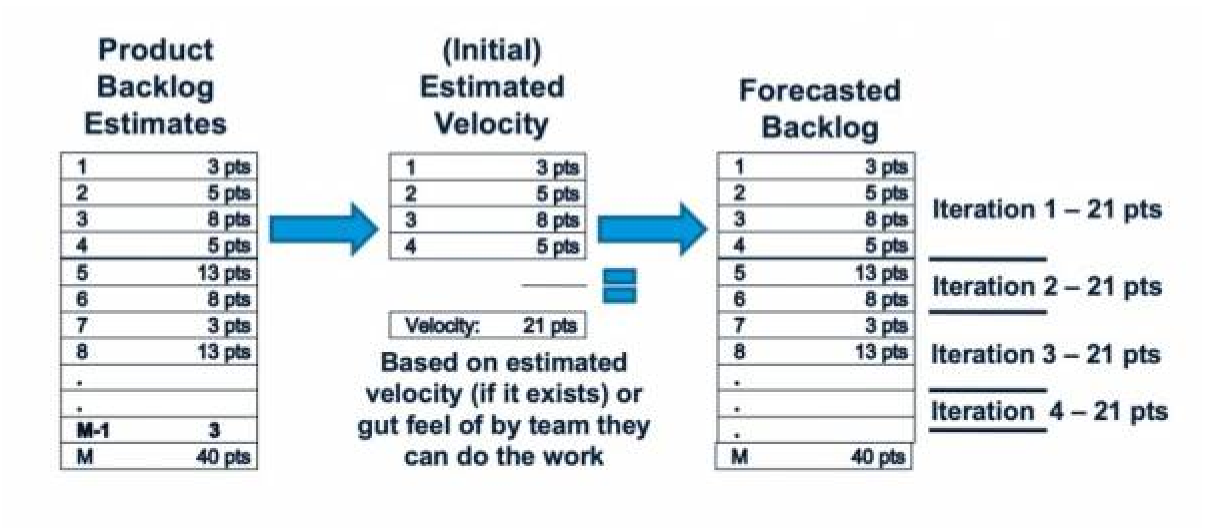
Table of Contents

How Do We Use Story Point Estimates To Forecast?

Team Velocity

We have Team based Story Point estimates. How can this information be used to forecast work?

Let's start with the simplest case first. Have a look at the following picture:



On the left you see a prioritized backlog of work in User Stories where the items in that Backlog all have Story Point estimates on them. The Team selects the set of work they think they can deliver in the next, say, 2 week period to form the “Iteration” or “Sprint” Backlog. Here the Team has selected the first 4 Stories. The estimates for these Stories are 3, 5, 8, and 5.

What the Team is saying at this point is that they think they can complete 3+5+8+5, or 21 Points of work in this 2 week period. If at the end of the 2 week period the Team actually delivers this set of work, then the 21 Points is the Team Velocity. We say “the Team's Velocity is 21 Points for this Iteration.” What this really is saying is that your Team can deliver 21 Points of User Stories in an Iteration. This is a measure of Team capacity to deliver value.

What is a good guess for how much the Team can deliver in the next Iteration? We know they can do 21 points, so we might want to use that data to say “They can deliver 21 Points in the next Iteration”. And 21 after that, and 21 after that … You get the idea. If we have a prioritized Backlog with estimates we can now forecast how many Iterations it is going to take to deliver however much of the Backlog we are interested in. (Note: This estimating approach is called “Yesterday's Weather” - What is a good guess for the weather tomorrow? Whatever it was today.)

