AOT LAB

Agent and **O**bject **T**echnology **Lab**
Dipartimento di Ingegneria dell'Informazione
Università degli Studi di Parma

Eclipse Modelling Framework (EMF)

Alessandro Negri
negri@ce.unipr.it
http://www.ce.unipr.it/people/negri/

What is EMF?

- A modeling & data integration framework
- Exploits the facilities offered in Eclipse to...
  - Generate code without losing user customizations (merge)
  - Automate important tasks (such as registering the runtime information)
  - Improve extensibility
  - Provide a UI layer
- What is an EMF “model”?
  - Specification of your application’s data
    - Object attributes
    - Relationships (associations) between objects
    - Operations available on each object
    - Simple constraints (eg. cardinality) on objects and relationships
  - Essentially it represents the class diagram of the application

AOT LAB

What does EMF Provide?

- From a model specification, EMF can generate efficient, correct, and easily customizable implementation code
- Out of the box, EMF provides support for
  - Java™ interfaces
  - UML
  - XML Schema
- EMF converts your models to Ecore (EMF metamodel)
- Tooling support within the Eclipse framework (UI, headless mode, Ant and standalone), including support for generating Eclipse-based and RCP editors
- Reflective API and dynamic model definition
- Persistence API with out of box support for XML/XMI (de)serialization of instances of a model
- And much more….

Why EMF?

- EMF is middle ground in the modeling vs. programming worlds
  - Focus is on class diagram subset of UML modeling (object model)
  - Transforms models into Java code
  - Provides the infrastructure to use models effectively in your application
- Very low cost of entry
  - EMF is free and open source
  - Full scale graphical modeling tool not required
  - Reuses your knowledge of UML, XML Schema, or Java
- It's real, proven technology (since 2002)

AOT LAB

Who is Using EMF Today?

- Eclipse projects
  - Foundation for the Modeling Project: Graphical Modeling Framework (GMF), EMF Ontology Definition Metamodel (EODM), UML2…
  - Other uses: Web Tools Platform (WTP), Test and Performance Tools Platform (TPTP), Business Intelligence and Reporting Tools (BIRT), Data Tools Platform (DTP), Visual Editor (VE)…
- Commercial offerings
  - IBM, Borland, Oracle, Omondo, Versata, MetaMatrix, Bosch, Ensemble…
- Applied sciences
  - Darmstadt University of Technology, Mayo Clinic College of Medicine, European Space Agency…
- Large open source community
  - Over 1,000,000 download requests in 2006

Creating the Ecore Model

- Representing the modeled domain in Ecore is the first step in using EMF
- Ecore can be created
  - Directly using the EMF editors
  - Through a graphical UI provided by external contributions
  - By converting a model specification for which a Model Importer is available
- Model Importers available in EMF
  - Java Interfaces
  - UML models expressed in Rational Rose® files
  - XML Schema
- Choose the one matching your perspective or skills

AOT LAB

Model Importers Available in EMF

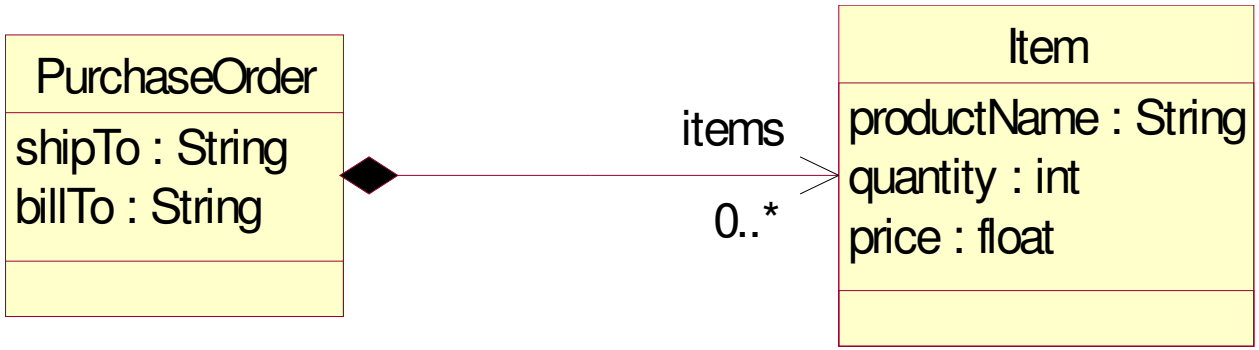
- Java Interfaces

```
public interface PurchaseOrder
{
  String getShipTo();
  void setShipTo(String value);
  String getBillTo();
  void setBillTo(String value);
  List getItems(); // List of Item
}
```

```
public interface Item
{
  String getProductName();
  void setProductName(String value);
  int getQuantity();
  void setQuantity(int value)
  float getPrice();
  void setPrice(float value);
}
```

Model Importers Available in EMF

- UML Class Diagram



AOT LAB

Model Importers Available in EMF

- XML Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            targetNamespace="http://www.example.com/SimplePO"
            xmlns:PO="http://www.example.com/SimplePO">
  <xsd:complexType name="PurchaseOrder">
    <xsd:sequence>
      <xsd:element name="shipTo" type="xsd:string"/>
      <xsd:element name="billTo" type="xsd:string"/>
      <xsd:element name="items"  type="PO:Item"
                minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="Item">
    <xsd:sequence>
      <xsd:element name="productName" type="xsd:string"/>
      <xsd:element name="quantity" type="xsd:int"/>
      <xsd:element name="price" type="xsd:float"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

