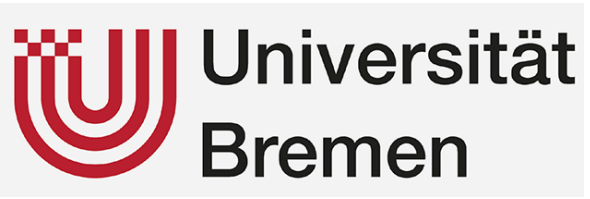
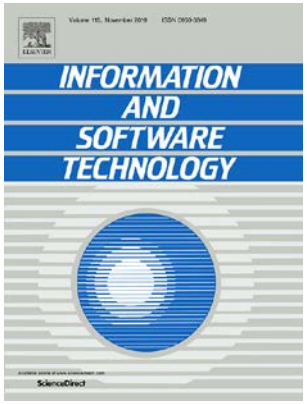


Native Support for UML and OCL Primitive Datatypes Enriched with Uncertainty in USE

Víctor Ortiz, Loli Burgueño, Antonio Vallecillo, and Martin Gogolla

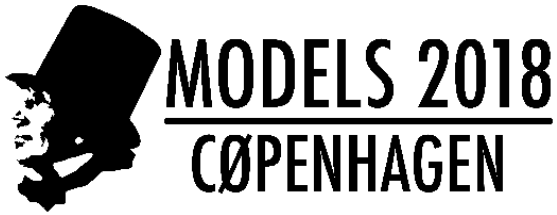
OCL Workshop
Munich, September 16, 2019

Previous works



L. Burgueño, T. Mayerhofer, M. Wimmer, A. Vallecillo:
Specifying quantities in software models.
Information & Software Technology 113: 82-97 (2019)

M. F. Bertoa, N. Moreno, G. Barquero, L. Burgueño, J. Troya, A. Vallecillo:
Expressing Measurement Uncertainty in OCL/UML Datatypes.
ECMFA 2018: 46-62



L. Burgueño, M. F. Bertoa, N. Moreno, A. Vallecillo:
Expressing Confidence in Models and in Model Transformation Elements.
MoDELS 2018: 57-66

Uncertainty

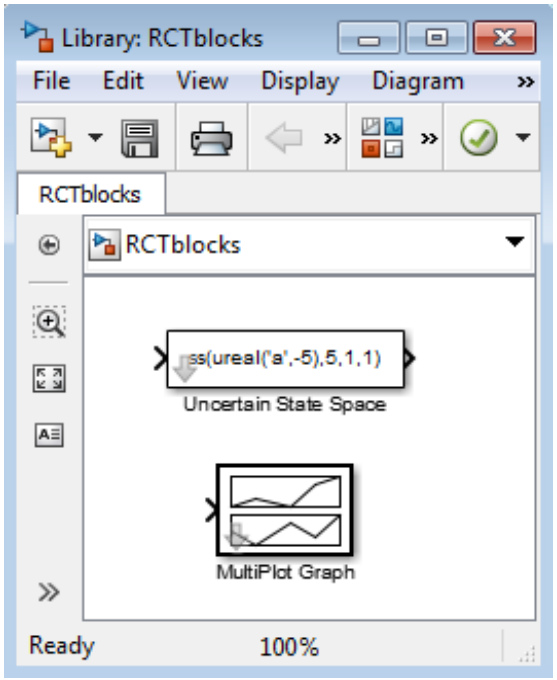
Uncertainty: Quality or state that involves imperfect and/or unknown information

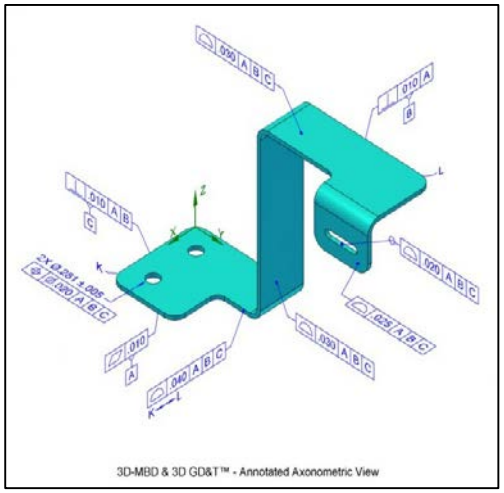
- It applies to: predictions of future events,
  estimations,
  physical measurements, or
  properties of a system, its elements or its environment
- due to:
  - Underspecification of the problem or solution domains
  - Lack of knowledge of the system, its environment, or its underlying physics
  - Lack of precision in measurements
  - Imperfect, incorrect, or missing information
  - Numerical approximations
  - Values and parameters indeterminacy
  - Different interpretations of the same evidences by separate parties

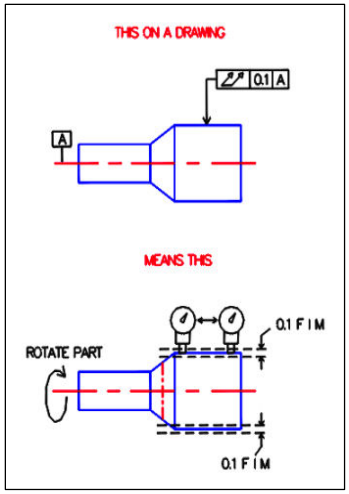
"There is nothing certain, but the uncertain" (*proverb*)

