

Chapter 3

Use Case Diagrams

As discussed in the previous chapters, textual requirements are an easy specification method but fails at specifying interactions between a system and its users. A *use case* describes an interaction scenario and its possible alternatives. A *use case diagram* graphically pictures several use cases, their actors, and their relationships.

3.1 Learning objectives of this chapter

The learning objectives of this chapter are the followings. At the end of this chapter, you shall

- know what is a (UML) use case;
- know what is a (UML) use case diagram;
- be capable for writing detailed (UML) use case descriptions;
- be capable of organising use cases in a (UML) use case diagram.

3.2 Use Case

3.2.1 Overview

A *use case* is a collection of interactions which delivers a valuable result to a user. A use case should realise a *goal* or objective of a user or a particular class of users. A more precise definition is given by Cockburn¹:

A use case captures a *contract* between the *stakeholders* of a *system* about its *behaviour*. The use case describes the system's behaviour under various conditions as the system responds to a request from one of its stakeholders, called the *primary actor*.

¹A. Cockburn. Writing Effective Use Cases. Addison-Wesley 2001.

The main applications of use cases are:

- early development steps;
- classifying requirements;
- guiding the rest of the development process;
- base of (acceptance) test cases.

Use cases are about *what* the system should do, from the perspective of a user. A use case describes a desired behaviour independently of implementation details. The goal of use cases is to capture all system-level functions that the users envision. Note that in relation to user requirements (see previous chapter) use cases will *refine* these requirements. Use cases are given a unique identifier to make them traceable. Let's look at an example.

Suppose that we would like to specify a telephone system, which comprises a number of telephone sets, a number of telephone lines and a switch center. The telephone system is used to make and receive telephone calls. Each telephone set is connected to a telephone line with a unique telephone number. Note that there may be more telephone sets connected to one line. In each telephone set, there is an answering machine functionality, which receives telephone calls, records and playbacks voice messages. The answering machine may be turned on or off. Moreover, there is a caller-id system, which, if enabled, shows the telephone number of the calling party when a call is received. The switch center takes care of receiving dial requests, and sending appropriate tones and signals to the lines, and eventually to the telephone sets.

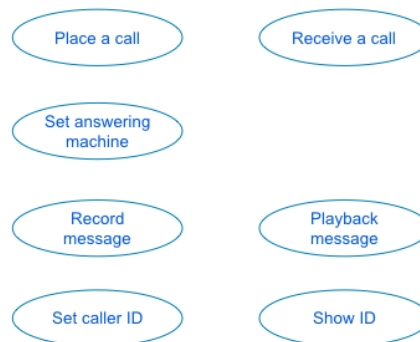


Figure 3.1: Use cases for the telephone system.

Use cases for this system are graphically depicted in Figure 3.1. You can notice that a use case is graphically represented by a solid ellipse with a name written in the middle. This is the notation adopted by the UML.

3.2.2 Detailed description

The detailed description of a use case includes the following parts:

- **pre-condition:** when a use case is available to its user
- **trigger:** what interactions initiate the use case
- **post-condition:** what the use case achieves
- **main scenario:** typical course of actions
- **alternatives:** alternative course of actions

Remember that use cases are specified at the system-level. They capture interactions initiated by and deliver results to the actors outside the system. Thus, pre-conditions, triggers, and post-conditions do not contain system details, such as internal state of a particular implementation.

| | |
|----------------|--|
| Use case ID | UC01 |
| Use case name | Place a call |
| Summary | A user places a telephone call. |
| Pre-condition | - the telephone set is connected to the telephone line "A", - the telephone set is on-hook, - there is no incoming call (it is not ringing) |
| Trigger | The user picks up the telephone hook of the telephone set connected to line "A" and dials number "B" |
| Post-condition | A communication between "A" and "B" is initiated. |
| Main scenario | (a) The user picks up the telephone hook connected to line "A" (b) If the line is free, the user receives a dial tone sent by the line (c) The user dials number "B" (d) The call request is forwarded to the switch center (e) If line "B" is not busy, the call request is forwarded to "B" and a tone is sent to "A" (f) "B" 's telephone rings (g) If somebody at "B" picks up the hook, the ringing tone at "A" is stopped and a telephone conversation is initiated. |
| Alternatives | (b) 1 If the telephone line is engaged in a conversation, the user will be connected to the same conversation. (b) 2 If the user does not dial a number for a certain amount of time, a permanent tone is emitted by the switch center, no further call will be accepted, the user has to replace the hook (e) 1 If line "B" is busy, and "B" does not have call waiting the user at "A" will receive a busy tone. (e) 2 If line "B" is busy, and "B" has call waiting the user at "A" will receive a call waiting tone from the switch center. When line "B" becomes free, sub-scenario (e)-(g) follows (g) 1 If nobody picks the call at "B", after a certain amount of time and a telephone set at "B" has its answering machine enabled, then the <i>Record Message</i> scenario at "B" follows. (g) 2 If nobody picks the call at "B", after a certain amount of time and no telephone set at "B" has its answering machine enabled, the user at "A" will receive a no response tone from the switch center. |

