Metamodeling and Domain-specific modeling

Horváth Ákos
Bergmann Gábor
Dániel Varró
István Ráth

MOTIVÁCIÓ

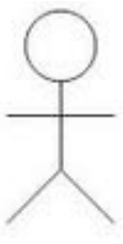
Szakterület-specifikus modellezési nyelvek

Szoftver-
fejlesztő

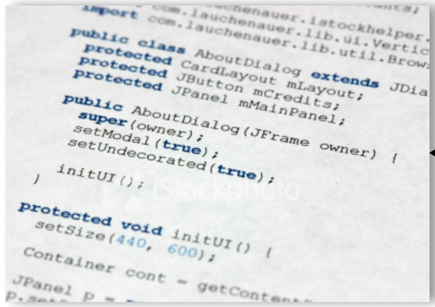
```
import com.lauchenauer.istockhelper.
       com.lauchenauer.lib.ui.Vertic
       com.lauchenauer.lib.util.Brow
public class AboutDialog extends JDia
  protected CardLayout mLayout;
  protected JButton mCredits;
  protected JPanel mMainPanel;

  public AboutDialog(JFrame owner) {
    super(owner);
    setModal(true);
    setUndecorated(true);

    initUI();
  }

  protected void initUI() {
    setSize(440, 600);

    Container cont = getContent
    JPanel p =
```

Programozási nyelv

Szakterület-specifikus modellezési nyelvek

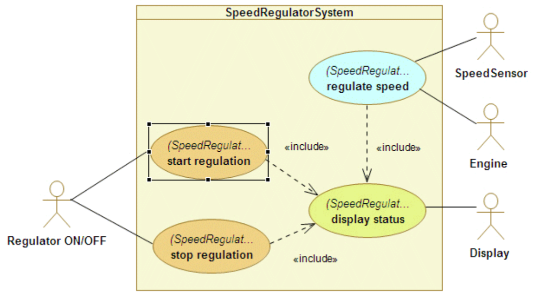


Szoftver-
fejlesztő



```
import  com.lauchenauer.istockhelper.
        com.lauchenauer.lib.ui.Vertic
        com.lauchenauer.lib.util.Brow

public class AboutDialog extends JDia
  protected CardLayout mLayout;
  protected JButton mCredits;
  protected JPanel mMainPanel;

  public AboutDialog(JFrame owner) {
    super(owner);
    setModal(true);
    setUndecorated(true);

    initUI();
  }

  protected void initUI() {
    setSize(440, 600);

    Container cont = getContent
    JPanel p =
```

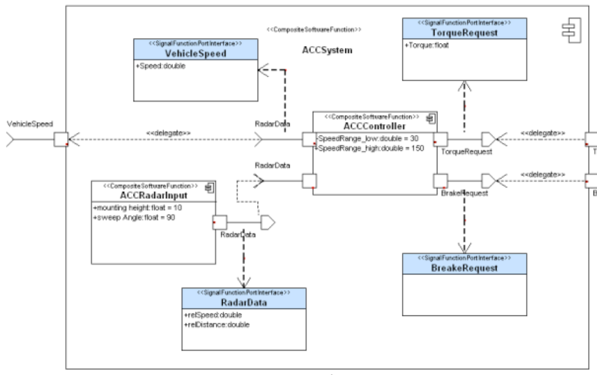
Programozási nyelv



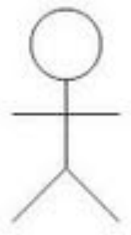
Szoftvermodell

Szoftver-
tervező

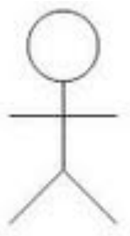
Szakterület-specifikus modellezési nyelvek



<<CompositeSoftwareFunction>>
ACCSystem
<<SignalFunctionPortInterface>>
VehicleSpeed
+Speed:double
<<SignalFunctionPortInterface>>
TorqueRequest
+Torque:float
VehicleSpeed
<<delegate>>
RadarData
<<CompositeSoftwareFunction>>
ACCController
-SpeedRange_low:double = 30
+SpeedRange_high:double = 150
TorqueRequest
<<delegate>>
TorqueRequest
RadarData
<<delegate>>
<<CompositeSoftwareFunction>>
ACCRadarInput
+mounting height:float = 10
+sweep Angle:float = 90
RadarData
BrakeRequest
BrakeRequest
<<SignalFunctionPortInterface>>
RadarData
+relSpeed:double
+relDistance:double
<<SignalFunctionPortInterface>>
BreakeRequest



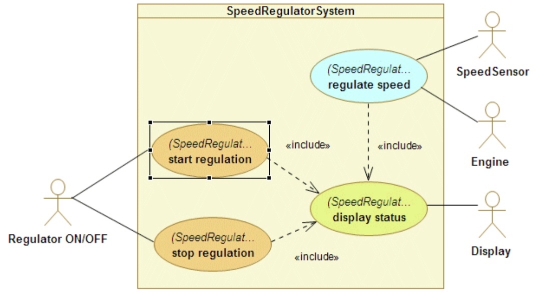
Rendszer-
tervező



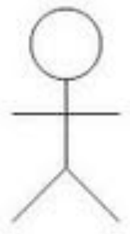
Szoftver-
fejlesztő

```
import com.lauchenauer.iatockhelper.
   com.lauchenauer.lib.ui.Vertic
   com.lauchenauer.lib.util.Brow
public class AboutDialog extends JDia
  protected CardLayout mLayout;
  protected JButton mCredits;
  protected JPanel mMainPanel;
  public AboutDialog(JFrame owner) {
    super(owner);
    setModal(true);
    setUndecorated(true);
    initUI();
  }
  protected void initUI() {
    setSize(440, 600);
    Container cont = getContent
    JPanel p =
```

Programozási nyelv



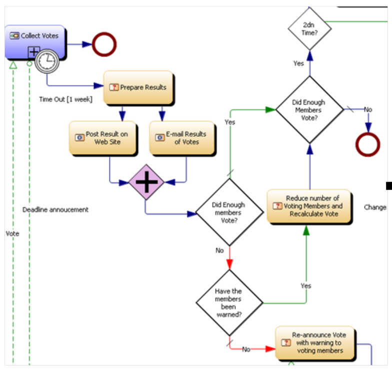
Szoftvermodell



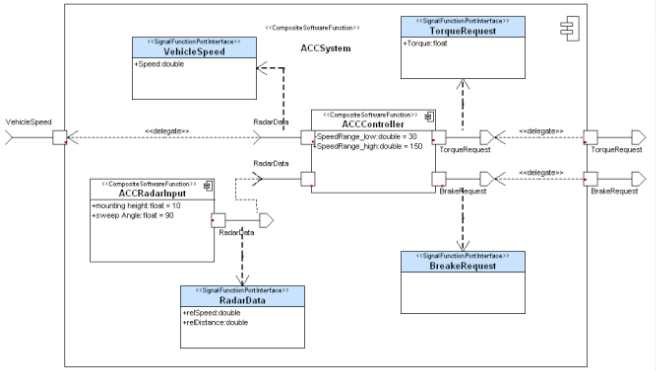
Szoftver-
tervező

Szakterület-specifikus modellezési nyelvek

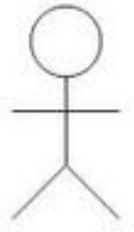
Üzleti elemző



Üzleti folyamat

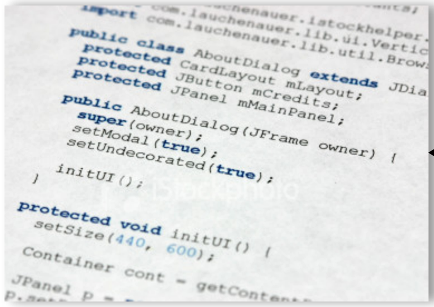


<<CompositeSoftwareFunction>>
ACCSystem
<<SignalFunctionPortInterface>>
VehicleSpeed
+Speed:double
<<SignalFunctionPortInterface>>
TorqueRequest
+Torque:float
VehicleSpeed
<<delegate>>
RadarData
<<CompositeSoftwareFunction>>
ACCController
-SpeedRange_low:double = 30
-SpeedRange_high:double = 150
TorqueRequest
<<delegate>>
TorqueRequest
RadarData
<<CompositeSoftwareFunction>>
ACCRadarInput
+mounting height:float = 10
+sweep Angle:float = 90
BrakeRequest
<<delegate>>
BrakeRequest
RadarData
<<SignalFunctionPortInterface>>
RadarData
+relSpeed:double
+relDistance:double
<<SignalFunctionPortInterface>>
BreakeRequest



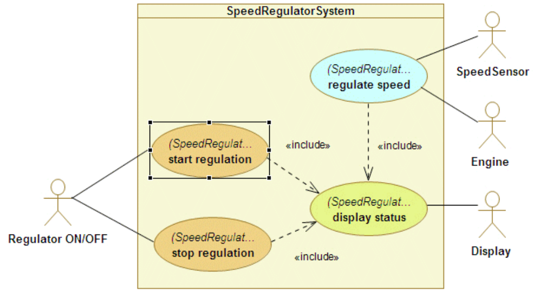
Rendszer-tervező

Szoftver-fejlesztő



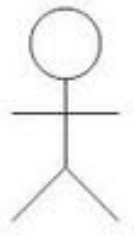
```
import com.lauchenauer.
import com.lauchenauer.lib.ui.Vertic
       com.lauchenauer.lib.util.Brow
public class AboutDialog extends JDia
  protected CardLayout mLayout;
  protected JButton mCredits;
  protected JPanel mMainPanel;

  public AboutDialog(JFrame owner) {
    super(owner);
    setModal(true);
    setUndecorated(true);
    initUI();
  }

  protected void initUI() {
    setSize(440, 600);
    Container cont = getConten
    JPanel p =
```

Programozási nyelv



SpeedRegulatorSystem
(SpeedRegulat... regulate speed
SpeedSensor
Engine
(SpeedRegulat... start regulation
«include»
«include»
Regulator ON/OFF
(SpeedRegulat... display status
Display
(SpeedRegulat... stop regulation
«include»

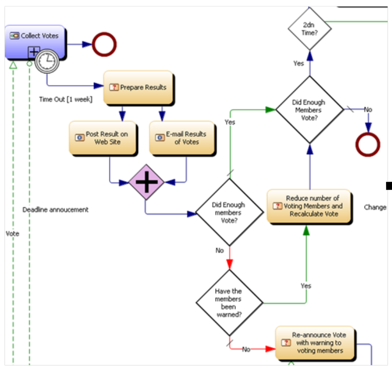
Szoftvermodell



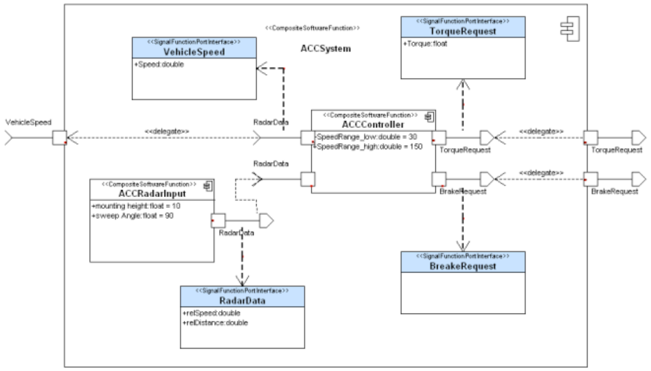
Szoftver-tervező

Szakterület-specifikus modellezési nyelvek

Üzleti elemző

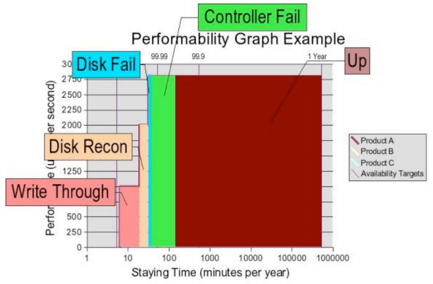


Üzleti folyamat



Rendszer-tervező

Megbízhatósági szakember

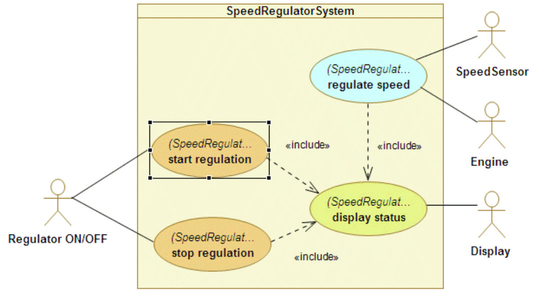


Megbízhatósági modell

Szoftver-fejlesztő

```
import com.lauchenauer.iatockhelper.
import com.lauchenauer.lib.ui.Vertic
import com.lauchenauer.lib.util.Brow

public class AboutDialog extends JDia
  protected CardLayout mLayout;
  protected JButton mCredits;
  protected JPanel mMainPanel;

  public AboutDialog(JFrame owner) {
    super(owner);
    setModal(true);
    setUndecorated(true);
    initUI();
  }

  protected void initUI() {
    setSize(440, 600);
    Container cont = getContent
    JPanel p =
```

Programozási nyelv



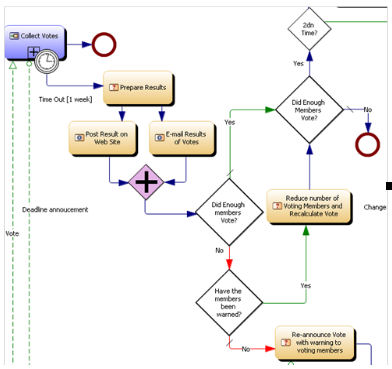
SpeedRegulatorSystem
(SpeedRegulat... regulate speed
SpeedSensor
Engine
(SpeedRegulat... start regulation
«include»
«include»
(SpeedRegulat... display status
Regulator ON/OFF
(SpeedRegulat... stop regulation
«include»
Display

Szoftvermodell

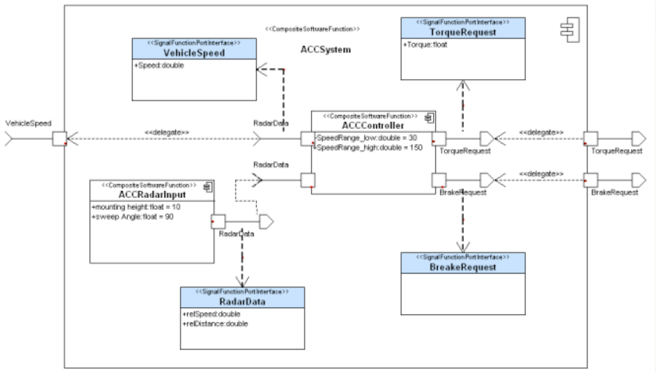
Szoftver-tervező

Szakterület-specifikus modellezési nyelvek

Üzleti elemző

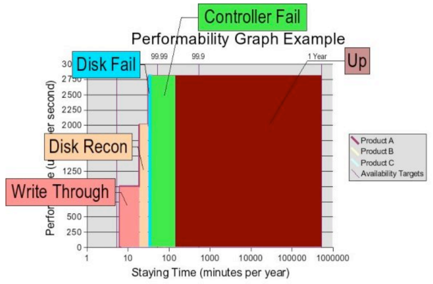


Üzleti folyamat

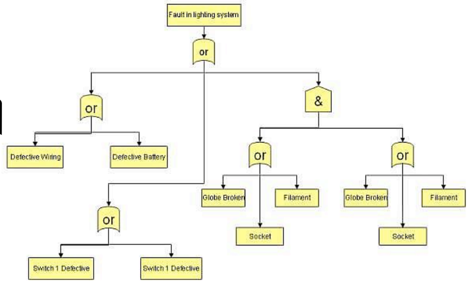

<<CompositeSoftwareFunction>>
ACCSystem
<<SignalFunctionPortInterface>>
VehicleSpeed
+Speed:double
<<SignalFunctionPortInterface>>
TorqueRequest
+Torque:float
VehicleSpeed
<<delegate>>
RadarData
<<CompositeSoftwareFunction>>
ACCController
+SpeedRange_low:double = 30
+SpeedRange_high:double = 150
TorqueRequest
<<CompositeSoftwareFunction>>
ACCRadarInput
+mounting height:float = 10
+sweep Angle:float = 90
BrakeRequest
<<SignalFunctionPortInterface>>
BreakeRequest
<<SignalFunctionPortInterface>>
RadarData
+relSpeed:double
+relDistance:double

Rendszer-tervező

Megbízhatósági szakember



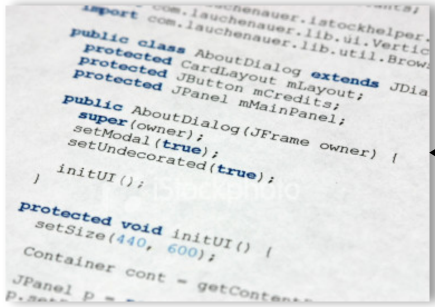
Megbízhatósági modell



Kockázati modell

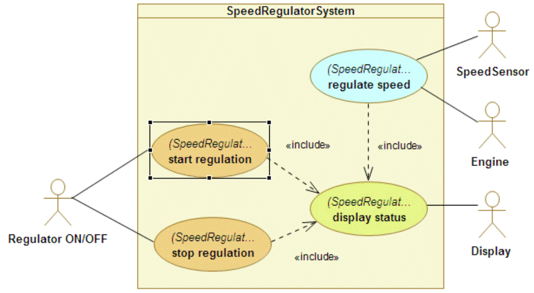
Biztonsági szakember

Szoftver-fejlesztő



```
import com.lauchenauer.iatockhelper.
       com.lauchenauer.lib.ui.Vertic
       com.lauchenauer.lib.util.Brow

public class AboutDialog extends JDia
  protected CardLayout mLayout;
  protected JButton mCredits;
  protected JPanel mMainPanel;

  public AboutDialog(JFrame owner) {
    super(owner);
    setModal(true);
    setUndecorated(true);

    initUI();
  }

  protected void initUI() {
    setSize(440, 600);

    Container cont = getContent
    JPanel p =
```

Programozási nyelv



Szoftvermodell

Szoftver-tervező

Szakterület-specifikus modellezés definíciója

Szakterület-specifikus modellezés

Dedikált modellezési nyelvek

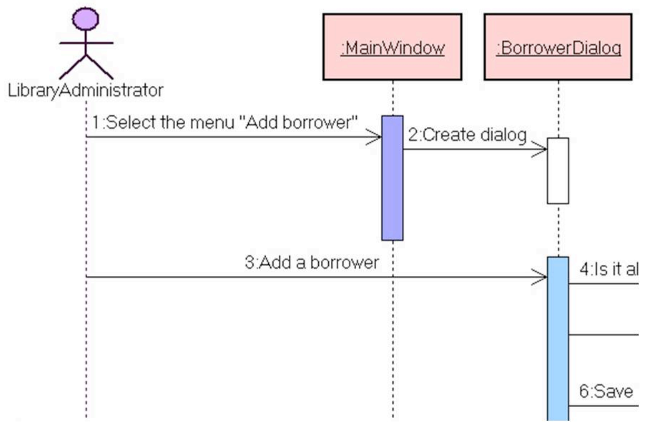
- Különböző absztrakciós szinten
- A teljes fejlesztési folyamatra

Generatív fejlesztés (kódgenerálás)

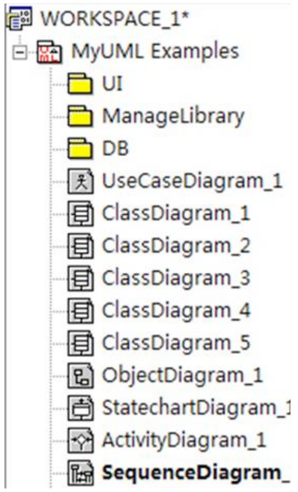
- Állapotgépek → Végrehajtható kód

Szakterület-specifikus nyelvek felépítése

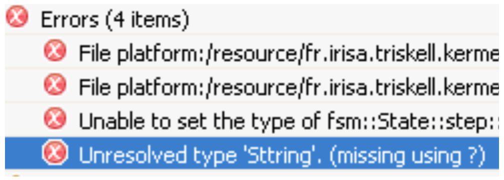
Konkrét szintakszis



Absztrakt szintakszis

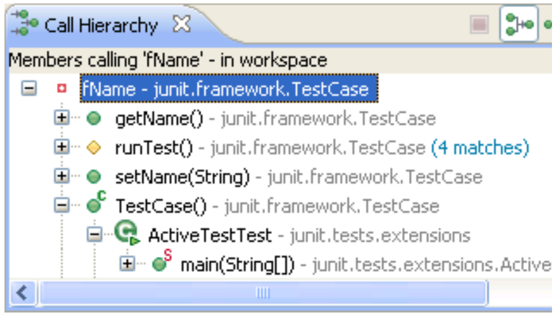


Jólformáltsági kényszerek



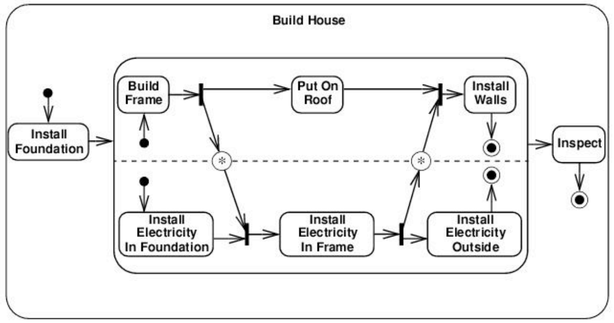
Errors (4 items)
File platform:/resource/fr.irisa.triskell.kerme
File platform:/resource/fr.irisa.triskell.kerme
Unable to set the type of fsm::State::step::
Unresolved type 'Sttring'. (missing using ?)

Viselkedési szemantika, Szimuláció, Refactoring



Call Hierarchy
Members calling 'fName' - in workspace
fName - junit.framework.TestCase
getName() - junit.framework.TestCase
runTest() - junit.framework.TestCase (4 matches)
setName(String) - junit.framework.TestCase
TestCase() - junit.framework.TestCase
ActiveTestTest - junit.tests.extensions
main(String[]) - junit.tests.extensions.Active

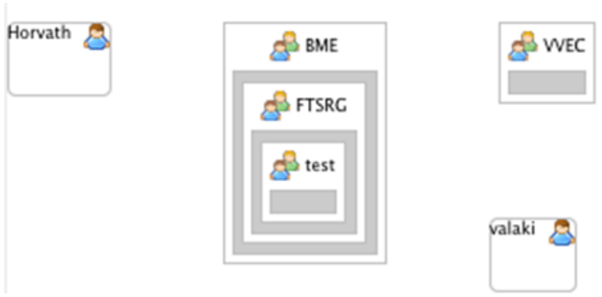
Hívási gráf (Nézet)


Build House
Install Foundation
Build Frame
Put On Roof
Install Walls
Inspect
Install Electricity In Foundation
Install Electricity In Frame
Install Electricity Outside

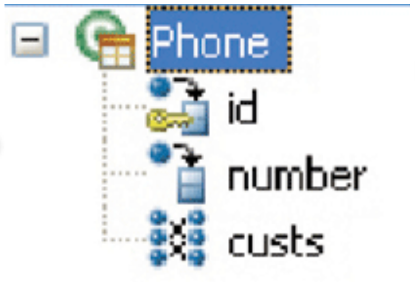
Állapottérkép (másik DSM)

Szakterület-specifikus nyelvek felépítése

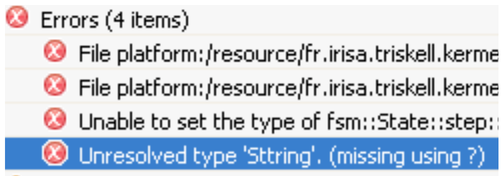
Grafikus szintakszis



Absztrakt szintakszis



Jólformáltsági kényszerek

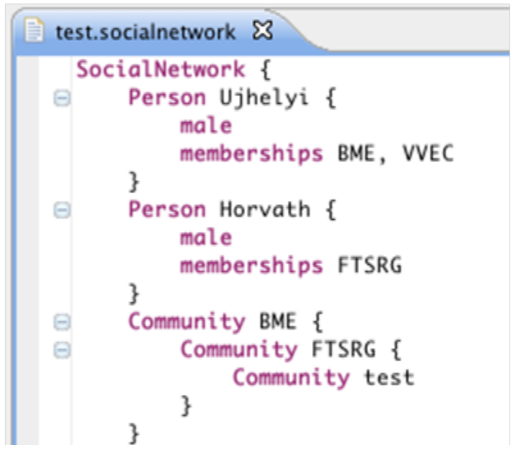


```
Errors (4 items)
  File platform:/resource/fr.irisa.triskell.kerme
  File platform:/resource/fr.irisa.triskell.kerme
  Unable to set the type of fsm::State::step::
  Unresolved type 'String'. (missing using ?)
```

Viselkedési szemantika, Szimuláció

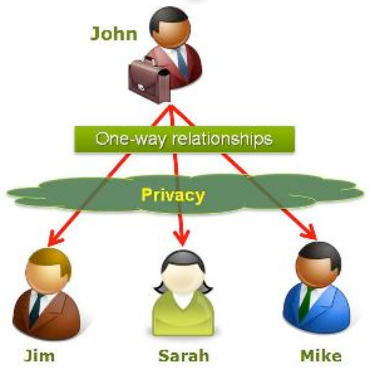
Leképezés

Kódgenerálás

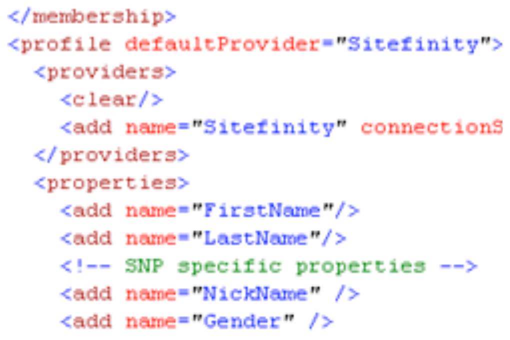


```
test.socialnetwork
SocialNetwork {
    Person Ujhelyi {
        male
        memberships BME, VVEC
    }
    Person Horvath {
        male
        memberships FTSRG
    }
    Community BME {
        Community FTSRG {
            Community test
        }
    }
```

Szöveges szintakszis



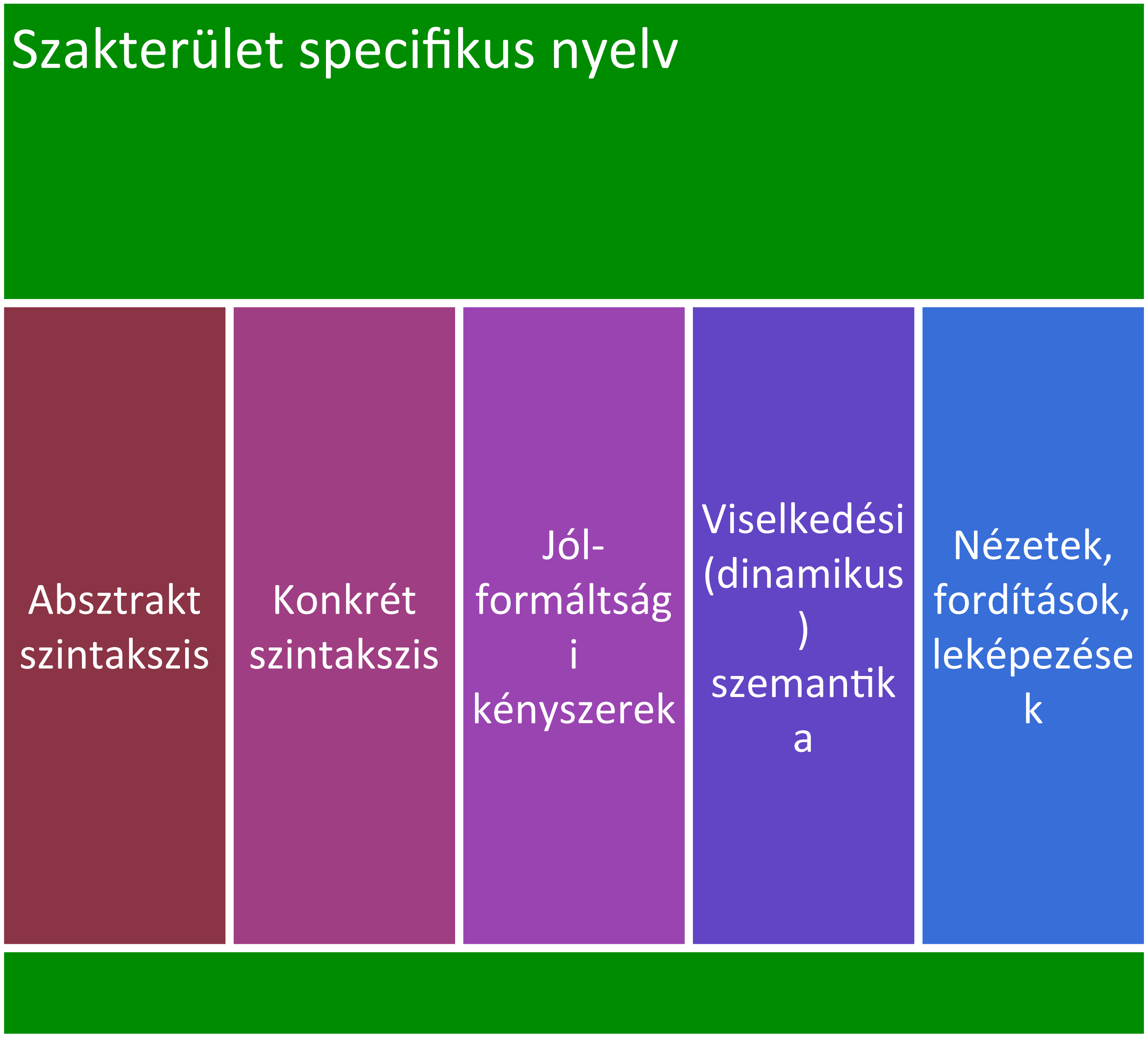
Nézet



```
</membership>
<profile defaultProvider="Sitefinity">
  <providers>
    <clear/>
    <add name="Sitefinity" connectionS
  </providers>
  <properties>
    <add name="FirstName"/>
    <add name="LastName"/>
    <!-- SNP specific properties -->
    <add name="NickName" />
    <add name="Gender" />
```

Kód (Dokumentáció, Konfigurációs fájl)

Szakterület-specifikus nyelvek aspektusai


Szakterület specifikus nyelv
Absztrakt szintakszis
Konkrét szintakszis
Jól-formáltsági kényszerek
Viselkedési (dinamikus) szemantika
Nézetek, fordítások, leképezések

METAMODELING

Why?

- Let’s do Model-based Development!
- Create **model**s that…
  - have well-defined, standardized form and meaning
  - are processable by computers
    - Storage, Parsing, Editing, Visualization,
    - Execution, Testing,
    - Analysis, Verification,
    - Translation, Transformation, Integration, Synchronization
  - are easy to use (create / understand)
- Need to design **modeling language**s

Where do you find metamodels?

