OO concepts

UML Representation

UNIVERSITÄT SALZBURG

Use Cases

UNIVERSITÄT SALZBURG

Use Case: First Artifact

- Use cases help in:
  - Better understanding of requirements
  - Documentation of requirements
- Use cases connect the different modelling views of a system

Use Case: Basis for Communication

Use Cases (scenarios) represent an **important communication conveyor**, by which the end users of a system and the developers exchange information.

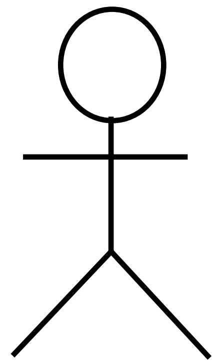
Components of a Use Case model:

- System functions (Use Cases)
- Environment (Actors)
- Relations between Use Cases and Actors (Use Case Diagrams)

UNIVERSITÄT SALZBURG

Actors

UML representation:



Examples of actors:

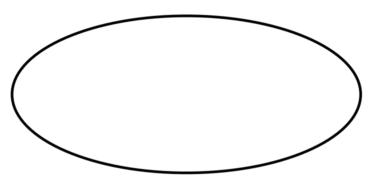
- Students who register for courses
- External account system
- Receptionist that serves a hotel reservation system

UNIVERSITÄT SALZBURG

Use Cases (I)

A Use Case models a dialogue between an actor and the system. It describes which functionality the system offers to an actor.

UML representation:



UNIVERSITÄT SALZBURG

Use Cases (II)

The following questions are helpful in defining use cases:

- What are the tasks of an actor?
- Will an actor produce information in the system, store it, change it, delete it, or just read it?
- Which use cases will produce, store, change, delete, or read this information?
- Does an actor have to be informed about certain events in the system?
- Can all functional requirements be fulfilled with the use cases ?

UNIVERSITÄT SALZBURG

Use Cases (III)

Example of the **short description** of a use case:

Name: *Student course registration*

This use case is started by a student. For a certain semester, a timetable can be read, changed, or deleted.

Flow of Events:

- Can be described in a text document
- Suggestion for a template:
  - Preconditions
  - Main flow and possible sub-flows
  - Alternative flows

UNIVERSITÄT SALZBURG

Use Cases (IV)

- Example: Selection (by professors) of offered courses

- Pre conditions:

  The use case "*Offering Courses*" must be achieved before this use case begins.
- Main flow

  This use case begins when a professor logs in the course management system and enters his/her password. The system verifies whether the password is valid (E-1) and requests the professor to select the current term or a future term (E-2). Afterwards the professor selects the desired activity: Add, delete, read, print or terminate.

UNIVERSITÄT SALZBURG

Use Case (V)

If adding is selected, the sub-flow S-1 is followed:

Add a course offer is performed.

…

- Sub-flows

S-1: Add a course offer

Course name and number can be entered by appropriate input fields
(E-3): The system connects the professor with the offered course
(E-4): The use case begins again.

…

- Alternative flows

(E-1): A wrong name or password was entered. The user can again enter both or terminate the use case.

Use Case Diagram (I)

Use case diagrams show some or all actors and use cases, as well as relations between these entities.

Typically there are:

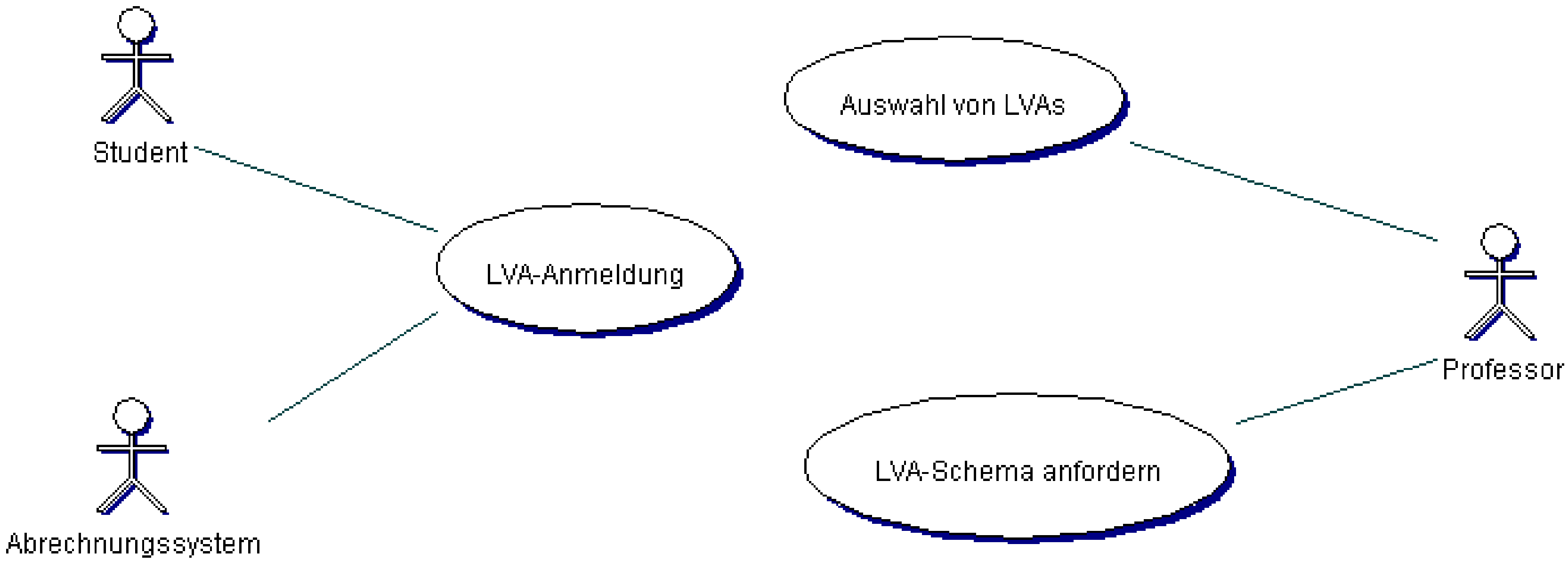
- A main use case diagram, which graphically depicts the most important actors and main functionality

- Further use case diagrams, e.g. :
  - A diagram that shows all use cases for a certain actor
  - A diagram that shows a use case and all its relations

UNIVERSITÄT SALZBURG

Use Case Diagram (II)

Example:



UNIVERSITÄT SALZBURG

Use Case Diagram (III)

- The “Uses” relationship shows that functionality in a use case is required in another use case.
- The “Extends” - relationship expresses optional behavior in a use case.

