



## UNIFIED MODELING LANGUAGE



Aitecon

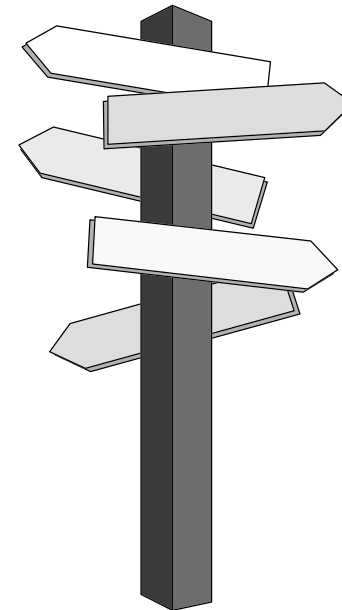
# Use Case Modeling

---

*Radovan Červenka, October 1998 (version 0.04)*

# Context

- ✓ Introduction
- ✓ Generic Mechanisms
- ➔ **■ Use Case Modeling**
  - Static Structure Modeling
  - Dynamic Behavior Modeling
  - Interaction Modeling
  - Physical Structure Modeling
  - General Extension Mechanisms



# Use Case Model

→ **functionality of the system and its surroundings**

## Consists of:

- Use Case Diagrams
- Use Case and Actor Descriptions

## Supported by:

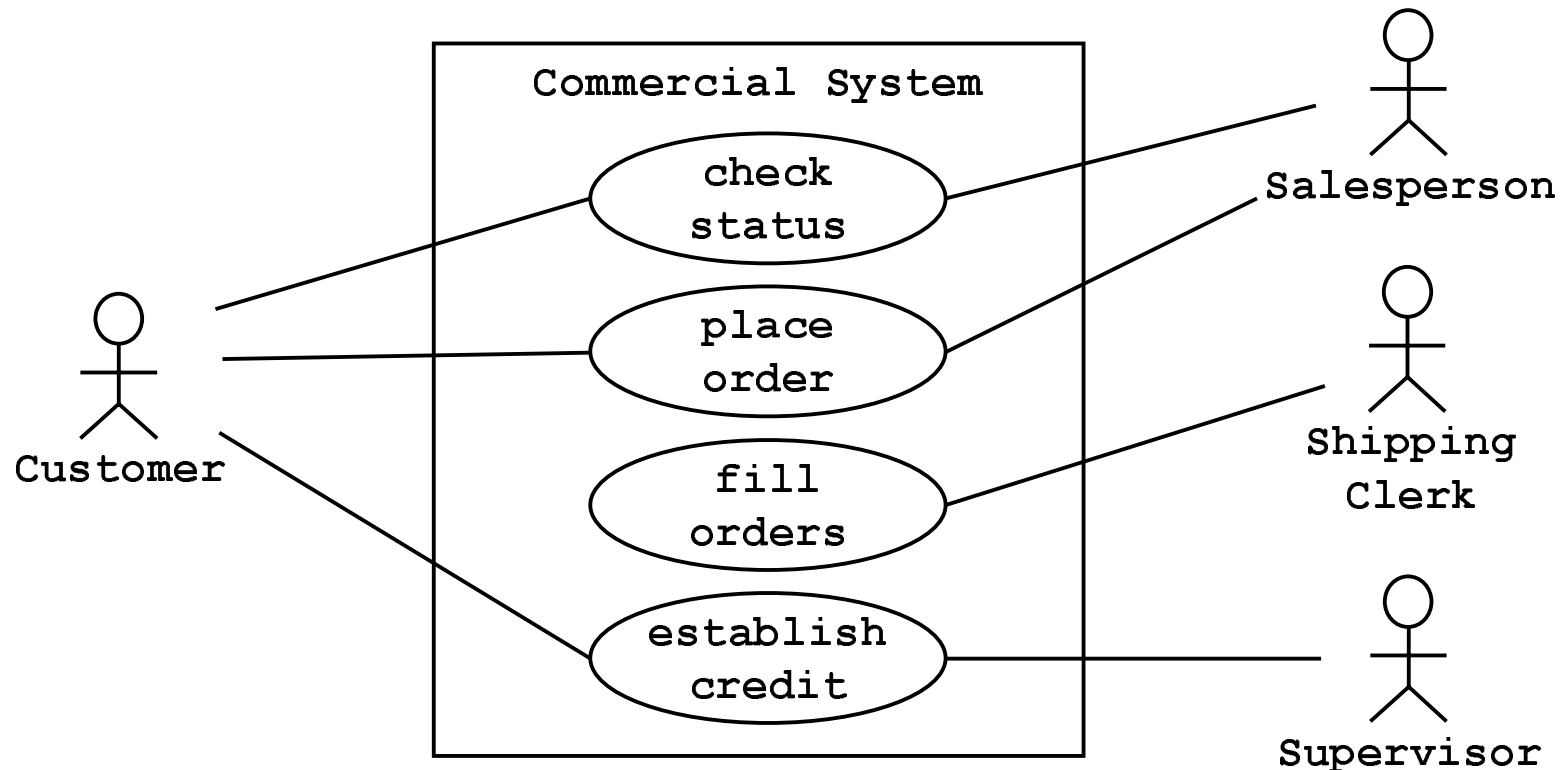
- UI Descriptions
- Non-functional Requirements Descriptions

