[Print](javascript:window.print())

**Course Transcript**

Software Practices (SCRUM): SCRUM Roles

**The Agile Manifesto**

[1. Course Introduction](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a01_it_enus/output/html/course_transcript.html#t2)

[2. The Agile Manifesto](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a01_it_enus/output/html/course_transcript.html#t6)

[3. Principles Behind the Agile Manifesto](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a01_it_enus/output/html/course_transcript.html#t10)

**Scrum Overview**

[1. What is Scrum?](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a01_it_enus/output/html/course_transcript.html#t15)

[2. Scrum Values](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a01_it_enus/output/html/course_transcript.html#t19)

[3. The Scrum Empirical Process](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a01_it_enus/output/html/course_transcript.html#t23)

[4. The Scrum Sprint](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a01_it_enus/output/html/course_transcript.html#t27)

[5. Protecting the Sprint](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a01_it_enus/output/html/course_transcript.html#t31)

[6. Ending a Scrum Project](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a01_it_enus/output/html/course_transcript.html#t35)

**Scrum Roles**

[1. The Three Scrum Roles](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a01_it_enus/output/html/course_transcript.html#t40)

[2. Committed and Involved Roles](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a01_it_enus/output/html/course_transcript.html#t44)

[3. The Product Owner Role](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a01_it_enus/output/html/course_transcript.html#t48)

[4. The Scrum Master Role](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a01_it_enus/output/html/course_transcript.html#t52)

[5. The Scrum Team Role](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a01_it_enus/output/html/course_transcript.html#t56)

[6. The Management Role](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a01_it_enus/output/html/course_transcript.html#t60)

[7. Scrum and Culture Change](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a01_it_enus/output/html/course_transcript.html#t64)

**Scrum Artifacts**

[1. The Four Scrum Artifacts](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a01_it_enus/output/html/course_transcript.html#t69)

[2. The Product Backlog](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a01_it_enus/output/html/course_transcript.html#t73)

[3. The Spike Product Backlog Item](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a01_it_enus/output/html/course_transcript.html#t77)

[4. Product Backlog Items (PBI)](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a01_it_enus/output/html/course_transcript.html#t81)

[5. Fat Product Backlog Items (PBI)](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a01_it_enus/output/html/course_transcript.html#t85)

[6. The Sprint Backlog](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a01_it_enus/output/html/course_transcript.html#t89)

[7. The Burn Down Chart](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a01_it_enus/output/html/course_transcript.html#t93)

[8. The Product Increment](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a01_it_enus/output/html/course_transcript.html#t97)

**The INVEST Guidelines**

[1. The INVEST Independent Criteria](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a01_it_enus/output/html/course_transcript.html#t102)

[2. The INVEST Negotiable Criteria](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a01_it_enus/output/html/course_transcript.html#t106)

[3. The INVEST Valuable Criteria](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a01_it_enus/output/html/course_transcript.html#t110)

[4. The INVEST Estimable Criteria](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a01_it_enus/output/html/course_transcript.html#t114)

[5. The INVEST Small Criteria](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a01_it_enus/output/html/course_transcript.html#t118)

[6. The INVEST Testable Criteria](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a01_it_enus/output/html/course_transcript.html#t122)

Course Introduction

Learning Objective

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

* *start the course*

**1. Introduction to the course**

SCRUM is an Agile development methodology and so follows the Agile manifesto. In this course, Brigitte Birze will explore the guiding principles of SCRUM – an overview of the process, the makeup of a SCRUM team, SCRUM artifacts, and the INVEST guidelines. I hope you enjoy the course.

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a01_it_enus/output/html/course_transcript.html#top)

The Agile Manifesto

Learning Objective

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

* *describe the Agile Manifesto*

**1. Describing the Agile manifesto**

If you're using an Agile methodology it's following the Agile manifesto, which is the umbrella over all the Agile software frameworks. The Agile manifesto declares the core values, which are the heartbeat driving all Agile software methodologies. But why and how was it created, and what are these very important values to finding Agile? Several software methodologies evolved in the mid 1990s as alternatives to the heavyweight waterfall-oriented processes. These new software methodologies advocated processes that were much lighter weight, resulting in a more nimble team which could respond quickly to changing requirements in dynamic development environments. The term *Agile* was used to describe how this class of software processes worked. The Agile movement was not anti-methodology, instead it strove to restore balance between required process and the need to get the work done as efficiently as possible.   
*Heading: The Agile Alliance.  
  
The Agile alliance is formed from Crystal, Pragmatic Programming, Adaptive Software Development, Extreme Programming, DSDM, Feature Driven Development, and SCRUM.  
  
The Manifesto for Agile Software Development can be found at http://agilemanifesto.org.*   
  
In February of 2001, leaders of these Agile software methodologies met a ski resort in Utah to find common ground. The Agile methods represented included Extreme Programming, SCRUM, DSDM, Adaptive Software Development, Crystal, Feature Driven Development, Pragmatic Programming, and more. This group named themselves the "Agile Alliance" and agreed on a set of common values, ideas, and themes that define the Agile software development approach. This is summed up in the simple and elegant manifesto for Agile software development. The Agile manifesto in its entirety reads, "We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value individuals and interaction over processes and tools, working software over comprehensive documentation, customer collaboration over contract negotiation, responding to change over following a plan. That's it, while there is value in the items on the right, we value the items on the left more." This is documented on the Agile manifesto website at AgileManifesto.org.   
*Heading: The Manifesto for Agile Software Development.  
  
On the left, the higher value options are individuals and interactions, working software, customer collaboration, and responding to change.  
  
On the right, the lower value options are process and tools, comprehensive documentation, contract negotiation and following a plan.*   
  
Agile development is not about eliminating steps such as modeling or documentation, it is about doing the correct amount of each step and relying on communication and collaboration between cross-functional teams and customers, to allow rapid and flexible responses to inevitable change. This is reflected in the values, where motivated individuals are key to the self organizing teams, and interaction is built-into the framework through collocation and interactions like product increment reviews and retrospective meetings. Working software is more valuable to the customer than documents and presentations. Even software with limited functionality allows the customer to give feedback early in the project where it's easy to incorporate new requirements or make changes to features. Continuous customer involvement and collaboration is key to uncovering missing requirements early, reducing the need to negotiate new contracts late in the project. An Agile development understands the entire scope cannot be captured at the beginning of the project, when the least is known about what is required. Since change will happen, responding to change is a key value built into the framework. Agile development is focused on quick response to change using continuous iterative development.

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a01_it_enus/output/html/course_transcript.html#top)

Principles Behind the Agile Manifesto

Learning Objective

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

* *list the 12 principles behind the Agile Manifesto*

**1. The Agile manifesto principles**

The first six Agile principles documented on the Agile Manifesto web site address communication and interaction. Specifically, how to manage the relationship between the customer and the team, and within the team itself. The customer focused principles lay the groundwork for managing changing requirements and customer collaboration. Since all the requirements are not known at the beginning of the project, continuous customer involvement is crucial to delivering a final project that satisfies and delights the customer. The first principle states, "Our highest priority is to satisfy the customer through early and continuous delivery of valuable software." This allows the customer to give feedback regularly, as the product evolves. The teams response to this feedback is encompassed in the second principle, "Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage." The third principle sums up how this is managed, "Deliver working software frequently, from a couple of weeks to a couple of months, with a preference for the shorter time scale."   
*Heading: Agile 12 Guiding Principles: Customer and Team.  
  
The three customer focused principles are early and continuous delivery of software, welcome changing requirements, and deliver working software frequently.*   
  
The next three Agile principles deal with the organization of the team and how it functions. First, "Business people and developers must work together daily throughout the project." This ensures the vision held by the business is understood by the developers creating the project. The next principle touches on the self-organizing and directed team. It states, "Build projects around motivated individuals, give them the environment and support they need, and push them to get the job done." And the sixth principle puts the emphasis on collocating teams to ensure easy communication. It states, "The most efficient and effective method of conveying information, to and within a development team, is face-to-face conversation." The next three Agile principles define the development environment. The seventh principle zeros in on why the project exists, "Working software is the primary measure of progress." Progress in Agile is not gauged by a Gantt chart or schedule on some managers office, but by working software, and the features and functions it performs.   
*The three team focused principles are business and developers work together daily, build projects around motivated individuals, and emphasis is on face-to-face conversation.  
  
Heading: Agile 12 Guiding Principles: Development and Self Improvement.  
  
The three development focused principles are working software shows progress, a sustainable development pace, and require technical excellence and good design.*   
  
Next, "Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely." This means the team should be able to complete their work in a 40 hour week with nights and weekends off. And this leads to alert workers that can adhere to the ninth principle, "Continuous attention to technical excellence and good design enhances agility. Best practices like unit test, quality assurance, and completed customer documents ensures that working software is free of bugs and ready for delivery, greatly reducing technical debt."   
  
The last three Agile principles round out what it means to be Agile. First, "Simplicity, the art of maximizing the amount of work not done, is essential. The working software in each iteration should contain the highest priority features, the ones which will be used the most, uncluttered by the often unused bells and whistles." This leads to an elegant, simple software design which is easy to maintain and enhanced. Principle 11 states, "The best architectures, requirements, and designs emerge from self-organizing teams. These teams are cross-functional and self-directed and depend on each other to accomplish the work the team has accepted." And lastly, "At regular intervals the team reflects on how to become more effective, then tunes and adjusts his behavior accordingly. This allows for continuous self improvement in the teams performance."   
*The three remaining principles are simplicity, self-organizing teams, and regular team retrospective and adaptation.  
  
The Agile team improvement is based on a simple lifecycle model that has three phases: retrospective process improvement, a simple iteration, and a product increment.*

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a01_it_enus/output/html/course_transcript.html#top)

What is Scrum?

Learning Objective

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

* *describe the Scrum framework*

**1. Describing the Scrum framework**

In rugby, a scrum is a way of restarting the game after a minor infraction. In the software project environment, *Scrum* is an incremental, iterative, Agile framework for software project development and delivery. But Scrum is not just for software, it can be used for any complex type of project. The software development lifecycle includes analysis and planning, design, development, test, and deployment. In the waterfall process, each step is done to completion before the next step is started. However, this means all the requirements must be determined at the beginning of the project when the least is known and the customer does not see the product until the end of the project, running the risk that what was built was not what the customer wanted. Scrum is based on an iterative approach to the software development lifecycle. In Scrum, the project is broken up into development units called sprints. The sprint is a time-boxed iteration of the complete software development lifecycle. The sprint includes planning and analysis, design, development, test, and delivery of the product increment to the customer. The product increment is potentially shippable software that contains the highest priority functionality of all the requirements in the project.   
*Heading: Scrum and the Software Development Lifecycle.  
  
The classic waterfall development model has five distinct stages: Requirements analysis, Design, Coding, Testing, and Deployment. Each stage leads into the next stage.  
  
The Scrum Development model is a series of sprints that are all on the same hierarchical level, with a product increment happening at each sprint.*   
  
As each sprint builds on the current product increment, the product increment grows in functionality with each completed sprint. At the end of each sprint, the customer sees a demonstration of the current product increment and gives feedback, allowing the team to quickly respond to changing requirements and ensuring the final product matches the customer's expectations. Allowing the team to quickly respond to changing requirements and ensuring the final product matches the customer's expectations. At the end of each sprint, a team retrospective is held on what went right and wrong with the last sprint, to ensure continual process improvement. Scrum contains three core roles: the product owner, the team, and the Scrum master. The product owner owns the vision of the product and interacts with the stakeholders to create the requirements held in the product backlog.   
*Heading: Scrum Roles and Artifacts.  
  
The artifacts are the Product backlog, sprint backlog, burn down chart, and product increment.  
  
The roles are product owner, Scrum master, and team.*   
  
The product backlog holds the features, bug fixes, and non-functional requirements needed to successfully deliver the completed product. Items in the product backlog are ordered by priority with the highest priority items at the top. The team is a cross-functional, self-organizing group that is responsible for delivering potentially shippable product increments at the end of each sprint. The team helps break down the items in the product backlog into manageable user stories that can be accomplished in a single sprint. During planning, the highest priority items are moved from the product backlog into the sprint backlog to indicate what will be delivered in the next product increment. The team then commits to the items in the sprint backlog and begins development. Development during a sprint is tracked with a burn down chart. The burn down chart is updated daily to show the remaining work in the sprint backlog, giving a quick visual of the team's progress. The Scrum master is responsible for ensuring the Scrum rules are followed. The Scrum master facilitates the key Scrum meetings and also protects the team in the sprint, removing impediments and ensuring the team can focus on the work in the sprint backlog.

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a01_it_enus/output/html/course_transcript.html#top)

Scrum Values

Learning Objective

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

* *use the five Scrum values to build a foundation for teamwork and continuous improvement for a team*

**1. Using Scrum values**

Each of the five Scrum values are a single word that can mean different things to different people. So it's a good idea to get the team together for an informal discussion early in the project, to determine what they mean to the team. A broad definition of a common set of values is given in the history of the Agile manifesto, on the Agile manifesto's web site. It states, "A set of values based on trust and respect for each other, and promoting organizational models based on people, collaboration and building the types of organizational communities in which we would want to work." So how does this relate to our five one-word Scrum values? The first value is Commitment; the team determines the amount of work it can accomplish in the next sprint, in the sprint planning meeting and commits to getting it done. This commitment is done as a team, as the entire team succeeds or fails in meeting its commitments. So there is also commitment for each team member to the team itself. Each team member will do what has to be done to help each other succeed, so the team will succeed.   
*Heading: Scrum Values.  
  
