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)*?
Meta-Modelling: Principle

Modelling vs. Meta-Modelling

```
Class name : Str
Property name : Str
Type name : Str

C
v : Int

S = ( {Int}, {v}, {C \mapsto v} ), D \Rightarrow \Sigma

σ = \{ u \mapsto \{ v \mapsto 0 \} \}
```

**Modelling vs. Meta-Modelling**

- **Meta-Model (M2)**
  - Class name: `Str`
  - Property name: `Str`
  - Type name: `Str`

- **Model (M1)**
  - `C`
    - `v : Int`
    - Class name: `C`
      - `name = C`
    - Property name: `v`
      - `name = v`
    - Type name: `Int`
      - `name = Int`

  \[ \mathcal{F} = \{ \{ \text{Int} \} , \{ C \} , \{ v \} , \{ C \mapsto v \} \} , \mathcal{D} \sim \Sigma \mathcal{D} \]

- **Instance (M0)**
  - `\text{instance-of} \sigma = \{ u \mapsto \{ v \mapsto 0 \} \} \in \mathcal{M}_U`

  \[ \sigma = \{ u \mapsto \{ v \mapsto 0 \} \} \]

- 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.

\[ \text{not self . allParents}() \rightarrow \text{includes(self)} \]" (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 \( \text{Inv}(\mathcal{M}_U) \)), 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.
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)

Root Diagram (OMG, 2007b, 25)
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 7.5 - The top-level package structure of the UML 2.1.1 Superstructure
7.3.8 Classifier (from Kernel, Dependencies, PowerTypes)

A classifier is a classification of instances; it describes a set of instances that have features in common.

**Description**

A classifier is a namespace whose members can include features. A classifier is an abstract metaclass. A classifier is a redefinable element, meaning that it is possible to redefine nested classifiers.

**Attributes**

- **name**: Name
  
  A classifier is a redefinable element, meaning that it is possible to redefine nested classifiers.

- **isAbstract**: Boolean
  
  A classifier may only specialize classifiers of a valid type.

- **general**: Classifier
  
  References the Classifiers that are redefined by this Classifier. Subsets

- **feature**: Feature
  
  Specifies each feature defined in the classifier. Subsets

- **generalization**: Generalization
  
  Specifies the general Classifiers for this Classifier. This is derived.

- **powertypeExtent**: GeneralizationSet
  
  Designates the GeneralizationSet of which the associated Classifier is a power type.

**Constraints**

- **general**: Classifier
  
  (1) The general classifiers are the classifiers referenced by the generalization relationships.

- **feature**: Feature
  
  (2) Generalizations between classifiers must be disjoint and acyclic. A classifier cannot be both a generally general and unambiguously specific classifier of the same classifier.

**Additional Operations**

- **generalization Artículo(\*)**: Specifies the Generalization relationships for the Classifier. Three Generalization Article in an abstract classifier is the same Generalization Article in an abstract classifier.

- **general Artículo(\*)**: Specifies each feature defined in the classifier. Subsets

- **feature Artículo(\*)**: Specifies each feature defined in the classifier. Subsets

- **general Artículo(\*)**: Specifies the general Classifiers for this Classifier. This is derived.

- **powertypeExtent Artículo(\*)**: Designates the GeneralizationSet of which the associated Classifier is a power type.

- **baseClassifier Article**: References the Classifiers that are redefined by this Classifier. Subsets

- **feature Article**: Specifies each feature defined in the classifier. Subsets

- **general Artículo(\*)**: Specifies the general Classifiers for this Classifier. This is derived.

- **powertypeExtent Artículo(\*)**: Designates the GeneralizationSet of which the associated Classifier is a power type.

- **feature Artículo(\*)**: Specifies each feature defined in the classifier. Subsets
### 7.3.8 Classifier (from Kernel, Dependencies, PowerTypes)

A classifier is a namespace whose members can include features. Classifier is an abstract metaclass.

#### Attributes
- **Associations**
  - / feature : Feature [*]  
  - xWin: XWindow
  - defaultSize: Rectangle
  - size: Area = (100, 100)
  - redefinedClassifier: Classifier [*]

#### Constraints
- **Generators**
  - / Classifier
  - / Classifier
  - / Classifier

#### References
- **Classifiers that are redefined by this Classifier.** Subsets
- **Substitutions that are owned by this Classifier.** Subsets

#### Package Dependencies
- **RedefinableElement (from Kernel)** on page 130
- **Package Dependencies**

---

**Reading the Standard Cont’d**

The notion of power type was inspired by the notion of power set. A power set is defined as a set whose members are subsets. In essence, draw a power type as a rectangle whose elements are subsets. The power type is associated with a set of a specific power type that has a common specific Classifier and it can be displayed, even if there is no value in the model.

