need to have both impartiality and a working knowledge of what we're prioritizing.

In our shop, a product marketing person who has a stake in nearly all of our projects is the ideal referee, taking into consideration that person's temperament, prior knowledge, and understanding of our products' challenges. But anyone who has some impartiality and familiarity with the projects could work. For instance, a scrum master from an external project could serve in this role as well, provided you give them the proper knowledge download.

Maintaining Office Hours

It is important to designate a regular time for discussing progress, doing impromptu grooming of a story, or following up on a question. Since four weeks would be a long time between stakeholder meetings, it is important to designate a regular time for discussing progress, doing impromptu grooming of a story, or following up on a question. Which brings us to Rule 4:

Rule 4: Maintain UX office hours to address product teams' ad-hoc needs.

For our team, this just means booking a conference room for one hour every Wednesday and waiting around for product owners to drop by. Sometimes, we spend just 15 minutes working with one product owner, but in other cases, we go overtime and have to book another

room to handle the demand. However, in general, spending just an hour a week to touch base and discuss whatever needs discussing is very helpful. Obviously, in your shop, this might not work perfectly right out the gate, but with adequate collaboration and finesse, it will work.

Conclusion

Although this process is a work in progress for us, we have been able to use it across several projects with promising results. I feel confident that we'll continue using it, perhaps making some small modifications as we go.

For many UX folks, it's easy to grouse about having to fit UX design into an agile development process, but it is important to remember that the reason agile exists is to facilitate communication and collaboration. These are principles that UX values as well. While UX design and agile may not be a perfect fit, marrying the two can deliver some outstanding results if you work at it. Just be mindful of the rules I've provided in this article:

1. If the duration of a story exceeds the number of days remaining in a sprint, it has to wait until the next one.
2. Only other Kanbans' in-progress stories can go into our Kanban's backlog.
3. You need a tie-breaker when you hit a prioritization impasse.

4. Maintain UX office hours to address product teams' ad-hoc needs.

If you decide to give our system a try, please be sure to drop me a line in the comments and tell me how it's going for you and what you've tried to improve the process.

Originally published on UX matters.

Agile Development Is No Excuse for Shoddy UX Research

By Atul Handa and Kanupriya Vashisht

Agile development and UX design are like a couple in an arranged marriage – a relationship between two strangers who are expected to coexist, develop trust and respect, and eventually, love each other. Throw UX research into the mix and you have the makings of an even more awkward alliance, as you can see in this typical conversation between a UX designer and a product owner, somewhere in the middle of Sprint 0:

> *Product owner: "Hey Jen, when can we see some wireframes?"*
>
> *UX designer: "Well, we're wrapping up our user interviews and putting together some personas – basically trying to get more clarity around our target users. We've already started on some sketches, but I expect we'll need to make some tweaks based on what we learn."*
>
> *Product owner: "That's all very good. But we can't afford the luxury of spending too much time on research. Sprint 0 ends next week. We can't keep the developers waiting! Let's speed things*

up. I'd really appreciate if you could get those wireframes going quickly?"

This is a familiar scenario across many agile UX teams. We're sure this story hits a raw nerve for many UX designers and researchers.

Whenever there's a time crunch in an agile setting, UX research is the first thing to get sacrificed – whether just watered down or cut altogether. The common perception is that UX research takes too long and should either be hastened or put on the back burner so User Experience can keep pace with agile development. Having worked on many different teams that have borne the labor pains of incorporating UX practices into an agile development–centric world, we've found UX research to be the most challenging piece of the puzzle – the piece that often gets brushed under the carpet!

Misgivings and Misconceptions

Whenever agile/Scrum teams discuss UX research, the following are some common refrains:

- "Who's got the time to fit slow, formal UX research into a fast-paced, agile/Scrum world in which teams expect to receive design deliverables quickly and at regular intervals? Rapid, informal, guerrilla research is the way to go."

- "We need to split all our work into specific, time-boxed user stories. Research is tricky to time-box because it may span not only a number user stories, but multiple sprints or even release cycles!"
- "We have to do our UX research up front, during Sprint 0 or even earlier. Once development kicks off, we won't have much time for research."

Alright. Let's clear up a few things here.

First of all, agile is *not* a fast way of developing software. No legitimate agile expert or literature relating to agile development would ever suggest that the goal of agile is to speed things up. Perhaps this common misperception stems from the nomenclature – *agile*, or *sprints* in Scrum.

In reality, agile/Scrum is *not* about speed, but quality. Time-boxing development efforts and working in iterative cycles, ensures teams don't stray too far in the wrong direction for very long. It also enables teams to make quick course corrections if requirements or priorities change.

There's no reason why these principles shouldn't apply equally to UX research. But unfortunately, teams often fail to get that, so they end up devoting little or no time to research, especially later in the development cycle. Agile becomes an excuse for doing poor research or not doing any research at all.

Cobbling rapid, informal UX-research efforts together, in an attempt to cope with agile development, results in unreliable and potentially misleading research outcomes. Granted, some research is better than none. But there's a very thin line between informal and sloppy.

On the other hand, how is doing a lot of up-front research useful if goals and requirements change down the line? Wouldn't this belie the main reason for adopting agile in the first place: to be better able to respond to changes and uncertainties?

The Heart of the Matter: The Backlog

The fundamental problem is that some think UX research has no rightful place in a sprint backlog. Agile/Scrum teams often view UX research as extraneous to development work. They consider development their key undertaking and first priority. UX research is perhaps a distant third priority, after design. The biggest obstacle to UX research routinely becoming part of the backlog is the unfortunate perception that it's a bottleneck that impedes the progress of design and development. Team members ask, "Wouldn't designers and, consequently, developers end up twiddling their thumbs, waiting for time-consuming research to take place?"

Well, yes, they might *if* your conception of agile is actually just a series of mini-waterfalls. If your sprints consist of linear hand-offs from research to design, then design to development, research would cause a holdup. But if your Scrum team works with a true agile mindset, it won't.

Here's an example: Let's say an agile team is in the early stages of designing a jobs Web site. UX research is under way, but nowhere near complete. The personas are still sketchy, and user requirements are not fully fleshed out. All the team currently has to go on are assumptions about who the users are, what their needs are, and how to best serve those needs, for example:

> *Our Web site caters to university and college students who are looking for part-time jobs to fund their education. The assumption*

> *is that they are interested in short-term work, with low barriers to entry. The competitive Web sites are too generic, don't cater to this specific job market, and fail to help users automatically match their profiles with potential job opportunities.*

What should the designers and developers do while waiting for the results of UX research? Instead of just waiting and complaining about UX research holding them back, the designers could begin interacting with researchers at the very outset of a project. At the very least, they could gain insights into the team's basic assumptions and hypotheses. Or better still, the designers and other team members could participate in some of the research to learn about users firsthand. Designers could simultaneously work on some preliminary sketches, basing them on whatever they've learned from the research so far – no matter how rudimentary. If the developers got those sketches right away, they could start working on their software architecture and code, implementing those sketches – even though they're likely to change later on. Remember, agile welcomes change. If everyone is working in parallel, no one is just waiting.

Of course, working in this way requires a high degree of open communication and collaboration on the team – another basic tenet of agile!

What if UX research were later to reveal that the primary user requirement is not merely finding jobs, but finding jobs within walking or biking distance of their home because very few members of the target user group own cars? The team could quickly add new user

stories to the backlog to cover additional features relating to job search by distance and location.

There's really no need to do big, up-front UX research; no need to wait for the research results; and no need to take shortcuts in doing research either. But you must consistently add UX research and any actionable items resulting from it to the backlog, prioritize the research, and undertake research work like any other work that occurs during sprints.

When planning a project, you must treat all work without distinction – whether research, design, or development. Encapsulate research in backlog items – epics, user stories, or tasks – and ensure that the team completes the highest-priority work, irrespective of whether it relates to research, design, or development.

Agile teams need to become truly interdisciplinary, not just in their composition, but also in their disposition.

Planning Your Research

Like design, UX research should ideally be iterative. While it may make sense for certain types of research such as evaluative research to iterate over shorter time spans – as Krug suggests, test early and test often – other types of research such as inquiries into sustained user behavior or changing trends might span relatively longer periods. Before making UX research part of a team's agile cadence, it is helpful

to categorize your research, then see how you can best time-box it to fit into the backlog. To that end, we suggest organizing UX research into three tracks, as shown in Figure 1:

1. Strategic track
2. Tactical track
3. Validation track

Figure 1 – UX research tracks

1. Strategic Track

The strategic track comprises any generative research that can help you shape the product vision, strategic goals, and product roadmap and identify the right target users. Research on this track is usually ongoing and exploratory. Typically, you'll begin this kind of research early, prior to design and development, then continue your research over relatively extended, iterative cycles, spanning months.

2. **Tactical Track**

   The tactical track comprises short-term, exploratory research that helps you define specific product features and desired experience outcomes. It helps product owners prioritize the work in the product backlog. You can do this research iteratively, within the timespan of a sprint, which is generally just a few weeks. This research helps you determine what features to include, modify, or sunset in any given release.

3. **Validation Track**

   As the name of this track suggests, validation entails evaluative research and testing, which proves or disproves assumptions, validates design decisions, determines any usability and accessibility issues, and measures user satisfaction and emotional response. Ideally, you should do research belonging to this track during every sprint cycle – which are usually two to three weeks in duration. This research could include usability testing, cognitive walkthroughs, or heuristic evaluations.

