
# UML and the development process

ABIS Training & Consulting
www.abis.be
training@abis.be

ABIS 2005

Document number: 1140_03a.fm
24 February 2005

Address comments concerning the contents of this publication to:
ABIS Training & Consulting, P.O. Box 220, B-3000 Leuven, Belgium
Tel.: (+32)-16-245610, Fax: (+32)-16-245691

# TABLE OF CONTENTS

**DEVELOPMENT PROCESSES & MODELING** **5**

*1 The (OO) development process* *6*

*2 The Unified Modeling Language (UML)* *8*

*3 The (R)UP* *10*

*4 Agile Modeling* *11*

*5 Goals of UML* *12*

*6 UML Diagrams Overview* *13*

*7 UML usage modes* *16*

**THE UML DIAGRAMS** **19**

*1 Use Case Diagram* *21*

*2 Class Diagram* *24*

*3 Sequence Diagram* *29*

*4 State(chart) diagram* *33*

*5 Activity diagram* *35*

*6 Package Diagram* *38*

*7 Component Diagram* *40*

*8 Deployment Diagram* *41*

# Development processes & Modeling

**Objectives :**

- **How do processes realize flexibility?**
- **Positioning UML**

# The (OO) development process



## Stages/activities:

- Requirements gathering
- Analysis
  - real world model
  - what, not how (no implementation details)
- Technical Design:
  - (software) objects & components
  - overall architecture (system)
- Implementation
- Deployment
- ...

**Development processes & Modeling**

1. The (OO) development process
2. The Unified Modeling Language (UML)
3. The (R)UP
4. Agile Modeling
5. Goals of UML
6. UML Diagrams Overview
7. UML usage modes

# Incremental and iterative development

- **Waterfall:**
  - **no iteration**
  - **milestones**
  - **easy to plan**
- **Incremental:**
  - **parts of system completed at**
    - different times
    - different rates
  - **no revision**
  - **relatively easy to plan**
- **Iterative:**
  - **reworking parts of system**
  - **aim: improve quality**
  - **difficult to plan**

# The Unified Modeling Language (UML)



- **UML (1997):**
  **common set of modelling constructs and notations**
- **Originally based on methods by Rumbaugh, Booch and Jacobson**



- **UML is not a standardized development process!**
- **Current version V1.5 (2003) >> V2.0 (2004)**

UNIFIED MODELING LANGUAGE

OBJECT MANAGEMENT GROUP

**Development processes & Modeling**

1. The (OO) development process
2. The Unified Modeling Language (UML)
3. The (R)UP
4. Agile Modeling
5. Goals of UML
6. UML Diagrams Overview
7. UML usage modes

# The Unified Software Development Process

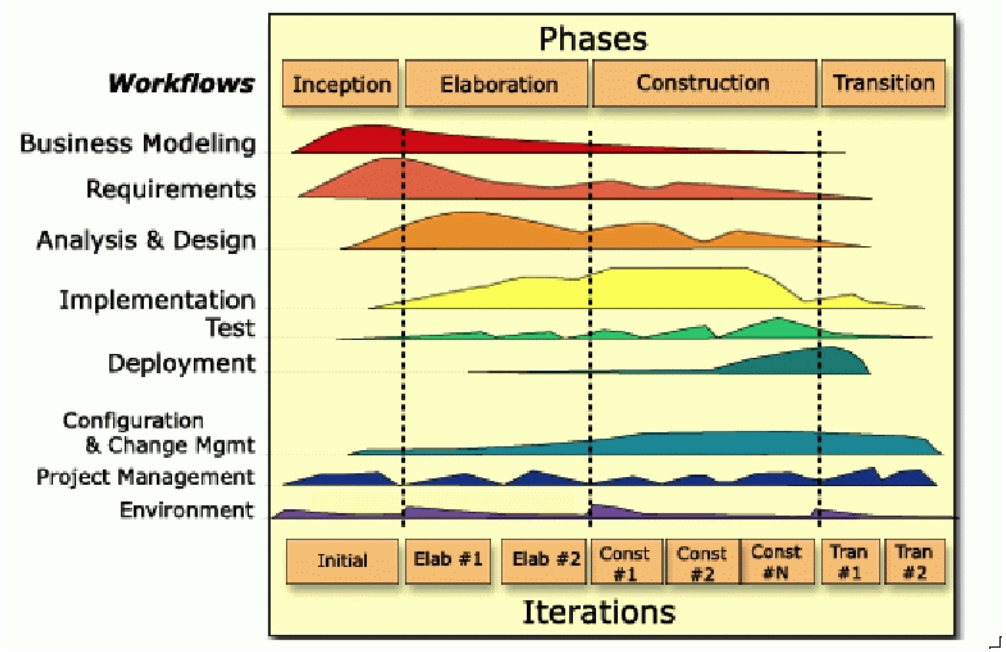
- By the same 3 Amigos (1999)
- principles:
  - use-case driven
  - architecture centric
  - iterative
  - incremental

# The (R)UP



## By the 3 amigos > Rational

## Iterating many times over several workflows (disciplines)




## Artifacts / Deliverables

**Development processes & Modeling**

1. The (OO) development process
2. The Unified Modeling Language (UML)
3. The (R)UP
4. Agile Modeling
5. Goals of UML
6. UML Diagrams Overview
7. UML usage modes

# Agile Modeling



## A more lightweight approach

## Popular examples:

- **Extreme Programming**
- **DSDM**
- **Scrum**

# Goals of UML



- Provide expressive modelling language (visual + ...)
- Be independent of
  - programming language
  - process
- Flexible and extensible
  - OCL (Object Constraint Language)
  - UML Profiles
- Encourage OO tools market >> exchangeability:
  - XMI
  - Diagram Interchange
- Support higher-level reuse concepts
- Model Driven Architecture (MDA)

**Development processes & Modeling**

1. The (OO) development process
2. The Unified Modeling Language (UML)
3. The (R)UP
4. Agile Modeling
5. Goals of UML
6. UML Diagrams Overview
7. UML usage modes

# UML Diagrams Overview



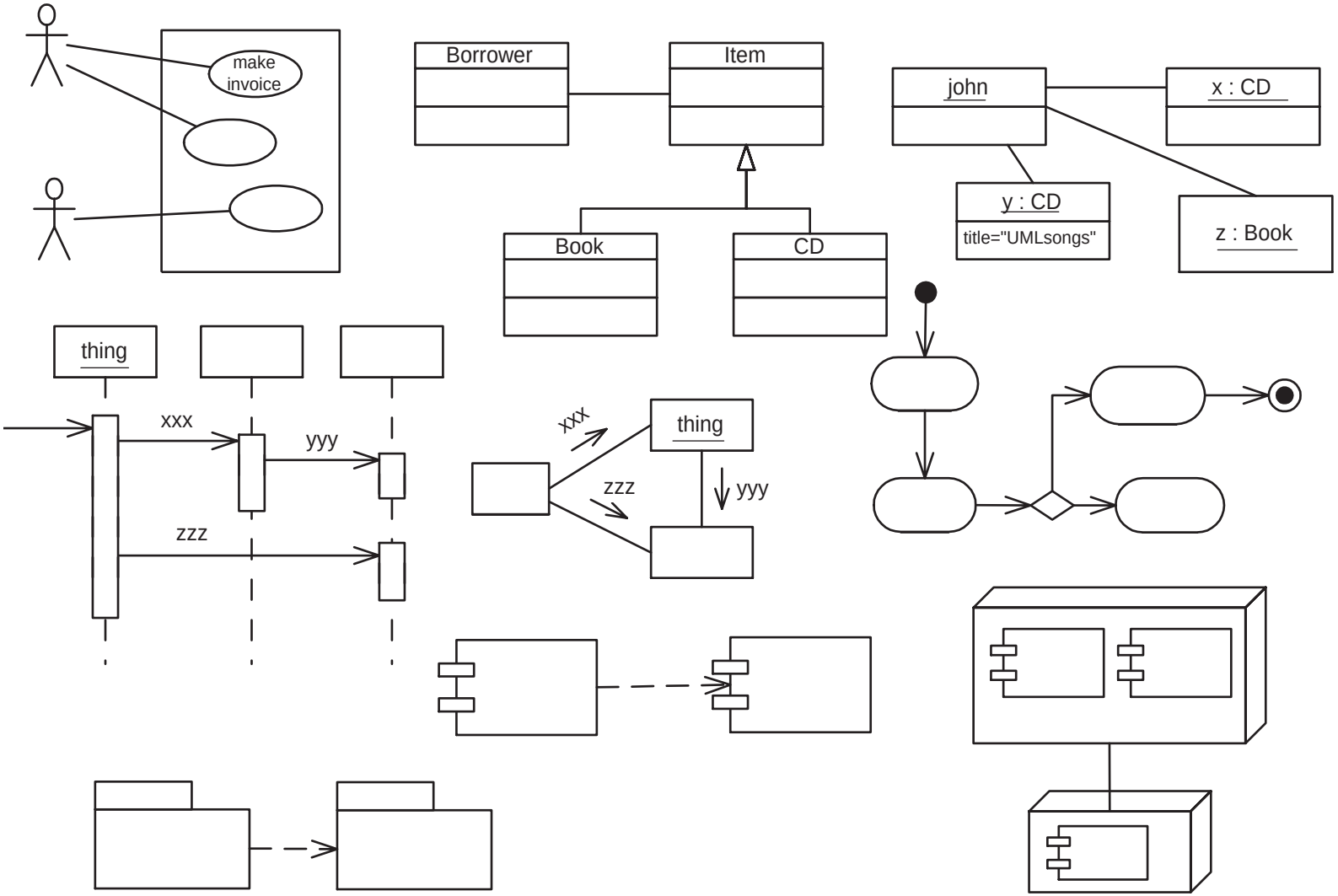
**Can be used at different stages of any process.**

