chapitre 3 uses case diagram

Dr Soumia LAYACHI

October 2025

1 Introduction

Use cases are used to express the system's behavior in terms of actions and reactions, *from the perspective of each user* . Use cases delineate the system, its functions (its cases), and its relationships with its environment. They are a means of *determining the system's needs* . Use cases are developed based on interviews with users.

2 How to identify use cases ?

Each use case therefore corresponds to a business function of the system, according to the point of view of one of its actors. Also, to identify the use cases, you must place yourself from the point of view of each actor and determine how and especially why they use the system. Name the use cases with an infinitive verb followed by a complement by placing yourself from the point of view of the actor and not that of the system. For example, an ATM will probably have a use case Withdraw money and not Distribute money.

In any case, it is important to keep in mind that there is no notion of time in a use case diagram.

3 Textual description of use cases

The use case diagram describes the broad functions of a system from the perspective of the actors, but does not detail the dialogue between the actors and the use cases. It is recommended to write a textual description, as it is a flexible form that is suitable in many situations.

A commonly used textual description consists of three parts.

3.1 The first part allows you to identify

the case and must contain the following information.
Name :
— use an infinitive verb followed by a complement.

— Objective :
— a summary description of the use case.
— Main actors : those who will carry out the use case
— Secondary actors : those who only receive information after the use case has been completed.
— Dates :
— the creation and update dates of the current description.
— Responsible :
— the names of those responsible.
— Version :
— number .

3.2

The second always contains a nominal sequence that describes the normal course of the case. In addition to the nominal sequence, there are often alternative sequences (branches in the nominal sequence) and exception sequences (which occur when an error occurs).

Preconditions :

— They describe what state the system (application) must be in before this use case can be triggered.
— Scenario :
— These scenarios are described in the form of event exchanges between the actor and the system. We distinguish the nominal scenario, which takes place when there is no error, alternative scenarios which are variants of the nominal scenario and finally exception scenarios which describe error cases.

Pos tconditions :

— they describe the state of the system at the end of the different scenarios.

3.3

The third part of the use case description is an optional section. It generally contains non-functional specifications (technical specifications, etc.). It may optionally contain a description of the graphical interface requirements .

4 Approach to using use cases

1. Identify the actors and the cases.
2. Describe the cases by writing scenarios in text form.
3. Draw the case diagrams.
4. Summarize the cases, structure them and ensure overall consistency

5 use case diagrams

5.1 Actor

An actor is a person or system that interacts with the system under study, exchanging information (input and output). Actors are found by observing the direct users of the system, those responsible for its maintenance, as well as other systems that interact with it. An actor represents a role played by a user who interacts with the system. The same physical person can play the role of several actors. On the other hand, several people can play the same role, and therefore act as the same actor.

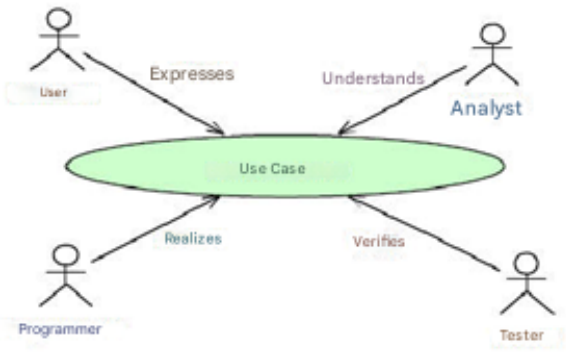
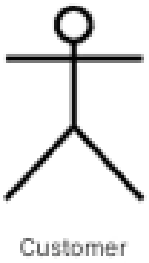


FIGURE 1 – Use cases, the common thread of the project.

— An actor is represented by a small man with his name (his role) written underneath. `<<actor>>`.
— It is also possible to represent an actor in the form of a stereotypical binder `<<actor>>`.



(a) Example of representation of an actor.

(b) Example of representation of an actor in the form of a workbook.

FIGURE 2 – Other representations for Actor

5.2 Use cases

Uses cases represent the dialogue between the actor and the system in an abstract way. A use case is a coherent unit representing an externally visible functionality. It realizes an end-to-end service, with a trigger, a flow, and an end, for the actor who initiates it.

A use case is represented by an ellipsis containing the name of the case (a verb in the infinitive), and optionally, above the name, a stereotype.

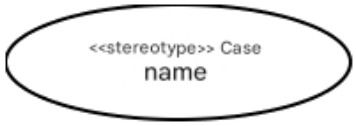


FIGURE 3 – : Example of a use case representation.

