| [Print](javascript:window.print()) | [Close](javascript:parent.window.close()) |

**Course Transcript**

Software Practices (SCRUM): SCRUM Meetings

**Scrum Meetings**

| [1. Course Introduction](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a02_it_enus/output/html/course_transcript.html#t2) |

| [2. The Scrum Meetings](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a02_it_enus/output/html/course_transcript.html#t6) |

| [3. The Sprint Planning Meeting](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a02_it_enus/output/html/course_transcript.html#t10) |

| [4. The Scrum Daily Standup Meeting](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a02_it_enus/output/html/course_transcript.html#t14) |

| [5. The Sprint Retrospective Meeting](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a02_it_enus/output/html/course_transcript.html#t18) |

| [6. Insights from the Sprint Retrospective Meeting](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a02_it_enus/output/html/course_transcript.html#t22) |

| [7. The Backlog Grooming Meeting](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a02_it_enus/output/html/course_transcript.html#t26) |

| [8. The Sprint Review Meeting](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a02_it_enus/output/html/course_transcript.html#t30) |

**Team Composition and Scale**

| [1. Scaling Scrum](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a02_it_enus/output/html/course_transcript.html#t35) |

| [2. The Definition of Done (DoD)](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a02_it_enus/output/html/course_transcript.html#t39) |

| [3. Why Should User Stories Be Small?](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a02_it_enus/output/html/course_transcript.html#t43) |

| [4. The Customer User Story](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a02_it_enus/output/html/course_transcript.html#t47) |

| [5. Sizing User Stories](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a02_it_enus/output/html/course_transcript.html#t51) |

| [6. Why Do Teams Break Down User Stories into Tasks](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a02_it_enus/output/html/course_transcript.html#t55) |

| [7. Encouraging Self-Organizing Team Behavior](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a02_it_enus/output/html/course_transcript.html#t59) |

**User Stories**

| [1. User Story Overview](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a02_it_enus/output/html/course_transcript.html#t64) |

| [2. Epics and User Stories](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a02_it_enus/output/html/course_transcript.html#t68) |

| [3. The Story Board](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a02_it_enus/output/html/course_transcript.html#t72) |

| [4. The User Story Lifecycle](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a02_it_enus/output/html/course_transcript.html#t76) |

| [5. User Story Breakdown](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a02_it_enus/output/html/course_transcript.html#t80) |

| [6. Heuristics for Splitting User Stories](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a02_it_enus/output/html/course_transcript.html#t84) |

| [7. Business and Data Patterns for Splitting User Stories](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a02_it_enus/output/html/course_transcript.html#t88) |

| [8. Patterns for Splitting User Stories](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a02_it_enus/output/html/course_transcript.html#t92) |

| [9. User Story Tasks](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a02_it_enus/output/html/course_transcript.html#t96) |

| [10. SMART Tasks](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a02_it_enus/output/html/course_transcript.html#t100) |

| [11. INVEST User Story Guidelines](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a02_it_enus/output/html/course_transcript.html#t104) |

**Technical Debt**

| [1. Accumulation of Technical Debt](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a02_it_enus/output/html/course_transcript.html#t109) |

| [2. Identifying and Reducing Technical Debt](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a02_it_enus/output/html/course_transcript.html#t113) |

**Managing Velocity**

| [1. Scrum and Deferring Detailed Specifications](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a02_it_enus/output/html/course_transcript.html#t118) |

| [2. Team Velocity](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a02_it_enus/output/html/course_transcript.html#t122) |

| [3. Managing the Sprint](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a02_it_enus/output/html/course_transcript.html#t126) |

| [4. Limiting Work in Progress](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a02_it_enus/output/html/course_transcript.html#t130) |

Course Introduction

Learning Objective

*After completing this topic, you should be able to*

* *start the course*

**1. Introduction to the course**

A large part of the Scrum process revolves around the Scrum meeting, and another large part of the Scrum process is managing user stories. User stories are short descriptions on non-technical language, of how a system is expected to behave. In this course, Brigitte Birze will cover the Scrum meeting and user stories in depth. She will also explore key concepts, such as velocity and technical debt. I hope you enjoy the course.

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a02_it_enus/output/html/course_transcript.html#top)

The Scrum Meetings

Learning Objective

*After completing this topic, you should be able to*

* *describe the four Scrum meetings, when they occur, who attends, and the discussions that take place in each meeting*

**1. Types of Scrum meetings**

There are four meetings or ceremonies defined in the Scrum framework. Each meeting happens at a specific time in each sprint. And each has a set purpose and defines what participants are required, and what artifact is produced by the meeting. The meetings give the sprint the same structure. This structure, with the iterative nature of the sprints, creates a rhythm for the entire project. As a Scrum project consists of multiple sprints, the Scrum meetings will reoccur in each sprint at specific times in the sprint. First the sprint planning meeting is held before the sprint begins, to define what will be delivered in the sprint. The product owner, Scrum master, and Scrum team are all required in the sprint planning meeting. In the sprint planning meeting, the top priority items are added to the sprint backlog. The team breaks each product item into smaller tasks and estimates the work for each task. Once the team has filled its capacity for the sprint, they stop taking new items, and commit to completing the items in the sprint backlog in the next sprint.   
*Heading: Scrum Meetings.  
  
A diagrammatical representation of the structure of a sprint timeline is displayed. The timeline begins with the product backlog, which is followed by a sprint backlog. The sprint meeting is displayed toward the end of the timeline and form parts of the sprint planning stage of the sprint timeline.*   
  
Once the sprint is started, a Scrum standup meeting is held daily for the Scrum master and Scrum team to check in. The daily Scrum meeting is short, fifteen minutes or less. And each member of the team tells the rest of the team what they did since the last Scrum meeting, what they will do before the next Scrum meeting, and what is blocking them. The Scrum meeting is not to solve issues, it is only to make them visible. Team members may agree to meet offline after the Scrum meeting to find a solution. Only the Scrum master and the Scrum team are required at the daily Scrum meeting, though the product owner is welcome to attend. The sprint review is held at the end of the sprint, to demonstrate the product increment to the customers and stakeholders. The product increment is working software in a potentially shippable form. And it should contain all the new features the team committed to deliver, in the sprint planning meeting. The product owner, Scrum master, and Scrum team are all required at the sprint review. And we want as many customers and stakeholders to attend as possible. So they can give valuable feedback on how the product is developing.   
*Heading: Scrum Meetings.  
  
The daily scrum is attached to the sprint meeting and a product increment graphic is displayed after the latter half of the sprint timeline, which is part of the sprint review phase of the timeline.*   
  
The sprint retrospective is held after the sprint, for the team to reflect and review what went right and wrong in this last sprint and initiate process improvement. The product owner, Scrum master, and Scrum team are all required in the Sprint Retrospective meeting. At the end of the sprint retrospective meeting, a few changes are selected by the team to implement in the next sprint to increase the efficiency and effectiveness of the team. Though not an official Scrum meeting, the backlog grooming meeting is often held during the sprint to allow the team to review the top priority items in the product backlog. In the product backlog grooming meeting, the top items in the product backlog are assigned a relative story point value by the team. Product items that are too big to fit into a single sprint are split into multiple user stories. The team and product owner work together to ensure that acceptance criteria is assigned. And the team can ask the product owner questions about each story so it is well understood. Ongoing product backlog grooming meetings ensures smooth sprint planning meeting as each high-priority item has already been reviewed, sized, and ambiguities have been resolved.   
*Heading: Scrum Meetings.  
  
At the very end of the sprint timeline and beyond, is the sprint retrospective phase, which contains a column chart representing process improvement. Backlog grooming is then added to the beginning of the timeline.*

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a02_it_enus/output/html/course_transcript.html#top)

The Sprint Planning Meeting

Learning Objective

*After completing this topic, you should be able to*

* *describe how the Sprint Goal and Sprint Backlog are created in the Sprint Planning meeting in Scrum software practices*

**1. Sprint planning meetings**

The product owner should show up to the sprint planning meeting with a properly groomed backlog, with the highest priority PBIs, or product backlog Items, at the top of the product backlog. And the high-priority story should be sprint ready, having already been groomed by the team in the product grooming meeting, where they have split the stories. So they should fit in a single sprint, agreed on acceptance criteria, and assigned a relative level of effort to each story. The product owner should also be prepared to answer questions about the highest priority items in the backlog. Now there are two parts to the sprint planning meeting; the first part works with goals, and the second part works with planning. The goal portion of the meeting is dedicated to giving the sprint a theme, in determining what will be worked in the sprint to satisfy the theme. The product owner begins by describing the product vision, and what the customers and stakeholders would like to see implemented in the upcoming sprint. The product owner will already have prioritized the backlog, to reflect the stakeholder's preferences. The product owner describes these top items on the product backlog, including how they relate to the stakeholder's top priorities.   
*An instance of team collaboration software is displayed. The interface contains four tabs: My Home, Backlog, Plan, and Track. Backlog is selected and displays stories ranked by priority, with individual IDs and plan estimates.*   
  
The team members can ask questions about the PBIs to get clarifications on their scope, and on how the PBIs relate to the goals the product owner has outlined. The product owner and team collaborate to decide on the goal statement for the upcoming sprint, and the team selects items to work that relate to the goal. In this case, the highest priorities items are display plant information on the web sites, save plant results displayed, saved detailed plant information displayed. Well we kind of have a goal there already. You could say the theme is to display plant information on the web site and save the results. Now this goal is important because it could be used to quickly convey what the team will be working on in the sprint to the stakeholders. And the sprint goal is also used during the sprint review meeting to assess the success of the sprint. Then the team picks the highest priority items from the backlog to pull into the sprint.   
*The presenter highlights rows where the user stories, Display Plant Information on web site, is ranked at 1 with a plan estimate of 3.00. Save Plant Results List Displayed on web site is ranked at 2 with a plan estimate of 2.00. Save Detailed plant Information Displayed on web site is ranked at 3 with a plan estimate of 1.00.*   
  
For teams with a stable velocity, using story points to estimate their capacity is a best practice because story points emphasize effort and complexity instead of duration, and are directly related to the team's velocity. So here, the team might say that they want to take in this one, say that their velocity is seven, they could take in the top three user stories that relate to the theme, and that gets them to a capacity of six. Well they have one more point; they can't really bring the next one and that's eight, that's actually too big. That's going to need to split in the next product grooming meeting. And the others, three is too large. Well they could bring in the SPIKE, that's the next highest priority item. It's going to be investigating portal solutions because we're going to have a portal implementation coming up. So the team could decide to take that into the next meeting to fill out to reach their capacity. Or they could come further down and here is a few defects, they could elect to take some defects in.   
*The presenter highlights rows where the story, User Registration, is ranked at 4 with a plan estimate of 8.00 and SPIKE: Investigate Customizable portal solutions is ranked at 6 with a plan estimate of 0.00.  
  
The presenter scrolls to where the story is Error found in US: Plants removed from results list are removed from database, which is ranked at 9 and has a high attention priority.*   
  
It's important to note that while the product owner has prioritized this list to show what they want have worked on, the team has sole authority to decide how much work they can commit to in a single iteration. When the team feels they have met their capacity, they stop accepting new PBIs into the sprint. Now once the PBIs to work have been chosen, they are then moved from the product backlog to the sprint backlog. So here's the product backlog, here's our sprint; so we'll move the top three stories over, and then we decide we're also going to work the SPIKE. Then the team says, "We're done, that's all we can handle." At this point, the team commits collectively as a team, to complete the items in the sprint backlog for the next sprint. Now the second part of the meeting is where the team works together to split the committed user stories into tasks that describe how the user stories will be implemented.   
*The presenter selects the Plan tab. The Plan tabbed page displays an Iteration planning Board, which has sections for Backlogs, sprint 4, sprint 5, and sprint 6. The presenter moves stories from the backlog section to the sprint 6 section.*   
  
If we look here in our previous sprint, we can see some of the tasks that they split this user stories up into. The tasks list that the team creates for each user story is a simple to-do list that describes how the team will satisfy the stories acceptance criteria. The team estimates each task with a duration on how long they think it will take to complete the task. The product owner does not have to be present for this portion of the sprint planning meeting, but they must be readily available in case the team has more questions. At the end of the sprint planning meeting, the team has a plan for the sprint and a list of items that will be completed during the sprint. This is reflected in the two artifacts created during the sprint planning meeting – the sprint goal and the sprint backlog.   
*The presenter selects the Tasks icon on one of the Sprints and a Task page is displayed alongside the sprint. It contains columns for ID, Name, Owner, State, Estimate (H), and To Do (H).*

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a02_it_enus/output/html/course_transcript.html#top)

The Scrum Daily Standup Meeting

Learning Objective

*After completing this topic, you should be able to*

* *describe how the Daily Standup meetings are structured with the Scrum Master and Scrum Team in Scrum software practices*

**1. The daily standup meeting**

The Scrum meeting is held daily and time-boxed; normally kept to no more than 15 minutes. It is often the practice to hold the meeting with everyone standing up to keep the meeting short and focused. Noting this practice, the Scrum meeting may also be referred to as the daily standup or just the standup. The Scrum meeting is used to discuss the progress of the work committed to during the sprint. So the required participants are the ones holding committed roles – the Scrum master and the Scrum team. The product Owner is not required at the Scrum meeting, but may attend to observe and provide clarification. However, they are not allowed to speak unless asked a question directly by the Scrum master or the team. The Scrum master facilitates the meeting, which is held at the same time and place each day – normally in the morning. The Scrum master should ensure the daily standup begins and ends on time, even if a team member is delayed. In the daily Scrum, the Scrum master asks each team member in turn, "What did you do yesterday? What will you do today, and are there any impediments in your way?"   
*Heading: The Scrum Meeting.  
  
A diagrammatical representation of the Scrum meeting is displayed. It includes the Team, the Scrum Master, and the Product Owner.*   
  
The team gains tremendous benefit from the daily Scrum meeting. During the meeting, everyone on the team receives information on what work has been completed by other team members and what work remains, but is in progress. They also get a heads-up on issues before they become a block to multiple team members. The three questions every team member is asked should be answered in a concise manner as the daily Scrum is intended to share information. It is not intended to discuss, resolve, or work problems. Instead, several team members may decide to take an issue offline to work, usually directly after the meeting ends. In this way, the daily Scrum facilitates teamwork and collaboration between the team. Also, when a team member says what they will work on, they are making a commitment to the entire team, especially if they commit to finishing a task. Team members use the commitments of other members stated in the daily Scrum to plan their own work. And the team as a whole comes to an understanding of the significance of the commitments, they make to each other and to the team.   
  
The Scrum master not only facilitates the meeting, but also uses the information since one of the main duties of the Scrum master is to remove impediments so the team can continue to make progress, though a team member can and should bring in an impediment to the Scrum master's attention as soon as it's discovered. The daily Scrum is a means of ensuring these issues do not remain uncovered for several days, giving the Scrum master time to work the issue before it becomes problematic. If an issue is something the Scrum master cannot work, like a technical decision, they can assign team members to work the issue after the meeting. Stakeholders often want to attend the daily Scrum, especially when they are new to Scrum and feel the need to be informed on the daily progress of the team, and the details of the issues it is experiencing. However, stakeholders need to be aware of the role they are playing, as it is important in how the Scrum meeting is conducted.   
  
In the daily Scrum, the Scrum master and Scrum team are considered committed roles or pigs. And the stakeholders are involved roles or chickens. This refers to a breakfast of eggs and ham where the chicken is involved but the pig is committed. The Scrum meeting is to discuss the work the pigs have committed to during the current sprint, in a short focused manner. The Scrum meeting is held standing up and time-boxed to 15 minutes, ensuring the discussions are brief and relevant. To ensure this happens, the Scrum master is responsible for enforcing the rule that while chickens are welcome to attend and observe the daily Scrum meeting to stay informed, only pigs may talk. Chickens must remain silent and are not allowed to ask questions or distract team members. This prevents the Scrum meeting from going off topic with questions or long discussions.

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a02_it_enus/output/html/course_transcript.html#top)

The Sprint Retrospective Meeting

Learning Objective

*After completing this topic, you should be able to*

* *describe how the Sprint Retrospective meeting assists the Scrum Master, Scrum Team, and Product Owner to adapt their practices and behaviors toward continual process improvement*

**1. Outlining sprint retrospective meetings**

The sprint retrospective is held after the sprint, for the Scrum team to reflect and review on the last sprint to determine what went right and wrong, and initiate process improvement. The meeting is facilitated by the Scrum master and attended by the team and product owner. However, managers and stakeholders should not attend, as it could cause teams to withhold honest observations, and have a chilling effect as the team discusses how to improve. As facilitator, the Scrum master must strive to generate an environment of trust and support, enabling all participants to feel safe in expressing what they think. Open, honest communication is key, with no judgment. It's a good idea for the Scrum master to open the meeting by thanking each participant for coming and reviewing the teams' agreement to engage in open and honest communication, delivered with respect for everyone involved.   
*Heading: The Sprint Retrospective Meeting.  
  
