Software Engineering
Lecture 3 – introduction to UML, Use-Case diagrams User stories

Bogumiła Hnatkowska

Lecture objectives

- Introduction to the UML
- Presentation of the use-case model:
  - Use-case diagrams
  - Use-case specifications
- Presentation of User stories

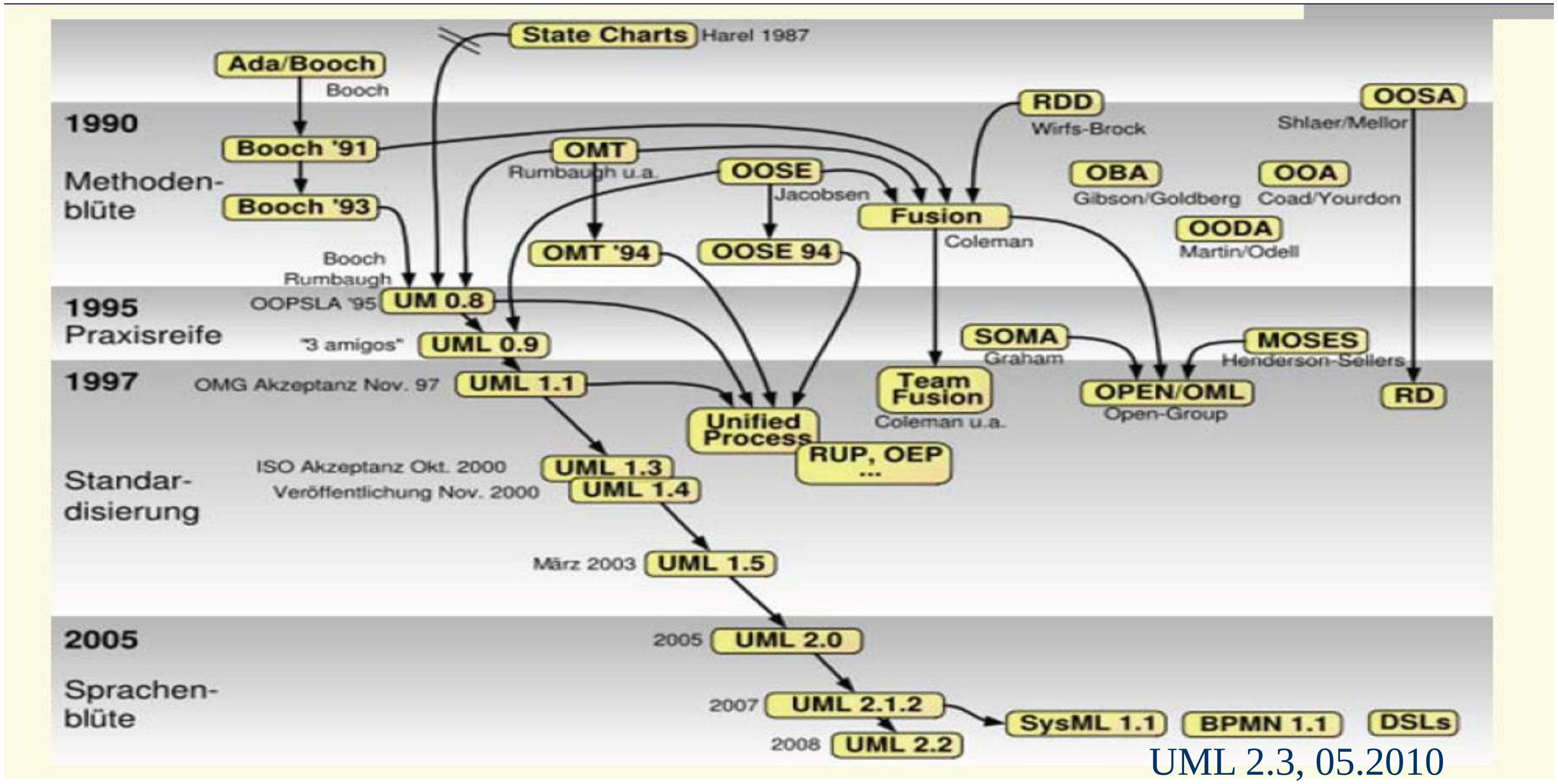
What is a model?

- A model – abstraction of something (exiting in a real word or artificially designed). Focuses on a particular view: static or dynamic.
- In SE models are used for: specification, visualization, documentation of:
  - Application domain
  - Developed system

UML - introduction

- UML (Unified Modeling Language) – commonly accepted graphical modeling language (notation) used for system: visualization, specification, documentation
- Promoted by Object Management Group (www.omg.org)
- UML definition:
  - abstract syntax - meta-diagrams
  - well-formedness rules - expressed in OCL
  - semantics - described in English prose

