

### Use case name

A use case 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.

### 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 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.

### Actors

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.

### 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

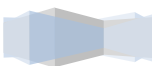
At a minimum, each use case should convey a *primary scenario*, or typical course of events, also called "basic flow", "happy flow" and "happy path". The main basic course of events is often conveyed as a set of usually numbered steps. For example: The system prompts the user to log on, the user enters his name and password, the system verifies the logon information, and the system logs user on to system.

### Alternative paths

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.

An example of an alternative path would be: "The system recognizes cookie on user's machine", and "Go to step 4 (Main path)". An example of an exception path would be: "The system does not recognize user's logon information", and "Go to step 1 (Main path)". According to Anthony J H Simons and Ian Graham (who openly admits he got it wrong - using 2000 use cases at Swiss Bank), alternative paths were not originally part of use cases. Instead, each use case represented a single user's interaction with the system. In other words, each use case represented one possible path through the system. Multiple use cases would be needed before designs based on them could be made. In this sense,



use cases are for exploration, not documentation. An [Activity diagram](#) can give an overview of the basic path and alternative path.

## Post conditions

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 specific to a use case or apply across all the use cases, or across the entire business. Use cases should clearly reference BRs that are applicable and where they are implemented.

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

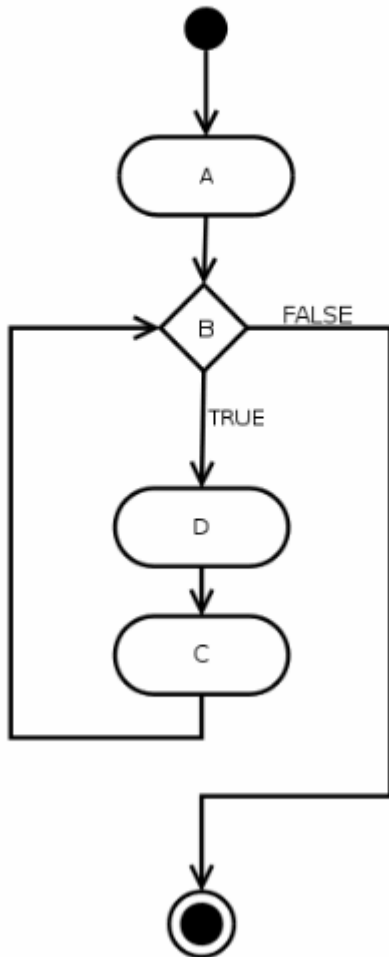
Experience has shown that however well-designed a use case template is, the analyst will have some important information that does not fit under a specific heading. Therefore all good templates include a section (eg "Notes to Developers") that allows less-structured information to be recorded.

## 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.

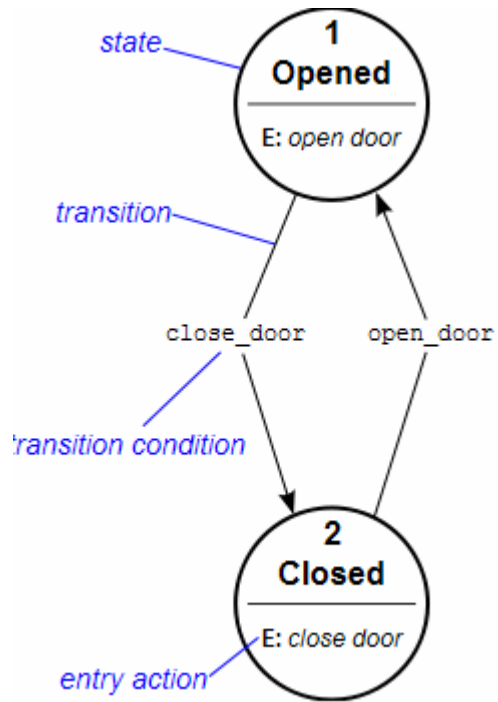


for(A;B;C)  
D;



Activity Diagram for Use Case





UML Diagram

