Case study

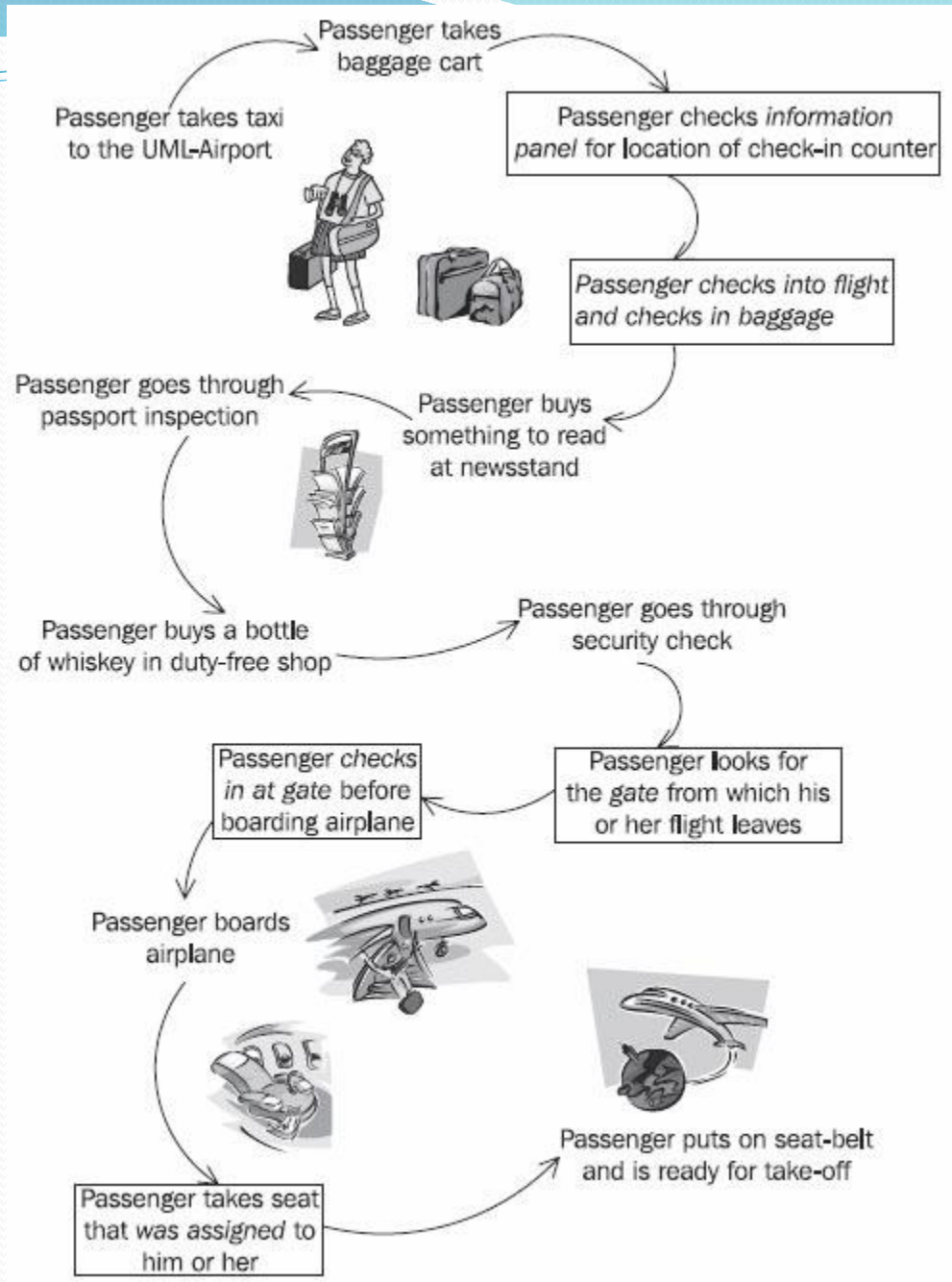
Case study to discuss relationships of classes

Class relationships

UML interaction Diagrams

Case Study - UML Airport

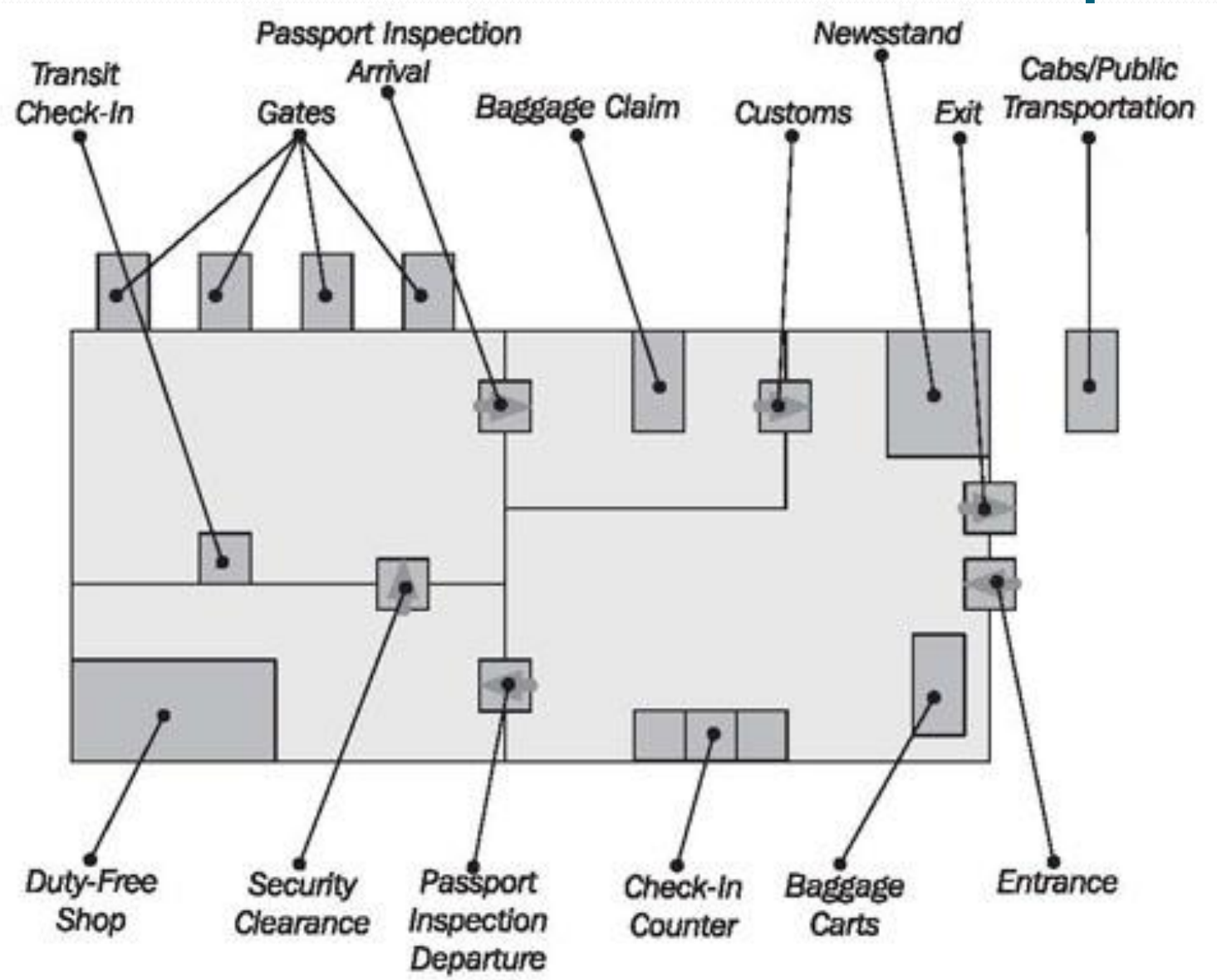
- We will restrict our example to those areas of the airport that passengers are in contact with during departure.
- Figure 2.1 shows the various stages that passengers go through until they are seated in the airplane
- Not all stages passengers go through are related to passenger services.


Passenger takes baggage cart
Passenger checks information panel for location of check-in counter
Passenger takes taxi to the UML-Airport
Passenger checks into flight and checks in baggage
Passenger goes through passport inspection
Passenger buys something to read at newsstand
Passenger buys a bottle of whiskey in duty-free shop
Passenger goes through security check
Passenger checks in at gate before boarding airplane
Passenger looks for the gate from which his or her flight leaves
Passenger boards airplane
Passenger takes seat that was assigned to him or her
Passenger puts on seat-belt and is ready for take-off

Exceptions

- The passenger only has carry-on luggage.
- Passenger doesn't buy anything at the newsstand.
- Passenger is running late and now has to check in as quickly as possible.
- Passenger loses his or her boarding pass.
- Passenger arrived by plane and merely has to change planes, i.e., that he or she doesn't leave the transit area.
- Passenger checks in, but falls asleep in the waiting area, and misses flight, despite being called repeatedly.
- Passenger doesn't get through passport inspection because his or her passport has expired.

