Agile Teamwork:

3 Ways to Minimize Handoffs

by Mike Cohn •

originally published in Better Software on 2010-04-05

Agile teamwork can really help to minimize handoffs. Teams using a sequential development process have become accustomed to handoffs between specialists. Analysts hand their work to designers, who hand it to programmers, who pass it on to testers. Teams that are new to Scrum often do not go far enough in eliminating these handoffs, wrongly assuming, for instance, that the programmers should finish programming a product backlog item before handing it off to the testers or that analysts should work at least one sprint ahead of the rest of the team.

High-performing Scrum teams, however, have learned to do a little bit of everything all the time during a sprint, thereby eliminating large handoffs. To do this effectively, scrum teams must make three changes: favor talking over writing, make handoffs very small and very often, and mix the size of items that are brought into each sprint.

Agile Teamwork - Stop Writing and Start Talking

There's a grand myth about requirements: If you write them down, users will get exactly what they want. That's not true. At best, users will get exactly what was written down, which may or may not be anything like what they really want. As such, Scrum teams forego a lengthy, up-front requirements phase and the resulting product specification in favor of a dynamic product backlog.

Because Scrum teams shift focus from writing about requirements to talking about them, conversations with the product owner—rather than a product spec—become the primary way of finding out how a new feature should behave. Team members talk with the product owner about how a feature should work, how quickly it should perform, what acceptance criteria must be passed, and so on. Scrum team members also talk with users, customers, and other stakeholders as appropriate and, perhaps most importantly, with each other.

On traditional projects, analysts often act as intermediaries, communicating customer needs to programmers. On Scrum teams, however, analysts function as facilitators, ensuring that appropriate discussions between team and product owner take place. For instance, an analyst might steer the product owner and team toward talking about a particular user story because it has more risk than another of going astray. At other times, an analyst might convey a top-level understanding of a new feature to the team before bringing the team and product owner together to discuss the details. In all cases, analysts on Scrum teams foster communication rather than distill information through a document.

All of this is not to say we should abandon written requirements documents. Rather, we should use documents where appropriate. Experienced Scrum teams lean heavily on cleanly written code and automated test cases. They then augment these forms of documentation with a written requirements document only to the extent that such a document is helpful or required for regulatory, contractual, or legal purposes. As Tom Poppendieck, coauthor of books on lean software development, once told me, “When documents are mostly to enable handoffs, they are evil. When they capture a record of a conversation that is best not forgotten, they are valuable.”

Agile Teamwork - Transfer Small Tasks Frequently

On Scrum teams, the unit of transfer between disciplines should be smaller than an individual product backlog item. That is, although there will always be some handoffs, the amount of work being transferred from one person to the next should generally be as small as possible.

As an example, suppose a team is developing a new eCommerce application. The team chooses to work on this user story from its product backlog: “As a shopper, I can select how I want items shipped based on the actual costs of shipping to my address so that I can make the best decision.” At the beginning of the sprint, those who are interested in or who will be involved in developing this feature begin to discuss it. Let's suppose this group includes the product owner, a business analyst, a tester, and a programmer. They begin by talking about some of the general requirements implicit in this feature: Which shipping companies (FedEx, DHL, etc.) do we support? Do we want to support overnight delivery? Two-day delivery? Three-day delivery?

As these discussions occur, the individuals involved naturally will begin thinking about how to get started. On a traditional project, each person would be able to start however he wanted (after the work was handed over). On a Scrum team, however, how to get started should be a

collaborative discussion among those who will work on this feature. For this example, let's assume that the programmer for some reason makes the case that it will be easier to start with FedEx. The tester agrees. The analyst states an intention to investigate DHL and learn more about the parameters that affect DHL shipping costs. The analyst's goal is to have that information available by the time the programmer and tester finish with FedEx.

When the programmer knows enough to get started coding, she does so. The product owner, analyst, and tester discuss high-level tests (Will our site ship any odd-sized items like skis?). After that discussion, the tester turns the high-level list of tests into concrete tests (boxes of this size and weight going to that destination). The tester creates test data and automates the tests. Some automation may be possible without any interim deliverables from the programmer, while full automation may require getting an early version from the programmer. While the tester is thinking of the concrete tests, he also should inform the programmer of any test cases that she may not be considering while she's programming.

When the programmer and tester have finished, they add support for calculating FedEx shipping costs into the build, complete with automated tests. Graphically, this can be depicted as shown in figure 1, where four individuals work together on one product backlog item rather than handing it off to each other.

Agile Team Article Figure 1

Image not readable or empty
/uploads/articles/figure1.jpg

Next, the programmer and tester check in with the business analyst, who hopefully has learned more about calculating DHL shipping costs. The process is repeated, and support for DHL shipping calculations is added to the application when the programming and testing are complete. The key element in figure 1 is that the team has learned to work by doing a little of everything all the time. Rather than an analysis phase (done without the programmer and tester) followed by a programming phase followed by a testing phase, a little of each of those activities is happening at all times.
“A symptom of continuing to hand off work in overly large chunks is a tendency for no product backlog items to be finished until the last few days of the sprint.”

A symptom of continuing to hand off work in overly large chunks is a tendency for no product backlog items to be finished until the last few days of the sprint. Testers on teams that work this way often complain that they are given nothing to test until two days before the end of a sprint

and then are expected to test everything that quickly. The best way to expose this problem is to create a chart of the number of product backlog items finished as of each day in the sprint. An example can be seen in figure 2a. As the ScrumMaster on a team, I often just hang this chart in the team area with no fanfare or explanation. Team members soon figure out the problem exposed by a chart like this and, hopefully, start to find ways to finish product backlog items sooner. The result often will be similar to figure 2b, which shows a much smoother flow through the sprint.

Agile Team Article Figure 2

Image not readable or empty
/uploads/articles/figure2.jpg

Agile Teamwork - Create a Manageable Mix

When planning a sprint, teams should pay attention to the sizes of the product backlog items to which they are committing. Some product backlog items are more complex than the FedEx/DHL example given. It's possible that some product backlog items might require a week or more of programming before a programmer can provide a tester with something even initially testable. That's OK. Not everything can be split as small as we might like. The key is to avoid bringing a bunch of items like this into the same sprint. Doing so will shift too much testing work to the end of the sprint.

Instead of planning a sprint with, for example, three very large items that cannot be partially implemented, bring one or two such items into the sprint along with two or three smaller items. Some of the programmers can work on the large items, handing them over to testers whenever possible. The remaining programmers can work on the smaller items, ensuring a somewhat smoother flow of work to testers early in the sprint.

Creating the right sense of teamwork can be challenging. ScrumMasters can help by ensuring that the team embraces the concept of whole-team responsibility and whole-team commitment to deliver working software at the end of each sprint. Though the team might struggle at first to break longheld habits of specialization and handoffs, increasing communication, decreasing the size of handoffs, and mixing the size of backlog items brought into a sprint will help individuals make the shift from sequential development to working as a team.

Posted: April 5, 2010

Tagged: product owner, product backlog, sprinting, scrummaster, teams, teamwork, backlog, programming, testing, customers

---

About the Author

Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of User Stories Applied for Agile Software Development, Agile Estimating and Planning, and Succeeding with Agile as well as the Better User Stories video course. Mike is a founding member of the Agile Alliance and Scrum Alliance and can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.
