Scrum and best practices

18/11/2018 • 12 minutos para ler • Colaboradores

Neste artigo

Azure Boards | Azure DevOps Server 2019 | TFS 2018 | TFS 2017 | TFS 2015 | TFS 2013

Sprint planning meetings

Much of sprint planning involves a negotiation between the product owner and the team to determine the focus and work to tackle in the upcoming sprint. It's useful to time-box the planning meeting, restricting it to 4 hours or less.

In the first part of the meeting, your product owner meets with your team to discuss the user stories that might be included in the sprint. Your product owner will share information and answer any questions that your team has about those stories. This conversation might reveal details such as data sources, user interface layout, response time expectations, and considerations for security and usability. Your team should capture these details within the backlog items form. During this part of the meeting, your team learns what it must build.

As you plan your sprints, you may discover additional requirements that you should capture and add to your backlog. Before your sprint planning meeting, you'll want to refine your backlog to make sure that it is well defined and in priority order.

Also, setting a sprint goal as part of your planning efforts can help the team stay focused on what's most important for each sprint.

After you've planned your sprint, you may want to share the plan with key stakeholders.

You can learn more from these resources:

- What is Scrum?
- Sprint planning white paper
- The Scrum Guide
- Build and manage the product backlog white paper

Set sprint goals

Scrum teams use sprint goals to focus their sprint activities. They often set this goal during their sprint planning meeting. The goal summarizes what the team wants to accomplish by the end of the sprint. By explicitly stating the goal, you create shared understanding within the team of the core objective. The sprint goal can also help guide the team when conflicts arise around priorities.

Tips from the trenches: Define sprint goals

The sprint goal defines what the product owner and the team consider as the ultimate target to accomplish that sprint. It's not a random selection of backlog items that don't really have a relationship, but a short piece of text that captures what the team should try to accomplish. Normally the product owner comes up with the sprint goal before selecting a number of items for the next sprint. The items for that sprint should all fit that common goal.

Sprint goals can be feature oriented, but might also have a large process component such as deployment automation or test automation.

For example:

- This sprint we will focus on a very simple user story and we will use it to prove that the proposed solution will work.
- This sprint will revolve around implementing the security features that will properly secure the administration section of the website.
- This sprint will be about integrating the most important payment gateways so that we can start collecting money.

Setting the sprint goals helps the team to stay focused. It will make it easier to define priority of tasks within a sprint and it will probably help limit the number of stakeholders and end-users that are involved.

During the sprint review the most important question you should ask yourself is whether you managed to achieve the sprint goal. How many stories you actually completed comes second. If the goal is accomplished, the sprint succeeds, even if not all stories were finished.

Contributed by Jesse Houwing, Visual Studio devops Ranger and a senior consultant working for Avanade Netherlands.

Tips for successful triage meetings

Fixing bugs represents a trade-off with regards to other work. Use your triage meeting to determine how important fixing each bug is against other priorities related to meeting the project scope, budget, and schedule.

- Establish the team's criteria for evaluating which bugs to fix and how to assign priority and severity. Bugs associated with features of significant value (or significant opportunity cost of delay), or other project risks, should be assigned higher priority and severity. Store your triage criteria with other team documents and update as needed.
- Use your triage criteria to determine which bugs to fix and how to set their State, Priority, Severity, and other fields.
- Adjust your triage criteria based on where you are in your development cycle. Early on, you may decide to fix most of the bugs that you triage. However, later in the cycle, you may raise the triage criteria (or bug bar) to reduce the number of bugs that you need to fix.
- Once you've triaged and prioritized a bug, assign it to a developer for further investigation and to determine how to implement a fix.

