DAIMLERCHRYSLER

Practical Usage of W3C's XML-Schema and a Process for Generating Schema Structures from UML Models

Mario C. Jeckle
DaimlerChrysler Research and Technology, Ulm, Germany
mario.jeckle@daimlerchrysler.com
mario@jeckle.de
www.jeckle.de

Overview

I Background and Problem Statement

- Practical usage of W3C's XML Schema Language (XSD)
- XSD as part of object oriented development processes

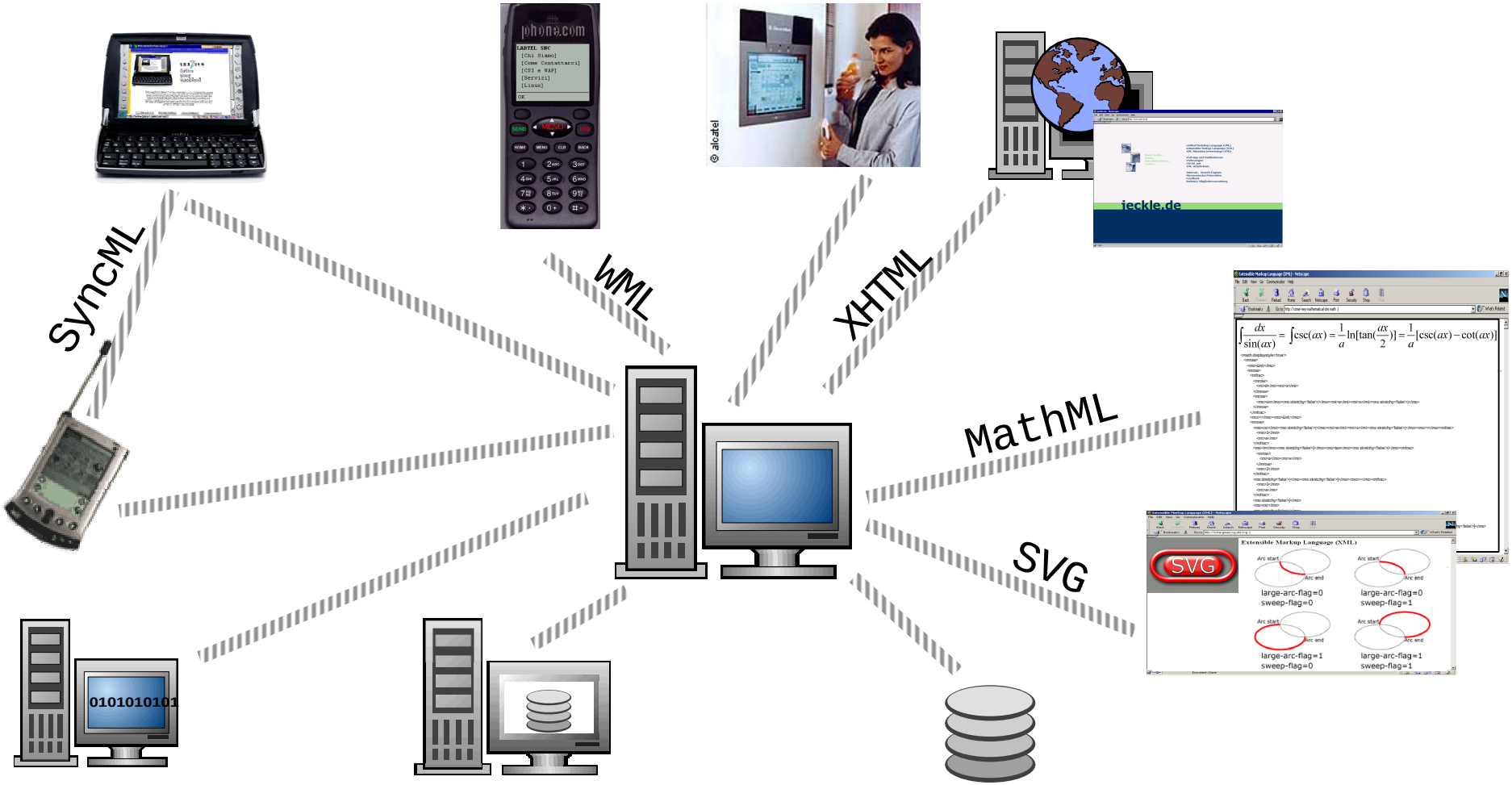
II Technical Aspects

- Unified Modeling Language (UML)
- XML Metadata Interchange (XMI)

