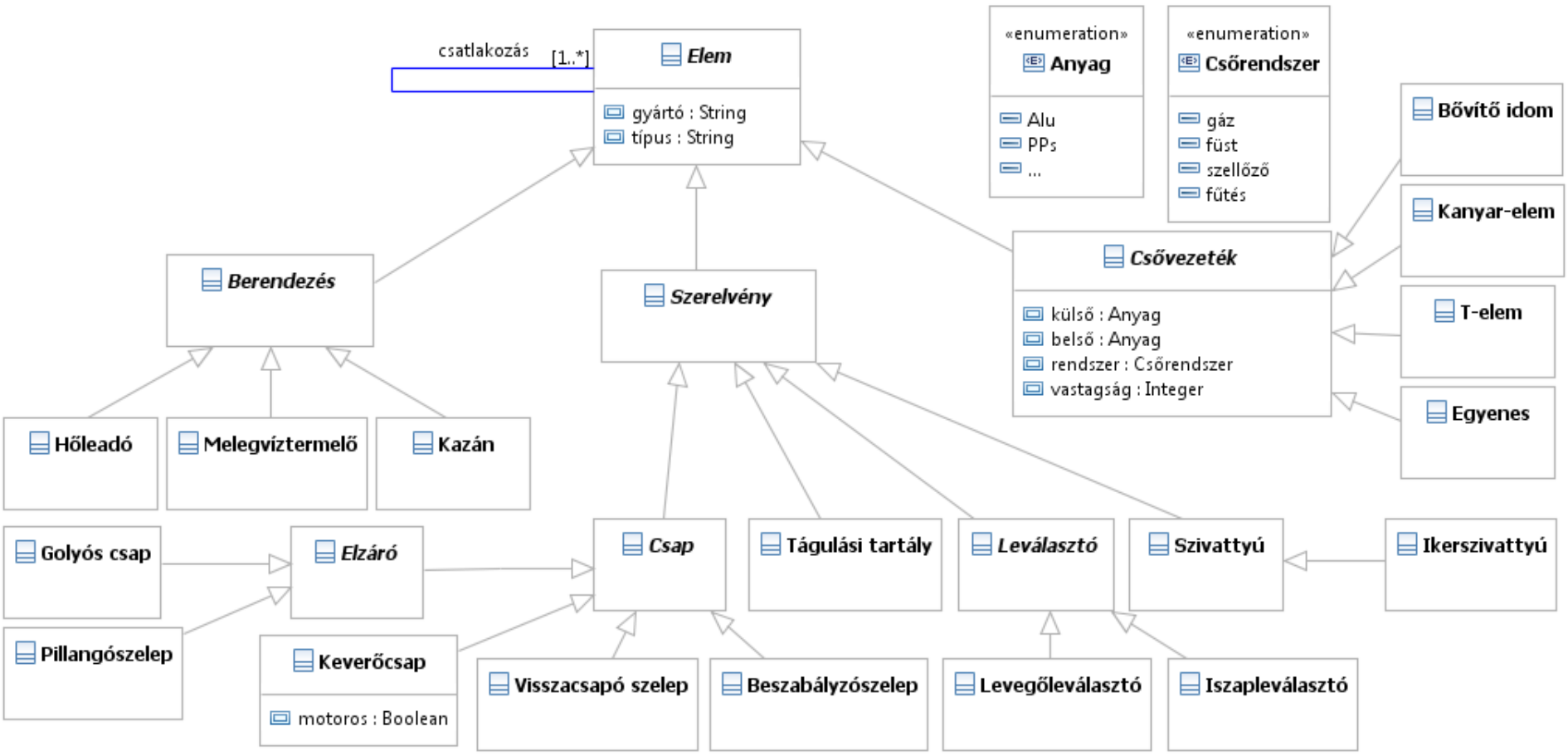
Domain-specific modeling
(and the Eclipse Modeling Framework)

Ákos Horváth
Gábor Bergmann
Dániel Varró
István Ráth

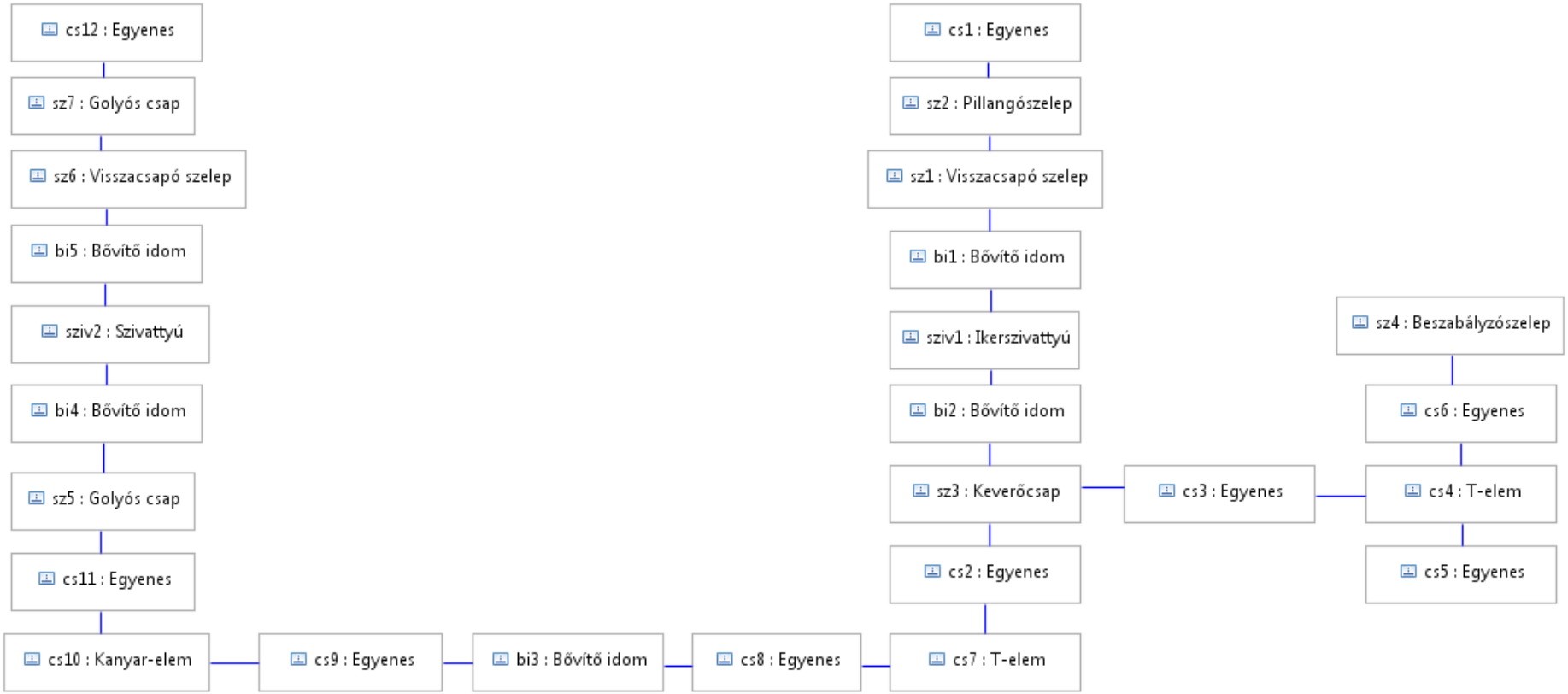
Model Driven Software Development
Lecture 7

MOTIVATION

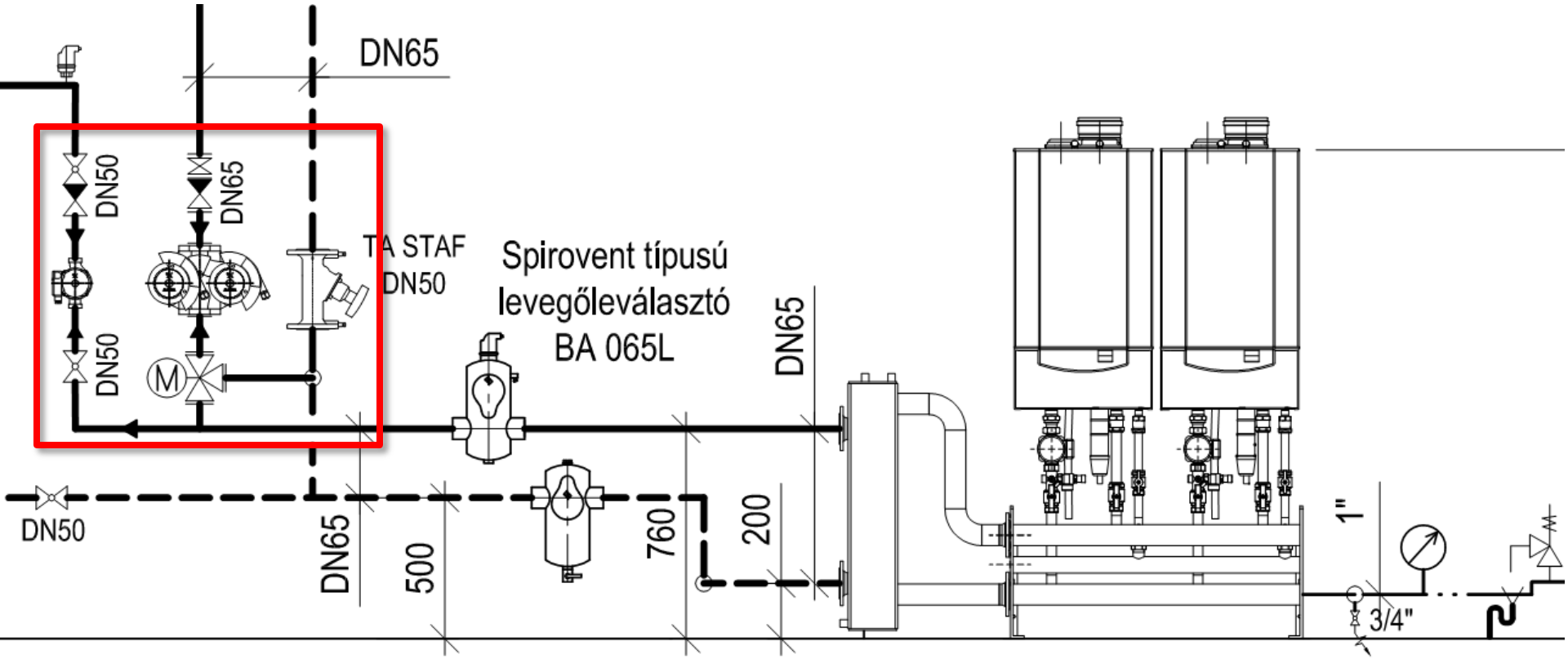
Example metamodel



Instance model, abstract syntax



Instance model, concrete syntax

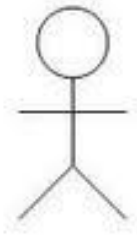


Honeywell keverőcsap DN50 K_{VS} 40

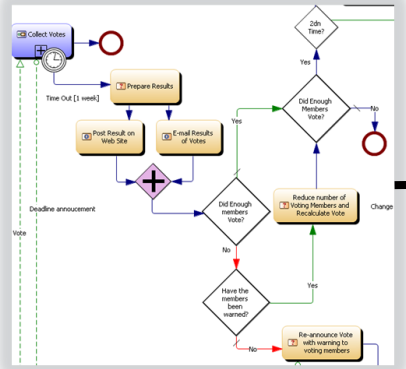
Spirovent típusú iszapleválasztó BE 065L

Remeha Quinta kaszkád rendszer hidraulikus váltóval

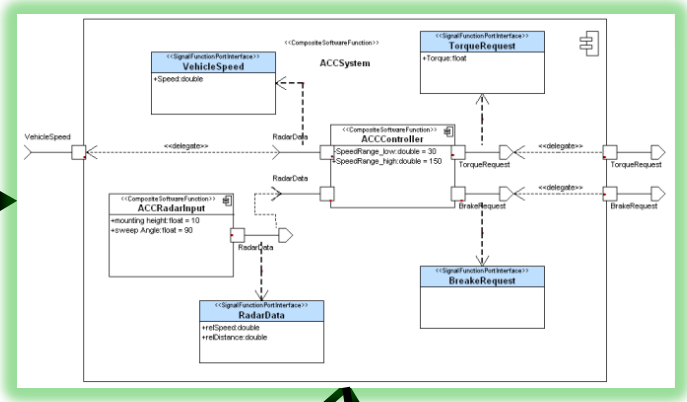
Domain specific modeling languages

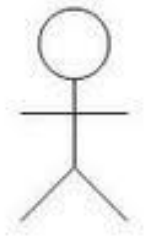


Business analyst

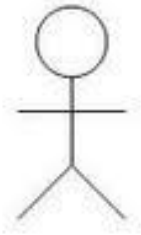


Business process

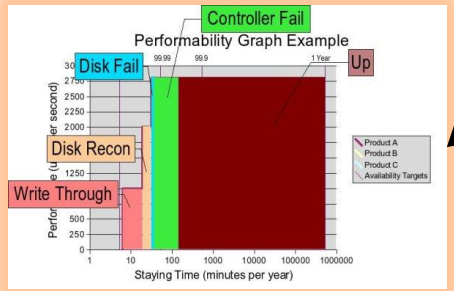




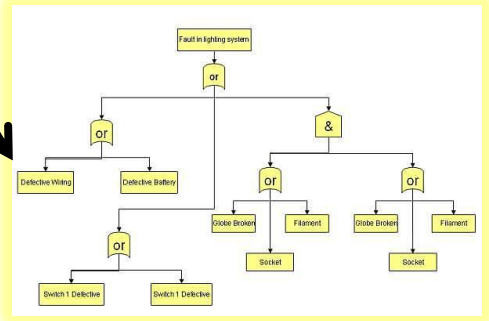
System designer



Dependability expert



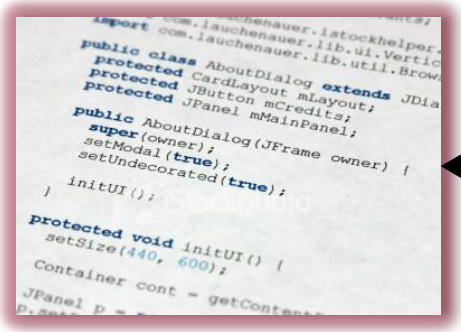
Dependability model



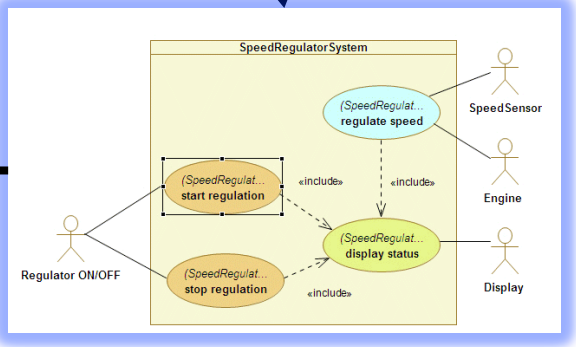
Risk model

Security expert

Software developer



Programming language

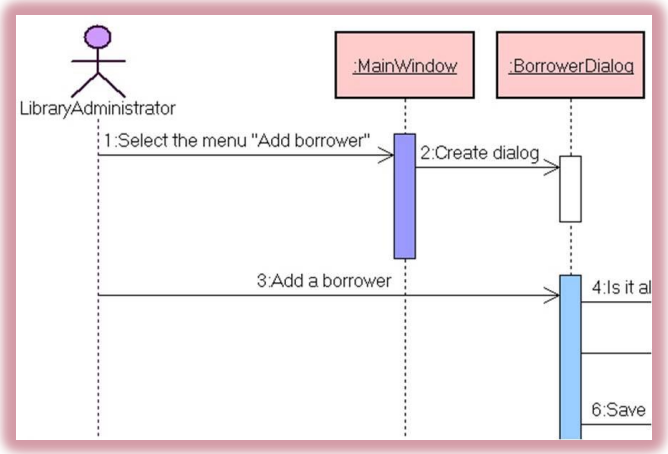


Software model

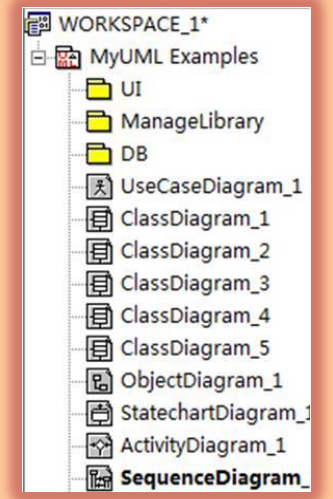
Software architect

Usage example of DSMs

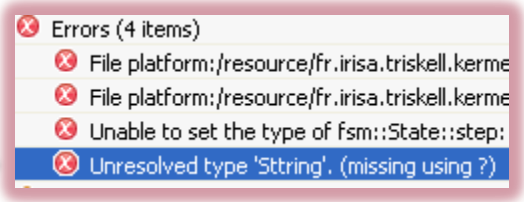
Concrete syntax



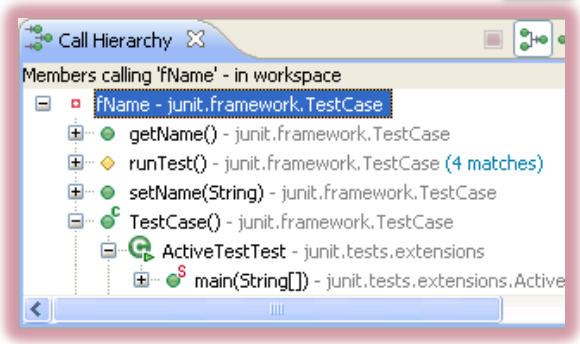
Abstract syntax



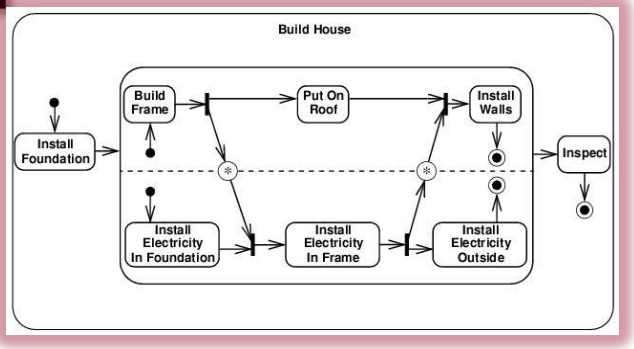
Well-formedness constraints



Behavioural semantics, simulation, refactoring



Call graph (view)



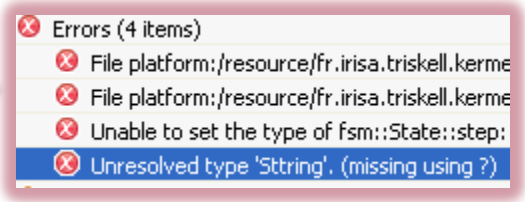
State machines (different DSM)

Structure of DSMs

Graphical syntax

Abstract syntax

Well-formedness constraints

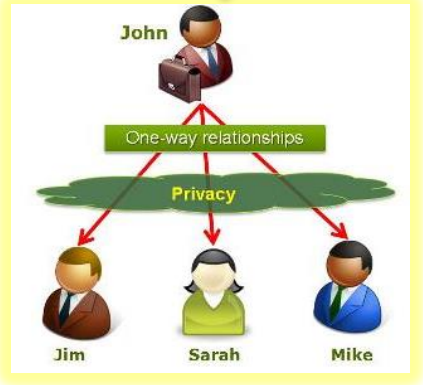


Behavioural semantics, simulation

Code generation

Mapping

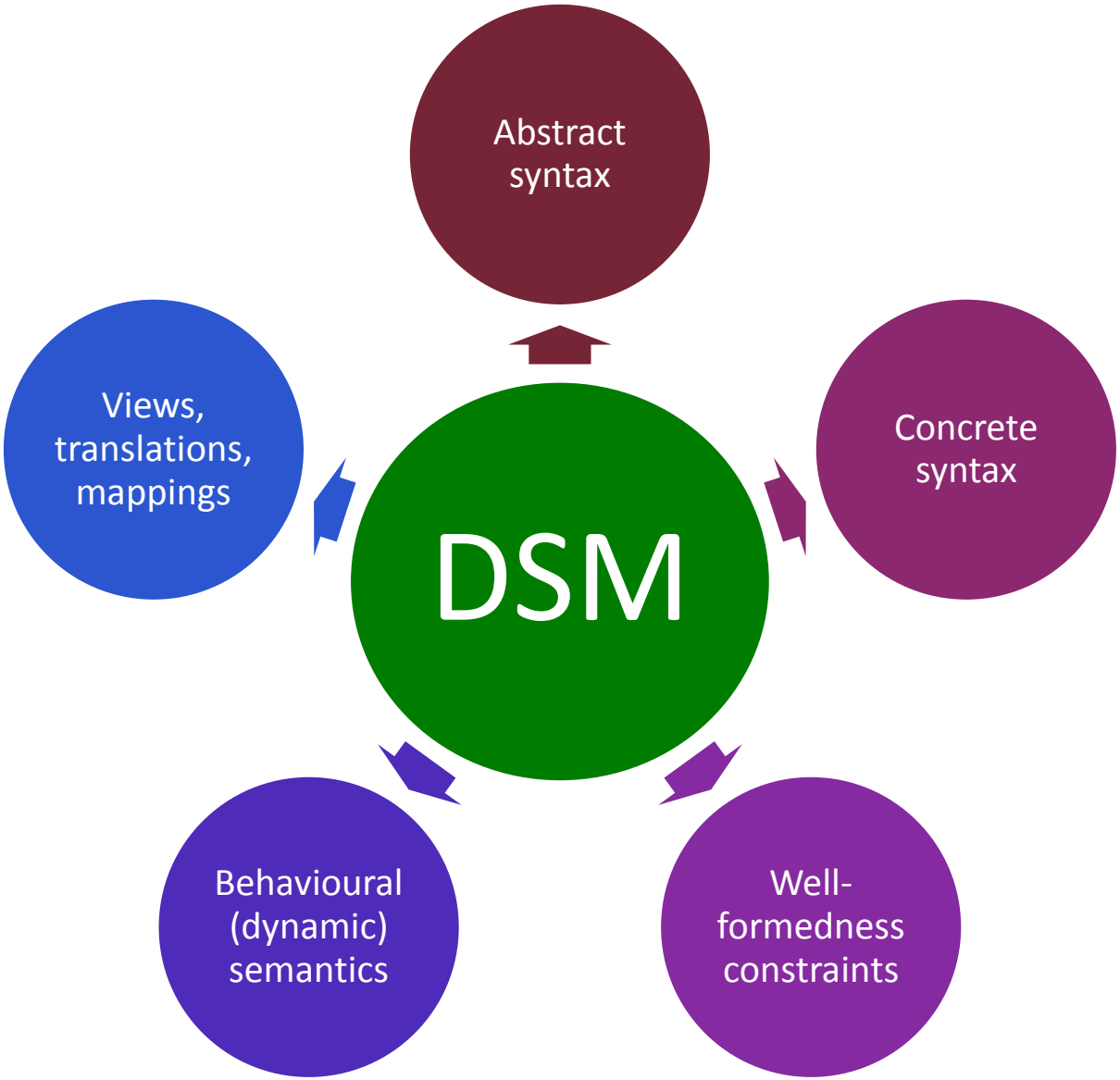
Textual syntax



View

Code (documentation, configuration)

DSM aspects

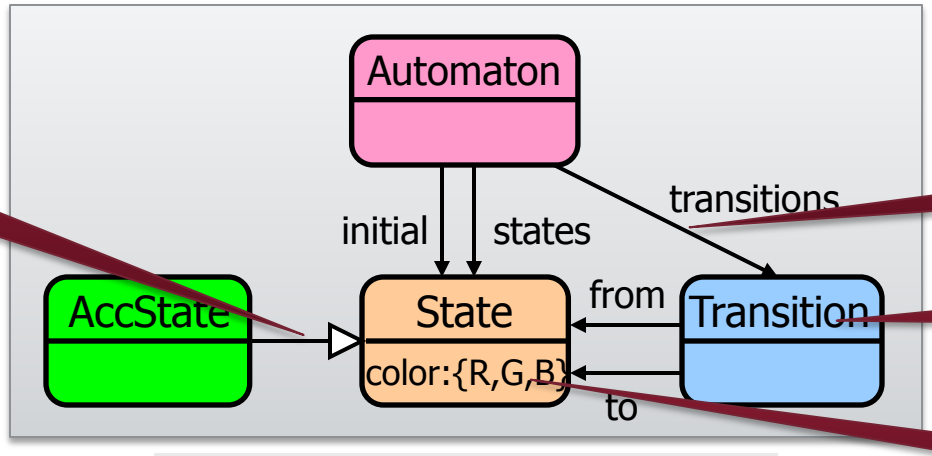

Abstract syntax
Concrete syntax
Well-formedness constraints
Behavioural (dynamic) semantics
Views, translations, mappings
DSM

