Model-Based Software Engineering

Lecture 02 – Metamodeling

Prof. Dr. Joel Greenyer

April 12, 2016

Learning Objectives

- Understanding the principle of metamodeling
- Understanding the principles of creating modeling languages
- Knowing important terms and concepts
  - formal languages
  - models and metamodels
  - meta levels
  - other relationships between models
- Application of metamodeling techniques in metamodeling frameworks

2.1. Formal languages and metamodeling

Metamodel and Metamodeling

- **Metamodeling** is the process of defining rules and constraints for creating models for a certain class of problems

- A **metamodel** defines rules and constraints for creating models

- Other definitions of **metamodel:**
  - a metamodel is the model of a model
  - "A metamodel is a model used to model modeling itself" (MOF 2.5)
  - A metamodel defines a formal modeling language

Formal Languages in Computer Science

- Defining **formal languages**, for example programming languages, is an established discipline of computer science
  - see for example also lecture “compiler construction”

- A **formal language definition** contains the definition of
  - the **abstract syntax**: defines its internal structure
    - Defines the language constructs and how they can be combined
  - the **concrete syntax**: defines its **notation,** its visual representation for the user (textual or graphical)
  - the **semantics**: defines the meaning of the language constructs and their combinations
  - (sometimes also) the **serialization syntax**: how are sentences of the language stored or exchanged by tools

Formal Languages cont.

- In the 1960s, John Backus and Peter Naur invented the Backus Naur Form (BNF)
  - used for defining the syntax of Algol 60
  - in the form of a **context-free grammar**
  - there is now also the **extended BNF** (EBNF) and and augmented BNF (ABNF)
- BNF is a **meta-language**, a language for defining languages

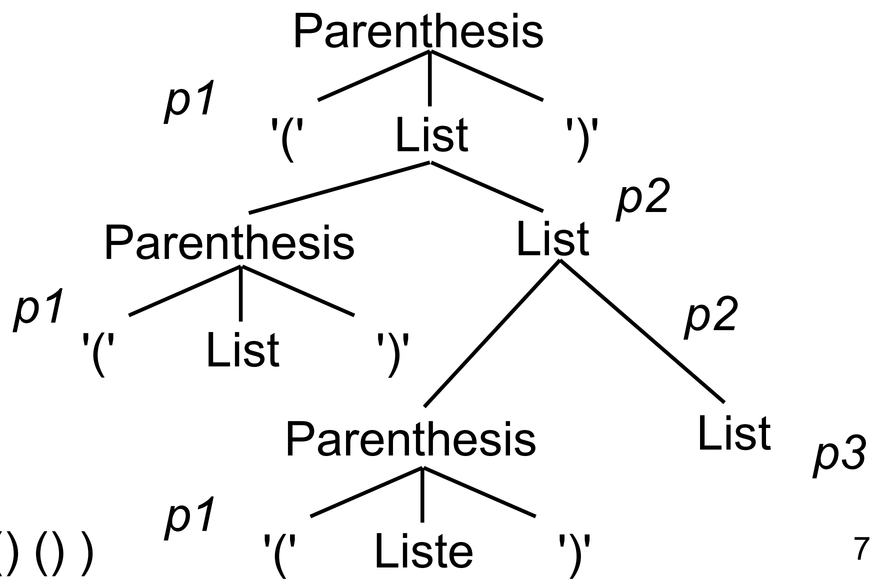
Context-free Grammars

- A **context-free grammar** describes a language
  - a **language** is a set of sentences
  - a **sentence** is a sequence of **terminals**
- For each sentence described by a context-free grammar, there is a **derivation tree** (syntax tree) that shows how the sentence can be derived by a applying **production rules**

Example:

terminals	T = { (,) }
non-terminals	N = {Parenthesis, List}
start symbol	S = Parenthesis

```
Productions       P = {
    Parenthesis ::= '(' List ')'
    List        ::= Parenthesis List
    List        ::=
}
```



Sentence: (() ())

XML and DTDs

- Similarly, DTDs and XML-Schema are meta-languages that define XML-based languages

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE adressBuch SYSTEM "adressBuch.dtd">
<?xml-stylesheet type="text/xsl" href="adressBuch.xsl"?>
<adressBuch>
    <adresse>
      <name vorname="Joel" nachname="Greenyer"/>
      <anschrift art="dienstlich">
         <strasse>Welfengarten 1</strasse>
         <ort>Hannover</ort>
         <plz>30167</plz>
      </anschrift>
    </adresse>
</adressBuch>
```

XML address book document

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT adressBuch (adresse)*>
<!ELEMENT adresse (name, anschrift)>
<!ELEMENT name EMPTY>
<!ATTLIST name vorname CDATA #IMPLIED
               nachname CDATA #REQUIRED>
<!ELEMENT anschrift (strasse, ort, plz)>
<!ATTLIST anschrift art
                (privat|dienstlich) #REQUIRED>
<!ELEMENT strasse (#PCDATA)>
<!ELEMENT ort (#PCDATA)>
<!ELEMENT plz (#PCDATA)>
```

DTD defining valid address book documents

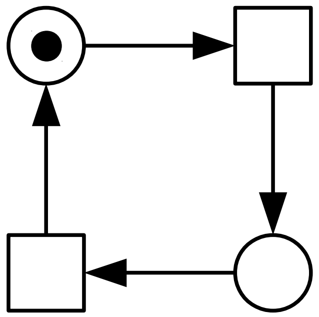
So what's new about metamodeling?

- In computer science, we defined formal languages for more than half a century
  - So what is new about metamodeling?
- Metamodeling and traditional definition of formal languages have a lot in common
  - definition of abstract/concrete syntax, semantics
- **Metamodeling** uses **rich techniques** based on **object-oriented modeling concepts** (related to UML)
- Modern metamodeling techniques were developed also in the effort to give a formal language definition for UML

2.2. Metamodels by example

Metamodeling Example: Define a Petri Net Language

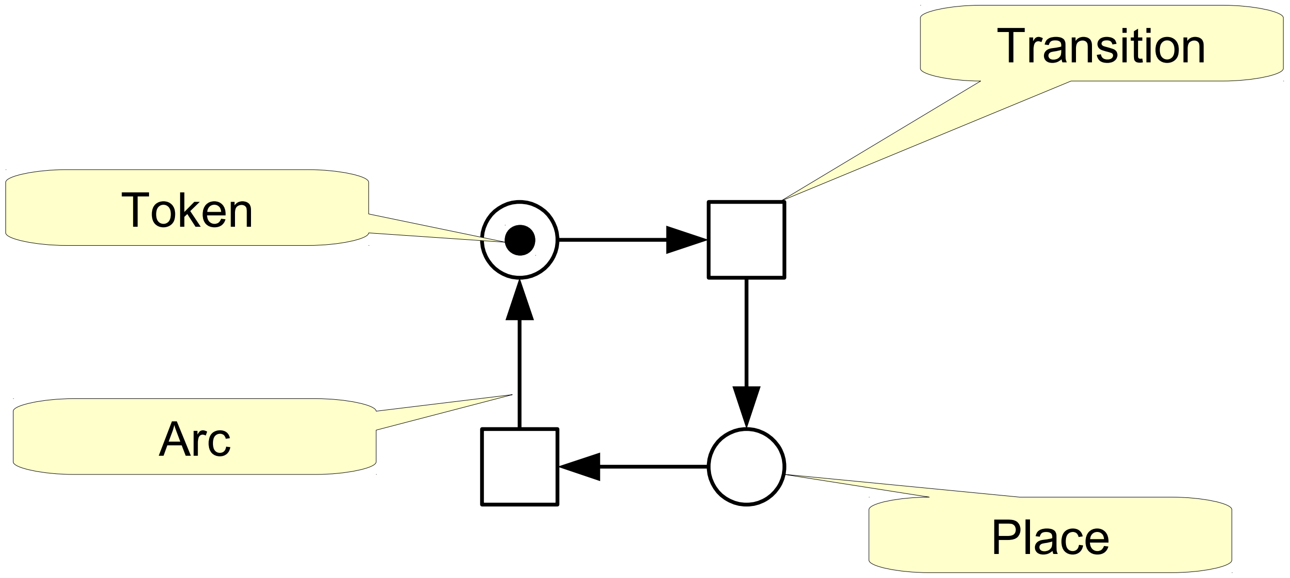
- How would you define a model for modeling Petri nets?
  - How would you metamodel Petri nets?



The Petri Net example on the following slides is based on the lecture “Course on Advanced Topics in Software Engineering” by Prof. Dr. Ekkart Kindler, Denmark Technical University, 2015.

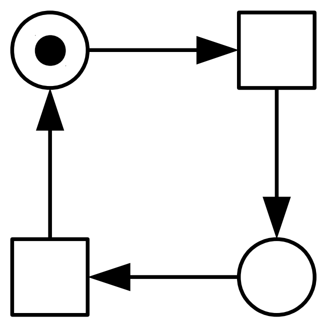
Object-Oriented Modeling Approach

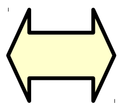
- What are the objects that we see here?

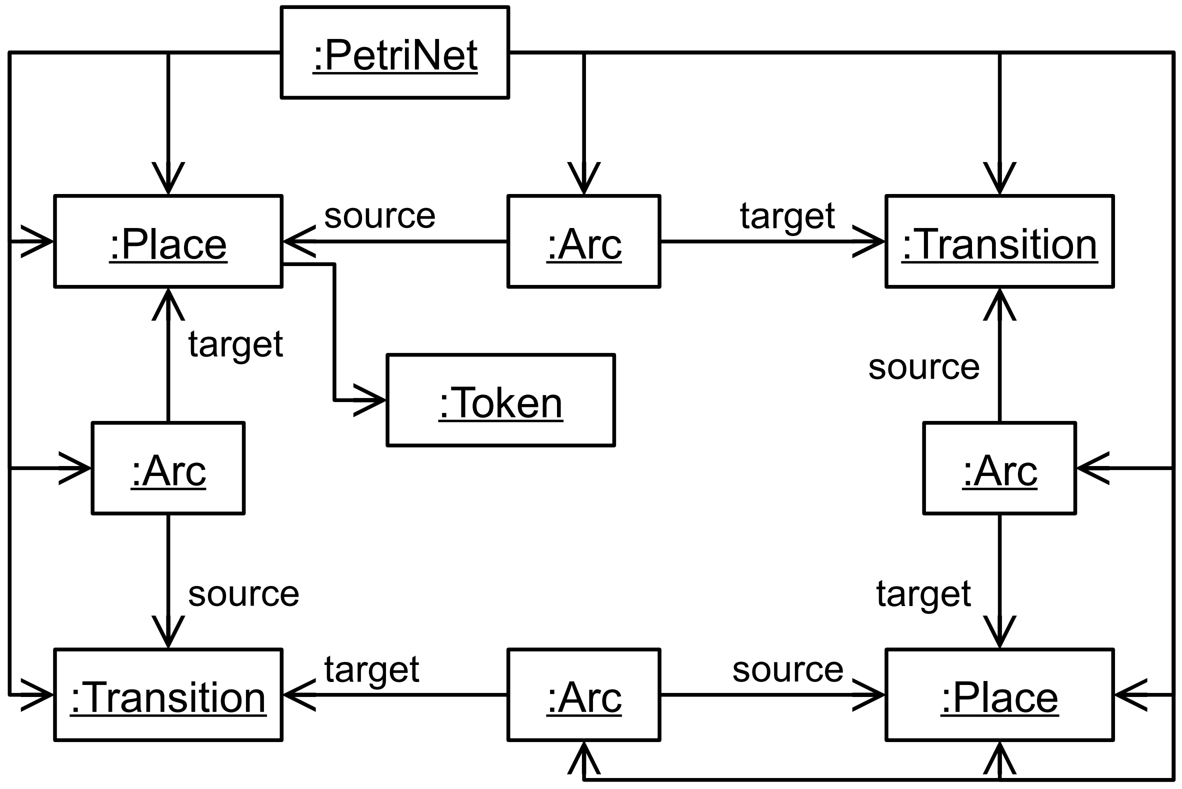


Object-Oriented Modeling Approach

- **Step 1**: Understand a model as a **structure of objects**
- For the example:





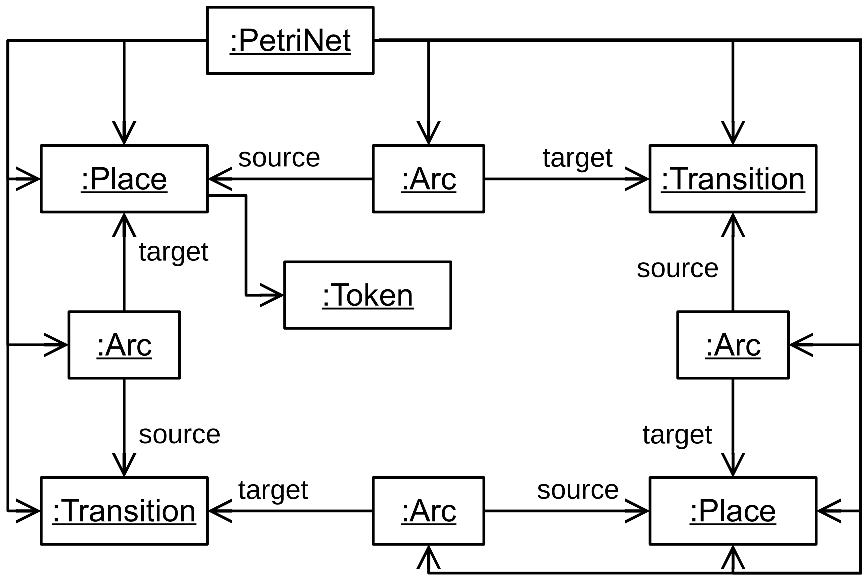


concrete syntax
(representation to the user)

