Alex Cowan @CowanSF
(https://twitter.com/cowanSF) - 12:47 PM Feb 04
(https://twitter.com/cowanSF)

YOUR BEST AGILE USER STORY

FEBRUARY 4, 2014 BY ALEX COWAN (HTTP://WWW.ALEXANDERCOWAN.COM/AUTHOR/ACOWAN/) - 10 COMMENTS (HTTP://WWW.ALEXANDERCOWAN.COM/BEST-AGILE-USER-STORY/#DISQUS_THREAD)

TABLE OF CONTENTS

It’s generally accepted that agile is the best organizing framework for innovative product development. And yet most organizations only use a fraction of agile’s capabilities- startup’s lose themselves in the urgencies of the moment and enterprises stifle it with complexity. This piece will equip you with foundation concepts and techniques you can use to maximize agile’s effectiveness.

AGILE: SHORT & SWEET

Agile's become kind of a 'big book' discipline- agile primers over 400 pages abound. This is ironic (not to mention unfortunate) since the original agile manifesto ran 68 words and its goal was to democratize and broaden participation in software development. Here's the core of it:

Individuals Interactions

Processes & Tools

Working Software

Comprehensive Documentation

Customer Collaboration

Contract Negotiation

Responding to Change

Following a Plan

Quick note: When they talk about 'documentation', they mean big, long specification plans. If you feel you need to deliver documentation, help, etc. to your user, agile doesn't suggest you do otherwise.

In the practice of agile, I've found two core practices particularly helpful: 1) the use of agile 'stories' and 2) the concept of small iterations.

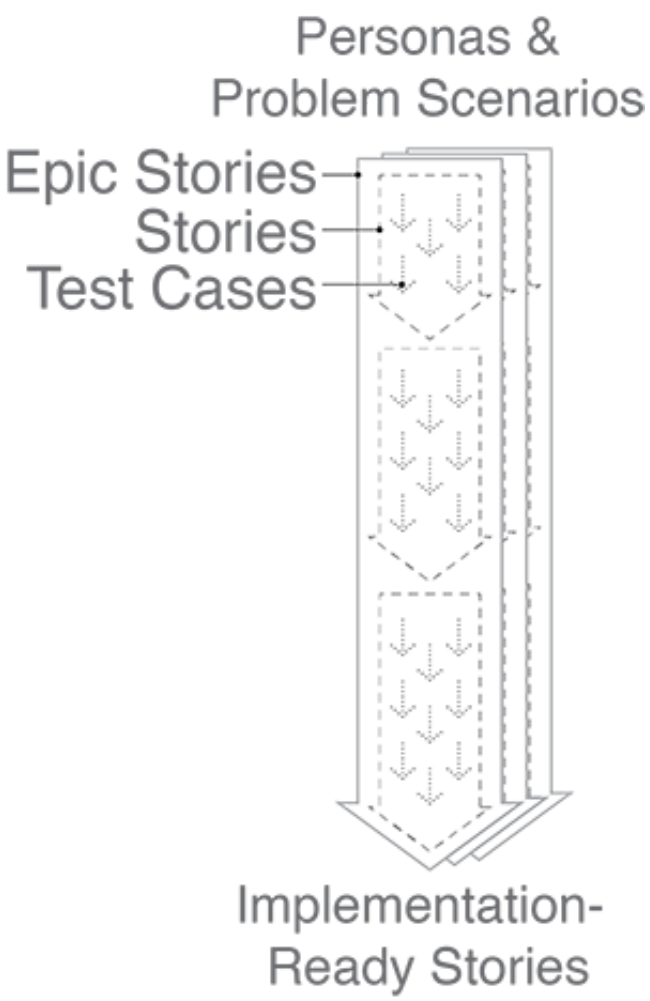
AGILE USER STORIES

User stories have a specific format, designed to help the author be descriptive and the reader (a developer) to take action:

> "As a [persona], I want to [do something] so that I can [derive a benefit]"

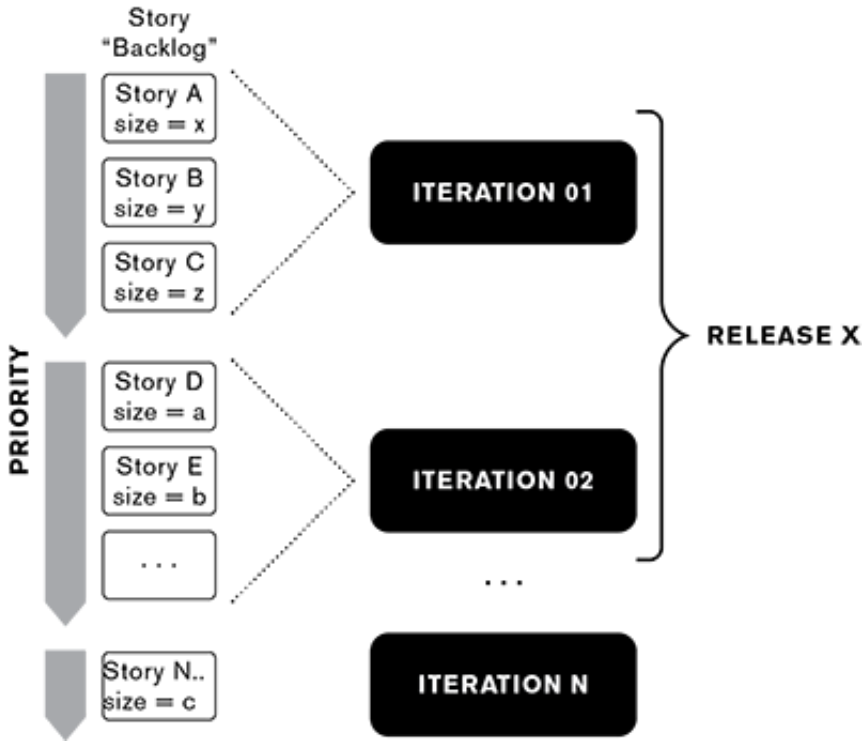
If you're the product person, your job is the fill in the (orange) blanks. The persona is a vivid, humanized, yet operational description of your user. The '[do something]' is a some action you assume the user wants to do, and the '[derive a benefit]' clause is a statement of why you've assumed the user persona wants to do what you describe. The [derive a benefit] clause is the most neglected by most creators of agile user stories and yet the most important. The reason why it's so important is that this is where you're saying why the

feature you're proposing to build makes sense. The world's an uncertain place and you don't necessarily have to be right about why but you *do need to state your point of view on why it makes sense in specific, testable terms*. If you're wondering if it's really worth the time, let me assure you that 98.6% of the time it is. It takes a relatively long time to design, code, test and support features and having a way of making sure they matter is always worth the investment. If you're wondering about how to do an outstanding job on figuring out users and what matters to them, look no further! I recommend this site's: Tutorial on Personas, Problem Scenarios, and Value Propositions (/tutorial-personas-problem-scenarios-user-stories/).



