THE PROFESSIONAL SCRUM TEAM

GROWING AND EMPOWERING CROSS-FUNCTIONALITY AND RESILIENCY IN A COMPLEX WORLD

PETER GÖTZ

with contributions by

UWE M. SCHIRMER *and* KURT BITTNER

Foreword by DAVE WEST

The Professional Scrum Series by Scrum.org

The Professional Scrum Team

with the Sprint plan but treats the plan as a forecast. If something unforeseen happens, the Scrum Team may need to change the scope of the Sprint. It should not change the Sprint Goal, though; that should be fixed.

Making it clear that the forecast from Sprint Planning is not a promise is very important. Stakeholders, especially in organizations that see missed plans as failure, need to understand this in order to be able to find value in Scrum. This inherent unpredictability in Scrum comes with some benefits that can be reaped only if you accept it. As a Development Team's experience with Scrum grows, its forecasts usually become more reliable. It is also better able to account for the "normal" unknowns that happen all the time, such as phone calls, meetings, and urgent and important ad hoc tasks.

When we see organizations use the word *commitment* in the old way, we gently try to steer them to the term *forecast* instead. Changing the word also changes their perception of a Sprint, as it improves transparency because teams don't need to use opaque buffering to accommodate unpredictability. Transparency enables organizations to work to remove impediments that lead to unexpected but avoidable work. Transparency also helps to improve understanding, collaboration, and trust among different teams and departments in an organization.

This doesn't mean that a Development Team can say, "We can't say if we can make it during this Sprint, as it takes as long as it takes." That would be the other extreme. A Scrum Team should refine the top of its Product Backlog into small enough items so that it can create a realistic forecast. This forecast can turn out to be wrong if something unforeseen happens, but otherwise it should be accurate.

A little after the Sprint Planning, the Development Team members are back in their room, starting to work on their Sprint Backlog.

"All those meetings! I am really glad that we are back to real work," one of the developers says. "Yesterday afternoon was the Sprint Review, then the Retrospective, and today we had two hours of Sprint Planning. That is really too much overhead. We hardly get any work done."

Too Many Meetings?

The Scrum Guide describes maximum time-boxes totaling twenty hours of events for a four-week Sprint plus up to 10 percent of the Development Team's capacity for Product Backlog refinement. That's four and a half working days. For the whole team. Isn't that a bit too much?

Many organizations complain about the high number of meetings in Scrum. They claim that the team members hardly have the chance to get anything meaningful done because they sit together and talk the whole time. And that is on top of all the other organizational meetings in which employees must participate.

This second complaint is the problematic one: Events in Scrum make many other meetings that organizations use to align their employees and teams obsolete. Separate status meetings, in which employees tell their superiors and each other what they are currently working on, should not be necessary to learn what is going on. When appropriate transparency is created, even some management meetings are unnecessary. In addition, many ad hoc meetings for clarification or alignment can be made obsolete by the Daily Scrum or regular Product Backlog refinement sessions.

Scrum creates transparency on several levels. Team members use the Daily Scrum to collaborate with each other and update their Sprint Backlog accordingly. Since the Sprint Backlog is transparent to anyone, management interested in short-term progress can get the status from there. The Scrum Team reviews the Increment with stakeholders at the end of each Sprint and asks for feedback, which takes the place of status/progress review meetings. Since Scrum Teams are self-organized, their management usually doesn't need very detailed status reports. Stakeholders can usually go visit the teams and see their status presented transparently on information radiators such as a task board or cumulative flow diagram.

Let's get back to the first complaint—all those meetings, and no real work gets done.

First, the maximum time-boxes for events are just that: *maximum* time-boxes. Teams don't need to use those time-boxes; they can optimize their work and finish meeting quicker. Second, if they can't finish quicker because there is so much to clarify, the events' durations shouldn't be a problem, as the time is obviously needed. Communication and collaboration prevent waste and defects.

Most teams we work with spent more time in meetings before they started with Scrum. Think about all those little alignments and clarifications, often done ad hoc, that you need in order to do the right things. Scrum defines fixed opportunities in every Sprint to do them and calls them *events*. Those events should replace and make obsolete those smaller meetings spread over our working days.

In the beginning of their journey, the Scrum Team members did the Sprint events on their own. For example, for Sprint Review, the Development Team showed the Product Owner what they had achieved and got feedback from her.

"Now we come to a new feature that we implemented. Patients now can make and cancel appointments via our portal. They don't have to call the doctor's office if they don't want to."

The Development Team walks the Product Owner through the Sprint Backlog and tells her what it was able to implement and how it is being used.

"Okay, that looks nice. Do patients always have to enter the date manually? What if they enter a wrong date?" the Product Owner wants to know.

"The date is being validated, and an error message is shown to the patient. The patient has to enter a correct date and select a time slot that is still available in order to store the appointment," a developer explains.

"Okay, got it. But wouldn't it be easier for patients to select a date in a visual calendar and then select a free slot on that day?"