UML - genesis

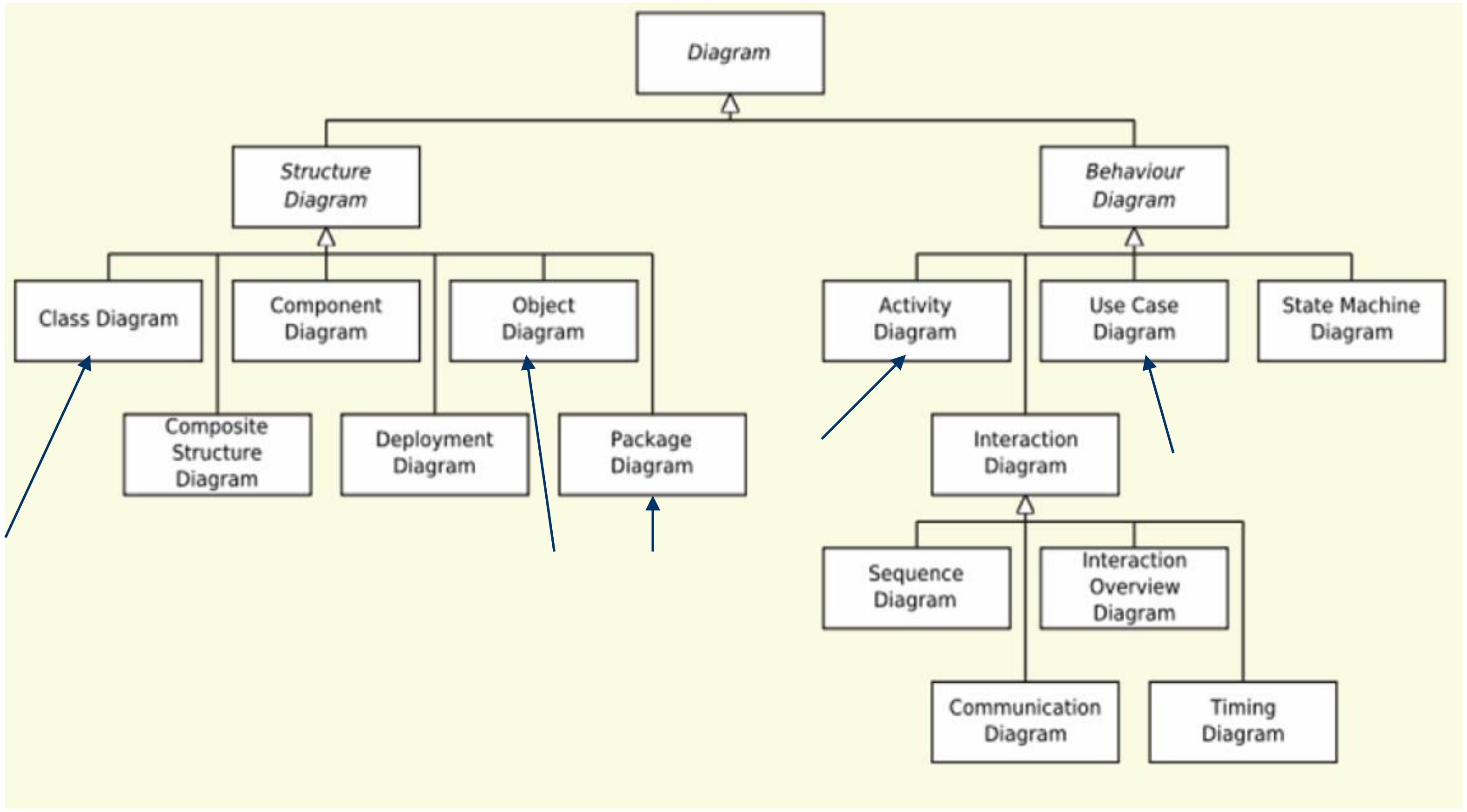


UML 2.3, 05.2010
UML 2.4.1, 08.2011
UML 2.5, 2015

Model in UML

- Model in UML – a consistent set of diagrams representing structural and/or behavior aspect of a modeled entity
- Pareto law:
  - 20% of the UML is enough to cope with 80% of software projects

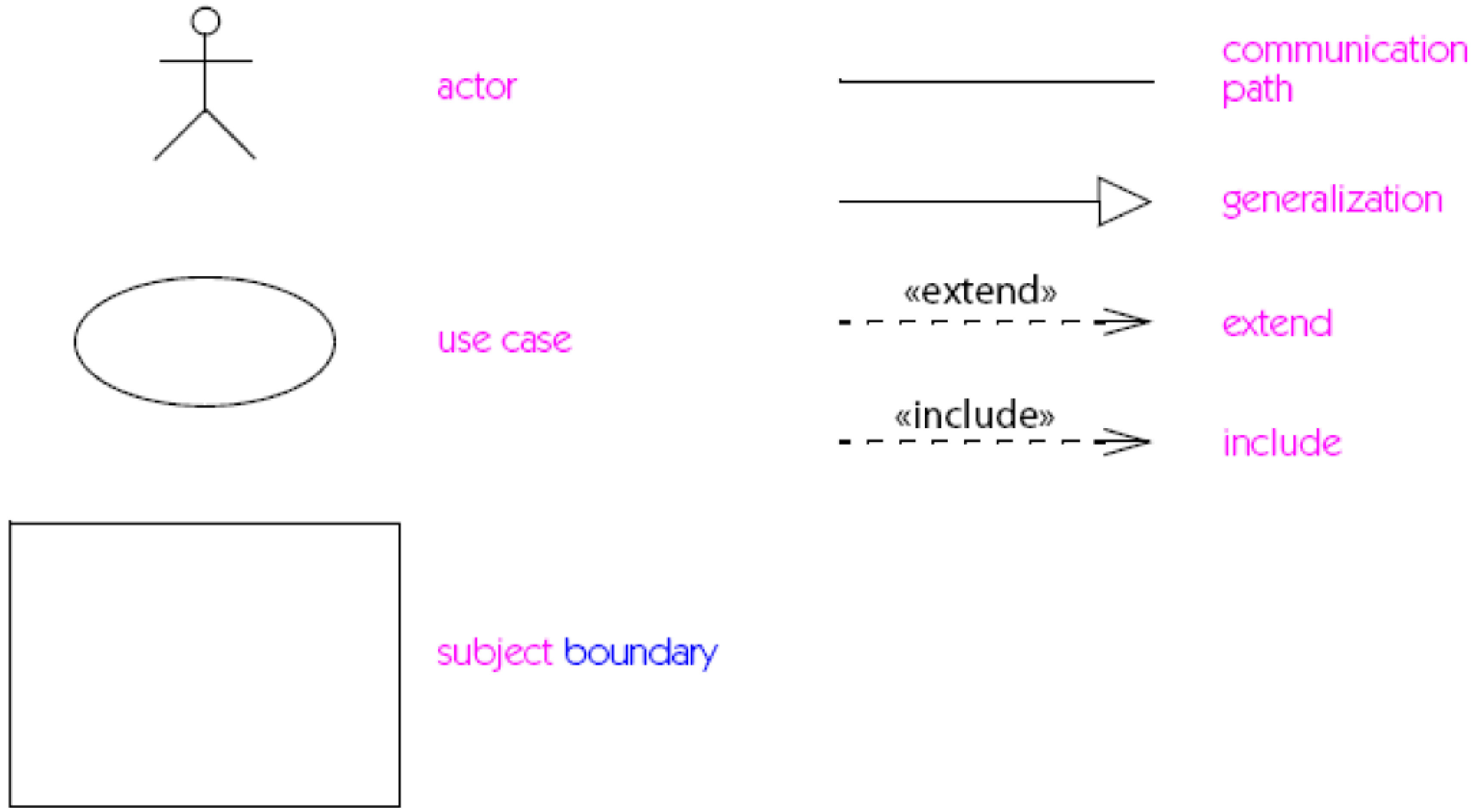
UML diagrams



Use-Case diagrams

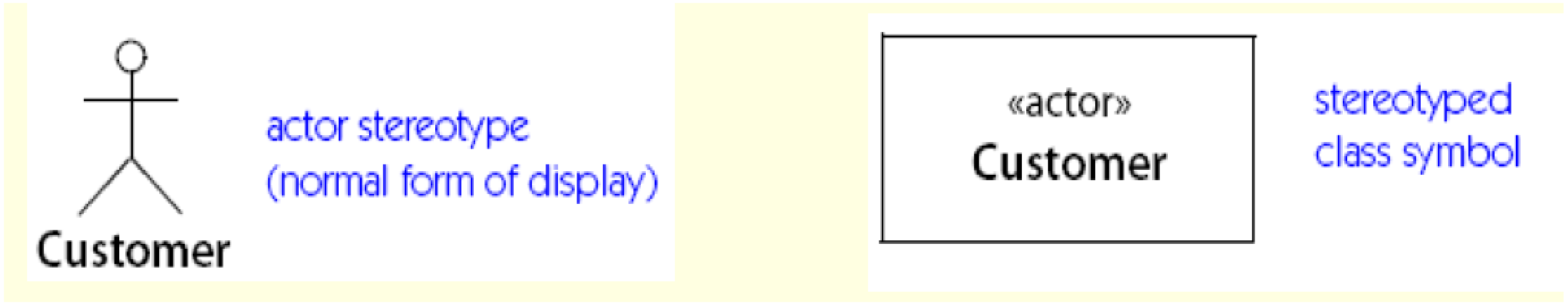
- Used for presentation of functional requirements (usually as a part of requirement specification document)
- A graph consisting of:
  - Nodes: actors, use-cases
  - Arcs: associations, generalizations, dependencies
- Additional elements:
  - subject boundary (can be a system, a system component or a complex class)

Use-Case diagrams, cont.



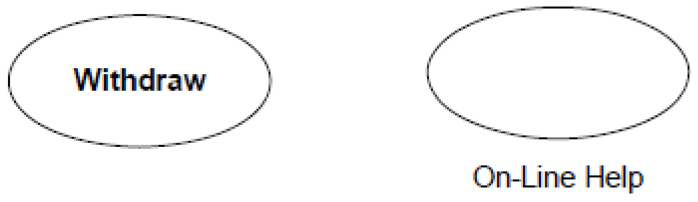
Use-case diagram, Actor

- Actor – something (a person, external hardware or another system) outside the modeled system that interacts with it
- An actor models a type of role played by an entity that interacts with the system.