Unifying Java, XML and UML Technologies

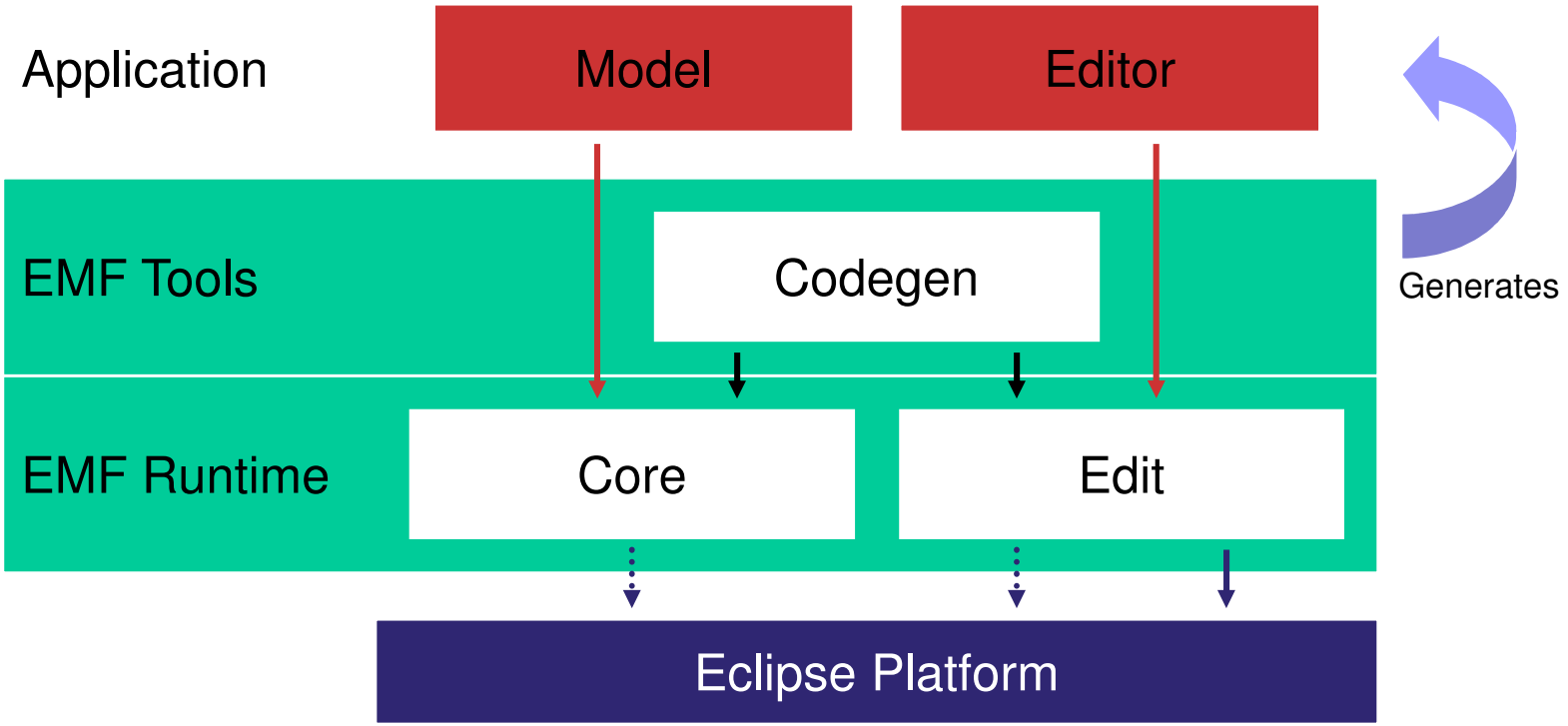
- The Model Importers available in EMF were carefully chosen to integrate today's most important technologies
- All three forms provide the same information
  - Different visualization/representation
  - The application's "model" of the structure
- From a model definition, EMF can generate
  - Java implementation code, including UI
  - XML Schemas
  - Eclipse projects and plug-in

AOT LAB

Typical EMF Usage Scenario

- Create an Ecore model that represents the domain you are working on
  - Import UML (e.g. Rose .mdl file)
  - Import XML Schema
  - Import annotated Java interfaces
  - Create Ecore model directly using EMF's Ecore editor or a graphical editor
- Generate Java code for model
- Prime the model with instance data using generated EMF model editor
- Iteratively refine model (and regenerate code) and develop Java application
  - You will use the EMF generated code to implement the use cases of your application
- Optionally, use EMF.Edit to build customized user interface

