Software Design, Modelling and Analysis in UML

Lecture 21: Meta-Modelling

2016-02-11

Prof. Dr. Andreas Podelski, **Dr. Bernd Westphal**

Albert-Ludwigs-Universität Freiburg, Germany

Contents & Goals

Last Lecture:

- Liskov Substitution Principle
- Inheritance: Domain Inclusion Semantics

This Lecture:

- **Educational Objectives:** Capabilities for following tasks/questions.
  - What is the idea of meta-modelling?
  - How does meta-modelling relate to UML?
- **Content:**
  - The UML Meta Model
  - Wrapup & Questions

Meta-Modelling: Idea

Meta-Modelling: Why and What

- **Meta-Modelling** is one major prerequisite for understanding
  - the standard documents OMG (2007a,b), and
  - the MDA ideas of the OMG.
- The idea is somewhat **simple**:
  - if a **modelling language** is about modelling **things**,
  - and if UML models are **things**,
  - then why not **model** UML models using a modelling language?
- In other words:

  Why not have a model $\mathcal{M}_U$ such that
  - the set of legal instances of $\mathcal{M}_U$

  is
  - the set of well-formed (!) UML models.

Meta-Modelling: Example

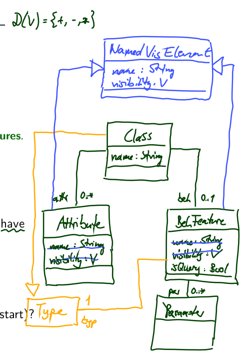
For example, let's consider a class.

- A **class** has (**among others**)
  - a **name**,
  - any number of **attributes**,
  - any number of **behavioural features**.

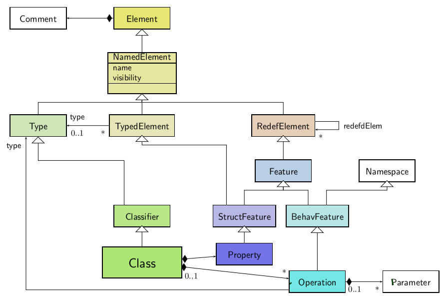
  Each of the latter two has
  - a **name** and
  - a **visibility**.

  Behavioural features in addition have
  - a boolean attribute **isQuery**,
  - any number of parameters,
  - a return type.

Can we model this (in UML, for a start)?

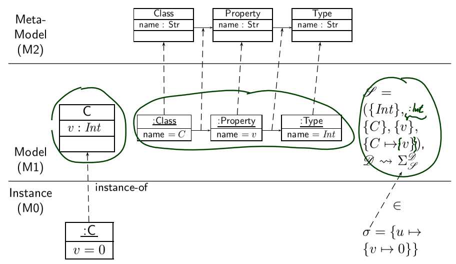


UML Meta-Model: Extract from UML 2.0 Standard

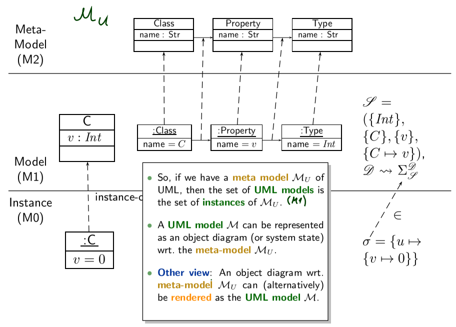


Meta-Modelling: Principle

Modelling vs. Meta-Modelling



Modelling vs. Meta-Modelling



- So, if we have a **meta model** $\mathcal{M}_U$ of UML, then the set of **UML models** is the set of **instances** of $\mathcal{M}_U$. (*)
- A **UML model** $\mathcal{M}$ can be represented as an object diagram (or system state) wrt. the **meta-model** $\mathcal{M}_U$.
- **Other view**: An object diagram wrt. **meta-model** $\mathcal{M}_U$ can (alternatively) be **rendered** as the **UML model** $\mathcal{M}$.

Well-Formedness as Constraints in the Meta-Model

- The set of **well-formed UML models** can be defined as the set of object diagrams satisfying all constraints of the **meta-model**.