III Creating XML Schema Structures from UML Models

- Datatypes
- Structures

Proliferation of XML



XML Schema

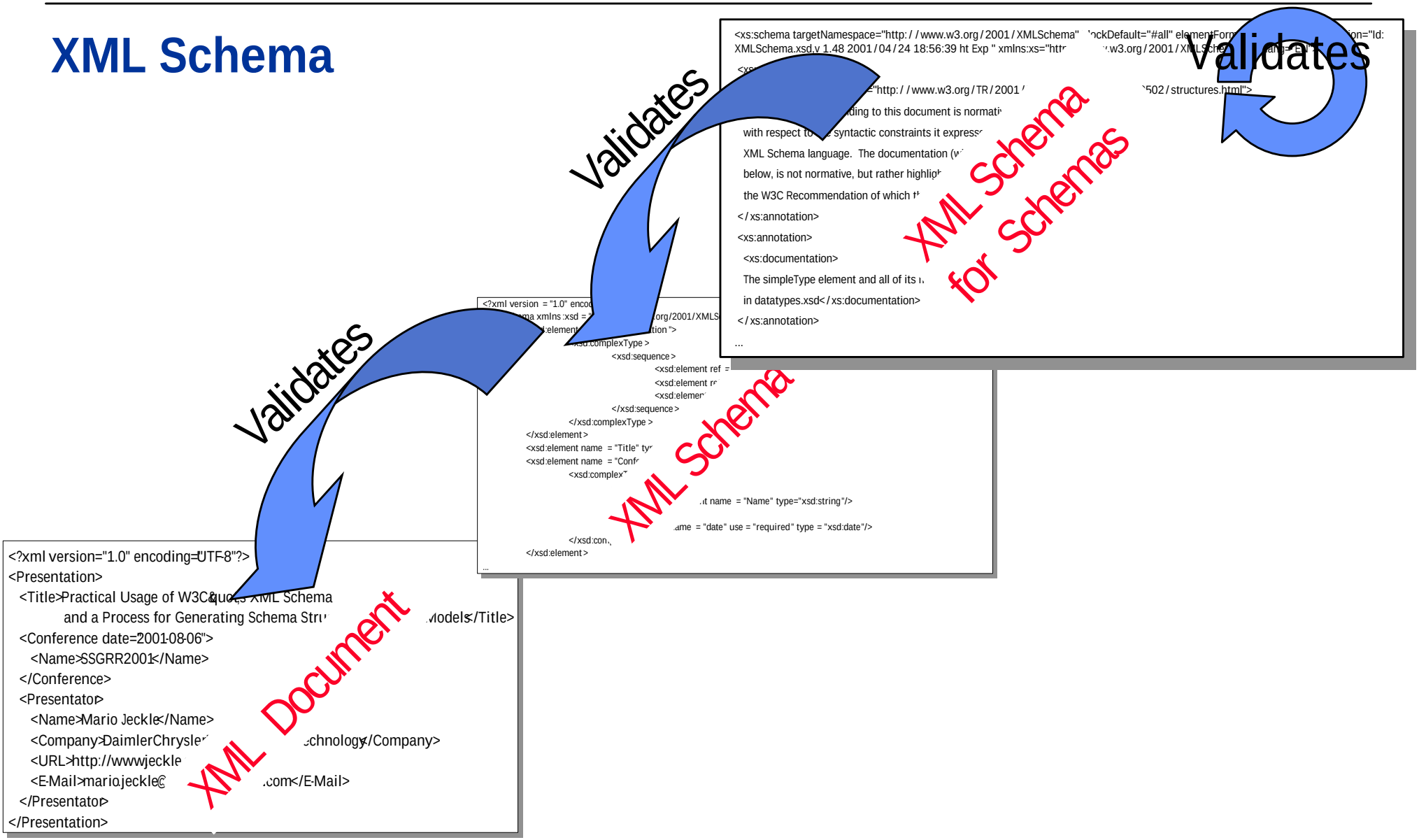
```
<?xml version="1.0" encoding="UTF-8"?>
<Presentation>
  <Title>Practical Usage of W3C&quot;s XML Schema
          and a Process for Generating Schema Structures from UML Models</Title>
  <Conference date="2001-08-06">
    <Name>SSGRR2001</Name>
  </Conference>
  <Presentator>
    <Name>Mario Jeckle</Name>
    <Company>DaimlerChrysler Research and Technology</Company>
    <URL>http://www.jeckle.de</URL>
    <E-Mail>mario.jeckle@daimlerchrysler.com</E-Mail>
  </Presentator>
</Presentation>
```

