DAIMLERCHRYSLER

Using XSLT to derive schemata from UML

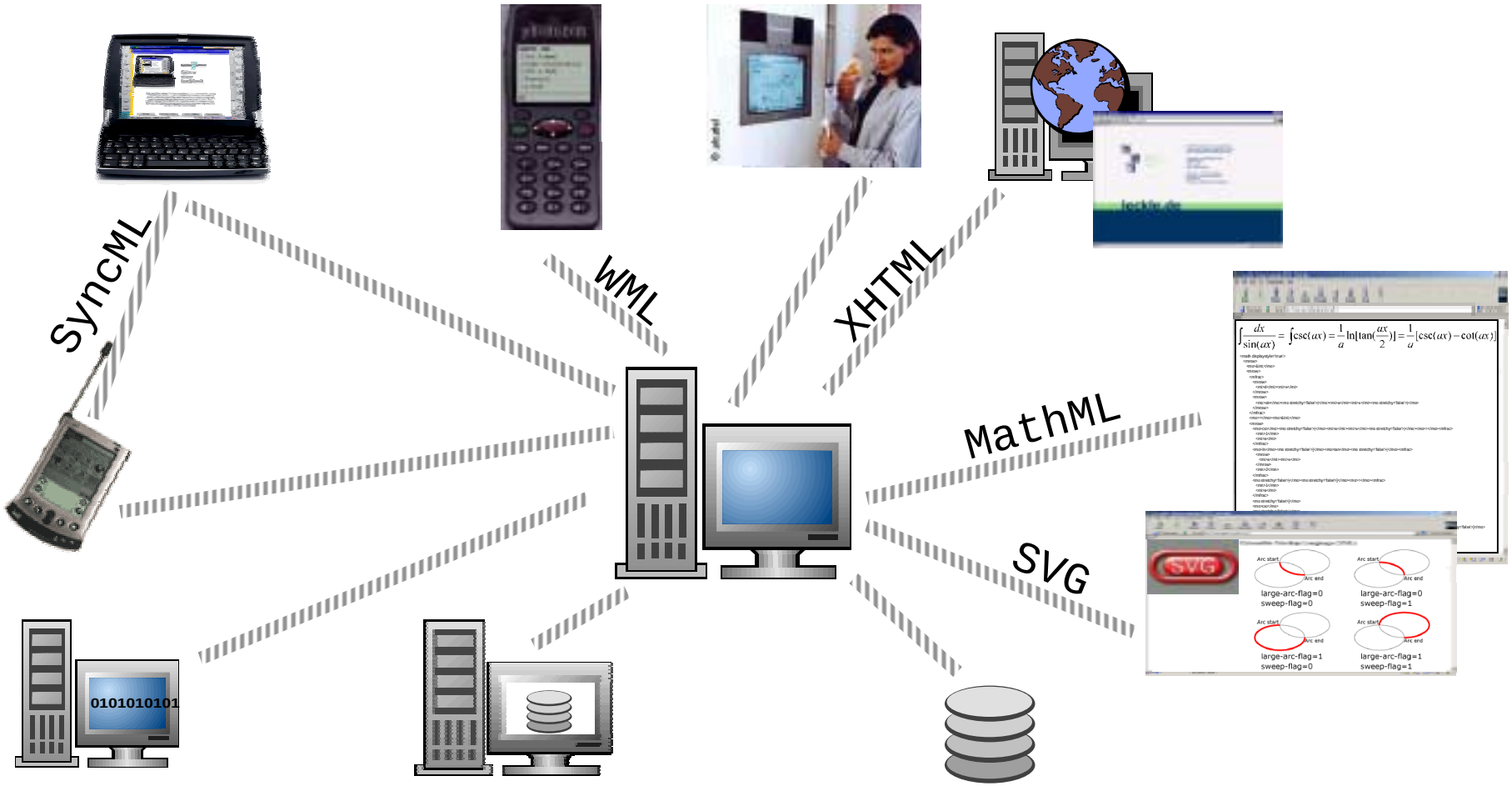
Mario Jeckle
DaimlerChrysler Research Center Ulm (Germany)
mario.jeckle@daimlerchrysler.com
mario@jeckle.de
www.jeckle.de

Overview

- Problem Statement
- The Big Picture:

  Seamless Derivation of XML-Schemata from UML Models
- UML a short introduction
- OMG's XML Metadata Interchange (XMI)
- W3C's XML Schema
- A Schema Derivation Algorithm based on XMI and UML
- Implementing Schema Derivation using XSLT

XML everywhere



- Access to Data, Systems and Services is getting *XML-ized*
- XML will become the ASCII of the 21st century

Problem Statement: Creation of XML Vocabularies

- **Flexibility**

  Future proof (i.e. easy to adapt) languages
- **Speed**

  Amount of XML-Languages needed for today's applications (e.g. B2B, ...)
- **Coherence**

  Need to keep application's data structures and XML format in sync
- **Accuracy**

  XML structures should reflect business structures and rules
- **Style**

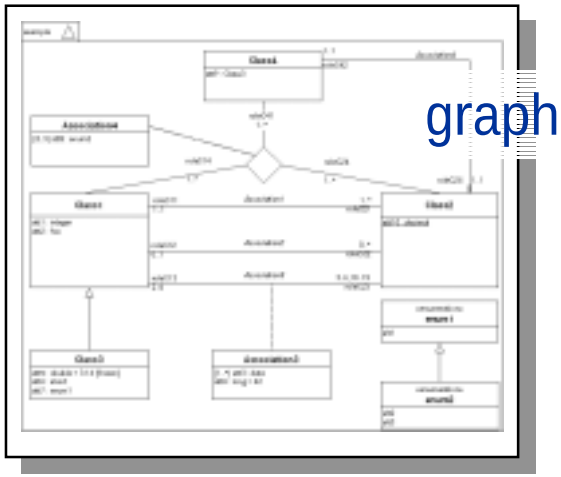
  Languages for similar applications domains should offer the same *look and feel*
- **Integration**

  Current development processes do not take care of serialization formats
- **Reuse**