- **Requirements:**
  - **use case diagram**
- **Static structure:**
  - **class diagram**
  - **object diagram**
- **Dynamic behaviour:**
  - **sequence & collaboration/communication diagrams**
  - **statechart diagram**
  - **activity diagram**
- **Architectural:**
  - **package diagram**
  - **component diagram**
  - **deployment diagram**

**Development processes & Modeling**

1. The (OO) development process
2. The Unified Modeling Language (UML)
3. The (R)UP
4. Agile Modeling
5. Goals of UML
6. UML Diagrams Overview
7. UML usage modes

# UML Diagrams (..)




**Development processes & Modeling**

1. The (OO) development process
2. The Unified Modeling Language (UML)
3. The (R)UP
4. Agile Modeling
5. Goals of UML
6. UML Diagrams Overview
7. UML usage modes

# ‘CASE’ Tools

- covering the development cycle from the top
- central repository
- sometimes including the implementation (coding / code generation)
- reverse engineering / round-trip engineering
- e.g.
  - Rational Rose
  - Together
  - System Architect
- trend: better integration with IDE (e.g. Rational XDE)
- (limited upper CASE:) e.g. Visio, ArgoUML, ...

# UML usage modes



## (according to Martin Fowler )

## Three ways to use UML:

- **UML as a sketch**
- **UML as a blueprint (from design to coding)**
- **UML as a programming language (>> MDA)**

**Development processes & Modeling**

1. The (OO) development process
2. The Unified Modeling Language (UML)
3. The (R)UP
4. Agile Modeling
5. Goals of UML
6. UML Diagrams Overview
7. UML usage modes

# MDA

## MDA = Model Driven Architecture


## Promoted by OMG:

- 1 PIM (Platform Independent Model)
- leads to many PSMs (Platform Specific Model)

## UML (UML2!) plays key role

**Development processes & Modeling**

1. The (OO) development process
2. The Unified Modeling Language (UML)
3. The (R)UP
4. Agile Modeling
5. Goals of UML
6. UML Diagrams Overview
7. UML usage modes

# The UML Diagrams

**Objectives :**

- **Why so many?**
- **Where & when do I use which?**

# About nails and screwdrivers

## Most UML diagrams are useful for multiple jobs in the development process:

## e.g. class diagram:

- **analysis > conceptual modeling**
- **design**
- **implementation**

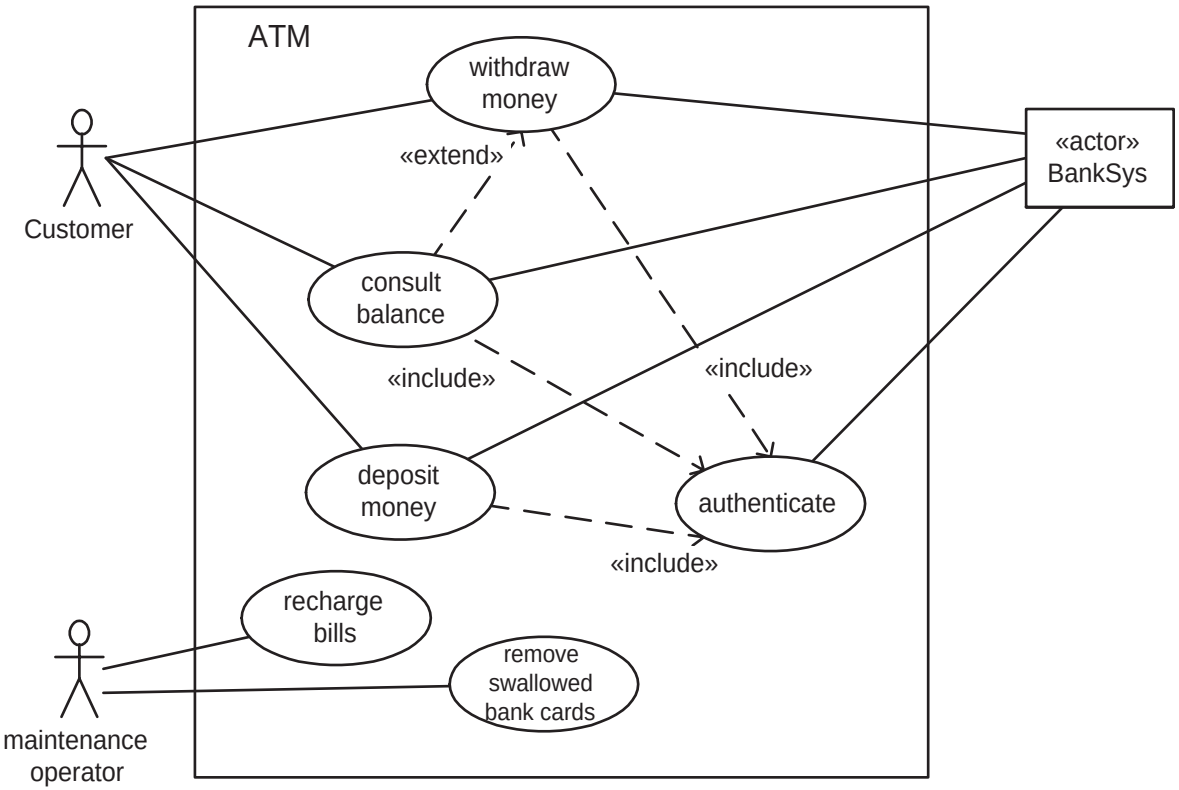
**The UML Diagrams**

1. Use Case Diagram
2. Class Diagram
3. Sequence Diagram
4. State(chart) diagram
5. Activity diagram
6. Package Diagram
7. Component Diagram
8. Deployment Diagram

# Use Case Diagram



## A technique for structuring functional requirements




**The UML Diagrams**

1. Use Case Diagram
2. Class Diagram
3. Sequence Diagram
4. State(chart) diagram
5. Activity diagram
6. Package Diagram
7. Component Diagram
8. Deployment Diagram

# Use Case Diagram (..)

- system: what are we building
  (> boundary!)
- actors (roles): who will use the system?
  (=people and machines)
  - primary actors
  - secondary/supporting actors
- use cases: what for?

A rather informal technique...