Of course, the names of the tracks we've suggested are generic. You could label or categorize them differently to suit the dynamics of your project. The purpose of these tracks is to help you better estimate and organize your research effort.

Adding UX Research to the Backlog

We've seen as many variations of a product backlog as the number of agile teams we've worked with. Each team uses slightly different labels and hierarchies for backlog items, as well as ways of structuring and organizing them.

Most commonly though, a Scrum product backlog comprises work items in the form of epics, user stories, and tasks. *User stories* – or simply, *stories* – are user requirements that a team can realize within the span of a sprint. *Epics* are stories that are too large in scope or complexity to complete them within one sprint. *Tasks*, the smallest work unit in the backlog, are activities team members need to accomplish to complete user stories.

There are some formal approaches that can be helpful in incorporating UX-research and design work into product backlogs – for example, Jon Innes's UXI matrix or Jeff Patton's user story–map backlog. These are definitely worth looking at. But, if you'd like to keep things simple and avoid adding too much overhead, you can just organize your research into the buckets, or tracks, we suggested earlier: Strategic, Tactical, and Validation.

For backlog items that are likely to extend over a longer period of time, follow these steps when adding them to the backlog:

1. Add a UX Research epic to the backlog.

2. Later, during backlog-grooming sessions, gradually start breaking the epic down into UX-research stories, which are essentially parts of the larger research effort that you can accomplish within a single sprint.

Adding Strategic-Track Backlog Items

Backlog items that fall under the strategic track are likely to extend over a longer period of time and, thus, necessitate your first writing a *UX Research* epic that you'll later break into user stories. Be mindful that you should be able to time-box one or more of these stories into a single sprint.

UX-research stories in the strategic track could be about formulating research questions, formulating a hypothesis, creating a screening questionnaire, recruiting participants, conducting a first round of interviews, or compiling and analyzing data, for example. You get the idea.

A UX-research epic or story does not differ greatly from other epics or stories, except that you should write them from the perspective of the *consumer* of that UX research – for example, the design team, product strategist, or business leadership. In other words, they must highlight *need*, *intention*, and *outcome*.

Here's a template you can use when writing a UX-research epic or story:

> *In the context of <a scenario/situation>, the <consumer of the research> wants to observe/understand <a phenomenon> or get an answer to <a research question>, so that <the desired outcome>.*

An example of a UX Research epic in this format might be:

> *In the context of a lower-than-anticipated conversion rate of 10%, the product strategist wants to understand why users are failing to complete registration and get through the profile-setup steps, so she can develop some strategies to encourage users to complete the registration and profile-setup process.*

Alternatively, you can use the following example template for each of the UX-research stories within an epic:

> *In reference to <research epic #>, we want to <subset/step of the research>, so that <the desired outcome>.*

Here's an example of a UX-research story that's part of our example epic:

> *In reference to research epic #123, we want to conduct contextual interviews with six participants so we can discover any issues with the registration and profile-setup process.*

Adding Tactical-Track Backlog Items

For backlog items you've identified and placed under the tactical track, if you expect the research to extend beyond the length of a sprint, you'll again need to create a UX Research epic, then break the epic down into research stories. But, if you can time-box the research into only one sprint, just create a single UX-research story, using the template for UX Research epics.

Here's an example of a standalone UX-research story:

> *In the context of tree-based location selection on the Job Search page, the product owner and design team need to know whether it is overly complex and would result in user confusion and a suboptimal user experience, so, if necessary, the design team can explore ways to improve or replace that feature.*

Adding Validation-Track Backlog Items

For backlog items under the validation track, if you'll need to validate more than one user story – for example, conduct usability testing relating to the design and functionality of multiple features that you've specified in various user stories – you may want to add a single UX-research story to the backlog, using this template:

> *When the <user type> views or interacts with the <page|section|-component> to <task>, we want to <observe/understand/record*

something> with respect to <criteria >, so <the desired outcome>. Refer to: <Story # or Stories #, #, #>

Here's an example validation-track story:

> *When a registered job seeker views or interacts with the Job Search page to search by location, we want to observe whether the user can interpret the search results and understand which jobs are within walking or biking distance, so the UX and visual designers can determine whether the format and visual design of the search results are optimal. Refer to: Story #123, Story #456.*

Alternatively, if you need to validate only one user story, we'd suggest that you instead add it as an acceptance criterion. Acceptance criteria are statements that accompany a user story and must be true to consider the story complete.

For example, consider this user story:

> *As a user who has signed in, I would like to have a way to easily change distance units from miles to kilometers, and vice versa, so when I search for jobs, I can view the distance to potential job locations in the units I prefer.*

One acceptance criterion for this story could be a validation criterion, for example:

> *A test user, when asked to change distance units, can promptly locate a link to **User Preferences** and successfully change the units from **Miles** to **Kilometers**, or vice versa.*

Of course, these are just suggested epic and story templates. You can tweak them for specific research items, according to your needs.

Prioritizing and Planning UX Research

Once you have incorporated UX research into your product backlog, prioritize the research in the same way you would any other stories. Backlog grooming should include time for going through the UX-research stories with all relevant stakeholders and fleshing out any additional details, stakeholder expectations, acceptance criteria, or dependencies. The product owner should determine the return on investment (ROI) of the UX-research items and prioritize them in the backlog accordingly.

During each sprint-planning meeting, discuss and estimate the UX-research stories with the entire team. Initially, it may be a challenge for team members who aren't UX researchers to vote on story sizing for research stories. But you should encourage them to make a guesstimate and discuss their vote with you, so you can help them to develop a better understanding of UX-research activities over time and to become more engaged with UX research.

Once team members have assigned the UX-research stories to themselves, encourage them to add *tasks* – definite steps they'll take toward completing the stories. They should also report their progress on these tasks during standups.

A Few Agile Research Tips

Make your research findings easily consumable. In the spirit of agile, don't rely on heavy documentation in reporting your research findings. Research results that get trapped in 100-page PDFs or on team wikis will likely never see the light of day, so won't have the desired impact. It is important that you deliver your research findings directly to the product team and stakeholders, during work sessions that accommodate discussions and dialogue and result in concrete actions and logical next steps. A good forum for this discussion might be a sprint review or demo session at which all team members and stakeholders are present.

Conclusion

UX design does not happen in a vacuum, but stems from context, which comprises specific business needs, user aspirations, and technical constraints. UX research helps designers gain an understanding of that context – of what they need to design, for whom, and why. Research can also help validate a design and prove or disprove a team's assumptions.

Some key outcomes of good UX research include, but are not limited to, gaining a competitive edge; improving customer acquisition, user engagement, and retention; getting a better return on investment; and reducing time and financial investments. In other words, UX research delivers all the same benefits as agile. Thus, UX research and agile are different means to the same end. So this arranged marriage could actually work!

Originally published on UX matters.

When Agile and User Experience Click

By Todd Zazelenchuk and Jeff Larson

Try this test: Find three UX friends and ask them about the compatibility of UX design with agile development. Odds are that one of them believes UX design and agile can work well together, one swears that they can't, and one has yet to decide. There are many reasons for the divided opinion on this issue.

In the same way that consumers' experiences are highly contextual, no two software development teams are the same. They may vary in their composition, experience level, the proximity of their members, and their organizations' willingness to embrace agile development methods. To be successful, it is necessary to apply strategies that best serve our team's context. In this article, we share our recent experience trying to make UX design and agile click by examining the design and development of the Android app Find MyHeadset™ and highlighting the strategies and activities that we found most helpful.

Find MyHeadset

We design software for Plantronics, a global leader in high-performance, audio communication devices. One of the company's most popular product categories is the compact Bluetooth headset. While these devices vary in their styling and feature sets, they come in a single size: extra small. Not surprisingly, they are easy to misplace. In May 2012, Plantronics formed a team to address this common problem.

Find MyHeadset is a mobile app for the Android platform that helps Plantronics customers locate their missing headsets. The app gives people two methods of finding their lost headset, as shown in Figure 1. If a missing headset is powered on and in range of its paired smartphone, a user can have the headset emit a tone to reveal its location. If a missing headset is powered off, out of range, or has no battery charge, the BackTrack feature lets users retrace their steps by displaying the most recent headset events and their corresponding locations. After four months on the market, customers have downloaded Find MyHeadset over 40,000 times and awarded it a 4.5-star rating on the Google Play Store.

The Find MyHeadset project was the first official agile initiative at Plantronics. The eight-member team was geographically distributed, with User Experience, Product Management, and Quality Assurance (QA) located in Santa Cruz, California, while Development and additional QA team members were nine time zones away in Eastern Europe.