DOMAIN SPECIFIC MODELING

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

Revisiting the example



Generalization

Association

Instantiation

Class

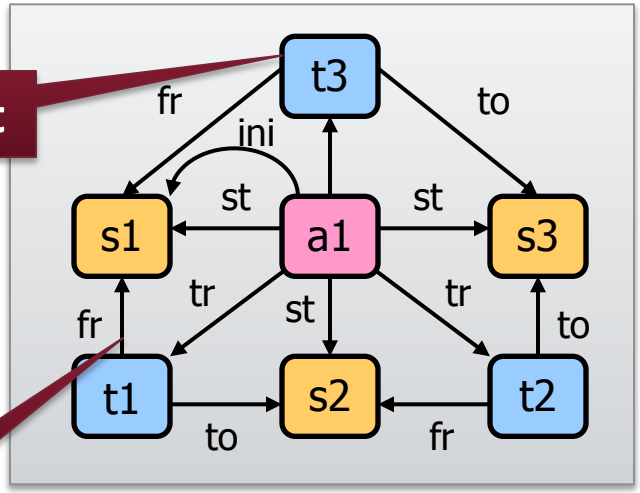
Attribute

Metamodel

Meta (Language) level

(Instance) Model level

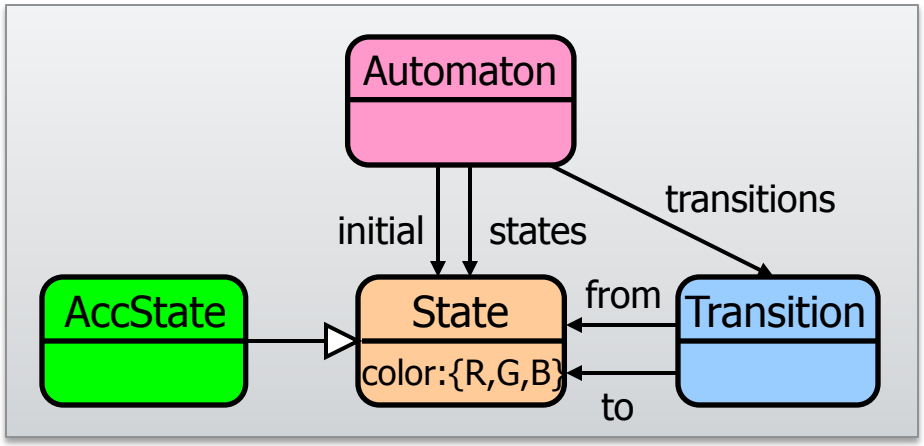
Object



Link

Model in abstract syntax

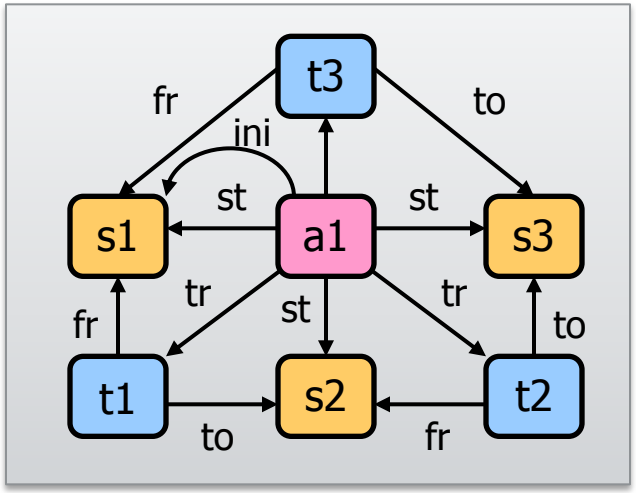
Relationship of concepts



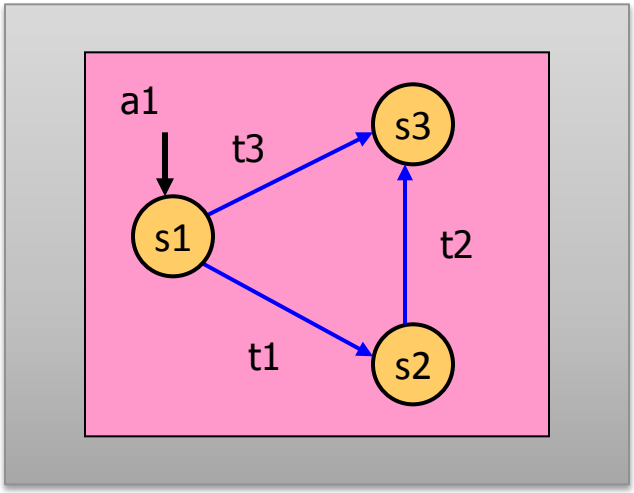
Meta (Language) level

Metamodel

Model level

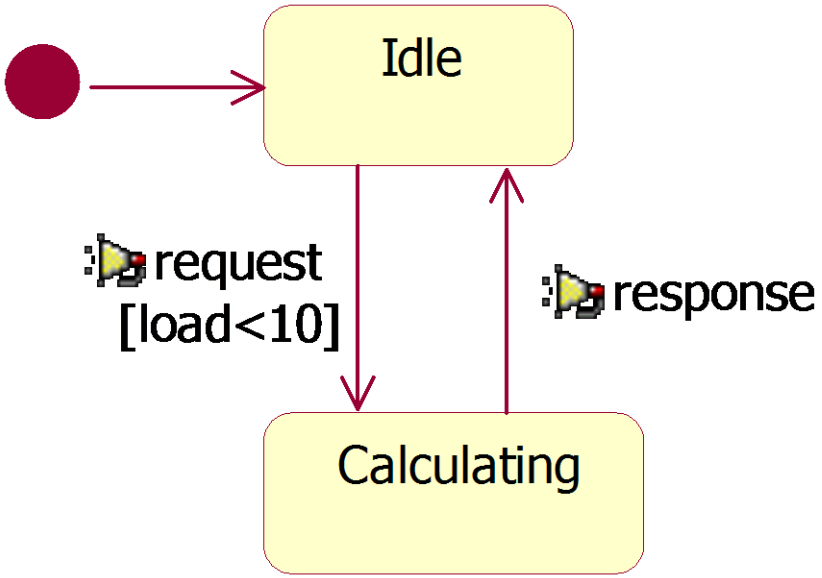


Abstract syntax



Concrete syntax

Example: Concrete Syntax




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

Graphical notation

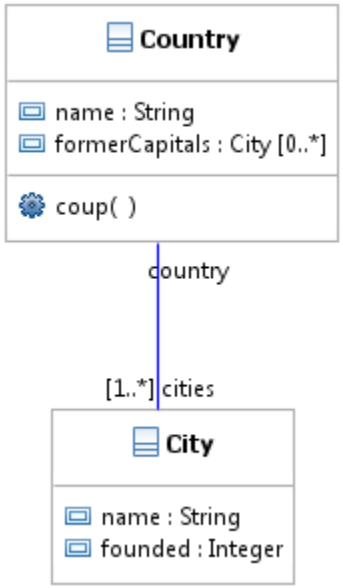
Textual notation

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: UML model

<Package> geography
- <Element Import> Boolean
- <Element Import> String
- <Element Import> UnlimitedNatural
- <Element Import> Integer
- <Class> Country
  - <Property> name : String
  - <Property> formerCapitals : City [0..*]
    - <Literal Unlimited Natural> *
    - <Literal Integer> 0
  - <Operation> coup ()
- <Class> City
  - <Property> name : String
  - <Property> founded : Integer
- <Association> A_country_cities
  - <Property> country : Country
    - <Literal Unlimited Natural> 1
    - <Literal Integer> 1
  - <Property> cities : City [1..*]
    - <Literal Unlimited Natural> *
    - <Literal Integer> 1




```
<?xml version="1.0" encoding="UTF-8"?>
<uml:Package name="geography" xmi:version="2.1"
   xmlns:xmi="http://schema.omg.org/spec/XMI/2.1"
   xmlns:uml="http://www.eclipse.org/uml2/3.0.0/UML"
   xmi:id="_7qi_AS2uEd-VCP9iY9GYHg">
[...]
  <packagedElement xmi:type="uml:Class" name="Country" xmi
    <ownedAttribute name="name" aggregation="composite" xm
      <type xmi:type="uml:PrimitiveType" href="pathmap://U
    </ownedAttribute>
    <ownedAttribute name="formerCapitals" aggregation="com
      <upperValue value="*" xmi:type="uml:LiteralUnlimited
      <lowerValue xmi:type="uml:LiteralInteger" xmi:id="_y
    </ownedAttribute>
    <ownedOperation name="coup" xmi:id="_fHicEC2vEd-VCP9iY
      <ownedParameter direction="return" xmi:id="_1e7b8C2v
    </ownedOperation>
  </packagedElement>
  <packagedElement xmi:type="uml:Class" name="City" xmi:id
    <ownedAttribute name="name" aggregation="composite" xm
      <type xmi:type="uml:PrimitiveType" href="pathmap://U
    </ownedAttribute>
    <ownedAttribute name="founded" aggregation="composite"
      <type xmi:type="uml:PrimitiveType" href="pathmap://U
    </ownedAttribute>
  </packagedElement>
  <packagedElement xmi:type="uml:Association" xmi:id="_Xq_
    <ownedEnd name="cities" type="__KgpUC2vEd-VCP9iY9GYHg"
      <upperValue value="*" xmi:type="uml:LiteralUnlimited
```

Abstract Syntax

Graphical notation (Class Diagram)

Textual notation (XMI 2.1)

Multiplicity of Notations

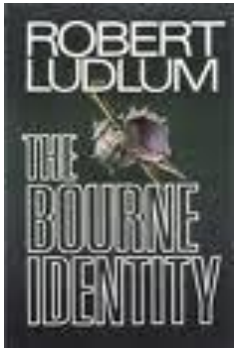
- One-to-many
  - 1 abstract syntax → many textual and visual notations
    - Human-readable-writable textual or visual syntax
    - Textual syntax for exchange or storage (typically XML)
    - In case of UML, each diagram is only a partial view
  - 1 abstract model → many concrete forms in 1 syntax!
    - Whitespace, diagram layout
    - Comments
    - Syntactic sugar
  - 1 semantic interpretation → many abstract models
    - e.g. UML2 Attribute vs. one-way Association

METALEVELS

Exercise Metalevels

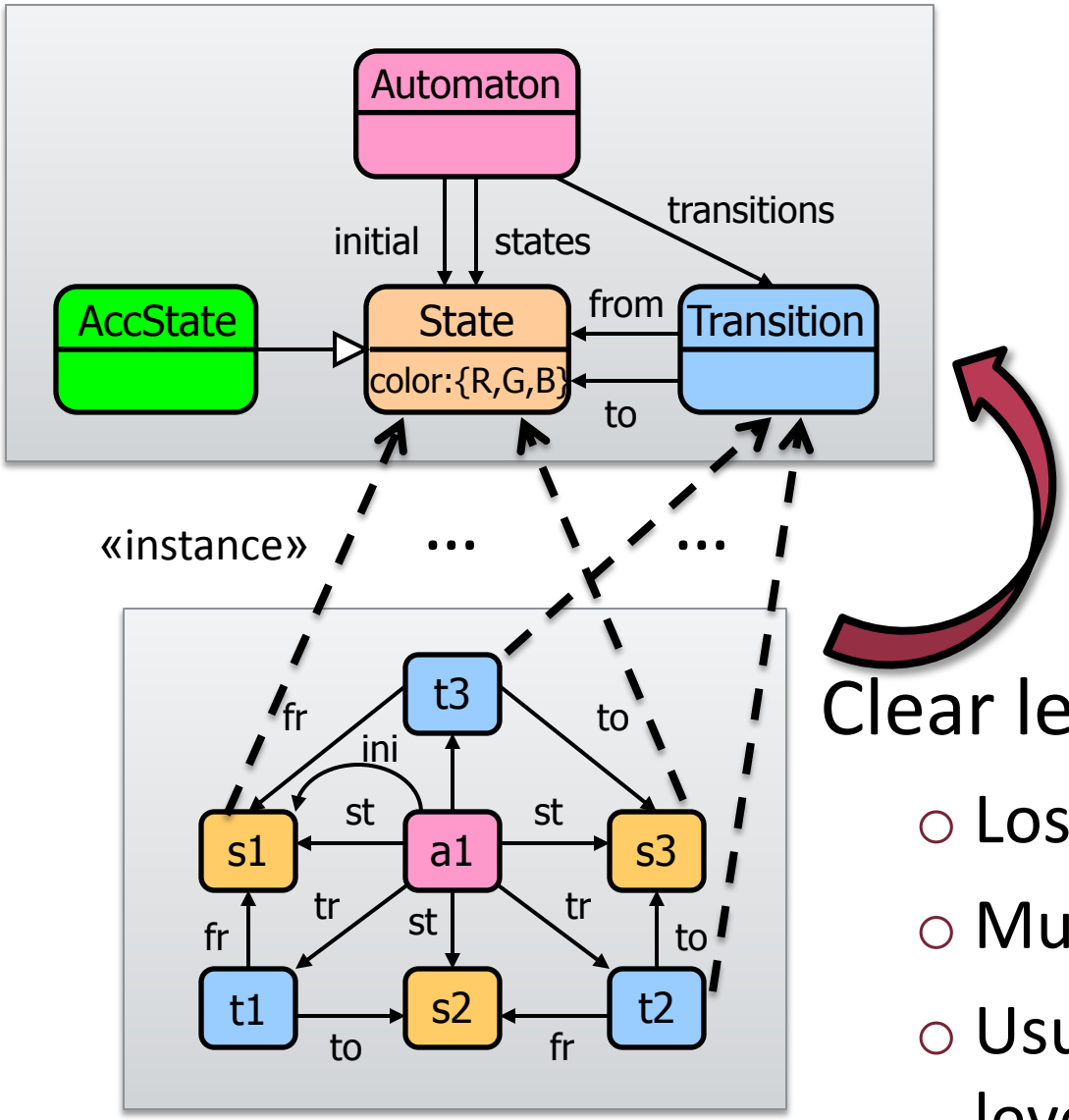
- Nodes
  - Film, Human, Novel, Psycho (film), Book, Man, Thriller, Work of Art, The Bourne Identity (novel), Genre, Robert Ludlum, Sir Alfred Hitchcock, this book here:

Demonstrated by the exercise:
- **Instantiation vs. subtyping**
- **Edge subtyping**
- **Metalevels**
- **Multi-level metamodeling**
- **Deep instantiation**



- Edges
  - written by, directed by, creator, subtype, instance

Metalevels



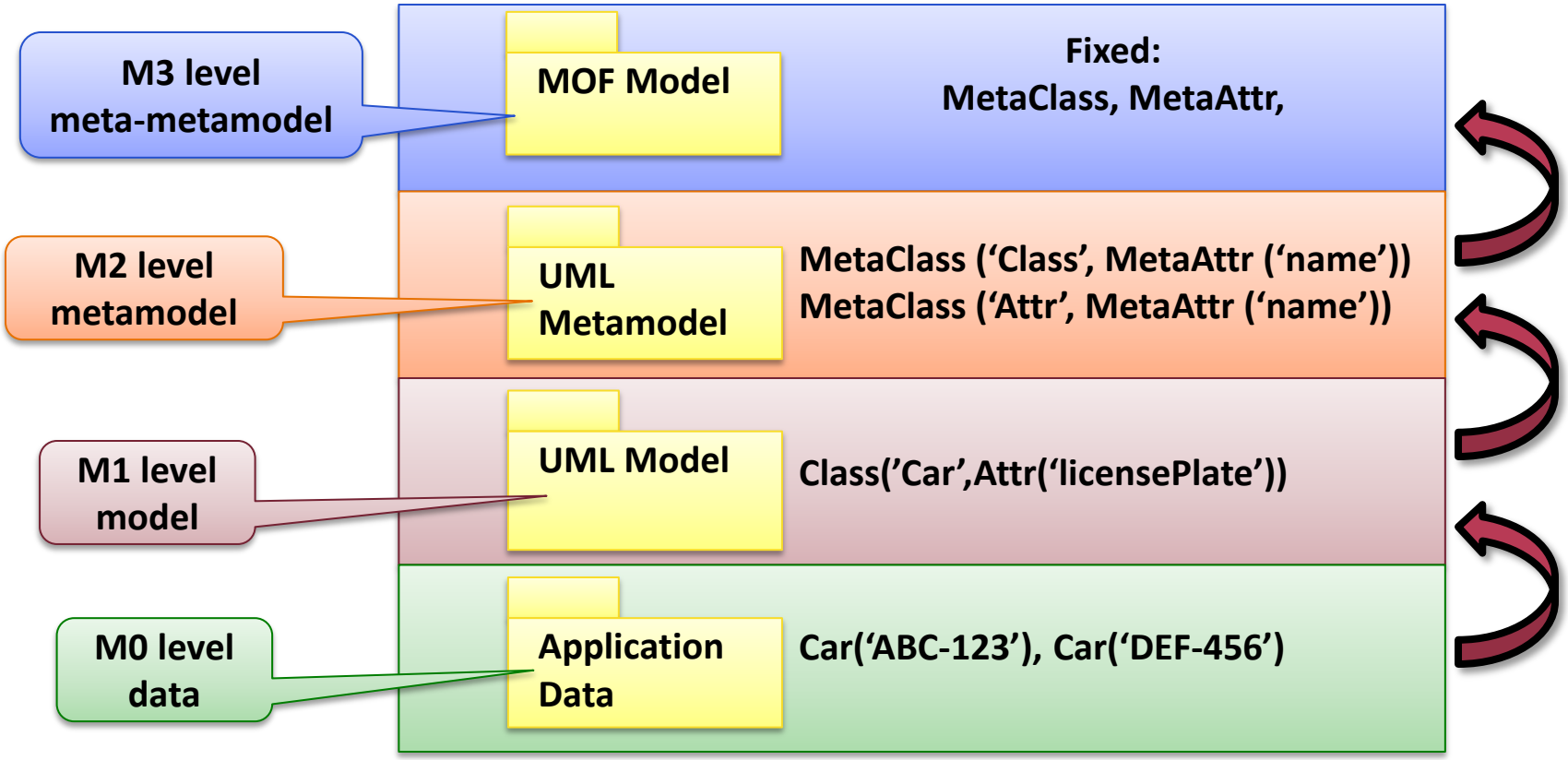
„Meta" relationship between models

Clear level separation:

- Loses some flexibility
- Much easier to understand
- Usually enough to keep two levels in mind at once

Metalevels in MOF

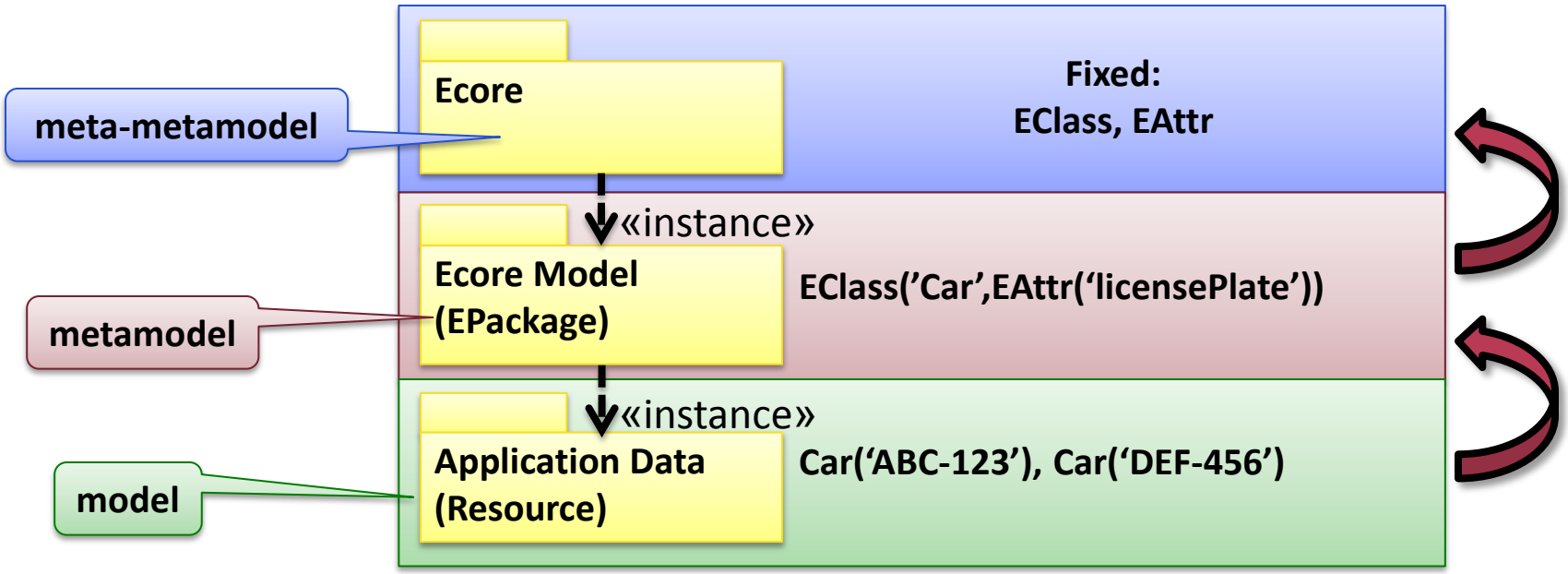
- OMG's MOF (Meta Object Facility)
  - 4-layer approach



  - Why exactly four levels?

Metalevels in other approaches

- EMF (Eclipse Modeling Framework)



- Multi-level metamodeling
  - VPM
  - Ontologies

SEMANTICS

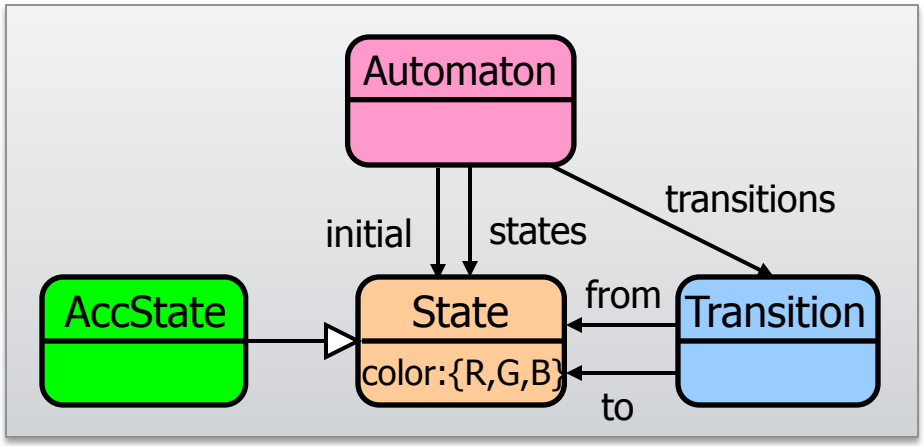
Semantics

- 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

- **Operational**
  - Modeling the operational behavior of language concepts
  - „interpreted”
  - e.g. defining how the finite automaton may change state at run-time
  - Sometimes dynamic features are introduced only for formalizing dynamic sematics
- **Denotational** (Translational)
  - translating concepts in one language to another language (called **semantic domain**)
  - „compiled”
  - E.g. explaining state machines as Petri-net

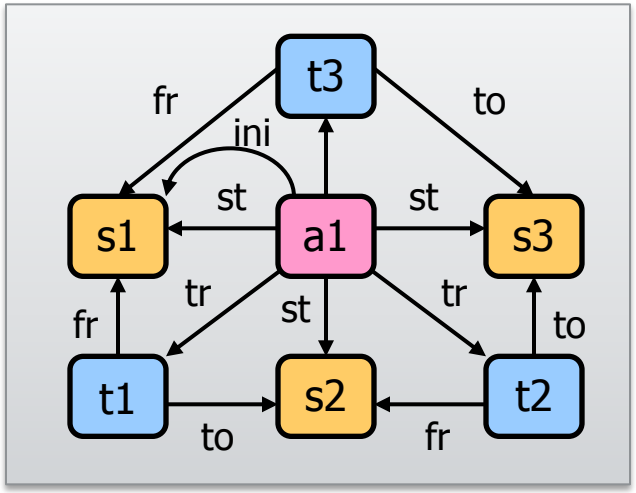
Example: Denotational semantics



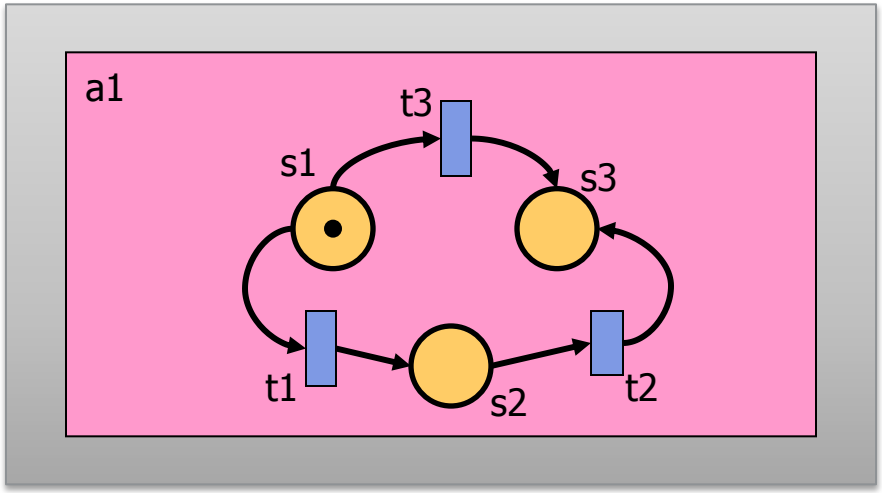
Meta (Language) level

Metamodel

Model level



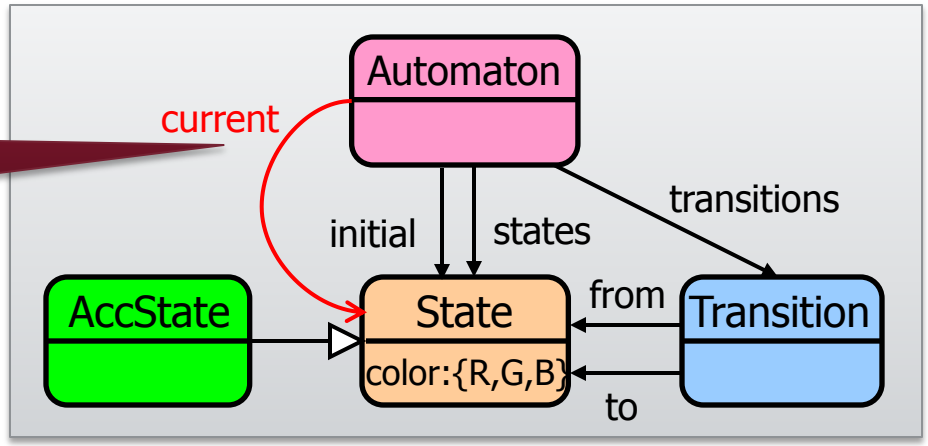
Abstract syntax



Semantic Domain

Example: Operational semantics

Dynamic feature

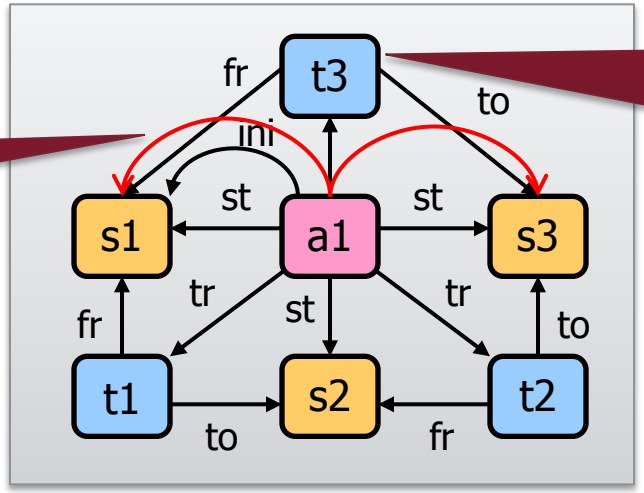


Meta (Language) level

Metamodel

(Instance) Model level

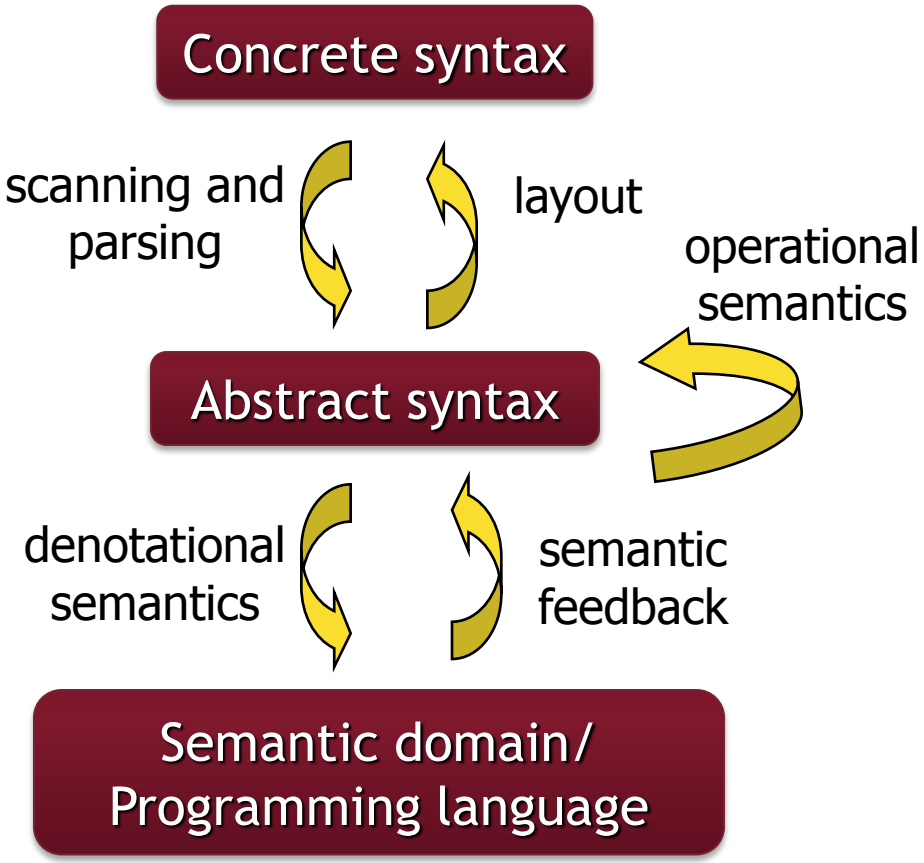
At first, 'current '= 'initial'



