Introduction to XML Metadata Interchange (XMI)

Department for Cooperative and Trusted Systems
Information and Communication Technology,
SINTEF, Forskningsveien 1, N-0314 Oslo, Norway
http://www.sintef.no

Context of this work

- The present courseware has been elaborated in the context of the MODELWARE European IST FP6 project (http://www.modelware-ist.org/).
- Co-funded by the European Commission, the MODELWARE project involves 19 partners from 8 European countries. MODELWARE aims to improve software productivity by capitalizing on techniques known as Model-Driven Development (MDD).
- To achieve the goal of large-scale adoption of these MDD techniques, MODELWARE promotes the idea of a collaborative development of courseware dedicated to this domain.
- The MDD courseware provided here with the status of open source software is produced under the EPL 1.0 license.

XMI from 30 000 feet

- XMI is short for XML Metadata Interchange
- XMI is a standard (and a trademark) from the Object Management Group (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
  - Definition of schema from any valid Meta Object Facility (MOF) model
  - Schema production
- XMI defines rules for metadata generation
  - Metadata according to a MOF metamodel is generated into XML according to the generated XML schema
  - Document production

Metamodels

- Metamodel: Model of a set of models
- Metamodels are specifications
  - models are valid if no false statements according to metamodel (i.e., well-formed)
- Metametamodel
  - model of metamodels
  - reflexive metamodel, i.e., expressed using itself
    - ref. Kurt Gödel
  - minimal reflexive metamodel
    - can be used to express any statement about a model

Meta levels in OMG

- M0 – what is to be modelled (Das Ding an sich)
- M1 – Models (Das Ding für mich)
  - May contain both class/type and instance models
- M2 – Metamodels
  - UML2.0 and CWM (Common Warehouse Metamodel)
- M3 – The metametamodel
  - MOF

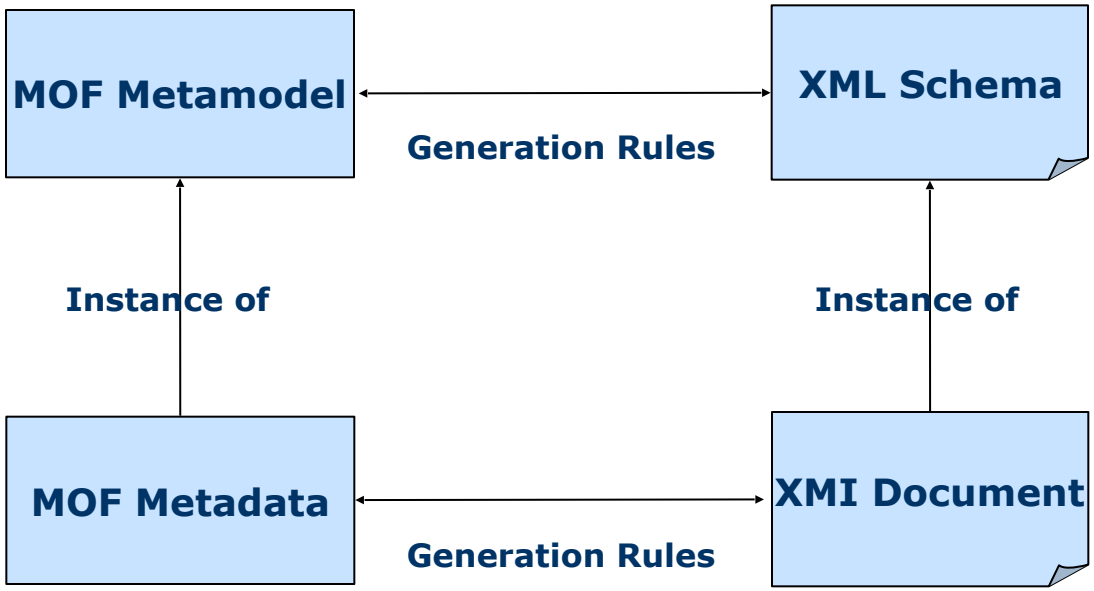
- Interpretation (not instantiation!) crosses meta-layers, theories reside in one layer (e.g., instance models can be deduced from class models)

