

The Project Manager's Guide to Use Cases

By Elizabeth and Richard Larson

We all know how difficult it is to achieve project success without complete product requirements. Yet gathering complete requirements without exhausting the project schedule and budget remains elusive for many project managers. When new technology is added to the mix, the challenges are even greater. This article answers questions frequently asked by project managers about one requirements-gathering technique called use cases, often associated with newer technology.

So what do project managers really need to know about use cases? Here are frequently asked questions and answers.

What is a use case?

A use case is a description of all the ways an end-user wants to "use" a system. These "uses" are like requests of the system, and use cases describe what that system does in response such a request. We can think of use cases as describing the conversation between a system and its user(s). Although the system is usually automated (such as an Order system), use cases also apply to equipment, devices, or business processes.

Let's use the example of a car. One of the ways I'd like to use my car is to have it stop when I want. The use case would describe the "conversation" with my car when I wanted it to stop. That conversation is facilitated through an interface, in this case a brake, which allows me to make my request of the car and for the car to respond to my request to stop. I need to talk my car's language; I cannot simply shout "STOP!" I need to use an interface to state my request. Hopefully the car responds by slowing down and coming to a halt.

. . . And why should project managers care?

Use cases provide a structure for gathering customer requirements and setting the project scope. It is also extremely useful for having the end users "test" the system as it's being designed, which leads to quicker development and a more useable system. While use case modeling does not provide a complete solution to gathering requirements, it does facilitate the development of user interfaces (screens), screen edits and messages, and acceptance test scenarios. Business analysts have traditionally struggled not only with how to translate what the end user wanted the system to do into a technical design, but also with how to have that same end user verify that the translation was correct well before the system was built. The pseudo code typically written by software developers was too technical to be verified by most end users. Use cases help solve this dilemma by providing a translation that end users can understand and react to before too much time has been invested in the project. Bottom line-- more can get done with less.

What are the key components of use cases?

The system. Often described as everything within control of the project, the system is the boundary of the application. For example, the boundaries of a project to develop a new Order system would include the entire application and its system interfaces, and therefore would be considerably broader than if the project were a small enhancement. The system boundary helps set the project scope, as we'll discuss later in the article.

Actors. Actors are external entities that interact directly with the system. Using our Order system example, when customer service representatives enter the item number into the system, they are actors. When customers enter items on a website, they are actors. If a customer orders by phone and a customer service representative enters that order, the customer is not an actor.

Actors can be people, other systems, or time. When the Order system goes to an Inventory system to determine whether or not that item is in stock, the Inventory system is another actor. If month-end processing triggers the Order system to create reports, for example, then time is an actor. Time actors are called temporal actors.

- People (Roles) Actors

When people are being considered as actors, it is important to understand that the people are playing roles (such as customer, maintenance worker, troubleshooting technician, installer, and manager), and they should be documented and described by those role names, not their titles. Sometimes people play more than one role, such as the following:

- Customer and employee
- Technician and manager
- Customer, account representative, and salesperson

It is important to think of actors as role players and not individual persons or job titles. Defining the actors by what interactions they wish to perform with the system will help define and differentiate the actors.

- System Actors

Systems can be any automated device with which the system will interface. This includes other computers and interface devices (if they are out of scope of the system being developed). For example, if the system is a Distribution Center system which reads data from scanned cartons, the devices used to scan the cartons would likely be included as actors. However, if the "system" were the scanning devices themselves, then the larger Distribution Center system would be an actor.

Other Systems can be any automated device which the system will interface.

Additional examples:

- Other computers
- Interface devices (if they are out of scope for the systems being developed)
- anything other than people and time
- Time Actors

Time actors are time-based initiators of processing activity for the system. Common timers are weekly, daily, and monthly.

Additional examples:

- 5:00 PM every Friday a weekly status report gets created
- 12:00 noon every last Tuesday of the month a corporate balance sheet gets created
- Every hour on the hour a time check is performed with the corporate timer system to ensure that the system clock is accurate

Use cases: are the processes that occur within the system. Like other processes, they have a process scope. That is, they begin and end. They also transform input into output. Use cases are named with an active verb and a noun, such as Fulfill Order

Additional examples:

- Process withdrawals
- Balance Accounting & Sales Ledgers
- Back Up Department Case Logs

These are considered processes that accomplish a task for the actor. Use cases should provide value to the actor. The goal is to discover all the possible use cases for all the actors previously defined, which helps define the scope of the system.

What is the difference between a use case model an a use case diagram?

A use case model consists of a use case diagram and narrative text explaining the use cases. The diagram is a picture of the system, actors, and use cases. It contains the system boundary, called a boundary box, the actors, and the use cases. Most diagrams are drawn using Unified Modeling Language (UML, see below).

The narrative text describing the use case is called a flow of events. UML does not address this textual description. Key components of the flow of event include:

- Pre-and post conditions, which set the scope of the use case. A pre-condition states the beginning state, or what has to happen before the use case can begin. The post-condition states the ending condition, or what has occurred as a result of the use case. As with any process, the boundaries of where it begins and ends are usually unclear. For example, where does Fulfill Order end? How do we know we're done fulfilling an order? Is it when the item ships? Is picked in a distribution Center? Is created in the system? Pre- and post-conditions set these boundaries.
- Primary process steps. Sometimes called the primary path, normal course of events, main flow, happy path, sunny day path, or routine process, this is the lists the steps that occur most of the time.
- Alternate steps. Sometimes called the alternate path or sub-flow, these are steps that can lead to the completion of the process, but are less common than the primary process steps.
- Exception steps. Sometimes called the secondary flow of events or exception flow, these are the conditions that could occur to prevent the system from getting to its post-condition(s).

