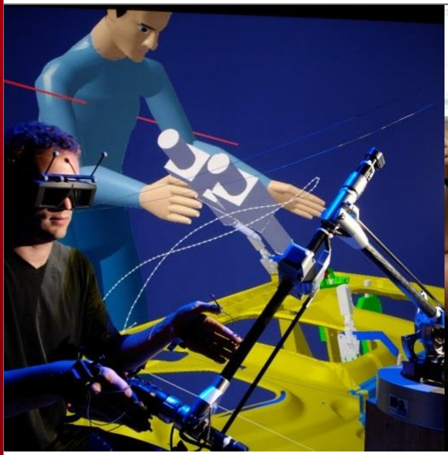
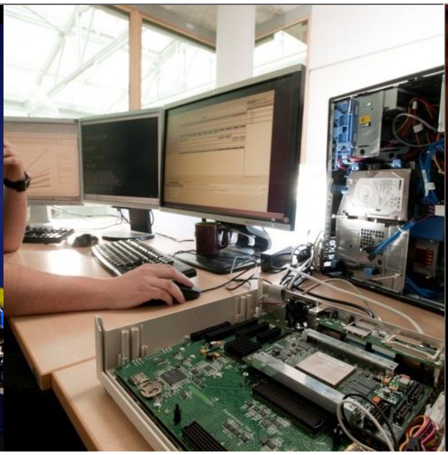
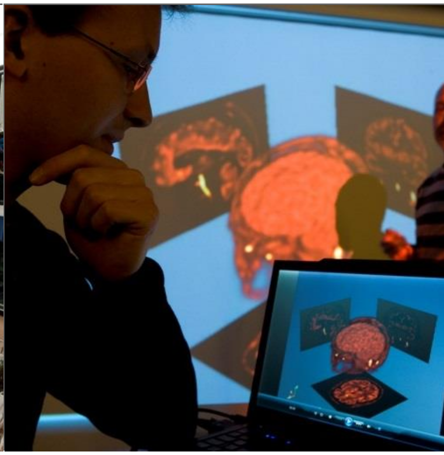
FROM RESEARCH TO INDUSTRY

REQUIREMENTS ANALYSIS AND SYSTEM BOUNDS DEFINITION

Introduction on UML for Industrial Systems

Shuai Li, Jérémie Tatibouët, François Terrier, Sébastien Gérard, **Asma Smaoui**

{first_name}.{last_name}@cea.fr

CONTENT

CONTENT

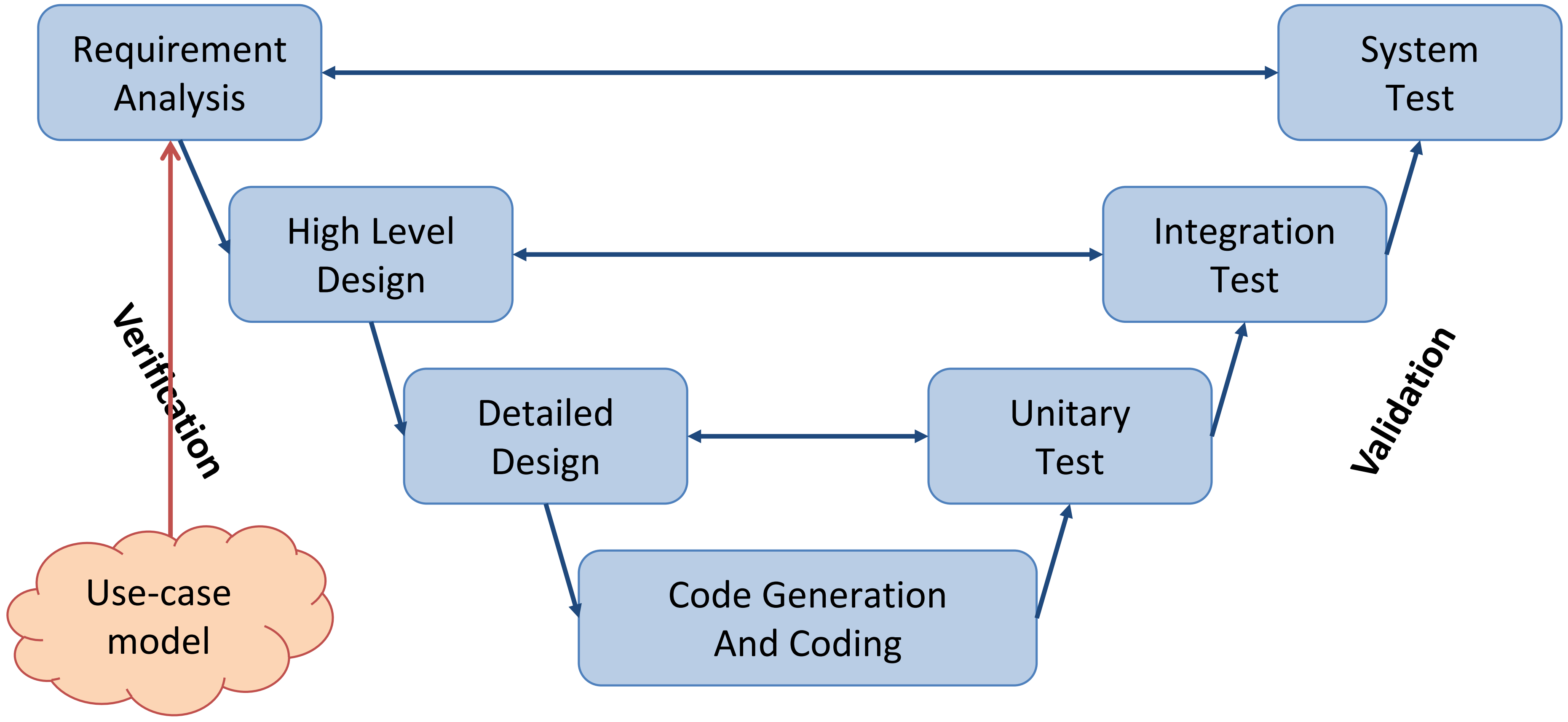
What are requirements and why model them?

