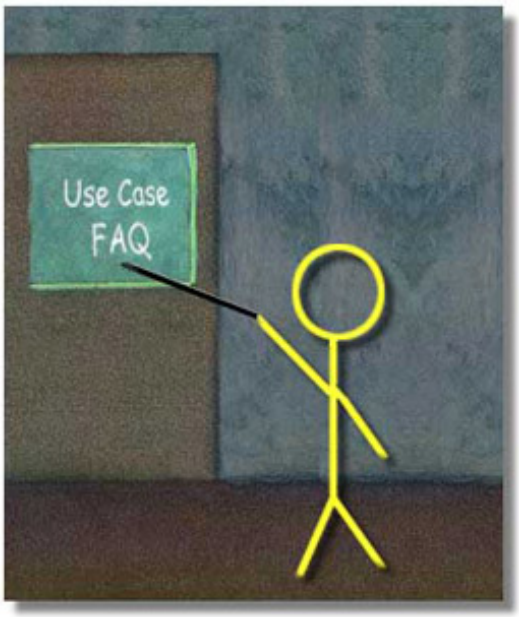
Use Cases: Just the FAQs (and Answers)

by **Kurt Bittner**
Director, Requirements Management Solutions
Rational Software



People learning about use cases have some fairly standard questions. Below, we provide short answers to get you started on common questions from "What is a use case?" to "How do you identify a use case at the right level?" For more detailed and comprehensive information on use cases, refer to the book ***Use Case Modeling****, which I co-wrote with Ian Spence, also of Rational.*[1]

The following questions were posed at a Birds of a Feather session on "Writing Effective Use Cases," held at the 2002 Rational User Conference in Orlando, Florida. During the session, which attracted more than 150 attendees, we answered some of the more common questions, and followed up later with the written summary presented below.

We view these responses as a way to "prime the pump" for an ongoing use-case discussion thread we plan to start soon on Rational Developer Network (RDN), a rich resource for information, downloads, and training for Rational customers (registration required). In the meantime, if you have any questions, please send them to kbittner@rational.com, and we'll update this FAQ as a living document.

For those who want more detailed information now, note that I refer throughout this article to examples in ***Use Case Modeling****. There is a also glossary of common terms in the sidebar.*

The FAQs

Q: What is a good use case?

A: A good use case provides something of significant value to at least one of its actors. By "significant," we mean something that satisfies a goal of one of the actors -- something that, if it were the only thing the system did, would still provide value to at least one of its stakeholders. Note that this applies only to concrete use cases, not to those that extend other use cases (extension use cases) or that are included by other use cases (included use cases).

A good use case is also independent -- it does not depend on any other use case in order to provide value. If a use case *must* be combined with other use cases to provide value to its actors, then it should be reconsidered. For example, for a typical Automated Teller Machine (ATM) application, "Enter PIN" would not be a good use case, because this action would never be performed on its own.

You'll find quite a lot of information on indentifying good use cases in Chapter 4 of *Use Case Modeling*: "Finding Actors and Use Cases."

Q: Where do you put GUI information?

A: GUI information goes into a companion artifact to the use case called a use-case storyboard. The storyboard depicts how the use case is realized and performed by the system's user interface. At its simplest, a use-case storyboard consists of a series of screen mock-ups, arranged in sequence, that depict one or more scenarios of the use case. If the user interface is particularly complex, consisting of a number of screens, dialog boxes, menus, fields, panes, and so on, the use-case storyboard may be expanded to include a more formal model of the user interface and the composition and interaction of user interface components.

The Rational Unified Processý has excellent descriptions of use-case storyboards and designing for usability.

Q: How do you know when you should write one or more use cases, instead of a basic flow or alternative flow?

A Basic Use-Case Glossary

Alternative Flow
A use-case flow that describes some alternative or exceptional behavior -- something that causes the system to deviate from the basic flow of events.

Basic Flow
The use-case flow that describes the most likely course of events the system will take when the use case is performed.

Extension Use Case
A "fragment" use case that adds behavior to at least one other use case (referred to as the "base" or "extended" use case). Extension is used to simplify the addition of behavior to an already complete use-case description.

Flow
A section of a use-case description that describes some full or partial path through the use case. A use case always has at least a basic flow and may have one or more alternative flows.

A: The use case should represent the complete set of flows that are associated with a particular "thing of value" that the system does for one or more of its actors. As a result, if a flow is related to producing the thing of value (the result produced by the use case) it should be grouped with other related flows. If the flows are related, don't break them into separate use cases unless:

- The flow occurs in more than one use case (this would result in ***inclusion***).
- The flow could be optional for some instances of the system (this would result in ***extension***).

Information about writing use case flows-of-events is covered in Chapters 8, 9, and 10 of *Use Case Modeling*.

Included Use Case
A "fragment" of a use case that is referred to by two or more other use cases (referred to as the "base" or "including" use cases). Included use cases are used to eliminate redundant description in two or more use cases.

Scenario
A scenario is a basic flow of events plus zero or more alternative flows.

Use Case
A use case is a description of how an actor or actors use a system to achieve a goal, including what the system does for the actor or actors to achieve that goal.