## Used (mainly) in:

- Requirements Capture ⇒ functional system requirements
- Analysis ⇒ building analysis and design models, models structuring
- Testing ⇒ test cases
- Management ⇒ planning of analysis/design, implementation and testing

# Use Case Diagram

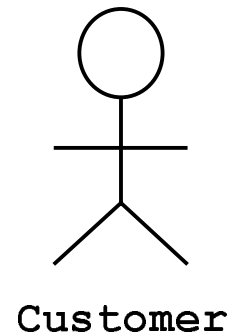
- **Actors (external entities), Use Cases (functions) of the System and their structure**
- **defines outer behaviour/functionality of the system without its inner structure**



# Actor

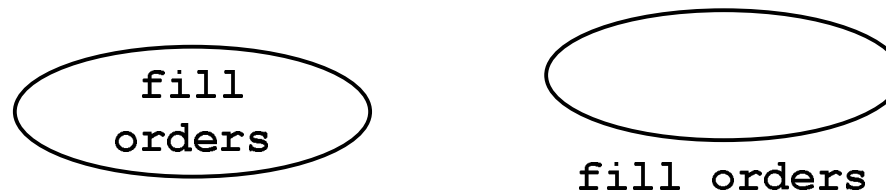
→ a *role* of outer object(s) interacting directly with the system

- external to the system
- can represent the type of a user or the other system
- one physical object may play several roles



# Use Case

- description of a coherent sequence of transactions in dialog between an actor and the system
- the way how an actor uses the system
- inside the system
- *primary & secondary actors* of a use case
  - a primary actor initiates a use case
  - secondary actors (usually machines) are called by the system to complete a use case
- represents *WHAT* the system must provide, not *HOW*



# Scenario

→ a session that an actor instance has with the system

- has details of real data and actual expected output
- potentially hundreds to thousands in an application

**1**

John enters his account# 404504  
John enters his pin# 9342  
John requests his average balance from 1/1/97 - 7/31/97  
System gives the average balance

**2**

Larry enters his account# 4343443  
Larry enters his pin# 84954  
Larry requests his average balance from 1/1/95 - 12/31/96  
System gives the average balance

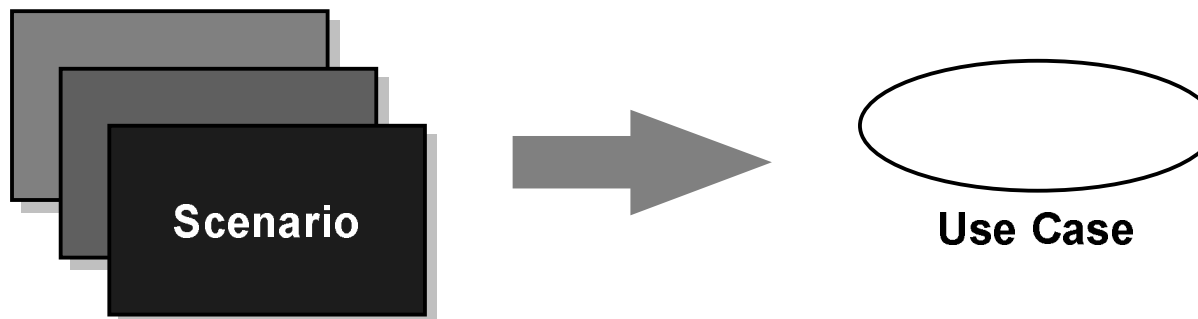
**3**

Mary enters her account# 34334  
Mary enters her pin# 4343  
System detects non valid pin and repeats the procedure