EMF Architecture

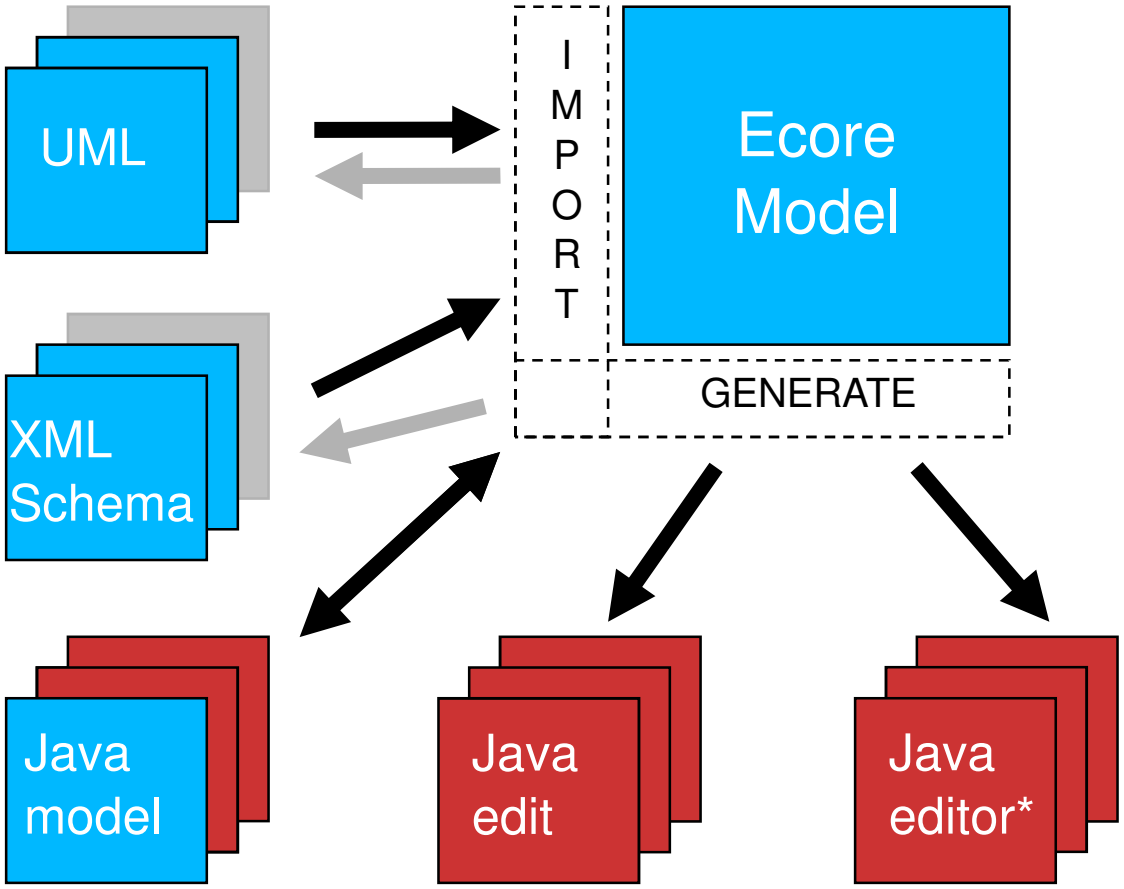


EMF Components

- EMF Core
  - Ecore metamodel
  - Model change notification & validation
  - Persistence and serialization
  - Reflection API
  - Runtime support for generated models
- EMF Edit
  - Helps integrate models with a rich user interface
  - Used to build editors and viewers for your model
  - Includes default reflective model editor
- EMF Codegen
  - Code generator for core and edit based components
  - Extensible model importer framework

AOT LAB

EMF Tools: Model Import and Generation



Generator Features:

- Customizable JSP-like templates (JET)
- JDT-integrated, command-line, or Ant
- Fully supports regeneration and merge

* Eclipse IDE-integrated or RCP-based

EMF Model Importers

- UML
  - Rational Rose .mdl file
  - Eclipse UML2 project provides importer for .uml2
- Annotated Java
  - Java interfaces representing modeled classes
  - Javadoc annotations using @model tags to express model properties not captured by method declarations
  - Lowest cost approach
- XML Schema
  - Describes the data of the modeled domain
  - Provides richer description of the data, which EMF exploits
- Ecore model (*.ecore file)
  - Just creates the generator model (discussed later)
  - Also handles EMOF (*.emof)

