Agile: Optimise for velocity over predictability

Abhishek Tiwari

Citation: *A. Tiwari*, ”Agile: Optimise for velocity over predictability”, Abhishek Tiwari, 2019. doi:10.59350/m6kre-nkk76

Published on: January 09, 2019

When it comes to agile estimation, quite often agile teams spend a big chunk of their time in heavy-weight processes like detailed story point estimation to improve the *predictability* or *accuracy* of their estimates. In other words, agile teams invest more time in estimation to reduce that variance (i.e. time estimated vs. time taken) which otherwise can be used to create features and add business value. In a nutshell, they are slowing down to be more predictable. Unfortunately, it's not good enough to be predictably slow.

In my experience, agile teams are rarely optimised for *velocity* or *productivity*. *Velocity* is a *productivity* measure of user/customer/business value delivered with each release. Ironically, agile estimates have become proxy for commitments and trust. Primarily because most organisations are wired for commitments and trust but not for *velocity* and *responsiveness*. 'Fake It Till You Make It' is a new norm in the business world which means there are a lot more pressure on software engineering teams to make commitments and push to deliver on those commitments.

Agile teams optimised for *velocity* generally deliver more value per release. Although I don't have empirical data to prove it there are clear qualitative indicators that optimising for velocity is more effective. For instance, three out of four key indicators of overall software delivery performance (i.e. *lead time*, *deployment frequency*, *mean time to restore*) are all about *velocity* than *predictability*. For curious readers, the fourth software delivery performance indicator is *change fail rate* - a measure of reliability.

Just to be clear, I am not suggesting that *predictability* is not important, all I am saying is we should optimise our agile teams for *velocity* first. Marrying predictability with productivity gives us effectiveness. If anything we want our agile teams to be effective. Also, there is a subtle difference between pushing your teams to be more productive vs. optimising for shipping the high volume of user/customer/business value with each release. Pushing for velocity obviously means culture is going to suffer. So let's not get into this debate and focus on optimisation first.

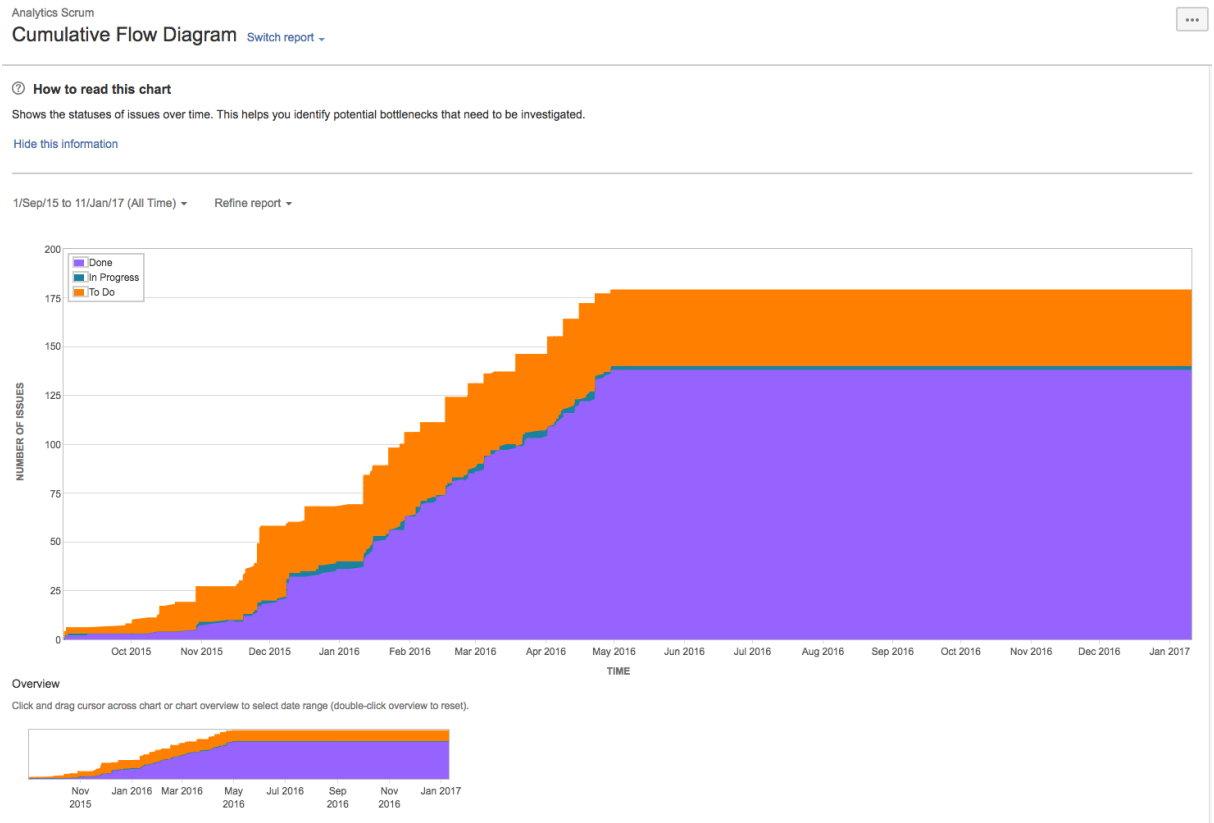

Figure 1: Are you slowing down to be more predictable?

Most importantly, optimising for *velocity* requires a culture of innovation, excellence, and empowerment. A culture where engineers have time and right kind of support to optimise end-to-end software delivery processes and their toolchains. For example, an agile team optimised for *velocity* will have the shortest continuous integration times and branch life. In other words, continuous integration typically provides the feedback in less than a few minutes and feature branches last hours or maximum a day. In this type of environment, a continuous integration job running for 30 minutes or more is considered an imposter.

Easy said than done. So how should we optimise for *velocity*? In next post we will cover more details so stay tuned.