Figure 3.2: Detailed description of the "Place a Call" use case

Figure 3.2 gives the detailed description of the use case "Place a Call". Note that "Record Message" is a sub-scenario that is referenced here but must be specified elsewhere. Sub-scenario (e-g) is an internal reference to the scenario specified in this use case.

3.3 Actors

The value of a use case is relevant to a particular class of users. Such a class is represented by an *actor*. An actor represents the role of a class of users interacting with

the system. An actor may characterise a number of human users, or even a computer system interacting with the system under development. One user or system may play different roles in interacting with the system and thus, be represented by more than one actor. For example, a bank manager should be able to both create new accounts, as an instance of the "manager" actor and if she has an account in the same bank, she should be able to deposit and withdraw money, as an instance of the "customer" actor. Here is a definition of an actor:

An actor is a class of entities – being human or computer – falling *beyond the system boundaries*, interacting with the system.

3.4 Relationships

A use case diagram describes relationships between actors and use cases and among different use cases.

3.4.1 Association

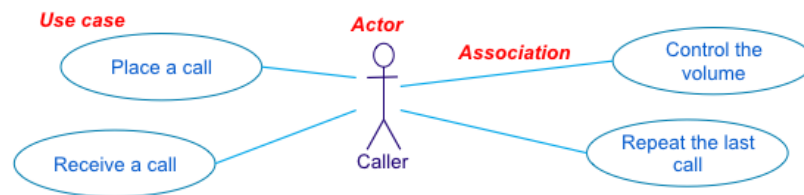


Figure 3.3: An actor, use cases and associations.

An association is a relation between an *actor* and a *use case*. It is the only possible relationship between an actor and a use case. An association indicates that an actor interacts with the system for the specified use cases. Associations between actors and use cases are not directed and have no name. They are represented by solid lines between actors and use cases. Figure 3.3 shows an example showing an actor, use cases, and association relations.

3.4.2 Generalisation

Actors may be related by *generalisation* relations. The source of a generalisation is a special case of the target. This means that the target can always be replaced by the source but not vice versa. A generalisation relation is represented by a directed arrow with a solid line and an empty triangle as arrow head. Figure 3.4 shows an example. A *User* is a generalisation of a *File Owner*. A *File Owner* is a generalisation of an *Administrator*. This means that in any use case, a *User* can be replaced by a *File Owner* but not the other way around. Intuitively, this means that there exist actions

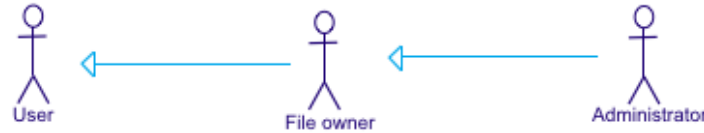


Figure 3.4: Actors and generalisation relations.

that a *File Owner* can perform but a *User* cannot. Think about deleting a file. In most operating systems, only the owner of the file can delete it. Administrators have often special rights allowing them to delete any file they want.



Figure 3.5: Use cases and generalisation relations.

Use cases may also be related by a generalisation relation. Figure 3.5 shows an example.

3.4.3 Dependencies: Include

A use case may depend on another use case to achieve its goal. An *include* dependency occurs when a use case *always* uses another one. For instance, a use case "Execute process" will always first run the use case "Check access rights" to know if the current user is allowed to run this process. Figure 3.6 shows such an example. The dependency must be read "Use case Execute always includes use case Check access rights". The direction of the arrow goes from the primary use case to the included use case.



Figure 3.6: Use cases and include dependencies.


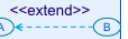
3.4.4 Dependencies: Extend

A use case may also call another use case but only under special circumstances, for instance, as alternatives. In that case, the dependency is an "extend" one. Figure 3.7 shows an example. The dependency must be read "Use case Get help on registration extends use case Register". The arrow goes from the extension to the extended use case.

Figure 3.8 summarises the include and extend dependencies.