Stories can be ever bit as structured as old school 'requirements'. Individual stories are nested within 'epic stories' that describe a larger, more general scope of activity by the user. For more detail, 'test cases' are associated to stories. Organizing inputs to development this way turns out to be a more descriptive, more actionable, more discussable way to define what you want to build. If you want to jump ahead, check out the Examples below and/or the User Stories Template (https://docs.google.com/a/alexandercowan.com/document/d/1M7UGO-XymKBNfFvqIjeWdABO7z1f4JOaje_VjCOi4ss#bookmark=id.349syyg7md24).

ITERATIONS



(http://141nh047iozd1l75s22eer06.wpengine.netdna-cdn.com/wp-content/uploads/2014/02/agile-iterations.png)If user stories encourage more thoughtful product design (/learn/product-design/) and the use of design thinking, iterations encourage better venture management (/learn/venture-planning-management/), particularly in conditions of uncertainty (which is just about everything, to a degree). An iteration has a prioritized set of stories as its input and a version of working software as its output. Whether or not you release that software (or system) to the customer or end user is up to you- there is additional overhead to doing that and it may or may not make sense. The virtue of this approach is that you don't make decisions too far ahead of a) what you can actually know about how the user responds and b) how much stuff you can actually sit down and design well (per the above section). An iteration is typically 2-6 weeks.

START RIGHT, END RIGHT: AGILE & DESIGN THINKING

A recognition that we could be building much better products is driving the sweeping popularity of design thinking. Historically, the connection to agile practices, specifically the creation of better user stories, has been relatively light. That's changing. The twin pillars of design thinking are empathy and creativity and it particularly emphasizes customer discovery 'outside the building' with customers in their natural

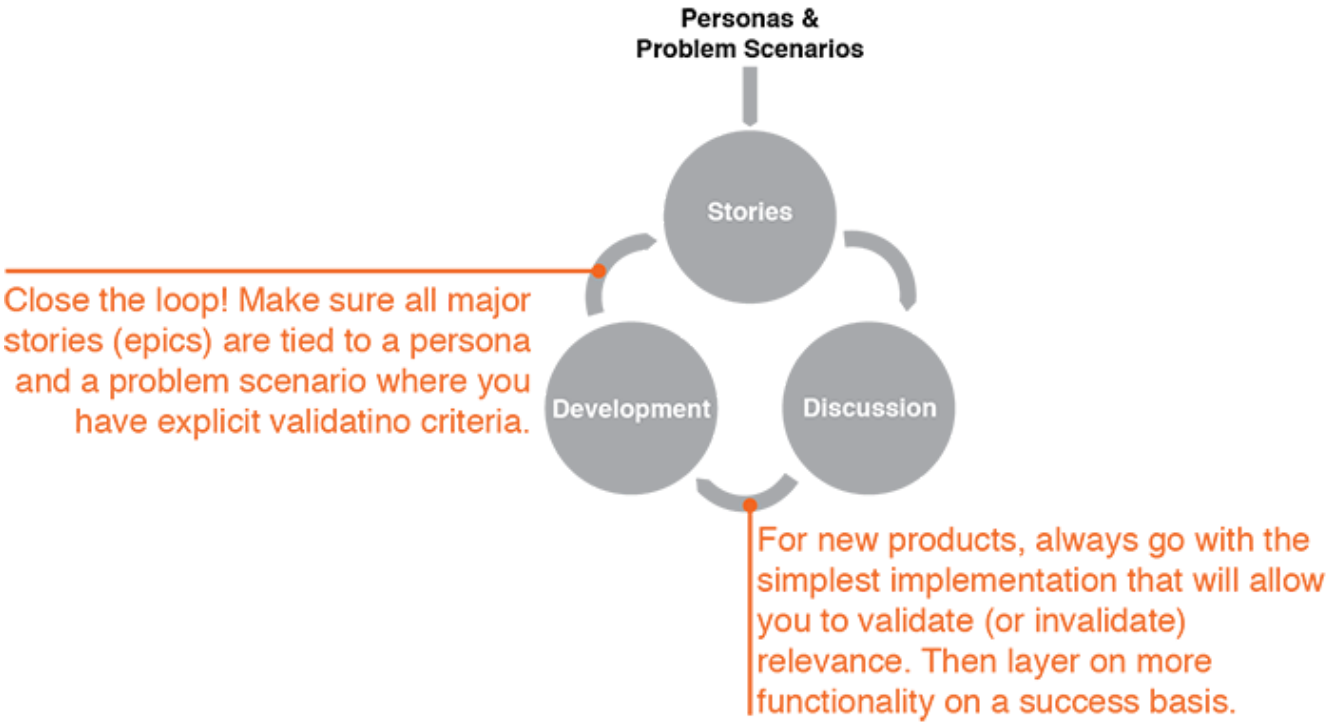
environment. This doesn't mean selling and it doesn't mean asking the customer 'Do you [would you] like my [product]?'. It means watching customers act naturally, speak sincerely about themselves, and make real tangible choices between alternatives.

(http://141nh047iozd1l75s22eer06.wpengine.netdna-cdn.com/wp-content/uploads/2014/02/Agile-Development-Process-Design-Thinking1.png)Actionable outputs for this discovery are personas and problem scenarios. Problem Scenarios are needs or habits that you might fulfill with your product. For each of these, you should be able to identify one or more alternatives that the customer is using today (if there's a real need, at some level they're doing something). You then lean your value propositions against the customer's alternatives- is what you have better enough for them to buy what you have? Every agile user story should involve one of these personas and tie back to a problem scenario/alternative/value proposition you've validated in the real world. For more on this, see the section here on product design (/learn/product-design/).

Once you have stories that tie back to strong, validated customer discovery, your job is to organize this material into compelling narratives your development team can understand. There are a few ways to do this, but you want to make sure your compiled personas and problem scenarios are highly visible. Personally, I use this Venture Design Template (http://ref.alexandercowan.com/pdesign) which ties those items together. Presentations are also good, particularly if the focus is you describing what it was like observing the customer. The key thing is to spur interest and discussion with your developers about your stories, a discussion that drives toward more customer-relevant implementations. If no one ever says 'I just built what you said' then you know you did a great job.

