

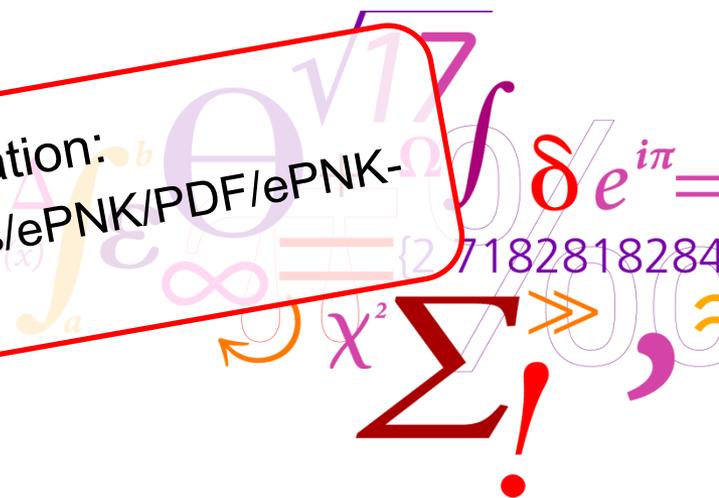
The ePNK: An extensible Petri net tool for PNML

Ekkart Kindler

DTU Informatics

Department of Informatics and Mathematical Modeling

For more details see ePNK documentation:
<http://www2.compute.dtu.dk/~ekki/projects/ePNK/PDF/ePNK-manual-1.0.0.pdf>



New Petri net type

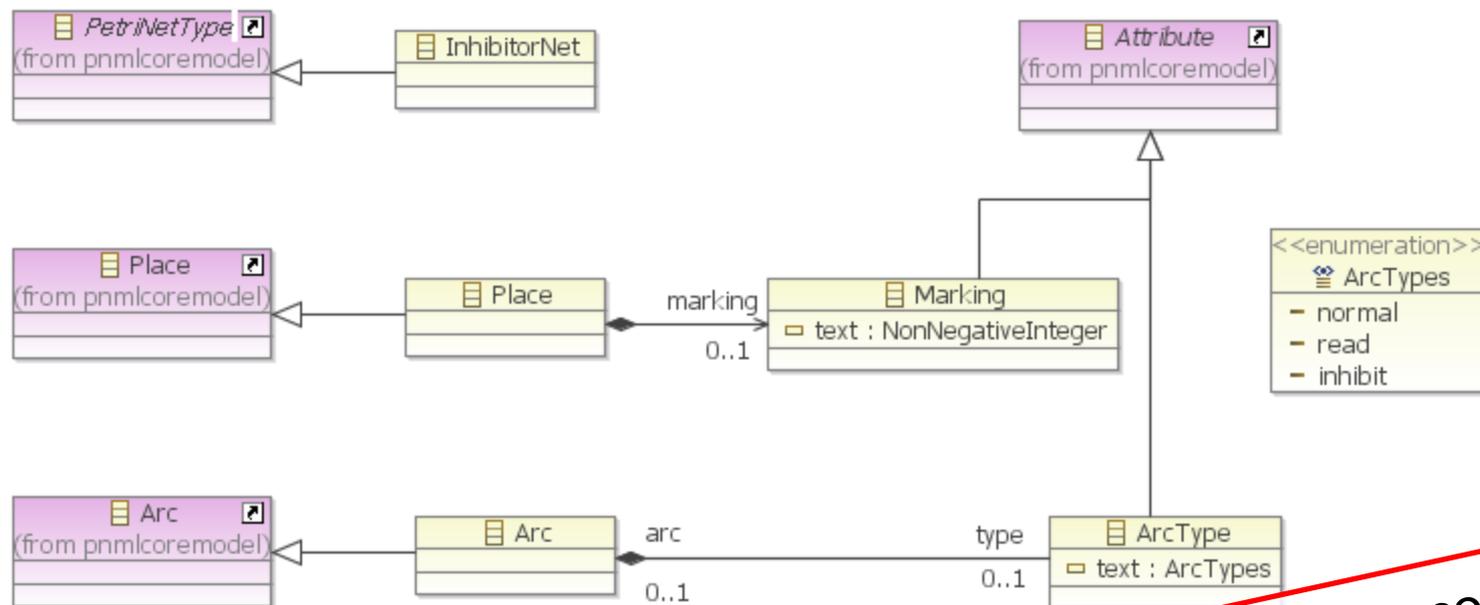
The screenshot displays the Eclipse IDE interface for editing a Petri net. The main editor shows a Petri net with the following components:

- Places:** A place with one token (top-left), a place with two tokens (bottom-left), and an empty place (middle-right).
- Transitions:** A dashed box labeled 'init', a transition labeled 't2' (circled in red), and a transition labeled 't3' (circled in red).
- Other Elements:** A transition labeled 'end' (a rectangle) and a page label 'Page: the page'.

