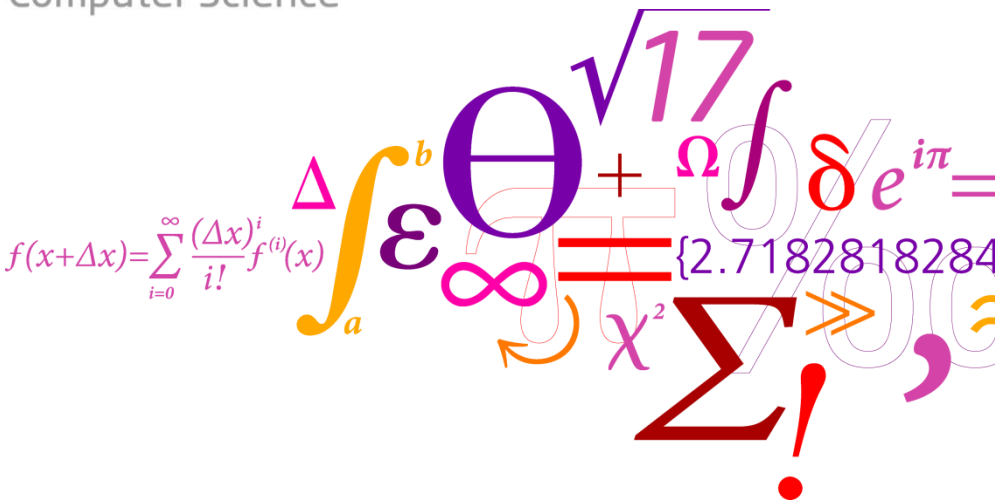
Model-based Software Engineering
(02341, spring 2016)

Ekkart Kindler

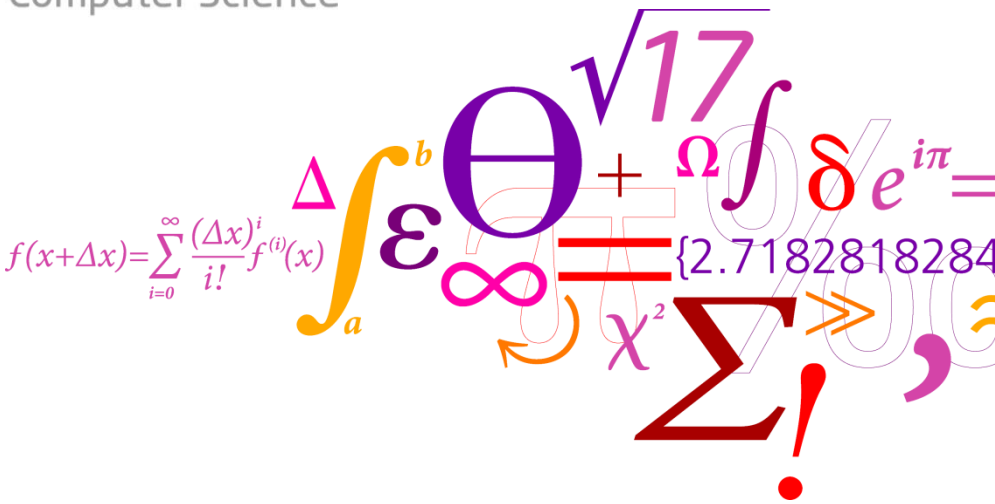
DTU Compute
Department of Applied Mathematics and Computer Science



V. The ePNK & PNML

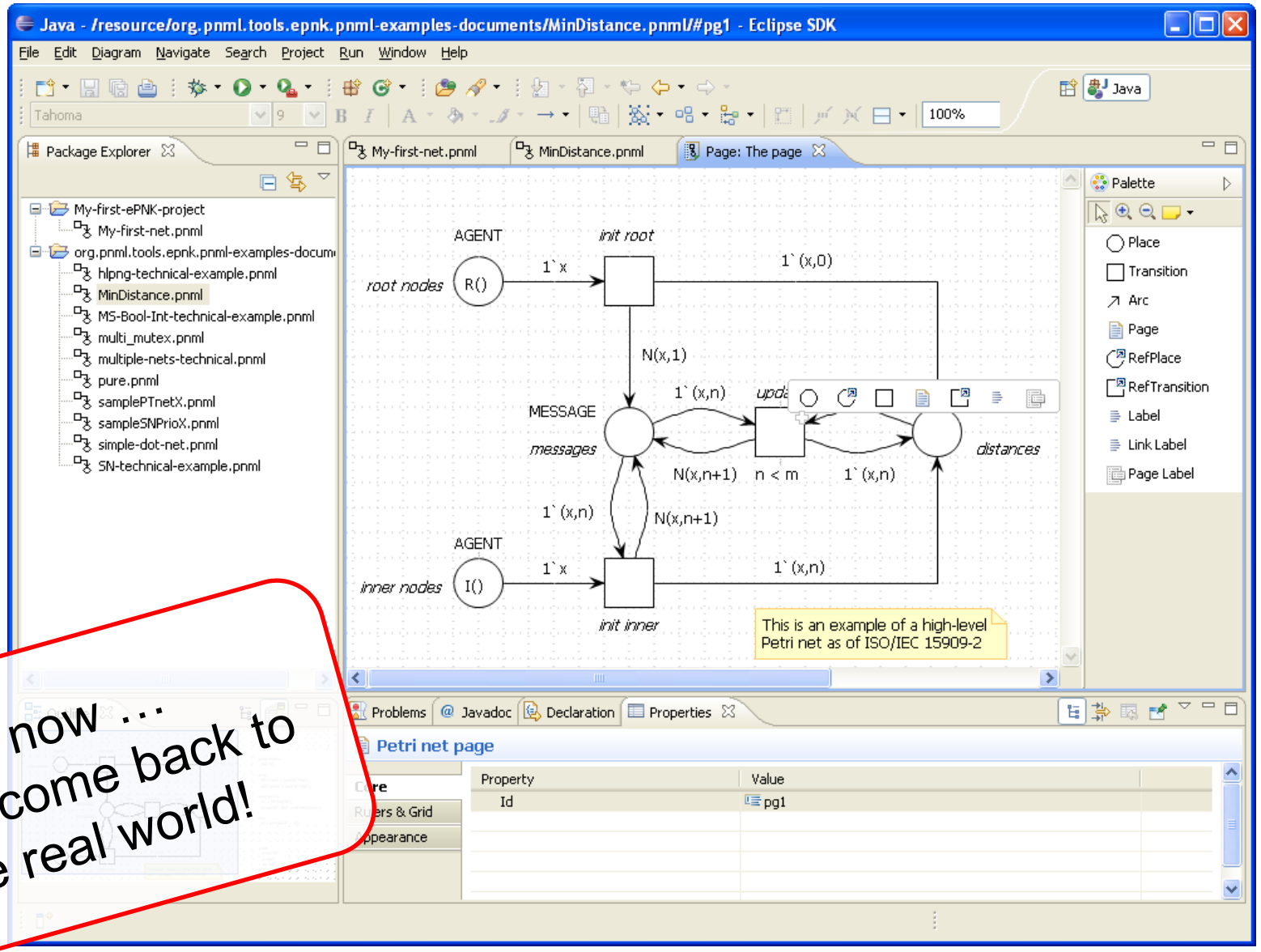
DTU Compute
Department of Applied Mathematics and Computer Science

No new slides



Motivation

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



Real world issues

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- Many more features:
  Pages, reference nodes, ...
- Need to define specific XML syntax (PNML)
- Different versions of Petri nets
  (each would need a separate GMF-editor)
- Definition of new versions of Petri net types
  without touching the existing tool,
  (almost)without programming

as well as some other extensions

2. 1. PNML in a nutshell

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- The Petri Net Markup Language (PNML) is an XML-based transfer format for “all kinds” of Petri nets.

