

3.1 Use Case Diagrams

Subject/Topic/Focus:

- Introduction to Use Cases

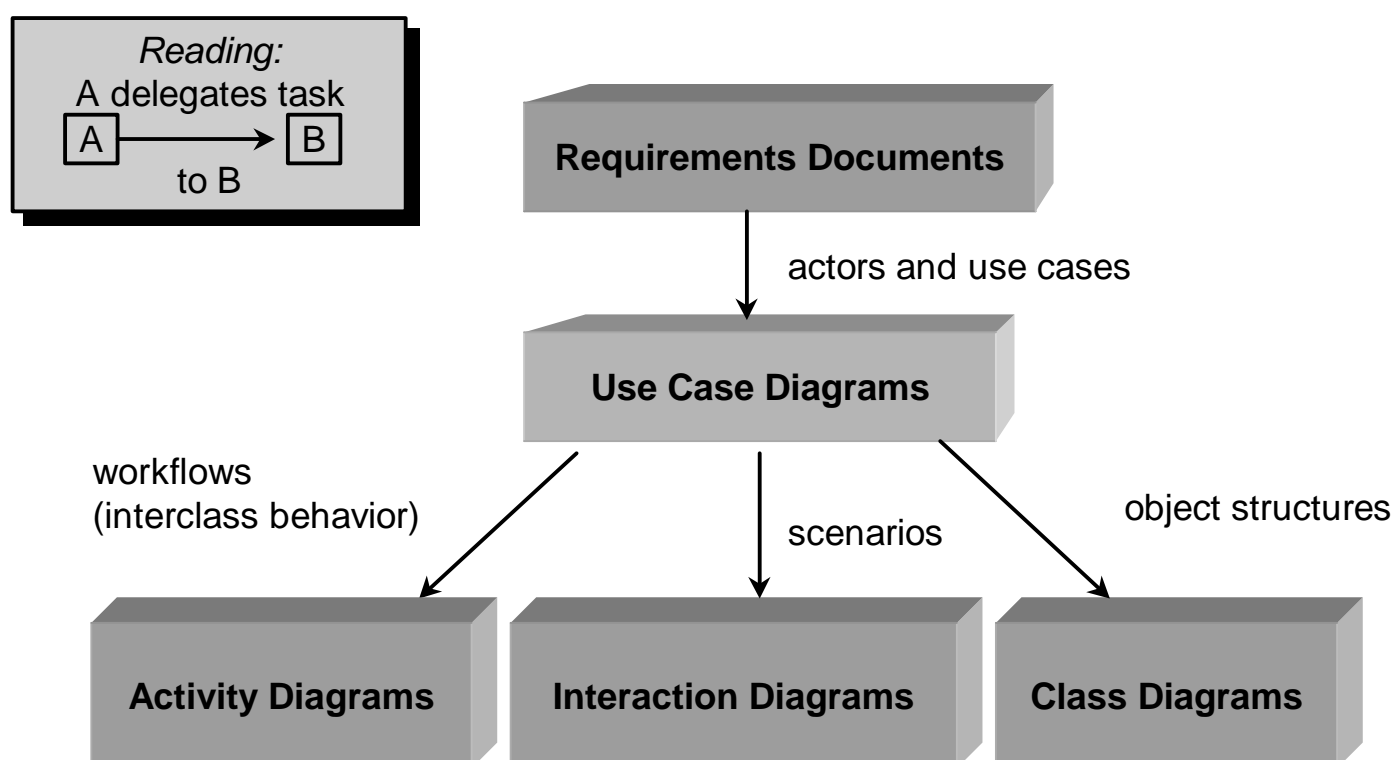
Summary:

- System Boundary
- Actors
- Use Cases
- Generalization, Inclusion, Extension

Literature:

- [Fowler99], UML Distilled, Second Edition
- [Booch98]

Role of Use Case Diagrams in UML



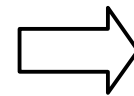
Scenarios, Workflows, Object Structures

- ❑ A certain **path** through a use case is called a **scenario**.

A scenario shows a particular set and combination of conditions within that use case.

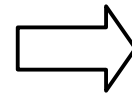
E.g., ordering some goods:

- Scenario 1: All goes well.
- Scenario 2: There are not enough goods.
- Scenario 3: Our credits are insufficient.



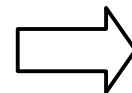
Interaction Diagrams

- ❑ **Processes** involving different use cases are shown in **workflows**, e.g., from ordering to delivery and payment.



