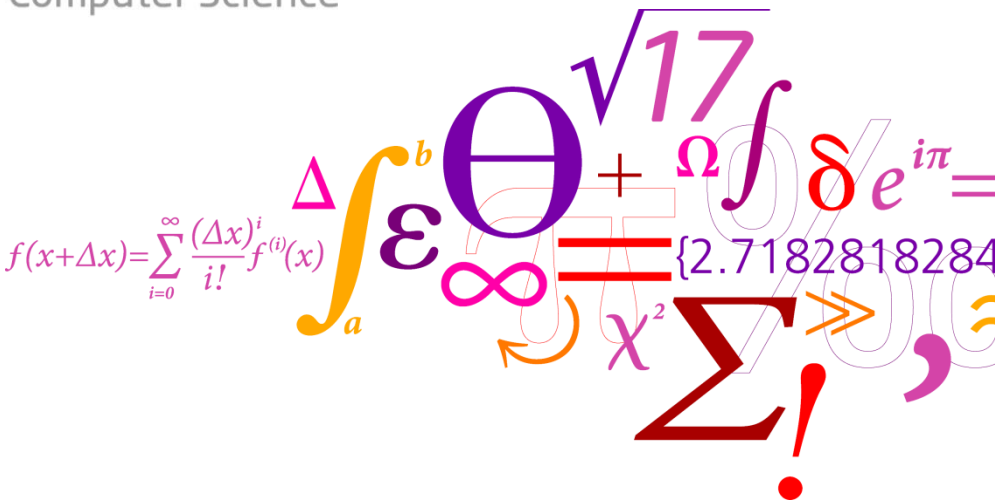
Model-based Software Engineering
(02341, spring 2017)

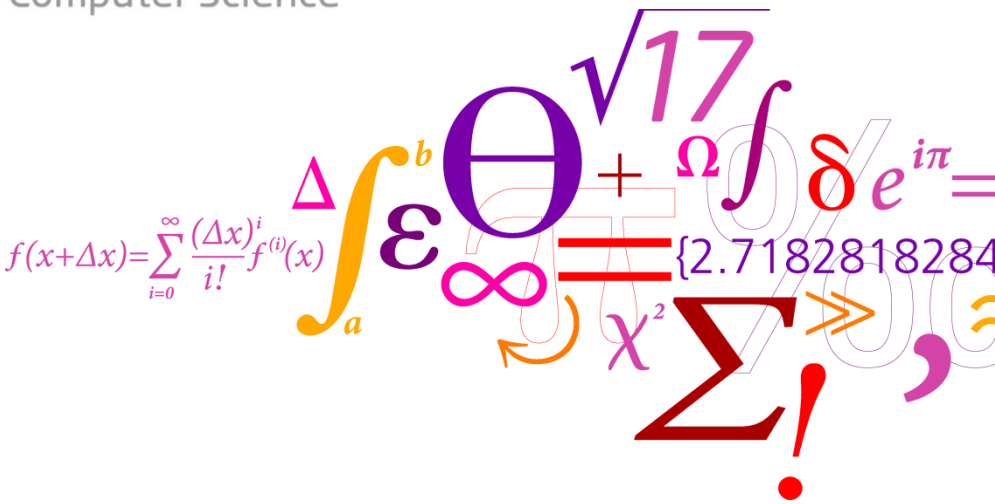
Ekkart Kindler

DTU Compute
Department of Applied Mathematics and Computer Science



VII. Constraints and Validation

DTU Compute
Department of Applied Mathematics and Computer Science



1. Motivation

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

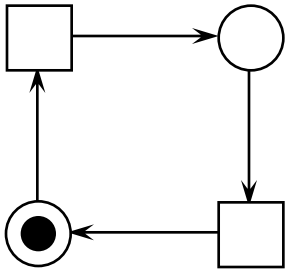
It is difficult, inadequate, or sometimes even impossible to express the exact relationship between some domain concepts in pure UML class diagrams

→ We need to express the precise nature of these relationships in a different way

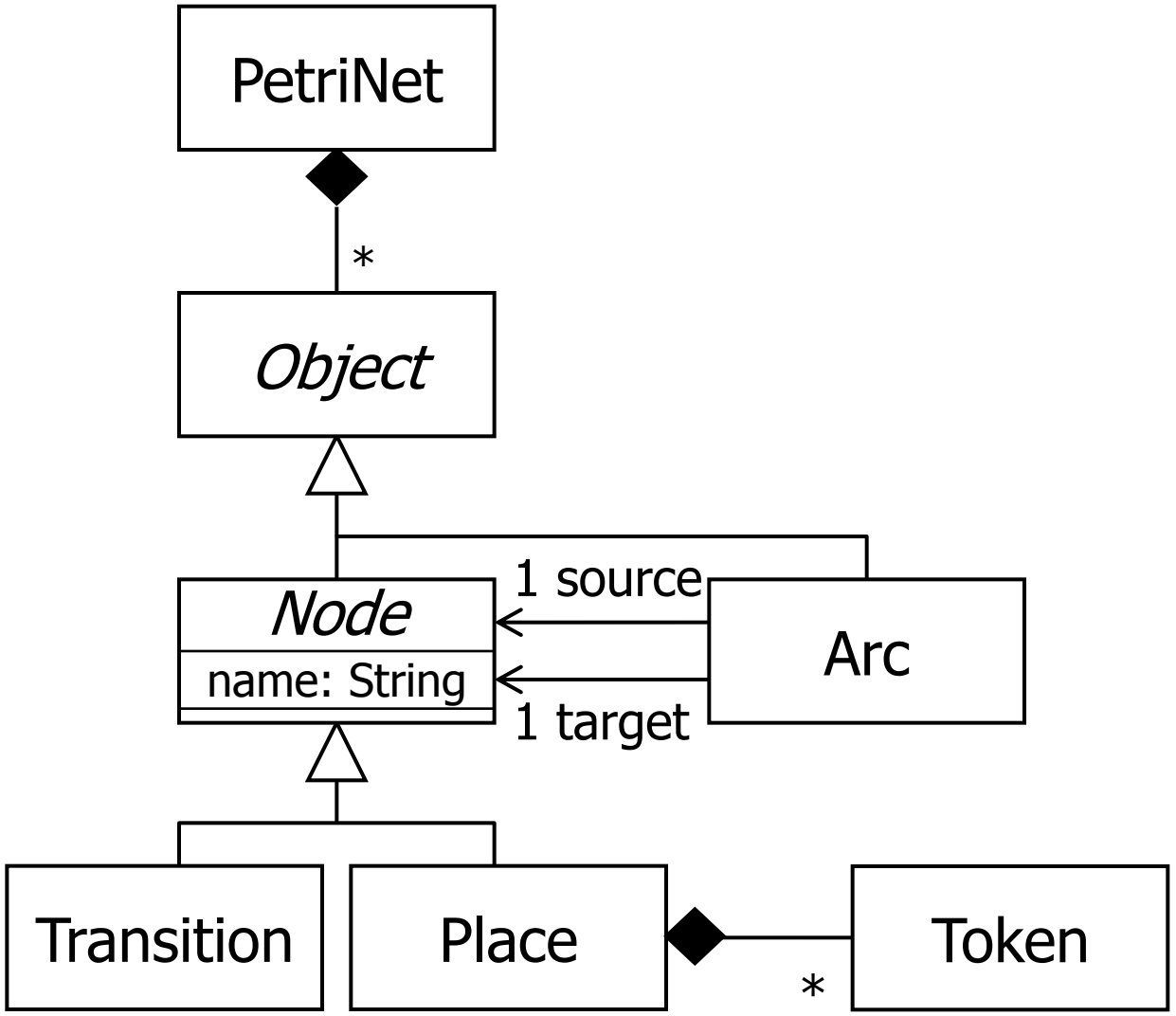
- UML class diagrams made slightly more general (not every instance of it is legal in the domain)
- Formulate some additional restrictions (which exclude the illegal instances)
- These additional restrictions are called **constraints**

