Requirement Elicitation
Exercise 1

A. The development must follow the MIL-STD-498 methodology
B. The system must list the hotels by price
C. The system must list the restaurants by category
D. The system must be designed to be portable
E. The system must handle 1000 users concurrently
F. The requirements must be specified using “use cases”
G. The communications must be secured

- Are they a functional requirement, a constraint, or a quality?
- Which of them must it be absolutely re-expressed?

Requirement Elicitation
Exercise 2

The university wants a new system
to manage students' marks.

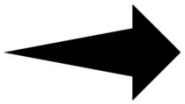
1. Who are the stakeholders for this project?
2. Identify possible conflicts

Requirement Elicitation
Exercise 3

Propose a reformulation for
the following requirements

- The reservation system must exhibit high performances
- The reservation system must be easy to use
- The reservation system must be available
- The reservation system must be secure enough



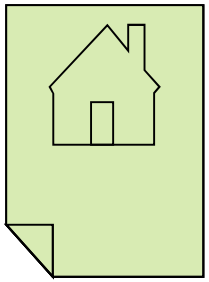


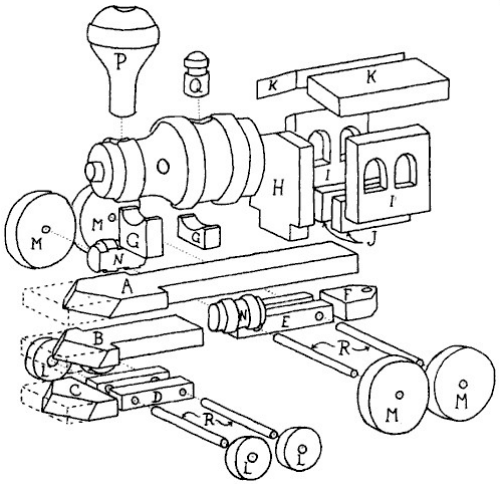


Requirement ellicitation

How to describe this specification?

Need of representations to discuss, organize, build, document…

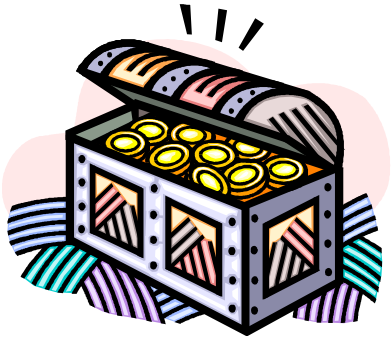












In SE, models are used:

- As a starting point to
  - abstract and to understand
  - support the discussion
  - organize, plan
- To design and detail
- As support at the end of development
  - To test
  - To document
  - To maintain













UML language - 1

Lydie du Bousquet

Lydie.du-bousquet@univ-Grenoble-alpes.fr

En collaboration avec J.-M. Favre, I. Parissis, Ph. Lalanda

UML = Unified Modeling Language

- A **language**
- For **modeling**
  - at the analysis and the design stages (Object-oriented)
- **Unified**
  - To cover as many **domains** as possible
  - To cover as many **notions** as possible

- Objective: different analysts can
  - Have a common language to discuss
  - Common tools

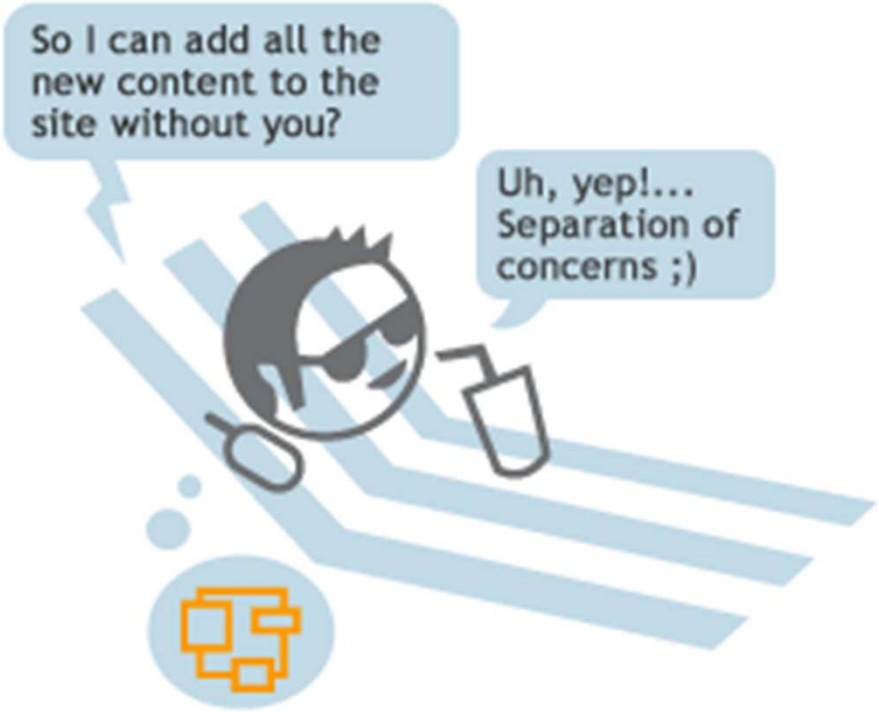


UML = standard

- International standard
  - Very large (many notions)
- More and more used in the industry
- Associated to several
  - methods
  - Tools
- Can be used with different level of details

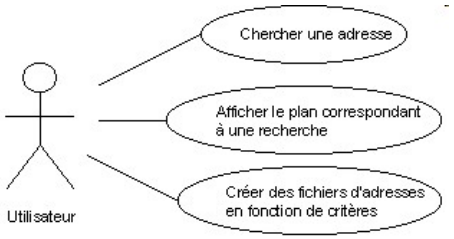
UML: a language, several views

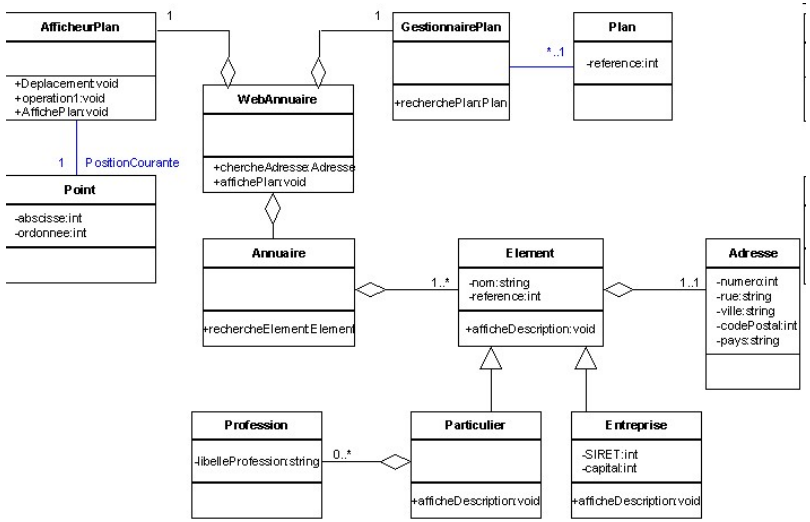
- Different needs
  - To model static or dynamical point of views
  - At different stage analysis, specification, design, …
- Using views
  - Separation of concerns