AOT LAB

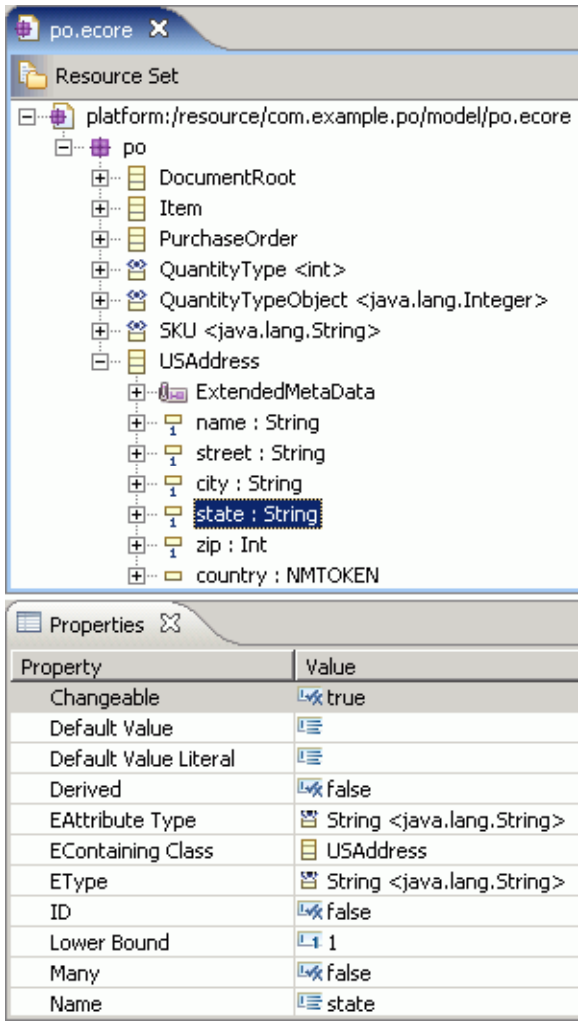
Ecore Model Creation

- An Ecore model is created within an Eclipse project via a wizard
- Input: one of the model specifications from the previous slide
- Output:
  - modelname.ecore
    - Ecore model file in XMI format
    - Canonical form of the model
  - modelname.genmodel
    - A “generator model” for specifying generator options
    - Decorates .ecore file
    - EMF code generator is an EMF .genmodel editor
    - Automatically kept in synch with .ecore file

AOT LAB

Ecore Model Editor

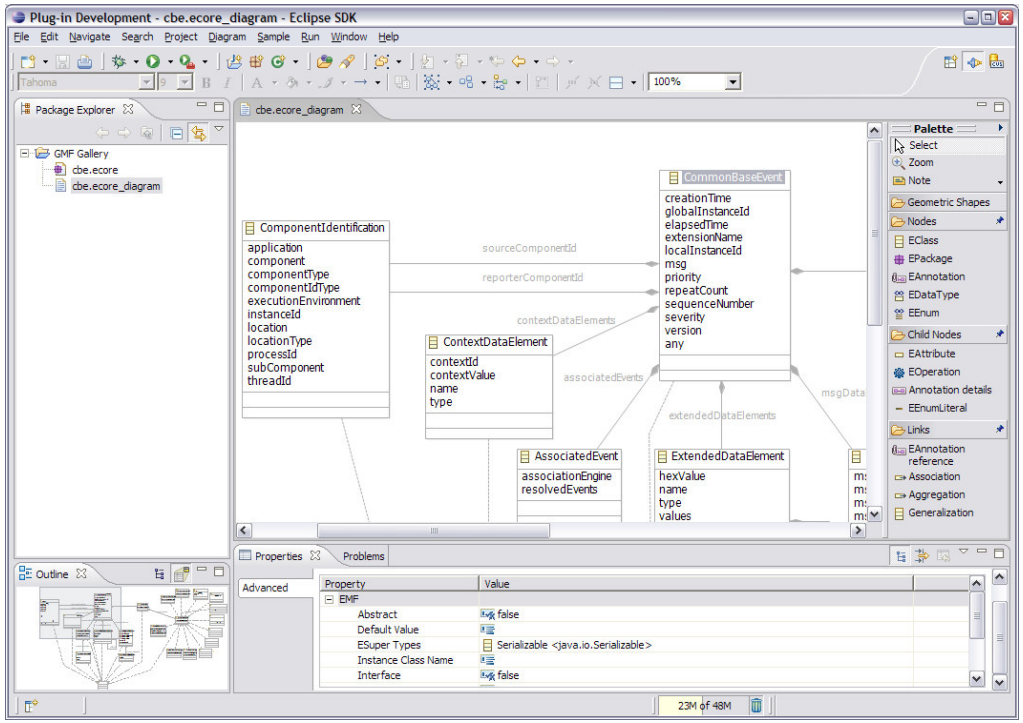
- A generated (and customized) EMF editor for the Ecore model
- Create, delete, etc. model elements (EClass, EAttribute, EReference, etc.) using pop-up actions in the editor's tree
- Set names, etc. in the Properties view



AOT LAB

Ecore Model Editor

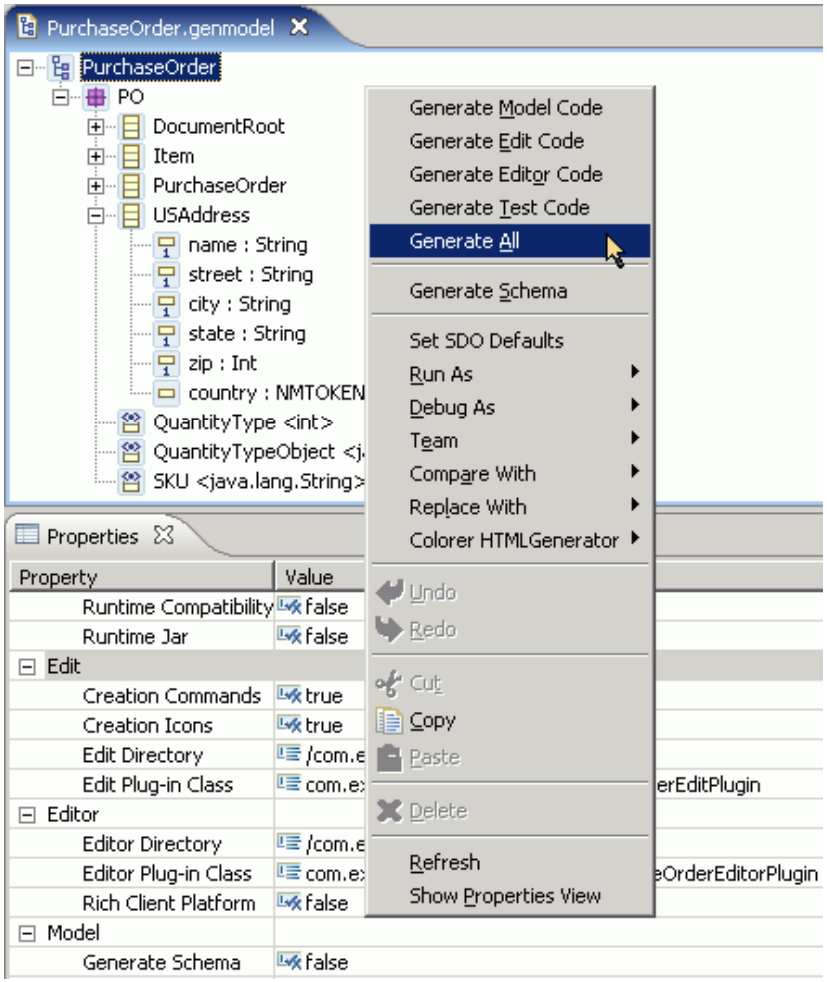
- A graphical editor is a better approach
  - GMF Ecore Diagram Example (http://www.eclipse.org/gmf/)
  - Omondo EclipseUML (http://www.omondo.com/)



