CHRISTIAN HUEMER
MARION SCHOLZ

Object-Oriented Modeling with UML
Part V – Use Case Diagram

Use Case Diagram

The Use Case Diagram

Christian Huemer und Marion Scholz

Presented by Nicholas Bzowski

Use Case (1/3)

- Use cases are the starting point for many object-oriented development methods.
- Often also the core concept that spans the entire analysis and design process.
- Initial questions for the implementation of use cases:
  - Why use use cases?
  - What do use cases look like?
  - What do you do with use cases once you have identified them?
- Use cases focus on the fundamental problem when developing a system: developing a solution for customers or users who have requested it.

Use Case (2/3)

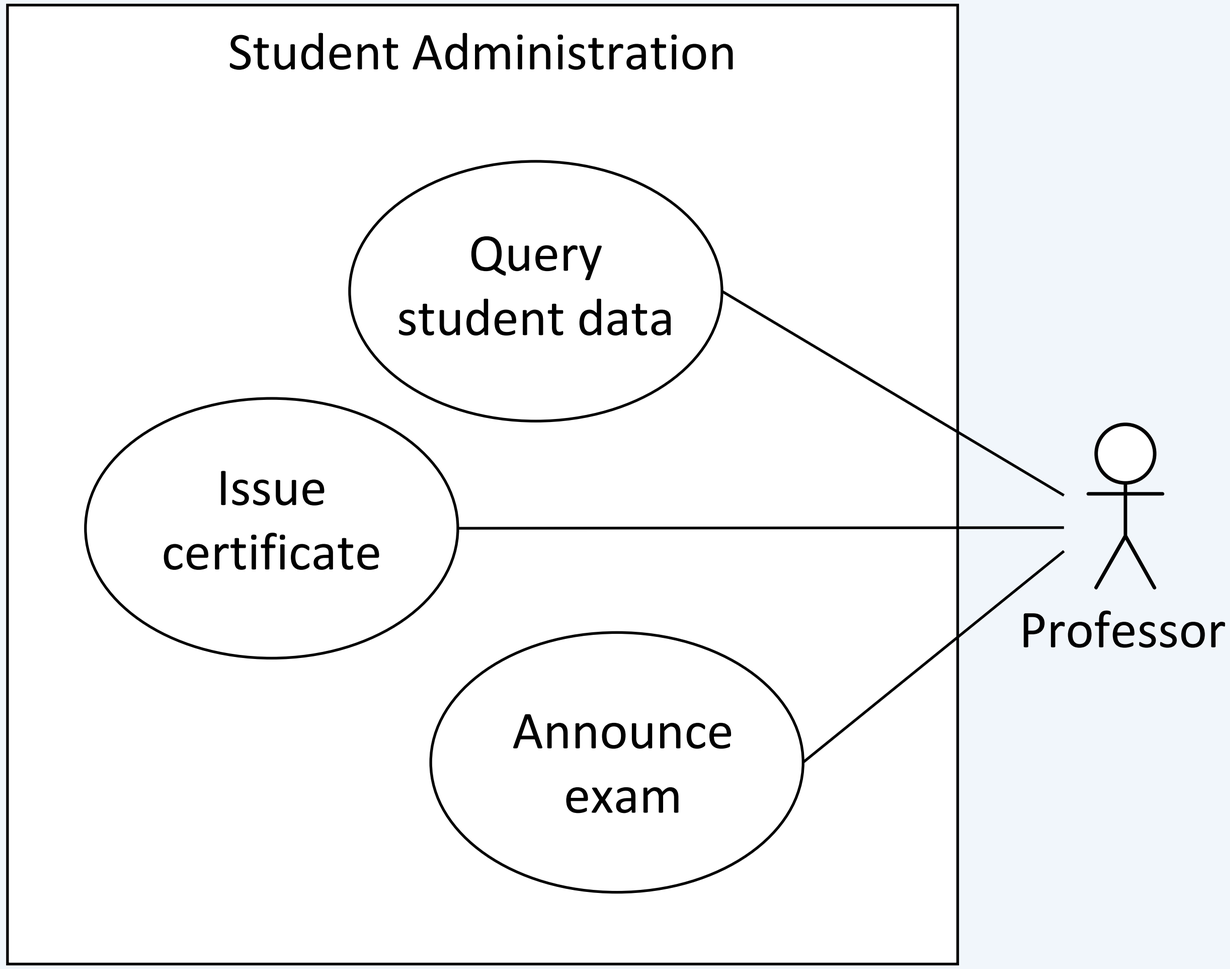
- Use cases represent the requirements of customers
- “A use case is a sequence of transactions within a system whose task is to generate an identifiable benefit for individual actors (users).” [Ivar Jacobson]
- Actors interact with the system in the context of the use cases
- Actor:
  - Role that someone or something assumes and that is related to the business area, or
  - Anything that interacts with the system

Use Case (3/3)

- Transactions within a system imply that actors are offered a range of opportunities to communicate with the system and that a measurable benefit is generated through them.
- A measurable benefit implies that the execution of a transaction has a visible, quantifiable and/or qualifiable impact on those things that lie outside the system, specifically on actors.

Example: Student Administration

- **System** (What is being described?)
  - **`Student Administration`**
- **Actor** (Who is using the system?)
  - **`Professor`**
- **Use cases** of **`Professor`**
  (What do the actors do?)
  - Query student data
  - Issue certificates
  - Announce exams



Use Case Diagram
Gumball Machine Example

Christian Huemer und Marion Scholz

Presented by Nicholas Bzowski


DOLLY
FRUITS
10 Cent
BUBBLE GUM

Use Case Diagram

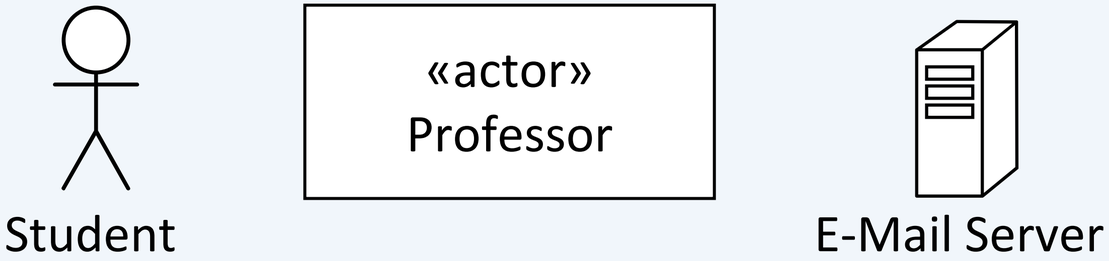
The Actor and The Use Case

Christian Huemer und Marion Scholz

Presented by Nicholas Bzowski

Actor

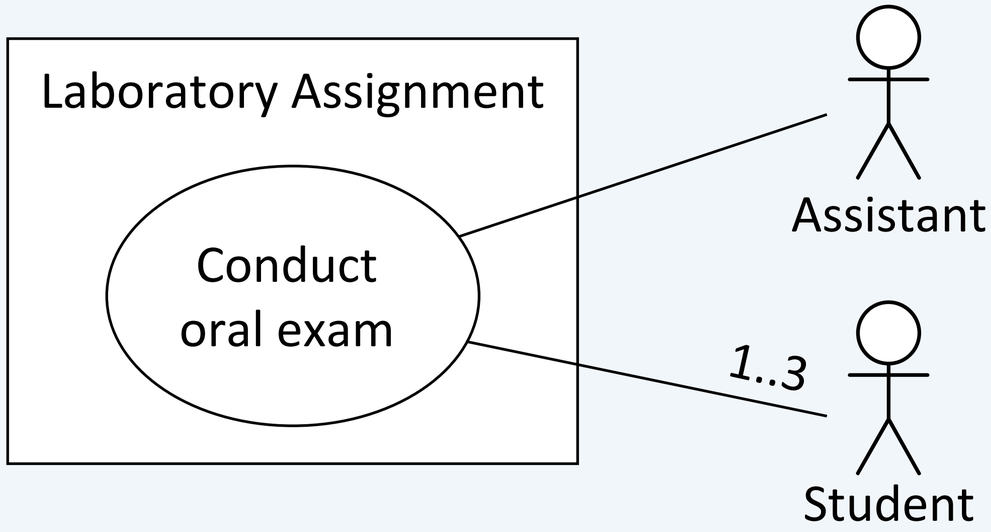
- Actors **interact** with the system…
  - by **using** the **System**,
  - by being **used by the System**
- Actor is **connected to use cases through associations,**
  i.e. he/she “communicates” with the system
- Each actor must communicate with
  at least one use case
- The association is binary and can have **multiplicities**
- Notation variants:



Actor - Characteristics

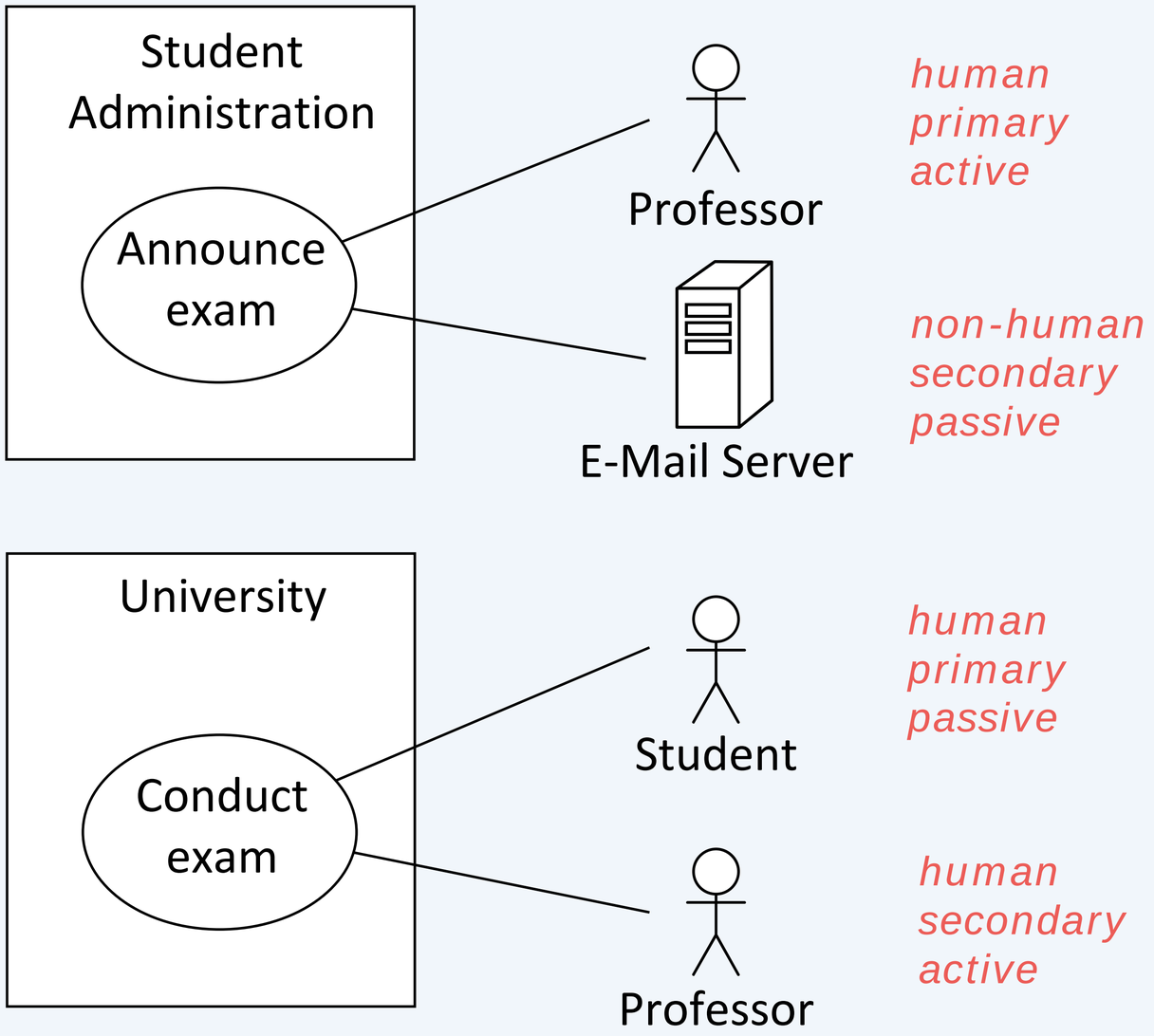
- Actors represent the **roles of the users**
  - Concrete users can play, accept and discard several roles at the same time
- Actors are **clearly outside** of the system boundaries
- User data is usually also managed within the system. These are modeled as objects or classes within the system.