Example (cf. L01)

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



Petri net model

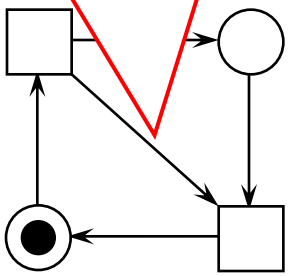


Domain model for Petri nets

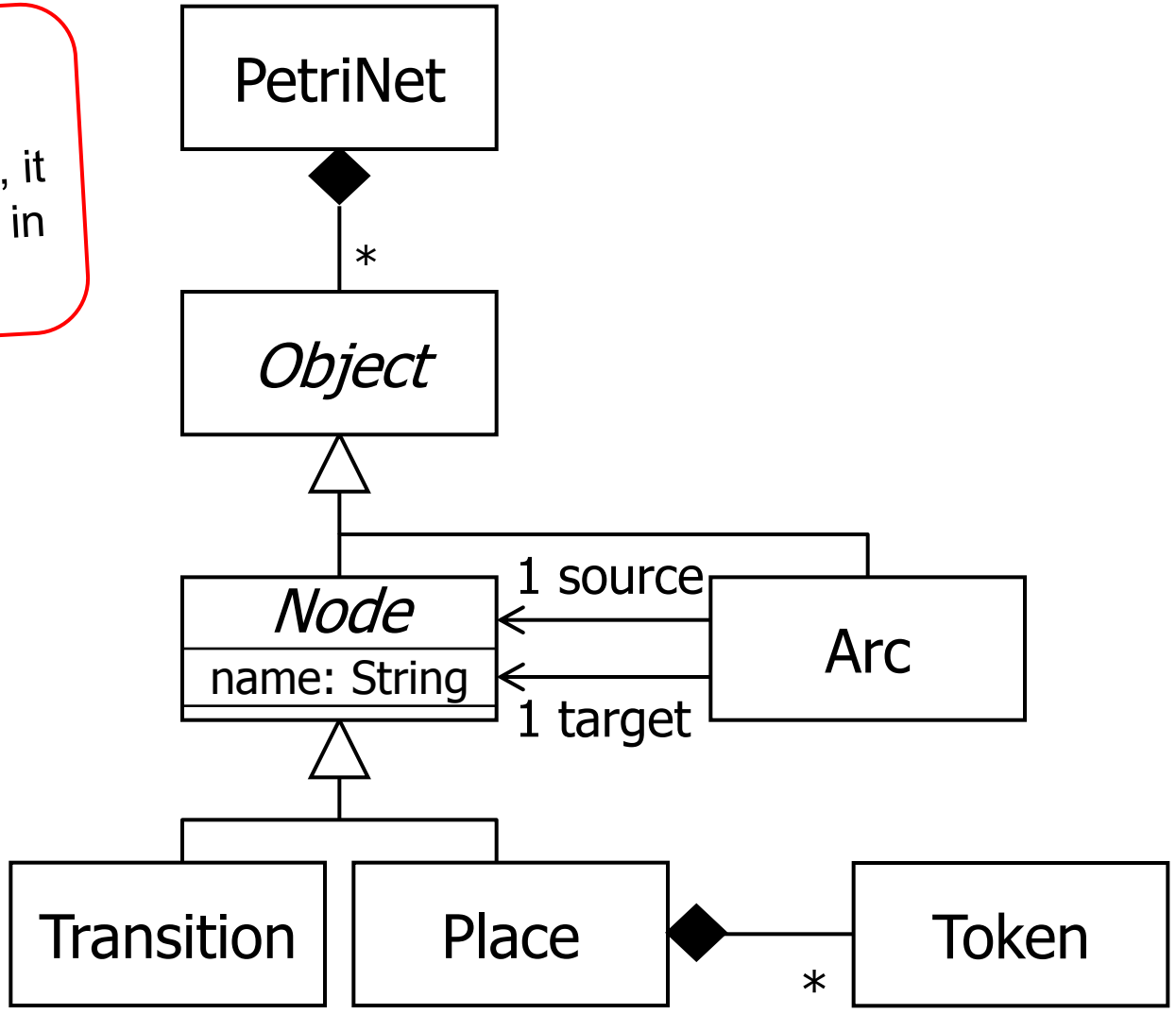
Example

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

According to the UML class diagram, this arc would be possible. But, it should not be possible in Petri nets!



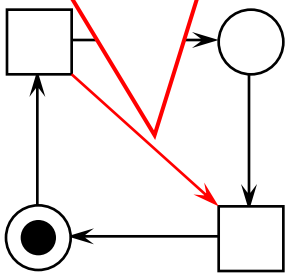
Petri net model



Domain model for Petri nets

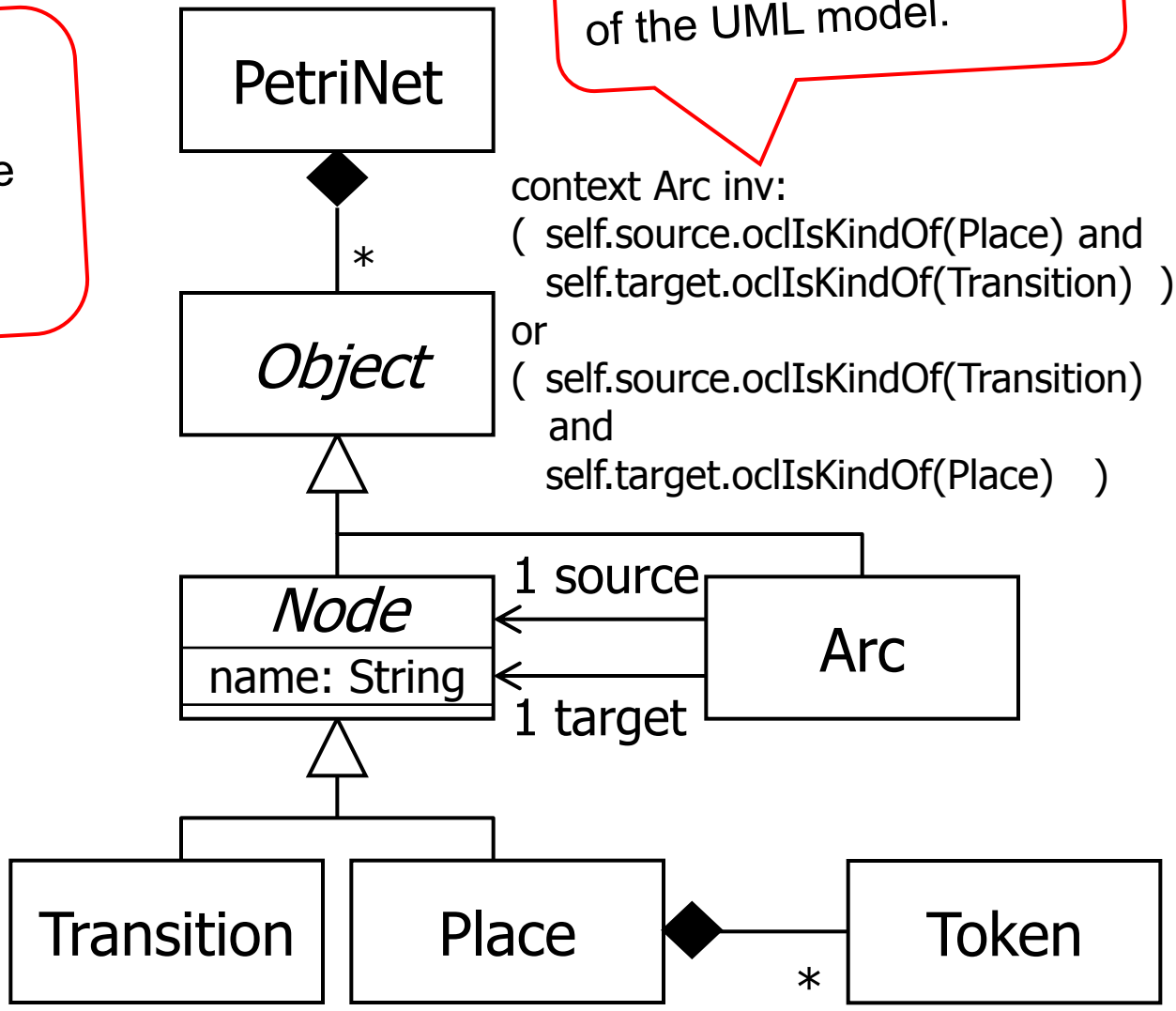
Example (cf. L01)