KNOWING WHAT YOU DON'T KNOW (YET): AGILE & LEAN STARTUP

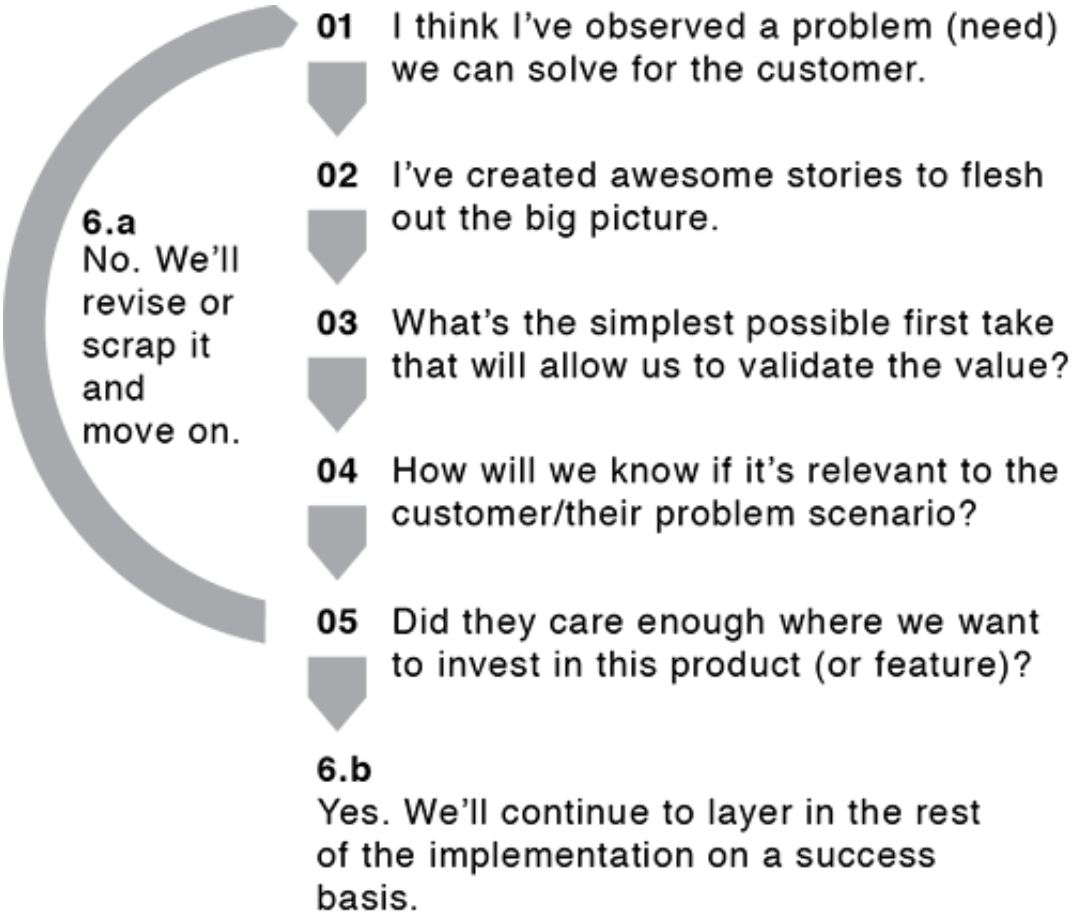
Most of us hate to admit we don't know things, particularly in a professional setting. Yet there is so much we don't know, particularly if we're innovating. The Lean Startup movement is at its core a way of saying 'There is a lot we don't know yet but here's how we're going to work through that with a minimum of waste so we can take multiple shots at a successful outcome".



At the core of the Lean Startup tool set there are: 1) prioritized assumptions 2) experiments paired to these and 3) the related idea of a 'minimum viable product' (MVP). With something innovative, by definition you're imagining a possible future where your product changes the customer's life in some way. There are things you don't know yet and it's important to articulate those things explicitly so you can honestly decide whether the idea is valid with a minimum of time and money spent. The Dropbox founders famously created a synthetic YouTube demo of their product, requesting signup's, to validate their assumption that a truly seamless experience would make direct file sharing succeed in the mainstream. The MVP converges both assumption and experiments: it is the simplest possible version of your product or feature that will allow you to validate it fundamental relevance with the customer. For more on this, see the Lean Startup materials in the Venture Planning (/learn/venture-planning-management/) section of the site.

Marrying Lean Startup and agile offers vast improvements in the output of your innovation cycles. The first step is excellent customer discovery, described above. Following that, always define and elect the simplest possible implementation of a new product or feature so you can validate its relevance before (over)investing in it. For example, when building something new, I like to publish an "MTP", minimum testable product, before even releasing the MVP. Why? The idea of an MTP is to decouple the core relevance of a feature from

its user interaction, making sure that its not the user interaction stalling user interest. The MTP is the minimum amount of product (usually just a front-end prototype) that allows you to artificially motivate a user to validate that your user interface makes sense. In general, the idea is to put as few chips on the table as you can to see if you've made a good bet.



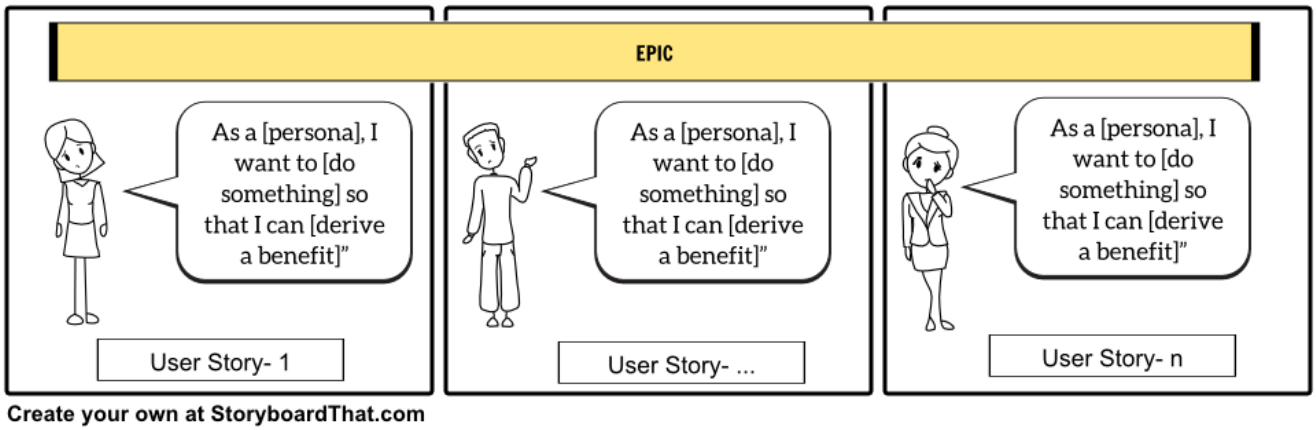
The real success driver is a rigorous process of validating new features- this is what separates simple 'output' (stuff you've created) from 'outcomes', things that actually matter to the customer. This is accomplished by threading stories back to value propositions that relate to a customer problem scenario. You're then testing whether your implementation delivers adequate value to the customer's problem scenario- is it better enough than their current alternatives?

EXIT FLATSVILLE: ENERGIZING YOUR AGILE STORIES WITH STORYBOARDS

Tying every major user story back to a vivid, real-world persona (/tutorial-personas-problem-scenarios-user-stories/#Example_Personas) & problem scenario (/tutorial-personas-problem-scenarios-user-stories/#Problem_Scenarios-2) and making sure each story has all three clauses is a good checklist for making sure you're using high quality inputs to drive your product development. Sadly, it's just a start. Particularly if

you're trying to change a culture of 'just writing requirements' or otherwise not substantially engaging with what's really making users tick, you'll need to be constantly coaching and showing. You'll need all the tools you can get.

Storyboarding is an excellent tool pushing yourself and your team to think substantially about your customers, what they want from you, and what they're really doing in real life. Especially if you're developing something relatively new, I highly recommend storyboarding your epic agile user stories (recap: epics are 'super stories' that broadly describe an action which is then broken down into individual stories). After you draft your epic, but before you detail additional stories within it, try describing it with a storyboard:



(http://www.storyboardthat.com/userboards/alexcowan/epic-story-template) Create a Copy (http://www.storyboardthat.com/userboards/alexcowan/epic-story-template/copy)

You do NOT have to know how to draw. I used an online tool from Storyboardthat.com (http://www.storyboardthat.com). If you prefer to work on paper, you can print out some ready-made squares here: storyboarding squares (/workshop-storyboarding/#Materials). Depending on what tools you use in the office, you may find another storyboarding tool you like.