AOT LAB

EMF Generator

- Similar layout to Ecore model editor
- Automatically keeps in synch with .ecore changes
- Generate code with pop-up menu actions
  - Generate Model Code
  - Generate Edit Code
  - Generate Editor Code
  - Generate Test Code
  - Generate All
- Code generation options in Properties view
- Generator > Reload to reload .genmodel and .ecore files from original model form



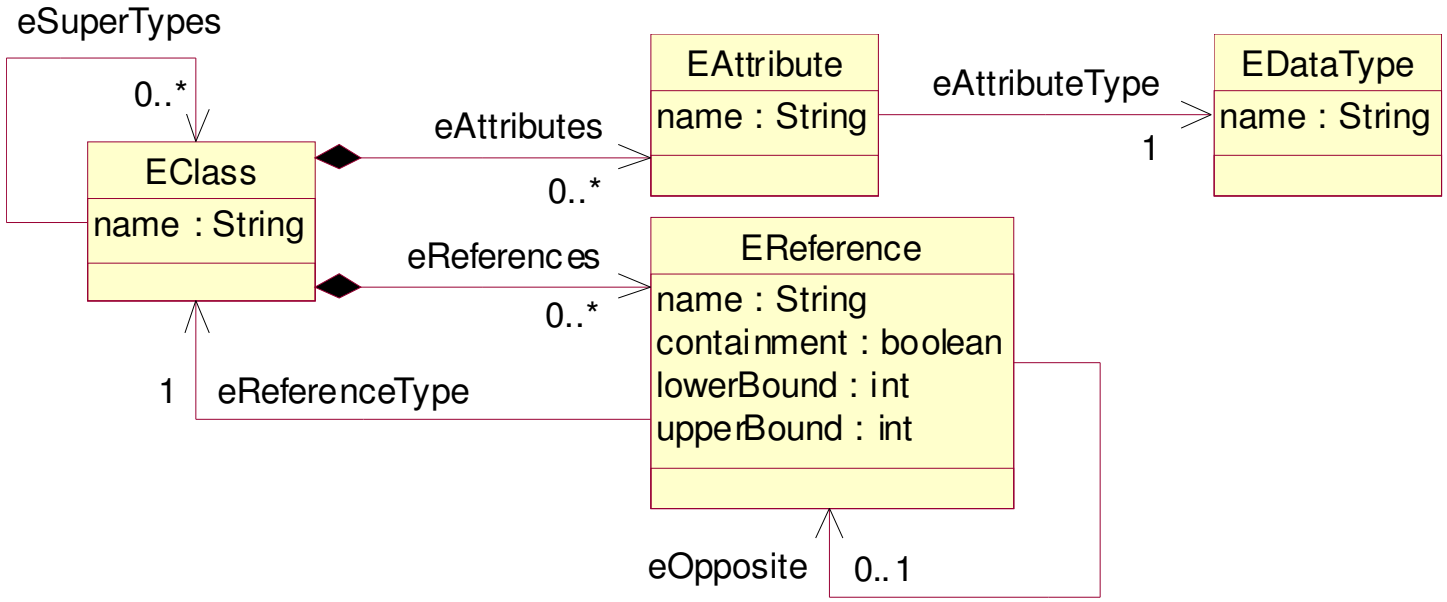
AOT LAB

Summary of Generated Artifacts

- Model
  - Interfaces and classes
  - Type-safe enumerations
  - Package (metadata)
  - Factory
  - Switch utility
  - Adapter factory base
  - Validator
  - Custom resource
  - XML Processor
- Edit (UI independent)
  - Item providers
  - Item provider adapter factory
- Editor
  - Model Wizard
  - Editor
  - Action bar contributor
  - Advisor (RCP)
- Tests
  - Test cases
  - Test suite
  - Stand-alone example
- Manifests, plug-in classes, properties, icons...

