Introduction to Software Engineering

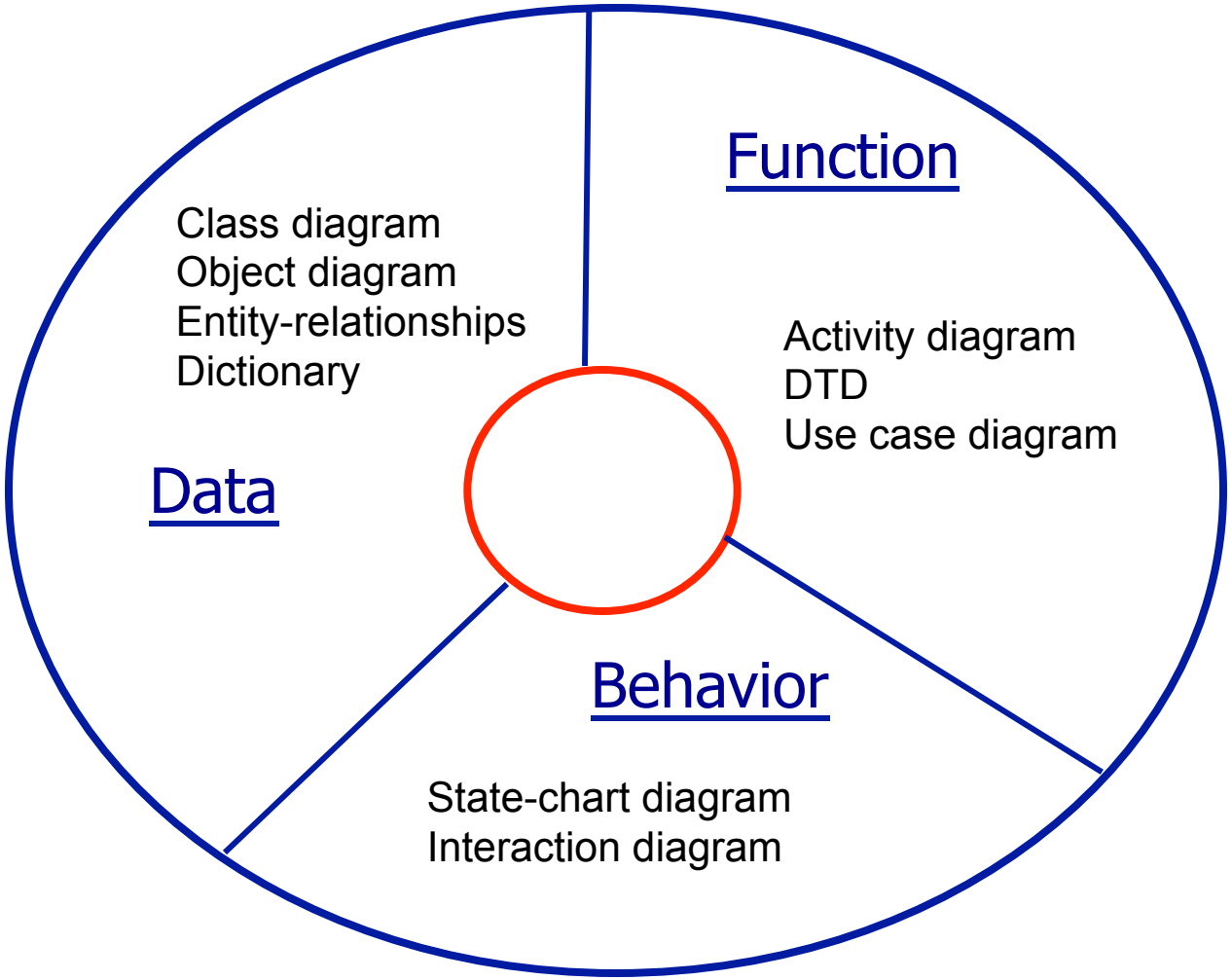
Use cases

Philippe Lalanda

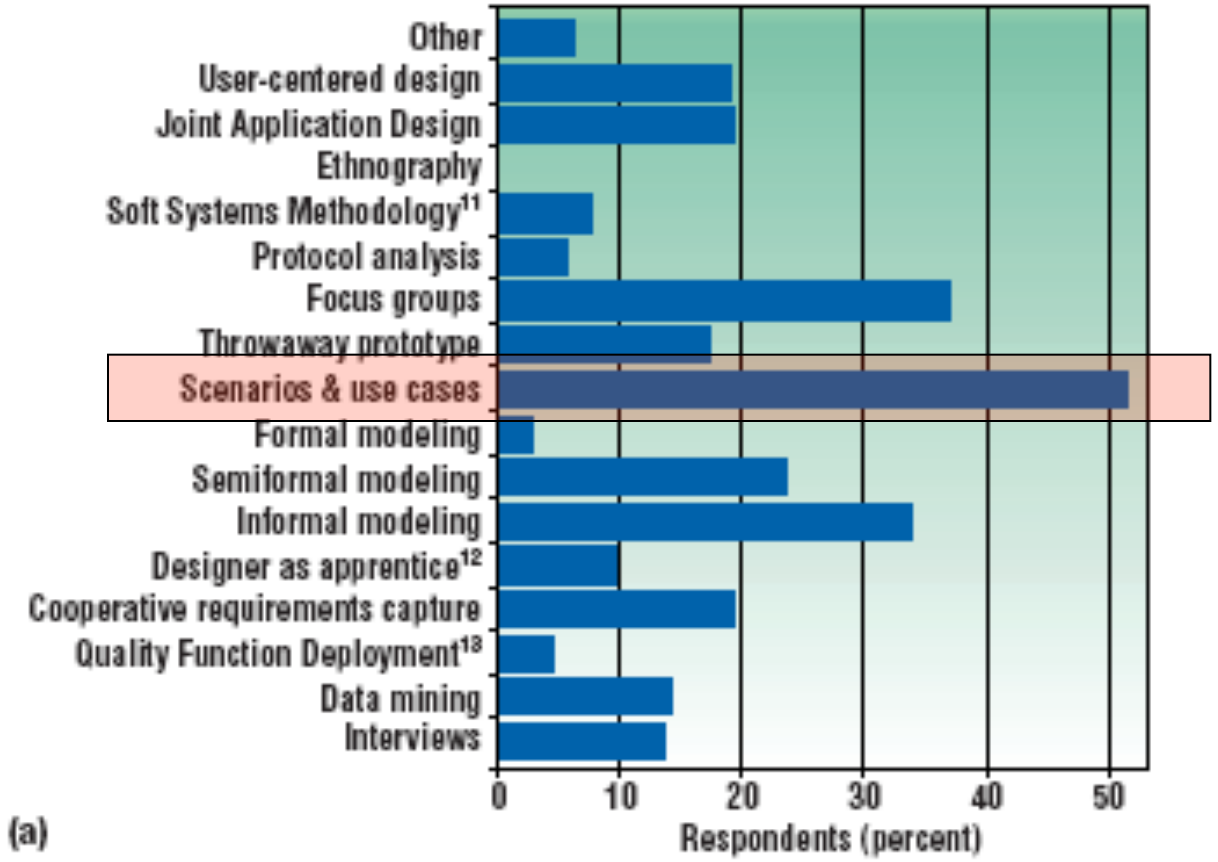
Philippe.lalanda@imag.fr

http://membres-liglab.imag.fr/lalanda/

Reminder


Function
Class diagram
Object diagram
Entity-relationships
Dictionary
Activity diagram
DTD
Use case diagram
Data
Behavior
State-chart diagram
Interaction diagram

Reminder



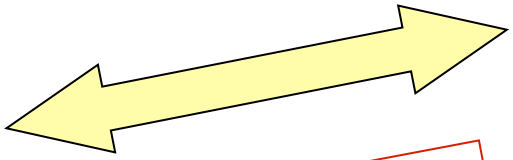
Requirements engineering: The state of the practice, Neill et Laplante, IEEE Software, 2003

Outline

- Definition
- Use cases and UML
- Unified process
- Recurring issues
- Conclusion

What is a use case

- A nominal usage of the system
  - By one or several users
  - Achieving a user's objective



Get some money

What is an objective

- An important, meaningful action to be achieved
  - Not too low level, not too high level

Use ATM

Withdraw money

Enter Pin

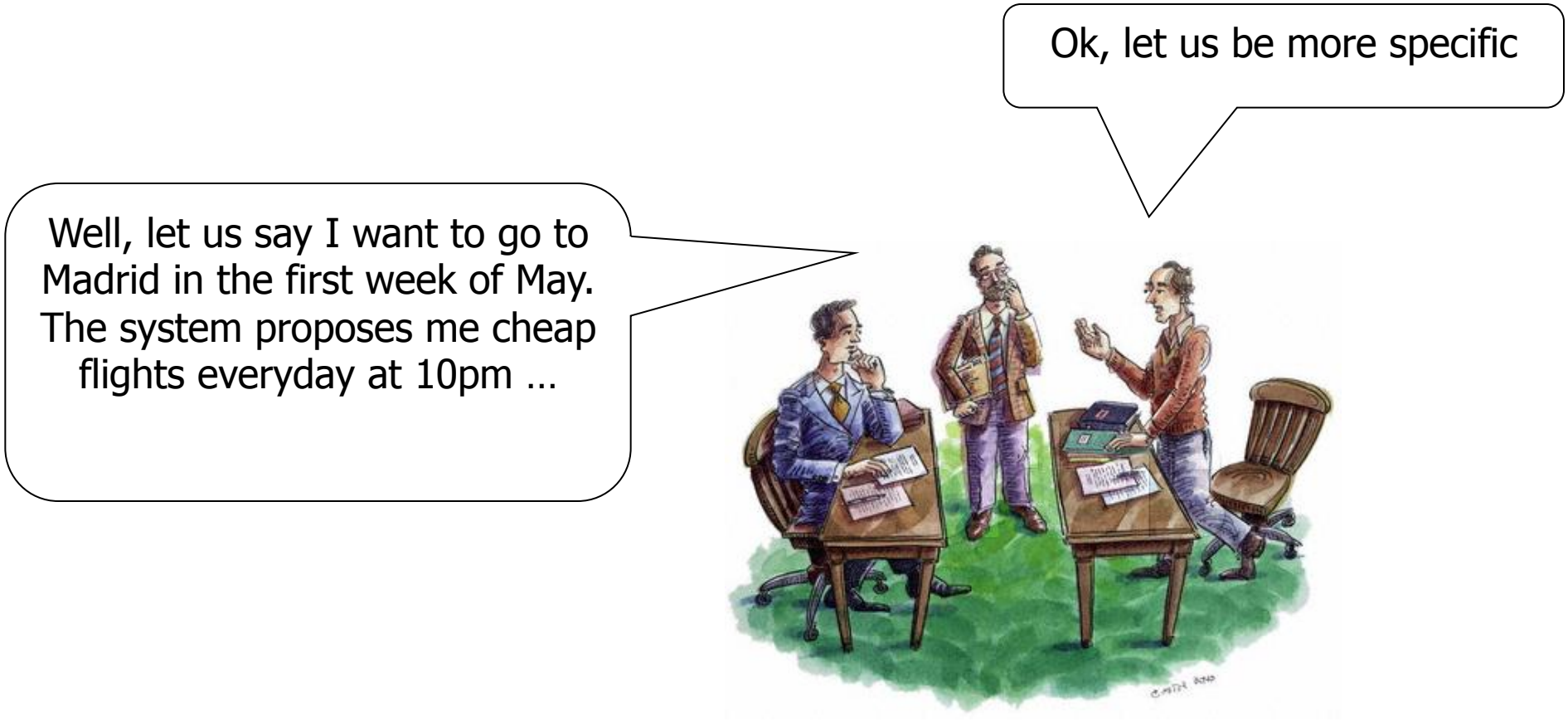
Use case description

- A use case describes precisely the interaction between a user and a system
  - Nominal interaction

Well, I use the system to reserve flights. To do so, first, I give my destination and a list of possible dates. Then, the system gives me flights listed by prices, …

What is a scenario

- A scenario is a use case *instance*



An analysis technique?