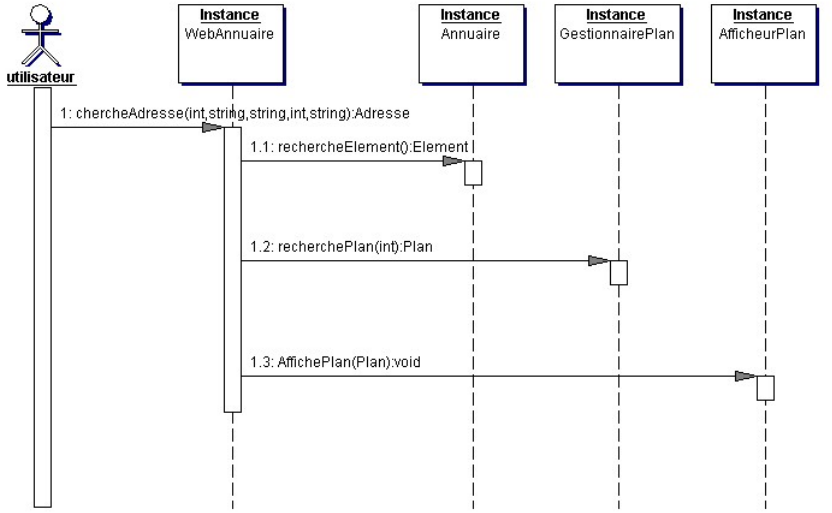


UML: a language, several views

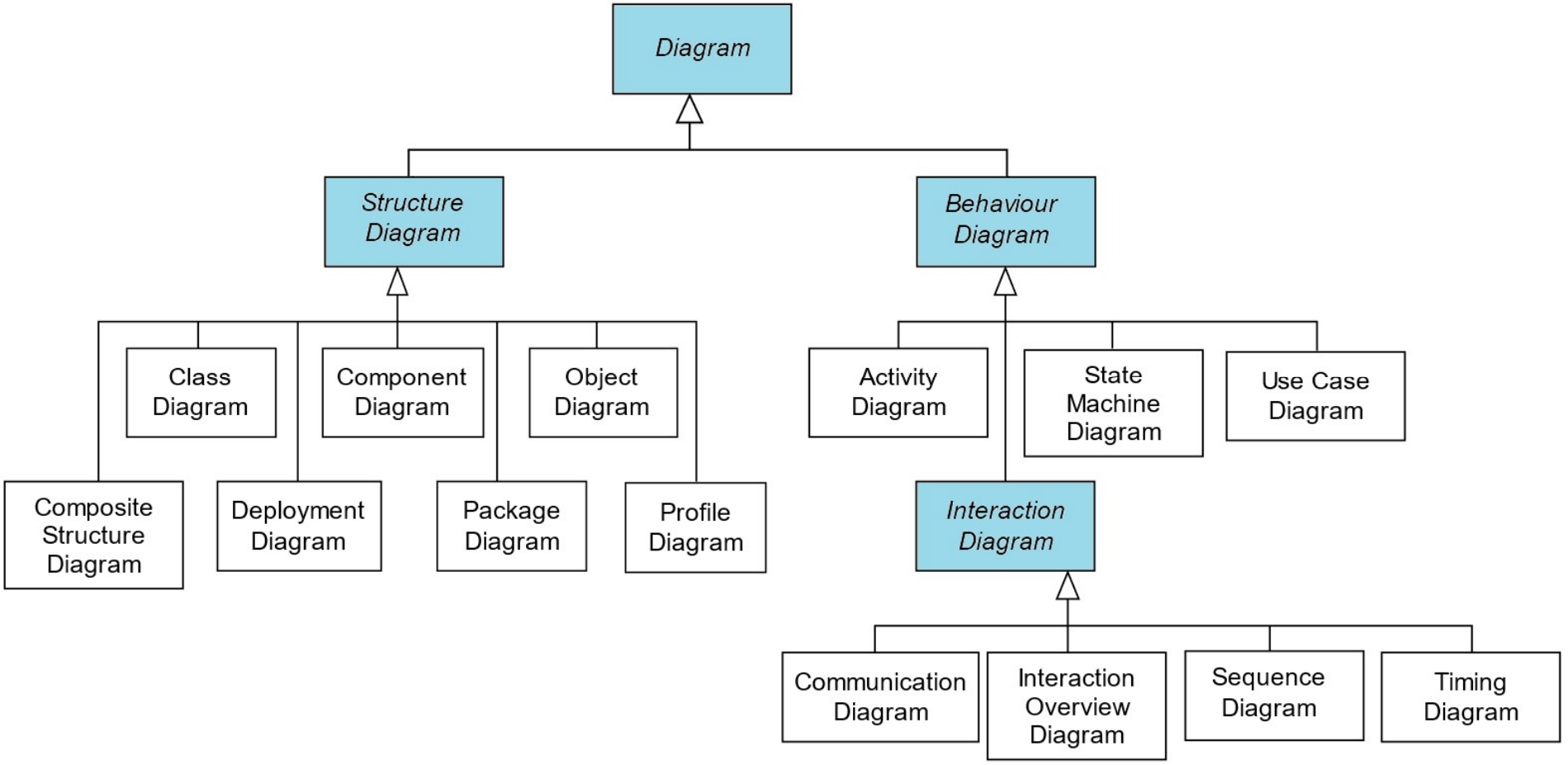








UML 2.2


Diagram
Structure Diagram
Behaviour Diagram
Class Diagram
Component Diagram
Object Diagram
Activity Diagram
State Machine Diagram
Use Case Diagram
Composite Structure Diagram
Deployment Diagram
Package Diagram
Profile Diagram
Interaction Diagram
Communication Diagram
Interaction Overview Diagram
Sequence Diagram
Timing Diagram

14 diagrams in UML 2.2

Structure diagrams

- **Class diagram**
- **Object diagram**
- Component diagram
- Composite structure diagram
- **Deployment diagram**
- Package diagram
- Profile diagram

Behaviour diagrams

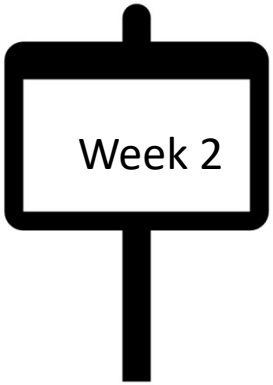
- **Use case diagram**
- **State Machine diagram**
- Activity diagram

Interaction diagrams

- **Sequence diagram**
- Communication diagram
- Interaction overview diagram
- Timing diagram

Functional requirements – Davis, 93

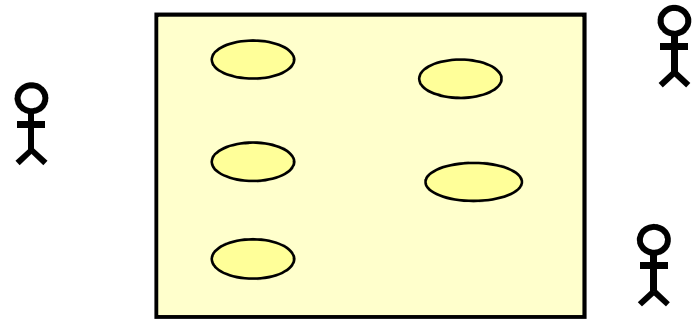
- A **function**
  - **Activity** clearly defined in the domain
- An **object**
  - **Entity** clearly identified
- A **state**
  - Characterize the situation of an entity



Use-case diagram

- Documents the **system**'s intended behavior
- Representation of the **relationships** between **actors** and **use-cases**
- Arrows and lines are drawn
  - between actors and use cases (by default «communicates»)
  - between use cases to show their relationships
  - Between actors to show their relationships

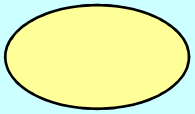
Use-case diagram



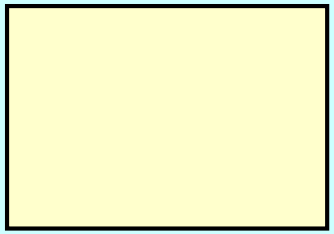
Actor



Use case

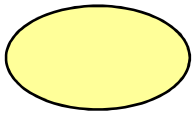


System



Use cases

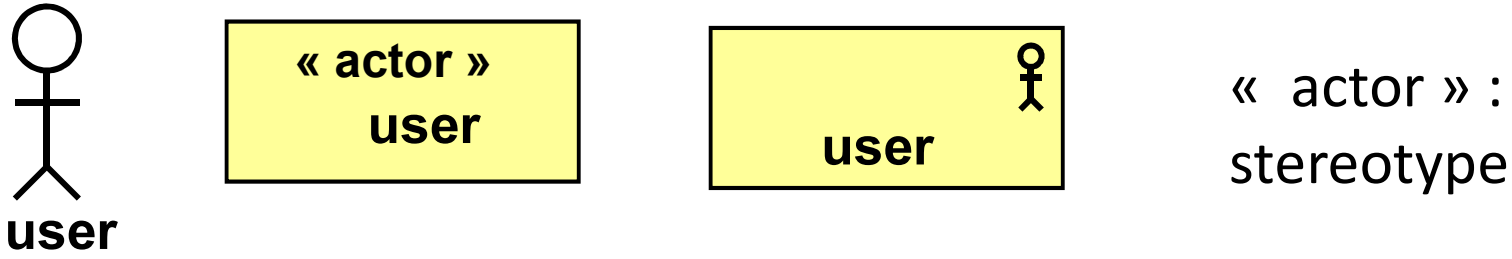
- They are drawn as an **ellipse**



- Names are normally made up of
  - an active verb and a noun
  - noun phrase
- Names should be **representative** of the behavior

Actors

- People or systems that interact with use cases
- They represents **roles**
  (not John or Mary, but assistant or supervisor)



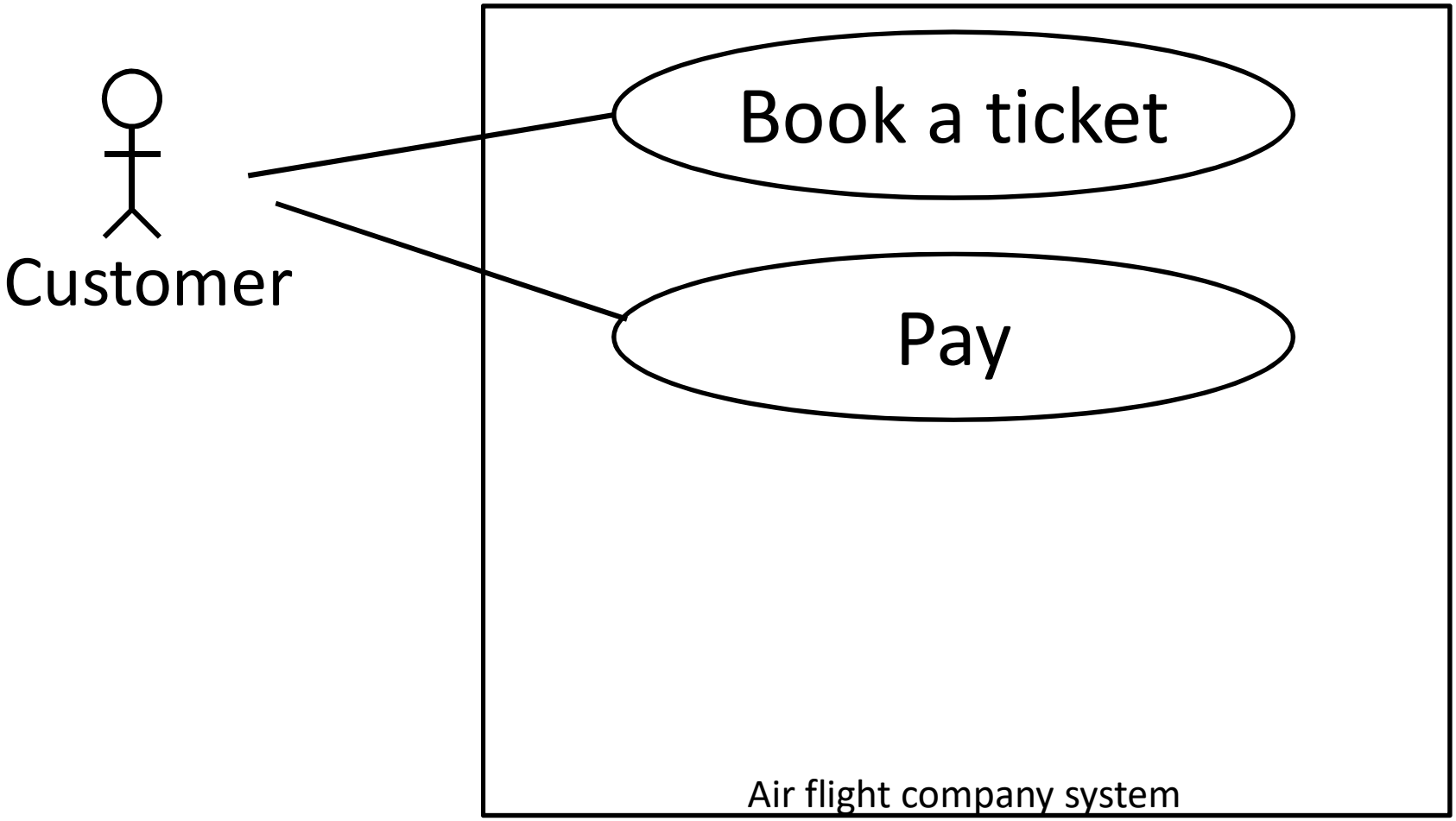
- They are **connected** with use cases with which they interact
- The **name** of the actors should be chosen carefully
- **Generalization** can be used between 2 actors

