Use Case Examples -- Effective Samples and Tips

By Darren Levy updated on October 14, 2014

These days the term "use case" isn't just something used by business analysts, product managers and developers. It's become a common-place term for start-up enthusiasts and those in the venture market. You might here someone ask, "..tell me about the use case of your business idea." And the recipient would know to use the use case not as the elevator pitch, but to tell the story and typical sequence of events that describe their consumer's/user's involvement with the business.

In the technology world, your use cases are only as effective as the value someone's deriving from them. What seems obvious to you may not be to your developers or customers. The success measurement for an effective written use case is one that is easily understood, and ultimately the developers can build the right product the first time.

A great way for writing effective use cases is to walk through a sample use case example and watch how it can be leveraged to something complex. By absorbing the meaning of use case diagrams, alternate flows and basic flows, you will be able to apply use cases to your projects. In some of the tips below, we'll use eBay features for example use cases.

Before we get into this blog post for writing effective use cases for software development and product management, let's quickly define a use case. If you had to find a single word it's synonymous with, I suppose you could say "scenario", but it's really much more than that. It's the particular types of scenarios that are made up activities. When you're talking to your friends about a new obscure start-up, a great question that I like to ask is "What's the primary use case for the customer?". It puts someone on the spot to tell a story from the customer's perspective from customer acquisition, to purchase, and on to engagement. Anyway, now let's get on to writing up some use cases!

Tip 1. When creating use cases, be productive without perfection

Tip 2. Define your use case actors

Tip 3. Define your "Sunny Day" Use Cases (Primary Use Cases)

Tip 4. Identify reuse opportunity for use cases

Tip 5. Create a use case index

Tip 6. Identify the key components of your use case

Tip 7. Name and briefly describe your use case

Tip 8. Create the use case basic flow

Tip 9. Create the use case alternate flows

Tip 10. Produce your use case document

Tip 11. Sample Use Case Model Diagram

Tip 12. Do you need User Stories?

Tip 13. Agile Development with Use Cases

Tip 1. Be productive without perfection

Be agile, be lean, don't be afraid to make mistakes. Often so many new product managers think being perfect will impress their audience, but having strongly written use cases with a few mistakes is FAR better than an over complicated detailed list that confuses and bores an audience.

When it comes to writing effective use cases, you don't need to be a perfectionist and concern yourself with getting it right the first time. Developing use cases should be looked at as an iterative process where you work and refine. You can always refine it later, so again, don't go for perfection from the get-go. Loosen up and have some fun while you're doing it. Remember, humans are reading your use cases, not a bunch of robots, so keep it interesting.

Tip 2. Get a good working list of your use case actors

What is an actor? (no not Brad Pitt in this context) Any "object" or person that has behavior associated with it. Generally, the users are actors but often systems can be actors as well.

There are possibly over a dozen actors that interact with Ebay, from buyers and sellers, down to suppliers, wholesalers, auditors, and customer service. But we're going for grass-roots, so who are the basic users of Ebay? BUYERS and SELLERS. So lets put them down as our first actors. (The visual notation in the figures below is based on UML -- Unified Markup Language for Use Cases)

Do you notice how the actors aren't John and Sue which would be people? While John may be a seller and Sue may be a buyer, an actor is a Role. And a role in this case would be that of a buyer and that of a seller. Now that things are clicking, lets throw some more actors on your paper just so we can try and identify more possible users.

Now we have a bunch of actors. Wait a minute? Paypal? That's not a person. An actor can be a system, because a system plays another role in the context of your new system and has goals and interacts with other actors as you will see later.

