Application of Models and Modelling Languages

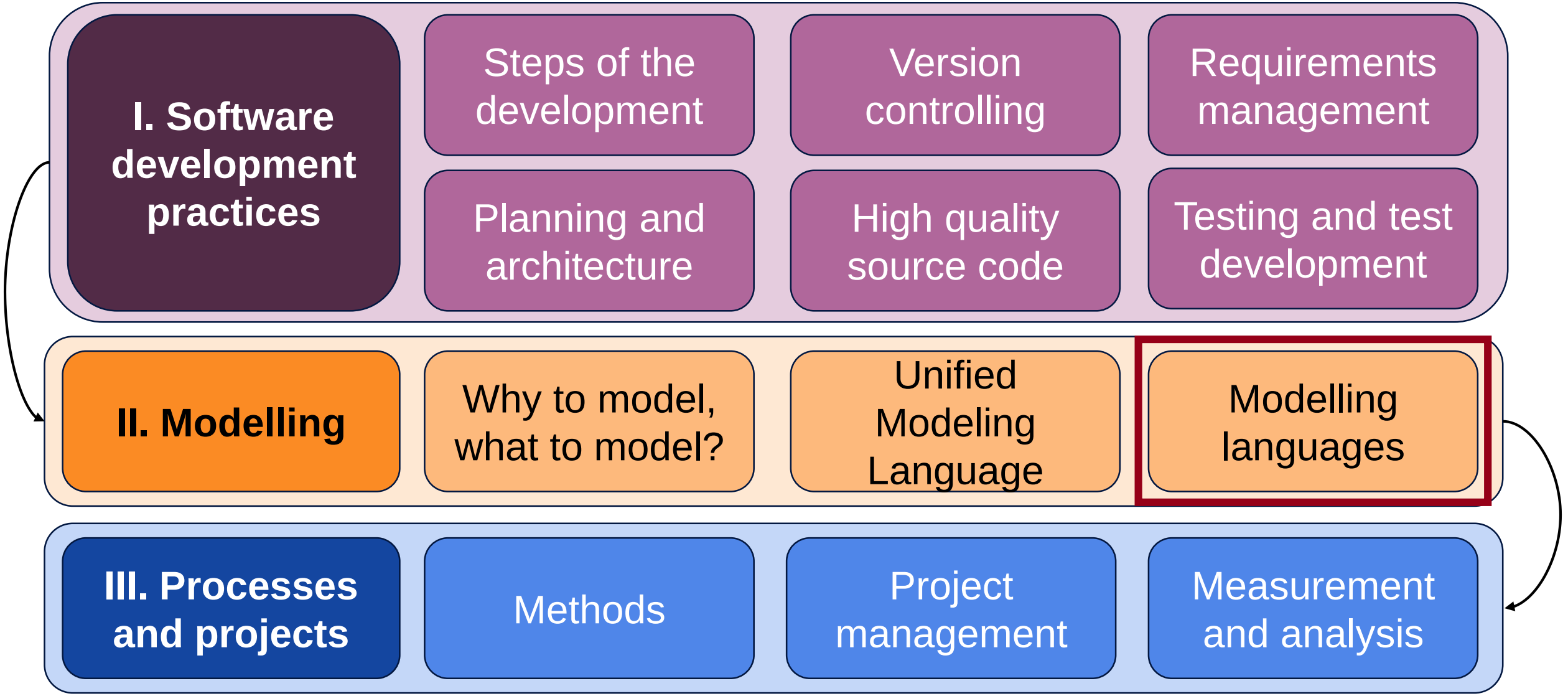
HUSZERL Gábor
huszerl@mit.bme.hu

MŰEGYETEM 1782
BME VILLAMOSMÉRNÖKI ÉS INFORMATIKAI KAR
Méréstechnika és Információs Rendszerek Tanszék
ftsrg Critical Systems Research Group

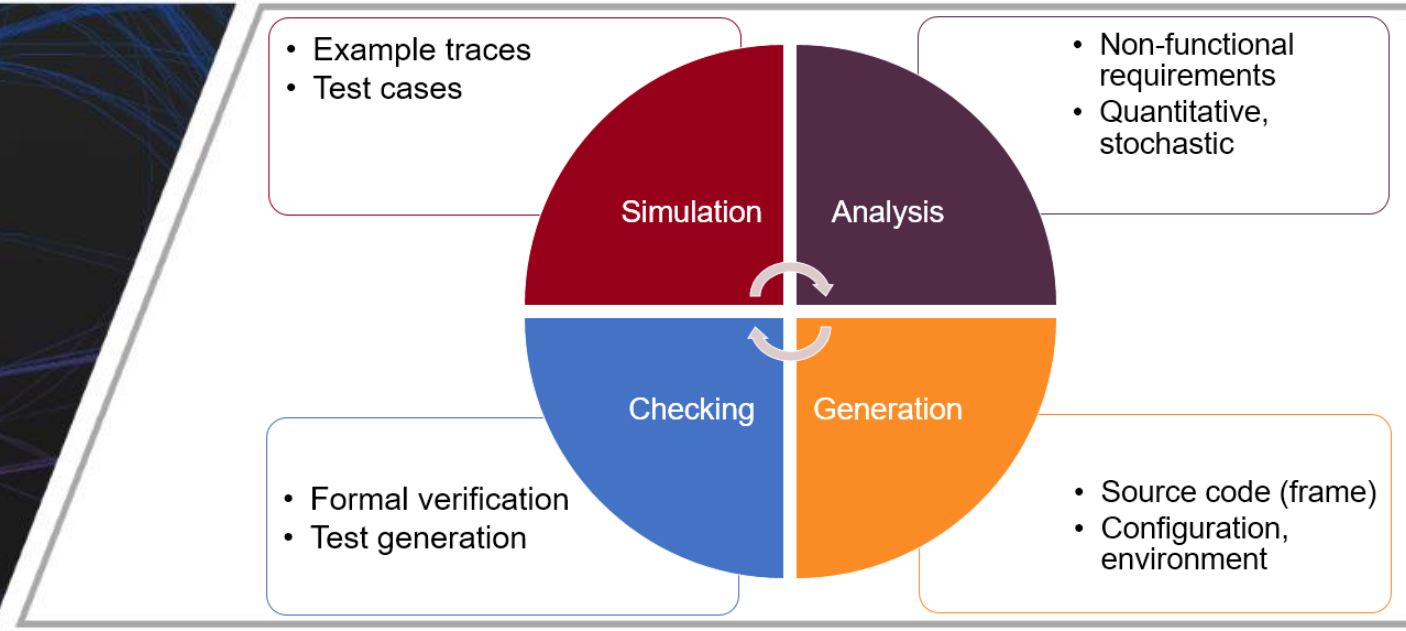
Learning Outcomes

- At the end of the lecture the students are expected to be able to
- (K1) list the purposes for which models can be used,,
- (K1) identify how modelling languages can be defined.