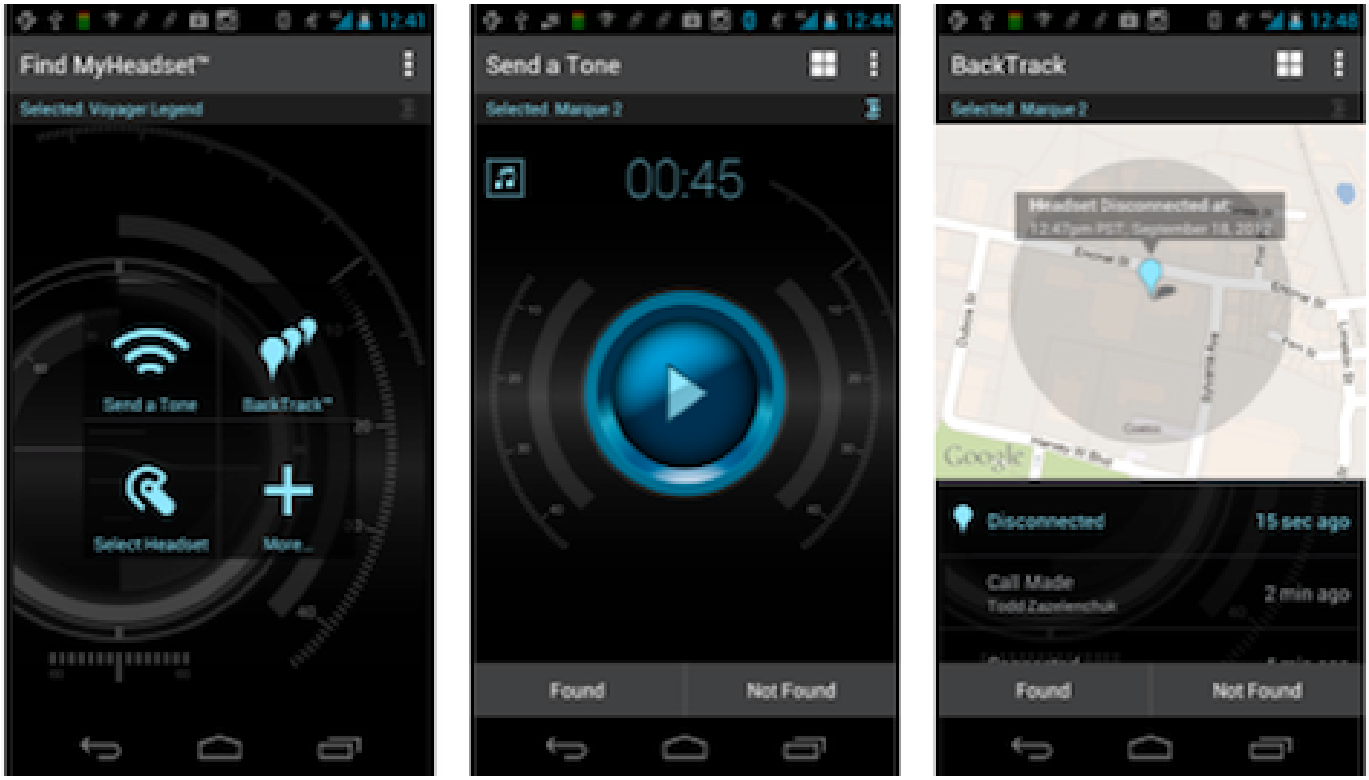


Figure 1—Find MyHeadset app helps people find a missing headset

Team members' experience with agile varied, ranging from extensive to none. The team adopted a Scrum approach and executed two – to three-week sprints to develop the application. We completed development in four months, launching Find MyHeadset on September 17, 2012.

Advice Worth Heeding

Both prior to and during the Find MyHeadset project, our team benefited greatly from advice and experiences shared by the global UX community. In turn, we'd like to share the strategies and activities that contributed to the success of Find MyHeadset in the marketplace.

1. **Follow Jeff Patton's 12 Best Practices for Agile UX**

   Jeff Patton's 12 best practices[1] should be required reading for any agile UX team that wants to sidestep the inherent challenges of designing in an agile development context. On the Find MyHeadset project, our team benefited from two of these best practices in particular:

   - **Use a RITE (Rapid Iterative Testing Evaluation) approach to iterate on concepts prior to development.** Although planning and implementing this approach can be challenging, its benefits make the effort worthwhile. For three of the five sprints on the Find MyHeadset project, we dedicated a day to usability testing concepts for future sprints. More than once, the results from these rapid tests led to significant changes that directly improved the user experience of the final product.

Figure 2—Creating sketches

   - **Prototype in low fidelity.** Low-fidelity prototyping was critical to our conducting effective evaluations of our design patterns and workflows. By beginning with sketches like those shown in

Figure 2, then progressing to mobile wireframes in a tappable PDF format, the Find MyHeadset team was able to collect highly relevant usability data that helped shape the architecture, functionality, and content of the final app.

2. Create a Holistic UX Workflow for an Application

Both Nielsen[2] and Gothelf[3] emphasize the importance of a UX team's developing and maintaining a holistic vision for an application. Without a clear vision, an agile project can quickly devolve into an incoherent collection of features and functionality that are driven solely by the prioritized user stories for each sprint.

For Find MyHeadset, we created a UX flow diagram to map out the core elements of the application and communicate our vision. While such a master plan may seem counter to an agile process, this holistic view helped the team understand what we were creating and how things were supposed to fit together.

In an agile development context, the key to creating an effective UX workflow is to keep it lean and incomplete, as shown in Figure 3. Focus on the portion of the workflow that supports the current sprint's user stories without losing sight of the overall vision. In our case, each iteration saw the flow diagram expand and adapt to reflect the parts of the app that we were targeting during a specific sprint. Because we updated our flow diagram regularly throughout the entire development cycle, it served as an evolving blueprint that QA could rely on to conduct its testing.

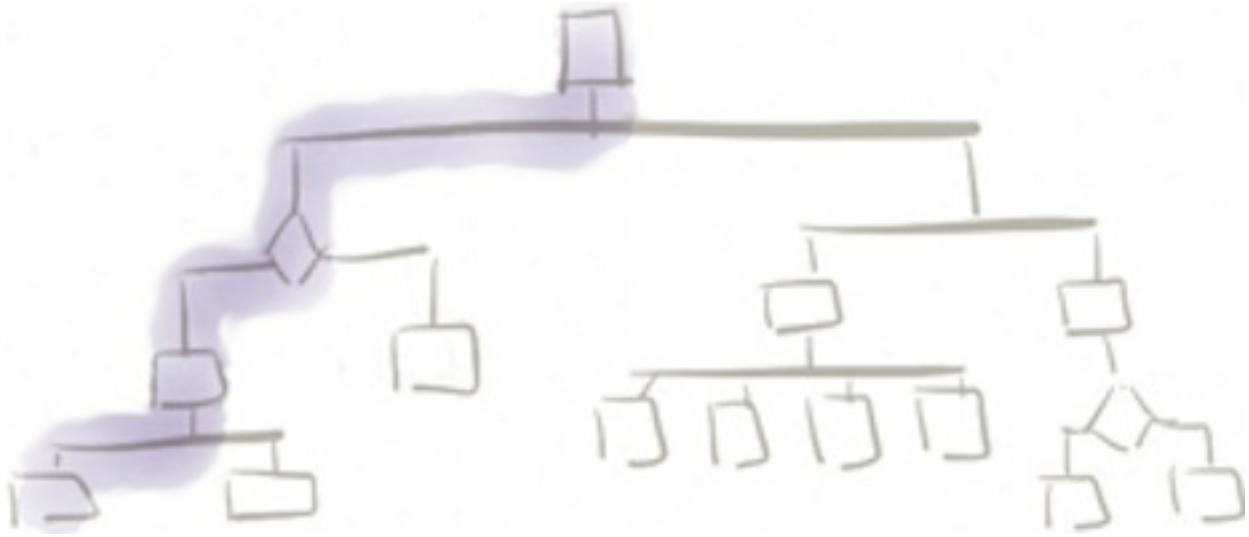


Figure 3—Maintaining a lean UX workflow

3. Get Ahead of Development During a Sprint Zero UX Design Phase

Creating a holistic UX workflow and vision for a product doesn't happen overnight. Depending on the scope of a project, it can require significant time at the front end of an agile project[4, 6]. Some people argue that a Sprint 0 design phase simply amounts to a waterfall approach, in which the UX team completes its design of an application up front. Others acknowledge the value of a Sprint 0 design phase, but question just how far in advance User Experience needs to design prior to embarking on a first sprint.

On the Find MyHeadset project, prior to the first sprint, the UX team dedicated two weeks to ideating on design concepts, exploring workflows relating to the app's two primary features—Send Tone and BackTrack—and conducting internal design reviews. This effort permitted team members to effectively plan the first sprint and establish a design framework that provided a foundation for detailed design during subsequent sprints.

4. Stay Ahead of Development by Dedicating Time for UX Design for Future Sprints During the Current Sprint

Agile planning tools can help UX teams stay ahead of Development, but UX designers must ensure that they attend to designs for future sprints even when the current sprint is demanding most of their attention. On the Find MyHeadset project, we earmarked 30–40% of each sprint for detailed design and testing of user stories for future sprints. Ideally, we would have allotted even more time. More than once, our cushion evaporated, and we failed to stay ahead of Development. Had the project been any larger in scope, we would have needed to implement a design spike[5] to get ahead of the Development team once again.

5. Partner Closely with the Product Owner

User stories are at the heart of an agile process. They drive what a product is supposed to be and do to meet the needs and goals of target users. Product Owners usually write the majority of user stories and own the responsibility for prioritizing them. By collaborating well with the Product Owner, a UX designer can directly influence the user stories and, in turn, help define the product.