Is the use case the most effective process modeling tool?

While the "system" can be defined as a business process, I don't think it's practical to do so. we have found that other process models, such as process maps, swim lanes, and activity diagrams are more effective. Use cases that model business processes can become text procedures. However, a graphical picture of the business process helps spot anomalies with the current process and ways to improve it. Textual procedures do not lend themselves as well to this type of analysis.

How do I find use cases?

I encourage documenting business processes using one of the models noted above and look for the touch points with automated systems. Those touch points are candidate use cases. Let's use the Order system as an example. Let's say that in the current process the customer service representative enters requested item information into the system. The point where that entry occurs is a candidate use case. The use case would describe what the order taker expects the system to do and what the system does to process the order.

I've heard a lot about UML (Unified Modeling Language) and the Unified Process. How do they relate to use cases?

UML. Introduced in 1997, UML provides notation guidelines for developing diagrams usually associated with object technology. Its main benefit is to help team members better communicate with each other. When diagrams follow the same conventions, business analysts, designers, developers, and testers can all interpret requirements in the same way, decreasing the risk that the requirements as stated by the business expert will be misinterpreted. As noted earlier, there are UML guidelines for use case diagrams, but not for the narrative text. It is not necessary to use UML when creating a use case diagram, but it is helpful.

Unified Process. This set of processes provides a framework for developing systems in general and objects in particular. It provides a development life cycle with standardized project phases, which include Inception, Elaboration, Construction, and Transition. Most of the requirements are defined during elaboration. However, it is not necessary to use the unified process when creating use cases.

How will use cases help me manage my projects?

Scope. The use case diagram is a particularly effective tool that helps identify and manage project scope. Although only one of the many aspects of project management, scope management is often considered the most difficult. The use case diagram helps identify scope in these ways:

- Sets the boundary for the project. The boundary box indicates the scope of the system. Everything within the box is included in the system; anything outside of the box is excluded. This diagram is a wonderful graphical representation of system scope.
- Shows the processes under consideration. Each use case is a process that delivers value to the end user. Each has a business objective or goal that is going to be accomplished.
- The flow of events confirms the scope. It provides the detail involved with each use case process and describes how big each process is.
- All the use cases need to link to the business and project vision and objectives. Any use case that does not provide this alignment, can be easily seen and removed.
- Showing system actors helps provide a picture of the interfaces that need to be modified. This picture aids in showing not only what's contained in the application, but also how many system interfaces need to be included in the project. It provides an effective communications tool and visual that may be helpful in explaining effort that may be transparent to business customers.

The use case diagram helps manage scope in these ways:

- Once use cases have been confirmed, new use cases that arise can be better managed. Changes can be matched to the vision and original diagram to see whether or not they belong. New requests can be translated into new use cases and placed in the diagram.
- If new actors surface, it is an indication that more work is required. Again, the visual serves as a way to communicate with business experts about the impact of their requests on the project.
- If new use cases cause any linkage to change, additional work will be needed as well.

If we create use cases do we still need to complete a requirements list?

Many of our customers have proprietary use case templates which include the associated business rule or requirement. They face the issue of whether or not to create a separate requirements list. Some have chosen to be aligned with IEEE, which does include a separate list in the requirements specification. Others have put the requirements/rules right into or at the end of the use case narrative description and not used a separate requirements list. Still others have done a combination of both.

We encourage a separate requirements list for traceability. The requirements list becomes the foundation for the traceability matrix, which is the premier tool for ensuring that every requirement is fulfilling a business need, and that approved requirements are actually delivered at the end of the project.

Do use cases capture all requirements?

Use cases can be manipulated to capture all requirements, but there are some types of requirements that lend themselves better to other models. One of the most commonly asked questions is why use other models in requirements definition. Here's my response:

- **Use case** models show processes, so if the project is primarily one that manipulates data (such as a data warehouse or reporting application), then other tools are more effective. UML guidelines suggest that the use case not list more than two pieces of data. If there are more, they should be referred to generically in the use case and cross-referenced in a glossary. For example, customer information contains many pieces of data. However, the use case states only 'customer information.' The attributes of customer are modeled in a logical data model or class diagram and listed in a glossary.
- **Process maps.** In addition to the benefits of process maps stated earlier, they also provide user interface (screen) navigation, which ideally follows the business processes (hopefully improved or reengineered). These maps can also drive out hidden business rules, e.g., we cannot delete an item when the on-order quantity is greater than zero.
- **Data models** provide what information appears each screen, for both data entry screens, as well as reports. They also provide many of the business rules, e.g., whether or not customers are required to have accounts.
- **Prototyping**, which is derived from the data, process, and use case models, allows for early feedback and helps drive out additional requirements during requirements analysis. These prototypes do not need to be completely designed or even built with automation. Some of the most effective prototypes are done on paper, using stickies to drive out the alternate and exceptions paths as described in the use cases.

It's no wonder, then, that one of the most common complaints we hear from project managers and business analysts is that they have done a thorough job of use case modeling and yet requirements still surface throughout and after the project!

One proven approach for quick and complete requirements-gathering is what I call "concurrent business modeling." This approach is different from concurrent component engineering, or concurrency, commonly used in developing objects and e-solutions. Concurrent business modeling focuses on obtaining business requirements, not on building software. It suggests that by modeling business data, business processes, system processes through use case modeling, and prototyping, requirements quickly surface. In addition, each type of modeling effort supports the other modeling efforts and encourages analysts to ask their customers the kinds of pertinent questions that drive out hidden requirements.

© 2004 Watermark Learning, Inc.