

# WORKING WITH AGILE PRODUCT TEAMS

*FINDING A COMMON LANGUAGE*

*Julia Harrison*

# What we'll cover

- About me
- About you - *quick survey*
- What is "agile"?
- Agile from the outside in
- What else? - Q&A

# About me

1994	Started tech career in desktop support
1997-2010	Various support roles and Windows integration projects
2010-2016	Mostly specialising in ITSM and Service Improvement projects
2013-2017	Agile product owner/product manager
2018	Head of Product

**Julia Harrison**  
[twitter.com/@JuliaFromIT](https://twitter.com/JuliaFromIT)

All opinions are my own, any examples do not refer to any specific client or employer, past or present.

In 2013 I was working with some project managers transitioning to being Scrum Masters. Some of the things I was hearing from them sounded a bit... odd. So when the opportunity came up to take Scrum Master training, my main motivation was to learn how to argue against what I thought sounded like nonsense!

After the course I read Ken Schwaber's book on Agile Project Management (<https://www.amazon.co.uk/Agile-Project-Management-Microsoft-Professional/dp/073561993X>) and started to see the value of what I'd learnt. Not long after I was given a service improvement project to fix a "broken" product – after a little digging, it turned out one of the biggest problems was the relationship between the development team and the internal customers. Putting in place a Product Owner seemed like a good place to start – I didn't have budget to hire anyone, so I 'temporarily' took on the role myself. I ended up managing that product for two years, and have been an agile product person ever since.

# What is agile?

*Sometimes it depends who you ask*

## Overheard in agile organisations

We're  
autonomous, you  
can't tell us what  
to do

We're agile, we  
don't write  
documentation

We can  
do more,  
faster

We did our agile  
transformation  
last year, it's  
finished now

There's only one  
way to do agile,  
and this is it

We won't  
need project  
managers

# What it means to be agile

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

Individuals and interactions over processes and tools  
Working software over comprehensive documentation  
Customer collaboration over contract negotiation  
Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

Manifesto for Agile Software Development [agilemanifesto.org](http://agilemanifesto.org) (2001)

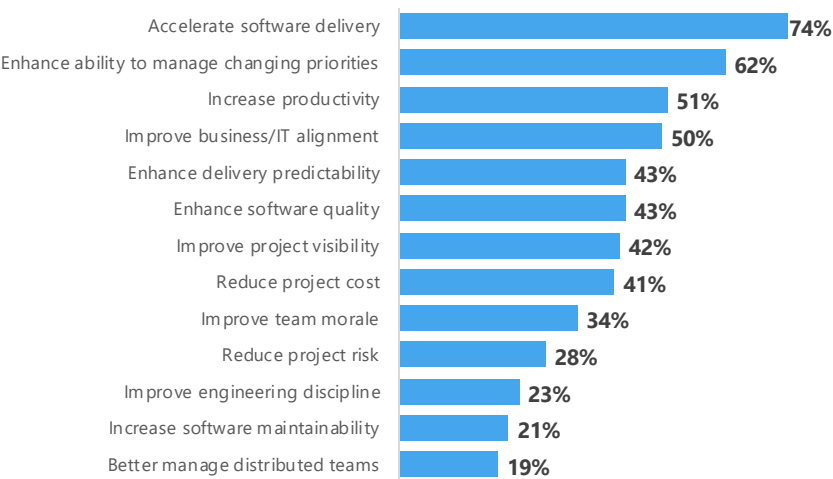
# What it means to be agile

- Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
- Welcome **changing requirements**, even late in development. Agile processes harness change for the customer's competitive advantage.
- Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
- Business people and developers must work together daily throughout the project.
- Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
- The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
- Working software is the primary measure of progress.
- Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
- Continuous attention to technical excellence and good design enhances agility.
- Simplicity--the art of maximizing the amount of work not done--is essential.
- The best architectures, requirements, and designs emerge from self-organizing teams.
- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Principles behind the Agile Manifesto [agilemanifesto.org](http://agilemanifesto.org)