Now, this arc is no longer a legal instance of the UML model + the OCL constraint!



Petri net model

An OCL constraint (inv) for arcs (context) on top of the UML model.



Domain model for Petri nets

Questions

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- How to formulate constraints?
- How and when to check them?

And more technically:

- Where and how to register constraints?

2. Formulating constraints

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

There are different ways to formulate constraints:

- Dedicated constraint languages (like OCL)
- Programming languages (like Java)
- Logic
- ...

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

Object Constraint Language (OCL)

- OCL is an OMG (Object Mangagement Group) standard in a family of standards related to UML
- OCL is dedicated to formulate additional constraints on top of UML models independently from a specific platform and programming language
- There are different technical ways to automatically check the validity of OCL constraint (dependent on the underlying modelling technology, see 4)
- OCL is actually more:
  - Formulate pre and post conditions for methods
  - ”Implement” methods

See “body” option later

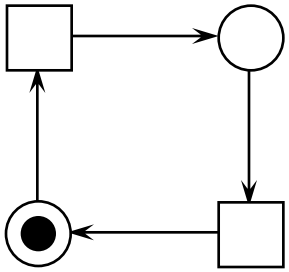
See context options later

Example

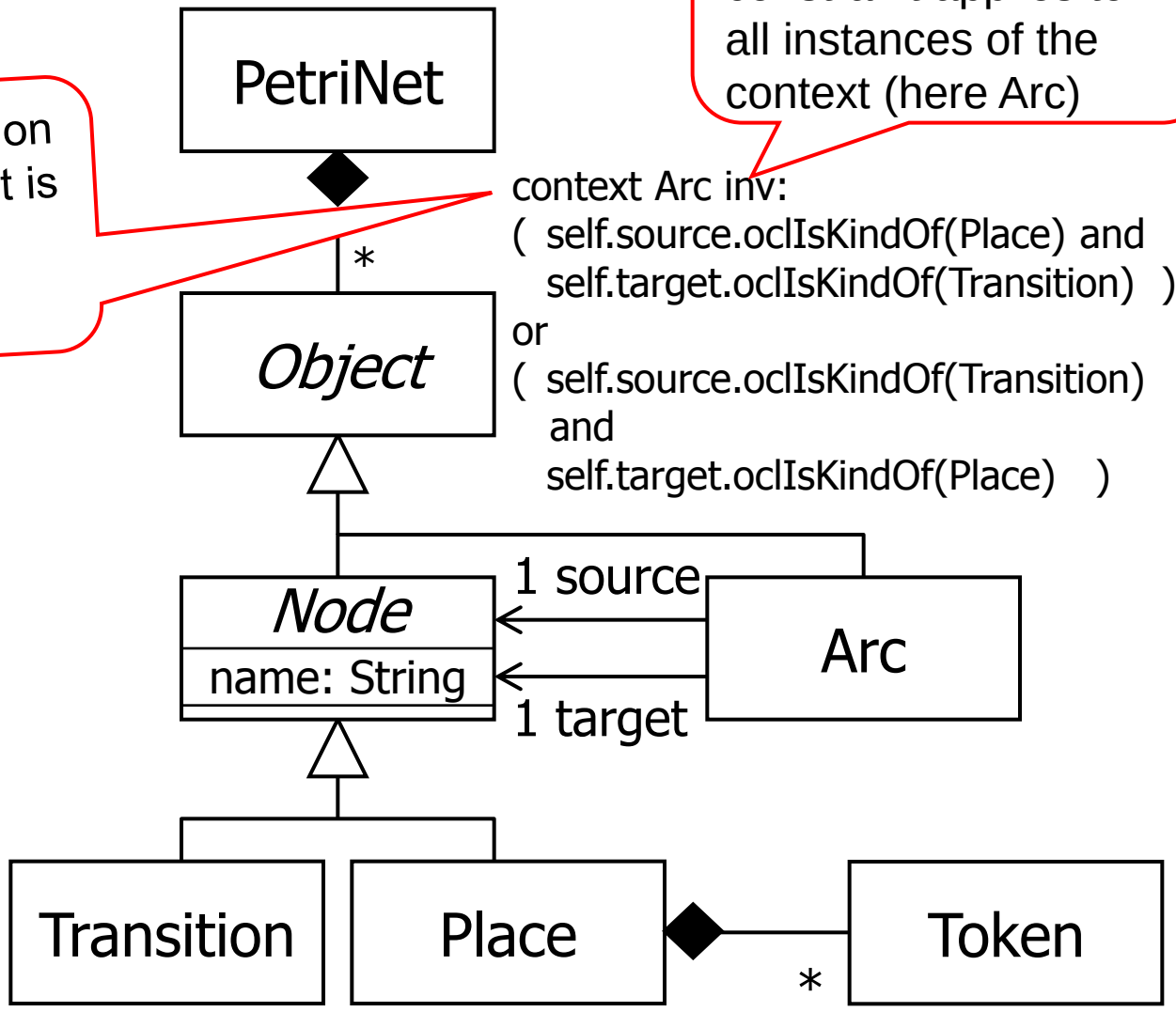
DTU Compute
Department of Applie
Ekkart Kindler

Stands for option <<invariant>>: constraint applies to all instances of the context (here Arc)

Defines the context on which this constraint is applied (here all objects of type Arc)