Both relations are represented by a dependence arrow and designated by **stereotyped names**.

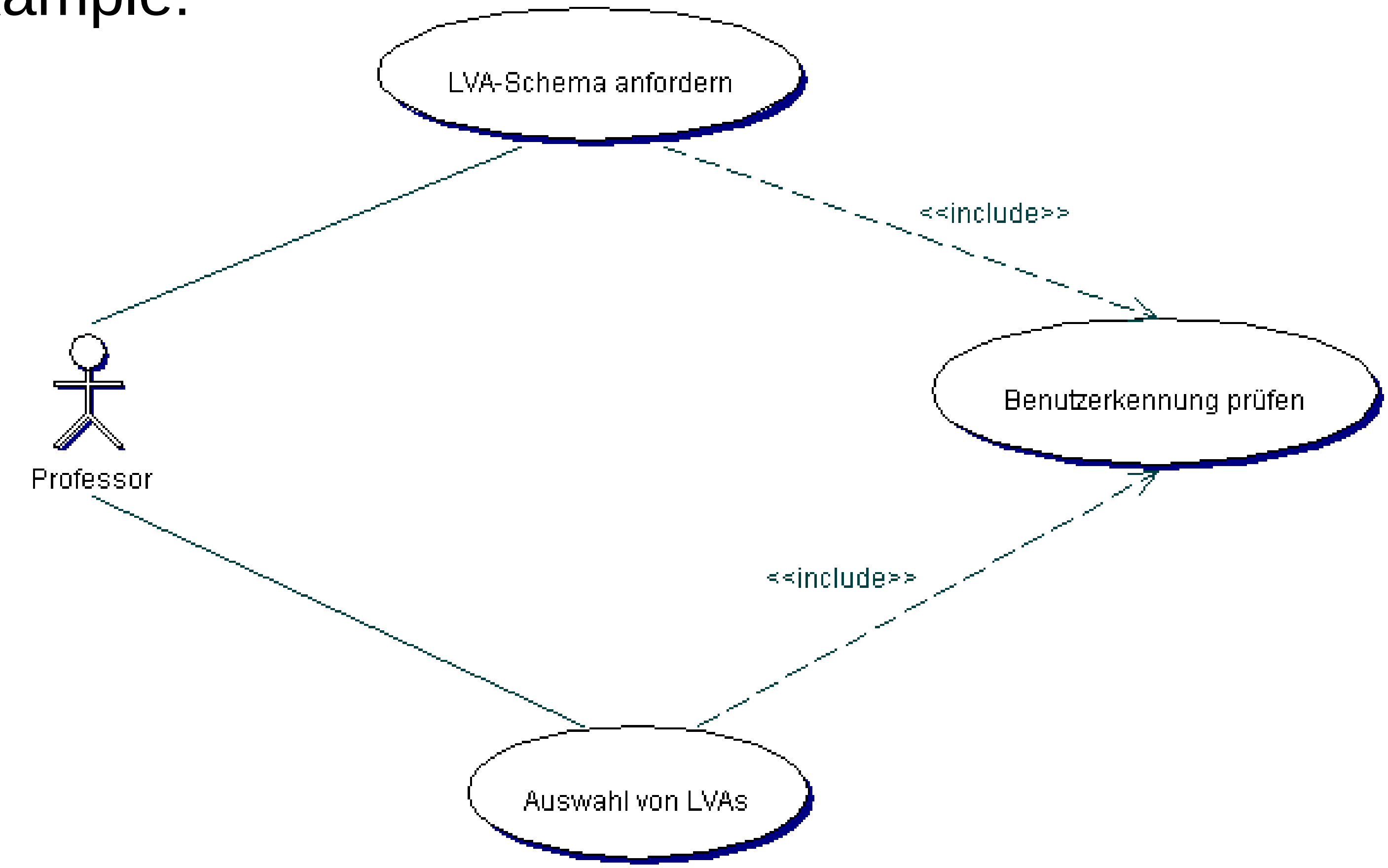
In UML there is the so-called **Stereotype** concept, which allows extensions of the fundamental modeling elements. The names of stereotypes are given between << and >>.

Stereotypes can be used to describe the relations between use cases.

