Toward a Catalog of Scrum Smells

By Mike Cohn

Even with Scrum things can go wrong. As diligent ScrumMasters it is our responsibility to constantly keep an eye on our projects and look for small problems before they can become big problems. In his book *Refactoring*, Martin Fowler introduced the term *smell* to refer to something that may not be right. Just because something smells doesn't mean there's a problem; it does mean, though, that further investigation is warranted.

This article is a first step toward collecting a catalog of Scrum smells; that is, signs that something may be amiss on a Scrum project.

Loss of Rhythm

Symptom: Sprints are not always the same length.

Discussion: On a well-executed Scrum project the team establishes a natural rhythm. Each sprint is the same length. Each sprint starts with a Sprint Planning Meeting where requirements are discussed and the sprint planned. Each sprint concludes with a Sprint Review where the team demonstrates a potentially shippable product increment. In-between the project rhythm is supported by daily scrums.

If sprints are sometimes two weeks and sometimes four weeks then this natural rhythm is never established. Sprints then begin to feel like arbitrary units of time with endpoints selected more by outside forces (perhaps political or competitive) rather than designed to enhance the overall productivity of the team. When sprint duration is allowed to vary teams have a harder time selecting the right amount of work for the sprint backlog, which may result is less commitment to completing all of the items in the sprint.

Talking Chickens

Symptom: Chickens attending the daily Scrum are allowed to ask questions or make observations.

Discussion: During the daily Scrum the only participants allowed to speak are the pigs (those committed to the project); chickens (those involved but not committed) may attend and observe but are not allowed to speak. Allowing chickens to talk can be a slippery slope. If a chicken is allowed to make a comment one time (when the comment is useful), how do we later prevent a chicken from commenting (when the comment may not be useful)?

A few months ago I took my young daughters to a local fair. The line for one ride led up to a short flight of stairs at the base of the ride. No one was allowed on those stairs, but many of the young kids wanted to sit on the stairs while waiting. The ride operator, though, held firm to her rule of no one on the stairs. She told them, "If I let you sit on the first step soon you'll be on the second step and then the third step." Clearly sitting on the first step would not have endangered any riders but the stairs were an obvious delineation and the ride operator used this as her simple rule. Not allowing chickens to talk during the daily meeting is one of Scrum's simple rules. Of course one comment from a chicken may not hurt—but it will lead to others and then there will be no easy place to draw the line.

Missing Pigs

Symptom: Not all pigs attend the daily Scrum meeting.

Discussion: When I started working there was no such thing as "flex time." Every company had a starting time and everyone was expected to be at work by that time. Flex time, combined with the nocturnal habits of many software developers, makes it common to have some members of a team arrive at 11:00 am, or even later. I worked with one team who had nicknamed one programmer "Captain Midnight" because he quite literally came to work around midnight each day.

All of this can make it very difficult for a team to get together for a daily Scrum every day. However, having a daily meeting and having it at the same time and in the same place each day help a project establish and maintain its rhythm. If too many pigs are missing daily Scrums too often then perhaps the meeting is taking too long or is deviating from the three standard questions. When run well the daily Scrum should almost never exceed fifteen minutes and will almost always be of value to each pig. Yes, sometimes we'll have specific deadlines that are so urgent and so imminent that we may be tempted to skip a daily Scrum. And that's not all that bad. It's when the situation becomes chronic or when too many pigs miss meetings that the situation begins to smell.

Persistent Signatures

Symptom: The wild fluctuations shown on a team's initial sprint burndown charts continue to be seen in much later sprints.

Discussion: In *Agile Software Development with Scrum*, Ken Schwaber and Mike Beedle write about each team having a unique signature—some teams bring in too much work at the start of a sprint, others bring in too little, and so on. However, it is important that a team learn from its past sprints. For example, suppose a team of five started the last sprint with an estimated 600 hours of work (the first point on their sprint burndown chart). Midway through that sprint they realized they were attempting too much and worked with the Product Owner to remove 100 hours of work. Now, in planning the next sprint they again want to bring in 600 hours of work. They need to look at that and answer for themselves why they think they can achieve 600 hours of estimated work this sprint when it was too much last sprint.

Over time, as teams learn about themselves and their project their sprint burndown charts should lose the wild variations that may have existed in initial sprints and begin to somewhat approximate the idealized straight line. If this trend is not apparent for a team then they are missing opportunities to learn from their own past performance.

ScrumMaster Assigns Work

Symptom: Work is assigned by the ScrumMaster rather than signed up for by developers.

Discussion: Self-organization is one of the underlying principles of Scrum. When a ScrumMaster assigns work it undermines the responsibility developers assume when they are allowed to self-organize around the achievement of a goal. The real danger here is that even an occasional assignment from a ScrumMaster can do a lot of damage. Teams need to feel completely in control of their own work.

The Daily Scrum is For the ScrumMaster

Symptom: The Daily Scrum feels like it is a status update from the team members to the ScrumMaster.

Discussion: Sometimes the daily Scrum begins to feel as though it exist solely for the ScrumMaster. You'll see the ScrumMaster furiously scrawling notes about who committed to what work and why some other task wasn't completed. Daily meetings like this take on the feeling of the status meetings of other development processes.

There are two main purposes of the daily Scrum and neither is to provide status information to the ScrumMaster. The first purpose of the daily Scrum is to provide a coordination mechanism for everyone on the project. Once a day it can be very useful for everyone to hear where everyone else is. Ideally there are many hallway or random conversations throughout the day but those don't usually include the full team. During the daily Scrum everyone gets a sense of where everyone else is.

The second purpose of the daily Scrum is for each team member to make commitments in front of his or her peers. When a developer answers the "What are you going to do today?" question she is making a commitment to her peers, not to the ScrumMaster. At the next meeting if that commitment has not been fulfilled it is not the ScrumMaster's role to say, "Tsk, tsk, Lucy, yesterday you told us you'd be done." If Lucy said she'd be done and isn't she probably feels bad enough. She knows what she told her peers and she knows if she didn't finish because of random bad luck, a lack of effort, misunderstanding the size of complexity of the task, or any of a myriad other reasons.

Specialized Job Roles

Symptom: A project team has highly specialized job roles or descriptions such as Architect, Designer, DBA, or Tester.

Discussion: Scrum teams need to have a “we’re all in this together attitude.” This can sometimes be undermined if a team has specialized job descriptions or roles. For example, if a team includes a dedicated “tester” then this may give the rest of the team an opportunity to shirk the responsibility to test.

With the highly complex technologies of the world today it is too simplistic to think that everyone can be a DBA and everyone can write server-side J2EE or .Net code. A successful Scrum team does not need to be comprised entirely of generalists. However, for a team of specialists to be successful each specialist must accept general responsibility for the system as a whole. I may not know how to solve our project’s most intricate Oracle problems but I’m going to do whatever I can to help, which may simply include taking on some of our database specialist’s other responsibilities to free her to solve the complex problems.