- Different application domains around UML
  - SysML (systems engineering)
  - SPEM (process modeling)
  - CWM (data warehousing)
  - MARTE (real-time and embedded systems)
- BPEL, BPMN (business processes)
- Tropos (requirements modeling)
- AutoSAR (automotive industry)
- …

Designing modeling languages

Designing modeling languages

- Core concept: **metamodeling**
  - Design methodology of modeling languages
  - Metamodel = model of a modeling language

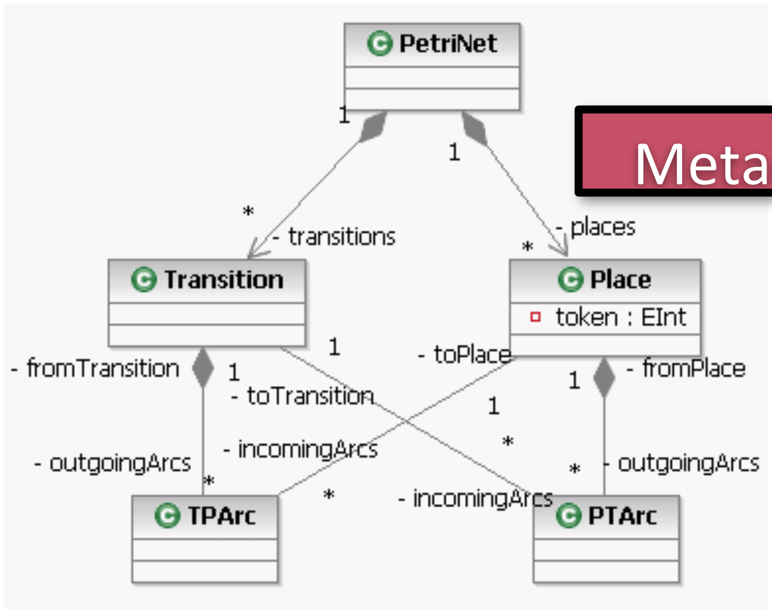
Designing modeling languages

- Core concept: **metamodeling**
  - Design methodology of modeling languages
  - Metamodel = model of a modeling language
- Language design checklist
  - **Abstract syntax** (metamodel)
    - Taxonomy and relationships of model elements
    - Well-formedness rules
  - **Semantics** (does not *strictly* belong to a language)
    - Static
    - Behavioural
  - ??? (something is missing… we'll come back later)

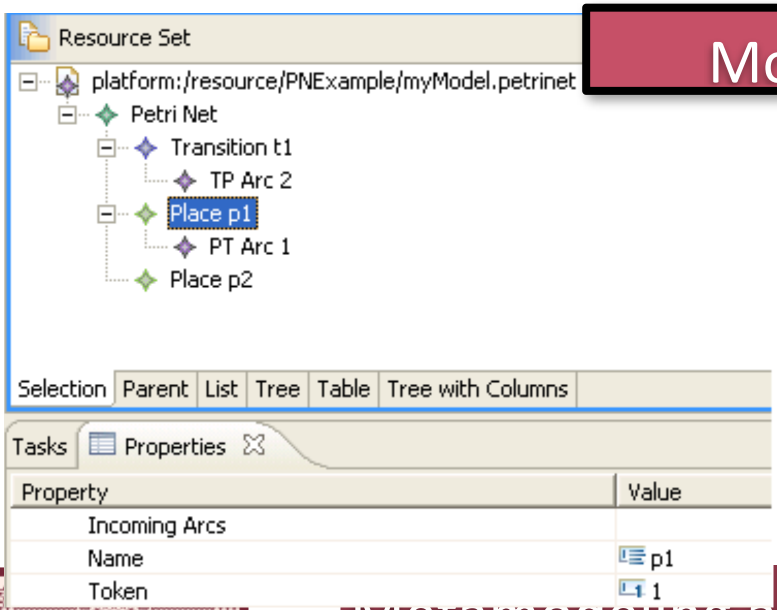
Abstract syntax (Metamodel)

- Metamodel = model of a modeling language
  - „Meta” = above, beyond, transcending
- Goal: to define
  - The vocabulary of concepts in the language
  - How they can be combined to form models
- Contents:
  - Definition of concepts
  - Relationships between these concepts
    - Abstraction/Specialization (Taxonomy)
  - Constraints, well-formedness rules (e.g. multiplicity)

Metamodell: Absztrakt szintakszis definíciója



Metamodell



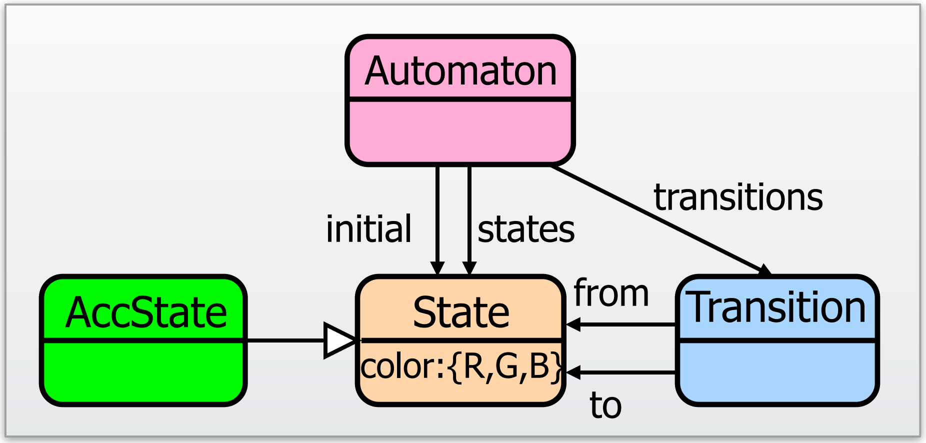
Resource Set
platform:/resource/PNExample/myModel.petrinet
Petri Net
Transition t1
TP Arc 2
Place p1
PT Arc 1
Place p2

Property	Value
Incoming Arcs	
Name	p1
Token	1

Modell

- Metamodell:
  - Modellezési nyelvek absztrakt szintakszisát definiáló DSM
- Célja: definiálni...
  - a nyelv alapfogalmait
  - a köztük lehetséges kapcsolatokat
  - az alapfogalmak attribútumait
  - absztrakció/finomítás (Taxonómia, Ontológia) az elemek között
  - tartalmazási, számossági kényszerek

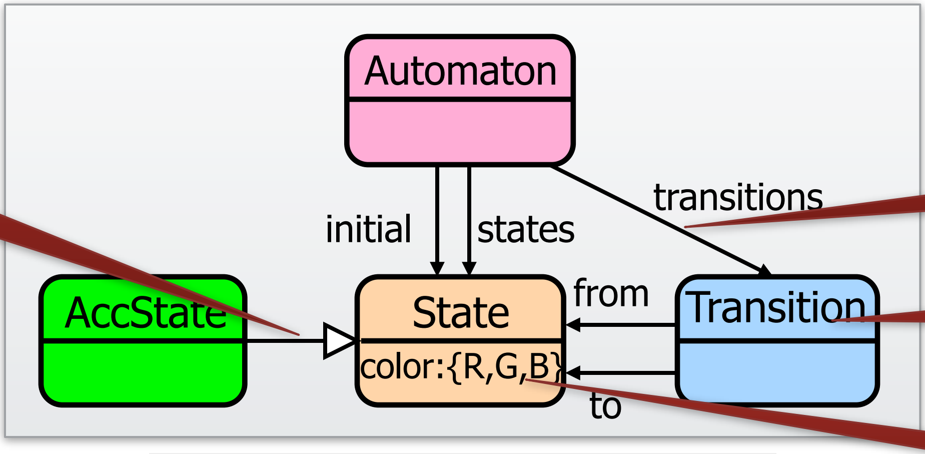
Example



Metamodel

Meta (Language) level

Example



Generalization

Association

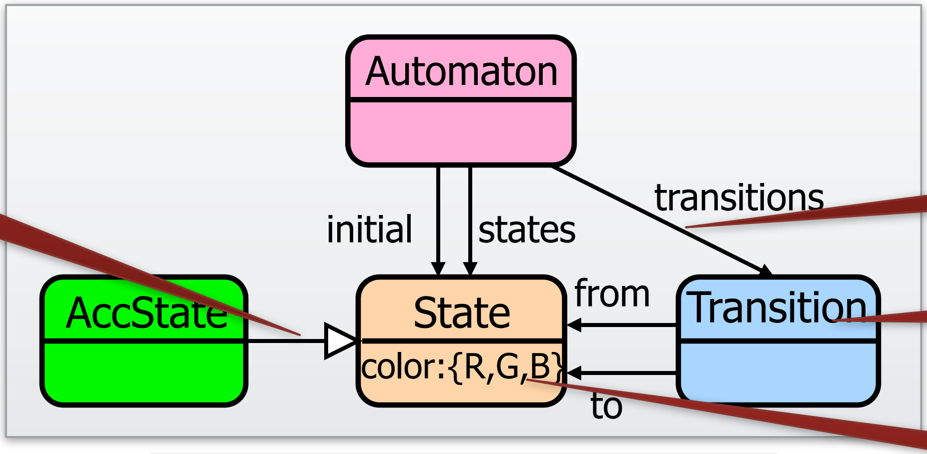
Class

Attribute

Metamodel

Meta (Language) level

Example



Generalization

Association

Class

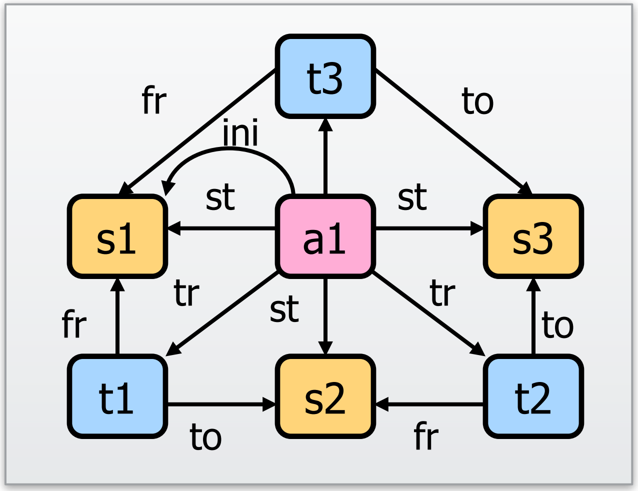
Instantiation

Attribute

Metamodel

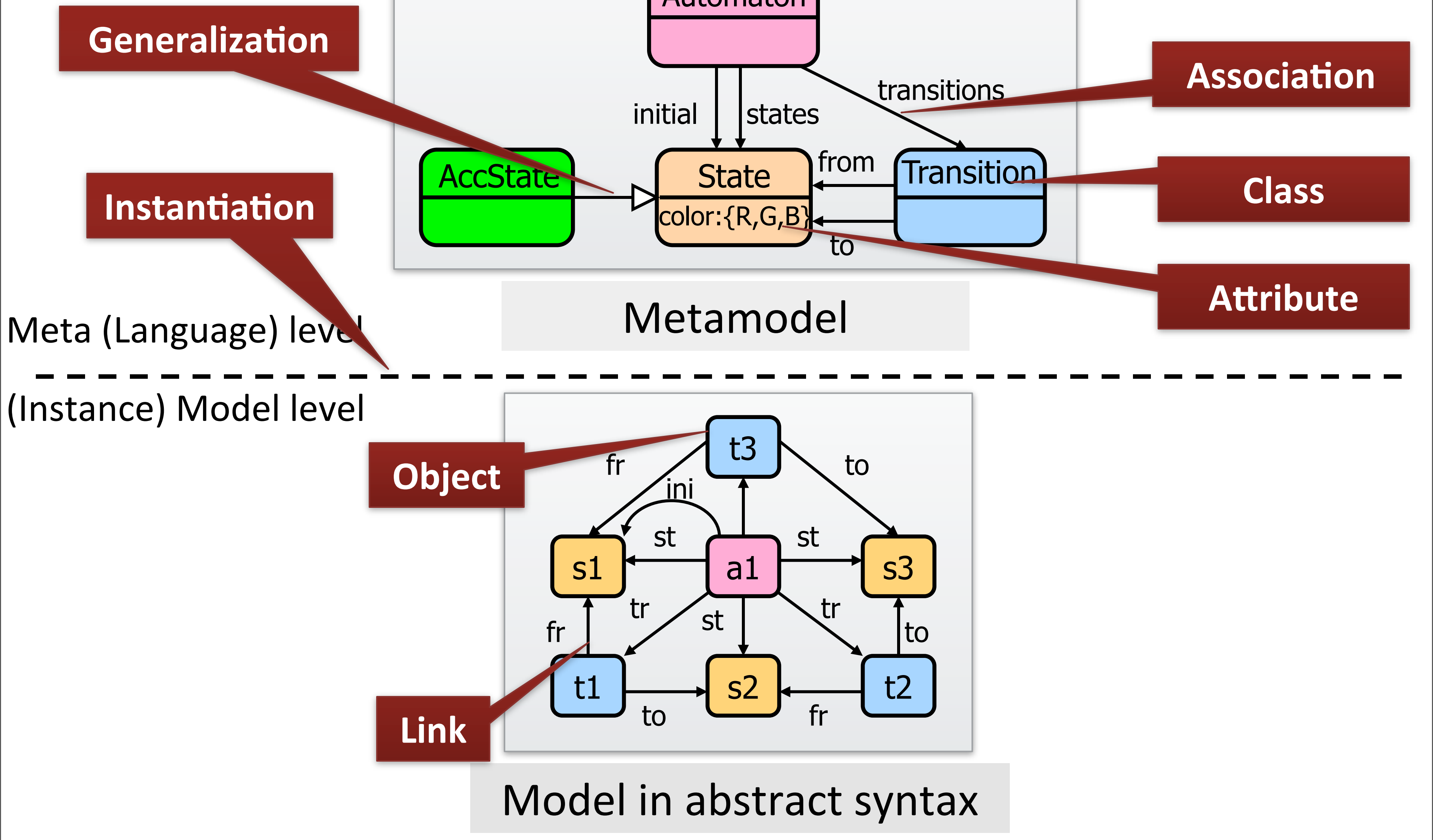
Meta (Language) level

(Instance) Model level



Model in abstract syntax

Example



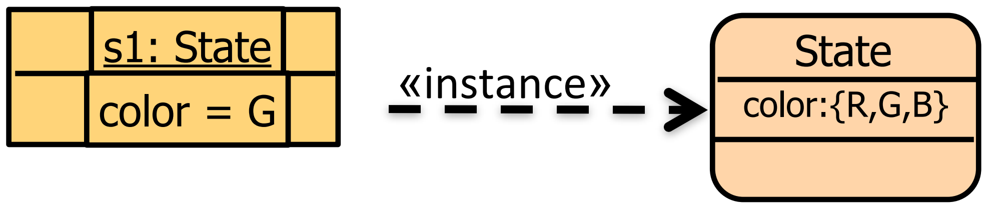
Generalization
Association
Class
Instantiation
Attribute
Object
Link
Automaton
initial
states
transitions
AccState
State
color:{R,G,B}
from
to
Transition
Metamodel
Meta (Language) level
(Instance) Model level
t3
fr
to
ini
st
s1
a1
s3
tr
fr
st
tr
to
t1
s2
t2
to
fr
Model in abstract syntax

Well-formedness rules

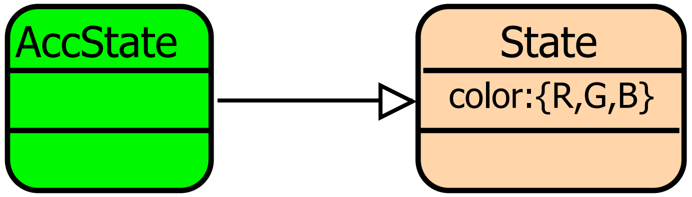
- Multiplicity constraints
  - At most one: 0..1 / Many: *
  - Lower bound is often meaningful (enforcement?)
- Aggregation/Containment
  - At most one parent for each model element
- Language specific constraints:
  - Examples
    - Each state of an automaton must have a unique name
    - Transitions must connect states of their own automaton
    - The initial state is one of the states of the automaton
  - Expressed in e.g. OCL

Instantiation and Generalization

- Classification/Typing
  - inverse: Instantiation

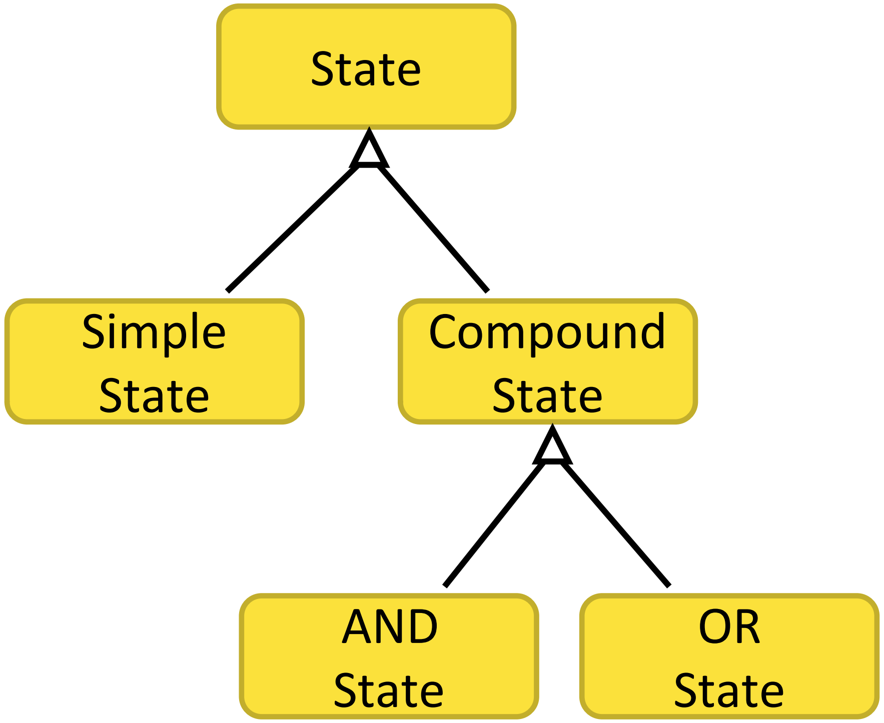


- Generalization / Supertyping / Abstraction
  - inverse: Specialization / Subtyping / Refinement



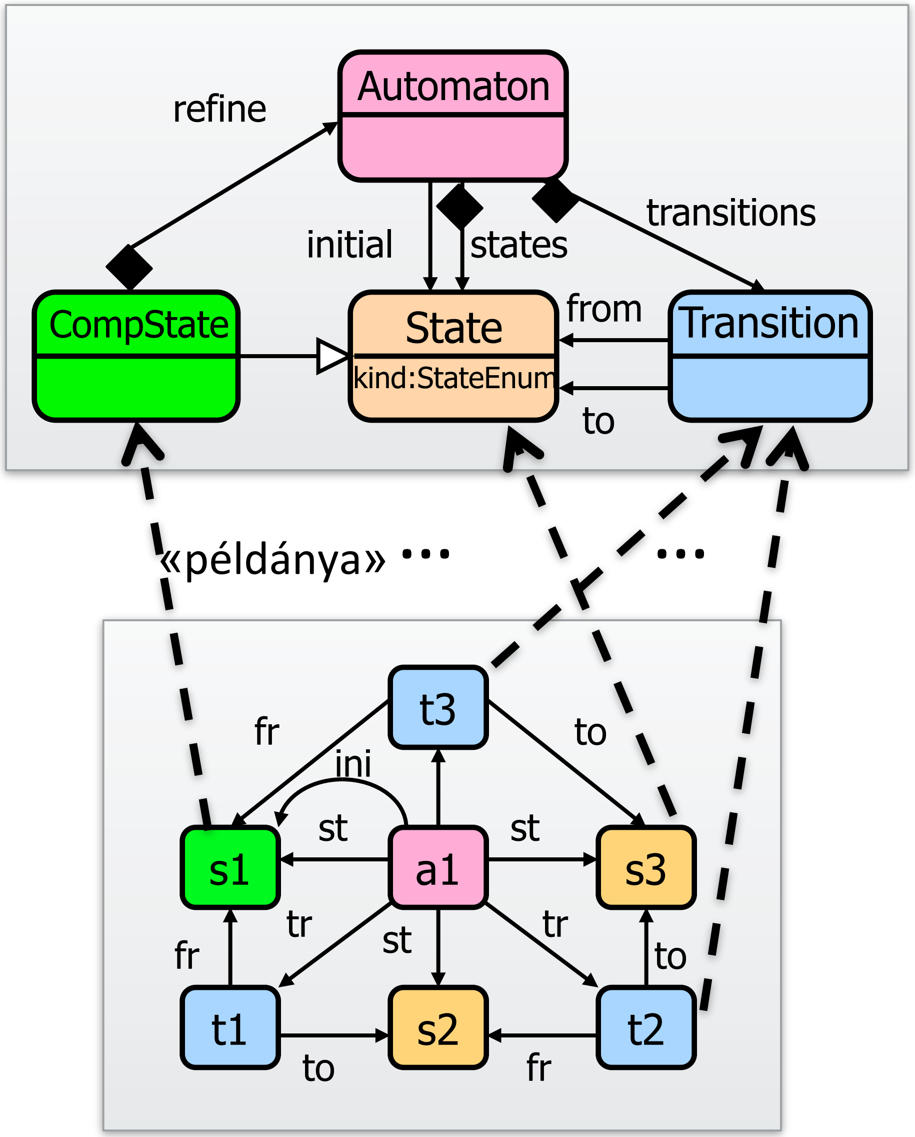
- More is implied than what is explicitly given
  - transitive semantics
    - **self**.supertypes→**includesAll**(**self**.supertypes.supertypes)
  - extends the typing relation
    - **self**. instances->**includesAll**(**self**. subtypes.instances)

Típushierarchia



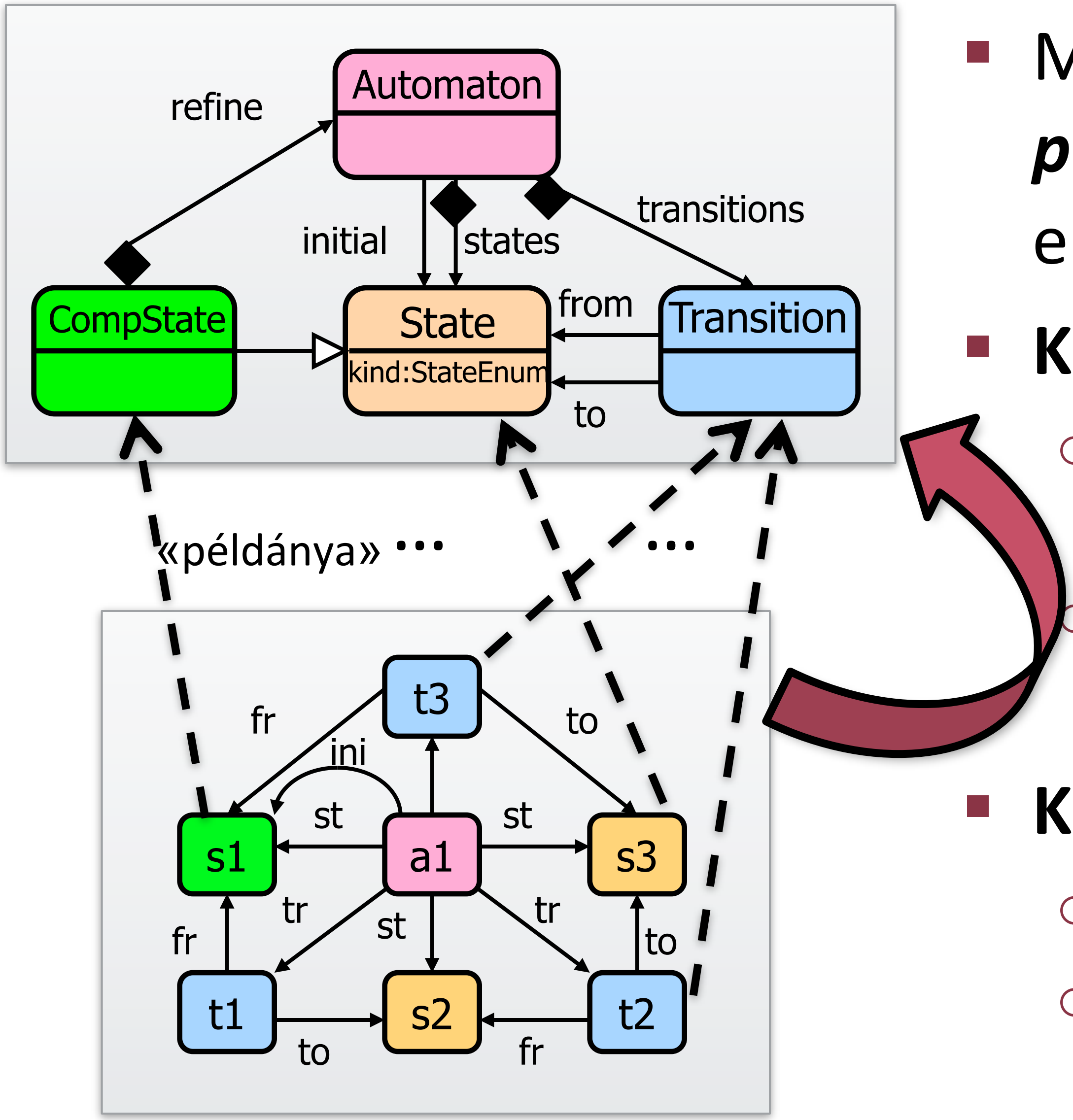
- Általánosítás (Generalization)
  - ≅ Öröklés
  - Tranzitív
  - Irreflexív?
- Helyettesíthetőség
  - ✓ Ős helyett Leszármazott
  - × ~~Leszármazott helyett Ős~~

Példányosítás



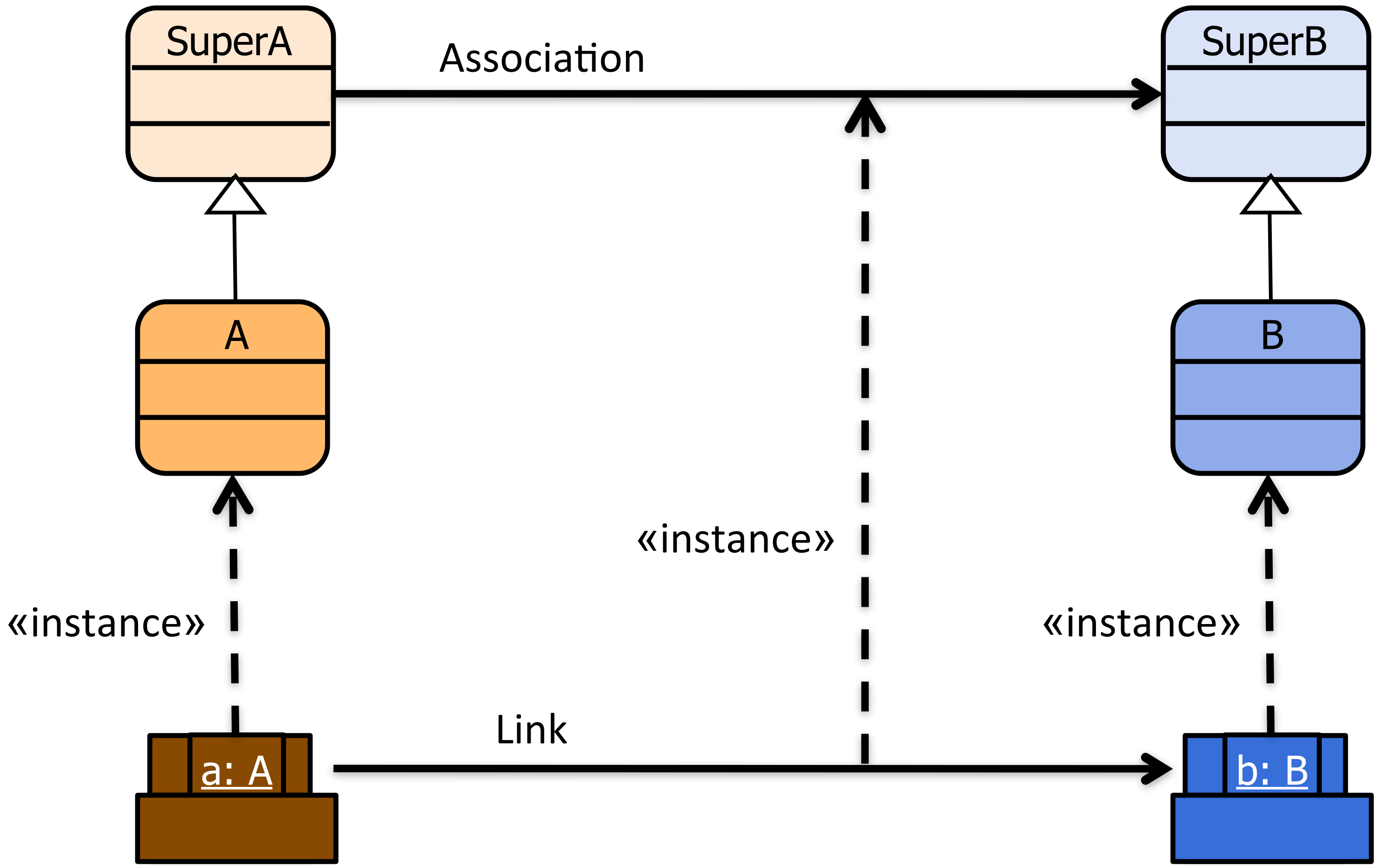
- Minden modellelem ***példánya*** a metamodell egy elemének
- **Közvetlen típus**:
  - Nincs precízebb (alacsonyabb) típusa
  - s1 → AccState
- **Közvetett típus**:
  - A közvetlen típus ősosztálya
  - s1 → State

Példányosítás



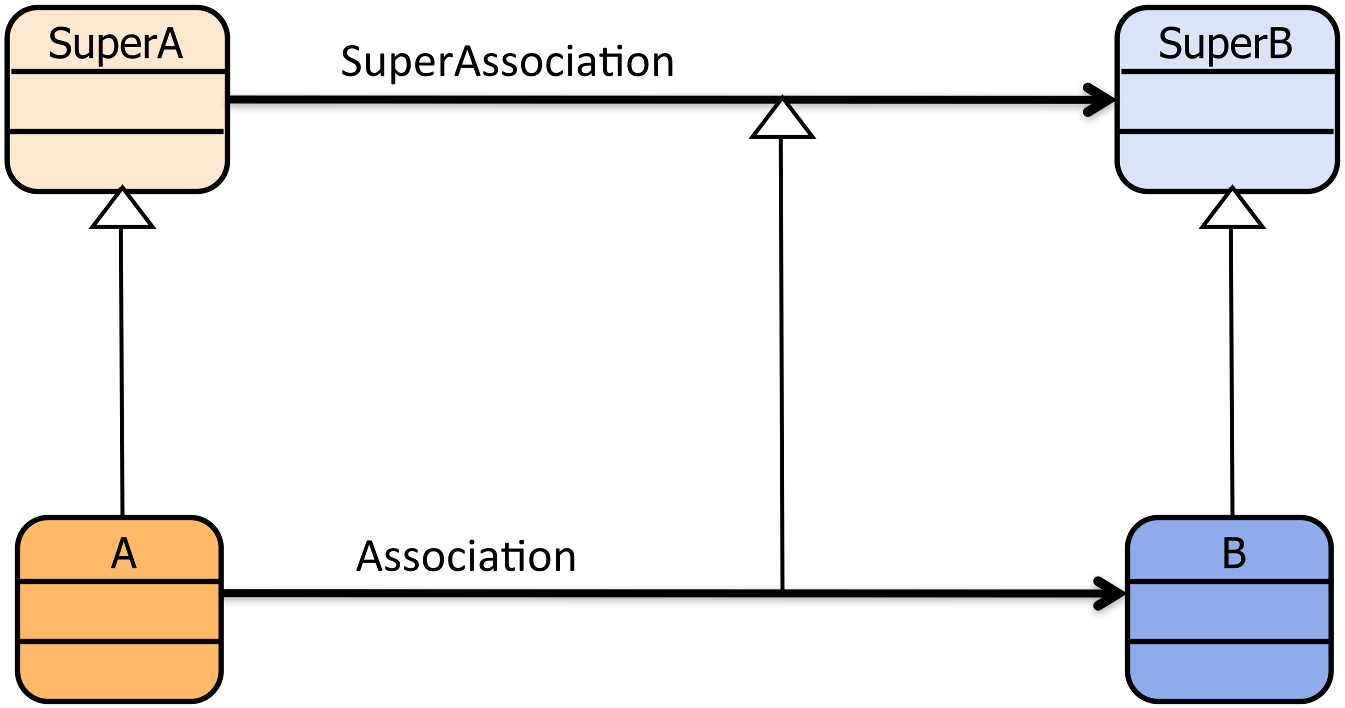
- Minden modellelem ***példánya*** a metamodell egy elemének
- **Közvetlen típus**:
  - Nincs precízebb (alacsonyabb) típusa
  - s1 → AccState
- **Közvetett típus**:
  - A közvetlen típus ősosztálya
  - s1 → State