According to Patton[1], "In the best teams, the UX folks have an active hand in deciding what is built, the overall business strategy that drives what's built, and the tactical prioritization of work done first. In some successful agile organizations, the UX team *is* the agile customer team or product owner team." On the Find MyHeadset team, User Experience and the Product Owner regularly

worked together to groom the backlog of user stories and ensure that their details and acceptance criteria were compatible with the holistic design vision for the app.

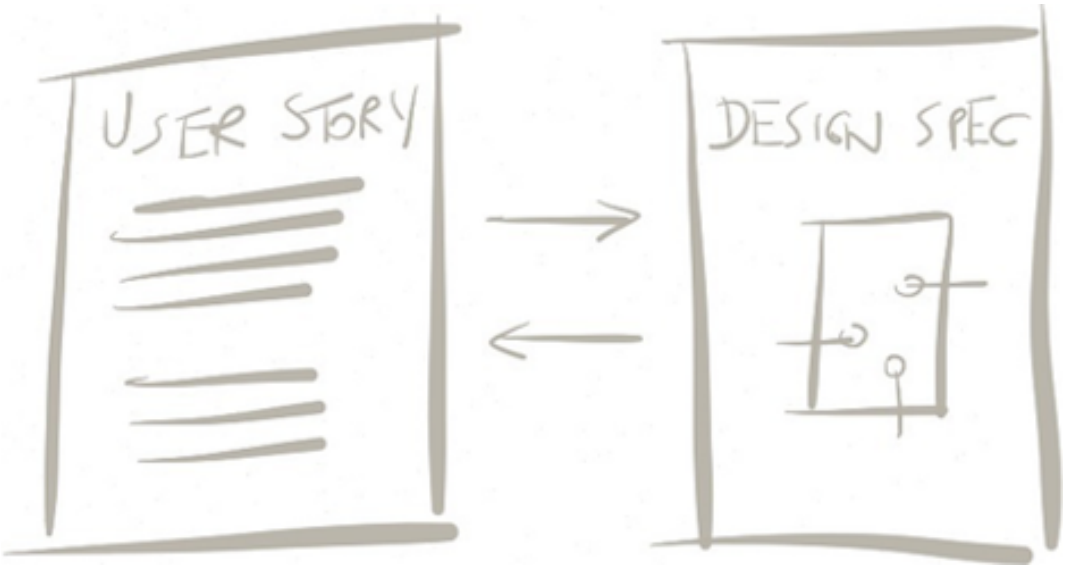


Figure 4—Collaborating on user stories and UX design specs

6. Schedule Dedicated UX Reviews with the Team as Part of Sprint Planning

When agile teams are entirely colocated and members are working in close proximity, they can communicate their ideas and solutions without additional effort. For distributed agile teams, however, you must take extra steps to avoid communication problems and delays. For the Find MyHeadset project, a regularly planned review of the UX design spec prior to each sprint planning meeting quickly emerged as a requirement for success.

The spec review helped the team visualize the stories that we had prioritized for the next sprint and understand the level of effort the proposed solutions would require. It also provided team members with an opportunity to give us their input and ensured that

we gave them the extra details they needed to identify relevant tasks and make accurate work estimations.

7. Determine the Appropriate Level of UX Design Documentation for Your Team

A common misconception about agile development is that documentation should be eliminated. This is especially untrue for projects with distributed teams, where members must make independent progress asynchronously. While it may be possible to *reduce* the formal documentation a project requires, it's still necessary to record detailed interaction design decisions to ensure clear understanding among team members.

Our perspective on this is counter to Gothelf's[3] view that a prototype *is* the documentation on an agile project and anything more is wasted effort. On the Find MyHeadset project, adopting an "it is what it is" approach to documentation would have made it very difficult to build and test the app effectively.

Mature organizations such as Plantronics often have intensive QA programs that rely on documented design details to build their testing protocols. From the outset, our team chose to follow Govella's[6] advice to implement well-annotated wireframes instead of producing heavy documentation. Publishing our specifications online made them easy to update and distribute to all team members. We simply emailed team members a link to the current version and provided a clear revision history.

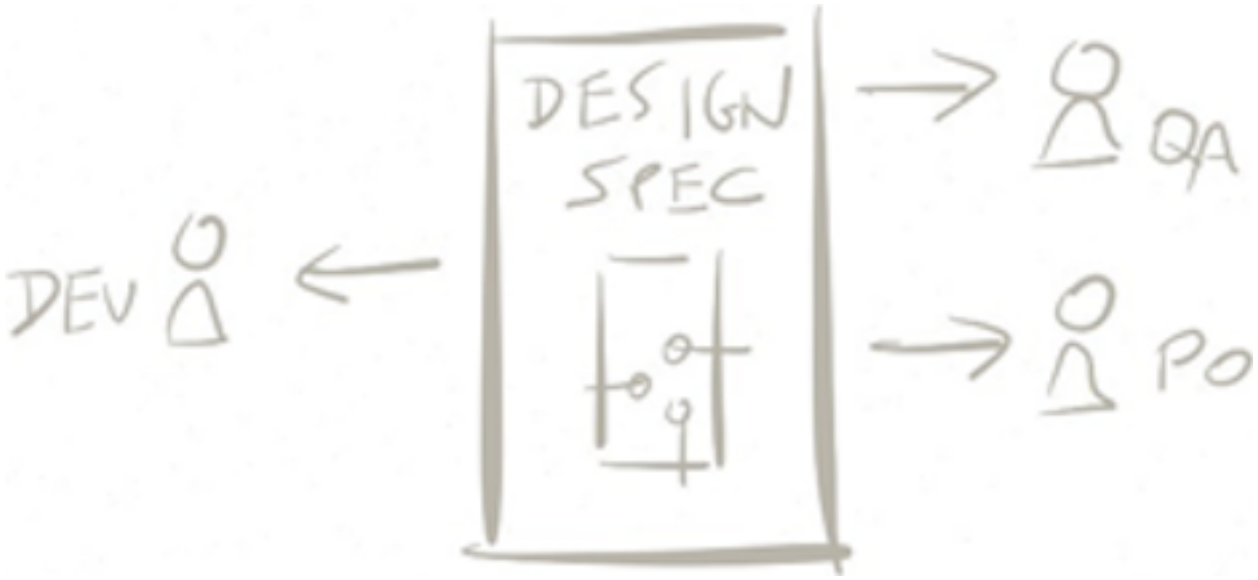


Figure 5—Providing specifications for our distributed team

8. **Extend Your UX Team to Include Front-end Development**
   Team members who can play multiple roles on an agile project help increase the speed of iterations. When a UX designer can work directly with the code to complete front-end development work, Development team members can take on other tasks.

   On the Find MyHeadset project, User Experience took responsibility for much of the presentation layer by working directly on branched code once the Development team had built the foundation. This permitted us to modify, review, and iterate on the design by creating our own builds and reviewing the latest code on demand. For a distributed team, even daily builds can be too infrequent to ensure rapid progress, so having User Experience assume some degree of ownership over front-end development can help a team to achieve a higher-quality result in less time.

More Work Ahead

On the Find MyHeadset project, our team discovered multiple strategies that made UX design and agile development click. However, we also identified several challenges with which we continue to wrestle:

- What is the best way to effectively manage certain UX roles such as visual design and user research that tend to experience peaks and valleys of intensity during a sprint?
- How can we optimize collaboration and productivity among the members of our distributed agile team?
- How can we meet our team's desire to be lean and take risks within a culture that prides itself on avoiding failure through detailed execution and testing?

We need to address these challenges and others before we can realize the full potential of agile development. As the UX community shares more examples of successful projects like Find MyHeadset, we'll all become more confident—and perhaps less divided—about the compatibility of UX design and agile development.

Thanks to Paul Bryan and Philip Hodgson for reading an early outline of this article.

Originally published on UX matters.

References

[1] Patton, Jeff. Twelve Emerging Best Practices for Adding UX Work to Agile Development. Agile Product Design, June 27, 2008. Retrieved December 9, 2012.

[2] Nielsen, Jakob. Agile User Experience Projects. UseIt, November 4, 2009. Retrieved December 9, 2012.

[3] Gothelf, Jeff. Lean UX: Getting Out of The Deliverables Business. Smashing Magazine, March 7, 2011. Retrieved December 9, 2012.

[4] Walker, Mat. About Agile, UX, and Sprint 0. MatWalker.co.uk, January 17, 2011. Retrieved December 9, 2012.

[5] Dimmick, Damon. Fitting Big-Picture UX into Agile Development. Smashing Magazine, November 6, 2012. Retrieved December 9, 2012.

[6] Govella, Austin. Agile + UX: Six Strategies for More Agile. Thinking and Making, May 7, 2008. Retrieved December 9, 2012.

Agile UX Case Study

Improving Design Sprint Length by 50%

Based in Seattle, LiquidPlanner offers a predictive project management platform to thousands of companies worldwide. Its enterprise customers include LinkedIn, Sumo Logic, and Redapt.

In this customer success story, you'll see how UX Designer Edward Nguyen used UXPin to help his team deliver better products faster.

