How to Write a Use Case

In: Requirements Models and Specifications By: Laura Brandenburg

As someone with a business background, do you find that many business analyst jobs involve some interaction with the IT team or IT systems? Are you wondering how you'll fulfill those responsibilities if you don't have much (or any) technology knowledge?

As a technical professional, would you be interested in learning a technique for sharing exactly what the technology can do for the organization? Have you been asked to talk about the technology using less "tech speak" but aren't quite sure what that means?

In what follows, I describe use cases – a commonly used business analysis technique that captures requirements for a software application. If you have an analytic mindset, **you'll be able to write use cases even if you don't understand or want to use the technical jargon**.

(By the way, if it's your first time here, you might want to check out our free business analyst career planning course.)

What Are Use Cases?

Use cases are a type of textual requirements specification that capture how a user will interact with a solution to achieve a specific goal. They describe the step by step process a user goes through to complete that goal using a software system.

Use cases capture all the possible ways the user and system can interact that result in the user achieving the goal. They also capture all the things that can go wrong along the way that prevent the user from achieving the goal.

For example, one use case I've created for just about every project captures the user logging into a software system. You could also have use cases for Manage Account, Create Order, or (why not?) Read Content on Bridging the Gap. Think of any piece of software you use, via the web or installed on your computer, television, or smart phone. There is a use case and probably many use cases written or waiting to be written to describe the system's functionality.

How Do You Write a Use Case?

Use cases contain the following elements:

- **Name** – A clear verb/noun or actor/verb/noun descriptor that communicates the scope of the use case.
- **Brief Description** – A brief paragraph of text describing the scope of the use case.
- **Actors** – A list of the types of users who can engage in the activities described in the use case. Actor names should not correspond to job titles.
- **Preconditions** – Anything the solution can assume to be true when the use case begins.
- **Basic Flow** – The set of steps the actors take to accomplish the goal of the use case. A clear description of what the system does in response to each user action.
- **Alternate Flows** – Capture the less common user/system interactions, such as being on a new computer and answering a security question.
- **Exception Flows** – The things that can happen that prevent the user from achieving their goal, such as providing an incorrect username and password.
- **Post Conditions** – Anything that must be true when the use case is complete.

We cover each section of a use case in more detail and provide several work samples in our Use Cases and Wireframes virtual course.

Use Cases Capture What the Software Does

In the vast majority of contexts, use cases should be used to capture what the software does, not how the software does it. Another way of saying this is that they are implementable – meaning a software developer clearly knows what needs to be built – without specifying the implementation details such as what coding language to use, how to connect various technical components, or what database fields to add (as those would go into a data dictionary).

One of the most common problems I see when reviewing use cases from our course participants is that the use cases either contain too much detail or not enough.

- Use cases that do not include enough detail tend to be more like business processes. They capture what the business user needs to do, but not what the software needs to do to support the business user.

- Use cases that include too much technical detail tend to be more like system design documentation. They read like pseudo code and sometimes lose the context of the user actions. (To be fair, the use case structure can be used to capture technical design details and technology systems can be "actors". But out of the few hundred use cases I've written in my BA career, I've only used them to capture the detailed technical design 2 times that I can explicitly remember. It can be done, but it's not common for a business analyst to do it.)

Use Cases Are Primarily Textual

One of the more common misconceptions among participants is that use cases must include complex visual diagrams, such as UML activity diagrams or actor-use case diagrams.

You can certainly add visuals to your use cases – I often find that simple workflow diagrams and user interface wireframes nicely complement the content of a use case and make them even easier for stakeholders to understand and provide feedback on.

But in essence, **use cases are textual models that capture the requirements in context.** That's why the use case templates we include as part of our BA Template Toolkit (they are also included with our Use Cases and Wireframes virtual course) are in Word and Excel.

Use Cases Help You to Get to the Right Details

As a business user, if you can write a use case that describes what a piece of software does or needs to do, you'll know enough about the "technology" to talk to technologists about software systems. You'll also go down a path of asking and answering many important questions about the requirements.

As a technical professional, if you can do the same, you'll learn how to talk about technology without using the technical jargon that is not relevant to the business community.

How to Learn More

Click here to read about my love affair with use cases

(I’ve been using them since my very early days as a business analyst and have a lot of personal stories to share. And that’s a big part of the reason I decided to capture my approach in our Use Cases and Wireframes virtual course.)