Use case details in

- text template
- (optional) activity diagrams
- (optional) system sequence diagram (SSD)

>> Use case driven process

# Use Case Template

| MAIN SUCCESS SCENARIO | Step | Action |
|---|---|---|
| | 1 | Customer feeds bank card into ATM. |
| | 2 | ATM identifies customer with BankSys. |
| | 3 | Customer enters PIN code. |
| | 4 | ATM gets customer and account details (balance, limits, etc) from BankSys. |
| | 5 | Customer specifies desired amount. |
| | 6 | ATM delivers the cash. |
| | 7 | Customer takes the cash. |
| | 8 | ATM delivers a printed ticket. |
| | 9 | ATM delivers the customer's bank card |
| | 10 | Customer takes the card and ticket. |
| **EXTENSIONS** | **Step** | **Branching Action** |
| | 2a | Invalid or unreadable bank card<br>2a1. ATM returns card; use case terminates. |
| | 4a | Invalid PIN code (1st-2nd attempt):<br>4a1. Use cases resumes at step 3 |
| | 4b | Invalid PIN code (3rd attempt):<br>4a. ATM swallows card; use case terminates. |
| | *a | Customer walked away without notice (time-out 30 seconds):<br>a1. ATM swallows card; use case terminates. |
| | etc... | etc... |

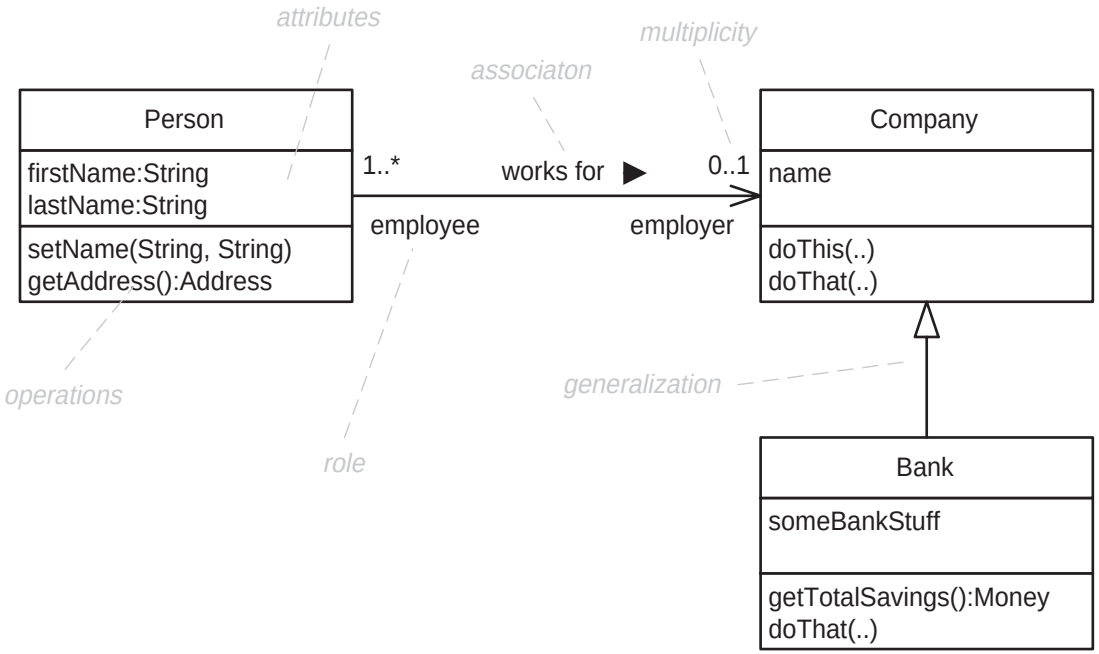
**The UML Diagrams**

1. Use Case Diagram
2. Class Diagram
3. Sequence Diagram
4. State(chart) diagram
5. Activity diagram
6. Package Diagram
7. Component Diagram
8. Deployment Diagram

# Class Diagram



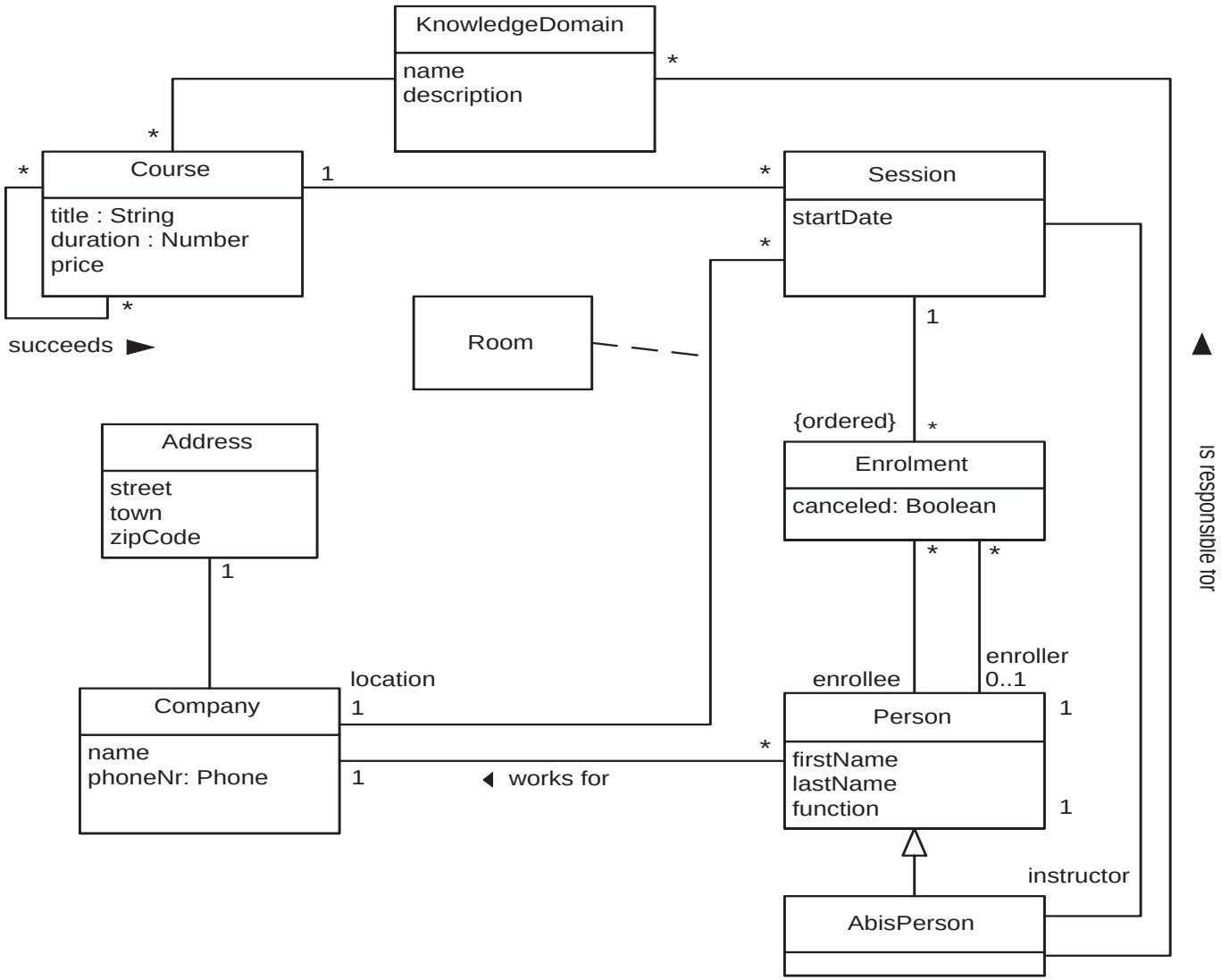
## The workhorse
## & most elaborated/matured/versatile of them all...




**The UML Diagrams**

1. Use Case Diagram
2. Class Diagram
3. Sequence Diagram
4. State(chart) diagram
5. Activity diagram
6. Package Diagram
7. Component Diagram
8. Deployment Diagram

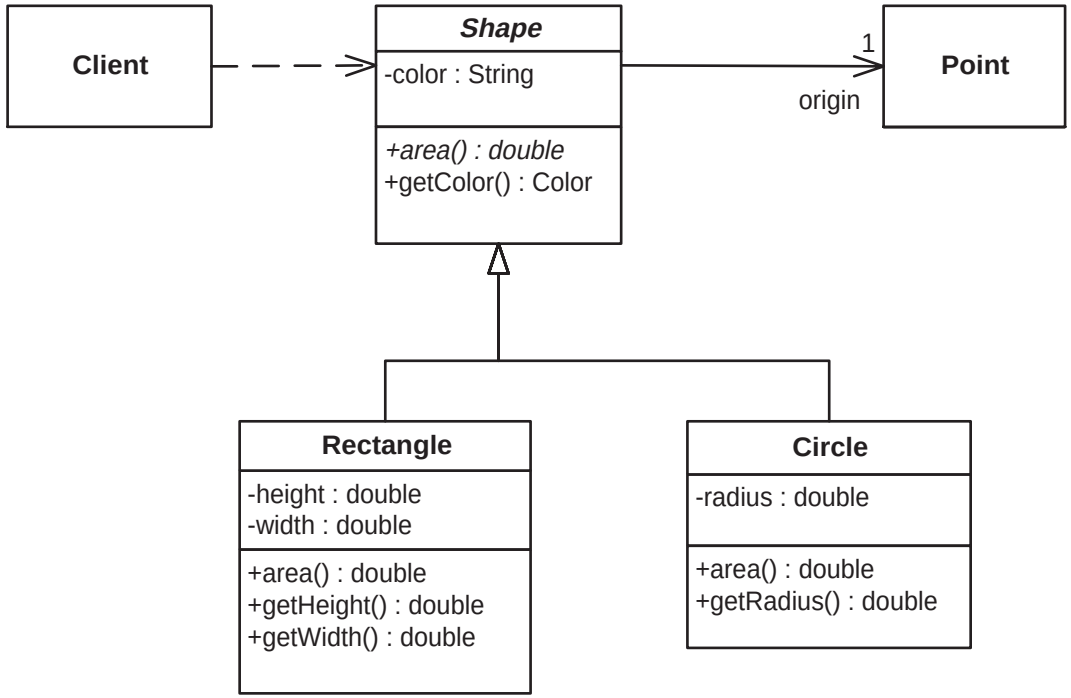
# Class Diagram: Domain Model




**The UML Diagrams**

1. Use Case Diagram
2. Class Diagram
3. Sequence Diagram
4. State(chart) diagram
5. Activity diagram
6. Package Diagram
7. Component Diagram
8. Deployment Diagram