Type Conformance of Edges

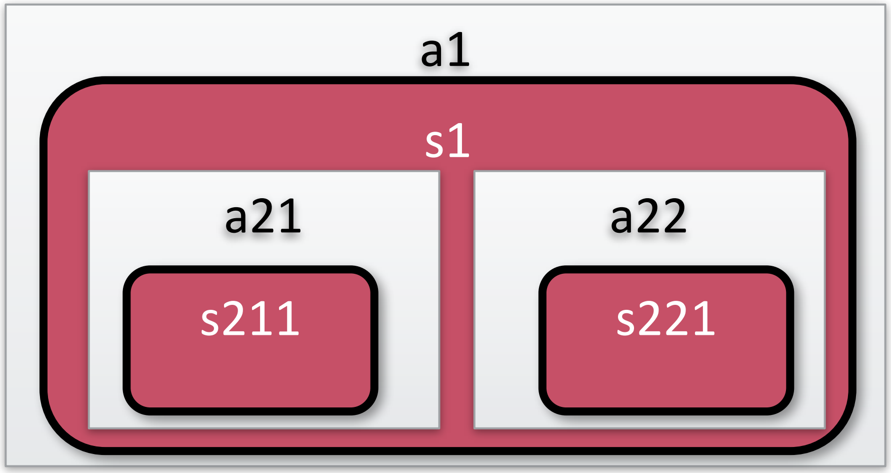


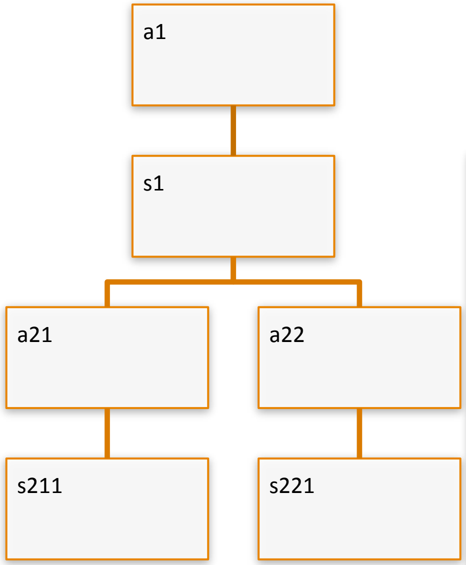
Type Conformance of Edges

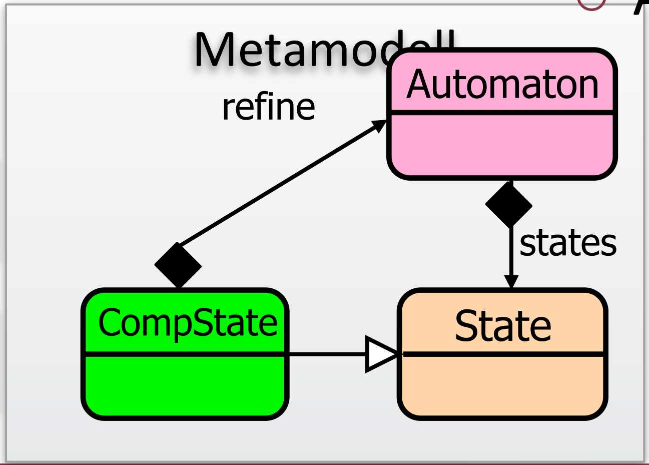
- Subtyping of edges
  - Not allowed in e.g. UML



Tartalmazási hierarchia



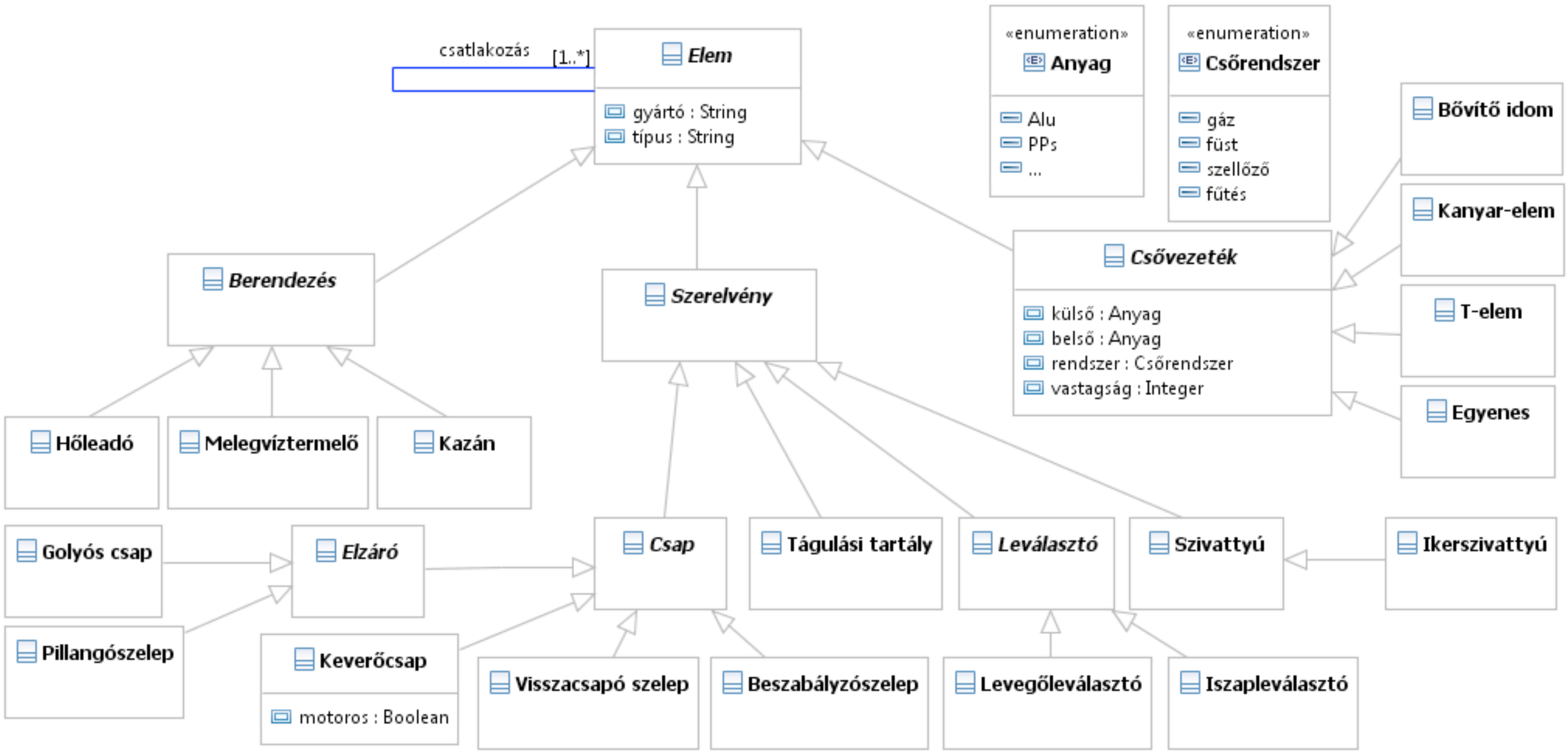




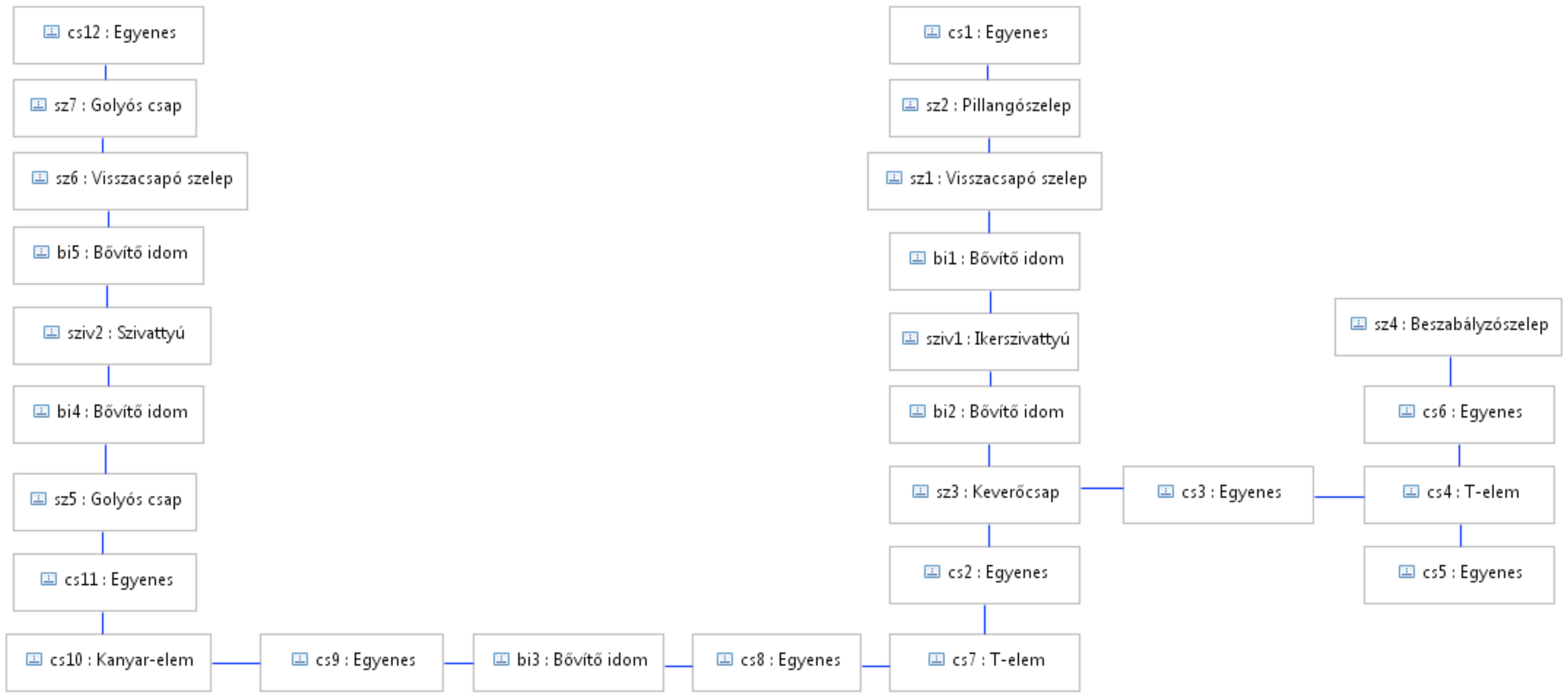
- Minden modellelem tulajdonosa egyértelmű
  - N Gyerek → 1 Szülő
  - Egyetlen gyökérelem
- Tartalmazási viszony:
  - Metamodellben definiáljuk
    - Élek (referencia) mentén
  - A modelleken értelmezzük

DOMAIN SPECIFIC MODELING

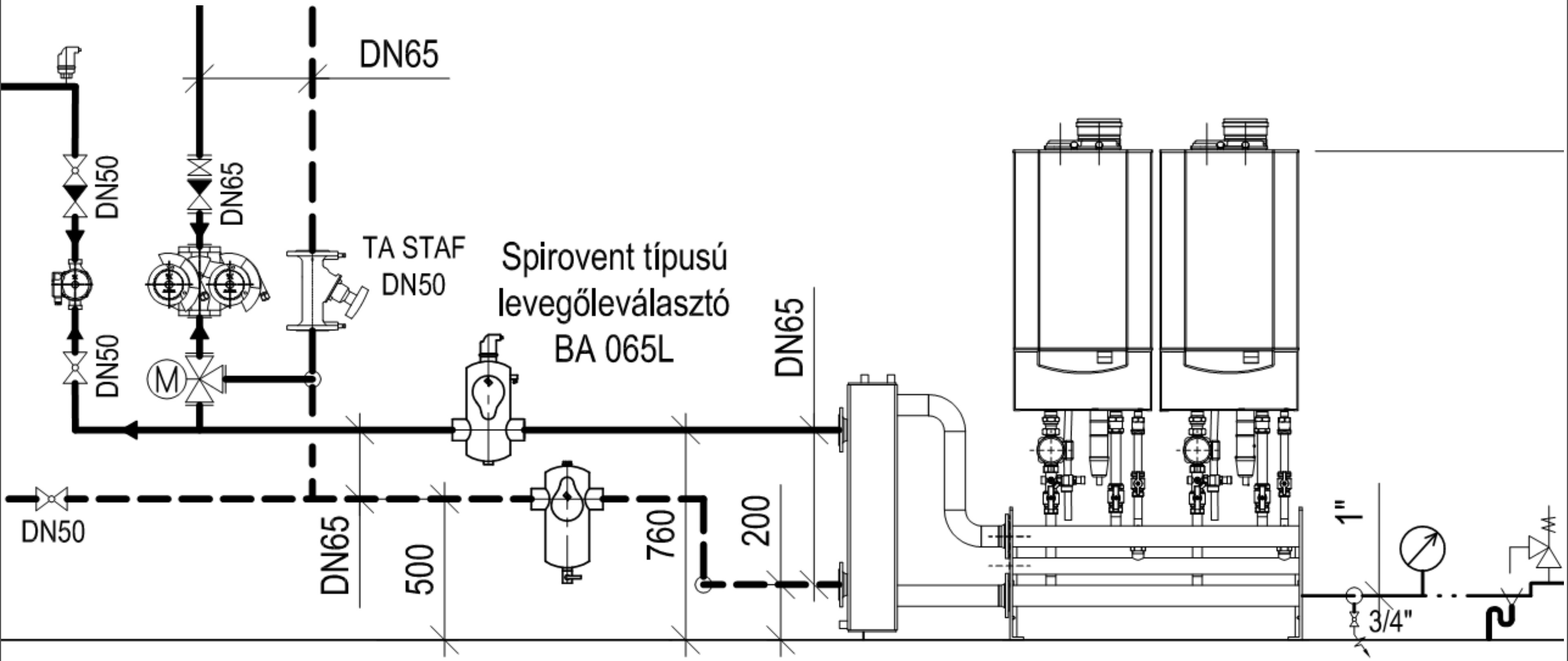
Example metamodel



Instance model, abstract syntax



Instance model, concrete syntax

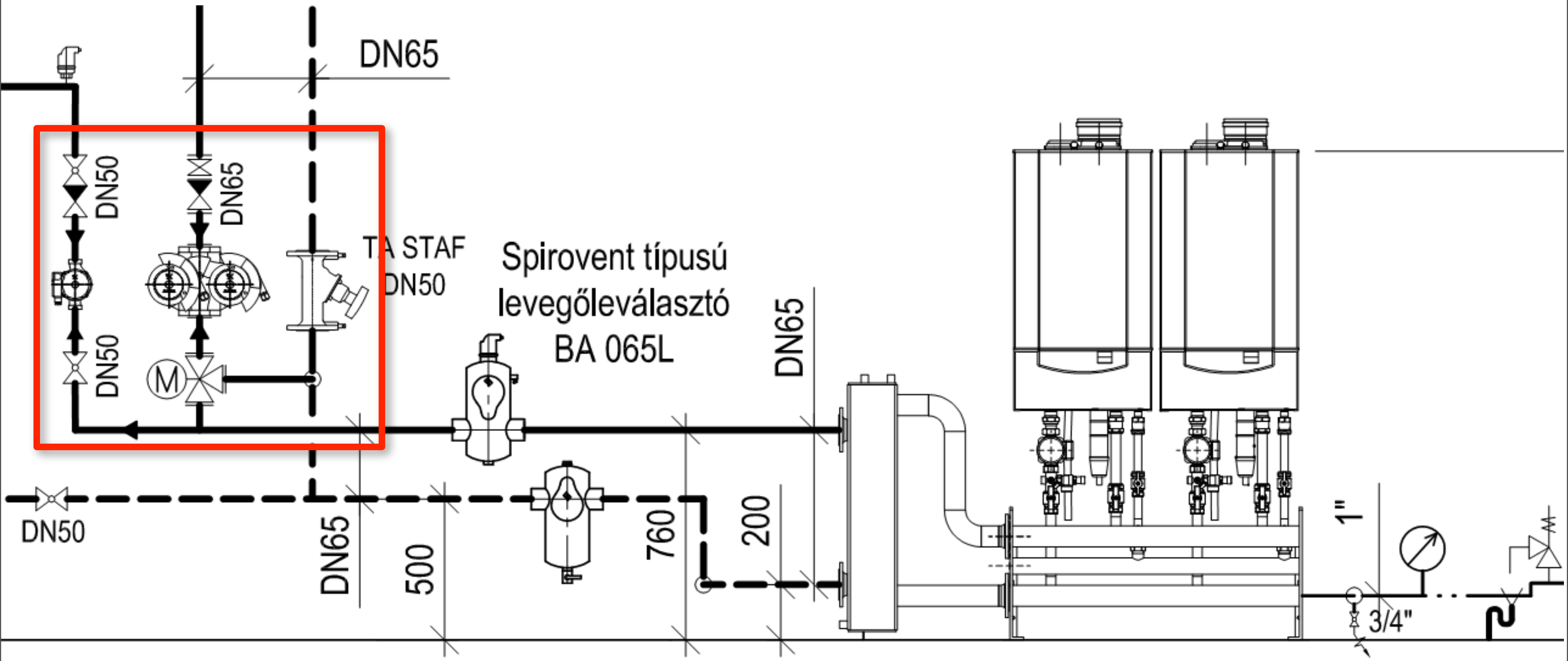


Honeywell
keverőcsap
DN50 Kvs 40

Spirovent típusú
iszapleválasztó
BE 065L

Remeha Quinta kaszkád
rendszer hidraulikus váltóval

Instance model, concrete syntax



Honeywell
keverőcsap
DN50 Kvs 40

Spirovent típusú
iszapleválasztó
BE 065L

Remeha Quinta kaszkád
rendszer hidraulikus váltóval

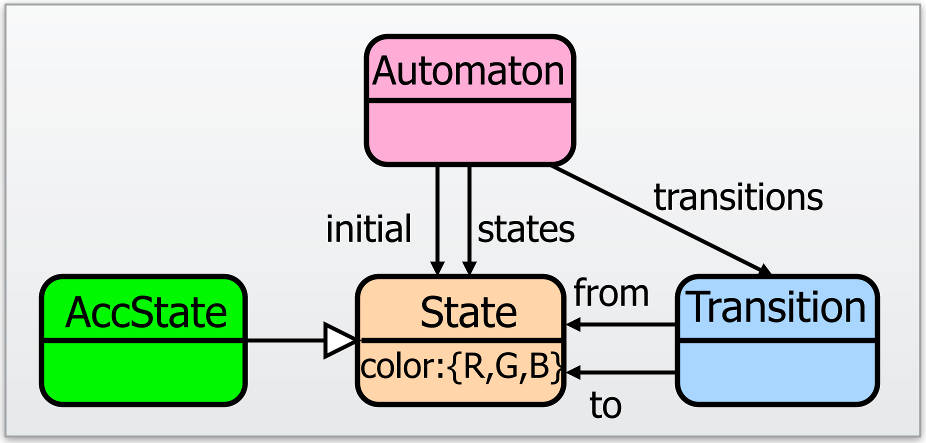
Designing modeling languages

Designing modeling languages

- Language design checklist
  - **Abstract syntax** (metamodel)
    - Taxonomy and relationships of model elements
    - Well-formedness rules
  - **Semantics** (does not *strictly* belong to a language)
    - Static
    - Behavioural
  - ??? (something is missing… we'll come back later)
  - **Concrete syntax**
    - Textual notation
    - Visual notation

Designing modeling languages

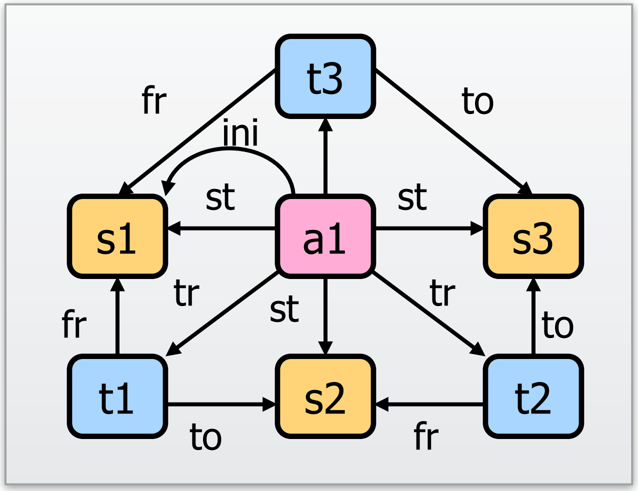
Relationship of concepts



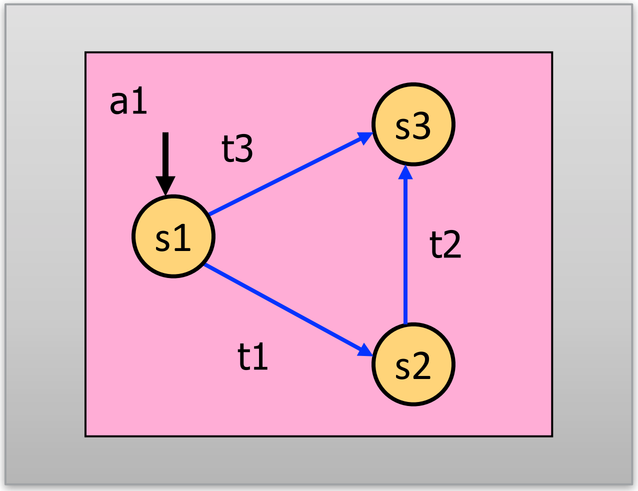
Metamodel

Meta (Language) level

Model level



Abstract syntax

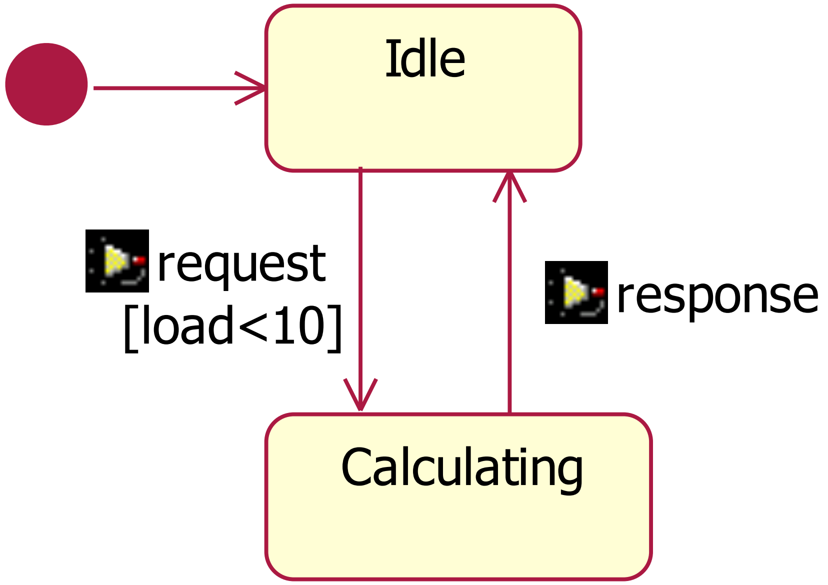


Concrete syntax

Textual vs. Visual

- Textual notation:
  - + Easy to write: Able to capture complex expressions
  - - Difficult to read: Difficult to comprehend and manage after certain complexity
- Visual notation:
  - + Easy to read: Able to express (selected / subset of) details in an intuitive, understandable form
  - + Safe to write: Able to construct syntactically correct models
  - - Difficult to write: graphical editing is slower

Example: Concrete Syntax



```
request() {
    if (state == "idle" &&
    this.load<10)
    state = "calculating";
}
response() {
    if (state == "calculating")
    state = "idle"
}
```

Graphical notation

Textual notation

Multiplicity of Notations

- One-to-many
  - 1 abstract syntax → many textual and visual notations
    - Human-readable-writable textual or visual syntax
    - Textual syntax for exchange or storage (typically XML)
    - In case of UML, each diagram is only a partial view
  - 1 abstract model → many concrete forms in 1 syntax!
    - Whitespace, diagram layout
    - Comments
    - Syntactic sugar
  - 1 semantic interpretation → many abstract models

Semantics

- Semantics: the meaning of concepts in a language
  - Static: what does a snapshot of a model mean?
  - Dynamic: how does the model change/evolve/behave?
- Static Semantics
  - Interpretation of metamodel elements
  - Meaning of concepts in the abstract syntax
  - **Formal**: mathematical statements about the interpretation
    - E.g. formally defined semantics of OCL

Dynamic Semantics

- **Denotational** (Translational): translating concepts in one language to another language (called **semantic domain**)
  - „compiled”
  - E.g. explaining state machines as Petri-net
- **Operational**: modeling the operational behavior of language concepts
  - „interpreted”
  - Sometimes dynamic features are introduced only for formalizing dynamic sematics

DOMAIN-SPECIFIC MODELING LANGUAGES IN ENGINEERING PRACTICE

Well known DSLs

- MATLAB, SQL, Erlang, Shell scripts, AWK, Verilog, YACC, R,S, Mathematica, Mata, XSLT, XMI, OCL, Template languages, …

Industry standard DSMLs

- Automotive
  - AUTOSAR, MATLAB StateFlow, EAST-AADL
- Aerospace
  - AADL
- Railways
  - UML-MARTE
- Systems engineering
  - SysML, UML-FT

Technologies

- MATLAB
- Rational Software Architect

COTS

- Eclipse
  - EMF
  - openArchitectureWare
- Microsoft
  - DSL Tools (Visual Studio)
- MetaCase
  - MetaEdit+
- JetBrains MPS

Language engineering (industry)

- GEMS, GME, ViatraDSM

Academia

SUMMARY

Summary

- Metamodeling
  - Structural, formal definition of domains
  - Abstract syntax
- Domain-Specific Modeling
  - Concrete notations
  - Syntax known by experts of the field
- Metalevels
  - Meta-relationship between models
- Semantics
  - Formal dynamic → Denotational / Operational

Eclipse Modeling Framework

Horváth Ákos

Dániel Varró

Table of Contents

What does EMF provide?

EMF = Eclipse Modeling Framework

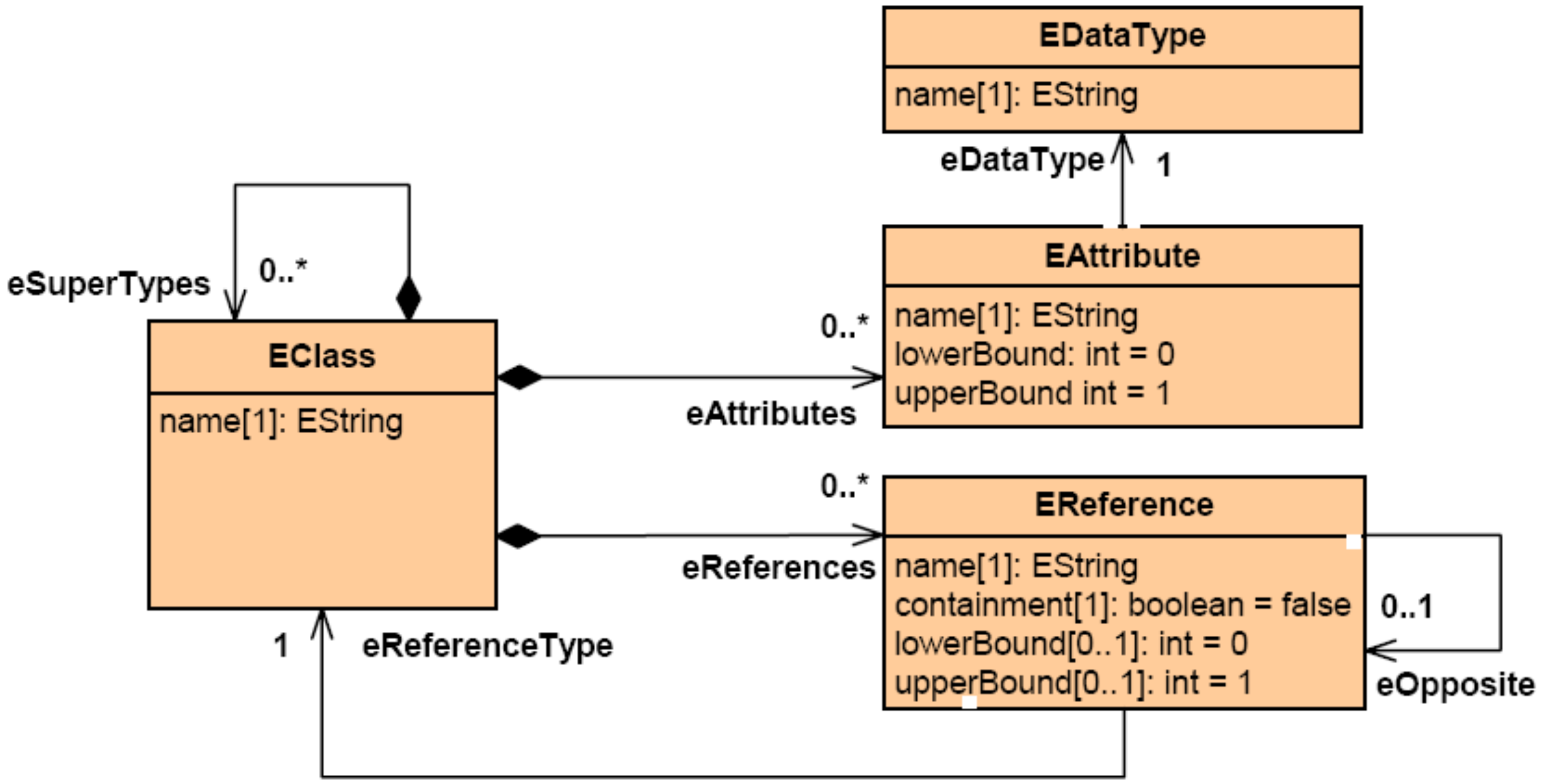
Reflective Metamodeling Core

Ecore - why?

