The Agile Organization

What we can learn from software development

snapp.6@osu.edu | Ohio State University Libraries

The Agile Organization...What we can learn from software development.

The outline for today's presentation:

1. What exactly is agile software development?
2. By extension, how does it help us define organizational agility?
3. Then I will turn it over to Terry [Reese] to talk about what makes Discovery an agile project.

A bit of background:

- Not many people know that I have a masters degree in sociology in addition to a degree in computer science and the two of them together has led me to be very interested in culture, structures, and methodologies.
- I stumbled upon Agile in 2007 when I was in [the College of] Arts and Sciences, and we started practicing Scrum--an Agile framework--on my team soon after.
- It was really successful, and agile software development became my area of expertise.

Now in the Libraries, we've been talking recently about being Lean and being Agile.

- We've just taken Yellow Belt training and learned about Lean Six Sigma, a framework of Lean, and agile software development has been heavily influenced by Lean Thinking.
- Secondly, we're engaged in the most agile project to date: the discovery project.
- And we've completed our Strategic Directions with Agile planning and operations as a focus area.

When we talk about agile management or agile planning or the agile organization today, we can trace its history directly back to a group of software developers in 2001.

When we first started practicing agile in [the College of] Arts & Sciences, it was a bit niche, and now it's become best practice in the majority of software development shops in the private sector.

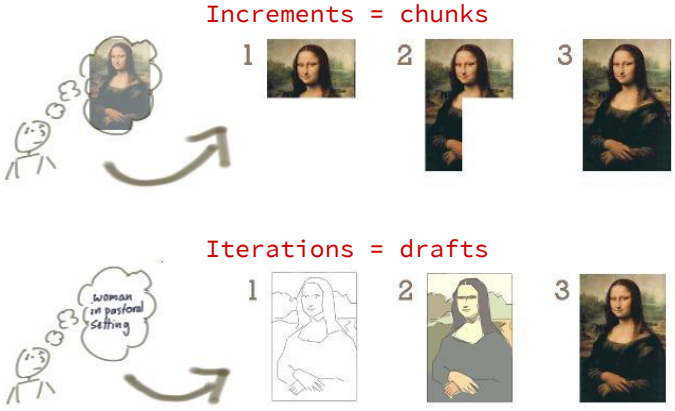
- There is still a lot of misunderstanding about what it means, especially in the last few years as it has become a mainstream concept in business in general, and has become a bit diluted.

- But most people would be able to say that agile development is iterative development.

Being Agile or Doing Agile?

Yes, Agile is iterative and incremental

But, not all iterative and incremental development is Agile.

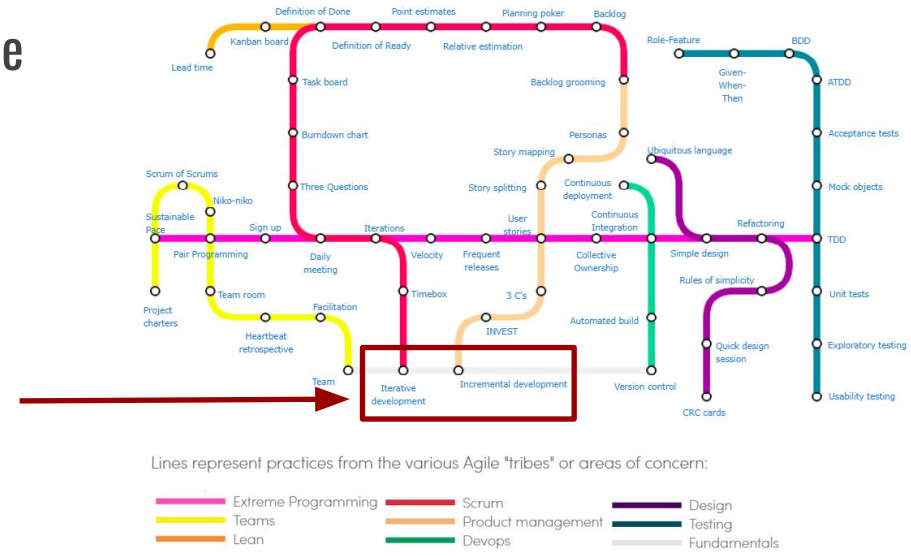


Images: https://jpattonassociates.com/dont_know_what_i_want/

I think that it's important to differentiate between practices and values.

- Yes, agile developers deliver products iteratively and incrementally. But not all iterative development is Agile.
- Iterations and increments (and there is a difference) are two desired and fundamental practices of agile software development.

Doing Agile



Images: https://www.agilealliance.org/agile101/subway-map-to-agile-practices/

And there are a whole lot of practices, as you can see on this map.

- The arrow points to increments and iterations.

Doing Agile

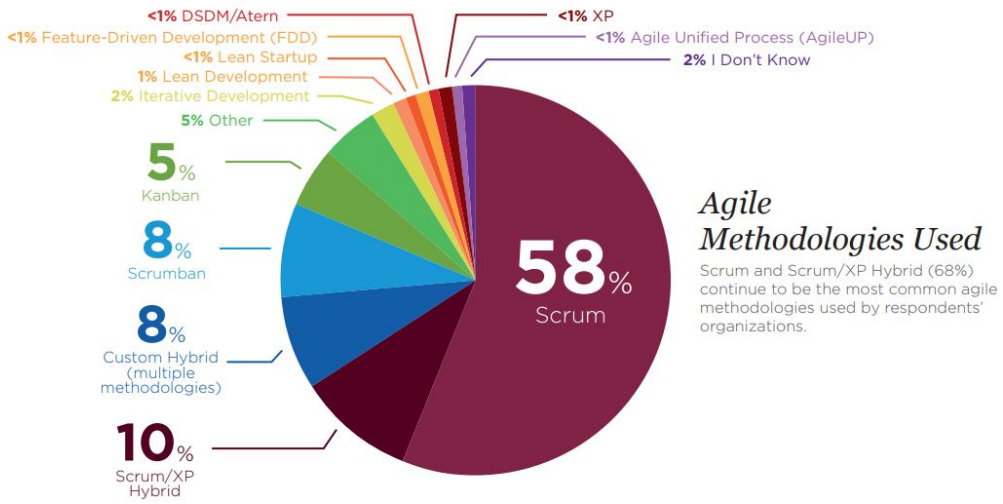


Chart: VersionOne 11th Annual State of Agile Report

Why so many practices?

- There are actually different agile frameworks or methodologies, such as Extreme Programming, Kanban, and Scrum and various hybrids.
- Scrum, by far, is the most popular. And that's the framework that we follow in AD&S [Applications Development & Support].

Agile: Scrum

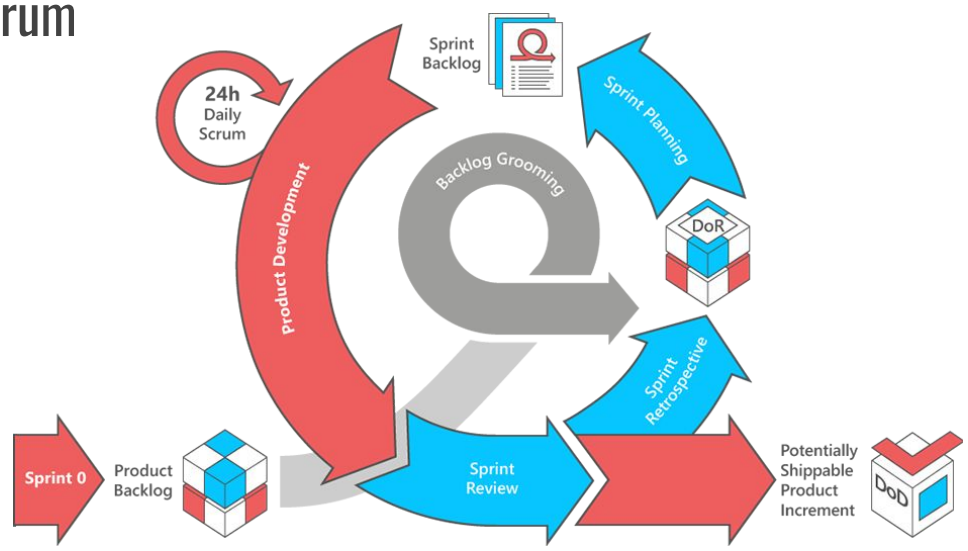


Image: http://www.scrumguides.org/scrum-guide.html

This is what Scrum looks like.

- Scrum is quite prescriptive and rule-based.

There are ROLES, ARTIFACTS, and EVENTS that are repeated in a time-boxed cycle.

- We have 2 week SPRINTS.
- We have the ROLE of the Scrum Master (also known as agile project managers) and the Product Owner--someone who provides us with subject matter expertise and represents the stakeholders and users when prioritizing features.
    - For the Discovery project, that's Terry [Reese]. For the website redesign, that's Robyn [Ness].

In terms of EVENTS:

- EVERY OTHER Friday, we all get together for a sprint review which is an opportunity for the developer to demonstrate what he or she has accomplished to the PO during the previous 2 weeks.
- Previous to the Friday, Scrum Masters have met with the product owners to review the PRODUCT BACKLOG of stories and select stories for the next sprint--called the SPRINT BACKLOG.
    - USER STORIES are very small chunks of requested functionality in non-technical language.

- In the sprint review meeting, Russell [Schelby] will review the objectives of the next sprint for each project.

- Every day, we have a daily standup to answer 3 questions: 1) what did I do yesterday? 2) what am I going to do today? 3) are there any impediments or blockers?

But we don’t follow all of the RULES.

- For example, we don’t estimate stories and derive tasks in a separate SPRINT PLANNING meeting or use burn down charts.

- We don’t ship an INCREMENT every 2 weeks.

- Our own invention: Maintenance Fridays to get caught up on non-project tickets.

It may look like chaos from the outside looking in--which probably means we’re doing it right--but a point I would like to make is agile frameworks are quite disciplined.

And Scrum is not just for software development:

- Russell [Schelby] and Cate [Putirskis] presented on how SCDA is using Scrum to manage their workflows.

Not-Agile

It's time for application leaders to recognize what the facts tell us: All other things being equal, traditional waterfall approaches are less reliable and more risky than a more duration-constrained approach. Gartner

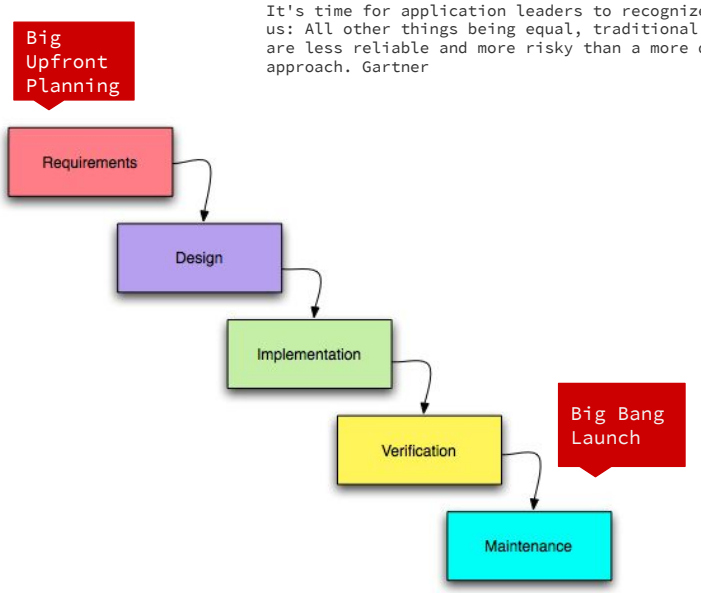


Image: http://www.umsl.edu/~hugheyd/is6840/waterfall.html

Now compare Scrum to the model of traditional project management--also called waterfall.

- It is sequential, phased, with upfront planning and a big launch at the end.
- There are handoffs to different groups at each phase--requirements from the business analysts to developers as specifications, into code to the testers.
- And because of the upfront planning and the handoffs, there tends to be a lot of documentation--maybe even more documentation than code.

Since we had been trained that this is the right way to do software development and project management, if we look at agile software development in retrospect, it has been a PARADIGM SHIFTER.

Being Agile

1. **Individuals and interactions** over processes and tools
2. **Working software** over comprehensive documentation
3. **Customer collaboration** over contract negotiation
4. **Responding to change** over following a plan

> Our highest priority is to satisfy the customer through **early and continuous** delivery of **valuable** software.

https://agilemanifesto.org (2001)

And it started with the Agile Manifesto of 2001 with 4 value statements and 12 principles--which defines what it means to BE AGILE--the Agile Mindset.

- This was a reaction against the heavy process of waterfall development which they believed was getting in the way of getting things done.
- They proposed that we start valuing people, working software, customer collaboration, and adapting to change more than tools, documentation, contracts, plans.

The first PRINCIPLE guides everything we do: early and continuous delivery of valuable software.

- “Early and continuous” explains iterations and increments as fundamental practices.

Early & Continuous Value

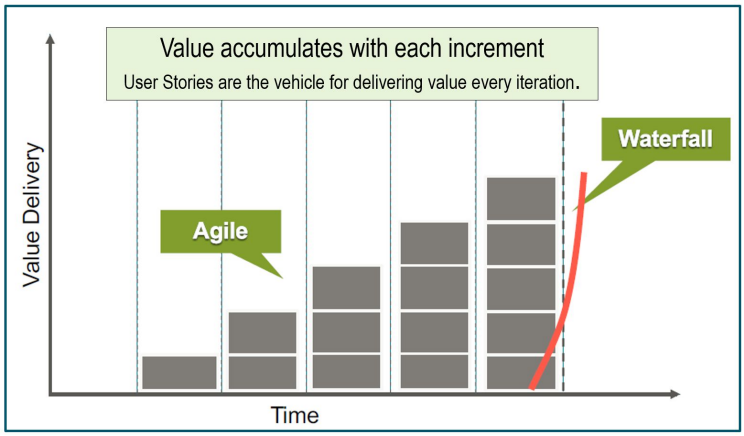


Image: Essays in agile software development (http://theburndown.com)

Why is early & continuous value given such a high priority?

- It makes more sense to deliver something early rather than waiting months or years to get something in front of the users.
- Delivering value to the business should be continuous, not a single point-in-time event.
- You can also reduce a lot of the riskiness of waterfall because we learn about potential problems earlier in the process.

Triangle of Constraints

What if we found ourselves building something that nobody wanted? In that case what did it matter if we did it on time and on budget?
Eric Ries

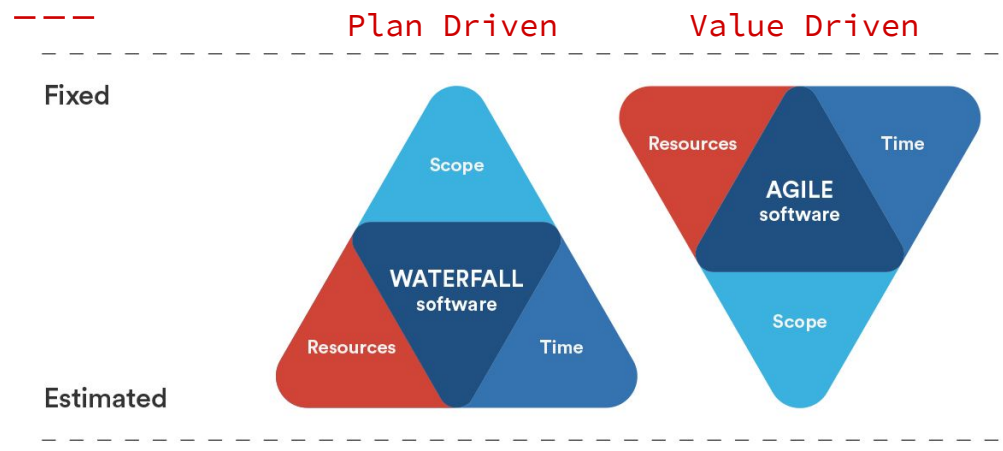


Image: https://www.atlassian.com/agile/agile-iron-triangle

That's the ESSENCE of the difference between waterfall and agile.

- Agile is VALUE-driven, not PLAN-driven.
- In terms of the TRIANGLE of constraints: scope, cost, time.

In TRADITIONAL project management, the requirements for functionality are fixed in advance.

- We ask you tell us everything you can possibly think of that you want this system to do.
- Then we project from that list of requirements, how long it's going to take and how many developers are going to be needed.
- This model can work well when you are working on something that has been done before, but when it's knowledge work with a high degree of uncertainty--like software development--it's very difficult to make accurate projections.
- Predictions inevitably are wrong and the project ends up being late and goes over budget--the stereotype of a software development project.

Agile flips the model UPSIDE-DOWN.

- We fix our resources and we fix the target date but scope (features) is not fixed.
- We continuously prune and reprioritize features as we get feedback and

- learn new things.
- We deliver not because the plan said so, but because we have determined that the selected features, based on all of the information we have collected up to that point, will provide the most value to the organization.

Another way to look at it is that waterfall is predictive. Agile is empirical.

Post-Agile is Lean

10 deploys per day (Flickr 2009)
23,000 per day (Amazon 2012)

- DevOps
- Continuous Delivery
- Lean Startup
- Lean UX

Image & Flickr Stats: https://www.slideshare.net/jallspaw/10-deploys-per-day-dev-and-ops-cooperation-at-flickr/76
Amazon Stats: IT Revolution DevOps Guide v1.pdf (https://dl.orangedox.com/devops-guide-v1)

2001 was a long time ago in internet time. How has agile changed since then?

- In 2009 an engineer from Flickr gave a presentation that became highly influential.
- Flickr was doing 10 deploys (code pushes to production) in per day. At the time, that was crazy fast.
- The argument was made that that agile thing was great for developers but they forgot about the people who actually supported their code in production.
- The only way to pull off that kind of speed was going to be cooperation between developers and operations.
- Ironically, Agile attempted to eliminate handoffs but didn't question that last handoff to production support.

The result has been what is called: DevOps--an extension of Agile--and it is gaining a lot of traction right now in the industry.

- It treats software development as a delivery pipeline owned by the entire project team including operations.
- It's about culture and engineering practices, such as automation.

Other important influences are CONTINUOUS DELIVERY and LEAN STARTUP.

- Lean UX--similar to DevOps--which is working to integrate UX into the software development lifecycle.

What can we learn?

1. Scope varies
2. Nimble teams
3. Adapt to change
4. Prioritize by value
5. Plan continuously
6. Lean is fundamental
7. Experiment to learn

Now that we have some historical context and know what it means to be agile in software development, what can we learn as an organization.

- I've boiled down to 7 core characteristics.

1. Scope varies

It's not that the schedule is overrun, it's that we are finding more things that need to be done. Documentary, The Shelbourne Hotel

- Deadlines are important *(if they're less than one month away)*
- Sprints = Timeboxes
- Every 2 weeks, we decide what is most important now

Image: Source unknown

We've learned that agile flips the iron triangle of constraints upside down.

- Deadlines are fixed and resources are fixed which is typical in higher ed.
- Scope (features) is what varies.

In an agile organization, we need to set and stick to deadlines, not as a punitive measure, but to force decisions about what's really important.

- In Scrum, checkpoints are built in every sprint as the opportunity for the product owner to reprioritize the backlog.
- Sprints are less than one month.
- Anything longer and time boxes lose their power as decision-making tools or a motivators.

2. Nimble teams

[O]rganizations which design systems ... are constrained to produce designs which are copies of the communication structures of these organizations (Conway's Law)

- Generalizing specialists
- Self-organization
- POs on the team
- Reduce handoffs
- Face-to-face is best
- Servant leadership

Image: https://www.flickr.com/photos/arizona_shona/305302247/

This principle is all about People and agile culture--individuals and interactions over processes and tools.

- Agile culture places a priority on self-organizing, autonomous teams.
- It is not command-and-control but rather, managers and ScrumMasters are servant leaders who protect and buffer the team and remove impediments to high performance.
- We also seek cross-functional teams of generalizing specialists: people who can play multiple roles--and we include a business representative on the team.

The idea is that the most successful projects and organizations will have nimble teams.

3. Adapt to change

The preference of an Originator is to challenge existing rules, politics and structures, resulting in fast, fundamentally different, even disruptive changes. Discovery Learning Inc.

- Why short increments are important
- Originators|conservers|pragmatists

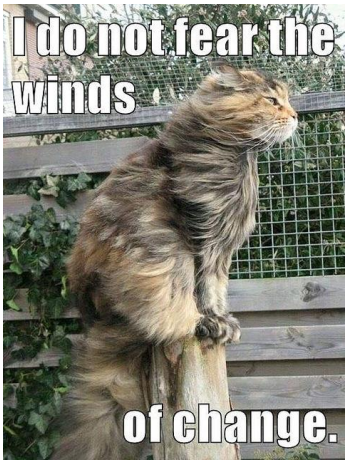


Image: https://plus.google.com/+LauraGibbs/posts/NnDRGYbRSfV

Software developers, and IT in general, are in the change business, so it's no surprise that embracing change is a core value.

- Many practices of agile frameworks are designed to account for change but also provide discipline.
- Working within time-boxed increments is an example: we can change course in a couple weeks if we really need to which is less costly than staying on path that is no longer valuable.

What can we do as an organization to adapt to change?

- If you remember the change style indicator assessment we completed in a training session--remember that we lined up in a row.
- The Originators in the crowd are the change agents in our organization.
- If organizational change is a goal, it would behoove us to look to Originators to lead by example in their approach to change.

4. Prioritize by value

[T]he enemy of Agility and Leanness is not scope creep, but *feature bloat* ... An Agile team adds business value by *not doing* the stories that represent bloat. Mark Schwartz

- Limited capacity is an opportunity
- Carefully choose what is most important

Image: https://www.flickr.com/photos/baggis/4967298714

We can't do everything and we'll never have enough people but what if we treated limited capacity as an opportunity, not necessarily a problem.

- If there is a limit to how many people we can hire, we will want to make sure to pick the work that is the most important and also eliminate any non value adding distractions.

So really, there is never too much work to do.

- It's all about making good decisions about priorities--brutal prioritization.

5. Plan continuously

No sense in being precise when you don't even know what you're talking about. Von Neumann

- Big upfront planning supports the "illusion of control"
- Agile = more time planning
- Adaptation, not fortune-telling
- Just in time vs just in case

Image: https://www.flickr.com/photos/jrladia/7245527662

The big upfront planning of traditional project management tricks us into thinking that we are in control of the project.

- We believe that the more we plan, the more successful our project will be.
- That is not always true.

But that doesn't mean we don't plan at all.

- It's a common stereotype of agile software developers that they don't plan which we saw from the Dilbert cartoon.
- Actually, there was a study that tracked different types of projects and the total number of hours engaged in planning was actually higher for agile projects--the key difference was when the planning was happening.
- It was happening continuously throughout the project.

As an organization, if we begin with the assumption that we aren't good fortune-tellers, we will want to plan just-in-time, making adjustments on the fly as we go to reach our target.

6. Lean is fundamental

It does not matter how fast we can build. It does not matter how fast we can measure. What matters is how fast we can get through the entire loop. Eric Ries

1. Eliminate waste
2. Amplify learning
3. Decide as late as possible
4. Deliver as fast as possible
5. Empower the team
6. Build integrity in
7. See the whole

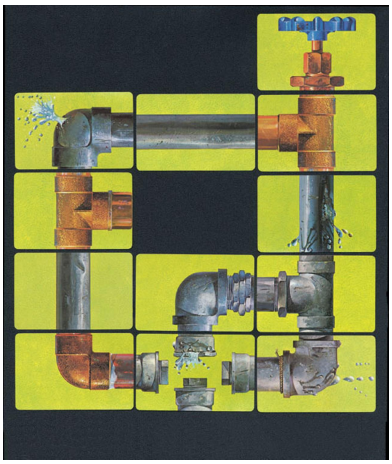


Image: JP Trostle, Waterworks game, Parker Brothers
Poppendieck, Mary, and Tom Poppendieck, Leading Lean Software Development: Results are Not the Point, 2009

To really understand what it means to be agile, we need to understand Lean.

- And we have learned what that means in our trainings: finding the bottlenecks in our workflows, eliminating waste in our processes, shortening feedback cycles.

7. Experiment to learn

Psychological safety is a belief that one will not be punished or humiliated for speaking up with ideas, questions, concerns, or mistakes. Amy Edmundson

- Get something real in front of real users: build-measure-learn
- Hypothesis-driven: are we building the *right* thing?
- Psychological safety is prerequisite

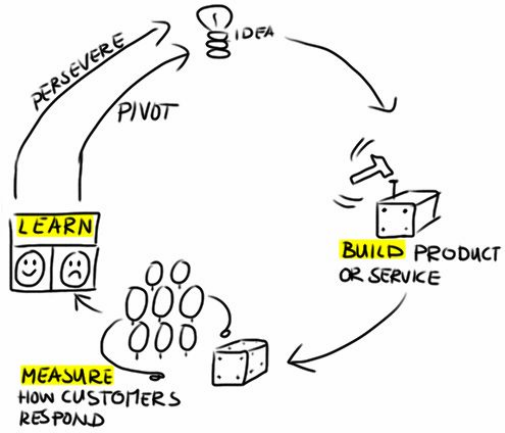


Image: http://theleanstartup.com/principles

And finally, experimenting to learn.

Lean Startup by Eric Ries is a book that has been highly influential in business and in software development--hypothesis driven product development--build measure learn.

- The unit of value in this case is not features but learning.

We talk about the learning organization: the agile organization is fundamentally a learning organization.

- There is a bias toward doing--building minimum viable products, getting them in front of real people to collect feedback and deciding whether to persevere or pivot.

The implication for an organization is that we need to be ok with less than perfect and less than complete.

- Sometimes things need to be perfect for safety or compliance reasons (air traffic control systems).
- Many times things don't need to be perfect.
- We need to be able to discern the difference.

And that requires a culture of psychological safety--safety to experiment.

Being Agile

as an organization

- ❏ Scope varies
- ❏ Nimble teams
- ❏ Adapt to change
- ❏ Prioritize by value
- ❏ Plan continuously
- ❏ Lean is fundamental
- ❏ Experiment to learn

To summarize, I think that being agile as an organization means being able to DISCERN:

- Discern when good enough, is good enough
- Discern what is most strategically important, and what is not
- Discern When planning crosses over into diminishing returns
- Discern When it's time to change, and when to stay the course

So how agile are we?

- I have put together an non-scientific organizational agility thermometer based on the 7 principles.
- I'm curious about how you all feel.
- And I'll send out a Qualtrics survey later and report back on the results.

Being Agile

as a project

- ❏ Scope varies
- ❏ Nimble teams
- ❏ Adapt to change
- ❏ Prioritize by value
- ❏ Plan continuously
- ❏ Lean is fundamental
- ❏ Experiment to learn

- **Anderson, David J.,** Kanban: Successful Evolutionary Change for Your Technology Business, 2010
- **Beck, Kent, and Cynthia Andres**, Extreme Programming Explained: Embrace Change, 2004
- **Brooks, Frederick,** The Mythical Man-Month, 1995
- **Cohn, Mike,** Agile Estimating and Planning, 2005
- **Duhigg, Charles**, What Google Learned from the Quest to Build the Perfect Team, The New York Times, 2016 https://www.nytimes.com/2016/02/28/magazine/what-google-learned-from-its-quest-to-build-the-perfect-team.html
- **Edmundson, Amy,** Building a psychologically safe workplace, TEDxHGSE, 2014
- **Griffiths, Mike,** PMI-ACP Exam Prep, Updated Second Edition: A Course in a Book for Passing the PMI Agile Certified Practitioner, 2015
- **Hotle, Matthew and Nathan Wilson**, The End of Waterfall as We Know It, Gartner, 2016
- **Jeremiah, John,** Survey: Is agile the new norm? https://techbeacon.com/survey-agile-new-norm
- **Kim, Gene, Kevin Behr, George Spafford,** The Phoenix Project: A Novel about IT, DevOps, and Helping Your Business Win, 2013
- **Kim, Gene, Patrick Debois, John Willis, Jez Humble,** The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations, 2016
- **Kniberg, Henrik,** Scrum and XP from the Trenches, 2007
- **Maurya, Ash,** Expose Your Constraints Before Chasing Resources, 2016 https://blog.leanstack.com/expose-your-constraints-before-chasing-additional-resources-cc17929cfac4
- **Poppendieck, Mary, and Tom Poppendieck,** Leading Lean Software Development: Results are Not the Point, 2009
- **Prince, Suzie,** The Product Managers' Guide to Continuous Delivery and DevOps, 2016
- https://www.mindtheproduct.com/2016/02/what-the-hell-are-ci-cd-and-devops-a-cheatsheet-for-the-rest-of-us/
- **Rework with Google**, https://rework.withgoogle.com/print/guides/5721312655835136/
- **Ries, Eric,** The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses, 2011
- **Rigby, Darrell K., Jeff Sutherland, Hirotaka Takeuchi,** Embracing Agile, Harvard Business Review, 2016 https://hbr.org/2016/05/embracing-agile
- **Royce, Winston,** "Managing the Development of Large Software Systems", Proceedings of the 9th International Conference on Software Engineering, 1987
- **Schwartz, Mark,** A Seat at the Table: IT Leadership in the Age of Agility, 2017
- **Vivian, Anthony**, The Lean UX Manifesto: Principle-Driven Design, https://www.smashingmagazine.com/2014/01/lean-ux-manifesto-principle-driven-design/

References & Influences