

# Software Design, Modelling and Analysis in UML

## Lecture 22: Wrapup

2014-02-12

Prof. Dr. Andreas Podolski, Dr. Bernd Westphal  
Albert-Ludwigs-Universität Freiburg, Germany

ADVERTISEMENTS:

THROUGH METHODS FOR C  
(Saurin)

REAL-TIME SYSTEMS

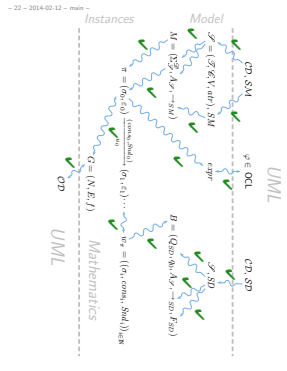
MASTER TEAM PROJECT

MASTER'S THESIS

### Content

- Lecture 1: Motivation and Overview
- Lecture 2: Semantical Model
- Lecture 3: Object Constraint Language (OCL)
- Lecture 4: OCL Semantics
- Lecture 5: Object Diagrams
- Lecture 6: Class Diagrams I
- Lecture 7: Class Diagrams II
- Lecture 8: Class Diagrams III
- Lecture 9: Class Diagrams III
- Lecture 10: Constructive Behaviour, State Machines Overview
- Lecture 11: Core State Machines I
- Lecture 12: Core State Machines II
- Lecture 13: Core State Machines III
- Lecture 14: Core State Machines IV
- Lecture 15: Core State Machines V, Runpoo
- Lecture 16: Hierarchical State Machines I
- Lecture 17: Hierarchical State Machines II
- Lecture 18: Live Sequence Charts I
- Lecture 19: Live Sequence Charts II
- Lecture 20: Inheritance I
- Lecture 21: Inheritance II, Inheritance III
- Lecture 22: Wrapup & Questions

### Course Path: Over Map



- Motivation
- Semantical Model
- OCL
- Object Diagrams
- Class Diagrams
- State Machines
- Live Sequence Charts
- Real-Time
- Components
- Inheritance
- Meta-Modeling (MDA, MDSE)

### Wrapup: Motivation

- Lecture 1: Motivation and Overview
- Lecture 2: Semantical Model
- Lecture 3: Object Constraint Language (OCL)
- Lecture 4: OCL Semantics
- Lecture 5: Object Diagrams
- Lecture 6: Class Diagrams
- Lecture 7: Class Diagrams I
- Lecture 8: Class Diagrams II
- Lecture 9: Class Diagrams III
- Lecture 10: Constructive Behaviour, State Machines Overview
- Lecture 11: Core State Machines I
- Lecture 12: Core State Machines II
- Lecture 13: Core State Machines III
- Lecture 14: Core State Machines IV
- Lecture 15: Core State Machines V, Runpoo
- Lecture 16: Hierarchical State Machines I
- Lecture 17: Hierarchical State Machines II
- Lecture 18: Live Sequence Charts I
- Lecture 19: Live Sequence Charts II
- Lecture 20: Inheritance I
- Lecture 21: Inheritance II, Inheritance III
- Lecture 22: Wrapup & Questions

### Wrapup: Motivation

- Lecture 1:
- Educational Objectives: you should
    - be able to explain the term model.
    - know the idea (and hopes and promises) of model-driven SW development.
    - be able to explain how UML fits into this general picture.
    - know what **we** we've done in the course, and why.
    - thus be able to decide whether you want to stay with us...
  - How can UML help with software development?
  - Where is which sublanguage of UML useful?
  - For what purpose? With what drawbacks?

## Wrapup: Examining Motivation

- what is a model? (or example?)
- “a model is an image or a pre-image” — of what? please explain!
- when is a model a good model?
- what is model-based software engineering?
- MDA? MDSE?
- what do people hope to gain from MSSE? Why? Hope Justified?
- what are the fundamental pre-requisites for that?
- what are purposes of modelling guidelines?
- could you illustrate this with examples?
- how can we establish/enforce them? can tools or procedures help?
- what’s the qualitative difference between the modelling guideline – all association ends have a multiplicity – and – all states/machines are deterministic? –
- . . . .

