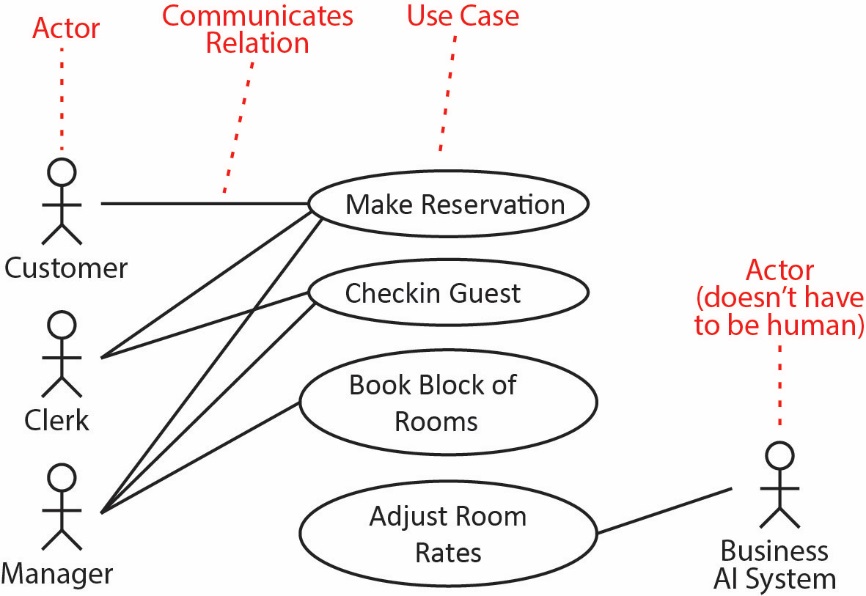
**Use Case Diagrams**

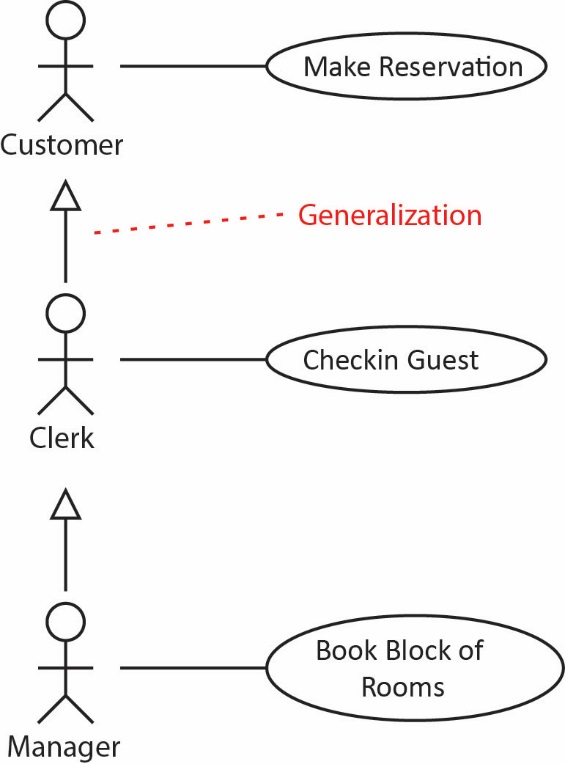
**Use Cases**

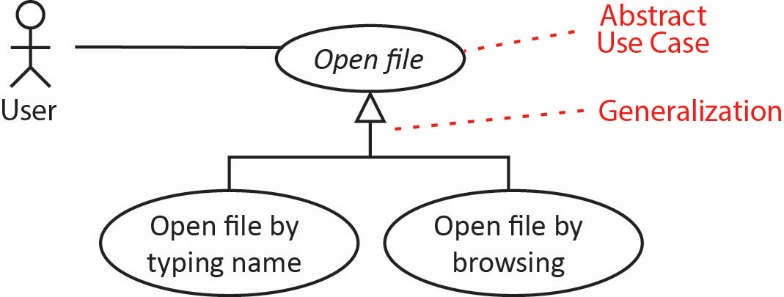
1. **Definitions**

* A *Use Case* represents a goal a user has when using a system. The name of the use case should reflect this goal.
* An *Actor* is a user of the system in a particular role. An actor is usually a human but can also be another system.
* A *Communicates Relation* exists if the actor plays some part in accomplishing the use case.
* A *Use Case Diagram* is a visual representation of the items above.