Manage your technical debt

Consider managing your bug bar and technical debt as part of your team's overall set of continuous improvement activities. You may find these additional resources of interest:

- Good and Bad Technical Debt (and how TDD helps) by Henrik Kniberg
- Managing Technical Debt posted by Sven Johann & Eberhard Wolff

Tips from the trenches: Agile Bug Management: Not an Oxymoron

by Gregg Boer, Principal Program Manager, Visual Studio Cloud Services at Microsoft

Every Sprint, Address any Known Bug Debt

Every sprint, the team looks at any bugs remaining in the bug backlog and allocates capacity to get that known set of bugs down to zero, or near-zero. Whether this is one day, one week or the entire sprint, they fix the bugs first. Bugs found later, within the sprint, are not considered part of that initial commitment. Unless they're very high priority, they're put on the bug backlog for the next sprint.

Many teams work in a commitment-based organization, where management places a high value on a team's ability to meet their commitments. Doing capacity planning against a known set of bugs makes sprint planning more deterministic, increasing their chance to meet commitments. Any new bugs discovered during the sprint are not a part of the initial commitment, and will be tackled next sprint.>

Managing Bug Debt across an Enterprise

An organization transitioning to a culture where debt is continually eliminated likely is dealing with the following question: How do you get teams to reduce their bug count without telling them exactly what to do? Leadership wants the team to change, yet gives the team autonomy to determine how they change. One option is to use a bug cap.

For example, consider a bug cap of three bugs per engineer. This means a team of 10 people should not have more than 30 bugs in its bug backlog. If the team is over its cap, it's expected to stop work on new features and get under the bug cap. A team is expected to be under its cap at all times, but the team decides how it wants to do that. The bug cap ensures that bug debt is never carried for too long, and the team can learn from the mistakes that causes the bugs to be injected in the first place.

Remember that the bug cap represents the bugs in the bug backlog. It does not include bugs found and fixed within the sprint in which a feature is developed. Those bugs are considered undone work, not debt.

While bugs contribute to technical debt, they may not represent all debt.

Poor software design, poorly written code, or short-term fixes in place of best, well-designed solutions can all contribute to technical debt. Technical debt reflects extra development work that arises from all these problems.

You need to track work to address technical debt as PBIs, user stories, or bugs. To track a team's progress in incurring and addressing technical debt, you'll want to consider how to categorize the work item and the details you want to track. You can add tags to any work item to group it into a category of your choosing.

Role of the Scrum Master

Scrum Masters help build and maintain healthy teams by employing Scrum processes. They guide, coach, teach, and assist Scrum teams in the proper employment of Scrum methods. Scrum Masters also act as change agents to help teams overcome impediments and to drive the team toward significant productivity increases.

Core responsibilities of Scrum Masters include:

- Support the team to adopt and follow Scrum processes. For example, you should not let the daily Scrum meeting become an open discussion that lasts 45 minutes.
- Guard against the product owner or team members from adding work after the sprint begins.
- Clear blocking issues that prevent the team from making forward progress. This might require you to perform small tasks, such as approving a purchase order for a new build computer or resolving a conflict within your team.
- Help the team work to resolve conflicts and issues that arise and learn from the process.
- Ask questions that reveal hidden issues and confirm that what people are communicating is well understood by the entire team.
- Identify and safeguard the team from potential issues becoming major issues. Just as it's cheaper to fix a bug soon after it's discovered, it's also easier and less disruptive to fix a team issue when it's small and manageable.
- Prevent the team from presenting incomplete user stories during a sprint review meeting.
- Gather, analyze, and present data to business stakeholders in a way that demonstrates how the team is improving and growing. For example, if your team has significantly increased the amount of value that it has delivered while generating fewer bugs, make the value visible through regular communications to business stakeholders.

Good Scrum Masters possess or develop excellent communication, negotiation, and conflict resolution skills. They actively listen to not only the words that people say and write but also how they deliver their messages (their body language, facial expressions, vocal pace, and other nonverbal communication).

Just as it's cheaper to fix a bug soon after it's discovered, it's also easier and less disruptive to fix a team issue when it's small and manageable before it grows into a major issue.

