
## UML TUTORIALS

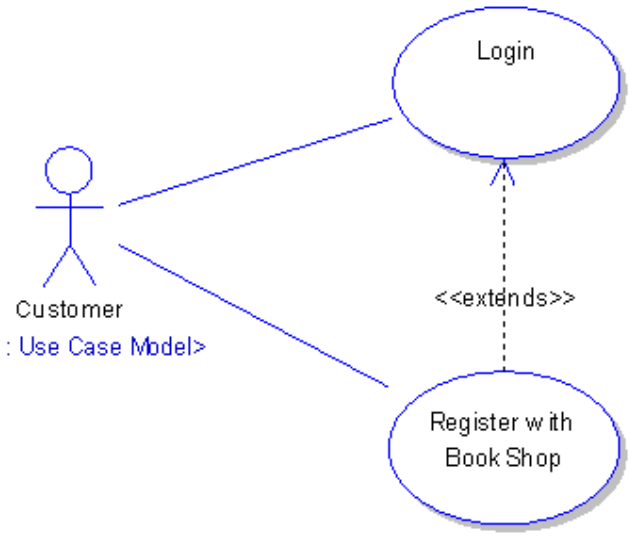
# THE USE CASE MODEL


www.sparxsystems.com.au

# The Use Case Model

The Use Case Model describes the proposed functionality of the new system. A Use Case represents a discrete unit of interaction between a user (human or machine) and the system. A Use Case is a single unit of meaningful work; for example login to system, register with system and create order are all Use Cases. Each Use Case has a description which describes the functionality that will be built in the proposed system. A Use Case may 'include' another Use Case's functionality or 'extend' another Use Case with its own behaviour.

Use Cases are typically related to 'actors'. An actor is a human or machine entity that interacts with the system to perform meaningful work.




A Use Case description will generally include:

1. General comments and notes describing the use case;
2. Requirements - Things that the use case must allow the user to do, such as <ability to update order>, <ability to modify order> & etc.
3. Constraints- Rules about what can and can't be done. Includes i) pre-conditions that must be true before the use case is run -e.g. <create order> must precede <modify order>; ii) post-conditions that must be true once the use case is run e.g. <order is modified and consistent>; iii) invariants: these are always true - e.g. an order must always have a customer number
4. Scenarios - Sequential descriptions of the steps taken to carry out the use case. May include multiple scenarios, to cater for exceptional circumstances and alternate processing paths;
5. Scenario diagrams -Sequence diagrams to depict the workflow - similar to (4) but graphically portrayed.
6. Additional attributes such as implementation phase, version number, complexity rating, stereotype and status

**Actors**
An Actor is a user of the system. This includes both human users and other computer systems. An Actor uses a Use Case to perform some piece of work which is of value to the business. The set of Use Cases an actor has access to defines their overall role in the system and the scope of their action.

### Constraints, Requirements and Scenarios

The formal specification of a Use Case includes:

1. Requirements. These are the formal functional requirements that a Use Case must provide to the end user. They correspond to the functional specifications found in structured methodologies. A requirement is a contract that the Use Case will perform some action or provide some value to the system.
2. Constraints. These are the formal rules and limitations that a Use Case operates under, and includes pre- post- and invariant conditions. A pre-condition specifies what must have already occurred or be in place before the Use Case may start. A post-condition documents what will be true once the Use Case is complete. An invariant specifies what will be true throughout the time the Use Case operates.
3. Scenarios. Scenarios are formal descriptions of the flow of events that occurs during a Use Case instance. These are usually described in text and correspond to a textual representation of the Sequence Diagram.

### Includes and Extends relationships between Use Cases

One Use Case may include the functionality of another as part of its normal processing. Generally, it is assumed that the included Use Case will be called every time the basic path is run. An example may be to list a set of customer orders to choose from before modifying a selected order - in this case the <list orders> Use Case may be included every time the <modify order> Use Case is run.