- Example: **`Laboratory Assignment`**
  - Actor **`Assistant`** interacts with the system **`Laboratory Assignment`** by using it
  - The class **`Assistant`** comprises objects that contain the user data (name, social security number, ...)



Actor - Classification

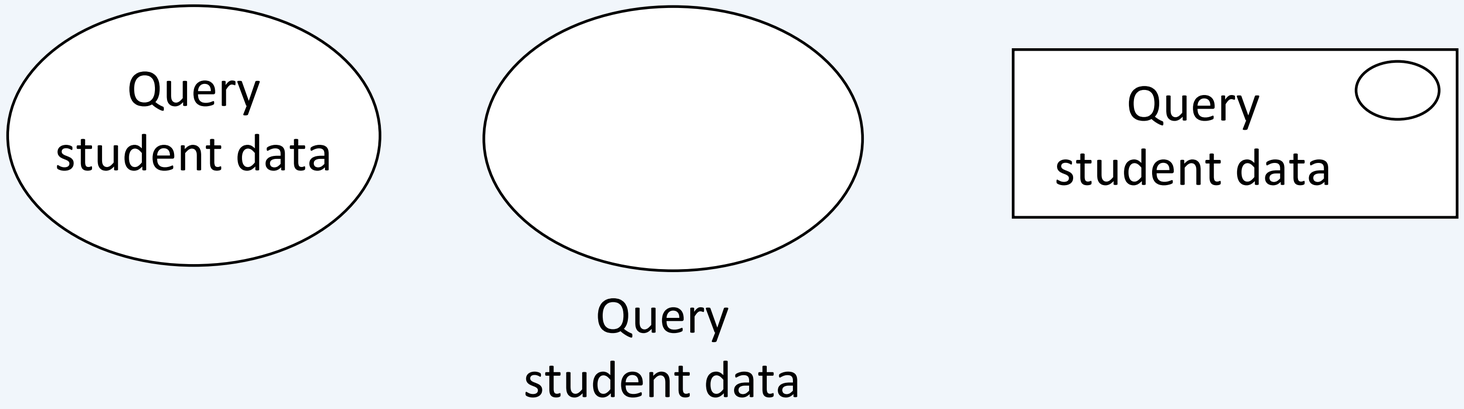
- **Human**
  - ex. **`Student`**, **`Professor`**
- **Non-human**
  - ex. **`E-Mail Server`**
- **Primary**: main beneficiary of the application
- **Secondary**: necessary for the system to function
- **Active**: initiates use cases itself
- **Passive**: does not initiate any use cases itself



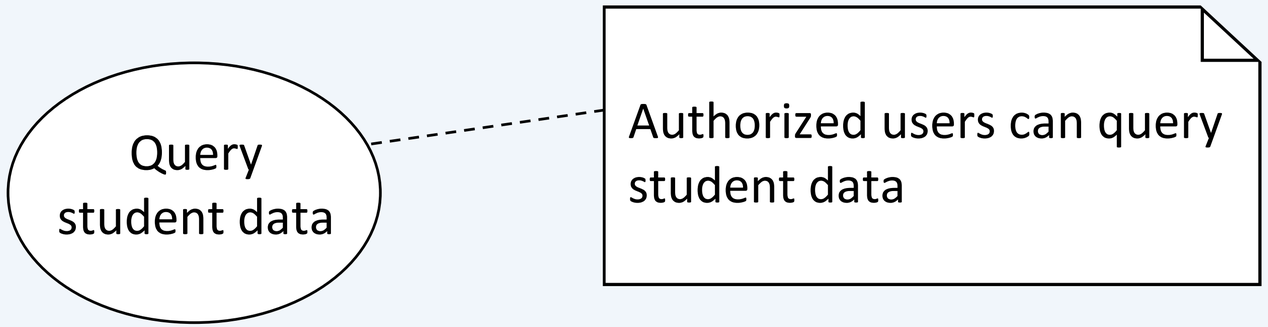
Use Case

- Use cases describe the **behavior** that is **expected** of the system to be developed

- Notation variants



- Short description as a note



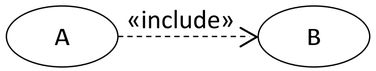
Use Case Diagram

Relationships Between Use Cases

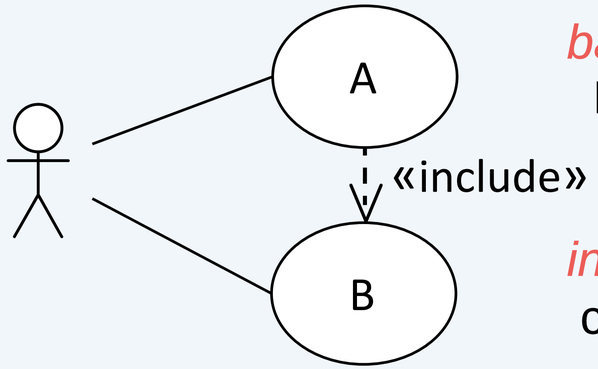
Christian Huemer und Marion Scholz

Presented by Nicholas Bzowski

«include» - Relationship



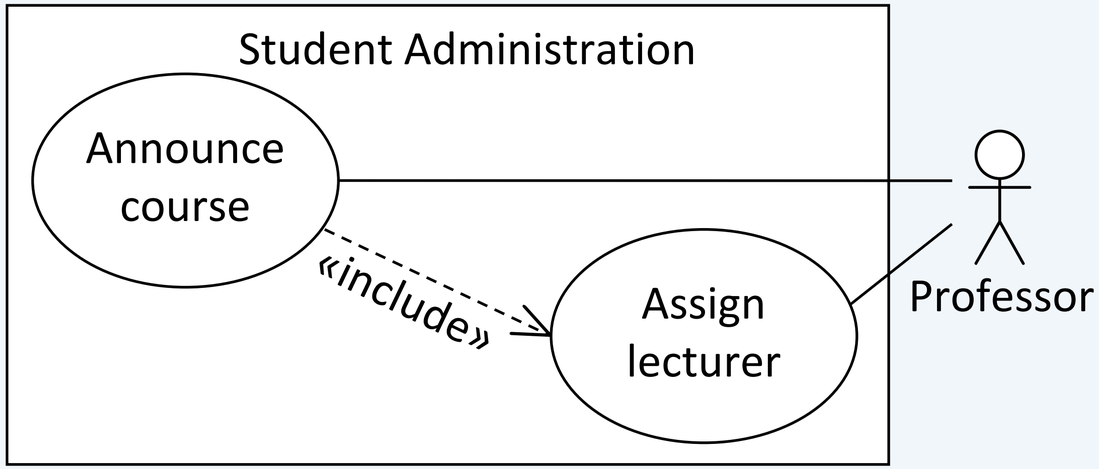
- The **behavior** of the used use case (included use case) is **integrated** into the using use case (base use case)

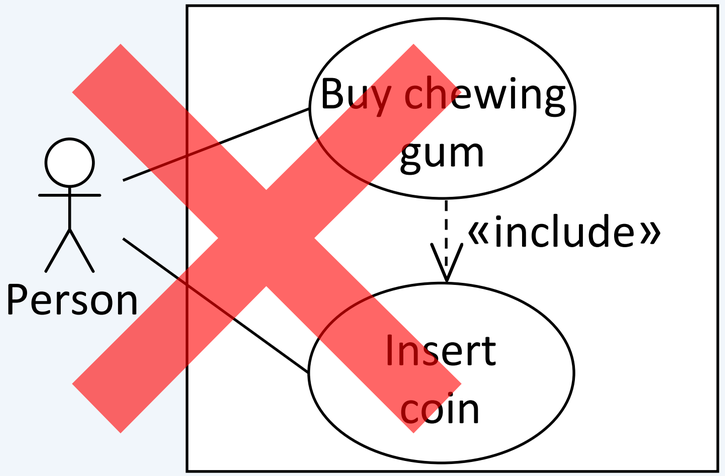


base use case
B required to ensure functionality

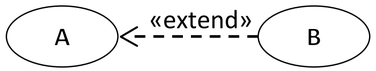
included use case
can also be executed separately

- Ex:

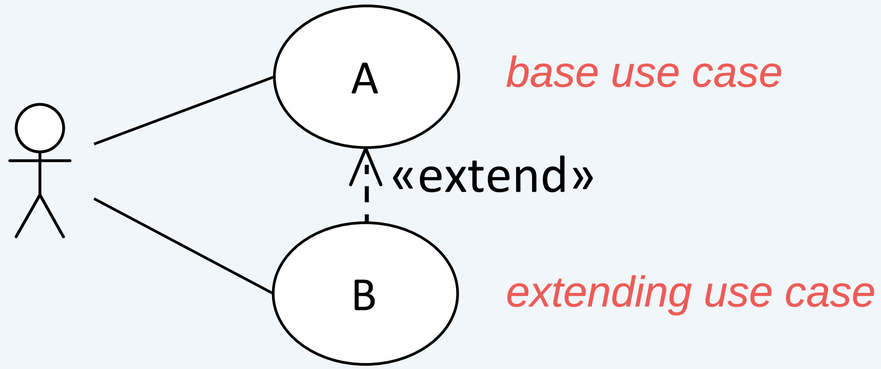




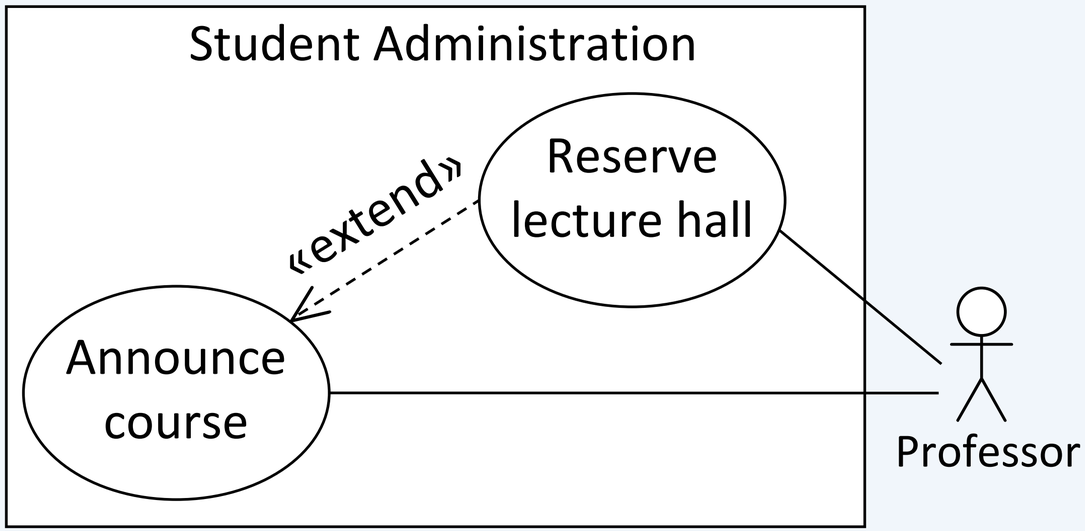
«extend» - Relationship



- The behavior of **B can be included** in **A**
  - **A** decides whether **B** is executed
- **A** and **B** can also be executed separately
- Indication of the “Where” by extension points in **A**
- Specification of the “when” by condition in **A** or as part of the **«extend»** relationship