- The Meta Object Facility (MOF)
  - Facility to handle models and instances of models at multiple levels
  - Contains self describing metametamodel (called MOF)

History - 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

MOF and XMI


MOF Metamodel
Generation Rules
XML Schema
Instance of
Instance of
MOF Metadata
Generation Rules
XMI Document

Schema production rules I

- A set of rules are defined by the standard
  - Some of them are quite complex
  - Typically intended to be implemented, not for human usage
    - EBNF (Extended Backus-Naur form) rules are supplied
- Metamodel Class
  - Mapped to xsd:element and xsd:complexType with same name as Metamodel Class
- Attribute of Class
  - Mapped to xsd:element and xsd:attribute if simple data type and the cardinality of the attribute is [1..1] or [0..1]
  - Mapped to element if xsd:complexType
    - Note: only possible in CMOF (Complete MOF)

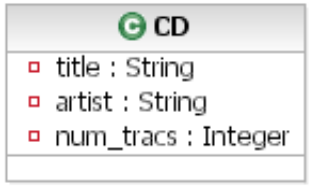
Schema Production Example

```
<xsd:schema xmlns:xsd="http://www.w3c.org/2001/XMLSchema"
        xmlns:xmi="http://www.omg.org/XMI"
        targetNamespace="http://www.sintef.org/CDs"
        xmlns:cds="http://www.sintef.org/CDs">

  <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: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

- According to schema generated in previous slide

Born to Run
Bruce Springsteen
8 tracks

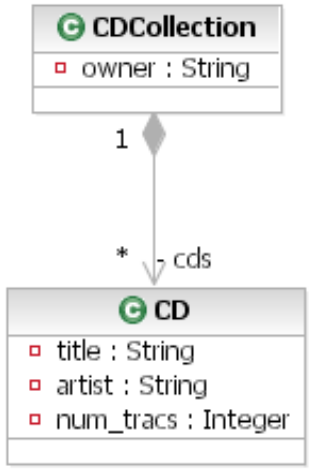
```
<?xml version="1.0" encoding="UTF-8"?>
<cds:CD xmlns:cds="http://www.sintef.org/CDs"
        xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
        xmlns:xmi="http://www.omg.org/XMI"
        xsi:schemaLocation="http://www.sintef.org/CDs"

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

Schema Production Rules II

- 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
    - enforceMinimumMultiplicity
    - enforceMaximumMultiplicity
- Inheritance
  - Challenge
    - 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

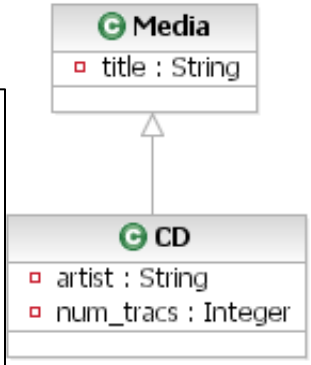
Schema with relationship




```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:cdlib="www.sintef.org"
            xmlns:xmi="http://www.omg.org/XMI"
            xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            targetNamespace="www.sintef.org">
 <xsd:import namespace=http://www.omg.org/XMI schemaLocation="XMI.xsd" />
 <xsd:complexType name="CD">
  <xsd:choice maxOccurs="unbounded" minOccurs="0">
   <xsd:element ref="xmi:Extension" />
  </xsd:choice>
  <xsd:attribute ref="xmi:id" />
  <xsd:attributeGroup ref="xmi:ObjectAttribs" />
  <xsd:attribute name="title" type="xsd:string" use="required" />
  <xsd:attribute name="artist" type="xsd:string" />
  <xsd:attribute name="num_tracs" type="xsd:int" />
 </xsd:complexType>
 <xsd:element name="CD" type="cdlib:CD" />
 <xsd:complexType name="CDLibrary">
  <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="CDLibrary" type="cdlib:CDLibrary" />
</xsd:schema>
```