Further Topics of the Subject



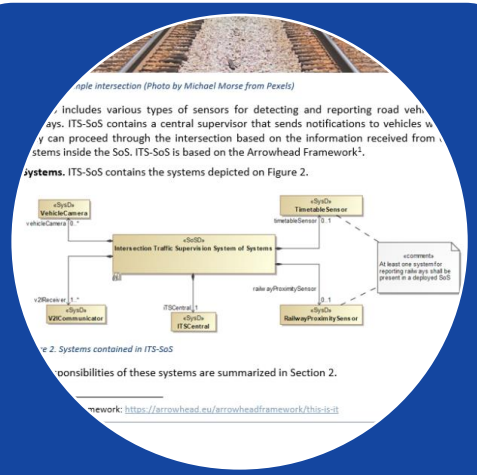
Application of Models



Not only nice pictures …

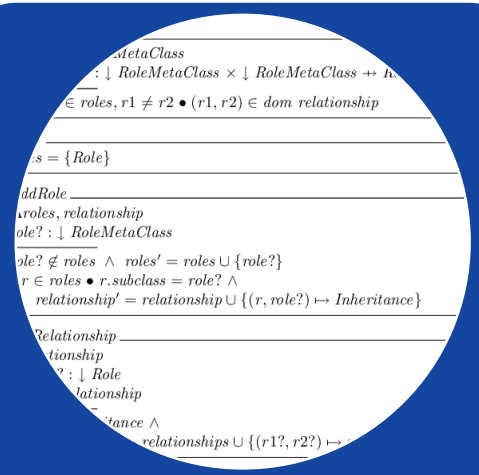
Styles of Modelling (Repetition)