Schematic illustration ofAirport



Related to passenger services

- Many areas around the main passenger services are related to passenger services. Some examples are:
  - Ticket sales
  - Newsstand
  - Duty-free shop
  - Passport inspection/immigration
  - Flight control
  - Information desk
  - Baggage check-in and transportation
- Passenger services have to exchange data with some of these areas.

Ticket booklet

- The plane ticket consists of the **actual ticket** and up to **four additional sections**. The *ticket* is the little booklet that has a separate coupon for every part of the trip. For example, a ticket could contain a coupon for the flight from Zurich to Frankfurt, one for the flight from Frankfurt to London, and one for the return flight from London to Zurich. Each time at check-in the appropriate coupon will be exchanged for a boarding pass. The ticket always stays with the passenger.

flight and a flight number

- We distinguish between a flight and a flight number. For instance, a *flight number* could be **PK708** or PK716. It stands for a regular flight that occurs at a certain time from the departure airport to the destination airport. A *flight*, on the other hand, would be, for example, **PK708 on 26th January, 2016**. It is, so to speak, an execution of a flight number. A flight could be canceled due to bad weather. A flight number is used as long as the airline offers a certain flight regularly.

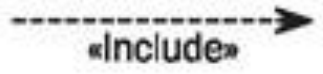
Passenger services

- Three options for check-in(issuing boarding pass):
- **Normal check-in** with luggage at a normal check-in counter.
- **Express check-in** without luggage at a special check-in counter.
- **Automated check-in** without luggage at a machine.

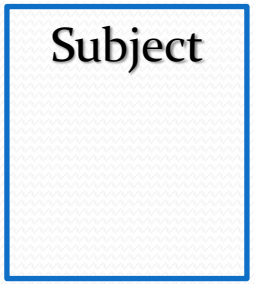
Use Case

- **Association** is the relationship between actor a use case. It indicates that actor can use a certain functionality of business use case.

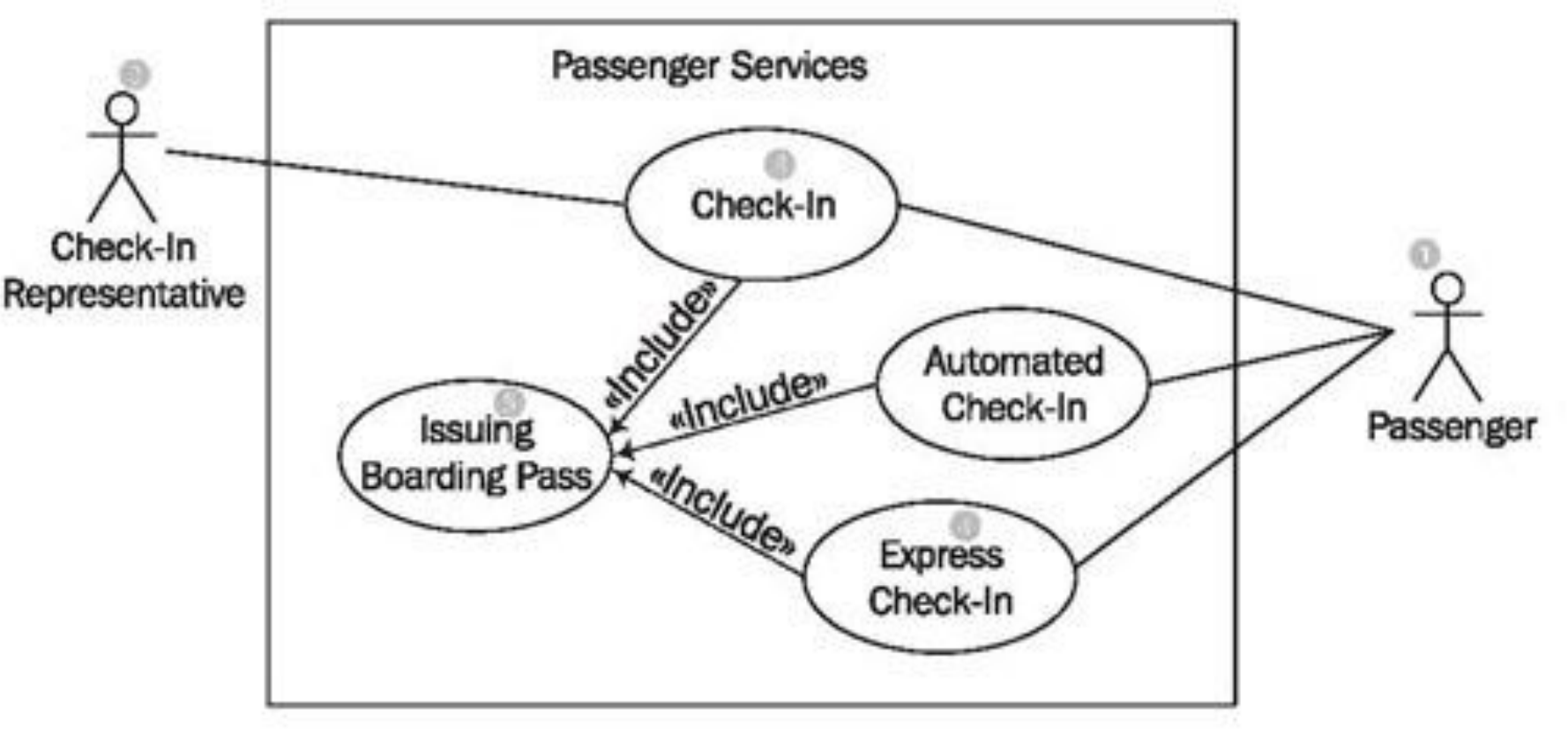
- **Include Relationship** is a relationship between 2 business use cases. Business use case on the side to which the arrow points is included in the use case on the other side of the arrow.



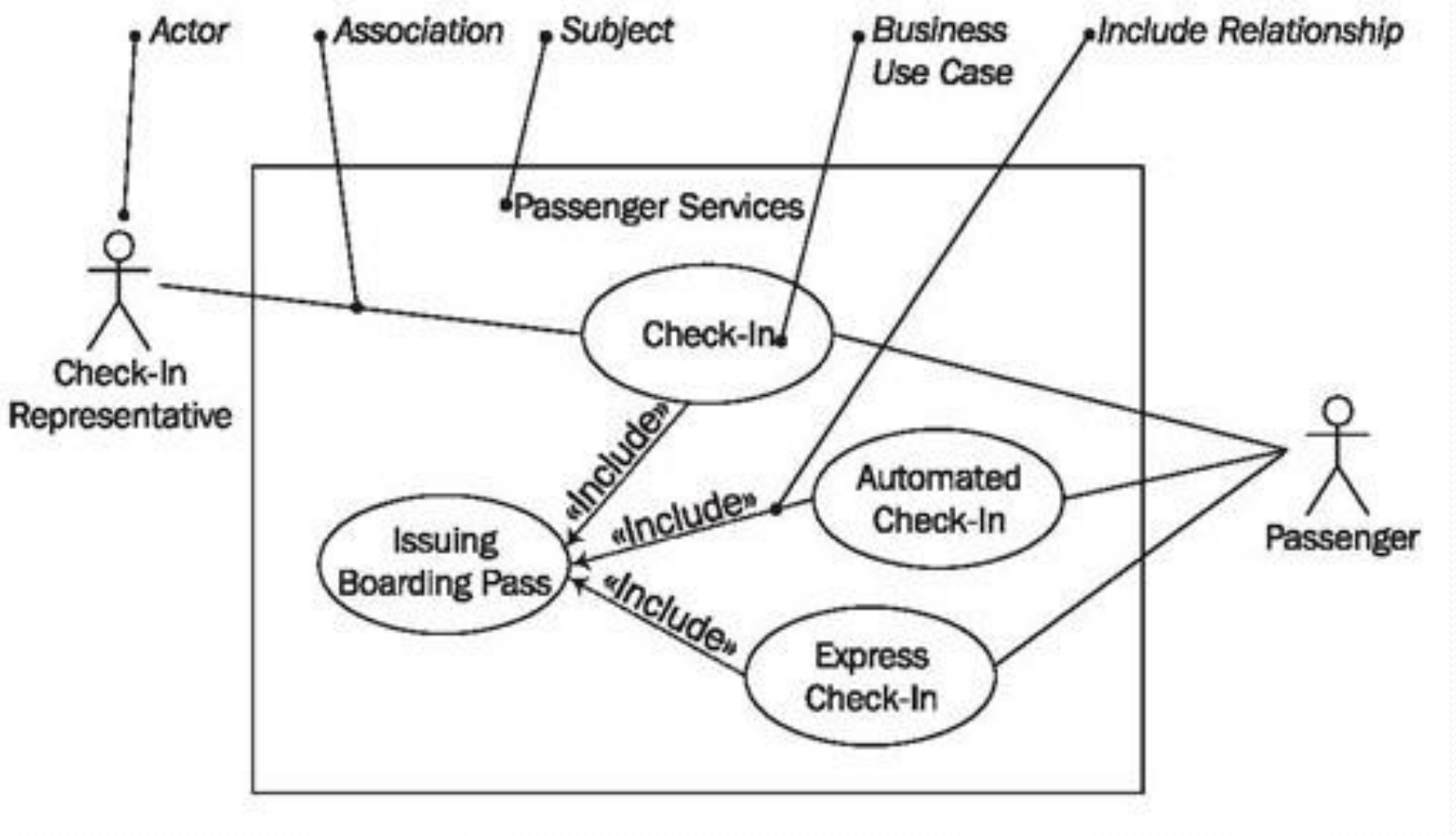
- **Subject** describes a business system that has one or more business use cases attached to it. A subject is represented by a rectangle that surrounds attached business use cases and is tagged with a name.