UNIVERSITÄT SALZBURG

Use Case Diagram (IV)

Example:

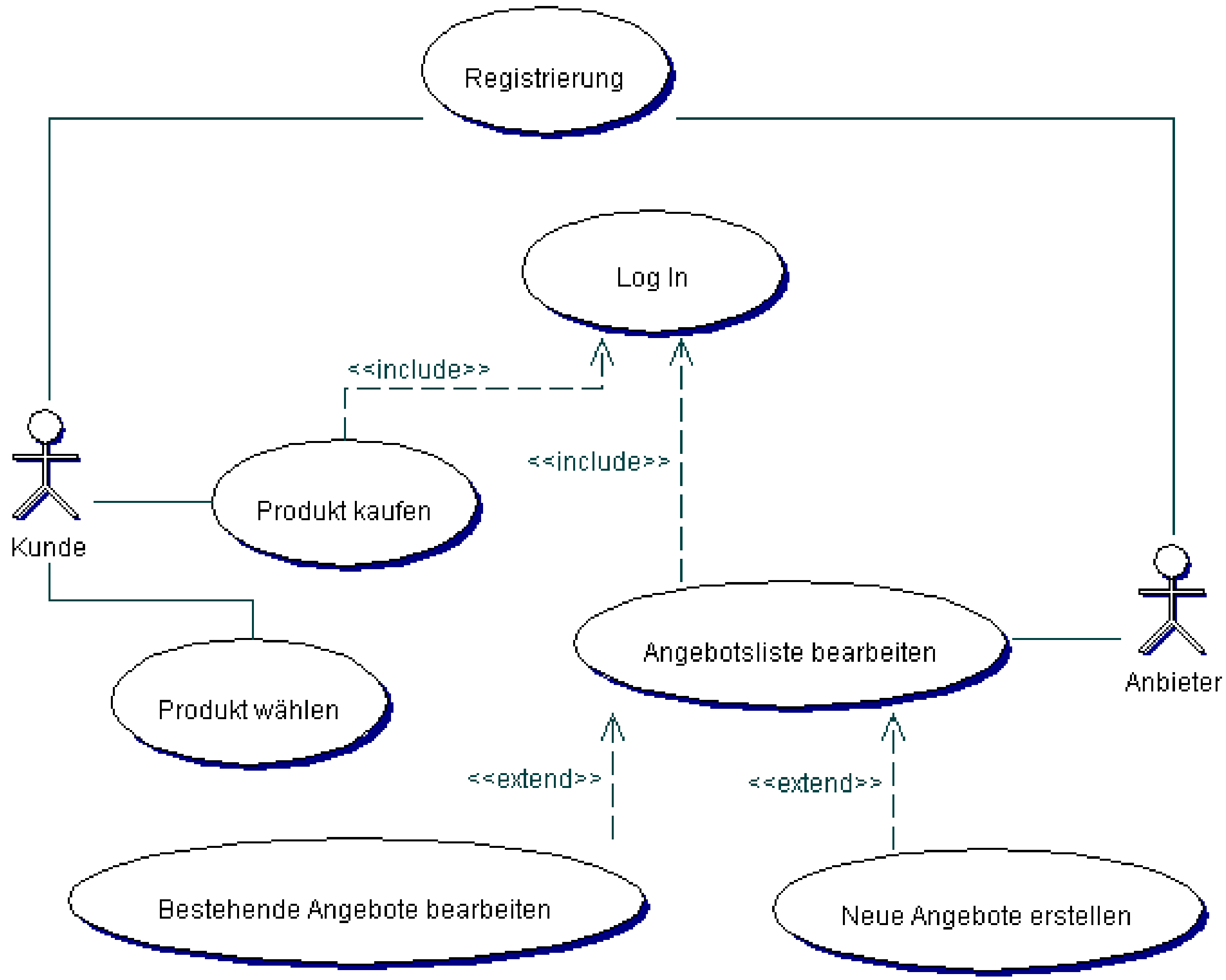


Hands-On Exercise (I)

- WebShop
  - Customer: -
    - Browses the offer
    - Selects a product
    - Pays with credit card or by bank transfer
  - Seller:
    - Introduces new products into the catalog
    - Removes old products from the catalog

UNIVERSITÄT SALZBURG

Hands-On Exercise (II)



UNIVERSITÄT SALZBURG

Hands-On Exercise (III)

- Registration: main flow

  The use case begins when the user selects the registration option.

  The system requests the user to fill out a form with its name, address, age, nickname and password (E-1).

  Afterwards the system sends an e-mail to the user to indicate a successful registration.

UNIVERSITÄT SALZBURG

Hands-On Exercise (IV)

- Alternative flows:
  - E-1: If the form is not completed, the user is requested to fill out the empty fields
  - E-1: If the nickname is already in use, the user is required to provide another nickname
  - ...

UNIVERSITÄT SALZBURG

CRC Cards

UNIVERSITÄT SALZBURG

CRC Cards (I)

- Which classes are used in order to model a scenario?
- How do these classes work together?

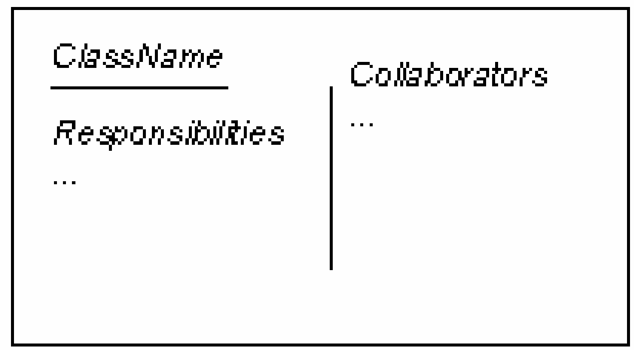
UNIVERSITÄT SALZBURG

CRC Cards (II)

- Class, Responsibility, Collaboration
- Beck and Cunningham, OOPSLA’89
  - Developed CRC-Cards in order to be able to descriptively teach the paradigm change from procedural to OO.
  - Direct introduction to the idea of Responsibility Driven Design (Wirfs-Brock 1990).

UNIVERSITÄT SALZBURG

CRC Cards (III)



- 4x6 Index Card
- Specifies:
  - Class name
  - Responsibilities
  - Collaborators

Example

Hotel	
Verwalte Kunden	Kunde
Verwalte Zimmer	Hotel-zimmer

Hotelzimmer	
Belegungsplan	Date, Reser-vierung
Erstelle Rechnungen	Kunde, Date

UNIVERSITÄT SALZBURG

CRC Cards (IV)

- Advantages
  - Communication between designers
  - From data containers to responsibilities
  - Collaboration between classes is more easily understood.
  - The card size determines a granularity of class description that enforces a high level specification of classes.