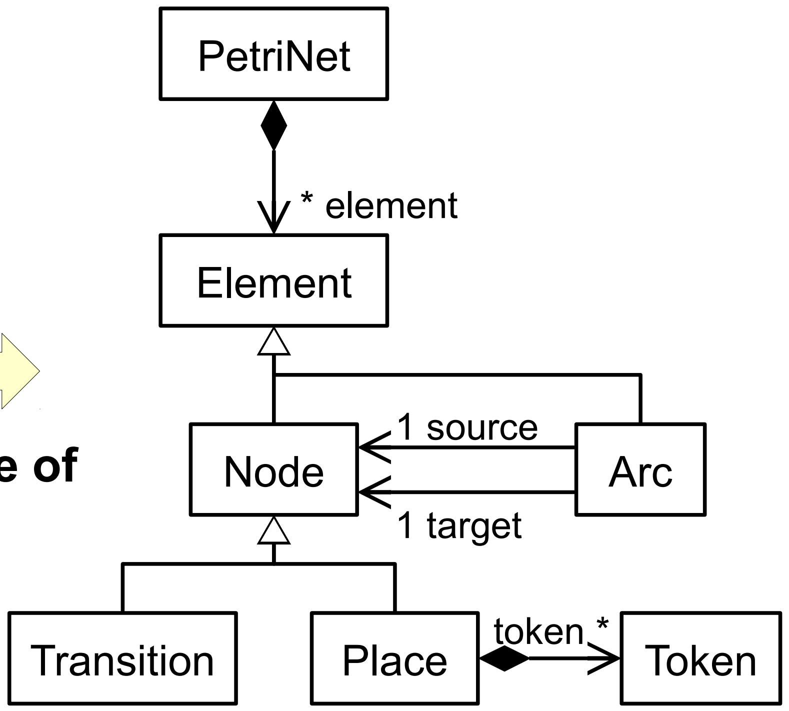
abstract syntax
(internal structure, occurrences of language constructs and their relationships)

Object-Oriented Modeling Approach

- **Step 2**: Create a model for all valid Petri nets (all object structures that represent valid Petri nets)

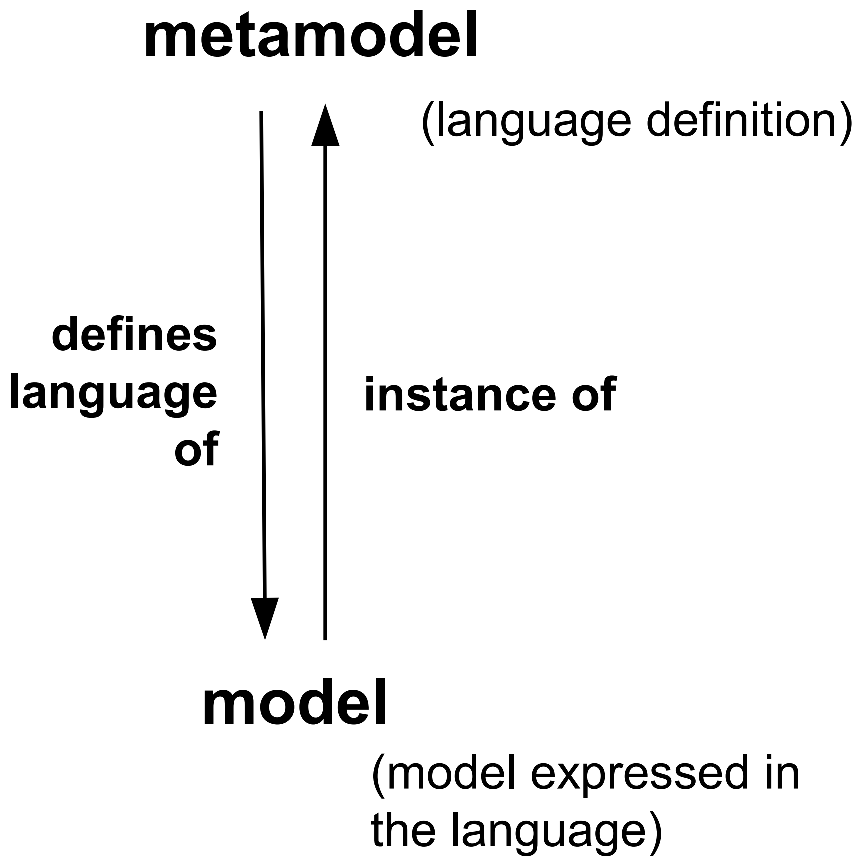


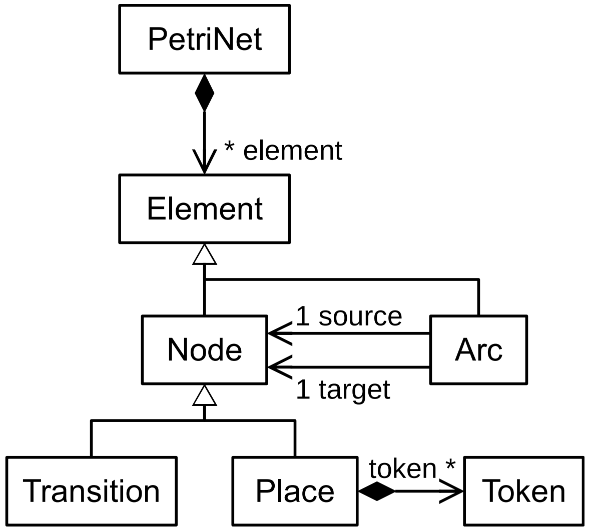
instance of

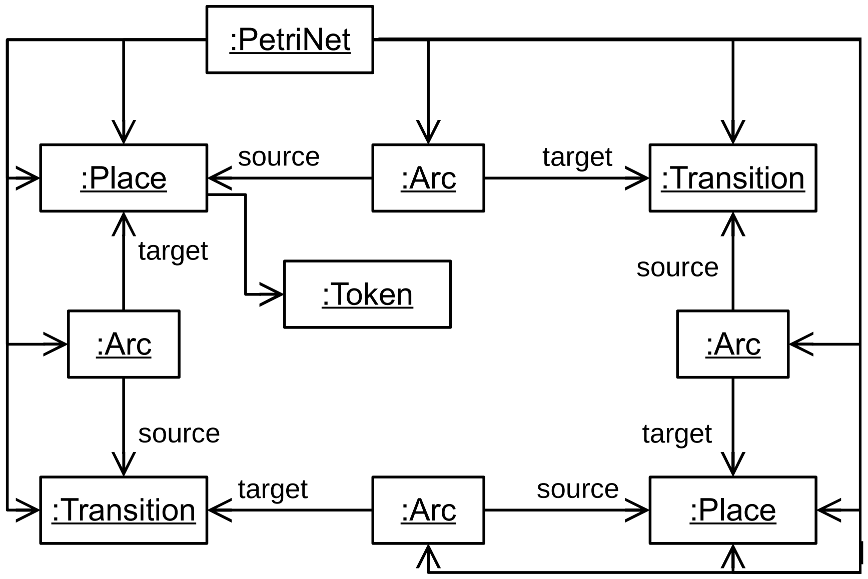


object model **class model**

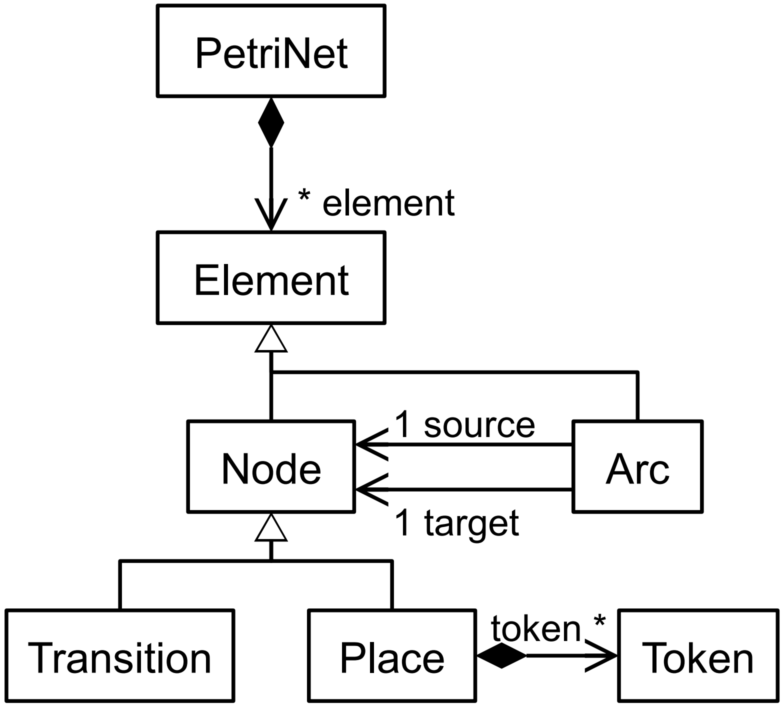
Model and Metamodel



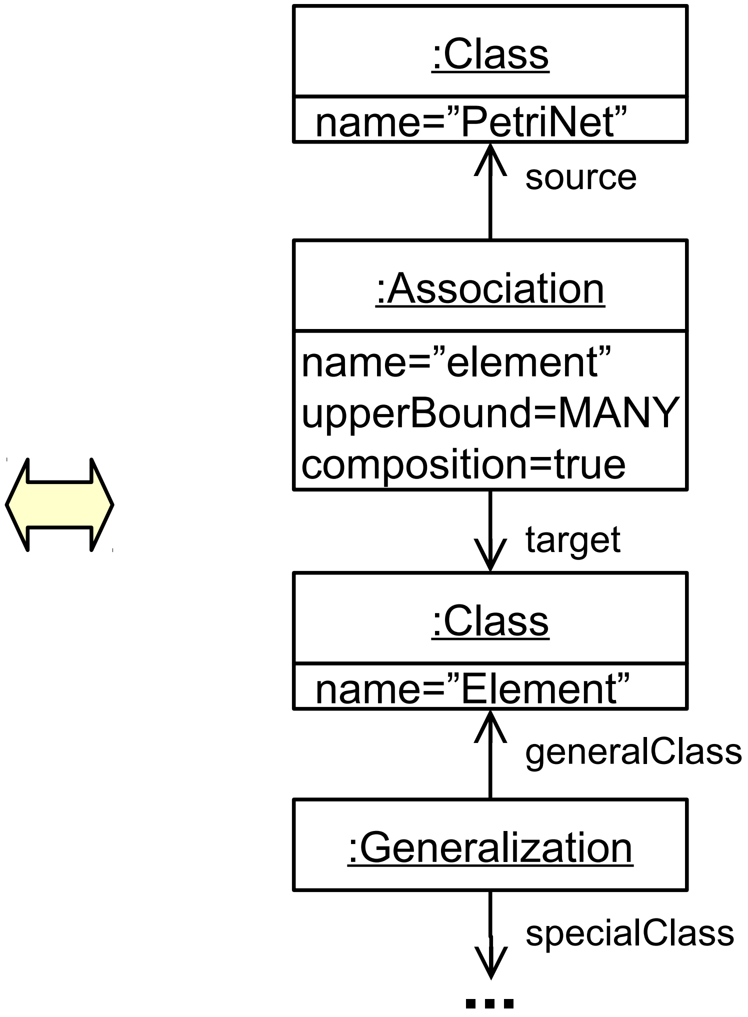




Class models are models, too!