The first Scrum value is commitment.*   
  
Commitment means the team must be focused on the work of the sprint. They cannot be pulled for other projects. The team focuses on a small list of items, the task accepted into the sprint backlog. Team focus on the same well-defined goals leads to high-quality work that delivers value quickly. Courage. As the team gains trust in their ability to work together and accomplish goals, they have courage to stretch and take on more work and greater challenges. Everyone makes mistakes. Courage is needed to point out the issues as they arise, allowing the team to fail early. Failing early may seem like a strange concept, but its highly desirable. Knowing of a failure early in the project allows quick adjustments and recovery, resulting in a successful project overall. While in traditional methodologies, the team does not realize they have failed till the end of the project. Courage means the team speaks up when things are not going well. And is also comfortable telling the product owner "no," when they feel they cannot accept more work into the sprint without going over capacity, which relates to the next value, Openness.   
*The next three Scrum values are focus, courage, and openness.*   
  
Openness relates to visibility, to be open about where the team is and how they are progressing. The team is open about issues, concerns, and blockages as they come up, so they can be addressed and removed. Openness is built-into the daily Scrum stand-up meetings, where each team member relates what they did the past day, what they will do today, and the blockages they are facing. It is also built into the sprint retrospective meeting, where the team discusses the things that went right and wrong in the last sprint. Openness also relates to the transparency inherent in Scrum, shown in the burn down chart that is publicly posted, and shows how much work has been completed and is left to do at anytime during the sprint.   
  
All of the other values come together in Respect. In Scrum we respect the team's ability to self-organize, and take on what they feel they can accomplish in the sprint timeframe. We do not try to bully the team into taking on more than they can perform at a sustainable pace, as sustainable development is the eighth Agile principle. We respect the team's need to focus on the goals of the sprint, and do not try to distract them with outside work. We respect the opinions of others, which gives them the courage to be open. We also respect the effort and commitment of others, even when they fail, as failing early allows the entire project to succeed in the end. And since teams commit to the work they will accomplish, they respect their commitments to accomplishing the goals and to their team. Operating on the five Scrum values is key to creating the culture that supports the Scrum process.   
*The fifth Scrum value is respect.*

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a01_it_enus/output/html/course_transcript.html#top)

The Scrum Empirical Process

Learning Objective

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

* *describe how inspection, adaptation, and transparency can improve how people work together toward common goals*

**1. Using the empirical process**

The three pillars of the Scrum empirical process are Transparency, Inspection, and Adaptation. Scrum provides transparency of the entire process to everyone associated with the project, including the committed core team roles and the involved stakeholders and managers. Everything associated with the project is openly observable as it is happening. The transparency is built into the Scrum artifacts, meetings, and processes. Artifacts produced within the sprint include the sprint backlog, which clearly shows what is going to be delivered in the sprint. The publicly posted burn down chart shows the team's progress at any point in the sprint. And the product increment delivered at the end of the sprint, is working software that can be exercised to demonstrate the completed features. The product backlog is ordered by priority, making it easy to see what features are considered more important and will be delivered first. Transparency is inherent in the Scrum meetings. In the daily stand-up Scrum meeting, each team member relates what they did yesterday, what they will do today, and what is blocking them. And the sprint retrospective meeting is all about the team coming together, to openly discuss what went wrong and right in the last sprint.   
*Heading: Scrum Empirical Process.  
  
The first pillar is Transparency.*   
  
In the sprint review, the customer receives a demonstration of the product increment, the working software in a deliverable form. The visibility afforded in these reviews, allows the customer to give feedback as the product is being developed, allowing adjustments to ensure the final product satisfies the customer. Inspection means taking advantage of the transparency, to observe each step of the process and critically evaluate how the project is progressing. This inspection is facilitated by the Scrum artifacts and meetings. The artifacts produced are used and inspected by the team and stakeholders daily, to determine how the project is progressing. These same artifacts are also used in the sprint retrospective meeting. The sprint retrospective is a key meeting, in which the entire Scrum team reviews the performance in the past sprint, discussing issues, their causes and how processes can be adjusted to alleviate the issues in the future.   
*Heading: Scrum Empirical Process.  
  
  
The second pillar is Inspection.*   
  
It's not enough just to make observations, in Scrum the observations must lead to adaptations. The insights gained from the sprint retrospective prompts the team to change how they do work, to foster continuous process improvement. Change is incremental, a few changes are chosen to implement in each next sprint. This ability to continually adjust and tweak the process, allows the team to become more and more efficient and effective over time. Use of the Scrum empirical process allows people from cross-functional groups, to communicate and collaborate, not only on the work at hand but how they are doing the work. This cycle of transparency, inspection, and adaptation over time, leads to strong cross-functional teams, that are continually improving how they work together toward common goals.   
*Heading: Scrum Empirical Process.  
  
The third pillar is Adaptation.*

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a01_it_enus/output/html/course_transcript.html#top)

The Scrum Sprint

Learning Objective

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

* *describe how the Product Increment is developed throughout the lifecycle of your project*

**1. Working with sprints**

Each sprint is the same length from 1 to 4 weeks long. The team determines how long a sprint will be, and the duration is consistent across all the sprints in the project. The work completed during the sprint results in potentially shippable software, called the product increment. In the beginning of the project, the product increment will be very simple with very basic functionality, but each sprint iteration, incrementally builds on the functionality already completed in the last product increment, which is why Scrum is described as iterative and incremental. And since each sprint includes the highest priority items in the product backlog, the resulting product increment will always include the features, which the customer considers the highest value. Each sprint begins with a sprint planning meeting, where items are moved from the product backlog to the sprint backlog, signaling they will be completed in the upcoming sprint. The product owner maintains the product backlog with the highest priority items at the top. So the product owner has authority to say what is in the sprint. The Scrum team says how the work will be completed, breaking each item into small tasks with a time estimate.   
*Heading: The Sprint.  
  
The Scrum development process is a series of sprints that are linked to each other, linearly. At the end of each sprint the product increment grows larger.  
  
Heading: Sprint Meetings.  
  
The Product Backlog is linked to the Sprint Backlog.  
  
Each sprint has a daily Scrum as one of its processes. The complete sprint stages are Sprint Planning, Sprint Review, and Sprint Retrospective.*   
  
The time estimates are used to determine when the work added to the sprint backlog meets the capacity of the team during the next sprint. At this point, the team commits to completing the work in the sprint backlog, which is frozen; no more items can be added to the sprint. During the sprint, the team owns the work, and retains absolute responsibility for completing it in the manner that best suits the team. Daily Scrum meetings are held with the team and the Scrum master, to check in and communicate where each team member is on the task they have accepted. During the Scrum meeting, each team member reports what they did yesterday, what they will do today and any impediments blocking them. The Scrum master is responsible for removing impediments so the team can continue to make progress. When the duration determined for the sprint is over, the sprint concludes with the sprint review meeting. In this meeting, the team demonstrates the functionality added to the product increment, to the product owner.   
  
Customers and stakeholders often attend the sprint reviews, to monitor the product increment as it grows and develops over multiple sprints. The sprint review also provides the opportunity for the stakeholders and customers, to give feedback to the product owner and team after they see the product increment demonstration. The performance of the team during the sprint will vary, especially at the beginning of the project when the members are not yet use to working as a team. That is why the sprint retrospective meeting is held after every sprint, so the team can openly discuss the events of the last sprint, what worked, and what issues were encountered. The team can then focus on the root cause of the issues and put processes in place, to ensure the same issues are not encountered in the next sprint, leading to continual process improvement.

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a01_it_enus/output/html/course_transcript.html#top)

Protecting the Sprint

Learning Objective

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

* *use the Scrum values and rules to assist the Scrum Master, team members, and stakeholders to protect the Sprints*

**1. Protecting the sprints**

It is essential for the Scrum process that the Scrum team has authority over the work they take on, and is allowed to focus on how that work is accomplished during the sprint. This enables the team to become self-organizing with no set leader. Over time, the team will establish a consistent velocity for the amount of work they can complete in a sprint. This helps the team accurately estimate how much work to take on in each sprint planning meeting. As the team gets better at working together, its velocity will increase and more work will be accepted and completed in each sprint. Until this point the Scrum master must protect the sprint, they must ensure the team does not exceed its capacity, when taking on work in the sprint planning meeting. Exceeding its capacity means the work will not be completed in the sprint. The associated user story will not be accepted as done, and the feature will not be demonstrated in the product increment in this sprint review. The work not completed is technical debt, which will need to be worked in a future sprint, reducing the new functionality that can be accepted from the product backlog in that sprint. The Scrum master must also ensure the Scrum values are followed within the team, until the team develops camaraderie. Especially important are the Scrum values of staying focused on the work in the sprint, and respect for everyone involved in the project, especially other team members.   
  
The Scrum master must also protect the team from outside influences. External stakeholders are the biggest threat to a self-organizing Scrum team. This group includes executives, customers, managers, and others with a vested interest in the project. External stakeholders normally contribute resources to the project, or they are responsible for aspects of the projects such as budgets, schedules, or priorities. So they often feel they have a right to step in and micro-manage a sprint, or overturn a team's decision. While stakeholders have a legitimate stake in the success of the project, they are not committed to completing the work of the sprint like the Scrum team. Also external stakeholders are not involved in the sprint planning meeting, or the day-to-day development. So their knowledge is incomplete on many facets, such as the task breakdown, how the task are being worked, or why the team estimated a user story as a large or small relative level of effort.   
  
Sometimes external stakeholders are called chickens, while Scrum members are called pigs. This is because when considering breakfast, the chickens are involved like the stakeholders, but pigs are committed like the Scrum team. The Scrum master must protect the Sprint, by ensuring chickens do not overly participate in the day-to-day activities of the team, which has a detrimental effect on the team's focus and can derail the entire Scrum effort. The Scrum master can do this by enforcing Scrum rules. First there is a rule that while stakeholders can observe the daily Scrum stand-up meetings, chickens don't talk in the daily Scrum meetings. The Scrum is a time-boxed meeting, if a stakeholder is talking and asking questions, the team is not communicating their status and the goal of the Scrum meeting is not met. A second rule is once the team commits to the items in the sprint backlog, no other items may be added to the sprint.   
*Heading: Protected Sprint  
  
The Scrum master enforces the Scrum rules, which include, chickens don't talk in the Scrum, the sprint backlog is frozen, has team authority over how work is completed, and Scrum artifacts and meetings are sufficient.*   
  
If new requirements are added once the sprint starts, the team will most likely be over capacity and not be able to complete the work in the sprint. Also the team will have limited time to breakdown the item into tasks and plan for it, if it is not accepted in the sprint planning meeting. Next the team has complete authority over how the user stories in the sprint are to be completed. The Scrum master must ensure no outside influence has the authority to oversee, or change the technical decisions of the team. Lastly, the Scrum artifacts provide transparency to the team's progress, and the Scrum meetings are well defined, providing a means for all Scrum members and external stakeholders to influence the project at the appropriate time. The team should not be required to attend additional status meetings, or give additional status reports, as this indicates an outside authority is operating to influence the team, undermining its effort to be self-organizing and autonomous.

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a01_it_enus/output/html/course_transcript.html#top)

Ending a Scrum Project

Learning Objective

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

* *use the roadmap release schedule to determine the end of a Scrum project*

**1. Using roadmap release schedules**

Any project begins with stakeholders agreeing to apply resources, to achieve something that has shown to have a satisfactory return on investment. Stakeholders will not agree to support a project that is not well defined. So before any project starts, there will be a roadmap tied to a release schedule. The more complex the project, the more planning is required before it starts. Releases with set dates and minimum functionality required in each release, should be defined before the project begins. However, the iterative nature of Scrum allows for discovery and adaptation to changing requirements. So there should be room in the release roadmap to add or remove functionality as priorities change. In these cases, the Scrum project ends when the milestones are met, the roadmap release schedule finishes, the project runs out of money, or a specified deadline arrives, and the product increment has an acceptable defect level. However, the Scrum project could be terminated at any point by the stakeholders, or the product owner. The product owner owns the product backlog, and knows the value each item will add to the end product. Through prioritizing the product backlog, the product owner determines what features will be delivered in each sprint. Though customers can request a feature be added to the product backlog, it does not mean it has sufficient value or high enough priority to make it into a sprint.   
*Heading: When to End a Scrum Project.  
  
There can be a planned schedule for the end of a project.  
  
There can be an early project termination based on the priority of items in the Product Backlog.*   
  
The product owner should continually evaluate the return on investment of the items, remaining in the product backlog. Product owners, stakeholders, or customers can negotiate to end the Scrum p if they feel the return on investment for the remaining product backlog items, is too low to continue. At the end of any sprint, the product increment will contain the highest value features, and be potentially shippable software complete with documentation. Since each sprint always works on the highest priority task in the product backlog, if a decision to release is made at the end of the particular sprint, the stakeholders know the most important features are part of that product increment, which will be deployed as a product. Since the product increment is potentially shippable software, the release is less dramatic than in other methodologies, where the software is not fully integrated, tested, or documented till the end of the project. However, there are still items that must be worked, to get a product increment ready to be released as a working product. These are completed in a release iteration or release sprint, which is used to prepare the product increment for deployment.   
*Heading: The Release Iteration.  
  
The project kickoff leads into a series of development sprints. The output of the development sprints is a product increment. The decision to release follows on from the development sprints.  
  
After the decision to release is made, there is a release sprint stage. This leads to deployment and a released project.*   
  
Activities that occur in the release sprint include finalizing the documentation, that is in a state of constant flux during the sprints, such as the architecture and design documents, and user manual showing screenshots. If multiple Scrum teams are working on different aspects of the same project, system integration and testing must be completed. On large international projects the screen could be internationalize for different locales, and the user documentation translated into different languages. Final defects are resolved and a final acceptance review is held with the stakeholders. Physical shippable items like the installation media, manuals, and packaging need to be finalized and produced. If the release is an upgrade, the field may need to upgrade hardware or software applications, or migrate to a new database schema. And end users and support workers need to be trained on the products, new functionality.   
  