XML Schema

```
<?xml version = "1.0" encoding = "UTF-8"?>
<xsd:schema xmlns:xsd = "http://www.w3.org/2001/XMLSchema" elementFormDefault = "qualified">
        <xsd:element name = "Presentation">
                <xsd:complexType>
                        <xsd:sequence>
                                <xsd:element ref = "Title"/>
                                <xsd:element ref = "Conference"/>
                                <xsd:element ref = "Presentator"/>
                        </xsd:sequence>
                </xsd:complexType>
        </xsd:element>
        <xsd:element name = "Title" type="xsd:string"/>
        <xsd:element name = "Conference">
                <xsd:complexType>
                        <xsd:sequence>
                                <xsd:element name = "Name" type="xsd:string"/>
                        </xsd:sequence>
                        <xsd:attribute name = "date" use = "required" type = "xsd:date"/>
                </xsd:complexType>
        </xsd:element>
...
```

```
<?xml version="1.0" encoding="UTF-8"?>
<Presentation>
  <Title>Practical Usage of W3C&quot;s XML Schema
        and a Process for Generating Schema Structures from UML Models</Title>
  <Conference date="2001-08-06">
    <Name>SSGRR2001</Name>
  </Conference>
  <Presentator>
    <Name>Mario Jeckle</Name>
    <Company>DaimlerChrysler Research and Technology</Company>
    <URL>http://www.jeckle.de</URL>
    <E-Mail>mario.jeckle@daimlerchrysler.com</E-Mail>
  </Presentator>
</Presentation>
```

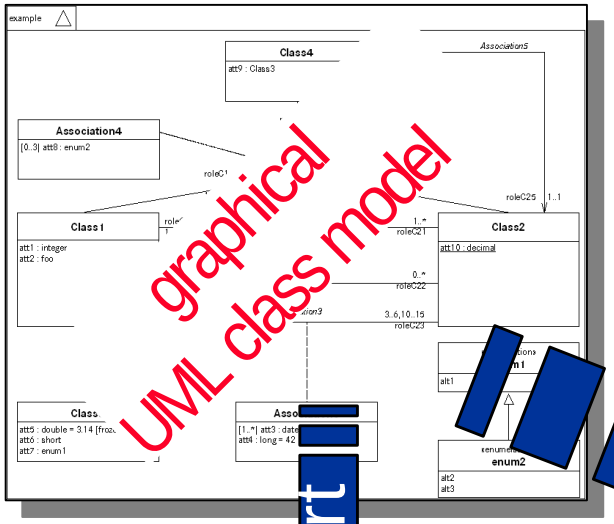
XML Schema


Validates
Validates
Validates
XML Schema for Schemas
XML Schema
XML Document
<?xml version="1.0" encoding="UTF-8"?>
<Presentation>
<Title>Practical Usage of W3C"s XML Schema
and a Process for Generating Schema Stru
Models</Title>
<Conference date="2001-08-06">
<Name>SSGRR2001</Name>
</Conference>
<Presentator>
<Name>Mario Jeckle</Name>
<Company>DaimlerChrysle
echnology</Company>
<URL>http://www.jeckle
<E-Mail>mario.jeckle@
.com</E-Mail>
</Presentator>
</Presentation>
<xs:schema targetNamespace="http://www.w3.org/2001/XMLSchema"
XMLSchema.xsd,v 1.48 2001/04/24 18:56:39 ht Exp " xmlns:xs="htt
<xs:annotation>
<xs:documentation>
The simpleType element and all of its
in datatypes.xsd</xs:documentation>
</xs:annotation>
...

XML Schema

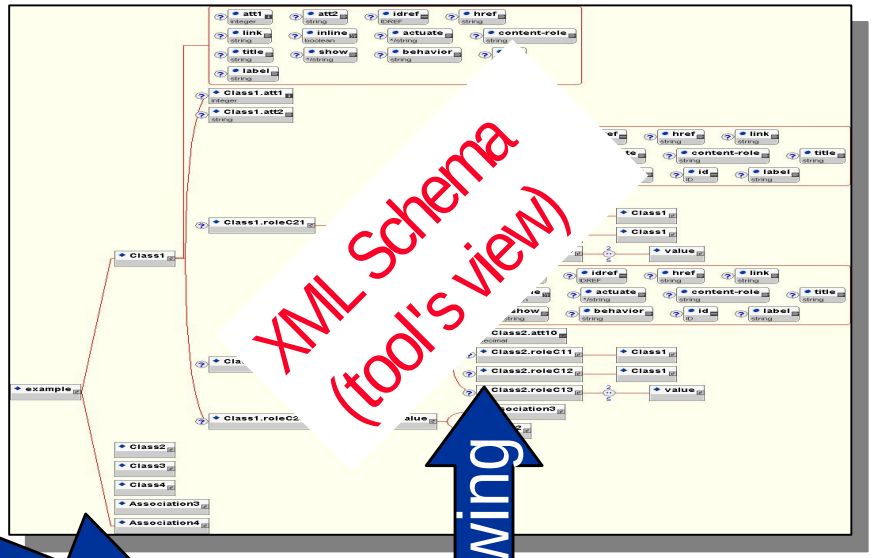
- Grammar for arbitrary XML vocabularies
- Part 1 describes structures and content of XML elements and attributes
- Part 2 describes datatypes used within part 1 and elsewhere
- Significantly improves DTD's expressive power (in the long term it will supersede it)
- XML schema is by itself a XML vocabulary
- Combines all of the major competing successors
- W3C recommendation since 2001-05-02
- Forms the basis of all W3C's second generation standards (e. g. XPath v2.0, XSLT v2.0, XHTML v2.0, SOAP, ...)
- Tool support available yet
- Only the first step towards data orientation as basis of the semantic web ...