**Semantic Diagrams**

Classification is an abstract model element, not a properly speaking has no notation. It is inconvenient to display in one place a default notation available for any concept. In the figure, Classifier is for the moment in the model.

### Style Guidelines
- **Attribute names** typically begin with a lowercase letter. Multi-word names are often formed by concatenating words.
- **Center** the name of the classifier in boldface.
- **Align** attributes and operations in plain face.
- **Center** keywords (including stereotype names) in plain face above the classifier name.
- **Use** default notation for a classifier.

---

**Figure 7.3-11**

![Classifier UML diagram](image_url)
A classifier is a namespace whose members can include features. Classifier is an abstract metaclass.

Properties:
- /attribute: Property [*]
- /inheritedMember: NamedElement [*]

Associations:
- /ownedElement
- /clientDependency
- /ownedCollaboration

Contributes:
- /associate: Reference
- /associateEnd: Reference

Generalizations:
- /general: Classifier
- /specific: Classifier

Attributes:
- /name
- /icon
- /title
- /type
- /summary
- /description
- /additionalComment

References:
- /redefinedElement
- /redefinition
- /redefines

Constraints:
- /conformsTo
- /realizes
- /extends

Semantics:
- /isAbstract
- /isSealed
- /isFinal
- /isLeaf
- /isStatic

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
Contents

- Lecture 1: Introduction

Contents & Goals

- Lecture 1: Introduction
- Lecture 2: Semantical Model
Contents & Goals

Last Lecture:
• Basic Object System Signature \( S \) and Structure \( D \), System State \( \sigma \in \Sigma_D \).

This Lecture:
• Educational Objectives: Capabilities for these tasks/questions:
  - Please explain this OCL constraint.
  - Please formalise this constraint in OCL.
  - Does this OCL constraint hold in this system state?
  - Give a system state satisfying this constraint?
  - In what sense is OCL a three-valued logic? For what purpose?
  - How are \( D(C) \) and \( T_C \) related?

Content:
• OCL Syntax
• OCL Semantics (over system states)

Contents & Goals

Last Lecture:
• Introduction: Motivation, Content, Formalia

This Lecture:
• Educational Objectives:
  - What is a signature, an object, a system state, etc.?
  - What is the purpose of signature, object, etc. in the course?
  - How do Basic Object System Signatures relate to UML class diagrams?

Content:
• Basic Object System Signatures
• Structures
• System States

Contents & Goals

Last Lecture:
• OCL Syntax

This Lecture:
• Educational Objectives:
  - Please un-abbreviate all abbreviations in this OCL expression.
  - Please explain this OCL constraint.
  - Please formalise this constraint in OCL.
  - Does this OCL constraint hold in this system state?
  - Give a system state satisfying this constraint?
  - In what sense is OCL a three-valued logic? For what purpose?
  - How are \( D(C) \) and \( T_C \) related?

Content:
• OCL Syntax
• OCL Semantics
• OCL Consistency and Satisfiability

Contents & Goals

Last Lecture:
• Lecture 1: Introduction

This Lecture:
• Lecture 2: Semantical Model
• Lecture 3: Object Constraint Language (OCL)
• Lecture 4: OCL Semantics
Lecture 6
Please formalise this constraint in OCL.

Content:
When is a set of OCL constraints said to be consistent?
Please un-abbreviate all abbreviations in this OCL expression.

Lecture 4
Introduction: Motivation, Content, Formalia
Could you please map this class diagram to a signature?
Example: Object Diagrams for Documentation

OCL Syntax
How are Capabilities for following tasks/questions.
How are OCL: consistency, satisfiability
Educational Objectives:
Educational Objectives:
Object Diagrams
How are system states and object diagrams related?
Give a system state satisfying this constraint?
When is an object diagram an object diagram (wrt. what)?
OCL Semantics (over system states)
What does it mean that an OCL expression is satisfiable?
Please formalise this constraint in OCL.

