XML Metadata Interchange (XMI)

- XMI is a **standard** (and a trademark) from the OMG.
- XMI is a **framework** for
  - defining, interchanging, manipulating and integrating XML data and objects.
- Used for **integration**
  - tools, applications, repositories, data warehouses
  - typically used as interchange format for UML tools

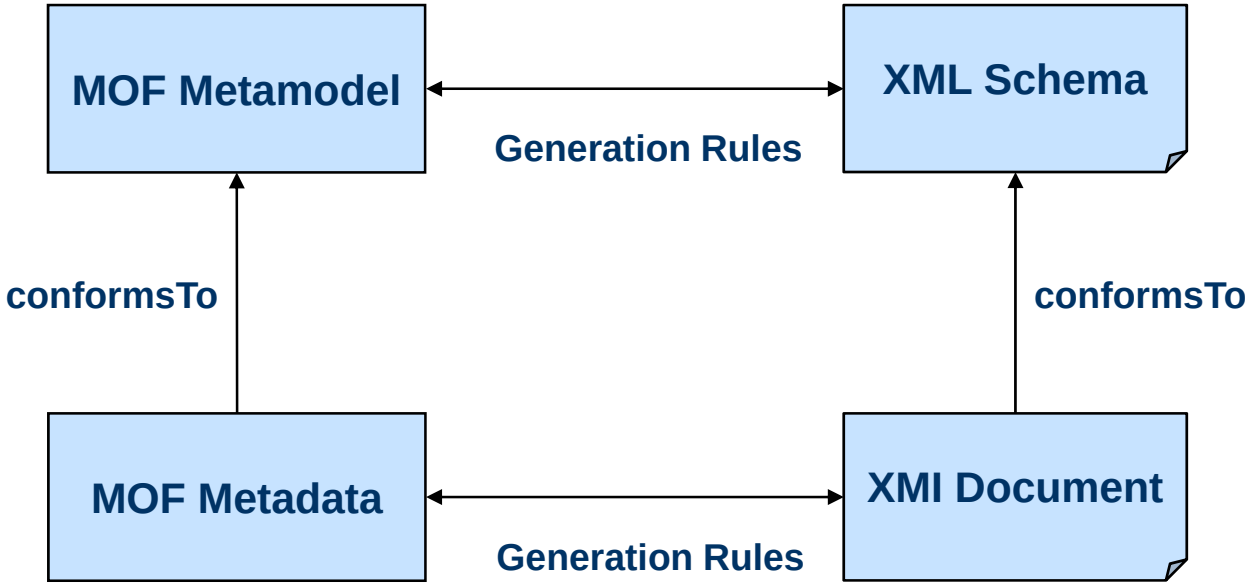
- XMI defines **rules for schema definition**
  - schema production — how is a metamodel mapped onto a grammar?
  - definition of schema from any valid Meta Object Facility (MOF) model
- XMI defines **rules for metadata generation**
  - document production — how is a model mapped onto text?
  - Metadata according to a MOF metamodel is generated into XML according to the generated XML schema.

http://www.omg.org/spec/XMI/2.4.1/

XMI versions and MOF versions

- XMI 1.1 corresponds to MOF 1.3
- XMI 1.2 corresponds to MOF 1.4
- XMI 1.3 (added schema support) corresponds to MOF 1.4
- XMI 2.0 (adds schema support and changes document format) corresponds to MOF 1.4
- **XMI 2.1** corresponds to **MOF 2.0**
- XMI 2.4.1 corresponds to MOF 2.4.1

MOF and XMI


MOF Metamodel
XML Schema
Generation Rules
conformsTo
conformsTo
MOF Metadata
XMI Document
Generation Rules

UML Superstructure as XMI document (1)

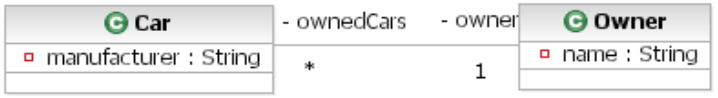
```
<?xml version="1.0" encoding="UTF-8"?>
<cmof:Package xmi:version="2.1"
              xmlns:xmi="http://schema.omg.org/spec/XMI/2.1"
              xmlns:cmof="http://schema.omg.org/spec/MOF/2.0/cmof.xml"
              xmi:id="_0" name="UML">
  <ownedMember xmi:type="cmof:Package" xmi:id="Actions" 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"/>
```

UML Superstructure as XMI 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`
  - ...