- Not really, but an excellent support for analysis
  - Use cases can be built during interviews, brainstorming session, seminars

Use cases and requirements (1)

- Use cases express a set of requirements in a specific form
  - Essential functions of the system and the way they are achieved in terms of interactions

Reserve a flight

Cancel a reservation

Get information about a flight

See all flights

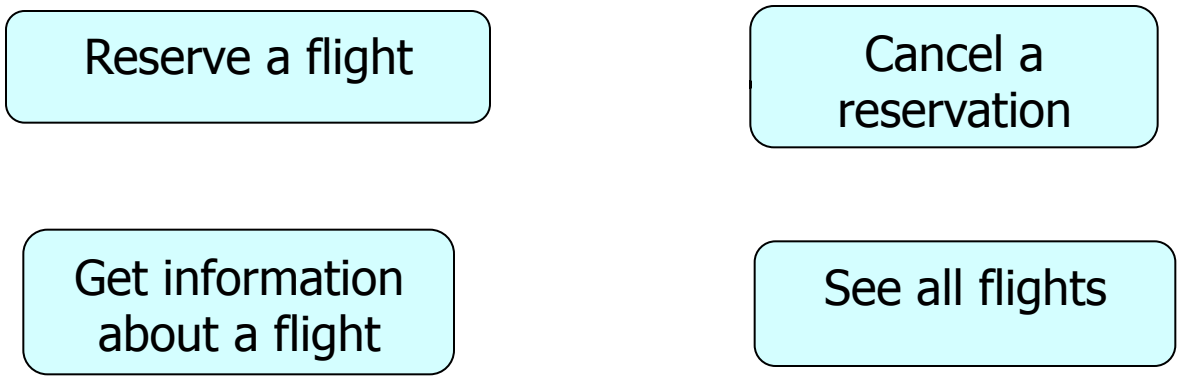
Use cases and requirements (2)

- The description of an interaction includes objects, functions, states

An important goal is to reserve a flight. To do so, first, I give my destination and a list of possible dates. Then, the system presents me flights listed by prices, …

Use cases and tests

- Use cases can be used to structure tests
  - Test definition
  - Integration



Outline

- Definition
- Use cases and UML
- Unified process
- Recurring issues
- Conclusion

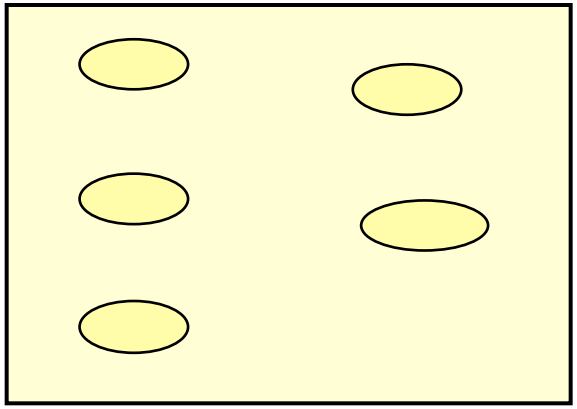
Use cases and UML

- UML is an object-oriented notation to build
  - analysis models
  - design models
  - (a bit of) implementation models

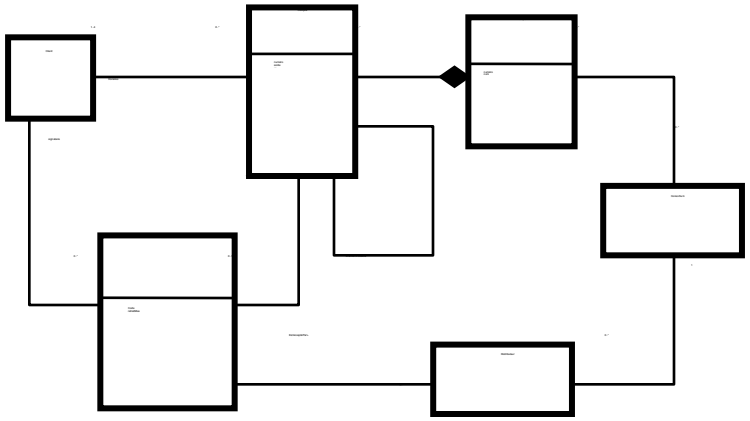
- UML acknowledged that OO analysis schemas were:
  - Not sufficient to capture all requirements
  - Not user friendly
  - Focused on domain data (objects)

UML and requirements engineering

- Use cases have been integrated by Ivar Jacobson
  - Requirements = Use cases + analysis diagrams



Essential functions



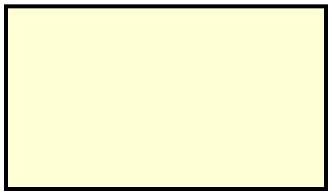
Essential objects

Use cases and UML

- UML provides simple notations to define use cases (and scenarios)
  - Very few symbols
  - Easy to understand
  - Informal (mainly textual)

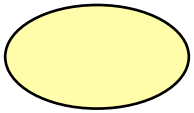
- Integrated in the RUP (Rational Unified Process)
  - A process defining and orchestrating UML-based activities

Base elements

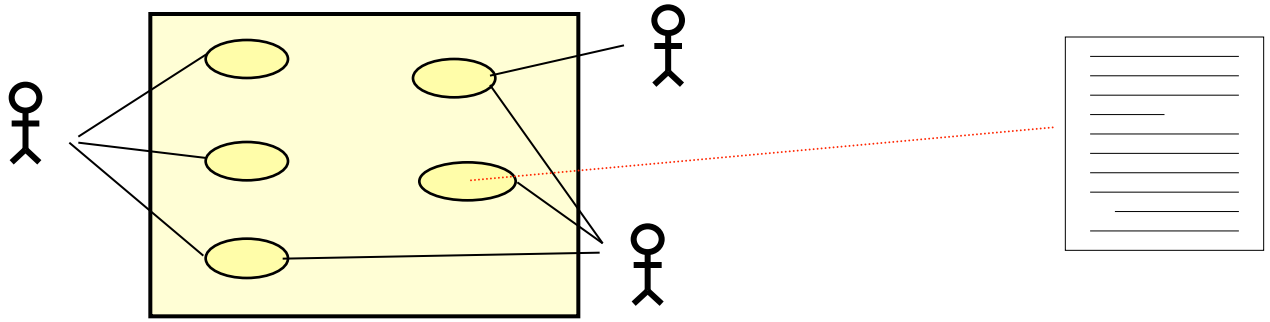


System

Actor



Use case

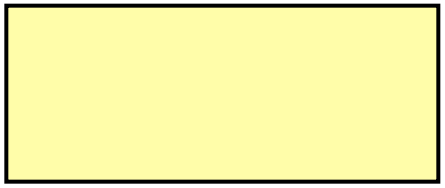


Use case diagram

Use case model

System

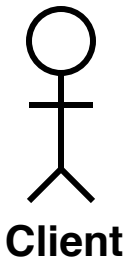
- A system is seen as a black box
  - It is defined through the functions it provides, the interactions it supports

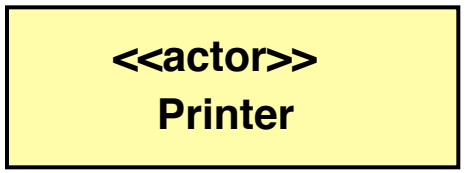


BooksOnLine

Actors

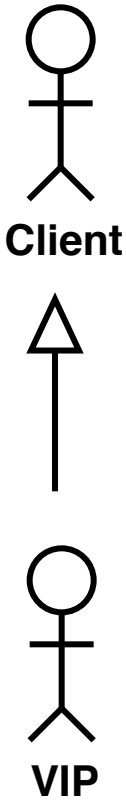
- An external element interacting with the system
  - He can be active (takes decision) or passive (gets information)
- Two kinds of actors
  - Human
  - System (external systems, devices, peripherals)
- Several notations



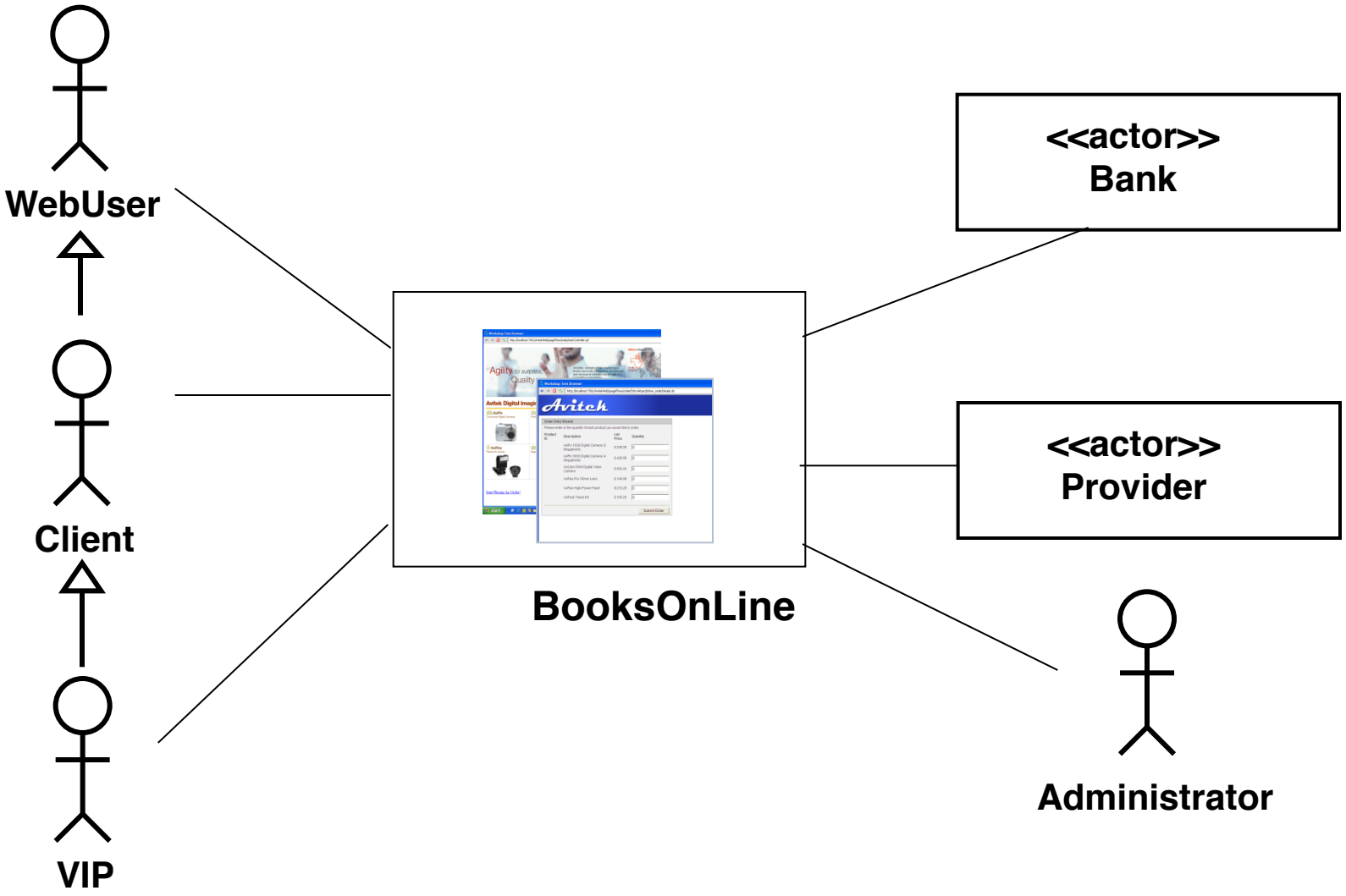


Actors and inheritance

- Only possible relation between actors
- The specialized actor inherits all the use cases of his father



