Why I Prefer Capacity-Driven Sprint Planning

by Mike Cohn • 36 Comments

Why I Prefer Capacity-Driven Sprint Planning

In the prior two blog posts, I've described two alternative approaches to sprint planning:

- Velocity-Driven Sprint Planning
- Capacity-Driven Sprint Planning

In this post, I want to address why I prefer capacity-driven sprint planning. First, though, here is a very brief refresher on each of the approaches.

Brief Summary of Sprint-Planning Approaches

In velocity-driven sprint planning, a team selects a set of product backlog items whose high-level estimates (usually in story points but possibly in ideal days) equals their average velocity.

In capacity-driven sprint planning, a team selects one product backlog item at a time by roughly identifying and estimating the tasks that will be involved, and stopping when they feel the sprint is full.

Velocity is Variable

To begin to see why I prefer a capacity-driven approach to sprint planning, consider the graph below which shows the velocities of a team over the course of nine sprints. The first thing you should notice is that velocity is not stable. It bounces around from sprint to sprint.

Image not found or type unknown

The first big drawback to velocity-driven planning is that velocity is too variable to be reliable for short-term (i.e., sprint) planning.

My experience is that velocity bounces around in about a plus or minus 20% range. This means that a team whose true velocity is 20 could easily experience a velocity of 16 this sprint and 24

next sprint, and have that just be random variation.

When a team that averages 20 sometimes completes 24 units of work, but sometimes completes only 16, we might be tempted to say they are accomplishing more or less work in those sprints.

And that is undoubtedly part of it for many teams. However, some part of the variation is attributable to the imprecision of the units used to estimate the product backlog items.

For example, most teams who estimate in story points use a subset of possible estimates such as the Fibonacci sequence of 1, 2, 3, 5, 8, 13 or a simple doubling (1, 2, 4, 8, 16). When a team using the Fibonacci sequence takes credit for finishing a 5-point story, they may have really only finished a 4-point story. This tends to lead to a slight overstating of velocity.

In the long-term this won't be a problem as the law of big numbers can kick in and things will average out. In the short term, though, it can create problems.

Anchoring

A second problem with velocity-driven sprint planning is due to the effect of anchoring. Anchoring is the tendency for an estimate to be unduly influenced by earlier information.

A good example of anchoring is the Christmas season. Suppose you go into a store and see a jacket you like. There's a sign saying $400 but that price is crossed out and a new price of $200 is shown. Your brain instantly thinks, "Awesome! A $400 jacket for only $200!" And, of course, this is why the store shows you the original price.

Who cares what that jacket used to sell for? It's $200 today. That should be all that matters in your buy-or-not decision. However, once that $400 price is in your brain, it's hard to ignore it. It anchors you into thinking you're getting a good deal when the jacket is $200.

I won't go into the details here, but the best paper I've read on anchoring is from Magne Jørgensen and Stein Grimstad and is called "The Impact of Irrelevant and Misleading Information on Software Development Effort Estimates.")

Anchoring and Velocity-Driven Planning

But what does anchoring have to do with sprint planning? Consider a team in a velocity-driven sprint planning meeting.

They've selected a set of stories equal to their average velocity. They then ask themselves if that set of stories feels like the right amount of work. (As described, in the post on velocity-driven sprint planning they may also identify tasks and estimate those tasks as an aid in answering that.)

A team doing velocity-driven sprint planning is predisposed to say, "Yes, we can do this," even if they can't. They are anchored by knowing that the selected amount of work is the same as their average velocity.

It's like me showing you that jacket with the $400 sign next to it and asking, "How much do you think this jacket sells for?" Even if you don't say exactly $400, that $400 is in your head and will anchor you.

So, because of anchoring, a team with an average velocity of 20 is inclined to say the sprint is appropriately filled with 20 points – even if that work is really more like 16 or 24 units of work.

This will lead to teams sometimes doing less than they could have. It could also lead to teams sometimes doing more. But in those cases, my experience is that teams will be more likely to drop a product backlog item or two. Experience definitely tells me that teams are more likely to drop than to add.

Velocity Is Great for Longer-Term Planning

By this point, you may be thinking I'm pretty opposed to velocity-driven planning. That's only half

true. Although I'm not a fan of using velocity to plan a single sprint, I am quite likely the world's biggest fan of using velocity to plan for the longer term. I've certainly written more about it than anyone I know.

The problem with velocity-driven sprint planning is that velocity is simply too variable to be useful in the short term. To illustrate why, suppose you get a job working at a car wash. On your first morning, your boss announces that you have a quota: You need to wash four cars per hour.

At the end of the first hour, though, you've only washed three cars. Should you be fired? Of course not. It was only one hour and perhaps you washed three large cars. Or perhaps it was overcast and only three drivers brought cars in to be washed.

What about at the end of your first day, an 8-hour shift? By then you should have washed 32 cars. But you've only washed 30. Should you be fired now? Again, almost certainly not.

What about the end of your first month? Figuring 20 days and 32 cars per day means you should have washed 640 cars. Suppose, though, you only washed 600. (That's the same percentage as washing 30 instead of 32 in a day.) Should your boss fire you now? Perhaps still not, but it's starting to be clear that you are not making quota.

What if you're off by the same percentage at the end of the year? If that quota was well chosen—and all other employees are meeting it—your boss at some point should consider letting you go (or dealing with your below expected productivity in some way, but this post isn't about good ways of dealing with productivity issues).

That quota is useful in the same way velocity is useful: over the long term. Think of how poorly run we'd consider that car wash if an employee was fired in the first hour of missing quota. The longest tenured employee would have been on the job for three days. So while that quota may be useful when measured over a month, it is not useful when measured hourly.

Velocity is useful in the long term, not the short term. A team with 30 product backlog items to deliver can sum the estimates (likely in story points) on those items and forecast when they'll be done. A team with three product backlog items to deliver would be better off doing good ol' task decomposition on those three items rather than relying on velocity.

So, Now What?

If you are already doing velocity-driven sprint planning and it’s working for you, don’t switch. However, if your team is new to Scrum or if your team is experiencing some of the issues I’ve described here, then I recommend using capacity-driven sprint planning.

Please let me know what you think in the comments below.

Posted: November 4, 2014

Tagged: product backlog, velocity, sprint planning, planning, commitment, capacity driven

---