Petri net model



Domain model for Petri nets

OCL basics

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

OCL expressions

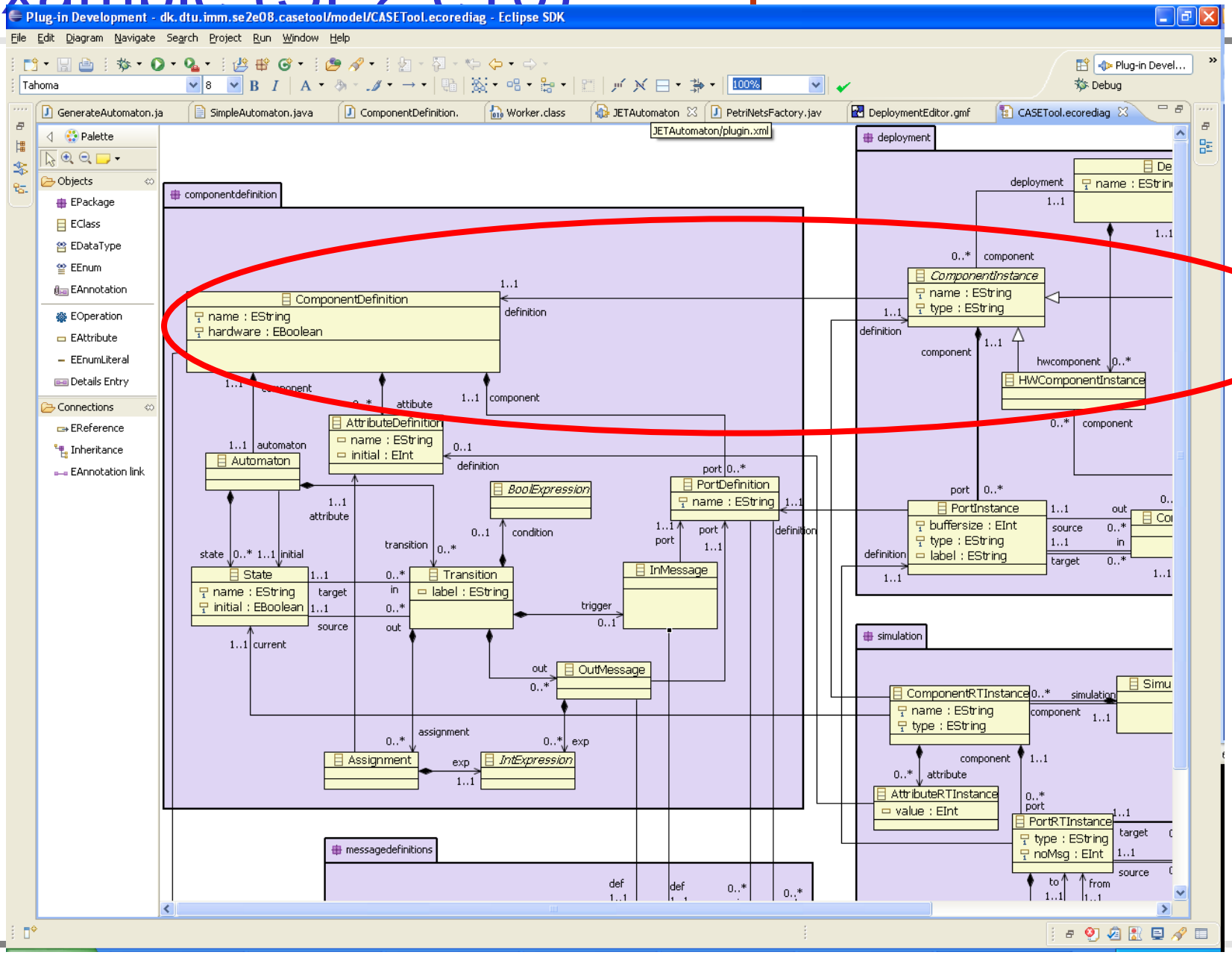
- start from the context object in OCL referred to by **self**

```
context Arc inv:
( self.source.oclIsKindOf(Place) and
  self.target.oclIsKindOf(Transition)  )
or
( self.source.oclIsKindOf(Transition)
  and
  self.target.oclIsKindOf(Place)  )
```

- from an object, may access attributes and operations (by dot-notation and resp. names)
- may navigate along associations
- may call operations and built-in functions

- can use Boolean operations: and, or, not, implies, ...
- and comparison operations: = , >, <, <=, ...

Example (SE2 e10)



Example (SE2 e10)

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

```
context HWComponentInstance inv:
    self.definition->size() > 0 and
    self.definition.hardware
```

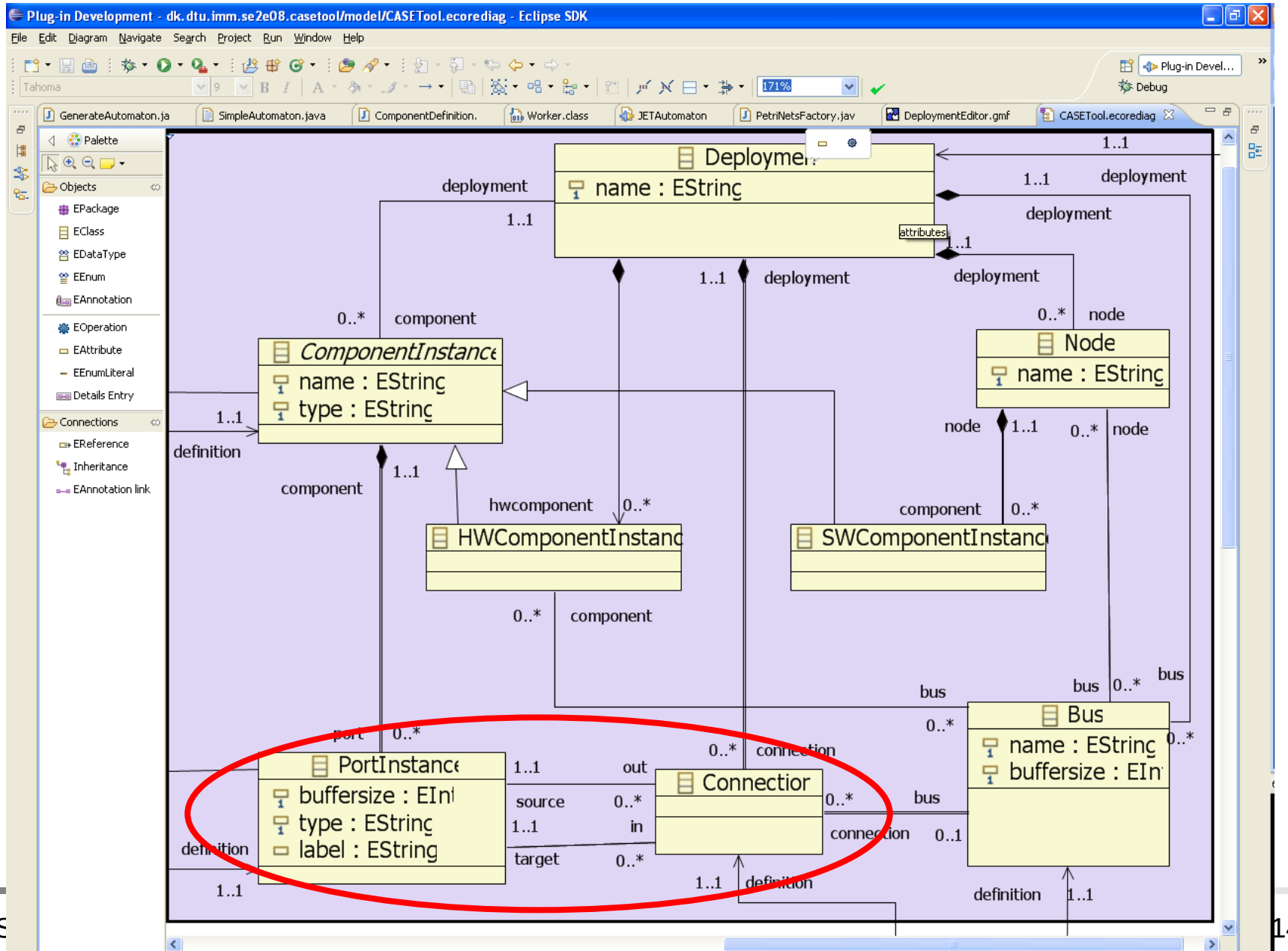
size() > 0 is one way of checking for non-null value.