Lecture 3
Capabilities for following tasks/questions.
OCL Semantics
What is a signature, an object, a system state, etc.?
OCL Syntax
In what sense is OCL a three-valued logic? For what purpose?
Basic Object System Signature
How are system states and object diagrams related?
Give a system state satisfying this constraint?
When is a set of OCL constraints said to be consistent?
Does this OCL constraint hold in this system state?
System States
Content:
Prepare (extend) definition of signature.
Please un-abbreviate all abbreviations in this OCL expression.

Lecture 5
Lecture 3
Please formalise this constraint in OCL.
Educational Objectives:
In what sense is OCL a three-valued logic? For what purpose?
Basic Object System Signature
How are system states and object diagrams related?
Give a system state satisfying this constraint?
When is a set of OCL constraints said to be consistent?
Does this OCL constraint hold in this system state?
System States
Content:
Does this OCL constraint hold in this system state?
For what purposes are class diagrams useful?
2
Lecture 1
2
Stereotypes.
Can you think of an object diagram which violates this OCL constraint?
Educational Objectives:
Capabilities for following tasks/questions.
Content:
OCL Semantics (over system states)
What does it mean that an OCL expression is satisfiable?
OCL Semantics
OCL Semantics
Lecture 4
What is the purpose of signature, object, etc. in the course?
How are system states and object diagrams related?
Capabilities for these tasks/questions:
Educational Objectives:
Object Diagrams
What is a signature, an object, a system state, etc.?
Content:
Does this OCL constraint hold in this system state?
For what purposes are class diagrams useful?
2
Please formalise this constraint in OCL.

What is a signature, an object, a system state, etc.?

Educational Objectives:
Map class diagram to (extended) signature cont’d.

What is visibility good for?

How are Basic Object System Signature related to UML class diagrams?

Study UML syntax.

How do Basic Object System Signatures relate to UML class diagrams?

When is an object diagram called partial? What are partial ones good for?

Lecture 6
Capabilities for these tasks/questions:

Educational Objectives:
Structures

Educational Objectives:

Does this OCL constraint hold in this system state?

Could you please map this class diagram to a signature?

Lecture 3
Content:
Can you think of an object diagram which violates this OCL constraint?

Please formalise this constraint in OCL.

When is a set of OCL constraints said to be consistent?

Object Diagrams

Could you please map this class diagram to a signature?

Lecture 7
Lecture 1
For what purposes are class diagrams useful?

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?

Please formalise this constraint in OCL.

Object Diagrams

Could you please map this class diagram to a signature?

Lecture 4
Content:

Visiblity as an extension of well-typedness.

Content:

OCL: consistency, satisfiability

Capabilities for following tasks/questions. When is an object diagram called partial? What are partial ones good for?

Object Diagrams

Give a system state satisfying this constraint?

Lecture 3
Lecture 2
Prepare (extend) definition of signature.

What are system states and object diagrams related?

What is "reading direction", "navigability", "ownership" , . . . ?

In what sense is OCL a three-valued logic? For what purpose?

Give a system state satisfying this constraint?

Lecture 6
Lecture 3
Lecture 4

Educational Objectives:

What does it mean that an OCL expression is satisfiable?

What if things are missing?

Please formalise this constraint in OCL.

What's the difference between "aggregation" and "composition"?

Please explain this class diagram with associations.

System States

Capabilities for following tasks/questions.

This Lecture:

Educational Objectives: Capabilities for following tasks/questions.

Could you please map this class diagram to a signature?

Lecture 2

Please formalise this constraint in OCL.

OCL Syntax

Could you please map this class diagram to a signature?

Lecture 1

Which annotations of an association arrow are semantically relevant?

Stereotypes – for documentation.

Please explain this OCL constraint.

Map class diagram to (extended) signature.

Lecture 3

For what purposes are class diagrams useful?

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?

Please formalise this constraint in OCL.

Object Diagrams

Could you please map this class diagram to a signature?

Lecture 7

Lecture 1

For what purposes are class diagrams useful?

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?

Please formalise this constraint in OCL.

Object Diagrams

Could you please map this class diagram to a signature?

Lecture 4
Content:

Visiblity as an extension of well-typedness.

Content:

OCL: consistency, satisfiability

Capabilities for following tasks/questions. When is an object diagram called partial? What are partial ones good for?

Object Diagrams

Give a system state satisfying this constraint?

Lecture 6
Lecture 3
Lecture 4

Educational Objectives:

What does it mean that an OCL expression is satisfiable?

What's the difference between "aggregation" and "composition"?

Please explain this class diagram with associations.

System States

Capabilities for following tasks/questions.

This Lecture:

Educational Objectives: Capabilities for following tasks/questions.