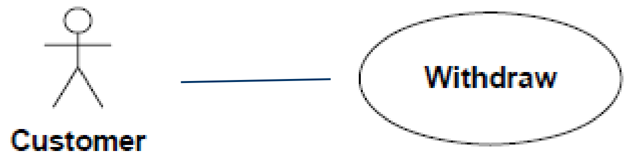
Use-case diagrams, Use-Case

- Use-Case – a service offered by the system to its actors, providing a measurable value
- The specification of sequences of actions (and/or messages exchanged), including variant sequences and error sequences, that a system (subsystem) performs by interacting with actors to provide a service
- Use-cases can be instantiated
- An instance of a use-case is a concrete sequence of actions a given actor performs when operating with a system

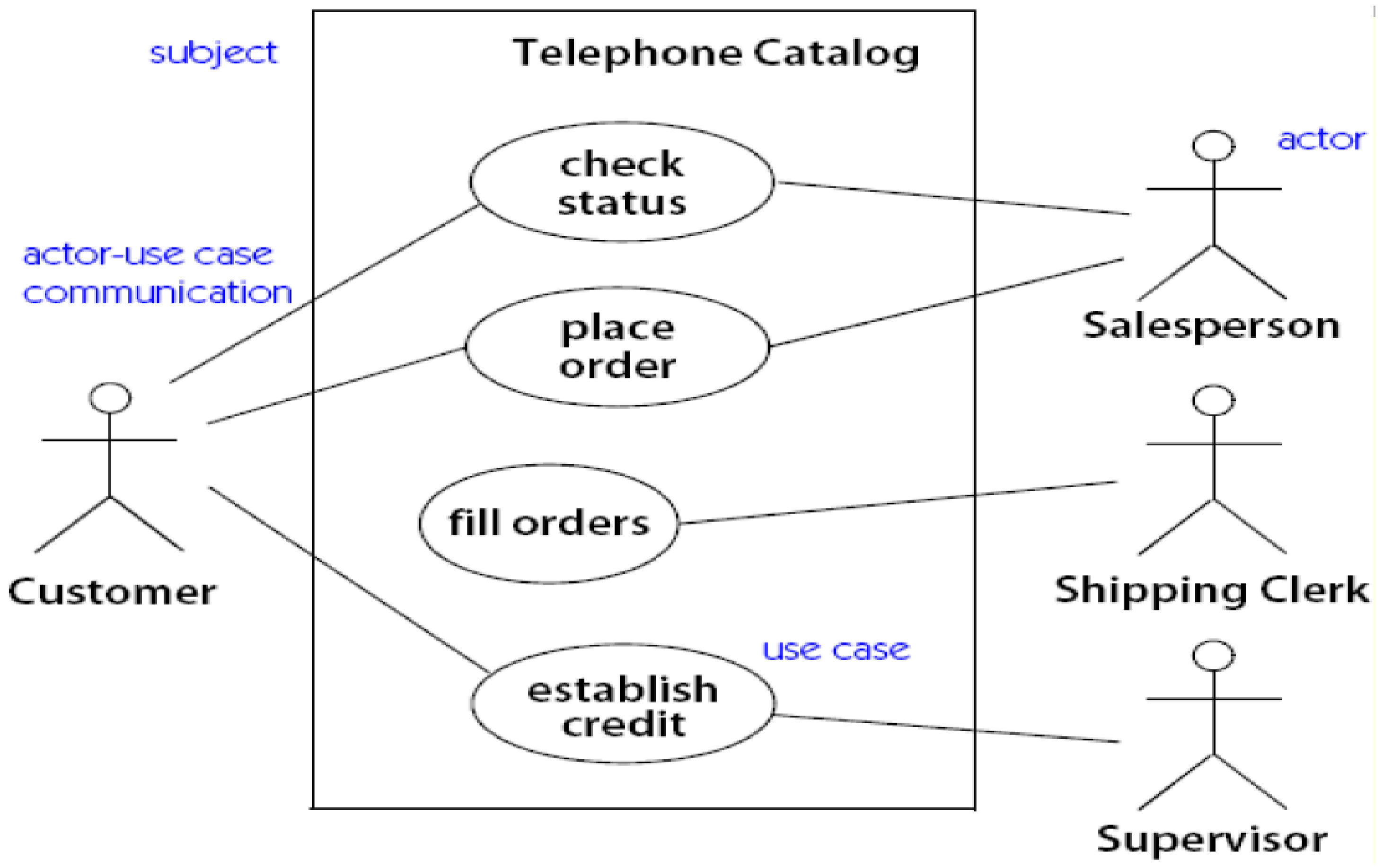


Use-Case diagrams, cont.

- An actor is linked to an use-case with **association** (represented by a line; it is a commonly used UML symbol)
- An actor is understood as a set of coherent roles
- An actor plays a given role in a given use-case, e.g. the person filling orders
- When **two** different actors are associated with a given use-case that means **that they play different roles**

