Model-Based Software Engineering

Lecture 02 – Metamodeling

Prof. Dr. Joel Greenyer

SOFTWARE ENGINEERING

April 12, 2016

Leibniz Universität Hannover

Learning Objectives

Learning Objectives

- Understanding the principle of metamodeling

Learning Objectives

- Understanding the principle of metamodeling
- Understanding the principles of creating modeling languages

Learning Objectives

- Understanding the principle of metamodeling
- Understanding the principles of creating modeling languages
- Knowing important terms and concepts

Learning Objectives

- Understanding the principle of metamodeling
- Understanding the principles of creating modeling languages
- Knowing important terms and concepts
  - formal languages

Learning Objectives

- Understanding the principle of metamodeling
- Understanding the principles of creating modeling languages
- Knowing important terms and concepts
  - formal languages
  - models and metamodels

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

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

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

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

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:**

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

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)

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

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”

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

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

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

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)

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

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)

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)

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

Context-free Grammars

- A **context-free grammar** describes a language
  - a **language** is a set of sentences

Context-free Grammars

- A **context-free grammar** describes a language
  - a **language** is a set of sentences
  - a **sentence** is a sequence of **terminals**

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**

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:

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 = { (,) }

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}

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

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 = {

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 ')'

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
```

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             ::=
```