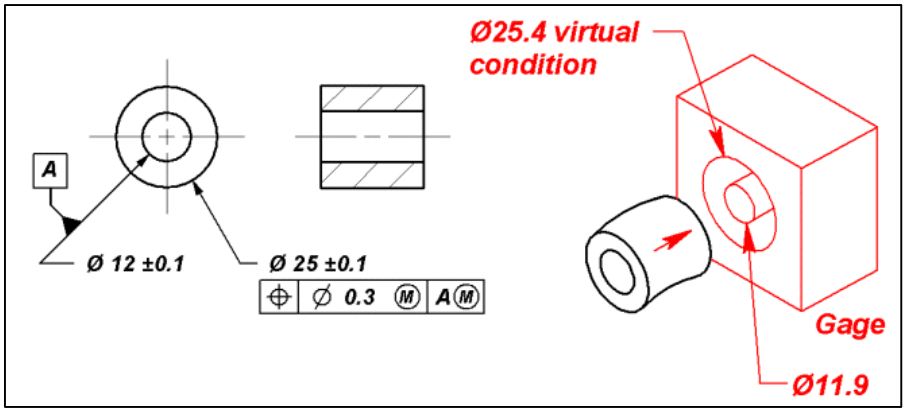
Measurement uncertainty

- Engineers naturally think about ***uncertainty*** associated with ***measured values***
- Uncertainty is explicitly defined in their models and considered in model-based simulations
- Precise notations permit representing and operating with uncertain values and confidences









Measurement uncertainty

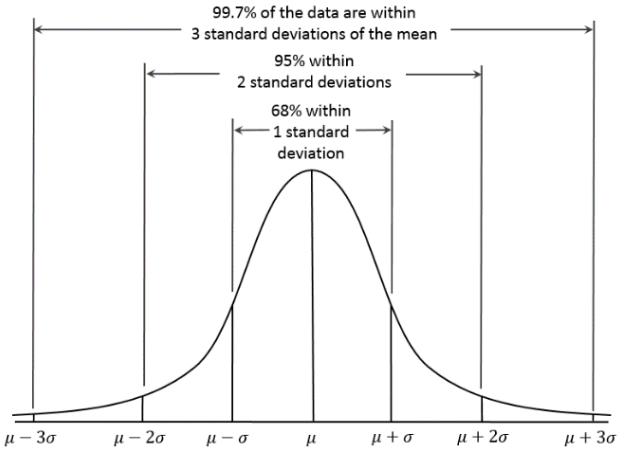
- **Measurement** uncertainty: A kind of *aleatory* uncertainty that refers to a set of possible states or outcomes of a measurement

- Normally expressed by a parameter, associated with the result of a measurement x, that characterizes the dispersion of the values that could reasonably be attributed to the measurand: the standard deviation u of the possible variation of the values of x

- Representation: $\boldsymbol{x \pm u}$ or (x, u)

- *Examples:*
  - Normal distribution: (x, σ) with mean x, and and standard deviation σ
  - Interval $[a, b]$: Uniform distribution is assumed

$$(x, u) \text{ with } x = \frac{a+b}{2}, u = \frac{(b-a)}{2\sqrt{3}}$$

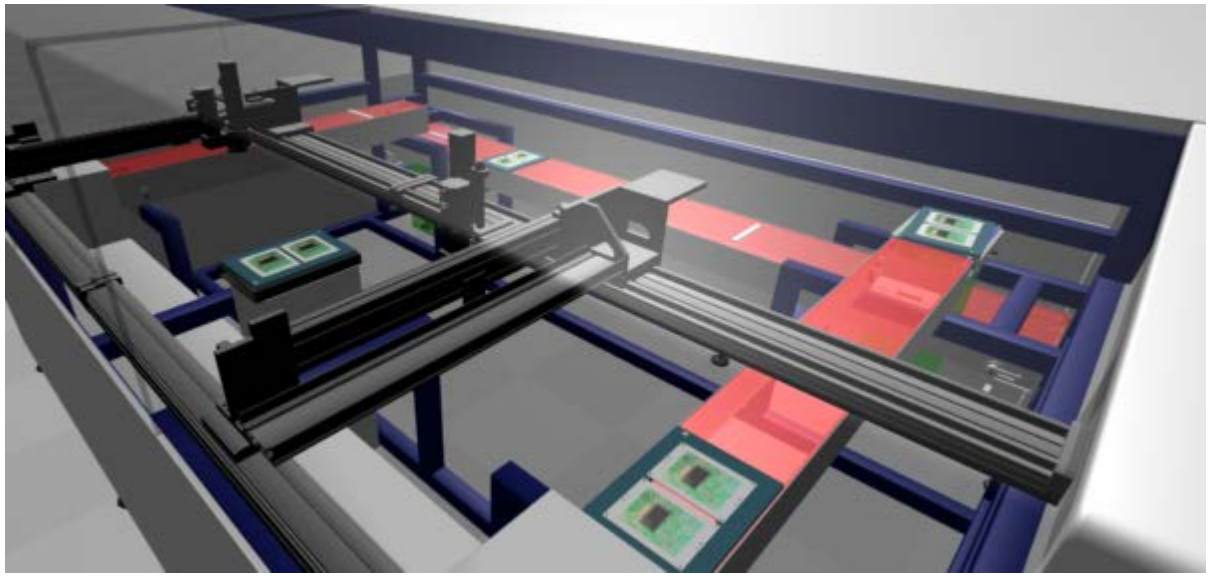


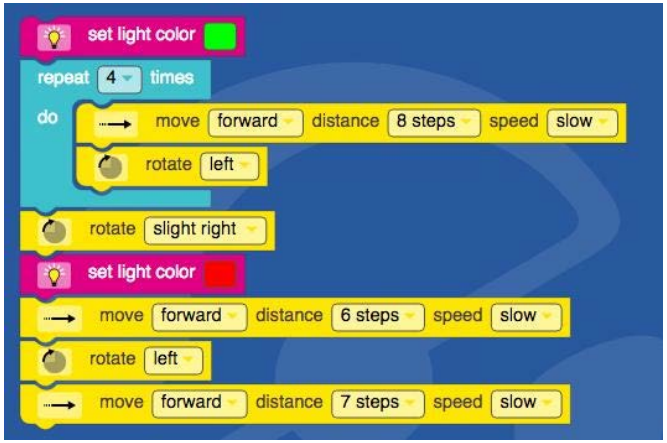
JCGM 100:2008. Evaluation of measurement data – Guide to the expression of uncertainty in measurement (GUM). http://www.bipm.org/utils/common/documents/jcgm/JCGM_100_2008_E.pdf