"Yes, sure, we can do that. It would take a bit of effort, but it is possible. Should we add this to the Product Backlog?"

"I will first check with our marketing department to see if our customers really need this. We can discuss that in our refinement next week."

No Stakeholder in Sprint Review

This is a typical situation in many organizations: The Scrum Team sits together for the Sprint Review and discusses the product Increment and what could be the next valuable improvements. The way our team executed its Sprint Review shows some of the positive effects of Scrum. The team's work results are inspected frequently, and feedback is being collected and used to further improve the product. But it is not as effective as it could be. There are a few points that can be improved.

First, the Product Owner isn't sure if the improvement she discussed with the team is valuable enough to justify the extra effort. She wants to discuss that with her stakeholders and will come back to the Development Team. It would be better to have the stakeholders in the Sprint Review to give feedback directly.

Doing so would provide the Development Team with firsthand feedback, positive as well as negative. When the people doing the work are connected to the people who use their results—their customers and users—empathy is improved in both directions. Development Teams empathize more with their customers and users and better understand their needs and wants. Stakeholders also empathize more with the Development Team. They learn about the team's constraints and understand the reason behind technical decisions.

Second, the Product Owner can't provide important feedback during the Sprint Review but has to check with her stakeholders after the review. This delayed feedback means a loss of days, sometimes weeks. Once the Development Team receives the feedback, it will need to adapt to it. If stakeholders are part of the Sprint Review, they can share their feedback immediately, and the team can take action as soon as the next Sprint.

The more stakeholders actively participate in the Sprint Review, the better the feedback will be. Therefore, instead of showing stakeholders the work results in a presentation or demonstration, let them use the product. This will greatly increase the quality of the feedback received during and after the Sprint Review.

When stakeholders are present in a Sprint Review, the Product Owner can share the current state in regard to midterm planning. Many Scrum Teams work from Sprint to Sprint and don't have enough focus on and visibility of the midterm planning. With stakeholders in the Sprint Review, the Product Owner can show what bigger goals are currently being worked on and what the status after this Sprint is. She can also give a rough forecast on what the goals of the next Sprint or Sprints will be. This creates transparency for stakeholders, who are usually interested in the big picture of product development and not so focused on the specifics of each Sprint.

Having stakeholders participate in Sprint Reviews has another indirect benefit. When the Product Owner is the target audience of the Sprint Review, there is not much need for in-Sprint collaboration. The Product Owner often sees the product Increment for the first time during Sprint Review. This is a big risk. A Development Team could work for a whole Sprint, up to a calendar month, without the Product Owner's feedback on its work results. A Sprint Review that targets the stakeholders usually changes this situation dramatically. The Product Owner becomes the host of the Sprint Review and needs to be prepared. Therefore, she collaborates with the Development Team during the Sprint and gives feedback on the way things are implemented. This feedback can already be included in the product Increment, which means a better result can be shown during the Sprint Review.

Having stakeholders in the Sprint Review has many advantages. Sometimes organizations claim that their stakeholders are too busy to join the Sprint Review. They want to be updated personally by the Product Owner afterwards. The downsides of this approach are serious enough, in our opinion, to justify stakeholders taking this time out of their busy calendars. We think that it is more efficient to make sure this time together happens instead of working on the wrong things or slowing down the whole Scrum Team unnecessarily.

"Okay, I understand why this might make sense. Let's see what the Scrum Guide says about it."

Scrum Is Not a Religion

The comment in this short scenario is repeated time and again in endless teams worldwide. Don't get us wrong—there is nothing wrong with checking the Scrum Guide if you want to read up on a specific rule and make sure you get it right. The problem starts when individuals, teams, and organizations start to see Scrum as some kind of religion and the Scrum Guide as its scripture.

Scrum is a framework that describes the fundamentals of how to solve complex adaptive problems. It can't and doesn't give guidance for every situation imaginable in an organization. On the contrary, it relies on self-organization and a simple set of values and principles.

Why is it counterproductive to try to follow Scrum as strictly as possible? The answer is quite simple: If you focus on following Scrum dogmatically, you lose the ability to act in a context-sensitive manner. All organizations have their own specific context. We have never seen a company or team that was able to do "perfect" Scrum, whatever that might be. You have to find your own specific implementation of Scrum to suit your challenges, context, and people.

Scrum is based on the principles of empiricism. It inspects the status quo frequently and adapts it for continuous improvement. Relying on the Scrum Guide to do the right thing and be done with it goes against these very principles.

If you find yourself discussing a question purely from the viewpoint of "doing Scrum right," please stop. Then think about what would make sense in your situation. Check whether your solution adheres to the values of Scrum and the principles of empiricism. Finally, you could check whether your solution already has a place in the framework; its roles, events, and artifacts; or the rules that bind them together. If one of those checks is negative, try to adapt your solution so that it adheres to the values and principles and, maybe, the Scrum Guide. If this is not possible but your solution still makes sense, implement it and let empiricism show whether this solution helps you or has some unwanted side effect. If it has, inspect and adapt.