Cheatography

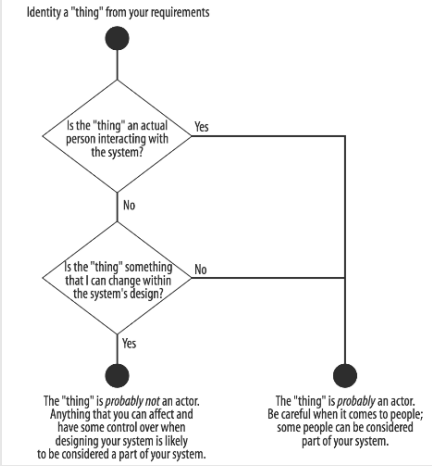
TTM4115 UML Cheat Sheet

by Ida via cheatography.com/27527/cs/8042/

Use cases

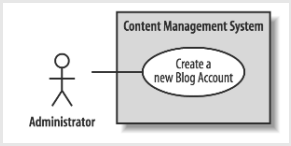
Use cases specify only what your system is supposed to do, i.e., the system's functional requirements. They *do not* specify what the system shall not do, i.e., the system's nonfunctional requirements.
Use cases are a means to bring those gaps in the user's requirements to the forefront at the beginning of a project.

How to identify an actor



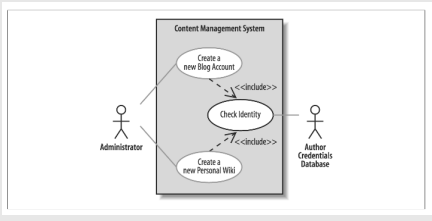
Actors is external to your system
Use cases is internal to your system

System Boundaries



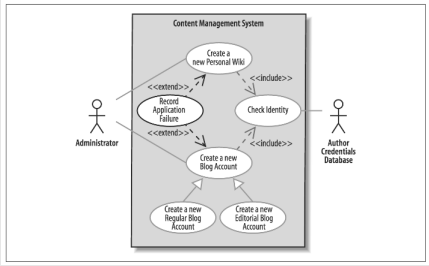
To show your system's boundary on a use case diagram, draw a box around all of the use cases but keep the actors outside of the box

The Include Relationship



The include relationship declares that the use case at the tail of the dotted arrow *completely* reuses all of the steps from the use case being included.

The extend Relationship



A use case *might* completely reuse another use case's behavior, similar to the include relationshop, but that this reuse is *optional* and dependent either on a runtime or system implementation decision.

Capturing a System Requirement

Look at the *things* that interact with your system. In use cases these external things are called *actors*.
A *shall requirement* must be fulfilled. A *should requirement* implies that the requirement is not critical to the system working but is still desirable.

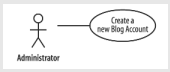
Use Cases

From a user's perspective, a use case is a complete use of the system; there is some interaction with the system, as well as some output from that interaction .

What makes a good use case?

A use case is something that provides some measurable result to the user or an external system.

Communication Lines



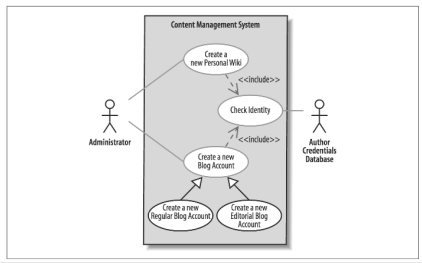
A *communication line* connects an actor and a use case to show the actor participating in the use case.

Use Case Descriptions

Use case description detail	What the detail means and why it is useful
Related Requirements	Some indication as to which requirements this use case partially or completely fulfills.
Goal In Context	The use case's place within the system and why this use case is important.
Preconditions	What needs to happen before the use case can be executed.
Successful End Condition	What the system's condition should be if the use case executes successfully.
Failed End Condition	What the system's condition should be if the use case fails to execute successfully.
Primary Actors	The main actors that participate in the use case. Often includes the actors that trigger or directly receive information from a use case's execution.
Secondary Actors	Actors that participate but are not the main players in a use case's execution.
Trigger	The event triggered by an actor that causes the use case to execute.
Main Flow	The place to describe each of the important steps in a use case's normal execution.
Extensions	A description of any alternative steps from the ones described in the Main Flow.

A use case's description completes the use caes; without a description a use case is not very useful.

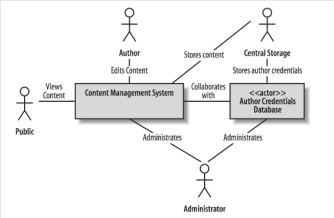
Use case inheritance



Use case inheritance is useful when you want to show that one use case is a special type of another use case. To show use case inheritance, use the generalization arrow to connect the more general, or *parent*, use case to the more specific one.

Be careful using inheritance - you are then saying that *every* step in the general use case *must* occur in the specialized use cases.

Use Case Overview Diagrams



Use Case Overview diagram give you an opportunity to paint a broad picture of your system's context or domain.

Use Case Overviews are a useful place to show any extra snippets of information. Those snippets often include relationships and communication lines between actors