However, the situation is not the same in software models ☹

RoundObject
+posX : Real +posY : Real +posZ : Real +weight : Real +width : Real +height : Real
+move(dX : Real, dy : Real, dz : Real) +catch() +drop() +fitsIn(other : RoundObject)

Useful applications in software simulation

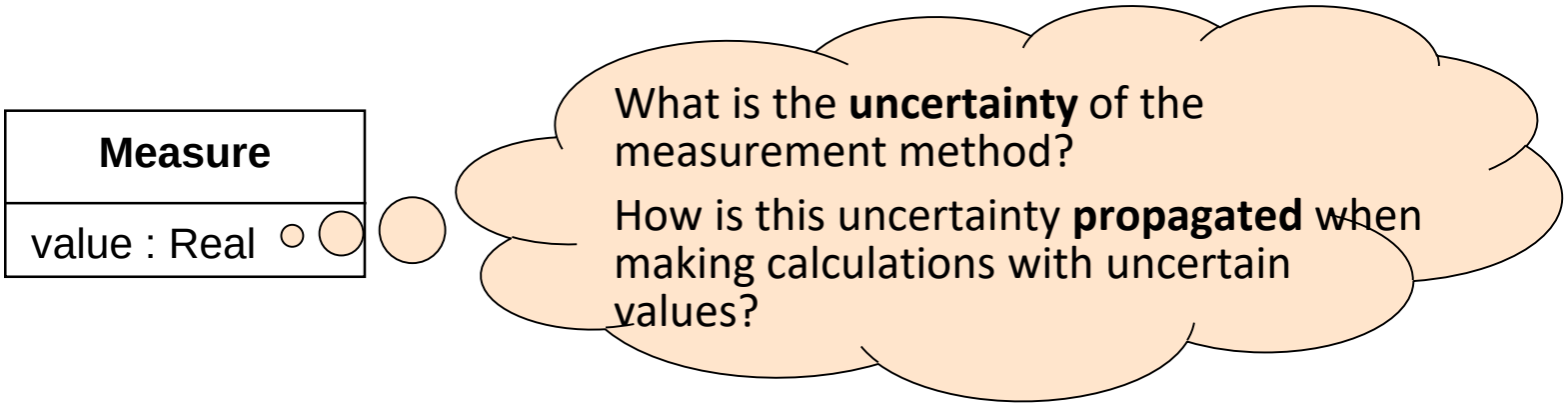




Motivation

Uncertainty in Software Engineering

- Very limited support for representing uncertainty in software models
- No support for considering such properties in model-based simulations
- Not part of their type systems!