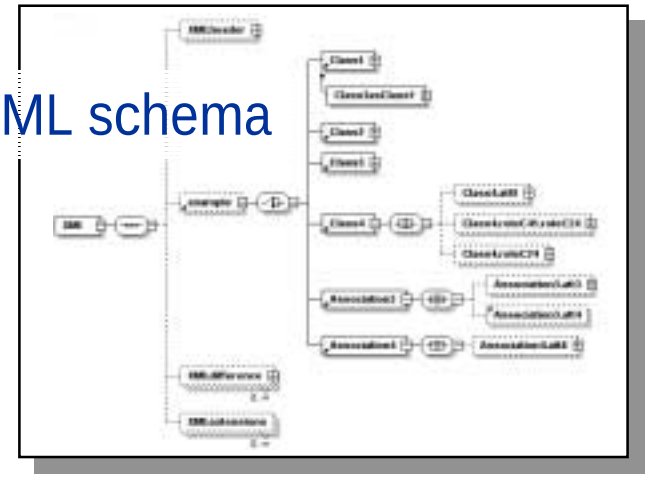
  Of existing design meta-knowledge

Seamless Derivation of XML-Schemata from UML Models

graphical UML model



graphical XML schema



export

visualization

XML representation

```
<?xml version="1.0" encoding="UTF-8"?>
<XMI xmi.version="1.0">
  <XMI.header>
    <XMI.documentation>
      <XMI.exporter>DaimlerChrysler Research and Technology FT3/EK</XMI.exporter>
      <XMI.exporterVersion>1.0</XMI.exporterVersion>
    </XMI.documentation>
    <XMI.metamodel xmi.name="UML" xmi.version="1.3"/>
  </XMI.header>
  <XMI.content>
    <Model_Management.Model xmi.id="Model:example">
      <Foundation.Core.ModelElement.name>example</Foundation.Core.ModelElement.name>
      <Foundation.Core.ModelElement.visibility xmi.value="public"/>
...
```

Derivation

using XSLT

XML Schema (XSD)

```
... encoding="UTF-8"?>
... xsd="http://www.w3.org/2001/XMLSchema"
... "http://www.w3.org/1999/xlink">
  <xsd:annotation>
    <xsd:documentation>Created by DaimlerChrysler's
    XSD-Generator 2001-04-05T14:34:00+2:00</xsd:documentation>
  </xsd:annotation>
  <xsd:attributeGroup name="LinkAttribs">
    <xsd:attribute name="href" type="xsd:string"/>
    <xsd:attribute name="idref" type="xsd:IDREF"/>
    <xsd:attribute name="link" type="xsd:string"/>
    <xsd:attribute name="xlink:inline" type="xsd:boolean"/>
    <xsd:attribute name="xlink:actuate">
      <xsd:simpleType>
...
```

OMG's Unified Modeling Language (UML)

- Object oriented modeling language for general purpose systems
- Includes most of precursor graphical modeling languages
- 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
- An OMG industry standard

- Defines no textual representation

  Subject to a separate RFP entitled *Stream-based Model Interchange Format*

OMG's XML Metadata Interchange (XMI)

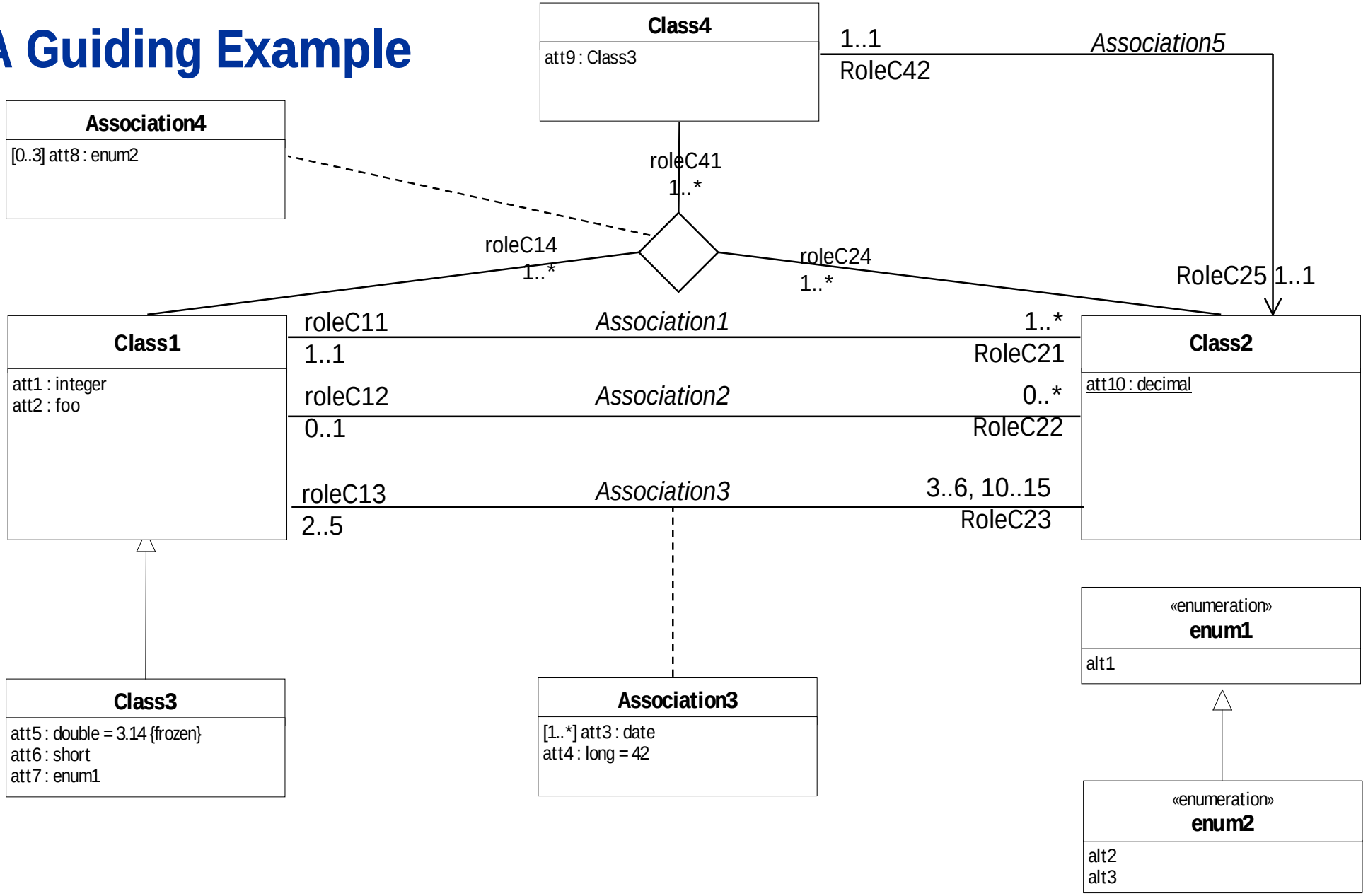
- **#1 goal**: Model Interchange a.k.a. Metadata Interchange among heterogeneous object oriented modeling tools

- **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

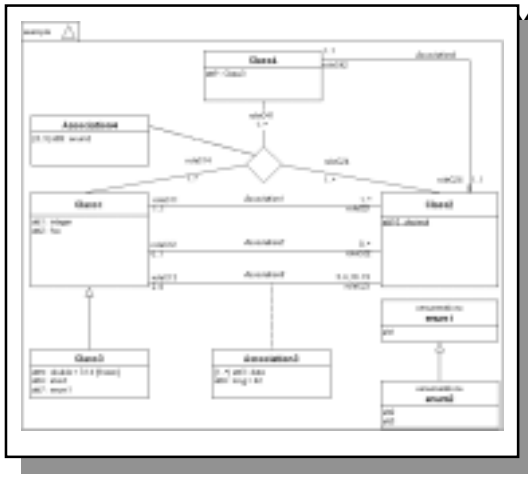
XMI Application Domains

- Model Interchange
- Metamodel Interchange
  - Interchange of Data Warehouse Models (e.g. OMG's CWM)
- Middleware related component information (e.g. CORBA components)
- Vertical Data integration (e.g. clinical information)
- Model Centric Development (not CASE-Tool centric)
  - Code generation (e.g. Java; also legacy systems)
  - Model assessments (e.g. metrics)
  - Assertion of modeling guidelines