  Constraint example,

  > "[2] Generalization hierarchies must be directed and acyclical. A classifier cannot be both a transitively general and transitively specific classifier of the same classifier.
  >
  > not self . allParents() -> includes(self)" (OMG, 2007b, 53)

- The other way round:

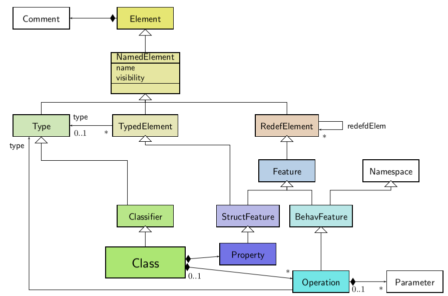
  Given a **UML model** $\mathcal{M}$, unfold it into an object diagram O_1 wrt. $\mathcal{M}_U$.

  If O_1 is a **valid** object diagram of $\mathcal{M}_U$ (i.e. satisfies all invariants from $Inv(\mathcal{M}_U)$), then $\mathcal{M}$ is a well-formed UML model.

  That is, if we have an object diagram **validity checker** for of the meta-modelling language, then we have a **well-formedness checker** for UML models.

The UML 2.x Standard Revisited

Claim: Extract from UML 2.0 Standard



Classes (OMG, 2007b, 32)

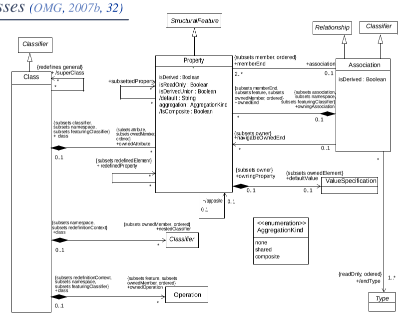


Figure 7.12 - Classes diagram of the Kernel package

Operations (OMG, 2007b, 31)

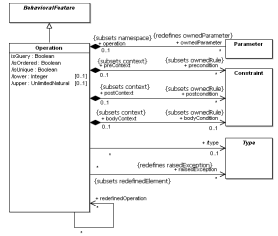


Figure 7.11 - Operations diagram of the Kernel package

Operations (OMG, 2007b, 30)

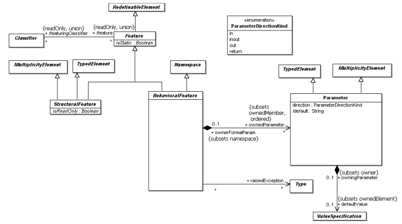


Figure 7.10 - Features diagram of the Kernel package

Classifiers (OMG, 2007b, 29)

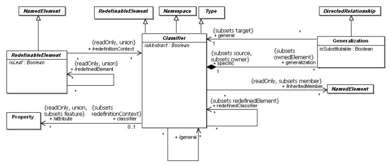


Figure 7.9 - Classifiers diagram of the Kernel package

Namespaces (OMG, 2007b, 26)

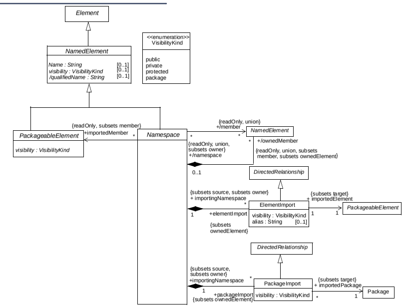


Figure 7.4 - Namespaces diagram of the Kernel package

Root Diagram (OMG, 2007b, 25)

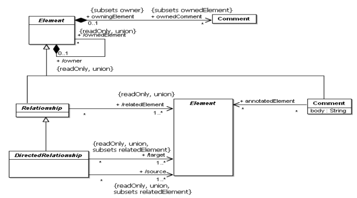


Figure 7.3 - Root diagram of the Kernel package

Interesting: Declaration/Definition (OMG, 2007b, 424)

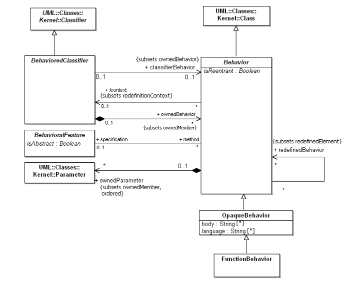
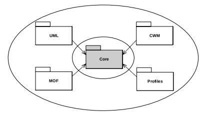


Figure 13.6 - Common Behavior

UML Architecture (OMG, 2003, 8)

- Meta-modelling has already been used for UML 1.x.
- For UML 2.0, the request for proposals (RFP) asked for a separation of concerns: **Infrastructure** and **Superstructure**.
- **One reason**: sharing with MOF (see later) and, e.g., CWM.