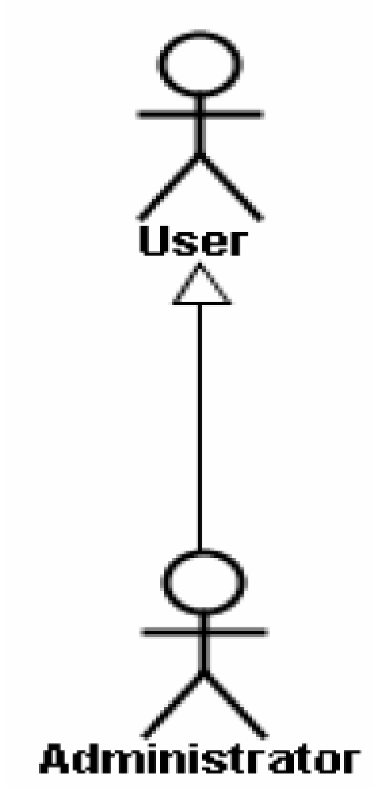


Use-case diagrams, example

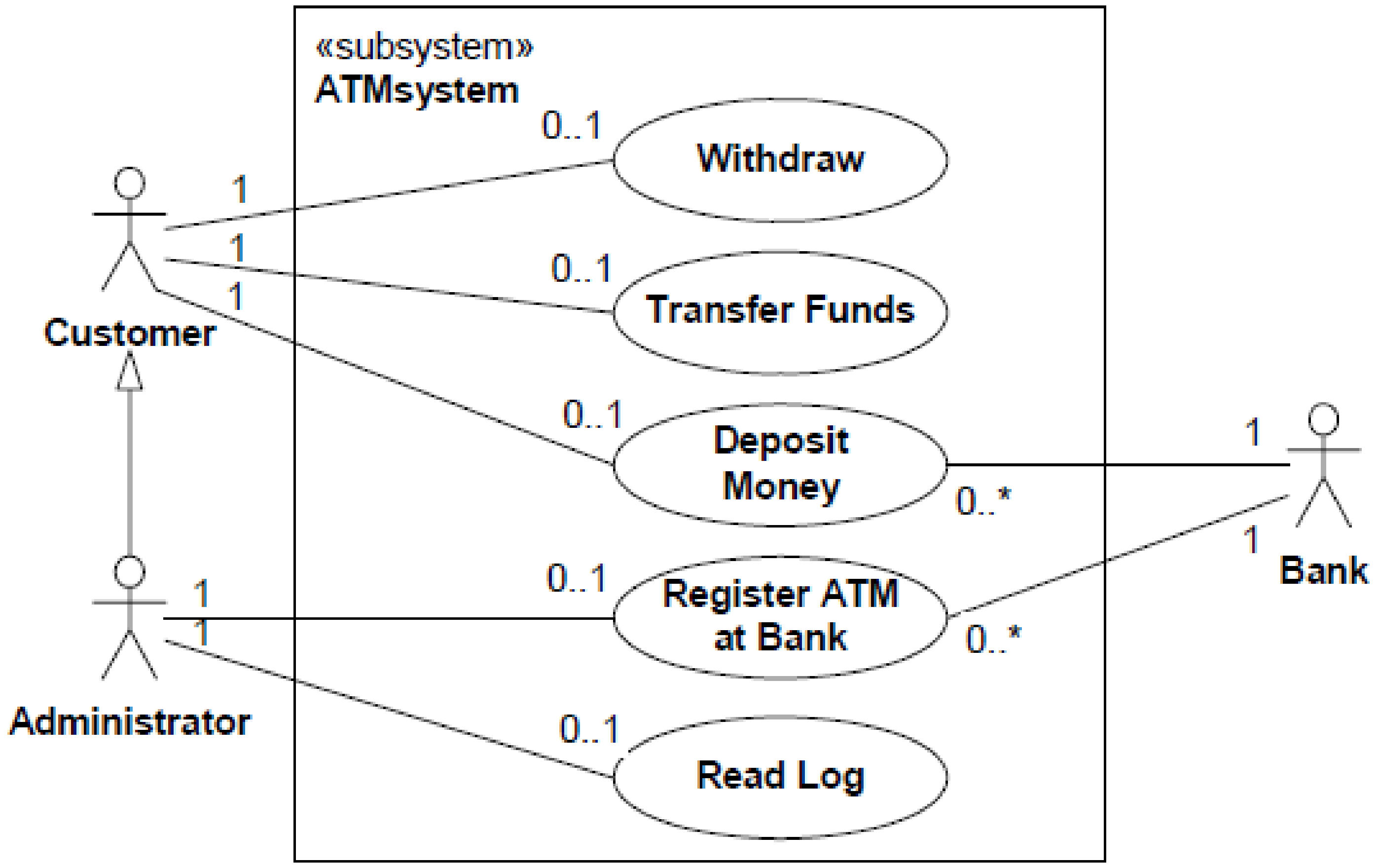


Relationships between actors

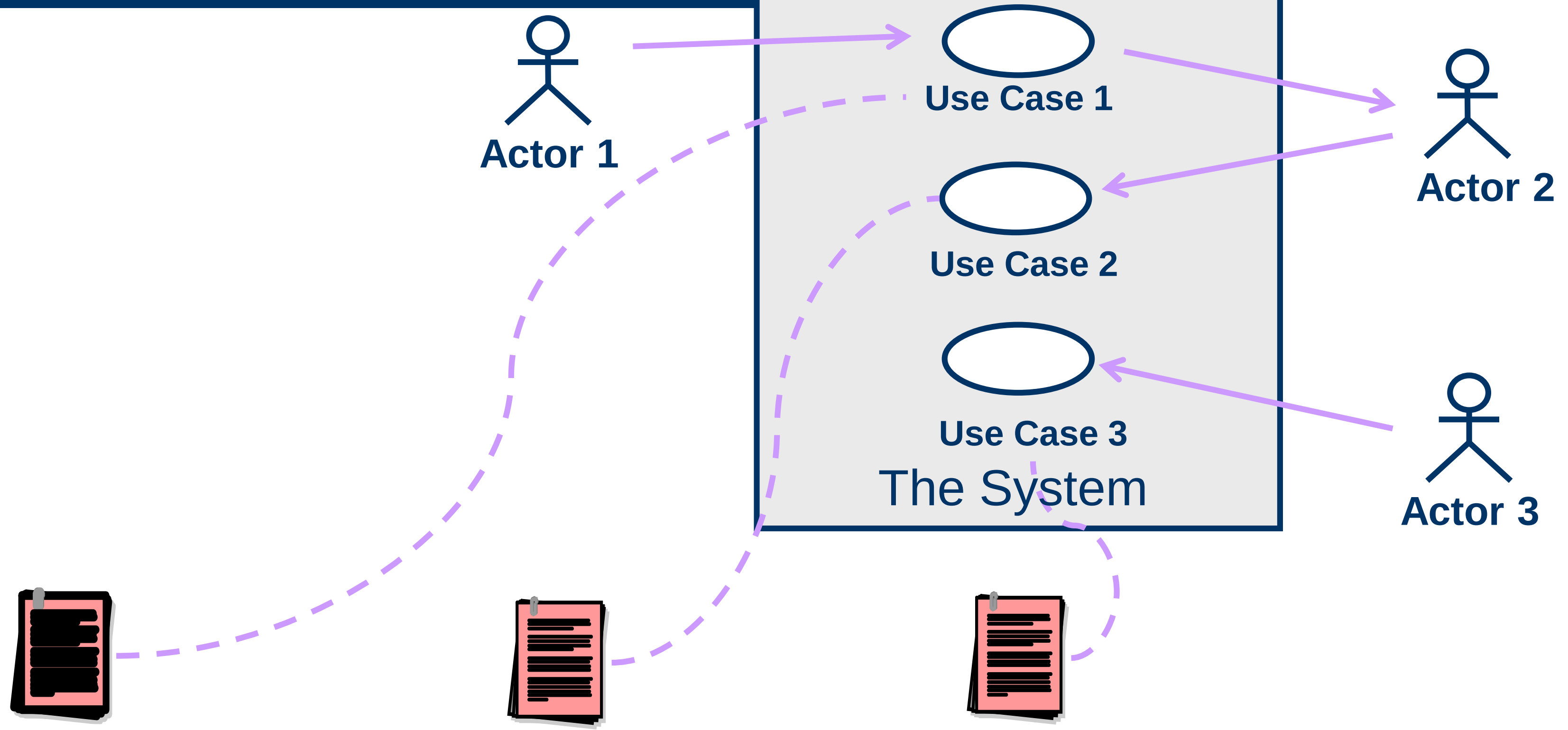
- There is possible to define inheritance relationships between actors
- The child-actor inherits all properties (e.g. associations with use-cases) from its parent-actor



Relationships between actors, cont. Multiplicities



The Use-Case Model



Use-Case 1 Spec
- Brief description
- Flows of events

Use Case 2 Spec
- Brief description
- Flows of events

Use-Case 3 Spec
- Brief description
- Flows of events

How to define use-cases?

- Textual descriptions in natural language – use case specifications
- Activity diagrams – see next lecture
- Sequence diagrams – out of scope of this lecture
- State machines – out of scope of this lecture