- **Before developing, it is important to know:**
  - What the system does
  - What kind of needs it fulfills
- **These are the requirements of the system that you can model in a languages like SysML with dedicated stereotypes**
- **Some requirements can be functional:**
  - List functionalities offered by the system
  - Organize functionalities between them, so to represent the relationships between them
  - Functionalities can be re-used

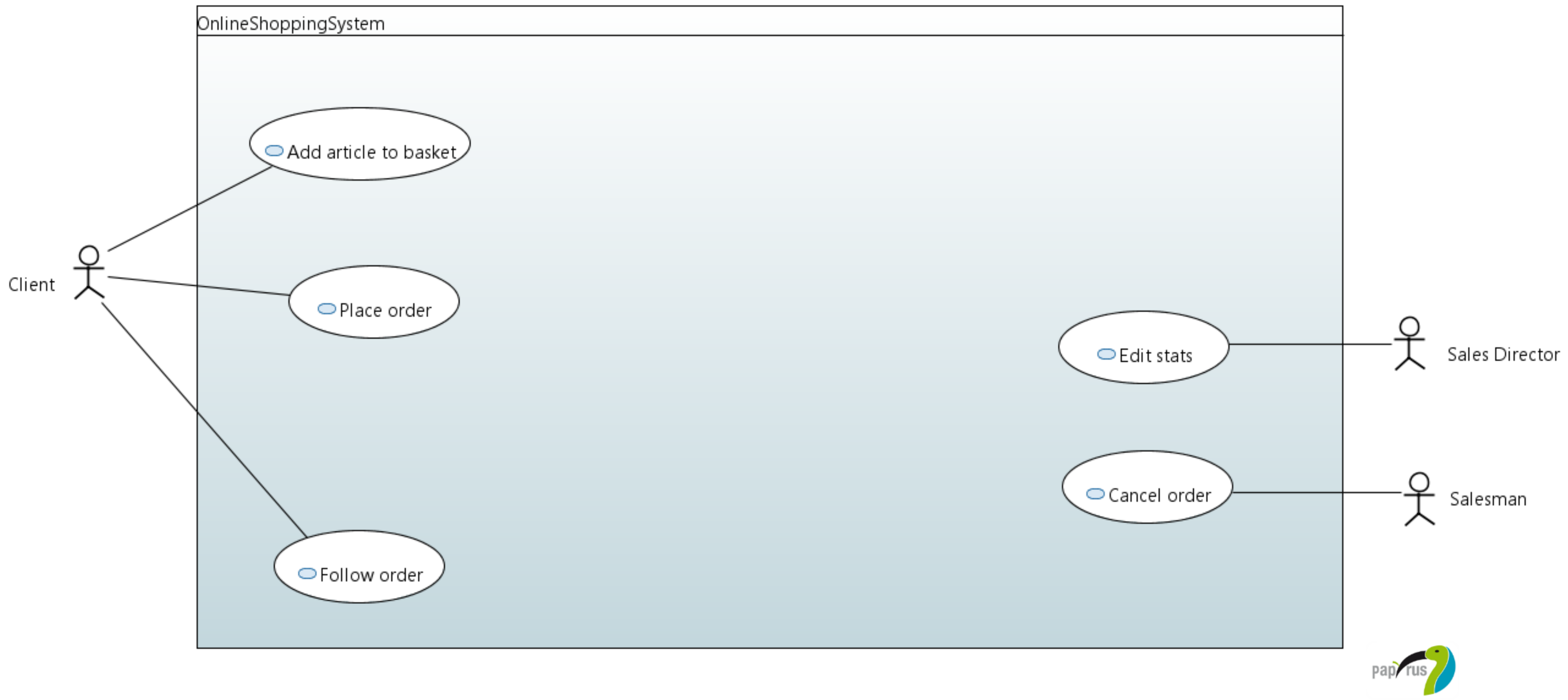
Functional requirements modeling in UML

- **Functional requirements can be modeled in UML with use-case diagrams**

When to model use-cases?



Example of simple use-case model



CONTENT

BASIC ELEMENTS OF USE-CASE MODEL

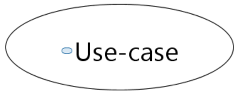
Subject

- The system that offers functionalities, seen as a black box
- Can be modeled with a class in UML

Reminder: Class = set of objects with same features, constraints and semantics

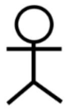
Use-case

- A functionality offered by the system
- Implies a series of elemental actions to be executed
- Example:



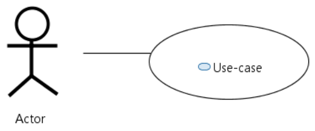
Actor

- Exterior entity to the system which interacts with the system
- Example:



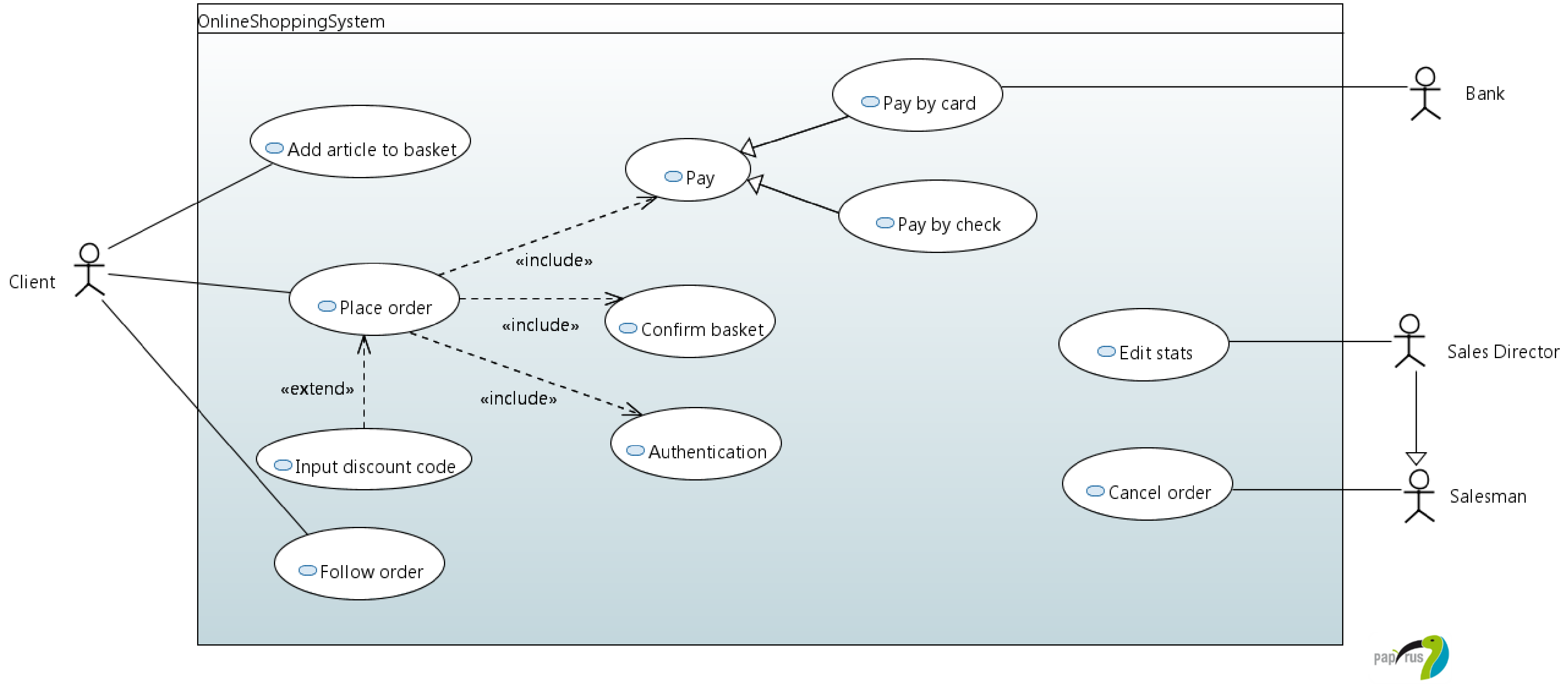
Relationship between actor and use-case

- Actor is related with use-case through an association
- Example:



CONTENT

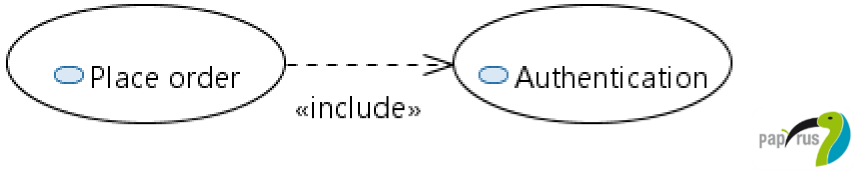
Example of complete use-case model



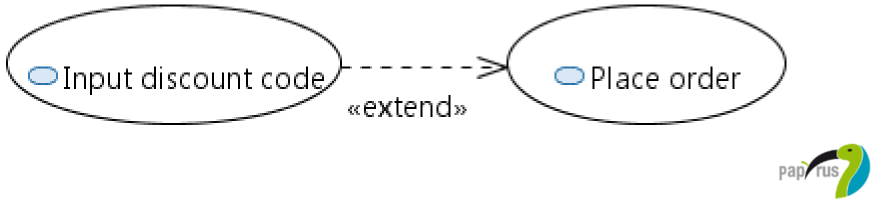
INCLUSION AND EXTENSION

Inclusion and extension

- Inclusion: describes a use-case A is an required part of use-case B
- Example: “Place order” includes “Authentication” means the client must authenticate to place an order



- Extension: describes a use-case A is an optional part of use-case B
- Example: “Input discount code” extends “Place order” means the client may use a discount code when placing an order



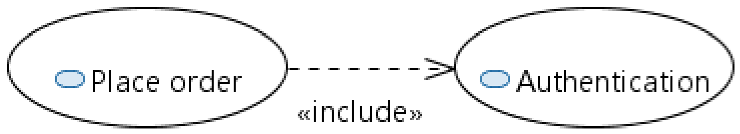
RELATIONSHIPS IN USE-CASE MODEL

Inclusion and extension are dependencies

- **When use-case A includes or extends B, A depends on B**

Reminder: When A depends on B, any modification to B may have an impact on A. In UML, dependencies are represented by dashed arrows.

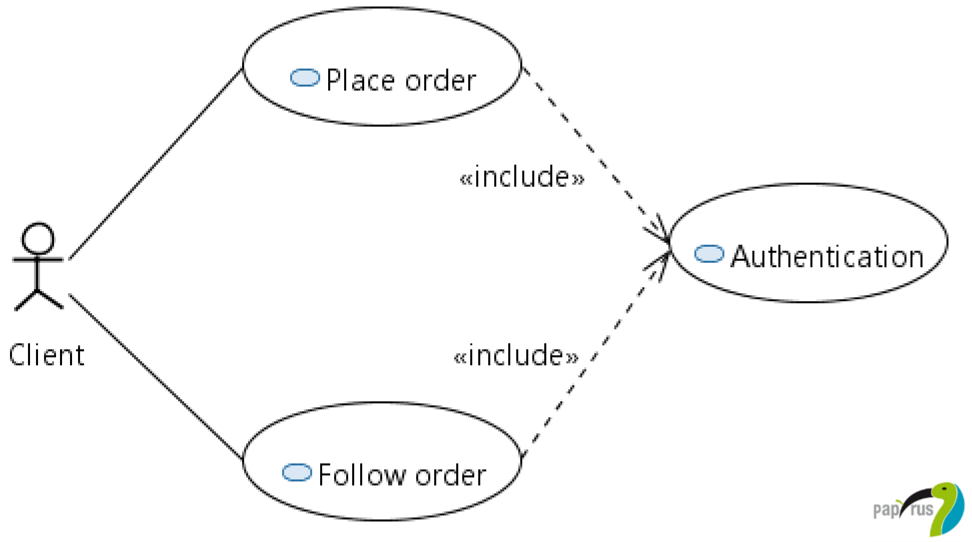
- **Note on graphical notation:**
  - Since inclusion and extension are dependencies, they are share graphical notations with dependency, with keywords <<include>> and <<extend>> to distinguish them



RELATIONSHIPS IN USE-CASE MODEL

Re-usability through inclusions

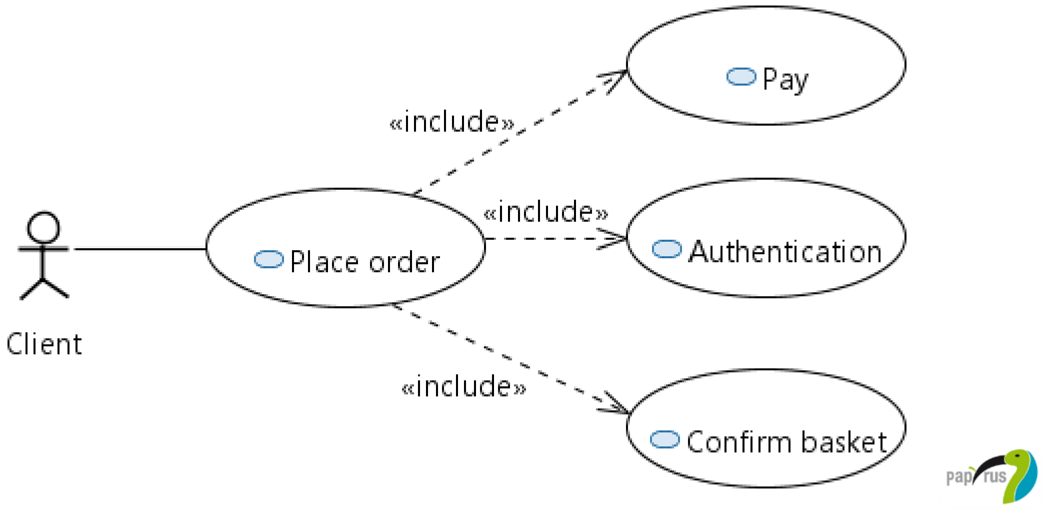
- Use-cases can be re-used through inclusions
- Example: “Authentication” is re-used for both “Place order” and “Follow order” use-cases