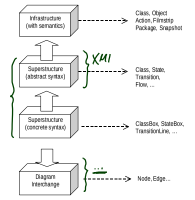


Figure 0-1 *Overview of architecture*

UML Superstructure Packages (OMG, 2007a, 15)

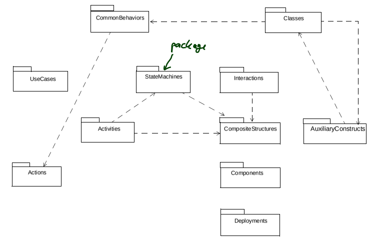


Figure 7.5 - The top-level package structure of the UML 2.1.1 Superstructure

Reading the Standard

Table of Contents

UML Superstructure Specification, v2.1.2 i

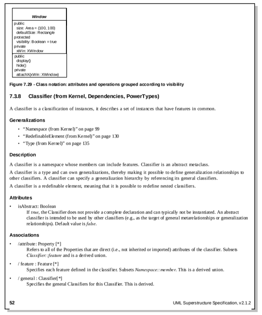
Reading the Standard

ii UML Superstructure Specification, v2.1.2

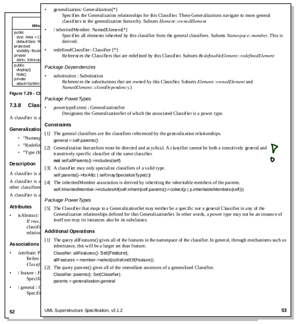
Reading the Standard

UML Superstructure Specification, v2.1.2 iii

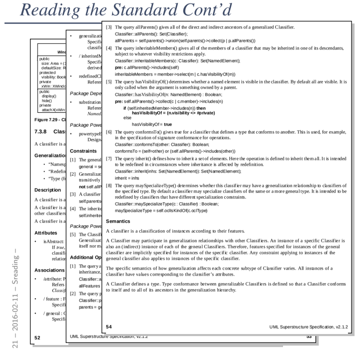
Reading the Standard Cont'd



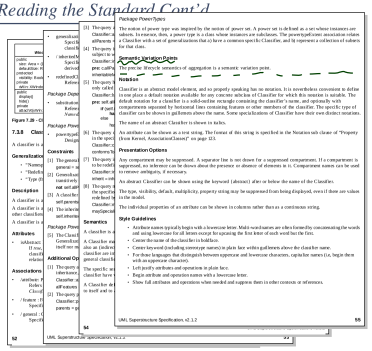
Reading the Standard Cont'd



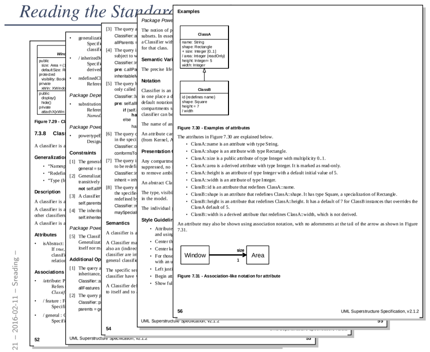
Reading the Standard Cont'd



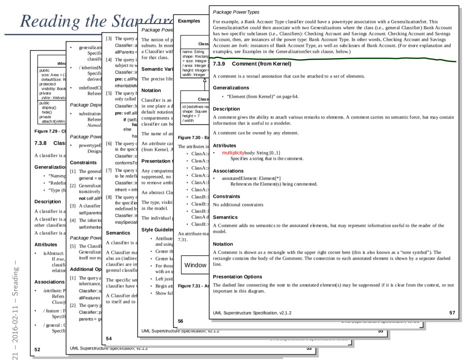
Reading the Standard Cont'd



Reading the Standard Cont'd



Reading the Standard Cont'd



Meta Object Facility (MOF)

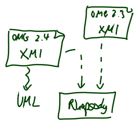
Open Questions...

- Now you've been "**tricked**".
  - We didn't tell what the **modelling language** for meta-modelling is.
  - We didn't tell what the **is-instance-of** relation of this language is.
- **Idea**: have a **minimal object-oriented core** comprising the notions of **class**, **association**, **inheritance**, **etc.** with "self-explaining" semantics.
- This is **Meta Object Facility** (MOF), which (more or less) coincides with UML Infrastructure OMG (2007a).
- So: things on meta level
  - M0 are object diagrams/system states
  - M1 are **words of the language UML**
  - M2 are **words of the language MOF**
  - M3 are **words of the language** MOF ?

