Last week I wrote about “Sprint Zero” and made the point that on the rare occasion when this might be a good idea, Scrum teams would be better off thinking of that time as a “project before the project.” Projects do not spring to life fully formed—that is, staffed and ready to be worked on. The vast majority of projects can, though, be instantly started and things like technical environments and an initial product backlog figured out during the first sprint—a good, old fashioned sprint number one.

But some projects should be preceded by an effort to decide what the project should entail and even whether there should be a project at all. This is the type of analysis work many teams put into a special Sprint Zero. While one sprint is often enough for this work, it's not hard to imagine a project large enough that it could benefit from a couple of sprints (albeit, usually short, one-week sprints). Although I think this is rare, I'd like to offer further advice on conducting such a “project-before-the-project.”

A project-before-the-project can be more simply thought of as an analysis project. It may answer questions such as:

- What features would be delivered?
- About how long would it take?
- How much investment will be necessary?
- Should the project be undertaken?

On a project like this, there is no “potentially shippable product increment” in the normal sense of working software delivered at the end of a sprint. So, what then does a project such as this deliver?

To answer questions like this about what to do when applying Scrum outside its home territory of product development, I find it helpful to consider the reasons why particular Scrum rules are in place. The requirement to deliver a potentially shippable product increment exists to help Scrum teams avoid analysis paralysis and to feel a sense a urgency to produce something
within the sprint. Just the right amount of urgency can generate greater creativity. The rule also exists to help Scrum teams stay honest—it prevents them from claiming progress when none has been truly made.

To see how these rules can apply on an analysis project, suppose a team is considering developing a new system for hospitals and is running one week sprints to answers questions like those above. At the end of a sprint, the team needs to be truly done with something. They can't be truly done developing a feature since they are only analyzing the system. But one way the team could be done with something is for them to have interviewed all the pharmacists who are stakeholders on the project. The team now knows everything pharmacists will need including new functionality relating to prescribing medications, looking up patient records, and so on. The team will not need to talk to pharmacists about their basic requirements again. They can be said to be done interviewing pharmacists.

Alternatively, the team could instead have talked to all stakeholders regarding their needs for the prescriptions part of the system. In this case, the team spends the sprint interviewing not just pharmacists but also some medical doctors, nurses, and so on. They don't talk to these stakeholders about anything but the prescriptions part of the system. They'll come back and talk to stakeholders about other functionality in successive interviews. For now, though, this team can be said to be done analyzing needs of all stakeholders involved with prescriptions.

These examples illustrate how to apply the principles that underlie Scrum rules that can be a bit software- or product-development specific. While they allow a team to apply Scrum outside its home territory, I want to repeat my caution that I'd prefer not to undertake such an effort at all. Most projects can just start, with these issues resolved as the project begins.
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 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.