Brainstorming



Illustration

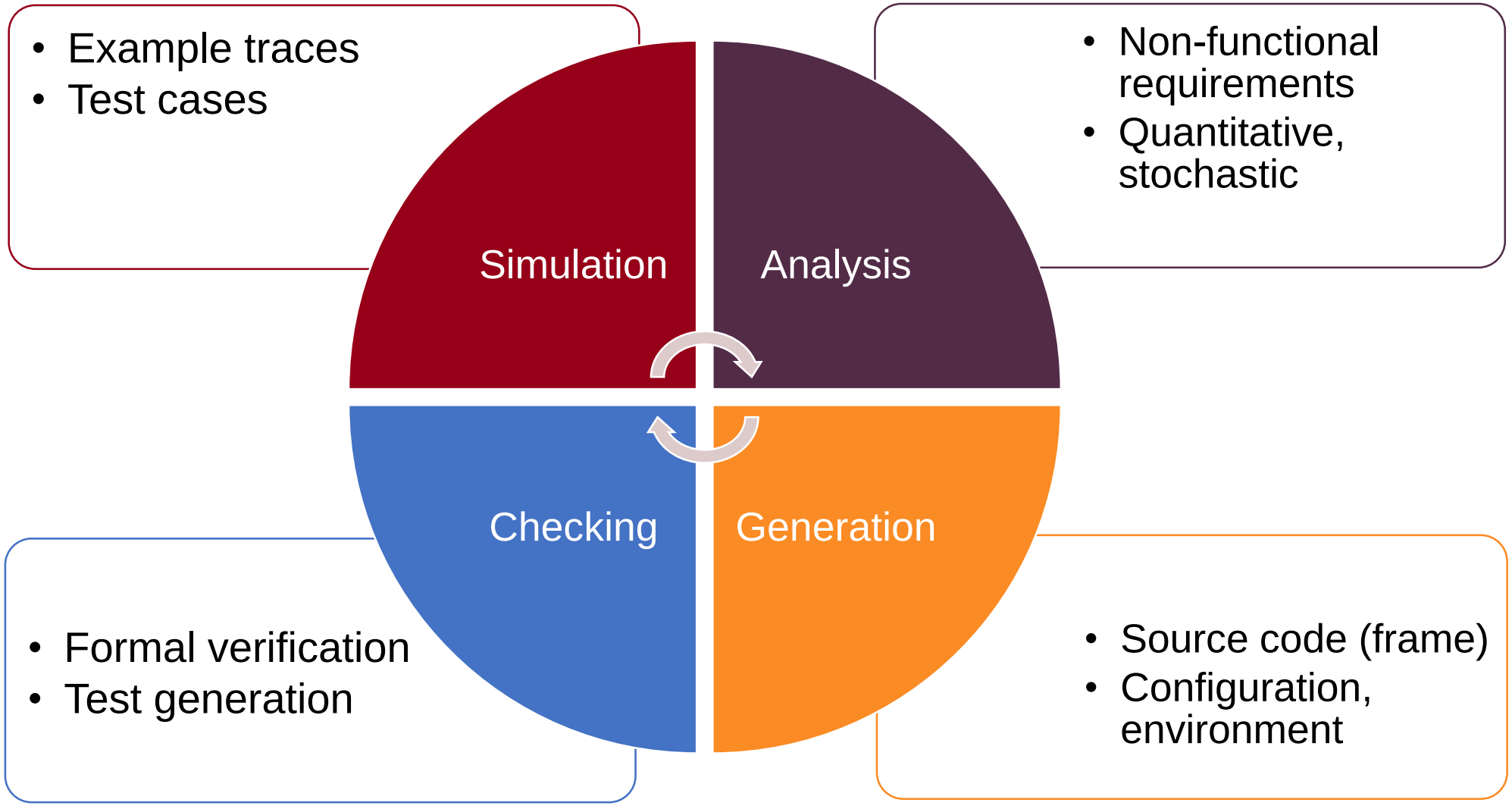
Detailed specification



Formal model

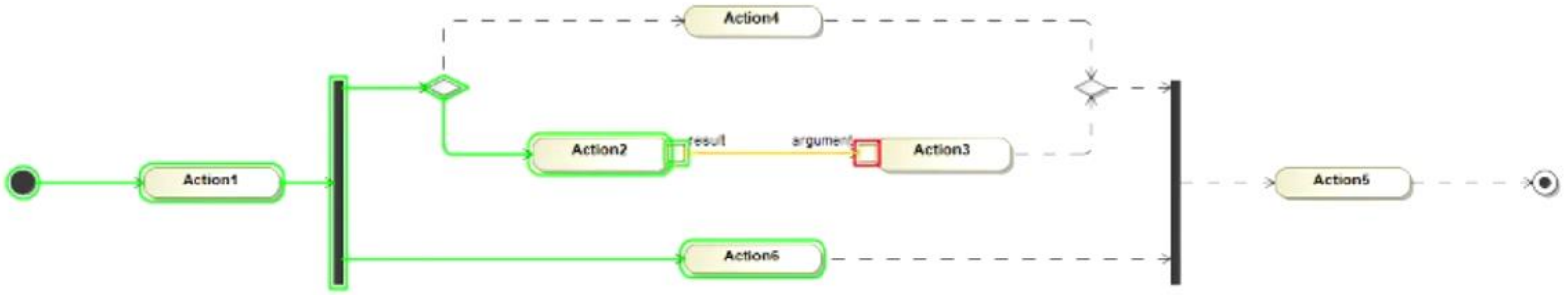
Increasingly precise models and modelling languages

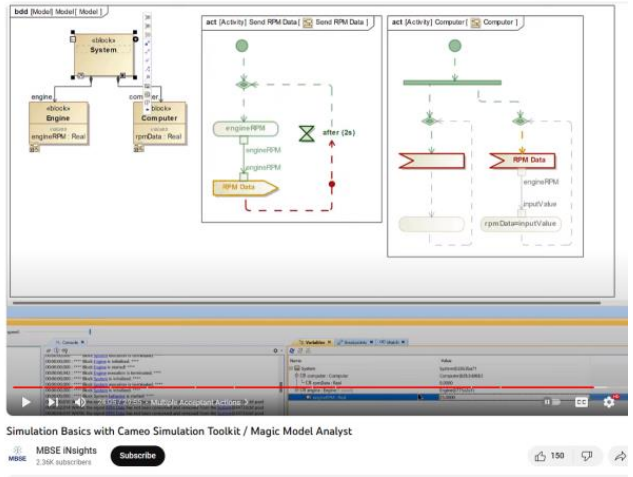
Application of Detailed Models



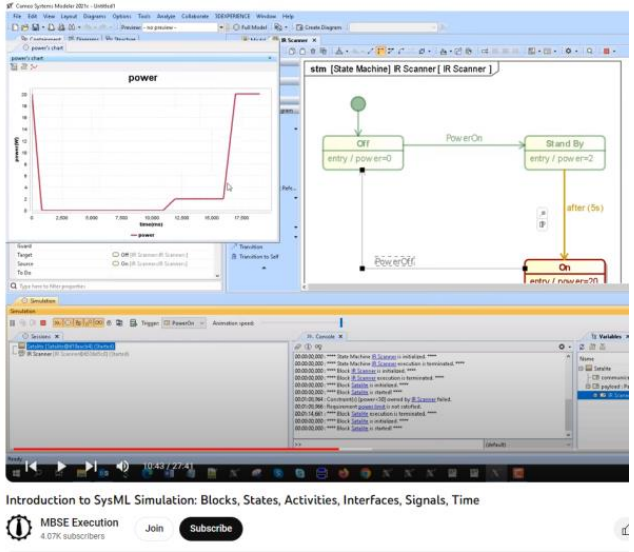
Simulation of Models

- Step-by-step execution of an activity or state machine
- Inputs manually or based on a sequence diagram
- Observing traces and variables





[Simulation Basics with Cameo Simulation Toolkit](#) (YouTube)



[Introduction to SysML Simulation](#) (YouTube)

See „System Modelling” course

Analysis Options

- Evaluation: timing, availability, dependability, …
- Statistic methods: analysing a large number of executions

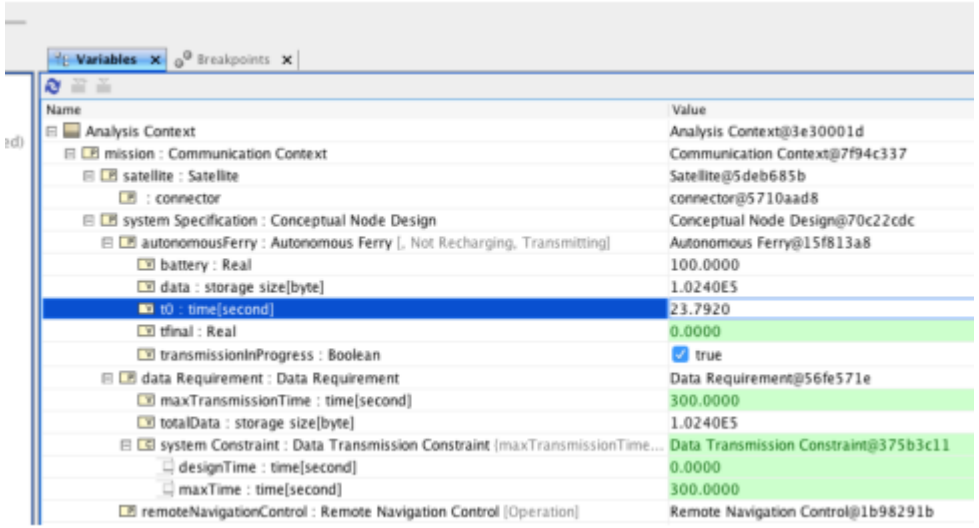
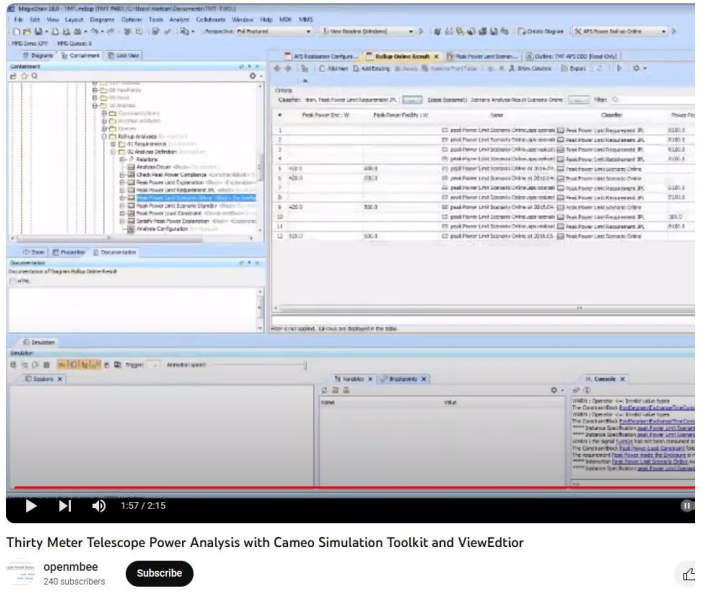