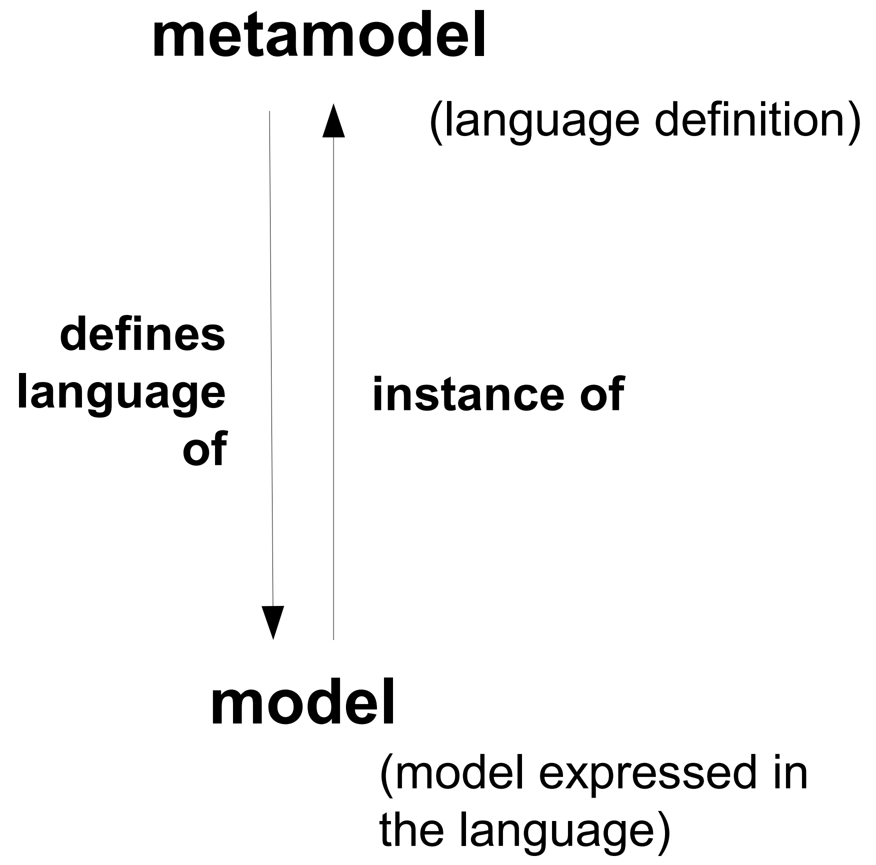


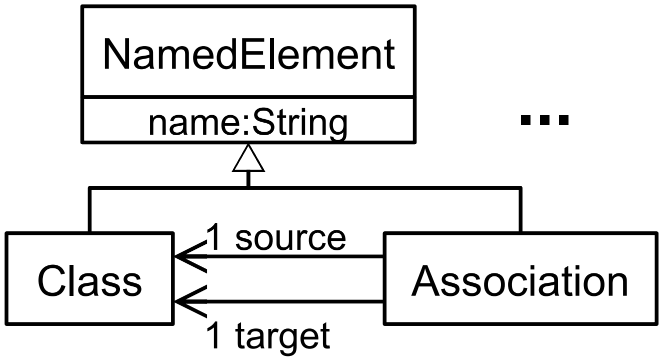
concrete syntax

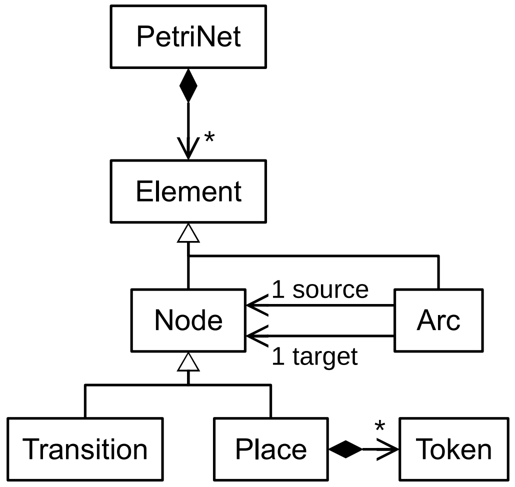


abstract syntax

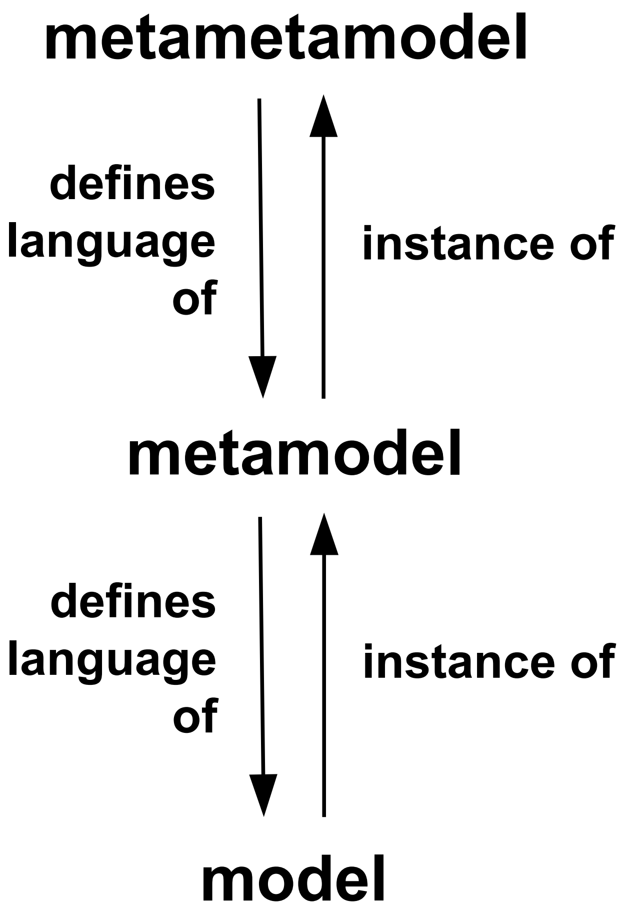
Model and Metamodel (Class models are models, too!)

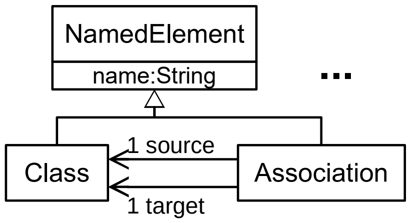


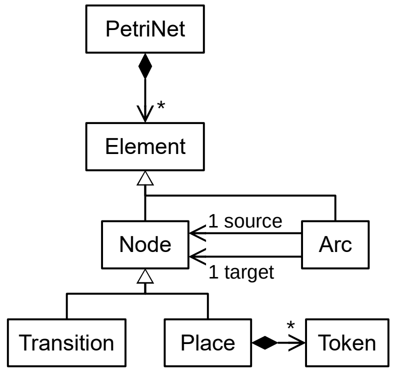


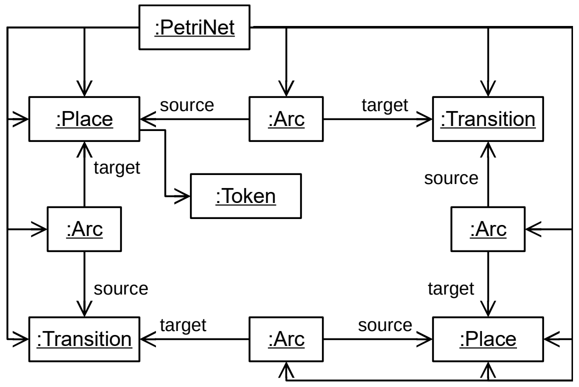


Multiple Meta-Levels



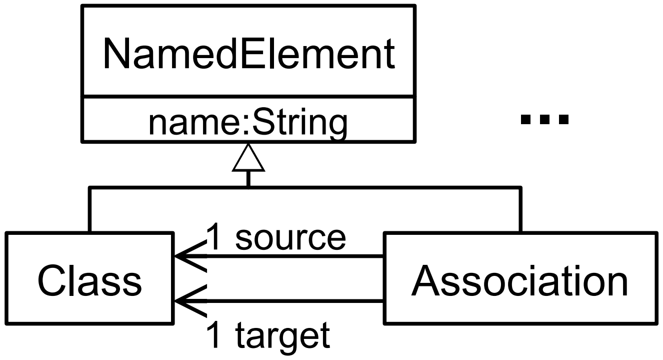






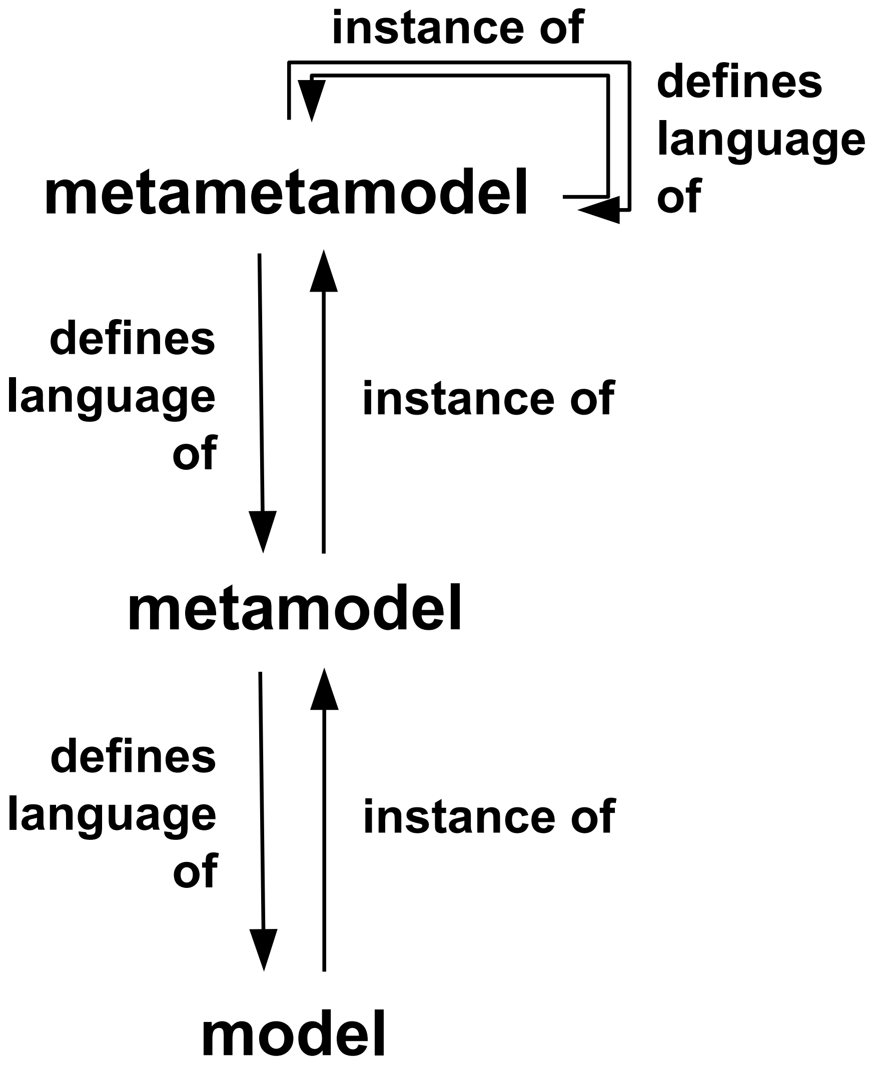
Meta-Levels

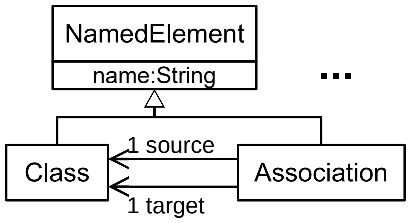
- Is there also a metamodel for this model?

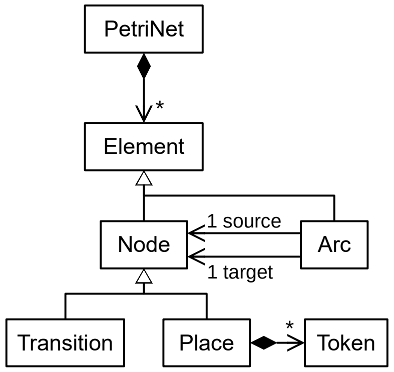


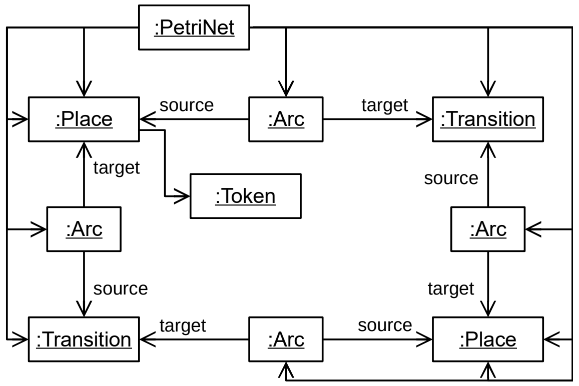
- **It can describe itself!**

Meta-Levels


instance of
defines language of
metametamodel
defines language of
instance of
metamodel
defines language of
instance of
model


NamedElement
name:String
...
Class
Association
1 source
1 target


PetriNet
*
Element
Node
Arc
1 source
1 target
Transition
Place
*
Token


:PetriNet
:Place
source
:Arc
target
:Transition
target
:Token
source
:Arc
:Arc
source
target
:Transition
target
:Arc
source
:Place

2.3. Meta-levels

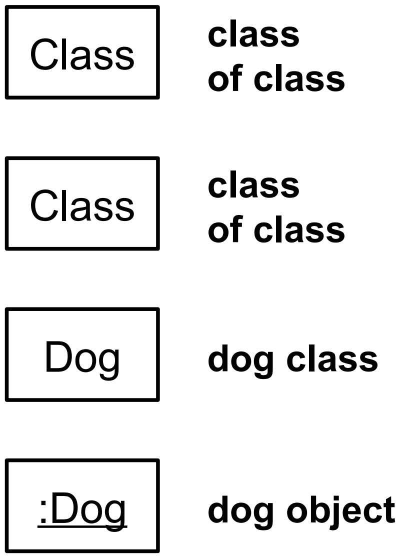
Typical Meta-Level Descriptions

- Sometimes, we refer to the **four meta-levels** (M0-M3) originally defined by the MOF standard
  - MOF: Meta-Object Facility, standard by the OMG (see http://www.omg.org/mof/)

M3	**meta-metamodel** to define metamodels on M2, also describes itself
M2	**metamodels**, for defining a modeling language on M1
M1	**models** of data or processes
M0	**instance-model,** concrete data

