

UML Superstructure as XML document (1)

```
<?xml version="1.0" encoding="UTF-8"?>
<cmof:Package xmi:version="2.1"
  xmlns:xmi="http://schema.omg.org/spec/XMI/2.1"
  xmlns:cmof="http://schema.omg.org/spec/MOF/2.0/cmof.xml"
  xmi:id="_0" name="UML">
  <ownedMember xmi:type="cmof:Package" xmi:id="Actions" name="Actions">
    <packageImport xmi:type="cmof:PackageImport"
      xmi:id="Actions-_packageImport.0"
      importedPackage="Activities"/>
    <ownedMember xmi:type="cmof:Package" xmi:id="Actions-CompleteActions"
      name="CompleteActions">
      <packageImport xmi:type="cmof:PackageImport"
        xmi:id="Actions-CompleteActions-_packageImport.0"
        importedPackage="StateMachines-BehaviorStateMachines"/>
      <packageImport xmi:type="cmof:PackageImport"
        xmi:id="Actions-CompleteActions-_packageImport.1"
        importedPackage="Classes-AssociationClasses"/>
      <packageImport xmi:type="cmof:PackageImport"
        xmi:id="Actions-CompleteActions-_packageImport.2"
        importedPackage="Classes-Kernel"/>
      <packageImport xmi:type="cmof:PackageImport"
        xmi:id="Actions-CompleteActions-_packageImport.3"
        importedPackage="CommonBehaviors-BasicBehaviors"/>
  </ownedMember>
  </ownedMember>
</cmof:Package>
```

UML Superstructure as XML document (2)

```
<ownedMember xmi:type="cmof:Class"
  xmi:id="Actions-CompleteActions-ReadExtentAction"
  name="ReadExtentAction" superClass="Actions-BasicActions-Action">
  <ownedComment xmi:type="cmof:Comment"
    xmi:id="Actions-CompleteActions-ReadExtentAction-_ownedComment.0"
    annotatedElement="Actions-CompleteActions-ReadExtentAction">
    <body>A read extent action is an action that retrieves the current
      instances of a classifier.</body>
  </ownedComment>
  <ownedRule xmi:type="cmof:Constraint"
    xmi:id="Actions-CompleteActions-ReadExtentAction-type_is_classifier"
    name="type_is_classifier"
    constrainedElement="Actions-CompleteActions-ReadExtentAction">
    <ownedComment ...
      <body>The type of the result output pin is the classifier.</body>
    </ownedComment>
    <specification xmi:type="cmof:OpaqueExpression"
      xmi:id="...-ReadExtentAction-type_is_classifier-_specification">
      <language>OCL</language>
      <body>true</body>
    </specification>
  </ownedRule>
  ...
</cmof:Package>
```

UML model as XMI document

```
<?xml version='1.0' encoding='UTF-8'?>
<xmi:XMI xmi:version='2.1' xmlns:uml='http://schema.omg.org/spec/UML/2.1.2'
        xmlns:xmi='http://schema.omg.org/spec/XMI/2.1'>
<uml:Model xmi:id='eee_1045467100313_135436_1' name='Data' visibility='public'>
  <packagedElement xmi:type='uml:Class' xmi:id='_477' name='Car' visibility='public'>
    <ownedAttribute xmi:type='uml:Property' xmi:id='_628' name='owner'
      visibility='private' type='_498' association='_627'>
      <upperValue xmi:type='uml:LiteralUnlimitedNatural' xmi:id='_680' visibility='public' value='1' />
      <lowerValue xmi:type='uml:LiteralInteger' xmi:id='_679' visibility='public' value='1' />
    </ownedAttribute>
    <ownedAttribute xmi:type='uml:Property' xmi:id='_681' name='manufacturer' visibility='private'>
      <type xmi:type='uml:PrimitiveType' href='http://schema.omg.org/spec/UML/2.0/uml.xml#String' />
    </ownedAttribute>
  </packagedElement>
  <packagedElement xmi:type='uml:Class' xmi:id='_498' name='Owner' visibility='public'>
    <ownedAttribute xmi:type='uml:Property' xmi:id='_629' name='ownedCars'
      visibility='private' type='_477' association='_627'>
      <upperValue xmi:type='uml:LiteralUnlimitedNatural' xmi:id='_678' visibility='public' value='-1' />
      <lowerValue xmi:type='uml:LiteralUnlimitedNatural' xmi:id='_677' visibility='public' value='-1' />
    </ownedAttribute>
    <ownedAttribute xmi:type='uml:Property' xmi:id='_685' name='name' visibility='private'>
      <type xmi:type='uml:PrimitiveType' href='http://schema.omg.org/spec/UML/2.0/uml.xml#String' />
    </ownedAttribute>
  </packagedElement>
  <packagedElement xmi:type='uml:Association' xmi:id='_627' visibility='public'>
    <memberEnd xmi:idref='_628' />
    <memberEnd xmi:idref='_629' />
  </packagedElement>
</uml:Model>
</xmi:XMI>
```



(MagicDraw 15.1, simplified)



Schema production

- Schema production defined by set of rules
 - Typically intended to be implemented, not for human usage
 - EBNF (Extended Backus-Naur form) rules are supplied
- Control of schema production by MOF tags
 - `nsPrefix`
 - `nsURI`
 - `useSchemaExtensions`
 - `enforceMinimumMultiplicity`
 - `enforceMaximumMultiplicity`
 - ...



Schema production rules: Classes and properties

- **Meta-model class**
 - Mapped to `xsd:element` and `xsd:complexType` with same name as metamodel class
- **Property of meta-model class**
 - Mapped to `xsd:element` and `xsd:attribute` if simple data type and the cardinality of the property is `[1..1]` or `[0..1]`
 - Mapped to `xsd:element` if `xsd:complexType`
 - Note: only possible in CMOF (Complete MOF)

Schema production: Example (1)

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3c.org/2001/XMLSchema"
            xmlns:xmi="http://www.omg.org/XMI"
            targetNamespace="http://www.example.org/CDs"
            xmlns:cds="http://www.example.org/CDs">
  <xsd:import namespace="http://schema.omg.org/spec/XMI/2.1"
            schemaLocation="XMI.xsd"/>
  <xsd:complexType name="CD">
    <xsd:choice minOccurs="0" maxOccurs="unbounded">
      <xsd:element name="title" type="xsd:string"/>
      <xsd:element name="artist" type="xsd:string"/>
      <xsd:element name="num_tracs" type="xsd:integer"/>
      <xsd:element ref="xmi:Extension"/>
    </xsd:choice>
    <xsd:attribute ref="xmi:id"/>
    <xsd:attributeGroup ref="xmi:ObjectAttribs"/>
    <xsd:attribute name="title" type="xsd:string"/>
    <xsd:attribute name="artist" type="xsd:string"/>
    <xsd:attribute name="num_tracs" type="xsd:integer"/>
  </xsd:complexType>
  <xsd:element name="CD" type="cds:CD"/>
</xsd:schema>
```

 CD

- ▣ title : String
- ▣ artist : String
- ▣ num_tracs : Integer

Document production: Example (1)

```
<?xml version="1.0" encoding="UTF-8"?>
<cds:CD xmlns:cds="http://www.example.org/CDs"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:xmi="http://schema.omg.org/spec/XMI/2.1"
        xsi:schemaLocation="http://www.example.org/CDs"

        artist="Bruce Springsteen"
        title="Born To Run" num_tracs="8"
        xmi:id="_1">
</cds:CD>
```

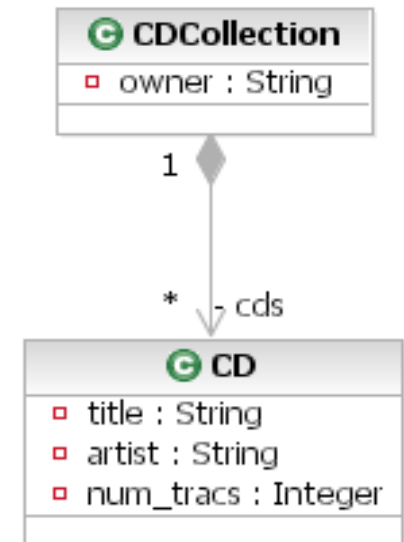
Born to Run
Bruce Springsteen
8 tracks

Schema production rules: Relationships

- **Association between classes**
 - An `xsd:element` is created with name set to the name of the reference and type set to the type name of the referenced class.
 - Multiplicity definitions are included if the appropriate parameters are set at the time of generation.
 - MOF tags `enforceMinimumMultiplicity` and `enforceMaximumMultiplicity`
- **Inheritance**
 - Problem
 - XML schemas only allow single inheritance
 - MOF allows multiple inheritance
 - Solution
 - XMI uses a copy down strategy to implement inheritance
 - For multiple inheritance properties that occur more than once in the hierarchy are included only once in the schema.
 - MOF tag `useSchemaExtensions` (if single inheritance only)

Schema production: Example (2)

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns:xmi="http://schema.omg.org/spec/XMI/2.1"
            targetNamespace="http://www.example.org/CDLib"
            xmlns:cdlib="http://www.example.org/CDLib">
  <xsd:import namespace="http://schema.omg.org/spec/XMI"
            schemaLocation="XMI.xsd"/>
  <xsd:complexType name="CD">
    ...
  </xsd:complexType>
  <xsd:element name="CD" type="cdlib:CD"/>
  <xsd:complexType name="CDCollection">
    <xsd:choice maxOccurs="unbounded" minOccurs="0">
      <xsd:element name="cds" type="cdlib:CD" />
      <xsd:element ref="xmi:Extension" />
    </xsd:choice>
    <xsd:attribute ref="xmi:id" />
    <xsd:attributeGroup ref="xmi:ObjectAttribs" />
    <xsd:attribute name="owner" type="xsd:string" />
  </xsd:complexType>
  <xsd:element name="CDCollection" type="cdlib:CDCollection"/>
</xsd:schema>
```



