

Procedure_206 – Writing Effective User Stories

Last Updated: 10/24/2016

Audience: CWDS Service Teams

Frequency: As needed

Purpose:

To provide service team members with training on how to write effective user stories for their product backlog. Anyone can write a user story. But it is the service manager's responsibility to make sure a product backlog of users stories exist and are prioritized according to customer needs.

Definitions:

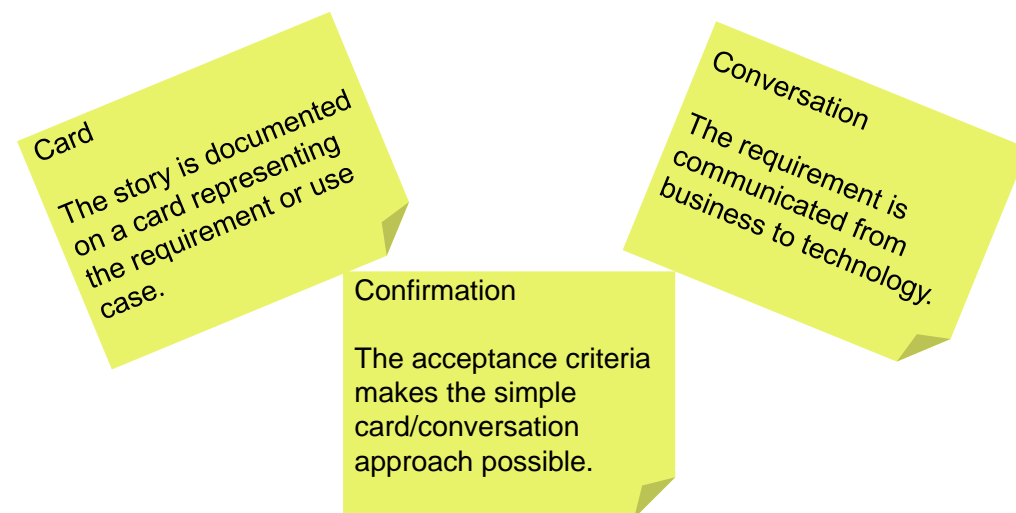
A **Product Roadmap** outlines the project scope elements and the description of each functional element is elaborated upon as learning occurs within each successive iteration. The backlog is a prioritized list of the functionality to be developed in a product or service and is comprised of epics and features, decomposed into user stories that “fit” within an iteration or sprint.

An **Epic** a large user story that covers large amounts of functionality. Epics will later be decomposed into smaller stories that fit more readily into a single iteration.

A **User Story** is a simple statement describing the value of functionality from a user's perspective. User Stories focus on the “what” and not the “how”. While traditional requirements try to be as detailed as possible, a user story is defined incrementally in three stages:

1. The brief description of the need
2. The conversations that happen during backlog refinement and iteration planning to firm up the details
3. The tests that confirm the story's satisfactory completion.

The 3 C's of a User Story:



It is impossible to know all the requirements for a product or project in advance. Emergent requirements are those that the users cannot identify in advance - every project has emergent requirements!

All user stories should strive to follow the INVEST model (<http://xp123.com/articles/invest-in-good-stories-and-smart-tasks/>). The INVEST mnemonic for agile software projects was created as a reminder of the characteristics of a good quality Product Backlog Item (PBI), commonly written in user story format.

Mnemonic	Value	Description
I	Independent	We want to be able to develop in any sequence.
N	Negotiable	Avoid too much detail; keep them flexible so the team can adjust how much of the story to implement
V	Valuable	Users or customers get some value from the story
E	Estimable	The team must be able to use them for planning.
S	Small	Large stories are harder to estimate and plan. By the time of iteration planning, the story should be able to be designed, coded, and tested within the iteration.
T	Testable	Document acceptance criteria, or the definition of done for the story, which lead to test cases.

There are other stories besides User Stories:

- **Research Story:** When a team wants to do some web research, poking around in existing code. Outputs are generally research/decision documents.
- **Spike Story:** when a team does some coding to bring life to their Research. Outputs are generally a proof of concept. It is perfectly acceptable to have ‘throw away code’ at the end of the sprint.

Research (←Action) Smart Distance algorithm for multi-paragraph web pages **so that** (←Value) we learn what search functionality exists

Prototype (←Action) a Smart Distance algorithm for multi-paragraph web pages **so that** (←Value) we understand what search functionality will work best for our customer

- **Device/System Story:** While the user story is the primary, and most desirable form of story, not every system under development interacts with an end user. Sometimes, the “user” is a device (example: printer) or other system (example: transaction server).