A Guiding Example

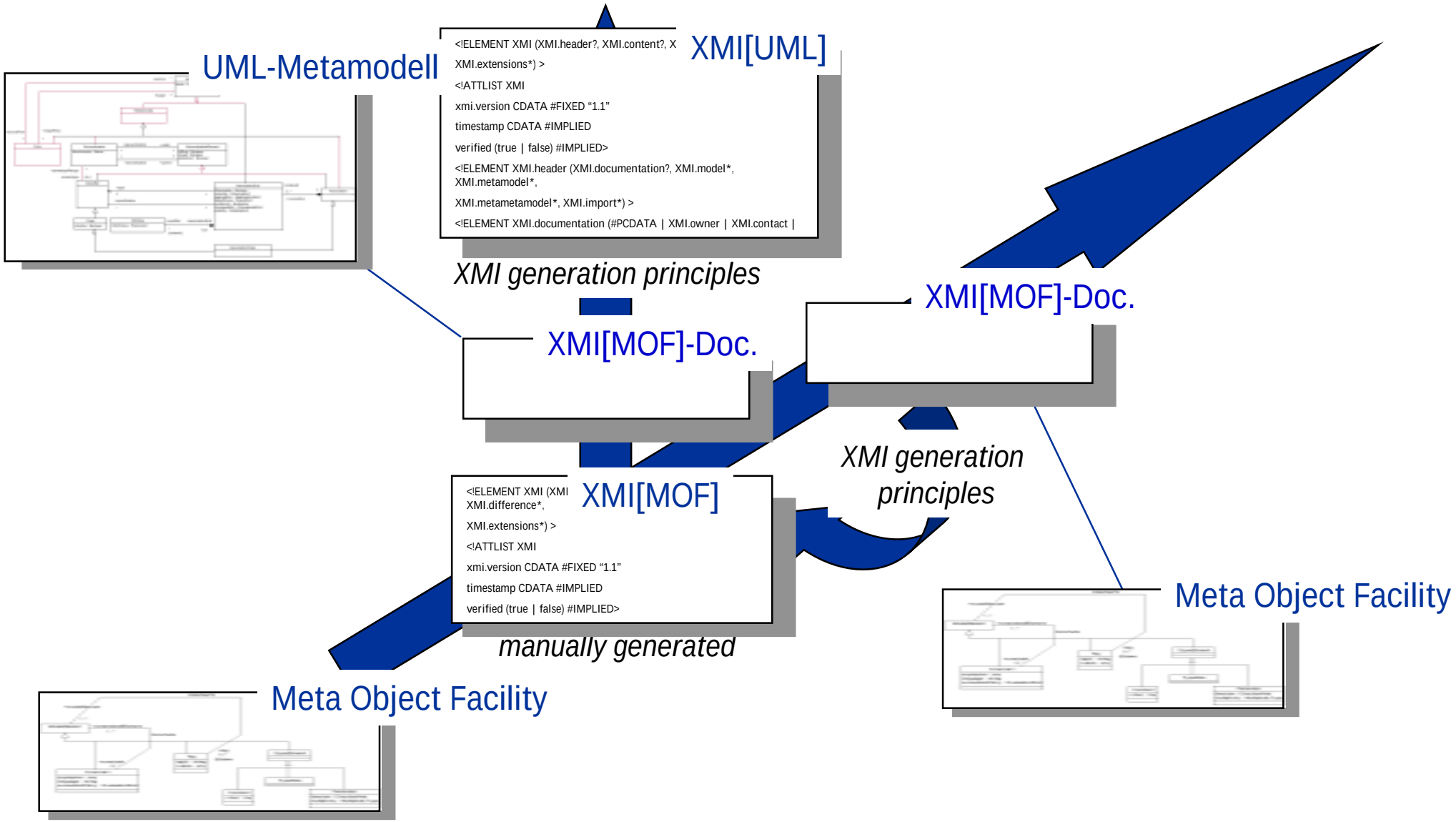


XMI[UML] to Encode Graphical UML Models Using XML

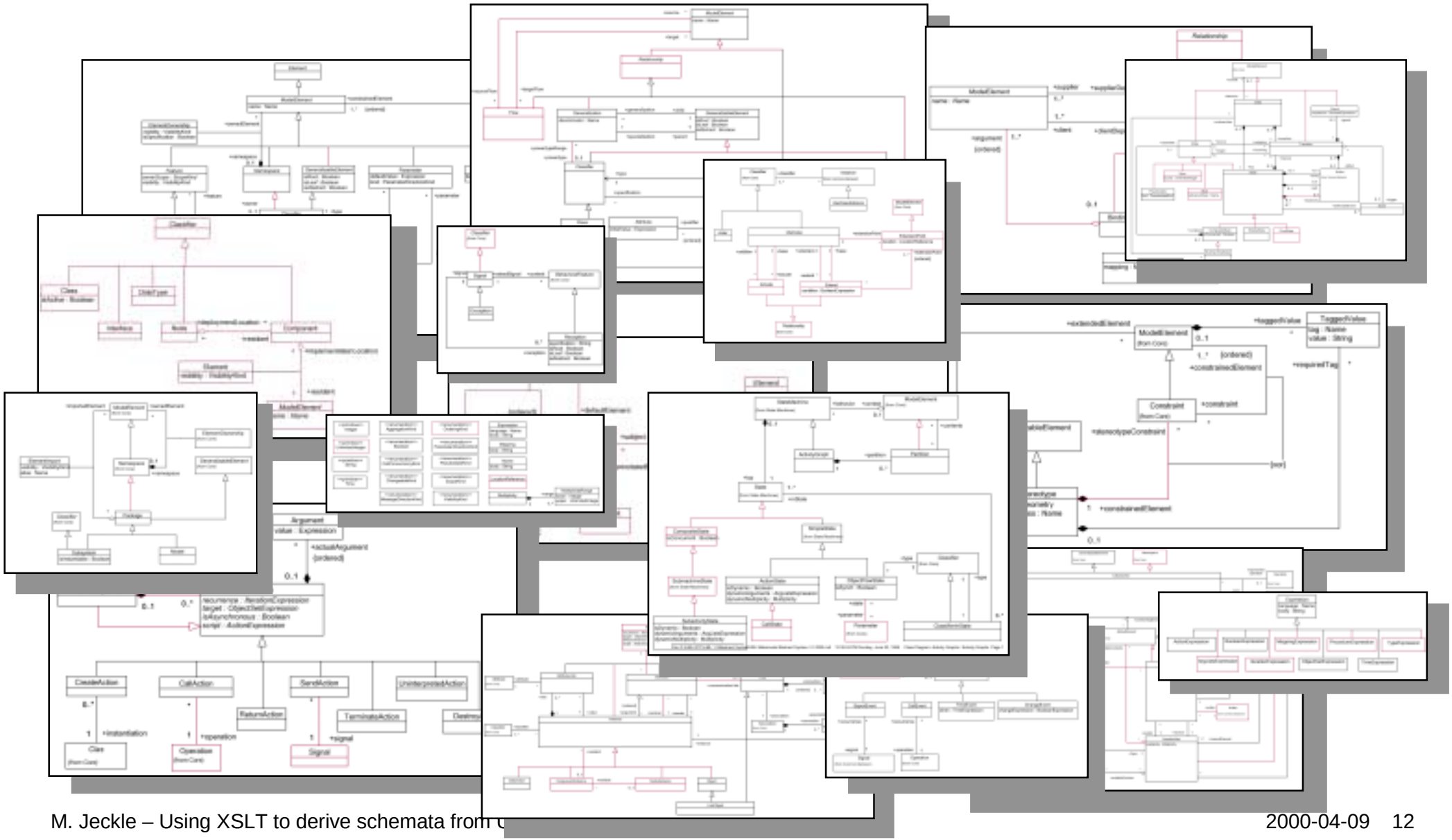


```
<?xml version="1.0" encoding="UTF-8"?>
<XMI xmi.version="1.0">
  <XMI.header>
    <XMI.documentation>
      <XMI.exporter>DaimlerChrysler Research and Technology FT3/EK</XMI.exporter>
      <XMI.exporterVersion>1.0</XMI.exporterVersion>
    </XMI.documentation>
    <XMI.metamodel xmi.name="UML" xmi.version="1.3"/>
  </XMI.header>
  <XMI.content>
    <Model_Management.Model xmi.id="Model:example">
      <Foundation.Core.ModelElement.name>example</Foundation.Core.ModelElement.name>
      <Foundation.Core.ModelElement.visibility xmi.value="public"/>