Example

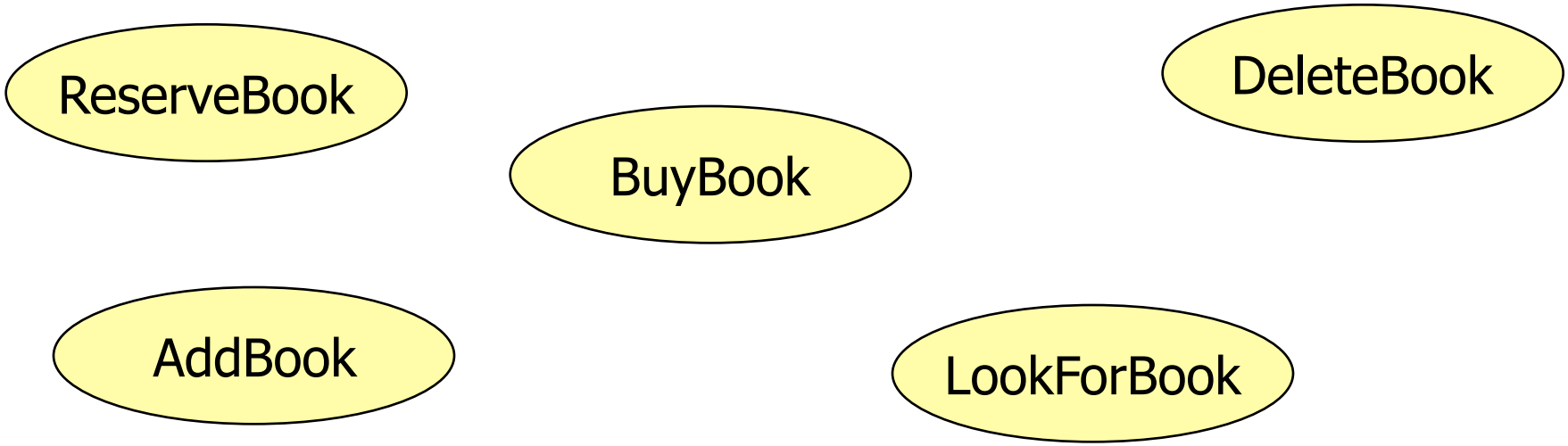

WebUser
Client
VIP
<<actor>>
Bank
<<actor>>
Provider
Administrator
BooksOnLine

Actors *vs.* users

- Actors and actual users must not be confused
  - An actor = a role
  - A user = a person
- Several persons can play a same role
  - *Paul* and *Pierre* are two *clients*
- A same person can play several roles
  - *Paul* can be a *CLient* and a *WebUser*
- A role relates to a system not to the organization using the system
  - *Client* rather than *Student*

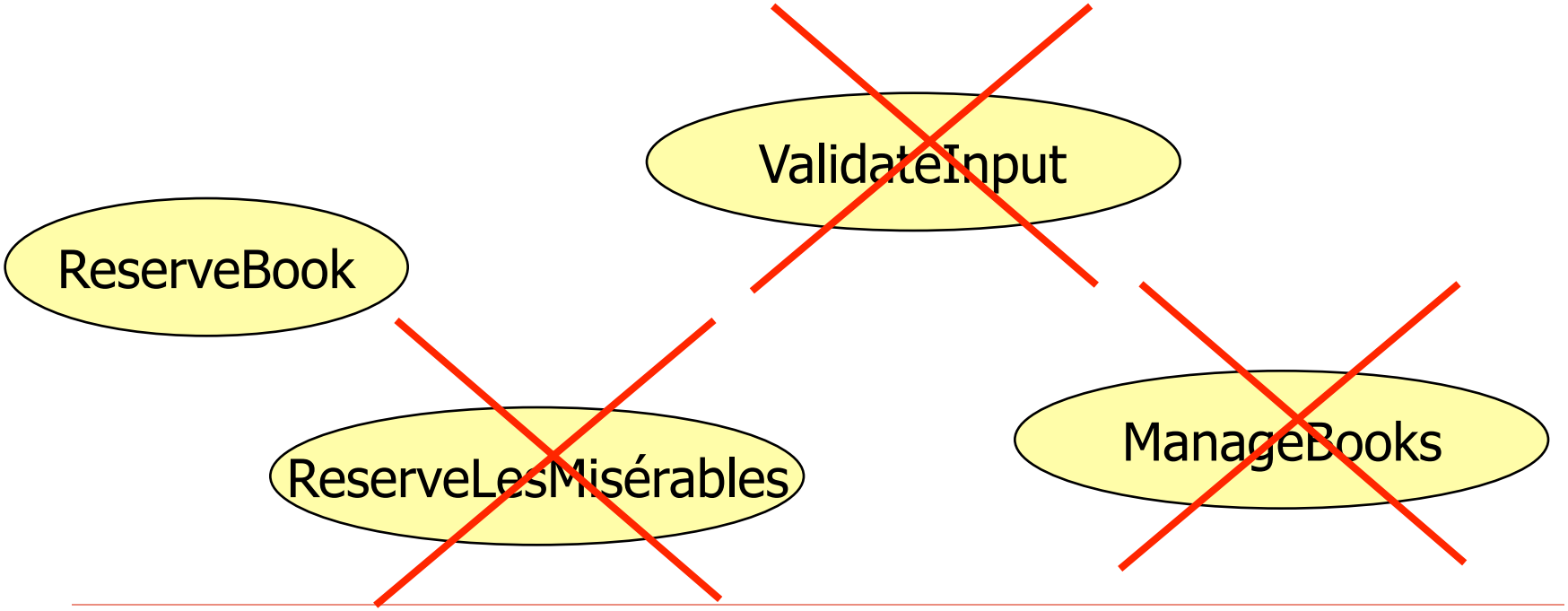
Use case

- A use case refers to an essential function
  - It may include several actors
  - It corresponds to a high level goals



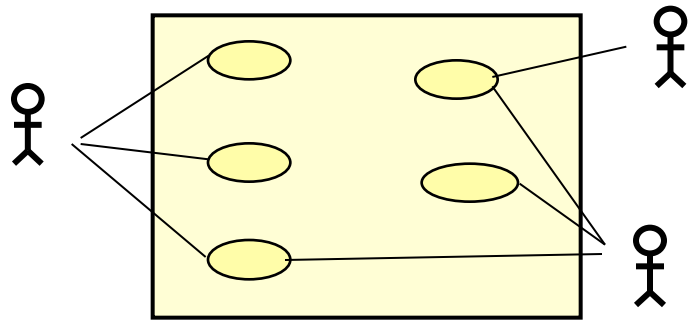
Use case

- A use case
  - Allows an actor to reach a goal
  - It corresponds a a function visible by the actor(s)
  - It has to make sense (specific enough)

