User stories aren’t use cases. By themselves, user stories don’t provide the details the team needs to do their work. The Scrum process enables this detail to emerge organically (largely), removing the need to write use cases.

Are user stories the same as use cases?
When running our Writing Great Agile User Stories workshop, I’m frequently asked “So – are user stories the same as use cases?”. Often it’s a business analyst who asks the question; they’re accustomed to working with use cases, and are wondering where use cases fit in a Scrum project, and if they’re replaced by a user story.

Looking around the web, there’s consensus that use cases and user stories are not interchangeable:

Alistair Cockburn: A user story is to a use case as a gazelle is to a gazebo
ExtremeProgramming.org: User stories serve the same purpose as use cases but are not the same.
Mike Cohn: User stories aren’t use cases.

My standard answer is that user stories are centred on the result and the benefit of the thing you’re describing, whereas use cases are more granular, and describe how your system will act. And then I say “Just bear me – it will all be clear in soon”. But I thought it was time to dig further down into this.

Use cases and user stories
Let’s start with some definitions.

A user story is a short description of something that your customer will do when they come to your website or use your application/software, focused on the value or result they get from doing this thing. They are written from the point of view of a person using your website or application, and written in the language that your customers would use.

A user story is usually written using the format canonised by Mike Cohn: As an [actor] I want [action] so that [achievement]. So, for example: As a Flickr member, I want to set different privacy levels on my photos, so I can control who sees which of my photos.

A use case is a description of a set of interactions between a system and and one or more actors (where ‘actor’ can be people, or other systems: for example, both online shoppers and PayPal can be actors). They are usually created as documents, and generally include this kind of information:

- Use case title
- Rationale/description/goal
- Actor/user
- Preconditions (the things that must have already happened in the system)
- Standard path or main success scenario (what will usually happen, described as a series of steps)
- Alternate paths or extensions(variations on the above/edge cases)
- Post conditions (what the system will have done by the end of the steps).

At first blush, use cases look like a much better way of writing requirements than user stories. How will a team be able to implement something as wafty as: As a Flickr member, I want to set different privacy levels on my photos, so I can control who sees which of my photos? Won’t it take some rigorous use cases to detail the requirements for the system? And that’s usually the point when someone in the workshop asks that question.
Writing use cases to flesh out user stories in Agile projects is certainly not unheard of (see here, and here). But it becomes clear as we move through the workshop that user stories are just the start of a process of understanding what the team is making that, by the end of its course, covers off everything a use case would have told you, but in an organic manner.

**Acceptance criteria**
User stories aren’t just single sentence affairs. The product owner also writes acceptance criteria, which define the boundaries of a user story, and are used to confirm when a story is completed and working as intended. For example, if this is your user story: *As a conference attendee, I want to be able to register online, so I can register quickly and cut down on paperwork*, the acceptance criteria could include:

- A user cannot submit a form without completing all the mandatory fields.
- Information from the form is stored in the registrations database.
- Protection against spam is working.
- Payment can be made via credit card.
- An acknowledgment email is sent to the user after submitting the form.

Writing the acceptance criteria is the first step of fleshing out the details of your user story.

**Sprint planning meetings**
In the sprint planning meeting, the product owner presents the user stories from the top of their product backlog (ie. their highest priority features) and the team commits to the stories they will complete in the sprint. As the product owner presents the stories, the team will ask questions to further clarify the user story and the acceptance criteria. Assumptions will quickly be confirmed or corrected, and any ambiguity about the requirements starts to disappear.

This assumption and ambiguity erasure continues as the team estimates the stories (if five people on the team rates a story as a 2, and one person rates it as a 5, there’s probably some questions that need answering). And it’s repeated again as the team writes the individual tasks for each story.

**Standups**
We’ve been fortunate in our Scrum projects, in that our product owners generally commit to attend the team stand-up. This is another chance for the team to ask questions, and also to make the product owner aware of restrictions, issues and opportunities are appearing as the story progresses.

**Wireframing**
I do the wireframing for some of our projects, and usually I start by talking to the product owner about the story, and sometimes making some paper or whiteboard sketches. I turn these into wireframes and then there’s normally a couple of quick iterations with the product owner as we ask each other questions, get feedback from other people, and hopefully squeeze in some user testing.

More recently, I’ve started reviewing the draft wireframes with the designers and developers working on the story. This helps flag up any questions they have, or restrictions I might not have been aware of. After the wireframes are approved by the product owner, I’ll brief the designers and developers again if needed.

**Design and development**
Although most of the details have been thrashed out during the wireframing more can crop up at this stage, and there are often more questions for the product owner about exactly how they want the backend of the system to behave. Pair programming is useful here, because two sets of eyes on a piece of functionality mean yet more questions and clarifications.
No user story is submitted for acceptance by the product owner until the acceptance criteria are satisfied and the definition of done is met.

**Overall**
This might sound like a lengthy process. In reality, it’s just what a Scrum team does all day. Rather than one person labouring over the use cases, the team works together to surface and satisfy all the requirements. The product owner can refine the original acceptance criteria in response to new information throughout a user story’s progress.

**And finally, in conclusion**
There are exceptions, of course – and there are times when the upfront research needed for use cases is important (I’ve got a blog post brewing on this). But my advice would be: don’t start writing use cases until your team specifically asks for them. And if your team does ask for them, spend some time in a retrospective digging into what they’re not getting from your current processes (for example – are the acceptance criteria unclear; is the product owner unavailable; are you working with shitty documentation for another system). Then decide as a team how to fix the root problem.

**Further reading**
- Advantages of user stories for requirements – Mike Cohn
- Requirements 101: User stories vs use cases – Andrew Stellman

Want to know more about user stories? Check out our introductory series:
- An introduction to user stories
- Adding acceptance criteria to user stories
- User stories and the development process
- Bringing stakeholders on board through user stories
- Creating user stories with story mapping
- Improving user stories with a definition of ready.

info@boost.co.nz / 04 939 0062
Level 8, 75 Ghuznee St, Wellington
©2016 boost