Daily Scrum meetings

Daily Scrum meetings help keep a team focused on what it needs to do the next day to maximize the team's ability to meet its sprint commitments. Your Scrum Master should enforce the structure of the meeting and ensure that it starts on time and finishes in 15 minutes or less.

Three aspects of successful Scrum meetings are:

- Everyone stands up (this helps to keep the meetings focused and short)
- They start and end on time and occur at the same time in the same location each day
- Everyone participates, each team member answers the three Scrum questions:
  - *What have I accomplished since the most recent Scrum?*
  - *What will I accomplish before the next Scrum?*
  - *What blocking issues or impediments might affect my work?*

Team members should strive to answer their questions quickly and concisely. For example:

> *"Yesterday, I updated the class to reflect the new data element that we pull from the database, and I got it to appear in the interface. This task is complete. Today, I will ensure that the new data element is correctly calculating with the stored procedure and the other data elements in the table. I believe I will accomplish this task today. I will need someone to review my calculations. I have no impediments or blocking issues."*

This response conveys what was accomplished, what will be accomplished, and that the team member would like some help looking at the code.

Contrast with this next example:

> *"Yesterday, I worked on the class, and it works. Today, I will work on the interface. No blocking issues."*

Here, the team member doesn't provide enough detail about what class they worked on nor which interface components they'll complete. In fact, the word accomplished never came up.

It's important that no one interrupts during report outs. Each person must have sufficient time to answer the three questions.

More in-depth and follow-up discussions should take place after the meeting, as people return to their desks or, if a significant amount of conversation is necessary, in a follow-up meeting.

Many teams delay discussions by using the "virtual parking lot" method. As topics come up that a team member believes warrants further discussion, they can quietly walk to a whiteboard or flipchart and list the topic in the parking lot. At the end of the meeting, the team determines how they'll handle the listed items.

Sprint review meetings

Conduct your sprint review meetings on the last day of the sprint. Your team demonstrates each product backlog item that it completed in the sprint. The product owner, customers, and stakeholders accept the user stories that meet their expectations and identify any new requirements. Customers often understand their additional needs more fully after seeing the demonstrations and may identify changes that they want to see.

Based on this meeting, some user stories will be accepted as complete. Incomplete user stories will remain in the product backlog, and new user stories will be added to the backlog. Both sets of stories will be ranked and either estimated or re-estimated in the next sprint planning meeting.

After this meeting and the retrospective meeting, your team will plan the next sprint. Because business needs change quickly, you can take advantage of this meeting with your product owner, customers, and stakeholders to review the priorities of the product backlog again.

Sprint retrospective meetings

Retrospectives, when conducted well and at regular intervals, support continuous improvement.

The sprint retrospective meeting typically occurs on the last day of the sprint, after the sprint review meeting. In this meeting, your team explores its execution of Scrum and what might need tweaking.

Based on discussions, your team might decide to change one or more processes to improve its own effectiveness, productivity, quality, and satisfaction. This meeting and the resulting improvements are critical to the agile principle of self-organization.

Look to address these areas during your team sprint retrospectives:

- Issues that affected your team's general effectiveness, productivity, and quality.
- Elements that impacted your team's overall satisfaction and project flow.
- What happened to cause incomplete backlog items? What actions will the team take to prevent these issues in the future?

  For example, consider a team that had several tasks that only one individual on the team could perform. The isolated expertise created a critical path that threatened the sprint's success. The individual team member put in extra hours while other team members were frustrated that they could not do more to help. Going forward, the team decided to practice eXtreme Programming to help correct this problem over time.

As a team, work to determine whether to adapt one or more processes to minimize the occurrence of problems during the sprint.

In some cases, your team may need to do some work to implement an improvement. For example, a team that found themselves negatively impacted by too many failed builds decided to implement continuous integration. Because they didn't want to disrupt process, they allocated a few hours to set up a trial build before turning it on in their production build. To represent this work, they created a spike and prioritized that work against the rest of the product backlog.

Related articles

- What is Scrum?
- Agile Retrospectives: Making Good Teams Great