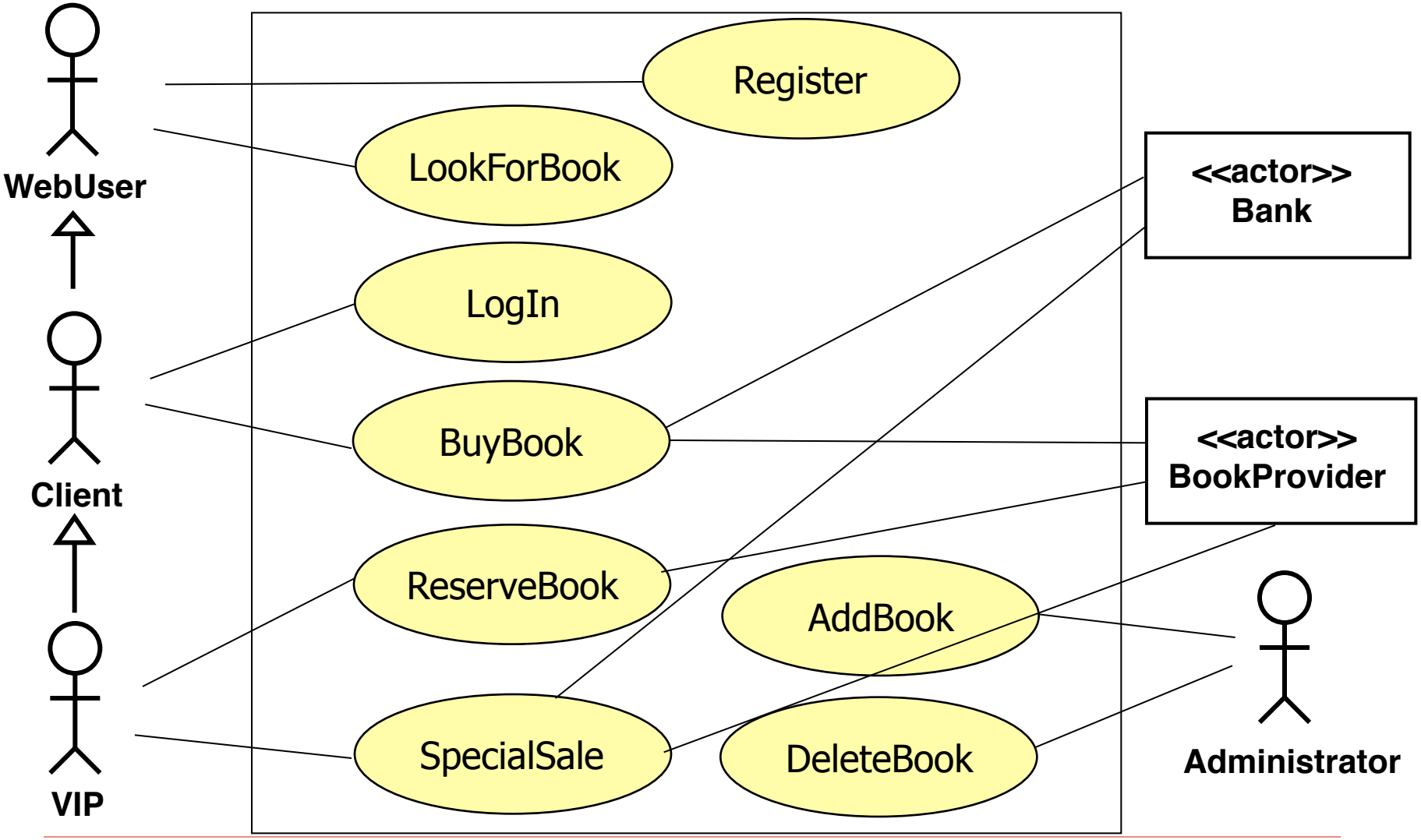


Use case diagram

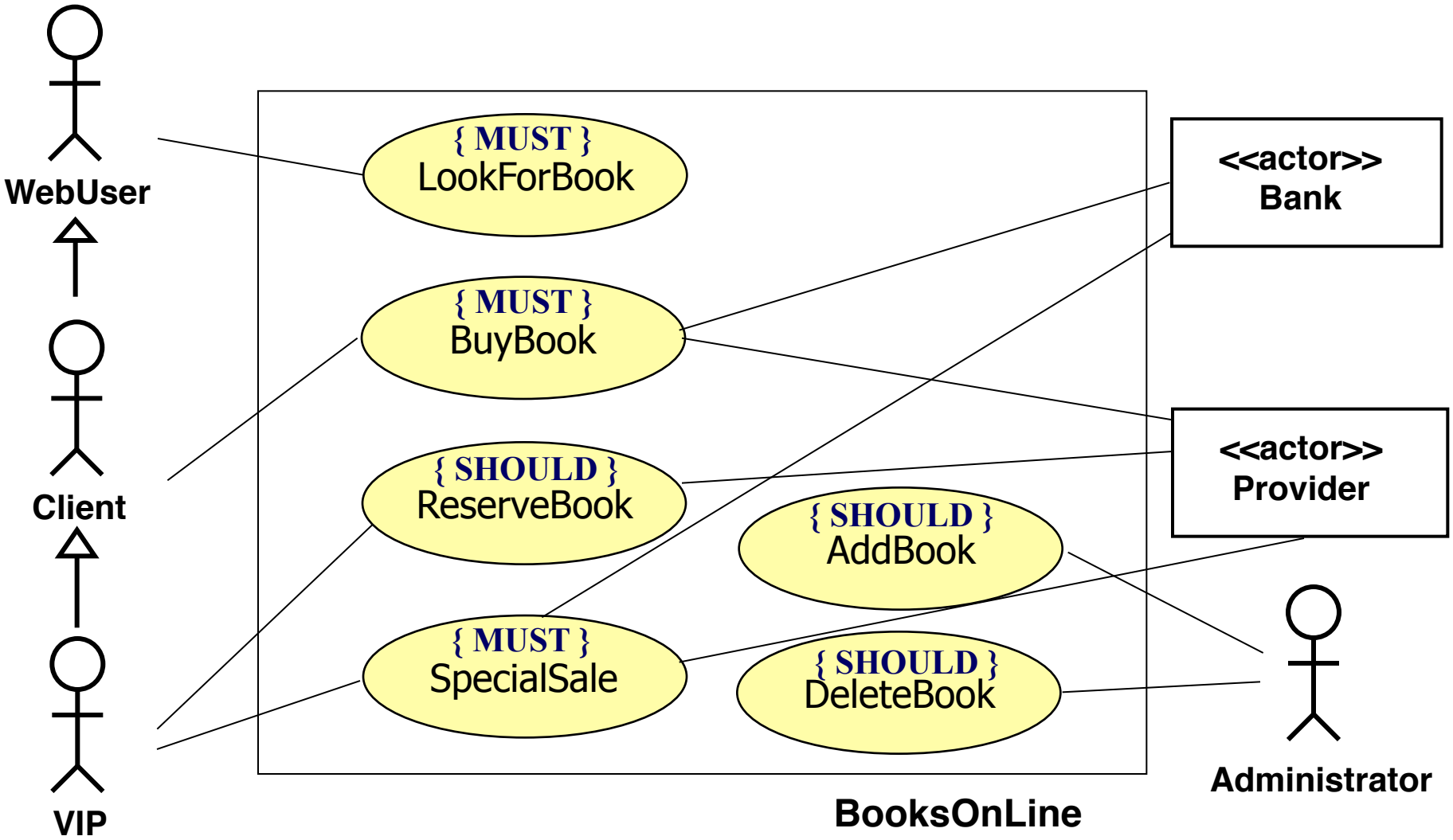
- Use case diagrams provide a global view of the functions
- It presents
  - All the actors
  - All the use cases



Example

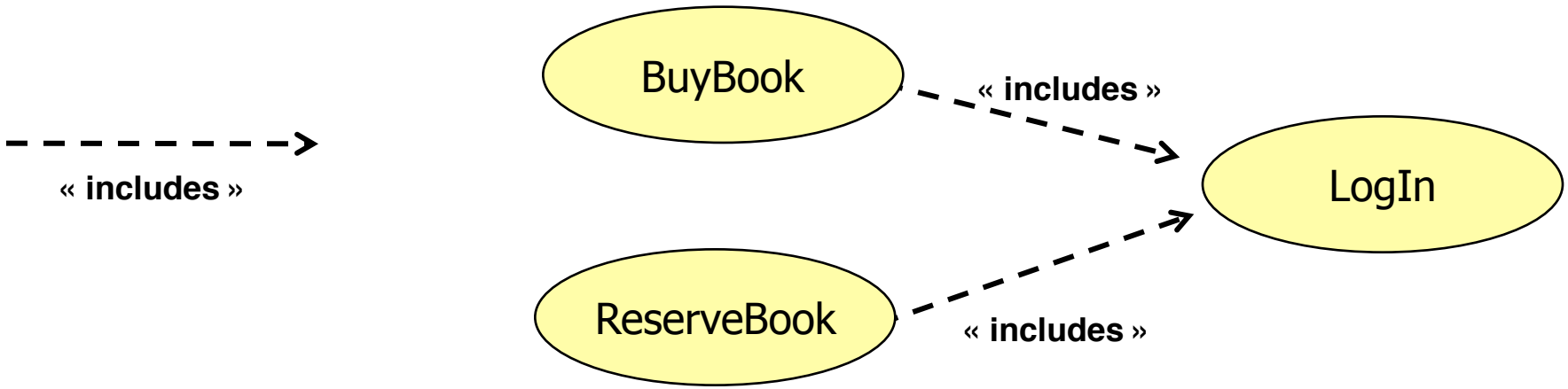

WebUser
Client
VIP
Register
LookForBook
LogIn
BuyBook
ReserveBook
SpecialSale
AddBook
DeleteBook
<<actor>>
Bank
<<actor>>
BookProvider
Administrator

Use case diagram decoration



Use cases inclusion

- For factorization purposes
  - May lead to ‘low level’ use cases



Side note

"The UML includes relationships between use cases beyond the simple « includes » such as « extends ».

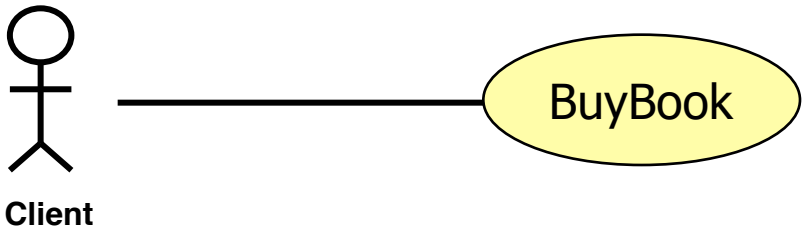
I strongly suggest that you ignore them. I've seen too many situations in which teams can get terribly hung up on when to use different use case relationships, and such energy is wasted.

Instead, concentrate on the textual description of a use case."

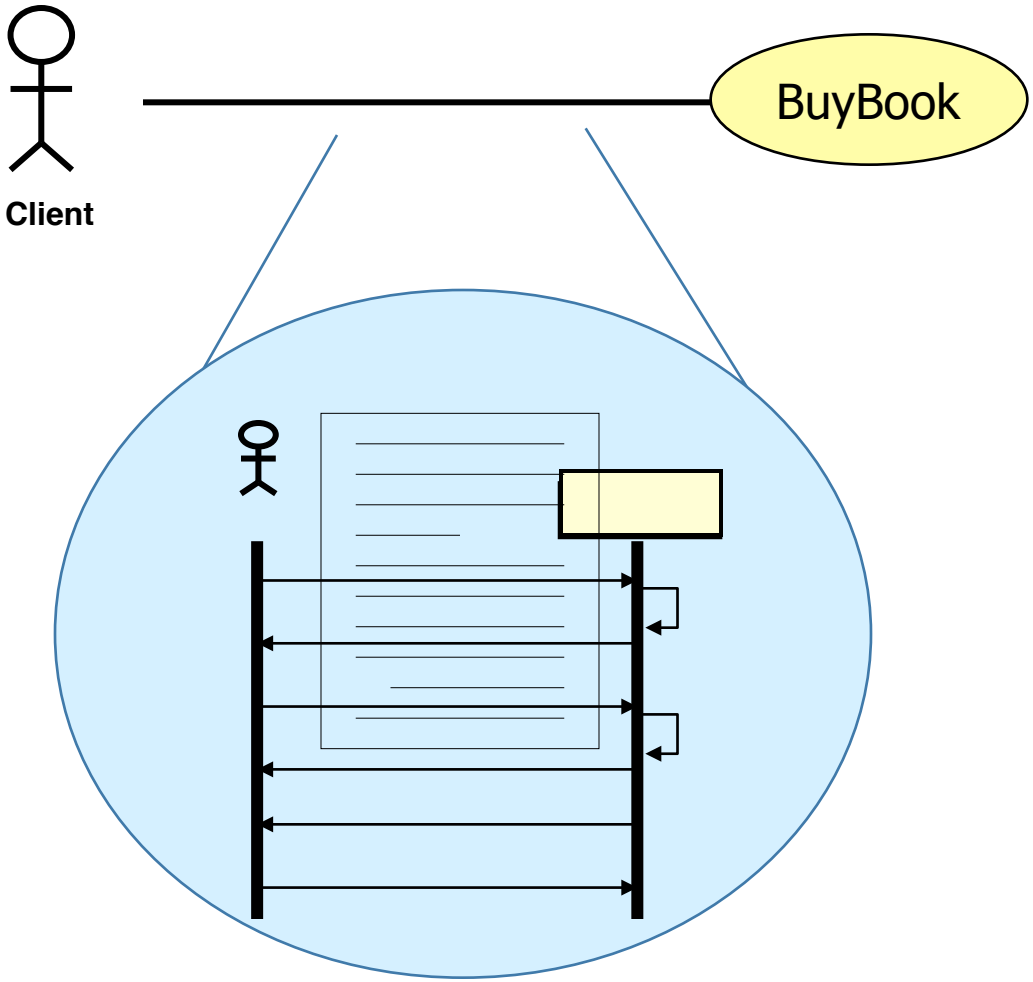
[UML Distilled, Martin Fowler]