Document production: Example (2)

```
<?xml version="1.0" encoding="UTF-8"?>
<xmi:XMI xmi:version='2.1'
  xmlns:xmi="http://schema.omg.org/spec/XMI/2.1"
  xmlns:cdlib="http://www.example.org/CDLib"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.example.org/CDLib">

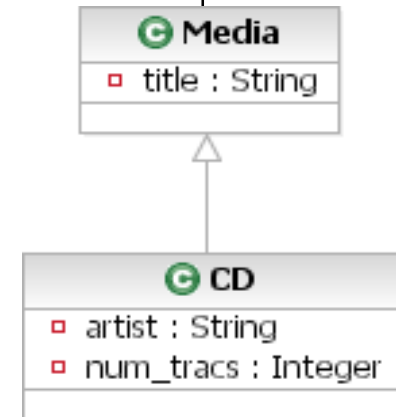
  <cdlib:CDCollection
    owner="Jon Doe"
    xmi:id="_1">

    <cds artist="Bruce Springsteen"
      title="Born To Run" num_tracs="8"
      xmi:id="_2">
    <cds artist="U2"
      title="Achtung Baby" num_tracs="13"
      xmi:id="_3">
  </cdlib:CDCollection>
</xmi:XMI>
```



Schema production: Example (3)

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns:xmi="http://schema.omg.org/spec/XMI/2.1"
            targetNamespace="http://www.example.org/MediaLib"
            xmlns:medlib="http://www.example.org/MedLib">
  <xsd:import .../>
  <xsd:complexType name="Media">
    <xsd:choice maxOccurs="unbounded" minOccurs="0">
      <xsd:element name="title" type="xsd:string"/>
      <xsd:element ref="xmi:Extension"/>
    </xsd:choice>
    ...
    <xsd:attribute name="title" type="xsd:string"/>
  </xsd:complexType>
  <xsd:element name="Media" type="medlib:Media"/>
  <xsd:complexType name="CD">
    <xsd:attribute name="title" type="xsd:string"/>
    <xsd:attribute name="artist" type="xsd:string"/>
    ...
  </xsd:complexType>
  <xsd:element name="CD" type="medlib:CD"/>
</xsd:schema>
```

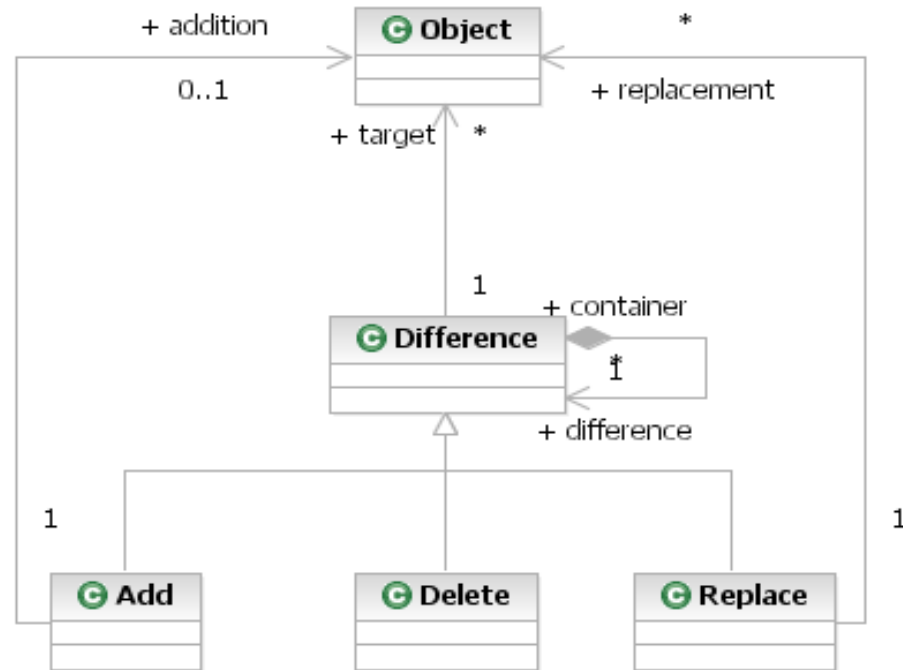


Differences

- XMI allows you to express differences in XMI documents
 - can be used to just communicate the changes in a document rather than the whole document

- **Types of differences**

- Add
- Delete
- Replace



Differences: Example

```
<xmi:XMI xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI">
  <MediaCollection xmi:id="_1">
    <items xmi:id="_2" name="Purple Rain" xmi:type="CD"/>
    <items xmi:id="_3" name="Pulp Fiction" xmi:type="DVD"/>
  </MediaCollection>
</xmi:XMI>
```

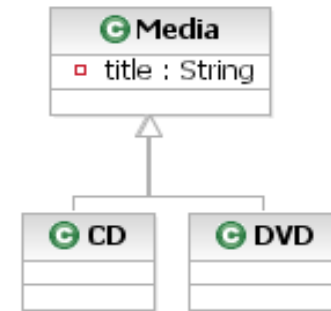
collection.xmi

```
<xmi:XMI xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI">
  <xmi>Delete>
    <target href="collection.xmi#_2"/>
  </xmi>Delete>
  <xmi:Add addition="NM1" position="1">
    <target href="collection.xmi#_1"/>
  </xmi:Add>
  <CD xmi:id="NM1" name="Thunder Road"/>
</xmi:XMI>
```

differences