Use-case specification

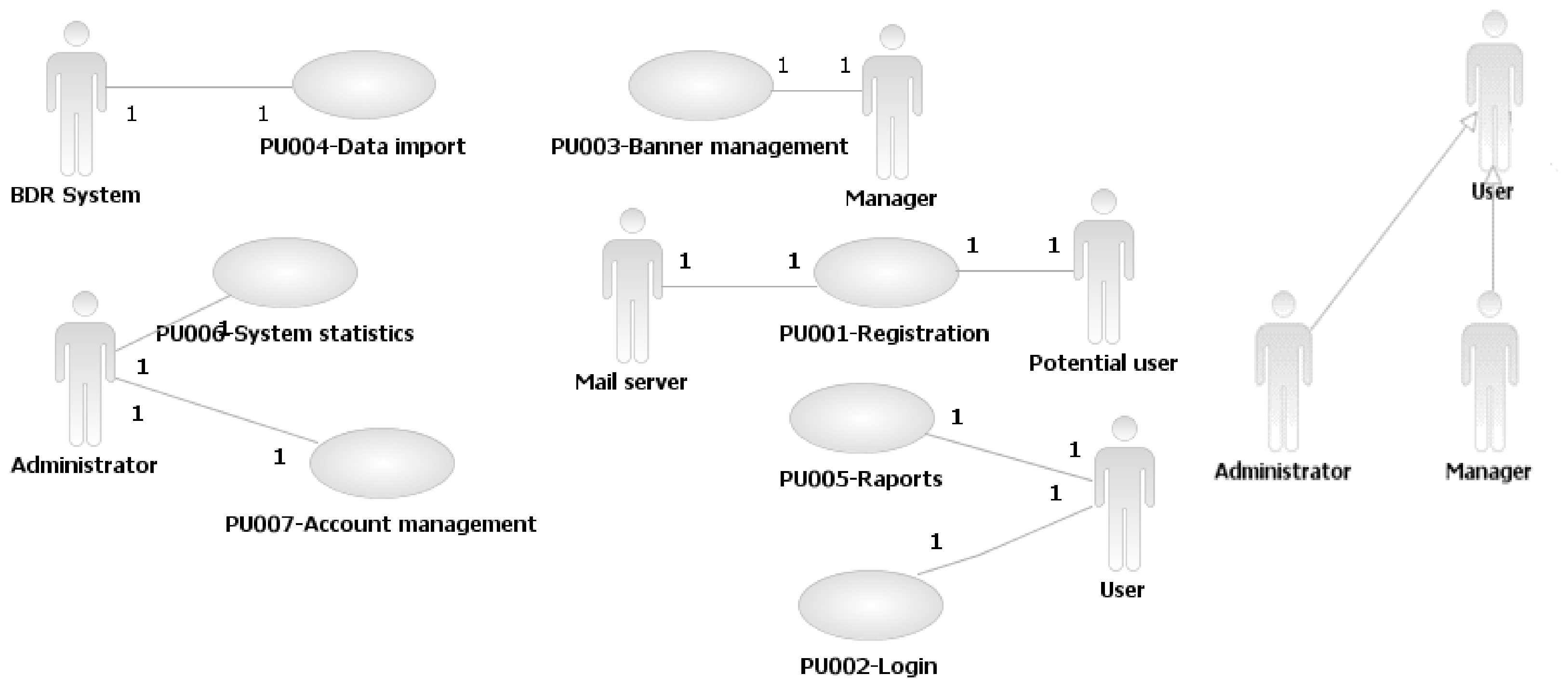
Use case id/name
Brief description
[trigger]
[pre-conditions]
[post-conditions]
Basic Flow of actions

1. A: First step
2. S: Second step
3. A: Third step

A1 Alternative flow 1
A2 Alternative flow 2
A3 Alternative flow 3
A4 Exceptional flow 1

Aktor	System
1. First step	
	2. Second step

Use-case model example 1 – information system



Use-case model example 1, cont.

- UC id UC001, Registration
- Brief description - Use case enables a potential user to register in the system. After the user provides personal data, system creates a new non-active user account and sends an e-mail to the user. User reads the mail and activates the account.

Use-case model example 1, cont.

- Pre conditions: The account with the same e-mail address doesn’t exist.
- Post conditions: A new user account is created and activated.
- Basic flow:

1. User wants to register to the system.
2. System asks about ***personal data****.*
3. User provides ***personal data***.
4. User asks for registration.
5. System validates personal data and creates a new non-active user account.
6. System prepares an e-mail and sends it to the user by mail server.
7. User reads the e-mail and activates the account.
8. System validates that user account exists, activates it and enables login to the system

Alternative flow 1 - Personal data are invalid

5a. System validates that personal data are improper. System informs about that the user. Goto step 2.

Exceptional flow – the non-active account doesn’t exists

8a. System validates that account doesn’t exists and informs about it the user.

- Personal data
  - First name: String[30], obligatory
  - Surname: String[30], obligatory
  - Birthdate: date, obligatory, a user has to have at least 18 years
  - E-mail: String[50], optional
  - Password: String[10], obligatory, at least 3 letters, at least 1 number, at least 1 special character (#,$,%,^,&,*)
  - Password repetition: String[10], obligatory, must be the same as password

Use Cases as stories

- Use-cases are easy to read, it does not mean easy to write.
- Usually written by engineers. Different styles and techniques.
- Need a mechanism to capture good writing practices – patterns (templates)

Use Cases as stories, cont.

- Book: Alistair Cockburn, **Writing effective use-cases**, Addison-Wesley, 2001
- Book: Steve Adolph, Paul Bramble, Alistar Cockburn, Andy Pols: **Patterns for Effective Use Cases (The Agile Software Development Series)**, Addison-Wesley Professional, 2002

Writing effective use-cases

- Guidelines for an action step:
  - Guideline 1: It uses simple grammar

    Subject ... verb... direct object... prepositional phrase.

    Example:

    The system ... deducts ... the amount ... from the account balance.
  - Guideline 2: It shows clearly, "Who has the ball”
  - Guideline 3: It is written from a bird's eye point of view

    Customer: puts in the ATM card and PIN.

    System: deducts the amount from the account balance.

Writing effective use-cases, cont.

- Guidelines for an action step
  - Guideline 4: It shows the process moving distinctly forward
  - Guideline 5: It shows the actor’s intent, not movements (user interface details are omitted)

    Before (wrong style):
    1. System asks for name.
    2. User enters name.
    3. System prompts for address.
    4. User enters address.
    5. User clicks ’OK’.
    6. System presents user’s profile.

    After (good style):
    1. User enters name and address.
    2. System presents user’s profile.

Writing effective use-cases, cont.

- Guidelines for an action step
  - Guideline 6: It doesn’t "check whether", it "validates”

    Wrong:

    2. The system checks whether the password is correct

    3. If it is, the system presents the available actions for the user.

    Good:

    2. The system validates that the password is correct

    3. The system presents the available actions for the user.

  - Guideline 7: It optionally mentions the timing
    - At any time between steps 3 and 5, the user can cancel the use-case

Writing effective use-cases, cont.

- Guidelines for an action step
  - Guideline 8: Idiom: "Do steps x-y until condition"
    - The user selects one or more products.
    - The user searches through various product catalogs until he finds the one he wants to use.
    - Customer repeats steps 3-4 until indicating that he/she is done.
    - Steps 2-4 can happen in any order.

Patterns for effective use-cases

- A general template for a use-case

  A use case describes an actor’s **CompleteSingleGoal,** written with a **VerbPhraseName** and organized as **ScenarioPlusFragments,** written with **LeveledSteps,** each step showing distinct **ForwardProgress,** with the **VisibleActorIntent** and in a way that is **TechnologyNeutral**

Patterns for effective use-cases, cont.

- **CompleteSingleGoal –** a use-case intends to satisfy one single goal the actor is trying to achieve
- **VerbPhraseName –** a meaningful name that adequately describes a use case's purpose
- **ScenarioPlusFragments** - unlike most stories, use cases have multiple scenarios leading to the CompleteSingleGoal which are best organized with the use of some structures
- **LeveledSteps –** excessively large or excessively small use case steps obscure the goal and make the use case difficult to read and comprehend.