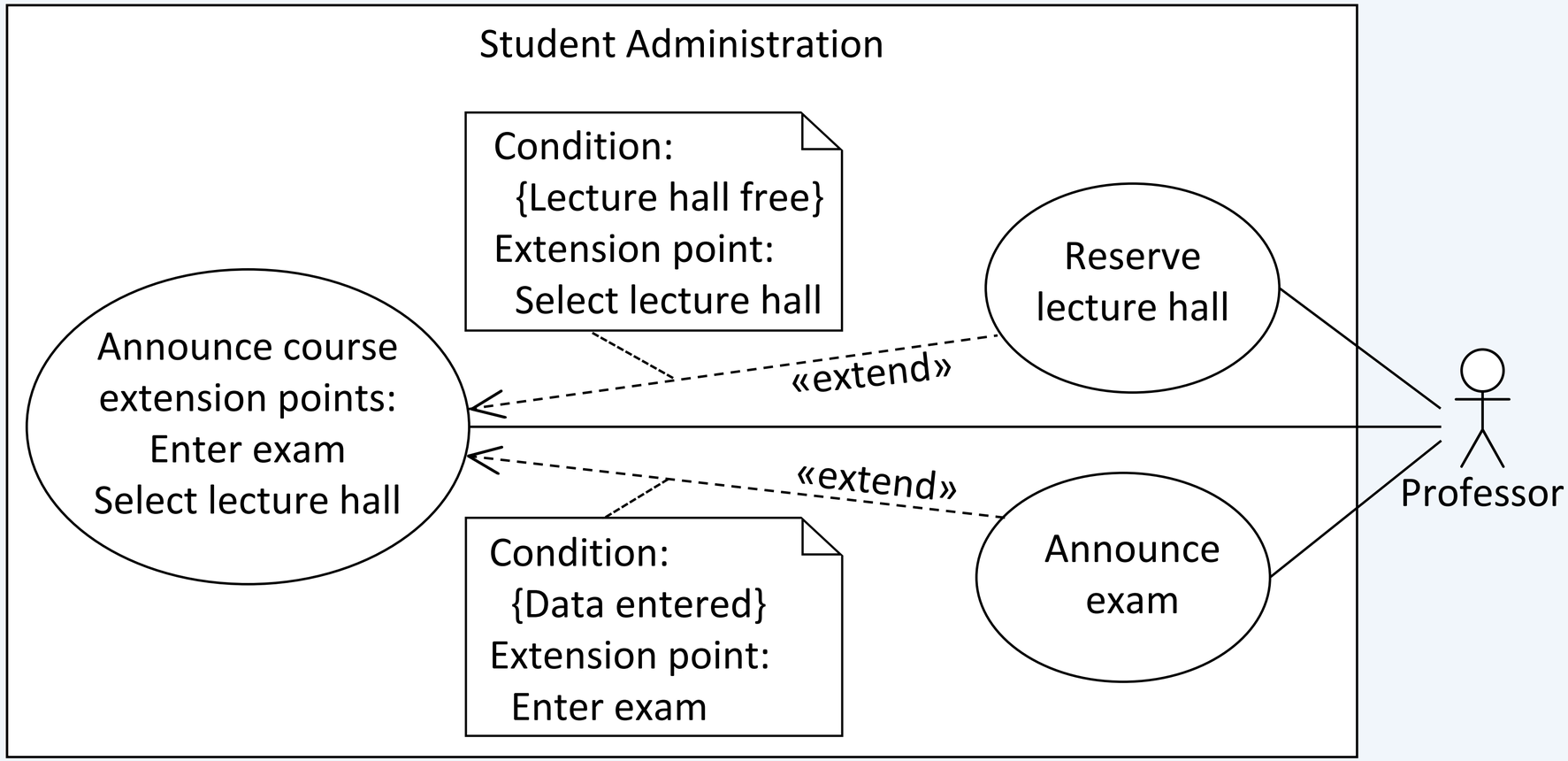
- Ex.:



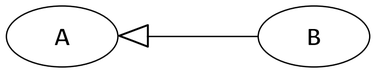
«extend» - Relationship: Extension Points

- Several extension points possible for each application
- Names of extension points must be unique

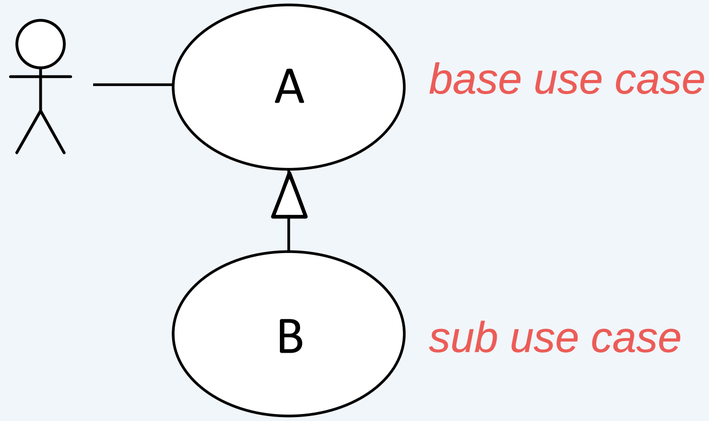
- Ex.:



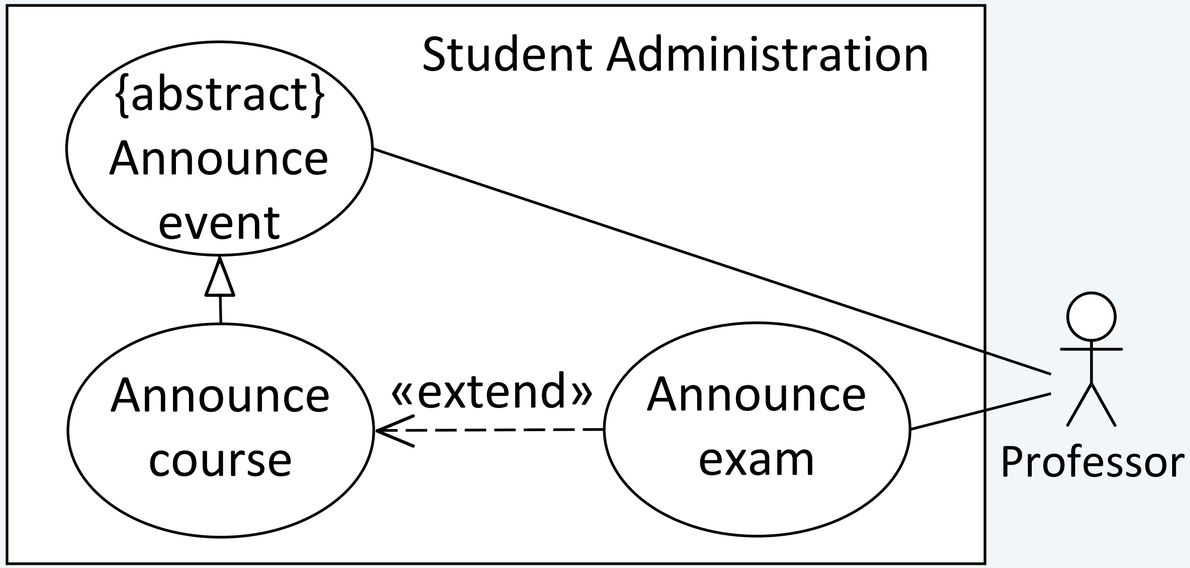
Generalization of Use Cases



- **B** inherits the behavior of **A** and can overwrite or augment it
- **B** inherits all relationships of **A**
- **B** takes over basic functionality of **A**
- **B** decides what is executed or changed by **A**



- Modeling of abstract use cases possible:
  `{abstract}`
  - abstract use cases cannot be executed!

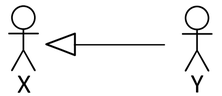


Use Case Diagram
Relationships Between Actors

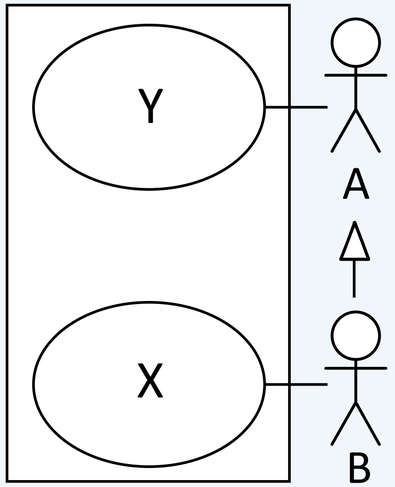
Christian Huemer und Marion Scholz

Presented by Nicholas Bzowski

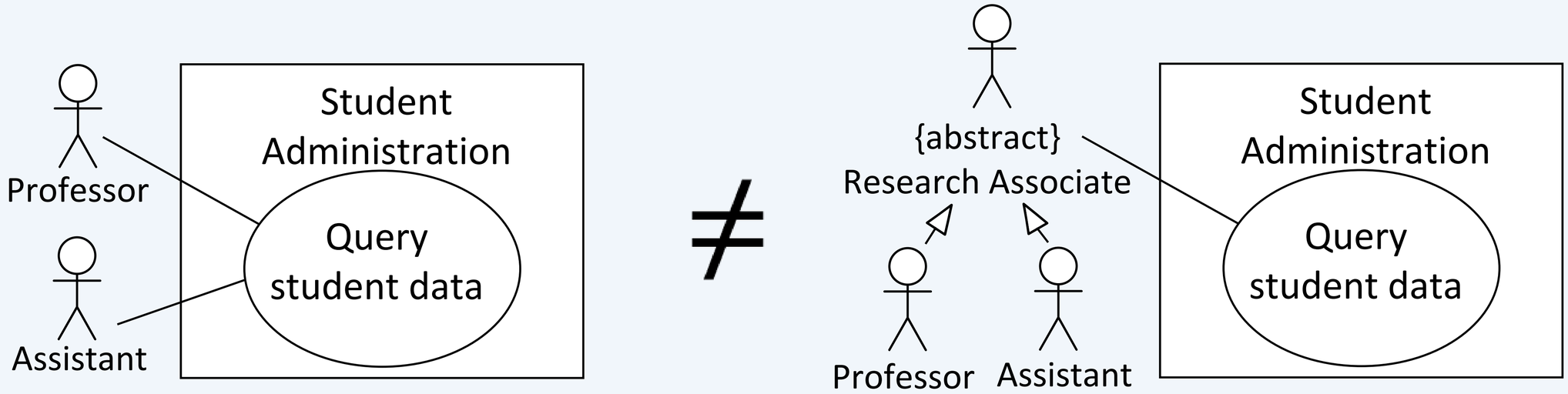
Generalization of Actors (1/2)



- Actor **B** inherits from actor **A**
- **B** can communicate with use cases **X** and **Y**
- **A** can only communicate with **Y**
- Multiple inheritance is allowed

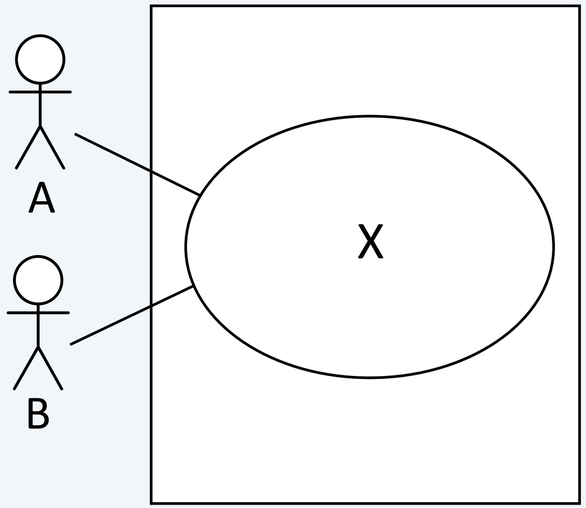


- Ex.:

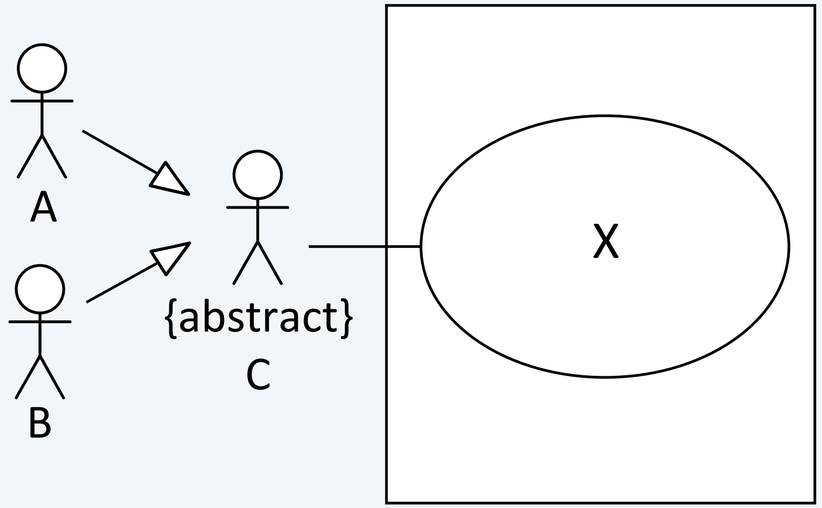


Generalization of Actors (2/2)

- Distinction as to whether several actors can or must communicate together with a use case.