MOF Semantics

- One approach:
  - Treat it with **our signature-based theory**
  - This is (in effect) the right direction, but may require new (or extended) signatures for each level.
- Other approach:
  - Define a **generic**, **graph based** "is-instance-of" relation.
  - Object diagrams (that **are** graphs) then **are** the system states — not **only graphical representations** of system states.
  - If this works out, good: We can easily experiment with different language designs, e.g. different flavours of UML that immediately have a semantics.
  - Most interesting: also do generic definition of behaviour within a closed modelling setting, but this is clearly still research, e.g. Buschermöhle and Oelerink (2008).

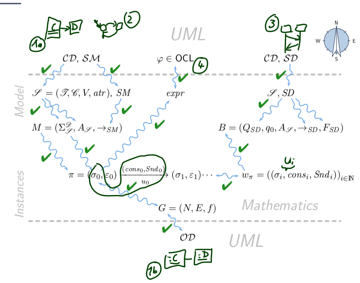
Benefits



- In particular:
  - Benefits for **Modelling Tools**.
  - Benefits for **Language Design**.
  - Benefits for **Code Generation and MDA**.

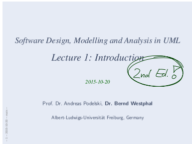
And That's It!

The Map



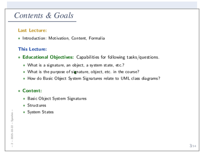
Content

- **Lecture 1**: Introduction



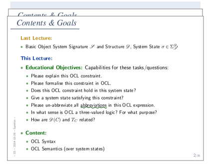
Content

- **Lecture 1**: Introduction
- **Lecture 2**: Semantical Model



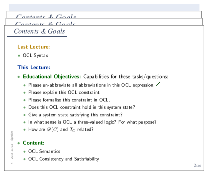
Content

- **Lecture 1**: Introduction
- **Lecture 2**: Semantical Model
- **Lecture 3**: Object Constraint Language (OCL)



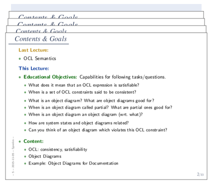
Content

- **Lecture 1**: Introduction
- **Lecture 2**: Semantical Model
- **Lecture 3**: Object Constraint Language (OCL)
- **Lecture 4**: OCL Semantics



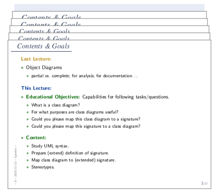
Content

- **Lecture 1**: Introduction
- **Lecture 2**: Semantical Model
- **Lecture 3**: Object Constraint Language (OCL)
- **Lecture 4**: OCL Semantics
- **Lecture 5**: Object Diagrams



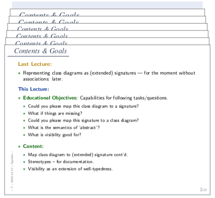
Content

- **Lecture 1**: Introduction
- **Lecture 2**: Semantical Model
- **Lecture 3**: Object Constraint Language (OCL)
- **Lecture 4**: OCL Semantics
- **Lecture 5**: Object Diagrams
- **Lecture 6**: Class Diagrams I



Content

- **Lecture 1**: Introduction
- **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



Contents & Goals

Last Lecture:
- Representing class diagrams as (extended) signatures — for the moment without associations: later.

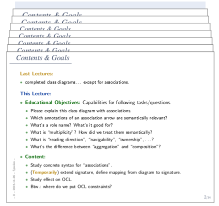
This Lecture:
- **Educational Objectives:** Capabilities for following tasks/questions.
  - Could you please map this class diagram to a signature?
  - What if things are missing?
  - Could you please map this signature to a class diagram?
  - What is the semantics of 'abstract'?
  - What is visibility good for?
- **Content:**
  - Map class diagram to (extended) signature cont'd.
  - Stereotypes – for documentation.
  - Visibility as an extension of well-typedness.

– 7 – 2015-11-17 – Sprelim –

2/23

Content

- **Lecture 1**: Introduction
- **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



Contents & Goals

Last Lectures:
- completed class diagrams... except for associations.

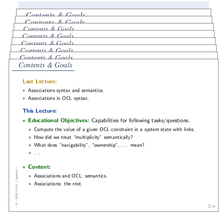
This Lecture:
- **Educational Objectives:** Capabilities for following tasks/questions.
  - Please explain this class diagram with associations.
  - Which annotations of an association arrow are semantically relevant?
  - What's a role name? What's it good for?
  - What is "multiplicity"? How did we treat them semantically?
  - What is "reading direction", "navigability", "ownership", ...?
  - What's the difference between "aggregation" and "composition"?
- **Content:**
  - Study concrete syntax for "associations".
  - (Temporarily) extend signature, define mapping from diagram to signature.
  - Study effect on OCL.
  - Btw.: where do we put OCL constraints?