Possible evolution: 'current' is redirected along a transition

Model in abstract syntax

Relationship of models



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

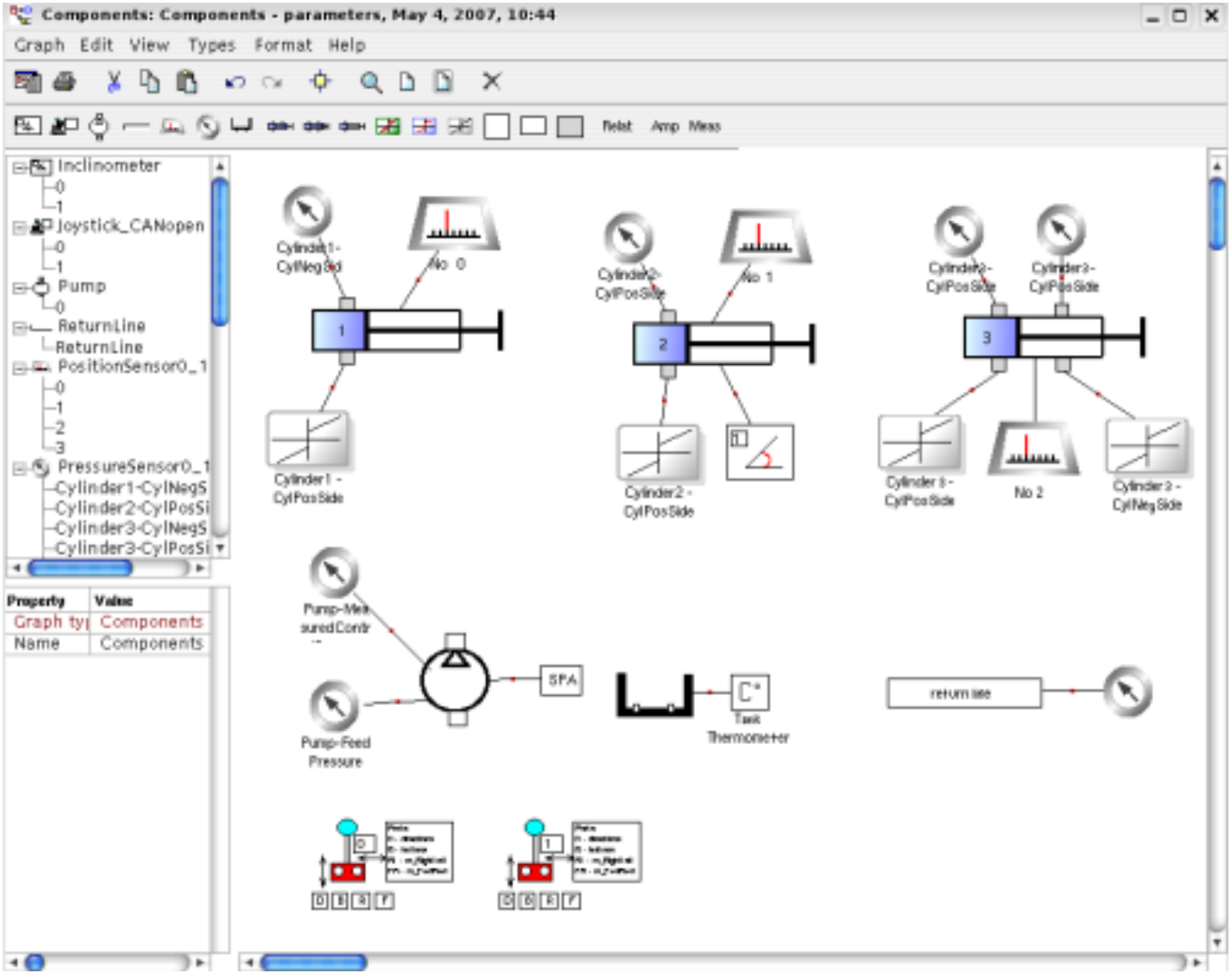
- Eclipse
  - EMF
  - Xtext/Xcore/etc.
- Microsoft
  - DSL Tools (Visual Studio)
- MetaCase
  - MetaEdit+
- JetBrains MPS

Language engineering (industry)

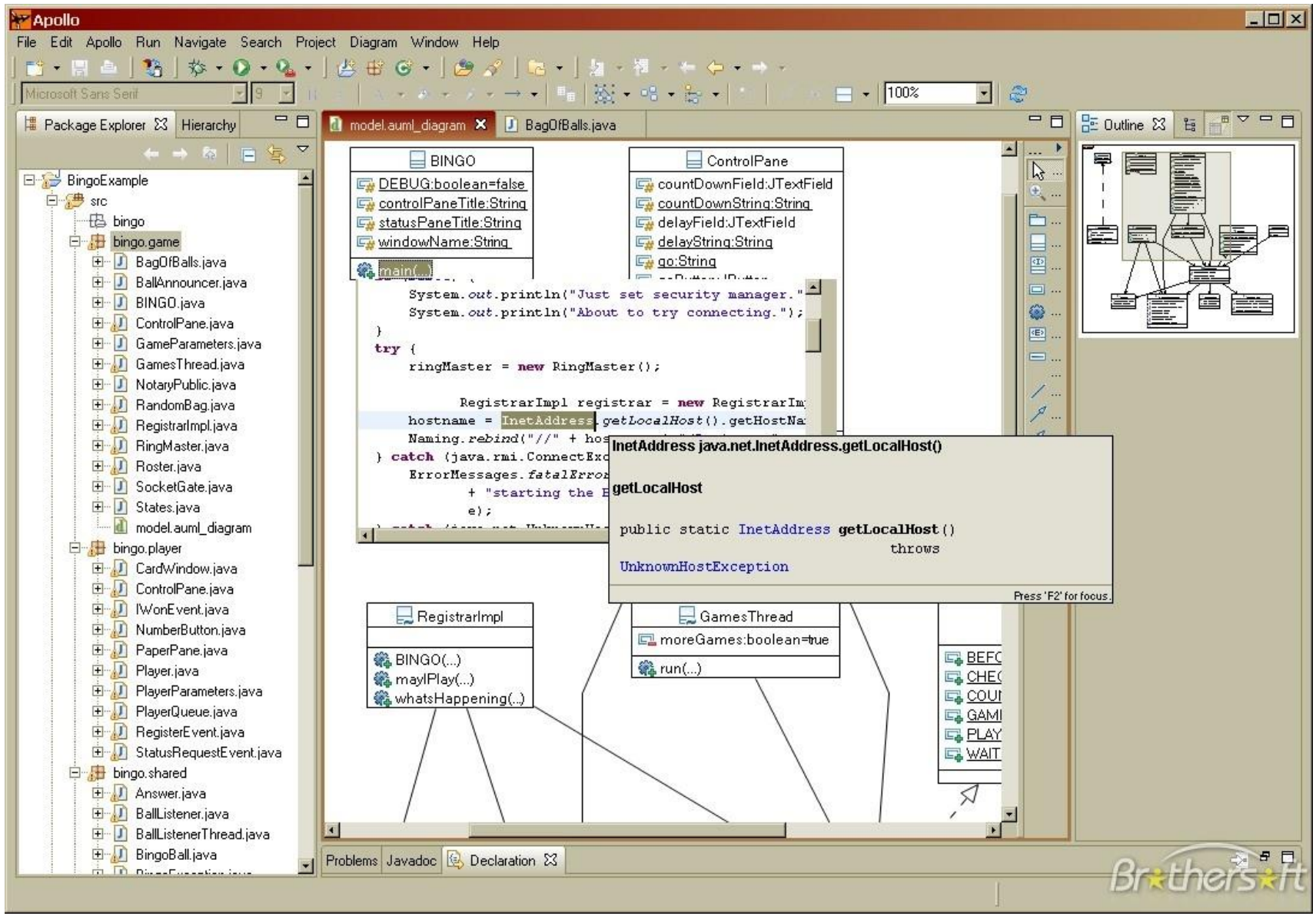
- GEMS, GME, ViatraDSM

Academia

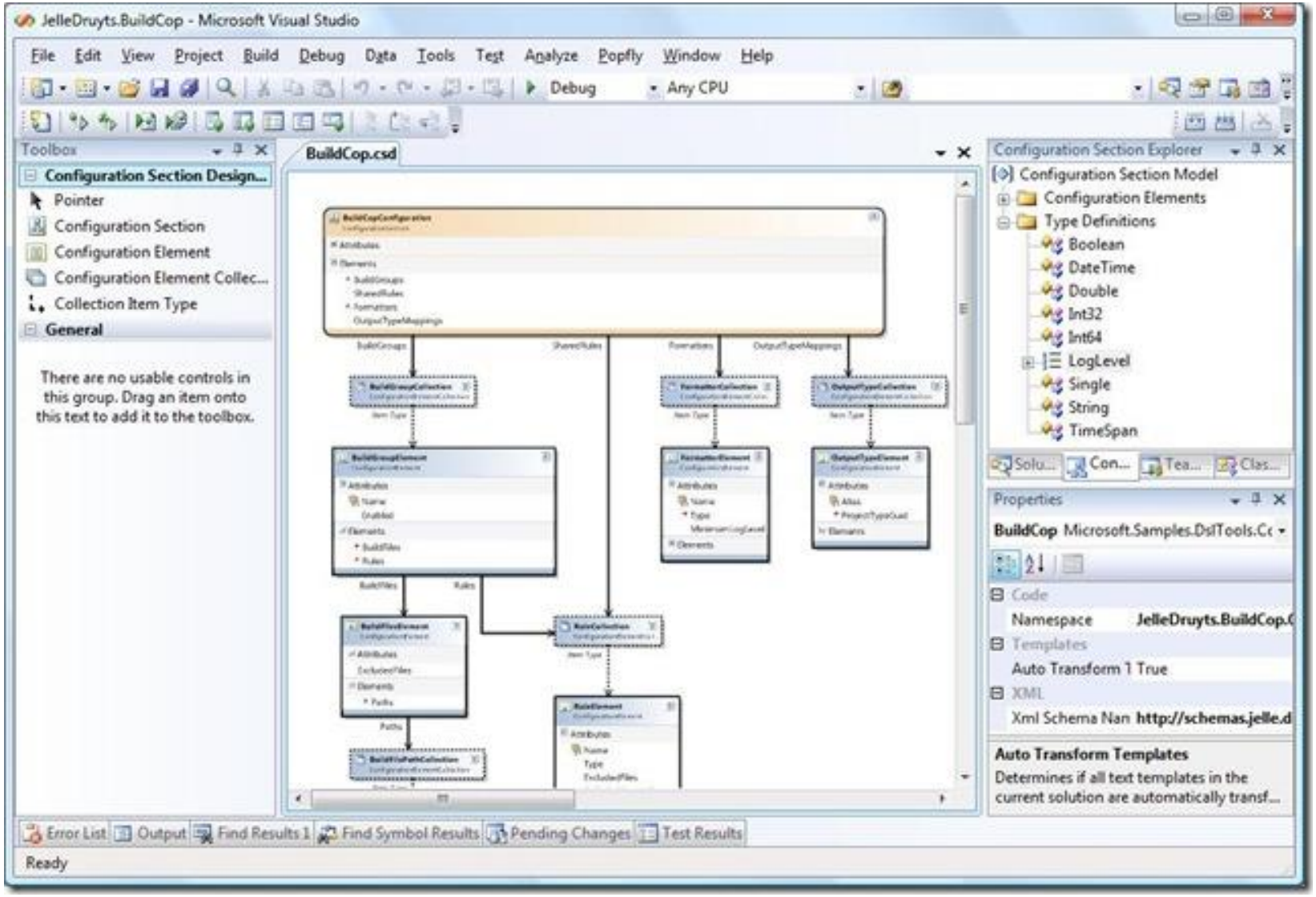
MetaEdit+


Components: Components - parameters, May 4, 2007, 10:44
Graph Edit View Types Format Help
Inclinometer
Joystick_CANopen
Pump
ReturnLine
PositionSensor0_1
PressureSensor0_1
Property
Value
Graph ty
Components
Name
Components
SPA
Tank
Thermometer
return line

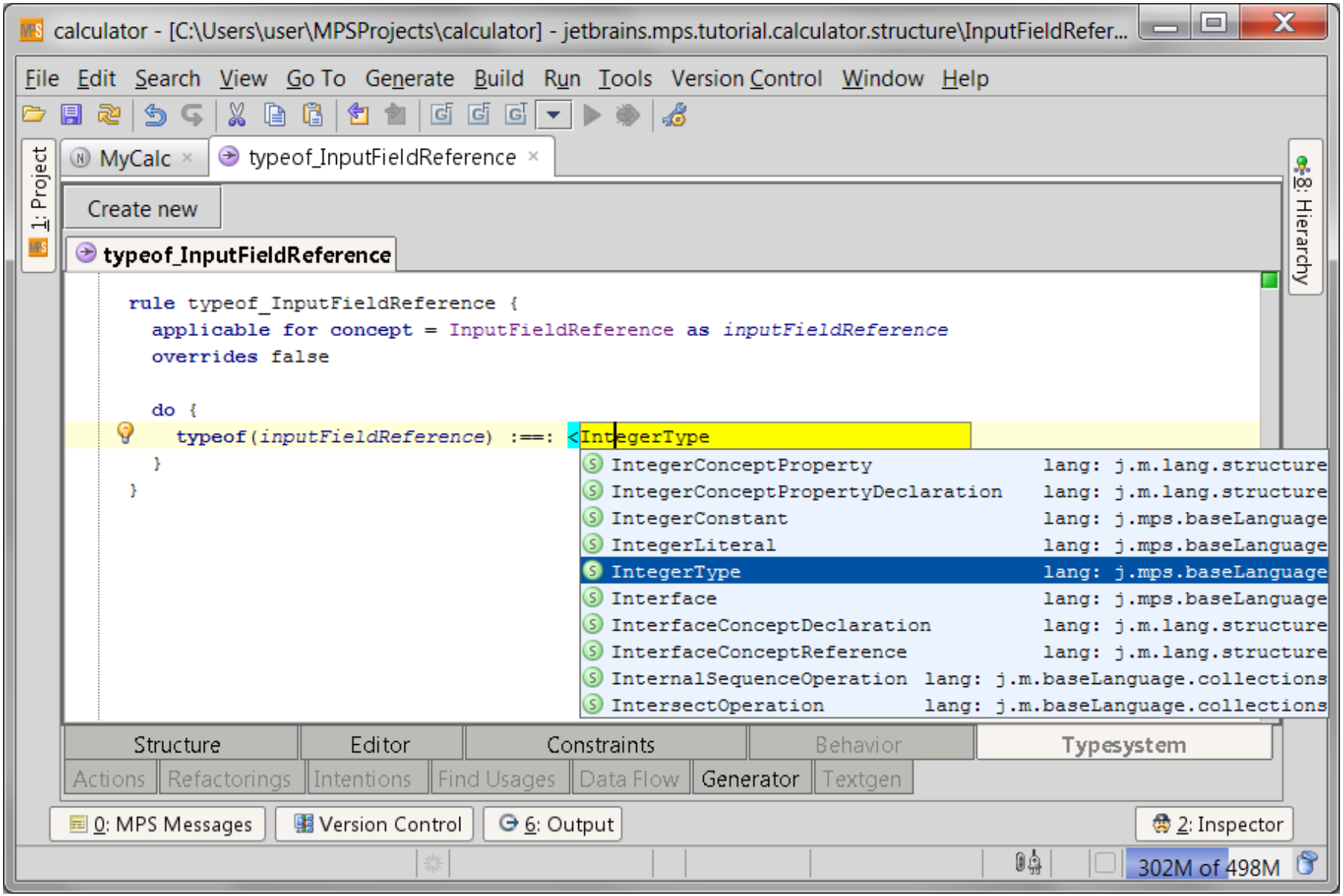
Eclipse GMF


Apollo
File Edit Apollo Run Navigate Search Project Diagram Window Help
Package Explorer
Hierarchy
BingoExample
src
bingo
bingo.game
BagOfBalls.java
BallAnnouncer.java
BINGO.java
ControlPane.java
GameParameters.java
GamesThread.java
NotaryPublic.java
RandomBag.java
RegistrarImpl.java
RingMaster.java
Roster.java
SocketGate.java
States.java
model.auml_diagram
bingo.player
CardWindow.java
ControlPane.java
IWonEvent.java
NumberButton.java
PaperPane.java
Player.java
PlayerParameters.java
PlayerQueue.java
RegisterEvent.java
StatusRequestEvent.java
bingo.shared
Answer.java
BallListener.java
BallListenerThread.java
BingoBall.java
model.auml_diagram
BagOfBalls.java
BINGO
DEBUG:boolean=false
controlPaneTitle:String
statusPaneTitle:String
windowName:String
main(...)
ControlPane
countDownField:JTextField
countDownString:String
delayField:JTextField
delayString:String
go:String
System.out.println("Just set security manager.");
System.out.println("About to try connecting.");
try {
ringMaster = new RingMaster();
RegistrarImpl registrar = new RegistrarIm
hostname = InetAddress.getLocalHost().getHostNa
Naming.rebind("//" + hos
} catch (java.rmi.ConnectExc
ErrorMessages.fatalError
+ "starting the R
e);
InetAddress java.net.InetAddress.getLocalHost()
getLocalHost
public static InetAddress getLocalHost()
throws
UnknownHostException
Press 'F2' for focus.
RegistrarImpl
BINGO(...)
mayIPlay(...)
whatsHappening(...)
GamesThread
moreGames:boolean=true
run(...)
BEFC
CHEC
COUI
GAMI
PLAY
WAIT
Outline
Problems
Javadoc
Declaration
Brothers-ft

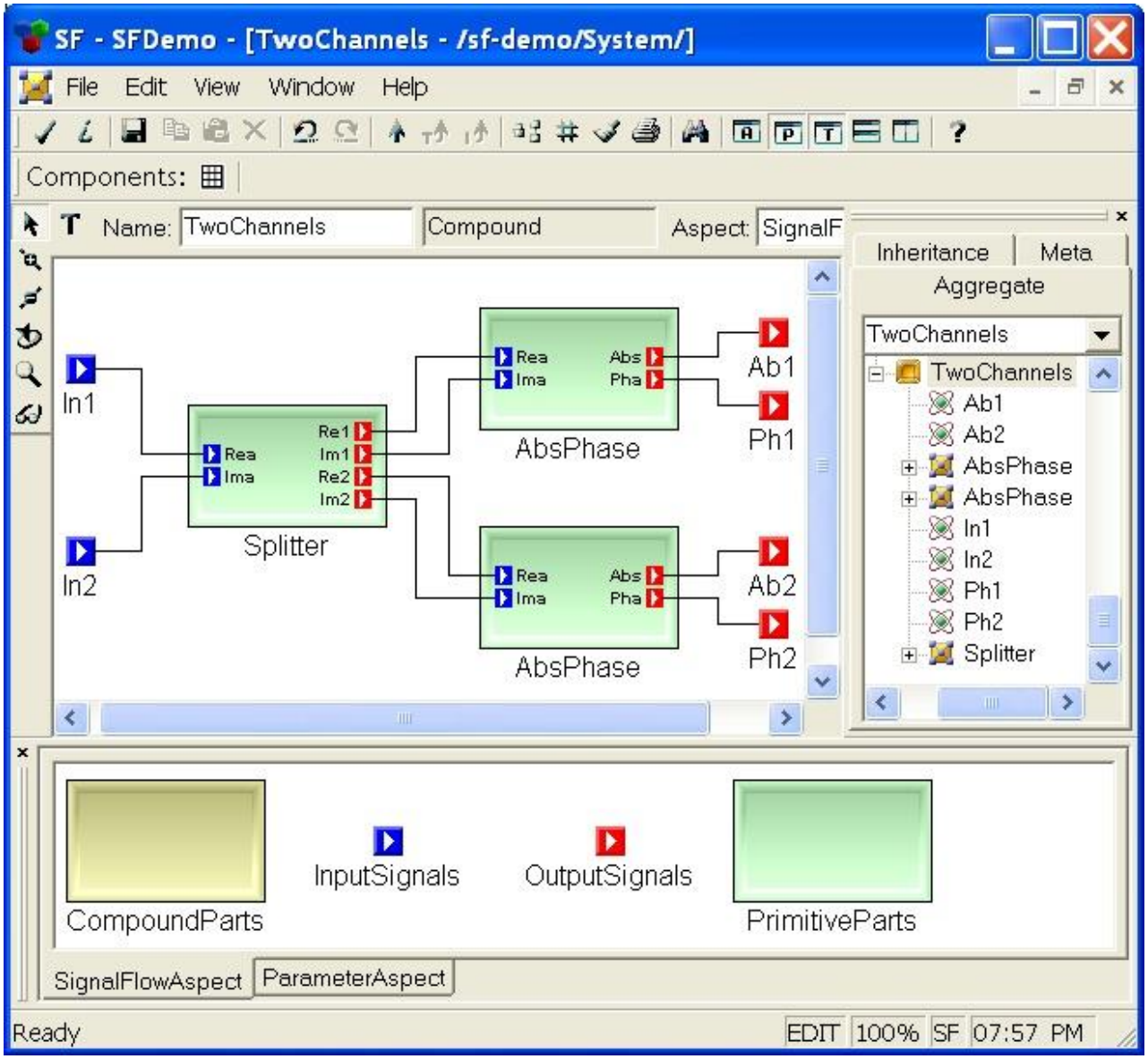
Microsoft DSL Tools



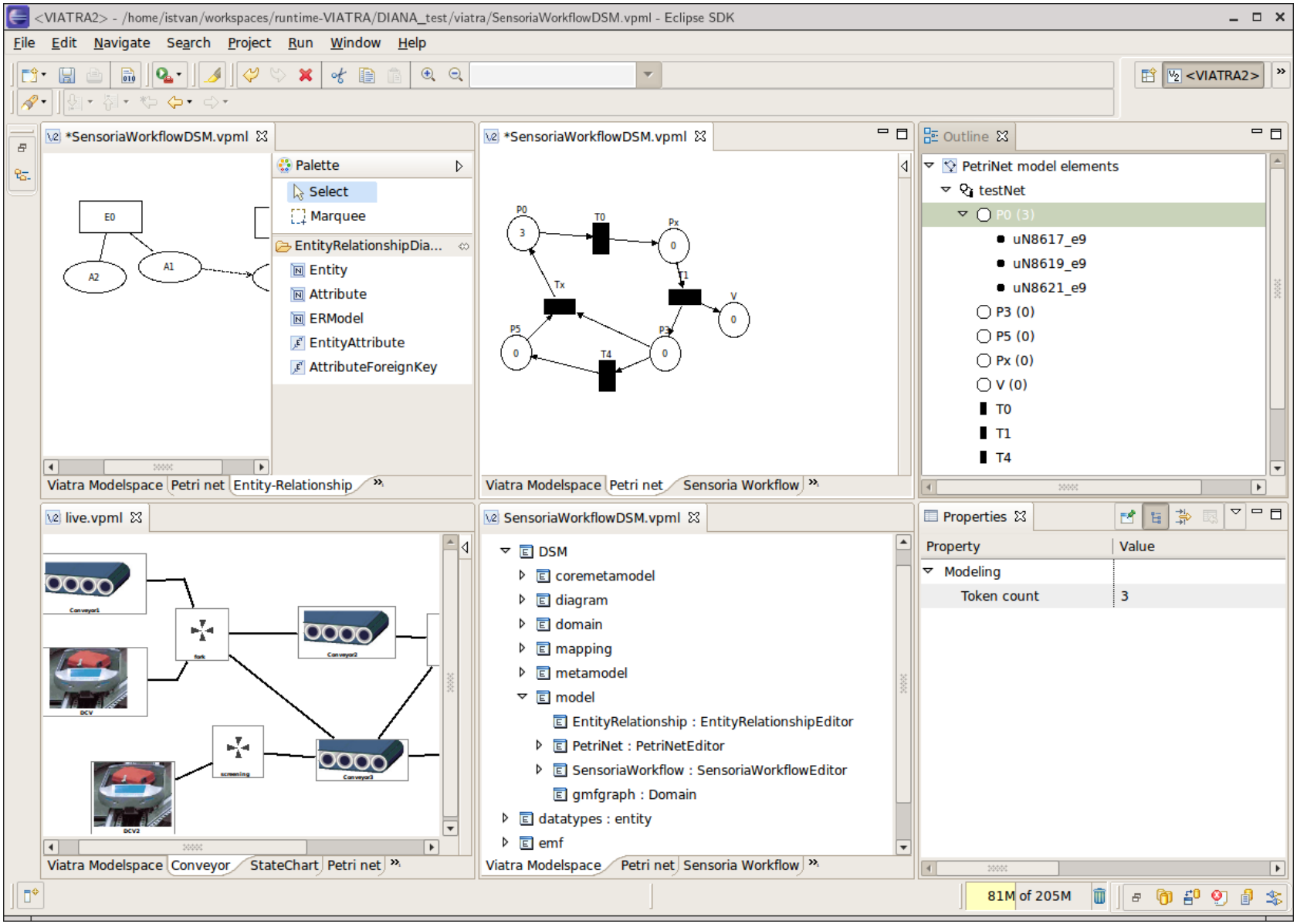
MPS


calculator - [C:\Users\user\MPSProjects\calculator] - jetbrains.mps.tutorial.calculator.structure\InputFieldRefer...
File Edit Search View Go To Generate Build Run Tools Version Control Window Help
1: Project
MyCalc
typeof_InputFieldReference
8: Hierarchy
Create new
typeof_InputFieldReference
rule typeof_InputFieldReference {
applicable for concept = InputFieldReference as inputFieldReference
overrides false
do {
typeof(inputFieldReference) :==: <IntegerType
}
}
IntegerConceptProperty lang: j.m.lang.structure
IntegerConceptPropertyDeclaration lang: j.m.lang.structure
IntegerConstant lang: j.mps.baseLanguage
IntegerLiteral lang: j.mps.baseLanguage
IntegerType lang: j.mps.baseLanguage
Interface lang: j.mps.baseLanguage
InterfaceConceptDeclaration lang: j.m.lang.structure
InterfaceConceptReference lang: j.m.lang.structure
InternalSequenceOperation lang: j.m.baseLanguage.collections
IntersectOperation lang: j.m.baseLanguage.collections
Structure
Editor
Constraints
Behavior
Typesystem
Actions
Refactorings
Intentions
Find Usages
Data Flow
Generator
Textgen
0: MPS Messages
Version Control
6: Output
2: Inspector
302M of 498M