- PNML is an International Standard: ISO/IEC-15909-2

  Part 2: focus on high-level nets

  Part 3: different extensions

  - modularity
  - type and feature definitions
  - particular versions of Petri nets
  - …

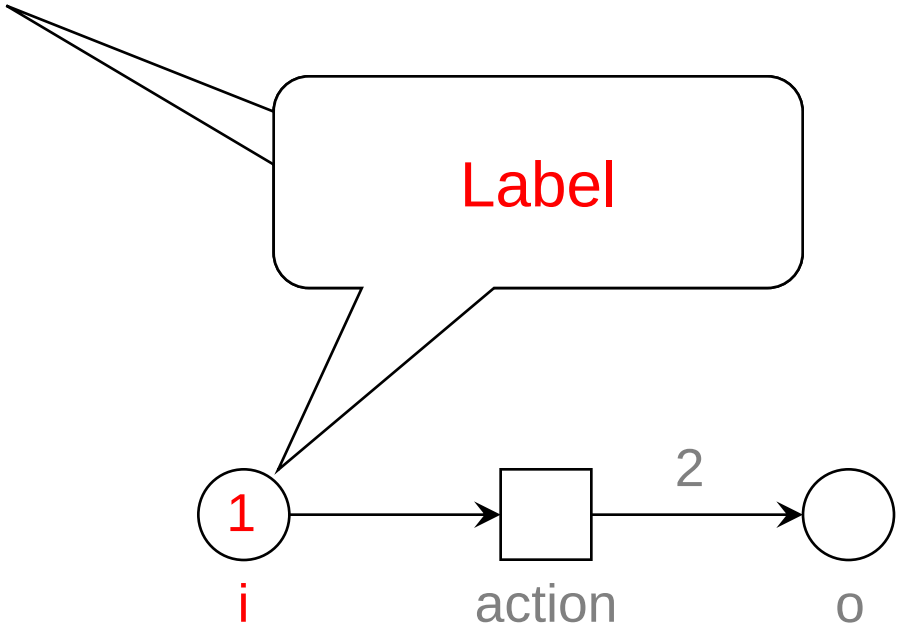
Note that Part 3 is not an international standard yet. Part 1 is under revision.

A first example

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

```
...
    <place id="p1">
      <name>
        <text>i</text>
      </name>
      <initialMarking>
        <text>1</text>
      </initialMarking>
    </place>
...
```

The particular kind of label depends on the „kind“ of Petri net.



Basic Idea

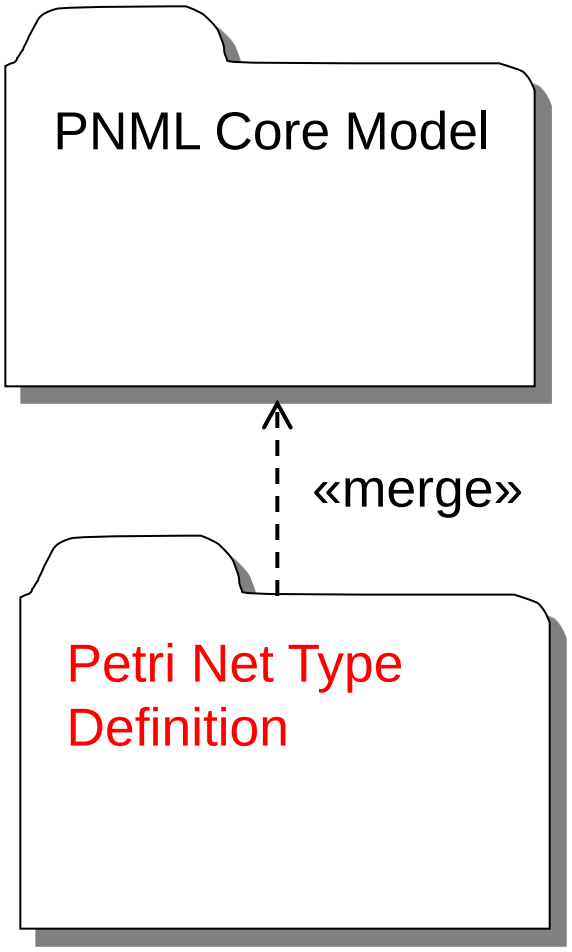
DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