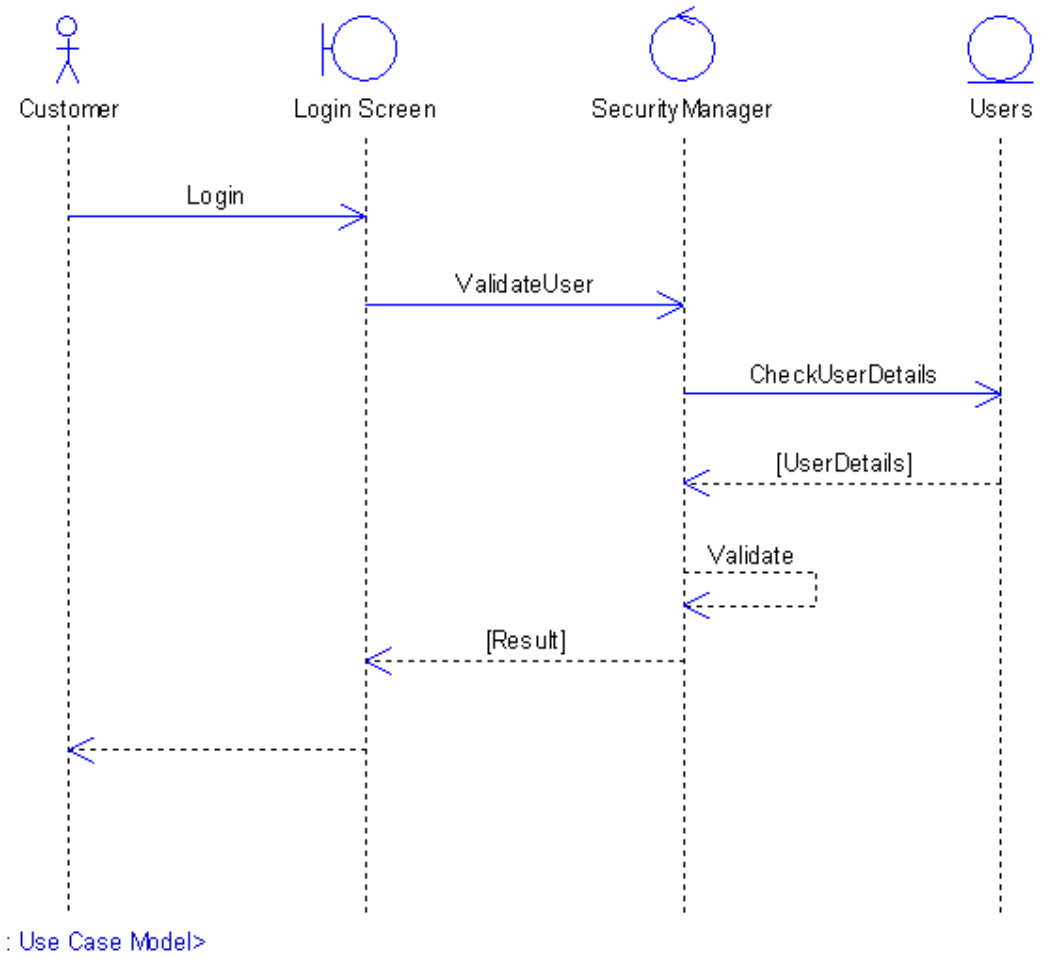
A Use Case may be included by one or more Use Cases, so it helps to reduce duplication of functionality by factoring out common behaviour into Use Cases that are re-used many times.
One Use Case may extend the behaviour of another - typically when exceptional circumstances are encountered. For example, if before modifying a particular type of customer order, a user must get approval from some higher authority, then the <get approval> Use Case may optionally extend the regular <modify order> Use Case.