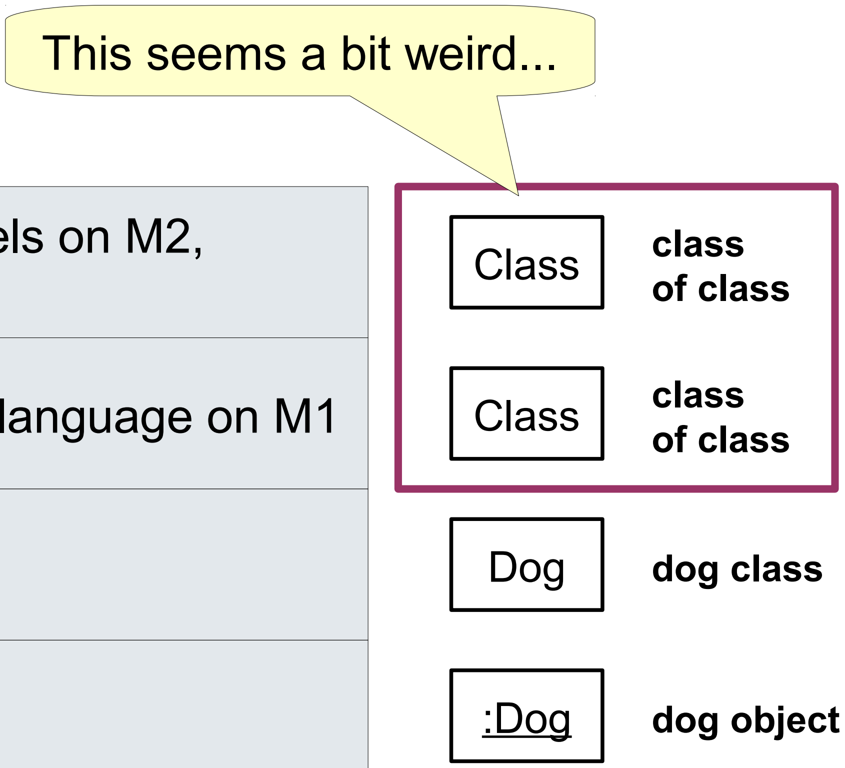
Meta-Levels for UML

M3	**meta-metamodel** to define metamodels on M2, also describes itself
M2	**metamodels**, for defining a modeling language on M1
M1	**models** of data or processes
M0	**instance-model,** concrete data



Meta-Levels for UML

M3	**meta-metamodel** to define metamodels on M2, also describes itself
M2	**metamodels**, for defining a modeling language on M1
M1	**models** of data or processes
M0	**instance-model,** concrete data



Meta-Object Facility (MOF)

- The MOF standard defines a meta-metamodel that is used to define UML as well as other languages defined by the OMG
- The MOF meta-metamodel is similar to the UML metamodel part that defines UML class diagrams
- You could use UML class diagrams to describe UML
  - in fact, this is also done
- But the MOF meta-model is more concise that UML
  - UML also defines Activity Diagrams, Sequence Diagrams, …
  - this is not necessary to define other meta-models

Taking a quick look at the OMG standards: MOF

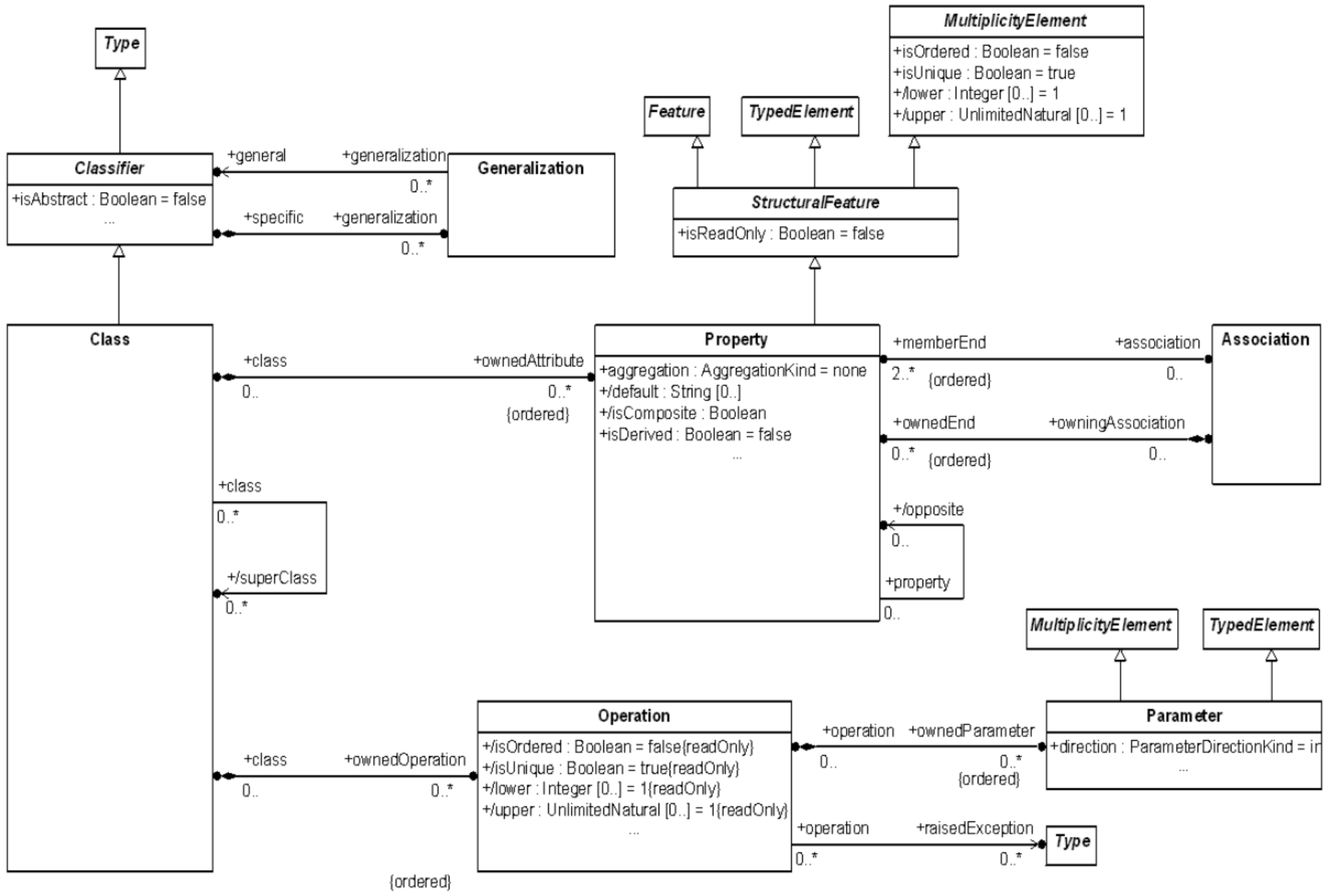
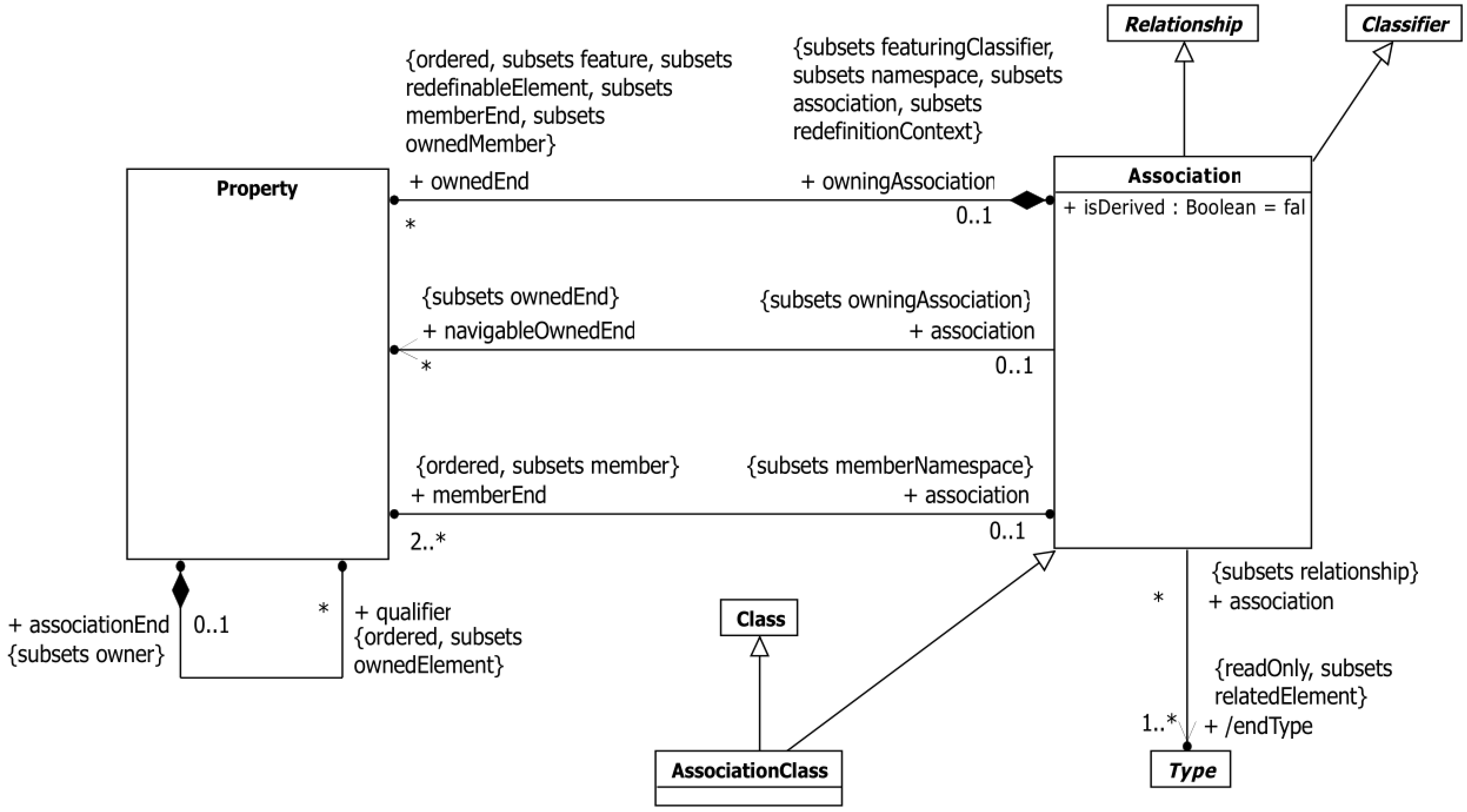


Figure 12.2 - EMOF Classes

(from http://www.omg.org/spec/MOF/2.5/PDF)

Taking a quick look at the OMG standards: UML



Relationship
Classifier
{ordered, subsets feature, subsets redefinableElement, subsets memberEnd, subsets ownedMember}
{subsets featuringClassifier, subsets namespace, subsets association, subsets redefinitionContext}
Property
+ ownedEnd
+ owningAssociation
Association
+ isDerived : Boolean = fal
*
0..1
{subsets ownedEnd}
+ navigableOwnedEnd
{subsets owningAssociation}
+ association
*
0..1
{ordered, subsets member}
+ memberEnd
{subsets memberNamespace}
+ association
2..*
0..1
+ associationEnd
{subsets owner}
0..1
*
+ qualifier
{ordered, subsets ownedElement}
Class
AssociationClass
*
{subsets relationship}
+ association
{readOnly, subsets relatedElement}
1..*
+ /endType
Type

Figure 11.25 Associations

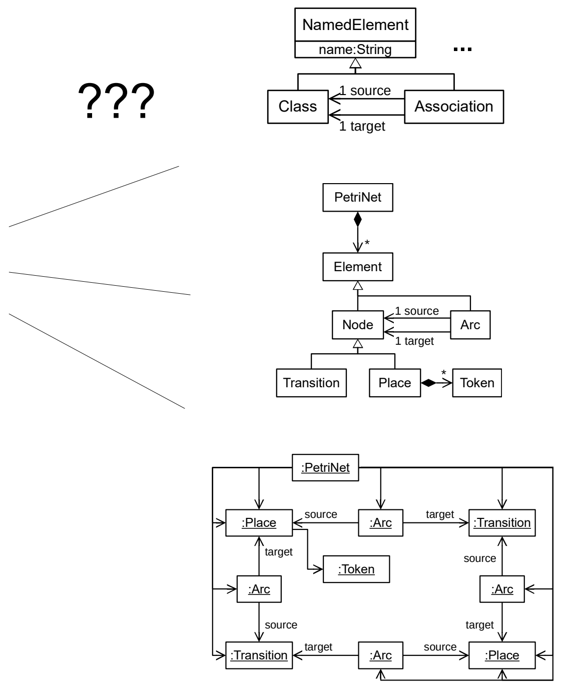
(from http://www.omg.org/spec/UML/2.5/PDF)

Meta-Levels for the Petri net

- sometimes it is difficult to allocate models and metamodels to the “four” meta-levels
  - sometimes there are more, sometimes less levels
  - this was reason for A LOT of discussions already!

