Six Agile Product Development Myths - Busted

by Mike Cohn • 46 Comments

Six Agile Product Development Myths - Busted

Organizations adopt an agile approach for many reasons. Some hope for greater productivity and a reduced time-to-market. Others for more successful products. Still others adopt agile to increase collaboration between developers and business people, to improve quality, or to increase team member job satisfaction.

And, of course, many organizations adopt agile in the hope of achieving some combination of these benefits.

But, for as many benefits as there can be to agile, there seem to just as many fallacies. In this article, I'd like to bust six myths about agile product development.

Myth #1: Isn't Agile Just for Software Development?

This is an intriguing myth because Scrum is the oldest agile approach and Scrum's origins are in physical product development. Some of the original Scrum projects were photocopiers, a Honda car, and cameras.

Agile these days is used for all forms of product development, from physical products to cloud-based software-as-a-service.

But beyond product development (both hardware and software), agile is being used successfully by

- Marketing teams to plan and execute campaigns
- Attorneys to manage cases and workload

- Organizational transformation efforts, in particular when transitioning to agile
- Senior leadership teams to manage their organizations
- Families for improving time together
- Couples planning weddings

And, of course, many more. Any project with a high degree of uniqueness (you haven't done it ten times before) and complexity is well suited for agile. If you find yourself hung up by references to software in published in documents such as the Agile Manifesto, just substitute the word *products*. Change *working software*, for example, to *working products* and read it again: [We value] working products over comprehensive documentation. It fits perfectly. Why? Because agile works with all kinds of products--software is only one of them.

Myth #2: Is It True That Managers Have No Role in Agile?

I think this myth persists because too many managers spend a lot of time doing things they shouldn't be doing, regardless of being agile or not.

For example, managers often work down at the level of assigning tasks to individuals. When they learn this isn't something they should do in agile, they feel like part of their job has been taken away.

No: Part of the job has not been taken away. Things like assigning tasks should never have been part of the job in the first place.

Managers should be focused on creating the environment and culture a team needs to thrive. Their time should not be consumed with minutiae like who will work on which task.

Peter Drucker was perhaps the leading management theorist of the 20th century. He is perhaps best known for the idea of management by objectives and creating the acronym SMART for goal setting in which goals are Specific, Measurable, Acceptable, Realistic and Time-bound). Drucker said managers have five jobs:

- Set objectives
- Organize people and work
- Motivate and communicate

- Measure
- Develop people

Sure, in some organizations, some of these responsibilities may be diminished. But others--such as developing people--become more important.

In all the years I've worked with agile and agile organizations, not one has decided to fire all managers. Yes, some managers have moved into the more focused roles of Scrum Master or product owner, but there is still a role for management in agile.

Myth #3: Can't Stakeholders Introduce Change Whenever They Want?

Not surprisingly, the myth that stakeholders can introduce change any time they want is most commonly believed by stakeholders themselves.

Development team members understand that introducing change at the wrong time comes with a cost.

Consider ordering a meal at a restaurant. You tell the waiter you'd like the chicken. Then instantly say, "No, make that the salmon instead."

There is no cost to this change.

There is, however, a cost if you change your mind later. If you tell the waiter to change your meal from chicken to salmon after the kitchen has begun cooking the chicken, there is the cost of wasted food (which the restaurant may well charge you for).

The cost is more obvious if you eat half of the chicken before telling the waiter you'd like to switch to the salmon.

A stakeholder introducing change into an iteration is like the diner changing from chicken to salmon. If the change is introduced at the right time, there may be little or no cost. Introduced at the wrong time, though, and there is a cost.

Being agile cannot eliminate all costs of stakeholders introducing change. However, good agile teams can reduce the cost of change regardless of when the change is introduced. Common ways of doing this are:

- short iterations
- small product backlog items
- Finishing each product backlog item as quickly as possible, usually by minimizing the number of items being worked on concurrently

None of this is to say that teams shouldn't welcome appropriate changes. Some stakeholder-requested changes can be very important. But, the benefit of making each change needs to be assessed against the cost of changing and that cost is not always zero.