# Use Case vs. Scenario

## ! Use Cases are not Scenarios

- use case represents a set of potential scenarios
- looking at a family of similar scenarios, you can gather the essence of what is typically done
- similar scenarios will follow similar patterns of work and provide similar types of results
- normally each use case focuses on a specific *goal*
  - \* e.g. to obtain the current account balance



# Use Case Description

## CHARACTERISTIC

**Name:** use case name

**Goal:** a longer statement of the goal

**Scope:** subsystem, app., ...

**Preconditions:** state before use case execution

**Post Conditions:** state after use case execution

**Trigger:** action upon which is use case started

**Primary Actor:** a role name

**Secondary Actors:** list of other needed roles or systems

## ALGORITHM

(primary scenario, extensions and variations)

steps of the algorithm:

*<step#> <action description>*

control expressions:

if then else, repeat, switch, ...

## RELATED INFORMATION

**Priority:** how critical

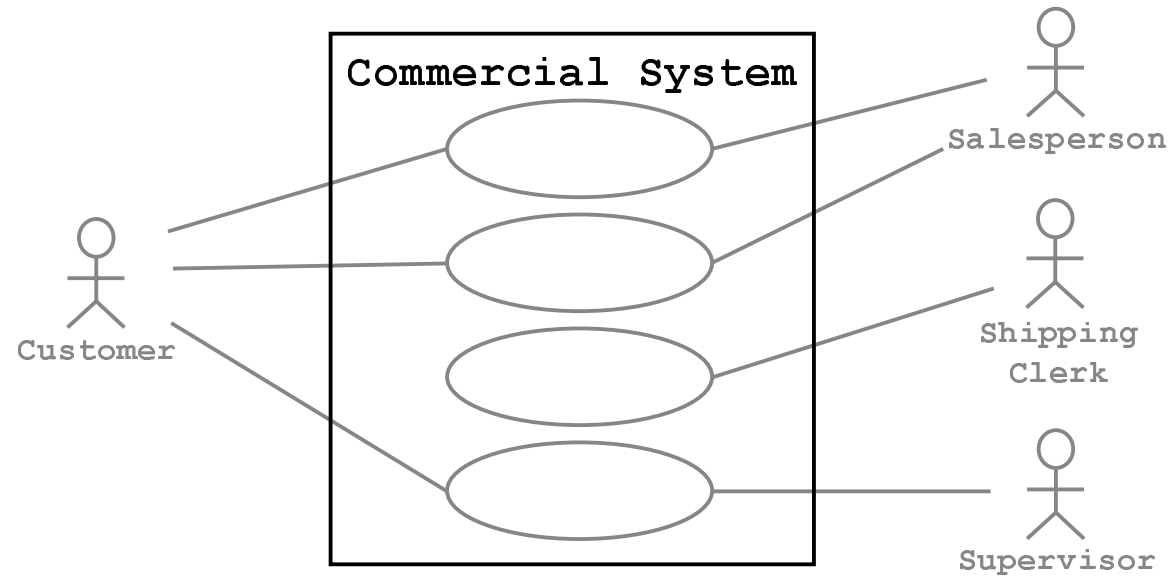
**Time:** a performance duration

**Frequency:** how often

**Other Non-functional Requirements:** ...

# System

- the functional boundary of the system
- is described by a finite set of use cases
  - actors are drawn outside of the system, use cases inside