A diagrammatical representation of the sprint retrospective – attended by the Scrum team, which consists of the product owner, Scrum master, and team – leading to overall process improvement is displayed.*  
  
During the sprint retrospective the participants as a group will discover insights and solutions, as they come to a deeper understanding of what occurred in the last sprint. At the end of the sprint retrospective meeting a few changes are selected by the team, to implement in the next sprint to increase the efficiency and effectiveness of the team. To do this, what went right and wrong must be honestly and courageously brought out. Each team member should offer what they thought worked well and should be continued. And concerns or practices that caused issues should be changed. In new teams, the Scrum master may need to jump-start each retrospective until the team develops trust. This could be done by collecting each member's three top observations before the sprint and organizing them into a starting list. The list is just to get the discussion started, though. It's up to the team to determine their top items to work and to come up with action items to implement in order to improve in the next sprint.   
  
Once the team has a list of events from the last sprint that were positive and negative, the team discusses the top items to gain insights into why the events happened. Getting a clear understanding of why events happened may show that multiple events had the same root cause. The team then picks which root causes they will further discuss to come up with ways to improve. However, all issues cannot be tackled at the same time or nothing will get done. The team should pick the top one or two items that will give the most bang for the buck if improved. When the root cause of an issue is understood action items can be created, to document the changes that must happen to eliminate the cause of the issue. The action items are the deliverable of the retrospective meeting. Without them the team is just engaging and griping and congratulating without adding value.   
  
The action items can be directed at the working environment, practices, or relationships, such as the way that the team and product owner collaborate. Each action item holds an experiment – something to try to see if things get better. Each action item needs to include something the team can execute, which is measurable and demonstratable. Do not spend time and energy on things that are not possible or cannot be done. Once the team collectively agrees on one to three action items to implement in the next sprint, they must all commit to executing them. The chosen action items should be recorded and posted in the public place, where they can be reviewed as part of the Scrum meeting, to ensure the team is implementing the process improvement. And don't forget to record what the team did correctly as well. This keeps the team focused on what they are doing right, reinforcing the behavior, practices and processes in future sprints. The retrospective meeting is not a gripe session or congratulatory session. While these activities are part of the meeting, they are not the reason for the meeting and should not be the focus. The focus is on the team and how it can become more efficient and agile. This will involve the team, Scrum master, and product owner adapting their practices and behaviors toward continual process improvement.

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a02_it_enus/output/html/course_transcript.html#top)

Insights from the Sprint Retrospective Meeting

Learning Objective

*After completing this topic, you should be able to*

* *describe how the Sprint Retrospective Meeting in Scrum can reveal insights into the last Sprint and identify areas that have room for improvement*

**1. Analyzing sprint retrospective meetings**

As facilitator of the sprint retrospective meeting, the Scrum master needs to show up prepared with a synopsis of the last sprint's events. The sprint and project artifacts can shed light on what went right and wrong, and why. This is the sprint tracking board; we're looking at Sprint 5. The team did a great job of finishing the work in the sprint backlog. Everything was accepted, they got all the story points for their velocity. But the burn down chart shows work was added after the sprint began. You can see the second day we only had 134 hours but it's way up here like 275, so what happened? Taking a closer look, we can see that New Task X was added with 18 hours of work. So the team must have worked really hard to get this new work done in the sprint. This is an example of something that went right – new work was added but the team worked together to complete it. And an example of something that went wrong – why was Task X added? What didn't the team know in the sprint planning meeting when the task breakdown occurred?   
*An instance of team collaboration software is displayed. The interface contains four tabs: My Home, Backlog, Plan, and Track.   
  
The Track tabbed page is open and displays an Iteration Tracking Board for Sprint 5. It contains the columns: Defined, In-Progress, Completed – which are currently blank – and an Accepted column, which has a few stories listed.  
  
The Tracking page includes an Iteration Burndown chart and sections for the Iteration Summary and Recent Activity.  
  
The Iteration Burndown chart contains a clustered column chart with markers. The X-axis contains dates ranging from 07-31 to 08-07 from left to right. The right of the Y-axis displays the numbers 0 to 7 in increments of 1 and the left of the Y-axis displays numbers 0 to 300 in increments of 50. Markers are placed at 275 of the Y-axis on the date 07-31 and the markers descend all the way to zero spanning the days on the X-axis ending at 08-07.  
  
The presenter clicks the Tasks icon of a story in the Accepted column and the task page is displayed. It contains columns for ID, Name, Owner, State, Estimate (H), and To Do (H) and shows that a new task was added and completed in an 18 hour estimate.*   
  
Let's look over at **Sprint 4**. Here we see the team did not finish the work in the sprint backlog. US2 was only 71% complete. Looking closer, we see the hold up was this development task. The development task only had two hours of work left in it, however since it was not complete QA here could not finish the testing and our tech writer could not finish writing the documentation or taking the screenshots. Maybe the root issue was the other team members not stepping up to help with tasks out of their normal area. The architect or GUI Developer could have helped our web developer up here, by either helping with development writing unit test, reviewing code, or assisting in whatever was keeping the task from being complete. Then the entire team could have jumped on the testing and getting it done and taken the screen shots for our tech writer.   
*The presenter navigates to sprint 4 on the Track tabbed page. One story, US2, is listed in the In-Progress column and two stories are listed in the Accepted column. The Defined and Competed columns are empty.  
  
The presenter clicks the Tasks icon of US2 and the Task page is displayed. The presenter scrolls through the tasks and finds the source of the delay - DEV. DEV had the task of accepting a request to delete items from DB search results and deleting it from results stored in user session, and to resend updated results to the web.  
  
The state for DEV and DOC shows In-Progress, while the other team members, GUI and QA, shows a completed state.*   
  
As the entire team succeeds or fails, all team members need to work any task possible in order to move user stories to done. So let's look at one more sprint – let's look at back at Sprint 1, our very first sprint. This also shows that the team did not complete all the user stories. Here the burn down chart shows the team did not get all their work added until the third day of the sprint. In this case, the sprint planning meeting could have been delayed or the team could have not gotten around to breaking down the user stories into tasks until the third day or maybe the product owner added new stories after the sprint backlog was frozen. Since this was the team's first sprint they were just getting used to the process. However, these type of logistic issues cause the team to start the sprint late and not complete the work. Other artifacts that shed light on the events of the last sprint are the task list itself as it shows work in progress.   
*The presenter navigates to Sprint 1 on the Track tabbed page. One story, US36, is listed in the In-Progress column and three stories are listed in the Accepted column. The Defined and Competed columns are empty.*   
  
Did one team member take on too many tasks at the same time and lose focus maybe causing other team members to wait for task to be completed? Did the team have too many user stories in progress at the same time instead of swarming a single user story to get it complete before moving on to the next user story? Defect trend charts could show a sharp increase in defects meaning the code quality needs improvement, maybe with instant tuning unit test or code reviews or the defect count could be going up because they are low on the product backlog and not getting worked. This would mean the team is working on a code base with an increasing amount of technical debt which would slow the team down. When preparing for a sprint retrospective, the Scrum artifacts from the last sprint are a wonderful place to start to gather events that went right and wrong and to gain insights into why they occurred in the first place.

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a02_it_enus/output/html/course_transcript.html#top)

The Backlog Grooming Meeting

Learning Objective

*After completing this topic, you should be able to*

* *describe how the Product Backlog Grooming Meeting in Scrum allows the team to review the top level user stories to determine whether they are ready to move into the next Sprint*

**1. Describing Backlog Grooming meetings**

So here we are in our backlog grooming meeting – the Scrum master's facilitating, the product owner and the Scrum team are both required attendants. And this is our product backlog. The bottom here has the low priority items – not well understood, very large, lots of times not broken down. What we're going to focus on is here are these top level stories, the ones that are the highest priority that will be going into the next sprints. And we want to ensure that they're well understood, they're good acceptance criteria, they have estimates, and that we feel comfortable that they'll all fit in a single sprint. So let's look at this highest top level priority item here, Website for Online Users. It's part of the User Experience EPIC. And if we look at it, we see that it's well defined, we have a good user story, they want to be able to access plant information on an online web site. The acceptance criteria – it looks like this is an infrastructure user story. We're going to have to create a web site, luckily our company has a web site in-house that is, has its own company web site and so we're just going to have to work with IT to get an area there to host our web site. We're going to have to register a domain name, make it accessible and then the GUI has some work because they're just starting out. They need to have their overall look and feel, get some templates set up and have a minimally designed home page so that the next time there's a sprint review the customer can see it and give feedback. And of course, it needs to be fully tested.   
*An instance of team collaboration software is displayed. The interface contains four tabs: My Home, Backlog, Plan, and Track. Backlog is selected. All items are ranked by priority with individual IDs and plan estimates. Column headings include: Rank, ID, Name, Plan Estimate, Priority, Owner, and Parent.  
  
The presenter highlights the row where the user story, Web site for Online Users, is ranked at 1 with an ID of US37, no plan estimate, and a parent value displayed as US20: EPIC: User Experience.  
  
The presenter clicks the ID and a pop-up box containing Description and Hierarchy tabs is displayed.  
  
The Description tab shows: As a user I want to be able to access plant information on an online web site so I can gather research about plants. The acceptance criteria includes the Infrastructure Platform which lists the following: Work with IT to get area to host web site on company web server and web domain name registered and accessible.  
  
The GUI Infrastructure section includes the overall look and feel and contains the points: web page templates defined and web home page minimally designed and operational.   
  
The end of the Description tab displays: Fully Tested.*   
  
And so with all that – down here Fully Tested – with all that the team looks at it and says, "We can do this in one sprint." And then they have to come up with an estimate. Now the estimate is not hours. The level of estimate is a relative thing. A lot of teams use T-shirt sizes; small, medium, large, extra large. Some use Fibonacci series. In our case, we're using numbers; either 1, 2, 4, 8, or 16 – 1 and 2 are small, 4 is medium, 16, I mean 8 is getting large, and 16 is definitely an Epic. And what's important is every member of the team has to agree on what the story points are for this user story. And they've all agreed that it's fairly small and so they are going to put a 2 on it. And if any member of the team doesn't agree that it's a 2, then there's some more discussion until they find out why they think it's more or less and everyone agrees. So now this user story has good acceptance criteria, it has an estimate on it, it's ready to go. It can be moved into a sprint in the next sprint planning meeting.   
*The presenter types 2 in the Plan Estimate column.*   
  
Now let's look at this next one, let's open him up, Display Plant Information. And you can see this one is quite large. The overall user story doesn't sound that bad, they just want to display detailed criteria about the plants. However, if you look at this acceptance criteria there's quite a lot in it. And as the team discusses it, they say well we have a search for information, well that makes sense and so these are all the items we want to search for. Those are on our database, we're fine. Multiple search criteria, okay, we can do that and then when we get it back the list is going to have thumbnail images and these names. Well, wait a minute we only have a regular image – now we're going to have schema change so that makes it larger. And then they want to be able to select a plant to see the detailed information and, oh, let's remove plants from the list. Now we have to have somewhere to store that list that that customer is looking at. So you can see as they talk through it, they can find more details and more items that could be large and not only do they have to search, but now they have to display the detailed information. And the image can be zoomed and then multiple images – well we were only having one image in our database schema so that's another schema change. And then there's this whole thing about being able to save the information, being able to take that information from either the search or the detailed plant information, creating a PDF file that can be printed or stored on the local disk.   
*The presenter highlights the row where the user story, Display Plant Information on web site, is ranked at 2. The presenter clicks the ID, US1, and the page, User Story US1: Display Plant Information on web site, is displayed.  
  
The Description section states: As an online user I want to be able to display detailed criteria about plants in web site so I will know if they are right for me to buy for my garden.  
  
The Acceptance Criteria's Search for information section displays info such as: Customer can search for plants by zone, sun/shade requirements, water needs, soil needs, height, or bloom time; and customer can specify multiple search criteria in a single search.  
  
The Detailed Plant Information section displays information, for example: Detailed plant information will show all records on the plant including the image.  
  
The Save Information section shows that customers can save detailed plant information.*   
  
Well the team says, "This has gotten out of hand, this is way too large for us to estimate, we can't get this done in a single sprint. This story needs to be split into multiple user stories." And you can kind of see how you could split it fairly easily, you can do the search in one user story, the detailed plant information displayed in another and saving the information in the next. So if you go back to the **Backlog**, we've already split out the search for that user story. And so this, this has just the search, nothing else. And the team says, "Well it has some database issues but we understand what the story is, it's fairly large but we think we can get it done in one sprint." And so they take that and they put a 4 on it. So now that user story has acceptance criteria, it's well understood, it has an estimate, and it can be brought into the next sprint. Doing this backlog grooming regularly ensures that these top level stories are well understood, they're ready for the next sprint and it really paves the way in the sprint planning meeting.   
*The presenter navigates to the Backlog tabbed page and the user story, Search for Plants by criteria on web site, is ranked at 3 with an ID of US2, no plan estimate, and a parent value displayed as US20: EPIC: User Experience. The presenter types 4 in the Plan Estimate column.*

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a02_it_enus/output/html/course_transcript.html#top)

The Sprint Review Meeting

Learning Objective

*After completing this topic, you should be able to*

* *describe how the Sprint Review Meeting allows Scrum members and stakeholders to inspect and adapt the product as it is being developed*

**1. Outlining sprint review meetings**

The sprint review is held at the end of each sprint and time-boxed to four hours. In the meeting, the work the team completed in the sprint is demonstrated as a product increment – potentially shippable working software. The meeting is critical because it allows a two way exchange of information between the Scrum members and the stakeholder. First it provides the stakeholder a snapshot of the final product with an incremental addition of features and improvements delivered each sprint. The features are actually demonstrated as a working product, which allows the customer to determine if the features are implemented as expected. Secondly, the sprint review meeting allows the product owner to collect feedback so the product vision held in the product backlog can be adjusted to better fit the collective vision held by the stakeholders. The goals of the sprint are related to the stakeholders along with the product demonstration. As each user story added to the sprint must provide value to the end user, the focus of the demonstration is showing how the incremental features slice added to the product increment in the last sprint benefits the end user.   
  
In addition to demonstrating the completed work in the product increment, the sprint review is also to review the work not completed. The product owner is responsible for determining when an item is done and therefore a part of the product increment. Software that is code complete but has not been fully tested or documented, is not potentially shippable code and therefore not done. Also, if the acceptance criteria and the user story is not fully met, then the user story as a whole is not considered done. The story points from the user stories that are considered done are added together to determine the team's velocity during the last sprint. Over time, the velocity of the team becomes a good indicator of how much work the team can complete in a single sprint. As the team's velocity stabilizes, it enables the product owner to estimate which features the team may be able to complete before the next release state. Items in the sprint backlog that were not completed are moved back to the product backlog in their entirety and prioritized with the other product backlog items for inclusion in future sprints. Also, it is not uncommon for product increment demonstrations and feedback to spark discovery of new features by the product owner and the stakeholders.   
  
Customers often ask for new features or changes in response to the demonstrations they have seen. These new features are converted into the product backlog items, which are prioritized on the product backlog by the product owner. Though the product owner is responsible for determining the priority of the items in the product backlog, collaboration with the customer is critical in determining priority, since new features could displace features originally defined in the scope of the product. In this way it is possible for the customer to replace the original project scope with a new revised scope of more high priority features as the project progresses. Note: preparation for the sprint review should be minimal, no more than two hours. There is no need for PowerPoint slides, as the main event is the demonstration of the working code which is the natural result of testing the items completed in the sprint. The sprint review meeting presents a forum to inspect and adapt the product as it is being developed and iteratively we find the understanding of the requirements for both the stakeholders and the Scrum members.

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a02_it_enus/output/html/course_transcript.html#top)

Scaling Scrum

Learning Objective

*After completing this topic, you should be able to*

* *describe how the Scrum of Scrums Meeting facilitates communication and collaboration between teams working on a large project*

**1. The Scrum of Scrums**

When learning Scrum, the topics often focus on a single Scrum group consisting of a product owner, Scrum master and Scrum team. But how does this scale up to very large projects that have hundreds of people working on them? One technique used to scale Scrum to large projects is to break the work down between multiple Scrum groups which communicate with each other in the Scrum of Scrums meeting. As the Scrum of Scrums participants form a team and will need to work together to resolve issues, if the Scrum of Scrums meeting becomes too large, it becomes difficult to come to consensus on any issue. In projects with many teams there will be multiple Scrum of Scrums meetings, a single representative from each Scrum of Scrums meeting will be sent to the next level Scrum of Scrum of Scrums meeting where a higher level view of the project would be visible. Even larger projects could add additional higher level Scrum of Scrum of Scrums teams until the meeting represents all the teams in the entire project.   
*Heading: Scrum of Scrums.  
  
A diagrammatical representation of a Scrum of Scrums hierarchy is displayed, where the topmost level is the Scrum of Scrum of Scrums, or SoSoS. This is broken down into the Scrum of Scrums, or SoS, teams, which is even further broken down to groups of Scrum teams.*   
  
Scrum of Scrums meeting are normally held two or three times a week with the first part of the meeting designated for every member to give status, much like a normal Scrum meeting. The status portion of the meeting should not take more than 15 minutes where each member answers four questions: "What has your team done since we last met? What will your team do before we meet again? Is there anything blocking your team or getting in their way? And are you about to put something in another team's way?" The last question gives other teams a heads-up that something is going to change that may affect them. It is also a good rule of thumb to not allow the status to include the names of people on the team as this tends to go into too much detail about team member activities. The Scrum of Scrums is for information about the team as a whole, not individual members. The last part of the meeting is where the Scrum of Scrums differs from the normal daily Scrum meeting. The daily Scrums are not for problem solving, problems are identified and taken offline after the Scrums dissolve.   
  