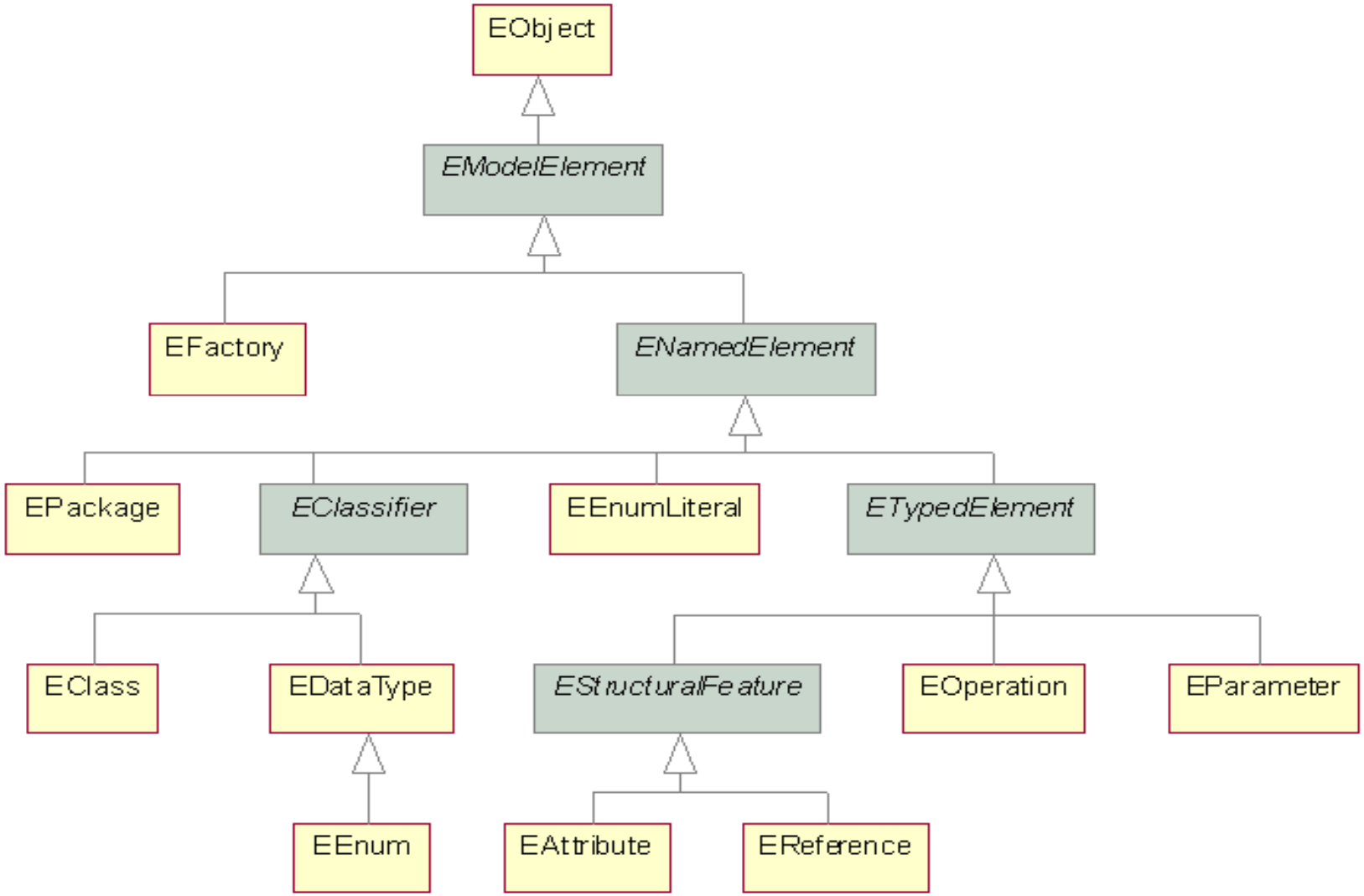
The Ecore (Meta) Model

- Ecore is EMF's model of a model
  - Also called a “metamodel”
  - Persistent representation is XMI