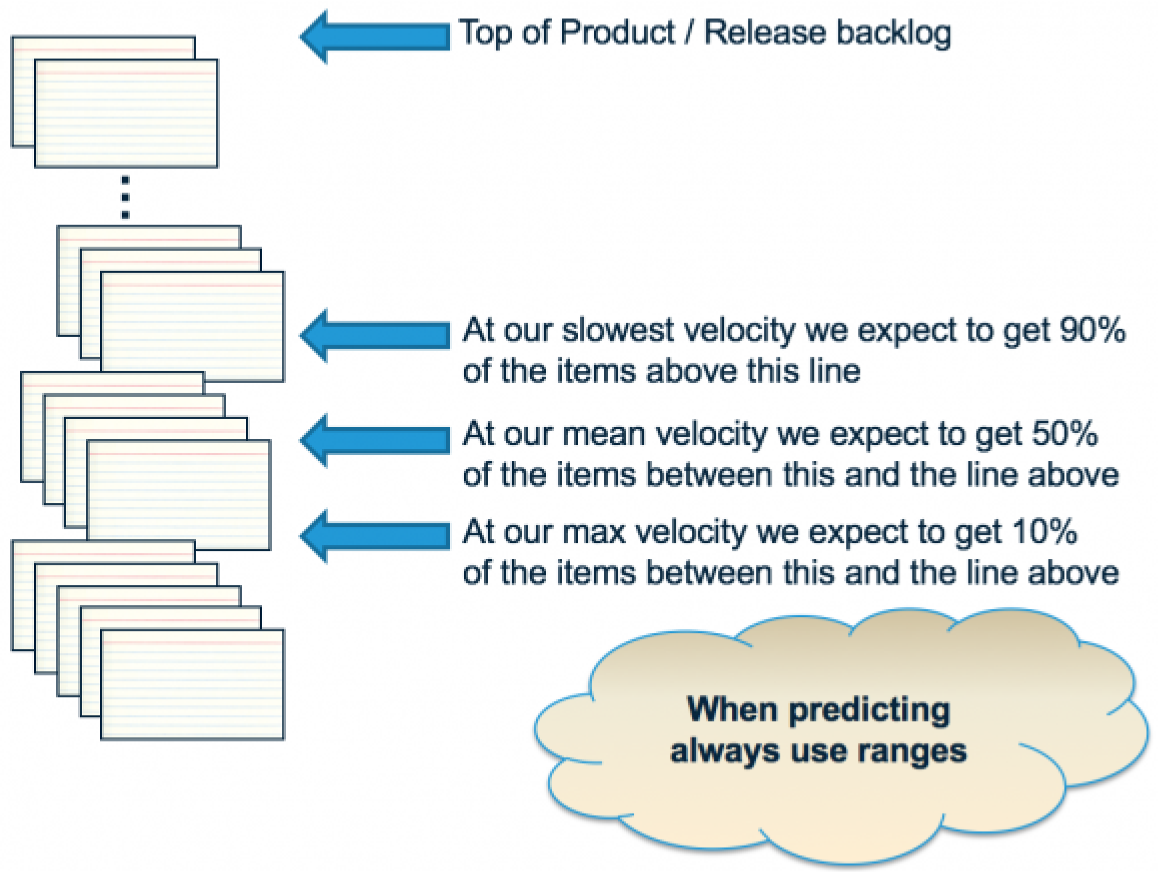
But what happens if the Team does not deliver the whole 21 Points. Perhaps they deliver on the first three items, but miss the fourth item. One of the basic philosophies of agile is that we only record “progress” when we complete work "Working software (solutions) is the primary measure of progress". If the Team has not completed the work of the fourth item, then we should not count it in the Team Velocity. The Velocity in this case would be 3+5+8, or 16 points. And, of course, the information we use to forecast should change as well. After all, the Team was not able to do the 21 Points they originally thought. A good guess for the next Iteration, based on the data we have now, is 16 and so on going forward.

Of course the reverse is also true. The Team might have completed the 21 points before the end of the Iteration. They taken on another Story. They think “Looking at the Backlog, the 13 is too big to do with what remains of this Iteration, so is the 8, but perhaps we can take on the 3.” If they take this on and complete the work, what is the Team's Velocity? We completed 3+5+8+5 + 3, so the Team Velocity is 24 Points and we can use that information for forecasting.