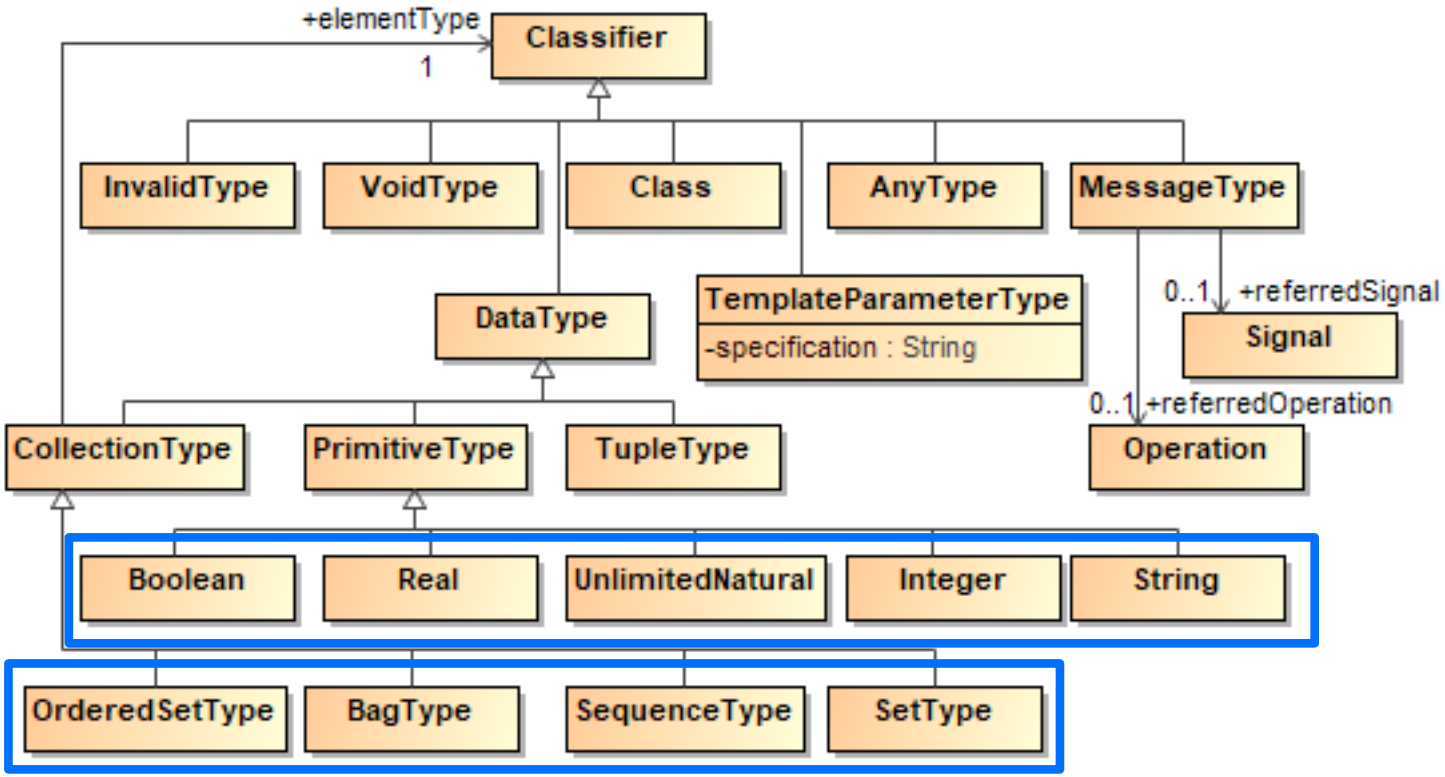


Some problems with Measurement Uncertainty

- Computations with uncertain values have to respect the *propagation of uncertainty* (uncertainty analysis)
  - In general this is a complex problem, which cannot be manually managed
- *Comparison of uncertain values* is no longer a Boolean property!
  - How to compare 17.7 ± 0.2 with 17.8 ± 0.2?
- Other primitive datatypes are also affected by uncertainty
  - Strings (OCR)
  - Enumerations
  - Collections

In our previous work…

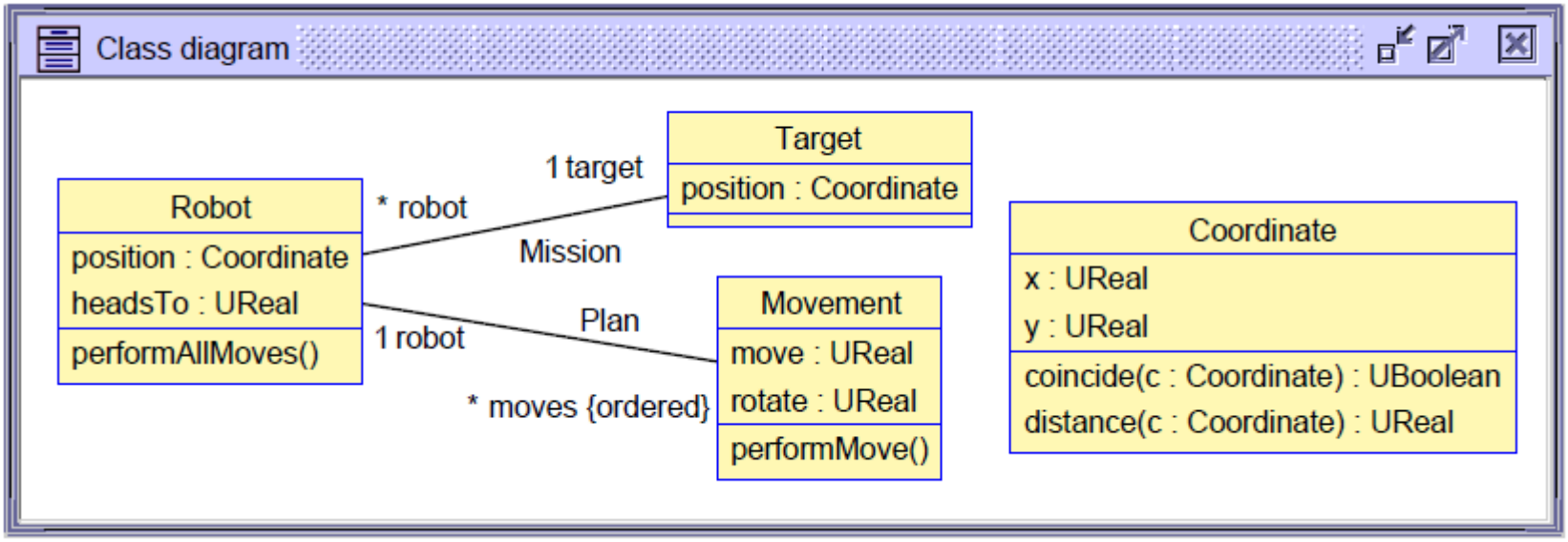
- Extension of the OCL/UML
  - Primitive types
  - Collections