Metamodeling language of EMF

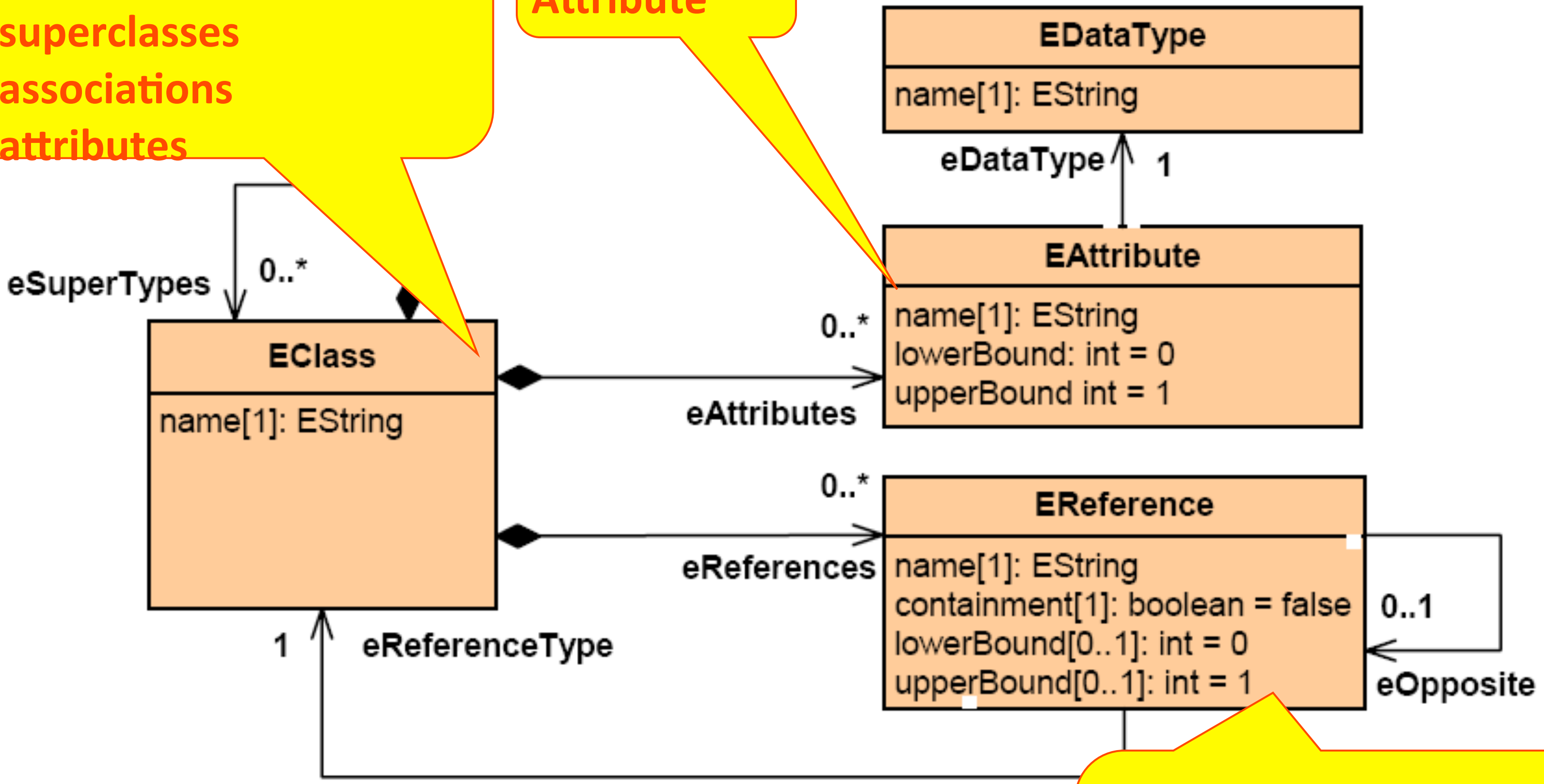
Metamodels are platform independent

Core EMF

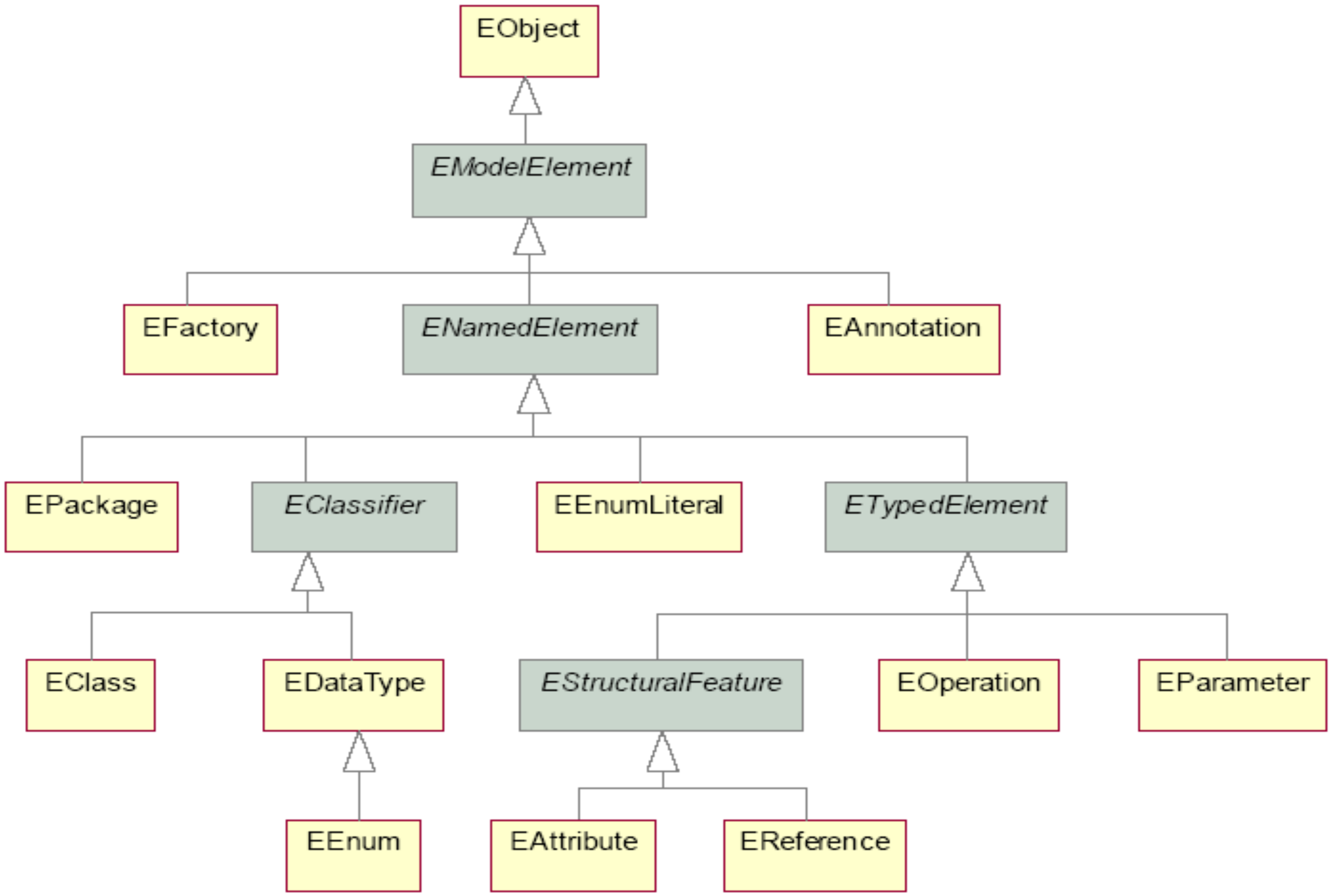

EDataType
name[1]: EString
eDataType
1
EAttribute
name[1]: EString
lowerBound: int = 0
upperBound int = 1
eSuperTypes
0..*
EClass
name[1]: EString
0..*
eAttributes
0..*
eReferences
EReference
name[1]: EString
containment[1]: boolean = false
lowerBound[0..1]: int = 0
upperBound[0..1]: int = 1
0..1
eOpposite
1
eReferenceType

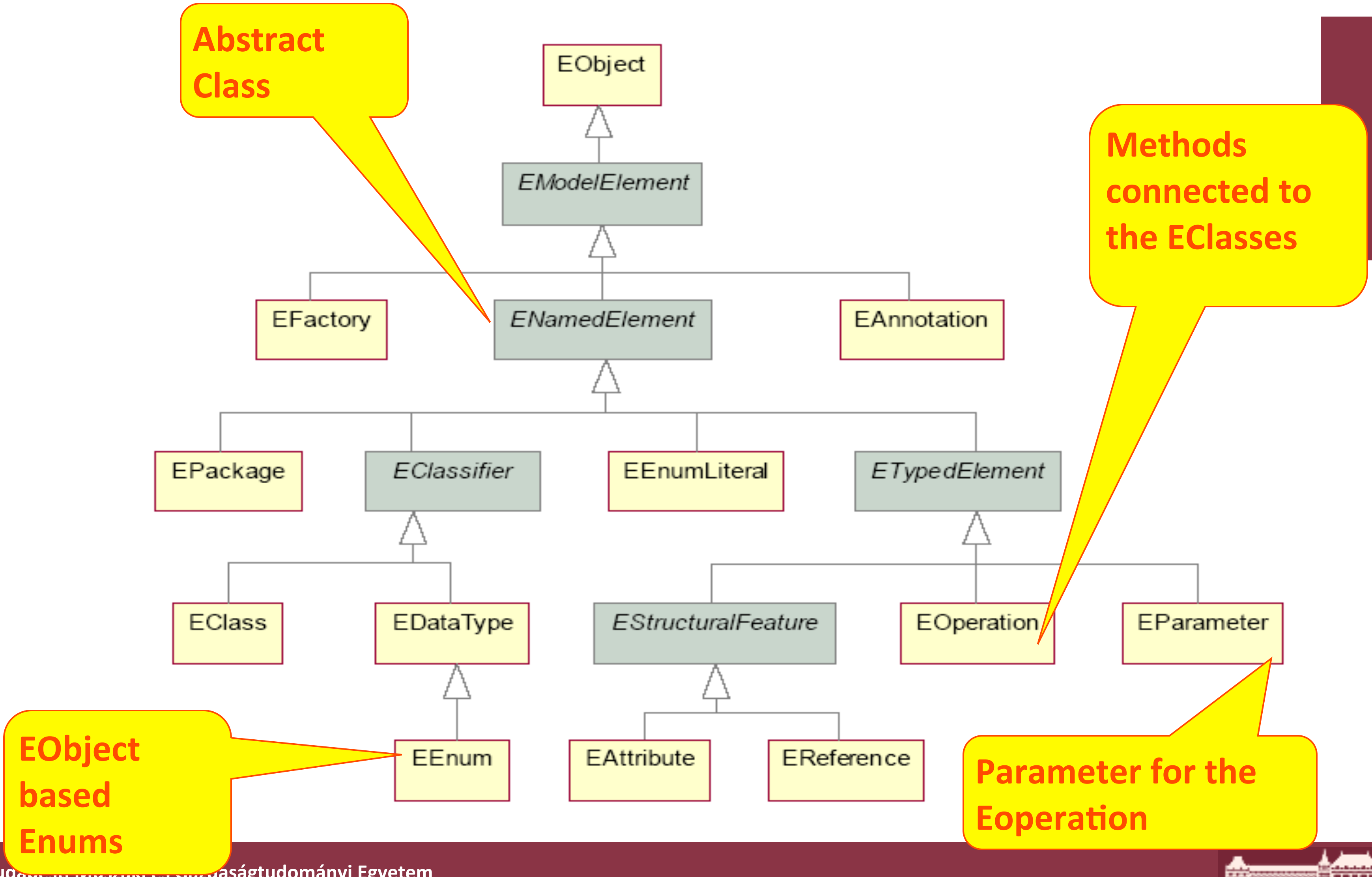
Class with arbitrary num. of
- superclasses
- associations
- attributes

Typed Attribute

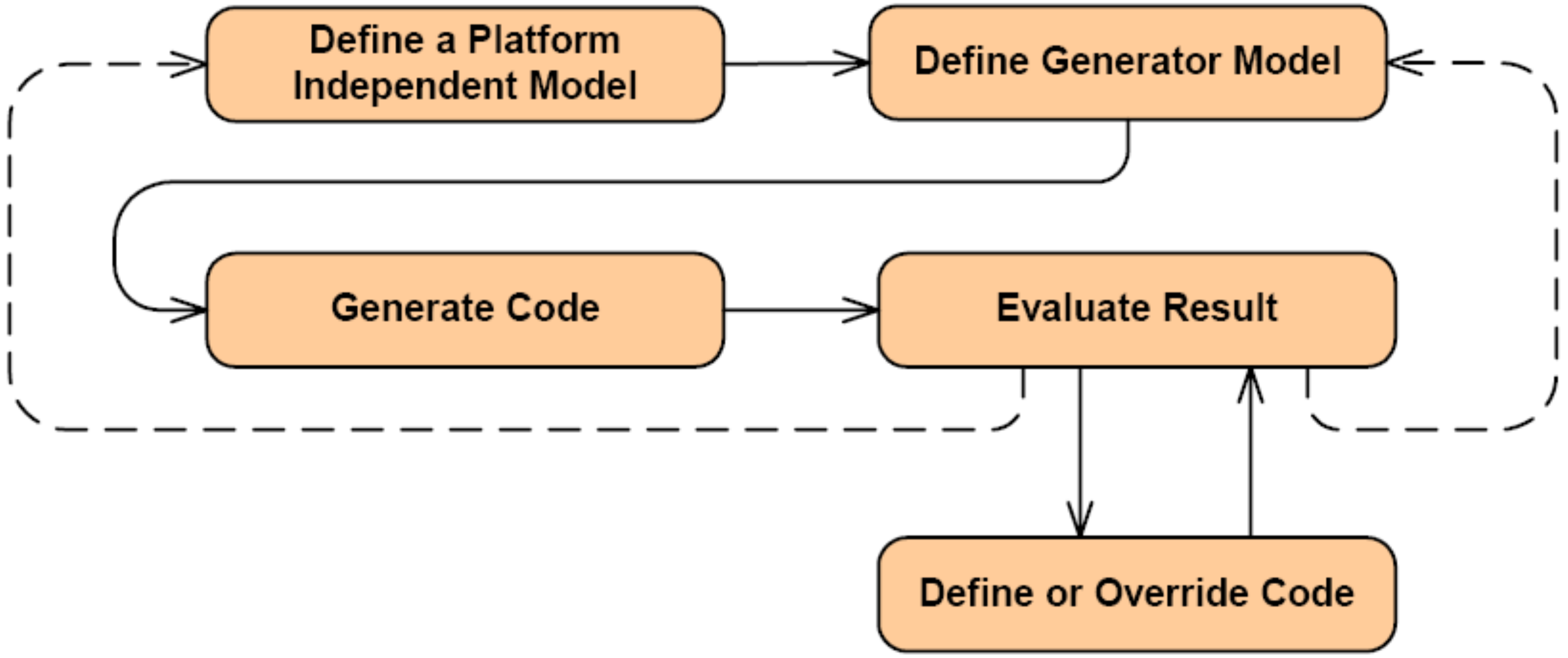


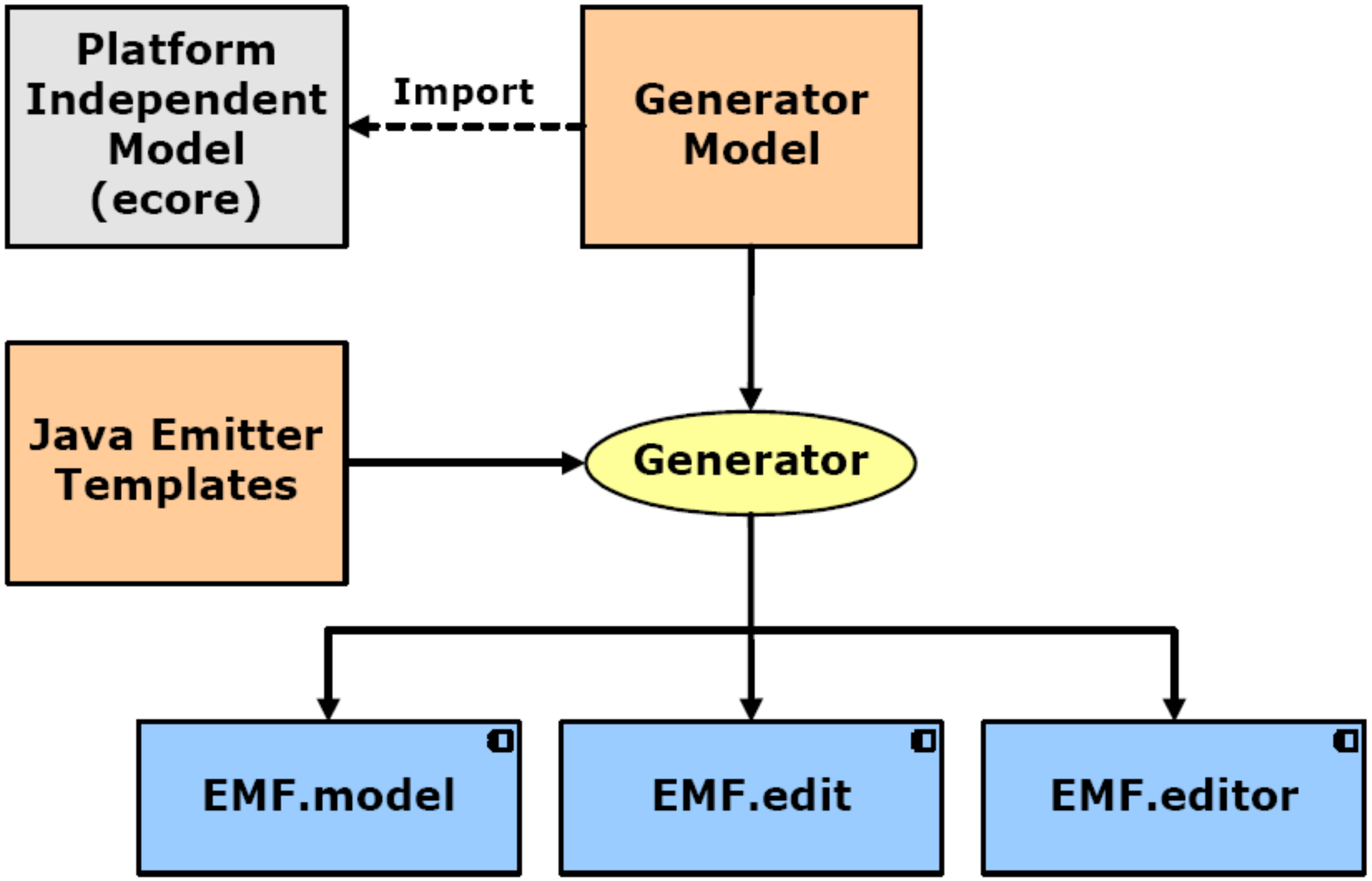
Unidirectional (binary) relation (Association)
- typed
- optional inverse end
- multiplicities

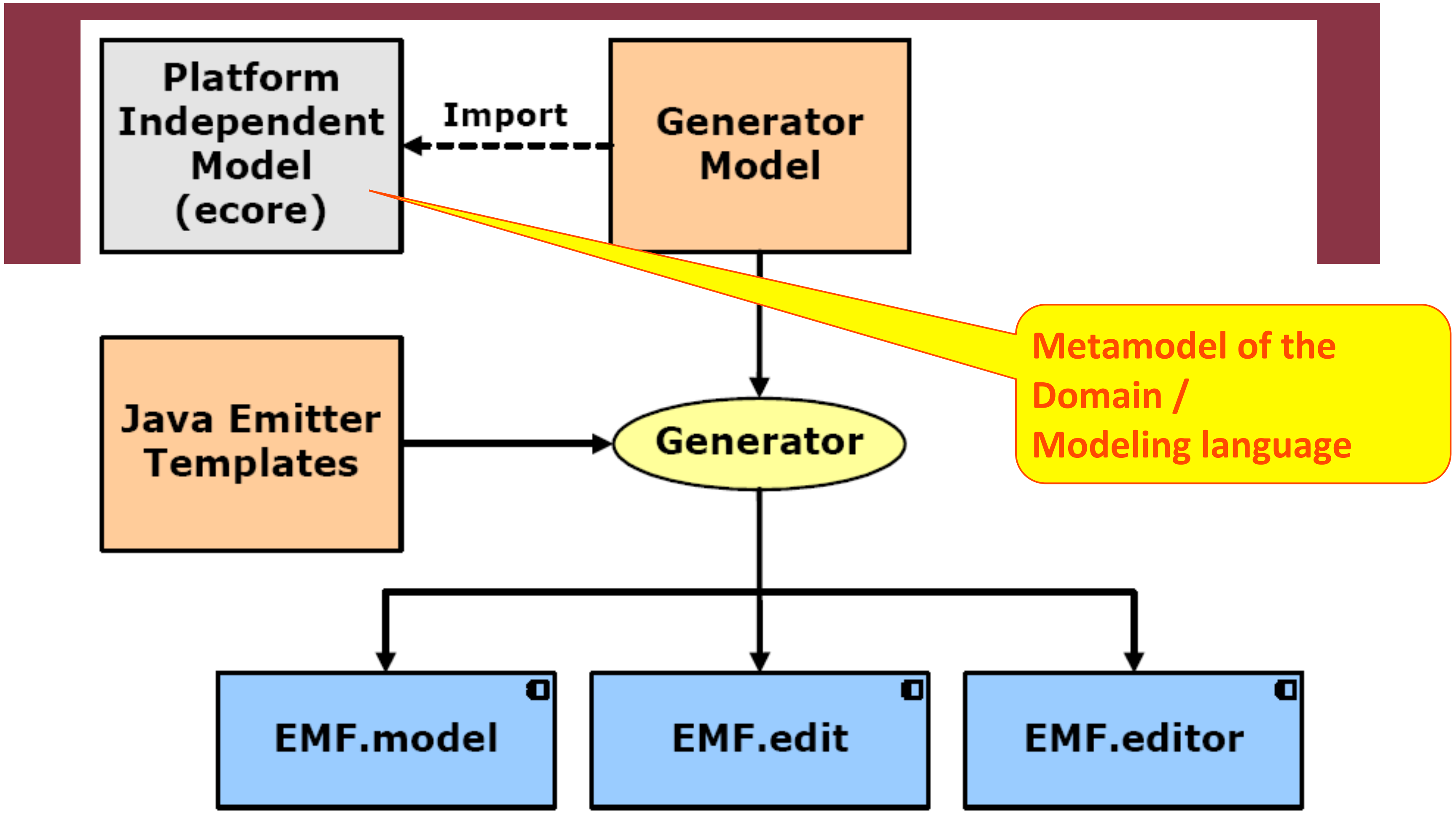

EObject
EModelElement
EFactory
ENamedElement
EAnnotation
EPackage
EClassifier
EEnumLiteral
ETypedElement
EClass
EDataType
EStructuralFeature
EOperation
EParameter
EEnum
EAttribute
EReference

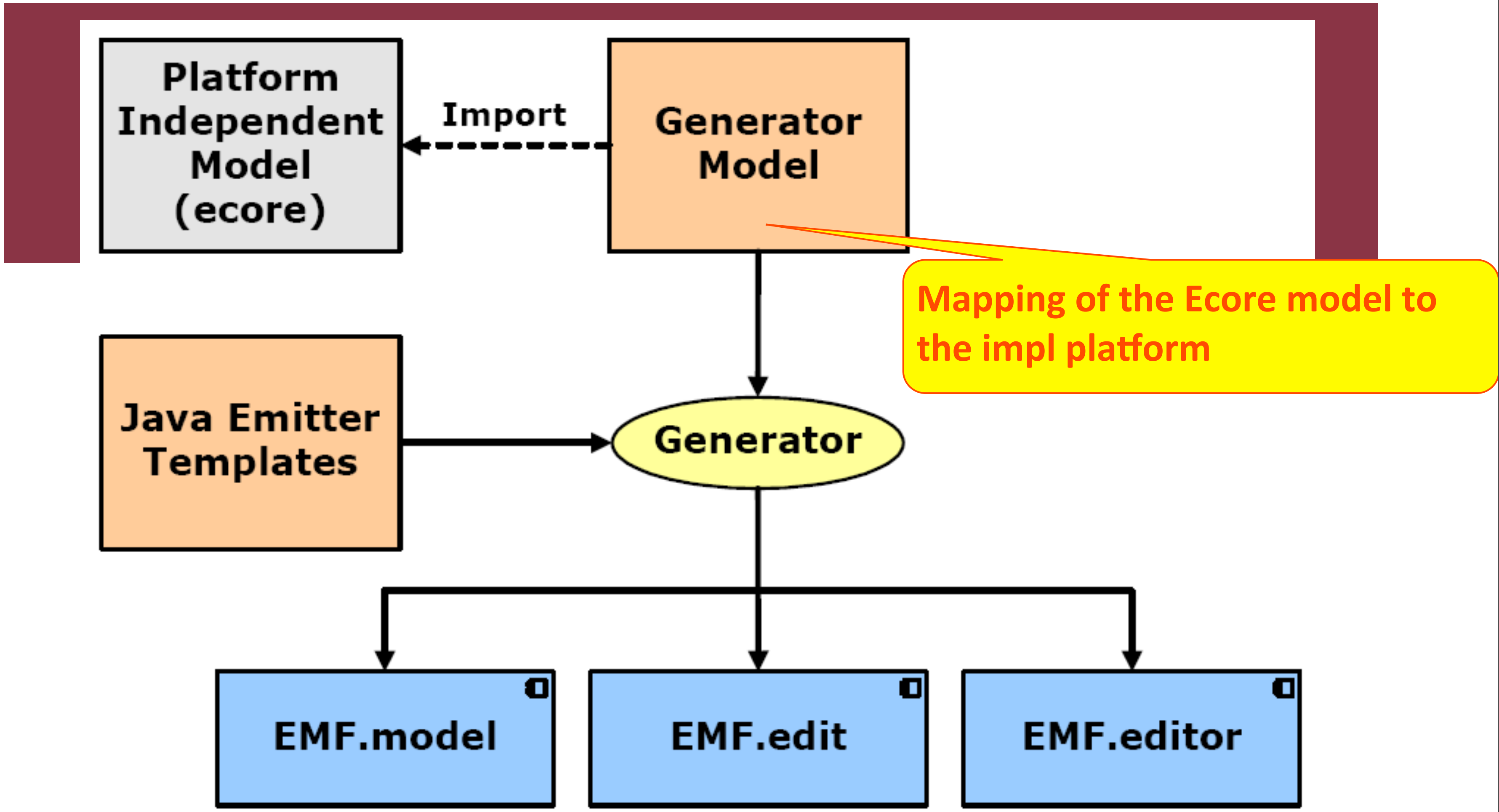

Abstract Class
EObject
EModelElement
EFactory
ENamedElement
EAnnotation
Methods connected to the EClasses
EPackage
EClassifier
EEnumLiteral
ETypedElement
EClass
EDataType
EStructuralFeature
EOperation
EParameter
EEnum
EAttribute
EReference
EObject based Enums
Parameter for the Eoperation

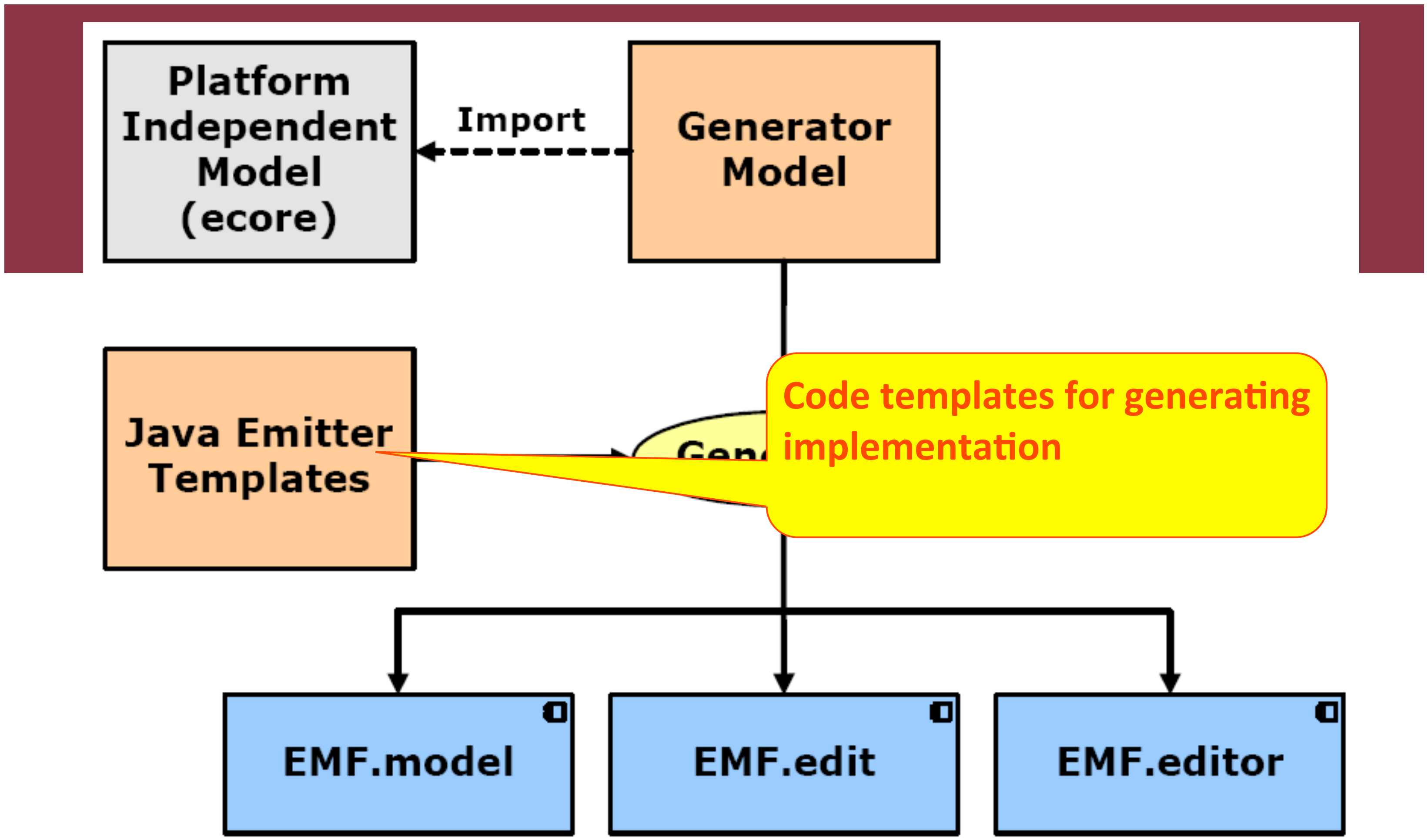
Defining a DSM
The EMF way

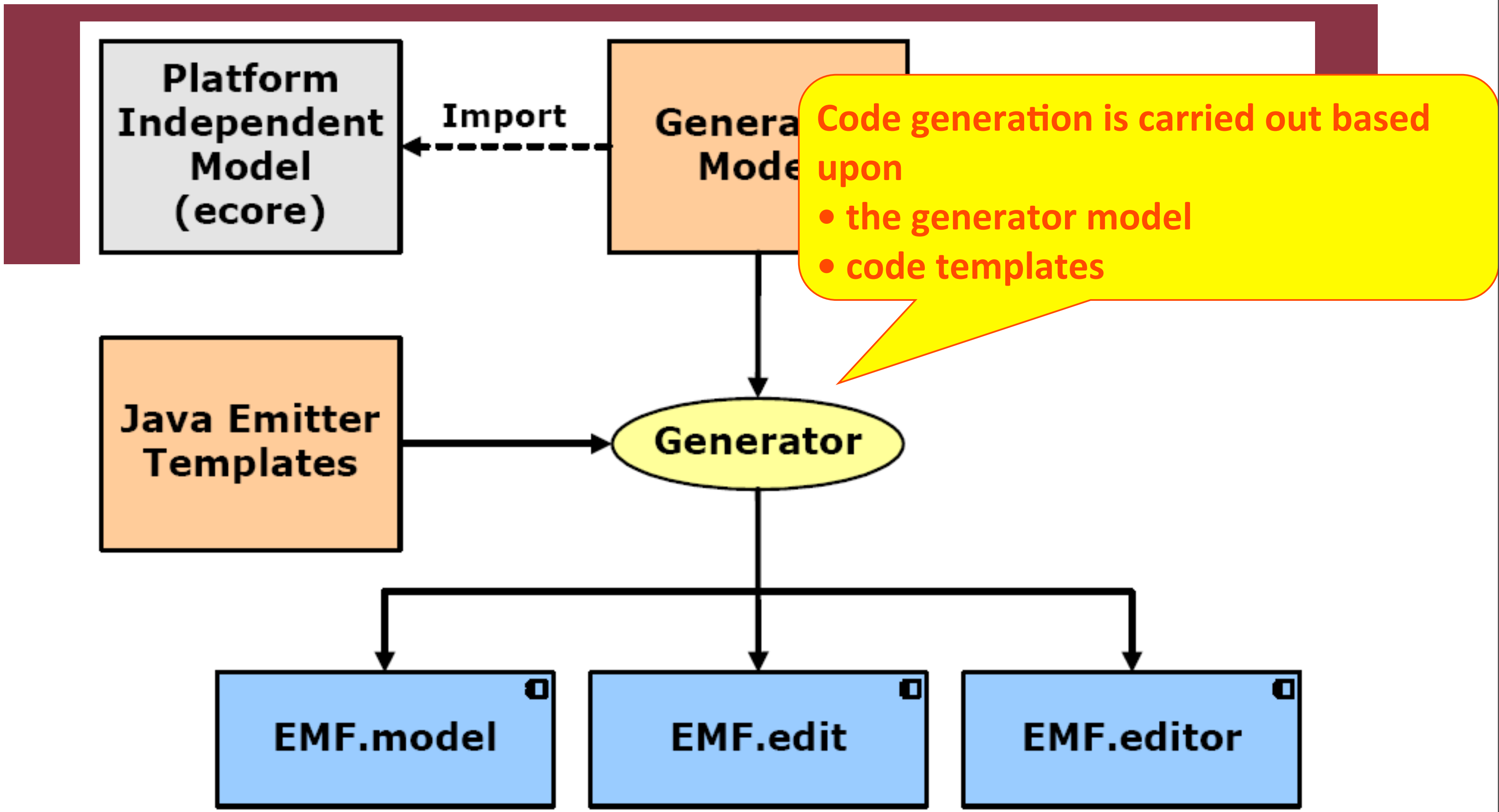

Define a Platform Independent Model
Define Generator Model
Generate Code
Evaluate Result
Define or Override Code