GME


SF - SFDemo - [TwoChannels - /sf-demo/System/]
File Edit View Window Help
Components:
Name: TwoChannels
Compound
Aspect: SignalF
Inheritance
Meta
Aggregate
TwoChannels
In1
In2
Rea
Ima
Re1
Im1
Re2
Im2
Splitter
Abs
Pha
AbsPhase
Ab1
Ph1
Ab2
Ph2
Ab2
Ph2
Splitter
CompoundParts
InputSignals
OutputSignals
PrimitiveParts
SignalFlowAspect
ParameterAspect
Ready
EDIT
100%
SF
07:57 PM

ViatraDSM


<VIATRA2> - /home/istvan/workspaces/runtime-VIATRA/DIANA_test/viatra/SensoriaWorkflowDSM.vpml - Eclipse SDK
File Edit Navigate Search Project Run Window Help
*SensoriaWorkflowDSM.vpml
Palette
Select
Marquee
EntityRelationshipDia...
Entity
Attribute
ERModel
EntityAttribute
AttributeForeignKey
E0
A2
A1
Viatra Modelspace Petri net Entity-Relationship
*SensoriaWorkflowDSM.vpml
P0
T0
Px
Tx
T1
V
P5
P3
T4
Viatra Modelspace Petri net Sensoria Workflow
Outline
PetriNet model elements
testNet
P0 (3)
uN8617_e9
uN8619_e9
uN8621_e9
P3 (0)
P5 (0)
Px (0)
V (0)
T0
T1
T4
live.vpml
Viatra Modelspace Conveyor StateChart Petri net
SensoriaWorkflowDSM.vpml
DSM
coremetamodel
diagram
domain
mapping
metamodel
model
EntityRelationship : EntityRelationshipEditor
PetriNet : PetriNetEditor
SensoriaWorkflow : SensoriaWorkflowEditor
gmfgraph : Domain
datatypes : entity
emf
Viatra Modelspace Petri net Sensoria Workflow
Properties
Property
Value
Modeling
Token count
3
81M of 205M

DSM 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

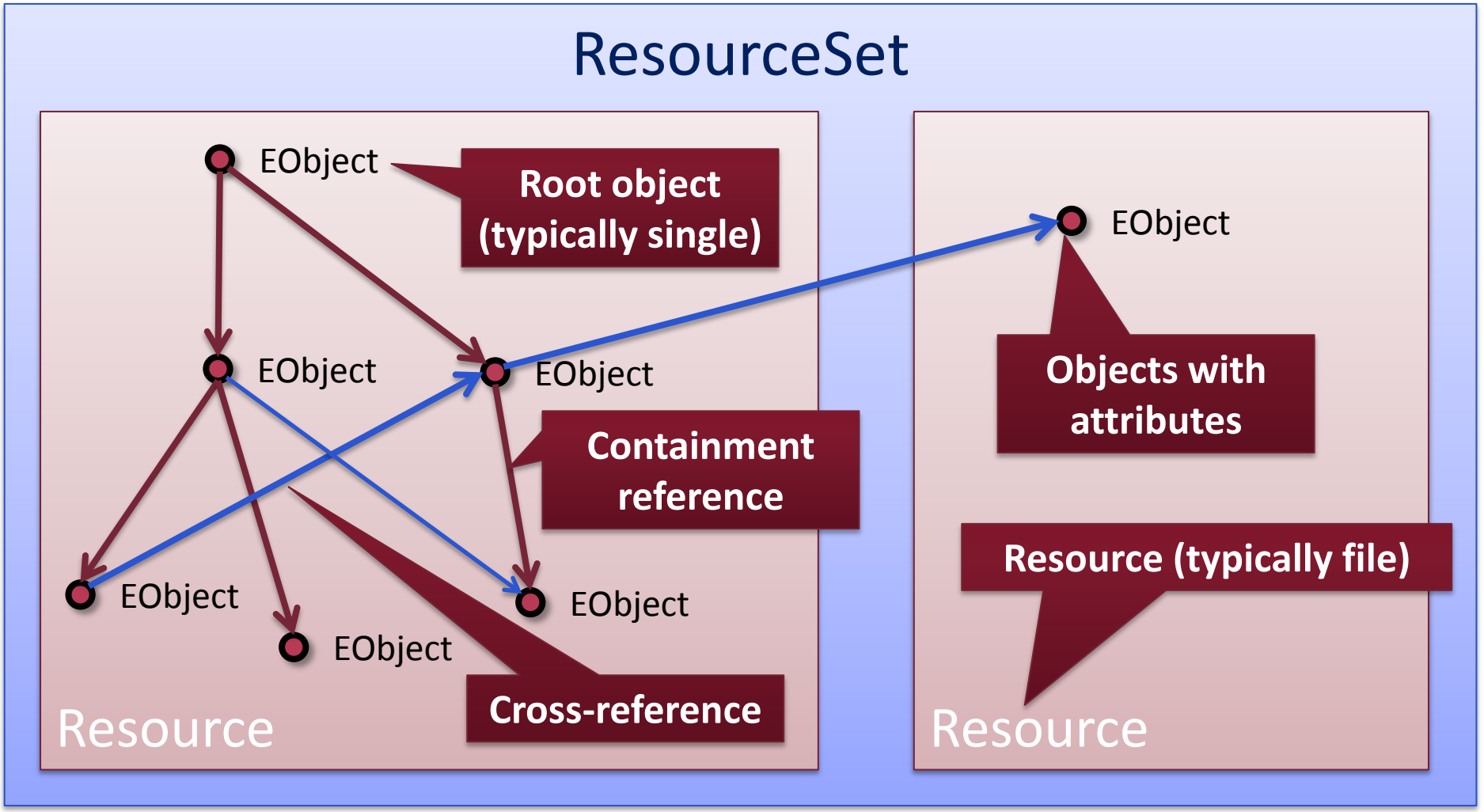
ECLIPSE MODELING FRAMEWORK

What does EMF provide?

- EMF = Eclipse Modeling Framework
  - Reflective Metamodeling Core (Ecore ➔ MOF 2.0)
  - Support for Domain Specific Languages
  - Editing Support (Notification, Undo, Commands)
  - Basic Editor Support
  - XMI Serialization, DB Persistence
  - Eclipse Integration

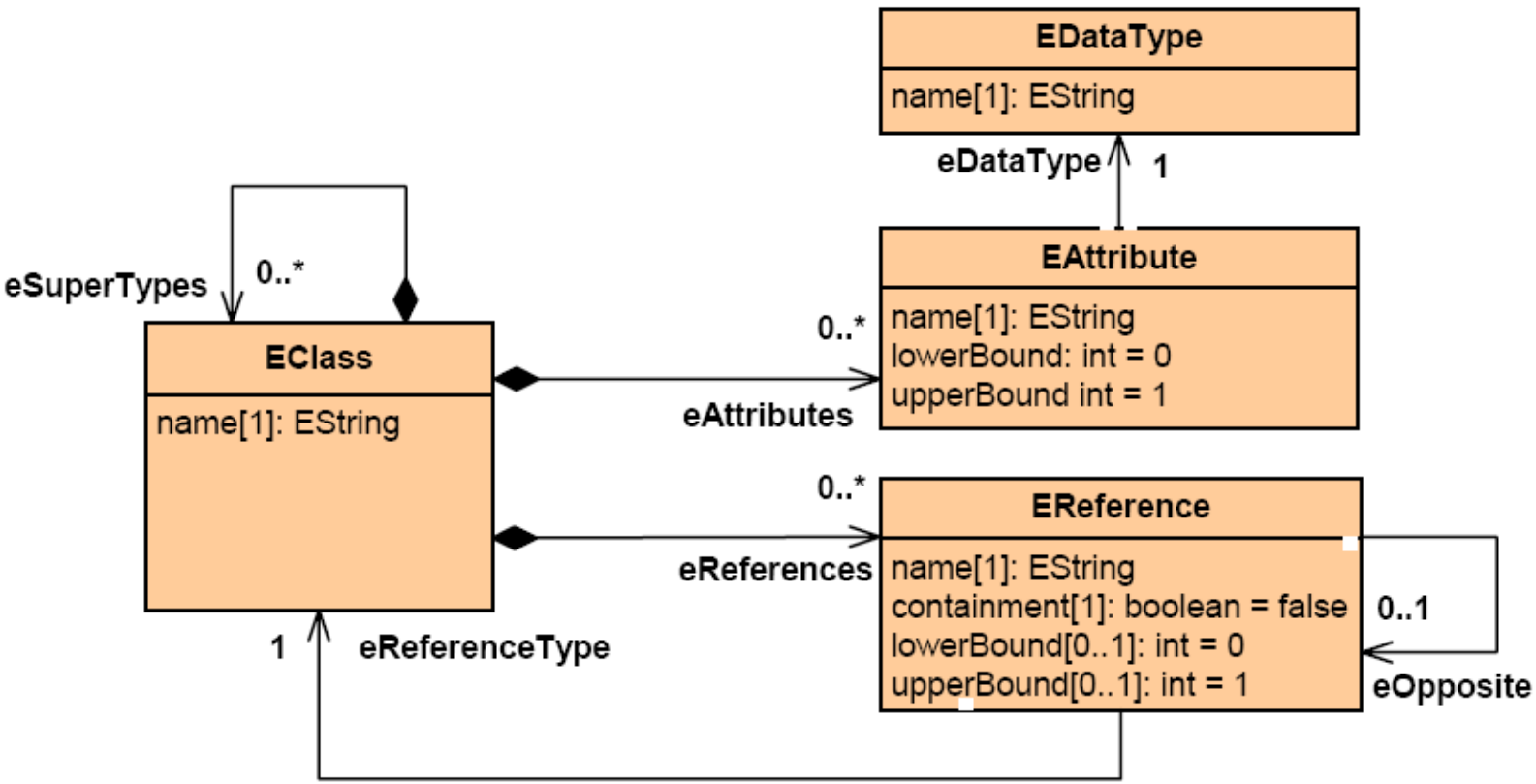
EMF model structure

- Containment hierarchy



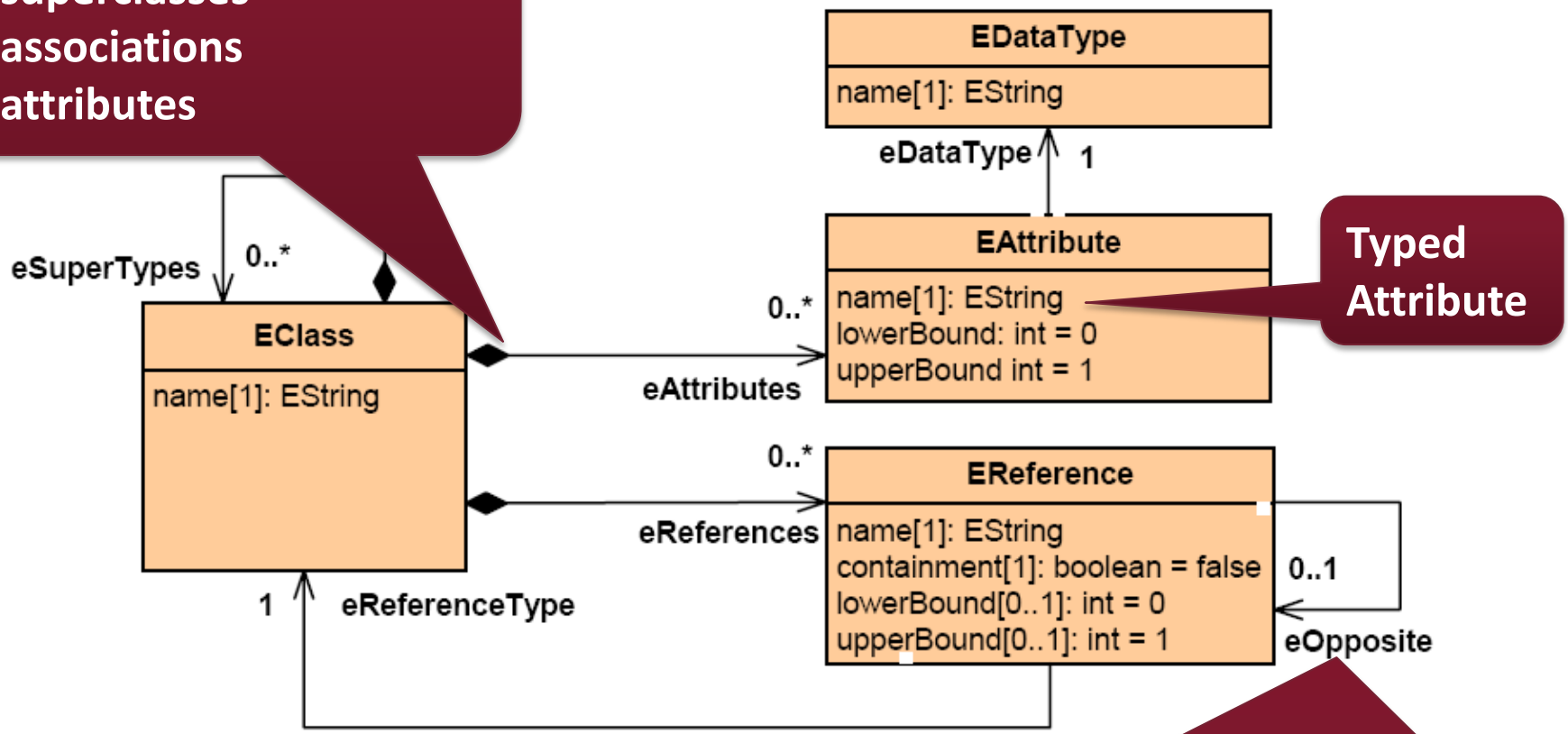
ECORE METAMODELLING

Core Ecore constructs



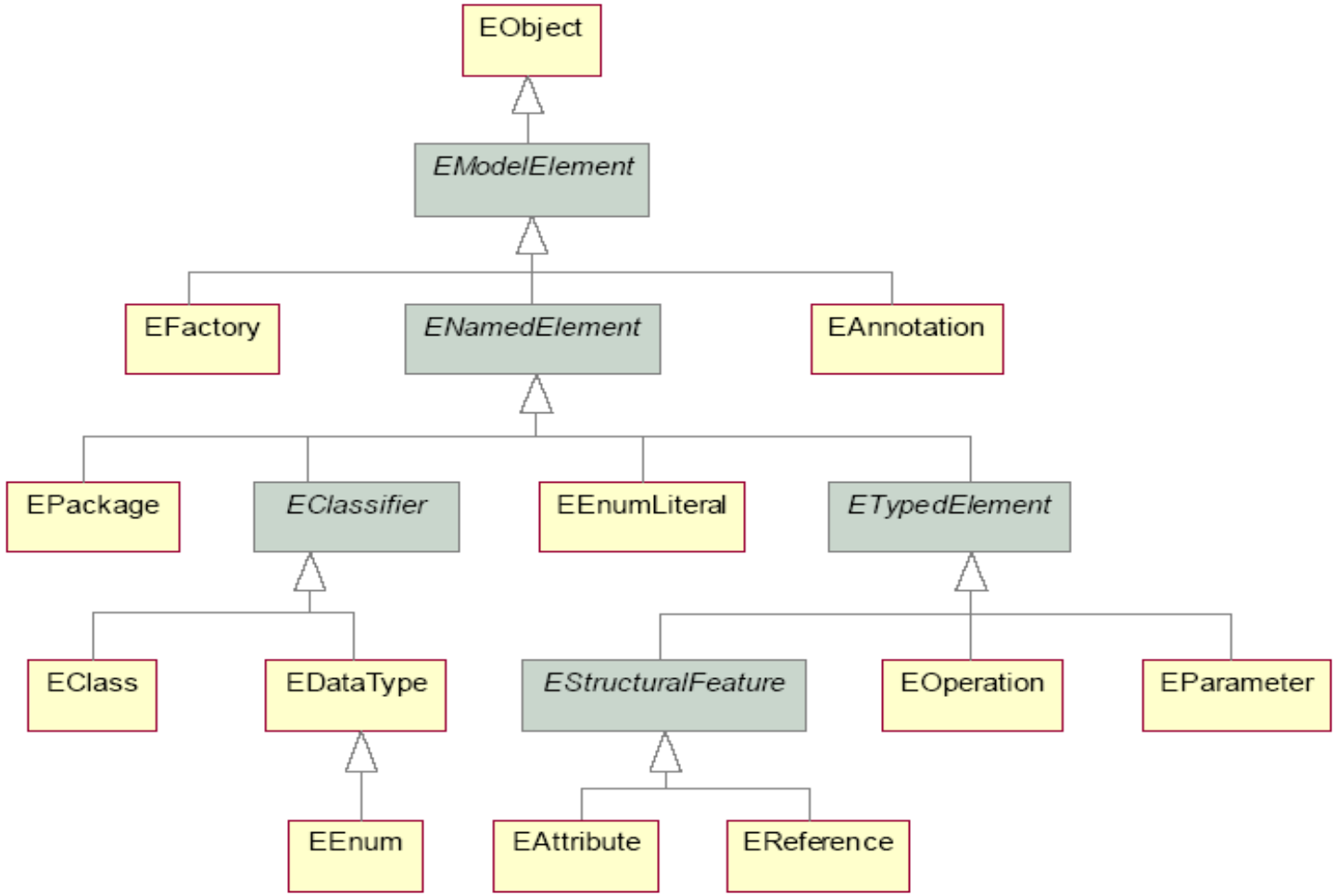
Core Ecore constructs

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

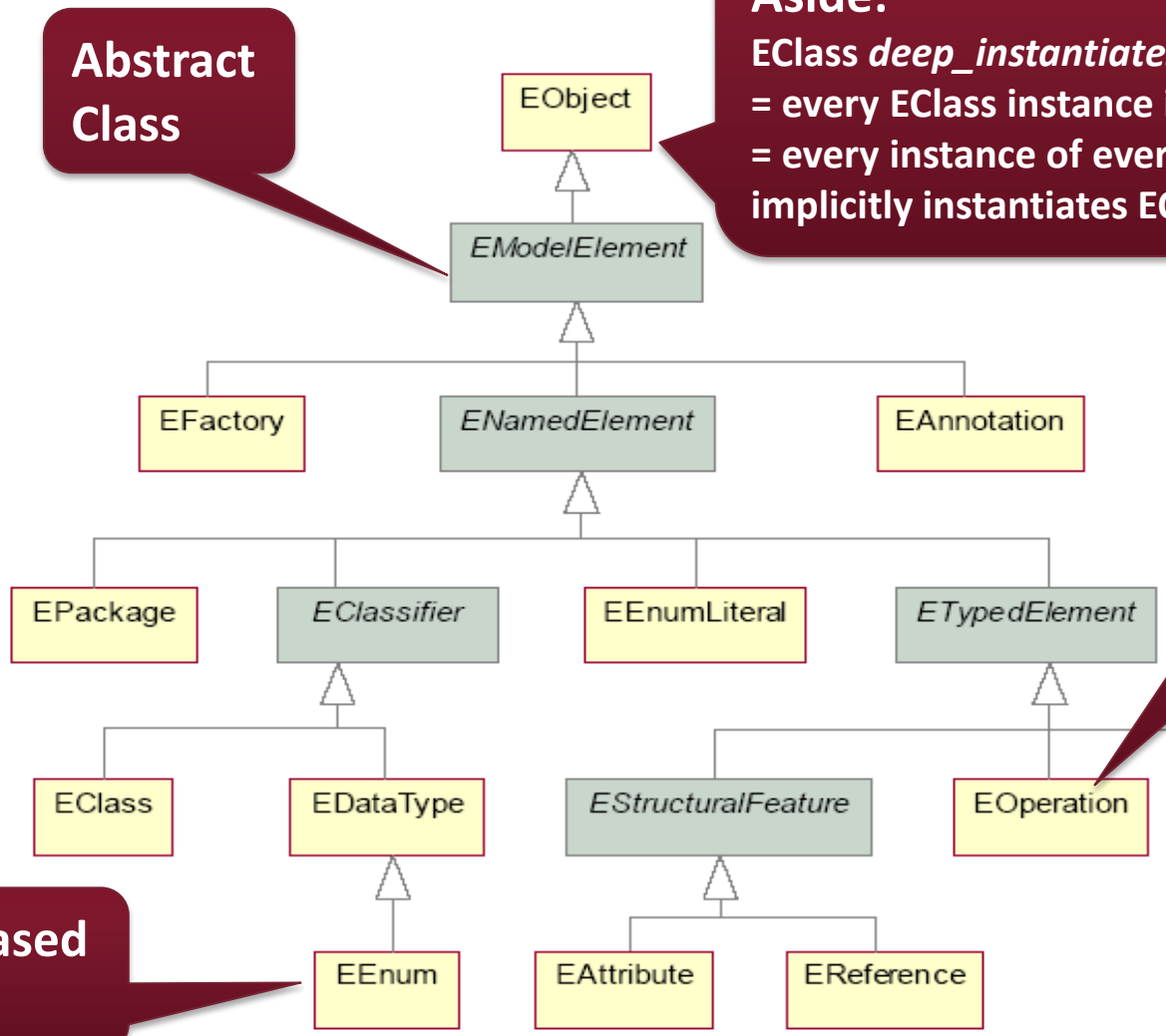


Unidirectional (binary) relation (Association)
- **typed**
- **optional inverse end**
- **multiplicities**

Complete Ecore hiearchy



Complete Ecore hiearchy



Abstract Class

Aside:
EClass *deep_instantiates(2)* EObject
= every EClass instance implicitly subtypes EObject
= every instance of every instance of EClass implicitly instantiates EObject