M3	**meta-metamodel** to define metamodels on M2, also describes itself
M2	**metamodels**, for defining a modeling language on M1
M1	**models** of data or processes
M0	**instance-model,** concrete data

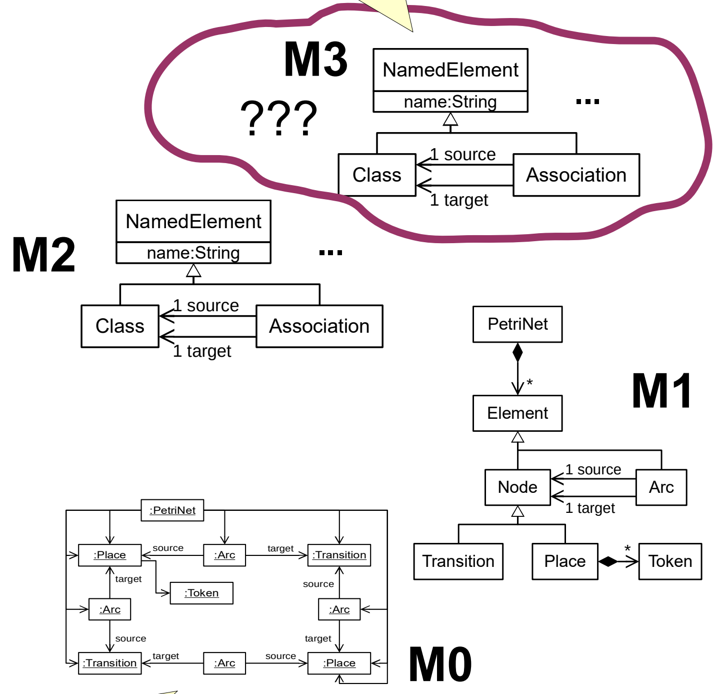
???



Meta-Levels for the Petri net

This seems artificial...

M3	**meta-metamodel** to define metamodels on M2, also describes itself
M2	**metamodels**, for defining a modeling language on M1
M1	**models** of data or processes
M0	**instance-model**, concrete data

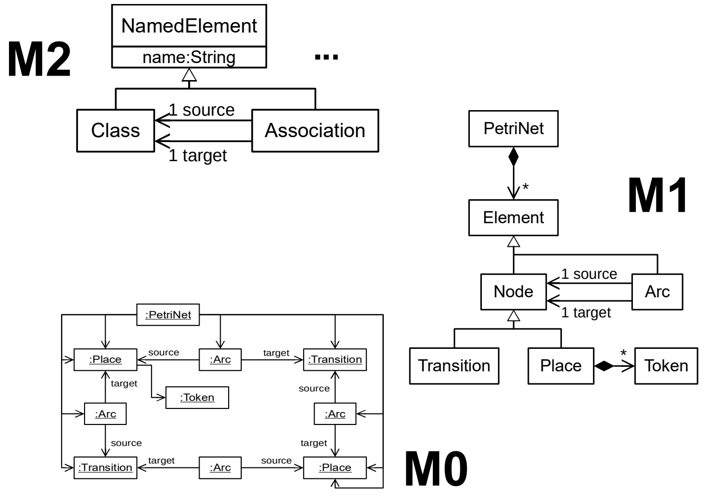


concrete instance of a Petri net
(e.g. diagram in an editor)

Meta-Levels for the Petri net

Maybe three meta levels are sufficient here...

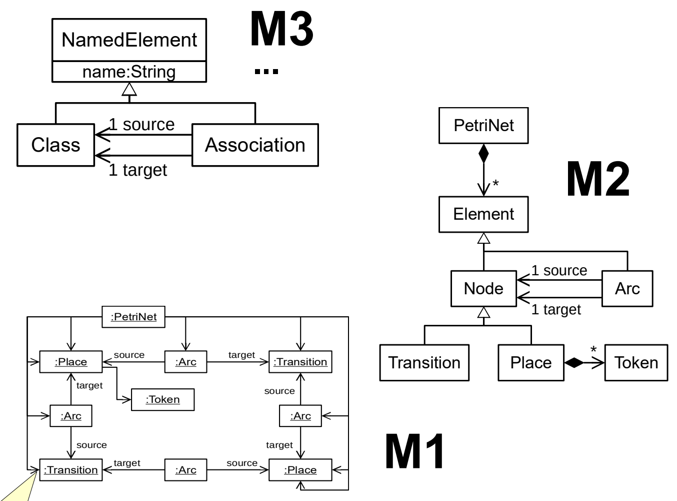
M2	**metamodel/meta-metamodel**
M1	**models** of data or processes
M0	**instance-model,** concrete data



Meta-Levels for the Petri net

Fits better into M1-M3

M3	**meta-metamodel** to define metamodels on M2, also describes itself
M2	**metamodels**, for defining a modeling language on M1
M1	**models** of data or processes
M0	**instance-model,** concrete data

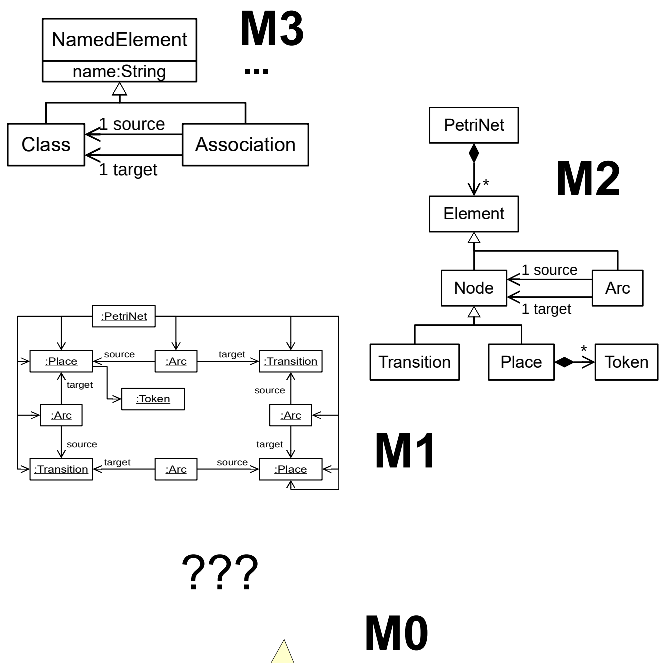


Especially: a Petri net is a *model of a process* – so, by definition of M1, it fits nicely in M1!

Meta-Levels for the Petri net

Fits better into M1-M3

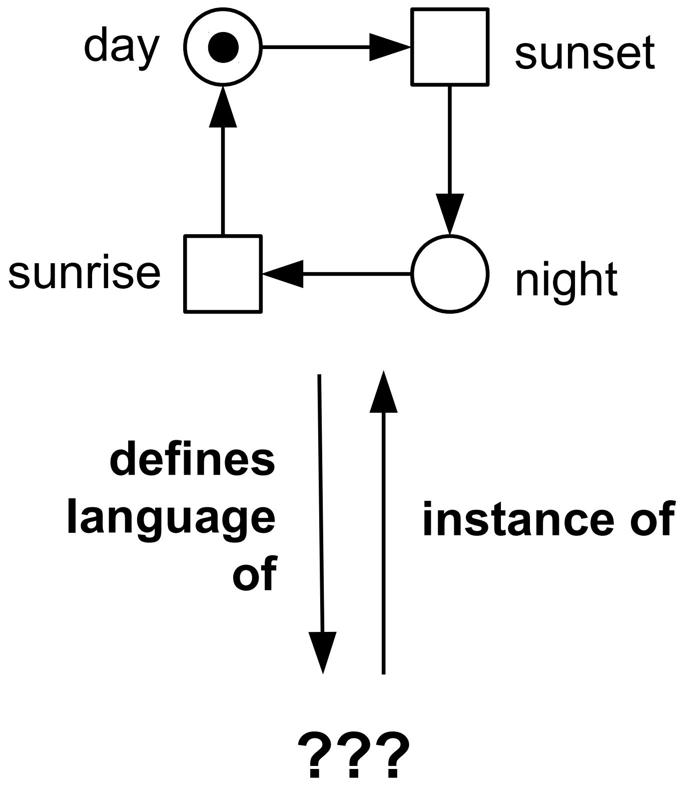
M3	**meta-metamodel** to define metamodels on M2, also describes itself
M2	**metamodels**, for defining a modeling language on M1
M1	**models** of data or processes
M0	**instance-model**, concrete data



but what is on M0 then?

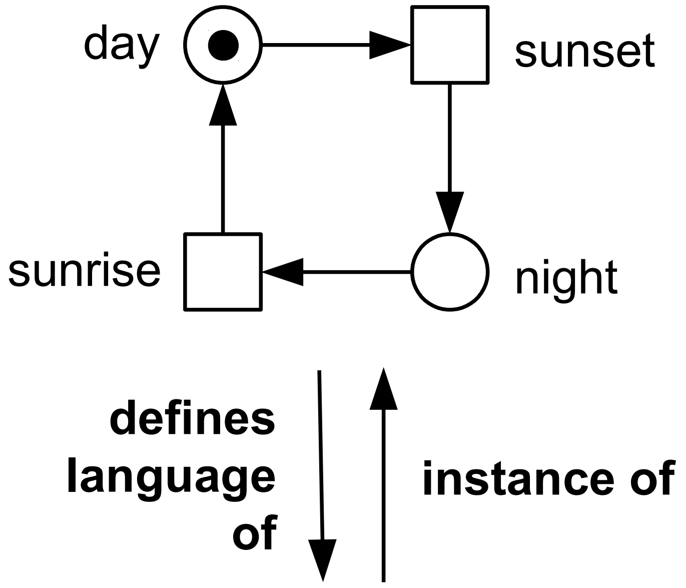
Meta-Levels for the Petri net

- If a Petri net is a model of a process..
- … then what is the instance of that model?



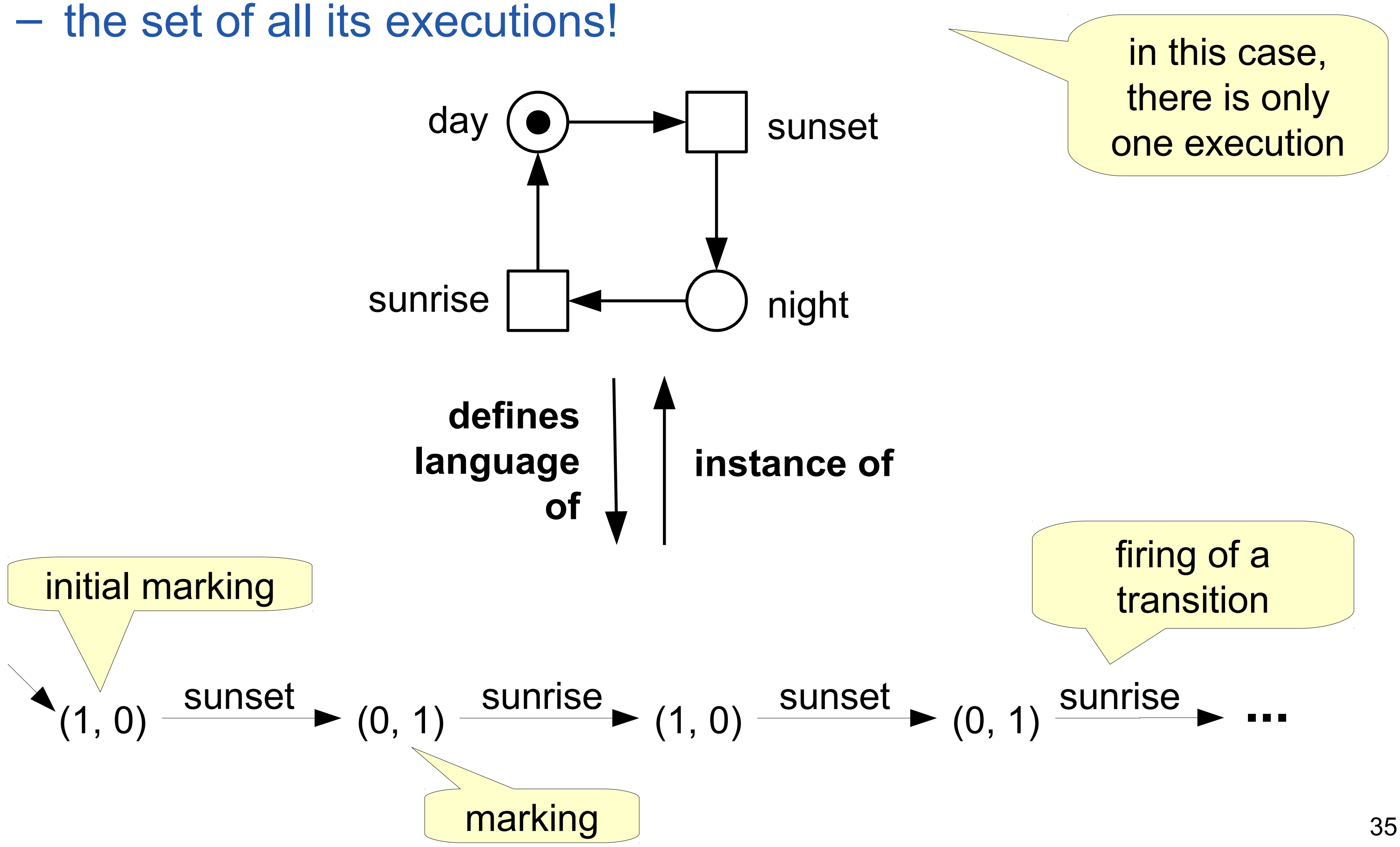
Meta-Levels for the Petri net

- What is the **language defined by** a Petri net?