```
<xmi:XMI xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI">
  <MediaCollection xmi:id="_1">
    <items xmi:id="NM1" name="Thunder Road" xmi:type="CD"/>
    <items xmi:id="_3" name="Pulp Fiction" xmi:type="DVD"/>
  </MediaCollection>
</xmi:XMI>
```

result



Tool interoperability

- Use of XMI for tool interoperability is not always straightforward
- Different XMI versions and different metamodel versions
- Take XMI for UML as an example
 - XMI 1.0 for UML 1.3
 - XMI 1.1 for UML 1.3
 - XMI 1.2 for UML 1.4
 - XMI 2.0 for UML 1.4
 - XMI 2.1 for UML 2.0
- Common to see that UML tools have a “choose XMI format” dialog when exporting to XMI

Eclipse Modeling Framework

Eclipse Modeling Framework (EMF)

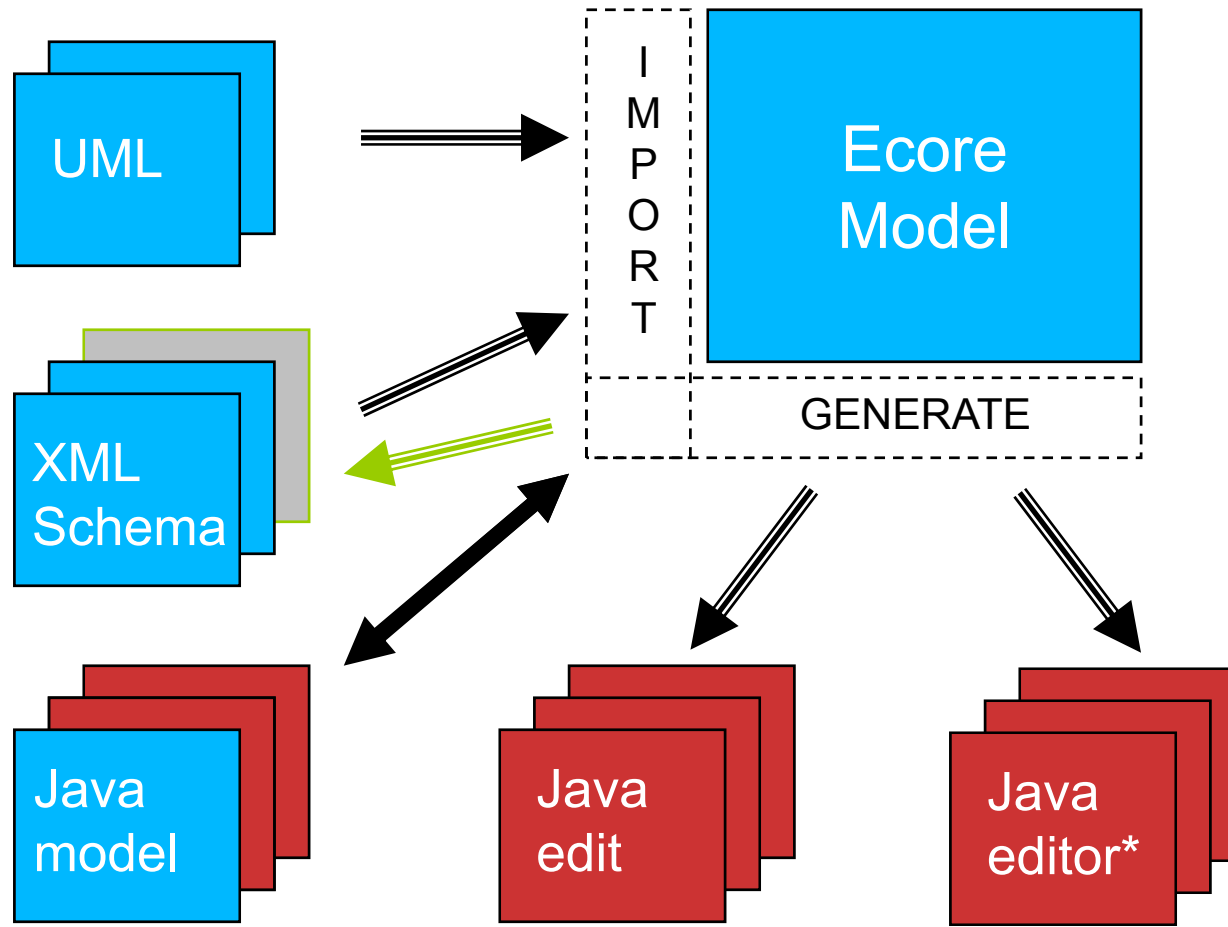
- Modelling — more than just documentation
- Just about every program manipulates some data model
 - It might be defined using Java, UML, XML Schemas, or some other definition language
- EMF aims to extract this intrinsic “model” and generate some of the implementation code
 - Can be a tremendous productivity gain
- EMF is one implementation of MOF (though it has differences)
 - EMF \approx EMOF

<http://www.eclipse.org/emf/>