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             ::=
}
```

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: (() ())

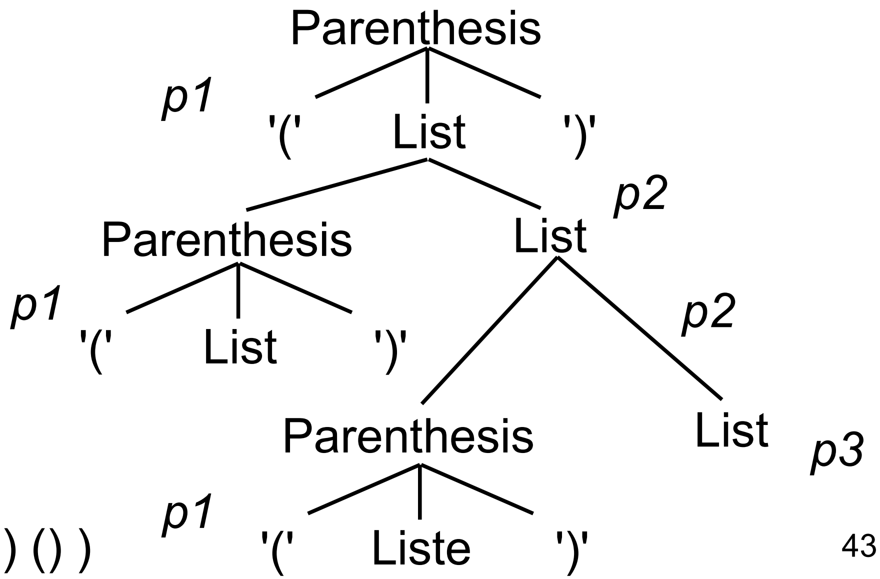
Context-free Grammars

- 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 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 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?

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

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)

So what's new about metamodeling?

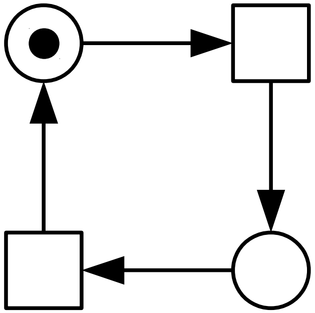
- In computer science, we defined formal languages for more than half a century
  - So what is new about metamodeling?

- 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

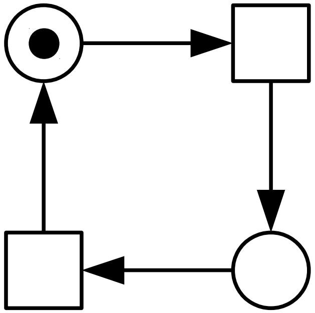
Metamodeling Example: Define a Petri Net Language

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



Metamodeling Example: Define a Petri Net Language

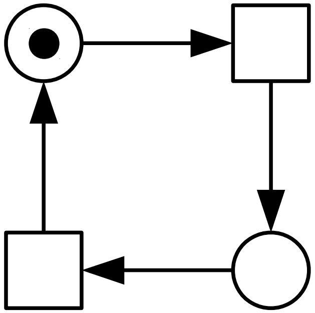
- How would you define a model for modeling Petri nets?
  - How would you metamodel Petri nets?



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.

Metamodeling Example: Define a Petri Net Language

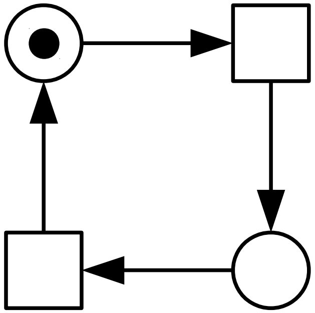
- How would you define a model for modeling Petri nets?
  - How would you metamodel Petri nets?



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.

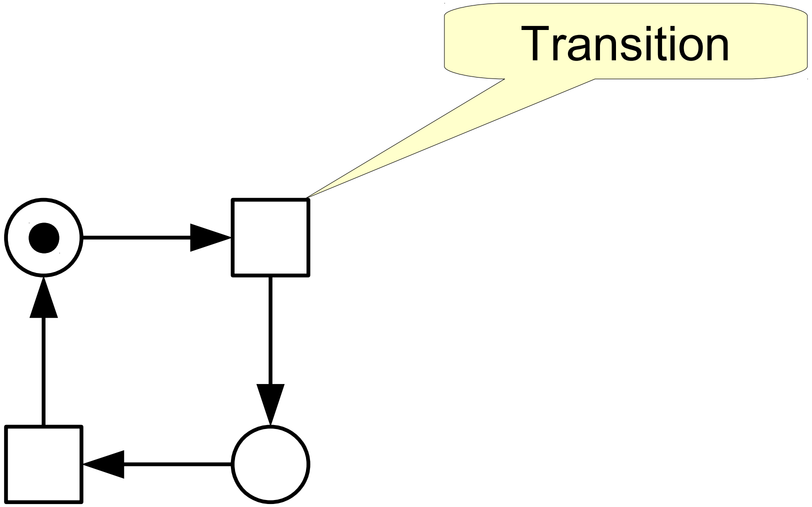
Object-Oriented Modeling Approach

- What are the objects that we see here?



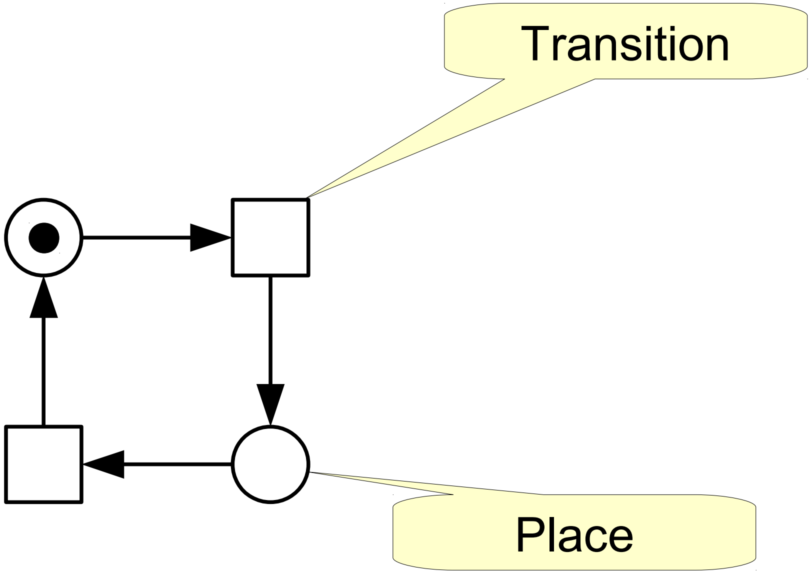
Object-Oriented Modeling Approach

- What are the objects that we see here?



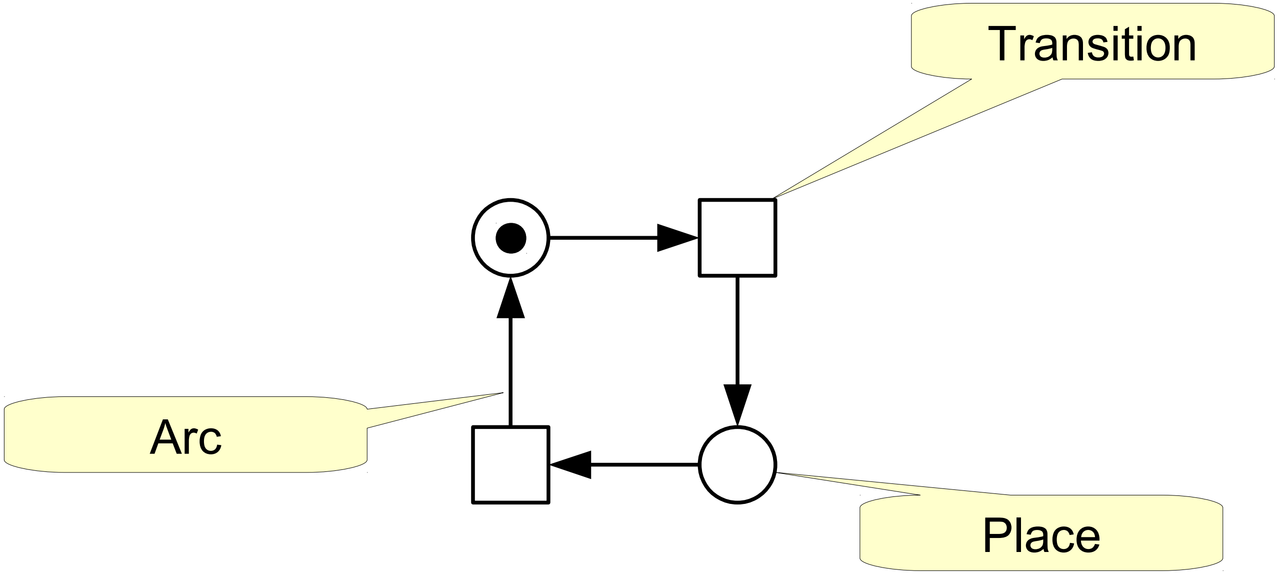
Object-Oriented Modeling Approach

- What are the objects that we see here?



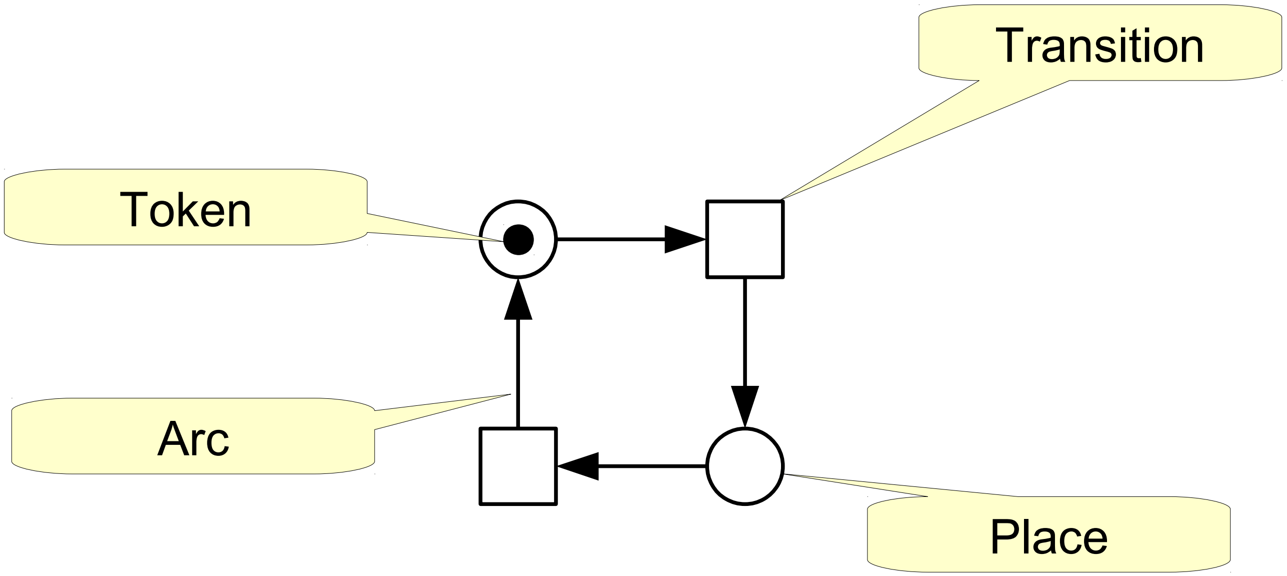
Object-Oriented Modeling Approach

- What are the objects that we see here?



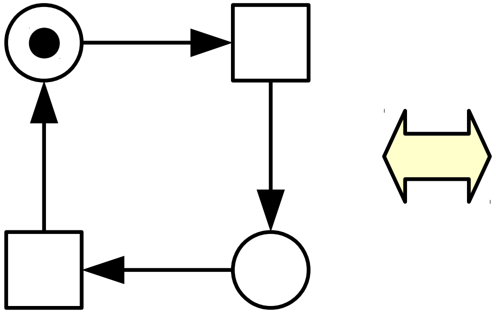
Object-Oriented Modeling Approach

- What are the objects that we see here?



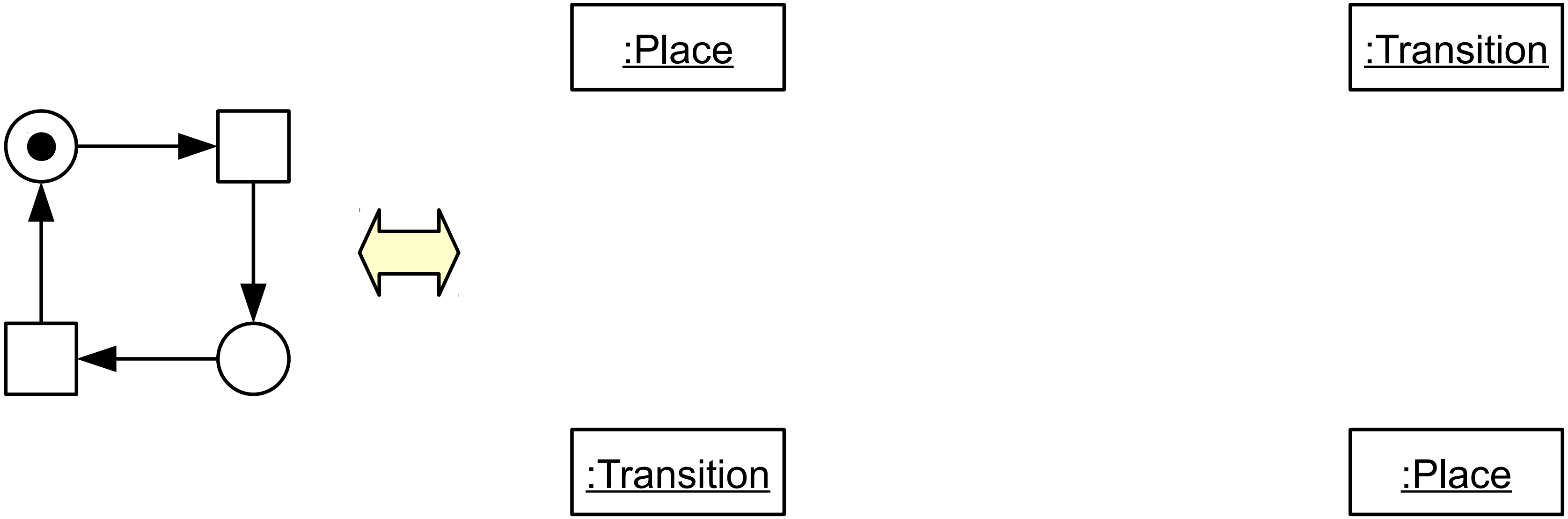
Object-Oriented Modeling Approach

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



Object-Oriented Modeling Approach

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



:Place

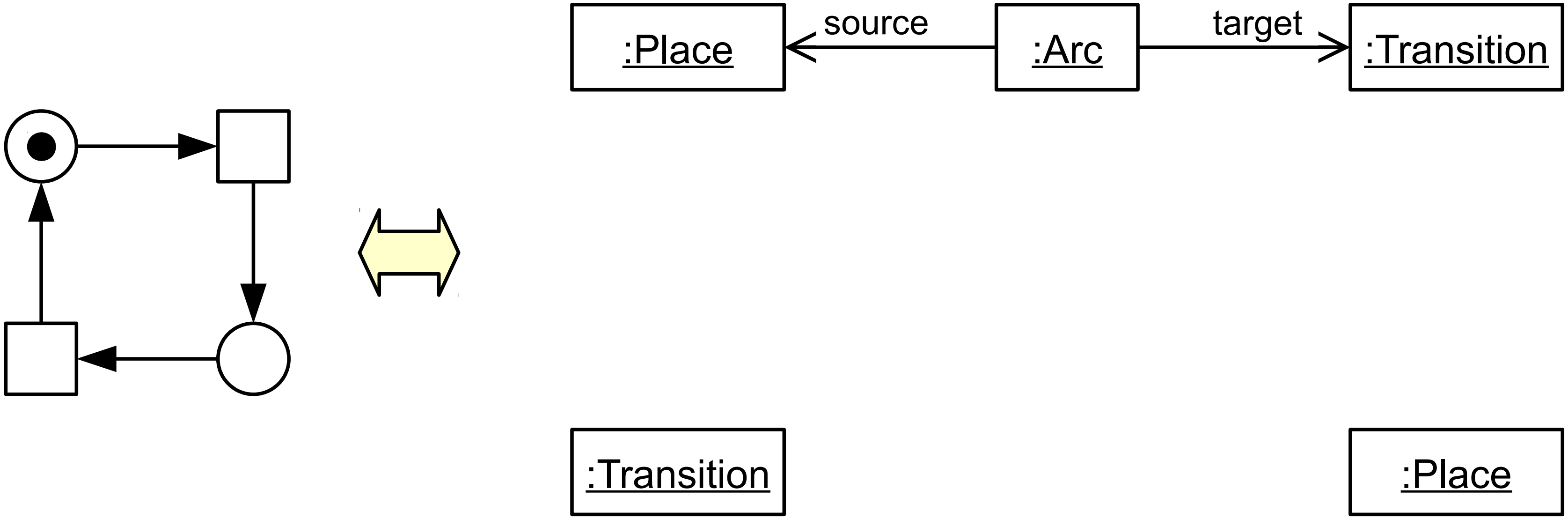
:Transition

:Transition

:Place

Object-Oriented Modeling Approach

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



:Place

source

:Arc

target

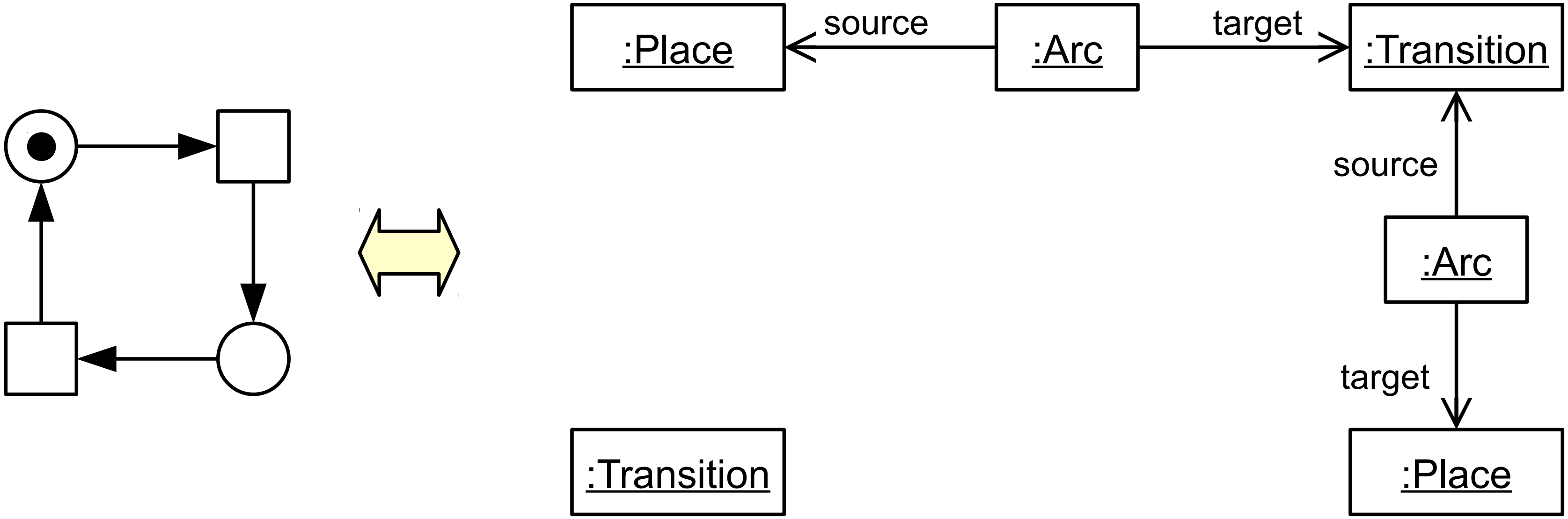
:Transition

:Transition

:Place

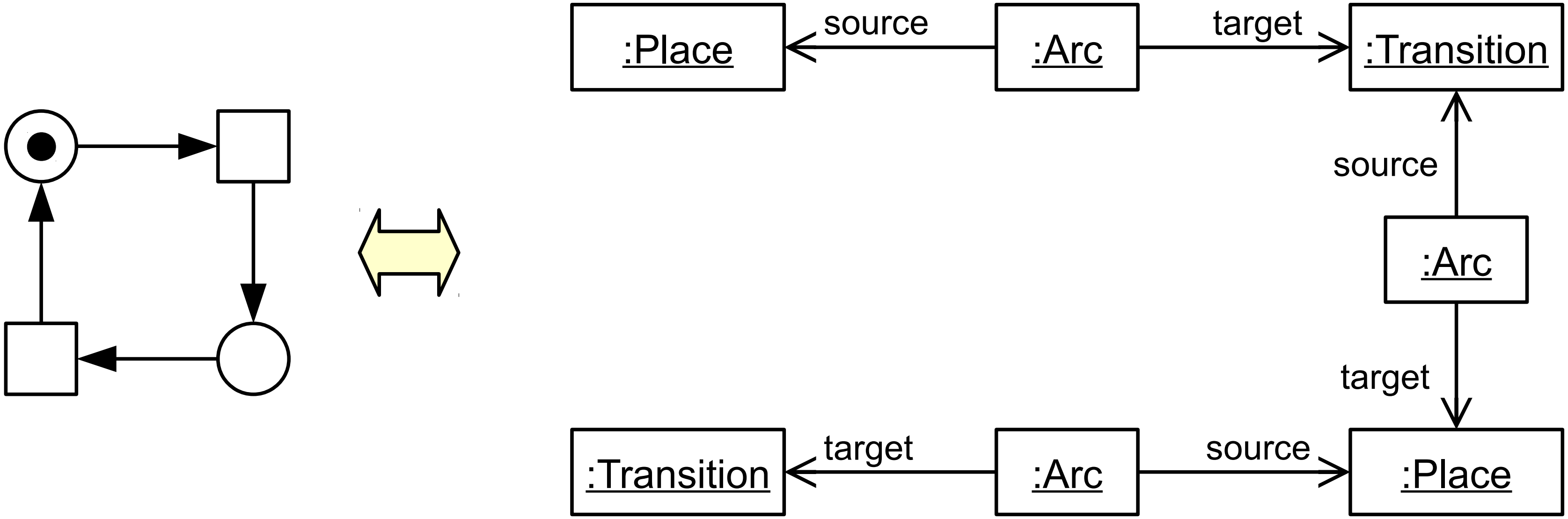
Object-Oriented Modeling Approach

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



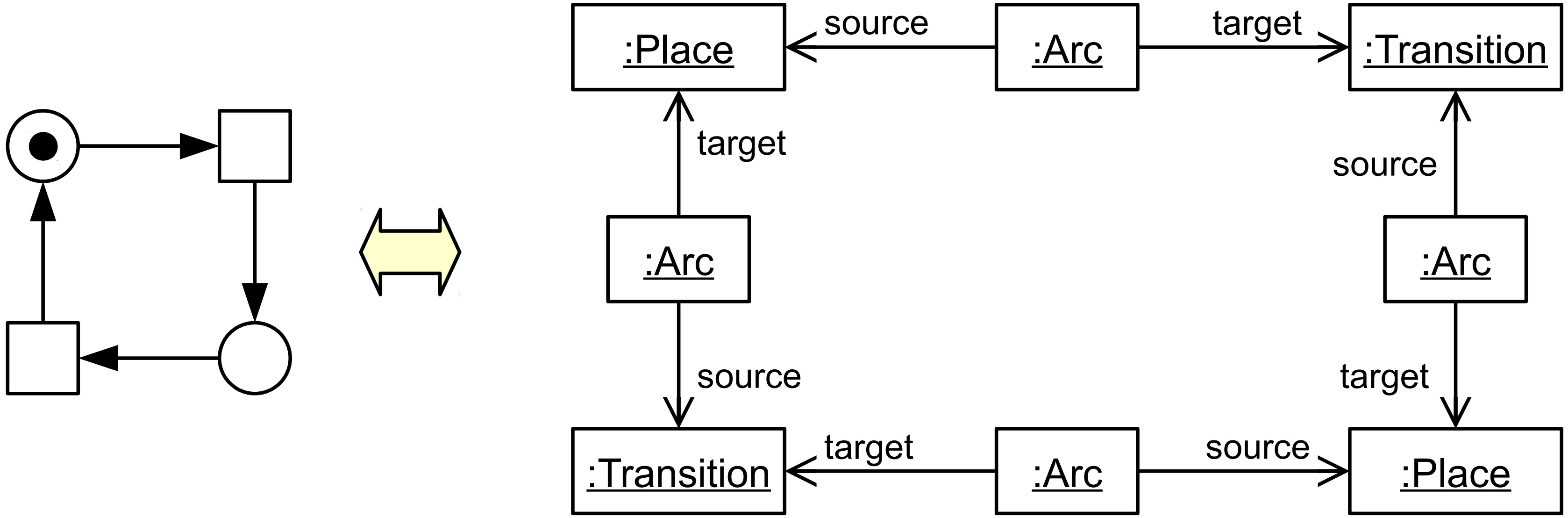
Object-Oriented Modeling Approach

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



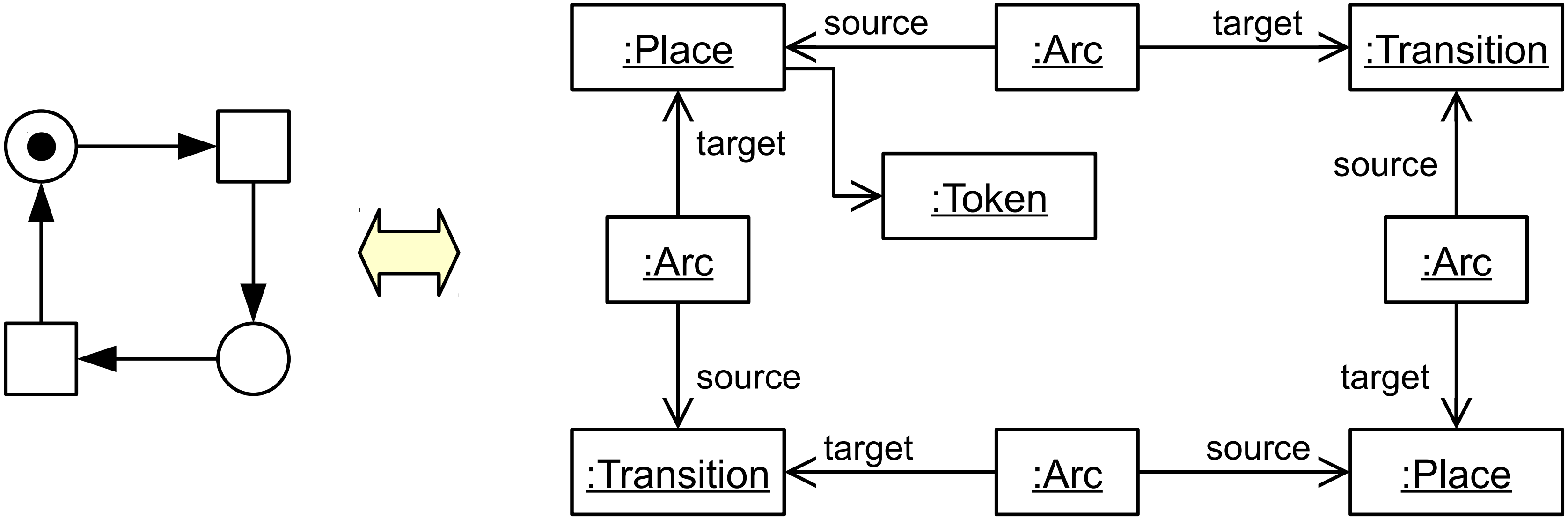
Object-Oriented Modeling Approach

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



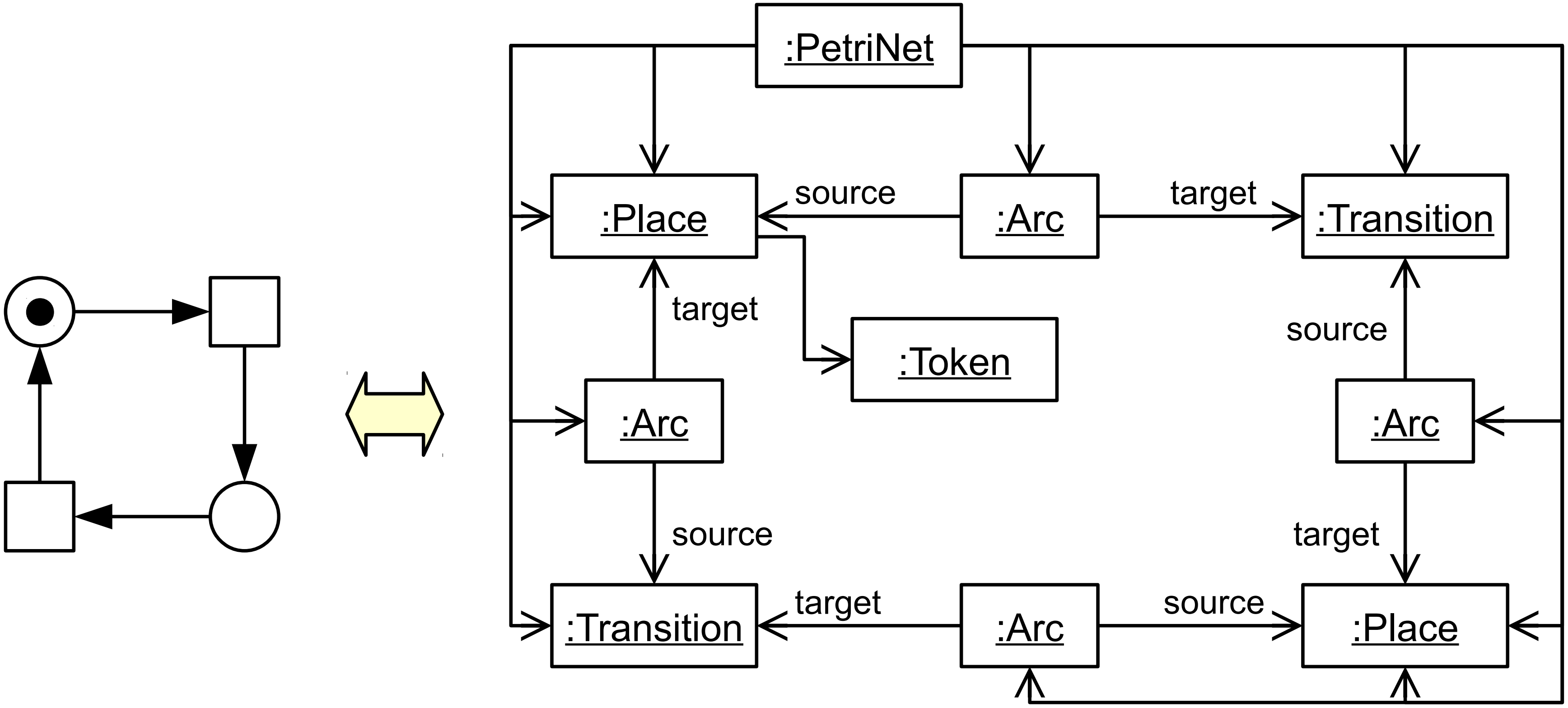
Object-Oriented Modeling Approach

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



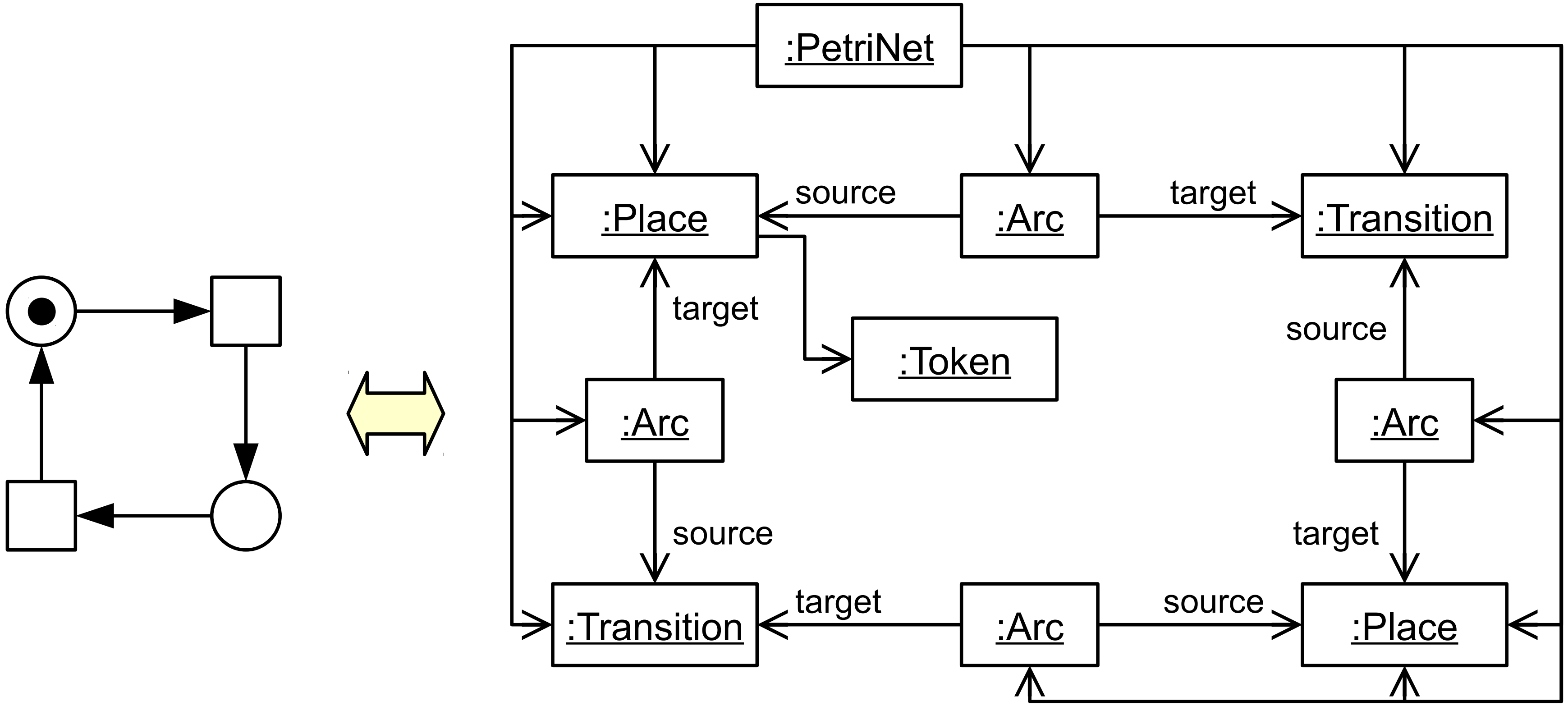
Object-Oriented Modeling Approach

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



Object-Oriented Modeling Approach

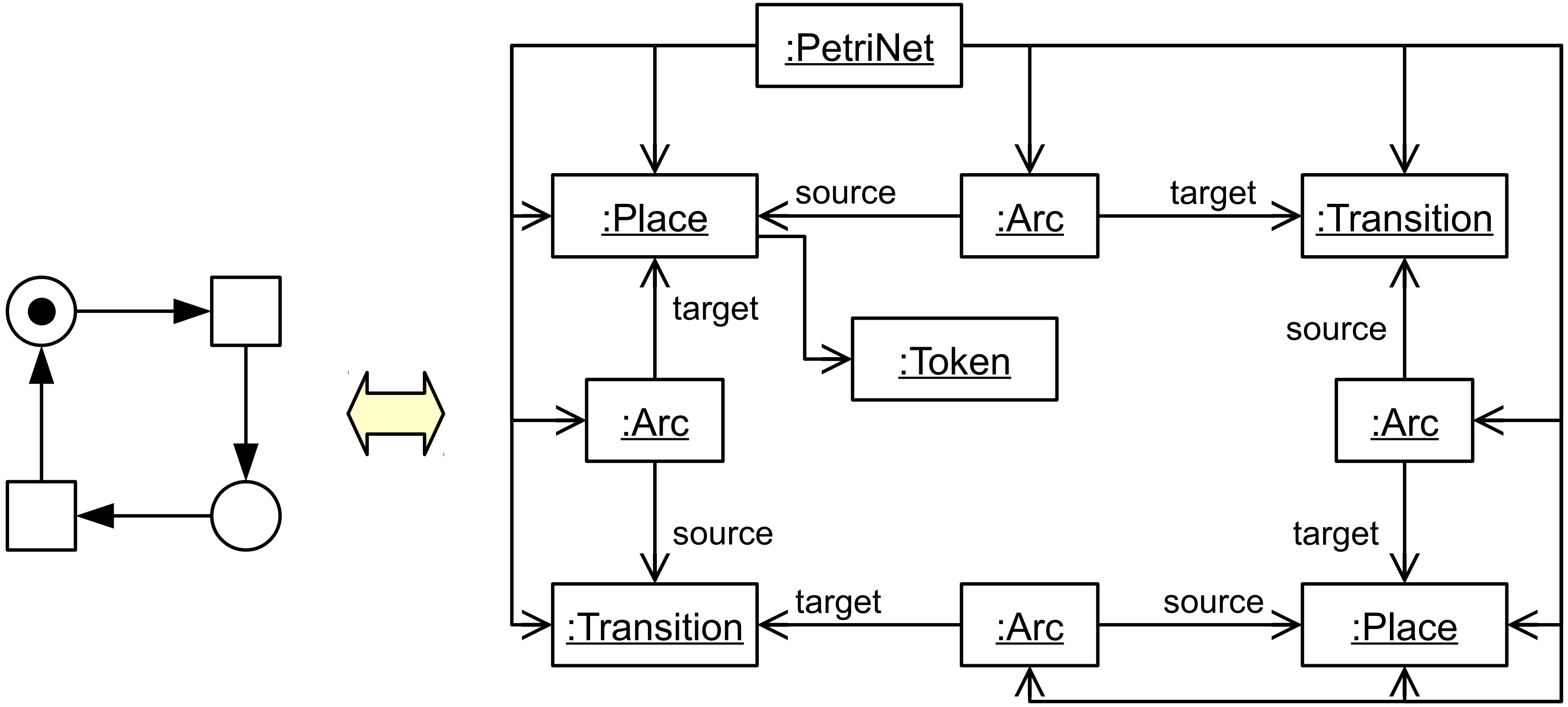
- **Step 1**: Understand a model as a **structure of objects**
- For the example:



concrete syntax

Object-Oriented Modeling Approach

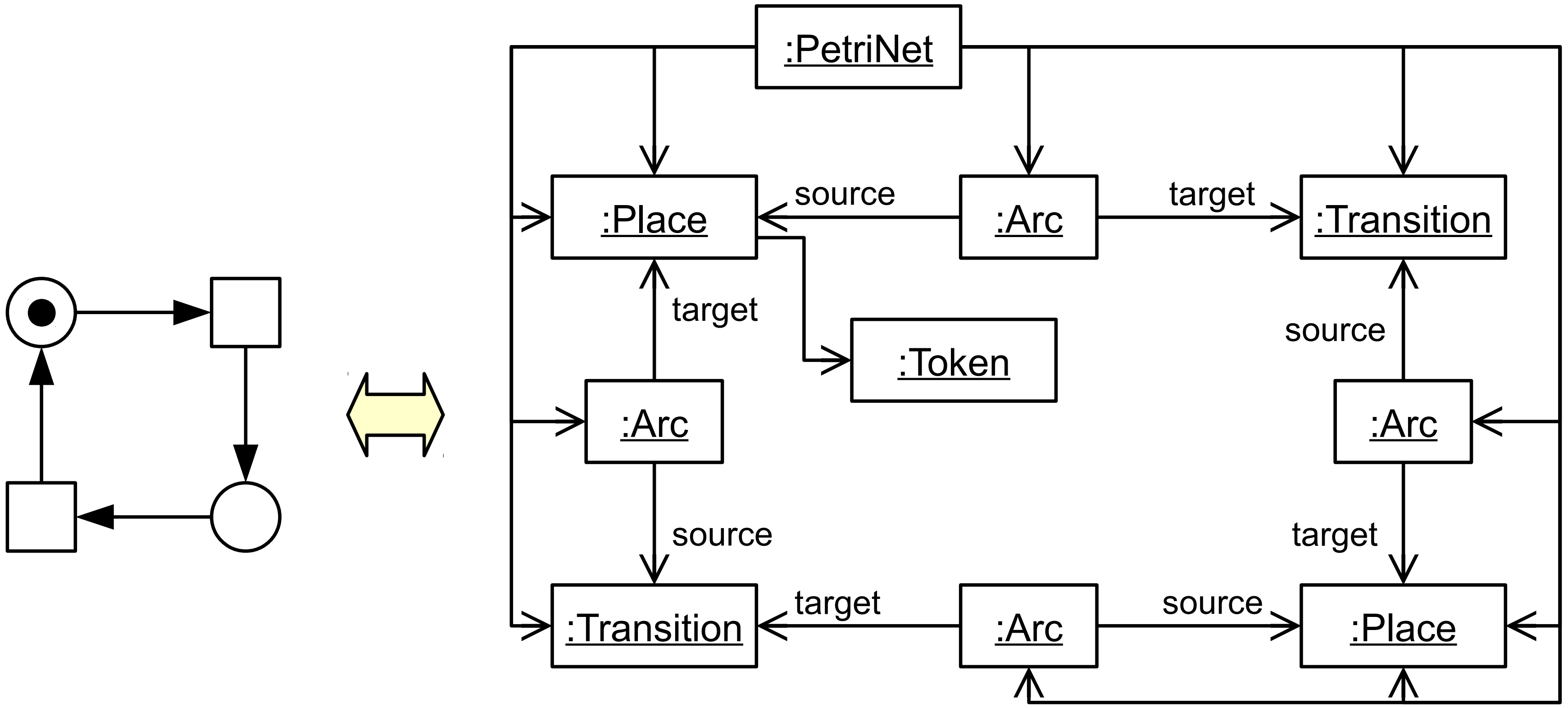
- **Step 1**: Understand a model as a **structure of objects**
- For the example:



concrete syntax
(representation to the user)

Object-Oriented Modeling Approach

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

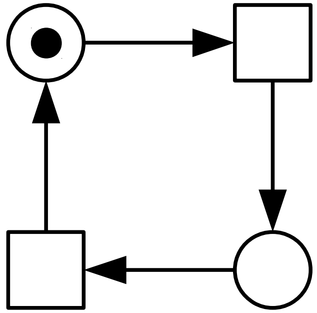


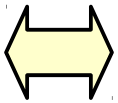
concrete syntax
(representation to the user)

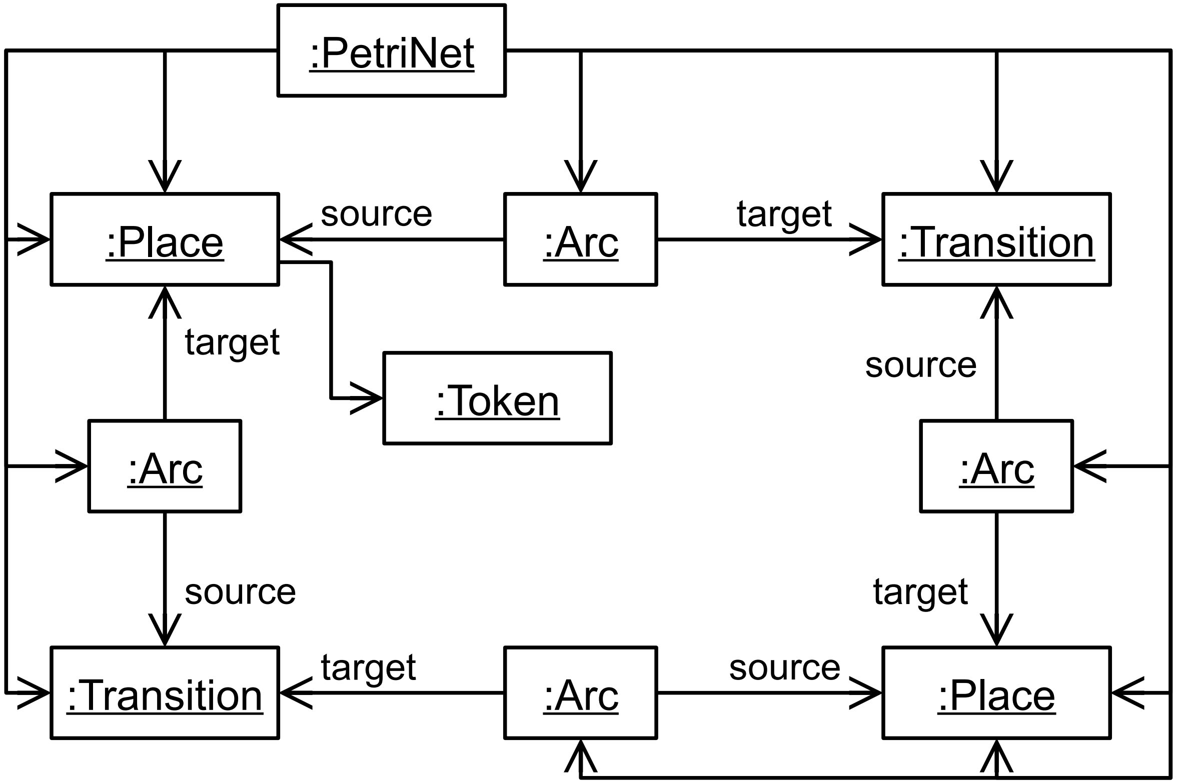
abstract syntax

Object-Oriented Modeling Approach

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





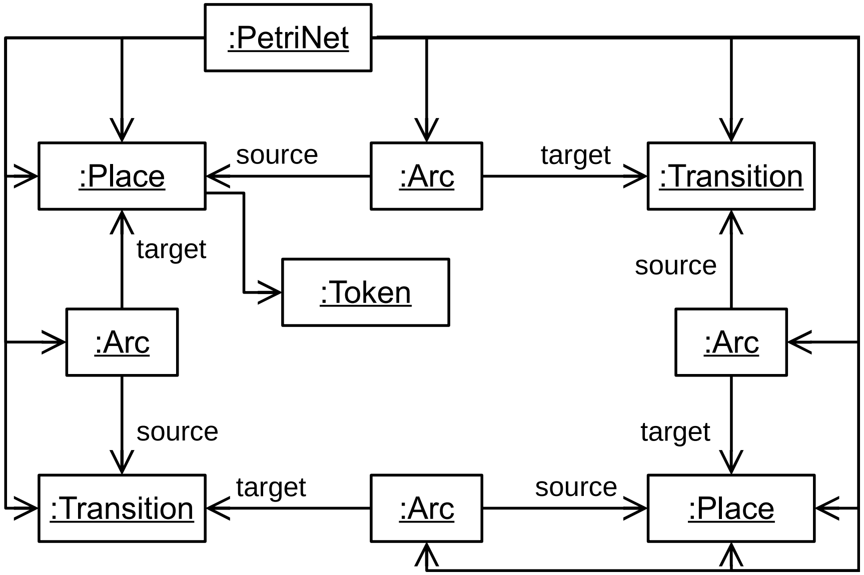


concrete syntax
(representation to the user)

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

Object-Oriented Modeling Approach

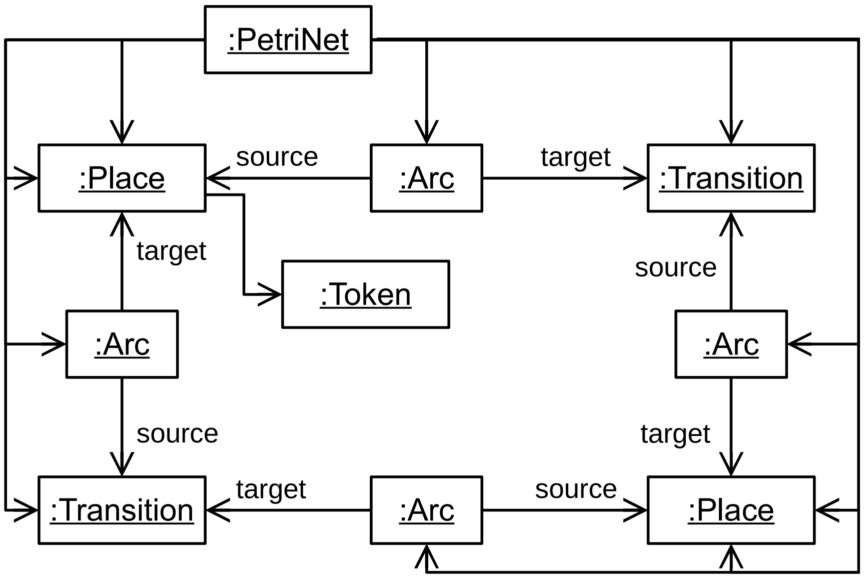
- **Step 2**: Create a model for all valid Petri nets (all object structures that represent valid Petri nets)



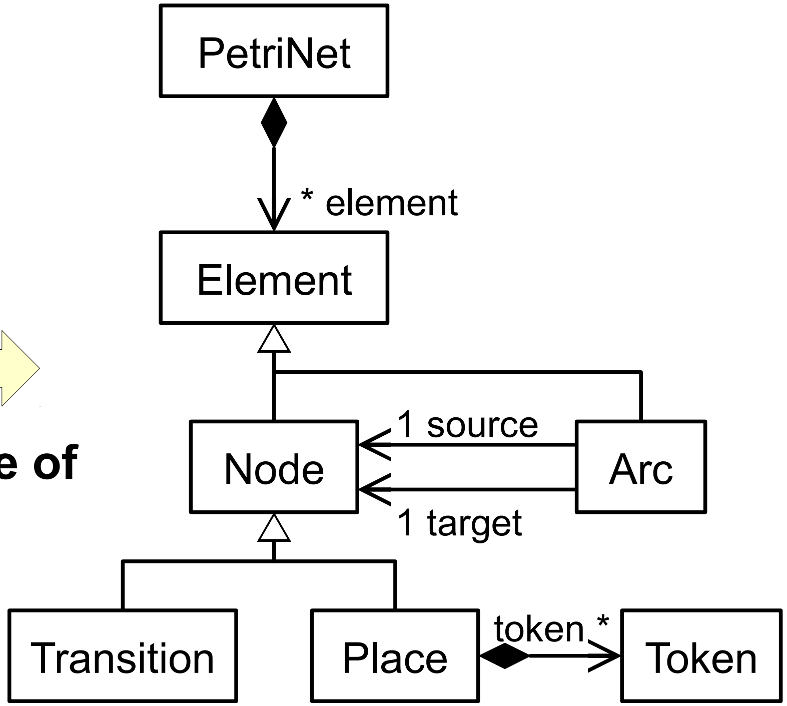
object model

Object-Oriented Modeling Approach

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



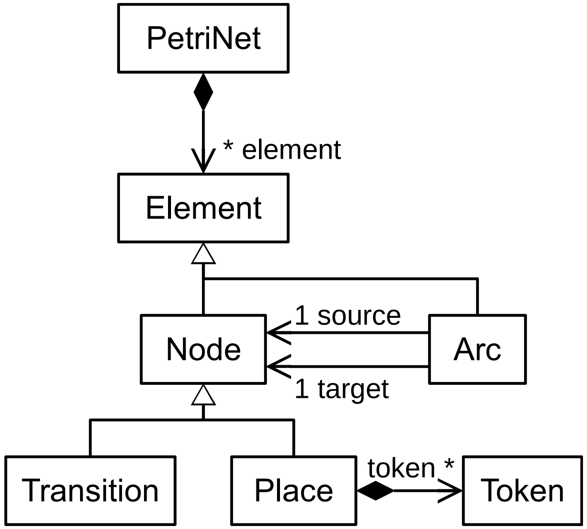
instance of

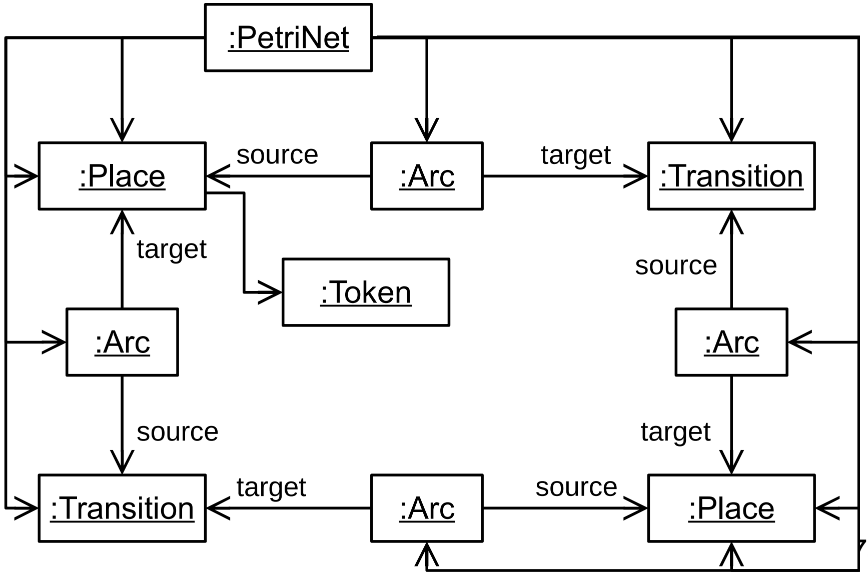


object model

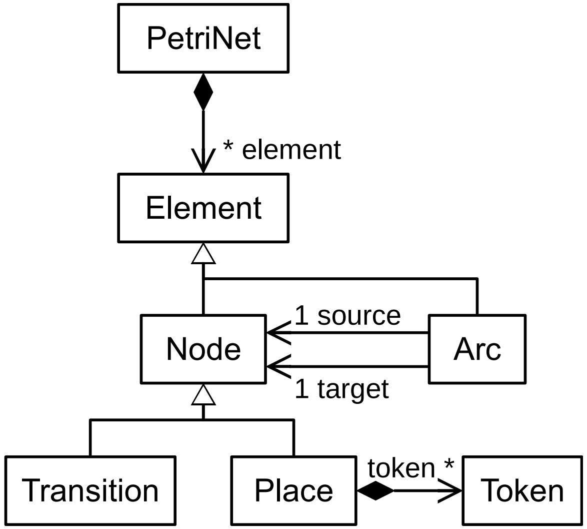
class model

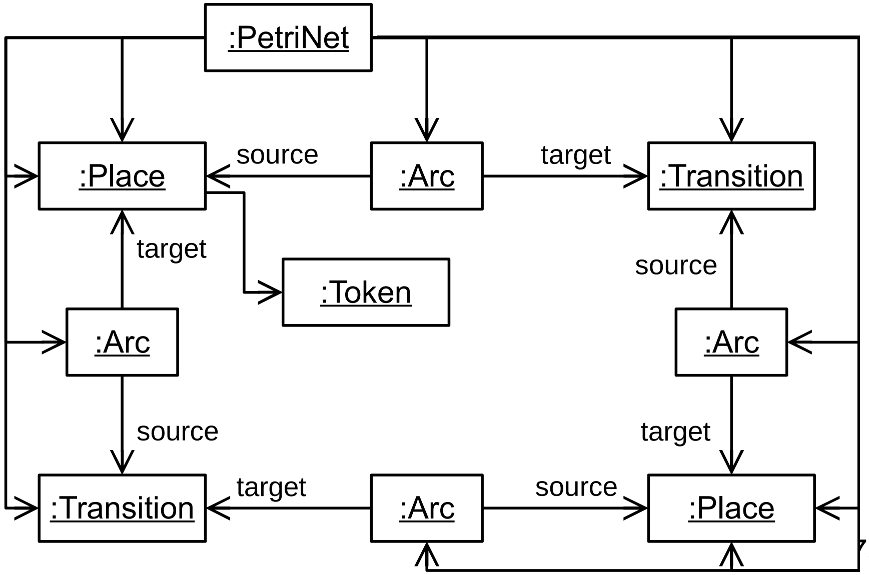
Model and Metamodel





Model and Metamodel





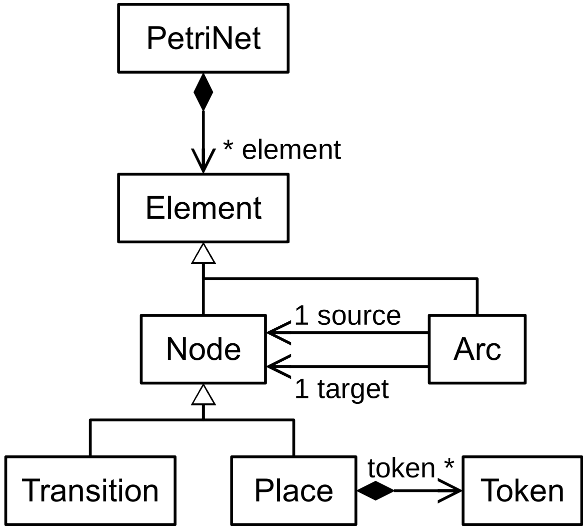
model

(model expressed in the language)

Model and Metamodel