Problem Statement: Seamless Schema Generation



export

derivative mapping



viewing


```
<?xml version="1.0" encoding="UTF-
<XMI xmi.version="1.0">
  <XMI.header>
    <XMI.documentation>
      <XMI.exporter>DaimlerChrysler Re                    .</XMI.exporter>
      <XMI.exporterVersion>1.0</XM
    </XMI.documentation>
    <XMI.metamodel xmi.nar
  </XMI.header>
  <XMI.content>
    <Model_Managen.                    ..example">

<Foundation.Core.ModelEl                    example</Foundation.Core.ModelElement.name>
    <Foundation.Core.ModelEl                ..visibility xmi.value="public"/>
...
```


XML / XMI representation

transformation


```
                encoding    = " UTF -8"?>
                = "http:/   www  .w3. org /2001/  XMLSchema                    ormDefault      = " qualified  ">
                    = " Presentation    ">
                    lexType    >
                    sd :sequence   >
                                <xsd :element ref    = " Title  "/>
                                <xsd :element ref    = " Conference    "/>
                                <xsd :element ref    = " Presentator    "/>
                            </ xsd :sequence   >
                        </ xsd :complexType    >
                </ xsd :element   >
                <xsd :element name     = " Title  " type =" xsd :string  "/>
                    element name     = " Conference    ">
                    complexType    >
                            <xsd :sequence   >
                                <xsd :elemen                    =" xsd :string  "/>
                            </ xsd :sequence   >
                            <xsd :attribute name     = " date                    equired   " type   = " xsd :date  "/>
                        </ xsd :complexType    >
                </ xsd :element   >
...
```


XML Schema

Quality Requirements

- **Flexibility**
  Future proof (i.e. easy to adapt) languages
- **Speed**
  Amount of XML-Languages needed for today's applications (e.g. B2B, A2A, ...)
- **Coherence**
  Need to keep application's data structures and XML format in sync
- **Accuracy**
  XML structures should reflect business structures and rules
- **Style**
  Languages for similar applications domains should offer the same *look and feel*
- **Integration**
  Current development processes do not take care of serialization formats
- **Reuse**
  Of existing design meta-knowledge

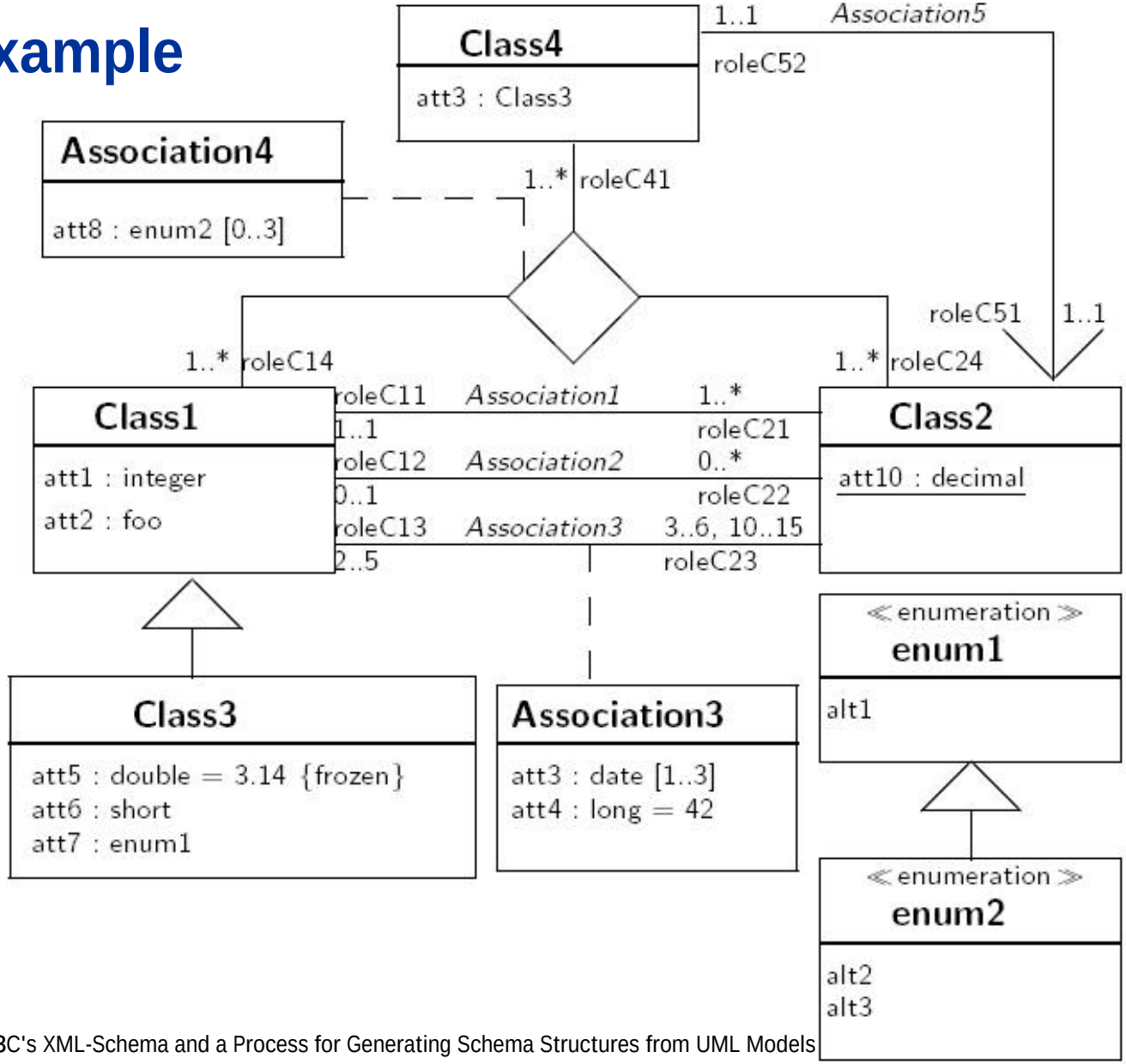
Unified Modeling Language (UML)