Use-Case Storyboard
A visual use-case description that depicts the flow of events with a sequenced series of screen shots, often supplemented or annotated with the actual use-case description. Used to visualize the behavior of the system or to gather feedback on a highly visual use-case description.

Q: Should you identify different use cases, one for each phase of the development effort, to facilitate iterative development?

A: Don't break the use case up to account for implementing different flows in different phases. Instead, use the concept of the scenario, *which consists of the basic flow of events plus zero or more alternative flows.* Scenarios also often have accompanying data sets -- values for the information that is exchanged in the use case. Identifying scenarios enables the use case to be "partitioned" across iterations; each iteration will address one or more scenarios for one or more use cases. And identifying scenarios has another benefit: It assists with creating test cases.

Q: How much detail do you put into a use-case description? When is enough, enough?

A: The use-case description needs to be sufficiently detailed so that all stakeholders understand what the system will do to provide the value required of it. This means that users, customers, developers, and testers all need to agree not only on what the system will do, but also on how they will measure whether it has done the things it needs to do. "Enough is enough" when everyone reaches a consensus that the description is detailed enough.

The extent of the description will vary, based on a number of factors:

- **The size and cohesiveness of the extended team.** Larger and less cohesive teams will require more detailed descriptions.
- **The familiarity of the extended team with the problem domain:** A team that is unfamiliar with the problem domain will require more detailed descriptions.
- **The complexity of the problem domain.** A more complex problem domain will require more detailed descriptions.

As we noted above, Chapters 8 and 9 of *Use Case Modeling* cover writing use-case descriptions in more detail.

Q: How can one capture requirements from an existing system as use cases if the original requirements document is either long gone or completely out of sync with the current system?

A: The first step is to understand what things of value the system does for its stakeholders; these things become the use cases. Do this by observing how people use the system and interviewing them to understand what things they achieve by using it. The original system may have been intended to do many more things, but if it is not being *used* for those things, then there is no need to document them as use cases.

When you observe how people use the system, be sure to record what they do and how the system responds. Keep your description *externally focused*; don't try to describe what happens inside the system. It should be detailed enough, however, to identify the basic flow and alternative flows.

The reason to document the use cases in this manner is that usually, some enhancements need to be made to the system. Armed with those *external* use-case descriptions, you should be able to identify scenarios, determine which scenarios will change with the new enhancements, and allocate these scenarios to iterations. In the iteration itself, you will detail the internal behavior of the system exercised by the use case, then plan and implement the required changes. You need not detail scenarios that will be unaffected by the planned changes -- that is, the alternative flows that are not involved in the planned changes may remain described at a *superficial* level.

Q: How do you avoid functional decomposition?

A: Principally, you avoid functional decomposition by:

- **Focusing on identifying concrete use cases** that provide real, independent value for at least one system stakeholder.
- **Limiting the use of inclusions** to representing descriptions shared by more than one use case.
- **Limiting extension**

to adding optional behaviors to an existing use case that, by itself, provides value.

Techniques for identifying at the right level are covered on pages 104-110 of *Use Case Modeling*. Information about using the *include* and *extend* associations properly is covered in Chapter 10, "Here There Be Dragons."

Q: How do you distribute a use-case model to stakeholders?

A: The best way to do this is to walk through the model with the stakeholders, explaining all the elements of the model and the behavior they represent. Distributing documents for review is rarely productive, because people often do not take the time to them. If you walk them through first the use-case model survey, and then each use-case description (usually in separate meetings), you will improve communication and confirm that everyone has the same understanding of the behavior the system will provide.

Separate the review of the use-case model survey (actors and use cases, shown in diagrams, supplemented by brief descriptions of the actors and use cases), from reviews of the use-case descriptions. **Don't move on to writing use-case descriptions until you have agreement on the use-case model survey.**

Use Case Modeling has useful information about reviewing use cases in Chapter 11, "Reviewing Use Cases." See especially pages 280-283.

Q: What is the difference between a requirement and a business rule?

A: Business rules are a kind of requirement. There is no precise definition for a business rule -- it may mean different things to different people. In some cases, a business rule refers to a sequence of events the system must provide; these are expressed as steps in a use-case description. In other cases, business rules refer to validation rules about key business entities (e.g., all orders must consist of one or more items; a customer may have one or more open orders; etc.), and should be captured in the domain model. In still other cases, business rules refer to nonfunctional attributes the system must satisfy (e.g., support more than 1,000 concurrent users; be serviceable without bringing the system down; etc.), and should be captured as special requirements that are traced to the use cases that must satisfy them.

Q: How do you identify a use case at the right level?

A: This is closely related to the previous questions about avoiding functional decomposition and determining whether a use case is a good one. The way to make sure that use cases are at the "right level" is to focus on the value they provide to stakeholders. A use case is a complete sequence of events that does something of value for at least one of its actors without having to be combined with some other use case. If the use case doesn't do something useful, then it is not at the right level.

More Questions?

As you discover more questions, please send them to me them at kbittner@rational.com. We'll use them in the upcoming discussion thread planned for RDN.

Notes

[1] Kurt Bittner and Ian Spence, *Use Case Modeling*. Addison Wesley, 2002.

For more information on the products or services discussed in this article, please click here and follow the instructions provided. Thank you!