STORYBOARDING AGILE EPICS

The epic storyboard example above has references to individual stories, but what I actually *do not* recommend working at that level when you first start. First, draft the epic story; then just let the storyboard write itself by thinking through the epic sequentially. Thinking sequentially doesn't mean thinking in a single line or thread- your epic may be more of a web where the user's experience may diverge into multiple possible sub-stories, converging later or not as the case may be.

The process of storyboarding will help you:

a. test the depth and scope of your understanding about the persona(s) and problem scenario(s)
b. stimulate interest and discussion in the story with your implementation team (and peers on the product side)
c. push you to think through the details so you've implementors don't have to start from scratch there

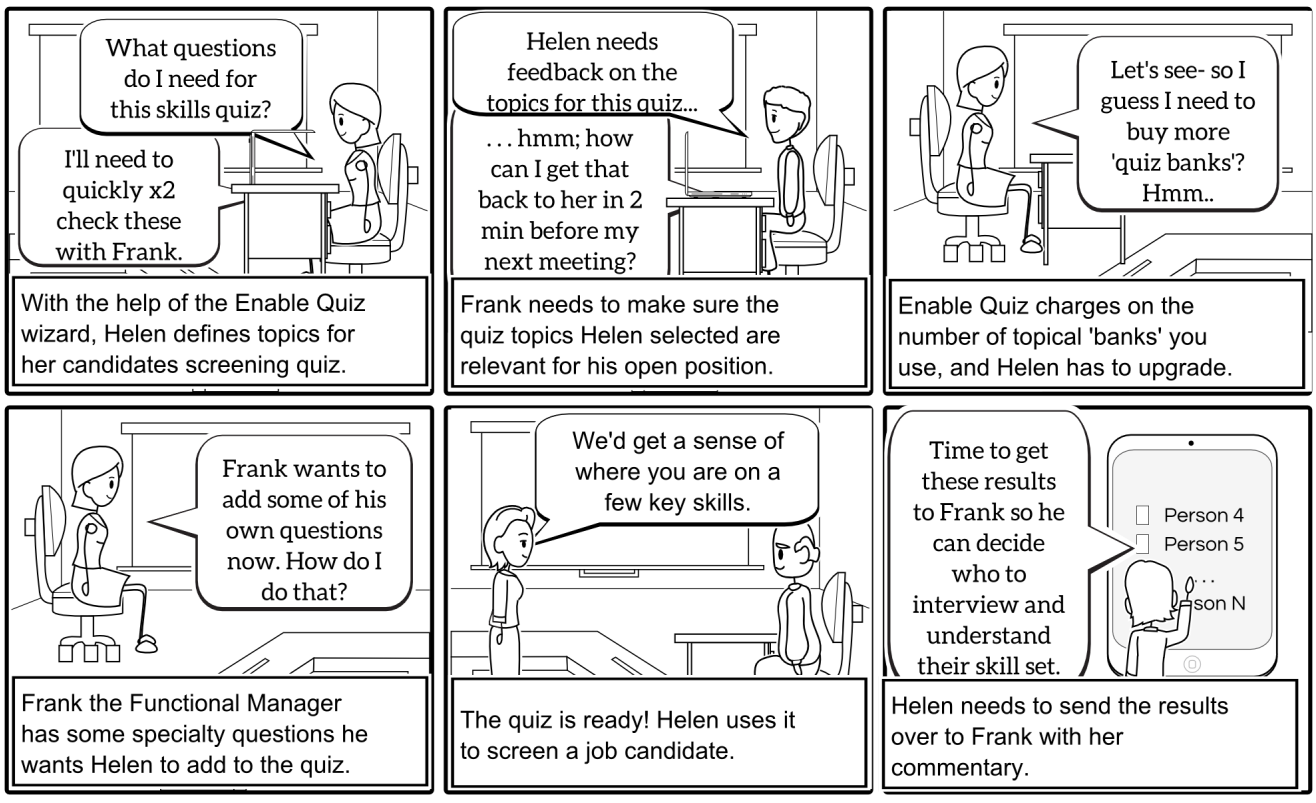
This last one is particularly important if you're new to product development. I coach lots of first-time entrepreneurs, and their #1 challenge at the product implementation phase is thinking through what they want in enough detail. Most of their stories are actually epic stories (or broader) and need more granularity and sequence- storyboards help. A classic failure mode is leaving your development team to design your product. This not to say your developers can't think creatively or answer these questions, and part of the point of these inputs is to stimulate creative discussions with them so the team's collective imagination and expertise is maximized, but your job as the product person is to provide a well-articulated, thoroughly considered starting point.

EXAMPLE EPIC I

Let's take a look at an example epic. I use a (fictional) company called Enable Quiz in my book (/starting-a-tech-business/). They offer lightweight technical quizzes. Managers use the quizzes to screen new candidates and also assess the skill sets of their existing teams so they can plan training, etc. Let's say the persona 'Frank the Functional Manager' wants to assess one of his teams so he can plan a skills development program. The 'epic user story' might be:

'As the HR manager, I want to create a screening quiz so that I can understand whether I want to send possible recruits to the functional manager.'

The storyboard below summarizes this epic story:



(http://www.storyboardthat.com/userboards/alexcowan/agile-epic--enable-quiz) Create a Copy (http://www.storyboardthat.com/userboards/alexcowan/agile-epic--enable-quiz/copy)

I made notes at the bottom of each panel since you're probably not familiar with the operations of Enable Quiz. If you're working with a team that's already familiar with your topic, I wouldn't worry about that additional level of detail.

DETAILING THE EPIC

In the step above, we wrote an epic and then used a storyboard to thinking through the course and sequence of the epic without necessarily worrying about how the storyboard panels might break down into individual stories. This is the best first step since you want to make sure you're creating a vivid, credible customer narrative first and then worry about formulating that narrative into stories. Agile user stories are a tool, not a strategy. They provide an opportunity for quality work, but by no means guarantee it.