RELATIONSHIPS IN USE-CASE MODEL

Decomposition through inclusions

- Inclusion can decompose a complex use-case into several use-cases
- Example: “Place order” is a complex use-case that includes several simpler use-cases



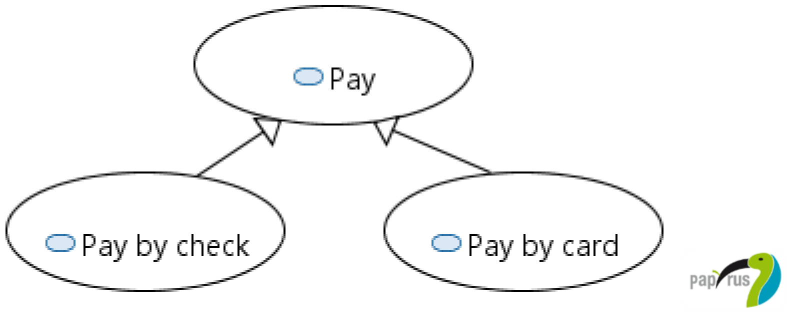
RELATIONSHIPS IN USE-CASE MODEL

Inheritance between use-cases

- A use-case can inherit another use-case

Reminder: Inheritance = relationship between a general element A, and a specialized element B. When B inherits A, B "is a" A with specific features. Inheritance is modeled in UML with a generalization. Graphically it is an arrow with a hollow triangle pointing to the general element.

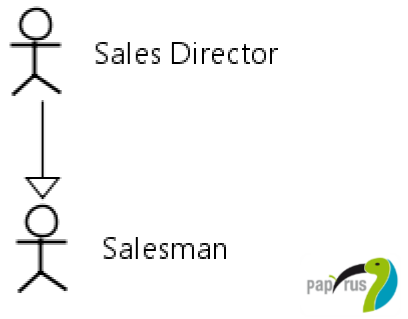
- Example: "Pay by card" inherits "Pay" because paying by card is a special kind of payment; same for "Pay by check"



RELATIONSHIPS IN USE-CASE MODEL

Inheritance between actors

- The only relationship possible between actors is inheritance
- Example: A “Sales Director” is a “Salesman”



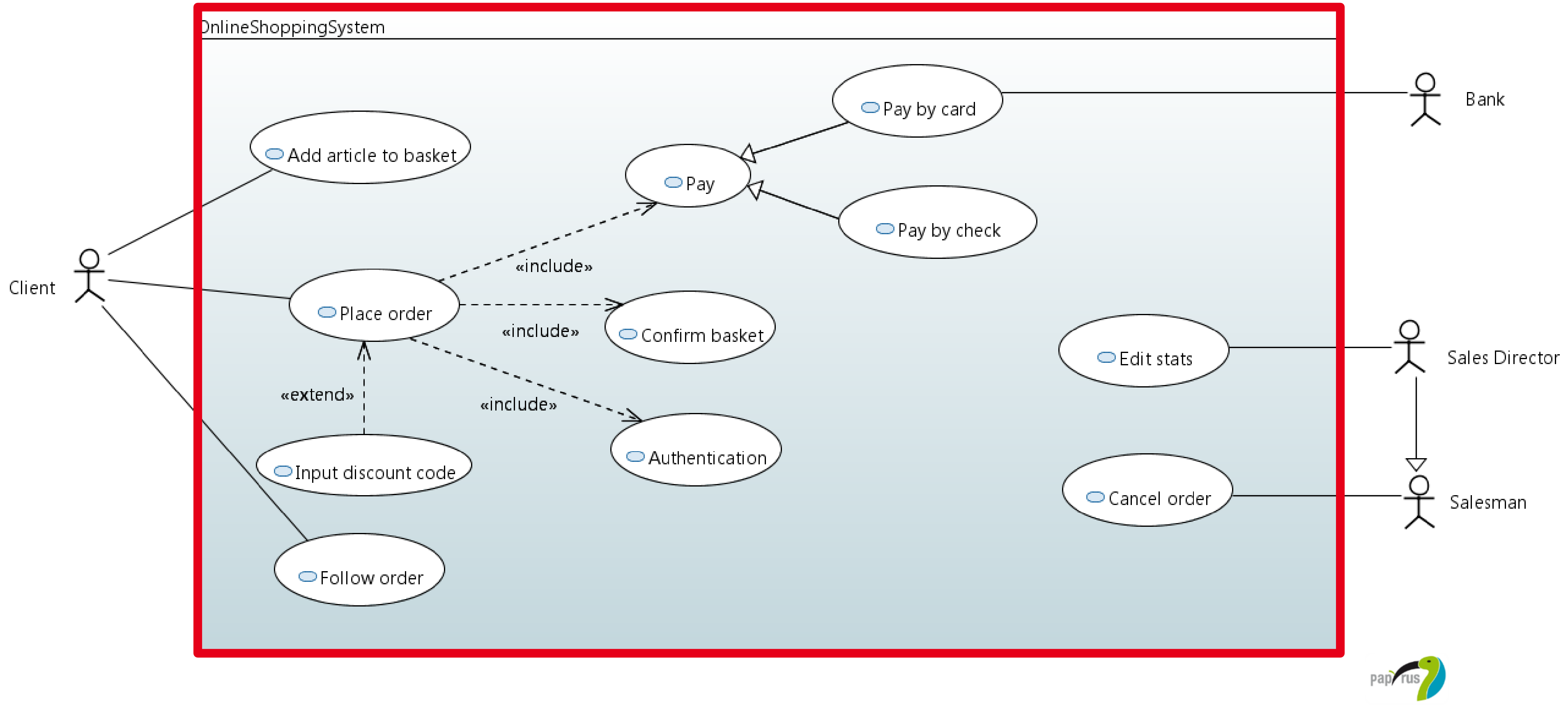
CONTENT

Identify actors

- **An actor is a user of the system**
- **An actor is a role, not necessarily a physical person:**
  - A same person can be represented as several actors if the person has several roles
  - If several persons play the same role relative to the system, they are represented as a same actor
- **Actors can also be, e.g.**
  - Devices used by the developed system (e.g. printer)
  - Available software that can be used by the developed system
  - Any external system that can interact with the developed system