Meta-Levels for the Petri net

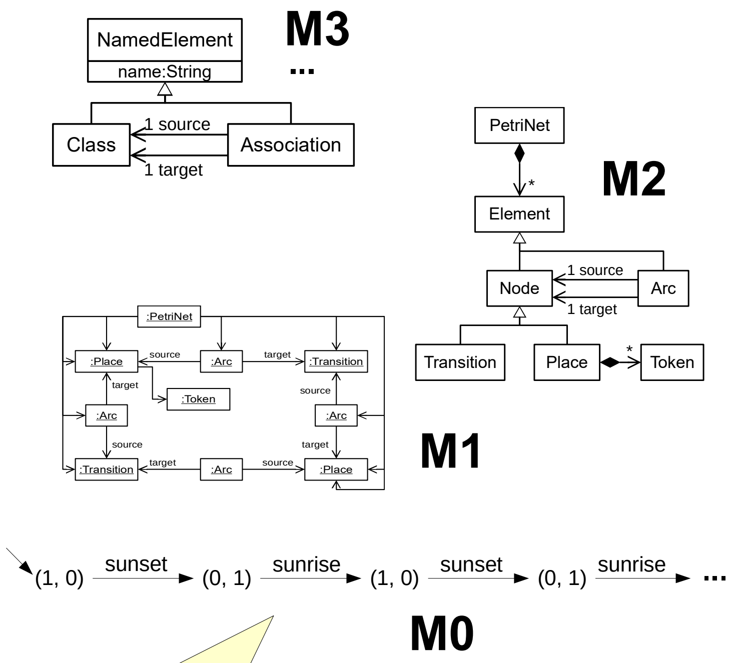
- What is the language defined by a Petri net?
  - the set of all its executions!


in this case, there is only one execution
day
sunset
sunrise
night
defines language of
instance of
initial marking
firing of a transition
(1, 0) sunset (0, 1) sunrise (1, 0) sunset (0, 1) sunrise ...
marking

SOFTWARE ENGINEERING

Meta-Levels for the Petri net

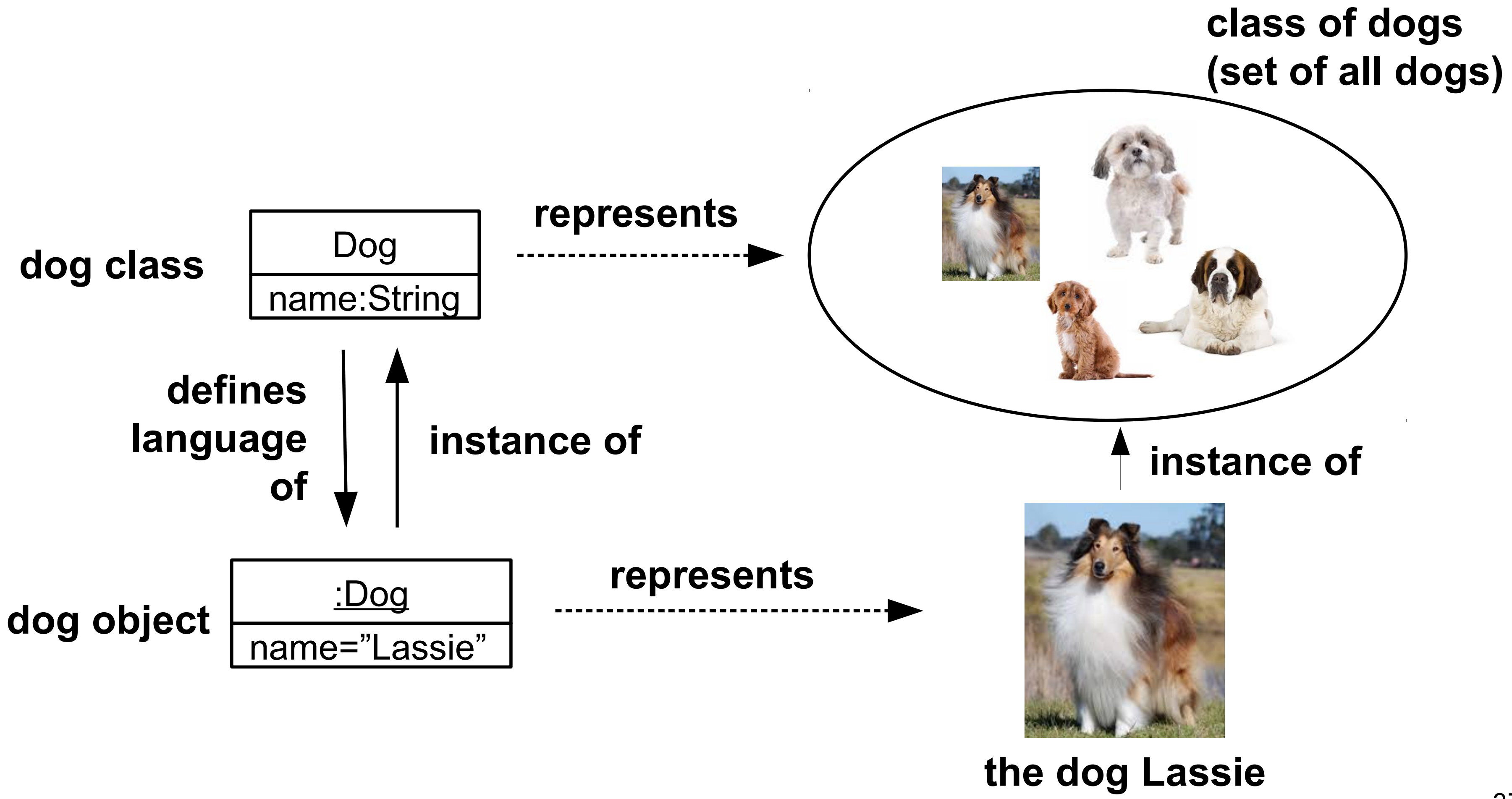
M3	**meta-metamodel** to define metamodels on M2, also describes itself
M2	**metamodels**, for defining a modeling language on M1
M1	**models** of data or processes
M0	**instance-model,** concrete data



exections of Petri nets

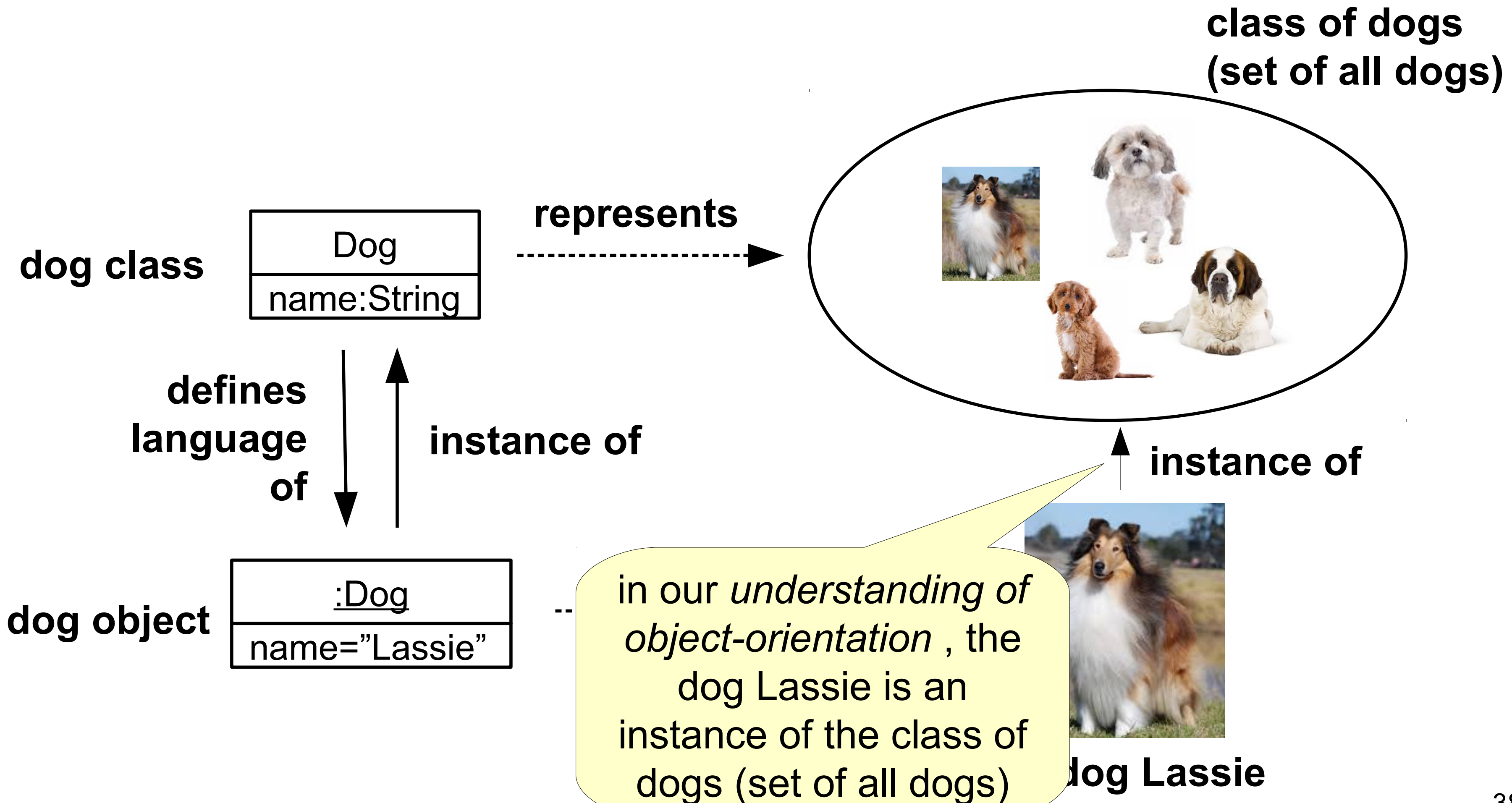
"Instance-of" vs. "Represents" Relationship

- A model **represents** an original
- A model is an **instance of** a metamodel



“Instance-of” vs. “Represents” Relationship

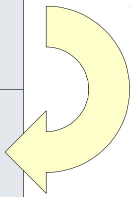
- A model **represents** an original
- A model is an **instance of** a metamodel



Meta-Levels gone wrong

- One problematic interpretation of meta-levels
  - appears in some sources

M3	**meta-metamodel** to define metamodels on M2, also describes itself
M2	**metamodels**, for defining a modeling language on M1
M1	**models** of data or processes
M0	generated code ???



code generation

Meta-Levels gone wrong

- One problematic interpretation of meta-levels
  - appears in some sources

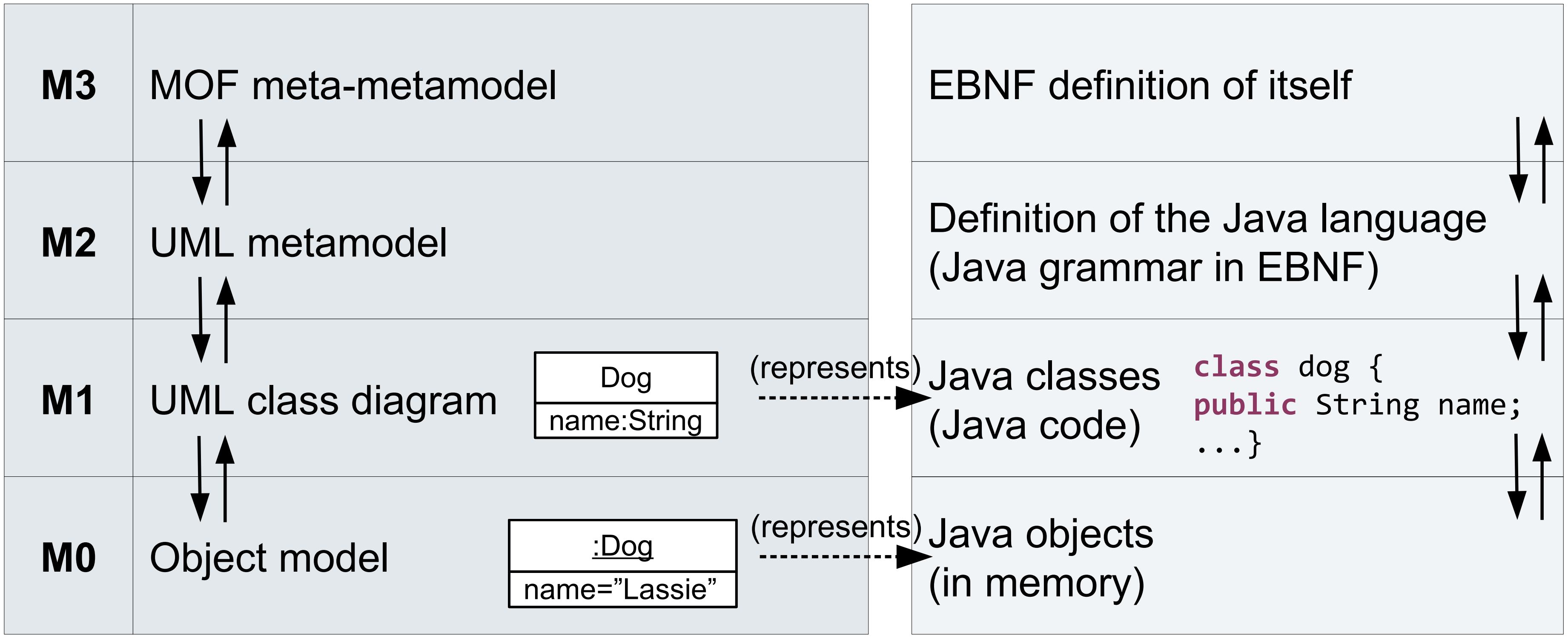
M3	**meta-metamodel** to define metamodels on M2, also describes itself
M2	**metamodels**, for defining a modeling language on M1
M1	**models** of data or processes
M0	generated code **NO!**

code generation

what is the relationship between, for example, a class diagram and code that is generated from it?