Activity Diagrams

- ❑ The **structure** of the objects (and actors) that occur in use cases are described by **classes**, e.g., customers, goods, invoices.



Class Diagrams

Use Case Model

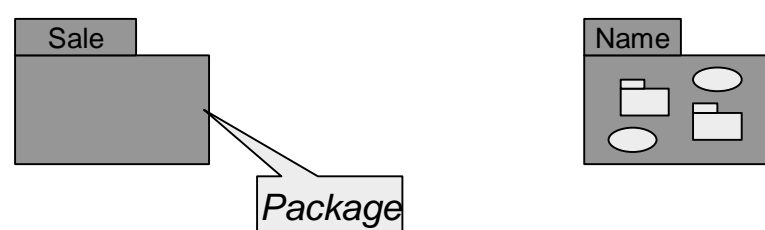
Requirements are described in UML by a **use case model**.

A use case model consists of

- ❑ single use case diagrams
- ❑ further (nested) **packages** of use case diagrams

The supreme package of the nested packages is the use case model.

Notation:



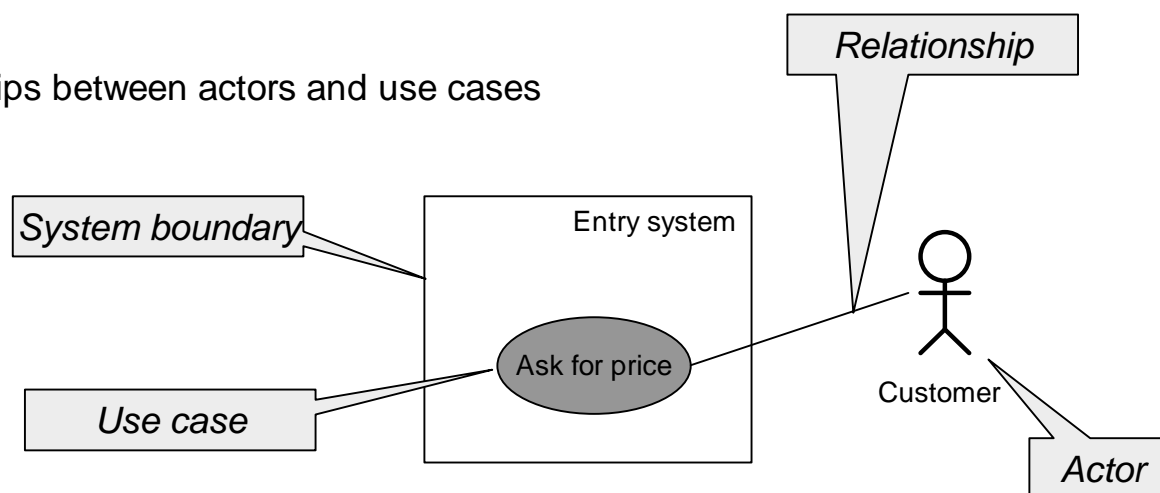
Use Case Diagrams

A **use case diagram** shows the relationship between actors and use cases in a system.

A use case diagram for a system consists of

- ❑ The system boundary
- ❑ Actors
- ❑ Use cases
- ❑ Relationships between actors and use cases

Notation:



Use Case Diagrams

3.1.5

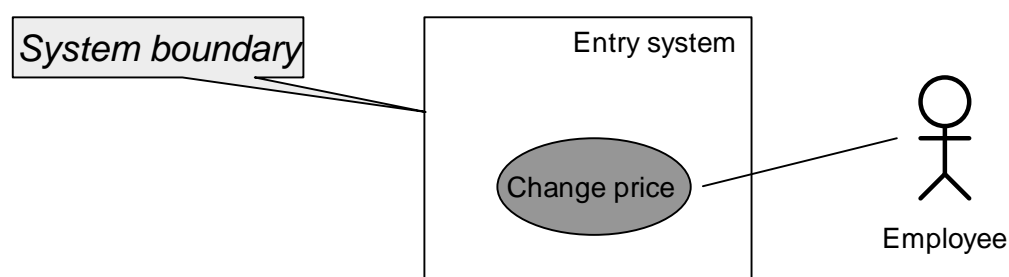
System Boundary

The use case view describes the functionality of a system from an *outside* point of view, that means from the *user's point of view*.

Only the interactions between actors (users) and system are shown, what happens inside the system is hidden.

This boundary is clarified by the **system boundary**.

Notation:



Observation: In further use cases a system can appear as an actor again, e.g., an entry system (of a travel agency) acting with another system (flight reservation system).

