Agile Project Management

Agile Metrics

Metrics are the indicators to track an agile project

Metrics tell you if you are on track and if you need to improve your efficiency

Metrics reduce confusion as they provide information about tracking, preformance and progress

The output of a project has to be:

building the right product, at the right time, for the right market

The term "**Done**" used to specify that the delivery of a requirement or a product does not tell you that you matched the a.m. requirement

Staying on track throughout the program means collecting and analysing some data along the way

In any agile program, it's important to track both business metrics and agile metrics

Business metrics focus on whether the solution is meeting the market need

Agile metrics measure aspects of the development process

AGILE METRICS

We will see and analyse the agile metrics focused on the delivery of software

Each of these agile metrics will help the team better understand their development process, making releasing software easier

This is valid for Scrum and Kanban teams

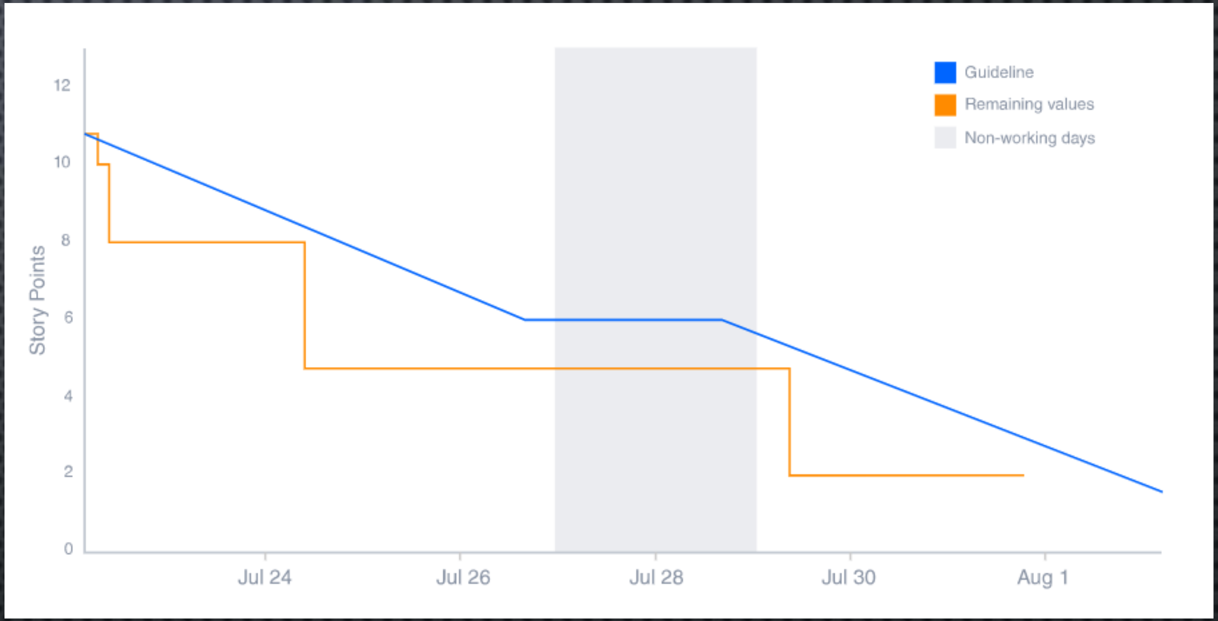
SPRINT BURNDOWN

A sprint burndown report tracks the completion of work throughout the sprint

The x-axis represents time

The y-axis refers to the amount of work left to complete, measured in either story points or hours

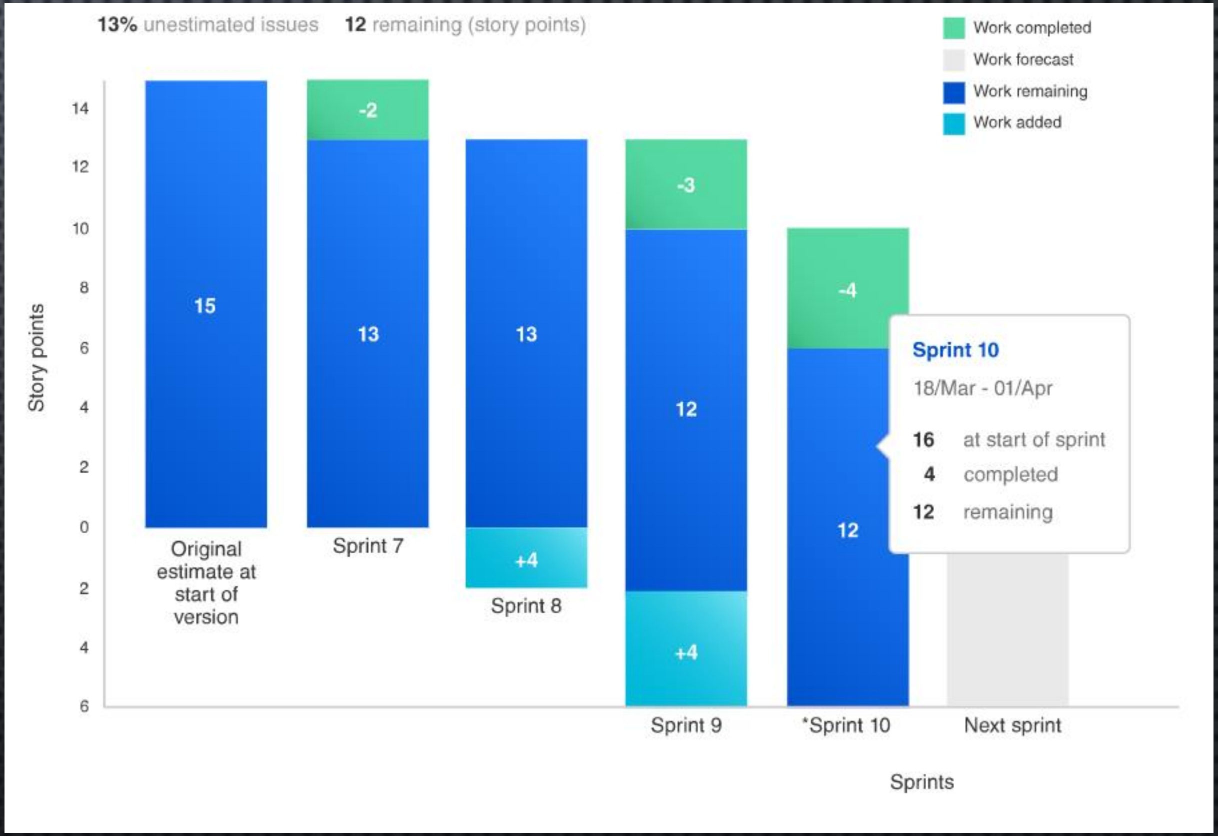
The goal is to have all the forecasted work completed by the end of the sprint