The Properties window at the bottom shows the following data for the selected transition:

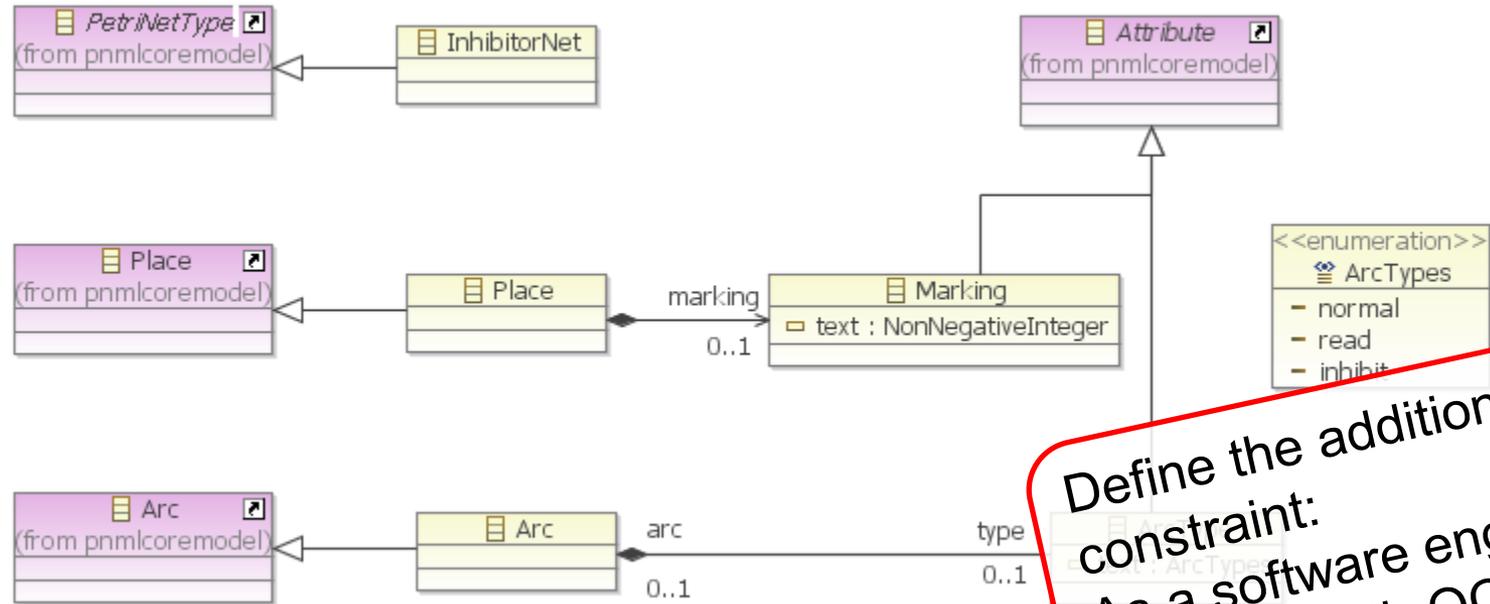
Core	Property	Value
Appearance	Id	t1
	In	Arc a1
	Out	Arc a2

What should it take to define this new Petri Net type conceptually?



Define the new concepts:
As a software engineer, I
do it that with a class
diagram

Define a Petri Net Type



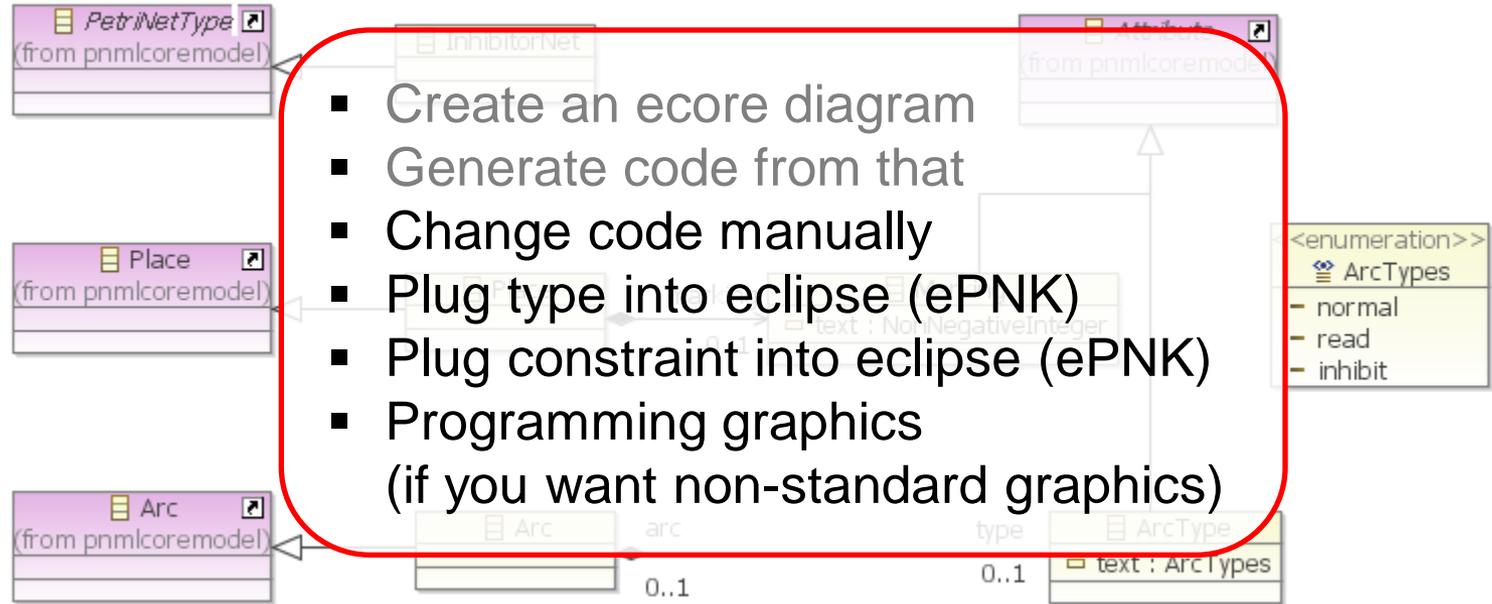
Define the additional constraint:
As a software engineer I do it that with OCL

```
( self.source.ocIsKindOf(pnmlcoremodel::PlaceNode) and  
  self.target.ocIsKindOf(pnmlcoremodel::TransitionNode) )
```