Use Case: Issuing boarding pass



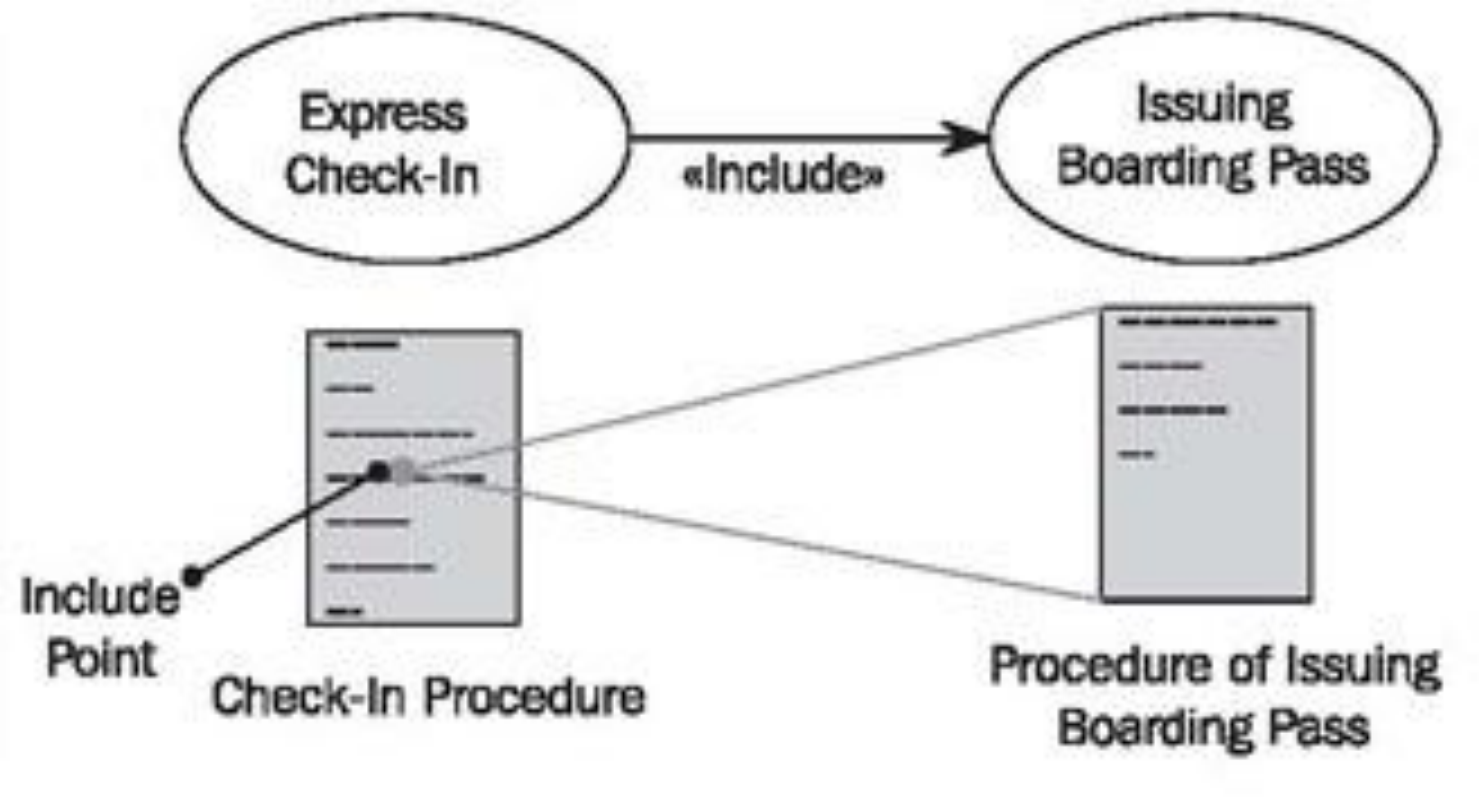
Use Case: Issuing boarding pass



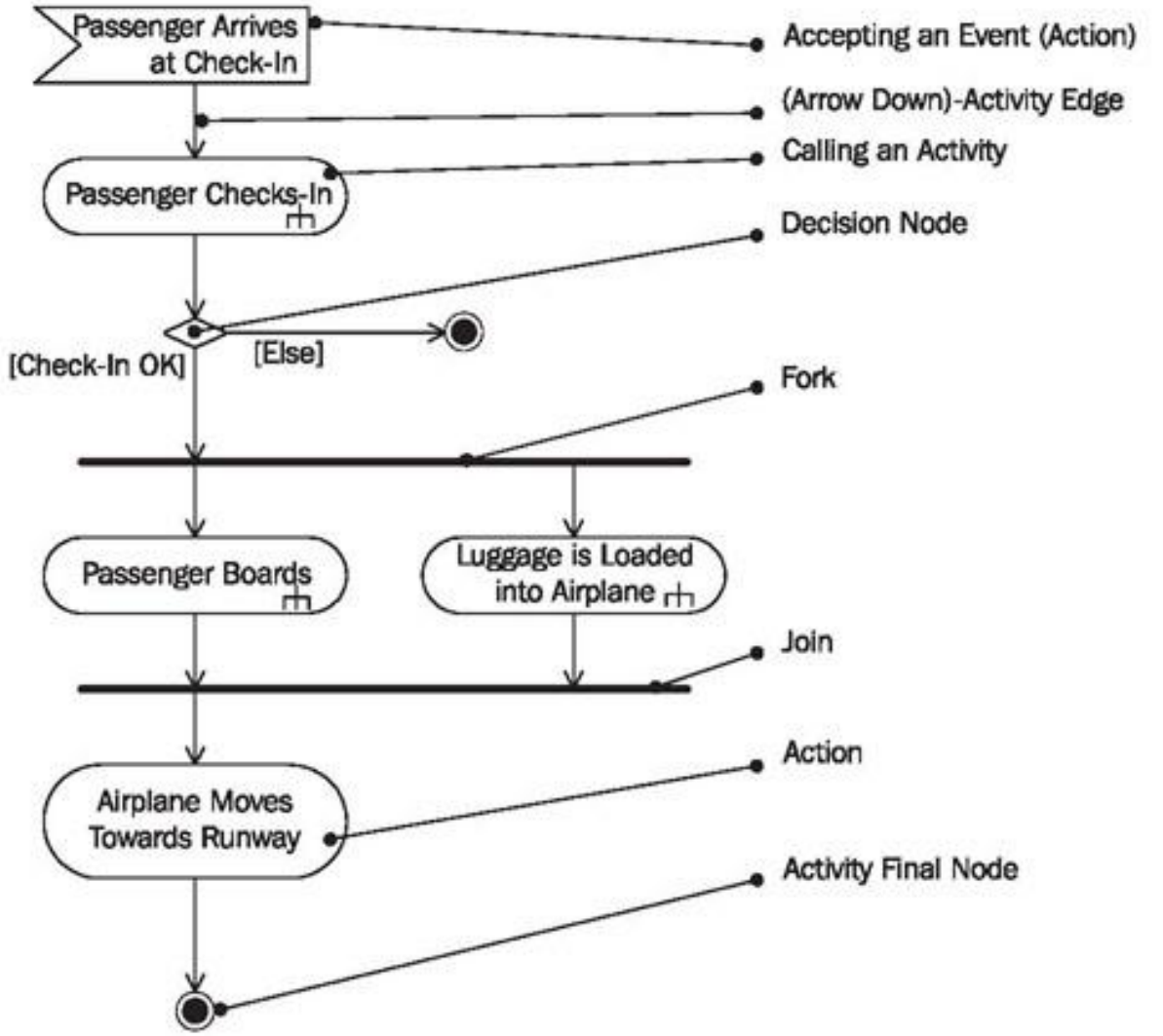
Limitations of Use case

- A use case diagram **does *not* document a meaningful order** in which business use cases could be conducted.
- The actor *check-in representative* also has an association to the business use case *check-in*. This means that not only the passenger, but also someone who represents him or her can check in.
- The actor, *passenger*, also has an association to the use case *check-in* means that the passenger and the check-in representative can both check-in.
- **However, what the diagram does not show clearly is that it *does not* mean that they perform the check-in together.**