"->" converts an object's attribute or reference to a set of its values

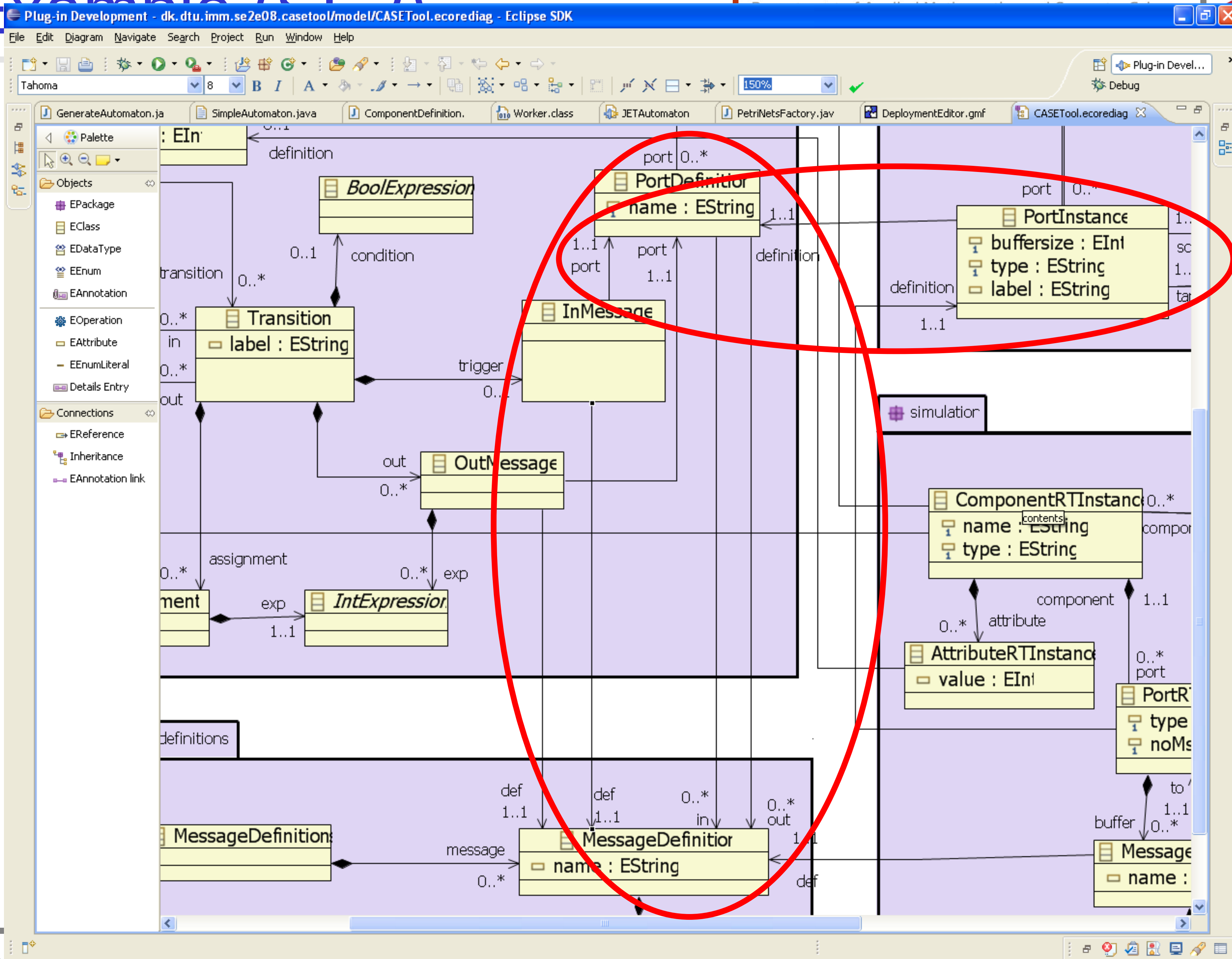
We can also check for *null* with … .isTypeOf(OclVoid)

Example (SE2 e10)

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



Example (SE2)


Plug-in Development - dk.dtu.imm.se2e08.casetool/model/CASETool.ecorediag - Eclipse SDK
File Edit Diagram Navigate Search Project Run Window Help
Plug-in Devel...
Debug
Tahoma
150%
GenerateAutomaton.ja
SimpleAutomaton.java
ComponentDefinition.
Worker.class
JETAutomaton
PetriNetsFactory.jav
DeploymentEditor.gmf
CASETool.ecorediag
Palette
Objects
EPackage
EClass
EDataType
EEnum
EAnnotation
EOperation
EAttribute
EEnumLiteral
Details Entry
Connections
EReference
Inheritance
EAnnotation link
BoolExpression
PortDefinition
name : EString
PortInstance
buffersize : EInt
type : EString
label : EString
Transition
label : EString
InMessage
OutMessage
IntExpression
simulation
ComponentRTInstance
name : EString
type : EString
contents
AttributeRTInstance
value : EInt
definitions
MessageDefinition
name : EString
definition
condition
transition
trigger
assignment
exp
port
in
out
def
message
component
attribute
buffer
to
0..*
1..1
0..1

Example (SE2 e10)

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

```
context Connection inv:
   self.source.definition.out->forAll( m1 |
     self.target.definition._in->exists(m2 | m1=m2 ))
  and
   self.target.definition.out->forAll( m1 |
     self.source.definition._in->exists(m2 | m1=m2 ))
```

"in" is a keyword of OCL; therefore the attribute "in" needs to be accessed via "_in" ("escaping" the keyword "in")

OCL basics

- o.oclIsTypeOf(T)
  checks if the type of o is **exactly** T

  Examples:
  self.oclIsTypeOf(pnmlcoremodel::Arc)
  self.oclIsTypeOf(yawl::Arc)

NB: Only one of them can be true on the same object

OCL basics

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- o.oclIsKindOf(T)
  checks if the type of o is T **or a subtype** of it
  (similar to Java instanceof)

- Examples:
  self.oclIsKindOf(pnmlcoremodel::Arc)
  self.oclIsKindOf(yawl::Arc)

If the first is true, the second will also be true!

OCL basics

- o.oclAsType(T)
  ”casts” object o to type T if possible

- Examples:
  context Tranistion inv:
  self.out->forAll(x |
    x.oclIsKindOf(yawl::Arc) implies
    not (x.oclAsType(yawl::Arc).type = yawl::AType::RESET))

Static type is pnmlcoremodel::Arc

Corresponds to type check (instanceof)

Casts to type (which then has the feature “type”)

OCL advanced

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- If an attribute or association has cardinality less or equal 1 the reference to that attribute or association returns a single value of the respecitive type (or ”null”, if it does not exist)
- If the cardinality is greater 1, the reference to it returns a set (collection) of the respective type
- These set operations are accessed via ->
- There are operations to select elements from sets and to quantify on sets.