- Java Library – behavior of uncertain datatypes

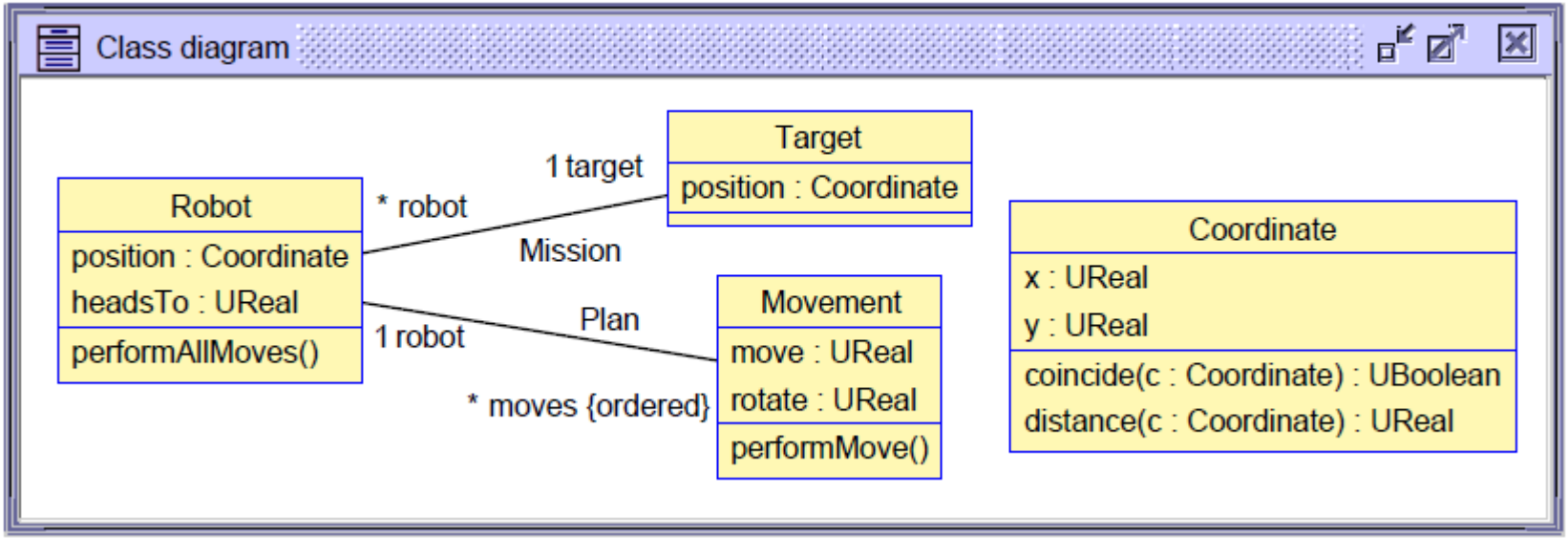
Our case study

- **Ozobot robot**
  - able to move in the direction its head points to
  - They accept two type of commands: (1) to rotate the head, and (2) to move forward
  - The mission determines the target position the robot is supposed to reach with the plan



Our case study

- We are interested in analyzing whether the sequence of movements defined in its fulfills the mission i.e., it reaches the target position

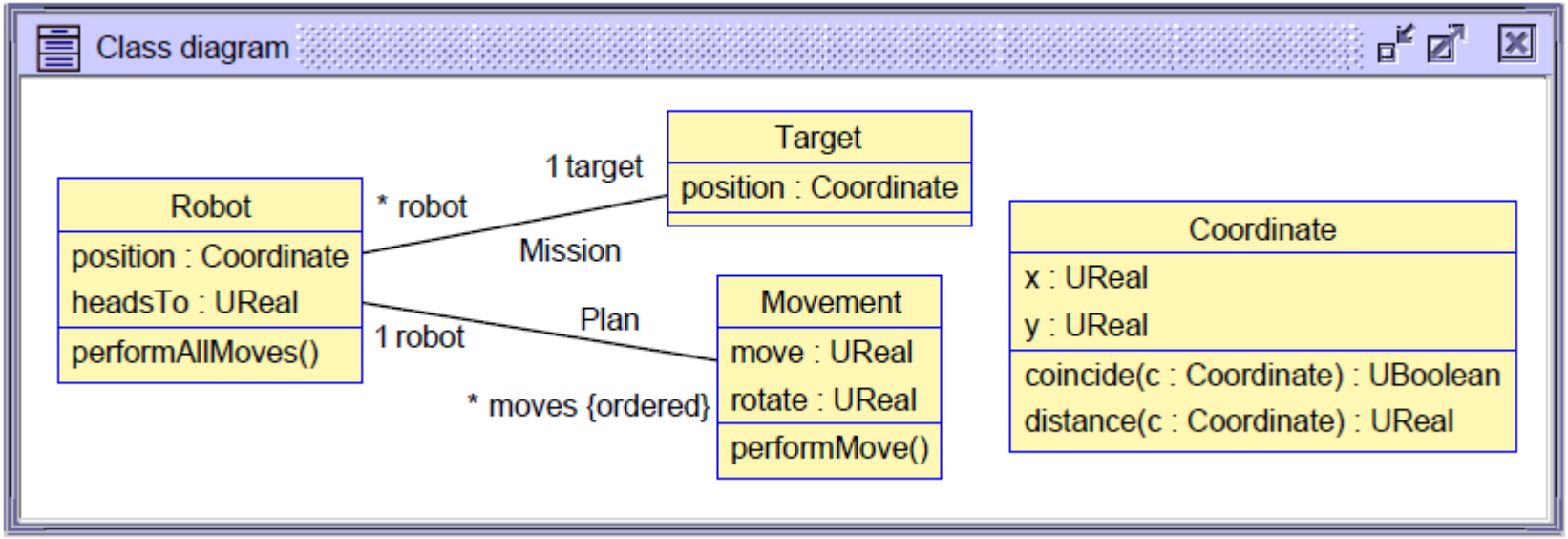


- Invariants:

```
context Coordinate::coincide(c:Coordinate):UBoolean =
    self.x = c.x and self.y = c.y
context Coordinate::distance(c:Coordinate):UReal =
    ((self.x-c.x)*(self.x-c.x) + (self.y-c.y)*(self.y-c.y)).sqrt()
```