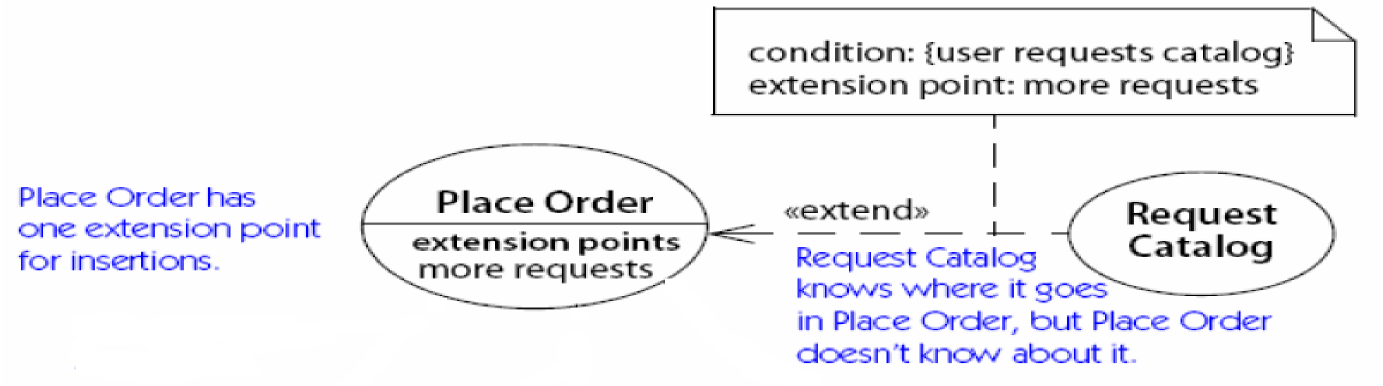
Patterns for effective use-cases, cont.

- **ForwardProgress –** eliminate steps that don’t advance the actor, simplify passages that distract the reader from this progress
- **VisibleActorIntent –** write each step to clearly show which actor is performing the action, and what that actor will accomplish
  - We recommend a clear sentence with straightforward grammar: actor, verb, direct object, and prepositional phrase.
  - "User enters name and address," or "System notifies user of results."
- **TechnologyNeutral –** Including technology constraints and implementation details in a use case description increases the complexity and obscures the goal of the use case.

Relationships between use-cases

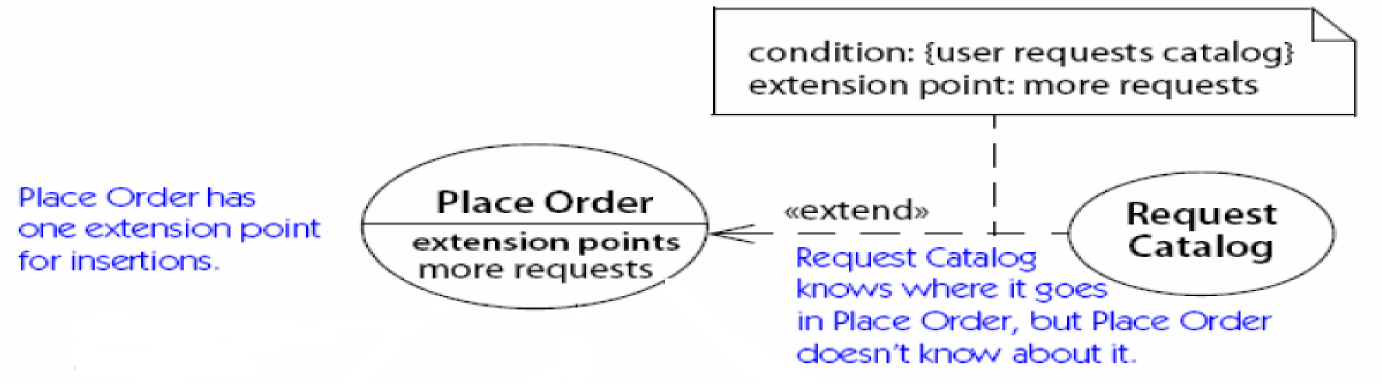
- Two use-cases can be related with:
  - <<extend>> relationship
  - <<include>> relationship
  - Generalization relationship

Relationships between use-cases, cont.


condition: {user requests catalog}
extension point: more requests
Place Order has one extension point for insertions.
Place Order
extension points
more requests
«extend»
Request Catalog
Request Catalog knows where it goes in Place Order, but Place Order doesn't know about it.

- Extend relationship
  - The client use-case (Request Catalog) adds incremental behavior to the base use-case (Place Order) by inserting additional action sequences into the base sequence.
- An extension point – the name of extension
- A condition – when to perform the extension (can be omitted = true)
  - if the extend relationship has more than one extension point, the condition is evaluated only at the first extension point prior to execution of the first segment.

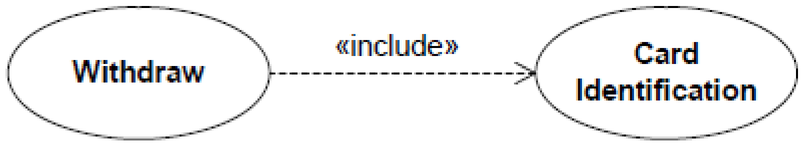
Relationships between use-cases, cont.


condition: {user requests catalog}
extension point: more requests
Place Order has one extension point for insertions.
Place Order
extension points
more requests
«extend»
Request Catalog
Request Catalog knows where it goes in Place Order, but Place Order doesn't know about it.

- Description of Place Order use-case
  - Action 1

  More requests: // potential insertion point (here the condition is // evaluated)
  - Action 2
- Description of Request Catalog use-case
  - Action 1
  - Action 2 …

Relationships between use-cases, cont.

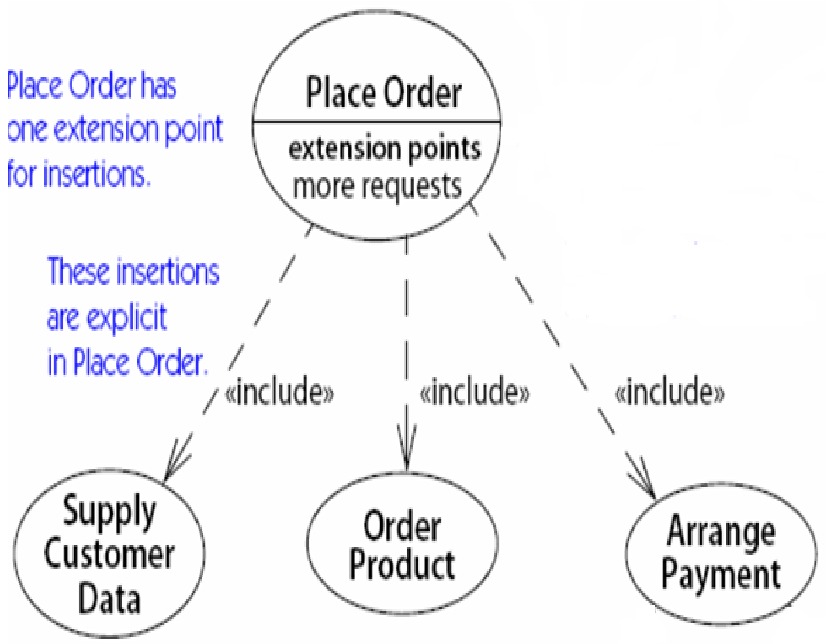


- Include relationship:
  - The inclusion of the behavior sequence of the supplier use-case (Card identification) into the inter action sequence of a client use-case (Withdraw), under the control of the client use-case at a location the client specifies in its description.
- The use-case instance is executing the client use-case. When it reaches the inclusion point, it begins executing the supplier use-case until it is complete.
- Then it resumes executing the client use-case beyond the inclusion location.

Relationships between use-cases, cont.