UNIVERSITÄT SALZBURG

Packages

and

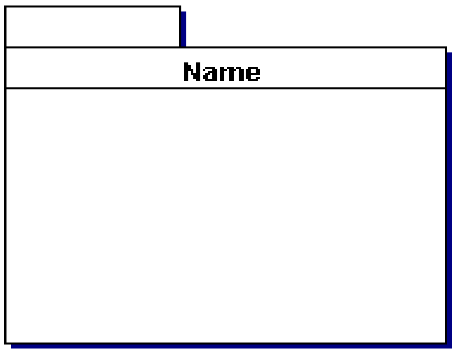
Package Diagrams

UNIVERSITÄT
SALZBURG

Packages

Packages are mainly used in order to group classes which belong together logically.

UML notation:



Packages (II)

Packages can be nested, in order to be able to better structure complicated architectures.

UML gives the option to list the names of the classes that belong to a package.

UNIVERSITÄT SALZBURG

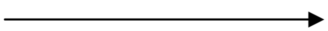
Package Diagrams

The following relations between packages can be defined:

- Dependence:

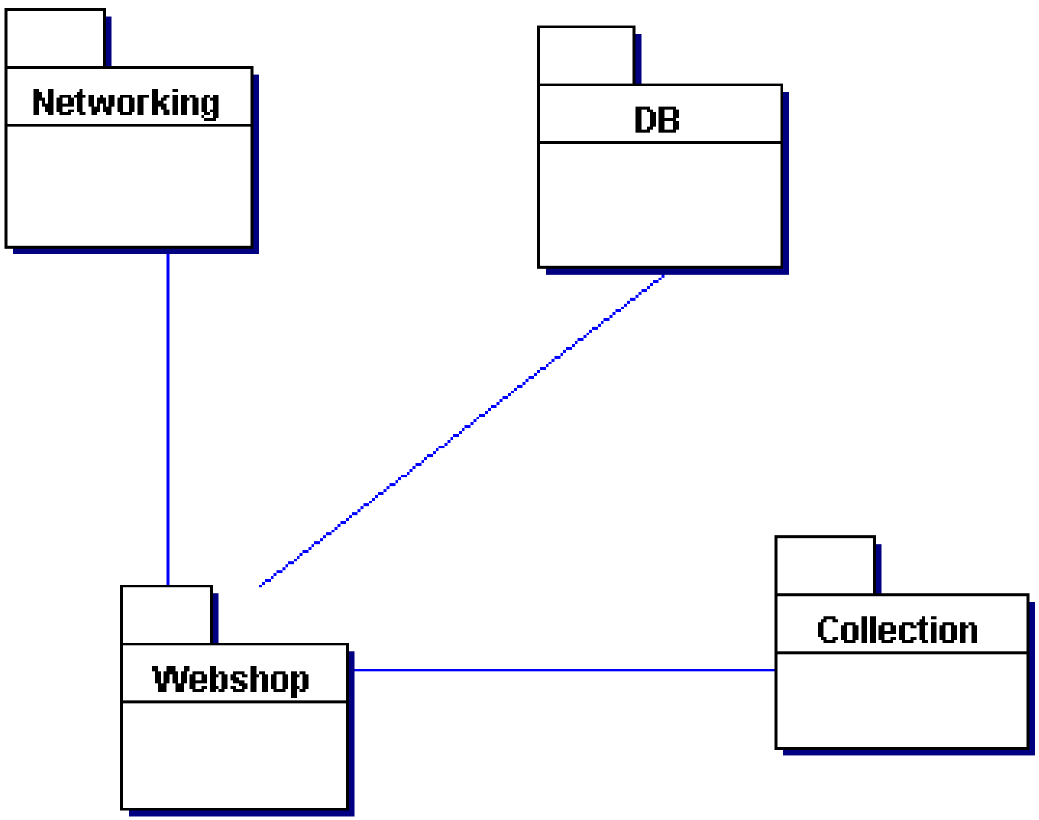
  --------------------->

  It is used to express that classes in a package use classes of another package.
- Generalization

  It is used to show that the classes in a package fulfill contracts of the classes of the other package

Example: E-Commerce Application



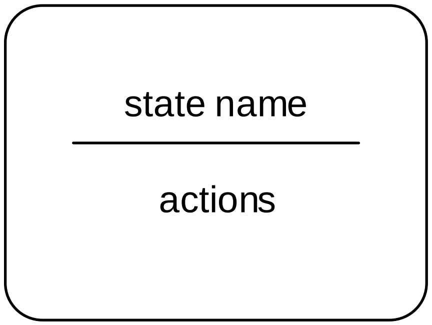
UNIVERSITÄT SALZBURG

State-Transition Diagrams

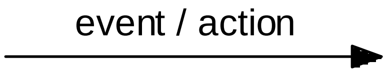
Notation Elements

State Transition Diagrams show the dynamic behavior of a class instance or of a whole system

- State symbol:



- Transition symbol:



UNIVERSITÄT SALZBURG

Notation Elements (II)