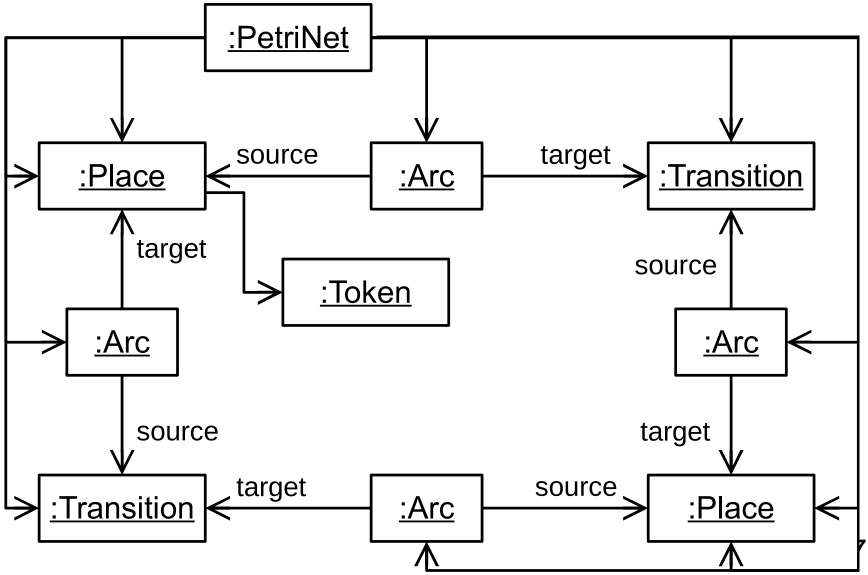
metamodel

(language definition)



model

(model expressed in the language)



Model and Metamodel

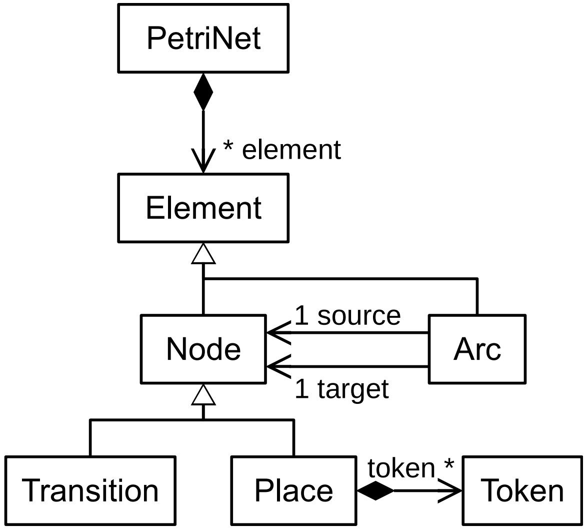
metamodel

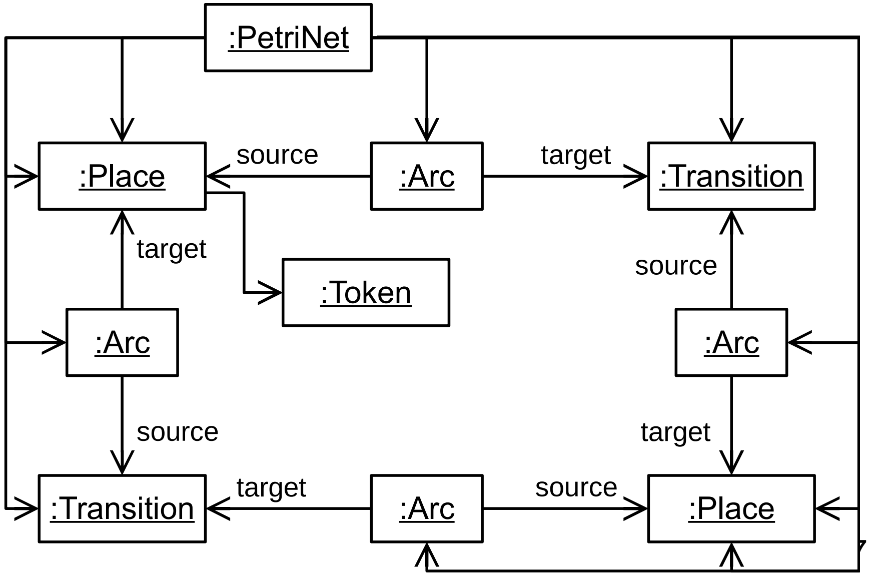
(language definition)

↑ **instance of**

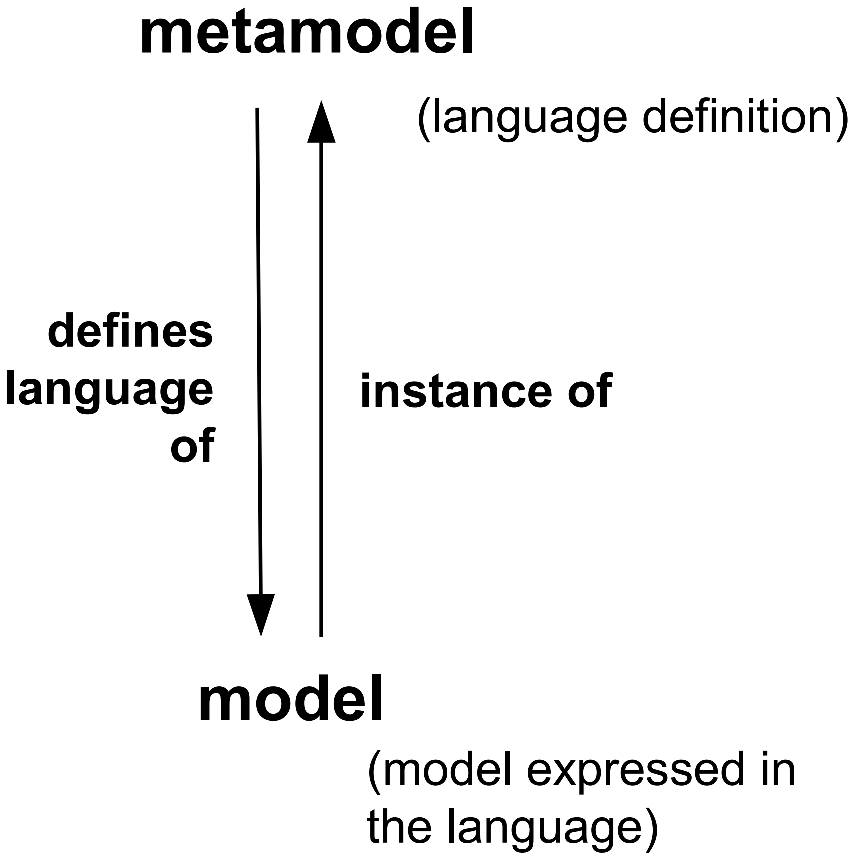
model

(model expressed in the language)





Model and Metamodel



metamodel

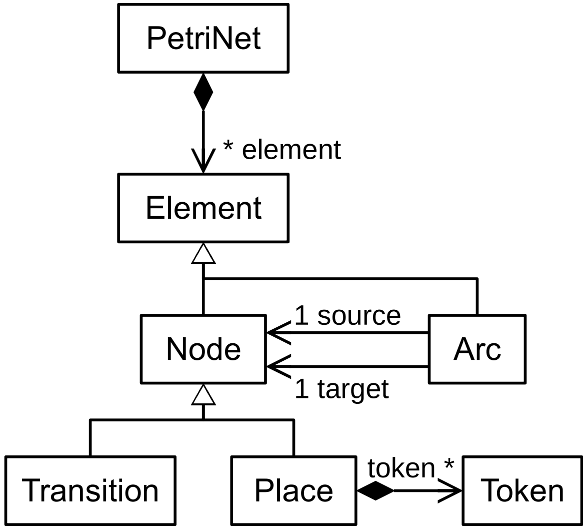
(language definition)

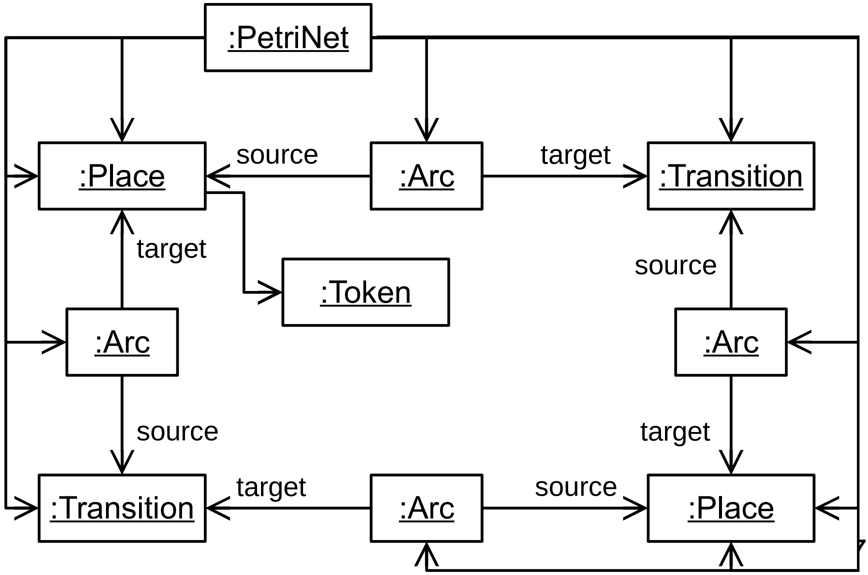
defines language of

instance of

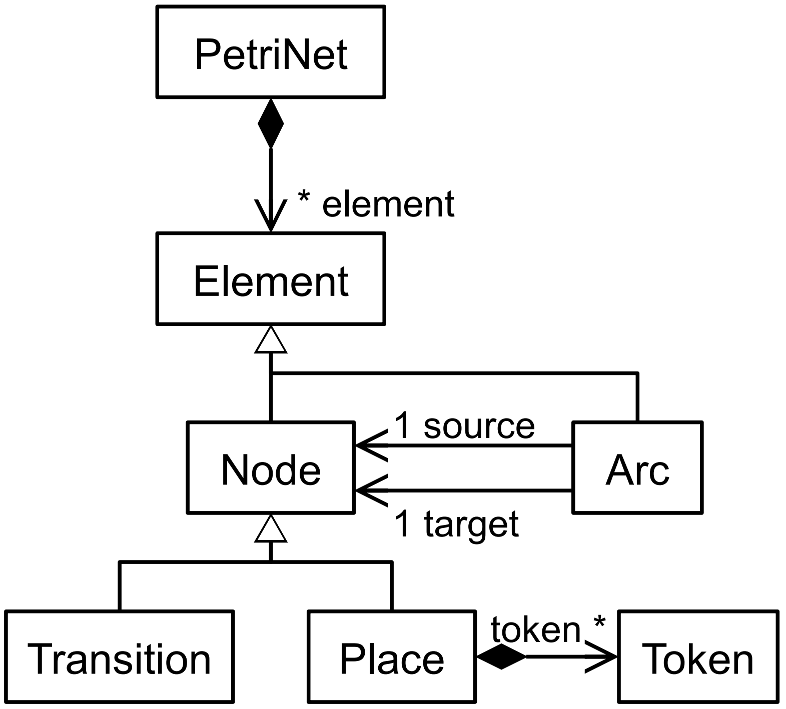
model

(model expressed in the language)

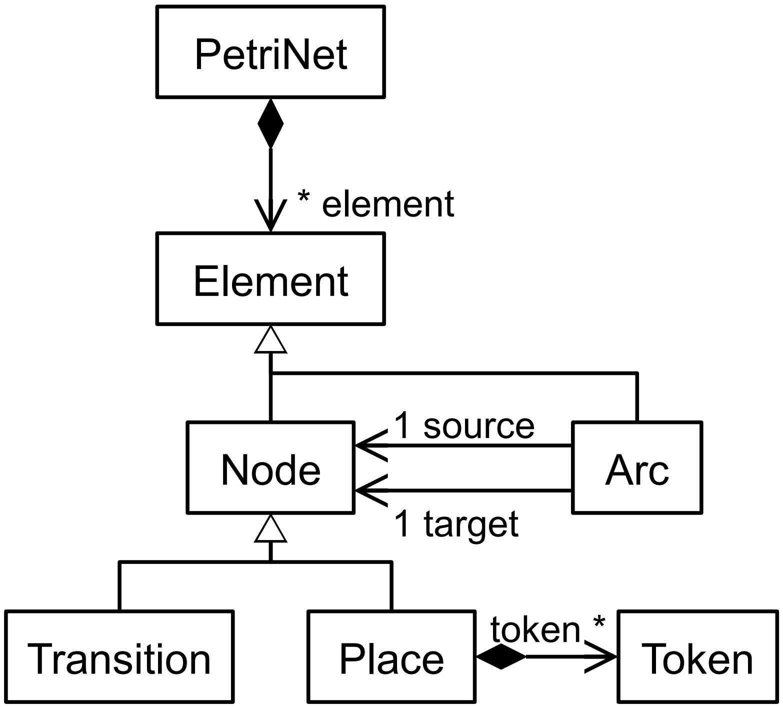


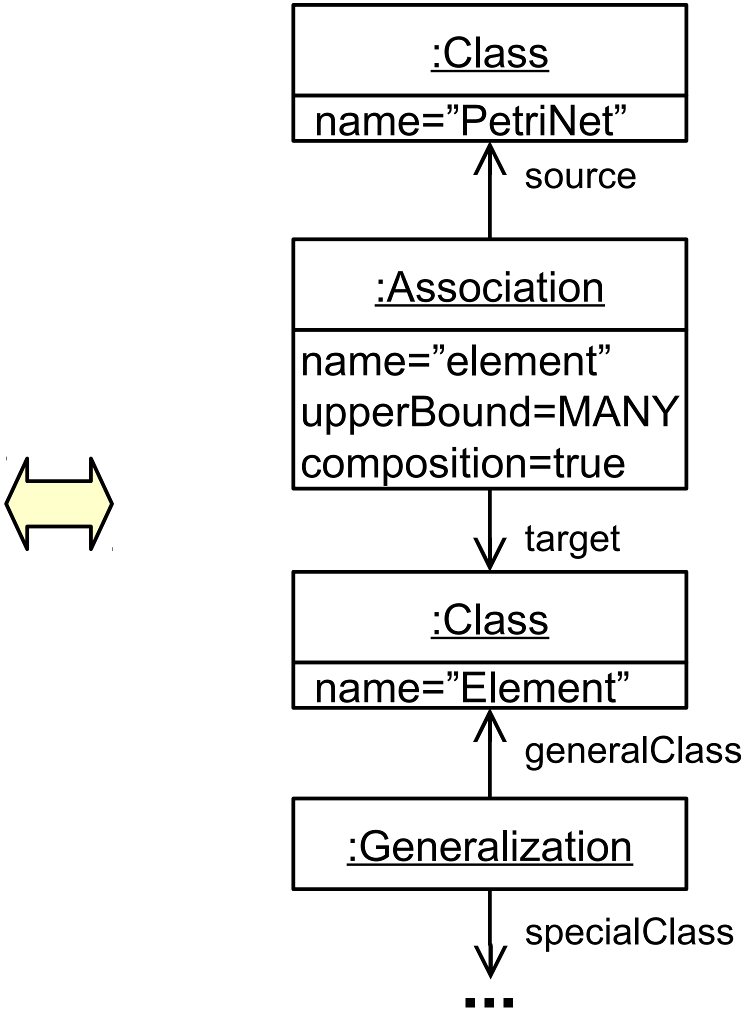


Class models are models, too!

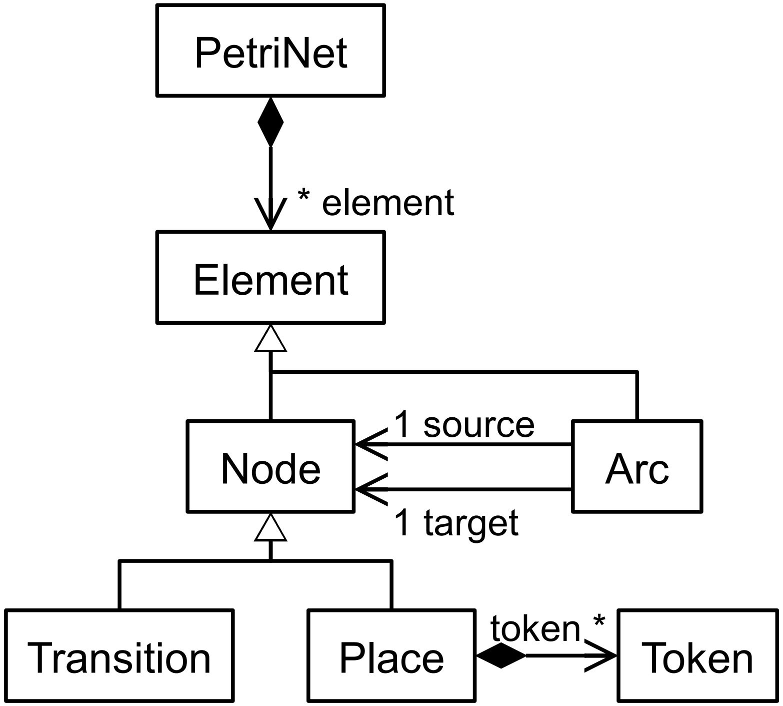


Class models are models, too!

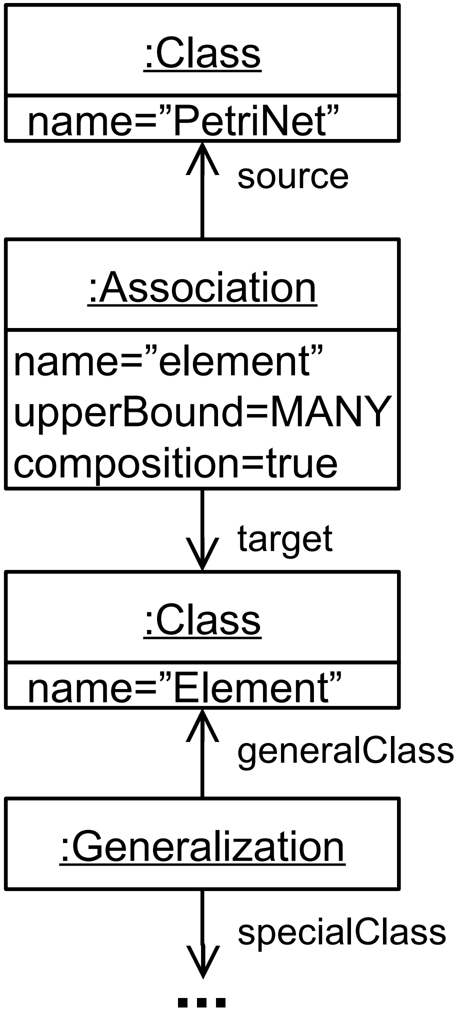




Class models are models, too!

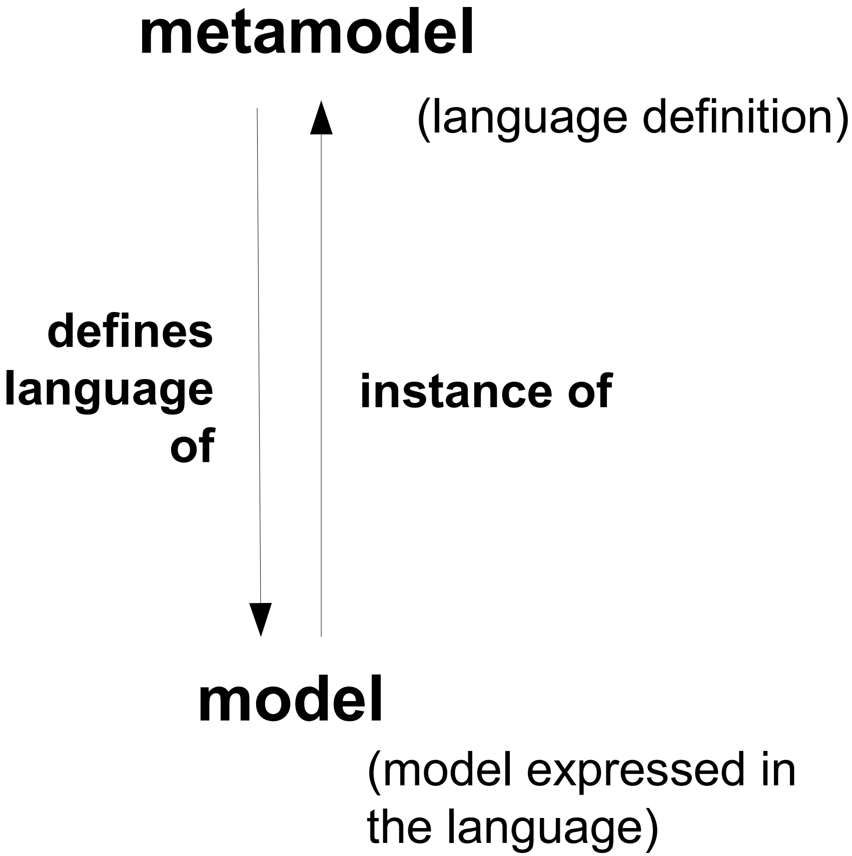


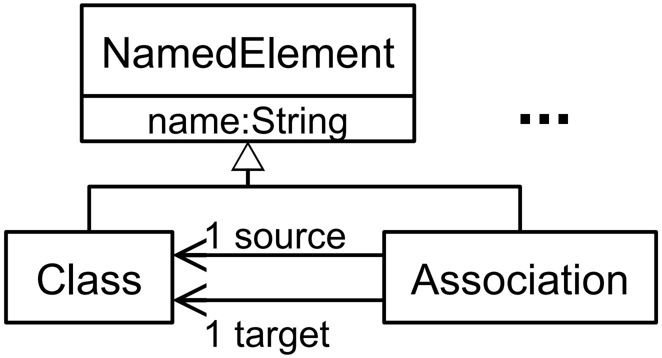
concrete syntax

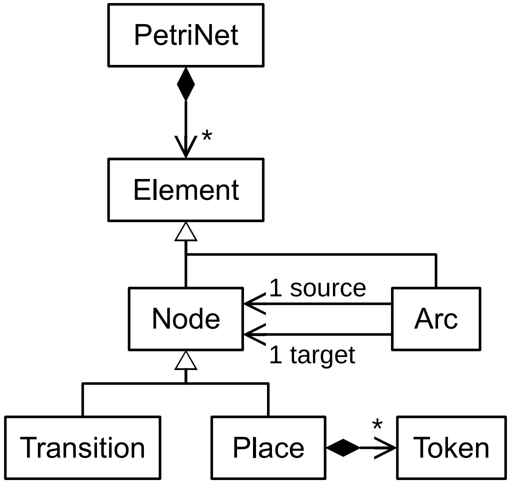


abstract syntax

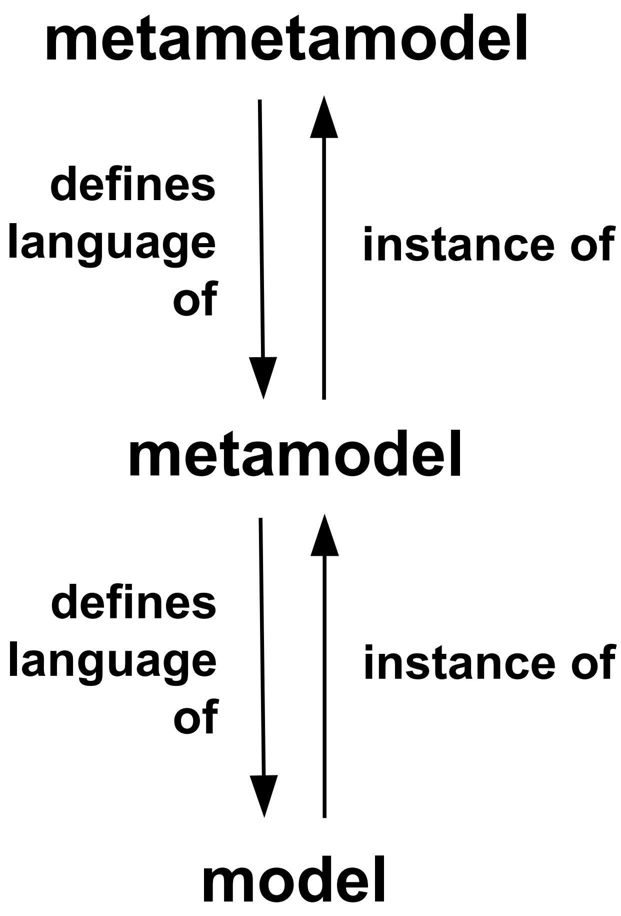
Model and Metamodel (Class models are models, too!)







Multiple Meta-Levels



metametamodel

defines language of

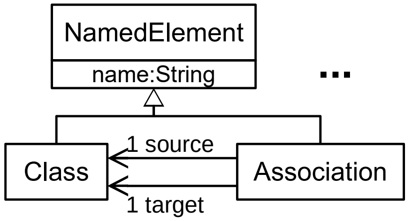
instance of

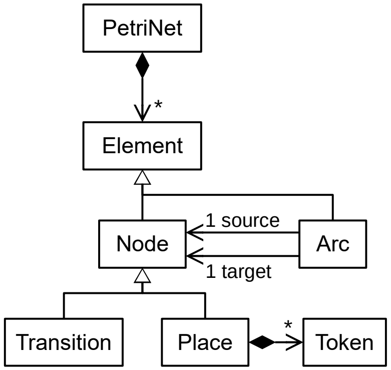
metamodel

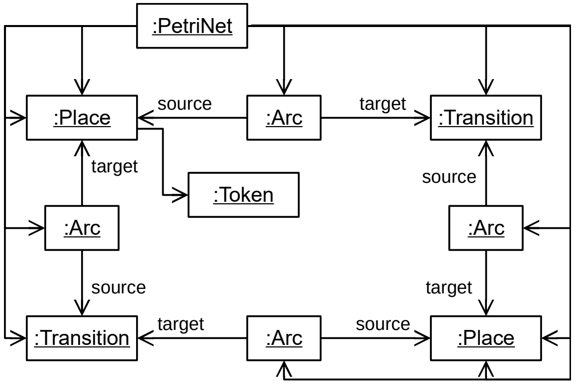
defines language of

instance of

model

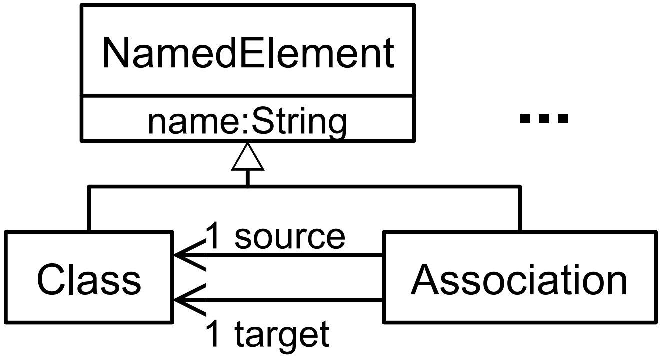






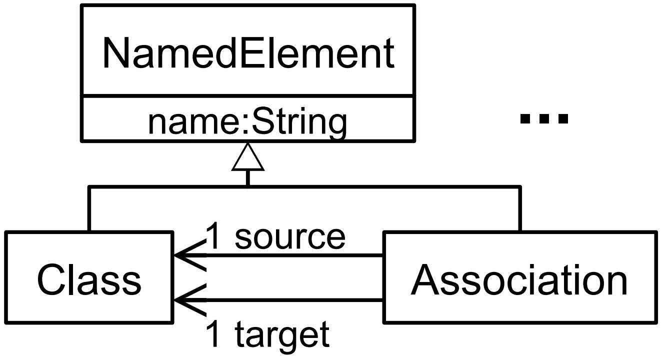
Meta-Levels

- Is there also a metamodel for this model?



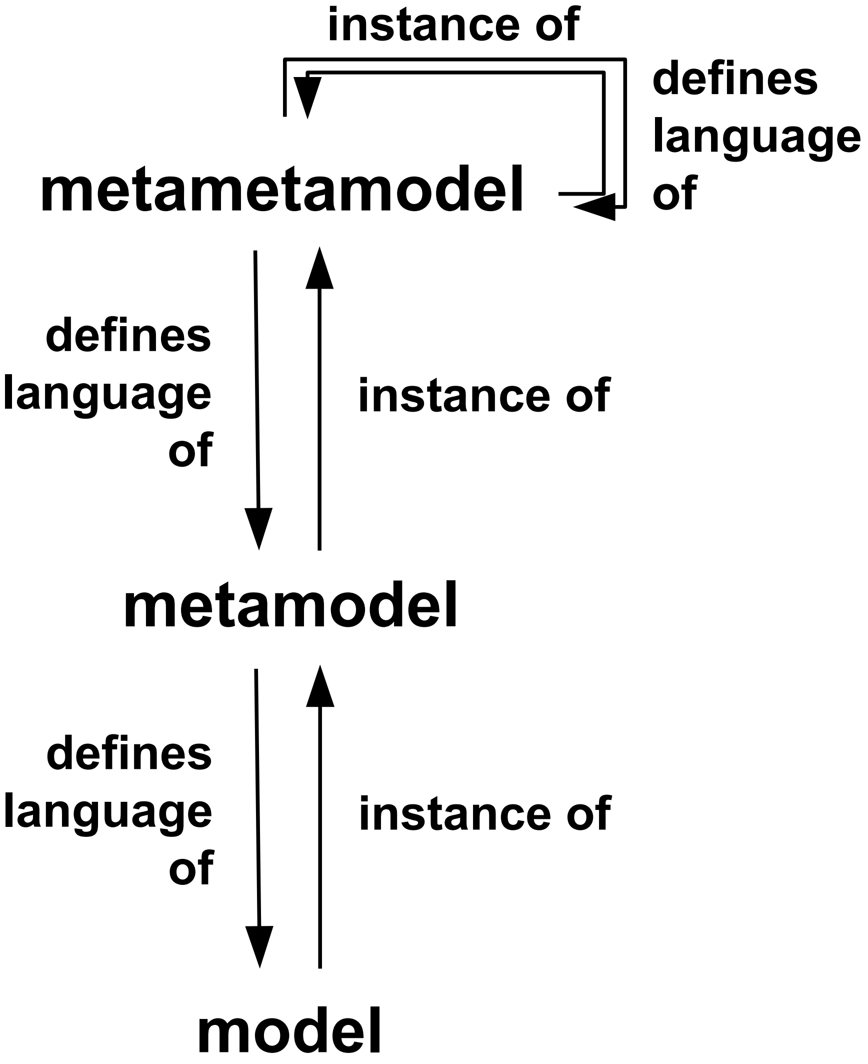
Meta-Levels

- Is there also a metamodel for this model?



- **It can describe itself!**

Meta-Levels



instance of

defines language of

metametamodel

defines language of

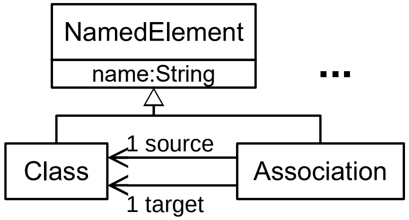
instance of

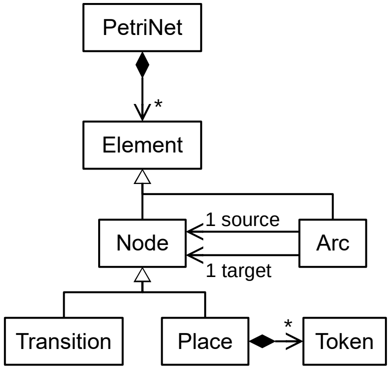
metamodel

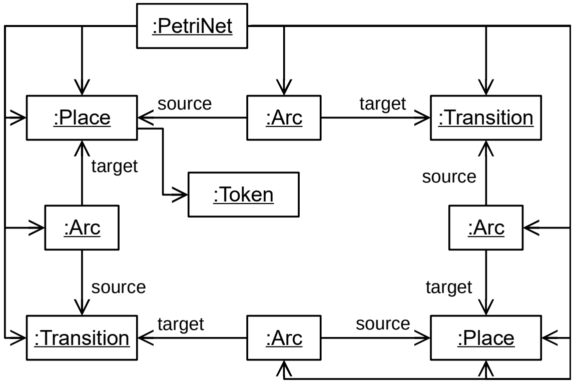
defines language of

instance of

model







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/)

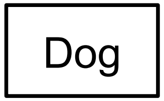
M3	**meta-metamodel** to define metamodels on M2, also describes itself
M2	**metamodels**, for defining a modeling language on M1
M1	**models** of data or processes
M0	**instance-model,** concrete data

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

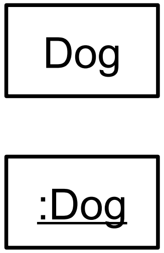


Dog

dog class

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

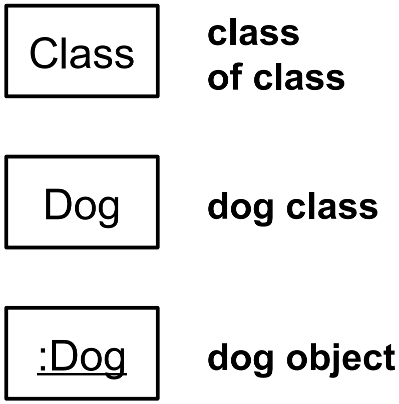


dog class

dog object

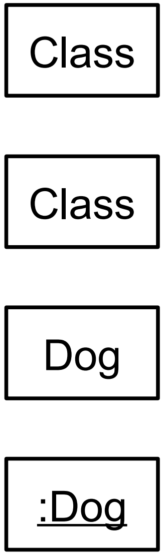
Meta-Levels for UML

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



class of class

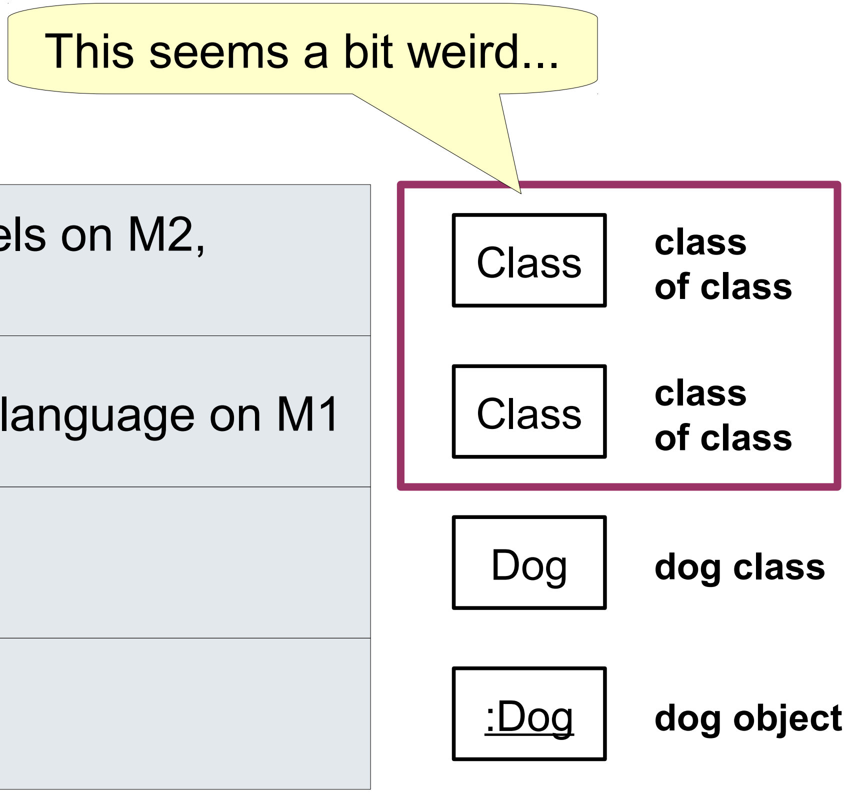
class of class

dog class

dog object

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



This seems a bit weird...

Class — **class of class**

Class — **class of class**

Dog — **dog class**

:Dog — **dog object**

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

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

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

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

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

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, …

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

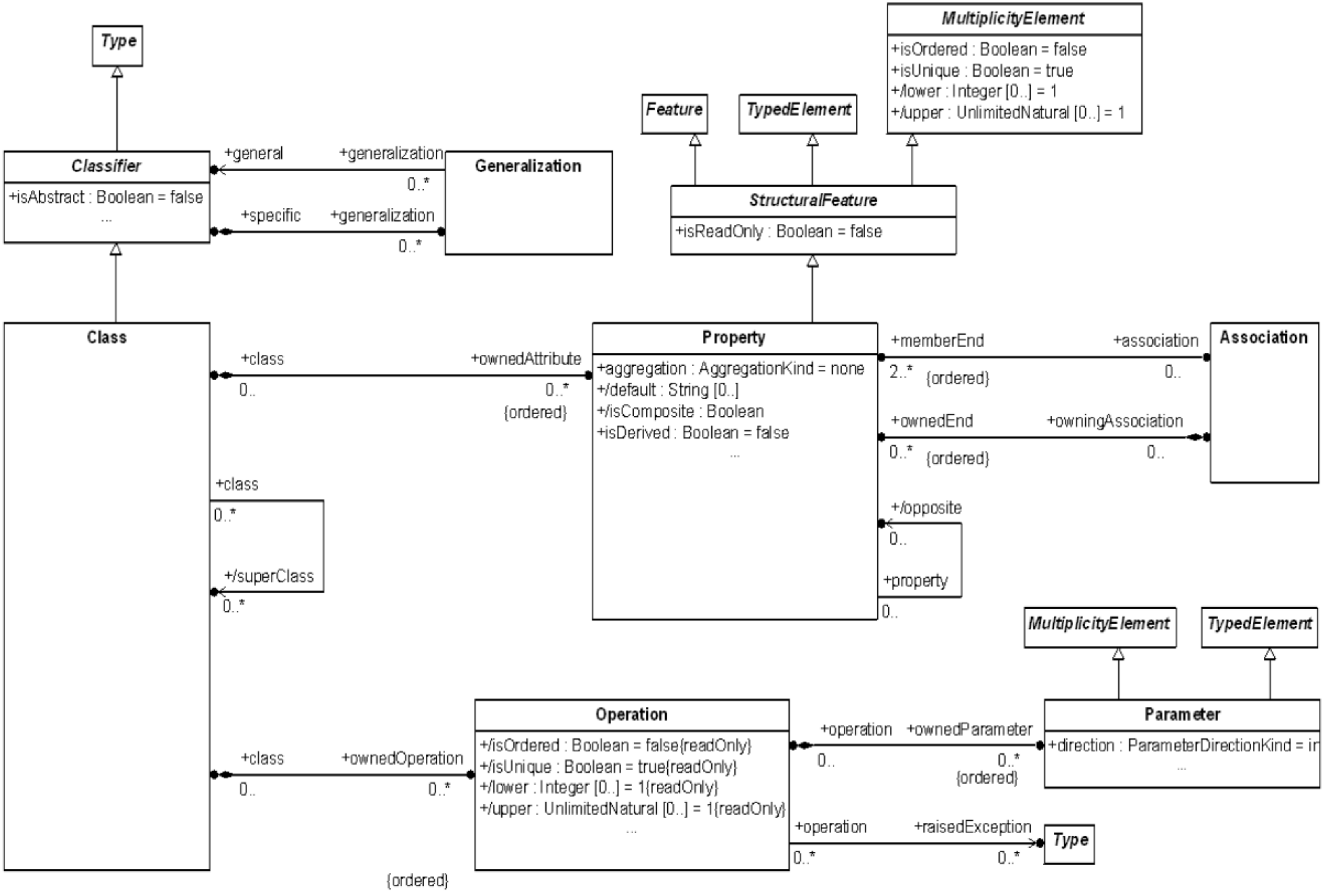
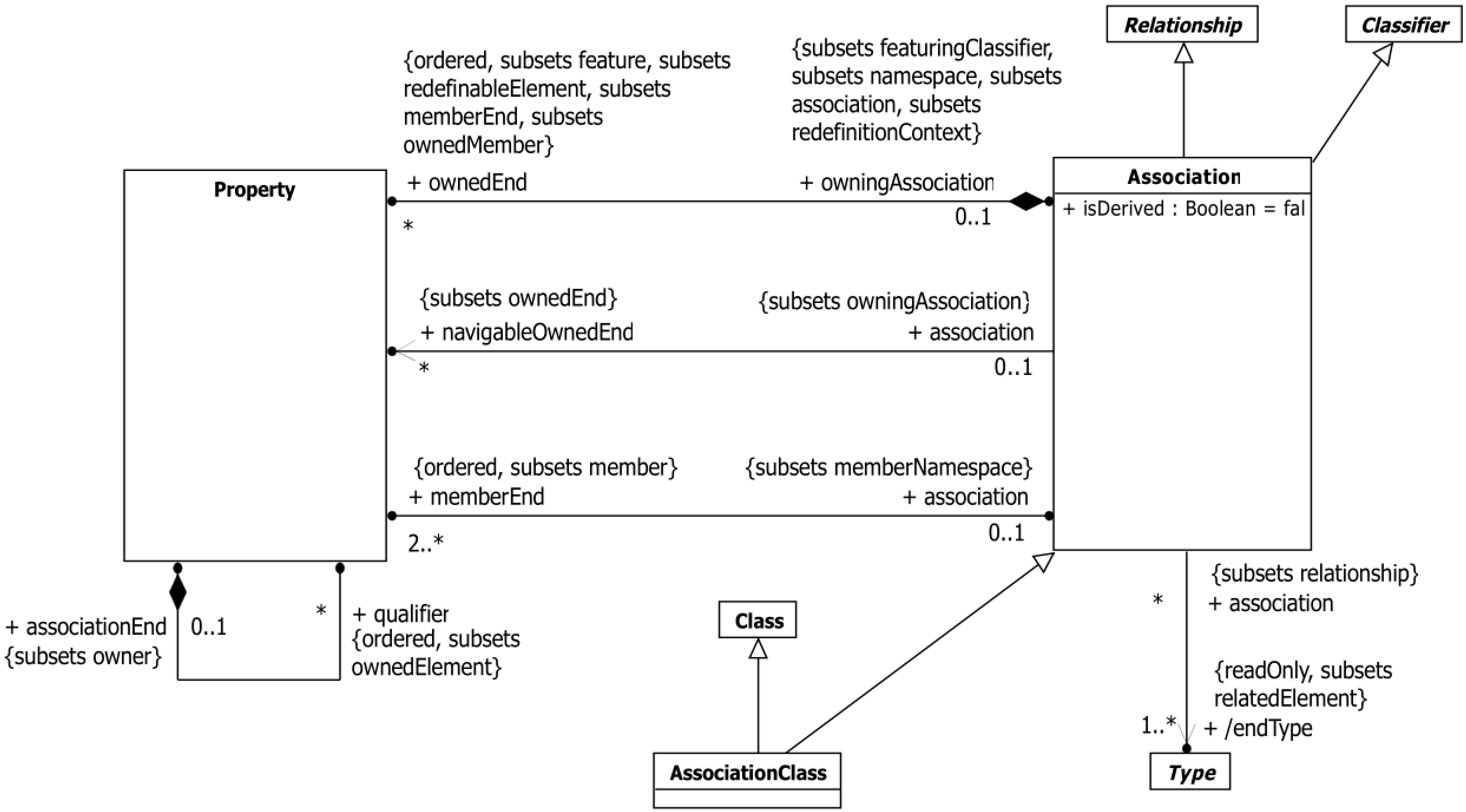


Figure 12.2 - EMOF Classes

(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

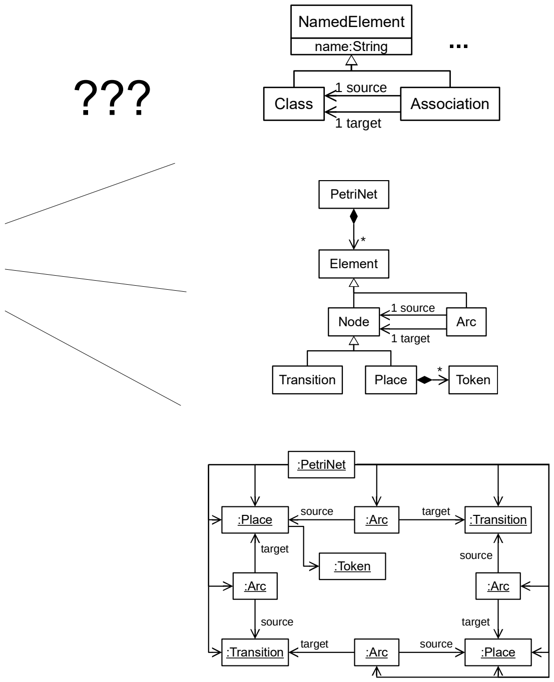
Figure 11.25 Associations

(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

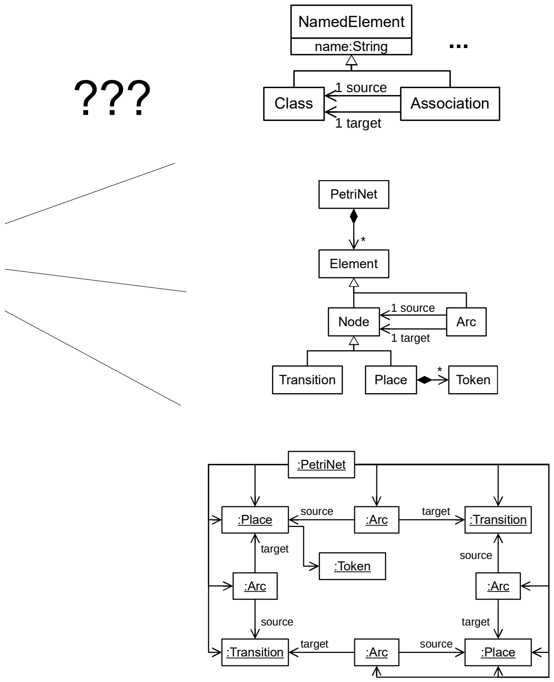
M3	**meta-metamodel** to define metamodels on M2, also describes itself
M2	**metamodels**, for defining a modeling language on M1
M1	**models** of data or processes
M0	**instance-model,** concrete data



Meta-Levels for the Petri net

- sometimes it is difficult to allocate models and metamodels to the “four” meta-levels
  – sometimes there are more, sometimes less levels

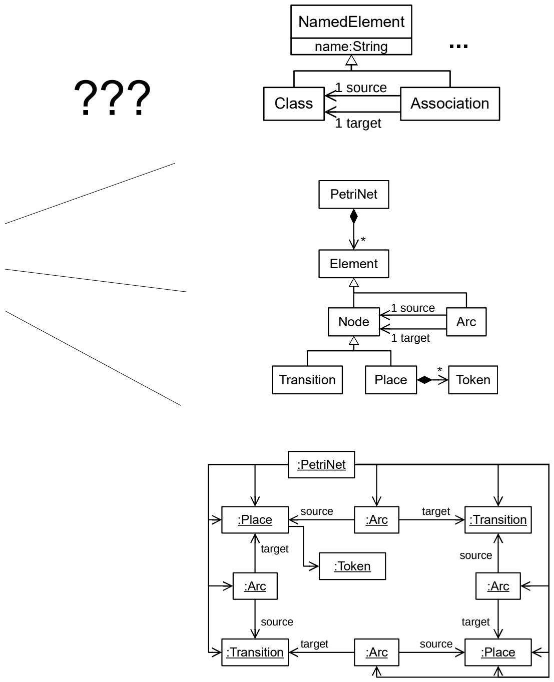
M3	**meta-metamodel** to define metamodels on M2, also describes itself
M2	**metamodels**, for defining a modeling language on M1
M1	**models** of data or processes
M0	**instance-model**, concrete data



Meta-Levels for the Petri net

- 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

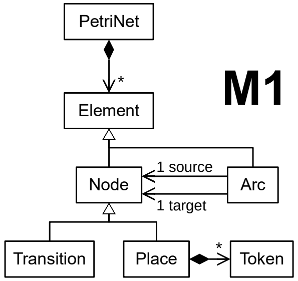


Meta-Levels for the Petri net

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

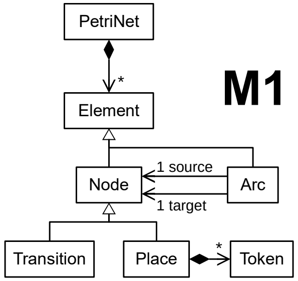
Meta-Levels for the Petri net

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

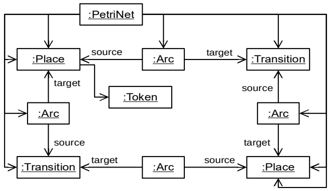


Meta-Levels for the Petri net

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



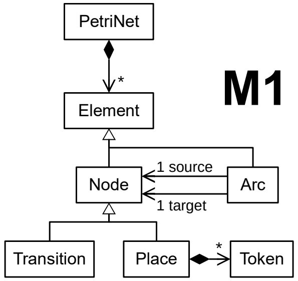