6/25

## Wrapup: Examining Motivation

- what is UML (definition)? why?
- why is it (definition) not why?
- how does UML relate to programming languages?
- what are the intentions of UML?
- what is the history of UML? Why could it be useful to know that?
- where can (what part of) UML be used in MSSE?
- for what purpose? to improve what?
- we discussed a notion of “UML mode” by M. Fowler.
- what is that? why is it useful to think about it?

7/25

## Wrapup: Examining “The Big Picture”

- what kinds of diagrams does UML offer?
- what is the purpose of the X diagram?
- what do the diagrams X and Y have in common?
- what is a UML model (our definition)? what does it mean?
- what is the difference between well-formedness rules and modelling guidelines?
- what is meta-modelling?
- could you explain it on the example of UML?
- what is a class diagram in the context of meta-modelling?
- what benefits do people see in meta-modelling?
- the standard is split into the two documents “infrastructure” and “superstructure” – what is the rationale behind that?
- in what modelling languages is UML modelled?

8/25

## Wrapup: Modelling Structure

- Lecture 1: Motivation and Overview
- Lecture 2: Semantical Model
- Lecture 3: Object Constraint Language (OCL)
- Lecture 4: OCL Semantics
- Lecture 5: Object Diagrams
- Lecture 6: Class Diagrams
- Lecture 7: State Diagrams and Visibility
- Lecture 8: Class Diagrams II
- Lecture 9: Class Diagrams III
- Lecture 10: Constructive Behaviour, State Machine Overview
- Lecture 11: Core State Machines I
- Lecture 12: Core State Machines II
- Lecture 13: Core State Machines III
- Lecture 14: Core State Machines IV
- Lecture 15: Core State Machines V, Rapsoy
- Lecture 16: Hierarchical State Machines I
- Lecture 17: Hierarchical State Machines II
- Lecture 18: Live Sequence Charts I
- Lecture 19: Live Sequence Charts II
- Lecture 20: Invariants I
- Lecture 21: Invariants II, Inheritance, Inheritance II
- Lecture 22: Wrapup & Questions

9/25

## Wrapup: Modelling Structure

- **Lecture 2:**
  - **Educational Objectives:** Capabilities for these tasks/questions:
    - Why is UML of the form it is?
    - Shall one feel bad if not using all diagrams during software development?
    - What is a signature, an object, a system state, etc? What’s the purpose in the course?
    - How do Basic Object System Signatures relate to UML class diagrams?
  - **Lecture 3 & 4:**
    - **Educational Objectives:** Capabilities for these tasks/questions:
      - Please explain/read out this OCL constraint. Is it well-typed?
      - Does this OCL constraint hold in this (completed) system state?
      - Can you think of a system state satisfying this constraint?
      - Please un-abbreviate all abbreviations in this OCL expression. In what sense is OCL a three-valued logic? For what purpose?
      - How are  $\exists^c()$  and  $\forall^c$  related?

10/25

## Wrapup: Modelling Structure

- **Lecture 5:**
  - **Educational Objectives:** Capabilities for following tasks/questions:
    - What is an object diagram? What are object diagrams good for?
    - When is an object diagram called partial? What are partial ones good for?
    - How are system states and object diagrams related?
    - What does it mean that an OCL expression is satisfiable?
    - When is a set of OCL constraints said to be consistent?
    - Can you think of an object diagram which violates this OCL constraint?
    - Is this UML model  $M_1$  consistent wrt.  $inv(M_1)$ ?
  - **Lecture 6:**
    - **Educational Objectives:** Capabilities for following tasks/questions:
      - What is a class diagram?
      - For what purposes are class diagrams useful?
      - Could you please map this class diagram to a signature?
      - Could you please map this signature to a class diagram?
      - What is a stereotype? What does it mean? For what can it be useful?