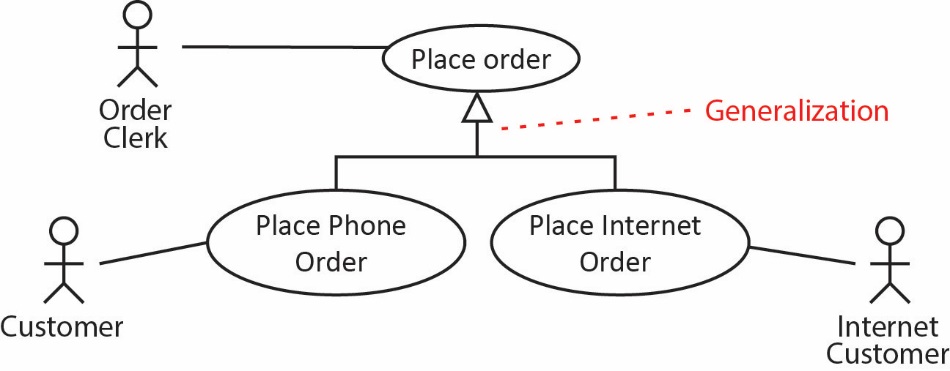
1. **Example** – A (partial) Use Case Diagram for a Hotel Management System



1. Use case names should begin with a strong verb. You should also imply timing considerations by putting use cases that typically occur first on top of others that occur later.
2. **Generalizations**
3. Actors can be generalized when an actor can play more than one role. This typically results in a Use Case Diagram that is easier to understand. In the example on the right, a *Clerk* can do anything a *Customer* can do and a *Manager* can do anything a *Clerk* (or *Customer*) can do.
4. Use Case generalization is used when you find two or more use cases that have commonalities in behavior, structure, and purpose. When this happens, you can describe the shared parts in a new, often abstract, use case that is then specialized by child use cases. In the example below *Open File* is abstract.



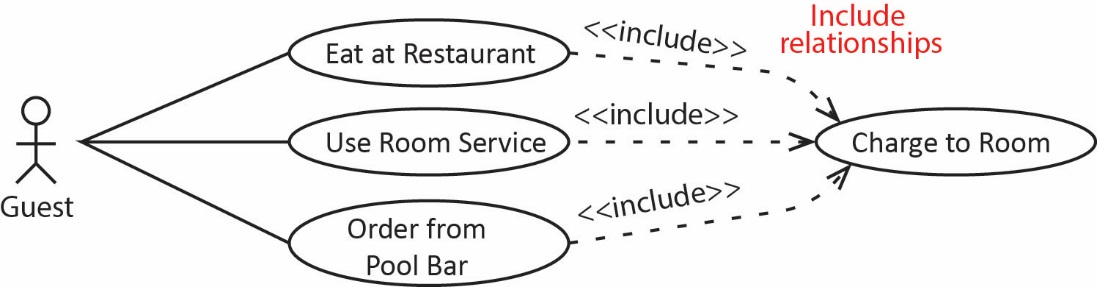
In the example below there are three ways to place an order:



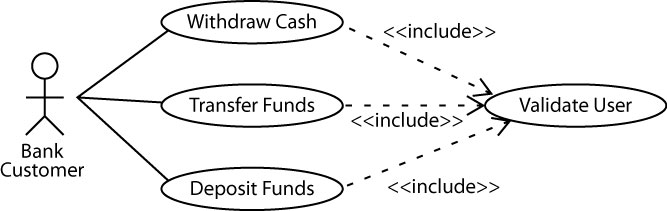
1. **Include Relationships**
2. There are two main types of extension mechanisms for use cases: **Include** and **Extend** relationships. We use an **Include** relationship to show:
3. Common functionality between use cases
4. The use of an existing component
5. The situation where we are creating a reusable component.

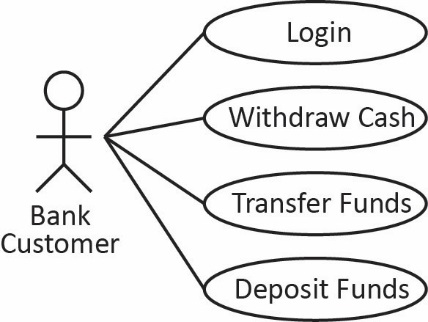
The include relationship connects a base use case to an inclusion use case. It describes a behavior that is inserted into a use-case. The base use case has control of the relationship to the inclusion and can depend on the result of performing the inclusion, but neither the base nor the inclusion may access each other's attributes. The inclusion is in this sense encapsulated, and represents behavior that can be reused in different base use cases.