– 8 – 2015-11-26 – Sprelim –

2/34

Content

- **Lecture 1**: Introduction
- **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 IV



Contents & Goals

Last Lecture:
- Associations syntax and semantics.
- Associations in OCL syntax.

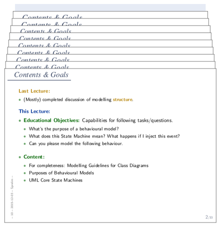
This Lecture:
- **Educational Objectives:** Capabilities for following tasks/questions.
  - Compute the value of a given OCL constraint in a system state with links.
  - How did we treat "multiplicity" semantically?
  - What does "navigability", "ownership", ... mean?
  - ...
- **Content:**
  - Associations and OCL: semantics.
  - Associations: the rest.

– 9 – 2015-12-01 – Sprelim –

2/40

Content

- **Lecture 1**: Introduction
- **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 IV
- **Lecture 10**: State Machines Overview



Contents & Goals

Last Lecture:
- (Mostly) completed discussion of modelling structure.

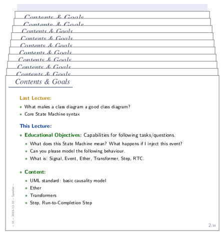
This Lecture:
- **Educational Objectives:** Capabilities for following tasks/questions.
  - What's the purpose of a behavioural model?
  - What does this State Machine mean? What happens if I inject this event?
  - Can you please model the following behaviour.
- **Content:**
  - For completeness: Modelling Guidelines for Class Diagrams
  - Purposes of Behavioural Models
  - UML Core State Machines

– 10 – 2015-12-03 – Sprelim –

2/33

Content

- **Lecture 1**: Introduction
- **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 IV
- **Lecture 10**: State Machines Overview
- **Lecture 11**: Core State Machines I



Contents & Goals

Last Lecture:
- What makes a class diagram a good class diagram?
- Core State Machine syntax

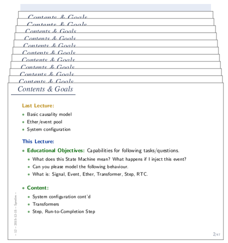
This Lecture:
- **Educational Objectives:** Capabilities for following tasks/questions.
  - What does this State Machine mean? What happens if I inject this event?
  - Can you please model the following behaviour.
  - What is: Signal, Event, Ether, Transformer, Step, RTC.
- **Content:**
  - UML standard: basic causality model
  - Ether
  - Transformers
  - Step, Run-to-Completion Step

– 11 – 2015-12-10 – Sprelim –

2/34

Content

- **Lecture 1**: Introduction
- **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 IV
- **Lecture 10**: State Machines Overview
- **Lecture 11**: Core State Machines I
- **Lecture 12**: Core State Machines II



Contents & Goals

Last Lecture:
- Basic causality model
- Ether/event pool
- System configuration

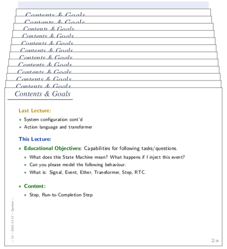
This Lecture:
- **Educational Objectives:** Capabilities for following tasks/questions.
  - What does this State Machine mean? What happens if I inject this event?
  - Can you please model the following behaviour.
  - What is: Signal, Event, Ether, Transformer, Step, RTC.
- **Content:**
  - System configuration cont'd
  - Transformers
  - Step, Run-to-Completion Step

– 12 – 2015-12-15 – Sprelim –

2/47

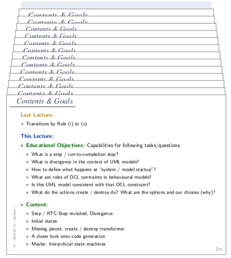
Content

- **Lecture 1**: Introduction
- **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 IV
- **Lecture 10**: State Machines Overview
- **Lecture 11**: Core State Machines I
- **Lecture 12**: Core State Machines II
- **Lecture 13**: Core State Machines III



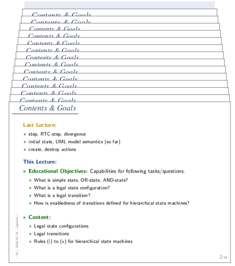
Content

- **Lecture 1**: Introduction
- **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 IV
- **Lecture 10**: 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



