ay_of_the_web_tester) (/about)

Getting Started (/) Videos (/videos) Courses (/bootcamp)

Learn More

Agile Myths

Over the years several myths have formed around Agile delivery. Here are some of the more popular ones.

- Agile is a silver bullet.
- Agile is anti-documentation.
- Agile is anti-planning.
- Agile is undisciplined.
- Agile requires a lot of rework.
- Agile is anti-architecture.
- Agile doesn't scale.

Agile is a silver bullet

I wish this were true - but it isn't. You can fail just as spectacularly on an Agile project as you can using any other traditional method. You'll fail faster using Agile (due to the transparency and visibility it brings) but unfortunately it's not a silver bullet or an excuse to stop thinking.

There's nothing inherently magical about Agile. It basically says

> Bring your development team and customer as close together as you can, give them what they need, and then get out of the way.

Now if you don't have people that like being empowered, taking initiative, and getting things done, that's a different problem. Agile just gives them permission to do their best work and be accountable for the results.

Agile is anti-documentation

Agile isn't anti-documentation. A more accurate way to say it would be Agile doesn't do documentation for documentation's sake.

Documentation gets treated like any other deliverable on an Agile project. It gets estimated, sized, and prioritized like any other user story (/user_stories).

Where Agile pushes back on documentation is as a means of communication. Agile prefers face-to-face communication over relying on the written word.

Agile is anti-planning

Not sure where this one comes from. There's actually a lot of planning that goes on in Agile projects.

You've got your:

1. Daily planning with the 10 minute daily standups.
2. Bi-weekly planning with the Iteration/Sprint Planning Meetings

3. Release planning where team's decide what to ship every three to four months.

But it wouldn't be fair to say Agile is anti-planning. If anything it is anti-static planning. Meaning Agilist's expect their plans to change and use tools like burndown charts to track and make these changes visible.

Agile is undisciplined

When Agile started gaining popularity, its reputation suffered a bit from some teams taking the easy parts of Agile (like attending daily standups) but leaving out the hard (like upfront testing and regularly shipping production ready working software).

The truth is Agile is a very disciplined way of delivering software.

- You have to test.
- You have to get feedback.
- You have to regularly ship software.
- You have to change and update the plan.
- You have to deliver bad news early.

This isn't easy stuff. It's not for the faint of heart and requires a lot of hard work, courage, and discipline.

Agile requires a lot of rework

Rework comes in two forms on an Agile project. You've got the rework of requirements - customers discovering what they really want. And you've got the rework of the software - development teams discover better ways to design the software.

Both need to be balanced and tempered. Just as business can't indefinitely keep changing their mind, development teams can't forever keep redesigning the software. At some point we have to ship.

Agile deals with this tension by empowering both sides with the power to iterate, so long as they work within the project's means.

