Revision / Change History

|  |  |  |
| --- | --- | --- |
| **Revision Level** | **Revision Date** | **Description of Changes** |
| 01 | 5/19/2011 | * Initial Version |

A use case is a description of a system’s behavior as it responds to a request that originates from outside of that system. In other words, a use case describes "who" can do "what" with the system in question. The specific system in question is not important (the use case should be system agnostic). The use case technique is used to capture a system's behavioral requirements by detailing scenario-driven threads through the functional requirements and focuses on describing how to achieve a goal or a task.

Use cases should not be confused with the features of the system. One or more features (a.k.a. "system requirements") describe the functionality needed to meet a stakeholder request or user need (a.k.a. "user requirement"). Each feature can be analyzed into one or more use cases, which detail cases where an actor uses the system. Each use case should be traceable to its originating feature, which in turn should be traceable to its originating stakeholder/user request.

Use cases treat the system as a black box, and the interactions with the system, including system responses, are perceived as from outside the system. This is a deliberate policy, because it forces the author to focus on what the system must do, not how it is to be done, and avoids making assumptions about how the functionality will be accomplished.

A use case should:

* Describe what the system shall do for the actor to achieve a particular goal.
* Include no implementation-specific language.
* Be at the appropriate level of detail.
* Not include detail regarding user interfaces and screens. This is done in user-interface design, which references the use case and its business rules.

A **business use case** is described in technology-free terminology which treats system as a black box and describes the business process that is used by its business actors (people or systems external to the process) to achieve their goals or complete transactions (e.g., manual payment processing, expense report approval, manage corporate real estate). The business use case will describe a process that provides value to the business actor, and it describes *what* the process does.

A **system use case** describes a system that automates a business use case or process. It is normally described at the system functionality level (for example, "create voucher") and specifies the function or the service that the system provides for the actor. The system use case details *what* the system will do in response to an actor's actions. For this reason it is recommended that system use case specification begin with a verb (e.g., *create* voucher, *select* payments, *exclude* payment, *cancel* voucher). An actor can be a human user or another system/subsystem interacting with the system being defined.

The following should be included in the Use Case:

**Use Case Name**

A descriptive name provides a unique identifier for the use case. It should be written in verb-noun format (e.g., *Borrow Books*, *Withdraw Cash*), should describe an achievable goal (e.g., *Register User* is better than *Registering User*) and should be sufficient for the end user to understand what the use case is about.

Goal-driven use case analysis will name use cases according to the actor's goals, thus ensuring use cases are strongly user centric. Two to three words is the optimum. If more than four words are proposed for a name, there is usually a shorter and more specific name that could be used.

**Use Case ID**

When the number of use cases grows beyond a few dozen, the engineering practice of using an immutable number along with the descriptive name will assist in managing identities. Naming conventions such as a prefix (eg: "UC") added to a number (eg: "UC-134") make it easily recognizable as a use case. The effect of using immutable numbers is to isolate external references to the use case when names (referenced perhaps hundreds of times in dozens of documents) are changed, and to ensure distinct names across many similar use cases (eg: an enterprise with "Inform Customer" in many different contexts). Typically the name and number (eg: "UC-134 Borrow Books") occur only in the use case itself, only the number ("UC-134") is used in external references, and a master list of names and numbers is maintained at the enterprise level.

**Version**

Often a version section is needed to inform the reader of the stage a use case has reached. The initial use case developed for business analysis and scoping may well be very different from the evolved version of that use case when the software is being developed. Older versions of the use case may still be in current documents, because they may be valuable to different user groups.

**Goal**

Without a goal a use case is useless. There is no need for a use case when there is no need for any actor to achieve a goal. A goal briefly describes what the user intends to achieve with this use case.

**Summary**

A summary section is used to capture the essence of a use case before the main body is complete. It provides a quick overview, which is intended to save the reader from having to read the full contents of a use case to understand what the use case is about. Ideally, a summary is just a few sentences or a paragraph in length and includes the goal and principal actor.

**Actor(s)**

An actor is someone or something outside the system that either acts on the system – a primary actor – or is acted on by the system – a secondary actor. An actor may be a person, a device, another system or sub-system, or time. Actors represent the different roles that something outside has in its relationship with the system whose functional requirements are being specified. An individual in the real world can be represented by several actors if they have several different roles and goals in regards to a system. These interact with system and do some action on that.

**Stakeholders**

A stakeholder is an individual or department that is affected by the outcome of the use case. Individuals are usually agents of the organization or department for which the use case is being created. A stakeholder might be called on to provide input, feedback, or authorization for the use case. The stakeholder section of the use case can include a brief description of which of these functions the stakeholder is assigned to fulfill.

**Preconditions**

A *preconditions* section defines all the conditions that must be true (i.e., describes the state of the system) for the *trigger* (see below) to meaningfully cause the initiation of the use case. That is, if the system is not in the state described in the preconditions, the behavior of the use case is indeterminate. Note that the preconditions are *not* the same thing as the "trigger" (see below): the mere fact that the preconditions are met does NOT initiate the use case.