Table 7. Analysis Results

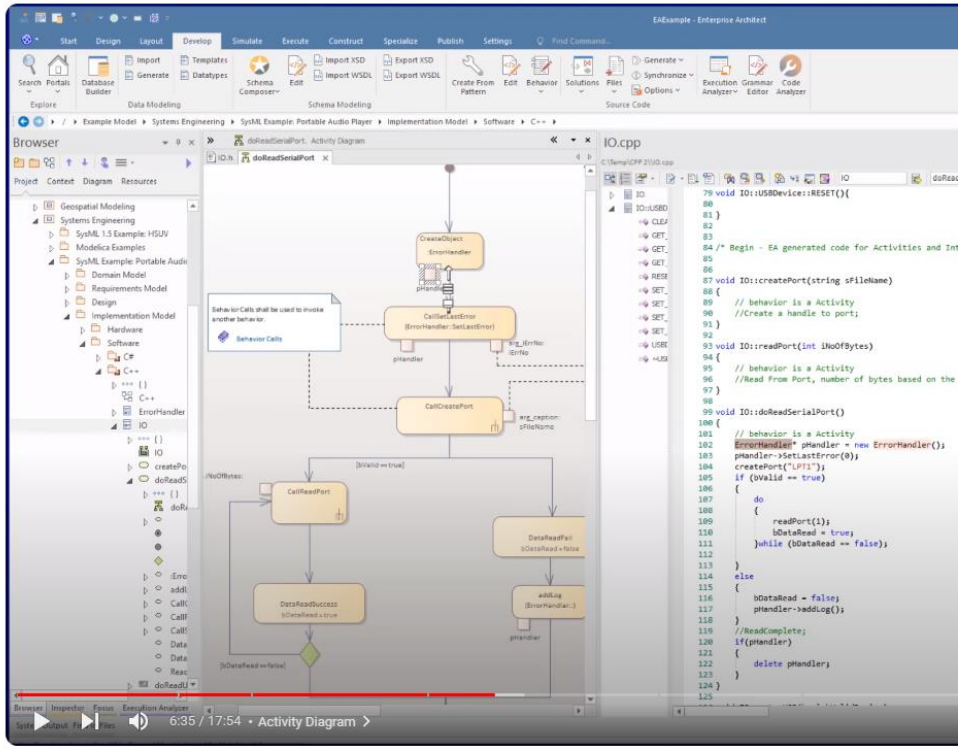
Name	Classifier	mission.conceptualDesign.autonomousFerry.tfinal : time[second]	mission.conceptualDesign.data Requirement.maxTransmissionTime : time[second]
analysis Context at 2017.08.02 08.31	Analysis Context	1620.0	300.0
analysis Context at 2017.08.02 08.36	Analysis Context	1622.5	2000.0



[Thirty Meter Telescope Power Analysis with Cameo Simulation Toolkit and ViewEditor](#) (YouTube)

Generation: From Source Code to Test Environment

- Generation of Code (framework)
  - Model → Code
  - Code → Model
- Generation of detailed behaviour
- Generation of configuration
  - Test environment
  - Application platform: cloud, mobile, embedded, …

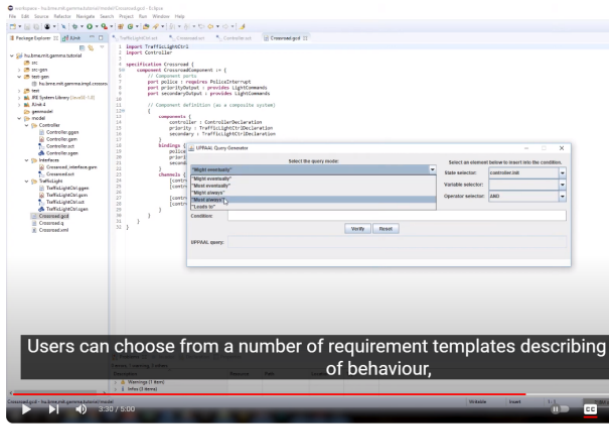


Generating code from Behavioral Models
Enterprise Architect (YouTube)

See „Automatized Software Development” course

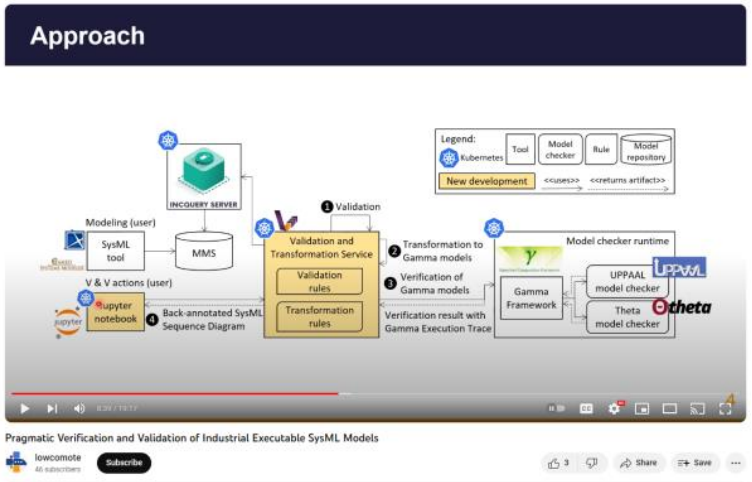
Checking

- Generation of test cases
  - Model-based Testing (MBT)
  - Coverage of model elements



[The Gamma Statechart Composition Framework]