Schema with inheritance

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xmi=http://www.omg.org/XMI
     xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="ml">
  <xsd:import namespace="http://www.omg.org/XMI"
     schemaLocation="XMI.xsd"/>
  <xsd:complexType name="Media">
    <xsd:choice maxOccurs="unbounded" minOccurs="0">
      <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:complexType>
  <xsd:element name="Media" type="ml:Media"/>
  <xsd:complexType name="CD">
    <xsd:attribute name="title" type="xsd:string"/>
    <xsd:attribute name="artist" type="xsd:string"/>
    <xsd:attribute name="num_tracs" type="xsd:int"/>
  </xsd:complexType>
  <xsd:element name="CD" type="ml:CD"/>
</xsd:schema>
```

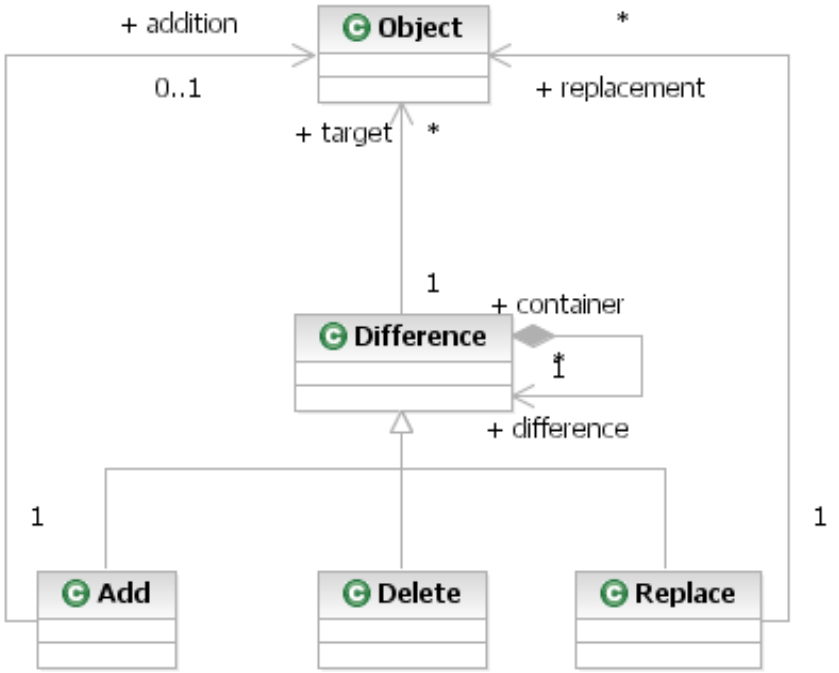


Document with relationship

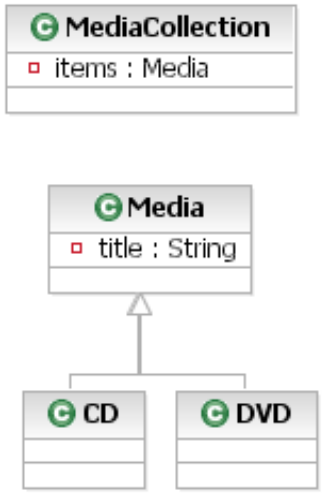

```
<?xml version="1.0" encoding="UTF-8"?>
<cdlib:CDLibrary xmlns:cdlib="www.sintef.org"
                 xmlns:xmi="http://www.omg.org/XMI"
                 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                 xsi:schemaLocation="www.sintef.org CDLibraryXMI.xsd http://www.omg.org/XMI XMI.xsd "
                 owner="Tor Neple" xmi:id="_1">
  <cds title="Born To Run" xmi:id="_2" artist="Bruce Springsteen" num_tracs="8"></cds>
  <cds title="Achtung Baby" xmi:id="_3" artist="U2" num_tracs="13"></cds>
</cdlib:CDLibrary>
```

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



Simple example of differences



collection.xmi