      <Foundation.Core.ModelElement.isSpecification xmi.value="false"/>
      <Foundation.Core.GeneralizableElement.isRoot xmi.value="false"/>
      <Foundation.Core.GeneralizableElement.isLeaf xmi.value="false"/>
      <Foundation.Core.GeneralizableElement.isAbstract xmi.value="false"/>
      <Foundation.Core.Namespace.ownedElement>
        <Foundation.Core.Class xmi.id = 'Class:Class1' >
          <Foundation.Core.ModelElement.name>Class1</Foundation.Core.ModelElement.name>
          <Foundation.Core.ModelElement.visibility xmi.value = "public"/>
          ...
```

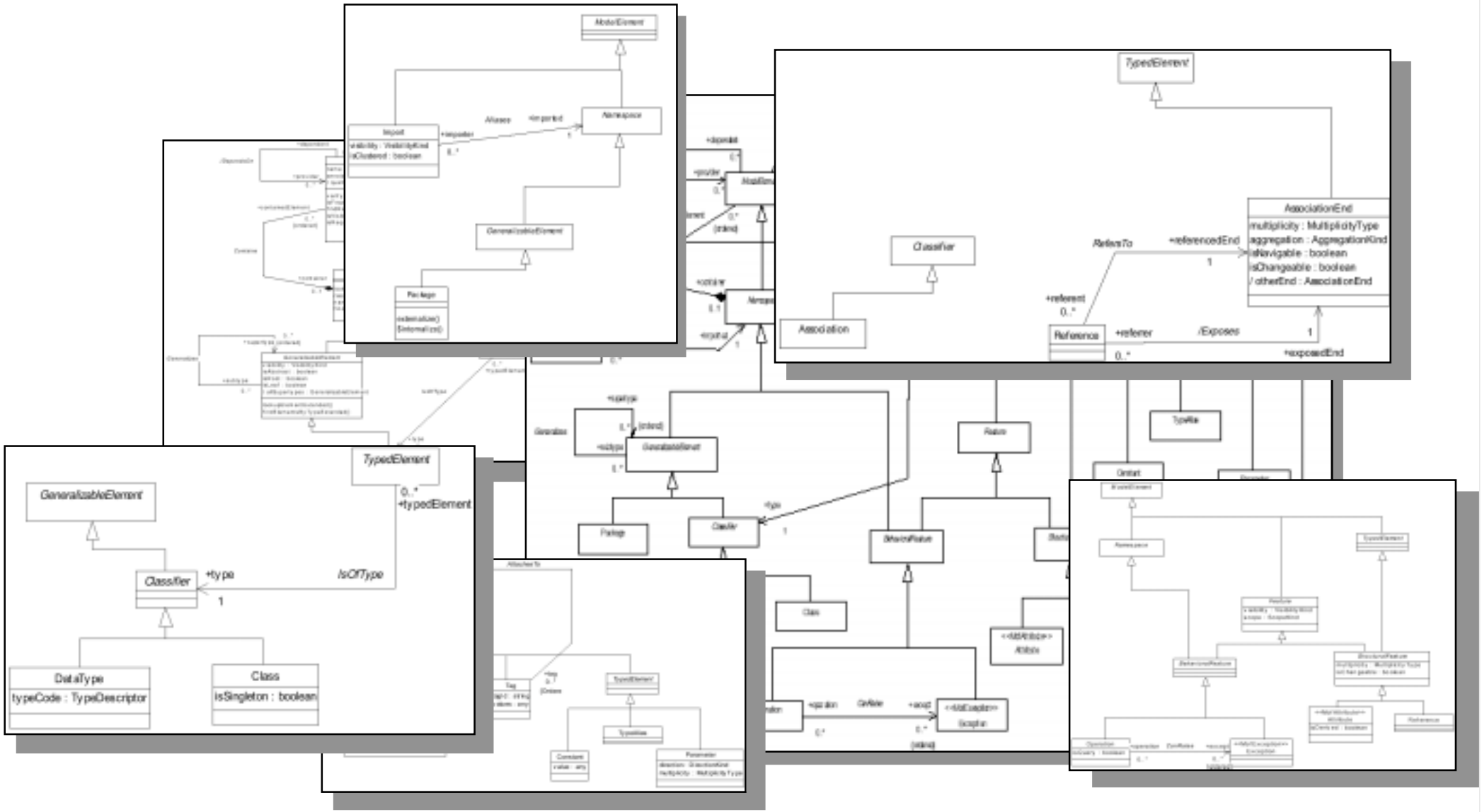
Generating the Standardized Document Type Definitions