The goal is to continue to ask who the story is for to try to get to the perspective of the end user/customer. Non-functional requirements fit nicely into this type of story (security, scalability, usability).

As a web crawler, I want a URL dictionary without duplicates to dead links so that the crawling process is faster.

- **Epic Story:** Epic Stories are developed at the high-level.
 - Example: As a user, I want to log into the system so that my information can only be accessed by me
 - It may be necessary to decompose an Epic into smaller, more manageable user stories: For example:
 - As a registered user, I can log into with my user name and password so that I can trust the system
 - As a new user, I want to register by creating a user name and password so that the system can remember my personal information
 - As a registered user, I can change my password so that I can keep it secure or make it easier to remember
 - As a registered user, I want the system to warn me if my password is easy to guess so that my account is harder to break into
 - As a forgetful user, I want to be able to request a new password so that I am not permanently locked out if I forget it

Why use User Stories:

1. They express user value
2. They avoid introducing detail too early that would prevent design options and inappropriately lock developers into one solution
3. They avoid the appearance of false completeness and clarity
4. They allow for small enough chunks that invite negotiation and movement in the backlog
5. They leave the technical functional to the architect, developers, testers, etc.

User Story Format:

The appropriate user story format is:

As a (who needs this?)

I want (what is needed?)

So that (what is the value?)

Examples:

- As a consumer, I want shopping cart functionality to easily purchase items online.
- As an executive, I want to generate a report to understand which departments need to improve their productivity.

A story should be small enough to be coded and tested within an iteration—ideally just a few days. When a story is too large, it is called an Epic. Backlog items tend to start as epics when they are lower priority.

Too Broad	Too Detailed	Just Right
<ul style="list-style-type: none"> ▪ A team member can view iteration status. 	<ul style="list-style-type: none"> ▪ A team member can view a table of stories with rank, name, size, package, owner, and status. ▪ A team member can click a red button to expand the table to include detail, which lists all the tasks, with rank, name, estimate, owner, status. 	<ul style="list-style-type: none"> ▪ A team member can view the iteration's stories and their status with main fields. ▪ A team member can view the current burndown chart on the status page, and can click it for a larger view. ▪ A team member can view or hide the tasks under the stories. ▪ A team member can edit a task from the iteration status page.

Acceptance Criteria:


All user stories should have acceptance criteria defined. Acceptance criteria is the confirmation that the user story has been satisfied. Acceptance criteria are short and easy to understand statements that are testable. The statements detail what the user expects the product/feature to do. Acceptance criteria enable developers to understand that they have satisfied the user requirement.

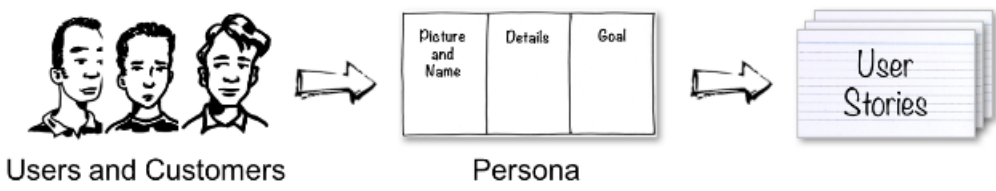

Process Steps for developing User Stories:


1. Gather Ideas
 - a. Hold focus groups, conduct interviews, solicit feedback from customers
 - b. Listen to your customers and be prepared for undeveloped ideas
2. Refine Ideas
 - a. Interpret user statements into user needs
 - b. Focus on the “what” and not the “how”
 - c. Avoid Assumptions and adding extra detail
3. Develop user stories

- a. Write in correct syntax
- b. Develop acceptance criteria for each story
- c. Develop tasks that are needed to complete the story. If the tasks are too onerous, decompose the story into smaller parts.