Edward Nguyen
UX Designer at LiquidPlanner

Challenges

LiquidPlanner's product development process was losing efficiency when shipping on tight timelines:

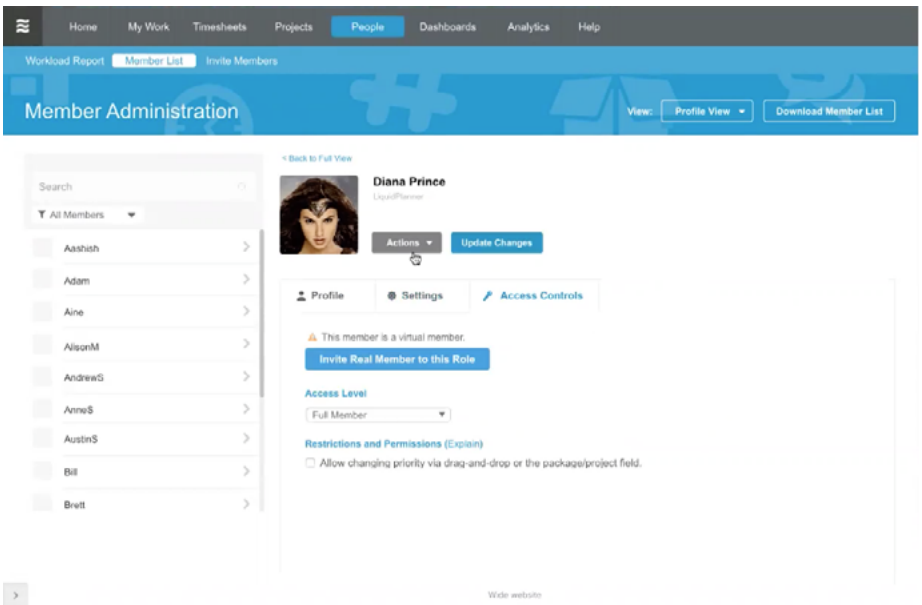
- **Inefficient development costs**. For a detailed prototype demanded by enterprise use cases, the design team sometimes required 4-8 hours of senior developer time.
- **Inefficient designer costs**. If developers weren't available, the design team might need up to 8-10 hours to code the prototype.
- **Inaccurate design tools.** ZURB Notable couldn't support the interactions demanded by enterprise products. Axure lacked the collaboration demanded by LiquidPlanner's Agile process.
- **Timelines slowed by documentation.** Lack of efficient prototyping required the design team to spend up to 8-10 hours before a sprint to work on feature specification with product managers.

"UXPin is a requirement for us. Notable, Adobe XD, Omnigraffle – none of those tools work for us." Edward says. "You absolutely need UXPin's pattern libraries and interactions to validate complex user scenarios quickly."

Solution

To bring the right ideas to market faster, LiquidPlanner turned to UXPin:

- **Accurate testing of complex interactions.** “The new UXPin engine and interface are awesome,” Edward says. “A user even mistook one of my hi-fi prototypes as the real deal, telling me to thank our dev team.”
- **Minimal documentation for acceptance criteria.** “QA now uses the UXPin prototype as acceptance criteria since they’re so real,” Edward says. “Before they’d either require heavy documentation or just test without criteria.”
- **Unlocks new solutions for product problems.** Because designers aren’t forced to code designs, they can prototype and test more ideas for a faster path to certainty.



LiquidPlanner created a UXPin prototype so powerful that users thought was already fully developed.

"It's amazing to confidently tell my team that I can validate a design in a few days" Edward says. "I can prototype on Monday, test it Tuesday and Wednesday, and show results on Thursday."

Results

Thanks to efficient collaborative design in UXPin, the team can achieve certainty with greater confidence and less cost:

Company Results:

- **Frees up UX development resources**. The design team can create accurate prototypes to test ideas with users without any code.
- **Saves development costs**. "I'm no longer stressed about coding," Edward says. Prototypes that previously required hours to code now only take minutes."
- **Cuts down design time by 50%**. Designers used to require 4-7 days with a developer (or more) to go from idea to tested prototype. Now they can confidently test ideas with users in 1-3 days.
- **Cut down documentation costs**. "When I showed two developers a 15-20 page spec document alongside a UXPin prototype, they said they didn't even need the documentation." Edward says. "They preferred the prototype as the specs. We don't need a ton of documentation anymore."

Product Results:

- **Launched a new enterprise feature in 4 months**. The team shipped a new dashboard feature prototyped in UXPin in less time (January to April 2016).
- **Increased speed of adoption.** Of the 17,000 dashboards ever created, 10% were created 2 weeks after launch with the feature prototyped in UXPin.
- **Increased user adoption.** 75% of new dashboards are now created with the feature prototyped in UXPin. A majority of high-revenue enterprise customers also enjoy the new feature, as it facilitates their complex projects.

"UXPin prototypes gave our developers enough confidence to build our designs straight in code," Edward says. "If we had to code every prototype and they didn't test well, I can only imagine the waste of time and money."

Jeff Veen, Design Partner at True Ventures

On Agile UX

You founded Typekit, lead teams at Google and Adobe, and now work in venture capital. How do you balance UX with Agile?

> I've seen a lot of different methodologies come and go, or come and merge.
>
> We adapted practices from Extreme Programming, Agile, and Lean Startup (which I think is an evolution of both of those ideas). We didn't follow "agile with a capital A", but just treated it as another inspiration for our hybrid process.
>
> The most steadfast "rule" was that every single person on the team is doing UX, whether they're a back-end developer, front-end developer, or designer. A Database Engineer is just as important to delivering the end-user experience as a Visual Designer.
>
> To make that hybrid process work, we found it helpful to merge the UX and development team. I just made one team and said, "You

> all have different roles towards making the best possible product, now let's follow a design process that works for all of us."
>
> For example, our daily standups weren't by the book at all. If you read the books about Agile, you'll find lots of discussion about tacking index cards up on the wall and moving things around. I don't really do a lot of stuff, but I love the spirit behind checking in every morning to gauge progress and morale.
>
> Once our teams grew, we scaled the format to accommodate size (sometimes 40+ people). You don't have ask every single person, "What did you get done yesterday, what do you want to do today, and what's blocking you?" Instead, you can appoint key people who have emerged as leaders in the organization to report out on these answers so the format is more "Here's what's happening in our company and team".
>
> Then, as a leader, you're better able to see if there's energy here or if you need to dig deeper into problems.

Was it hard getting buy-in for that more hybrid design process?

> It was definitely more challenging for companies that were still following a waterfall process.
>
> For example, you go to a place like Adobe, which is really entrenched in an org chart with separate disciplines and everybody is centralized around their discipline, and engineers with years of

> experience will say, "Would you just please give me a mockup, so I can go build it?"
>
> That can be really challenging. To get buy-in for a more hybrid process, I would always try to show progress by tackling smaller projects for quick wins. That way, others could immediately experience the improved process versus what happens when a design team is battling priorities from a department that has nothing to do with the current product.

What do you like most about those different methodologies that inspired your design process?

> Shifting the thinking so that every technical project must be rooted in an identifiable user need.
>
> The other part I liked is that Lean, Agile, and Extreme Programming are all structured around momentum. And that means going fast, doing small amounts of work, and launching as quickly as possible so you learn from the real world.
>
> That turns a design process into a much tighter prioritized set of goals. Instead of designing a complete architecture and launching all 150 features at once after 6 months, we launch two things next week and learn how to immediately improve.
>
> Once you've shifted that mindset, it's so much easier to follow a flexible process based on the project needs. Everyone ends up

> feeling good about their work, they’re getting constant feedback, and they’re shipping quickly as opposed to these death marches towards giant releases.

What pain points did you face in your hybrid design process?

> Flexible methodologies are always difficult to introduce to talented people who are already well established in their career.
>
> When I describe this hybrid collaborative process, a lot of people can believe that it sounds like a waste of time and we should spend that time coding. You need to build trust in the team to say, "Look, we're not typing at our computers for the next week, we're doing work that helps us understand the problems we need to solve".
>
> On the other side of the coin, going fast all the time with sprints can be frankly exhausting. It's like you're sprinting through a marathon. Leadership needs to regularly gauge the overall sense of fatigue in a team.
>
> Sometimes we needed to go a little slower and we planned a few sprints to pay back the technical debt we accumulated from moving so fast. You need those “payback” sprints because you want the team to feel satisfied with work that reflects their reputation.

For this hybrid design process you tried at Typekit and Adobe, how did you decide who was the product owner?

> We always designated one person as the product owner, but that person wasn't a pure product manager. I tend to look for product management skills in developers and designers or even user researchers. For the product management and product owner role, I find people are more successful if they first have a strong background in a design or technical role and they're shifting to this role for more influence over the product.
>
> At Adobe, a lot of the product management were MBA types. It's not a bad thing at all, but their method of working was around developing models, spreadsheets, and Powerpoint decks. Very little user research involved. They would examine problems like "What's the total addressable market, and how do we achieve that?" Those questions are totally important, but your method of achievement must be based in user-centered design.
>
> Therefore, in order to capture that market share, you need clearly designated product managers and product owners with a solid grasp of UX.

How can enterprise designers better sell the value of user research in an Agile process?