- Object oriented modeling language for general purpose systems
- Includes most of precursor graphical modeling languages (e.g. Booch, OMT, Use Cases, ...)
- Various diagram types supporting different development phases
  (e.g. *use case, class, sequence, activity, collaboration diagram, ...)*
- Defines no development process, i.e. precludes none
- Widely adopted by tool vendors (e.g. see www.jeckle.de/umltools.html)
- Standardized under the auspice of the OMG
- Defines no textual representation
  Subject to a separate RFP entitled *Stream-based Model Interchange Format*

Stream-based Model Interchange: XML Metadata Interchange (XMI)

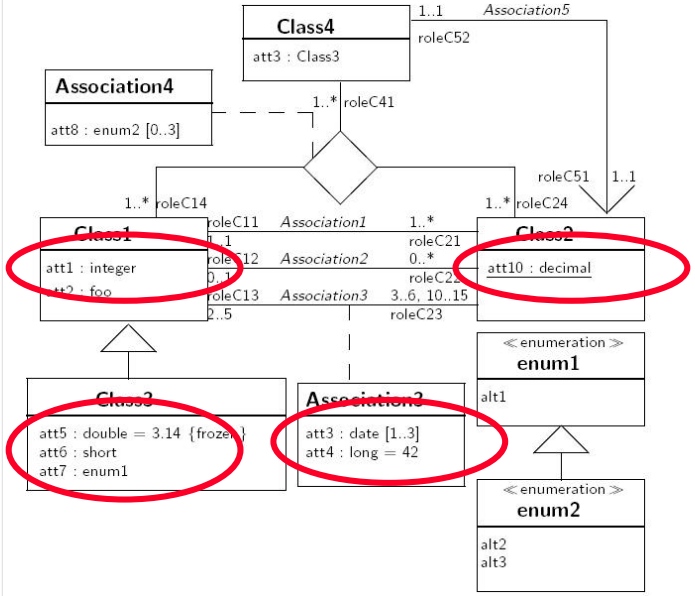
- **#1 goal**: Model Interchange a.k.a. Metadata Interchange among heterogeneous object oriented modeling tools
- **Purpose**: Encoding of MOF-based metamodels i.e. UML models and UML-based metamodels
- **Developed by**: Unisys, IBM, DSTC, Oracle, Platinum, Fujitsu, Softeam, Recerca Informatica, DaimlerChrysler
- **Supported by**: Genesis, Inline, Rational, Select, Sprint, Cayenne, Sybase, Xerox, Boeing, Ardent, MCI Systemhouse, Aviatis, ICONIX, Integrated Systems, Verilog, Nihon Unisys, NTT, Telefonica I+D, NCR, Universitat Politecnica de Catalunya

A Guiding Example



Generation of Schema Structures: Type Interference

- Interfering the best matching type (under a semantic's point of view)
  Mapping directly to datatypes predefined by UML / MOF / CORBA