There may be several release iterations to accommodate different international markets, allow time to train a large number of users, or to accommodate the different schedules of different stakeholders. Once the final release is shipped and the Scrum project is over, it is often better to move new work to the established team, rather than moving team members to new work. This is because it takes time for the team to gel and learn to work together efficiently. Teams that are already self-organizing and have developed team unity, are one of the major productivity benefits of Scrum. These teams should be kept together to work on the next project whenever feasible.

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a01_it_enus/output/html/course_transcript.html#top)

The Three Scrum Roles

Learning Objective

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

* *list the three core roles defined in the Scrum framework*

**1. Defining the three Scrum roles**

The Scrum framework defines three core roles: the product owner, the team, and the Scrum master. Each role has a function and owns a different parts of the Scrum process and they must all work closely together with the stakeholders to successfully deliver the Scrum project. The product owner is the key stakeholder and holds the vision of the final product, making decisions to achieve a good return on investment on the project and the final product. The product owner interacts with the other stakeholders to create a list of features held in the product backlog. The product backlog is owned by the product owner who must assign a priority to each product backlog item. Then the product backlog is ordered with the highest priority items at the top and the lowest priority items at the bottom. The items at the top of the product backlog are the ones that'll be worked first. So by prioritizing the product backlog items, the product owner determines what features are included in each sprint. As the product owner represents the users and stakeholders they are required to attend the sprint planning, sprint review, and product backlog grooming meetings. They may also attend the daily Scrum meeting to answer questions, and are valuable in the sprint retrospective meeting as well.   
*Heading: Scrum Roles.  
  
The first role is the Product Owner, who has the product vision.*   
  
The Scrum team does the work necessary to deliver potentially shippable software at the end of each sprint. The team is typically small, between five and nine people, which work in a cross-functional self-organizing manner. While each member will have their own expertise say in architecture, QA, tech writing, or web design, there is no boundary lines between engineering roles. Each member of the team works on whatever task is needed to ensure the work the team committed to is completed within the sprint. The team is also self-directed and self-organizing with whole team accountability. The entire team succeeds or the entire team fails. There is no team leader; the team itself determines who will work on each task. The team works with the product owner in the backlog grooming meeting to break down the items in the product backlog into manageable user stories that can be accomplished in a single sprint. In this meeting, the team assigns a rough estimate of the level of effort it will take to complete the highest priority backlog items, and ensures their acceptance criteria is well understood. The team and the product owner also work together in the sprint planning meeting, moving the top priority items from the product backlog into the sprint backlog. The team breaks each item down into small tasks, putting estimates on each task.   
*The next Scrum role is the team, who are responsible for the product development.*   
  
When the estimated effort on the work in the sprint backlog reaches the capacity of the team in the next sprint, no more items can be added to the sprint. In this way the product owner determines what goes in each sprint while the team determines how the items will be accomplished and how long it will take. The product owner cannot overrule the team's estimates on how long it will take to complete tasks during a sprint. The team is an active participant in all the remaining Scrum meetings including the daily Scrum standup, sprint review, and sprint retrospective. Teamwork is vital and facilitated by the Scrum master who is responsible for making sure that the team is as productive as possible. The Scrum master does this by removing impediments blocking progress and by protecting the sprint. While the product owner owns the product vision and the team owns the product development, the Scrum master owns a Scrum process. As such the Scrum master is the enforcer of Scrum rules and is responsible for making sure a Scrum team lives by the values and practices of Scrum.   
*The third role is Scrum master, who monitors the Scrum process. Stakeholders are also a part of the Scrum process.*   
  
The Scrum master facilitates the key Scrum meetings including the daily Scrum, where the team relates impediments they are facing and how they are progressing on the tasks in the sprint backlog. The Scrum master assumes a servant leader role; they do not direct the team but instead facilitate teamwork, adherence to Scrum values and rules, and shields them from outside distracting influences so they can focus on the sprint tasks. Though not one of the official Scrum roles, the stakeholders are responsible for the projects existence. Stakeholders provide the funding and resources to produce the product desired. Stakeholders include managers, executive, customers, and others within an interest in the project. The stakeholders work with the product Owner to ensure their features and requirements are included in the product backlog. Anyone can add items to the product backlog, but only the product Owner has authority to prioritize the product backlog so the highest value features are delivered first.

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a01_it_enus/output/html/course_transcript.html#top)

Committed and Involved Roles

Learning Objective

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

* *define the two types of project members and the rules they must obey in Scrum development*

**1. Defining project member types**

In Scrum, the story of the chicken and the pig is often used to describe committed and involved roles. As the story goes, a chicken and pig were walking together when the chicken suggests they open a restaurant. The pig agrees and asks what they should call it, to which the chicken replies Ham & Eggs. After a moment of thought the pig says, "No thanks, I'd be committed but you'd only be involved." The three Scrum roles are committed, or the pigs. Pigs are directly accountable for completion of the task at hand. In other words, their bacon is on the line if the task isn't completed on time. They include the product owner, the Scrum master, and the team; the people who together get the work done. The Scrum values dictate that pigs be allowed to focus, so they can be completely committed to the project. As the pigs have accountability they are also empowered to provide estimates for the task and committing to the work to drive the project. The pigs must have the authority to do what is necessary to ensure success, as they are responsible for the projects return on investment.   
*Heading: Committed and Involved Roles.   
  
The Involved roles are regarded as chickens and the Committed roles are regarded as pigs. The committed roles include the Scrum master, the team, and the product owner.*   
  
The managers, customers, and others stakeholders are involved, or chickens. Chickens play a supplemental role, they have an interest in the project, and are consulted on the work to be done, and informed on the projects progress. However, they do not actually do any of the work and are not liable if the work is not done on schedule. And while interested in the projects return on investment, they are not accountable. Both chickens and pigs are necessary for a successful project, but for pigs and chickens to work together, there must be an awareness of their different roles and some rules must be followed. First stakeholders can meet with the product owner to ensure their features are added to the product backlog. But only the product owner can prioritize the items to ensure the highest value features are worked in the sprints. Stakeholders cannot change the priority of their desired features in the product backlog.   
*The involved roles consist of stakeholders.*   
  
Also the stakeholder should not be allowed to distract the team from their task at hand, so they can focus on the work being done. Team members cannot be moved to other teams in the middle of the sprint as this jeopardizes the entire sprint's success. And new requirements and tasks cannot be added to the sprint backlog once the team has committed to the work in the sprint planning meeting. The difference in involved and committed roles is also important in how the daily Scrum meeting is facilitated. As the Scrum meeting is time-boxed to 15 minutes, the discussions must be brief and relevant. What did each member do yesterday, what will they do today, and what impediments are blocking them? To help keep the Scrum meeting brief, it is often held with all participants standing. 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 make faces or create a distracting presence. This prevents the Scrums from going off topic with questions or long discussions. Though the product owner is a pig, they do not give status in the Scrum meetings and only talk if the team asks them a question to clarify a requirement. The Scrum master is responsible for enforcing the only pigs can talk rule. If too many chickens attend the daily Scrums, the Scrum master can limit attendance so the meetings remained focused.

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a01_it_enus/output/html/course_transcript.html#top)

The Product Owner Role

Learning Objective

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

* *describe the key Product Owner role in a Scrum software development team*

**1. Describing the product owner**

There can only be one product owner and they have the final say on what features go into the product. And the product owner gains this knowledge by interacting with the stakeholders to determine their interests and the features they want to see in the final product. The product owner is normally a key customer, or from product management or marketing, and is familiar with the competition, current trends, users, and popular product features. Through discussions and their own knowledge, the product owner forms a vision of what will be in the final product and conveys this vision to the Scrum team. This allows the team to focus on the work of each sprint without being distracted by requests from multiple stakeholders with different interests and priorities. The artifact used to hold this final product vision is the product backlog, which is owned by the product owner. The product owner adds the stakeholder's features and requirements to the product backlog as user stories. Each user story is written from a user-centric viewpoint. The only way a feature can get into the final product is through the product backlog. So the product owner has complete authority on what features are in the final product and thus has responsibility for the products success.   
*Heading: The Product Owner.  
  
The product owner controls the product backlog, which has user stories arranged in order from higher priority to lower priority.*   
  
The product owner is also responsible for prioritizing the product backlog, with the highest value features at the top and the lowest value features at the bottom. The Scrum team will always choose the features from the top of the product backlog to implement in the next sprint. So the prioritization chosen by the product owner will determine when a specific feature will appear in the product. And the product owner also decides when the product increment should be released, and they can accept or reject the work created by the team. As the product owner represents the users and stakeholders, they are required to attend the sprint planning meeting, the sprint review meeting, and the product backlog grooming meeting. They may also attend the daily Scrum meeting to answer questions, and can also provide valuable insights in the sprint retrospective meeting. The product owner must have in-depth knowledge of the final product, from the perspective of both the user's needs and the business needs. And they must be a good collaborator, as they will collaborate with both the team and the stakeholders.   
  
The product owner must help the team understand requirements while trying to understand the technical process and the difficulties of the task the team works. The product owner determines what goes into the product, but the team decides how the work is done and the level of effort it will entail. The product owner cannot overrule the team's estimates for the tasks accepted in the sprint. Additionally, the product owner should not push the team to take on more work than it believes it can accomplish in the sprint, and must not add additional work to the sprint once it is in progress. The product owner gathers new requirements and features by collaborating with the stakeholders, those with a vested interest in the project. New features and requirements can be requested at any time, which the product owner must add to the product backlog and prioritize. So keeping the product backlog up-to-date and prioritized is an ongoing task.   
  
Lastly the product owner must make the hard choices on the project, constantly balancing competing interests. They must balance deadlines against the readiness of the product for release. Balance the interest of competing stakeholders when determining the priority ranking of each product backlog item. They must balance the need to demonstrate functionality in the product increment with the need to complete all the tasks in the user story before it can be declared done to stem off technical debt. And they are the ones the Scrum team looks to as the final authority on what the user story is really asking for, and what is the correct acceptance criteria.

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a01_it_enus/output/html/course_transcript.html#top)

The Scrum Master Role

Learning Objective

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

* *describe the Scrum Master role in a Scrum software development team*

**1. Describing the Scrum master**

The Scrum master is a combination of coach, referee, fixer, mediator, facilitator, and score keeper. They are responsible for the team's productivity which is enhanced by the Scrum values, principles, and empirical process. Therefore, the Scrum master relies heavily on these pillars of the Scrum framework, while carrying out their responsibilities to ensure the team's success. And the Scrum master has many responsibilities. The Scrum master's most important task is removing impediments which are blocking the team's progress. They also help the team stay focused on the tasks in the sprint. They ensure quality is maintained by ensuring best practices like unit tests are included in the user story Definition of Done tasks and completed before each user story is accepted as done.   
*Heading: The Scrum Master  
  
The Scrum master controls the Scrum process, which is based on the pillars – Values, Principles, and Empirical Process.*   
  
The Scrum master is a liaison between the team and the business stakeholders, like the managers and marketing. The Scrum master is responsible for keeping the business stakeholders informed on the team's progress, while also providing a buffer to shield the team from distractions or requests for outside work. The business stakeholders are also a key resource the Scrum master can use, to help remove impediments blocking the team's progress. The Scrum master runs the daily Scrum meetings which is time-boxed and held at the same time and place each day. During the daily Scrum, the Scrum master asks each team member three questions: What did you do yesterday? What will you do today? And are there any impediments in your way? The Scrum master may be the one using this information to update the burn down chart and the sprint backlog, so that the team's progress is always visible. The Scrum master is also responsible for removing the impediments the team reports. By working with other teams or managers, acquiring the resources, or finding the answer to questions.   
  
The Scrum master also organizes and facilitates the established Scrum meetings, including the sprint planning, sprint review, sprint retrospective, and backlog grooming meetings. In these meetings, the Scrum master ensures the Scrum process is followed, while empowering the team to reach consensus and work together making commitments as a team. The Scrum master also works with the product owner outside these meetings to ensure the product backlog is always ready for the next sprint planning meeting. The Scrum master is also a change agent for the Scrum process, pushing its benefits throughout the organization and ensuring the process is followed within the team. They ensure implementation of the Scrum framework through teaching and coaching, and when necessary through enforcing the Scrum rules. The Scrum master does all this with no direct authority; they must earn respect and authority through their knowledge of the Scrum principles and practices. The authority given to the Scrum master is granted by the team. The Scrum master is not a manager or project leader; they do not have any people management responsibilities and are not held accountable for the outcome of the Sprint or the project. The Scrum master has process related responsibilities, with authority on how to implement the Scrum process, and as the enforcer of rules.   
  
The Scrum master role is often called a servant leader, they are a servant to the team, empowering them, helping them reach consensus, and removing impediments so they can be successful, yet they have no authority over how the team does their work. The Scrum master does not run the sprint, as the team as a whole is responsible for the outcome of the sprint, the team has complete authority on how the sprint is run. Also, the Scrum master has no ability to commit the team to work they can complete within delivery dates, though they can help the team understand how user stories are broken down into tasks with estimates. And they can ensure the team does not over commit to the work they can achieve in the sprint planning meeting. They also can urge the team to take on more work when they have fallen into complacency. The Scrum master cannot make decisions for the team. This includes technical decisions and organizational ones, like who will work which task. However, the Scrum master can mediate discussion to ensure all team members are heard and respected. In the end the Scrum master is the keeper of the Scrum process. They ensure the team, product owner, and stakeholders all follow the process. They protect the team from anyone that tries to go around the process, to keep the team focused. And they use the process to track progress, ensuring the sprint backlog and sprint burn down chart are kept up to date so they reflect the true progress of the team.

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a01_it_enus/output/html/course_transcript.html#top)

The Scrum Team Role

Learning Objective

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

* *describe the role of a Scrum Team within a Scrum software development team*

**1. Describing the Scrum team**

The Scrum team is self-organizing, with no team leader. It is whole team accountable, meaning the entire team commits to the work as a whole, and the entire team succeeds or fails to deliver that work. Normally a Scrum team is small, with seven plus or minus two people. They should work in close proximity to each other, as one of the guiding principles of Agile development is an emphasis on face-to-face conversation. Each team member should be assigned full time to the team, since the team as a whole is depending on each member to complete their part of the committed tasks. Distraction with outside task or responsibility would violate the Scrum value of Focus. The entire team must be focused on the work of each sprint. The work delivered is the product increment which is potentially shippable software. To create potentially shippable software the team needs to be cross-functional, including all the skills needed to achieve this. These skills include not only development but also QA, UI design, and technical writers.   
  