# Agile in general

Reasons  
companies  
adopt agile

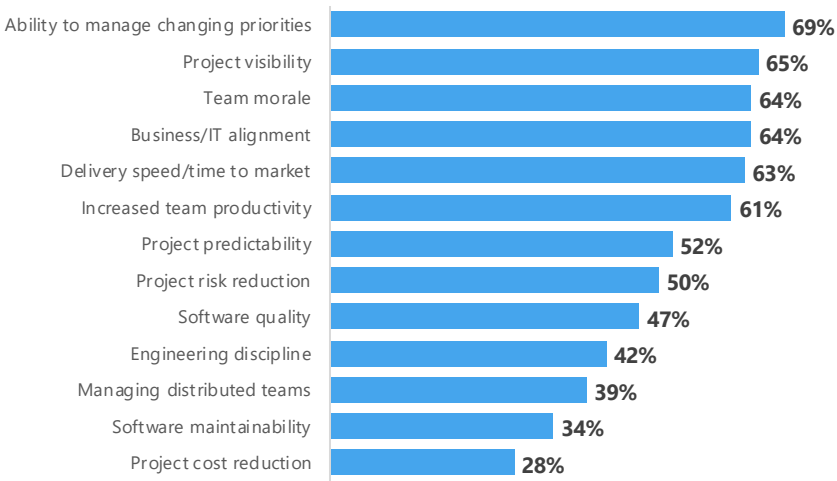


13<sup>th</sup> Annual State of Agile report, 2018 (1,319 responses) [stateofagile.com](https://stateofagile.com)



# Agile in general

## Benefits of adopting agile



13<sup>th</sup> Annual State of Agile report, 2018 (1,319 responses) [stateofagile.com](https://stateofagile.com)

## What it means (to me) to be agile

- The real value of what we make is only unlocked when people use it.
- Value is created soonest not by working faster, but by reducing batch size.
- We deliver more value not by doing more, but by reducing wasted effort on invalid assumptions.
- The start of any piece of work is when we know the least. That's why we sequence work to learn fast.
- We learn fastest by doing things and getting feedback on what we've done.
- We use a mix of quantitative and qualitative data to learn how people use our products and how we should improve them.
- We regularly reflect on how we work so we can improve.

All of the above is much easier if we communicate openly and trust each other.

Julia Harrison, from various sources, 2019

## Overheard in agile organisations

We're  
autonomous, you  
can't tell us what  
to do

We're agile, we  
don't write  
documentation

We can  
do more,  
faster

We did our agile  
transformation  
last year, it's  
finished now

There's only one  
way to do agile,  
and this is it

We won't  
need project  
managers

## Overheard in agile organisations

The organisation sets objectives, the team figures out how to meet them

We aim to do as much documentation as needed – no more, no less

We deliver value faster by working in smaller batches and reducing waste.

Agile teams and organisations are always improving how they work

A big part of agile is all about finding the best ways of working for you and your organisation