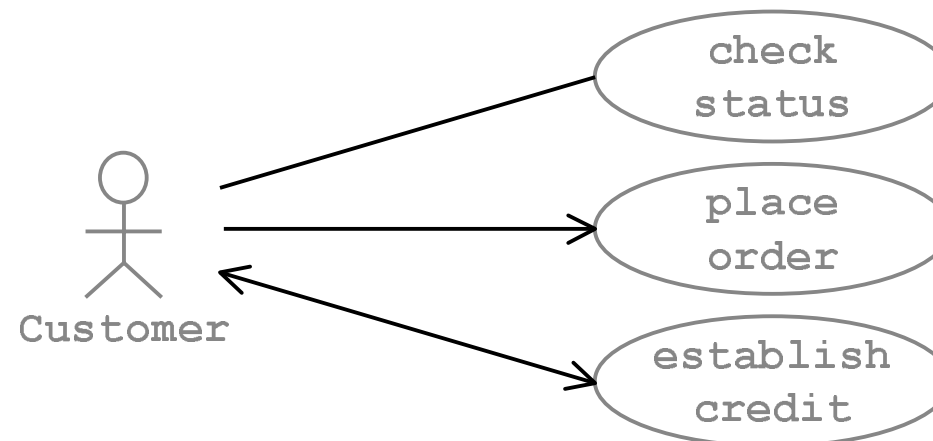


# Use Case Relationships <sup>1</sup>

## Communicates

→ the participation of an actor in a use case

- only between an actor and a use case
- notation: an association with stereotype «communicates»
  - \* the stereotype is optional
  - \* if necessary, the communication direction can be expressed by navigation arrows



# Use Case Relationships 2

## Extends

- indicates the possibility of inclusion of the behaviour of *extending use case* into an instance of the *base use case*
- the base use case is unaware of the extension
  - extension often represents unusual behaviour, such as an exception or error handling
  - notation: a generalization with stereotype «extends»
    - base use case's behaviour plus more



# Extension Example

## Create New Employee - base use case

- Enter Employee Information
- Enter Salary Information
- Enter Title
- Save Entry
- System validates
- If no problems, systems setups new employee record