Tip 3. Define your Sunny Day Use Cases

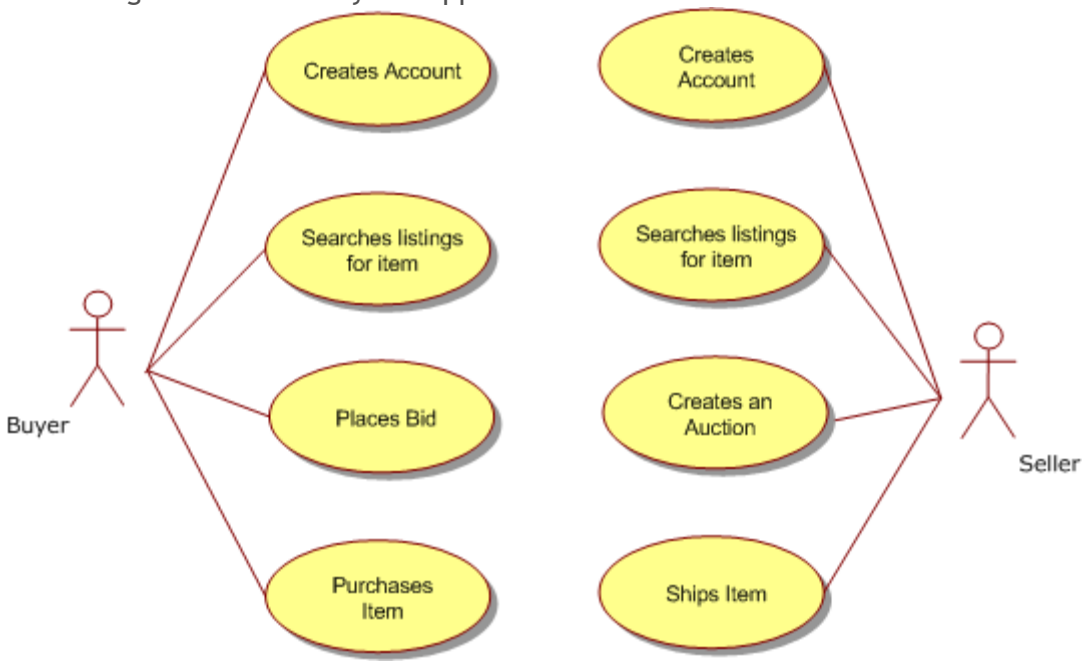
For those of you who haven't heard the expression, "Sunny Day" use cases, it is in reference to the use cases that are most likely going to occur when all goes well. These are somtimes referred to as your primary use cases. You always want to focus on the sunny day scenarios first because you can then pivot off these and figure out your "rainy day" scenarios (or edge cases) later.

Use the 80/20 rule -- if you write an exhaustive list of all possible use cases, typically 20% of the use cases will account for 80% of the activity. The other 80% of the use cases would support 20% of the activity.

In my experience in various offices, the perfectionists will say, "well what about this? isn't that possible?" referring to an edge case. The product manager should be able to discern a common use case from the edge case and prioritize accordingly. So, once you are done with your sunny-day use cases, distribute it among your project team and get consensus that you have covered them all.

Now Collect your Rainy Day Use Cases After you have a well-defined list of your primary use cases, you'll want to collect the list of edge cases (rainy-day) and with the help of the product manager or stake-holder, prioritize them in terms of likelyness. It should be a business question as far as how much software development costs do you want to spend on

something that is not likely to happen.