EMF

- EMF is a **modelling framework** and **code generation facility** for building tools and other applications based on a structured data model.
- From a model specification described in XMI, EMF provides
 - tools and runtime support to **produce a set of Java classes for the model**,
 - **adapter classes that enable viewing** and **command-based editing** of the model,
 - and a **basic editor**.
- Models can be specified using
 - Annotated Java
 - XML documents
 - Modelling tools like Rational Rose, MagicDraw, ...
 - ...
- **EMF provides the foundation for interoperability** with other EMF-based tools and applications.

EMF architecture: Model import and generation



Generator features:

- Customizable JSP-like templates (JET)
- Command-line or integrated with Eclipse JDT
- Fully supports regeneration and merge

* requires Eclipse to run

EMF — Fundamental Pieces

- **EMF**

- The core EMF framework includes a **meta-model (Ecore)**
 - for describing models
 - **runtime support** for the models including change notification,
 - **persistence support** with default XMI serialization,
 - **reflective API** for manipulating EMF objects generically.

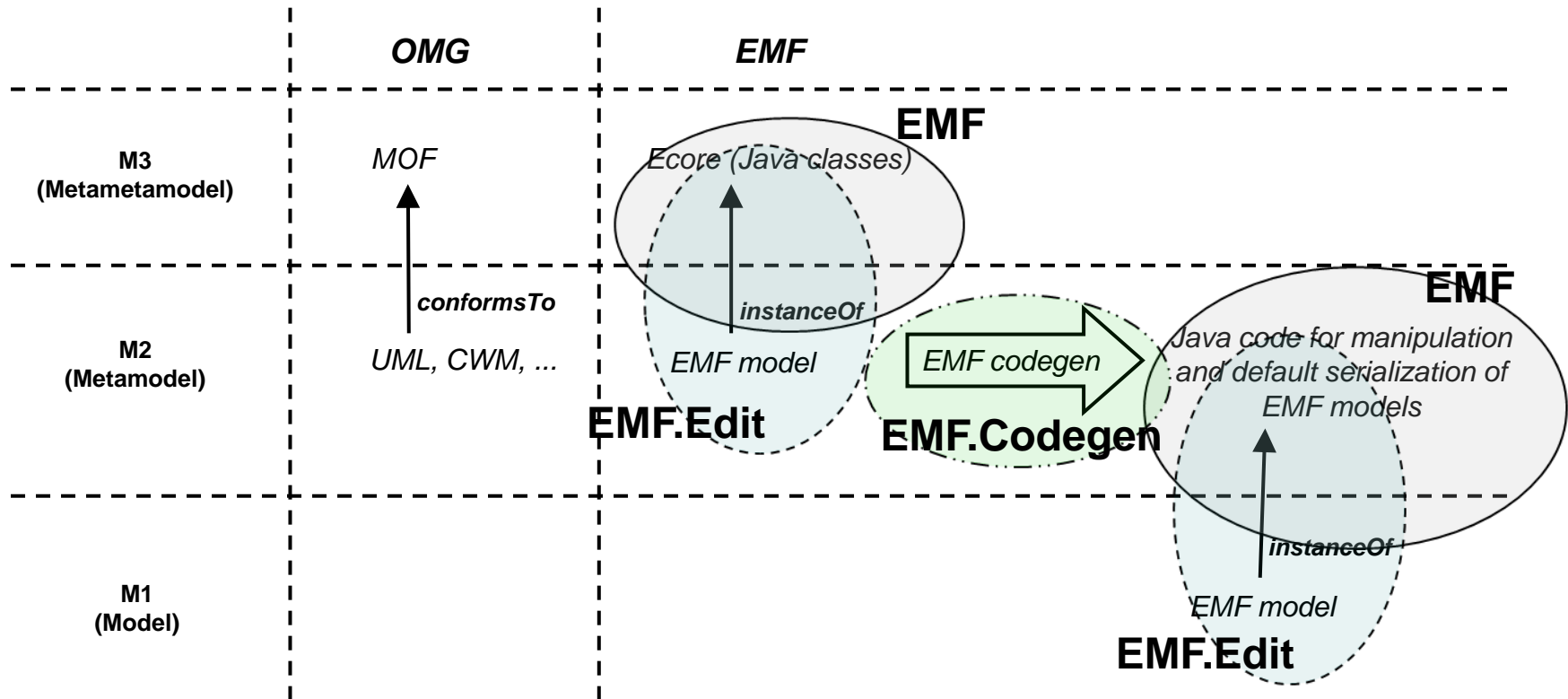
- **EMF.Edit**

- Generic reusable **classes for building editors** for EMF models.

- **EMF.Codegen**

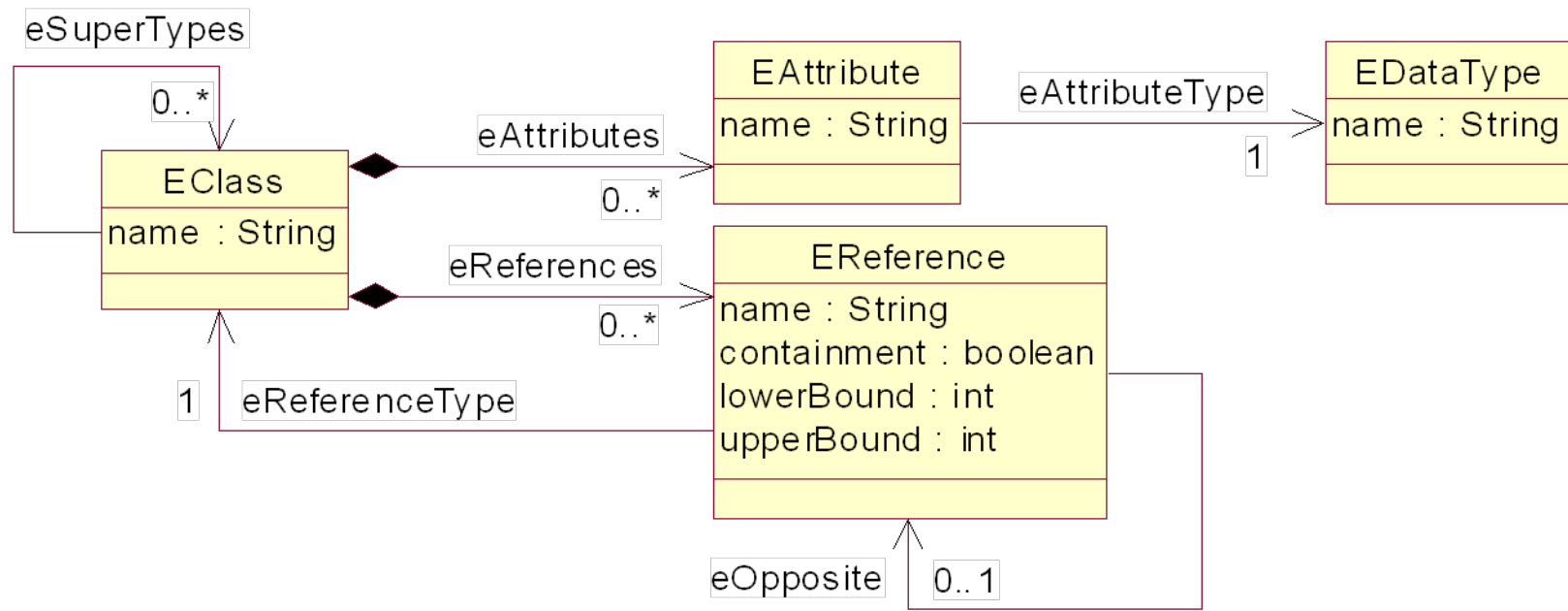
- Capable of generating everything needed **to build a complete editor** for an EMF model.
- **Includes a GUI** from which generation options can be specified, and generators can be invoked.

EMF in the meta-modelling architecture



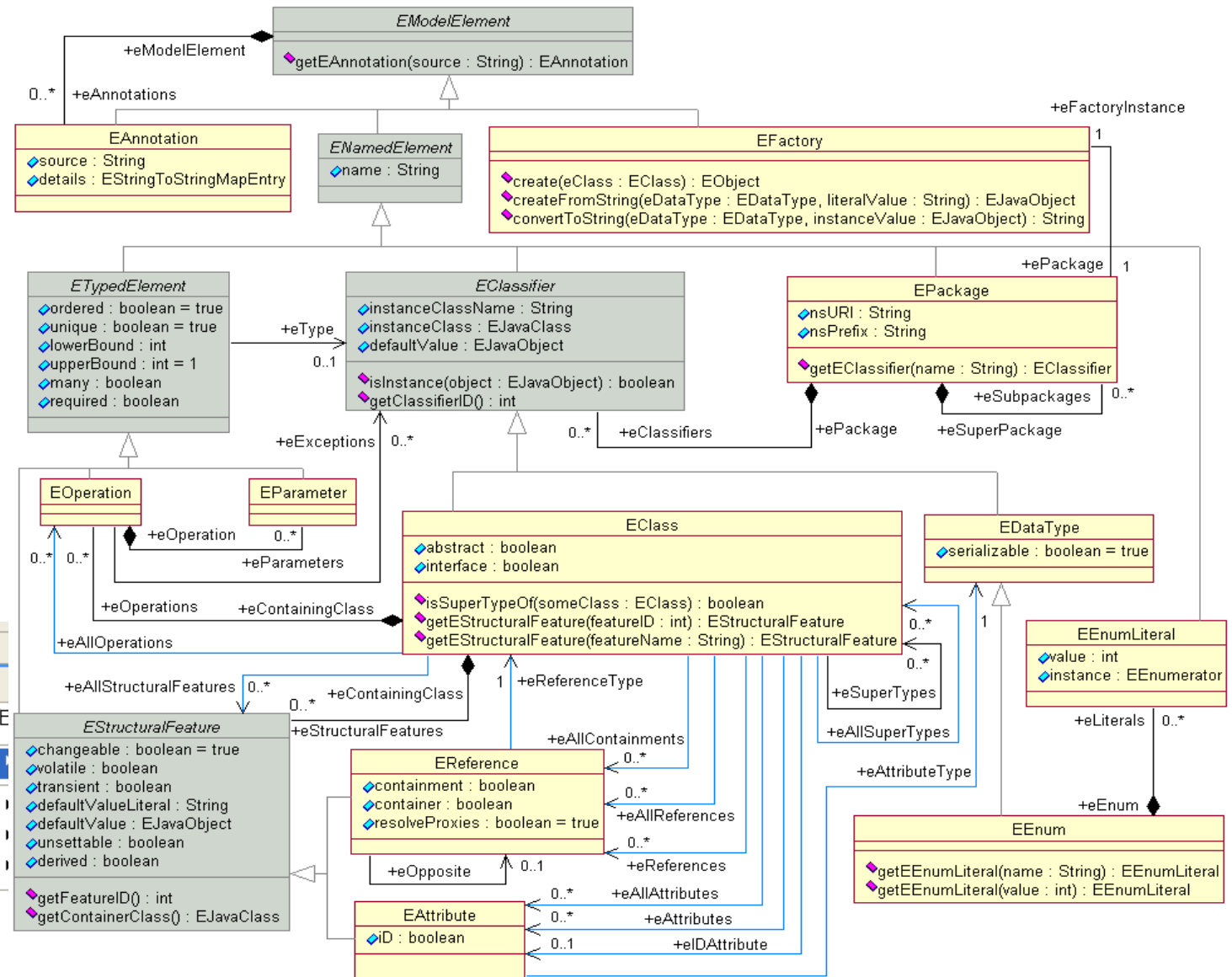
EMF architecture: Ecore

- Ecore is EMF's model of models (meta-model)
 - Persistent representation is XMI
 - Can be seen as an implementation of UML Core::Basic

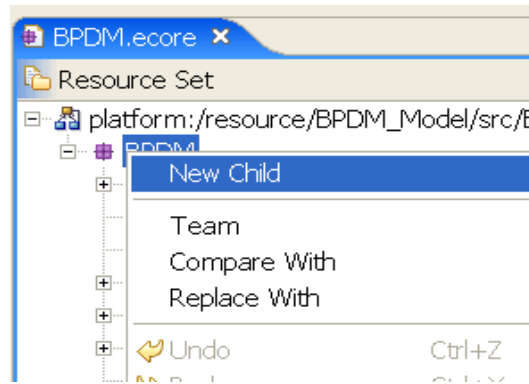


Ecore: Overview

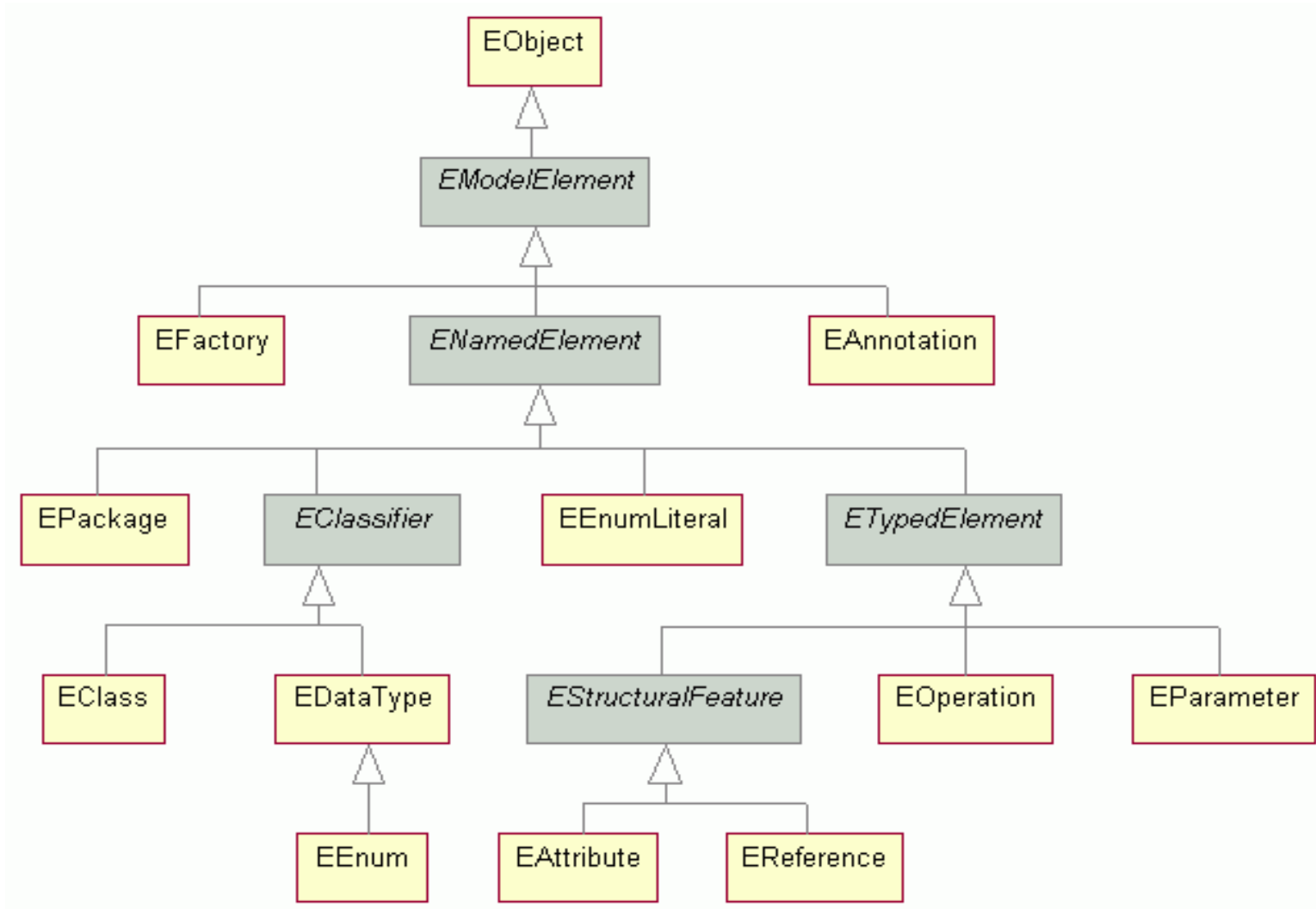
used for
meta-modelling



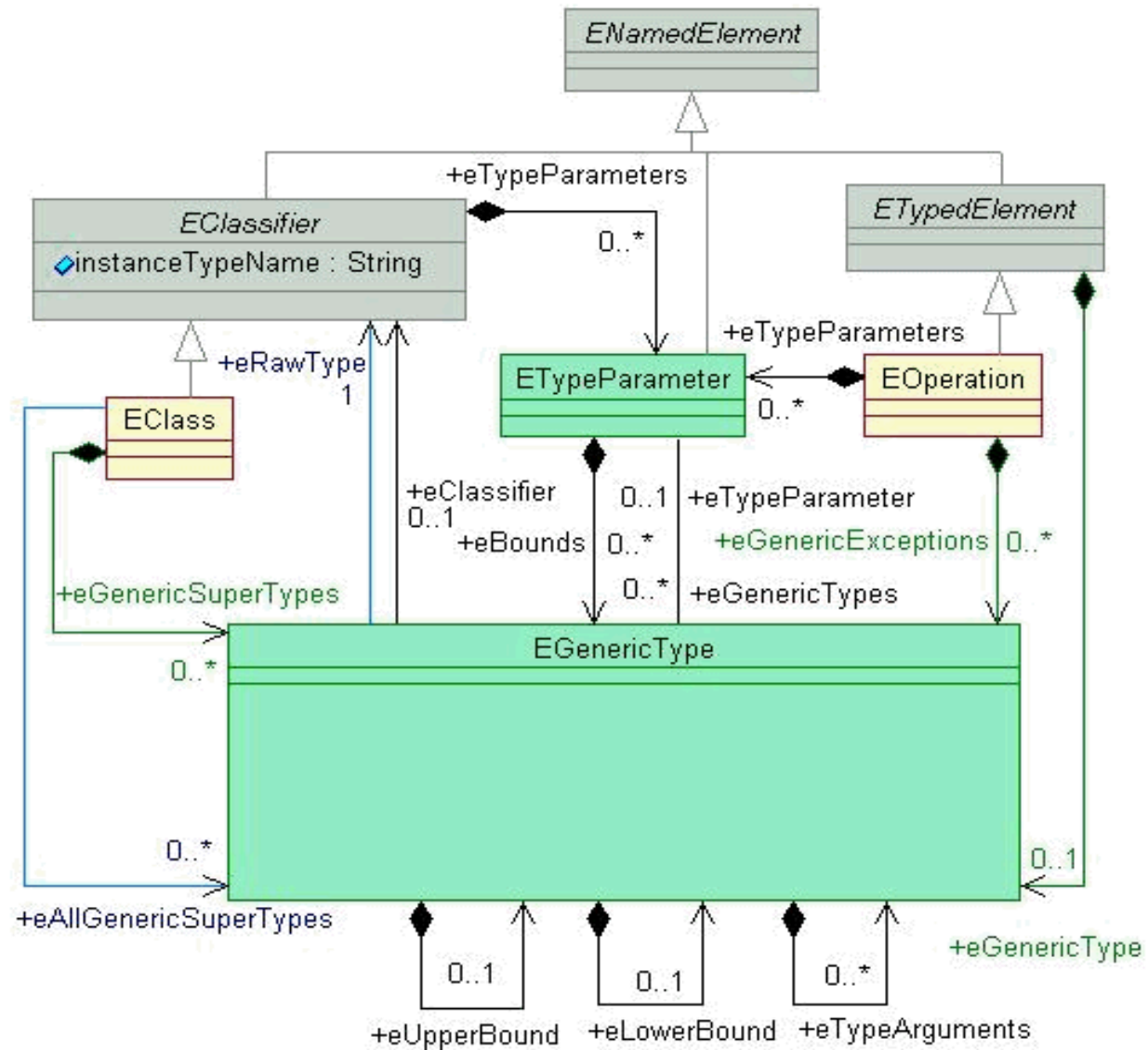
Modelling with UML, with semantics



Ecore: Inheritance hierarchy



Ecore: Generics



EMF model definition (1)

- Specification of an application's data
 - Object attributes
 - Relationships (associations) between objects
 - Operations available on each object
 - Simple constraints (e.g., multiplicity) on objects and relationships

EMF model definition: Programming (1)