Platform Independent Model (ecore)
Import
Generator Model
Java Emitter Templates
Generator
EMF.model
EMF.edit
EMF.editor


Platform Independent Model (ecore)
Import
Generator Model
Java Emitter Templates
Generator
Metamodel of the Domain / Modeling language
EMF.model
EMF.edit
EMF.editor

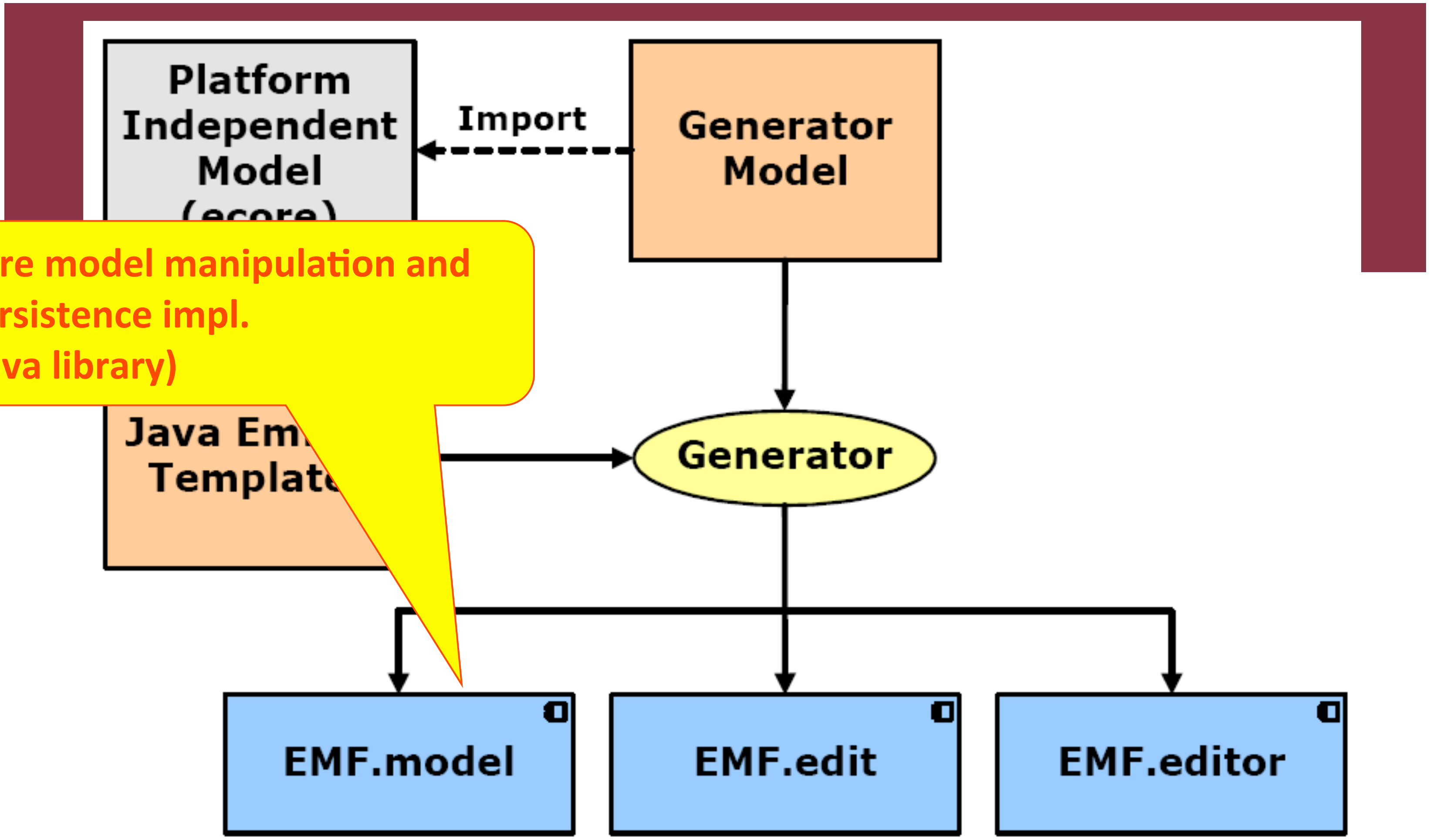

Platform Independent Model (ecore)
Import
Generator Model
Mapping of the Ecore model to the impl platform
Java Emitter Templates
Generator
EMF.model
EMF.edit
EMF.editor


Platform Independent Model (ecore)
Import
Generator Model
Java Emitter Templates
Gen
Code templates for generating implementation
EMF.model
EMF.edit
EMF.editor

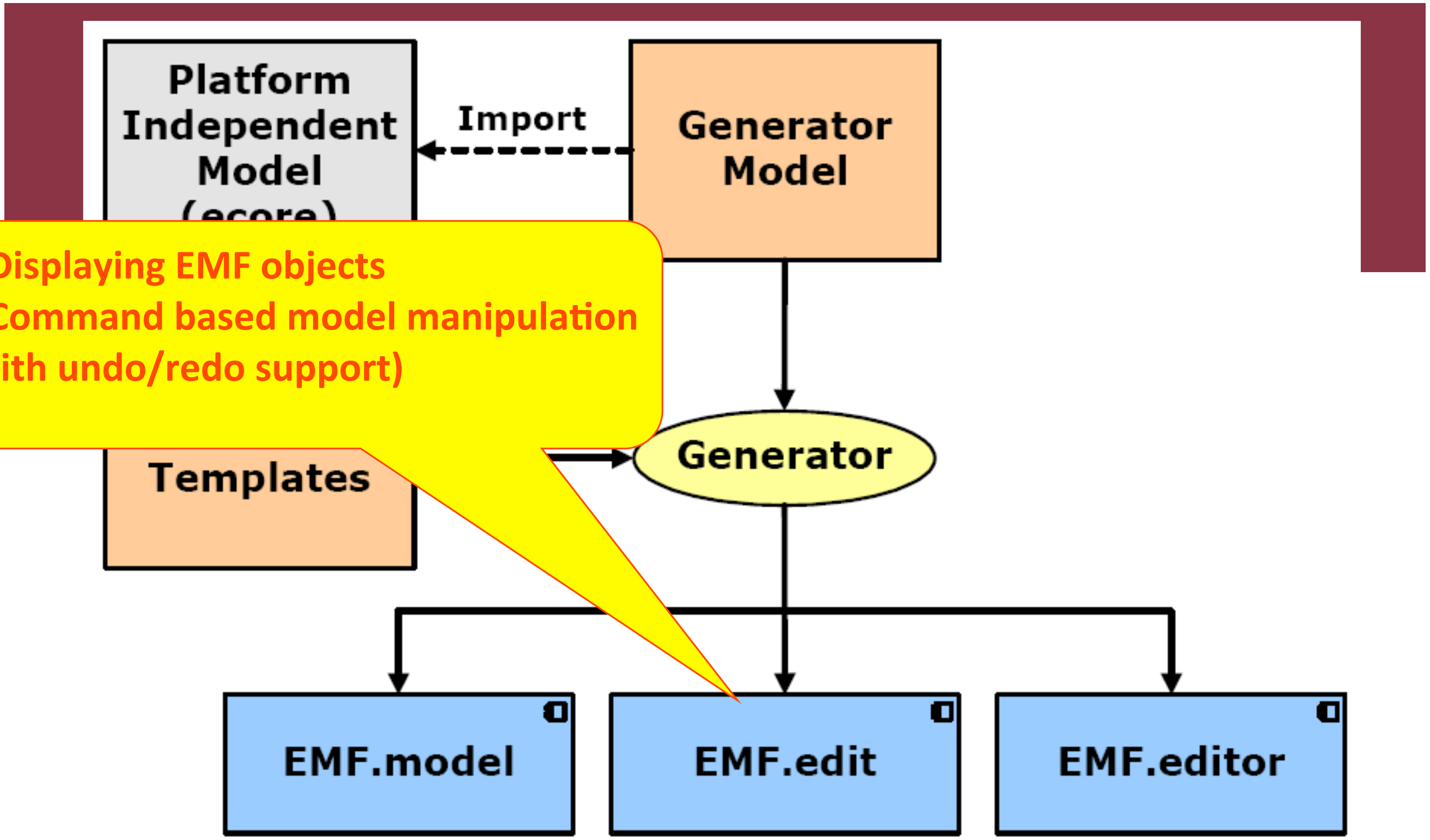


Code generation is carried out based upon

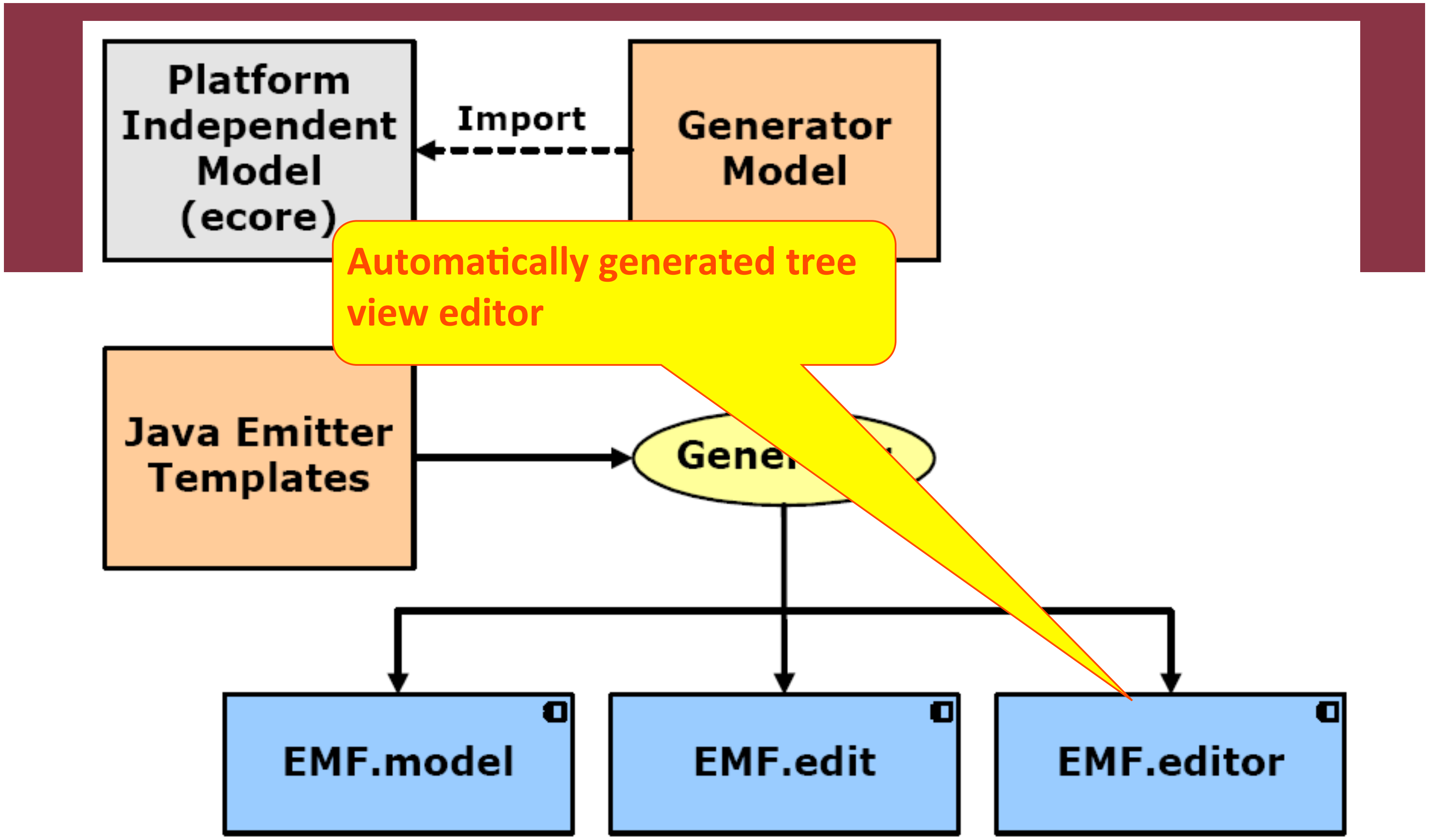
- the generator model
- code templates



Core model manipulation and persistence impl.
(Java library)



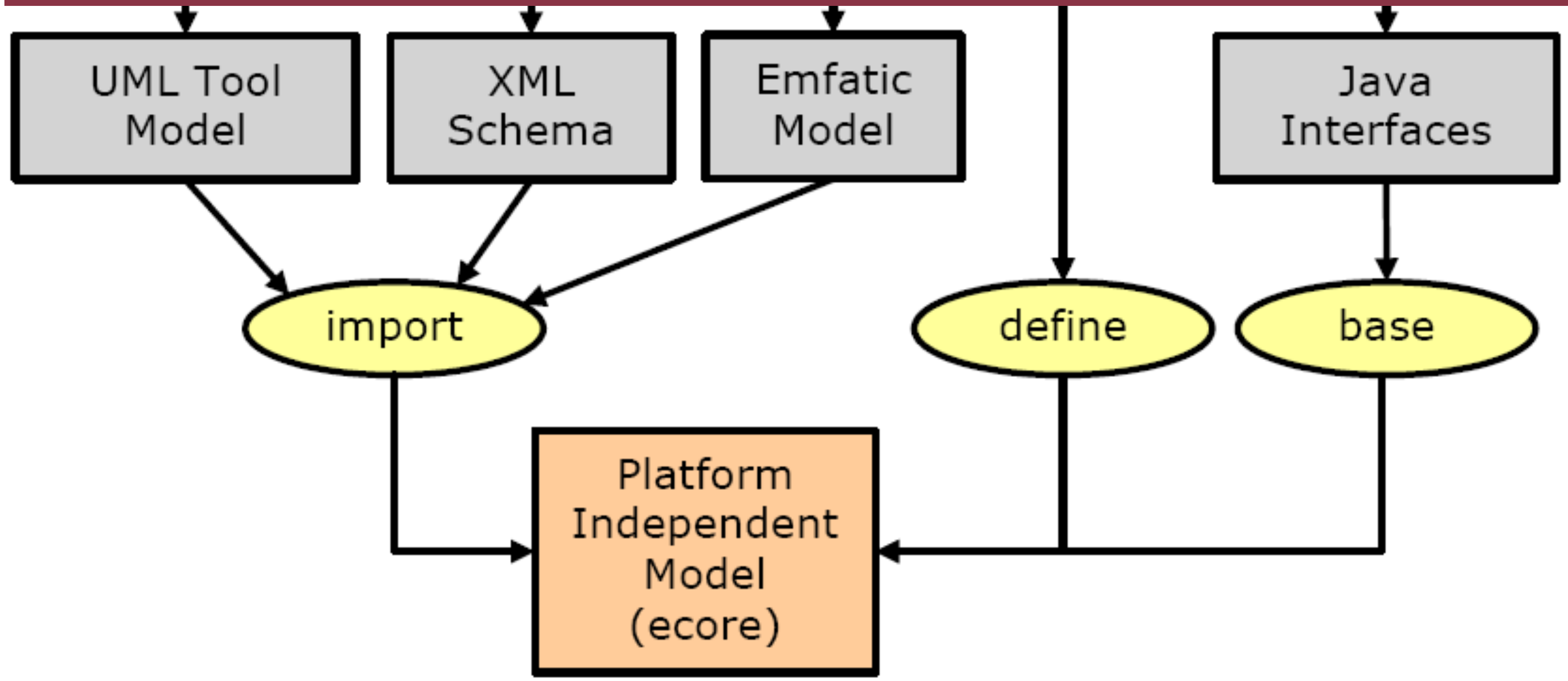
- Displaying EMF objects
- Command based model manipulation (with undo/redo support)



Automatically generated tree view editor

Creating an Ecore model (metamodel)

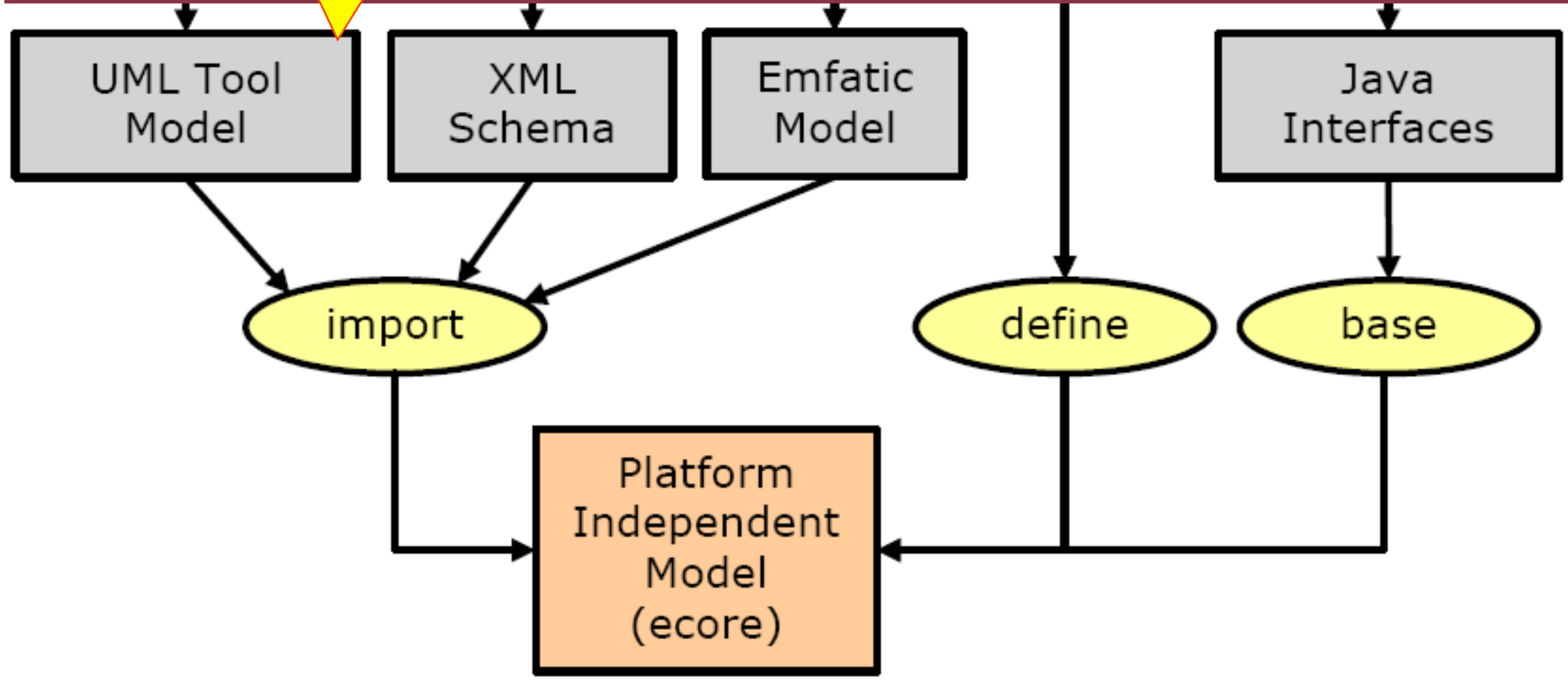
Creation of ECore models



UML class diagram

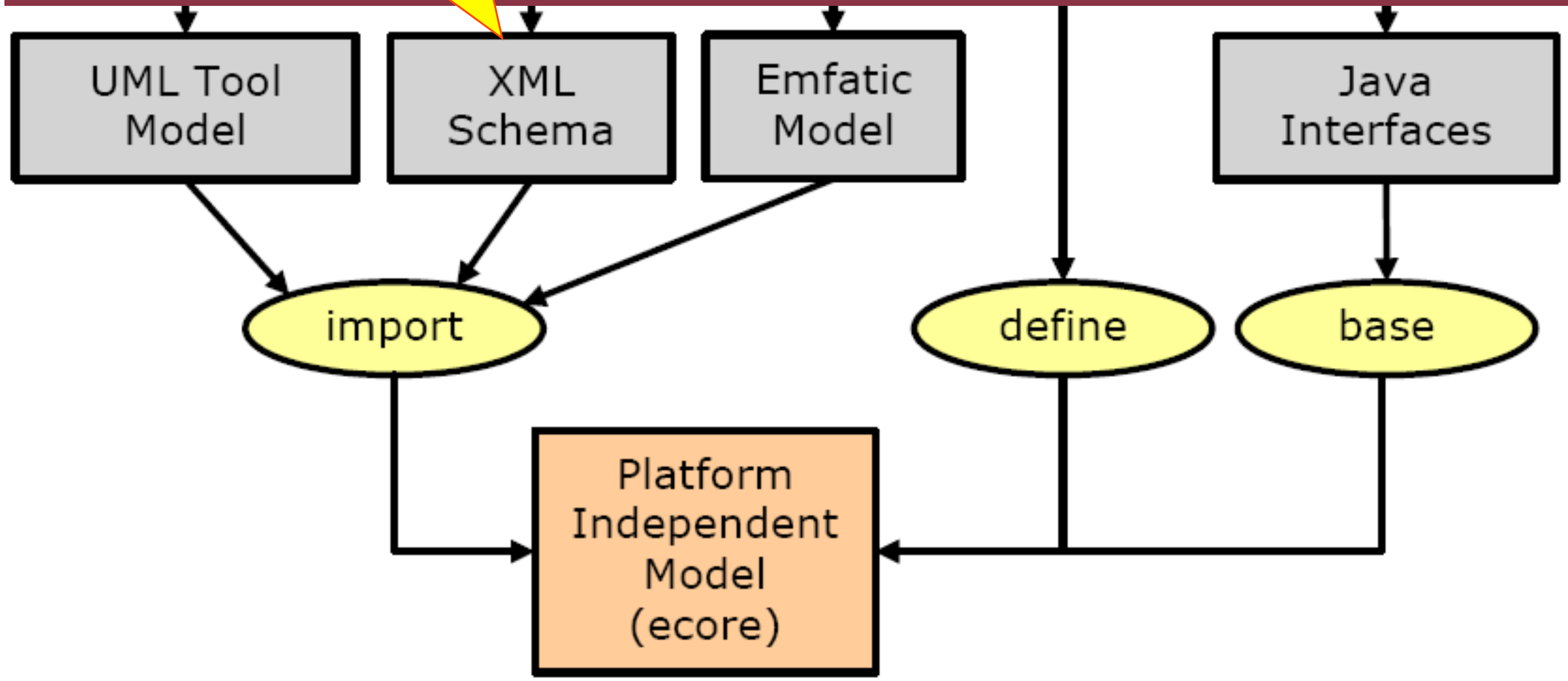
- **Rational Software Architect**
- **EclipseUML (Omondo)**
- **Borland Together Architect**

ation of Ecore models



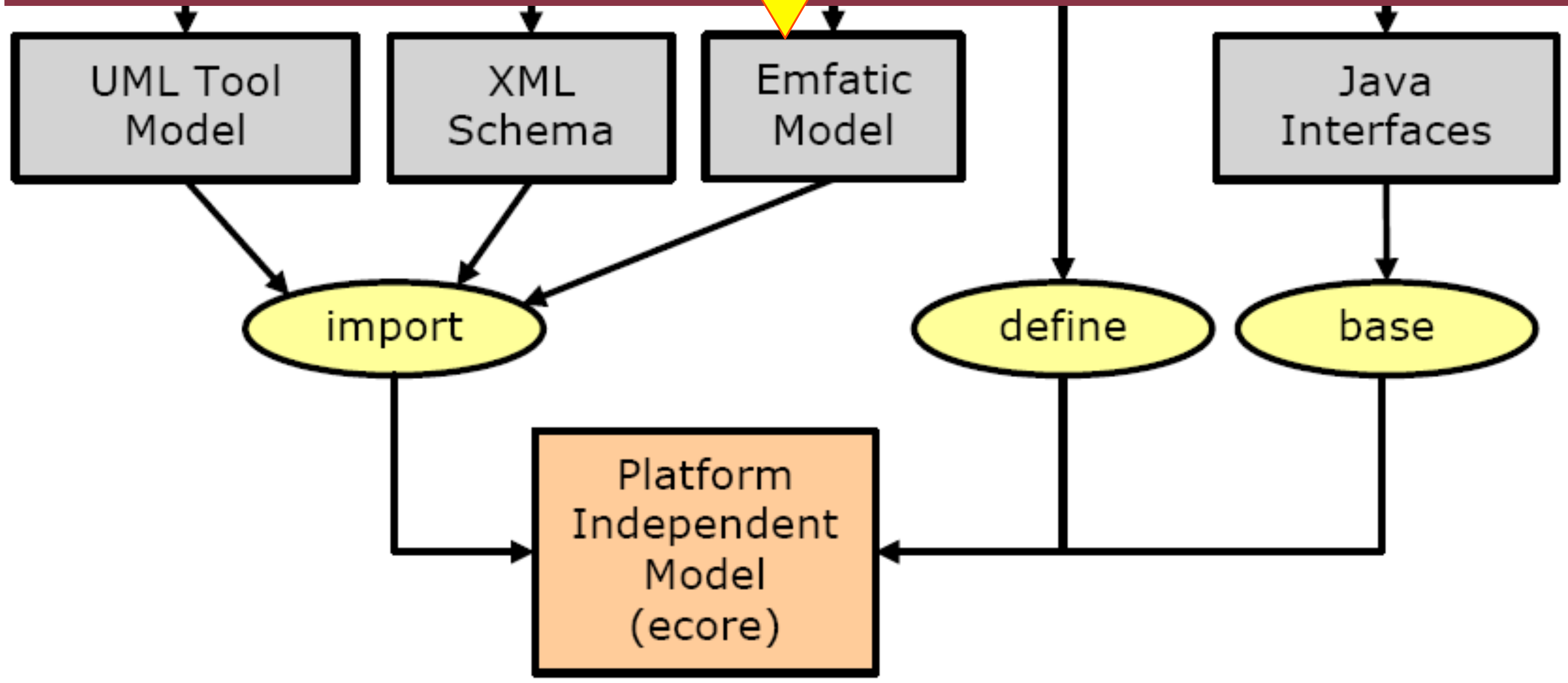
ore models

An XML schema is the metamodel of an XML document



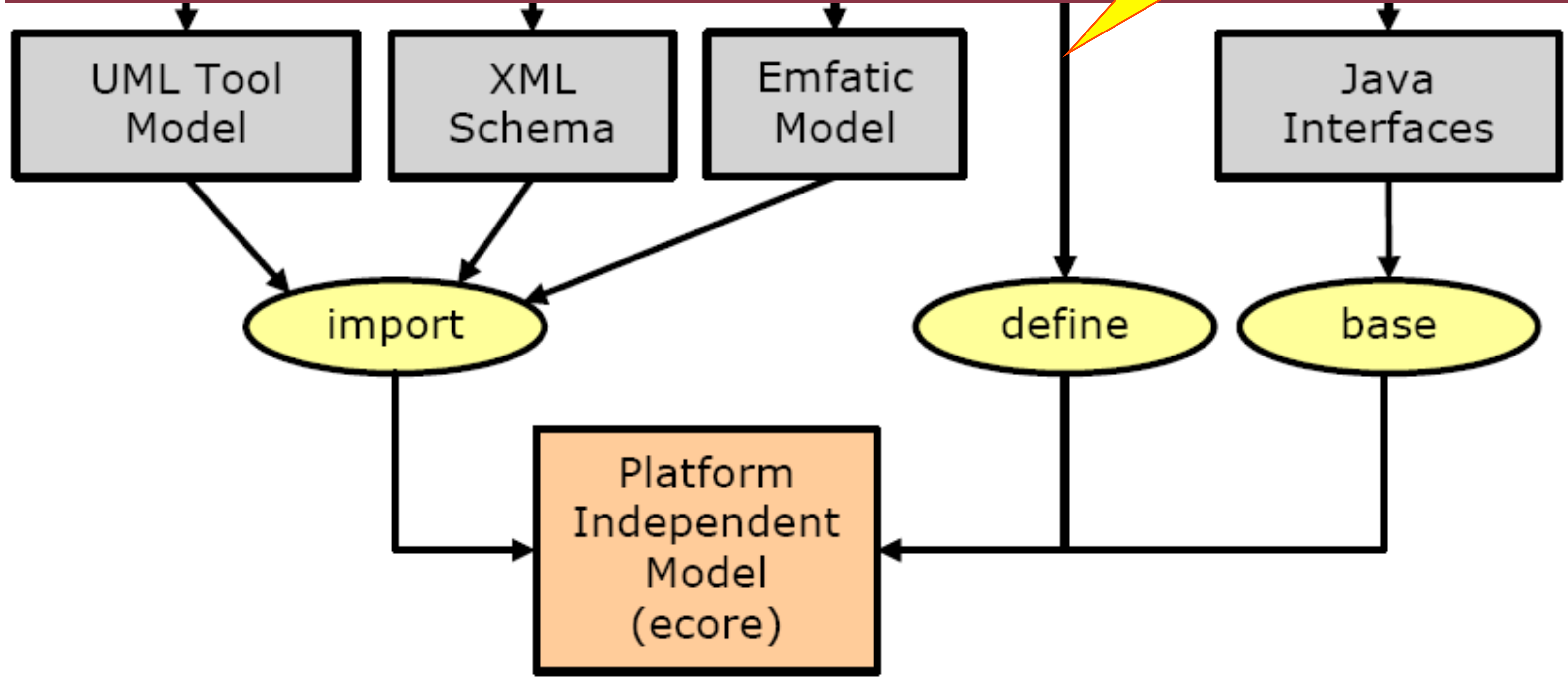
Creation of Ecore models

Simple programming language for defining Ecore models



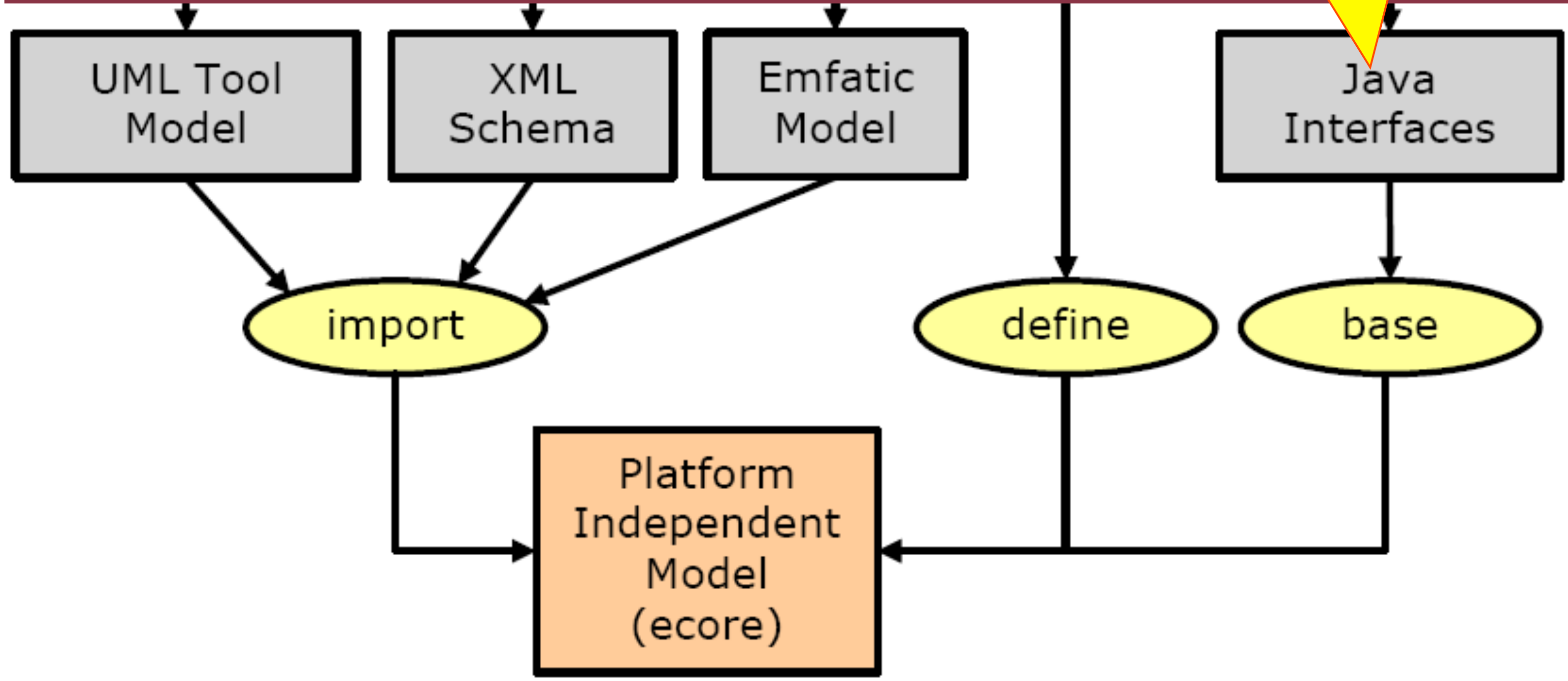
Creation of ECore mod

Direct Ecore defining
(e.g., Ecore tree editor)

