How to write extension points in use cases correctly

The following two use cases show an example of a correct application of an extension point.

In use case 17 at step 4, the receiver can either say “no” or “yes”. The right way to do this is to choose the default situation and finish the flow of events with that choice. In our case, we choose “yes” as the default answer and continue.

We can then add an extension point at step 4 and refer to use case 19 for the “no” answer. By taking the extension point, we discard steps 4 and all following steps. Instead, we take the steps in use case 19. An important implication is that the entry condition of use case 19 needs to be met after step 3 of use case 17.

Of course, there are even more extension points in the example below, e.g. if the receiver's terminal is in DND mode.

Use case Number: 17
Use case Name: Connect to a chat
Participating Actors: Passengers “sender”, “receiver”
Flow of events:

1. The sender enters a seat-number to which passenger terminal he would like to connect.
2. The system display notification to the sender that it tries to connect.
3. The system asks the receiver whether he would like to start a chat with the sender.
4. The receiver answers “yes”.
5. The system notifies the sender that the receiver agreed to participate in the chat.
6. The system establishes the chat connection between sender and receiver.

Entry Condition: The sender terminal is active, not locked, not in DND mode, and the chat plug-in is active.
Exit Condition: The chat connection between sender and receiver is now established.
Extension points: ***4: Use case 19 “Notify a rejection of chat”*** , [...]

Use case Number: 19
Use case Name: Notify a rejection of chat
Participating Actors: Passengers “sender”, “receiver”
Flow of events:

1. The receiver answers “no”.
2. The system notifies the sender that the receiver rejected to start the chat with him.

Entry Condition: The terminal of the receiver is active, not locked, not in DND mode, and waits for a decision whether to start a chat with the sender or not.
Exit Condition: The sender has been notified that the receiver rejected to start a chat with him.

How to include use cases

The following two use cases show the correct application of an inclusion relation of use cases. At the right step in the flow of events, we clearly specify what use case we want to include.

An included use case is comparable to a method call. We need to ensure that we established the entry condition of the included use case at the point of the inclusion and that the exit condition of the included use case enables us to continue with the flow of event.

In our example, this is the case as the passenger terminal is off at this time and the seat is occupied. After we take all steps of the included use case, we also need to be sure that the exit condition of use case 2 leads to the exit condition for use case 1.

Use case Number: 1
Use case Name: Start the system
Participating Actors: Flight Attendant, Passenger
Flow of events:
1. The flight attendant presses the start button
2. The system starts the flight attendant terminal
3. The system starts the security officer terminal
4. For each seat: ***Include use case 2 "Start passenger terminal"***

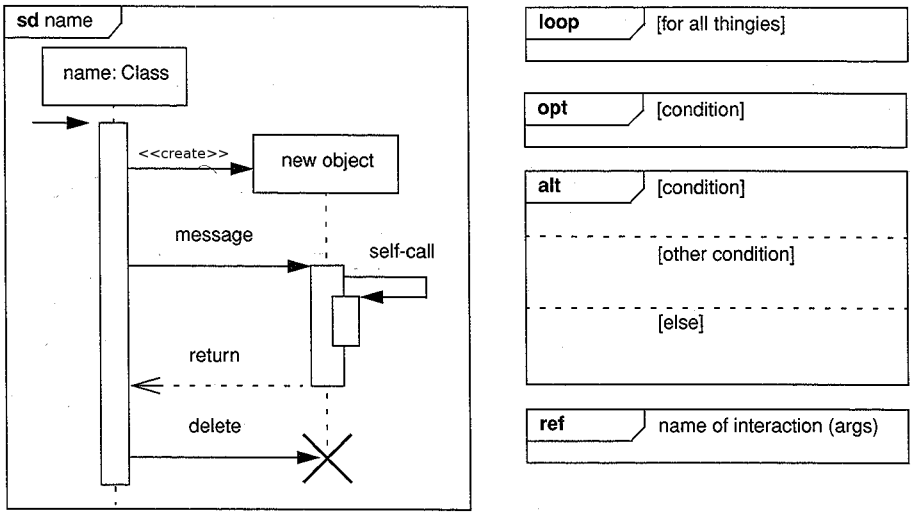
Entry Condition: The system is powered down.
Exit Condition: All terminals are operational.
Quality Requirements: The system must be operational within 15 minutes.

Use case Number: 2
Use case Name: Start passenger terminal
Participating Actors: Passenger
Flow of events:
1. The system starts the passenger terminal
2. The passenger terminal downloads the applications and the data from the server according to it's class

Entry Condition: The passenger terminal is off.
Exit Condition: The passenger terminal is operational.

How to draw sequence diagrams

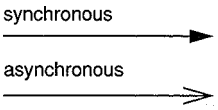
We want to point out some common mistakes when drawing sequence diagrams. In the picture below, you see the most important concepts put together.



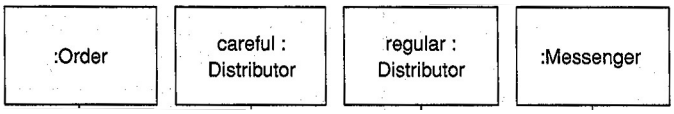
Activation of objects: You indicate activation of an object by drawing a box along the lifeline of the object. If you send a synchronous message to another object, the caller needs to be active as long as the callee is active.

However, if you choose to have an asynchronous message, the activation of the caller may stop immediately after having sent the message to the callee.

And remember:



Name of the object: If you have several objects of the same type in the sequence diagrams use the following notation for the boxes describing the object.



In this example, there are four objects involved in a sequence chart, one of type *Order*, one of type *Messenger*, and two of type *Distributor*. To be unambiguous, we give the two objects of type *Distributor* a name. We could also give names to the other objects, but it's not really necessary.

All pictures are borrowed from *UML Distilled, Third Edition, Martin Fowler, Addison-Wesley 2004, ISBN 0-321-19368-7*