A and B communicate with **X**



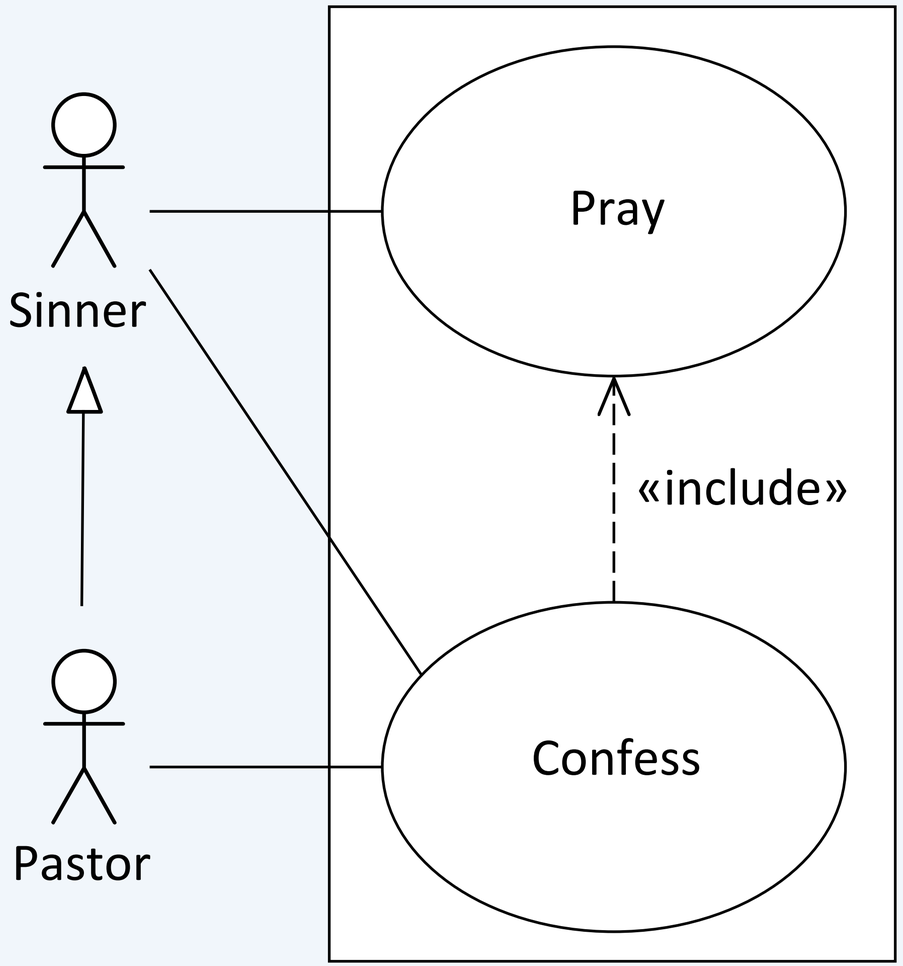
A or B communicate with **X**

Use Case Diagram
Examples

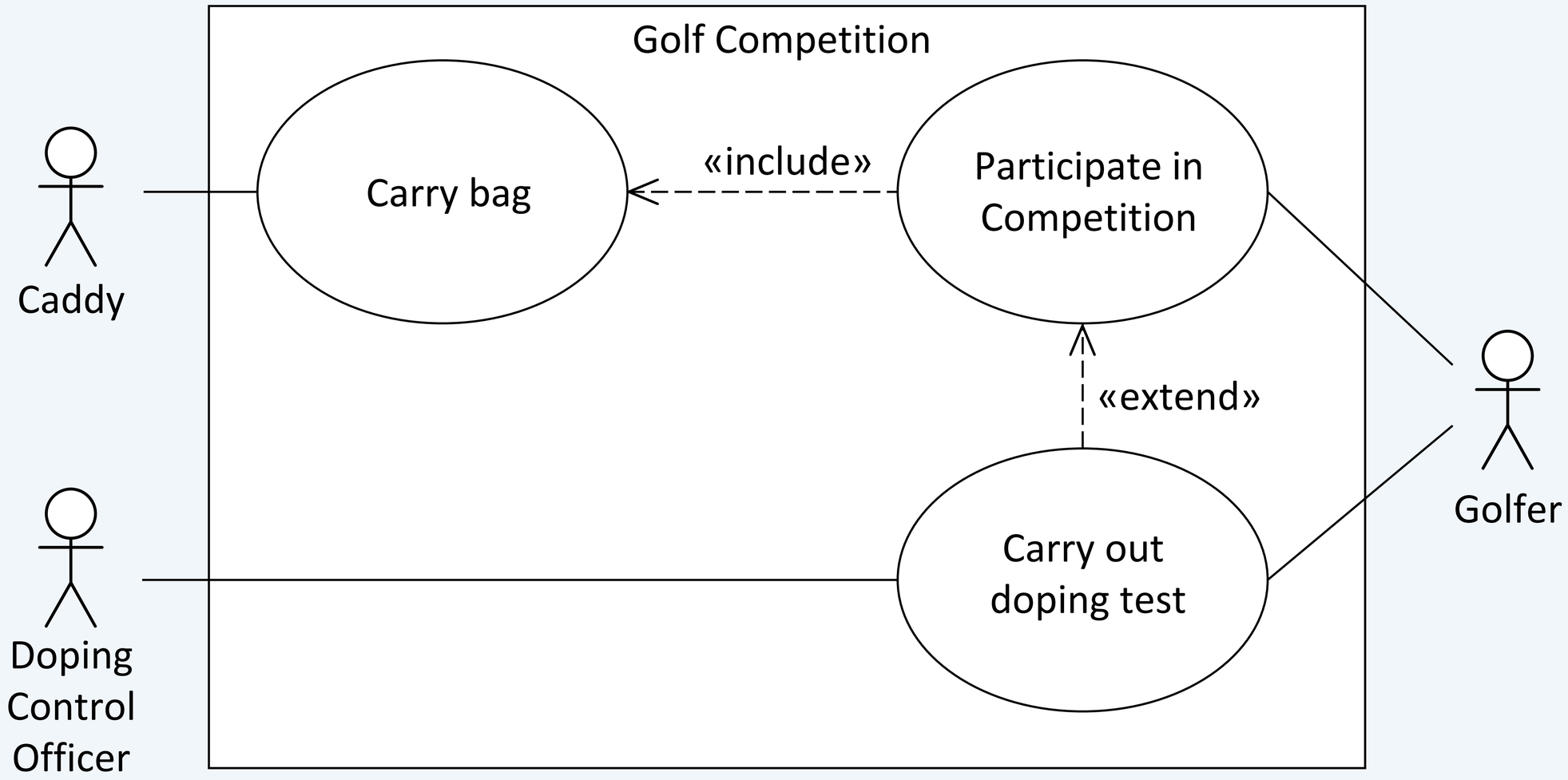
Christian Huemer und Marion Scholz

Presented by Nicholas Bzowski

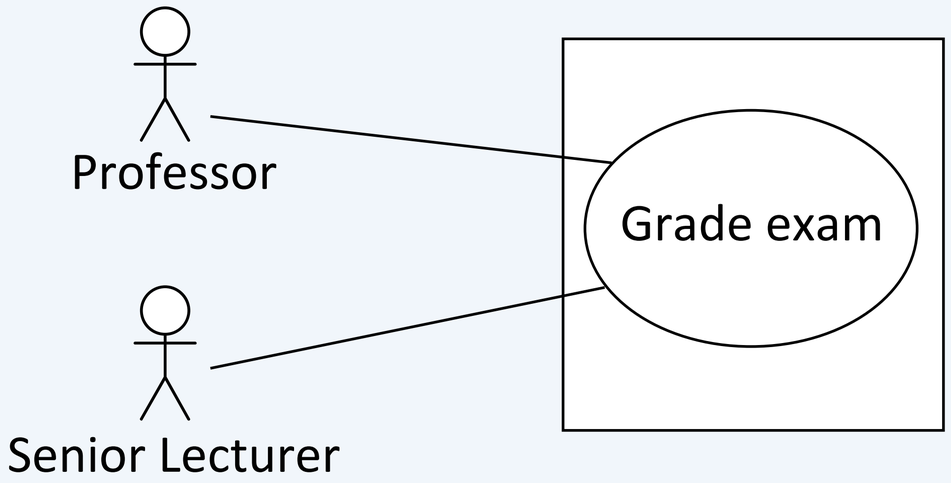
Example: Confession

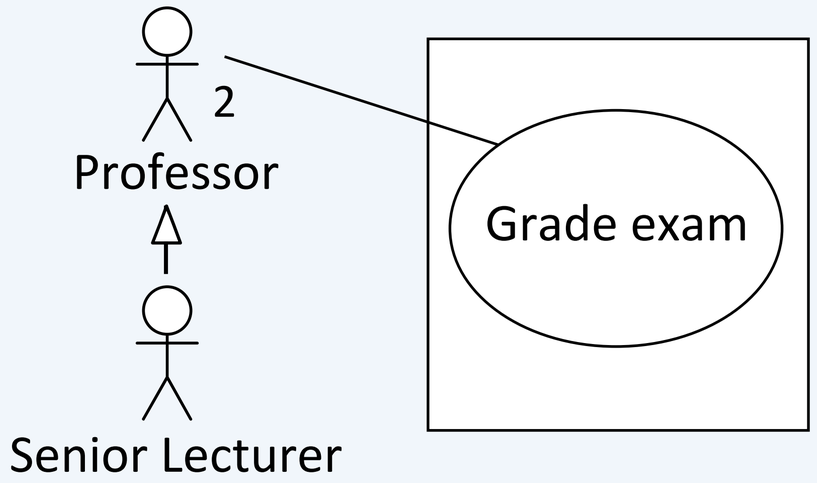


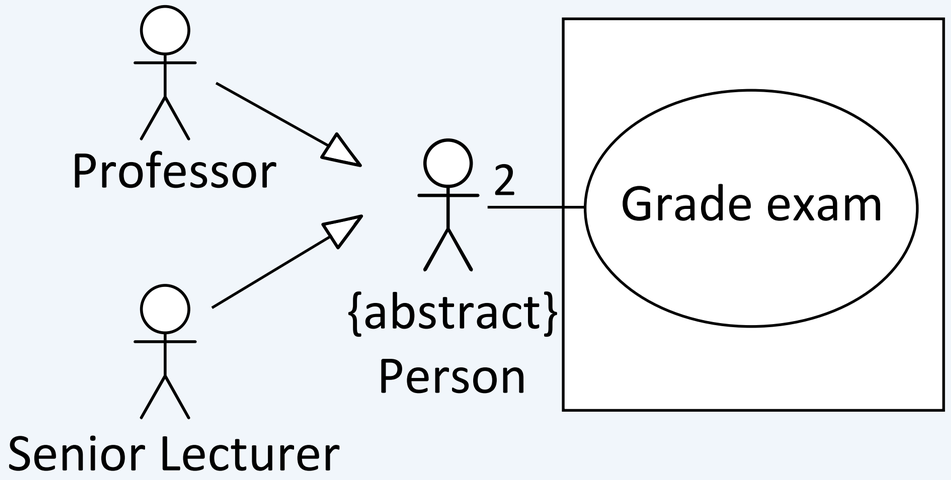
Example: Participating in a golf competition

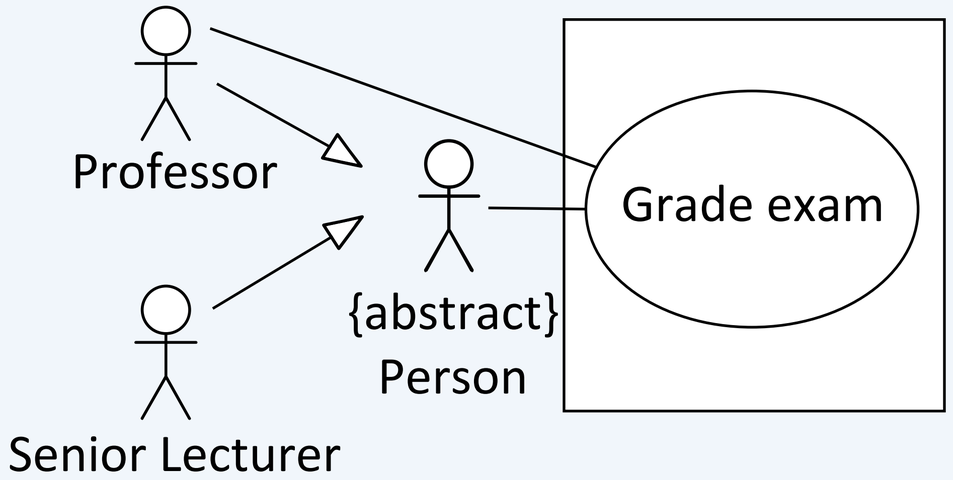


Example: Different types of exam grading









Use Case Diagram

Use Case Modeling in Practice

Christian Huemer und Marion Scholz

Presented by Nicholas Bzowski

Identification of Actors

- Who **uses** the main use cases?
- Who needs the system to support their **day-to-day work**?
- Who is responsible for **system administration**?
- Which external **devices / (software) systems** must the system be able to communicate with?
- Who or what is interested in the **results** of the system?