Could you please map this class diagram to a signature?

Lecture 2

Please formalise this constraint in OCL.

OCL Syntax

Could you please map this class diagram to a signature?

Lecture 1

Which annotations of an association arrow are semantically relevant?

Stereotypes – for documentation.

Please explain this OCL constraint.

Map class diagram to (extended) signature.

Lecture 3

For what purposes are class diagrams useful?

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?

Please formalise this constraint in OCL.

Object Diagrams

Could you please map this class diagram to a signature?

Lecture 7

Lecture 1

For what purposes are class diagrams useful?

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?

Please formalise this constraint in OCL.

Object Diagrams

Could you please map this class diagram to a signature?

Lecture 4
Content:

Visiblity as an extension of well-typedness.

Content:

OCL: consistency, satisfiability

Capabilities for following tasks/questions. When is an object diagram called partial? What are partial ones good for?

Object Diagrams

Give a system state satisfying this constraint?

Lecture 6
Lecture 3
Lecture 4

Educational Objectives:

What does it mean that an OCL expression is satisfiable?

What's the difference between "aggregation" and "composition"?

Please explain this class diagram with associations.

System States

Capabilities for following tasks/questions.

This Lecture:

Educational Objectives: Capabilities for following tasks/questions.

Could you please map this class diagram to a signature?

Lecture 2

Please formalise this constraint in OCL.

OCL Syntax

Could you please map this class diagram to a signature?

Lecture 1

Which annotations of an association arrow are semantically relevant?

Stereotypes – for documentation.

Please explain this OCL constraint.

Map class diagram to (extended) signature.

Lecture 3

For what purposes are class diagrams useful?

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?

Please formalise this constraint in OCL.

Object Diagrams

Could you please map this class diagram to a signature?

Lecture 7

Lecture 1

For what purposes are class diagrams useful?

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?

Please formalise this constraint in OCL.

Object Diagrams

Could you please map this class diagram to a signature?
Educational Objectives:
Please formalise this constraint in OCL.
Could you please map this signature to a class diagram?
How are system states and object diagrams related?
Btw.: where do we put OCL constraints?
Could you please map this signature to a class diagram?

Content:
Capabilities for these tasks/questions:

OCL Syntax
What is “reading direction”, “navigability”, “ownership” . . . ?
Please explain this OCL constraint.
What is a signature, an object, a system state, etc.?

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 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 6
What is the purpose of signature, object, etc. in the course?

OCL: consistency, satisfiability
Btw.: where do we put OCL constraints?
Please explain this OCL constraint.

Capabilities for following tasks/questions.
structure
Give a system state satisfying this constraint?
How are system states and object diagrams related?
Stereotypes – for documentation.
Capabilities for following tasks/questions.

Basic Object System Signatures
(Mostly) completed discussion of modelling

Educational Objectives:

Content:
What is a class diagram?

Lecture 10
Study UML syntax.
Please explain this OCL constraint.
partial vs. complete; for analysis; for documentation. . .

OCL Semantics (over system states)

Object Diagrams
Educational Objectives:
What's the purpose of a behavioural model?
What does it mean that an OCL expression is satisfiable?
Could you please map this class diagram to a signature?

Educational Objectives:
Please un-abbreviate all abbreviations in this OCL expression.

UML Core State Machines
Capabilities for these tasks/questions:
partial vs. complete; for analysis; for documentation. . .

Content:
Step, Run-to-Completion Step
Study effect on OCL.
In what sense is OCL a three-valued logic? For what purpose?

OCL Consistency and Satisfiability
completed class diagrams. . . except for associations.

Content:

Lecture 11
Content:
Object Diagrams
Basic causality model
Lecture 11
How are

OCL Semantics
Capabilities for following tasks/questions.
Associations syntax and semantics.
Lecture 8
Capabilities for these tasks/questions:
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
- Lecture 15: Hierarchical State Machines I
- Lecture 16: Hierarchical State Machines II

Last Lecture:
- step
- RTC-step, divergence
- initial states, UML model semantics (so far)
- create, destroy actions

This Lecture:
- Educational Objectives: Capabilities for following tasks/questions.
  - What makes a class diagram a good class diagram?
  - What is a legal state configuration?
  - What is a legal transition?
  - How is model semantics defined for hierarchical state machines?
- Content:
  - Legal state configurations
  - Legal transitions
  - \sigma \rightarrow \sigma'

Content
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
Could you please map this class diagram to a signature?