M1



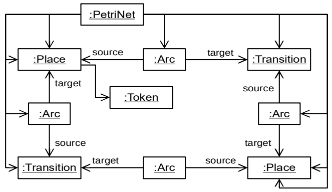
M0

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



M1

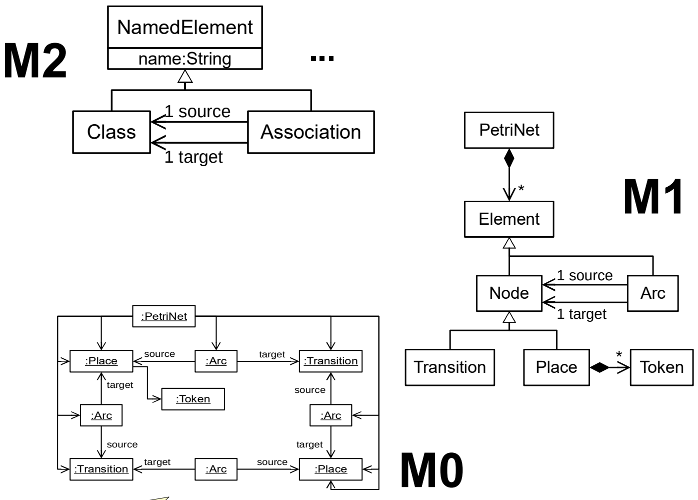


M0

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

Meta-Levels for the Petri net

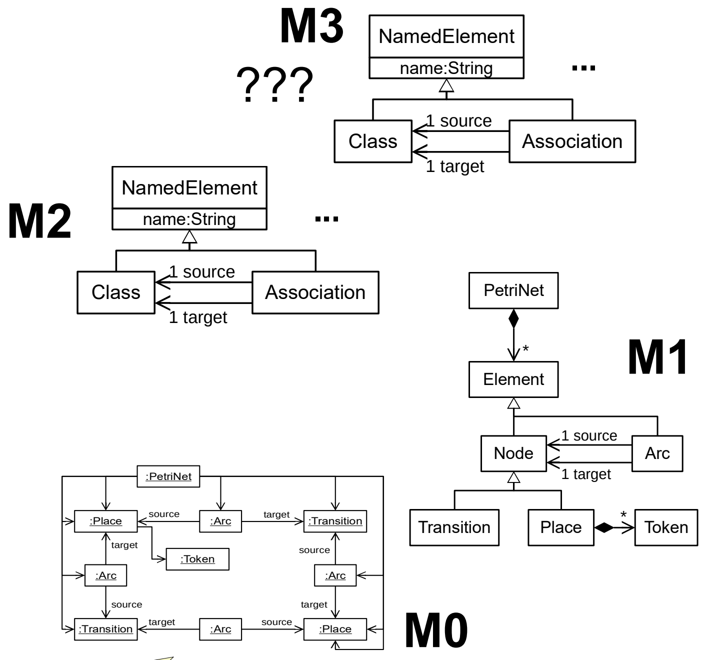
M3	**meta-metamodel** to define metamodels on M2, also describes itself
M2	**metamodels**, for defining a modeling language on M1
M1	**models** of data or processes
M0	**instance-model**, concrete data



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

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

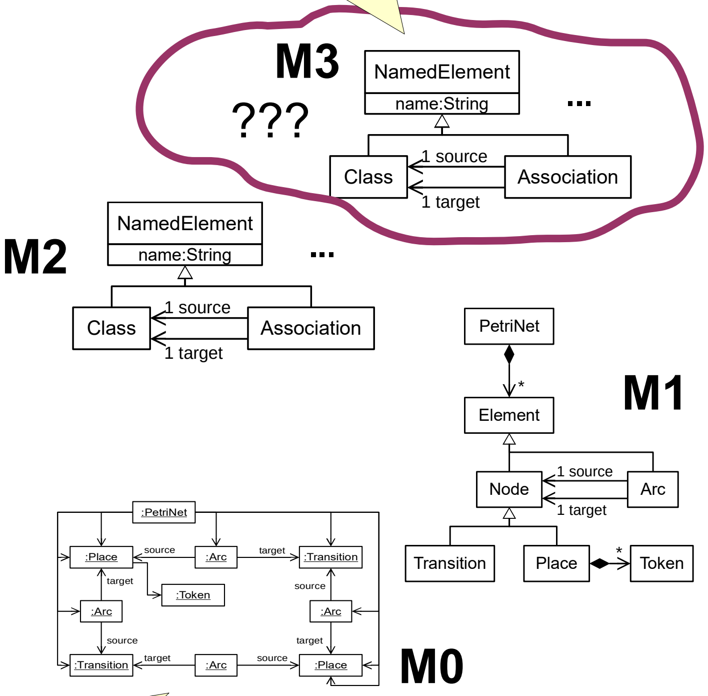


concrete instance of a Petri net
(e.g. diagram in an editor)

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

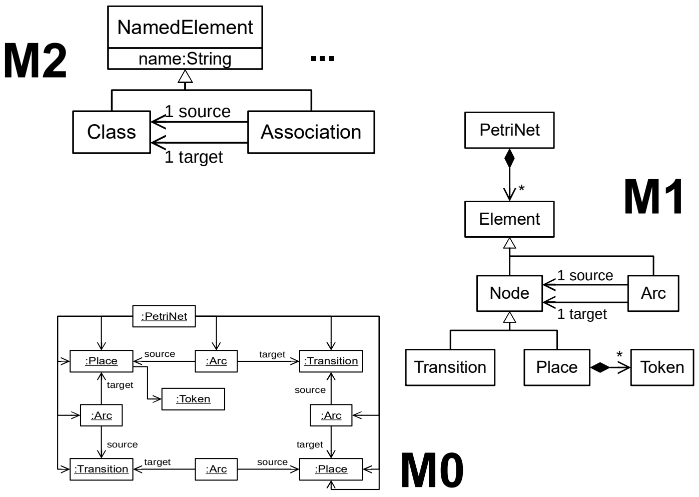


concrete instance of a Petri net
(e.g. diagram in an editor)

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

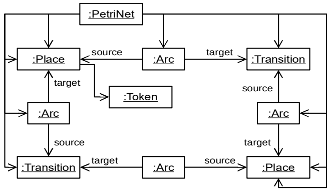


Meta-Levels for the Petri net

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

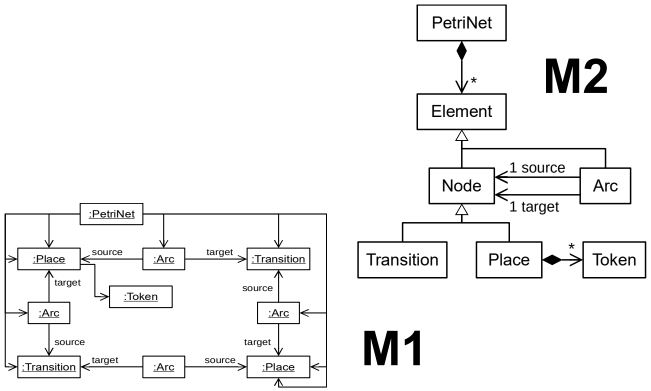
M3	**meta-metamodel** to define metamodels on M2, also describes itself
M2	**metamodels**, for defining a modeling language on M1
M1	**models** of data or processes
M0	**instance-model,** concrete data



M1

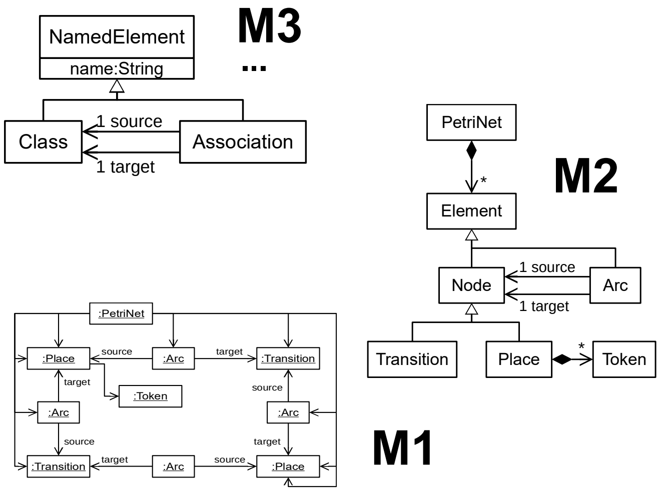
Meta-Levels for the Petri net

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



Meta-Levels for the Petri net

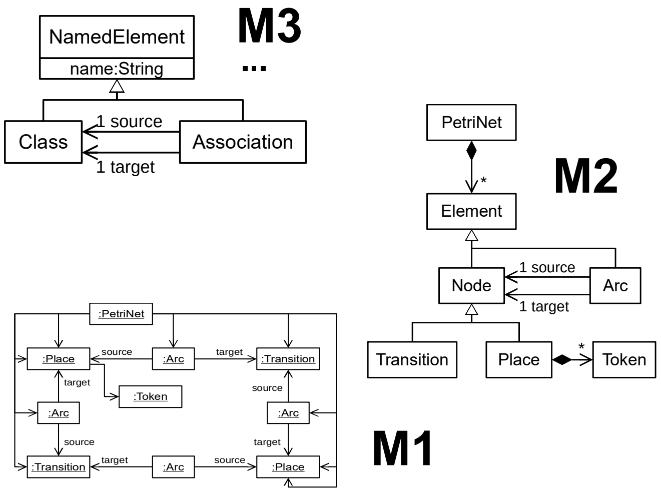
M3	**meta-metamodel** to define metamodels on M2, also describes itself
M2	**metamodels**, for defining a modeling language on M1
M1	**models** of data or processes
M0	**instance-model,** concrete data



Meta-Levels for the Petri net

Fits better into M1-M3

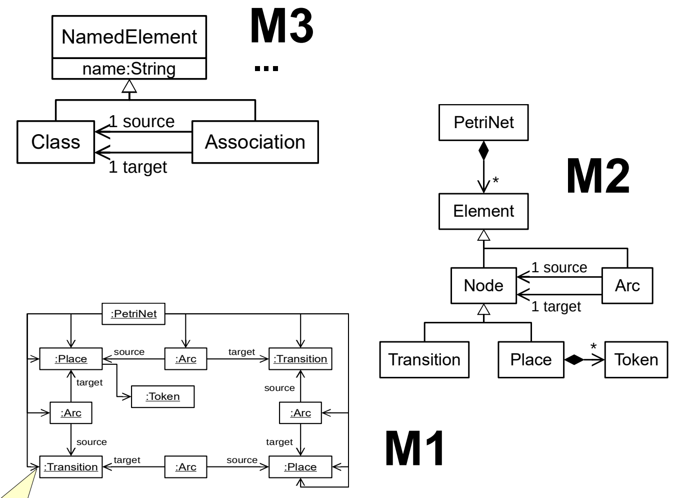
M3	**meta-metamodel** to define metamodels on M2, also describes itself
M2	**metamodels**, for defining a modeling language on M1
M1	**models** of data or processes
M0	**instance-model**, concrete data



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

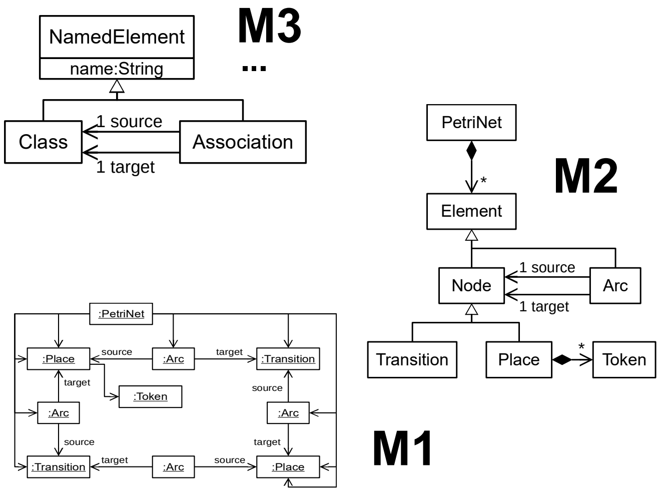


Especially: a Petri net is a *model of a process* – so, by definition of M1, it fits nicely in M1!

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



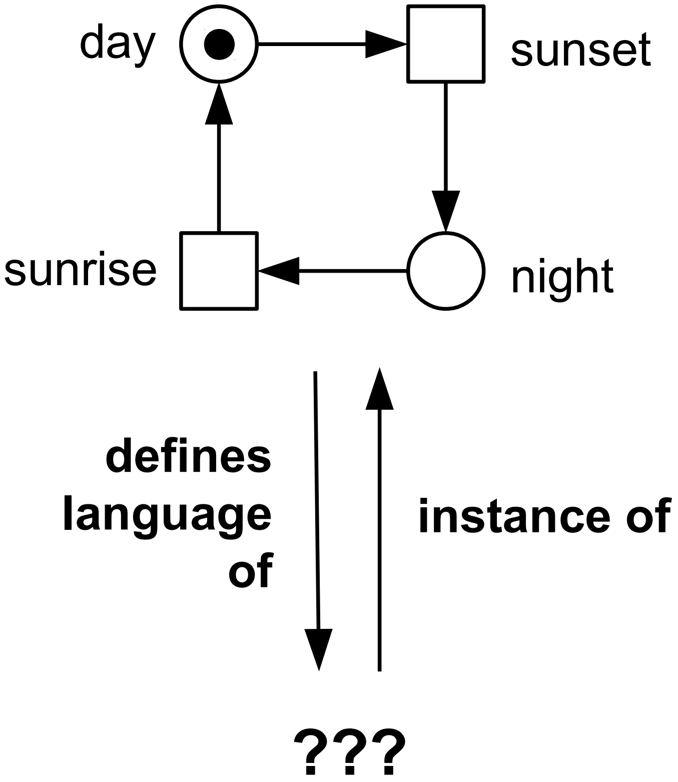
???

M0

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?



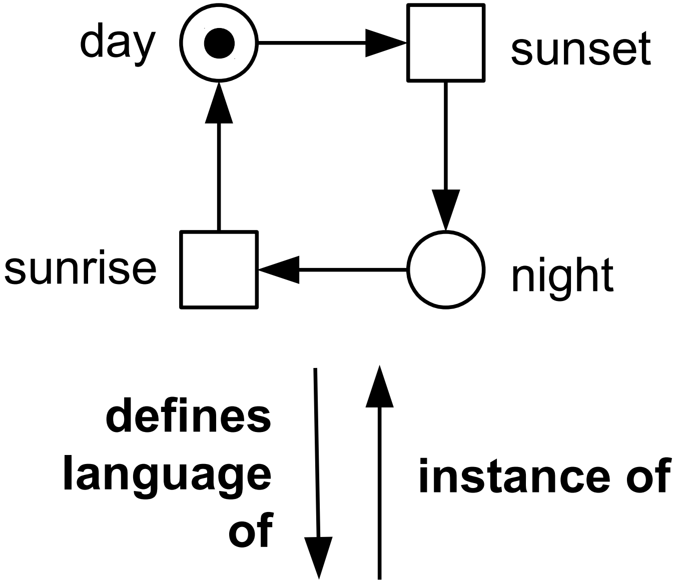
defines language of

instance of

???

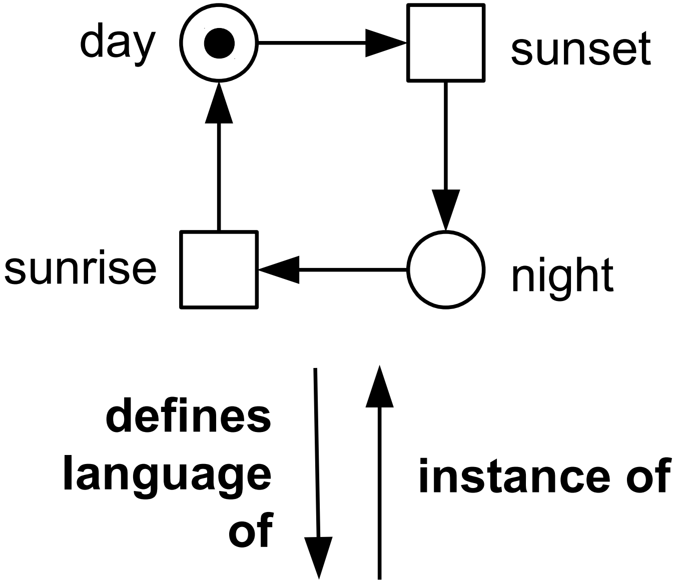
Meta-Levels for the Petri net

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



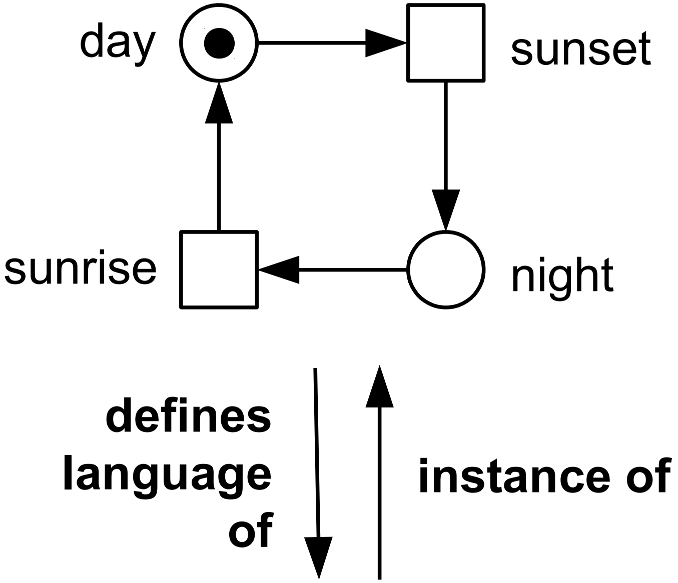
Meta-Levels for the Petri net

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



Meta-Levels for the Petri net

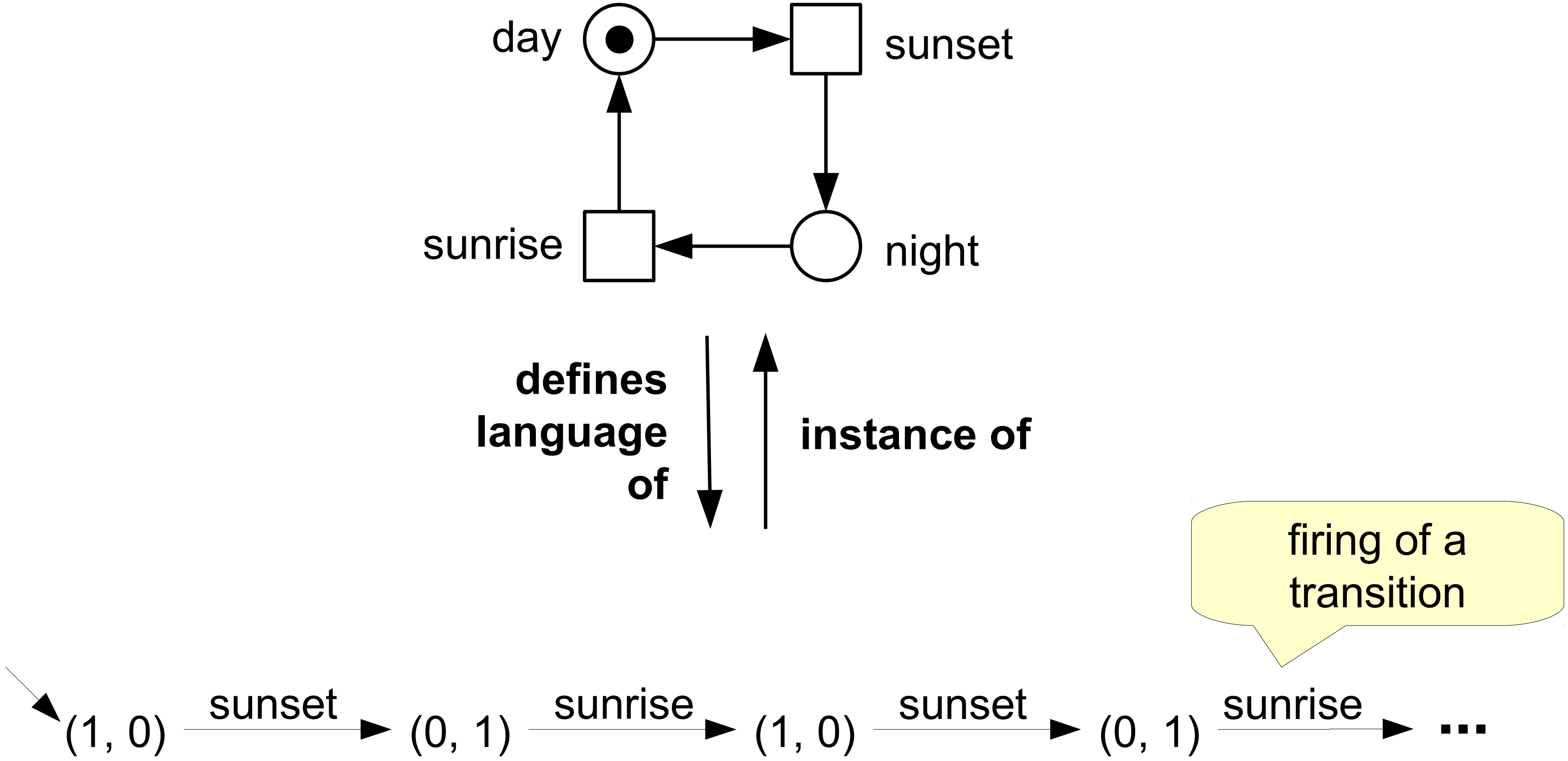
- What is the language defined by a Petri net?
  - the set of all its executions!



$$\to (1, 0) \xrightarrow{\text{sunset}} (0, 1) \xrightarrow{\text{sunrise}} (1, 0) \xrightarrow{\text{sunset}} (0, 1) \xrightarrow{\text{sunrise}} \ldots$$

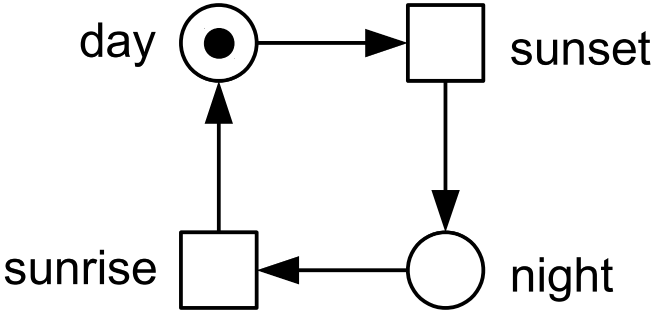
Meta-Levels for the Petri net

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



Meta-Levels for the Petri net

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



defines language of

instance of

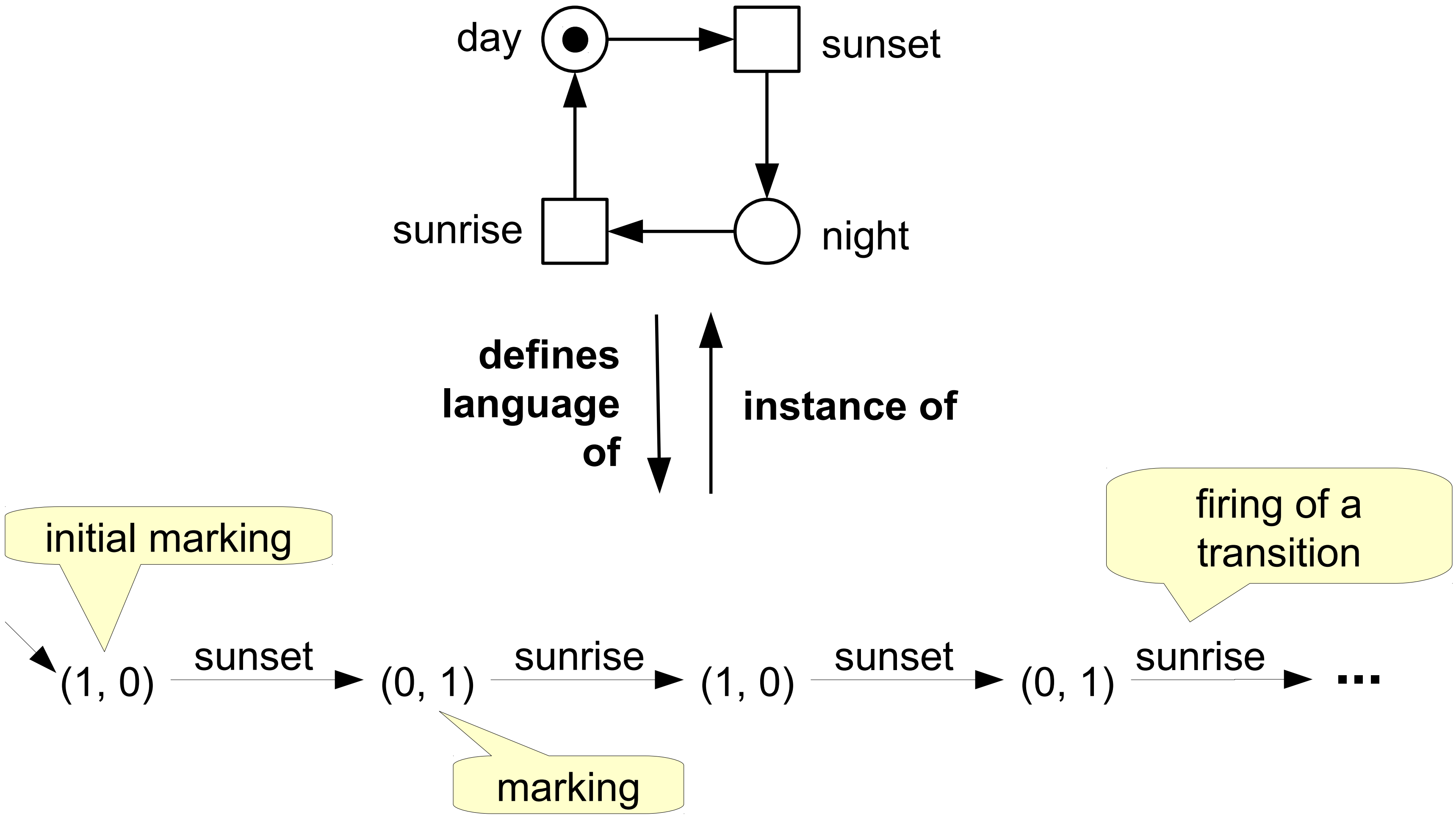
firing of a transition

$$\rightarrow (1, 0) \xrightarrow{\text{sunset}} (0, 1) \xrightarrow{\text{sunrise}} (1, 0) \xrightarrow{\text{sunset}} (0, 1) \xrightarrow{\text{sunrise}} \ldots$$

marking

Meta-Levels for the Petri net

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



defines language of

instance of

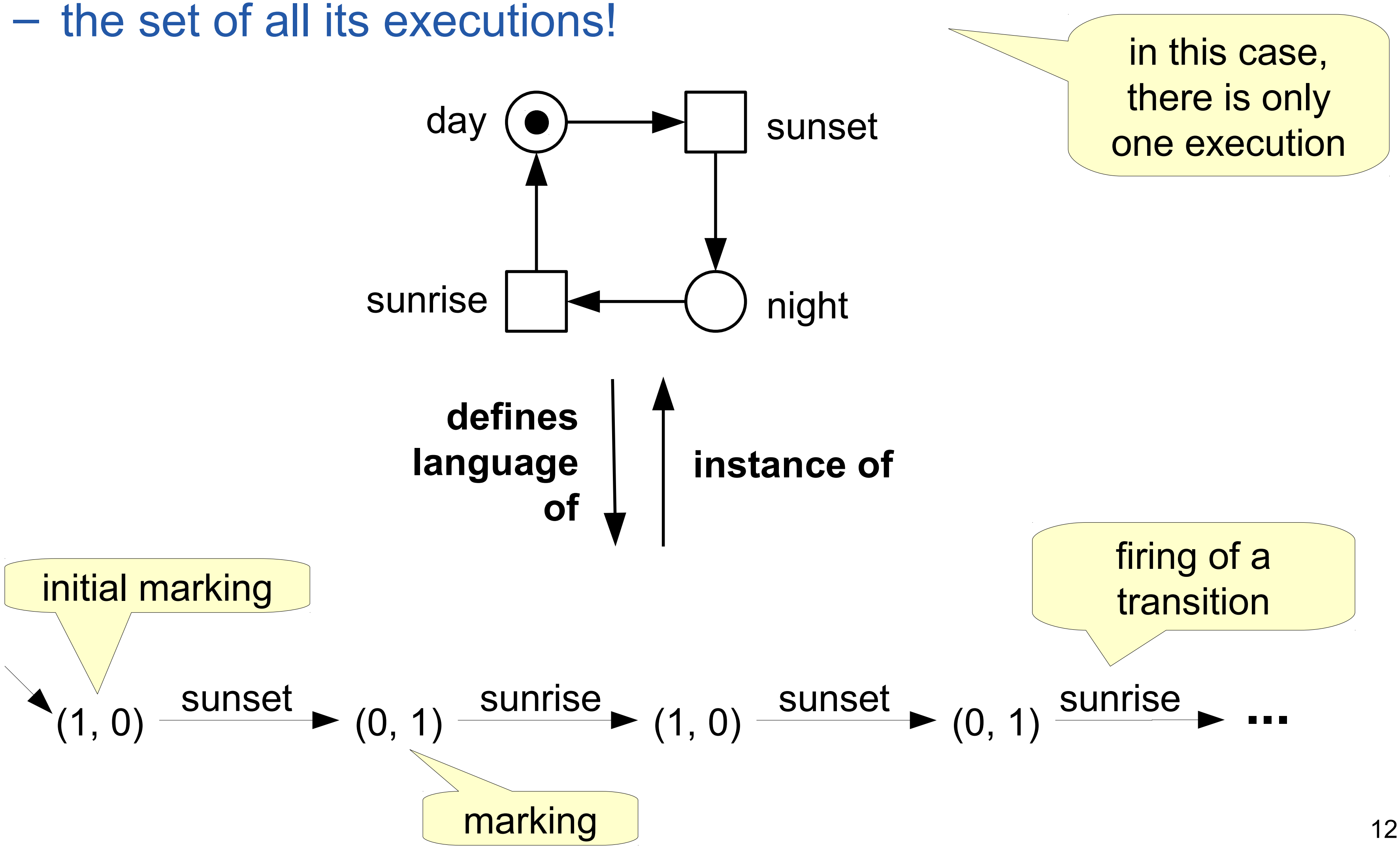
initial marking

firing of a transition

marking

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

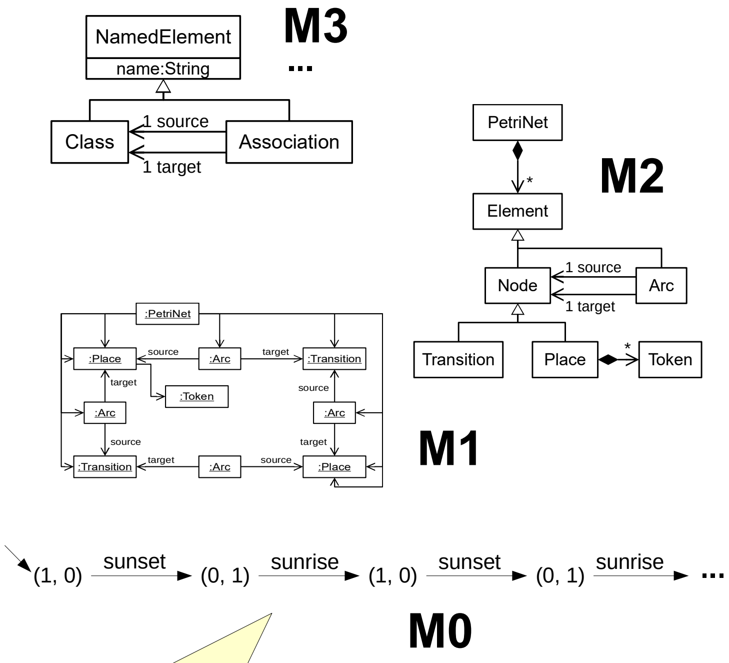
initial marking

firing of a transition

marking

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

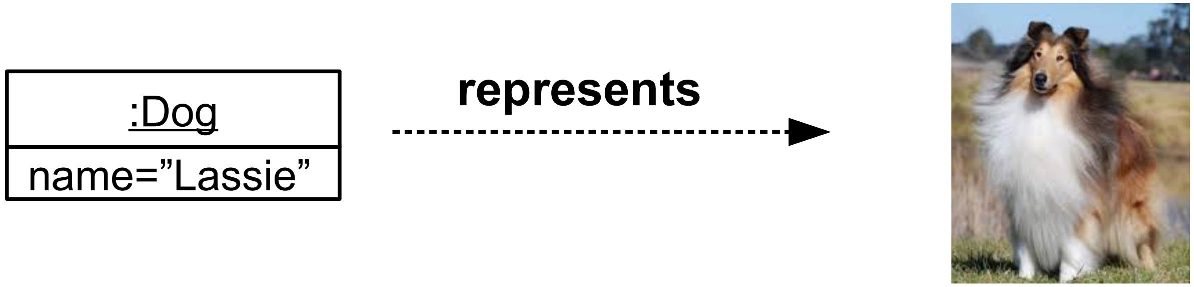
“Instance-of” vs. “Represents” Relationship

- A model **represents** an original

“Instance-of” vs. “Represents” Relationship

- A model **represents** an original

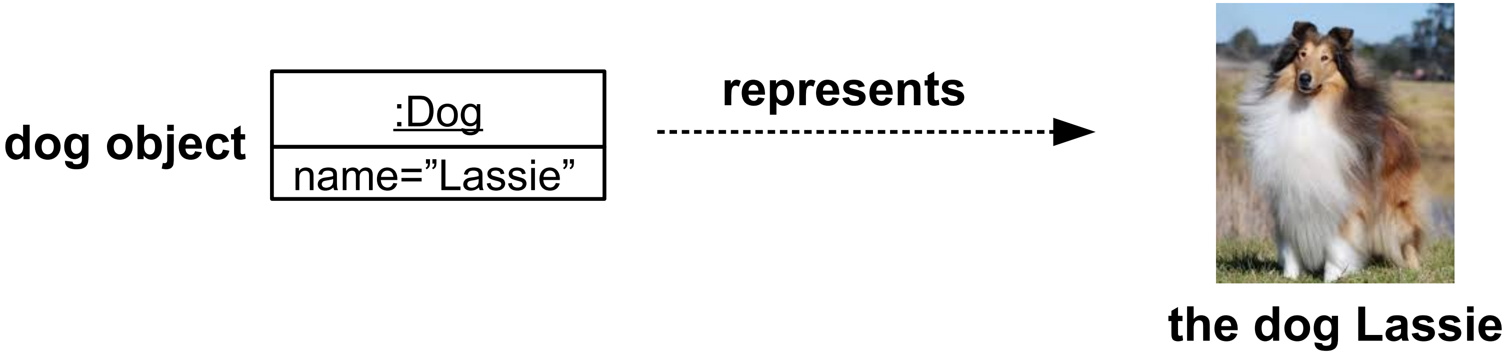
dog object



the dog Lassie

“Instance-of” vs. “Represents” Relationship

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



dog object

:Dog

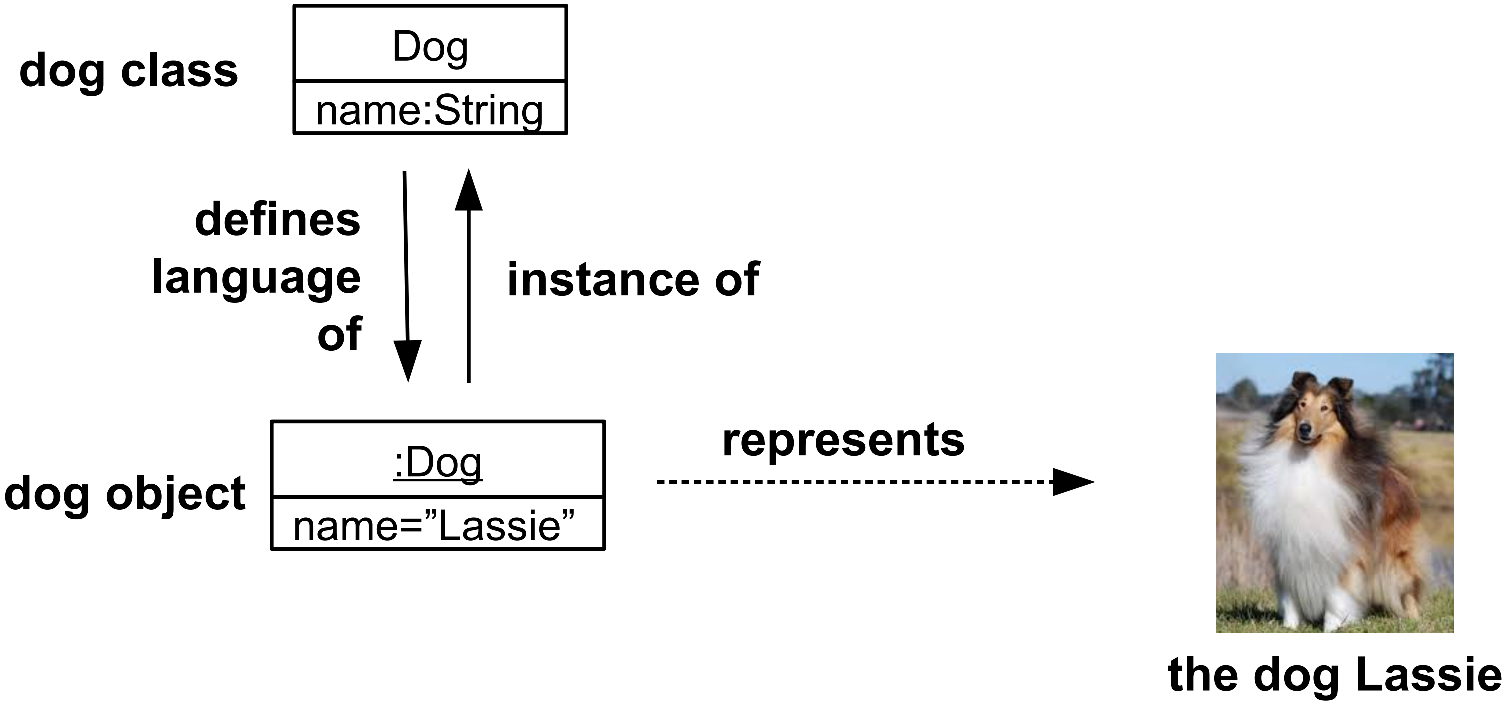
name=”Lassie”

represents

the dog Lassie

“Instance-of” vs. “Represents” Relationship

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



dog class

Dog
name:String

defines language of

instance of

dog object

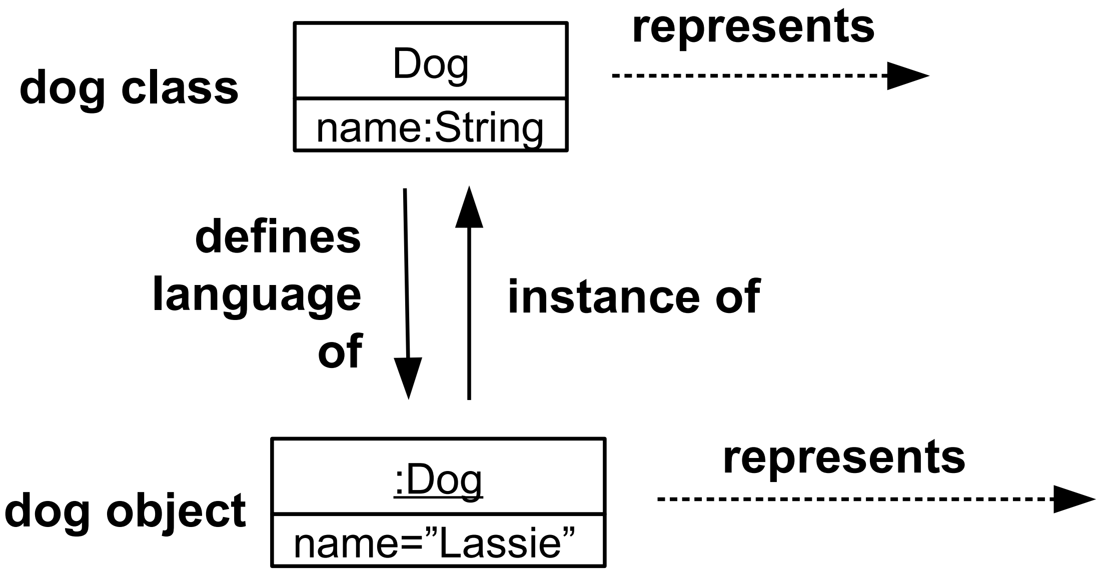
:Dog
name=”Lassie”

represents

the dog Lassie

“Instance-of” vs. “Represents” Relationship