However, in the Scrum of Scrums the right people from across the organization are all in the room, able to work any issues that are identified. And if an issue was brought up in the Scrum of Scrums meeting it is often affecting multiple teams and needs to be addressed and resolved in the meeting if possible. Any issues not resolved in the Scrum of Scrums meeting should be placed on the Scrum of Scrums backlog on an action item list that can be brought to the next higher level Scrum of Scrum of Scrums for resolution. The Scrum of Scrums follows the same rules as the daily Scrum, in that stakeholders are allowed to attend to observe. However, they are considered chickens in the meeting and are not allowed to speak. This keeps the meeting productive and short, focused on the coordination and issue resolution between technical teams. While the Scrum of Scrums meeting is very useful to facilitate communication and collaboration between teams on a large project, it should be noted that is a bottom up approach. The Scrum of Scrums does not replace the need for leadership providing the holistic planning, context and framework for the entire project.

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a02_it_enus/output/html/course_transcript.html#top)

The Definition of Done (DoD)

Learning Objective

*After completing this topic, you should be able to*

* *use the Definition of Done (DoD) checklist in Scrum to ensure that the best practices are followed during the development of high-quality software*

**1. The DoD checklist**

The team comes up with their own DoD, or Definition of Done, but the DoD can mean different things to different activities. The DoD checklist for a feature could involve unit test, QA testing, documentation, refactoring, or code reviews while the DoD for a sprint would be completion of all the items in the sprint backlog. And the DoD for a release would involve steps to transform the potentially shippable product increment into a release product. The team needs to have a clear DoD checklist for any activity they engage in that will be reported. So when they say the activity is done, there is no misunderstanding on what they have accomplished. And the team's DoD checklist is not set in stone; it can be modified or expanded in-between sprints. As process improvement action items are identified in the sprint retrospective meeting, they can be added to the team's Definition of Done to ensure they are implemented.   
 *An example of a Track tabbed page displaying an Iteration Tracking Board is displayed. It contains the columns: Defined, In-Progress, Completed,and Accepted. Two stories – US40 and US32 – are listed in the Accepted column and one story – US41– is listed in the Completed column.  
  
The Tracking page also includes an Iteration Burndown chart and sections for the Iteration Summary and Recent Activity.*   
  
One of the most important Definitions of Done involve user stories as the features they describe cannot be demonstrated in the product increment until the user story is done. Here in our sprint tracking board, the items here in the Accepted column are the ones that are done. User stories in the sprint backlog are broken down into tasks which, when completed, should produce software that satisfies the user stories acceptance criteria. However, the team might find during development that a task was not needed. You can see here, this is only 94% complete but it is a done user story so not all the tasks need to completed for the user story to be done. Once the tasks needed to satisfy the acceptance criteria are done the user story is complete, but it still might not be done. To be done, the user story's features must be demonstratable in working software that's in potentially shippable form and the product owner is the one who makes the call.   
*One of the stories in the Accepted column shows 94% complete. The presenter hovers over the completed story's ID and the acceptance criteria regarding that story displays.*   
  
The user story is not done until the product owner verifies the software satisfies the acceptance criteria, as we've shown here, and all the value-added activities on the user story's DoD checklist are also complete. If the DoD value-added tasks are not finished the software is not potentially shippable. Potentially shippable code is designed, coded, QA tested and documented at the very least. It could also include the code being checked into the CM system and tagged as a release, integrating the code base of all the remote teams or that the code was reviewed and followed the team's coding standards. And don't forget about updating the installation software to including any new components or configuration. In the case of US41 here, all the tasks were complete but it was not accepted as done.   
  
So some activity on the DoD checklist was not finished and the product owner did not declare the story done. As the team is not awarded a user story points for their velocity until the story is done, the DoD is an important tool to ensure not just the code but the value-added tasks are also complete. So the software is the potentially shippable form. This is where the DoD's explicit checklist of demonstratable tasks comes in to verify the user story is truly done. Now if the user story is declared done, its feature is accepted into the product increment to demonstrate to the customer after which the user story can be discarded. If the user story is declared not done it is returned to the product backlog to be included in a future sprint. This unfinished work is technical debt, which should be completed as soon as possible. Allowing technical debt to accumulate in the product leads to a code base that is an undefined state and if the incomplete tasks include activities like refactoring or bug fixes the product could become unstable as well. The Definition of Done is directed at the quality of the product produced, not the features or functionality the product delivers. And the checklist of value-added activities ensures best practices are followed which leads to high quality products.

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a02_it_enus/output/html/course_transcript.html#top)

Why Should User Stories Be Small?

Learning Objective

*After completing this topic, you should be able to*

* *describe the advantages of using smaller user stories when preparing for a Sprint in Scrum*

**1. Advantages of smaller user stories**

User stories at the top of our product backlog here are high priority. They're ready to be included in the sprint and they need to be small. Now a team should be able to complete a user story in a few person hours to several person days, and some teams size them so small that every team member can complete a single user story a day. Now this isn't always possible but the biggest user stories should not be more than a quarter of the team's capacity in a single sprint and understand, these larger user stories are adding risk to the sprint. Now if we look at our user stories here, we see these two are very small, they're sized well, this one probably is way too big, it's going to have to be split – three, two, one – so these are sized fairly small in the estimate of the team. If we look at some of these, like let's look at this one here, US47, it's nice and small and direct, "As an online user I want to be able to save or print the detailed plant information so I can use it as reference for plants." The plant web site is going to be detailed plant information and we want to be able to print it or save it. Acceptance criteria – they can save as the PDF file, create a PDF file that can be printed or it can be saved to a local disk. "Unit tests complete. User documentation complete. Fully tested."   
*An instance of team collaboration software is displayed. The interface contains four tabs: My Home, Backlog, Plan, and Track. The Backlog tabbed page is displayed and stories are ranked by priority with individual IDs and plan estimates. Most of the stories have low plan estimates except for one, which has a planned estimate of 8.  
  
The presenter opens a story named Save Detailed Plant Information Displayed on website, which has an ID of US47.*   
  
So this is good acceptance criteria, it's easy to understand, someone can take this on the team and get it done very quickly. It only has one story point and we will be able to test it. Small user stories like this one provide focus for the team, it's hard for them to get lost in the details. It allows the testers to start testing sooner since there are handoffs happening throughout the sprint so you don't get the backlog at the end of the sprint where the tester is the only one that has any tasks left, trying to test everything that just got completed. And we know that during any development effort there is going to be discoveries and adaptations that need to be happening and it's easier for the team to work around any issues or discoveries that they find if these stories are small. Also smaller stories almost guarantee the team will deliver something of value at the end of the sprint. And there are several issues with the larger stories. The larger the story is, the more it is prone to getting blocked or developing bottlenecks and this could tie up team members that are needed on other stories in the sprint, leading to low productivity. Also remember the entire story in a sprint is either done or not. Larger stories taking up most of the capacity of the sprint are risky because if most of the work is done with only a few small items unfinished at the end of the sprint, the entire story is not done and will be left out of the sprint review, giving the impression that the team did not complete much work in that sprint.   
  
We can see that if we go and look at Sprint 4. This sprint is complete and we can see that we got this user story done with two story points and we got a spike done but we have this user story that's still in progress, it wasn't complete, it was not added to the product increment, was not able to be demonstrated at the sprint review. We got it 71% complete so let's take a look at the acceptance criteria. We can see here there's quite a lot of it. That's a flag right there that this might be too big. So we want to be able as an online user to search for plants that fit criteria so we can find the plant to buy that will thrive in my garden. We want a search of the information, we can search by all these different zones, sun/shade requirements and so on. Second one, we wanted to search by multiple of these criteria. Three, when we submit, we're going to get this list back with a thumbnail image and the common Latin name. And then four, the customers can remove the plants from the result list, they don't want them in their list and of course our Fully tested, Fully documented in Unit test.   
*The presenter selects the Track tab and displays the Iteration Tracking Board for Sprint 4. There are two stories in the Accepted column and one story in the In-Progress column. The presenter hovers over the story ID – US2 and a pop-up box displays the lengthy acceptance criteria for the story.*   
  
Well we can see right away that this is large, hard to understand. In fact, it had six story points so it's quite large and we can also see that there's some things that could have been taken out – split this to make it smaller, for one thing item 4 –"Customers can remove plants from the list to narrow down their results" – that could be its own user story after the results came back in the first user story. Customers could search on one of these criteria and then we could have another story to say, okay, you can search on multiple. So I see at least three user stories that could be broken out here as slices of functionality or features. If we look at the tasks that were completed or not, the team did a lot of work, they got a lot of tasks done, all these are complete. Let's go to the next page, okay, here's our In-Progress: Development. The task was to "accept request to delete items from the database search result list, delete the results stored in the user session, resend updated results to the web."   
*The presenter clicks the Tasks icon and the Tasks page displays a list of 17 tasks. The presenter navigates to and highlights one of the In-Progress tasks from the DEV team.*   
  
Well the developers working on that, they still have two hours to go at the end of the sprint, so it's In-Progress. Since it's In-Progress, QA's task to test it are Defined, they're not even in progress. You come down here, In-Progress documentation, "Document search functionality with result list deletions." Well they have two hours left, they can't document the deletion because the developer wasn't done and they can't get a screenshot of it – it's two hours left there. So even though the team did a lot of work they got 71% of this big user story done. This user story, the acceptance criteria was not met and so the user story was not done, was not accepted into the product increment, could not be shown at the sprint review. And so the team did not get the velocity points – all six of them there. When it comes to user stories, just remember smaller is definitely better.   
*The task list shows the DEV, QA, and DOC teams are all delayed because of this one incomplete task.*

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a02_it_enus/output/html/course_transcript.html#top)

The Customer User Story

Learning Objective

*After completing this topic, you should be able to*

* *describe how to write consumer user stories*

**1. The consumer user story**

Customer user stories capture functionality needed in the product increment. Each user story will have a name and a description. The user story descriptions follow a template, which is from the user's perspective. The form is, as user I want <function>, so <value>. So as an online user, I want to be able to save or print the results of my search – that's the function. What's the value? So I can use them to shop for plants. This gives the who, what and why of the user story in the user's voice and ensures every user story is focused on how it delivers value to the end user. User stories bridge the communication gap between customers and technical workers because they are written in the language of the user, not the developer's technology language. User stories provide a common language understandable to both the user and the technical team. The product backlog and sprint backlog give the customer visibility into when they can expect each feature to appear in the product increment. The product backlog is prioritized by the product owner, with high priority items at the top and low priority items at the bottom.   
*An instance of team collaboration software is displayed. The interface contains four tabs: My Home, Backlog, Plan, and Track.  
  
An example of a user story with the story ID as US1 is displayed. The presenter focuses on the Description section which states: As an online user I want to be able to save or print the results of my search so I can use them to shop for plants.  
  
The presenter selects the Backlog tab. The topmost story – User Registration – has a rank of 1, an ID of US49, and a plan estimate of 5.00.   
  
The presenter scrolls to the end of the page where the last story is EPIC: Data Mining Synchronization, which has a rank of 33, an ID of US46, and no plan estimate.*   
  
The customer knows that these user stories at the top of the product backlog are ready to be pulled into the next sprints and it will be implemented in the near future while user stories at the bottom of the product backlog will not be worked until sometime in the future. The lower the item's position on the product backlog, the more distant the future when it will be worked. In fact these low priority items are large, inviting a conversation between the team, product owner and end user in order to fill in the details, but these conversations are always deferred until the user story reaches high enough priority that is close to being pulled into the sprint. Deferring elaboration of the user story allows it to be detailed in the context of where the project is at the time, when the customer has more knowledge of what they want then it was possible at the start of the project. During the sprint planning meeting, the highest priority items on the product backlog are moved into the sprint backlog for the next sprint. These are the features that'll be implemented in the next sprint.   
*The presenter selects the Plan tab and the page displays an Iteration Planning Board which has stories in each section. The sections include Backlogs and Sprints 5, 6, and 7. There are no stories in Sprint 7.*   
  
By looking at the current sprint backlog the customer will always know what they're going to see demonstrated in the sprint review meeting at the end of the sprint. Here on our tracking board we see the user stories in the current sprint – which are Accepted, which are In-Progress, the customer can see this and know exactly what they're going to get in the sprint review meeting. Now user stories are different from requirements written for the waterfall project. To capture what needs to be created, requirements used the strict, you shall do mode, for describing features in great detail while user stories are more informal and open-ended. An "I want something like this" kind of statement, which leads to a conversation with the end user to clarify details, just before the feature is implemented when the most is known about it and the project. Since requirements represent a contract of what the product shall do and the customer doesn't get the product till the end of the project, requirements need to be approved by the company and the customer, logged, tracked during the project and archived after the project is complete.This allows them to be referenced after product delivery to determine if the requirements were met.   
*The presenter selects the Track tab and the page displays the columns: Defined, In-Progress, Completed, and Accepted. The page also has sections for a graphical representation of a Iteration Burndown, an Iteration summary, and Recent Activity.  
  
The presenter hovers over the story ID, US41, in the Accepted column and a pop-up window displays the customer statement and the acceptance criteria of that story.   
  
In this example, the customer statement is: As an online user I want to be able to display detailed criteria about plants on the web site so I will know if they are right for me to buy for my garden.  
  
The acceptance criteria is: Displayed Plant information; detailed plant information page will show all records on the plant including the image; image may be zoomed; and multiple images may be stored and shown as thumbnails which the customer can select.*   
  
By contrast, user stories can be submitted by anyone, the customers, stakeholders, product owner or the team, any time during the project. They are stored in the product backlog, prioritized with other user stories, but then they are split into smaller user stories as they move up the product backlog. Just before the user story is worked, the important acceptance criteria is added, representing the conditions of satisfaction or exactly what the team must do in order to satisfy the user story's requirement. Once the acceptance criteria is met, the user story is done. The feature is demonstrated to the customer in the sprint review meeting and the user story can be discarded. The customer has seen the working software written for the user story requirement which satisfies the acceptance criteria, there is no need for a permanent record for the customer. In Scrum, the customer plays a vital role as they are involved in creating the user stories describing the final products features, discussing and refining the team's understanding of the user story and, through the product owner, supplying the acceptance criteria that determines if the implementation satisfies the user story requirement. And the Scrum artifacts and the sprint review meeting keep the customer involved and engaged in refining the final vision of the product.

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a02_it_enus/output/html/course_transcript.html#top)

Sizing User Stories

Learning Objective

*After completing this topic, you should be able to*

* *size user stories in a product backlog using Scrum principles*

**1. Sizing user stories**

Here is our product backlog, with the highest priority user stories on the top and user stories that have not yet been estimated further down in the list. These are the ones the team will size during the product grooming meeting. The size is not an absolute value – instead it shows how big the story is, relative to the other user stories in the product backlog. The team will size these user stories with a scale with a large granularity. Some teams use T-shirt sizes of small, medium, large, extra large, and extra-extra large. A small or medium estimate means the story would easily fit into a sprint. A large estimate would be suspect, while extra large is too big and must be split. Extra-extra large would be an epic. Other teams use a number sequence – like a modified Fibonacci series – as we're doing here in our project.   
*An instance of team collaboration software is displayed. The interface contains four tabs: My Home, Backlog, Plan, and Track. The Backlog tabbed page displays stories that are ranked by priority, with individual IDs and plan estimates.*   
  
Using the numbers 1, 2, 3, 5, and 8 maps to small, medium, large, extra large, and extra-extra large with the same meaning. These number estimates are called story points. At the end of the sprint, the team is awarded the story points of all finished user stories. Adding up the story points awarded gives the team's velocity for the sprint. Now constraining the estimates that can be given to a limited set of well-known values, limits disagreement on the details. Small, medium, large and extra large are easier to understand and come to consensus on than, say, sizes 2 through 20. As only the team who would be doing the work can decide on the size of the stories, the team decides on the scale it will use and what each size means. What's most important is the team becomes consistent with their estimates. Now looking at US17 down here, "As an administrator I want to have a record of the customer's address and past transactions so I can personalize their experience on the web site."   
*The presenter scrolls to the story Customer record in database. It has an ID of US17 and a rank of 16 with no plan estimate. The presenter hovers over the ID and the pop-up box displays the description: As an administrator I want to have a record of the customer's address and past transactions so I can personalize their experience on the web site.*   
  
Now as the team discusses the story with a product owner, a number of questions come up. How would the customer's address be used? Will we need to access outside servers to find weather or zone information? Will the past transaction be used to select targeted advertising – special offers or something else? And what type of personalization is possible and expected? The team thinks – at a minimum – this would involve database schema changes and complex software to mine the data and make assumptions on how to personalize the customer's experience. As well as accessing outside servers for relevant information. They give an estimate of 8: Epic. If the product owner could answer their questions, they could work on splitting the story. Otherwise the product owner would need to talk with the stakeholders to get answers for the team so the story was better understood. But what happens if different members of the team disagree on the story's estimate? Let's look up here at US31. "As a registered online user I want to be able to customize my login home page so I have easy access to areas of the web site of interest like past searches."   
*The presenter scrolls to the story Customer can customize their portal home page. It has an ID of US31 and a rank of 13 with no plan estimate. The presenter hovers over the ID and the pop-up box displays the description: As a registered online user I want to be able to customize my login home page so I have easy access to areas of the web site of interest like past searches.*   
  
Now in a cross-functional team, different team members have different skill sets and different perspectives. The GUI developer could see this as challenging but doable. They've implemented portals before – maybe a 3. While our QA tester could see it as a 5, because of all the possible ways the portal could be personalized – it would take a long time to test them all. And the web developer might think of it as a 2 since the server was architected with components, each retrieving different information. It's an easy job to just call the component for whatever the customer wants. This discrepancy in estimates could start a chaotic discussion with everyone trying to state their opinion for the team. To bring order, many teams use Planning Poker to estimate user stories or tasks and come to consensus. In Planning Poker, each team member is given a set of cards representing different estimate values. In our case, there would be a card for 1, 2, 3, 5 and 8 – the valid story point values.   
  