1. Example – A guest in a hotel can obtain services in the hotel and charge them to his room.

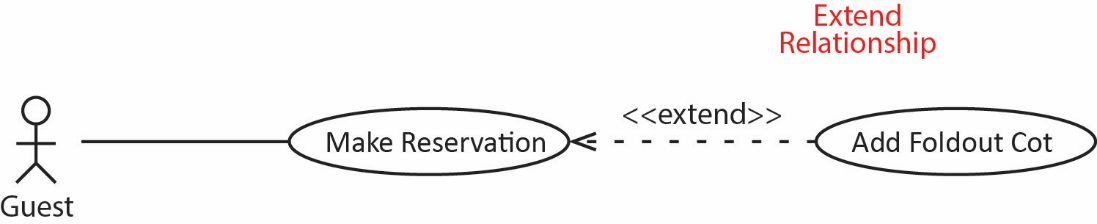


1. Example:

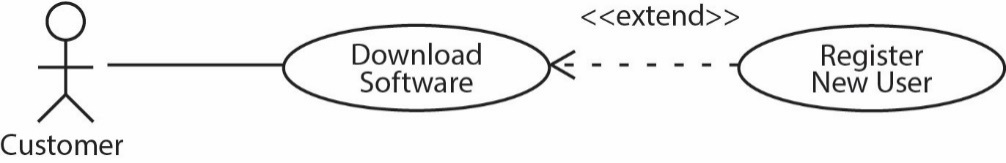


You see this type of example a lot on the internet. It may be technically true, but I think it clutters the diagram too much. I prefer to simply show a *Login* use case, as shown on the right and for the other use cases specify in the description a precondition that the *Login* use case has succeeded.

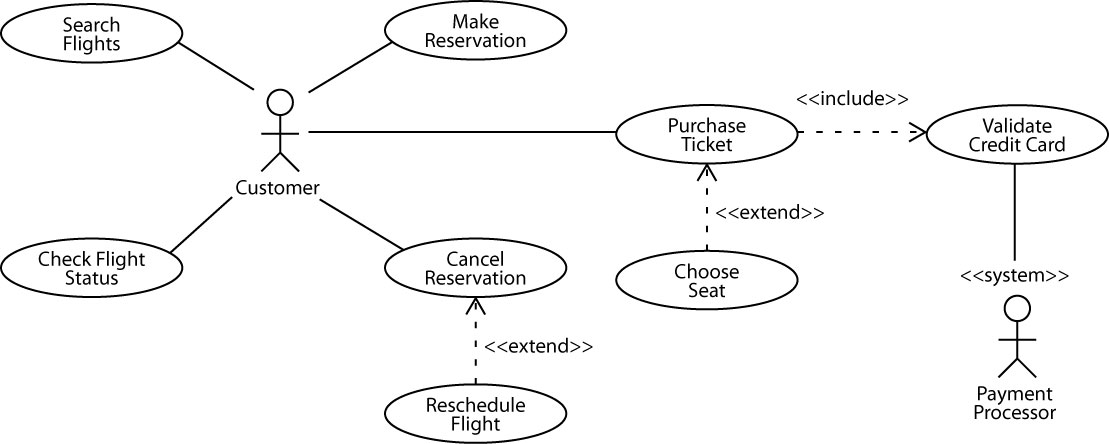
1. There are also several dangers associated with identifying shared functionality:
2. A serious effort at this might force us into the traditional top-down functional decomposition of the system which could produce a design that is inflexible and doesn’t adhere to OO principles.
3. Harder for someone without UML experience to understand the diagrams.
4. A project will probably utilize several versions of the use case model, simple versions those that show more complex relationships between use-cases. Example: <http://www.uml-diagrams.org/examples/online-shopping-use-case-diagram-example.html?context=uc-examples>
5. **Extend Relationships**
6. We use an **Extend** relationship when a use case incorporates two or more significantly different scenarios. In other words, depending on the circumstances, one of several different things can happen. We usually choose to document this as a main (normal) use case, when everything goes correctly, and as one or more subsidiary (unusual, variant) cases that extend the main use case. The extension is conditional, which means its execution is dependent on what has happened while executing the base use case.
7. Example – When a guest is making a reservation, they may decide they need a foldout cot. This may be a valid extension use case because perhaps it would need to check inventory to see if one if available and to notify maintenance to deliver the cot.