Our case study

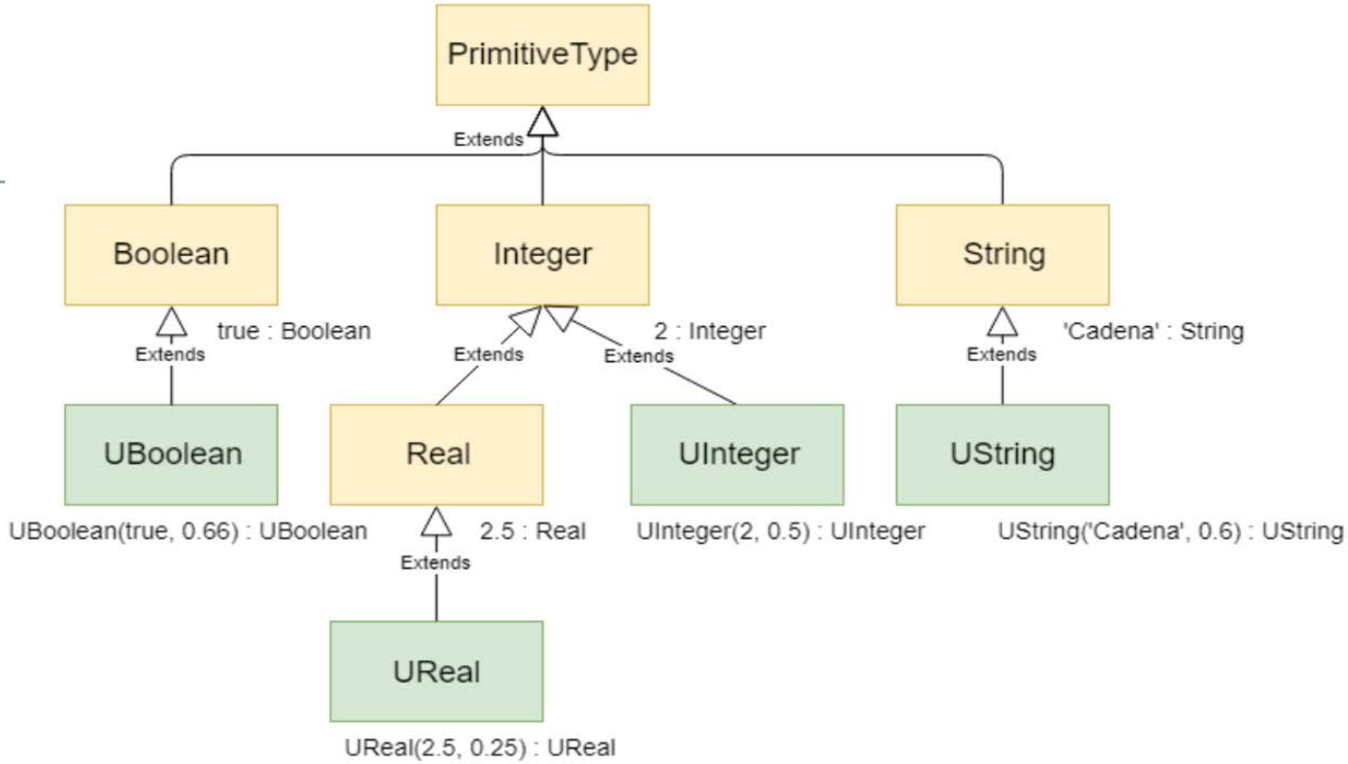
- We are interested in analyzing whether the sequence of movements defined in its fulfills the mission i.e., it reaches the target position



Uncertainty is important!

It should be <u>easily</u> captured and propagated

Our approach




```
C:\Windows\system32\cmd.exe
USE version 5.0.1, Copyright (C) 1999-2018 University of Bremen
use> ? UReal(2.5, 0.25) + 2.3 - 1 + UInteger(2, 0.005)
-> UReal(5.8, 0.250049995) : UReal

use> ? UReal(2.5, 0.25).sin()
-> UReal(0.5984721441, 0.2002859039) : UReal

use> ? o.lenght * o.height
-> UReal(50.0, 5.153882032) : UReal
```

Scope and goal

Test our Java library

Extend USE

- Grammars
- Java source code

Test USE (old + new functionality)

Extension of USE

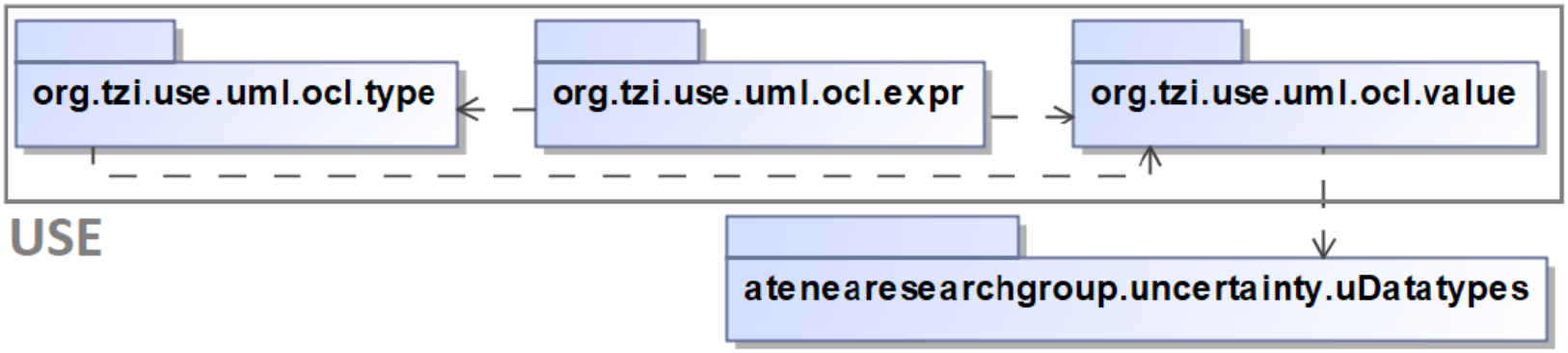
- Despite its graphical interface, the specification of models in USE is textual
- We had to modify its OCL grammar:
  - file org.tzi.use.parser.base.OCLBase.gpart

```
literal returns [ASTExpression n] :
      t='true'   { $n = new ASTBooleanLiteral(true); }
    | f='false'  { $n = new ASTBooleanLiteral(false); }
    | i=INT      { $n = new ASTIntegerLiteral($i); }
    | r=REAL     { $n = new ASTRealLiteral($r); }
    | s=STRING   { $n = new ASTStringLiteral($s); }
    | 'UString' LPAREN usve=additiveExpression COMMA
        usue=additiveExpression RPAREN
        { $n = new ASTUStringLiteral($usve.n,$usue.n); }
    | 'UReal' LPAREN urve=additiveExpression COMMA
        urue=additiveExpression RPAREN
        { $n = new ASTURealLiteral($urve.n, $urue.n); }
    | 'UBoolean' LPAREN ubve=conditionalImpliesExpression COMMA
        ubpe=additiveExpression RPAREN
        { $n = new ASTUBooleanLiteral($ubve.n, $ubpe.n); }
    | 'UInteger' LPAREN uive=additiveExpression COMMA
        uiue=additiveExpression RPAREN
        { $n = new ASTUIntegerLiteral($uive.n, $uiue.n); }
    ... ;
uncertaintyType returns [ASTType n] :
    name=('UReal'|'UInteger'|'UBoolean'|'UString')
    { $n = new ASTSimpleType($name); } ;
```