Relationships

- Association between actor and use-case
- Generalization between two actors
- Association between two use-cases
  - Include
  - Extend
- Generalization between two use-cases

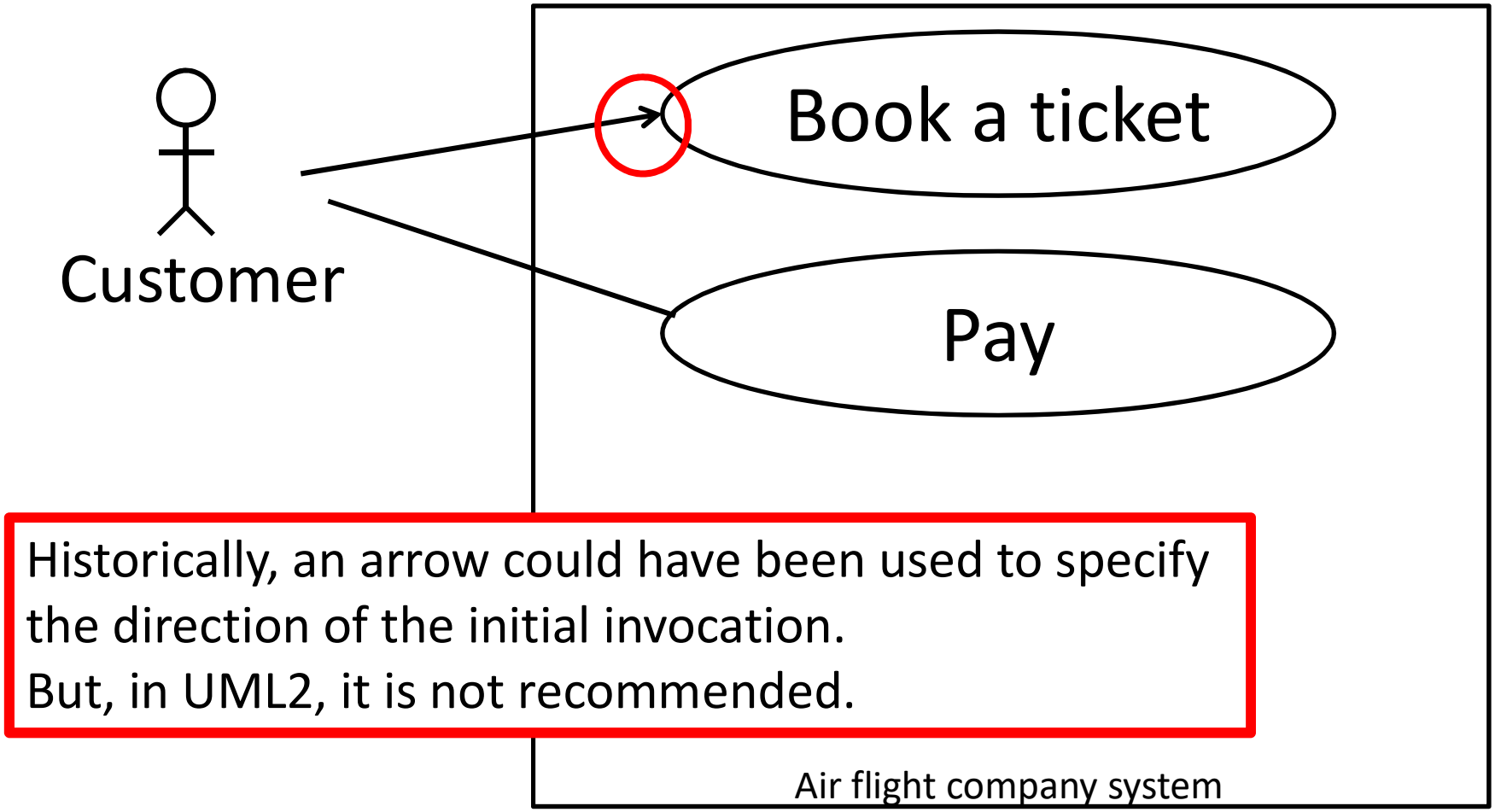
Association between actor and use-case

Example: Air flight company system



Association between actor and use-case

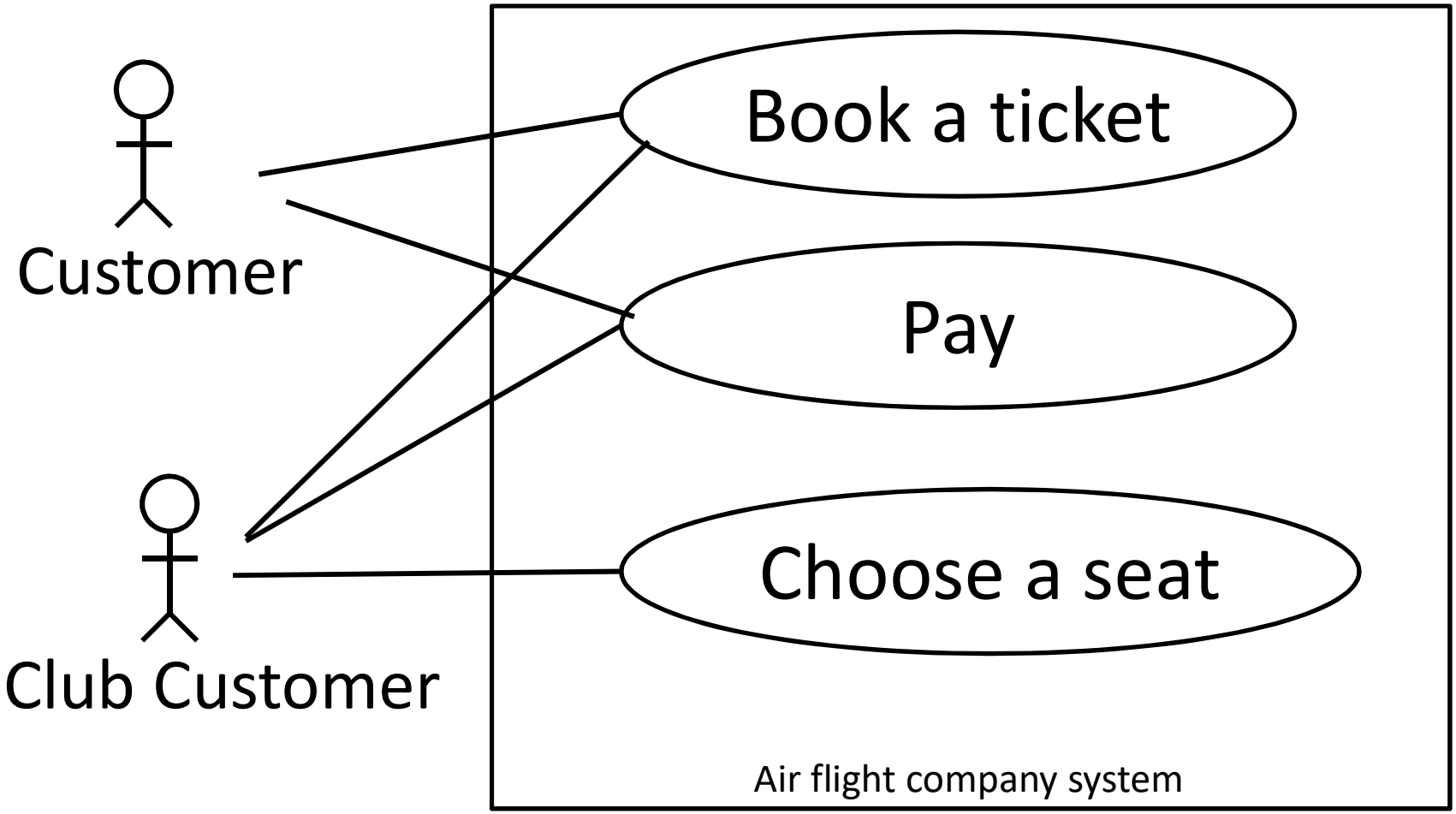
Example: Air flight company system



Historically, an arrow could have been used to specify the direction of the initial invocation.
But, in UML2, it is not recommended.