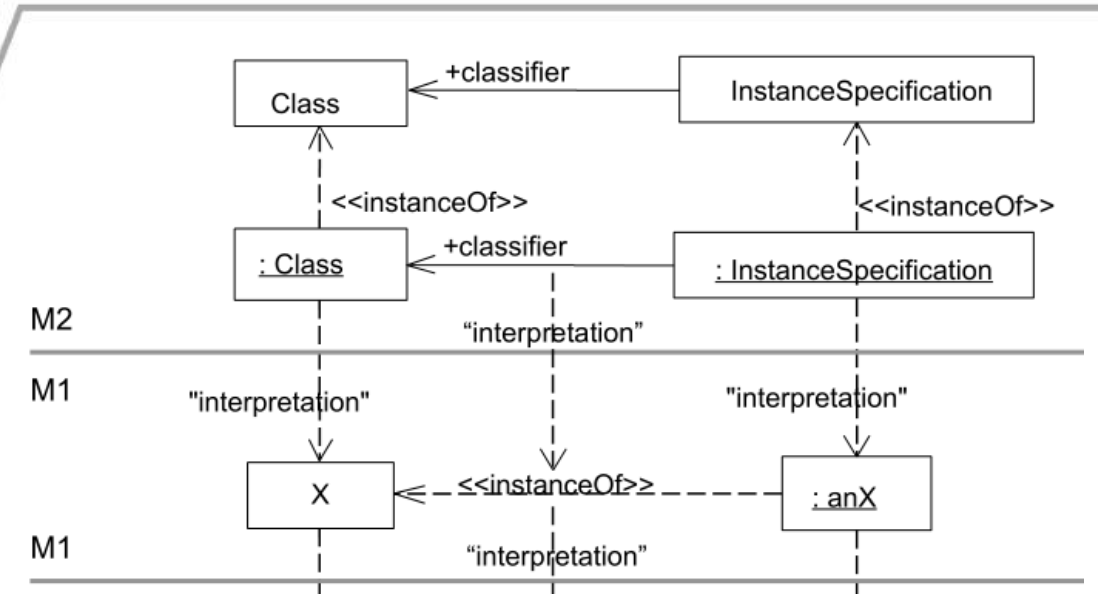
- Formal verification
  - Exhaustive checking of the model behaviour
  - Model checking: state space exploration
  - (Runnable) Counterexample or proof



[Pragmatic Verification and Validation of Industrial Executable SysML Models]

INCQUERY JKU JOHANNES KEPLER UNIVERSITÄT LINZ NASA Jet Propulsion Laboratory California Institute of Technology

Modelling Languages



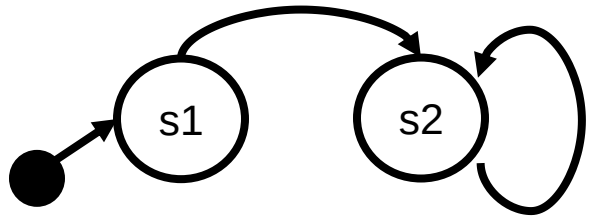
Syntax and semantics

Modelling Language (Rep.)

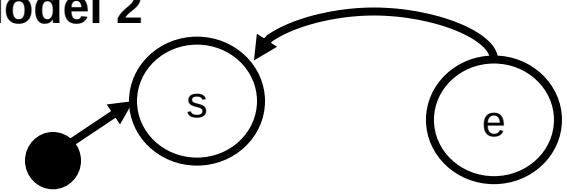
- Defines what are the well-formed models

- Required to define a language:
  - **Syntax**: what rules can be used to build a valid model?
  - **Semantics**: what does the model mean? What domain can we map it to?

Model 1



Model 2



Modelling language

Syntax
- Transition (arrow) connects source and target states (circle).
- The source and target states can be the same.

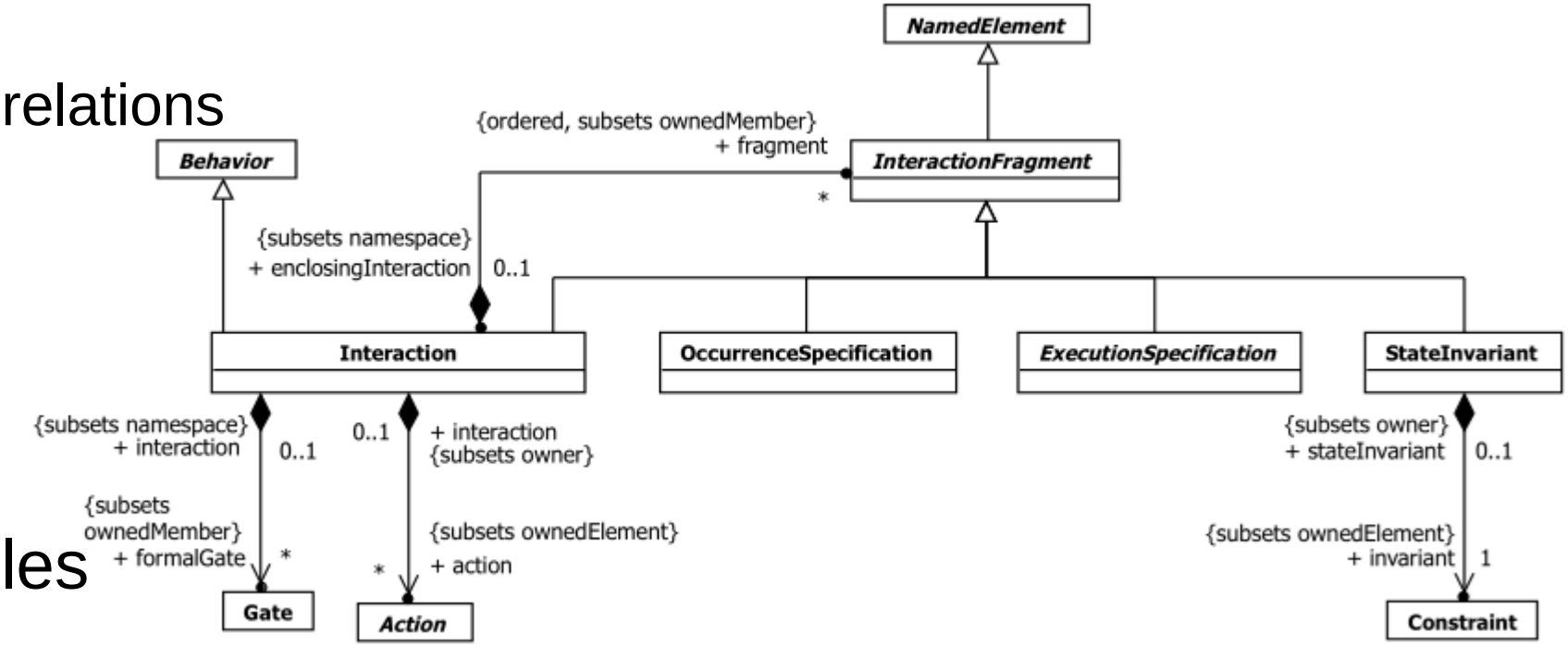
Semantics
- Transitions represent state changes in response to events
- Model's meaning is a sequence of state changes