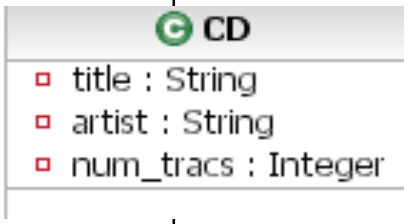
Schema production rules: Classes and properties

- 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>
```



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

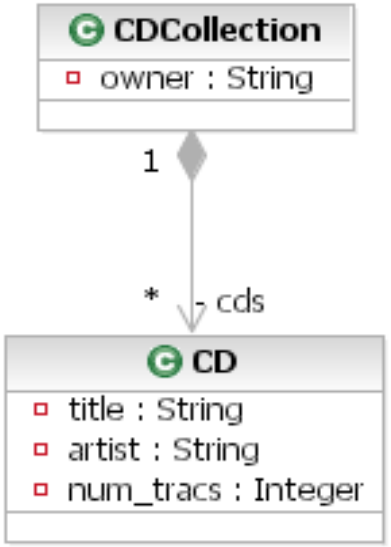
Schema production rules: Relationships

- **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)

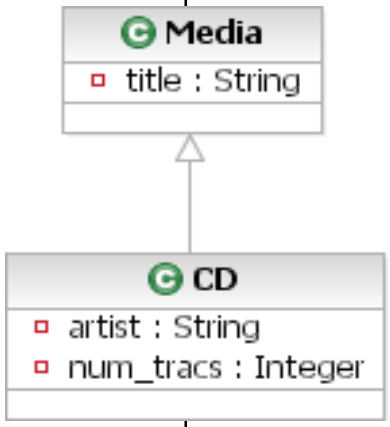
```
<?xml version="1.0" encoding="UTF-8"?>
<xmi:XMI xmi:version='2.1'
  xmlns:xmi="http://schema.omg.org/spec/XMI/2.1"
  xmlns:cdlib="http://www.example.org/CDLib"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.example.org/CDLib">

  <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)

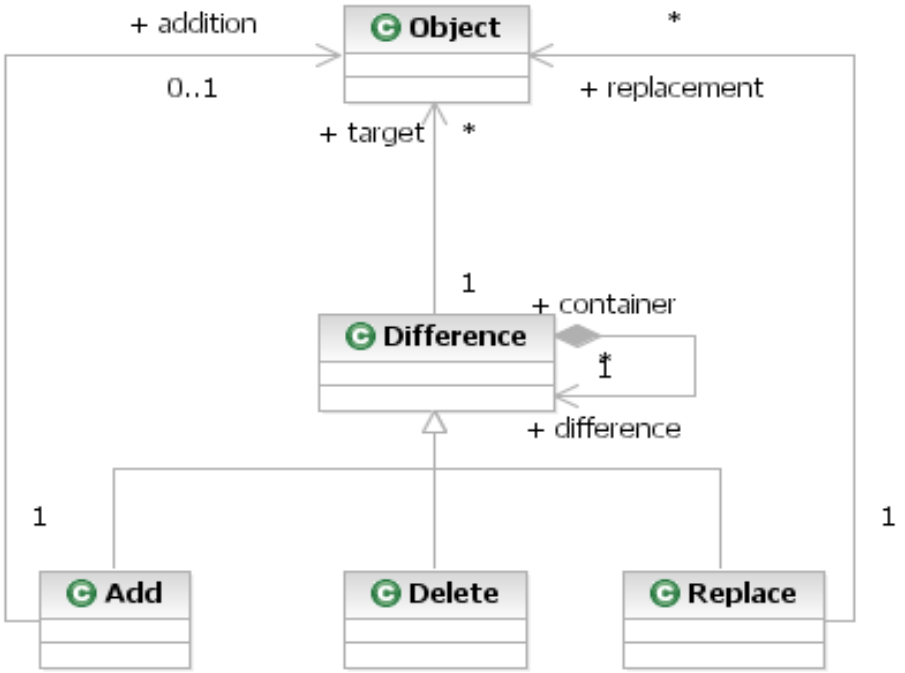
```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns:xmi="http://schema.omg.org/spec/XMI/2.1"
            targetNamespace="http://www.example.org/MediaLib"
            xmlns:medlib="http://www.example.org/MedLib">
  <xsd:import …/>
  <xsd:complexType 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

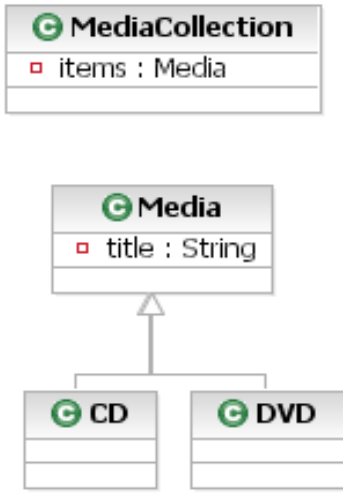
- **Types of differences**
  - Add
  - Delete
  - Replace



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