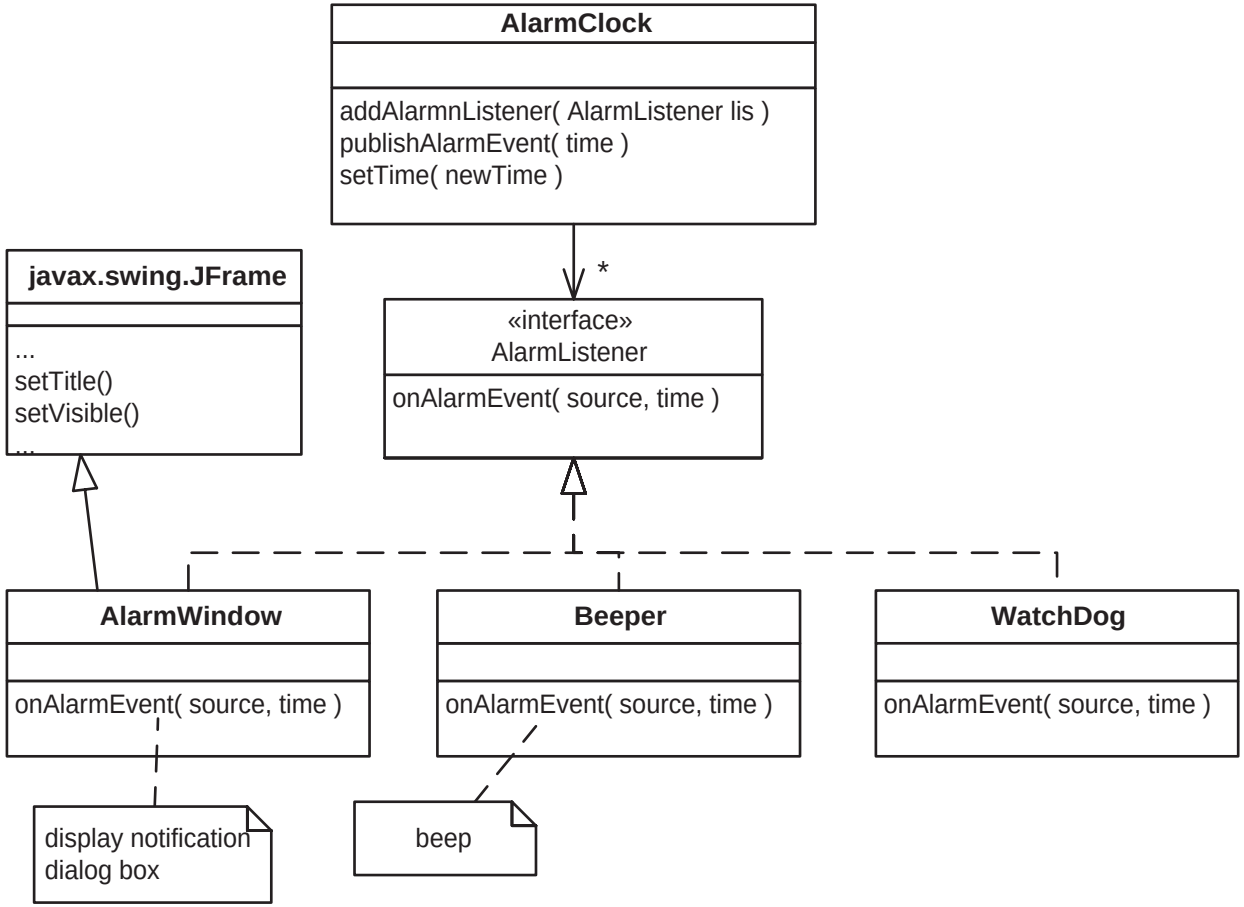
# Design Class Diagram (DCD)




**The UML Diagrams**

1. Use Case Diagram
2. Class Diagram
3. Sequence Diagram
4. State(chart) diagram
5. Activity diagram
6. Package Diagram
7. Component Diagram
8. Deployment Diagram

# DCD (..)


AlarmClock
addAlarmnListener( AlarmListener lis )
publishAlarmEvent( time )
setTime( newTime )
*
javax.swing.JFrame
...
setTitle()
setVisible()
...
«interface»
AlarmListener
onAlarmEvent( source, time )
AlarmWindow
onAlarmEvent( source, time )
Beeper
onAlarmEvent( source, time )
WatchDog
onAlarmEvent( source, time )
display notification
dialog box
beep

# Collaborating objects

**From analysis to design:**

- **design software objects (based on domain model + ...)**
- **realizing the ‘responsibilities’ as imposed by the use cases.**

**This is the real object stuff:**

- **beyond static structure**
- **show how objects interact**

**Interaction through messages:**

- **objects invoking operations on other objects**
- **getting returns**

**Two UML diagram types:**

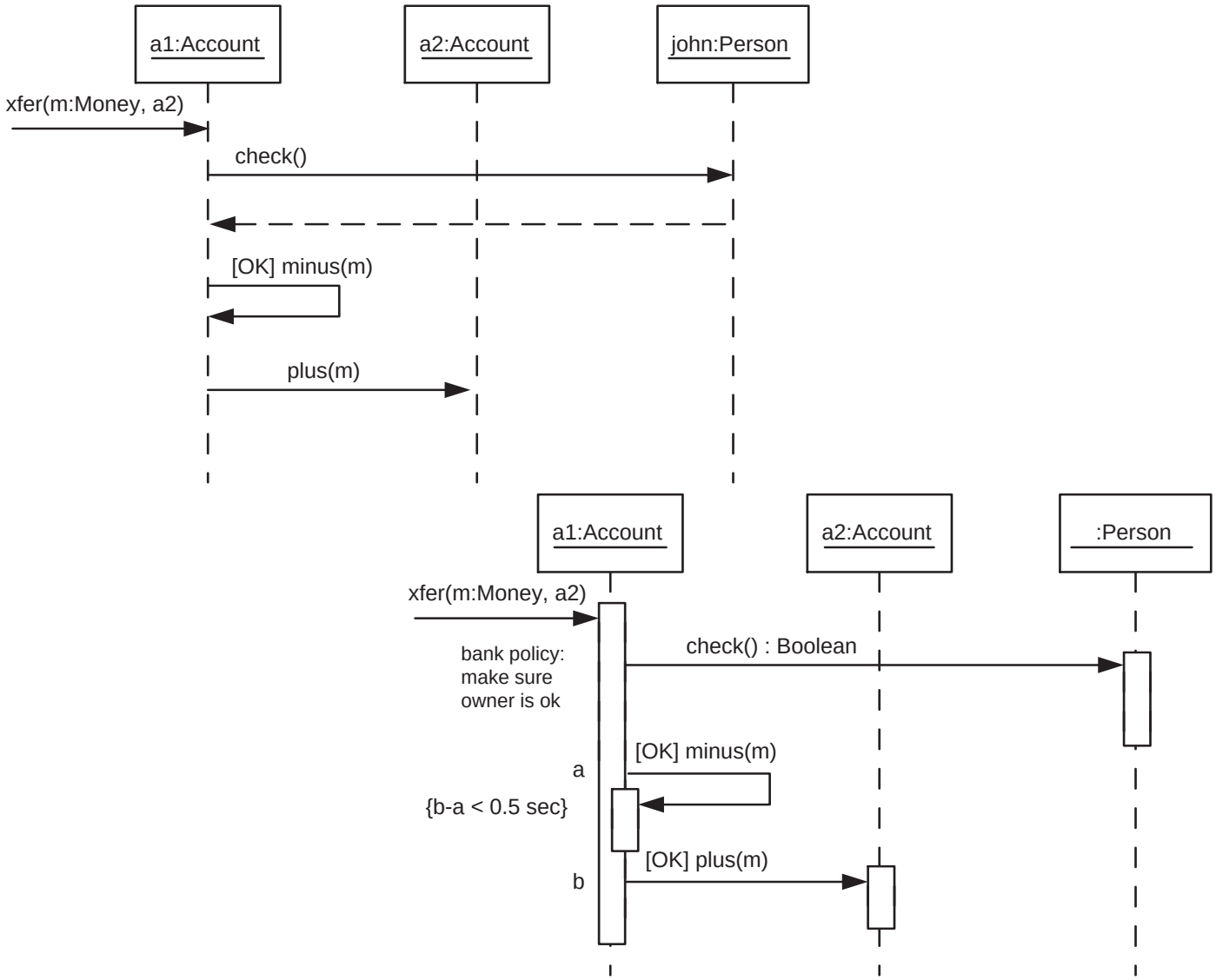
- **sequence diagram**
- **collaboration/communication diagram**