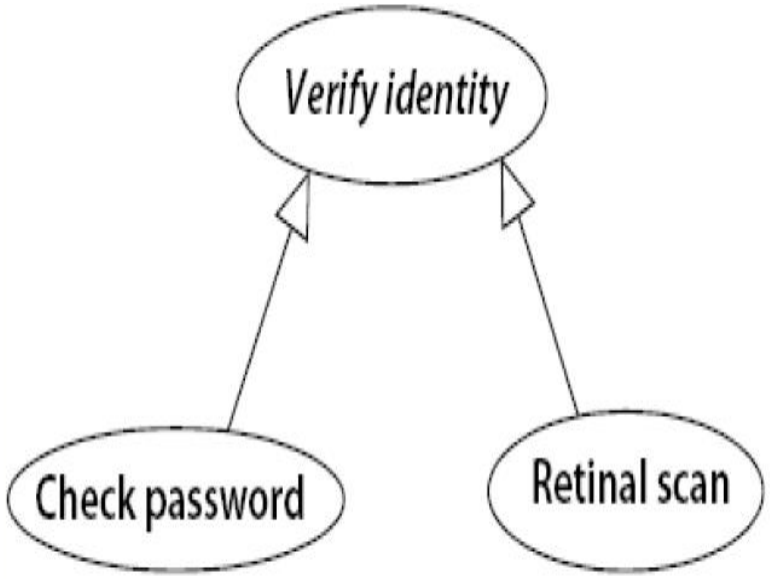
- Description of **Place Order (orchestrating use-case)**
  - Action 1
  - **Include: Request Catalog**
  - Action 2
  - **Include: Supply Customer Data**
  - Action 3
  - **Include: Arrange Payment**

- Description of **Order Product**
  - Action 1
  - Action 2 …
- Description of **Supply Customer Data**
  - Action 1
  - Action 2 …
- Description of **Arrange Payment**
  - Action 1
  - Action 2 …



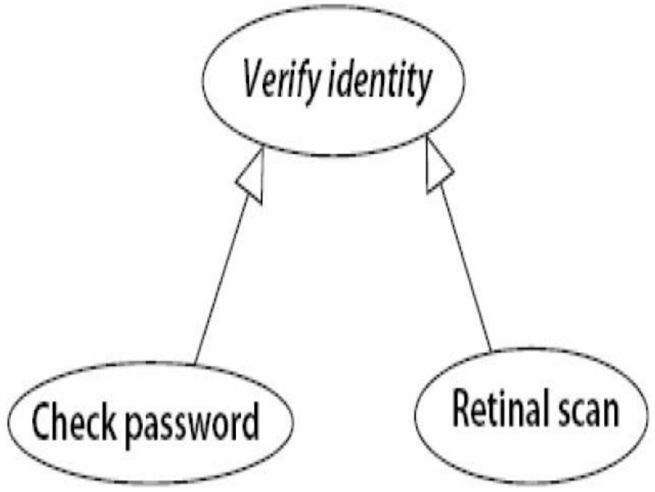
Relationships between use-cases, cont.

- There is possible to define inheritance relationships between use-cases
- The child-use case inherits all properties (sequence of actions) from its parent-use case
- The child use-case adds incremental behavior to the parent use-case by inserting additional action sequences into the parent sequence at arbitrary points. It may also modify some inherited operations and sequences, but this must be done with care so that the intent of the parent is preserved.

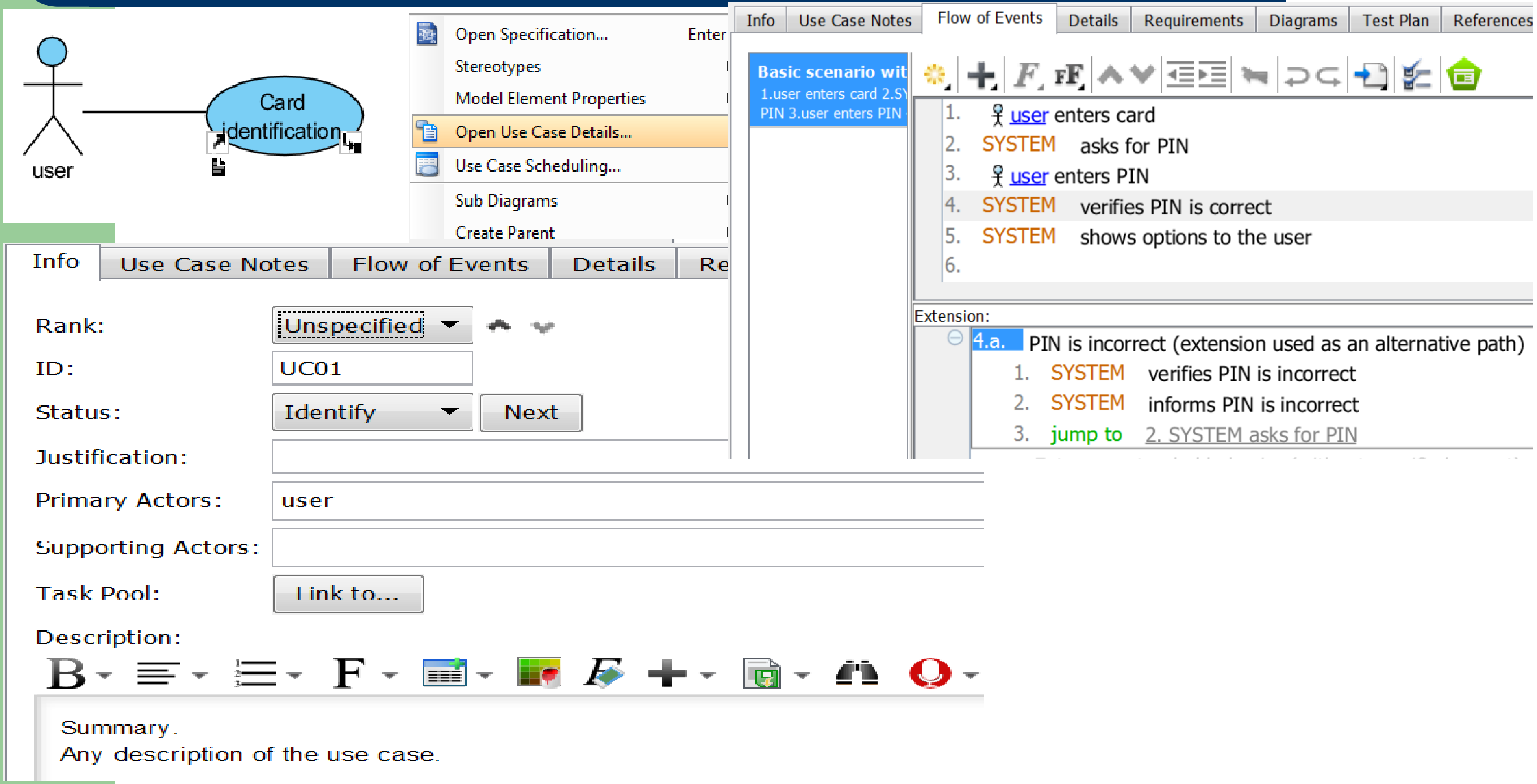


Relationships between use-cases, cont.

- Use-case behavior description for Verify identity
  - The use-case is abstract (has no direct instances)
  - The concrete descendants must provide behavior sequence
- Use-case behavior description for Check password
  - System asks for a password
  - Users supplies password
  - System verifies the provided password is correct
- Use-case behavior description for Retinal scan
  - System asks a user to provide a retina
  - User supplies retina
  - System scans user's retina
  - System verifies the retina belongs to any user



How to specify use-case in VP



1. user enters card
2. SYSTEM asks for PIN
3. user enters PIN
4. SYSTEM verifies PIN is correct
5. SYSTEM shows options to the user
6.