Defining Modelling Languages

- Abstract syntax
- Concrete syntax
- Well-formedness rules
- Semantics

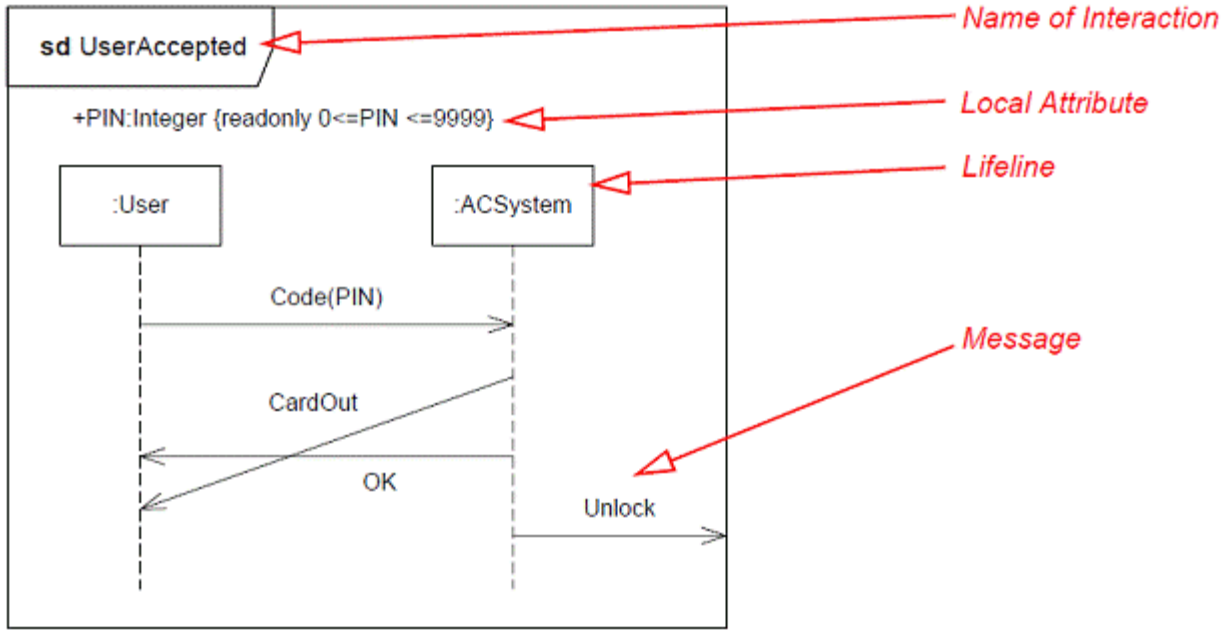
Defining Modelling Languages

- **Abstract syntax**
  - Elements and their relations
  - “Metamodel”
  - Grammatical rules
- Concrete syntax
- Well-formedness rules
- Semantics



Defining Modelling Languages

- Abstract syntax
- **Concrete syntax**
  - Textual or graphical
  - How the elements of the syntax are represented
  - Different ones may exist!
- Well-formedness rules
- Semantics



Defining Modelling Languages

- Abstract syntax
- Concrete syntax
- **Well-formedness rules**
  - Further constraints
  - E.g. in Object Constraint Language (OCL)
    - Declarative language
    - “What” and not “how”
- Semantics

17.12.3.7 Constraints

- break
  If the interactionOperator is break, the corresponding InteractionOperand must cover all Lifelines covered by the enclosing InteractionFragment.

```
inv: interactionOperator=InteractionOperatorKind::break  implies
enclosingInteraction.oclAsType(InteractionFragment)->asSet()->union(
   enclosingOperand.oclAsType(InteractionFragment)->asSet()).covered->asSet() =
self.covered->asSet()
```

- consider_and_ignore
  The interaction operators 'consider' and 'ignore' can only be used for the ConsiderIgnoreFragment subtype of CombinedFragment.

```
inv: ((interactionOperator = InteractionOperatorKind::consider) or (interactionOperator =
InteractionOperatorKind::ignore)) implies oclIsKindOf(ConsiderIgnoreFragment)
```

See „Declarative Programming” course

Defining Modelling Languages

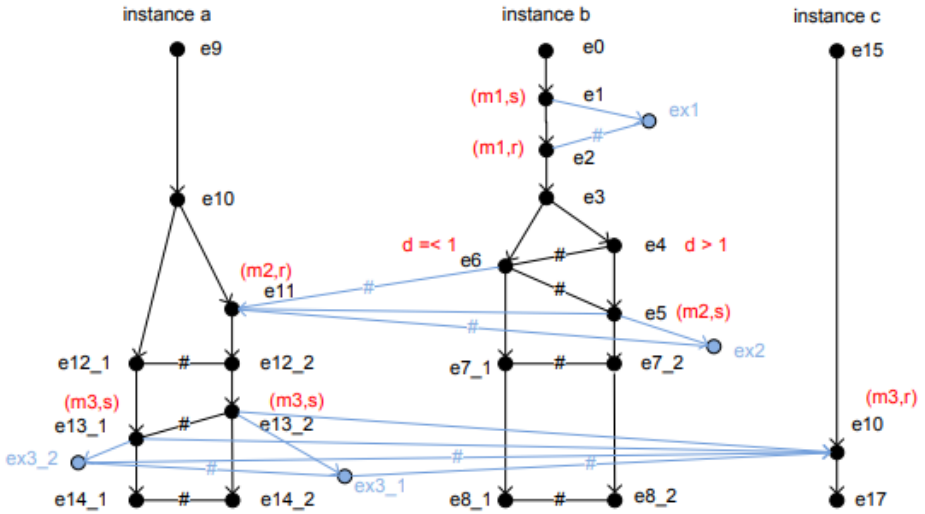
- Abstract syntax
- Concrete syntax
- Well-formedness rules
- **Semantics**
  - What a model means?
  - Particularly difficult to specify precisely
  - It is usually tried to map to mathematical concepts

Declarative semantics

The alt construct defines potential traces. The semantics is the union of the trace sets for both positive and negative:

$$[\![d_1 \; alt \; d_2]\!] \stackrel{\text{def}}{=} \{(p_1 \cup p_2, n_1 \cup n_2) \mid (p_1, n_1) \in [\![d_1]\!] \wedge (p_2, n_2) \in [\![d_2]\!]\}$$