Meta-Levels done right!

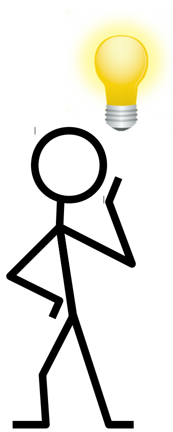
- Classes in the class diagram describe Java classes

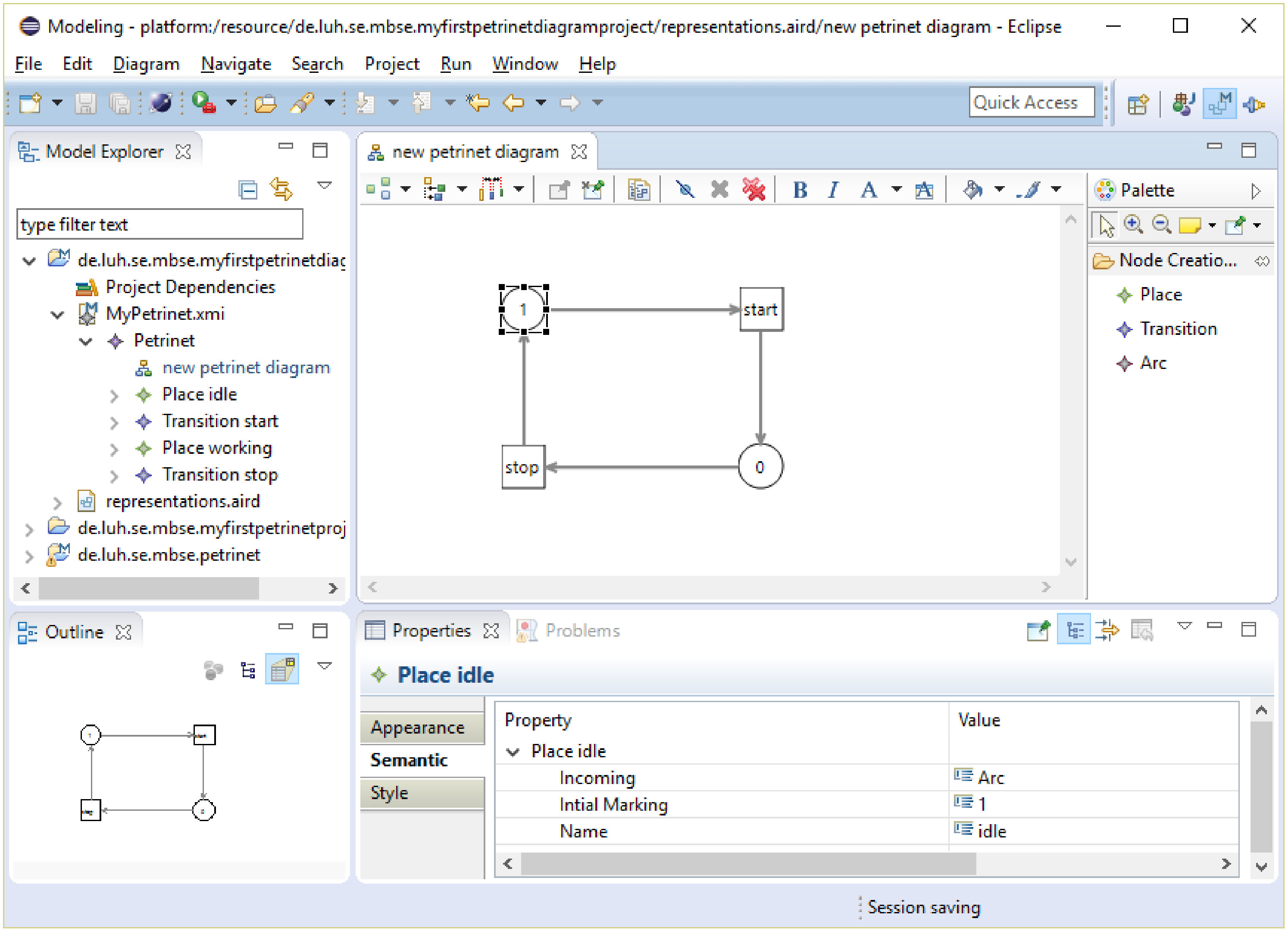


```
class dog {
public String name;
...}
```

2.4. Metamodeling frameworks

Vision: Build a Petri Net Modeling Tool




Modeling - platform:/resource/de.luh.se.mbse.myfirstpetrinetdiagramproject/representations.aird/new petrinet diagram - Eclipse
File Edit Diagram Navigate Search Project Run Window Help
Quick Access
Model Explorer
new petrinet diagram
Palette
Node Creatio...
Place
Transition
Arc
1
start
stop
0
Outline
Properties
Problems
Appearance
Semantic
Style
Session saving

type filter text

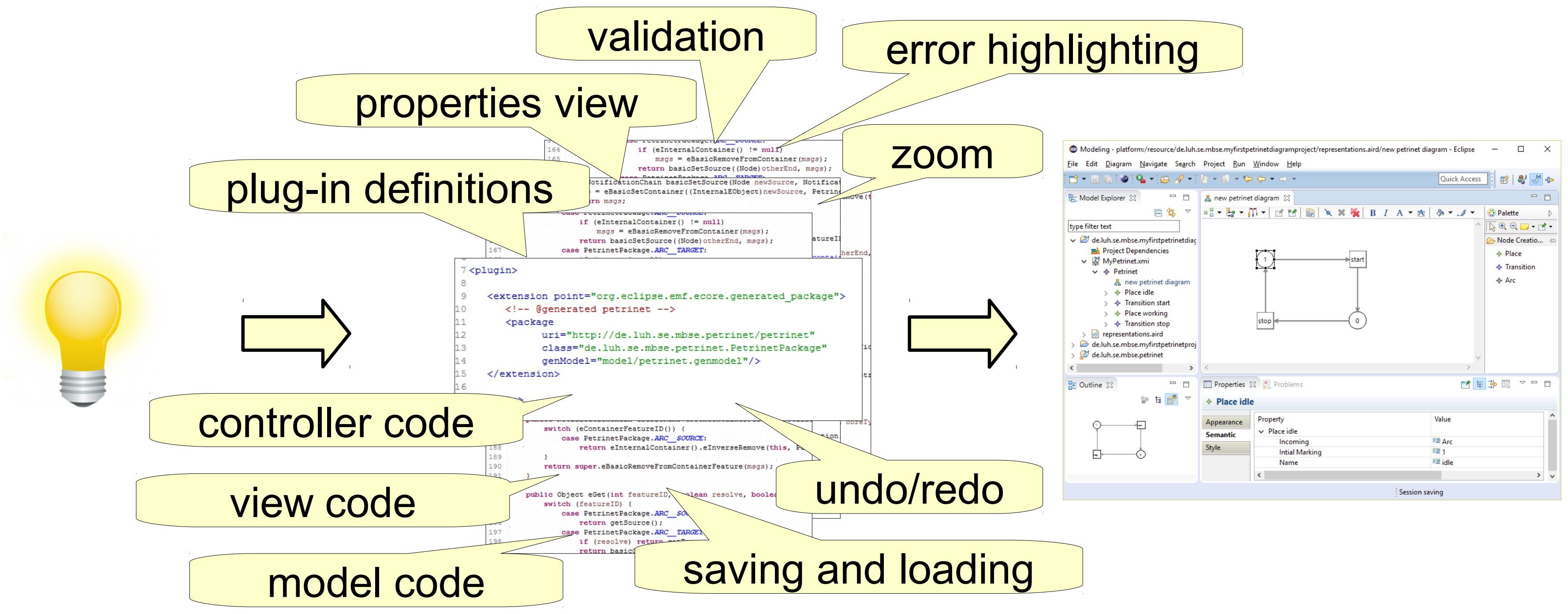
- de.luh.se.mbse.myfirstpetrinetdiag
  - Project Dependencies
  - MyPetrinet.xmi
    - Petrinet
      - new petrinet diagram
      - Place idle
      - Transition start
      - Place working
      - Transition stop
  - representations.aird
- de.luh.se.mbse.myfirstpetrinetproj
- de.luh.se.mbse.petrinet

Place idle

Property	Value
Place idle	
Incoming	Arc
Intial Marking	1
Name	idle

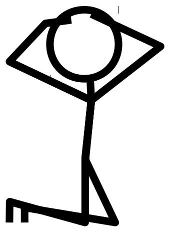
Build a Petri Net Modeling Tool

- Manual implementation: A lot of repetitive or generic code



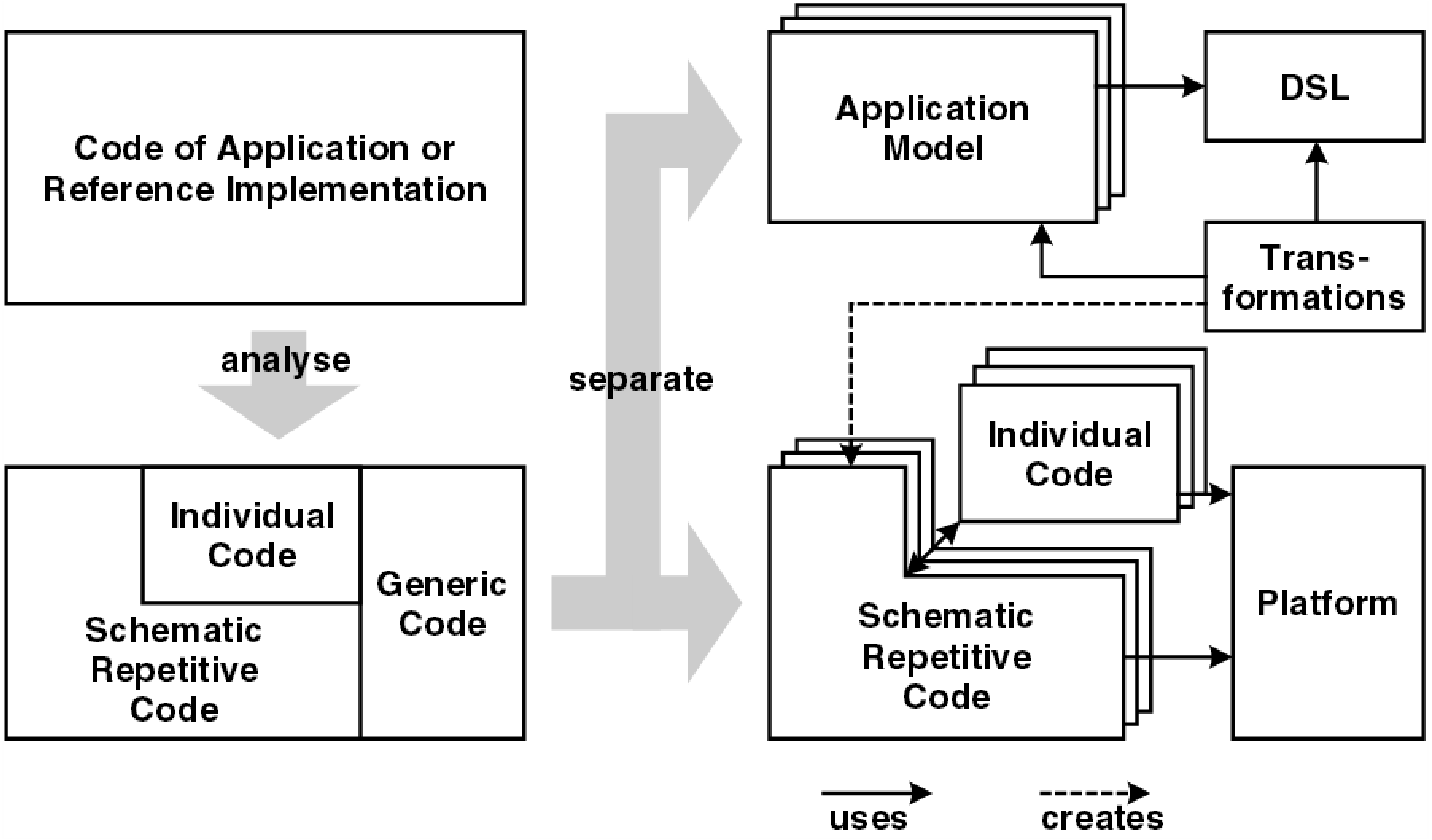
```
<plugin>

   <extension point="org.eclipse.emf.ecore.generated_package">
      <!-- @generated petrinet -->
      <package
            uri="http://de.luh.se.mbse.petrinet/petrinet"
            class="de.luh.se.mbse.petrinet.PetrinetPackage"
            genModel="model/petrinet.genmodel"/>
   </extension>
```