However, it is theoretically possible *both* that a use case should be initiated whenever condition X is met *and* that condition X is the only aspect of the system that defines whether the use case can meaningfully start. If this is really true, then condition X is *both* the precondition and the trigger, and would appear in both sections. But this is *rare*, and the analyst should check carefully that they have not overlooked some preconditions which are part of the trigger. If the analyst has erred, the module based on this use case will be triggered when the system is in a state the developer has not planned for, and the module may fail or behave unpredictably.

**Triggers**

A 'triggers' section describes the event that causes the use case to be initiated. This event can be external, temporal or internal. If the trigger is not a simple true "event" (e.g., the customer presses a button), but instead "when a set of conditions are met", there will need to be a triggering process that continually (or periodically) runs to test whether the "trigger conditions" are met: the "triggering event" is a signal from the trigger process that the conditions are now met.

There is varying practice over how to describe what to do when the trigger occurs but the preconditions are not met.

* One way is to handle the "error" within the use case (as an exception). Strictly, this is illogical, because the "preconditions" are now not true preconditions at all (because the behavior of the use case is determined even when the preconditions are not met).
* Another way is to put all the preconditions in the trigger (so that the use case does not run if the preconditions are not met) and create a different use case to handle the problem. Note that if this is the local standard, then the use case template theoretically does not need a preconditions section!

**Basic Course of Events**

At a minimum, each use case should convey a *primary scenario*, or typical course of events, also called "basic flow", "normal flow," "happy flow" and "happy path". The main basic course of events is often conveyed as a set of usually numbered steps. For example:

1. The system prompts the user to log on,

2. The user enters his name and password,

3. The system verifies the logon information,

4. The system logs user on to system.

**Alternative Paths or Extensions**

Use cases may contain secondary paths or alternative scenarios, which are variations on the main theme. Each tested rule may lead to an alternative path and when there are many rules the permutation of paths increases rapidly, which can lead to very complex documents. Sometimes it is better to use conditional logic or activity diagrams to describe use case with many rules and conditions.

Exceptions, or what happens when things go wrong at the system level, may also be described, not using the alternative paths section but in a section of their own. Alternative paths make use of the numbering of the basic course of events to show at which point they differ from the basic scenario, and, if appropriate, where they rejoin. The intention is to avoid repeating information unnecessarily. The description of an exception should indicate how the system will respond to, or (if possible) recover from, the error condition.

An example of an alternative path would be: "The system recognizes a cookie on the user's machine", and "Go to step 4 (Main path)". An example of an exception path would be: "The system does not recognize a user's logon information", and "Go to step 1 (Main path)".

(NOTE) Many use case designers prefer to put the complete series of steps in an Alternate Flow rather than referring back (rejoin) to steps in the Primary or Happy Path Flow. The reason for this preference is that Test Engineers may be receiving segments of a use case in order to design test cases. As such, they may not have the full picture of the sequences. Another reason of making the Alternate Flow stand on its own is that of reuse. In the area of error control and reporting, an Alternate Path may be identical in all aspects except for the error message. Being able to reuse the Alternate Flow can save significant design time. And finally, it is much easier to read and follow an Alternate Path when the steps are present rather than having to jump between the Primary Path and the Alternate Path. Although it is a preference to make the use case as readable as possible, there are few set rules on how to define the paths.

**Postconditions**

The *post-conditions* section describes what the change in state of the system will be after the use case completes. Post-conditions are guaranteed to be true when the use case ends.

**Business rules**

Business rules are written (or unwritten) rules or policies that determine how an organization conducts its business with regard to a use case. Business rules are a special kind of requirement. Business rules may be apply to a specific use case (local business rule) or to many use cases across an enterprise (common business rule). Use cases should clearly reference business rules that are applicable and where they are implemented. Naming/numbering conventions can be used for this purpose, such as using the prefix "BR" (eg: BR01, BR02) for local business rules and the prefix "CBR" for common business rules (eg: CBR354, CBR9210).

Business Rules should be encoded in-line with the Use Case logic and execution may lead to different post conditions. E.g. Rule2. that a cash withdraw will lead to an update of the account and a transaction log leads to a post condition on successful withdrawal - but only if Rule1 which says there must be sufficient funds tests as true.

**Notes**

Experience has shown that however well-designed a use case template is, the analyst will have some important information that does not fit under a specific heading. Therefore all good templates include a section (e.g. "Notes to Developers") that allows less-structured information to be recorded.

**Author and Date**

This section should list when a version of the use case was created and who documented it. It should also list and date any versions of the use case from an earlier stage in the development which are still current documents. The author is traditionally listed at the bottom, because it is not considered to be essential information; use cases are intended to be collaborative endeavors and they should be jointly owned.