User Stories in Scrum

In the world of Scrum, there is no recommended method for stocking the product backlog. Mike Cohn states in his book, *User Stories Applied: For Agile Software Development*, "I have had great success expressing Scrum backlog items in the form of user stories. Rather than allowing product backlog items to describe new features, issues to investigate, defects to be fixed, and so on, the product backlog must describe some item of value to a user or to the product owner." This Analyst Brief discusses the benefits of user stories, how to gather user stories using Scrum methods, and how to write good user stories.

Benefits of User Stories in Scrum

One of the most useful aspects of using user stories to define the product backlog is that user stories are written in an easily understandable language. Because of this, it is easier for the product owner to re-prioritize the product backlog—trade off decisions between one user story and another are much easier to make whenever understandable terms are used. User stories are also useful because they force each item in the product backlog to deliver value that the customer can conceptualize and development or users can test. Sprint planning meetings will go much faster if the value of an item in the product backlog doesn't have to be dissected, explained, and then explored to find its benefit, and users will be able to see immediate value in every release.

It's very simple to know when, or what parts of a sprint are completed during Sprint Reviews whenever utilizing user stories because completed items will clearly portray user or customer value and be broken down into steps, or tasks. Tasks are work that will need to be done on each software element a user story touches. During the daily Scrum, user stories are useful for keeping day-to-day work and activities customer and user focused because they so clearly demonstrate something the user wants. At the beginning of a Scrum project, the Scrum team doesn't know exactly what the final product should be, so user stories are also useful to remind the development team why whatever is being worked on is being developed, and for determining if something has been over or under developed.

How to Gather User Stories with Scrum

The process of gathering user stories using an Agile method is very different than the process for gathering requirements while using a Waterfall method. Unlike in waterfall, where requirements are often elicited at the beginning of the project and then the users aren't contacted again until testing, users and customers will be involved throughout the entire Scrum process.

1. **Determine User Types**. Determining who will be using the new software should be the first step in writing users stories. User stories are most effective when written by the people who will be using the new software because they will have the most knowledge on what they think is valuable. (Remember, user stories must portray user value.) For example, two different types of users for a college grade reporting software could be student and professor. As many representatives from the various types of users as possible should be contacted to write user stories so that the most well-rounded and value delivering software possible is developed.
2. **Determine Epics**. Before starting on development, it is important for the Scrum team to gain an understanding of the problem the solution is to solve and develop a high level concept of the proposed solution. High-level concepts of the proposed solution are called Epics, and are made up of a group of related user stories. Some organizations use epics whenever there is features that have not yet been analyzed enough to break down into user stories, which should be done before bringing an epic into a sprint to reduce uncertainty. Reaching agreement on epics up front prevents many problems down the road and prevents significant scope creep that often happens on agile projects. Epics should be prioritized based on which deliver the most business value, and be sufficiently understood to determine the outline scope of the product and produce high-level budgetary estimates, but no more.
3. **Capture User Stories**. After epics have been defined and agreed to, it is time to start capturing requirements at a high level of detail in the form of a user story. Focus on gathering user stories for each user type to come away with the most representative user stories possible. Whenever it is possible, user stories should be written directly by the users. However, depending on the type of project and organizational specifics, user stories may be written by development team members and/or the product owner, as well.

   The development team will often write "obvious" user stories that are very abstract and contain large, assumable features. User stories written by the development team alone will most likely be broken down into multiple user stories as development progresses and will be subject to more elaboration than user stories written users or the product owner. Whenever the product owner writes user stories, they are expected to use business analysis elicitation techniques such as interviews, questionnaires, observations, and user story writing workshops to ensure that the user stories accurately reflect user need.

How to Write Good User Stories

Good user stories are written in easily understandable terminology that captures the "who," "what," and "why" of a requirement, along with acceptance testing procedures for developers to determine when the user story has been completed. Whenever writing a user story, remember that a good user story...

A. contains "just enough" information,

B. portrays the six attributes of INVEST,

C. can be written in a simple format, and

D. consists of a Card, Conversations, and Criteria.

A. User stories are meant to contain just enough information to be ready just in time for development. Just in Time" (JIT) has been applied to manufacturing for several decades, and is the practice of providing just enough user story details, just in time for development. Providing too much detail is wasteful and time consuming because if a user story changes, the details will need to be changed, as well. However, providing too little detail often results in significant rework and forces developers to make incorrect assumptions. They should be written to be barely sufficient; that is, user stories should only contain the absolute minimum amount of information required to enable development and allow testing to proceed with reasonable efficiency. The rationale for this is to minimize time spent on anything that doesn't add value to the end product. For example, if you're willing to defer many of the ultimate decisions about product capabilities and characteristics to the developers, you don't need to include as much information in the user story. If you want to describe exactly what you expect to be delivered, more complete specifications are necessary.

B. It is also recommended that every user story adhere to the attributes of INVEST and be independent, negotiable, valuable, estimable, small, and testable as described below.

Independent: Can it be built independently? The user story must be defined, analyzed, built, and managed separately from all other features.

Negotiable: Are the details of how the user story is to be implemented negotiable? The user story is not an explicit contract. Ensure tradeoffs can be made between user stories if necessary.

Valuable: Will it deliver business value and meet an organizational need? The user story must be valuable to stakeholders.

Estimable: Can the project team determine a good approximation of effort? It should be possible to determine the relative size of each user story to ensure costs can be estimated and work activities can be planned.

Sized Right: Is it appropriately sized to be feasibly built? All user stories should have the same level of detail. This element of the acronym relates more to the tasks involved with each user story.

Testable: Can you clearly identify a test to ensure the user story has been delivered? The project team must be able to verify whether the user story was successfully delivered.

C. Whenever writing user stories, it is useful to maintain the following format recommended by Mike Cohn because it is simple, easy to understand, and clearly identifies the user, action, and desired result:

"As a...I want to...so that...

D. A well-formed user story consists of three components: the Card, Conversations, and Criteria.

Because user stories are elaborated later on in the Scrum process, they are typically recorded on notecards (hence the user story component names). A 3x5" note card should efficiently contain all the information needed for the Card. The format of "As a...I want to...so that..." should contain all the information needed for the card component.

Conversations represent discussions between the developer and the product owner or other stakeholders. Before user stories are about to be placed into a sprint, the development team or product owner should talk to customers about their user story (or a user story that pertains to their business domain) for elaboration. Elaboration from and conversations with users are necessary because a user story may be difficult to interpret, some background knowledge could be needed for implementation, or the requirements could have changed since the story was written.

The final component of user stories is acceptance criteria. User story criteria generally include acceptance criteria defined before development begins to determine when each user story is finished and working as the user intended. Acceptance criteria can be used to demonstrate a user story's boundaries, and are often thought up during conversations between the product owner, development team, and users. It is recommended that users are the ones to write acceptance criteria because each user story was written from a user's point of view—the tests ensuring a user story's completion and satisfaction should be written by the users, too.