Should you write part of your product backlog and user stories before your first sprint? Some people want to take the stance that no work should be done in advance of the sprint. That is clearly untenable. To see why, let’s take that view to its extreme: If we did nothing in advance to understand what we’re building, we’d show up at the planning meeting and say, “Hey, what should we build this sprint? We were working on an eCommerce site yesterday, but I think maybe we should switch to writing a word processor…” The team would literally have nothing written down—no user stories and no product backlog.

This approach leads to what I call “billiard ball sprints.” Without any planning, the team starts a new sprint on the day after the prior one ends, but spends the first two to five days “getting ready” to start the sprint. The start date of the second sprint is knocked forward by the end of the first—like one billiard ball knocking into another.

Having dispensed with the possibility that no work should be done in advance we are left to consider the question of how much work should be done in advance. Many say that as little as possible should be done: the team should do no more than write a few words of a user story on an index card to represent each product backlog item. While it’s true that this may be sufficient in many cases, it would clearly not be adequate in all situations.

My view is that each product backlog item (usually reflected as a user story by teams I train or coach) should be captured just in time and in just-enough detail for the team to go from product backlog item to working, tested feature within a sprint. To see why just-in-time and just-enough are appropriate targets, suppose we decide to write a checkbook program to compete with Intuit’s Quicken product. We create an initial product backlog that includes these two user stories:

- As a user I can see the current version, company website, and copyright in a “help about”
dialog so that I know useful information about the product.

- As a user I can enter a check in my register so that I can track who I pay.

That first story can go from that short description to implemented code quite easily within a sprint. The analyst will have time to figure out exactly what the text should say; the user experience designer (UED) will have time to mock up two or three screens, show them to some users, get feedback, and create the best possible “help about” screen. And the programmers will have time to code it and the testers to test it. In short, it meets the just-enough-detail criteria.

The second user story, “As a user I can enter a check in my register,” is too big to be done in a single sprint. It encompasses the entire main user interface (UI) metaphor—will our system look like a paper check register? How many rows in the register? How will users choose things like check numbers or electronic funds transfer (EFT) transactions, and so on. There is no way the UEDs can figure all that out in a two-week sprint. This user story is not explained in just-enough detail. This is where our second attribute comes in: detail should be added to this user story (product backlog item) just in time. If the team will not work on this user story for six months there is no need to expand on what is already written. On the other hand, if the team hopes to start it in a few sprints, it is likely time for the UEDs to figure out the answers to the high-level questions listed above.

To do this, they’ll need to allow time to “figure out the answers” in the upcoming sprints. Let’s say that during the first sprint the team chooses to work on the “help about” story and likely a few others (not shown in our sample product backlog above). They also agree to spend some time figuring out what the main UI metaphor for the system will be. During this time the UED may design a couple of screens and show them to a few dozen or hundred users, get feedback, and do it again. This may happen a few times. When they finish they may have turned that one user story (as a user I can enter a check in my register) into a dozen smaller stories, perhaps specifying what fields are needed and other details such as:

- As a user, when I enter a new check it defaults to the last check number + 1
- As a user, I can enter a memo for each check, etc.
- As a user, I can double-click on a check in a list and see the current item look like a paper check.

In some cases splitting a large story into smaller ones is sufficient to show the new items at the
right (just-enough) level of detail. When it’s not, we can augment the user story with additional information. I like to write user stories on index cards. That was sufficient for the “help about” story. It’s not for the initial large user story encompassing our system’s main UI metaphor. If splitting a story does not add just-enough detail, the UED and others involved may want to staple a UI design spec to the index card. (If you are using a software system for managing your product backlog, your virtual staple may be a hyperlink to a wiki page.)

The UI design spec that is stapled to the user story card will not yet be perfect; rather it will be close enough that remaining details can be figured out during the sprint. For example, the UED may not yet have decided if each check should take two rows or three on the screen and wants to do a bit of user testing early in the sprint while the team codes that story. He’ll get them an answer early enough that they can do it either way during the sprint.

The goal of the UED (and anyone else involved at the time) is to add detail to the story in a just-in-time manner. There is usually no value to splitting a large user story up now or stapling a document to it if we won’t work on that product backlog item for quite awhile. While working to add detail in a just-in-time manner, those doing so should strive to add just-enough detail that no individual item will take more than one sprint to complete. Just in time and just enough become the targets for establishing a smooth flow from product backlog to sprint backlog.
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.