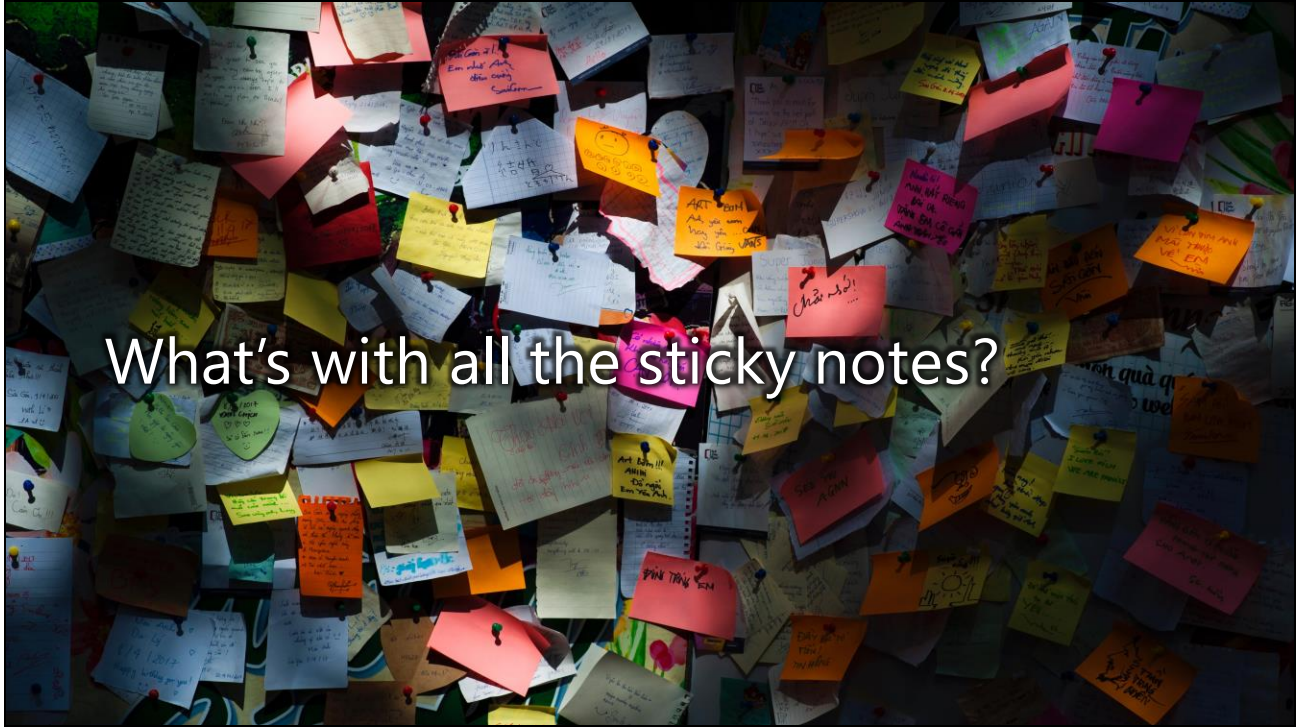
We work in multi-disciplinary teams where everyone is valued  
(but yes, former Project Managers will usually take a different role)

# Things that hinder or help

Hinder	Help
Funding based on "doing these things" <i>We will deliver X, Y and Z</i>	Funding based on "achieving these objectives" <i>We will improve customer retention by X%</i>
Top-down promises to deliver a thing on a date <i>In September we will launch X</i>	Talking about direction, what our priorities are <i>We're making it easier for users to do X</i>
Celebrating "doing the thing by the date" <i>Yay, we delivered this feature!</i>	Celebrating achieving objectives <i>Yay! 15% more customers say they're very happy!</i>
Writing lengthy requirements specs up-front	Focusing on what the product should do, not how it should do it - outcomes more than outputs
Trying to run projects the same as before, but in two-week bursts, with daily standups	Understanding that agile means working differently and thinking differently
The things on the left don't stop us, but they limit what we can achieve while we work around them	

# Agile from the outside in

*The challenges and (no, really) opportunities*



If you've ever thought that the agile movement looks a bit like a stationery-worshipping cult...

# What's with all the sticky notes?

1. Kanban: make work visible
2. Lean: reduce waste (spot bottlenecks)
3. Keeping it simple
4. We also like stationery



“make work visible” is an objective of Kanban, which derives from Lean. Both are very closely related to the agile movement.

When teams are located in one physical space, covering the walls in sticky notes is a very quick way to start organising work and making it visible. When we can't sit together we tend to use apps that mimic the layout of a physical Kanban board.