**The UML Diagrams**

1. Use Case Diagram
2. Class Diagram
3. Sequence Diagram
4. State(chart) diagram
5. Activity diagram
6. Package Diagram
7. Component Diagram
8. Deployment Diagram

# Sequence Diagram

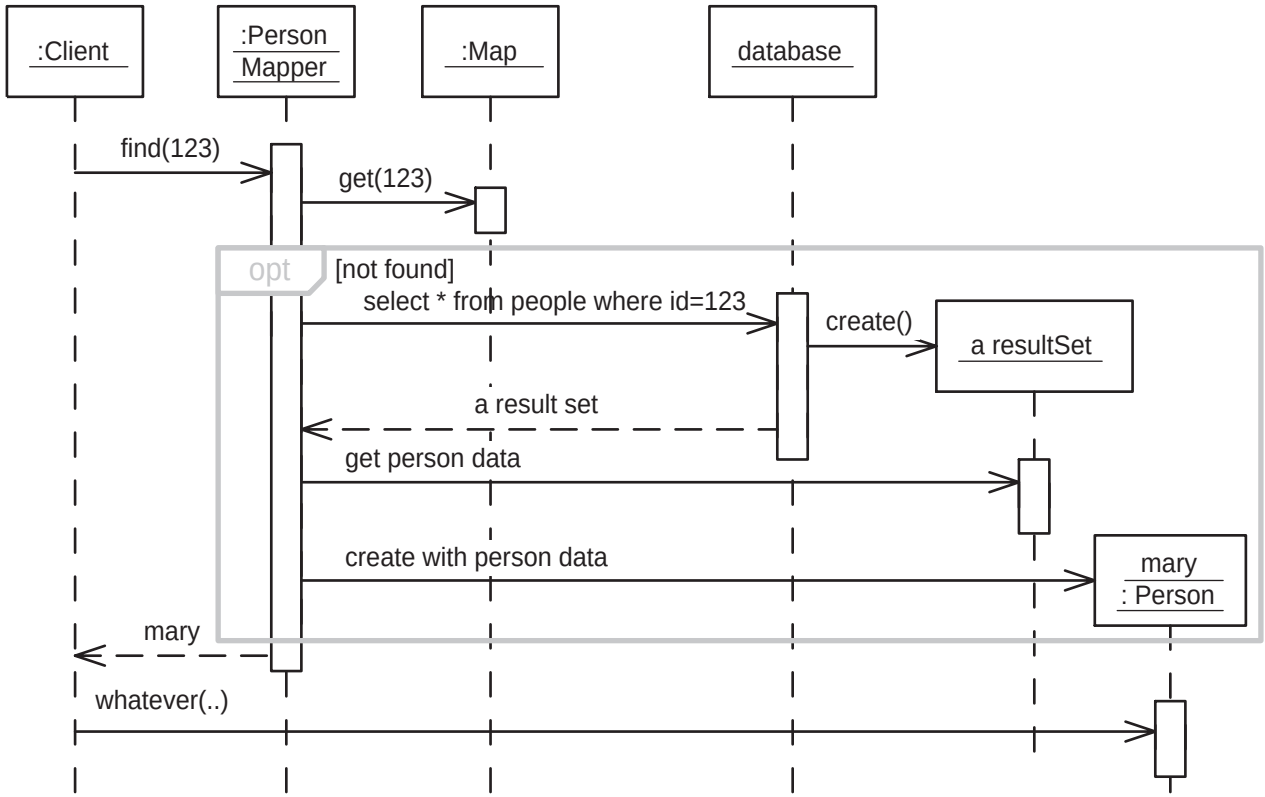





**The UML Diagrams**

1. Use Case Diagram
2. Class Diagram
3. Sequence Diagram
4. State(chart) diagram
5. Activity diagram
6. Package Diagram
7. Component Diagram
8. Deployment Diagram

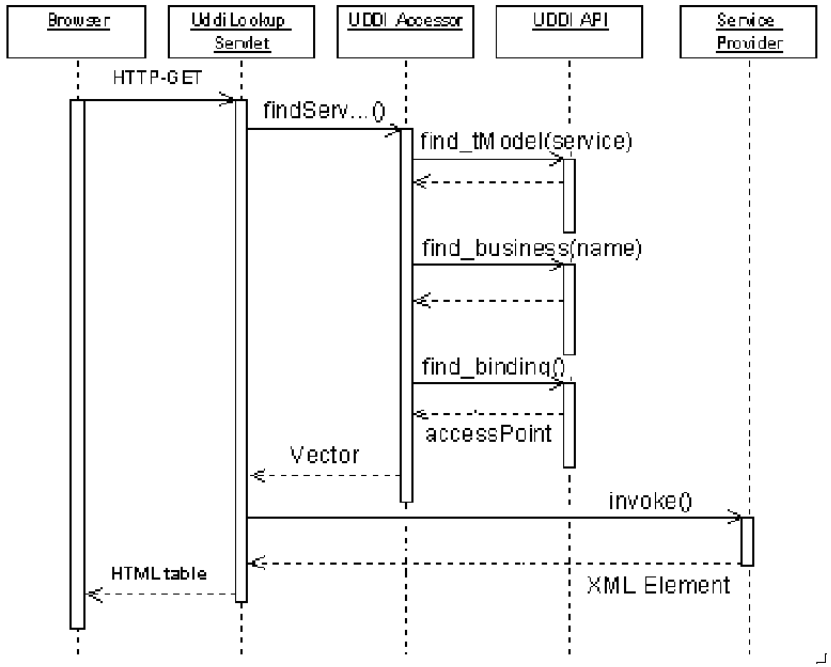
# Sequence Diagram (..)




**The UML Diagrams**

1. Use Case Diagram
2. Class Diagram
3. Sequence Diagram
4. State(chart) diagram
5. Activity diagram
6. Package Diagram
7. Component Diagram
8. Deployment Diagram

# Sequence Diagram (..)



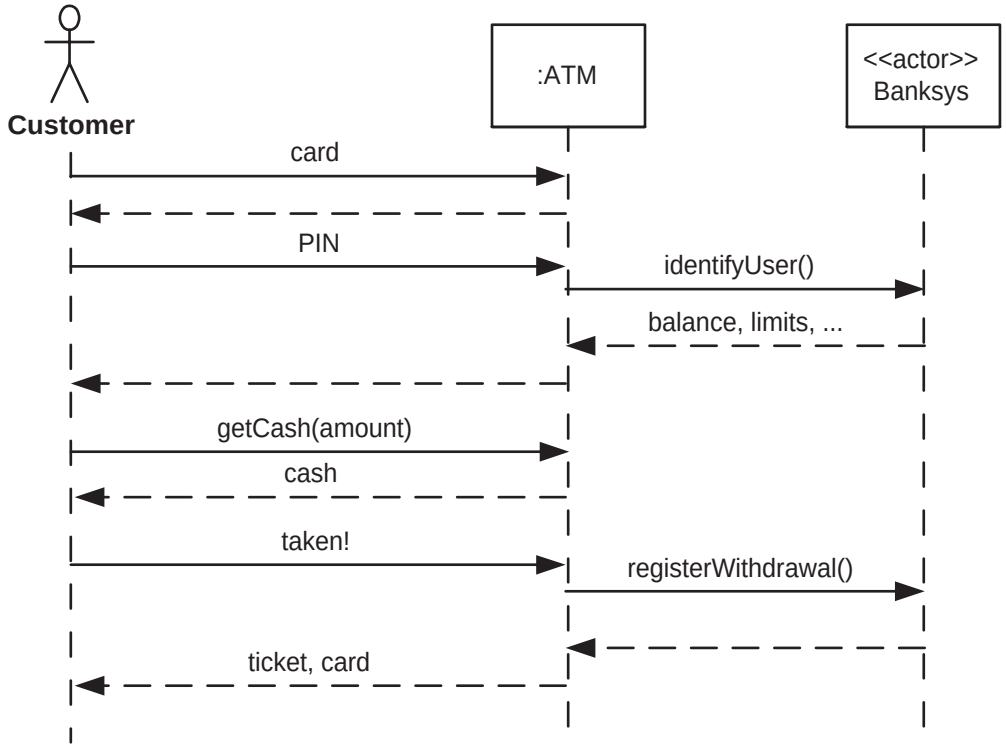

## All kind of participants:

- business objects, controllers, GUI components
- entire (sub)systems
- persons, organisations