EPIC AND RELEASE BURNDOWN

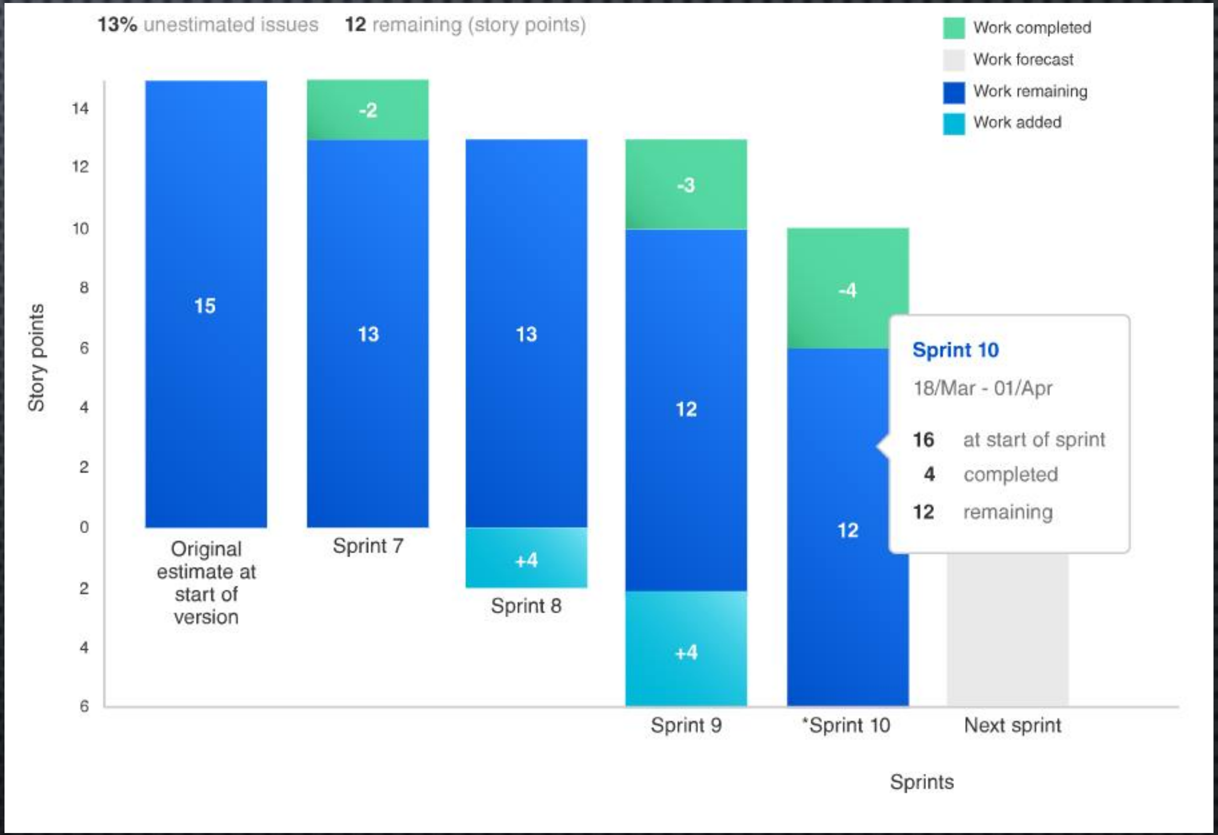
Since a sprint (for scrum teams) may contain work from several epics and versions, it's important to track both the progress of individual sprints as well as epics and versions

Epic and release burndown charts track the progress of development over a larger amount of work than the sprint burndown



EPIC AND RELEASE BURNDOWN

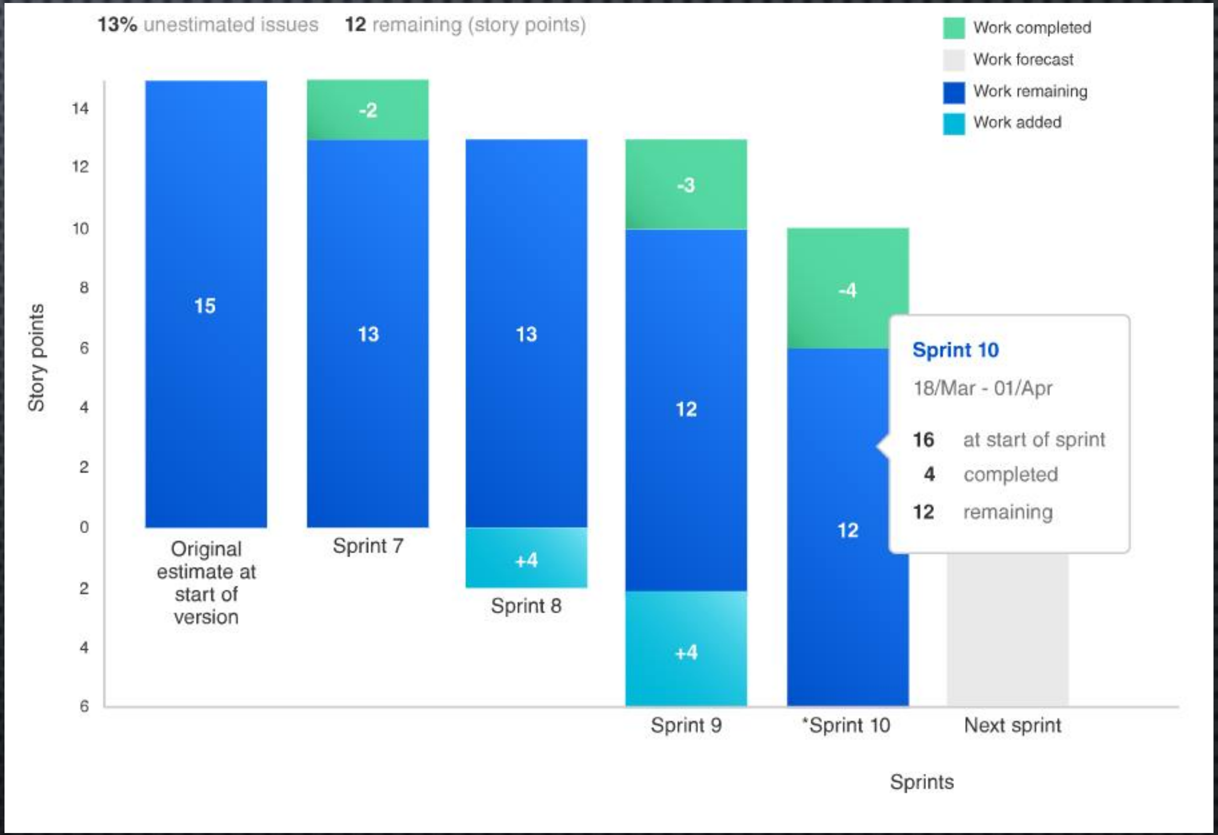
Adding new requirement is common for epics and releases
During a sprint it is a bad practice to add requirements, while scope change within epics and versions is a natural consequence of agile development