Integer → integer

String → string

Name → Name

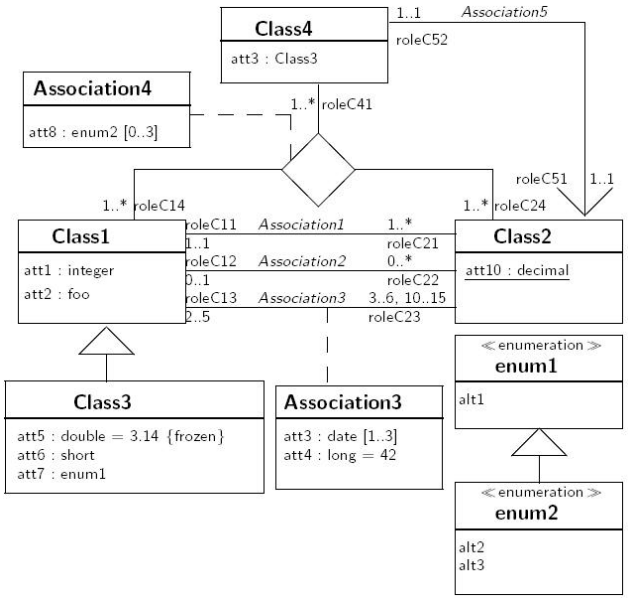
Boolean → boolean

Time → ???

Uninterpreted → { base64Binary, hexBinary }

Generation of Schema Structures: Type Matching

- Interfering the best matching type (under a semantic's point of view)
  (pre-)Creating new standard types

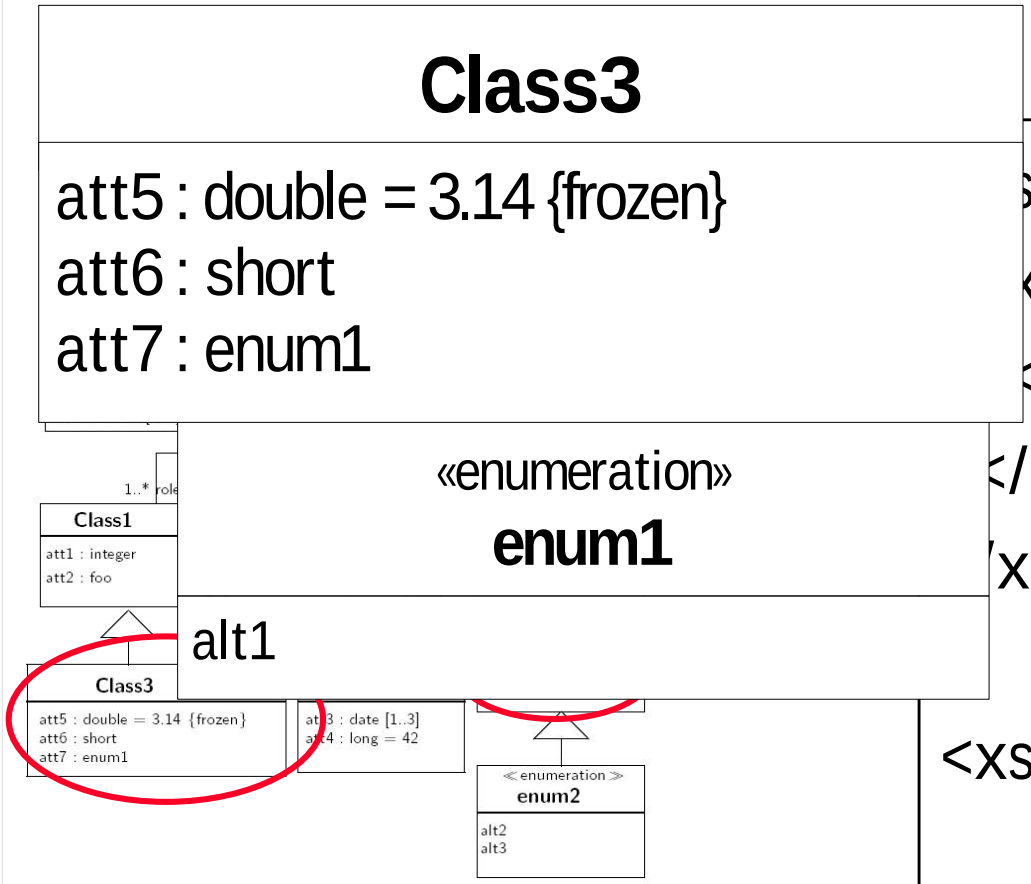


Time ⟶ <xsd:simpleType name="UDtime">

```
<xsd:union
  memberTypes="
   xsd:dateTime
   xsd:time
   xsd:date
   xsd:gMonth
   xsd:gYear"/>
</xsd:simpleType>
```

Generation of Schema Structures: User Defined Types

- Interfering the best matching type (under a semantic's point of view)
  Transferring user's extensions to the UML metamodel into new XSD types