Myth #4: Doesn't Everyone Need to Be a Generalist on an Agile Team?

For some reason, a popular myth about agile is that every team member needs to be able to do everything.

This myth implies that if you have hired the world's greatest Oracle database administrator that you want that DBA to also be an amazing JavaScript developer. And your amazing JavaScripter should be fully proficient at security testing in case there isn't much JavaScript needed one iteration.

This is entirely false.

Agile teams don't need everyone to have every skill. Instead, what agile teams need is to value any individuals who do possess skills in multiple disciplines.

To understand the falseness of this myth, consider a well-run fancy restaurant. From watching too many TV cooking shows, I've learned such a restaurant may have a *pastry chef*. The pastry chef is skilled in making pastries, desserts, bread and other baked goods.

This sounds like a highly skilled but specialized role. Another specialized role in the kitchen could be the *saucier*, who prepares sauces, stews, and other similar items.

In a well-run kitchen, it would be nice if the pastry chef could help the saucier, perhaps by slicing some onions during a sudden onion shortage emergency. But neither the pastry chef nor the saucier would be expected to be fully capable of doing job of the other.

In today's world of complex technologies, it is unrealistic to expect team members to be fully proficient in all the skills of the team. Instead, good agile teams learn to value multi-skilled team members.

Having a few team members with multiple skills helps manage the balance of types of work. That is, sometimes the team needs more testing capacity. Having a team member or two who can shift into doing helps incredibly.

But this can be accomplished on most teams even if a few team members are truly specialists and adept at only one discipline.

Myth #5: I've Heard that Agile Teams Don't (or Can't) Plan.

Most good agile teams do plan. But that planning is often less visible than on traditional projects because agile teams don't have an upfront *planning phase*.

Instead, good agile teams conduct planning as a series of smaller, recurring activities that ensure their plans always reflect the realities of the current situation.

In this way, teams develop plans the same way they develop products: by inspecting and adapting.

For any agile team, its plan should project no further into the future than absolutely necessary for making important decisions. But most organizations and teams have a plan to approximate at some level what is coming next and when it's coming.

In fact, despite this myth, agile teams have an easier time creating reliable plans because agile teams get earlier insight into how quickly they are producing software.

Consider a traditional team with analysis, design, coding and testing phases. If lucky, that team may delay committing to a plan until the end of the design phase. But at that point, this team has no idea how fast they are at coding and testing--they haven't done any of those activities yet.

In contrast, an agile team turns the crank of the entire development process every iteration. Each iteration includes a little analysis, design, coding, and testing. This gives the agile team more and earlier insight into how quickly it can turn ideas into new features.

Myth #6: Don't Agile Teams Create Products with No Architectural Plan?

It's time to bust our final myth: the myth that agile teams don't architect or design their products.

Agile teams definitely design their products. But, in the same way they plan incrementally, agile teams architect and design incrementally. This allows them to inspect and adapt their architectures and designs so they become the best possible.

This means there is no upfront phase during which all architectural decisions are made. Instead the architecture of a product *emerges* over time.

This occurs by technical team members focusing first on any aspects of a product they consider risky. For example, if delivering a product with the needed throughput will be challenging, the team and product owner would elect to work on functionality early on that reduces that risk.

In this way, the emergent architecture of an agile product is also *intentional*. The architecture doesn't just show up one day. It emerges gradually and guided by the intent of the technical

team members.

This means that agile products do possess an underlying architecture. But decisions about that architecture are made as needed through the course of the project rather than being made entirely at the start of the project.

What Other Agile Myths Need Busting?

I've busted six myths surrounding agile product development. What other myths have you encountered in your organization? Please share your thoughts in the comments below.

Posted: February 19, 2019

Tagged: management, planning, musings, specialist, stakeholders, architecture, design, manager, myths

---