Reading: Anatomy of a user story

Note: this article is not directly derived from the BABOK Guide and is a supplementary material. Information contained in the article serves for professional development purposes but is not mandatory to pass IIBA exams.

Anatomy of a user story

Like many, for years I have been using user stories as one of the main means of capturing requirements. Over that time, I have learned a few tricks about structuring the stories that made my life easier. I'd love to share them with you.

This time I'm going to focus on the structural aspect of writing user stories, consciously leaving other aspects aside (such as the topics of story slicing, story mapping, the level of detail, or whether to use stories at all).

Following the basic format

It may sound obvious, but just thoroughly following the basic most popular format of writing stories can improve your process of gathering and elaborating requirements. Just follow this:

As a <persona> **I want** <a capability> **So that** <value>

I am calling this out because quite often I have witnessed the teams skipping elements of stories when preparing the backlog. The excuses people use vary from "*it is obvious, why bother*" to "*we have no time*" to even "*we have no space in the ticket fields in the system*". So instead of a nice narrative of a story, they end up with one-two word declarative statements like "login", "quick payment", etc…

There are two reasons I don't like it when people skip elements of a story:

1. First, following the format you connect the important elements of the requirement together: the customer we are helping ("as a") to the need ("I want") and the why ("so that"). The "customer" part of it keeps the connection to the end beneficiary helping to focus on the customer and the "why" helps to remember the expected value you are going to deliver. Skipping things does not help keeping the important elements in mind and deliver valuable products.
2. Second, half-baked stories do not contain enough information to pick them up. A person new to the project will never understand what is expected. More than that, the same team picking them up a few sprints later will not remember what they intended to deliver.

Adding context

A very helpful technique that I use all the time is adding context to user stories. Typically, it means adding a new key word to the basic structure:

As a <persona> **who** <context> **I want** <a capability> **So that** <value>

Obviously, one can start all the user stories with “*As a user I want..*” It will follow the letter of the law of story writing, but not the spirit of it. When describing the customer in a story, you develop a connection with the end user/beneficiary, so the final product is designed and built for them. Adding a “*who*” part can become crucial to design a better solution. Let’s have a look at this example:

***As a** user*
***I want** to see the customer support SLAs*
***So** I know how long it may take to process my request*

The solution can be a long table of support SLAs on a page grouped by services. However, if we add context the solution may immediately change. Compare to this:

***As a** user **who** has just submitted a claim*
***I want** to see the customer support SLAs*
***So** I know how long it may take to process my request*

The solution will magically change to just one figure which is tailored to the specific request category and is much more personalised, actually solving the customer’s problem.

Some people take it to an extreme getting rid of the “user” bit altogether and replacing it with a contextual statement. This approach is called “job stories” [1]. Honestly, I quite like it: “***When**<context>**I want**<motivation>**So that**<outcome>*”.

Regardless of the format, make sure you take the context into account.

Know your purpose

I’ve seen teams using user stories to solve completely different problems. Some teams use the stories to slice the backlog into development work ready to start. Some teams use the stories to scope the research and design activities. Some teams do both :)

In an ideal world one story will have the solution designed and built in one iteration, so such a problem will not exist. This is not however what the majority of projects encounter. Often, a degree of research and experimentation happens before we know what is it we need to develop.

As a result, many teams actually need to write stories twice: first, to feed and scope the research and design process. Second, to feed the development. Depending on the purpose, your stories will have a different level of fidelity and a different focus.

A development ready story will focus on the capability that needs to be delivered. This capability, or a feature, will be described pretty well and the solution will be tested against it.

A story that you use to scope the design process (I use “design” in a broad sense here — it may be visual, functional, or technical design) will focus on the context and the outcome. Quite often you don’t yet know the feature you want to deliver or you don’t want to focus on it (this is when job stories format comes handy ;)).

Compare these two.

As a *user* ***who*** *has just registered*
I want *to watch a welcome video*
So *I can learn how to use the system*

This is very detailed about the functionality to be built and can be a good dev story. It assumes that the welcome video is a preferred solution to the user's problem. It could however look different:

As a *user* ***who*** *has just registered*
I want *to be presented with a welcome tutorial*
So *I can learn how to use the system*

This is much less prescriptive about the solution and is open for interpretation — exactly what you want to start shaping the solution and do some research to find the best solution option (it may be a video, or a pdf brochure, or an interactive guide, etc.).

In conclusion

A few key elements to get a structurally good story:

1. Decide which format you want to use and stick to it where possible.
2. Consider context in your stories
3. Understand who you write the stories for and which purpose they are going to serve.

References

1. https://jtbd.info/replacing-the-user-story-with-the-job-story-af7cdee10c27