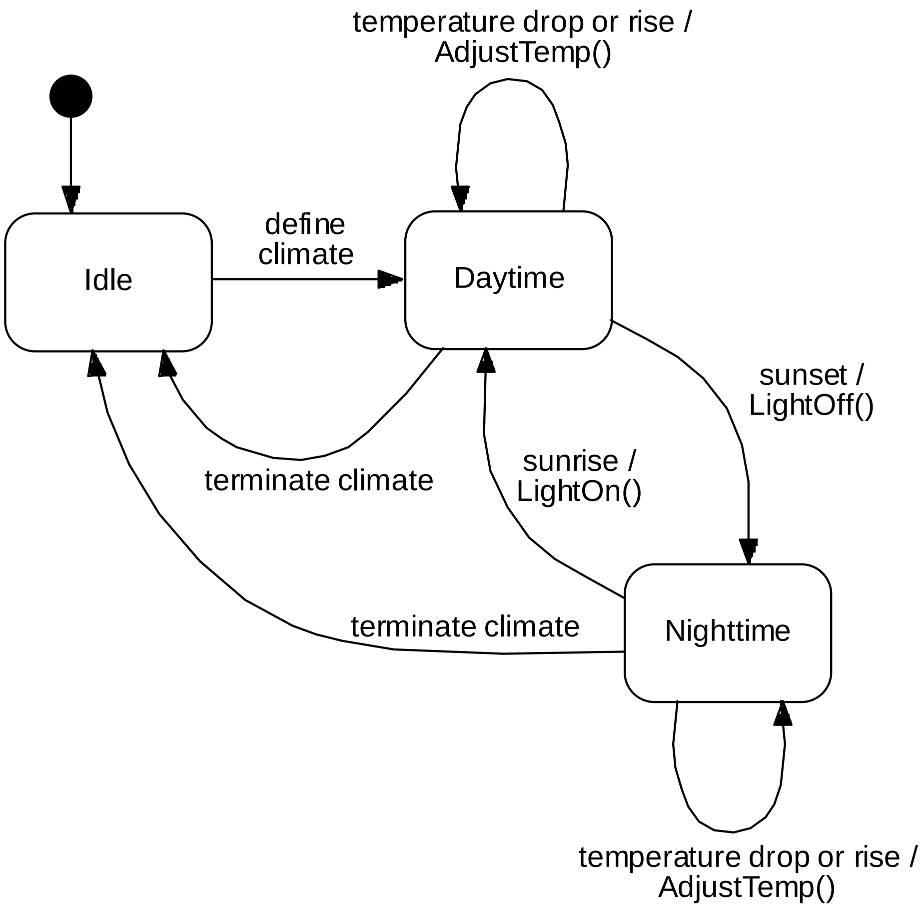
An action can be written as follows:

- Method call e.g. converter.ReadFile()
- Event triggering e.g. DeviceFailure
- Begin activity e.g. Start Converting
- Stop activity e.g. Stop Converting

UNIVERSITÄT SALZBURG

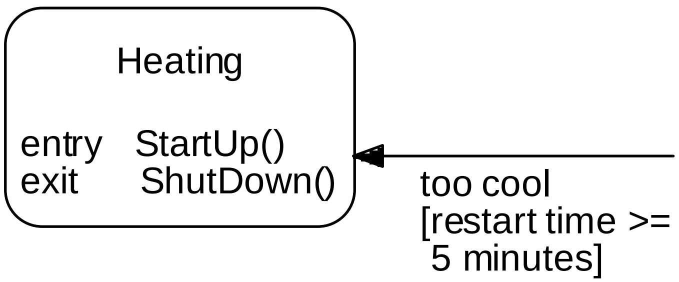
Example

- *Controller* in a greenhouse:



Additional notation (I)

- Actions can also be defined within a state:
  - If the system enters the state, if the system exits the state



  - If the system is in a state: e.g. do Heating
  - Transitions can have attached conditions (guards), which are indicated in square brackets.

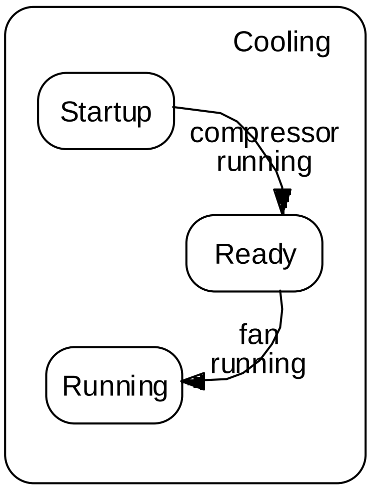
UNIVERSITÄT SALZBURG

Additional notation (II)

- Conditions can contain also time limits:

  timeout (Heating, 30s) TRUE, if system is longer than 30 sec. in the state Heating

- States can be nested, if needed:

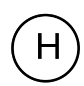


UNIVERSITÄT SALZBURG

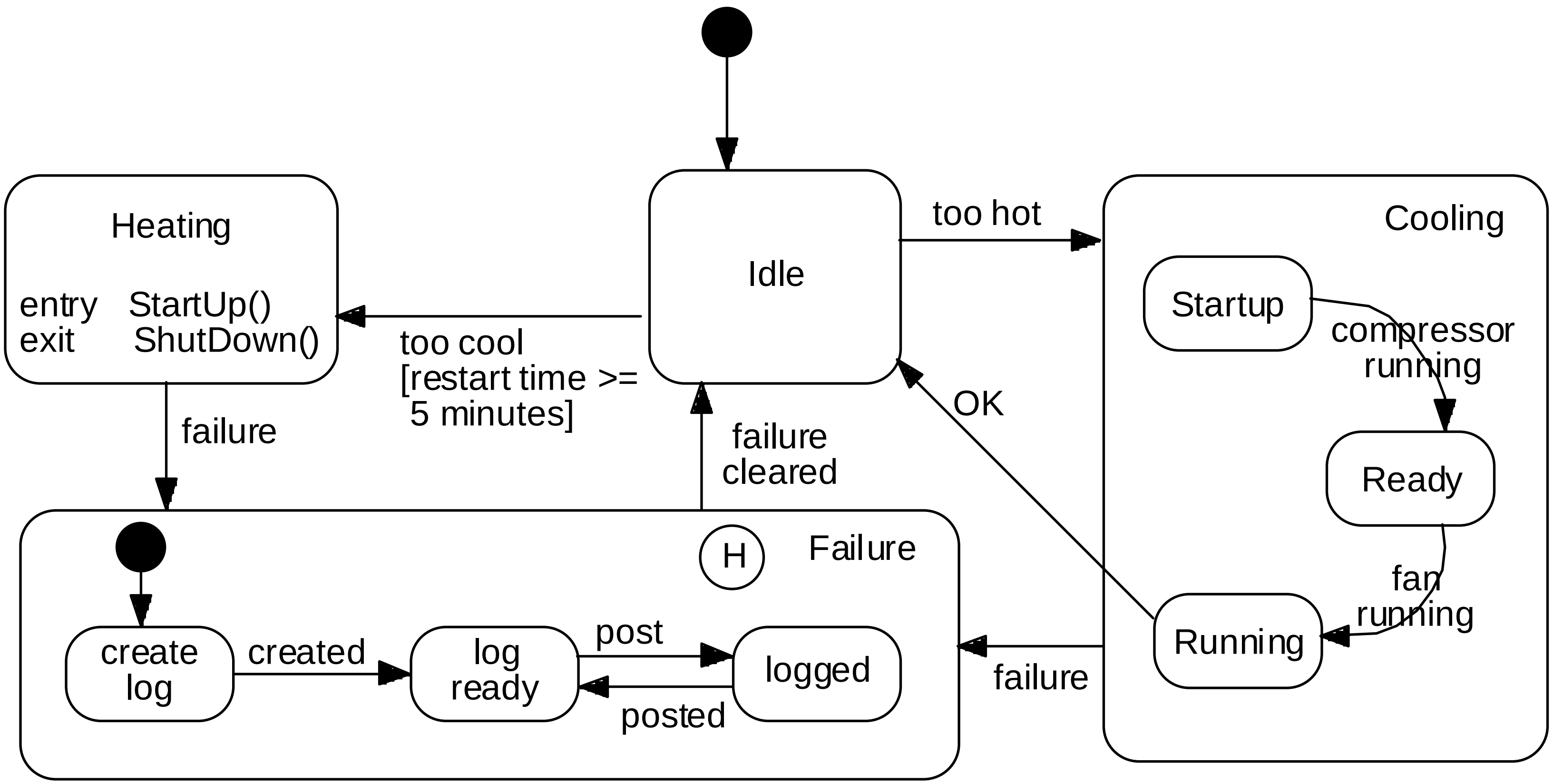
Additional Notation (III)

- State with history:
  - A state which contains sub-states may have a history mark
  - When the state is exited, the last active sub-state is remembered
  - When the state is re-entered, the last active sub-state is entered

  - History is indicated with the decoration



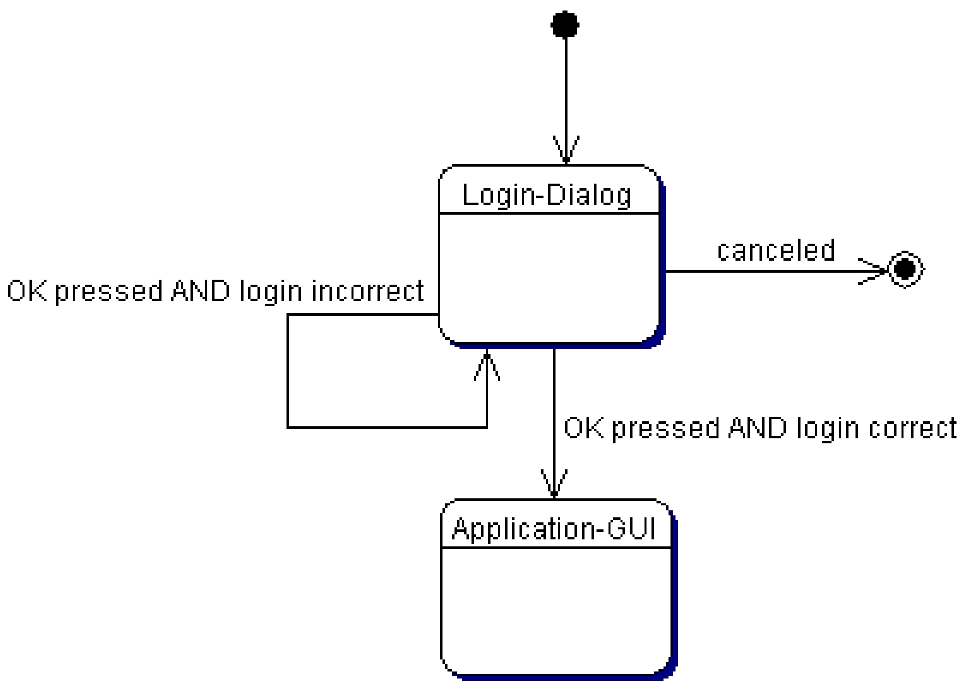
Example



UNIVERSITÄT SALZBURG

Example: GUI

Dialog sequence as state-transition diagram:



UNIVERSITÄT SALZBURG

Hands-On Exercise

Hands-On Exercise (II)

- Which states can a telephone have?
- Are there substates?
- Which transitions are there?
- Are there conditions for the transitions?

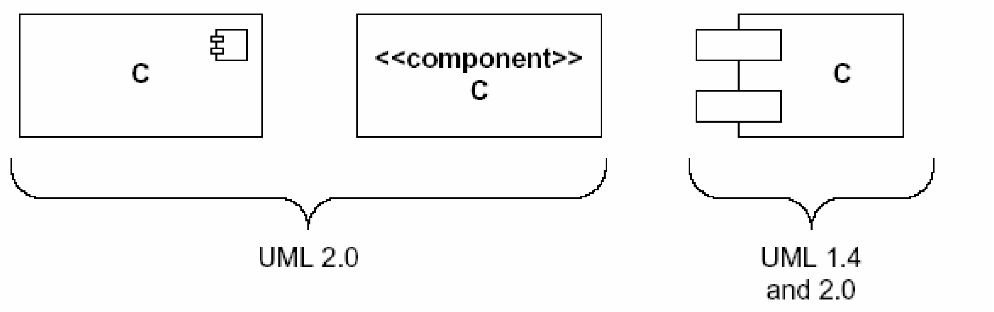
UNIVERSITÄT SALZBURG

Component Diagrams

UNIVERSITÄT SALZBURG

Components

Classes can be grouped in components. In UML, a component can be represented as follows:



Components correspond to modules in module-oriented languages.