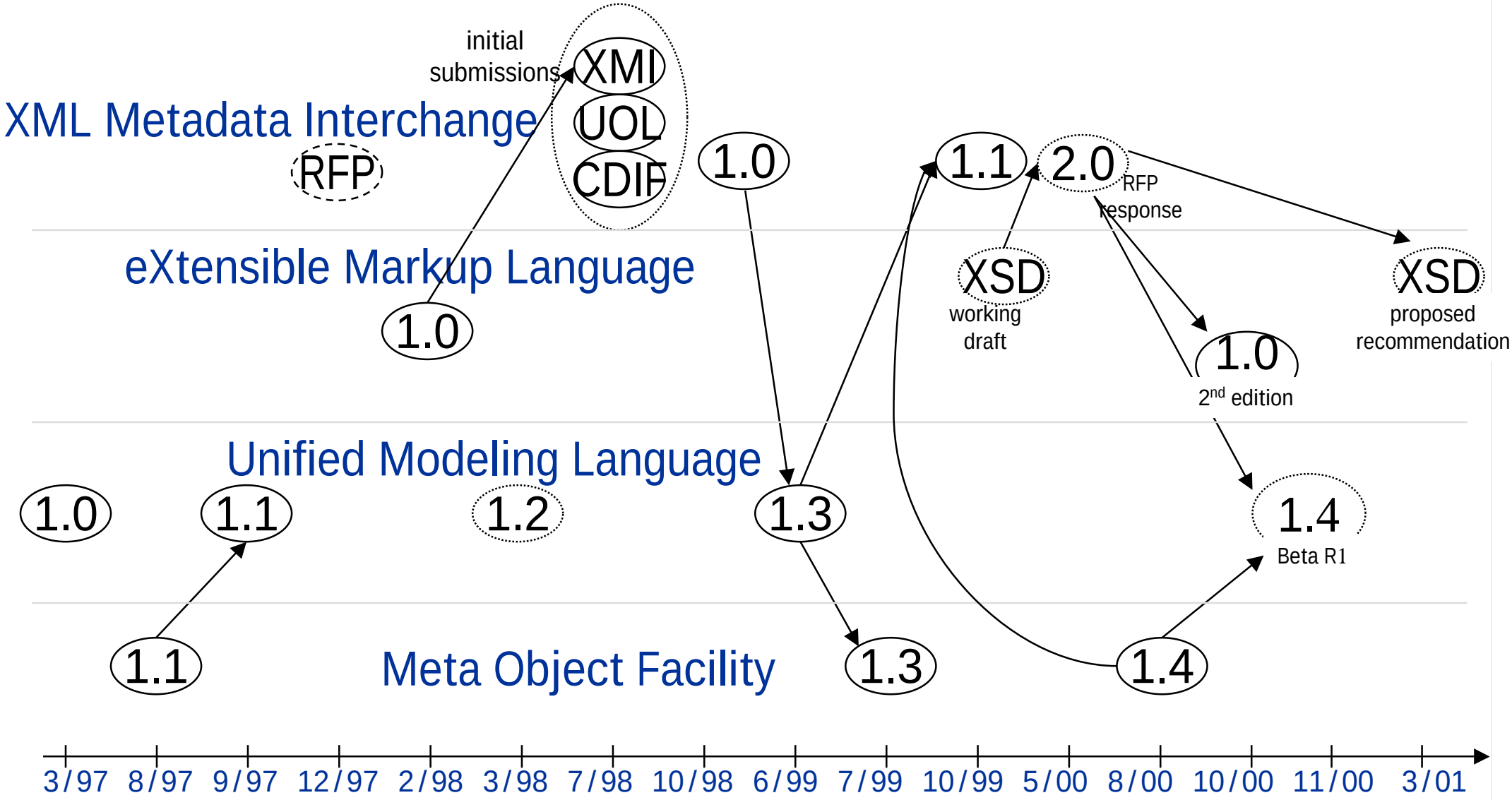
UML-Metamodell



MOF-Meta-Metamodell



XMI -- part of a world of interconnected standards

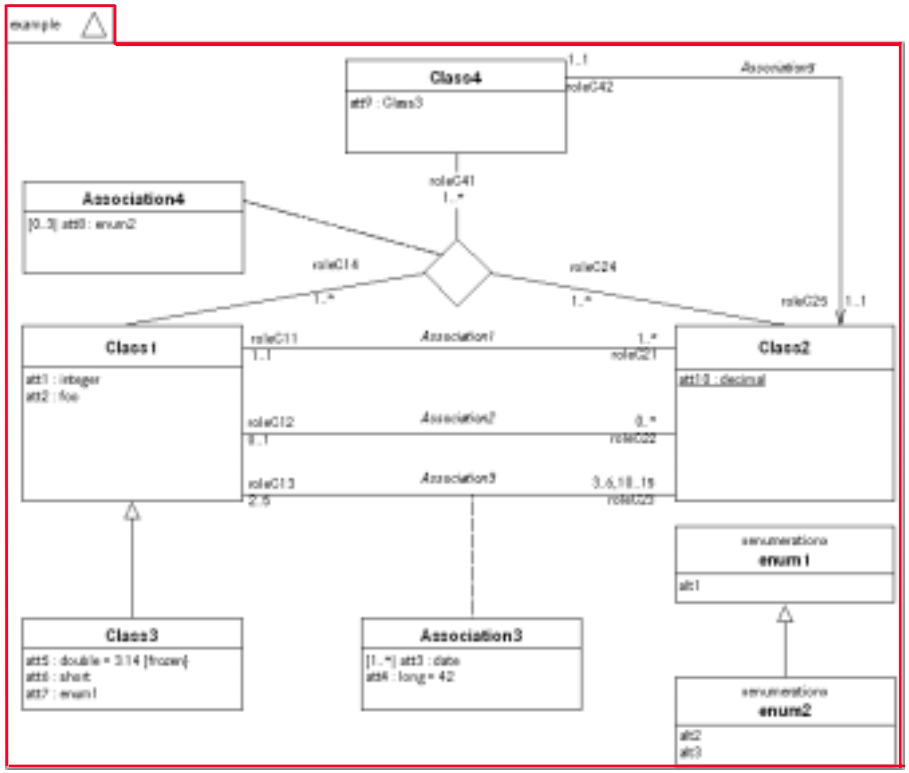


Structural Concepts of UML Relevant to Derivation Process

- Classes
  - Specialization relationships
  - Association classes
- Attributes
  - Constants
  - Default values
  - Multiplicity-aware (i.e. optional and multi valued attributes)
- Data types (primitive and complex)
  - pre-defined (by UML, MOF, and CORBA)
  - used-defined
- Associations
  - Navigability
  - Multiplicies