11/25

Lecture 7:

- **Educational Objectives:** Capabilities for following tasks/questions.
  - Is this OCL expression well-sorted or not? Why?
  - How/in what form did we define well-definedness?
  - What is visibility good for? Where is it used?

Lecture 8 & 9:

- **Educational Objectives:** Capabilities for following tasks/questions.
  - Please explain/illustrate this class diagram with associations.
  - Which annotations of an association arrow are (semantically) relevant? In what sense? For what?
  - What's a role name? What's it good for?
  - What's "multiplicity"? How did we treat them semantically?
  - What is "reading direction", "navigability", "ownership", ...?
  - What's the difference between "aggregation" and "composition"?

12/25

Lecture 9:

- **Educational Objectives:** Capabilities for following tasks/questions.
  - What are purposes of modelling guidelines? (Example?)
  - When is a class diagram a good class diagram?
  - Discuss the style of this class diagram.

Lecture 20 & 21:

- **Educational Objectives:** Capabilities for following tasks/questions.
  - What's the effect of inheritance on System States?
  - What does the Liskov Substitution Principle mean regarding structure?
  - What is the subject, what the uplink semantics of inheritance?
  - What's the idea of Meta-Modelling?

13/25

- Lecture 1: Motivation and Overview
- Lecture 2: Semantical Model
- Lecture 3: Object Centred Language (OCL)
- Lecture 4: OCL Semantics
- Lecture 5: Object Diagrams
- Lecture 6: Class Diagrams
- Lecture 7: State Diagrams I: Visibility
- Lecture 8: Class Diagrams II
- Lecture 9: Class Diagrams III
- Lecture 10: Constructive Behaviour, State Machines Overview
- Lecture 11: Core State Machines I
- Lecture 12: Core State Machines II
- Lecture 13: Core State Machines III
- Lecture 14: Core State Machines IV
- Lecture 15: Core State Machines V: Rigorously
- Lecture 16: Hierarchical State Machines I
- Lecture 17: Hierarchical State Machines II
- Lecture 18: Live Sequence Charts I
- Lecture 19: Live Sequence Charts II
- Lecture 20: Interference I
- Lecture 21: Interference II
- Lecture 22: Wrapup & Questions

14/25

Main and General:

- **Educational Objectives:** Capabilities for following tasks/questions.
  - What does this State Machine mean?
  - What happens if I inject this event?
  - Can you please model the following behaviour. (And convince readers that your model is correct.)

15/25

Lecture 10:

- **Educational Objectives:** Capabilities for following tasks/questions.
  - What's the difference between reflexive and constructive descriptions of behaviour?
  - What's the Basic Causality Model?
  - What does the standard say about the dispatching method?
  - What is (intuitively) a run-to-completion step?

Lecture 11:

- **Educational Objectives:** Capabilities for following tasks/questions.
  - Can you please model the following behaviour.
    - What is "trigger" guard action?
    - Please underline this abbreviated transition annotation.
    - What is an ether? Example? Why did we introduce it?
    - What's the difference: signal, signal event, event, trigger, reception, consumption?
    - What's a system configuration?
    - When is an object stable (intuitively, formally)?

16/25

Lecture 12 & 13:

- **Educational Objectives:** Capabilities for following tasks/questions.
  - What is a transformer? Example? Why did we introduce it?
  - What is a re-use semantics?
  - What is a non-re-use semantics?
  - What labelled transition system is induced by a UML model?
  - What is discard, dispatch, comment?
  - What's the meaning of stereotype "signal env"?
  - Does environment interaction necessarily occur?
  - What happens on "division by 0"?

Lecture 14 & 15:

- **Educational Objectives:** Capabilities for following tasks/questions.
  - What is a step (definition)? Run-to-completion step (definition)? Microstep (intuition)?
  - Do objects always finally become stable?
  - In what sense is our RTC semantics not compositional?

17/25

## Wrapup: Modelling Behaviour Constructive

### Lecture 16:

- **Educational Objectives:** Capabilities for following tasks/questions.
  - What's a kind of a state? What's a pseudo-state?
  - What's a region? What's it good for?
  - What is: entry, exit, do, internal transition?
  - What's a completion event? What has it to do with the ether?