- Using the ANTLR tools, the Java lexer, parser, tokens and listeners were automatically generated

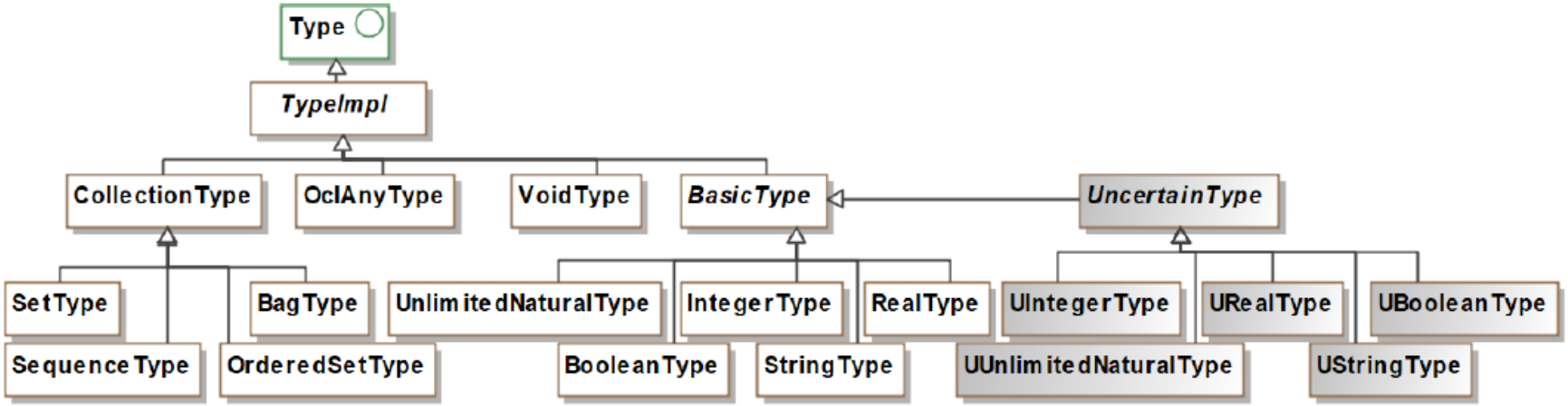
Extension of USE

- The implementation of datatypes in USE is done in a modular way
- It distinguishes between values and expressions
  - Both have a type



Extension of USE

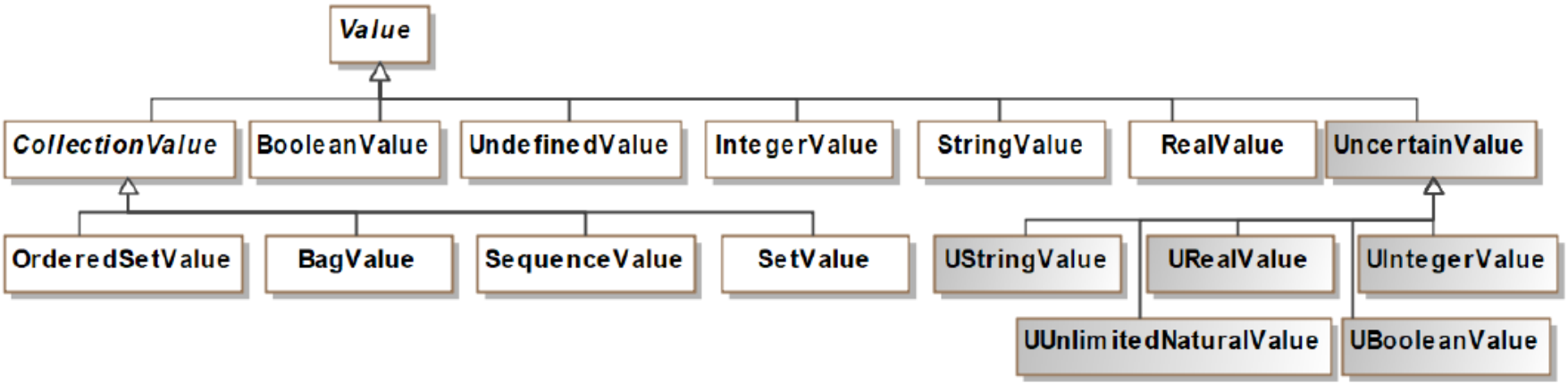
- org.tzi.use.uml.ocl.type



- Each class contains:
  - Constructor
  - methods such as isTypeOfUReal, isKindOfUReal, isKindOfNumber and isKindOfOclAny

Extension of USE

- org.tzi.use.uml.ocl.value



- each class applies Adapter design pattern
- and acts as a wrapper for the classes in the library