Operational semantics



Meta-Levels: Model and Metamodel

- Metamodel: Model of a modelling language

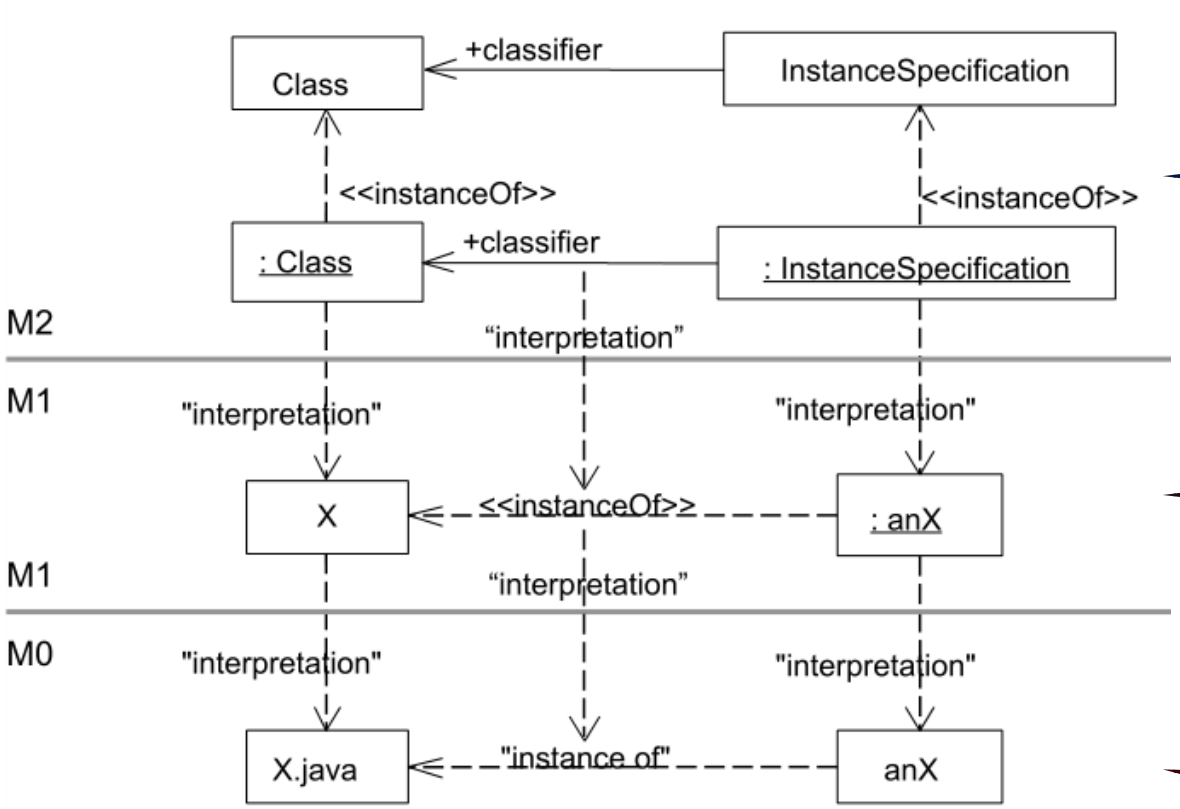


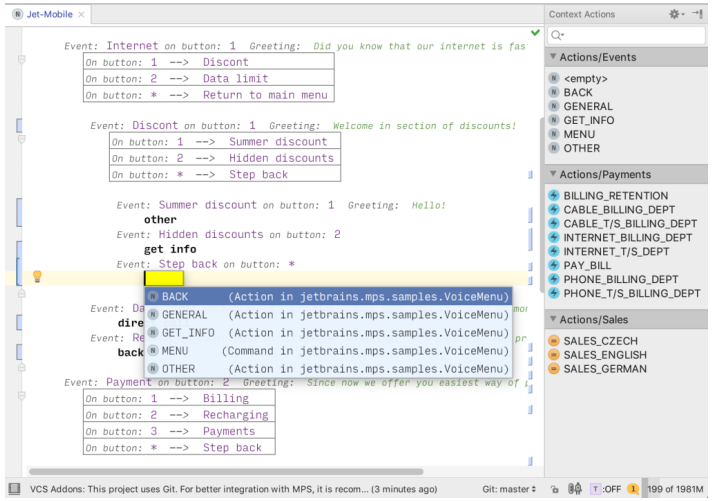
Figure 6.5 - Interpretation across Meta-Layers M2 and M1

Metamodel: Model of the UML language (Class, Property…)

Model: Model described in UML (specific classes)

Semantic domain: what is described by the model („reality”, Java program, …)

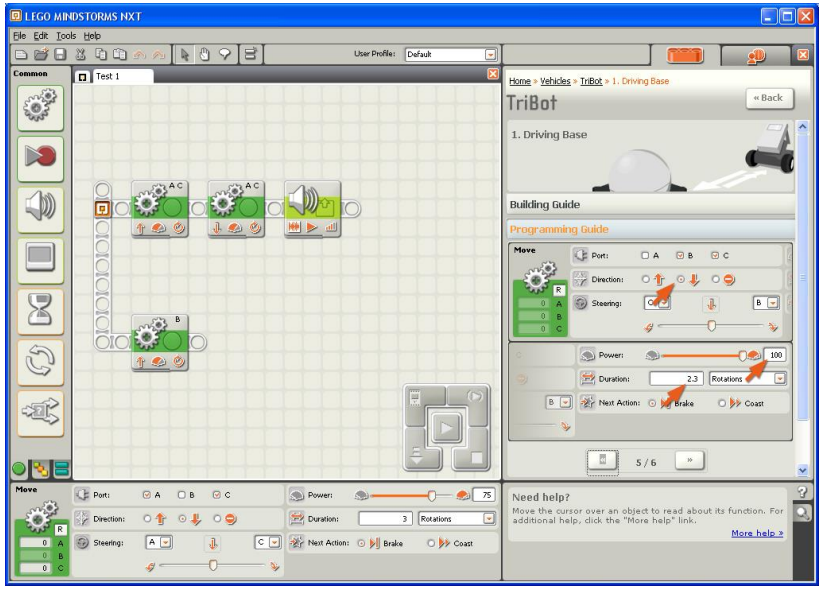
Domain Specific Languages



(DSL)

General vs. Domain Specific Languages

- UML: a general modelling language
  - Advantage: it can be used in many different domains
  - Disadvantage: complicated language
- Targeted languages for specific problems?