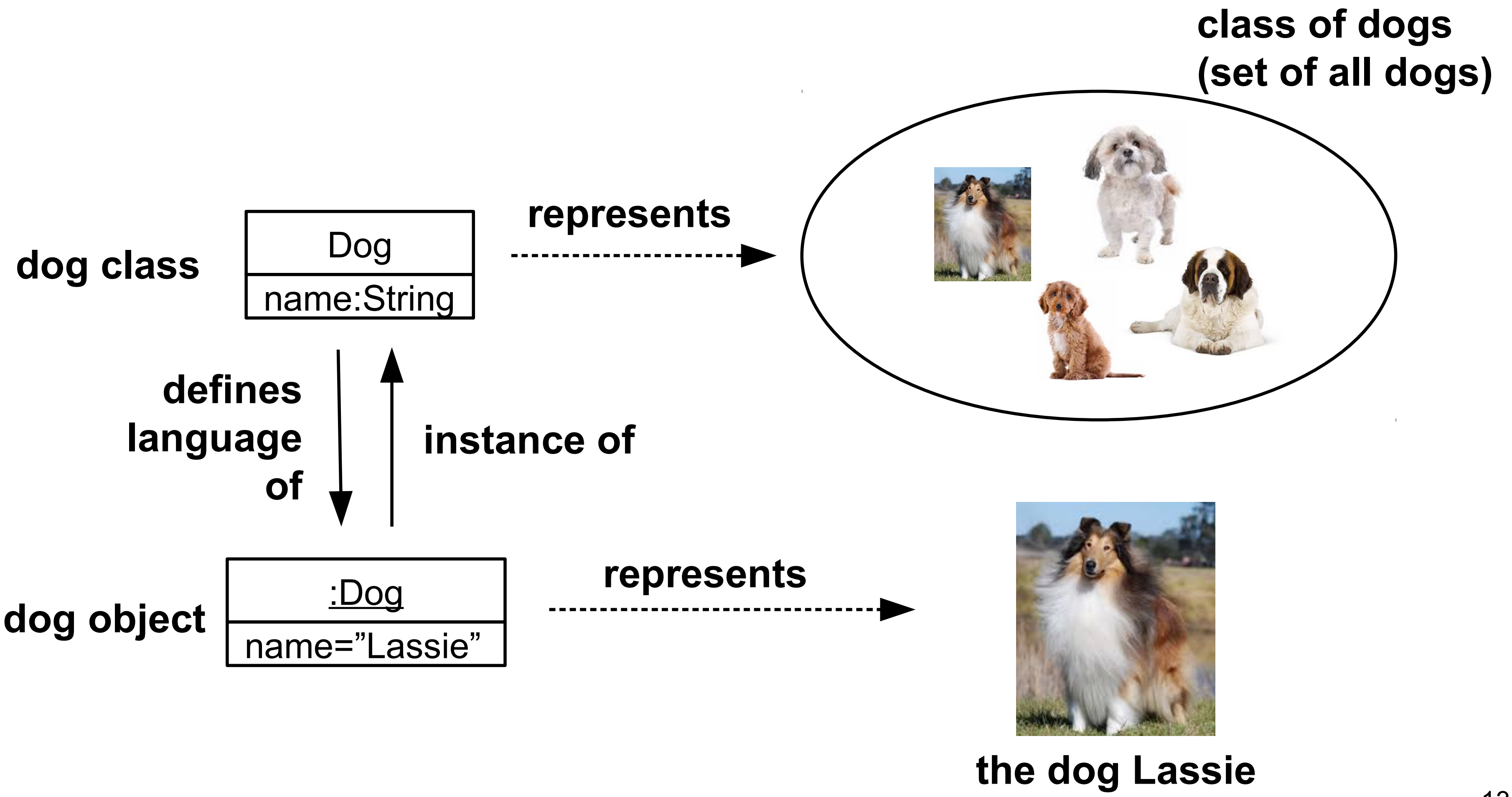
- A model **represents** an original
- A model is an **instance of** a metamodel



the dog Lassie

“Instance-of” vs. “Represents” Relationship

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



class of dogs (set of all dogs)

dog class

Dog
name:String

represents

defines language of

instance of

dog object

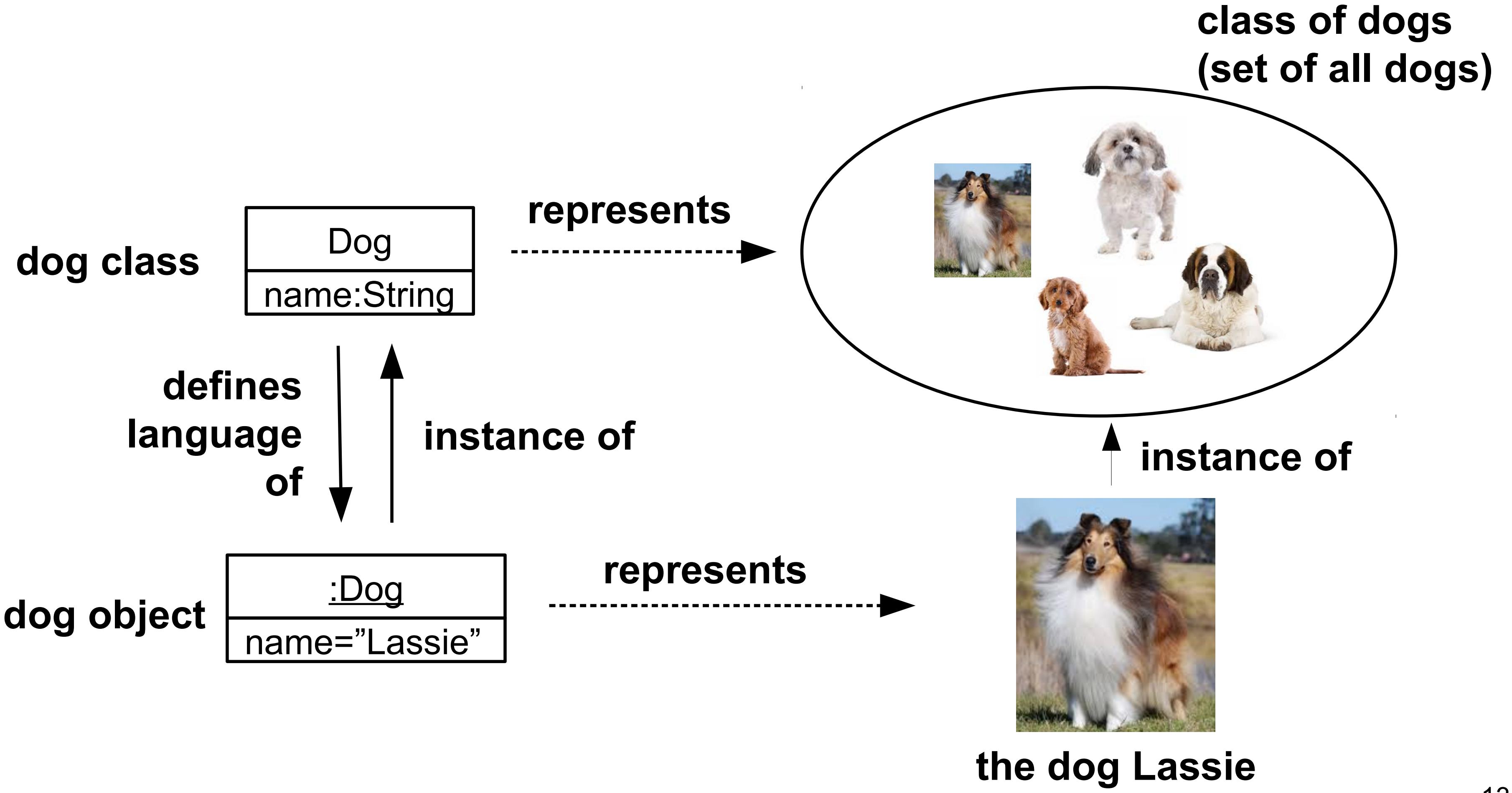
:Dog
name=”Lassie”

represents

the dog Lassie

“Instance-of” vs. “Represents” Relationship

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



class of dogs (set of all dogs)

dog class

Dog

name:String

represents

defines language of

instance of

instance of

dog object

:Dog

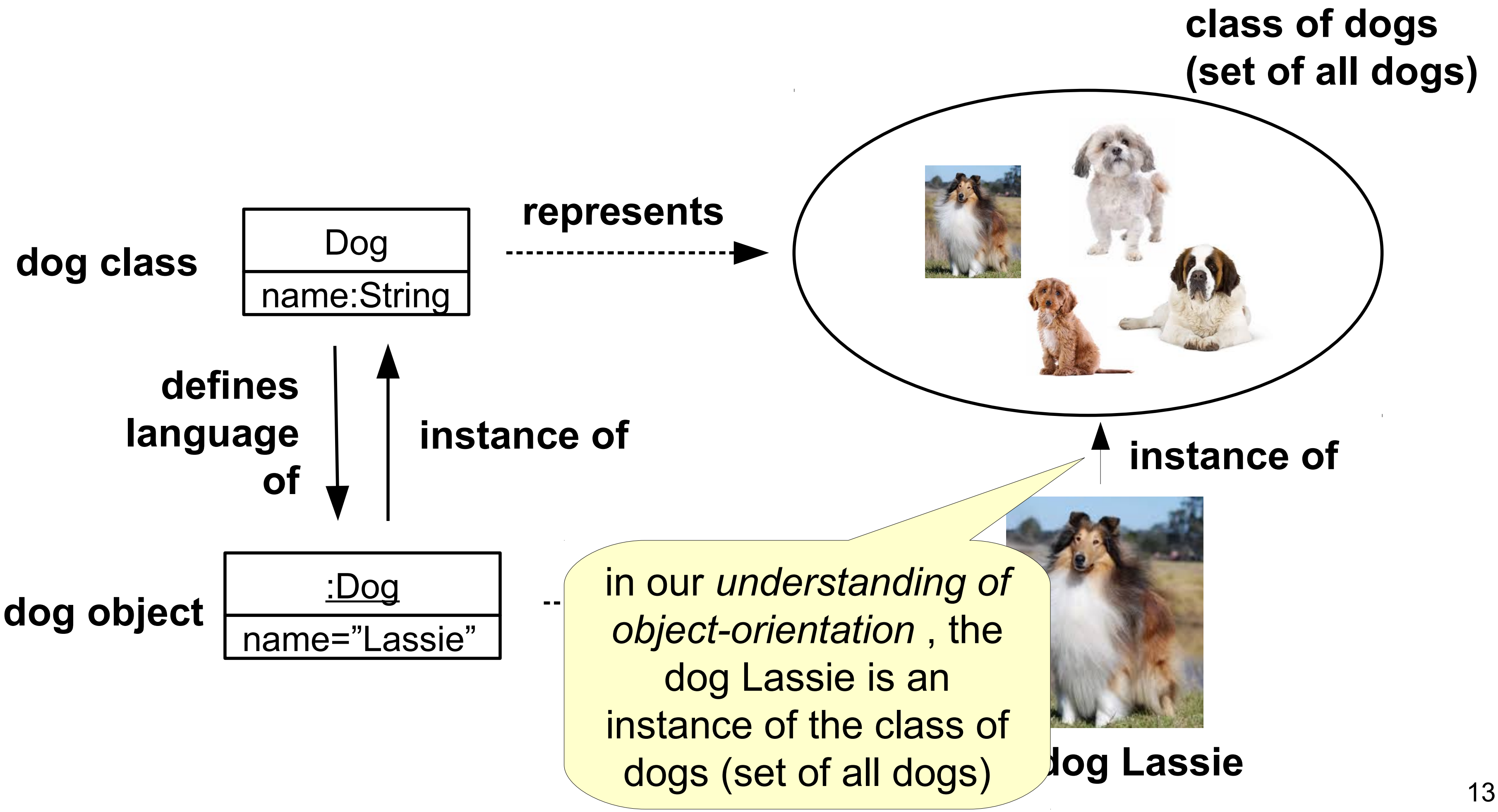
name=”Lassie”

represents

the dog Lassie

“Instance-of” vs. “Represents” Relationship

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



class of dogs (set of all dogs)

dog class

Dog

name:String

represents

defines language of

instance of

instance of

dog object

:Dog

name=”Lassie”

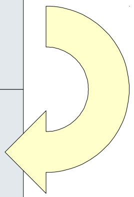
in our *understanding of object-orientation* , the dog Lassie is an instance of the class of dogs (set of all dogs)

dog Lassie

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 gone wrong

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

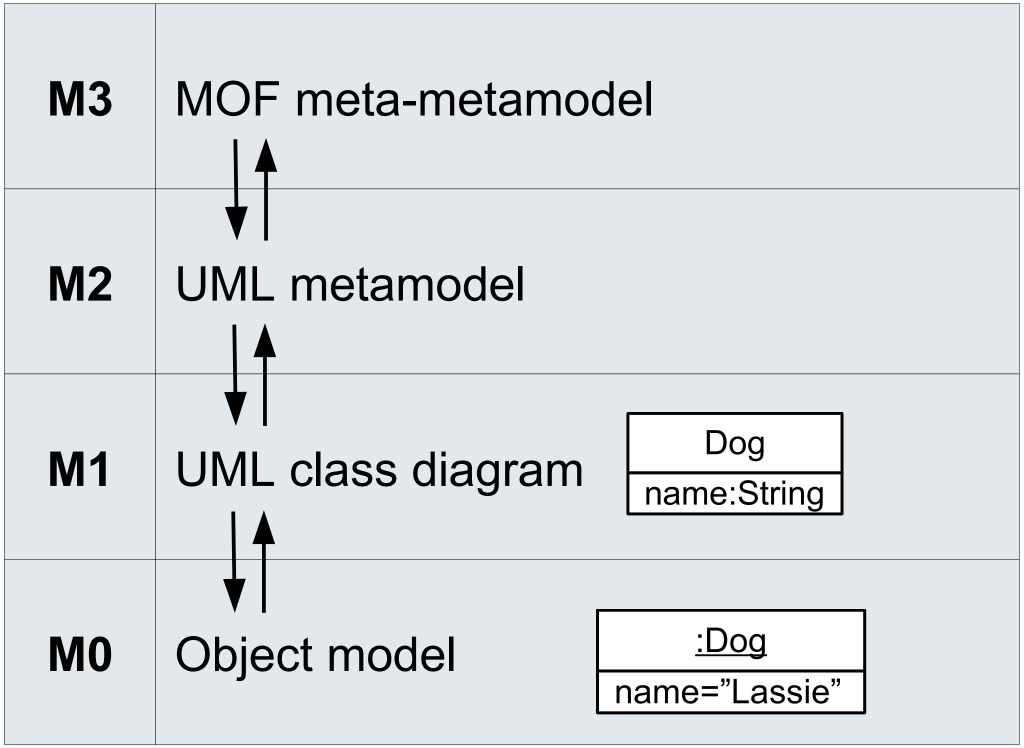
M3	**meta-metamodel** to define metamodels on M2, also describes itself
M2	**metamodels**, for defining a modeling language on M1
M1	**models** of data or processes
M0	generated code **NO!**

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



M3	MOF meta-metamodel
M2	UML metamodel
M1	UML class diagram
M0	Object model

Dog

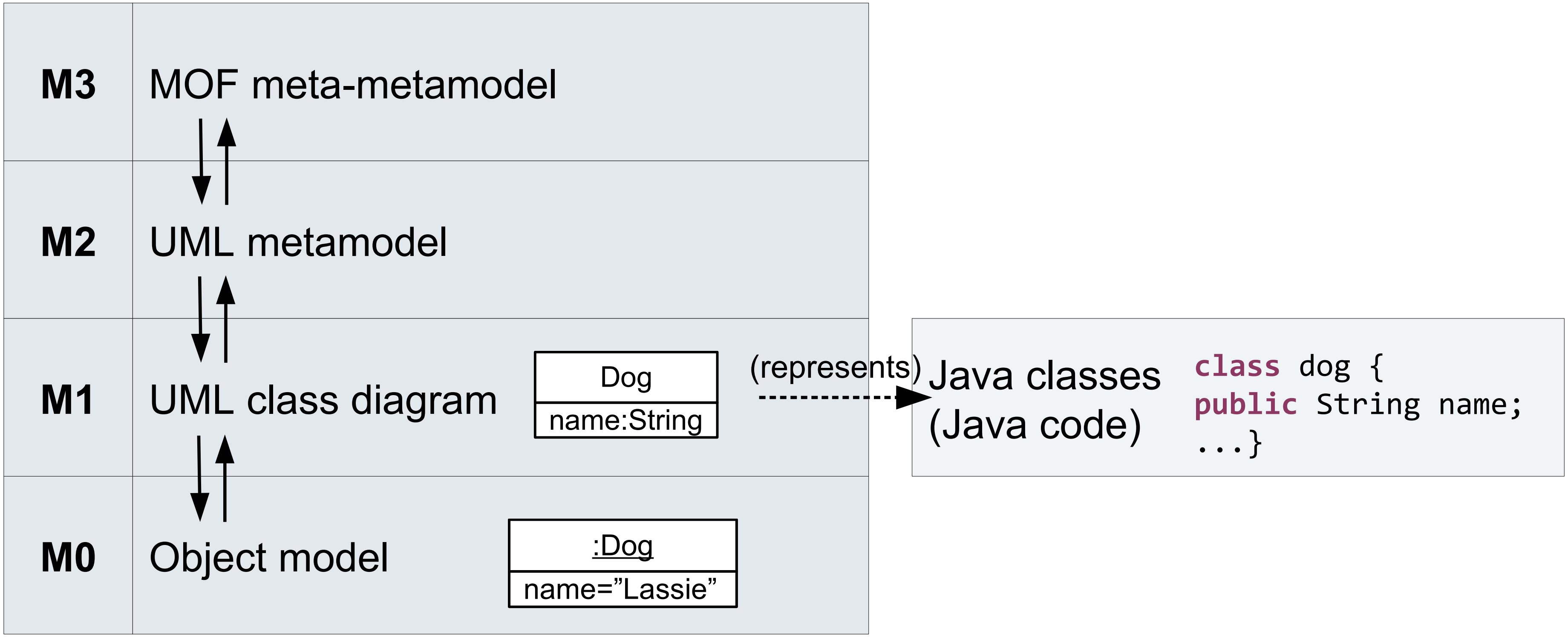
name:String

:Dog

name=”Lassie”

Meta-Levels done right!

- Classes in the class diagram describe Java classes



M3	MOF meta-metamodel
M2	UML metamodel
M1	UML class diagram
M0	Object model

Dog
name:String

:Dog
name=”Lassie”

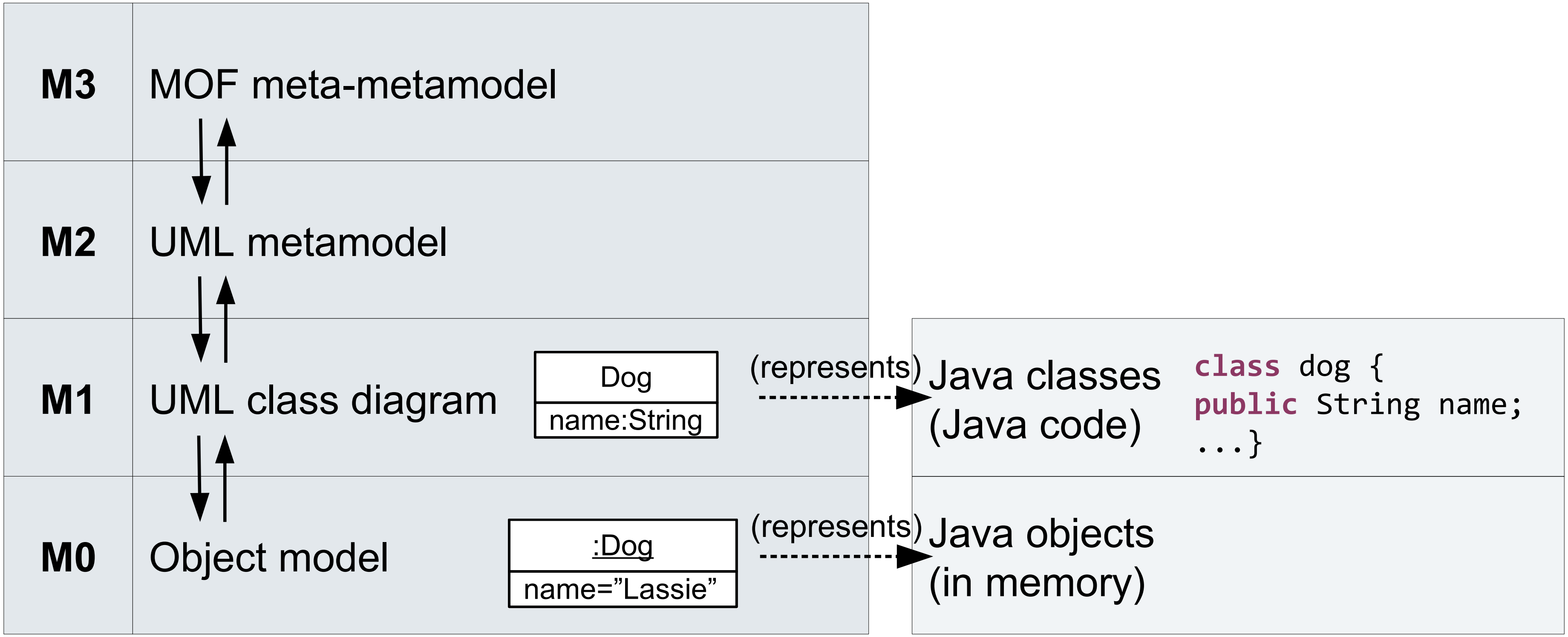
(represents)

Java classes
(Java code)

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

Meta-Levels done right!

- Classes in the class diagram describe Java classes



M3 MOF meta-metamodel

M2 UML metamodel

M1 UML class diagram

Dog

name:String

(represents)

Java classes (Java code)

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

M0 Object model

:Dog

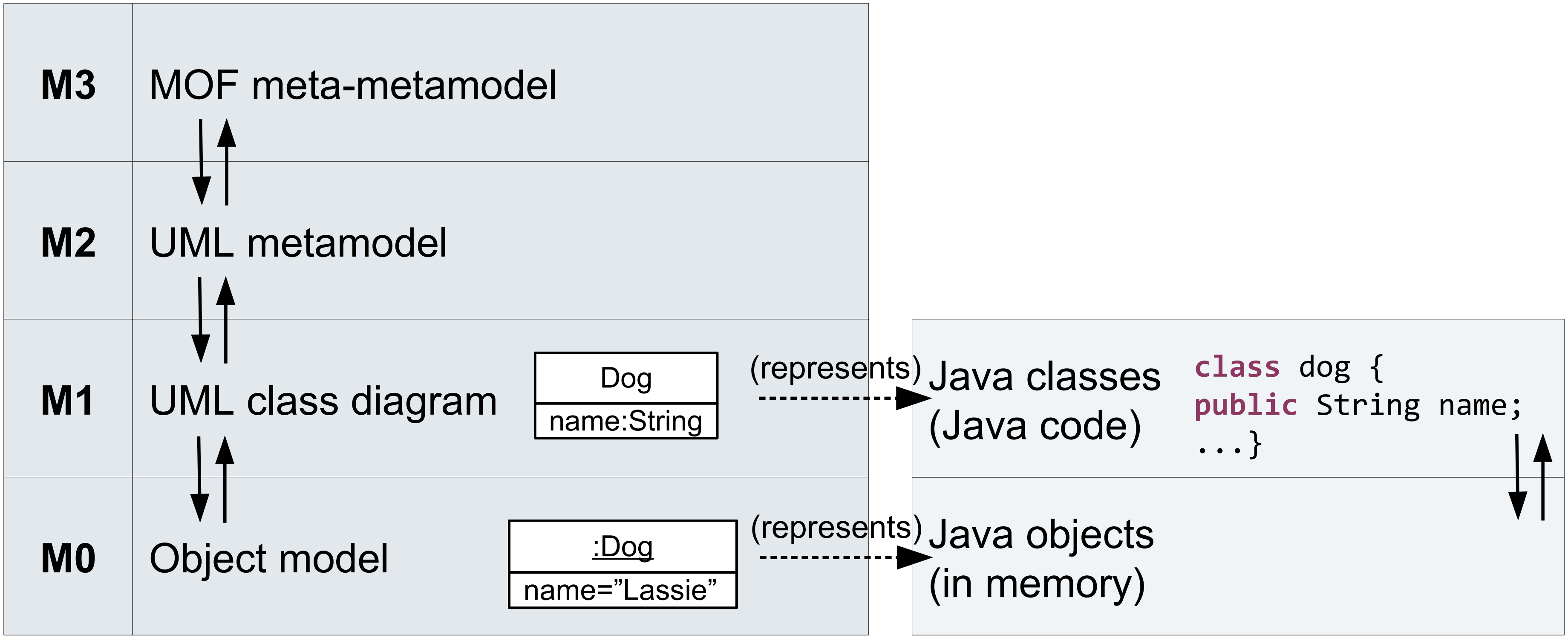
name="Lassie"

(represents)

Java objects (in memory)

Meta-Levels done right!

- Classes in the class diagram describe Java classes



M3 MOF meta-metamodel

M2 UML metamodel

M1 UML class diagram

Java classes (Java code)

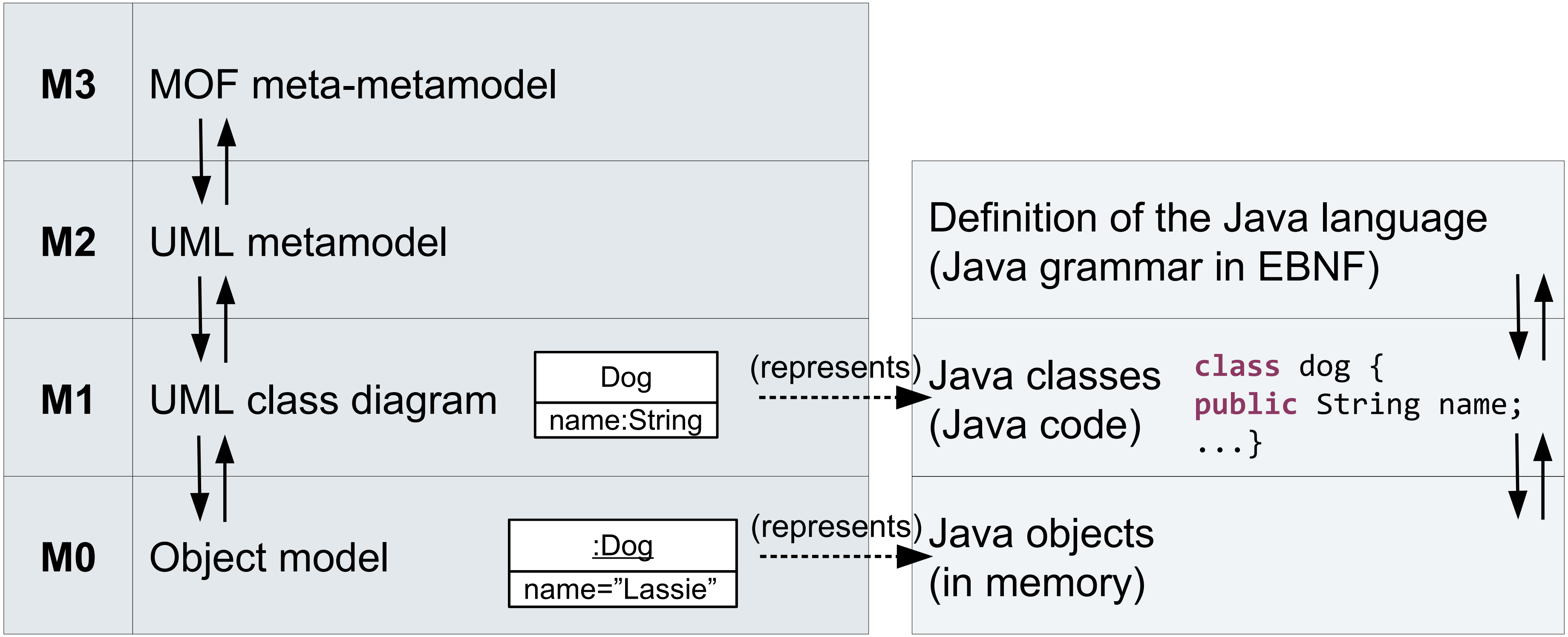
```
class dog {
public String name;
...}
```

M0 Object model

Java objects (in memory)

Meta-Levels done right!

- Classes in the class diagram describe Java classes

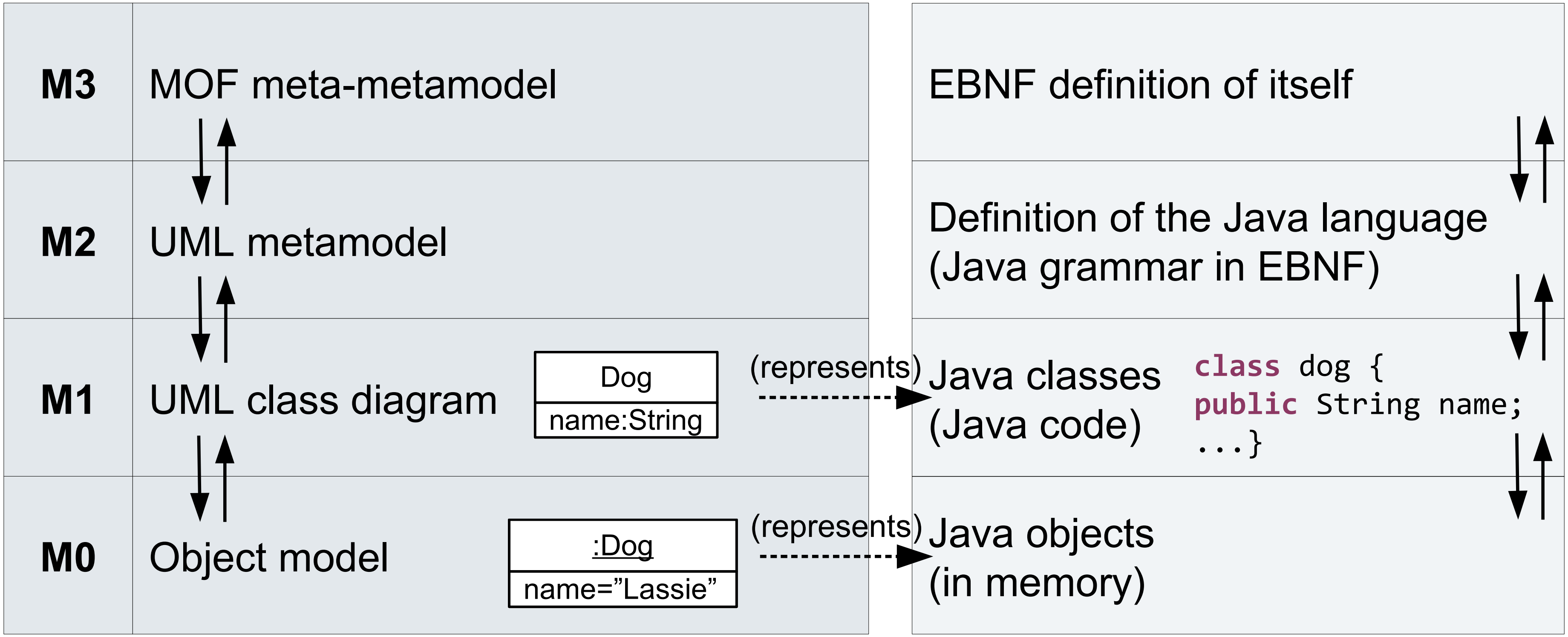


M3	MOF meta-metamodel				
M2	UML metamodel			Definition of the Java language (Java grammar in EBNF)	
M1	UML class diagram	Dog name:String	(represents)	Java classes (Java code)	`class dog { public String name; ...}`
M0	Object model	:Dog name="Lassie"	(represents)	Java objects (in memory)	

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

Meta-Levels done right!

- Classes in the class diagram describe Java classes



M3 MOF meta-metamodel

M2 UML metamodel

M1 UML class diagram

Dog | name:String

(represents)

M0 Object model

:Dog | name=”Lassie”

(represents)

EBNF definition of itself

Definition of the Java language (Java grammar in EBNF)

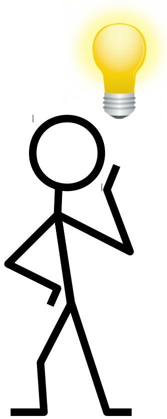
Java classes (Java code)

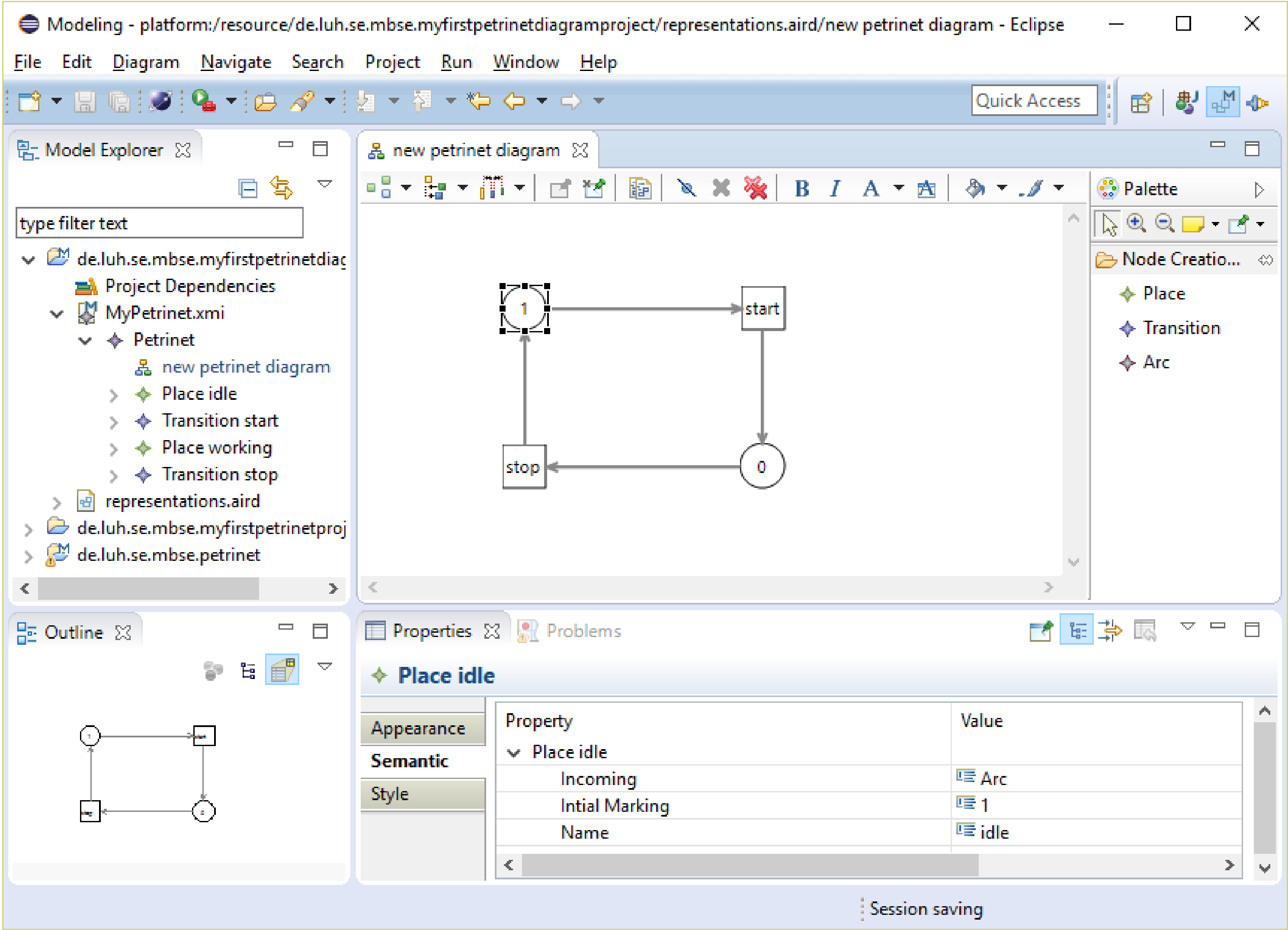
```
class dog {
public String name;
...}
```

Java objects (in memory)

2.4. Metamodeling frameworks

Vision: Build a Petri Net Modeling Tool





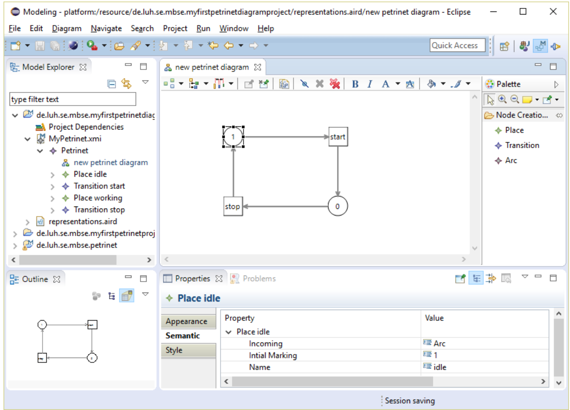
Modeling - platform:/resource/de.luh.se.mbse.myfirstpetrinetdiagramproject/representations.aird/new petrinet diagram - Eclipse

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



Modeling - platform:/resource/de.luh.se.mbse.myfirstpetrinetdiagramproject/representations.aird/new petrinet diagram - Eclipse

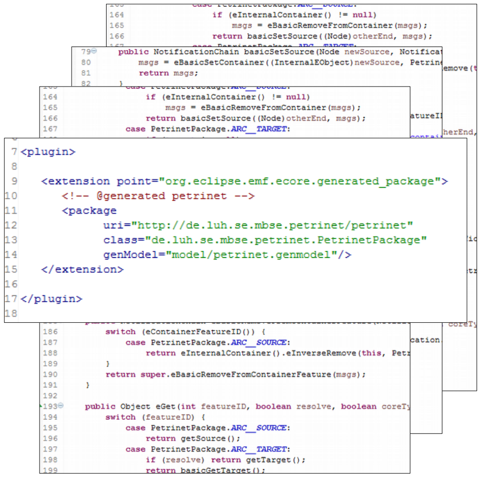
```
de.luh.se.mbse.myfirstpetrinetdiag
  Project Dependencies
  MyPetrinet.xmi
    Petrinet
      new petrinet diagram
      Place idle
      Transition start
      Place working
      Transition stop
  representations.aird
de.luh.se.mbse.myfirstpetrinetproj
de.luh.se.mbse.petrinet
```

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



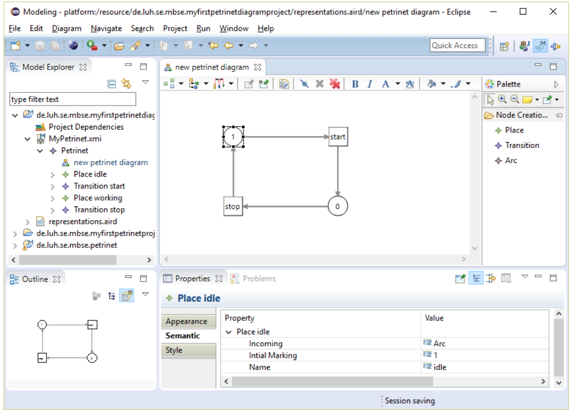
```
            if (eInternalContainer() != null)
                msgs = eBasicRemoveFromContainer(msgs);
            return basicSetSource((Node)otherEnd, msgs);
```