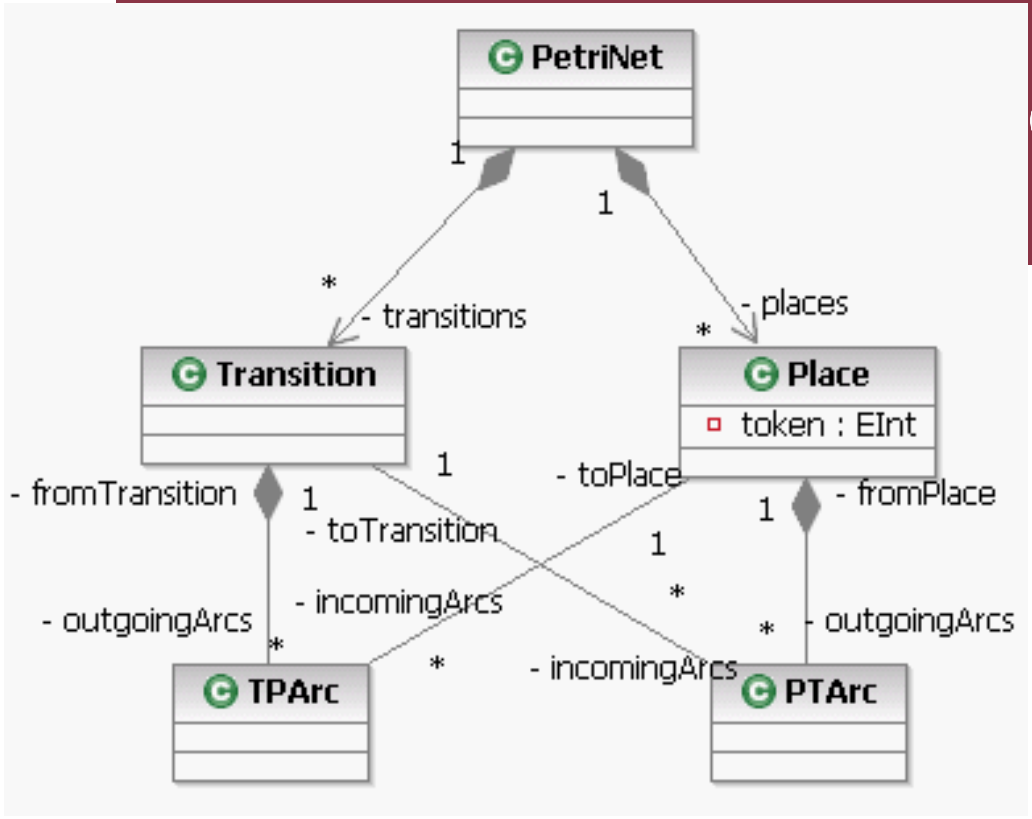


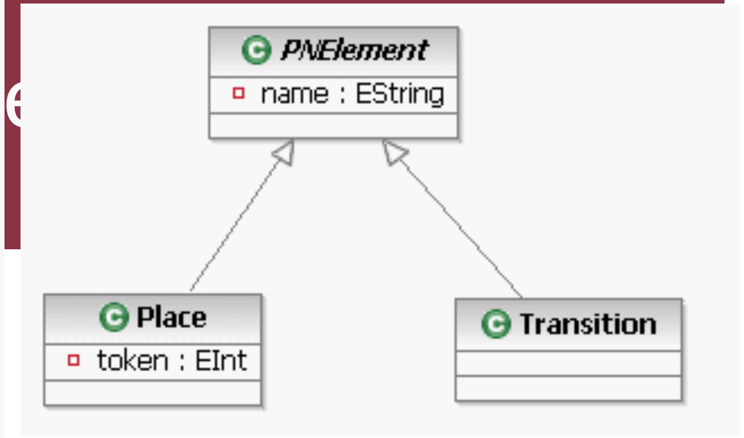
Creation of ECore mo

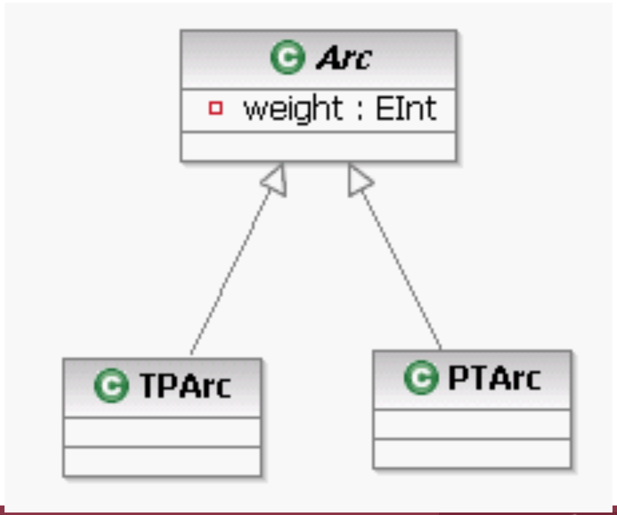
Specially annotated Java interfaces, defines the metamodel



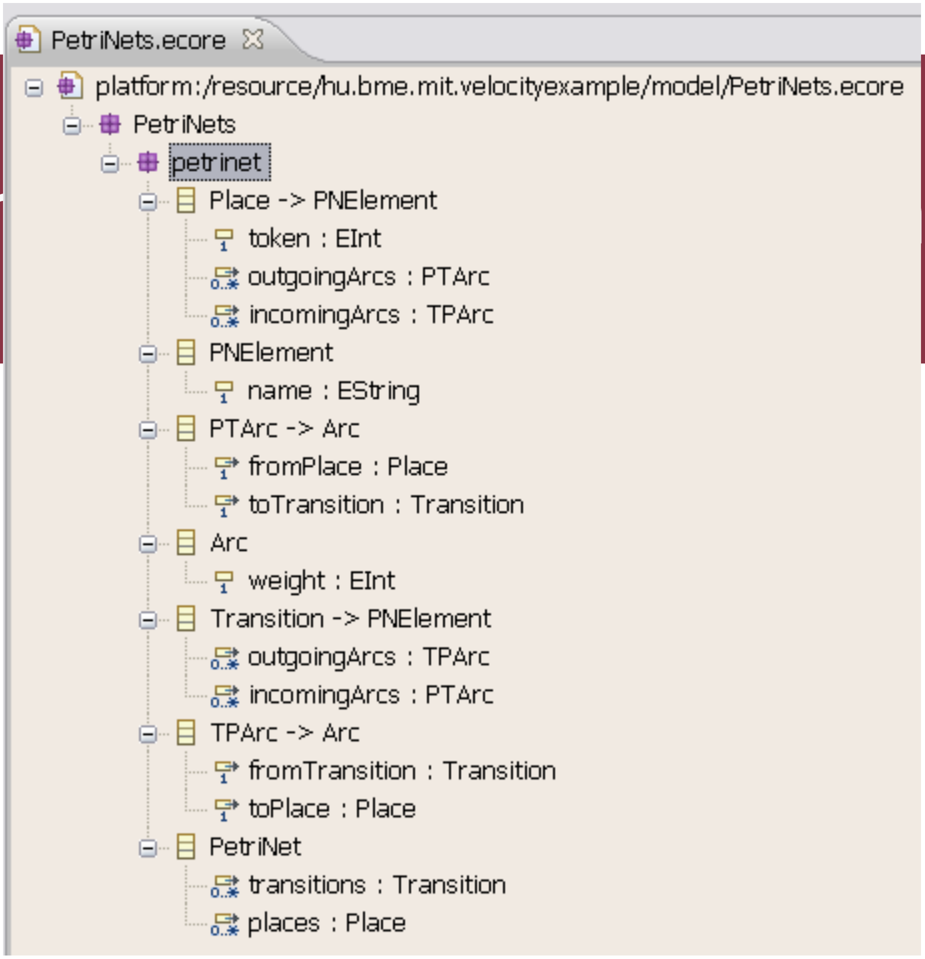
The Petri net Example


PetriNet
1
1
*
- transitions
- places
*
Transition
Place
token : EInt
1
- toPlace
- fromTransition
1
- toTransition
1
- fromPlace
1
*
- incomingArcs
- outgoingArcs
*
*
- outgoingArcs
*
- incomingArcs
TPArc
PTArc


PNElement
name : EString
Place
token : EInt
Transition

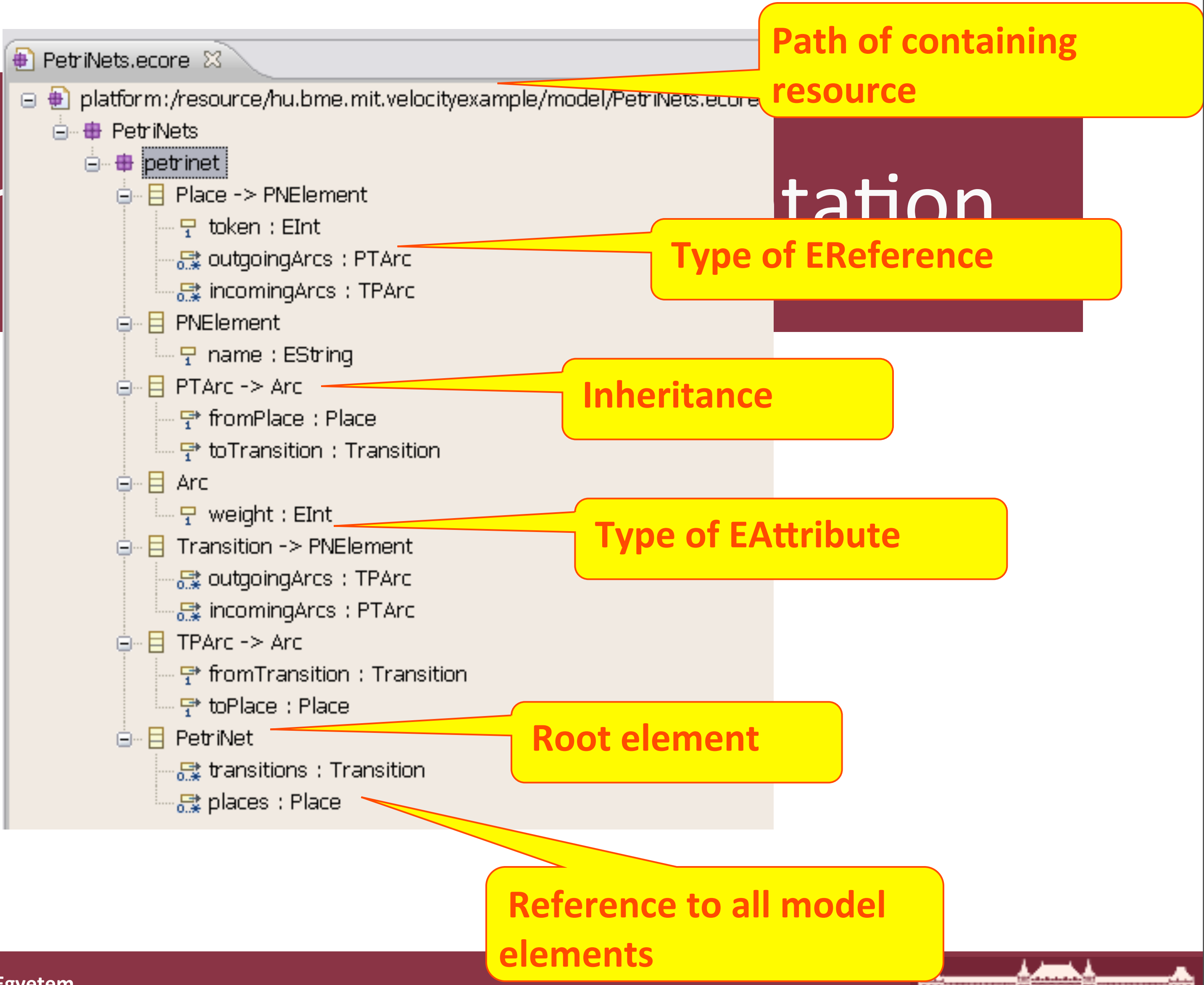

Arc
weight : EInt
TPArc
PTArc

EMF r tation



```
PetriNets.ecore
platform:/resource/hu.bme.mit.velocityexample/model/PetriNets.ecore
  PetriNets
    petrinet
      Place -> PNElement
        token : EInt
        outgoingArcs : PTArc
        incomingArcs : TPArc
      PNElement
        name : EString
      PTArc -> Arc
        fromPlace : Place
        toTransition : Transition
      Arc
        weight : EInt
      Transition -> PNElement
        outgoingArcs : TPArc
        incomingArcs : PTArc
      TPArc -> Arc
        fromTransition : Transition
        toPlace : Place
      PetriNet
        transitions : Transition
        places : Place
```

EMF r… …tation



```
PetriNets.ecore
platform:/resource/hu.bme.mit.velocityexample/model/PetriNets.ecore
  PetriNets
    petrinet
      Place -> PNElement
        token : EInt
        outgoingArcs : PTArc
        incomingArcs : TPArc
      PNElement
        name : EString
      PTArc -> Arc
        fromPlace : Place
        toTransition : Transition
      Arc
        weight : EInt
      Transition -> PNElement
        outgoingArcs : TPArc
        incomingArcs : PTArc
      TPArc -> Arc
        fromTransition : Transition
        toPlace : Place
      PetriNet
        transitions : Transition
        places : Place
```

Class Definition in PetriNet.ecore

```
<eClassifiers xsi:type="ecore:EClass" name="Place"
eSuperTypes="#//petrinet/PNElement">

<eStructuralFeatures xsi:type="ecore:EAttribute"
```

Class

Class Definition in PetriNet.ecore

Attribute

Reference

Containment

<eClassifiers xsi:type="ecore:EClass" name=

eSuper

Multiplicity

/PNElement">

<eStructuralFeatures xsi:type="ecore:EAttribute"

Type

Opposite End

Code generation from Ecore

Code Generation from Ecore (.genmodel)

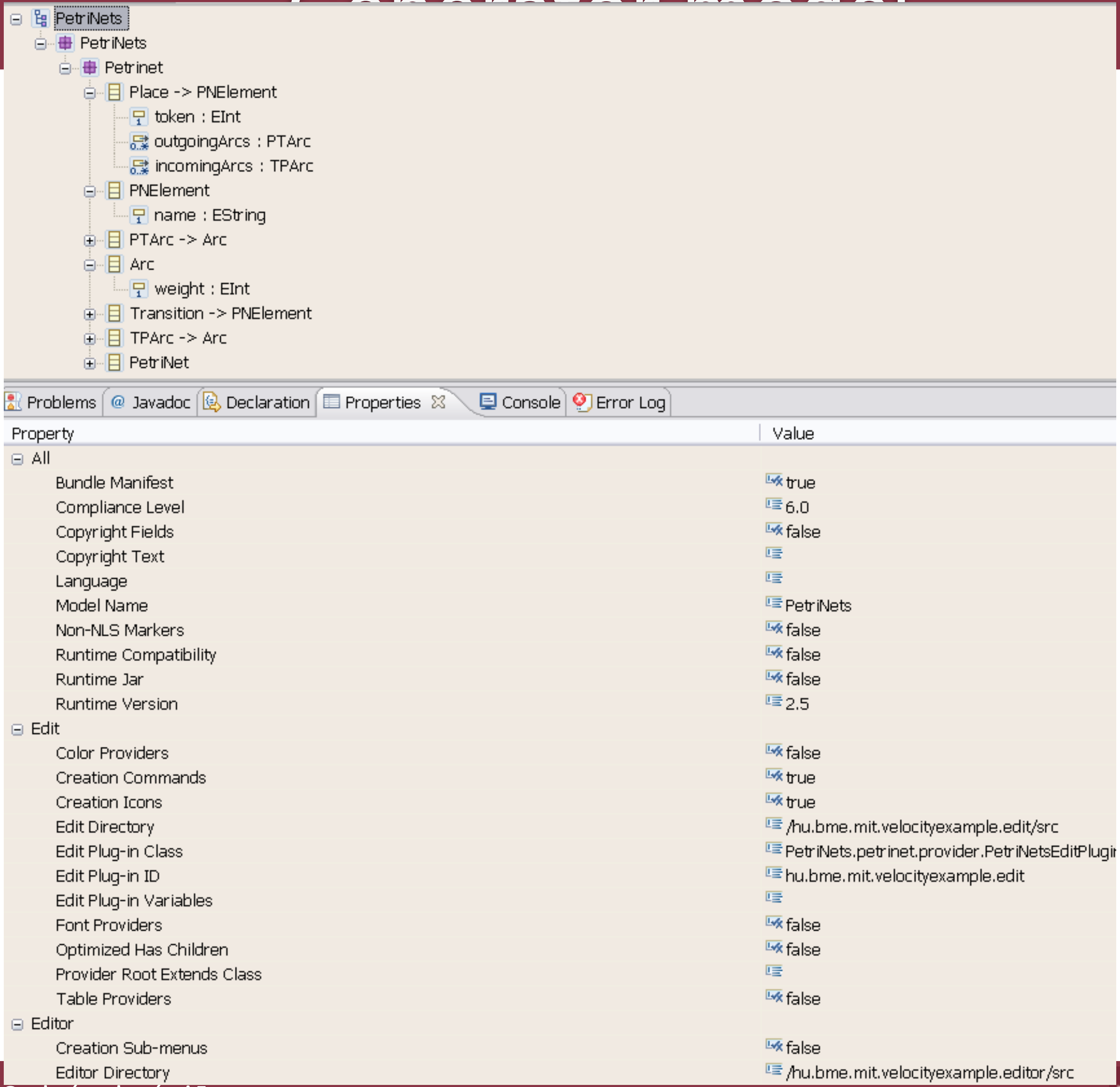
ECore model remain pure and independent

Generator model

Goal:

Specify the attributes of the code

Generator model

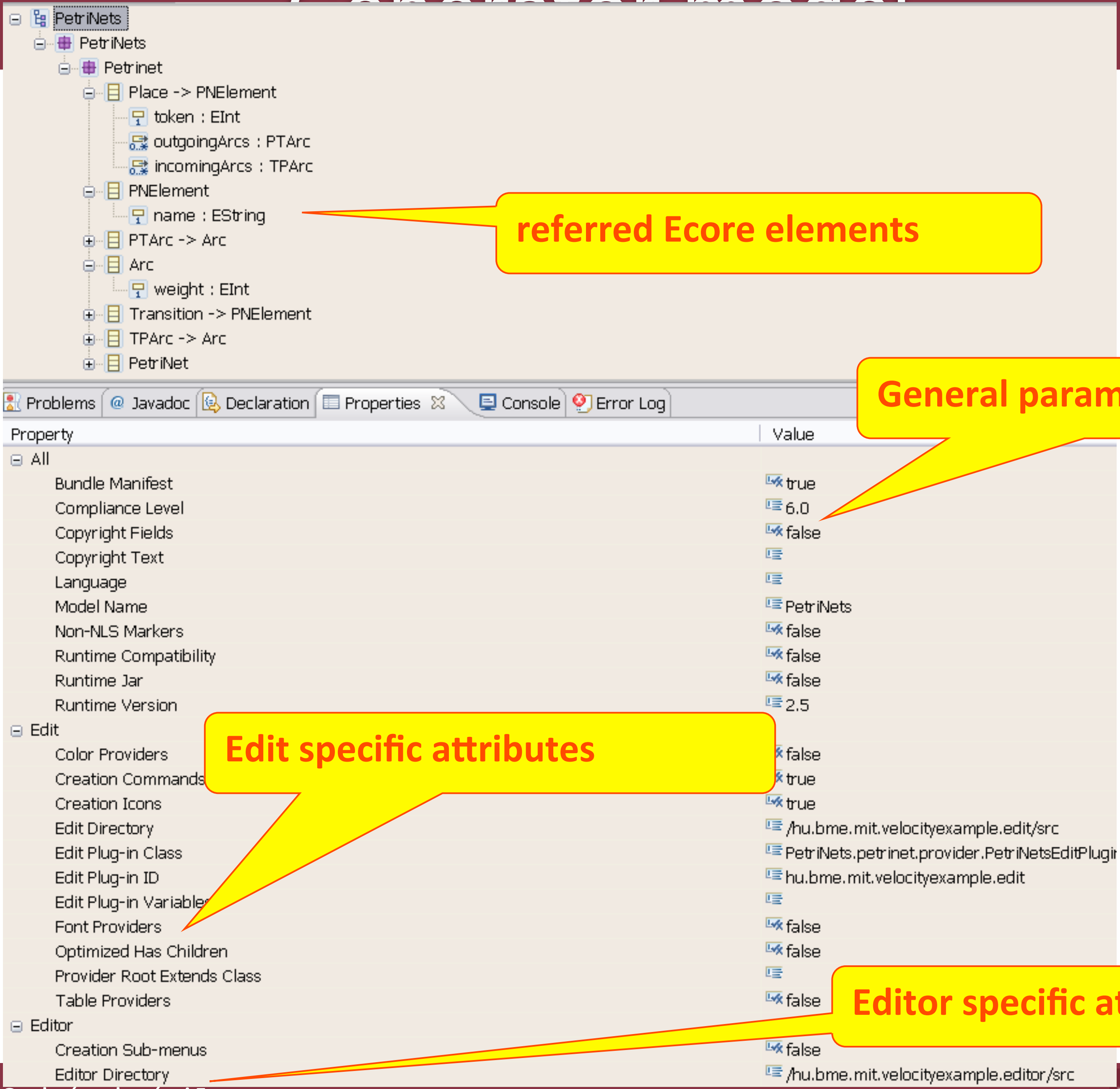


```
⊟ PetriNets
  ⊟ PetriNets
    ⊟ Petrinet
      ⊟ Place -> PNElement
          token : EInt
          outgoingArcs : PTArc
          incomingArcs : TPArc
      ⊟ PNElement
          name : EString
      ⊞ PTArc -> Arc
      ⊟ Arc
          weight : EInt
      ⊞ Transition -> PNElement
      ⊞ TPArc -> Arc
      ⊞ PetriNet
```

Problems | @ Javadoc | Declaration | Properties | Console | Error Log

Property	Value
⊟ All	
Bundle Manifest	true
Compliance Level	6.0
Copyright Fields	false
Copyright Text	
Language	
Model Name	PetriNets
Non-NLS Markers	false
Runtime Compatibility	false
Runtime Jar	false
Runtime Version	2.5
⊟ Edit	
Color Providers	false
Creation Commands	true
Creation Icons	true
Edit Directory	/hu.bme.mit.velocityexample.edit/src
Edit Plug-in Class	PetriNets.petrinet.provider.PetriNetsEditPlugi
Edit Plug-in ID	hu.bme.mit.velocityexample.edit
Edit Plug-in Variables	
Font Providers	false
Optimized Has Children	false
Provider Root Extends Class	
Table Providers	false
⊟ Editor	
Creation Sub-menus	false
Editor Directory	/hu.bme.mit.velocityexample.editor/src

Generator model



```
⊟ PetriNets
  ⊟ PetriNets
    ⊟ Petrinet
      ⊟ Place -> PNElement
          token : EInt
          outgoingArcs : PTArc
          incomingArcs : TPArc
      ⊟ PNElement
          name : EString
      ⊞ PTArc -> Arc
      ⊟ Arc
          weight : EInt
      ⊞ Transition -> PNElement
      ⊞ TPArc -> Arc
      ⊞ PetriNet
```

Problems | Javadoc | Declaration | Properties | Console | Error Log

Property	Value
All	
Bundle Manifest	true
Compliance Level	6.0
Copyright Fields	false
Copyright Text	
Language	
Model Name	PetriNets
Non-NLS Markers	false
Runtime Compatibility	false
Runtime Jar	false
Runtime Version	2.5
Edit	
Color Providers	false
Creation Commands	true
Creation Icons	true
Edit Directory	/hu.bme.mit.velocityexample.edit/src
Edit Plug-in Class	PetriNets.petrinet.provider.PetriNetsEditPlugi
Edit Plug-in ID	hu.bme.mit.velocityexample.edit
Edit Plug-in Variables	
Font Providers	false
Optimized Has Children	false
Provider Root Extends Class	
Table Providers	false
Editor	
Creation Sub-menus	false
Editor Directory	/hu.bme.mit.velocityexample.editor/src

referred Ecore elements

General parameters

Edit specific attributes

Editor specific attributes

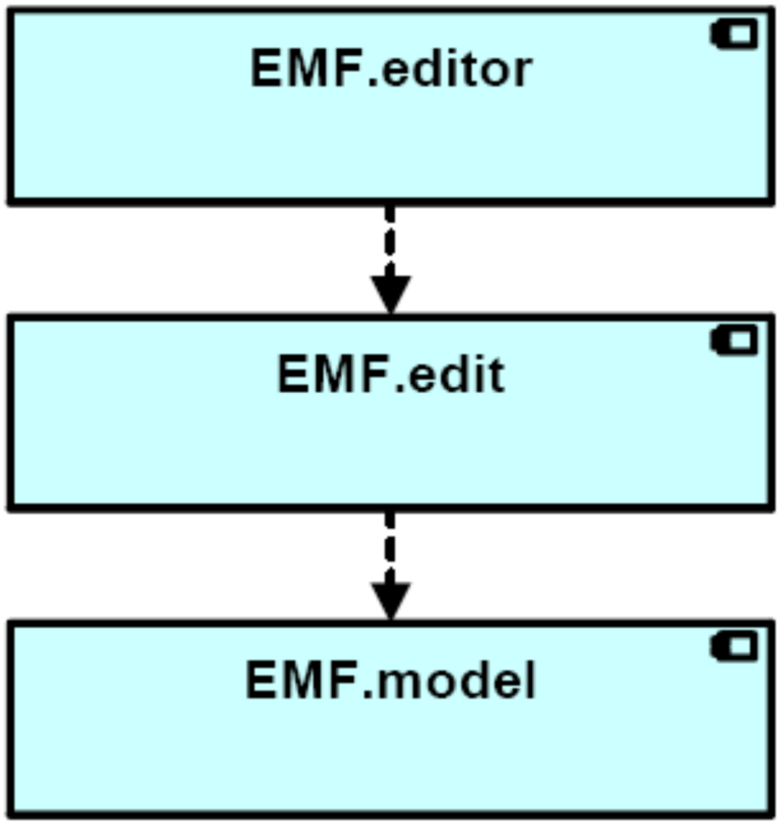
Generated EMF components

Automation in EMF

Model manipulation

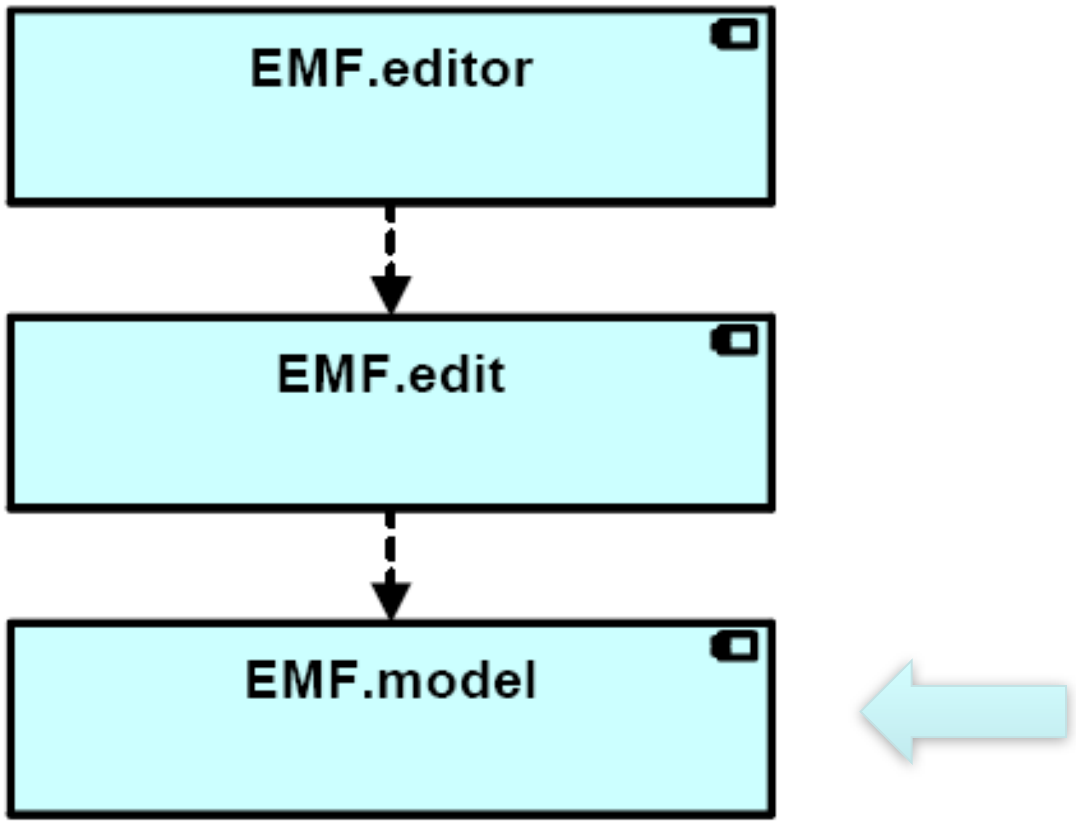
Bidirectional references (also m..n), Containments