include relationship

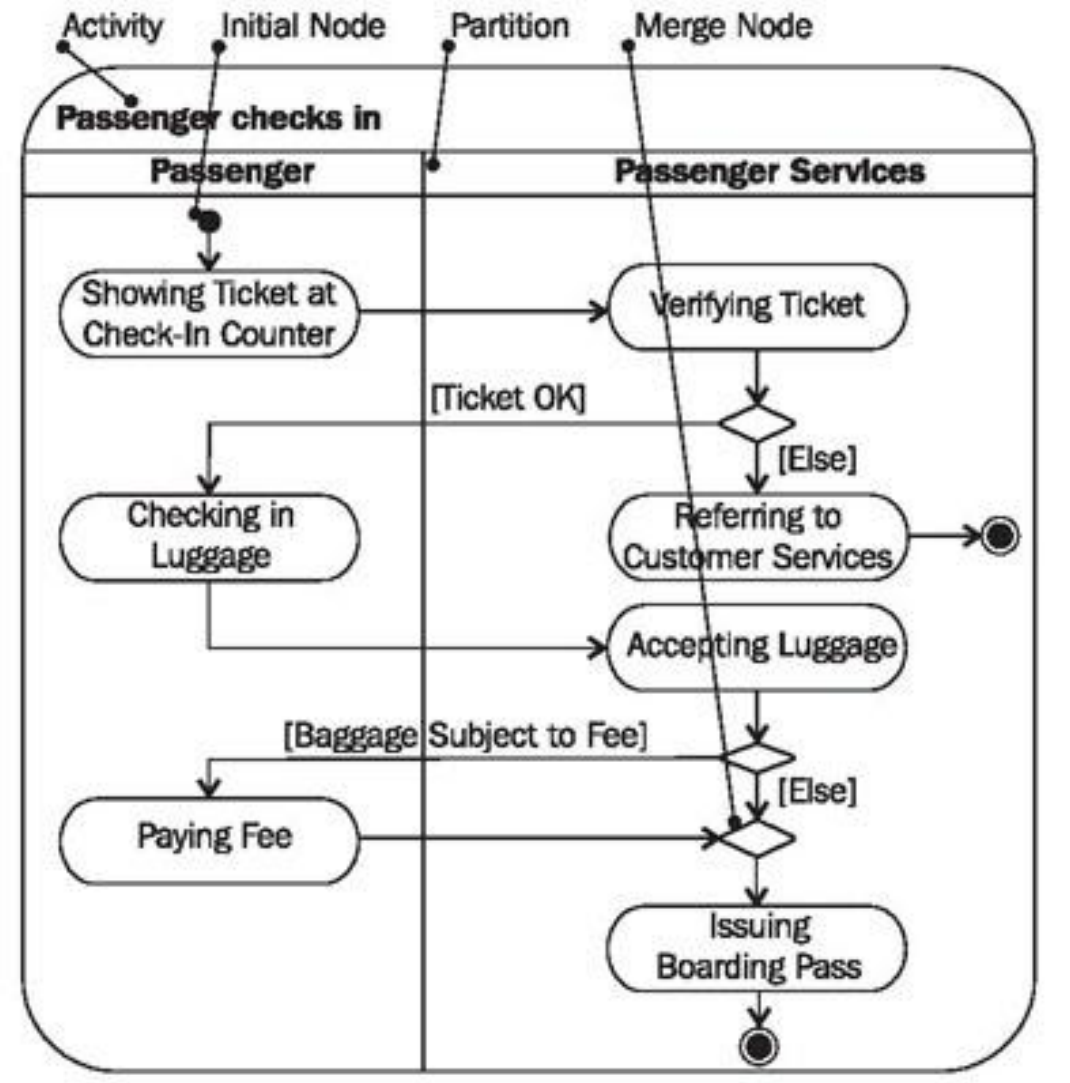


Activity Diagram



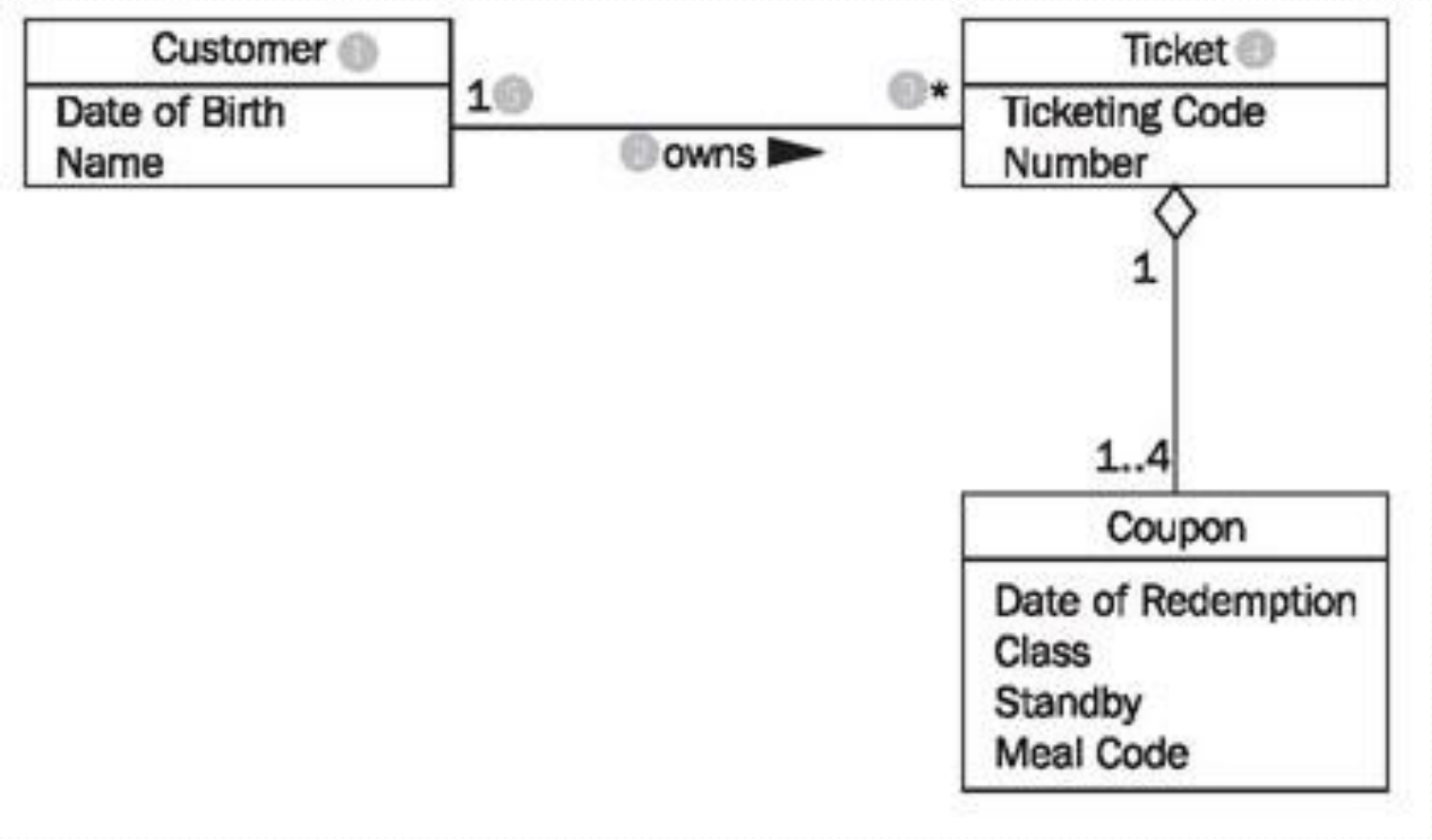
Activity diagram "Passenger Services" with a low level of detail ("High Level")

Activity Diagram

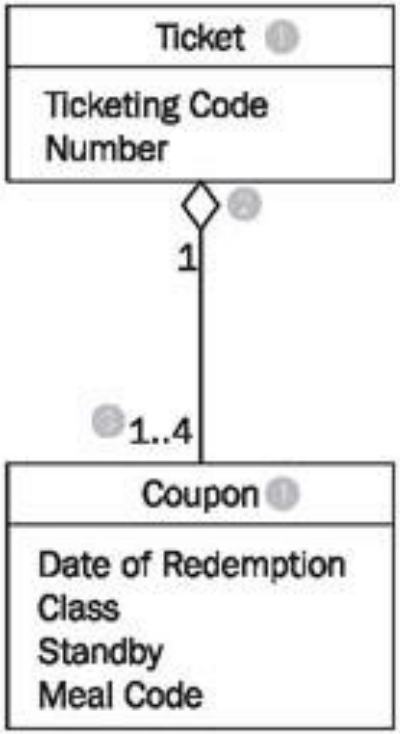


Activity diagram of the activity "Passenger checks in"

