Basics of a Successful Scrum

Agile and Scrum

Scrum is Agile. Agile is not Scrum.

Every project is different, and each presents unique challenges. We often address some of these challenges by modifying our software development processes.

After completing many projects, we at treXis asked ourselves, “What development processes were common in the projects that went smoothly?”

What emerged were a few key aspects of the Scrum methodology that supported our team in delivering high-quality solutions within time and budget constraints. These aren’t a silver bullet, but they can help maintain clarity when unforeseen issues suddenly make delivery much more complicated.

A | Agile

If you read the 12 Principles of the Agile Manifesto, you will not find the word ‘sprint’ or ‘story’ or any of the terms people associate with Agile. Instead, you will find ideals to follow and goals to pursue. These goals are important to keep in mind because they explain **why** we follow an Agile process; **what** it is supposed to accomplish; and provide guidance on **how**. However, it doesn’t explain *exactly* how.

Words such as sprint and story are commonplace because of the popularity of the Scrum framework. This paper expects the reader to have some familiarity with Agile and Scrum and focuses on the key factors for success.

S | Scrum

Scrum is a methodology for delivering Agile results. It specifies processes which — when done properly — should support the realization of the 12 Agile Principles.

Scrum is popular because it also allows project management to answer the kinds of business questions that come from senior management.

- How long will this take?
- What are we going to have done by the deadline?
- How will it impact the project if we decided to introduce a new requirement at this point?
- What was the business value driving the implementation of a particular feature?

We’ll revisit these questions later and see how Scrum helps you answer them.

Three Keys to Scrum Success

If you get nothing else right, get this right.

1 Velocity

Velocity is the number of story points the team can complete per sprint. This is the critical KPI for the software development team.

Successful management of a high-performance team relies primarily on establishing an accurate and stable velocity measurement. Furthermore, the team should be able to increase its velocity by 10% to 20% over the course of a project. Establishing a meaningful velocity value depends on:

- **A stable team:** Team changes affect the team's velocity. Imagine replacing a mid-level developer with a rock-star senior; you would expect the velocity to increase. That said, changes in team composition tend to reduce team cohesion and reduce velocity. If a change introduces a more productive member, you might see a net increase in velocity but it might not happen right away.
- **Uniform skills:** Velocity is a team metric. Any story assigned to the team should be assignable to any team member. Team members may vary in their level of skills, but they should have the skillset to work any sprint story. Another way to put this: *story points must be fungible.*
- **Good estimation:** Story points are somewhat abstract but should be a meaningful measure of work. When the team estimates a particular story, they need to estimate it consistently with other stories. There are techniques such as Planning Poker that achieve this.

2 Estimations

As noted above, a great deal rests on the ability to accurately estimate the story points needed to work a story. This is important for bringing the correct amount of work into a sprint *and* answering management questions about project timelines. Estimates start with giving each backlog item a T-shirt size. Anything larger than a medium should be broken into smaller stories. Smaller stories are easier to estimate accurately. Story point estimates must be made on stories as they are added to the sprint, and preliminary story point estimates *should* be made on stories for the upcoming sprints. T-shirt sizes can be used for planning by converting them into estimated story points after there is some historical basis for this conversion.

3 Backlog

There are two important goals for the backlog that must be maintained and improved in Backlog Refinement sessions. The first is that the backlog must be sorted in priority order. The highest priority stories should be prioritized more carefully as they are candidates for upcoming sprints. Priority is determined by the product owner (PO). The second goal is that all backlog items must be estimated or at least given a T-shirt size. The T-shirt sizes can be converted to estimated story points for the purposes of planning. Estimation is done with the team by the Scrum master (SM). The SM never determines priority and the PO never assigns work to a sprint.

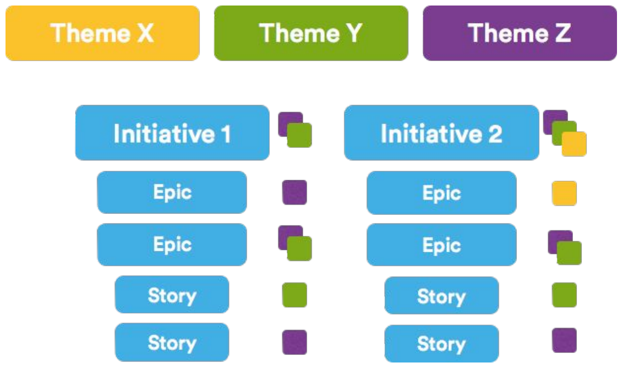
Stories and Sprints

Sprints are the engine of producing code. Stories are the fuel.

S

Stories

Stories are workable pieces of functionality. Stories should be formulated using INVEST criteria



Smaller stories are better: they are easier to accurately estimate and less likely to have surprises. No story should be expected to take more than one sprint.

The work needed to complete a story is estimated in 'points'. Points are a subjective measure unique to a team. If you must think in time, start with one point = ½ day.

The project should be broken down hierarchically. This makes it easier to break down the work and allows tracking of how each work item relates to business priorities.

- **Themes:** Broad business goals, e.g. 'Expand online presence'. Work items may align with multiple themes.
- **Initiative:** This is a specific business goal that aligns with one or more themes.
- **Epic:** A collection of functionality that implements an initiative.
- **Story:** One item of functionality within an epic.

S

Sprints

A sprint is a fixed time during which the team produces a new releasable increment (RI). The RI may not actually be released but it should be of releasable quality.

Each sprint boundary offers the opportunity to course-correct the project. The longer the sprint, the fewer opportunities you have over the life of a project. Two weeks is standard but if the project direction is volatile, shorter may be better. If the project direction is very stable, longer may be more productive.

A sprint is bounded by ceremonies:

- **Sprint planning:** Choose stories to fit the sprint. Points should equal team velocity. This is why estimating points accurately is critical, as is measuring team velocity.
- **Sprint review:** Meet with the product owner (PO) to demo the work done during the sprint. The PO should accept any story that meets its acceptance criteria (AC). (This story is then done.)
- **Sprint retrospective:** A meeting for the team to identify things to stop doing, start doing, or keep doing to make sprints productive.

The Backlog

The backlog tracks all of the work needed to fulfill the requirements.

The backlog is all the work left to do. It should be sorted in *descending order* of business value. This means all the completed stories and all of the stories in the current sprint are more valuable than any story left in the backlog. *An agile project always completes the most valuable work first.*

At some point after starting, the project will reach the status of minimum viable product (MVP). This marks the point where enough has been achieved that the project could be released and provide some meaningful value. This may not be enough value to make it worth releasing, however.

After MVP, further work increases the value of the product, with the most valuable contributions coming first.

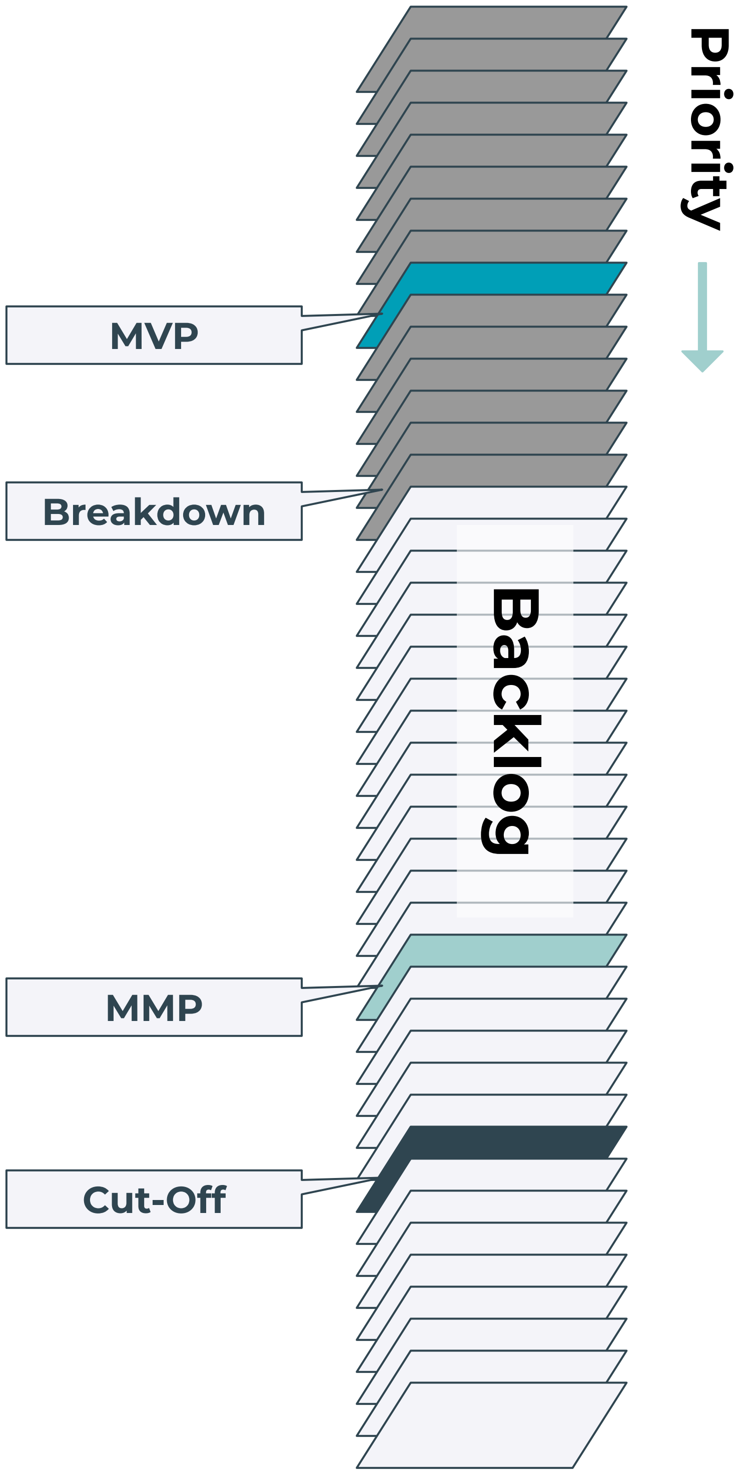
There may be another milestone, minimum marketable product (MMP), indicating the product has enough value that it would be worth the effort to release and promote to the end users.

After MVP (or MMP if defined), the product is clearly an asset to the organization and further work makes it a more valuable asset.

Theoretically, the project ends when the backlog is consumed. Consuming the backlog is called 'burndown'.

Realistically, the project ends when funding runs out or the market window begins to close, usually at a given date. This will usually leave items remaining in the backlog. This is why it is critical to work in order of decreasing priority.

If the MMP point in the backlog is past the cut-off point, the project will need to be extended or will go out with a reduced scope for MMP.



Taming the Backlog with Velocity and Estimation

How to answer pesky questions about timelines, costs and costs of scope creep.

How long will all this take?

$$T = P_{Backlog} \div V \times L_{sprint}$$

To determine the time for a complete burndown, divide the remaining points by the velocity and multiple by sprint length.

What will be completed by the deadline?

$$P_{end} = V \times (\text{remaining sprints})$$

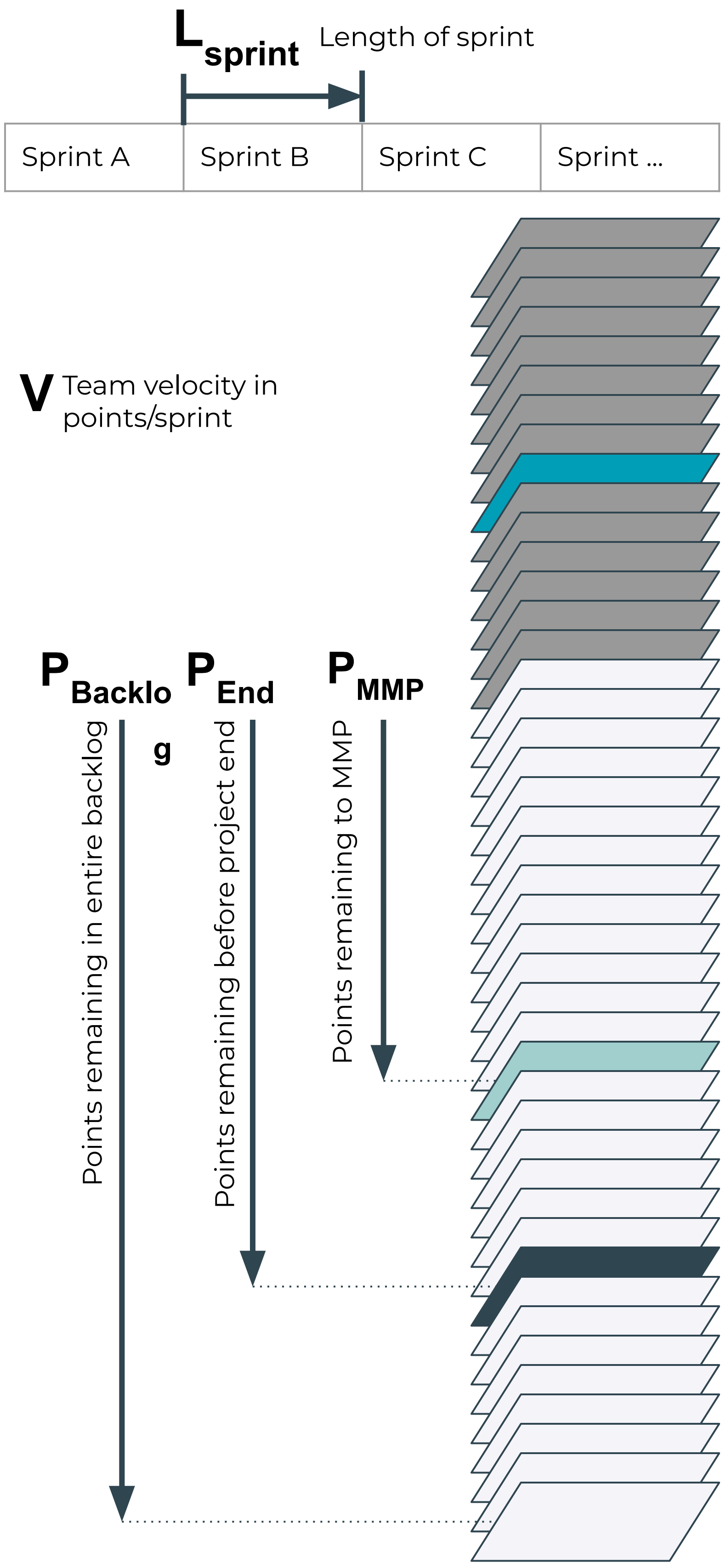
Gather stories from the backlog (in priority order) until the points equal P_{end}. This is the scope of achievable work in the time remaining.

How will it impact the project if we decided to introduce a new requirement at this point?

Add stories to backlog in their proper priority position. Can P_{end} change? If not, the insertion of new stories will bump some lower-priority stories past the cut-off. If P_{end} can change, how much would it need to push right? Determine what stories would be lost, or how much extra time is needed to avoid dropping stories.

What was the business value driving the implementation of a particular feature?

Use Agile management application (e.g. JIRA) to connect a *task* to its enclosing *story*, to the enclosing *epic*, and finally to tagged *initiatives* and *themes*. This will provide an answer as to the overall strategic value of a specific feature.



L sprint
Length of sprint
Sprint A
Sprint B
Sprint C
Sprint ...
V Team velocity in points/sprint
P Backlog
P End
P MMP
Points remaining in entire backlog
Points remaining before project end
Points remaining to MMP

Thank you.

BY ENGINEERS FOR ENGINEERS