New teams may start small and add on resources with needed skillsets as they progress and learn to work together. However, even though a specific team member has the skillset for a specific task, the entire team is responsible for completing all the tasks. For instance, if QA has too many tasks to complete at the end of the sprint, the entire team needs to pitch in, to complete the testing so the sprint will be successful. The cross-functional nature of the team makes it outward-facing, as everything it needs to complete its work is located inside the team itself. The team is self-directed; team members collaborate and manage each other without direction from a manager. All this is accomplished and supported by the Scrum process. The work the team commits to in each sprint is held in the sprint backlog. In the sprint planning meeting, the team meets with the product owner and Scrum master to move the highest priority items from the product backlog to the sprint backlog.   
*Heading: The Scrum Team.  
  
Items are taken from the product backlog, which gets a team commitment, and added to the sprint backlog.*   
  
Each product backlog item must be fully defined, be small enough to be completed within a sprint, and must have acceptance criteria that definitively determines when the item is done. The team breaks each item down into small tasks and assigns a level of effort to each task. Once the level of effort reflected in the sprint backlog reaches the team's capacity, the backlog is frozen and the team commits to completing the work in the next sprint. During the sprint, the team works together to complete all the tasks in the sprint backlog. They are aided by the Scrum master, who holds the daily Scrum meetings, ensures the burn down chart and sprint backlog are continually updated to show completed and in-progress work, and removes any impediments that are blocking the team's progress. The Scrum master also protects the team from outside influences, so they can focus on their committed work. The team's work during this sprint results in the product increment, containing the functionality in the sprint backlog built upon the functionality delivered in previous sprints. The product increment is demonstrated to stakeholders at the end of the sprint in the sprint review meeting, and the entire cycle begins again with the next sprint.   
*Heading: The Scrum Team.  
  
The task is added to the sprint backlog. The output from the sprint backlog is a product increment.*   
  
An important characteristic of the team is its focus on continual process improvement as dictated by the 12th Agile principle. This activity is supported by the sprint retrospective meeting, where the team reflects on their performance in the last sprint, pointing out what worked and what didn't. The team as a whole decides on process improvements to implement in the next sprint, so their progress is always getting better. Periodically the team works with the product owner and Scrum master to groom the product backlog. In the product backlog grooming meeting, a few of the top priority tasks on the product backlog are discussed. Once they are well understood, the team assigns a rough estimate of effort to each item. If the effort estimate will not fit into a single sprint, the item is split into multiple items that will deliver functionality that can be completed within a sprint. The product backlog grooming meeting is time-boxed, as only the top priority items need to be examined. Product backlog grooming ensures the highest priority items are sized correctly and understood, so the sprint planning meeting can focus on breaking down these items into small task with estimates.   
*Heading: The Scrum Team.  
  
Process improvement takes place at the sprint retrospective meeting.*   
  
The Scrum team is accountable for product delivery, so the team is given the authority to determine the effort the work will take, how much work they can accomplish in a sprint, and how the work will be accomplished. The team commits as a whole, and succeeds or fails as a team. The team success is more important than individual member success, and since everyone on the team must work together to complete the work they committed to, teamwork is vital and is facilitated by the Scrum master. Over time, Scrum teams develop a deep form of camaraderie, and learn how to work together to accurately estimate effort and produce high-quality work. Teams that are self-organizing and have developed team unity are one of the major productivity benefits of Scrum. When possible, established Scrum teams should be kept together when a project ends, moving new work to the team rather than moving team members to new projects.

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a01_it_enus/output/html/course_transcript.html#top)

The Management Role

Learning Objective

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

* *describe the Project Manager's role responsibilities required by the Scrum environment*

**1. Describing the Project Manager's role**

When talking about how the manager's role changes when moving to Scrum, it helps to start with the manager's responsibilities that do not change. First the project manager is responsible for much more than just software development. They are responsible for communication and collaboration across the company for the new product. Working with marketing, technical support, logistics, and the executive team. The manager is responsible for controlling the work environment. They must acquire the resources needed, both hardware and software, facilities and workers. The project manager tracks the budget, and works with Human Resources to ensure company policies and procedures are enforced. The project manager is also responsible for people management, including hiring and firing, salaries, performance reviews, discipline, and dealing with outside efforts such as H-1B visa applications. Also, the project manager may be responsible for multiple projects within the same program. Within a product line this could include managing the deployment and support of one release, while the next release is in development. It could include maintenance upgrades of supporting software or hardware in the field, or managing multiple development groups working on different aspects of a very large project.   
*Heading: The Manager's Role.  
  
The consistent roles of the manager include cross department collaboration, acquisition of resources and budgets, people management, and managing multiple projects within a program.*   
  
What changes is the relationship between the project manager and the Scrum development team. In the Scrum world, the project manager is an involved role; they have an interest in the project but are not committed like the Scrum core roles. The Scrum core roles are accountable for the success of the product delivered, so they need to have the authority to create it. The Scrum team is self-directed and does not report to a manager regarding what they work on, how long it will take, or how they do their work. So the project manager does not have a command and control role, instead they need to base their relationship with the Scrum team on trust and collaboration. Also, the manager does not have the authority to direct the team to put new features in the product. Instead, they must work with the product owner to get the requirements into the product backlog. And only the product owner has the authority to prioritize the product backlog, meaning their requirement may not be seen as a high priority item.   
*Heading: The Manager's Role.  
  
The changing roles of the manager include having no command and control of development, giving input through the Product Backlog, and getting status through Scrum meetings.*   
  
How the manager is used to receiving status also changes. The manager does not call their own development status meetings, as the team must remain focused on the work of the sprint. Instead, the Scrum framework itself provides the transparency needed for stakeholders to stay informed. The manager can view the prioritized product backlog to see the work remaining on the project. And the burn down chart and sprint backlog give visibility into how the current sprint is progressing. At the end of each sprint, the manager can attend the sprint review meeting, to monitor the progress of the project and ask questions. And the manager is welcome at the daily Scrum meetings, although they must attend as an observer and are not allowed to talk during the meeting. When using Scrum, the manager should take on two new responsibilities to ensure the success of the project. First, the manager needs to back up the product owner who has the challenging job of getting requirements from stakeholders and prioritizing them. Stakeholders can include customers, managers, executives, and anyone that has an interest in the project but is not actually performing the work.   
*Heading: The Manager's Role.  
  
The new roles of the manager are backing up the product owner and the Scrum manager.*   
  
Sometimes stakeholders try to bully the product owner in order to get a higher priority on their pet feature, or to influence which features are included in the sprint. Balancing competing requirements is not easy, especially when others are pushing their requirements through direct communication or by getting the ear of upper management. The manager must assert the product owner's authority to prioritize requirements, as they own the vision of the final product. That doesn't mean the manager cannot have private meetings to discuss any disagreements with the priorities chosen, as long as in public the manager has the product owner's back. Also, the project manager must back up the Scrum master, as they protect the team from distraction, remove impediments blocking the teams work, and facilitate implementation of the Scrum process. The project manager is critical as an escalation point for the Scrum master to raise team impediments, as the Scrum master has no authority on their own.   
  
In Scrum, the project manager plays a servant leader role to the Scrum core roles, trading control for leading and helping the team be successful. This is a hard transition for many managers, but managers that hang onto the command and control mentality can sabotage a Scrum project. There are many ways a manager can cause a Scrum project to fail. Some of these include pulling team members off the sprint to work on side projects, micromanaging the team, directing who will work each task or making technical decisions instead of letting the team own how the work will be done. They could impose a schedule instead of letting the product backlog determine what is worked, and the team determine the level of effort of each item. They could add new requirements to an in-progress sprint, instead of working with the product owner to add the items to the product backlog. Lastly, a manager can make a Scrum project fail by changing the sprints duration to accommodate unfinished tasks; instead of letting the team determined the root cause of the failure in the sprint retrospective, which leads them to implement their own process improvement actions.

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a01_it_enus/output/html/course_transcript.html#top)

Scrum and Culture Change

Learning Objective

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

* *describe best practices when assigning Scrum roles*

**1. Assigning Scrum roles**

Implementing Scrum involves not just the adoption of new business processes or a framework for managing software development, but rather a fundamental transformation of the way work is managed in a company. Authority and ownership change, as does the emphasis from command and control to trust, communication, and collaboration. Scrum involves a new way of thinking, speaking, and working for both managers and workers based on the 12 Agile principles and four Scrum values. We can examine the transition in the main Scrum roles, the product owner, Scrum team and Scrum master. The product owner normally comes from product management, marketing, or is a key customer. In these traditional roles, the product owner would be the one badgering the Scrum team to commit to more work than they could accomplish, and constantly adding new requirements, all set to the highest priority. However, in Scrum, the product owner and the team work together, to build a high quality product at a sustainable pace which includes the customer's highest priority items. This is facilitated by the prioritize product backlog, which is owned by the product owner and can only have one item at the highest priority. The product owner meets with the stakeholders, collecting and prioritizing their requirements, and shielding the team from distraction.   
  
The Scrum team is different than a normal development team in that it is self-directed and self-organizing. In Scrum, the team has total responsibility for implementing the product, so they are given enhanced authority to complete the work. The Scrum team does not take direction from a manager for the work it is doing, how it is done, or when it will be done. The team determines the level of effort for each task in the sprint, as well as how much work it will take on in each sprint, allowing it to work at a sustainable pace indefinitely. The Scrum team is self-directed, deciding how the work will be distributed among themselves. The Scrum master is a new servant leader role, critical to the success of the Scrum project but not given any authority. The Scrum master owns the Scrum process and ensures it is implemented correctly. They also work with the team helping them to be productive, facilitating teamwork, and removing impediments blocking their progress. However, they cannot direct the team, other than telling them when they are not following the Scrum rules, values, and processes.   
  
This leads us to the project managers, who do not have a core Scrum Role. They are considered involved, but not committed like the Scrum master, product owner, and Scrum team. Though there are still many project manager responsibilities in the Scrum project, directing the development team is no longer one of them. This can cause tension when Scrum is introduced into a typical hierarchical management structure. And it can lead to ScrumBut, where the Scrum framework is partially implemented but other Scrum practices are ignored. This can quickly lead to not realizing the promised productivity gains of Scrum at best, or product failure at worst. Many project managers look for a way to transition into one of the main Scrum roles when Scrum is introduced. During transition to Scrum, a number of project managers take on the role of Scrum master while retaining their current role as a manager. It is a widely held belief in the Scrum world that a current project manager should not become a Scrum master for several reasons. For one, project managers control and manage development teams, while Scrum masters are servant leaders. They are not in charge, but instead facilitate empowering the team to become self-directed.   
  
A typical project manager who is used to using their authority to make things happen, normally has a difficult time transitioning into the servant leader role. Also, if the project manager transitions into the Scrum master role, everyone in their organization will still perceive them as the boss in a continuation of their old role. They will ask the new Scrum master to direct the team as in the past, which would be in opposition with the Scrum master's role. Another reason this often fails is due to the project manager's workload. The Scrum master is constantly working with, and for the team, facilitating the daily Scrum meeting and working to remove impediments. Most project managers are so busy with their management responsibilities, they don't have time to run, or even attend the daily Scrum standup. They don't have the capacity to keep up with the team status or their impediments, keep the team focused, or work to protect the sprint.   
  
Project managers can make good product owners. Interfacing with multiple stakeholders to transform a never ending flow of change request into a prioritized list is something the project manager can do well, and has most likely been doing in the past. The project manager as product owner just needs to understand they can only dictate what is worked in the next sprint. They have no authority to determine the number of items worked in the sprint, the time it will take to complete each task, or how the team creates or works the task to complete development of each item. It is a commonly held belief in Scrum that the product owner and Scrum master must be two different people. Thinking in common hierarchical structures, the two roles would be opposing each other on every issue. The product owner represents business stakeholders, who want the most functionality in the product backlog deployed as soon as possible. While the Scrum master is focused on protecting the team from business pressure, making sure they have time to concentrate on their work, being certain there was nothing standing in their way, and ensuring the product increment is potentially shippable software, fully tested and compliant before being shown to the customer in the sprint review meeting. However, in Scrum, the product owner and Scrum master have the same goals – to deliver a high quality product, with the highest value to the customer, as soon as possible. The Scrum framework creates the environment where this can happen, through collaboration, teamwork, and shared goals.

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a01_it_enus/output/html/course_transcript.html#top)

The Four Scrum Artifacts

Learning Objective

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

* *describe the four Scrum artifacts and their role in the context of the Scrum Team and Scrum Theory*

**1. Describing the four Scrum artifacts**

Scrum is often succinctly defined by describing its roles, ceremonies, and artifacts. The Scrum ceremonies are meetings that occur at set times during each sprint, each with its own purpose. And the Scrum artifacts are objects created during the Scrum project. The ceremonies give the project structure and the artifacts provide transparency into the project's progress. The four Scrum artifacts are the product backlog, the sprint backlog, the burn down chart, and the product increment. The product backlog is the record of all the work that is yet to be completed in the project. In the beginning, it is created with the customer's requirements in the form of user stories, but as the project progresses, it will also include defects, technical architecture tasks, refactoring to reduce technical debt, and spikes to investigate areas of uncertainty. The product backlog is owned by the product owner, who is responsible for updating it with new requirements and ensuring it is prioritized with the highest-priority items at the top.   
*Heading: Scrum Artifacts.  
  
The four Scrum artifacts are the Product Backlog, Sprint Backlog, Burn Down Chart, and the Product Increment.  
  