**The UML Diagrams**

1. Use Case Diagram
2. Class Diagram
3. Sequence Diagram
4. State(chart) diagram
5. Activity diagram
6. Package Diagram
7. Component Diagram
8. Deployment Diagram

# SSD: System Sequence Diagram




- **optional, summarizes use case**
- **system events & system operations**
- **starting point for use case realization**

# State(chart) diagram



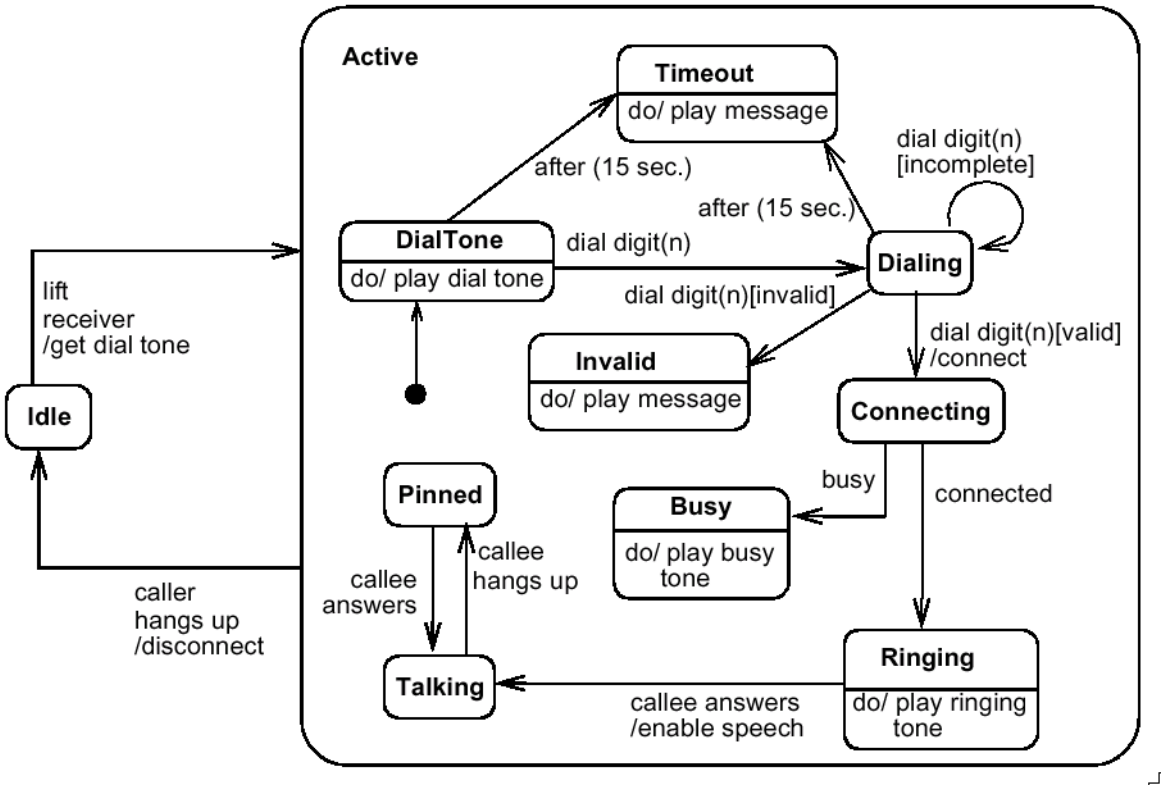
## Modeling the lifecycle of an object:

- state = condition of an object between events
  (a way of behaving, of reacting to events)
- event = something that happens
- state transition = change of state, caused by an event

**The UML Diagrams**

1. Use Case Diagram
2. Class Diagram
3. Sequence Diagram
4. State(chart) diagram
5. Activity diagram
6. Package Diagram
7. Component Diagram
8. Deployment Diagram

# Statechart diagram (..)




## Use ‘ad lib’ for modeling life cycle of ‘interesting’ objects e.g.

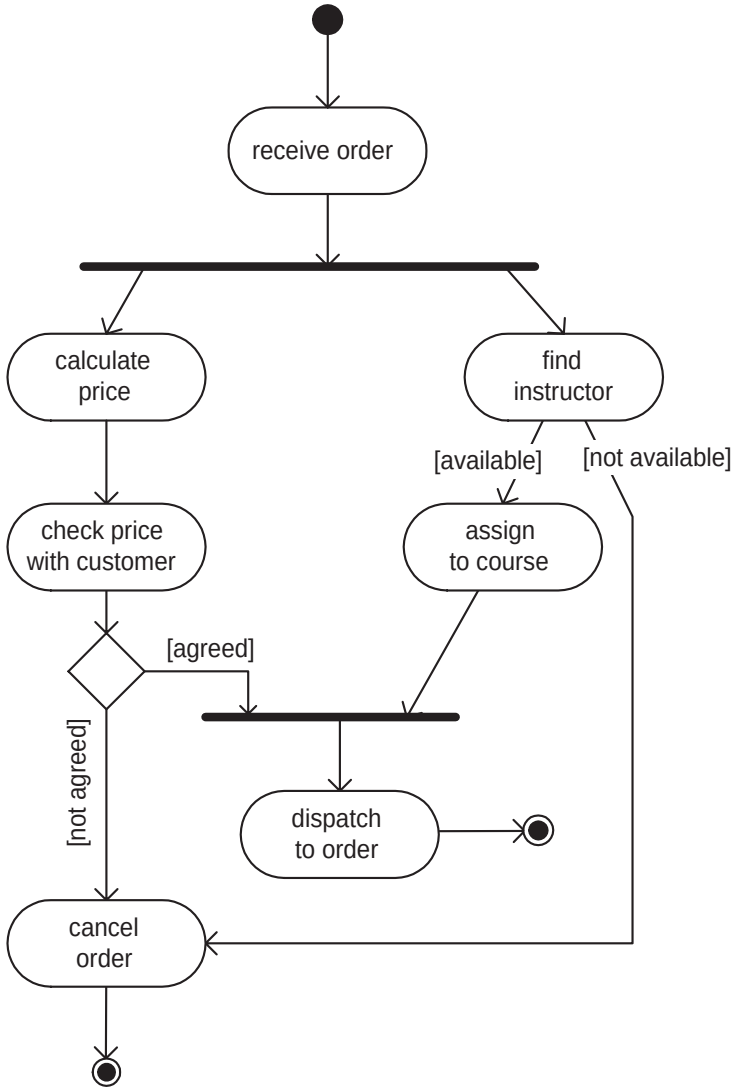
- devices (living vs. dead objects)
- use cases, (G)UI navigation, transactions

**The UML Diagrams**

1. Use Case Diagram
2. Class Diagram
3. Sequence Diagram
4. State(chart) diagram
5. Activity diagram
6. Package Diagram
7. Component Diagram
8. Deployment Diagram