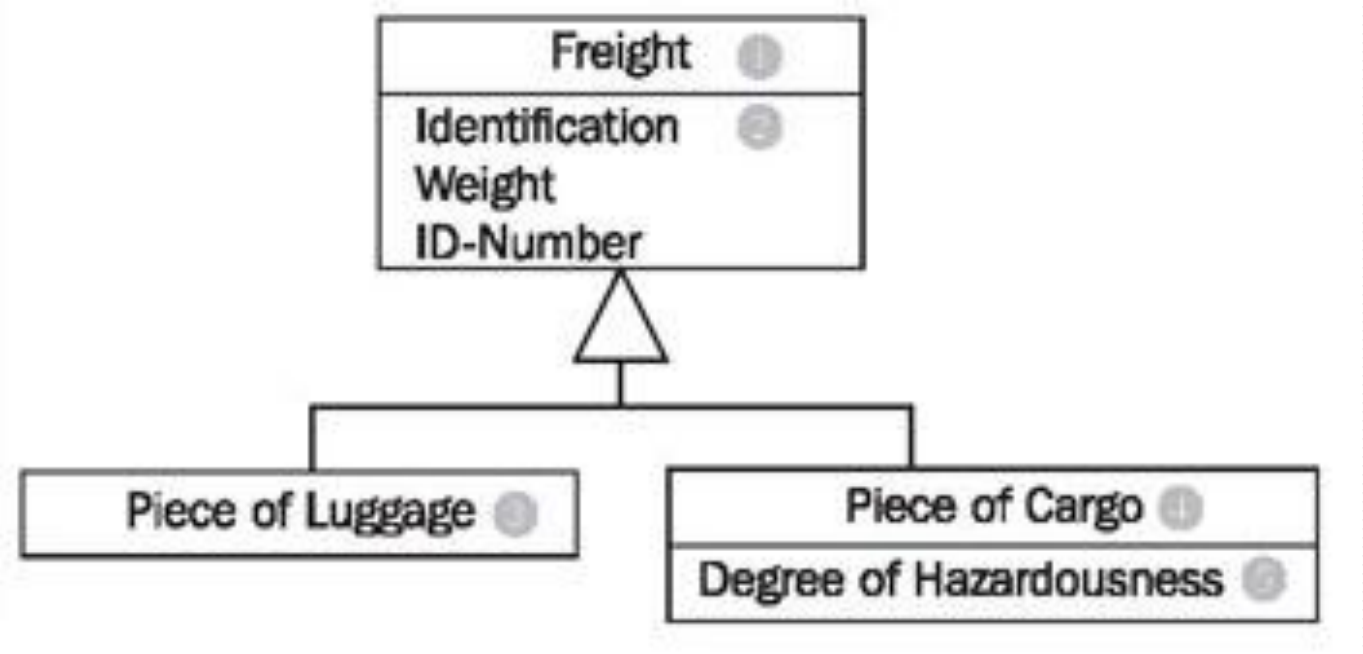
Class diagram: Associations



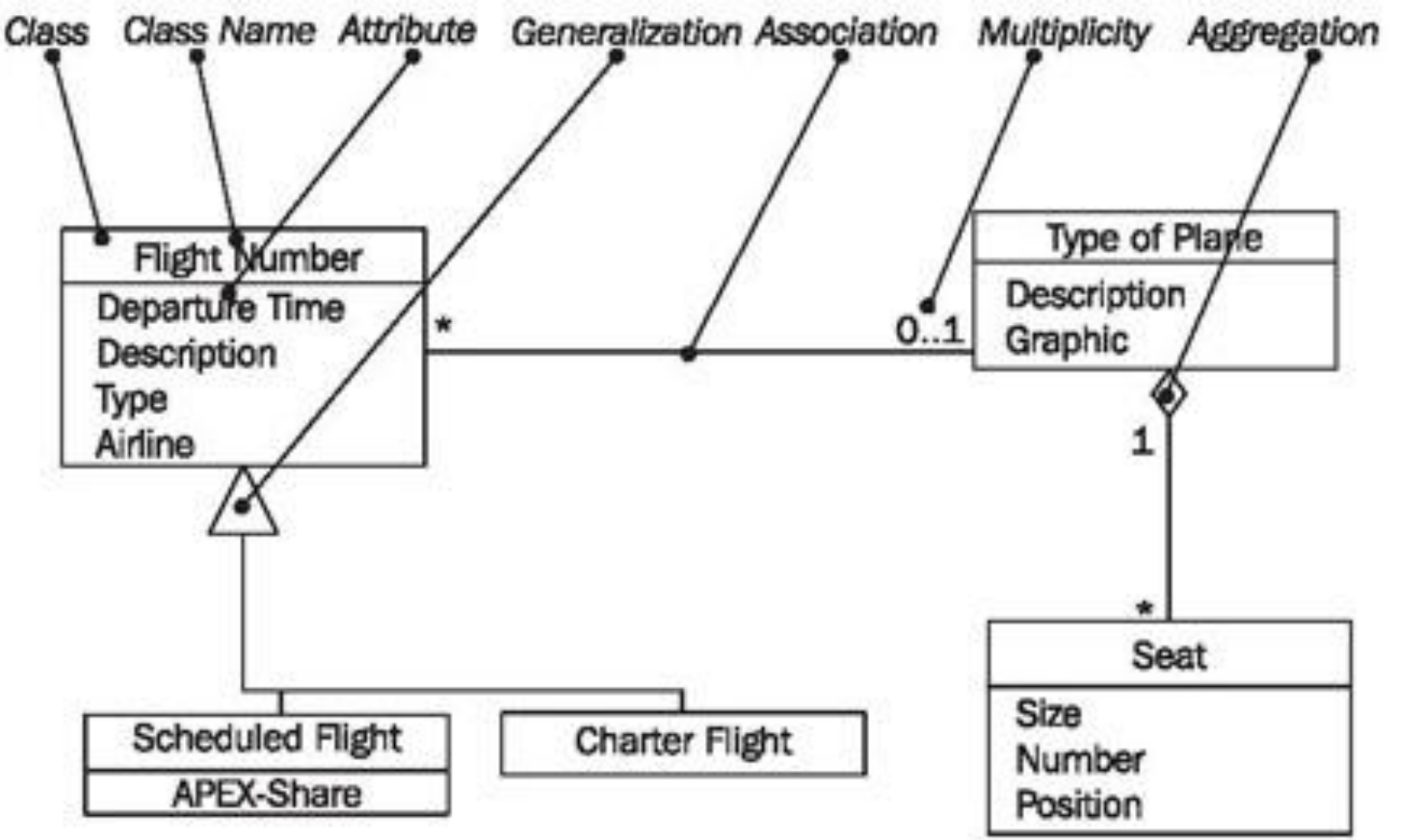
Class diagram : Aggregation



Class diagram: Generalization



Elements of the class diagram



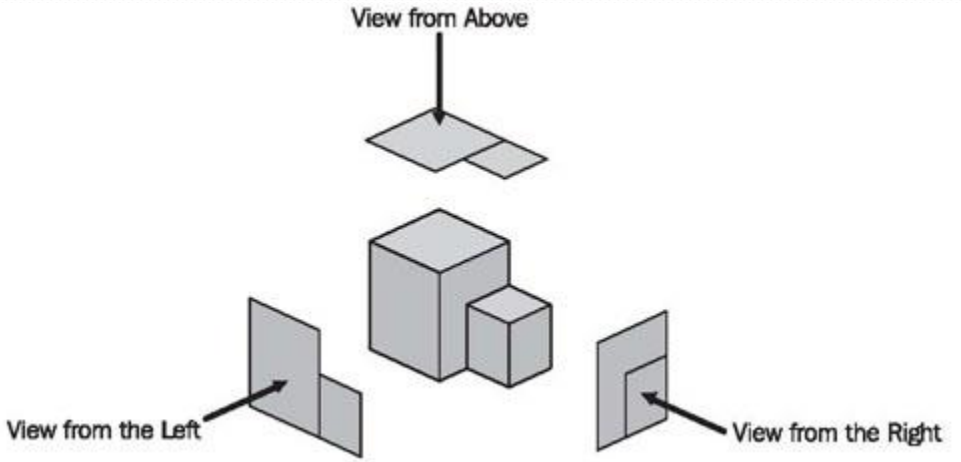
Models, Views, and Diagrams

What is a Model?

- Models are often built in the context of business and IT systems in order to better understand existing or future systems.
- However, a model never fully corresponds to reality.
- Modeling always means *emphasizing* and *omitting*: emphasizing essential details and omitting irrelevant ones.

Views

- The more information a model gives, the more complex and difficult it becomes.
- A map of Europe, for example, that simultaneously contains political, geological, demographic, and transportation-related information is hardly legible. The solution to this problem is to convey the different types of information on individual maps.
- *Different views* are formed of the objects. These views are interconnected in many ways.



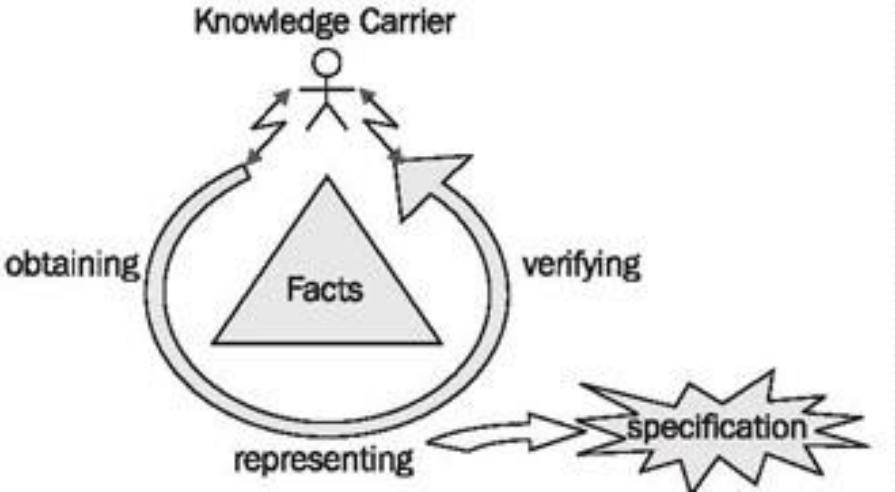
Different views of an object

Why do we Need Models?