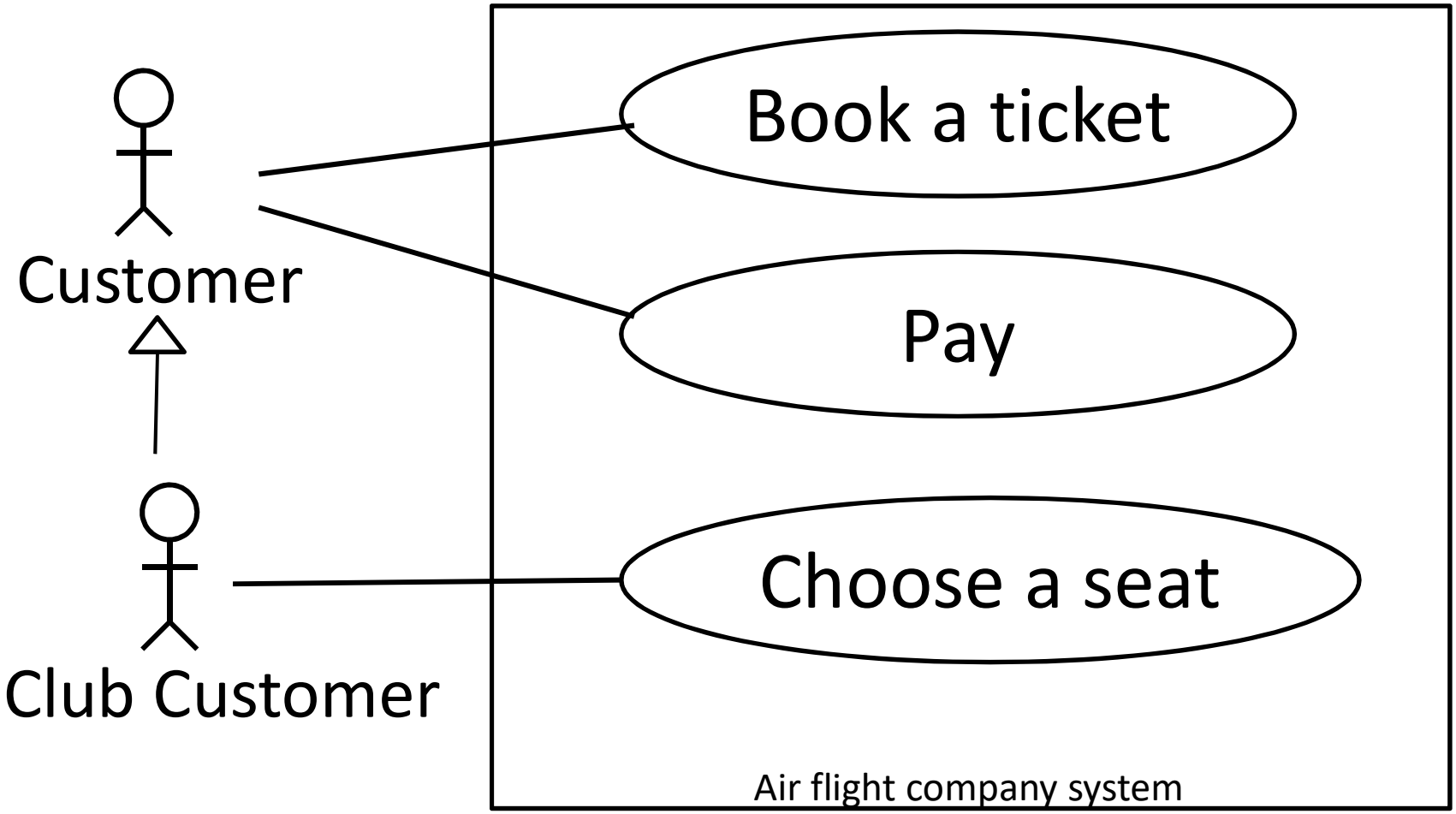
Generalization between two actors

Example: Air flight company system



Generalization between two actors

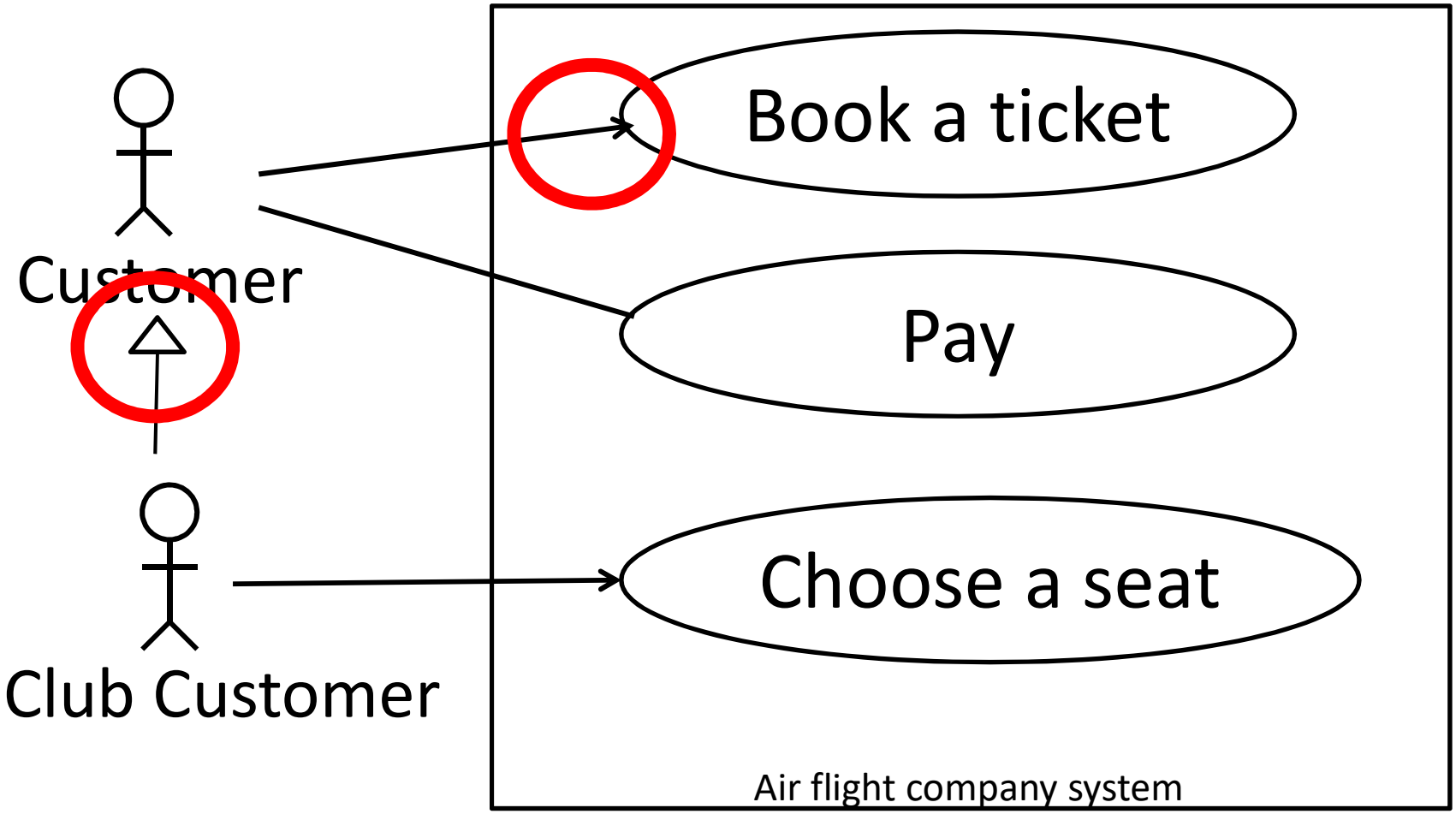
Example: Air flight company system



A club customer is a special customer: generalization relation.

BE CAREFUL with the type of the arrows!

Example: Air flight company system

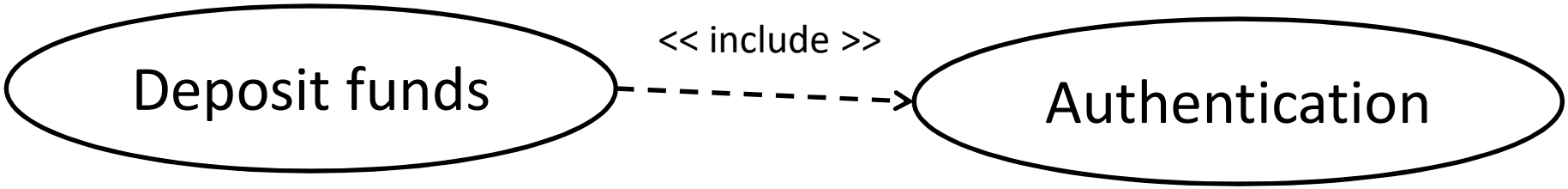


Association between 2 use-cases

Include and extend relationships

<< include >>

- encapsulates a functionality used at several point in the systems
- to avoid repetition



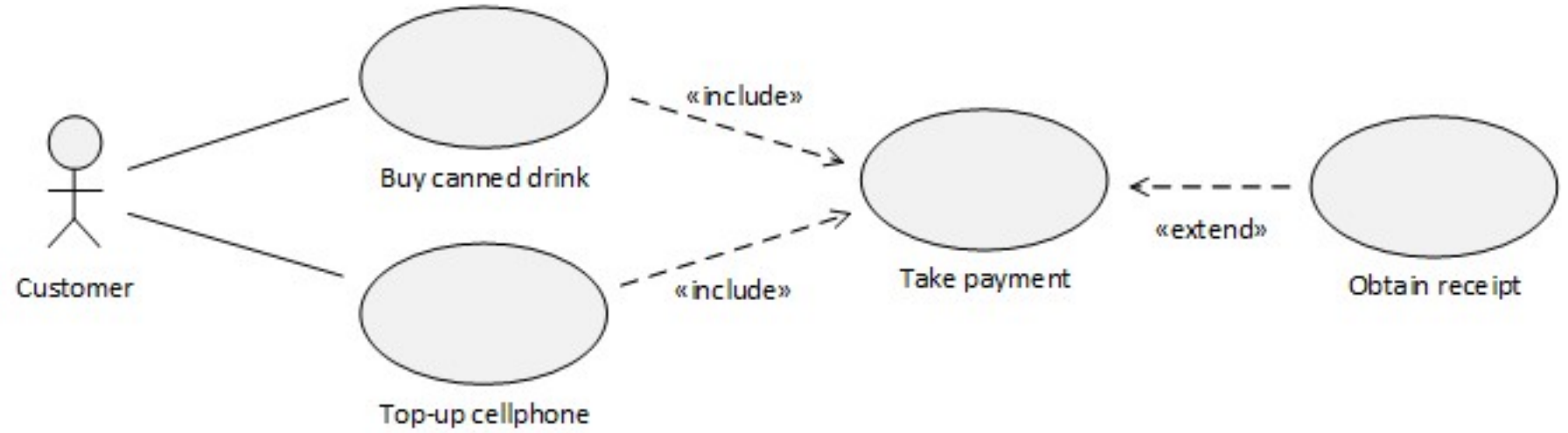
To deposit funds, it is necessary to achieve authentication

