




PRODUCT OWNERSHIP ([HTTPS://BLOG.THIGA.COM.AU/.PRODUCT-OWNER](https://blog.thiga.com.au/.PRODUCT-OWNER))

Write great user stories

 Hugo Geissmann (https://blog.thiga.com.au/author/thiga_australia_admin)

 October 27, 2017

(<https://blog.thiga.com.au/product-owner/write-great-user-stories>)



User story format

The correct way to write user stories is a hotly debated subject. Throughout this creed, we'll try and give you some tips and pointers to get started in what we think is the right way. However, always remember that everything here is a recommendation – the right way to write user stories should always be the way that works best for you and your team.

As with a lot of agile concepts, the development team has the last word regarding the readiness and clarity of a user story. As a Product Owner, you need to be working to ensure that your stories fit your team, and not the other way around. No matter what you've read about user story formats and best practices, you need to listen to your team.

With that said, this is a generally accepted user story format that you can get started with, get inspired by, and iterate on.

A user story should contain:

- A title: "Payment for a customer with a VISA card".
- A narrative description built around the structure of "As a", "I need to", "In order to". For example: "As a customer with a VISA card, I need to enter my card details in order to pay for the goods in my shopping cart". Using this approach has the benefit of ensuring you're writing your users stories with your users in mind, by stating what they are trying to do and why.
- A list of acceptance criteria. These help you mention all the details that are not mentioned in the short description above, and help you express all the things you expect to see developed around the core of what you're describing in your user story. For example, in our case, an acceptance criteria could be that the VISA card format has to be verified as the customer writes it (more on this later).

In some teams, a bare-bones user story should be enough to get your team developing the right feature. In others, user stories may resemble a mini-specification. Of course, the detail you decide to include in each story can vary depending on its complexity. We usually observe that the closer a Product Owner is to his team, the shorter and more concise the user stories are. However, as well as you may get along with your development team, every user story you write should pass every step of the INVEST framework:

- **Independent:** A user story should be self-sufficient, that is to say, it should not have any dependencies on other stories as this can cause testing and planification issues.
Negotiable: A story should always be up for discussion and iteration. Your development and testing team need to have their say.
Valuable: Each and every story must bring tangible value to your users, which is why they're expressed by a user-focused narrative description.
- **Estimable:** A user story has to be sufficiently clear and precise to be estimable by the development team.
- **Small:** Always try to write user stories that describe development work that is achievable in a relatively short amount of time and that don't take on too much complexity in one swoop. This avoids tunnel vision and ensures you'll be able to deliver a new version of your product frequently.
- **Testable:** A user story tells an objective-driven narrative, meaning you should be able to test every story once completed.

Of course, the INVEST framework is a set of guidelines that you will not always be able to follow. For example, you'll sometimes need to have interdependency between user stories, meaning you'll have to prioritise them carefully. The framework is a tool to help Product Owners improve their user stories by trying to follow a set of guidelines.

Always keep in mind that a Product Owner must never suppose that a user story contains the right solution to his or her users' problems. Each story should be written after gathering the necessary information from business teams, users themselves, and anyone else able to offer insight.

Also, don't hesitate to rely on mockups, diagrams, flowcharts and other visual tools to explain and clarify your user stories. Always keep in mind a story is a narrative that is meant to spark discussion and ensure everyone on your team is on the same page and understands the underlying needs and reasons behind development efforts.

To come back to our initial point: never forget that the right format of user story is the one that works for your team. Avoid religiously following a framework or adopting a method you read about in a blog article without adapting it. Frequently ask your development team if they're happy with the stories

they're working on, and how you could improve them. This is something we encourage you to talk about with your team in your regular retrospectives.

Acceptance criteria: the three amigos workshop

Acceptance criteria are a set of criteria that will enable you to stamp a user story as done once development is finished and the criteria are met. They will guide the development and testing teams in their work, so it's important that everyone has their say and understands them perfectly. The three amigos workshop aims to enable just that.

Before the workshop, the Product Owner should have finished the acceptance criteria for the stories to be discussed and shared them with the other participants. Developers and testers should have read them and arrive to the workshop with a list of questions and suggestions.

After an initial discussion during which the Product Owner will answer various questions about the development work described in the user stories, you should collaboratively draft a list of what needs to be tested for each story. The workshop isn't about building a broad testing plan, but mainly about identifying the main scenarios that need to be tested, and how they'll be tested (manual test, unit testing...).

What should come of the Three Amigos:

What will be produced is a list of acceptance tests, which will be familiar to those working in a Behavior Driven Development (BDD) environment. This is usually formalised after the workshop by either the Product Owner or testing team. They can be written in the following way:

- **GIVEN** (a context)
- **WHEN** (the user does certain things)
- **THEN** (the observable result of the given actions)

Once your acceptance criteria are written this way, we recommend having everyone reread them to make sure the team is still on the same page. This form of syntax, named Gherkin, is used in many frameworks that enable automatic feature-level testing, like jBehave (<http://jbehave.org/>) and Cucumber (<https://cucumber.io/>).

When using this form of acceptance criteria, you have to write precise criteria with real data, not generic phrases. If we go back to our VISA card payment user story, the acceptance criteria could be written as follows:

- **GIVEN** a user with a VISA card number 4672 6677 8183 8471, valid until 01/2020 and with the cryptogram 436,
 -
 - **WHEN** the user enters his card number 4672 6677 8183 8471
- And** the expiry date of 01/2020
- And** the cryptogram 436

THEN the payment is accepted

And the user is redirected to the payment confirmation page

These acceptance criteria are great, but the workshop also has the benefit of ensuring everyone has the same comprehension of the work to be done in each user story. Now that the whole team shares the same vision of the story, it should be easier to develop and should require less back and forth.

Be careful with the Amigos

As with everything we suggest, these workshops are to be used only when they make sense. It's not always necessary to have three people to define acceptance criteria for a basic feature, and even less so for a bugfix (the team already knows what isn't working!). However, we highly recommend using this workshop, especially at the beginning of a new project, or the onboarding of a new team member, to ensure everyone has a shared vision of how a story should be written. Hopefully this will mean you'll have less comprehension issues in the future, and spend less time rewriting stories or explaining them to your team.