```
import java.io.*;
import java.util.*;
import org.eclipse.emf.ecore.*;
import org.eclipse.emf.common.util.URI;
import org.eclipse.emf.ecore.resource.*;
import org.eclipse.emf.ecore.resource.impl.*;
import org.eclipse.emf.ecore.xmi.impl.EcoreResourceFactoryImpl;

public class EMFTest {
    public static void main(String[] args) {
        EcoreFactory ecoreFactory = EcoreFactory.eINSTANCE;
```



factory for Ecore meta-models

EMF model definition: Programming (2)


```
EPackage aPackage =.ecoreFactory.createEPackage();
aPackage.setName("somePackage");
aPackage.setNsPrefix("pkg");
aPackage.setNsURI("urn:www.pst.ifi.lmu.de/knapp/pkg");

EClass aClass =.ecoreFactory.createEClass();
aClass.setName("SomeClass");
aPackage.getEClassifiers().add(aClass);

EAttribute anAttribute =.ecoreFactory.createEAttribute();
anAttribute.setName("someAttribute");
anAttribute.setEType(ecoreFactory.getEcorePackage().
    getEString());
aClass.getEStructuralFeatures().add(anAttribute);

EReference aReference =.ecoreFactory.createEReference();
aReference.setName("someReference");
aReference.setEType(aClass);
aClass.getEStructuralFeatures().add(aReference);
```

namespace settings



EMF model definition: Programming (3)

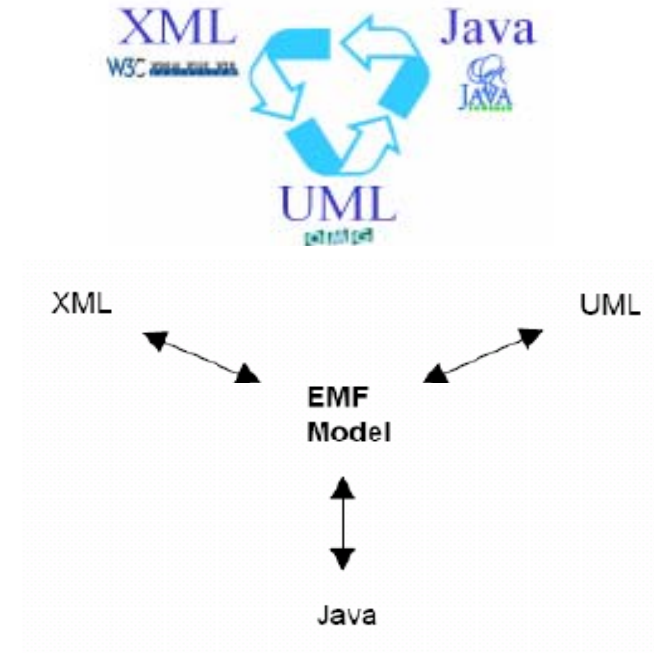
```
try {
    Resource.Factory.Registry.INSTANCE.
        getExtensionToFactoryMap().put("ecore",
            new EcoreResourceFactoryImpl());
    ResourceSet resourceSet = new ResourceSetImpl();
    Resource resource = resourceSet.
        createResource(URI.createFileURI("test.ecore"));
    resource.getContents().add(aPackage);
    StringWriter stringWriter = new StringWriter();
    URIConverter.WriteableOutputStream outputStream =
        new URIConverter.WriteableOutputStream(stringWriter, "UTF-8");
    Map<String, String> options = new HashMap<String, String>();
    resource.save(outputStream, options);
    System.out.println(stringWriter.toString());
} catch (IOException ioe) {
    ioe.printStackTrace();
}
```

for saving as Ecore meta-model

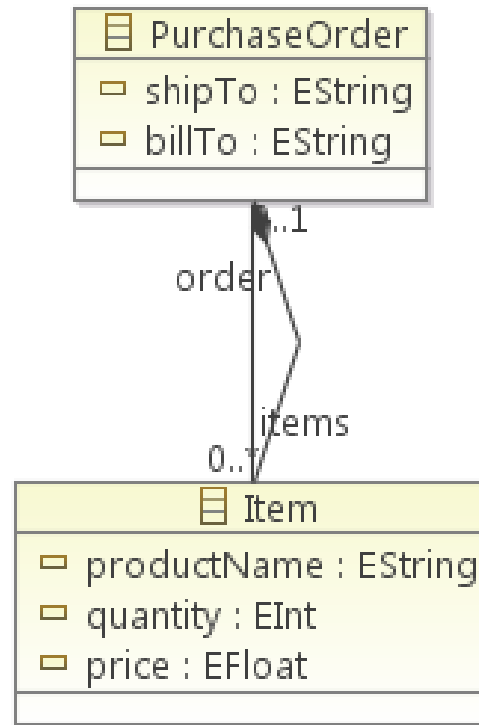
options for resources
(compress, encrypt, save only when
modified, progress monitor, &c.)

EMF model definition (2)

- Unifying Java, XML, and UML technologies
- All three forms provide the same information
 - Different visualization/representation
 - The application's "model" of the structure
- EMF models can be defined in (at least) four ways:
 1. ECore diagram
 2. Java interfaces
 3. UML Class Diagram
 4. XML Schema
- EMF can generate the others as well as the implementation code



EMF model definition: ECore diagrams



EMF model definition: Annotated Java interfaces

```
/** @model */
public interface PurchaseOrder {
    /** @model */ String getShipTo();
    /** @model */ String getBillTo();
    /** @model containment="true" opposite="order" */
    List<Item> getItems();
}

/** @model */
public interface Item {
    /** @model opposite="items" */
    PurchaseOrder getOrder();
    /** @model */ String getProductName();
    /** @model */ int getQuantity();
    /** @model */ float getPrice();
}
```

- Setter methods for attributes generated

EMF model definition: UML class diagrams



- From Rational Software Architect, Eclipse UML 2, &c.

EMF model definition: XML Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
        targetNamespace="http://www.example.org/purchase"
        xmlns:tns="http://www.example.org/purchase">
  <complexType name="PurchaseOrder">
    <sequence>
      <element name="shipTo" type="string"/>
      <element name="billTo" type="string"/>
      <element name="items" type="tns:Item"
                minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
    <attribute name="id" type="ID"/>
  </complexType>
  <complexType name="Item">
    <sequence>
      <element name="order" type="IDREF" minOccurs="1" maxOccurs="1"/>
      <element name="productName" type="string"/>
      <element name="quantity" type="int"/>
      <element name="price" type="float"/>
    </sequence>
  </complexType>
</schema>
```

EMF architecture: Ecore/XMI (1)

```
<?xml version="1.0" encoding="UTF-8"?>
<ecore:EPackage xmi:version="2.0"
  xmlns:xmi="http://www.omg.org/XMI"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:ecore="http://www.eclipse.org/emf/2002/Ecore" name="purchase"
  nsURI="http://www.example.org/purchase" nsPrefix="purchase">
  <eClassifiers xsi:type="ecore:EClass" name="Item">
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="order"
      lowerBound="1" eType="ecore:EDataType
        http://www.eclipse.org/emf/2003/XMLType#//IDREF"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute"
      name="productName" lowerBound="1"
      eType="ecore:EDataType
        http://www.eclipse.org/emf/2003/XMLType#//String"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="quantity"
      lowerBound="1" eType="ecore:EDataType
        http://www.eclipse.org/emf/2003/XMLType#//Int"/>
    <eStructuralFeatures xsi:type="ecore:EAttribute" name="price"
      lowerBound="1" eType="ecore:EDataType
        http://www.eclipse.org/emf/2003/XMLType#//Float"/>
  </eClassifiers>
```

EMF architecture: Ecore/XMI (2)

```
<eClassifiers xsi:type="ecore:EClass" name="PurchaseOrder">
  <eStructuralFeatures xsi:type="ecore:EAttribute" name="shipTo"
    lowerBound="1" eType="ecore:EDatatype
      http://www.eclipse.org/emf/2003/XMLType#//String"/>
  <eStructuralFeatures xsi:type="ecore:EAttribute" name="billTo"
    lowerBound="1" eType="ecore:EDatatype
      http://www.eclipse.org/emf/2003/XMLType#//String"/>
  <eStructuralFeatures xsi:type="ecore:EReference" name="items"
    upperBound="-1" eType="#//Item" containment="true"/>
  <eStructuralFeatures xsi:type="ecore:EAttribute" name="id"
    eType="ecore:EDatatype
      http://www.eclipse.org/emf/2003/XMLType#//ID" id="true"/>
</eClassifiers>
</ecore:EPackage>
```

- Alternative serialisation format is EMOF/XMI

EMF.Codegen: Interface completion

```
/** @model */
public interface Item extends EObject {
    /** @model opposite="items" */ PurchaseOrder getOrder();
    /** @generated */ void setOrder(PurchaseOrder value);
    /** @model */ String getProductName();
    /** @generated */ void setProductName(String value);
    /** @model */ int getQuantity();
    /** @generated */ void setQuantity(int value);
    /** @model */ float getPrice();
    /** @generated */ void setPrice(float value);
}
```

- No regeneration of implementations when changing @generated to @generated NOT

EMF.Codegen: Implementation of associations (1)

- Proper handling of binary associations
 - changes on either side of an association propagated to the other
- Special handling of composite associations
 - only a single container, stored in `eContainerFeatureID`

```
public PurchaseOrder getOrder() {  
    if (eContainerFeatureID != PurchasePackage.ITEM__ORDER)  
        return null;  
    return (PurchaseOrder)eContainer();  
}
```

EMF.Codegen: Implementation of associations (2)

```
public void setOrder(PurchaseOrder newOrder) {
    if (newOrder != eInternalContainer() ||
        (eContainerFeatureID != PurchasePackage.ITEM__ORDER &&
         newOrder != null)) {
        if (EcoreUtil.isAncestor(this, newOrder))
            throw new IllegalArgumentException("Recursive containment " +
                "not allowed for " + toString());
        NotificationChain msgs = null;
        if (eInternalContainer() != null)
            msgs = eBasicRemoveFromContainer(msgs);
        if (newOrder != null)
            msgs = ((InternalEObject)newOrder).eInverseAdd(this,
                PurchasePackage.PURCHASE_ORDER__ITEMS,
                PurchaseOrder.class, msgs);
        msgs = basicSetOrder(newOrder, msgs);
        if (msgs != null) msgs.dispatch();
    }
    else
        if (eNotificationRequired())
            eNotify(new ENotificationImpl(this, Notification.SET,
                PurchasePackage.ITEM__ORDER, newOrder, newOrder));
}
```

single container

consistent update for binary associations

EMF: Creating models with generated code

```
PurchaseFactory purchaseFactory = PurchaseFactory.eINSTANCE;  
PurchaseOrder order1 = purchaseFactory.createPurchaseOrder();  
order1.setBillTo("X");  
order1.setShipTo("Y");  
Item item1 = purchaseFactory.createItem();  
item1.setProductName("A");  
item1.setQuantity(2);  
item1.setPrice(10.0f);  
item1.setOrder(order1);  
Item item2 = purchaseFactory.createItem();  
item2.setProductName("B");  
item2.setQuantity(3);  
item2.setPrice(100.0f);  
item2.setOrder(order1);
```