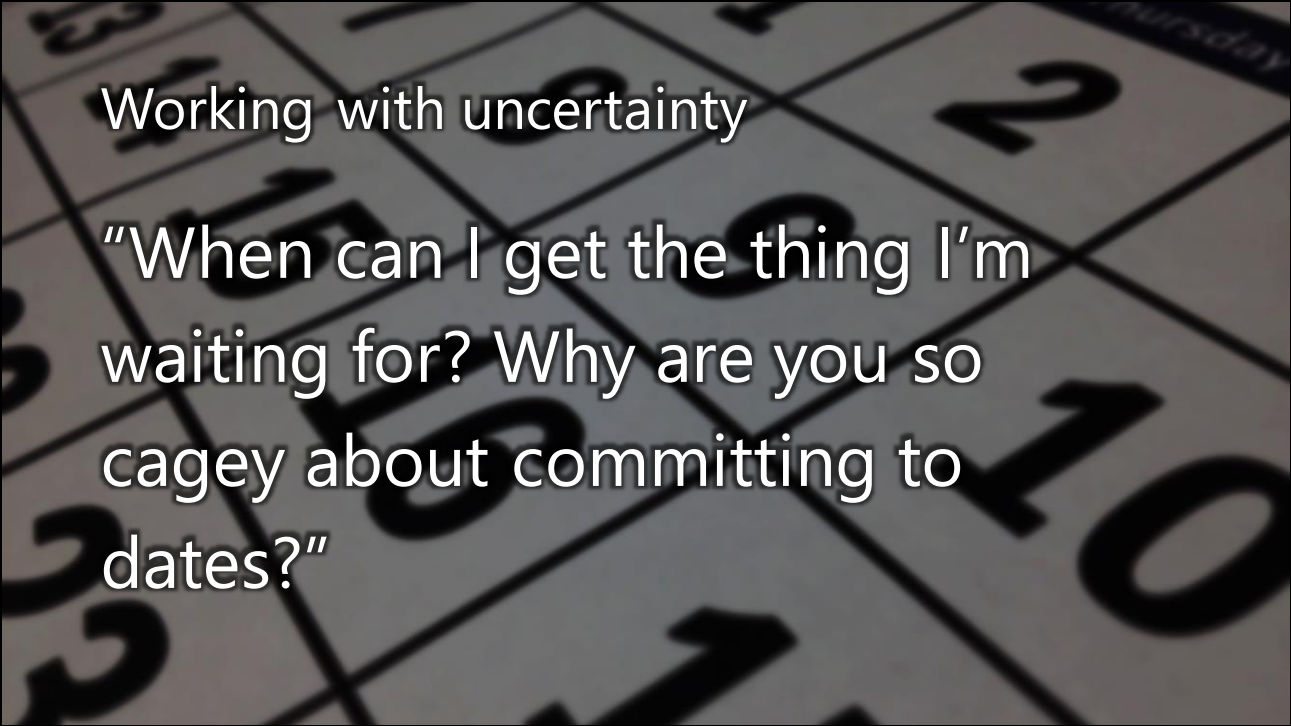
A close-up photograph of a woman with short brown hair and light-colored eyes. She has a shocked or overwhelmed expression, with her hands pressed against her temples. The background is plain white. Overlaid on the lower part of her face is the text: "Is it a product? Is it a service? What happened to projects?"

Is it a product? Is it a service?  
What happened to projects?

# Product vs. Project vs. Service

	Project (PRINCE2)	Service	Product
Team	A "temporary organisation"	Usually aligned by function, often across many services	Long-lived team looking after one product or feature
Starts with	Mandate (do a thing) Business case High-level plan	Service (business) requirements	Objective Hypothesis Experiments (MVP)
Ends with	Delivery is complete	Service retirement	Product retirement
Managed by	Stages Continual business justification	Periodic service reviews Ongoing measurement of SLAs, OLAs, contracts	User journey measures User satisfaction Business objectives
Measured by	Delivery of products on time and on schedule, to required level of quality	Service Level Agreements (SLAs)	Objectives and Key Results (OKRs)

Services and products are very similar in that they tend to be long-lived and provide an end-to-end user experience. The language we use and the way we organise around them and measure them is usually different.



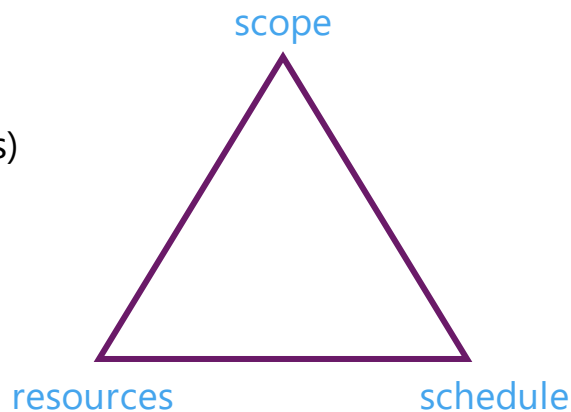
Working with uncertainty

"When can I get the thing I'm waiting for? Why are you so cagey about committing to dates?"