The reality is that Teams do not produce the same Velocity every Iteration. Sometimes they do more. Sometimes less. Over time Teams typically get better as they learn how to work together. Every couple of weeks there is another data point. So a Team might see:

Iteration	**Velocity**
1	21
2	24
3	20
4	30
5	28

The Product Owner should leverage this information in their forecasts. A typical approach is to use three data points - the minimum, the average, and the maximum - to be more informed about the forecast. So here the minimum is 21 Points, the maximum is 30, and the average is around 24-25. We can use this information with our prioritized backlog:


Top of Product / Release backlog
At our slowest velocity we expect to get 90% of the items above this line
At our mean velocity we expect to get 50% of the items between this and the line above
At our max velocity we expect to get 10% of the items between this and the line above
When predicting always use ranges

The way to think about this is to look at the items between the average and maximum Velocity - these have a significant risk of not happening based on current knowledge. If you see things there that are a significant problem, then these need to be worked now. Perhaps we need to re-prioritize the Backlog to bring in the items of concern. Perhaps we need to find another Team to do the work. Perhaps …

Many people worry “But what if we get the estimates wrong?” Here's what is interesting about the approach. If we had guessed (OK, estimated) the set of items above as 5+8+13+8, instead of 3+5+8+5, and we completed this work in the iteration. Instead of a 21 point Velocity, the Team now has a 34 point Velocity. If we assume the rest of the data in the Backlog has a similar bias (a pretty good assumption), and guess 34 points for the next Iteration, then … it will produce about the same number of items in the forecast. In other words, this approach to estimating is self-correcting with respect to the forecast. That's not to say that you don't have to work to improve estimates (see What Can We Do To Improve Our Point Based Estimates?), but rather there is less concern here than at first blush.

Note that some people worry that this isn't enough data to be significant. It may not be in a statistical sense, but it is sufficient for our purposes in that it is recent and current. And the reality is that you often have more data than you think - see How Can We Forecast When We Do Not Have a Lot of Data? for more information.

How Do We Scale Forecasting?

It should come as no surprise that the scaling for forecasting the data takes a similar approach to the approach we take when scaling the estimating approach beyond a Team.

To create a Team Velocity, we need to have Estimates on the Stories, and we needed a time-box (2 week Iteration) at which point we measure the amount of “done” work. The Team Velocity is the sum of the Story estimates completed in the 2 week period.

We have estimates against the Features and the Epics, whether via pure Feature / Epic Points or via Summated Story Points. If we decide we want to understand the capacity of a team-of-teams (Train) organization to deliver Features, all we need to do is determine the time-box to measure this over. Most organizations settle on some kind of quarterly cadence. To calculate the Train Velocity, sum up the estimated Feature Points completed in the quarter. Lets say we get a value of 500 Points. This means the Train can deliver 500 Points of value per quarter - a good indicator for the capacity of the Train for the future.

Similar thinking can be applied to Epics at the Portfolio level.

How Do We Deal with Carry-Over Features?

If we are in the first quarterly increment (SAFe's Program Increment or PI) 1 and we find that we have not completed work on a Feature, what should we do with that Feature as we head into PI 2. In general, what we would like to do is determine whether it is better to work on new PI 2 Features, or finish off the remaining amount of work on the Feature from PI 1. To do this “correctly” need to determine whether it it better to invest our capacity in the remaining work on the Feature or to stop and work on new Features. This means, for the purposes of scheduling the carry-over Features, we need the “job size” to reflect remaining work when doing the WSJF calculation. This will have the natural effect of moving unfinished work higher because you'd expect it to be smaller. If we only have a couple of hours left on something and we get “all this value” then we should do this first.