Identify actors

- To identify actors, think about the system border

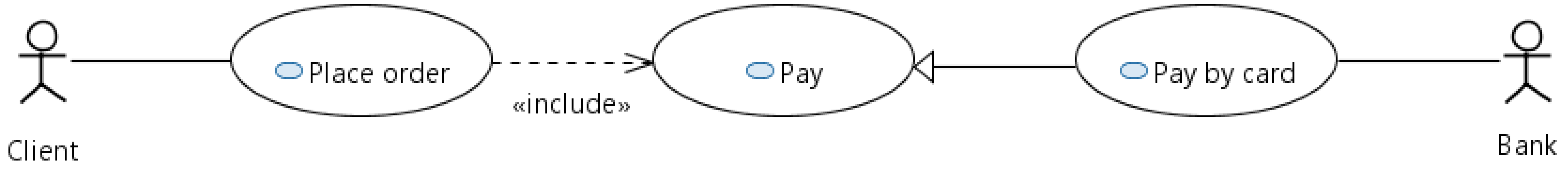


🟥 System border

DESCRIBING A USE-CASE MODEL

Primary and secondary actors

- An actor is said to be primary for a use-case, if the actor initiates the use-case
- A use-case may interact with a secondary actor that isn't the one that initiated the use-case
- E.g. secondary actors may be devices connected to the developed system
- Example: The "Client" initiates the use-case to "Pay by card" which interacts with the "Bank"; "Bank" is thus a secondary actor



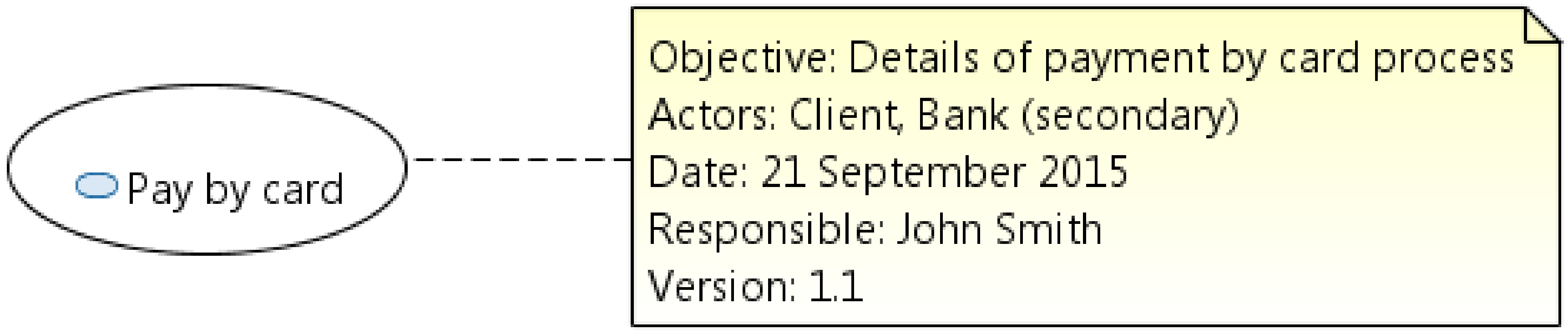
DESCRIBING A USE-CASE MODEL

Identify use-cases

- Identifying use-cases is not automatic
- Remember the subject is a black box: use-cases describe functionalities as seen by the outside world!
- Consider you are the actor and think of how you will use the system, in which cases do you use it, what are the services that you want to access
- Avoid redundancy and limit the number of use-cases by choosing the correct level of abstraction (e.g. several actions may be part of a same use-case)
- You are not modeling the details of each use-case, so stay at a level of abstraction that only describes the main services of the system
- Finding the correct level of abstraction is not simple, but you will become more experienced as you model use-cases

Describe use-cases

- Too much detail in a use-case model is not good but too less is insufficient: a simple name for a use-case does not describe all aspects of a use-case
- The use-case must be described textually and the text may be in a UML comment
- Example:

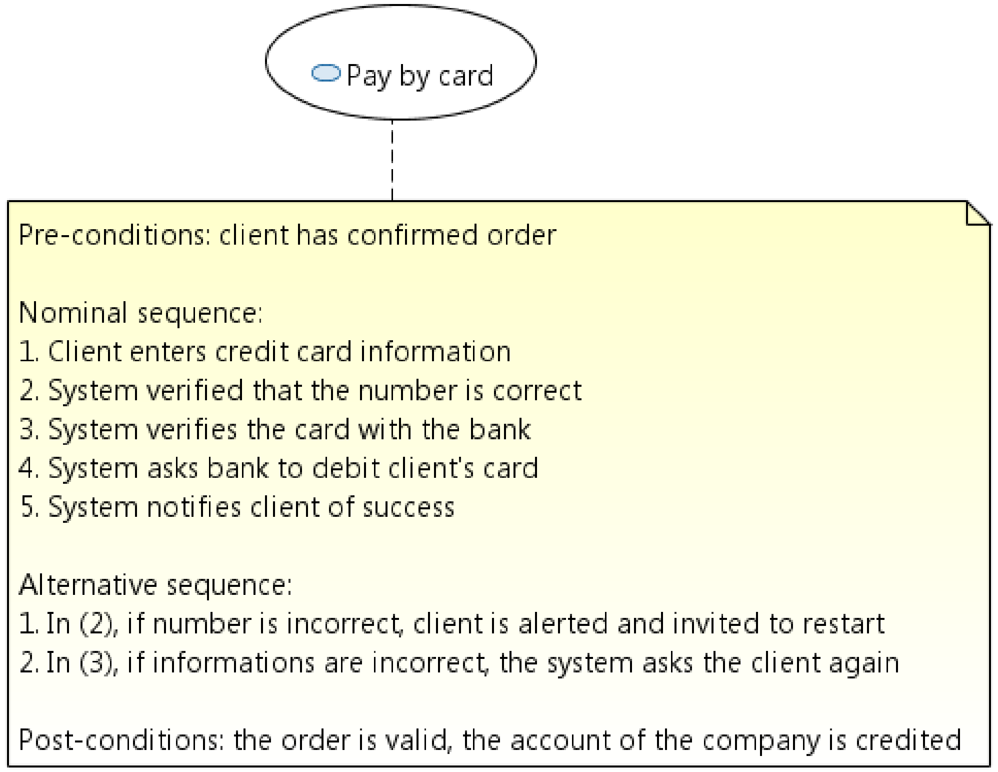


Objective: Details of payment by card process
Actors: Client, Bank (secondary)
Date: 21 September 2015
Responsible: John Smith
Version: 1.1