Association between 2 use-cases

Include and **extend** relationships

<< extend >>

- denotes the fact that the use case may be optionally extended

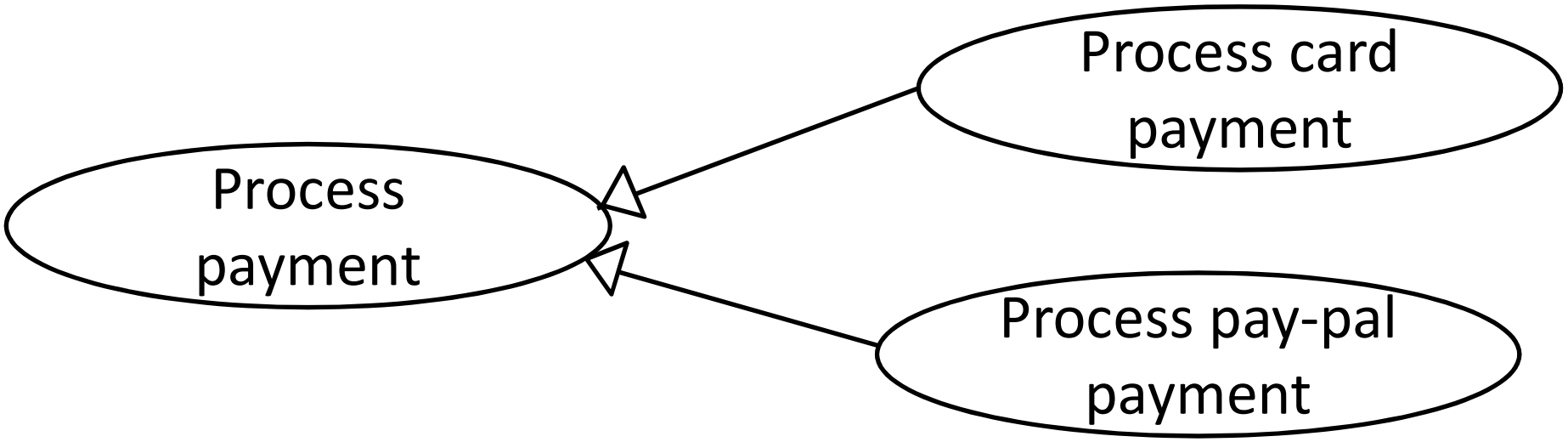


http://www.ropley.com/use-case-extend-relationships.aspx

Association between 2 use-cases

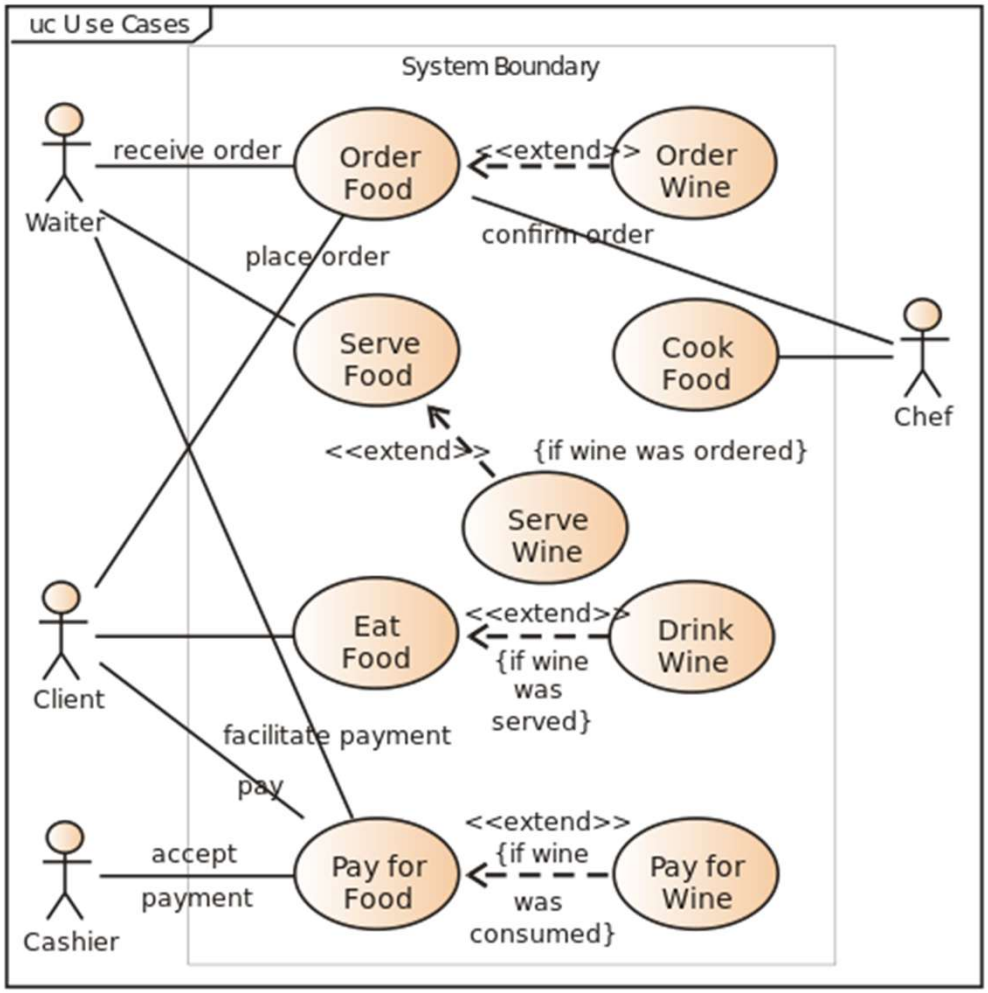
Generalization between two use cases

- A parent use case may be specialized into one or more child use cases that represent more specific forms of the parent.



Express in UML the list of Cyber-video functions as a use-case diagram

Use-case diagram example



use case diagram for the interaction of a client (actor) within a restaurant (system)
https://en.wikipedia.org/wiki/Use_Case_Diagram

Behavior specification

- Use case **diagram** vs use case **model**
- A use case represents a **sequence of activities** that results in some observable outcome
- The activities have to be **documented**
  - Simple paragraph
  - Two-column presentation (actor and system)
  - UML sequence diagram, communication diagram, …

Unified Process

(1) Define the use case **model**

   (1) Introduce the system (small description)
   (2) Find the actors
   (3) Give a small description of each actor
   (4) Find use-cases, express the relations
   (5) Describe the system as a whole

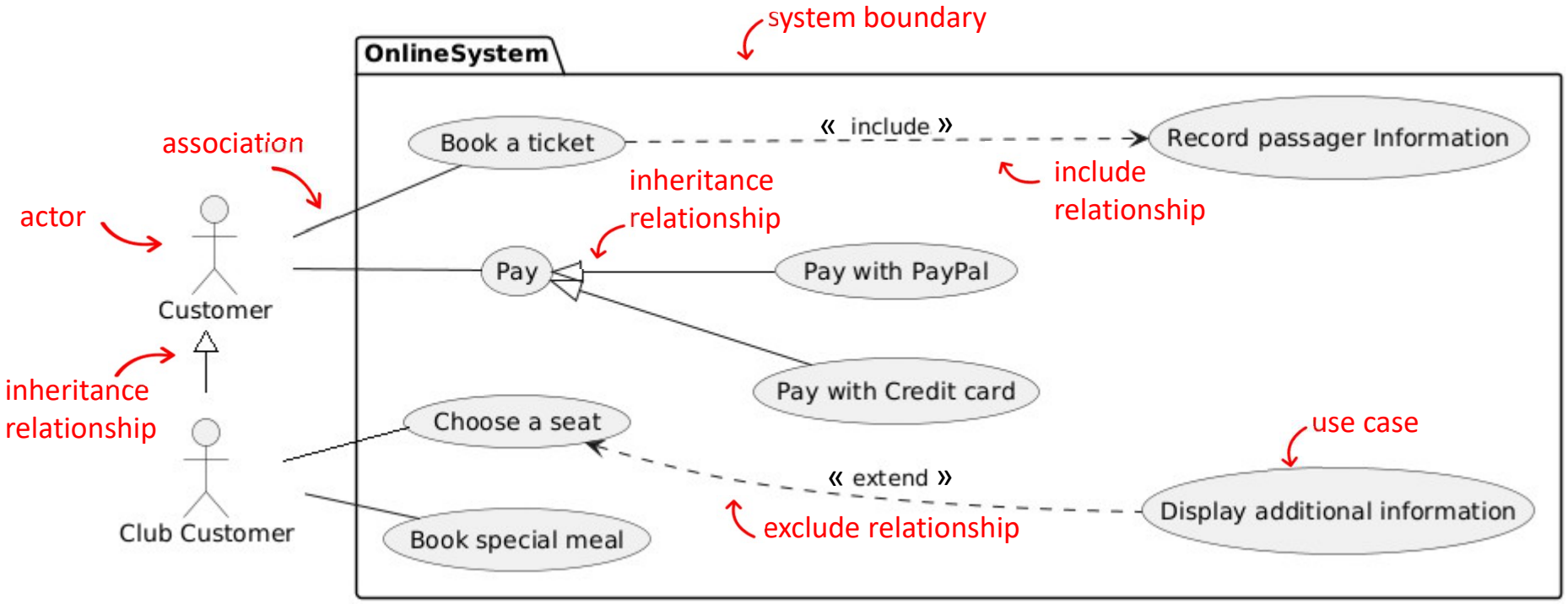
(2) Define the priority among the use cases

(3) Detail the use cases (w.r.t. the priorities)

Advices

- Use **strong** verb and **domain** vocabulary for use cases:
  - *withdraw is stronger than perform withdrow*
  - *Convey pakage vs deliver shipment*
- Name actor with domain-relevant name and place primary actor on the top left
- Use an actor « time » to initiate scheduled event (can be represented as ⌛)

Use Case diagram



BE CAREFUL with the type of the arrows!