This thinking is really an example of:

> “what is the best use of our people’s time?“

In reality re-estimating the Features can be a lot of work and the end result is often “we need to finish the Features we have started.” Most organizations end up saying that “we will treat Feature carry over work differently”. Let's say we have some uncompleted Features in the current PI, PI 1. Rather than re-estimating the Feature for work that is carried over, we will simple treat carry-over work as “top priority” and assume that it will be scheduled first. These become the top items on the list for PI 2. Then the only thing we do a WSJF on is “new” PI 2 stuff that is coming out of the Program Backlog.

If you do this process, you would want to do a quick review of the carry over work asking “have we learned something that would say we shouldn’t just finish the work of this feature in the PI?” You are looking for things that still have a lot work to complete. For example:

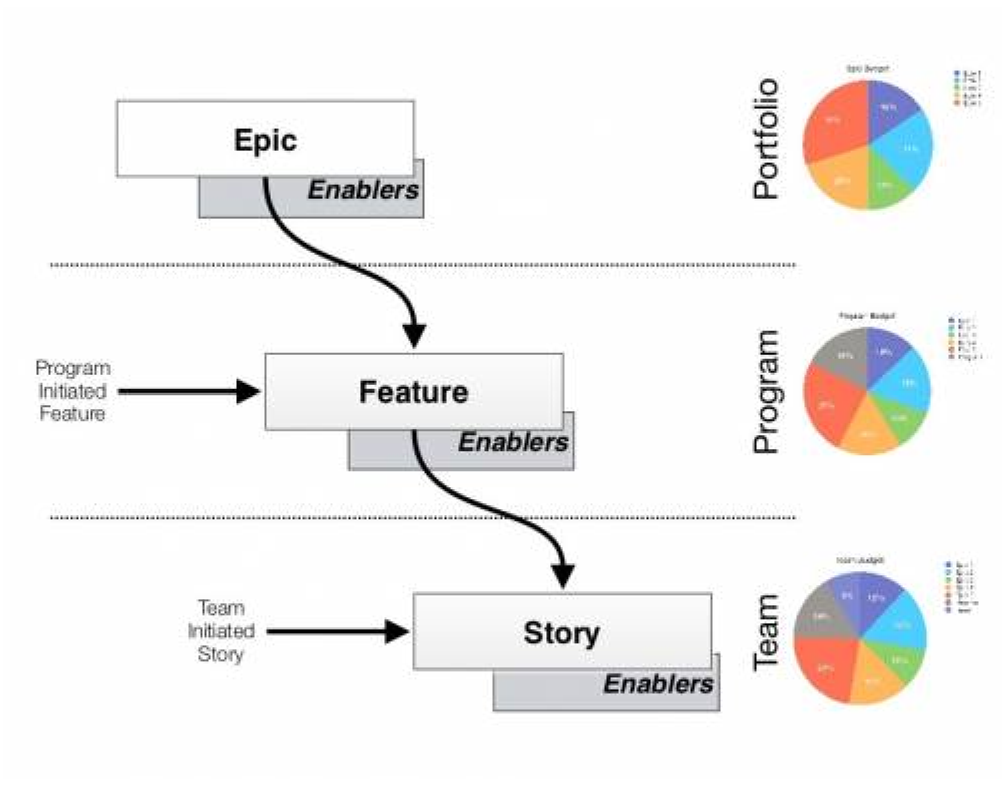
- Perhaps you have an item that actually was never started in the PI 1, or only had a small amount of work completed, leaving a lot.
- Perhaps now that we have done the work in the PI 1 we find that that the remaining work is “another PI or 2” to complete.

These items should not be treated as carry-over but rather should be compared (WSJF) to all the other PI 2 items coming in to determine if they really are the most important things to work on in PI 2.