```
    public NotificationChain basicSetSource(Node newSource, Notificat
        msgs = eBasicSetContainer((InternalEObject)newSource, Petrin
        return msgs;
```

```
            if (eInternalContainer() != null)
                msgs = eBasicRemoveFromContainer(msgs);
            return basicSetSource((Node)otherEnd, msgs);
        case PetrinetPackage.ARC__TARGET:
```

```
<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>

</plugin>
```

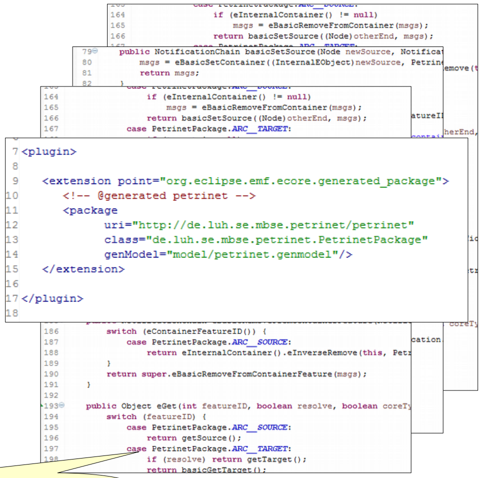
```
        switch (eContainerFeatureID()) {
            case PetrinetPackage.ARC__SOURCE:
                return eInternalContainer().eInverseRemove(this, Petr
        }
        return super.eBasicRemoveFromContainerFeature(msgs);
    }

    public Object eGet(int featureID, boolean resolve, boolean coreTy
        switch (featureID) {
            case PetrinetPackage.ARC__SOURCE:
                return getSource();
            case PetrinetPackage.ARC__TARGET:
                if (resolve) return getTarget();
                return basicGetTarget();
```



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



```
                if (eInternalContainer() != null)
                    msgs = eBasicRemoveFromContainer(msgs);
                return basicSetSource((Node)otherEnd, msgs);
```

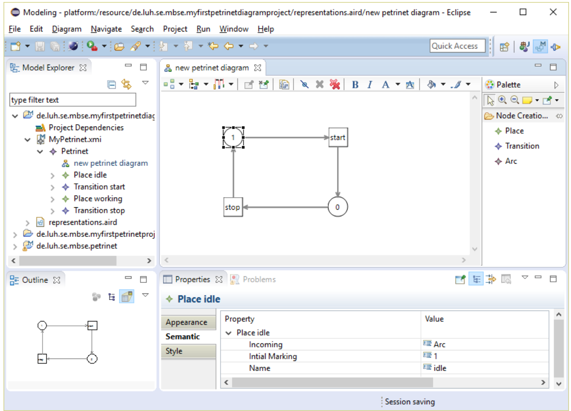
```
    public NotificationChain basicSetSource(Node newSource, Notificat
        msgs = eBasicSetContainer((InternalEObject)newSource, Petrin
        return msgs;
```

```
                if (eInternalContainer() != null)
                    msgs = eBasicRemoveFromContainer(msgs);
                return basicSetSource((Node)otherEnd, msgs);
            case PetrinetPackage.ARC__TARGET:
```

```
<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>

</plugin>
```

```
        switch (eContainerFeatureID()) {
            case PetrinetPackage.ARC__SOURCE:
                return eInternalContainer().eInverseRemove(this, Petr
        }
        return super.eBasicRemoveFromContainerFeature(msgs);
    }

    public Object eGet(int featureID, boolean resolve, boolean coreTy
        switch (featureID) {
            case PetrinetPackage.ARC__SOURCE:
                return getSource();
            case PetrinetPackage.ARC__TARGET:
                if (resolve) return getTarget();
                return basicGetTarget();
```

model code



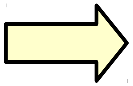
Modeling - platform:/resource/de.luh.se.mbse.myfirstpetrinetdiagramproject/representations.aird/new petrinet diagram - Eclipse

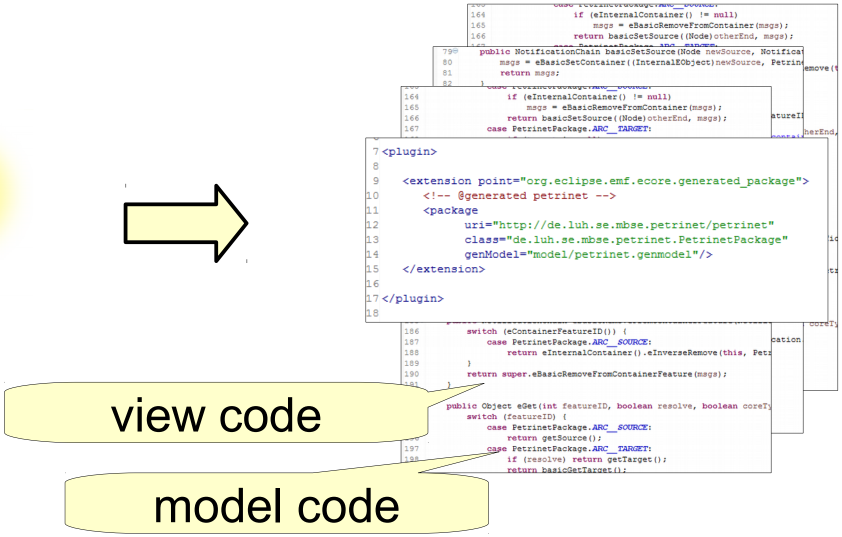
Place idle

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





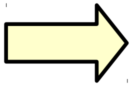
```
            if (eInternalContainer() != null)
                msgs = eBasicRemoveFromContainer(msgs);
            return basicSetSource((Node)otherEnd, msgs);
        case PetrinetPackage.ARC__TARGET:
```

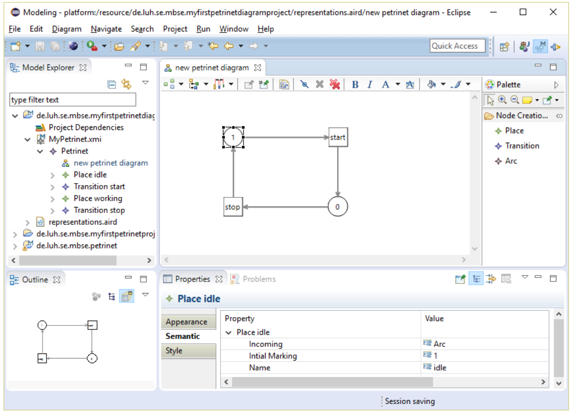
```
<plugin>

   <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>

</plugin>
```

```
      switch (eContainerFeatureID()) {
         case PetrinetPackage.ARC__SOURCE:
            return eInternalContainer().eInverseRemove(this, Petr
      }
      return super.eBasicRemoveFromContainerFeature(msgs);
   }
```

view code

model code



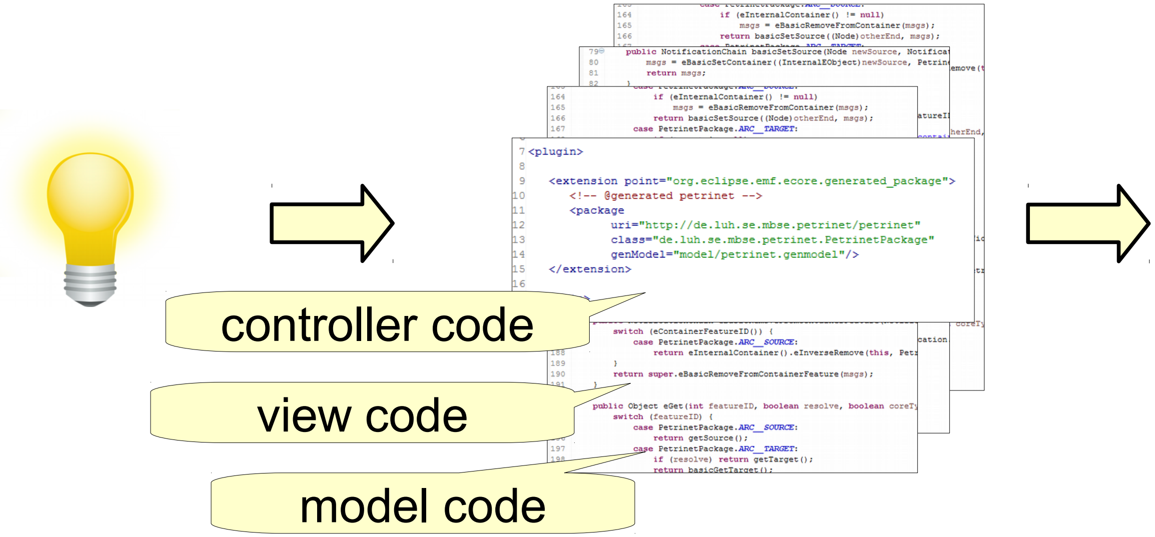


Modeling - platform:/resource/de.luh.se.mbse.myfirstpetrinetdiagramproject/representations.aird/new petrinet diagram - Eclipse

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



```
            if (eInternalContainer() != null)
                msgs = eBasicRemoveFromContainer(msgs);
            return basicSetSource((Node)otherEnd, msgs);
```

```
    public NotificationChain basicSetSource(Node newSource, Notificat
        msgs = eBasicSetContainer((InternalEObject)newSource, Petrin
        return msgs;
```