Use case model

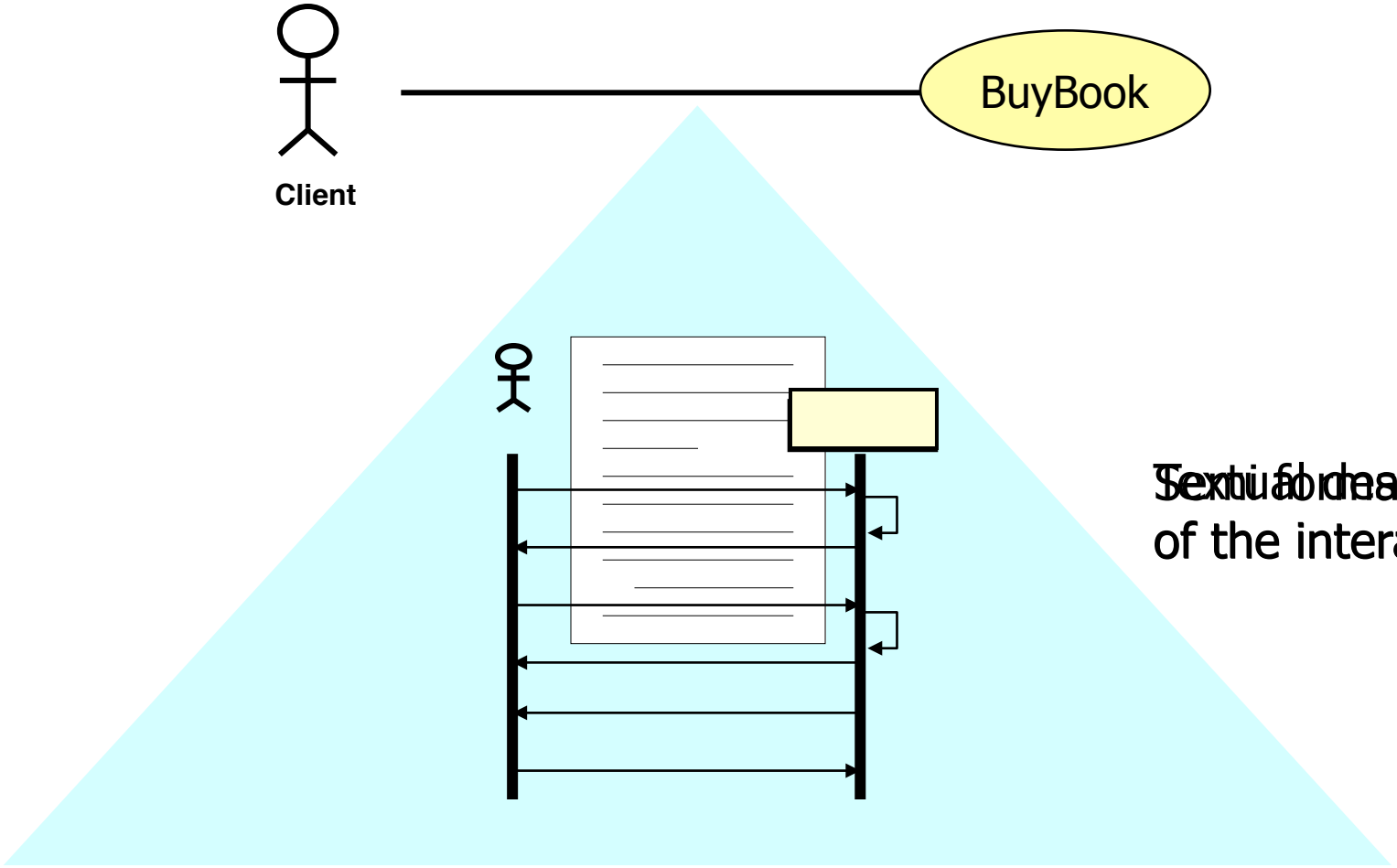
- A use case model describes a use case
  - Specifies the interaction
- Not standardized
  - Different styles
  - Different interpretations



Two ways to describe interactions



Two ways to describe interactions

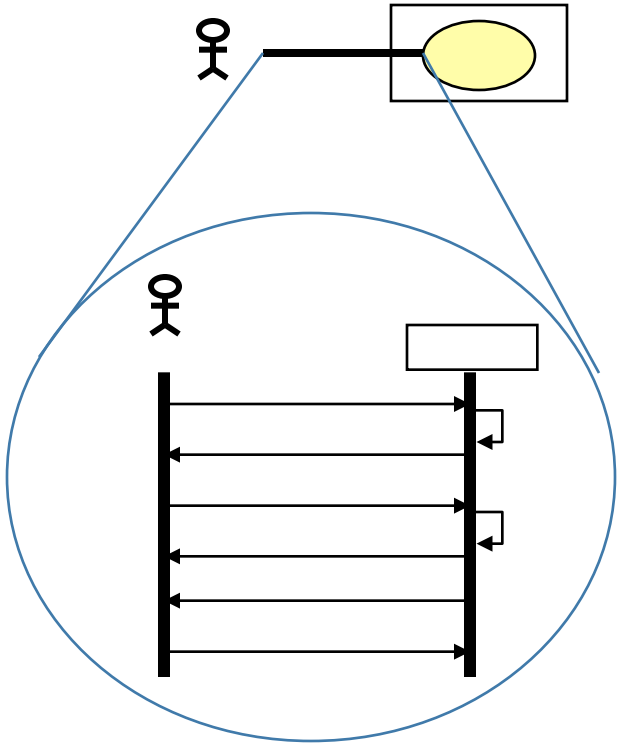


Textual description of the interactions

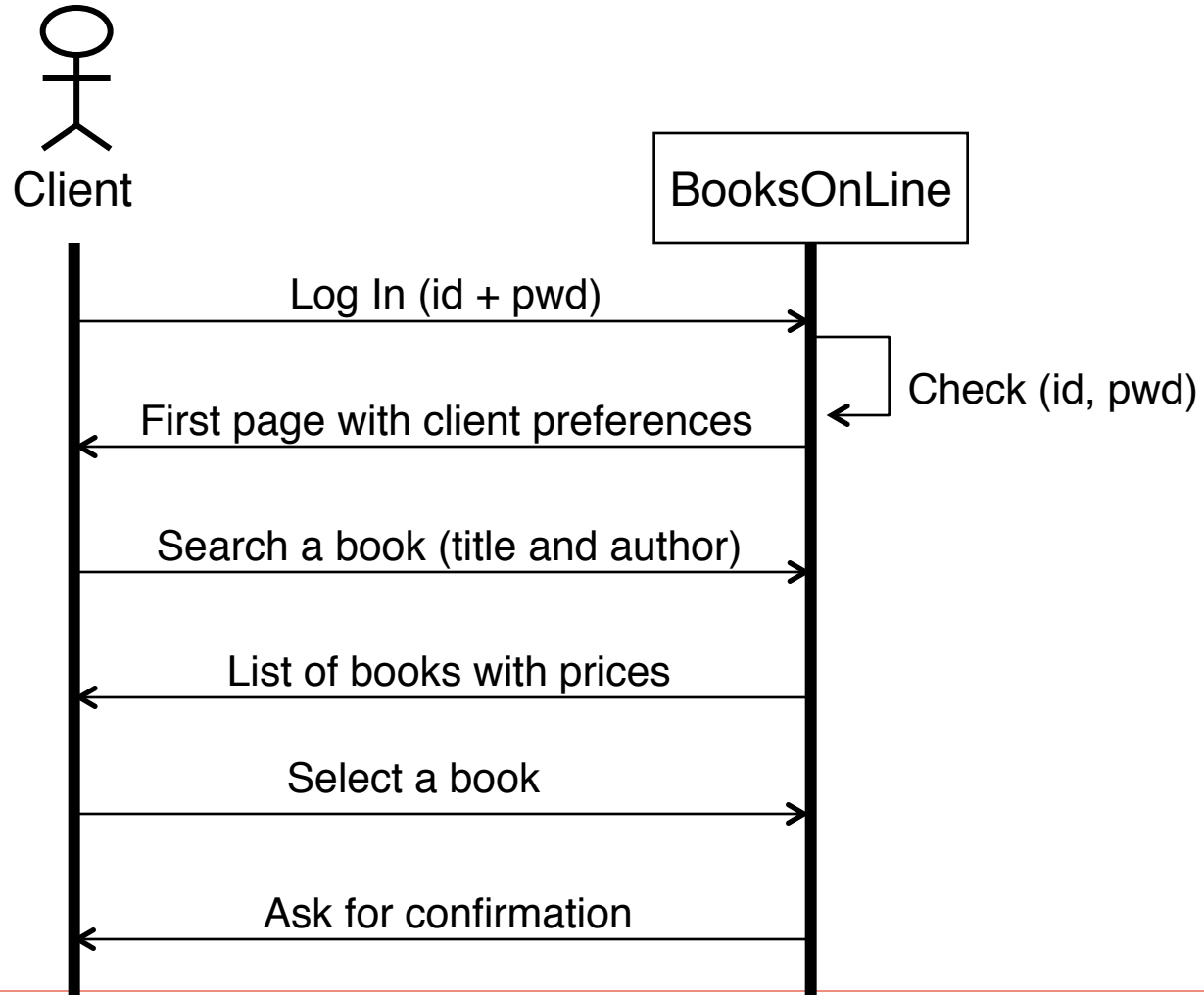
Describing a use case (1)

- A sequence diagram allows to describe a sequence of messages between different entities

- An UML notation
  - Can be used in many contexts (not invented by UML btw)
  - Different detail levels



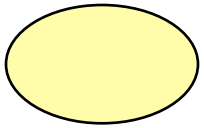
Example “book search” use case



Describing a use case (2)

- Textual form
  - UC begin, pre-conditions
  - UC end, post-conditions
  - Nominal path (behavior)
  - Possible variants and error situations
  - Interactions between system and actors
  - Exchanged information
  - Possible non functional needs

Example



BuyBook

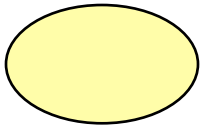
Pre-condition :
The server is running, the maximum number of visitors is not reached.

Begin : When the site is called from a web browser

End : When a confirmation is sent by email or whenever the client quits

Post-condition :
The bank account is updated.
The book provider has received an order.

Example

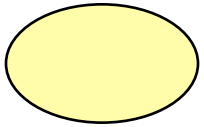


BuyBook

Nominal behavior :
(1) The *client* logs in
(2) The *system* present a first page, containing client expected preferences
(3) The *client* asks for a book (title and author)
(4) The *system* presents a list of corresponding books with the prices
(5) The *client* selects a book
(6) The *system* asks for confirmation
(7)…

Variants :
(A) Invalid Id or password : during step (1) …
(B) Invalid book : during step (3) ...

Example



BuyBook

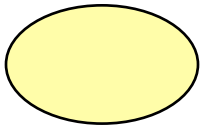
Non functional constraints :

(A) Performance : the system has to react in less than 5 seconds, whatever the operation is

(B) *Fault tolerance* : if a fault occurs during the use case, transaction has to be cancelled. Bank account not updated, no order to the books provider.
The system must be able to restart in a coherent state without human intervention.