DESCRIBING A USE-CASE MODEL

Describe use-cases

- You can also describe the behavior of a use-case
- Behavior can be modeled in UML as we will see in the future… for now let us consider a textual description only
- Example:



Pre-conditions: client has confirmed order

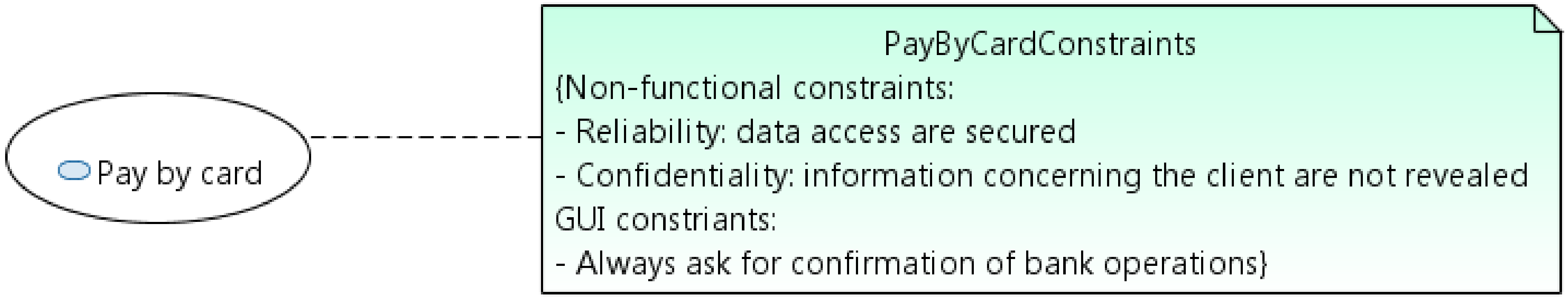
Nominal sequence:
1. Client enters credit card information
2. System verified that the number is correct
3. System verifies the card with the bank
4. System asks bank to debit client's card
5. System notifies client of success

Alternative sequence:
1. In (2), if number is incorrect, client is alerted and invited to restart
2. In (3), if informations are incorrect, the system asks the client again

Post-conditions: the order is valid, the account of the company is credited

Describe use-cases

- Finally you can also constrain a use-case
- UML constraints can be used for this purpose
- Example:



{Non-functional constraints:
- Reliability: data access are secured
- Confidentiality: information concerning the client are not revealed
GUI constriants:
- Always ask for confirmation of bank operations}

CONTENT

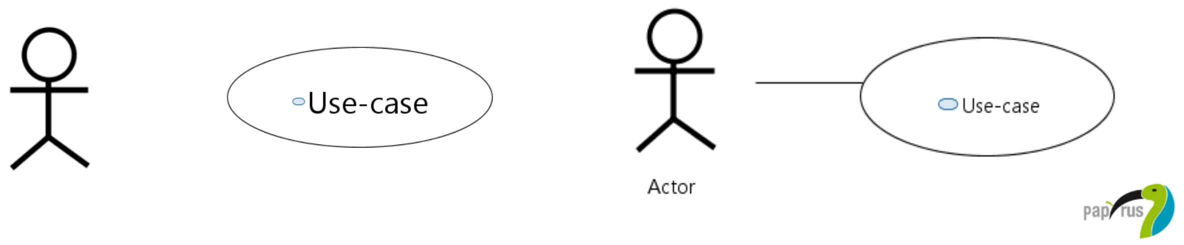
QUIZ

?

What are the basic elements of a use-case model?

!

Actor, use-case, and association actor-use-case



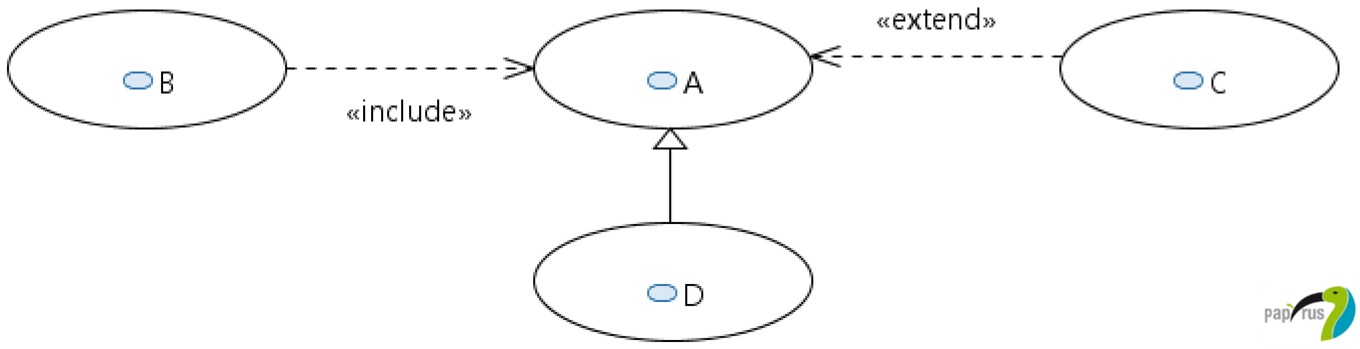
QUIZ

?

What are the relationships between use-cases?

!

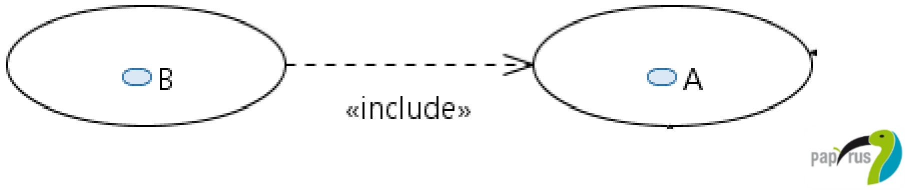
Inclusion, extension, generalization



QUIZ

?