```
            if (eInternalContainer() != null)
                msgs = eBasicRemoveFromContainer(msgs);
            return basicSetSource((Node)otherEnd, msgs);
        case PetrinetPackage.ARC__TARGET:
```

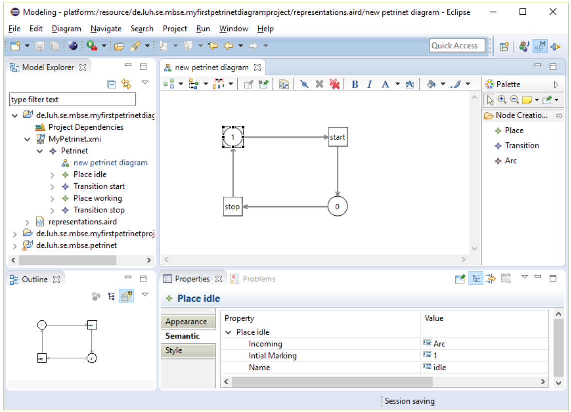
```
<plugin>

   <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>

```

```
        switch (eContainerFeatureID()) {
            case PetrinetPackage.ARC__SOURCE:
                return eInternalContainer().eInverseRemove(this, Petr
        }
        return super.eBasicRemoveFromContainerFeature(msgs);
    }

    public Object eGet(int featureID, boolean resolve, boolean coreTy
        switch (featureID) {
            case PetrinetPackage.ARC__SOURCE:
                return getSource();
            case PetrinetPackage.ARC__TARGET:
                if (resolve) return getTarget();
                return basicGetTarget();
```

controller code

view code

model code



Modeling - platform:/resource/de.luh.se.mbse.myfirstpetrinetdiagramproject/representations.aird/new petrinet diagram - Eclipse

- 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

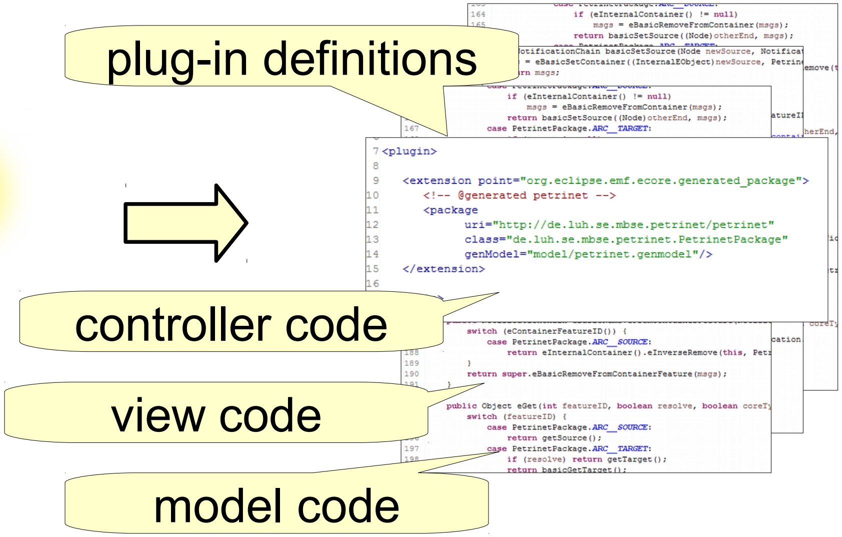
Place idle

Property	Value
Place idle	
Incoming	Arc
Intial Marking	1
Name	idle

Session saving

Build a Petri Net Modeling Tool

- Manual implementation: A lot of repetitive or generic code



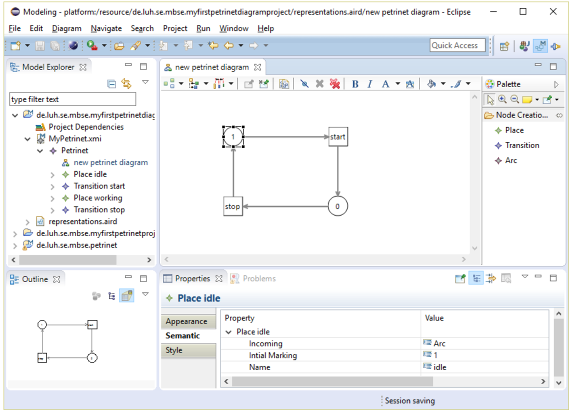
```
            if (eInternalContainer() != null)
                msgs = eBasicRemoveFromContainer(msgs);
            return basicSetSource((Node)otherEnd, msgs);
```

```
NotificationChain basicSetSource(Node newSource, Notificat
  = eBasicSetContainer((InternalEObject)newSource, Petrin
rn msgs;
```

```
            if (eInternalContainer() != null)
                msgs = eBasicRemoveFromContainer(msgs);
            return basicSetSource((Node)otherEnd, msgs);
        case PetrinetPackage.ARC__TARGET:
```

```
<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>
```

```
    switch (eContainerFeatureID()) {
        case PetrinetPackage.ARC__SOURCE:
            return eInternalContainer().eInverseRemove(this, Petr
    }
    return super.eBasicRemoveFromContainerFeature(msgs);
}

public Object eGet(int featureID, boolean resolve, boolean coreTy
    switch (featureID) {
        case PetrinetPackage.ARC__SOURCE:
            return getSource();
        case PetrinetPackage.ARC__TARGET:
            if (resolve) return getTarget();
            return basicGetTarget();
```



Modeling - platform:/resource/de.luh.se.mbse.myfirstpetrinetdiagramproject/representations.aird/new petrinet diagram - Eclipse

type filter text

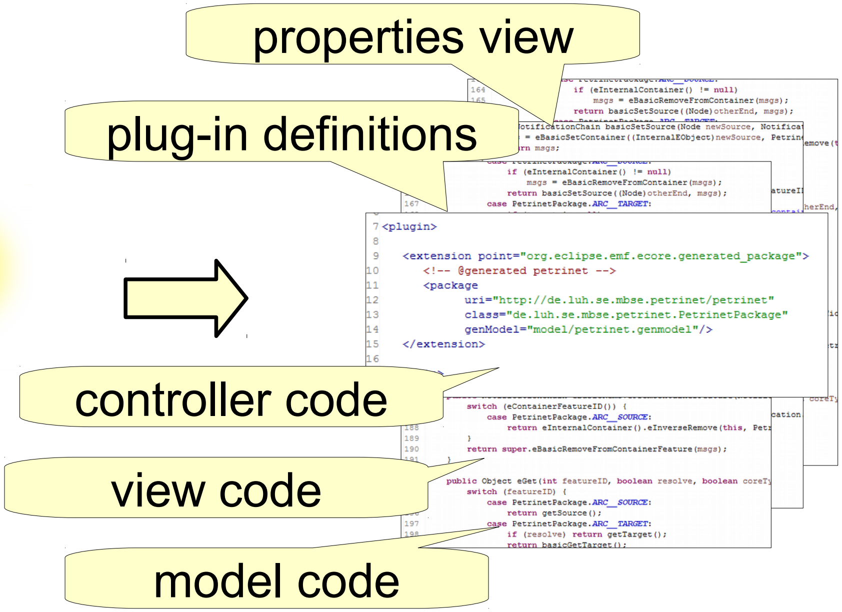
- de.luh.se.mbse.myfirstpetrinetdiag
  - Project Dependencies
  - MyPetrinet.xmi
    - Petrinet
      - new petrinet diagram
      - Place idle
      - Transition start
      - Place working
      - Transition stop
  - representations.aird
- de.luh.se.mbse.myfirstpetrinetproj
- de.luh.se.mbse.petrinet

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



properties view

plug-in definitions

controller code

view code

model code

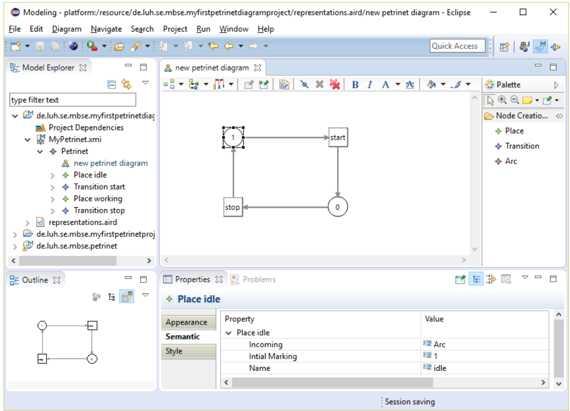
```
                if (eInternalContainer() != null)
                    msgs = eBasicRemoveFromContainer(msgs);
                return basicSetSource((Node)otherEnd, msgs);
```

```
NotificationChain basicSetSource(Node newSource, Notificat
= eBasicSetContainer((InternalEObject)newSource, Petrin
rn msgs;
```

```
                if (eInternalContainer() != null)
                    msgs = eBasicRemoveFromContainer(msgs);
                return basicSetSource((Node)otherEnd, msgs);
            case PetrinetPackage.ARC__TARGET:
```

```
<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>

```

```
        switch (eContainerFeatureID()) {
            case PetrinetPackage.ARC__SOURCE:
                return eInternalContainer().eInverseRemove(this, Petr
        }
        return super.eBasicRemoveFromContainerFeature(msgs);
    }

    public Object eGet(int featureID, boolean resolve, boolean coreTy
        switch (featureID) {
            case PetrinetPackage.ARC__SOURCE:
                return getSource();
            case PetrinetPackage.ARC__TARGET:
                if (resolve) return getTarget();
                return basicGetTarget();
```



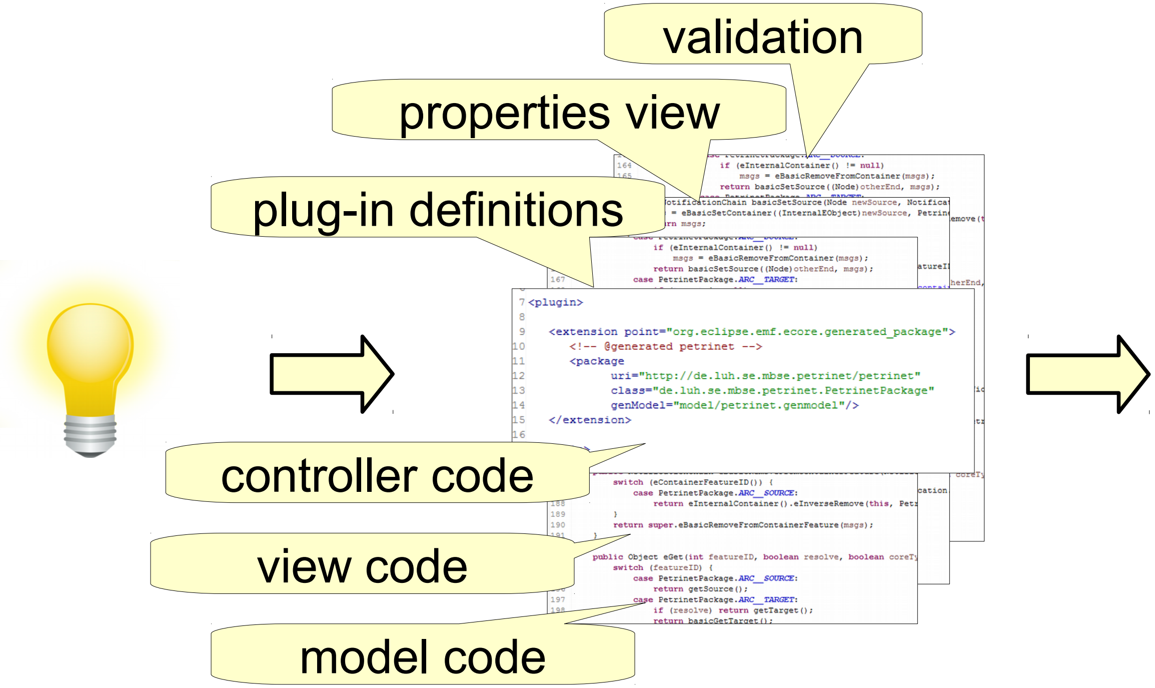
Modeling - platform:/resource/de.luh.se.mbse.myfirstpetrinetdiagramproject/representations.aird/new petrinet diagram - Eclipse

Place idle

Property	Value
Place idle	
Incoming	Arc
Intial Marking	1
Name	idle

Build a Petri Net Modeling Tool

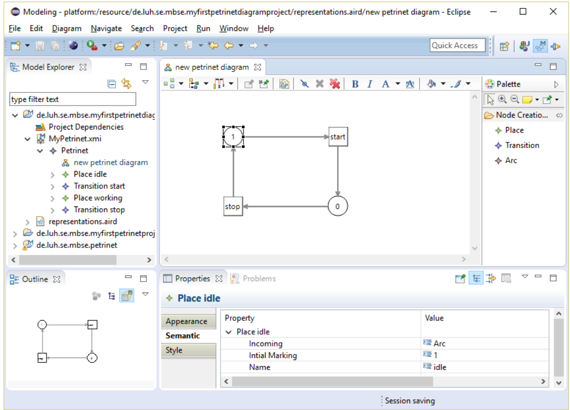
- Manual implementation: A lot of repetitive or generic code



```
<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>

```

```
            if (eInternalContainer() != null)
                msgs = eBasicRemoveFromContainer(msgs);
            return basicSetSource((Node)otherEnd, msgs);
        case PetrinetPackage.ARC__TARGET:
```

```
        switch (eContainerFeatureID()) {
            case PetrinetPackage.ARC__SOURCE:
                return eInternalContainer().eInverseRemove(this, Petr
        }
        return super.eBasicRemoveFromContainerFeature(msgs);
    }

    public Object eGet(int featureID, boolean resolve, boolean coreTy
        switch (featureID) {
            case PetrinetPackage.ARC__SOURCE:
                return getSource();
            case PetrinetPackage.ARC__TARGET:
                if (resolve) return getTarget();
                return basicGetTarget();
```

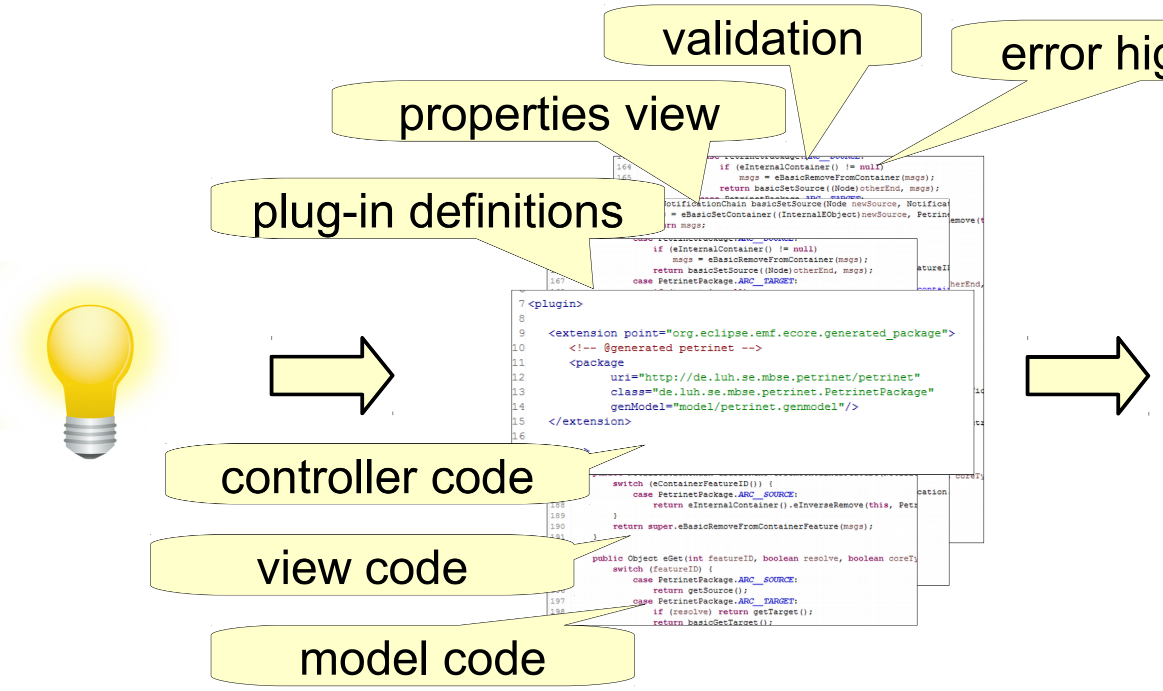

Modeling - platform:/resource/de.luh.se.mbse.myfirstpetrinetdiagramproject/representations.aird/new petrinet diagram - Eclipse

Place idle

Property	Value
Place idle	
Incoming	Arc
Intial Marking	1
Name	idle

Build a Petri Net Modeling Tool

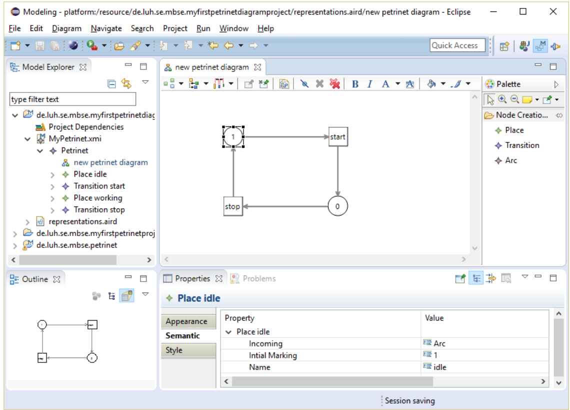
- Manual implementation: A lot of repetitive or generic code



```
<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>
```

```
            if (eInternalContainer() != null)
                msgs = eBasicRemoveFromContainer(msgs);
            return basicSetSource((Node)otherEnd, msgs);
        case PetrinetPackage.ARC__TARGET:
```

```
        switch (eContainerFeatureID()) {
            case PetrinetPackage.ARC__SOURCE:
                return eInternalContainer().eInverseRemove(this, Petr
        }
        return super.eBasicRemoveFromContainerFeature(msgs);
    }

    public Object eGet(int featureID, boolean resolve, boolean coreTy
        switch (featureID) {
            case PetrinetPackage.ARC__SOURCE:
                return getSource();
            case PetrinetPackage.ARC__TARGET:
                if (resolve) return getTarget();
                return basicGetTarget();
```

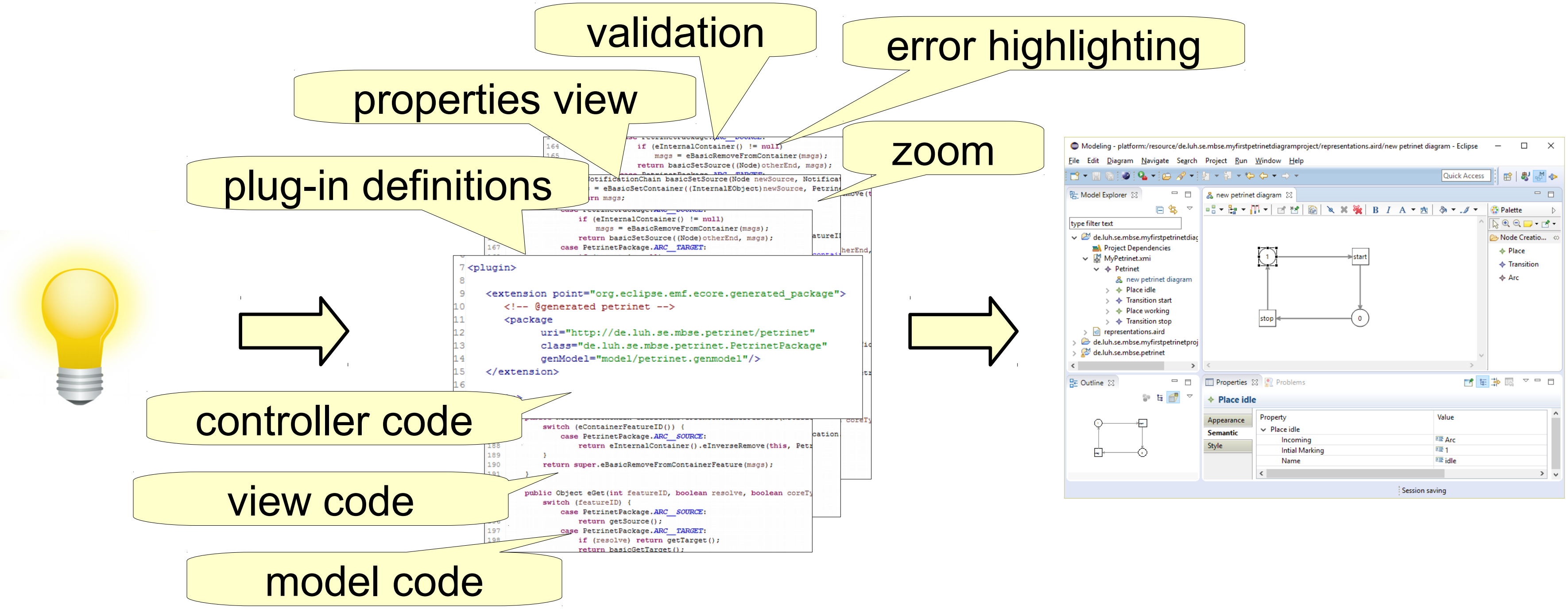


Place idle

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



validation

error highlighting

properties view

zoom

plug-in definitions

controller code

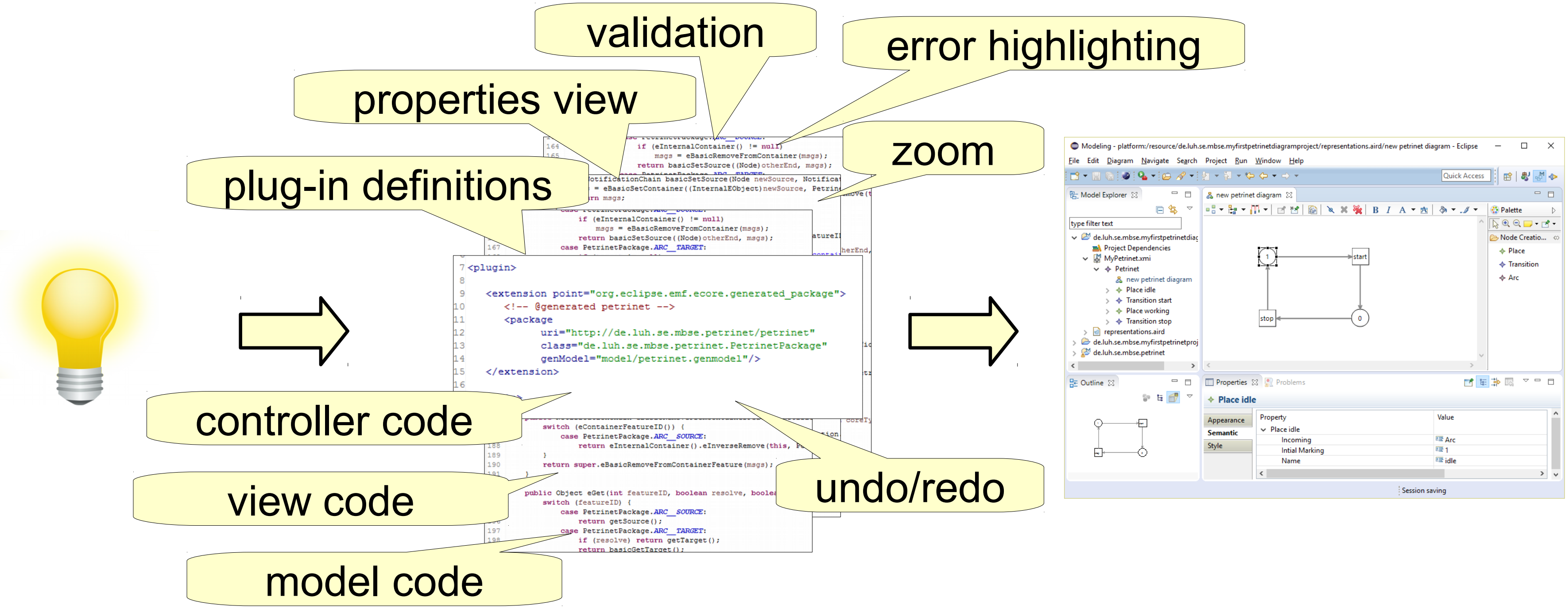
view code

model 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>
```

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>
```

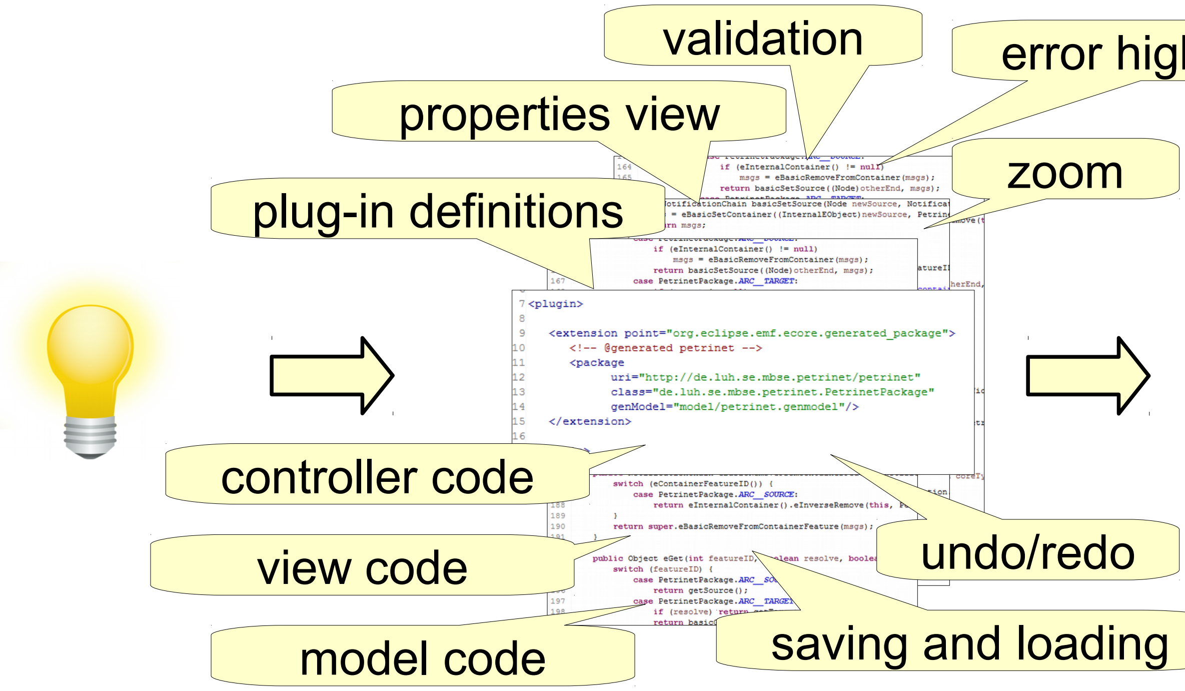
```
         if (eInternalContainer() != null)
            msgs = eBasicRemoveFromContainer(msgs);
         return basicSetSource((Node)otherEnd, msgs);
   NotificationChain basicSetSource(Node newSource, Notificat
      msgs = eBasicSetContainer((InternalEObject)newSource, Petrin
      return msgs;
```

```
         if (eInternalContainer() != null)
            msgs = eBasicRemoveFromContainer(msgs);
         return basicSetSource((Node)otherEnd, msgs);
      case PetrinetPackage.ARC__TARGET:
```

```
      switch (eContainerFeatureID()) {
         case PetrinetPackage.ARC__SOURCE:
            return eInternalContainer().eInverseRemove(this,
      }
      return super.eBasicRemoveFromContainerFeature(msgs);
   }

   public Object eGet(int featureID, boolean resolve, boolea
      switch (featureID) {
         case PetrinetPackage.ARC__SOURCE:
            return getSource();
         case PetrinetPackage.ARC__TARGET:
            if (resolve) return getTarget();
            return basicGetTarget();
```

Build a Petri Net Modeling Tool

- Manual implementation: A lot of repetitive or generic code



validation

error highlighting

properties view

zoom

plug-in definitions

controller code

undo/redo

view code

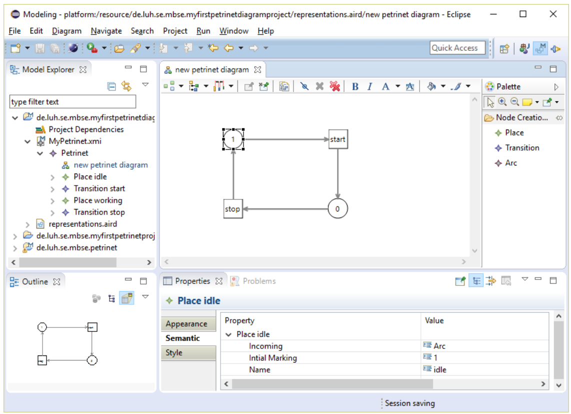
saving and loading

model 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>
```

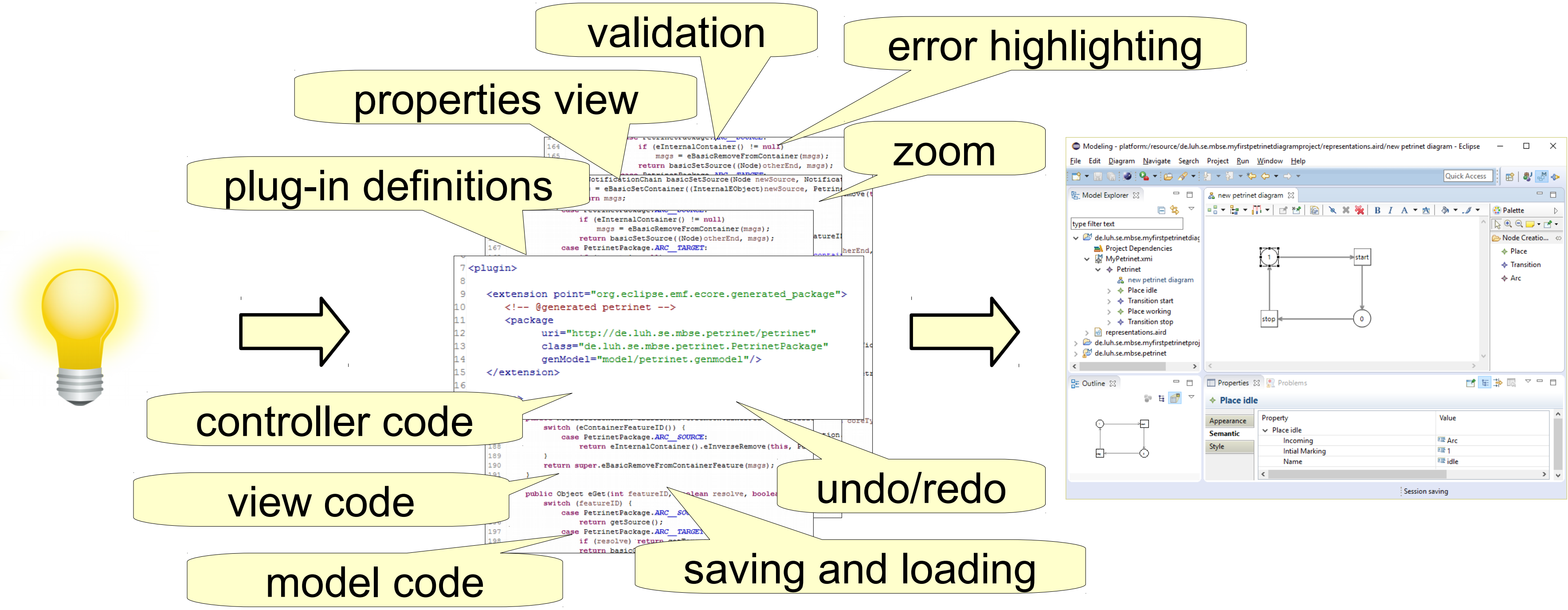
```
            if (eInternalContainer() != null)
                msgs = eBasicRemoveFromContainer(msgs);
            return basicSetSource((Node)otherEnd, msgs);
        case PetrinetPackage.ARC__TARGET:
```

```
        switch (eContainerFeatureID()) {
            case PetrinetPackage.ARC__SOURCE:
        }
        return super.eBasicRemoveFromContainerFeature(msgs);
    }
```



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>
```

```
            if (eInternalContainer() != null)
               msgs = eBasicRemoveFromContainer(msgs);
            return basicSetSource((Node)otherEnd, msgs);
```

```
   NotificationChain basicSetSource(Node newSource, Notificat
      = eBasicSetContainer((InternalEObject)newSource, Petrin
      rn msgs;
```

```
            if (eInternalContainer() != null)
               msgs = eBasicRemoveFromContainer(msgs);
            return basicSetSource((Node)otherEnd, msgs);
         case PetrinetPackage.ARC__TARGET:
```