„All kinds“ of Petri nets can be represented by

- places
- transitions, and
- arcs

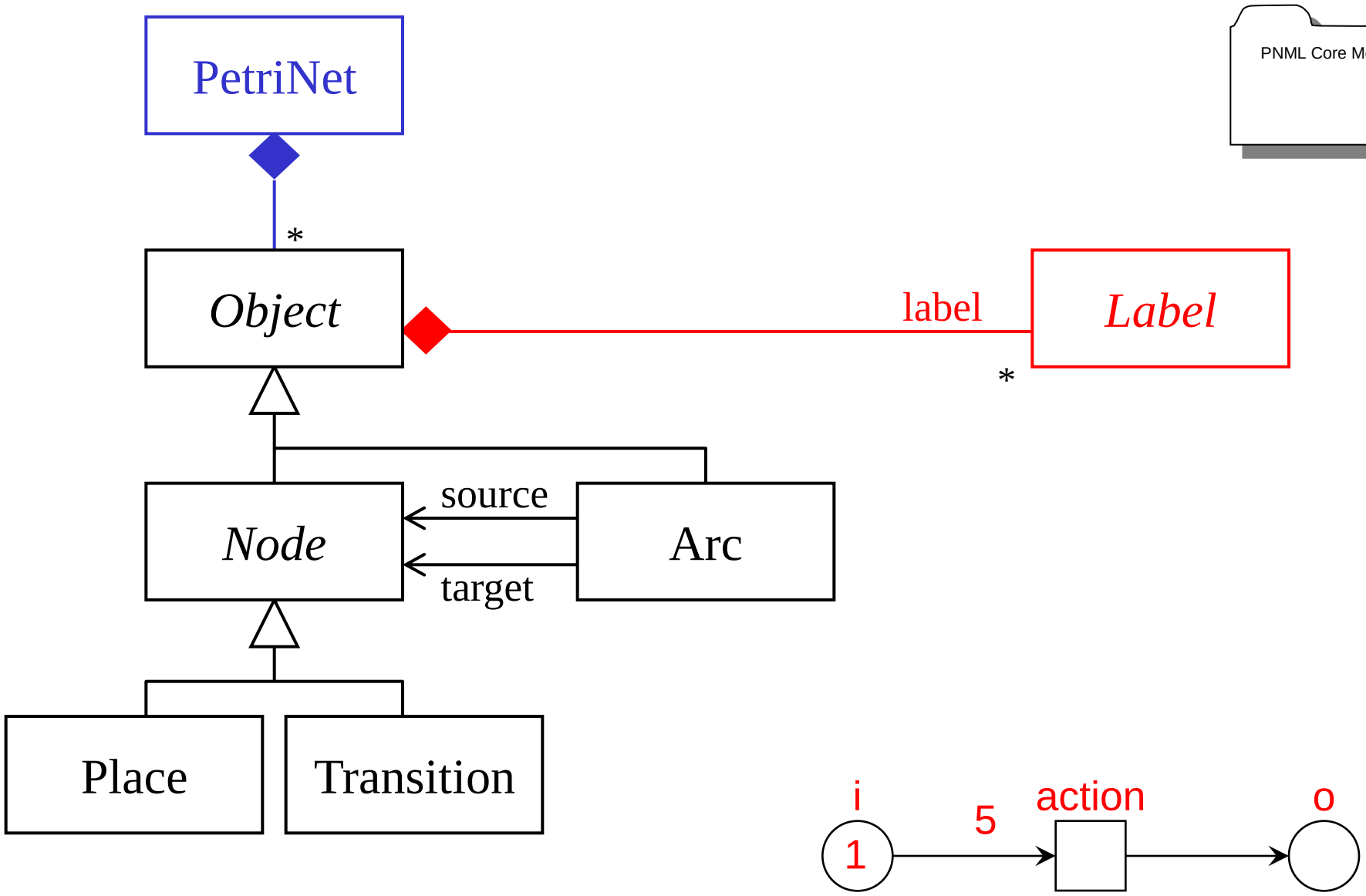
along with some

- labels



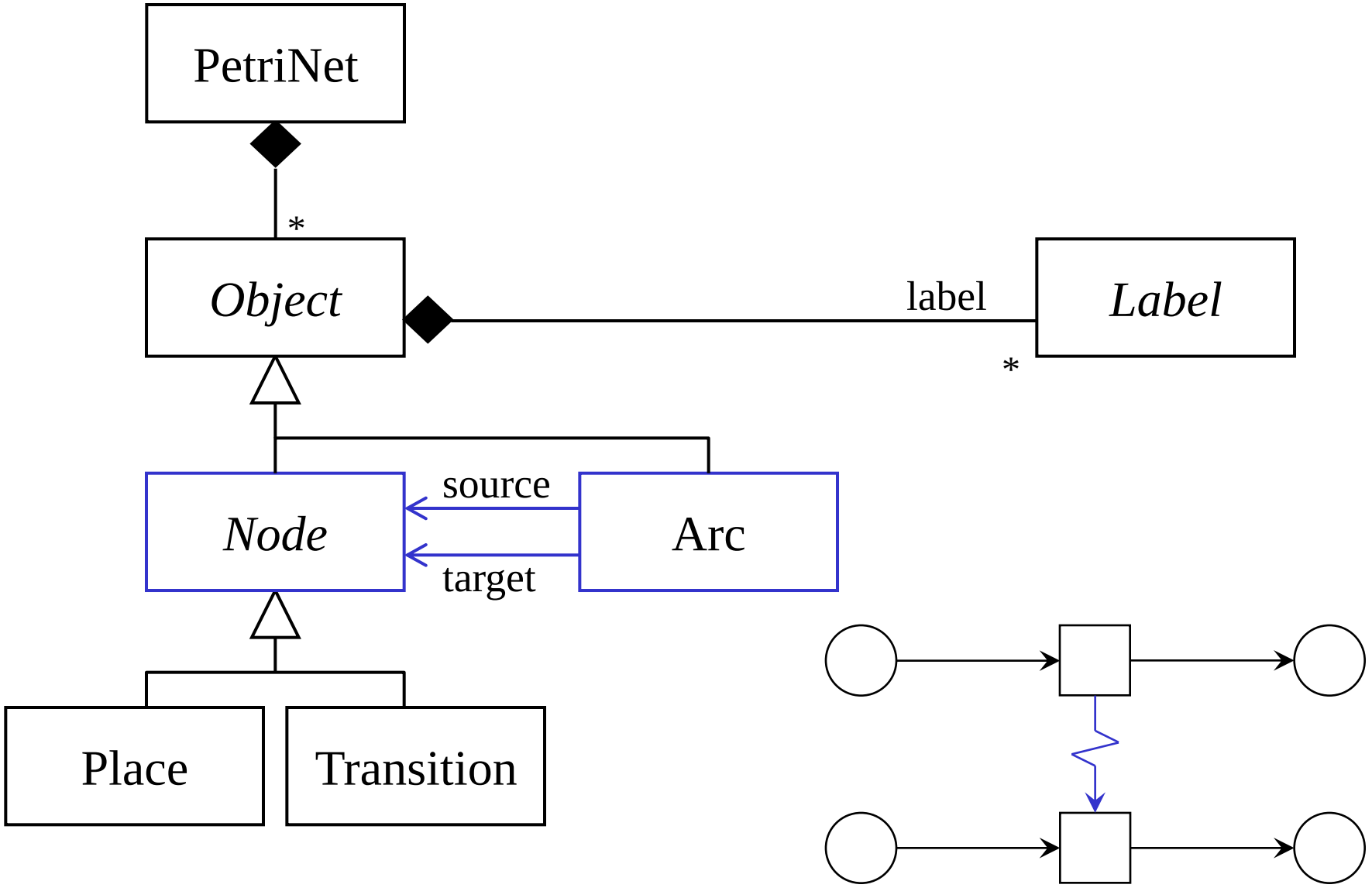
2.2 Core Model (overview)

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



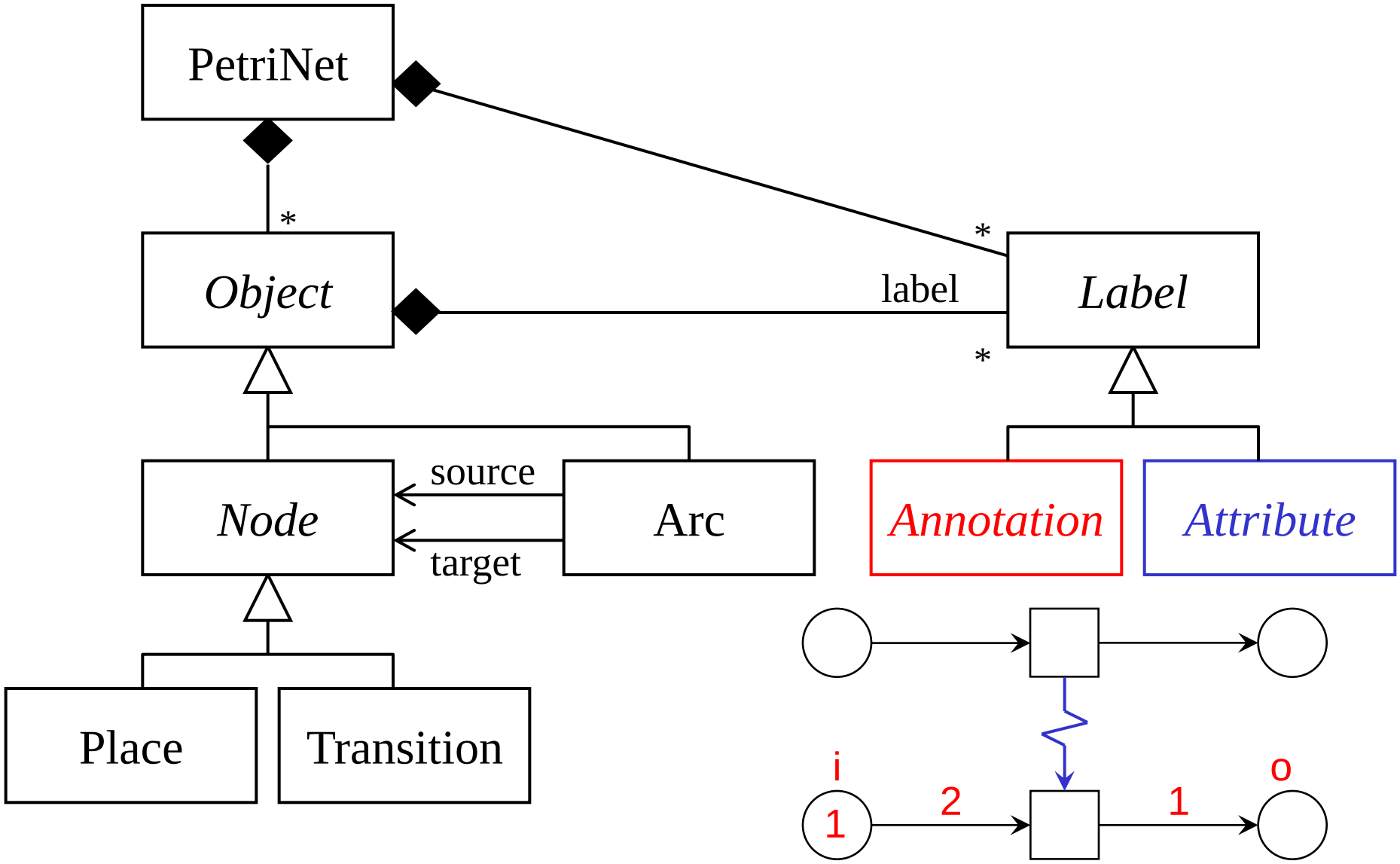
Core Model (overview)