The Product Backlog contains work not yet completed and is ordered from highest to lowest priority.*   
  
The product backlog items are ordered in a single queue. There can never be more than one highest-priority item at any time. Typically, the user stories are quite large when they are added to the product backlog, capturing a broad feature – like allowing a customer to buy products from a web site. This type of user story is too large to estimate without breaking it down into several smaller, more-manageable, stories. This is done by the product owner and the team during the product grooming meeting. Only the highest-priority user stories are reviewed during the meeting. So the items at the top of the product backlog tend to be smaller, well-understood user stories, while the lower-priority items at the bottom – that have not yet been groomed – tend to be very large and not well understood. The highest-priority items on the product backlog are given a relative estimate of effort by the team. This is often in the form of a number called story points, which represents the effort needed to implement the user story.   
  
The sprint backlog contains the items from the product backlog that will be worked in the next sprint. During the sprint planning meeting, the team uses the user story's priority and the story-point estimates to select a number of items to move from the product backlog to the sprint backlog, thus removing them from the product backlog. The team and the Scrum Master must ensure the cumulative number of story points in the sprint backlog is consistent with the velocity of the team, or the amount of work that they have historically completed in the sprint. The team commits to completing the items in the sprint backlog during the next sprint, and the sprint backlog is then frozen. No more items can be added. The user stories in the sprint backlog are then broken into small tasks, normally taking from four to twelve hours to complete. These tasks are recorded in the sprint backlog and tracked daily to show the progress of the sprint.   
*Heading: Scrum Artifacts.  
  
The sprint backlog contains items to be completed in the sprint and specifies user story estimates in story points.*   
  
Sometimes the team may add, delete, or modify tasks in the sprint backlog – as the team works through the sprint and gains a deeper understanding of what is needed to complete a user story. However, no new functionality can be added while the sprint is in progress. As the sprint progresses, the user stories are moved from Defined, to In Progress, to Completed, and then Verified or Accepted. When all the tasks of a user story are complete and the acceptance criteria is met, the user story is Done – and can be included as work done in the sprint review meeting. The story points from Done stories are included in the team's velocity for the sprint. The burn down chart is a graph, posted in a public location visible to all, and is used to show the progress of the current sprint. The burn down chart shows time on the horizontal axis, and work still to be completed on the vertical axis. An ideal-effort guideline is drawn from the top left-hand corner to the bottom right-hand corner, showing what the burn down chart would look like if equal work was completed each day and all work was completed by the end of the sprint. Each day, the Scrum master or the team updates the chart with the team's current progress.   
*The Burn Down Chart details work left to do in the Sprint. It is public and provides transparency into Sprint.*   
  
The team's progress rarely matches the ideal guideline, but should show a steady burn down of the work still to be completed. If the burn down chart shows the same amount of work needs to be completed over several days, it typically means that the team is blocked and cannot advance. The product increment is the software the team has been working on during the sprint. The product increment is potentially shippable working code which has been tested and includes documentation. Any items in the sprint backlog, that were completed during the sprint, can be demonstrated as a Done feature during the sprint review meeting. However, a feature that is working in the code, but does not meet the definition of Done agreed upon by the product owner and the team, is not accepted by the product owner into the sprint review. The feature is not demonstrated to the customer in the product increment, even though the code is complete. Even though each sprint produces a potentially shippable product iIncrement, it is not a product unless the product owner decides it is ready to release. Each sprint will build on the current product increment, adding new features. In this way, the product increment will grow and become more fully featured, with each sprint always containing the customer's highest-priority features.   
*Heading: Scrum Artifacts.  
  
The Product Increment is working potentially shippable software. It is the functionality that's added incrementally during each Sprint.*

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a01_it_enus/output/html/course_transcript.html#top)

The Product Backlog

Learning Objective

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

* *describe the Product Backlog*

**1. Describing the product backlog**

In our product backlog, we have a number of product backlog items. We have user stories, we have spikes, there are some defects. Now the user stories are most common, because they represent the features and the requirements that the user wants – they're valuable to the users. But product backlogs can also contain lots of other items. They can have ideas, risks that need to be mitigated, action items that need to be tracked – all these can be placed in the product backlog. This represents an ordered list of user needs and everything else that needs to be done on the project. The product backlog represents requirements, so every one of these items is saying what needs to be done. The product backlog items do not say how the team should fulfil the item. The How is included in the paths breakdown when the team commits to the item in the sprint planning meeting. Now the product owner is the one that owns this product backlog, and is responsible for keeping it updated. They add new product backlog items to the list as they are discovered, and the product backlog is the single source of requirements on the project.   
*The Plant website is open on the Backlog page. The page contains the tabs: My Home, Backlog, Plan, and Track. Backlog is selected.  
  
The data items are listed in a table that has the columns: Rank, ID, Name, Plan Estimate, Priority, Owner, and Parent. Each row in the table has a checkbox and a settings cog.  
  
The list items are arranged in order from highest rank to lowest rank.  
  
The instructor hovers the cursor over US41 in the ID column. A pop-up box displays information for the user story "Display Plant information."*   
  
Now the product owner is also responsible for prioritizing these; you can see that they're ranked. Now note...this is a single-item queue, so there can only be one top-priority item at any time. During the sprint planning meeting, these highest-priority items are going to be moved to the sprint backlog and be committed to being developed in the next sprint, while the lowest-priority items – down here at the bottom – these are going to be the last ones to be implemented – and sometimes they may not even be implemented at all. In this way, the current sprint is always going to be having the features that are the highest priority of the customer. Now any stakeholder can view this product backlog to see what features will be implemented next, and where the feature is important to them, maybe – down here, or where they appear in the product backlog. Now this provides transparency for the stakeholders on what will be included in the upcoming sprints. Also, anyone can ask the product owner to add a new item to the product backlog. It tends to grow and change as the project goes on, and more is discovered about the project and the customer needs. However, only the product owner is the one that can prioritize where existing or new items fit in to the prioritized list.   
  
Now the user stories initially added to the product backlog are quite large, capturing broad features, like these at the bottom of the list. You can see "Pull in Customer information for marketing campaign" – that's part of a CRM Initiative. That's really hard to tell what's in there; it sounds huge to me. Also, "Use customer buying records to targeted advertising when logged in." Well that's huge too; that's just a broad requirement. However, user stories are refined and split into smaller user stories during the Backlog grooming meeting. During this meeting, the highest-priority items on the list – those at the top – are discussed, acceptance criteria is added, and the item is estimated. So you see these items here at the top of the list are much smaller, they have estimates and, if you look at them, each one includes acceptance criteria. And the user stories are sized. If they're too large, they're split into smaller user stories until they're sized to fit in a single sprint.   
  
However, the product grooming meeting only focuses on these higher-priority items at the top of the list. So the items at the top of the backlog tend to be well understood and smaller – able to be completed in a single sprint, while the lower-priority items at the bottom tend to be quite large, with very few details. Now during the sprint planning meeting, these items at the top of the list will be moved from this product backlog to the sprint backlog – as the team commits to completing the work in the next sprint. So as the top-priority items are moved to the sprint backlog, the lower-priority items will move up into the list – in the ranking. Then they will be groomed in future product backlog grooming meetings. In this way, the product backlog should always be ready for the next sprint backlog meeting, with well-understood, high-priority items at the top of the list.

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a01_it_enus/output/html/course_transcript.html#top)

The Spike Product Backlog Item

Learning Objective

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

* *use SPIKEs to discover more information in order to resolve an issue*

**1. Using SPIKEs**

SPIKEs are not unusual and come up naturally during other SCRUM activities. Here we see we have not finished User10 in fact it's blocked. If we look at the task to find out why it's blocked, we see here In-Progress that returning a confirmation of an error to the database delete transaction is In-Progress. And also related to that is the testing of that, the QA cannot test the errors. Well what happened was the developer wasn't quite sure how to do this, so he went to the product owner and they had a discussion which immediately brought up all kinds of other questions, like internationalization concerns. And so the product owner started a discussion leading to an error reporting strategy. It became apparent that the team needed to discuss the issue and come up with a holistic approach to handling errors. Our GUI developer here suggested the creation of a SPIKE backlog item to research the issue and come up with a strategy. Now we can see if we look at the **Backlog** that the product owner did create a SPIKE user story, and put it at the very top of the list as highest priority. This is appropriate since user story 10 in our sprint cannot be considered Done until we know how to display the errors. So it is blocked and any other user story will be blocked as well that needs to deal with errors. So...   
*The Plant web site is open on the Iteration Status page. The Iteration Tracking Board tracks items in a table using the columns Defined, In-Progress, Completed, and Accepted. On the web page there are also panels for Iteration Burndown, Iteration Summary, and Recent Activity.  
  
On the Iteration Tracking Board US10 is blocked. The instructor clicks the task in the US10 box. The Tasks list opens.  
  
The instructor points to TA79. The name is DEV: Return confirmation of error from database delete transaction to web page. It's state is In-progress. Then the instructor scrolls to TA84. The name is QA: Ensure correct error message is displayed when database is unreachable. It's state is Defined.  
  
In the Recent Activity panel, a message is displayed from the product owner relating to US10. The team:GUI Developer response is, "Looks like we need a Spike to do research."  
  
The instructor selects the Backlog tab on the web page. The Backlog tabbed page opens. The first item in the Backlog, with the ID US38, is the Spike, which is named SPIKE: Error Reporting Design. The instructor clicks US38.*   
  
...there are a few important things to remember about SPIKEs. First they are for researching, learning, and discovery. Even if a prototype is created it's meant to be thrown away. The SPIKE is not meant to implement any completed work for the product increment. And as such, SPIKEs do not have user story points. We can see down here, we've zero points and they do not count towards the team's Velocity. This sets them an error to grief ratio as the time spent on the SPIKE during the sprint will not be spent finishing work and achieving velocity points. The times did will be counted as if the team members working on the SPIKE were out of the office. Next the SPIKE must have clear objectives and an outcome. Here our SPIKE has a clear objective; stating the issue we are addressing. This includes questions which are showing the knowledge we're trying to gain. And we can see during coming up with these questions, when we talk about logging errors well we might need a SPIKE for logging in general. So here another SPIKE has come up that might be a lower priority. And the outcome is stated: for a holistic strategy and overall system design to handle and report errors.   
*The user story page for US38 opens. The Details tab is selected. The Details page has the sections General, Hierarchy, and Status. The General section has fields for ID, Name, tags, Description, which also includes an Outcome field, Attachments, and Owner.  
  
The Schedule section contains fields for specifying duration and points to be allocated.*   
  
Now being exact on a SPIKE's objective and outcomes keeps the team from investigating related issues that might be fun but aren't relevant to the SPIKE's objective. And the SPIKE should be time-boxed so work does not drag on indefinitely. Now the SPIKE should never take more than one sprint to complete. And often our time-boxed for shorter durations, like a day. You can see here our SPIKE has been time-boxed for, "Two developers 1 day," each working together to come up with a strategy. The team should have just enough time to get the knowledge required and put it to use to produce the outcomes in the SPIKE. Now some indications that a team needs a SPIKE is if they're having trouble estimating a user story. It could mean that they need more information to understand the technical implications. SPIKEs can also be used to cleanup technical debt that's crept into the software. Some indication of technical debt is the team assigning very high user story points to something the product owners sees as simple. Or the team's velocity steadily decreasing over time as they try to work around the problem code. A team's velocity should increase overtime as the team learns to work together efficiently and honor their commitments. Additionally the team could identify a problem area at the team's retrospective meeting and request a SPIKE to investigate or clean up the problem code.

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a01_it_enus/output/html/course_transcript.html#top)

Product Backlog Items (PBI)

Learning Objective

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

* *describe how User Stories are defined, constructed, prioritized, and groomed for inclusion into the Sprint Backlog during Sprint Planning*

**1. Working with user stories**

So here's our product backlog. We have a number of product backlog items in it. Most of them are user stories. We have some defects and we have some spikes as well. Now a PBI holds requirements needed to complete the project. Most are in the form of user stories. Each should describe what the customer wants and why it is valuable. The PBI should not describe how the team will develop the feature. And each PBI should describe something that is of value to the stakeholder. Now when they're first added to the product backlog, these PBIs are normally broad descriptions of large epic areas. This is an example here. They're going to contain a title, which should be unique. Now the format of the user story titles should be decided upon by the Scrum members; and the format is often a descriptive title, a unique identifier, or a combination of both. Here we're using both. We have US to signify this is a user story with a unique identifier, and then we have a descriptive title, "Store customer plant criteria from past purchases."   
*The Plant web site is open on the Backlog tabbed page. Several PBIs are listed. The instructor scrolls to the bottom of the backlog list and hovers the cursor over the ID US24. The user story box opens.  
  