← factory for purchase models


EMF: Saving models

```
ResourceSet resourceSet = new ResourceSetImpl();
resourceSet.getResourceFactoryRegistry().
    getExtensionToFactoryMap().put("xmi", new XMIResourceFactoryImpl());

URI fileURI = URI.createFileURI(new File("orders.xmi").getAbsolutePath());
Resource resource = resourceSet.createResource(fileURI);


resource.getContents().add(order1);

try {
    resource.save(System.out, Collections.EMPTY_MAP);
    resource.save(Collections.EMPTY_MAP);
}
catch (IOException ioe) {
    ioe.printStackTrace();
}
```

 **mind containment**
not resource.getContents().add(item1);

EMF: Ecore/XMI from generated code

```
<?xml version="1.0" encoding="ASCII"?>
<xmi:XMI xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI "
  xmlns:purchaseJava="http://purchaseJava.ecore">
  <purchaseJava:PurchaseOrder shipTo="Y" billTo="X">
    <items productName="A" quantity="2" price="10.0"/>
    <items productName="B" quantity="3" price="100.0"/>
  </purchaseJava:PurchaseOrder>
</xmi:XMI>
```

 **containment**

EMF: Querying with OCL (1)

```
import org.eclipse.ocl.ecore.OCL;
import org.eclipse.ocl.ParserException;
import org.eclipse.ocl.OCLInput;
import org.eclipse.ocl.ecore.Constraint;
```

```
OCL purchaseOCL = OCL.newInstance();
try {
    purchaseOCL.parse(new OCLInput("package purchaseJava " +
        "context Item " +
        "inv: price < 100.0 " +
        "endpackage"));
    for (Constraint constraint : purchaseOCL.getConstraints()) {
        System.out.println(purchaseOCL.check(item2, constraint));
    }
}
catch (ParserException e) {
    e.printStackTrace();
}
```

EMF: Querying with OCL (2)

```
import org.eclipse.ocl.ecore.OCL;
import org.eclipse.ocl.ParserException;
import org.eclipse.ocl.expressions.OCLExpression;
import org.eclipse.ocl.helper.OCLHelper;
```

parametric in classifier, operation, property, and constraint representation
of the meta-model

```
OCL purchaseOCL = OCL.newInstance();
OCLHelper<EClassifier, ?, ?, ?> purchaseOCLHelper =
    purchaseOCL.createOCLHelper();
purchaseOCLHelper.setContext(PurchaseJavaPackage.Literals.ITEM);
try {
    OCLExpression<EClassifier> priceExpression =
        purchaseOCLHelper.createQuery("price");
    System.out.println(purchaseOCL.evaluate(item1, priceExpression));
}
catch (ParserException e) {
    e.printStackTrace();
}
```

convenient for embedded OCL constraints

EMF: Querying with OCL (3)

```
import org.eclipse.emf.query.conditions.eobjects.EObjectCondition;
import org.eclipse.emf.query.ocl.conditions.BooleanOCLCondition;
import org.eclipse.emf.query.statements.FROM;
import org.eclipse.emf.query.statements.SELECT;
import org.eclipse.emf.query.statements.WHERE;
import org.eclipse.emf.query.statements.IQueryResult;
```

```
try {
    parametric in classifier, class, and element of the meta-model
    EObjectCondition itemsOK =
        new BooleanOCLCondition<EClassifier, EClass, EObject>(
            purchaseOCL.getEnvironment(),
            "self.quantity < 2", PurchaseJavaPackage.Literals.ITEM);
    IQueryResult result = new SELECT(
        new FROM(resource.getContents()),
        new WHERE(itemsOK)).execute();
    for (Object next : result) {
        System.out.println("Quantity too little in " +
            ((Item) next).getProductName());
    }
} catch (ParserException pe) {
    pe.printStackTrace();
}
```

context

EMF: Querying UML models with OCL

```
import org.eclipse.uml2.uml.UMLFactory;
```

```
UMLFactory umlFactory = UMLFactory.eINSTANCE;  
org.eclipse.uml2.uml.Activity activity = umlFactory.createActivity();  
activity.setName("test");  
OCL umlOCL = OCL.newInstance();  
try {  
    umlOCL.parse(new OCLInput("package uml " +  
                               "context Activity " +  
                               "inv: name <> ' ' " +  
                               "endpackage"));  
    for (Constraint constraint : umlOCL.getConstraints()) {  
        System.out.println(umlOCL.check(activity, constraint));  
    }  
} catch (ParserException e) {  
    e.printStackTrace();  
}
```

Xtext

Xtext

- Grammar language for describing domain-specific languages textually
 - Based on LL(*)-parser generator ANTLR
 - Generation of Eclipse-integrated editor (with validator, content assist, outline, formatting, ...)
- Tightly integrated with EMF
 - Ecore meta-model inference from grammar
- Model querying (and transformation) with Xtend
- Model-to-text transformation with Xpand
- Integration into EMFT's Modeling Workflow Engine (MWE)
 - Dependency injection using Google's Guice
- Originally developed in the openArchitectureWare project (2006)
- Since 2008 integrated in the Textual Modeling Framework (TMF) of EMF
 - Current version (July 2011): Xtext 2.0

<http://www.eclipse.org/Xtext>

DSL example: Secret compartments (1)

“I have vague but persistent childhood memories of watching cheesy adventure films on TV. Often, these films would be set in some old castle and feature secret compartments or passages. In order to find them, heroes would need to pull the candle holder at the top of stairs and tap the wall twice.

Let’s imagine a company that decides to build security systems based on this idea. They come in, set up some kind of wireless network, and install little devices that send four-character messages when interesting things happen. For example, a sensor attached to a drawer would send the message D2OP when the drawer is opened. We also have little control devices that respond to four-character command messages—so a device can unlock a door when it hears the message D1UL.

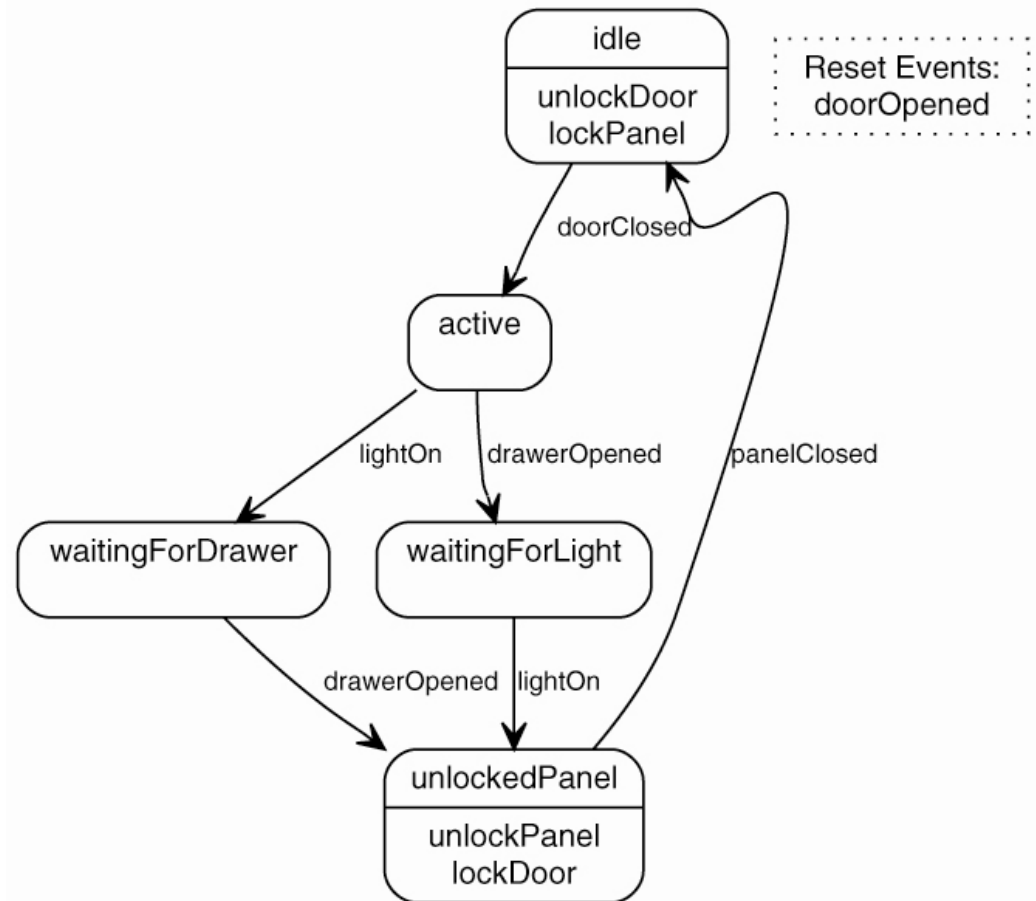
At the center of all this is some controller software that listens to event messages, figures out what to do, and sends command messages. The company bought a job lot of Java-enabled toasters during the dot-com crash and is using them as the controllers. So whenever a customer buys a gothic security system, they come in and fit the building with lots of devices and a toaster with a control program written in Java.