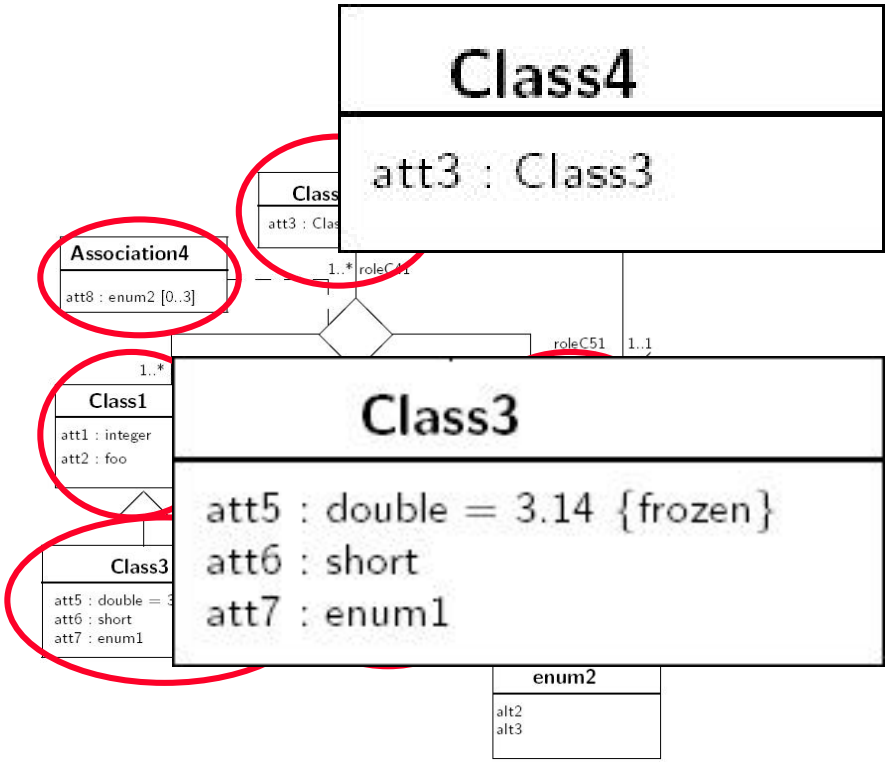



```
sd:simpleType name="enum1Type">
xsd:restriction base="xsd:string">
<xsd:enumeration value="alt1"/>
</xsd:restriction>
/xsd:simpleType>

<xsd:element name="att7"
                type="enum1Type"/>
```

Generation of Schema Structures: Classes and Attributes

- UML classes are transferred into XSD elements allowing re-use and referenciation
- UML attributes are transferred also into XSD elements, since XSD attributes differ significantly (the identical named concepts are just a misnomer)

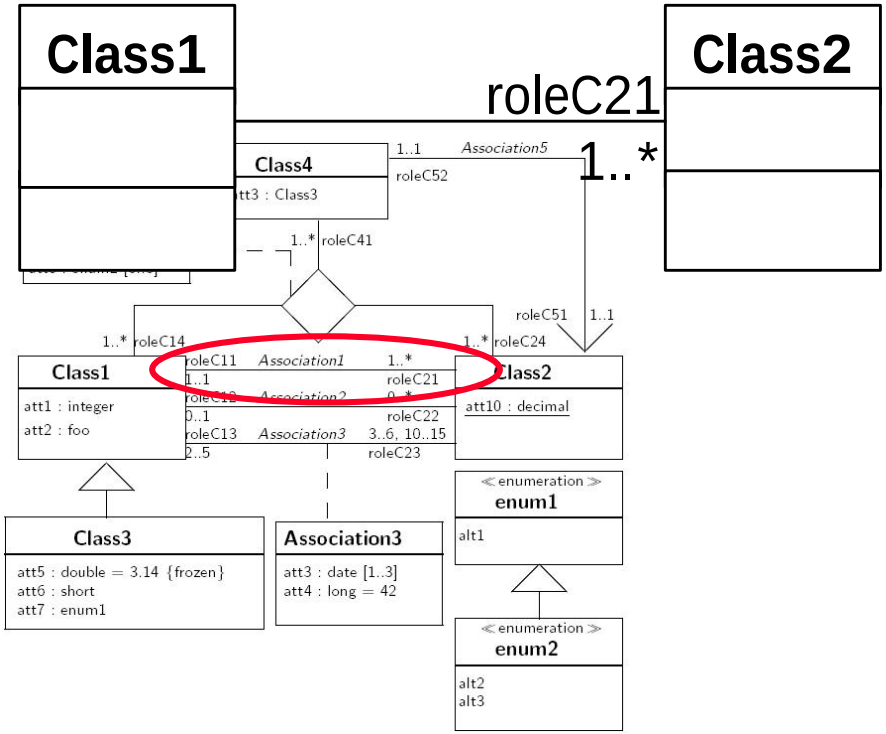



```
<xsd:element name="Class3">
<xsd:element name="att5"
    type="xsd:double">
<xsd:element name="att6"
    type="xsd:short">

<xsd:element name="Class4">
    <xsd:element name="att3"
    ref="Class3">
```