Identification of Use Cases

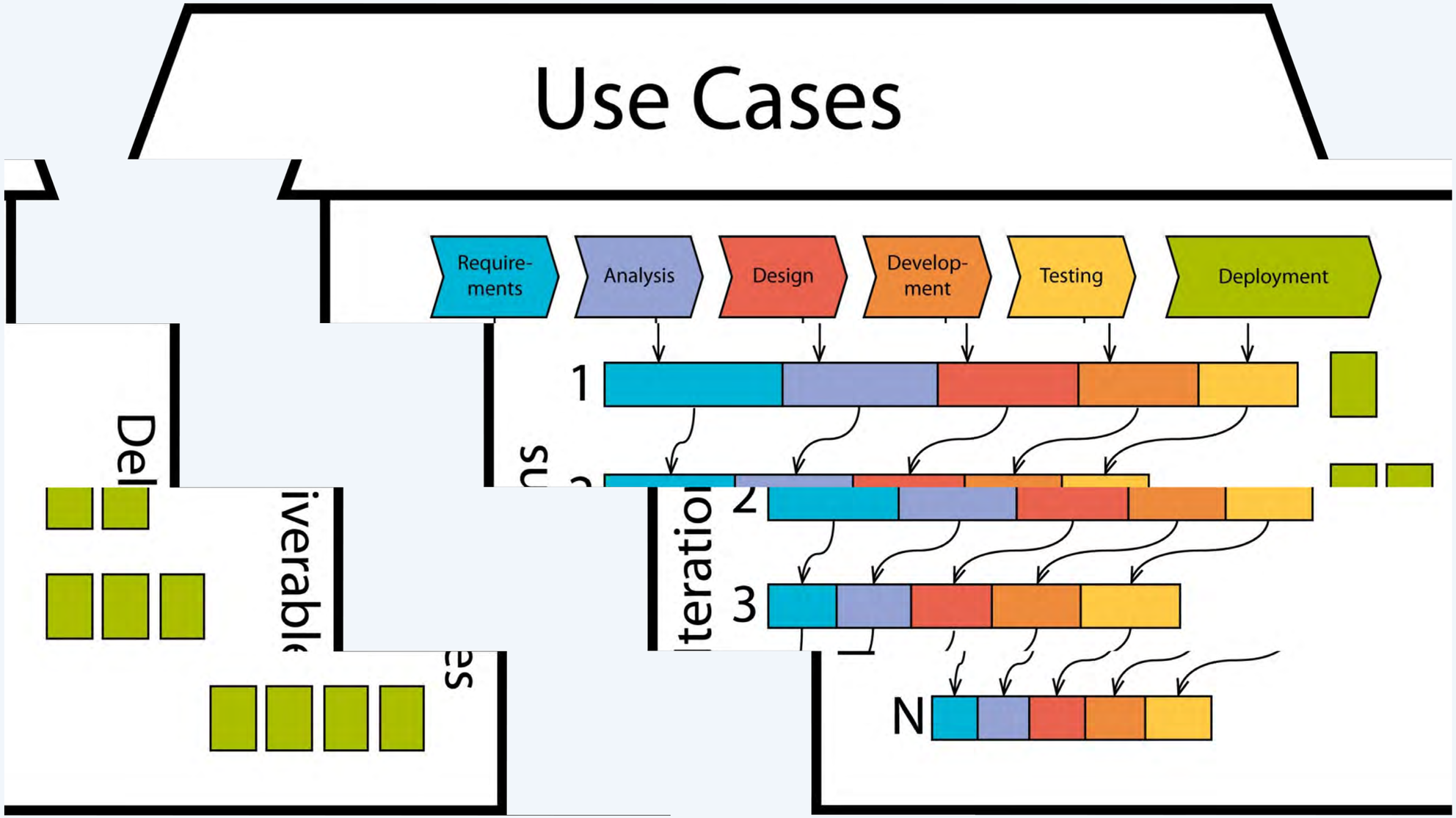
- After identifying the actors
- Search for triggers for use cases
  - Trigger = events that must occur to cause the system to produce a result.
  - The system is often called by the actor, who thus becomes the actor of the use case.
  - The following triggers are distinguished:
    - Internal
    - External
    - Temporal

Use Case Description

- Name
- Short description
- Precondition: Prerequisite for successful execution
- Postcondition: System status after successful execution
- Error situations: errors relevant to the problem area
- System state in the event of an error
- Actors who communicate with the use case
- Trigger: triggering events for the use case
- Standard process: individual steps
- Alternative processes: Deviations from the standard process

Rules for Use Case Modeling

- The most important functional requirements must be defined in the use cases.
- A use case....
  - describes a transaction for the client to pay.
  - describes a typical case of using one system and no more.
  - is like a play. The use case description contains the choreography.
  - has an introduction, a body and a conclusion.
  - should be described as simply as possible.
  - must be defined precisely.
  - is fully described when customers, users and software developers accept it.
  - provides the basis for a system test.
  - should be described on a maximum of two pages.


Use Cases
Require-
ments
Analysis
Design
Develop-
ment
Testing
Deployment
1
2
3
N

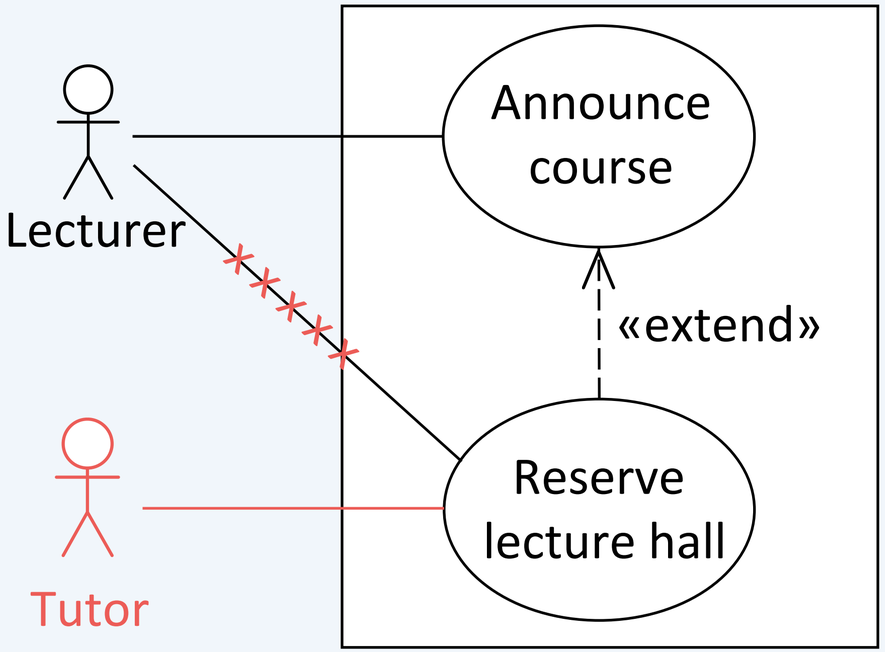
Use Case Diagram
Common Modeling Errors

Christian Huemer und Marion Scholz

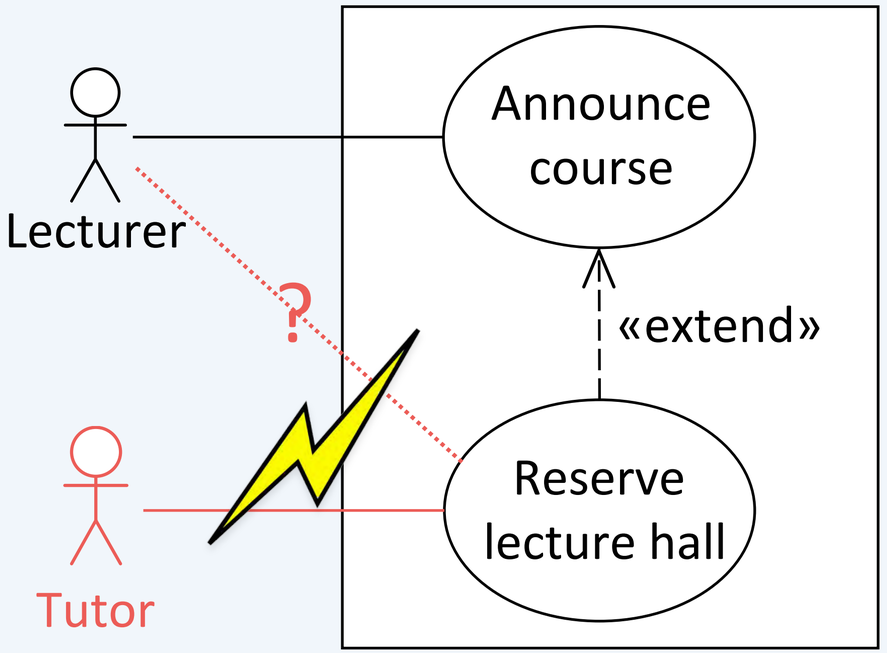
Presented by Nicholas Bzowski

«extend»

UML Standard..



..in practice often:



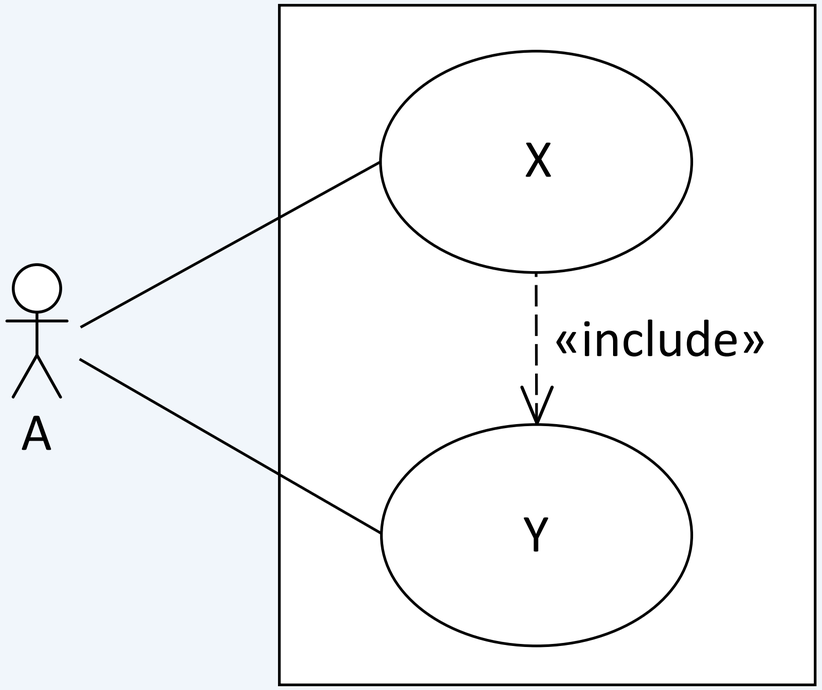
unclear who reserves the lecture hall!

How to show that someone else (alone) reserves the lecture hall?