Exercises

3.1

3.2

14 diagrams in UML 2.2

Structure diagrams

- **Class diagram**
- **Object diagram**
- Component diagram
- Composite structure diagram
- **Deployment diagram**
- Package diagram
- Profile diagram

Behaviour diagrams

- Use case diagram
- **State Machine diagram**
- Activity diagram

Interaction diagrams

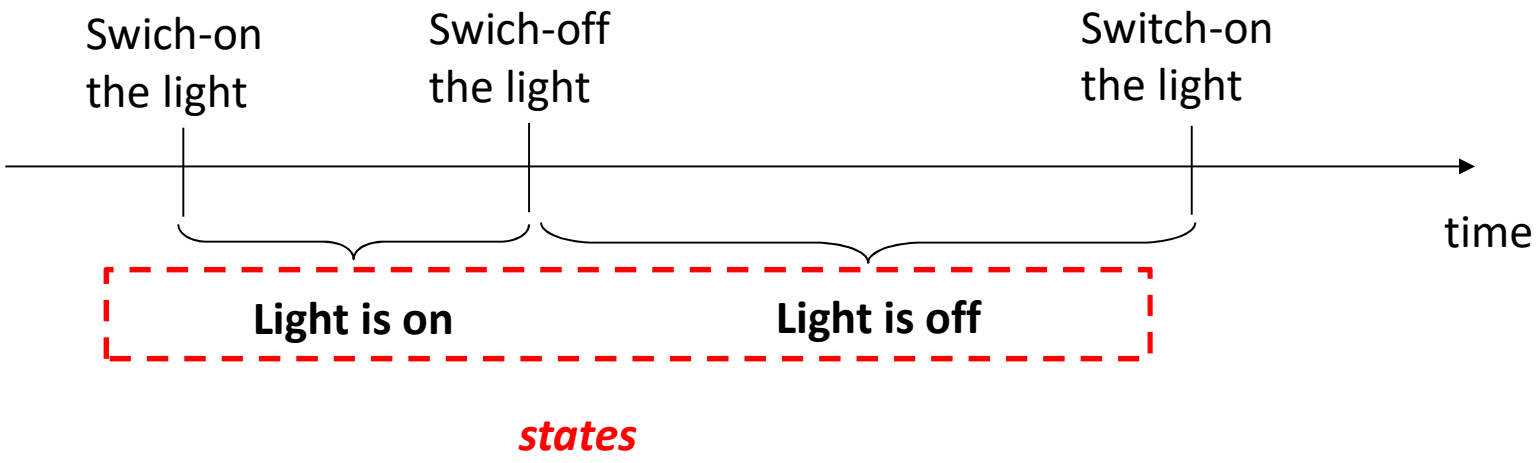
- **Sequence diagram**
- Communication diagram
- Interaction overview diagram
- Timing diagram

UML State Machine diagram (statechart)

- Focused on a specific object
- Describes the flow between states
- Mostly used to describe the states of a class
- Can be used during the analysis, design, …
- Main elements of the diagrams
  - States
  - Events : they trigger the state change
  - Transition : links two states, represented as arrow labeled with event

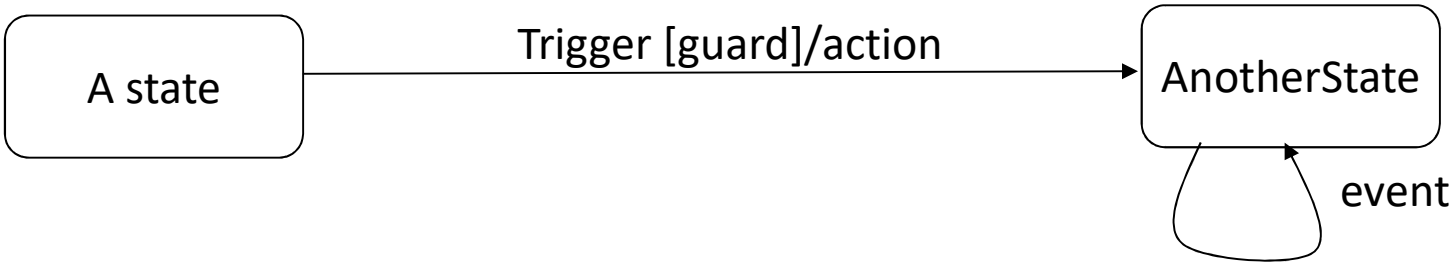
States in UML State machine diagram (statechart)

- It is a moment in the life cycle
- It corresponds to the presence or an absence of activity



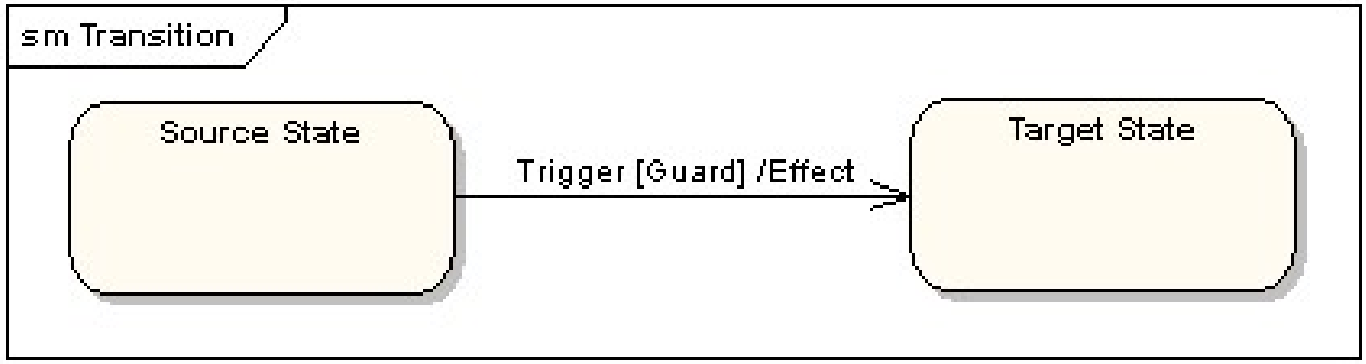
Transition

- Movement from one state to another one
- Drawn with an **arrowed** line
- A transition is always in response to an event (trigger)
- Some condition and action can be handle by transition



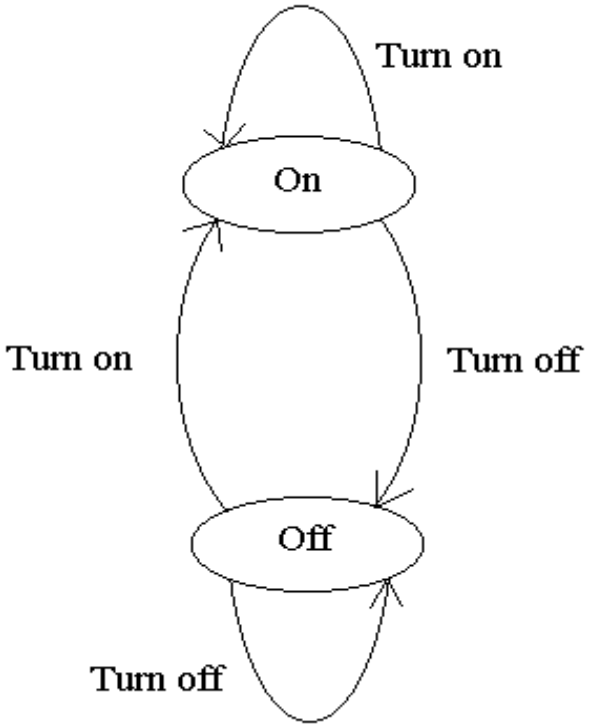
Transitions : Triggers [Guard] / Effect

- **Trigger** = cause of the transition
- **Guard** = condition which must be true in order for the trigger to cause the transition
- **Effect** = action which will be invoked directly on the object that owns the state machine as a result of the transition

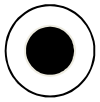


Example 1: a light

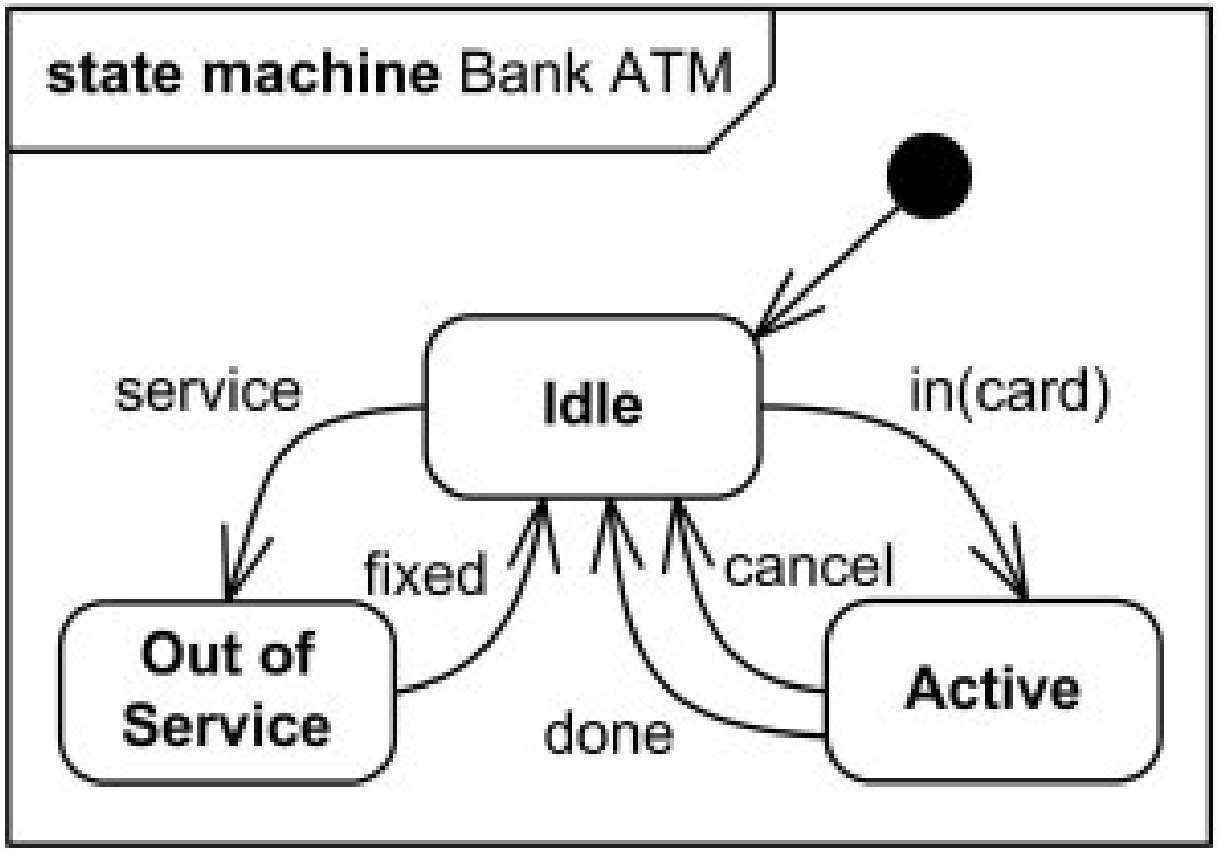
A light is « on » or off (states).
The light can be « turned on » or « turned off » (triggers).
When a light is on, nothing happens if it is turned on (the state do not change).