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



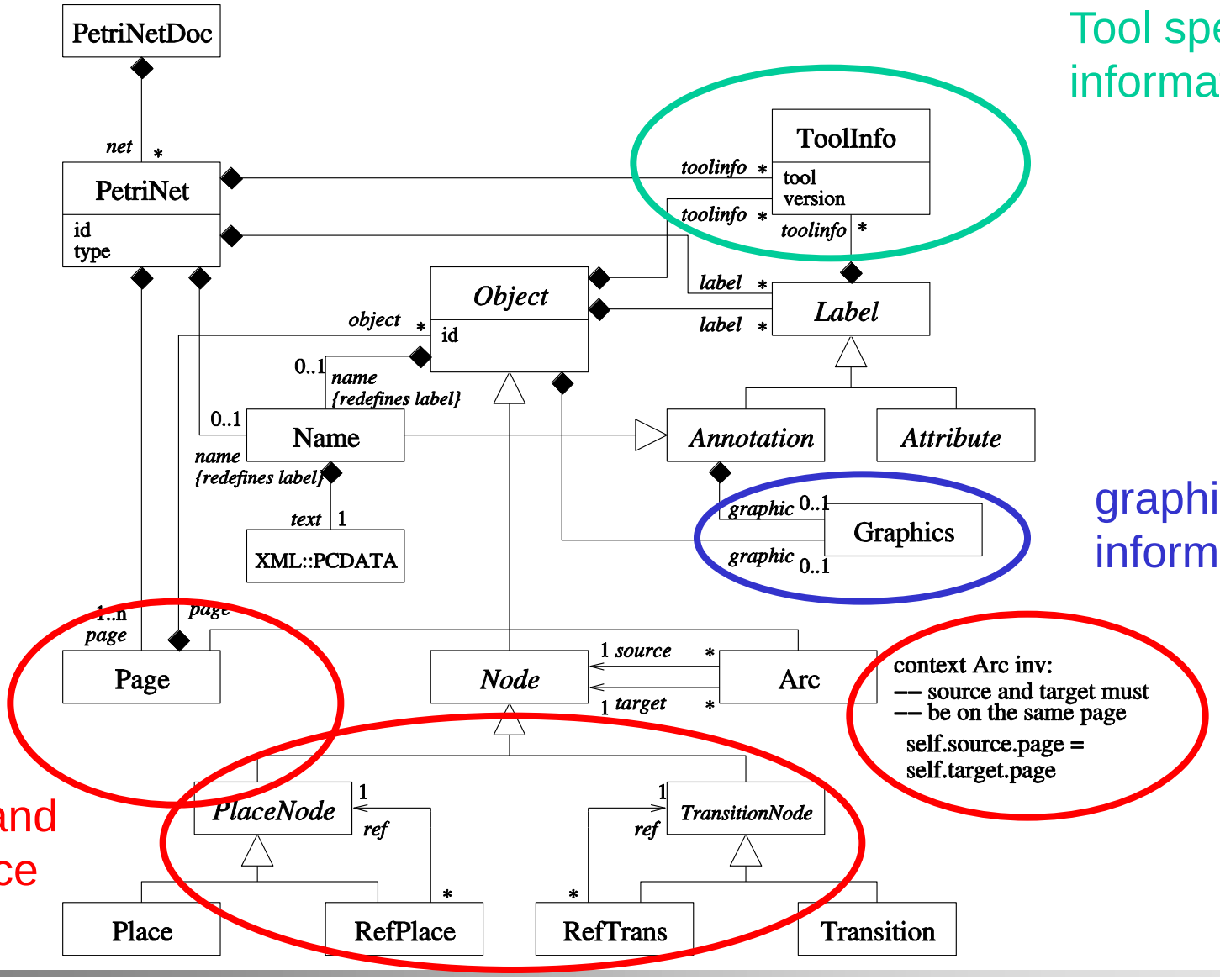
Core Model (overview)

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



PNML Core Model

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



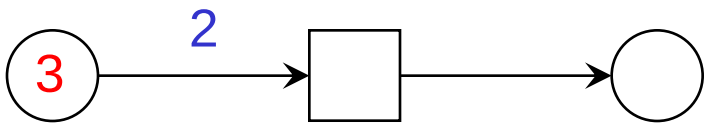
Tool specific information

graphical information

pages and reference nodes

2.3 Type Definition

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



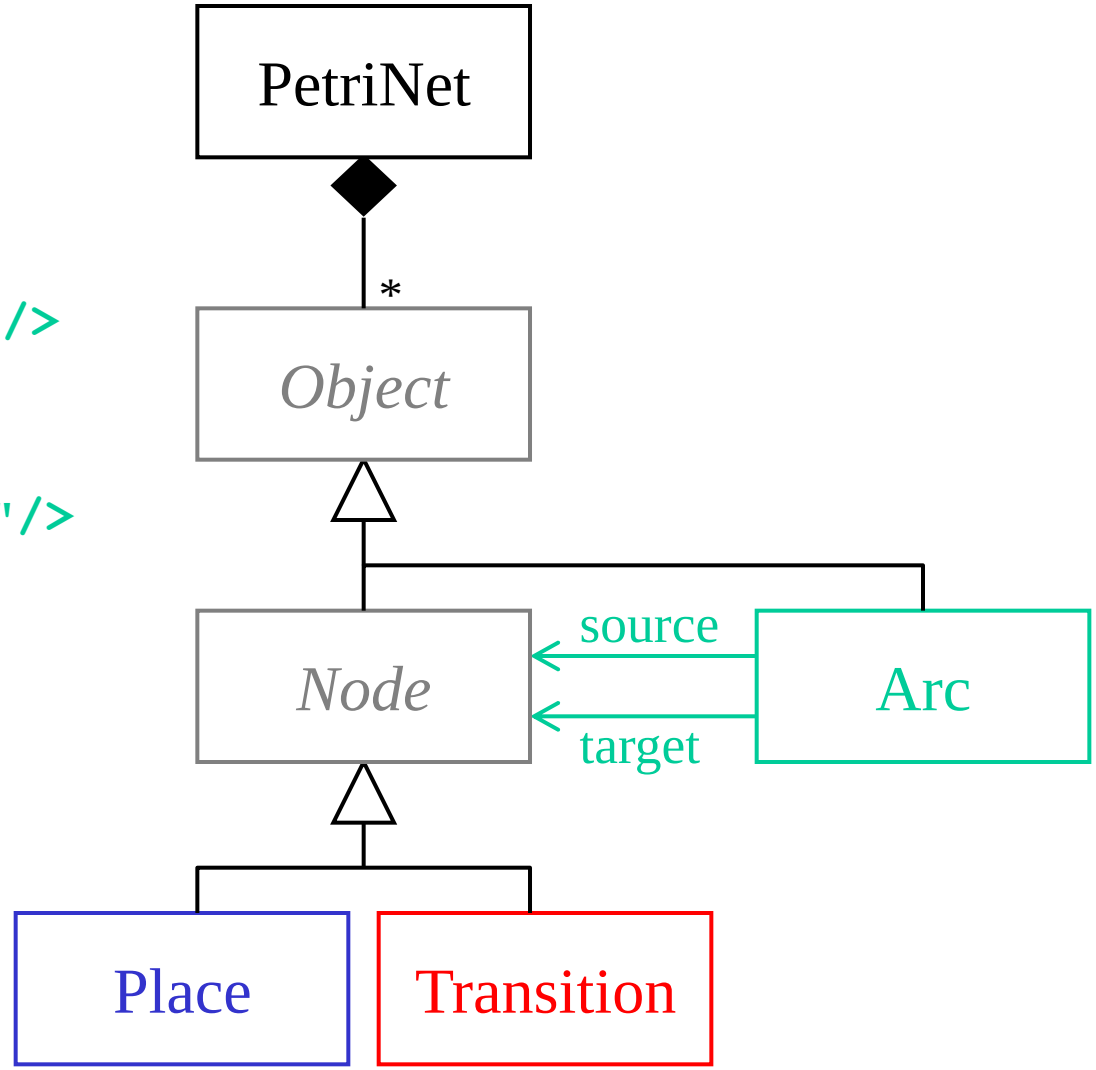
Outline

- Motivation
- PNML
  - Overview
  - Core model
  - Type model
  - Mapping to XML
- Experiences and statistics
- ePNK: Tutorial