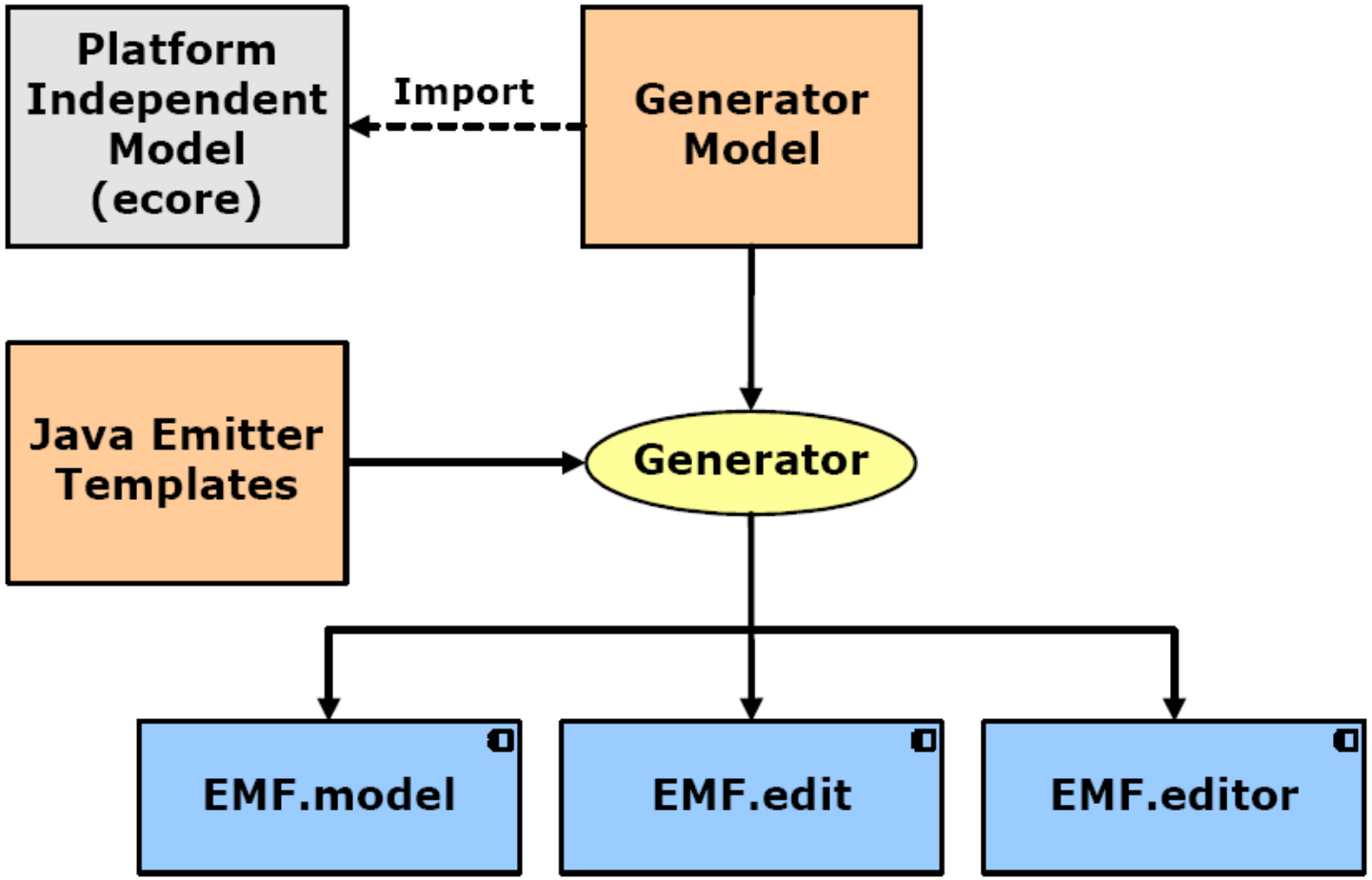
Methods connected to the EClasses

EMF-based Enums

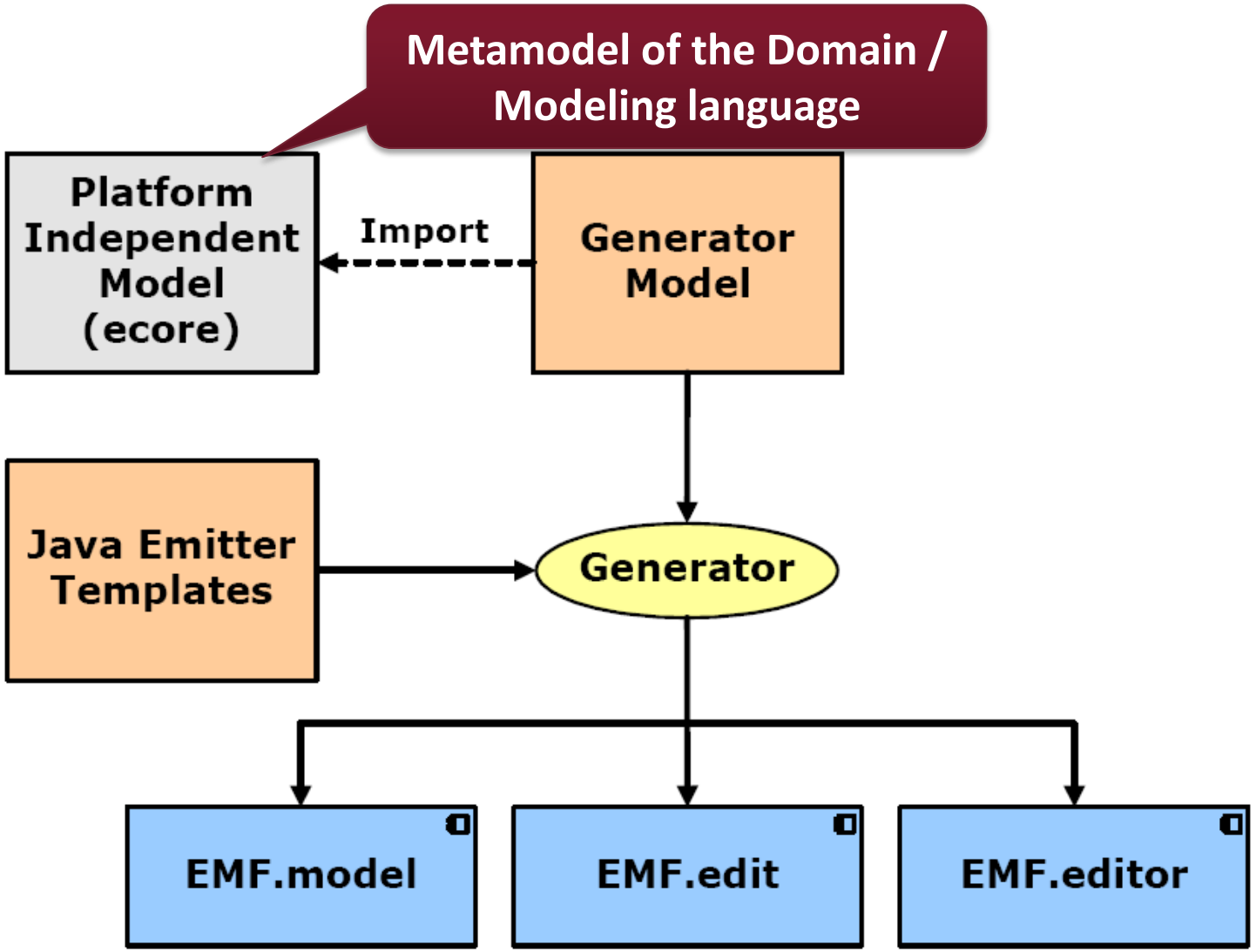
Parameter for the Eoperation

DEFINING A DSM
…THE EMF WAY

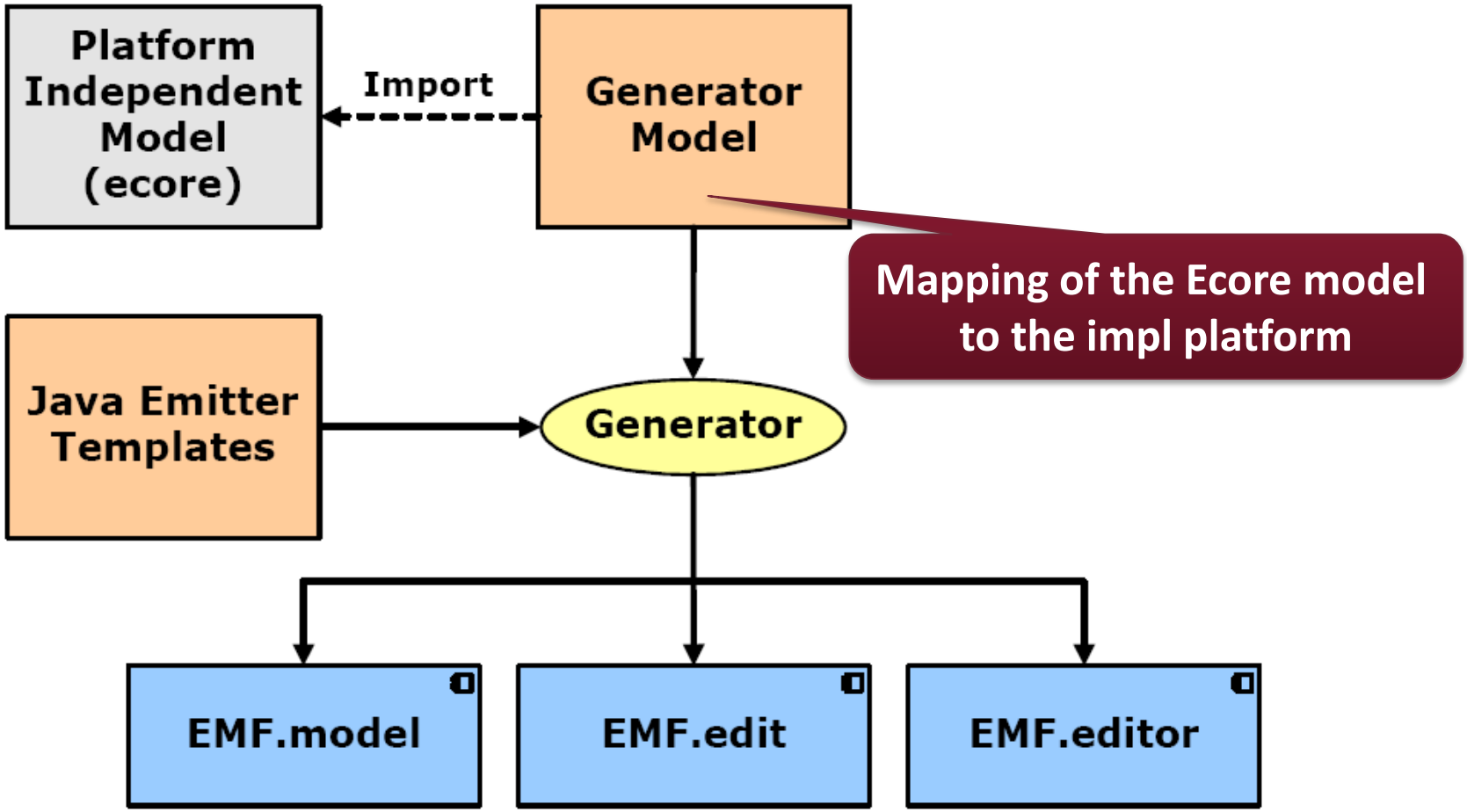
The EMF Toolkit



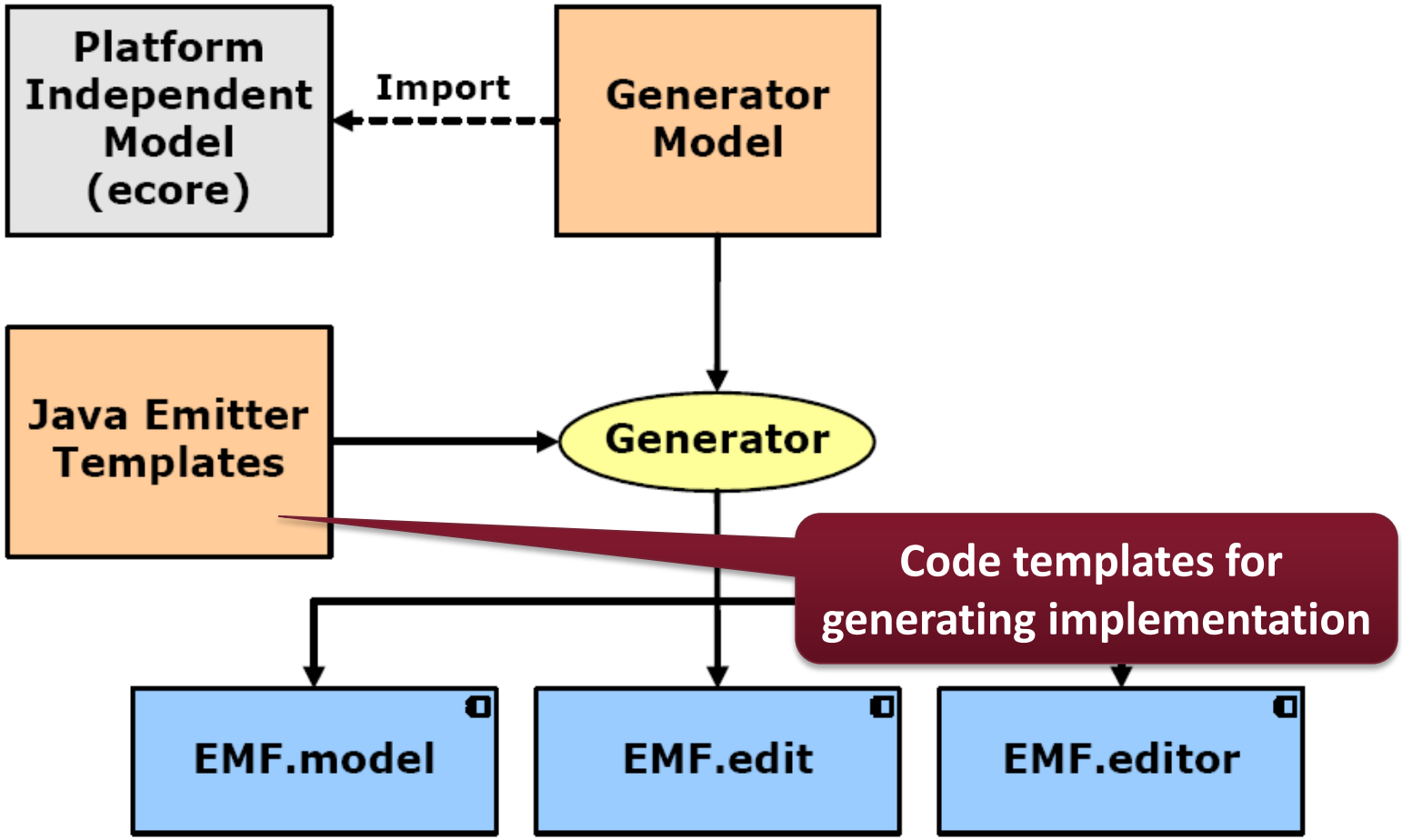
The EMF Toolkit



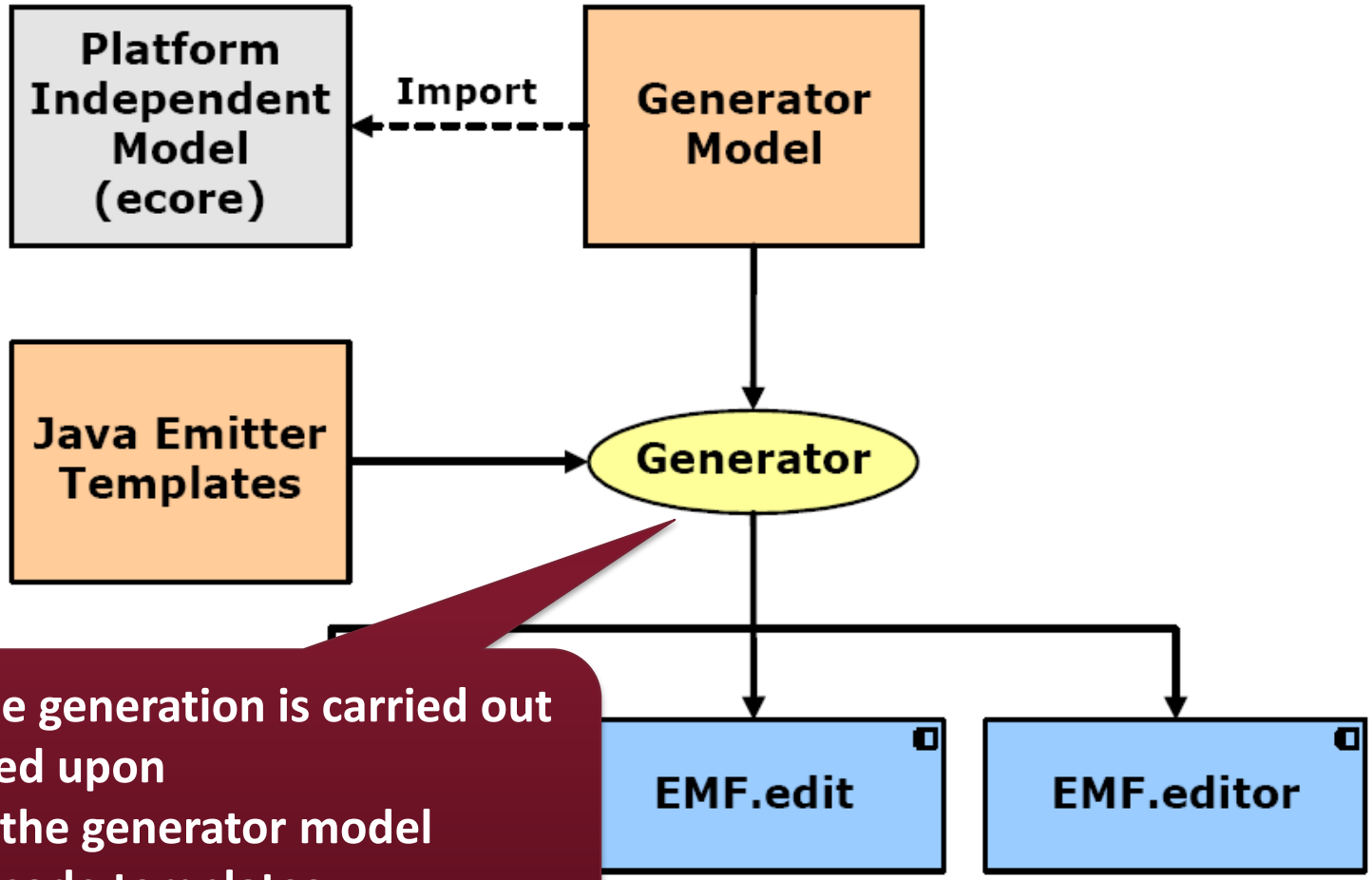
The EMF Toolkit



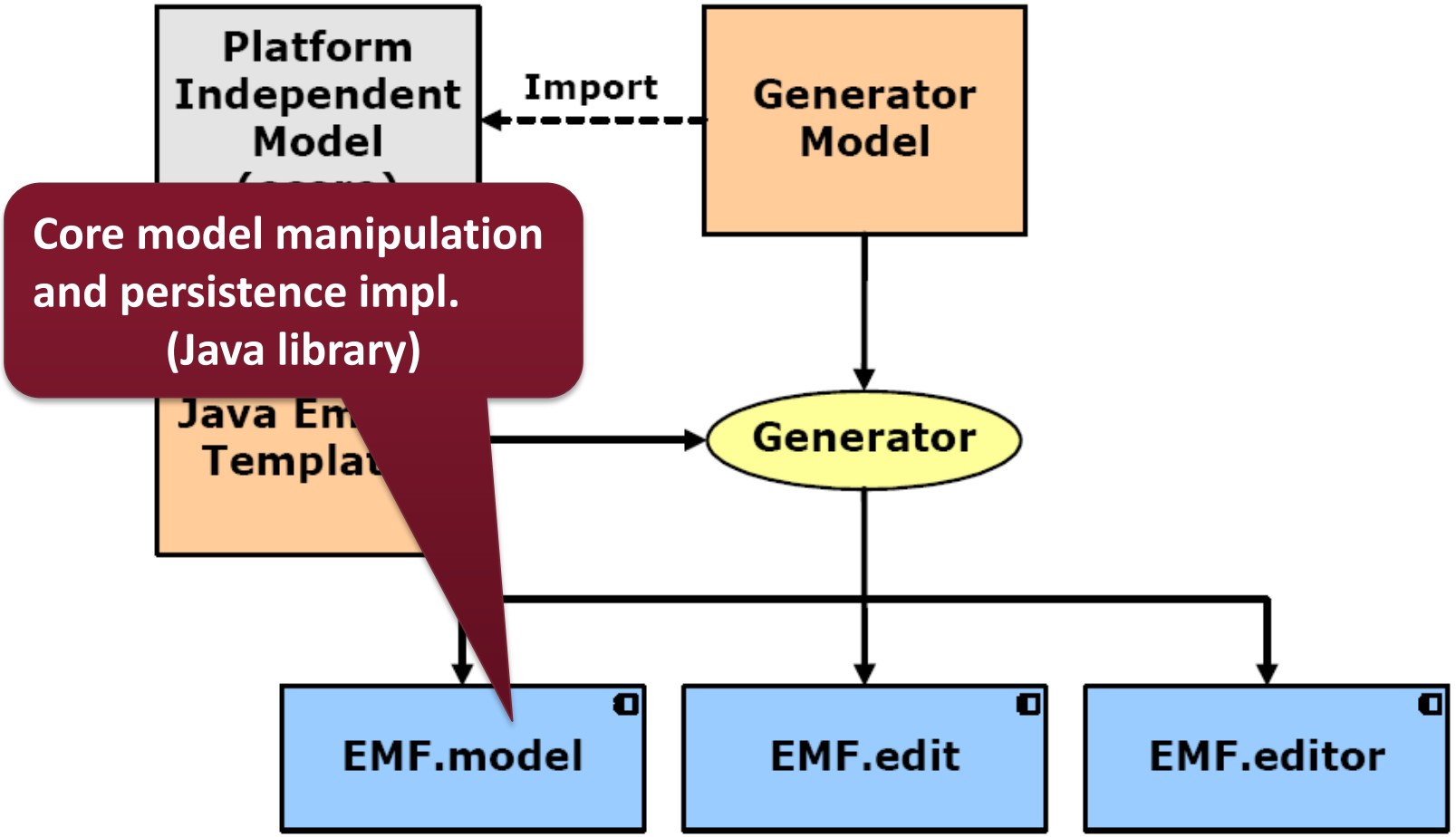
The EMF Toolkit



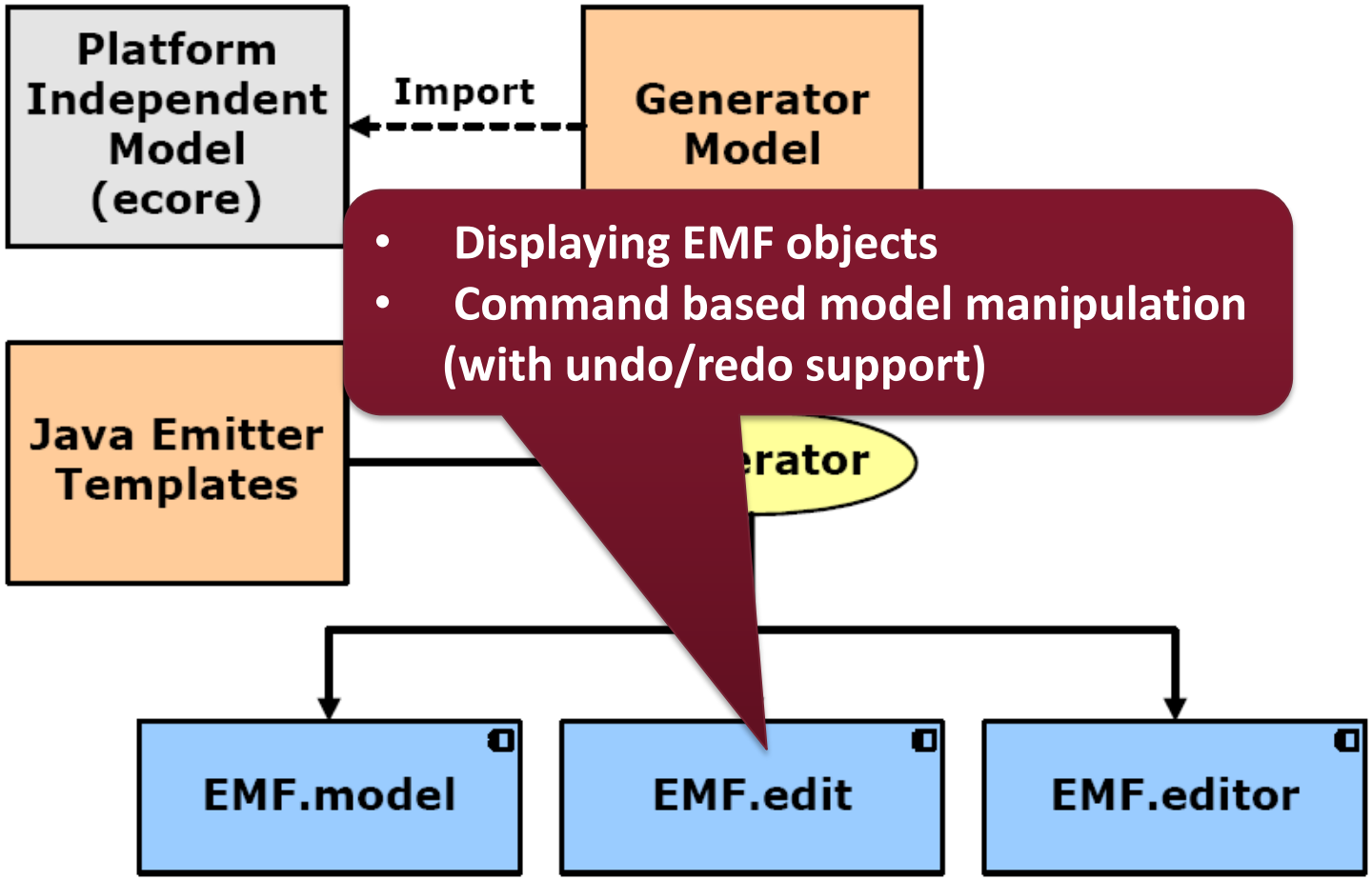
The EMF Toolkit



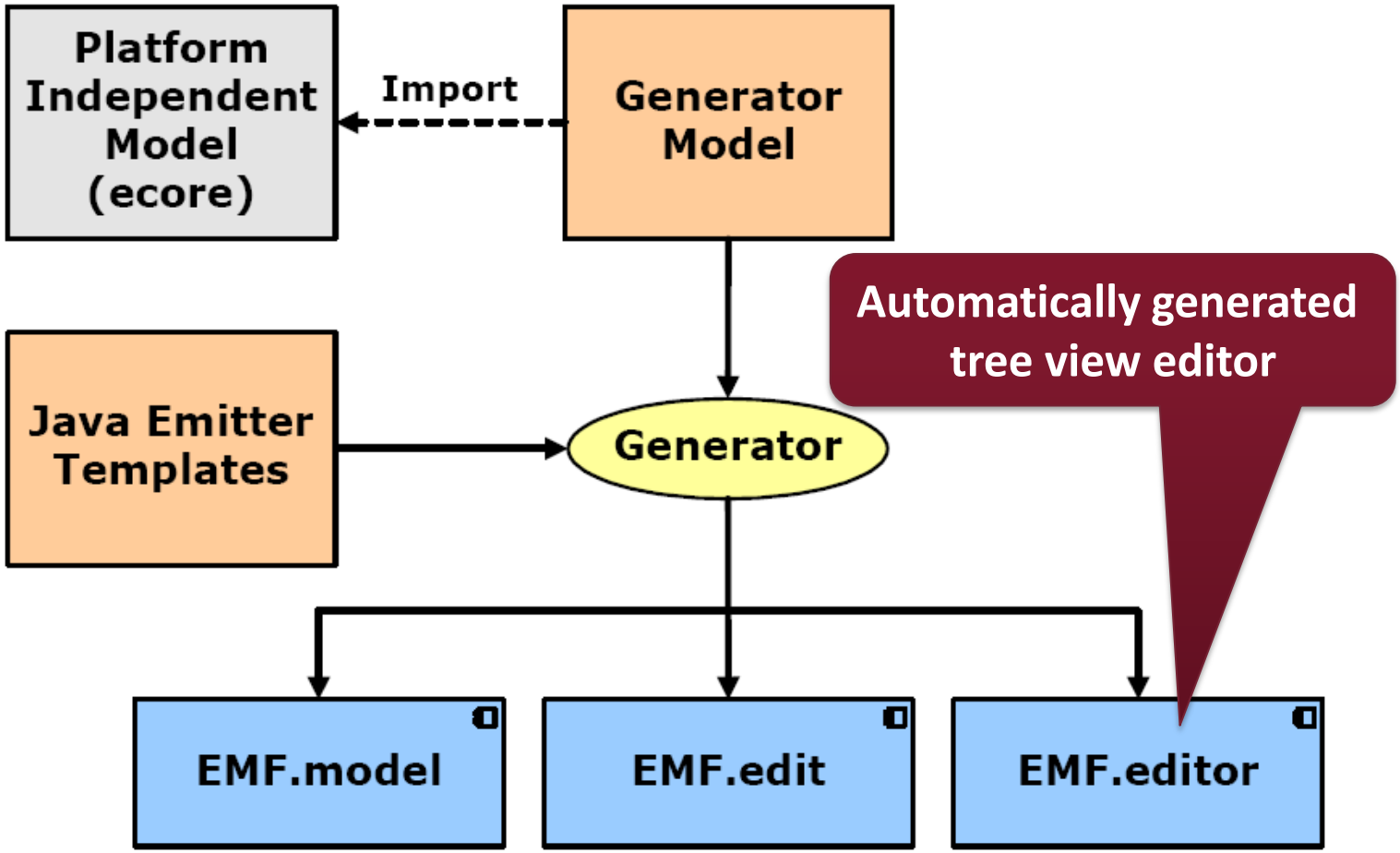
The EMF Toolkit



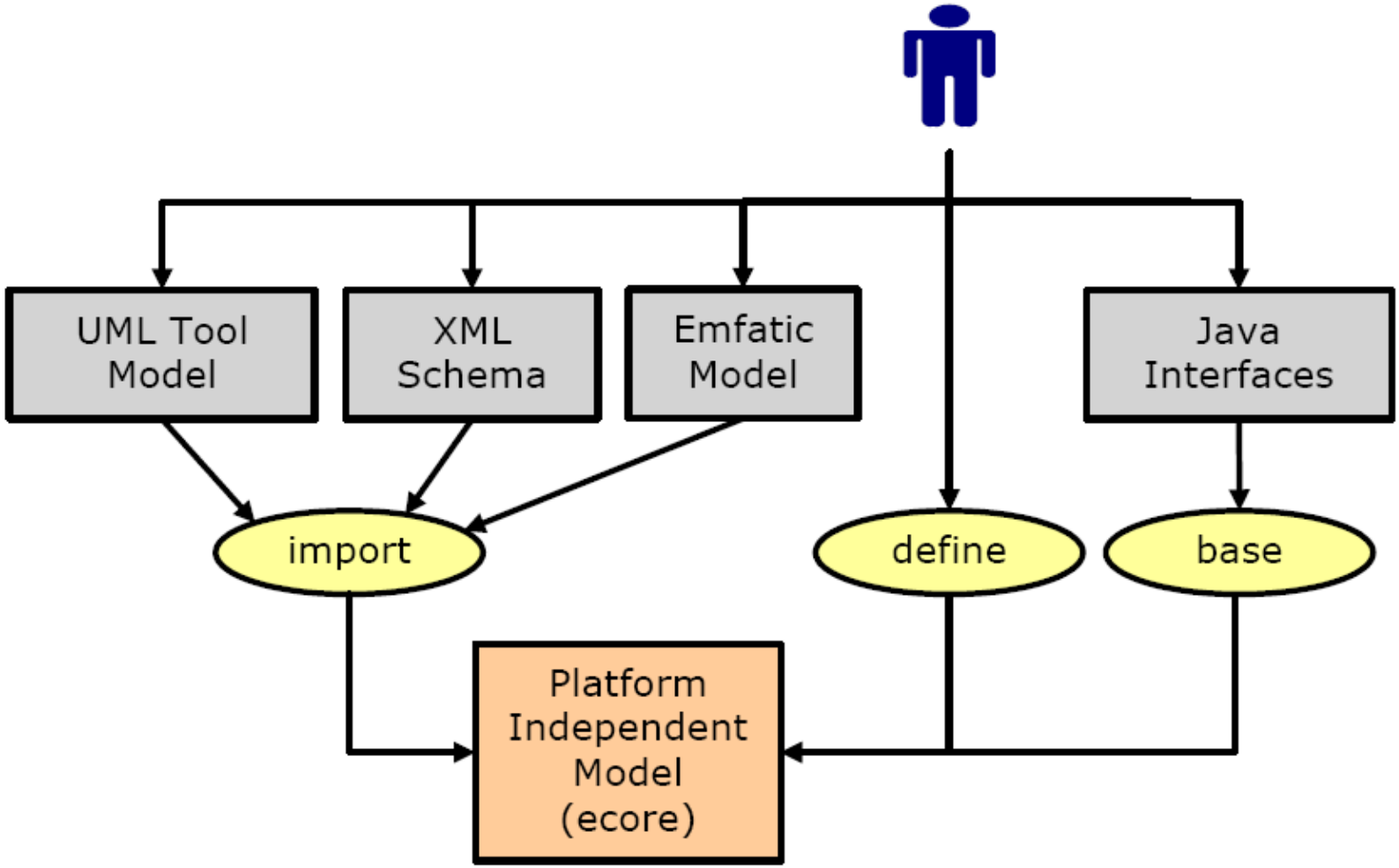
The EMF Toolkit



The EMF Toolkit

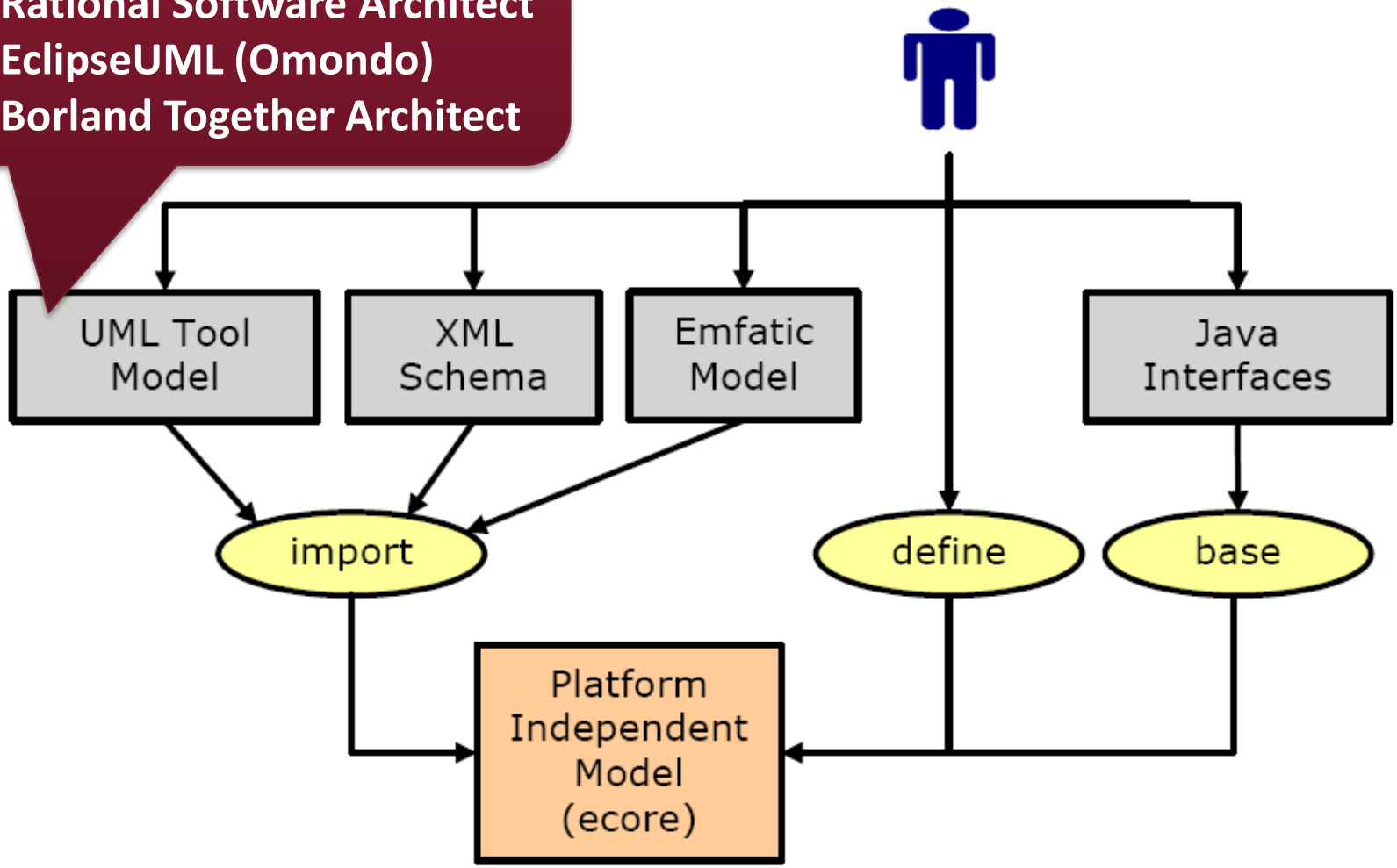


Creation of Ecore metamodels

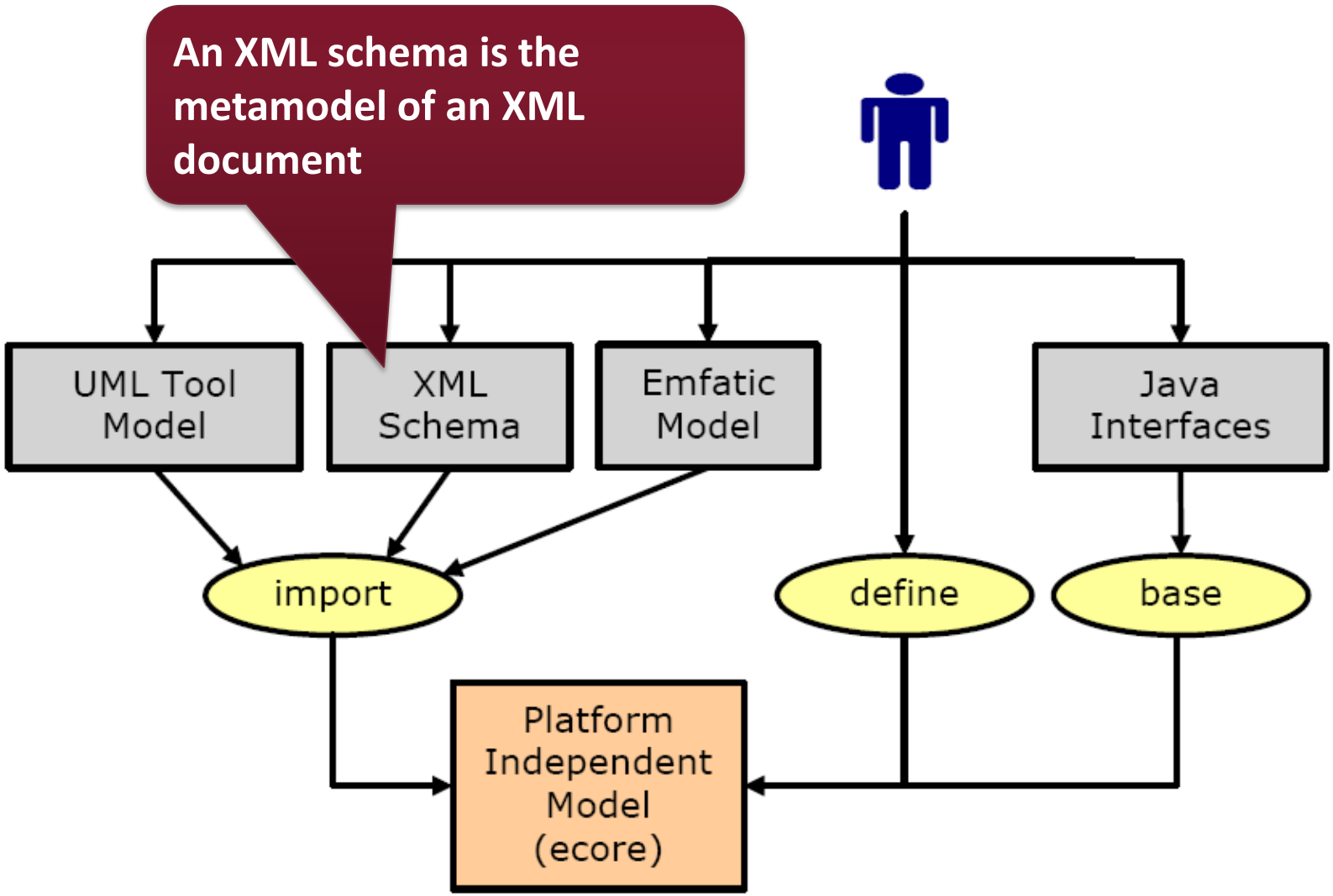


Creation of Ecore metamodels

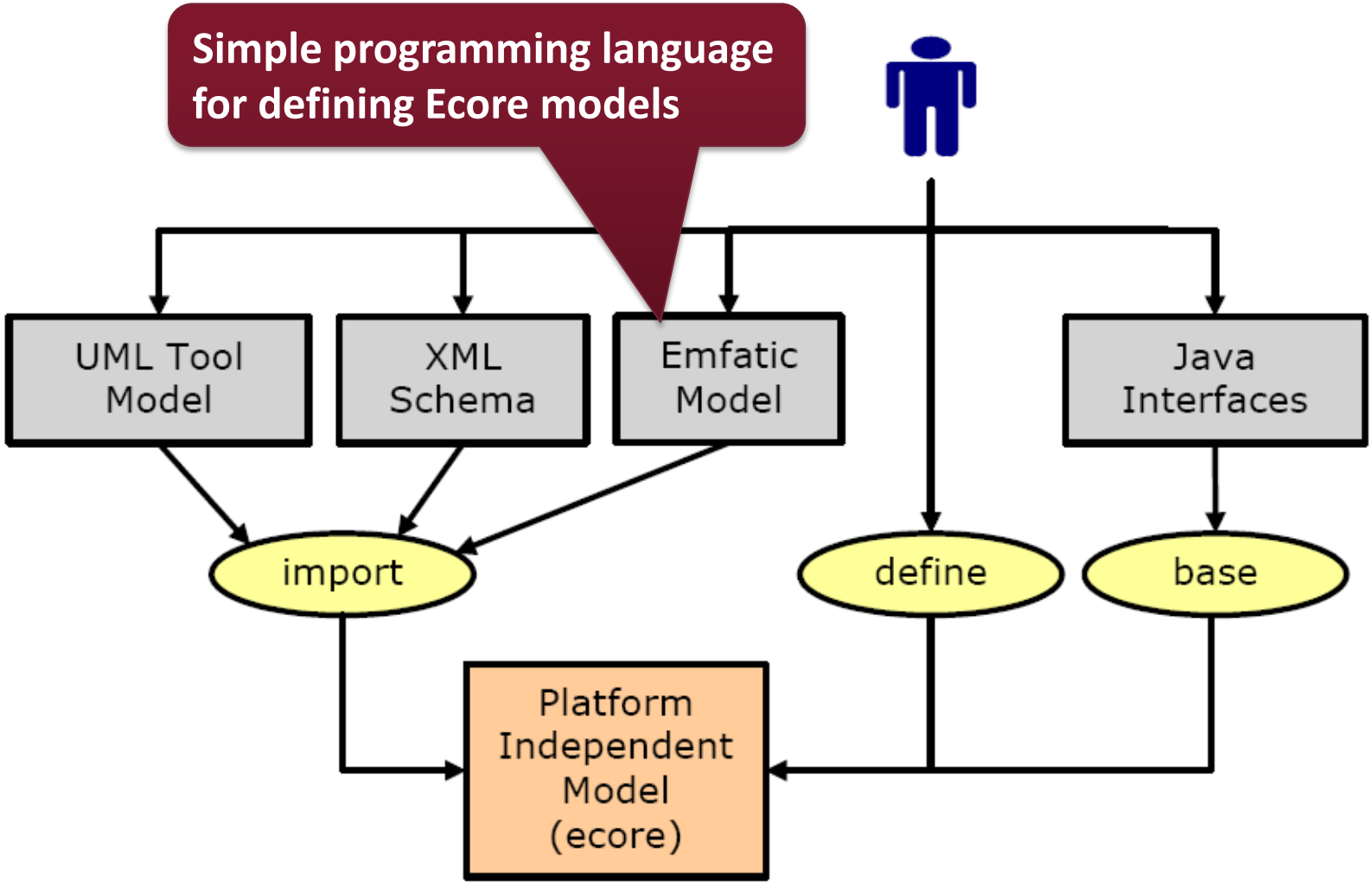
UML class diagram
- **Rational Software Architect**
- **EclipseUML (Omondo)**
- **Borland Together Architect**



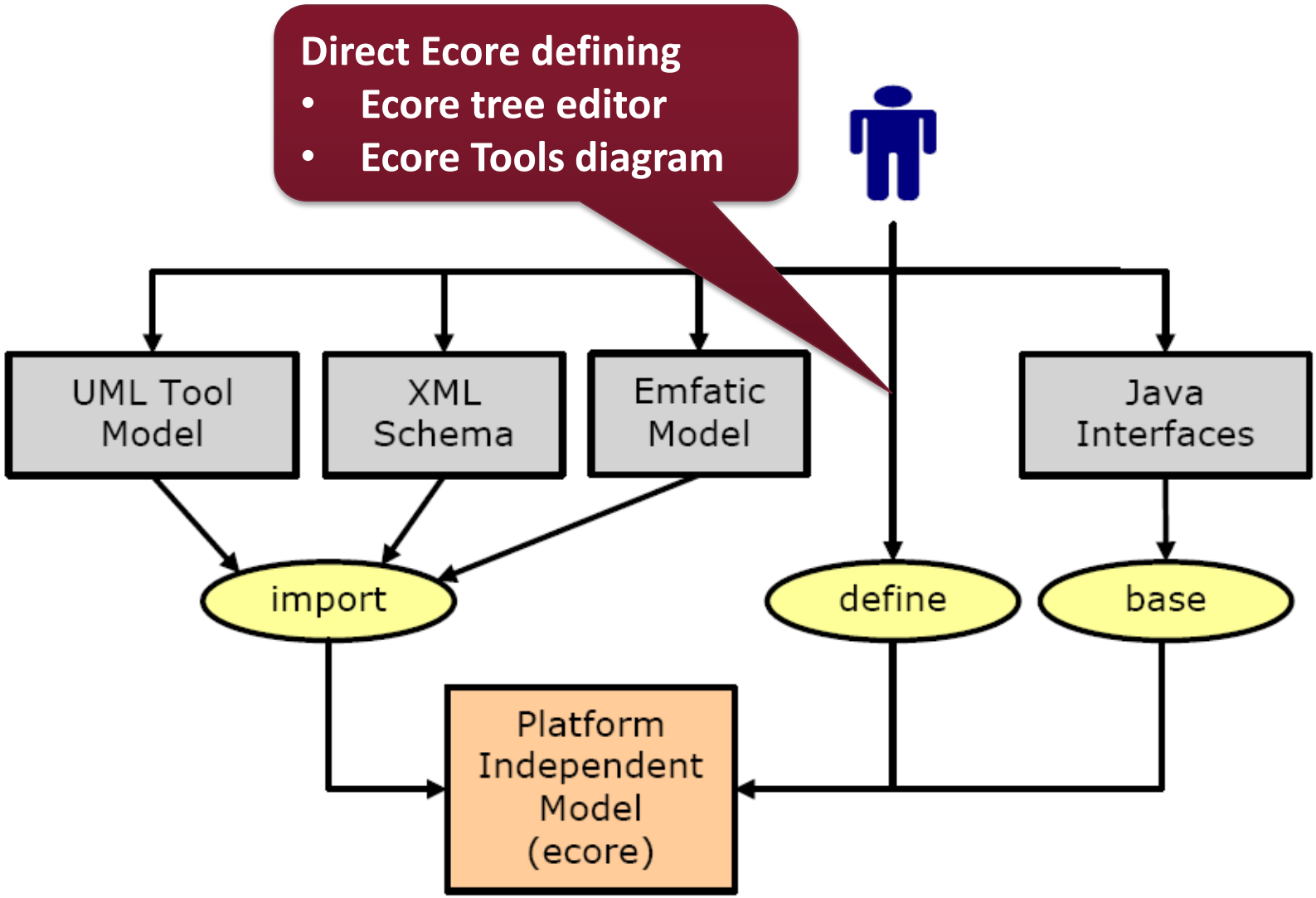
Creation of Ecore metamodels



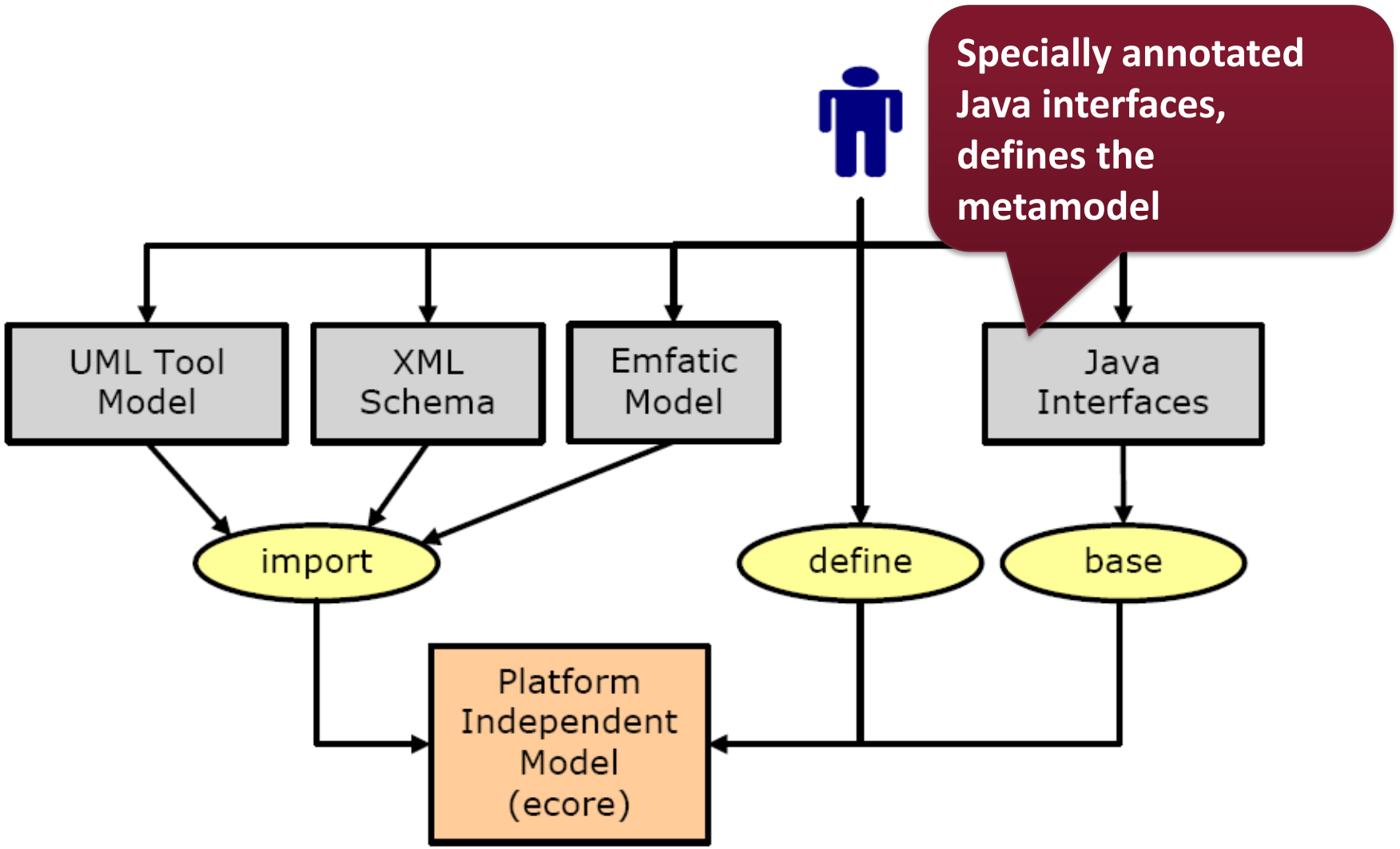
Creation of Ecore metamodels



Creation of Ecore metamodels

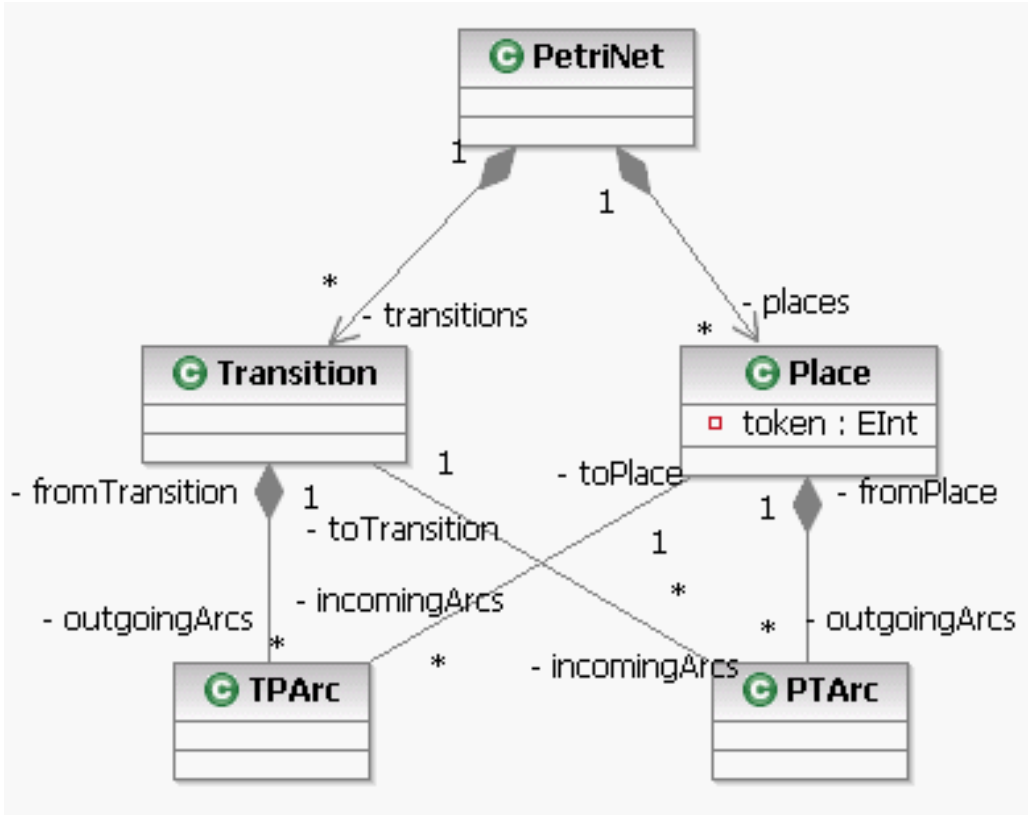


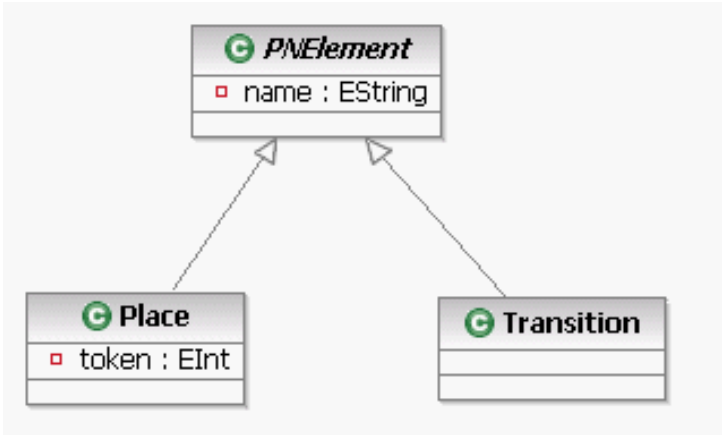
Creation of Ecore metamodels

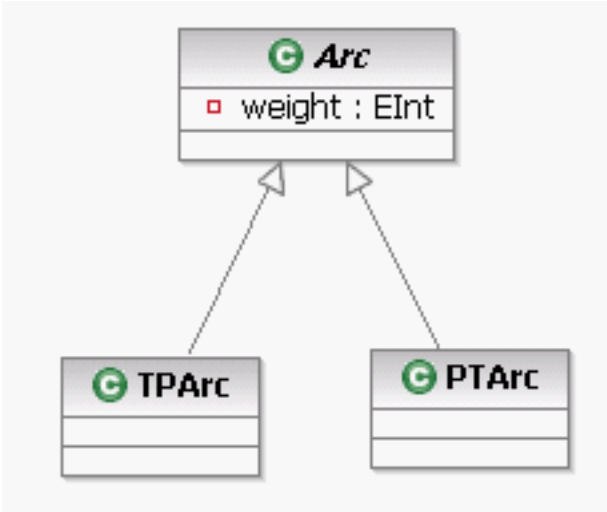


THE PETRI NET EXAMPLE

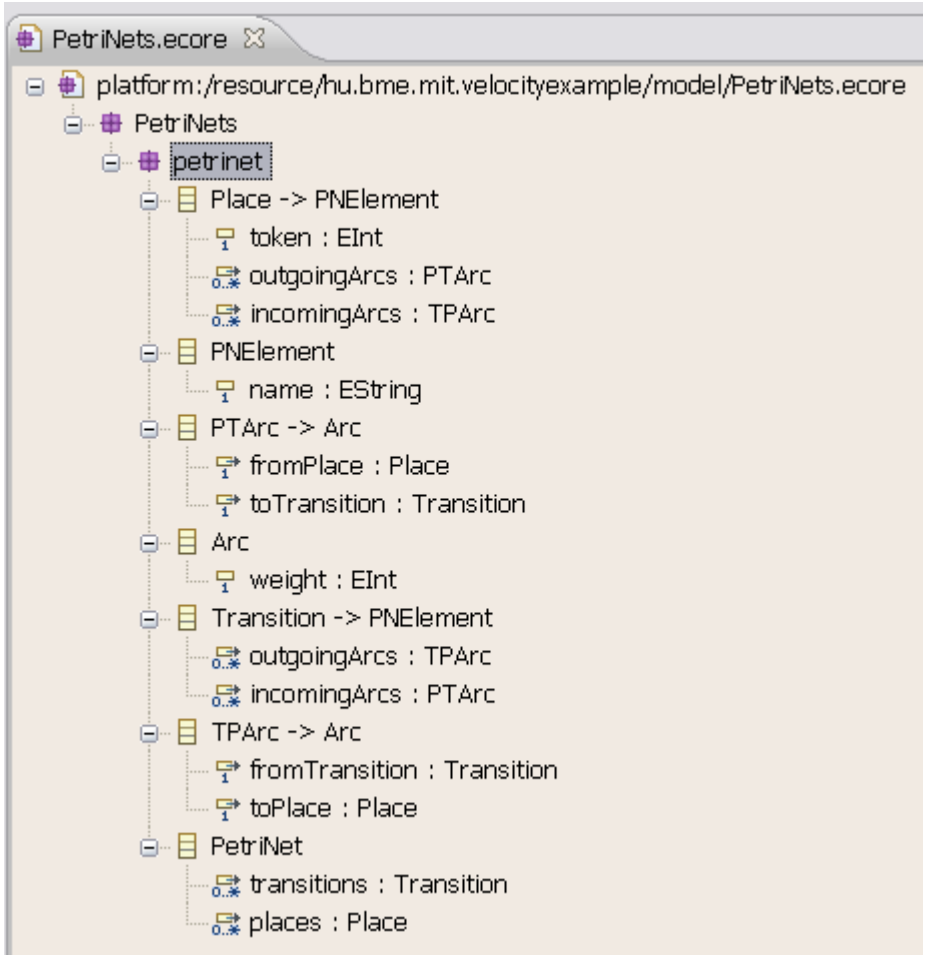
Domain Metamodel: Petri Nets



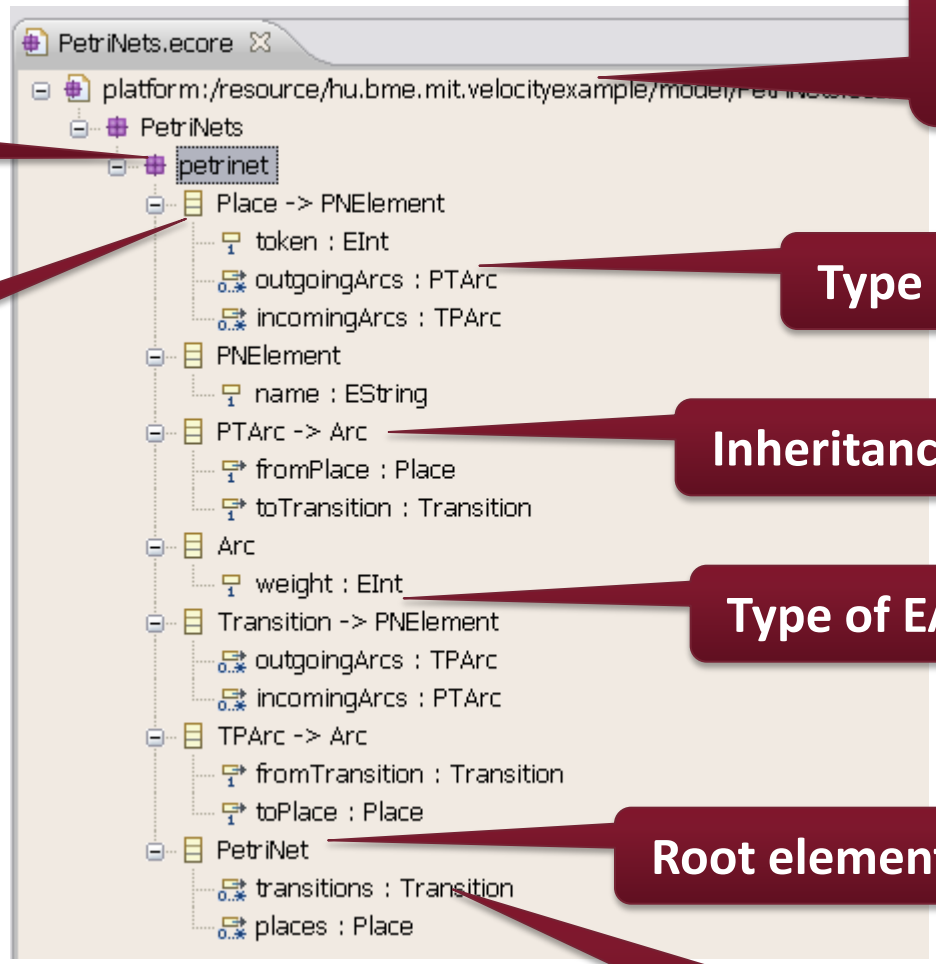




EMF model Ecore representation



EMF model Ecore representation



Path of containing resource

EPackage

EClass

Type of EReference

Inheritance

Type of EAttribute

Root element

Reference to all model elements

Class Definition in PetriNet.ecore

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

 <eStructuralFeatures xsi:type="ecore:EAttribute" name="token" lowerBound="1"
    eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EInt"/>

 <eStructuralFeatures xsi:type="ecore:EReference" name="outgoingArcs"
    upperBound="-1"
    eType="#//petrinet/PTArc" containment="true"
    eOpposite="#//petrinet/PTArc/fromPlace"/>

 <eStructuralFeatures xsi:type="ecore:EReference" name="incomingArcs"
    upperBound="-1"
    eType="#//petrinet/TPArc" eOpposite="#//petrinet/TPArc/toPlace"/>
</eClassifiers>
```

Class Definition in PetriNet.ecore

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

 <eStructuralFeatures xsi:type="ecore:EAttribute" name="token" lowerBound="1"
    eType="ecore:EDataType http://www.eclipse.org/emf/2002/Ecore#//EInt"/>

 <eStructuralFeatures xsi:type="ecore:EReference" name="outgoingArcs"
    upperBound="-1"
    eType="#//petrinet/PTArc" containment="true"
    eOpposite="#//petrinet/P

 <eStructuralFeatures xsi:type="ecore:EReference" name="incomingArcs"
    upperBound="-1"
    eType="#//petrinet/TPArc" eOpposite="#//petrinet/TPArc/toPlace"/>
</eClassifiers>
```

Class

Attribute

Reference

Multiplicity

Containment

Type

Opposite End

CODE GENERATION FROM ECORE

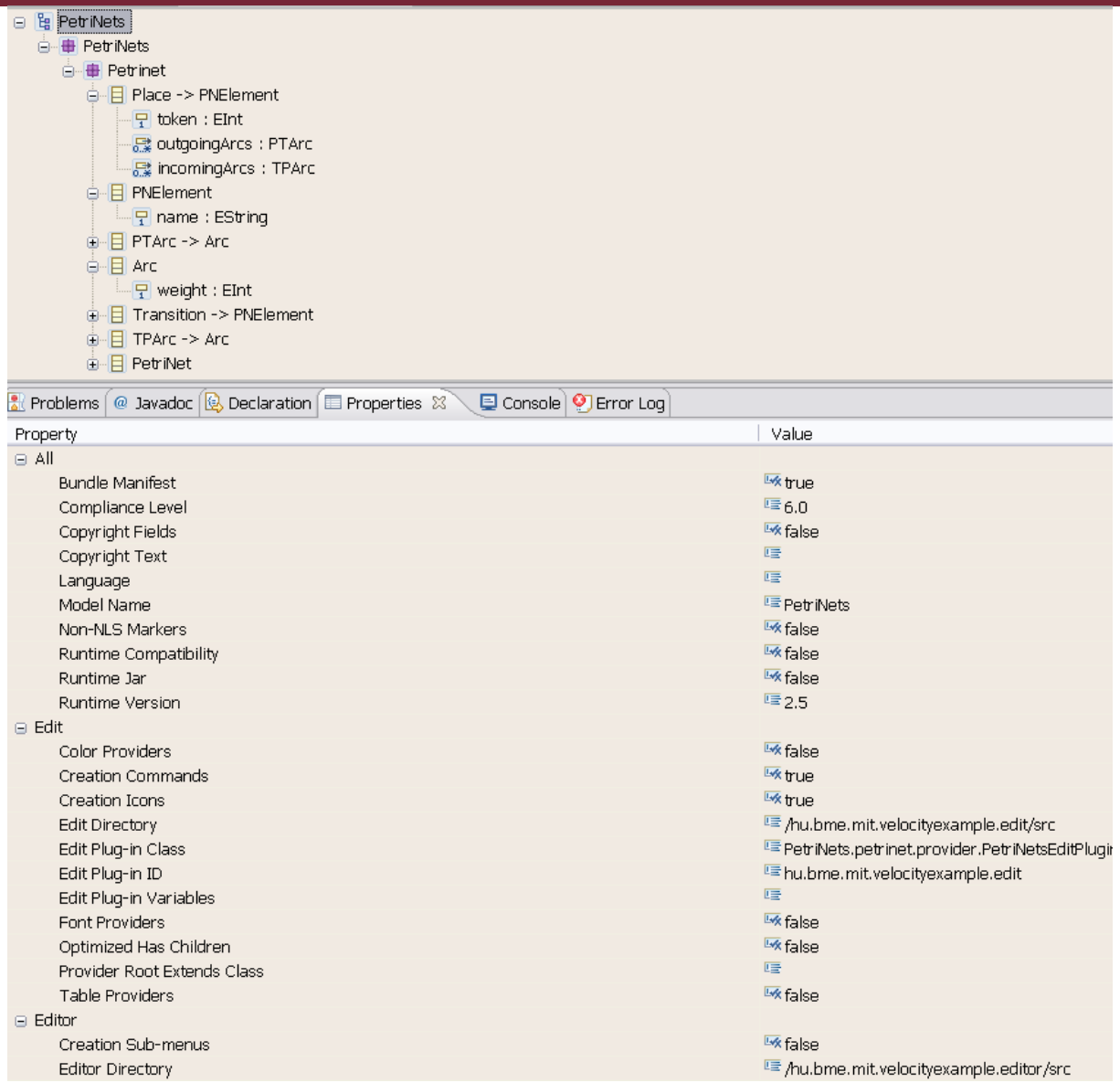
Generator model (.genmodel)

- Goal:
  - Specify the attributes of the code generation
- EMF model
  - Tree Editor
  - Refers to the ecore model
- Code generation attributes
  - Java version (e.g., use Enums in case of Java 5 and higher)
  - Package/project names
  - …

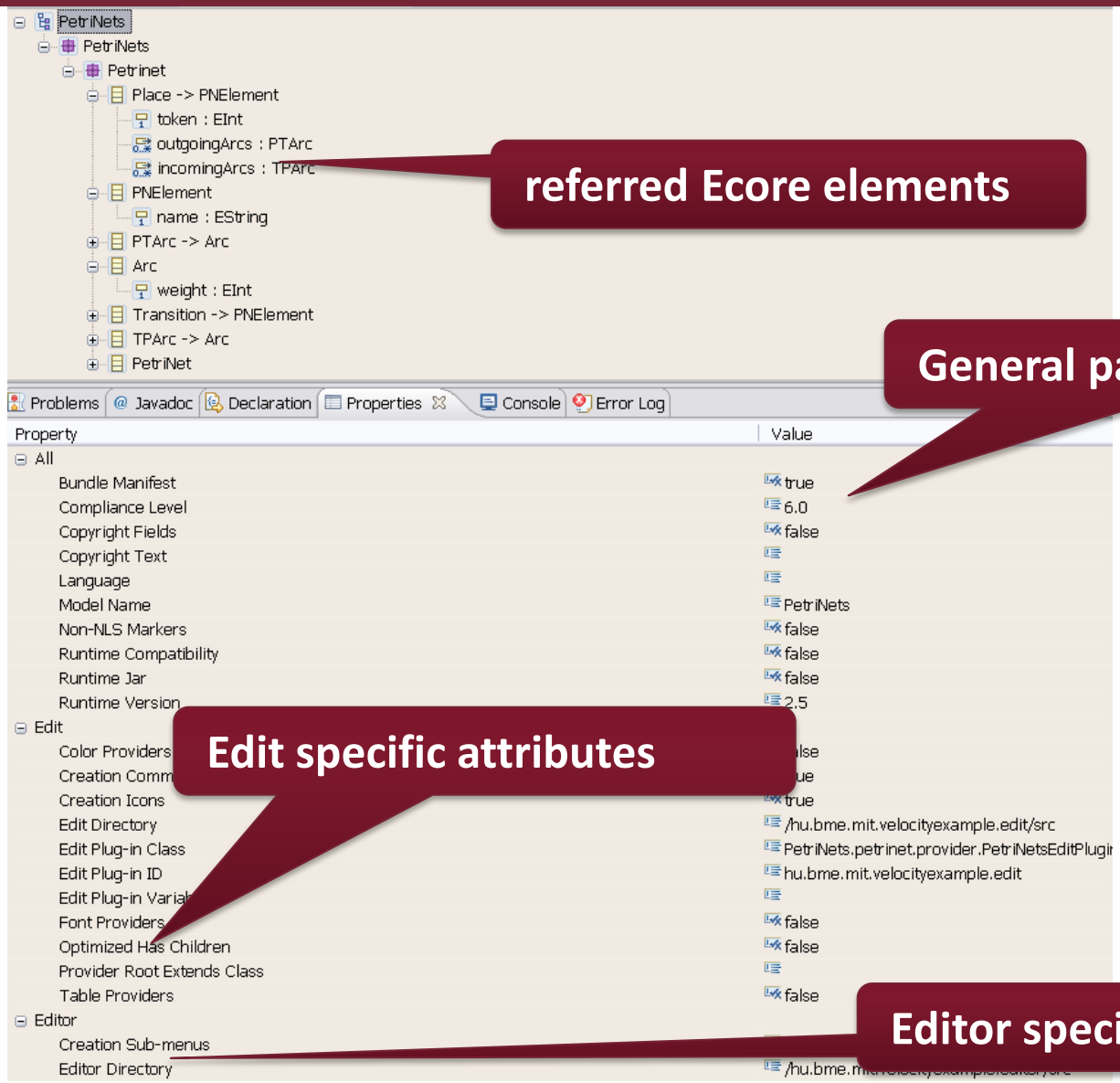
Code Generation from Ecore (.genmodel)

- Ecore model remains pure and independent
- Customizable (wrappers, code formatters, etc.)
- Generated plugins:
  - Model persistency (EMF.model)
  - Model management (EMF.edit)
  - Model editor (EMF.editor)
- Has some limitations
  - What happens when the underlying .ecore changes?

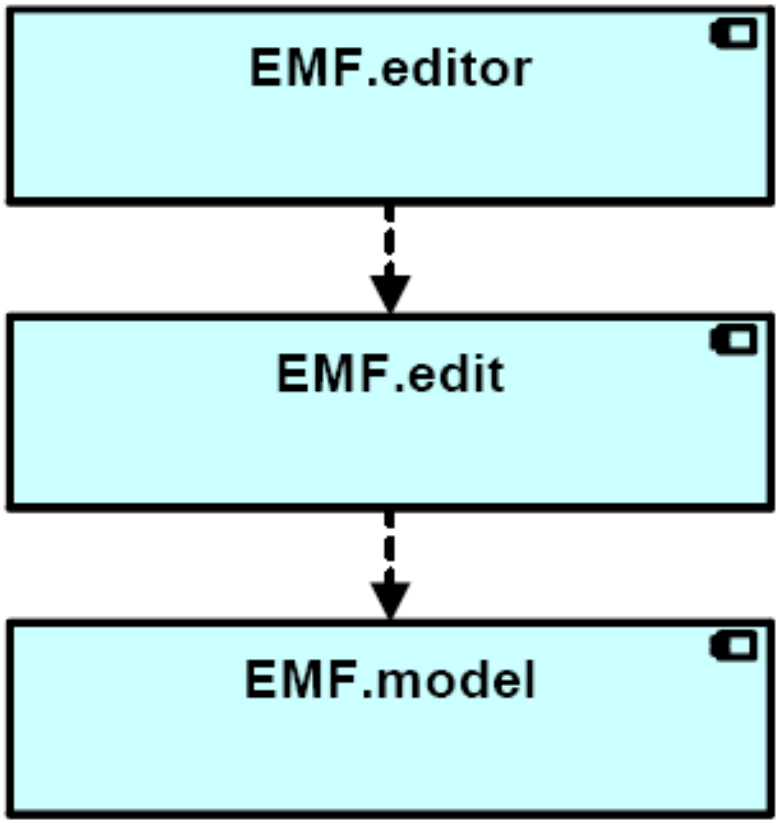
Generator model



Generator model

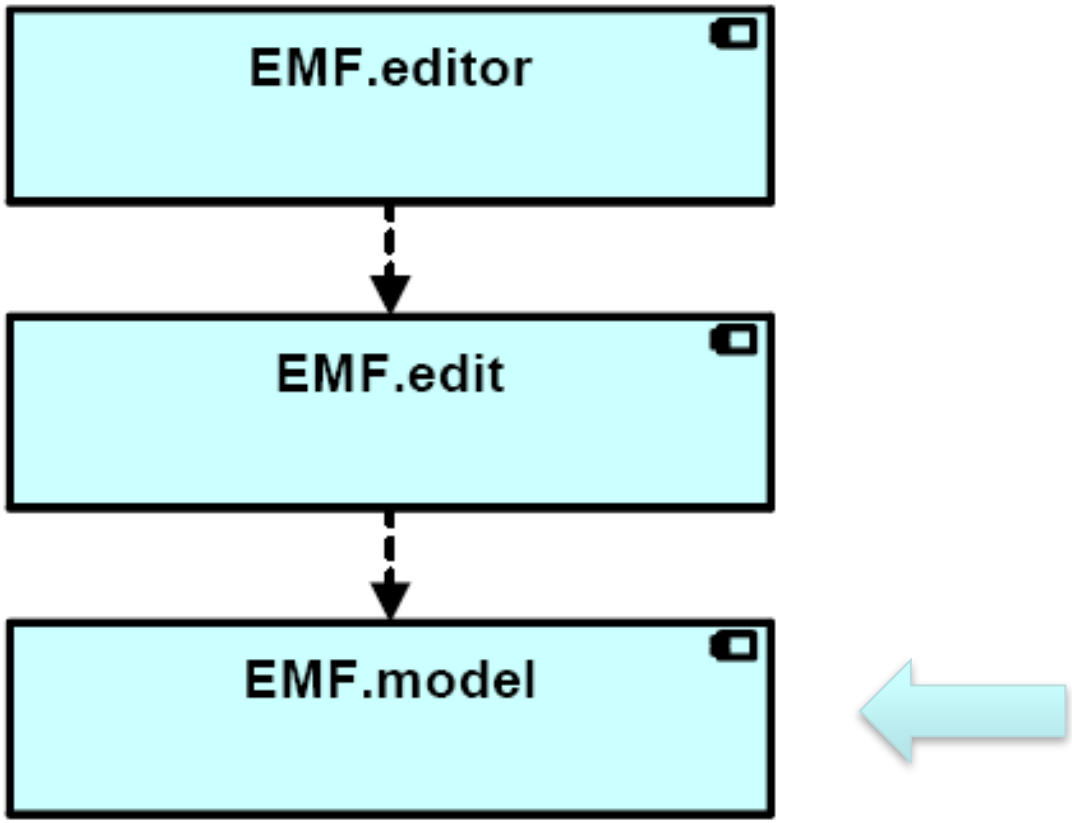


Generated EMF components



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

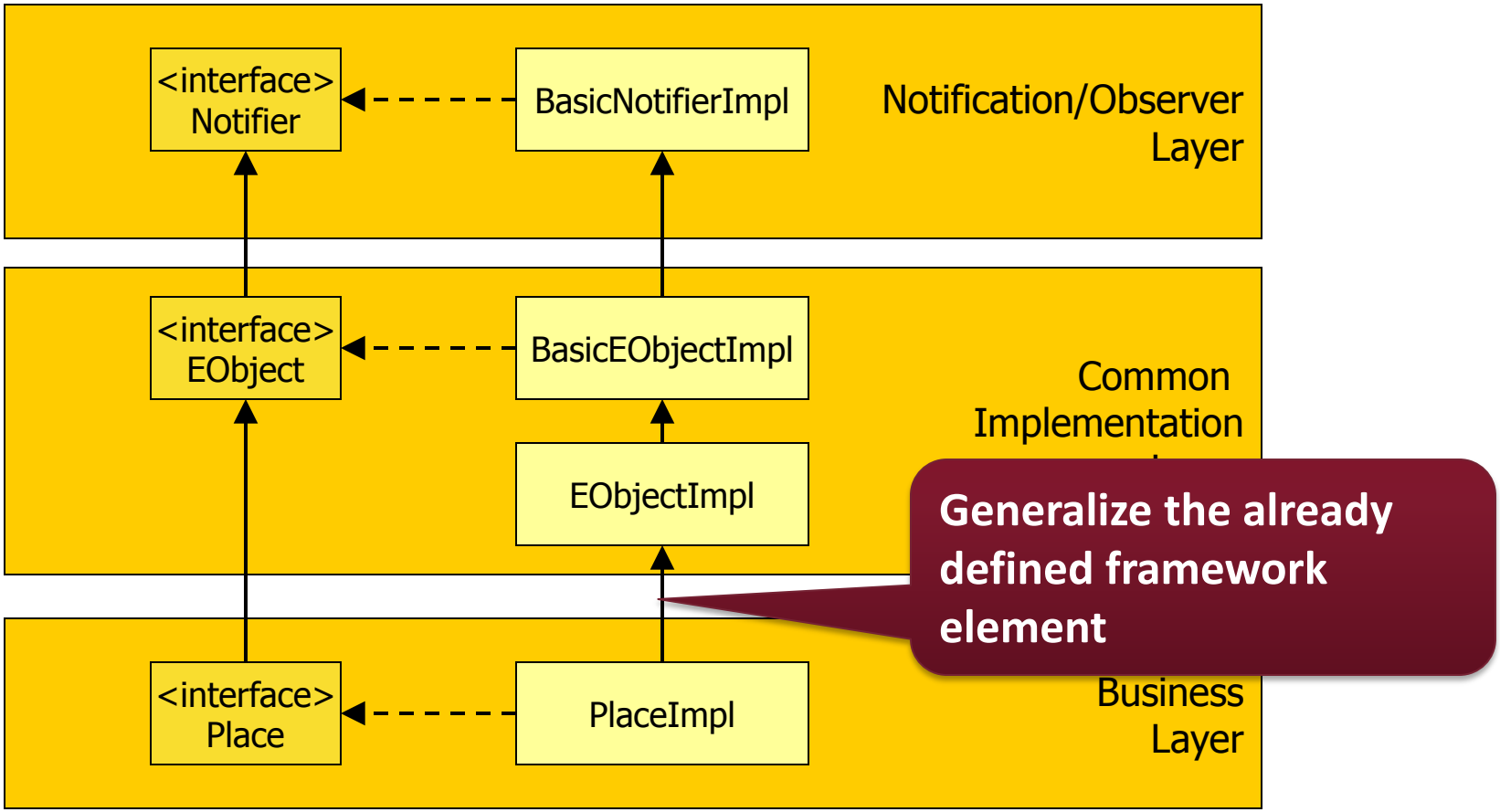
Generated EMF components



EMF.model

- Optimized persistency handling
- Fully featured Java code of the Ecore model
- Specific factories for all packages
- Notification mechanism (observer pattern)
- Possible extension points:
  - Advanced editor
  - Own file format with parser

EClass implementation



Auto-Generated Interface

```
 * @model
 * @generated
 */
public interface Place extends PNElement {
    /**
     * @model required="true"
     * @generated
     */
    int getToken();

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

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

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

} // Place
```

ESuperClass

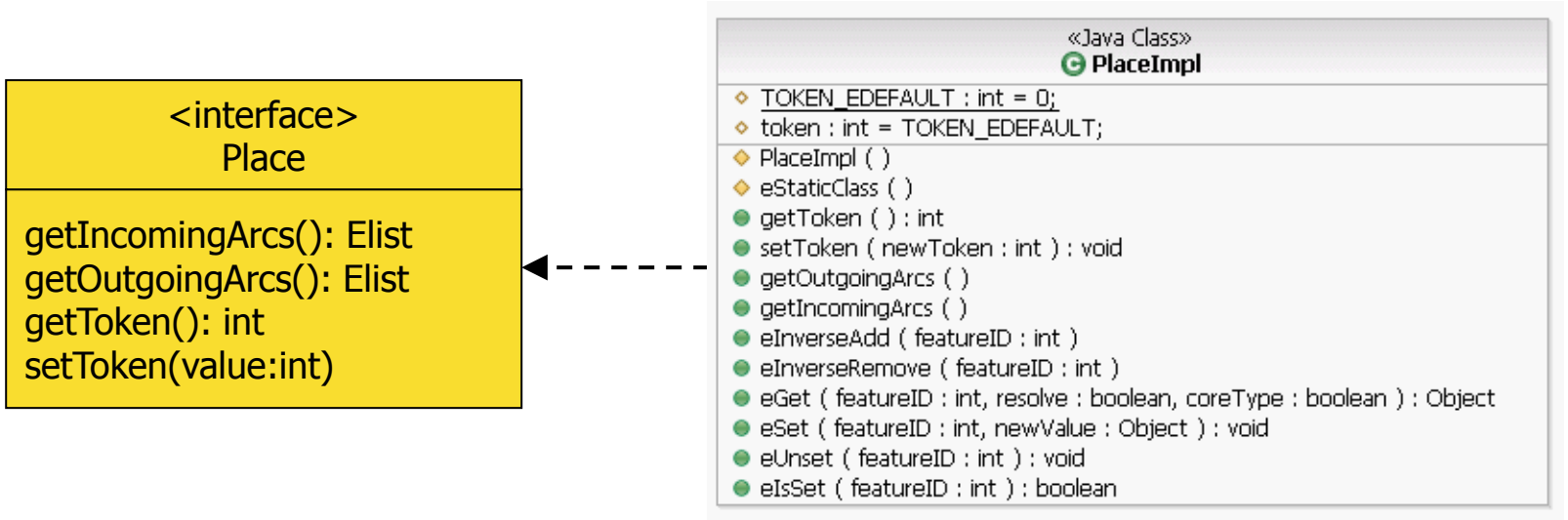
EMF specific „annotations”

Getters/Setters for attributes

No setter when multiplicity > 1 (use add/remove instead)

EList: EMF list interface (~10 implementations

EObject API



- Every class contains framework-specific methods:
  - Reflective get/set (eGet, eSet)
  - Consistent manipulation (eInverseRemove)
  - Notifications for feature changes (very useful e.g. in GUI!)
- Inherited from common supertype EObject
  - see deep instantiation earlier

EOperation Implementation

```
public class XImpl extends EObjectImpl implements X {

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

- Represents the frame of a Java method
- Present in both the interface and implementing class
- Important:
  - Have to change the generated annotation to NOT
    - …so that next code generation phase does not overwrite it
  - Have to implement the method manually

Client Programming with EMF

```
Place p1 = PetrinetFactory.eINSTANCE.createPlace();
p1.setName("p1");
Place p2 = PetrinetFactory.eINSTANCE.createPlace();
p2.setName("p2");
Transition t1 = PetrinetFactory.eINSTANCE.createTransition();
t1.setName("t1");

// Inverse direction (p1.outgoingArcs) is set automatically
PTArc a0 = PetrinetFactory.eINSTANCE.createPTArc();
a0.setFromPlace(p1);
a0.setToTransition(t1);
TPArc a1 = PetrinetFactory.eINSTANCE.createTPArc();
a1.setToPlace(p2);
a1.setFromTransition(t1);
```

Create a place

Create a transition

Create a PT arc

Set source of PT arc

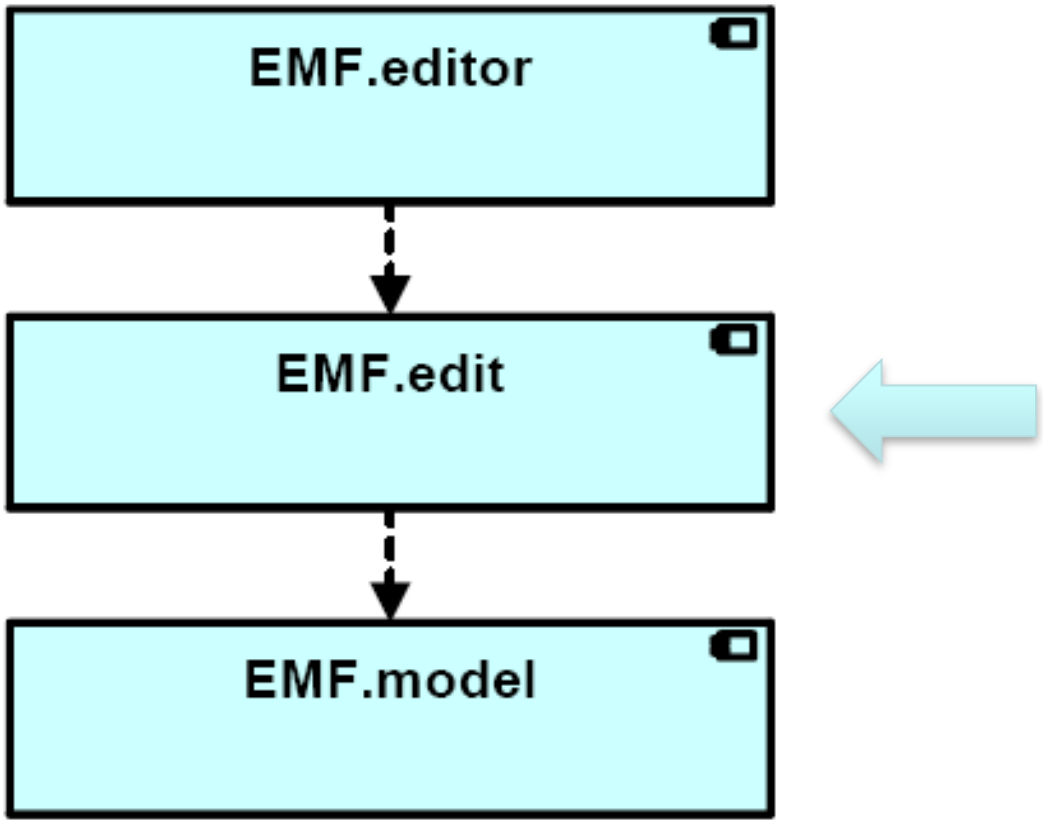
Set target of PT arc

Advanced client programming: Reflective Ecore API

The org.eclipse.emf.ecore.util Package

- Contains utility classes and interfaces:
  - *ECoreEContentAdapter*: maintains itself as a notification adapter for a whole containment (sub)tree
  - *UsageCrossReferencer*: finds each ModelElement pointing to the corresponding EObject
  - *ContentTreeIterator*: An iterator over the tree contents of a collection of EObjects
  - *Copier*: deep copy of EObject Elements and EReferences
  - Etc.
- (This is not generated but a generic component)

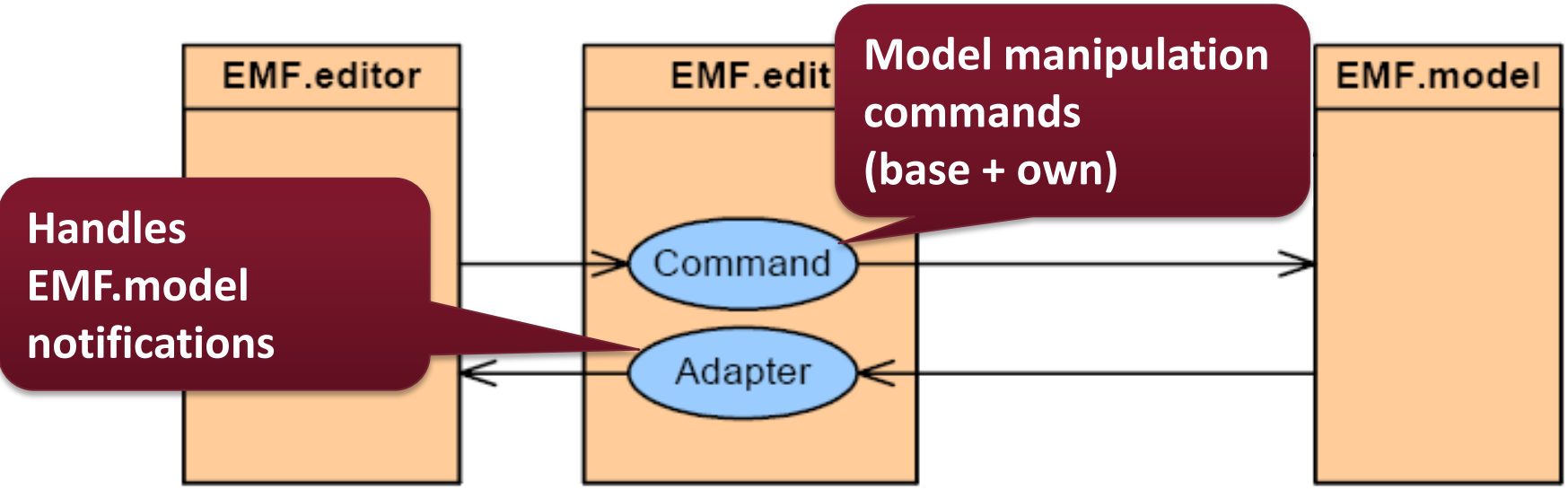
Generated EMF components



EMF.Edit

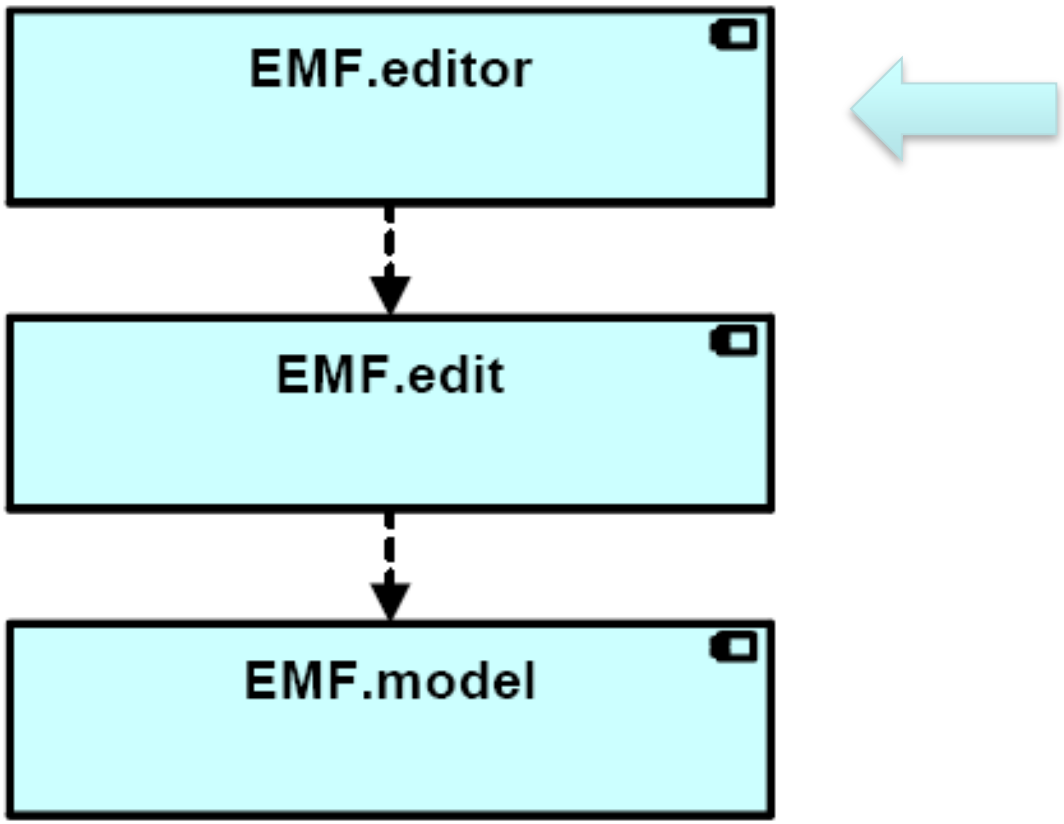
- Separates the GUI and the model
- Generator pattern:
  - Provider class for each model element
  - Base class: ItemProvider
  - Forward EMF model change notifications to the viewer
- Provides:
  - Element text
  - Icon
  - Description of features in EClass

EMF.Edit

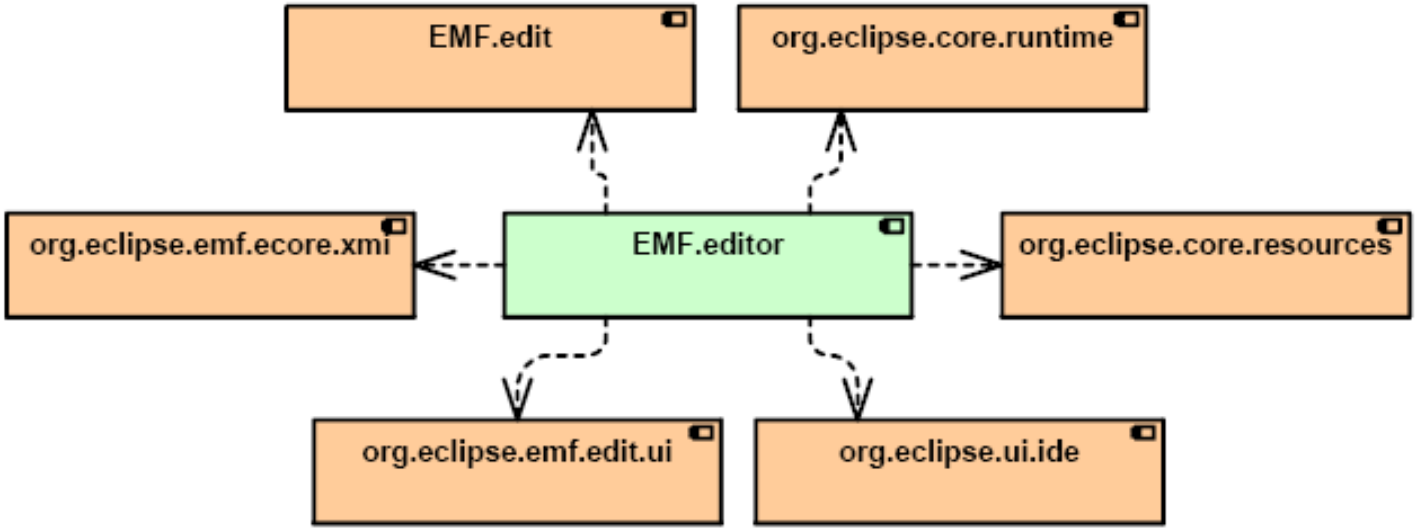


- Converts model notifications to GUI notifications
- Model manipulation through commands
  - Possible alternative to direct setters
  - Undoable, redoable
  - ItemProvider.create*Add*Command(…) etc.

Generated EMF components

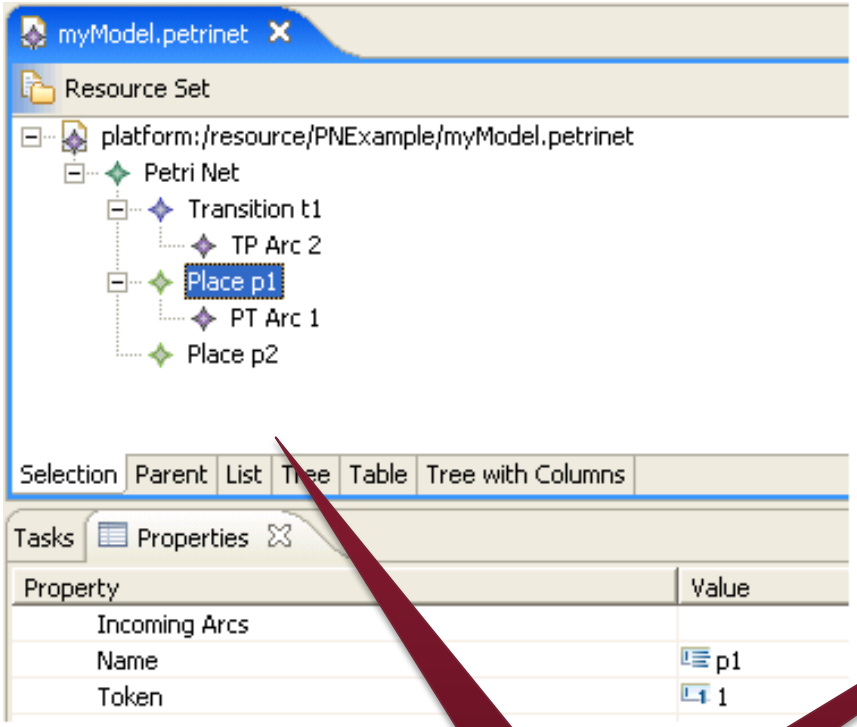


EMF.Editor



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

The editor of Petri Net models




```
<?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

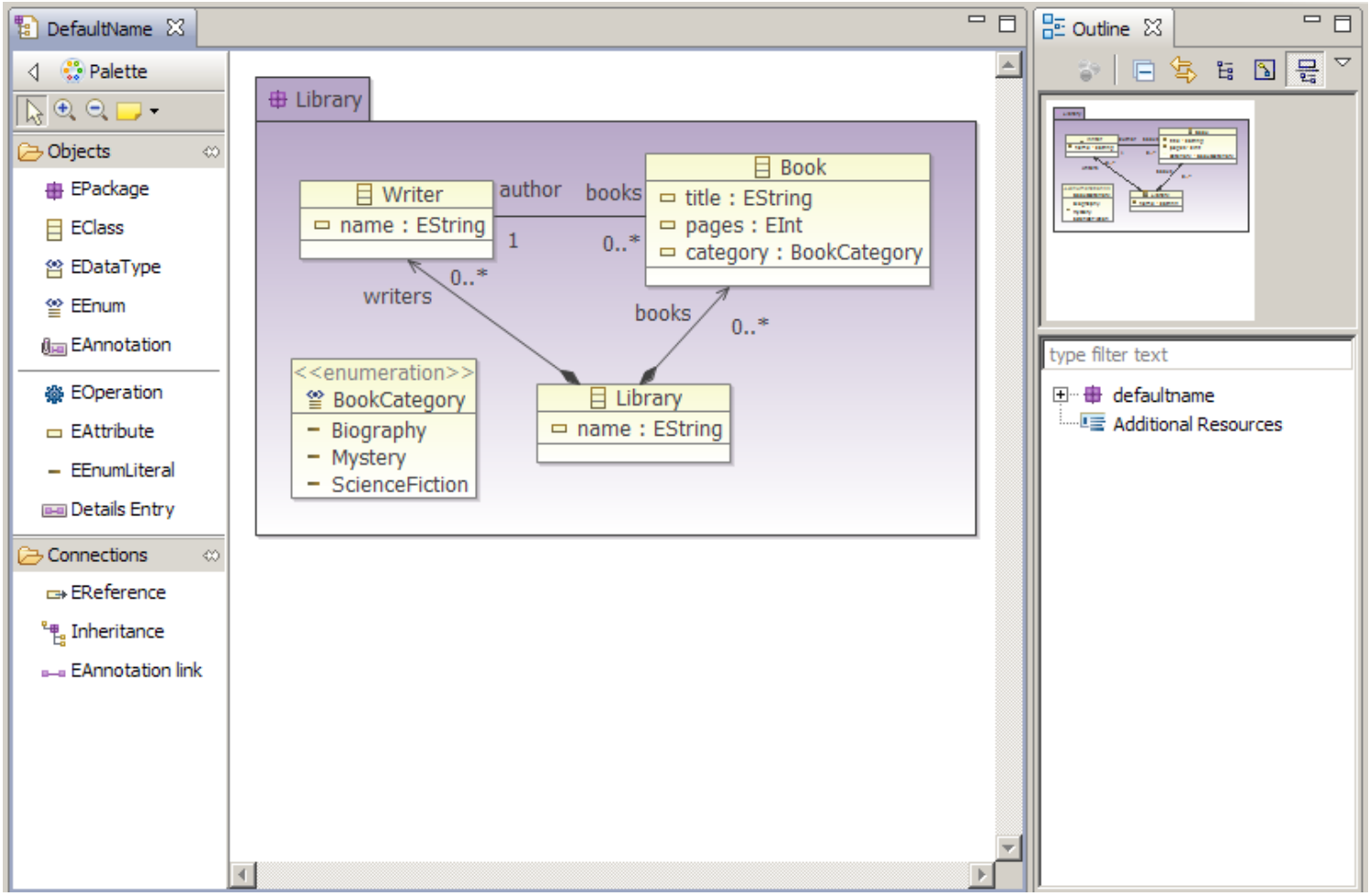
TOOLS, API AND UTILITIES

Basic EMF tools

- Validation
  - Validate constraints over EMF models
- Query
  - High-level query language for EMF
  - See also: EMF-IncQuery ☺
- Compare
  - To structurally compare EMF models (e.g., versioning)
- Teneo
  - Persistency layer over relation databases
- SDO
  - Service Oriented Architecture based on EMF
- CDO
  - distributed, client-server EMF models

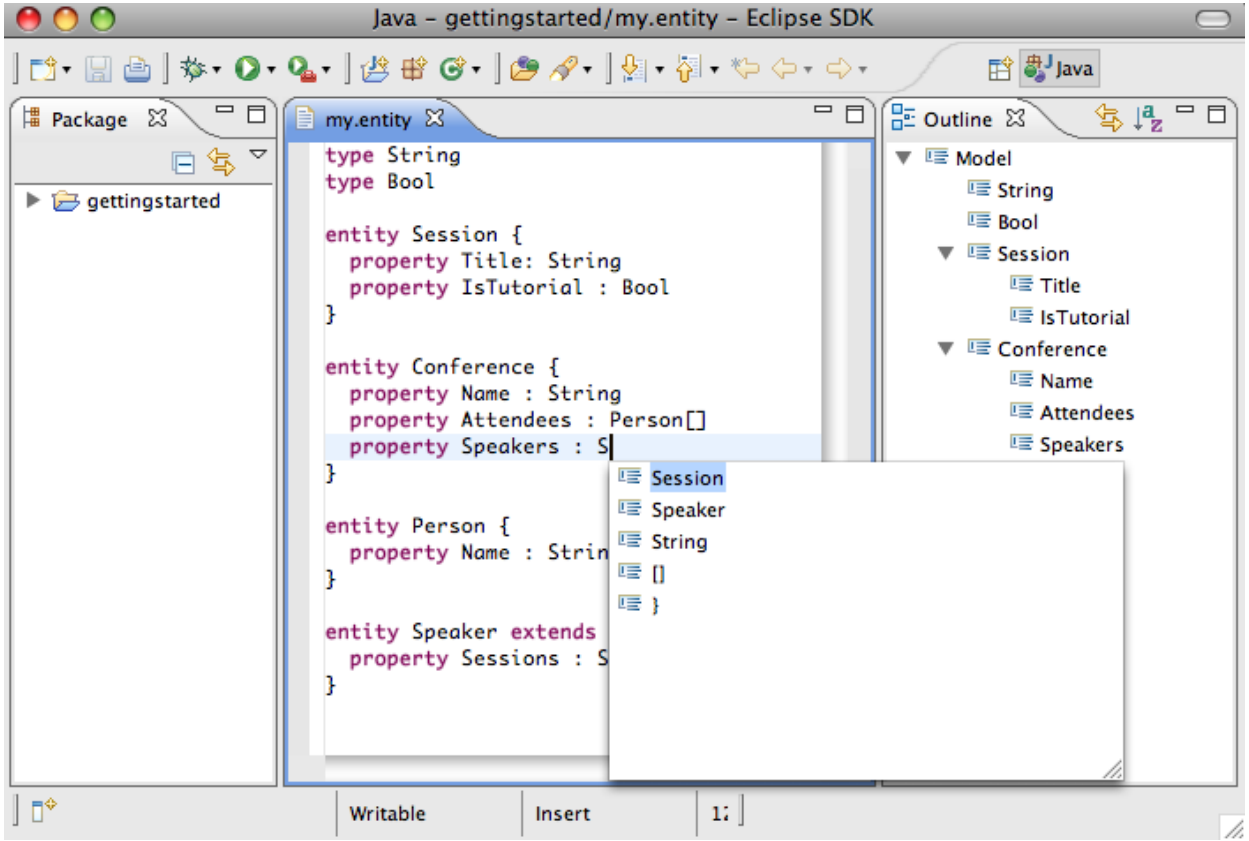
Ecore Tools: Ecore Diagram Editor

- Graphical DSL to define EMF metamodels
  - Based on GMF

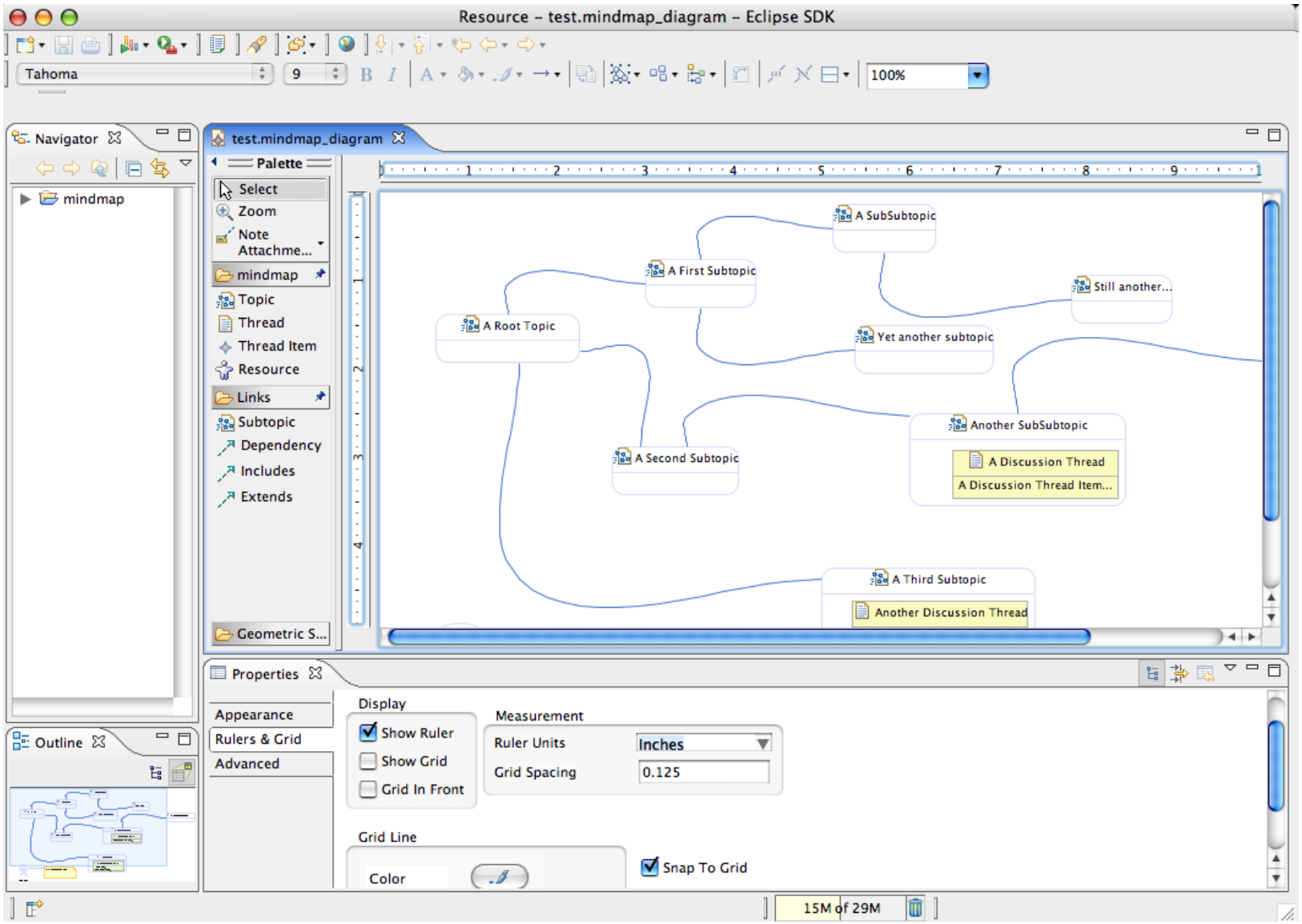


Xtext

- Textual DSL for defining metamodel + textual syntax
- Context-free grammar!
- Generates:
  - Metamodel
  - Parser
  - Editor features



GMF


Resource – test.mindmap_diagram – Eclipse SDK
Tahoma
9
100%
Navigator
mindmap
test.mindmap_diagram
Palette
Select
Zoom
Note Attachme...
mindmap
Topic
Thread
Thread Item
Resource
Links
Subtopic
Dependency
Includes
Extends
Geometric S...
A SubSubtopic
A First Subtopic
Still another...
A Root Topic
Yet another subtopic
Another SubSubtopic
A Second Subtopic
A Discussion Thread
A Discussion Thread Item...
A Third Subtopic
Another Discussion Thread
Outline
Properties
Appearance
Rulers & Grid
Advanced
Display
Show Ruler
Show Grid
Grid In Front
Measurement
Ruler Units
Inches
Grid Spacing
0.125
Grid Line
Color
Snap To Grid
15M of 29M

Köszönetnyilvánítás

Az oktatási anyag elkészítése az Európai Uniós és Magyarország támogatásával a TÁMOP 4.2.4.A/1-11-1-2012-0001 azonosító számú „Nemzeti Kiválóság Program – Hazai hallgatói, illetve kutatói személyi támogatást biztosító rendszer kidolgozása és működtetése országos program” című kiemelt projekt keretei között valósult meg.