OCL advanced

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- operations on sets
  - set->size()
  - set->iterate(x; res = init | exp(x,res))
  - ...
- Quantification on sets:
  - set->forAll(x | exp(x))
  - set->exists(x | exp(x))
    (can be nested)
  - ...
- OCL built-in operations and types

See http://www.omg.org/spec/OCL/2.4/PDF
for more details on OCL (version 2.4)

OCL expressions: Summary

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- There is much, much more (see OCL standard)!!
- It is important
  - to know that OCL exists,
  - be able to read OCL constraints,
  - be able to formulate simple OCL constraints,
  - be able to look up constructs in the standard
- Advantages
  - Simple things can be expressed quickly
  - Constraints can be expressed independently from a technology and programming language
- Disadvantages
  - Readability and expressiveness

The “Interactive OCL” console might help a lot (see pp. 37ff)

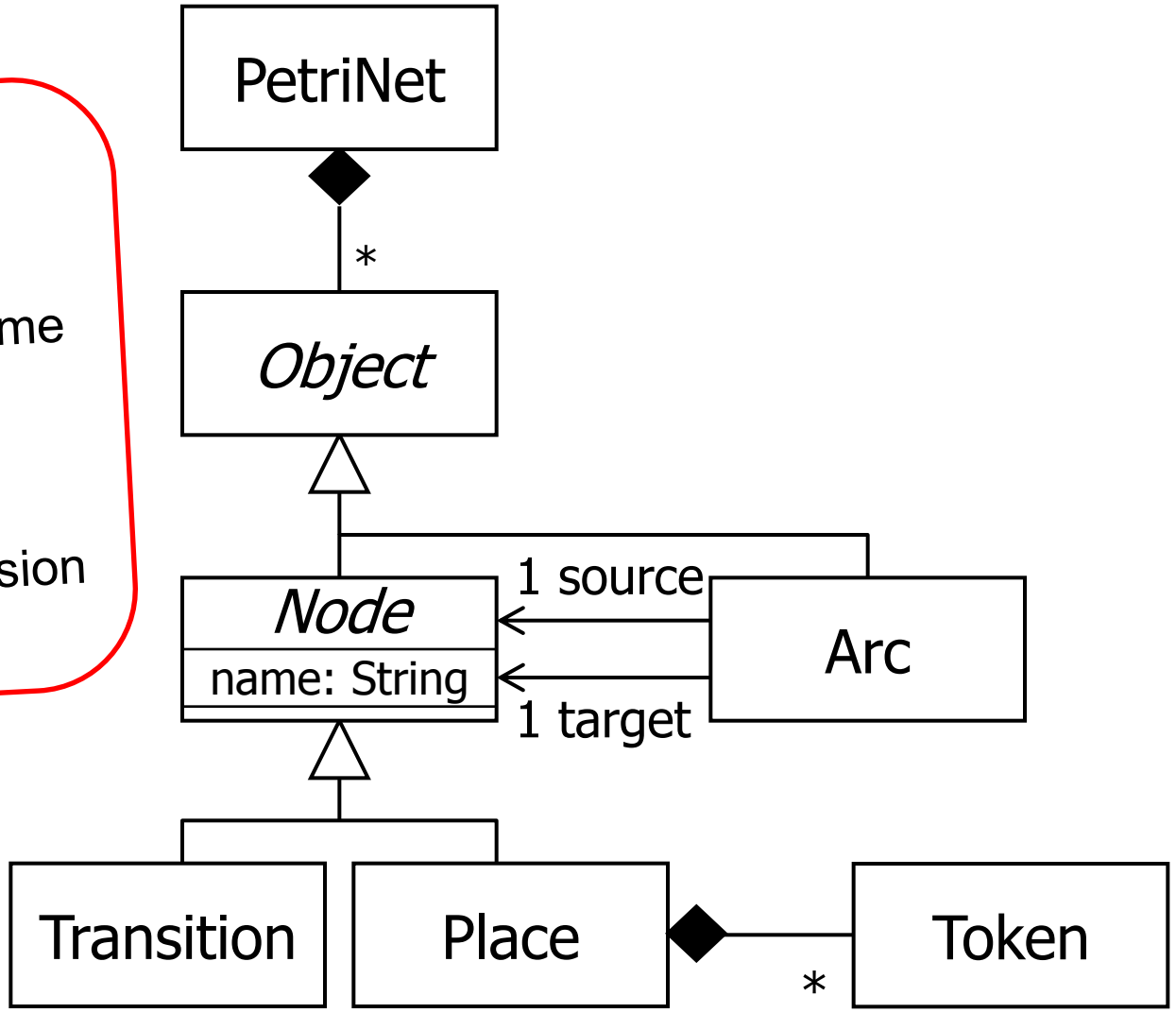
Exercise

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

Formulate

- No duplicate arcs
- Arc, its source and target belong to same net

→ Group work and blackboard discussion



Domain model for Petri nets

OCL contex

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- The context can be a class
  (self refers to an instance of that class)
  - option inv

Today, it depends on tool / framework in which way an expression is attached to a UML model and a context (see 4.)

- The context can be an operation
  (self refers to the object on which the operation is called)
  - options pre, post and body:
    pre and post define a pre- and post-condition
    body defines the result of an operation

In post, we can refer to values before execution by the @pre tag

- The context can be an attribute or association
  (self refers to the object to which it belongs)
  - options init and derived

OCL Opinion

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- OCL looks and feels much like programming with a flavour of logic
- Programmers are not so used to it, and often get OCL wrong

- In most modelling frameworks, it is possible to formulate constraints in your favourite programming language (for complex constraints this might be easier for you)

We will see an example in 2.2 / 4.

2.2 PL (Java)

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

Sometimes, it is more convenient to express a constraint in a programming language (typically, in the target language of the generated code).

The way this is done depends on the specific technology used.

But typically, there is an abstract class for constraints with some validation method which needs to be extended.

Example (see tutorial 7)

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

```
public class SomeConstraint extends AbstractModelConstraint {

  public IStatus validate(IValidationContext ctx) {
    EObject object = ctx.getTarget();
    // do whatever you need to do to establish whether
    // the constraint is violated or not

    if (object instanceof YAWLNet) {
      EObject container = object.eContainer();
      if (container instanceof PetriNet) {

        // return a failure in case the constraint is violated
        // for the given context
        return ctx.createFailureStatus(new Object[] {container});

        // and return a success otherwise
        return ctx.createSuccessStatus();
    } }
} }
```

Validation context (with target object); the actual target objects are defined somewhere else (see 4.)

3. When to validate

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

Two different policies when to **validate a constraint**:

Live: Whenever an instance is changed (by some command), the ”relevant” constraints are automatically checked; if a constraint is violated, the command is ”rolled back” (undone automatically)

→ live contraints cannot be violated

See 4.

Batch: Only checked when a validation is requested (programmatically or by the end-user)

→ batch constraints may be invalid for a while; violations are detected at validation time only;

→ the end-user has the responsibility to fix them

Discussion

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

Live or batch: Which is better?

As always:

It depends on the constraint and the domain, which is better.

We need to specify for each constraint which ”mode” it is (see 4. for details).

4. Validation framework

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- OCL has a precisely defined meaning (independently from a specific programming language or implementation of the model)

- The way to ”hook in” OCL constraints, however, depends on the used technology (e.g. EMF Validation Framework)

For examples, see Sect. 4.5.1.4 of the ePNK manual.

Example

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

```
<extension point="org.eclipse.emf.validation.constraintProviders">
   <constraintProvider cache="true">
      <package namespaceUri="http://se.compute.dtu.dk/mbse/yawl"/>
      <constraints categories="org.pnml.tools.epnk.validation">


         <constraint
            id="dk.dtu.compute.mbse.yawl.validation.correct-arc-connection"
            lang="OCL"
            mode="Live"
            name="Arc connection constraint for YAWL nets"
            severity="ERROR"
            statusCode="401">
           <message>
             The arc {0} with this arc type is not allowed between these elements.
           </message>
           ...
```

Constraints are plugged in as part of a constraint provider.

Constraints need to be plugged in wrt. a defined category (here we use the one defined by the ePNK).