Use-case B includes A means A is:
(1) Optional (2) Required



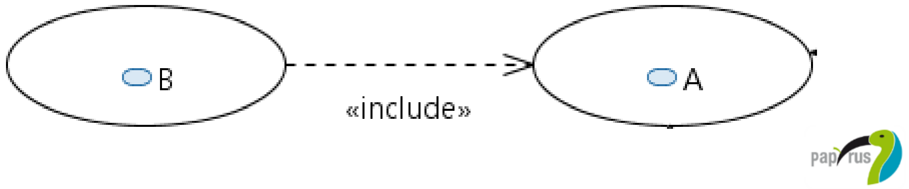
!

(2) Required

QUIZ

?

In the model below, who depends on who? Who is included in who?



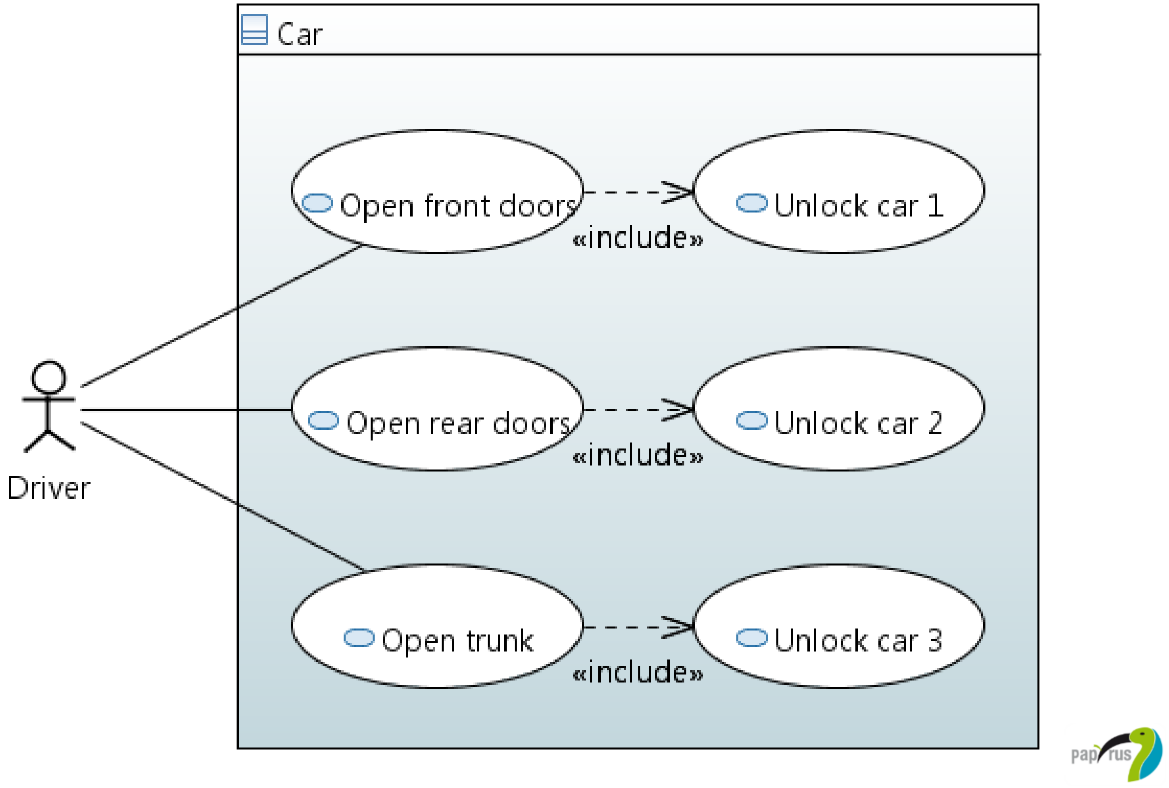
!

B depends on A, and A is included in B.

QUIZ

?

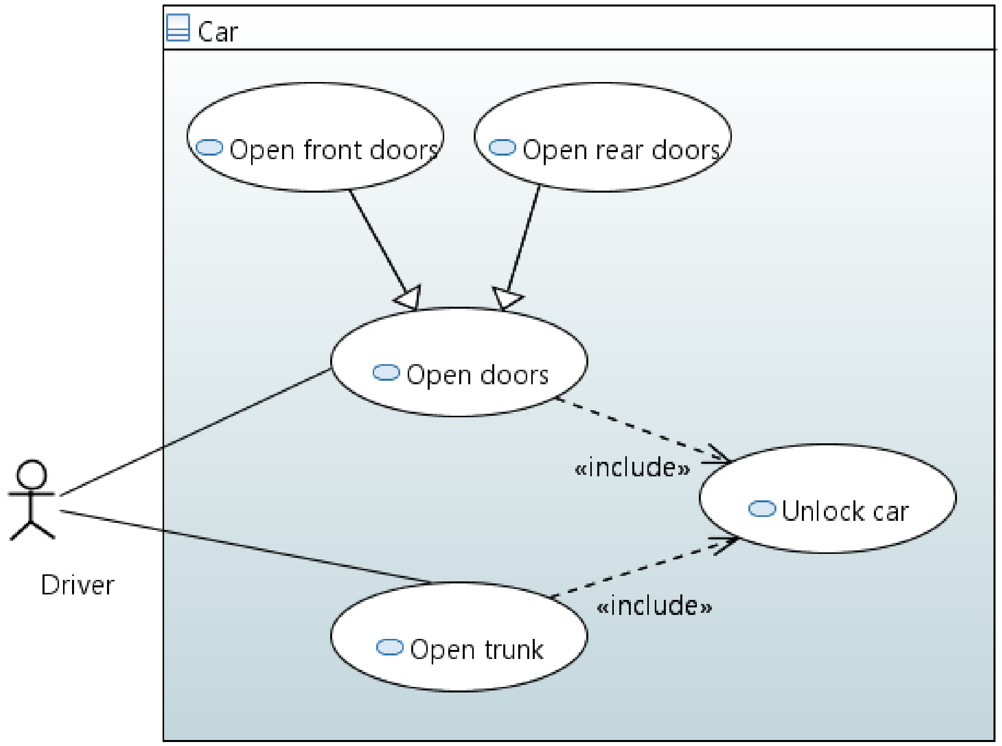
Is the model below optimal? If not, what is one way to optimize it?