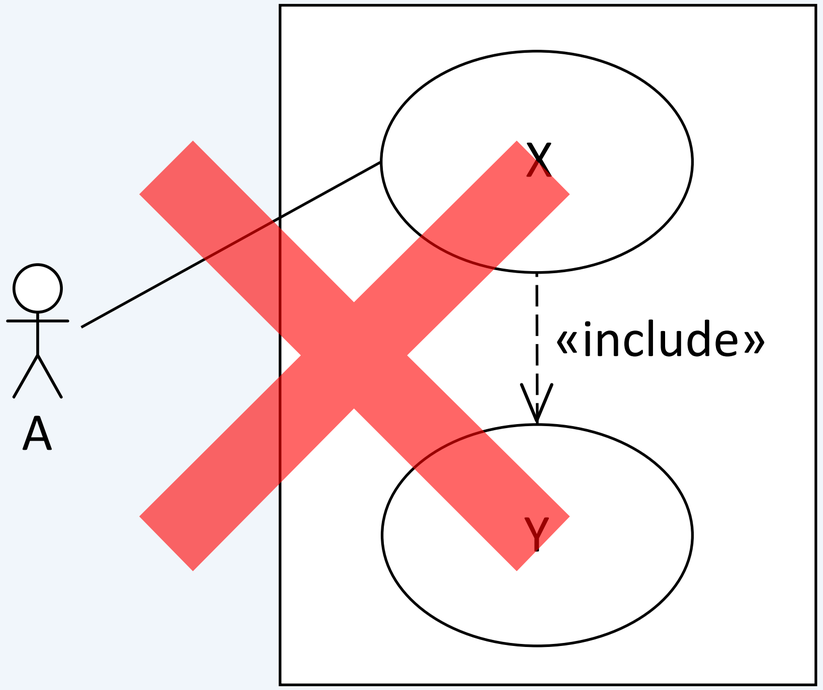
«include»

- similar to **«extend»**

UML Standard..

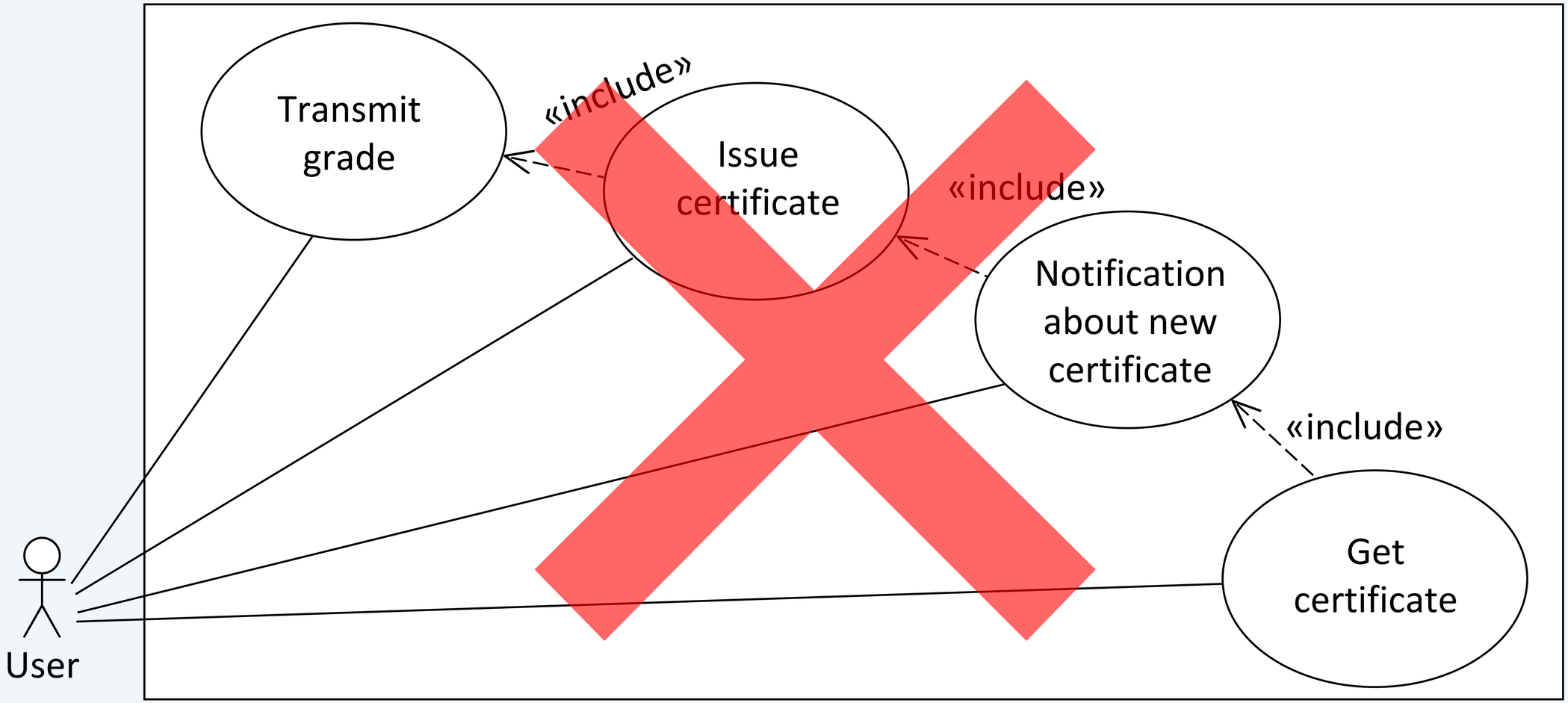


..in practice often:



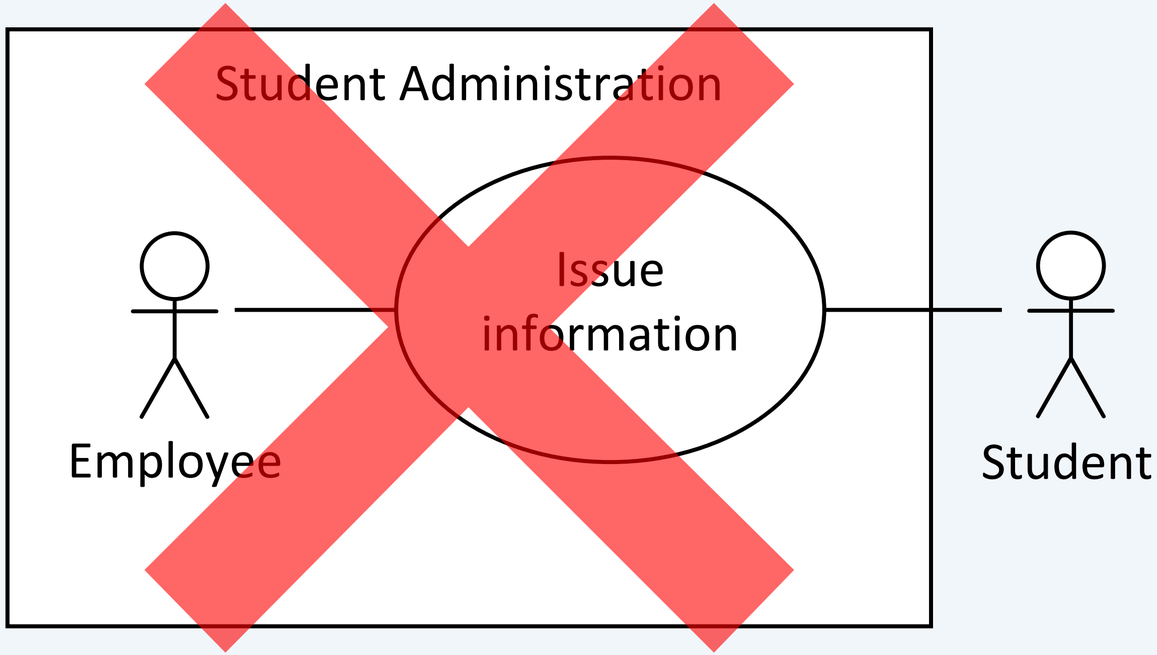
Processes

- Use case diagrams **do not model processes**!



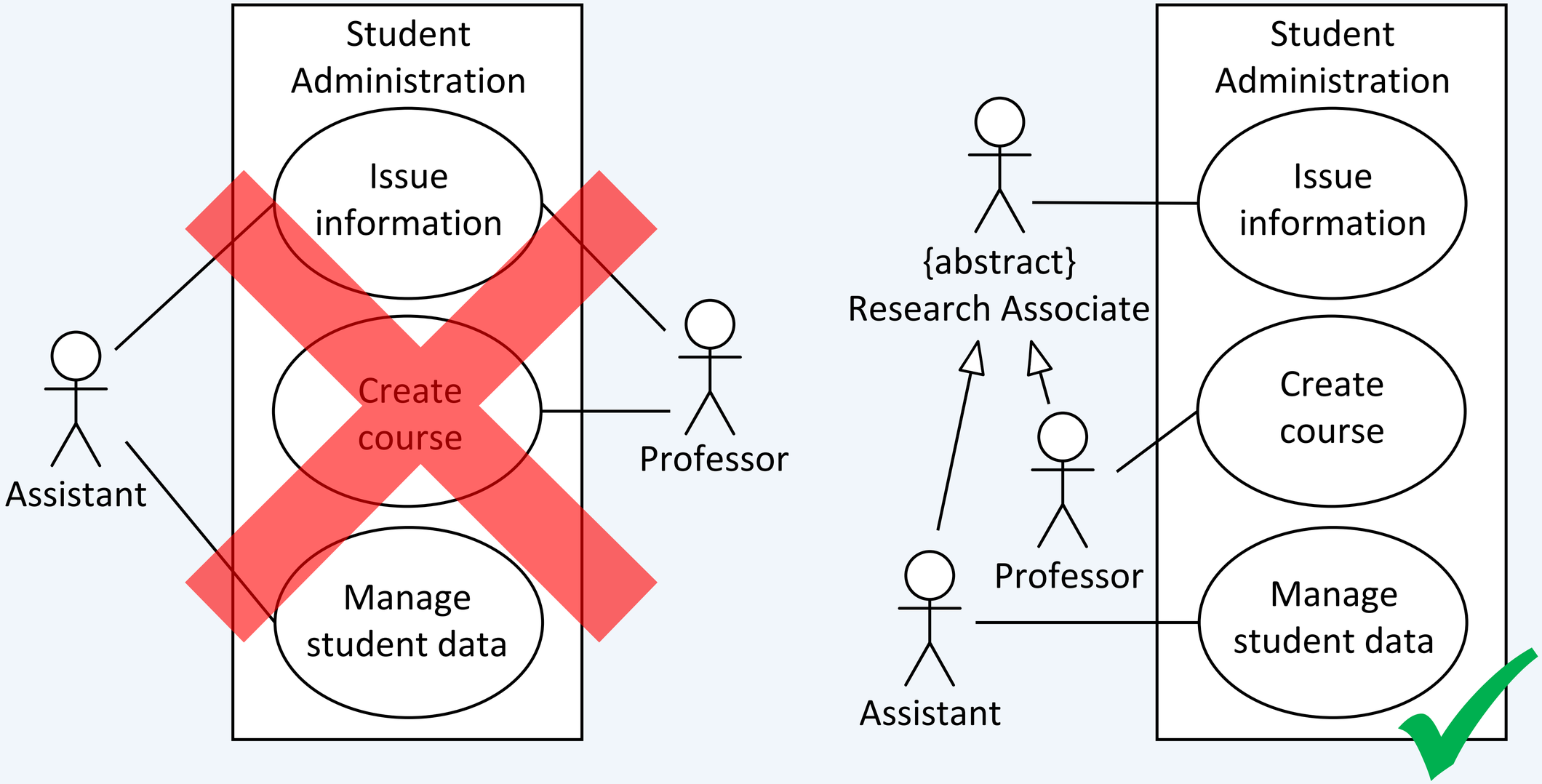
System Boundaries

- Actors are always outside the system boundaries!



