Why Getting to Done Is So Important

by Mike Cohn • 42 Comments

Why Getting to Done Is So Important

One of tenets of Scrum is the value of getting work done. At the start of a sprint, the team selects some set of product backlog items and takes on the goal of completing them.

A good Scrum team realizes they are better off finishing 5 product backlog items than being half done with 10.

But why?

Faster Feedback

One reason to emphasize getting work to done is that it shortens feedback cycles. When something is done, users can touch it and see it. And they can provide better feedback.

Teams should still seek feedback as early as possible from users, including while developing the functionality. But feedback is easier to provide, more informed, and more reliable when a bit of functionality is finished rather than half done.

Faster Payback

A second reason to emphasize finishing features is because finished features can be sold; unfinished features cannot.

All projects represent an economic investment--time and money are invested in developing functionality.

An organization cannot begin regaining its investment by delivering partially developed features.

A product with 10 half-done features can be thought of as inventory sitting on a warehouse floor. That inventory cannot be sold until each feature is complete.

In contrast, a product with 5 finished features is sellable. It can begin earning money back against the investment.

Progress Is Notoriously Hard to Estimate

A third reason for emphasizing getting features all the way to done is because progress is notoriously hard to estimate.

Suppose you ask a developer how far along he or she is. And the developer says “90% done.”

Great, you think, it’s almost done. A week later you return to speak with the same developer. You are now expecting the feature to be done--100% complete. But the developer again informs you that the feature is 90% done.

How can this be?

It’s because the size of the problem has grown. When you first asked, the developer truly was 90% done *with what he or she could see of the problem*. A week later the developer could see more of the problem, so the size of the work grew. And the developer is again confident in thinking 90% of the work is done.

This leads to what is known as the 90% syndrome: Software projects are 90% done for 90% of their schedules.

Not Started and Done

In agile, we avoid the 90% syndrome by making sure that at the end of each iteration, all work is either:

- Not started
- Done

We're really good at knowing when we haven't started something. We're pretty good at knowing when we're done with something. We're horrible anywhere in between.

What's Your Experience?

Have you experienced problems with teams being 90% done? How have you overcome these problems? Please share your thoughts in the comments below.

Posted: April 11, 2017

Tagged: sprinting, definition of done, feedback, done, finishing work

---