(C) *Load* : the system must be able to manage 1000 users concurrently

Example of scenario

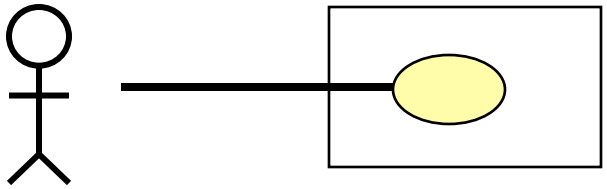


BuyBook

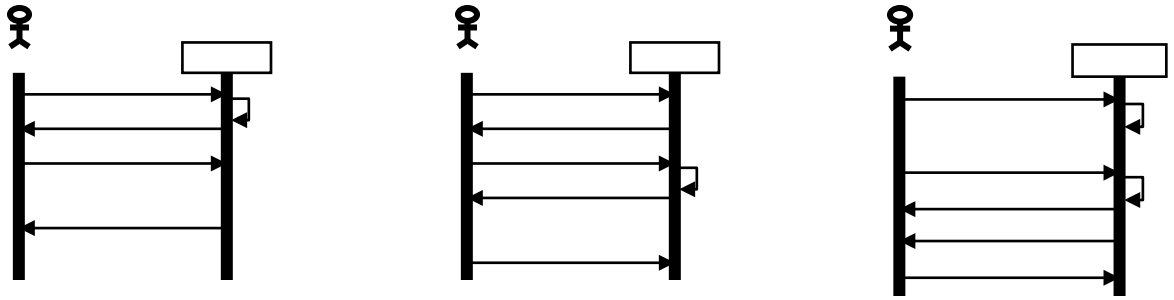
SCENARIO 4

- *Paul* logs in on Sunday morning (at 7:00)
- The *system* presents the most recent books about SciFi
- *Paul* looks for *Robots* by *Asimov*
- The *system* provides 3 books corresponding to the request
- *Paul* selects the first one (the cheapest)
- The *system* asks for confirmation
- …

Use case vs. scenarii



model



instances

Outline

- Definition
- Use cases and UML
- Unified process
- Recurring issues
- Conclusion

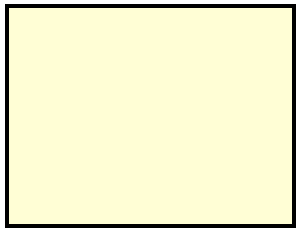
Unified process

- An accompanying process to define and organize UML activities
  - Based on UML notations

- Activities
  1. System preliminary description
  2. Define use case model
  3. Define relationships between use cases
  4. Detail each use case (considering priorities)

System preliminary description

- Give a name to the system
  - As soon as possible
  - Likely to be used all along the development
- Brief textual description (a few lines)

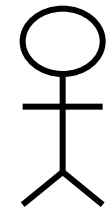


BooksOnLine

The software system BooksOnLine is a Web site which purpose is to sell books. It is specialized on novels. It gives the possibility to reserve coming books and provides special offers to its clients.

Define use cases - 1

- Find out actors and describe them
  - Select a relevant identifier revealing its role
  - BE concise and meaningful
  - No stupid name: used in many places
  - Provide a brief textual description

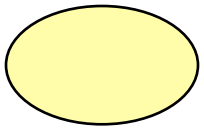


Client

A **client** is a person or an organization who has registered on the site and who has the possibility to buy one or several books.

Define use cases - 2

- Find out use cases and describe them
  - Select a relevant identifier
  - Be concise and meaningful
  - Provide a simple textual description
  - Realized function has to be understandable by everyone
  - Not too many details, concentrate on nominal scenario

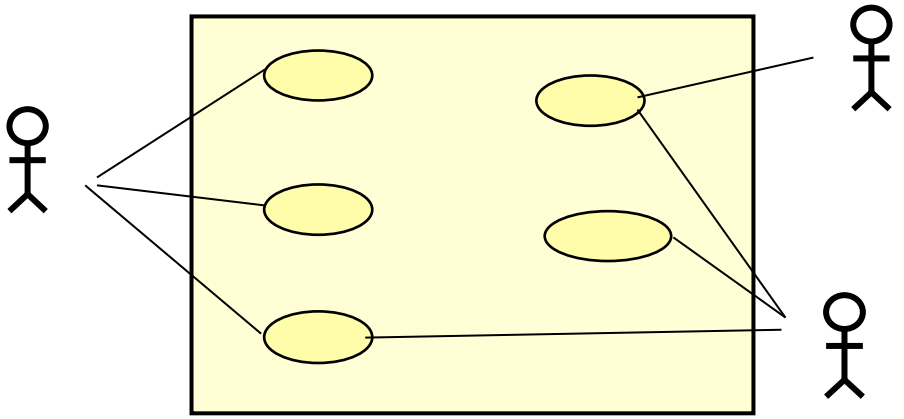


BuyBook

- *Paul* logs in on Sunday morning (at 7:00)
- The *system* presents the most recent books about SciFi
- *Paul* looks for *Robots* by *Asimov*
- The *system* provides 3 books corresponding to the request
- *Paul* selects the first one (the cheapest)
- The *system* asks for confirmation

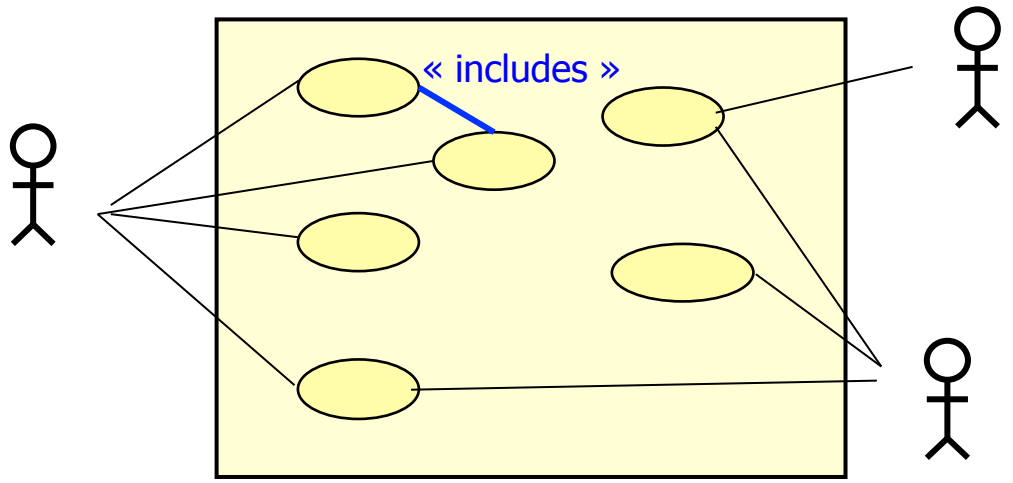
Define use case model

- Find out the right number of schemas
  - Must be coherent, easy to read
  - Define priorities
  - Decorate if needed



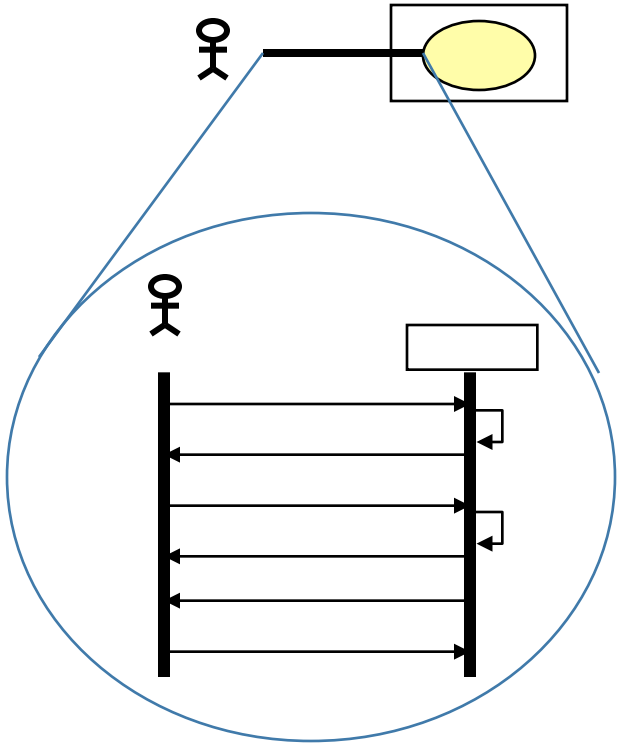
Define relationships

- A time for “optimization”
  - Additional use cases
  - Be very careful!
  - May be ignored



Describe use cases

- Textual form and/or interaction diagram
  - The very important activity
  - Followed by reviews



Outline

- Definition
- Description
- RUP
- Recurring issues (see JM Favre for more)
- Conclusion

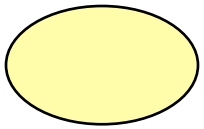
Recurring issues

- Issues tackled here are recurring in practice
  - Not solved by the standard (too poor)
  - Different interpretations and solutions
- Presented issues
  - Solution oriented use cases
  - Intermediaries issues
  - Shared used case vs. collaborative use cases
  - Get the right level of abstraction

Issue: solution-oriented use case

- Do not describe concrete interfaces
- Focus on
  - Actors objectives and intentions
  - System responsibilities
  - « abstract interactions »

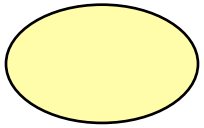
Re-writing in a use case style (1)



BuyBook

- the *client* enters his/her name and his/her password
- the *system* verifies the name/password
- …

Focus on "analysis" information



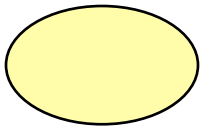
BuyBook

- the *client* identifies himself/herself
- the *system* checks the identification
- …

Issue: Natural language

- Use cases descriptions are based on structured natural language
  - Ambiguity
  - Incompleteness
  - Imprecision
  - …

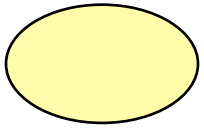
Re-writing in a use case style (2)



BuyBook

- the *client* enters his/her name and his/her password
- the *system* verifies the name/password
- the *client* enters the references of the desired book
- …

Be precise

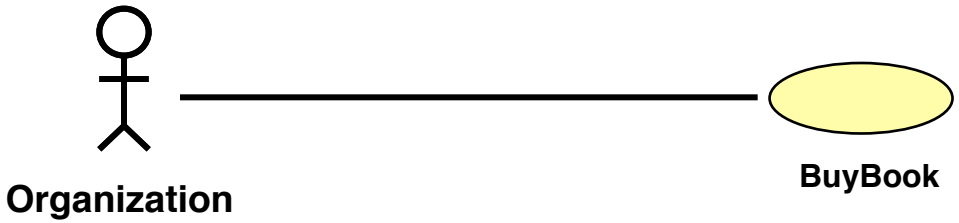


BuyBook

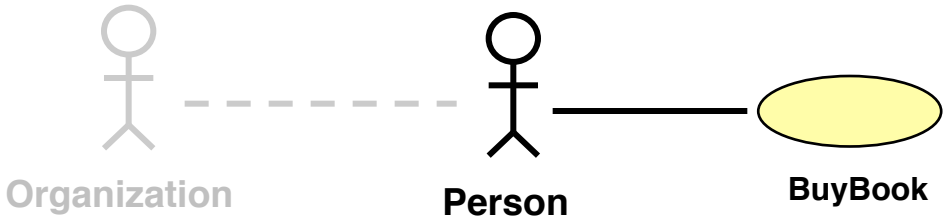
- the *client* identifies himself/herself
- the *system* checks the identification
- the *client* enters the title and authors of the desired book
- …

Issue: intermediaries issues

- Should intermediaries between system and main actors be represented ?
- Different point of views