**Sequence Diagrams**

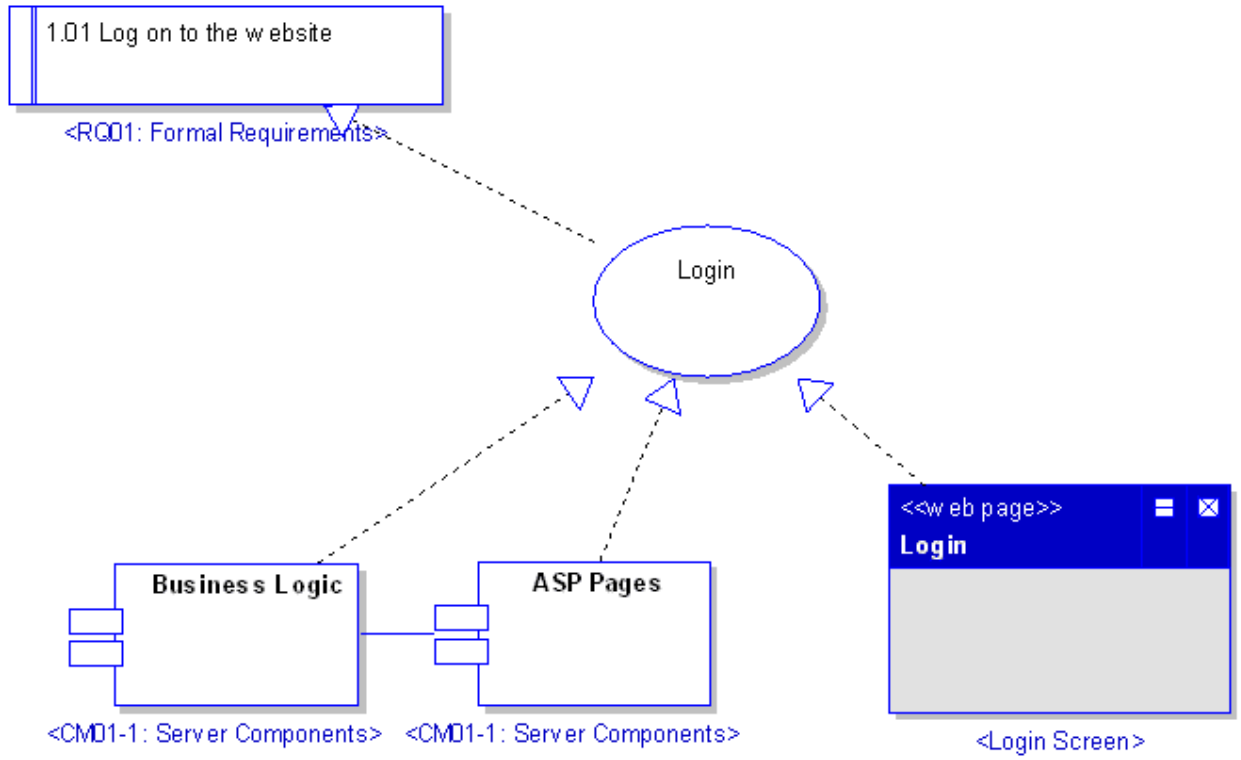
UML provides a graphical means of depicting object interactions over time in Sequence Diagrams. These typically show a user or actor, and the objects and components they interact with in the execution of a use case. One sequence diagram typically represents a single Use Case 'scenario' or flow of events.

Sequence diagrams are an excellent way to document usage scenarios and to both capture required objects early in analysis and to verify object usage later in design. Sequence diagrams show the flow of messages from one object to another, and as such correspond to the methods and events supported by a class/object.

The diagram illustrated below shows an example of a sequence diagram, with the user or actor on the left initiating a flow of events and messages that correspond to the Use Case scenario. The messages that pass between objects will become class operations in the final model.



**Implementation Diagram**
A Use Case is a formal description of functionality the system will have when constructed. An implementation diagram is typically associated with a Use Case to document what design elements (eg. components and classes) will implement the Use Case functionality in the new system. This provides a high level of traceability for the system designer, the customer and the team that will actually build the system. The list of Use Cases that a component or class is linked to documents the minimum functionality that must be implemented by the component.




The example above shows that the Use Case "Login" implements the formal requirement "1.01 Log on to the website". It also states that the Business Logic component and ASP Pages component implement some or all of the Login functionality. A further refinement is to show the Login screen (a web page) as implementing the Login interface. These implementation or realisation links define the traceability from the formal requirements, through Use Cases on to Components and Screens.