2.4 Core Model in XML

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

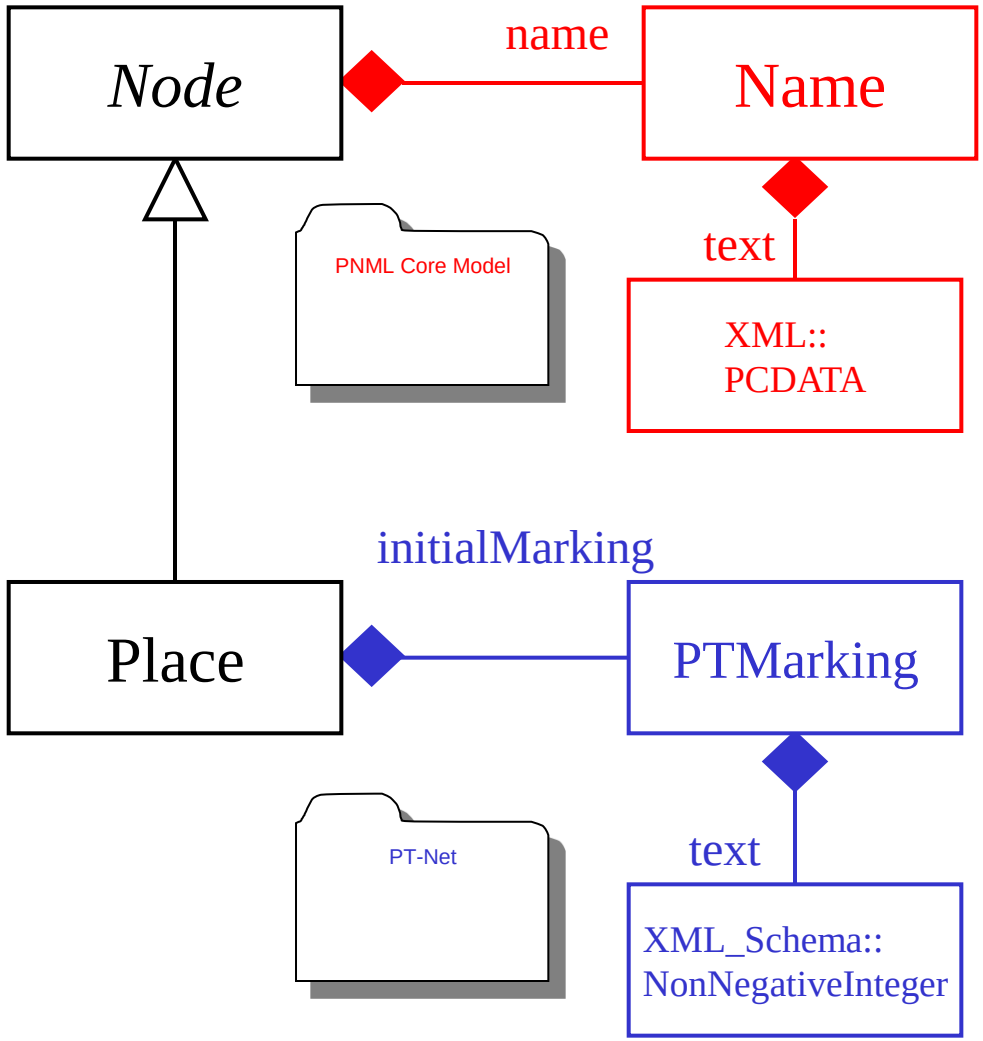
```
<pnml xmlns="http:...">
 <net id="n1" type="...">
  <place id="p1"/>
  <arc id="a1" source="p1"
               target="t1"/>
  <transition id="t1"/>
  <arc id="a2" source="t1"
               target="p2"/>
  <place id="p2"/>
 </net>
</pnml>
```



Labels in XML

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

```
...
  <place id="p1">
    <name>
      <text>i</text>
    </name>
    <initialMarking>
      <text>1</text>
    </initialMarking>
  </place>
...
```



In general

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- How can this mapping be defined in general?
  - Core model:
    Just implement it
  - Petri net type:
    Just implement it
    - code it for every new type!
    - interface with rest?

Better idea: use infrastructure to map model concepts to XML (ExtendedMetadata) + some additional ePNK specific mechanisms.

Not discussed in this course, you do not need to define mappings to XML (use the default mapping).

Outline

- Motivation
- PNML
  - Overview
  - Core model
  - Type model
  - Mapping to XML
- Experiences and statistics
- ePNK: Tutorial

3. Experience

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- Time effort: Altogether (up to version 0.9.0) < 5weeks
  - ca. **1 week** for making the core model and implementing core infrastructure (only EMF, generic Petri net types, XML mapping mechanism)
  - ca. **1 week** for HLPNG Petri net type, the model, its PNML-mappings and the parser for labels (Xtext)
  - ca. **½ week** for extending the PNML-mapping infrastructure so that all HLPNG features can be mapped to XML
  - ca. **½ week** for implementing the validation constraints for HLPNG (correct typing of expressions, resolution of types, …)
  - ca. **1 week** for graphical editor
  - ca. **½ week** for brushing up the graphical editor (and cleaning a bit up behind the scenes)

Part of that 1 week of debugging! 2 days my own bugs; 3 days replacing missing documentation!

As of version 0.9.2 with some extra features:
8 weeks!

Code statistics

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- Petri Net Type: P/T-Net plugin
  - Ecore model for P/T-nets
  - XML-mapping: 2 lines
  - manual changes in one generated class (4 lines, 2 of them for the above XML-mapping)
  - 1 OCL constraint

Implementing the complete type for inhibitor nets type took *me* 33 minutes!

- Tool-specific extension: Token position plug-in
  - model for token positions
  - no XML-/PNML mapping
  - manual creation of one class (25 lines, making the “pieces” know to Eclipse)

- GMF/EMF-editor integration
  - 45 @generated NOTs

These figures refer to ePNK version 0.9.0!

Code statistics

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- Petri Net Type: HLPNG plug-in
  - model for HLPNG-nets
  - Xtext grammar for concrete syntax
  - PNML-mapping: ca. 70 entries (+ Factory)
  - manual changes in generated classes: ca. 130
    (mostly functionality implementing type and sort resolution functions and helpers)
  - 1 OCL constraint, and 11 constraint classes (complex constraints)

These figures refer to ePNK version 0.9.0!

Statistics

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- Project contains
  - 20 eclipse plug-in projects (11 automatically generated)
  - 10 models (+ 1 grammar)
  - 125 model classes (and interfaces)
  - ca. 800 code classes
  - ca. 36.000 MLOC (> 50.000 TLOC)
  - ca. 220 “@generated NOT” tags
  - (guess < 2000 manual lines of code)

These figures refer to ePNK version 0.9.0!