Derivation Patterns -- Model and Packages

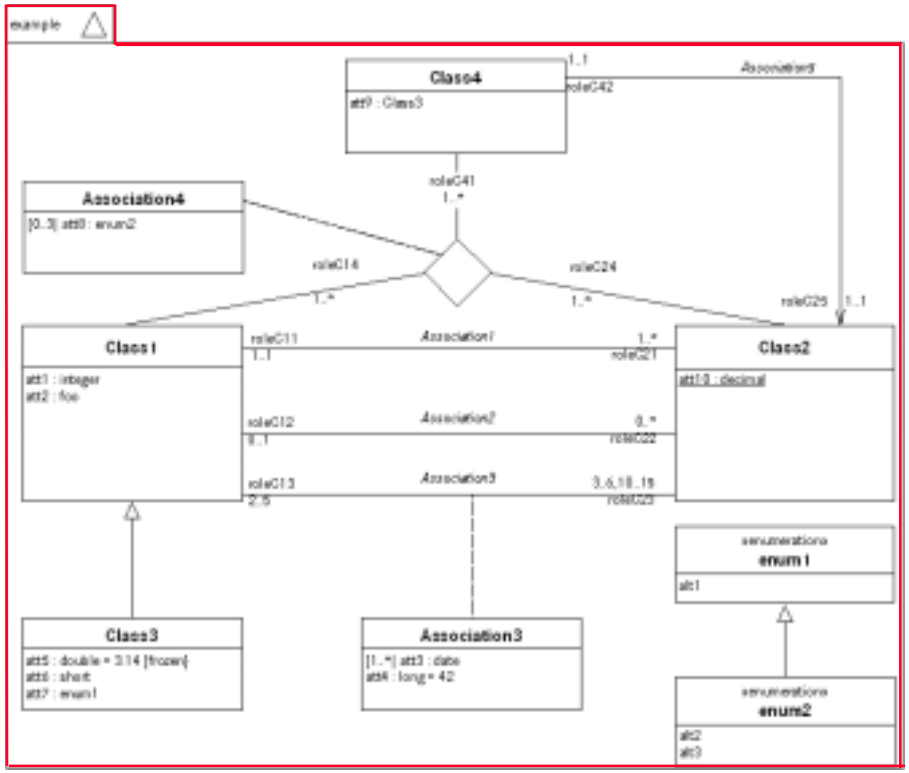
Match pattern:
/XMI/XMI.content/Model_Management.Model/Foundation.Core.ModelElement.name




```
<xsd:complexType>
  <xsd:choice minOccurs="0"
    maxOccurs="unbounded">
    <xsd:element ref="Class1"/>
    <xsd:element ref="Class2"/>
    <xsd:element ref="Class3"/>
    <xsd:element ref="Class4"/>
    <xsd:element ref="Association3"/>
    <xsd:element ref="Association4"/>
  </xsd:choice>
</xsd:complexType>
```

Derivation Patterns -- Model and Packages

Match pattern:
/XMI/XMI.content/Model_Management.Model/Foundation.Core.ModelElement.name




```
<xsd:complexType>
  <xsd:choice minOccurs="0"
    maxOccurs="unbounded">
    <xsd:element ref="Class1"/>
    <xsd:element ref="Class2"/>
    <xsd:element ref="Class3"/>
    <xsd:element ref="Class4"/>
    <xsd:element ref="Association3"/>
    <xsd:element ref="Association4"/>
  </xsd:choice>
</xsd:complexType>
```

Derivation Patterns -- Classes

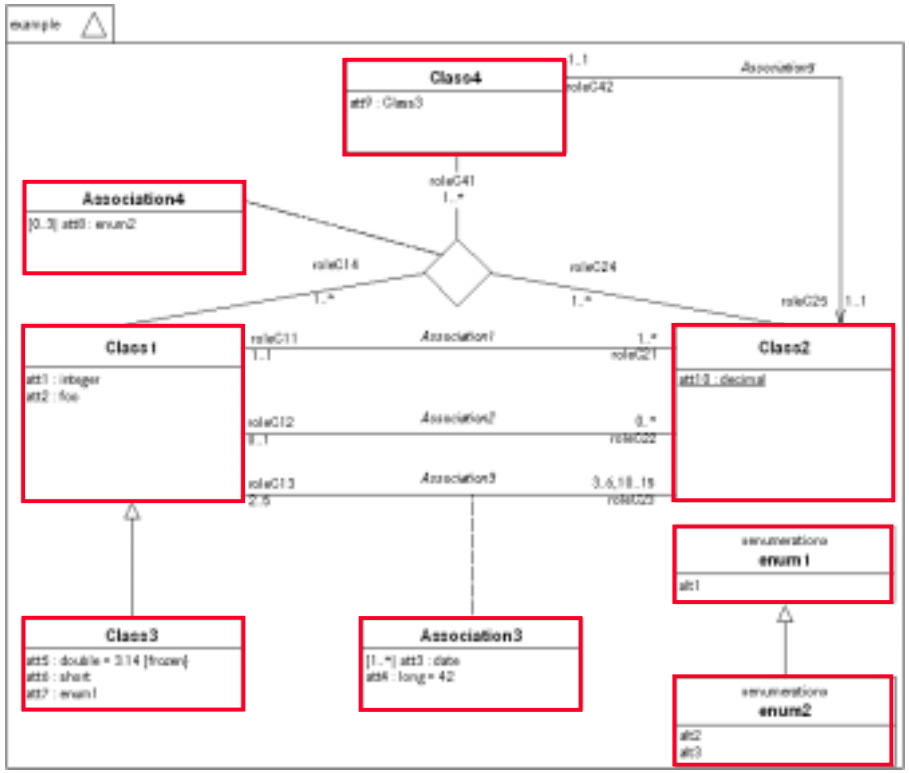
Purpose: Get all classes including association classes not stereotyped as *enumeration.*

Match pattern:

```
/XMI/XMI.content/Model_Management.Model/
Foundation.Core.Namespace.ownedElement//Foundation.Core.Class
[not(//Foundation.Extension_Mechanisms.Stereotype
[Foundation.Core.ModelElement.name='enumeration']/
@xmi.id=Foundation.Core.ModelElement.stereotype/
Foundation.Extension_Mechanisms.Stereotype/@xmi.idref)]
| /XMI/XMI.content/Model_Management.Model/
Foundation.Core.Namespace.ownedElement//Foundation.Core.AssociationClass
[not(//Foundation.Extension_Mechanisms.Stereotype
[Foundation.Core.ModelElement.name='enumeration']/
@xmi.id=Foundation.Core.ModelElement.stereotype/
Foundation.Extension_Mechanisms.Stereotype/@xmi.idref)]
```

Derivation Patterns -- Classes

Purpose: Get all classes including association classes not stereotyped as *enumeration.*