(http://www.alexandercowan.com)

notes and individual stories in a simple table. Here's the epic above broken down (in this example without any further edits):

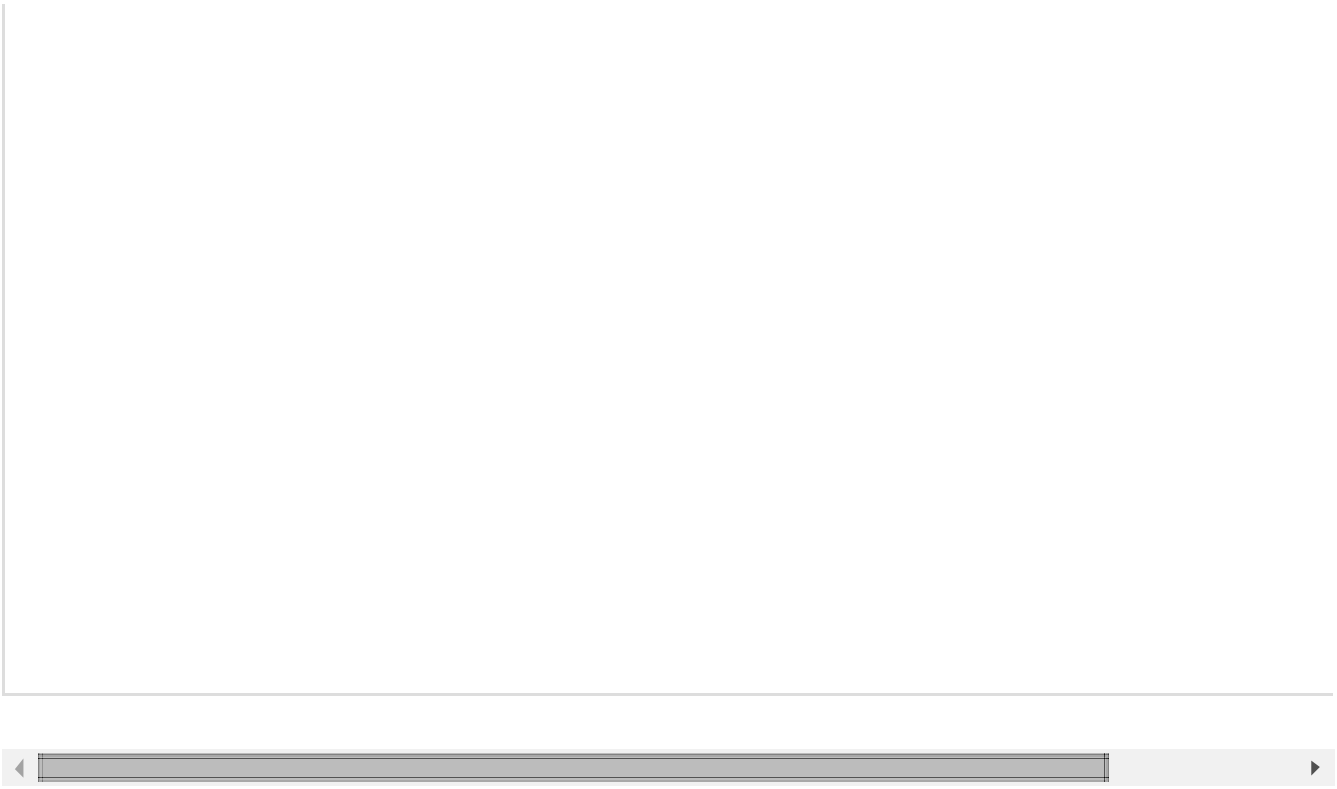
NOTES
This epic assumes Helen the HR Manager and her company have already signed up for the Enable Quiz servi screen job candidates. Helen's first goal is to set up a quiz to screen candidates for a new engineering positio Frank the Functional Manager has open. The idea with the system is Helen can do more on her own to scree candidates, so she wants to take the initiative and set up this initial screening quiz. This first step covers two stories:A) "As an HR manager, I want to browse the banks of available quiz questions, matching them with t skill sets described in an open job recruitment so that I can draft a screening quiz with relevant questions." B) "As an HR manager, I want to check the quiz topics I've selected with Frank the Functional Manager so t can make sure I have the right topics."
Frank is super busy and yet has a very clear idea of what topics are relevant, so the name of the game here is make it easy for him to answer Helen.C) "As a functional manager, I want to review, revise and confirm the question topics the HR manager has selected so the quiz will be ready to use for first round candidates."Note raises some interesting and possibly non-obvious questions about how Helen will do this and how we can ma easier for her to collaborate and close on this with Frank.

Enable Quiz charges based on the number of question topics (organized in ‘banks’) that the customer wants t
In our narrative, Helen finds she needs to up her limit on available question banks.D) “As a manager, I want
increase the limit on how many quiz question ‘banks’ I can use so that I can move ahead with the quiz I’ve
formulated.”Note: I’m not adding test cases against these stories in the example, but a good one here would l
make sure that users can put together quizzes with as many question banks as they want, getting warnings ab
their limit but only encountering a hard requirement to up it once they want to actually use the quiz. Bored v
the level of detail here? Well, good products are the accumulation of lots of small, thoughtful details.

Now Helen gets an email from Frank- after seeing the draft quiz, he has a couple of his own questions he’d l
add!E) “As a manager, I want to create custom quiz questions so I can add them to my quiz.”

We're ready! Helen's using the quiz with a candidate for Frank's open engineering position.F) "As a manage want to administer the quiz to a recruit so I can understand where they are on key skill sets needed for the position."Note: It's very likely with a little more digging this story is actually an epic itself and needs more d but I'm trying not to bore you, reader, with too much stuff!

Helen's administered the quiz to some recruits and wants to send the results to Frank, possibly with supplem notes in a few cases.G) "As an HR manager, I want to send the results of our screening quizzes to the Functi Manager so he can decide who he wants to interview."H) "As an HR manager, I want to add supplemental no results for some of the candidates so I can share additional thoughts on the candidates with the functional manager."



For the child stories of this epic detailed with test cases, see the final section here, ‘Example User Stories‘.

For more on the practice of storyboarding, see: Storyboarding Tutorial (/storyboarding-tutorial/).

In this next section, we’ll look at one of the most durable checklists for creating great stories: INVEST.

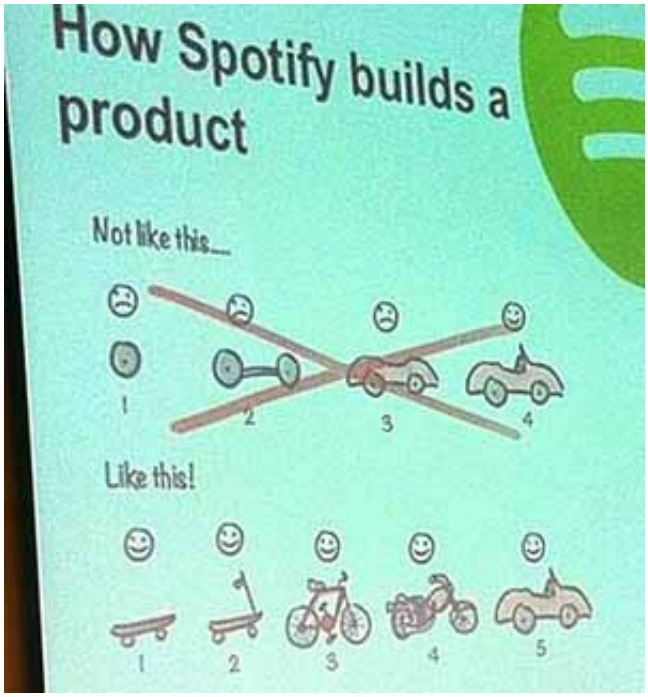
REVISITING INVEST FOR BETTER STORIES

Bill Wake (@wwake (https://twitter.com/wwake)) introduced the INVEST mnemonic in his seminal post on creating better stories, suggesting they should be Independent, Negotiable, Valuable, Estimable, Small, and Testable. An acute observer of common disconnects between product people and developers, his mnemonic remains an excellent checklist for the creation of high quality product design inputs.