For this example, I’ll focus on this control program. Each customer has individual needs, but once you look at a good sampling, you will soon see common patterns.”

DSL example: Secret compartments (2)

“Miss Grant closes her bedroom door, opens a drawer, and turns on a light to access a secret compartment. Miss Shaw turns on a tap, then opens either of her two compartments by turning on correct light. Miss Smith has a secret compartment inside a locked closet inside her office. She has to close a door, take a picture off the wall, turn her desk light on three times, open the top drawer of her filing cabinet—and then the closet is unlocked. If she forgets to turn the desk light off before she opens the inner compartment, an alarm will sound.”

Martin Fowler. Domain-specific Languages, 2010.




DSL example: Secret compartments (3)


```
events
  doorClosed D1CL  drawOpened D2OP  lightOn L1ON  doorOpened D1OP  panelClosed PNCL end
resetEvents
  doorOpened end
commands
  unlockPanel PNUL  lockPanel PNLK  lockDoor D1LK  unlockDoor D1UL end


state idle
  actions { unlockDoor lockPanel }
  doorClosed => active end
state active
  drawOpened => waitingForLight
  lightOn => waitingForDraw end
state waitingForLight
  lightOn => unlockedPanel end
state waitingForDraw
  drawOpened => unlockedPanel end
state unlockedPanel
  actions { unlockPanel lockDoor }
  panelClosed => idle end
```

Secret compartments in Xtext: Grammar (1)

```
grammar org.eclipse.xtext.example.fowlerdsl.StateMachine
  with org.eclipse.xtext.common.Terminals

generate stateMachine "http://www.eclipse.org/xtext/example/fowlerdsl/StateMachine"
   name and nsURI of EPackage

StateMachine :
  {StateMachine}  action generating an Ecore object
  ('events'
    events+=Event+
    'end')?
  ('resetEvents'
    resetEvents+=[Event]+
    'end')?
  ('commands'
    commands+=Command+
    'end')?
  states+=State*
;

 cross-reference
```

Secret compartments in Xtext: Grammar (2)

```
Event:  
  name=ID code=ID  
;  
  
Command:  
  name=ID code=ID  
;  
  
State:  
  'state' name=ID  
  ('actions' '{' actions+=[Command]+ '}')?  
  transitions+=Transition*  
  'end'  
;  
  
Transition:  
  event=[Event] '=>' state=[State]  
;
```

← identifier token from terminals

Secret compartments: Code generation with Xtend/Xpand (1)

```
package org.eclipse.xtext.example.fowlerdsl.generator

import org.eclipse.emf.ecore.resource.Resource
import org.eclipse.xtext.generator.IGenerator
import org.eclipse.xtext.generator.IFileSystemAccess
import org.eclipse.xtext.example.fowlerdsl.statemachine.Statemachine
import org.eclipse.xtext.example.fowlerdsl.statemachine.Event
import org.eclipse.xtext.example.fowlerdsl.statemachine.Command
import org.eclipse.xtext.example.fowlerdsl.statemachine.State

class StatemachineGenerator implements IGenerator {
    override void doGenerate(Resource resource, IFileSystemAccess fsa) {
        fsa.generateFile(resource.className+".java",
            toJavaCode(resource.contents.head as Statemachine))
    }

    def className(Resource res) {
        var name = res.URI.lastSegment
        return name.substring(0, name.indexOf('.'))
    }
}
```

Secret compartments: Code generation with Xtend/Xpand (2)

```
def toJavaCode(Statemachine sm) '''
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

public class «sm.eResource.className» {
    public static void main(String[] args) {
        new «sm.eResource.className»().run();
    }

    «FOR c : sm.commands»
        «c.declareCommand»
    «ENDFOR»

protected void run() {
    boolean executeActions = true;
    String currentState = "«sm.states.head.name»";
    String lastEvent = null;
    while (true) {
        «FOR state : sm.states»
            «state.generateCode»
        «ENDFOR»
    }
}
```

Secret compartments: Code generation with Xtend/Xpand (3)

```
    «FOR resetEvent : sm.resetEvents»
      if («resetEvent.name».equals(lastEvent)) {
        System.out.println("Resetting state machine.");
        currentState = «sm.states.head.name»;
        executeActions = true;
      }
    «ENDFOR»
  }
}

private String receiveEvent() {
  System.out.flush();
  BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
  try {
    return br.readLine();
  } catch (IOException ioe) {
    System.out.println("Problem reading input");
    return "";
  }
}
}
'''
```


Secret compartments: Code generation with Xtend/Xpand (4)

```
def declareCommand(Command command) '',
    protected void do«command.name.toFirstUpper»() {
        System.out.println("Executing command «command.name» («command.code»");
    }
'''

def generateCode(State state) '',
    if (currentState.equals("«state.name»")) {
        if (executeActions) {
            «FOR c : state.actions» do«c.name.toFirstUpper»(); «ENDFOR»
            executeActions = false;
        }
        System.out.println("Your are now in state '«state.name»'. Possible events are
[«state.transitions.map(t | t.event.name).join(', ')>].");
        lastEvent = receiveEvent();
        «FOR t : state.transitions»
            if ("«t.event.name»".equals(lastEvent)) {
                currentState = "«t.state.name»";
                executeActions = true;
            }
        «ENDFOR»
    }
'''
}
```

Secret compartments: Modelling workflow (1)

```
module org.eclipse.xtext.example.fowlerdsl.GenerateStateMachine

import org.eclipse.emf.mwe.utils.*
import org.eclipse.xtext.generator.*
import org.eclipse.xtext.ui.generator.*

var grammarURI = "classpath:/org/eclipse/xtext/example/fowlerdsl/StateMachine.xtext"
var file.extensions = "statemachine"
var projectName = "org.eclipse.xtext.example.fowlerdsl"
var runtimeProject = "../${projectName}"

Workflow {
  bean = StandaloneSetup {
    scanClassPath = true
    platformUri = "${runtimeProject}/.."
  }
  component = DirectoryCleaner {
    directory = "${runtimeProject}/src-gen"
  }
  component = DirectoryCleaner {
    directory = "${runtimeProject}.ui/src-gen"
  }
}
```

Secret compartments: Modelling workflow (2)

```
component = Generator {
  pathRtProject = runtimeProject
  pathUiProject = "${runtimeProject}.ui"
  pathTestProject = "${runtimeProject}.tests"
  projectNameRt = projectName
  projectNameUi = "${projectName}.ui"
  language = {
    uri = grammarURI
    fileExtensions = file.extensions
    fragment = grammarAccess.GrammarAccessFragment { }
    fragment =.ecore.EcoreGeneratorFragment { }
    fragment = serializer.SerializerFragment { }
    fragment = resourceFactory.ResourceFactoryFragment {
      fileExtensions = file.extensions
    }
    fragment = parser.antlr.XtextAntlrGeneratorFragment { }
    fragment = validation.JavaValidatorFragment {
      composedCheck = "org.eclipse.xtext.validation.ImportUriValidator"
      composedCheck = "org.eclipse.xtext.validation.NamesAreUniqueValidator"
    }
    fragment = scoping.ImportNamespacesScopingFragment { }
    fragment = exporting.QualifiedNamesFragment { }
    fragment = builder.BuilderIntegrationFragment { }
```

Secret compartments: Modelling workflow (3)

```
fragment = generator.GeneratorFragment {
    generateMwe = true
    generateJavaMain = true
}
fragment = formatting.FormatterFragment {}
fragment = labeling.LabelProviderFragment {}
fragment = outline.OutlineTreeProviderFragment {}
fragment = outline.QuickOutlineFragment {}
fragment = quickfix.QuickfixProviderFragment {}
fragment = contentAssist.JavaBasedContentAssistFragment {}
fragment = parserantlr.XtextAntlrUiGeneratorFragment {}
fragment = junit.Junit4Fragment {}
fragment = types.TypesGeneratorFragment {}
fragment = xbase.XbaseGeneratorFragment {}
fragment = templates.CodetemplatesGeneratorFragment {}
fragment = refactoring.RefactorElementNameFragment {}
fragment = compare.CompareFragment {
    fileExtensions = file.extensions
}
}
}
}
```