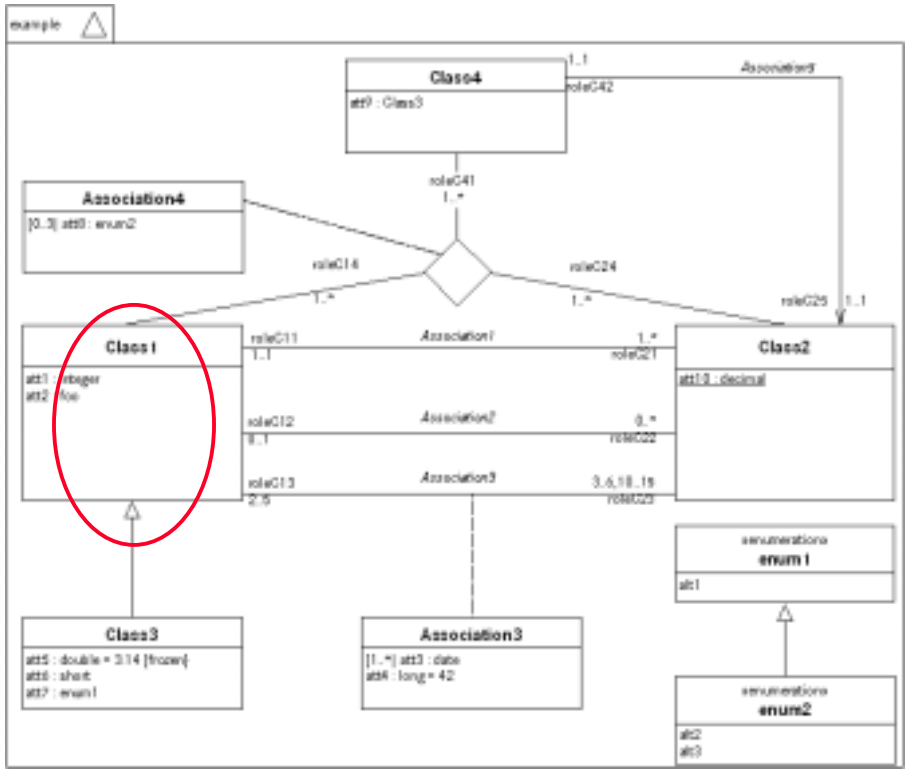



```
<xsd:complexType name="Class1Type">
   ...
</xsd:complexType>

<xsd:element  name="Class1"
              type="Class1Type"/>
```

Derivation Patterns -- Specialization of classes (Inheritance)

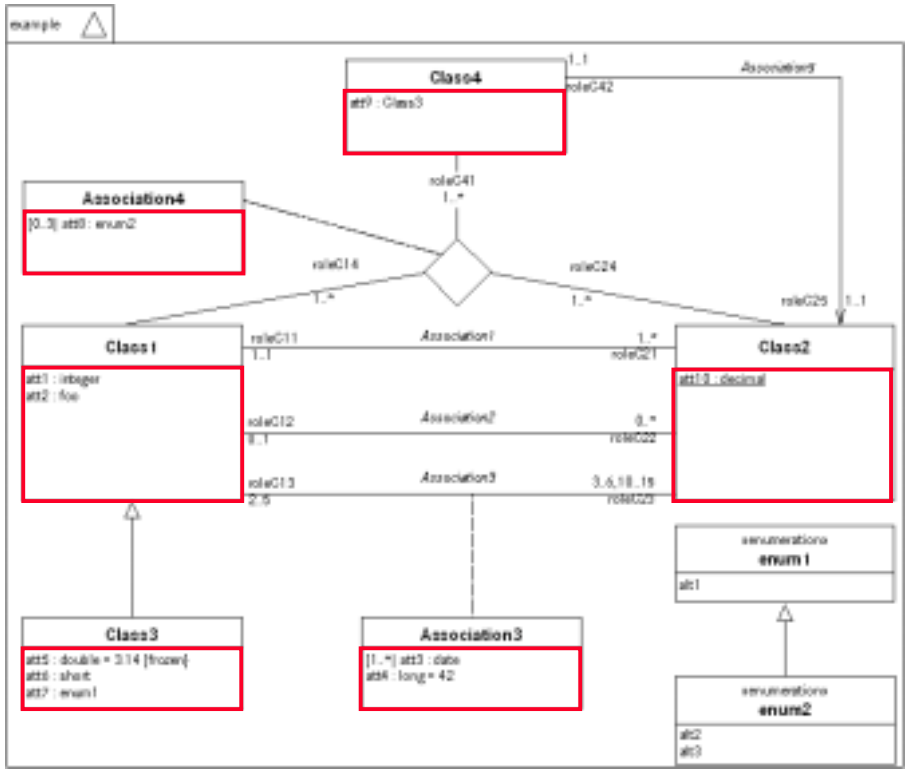
Purpose: Usage of XSD's *extension* mechanism for single inheritance
Processing of multiple generalizations by hand ...
Therefore, usage of *substitutionGroup*s, implementation of copy-down semantics by hand




```
<xsd:complexType name="Class3Type">
  <xsd:complexContent>
    <xsd:extension base="Class1Type">
      ...
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

Derivation Patterns -- Attributes

Purpose: Derivation of XSD content models according the the semantics (e.g. data types) and meta semantics (e.g. UML structuring principles) formulated by the UML model

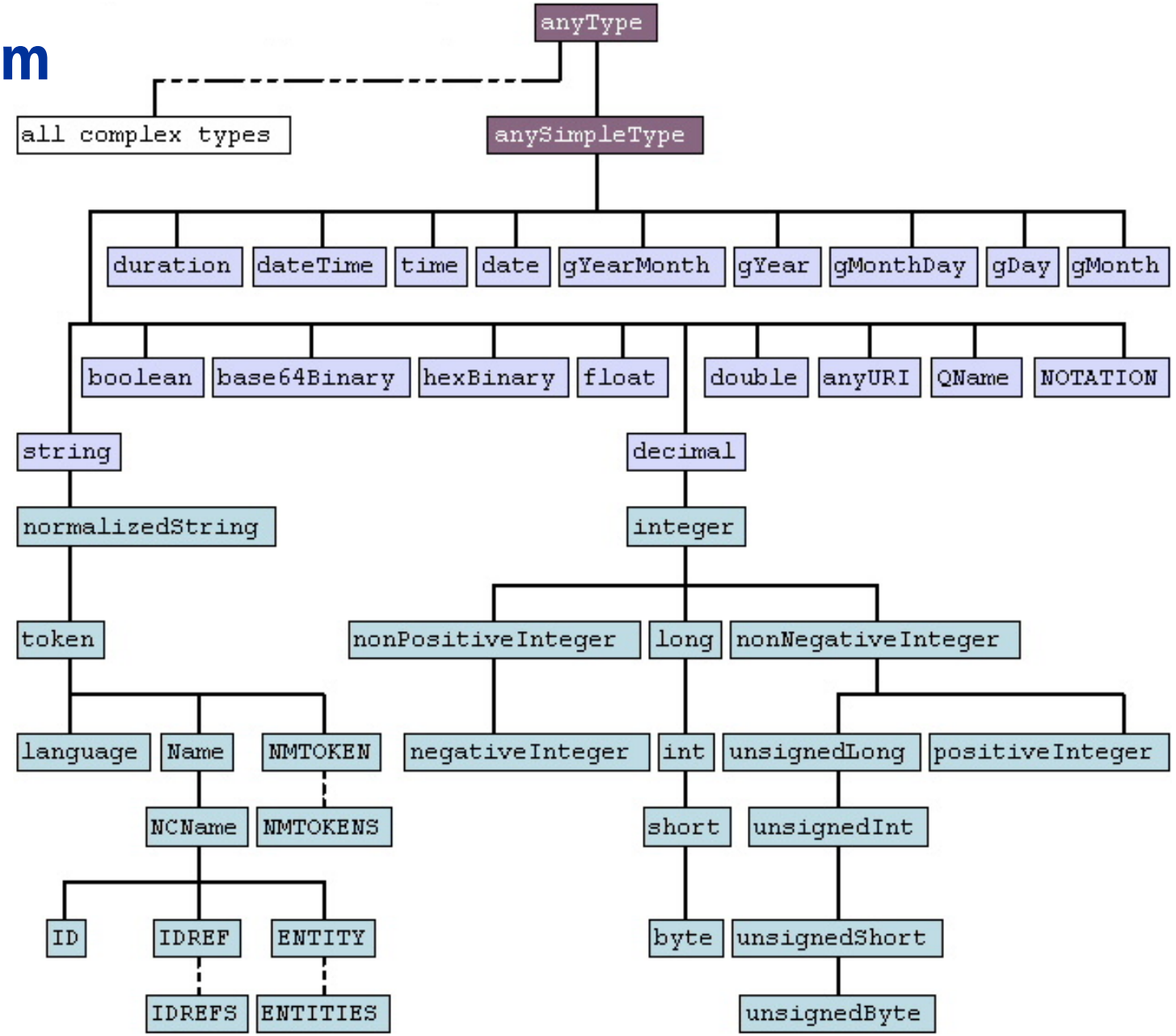