After the user story is read, team members simultaneously display a card representing their estimates. Estimates that do not agree are an invitation to a conversation. The team members with the highest and lowest estimates are asked what prompted their choice. In the discussion that follows the entire team uncovers assumptions and questions they may not have thought of individually. Sometimes the discussions result in consensus. Other times another simultaneous vote is needed until the entire team agrees on the estimate. Planning Poker combines individual estimates with group discussions, helping the entire team learn more about why each member gave their estimate, which results in them quickly reaching a consensus. Everyone on the team has an opinion which is heard. And it's a fun, efficient way for the team to find missing assumptions and come to agreement.

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a02_it_enus/output/html/course_transcript.html#top)

Why Do Teams Break Down User Stories into Tasks

Learning Objective

*After completing this topic, you should be able to*

* *describe how and why teams split user stories into tasks to track user stories' progress in a Sprint*

**1. Splitting user stories**

User stories say what the team must implement. Here US32 says, "As an administrator I want to be able to edit any plant records so I can ensure the information in the database is correct." This is what the user wants to do and the acceptance criteria makes it clear exactly how the team will know that they have done everything they need to do what the customer is wanting. And the task breakdown tells how the team will implement this. So tasks help the team mindset move from analysis or what needs to happen to design or how it would be done. The team breaks a user story into tasks together. So the team will emerge from the experience with a shared experience of what needs to be done for each user story and how it will be accomplished. Tasks should be independent. We see here our task are broke out: we have QA tasks, we have presentation tasks, here our architect is going to do some design. We have our web developers – going to do some implementation on the server – separate from the presentation. And of course we have our documentation.   
*An instance of team collaboration software is displayed. The interface contains four tabs: My Home, Backlog, Plan, and Track.  
  
An example of a user story is displayed where the ID is US32. In the General section, the description is: As an administrator I want to be able to edit any plant records so I can ensure the information in the database is correct.   
  
The assumption is: Build on existing Administrator Utility with authentication. The acceptance criteria is: Search and view plant records; and enter edit mode which allows update of plant record. Update text information, update bounded data (zone, type, bloom season, etc), and update image by uploading a new image. Save data and ensure data is updated in the database. Ensure updates are maintained over a database restart. Unit Tested. Fully Documented. Fully Tested.  
  
From the menu, the presenter clicks the Tasks link. The Tasks page displays tasks under the columns: Order, ID, Name, Iteration, State, Estimate, To Do, and Owner.  
  
For example, one task is QA: Create test plan for edit plant information. It has an ID of TA136; an iteration of Sprint 5; a state which includes defined, in-progress and completed statuses; an estimate of 12 hours and 0 to do hours; and the owner is the Tester team.*   
  
These tasks are independent, which allows multiple team members to work on the same user story without overlapping work. So the team can swarm the highest priority user story like this one and complete it as much as possible before any team member moves on to the next user story. The entire team working in parallel ensures the user story will be completed rapidly. Now tasks are recorded in the sprint backlog along with their user story. A single team member commits to completing a task – taking ownership of the task. Here our ownership column shows who owns each of these tasks. So tasks are a way of saying who is doing what to complete the user story. And assigning task owners makes each team member accountable to the rest of the team. Tasks help the team organize their work, as each team member is responsible for completing their tasks, communicating when the task is done, or any issues or blockages they're encountering. And starting work on their next task when the current task is complete. No direction from management is needed.   
  
Since the tasks tell the story of how the team will complete the What of the user story, they should be focused on satisfying the stories' acceptance criteria. So when the tasks are all complete, the user story will be accepted and declared done. And tasks are used to show team progress. Tasks are tracked daily to show the progress of the sprint. Here we see we've each task you'll see there under Defined, In-Progress, or Completed state. And the tasks owner also updates how many hours are left to complete the task. Now the task hours that were originally estimated and the ones left to complete the task may not always match as the sprint progresses, as the work needed to complete the task may grow or shrink as the team member learns more about the task. If more hours are needed to complete the task that's what should be shown in the task status to give a realistic picture of how the team is doing.   
  
Now tasks are not set in stone. As the team works through the sprint and gains a deeper understanding of what is needed to complete the user story, they may uncover invalid assumptions or discover new work that was not recorded in the task breakdown. It's common for tasks to be added, deleted, or modified from the sprint backlog while the sprint is in progress. However, there needs to be some discipline – as adding and deleting many tasks makes it hard to measure how the team is doing in the sprint. Here we see New Task Z was added to the sprint. It's a quite involved task, because it's going to involve four different team members and there is a lot of work, a lot of hours that have been estimated for it. Let's see how it does with our tracking. Here we're on our Iteration Status board and we see that we have a user story complete, and a SPIKE complete, and here is the user story we've been looking at.   
*The presenter focuses on the rows with New Task Z. The tasks involve four different teams: GUI, DEV, QA, and DOC. The estimated hours is between 8 to 12 hours for each team.  
  
The presenter selects the Track tab which displays the Iteration Tracking Board for Sprint 5. The Defined column is empty, the In-Progress column shows story ID US32, the Completed column is empty, and the Accepted column contains two stories – US40 and US41.*   
  
Now adding or deleting tasks messes with your burn chart's ideal guideline. We see here – the first day of our sprint – we hadn't added any tasks yet so we were late on a sprint planning meeting. The first day we had a 134 hours and then the next day we went up to a 136 hours. And then today we went all the way up to a 161. We've been adding tasks – and that's even with the completed user story of three story points. There were quite a lot of tasks in this user story already with hours that were completed. So our ideal burn down started at 275 even though we started the sprint with only a 134 hours. Now here we're using an electronic tool, so the guideline was altered to show the total amount that changed with the work added or deleted.   
*The presenter focuses on the Iteration Burn down chart – a graphical representation of the work done each day.  
  
The Iteration Summary section shows 3 Days Remaining in a 6 day Iteration and 42% of the work – or 3 of 7 points – is Accepted.*   
  
If the team was using a manual method, like a burn down chart on a whiteboard, the original guideline probably wouldn't have been changed and it would have started down here and gone right through the new tasks. In either case, altering the original task makes it hard to get a clear picture of how the team is progressing on the sprint. Any task that's added while the sprint is in progress must be needed to satisfy the user stories acceptance criteria. This keeps the team focused on the user story and keeps irrelevant tasks out of the sprint backlog. And all team members need to be aware of the new tasks and what they entail. So the collective image of what needs to be done to complete the user story is preserved.

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a02_it_enus/output/html/course_transcript.html#top)

Encouraging Self-Organizing Team Behavior

Learning Objective

*After completing this topic, you should be able to*

* *describe how the Scrum Master can encourage self-organization within the Scrum Team*

**1. Encouraging self-organizing behavior**

The Scrum master is essential to the success of the team. Not only as the enforcer of the Scrum process but also as the cheerleader touting the Scrum's values and Agile principles. The Scrum master is responsible for making the team as productive as possible. Part of this is facilitating the sprint planning meeting so the sprint is correctly sized. During the meeting, the Scrum master monitors the work the team accepts, comparing the work taken on by the team against its velocity. Our team has a velocity of 7. If the team exceeds its capacity like here in Sprint 4 – they ended up taking on 8 story points – there is a danger the work accepted will not be completed in the sprint, and that's what happened here as US2 was not completed. And so those 6 story points were not added to the team's velocity. However, the team should always be striving to improve. If the team seems to be slacking like in Sprint 6 here, they only took on 5 story points and a SPIKE, they should be encouraged to take on enough work to reach their velocity.   
*An instance of team collaboration software is displayed. The interface contains four tabs: My Home, Backlog, Plan, and Track.  
  
The Plan tabbed page is displayed and stories are listed in the Backlog column, as well as in the Sprint 4, 5, and 6 columns on the Iteration Planning Board.  
  
Two of the stories listed in the Sprint 4 column make up eight story points. One of the stories, US2, making up  six of the points, displays 71% completion with two active task remaining.  
  
The Sprint 5 column displays three stories making up five story points.*   
  
Sizing a sprint so the team has enough work to push themselves to greater productivity while not being over worked, will encourage them to communicate, collaborate, and help each other to complete the work they committed to as a team. Now during the sprint, the team should be alert to signs of the team not modeling team behavior, as commitment is one of the Scrum values. During the daily Scrum, the Scrum master should hold each team member to their commitments. If a team member does not live up to their commitments, the Scrum master can say, "You committed to completing the task but didn't do it, is there anything blocking you?" If there's a blockage the Scrum master can facilitate getting it removed. If there was no blockage, team peer pressure will raise honor and commitments and importance for the entire team, especially if a team member is blocked because the commitment was not honored. The Scrum master may need to remind the team that while the skills required to complete a high priority task may align well with the skills of one team member, the entire team is responsible for all the tasks.   
  
If we look in Sprint 4, US2, this one that didn't get completed, this is a really large user story. It has 6 story points, and it got 71% complete, so the team did a lot of work here. Let's look at the tasks to find out what happened. Well we can see most of the tasks were complete, but here we have a development task. This is deleting items from the database search result, deleting it from the result sets stored, and resending it to the Web. And that was not complete; it was in progress, there was only two hours left on that. But since this development task was not completed, QA could not test it. So there's two tasks there that are not complete; documentation could not complete. They have two hours left and they couldn't finish taking all their screenshots.   
*The presenter opens the story ID, US2. The page User Story US2: Search for Plants by criteria on website is displayed.  
  
The presenter clicks the Tasks link and highlights the row where the task, DEV: Accept request to delete items from DB search results and delete it form results stored in user session, resend updated results to web, displays. The estimated hours shows 6 and the to do hours is 2. The state is defined and in-progress.  
  
Two related QA tasks follow – ensure item deleted from search result no longer appears in list and ensure item deleted from list reappears if same search criteria is used in new search – with 6 estimated hours and 6 to do hours each. The DOC team also has two tasks – document search functionality with result list deletions and task screen shots of all screens – showing a sum of 18 estimated hours and 4 to do hours relating to the incomplete DEV task.*   
  
Now the team succeeds or fails to completing the user stories in the sprint as a team, so if one member falls behind, the other members of the team should be encouraged to pitch in to ensure all tasks are completed. In this case, the architect or GUI developer could have helped the web developer finish up these two hours. And then anyone else on the team could have picked up one of these QA tasks to ensure that the testing was done in parallel and happened as quickly as possible. And someone else could have definitely taken the screenshots for the documentation...team member. And there's some ways the Scrum master could have encouraged this optimal behavior within the team. The web developer was obviously working on a high priority task that was blocking other tasks. During the daily Scrum the blocked task should have been reported. This could prompt the Scrum master to ask if there is anything the other team members could do to help move this task to "Done." This could involve auxiliary efforts such as performing a code review, assisting with a bug, or fixing a broken build. Or it could involve helping with the task or taking on the task itself, as in the case of the QA tasks.   
  
Asking this question helps widen the team's perspective to what the entire team is doing rather than just focusing on their own task work. After all, the entire team is collectively responsible for completing the user stories, so the entire team succeeds or fails. Asked enough time, and the team will start to model this behavior automatically when choosing what they will work on next. Focus is another Scrum value. The Scrum master can facilitate limiting the team's work and progress, both as a team and in individual team members, to help the team stay focused. If the team is encouraged to work on the same user story, they will share the same focus, collaborating and working together to move it to "Done." Working simultaneously on too many user stories as a team, or too many tasks as an individual, leads to context switching which breaks the team's focus and decreases productivity.

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a02_it_enus/output/html/course_transcript.html#top)

User Story Overview

Learning Objective

*After completing this topic, you should be able to*

* *describe User Stories*

**1. Describing user stories**

User stories capture the features and functionalities the customer wants to see in the final product. Sounds pretty simple, but there's a lot more that goes into writing a user story that captures the value of the feature to the user and also invites a conversation for the team to learn more. A user story describes a feature a customer or a stakeholder wants to see in the final product. A user story starts up very simple, with a filled in user story template. Now the user story template has three blanks; user, function and value. And here I have it in our user story; I wrote it in here. The template goes, "As a <user>, I want <function>, so that <value>. This captures the who, what, and why of the feature. A user story never describes the How. And up here we've the simple user story, "As a marketing Product Owner I want to store each customer's past transactions, so I will know their tastes and use it for direct marketing."   
*An example of a user story is displayed. The name of the story is Store customer plant criteria from past purchase. The description is: As a Marketing Product Owner I want to store each customer's past transactions so I will know their tastes and use it for direct marketing.  
  
The user story template displays: As a <user> I want <function> so that <value>.  
  
The owner is the product owner.*   
  
The Who is the user – here it's the Marketing Product Owner. Now the user must be a person, not a system or company. The What down here is the function. In ours, we want to store each customer's past transactions. What the user wants must be unique to the story in the project. No other stories should deliver the same function. And then we're down to the value, this is the Why. The value is so that they will know their taste and use it for direct marketing. The value is a benefit gained by the user or for another person or system related to the user. Now this user story template format forces the story to deliver value to the end user in a way that is understandable to the team. This is the way a user story starts out its lifecycle, very simple and open ended. And you don't need an electronic software application to create your user stories or manage a SCRUM project. Many projects put their user stories on index cards, which they can move around on a whiteboard, a cork board, or even a hallway wall to prioritize them. In fact, the index card is the first C in the three Cs involved in the user story.   
  
The user story three Cs are Card, Conversation, and Confirmation. The card is this initial user story with a simple story template filled out. Now there's not enough information here in the template to enable the team to estimate and develop the features. So the card is an invitation to a conversation. That's the next C, the conversation. This is normally multiple conversations held at different times. Where the team, product owners, and stakeholders drill down into the details of the story to come to a shared understanding of what the user story on the card really means. The user story might change as details are added. As the details of the user story are uncovered, it's normally split into multiple user stories. You see here our epic experience, there is a couple of user stories that were split in two. And then we have this web site customer portal. There are several from that. So these large user stories that are very ill-defined are split into smaller ones that have more focus and are easier to estimate and understand.   
*The presenter selects the Backlog tab. The page displays rows of user stories ranked by priority with individual IDs and plan estimates.  
  
The presenter focuses on the Parent column which displays different Parent values for different rows. The various Parent values displayed are: US20: EPIC: User Experience, US30: EPIC: Web site Customer Personal Portal experience, and  US11: EPIC: Plant Database.*   
  
Looking at **US47** here, we can see that this user story is small and focused. It says as, "As an Online user I want to be able to save or print the detailed plant information so I can use it for reference for my plants." Now this is much smaller and focused than what we had before and that very large user story. And so the team can understand this and estimate it. And all they need to know is when they are done with it. Which brings us to the last C, Confirmation; the confirmation is the completed acceptance criteria. With their shared understanding, the product owner and team agree on acceptance criteria, which will be the definitive authority on when this user story is done. The acceptance criteria must be clear, it must be testable. And when the software developed satisfies this acceptance criteria, the story is done and can be included in the product increment. User stories start out general, as a filled out user story template which captures the essence of the customer's requirements on the card. The card is open ended which leads to a conversation, where the team asks questions and comes to a shared understanding of the user story requirements with the stakeholders and the product owner. This allows the creation of acceptance criteria or confirmation, enabling the team to develop the feature and know when they are done with the user story.   
*The presenter clicks story ID US47 and the details for this story are displayed. The description is: As an Online user I want to be able to save or print the detailed plant information so I can use it for reference for my plants.  
  
The acceptance criteria is: Save Information: The customer can save detailed plant information, create a PDF file of detailed plant information that can be printed, and enable ability to save PDF file of detailed plant information to local system.  
  
Unit tests complete. User documentation complete. Fully tested.*

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a02_it_enus/output/html/course_transcript.html#top)

Epics and User Stories

Learning Objective

*After completing this topic, you should be able to*

* *split EPICs into a hierarchy of User Stories that can be completed in a single Sprint*

**1. Splitting epics into user stories**

If you think about it, most projects start out with a single idea. In our project – it's our Plant web site. Then as the item is discussed and analyzed, it's broken into larger feature sets, which are broken down into smaller features and components, which have a more narrow scope until the component is small enough to understand, estimate, and create. Well Scrum is no different. The vision of the finished product is broken down into epics, or very large user stories that are captured in our product backlog. However, these epics are only place holders for that final feature set, because they describe features that cannot be reasonably estimated, due to complexity, too big a scope or there are number of unknowns that must be defined. Here we have a couple of epics here at the bottom, you can see one is user training. And if we look at it we see, "As an administrator, I want to be able to train on the new web site so I will know how to manage the site." So this is a whole user training initiative. But then there is this note at the bottom, we also may need online videos and extensive help for customers using the site. Well that probably needs to be another epic. This is way too big for us to be able to estimate or get our head around even.   
*An instance of team collaboration software is displayed. The interface contains four tabs: My Home, Backlog, Plan, and Track. The Backlog tabbed page displays rows of stories where the parent values are various epics such as US11: EPIC: Plant Database, US30: EPIC: Web site Customer Personal Portal Experience, US19: EPIC: CRM Initiative, US14: EPIC: Customer Shipping, and US12: EPIC: Purchase Plants.  
  