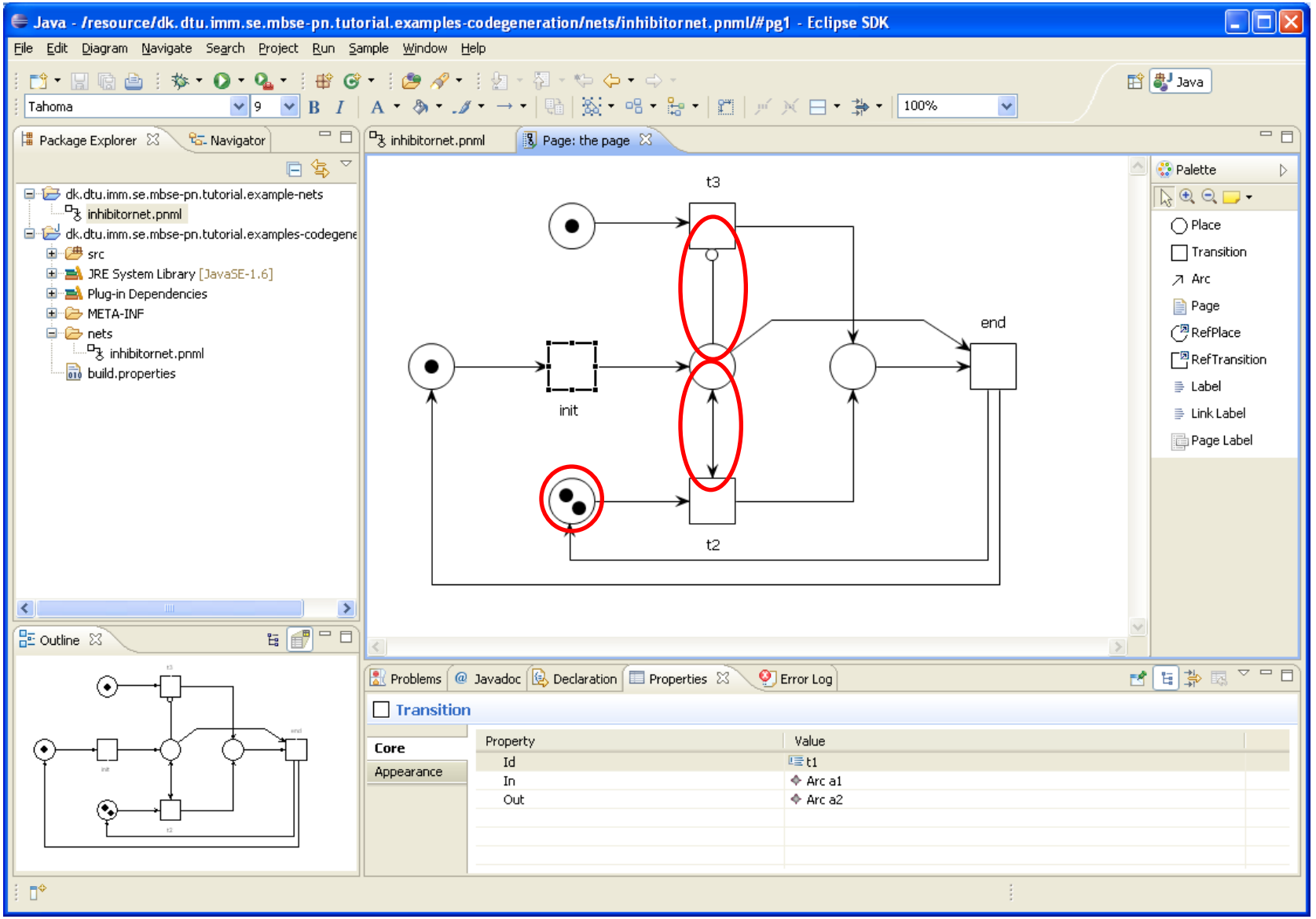
Outline

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- Motivation
- PNML
  - Overview
  - Core model
  - Type model
  - Mapping to XML
- Experiences and statistics
- ePNK: Tutorial

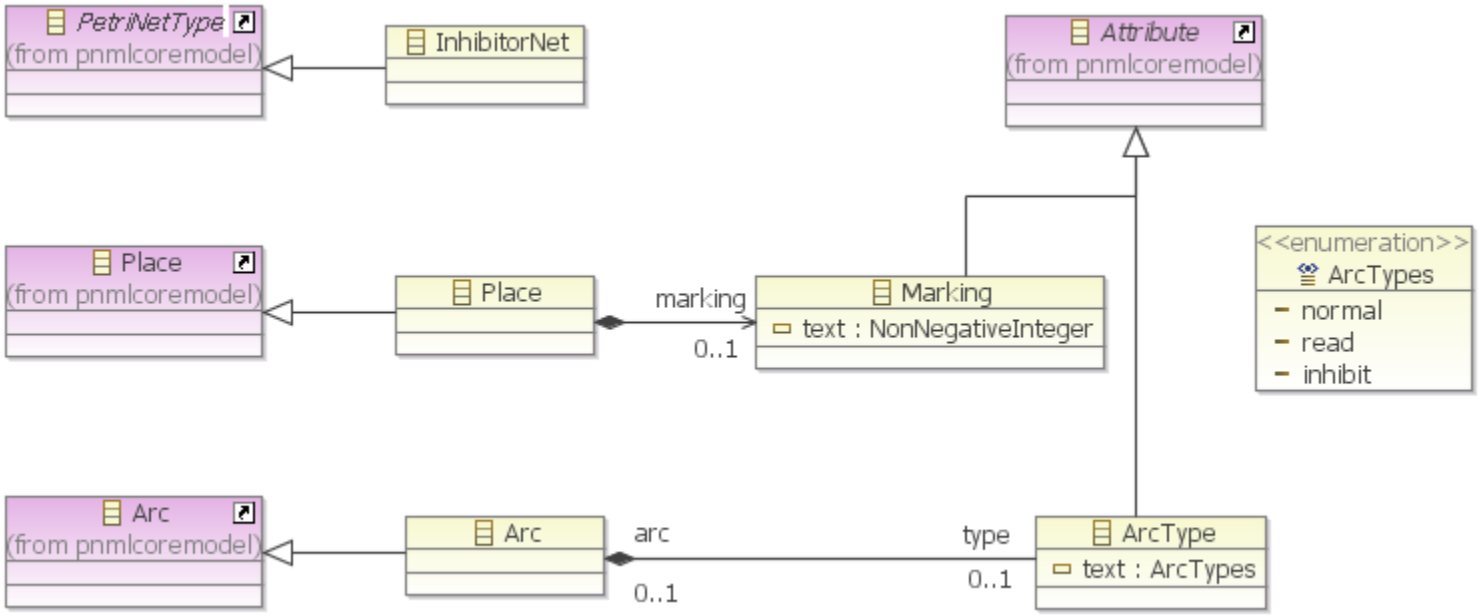
4. The ePNK: Tutorial

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



Define a Petri Net Type

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler




```
( self.source.oclIsKindOf(pnmlcoremodel::PlaceNode) and
  self.target.oclIsKindOf(pnmlcoremodel::TransitionNode) )
or
( self.source.oclIsKindOf(pnmlcoremodel::TransitionNode) and
  self.target.oclIsKindOf(pnmlcoremodel::PlaceNode) and
  not ( self.type.text = ArcTypes::inhibit ) )
```