These are ideas, not commandments, as Bill Wake makes clear in his post (http://xp123.com/articles/invest-in-good-stories-and-smart-tasks/). When in doubt, always do what *you* think makes sense.

Here are a few notes on INVEST, reflecting also what we’ve covered here:

INDEPENDENT



Ideally, you should be able to implement your stories in any order while still realizing the benefit of seeing something working at the end. This is important for two reasons. First, you lose a lot of the adaptability agile's supposed to deliver if your stories have (many) interdependencies. Second, it makes it much more likely you'll end up at the end of an iteration without working product that you can validate to make sure (ala Lean Startup) that you're moving in a valuable direction.

Spotify is known for its success with agile and other adaptive techniques and this slide from one of their talks summarizes the value of Independent (and Valuable and Testable).

NEGOTIABLE

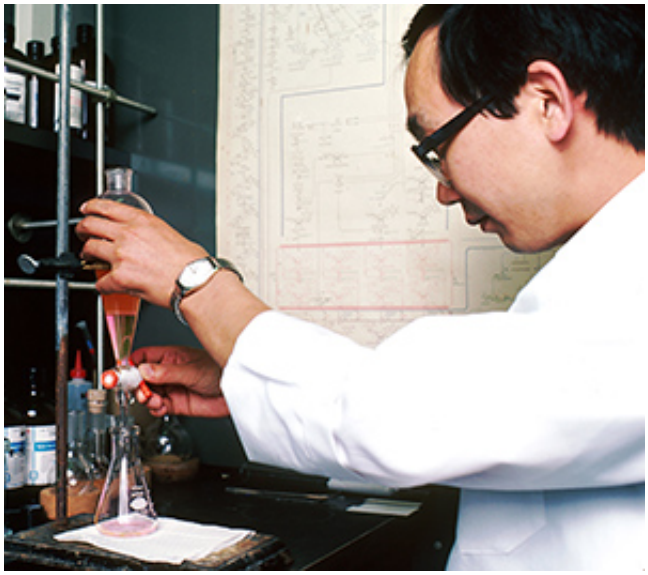
User stories are not requirements. They were meant to replace requirements, which suck. They're not another just another template to put on a piece of paper- they're part of a process where the product person (original drafter) of the stories co-creates a working incarnation of the story with the implementer. This way, the original author gets someone to help them improve/"edit" their story and make sure the story's interpreted into product in a way that delivers on something Valuable for the user.

VALUABLE

Each story once implemented should be both functional and relevant to the customer/user. In terms of the material above on design thinking, the story should tie back to a validated problem scenario for one of your personas (and this is true even if the persona is internal, like an administrator). This helps avoid waste and pro-forma deliveries.

A quote from Bill Wake's original post on why this is important in software development projects: *Developers often have an inclination to work on only one layer at a time (and get it "right"); but a full database layer (for example) has little value to the customer if there's no presentation layer.*
We're hard wired to be hyper-sensitive to criticism/critiques, even when we know it helps. Don't feel bad about it, just try your best not to do it.

ESTIMABLE



It should be possible for a developer with relevant experience to roughly estimate a story. That estimate's then used by the product person to prioritize their list of stories, taking account of both value and cost (in terms of developer time).

Estimating's kind of controversial in the agile community and not without good reason: It can lead to disproportionate/non-effective levels of overhead and pointless recriminations when the final output doesn't 'agree' with the original estimates. These are all things agile strongly tries to help avoid.

If I were to offer three silver bullets on this, they would be: 1) Write good stories using the rest of INVEST and the material you see here. 2) Make sure everyone agrees the estimates are very approximate (like Small-Medium-Large) and for purposes of prioritization. 3) If you want to do post-mortems on estimates vs. actuals, make sure there's general agreement that it's to help the team in some actionable way and not just because someone's frustrated there isn't more implementation done.

SMALL

Your unimplemented stories are like a box of chocolates…ah, scratch that, reader; never mind.

Basically, your stories need to be Small for you to get the adaptability and reduction in overhead that agile offers. Big stories reduce opportunities for validation and require more elaborate planning.

TESTABLE

As the author of a story, you should be able to write tests that would allow you to validate that an implementation delivers on your intent. Many teams operate this way, but regardless of what everyone else does, you'll do yourself a big favor by writing some functional test cases in advance- you'll be much more likely to get what you want more easily.

The tests above refer to validating the 'output' of development- that the story delivers on your intent. As we discussed in the section on Agile and Lean Startup, if you're building something new and innovative, you'll also need test to validate 'outcome'- that what you've implemented delivers on a value propositions you've attached to one or more of your personas.

AGILE'S TOP 5 FAILURE MODES AND HOW TO AVOID THEM

Agile's been around since 2001 and widely practiced, to varying degrees. Few communities are more prolific and there's a lot you can read about how to approach it. The items you see here are my take on how to make it go right, how to get the most out of agile. To help show the importance of these items, here is a top 5 list of things that I've seen go wrong with the practice of agile:

1. Weak Product Design Inputs

Garbage in, garbage out (GIGO), as they saying goes. While you can have weak inputs and still in theory leverage the adaptability you get with small batch iterations, in most cases you're getting way less than half of what you could be getting from agile.

Quality customer discovery packaged into thoughtfully written, INVEST'able user stories that thread back to personas and problem scenarios is the best way to ensure this. This is particularly important if you're doing something new and innovative. See the above section on agile & design thinking and INVEST for more on this.

2. Poor Communication with Engineering/Development

The disconnect between 'the business slide' and 'the engineering side' has spawned countless books, comic strips, floor plans, and water cooler conversations. It's one of the top reasons statup's innovate so much more per person than enterprises. If you're on the 'business side' do great customer discovery and create great inputs and set aside time withe engineering to talk about them. Better yet, see if you can invite an engineer to come with you on a customer discovery exercise.

If you're in engineering, try framing questions in terms of personas, problem scenarios. Draft these things if you're not getting them from the business side- showing is 100x better than telling.

3. No External Validation Criteria

Often the business side is obliged to sign their name to 'requirements', providing some guarantee that they're right about what should be built. That's counterproductive, unreasonable, and above all, impossible. No one knows exactly how a customer is going to react. No substantially premeditated plan survives contact with the real world. What does survive is a plan optimized for actionable learning and adaptive change- that's what the above section an agile & the Lean Startup is about. It's also the T in INVEST.

It's silly to hold your nose, take the plunge and hope you're right. It's practical and effective to max out what you can know without spending a lot of time and money, start with the simplest possible implementation of a new feature, and then have an explicit plan to get a definitive result about its relevance. This is how a lot of actual innovation happens- effective experimentation (aka effective trial and error).

4. No Continuous Improvement Program

At the beginning of this resource, I warned against going overboard on agile orthodoxy and that's still the case. One area where a bit more structure (and I'd go for the simpler version) is helpful is the use of 'story points' and burndown's for looking at the effectiveness and ongoing improvement of your agile practice. It's beyond the scope of this piece, but you'll find ample resources online and if you're looking for something more comprehensive I recommend Greg Cohen's book Agile Excellence for Product Managers (http://www.agile-excellence.com/book/agile-excellence/).

5. Scrumfall

This refers to the practice of actually making a detailed long-term plan but calling it agile (among other things). This largely breaks your ability to leverage agile's adaptability against the validated learning you acquire on customers/personas and problem scenarios.

EXAMPLE AGILE USER STORIES

The second epic story below is organized in a more conventional fashion (vs. the epic above that's storyboarded). While the first epic dealt with a company using Enable Quiz to screen candidates for a new job, this one deals with a second scenario: a company using Enable Quiz to gauge their existing staff's familiarity with a set of technical topics, a "skills audit".

EXAMPLE EPIC I

Epic Story: "As the HR manager, I want to create a screening quiz so that I can understand whether I want to send possible recruits to the functional manager."

STORY	TEST CASES
As a manager, I want to browse my existing quizzes so I can recall what I have in place and figure out if I can just reuse or update an existing quiz for the position I need now.	Make sure it's possible to search by quiz name Make sure it's possible to search by quiz topics included. Make sure it's possible to search by creation and last used date.
As an HR manager, I want to match an open position's required skills with quiz topics so I can create a quiz relevant for candidate screening.	Make sure the quiz topics are searchable by name. Make sure the quiz topics have alternate names, terms for searching
As an HR manager, I want to send a draft quiz to the the functional manager so I make sure I've covered the right topics on the screening quiz.	Make sure it's possible to add another user by email in this flow Make sure it's possible to include notes and customize the email

	Make sure it's possibly to just copy a link (for case where HR manager wants to send their own email)
As a functional manager, I want to send feedback on the screening quiz to the HR manager so I make sure I'm getting the best possible screening on candidates.	Make sure it's possibly to supply comments in-app. Make sure the above are quiz-specific by default but can also be general. Make sure it's also easy to copy the name or URL of the quiz for their own correspondence.
As an HR manager, I need to purchase an upgraded service tier so I can add additional topics to my quiz.	Make sure that users with billing privileges can upgrade the service. Make sure that If the users don't have billing privileges, they see a list of those that do and can contact them. Make sure the charges are correctly prorated against the billing anniversary of the account.
As an HR manager, I want to add custom questions to the quiz so we cover additional topics that are important to the functional manager.	Make sure the customer is not charged for this bank. Make sure the custom bank is owned by the client organization and not accessibly by any other accounts on the system.

Here are a few other epics that might follow this one.

EXAMPLE EPIC II

'As the HR manager, I want to try out the screening quiz so that I can make sure it works as I expected and that I'm ready to both give it to candidates and share the results with the functional manager.'

EXAMPLE EPIC III

'As the HR manager, I want to give the screening quiz to a job candidate so I can assess their skill sets against the needs of the position.'

EXAMPLE EPIC IV

'As the HR manager, I want to share and explain the results of our screening with the functional manager so they can decide who they want to interview.'

IMAGE CREDITS
Pair Programmers: By Lisamarie Babik (Ted & Ian Uploaded by Edward) [CC-BY-2.0 (http://creativecommons.org/licenses/by/2.0)], via Wikimedia Commons
Chemist: By Linda Bartlett (Photographer) [Public domain or Public domain], via Wikimedia Commons
Present: By Kgbo (Own work) [CC-BY-SA-3.0 (http://creativecommons.org/licenses/by-sa/3.0)], via Wikimedia Commons
Chocolates: By Dwight Burdette (Own work) [CC-BY-3.0 (http://creativecommons.org/licenses/by/3.0)], via Wikimedia Commons
Meter: By Iainf 05:15, 12 August 2006 (UTC) (Own work) [GFDL (http://www.gnu.org/copyleft/fdl.html), CC-BY-SA-3.0 (http://creativecommons.org/licenses/by-sa/3.0/) or CC-BY-2.5 (http://creativecommons.org/licenses/by/2.5)], via Wikimedia Commons

Nathanael Coyne (Boehm) • a year ago

Thanks Alex, appreciate the insight and examples as I'm trying to figure out how I want to transition to Scrum and user stories, and dialing back on the wireframes and UI specifications.

The hierarchy of epics and stories is something I'm struggling to get my head around; it seems very ... organic? Some epic-level stories have as much detail as I'd expect to produce while other epics could be broken down into a dozen sub-stories and even some of them could be broken down futher to ensure they're testable.

Also, with your example epics I'm not sure if these are expected to have user stories under them but I'd have thought there would be more test cases that test the [derive a benefit] clause of the stories; I felt that the lack of test cases that explored that aspect of the user stories was a missed opportunity to increase the likelihood of delivering a successful product.

4 • Reply • Share ›

Alex Cowan Mod ➚ Nathanael Coyne (Boehm) • a year ago

Hi Nathanael-

First off, sorry for the slow response.

Second- thank you so much for your thoughtful comments & questions. I invest a lot of time maintaining this material and I'm thankful for every single visitor. That said, notes and questions likes this are so crucial to advancing the material.

Now, to your actual questions:

- Hierarchy and scope of stories: The above stories are examples I created for flexibility and congruence with the rest of the material on the site. It's possible (scratch that- likely) that they'd evolve during the discussion and implementation process.

I think the best way to scope epics+their children is to draft early and often, getting them in front of your implementation team (or proceeding with the implementation yourself) as soon as possible. Not to suggest that this is what

see more

• Reply • Share ›

Greg Cohen • 2 years ago

Great stuff Alex. This really places all the key pieces in context, especially how design thinking, lean startup and agile all fit together. And I appreciate you giving focus to the problem space because I see so many failures from solution first thinking.

4 • Reply • Share ›

Alex Cowan Mod → Greg Cohen • a year ago

Thanks Greg! I'm so glad you found it useful.

• Reply • Share ›

Jim Stover • 2 years ago

Outstanding post Alex. One thing I sometimes like to add to the end of the 'INVEST' criteria is an 'D for Demonstrable. At least for functional user stories, I find it helpful for the team to keep in mind that at the end of each sprint one of the goals is to demo the completed functionality to the customers / stakeholders / etc...

Adding onto the challenges or failure modes of agile, in my experience it aligns very closely with the results from VersionOne's State of Agile Survey: Company culture at odds with agile principles and values; external pressures to follow 'the norm' and what is known and understand in an organization (traditionally Waterfall - I have grown a gag reflex to saying Waterfall haha; and generally just broader organizational issues impacting an agile transformation.

3 • Reply • Share ›

Alex Cowan Mod → Jim Stover • a year ago

Thanks Jim! I'm so glad you liked it.

Demonstrable- I love it! Can we draw it on the whiteboard? Do a quick mockup? Storyboard or generally talk through the user experience? Demonstrable's a great criteria.

• Reply • Share ›

David Siegel • 8 months ago

I am an agile outsider. I have always had a crush on agile as a concept but never worked in a context that allowed me to explore it or actually learn it. Before I read this article, I had a vague notion of user stories. I thought of them a way to get in the skin of the worker to create functionality that addresses real needs. This article helped me refine that notion and was a perfect primer for someone who had only passing understanding of agile concepts.

I know you advocate this tool in the broader context of a total agile environment but I can use these concepts to make my requirement specs better. (Don't gag.)

I am glad I found this. Thanks for your contribution.

1 • Reply • Share ›

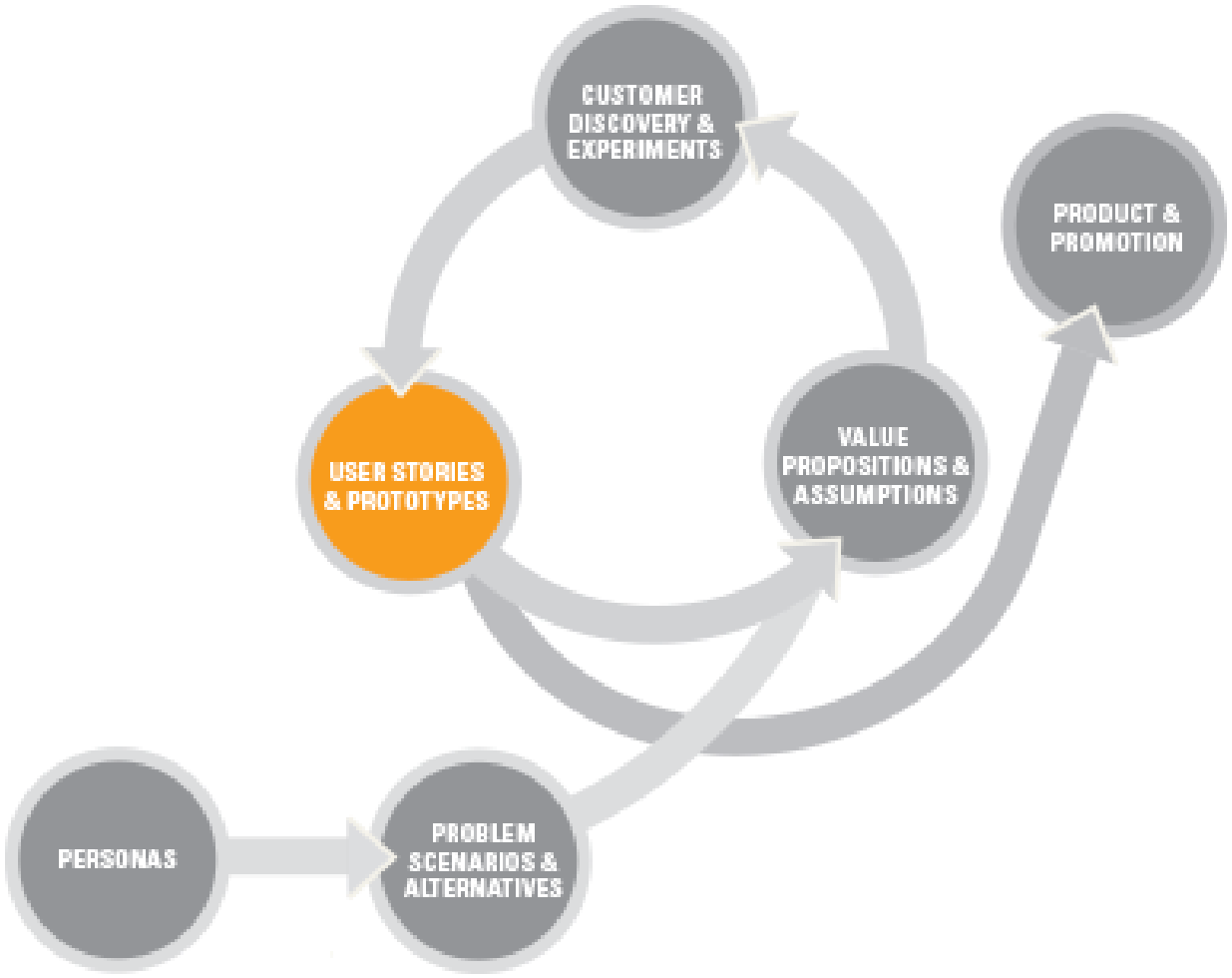
Alex Cowan Mod → David Siegel • 8 months ago

Hey David-

I'm so glad you found it helpful and that's great!

I think all the core pieces of agile are helpful, together or individually. One of the worst things that's happened to agile is that (in some quarters) there's this perception that it's a monolithic, all or nothing proposition. Good on you for trying it out in small bits!!

VENTURE DESIGN PROCESS:
YOU ARE HERE



OVERVIEW (http://www.alexandercowan.com/venture-design/)

START CREATING STORIES!

User Stories Template on Google Doc's (http://www.alexandercowan.com/user-stories-template-google-docs/)

This simple template will help you and your team get started with user stories.

CONTACT ALEX

Writing Better Agile User Stories (http://www.alexandercowan.com/workshops/writing-better-agile-user-stories/)

In two hours, I can help you and your team/group improve the quality of your agile user stories, leveraging best practices from design thinking and lean/Lean Startup.

CURRICULUM (HTTP://WWW.ALEXANDERCOWAN.COM/LEARN/)

Product Design (http://www.alexandercowan.com/learn/product-design/)
Venture Management (http://www.alexandercowan.com/learn/venture-planning-management/)
Product Development (http://www.alexandercowan.com/learn/product-development/)
Management Hacks (http://www.alexandercowan.com/learn/management-hacks/)
Software for Businesspeople (http://www.alexandercowan.com/learn/software-for-businesspeople/)
Marketing 2.0 (http://www.alexandercowan.com/learn/marketing-2-0/)
Operational Design (http://www.alexandercowan.com/learn/operational-design/)

EXECUTION TOOLS (HTTP://WWW.ALEXANDERCOWAN.COM/LEARN/EXECUTE/)

The 20-Minute Innovator (http://www.alexandercowan.com/learn/execute/20-minute-innovator/)
Startup Sprints (http://www.alexandercowan.com/learn/execute/startup-sprints/)

FOR YOU (HTTP://WWW.ALEXANDERCOWAN.COM/FOR-YOU/)

The Entrepreneur (http://www.alexandercowan.com/for-you/entrepreneur/)
The Intrapreneur (http://www.alexandercowan.com/for-you/intrapreneur/)
The Instructor/Mentor (http://www.alexandercowan.com/for-you/instructors-mentors/)
The Manager (http://www.alexandercowan.com/for-you/manager/)

THE BOOK

Starting a Tech Business (http://www.alexandercowan.com/starting-a-tech-business/)
Book Alex (http://www.alexandercowan.com/contact/)

ABOUT

Alex Cowan (http://www.alexandercowan.com/about/)
Blog (http://www.alexandercowan.com/blog/)
Calendar (http://www.alexandercowan.com/calendar/)
Legal (http://www.alexandercowan.com/legal/)

CONTACT

Contact Form (http://www.alexandercowan.com/contact/)
Alex on Twitter (http://twitter.com/cowanSF)
Alex on Facebook (https://www.facebook.com/StartingATechBusiness)
Alex on LinkedIn (http://www.linkedin.com/in/alexcowan)
Youtube Channel (http://www.youtube.com/user/AlexanderCCowan)
Google+

TOP