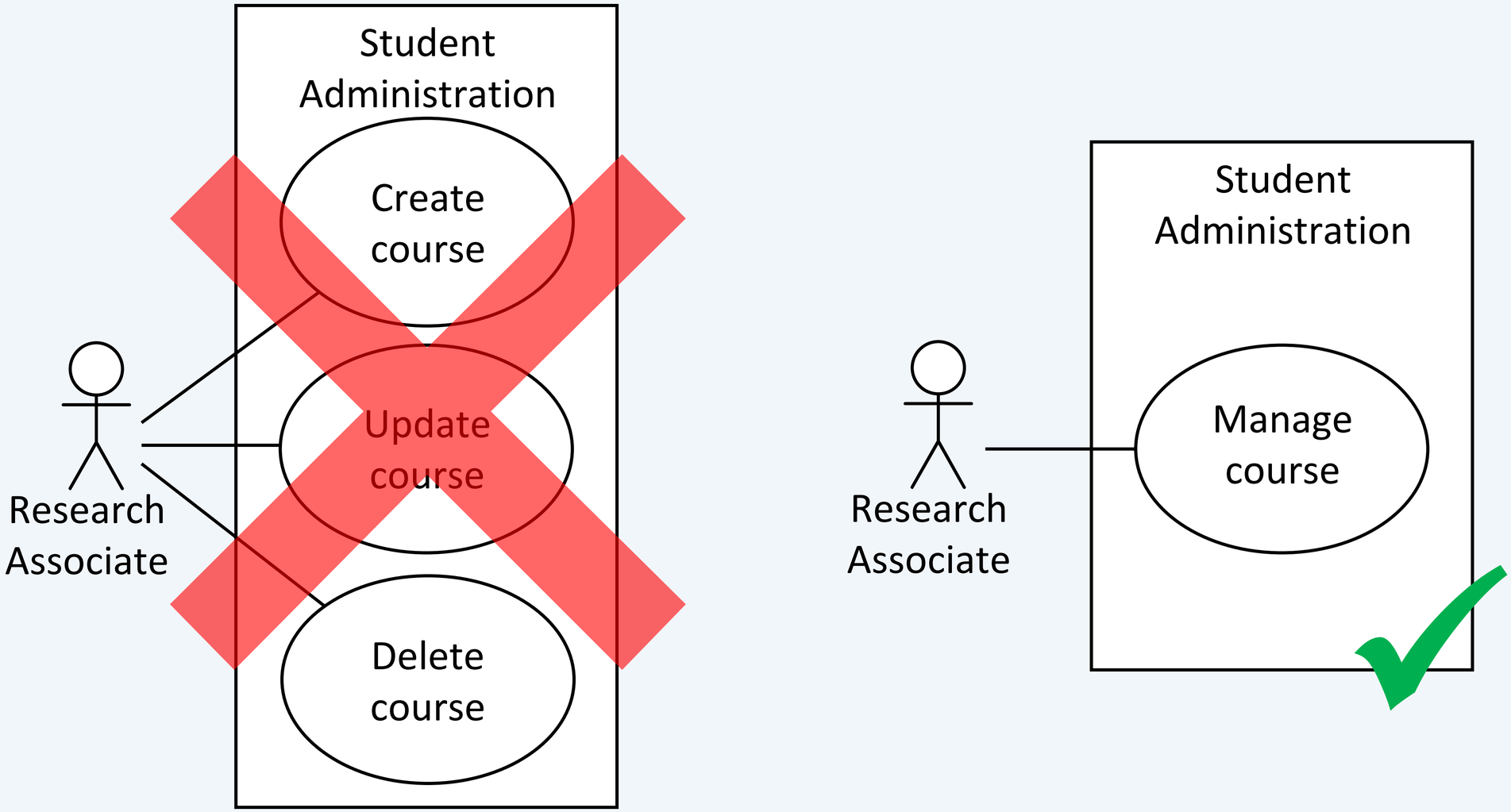
“and” or ”either or”

- Either **`Assistant`** or **`Professor`** are involved in **`Issue information`**



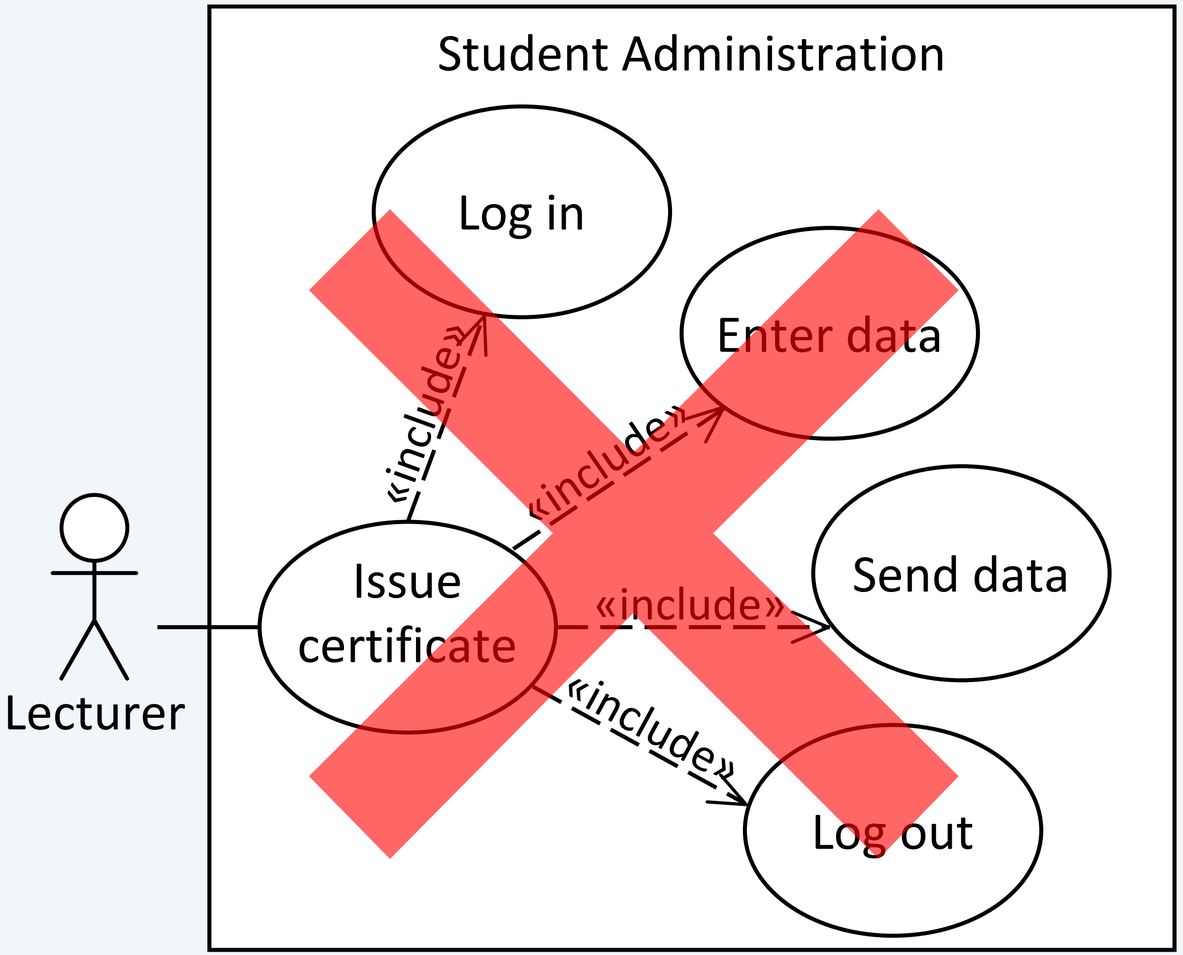
Level of Granularity

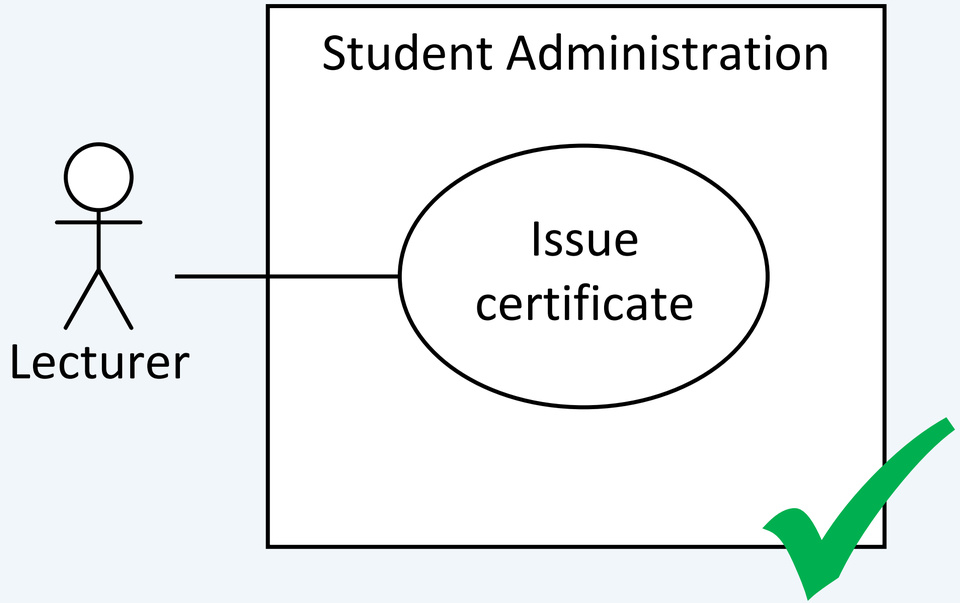
- Many small use cases are combined into one use case that contains the different possibilities



Functional Decomposition

- The individual steps are always executed together in a predefined sequence, so this is only a **single** use case ⇒ no functional decomposition





Use Case Diagram
Student Administration Example

Christian Huemer und Marion Scholz

Presented by Nicholas Bzowski

Example: Student Administration

- **Objective:** simplified representation of the “ student administration” system of a university

Verbal Description

- Many important administrative activities at a university are handled by the Student Administration Office. Students can enrol and register here, as well as withdraw from the study program.
- Students receive certificates here. Employees print out the certificates. Lecturers send certificate-relevant data to the Student Administration Office. Students are then automatically informed by the notification system.
- A distinction is made between 2 types of employees: a) those who deal exclusively with the administration of student data (**`service employee`**) and b) those who perform all other tasks (**`administration employee`**), but both groups provide information.
- **`administration employee`** issue certificates as soon as students collect them. They also create courses. When a course is created, a lecture hall can be reserved at the same time.

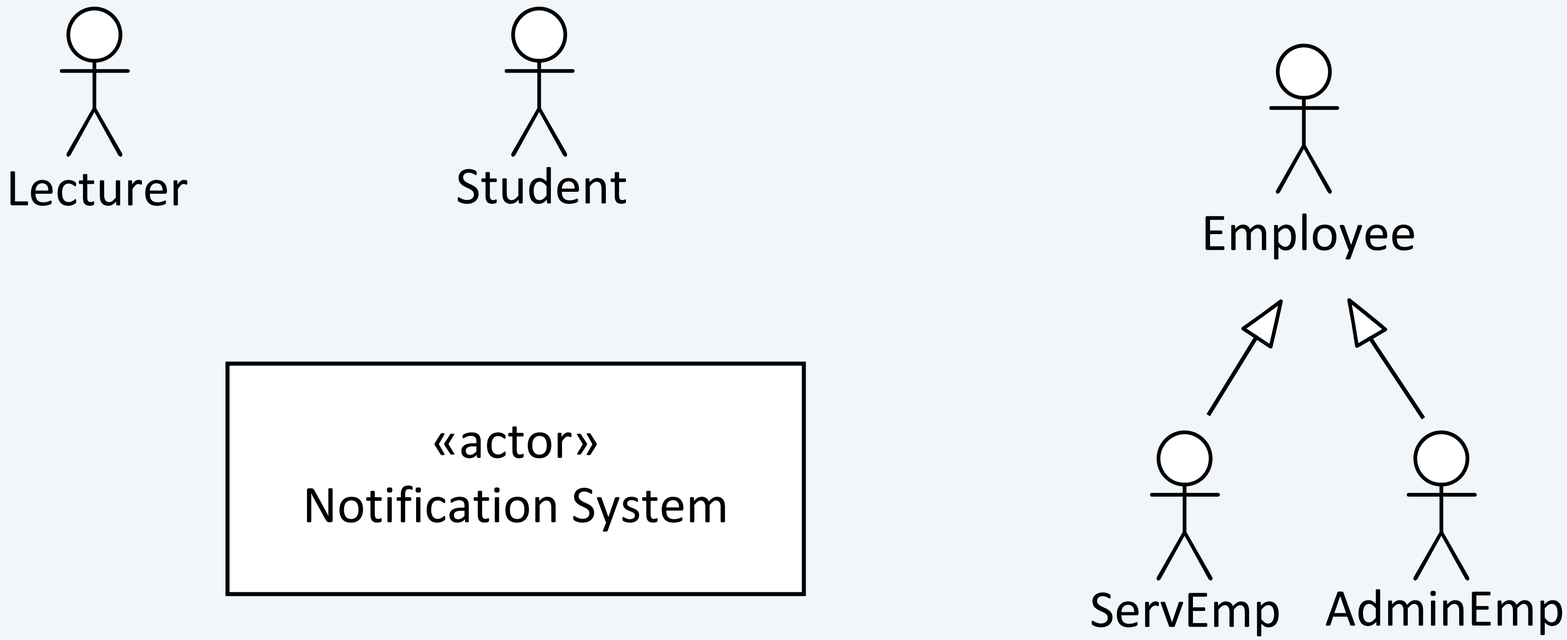
Example: Actors (1/2)

Who are the users?