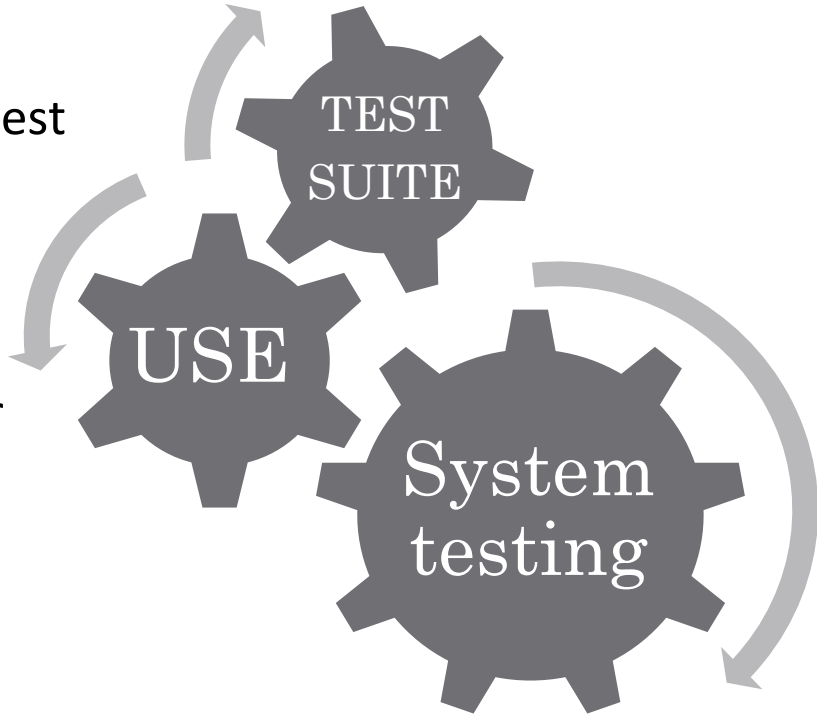
Extension of USE

- org.tzi.use.uml.ocl.expr

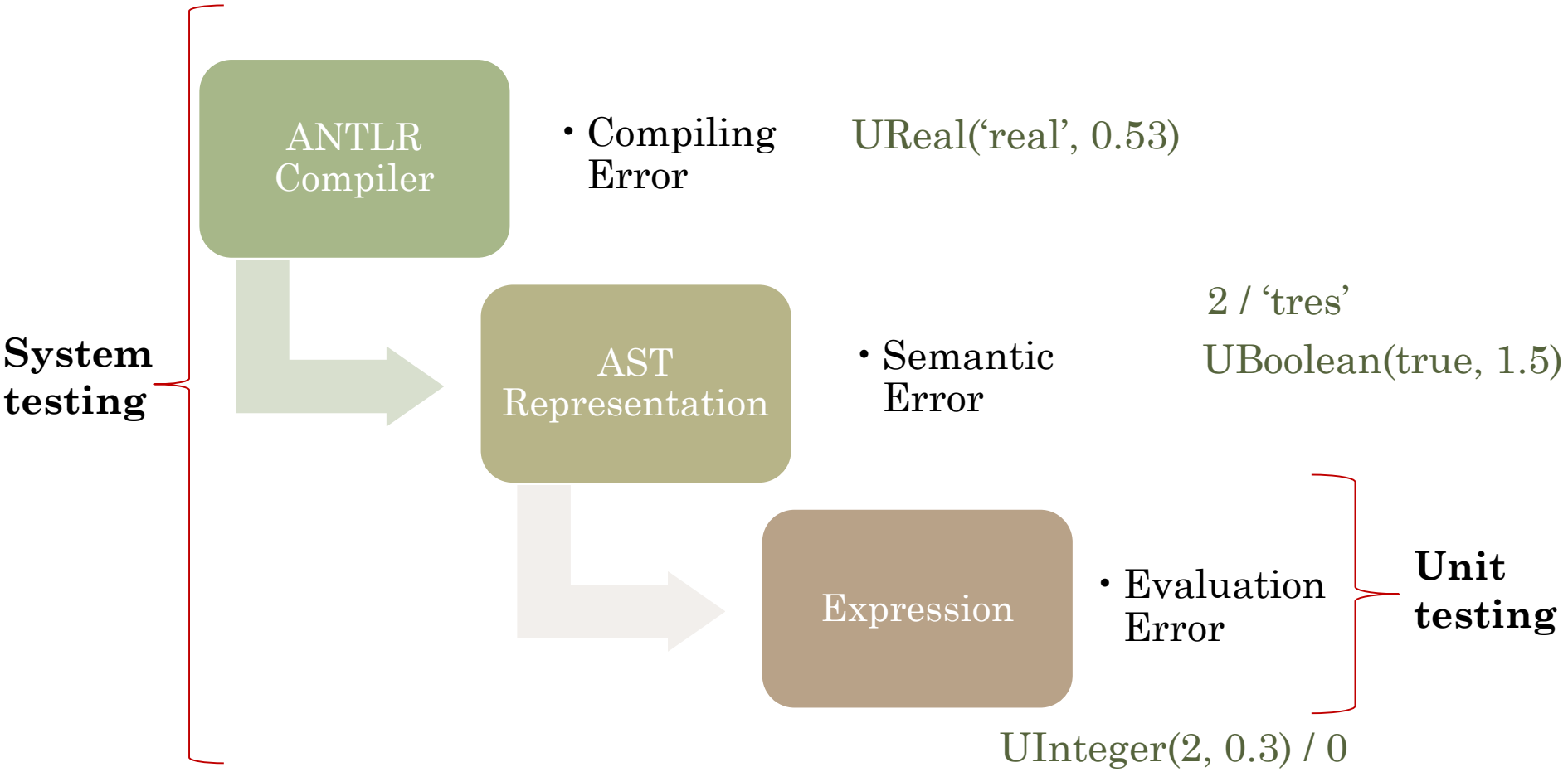
- Once the types and values of the new datatypes were created,
  - make them available for their use inside OCL expressions
  - make their operations available
    - overload the existing operators such as “+”

Extension of USE

- We followed a test-driven methodology when extending use. Thus,
  - we extensively used the testing facilities that USE provides for unit and system testing
- We included our tests under the folder src/test and executed them in batch using ANT
- We check both its grammar and its behavior



Extension of USE - Testing



Conclusions and Future work

- Extension of the tool USE to enable the application of native uncertain types for capturing measurement uncertainty
- We have shown how we structured and implemented the extension

Future

- Check and improve (if needed) the efficiency of the execution of operations
- Extend the evaluation browser with aspects of uncertainty
- Check how far other OCL evaluators can be extended in this way
  - and study the effort required to do so

Native Support for UML and OCL Primitive Datatypes Enriched with Uncertainty in USE

Víctor Ortiz, Loli Burgueño, Antonio Vallecillo, and Martin Gogolla

OCL Workshop
Munich, September 16, 2019