The presenter scrolls to the bottom of the list of user stories to the rows named EPIC: user training and EPIC: Data Mining Synchronization.  
  
The presenter hovers over the epic with the ID US44. The information displayed in the pop-up box is: As an administrator, I want to be able to train on the new web site so I will know how to manage the site." A note at the bottom of the pop-box states: Also may need online videos and extensive help for customers using the site.*   
  
And then here we have another epic. That's our data mining epic. As a marketing professional I want to access the customer information, so I can feed it to our data mining application to extract useful information. Well that's a very broad feature. And here we have some notes that they wanted the data massaged so it'll synchronize with the databases. This is huge, this is going to take several sprints. And that's all we know about it, they're very broad. The epics are not defined yet. Now in the backlog grooming meeting, the highest priority epics can be broken down into multiple child user stories as they're discussed in details as that it. So as you go further up here you can see we have user stories that are part of epics. And you have this CRM initiative that shows up several times and the user stories have been broken out from the CRM epic. Some detail has been added. However, they're still not ready to go into sprint. They...if you look here, Store Customer information, it doesn't have any acceptance criteria. It's just a simple user story, there is no estimates, these are still very large.   
*The presenter hovers over the epic, US46, Data Mining Synchronization. The pop-box contains the message: As a marketing professional I want access to the customer information so I can feed it into our data mining applications to extract useful information. Marketing has expressed an interest in the captured customer information. They want data massaged so it will synchronize with their databases so they can feed it into their data mining applications to product reports.  
  
The presenter scrolls through the rows of user  stories to those whose Parent value is US19: EPIC: CRM Initiative. The names of the user stories are Customer record in database, Store customer information in database, and Store Customer address.  
  
The presenter hovers over the user story, Store Customer address, and the pop-up box displays the information: As a Marketing Product Owner I want to store each customer's address so I will know their zone and planting dates for direct marketing. No acceptance criteria is visible.*   
  
Now also in the product grooming meeting these bigger user stories are broken up into smaller ones. And these user stories, they can have acceptance criteria, they can be well understood and the team will know when these are done and they'll have an estimate assigned to them. These are ready to go into the sprint. As we're going up in the hierarchy, in the product grooming meeting, we're breaking down the scope of each user story. So it's small enough for the team to estimate, commit to, and put in a single sprint. If the team is not comfortable estimating and committing to a story. it's still too big, it needs to be split into multiple user stories. Since only the highest priority items are discussed in the Product Backlog meeting, the items here at the bottom tend to be very large and fuzzy, with few details. These could take many iterations to complete. The items in the backlog middle all tend to be larger stories with a little bit more detail, but still not well defined enough to fit into a single sprint.   
*The presenter scrolls to the top of the list of user stories where the stories have plan estimates and their descriptions include acceptance criteria.*   
  
Well the user stories at the top are much smaller in size and scope, well understood, ready to be accepted into a sprint to be worked. And as the project progresses these user stories are taken off the backlog and put into the sprint backlog and the user stories at the bottom are moved up. And then in the product grooming meeting they are refined and acceptance criteria is added and planned estimates. And these at the bottom, these large ones move up, and will be eventually broken down into larger user stories with a little bit more definition. So our epics become containers of smaller user stories, which have more narrowly defined scope, more detail, and acceptance criteria telling the team when they're done. This forms a hierarchy. We look here in the epic of the user experience. We can see it has four children, four user stories, and each one is further refining this user story.   
*The presenter clicks the Parent value, US20: EPIC: User Experience.   
  
The user story US20: EPIC: User Experience page displays a menu item indicating four children stories. The presenter clicks the Children (4) link in the menu and the Children page displays the four rows of stories below the column headings: Rank, ID, Name, Iteration, State, Plan Est, Task Est, To Do, and Owner.*   
  
And so every epic is going to have an hierarchy of user stories eventually that further break it down. While these user stories need to be done in a single sprint, they need to get to that point. And these have estimates so they have already been broken down to that point. The epics can span multiple releases. And if you've a very large project with a large number of epics they may group them together into multiple themes which are described in even broader top level function. We go to the **Backlog** here. We can see that maybe we have this user experience and the customer portal and the plant database. Well that might be a theme that just groups together some basic functionality that needed to make this viable to even put out in a release. Themes add another layer to the hierarchy above the epic. And they're useful for long range planning by the stakeholders. Themes help break down very large projects into a roadmap and help with the release planning. However, the Scrum team very rarely deals with themes, as they're focused on breaking down these epics into multiple user stories which are small enough to estimate and complete within a single sprint.   
*The presenter selects the Backlog tab and highlights different epic parent values.*

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a02_it_enus/output/html/course_transcript.html#top)

The Story Board

Learning Objective

*After completing this topic, you should be able to*

* *describe the SCRUM Story Board*

**1. Describing the Scrum story board**

The story board must be posted in the public place available to anyone that wants to look at it. It can be kept in an electronic tool like this one or it can be as simple as just index cards and sticky notes placed on a whiteboard or a court board. What's important is for the team to continually update the story board during the sprint so it always reflects reality. Now the story board shows the progress of the single sprint, here we're in sprint 4 and we can see the headings here Defined, In-Progress, Completed, and Accepted and inside those...underneath each of those columns we have the user stories that we accepted during the sprint backlog meeting. So when the sprint starts all the user stories that we accepted are going to be under Defined. They're defined we're understand them, but no work has been done. As soon as the task inside the user story is taken by a team member to start working on, the user story moves to In-Progress, but it stays In-Progress until the acceptance criteria is met. When the acceptance criteria is met, it's moved into Complete, where it's ready for the product owner to review it. Now the product owner can either accept the user story into the product increment and then it goes into the sprint review and can be demonstrated, or they can say it's not done and not accepted. Now we can open up the stories to see which tasks are just Defined, or In-Progress, or Completed.   
*An example of a Backlog tabbed page is displayed where the Iteration Task Status for Sprint 4 is given. The Iteration Tracking Board displays four columns; Defined, In-Progress, Completed, and Accepted. Each has a user story listed, with the exception of the Defined column.  
  
The presenter opens the story in the In-Progress column and the Tasks page displays the list of tasks involved in the user story broken down in columns by ID, Name, Owner, State, Estimate (H), and To Do (H).*   
  
Now one gauge of a completed user story is that all the tasks are complete, but all the tasks do not have to be completed for the user story to be complete. We might have during the sprint backlog come up with some tasks that during development we noticed that we really didn't need to complete but they're in there. So the only thing that really needs to make a user story complete is this acceptance criteria; this is what's important, this is what the product owner and the team agreed to as saying the user story was done. As soon as all of the acceptance criteria is complete then the user story can be moved into the Complete column. Some teams use this method but they track the individual task instead of the user stories as it gives a lot more detail on what's being worked, these are called task boards and I can show you one here let's make it **View Full Screen**. Now you can see the task board has the same Defined, In–Progress, and Completed columns except over here in the first column is the user story. So this user story, Search for Plants by criteria on web site, has all these tasks. So it's a whole row. And each of these tasks show clearly at a glance which ones are Completed, which are In-Progress, and which are Defined and still ready to be worked.   
*The presenter hovers over the user story ID and the pop-up box displays the acceptance criteria for that story.  
  
The presenter navigates to the My Home tab and maximizes the Task Board example. The Task Board consists of tasks listed in Defined, In-Progress, and Completed columns. A separate section displays the user story details.*   
  
Now the team updates these task boards continually during the sprint but they should be done at least once a day after the daily Scrum meeting. In the daily Scrum, each team member is saying what they worked on yesterday, what they'll work on today and what they completed and so they're going to be updating this to reflect where they're at. So when a team member needs new work they'll come here to the Defined column to find a new task that needs to be done that no one is working on. They'll move it to the In-Progress. As they complete that task they can move it to the Completed. Now you might get in a situation where you've a GUI developer or a web developer that comes in and says, "Oh, I have time for new task but there is nothing here for me to develop. This is QA and Documentation." But remember the entire team either succeeds or fails. So any team member can take on these QA and documentation tasks to ensure that the entire team succeeded. Now story boards and task boards are essential tools for the Scrum team as it gives them the big picture of how the whole team is doing and helps them organize their work between the different team members. These tools also provide transparency in to how the team is progressing for the product owner, Scrum master, and the stakeholders.

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a02_it_enus/output/html/course_transcript.html#top)

The User Story Lifecycle

Learning Objective

*After completing this topic, you should be able to*

* *describe the lifecycle of a User Story*

**1. The lifecycle of a user story**

A user story starts outs with the customer or stakeholder having an idea of a feature they want in the final product. To capture this feature, they write a short, informal description of the feature in the form of a user story, which has the form, "as a user I want a <function> so that a <value>." Here we see an example of that, "As an online user" – that's the user – "I want to purchase plans if I'm on the plant web site" – that's the function – and why, "So I don't have to leave the house to buy plants" – that's the value. So we've a Who, and What, and Why. And anyone can write a user story. Anyone who has an idea of what they want in the final product; but this is very simple, it doesn't have much definition. Now the product owner collects all these user stories and puts them together in the product backlog with the highest priority user stories at the top and the lowest priority ones at the bottom.   
*An instance of team collaboration software is displayed. The interface contains four tabs: My Home, Backlog, Plan, and Track.  
  
An example of a user story page is displayed. It contains a General and Schedule section. The presenter focuses on the General section which includes the user story ID, name, tags, description, any attachments, and the name of the product owner.   
  
The description is: As an online user I want to purchase plants I find on the plant web site so I don't have to leave the house to buy plants.  
  
The presenter selects the Backlog tab and the Backlog tabbed page displays rows of user stories.*   
  
And the ones here at the bottom are quite large. They just have the user story template. And so these are really hard to estimate – they're quite large, they don't have a lot of detail. And so in the product grooming meeting these large stories are discussed and broken into smaller user stories. You can see here the epic shipping the purchase plants fee, we were looking at, that epic is been broken down into several user stories: shopping carts, transaction with debit cards, PayPal. And so now that they have been broken down, they're still too big really to estimate. But they're smaller than the epic and easier to discuss and the product owner now has several new user stories that can prioritize in the backlog. You notice over here if you look at the epics, the user stories don't really go. It isn't like you have one epic with all their user stories and an X. You can mix and match the user stories now between different epics depending on what the priority is of the product. And the product grooming meeting is also used to discuss the highest priority user stories. And then get them ready for the next sprint.   
*The presenter scrolls to the bottom of the Backlog page and highlights the rows of user stories where the Parent column for each of the user stories displays US12: EPIC: Purchase Plants.  
  
The user stories are secure buy traction with debit card, secure buy transaction with payPal, and secure buy transaction with credit card.  
  
The presenter scrolls through the rows where different epics are displayed in the Parent column. An example of a user story and epic which doesn't match is the user story named Store customer address with the Parent column value as US19: EPIC: CRM Initiative.*   
  
Here at the top are, our highest priority user stories. The user stories that we had down here, that were still quite big, are split even further into small feature slices that describe a small, focused, and easy to understand feature that we want in the product. Acceptance criteria is added so that we know when we're done, how we're going to test this. How do we know when we've completed this feature. And the team also gives an estimate of effort, this is a relative estimate. So now that we have all these user stories ready for the sprint, we're ready for the sprint planning meeting. The sprint planning meeting is where the team accepts the highest priority user stories to be included in the next sprint. Now these user stories are already well understood, they have acceptance criteria, they have a level of estimate; the team understands them. So all they have to do is decide which user stories that they're going to accept into the sprint. And the level of estimate helps them decide how much to accept.   
*The presenter hovers over a user story. A pop-up box displays the description and acceptance criteria of that story.  
  
The presenter focuses on the columns of the Backlog page: Rank, ID, Name, Plan Estimate, Priority, Owner, and Parent. The plan estimate for the first user story is 2.00.  
  
The presenter selects the Plan tab and the Plan page displays the Iteration Planning Board which contains the Backlog, Sprint 4, Sprint 5, and Sprint 6 columns. All the column are populate with user stories with the exception of Sprint 6.   
  
The presenter drags user stories from the Backlog column and drops them in the Sprint 6 column.*   
  
So once the team has accepted their user stories, at this point they know what they need to do, but they haven't defined how they will do it. The last part of the sprint planning meeting is spent breaking down the user stories into small tasks with time estimates that they can use, in hours. So here we have with this user story, all the tasks that were broken out and the estimate. And once all these tasks are done the acceptance criteria for this user story will be done. So the task breakdown is the team's plan on how this user story will be developed to enable them to satisfy the acceptance criteria. So then during the sprint the user stories are tracked. They start out in the Defined column and then when any of their tasks go from Defined...   
*The presenter navigates to Sprint 4 and clicks the Tasks icon of one of the stories in the sprint. The Tasks page displays rows of tasks broken down into columns by ID, name, owner, state, and estimated hours.  
  
The presenter selects the Track tab and the Track tabbed page displays the Iteration Tracking Board for Sprint 4. The Iteration Tracking board displays stories listed in the various columns which are Define, In-Progress, Completed, and Accepted.*   
  
...to In-Progress, the entire user story moves to In-Progress. The team tracks these tasks. They start as Defined when the team grabs one to start working on, it goes to In- Progress. When they complete it, the task goes to Complete. When all the tasks are Completed or when the acceptance criteria is met, then the user story gets moved into the Completed column. Now the user stories is at a crossroads, it's up to the product owner to determine which path it'll take. If the product owner decides the acceptance criteria is met, the user story is done and is moved into the Accepted column as this one was. The features demonstrated the next sprint review and the team adds the user story points to their velocity. At this point the user story is done, its features have been implemented and its lifecycle is over. However if the acceptance criteria is not met the user story is not accepted instead it needs to be return back to the product backlog and prioritized with the other user stories. It's not demonstrated at the sprint review and the team does not get the user story points added to their velocity. Rejected user stories are a form of technical debt. These stories need to be completed in future sprints which takes away from the time the team can work on new functionality.   
*The presenter clicks the Tasks icon on a user story in the In-Progress column and the Tasks page displays rows of tasks in different states such as Defined, Completed, and In-Progress. The tasks are also listed by team and the estimated hours of the task and how many hours of each task is left.*

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a02_it_enus/output/html/course_transcript.html#top)

User Story Breakdown

Learning Objective

*After completing this topic, you should be able to*

* *break down an EPIC into component User Stories*

**1. Breaking down an epic**

Here we have a very large epic. It's purchasing plants; you can see, "As an online user I want to purchase plants I find on our plant web site so I don't have to leave the house to buy plants." Now when you think about this – how what it takes to purchase something on a web site, the shopping cart, the session management, the getting the address information in the freight, and then actually doing the transaction – this is a very, very large story. It's going to have to be split into multiple user stories. So how do you go about that? Well most technical teams are used to breaking down work into functional areas and technical components. Think of an architectural diagram with the network layer, and then a persistence layer, and then application layer with your logic, and then of course you've your presentation user interface layer. We could break this user story down in that typical way. We would have a user story for getting the different web pages design, for the shopping cart, gathering shipping information, selecting payment confirmation of that transaction. Another user story could handle communication with the different credit and debit card vendors. The under the cover is networking and B2B communication.   
*An example of a user story page is displayed. The presenter focuses on the description, which is As an online user I want to purchase plants I find on the plant web site so I don't  have to leave the house to buy plants.*   
  
Another user story could tackle the session management logic and we'll probably need a user story to handle persisting the transaction and the customer information. A schema change in our database for sure. But did you notice, these user stories I talked about, that focus on the horizontal slices of functionality will not deliver value on their own. All of those user stories will need to be implemented and integrated together before the layers work together to show a feature that can be demonstrated to the end user. Splitting user stories along architectural or technology lines creates user stories with no value for the end user and this violates the Agile principle of continuous delivery of valuable software. Instead the team needs to break user stories into small incremental feature slices. Let's look at the children that we have for this user story. Here we have shopping carts to buy plants,gathering customer shipping information,freight selection,a secure buy transaction with a debit card,PayPal,and credit card.   
*The presenter clicks the Children link in the navigation pane. The Children page displays rows of user stories broken down by ID, name, iteration, state, estimates, and owner.*   
  
Not only do these user stories deliver demonstratable features of value to the customer. You'll notice each slice of functionality holds a portion of each of the horizontal layers. The shopping cart to buy plants, that's going to have a presentation; you have to have the web page; you're going to have to have the session management to be able to hold information for that user session; and you have to know when this user session is terminated. You're going to have to have interaction with the database because you're going to have to show the plants so they can select them, and you're going to have to grab that information from the database. So you have persistence and networking and logic. Down here with these transactions, not only are you going to have to have the presentation but you're going to have to have the logic down here with this credit card, you're probably going to break that down into multiple user stories, one for each different type of credit card. There's the B2B transaction. There's holding that information in the persistence, in the database, in the transaction record.   
  
And then Freight selection...well, you not only have to have the presentation, the GUI, to allow the customer to select a freight. Then you're going to need the interaction not only with the US mail but also with the other freight vendors. So that's going to have some B2B communication as well and the logic to add the different prices of that freight selection and to present it to the user. So each of these user stories delivers value and has a slice of the infrastructure that they're working on and delivering with that value to show a demonstratable feature. Splitting user stories in this way creates narrow, high value features from our large epic. And making each slice valuable to the user, supports Agile's "pay as you go" attitude toward technical infrastructure.