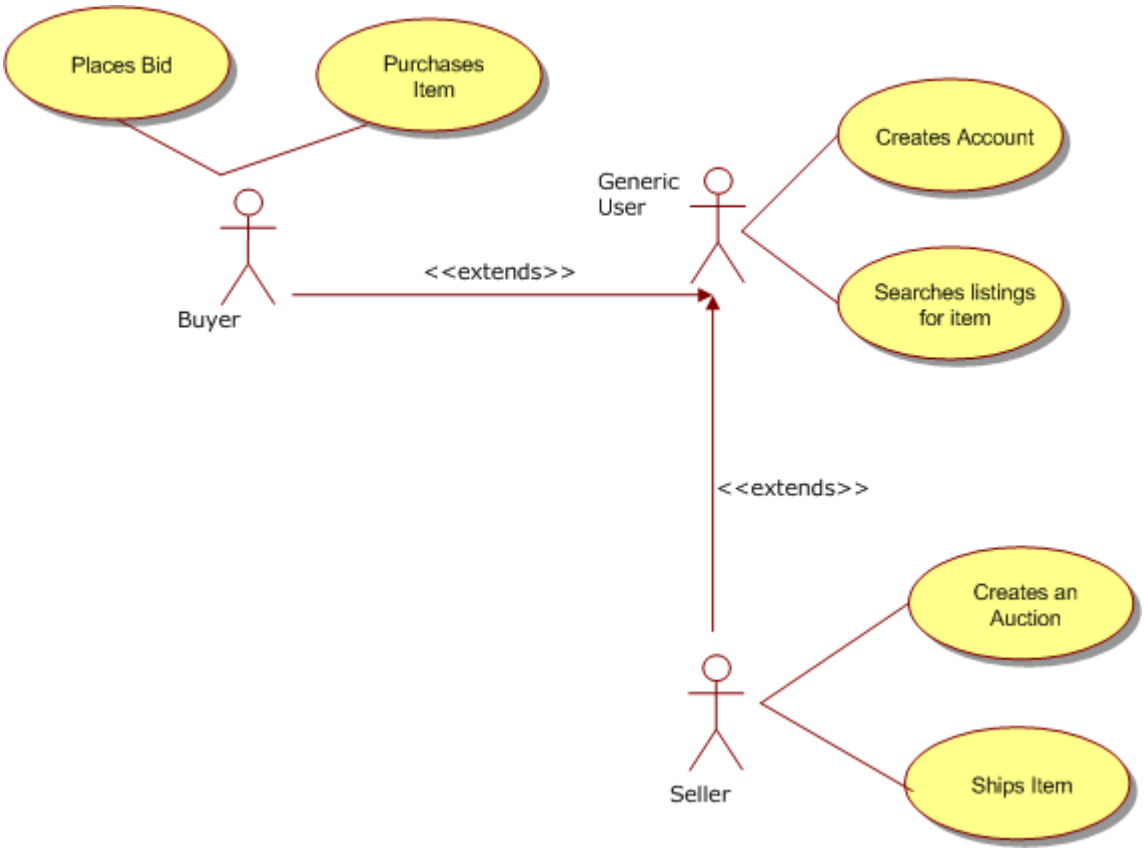


Tip 4. Identify reuse opportunity for use cases

In this step, you are going to cross the bridge into object modeling. Don't get overly concerned about terms like generalization, inheritance and extends. The goal of this Ebay use case example is to keep it understandable so we will explain this concept in terms of the example.

What does the word general mean? Something is broad and not as detailed. Generalization is when you "inherit" from something general and then add more detail. A "person" is very general. A "man" is still general, but not as general as a "person". You can say that a "man" inherits behavior and atributes of a "person".

Look at the requirements management use case diagram above and you will see there is duplicate behavior in both the buyer and seller which includes "create an account" and "search listings". Rather than have all of this duplication, we will have a more general user that has this behavior and then the actors will "inherit" this behavior from the new user.



The above use case example diagram illustrates that a generic user creates accounts and search listings and that a buyer and a seller have their own behavior but also have the behavior of the generic user. The benefits of generalization are that you eliminate duplicate behavior and attributes that will ultimately make the system more understandable and flexible. We will see in later steps that this inheritance applies both to use cases and to the actors.

Tip 5. Create a use case index

After producing your initial visual list of use case actors and goals, we can take this list and create an initial use case grid which provides the basis for the use case index. Every use case will have various attributes relating both to the use case iteself and to the project. At the project level, these attributes include scope, complexity, status and priority. (see the sample image below)

Use Case ID	Use Case Name	Primary Actor	Scope	Complexity	Priority
1	Places a bid	Buyer	In	High	1
2	Purchase an item	Buyer	In	High	1
3	Creates Account	Generic User	In	Med	1
4	Searches listings	Generic User	In	Med	1
5	Provides Feedback	Generic User	In	High	1
6	Creates an auction	Seller	In	High	1
7	Ships an item	Seller	In	High	2

This use case index should be used by the project team to define the use cases against. It will serve as a master inventory to help writ effective use cases for the requirements phase of the project.

Tip 6. Identify the key components of your use case

The actual use case is a textual representation illustrating a sequence of events. In our use case example, you will see that there are several components of a use case which we will review. In the mean time, review the table below to get a basic understanding of what is in the use case and then we will review each element as we progeress through our use case example.

Use Case Element	Description
Use Case Number	ID to represent your use case
Application	What system or application does this pertain to
Use Case Name	The name of your use case, keep it short and sweet
Use Case Description	Elaborate more on the name, in paragraph form.
Primary Actor	Who is the main actor that this use case represents
Precondition	What preconditions must be met before this use case can start
Trigger	What event triggers this use case
Basic Flow	The basic flow should be the events of the use case when everything is perfect; there are no errors, no exceptions. This is the "happy day scenario". The exceptions will be handled in the "Alternate Flows" section.
Alternate Flows	The most significant alternatives and exceptions

Tip 7. Name and briefly describe your use case

Now that you have a general understanding of what a use case consists of, we are ready to start creating our use case. Typically, while the name of your use case is being discussed, people will start briefly describing the use case. Use plain english and keep it simple. Getting back to our use case example, I will begin with use case #1 from step number four.