## Create Body Guard

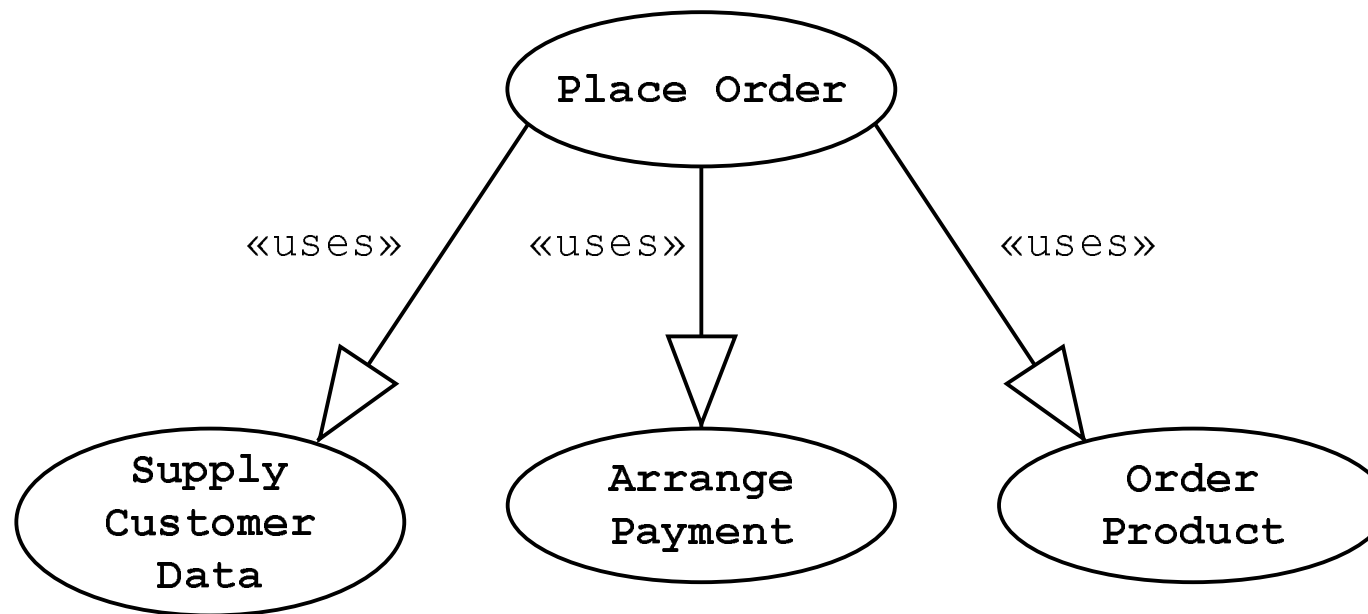
- System checks employee police record



# Use Case Relationships <sup>3</sup>

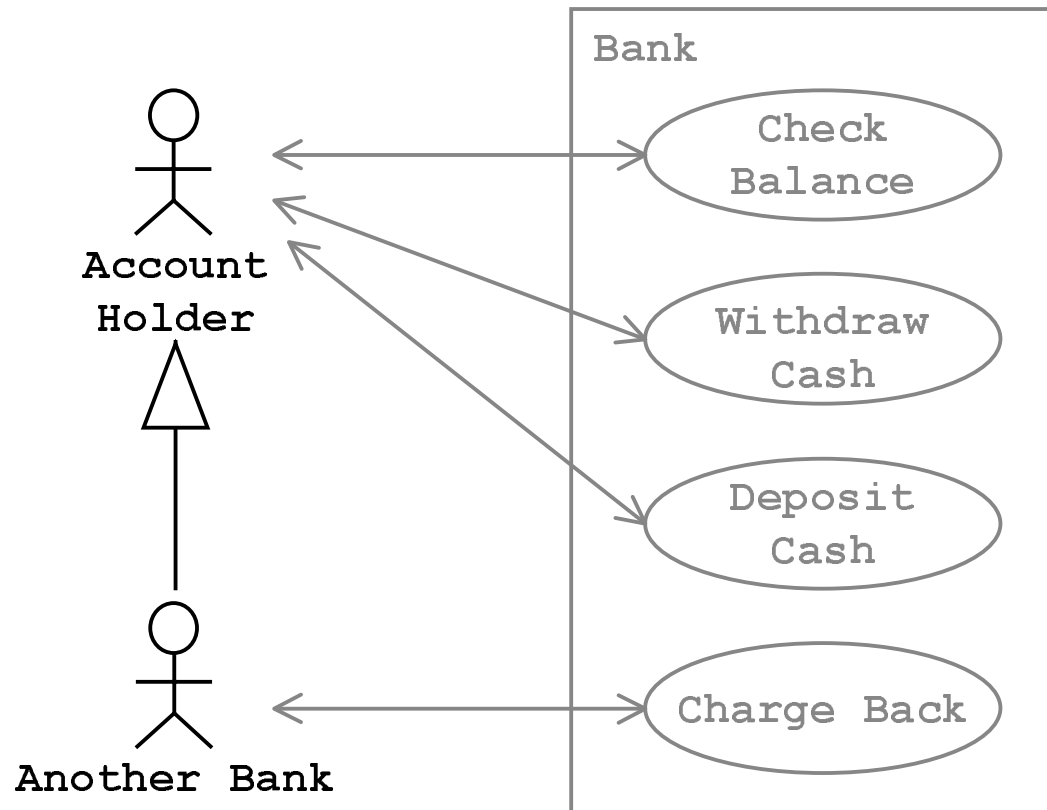
## Uses

- indicates that an instance of the use case will also include the behaviour specified by another use case
- being used use case obviously implements a behaviour shared by a number of use cases (“subroutine”)
  - notation: a generalization with stereotype «uses»



# Actor Inheritance

- the specialized actor gets all capabilities of the parent actor
- simplifies the diagram, without losing detail
- \* *abstract actors* can be also introduced

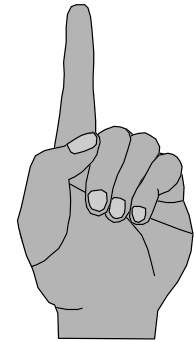


# Process of Use Case Modeling

- **Capture a common vocabulary (*Glossary*)**
- **Find actors**
- **For each actor determine a set of “its” use cases**
- **Briefly describe use cases and actors**
- **Package use cases and actors**
- **Present the use case model in use case diagrams**
- **Structure the use case model**
- **Prioritize use cases**
- **Describe use cases**
- **Review the use case model**

# Rules for Use Case Diagram

- Differentiate primary and secondary actors
- UC must provide a real service to the user
- Keep drawings clear and neat
- Do not put too many use cases in one diagram
  - simpler diagrams are easier to understand
- Split up the UC model into Use Case Packages
  - packages of related use cases
  - *functional decomposition* of the system



# Summary

- **Use Case Diagram**
- **Actor**
- **Use Case**
- **Scenario**
- **Use Case Description**
- **System**
- **Use Case Relationships**
- **Actor Inheritance**
- **Process of Use Case Modeling**
- **Rules for Use Case Diagram**