Defines an OCL constraint

Defines the mode/policy

Defines the severity

Some status code (in the ePNK numbers > 400 should be used for PNTD violations)

Example

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

```
<description>
   Arcs must be between a place and a transition, a transition and a place, or
   between two transitions. Only arcs between a place and a transition may
   have a type!
</description>
<target class="Arc:http://se.compute.dtu.dk/mbse/yawl">
   <event name="Set">
      <feature name="source"/>
      <feature name="target"/>
      <feature name="type"/>
   </event>
</target>
```

...

Defines the target objects (instances of Arc class from the Ecore package for YAWL nets)

For a live constraint, we need to define which events should issue a validation of this constraint. Here: the set event on the features source, target or type of an Arc object.

Example

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

```
<![CDATA[
   ( self.source.oclIsKindOf(pnmlcoremodel::PlaceNode) and
     self.target.oclIsKindOf(pnmlcoremodel::TransitionNode) )
   or
   ( self.source.oclIsKindOf(pnmlcoremodel::TransitionNode) and
     self.target.oclIsKindOf(pnmlcoremodel::PlaceNode) and
     self.type->size() = 0 )
 ]]>
</constraint>
...
```

It is quite tricky to get the syntax of OCL correct. Use the **"Interactive OCL" console** to check the syntax before you plug in the constrain (see slide 32ff for details).

The actual OCL constraint (note that the context (Arc) and the option (inv) are not mentioned – have been defined in the xml code above.

Discussion: what does self.type-> size() = 0 do?

Example


```
<constraint
    lang="Java"
    class="dk.dtu.compute.mbse.yawl.constraints.StartEndConditions"
    severity="ERROR"
    mode="Batch"
    name="One start and end place"
    id="dk.dtu.compute.mbse.yawl.validation.one-start-and-end-place"
    statusCode="402">
  <target class="YAWLNet:http://se.compute.dtu.dk/mbse/yawl"/>
  <description>
    A YAWL net must have one start and end place.
  </description>
  <message>
    The net {0} does not have exactly one start place and one end place.
  </message>
</constraint>
...
```

Defines an Java constraint

FQN of Java class implementing the constraint

Defines the mode/policy

Message in case the validation fails ({0}, {1}, ... refer to the objects returned by the validation).

Example


```
... <!-- there could be more Java or OCL constraints here -->

    </constraints>
  </constraintProvider>
</extension>
```

5. Summary

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- Constraints conceptually belong to the domain model
- Different ways to formulate constraints (OCL, Java, ...)
- Uniform way to validate them in all applications (properly using the model) via the Validation Framework; the framework defines how to technically add constraints to a model
- Important: chose the right validation policy/mode (live/batch)

6. Interactive OCL Console

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

The “Interactive OCL” console will help you checking whether the syntax of your OCL expressions is correct and even help you come up with expressions in correct syntax.

- Install the “Interactive OCL” console via
  Help → Install New Software...

- Work with update site:
  Neon - http://download.eclipse.org/releases/neon/

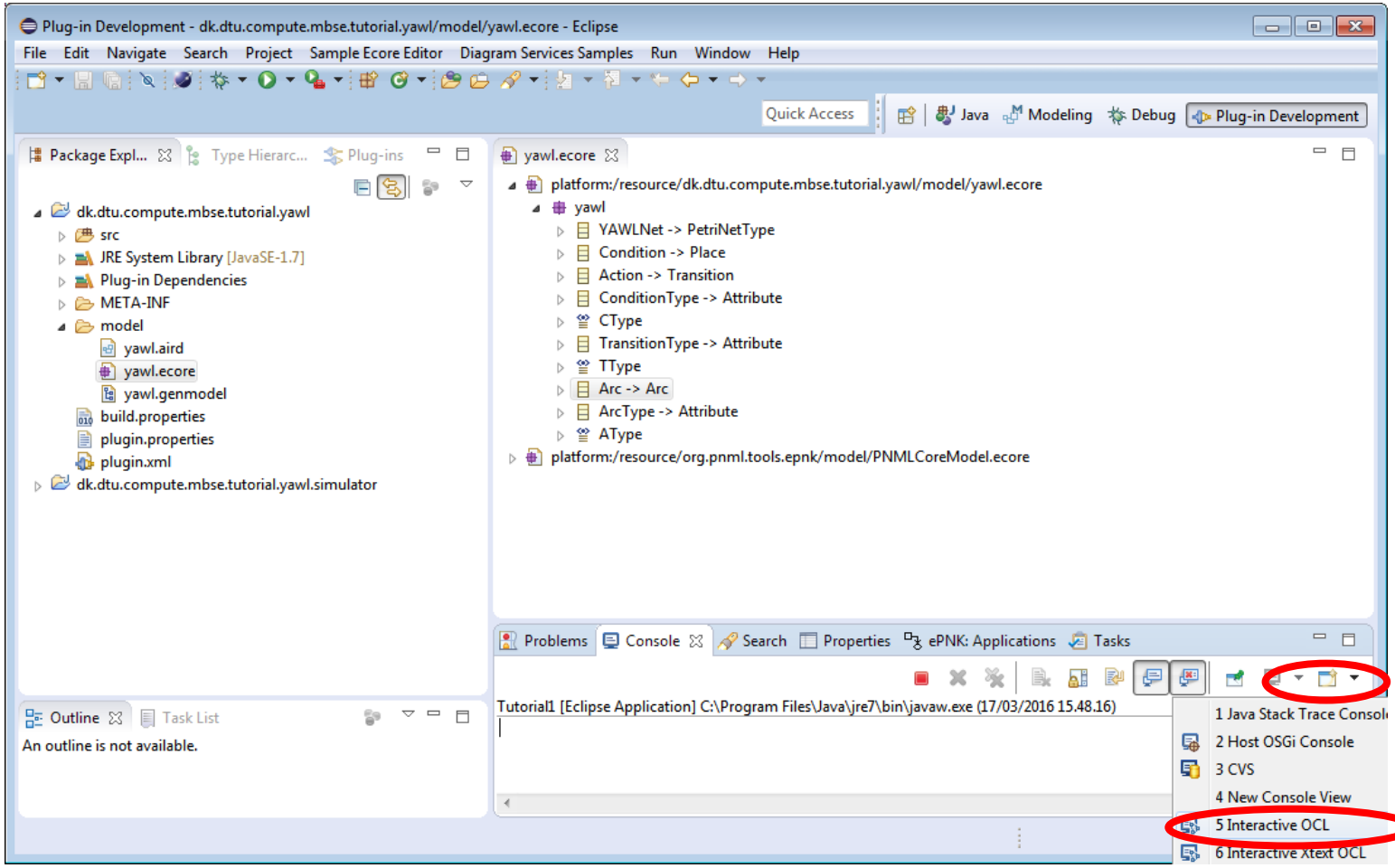
- Select feature:
  “OCL Examples and Editors SDK”
  from the category “Modeling”

Interactive OCL Console

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

After successful installation (and restart):