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a02_it_enus/output/html/course_transcript.html#top)

Heuristics for Splitting User Stories

Learning Objective

*After completing this topic, you should be able to*

* *split User Stories into small slices using heuristics*

**1. Splitting user stories using heuristics**

We can see three of our past sprints here. Since our team was new to SCRUM, they did some things right and made some mistakes in splitting their user stories as well. Let's run through some of the heuristics to see how they did. We will want to ask some questions. First is, anything that we were doing follow CRUD, which is an acronym for Created, Read, Updated, and Deleted. If so, we can put each of these operations in a different user story. Like here in Sprint 3, we had our "Administer can View plant information in the database" and the "Administer can Delete plant records in the database". This is probably a user story for modifying the information and another one for inserting it as well. Then we can ask is acceptance criteria very large or unfocused, representing more than a single slice of functionality? Here in our user story that's still in the backlog we could see the acceptance criteria has CRM Initiative, Security, User Portal Experience, it's very large. This is unfocused and it's not focusing on a single slice of functionality or a single feature. This is going to need to be split.   
*The Plan tabbed page displays an Iteration Planning Board. Backlog, Sprint 3, Sprint 4, and Sprint 5 columns are shown with user stories in each.  
  
The presenter highlights the Sprint 3 column which contains two stories. They are Administrator can View plant information in the database and Administrator can Delete plant records in the database.  
  
The presenter hovers over a user story ID in the Backlog column and the pop-up box displays extensive information in different sections, namely; CRM Initiative, Security, and User Portal Experience.*   
  
And you can ask, Does a user story have a happy path, the one that just works? If so, we can do that first then add the variations, error conditions, and complex interactions in other user stories. Now both the happy path and the error path user stories will need to be done for shippable functionality, but they will be smaller, more manageable stories if implemented separately. And then you can focus on the UI, it's probably going to have many complex features but they can be added one at a time with the functionality underneath them in their own user story. And if we look here at Sprint 5, we have, "As an online user, I want to be able to display detailed criteria about plants on the web site. So I know if they're right for me to buy for my garden." Now think about splitting the utility of the feature or just getting it done into a different story than the usability of the feature, where it's easy to use. Like here if we look at our acceptance criteria, we want, "Detailed plant Information page to show all the records on the plant including the image." The image can be zoomed and multiple image can be stored and shown as thumbnails which the customer can select. Well the utility are just getting it done, is just showing the detailed plant information where the usability making it nice and easy for the user is zooming the image and having multiple images they can select.   
*The presenter hovers over the story ID of the user story, Display plant information on web site, in Sprint 5. The pop-up box displays the information about the story: As an online user I want to be able to display detailed criteria about plants on the web site so I know if they're right for me to buy for my garden. The acceptance criteria is displayed plant information, detailed plant information page will show all records  on the plant including the image, the image may be zoomed, and multiple images may be stored and shown as thumbnails which the customer can select.  
  
Unit tests written and passed. Fully tested. Fully documented with screen shots of the screen.*   
  
And so this could be separated into multiple user stories – one for the utility and then actually two for the usability. Both utility and usability stories deliver value to the end user and the team is going to get valuable feedback on the utility story ensuring it's what the customer wants before spending time making it easy to use. Also ask if multiple items are needed. If so, you can get one done first and then work on adding the others. If we look here at our "Search for plants by criteria on web site" user story, we see the customers can search by a number of different criteria: sun, shade, water need, soil, height, and bloom time. And then the second acceptance criteria is, "Customers can specify multiple search criteria in a single search." Well one user story could get just the simple search done with only one criteria, or a separate user story could do multiples.   
*The presenter hovers over the story ID of the user story, Search for Plants by criteria on web site. The presenter scrolls to the Acceptance Criteria section of the pop-box.*  
  
You also want to look to see if there are several different roles. Here we have an online user to be a customer but from Sprint 3 you can see we have administrators too. So we have different roles. If there are different roles with different authorization you want to do the simplest most restricted role first. You can also slice and dice the scope in different ways. This can be done by limiting input by the amount of data, or where the data comes from static or dynamic, or maybe the data can be accepted as is and validated in the later user story, or you can look at architecture scope which can be limited by stepping out components, or deferring capability like internationalization or support for other platforms. In this user story we could have hard-coded the plant information that was returned and added our database connectivity later. Also, algorithms can sometimes be broken down into separate pieces or constants can be used in the initial user story with real-time values being retrieved and added in separate user stories.   
*The presenter scrolls to the description: As an online user I want to search for plants that fit criteria so I can find plants to buy that will thrive in my garden.  
  
The presenter points out the user stories in the Sprint 3 column which have the Administer as the customer.*   
  
You also want to look through words that signify a compound task, words like "and," "or" are multiple complete sentences each with their own separate idea. Now keep in mind that overly complex user stories with large areas of uncertainty may need a SPIKE to investigate the area and come to an understanding that allows the team to split the story. Here we have the SPIKE when coming up with an "Error Reporting Design" or the user story could be split into SPIKE to understand the area and design a solution and another user story to implement the solution in a future sprint.   
*The presenter points out the Sprint 4 column includes SPIKE: Error Reporting Design, while the Sprint 5 column has a user story named Implement Error Reporting design.*

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a02_it_enus/output/html/course_transcript.html#top)

Business and Data Patterns for Splitting User Stories

Learning Objective

*After completing this topic, you should be able to*

* *split large User Stories into small slices using business data patterns*

**1. Using business data patterns**

This is one of the epics in our project, the one to "Purchase Plants." Our online user wants to purchase plants they find on the plant web site so they don't have to leave the home to buy their plants. Let's see how many patterns for splitting user stories we can find in this epic. First this pretty much says that you're going to have the workflow pattern. Think about how you purchase anything on a web site, there are very specific steps you go through. And the workflow pattern breaks down the process into those workflow steps. Let's look at our **Children** user stories to see how these works. First we have our shopping cart to buy plants, we have a Purchase Checkout, we Gather Customer Shipping Address, Freight selection, secure transaction, and Confirm Transaction. Whenever you have something with very definite workflow it's a very good way to breakdown your user stories.   
*An example of a story page called User Story US12: EPIC: Purchase Plants is displayed. The description is As an online user I want to purchase plants I find on the plant web site so I don't have to leave the house to buy plants.  
  
The presenter clicks the Children link in the navigation pane. The Children page lists rows of user stories under columns named Rank, ID, Name, Iteration, State, Plan Est, Task Est, To Do, and Owner.  
  
The user story names listed are Shopping cart to buy plants, Purchase Checkout, Gather Customer Shipping Address, Freight selection, Secure buy transaction with debit card, Secure buy transaction with PayPal, Secure buy transaction with credit card, and Confirm Transaction.*   
  
And we can see we have a business rule variation pattern here where the same feature can be done in various ways. That leads to a nice breakdown on the variation. We see payment on the Web here can be done with the debit card, PayPal, or credit card. This is the same thing; we're securing a buy transaction, we're just doing it in a variety of different ways. So there is a different user story for each of those variations. The Freight selections user story could be further broken down using the major effort pattern, where a major infrastructure piece needs to be implemented first, then the other feature slices are implemented on top of it. If you think about it there is several different freight options; US mail itself has several different classes. And then there is different freight vendors that we might want to give our customer the ability to select. Our first user story could contain the infrastructure needed for all the different freight options, the presentation, calculating final purchase price, and storing the freight selection in the database. This would also include the B2B component to get the rates from the single vendor.   
  
The following user stories would be much smaller with just their own B2B communication but using the infrastructure setup in the first user story with maybe some slide variations in the presentation so the customer could select them. Also look for patterns that involve variations in data. Once this entire epic is implemented and all these user stories are done, there may be a need for the pages to be internationalize to different languages. This is a variation in the data that is displayed. Each translation to a different language could be a different user story. Related is the low fidelity to high fidelity pattern where there maybe variations in the quality of the feature similar to the differences between a car radio and a high fidelity sound system. Consider the display for this epic: the user story to complete the feature could only work on monitors with at least the 1250x750 display. Future user stories could be added to an enable the GUI to work on smaller displays, like tablets and smartphones.   
  
Lastly be on the lookout for system quality patterns where the final product may require system metrics for features like the number of simultaneous registered users. You'll want to defer any system quality work to future user stories. Initially the website could be implemented as a single user application first to get the basic functionality done, with a different user story for making it handles multiple user simultaneously. Later other user stories could be added to scale the application to larger and larger numbers of simultaneous users as the business needs make it necessary.

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a02_it_enus/output/html/course_transcript.html#top)

Patterns for Splitting User Stories

Learning Objective

*After completing this topic, you should be able to*

* *split User Stories by identifying the simple and complex patterns*

**1. Splitting user stories using patterns**

Here we're looking at one of the epics in our project: the "Website Customer Personal Portal experience". As an online user our users wants to have their own personalized portal page so they can access the site more efficiently. Now this epic will show us some of the patterns we want to be on the lookout for. These patterns will help us breakout the simple from the complex as we split our user stories. So let's look at the **Children**. First we want to be on the lookout for operations. These can form patterns which can be used to split user stories. Here we have a very well-known CRUD pattern over here with our User Registration, User View registration, User Modify registration, and User Delete registration. CRUD stands for Create, Read, Update, and Delete. This is a very common and if you see it, it makes it very easy to breakout your user stories into the four CRUD acronyms. But that isn't all. You need to look for words like "Manage" or "Control" and these point to multiple operations. If we had a user story that said manage my account, you can start thinking of what could mean; it could mean register, edit the settings, edit your password, change your e-mail subscriptions, configure the portal, cancel a registration. These are all operations and they break out nicely into user stories.   
*The example User Story US30: EPIC: Web site Customer Personal Portal experience page displays. The description is As an online user I want to be able to have a personalized portal page so I can access the site more efficiently.  
  
The presenter clicks the Children link. The Children page lists rows of user stories below the columns named Rank, ID, Name, Iteration, State, Plan Est, Task Est, To Do, and Owner.  
  
The user stories listed are User Registration, User View Registration, User Modify registration, User Delete registration, Store customer registration information, User authentication with personal portal, Customize customer home page, and Registered Customer comments on plant information.*   
  
You also want to be on the lookout for the simple to complex pattern. This is evident when a team is asking questions like, what if X happens? Or if Y happens – it'll change the outcome. This indicates some dependencies on outside events which are making the user story complex. If we look down here at US29, Registered Customer comments for plant information. Well what if the customer is registered but not logged in? What if they're disruptive or abusive in their comments, should we limit their comment privileges? What if they want to edit or delete their comments, or can they comment on another user's comments? In this case, you'll want to use the Agile principle: do the simplest thing that could possibly work. First do the most basic user story where nothing is affected by X or Y. Then write user stories to handle each of the complex variations. Now US31 right above, calls for a customized user portal page, which says we have a user interface complexity pattern.   
  
The final UI does not have to be created at the beginning of the project. As feature slices are added, the UI will change to accommodate the new features and be demonstrated to the customer who will have suggestions and updates. So start with the simple UI and add to it with every user story. In this case, there are many, many ways you could customize this homepage, do the simplest and add them one at a time. Also in some cases like where the User Delete registration, the UI could be simplified by demonstrating the feature with the simple command line user interface and then adding the GUI presentation in a later story, or maybe with the user registration. This could be done with very simple widgets to demonstrate the layout and functionality, saving the polishing of the UI style with custom widgets for a later user story. Be aware that if there is any features that need batch, the feature could be implemented in a batch user story first and then the user interface for real-time user direction of the feature could be done later. Lastly, if use-cases exist they were probably created to understand the more complex interactions of this feature. This means someone has already done a breakdown at individual scenarios and that breakdown should work nicely as a guideline to breaking down the user stories.

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a02_it_enus/output/html/course_transcript.html#top)

User Story Tasks

Learning Objective

*After completing this topic, you should be able to*

* *use tasks to measure a team's progress and keep the entire team on track*

**1. Using tasks to measure progress**

Here is our tasks list for our user story, the "Administer can View plant information in the database." The tasks here need to be small, simple, and be associated with an estimate. This should be quick to implement, normally taking from 4 to 12 hours to complete. Anything less than 4 hours is too small a granularity, you're wasting time breaking down a user story that far. And normally a team member should be able to complete a task in about 5 hours. Small stories like this mean each team member should have a new status each day in the Scrum meeting, when they talk about what they finished and what they will work on that day. Larger task requiring several days of work should be broken down into multiple tasks. Large tasks often hold several smaller tasks and bundling them together hinders the team from working in parallel. If we look down here at TA44, we see this Tech document task is going to take 24 hours to complete.   
*An example of a Tasks page displays for User Story US9: Administer can View plant information in the database. Tasks are listed by ID, name, iteration, state, estimate and owner.   
  
The presenter scrolls to the task Tech: document user interaction to search and view records. The estimate is 24 hours.*   
  
Now if our tech writer was out sick or got in the bind and they had already started this, there is no way anyone is going to be able to help them with this task. But we see there are several web pages up here. If this had been broken out into documenting each of these web pages, this would be smaller tasks and the other team members could have pick them up and help the tech writer out. Also large coding tasks are more prone to breaking existing code in the repository when checked in. This is because the task was developed on a code base that was pulled days ago. Small task enabling team members to check their code into the software repository often lead to continual integration. Checking in completed code and pulling new code for new tasks happens daily if task was small enough to be completed in a day. Tasks should be quick to create and record in the sprint backlog. They are not user stories and should not be associated with acceptance criteria, copious notes, or other agreements. Tasks represent one line on a To Do List, the simple steps that need to be completed for the team to meet the acceptance criteria of the associated user story. And the focus should always be on the user story, since if it's not done no value will be delivered to the end user.   
*The presenter points out several DEV tasks in the rows above the Tech user story.*   
  
Task by themselves deliver no value. They are just a means to an end of delivering value through their user story. Task should be independent so the team can work in parallel on the same user story. We see this testing effort has five different tasks. Each testing a different area of the user story. These are independent of each other. And this would allow five different team members to work through the test plan in parallel, significantly reducing the time it would take to complete the user story if testing got in a bind. And those tasks collectively tells the story of how the user story will be implemented. Tasks should be written from a functional view point, saying what must be done, not how to do it. We look at Task 41 here, "Dev: Accept search criteria and transform into SQL to retrieve results from DB". That says what the developer needs to do. It doesn't say how to write the SQL, how to connect to the database. The task owner will figure all that out themselves. And since this task says what will be accomplished, it's easy to determine when it's done and can be move to Complete.   
*The presenter indicates five rows of QA tasks for the testing efforts.*   
  
Senses Sprint is a miniature version of the software lifecycle, you would expect all the phases of analysis, design, implementation, and test to be represented in the task of every user story. The task list should include both development tasks like our TA41 here, and engineering tasks like TA43,Designing thedetailed interface between components. Or TA44 – documenting how the user will interact with the application. Non-functional tasks like logging, security, scaling, and refactoring are considered part of the task unless they're large enough effort that they change the tasks estimate then the task should be split to track the new work that was discovered. And though there is no unit test task in this task list, if we go back to our user story we see they're explicitly called out for in our acceptance criteria. So development task cannot be considered complete until the unit tests are written or the user story will never be done.   
*The presenter clicks the Details for US9 link in the navigation pane. The user story information displays and the presenter focuses on the Acceptance Criteria section.  
  
The acceptance criteria includes an Administrator Utility part which lists the following: stand alone GUI utility with authentication to gain access, utility configurable to point to plant database; and utility tied to Db user id with authorization to insert, update, delete, and view records.  
  
The Search and View Plant Records part states: Page to allow enter of search criteria to include plant name, plant type, plant zone, or all plants in database. It also involves retrieving records from the database that match search criteria and displays them in alphabetical order, as well as allowing the user to select one record from a returned list to view all the details of a record.  
  
Unit tests for each developed component. Fully documented. Fully QA tested.*

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a02_it_enus/output/html/course_transcript.html#top)

SMART Tasks

Learning Objective

*After completing this topic, you should be able to*

* *use the SMART tasks to break down tasks into manageable portions of the User Story*

**1. Using SMART tasks**

This is one of the team's early user stories, which added the base infrastructure for creating the initial web site. Even so, it's written to show value from the user. "As a user I want an online web site about plants so I can gather information about plants from my home," and it has a task breakdown. Let's go through the SMART acronym to see how the team did on this task breakdown. First there is specific, the task has to describe one thing with no overlap with other tasks. Anyone reading the task should be able to understand what it will accomplish, what is needed to complete the task, and how it relates to the completion of the user story. If you look, we did pretty good. These are pretty easy to understand:work with corporate to select acceptable domain name, register domain name with IP address, develop test plan for user story.   
*The User Story US37: Website for Online Users page displays. The description is As a user I want an online web site about plants so I can gather information about plants from my home.  
  
The presenter clicks the Tasks link. The Tasks page displays rows of tasks under the headings: ID, Name, Iteration, State, Estimate, To Do, and Owner.  
  