- **Communication** between all involved parties:
  - In order to build the right system, all parties think along the same lines. Important that everyone understands the same requirements, that developers understand these requirements, and that the decisions made can still be understood months later.
- **Visualization** of all facts:
  - All accumulated facts relevant to the system need to be presented in such a way that everyone concerned can **understand** them.
- **Verification** of facts in terms of **completeness**, **consistency**, and **correctness**: In particular, the clear depiction of interrelationships makes it possible to ask specific questions, and to answer them.

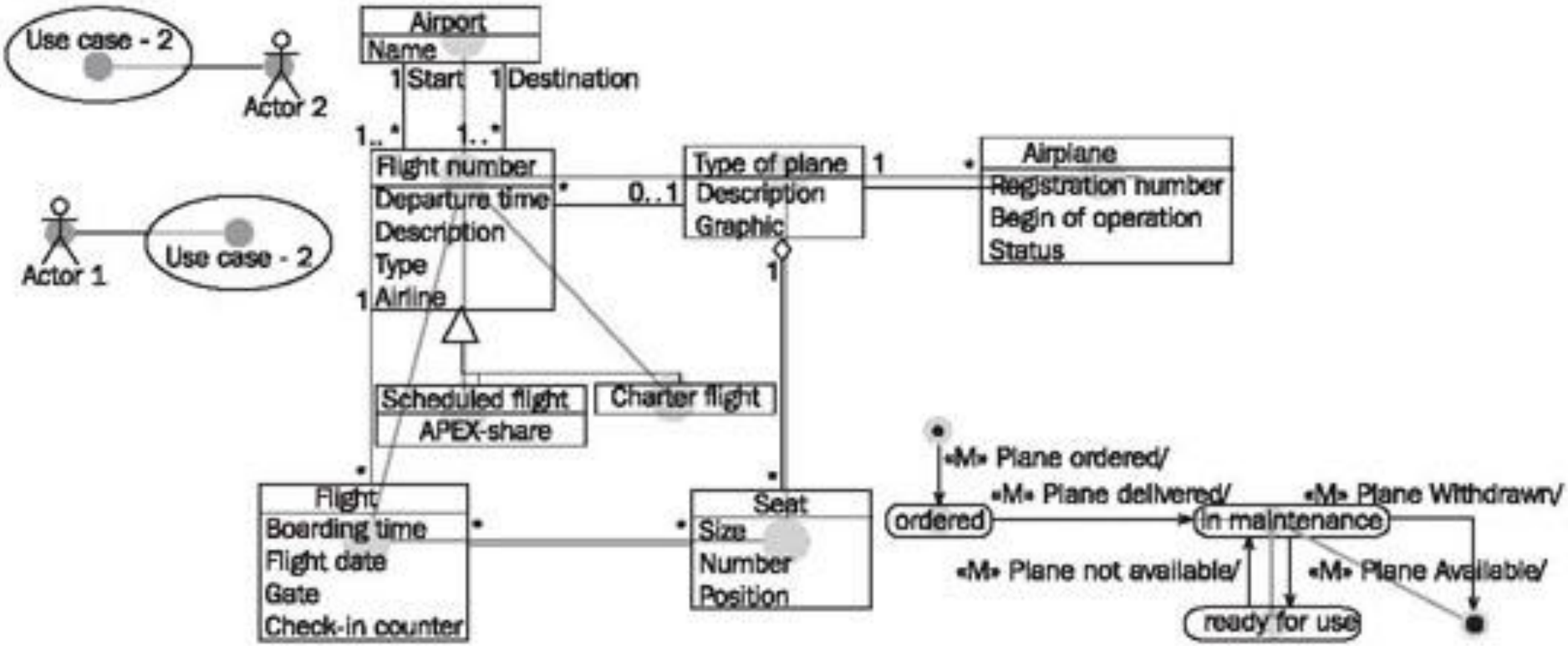
Process of Analysis

- Facts are **obtained** by collaboration between analysts and domain experts in which knowledge carriers contribute domain knowledge and analysts contribute methodological knowledge.
- Facts are **represented** in diagrams and documents, which are usually prepared by the analyst.
- Facts are **verified** only by knowledge carriers, since they alone can decide if the presented facts are correct.



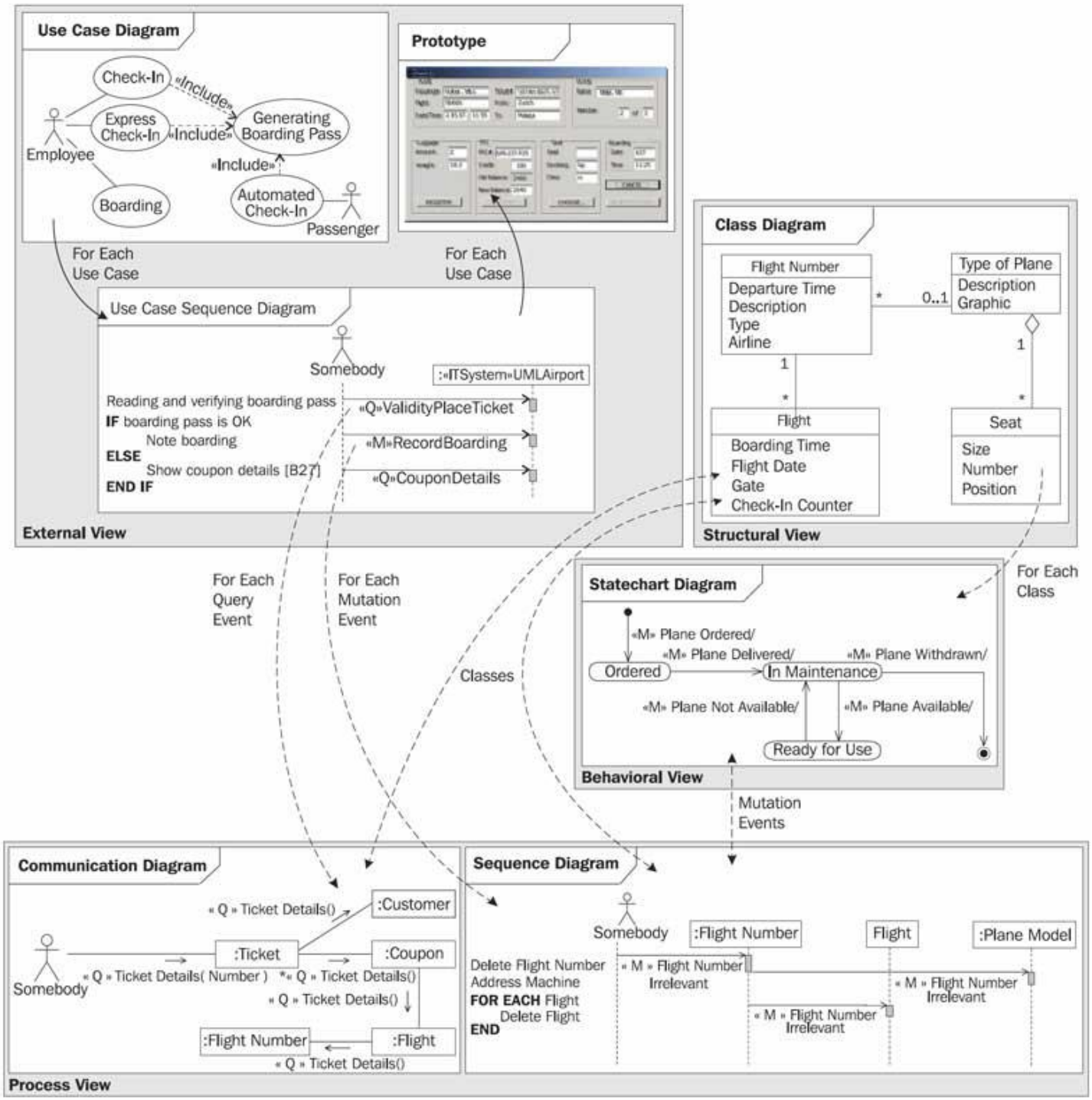
Diagrams as Views

- Each particular UML diagram corresponds to one **view** of a model of a system.



