**UML - Unified Modeling Language**

Christoph Kessler, IDA, Linköpings universitet

Most slides by courtesy of Kristian Sandahl

---

**Software engineering process**

- **Requirements analysis**
- **System design**
- **Program design**
- **Coding**
- **Unit & integration testing**
- **System testing**
- **Acceptance testing**
- **Operation & Maintenance**

---

**Modeling as a Design Technique**

- Testing a physical entity before building it
- Communication with customers
- Visualization
- Reduction of complexity

- Models **supplement** natural language
- Models support understanding, design, documentation
- Creating a model forces you to take necessary design decisions
- **UML** is now the standard notation for modeling software.

---

**Literature on UML**

- **Official standard documents by OMG:**
  - [www.omg.org](http://www.omg.org), [www.uml.org](http://www.uml.org)
- **Current version is UML 2.0 (2004/2005)**
  - OMG documents: **UML Infrastructure, UML Superstructure**
- **Books:**
  - Pfleeger: *Software Engineering 3rd ed.*, 2005 (mostly Chapter 6)
  - And many others…

---

**UML: Different diagram types for different views of software**

**Modeling (logical) structure of software:**
- Static view: Class diagram
- Design view: Structure diagram, collaboration diagram, component diagram
- Use case view: Use case diagram

**Modeling behavior of software:**
- Activity view: Activity diagram
- State machine view: State machine diagram
- Interaction view: Sequence diagram, communication diagram

**Modeling physical structure of software:**
- Deployment view: Deployment diagram

**Modeling the model, and extending UML itself:**
- Model management view: Package Diagram
- Profiles

---

**Use-case modelling**

A use-case is:

“… a particular form or pattern or exemplar of usage, a scenario that begins with some user of the system initiating some transaction of sequence of interrelated events.”

Jacobson, m fl 1992: Object-oriented software engineering, Addison-Wesley
A CoffeeDrinker approaches the machine with her cup and a coin of SEK 5. She places the cup on the shelf just under the pipe. She then inserts the coin, and presses the button for coffee to get coffee according to default settings. Optionally, she might use other buttons to adjust the strength and decide to add sugar and/or whitener. The machine processes the coffee and rings a bell when it is ready. The CoffeeDrinker takes her cup from the shelf.
Extended class model:

CoffeeCustomer \( \xrightarrow{buys} \) CupOfCoffee

Porter \( \xrightarrow{buys} \) CanOfCoffee

Revised class model:

CoffeeCustomer \( \xrightarrow{buys} \) CupOfCoffee

Porter \( \xrightarrow{buys} \) CanOfCoffee

Class model with navigability:

CoffeeCustomer \( \xrightarrow{buys} \) CupOfCoffee

Porter \( \xrightarrow{buys} \) CanOfCoffee

Class model with inheritance and abstract classes:

CoffeeCustomer \( \xrightarrow{pay\ c\ coin\ )} \) IndividualCustomer

Porter \( \xrightarrow{get\ Cup\ ()\ get\ Can\ ()} \)

Abstract class (cannot be instantiated, only extended/specialized)

pay() method is inherited from CoffeeCustomer

Class model with aggregation:

Aggregation: part-of relationship

Machine

Interface

CoinHandler

Brewer

More relations between classes:

Topic \( \xrightarrow{1^{+}} \) Link

Encyclopedia \( \xrightarrow{1} \) Volume

Board \( \xrightarrow{row \[1..8\], column \[1..2\]} \) Square

Copy \( \xrightarrow{1} \) Book

Journal

aggregation

composition

Stronger form of aggregation: Composite has sole responsibility for managing its parts, e.g., allocation/deallocation, qualified association

constraint
The coffee machine class model

Classes and objects

Even small models take space. You need good drawing tools and a large sheet.

Reasoning about an arbitrary object

Sequence diagram

Sequence diagram with several objects

Communication diagram

Shows message flows with sequence numbers
Similar information as sequence diagram
For class CoinHandler:

- Checking:
  - falseCoin(): returnCoin(self)
  - insertCoin(): checkCoin(self)

For class CoinHandler:

- Start state marker
- State:
  - Idle
  - Event, causing transition
  - Action, reaction to the event

Activity Diagram

- Graph
  - Nodes are activities (actions)
  - Method invocations, operations, sending/receiving messages, handling events, creating/accessing/modify/delete objects, variables...
  - Data flow by input and output parameter pins

- Edges are control flow transitions
- To some degree dual to the state diagram

- Might be refined to a low-level specification:
  - cf. control flow graph (~ compiler IR)

- A Petri Net
  - Interpretation by moving tokens along edges
  - Models concurrency by multiple tokens for "current state"
  - Fork/join for synchronization

- Models real-world workflows

Activity Diagram

- Insert coin
- Brew coffee
- Add hot water to adjust strength
- Pour coffee

Other features...

- Comments
- Constraints in OCL (Object Constraint Language)
- Profiles: Collections of stereotypes for specific domains, e.g. Realtime-profile for UML
  - Customize (specialize) UML elements, e.g. associations
  - Can introduce own symbols

- MOF (Meta-Object Facility):
  - UML is specified in UML
  - Powerful mechanism for extending UML by adding new language elements

Homework Exercise

- Draw a class diagram for the following scenario:

  A customer, characterized by his/her name and phone number, may purchase reservations of tickets for a performance of a show. A reservation of tickets, annotated with the reservation date, can be either a reservation by subscription, in which case it is characterized by a subscription series number, or an individual reservation. A subscription series comprehends at least 3 and at most 6 tickets; an individual reservation at most one ticket. Every ticket is part of a subscription series or an individual reservation, but not both. Customers may have many reservations, but each reservation is owned by exactly one customer. Tickets may be available or not, and one may sell or exchange them. A ticket is associated with one specific seat in a specific performance, given by date and time, of a show, which is characterized by its name. A show may have several performances.