# Working with uncertainty

## The Iron Triangle

(you can only fix two points)



We need flexibility on at least one point of the triangle. Resources (people, or team size) is usually the most difficult and least useful to flex. Even if we add people to a team, the time it takes them to come up to speed sometimes means we don't get much benefit.

One thing we can be flexible on is schedule – this means we can give estimates on when something will be ready, but fixing a date means something else has to flex, usually scope (what we deliver). Often teams are reluctant to give estimates because they worry they'll be miscommunicated as a commitment. (Sometimes based on bitter experience!)

I wrote some more about this here: <https://medium.com/p/iron-triangles-and-vicious-circles-2398561f1fee>

# Working with uncertainty

User stories:

As a [type of user]

I need a [thing]

So I can [achieve my goal]

This gives the development team freedom to find a solution you might not have thought of

Writing user stories, which focus on the problem or need, rather than the expected solution, is a way to build in flexibility around scope.

Example: “As a service desk agent, I need a scrolling ticker so I can see status of ongoing major incidents”.

Let’s say there isn’t a scrolling ticker in use anywhere else in the product, so the development team doesn’t have a design pattern for it. Following good design practice, we’d want to get a user experience designer to work on that. If a designer isn’t available, maybe there’s another way for service desk agents to see the status of ongoing major incidents that could be delivered sooner.

Writing user stories instead of specifications makes it easier for the product team to suggest other solutions. In this case the product manager could sit down with someone from the service desk to figure out if there’s a different way to quickly achieve their objective.

# Working with uncertainty

Suggested conversations:

- If we can't do all of it by [date], it would be valuable if we can at least [enable users to do a thing]
- Can you give me an idea of a likely date range?
- Can we see your backlog so we know where we sit in the prioritisation?
- [In extreme cases] This is critically important. Who do we need to talk to about making this your top priority/getting more people to work on this?

On date ranges: it's important to build trust, so this might have to come with a promise not to communicate the shortest date as if it's a commitment! ("It's important you can trust me so I promise...")

# Trust and relationships

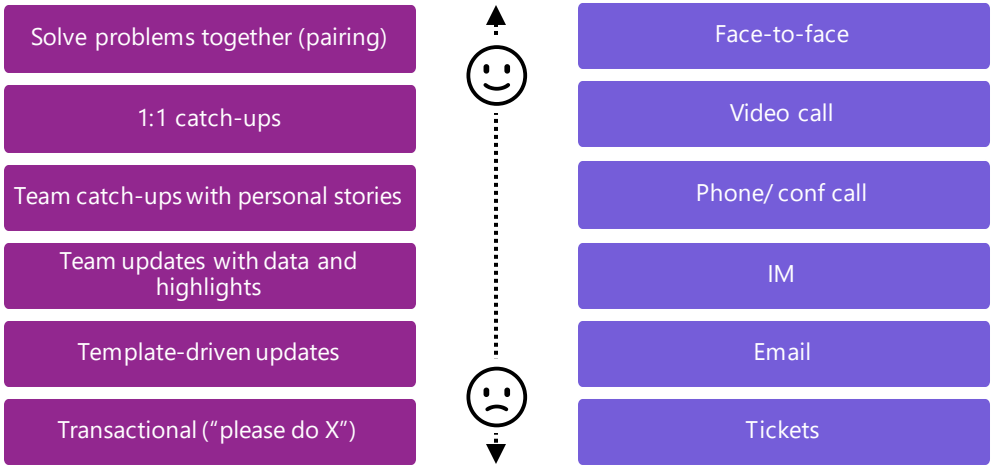
Opportunity:

- Get to know your product owner(s)
- Ask how you can help them
- Voice of the customer – use verbatim quotes
- Service desk data is product data
- Proactive problem management data is product **GOLD**
- How can your teams have more contact?