```
<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="DV"/>
  </MediaCollection>
</xmi:XMI>
```

differences

```
<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>
```

result

```
<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>
```

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

XMI in Eclipse

- Eclipse uses the Eclipse Modeling Framework (EMF) as its Meta Object Facility
- ECORE (EMF metamodel) is similar to EMOF
- The EMF project contains tools to
  - Create ECORE metamodels
  - Create XMI schemas for ECORE models
  - Create models according to the defined metamodels
  - Reading and writing such models in XMI format
- Ecore XMI != OMG XMI
  - Some small differences since there are some differences between ECORE and EMOF (Essential MOF)

SIMPLE UML model in XMI format


```
<?xml version="1.0" encoding="UTF-8"?>
<uml:Model xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI"
  xmlns:uml="http://www.eclipse.org/uml2/1.0.0/UML" xmi:id="_UwyEFBV_Edqs_vsvaW3hqA" name="CarOwner" >
  <ownedMember xmi:type="uml:Class" xmi:id="_UwyEHxV_Edqs_vsvaW3hqA" name="Car">
    <ownedAttribute xmi:id="_UwyEIBV_Edqs_vsvaW3hqA" name="owner" visibility="private"
          type="_UwyEIhV_Edqs_vsvaW3hqA" association="_UwyEJxV_Edqs_vsvaW3hqA"/>
    <ownedAttribute xmi:id="_UwyEIRV_Edqs_vsvaW3hqA" name="manufacturer" visibility="private">
      <type xmi:type="uml:PrimitiveType"
          href="pathmap://UML2_LIBRARIES/UML2PrimitiveTypes.library.uml2#_IXlH8a86EdieaYgxtVWN8Q"/>
    </ownedAttribute>
  </ownedMember>
  <ownedMember xmi:type="uml:Class" xmi:id="_UwyEIhV_Edqs_vsvaW3hqA" name="Owner">
    <ownedAttribute xmi:id="_UwyEIxV_Edqs_vsvaW3hqA" name="ownedCars"
          visibility="private" type="_UwyEHxV_Edqs_vsvaW3hqA" association="_UwyEJxV_Edqs_vsvaW3hqA">
      <upperValue xmi:type="uml:LiteralUnlimitedNatural" xmi:id="_UwyEJBV_Edqs_vsvaW3hqA" value="-1"/>
      <lowerValue xmi:type="uml:LiteralInteger" xmi:id="_UwyEJRV_Edqs_vsvaW3hqA"/>
    </ownedAttribute>
    <ownedAttribute xmi:id="_UwyEJhV_Edqs_vsvaW3hqA" name="name" visibility="private">
      <type xmi:type="uml:PrimitiveType"
          href="pathmap://UML2_LIBRARIES/UML2PrimitiveTypes.library.uml2#_IXlH8a86EdieaYgxtVWN8Q"/>
    </ownedAttribute>
  </ownedMember>
  <ownedMember xmi:type="uml:Association"
          xmi:id="_UwyEJxV_Edqs_vsvaW3hqA" memberEnd="_UwyEIBV_Edqs_vsvaW3hqA _UwyEIxV_Edqs_vsvaW3hqA"/>
</uml:Model>
```

Simplified UML2.0 xmi file exported from Rational Software Architect
EMF based, not MOF XMI compliant

Summary and Conclusion

- XMI allows for defining, interchanging, manipulating and integrating XML data and objects
  - Defines rules for creation of schema from metamodel and document from model
- Works on MOF based models
- Widely used for model interchange between UML tools
  - Great possibilites for version conflicts