# Activity diagram

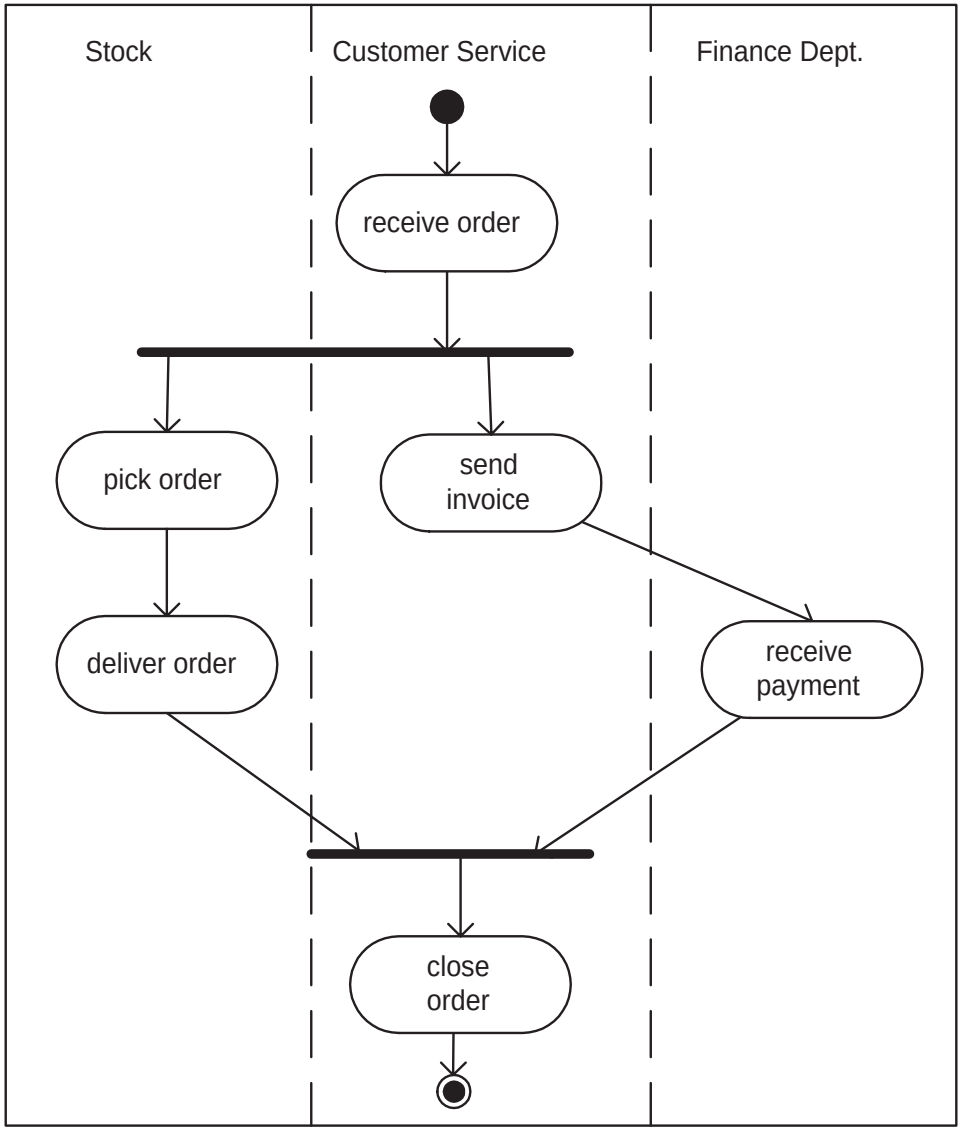





**The UML Diagrams**

1. Use Case Diagram
2. Class Diagram
3. Sequence Diagram
4. State(chart) diagram
5. Activity diagram
6. Package Diagram
7. Component Diagram
8. Deployment Diagram

# Activity diagram (..)



# Activity Diagram (..)

## Kind of degenerated statechart (at least in UML1...)

## Polyvalent use for flow of activities:

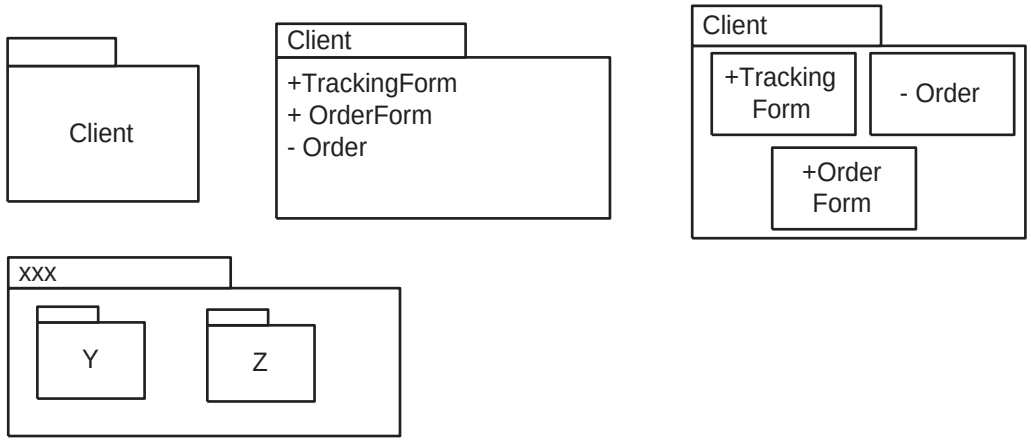
- business analysis & workflow
- use cases
- design level: operations, algorithms, ...

**The UML Diagrams**

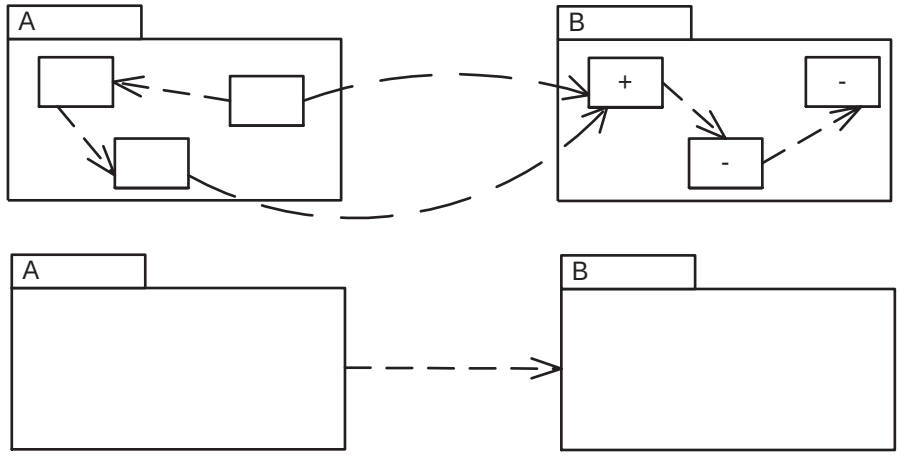
1. Use Case Diagram
2. Class Diagram
3. Sequence Diagram
4. State(chart) diagram
5. Activity diagram
6. Package Diagram
7. Component Diagram
8. Deployment Diagram