or

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

What does it take to implement it now?



```
( self.source.oclIsKindOf (pnmcoremodel::PlaceNode) and  
  self.target.oclIsKindOf (pnmcoremodel::TransitionNode) )
```

or

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

```
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";  
    }
```

```
}
```

Making constructor of generated code public is not so nice! Extend the class and make the constructor of the extended class public.

Plug-in Development - org.pnml.tools.epnk.extensions.tutorial.types/plugin.xml - Eclipse

File Edit Navigate Search Project Sample Run Window Help

Quick Access Java Modeling Debug Plug-in Development

ModuleGener... org.pnml.to... org.pnml.to... APetriNetEdi... org.pnml.to... »12

Extensions

All Extensions

Define extensions for this plug-in in the following section.

type filter text

- org.eclipse.emf.ecore.generated_package
- org.eclipse.emf.ecore.extension_parser
- org.pnml.tools.epnk.pntd**
 - ARCTYPESImpl (type)**
- org.pnml.tools.epnk.diagram.graphics

Add... Remove Up Down

Extension Element Details

Set the properties of 'type' Required fields are denoted by '*'.
class*: org.pnml.tools.epnk.extensions.tutorial.typ Browse...
description: Net with different arc types

Overview Dependencies Runtime Extensions Extension Points Build MANIFEST.MF **plugin.xml** build.properties

```
<extension
  id="org.pnml.tools.epnk.extensions.tutorial.types.arctypes"
  name="ArcTypes"
  point="org.pnml.tools.epnk.pntd">
  <type
    class="org.pnml.[...].tutorial.types.arctypes.arctypes.impl.ARCTYPESImpl"
    description="Net with different arc types">
  </type>
</extension>
<extension
  id="org.pnml.tools.epnk.extensions.tutorial.types.arctypes.graphics"
  name="ArcTypes Net graphical extensions"
  point="org.pnml.tools.epnk.diagram.graphics">
  <graphicsextension
    class="org.pnml.tools.epnk.[...].ArcTypesGraphicalExtension"
    description="Special graphics for arcs in this kind of net">
  </graphicsextension>
</extension>
```

```
<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>
```

Unfortunately, this is a bit technical
Constraints can also be programmed

```
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!

The screenshot displays the Eclipse IDE interface for editing a Petri net file named `inhibitornet.pnml`. The main workspace shows a Petri net with the following components:

- Places:** Three places on the left, each containing a different number of tokens (1, 2, and 3 respectively).
- Transitions:** Three transitions labeled `t1`, `t2`, and `t3`, and an `end` transition.
- Connections:** Directed arcs connect the places to the transitions and between transitions.

The right-hand side features a **Palette** with the following elements:

- Place
- Transition
- Arc
- Page
- RefPlace
- RefTransition
- Label
- Link Label
- Page Label

The bottom panel shows the **Properties** view for the selected transition `t1`:

Property	Value
Core	
Id	t1
Appearance	
In	Arc a1
Out	Arc a2

You can look up some details in the version of the ePNK that was deployed in the SE2 course:

```
org.pnml.tools.epnk.extensions.tutorial.types
```

```
org.pnml.tools.epnk.extensions.tutorial.types.edit
```

In order to see these projects to your workspace:

- Open “Plug-ins” view
- Select the resp. plug-in project(s)
- Right-click and choose
Import As → Source Project