If you want to represent the attributes or operations of the use case, it is best to represent it in the form of a stereotyped "use case" workbook. 4

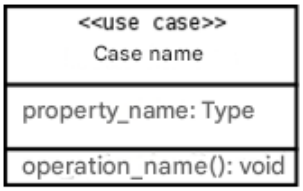


FIGURE 4 – : Example of a use case representation in the form of a workbook.

6 Representation of a use case diagram

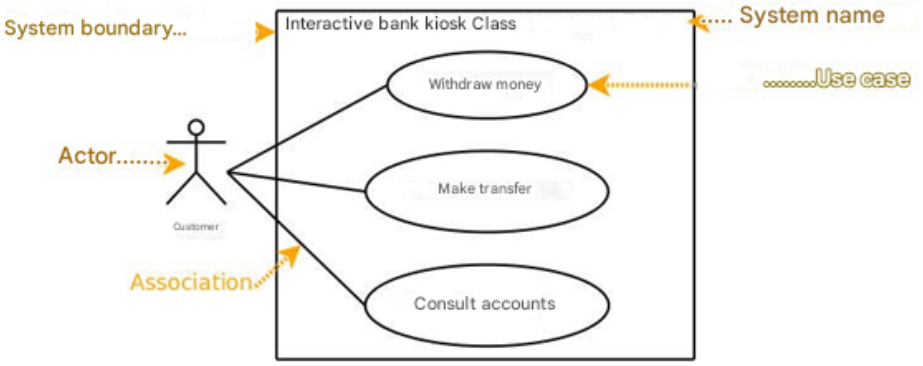


FIGURE 5 – : Simplified example of a use case diagram modeling a bank access terminal. The actors are on the outside and the use cases are on the inside.

6.1 Relationships in use case diagrams

6.1.1 Relationships between actors and use cases

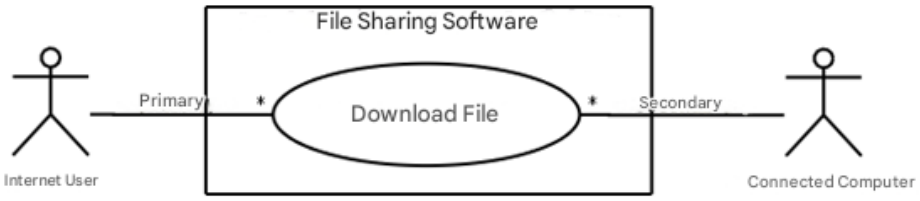


FIGURE 6 – : Use case diagram representing a file sharing software.

Association relationship An association relationship is a communication between an actor and a use case and is represented by a continuous line. When an actor can interact multiple times with a use case, it is possible to add multiplicity on the association on the use case side.

Main and secondary actors An actor is called primary for a use case when that use case provides a service to that actor. The other actors are then called secondary. A primary actor obtains an observable result from the system

while a secondary actor is requested for additional information. The "primary" stereotype is used to link the primary actor to the use case ; the "secondary" stereotype is used for secondary actors.

Internal use cases When a case is not directly linked to an actor, it is qualified as an internal use case.

6.1.2 Relationships between use cases

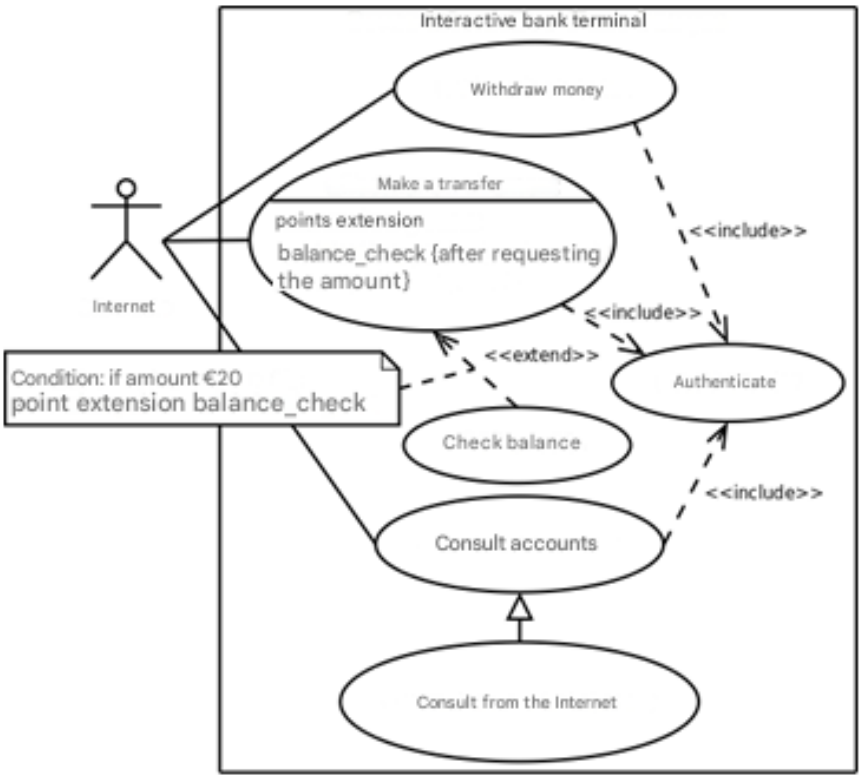


FIGURE 7 – : Example use case diagram.