EPIC AND RELEASE BURNDOWN

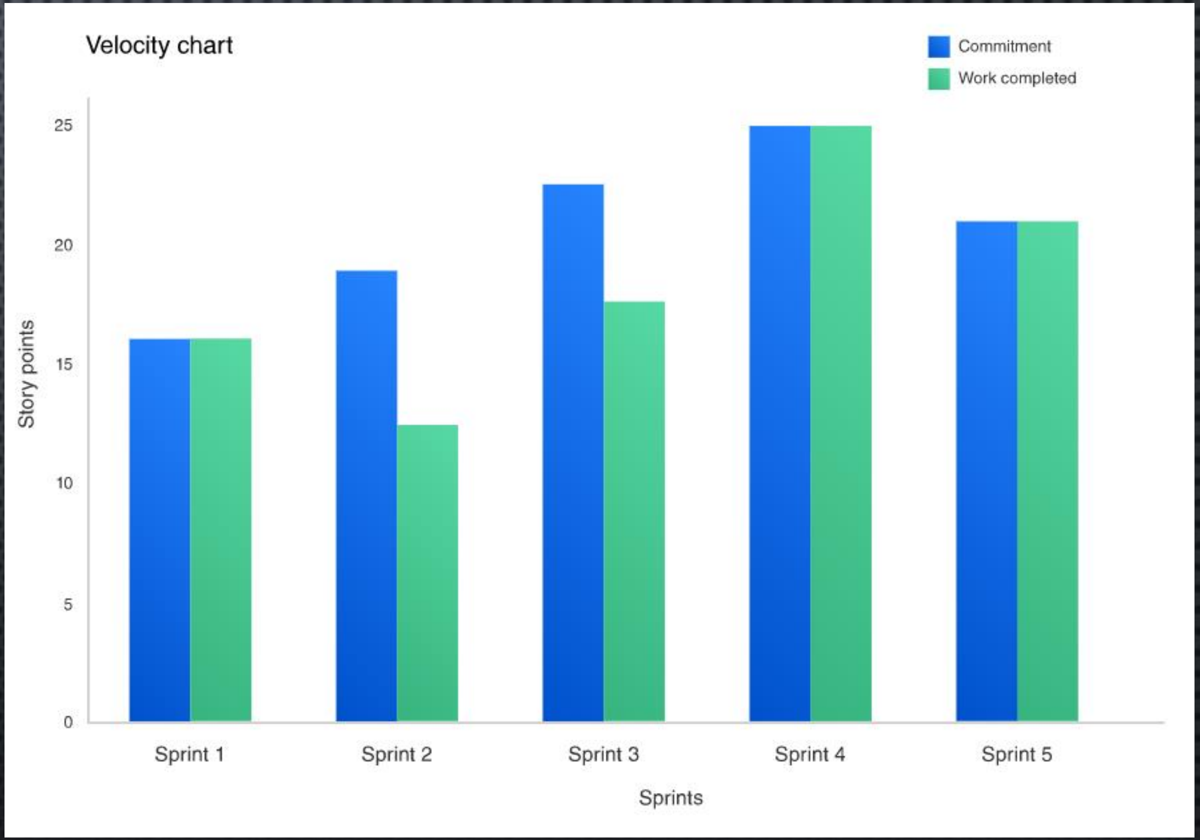
As the project progresses, the product owner may decide to take on or remove work based on what the team and stakeholders are learning

The epic and release burn down charts show the work performed inside the epic and version