- Many important administrative activities at a university are handled by the Student Administration Office. **Students** can enrol and register here, as well as withdraw from the study program.
- Students receive certificates here. Employees print out the certificates. **Lecturers** send certificate-relevant data to the Student Administration Office. Students are then automatically informed by the **notification system**.
- A distinction is made between 2 types of **employees**: a) those who deal exclusively with the administration of student data (**`service employee`**) and b) those who perform all other tasks (**`administration employee`**), but both groups provide information.
- **`administration employee`** issue certificates as soon as students collect them. They also create courses. When a course is created, a lecture hall can be reserved at the same time.

Example: Actors (2/2)

- Who are the users?



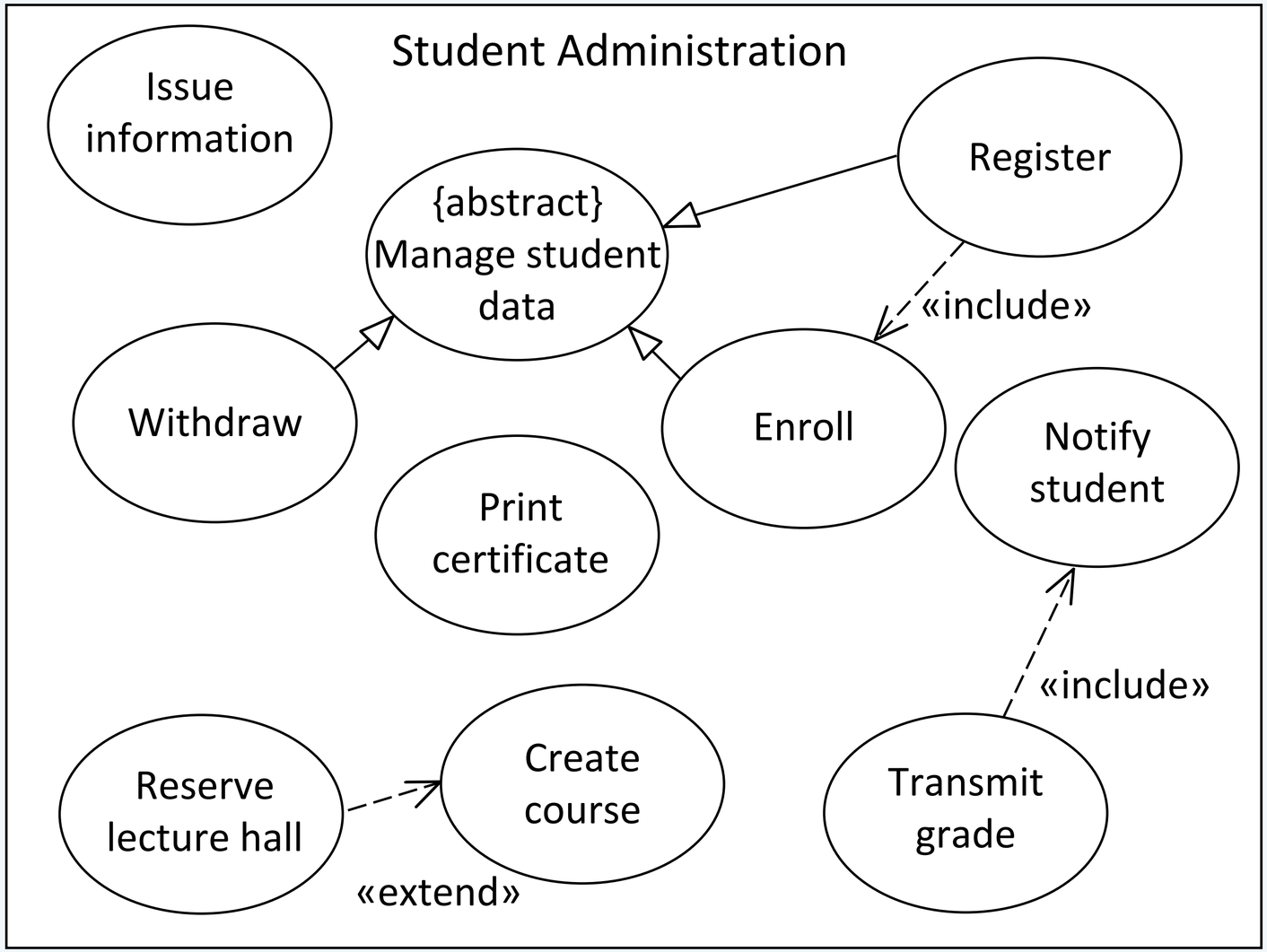
Example: Use Cases (1/2)

What services does the student administration office offer?

- Many important administrative activities at a university are handled by the Student Administration Office. Students can **enrol** and **register** here, as well as **withdraw** from the study program.
- Students receive certificates here. Employees **print out the certificates**. Lecturers **send certificate-relevant data** to the Student Administration Office. **Students** are then automatically **informed** by the notification system.
- A distinction is made between 2 types of employees: a) those who deal exclusively with the administration of student data (**`service employee`**) and b) those who perform all other tasks (**`administration employee`**), but both groups **provide information**.
- **`administration employee`** issue certificates as soon as students collect them. They also create courses. When a **course is created**, a **lecture hall can be reserved** at the same time.

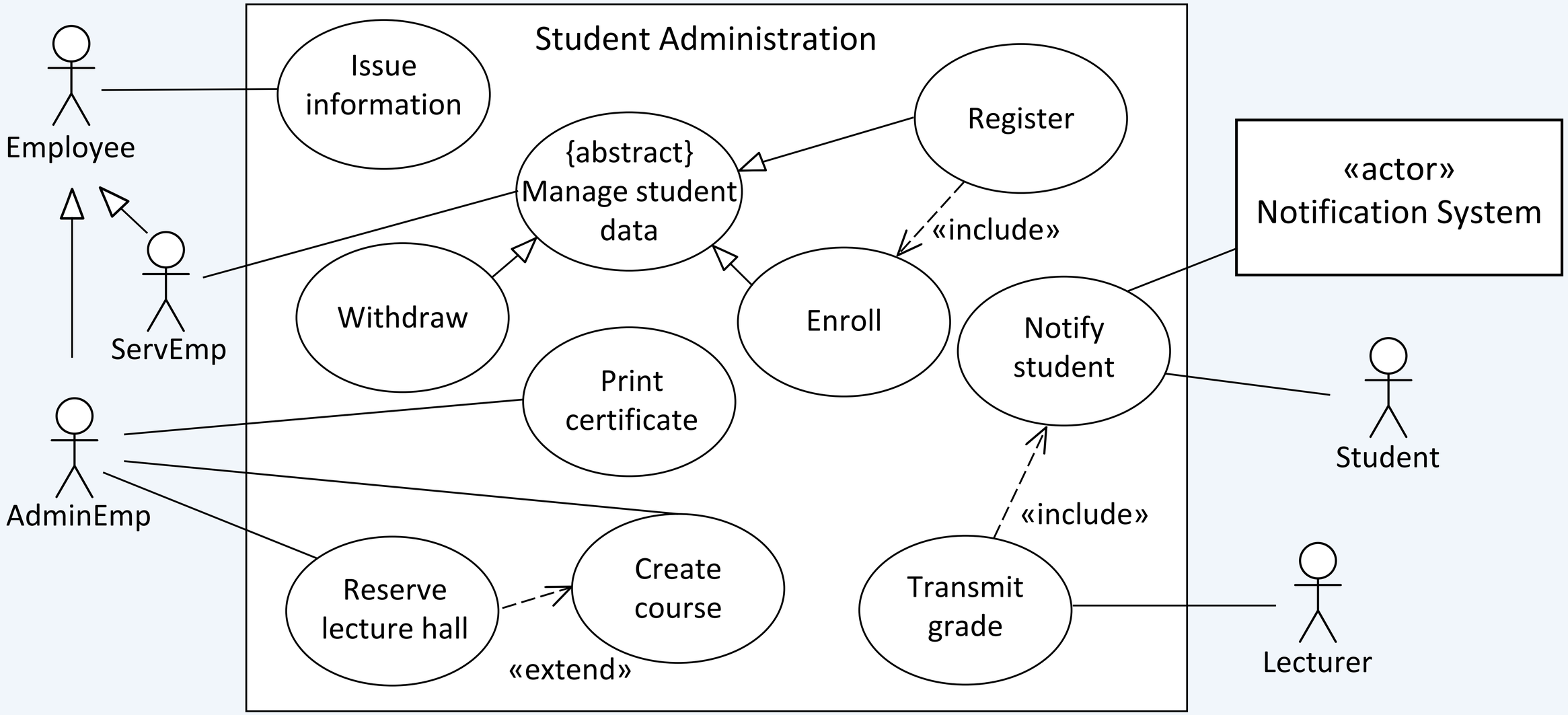
Example: Use Cases (2/2)

- What services does the student administration office offer?



Example: Associations

- Which actors use which use cases?



Example: Description (1/2)

- **Name:** Print certificate
- **Short description:** At the request of a student, an **`AdminEmp`** prints the certificate on paper.
- **Precondition:** Student has already been graded and all relevant data has already been submitted.
- **Postcondition:** Certificate is in printed form for the student
- **Error situations:** The printer does not work
- **System state in the event of an error:** Certificate was not printed
- **Actors who communicate with the use case**: **`AdminEmp`**
- **Trigger:** Student would like a printed certificate

Example: Description (2/2)

- **Standard process:**

  (1) Student goes to the study department and asks for a printed certificate.

  (2) **`AdminEmp`** enters the student number.

  (3) **`AdminEmp`** selects the certificate.

  (4) **`AdminEmp`** selects “print”.

  (5) System confirms that the certificate has been printed

  (6) Certificate is handed over to the student.

- **Alternative processes:**

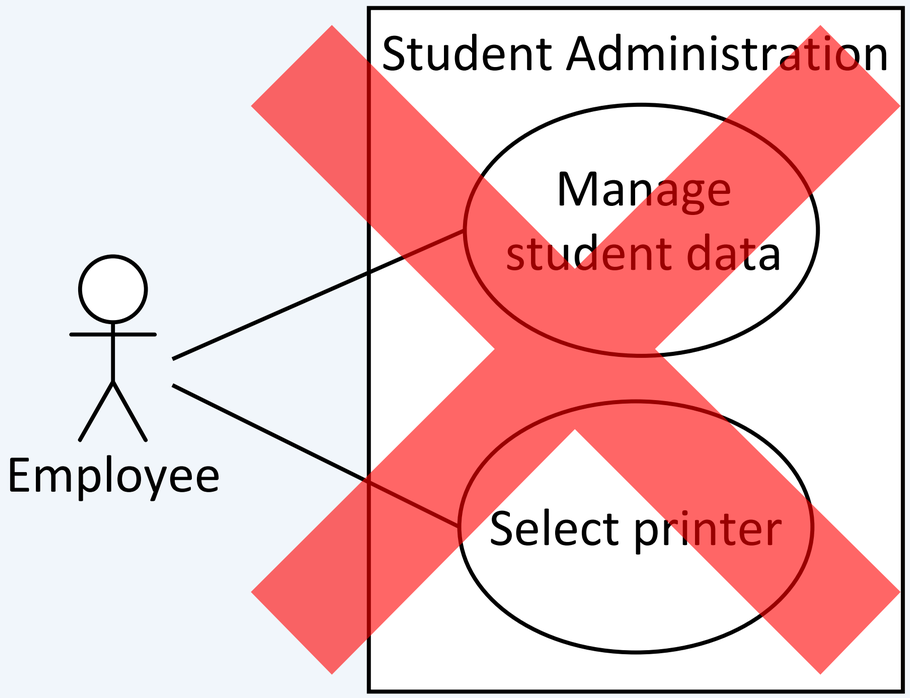
  (1') Student requests the printed certificate via email.

  (2-5) same as above

  (6') Certificate is sent by post

Example: Modeling Tips (1/2)

- The use cases should be at the same level of abstraction.



Example: Modeling Tips (2/2)

- Only use cases and actors that are relevant should be included.
- Of course, there are other people who have something to do with the student administration office, but they are negligible in this context of modeling.

- Ex.: