```
Errno::ENOENT: No such file or directory @ rb_sysopen - /Users/rpowell/Sites/ee.
```

Differences Between Scrum and Extreme Programming

by Mike Cohn • 106 Comments

This article has an audio version: [Download Audio Version](#)

Image not found or type unknown

Differences Between Scrum and Extreme Programming

Scrum and Extreme Programming (XP) are definitely very aligned. In fact, if you walked in on a team doing one of these processes you might have hard time quickly deciding whether you had walked in on a Scrum team or an XP team. The differences are often quite subtle, but they are important. I think there are four main differences between Scrum and XP:

1. Scrum teams typically work in iterations (called sprints) that are from two weeks to one month long. XP teams typically work in iterations that are one or two weeks long.
2. Scrum teams do not allow changes into their sprints. Once the sprint planning meeting is completed and a commitment made to delivering a set of product backlog items, that set of items remains unchanged through the end of the sprint. XP teams are much more amenable to change within their iterations. As long as the team hasn't started work on a particular feature, a new feature of equivalent size can be swapped into the XP team's iteration in exchange for the unstarted feature.
3. Extreme Programming teams work in a strict priority order. Features to be developed are prioritized by the customer (Scrum's Product Owner) and the team is required to work on them in that order. By contrast, the Scrum product owner prioritizes the product backlog but the team determines the sequence in which they will develop the backlog items. I've never seen a Scrum team not choose to work on the highest-priority item. And a Scrum team will very likely choose to work on the second most important. However, at some point one of the high priority items may not be a good fit for the sprint being planned—maybe a key person who should work on it will be swamped by work on higher priority items. Or maybe it makes sense to work on a slightly lower priority item (let's say #10 on the product

backlog instead of #6) because the team will be working in the code where #10 would be implemented.

4. Scrum doesn’t prescribe any engineering practices; XP does. I love the XP engineering practices, particularly things like test-driven development, the focus on automated testing, pair programming, simple design, refactoring, and so on. However, I think it’s a mistake to say to the team “you're self-organizing, we trust you, but you **must** do these specific engineering practices….” This sends a mixed message to the team that causes confusion. I love the XP practices but don’t like mandating them. I want teams to discover the value on their own.

These are small and often subtle differences between Scrum and XP. However, they can have a profound impact on the team. My typical advice to teams is “start with Scrum and then invent your own version of XP.” The XP practices are wonderful but they work best and teams commit to them the most stridently if they discover them themselves rather than having them mandated. I help teams do this in my coaching by asking questions like, “Would this bug have happened if we'd been doing test-driven development?” and “Would we have made that mistake if we were pairing?”• I find true XP to be a small target off in the distance. If a team can aim at that and hit the bull’s eye, wonderful. If not, however, they are likely hacking (e.g., refactoring without any automated testing or TDD). Scrum is a big bull’s eye that on its own brings big improvements simply through the additional focus and the timeboxed iterations. That’s a good starting point for then adding the XP practices.

Posted: April 6, 2007

Tagged: audio, musings