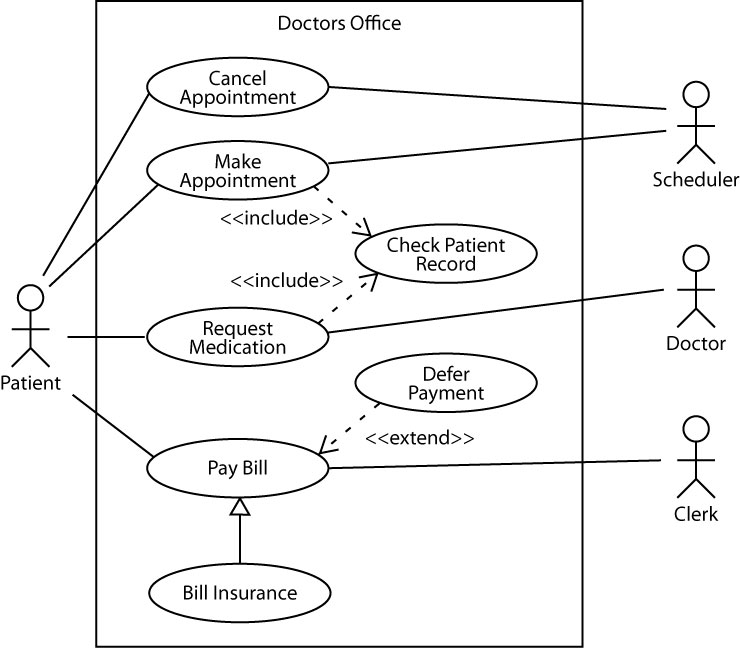


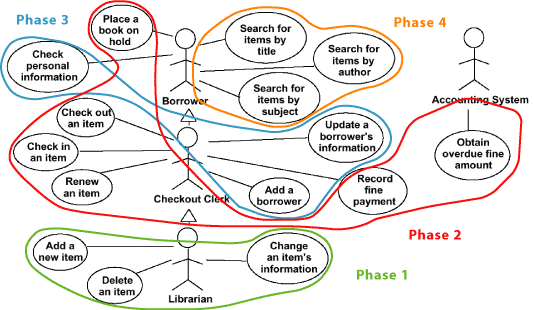
1. Example



1. A use case model is good for:
   1. Capturing and validating requirements.
   2. Prioritizing use cases to determine the scope of the system or sprint.
   3. Planning and scheduling.
   4. Designing Tests.
   5. Writing a user's manual, help system, or training guide.
2. **Examples – Airline Reservation System:**

****

1. Example – Doctor’s Office. Note, a box can be drawn around the use cases to denote the **system boundary**. The actors are drawn outside the system boundary. This is most useful when you have complicated systems that have been broken down into components or subsystems.
2. **Planning –** A Use Case diagram can be used to specify the phases that will implement the Use Cases, as shown in the figure below. Note, in a similar way you could convey Teams, Components, etc.



1. **Example –** Consider this example of an attempt at a use case diagram:

<http://stackoverflow.com/questions/21404278/is-my-use-case-diagram-too-complicated-and-activity-diagram-too-dense>

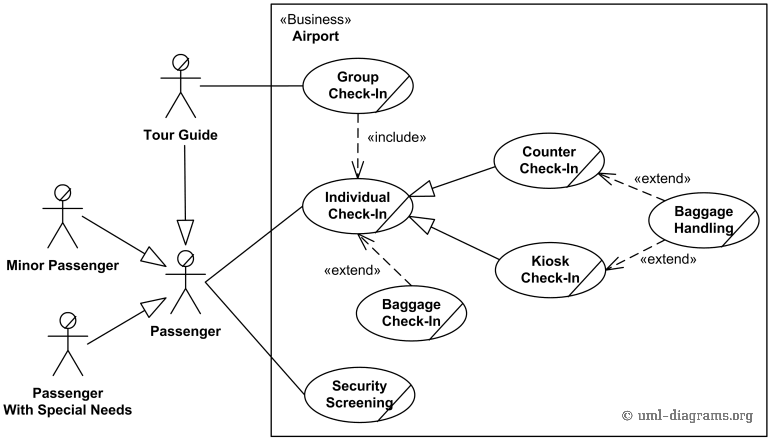
The solution has many problems. **Note that I regularly see use case diagram like this submitted for the project for this course.**

**Practice Homework**

1. Interpret this use case model:



1. Interpret this use case model:



Source: <http://www.uml-diagrams.org/airport-checkin-uml-use-case-diagram-example.html>

**Documenting Use Cases**

We will use a different approach for describing use cases: user stories.