The title of the user story is US24: Store customer plant criteria from past purchases. There are two tabs: Description and Hierarchy. The Description tabbed page is open. 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 PBI will also hold a description saying who wants what and why. Here is a Marketing Product Owner...that's who...sorry, bring that back up again. I want to store each customer's past transaction. That's what they want to do. And why do they want to do it? So I will know their taste and use it for direct marketing. Now that's a well-defined user story. It's nowhere near detailed enough to be able to estimate it in any way, but it's captured the broad requirements the Marketing Product Owner wanted to do. And he's had this hierarchy here, this hierarchy...you can have huge epics, like that CRM Initiative that can be broken up in multiple user stories. So, this is showing that the parent of this user story is the epic CRM Initiative. We can see over here – we have three different user stories from that same initiative. All are very large.Now these low-priority PBIs at the bottom of the product backlog need just this information to capture the required functionality and the user story. However, when you go to the top of the backlog, these are the high-priority items. These are the items that must be sprint ready, because the next sprint planning meeting...these items are going to be pulled – these high-priority items – to go into the sprint backlog and be included in the next sprint. They have to be sized to fit into a single sprint, with acceptance criteria clearly describing when the PBI is done.   
*The instructor selects Hierarchy. On the Hierarchy tabbed page, the parent node is US19: EPIC: CRM Initiative and the child node is US24: Store customer plant criteria from past purchases.*   
  
This sizing and adding of detail is done in the product grooming meeting. In the product grooming meeting, the Scrum members assess the high-priority PBIs, grooming them to make them sprint ready. Now a groomed, sprint-ready PBI should be concise and specific, focusing on a single slice of functionality enabled to be completed within a single sprint. Here we have our normal criteria. As an online user I want to be able to display detailed criteria about plants in the web site, so I'll know if they're right for me to buy for my garden. This was probably part of a much larger user story that was split several times, to make smaller, correctly-sized user stories that could fit in the sprint. We've added acceptance criteria here. The detailed plant information...it needs to show all the records on the plant including the image. Image can be zoomed, and multiple image can be stored. Has to be fully tested and fully documented. So this is a single slice of functionality. It's been sized so it can fit the single sprint, and it has acceptance criteria.   
*The instructor scrolls to the top of the backlog list and hovers the cursor over the ID US41. The user story box opens. An Acceptance Criteria section has been added to this user story.*   
  
Now this makes the PBI easy to understand, easy to estimate, and enables the team and product owner to write clear, definite acceptance criteria used to determine when the PBI is done. The slices at the bottom, like we saw down here – that are very large – are hard to understand and estimate, and will not be clear how to determine if the PBI is done. As they get moved up in the list...as we get more done from the top and it gets top...of the list gets put into the sprint backlog, they'll moved up and the product grooming will split them up and make them sprint ready. Now consider – each PBI holds the requirement of what must be done. And the product owner has the responsibility of prioritizing the PBIs, with high-priority items being at the top. And these are the ones that are moved into the sprint and added to the product increment. This means the product owner has the power to dictate what small slice of functionality will be added to the final product in each sprint. And each of these slices of functionality are fully described in a PBI, which includes acceptance criteria on when the development of the item is complete.

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a01_it_enus/output/html/course_transcript.html#top)

Fat Product Backlog Items (PBI)

Learning Objective

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

* *identify fat Product Backlog items (PBIs) that require splitting into smaller User Stories in a Scrum project*

**1. Identifying fat PBIs**

When the product owner shows up at the sprint planning meeting, they should have a prioritized product backlog, with the highest-priority backlog items in sprint-ready form. A sprint-ready PBI describes a single slice of functionality, which is of value to the customer. It has clear, concise acceptance criteria describing what needs to be demonstrated for the PBI to be done. And it should be easy to understand, estimate, and break down into small tasks to complete the PBI. Looking here, at this top-level PBI, this is a sprint-ready PBI. It has a small slice of functionality; you want to be able to display detailed criteria about plants in the website, so we'll know if they're right for me to buy for my garden. This is an online user, and we have good, detailed acceptance criteria. We want to show all the records for the plant, the image can be zoomed, multiple images may be shown as thumbnails which the customers can select, we want to be fully tested and document. This is a sprint-ready PBI. However, sometimes a PBI is too fat; holding too large a slice of functionality or too many loosely-related items. Slices which are too large are hard to understand. If a team is having a hard time estimating the PBI, or the acceptance criteria determining if the PBI is done is hard to write or lacks focus, it's a good indicator that the PBI is too fat.   
*The Plant website is open. The Backlog tabbed page is open and various PBIs are listed. The instructor hovers the cursor over the ID US41 at the top of the list. The user story box opens. There is a brief description as well as detailed acceptance criteria.*   
  
In the backlog grooming meeting, the top PBIs in the product backlog here are analyzed by the product owner, team, and Scrum master. One of the main task in this meeting is to identify PBIs that are too large, and split them into smaller PBIs which are sized to fit in a single sprint. So if we look at the user registration user story here – this PBI – let's identify some of the indicators that signal a PBI is too fat and needs to be split. Our user story here shows several indicators. First of all, once we're on the CRM Initiative – the second one here: capture all data activities and transactions when logged in for CRM initiative. Well that's a red flag there. Whenever you see words like "all," "any," or "every" – these are all-encompassing words. They don't give specifics and this makes this PBI not testable. What are all these user activities and transactions? If the team doesn't know what they are, the team can't build the space in the database to hold them, and be able to know what to capture. So this would have to be rewritten so that it's specific; these are the type of user activities and transactions we want to capture.   
*The instructor clicks US27. The user story US27 page opens. The general description of the user story is listed and then many items are listed in the Acceptance Criteria section.  
  
The Acceptance Criteria has three sub-headings – CRM Initiative, Security, and User Portal Experience. CRM Initiative has three listed items, Security has three listed items, and User Portal Experience has five listed items.*   
  
Look here at the CRM initiative, number 3. See attached CRM vision for other requirements. Well that's a red flag. Whenever you have any attached documents on a PBI or user story, that says that there's too much information you're trying to give. And something like a CRM vision document is going to be way too high level to give the team the specifics they need – to know what they need to build. All the information that's needed on what should be done, should fit in the PBI's description. Another indicator here, is lots of information. You can see we have things about the CRM initiative, we have things about security, we have criteria about the user portal experience, including multimedia, sending e-mail, and multimedia site tie-ins. This is just way too much information. In fact, the product owner, when they presented this PBI to the team, probably needed to give a summary before presenting it. And the description is so large that you require scrolling. That's a real good indication that there's too much information in here. So this PBI needs to be split into several smaller user stories. Remember, a sprint-ready PBI needs to be clear and concise, focusing on a single slice of functionality, with clear acceptance criteria.

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a01_it_enus/output/html/course_transcript.html#top)

The Sprint Backlog

Learning Objective

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

* *describe the Scrum Sprint Backlog*

**1. Describing the sprint backlog**

During the sprint planning meeting, the team, product owner, and Scrum master meet to determine what will be included in the next sprint. Now, here's our product backlog. It should be prioritized, with the highest-priority items containing acceptance criteria. They should have a relative size and story points, an estimate of effort, and they should be sized so they can fit in a single sprint. The work to get the backlog and the PBI sprint ready is done in the product backlog grooming meeting. When given a well-groomed product backlog, the first part of the sprint backlog meeting is spent with the product owner answering any final questions the Scrum team has about the highest-priority items in the product backlog. The Scrum team needs to understand each story's acceptance criteria, so they'll know when the work they do to implement the story will be done. And they need to be able to break down the story into detailed tasks which, when complete, will satisfy the acceptance criteria. So the team's going to select the highest-priority items, and move them from the backlog to their sprint backlog. And this signifies they will complete that PBI in the next sprint. Now once the team feels that they have reached capacity...   
*The Plant website page is open on the Plan tabbed page. It includes an Iteration planning board, which lists the backlog items, as well as the sprints to be used. Each listed sprint has a time frame specified for its completion. sprint 4 is the current sprint.  
  
The instructor drags US 41, from the top of the backlog list, to sprint 5.*   
  
...the team commits to completing the items on the sprint backlog during the next sprint. And the sprint backlog is then frozen. No more items can be added from the product backlog. The next portion of the sprint planning meeting, the team breaks down each PBI on the sprint backlog into small tasks which describe how the PBI will be completed. So if we go to our previous sprint here, and look at the tasks for this user story, we can see that there's quite a few of them, but they're simple. The tasks do not need to contain much detail, since the PBI is well understood by the team. However, each task should have an estimate on how long it will take to complete the task. When the sprint planning meeting is complete, the sprint backlog...here – we have two sprints showing...the sprint backlog will hold the team's plan for turning the PBIs they committed to completing in the sprint, into potentially shippable functionality in the product increment.   
*Then she drags two more user stories to sprint 5.  
  
In sprint 4, the instructor clicks the task for US37. The Tasks list opens. The tasks are arranged in a table that has columns ID, Name, Owner, State, Estimate(H) and To Do (H). Each listed task also has a settings cog.*   
  
Now during the sprint, the sprint backlog is used to track which tasks are in progress or completed. So, going back here to the sprint that we're currently in, we can see that the tasks are Defined. And then when someone starts working on them, they go to In Progress and when they're complete, they say they're Completed. The sprint backlog is updated each day by the Scrum master after the Scrum meeting. Now this provides transparency to the stakeholders on what's being worked, what is completed, and what has yet to be started. And the team also uses the sprint backlog themselves, as each team member determines what items they will work on each day. As the sprint progresses, the PBIs in the sprint backlog are moved from Define to In Progress – once the tasks start to get worked, to Completed. Now when the product owner decides a completed PBI's acceptance criteria is met, the PBI is considered done and can be included as work done in the sprint review meeting. The team's work during the sprint, results in a product increment containing the functionality in the PBIs completed in the sprint backlog.

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a01_it_enus/output/html/course_transcript.html#top)

The Burn Down Chart

Learning Objective

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

* *describe the Scrum Sprint Burn Down Chart*

**1. Describing the burn down chart**

This here is an example of an iteration burn down chart. The burn down chart is a graph posted in a public location; it's visible to all, and it shows the progress of the current sprint. So the burn down chart here shows time. On the horizontal axis, we have a five-day sprint and on the vertical axis, we have the number of story points – the amount of work that needs to be done. And then you have this ideal burn down line. This shows that, if you started with this much work to do and you did the same amount of work each day, you'd be able to finish at the end of the sprint. Now the real progress rarely matches the ideal guideline, but it should show steady progress. So here in sprint 2, we started out with our work and we had pretty steady burn down. We had a little bit of a slow start, but we caught up and we ended up completing all our story points. But sometimes these blockages will stay. Like if we look at **sprint 3** here, we can see we stayed blocked for a while. That was probably due to this...this blockage. This user story got blocked, we couldn't complete it. We still have eight story points here, so we didn't get all our user stories done. We didn't get those two story points. So you can see, here's the ideal burn down line and we really didn't follow it very well. We did recover fairly well, but we didn't finish our user stories.   
*The Plant website is open on the Iteration Status page. In the Iteration Burndown panel, a Burndown chart is displayed. The horizontal axis has five indicators, one for each day it represents. On the right vertical axis, work points are arranged in multiples of 1, from 0 to 8. On the left vertical axis, the amount of work is arranged in multiples of 20, from 0 to 180.  
  
The ideal burn down line starts on the first day at 180, the second day it is about 130, the third day 80, the fourth day 40, 0 on the fifth day. In this instance, the first two days were slow, but then there was improvement.*   
  
And then sometimes you'll get a strange burn down chart because of some other things you're doing. Like, if we go to **sprint 4** – and this is the current sprint we're in, and we're doing a pretty good job on our burn down, but then we had this spike. So lots of times if you have a spike, you'll get a flat rate for a while. And the reason is because a spike has no story points. Remember, spikes are used to investigate areas of uncertainty. They do not add functionality to the product increment, so they're not going to have any story points. And your burn down chart is not going to show work getting done when you have a spike, so lots of times it'll look like you're not doing any work, even though you are. Now normally, the Dev team is responsible for updating the burn down chart; sometimes the Scrum master does it. It's something that your team's going to have to decide. And as tasks are completed, the burn down chart is updated – so it always reflects how the team is performing during the sprint.   
*The instructor clicks the Iteration Task Status drop-down list and selects Sprint3. The burn down chart for sprint 3 also has a five day sprint. On the right vertical axis, work points are arranged in multiples of 1 from 0 to 7. On the left vertical axis, the amount of work is arranged in multiples of 50 from 0 to 250. In this instance, the amount of work never gets to below the ideal burn down line, so the sprint is incomplete.  
  
The instructor selects Sprint4 from the drop-down list. In this burn down chart, the work is proceeding as expected by the burn down line.*   
  
An up-to-date burn down chart is a great tool to use during the daily Scrum. If the burn down chart is not moving...down, then it's a good indication there's a blockage that the Scrum master needs to be aware of, so it can be worked. So at any point in time, the burn down chart is going to show – at a glance – the initial amount of work taken on by the team in the sprint planning meeting, and the estimated amount of work that's still remaining to complete during the sprint. So the burn down chart is also giving us our velocity of that sprint. This gives the stakeholders an early indication of how well the team is completing the necessary tasks that they need to do during the sprint, and it gives them a good indication if they're going to complete them by the end of the sprint.

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a01_it_enus/output/html/course_transcript.html#top)

The Product Increment

Learning Objective

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

* *describe the Scrum Product Increment*

**1. Describing the product increment**

One of the fundamental concepts laid out in The Scrum Guide, is the team must deliver a potentially shippable increment of code each sprint. This increment must include complete slices of product features, and it must be "Done." Additionally, the increment needs to be integrated with all the increments produced in past sprints. This is a radical mindset change for anyone used to dealing with the Waterfall process. How does a project produce a potentially shippable product in a two-to-four week sprint? Well it's worth making this mindshift. Note: "potentially shippable" does not mean it has to be deployed with all the work involved in actually releasing software, like support from logistics, marketing, product management, training tech support, printing documents and flyers, and executive oversight. The product increment enables the team to deliver products to the customer frequently, at the end of each sprint. "Potentially shippable" means the customer can see the product working as if it were released. We're not talking prototypes here; this is actual working code. This gives the customer the opportunity to point out what is not working as expected, and to give new ideas to make the product better.   
*Heading: The Product Increment.  
  
A quotation from the Scrum Guide is displayed: "Scrum requires Teams to build and increment of product functionality every sprint. This increment must be potentially shippable...the increment must be a complete slice of the product. It must be "done." Each increment should be additive to all prior increments and thoroughly tested, ensuring that all increments work together."  
  