QUIZ

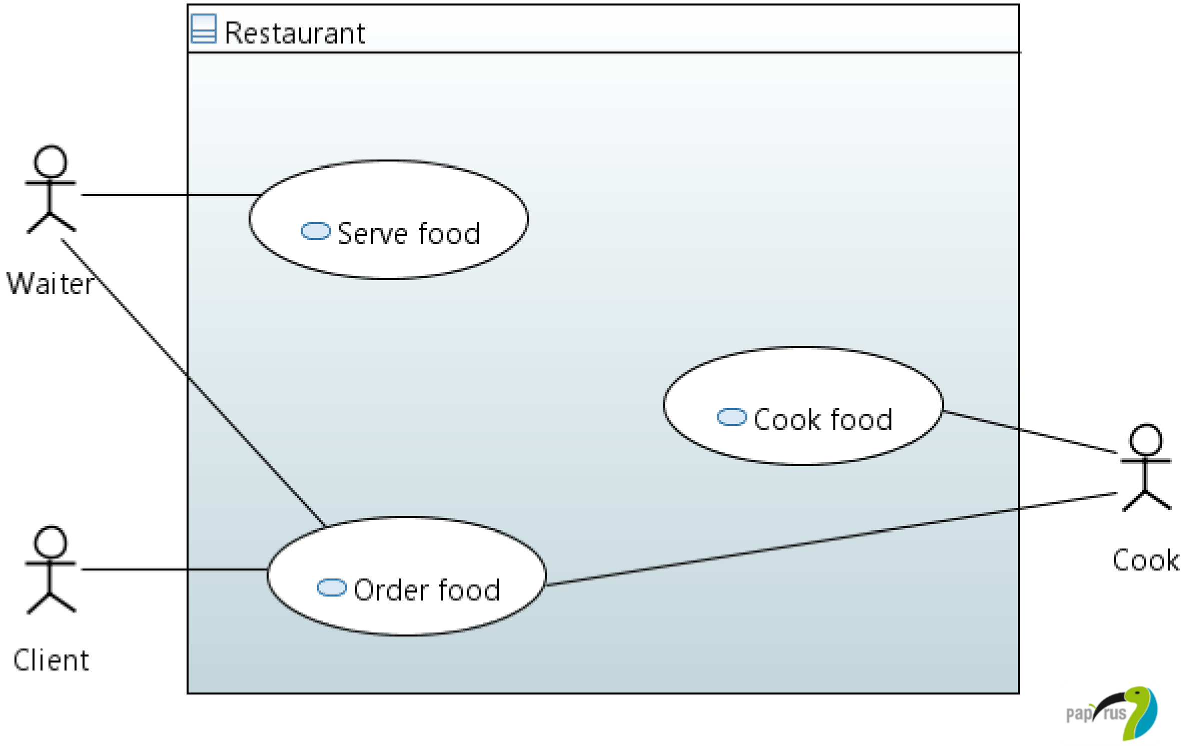
!

"Open front doors" and "Open rear doors" should be a generalization of "Open doors". "Unlock car" should not be several use-cases, but one that is re-used.



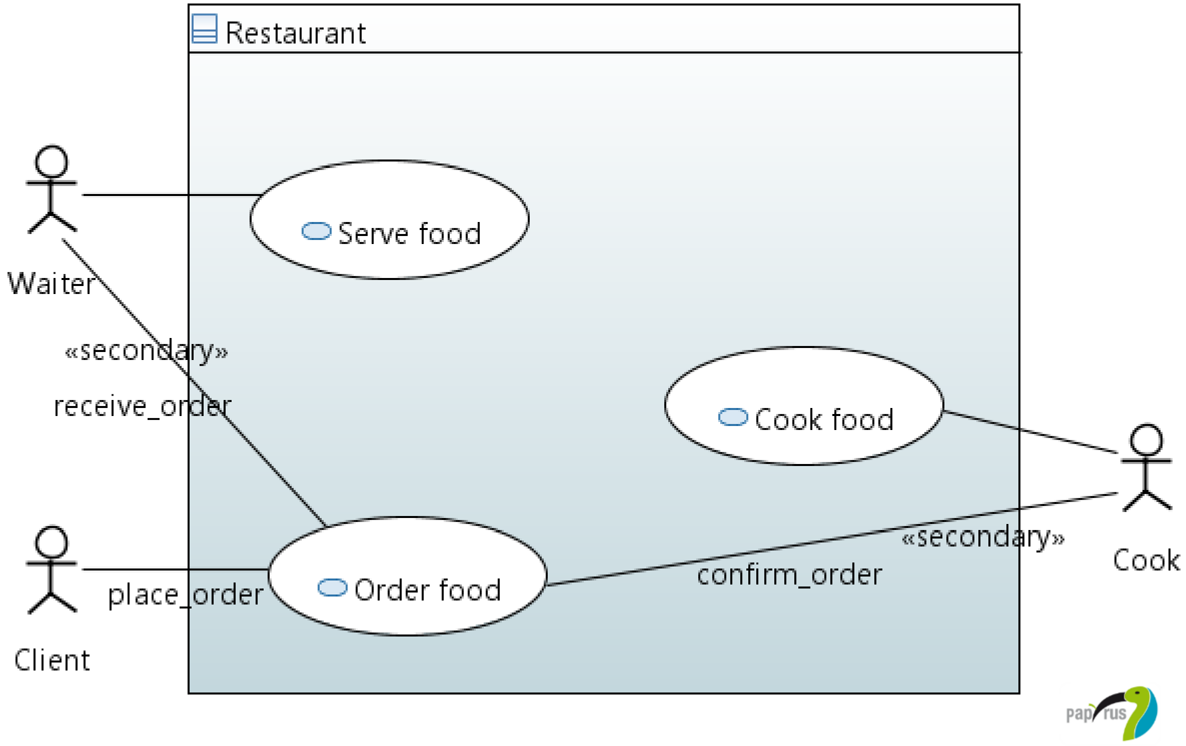
QUIZ

Who are the secondary actors?



!

"Waiter" and "Cook" are secondary actors for "Order food" but they are also primary actors for "Serve food" and "Cook food" respectively.



CONTENT

SUMMARY

- Before developing a system, it is important to analyze its requirements to show what it does, what needs it fulfills
- In UML, use-case models are used
- A use-case model has actors that interact with a system composed of use-cases
- Use-cases may be organized through inclusion, extension, and generalization relationships
- Actors are roles and they are identified by distinguishing the border of the system
- Use-cases must be modeled with correct level of abstraction…
- …but they must also be described, e.g. textually with UML comments and constraints