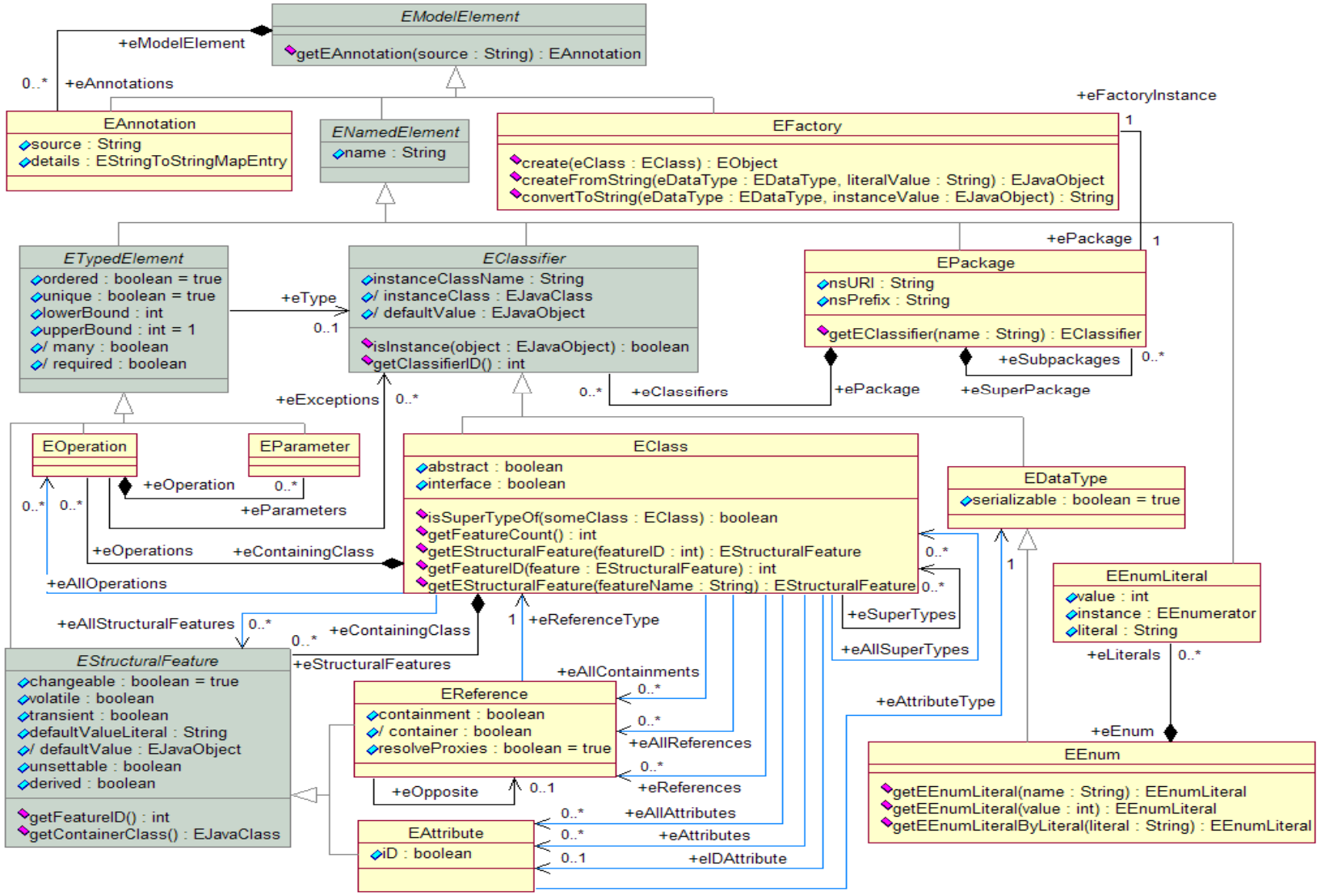
AOT LAB

The Ecore Metamodel



AOT LAB

The Ecore Metamodel



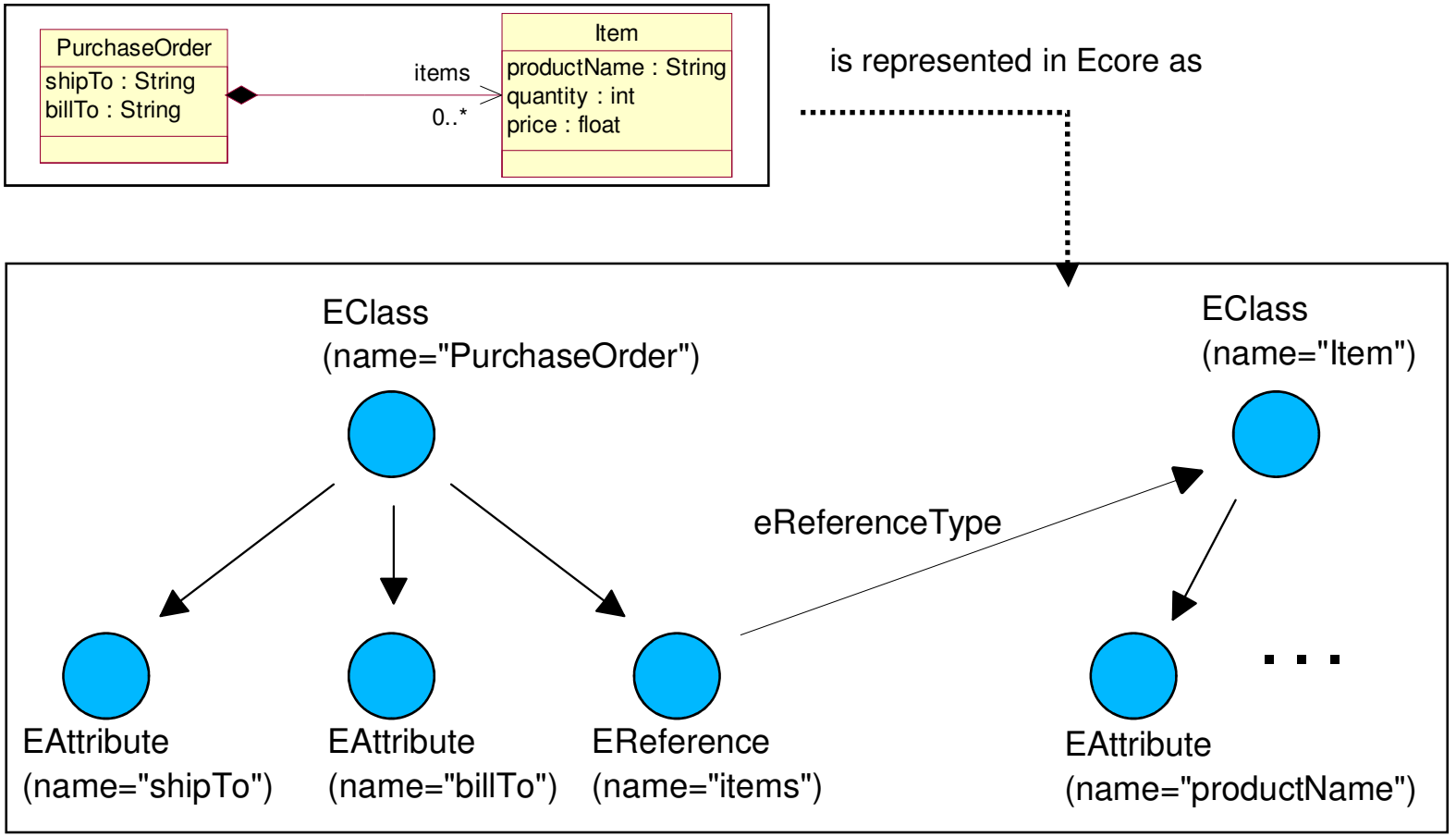
AOT LAB

Partial List of Ecore Data Types

Ecore Data Type	Java Primitive Type or Class
EBoolean	boolean
EChar	char
EFloat	float
EString	java.lang.String
EByteArray	byte[ ]
EBooleanObject	java.lang.Boolean
EFloatObject	java.lang.Float
EJavaObject	java.lang.Object

AOT LAB

Ecore Model for Purchase Orders



Purchase Order Ecore XMI

```
<eClassifiers xsi:type="ecore:EClass"
  name="PurchaseOrder">
 <eReferences name="items" eType="#//Item"
   upperBound="-1" containment="true"/>
 <eAttributes name="shipTo"
   eType="ecore:EDataType http:...Ecore#//EString"/>
 <eAttributes name="billTo"
   eType="ecore:EDataType http:...Ecore#//EString"/>
</eClassifiers>
```

- Alternate serialization format is EMOF (Essential MOF) XMI
  - Part of OMG Meta Object Facility (MOF) 2.0 standard (http://www.omg.org/docs/ptc/04-10-15.pdf)

Code Generation

- EMF framework is lightweight
  - Generated code is clean, simple, efficient
- EMF can generate
  - Model implementation
  - UI-independent edit support
  - Editor and views for Eclipse IDE-integrated or RCP application
  - JUnit test skeletons
  - Manifests, plug-in classes, properties, icons, etc.

Generated Model Code

- Interface and implementation for each modeled class
  - Includes get/set accessors for attributes and references

```
public interface PurchaseOrder extends EObject
{
  String getShipTo();
  void setShipTo(String value);
  String getBillTo();
  void setBillTo(String value);
  EList<Item> getItems();
}
```

- Usage example

```
order.getItems().add(item);
```

Generated Model Code

- Factory to create instances of model objects

```
POFactory factory = POFactory.eINSTANCE;
PurchaseOrder order = factory.createPurchaseOrder();
```

- Package class provides access to metadata

```
POPackage poPackage = POPackage.eINSTANCE;
EClass itemClass = poPackage.getItem();