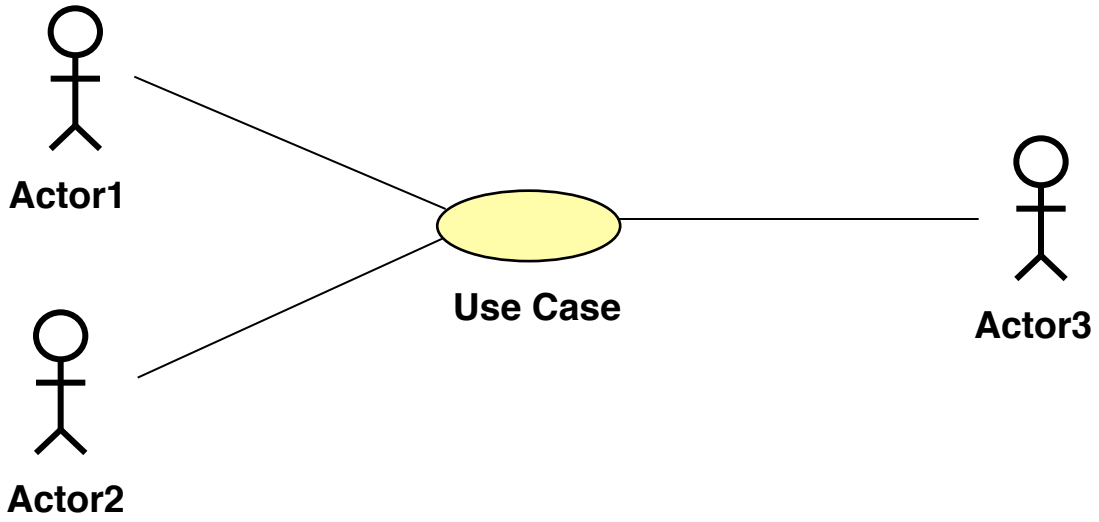
The focus is on the objectives (interface related aspects are kept hidden)



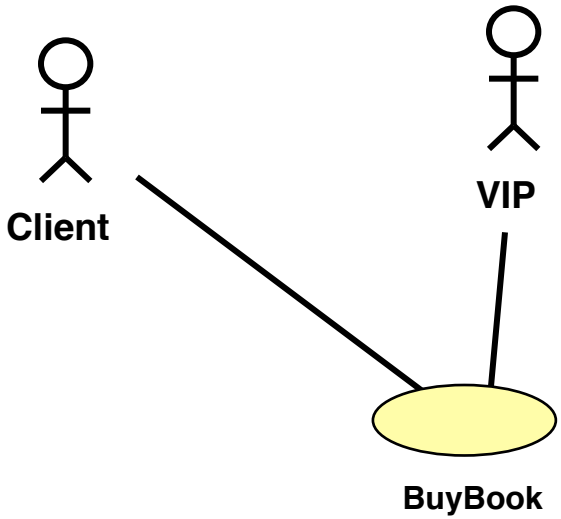
The focus is on the comm. link (message exchanges and interface)

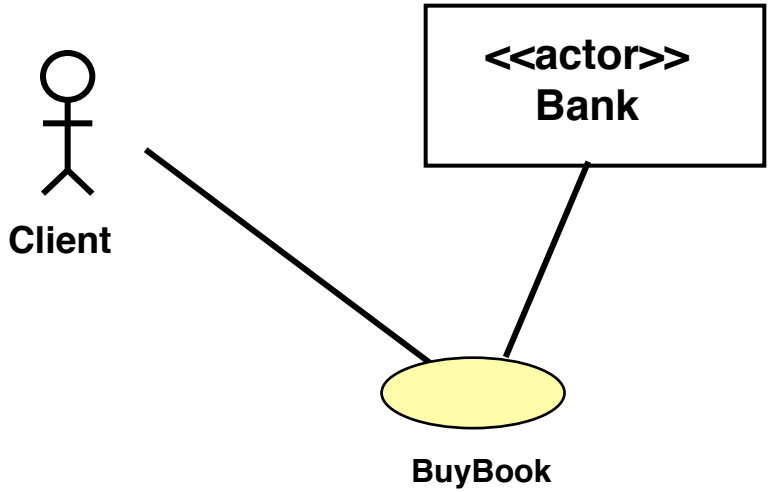
Issue: A non informative notation

- Confusion when there are several actors
  - who realizes the use case ?
  - Who collaborates ?
  - Which actors collaborate on a single use case ?



One notation - two interpretations





(1) « Shared » use case

Two actors can perform the use case to meet their own objectives

(2) « Collaborative » use case

Two actors work together to meet a common objective. The system interacts with both of them.

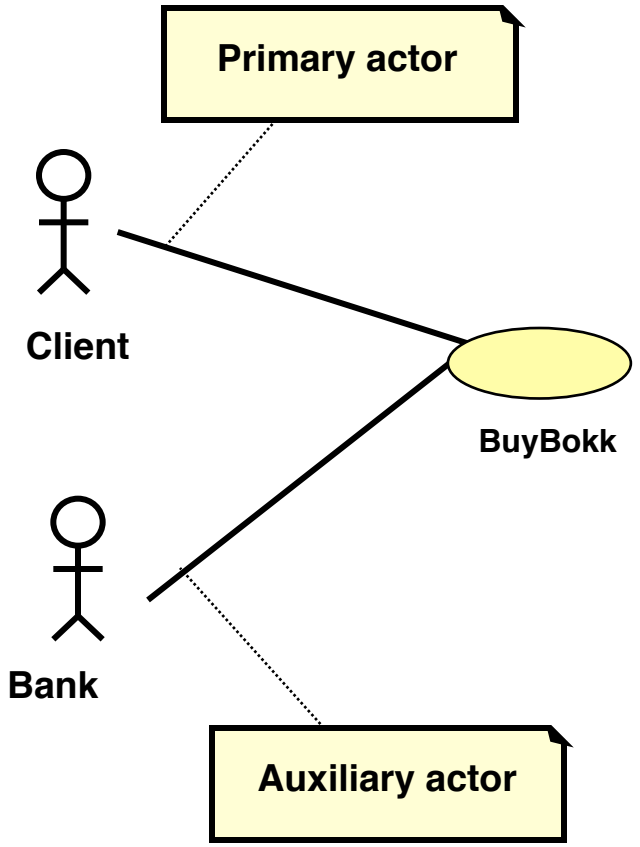
Issue: Collaborative use cases

« Primary » actor

- Uses the system to meet a goal
- Generally starts the interaction

« Auxiliary » actors

- Comes after the primary actor
- Generally provides services to the system



Different styles in practice

- « primary » style
  Just represent primary actors in diagrams
- « decorated » style
  Use a specific decoration (e.g. auxiliary or initiator)
- « left / right » style
  primary actors are on the left, auxiliary on the right
- « arrow-based » style
  Use arrows to indicate primary actors (to be avoided)

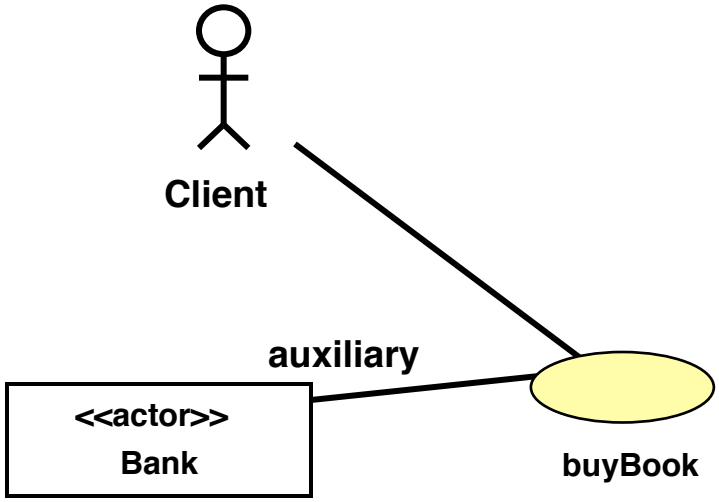
« primary » style

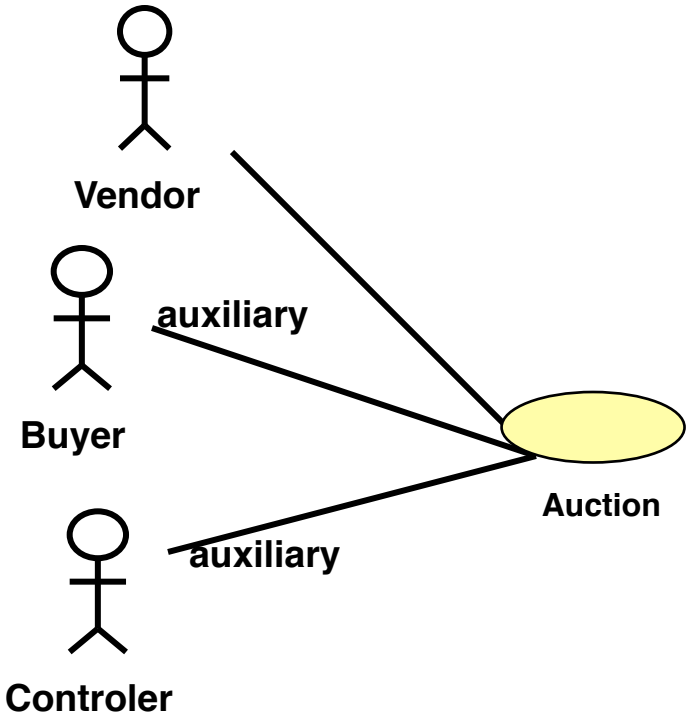
- Just represent the primary actor

- Advantages: simple, readable
- Best solution during first iterations

« decoration » style

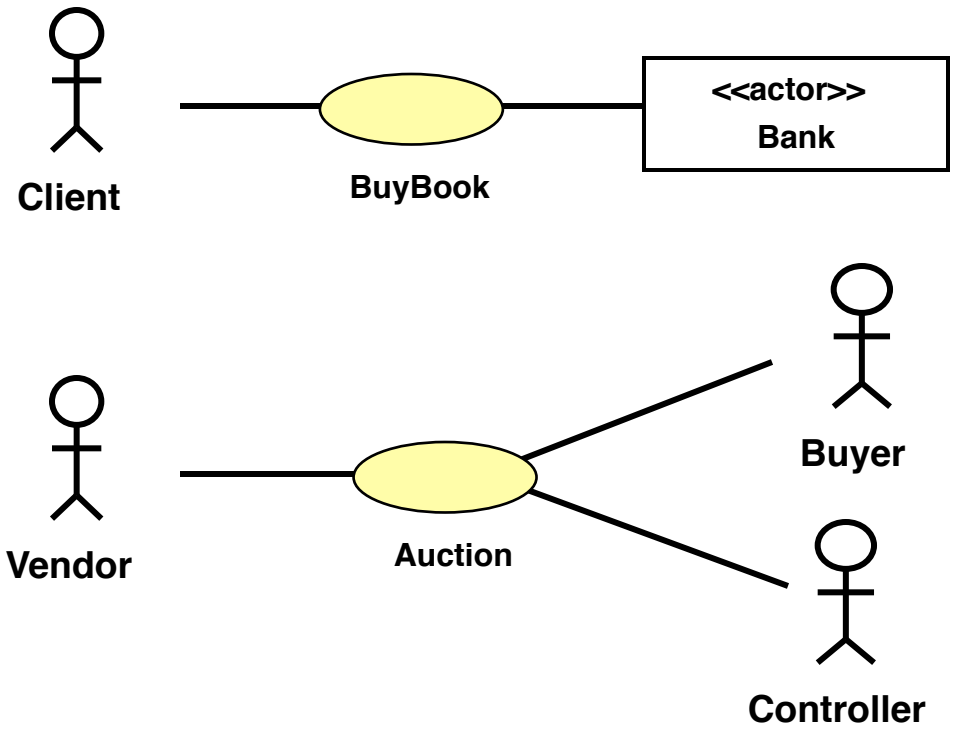
- Use a specific decoration (e.g. auxiliary or initiator)





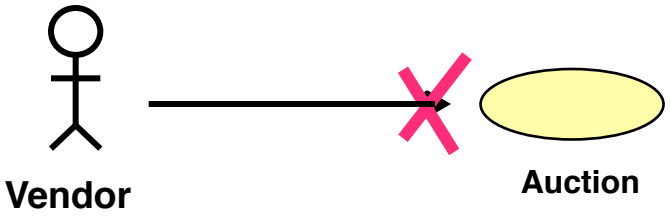
« Left / right » style

- Primary on the left, secondary on the right
  - Convention is not explicit



Avoid arrows !