The idea of MBSE/MDSD in a bit more detail

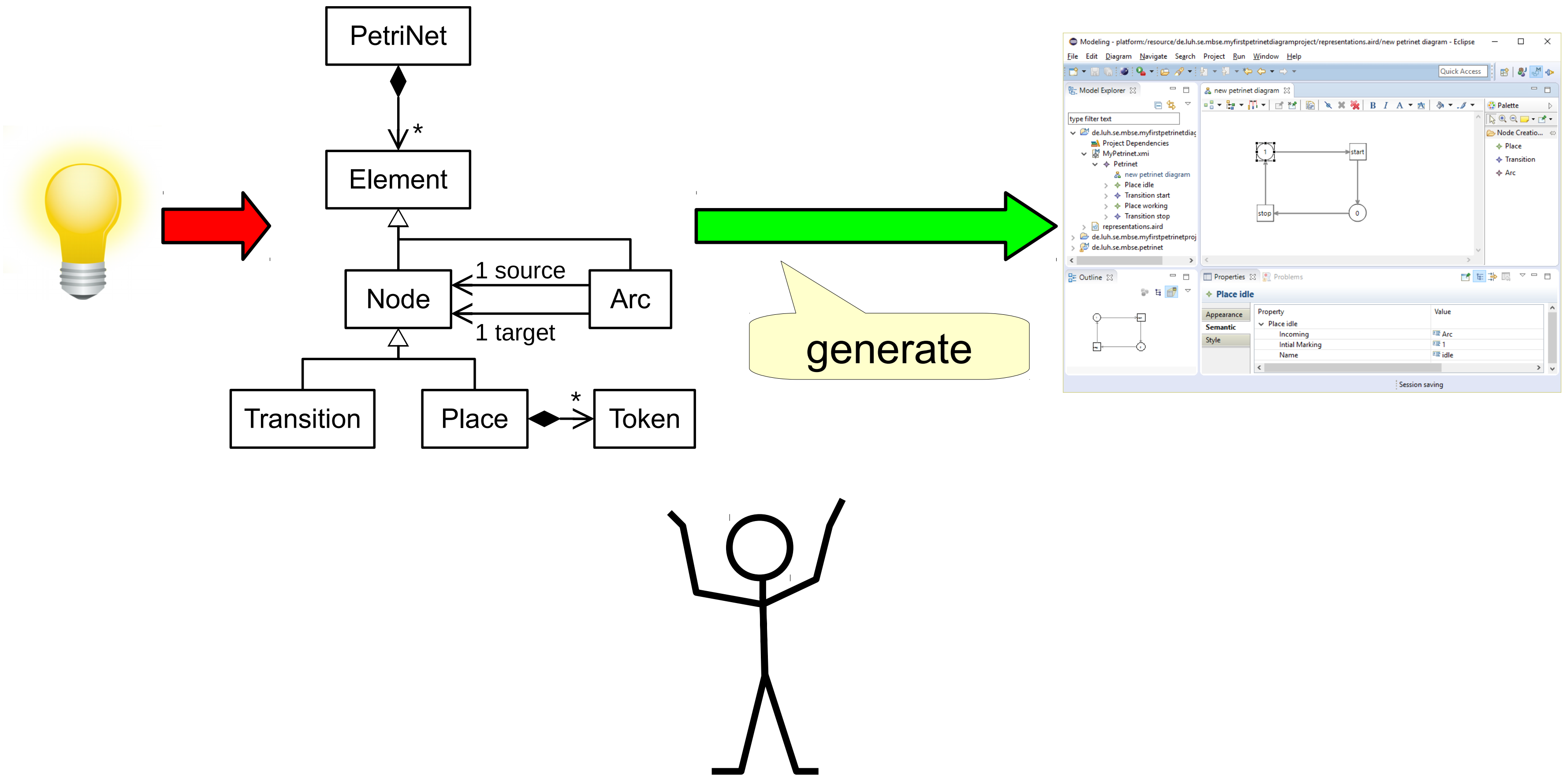
in the last lecture...



from Stahl, Völter: Model-Driven Software Development: Technology, Engineering, Management, Wiley, 2006.

Build a Petri Net Modeling Tool

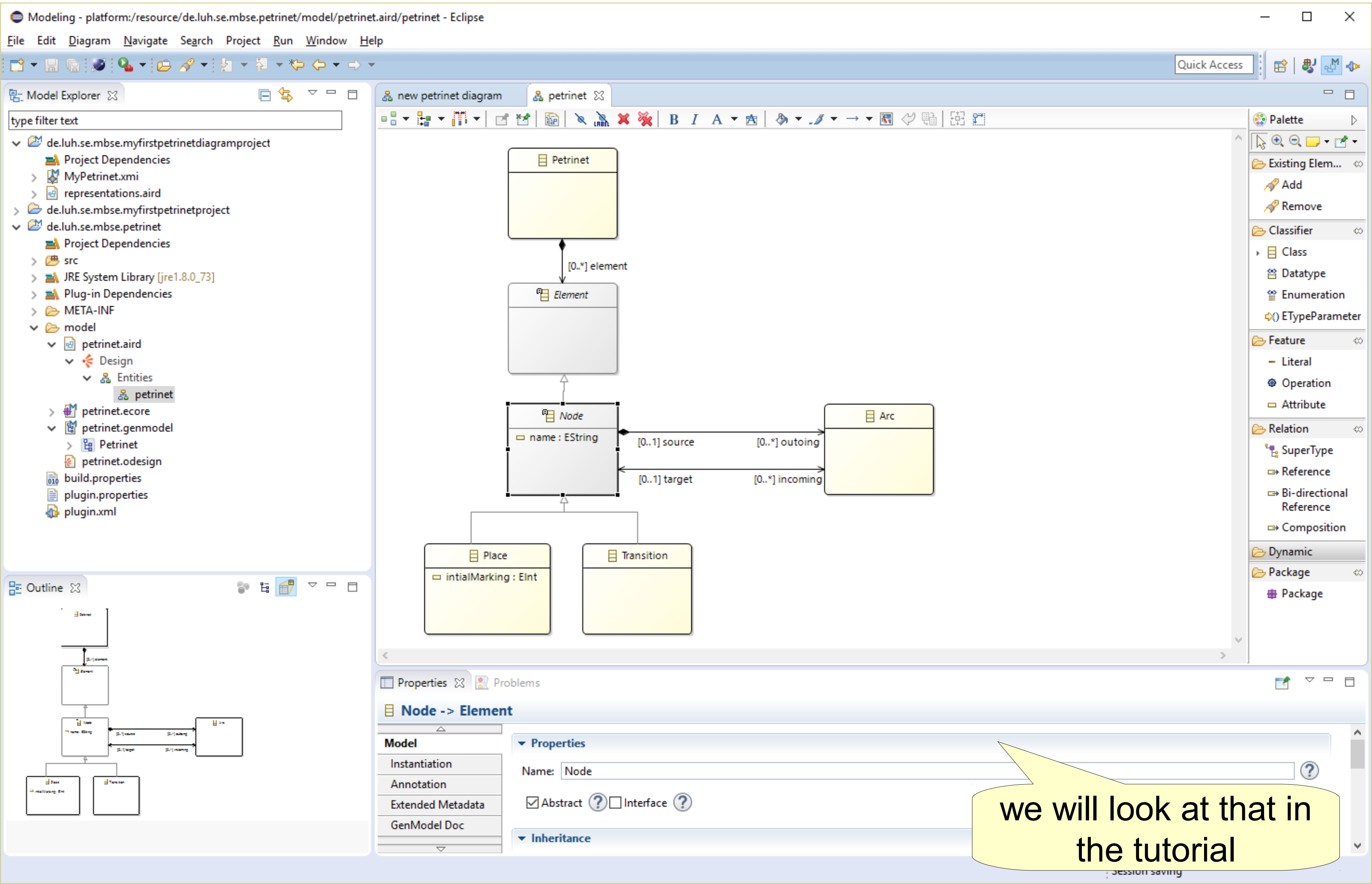
- Model-based approach for building modeling tools: Provide only a few conceptual models and generate tool automatically



Eclipse Modeling Framework

- The Eclipse Modeling Framework (EMF) is a metamodeling framework for Eclipse
- It allows us to build modeling tools inside Eclipse
  - but it can also be used outside of Eclipse
- EMF provides the Ecore meta-metamodel
  - reference implementation of OMG's EMOF, “Essential MOF”
- Many other frameworks build on EMF
  - Eclipse UML2 (reference implementation of OMG's UML2)
  - ...

Eclipse Modeling Framework (Modeling a Petri Net Metamodel)



Node -> Element

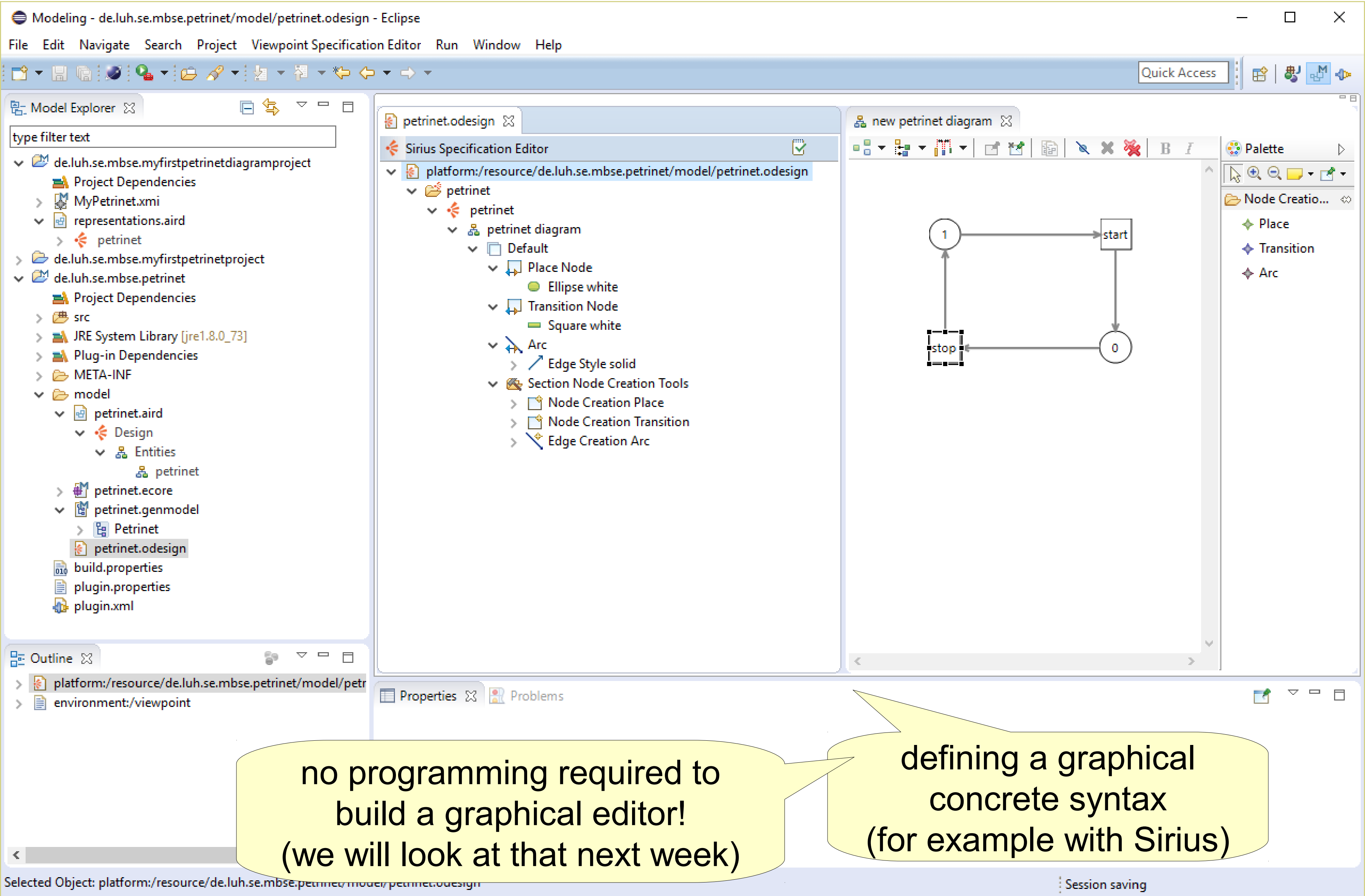
Properties

Name: Node

[✓] Abstract [ ] Interface

we will look at that in the tutorial

Graphical Editors (for example with Sirius)



Modeling - de.luh.se.mbse.petrinet/model/petrinet.odesign - Eclipse

File Edit Navigate Search Project Viewpoint Specification Editor Run Window Help

Model Explorer

- de.luh.se.mbse.myfirstpetrinetdiagramproject
  - Project Dependencies
  - MyPetrinet.xmi
  - representations.aird
    - petrinet
- de.luh.se.mbse.myfirstpetrinetproject
- de.luh.se.mbse.petrinet
  - Project Dependencies
  - src
  - JRE System Library [jre1.8.0_73]
  - Plug-in Dependencies
  - META-INF
  - model
    - petrinet.aird
      - Design
        - Entities
          - petrinet
    - petrinet.ecore
    - petrinet.genmodel
      - Petrinet
    - petrinet.odesign
  - build.properties
  - plugin.properties
  - plugin.xml

Outline

- platform:/resource/de.luh.se.mbse.petrinet/model/petr
- environment:/viewpoint

petrinet.odesign — Sirius Specification Editor

- platform:/resource/de.luh.se.mbse.petrinet/model/petrinet.odesign
  - petrinet
    - petrinet
      - petrinet diagram
        - Default
          - Place Node
            - Ellipse white
          - Transition Node
            - Square white
          - Arc
            - Edge Style solid
          - Section Node Creation Tools
            - Node Creation Place
            - Node Creation Transition
            - Edge Creation Arc

new petrinet diagram

Properties Problems

Selected Object: platform:/resource/de.luh.se.mbse.petrinet/model/petrinet.odesign

Session saving

no programming required to build a graphical editor! (we will look at that next week)

defining a graphical concrete syntax (for example with Sirius)