Refinery

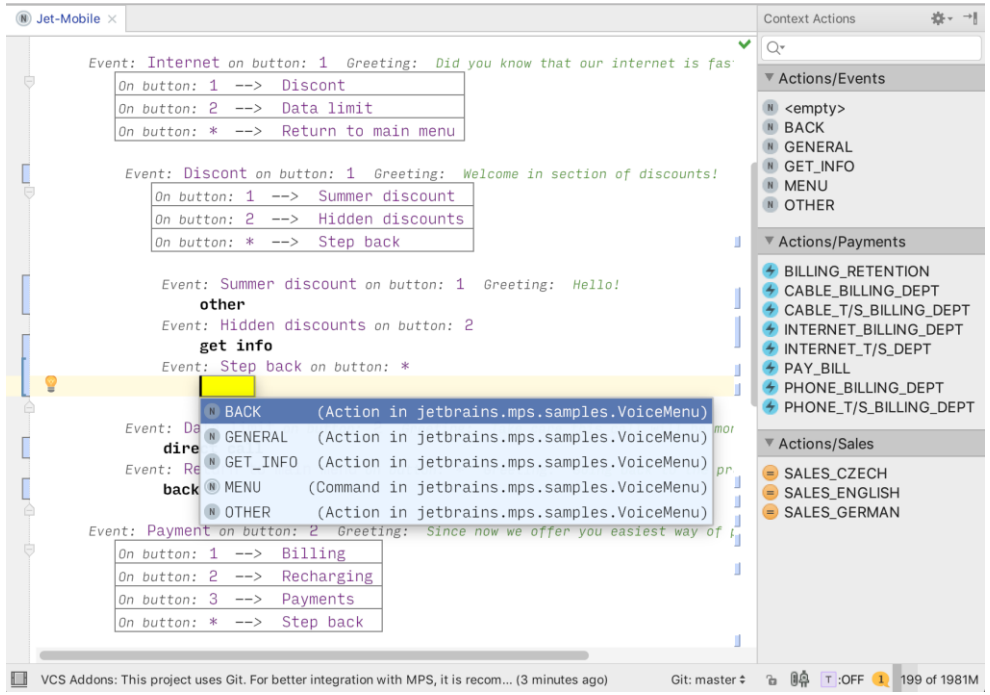
```
Family(family).
members(family, anne).
members(family, bob).
members(family, ciri).
children(anne, ciri).
?children(bob, ciri).
default children(ciri, *): false.
taxStatus(anne, ADULT).
age(bob): 21..35.
age(ciri): 10.

scope Family = 1, Person += 5..10.
```

Domain Specific Languages

> **„Domain-Specific Language** (DSL) is a computer language that's targeted to a particular kind of problem, rather than a general purpose language that's aimed at any kind of software problem.” [Martin Flower]

- „Language engineering”
- Can be textual or graphical
- Specialised development tools
  - E.g. JetBrains MPS, Langium, Racket

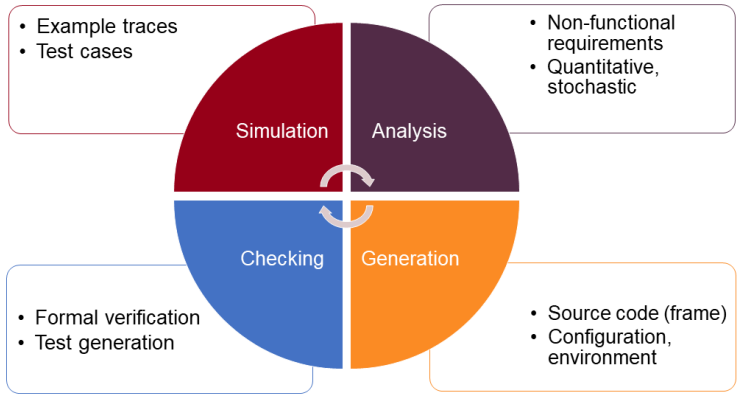


See „Model-based Software Development” (Msc) course

Summary

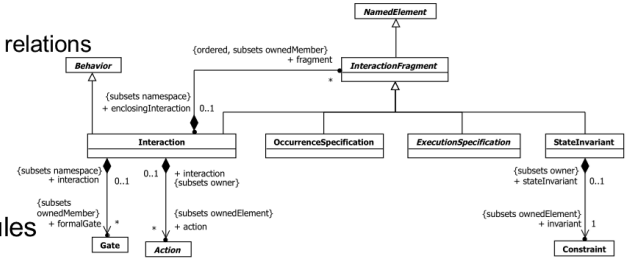
Summary

Application of Detailed Models



Defining Modelling Languages

- **Abstract syntax**
  - Elements and their relations
  - "Metamodel"
  - Grammatical rules
- Concrete syntax
- Well-formedness rules
- Semantics



Meta-Levels: Model and Metamodel

- Metamodel: Model of a modelling language

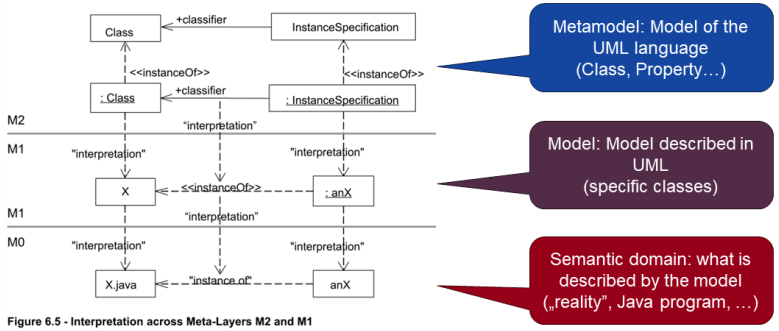
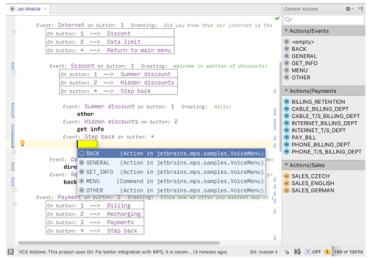


Figure 6.5 - Interpretation across Meta-Layers M2 and M1

Domain Specific Languages

> **„Domain-Specific Language** (DSL) is a computer language that's targeted to a particular kind of problem, rather than a general purpose language that's aimed at any kind of software problem." [Martin Flower]

- „Language engineering"
- Can be textual or graphical
- Specialised development tools
  - E.g. JetBrains MPS, Langium, Racket



See „Model-based Software Development" (Msc) course