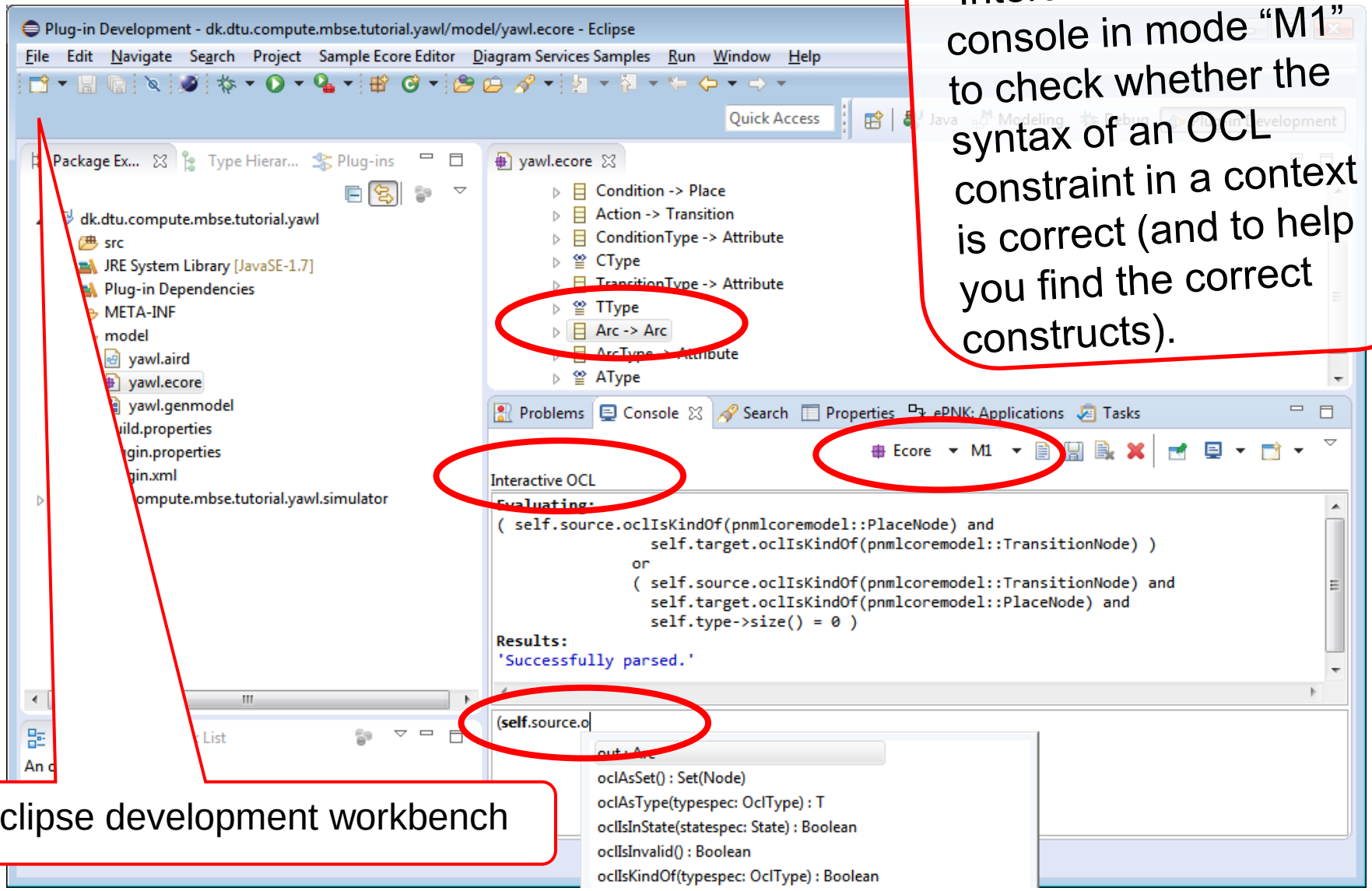
Open the console view and select “Interactive OCL”



Interactive OCL Console (M1)

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

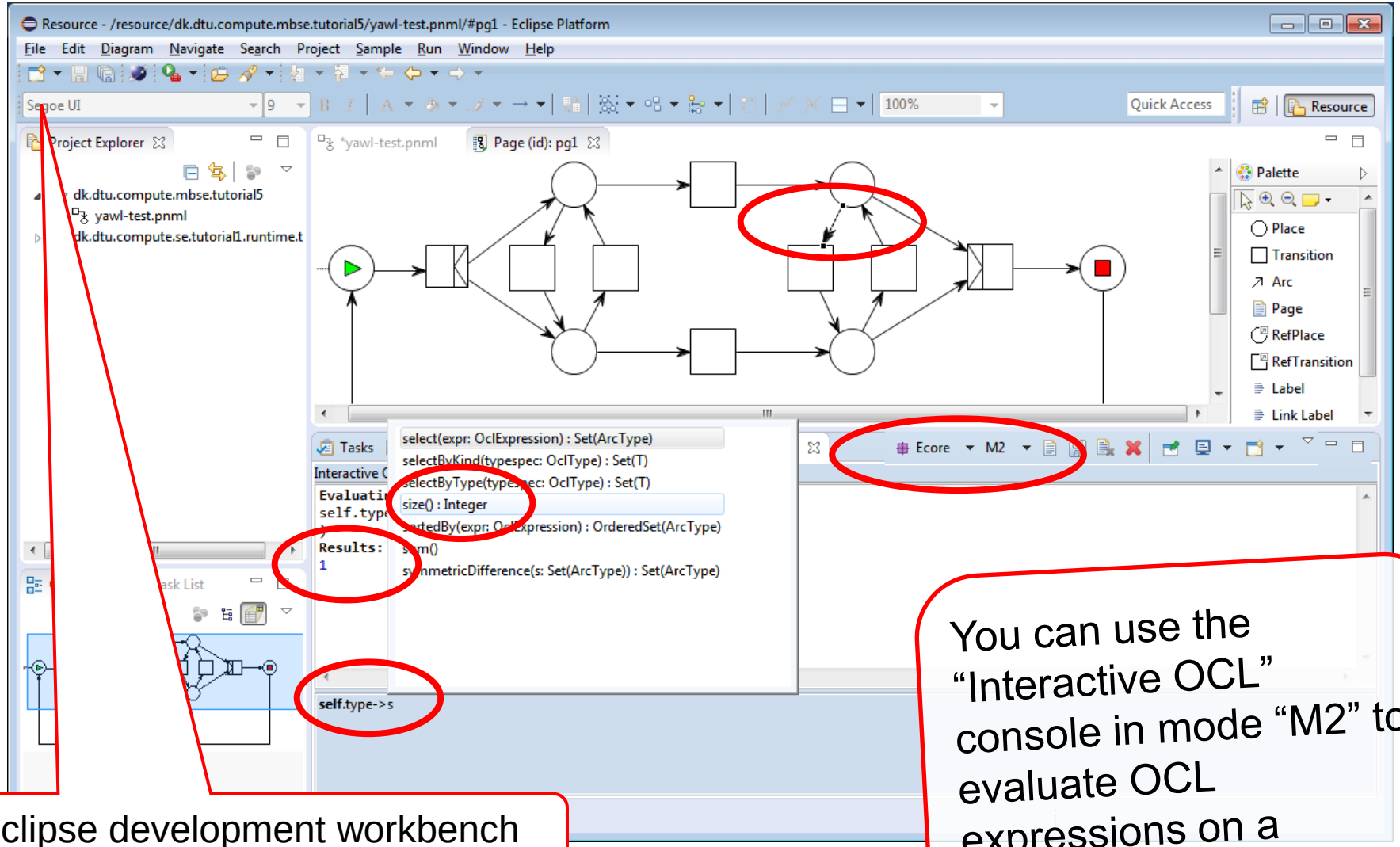
You can use the "Interactive OCL" console in mode "M1" to check whether the syntax of an OCL constraint in a context is correct (and to help you find the correct constructs).



Eclipse development workbench

Interactive OCL Console (M2)

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



Eclipse development workbench

You can use the "Interactive OCL" console in mode "M2" to evaluate OCL expressions on a selected target element.