Know Exactly What Velocity Means to Your Scrum Team

by Mike Cohn •

33 Comments

The following was originally published in Mike Cohn's monthly newsletter. If you like what you're reading, sign up to have this content delivered to your inbox weeks before it's posted on the blog, here.

When talking about it informally, I define velocity as simply a measure of how fast a team is going. And for most purposes, this definition works quite well. However, it creates confusion on some of the finer points of what should count in calculating a team's velocity. This confusion comes about because there are really two more precise ways of defining velocity. Let's see what they are.

1) Velocity measures how much functionality a team delivers in a sprint.
2) Velocity measures a team's ability to turns ideas into new functionality in a sprint.

Those may sound the same. They are subtly different. To see how, suppose you hop in a river and begin swimming. After an hour, you measure how far you've traveled, and you are 2 kilometers from where you began. In agile terms, we might want to call this your velocity and say you swim 2 kilometers per hour or per sprint, if a swimming sprint is an hour long.

But, what if the river had been flowing at the rate of one kilometer per hour against you while you swam? In that case, you really swam 3 kilometers. Measured against the banks of the river, you've only moved two kilometers of physical distance. But while going forward two kilometers, you overcame being pushed backward one kilometer by the river.

So, is your velocity two or three? If we use our first definition—velocity is how much a team delivers in a sprint—then velocity is two. This swimmer delivered 2 kilometers of progress.

If we use our second definition—velocity is the ability to turn ideas into new functionality—then velocity is three. This swimmer has the ability to deliver 3 kilometers of progress per sprint, and we'd see that he was swimming in a lake with no current instead of a river.

To see how this applies to an agile project, consider the issue of whether a team should earn velocity credit for fixing bugs during a sprint. A team that uses velocity to measure how much functionality is delivered in a sprint will not claim credit for bug fixes. No new functionality has been delivered. So no points are earned.

A team using defining velocity as the ability to turn ideas into functionality, on the other hand, will claim credit for bug fixes. Their logic is that the time spent on bug fixing could have been spend adding new functionality except the product owner prioritized different work for them.

For many teams, the two definitions will yield the same value. Values will differ most for teams doing work for which they are not taking velocity credit--usually teams doing things like a lot of bug fixing or doing large amounts of refactoring.

Neither of these subtle differences in the definition of velocity is always better than the other. The one you use should largely depend on what you hope to learn by measuring it and by your expectations about the future.

If you expect the future to be just like today—that is, the team will spend the same amount of time doing bug fixing, refactoring and the like as they do now, then using velocity as a measure of how much forward progress is made will be the right answer for you.

However, if you expect the future to be different—perhaps the large refactoring and time spent fixing bugs will soon be over—then you may want to define velocity as the team's ability to turn ideas into functionality, and would then add to velocity the points given to those activities.

The most important thing is to clarify with everyone on the team, including the product owner and ScrumMaster, is exactly what your team means when they use the term “velocity.” Having a precise definition makes it very easy to answer questions that come up around what should be counted when measuring velocity.

Posted: March 18, 2014

Tagged: product owner, sprinting, scrummaster, product backlog, velocity, defect management

About the Author

As the founder of Mountain Goat Software, Mike Cohn specializes in helping companies adopt and improve their use of agile processes and techniques to build extremely high-performance teams. He is the author of *User Stories Applied for Agile Software Development*, *Agile Estimating and Planning*, and *Succeeding with Agile*. Mike is a founding member of the Agile Alliance and Scrum Alliance. Mike can be reached at hello@mountaingoatsoftware.com. If you want to succeed with agile, you can also have Mike email you a short tip each week.