Figure 3.7: Use cases and extend dependencies.

| |  |  |
|-------------------------|---|--|
| Read | A includes B | B extends A |
| A can operate without B | no | yes |
| B is aware of A | no | yes |

Include:

An Include relationship specifies that **a UseCase contains** the behaviour defined in **another UseCase**.

Extend:

A relationship **from an extending** UseCase **to an extended** UseCase that specifies how and when the behaviour defined in the extending UseCase can be inserted into the behaviour defined in the extended UseCase.

Figure 3.8: Include and extend overview

3.5 Complete Diagrams

A use case diagrams puts all the ingredients described so far together. Consider as an example a file system described as follows. In the file system, users can create new files, execute, display (on different output devices) and delete existing files. There is a special type of delete, which removes the file permanently from the file system. The file system makes use of an access right system which specifies who the owner of each file is and what operations are allowed by which users. The owner of each file may change the access rights to the file and give or take other people's permissions to access the file. In addition to the person who creates the file, the administrator is considered the owner of all files. Figure 3.9 depicts the use case diagram for this file system example.

3.6 Conclusion

A use case is a textual description illustrating an interaction scenario between users and the system. Use cases and their relationships can be pictured in a UML use case diagram. Use cases are more involved than textual requirements. They require some basic knowledge about simple UML notations (like dependency arrows). In contrast to textual requirements, use cases show interactions. The drawback is that they are less compact and require more knowledge. In the next chapter, we will discuss informal techniques to specify the structure of a program: the class diagram.

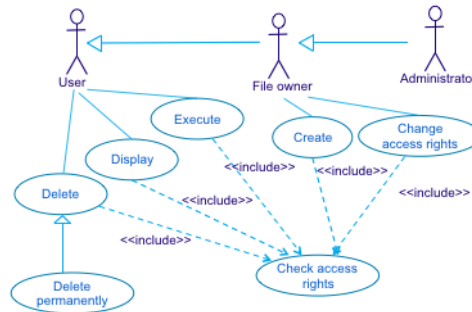


Figure 3.9: Use case diagram for the file system example.

3.7 Exercises

Exercise 3.7.1. Consider the Bookshop specified below.

The bookshop has a number of books from different titles. Each book may appear in two versions: hard-cover or soft-cover and thus may have two different prices. A user shops for a book by searching for the book title and receiving the prices of available versions of the title. Afterwards, the user either pays the price and buys the book using a credit card or cancels the purchase. Credit card payment concerns a credit card number, name of the owner, expiration date and the amount to be withdrawn. A credit card payment should be authorised by the bank. The bookshop owner can add books (of possibly new titles) to its stock.

1. Draw a use case diagram for the system
2. Give a detailed description of each use case

Exercise 3.7.2. Consider a transport company specified below.

The company owns a number of vehicles of different sizes which can transport goods. A client submits a request for transportation by specifying the size of the package to be transported, its source and destination. The distance between source and target determines the amount of time during which the vehicle will be en route. The company then sends an offer to the client by finding the first possible period during which a vehicle of an appropriate size is available. If the client agrees with the terms of the offer, it provides an account number and the authorisation to withdraw the amount of the offer from the account. Upon a successful transaction with the bank (given the account information provided by the user), the amount of money will be transferred to the company's account and the company will schedule the transport as specified in the offer.

1. Draw a use case diagram for the system
2. Give a detailed description of each use case



Figure 3.10: Schematic view of the railway system.

Exercise 3.7.3. Consider the embedded system of a railway controller. The railway controller controls three tracks, two signals, and a crossing as depicted in Figure 3.10.

A signal shows either green or red. Each track is equipped with a sensor that shows the presence of a train and the direction of its movement. The supervisor of the control system may decide to change the status of the signal (from green to red or vice versa), or change the status of the crossing (from closed to opened or vice versa).

1. If the supervisor decides to change a signal to green, the controller should make sure that:

- (a) there is no train coming in the opposite direction towards the signal,
- (b) the crossing is closed, and
- (c) the signal in the opposite direction is red.

If the above checks are successful, then the signal will be changed or otherwise the supervisor will be notified of the impossibility.

2. If the supervisor decides to change a signal to red, the controller should make sure that there is no train on the track before the signal. If this is the case, then the signal will be changed or otherwise, the supervisor will be notified of the impossibility.
3. If the supervisor decides to change a crossing to opened, the controller should make sure that:

- (a) there is no train on the track, where the crossing resides, and
- (b) the signals before and after the crossing are red.

4. Closing a crossing is always possible.

Your tasks:

- (a) Draw a use case diagram for the system
- (b) Give a detailed description of each use case