Generated EMF components



- 3. Tree Editor
- 2. Model Manipluation
- 1. Model Persistency

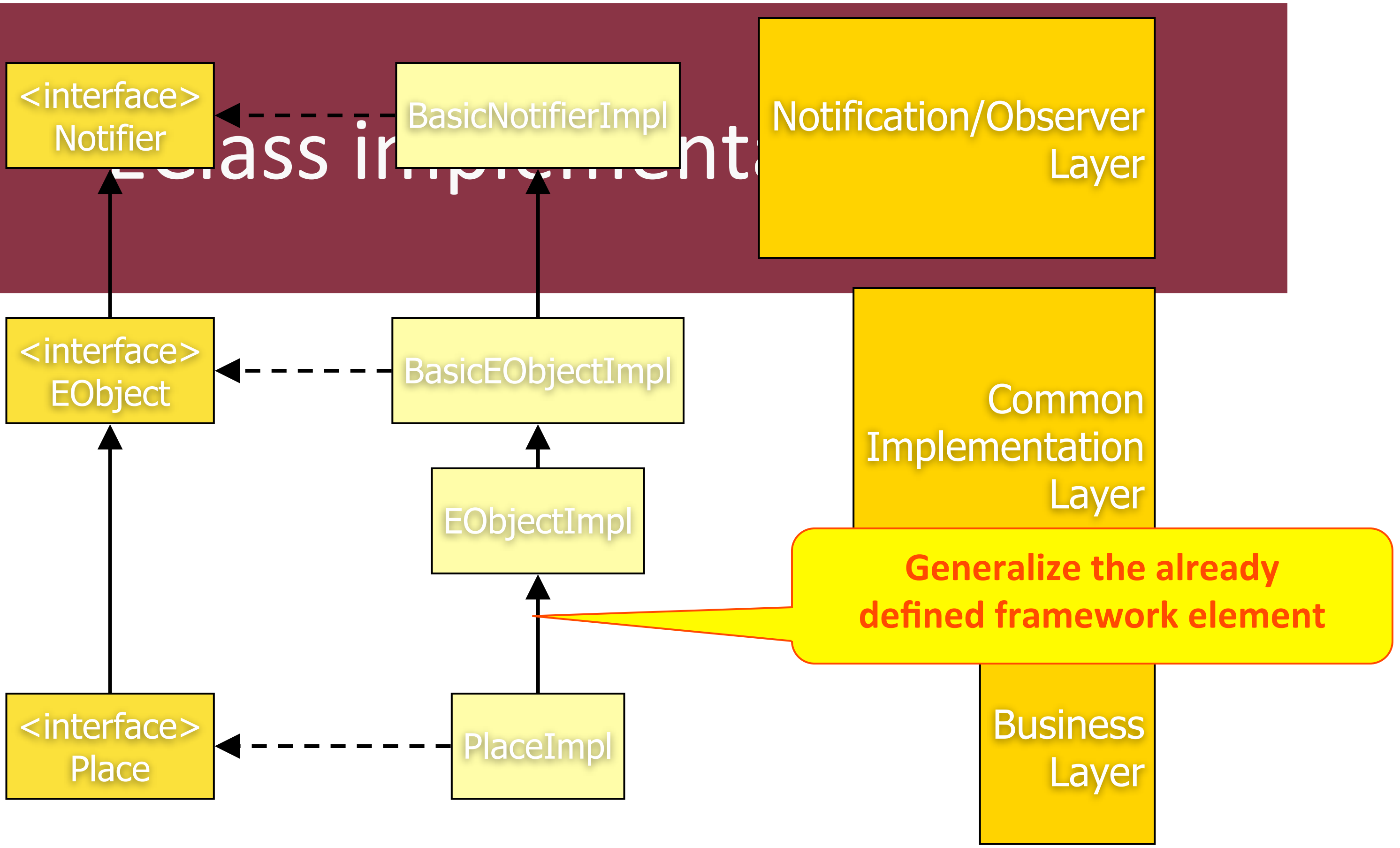
Generated EMF components



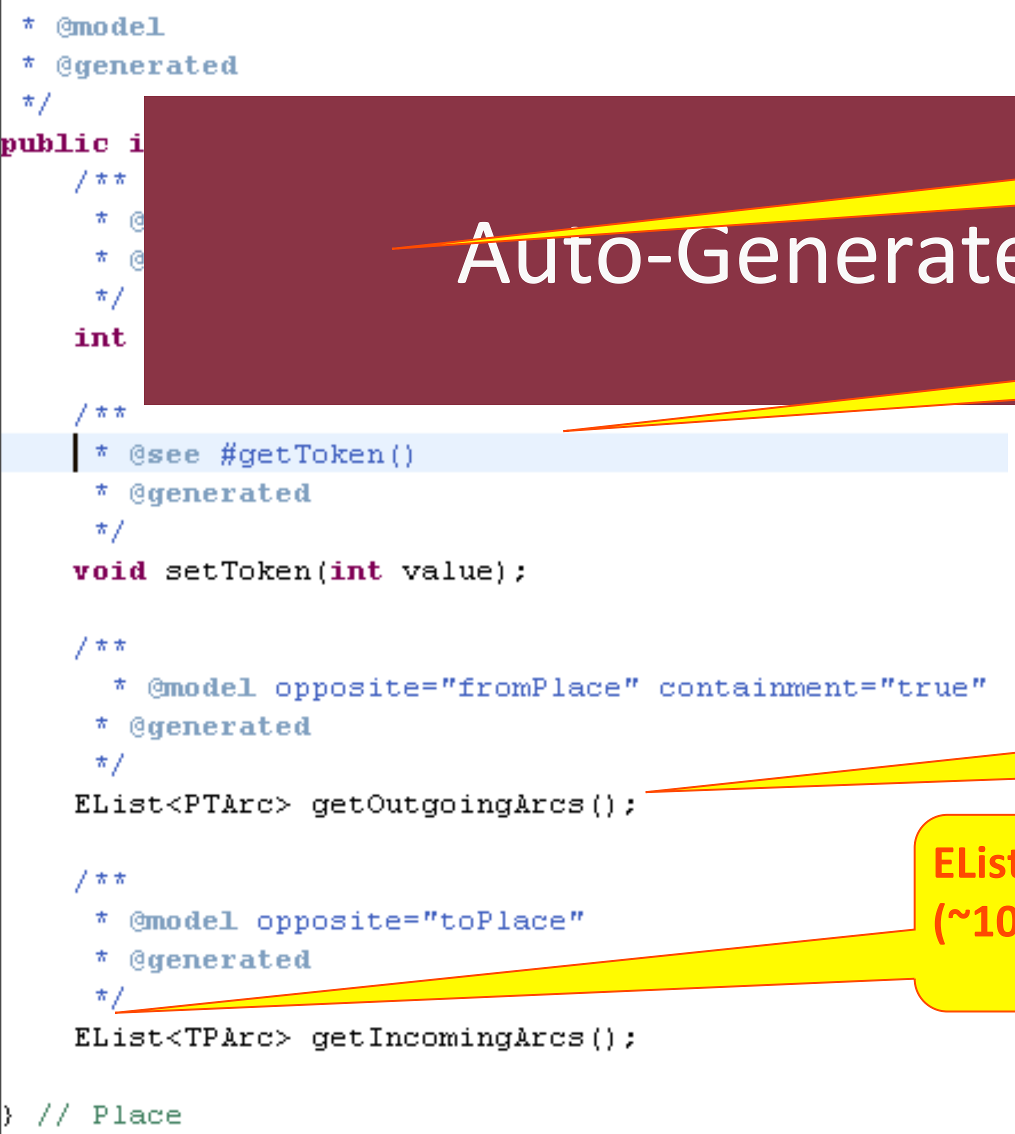
EMF.model

Optimized persistency handling

Fully featured Java code of the ECore


<interface> Notifier
BasicNotifierImpl
Notification/Observer Layer
<interface> EObject
BasicEObjectImpl
Common Implementation Layer
EObjectImpl
Generalize the already defined framework element
<interface> Place
PlaceImpl
Business Layer

Auto-Generated Interface



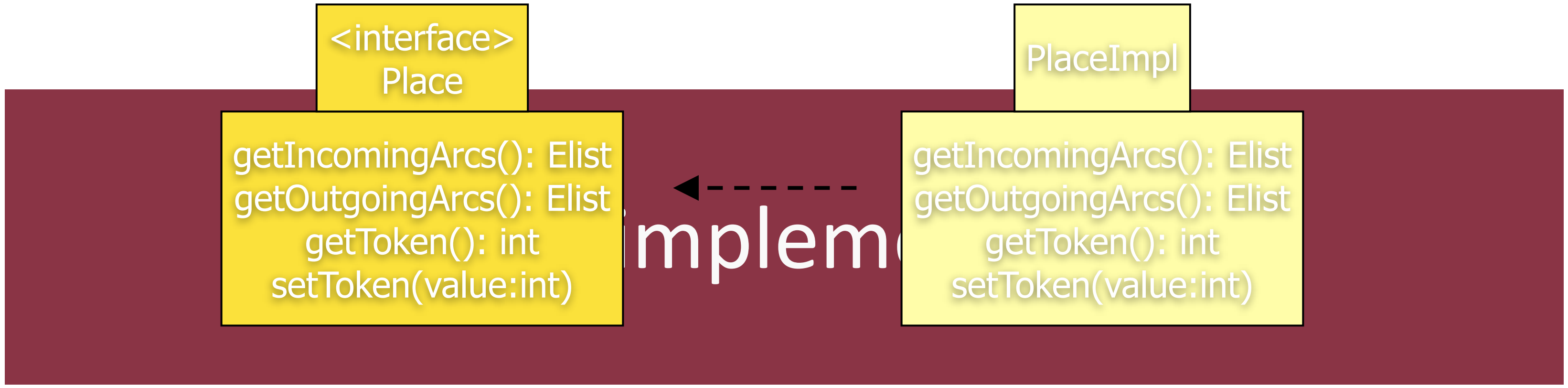
```
 * @model
 * @generated
 */
public i
    /**
     * @
     * @
     */
    int

    /**
     * @see #getToken()
     * @generated
     */
    void setToken(int value);

    /**
      * @model opposite="fromPlace" containment="true"
     * @generated
     */
    EList<PTArc> getOutgoingArcs();

    /**
     * @model opposite="toPlace"
     * @generated
     */
    EList<TPArc> getIncomingArcs();

} // Place
```

EMF specific „annotations”

Getters/Setters for attributes

Only getter when Multiplicity > 1

EList, EMF list interface (~10 implementations

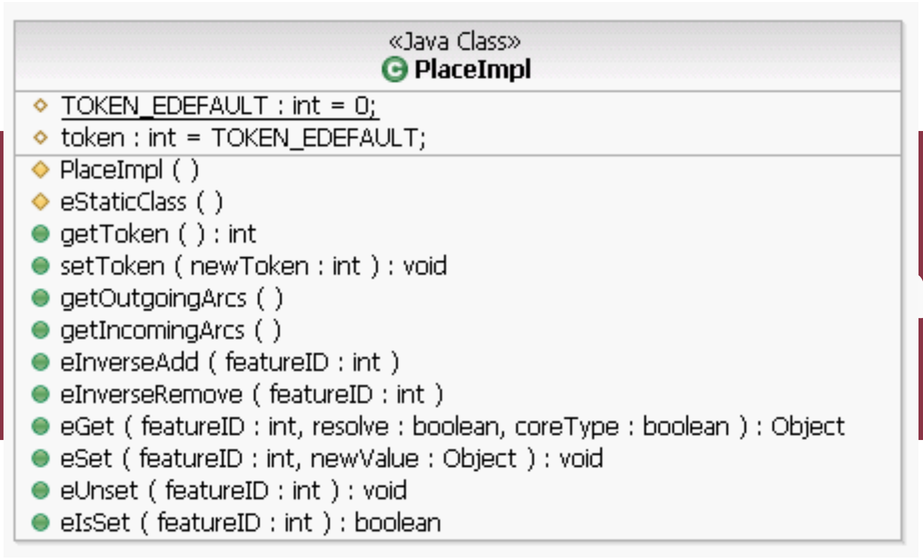


EAttribute -> get/set methods (multiplicity <= 1)

EReference:

„many” -> get

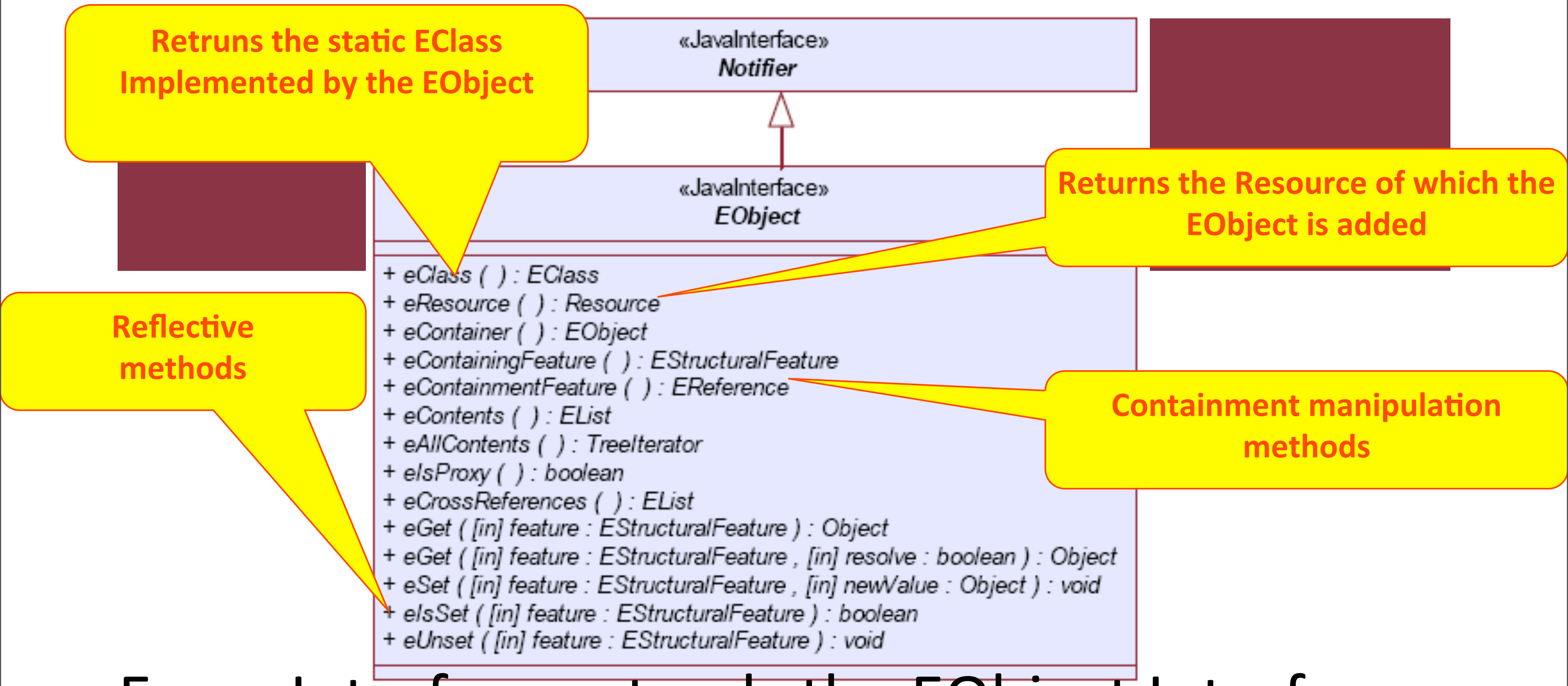
„one” -> get / set


```
«Java Class»
PlaceImpl
TOKEN_EDEFAULT : int = 0;
token : int = TOKEN_EDEFAULT;
PlaceImpl ( )
eStaticClass ( )
getToken ( ) : int
setToken ( newToken : int ) : void
getOutgoingArcs ( )
getIncomingArcs ( )
eInverseAdd ( featureID : int )
eInverseRemove ( featureID : int )
eGet ( featureID : int, resolve : boolean, coreType : boolean ) : Object
eSet ( featureID : int, newValue : Object ) : void
eUnset ( featureID : int ) : void
eIsSet ( featureID : int ) : boolean
```

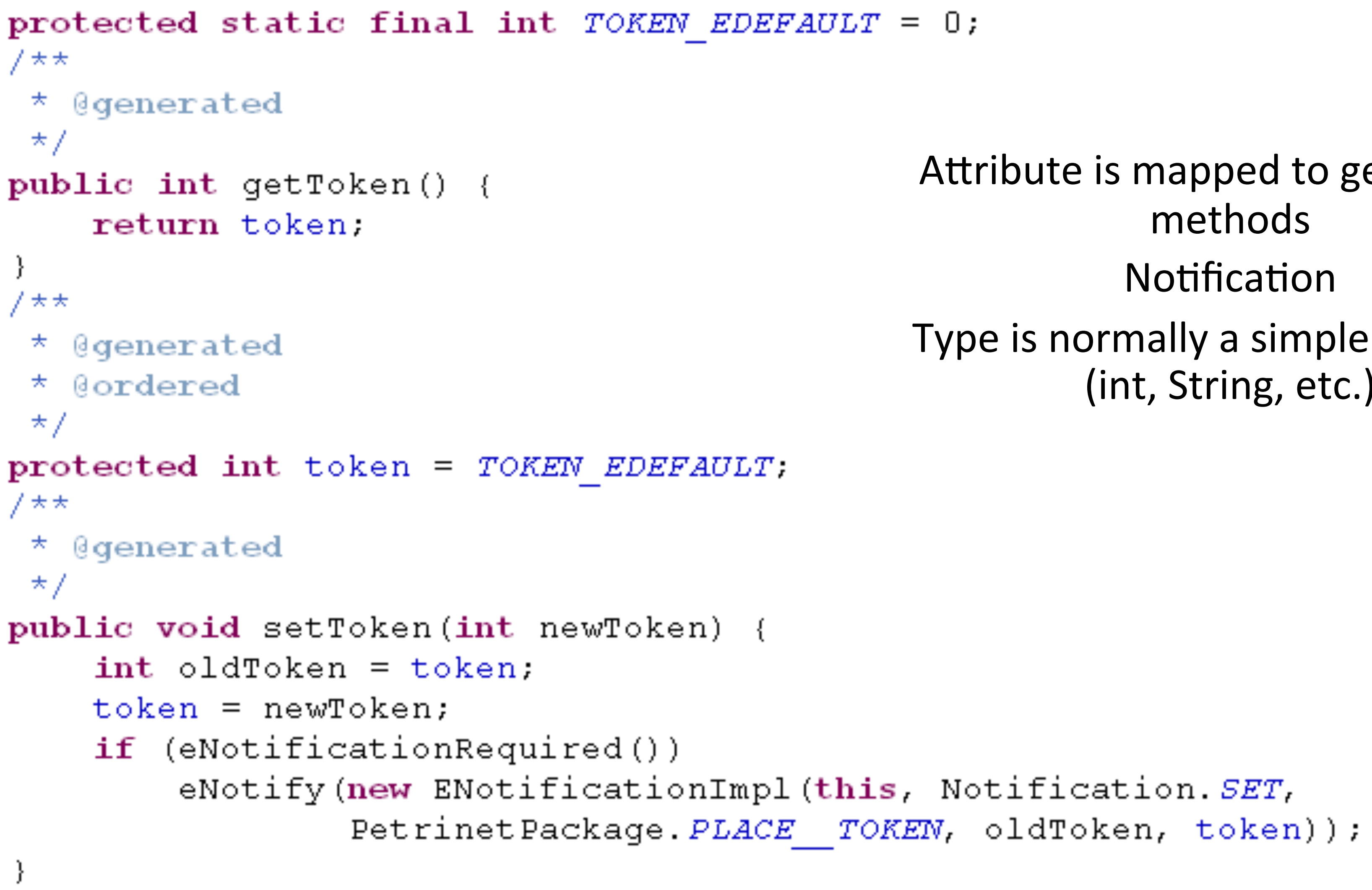
Every Class contains Framework specific methods:

Reflective get/set (eGet, eSet)

Consistency manipulation (eInverseRemove)


Retruns the static EClass Implemented by the EObject
«JavaInterface» Notifier
«JavaInterface» EObject
Returns the Resource of which the EObject is added
Reflective methods
Containment manipulation methods
+ eClass ( ) : EClass
+ eResource ( ) : Resource
+ eContainer ( ) : EObject
+ eContainingFeature ( ) : EStructuralFeature
+ eContainmentFeature ( ) : EReference
+ eContents ( ) : EList
+ eAllContents ( ) : TreeIterator
+ eIsProxy ( ) : boolean
+ eCrossReferences ( ) : EList
+ eGet ([in] feature : EStructuralFeature) : Object
+ eGet ([in] feature : EStructuralFeature , [in] resolve : boolean) : Object
+ eSet ([in] feature : EStructuralFeature , [in] newValue : Object) : void
+ eIsSet ([in] feature : EStructuralFeature) : boolean
+ eUnset ([in] feature : EStructuralFeature) : void

Every Interface extends the EObject Interface



```
protected static final int TOKEN_EDEFAULT = 0;
/**
 * @generated
 */
public int getToken() {
    return token;
}
/**
 * @generated
 * @ordered
 */
protected int token = TOKEN_EDEFAULT;
/**
 * @generated
 */
public void setToken(int newToken) {
    int oldToken = token;
    token = newToken;
    if (eNotificationRequired())
        eNotify(new ENotificationImpl(this, Notification.SET,
                PetrinetPackage.PLACE__TOKEN, oldToken, token));
}
```

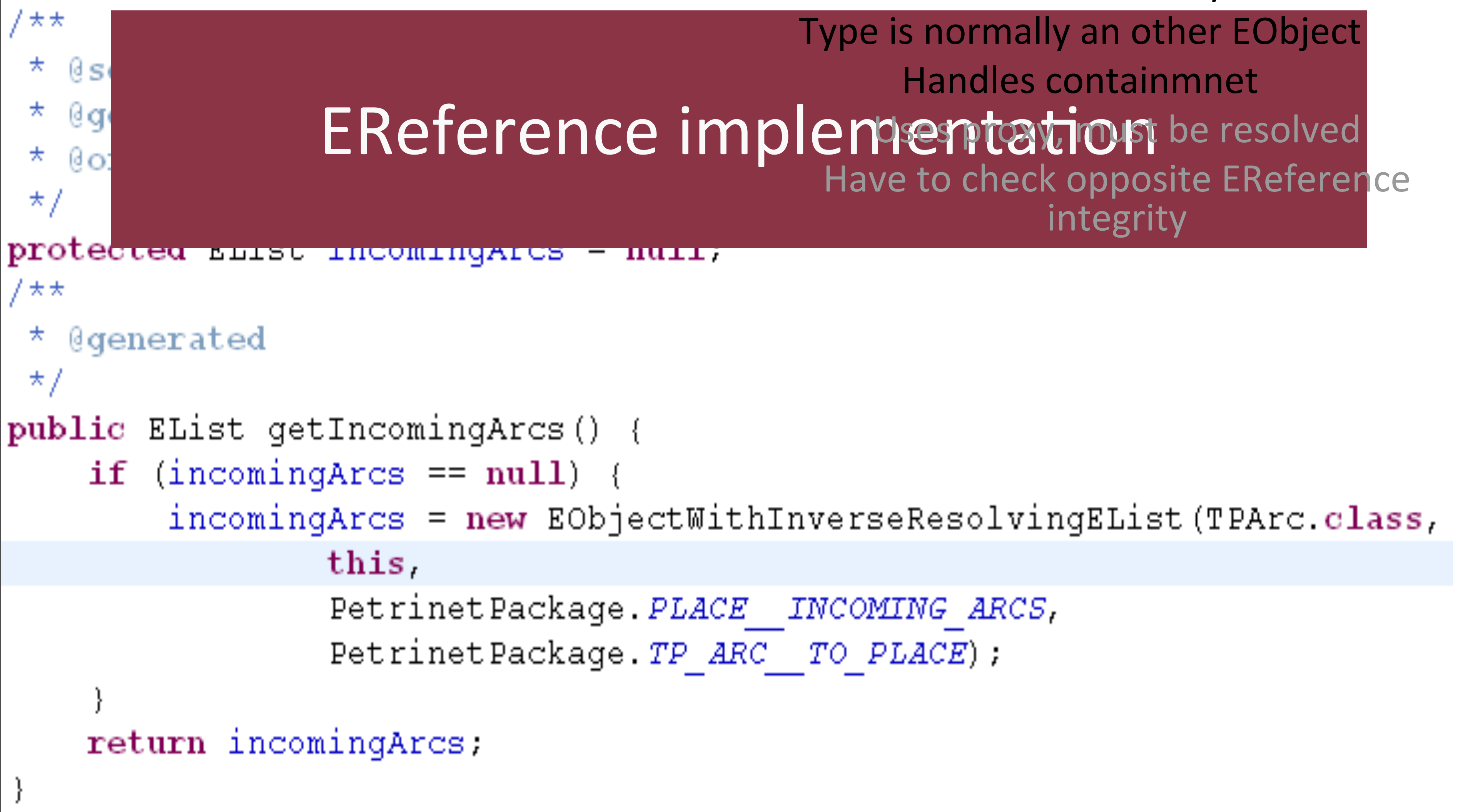
Attribute is mapped to get and set methods

Notification

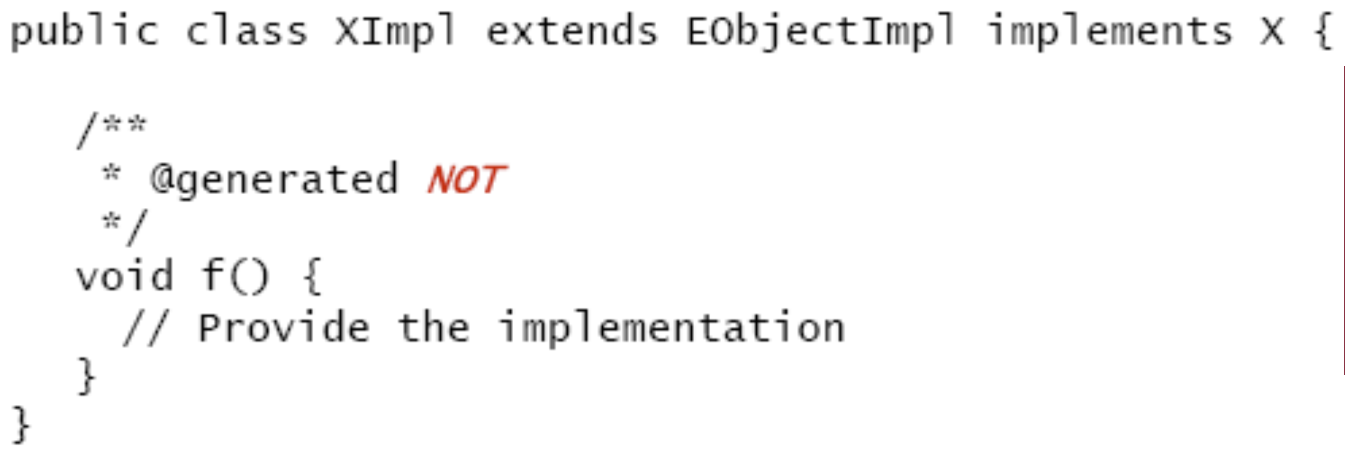
Type is normally a simple Java type (int, String, etc.)