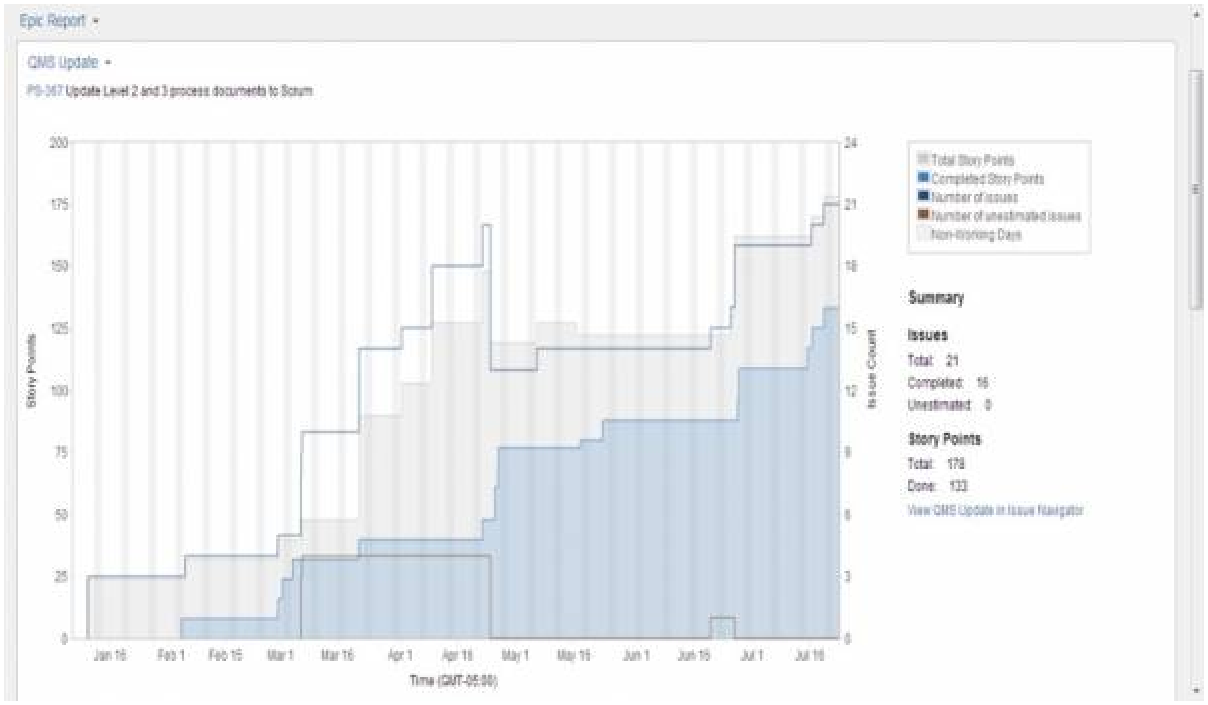
Similar thinking can be applied to Epics at the Portfolio level.

How Do We Show Progress of Work for a Feature or Epic?

Requirements typically are represented in an organization as a hierarchy: Epics breaking down into a number of Features, Features into a number of Stories:



This structure is typically reflected in the work management tool you have. So Stories will be parented by Features. Features by Epics. You can then show completion of Features and Epics by showing the Story Points completed versus the Story Points still to be completed. The follow chart is an example showing progress toward completing an Epic:



Summary

Issues

Total: 21
Completed: 16
Unestimated: 0

Story Points

Total: 178
Done: 133

View QMS Update in Issue Navigator

Notice that the amount of work we have associated with the Epic goes up as well as down. This is expected. As we do the work, we learn more and more about the business need and how we address that need. Sometimes this means we identify additional work to do; sometimes it means we can reduce the amount of work to do. This is the point of agile - scope is not fixed - and we will evolve our plan by continuously re-planning the work based on what we know today.

Want to Know More?

- Why a Plan Based on Average Velocity Will Fail?
- Why Should We Work Harder to Eliminate the Effect of Dependencies?
- How Can We Forecast When We Do Not Have a Lot of Data?

Team, Estimates, Forecast, FAQ, Points, StoryPoints, FeaturePoints