Use Case Diagrams

3.1.6

Actors

An **Actor** is an abstraction for an object outside the system interacting with the system, e.g., a user. This enables:

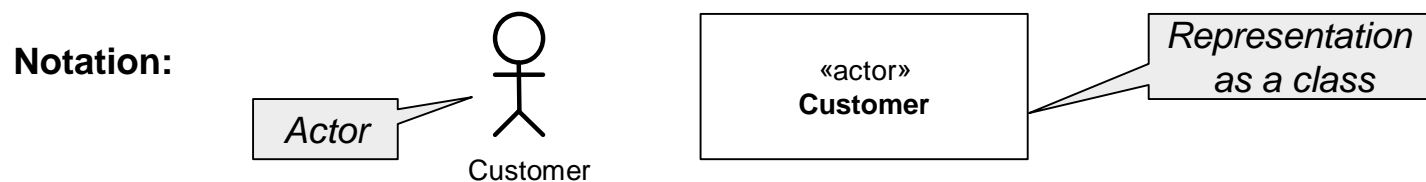
- several roles for one user
- several users with the same role
- systems as actors.

Properties of an actor are:

❑ **Name:** noun

❑ **Description:**

- textual description of its role, operations, faculties, protocols, ...
- all descriptions together built the **actors-catalog**



Use Case Diagrams

3.1.7

Use Cases (1)

A **use case** is the specification of a sequence of interactions between an actor and the computer system which should be inspected.

Use cases describe *user-visible* functionality.

An **instance** of a use case (instantiation) is the execution of the described interactions.

A use case is defined by:

❑ **Name:** Verb (or noun), name of the procedure

❑ **Priority:** Priority in development process

❑ **Description:**

- textual description of the interaction between user and system in reference to the requirements specification (e.g., /LF70/)
- sequence of the interaction steps using the terms of a **glossary**
- all descriptions together built the **use case catalog**.

Use Case Diagrams

3.1.8

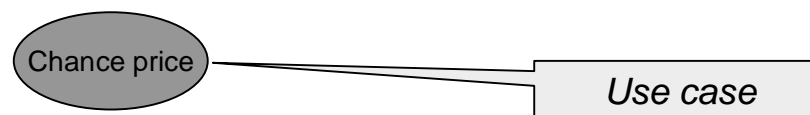
Use Cases (2)

Description of a use case in the use case catalog:

Example: Use case "Change price"

1. The employee searches the desired product.
2. The system shows all product data.
3. The employee enters the new price.
4. The system proofs if the price is in the limits.
5. The system proofs if actual orders are concerned and asks whether they should be changed to the new price. The employee can cancel the changes here.
6. The new price becomes active.

Notation:



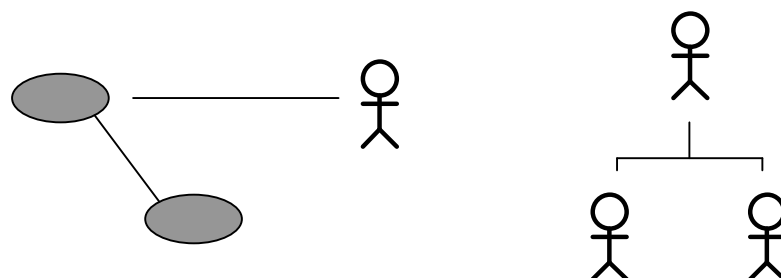
Use Case Diagrams

3.1.9

Relationships

Relationships between use cases and actors

- are represented by directional or non-directional arrows with different arrow peaks,
- may be annotated by **stereotypes**;
Notation: « stereotype » ,
- may consist between two use cases,
- may consist between two actors or
- may consist between an use case and an actor.



Use Case Diagrams

3.1.10

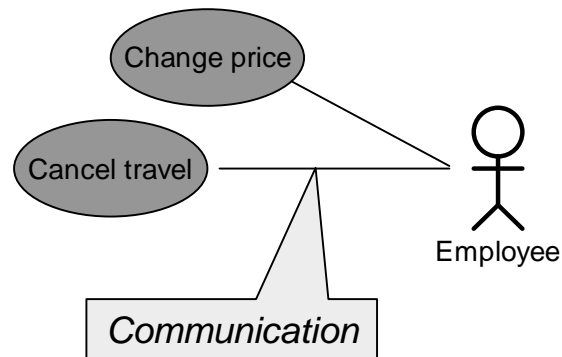
Communication

The **communication** between an actor and the system is represented by a edge between actor and use case.