Modeling IT Systems

- a *conceptual model* of an IT system can be developed with the help of UML.
- The IT system model consists of four different views, each of which emphasizes certain aspects and which are closely related to each other.
- **External View**—Use case diagram and use case sequence diagram
- **Structural View**—Class diagram
- **Interaction / Process View**—Sequence diagram and communication diagram
- **Behavioral View**—Statechart diagram

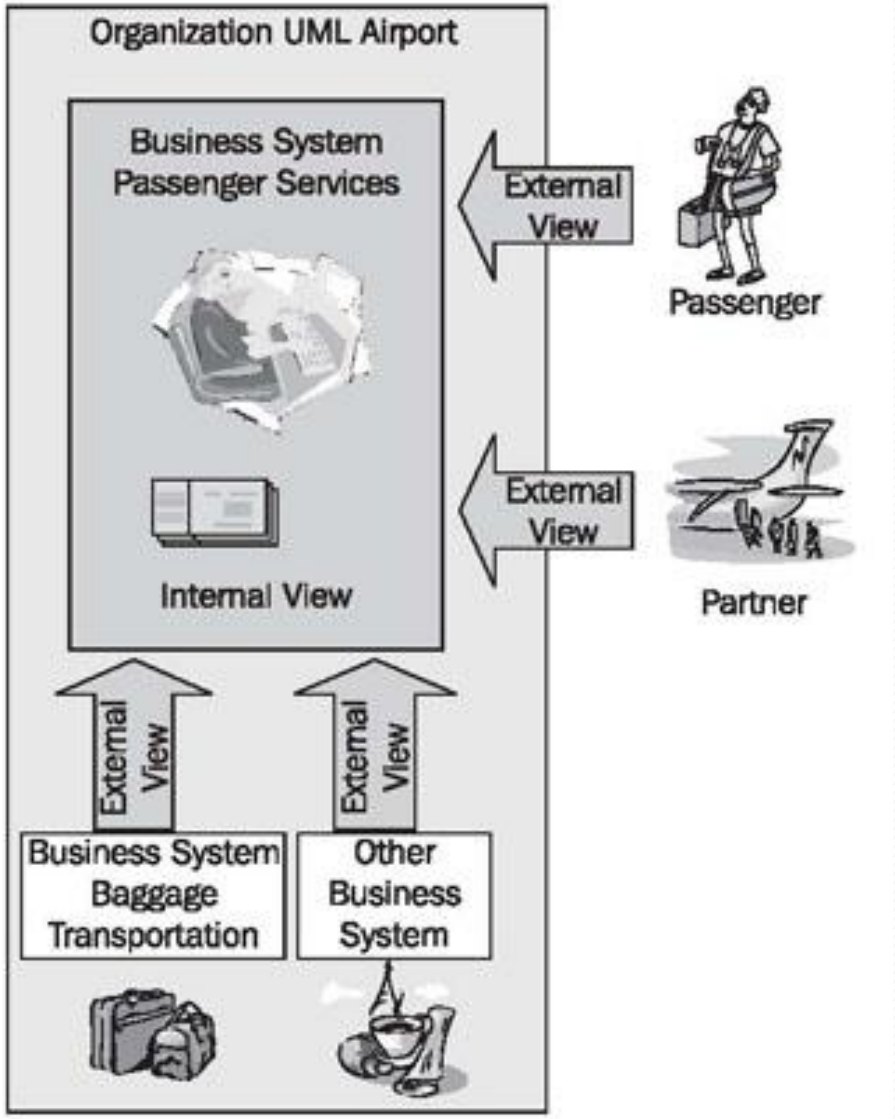

Use Case Diagram
Check-In
«Include»
Express Check-In
«Include»
Generating Boarding Pass
Employee
«Include»
Boarding
Automated Check-In
Passenger
Prototype
For Each Use Case
For Each Use Case
Use Case Sequence Diagram
Somebody
:«ITSystem»UMLAirport
Reading and verifying boarding pass
IF boarding pass is OK
Note boarding
ELSE
Show coupon details [B27]
END IF
«Q»ValidityPlaceTicket
«M»RecordBoarding
«Q»CouponDetails
External View
Class Diagram
Flight Number
Departure Time
Description
Type
Airline
Type of Plane
Description
Graphic
Flight
Boarding Time
Flight Date
Gate
Check-In Counter
Seat
Size
Number
Position
Structural View
For Each Query Event
For Each Mutation Event
Classes
For Each Class
Statechart Diagram
«M» Plane Ordered/
«M» Plane Delivered/
«M» Plane Withdrawn/
Ordered
In Maintenance
«M» Plane Not Available/
«M» Plane Available/
Ready for Use
Behavioral View
Mutation Events
Communication Diagram
« Q » Ticket Details()
:Customer
Somebody
:Ticket
:Coupon
« Q » Ticket Details(Number)
*« Q » Ticket Details()
« Q » Ticket Details()
:Flight Number
:Flight
« Q » Ticket Details()
Sequence Diagram
Somebody
:Flight Number
Flight
:Plane Model
Delete Flight Number
Address Machine
FOR EACH Flight
Delete Flight
END
« M » Flight Number
Irrelevant
« M » Flight Number
Irrelevant
« M » Flight Number
Irrelevant
Process View