10 Tips for Writing Effective User Stories:

Step	Instructions
1	<p>Users Come First</p> <p>As its name suggests, a user story describes how a customer or user employs the product; it is written from the user's perspective. What's more, user stories are particularly helpful to capture a specific functionality, such as, searching for a product or making a booking. If you don't know who the users and customers are and why they would want to use the product, then you should not write any user stories. Carry out the necessary user research first, for example, by observing and interviewing users. Otherwise, you take the risk of writing speculative stories that are based on beliefs and ideas—but not on data and empirical evidence.</p>  <p style="text-align: center;">User or Customer Product</p>
2	<p>Use Personas to Discover the Right Stories</p> <p>A great technique to capture your insights about the users and customers is working with personas. Personas are fictional characters that are based on first-hand knowledge of the target group. They usually consist of a name and a picture; relevant characteristics, behaviors, and attitudes; and a goal. The goal is the benefit the persona wants to achieve, or the problem the character wants to see solved by using the product.</p> <p>For more detail, please review Procedure 215 - Developing User Personas</p>

Step	Instructions
	 <p>Users and Customers → Persona → User Stories</p>
3	<p>Create Stories Collaboratively</p> <p>A user story is not a specification, but a communication and collaboration tool. Stories should never be handed off to a development team. Instead, they should be embedded in a conversation: The product owner and the team should discuss the stories together.</p> <p>You can take this approach further and write stories collaboratively, for instance, as part of your product backlog refinement process. This leverages the creativity and the knowledge of the team and results in better user stories.</p>  <p>Ideas, Feedback, Data → Product Owner and Dev Team → User Stories (New or Update)</p>
4	<p>Keep your Stories Simple and Concise</p> <p>Write your stories so that they are easy to understand. Keep them simple and concise. Avoid confusing and ambiguous terms, and use active voice. Focus on what's important, and leave out the rest.</p>
5	<p>Start with Epics</p> <p>An epic is a big, sketchy, coarse-grained story. It is typically broken into several user stories over time—leveraging the user feedback on early prototypes and product increments.</p> <p>Starting with epics allows you to sketch the product functionality without committing to the details. This is particularly helpful for describing new products and features: It allows you to capture the rough scope, and it buys you time to learn more about how to best address the needs of the users. It also reduces the time and effort required to integrate new insights. If you many detailed stories in the product backlog, then it's often tricky and time-consuming to relate feedback to the appropriate stories and you have to be careful not to introduce inconsistencies.</p>
6	<p>Refine the Stories until they are Ready</p> <p>Break your epics into smaller, detailed stories until they are ready: clear, feasible, and testable. All development team members should have a shared understanding of the</p>

Step	Instructions
	<p>story's meaning; the story should not be too big and comfortably fit into a sprint, and there has to be an effective way to determine if the story is done.</p> <div data-bbox="321 380 886 575">  <p>Big, coarse-grained, sketchy Small, detailed, specific</p> </div>
7	<p>Add Acceptance Criteria</p> <p>As you break epics into smaller stories, remember to add acceptance criteria, which are pre-established standards or requirements a product or project must meet. These criteria define the boundaries and parameters of a User Story/feature and determine when a story is completed and working as expected. They add certainty to what the team is building.</p> <p>Acceptance criteria complement the narrative: They allow you to describe the conditions that have to be fulfilled so that the story is done. The criteria enrich the story, they make it testable, and they ensures that the story can be demoed or released to the users and other stakeholders.</p>
8	<p>Develop Tasks for your User Story</p> <p>A task is a step taken by a service team that is required to fulfill the requirements of the user story. Tasks are fine-grained and independent. Tasks are used to organize the activities needed to complete each story, and in some cases, to give the story visibility.</p> <p>You need to add tasks that help you achieve the story's Definition of Done and meet the Acceptance Criteria of the story.</p> <p>For more information about assigning tasks to a User Story, see the training presentation Assigning Tasks to User Stories.</p>
9	<p>Keep your User Stories Visible and Accessible</p> <p>Stories want to communicate information. Make them visible, for instance, by putting them up on the wall. This fosters collaboration and creates transparency. A handy tool to discover, visualize, and manage your stories is the Sprint Task Board.</p> <p>For more information, see Procedure 213 - Developing a Sprint Task Board.</p>
10	<p>Don't Rely only on User Stories</p> <p>Creating a great user experience requires more than user stories. User stories are helpful to capture product functionality, but they are not well suited to describe the user journeys and the visual design. Therefore complement user stories with other techniques, such as, story maps, workflow diagrams, storyboards, sketches, and mockups.</p>

Step	Instructions
	<p>Additionally, user stories are not good capturing technical requirements. If you need to communicate what an architectural element like a component or service should do, then write technical stories.</p> <p>Finally, writing user stories is worthwhile when you develop software that's likely to be reused. But if you want to quickly create a throwaway prototype or mockup to validate an idea, then writing stories may not be necessary.</p> <p>Remember: User stories are not about documenting requirements; they want to enable you to move fast and develop software as quickly as possible—not to impose any overhead.</p>