### Lecture 17:

- **Educational Objectives:** Capabilities for following tasks/questions.
  - What's a state configuration?
  - When are two states orthogonal? When consistent?
  - What's the depth of a state? Why care?
  - What is the set of enabled transitions in this system configuration and this state machine?

## Wrapup: Modelling Behaviour Constructive

### Lecture 18:

- **Educational Objectives:** Capabilities for following tasks/questions.
  - What's a history state? Deep vs. shallow?
  - What is: junction, choice, terminate?
  - What is the idea of "deferred events"?
  - What is a passive object? Why are passive reactive objects special? What did we do in this case?
  - What's a behavioural feature? How can it be implemented?

## Wrapup: Modelling Behaviour Reflective

- Lecture 1: Motivation and Overview
- Lecture 2: Semantical Model
- Lecture 3: Object Constraint Language (OCL)
- Lecture 4: OCL Semantics
- Lecture 5: Object Diagrams
- Lecture 6: Class Diagrams
- Lecture 7: State Diagrams and Visibility
- Lecture 8: Class Diagrams II
- Lecture 9: Class Diagrams III
- Lecture 10: Constructive Behaviour, State Machines Overview
- Lecture 11: Core State Machines I
- Lecture 12: Core State Machines II
- Lecture 13: Core State Machines III
- Lecture 14: Core State Machines IV
- Lecture 15: Core State Machines V, Runloop
- Lecture 16: Hierarchical State Machines I
- Lecture 17: Hierarchical State Machines II
- Lecture 18: Live Sequence Charts I
- Lecture 19: Live Sequence Charts II
- Lecture 20: Inheritance I
- Lecture 21: Inheritance II, Inheritance III
- Lecture 22: Wrapup & Questions

## Wrapup: Modelling Behaviour Reflective

### Lecture 18: & 19:

- **Educational Objectives:** Capabilities for following tasks/questions.
  - Is each LSC description of behaviour necessarily reflective?
  - There exists another distinction between "intra-object" and "inter-object" behaviour. Discuss in the context of UML.
  - What does this LSC mean?
  - Are this UML model's state machine consistent with the interactions?
  - Please provide a UML model which is consistent with this LSC.
  - What is: activation (mode, condition), hot/cold condition, pre-chart, out, hot/cold location, local invariant, legal ext., hot/cold chart etc.?

## Wrapup: Inheritance

- Lecture 1: Motivation and Overview
- Lecture 2: Semantical Model
- Lecture 3: Object Constraint Language (OCL)
- Lecture 4: OCL Semantics
- Lecture 5: Object Diagrams
- Lecture 6: Class Diagrams
- Lecture 7: State Diagrams and Visibility
- Lecture 8: Class Diagrams II
- Lecture 9: Class Diagrams III
- Lecture 10: Constructive Behaviour, State Machines Overview
- Lecture 11: Core State Machines I
- Lecture 12: Core State Machines II
- Lecture 13: Core State Machines III
- Lecture 14: Core State Machines IV
- Lecture 15: Core State Machines V, Runloop
- Lecture 16: Hierarchical State Machines I
- Lecture 17: Hierarchical State Machines II
- Lecture 18: Live Sequence Charts I
- Lecture 19: Live Sequence Charts II
- Lecture 20: Inheritance I
- Lecture 21: Inheritance II, Inheritance III
- Lecture 22: Wrapup & Questions

## Wrapup: Inheritance

### Lecture 20 & 21:

- **Educational Objectives:** Capabilities for following tasks/questions.
  - What's the effect of inheritance on LSCs, State Machines, System States?
  - What is the Liskov Substitution Principle?
  - What is commonly understood under (behavioral) subtyping?
  - What is the subtle: what the uplink semantics of inheritance?
  - What is late/early binding?
  - What's the idea of Meta-Modelling?

*Meta*

24/a

*Hmm...*

\* Open book or closed book...?

25/a