C++: Reproduction of modules through .h, .c files

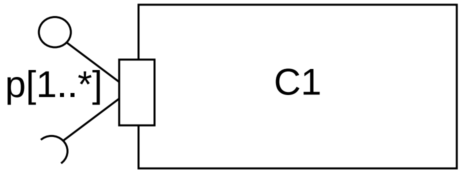
Smalltalk: Groups of classes, no modules

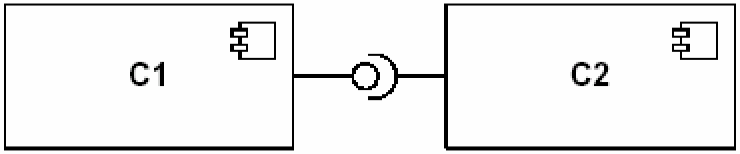
Oberon and Java: Modularity supported directly by the language

UNIVERSITÄT SALZBURG

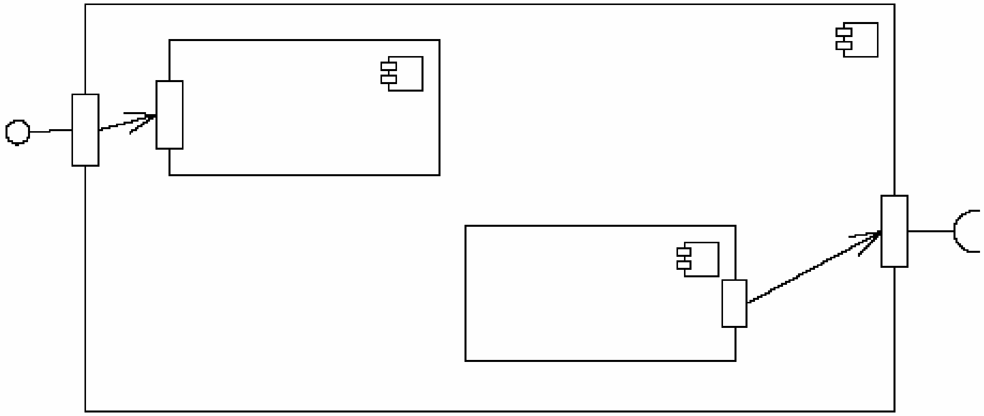
Ports, Interfaces and Connectors

- Ports: interaction points
- Interfaces:
  - Provided
  - Required





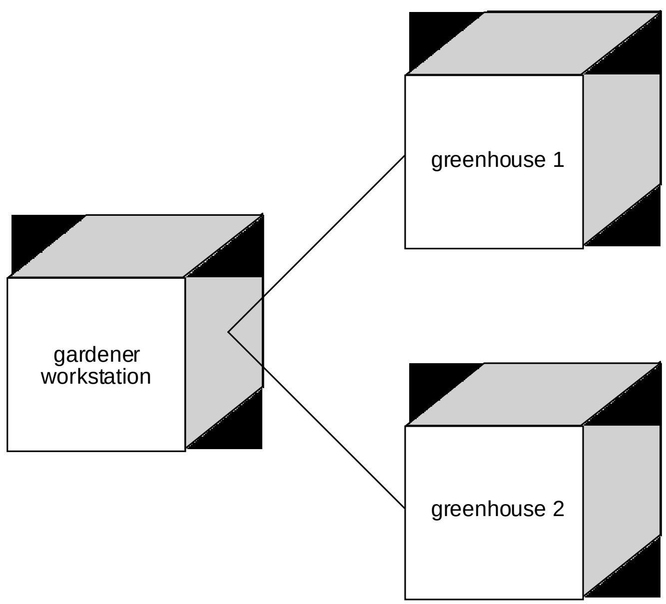
- Connectors:
  - Assembly
  - Delegation