> You can use some sleight of hand in the beginning of a project, and say "We need to work on all these big things. We need to develop some kind of overall architecture. We need to choose a technology platform. Oh, and we need to do a bunch of customer research to

> make sure none of that goes sideways." You place user research in the same category of upfront work as technical scoping.
>
> You also frame the research in a way that generates revenue.
>
> When working at Google on their analytics and Adwords platform, I said "Look, we have to talk to people paying for AdWords and find out their reporting needs so that we can help them spend more money."
>
> As a result, we spent our first few sprints (a couple months) diving deep into understanding ad spending, online marketing, and publishing efficacy of customers.
>
> We met with hundreds of potential customers in the agency world and in-house teams at large companies. When you listen to even 20 people with the same job talk about their role, and you visit their office and you see the sticky notes on their monitor and all the other workarounds, you see a ton of trends emerge. You analyze the transcripts of all of these interviews and say, "This is a task, and this is a task they do, and this is a task." Group the tasks together and you've built a solid mental model and product architecture.
>
> All without coding anything or designing any screens.

On the other end of that spectrum, how did you fit user research into the the process you practiced at Typekit?

> Same approach. We did two large research sprints upfront to learn about web designers and people who design and sell type for a living, then continued to follow up throughout the process.
>
> In the beginning, my cofounders and I would spend 30 minutes at a time interviewing a total of 40 users (Skype, coffee, etc.).
>
> From those interviews, we learned about critical issues and had plenty of source material to adapt the rest of our design sprints.
>
> At Google and Typekit, I didn't write up a research brief with a protocol and a structured plan. I just told everyone "No, I should go talk to as many people as I can to figure out if I'm going crazy or not."

In this hybrid process, did you run post-mortems and retros on a regular basis? If so, what format did you find successful?

> We ran post mortems and retros all the time.
>
> That's incredibly important for freeing everyone up to be creative. You don't look for blame or credit, but simply "What did we learn from what we just did".
>
> For each post mortem or retro, someone on the team was tasked with investigating the situation and returning with insights. They'd share them with the rest of the team for alignment, and then we'd dive right in after the morning standup.

> The “investigator” would share the vetted results with the team, then we’d start an open discussion based on the 5 Why’s of Lean Startup to dive into issues discussed. We kept the post mortems and retros short and focused to prevent them from becoming a complaint session.
>
> And of course, during post mortems for projects that failed, leadership needs to identify the circumstances, not the attributes, of the people involved.

What would your ideal design process look like?

> I'm not quite sure I have a perfect process because there's no perfect team.
>
> The most important thing to remember is that a team is just a collection of humans. Humans have some combination of rational thoughts and emotional responses to everything. And every process must account for that particular blend of rationality and emotions.
>
> Here’s the three key elements crucial to any design process:
>
> - A deep understanding of the people on your team.
> - From that comes culture, a clear mechanism for communication that everybody buys into.
> - That culture encourages a sense of momentum in which everybody feels like progress is happening and they're thrilled everyday to see the product get better and better.

> The best process is a flexible process. Steal the best techniques from multiple methods, then adapt them to the context of your product and team.
>
> That approach always scales over time.

What collaborative tools do you think help facilitate Agile UX?

> I'm a huge proponent of any tool that facilitates collaboration and communication across teams. I've seen the power of this when working in highly productive teams who embrace tools like Slack or Trello or Asana.
>
> But what I've been most interested in lately are processes and tools that encourage collaboration around design artifacts – the files that designers use to communicate their interface solutions or interaction flows.
>
> As an investor, that's where I think UXPin really shines: it allows not just teams, but whole organizations to seemlessly work on designs without emailing around images and haphazardly collect feedback. It's super efficient and organized, and I think more and more Agile companies will be embracing this type of design-centered collaboration in the future.

Adrian Howard, Partner at Quietstars

On Agile UX

Have you seen any gaps between Agile processes and user-centered design?

> I don't think Agile and UCD are misaligned in intent. We only create gaps through malpractice and misunderstanding.
>
> People often say Agile is a very developer-focused set of practices. That's not entirely true.
>
> While Agile definitely originated from the developer world, the focus was very much on, "Let's stop building crap products. Let's start making our customer happier. Let's start listening to our customer. Let's have much tighter feedback loops. Let's deliver actual value." All those principles completely align with UX goals.
>
> You'll find a lot of stuff out there that's Agile in name only. It's the same as passing off rubbish design in Photoshop as UX work. There's

> a lot of, "We're going to jump on the latest buzzword because we hear it's good."
>
> But the company does'nt understand the underlying strategy behind Agile. So you get a lot of Cargo Cult Scrum, especially in larger organizations where people run the sausage machine at the development team as fast as possible, pushing functionally sound features out all the time without enough user validation.
>
> As far as I'm concerned, that's not Agile at all. That's just a superficial interpretation.

What is one of the top mistakes you've seen companies repeat as they practice UX in an Agile environment?

> Because a consistent stream of feedback usually pours in during Agile product development, some companies might assume usability testing and user research skills aren't required. The product team is already pushing incremental features live and hearing from customers quickly, so what more do they need?
>
> The fallacy is that unless your team has at least basic skills in user research or usability testing, you might hear about the problems from users, but you won't know how to steer the product out of those problems.

So how can Agile teams build up more UX competency to better act on user feedback?

Increase the whole product team's user exposure hours. Don't just rely on passive user feedback coming through customer support, or usability reports delivered by a researcher or designer.

In fact, if you look at the origins of Extreme Programming, the creator Kent Beck actually embedded a user into the product development for Chrysler's Compensation System (the test bed for his XP methdology).

Of course, that kind of Stockholm Syndrome-esque approach does come with potential bias since the customer becomes part of the development team. But still, even at that level, the whole team at least was in regular contact with the end-user.

Usability reports are easily ignored by development teams. However, if a developer attends even just 1 hour of usability testing a week, the feedback hits home immediately. People will instantly believe and understand a designer's recommendations because they've experienced the visceral power of seeing a poor old end-user having a horrible time with the product. You don't need as much documentation to convey your point.

Try an incremental process of drip-feeding your exposure hours. Every other week spend a few hours with developers interviewing users and moderating usability testing. Incremental research provides more value because people apply insights by the next sprint.

> Spreading out your user research is an easier sell to organizations than trying to convince them to spend $60-70,000 on a month's worth of user interviews all at once. You spend less upfront, show results quicker, which earns more support for thorough research.
>
> You spend much less on user research upfront, and you gain buy-in more quickly since you can show the problems you fixed. Plus, you're constantly steering the product in the right direction as you uncover new customer needs since the time you started the project.
>
> I have a cliche slogan which is: "Do less more often together to do more".

Do you think component-based design helps with UX in Agile processes?

> It certainly relieves many pain points.
>
> For example, I run this workshop exercise where I show wireframes of popular sites to developers, and designers. I give everyone a few minutes to write down their observations. Developers will say "These are the elements of the application, this is the form X1 for next the page, this is the edge case, etc". Designers will talk about vertical rhythm, typography, and overall interaction flow.
>
> Everyone sees a different part of the elephant. Component-based design helps you reduce that misinterpretation and resulting approval bottlenecks.

> When you have a shared design language that breaks down into universal UI patterns and elements, designers no longer need to create everything for signoff. The whole team now has a toolbox for exploring new ideas or refactoring the existing UI.
>
> As a result, the product team's conversations are much more strategic.
>
> Instead of developers wading through minutaie when asking a designer to recreate a whole page, everyone knows how the pieces fit together. We know the forms we should use and the interactions we should support. Your sprint release quality improves since developers with little design background can implement mockups and prototypes with less risk of inconsistency.
>
> By democratizing the design implementation, the designers are then freed up to focus on the more difficult business problems. You get to "done" faster and better.

Does the fast pace of Agile UX increase risk of experience debt?

> Only if you focus purely on speed of release.
>
> Agile is not just about delivering faster. In fact, the more crap you release, the more crap you need to wade through, and the slower you'll become anyways.

The issue of UX debt in an Agile process all depends on your organizational values. Does the company care more about keeping the development team at 100% capacity, throwing out tons of features, or about delivering value?

Because if it's the former, then nobody is really incentivized to dedicate time to "payback sprints" to clean up as they go.

You need to develop against a hypothesis (like Lean UX advocates) and prioritize your work according to the Kano model, otherwise you will inevitably create unmanageable UX debt.

Going back to my earlier point, you need collaborative incremental user research to keep your Agile sprints in check. When the product manager and developer sees five people in a usability testing all fail miserably to register for an account, the designer doesn't need to fight as hard to justify an iteration sprint versus being forced to dive into the next feature.

When an experimental strategy drives the Agile UX process, you can better remove the danger of people sticking to their plans for fear of losing face.

Laura Klein, Principal at Users Know & Author of Lean UX for Startups

On Agile UX

How do you see Agile and UX working together?

> Agile was created for engineers, so it originally didn't address UX – and honestly, that's kind of fine. We've learned to adapt.
>
> The saying is that "Lean helps you build the right thing. Agile helps you build the thing right."
>
> Lean Startup is about constantly making sure that you're working on things that customers will use. Agile is about delivering those things to them quickly and efficiently. They're both about helping you learn quickly and adapt.
>
> I would argue that any kind of user-centered design contributes to all of that, especially the learning. If you use all three of them together it's easier to build the right thing, and it's easier to build it the right way, and that is helpful.

> But I'm not religious about it, the only thing I don't love is building things without talking to users.

Could you help walk us through how you've seen that Lean/Agile/UCD hybrid process play out successfully in projects?

> The Lean Startup process is very much about understanding what you want to build by testing your assumptions with customers. You'll see that overlaps with UCD methodologies, since customer development is a form of user research. We've been observing users in the UX community for many years already, so the two work very nicely together.
>
> On the other hand, Agile is a way of organizing your engineering team to release features faster and deliver customer value more quickly.
>
> I'm a big fan, not just of the Lean and Agile philosophy, but also the related discipline of Extreme Programming which includes things like test-driven development, pair programming, and continuous deployment.
>
> Continuous deployment helps deliver the hybrid process - Lean Startup, Agile, and User Centered Design -because we can be talking to users, observing them, studying their problems, coming up with solutions for them quickly and getting them out in front of users and then measuring the results within a very short period of time.

This can be a very Lean Startup, hypothesis-driven process: "If we make this change, we predict it will have this effect. Now let's test to see if we were right or wrong." A great advantage, of course, is that we can complete the whole process from hypothesis to validated solution in a matter of hours or days rather than weeks or months.

The key is to always drive your process based on a clear hypothesis. You want to design a small complete thing (not an under-designed thing) as a quick experiment. You can then use Agile processes to break down that complete thing into bite-sized tasks for engineers.

If you involve engineers in the user research, you also gain a greater shared understanding of the problem, which always helps with efficiency. So we come up with our experiment, build it, ship it, measure it, and all of a sudden we've gathered meaningful results in a very short period of time. That's a great cycle to follow, and how I'd combine the different processes.

Now you'll notice that I didn't say things like, "Well you always need two-week sprints, and you need a planning meeting, etc". Daily standups and Scrum are not the point. Those are tactics. They're not bad! In many cases, they're incredibly helpful tactics. But they don't define the process.

The strategy behind Agile is being able to focus on small chunks of work that you can quickly get in front of people to learn if you need to change direction. Continuous integration and continuous

> deployment help you figure out very quickly if things are broken. You don't find out 6 months later and then cycle back into the process again.
>
> Agile is not specific tactics. Just like Lean Startup isn't only A/B testing or customer development and user centered design isn't just writing things on post it notes or making wireframes. Those are just tools. Occasionally using a hammer doesn't make you a carpenter.
>
> Know the strategies behind your process and adapt the tactics and tools to your team.

How do you fit user research into an Agile process?

> Companies newest to Agile struggle most with fitting user research into the process.
>
> They worry about design teams working ahead of engineering, which is really tough at the very beginning of the Agile transition, because it means that engineers are sitting around waiting for people to finish designing the stuff that they're supposed to start building. So, people try to dramatically shorten the design process.
>
> You end up with a collection of tiny waterfalls.
>
> Of course, you can't avoid that altogether. I don't believe teams need to do everything together. Not everyone needs to do user

> research together or sit around and create wireframes together. Designers will need to do some work ahead of engineering so that engineers have something to work on. But they don't need to design everything ahead of time.
>
> Also, don't think of user research as something that only happens at the beginning of a project. Certain types of user research does tend to happen early - the generative stuff where you're learning a lot about your prospective customers and their needs often gets frontloaded – but we need to stay in constant contact with users throughout the project and involve everybody on the team in some of the regular research activities.

In your almost 20+ years in tech as a designer and developer, what are some of the most dangerous UX mistakes you see Agile companies make?

> Setting the designer up as the user champion. We should all be user champions.
>
> The other problem I see is creating this series of mini-waterfalls where engineers are just handed very specific things to build with no input into the design or research process, so they don't have the context to make good decisions.
>
> We need to work together on many product decisions. You can't just say "I did all the work upfront, now it's your turn".

> The situation is even worse honestly when you're dealing with a design agency where the conversation is very much "Here are some comps, implement this," and then all of a sudden the engineers are left with so many questions that they just answer in the simplest way possible.
>
> It also absolutely drives me nuts when it's set up as designers versus engineers, and every decision is, "Oh, is this going to be good for the user or good for the engineering team?" We should have a process that's good for everybody, and that comes from collaborative teams where everybody has an understanding of the user and the ability to contribute to important decisions.

How do you run retros so that people actually "fail better"? instead of just "fail faster"?

> In terms of time length, the retros usually run 15-30 minutes. Any longer and it costs a lot – not just in terms of salaries, but also morale. Designers and developers hate disrupting their flow with meetings.
>
> To focus the discussion, I start every retro with the Start-Stop-Keep format. Based on our progress in the past sprint:
>
> - What should we start doing that we haven't already?
> - What did we try that we should stop doing?
> - What worked so well that we should keep doing it?

> We also review metrics of any recent features that shipped. One thing I don't like about Agile is the misleading feeling of "Done".
>
> We even have a column on the Kanban board that's called 'Done', right? So, "Oh, I moved it to Done, that task is done."
>
> That task is not done. It's just in users' hands. It's not done until it's changed the user behavior in a way that meets the business goal we set. Until then, all you've created is an output without an outcome.
>
> You might ask some other variations of the questions in your retro, but you must change the conversation from "What did we accomplish in terms of getting working features out the door?" to "What did we accomplish in terms of impacting the business and changing user behavior?"

In an Agile UX process, what else can we do to redefine the meaning of "Done"?

> Understand that the job of anyone on a product team is not to ship a feature. Your job is not to write code, your job is to increase a business metric. We need to show that number on a big information dashboard so that everyone can see it.
>
> If we're telling engineers, "Hey, your job is to knock off 6 tasks a day and hit this particular velocity and close this many issues," that's all you will get.

> We need to tell the whole team "Hey, your job is to increase this number. Nothing is done until that number's where it needs to be (and these other numbers haven't suffered)."
>
> For example, "Increase user activation from 15% to 20% without decreasing retention".
>
> When you focus on a common goal like that, it also creates a stronger sense of "the product team" versus "the development team" and "the UX team". Companies need to reward their teams fairly based on those business goals, which is admittedly really hard to do.

Daniel Castro, Design Director at Sumo Logic

On Agile UX

What common mistakes have you seen companies make as they fit user-centered design into Agile?

> The first time I heard of Agile was when I was designing at Verizon. One of the Agile evangelizers who developed the manifesto presented a really detailed deck to the company.
>
> I remember the response from the whole design team was "This whole process is never going to work." At that time, the concept of a UX "Sprint 0" just didn't exist. We were just expected to start designing and developing immediately.
>
> So, as with a lot of processes, you need to pick out what applies and what doesn't. Since the early 2000s, we've seen so many different flavors of Agile that sometimes it almost feels like we've just created micro-waterfall.

> In fact, a developer here at Sumo Logic actually has a sign in his cube that said “Drink coffee, do stupid things faster and with more energy!”. Agile is the coffee for your design process – it is a fantastic tool for efficiency, but it doesn’t deter you from doing dumb things.
>
> The biggest mistake is thinking that Agile will help you move in the right direction. You need to build in a series of checks and balances to ensure you’re building the right thing.

How do you strike that balance between user-centered design and Agile processes?

> You take advantage of Agile’s collaborative nature to expose everyone to the value of user-centered design.
>
> User-centered design is naturally iterative, but in reality you only have a few shots at revisions before you frustrate engineers. They’ll get cranky because the requirements keep changing.
>
> To make the most of the iteration sprints you can realistically run, one of our most successful tactics is ensuring everyone understands that each design change is supported by usability research and testing.
>
> In our design process, the first step is to validate whatever problem a product manager presents with customer interviews. Once we’ve spent time digesting the feedback in the form of customer journey maps, our design team clears out their calendar for ~2 days. We

email the whole company and advise certain stakeholders to be on-call for feedback.

For those two days, we'll hold a highly focused "Swarm" session in which we dive deep into divergent and convergent design around the problem. The Swarm also helps us align to the product vision for the ensuing sprints and the business goals. We might also create additional generative research questions with developers and PMs, which then breaks down to a series of tasks you test with users with alternating sprints of testing and design.

That list of tasks is the contract binding developers and designers together for each sprint. You reach a solid middle ground where developers aren't pushing back on every change, and designers aren't lost in blue-sky thinking.