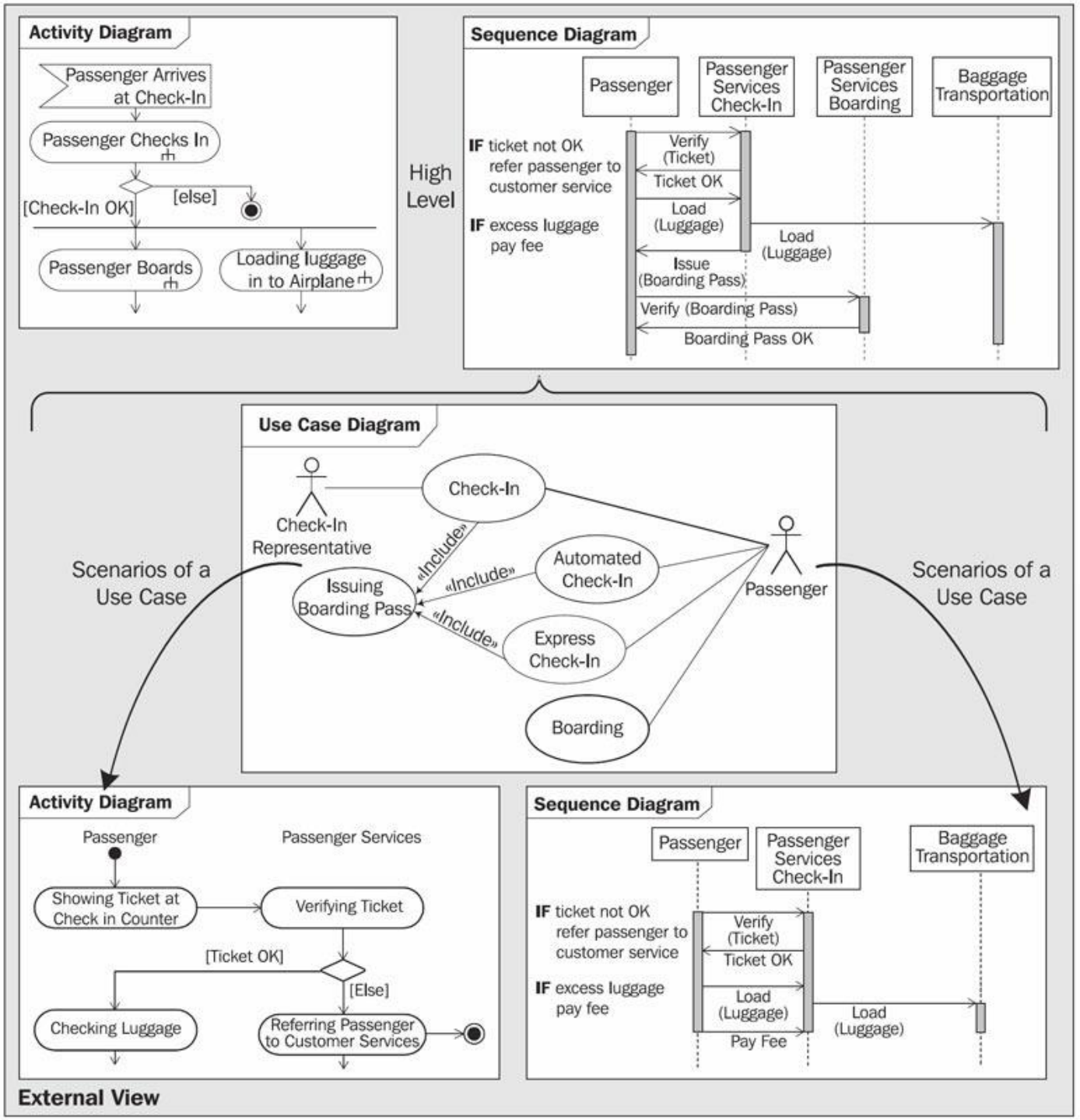
One Model—Two Views

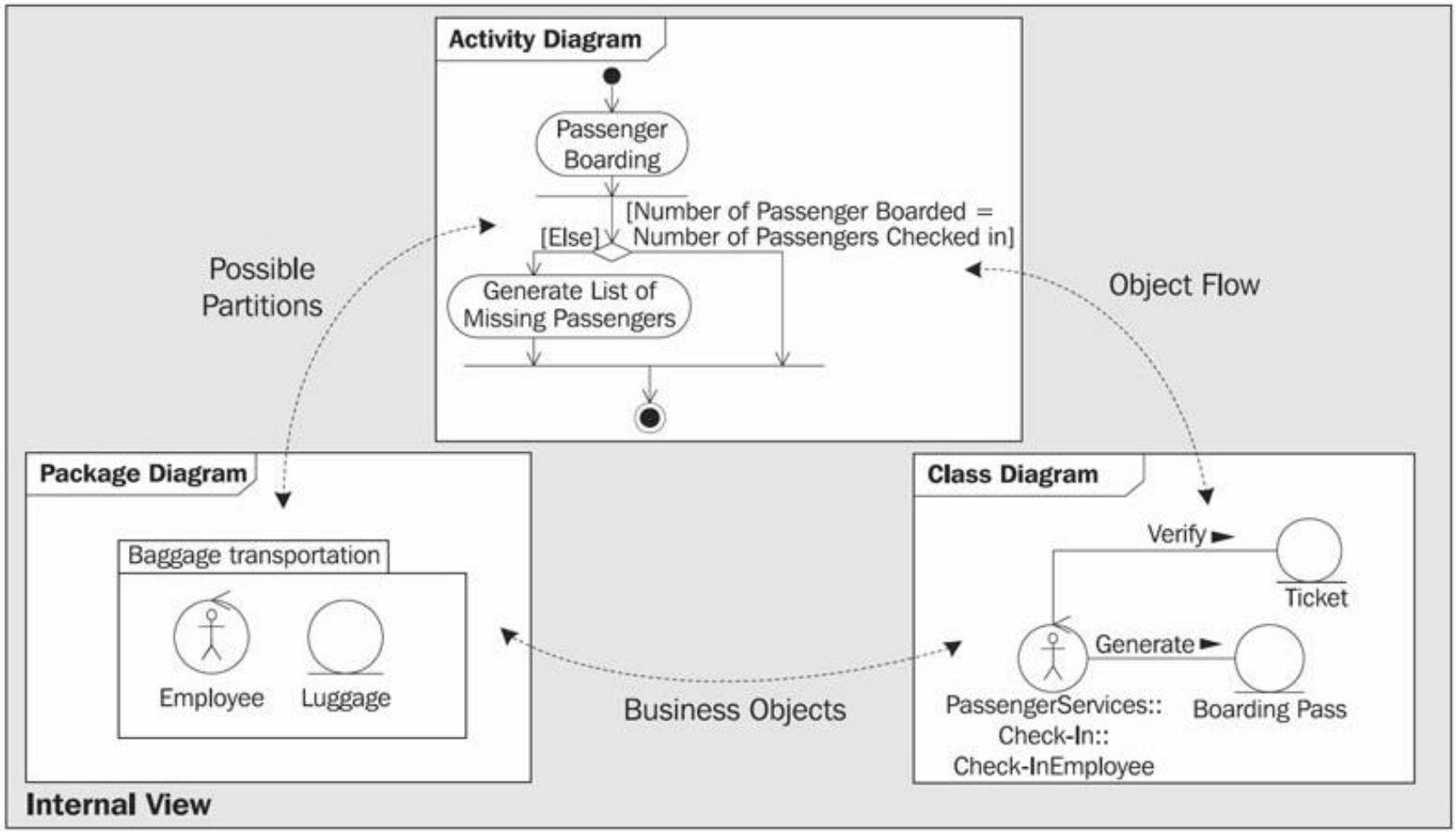
- Viewing a business system from the outside, we take on the role of a customer, a business partner, a supplier, or another business system. From this **external view**, only those business processes that involve outsiders are of interest. The external view describes the environment of a business system. The business system itself remains a black box.
- Within the business system, we find *employees* and *tools* that are responsible for fulfilling the demands of the environment, and for handling the necessary business processes. Behind the business processes are *workflows* and *IT systems*. Each individual employee is part of the *organizational structure*. Normally this **internal view** remains hidden to outsiders.

External and internal view

External and internal view of the business system




Activity Diagram
Passenger Arrives at Check-In
Passenger Checks In
[Check-In OK]
[else]
Passenger Boards
Loading luggage in to Airplane
High Level
Sequence Diagram
Passenger
Passenger Services Check-In
Passenger Services Boarding
Baggage Transportation
IF ticket not OK refer passenger to customer service
IF excess luggage pay fee
Verify (Ticket)
Ticket OK
Load (Luggage)
Load (Luggage)
Issue (Boarding Pass)
Verify (Boarding Pass)
Boarding Pass OK
Use Case Diagram
Check-In Representative
Check-In
«Include»
Issuing Boarding Pass
«Include»
Automated Check-In
«Include»
Express Check-In
Boarding
Passenger
Scenarios of a Use Case
Scenarios of a Use Case
Activity Diagram
Passenger
Passenger Services
Showing Ticket at Check in Counter
Verifying Ticket
[Ticket OK]
[Else]
Checking Luggage
Referring Passenger to Customer Services
Sequence Diagram
Passenger
Passenger Services Check-In
Baggage Transportation
IF ticket not OK refer passenger to customer service
IF excess luggage pay fee
Verify (Ticket)
Ticket OK
Load (Luggage)
Load (Luggage)
Pay Fee
External View

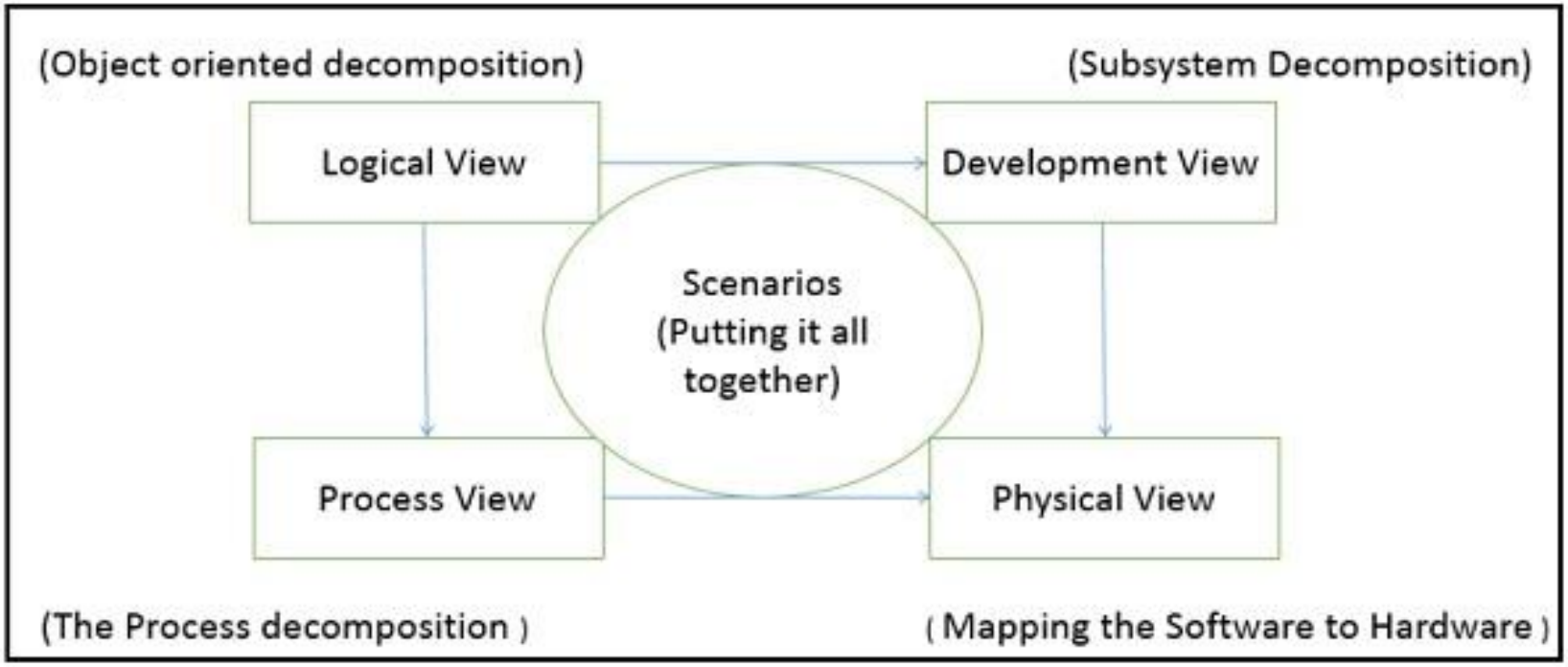

Activity Diagram
Passenger Boarding
[Number of Passenger Boarded = Number of Passengers Checked in]
[Else]
Generate List of Missing Passengers
Possible Partitions
Object Flow
Package Diagram
Baggage transportation
Employee
Luggage
Class Diagram
Verify
Ticket
Generate
Boarding Pass
PassengerServices::
Check-In::
Check-InEmployee
Business Objects
Internal View

4+1 View Model

- It provides 4 essential views:
  - **Logical / conceptual view:** describes object model of design.
  - **Process view**: describes activities of system, captures concurrency & synchronization aspects of the design.
  - **Physical view:** Describes the mapping of software onto hardware and reflects its distributed aspect.
  - **Development view:** It describes static organization / structure of software in its development environment.
- This view model can be extended by adding one more view called **scenario view** or **use case view** for end-users or customers of software systems.

4+1 View Model

- The following figure describes the software architecture using five concurrent views (4+1) model.



https://www.tutorialspoint.com/software_architecture_design/architecture_models.htm