There are mainly two types of relationships :
— • stereotypical dependencies, which are explained by a stereotype (the most used are inclusion and extension) ;
— • and generalization/specialization.

a. Inclusion relationship A case A includes a case B if the behavior described by case A includes the behavior of case B : case A depends on B. When A is requested, B is necessarily requested, as a part of A. This dependence is

symbolized by the stereotype "include". For example, access to information of a bank account necessarily includes an authentication phase with a username and password. Inclusions also make it possible to break down a complex case into simpler subcases. Figure 8.

Caution!!! Use cases do not follow one another, since there is no temporal representation in a use case diagram.

b. Extension relationship A use case A is said to extend a use case B when use case A can be called during the execution of use case B. Executing B can optionally cause A to execute : unlike inclusion, extension is optional. This dependency is symbolized by the stereotype "extend". An extension is often conditional. Graphically, the condition is expressed in the form of a note. The case is explained in Figure 8.

c. Generalization relationship A case A is a generalization of a case B if B is a particular case of A. This generalization/specialization relationship is present in most UML diagrams and is reflected in the concept of inheritance in object-oriented languages.

6.1.3 Relationships between actors

The only possible relationship between two actors is generalization : an actor A is a generalization of an actor B if actor A can be substituted by actor B. In this case, all use cases accessible to A are also accessible to B, but the reverse is not true. The symbol used for generalization between actors is an arrow with a solid line whose tip is a closed triangle designating the most general actor (as we have already seen for the generalization relationship between use cases).

Références

[1] G. Booch, J. Rumbaugh, I. Jacobson, *The Unified Modeling Language (UML) User Guide*, Addison-Wesley, 1999.

[2] Benoît Charroux, Aomar Osmani, Yann Thierry-Mieg, *UML 2 Synthèse et Exercices*, Pearson, édition française, ISBN 2-7440-7124-2, 2005.

[3] G. Booch et al., *Object-Oriented Analysis and Design with Applications*, Addison-Wesley, 2007.

[4] Laurent Audibert, Cours UML 2.0, disponible sur : `http://www.developpez.com`

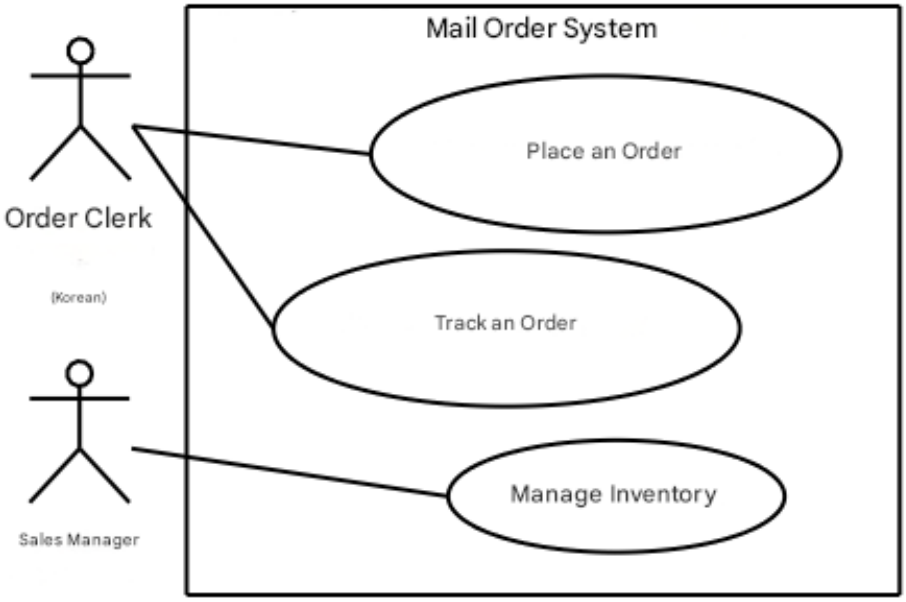


FIGURE 8 – : Relationships between actors.