# Table of Contents

## Development Processes & Modeling

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

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
- ...

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!
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
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)
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
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, ...
Three ways to use UML:

- UML as a sketch
- UML as a blueprint (from design to coding)
- UML as a programming language (>> MDA)
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
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
A technique for structuring functional requirements

Use Case Diagram

- Customer
- ATM
- BankSys
- Maintenance operator

Use Case Diagram is a technique for structuring functional requirements.
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

<table>
<thead>
<tr>
<th>MAIN SUCCESS SCENARIO</th>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>Customer feeds bank card into ATM.</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>ATM identifies customer with BankSys.</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>Customer enters PIN code.</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>ATM gets customer and account details (balance, limits, etc) from BankSys.</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>Customer specifies desired amount.</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>ATM delivers the cash.</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>Customer takes the cash.</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>ATM delivers a printed ticket.</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>ATM delivers the customer’s bank card</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>Customer takes the card and ticket.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>EXTENSIONS</th>
<th>Step</th>
<th>Branching Action</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2a</td>
<td>Invalid or unreadable bank card</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2a1. ATM returns card; use case terminates.</td>
</tr>
<tr>
<td></td>
<td>4a</td>
<td>Invalid PIN code (1st-2nd attempt):</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4a1. Use cases resumes at step 3</td>
</tr>
<tr>
<td></td>
<td>4b</td>
<td>Invalid PIN code (3rd attempt):</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4a. ATM swallows card; use case terminates.</td>
</tr>
<tr>
<td></td>
<td>*a</td>
<td>Customer walked away without notice (time-out 30 seconds):</td>
</tr>
<tr>
<td></td>
<td></td>
<td>a1. ATM swallows card; use case terminates.</td>
</tr>
<tr>
<td></td>
<td>etc...</td>
<td>etc...</td>
</tr>
</tbody>
</table>

---

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

Person
- firstName: String
- lastName: String
- setName(String, String)
- getAddress(): Address

Company
- name
- works for
- employee
- employer
- doThis(..)
- doThat(..)

Bank
- someBankStuff
- getTotalSavings(): Money
- doThat(..)
**Class Diagram: Domain Model**

- **KnowledgeDomain**
  - name
  - description

- **Course**
  - title : String
  - duration : Number
  - price

- **Address**
  - street
  - town
  - zipCode

- **Company**
  - name
  - phoneNr : Phone

- **Session**
  - startDate

- **Room**

- **Enrolment**
  - canceled : Boolean

- **Person**
  - firstName
  - lastName

- **AbisPerson**

**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)

Client

Shape
  - color : String
  + area() : double
  + getColor() : Color

Point

1 origin

Rectangle
  - height : double
  - width : double
  + area() : double
  + getHeight() : double
  + getWidth() : double

Circle
  - radius : double
  + area() : double
  + getRadius() : double

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
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**

- **AlarmClock**
  - addAlarmnListener( AlarmListener lis )
  - publishAlarmEvent( time )
  - setTime( newTime )

- **AlarmWindow**
  - onAlarmEvent( source, time )
  - display notification dialog box

- **Beeper**
  - onAlarmEvent( source, time )
  - beep

- **WatchDog**
  - onAlarmEvent( source, time )

- **javax.swing.JFrame**
  - ...setTitle() setVisible()...
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
**Sequence Diagram**

```plaintext
xfer(m:Money, a2)

a1:Account
check() : Boolean
[OK] minus(m)

b:Person

[b-a < 0.5 sec]

a2:Account
[OK] plus(m)

bank policy:
make sure owner is ok

a

[OK] minus(m)

a

[OK] plus(m)

b
```

---

**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 (..)

-Client

:Person

:Map

database

find(123) → get(123)

/map

[not found]

select * from people where id=123

create()

a result set

get person data

create with person data

mary

whatever(..)

mary : Person

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

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

1. Receive order
2. Calculate price
3. Check price with customer
4. If agreed: Assign to course, dispatch to order
   - Else: Find instructor
5. If instructor available: Assign to course
   - Else: Not available
6. Cancel order
7. Check price with customer
8. Cancel order

Activity diagram (..)

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:
- Stock
- Customer Service
- Finance Dept.

Events:
- receive order
- send invoice
- receive payment
- pick order
- deliver order
- close order

Flow:
- Order flow from Stock to Customer Service to Finance Dept.
- Payment flow from Customer Service to Finance Dept.
- Order closure from Finance Dept.
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, ...
Package Diagram

- Grouping interrelated classes
- Controlling dependencies
Package Diagram & System Architecture

Order UI

Swing

Order Application

Common (global)
- Date
- Money
- etc.

Domain

Orders

Customers

Database
• a modular, deployable, replaceable part of system (= physical!)
• encapsulates implementation
• exposes interfaces
Deployment Diagram

Shows where instances of components are running: on physical nodes (= some computer).
Want some more diagrams?

Check UML2, e.g.:
• Composite Structure
• Interaction Overview
• Timing
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