Deployment Diagrams

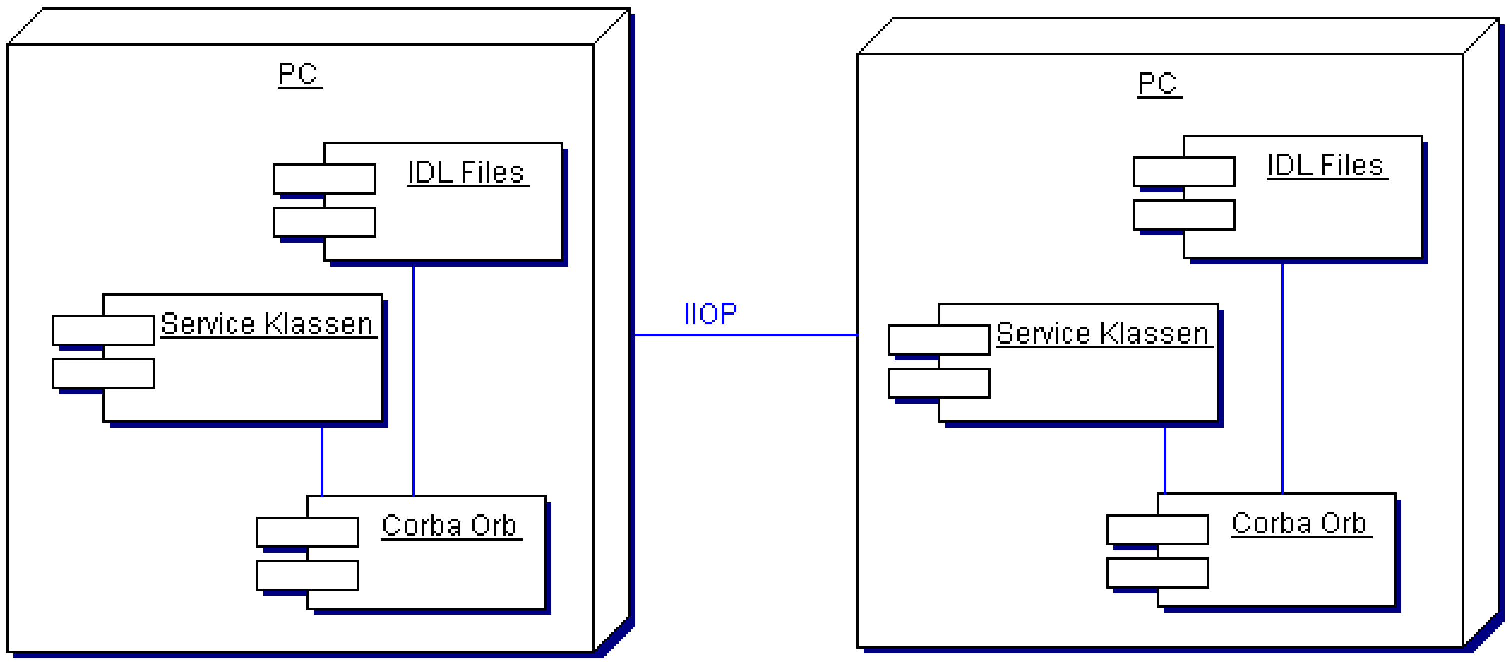
Notation

This representation is developed from Booch' s process diagram. It expresses the assignment of main programs and/or active objects to processors **for distributed systems running on multiple processors**.



UNIVERSITÄT SALZBURG

Example: CORBA

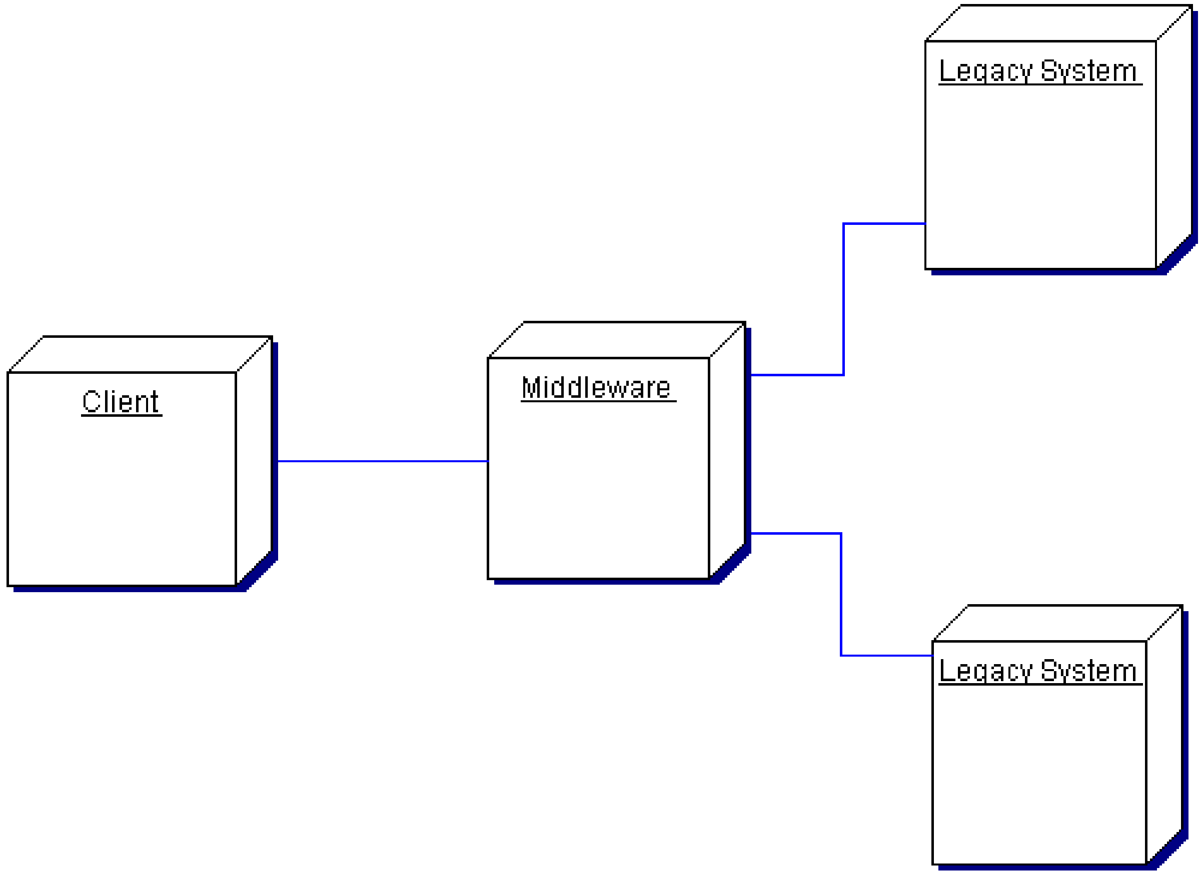


Hands-On Exercise: Web Shop

- A Webshop is typically a distributed application. Normally three layers are involved.
- How could the topology of the system look?
- Which components are on which computational nodes?

UNIVERSITÄT SALZBURG

Three-tier Architecture



Web Shop: Topology