Sprint Planning

SWEN-261
Introduction to Software Engineering

Department of Software Engineering
Rochester Institute of Technology

By Lakeworks - Own work, GFDL, https://commons.wikimedia.org/w/index.php?curid=3526338
A sprint plan is a plan to build a small, working increment of the product.

- Each sprint is time-boxed; your sprints will be about three weeks long
- Each sprint has a backlog of work to be done during the sprint
  - This is the team's commitment to the Product Owner
  - It is the sole focus on the team's effort during this sprint
- The sprint planning meeting establishes the next sprint's backlog and launches the sprint
- Let's first consider how a sprint flows...
So we start with a backlog for the new sprint.

- During the Sprint Planning meeting, stories from the Product Backlog are put on the Sprint Backlog.

Let's assume that the team can do at most 25 points worth of work in a sprint.

Stories are in the product backlog in priority order and you select stories in priority order until you are near the sprint work limit.

S9 would fit into the sprint but may have been left out because of a dependency on an unimplemented higher priority user story.

**NOTE:** In this illustration story points for each Story are provided. A future lesson will show you how to play *planning poker* so these numbers will come from the team.
How does a story get from Backlog to Done?

- When do you know that you're done with a story?

<table>
<thead>
<tr>
<th>Product Backlog</th>
<th>Sprint Backlog</th>
<th>Done</th>
</tr>
</thead>
<tbody>
<tr>
<td>S3 (21)</td>
<td>S1 (5)</td>
<td></td>
</tr>
<tr>
<td>S6 (2)</td>
<td>S2 (13)</td>
<td></td>
</tr>
<tr>
<td>S7 (13)</td>
<td>S4 (3)</td>
<td></td>
</tr>
<tr>
<td>S8 (5)</td>
<td>S5 (3)</td>
<td></td>
</tr>
<tr>
<td>S9 (1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>S10 (3)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>S11 (5)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(S10) (3)
The team must establish a *Definition of Done*.

- **Core activities:**
  - Acceptance criteria are defined
  - Solution is designed
  - Solution is developed; aka "code complete"
  - Feature has been tested (manually)

- **Other activities we'll add throughout the course:**
  - Feature branches (configuration management)
  - Unit testing
  - Code coverage analysis (goals)
  - Code reviews
  - Demo increment to stakeholders
A story goes through five "gates" to get to *Done*.

<table>
<thead>
<tr>
<th>Product Backlog</th>
<th>Sprint Backlog</th>
<th>In Dev</th>
<th>Ready for Test</th>
<th>In Test</th>
<th>Done</th>
</tr>
</thead>
<tbody>
<tr>
<td>S3 (21)</td>
<td>S1 (5)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S6 (2)</td>
<td>S2 (13)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S7 (13)</td>
<td>S4 (3)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S8 (5)</td>
<td>S5 (3)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S9 (1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S10 (3)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S11 (5)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

A story goes through this gate when *acceptance criteria* and *solution tasks* are defined, it is estimated, and then selected to be worked on in the next sprint.

A story goes through this gate when a free developer starts implementing it.

A story goes through this gate when it has been *verified* to pass all of the *acceptance criteria* and it is considered done.

A story goes through this gate when a team member starts testing it.
A sprint holds a mini-waterfall of activities

- **Requirements**

  - S3 (21)
  - S6 (2)
  - S7 (13)
  - S8 (5)
  - S9 (1)
  - S10 (3)
  - S11 (5)

- **Design**

  - S1 (5)
  - S2 (13)
  - S4 (3)
  - S5 (3)

- **Product Backlog**

- **Sprint Backlog**

- **Implementation**

- **Maintenance**

- **Verification**

- **Ready for Test**

- **In Test**

- **Done**
When members work on a Story they should be assigned to the card.
As solution tasks are completed, the person doing the work checks off the task as done.
Check-off *Definition of Done* items as the work gets completed.
The tester checks off **Acceptance Criteria** as they validate the behavior of the system.
On any given day, the sprint board tells us the status of all sprint work.

<table>
<thead>
<tr>
<th>Product Backlog</th>
<th>Sprint Backlog</th>
<th>In Dev</th>
<th>Ready for Test</th>
<th>In Test</th>
<th>Done</th>
</tr>
</thead>
<tbody>
<tr>
<td>S3 (21)</td>
<td>S4 (3)</td>
<td>S2 (13)</td>
<td>S5 (3)</td>
<td>S1 (5)</td>
<td></td>
</tr>
<tr>
<td>S6 (2)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S7 (13)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S8 (5)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S9 (1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S10 (3)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S11 (5)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
At the end of a sprint, unfinished work gets put back onto the Product Backlog.
And the process starts all over again at the start of the next sprint.

The team has some flexibility in what they are willing to commit to the product owner for a sprint.
But how do you really know how many story points the team can accomplish in a sprint?

- Velocity is measured from the average of the last three sprints.
  - *Only use the completed stories.*
  - *This rolling average will change over time.*

- The average velocity is then used to cap the number of story points for the next sprint.

- Example:
  - *Sprint 7:* 45 story points committed; 42 completed.
  - *Sprint 8:* 40 committed; 50 completed
  - *Sprint 9:* 48 committed; 47 completed
  - *Velocity* = \((42 + 50 + 47) / 3 \approx 46.3333\)
  - *Sprint 10 will be capped at 46 story points.*
Velocity is specific to one team working on one project.

- This assumes that sprint length and team membership remain consistent.
  - *If either of these two change, then velocity measurement must start over with a new running average.*

- Velocity is only measured for a single project, single team.
  - *Story points are level of effort estimates*
  - *Story points are determined by the team, for the team*
  - *Thus you cannot compare velocity's between teams*

- Management cannot set a team's velocity.
Velocity is not the same as the team's overall capacity.

- Velocity is only a measure of effort for working on user stories.
  - *It does not include company meetings, email communication, small "outside" tasks*
  - *For class, it does not include most before- and after-class activities*

- What about these issues?
  - *Holidays or vacations*
  - *Members given large, outside tasks*

- Normally these issues can be ignored being smoothed out by the averaging process.

- In extreme cases, the team can make adjustments to the calculated velocity usually by lowering it.