The Scrum Guide can be accessed at the URL http://www.scrum.org/scrumguideenglish/.*   
  
If you think about it, the product increment idea is brilliant. It allows direct access to customer feedback from working software without any overhead, other than the sprint review meeting. The team does not waste time dealing with code freezes, bug triage, or an end-of-game QA bug-fix cycle – and it happens at the end of each sprint. In each sprint, the team implements user stories which hold small, single-feature product slices. The Increment is the sum of all the product backlog items completed during the current sprint, and all previous sprints as well. Each Increment is additive to all prior Increments and thoroughly tested, ensuring that all Increments work together. At the end of a sprint, every feature in the Increment must meet the definition of "Done." Every feature defined as a user story must be considered done before it can be demonstrated in the product increment. This ensures the software best-practice activities like unit test, QA, integration testing, and documentation are completed before the customer can exercise the new functionality slice in the working code.   
  
The definition of "Done" for each user story is defined and agreed upon by the product owner, Scrum master, and all members of the team. It is a combination of the acceptance criteria included in the user story and the best software practices required for releasing code. Without this definition of "Done," there is no way to ascertain if the work completed should be accepted into the product increment. The team uses the "Done" criteria during development to gauge when a task is finished. And the definition of "Done" is used by the product manager to decide when completed work should be accepted, and thus included in the product increment. Since only done user stories gain the team the user stories' velocity points, it is an important metric to the team. If the definition of "Done" is not working for the team, it can be changed for the next sprint during the sprint retrospective.   
  
Even though each sprint produces a potentially shippable product increment, it is not a product unless the product owner decides it's ready to release. As potentially releasable, Increments must be thoroughly tested, well-designed with good structure, and written in high-quality code with good documentation on how the user will operate the delivered features. The product increment is valuable to all the roles in the Scrum project. The product owner and stakeholders can evaluate the current return on investment from the functionality available to the customer at the end of each sprint. And the team's camaraderie and pride in achievement grows with the functionality of the product in every sprint, as they deliver potentially shippable code from the commitments they made as a team in the sprint planning meeting.

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a01_it_enus/output/html/course_transcript.html#top)

The INVEST Independent Criteria

Learning Objective

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

* *describe the independence criterion in the INVEST guidelines*

**1. Describing the independence criterion**

The "I" in the INVEST guidelines for writing high quality user stories stands for Independent. An independent user story can be estimated, developed, tested, and demonstrated independently. This allows it to be positioned in the product backlog by its priority, independently valued, and not dependent on any other user story before it can be implemented. According to the INVEST guidelines, a high quality user story can be understood, implemented, tested, and demonstrated on its own. It can also be independently valued by itself in the product backlog. This allows it to move up and down in the product backlog as its priority changes, without affecting any other user stories. We want the priority to drive when the user story is implemented so if its priority goes lower, it goes lower in the backlog, and as its priority goes up, it can be moved up in the backlog so it is implemented sooner. When a user story is worked should be based on its priority, independent of other user stories in the backlog. If user stories are dependent on each other, the dependency will determine when the user story is worked rather than the priority.   
*The Plant website is open on the Backlog tabbed page.*   
  
In the beginning you want to write user stories to establish the system's foundation. These early user stories will deliver a slice of functionality to the user that is demonstratable. After that, each story will be somehow dependent on functionality or architecture being in place. This is sequential dependency, which is expected. The base functionality will need to be delivered first before the enhanced features or error paths that depend on it. However, even with the sequential dependency on existing infrastructure, each user story should contain an independent feature slice. So what do I mean by that? Here's our EPIC for Website Customer Personal Portal experience. Looking at its **Children**, we see User Registration here at the top. Now this is a user story that's going to deliver value to the customer. They're going to be able to register themselves with the user ID and password. And this also includes infrastructure layer slices, some enhancements tool, you're going to need some enhancements to the GUI, to the database, there's going to be logic in there. So this is a vertical feature slice through the horizontal infrastructure layers. We want our stories to represent both demonstratable functionality and infrastructure enhancements that by itself delivers value to the customer.   
*The instructor clicks US30 in the list. User story US30 opens on the Details page. Its name is EPIC: Website Customer Personal Portal experience. The presenter clicks the Children link.  
  
There are nine children in the list. The first item in the list is US49 – User Registration. Some of the other list items are User View registration, User Modify registration, and User Delete registration.*   
  
Independent feature slices of functionality mean a feature may need several user stories to be complete. Here we have the User Registration, then we have User View registration, Modify registration, and Delete registration. All four of those will need to be done before the registration feature is complete. This is a sequential dependency, because these are sliced, so the base functionality in US49 is seen as system foundation for the view, modify, and delete user stories. These user stories, though related, are independent of each other, each describing an atomic unit of work. We don't want the concepts described in the user stories to overlap with concepts in other user stories. Each user story should be independently valuable, delivering demonstratable value on its own. What we want to eliminate are non-value dependencies. This often shows up as infrastructure stories. For example, look down here at US29, Register Customer comments on plant information.   
*The instructor hovers the cursor over the last item on the list – US29. The User Story box opens.  
  
US29 is named Registered Customer comments on plant information. The description is: As a registered Online User I want to be able to contribute my own observations about specific plants So the plant data is more comprehensive and everyone can see my comments.  
  
The instructor hovers the cursor over US54. It's name is Update database to store Customer comments. It's description is: As an administrator I want to be able to store customer's comments in the database So I can display them with the plant information.*   
  
As a Registered Online User I want to be able to contribute my own observations about specific plants. So the plant data is more comprehensive and everyone can see my comments. Okay, that's a feature that's going to deliver value to the user. But look right above it, says Update Database to store customer comments. Well if we look at that user story, as an administrator I want to be able to store customer comments in the database so I can display them with the plant information. Well for the customer to be able to contribute their comments, this is definitely going to have to be there, we're going to need the database schema updated so we can store the comments. So there's a dependency there. However this does not deliver value to the user, this is just a slice of the infrastructure, so this is a non-value dependency, we can't deliver registering customer comments until we deliver US54, but doing US54 is giving us no value.   
  
The same thing is happening up here in US28 and 26. The User authentication with personal portal, they want to be able to logon with their user ID and password so they can get to the personal portal. But right above it the administrator wants to be able to store the customers user id and password so they can be authenticated when they try to log into the website. Well just storing a user ID and password does not give value and you can only demonstrate it by showing the database. This should probably get combined with the user registration because then you would have a vertical slice through the infrastructures. You'd be able to show a user registration and the horizontal infrastructures slice of storing it in the database would be part of that. The 'I' in the INVEST guidelines stands for Independent, where each user story can stand on its own and can be developed, tested, and delivered independently. Related but independent user stories can be combined within a single sprint or they can be split across different iterations depending on their priority. That's the flexibility delivered by writing independent user stories. And with independent user stories the Product Increment will be potentially shippable at any point in development without requiring or being dependent on another user story's functionality.   
*The User Story – US26 is User authentication with personal portal has the description: As an online user I want to be able to login with my user id and password So I can get to my personal portal.  
  
The User Story – US28 – Store customer registration information has the description: As an administrator I want to be able to store the customers user id and password So they can be authenticated when they try to log into the website.*

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a01_it_enus/output/html/course_transcript.html#top)

The INVEST Negotiable Criteria

Learning Objective

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

* *describe the negotiable criterion in the INVEST guidelines*

**1. Describing the negotiable criterion**

According to the INVEST guidelines, a good user story should capture the essence of the customers requirements. It's not an explicit contract. As you can see in this user story, Secure buy transactions with credit cards, we have our user story template, as an Online Customer I want to be able to use a credit card to complete my transactions so I can buy plants from the website. This is great. It captures the essence of what the customer wants, it's easy to understand, but there's no way the user story template of, as a user I want a function so I can have a value, is going to give all the information needed to do the analysis, design, code, and test of this feature. The teams is going to have some questions. So the team, product owner, and stakeholders need to have a few conversations to negotiate what this user story is really trying to express. Now as the user story is developed through these negotiations, details are added until the team understands exactly what the story is describing enough to estimate a level of effort. One of the most important questions the team should ask is how will I know I have done that? This tells them how to write acceptance criteria to determine when the story is done.   
*The User Story US5 is open in the Web browser. The specified Name is Secure buy transaction with credit card. The Description is – As an Online Customer I want to be able to use a credit card to complete my transaction. So I can buy plants from the website. The instructor clicks Backlog. The Backlog tabbed page opens.*   
  
Now clear acceptance criteria tells the developer how the feature will be tested and gives QA a solid foundation for writing the test plans. If the product owner cannot describe how to know when the feature is complete, more research is needed. While in the product backlog, user stories should be thought of as malleable. During the product grooming meeting, user stories can be changed, rewritten, split, or even deleted, depending on the state of the project and the business needs. And those business needs change as the project progresses. With the product owner's vision, the team will fill in the details of how these user stories are implemented during the Sprint. And the customer will correct that vision or add to the details when the feature is demonstrated at the sprint review meeting, which could lead to modifying existing user stories in the product backlog or adding new ones. What is crucially important is to capture any and all agreements made during collaboration and negotiations into the related user stories, so everyone is on the same page and agreements are not lost.   
*The instructor clicks US26. The User Story page opens. The Name is User authentication with personal portal. The Description is – As an online user I want to be able to login with my user id and password So I can get to my personal portal.*   
  
So how do we write a user story that's not negotiable? Let's look at 26 here, User Authentication. Now we see the story template is, As an online user I want to be able to login with my user id and password So I can get to my personal portal. This is a essence of the customer, what they want, it's open-ended, but then these notes have been added. We want to use the attached document and force company policy on passwords, okay, password must expire and be reset every 60 days, maybe that's a policy, user IDs must be the user's e-mail address, password field must be obscured as the user is typing. Then they're telling us what kind of a protocol, HTTPS, we're using to transport the credentials. They're even going to tell us how to layout our screen for registering the user ID and for the login screen and they're going to give us those so I guess our GUI team member won't have to do any wireframes. The user ID and password will be stored on the company's secure LDAP server, unit test are mandatory, and the company test harness will be used to run unit tests nightly. Well this is saying how we're going to develop, code will be run through a nightly build, code will be stored on PlantWebsite Subversion.   
  
Well there's just way too much information here, there's no room to negotiate, and this user story includes a lot of Hows, which is a violation of the Scrum rules. Remember, the product owner gets to say what is going to be developed in these user stories, but the team gets to say how it'll be developed. If there's too much information in the description of the user story, it cuts off the conversation. The team has nothing left to ask, everything's documented. It becomes just the old school way of writing requirements, with no collaboration opportunities between the team and the end user or the stakeholders. Now there will always be some non-negotiable items, an example may be a company mandate like the strong user ID specifications, or performance requirements like the number of simultaneous users that the system has to handle. This is fine, as long as the details of the feature itself are open-ended inviting a conversation. The "N" in the INVEST guidelines stands for Negotiable. The art of writing open-ended user stories that invite negotiations enables a collaboration between the team and the stakeholders, which leads to the creation of product which satisfies the users needs, resulting in happy customers.

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a01_it_enus/output/html/course_transcript.html#top)

The INVEST Valuable Criteria

Learning Objective

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

* *describe the valuable criterion in the INVEST guidelines*

**1. Describing the valuable criterion**

A good user story must define something that is of value to the end user. This could include a feature, performance requirement, or something the user has asked for and is demonstratable in the sprint review meeting. You can see in our user story here, 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. This is a user story that has a final requirement that the customer wants in that website, it gives value to the end user. But developer user stories are also necessary, an example is a SPIKE. A SPIKE is where the team investigates, researches, and prototype so they understand a fuzzy area well enough to be able to estimate it. But SPIKES need to be time-boxed, they have to have clear goals, and an outcome that's expected. This ensures the SPIKE stays focused using only the time needed to satisfy the goals and the outcome. Letting a SPIKE turn into an opportunity for the team to do fun research and prototyping not related to the issue in the SPIKE, does not deliver value to the end user, which violates the Agile principle of continually delivering valuable software.   
*The Plant website is open on the Plan tabbed page. On the Iteration Planning Board, the instructor has opened the User Story box for US2 in Sprint 4.  
  
The name of the user story is Search for Plants by criteria on website. There is a basic description and there is specified acceptance criteria.  
  
The instructor clicks US38 in Sprint 4. The User Story page opens for the User Story SPIKE: Error Reporting Design.*   
  
Now developer user stories can also describe the architecture and infrastructure needed in order to have a potentially shippable increment. Here if we go back to our very first sprint, we have this Create plant table in the database user story, which actually was used to create the database itself. We see from the acceptance criteria we had to create our database, we had to create our schema for table structure for our plants, and then users, we want users to be able to logon to the database to view the structures, add and review plant records, and management here, that the data remains over database restart. Now since this is infrastructure, it's clearly needed, we can't have our plant web site without a database. And we could have written this user story in a way that just said we need this infrastructure item otherwise we can't have a web database. But that doesn't really show value to the user, developer user stories need to be written in a way that describes its value to the customer. Here we've said, as an online user I want to have access to detailed plant information So I will know if the plant will work in my garden. And we have done that, we have a way to store the information, all the information, the detailed plant information the customer wants, and we've given them a way to demonstrate it, so this can be actually demonstrated in the sprint review.   
*The instructor goes back to the  Plan tabbed page. The instructor clicks US13 in Sprint 1. The User Story page for US13: Create plan table in the database opens.  
  