Forbidden scenarios

What is "reading direction", "navigability", "ownership" , . . . ?

Content:

Stereotypes – for documentation.

System configuration

Lecture 12

Forbidden scenarios

What about junction, choice, terminate, etc.?
What does "navigability", "ownership", . . . mean?

Legal transitions

Capabilities for following tasks/questions.

What is a legal transition?

Inheritance in UML: concrete syntax

Please formalise this constraint in OCL.

What is a cut, fired-set, etc.?

Capabilities for following tasks/questions.

Educational Objectives:

What's the purpose of a behavioural model?

Associations syntax and semantics.

How are passive reactive objects treated in Rhapsody's UML semantics?

Legal transitions

Rules (i) to (v) for hierarchical state machines

What does this State Machine mean? What happens if I inject this event?

System configuration cont'd

Educational Objectives:

For what purposes are class diagrams useful?

Lecture 12

Educational Objectives:

Capabilities for these tasks/questions:

A closer look onto code generation

Associations in OCL syntax.

Is this UML model consistent with that OCL constraint?

structure

What is: Signal, Event, Ether, Transformer, Step, RTC.

Forbidden scenarios

When is an object diagram called partial? What are partial ones good for?

Btw.: where do we put OCL constraints?

Btw.: where do we put OCL constraints?

Capabilities for following tasks/questions.

Content:

Firedset, Cut

What is a signature, an object, a system state, etc.?

Lecture 13

Lecture 19

What is a legal state configuration?

Rhapsody code generation

Visibility as an extension of well-typedness.

LSC syntax

Legal transitions

What makes a class diagram a good class diagram?

Missing pieces: create / destroy transformer

Can you please model the following behaviour.

What is "multiplicity"? How did we treat them semantically?

2

What makes a class diagram a good class diagram?

Capabilities for these tasks/questions:

What is visibility good for?

Please un-abbreviate all abbreviations in this OCL expression.

OCL Semantics

Lecture 15

2

Capabilities for following tasks/questions.

Map class diagram to (extended) signature cont’d.

Behavioural features

Compute the value of a given OCL constraint in a system state with links.

In what sense is OCL a three-valued logic? For what purpose?

Legal state configurations

Please un-abbreviate all abbreviations in this OCL expression.

Could you please map this class diagram to a signature?

OCL Syntax

Lecture 18

Step, Run-to-Completion Step

OCL Syntax

How is the semantics of LSCs constructed?

What's a role name? What's it good for?

Educational Objectives:

. . .

What is the idea of deferred events?

When is an object diagram called partial? What are partial ones good for?

How is enabledness of transitions defined for hierarchical state machines?

How is the semantics of LSCs constructed?
Give one example which (non-)trivially satisfies this LSC.

What are constructive and reflective descriptions of behaviour?

Lecture 14: Interactions: Live Sequence Charts

The UML Meta Model

Lecture 4: What does “navigability”, “ownership”, . . . mean?

Study effect on OCL.

Stereotypes.

Compute the value of a given OCL constraint in a system state with links.

Educational Objectives:

Inheritance: Domain Inclusion Semantics

What is a cut, fired-set, etc.?

What if things are missing?

Step, Run-to-Completion Step

What does this State Machine mean? What happens if I inject this event?

What’s the effect of inheritance on LSCs, State Machines, System States?

Ethereal event pool

How to define what happens at “system / model startup”?

Capabilities for following tasks/questions.

Content:

Can you please model the following behaviour.

structure

What does this State Machine mean? What happens if I inject this event?

OCL Consistency and Satisfiability

Lecture 19: Hierarchical state machines: the rest

What about methods?

Behavioural features

Content:

Study UML syntax.

Legal state configurations

Btw.: where do we put OCL constraints?

Cut Examples, Fired-set

How are system states and object diagrams related?

Capabilities for these tasks/questions:

OCL Semantics

Legal transitions

When is an object diagram an object diagram (wrt. what)?

Entry / exit / do actions, internal transitions

Capabilities for following tasks/questions.

Educational Objectives:

Structures

What is a legal transition?

Which annotations of an association arrow are semantically relevant?

Liskov Substitution Principle

System configuration cont’d

Content:

Can you please model the following behaviour.

structure

What does this State Machine mean? What happens if I inject this event?

OCL Semantics (over system states)

Could you please map this class diagram to a signature?

What is a class diagram?

Capabilities for following tasks/questions.

What is a signature, an object, a system state, etc.?

References
References