The presenter focuses on the first few rows of tasks: DEV: work with IT to get hosted web site on company web server, DEV: work with corporate to select acceptable domain name, DEV: Register domain name with hosted IP address, and QA: Develop test plans for user story.*   
  
Well let's look back at that one. Remember any team member can take on any task. Well we don't want to put a lot of detail on our tasks. It maybe hard for our non-QA team member to pick this task up if needed. There are three QA tasks associated with actually doing the test. They're more specific. Perhaps the test appear for writing down the test plans with the user story should include in its title these three component names or maybe there should be a different task for each of the tests that are going to happen to make it more specific. Now next is Measurable. A task needs to be clear when the task has been completed. It must have a well-defined end. Looking at TA104 here, we're going to QA: Test the homepage rendering on mobile devices, smartphones, tablets. Well there is so many mobile devices and smartphones. When will testing be done on enough of them for this task to be complete?   
*The presenter points out QA tasks. They are QA: test access to hosted web site from LAN and from Internet, QA: test web page rendering on workstations with different resolutions, QA: Test home page rendering on mobile devices, smart phone and tablet, and QA: Develop test plans for user story.*   
  
The task should list the mobile devices which will be tested, so it's clear when the task is complete. And also the team must decide if development processes are included in the task's measure are done. Processes like unit test, refactoring code, getting the Product Owner to approve these GUI wireframes. If the team decides on processes which are not followed, they can be added as separate tasks, though doing so would break the continuity of the team member owning the task and its associated processes. Achievable says the team member that takes on ownership of the task should have the expectation of being able to achieve or complete the task. The resources must be available and the skill set of the team member must be appropriate. If the task is not realistically achievable, it needs to be revised, reassigned, or the blockage – like unavailable resources – needs to be resolved.   
*The presenter points out the GUI tasks: GUI: develop wireframe of minimal home page; GUI: work up several templates and color schemes for PO review and selection; and GUI: create web page templates to enforce selected look and feel of site.*   
  
Relevant means the completion of the task is directly related to satisfying the acceptance criteria of our user story. The user story says what needs to be implemented. The task list tells the story of how the team will do it. Now the tasks help the team organize their work and monitor their progress. But all the work in the task should link to some value to the end user of the user story. And lastly we have time-boxed, which just means limited to a specific duration of time. In our team we're using an estimate of hours, but the task should all be completed within the time-box of the sprint. These means each team member needs to be trusted to ask for help if they become stuck or the task takes more effort than given in the estimate. The team is responsible for organizing their work, which includes splitting large task, changing ownership of a task, or anything else that enables the task to be completed to move the user story closer to done.

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a02_it_enus/output/html/course_transcript.html#top)

INVEST User Story Guidelines

Learning Objective

*After completing this topic, you should be able to*

* *use INVEST guidelines to develop high-quality User Stories*

**1. Using INVEST guidelines**

The product backlog holds all the work not yet completed in the project. This includes not only the customer's requirements and features but also defects, technical architecture tasks, refactoring to reduce technical debt, and SPIKES to investigate areas of uncertainty. By far, the most important product backlog items are the requirements and features the customer wants in the final product. These are entered as user stories. Not all user stories are created equally though. High quality user stories with good descriptions and clear acceptance criteria reduce uncertainly and lead to a high quality product and satisfied customers. The INVEST guidelines are a way to gauge the quality of the user story. Good user stories need to be independent. The product backlog is prioritized and a user story can be moved up or down in the product backlog, relative to its priority. If user stories are dependent on each other, the dependencies will determine when the user story is worked rather than its priority. If the team finds two user stories that are closely dependent on each other, they should consider combining the stories if possible.   
*Heading: INVEST Guidelines.  
  
The acronym, INVEST, is displayed with definitions for each letter. I is for Independent, N is for Negotiable, V is for Valuable, E is for Estimable, S is for Small, and T is for Testable.  
  
A further breakdown is given for each definition. Independent includes self-contained and has no dependencies*   
  
High quality user stories are negotiable, user stories should be considered malleable while they were still in the product backlog. They can be changed, rewritten or the user story can even be deleted depending on the state of the project and the business needs at that time. During the product grooming meeting, the highest priority user stories are examined by the team and product owner to gain an understanding of what is required into assigned acceptance criteria and estimate a rough level of effort. During these discussions user stories are often rewritten as technical or business needs are uncovered. A good user story should capture only the essence of the customer's requirement. The detail should be discovered by the team during development and verified by the customer at the sprint review meeting. A user story must define something that is of value to the end user. This could include a feature or performance requirement the user has asked for. And it's something that the team can demonstrate in working software during the sprint review meeting. A user story must be estimable. The features are requirements described in the user story, must be clearly understood by the product owner so they can answer the team's question, allowing the team to provide an estimate of effort.   
*Heading: INVEST Guidelines.  
  
Negotiable means modifiable until included in a sprint.  
  
Valuable: Adds user value to the product increment.  
  
Estimable means the team can estimate the level of effort required.*   
  
If the user story is not clearly defined or scoped the team will not be able to estimate the level of effort and it will never be implemented. When a user story cannot be estimated, it's a sign that the story is too big and should be split or that the area is not well understood and research must be done, usually in the form of a SPIKE. User stories need to be small or sized appropriately. The team should be able to complete a user story in a few hours to a few days. User stories that cannot be completed in a single sprint should be split into multiple stories. User stories that spend many weeks or even months are too big and complex for the team to provide an estimate on level of effort with any degree of certainty. These large user stories are considered epics and should be split into several user stories that are small enough to understand and estimate. And a user story has to be testable. An important portion of a user story is the acceptance criteria. The acceptance criteria is agreed upon by the product owner, team, and Scrum master, and is used to determine when the team is done implementing the user story. The acceptance criteria must include a definitive description of what is considered "Done" that can be validated during testing.   
*Heading: INVEST Guidelines.  
  
Small means the effort can be completed in a single Sprint. Testable means providing information needed for test development*

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a02_it_enus/output/html/course_transcript.html#top)

Accumulation of Technical Debt

Learning Objective

*After completing this topic, you should be able to*

* *describe some of the ways technical debt accumulates and makes a product unstable*

**1. Debt accumulation**

Technical debt is work that is not completed. And normally not very visible so it's not being worked. It accumulates in the product itself and grows into an area of code that is brittle, hard to work on, and not understood by many on the team. As technical debt grows, it can make the product unstable and make it difficult to add new functionality. Let's take a look at some of the ways technical debt accumulates. Our project hasn't been going on very long. We're only in our fourth sprint, we've completed three sprints but already we're taking on some technical debt. Sprint 1 here, if you look at this user story it's not finished. 83% of the tasks were complete but we still have 8 hours left. And if we look at those tasks, we can see what didn't get complete was testing. We created our database schema, we developed our test plan, but we never tested to make sure that the data that was added to the database was inserted correctly. Here in Sprint 2 we also have a user story that wasn't complete, it's only 67% complete. And here if we look at it we can see okay, again testing got the short end of the stick. They didn't get to complete ensuring that the plant records displayed on the GUI match the database records.   
*The Plan tabbed page displays an Iteration Planning board where Backlog, Sprint 1, Sprint 2 and Sprint 3 columns display user stories in each.  
  
One of the user stories in Sprint 1 has an Unfinished status: Add plant information to the database.  
  
The presenter clicks the Tasks icon and the Tasks page displays. It shows that all the tasks were completed except for the last one which was QA: Test data correctly inserted into database. This task has 8 To Do hours left.  
  
The presenter clicks the Tasks icon of a user story – which shows 67% complete – in the Sprint 2 column. The Tasks page displays where one of the tasks, QA: ensure plant records displayed on GUI match DB records, show an In-Progress status.*   
  
So user stories that are accepted by the product owner before they're done. There are other things that may not be properly tested. As in this case, they might not be documented they might be missing functionality that's required for modifications, planned in future sprints. In this case, here in Sprint 3, how can we know we're viewing plant information and deleting plant information correctly if we haven't even tested to make sure that the plant information that is displayed on the GUI is correct. We can also see that we've a few bugs here. Now if we go to the **Backlog** in our project, we're putting our defects in the product backlog. And the product owner is getting to prioritize that with the other things in the product backlog. But we can see the product owner is putting user stories ahead of defects, even this defect. The plants that returned from a search, if you removed one from that search it was removed from that database. Well that's a high attention, that wasn't supposed to happen but it's still behind two of the user stories. And we have other defects down here that are behind other user stories. So our product owner is digging a technical debt hole for the team here. Also if you look you can see these two, there is two defects on two different user stories US9 and US2, and they both had to do with displaying information that's not scaling for the mobile device.   
*The presenter highlights the user stories in the Sprint 3 column: Administer can View plant information in the database and Administer can Delete plant records in the database.  
  
The presenter points out bug icons on a few of the user stories.  
  
The presenter selects the Backlog tabbed page and points out the first three rows of user stories. The defect, DE7, Error found in US2: Plants removed from results list are removed from database, is ranked at three with a high attention priority.  
  
The user stories ranked above DE7 are US1 – Save Displayed Plant Information on web site and US27 – User registration with plan estimates of 2.00 and 5.00 respectively.  
  
The presenter goes on to point out more defects listed after other user stories further down the list.  
  
DE6 – Error found in US9: Displayed information does not scale for mobile devices and DE4 – Error found in US2: Search criteria does not scale for mobile devices show no priority value in the Priority column.*   
  
Now this is going to continue to come up on every new screen we develop. So working it needs to be a higher priority in the list, probably with a SPIKE. So that we can come up with a way that we can effectively deal with mobile devices so we scale correctly. Now there's other ways to accumulate technical debt. The product owner could be pushing the team to take on too much work. This could cause them to work long hours and make mistakes or write sloppy code. Our areas needing research, like this mobile device area, that's not worked in a SPIKE. So the issue just keeps growing, as more screens are added. Ideally the team should strive to not accumulate any technical debt, or no more than they have to. Because if they don't have time to fix a problem when it's encountered, they're not going to have time to fix it later while trying to add new functionality to the product increment. And the ninth Agile guiding principle is continuous attention to technical excellence and good design enhance its agility. While we've seen technical debt is a drag on agility. Technical debt just needs to be first identified. Then it needs to be given priority so it can be taken out of the product before it ruins the team's agility and the stability of the product.

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a02_it_enus/output/html/course_transcript.html#top)

Identifying and Reducing Technical Debt

Learning Objective

*After completing this topic, you should be able to*

* *describe how to bring technical debt under control*

**1. Controlling technical debt**

The Scrum artifacts will give us an easy way to identify some of the technical debt we're accumulating. And the Scrum process will give us a way to get the technical debt under control and most of it out of our product and our project. One thing we can look at is the unfinished user stories. We see here in Sprint 1 we have some tasks that were not complete, we're only 83% complete in our task and in Sprint 2 we're only 67% complete on this user story. Now having task that aren't complete is not automatically a problem. During the sprint planning we could have come up with some tasks that we just didn't need after we got into development. However, if we look at the tasks here that were not completed we can see it with testing. Now that's a problem, because you don't want to cut your testing because it's going to lead to an unstable product, you're going to have bugs in there, you're going to have things you didn't find. In the test that was not done, was ensuring the plant records displayed on the GUI match the database records.   
*The Plan tabbed page displays an Iteration Planning Board where Backlog, Sprint1, Sprint 2, and Sprint 3 columns contain various user stories.   
  
The presenter clicks the Tasks icon of the user story, US8 – which shows 67% completion – in the Sprint 2 column.  
  
The Tasks page lists three tasks. Two are complete and one – QA: ensure plant records displayed on GUI match DB records – is in an In-Progress state.*   
  
Well if we look at our acceptance criteria, we definitely have an acceptance criteria to verify the data match to input file data. So this means that the product owner is allowing the team to cut corners on the acceptance criteria. And since these user stories weren't back in the product backlog that means that the product owner accepted them into the product increment, even though they were not done because their acceptance criteria was no met. So allowing user stories into the product increment that really are not done is going to built up your technical debt. The other thing you want to look at is your defect trends. Now if your defects are going up over time and you're not really working them, you're accumulating technical debt. And your product increment is becoming more and more buggy and unstable. Now our product defects are kept on a **Backlog** and the product owner gets to prioritize them. And here's the rank, the highest priority on top. And you can see he has a couple of user stories and then one defect, another user story and then the other defects are down here. New functionality is given a higher priority, which means our product increment technical debt is growing all the time.   
*The presenter hovers over the user story ID, US8. The pop-up box displays the description and acceptance criteria. The presenter focuses on the acceptance criteria: Able to import data from an excel spreadsheet into the database, verify data matches input file data, and verify data remains over a database restart.  
  
The presenter navigates to My Dashboard on the My Home page and points out the Defect Trend chart, which shows that the defects are increasing over time.  
  
The presenter navigates to the Backlog tabbed page and highlights the ranks of the user stories and defects listed. The defects are not ranked in between the user stories.*   
  
Now though they're lower priority on the product backlog, remember that the team determines what work they will be including into their own sprint backlog in the sprint planning meeting. So the team needs to ensure some of these defects are moved into the sprint backlog with each sprint. So they will be worked and won't be adding to their technical debt. There's two different defects from two different user stories both dealing with the web pages not scaling for a mobile device. Okay, this is getting to be a trend and the more screens we add to our web site, the more technical debt we are accumulating because we know we're going to have to go back there and fix them, so they'll all be able to scale. And since this has happened twice we probably don't really know, the team doesn't really know how to do it yet. So this is a good place to get a SPIKE. You do a SPIKE, put it into the backlog. It should probably be pretty high up because if we start doing these other user stories, user registration, storing customer registration these kind of things, they're going to have screens.To show you how that looks...because we did it correctly for our Sprint 4, 3 and 4...   
*The presenter points out two defects ranked 5 and 6. They are DE5 – Error found in US9: Displayed information does not scale for mobile devices and DE6 – Error found in US2: Search criteria does not scale for mobile devices. The defects have no priority value in the Priority column.*   
  
...here we have a story that was blocked, we didn't complete it, And if we look at the tasks, we can see the one that isn't complete is here. We needed to return an error from the database and we weren't sure how to handle that error. We needed a SPIKE to determine a strategy, our error logging strategy for the entire system. And we did that here in Sprint 4. We had a SPIKE and we time-boxed it. We said who is going to work on it, and for how long, and we had questions we needed to ask and what the outcome was. And we completed that, we can work that in a future sprint, actually put our error logging in. Now if you have taken on technical debt already, as it grows it'll be apparent that there is an issue in both the sprint planning meeting and the sprint retrospective meeting. You'll start to see patterns of difficulty. Stories from certain areas, like here, will have large estimates on them, the teams giving them large estimates even though the Product Owner may see them as simple. Our user stories from some area will consistently be delayed or difficult for the team to complete. Here we see we are only 12% done with this user story, while the other user stories are complete.   
*The presenter opens the Plan tabbed page and in the Sprint 3 column, the user story is US10: Administer can Delete plant records in the database.  
  
The presenter clicks the Tasks icon and the Tasks page displays a list of tasks for the user story. She highlights the task, TA79 – DEV: Return confirmation or error from database delete transaction to web page. This task is in the In-Progress state with an estimate of 4 hours and the To Do column shows 4 hours as well.  
  
The presenter focuses on the user story, US38 – SPIKE: Error Reporting Design, in the Sprint 4 column of the Backlog page.  
  
The presenter hovers over the user story ID – US38. The pop-up box displays the details: Two developers 1 day each to investigate reporting errors on the web site. How much information? How should it be displayed to the user? Where on the web site will it be displayed? Do we need to log errors fro debugging or reports? (May need a spike on logging in general) I18N concerns. Outcome: holistic strategy and system error handling design.  
  
The presenter points out the user story, US2: Search for Plants by criteria on website, which has a 12% completion indicator, an estimate of six, and a bug indicator with a value of two.*   
  
Another indicator is a high defect count associated with certain user stories. So this might be an area that we might want to look at and see what kind of technical debt we've taken on in this area of this user story. Now during the retrospective, the team may consistently refer to difficulties with the technology or current architecture or design. That's another indicator of technical debt. Or the team could just state that there is a problem and it has to be fixed, otherwise new functionality cannot be easily added. More subtly you may notice the team seems worried about creeping technical issues. Now in Scrum, one way to reduce accumulation of technical debt is the Done criteria. Every user story that's accepted in the sprint backlog needs to have clear acceptance criteria. And here we see at the bottom it says it must be fully tested, unit tested and fully documented. So this user story should not be accepted until all of these criteria are met, including these at the bottom.   
*The presenter hovers over the user story ID, US2, and the pop-up box displays the description and acceptance criteria for the user story. The presenter scrolls to the bottom of the box where it states: fully tested, unit test created, fully documented.*   
  
The Scrum master needs to enforce the Scrum process of not allowing a user story to be declared done until the acceptance criteria has been fully satisfied. User stories that are not done at the end of the sprint should be placed back on the backlog. So these two unfinished user stories should have gone back on the backlog and they should have been completed, worked in another sprint and completed before they could be demonstrated as potentially shippable functionality at the sprint review meeting. Still you're not going to get rid of technical debt completely. Every defect found is a form of technical debt. And since technical debt is an impediment to the team's progress the Scrum master, especially, has a responsibility to watch for signs of it creeping into the product. However, since technical debt leads to an unstable product and lower team velocity, all committed roles – the Product Owner, the Scrum master, and the team – need to be on the look out for technical debt as it's arising and come up with ways to work it and eliminate it as it's identified.