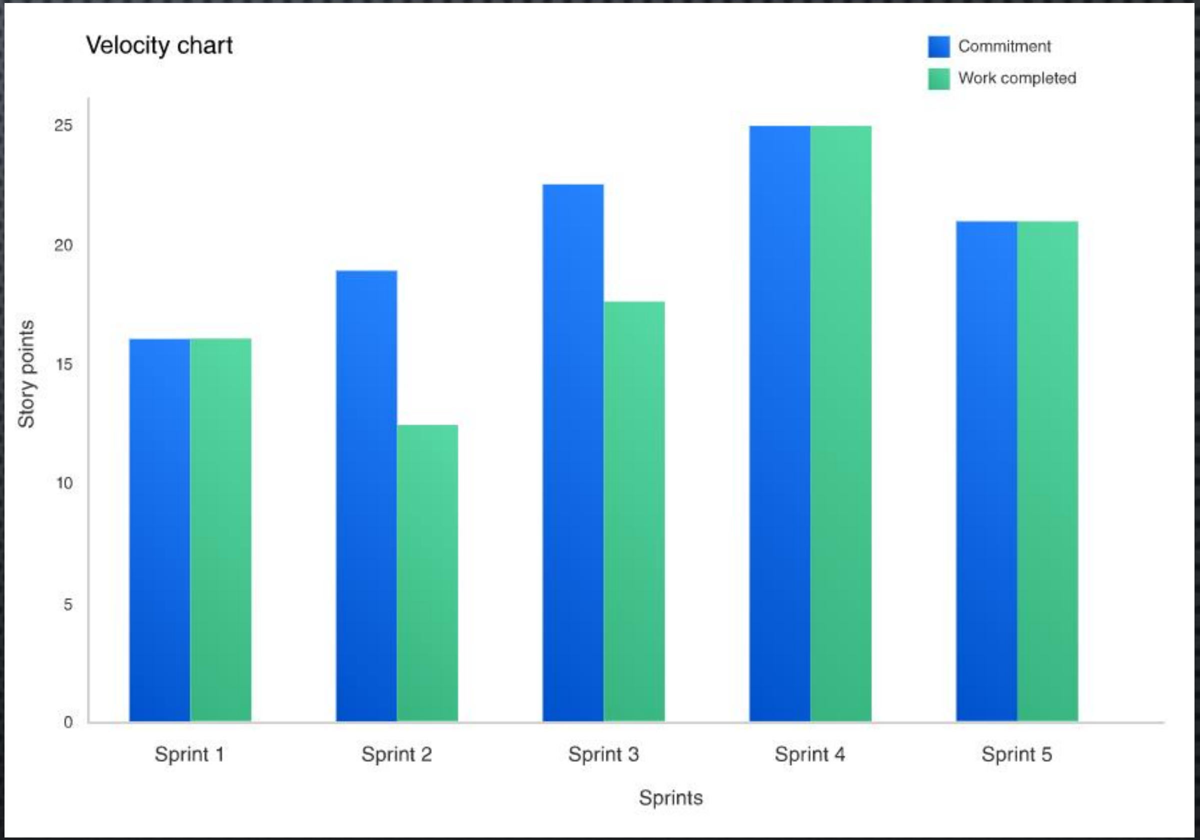
VELOCITY

Velocity is the average amount of work a scrum team completes during a sprint
It is measured in either story points or hours, and is very useful for forecasting
The product owner can use velocity to predict how quickly a team can work through the backlog



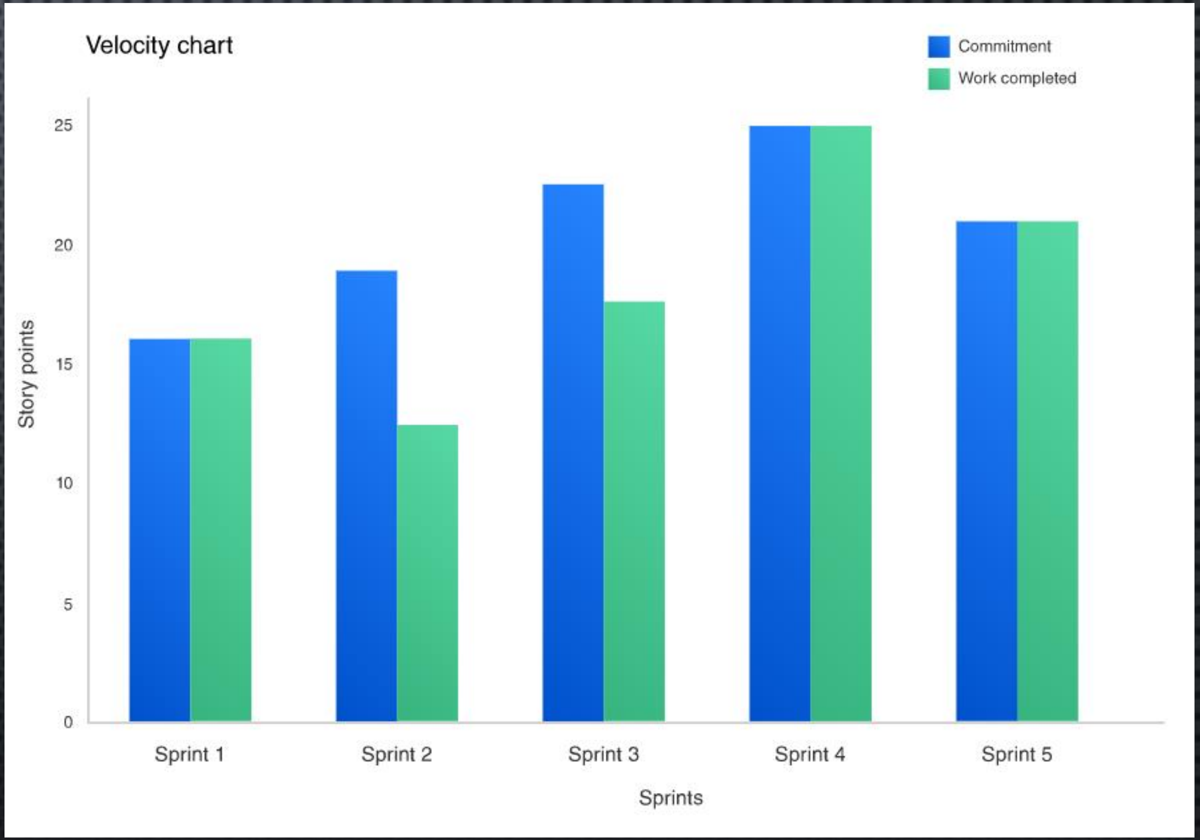
VELOCITY

In fact the report tracks the forecasted and completed work over several iterations
The more iterations, the more accurate the forecast



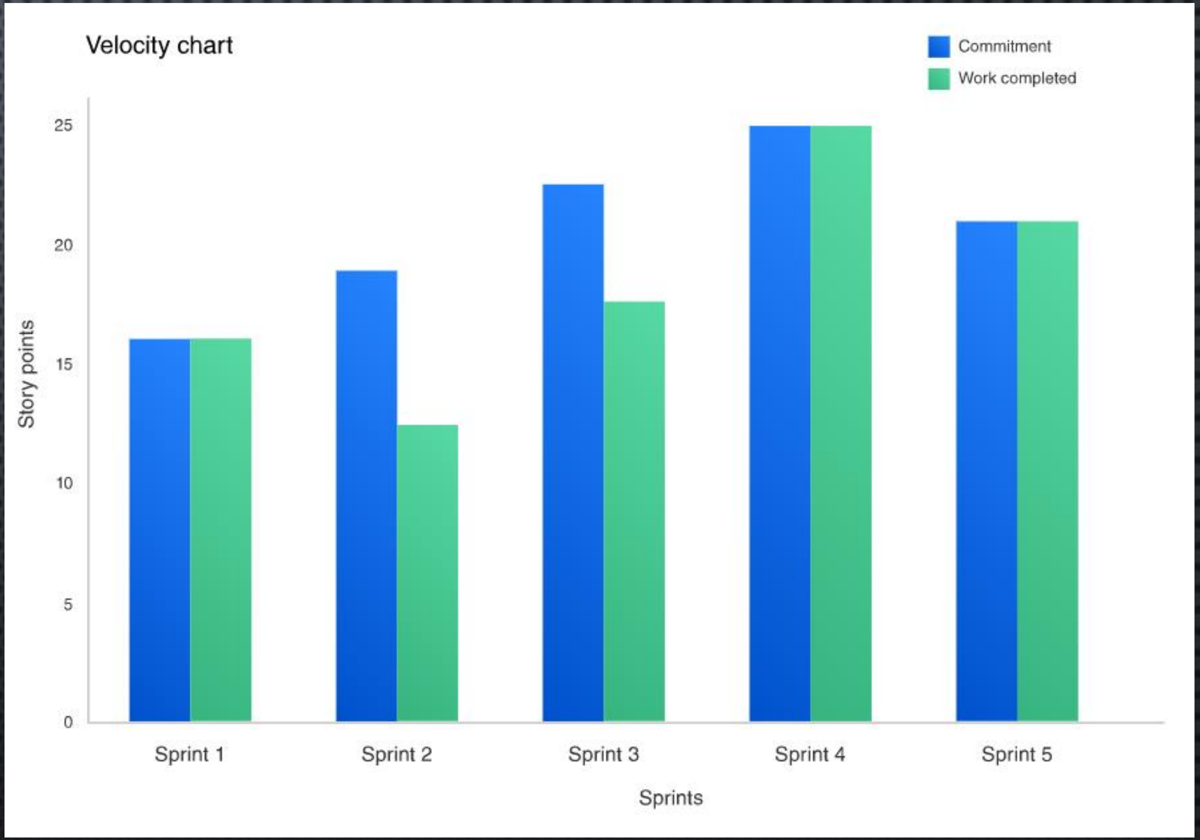
VELOCITY

Let's say the product owner wants to complete 500 story points in the backlog
We know that the development team generally completes 50 story points per iteration
The product owner can reasonably assume the team will need 10 iterations (give or take) to complete the required work



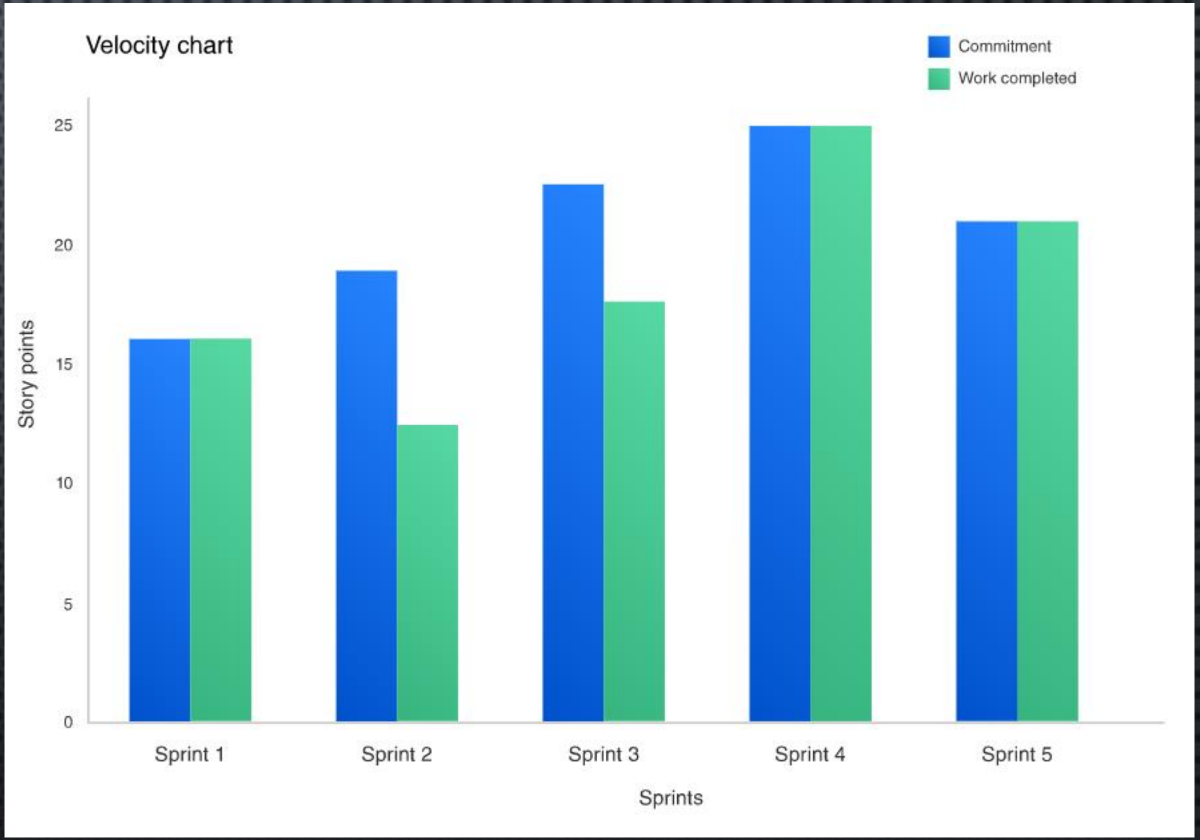
VELOCITY

It's important to monitor how velocity evolves over time
New teams can expect to see an increase in velocity as the team optimizes relationships and the work process
Existing teams can track their velocity to ensure consistent performance over time, and can confirm that a particular process change made improvements or not



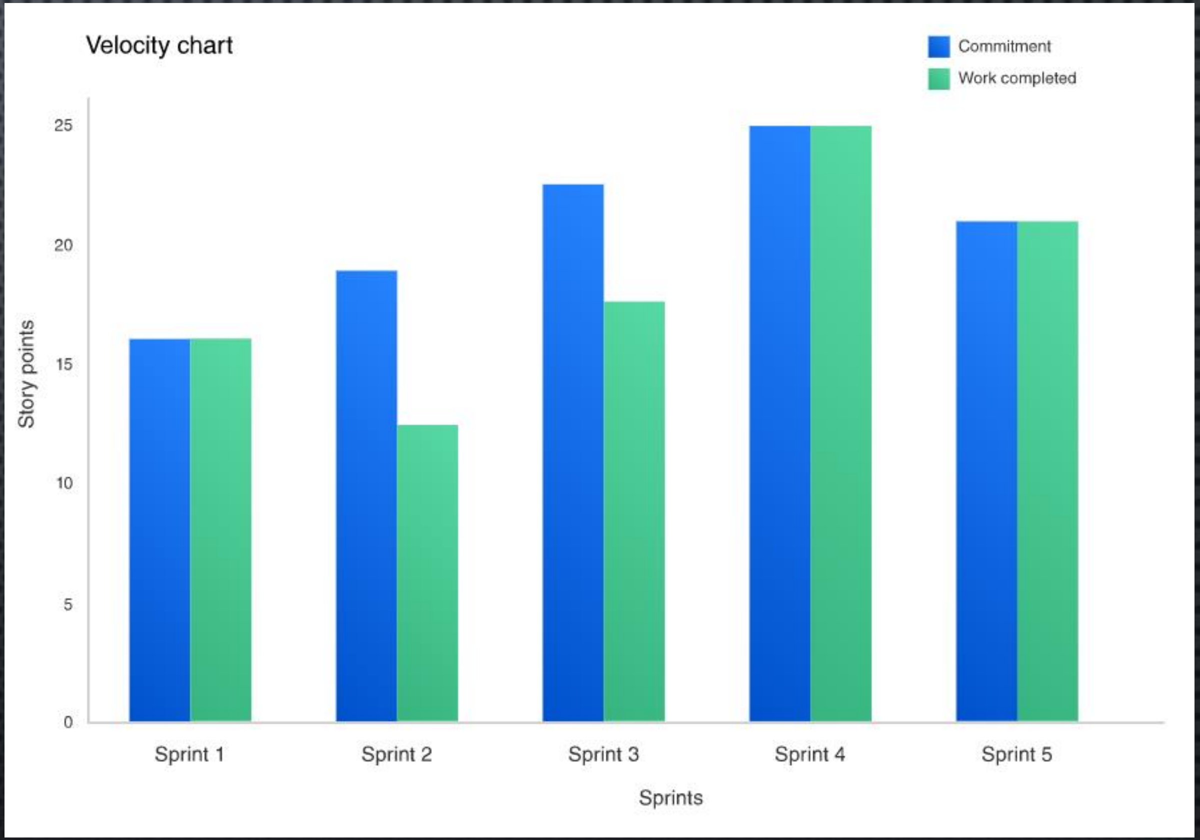
VELOCITY

A decrease in average velocity is usually a sign that some part of the team's development process has become inefficient and should be brought up at the next retrospective



VELOCITY

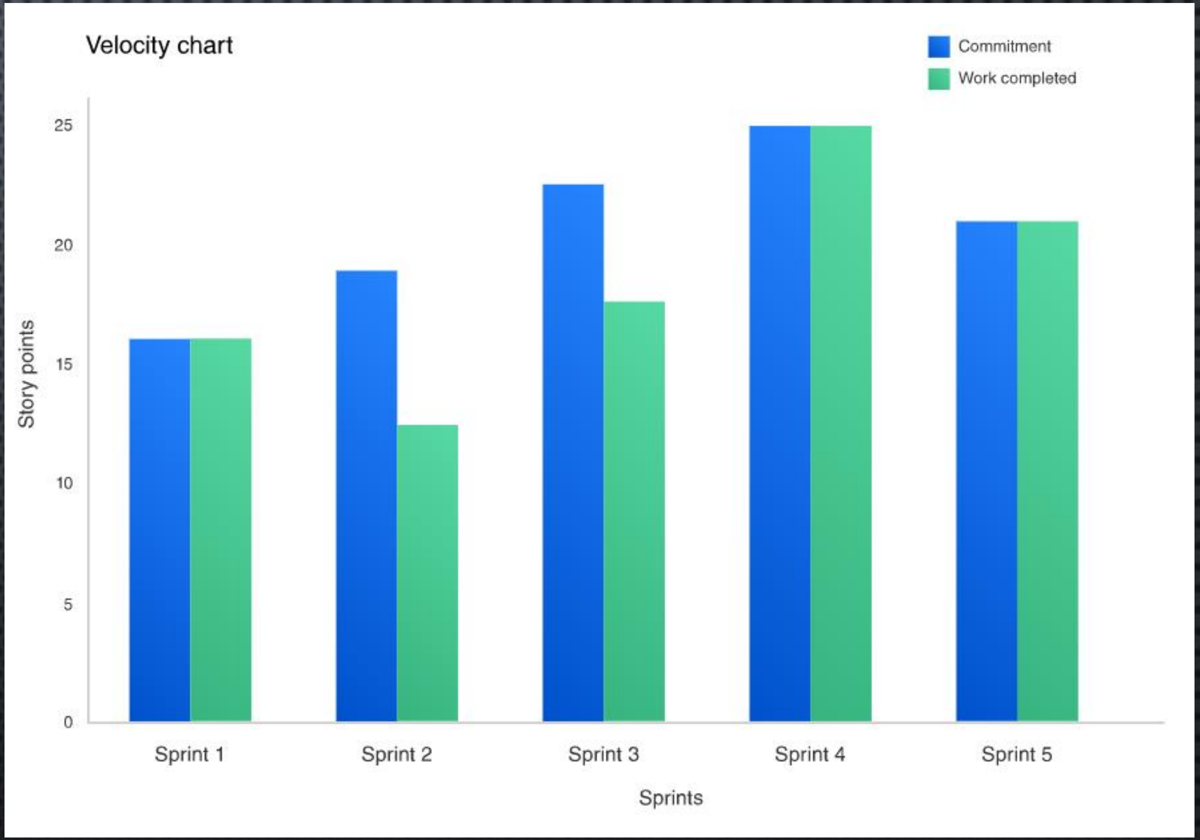
Each team’s velocity is unique
If team A has a velocity of 50 and team B has a velocity of 85, it doesn't mean that team B has higher throughput
Since each team's estimation culture is unique, their velocity will be as well



VELOCITY

Do not compare velocity across teams

Measure the level of effort and output of work based on each team's unique interpretation of story points



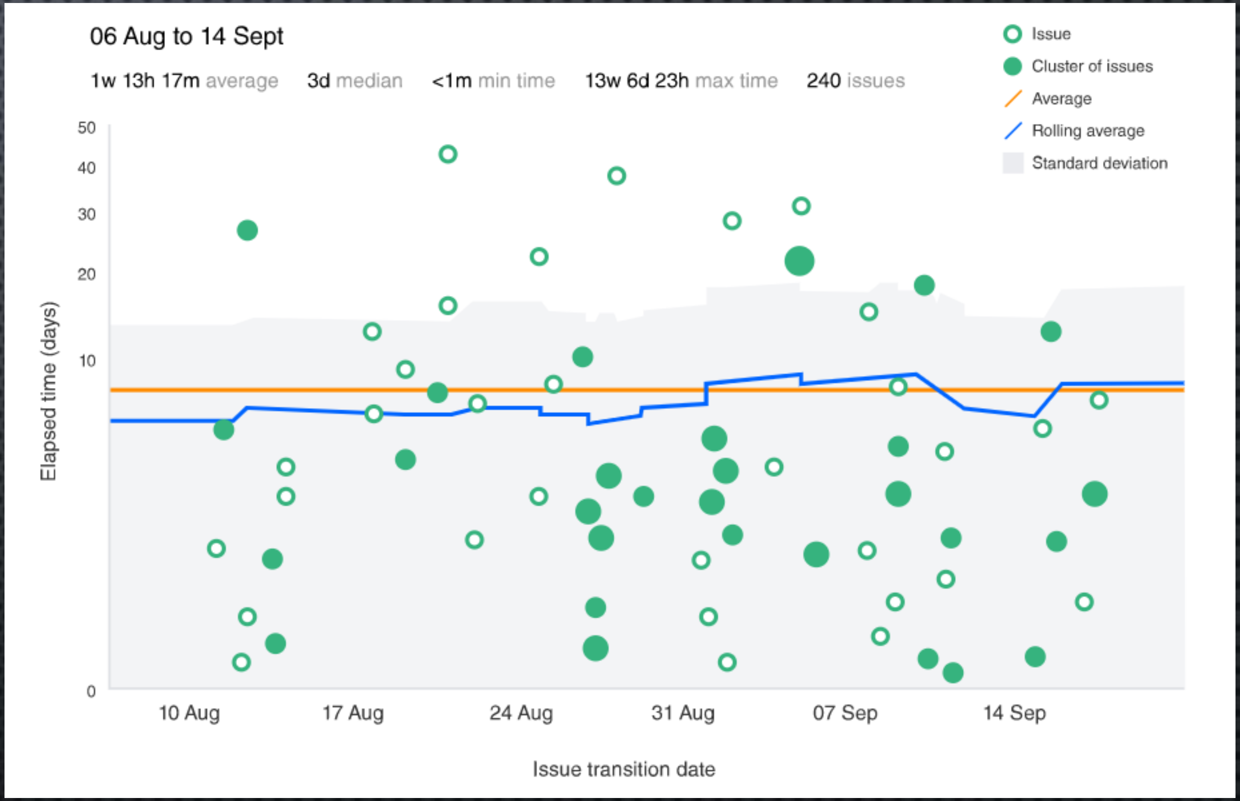
CONTROL CHART

Control charts focus on the cycle time of individual issues

The total time from **"in progress" to "done"**

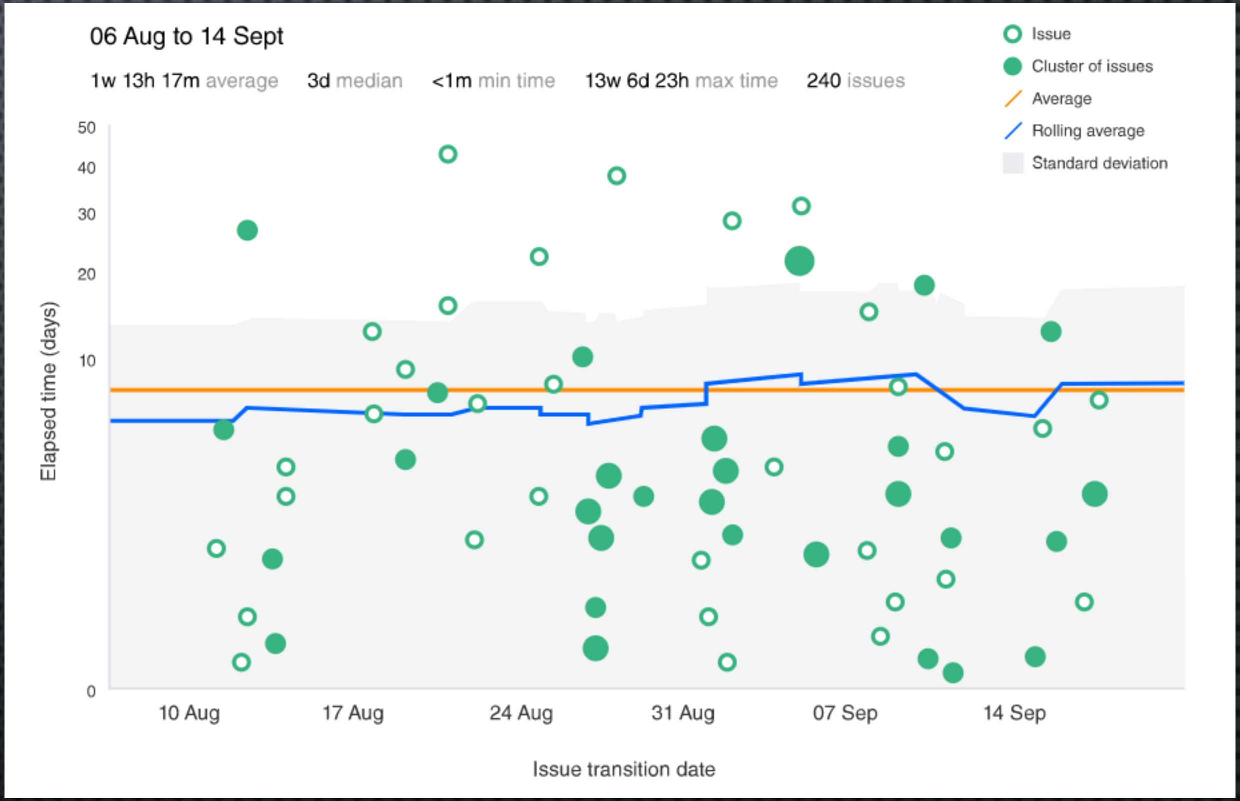
Teams with shorter cycle times are likely to have higher throughput

Teams with consistent cycle times across many issues are more likely in delivering work



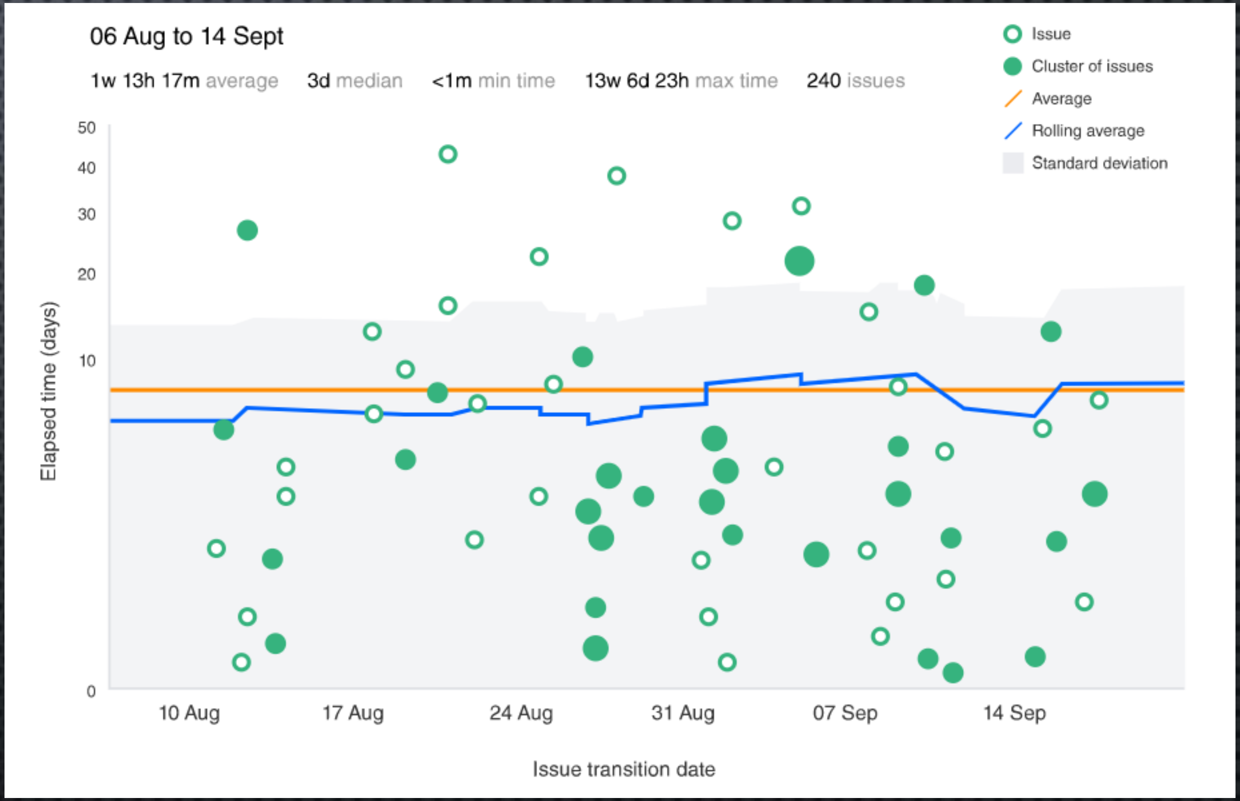
CONTROL CHART

Measuring cycle time is an efficient and flexible way to improve a team's way of working and workflow
In fact the results of changes are retrieved almost immediately and the team is thus able to do fixes immediately



CONTROL CHART

The end goal is to have a consistent and short cycle time, regardless of the type of work



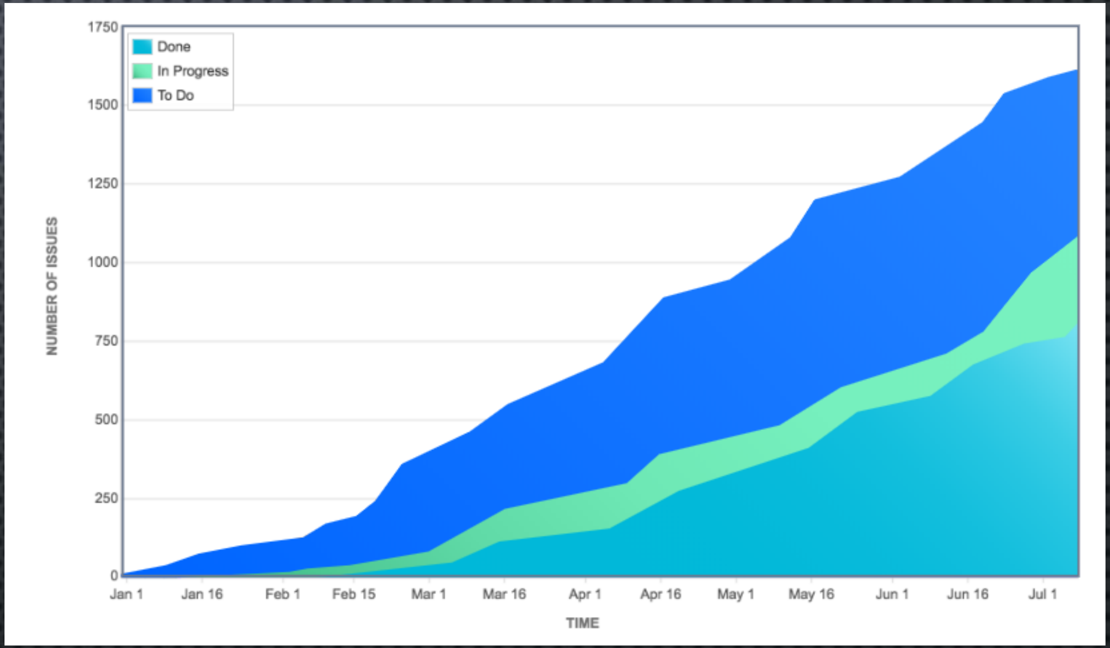
CUMULATIVE FLOW DIAGRAM

The cumulative flow diagram helps Kanban teams to verify the consistency of the flow of their work

On the Y axis the number of issues

On the X axis the time

For each status of the workflow a different colour is used



Even more metrics **(talk only)**
Good metrics aren't limited to the reports discussed above. For example, quality is an important metric for agile teams and there are a number of traditional metrics that can be applied to agile development:

How many defects are found...
during development?
after release to customers?
by people outside of the team?
How many defects are deferred to a future release?
How many customer support requests are coming in?
What is the percentage of automated test coverage?
Agile teams should also look at release frequency and delivery speed. At the end of each sprint, the team should release software out to production. How often is that actually happening? Are most release builds getting shipped? In the same vein, how long does it take for the team to release an emergency fix out to production? Is release easy for the team or does it require heroics?

Metrics are just one part in building a team's culture. They give quantitative insight into the team's performance and provide measurable goals for the team. While they're important, don't get obsessed. Listening to the team's feedback during retrospectives is equally important in growing trust across the team, quality in the product, and development speed through the release process. Use both the quantitative andqualitative feedback to drive change.