- Avoid arrows in UML:
  - No clear meaning



- Diverse interpretations:
  - « the actor is the initiator »
  - « communication goes one way only »
  - « could not get rid of this arrow in the tool ! »

Issue: abstraction level

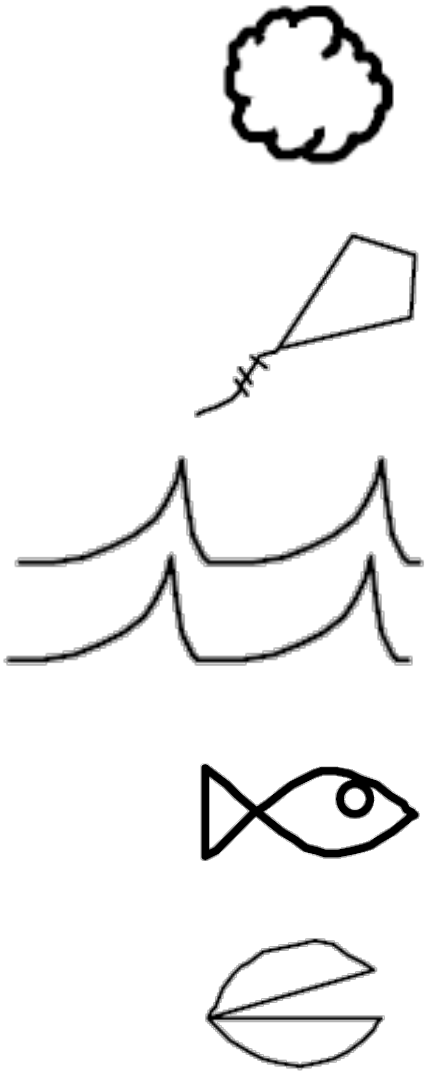
- At what abstraction level should use cases be described ?
  - Good question ... but no easy answer
- Too high level
  - Too far from the system
  - Too abstract and fuzzy
- Too low level
  - Too many use cases
  - Too close to the interface
  - Too far from business needs

Issue: abstraction level

- No silver bullet!
  - But always try to characterize the level of abstraction of your use case
- Abstract use cases may be necessary
  - But justify it!
- Low level use cases may be necessary
  - But justify it!

Issue: abstraction level

- All use cases don't have to be at the same abstraction level
- Different detail (abstraction) levels
- Cockburn style: non standard but useful and intuitive



- Most use cases should be at the sea level

Abstraction level

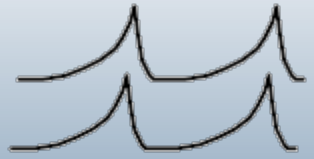
Clouds Level

- Too high

Kite Level

- Synthetic level : describe a grouping related to a more global objective

Sea Level

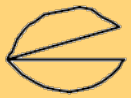


- **Normal level** : describe an actor's goal which can be met through interaction with the system

Fish Level

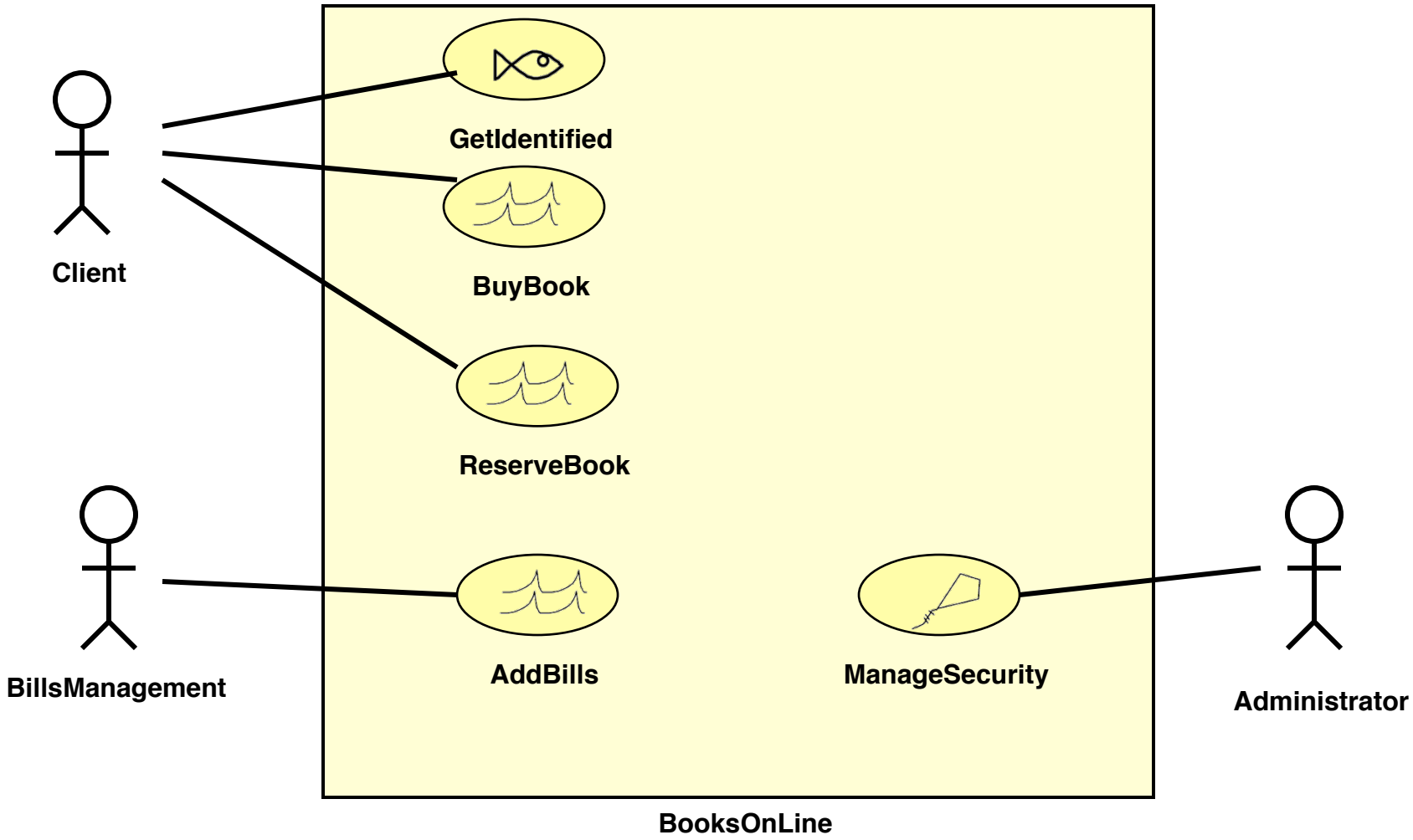
- Detailed level : describe an interaction with the system, not a goal

Clam Level



- Too low

Example


GetIdentified
Client
BuyBook
ReserveBook
BillsManagement
AddBills
ManageSecurity
Administrator
BooksOnLine

Outline

- Definition
- Description
- Detailed description
- Recurring issues
- Conclusion

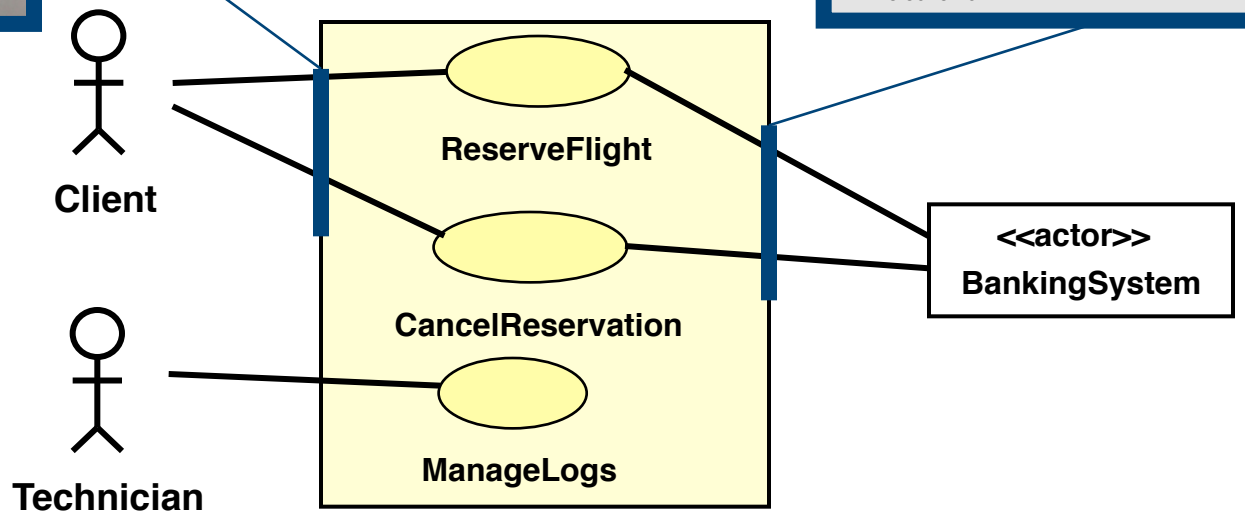
Use cases popularity

- Use cases are today very popular
  - User centered
  - Informal
  - Communication based
  - Incremental
- But
  - Hard to find the right set
  - Not enough for complete specification

Use cases and requirements (1)

- Use cases define the system frontiers

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="XMLSchema">
   <xs:element name="note">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="to" type="xs:string"/>
            <xs:element name="from" type="xs:string"/>
            <xs:element name="heading" type="xs:string"/>
            <xs:element name="body" type="xs:string"/>
         </xs:sequence>
      </xs:complexType>
   </xs:element>
</xs:schema>
```



Side note - 2

"A common sign of a novice (or academic) use case modeler is a preoccupation with use case diagrams and use case relationships, rather than writing text. ... Use case diagrams and use case relationships are secondary in use case work. Use cases are text documents. Doing use case work means to write text."

[Applying UML and Patterns, Craig Larman]

Conclusion

- Keep it simple!

- *“A big danger of use cases is that people make them too complicated and get stuck. Usually you'll get less hurt by doing too little than by doing too much*".

[UML Distilled, Martin Fowler]