

## Should Scrum Teams Re-estimate Stories that Can't Be Completed During the Sprint?

Posted by Jim Schiel on July 10, 2008 at 5:22 a.m.

Jim Schiel is a CST with a strong background in enterprise level Scrum installations. Prior to joining Danube, Schiel worked at Siemens for 22 years, initially as a developer, then as a manager for 16 years, and eventually playing an instrumental role in creating one of the largest Scrum installations in the world. As a business process engineer at Siemens, he helped identify, document, and implement best practices for enterprise Scrum.

If a user story isn't completed during a Sprint, the cleanest way to deal with it is to leave the story point value as is and return it to the Product Backlog. When Sprint Planning occurs, if you're using a commitment-driven model as we often do in CSM training, you can make the team aware of how much of the story is left to do when they task it out (frequently, you don't even have to task it out, because the tasks came with the story from the previous Sprint Backlog).

This seems a little strange, I know, but it helps ensure that the release burndown isn't reflecting a reduction based on partially completed stories that aren't really done and can't be delivered to a customer. This doesn't seem like much when we're talking about one story, but guess what happens to the release burndown when it shows you the result of 40 stories that are ALMOST finished.

Some complain that this artificially reduces the team's velocity during one month while artificially inflating it during the next. So what? Over the course of several months, a team's velocity is clear and, over the short term, the team understands the impact of the incomplete story and can discuss true velocity amongst themselves should they need to.

Others suggest re-estimating the story based on the remaining work. However, what this often misses is the extra time that is usually required to remember where the team was with a story before they can actually get the work done (though this may be minimal if there's only a task or two left to do). It also completely misses what happens if the story gets scoped out of the release ("since it isn't done, why not scope it out if I need to," says the PO) and effort has to be spent removing the code already added to the code base.

Another option is to slice the story into two pieces – the piece that's done as opposed to the piece that isn't done. I don't advise doing this unless you can really slice the story on sub-functional boundaries (with one or two tasks remaining, you usually can't really do that without creating imaginary sub-functional boundaries).

The last thing to remember is that there's no penalty for not getting everything done in a Sprint. There's no reason to cut corners or go to extraordinary measures to finish that final task. Work does not fit

**danube**

Danube Technologies, Inc.  
520 SW 6th Ave, Suite 940  
Portland, OR 97204 USA  
1.888.5.danube  
T: +1.503.248.0800  
F: +1.503.248.0801

perfectly into a Sprint. The whole point of a Sprint is to “chunk-up” the work into a simple schedule and, after the end of each period, to see completed functionality and make the right decisions on how to proceed based on that demonstration.

**danube**

Danube Technologies, Inc.  
520 SW 6th Ave, Suite 940  
Portland, OR 97204 USA  
1.888.5.danube  
T: +1.503.248.0800  
F: +1.503.248.0801