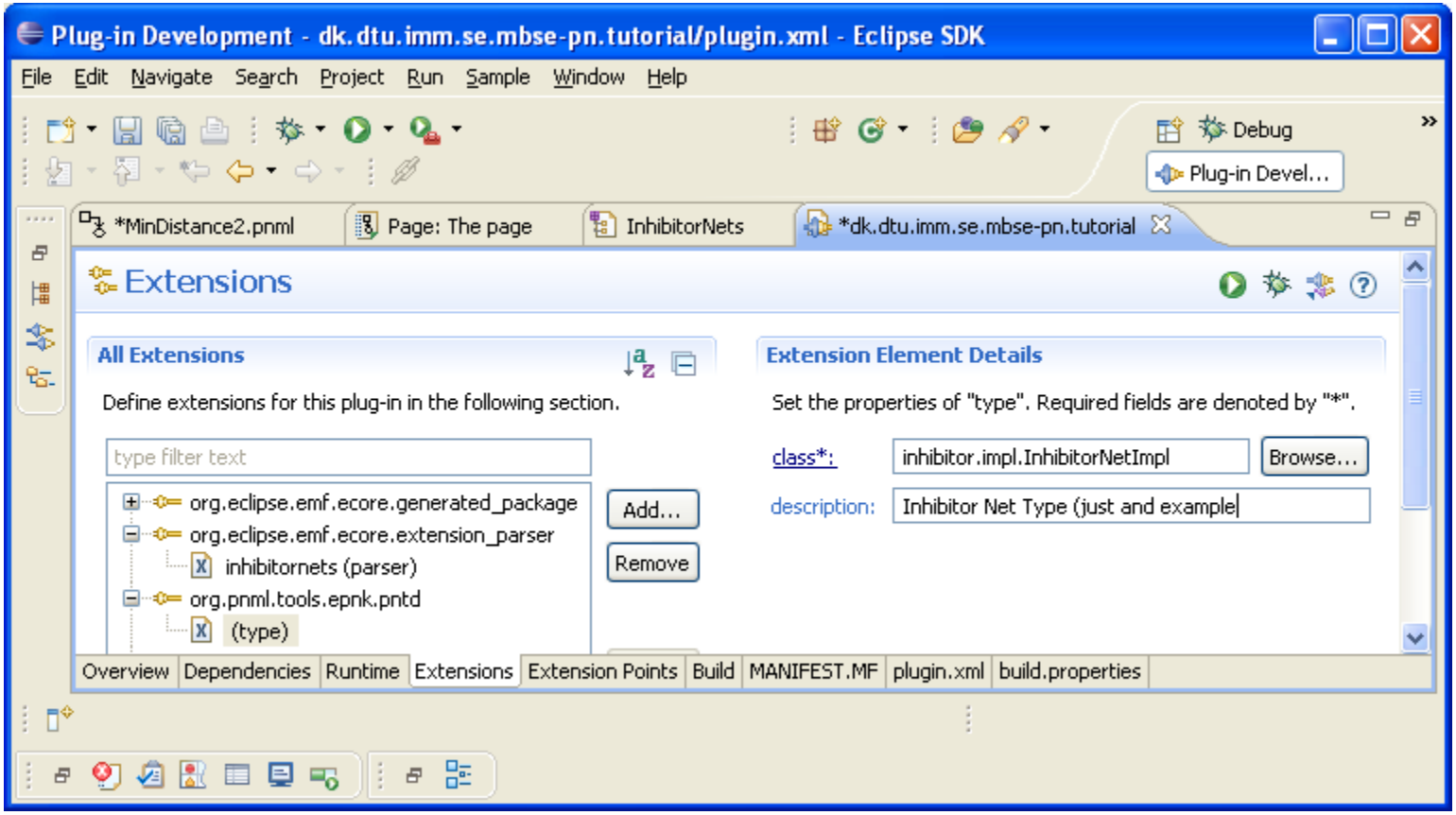
Change code manually

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

```
package inhibitornets.impl;

[…]

public class InhibitorNetImpl extends PetriNetTypeImpl implements
   InhibitorNet {

   public InhibitorNetImpl() {
        super();
   }

   protected EClass eStaticClass() {
        return InhibitornetsPackage.Literals.INHIBITOR_NET;
   }

   public String toString() {
        return "http://dk.dtu.imm.se.mbse-pn.tutorial.inhibitornet";

   }

}
```

Plug in net type

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



Plug in net type

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

```
<extension
    id="inhibitornets"
    name="Inhibitor Nets"
    point="org.pnml.tools.epnk.pntd">
    <type
        class=" inhibitornets.InhibitorNetImpl"
        description="Inhibitor nets">
    </type>
</extension>
```

Fully qualified (Java) class name

Plug in constraint

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

```
<extension
    point="org.eclipse.emf.validation.constraintProviders">

      [about 40 other boring but technically important lines]


  <![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
    not ( self.type.text = ArcTypes::inhibit ) )
  ]]>
     </constraint>
    </constraints>
  </constraintProvider>
</extension>
```

Programm Graphics

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

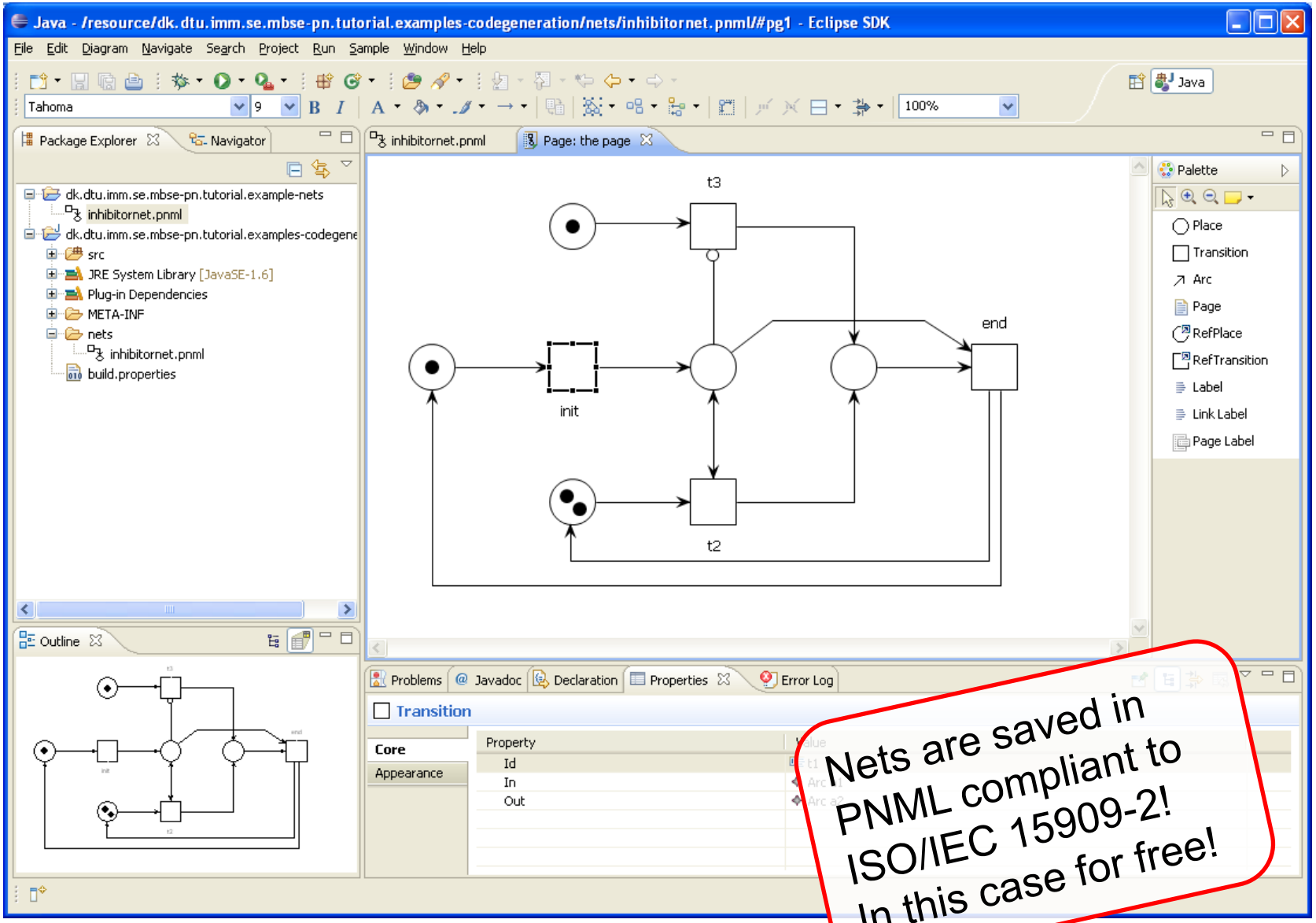
```
public class InhibitornetsArcFigure extends ArcFigure {
[...]
  private void setGraphics() {
    RotatableDecoration targetDecorator = null;
    RotatableDecoration sourceDecorator = null;

    if (type.equals(ArcTypes.NORMAL)) {
      targetDecorator = new ReisigsArrowHeadDecoration();

    } else if (type.equals(ArcTypes.INHIBIT)) {
      CircleDecoration circleDecoration =
        new CircleDecoration();
      circleDecoration.setLineWidth(this.getLineWidth());
      targetDecorator = circleDecoration;

    } else if (type.equals(ArcTypes.READ)) {
      targetDecorator = new ReisigsArrowHeadDecoration();
      sourceDecorator = new ReisigsArrowHeadDecoration();
[...]
```