Generation of Schema Structures: Associations

- Purpose: Transformation from the net to structures of trees
- Basic Idea: Modification of all minimal multiplicities to zero

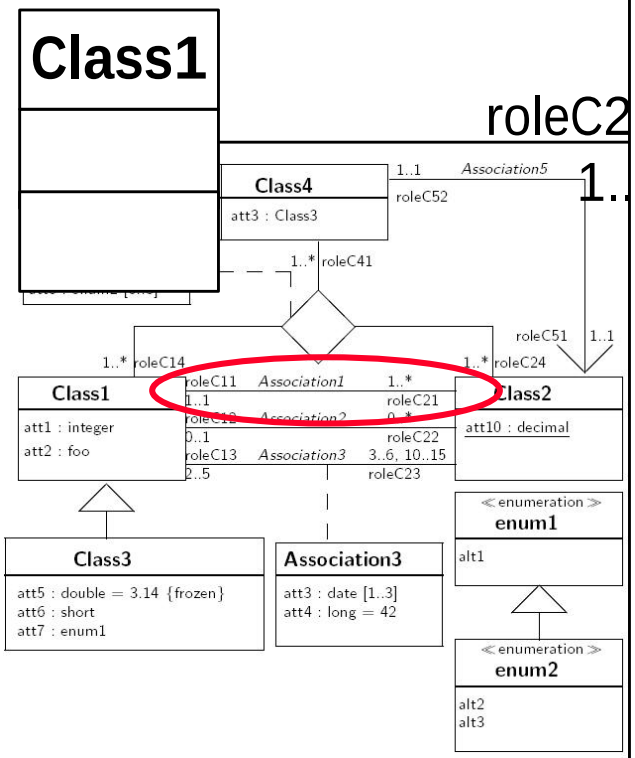



```
<xsd:element name="Class1">
<xsd:element name="Class1.roleC21"
    ref="Class2" minOccurs="0"
    maxOccurs="unbounded">

<xsd:element name="Class1.roleC21"
    type="Class2">
```

Generation of Schema Structures: Avoiding Redundancy

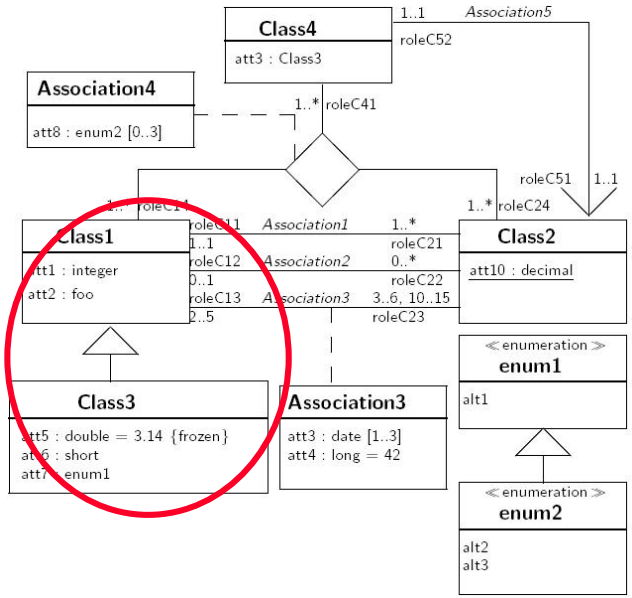
- Purpose: Transformation from the net to structures of trees
- Usage of XML standard XLink to facilitate flexible referenciation




```
<xsd:element name="Class1">
  <xsd:all>
    <xsd:element name="att1"
     minOccurs="0"/>
    ...
  </xsd:all>
  <xsd:anyAttribute
  namespace="http://www.w3.org/1999/xlink"/>
..
</xsd:element>
```

Generation of Schema Structures: Inheritance

- Two faces of inheritance: copy-down semantics for all characteristics, and substitutability according to Liskov's principle
- XSD's inheritance mechanism is not semantically equivalent to the one offered by UML




```
<xsd:element name="Class2">
  <xsd:element ref="Class2.role21"
     minOccurs="0" maxOccurs="1"/>
    ...
<xsd:element name="Class2.role21"
   <xsd:choice>
      <xsd:element ref="Class1"/>
      <xsd:element ref="Class3"/>
   </xsd:choice>
...
```

Summary

- Derivation algorithm is stable and could be used widely
- Flexible, since new vocabularies could be easily re-generated
- Fast, since automatic derivation is deployed instead of a error-prone non-repeatable craftsmen's approach
- Coherent, built-in synchronization between data model and XML vocabulary
- Style, UML model (with it's modeling style) determines *style* of XML vocabulary. Meta-Structure is identical due to consistent derivation rules.
- If UML model reflects business structures and rules, derived XML schema also does
- Integrated, plug able into every object oriented development process
- Reuses existing data modeling knowledge (e.g. design patterns)
- Portable since XSLT implementation precludes neither programming language nor operating system