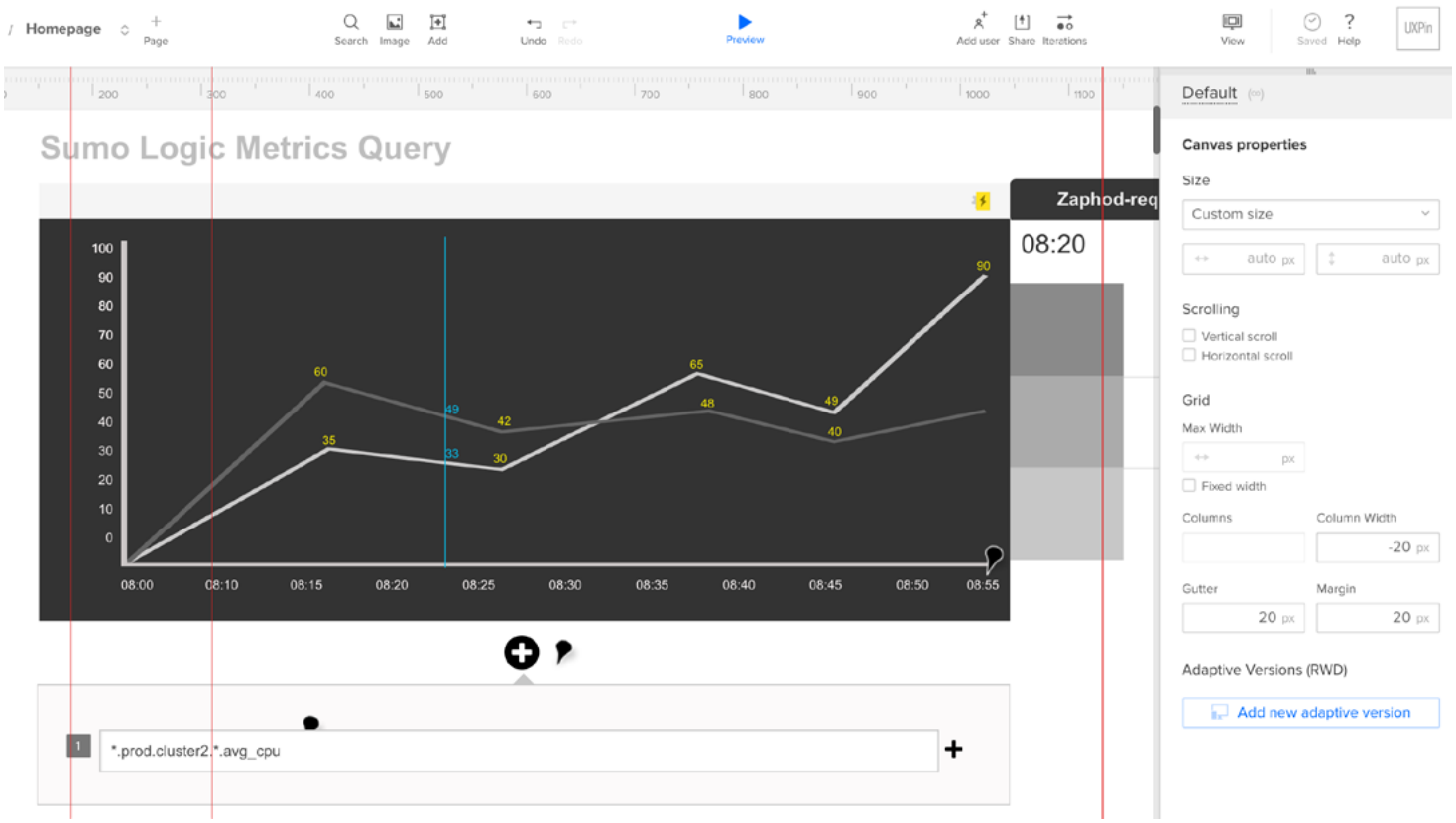
You get more buy-in for unexpected iterations because the context changes completely. Designers aren't forcing the developer's hand – the customer is guiding everybody along.

How do you approach documentation in the Agile UX process?

You can't completely throw out your documentation just because you're Agile. You need to create smarter documentation.

Documentation doesn't need to live in a document. You can be nimbler, like Slacking your team a screenshot of your whiteboard session. Don't obsess over the form or format – the singular goal

> of documentation is to ensure that everyone agrees to the success criteria.
>
> For example, if I need to convey more detail after a whiteboard sketching session, I'll create a quick UXPin prototype and share with the team instead of marking up a huge product specs document.



Sumo Logic prototype created during a discovery phase for their Unified Logs Metrics product

> In the platform, I can literally just drag and drop objects from the dozens of libraries to convey the concept that our team discussed. Add some notes to the prototype and share with the team, and now everyone has naturally created documentation together that actually helps us make better decisions down the line.
>
> Understand the collaborative spirit of documentation, not the tactics.

Does this natural form of documentation help you mold your product strategy?

> Absolutely.
>
> For Sumo Logic, the UXPin platform really shines at helping us create collaborative visualizations to help PMs and developers better understand both the horizontal requirements (overall feature set) and vertical requirements (depth of each feature).
>
> Even if the PMs and developers aren't physically present, we can can send them a project link for their comments. They suddenly have clear visibility into how the scenarios play out for each persona, and how it all comes together to form the overall customer experience.
>
> Collaborative prototyping helps the team understand the end-to-end solution and how it breaks down into chunks of work for sprint planning. Very early on, the team can start to understand what Release 1, 2, and 3 looks like and how they all map back to the core vision. Everyone gets a better sense of what they should and shouldn't build.
>
> The prototype naturally joins the product strategy and design tactics together. Even though your final product might not look anything like the prototype, you've created a living representation of the core success criteria. You can't really see that with paper documentation.

How has prototyping helped you define success criteria in Agile UX?

> So, after we've held the "Swarm" sessions and conducted user research, we'll sketch through all the feedback. Once the sketches start to convey flow, we'll dive into a lo-fi prototype.
>
> The prototype flows are the heart of the design. They inform all the tasks, which of course define success for the product.
>
> With a prototype, you can tell confidently tell a developer "Hey, we tested this design and it's ready for you". After the first formal build, you can compare the coded design back to the prototype and list of usability tasks.

For each sprint, how do you build the components without losing sight of the overall strategy?

> You need to set success criteria in each sprint on two levels.
>
> Let's return to the bicycle analogy.
>
> First, you need the horizontal success criteria: the person needs to sit down, hold down the handlebars, pedal, and move from A to B.
>
> Secondly, you need vertical success criteria: the frame design will be aluminum, the pedals are a certain size, etc.

> Not only does the team need to understand both levels of success criteria, but you also need to ensure you're testing at both levels.
>
> You can't test features in isolation. You need to test the entire end-to-end solution as early as possible. Otherwise, you get lost in vertical paths. You build the most amazing tires but Billy can't ride the bike because the tires don't fit on the frame.
>
> Of course, prototyping helps us get that perspective at each step in the process. It's a living contract for both sets of features. Anyone can have high-level conversations about strategy, or dive deep into individual components.

What Agile principles have you found useful to user-centered design?

> Agile gives you a concrete structure for planning ahead. It brings designers back to earth and prevents them from wanting to design forever.
>
> Agile has also democratized the design process so that even sales and customer success can contribute to the product.
>
> We're able to break up work, encourage open contribution, and move away from the "lone designer" mentality.

On the other side of the spectrum, how do you prevent too much design collaboration (e.g. design by committee)?

That's always a tough issue.

It starts with the designer's attitude towards collaboration.

First, they need to see themselves as a facilitator for gathering, shaping, and testing input.

Secondly, consider holding 1:1 sessions with vocal stakeholders. During design reviews, let them air their thoughts, but if they dive too deep into prescriptive advice, tell them you care so much about their ideas that you want to dedicate focused time to discuss.

Sometimes, you'll find that stakeholders might even reconsider since they realize they need to separate personal opinions from facts. It seems like a paradox, but sometimes being overly open actually helps people better accept dissenting opinions.

For example, we were working with a product manager who wasn't exposed much to the design team. We had trouble communicating our approach, and he was adamant about a certain feature. We simply responded with "We don't think it will work, but let's test it and see".

Coming out of that usability test, the original idea was invalidated, but we found pieces of it that could work in a different context. So even if the testing validates the designer's idea, you can't tell others "I told you so". You need to convey a spirit of "We're willing to work with you".

> Finally, designers can try reframing the conversation by saying "Look, we're trying to make you look amazing. We want to release a product that will make people want to write about your features." versus "Look, my idea is right because…"
>
> All of a sudden, something clicks and the other stakeholder realizes that the designers are on their side – and sometimes that means saving them from themselves.

Do you follow a component-based design system (e.g. Lego-block approach to design)?

> We're currently in the middle of a reboot project that will help us work even closer along those lines.
>
> Luckily, we've been blessed with UI developers who sit inside the UX team. As such, designers and developers can define the major components of a pattern library together.
>
> Once you have that common design language that breaks down to patterns and elements, everyone does less redundant work. Designers can focus on solving the large business problems, and developers feel empowered to use vetted components instead of asking designers to verify everything.
>
> We aren't 100% there yet. But after this reboot project, we'll be much closer. You can imagine how exciting it will be to just go into

> a shared environment and grab the exact component to match your context of use.

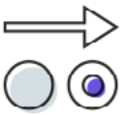
How would you improve the Agile UX process?

> I'm all about expectations, whether they're big or small.
>
> The best process starts with collaboratively defining the requirements for success. From there, you can chunk out all the pieces into your sprints.
>
> Within each sprint, you also set expectations for each chunk of work. What are the goals? What are the discovery questions? What are the tasks? How will we know customers love this?
>
> You repeat the process, ensuring that everyone on the product team always knows the answers to the four questions above.

The Full-Stack UX Platform

Your entire UX process in one place

Design:
Create lifelike prototypes quickly with Photoshop and Sketch integration.

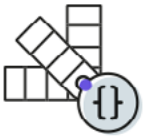


Iterate:
Built-in version control improves efficiency and eliminates confusion.

Document:
Cleanly annotate your designs. Insert custom code snippets that travel with elements.

Collaborate:
Get feedback and co-design on any project anywhere.

Scale:
Automate consistency and documentation with design systems (syncs with Sketch).



Implement:
Auto-generate style guides, assets, and specs for developers.

Try UXPin now