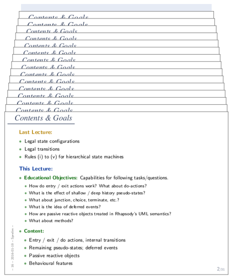
Content

- **Lecture 1**: Introduction
- **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 IV
- **Lecture 10**: 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**: Hierarchical State Machines I



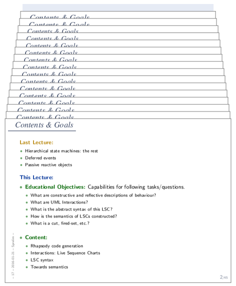
Content

- **Lecture 1**: Introduction
- **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 IV
- **Lecture 10**: 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**: Hierarchical State Machines I
- **Lecture 16**: Hierarchical State Machines II



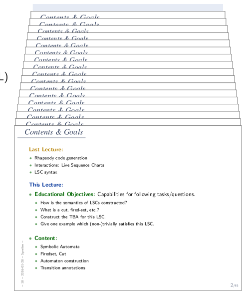
Content

- **Lecture 1**: Introduction
- **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 IV
- **Lecture 10**: 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**: Hierarchical State Machines I
- **Lecture 16**: Hierarchical State Machines II
- **Lecture 17**: Live Sequence Charts I



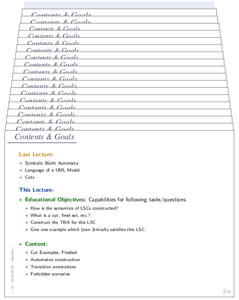
Content

- **Lecture 1**: Introduction
- **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 IV
- **Lecture 10**: 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**: Hierarchical State Machines I
- **Lecture 16**: Hierarchical State Machines II
- **Lecture 17**: Live Sequence Charts I
- **Lecture 18**: Live Sequence Charts II



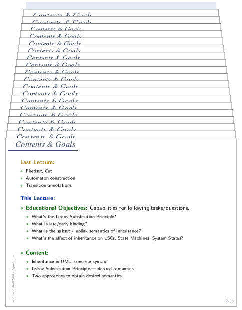
Content

- **Lecture 1**: Introduction
- **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 IV
- **Lecture 10**: 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**: Hierarchical State Machines I
- **Lecture 16**: Hierarchical State Machines II
- **Lecture 17**: Live Sequence Charts I
- **Lecture 18**: Live Sequence Charts II
- **Lecture 19**: Live Sequence Charts III



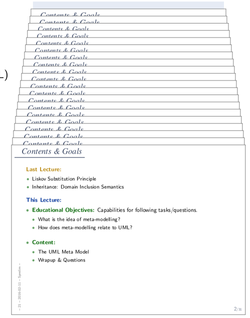
Content

- **Lecture 1**: Introduction
- **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 IV
- **Lecture 10**: 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**: Hierarchical State Machines I
- **Lecture 16**: Hierarchical State Machines II
- **Lecture 17**: Live Sequence Charts I
- **Lecture 18**: Live Sequence Charts II
- **Lecture 19**: Live Sequence Charts III
- **Lecture 20**: Inheritance



Content

- **Lecture 1**: Introduction
- **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 IV
- **Lecture 10**: 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**: Hierarchical State Machines I
- **Lecture 16**: Hierarchical State Machines II
- **Lecture 17**: Live Sequence Charts I
- **Lecture 18**: Live Sequence Charts II
- **Lecture 19**: Live Sequence Charts III
- **Lecture 20**: Inheritance
- **Lecture 21**: Meta-Modelling



References

References

Buschermöhle, R. and Oelerink, J. (2008). Rich meta object facility. In *Proc. 1st IEEE Int'l workshop UML and Formal Methods*.

OMG (2003). Uml 2.0 proposal of the 2U group, version 0.2, `http://www.2uworks.org/uml2submission`.

OMG (2007a). Unified modeling language: Infrastructure, version 2.1.2. Technical Report formal/07-11-04.

OMG (2007b). Unified modeling language: Superstructure, version 2.1.2. Technical Report formal/07-11-02.

OMG (2011a). Unified modeling language: Infrastructure, version 2.4.1. Technical Report formal/2011-08-05.

OMG (2011b). Unified modeling language: Superstructure, version 2.4.1. Technical Report formal/2011-08-06.