  //or POPackage.Literals.ITEM

EAttribute priceAttr = poPackage.getItem_Price();
  //or
itemClass.getEStructuralFeature(POPackage.ITEM__PRICE)
  //or POPackage.Literals.ITEM__PRICE
```

- Also generated: switch utility, adapter factory base, validator, custom resource, XML processor

Generated Edit/Editor Code

- Viewing/editing code divided into two parts
  - UI-independent code
    - Item providers (adapters)
    - Item provider adapter factory
  - UI-dependent code
    - Model creation wizard
    - Editor
    - Action bar contributor
    - Advisor (RCP)
  - By default each part is placed in a separate Eclipse plug-in

Dynamic EMF

- Ecore models can be defined dynamically in memory
  - No generated code required
  - Dynamic implementation of reflective EObject API provides same runtime behavior as generated code
  - Also supports dynamic subclasses of generated classes
- All EMF model instances, whether generated or dynamic, are treated the same by the framework
- A dynamic Ecore model can be defined by
  - Instantiating model elements with the Ecore API
  - Loading from a .ecore file

Dynamic EMF Example

- Model definition using the Ecore API

```
EPackage poPackage =
EcoreFactory.eINSTANCE.createEPackage();
poPackage.setName("po");
poPackage.setNsURI("http://www.example.com/PurchaseOrder");

EClass poClass = EcoreFactory.eINSTANCE.createEClass();
poClass.setName("PurchaseOrder");
poPackage.getEClassifiers().add(poClass);

EAttribute billTo =
EcoreFactory.eINSTANCE.createEAttribute();
billTo.setName("billTo");
billTo.setEType(EcorePackage.eINSTANCE.getEString());
poClass.getEStructuralFeatures().add(billTo);
...

EObject po = EcoreUtil.create(poClass);
po.eSet(billTo,"123 Elm St.");
```

AOT LAB

Agent and **O**bject **T**echnology **Lab**
Dipartimento di Ingegneria dell'Informazione
Università degli Studi di Parma

Eclipse Modelling Framework (EMF)

Alessandro Negri
negri@ce.unipr.it
http://www.ce.unipr.it/people/negri/