# Package Diagram





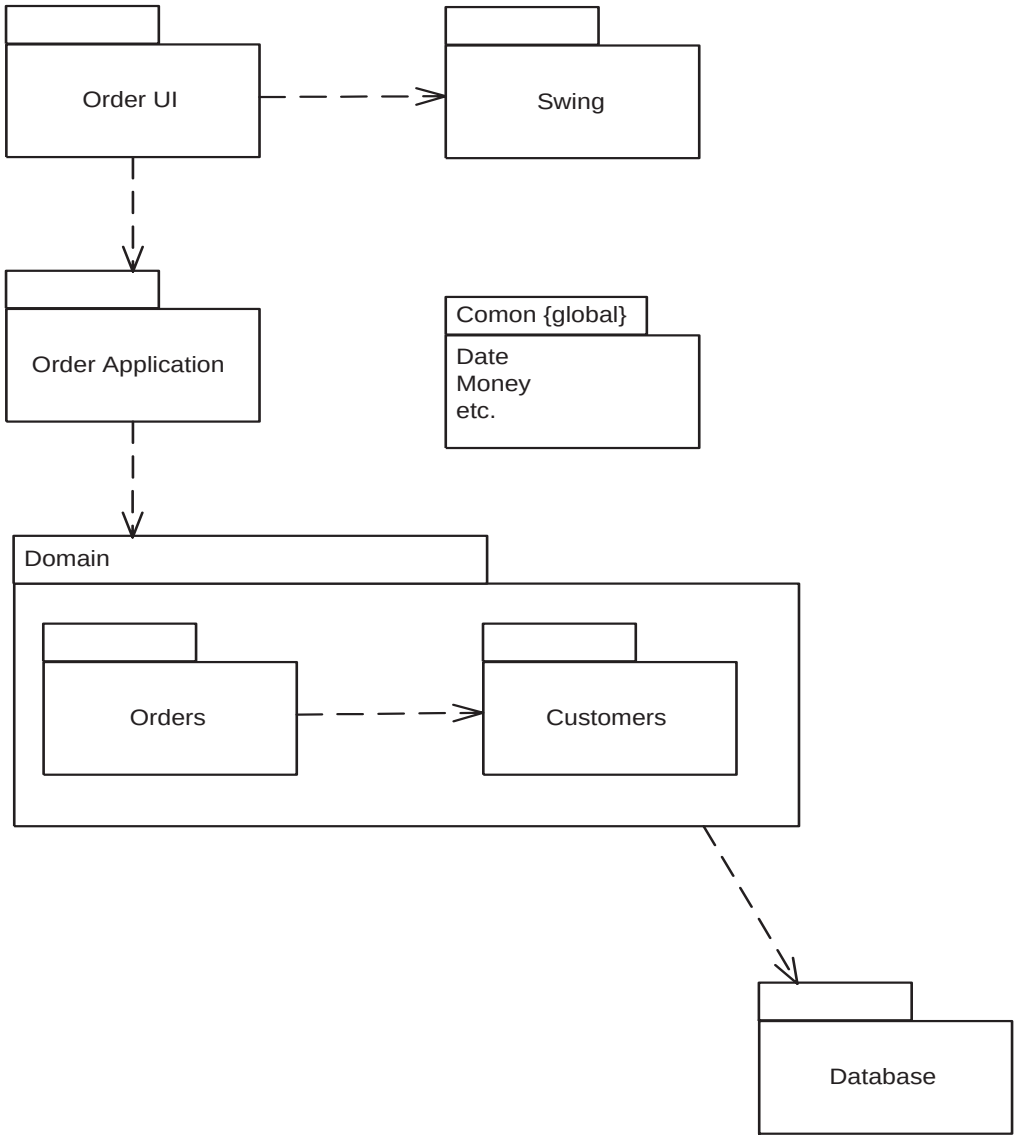

- **Grouping interrelated classes**
- **Controlling dependencies**




**The UML Diagrams**

1. Use Case Diagram
2. Class Diagram
3. Sequence Diagram
4. State(chart) diagram
5. Activity diagram
6. Package Diagram
7. Component Diagram
8. Deployment Diagram

# Package Diagram & System Architecture

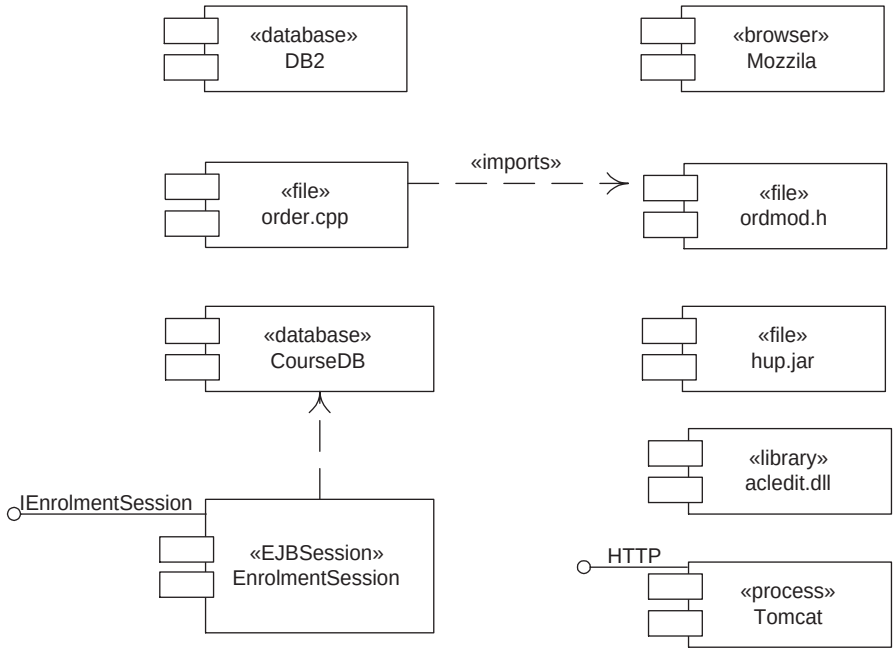



**The UML Diagrams**

1. Use Case Diagram
2. Class Diagram
3. Sequence Diagram
4. State(chart) diagram
5. Activity diagram
6. Package Diagram
7. Component Diagram
8. Deployment Diagram

# Component Diagram





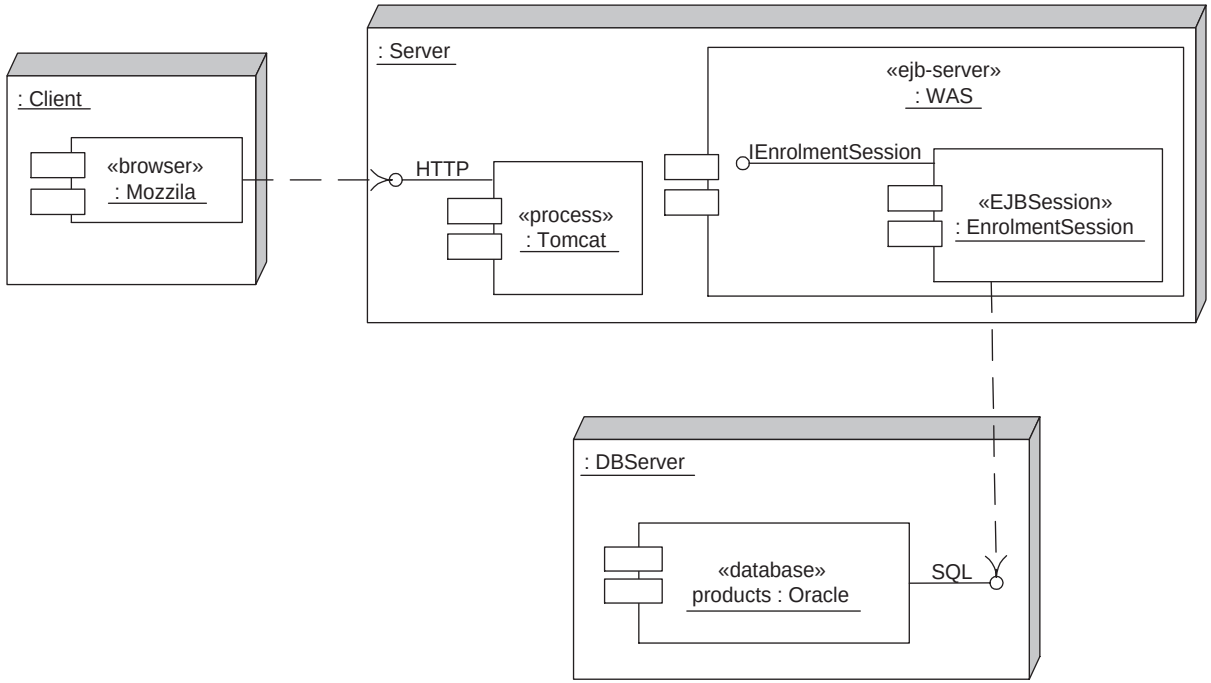

- **a modular, deployable, replaceable part of system (= physical!)**
- **encapsulates implementation**
- **exposes interfaces**

**The UML Diagrams**

1. Use Case Diagram
2. Class Diagram
3. Sequence Diagram
4. State(chart) diagram
5. Activity diagram
6. Package Diagram
7. Component Diagram
8. Deployment Diagram

# Deployment Diagram






## Shows where instances of components are running: on physical nodes (= some computer).

**The UML Diagrams**

1. Use Case Diagram
2. Class Diagram
3. Sequence Diagram
4. State(chart) diagram
5. Activity diagram
6. Package Diagram
7. Component Diagram
8. Deployment Diagram

# Want some more diagrams?

## Check UML2, e.g.:

- **Composite Structure**
- **Interaction Overview**
- **Timing**

**The UML Diagrams**

1. Use Case Diagram
2. Class Diagram
3. Sequence Diagram
4. State(chart) diagram
5. Activity diagram
6. Package Diagram
7. Component Diagram
8. Deployment Diagram

# Conclusion?

## UML is a well-filled toolbox,

## but YOU as a developer are still the craftsman (or the artist) ...

**The UML Diagrams**

1. Use Case Diagram
2. Class Diagram
3. Sequence Diagram
4. State(chart) diagram
5. Activity diagram
6. Package Diagram
7. Component Diagram
8. Deployment Diagram

**The UML Diagrams**

1. Use Case Diagram
2. Class Diagram
3. Sequence Diagram
4. State(chart) diagram
5. Activity diagram
6. Package Diagram
7. Component Diagram
8. Deployment Diagram