Use Case Number:	1
Use Case Name:	Buyer Places a Bid
Description:	An EBAY buyer has identified an item they wish to buy, so they will

place a bid for an item with the intent of winning the auction and paying for the item.

Tip 8. Create the use case basic flow

The basic flow of a use case represents the most important course of events or what happens most of the time, sometimes referred to as the 'Happy Day Scenario' because it is what occurs when everything goes well -- no errors or exceptions. Another reason why the basic flow is so critical is because it's much easier to fully comprehend the exceptions once the norm is understood and if the basic flow represents 70% of the system, the development staff is much more prone to implementing the correct code in the first pass.

For our use case example, the basic flow should be to describe the happy day scenario for your use cases such as "placing a bid". For a consumer to play a successful bid, what is the primary flow when everything goes as planned. An effective use cases needs to have the basic flow before moving forward with writing the alternate flows.

Tip 9. Create the use case alternate flows

The basic flow is the key ingredient to your use case and some can argue that you can stop once you're done with the basic flow. It really depends on the level of detail you wish to achieve. However, providing more detail to the consumers of your use case is always a good thing.

The alternate flows providing the following:

- An exception or error flow to any line item in your basic flow
- An additional flow, not necessarily error based, but a flow that COULD happen

A few examples of alternate flows are:

- While a customer places an order, their credit card failed
- While a customer places an order, their user session times out
- While a customer uses an ATM machine, the machine runs out of receipts and needs to warn the customer

Tip 10. Produce your effective use case document

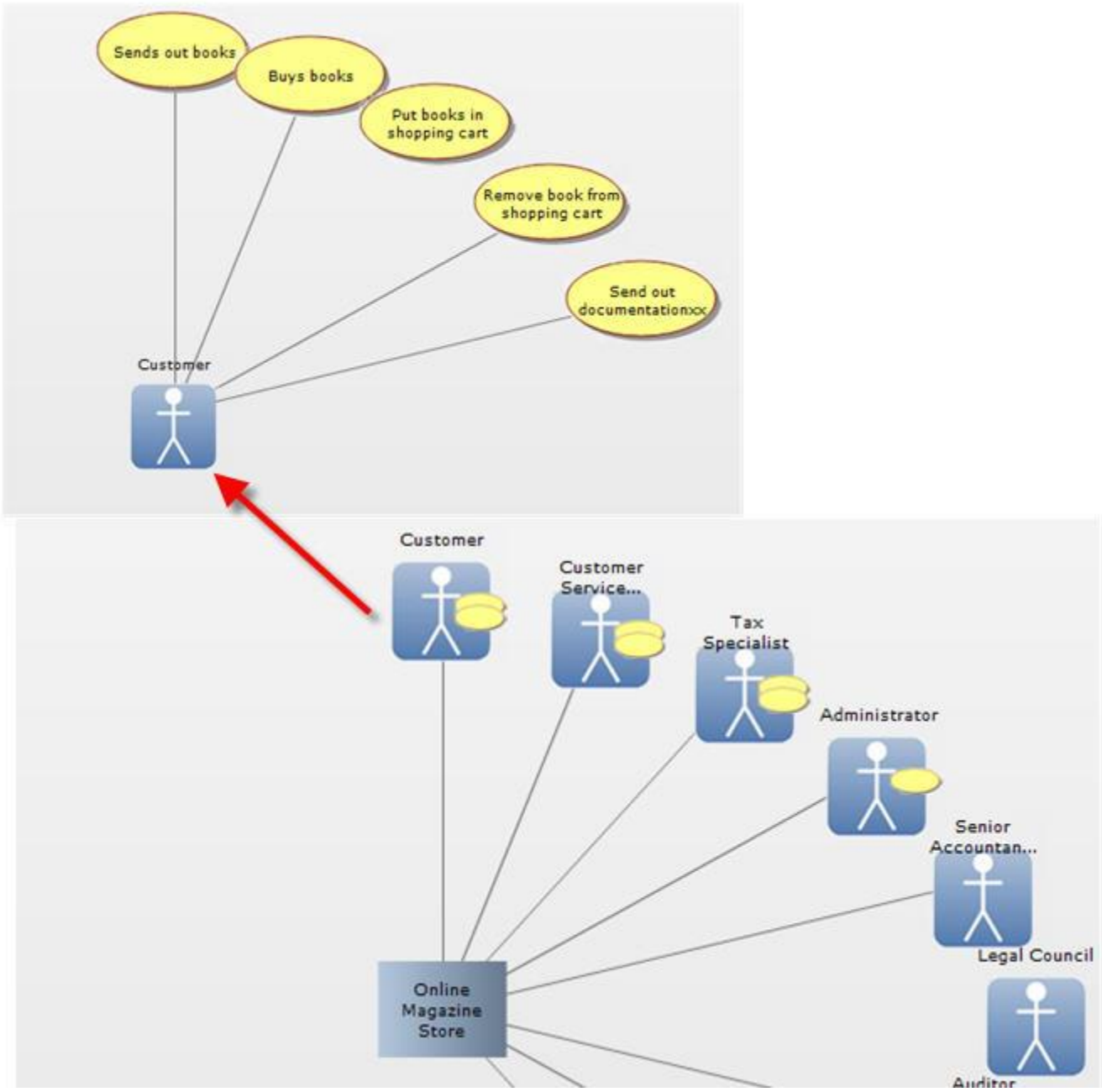
Recently at a new project assignment, I introduced a mid level developer to the concept of use cases which was totally foreign to him. Once walking him through the basic concepts and showing him the use case example, the lightbulb went off in his head on how convenient and simple it was to grasp the project.