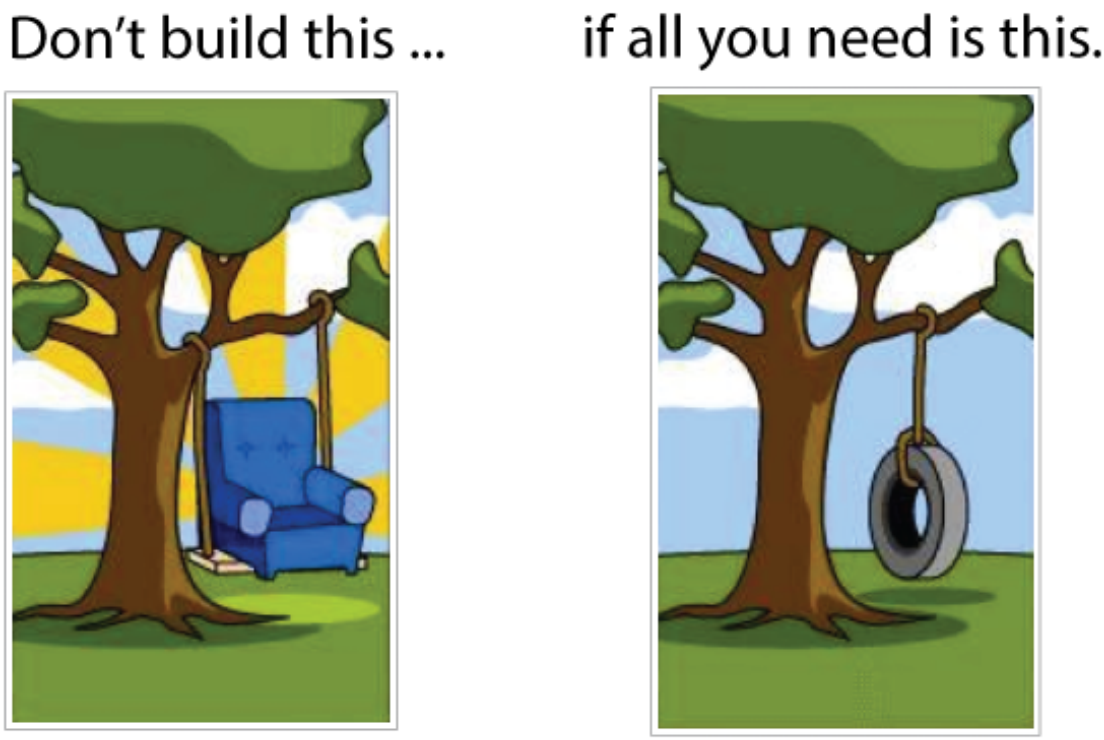
Burndown charts play in big role in tracking how Agile project are doing. Just as tools like the Agile Inception Deck (http://agilewarrior.wordpress.com/2010/11/06/the-agile-inception-deck/) make sure everyone is on the same page with regards to time and money.

It's a balancing act not unique to software delivery. Any creative work with a deadline (i.e. plays, movies making, or the publishing of daily papers) faces the same challenges.

The trick is to do the best work you can, with the time and resources you've got.

Agile is anti-architecture

Something we got really good at as an industry in the 1990's was building big, complex, expensive, hard to maintain systems.



Agile pushed back on this over engineering by creating terms like YAGNI (You Aint Gonna Need It) (http://en.wikipedia.org/wiki/You_aren't_gonna_need_it) to remind teams to keep things simple until proven otherwise.

That doesn't mean Agile teams stop thinking, or don't leverage previous experiences.

It's more an attitude that the best way to build systems is to keep things simple, and only add the complexity when you need it.

Agile doesn't scale

Agile scales like any other software delivery process. Not that well.

Look - scaling is hard. There is no easy way to magically coordinate, communicate, and keep large groups of people all moving in the same direction towards the same cause. It's hard work.

The one thing Agile does bring to the conversation, is instead of looking for ways to scale up your project, look for ways to scale things down.

In other words, if we know we are really good at delivering with small, nimble, agile teams of ten, why don't we structure our work that way. More on this here (http://martinfowler.com/articles/canScaling.html).

← Previous (/how_is_it_different)

Next → (/agile_vs_waterfall)

IN A NUTSHELL

What is Agile? (/what_is_agile)

How does it work? (/how_does_it_work)

How is it different? (/how_is_it_different)

Agile Myths (/agile_myths)

Agile vs Waterfall (/agile_vs_waterfall)

FUNDAMENTALS

User Stories (/user_stories)

Estimation (/estimation)

Iterations (/iterations)

Planning (/planning)

ENGINEERING

Unit Testing (/unit_testing)

Refactoring (/refactoring)

Continuous Integration (/continuous_integration)

Test Driven Development (/test_driven_development)

CONCEPTS

Burndown Charts (/burndown)

Cone of Uncertainty (/cone_of_uncertainty)

Management by Miracle (/management_by_miracle)

Three Simple Truths (/three_simple_truths)

XPISMS

Bill of Rights (/bill_of_rights)

YAGNI (/yagni)

Yesterday's Weather (/yesterdays_weather)

The Simplest Thing (/simplest_thing)

Testing (/could_possibly_break)

Production Readiness (/production_readiness)