Table of Contents

How Do We Split a User Story?

> Basic answer is “By end user value - ask yourself 'What kind of feedback do you expect from the customer at the Sprint Review.'”

User stories have a life-cycle. When we start into a release planning process, we might only have a “top 10” list of features required for this release. These are typically called Epics. Scrum Teams are told that they have to complete an entire user story before moving on to the next one in a Sprint and further that they should have between say 5-10 user stories per sprint. How do we get from the Epic to User Stories of this size? The answer is that we increasingly detail the requirements by splitting the Epic and subsequent User Stories into smaller and smaller User Stories.

Approach

The recommendation is that, where possible, you split large user stories into smaller ones but focus on still providing end user value even when you do the split. In other words as you divide stories up ensure there is some valuable functionality with each story. This contrasts with the approach a typical development organization would take when given a large amount of work to do where the instinct is to split the effort along architectural boundaries (data access layer, middle tier, client tier). The problem with the architectural approach is that:

- Development needs to continue for a long time, developing and integrating all layers, before anything is provided to an end user.
- There is no proof that the architecture will work until the end when end user functionality is exposed.

In other words, resist dividing the story up along “technical” boundaries as this will result in iterations of development with nothing complete (and valuable) to the end user.

The question is then “how to I split a story if I don't use architectural boundaries?” One basic approach is to take the “Acceptance Criteria (or Conditions of Satisfaction)” for a User Story and use that to split the story up. So for example if the user story was “As a customer I want to pay with a credit card so I can get the item on sale”, the “Acceptance Criteria (or Conditions of Satisfaction)” might be “Accept Visa; Accept MasterCard; Accept American Express” and so on. If it was determined that the Story is too big to fit in a Sprint, then you could split the Story as “As a customer I want to pay with a Visa card so I can get the item on sale” with its own Acceptance Criteria (or Conditions of Satisfaction), and so on. Keep going until you have high priority items in the Product Backlog that are right-sized for a Sprint.

This type of thinking leads to the following general possibilities (using a different example):

- Data boundaries: Split a large stories along the boundaries of the data supported by the story. For example the story for a banking application might say “As a user, I want to get detailed account information”. If this is too big given the amount of accounts a user might have, you could split this into “As a user, I want to get detail savings account information …”, “As a user, I want to get detail

check account information …", "As a user, I want to get detailed car loan information …" and so on.
- Operational boundaries: Split large stories based on the operations that are performed within the story. A common approach to doing this is to split a story based on common CRUD (create, read, update and delete) boundaries. For example the story for a Swim Stats system might include the story "As a coach, I can manage the swimmers on my team". If this is too big to fit in a sprint, we could split the story into "As a coach, I can add new swimmers to my team", "As a coach, I can edit information about swimmers already on my team" and "As a coach, I can delete swimmers who are no longer on my team."
- Cross-cutting concerns: Consider removing cross-cutting concerns (such as security, logging, error handling, and so on) and creating two versions of the story; one with and one without the cross-cutting concern. For example, a user story might be "As a user, I am required to log in with a user name and password before using the system …". During the team discussion, it becomes clear that there is an implied set of restrictions on the password in this user story - it must have 8 characters, must have at least 1 digit, must be encrypted when transmitted and so on. While none of this is particularly time consuming it might be a problem in aggregate. This story could be split into a secure and non-secure version with the non-secure version supporting the basic log in and the non-secure version including planned security precautions.
- Performance constraints: Consider splitting a large story by separating functional and non-functional aspects. In other words follow the advice of Kernighan and Plauger (1974) - "Make it work, then make it faster" (to which I would add "Make it work, make it faster, then make it easier").
- Mixed priority: Sometimes a story is actually a number of small sub stories that are of different priority. These should be split along priority lines. For example, look at the normal log in story "As a user, I am required to log in with a user name and password before using the system …". The implication might be that "If a user enters a invalid password three times in a row, they are denied access until they call customer service." In most cases, the second part of this story is of lower priority and should be split from the main story.

To make sure you have something valuable ask yourself "Will the Customer be able to provide feedback on this functionality at the Sprint Review?" If you cannot figure out what kind of feedback the customer will provide, you probably have a user story that is more technical than from the business value that is being provided.

Want to Know More?

- Splitting Story Flowchart - An excellent reference for patterns for splitting stories.
- How Do We Build and Maintain Context When All We Have Is a Backlog List?

Consultant, Tools, UserStory, Stories, Splitting, FAQ