An actor can be associated with several use cases and vice versa.

Exact term: **Actor-Use Case Communication**

Notation:



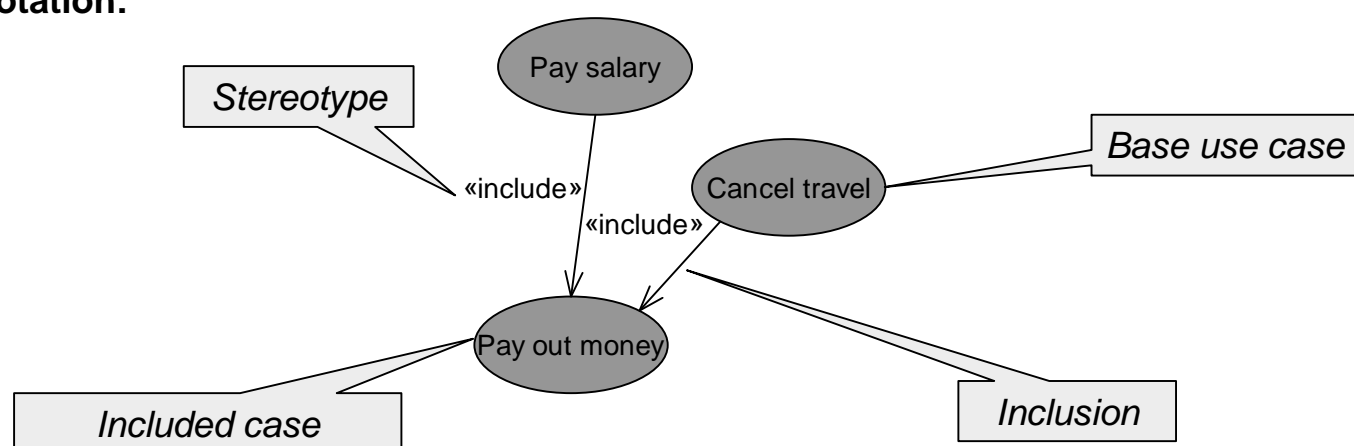
Possible meanings are: „A is concerned in B“, „A releases B“, „A communicates with B“, „A uses B“, ...

Include Use Cases

Include means the inclusion of particular action sequences in the base use case.

The included use case can be used independent of the base use cases.

Notation:

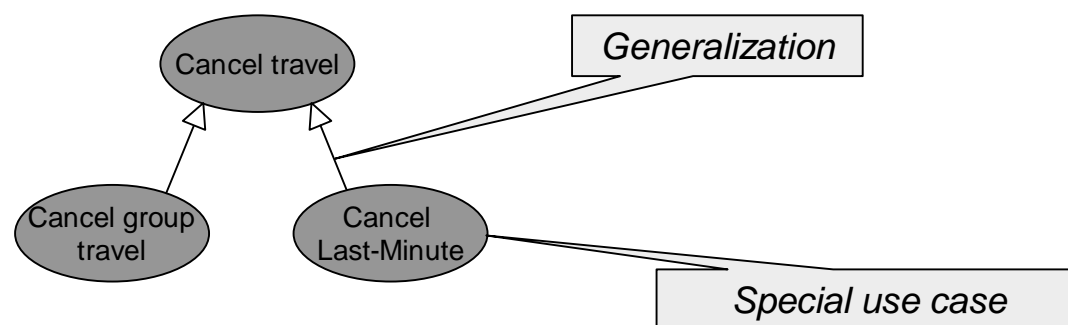


Generalization of Use Cases

Generalization correlates (taxonomically) more specialized use cases and more general ones. The special case inherits **every** property from the general and adds incrementally further properties or **replaces** them.

Special use cases can substitute general ones.

Notation:



Use Case Diagrams

3.1.13

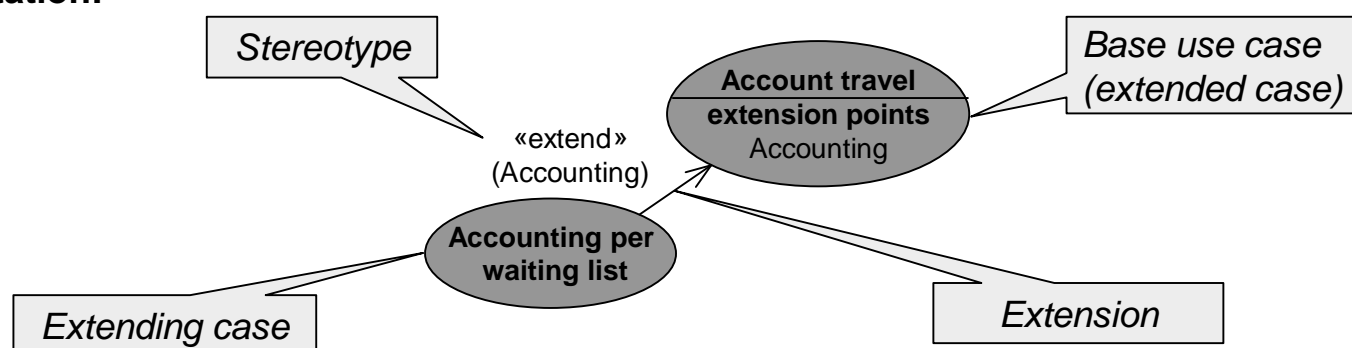
Extend Use Cases

Extensions define variations and special cases with the meaning, that the extending case completes the base use case, i.e., may be inserted in the behavior of the base case.

Extensions are included, maybe optional, at **extension points**.

- A use case may have many extension points
- An extending use case may extend one or more of these extension points

Notation:



Use Case Diagrams

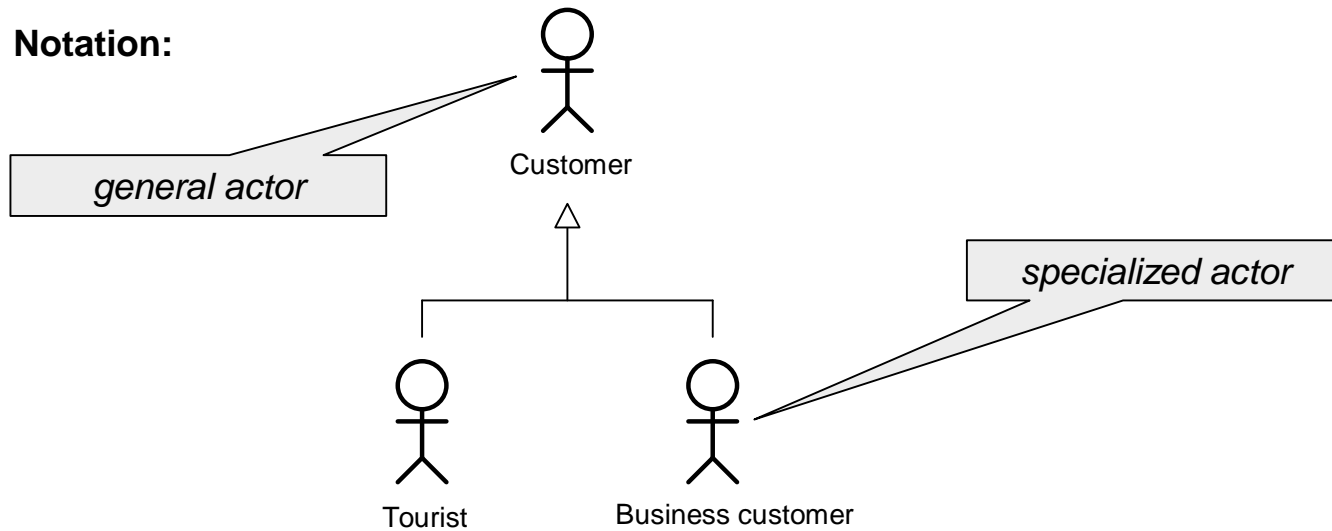
3.1.14

Generalization of Actors

Actors may have a **taxonomic relationship**.

In use case diagrams the taxonomy is usually not shown. Separate actor diagrams show the relationships between actors.

