Use Case Diagrams

- Basic Concepts
- Actor
- Use Case
- <<includes>>
- <<extends>>

Basic Concepts

- Use cases are not inherently object-oriented
  - An external (user) view of the system
  - Intended for modelling the dialog between the users and the system

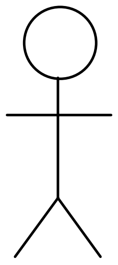
- The main concepts in use cases are
  - Actor
  - Use Case
  - <<includes>>
  - <<extends>>

Actor

• An Actor is a role of an object or objects outside of a system that interacts directly with it as part of a coherent work unit (a use case)

- One physical object (or class) may play several different roles and be modeled by several actors

• Notation



Reservation Agent

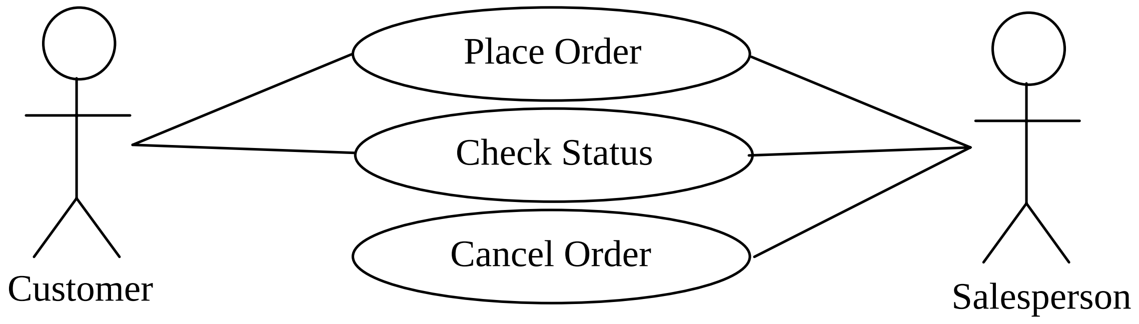
• Example actors for an Airline Reservation system

- Airline administrators (fare/schedule setting)
- Travel Agent
- Airline Reservations Agent
- Check-in Agents at Airport
- Gate Agent at Airport
- …

Use Case

- A Use Case captures some actor-visible function
  - Achieves some discrete (business-level) goal for that actor
  - May be read, write, or read-modify-write in nature

- Notation



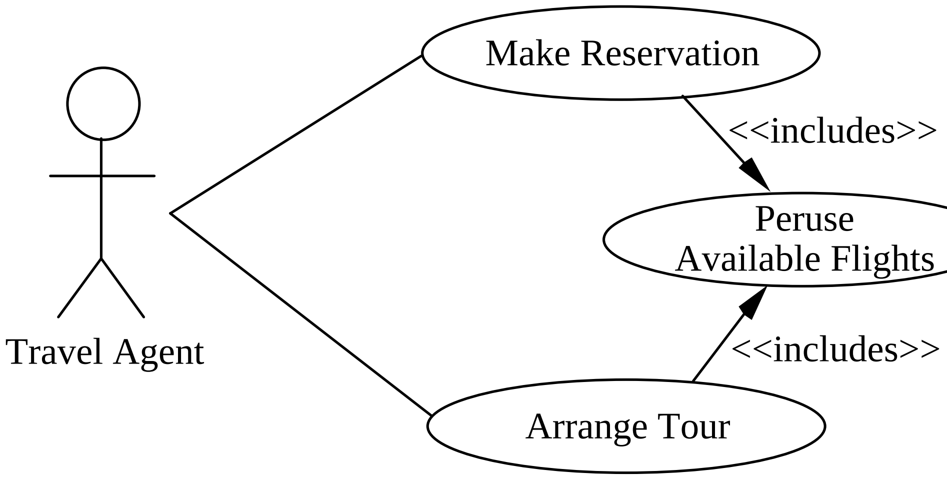
- Use cases within an Airline Reservation system might include
  - Checking in for a flight
  - Assigning a seat
  - Checking baggage
  - …

<<includes>>

• One common fragment of a user-perceivable action has been pulled out into a separate use case

- Like a “use case subroutine”

• Example



- Make Reservation and Arrange Tour both depend on Peruse Available Flights
    * Note that the arrows go from the dependent use cases

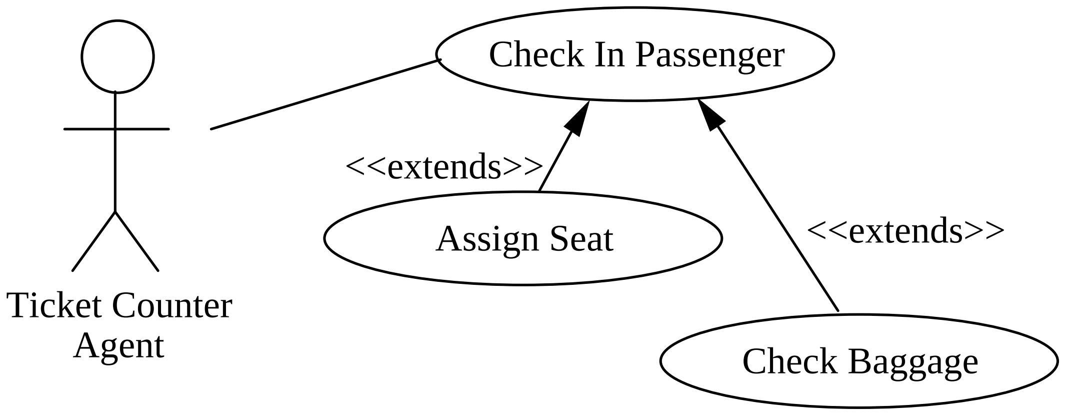
• Typically used when the same unit of functionality is part of more than one use case

- The base use cases are, in a sense, incomplete without the included use case

<<extends>>

• A significant alternative course of action exists within the use case
  - Like “use case inheritance”

• Example



  - Assign Seat and Check Baggage both depend on Check In Passenger
    * Note that the arrows go from the dependent use cases

• Typically used when there are important, *optional* variations on the basic theme of the base use case
  - The base use case is complete in and of itself

Key Points

- Use cases are not inherently object-oriented
  - An external (user) view of the system
  - Intended for modelling the dialog between the users and the system
- An Actor is a role of an object or objects outside of a system that interacts directly with it as part of a coherent work unit (a use case)
- A Use Case captures some actor-visible function that achieves some discrete (business-level) goal for that actor
- <<includes>> and <<extends>> allow common fragments of use cases to be pulled out into a separate use cases
  - <<includes>> is like a “use case subroutine”
  - <<extends>> is an alternative course of action