Extension:

4.a. PIN is incorrect (extension used as an alternative path)

1. SYSTEM verifies PIN is incorrect
2. SYSTEM informs PIN is incorrect
3. jump to 2. SYSTEM asks for PIN

Rank:	Unspecified
ID:	UC01
Status:	Identify (Next)
Justification:	
Primary Actors:	user
Supporting Actors:	
Task Pool:	Link to...

Description:

Summary.
Any description of the use case.

Use-case model – example 2

- E-mail system enables its users:
  - editing,
  - sending, and
  - receiving

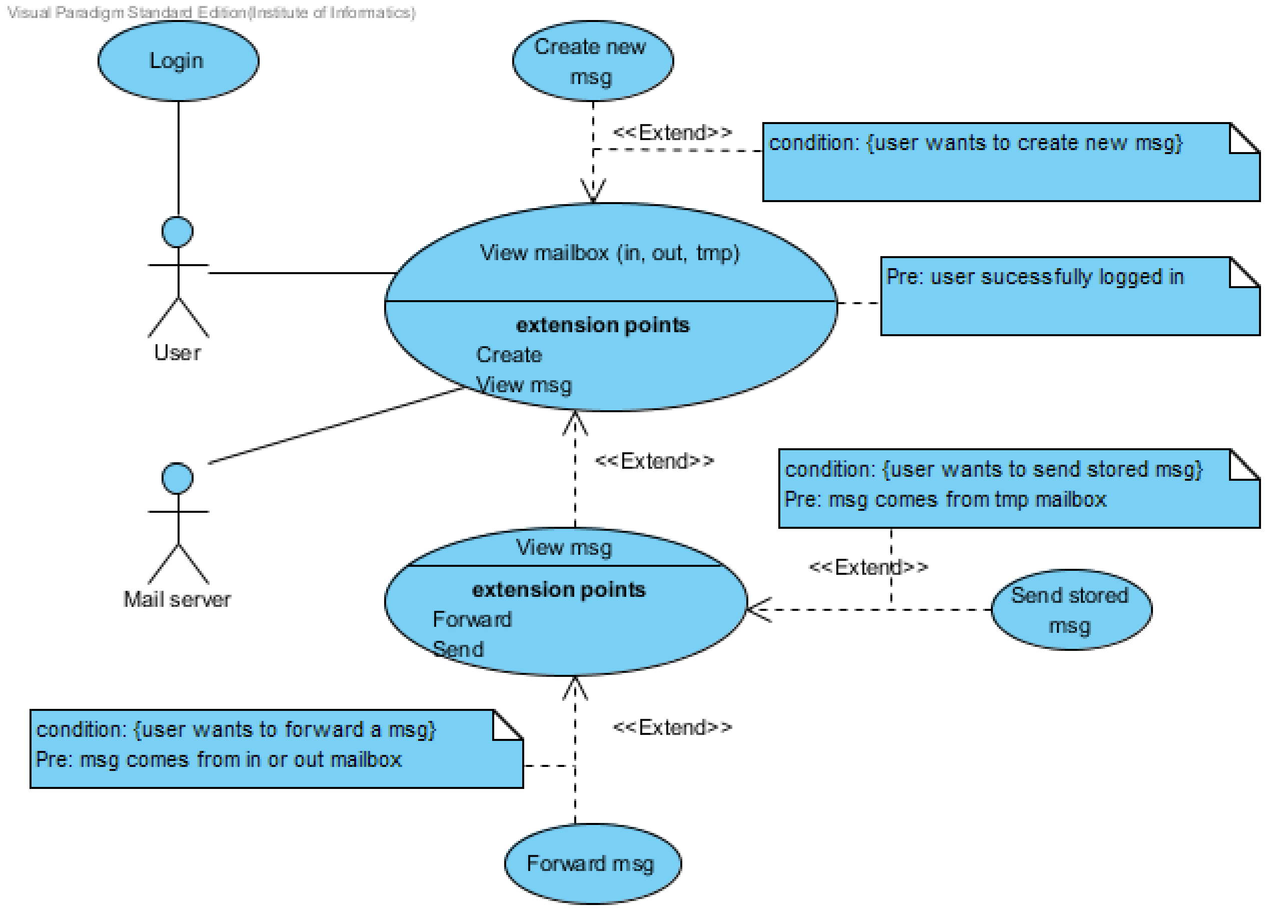
messages (letters) in electronic form.

- Each user has his/her unique name, password, and e-mail box or messages.
- To begin a work with the e-mail system user has to login, entering his/her name and password

Use-case model – example 2

- A user may check the state of his e-mailbox.
- Before sending e-mail user has to define recipient's address and prepare contents.
- E-mail system checks validity of the address. If the address is not valid the letter is not sent and an exception arises.
- Any received letter maybe forwarded to another recipient.

Use-case model example 2 – e-mail system


Visual Paradigm Standard Edition(Institute of Informatics)
Login
Create new msg
<<Extend>>
condition: {user wants to create new msg}
View mailbox (in, out, tmp)
extension points
Create
View msg
Pre: user sucessfully logged in
User
Mail server
<<Extend>>
condition: {user wants to send stored msg}
Pre: msg comes from tmp mailbox
View msg
extension points
Forward
Send
<<Extend>>
Send stored msg
<<Extend>>
condition: {user wants to forward a msg}
Pre: msg comes from in or out mailbox
Forward msg

Exercise

- Propose a use-case model (fragment) for an Internet shop
- Hint: start from actor identification and their needs
- Provide at least short descriptions for use-cases

User stories

- User story – a short sentence describing functionality of a system that will be valuable to **a stakeholder** (not a team) of a system or software.
- Mostly used within agile methodologies which assume that a representative of a customer is a part of development team

User stories templates

- As a <role>, I want <goal/desire> so that <benefit> („so that” clause is optional)
- In order to <benefit> as a <role>, I want <goal/desire>
- As <who> <when> <where>, I <what> because <why>

User stories examples

- As a user, I want to search for my customers by their first and last names so that I can find contact data to them
- As a user, I want to start an application with the last edit so that I can spare time
- As a user closing the application, I want to be prompted to save if I have made any change in my data since the last save so that I won’t loose my work accidentaly

User stories limitations

- Scale-up problem – a big number of user-stories as they represent small chunks of functionality
- Vague, informal, incomplete:
  - Are regarded as conversation starters with the assumption that a customer or end user representative is always a part of the team
  - Do not include any details of user interactions with the system

Use-cases vs user stories

- Similarities:
  - Usually written in natural language
  - Testable, can be accompanied with tests cases
- Differences:
  - User stories: small-scale requirements with little details; chunks for planning purposes
  - Use-cases: larger, sequences of interactions, should contain sufficient detail to be understood on its own
  - A small use case may correspond entirely to a story; however a story might be one or more scenarios in a use case, or one or more steps in a use case

Summary

- UML is commonly used graphical language used for different purposes
- UML provides many types of diagrams for presenting static and dynamic aspects of a modeled entity
- Use-case diagram is used for presentation of system services and system surrounding; it is expressed in terms of actors and use-cases
- The use-case diagram is not enough to present system functionality – it must be accompanied with use-case specifications
- Use-case specification should follow specific guidelines to fulfill quality criteria
- User stories are alternative representation of functional requirements; they require customer/end user involvement

Revision

- What is a model?
- What is the UML for?
- What are the main elements of the use-case diagram?
- What relationships may be defined between actors?
- What relationships may be defined between use-cases?
- What is a typical structure of UML use-case specification?
- Define at least 3 exemplary guidelines how to write good use-cases
- What is a user-story? What it is for?