Agile Record

The Magazine for Agile Developers and Agile Testers

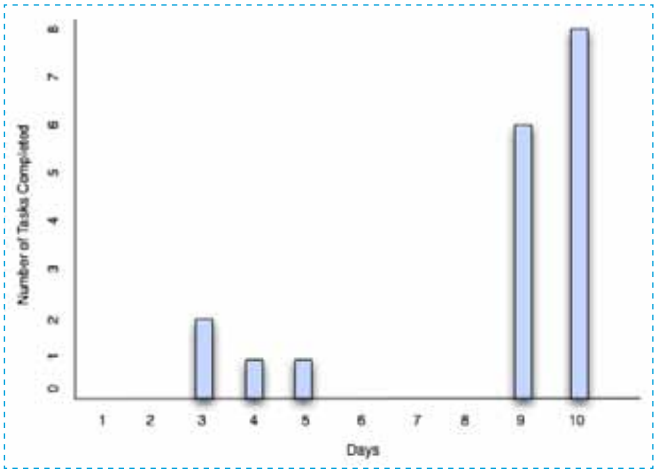
www.agilerecord.com free digital version made in Germany ISSN 2191-1320 April 2011 issue 6

Smoothing Out Lumpy Sprints

by Catherine Powell

Agile teams generally work in sprints or iterations. Using shorter time periods involving a sprint instead of a release cycle before confirming that the software works and is stable helps make development more efficient and easier. It turns development into many small tasks, instead of several large tasks and a long (and messy!) integration cycle. Within an iteration, though, many teams do a lot of things partially, and then finish the sprint with a rush to check in. These teams still have tasks and then a messy integration cycle; it's just shorter! It's a smaller mess, but it's still a lumpy mess of a sprint.

A Lumpy Sprint



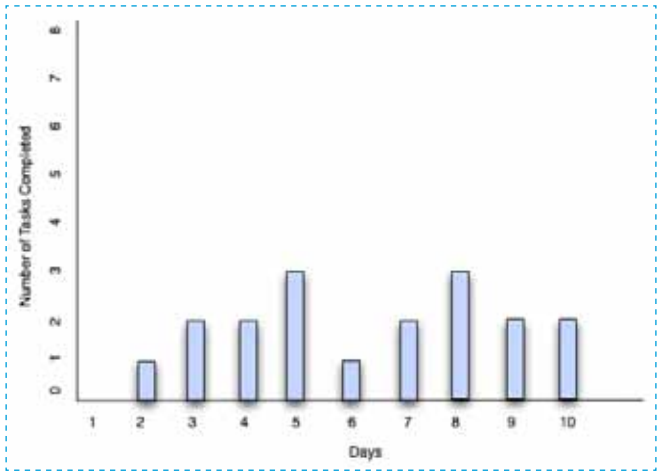
This is a fairly typical lumpy sprint. The sprint starts and each member of the team picks up a task or two. Everything's going well, but then something takes a little longer than expected, or a bug comes in from the field and needs immediate attention. The task in progress stays in progress. Another developer decides he's just "got a few little tweaks left" on his first task, so he puts it aside and starts a second task "just to make sure there aren't any surprises." Repeat this across a team and the "mostly done" tasks proliferate. Getting from 80% done to 100% done all happens late in the sprint, and there's a rash of check-ins, finishing and testing at the end of the sprint.

The Problem With Lumps

As long as all tasks are getting done within the sprint, does it really matter whether they all get finished close to the end? In a word, yes. A flurry of check-ins at the end of a sprint increases the likelihood of integration problems and forces the final testing to be rushed. In addition, starting many tasks without finishing them makes it more difficult to drop items from the sprint if that's needed. It's a lot more difficult to remove something that's partly done from a code base than to remove something that hasn't been started yet.

A Smooth Sprint

This is a smoother sprint. Each team member working on a task finishes that task before starting the next one. If the task is interrupted (for example, by an urgent bug at a customer that needs to be diagnosed), another developer finishes the task instead of starting something else. Tasks are finished throughout the sprint and integration occurs in smaller chunks as the sprint progresses rather than all happening at the end. If something needs to be



removed from the sprint, there are more choices: anything that hasn't started can be removed.

Tips for a Smoother Sprint

Making a sprint smooth is about making sure tasks are getting finished throughout the sprint. Spreading out the points where tasks are completed and checked in makes the sprint smoother, easier to test, and more stable. To make sprints smoother:

- **Finish something all the way**. When you finish a task, do all of it, including the tests, documentation and fixing any issues or integration problems that arise. Don't give into the temptation to move on to something else while there are still "tweaks" since those tweaks may be larger than they seem.
- **Work together on a task**. Grab someone else on the team and start a task together. That way, if one of you gets pulled off, the other can still finish the task. It doesn't work for really tiny tasks, but it makes sense for any task more than a few hours long. Working together also helps to make sure that there aren't too many tasks going on simultaneously when the team is fairly large.
- **Don't start more than one task at a time**. Any person can really only work on one thing at a time (multi-tasking is a myth!), so don't try. Pick up a task and work on it until it's done or you're blocked completely. Then don't start a new task; help resolve the blockage instead. Finish a task, then start a second one.

Conclusion

The two most stable states for a sprint task are: (1) not started; and (2) completed. A task that hasn't started yet has had no effect on the system at all. It can be removed with no consequences for the rest of the system (if it doesn't exist, removing it won't destabilize the system). A task that is completed can be shipped at any point and won't change; it provides a stable base for other work. To make a sprint smoother, keep as many tasks as possible in the stable states; don't start them until you can complete them as quickly as possible. A smooth sprint is a stable sprint, and that's a lot easier for everyone involved to work with.

> About the author

Catherine Powell

has been testing and managing for about ten years. She is a manager, a tester, an author and a formal mentor to testers and test managers. Catherine has worked with a broad range of software, including an enterprise storage system, a web-based healthcare system, data synchronization applications on mobile devices, and webapps of various flavors.

With a focus on the human side of software development, Catherine builds strong teams spanning testers, developers, support, product management, and all the people involved in turning an idea into reality. She emphasizes the generation of information and pragmatic decision making using a myriad of approaches. With thoughtful techniques that access the strengths of the human team members combined with the needs of the system and its users, Catherine guides both developers and testers to be a valuable part of the decision making required to create rock-solid software.

Catherine focuses primarily on the realities of shipping software in small and mid-size companies. Specifically, she highlights and works to explicate the „on-the-ground" pragmatism necessary for an engineering team to work effectively with both software and humans from product definition through release, and in the field.