States in UML State diagram (statechart)

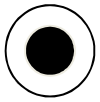
- Initial 
- End 
- Simple 

Example 2



High level behavioral state machine for bank ATM

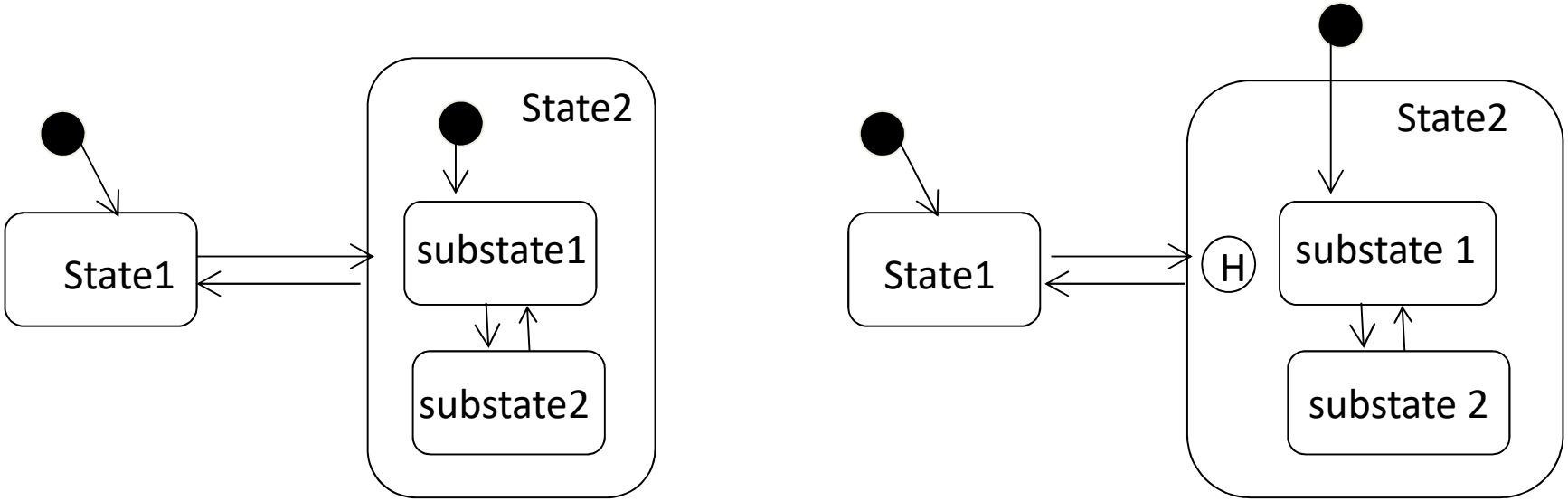
States in UML State diagram (statechart)

- Initial 
- End 
- Simple 

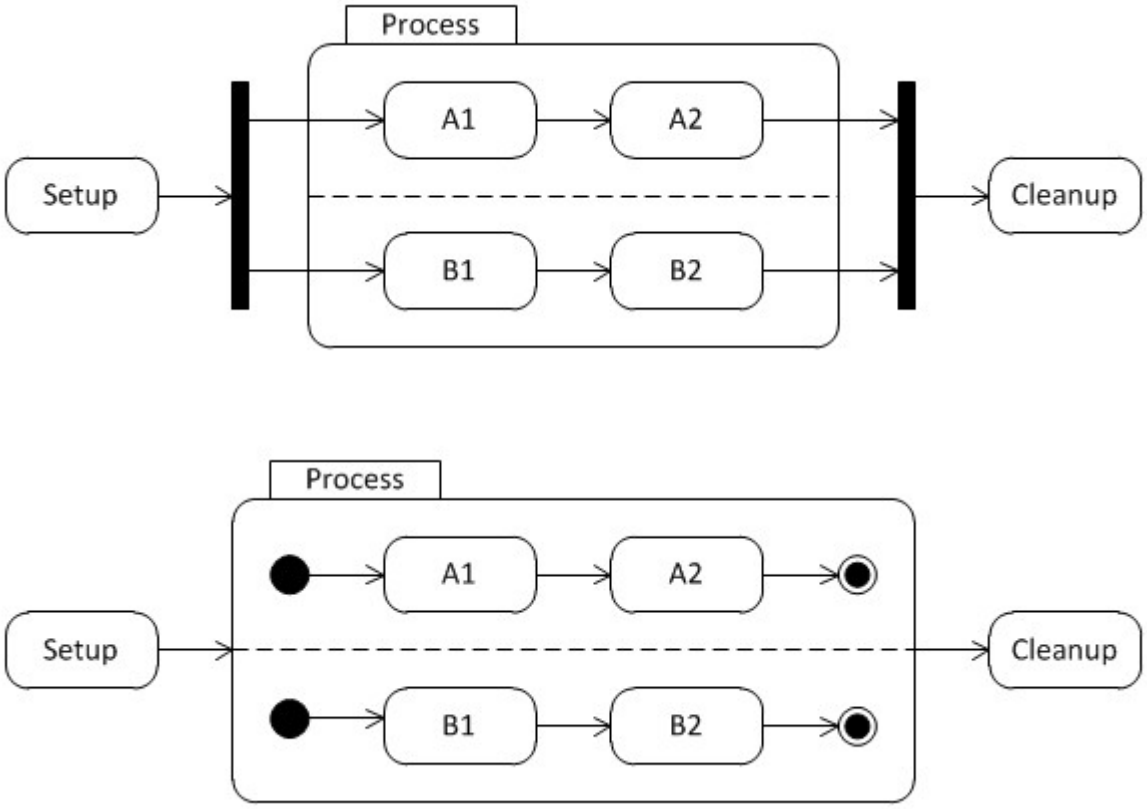
- Composite
  - hierarchically nested states and
  - orthogonal regions

Composite states

- A composite state
  - Can have one, several or none states
- When leaving a state,
  - It is possible or not to keep memory of the left state