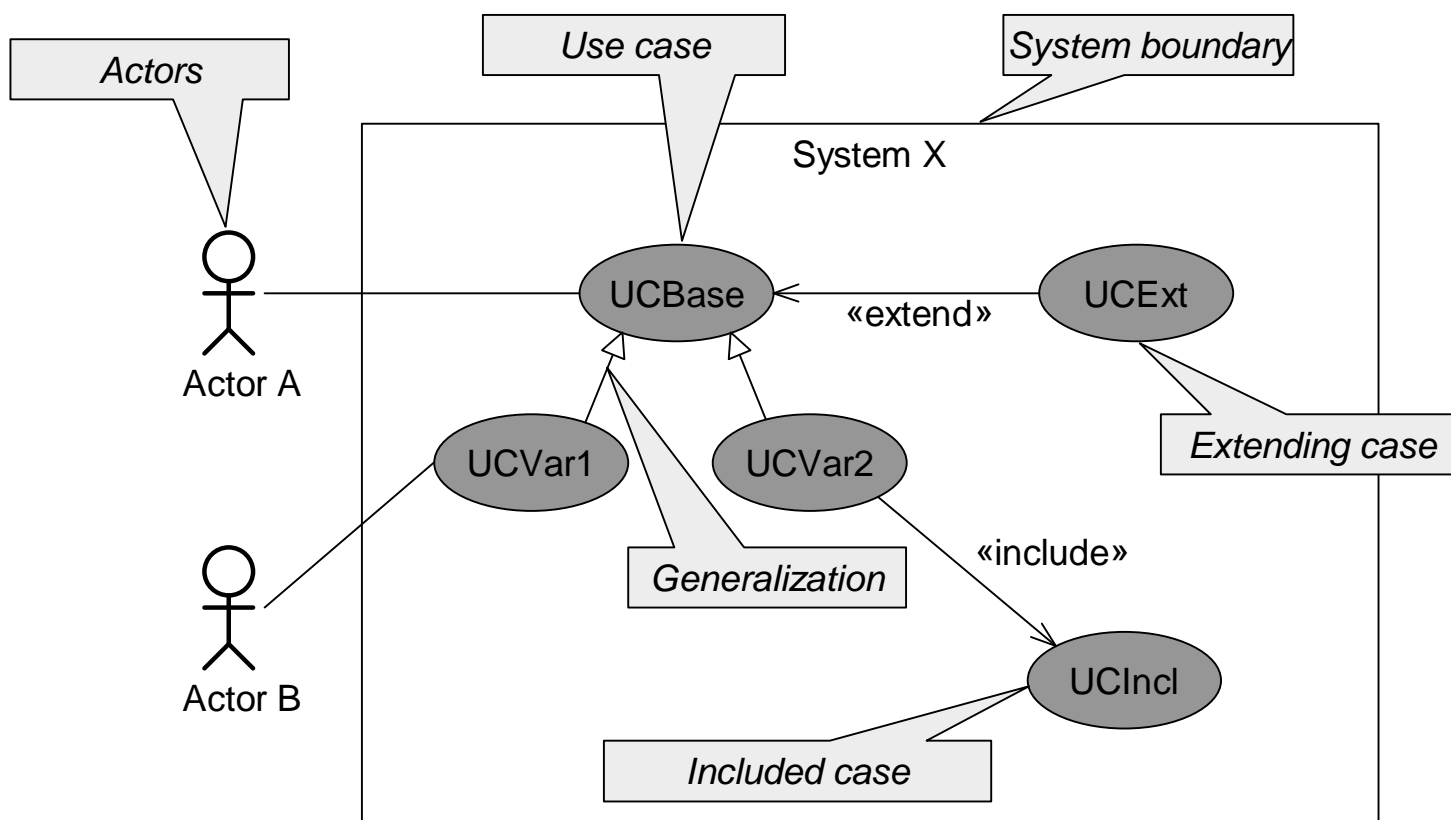
Notation:



Use Case Diagrams

3.1.15

Summary of Concepts



Use Case Diagrams

3.1.16

Algorithm: Use Case Analysis with UML

Input: requirements specification

Output: use case model, actors catalog, use case catalog, glossary

X:= set of use cases and actors mentioned in the requirements specification
begin with empty use case model, actor catalog, ...

WHILE X != \emptyset **DO**

 choose a use case or actor **x** of **X**

 together with the customer **DO**

 describe use case

 identify the participating actors and enter them into the actor catalog

 create the **use case diagrams** and enter them into the **UC catalog**

if necessary

 develop prototypical user interface as discussion foundation (if necessary)

 actuate glossary

 create conceptual class, activity and interaction diagrams

 reconstruct the use case model (include, extend, generalize)

X:= **X** - {**x**} + new detected use cases and actors

END WHILE

Use Case Diagrams

3.1.17

Tip: Interaction with External Systems

There are 4 approaches to the interaction with external systems:

1. Show **each** interaction with remote systems in the diagram.
2. Only show use cases for external interaction, if the **other** system **initiates** the contact.
3. Only show system actors, if they need the already identified use cases.
4. Don't treat systems as actors, but use the user's requests.

Use **external events** to identify use cases not covered by actors. Find every possible event of the „world outside“ needing a reaction.

Use Case Diagrams

3.1.18