The service desk isn't the voice of the customers, but probably has frequent direct contact with them.

Verbatim quotes are powerful. They can grab their attention in a way that data alone sometimes can't. This is true not just for your communications with the product team, but also when information is passed to decision-makers up the chain.

# Building relationships



How we communicate is important.

On the left we have the types of communication, on the right we have the communication media.

From bottom to top we increase the usefulness of each type of communication in building relationships.



# Building relationships

	Tickets	Email	IM	Phone/ conf call	Video call	Face-to- face
Solve problems together (pairing)		😊	😊	😊	😄	😄
1:1 catch-ups		😊	😊	😊	😊	😄
Team catch-ups with personal stories		😊	😊	😊	😊	😊
Team updates with data and highlights		😐	😐	😊	😊	😊
Template-driven updates		😐	😐	😐	😐	😊
Transactional ("please do X")	😞	😐	😐	😐	😐	😊

When we multiply them together, we get the greatest opportunities to build relationships at the top-right corner of this chart.

I wrote some more about this here: [https://medium.com/swlh/building-relationships-across-teams-90ffa8baf69e?source=friends\\_link&sk=1d300b2e4b2f1837fccd36d754af6b8a](https://medium.com/swlh/building-relationships-across-teams-90ffa8baf69e?source=friends_link&sk=1d300b2e4b2f1837fccd36d754af6b8a)

## Prioritisation and MoSCoW

“Why does development stop after the bare minimum has been delivered?”

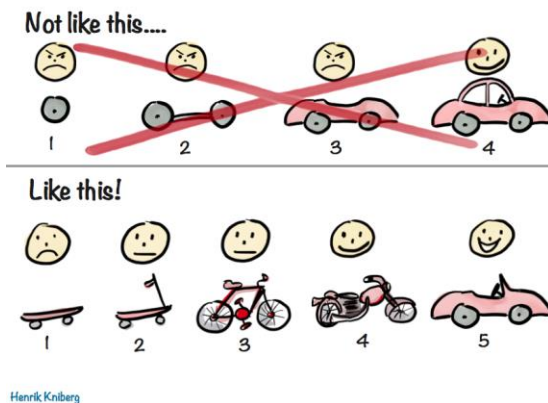
Answer:

## Prioritisation and MoSCoW

- **Must** have
- **Should** have
- **Could** have
- **Won't** have this time

We **MUST** have our **Musts**,  
we **SHOULD** have our **Shoulds**.  
(This is not universally  
observed.)

# Prioritisation and MoSCoW



## Minimum **Viable** Product



what's viable in  
your context?

This diagram is often used when explaining agile. The idea is that we should deliver something small that meets the users need (get from A to B faster) and build on it, rather than build the 'finished product' in increments which only deliver user value at the end. You can read Henrik Kniberg's post about this here:

<https://blog.crisp.se/2016/01/25/henrikkniberg/making-sense-of-mvp>

The problem is that the diagram is often misunderstood. The skateboard is great if you're the chief innovator in a Neolithic village and the wheel was just invented last week. If you're Toyota, probably not so much. The point is, by putting something in the hands of the user, we'll learn quickly about what works for them and what doesn't.

What's viable is different across different contexts.

## Prioritisation and MoSCoW

Suggested conversations:

- What objective are we trying to meet?
- Here's what I'm being measured by. What are you being measured by?
- We're hearing from users that [...] – if I send you some data will it help prioritise the improvement?

Often the product team are under pressure from elsewhere in the organisation to move on to delivering the next “thing” quickly. They may be as frustrated as you are that they can't deliver the thing you want.

These conversations might help find something that's viable for you or your customer and achievable for them, or possibly help them with prioritisation against other objectives.

## Key takeaways

- You probably want the same thing – happy users.
- Sometimes you're measured by different things.
- Product owners love data about their users. And you have data!
- Ask for things as outcomes rather than outputs  
*("users want to understand their bill" over "users want pie charts")*
- Everything is easier if you can build relationships and get to know each other.