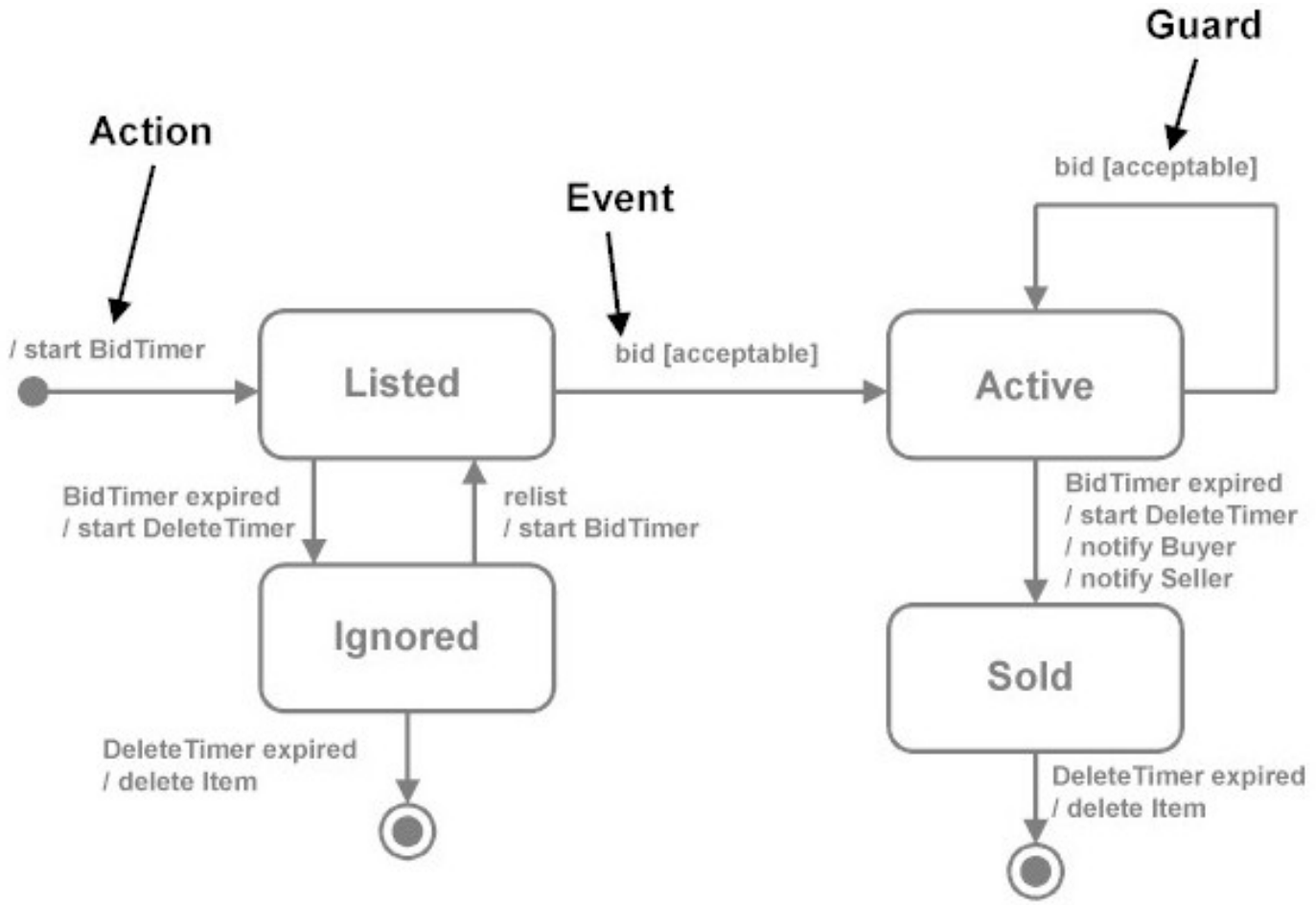


Composite State with parallel activities



« UML, second edition », Schaum's Outline. S. Bennett, J. Skelton, K. Lunn

Example 3 : an object in a bidding system



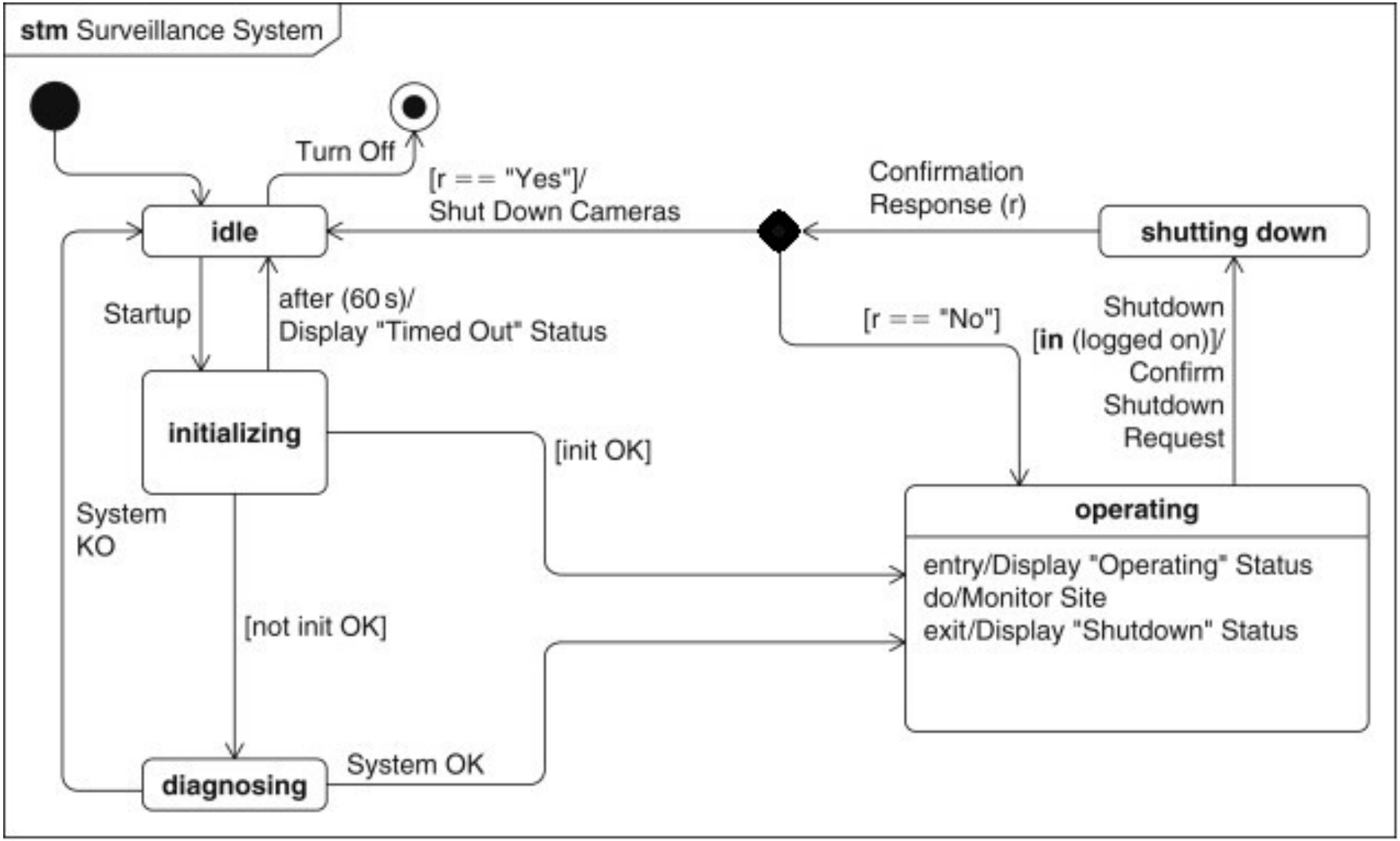
https://www.stickyminds.com/article/state-transition-diagrams

States and labels

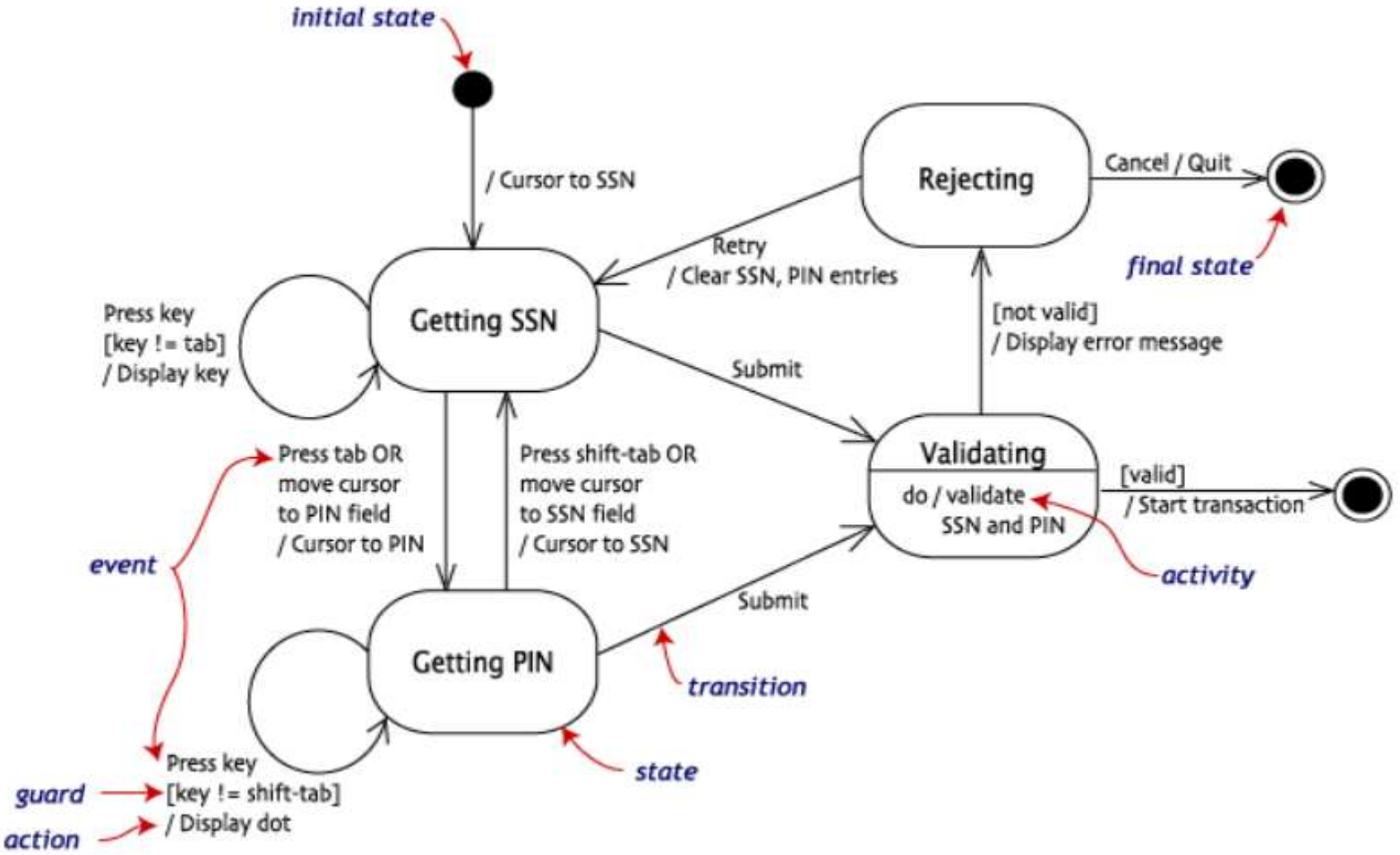
- **Entry**: behavior performed upon entry to the state
- **Do**: ongoing behavior, performed as long as the element is in the state
- **Exit**: behavior performed upon exit from the state



Example 4


stm Surveillance System
Turn Off
[r == "Yes"]/
Shut Down Cameras
Confirmation
Response (r)
idle
shutting down
Startup
after (60 s)/
Display "Timed Out" Status
[r == "No"]
Shutdown
[in (logged on)]/
Confirm
Shutdown
Request
initializing
[init OK]
System
KO
operating
entry/Display "Operating" Status
do/Monitor Site
exit/Display "Shutdown" Status
[not init OK]
System OK
diagnosing

UML State Machine diagram



https://people.eecs.ku.edu/~hossein/810/Readings/UML-diagrams.pdf

Exercises

4.1

4.2

4.4

For the evaluations, you should be able to

- @Midterm
  - Understand a diagram and reformulate it in a textual description
  - Produce a diagram from a simple description
- @Final
  - Reformulate a medium size textual description with the appropriate UML diagram