A few reasons why it's that much easier to learn a system through use cases then a traditional requirements document is probably because with use cases, you are introduced to concepts at a high level, walk through a living scenario and then presented with specifications last.

In several places in this document, I have stated "effective use cases" rather than just "use cases". The purpose of the use cases is for effective knowledge transfer from the domain expert to the software developer -- these use cases will serve as the software requirement specifications. If they don't make sense to the person building the software, they are not effective. There are several sources on the web for writing effective use cases including the book by Alistair Cockburn.

Tip 11. Sample Use Case Model Diagram

You can use the Gatherspace.com use case modeling tool to produce a sample use case model within a few clicks. Once you define your use cases and actors, just go into the reporting section and click on the 'Use Case Model' report and that's it. See the image below for a sample of the use case model.



In several places in this document, I have stated "effective use cases" rather than just "use cases". The purpose of the use cases is for effective knowledge transfer from the domain expert to the software developer -- these use cases will serve as software requirements. If they don't make sense to the person building the software, they are not effective. There are several sources on the web for writing effective use cases including the book by Alistair Cockburn.

Tip 12. What's the difference between a User Story and a Use Case?

With so many engineering teams making the paradigm shift from waterfall to Agile Software Development, people often get caught up in having a pure Agile process which would include the use of User Stories. So what's all of the hoopla with User Stories? What are they, how are they different from use cases, do I need them, and where do they fit in the process?

- **What is a User Story?** Simply put, written from the context of the user as a simple statement about their feature need. They should generally have this format. "As a -role-, I want -goal/desire- so that -benefit-"
- **How is a User Story different than a Use Case?** While a use case is highly structured and tells a story, the User Story sets the stage by stating the need. A User Story is the prelude to the use case by stating the need before the use case tells the story.
- **How does the User Story fit into the process?** User Stories are great as an activity in collecting and prioritizing the high level features. Getting this initial feedback from the customer is a simple way of trying to get all of their needs identified and prioritized. The User Stories will then morph themselves into the business requirements and use cases.

Tip 13. In Agile Development, Keep Use Cases Agile, Mean and Lean

A common myth with Agile Development is that you *must* use user stories, and not use cases. Like anything else in life, nothing is black and white -- being Agile is really about smaller iterations, learning and adapting to the market.

If you are using Agile, Scrum and moving away from waterfall, what you want to do is make sure to iterate with your use cases. All that means is that your flows will be smaller and less feature rich. While the theme of the use case may appear the same from iteration to iteration, what is changing is the level of detail and the features inside the particular sprint.

Creating a use case to long winded with too many features can potentially put a product at risk. What happens is that you can extend your release to market from two weeks to several months without the ability to learn from the iteration and adapt to the market. Keep those use cases leaner!