EReference implementation

- Similar to the EAttribute
- Notification thrown by EList
- Type is normally an other EObject
- Handles containmnet
- Uses proxy, must be resolved
- Have to check opposite EReference integrity



```
/**
 * @s
 * @g
 * @o
 */
protected EList incomingArcs = null;
/**
 * @generated
 */
public EList getIncomingArcs() {
    if (incomingArcs == null) {
        incomingArcs = new EObjectWithInverseResolvingEList(TPArc.class,
                this,
                PetrinetPackage.PLACE__INCOMING_ARCS,
                PetrinetPackage.TP_ARC__TO_PLACE);
    }
    return incomingArcs;
}
```



```
public class XImpl extends EObjectImpl implements X {

    /**
     * @generated NOT
     */
    void f() {
      // Provide the implementation
    }
}
```

on

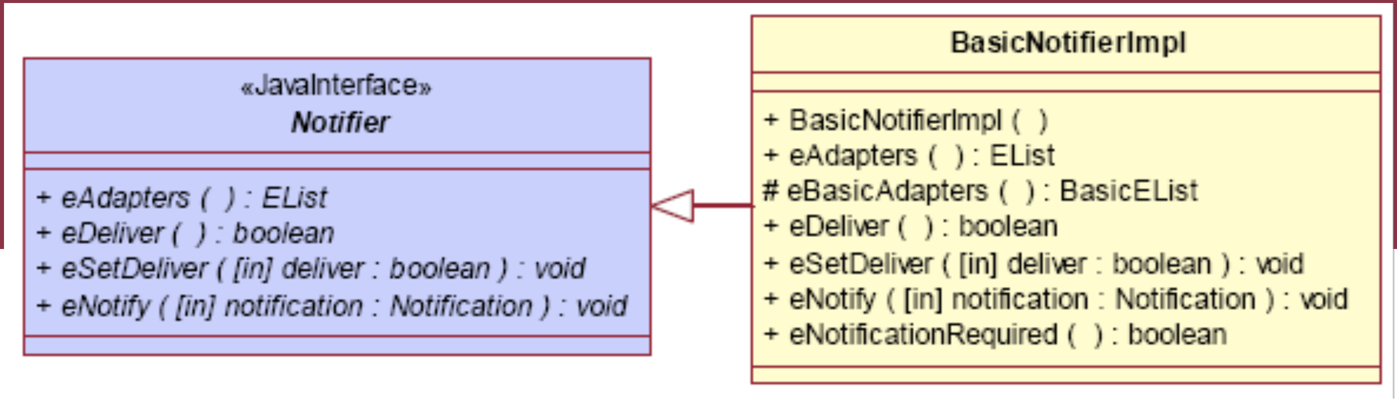
Represents the frame of a Java method

Represented both in the interface and implementing class

Important:

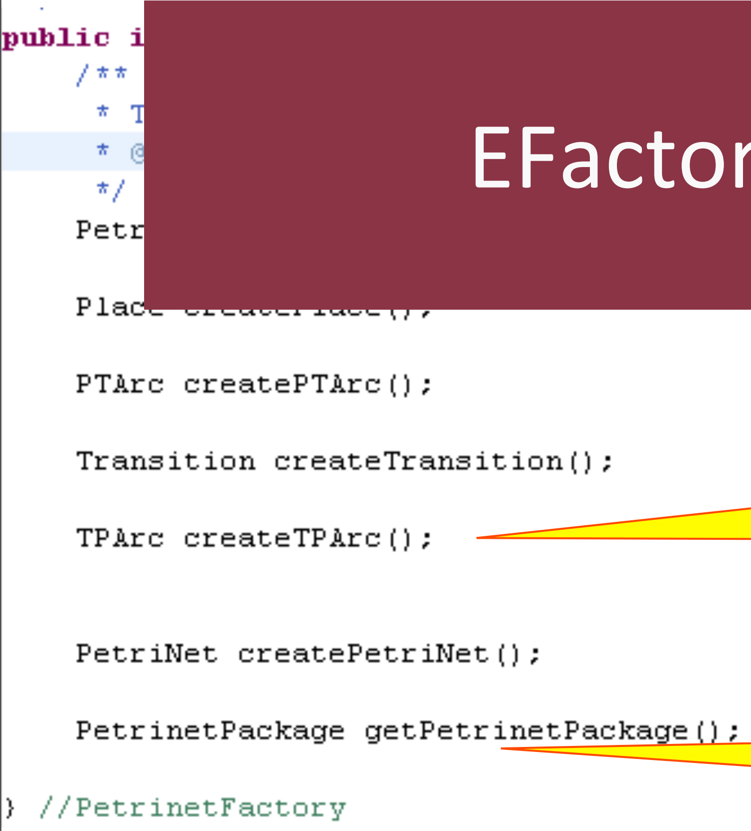
Have to change the generated annotation to NOT

Have to implement the method manually


«JavaInterface»
Notifier
+ eAdapters ( ) : EList
+ eDeliver ( ) : boolean
+ eSetDeliver ([in] deliver : boolean) : void
+ eNotify ([in] notification : Notification) : void
BasicNotifierImpl
+ BasicNotifierImpl ( )
+ eAdapters ( ) : EList
# eBasicAdapters ( ) : BasicEList
+ eDeliver ( ) : boolean
+ eSetDeliver ([in] deliver : boolean) : void
+ eNotify ([in] notification : Notification) : void
+ eNotificationRequired ( ) : boolean

- Observer pattern
- Behavior extension
- Events stored in a Notification class
- Can be parametrized in the genmodel

EFactory implementation



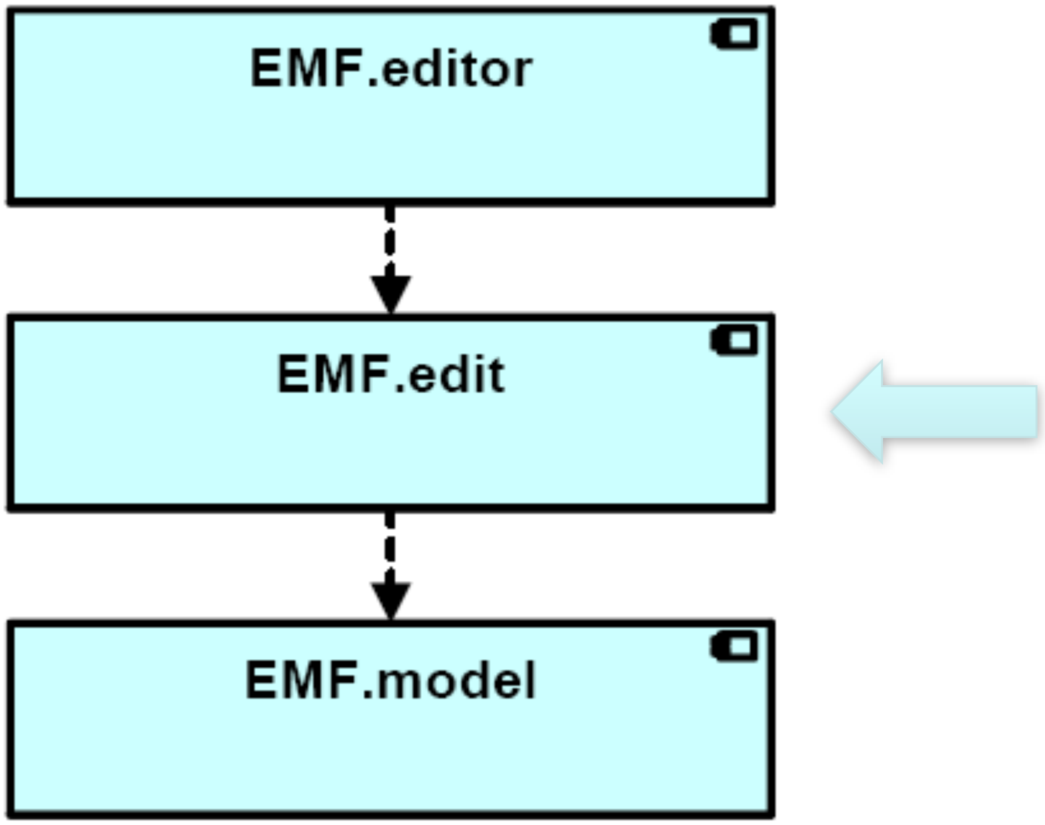
```
public i
    /**
     * T
     * @
     */
    Petr

    Plac

    PTArc createPTArc();

    Transition createTransition();

    TPArc createTPArc();


    PetriNet createPetriNet();

    PetrinetPackage getPetrinetPackage();

} //PetrinetFactory
```

Single instance

Create methods by type

One-to-one reference to Petrinet EPackage

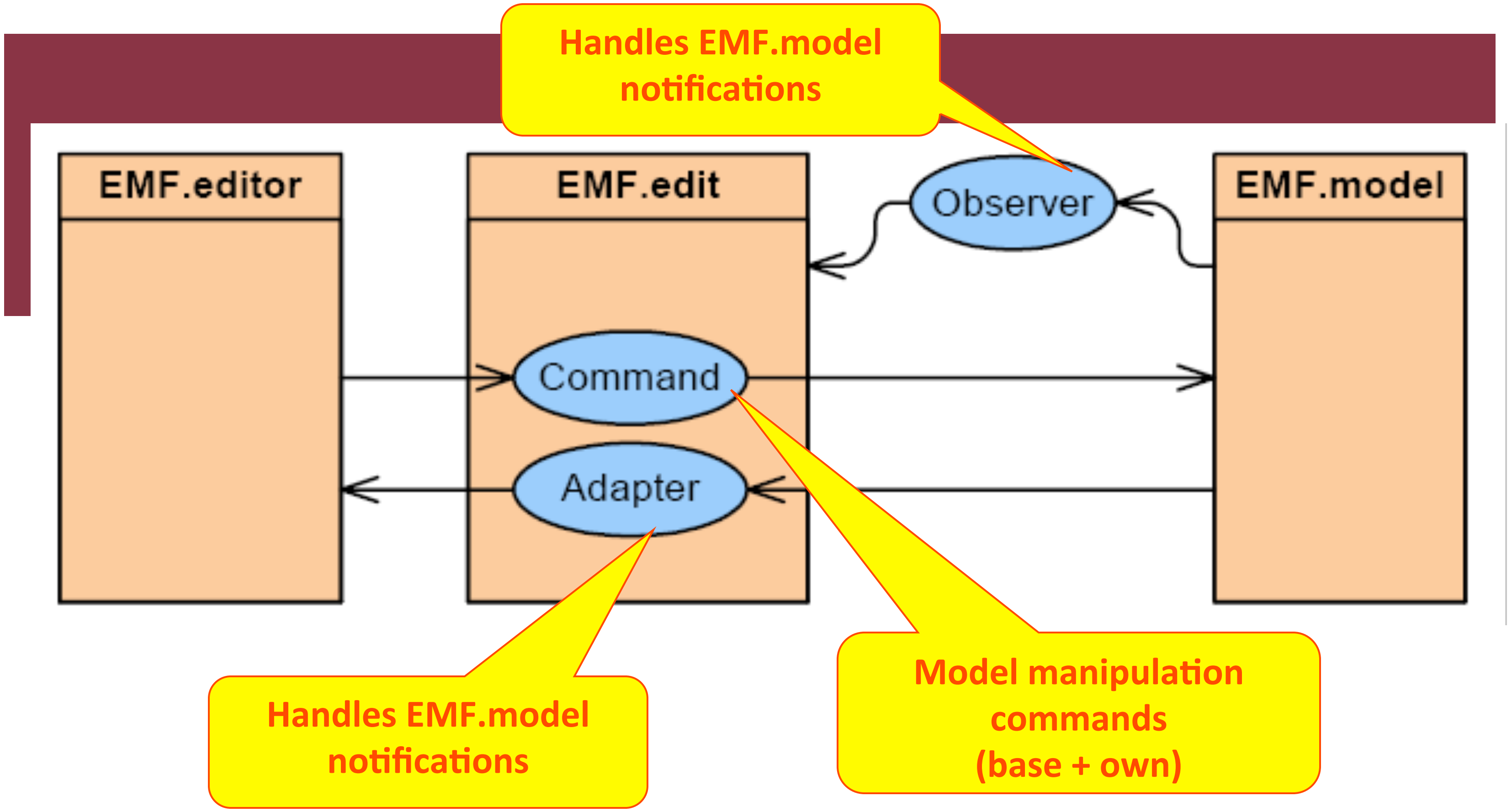
Generated EMF components



EMF.Edit

Separates the GUI and the model

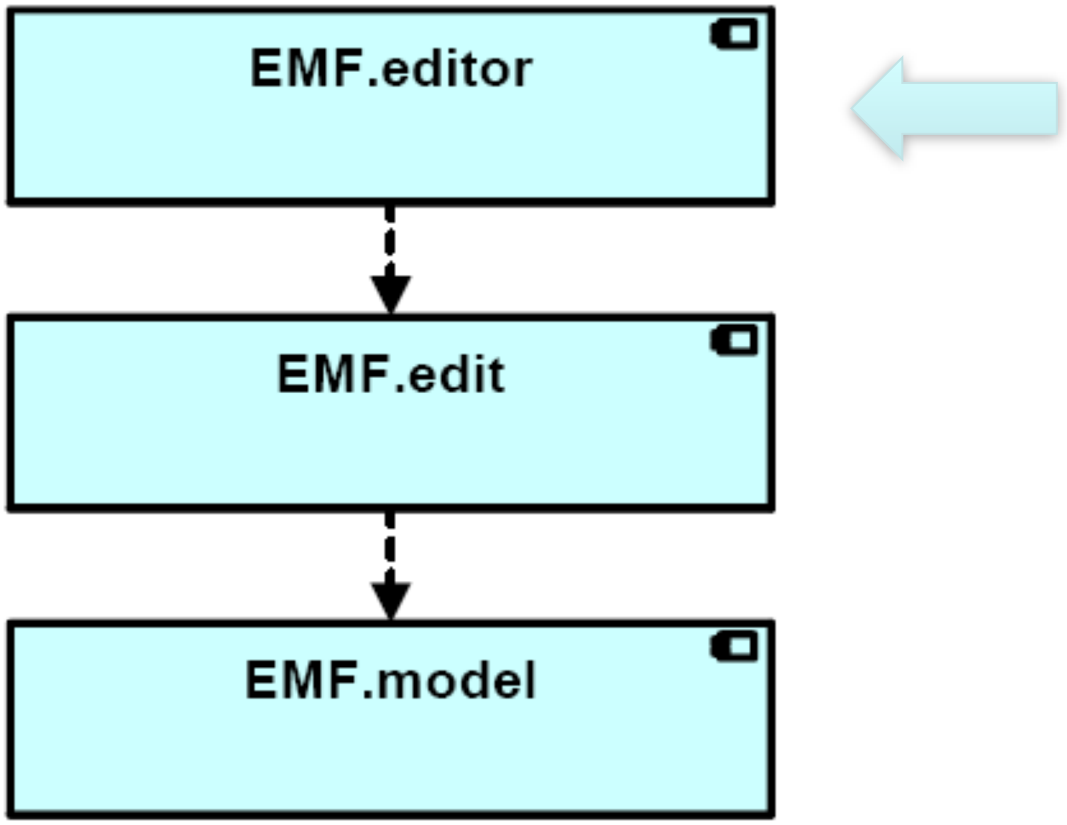
GUI independent commands

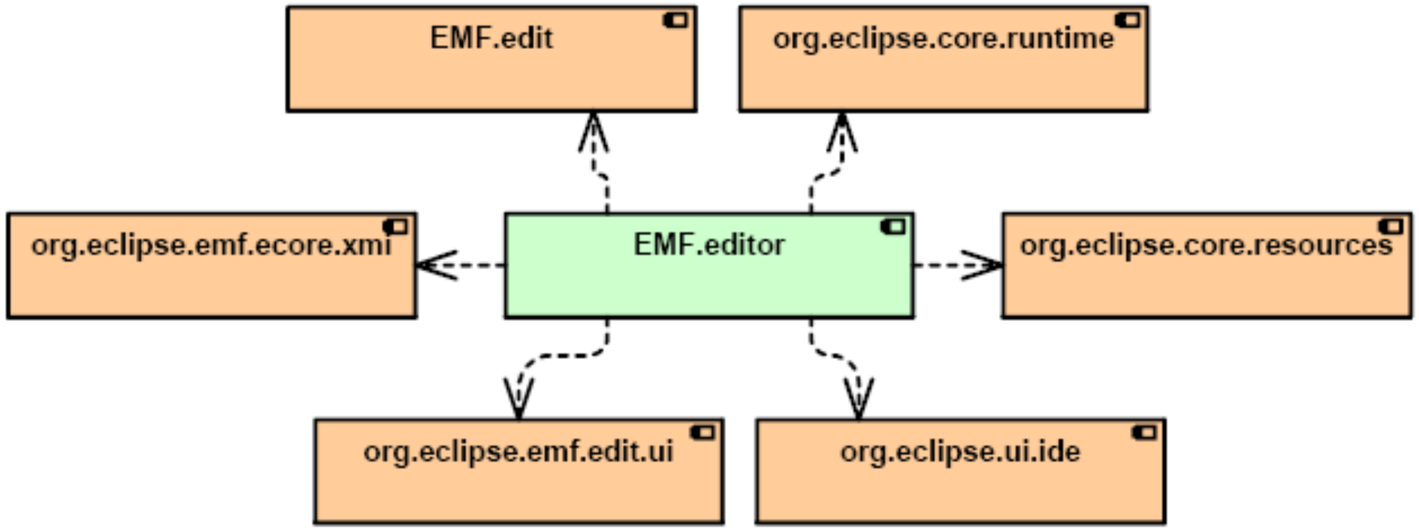

Handles EMF.model notifications
EMF.editor
EMF.edit
Observer
EMF.model
Command
Adapter
Handles EMF.model notifications
Model manipulation commands (base + own)

EMF.Edit Command

- All model manipulation through commands
- Based on template method pattern
- The ItemProvider implements the createCommand(…) method, which calls one of its protected command method
- Customizable (usually, modify already implemented „protected” commands)

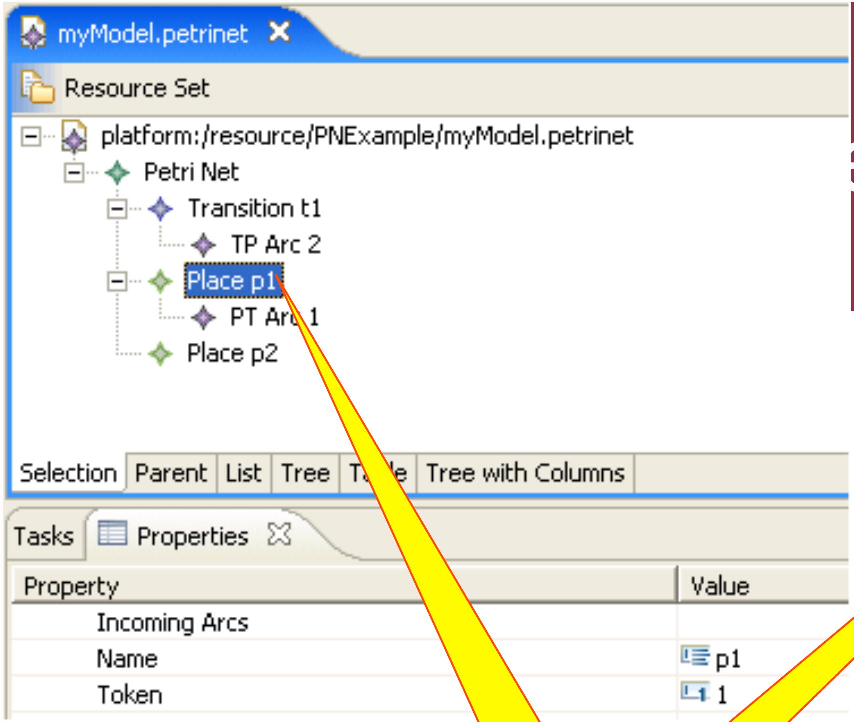
Generated EMF components





- EMF.Editor generates the SWT/JFace for the graphical editor
- Generates:
  - Tree editor
  - Wizards
  - Menus
  - plugins

etri Net models



myModel.petrinet

Resource Set

- platform:/resource/PNExample/myModel.petrinet
  - Petri Net
    - Transition t1
      - TP Arc 2
    - Place p1
      - PT Arc 1
    - Place p2

Selection | Parent | List | Tree | Table | Tree with Columns

Tasks | Properties

Property	Value
Incoming Arcs	
Name	p1
Token	1

```
<?xml version="1.0" encoding="UTF-8"?>
<PetriNets.petrinet:PetriNet xmi:version="2.0"
xmlns:xmi="http://www.omg.org/XMI"
xmlns:PetriNets.petrinet=
"http:///PetriNets/petrinet.ecore">
<transitions name="t1" incomingArcs=
"//@places.0/@outgoingArcs.0">
<outgoingArcs weight="2" toPlace="//@places.
1"/>
</transitions>
<places name="p1" token="1">
<outgoingArcs weight="1" toTransition="//
@transitions.0"/>
</places>
<places name="p2" incomingArcs=
"//@transitions.0/@outgoingArcs.0"/>
</PetriNets.petrinet:PetriNet>
```

Place p1

Reference: URI (or XMI.id)

Tree View

XMI 2.0 View

Tools, API and Utilities

The org.eclipse.emf.ecore.util Package

Contains a number of miscellaneous utility classes and interfaces:

Client Programming with EMF

```
Place p1 = PetrinetFactory.eINSTANCE.createPlace();
p1.setName("p1");
Place p2 = PetrinetFactory.eINSTANCE.createPlace();
p2.setName("p2");
```

Create a place

Create a transition

Create a PT arc

Set source of PT arc

Set target of PT arc

Advanced client programming: Reflective Ecore API

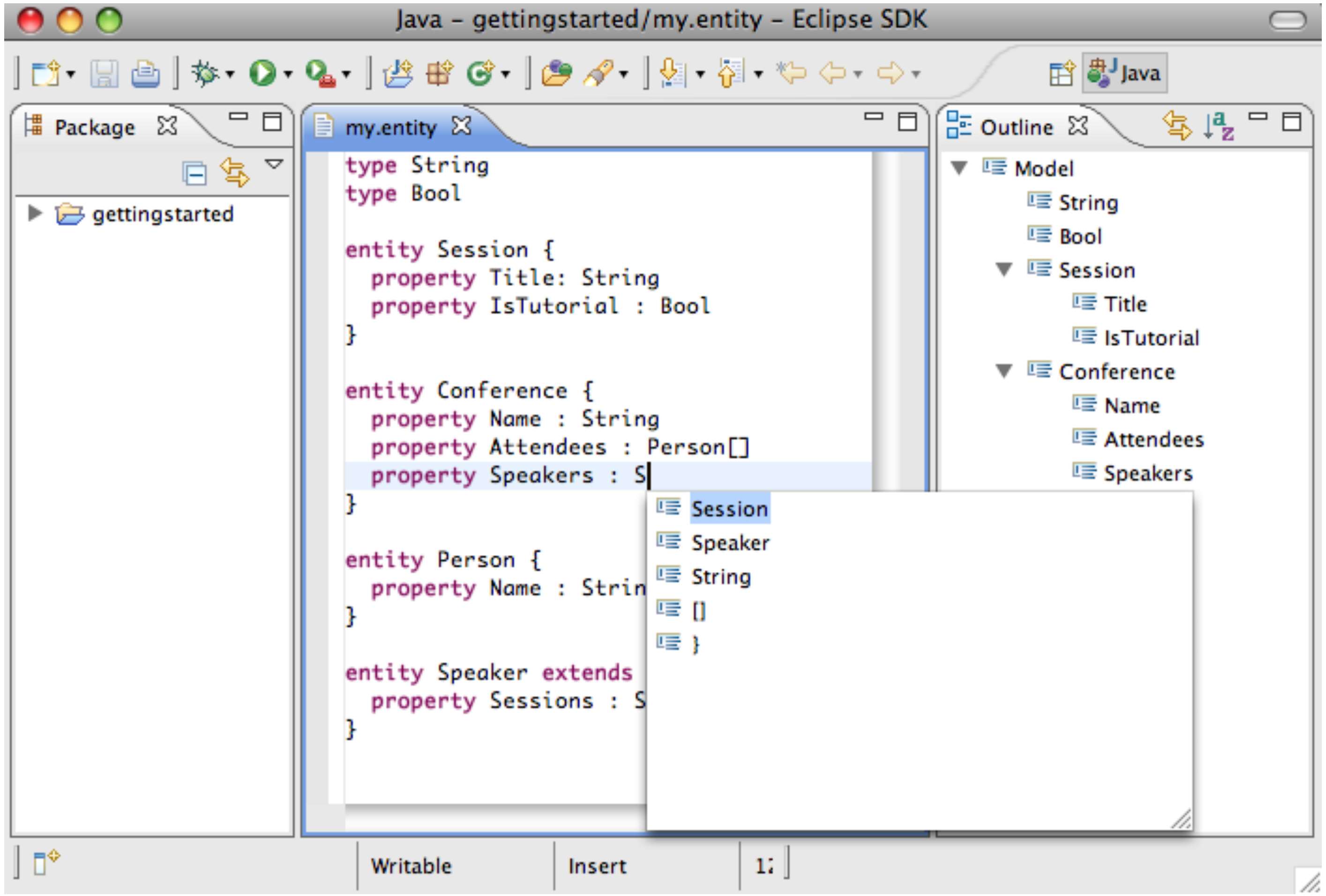
Basic EMF tools

Validation

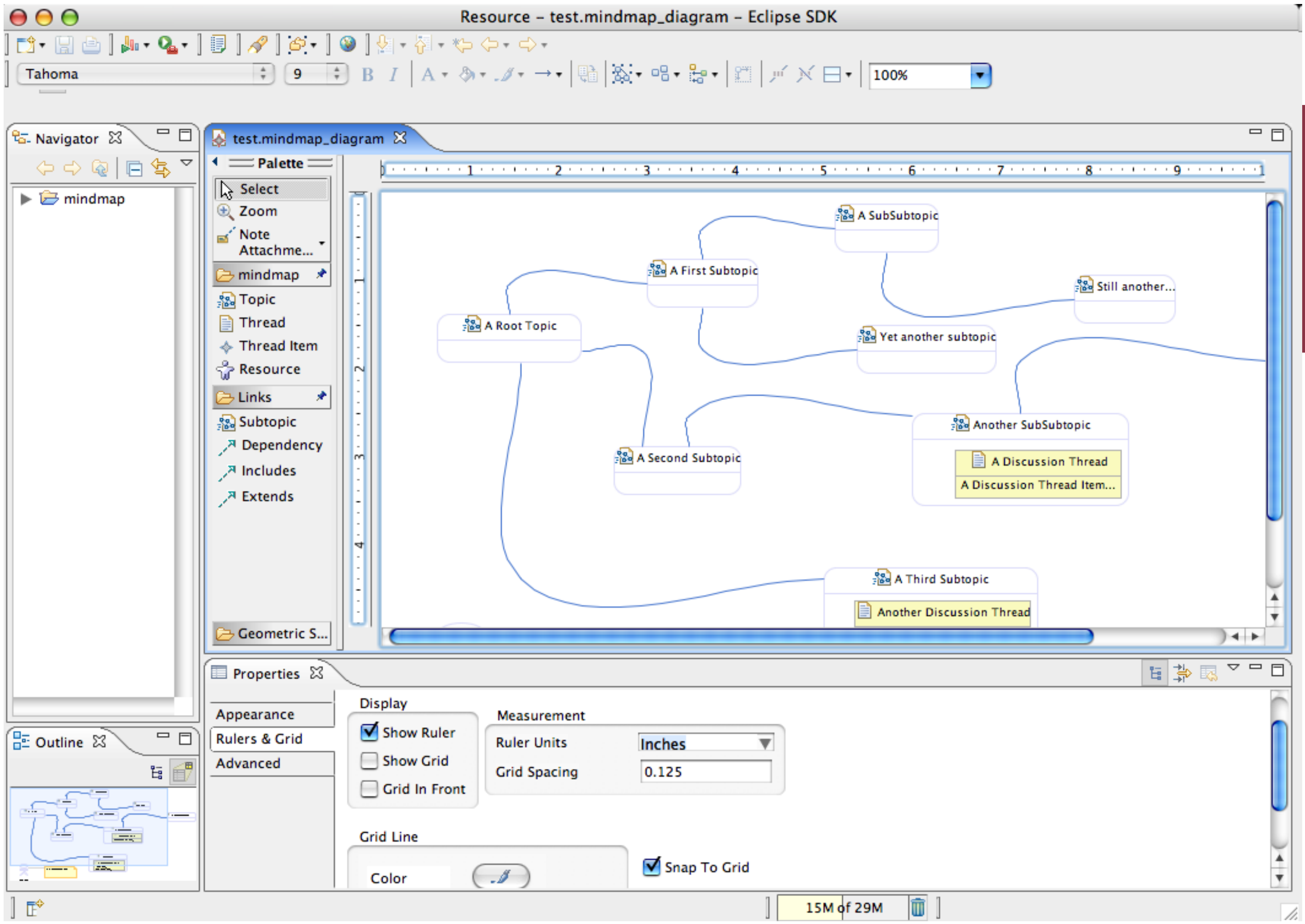
Validate constraints over EMF models

Query

XText



```
type String
type Bool

entity Session {
  property Title: String
  property IsTutorial : Bool
}

entity Conference {
  property Name : String
  property Attendees : Person[]
  property Speakers : S
}

entity Person {
  property Name : Strin
}

entity Speaker extends
  property Sessions : S
}
```



Display
- [✓] Show Ruler
- [ ] Show Grid
- [ ] Grid In Front

Measurement

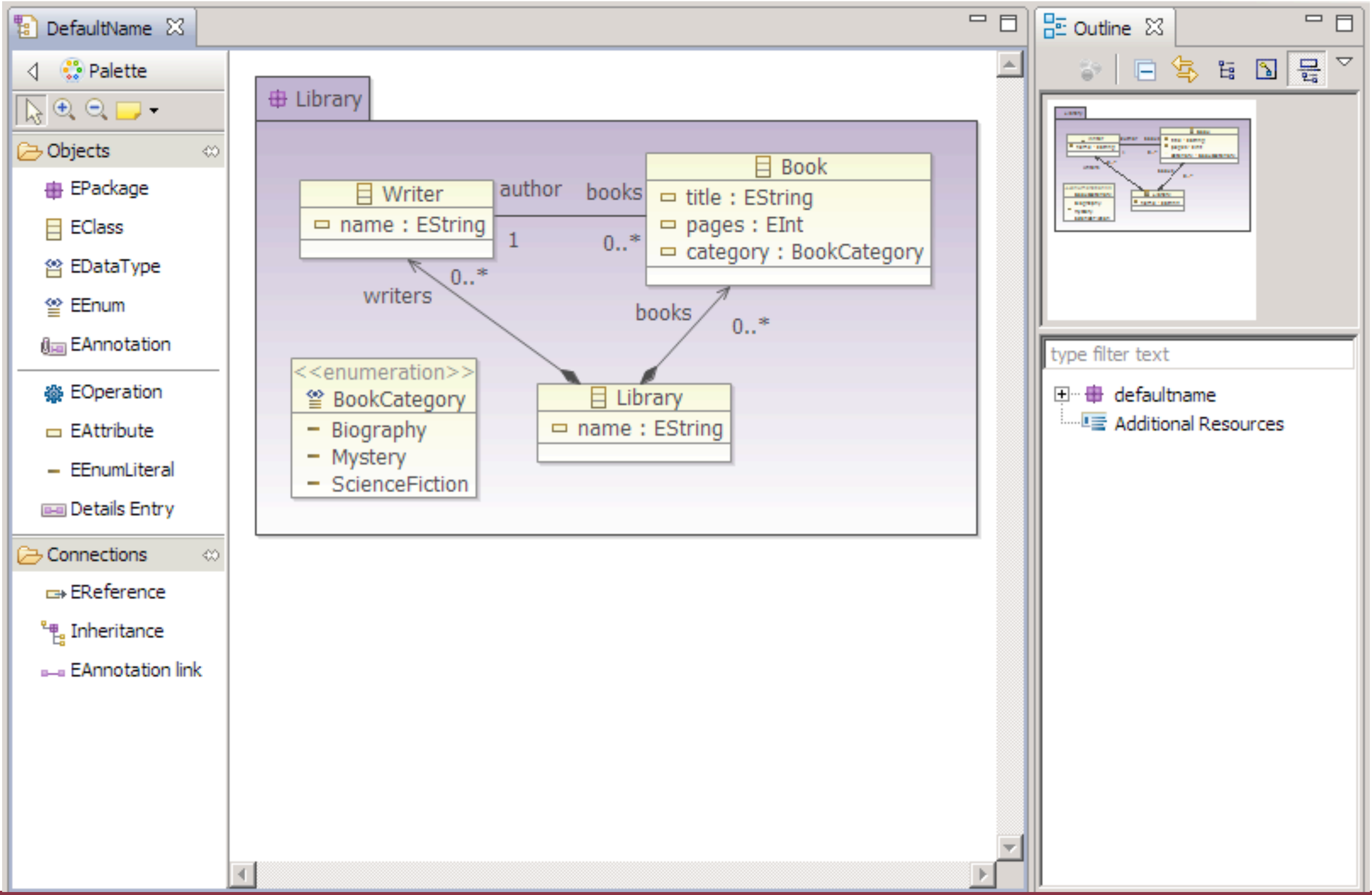
Ruler Units	Inches
Grid Spacing	0.125

Grid Line

Color

- [✓] Snap To Grid

Ecore Tools: Ecore Diagram Editor (GMF)


DefaultName
Palette
Objects
EPackage
EClass
EDataType
EEnum
EAnnotation
EOperation
EAttribute
EEnumLiteral
Details Entry
Connections
EReference
Inheritance
EAnnotation link
Library
Writer
name : EString
author
books
1
0..*
Book
title : EString
pages : EInt
category : BookCategory
writers
0..*
books
0..*
<<enumeration>>
BookCategory
Biography
Mystery
ScienceFiction
Library
name : EString
Outline
type filter text
defaultname
Additional Resources