1. **Describing a Use Case**
2. A *use case description* is used to document a use case. It should:
3. Cover the *full sequence of steps* from the beginning of a task until the end.
4. Describe the *user’s interaction* with the system, **not** the computations the system performs.
5. Be written so as to be as *independent* as possible from any particular user interface design.
6. Only include actions in which the actor interacts with the computer, **not** actions a user does manually
7. Example

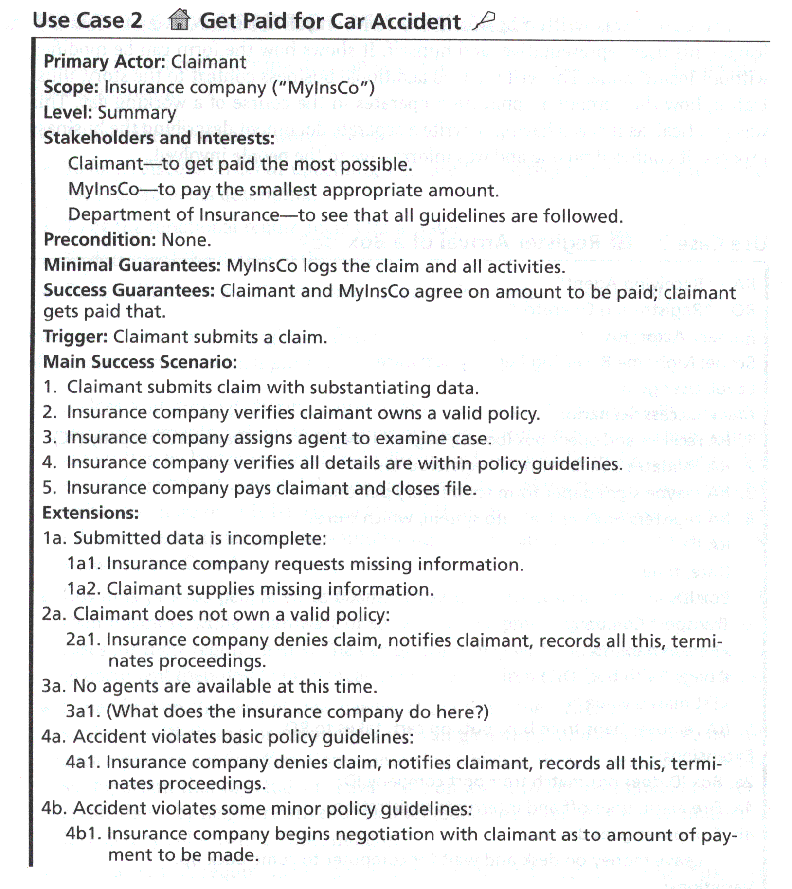
|  |  |
| --- | --- |
| Use Case ID | 1 |
| Use Case Name | Make Reservation |
| Actor(s) | Guest, Clerk, Manager |
| Precondition(s) | Actor is logged in |
| Postcondition(s) | *n/a* |
| Normal Flow | 1. System prompts for arrival date, number of nights, number of guests 2. User enters arrival date, number of nights, number of guests. 3. System shows available rooms and prices 4. User selects a room. 5. System prompts for credit card information 6. User enters credit card information 7. System prompts for confirmation 8. User confirms 9. System completes financial transaction 10. System stores reservation data in management system 11. System stores reservation in user’s reservations list |
| Alternate Flow(s) | 4a User indicates that they have a pet   1. System prompts for type of pet and weight 2. User enters type of pet and weight 3. System records this on reservation   4b User indicates that they need a foldout cot   1. System verifies availability 2. System informs house keeping of delivery date |
| Exception(s) | 8a. User doesn’t confirm. The use case ends.  9a. Credit card is declined   1. User is informed 2. Return to step 5 in main flow |

1. **Examples:**

<http://epf.eclipse.org/wikis/openup/core.tech.common.extend_supp/guidances/examples/use_case_spec_CD5DD9B1.html>

<http://tynerblain.com/blog/2007/04/09/sample-use-case-example/>

1. **Example**



Source: <http://www.cs.colorado.edu/~kena/classes/6448/s05/reference/usecases/examples.html>

1. **Use Case Senario –** A use case scenario is an instance of a use case. Many scenarios can be used to describe a use-case. It is a specific occurrence of the use case: a specific actor, at a specific time, with specific data. These are used for understanding business processes and usability design.

More information:

<http://tynerblain.com/blog/2007/04/10/what-are-use-case-scenarios/>