Just a glimpse!

→ details Tutorial

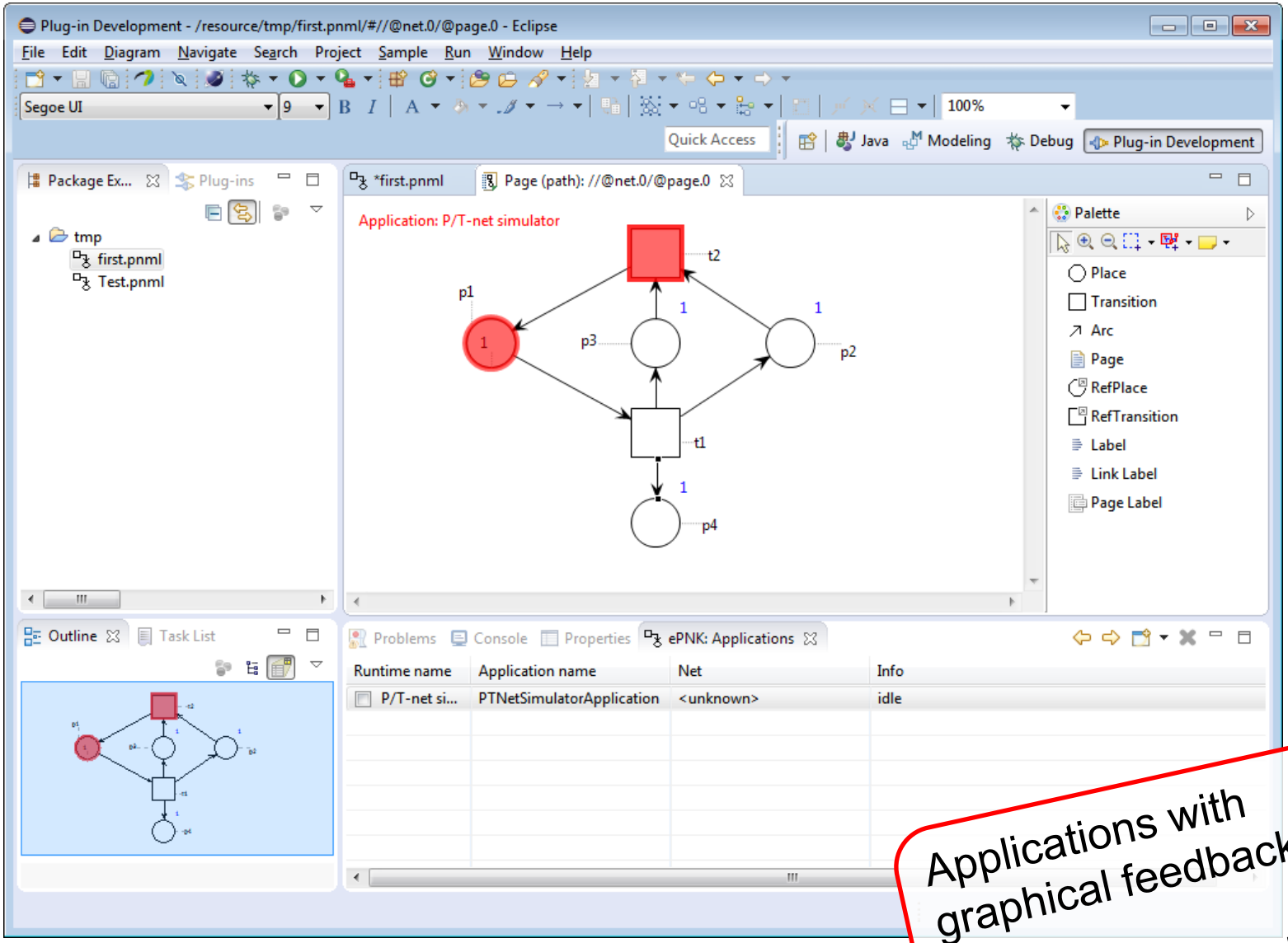
Result

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



Applications (on nets)

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



Applications with graphical feedback

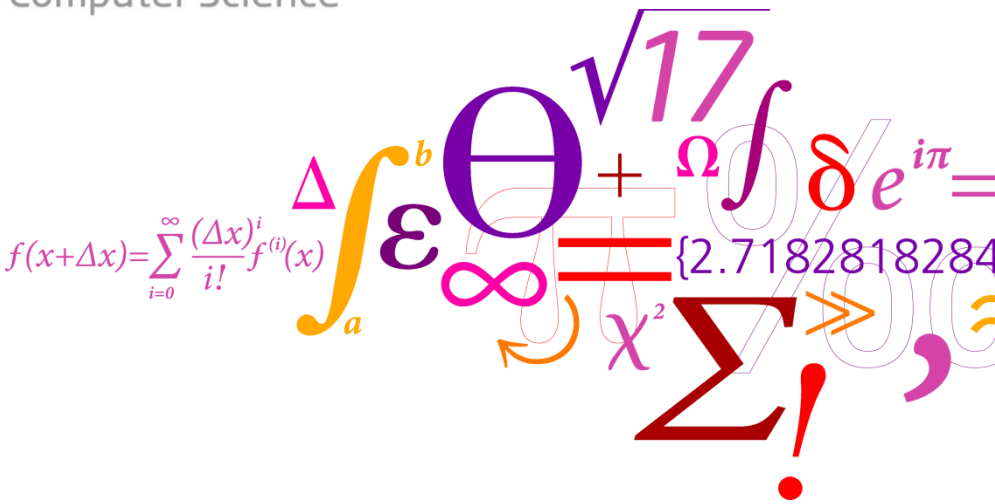
→ Details Tutorial 7 & 8

VI. Modelling Software Architecture

DTU Compute
Department of Applied Mathematics and Computer Science

By example: ePNK

→ Black board



4 Architectural Views

(after Sommerville)

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- Logical view: ... shows the abstractions in the system ...
- Development view: ... shows how the software is decomposed for the development ...
- Process view: ... how the system is composed of interaction processes at runtime ...
- Physical view: shows hardware and how software components are distributed across it ...

4 + 1 Architectural views

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- + Use cases

4 Architectural Views

(after Sommerville)

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- Logical view: ... shows the abstractions in the system ...
- **Development view: ... shows how the software is decomposed for the development ...**
- Process view: ... how the system is composed of interaction processes at runtime ...
- Physical view: shows hardware and how software components are distributed across it ...

Mostly "model" in the sense of MVC and domain model

By example: architecture of the ePNK

→ Black board