```
      switch (eContainerFeatureID()) {
         case PetrinetPackage.ARC__SOURCE:
            return eInternalContainer().eInverseRemove(this, Pe
      }
      return super.eBasicRemoveFromContainerFeature(msgs);
   }

   public Object eGet(int featureID, boolean resolve, boolea
      switch (featureID) {
         case PetrinetPackage.ARC__SO
            return getSource();
         case PetrinetPackage.ARC__TARGET
            if (resolve) return
            return basic
```

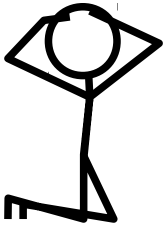
Modeling - platform:/resource/de.luh.se.mbse.myfirstpetrinetdiagramproject/representations.aird/new petrinet diagram - Eclipse

- 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

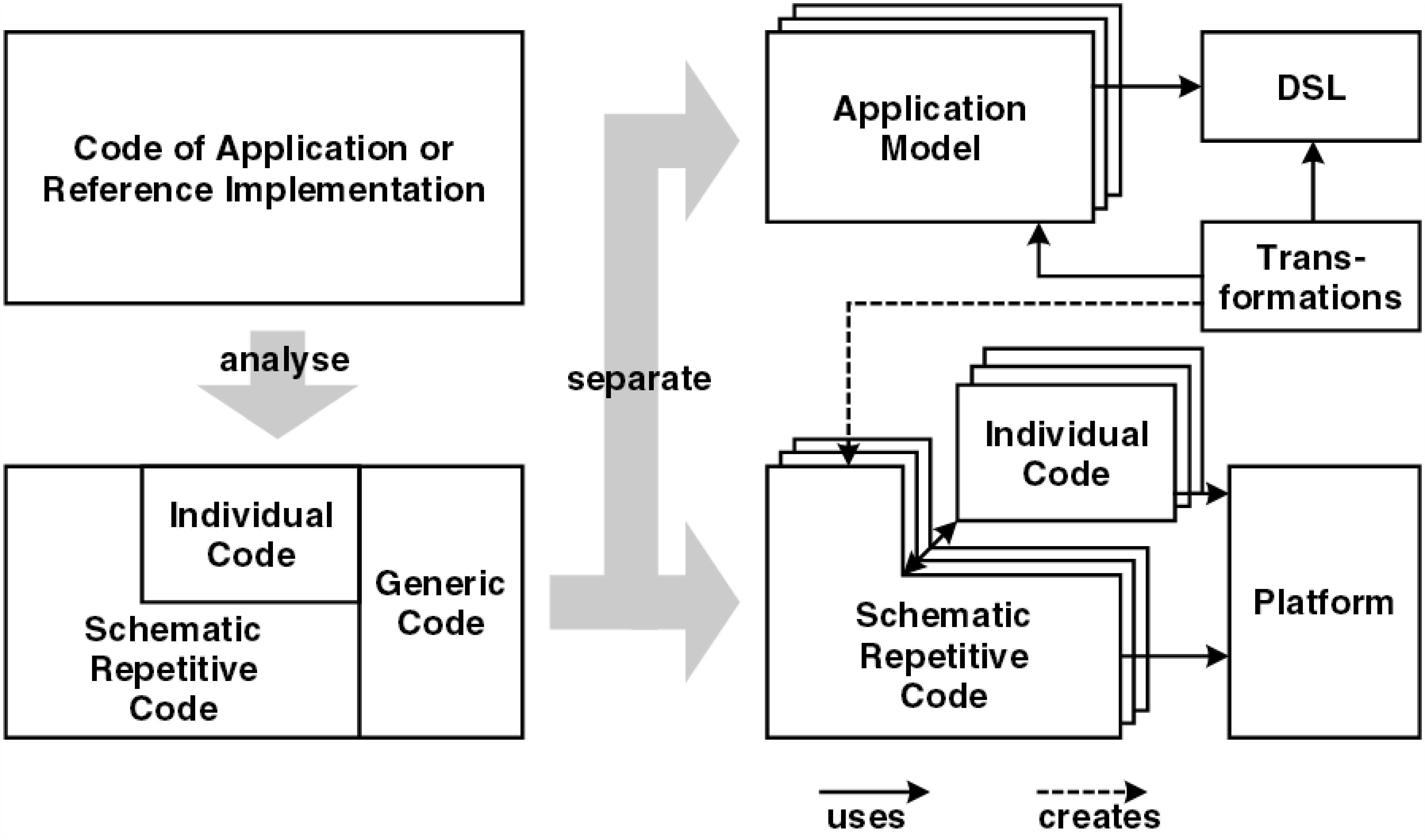
Property	Value
Place idle	
Incoming	Arc
Intial Marking	1
Name	idle

Session saving



The idea of MBSE/MDSD in a bit more detail

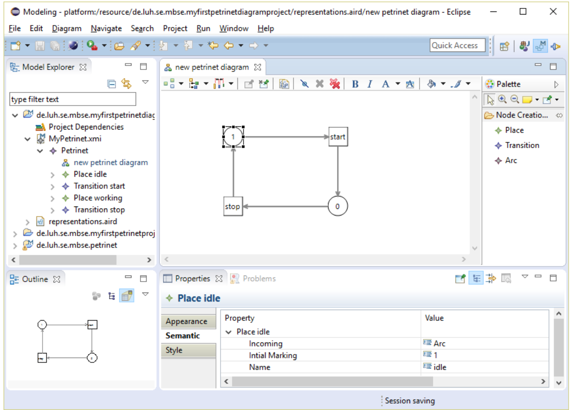
in the last lecture...



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

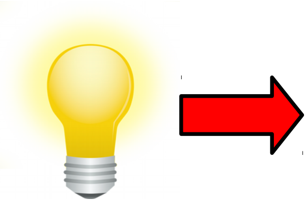


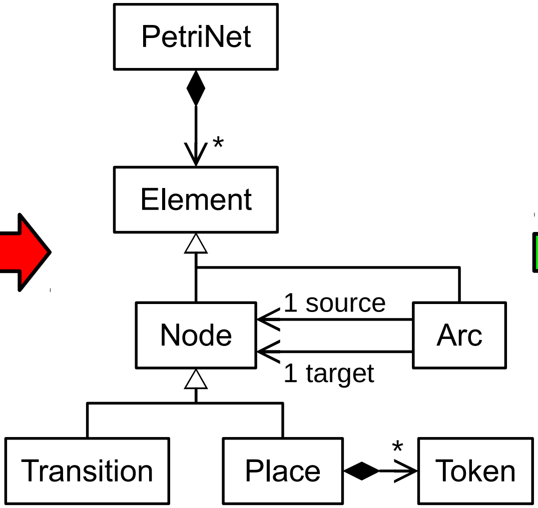
Place idle

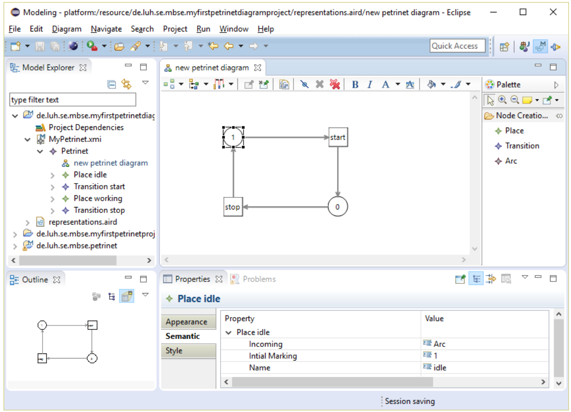
Property	Value
Place idle	
Incoming	Arc
Intial Marking	1
Name	idle

Build a Petri Net Modeling Tool

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







Modeling - platform:/resource/de.luh.se.mbse.myfirstpetrinetdiagramproject/representations.aird/new petrinet diagram - Eclipse

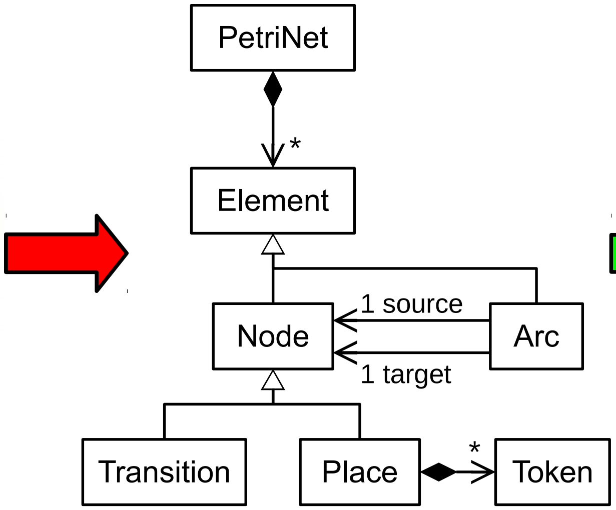
- de.luh.se.mbse.myfirstpetrinetdiag
  - Project Dependencies
  - MyPetrinet.xmi
    - Petrinet
      - new petrinet diagram
      - Place idle
      - Transition start
      - Place working
      - Transition stop
  - representations.aird
- de.luh.se.mbse.myfirstpetrinetproj
- de.luh.se.mbse.petrinet

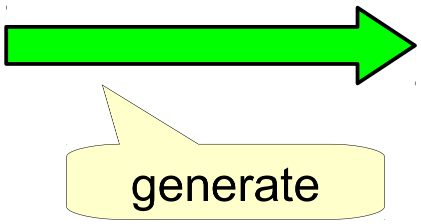
Place idle

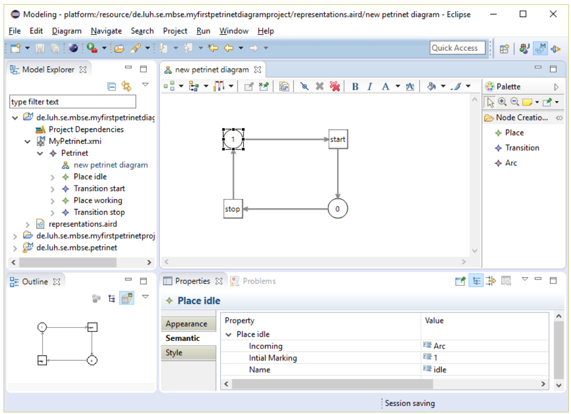
Property	Value
Place idle	
Incoming	Arc
Intial Marking	1
Name	idle

Build a Petri Net Modeling Tool

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






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

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

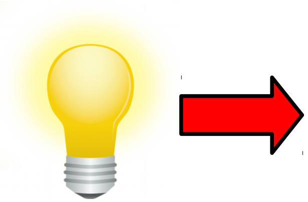
Place idle

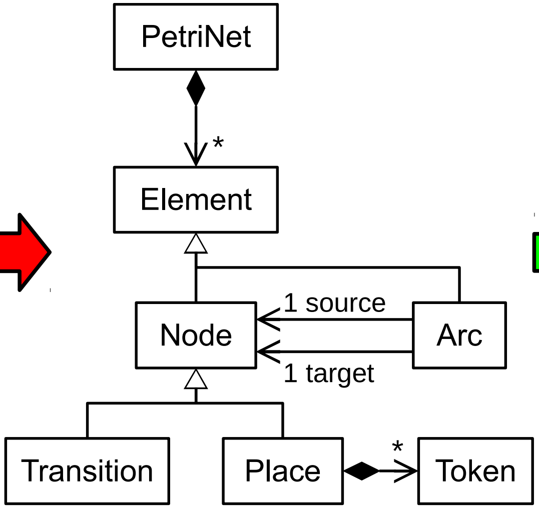
Property	Value
Place idle	
Incoming	Arc
Intial Marking	1
Name	idle

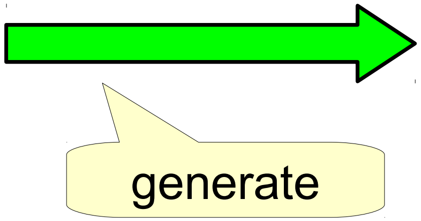
Session saving

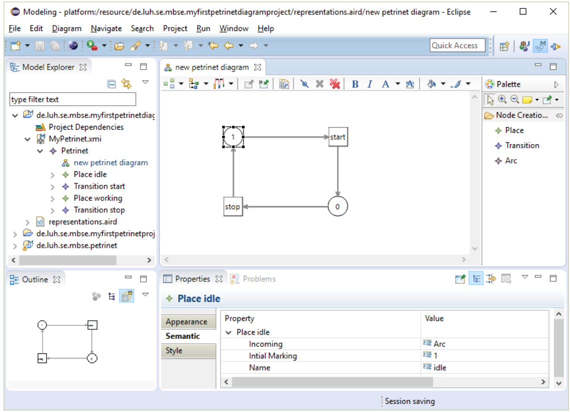
Build a Petri Net Modeling Tool

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

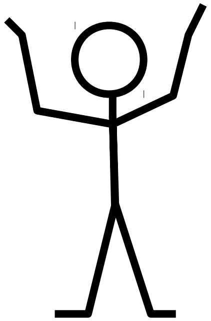







Modeling - platform:/resource/de.luh.se.mbse.myfirstpetrinetdiagramproject/representations.aird/new petrinet diagram - Eclipse

Property	Value
Place idle	
Incoming	Arc
Intial Marking	1
Name	idle



Eclipse Modeling Framework

- The Eclipse Modeling Framework (EMF) is a metamodeling framework for Eclipse

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

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

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”

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 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

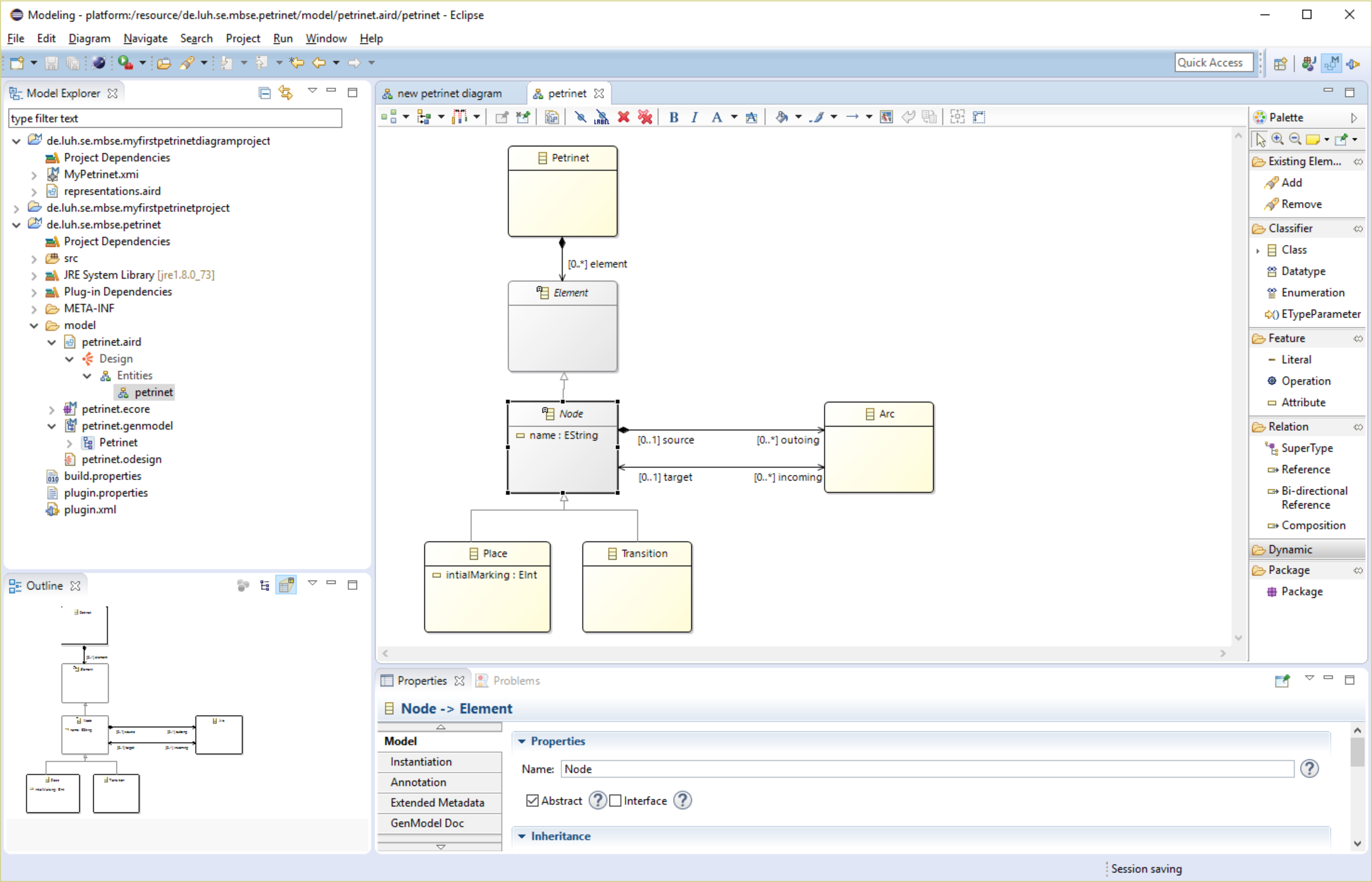
- EMF provides the Ecore meta-metamodel
  - reference implementation of OMG's EMOF, “Essential MOF”

- Many other frameworks build on EMF
  - Eclipse UML2 (reference implementation of OMG's UML2)

Eclipse Modeling Framework

- 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)



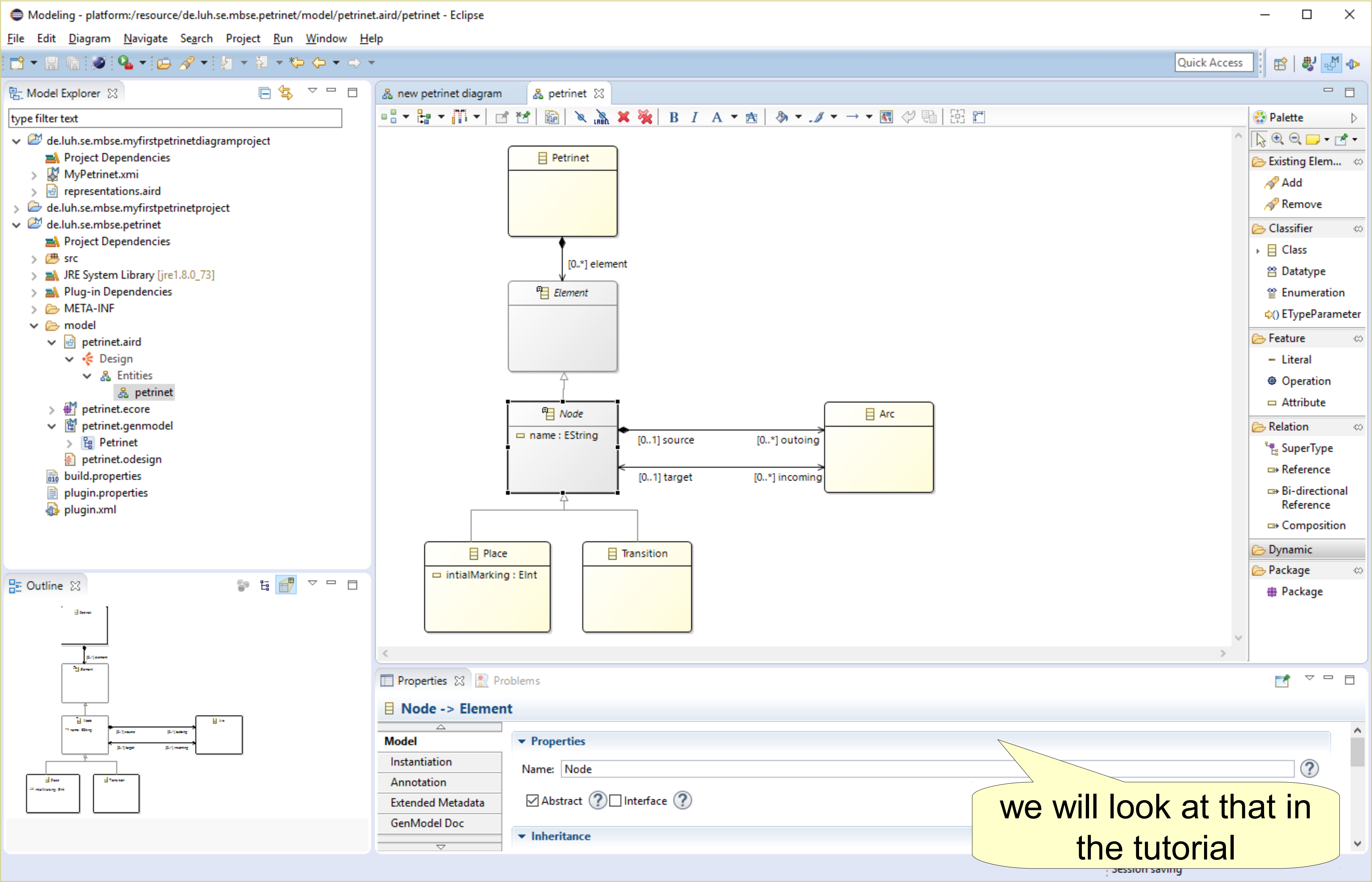
Modeling - platform:/resource/de.luh.se.mbse.petrinet/model/petrinet.aird/petrinet - Eclipse

Node -> Element

Name: Node

☑ Abstract ☐ Interface

Eclipse Modeling Framework (Modeling a Petri Net Metamodel)



Modeling - platform:/resource/de.luh.se.mbse.petrinet/model/petrinet.aird/petrinet - Eclipse

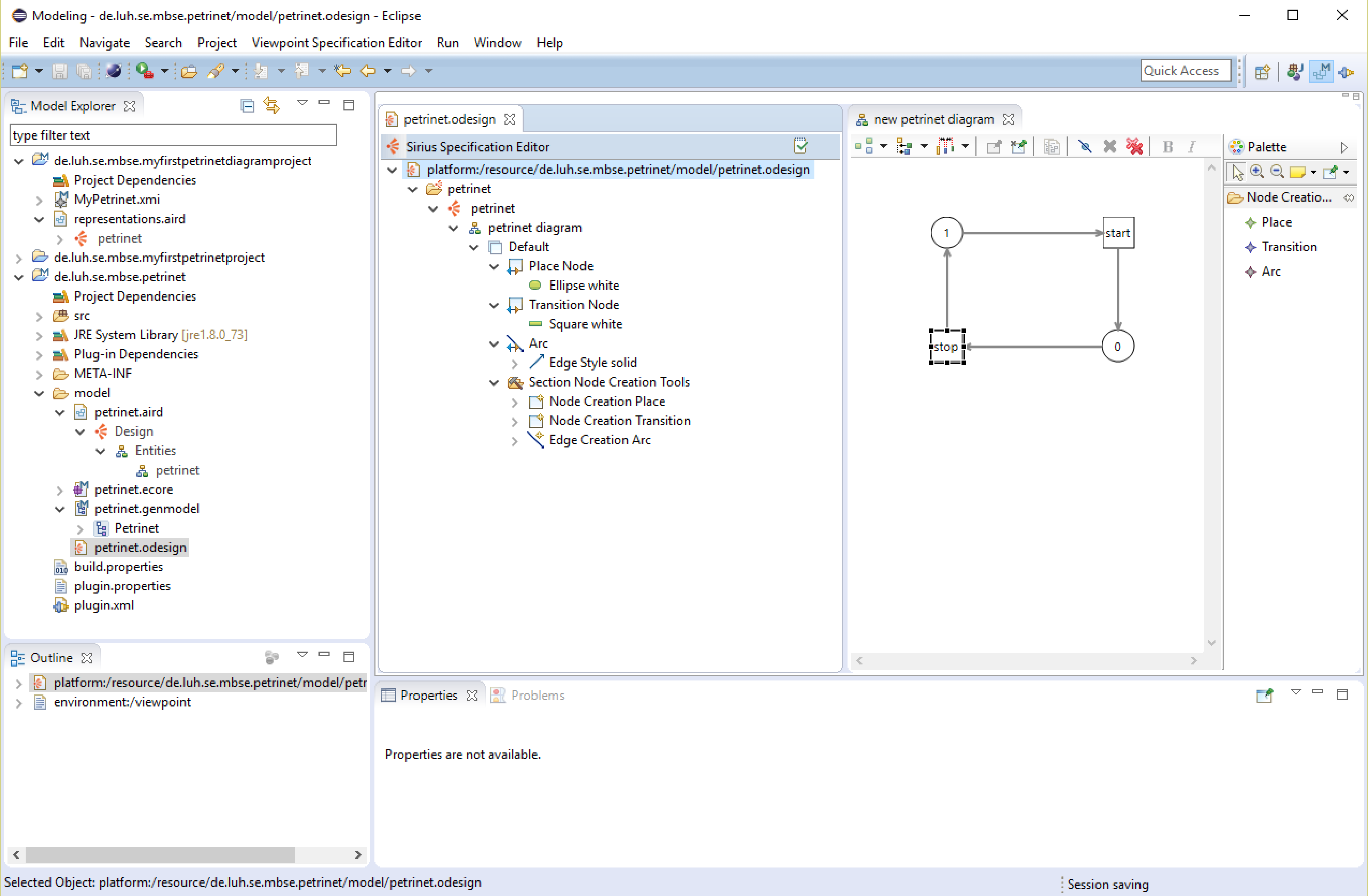
Node -> Element

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

Quick Access

Model Explorer

type filter text

- 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

Palette

Node Creatio...

- Place
- Transition
- Arc

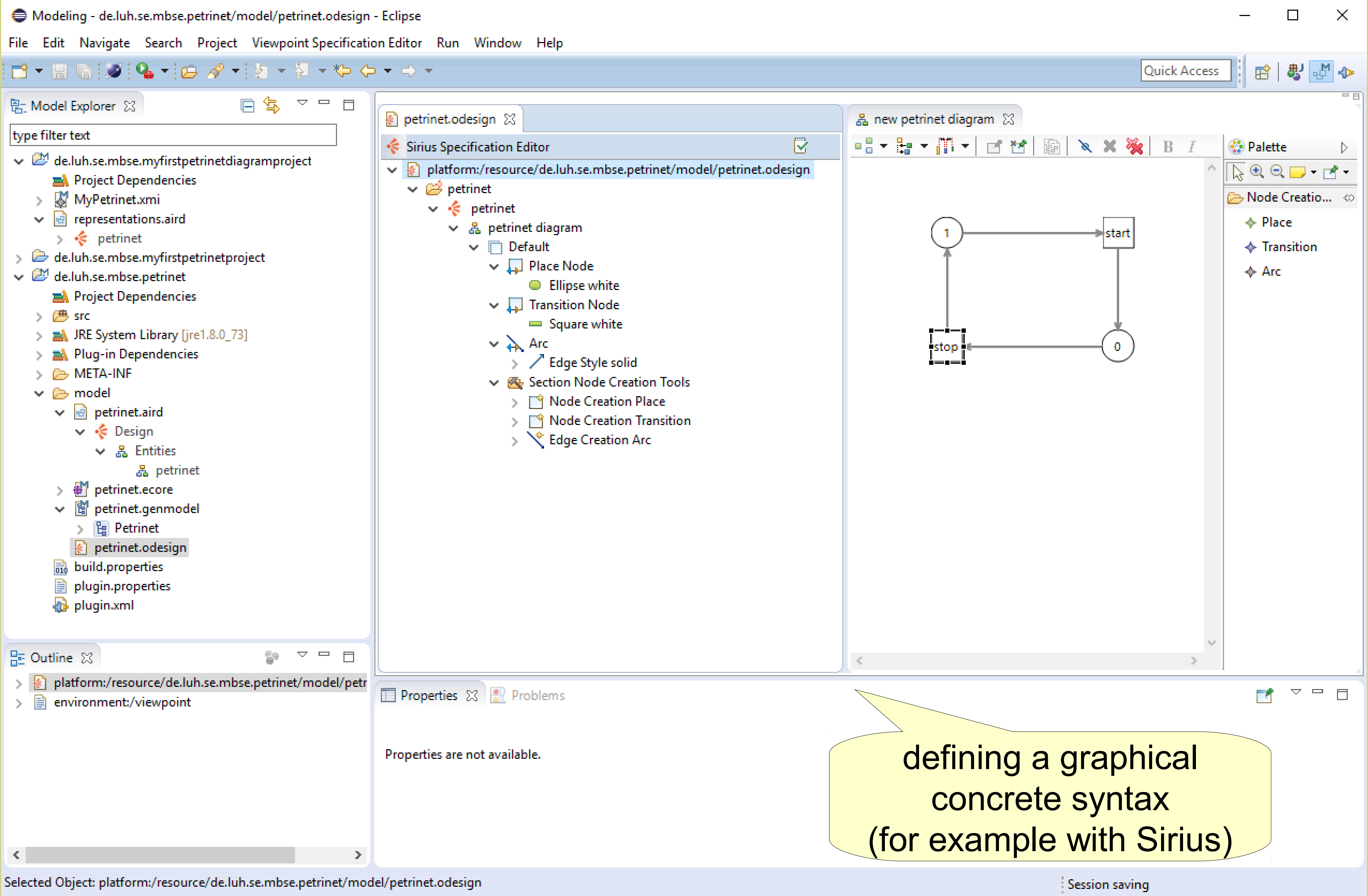
Properties Problems

Properties are not available.

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

Session saving

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
Quick Access
Model Explorer
type filter text

- 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

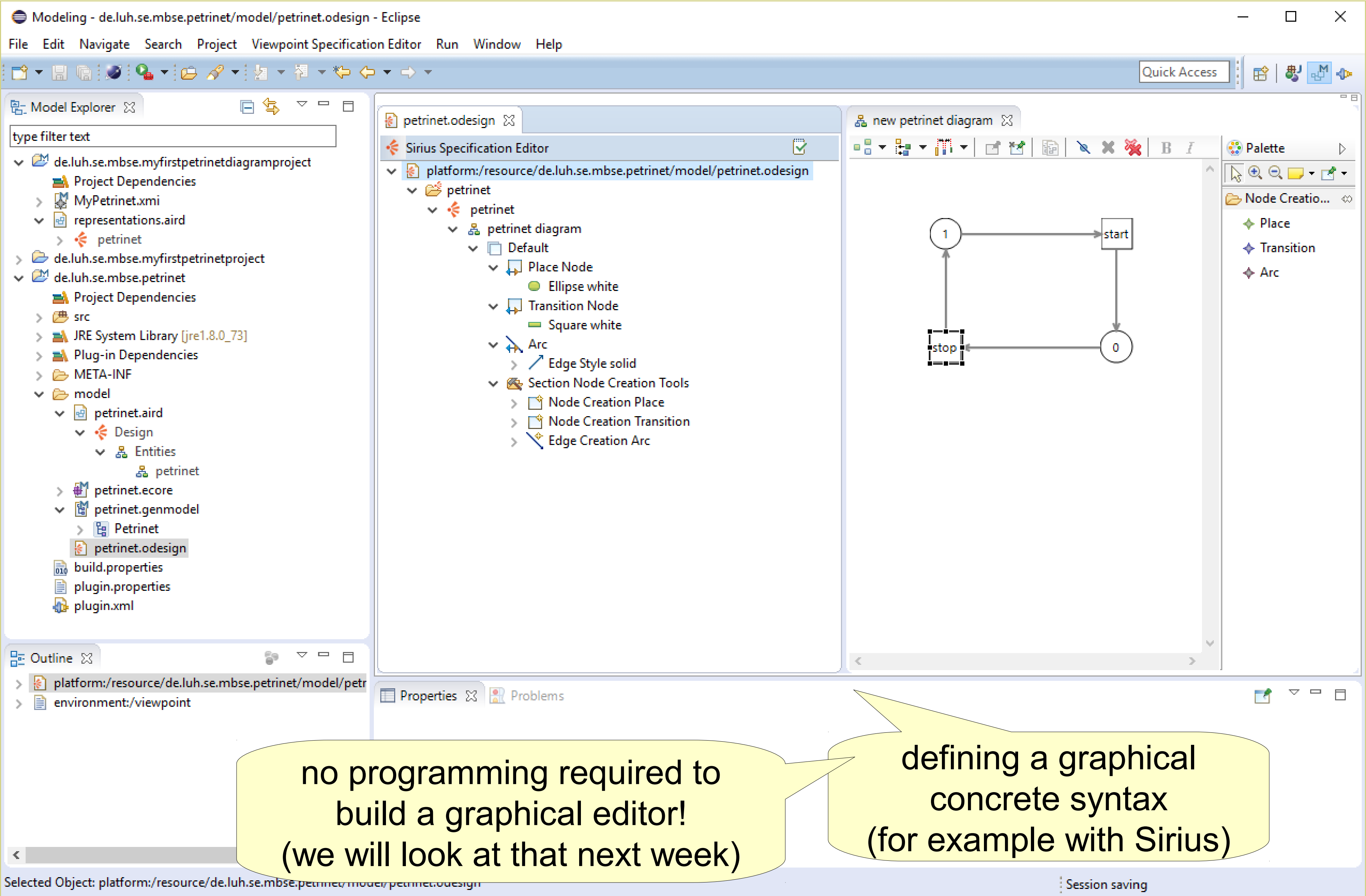
Properties are not available.

defining a graphical concrete syntax (for example with Sirius)

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

Session saving

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
Quick Access
Model Explorer
type filter text

- 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

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

- 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

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)