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a02_it_enus/output/html/course_transcript.html#top)

Scrum and Deferring Detailed Specifications

Learning Objective

*After completing this topic, you should be able to*

* *describe the advantages associated with deferring detailed specifications in Scrum*

**1. Deferring specifications in Scrum**

In well-known waterfall processes, specifications of all the items to work in the project is done upfront. And the development team is given a detailed requirements document, fully describing what must be built. In Scrum, defining details is differed until the feature's about to be worked. This can make some people uncomfortable. How can you build something if you don't know exactly what you're going to build? But there are some really good reasons why Scrum defers defining details for as long as possible. Putting off defining detail until the last possible minute is a good thing. Here's our product backlog. It's prioritized with the highest priority user stories at the top. These are the ones...they're about to be pulled into a sprint. And at the bottom of our product backlog are the lowest priority user stories. Now there's no reason for the team to spend time defining details for these epics at the bottom of our product backlog.   
*The Backlog tabbed page displays rows of user stories with columns for plan estimates, priority indicators, owners, and parents.  
  
The user stories at the top have priority values and estimates whereas the epics at the bottom of the page – EPIC: user training and EPIC: Data Mining Synchronization – have no priority values or estimates.*   
  
As the items at the bottom of the product backlog may change as the project progresses and may never be worked at all. And when these low priority epics and user stories rise higher up in the product backlog, more will be known about the entire product by then, making the stories easier to understand and breakdown into smaller user stories with more details. Also as the project progresses, the intent of the entire epic may change as the customer sees functionality added to the product increment in each sprint. By the time, this Data Mining epic rises up high enough in the product backlog to be worked, it may change due to new technology, new information that should be mined, or the customer could decide they don't need this capability at all. Anyone can add items to the product backlog at any time, so the entire scope of the project can change if these items are set at a higher priority than the items in the original scope.   
  
It's better to defer adding details to any low priority user story or epic until the priority places them higher in the product backlog. And the team should never estimate an epic; they are too large to fully understand with too many unknowns. When a team agrees to estimate an epic they are giving the product owner a false sense of security, that the epic is known and the estimate is valid. The team's estimate may then be used to plan releases and for budget forecasts. Once the budget is established, the team's estimate on the epic is solidified as truth, and the team will be expected to meet the schedule the budget and estimate define. However, the team's estimate was just a shot in the dark, as it was not based on any concrete knowledge of the unknowns, the task breakdown, or the details of how the broad user story should be developed to match the product owner's and the customer's vision.   
  
As an epic or very large user story become a priority, they will naturally rise up higher in the product backlog. This is because the product owner is tasked with maintaining the product backlog, keeping it prioritized. Large user stories getting near the top of the product backlog will be discussed in the product grooming meeting, where they will be split into smaller user stories that are easier to understand. By the time the user story becomes a high priority at the top of the product backlog, it will be small enough to fit into a single sprint, and easy enough to understand that the team can give it a relative estimate. But this work of understanding the user story and detailing its specifications enough to give it an estimate is done just in time, deferred until the story is high priority, ready to be pulled into a sprint. By deferring any work on product backlog items as late as possible, the team is always focused on the highest priority features, the ones about to be implemented, and they don't waste any time understanding or estimating low priority features that will not be implemented any time soon.

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a02_it_enus/output/html/course_transcript.html#top)

Team Velocity

Learning Objective

*After completing this topic, you should be able to*

* *use the team's velocity to estimate their capacity in the next Sprint*

**1. Using team velocity to estimate capacity**

During the sprint planning meeting, the team must determine how much work it can take on in the next sprint. The team wants to maximize the number of features they can add to the product increment, while not taking on more PBIs than they can complete. Now estimating the amount of work that a team can do in a sprint is often difficult when the team is new. However, over time the team will become more proficient estimating their capacity. And one metric that can help is the team's past velocity, or the amount of work they have completed in the past sprints. PBIs are assigned a relative level of effort by the team in the product backlog grooming meeting. The velocity of the team is measured in the same units as these PBI estimates. In our project here we're using user story points. With 1 and 2 here, being small, 3 being medium, 4 is a larger effort, and 5 is considered an epic, very large. Now we can see in our last sprints here, the user stories that the team took on during those sprint planning meeting. The first sprint they took on seven points. They were a brand new team and they just kind of came up with a number. But then they used that the next time for the next sprint planning meeting for Sprint 2 and they took on eight story points and in the last sprint they took on seven story points. So they have been pretty consistent over each sprint on the number of story points they came up with.   
*An Iteration Planning Board displays Backlog, Sprint 1, Sprint 2 and Sprint 3 columns containing various user stories.*   
  
But story points are not added to the team's velocity unless they complete the user story. If the user story is not completed, the story points for that story are not added to the team's velocity. So let's see how the team really did on each of these sprints. If we go to the tracking and let's look at **Sprint 1**. We can see here, they took on seven story points, but they only completed four of them. For whatever reason, even though the tasks were all done here, on this user story the acceptance criteria must not have been met and the product owner did not include it in the product increment. That means that the team only got four story points their velocity is four in this sprint. Now note that defects and SPIKEs do not have story points, they don't add to the functionality of the product increment. And so they do not have any user story points. And even though the team has to work these they are not counted to the team's velocity.   
*The presenter opens the Track tabbed page and selects Sprint 1 from the Iteration Task Status drop-down list.  
  
The presenter opens the Track tabbed page and selects Sprint 3 from the Iteration Task Status drop-down list. The Iteration Tracking Board includes the columns: Defined, In-Progress, Completed, and Accepted.  
  
No user stories are listed in the Defined and In-Progress columns. One unfinished user story with three points is listed in the Completed column and one story worth four points is listed in the Accepted column. The Accepted column also contains two defects.*   
  
So let's see what happened in **Sprint 2**. In Sprint 2 they took on eight story points and it looks like they completed in both, they have a nice burn down chart. So at the end of Sprint 2 they had two sprints. The first one they got four story points, the second eight, so their velocity would be six; the average of those two. So let's go to **Sprint 3**, that's the last Sprint that we completed. And in this sprint they took on seven story points, they completed five of them, they complete the user story with five story points. But this user story was blocked and so they don't get those two and so they have a velocity of five for this sprint. Now overtime you can track the velocity of the team and it should stabilize overtime. In fact it should not only stabilize but it should go up as the team learns how to be self-organizing and gains efficiency. So here from our Velocity Chart we see our last three sprints, we took on eight the first two – eight story points – and we took on seven in the last sprint. We completed five, completed eight, completed five; so our velocity the trend of our velocity is right under seven story points.   
*The presenter navigates to the Sprint 2 Iteration Tracking Board. No user stories are listed in the Defined, In-Progress, and Completed columns. Two stories are listed in the Accepted column – one story is worth three points and the other is worth five points.  
  
The Iteration Burndown displays the graphical representation of the five day planning showing all tasks completed and accepted in the time given.  
  
The presenter navigates to the Sprint 3 Iteration Tracking Board. No user stories are listed in the Defined and Completed columns. One story worth two points is listed in the In-Progress column; and one story worth five points and a defect is listed in the Accepted column.   
  
The presenter navigates to the Administration page and focuses on the Velocity Chart for Sprint 1, Sprint 2, and Sprint 3. The chart shows that for Sprint 1, eight points were taken on but only five were completed and accepted. For Sprint 2, eight points were taken, completed, and accepted. And in Sprint 3, seven point were taken on but only five were completed and accepted.The velocity average shows just below seven points.*   
  
Now I should point out that even though velocity is a metric of performance, the team and the product owner should not be trying to maximize velocity. They should be trying to discover the team's optimal velocity over time. This is because if the team is pushed to maximize velocity, they could avoid best practices like unit testing. They could skip bug fixes or they could work long hours that leads to sloppy work. All of these tend to lead to technical debt and an unstable product increment. So you really want an optimal velocity, something the team can sustain over long periods of time. And when the team's velocity stabilizes it can be used for both short term and long term estimates. With a stable team velocity, the team in the Scrum master can ensure the cumulative number of story points accepted into any sprint backlog is consistent with the velocity of the team or the amount of work that they have historically completed in the sprint. This keeps them from being over committed. Also a stable team velocity is valuable to the product owner. The team's velocity gives the product owner a metric estimate of how much work the team can complete before the next release. They just have to multiply the team's velocity by the number of sprints before the next release, and then they can go to their product backlog and look at the highest priority PBIs and their estimates to get an estimate about which PBI features could make it into the next release.

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a02_it_enus/output/html/course_transcript.html#top)

Managing the Sprint

Learning Objective

*After completing this topic, you should be able to*

* *describe how to manage the Sprint*

**1. Managing the sprint**

The Scrum master meets daily with the team, in the Scrum meeting. This is where the Scrum master takes the pulse of the team, learning what each team member worked on yesterday, and what they will work on today. This daily standup provides the Scrum master with an opportunity to encourage optimal team behavior, backing up suggestions with the Agile principles and Scrum values. So what should be encouraged? This is the tracking Board for our current sprint. And we see right away that all of our sprint PBIs are In-Progress. With so much work In-Progress the team is not focused on the same thing, which leads to context switches as they try to work on tasks from one user story while still helping their team-mates complete tasks in another user story. Context switches cause a loss of productivity and should be minimized as much as possible. The way to do that is to minimize the work In-Progress at any one time, which will maximize the number of user stories completed in the sprint.   
*An Iteration tracking board is displayed. No user stories are listed in the Defined, Completed, and Accepted columns. Two user stories and a SPIKE are listed in the In-Progress column.  
  
The two user stories are US41: Display Plant Information on web site, with a completion status of 23%, 43 hours of work left on it, and a point value of three; and US32: Administer can Edit plant information in the database, with a completion status of 8%, 93 hours of work left on it, and a point value of four.*   
  
Teams should be encouraged to swarm on the task of the highest priority PBI, completing it before moving to the next PBI. Limiting work in progress helps the team stay focused and the user stories are completed one after the other during the sprint. This is much more preferable than having all the PBIs being in progress and not completing until right near the end of the sprint, which makes it extremely hard to track the team's daily progress or get a good reading on how the sprint is progressing. To ensure the highest priority PBIs are completed, as well as the maximum number of PBIs, the Scrum master should encourage each team member to work on the highest priority task they have the skill set to accomplish, favoring tasks in the higher priority stories over the lower priority ones while also favoring user stories with few tasks left to complete as this will shorten the cycle time on the user story, maximizing the number of user stories completed.   
  
Now **US41** here is at the top of our list, is the highest priority user story and we can see it only has 43 hours left. Where there's 93 hours left on this one. So this is the highest priority user story and it has the fewest number of tasks. So it's closer to being done. So if we look at it, these are the tasks for the user story. The ones that are not completed yet are the ones that the team members should choose to focus on and work on next. So this user story will be completed before they move on to the next user story. Also part of managing the sprint is protecting it from outside distractions. As the team learns to work together, depending on each other, losing a team member for even a short period of time can play havoc with the work flow. These situations show up as directives to the team members to drop everything and work on a hot fix. Or as soon as you finish that, this client requires a high priority thing be worked next. These directives lead to context switches, which destroys the team's focus. People are more productive if they can focus on one problem at a time.   
*The presenter navigates to the Tasks page for US41 where the tasks have different estimated hours and states – Defined, In-Progress, and Completed.*   
  
Also according to the Scrum rules, the team has the authority to organize how they complete their work and has planned together how they will manage the work of the sprint by breaking their PBIs into tasks, independent tasks that allow them to work in parallel. Stakeholders outside the team are not privy to the team's shared vision of the work to be done. Their directives to yank a team member out of the work flow can easily disrupt the work of the rest of the team putting the entire sprint at risk. You can see here if a GUI developer was taken out, then the "Detailed plant information" page would not be completed. There's only four hours left on that and without it being able to complete it, our web developer cannot send the results to the website. That's only one hour left there. And our QA is almost done with their test plan, where they are going to want to move on to testing. But they can't ensure the "Detailed plant" displays correct information if it is not done yet. So taking our GUI developer out for even a day is going to put a block on our development and our QA. Protecting the sprint means minimizing these types of outside distractions so that the team can work together, focusing on the work in the sprint.   
*The presenter points out various tasks that are dependent on other tasks and teams being present to complete those tasks.*

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a02_it_enus/output/html/course_transcript.html#top)

Limiting Work in Progress

Learning Objective

*After completing this topic, you should be able to*

* *describe how to limit the work in progress*

**1. Limiting work in progress**

During the sprint, the team should strive to limit the number of PBIs that are in progress at the same time. If the work in progress is not limited, it leads to team members context switching. Teams can very easily lose focus and become inefficient through working on too many things at the same time. Your sprint tracking board will show you which PBIs are Defined, which are In-Progress, which means some of the tasks are being worked on, which are Completed, and then which are Accepted, and Done. If you notice many PBIs in progress at the same time, it could mean that the team is focusing on their individual tasks and not on completing user stories. This can lead to having user stories almost done at the end of the sprint, where the team has done most of the work but since the user stories are not done, the team does not get the story's velocity points and they do not get to demonstrate the new functionality in the sprint review.   
*An example of an Iteration Tracking Board for Sprint 5 is given.There are no user stories in the Completed and Accepted columns, one SPIKE – US40 – in the Defined column, and two user stories – US41 and US32 – in the In-Progress column.*   
  
The Scrum is all about focusing on the top priority PBIs, as demonstrated in the product backlog prioritization. This priority focus should be carried into the sprint. The team should swarm on the highest priority PBI and complete it before moving to the next PBI to work. This has the effect of shortening the user story cycle time. The story is moved to done much faster and it encourages the team to work together, ensuring the highest priority PBIs are completed in the sprint and actually maximizes the velocity of the team as well. We see that team has two PBIs in progress here. Let's take a closer look at the task to see why. Limiting work in progress relates to the both top level PBIs and the tasks started by each team member.   
  
We see here this is our top-level user story and these are the tasks that are defined, the ones in progress, and the one's completed. Our GUI developer has completed the wire frame, our architect has completed the design, and the documentation of the interface between the presentation and the server components. And our developer, web developer, has completed the SQL to retrieve the detailed plant information. Here In-Progress, we see that our QA is just about done writing the unit test plan, our GUI developer is creating the actual page to display the plant information and also working on the zoom, and then our web developer wants to send the database results back to the web page and our documentation person is working on documentation. Well you can see the GUI person has two different tasks they are working on and both are just about halfway done. And then if you look at the next user story down here, well they've already started working on the wireframe and implementation on this one, which is editing the plant information. They have too many tasks in progress at the same time. They are not focusing on the top-level user story since they started the other one. And they are not really focusing on their top priority task.   
*The Defined column contains the tasks: GUI: Implement selection of thumbnails as main image, DOC: capture screen shots of detailed plants web page, QA: Ensure detailed plant displays correct information, QA: Ensure image zooms correctly, and QA: test multiple image selection and zoom of new image.  
  
The Completed column contains the tasks: GUI: Wire frame detailed plant page, DEV: Design and doc plant display I/F between presentation and server, and DEV: Create SQL to retrieve detailed plan information.  
  
Tasks in the In-Progress column for US32 are: QA: Write Unit Test Plan, GUI: create detailed plant information page, GUI: Implement zoomed image, DEV: send database results to web page, DOC: document detailed plant information page use.  
  
The presenter scrolls to the user story US32: Administer can Edit plant information in the database.   
  
The Defined column contains the tasks: QA: Create test plan for edit plant information, DEV: create SQL from web input to update plant record in database, GUI: implement uploading new image, and DEV: implement accepting new image and SQL to update image in database.  
  
The In-Progress column contains the tasks: GUI: wire frame edit plant information page and GUI: implement edit plant information page.*   
  
Until they actually create the page, the developer can't finish, you see web developer has only one hour left. They are probably waiting on this page to be done so they can send the results back to it. Also our QA is just about done with the test plan. They are going to want to move over to testing the actual page, but until the page is done, they can't do that, so they could be blocked very quickly. Each team member limiting their work in progress to a minimum number of tasks helps them focus, completing tasks that could block other team members, if not completed in a timely manner. Also to limit work in progress, each team member needs to focus on the user story more than their own individual tasks. We see our architect here has finished designing the interface between the components; the web server and the presentation. But then they came down and started working on the next user story, their next task, which is designing the interface here. However, the user story isn't complete yet, it is still in progress. To minimize the cycle time it takes to finish this user story, all the team members should be working on it at the same time. Our architect here instead of going to the next user story could have helped the GUI implement the zoomed image, so our GUI developer could concentrate on finishing the detailed plant information which would help the overall work flow.   
  
Now if one team member has finished their assigned tasks and their user story – like our architect – but the story is not complete, before starting tasks on a new story they should first see if there's any way they can help their team members complete their in-progress tasks. This enhances the team's commitment and team work, while moving the entire user story to Done, which benefits the entire team. Now there are always times when multiple user stories will be in progress at the same time. If there is an impediment...the Scrum master's working...the team should do all they can to help clear the blockage. But if there's nothing they can do, they should not sit idly by waiting for the blockage to be removed, instead they could move on to the next highest priority task they have their skills set to accomplish, even though it would cause a context switch. This could be a task in the next user story. Also a team member may not have the skill set needed to tackle any task in the current user story or to help any other team member, then they should move on to the next task in their next user story. However, the team members should move to the same next user story, the highest priority one left to complete. Limiting work in progress provides focus as the team is collectively focusing on completing a single user story, which increases the team's productivity and their velocity.

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a02_it_enus/output/html/course_transcript.html#top)

© 2016 Skillsoft Ireland Limited