Sarah Wood

Tools for Software Development

ACC 1-1

1. How do you think Agile methodology can improve the software development process

Agile methodology does improve the software development process by keeping the customer in the foreground of view. You are building for them, right? Agile emphasizes delivering tested, reliable software on a regular basis. Gone are the days of a six-month (or longer!) build where the developers burrow into their nests and make something that

Isn’t what the customer wanted

Doesn’t work reliably

Is a nightmare to read, maintain, or add to

Has become obsolete while it was being created

Is impossible to change: after all, the spec is set in stone, right?

Building something small, solid and modular is a bit like making a LEGO brick. If the customer asks for a house, you can show that you are building bricks and stacking them to look like parts of a house. When the inevitable changes happen, you can make more bricks, or rework your existing LEGOs to fit the new spec. You know you can rely on your LEGOs because you built them to be reliable, and you tested every brick before it left the brickworks.

1. Why do you think that not every software developer likes to use Agile methodology

Many developers I have worked with in the past have worked in larger organizations. They have been so far removed from the customer and end-users that they don’t even give them a second thought. They have been trained to work from a spec generated from another department that was written by someone who has never written a single line of code. They don’t test their own code: there is another whole department for that. Debugging is a thing you do just before you go into production: some companies make a month-long party of it, with incentives and prizes!

Apart from organizational and historic factors, some developers just like to be left alone to do their “thing”. Give them a spec, and they’ll just write to that. The customers don’t really know what they want anyway. What did the testing reports say? We’ll fix those bugs, but we won’t go looking on our own; heaven knows what we might find. Change, no thank you. We’ve been building this thing for (X amount of time made to sound impressive), there’s *no way* we can change now.

The irony of all this is that every one of those individuals misses the “old days” of writing code in small teams, in Joe’s Garage, really getting into what they were doing. They miss having control over their own workplace, and approaching the full development “stack” as a small team, playing to strengths found in that team. That’s why so many career developers are found writing code on the side for open-source projects. Many even do anonymously, so as not to get fired from their day jobs. Despite decades of conditioning, they want a project they can care about.

1. Do you believe using Agile will produce better software in less time, if so why (or why not).

Agile probably isn’t going to de-throne Waterfall for large, clearly-defined processes or off-the shelf software products. Then again, where’s the fun in that? Any project that is small by nature, or can be broken down into components, can be improved with Agile methodologies. At its heart, Agile is about doing what works. Remember who you’re building all this for.

Certainly in the case given for this course, an Agile team would produce better software much faster than a rigid Waterfall organization. It’s small, easily broken into components, and we have an involved, motivated customer. We would regularly meet with the customer to show off what we made this week, and check-in to make sure we’re going in the direction that the customer actually wants. We are all the departments in the stack: that alone is a massive savings in time, trouble and miscommunication.

In short, Waterfall documents are usually set in stone. Agile index cards are easy to change and move around to fit what we need today and adapt to what we’ll need tomorrow.