The Description is As an online user I want to have access to detailed plant information So I will know if the plant will work in my garden. There is Acceptance Criteria specified.*   
  
This is related to the V in INVEST; besides standing for valuable, it can also refer to vertical because when splitting user stories we want to split them vertically instead of horizontally. A horizontal user story will include a complete layer of functionality - network, database architecture, persistent data, some logic, or the user interface. So this user story could have been written to just we're going to create the database and that's it, we have a database at the end. But horizontal slices are not of value to the customer. What good is a database without the user interface so it can be viewed and demonstrated. A vertical slice of functionality will have a little of each of those horizontal layers so there will be some database, some logic, some presentation in each user story. Something that can be demonstrated as working code. If we go back and look at a user story further on in the sprint here in 5, US41, this is displaying plant information on the web site. As an online user I want to be able to display detailed criteria about the plants in the website so I will know if they're right for me to buy for my garden.   
*The instructor goes back to the Plan tabbed page. Then she clicks US41 in Sprint 5. The Description on the User Story page is As an online user I want to be able to display detailed criteria about plants in website So I will know if they are right for me to buy for my garden.*   
  
You can see down here we're going to display the detailed plant information, we've got images that can be zoomed, we want to have multiple images that are stored as thumbnails so the customer can select from them. This is a vertical slice of those horizontal features. We're going to have some GUI, we're going to have some logic for the zooming, we're going to have to be able to go to the database to retrieve the information, so there's some networking, the database schema might have to be updated so that we can store multiple images. This is a small vertical slice and it's a value to user. Now it's important to remember, value is not value in the business sense that you can sell this feature. This user story is much too small a slice of functionality, it's not going to be a business money maker all by itself. And normally, several of these user stories are needed to completely flush out a full feature. However, each user story needs to provide a slice of the feature that is of value to the user. The "V" in the INVEST guidelines stands for Valuable and what is considered valuable is always in the eyes of the end user.

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a01_it_enus/output/html/course_transcript.html#top)

The INVEST Estimable Criteria

Learning Objective

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

* *describe the estimable criterion in the INVEST guidelines*

**1. Describing the estimable criterion**

The INVEST guidelines say a high quality user story must be estimable. But this isn't an exact estimate of days or hours. The estimate is a relative one; relative to the rest of the user stories in the backlog. To facilitate this, sometimes estimates are done in t-shirt sizes: small, medium, large, and extra-large. Or the Fibonacci series is popular, which we're using here to do our estimates. The estimate can be in any increment the team selects, even something as simple as the numbers one through five. Now the effort expended by the team to put a relative estimate on a user story should be minimal, but all team members must agree on the estimate before it is recorded. Now to give a good estimate the team must understand the user story. The user story must be small and focused, like US1 here, Save Plant Results List Displayed on the website. 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. And the acceptance criteria needs to be known and clear. Here we have our acceptance criteria, the customer can save the plants returned by creating a PDF of the detailed plant information that can be printed, or it can be saved to the local system.   
*The Plant website is open on the Backlog tabbed page. The instructor hovers the cursor over US1, the first PBI on the list. The User Story box pops-up.  
  
The name of US1 is Save Plant Results List Displayed on website. The description is – 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. Acceptance Criteria is also specified.*   
  
We have our user test, user document, and fully tested. Now getting a user story to this point requires negotiation between the team, product owner, and customer. If the user story is not clearly defined or scoped, the team will not be able to estimate the level of effort. And the estimate is important because it's used to determine if the user story can be completed in the single sprint. Once the team has a stable velocity, the user story estimates help them determine which user stories will be pulled in the next sprint. Let's say that our team has a velocity of seven, here they're looking at the top-level user stories, they'll have, this one's two story points, one story point, well this one's eight. They won't be able to pull this one in the same sprint because then they would have 11 story points and their velocity is seven. Also the rule that every team member needs to agree on the estimate almost guarantees discussions that'll bring out uncovered assumptions and missing requirements.   
*US1, US47, US49, US28, and US26 are the first five PBIs in the Backlog. The Plan Estimate for US1 is 2, US47 is 1, US49 is 8, US28 is 3, and US26 is 5.  
  
The instructor points to US31: customer can customize their portal home page. There is no value for the Estimate.*   
  
If the QA team member thinks the story is large, while the GUI team member thinks it's small, there's obviously something one knows or assumes the others is not aware of. More discussion is needed to fully understand the user story. This discussion around estimates brings the entire team to a shared understanding of what is required to complete the user story. Now when a user story cannot be estimated, it is because the team cannot get their collective head around the entire story. This is normally because the story is too big or has several different features in the same story. In this case, the story should be split into multiple stories. In other cases, the story has too many unknowns requiring research. US31 here is one of these: the customer can customize their portal home page. Now say our team had never done a customized portal before, where the user could customize that home page. This is an unknown for them, so they can't really come up with an estimate. In this case, the story is split into two stories, one is a SPIKE, to investigate customized portal solutions and come up with a solution, and then the second user story to actually implement it.   
  
Now a user story that is unclear will normally be estimated quite large by the team. Like US49 here: it's estimated at eight story points. In this case the user story's value and estimates may be at odds, this user story has a high priority; it's ranked third in the product backlog but it's so large it does not fit into a sprint because our teams velocity is seven. Even if a user story is high priority, if it can't be pulled into a sprint, it can't be worked. Now these large user stories probably contain multiple features which are each at different priorities. We see here we have an acceptance criteria, a whole section for CRM, and security, and the user portal experience. This user story could clearly be split into at least three different user stories, each with a smaller estimate.   
*The instructor clicks US49. The User Story page for US49: User Registration opens. The Acceptance Criteria in the Description section has three sub-sections. The instructor returns to the Backlog tabbed page.*   
  
Splitting a user story into it's component features allows each feature to be prioritized individually in the product backlog. Lastly, the team is solely responsible for estimating the level of effort. The product owner says what will be delivered by maintaining and prioritizing the product backlog, but the team says how much effort each product backlog Item will take to implement. The "E" in the INVEST guidelines stands for Estimable. The feature or requirement described in the user story and it's acceptance criteria must be clearly understood by the team in order for them to give an accurate estimate of effort.

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a01_it_enus/output/html/course_transcript.html#top)

The INVEST Small Criteria

Learning Objective

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

* *describe the small criterion in the INVEST guidelines*

**1. Describing the small criterion**

According to the INVEST guidelines, high quality user stories are small or sized appropriately. In our product backlog here, we have some nicely sized user stories. Our story points are 1, 2, 4, 8, and 16, so we have some nicely sized user stories. These are small, 3 is a little bit larger, medium size, 2, 3, but then we have this one that's 8. That's very large, that's the User registration user story. Since it's so large relative to the other user stories, let's take a look at it and see what's going on. Now we can see here the user story template's pretty simple, As an online user I want to be able to register a user name and password So I can have my own personal portal on the website. Well right off the bat that's a very large customer initiative, having your own personal portal, but let's see what happens down here in the acceptance criteria, this got out of hand, it looks like someone was just brainstorming here. They have a whole section for this CRM Initiative, when I capture all the user activities and when logged in for the CRM initiative, that's hard to understand, we have a security here, the whole logging on, and the user ID and password, and then the user portal experience – what the user wants to be able to see and do while they're logged in.   
*The Plant website is open on the Backlog tabbed page. The instructor clicks US27, which has a Plan Estimate of 8. The User Story for US27: User Registration opens.  
  
The description is: As an online user I want to be able to register a user name and password So I can have my own personal portal on the website. The Acceptance Criteria that's specified is listed in three sections – CRM Initiative, Security, and User Portal Experience.*   
  
This is a lot and it's very fuzzy. We also have some attached documentation, which is another flag that this might be way too big. We have the strong user ID specification doc, well that might be something that the company has on saying how to do a strong user ID, but you can see here the first initiative is use strong user id criteria unless customer survey shows this is negative to the customer. So that isn't even really relative unless we know we want to use it and we don't know that until the survey is over. And also there's the CRM Vision doc, well that's way too high level for the team to really understand what's required there, that's just a vision. So you can tell the team had a really hard time estimating this because they ended up with eight story points. If an estimate for a user story is large there normally isn't enough detail or too many unknowns to provide a good estimate. In this case, there's just too much user story. These types of user stories are considered EPICs and they should be split into several user stories that are small enough to understand and estimate. The team will provide more accurate estimates for smaller user stories.   
  
Now if we go back to the backlog, I would like to point out that splitting user stories is a deferred activity. Stories at the bottom of our backlog here are going to be very large and fuzzy, with very little detail. We have some EPICs, we have some large user stories, and that's fine because these are low priority, we're not going to be working them anytime soon and we really don't want to waste time defining them because as they move up in the backlog as time progresses, the whole project could change, the customer focus could change, and these could change. So it's okay for them to be fuzzy and large down here. As items at the top of the product backlog over the high priority items, these will move up and they need to be split into large user stories, they're not small enough to really do an estimate for yet, but they're much easier to understand and discuss to see how to split them up even further. Once they get to the top of the product backlog up here, this is where they have to be sized appropriately because these are about ready to be worked, to be pulled into a sprint to be worked. They need to be discussed, details added, acceptance criteria has to be good, and they need to have a good estimate assigned to them.   
*The instructor selects the Backlog tabbed page.*   
  
Now if we look at a good user story, here is US1, this is Save Plant Result List Displayed on the website. This is only two user points; 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. That's very focused...the customer can save the list of plants returned, we can create a PDF file so they can print it or they can save it to their local system. Unit tests are complete, User documentation complete, Fully tested. Now this is good acceptance criteria, it's very focused, it's very easy to understand, it's easy to test.This user story, the team will be able to estimate it and complete it very quickly. Additionally, the task for the story will be complete early in the sprint allowing QA to test it during the sprint while the larger user stories are being worked. And so QA can be utilized throughout the sprint rather than having the backlog like you do a lot of times in other processes, where everything gets thrown over to the QA team at the end and they really don't have time to test it all fully. So remember the "S" in the INVEST guidelines stands for small or sized appropriately. The quality of a user story is definitely enhanced by keeping it small.   
*The instructor clicks US1. The US1: Save Plant Results List Displayed on website User Story page opens. The description is: 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 Acceptance Criteria is Save information – Customer can save list of plants returned as a printable PDF or a PDF saved to the local system. The other criteria are Unit tests complete, User documentation complete, and Fully tested.*

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a01_it_enus/output/html/course_transcript.html#top)

The INVEST Testable Criteria

Learning Objective

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

* *describe the testable criterion in the INVEST guidelines*

**1. Describing the testable criterion**

According to the INVEST guidelines, high-quality user stories are testable. The acceptance criteria in the user story, must include a definitive description of what is considered "Done". Here in our user story, Search for plants by criteria on the website, we have acceptance criteria that tells us, the customer should search by all these different properties for their plants. They can search on multiple properties in a single search. When they submit the system's going to return them a list of plants that match that criteria, including a thumbnail image, the plants common and Latin name, and then customers can remove plants from the list to narrow down the results. This user story cannot be considered "Done" unless the acceptance criteria is satisfied. And the only way we know if it's satisfied is by verifying it through tests, but acceptance criteria is not test cases. Acceptance criteria answers the question, how will I know when I'm done with this user story? Test cases lists the steps needed to test the functionality.   
*The User Story page for US2: Search for plants by criteria on website is open. The description is: 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. There is also an Assumptions section and Acceptance Criteria is specified.*   
  
Now when writing acceptance criteria, individuals from three different disciplines are required, to collectively understand what should be tested and how. The product owner, the team members who will be doing the development, and the ones who will be doing the testing, all three must be in agreement on what needs to be done in order to satisfy the user story. Acceptance criteria should be detailed enough to define when the user story is satisfied, yet not so detailed as to inhibit collaboration. And all agreements should also be captured in the user story. So there are no misunderstandings if someone new is added to the team and ends up doing the work. The acceptance criteria is used to create the test plan, and it's a best practice to write the test plan before developing the code. It makes it very clear to the team what they need to do to move this user story to "Done". If the team knows exactly how the feature will be tested, they will know exactly how to implement it, which increases the team's productivity. A way to ensure the user's story is testable, is have the customer that submitted this user story tell the team how to test it. If the customer cannot describe how to test a feature, it means there's more discussion needed. If the test criteria is difficult to write, then there's a problem with the user story.   
*The instructor selects the Backlog tabbed page.*   
  
Let's look at a user story that has some issues. Now if the test criteria is difficult, it could be that the user story is asking for horizontal layer or technical components, instead of a vertical slice of functionality. Here you see the CRM initiative, well this is a whole component all by itself, there's no vertical slice of functionality that's demonstrable to the user and it's quite large, which is another reason you might have problems writing test criteria. This user story might be too big or too complex with too many disparate functions in a single story. Here we see there's three different areas, CRM, Security, and User Portal Experience. This should really be broken up into multiple stories, or this story could be dependent on other user stories that have not yet been completed. When writing the test criteria be aware of descriptions that are not testable. Words must be exact. Words like "all" mean you never know when you're done testing. Here if we look at the User Portal Experience, number 5, site should be linked to any and all multimedia sites for customers to access. Well there is so many multimedia sites and new ones could be added everyday, how do you know when you're done testing this? And words should not be fuzzy, like here, on number 1, Interface should be user friendly, clean, and easy to use. Well that sounds nice but fuzzy words like quickly, nice, clean, user friendly, manageable they're all subjective and they create a description that is not testable.   
*The instructor clicks US49. The User Story page for US49: User Registration opens. The Description section has an Assumptions section. The Acceptance Criteria in the Description section has three sub-sections – CRM Initiative, Security, and User Portal Experience.*   
  
Now when the user story is new and very large like an epic, fuzzy words like this can promote a conversation and negotiations. In this case, they would help the team and stakeholders, come to a single idea of what it means to be user friendly or clean. But they don't belong in that acceptance criteria, which you're going to be using to write your test plans. Now all these agreements that the team comes to, must be written in the user stories acceptance criteria in clear, concise, testable language. Also what's essential to capture notes, assumptions, and agreements in the user story. Too much information in a description can lead to missing information and acceptance criteria. Here we see, customers should be restricted to more than five comments in a single day. Well that's something that should be in the test plan, but down here, logged in user should be able to submit comments on plant files relaying to their own experience. This doesn't bring that in and so it could be left out of the test plan. Ensure that the assumptions captured in the description are also reflected in the acceptance criteria. The "T" in the INVEST guidelines stands for Testable. Writing clear, testable acceptance criteria is crucial as it's the only way the team, product owner, and customer, can agree on when the user story is done.

[Back to top](http://xlibrary.skillport.com/courseware/Content/cca/sd_spsc_a01_it_enus/output/html/course_transcript.html#top)

© 2017 Skillsoft Ireland Limited