```
<xsd:element name="attribut'sName Type"
              type="matchingType"/>

<xsd:complexType name="Class1Type">
  ...
  <xsd:element ref=" attribut'sName Type"/>
  ...
</xsd:complexType>
```

XSD's Type System

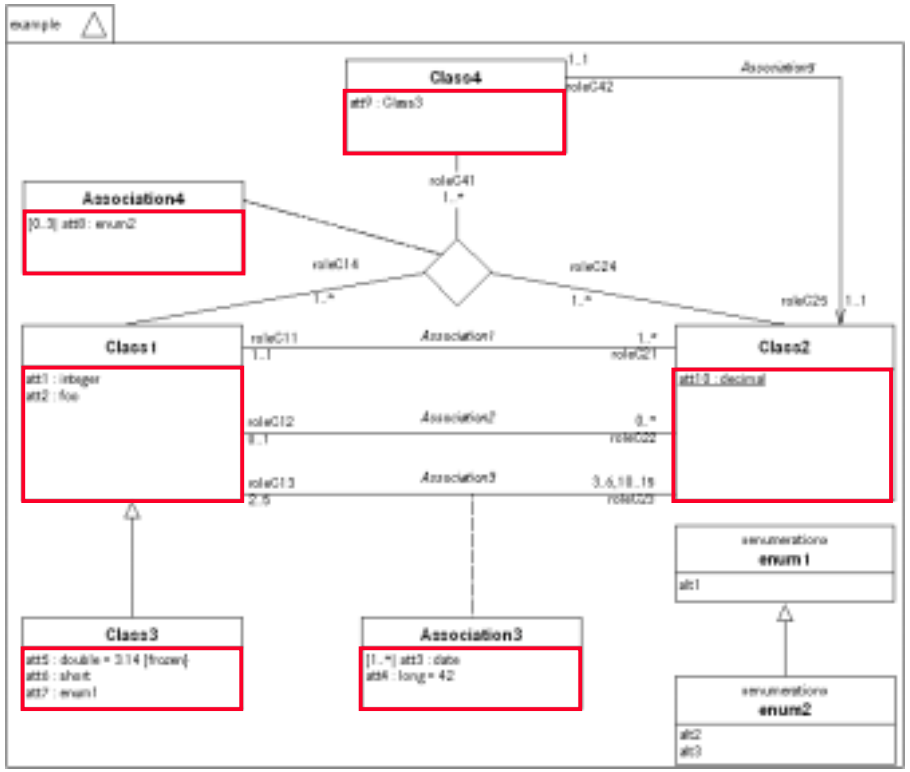
- ur type
- built-in primitive types
- built-in derived types
- derived by restriction
- derived by list



cf. XML-Schema Part 2: Datatypes, PR 2001-03-30, chap. 3

Derivation Patterns -- Attributes

Purpose: Identify the *matchingType* ...
I. data types predefined by UML / MOF / CORBA



Integer → integer

String → string

Name → Name

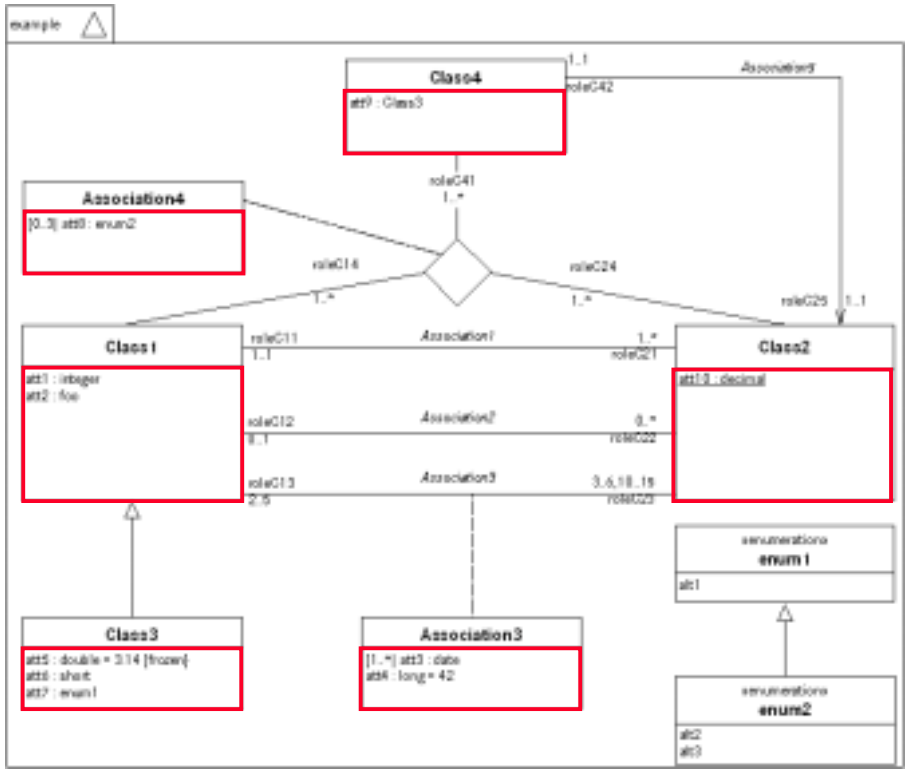
Boolean → boolean

Time → ???

Uninterpreted → base64Binary, hexBinary

Derivation Patterns -- Attributes

Purpose: Identify the *matchingType* ...
I. data types predefined by UML / MOF / CORBA

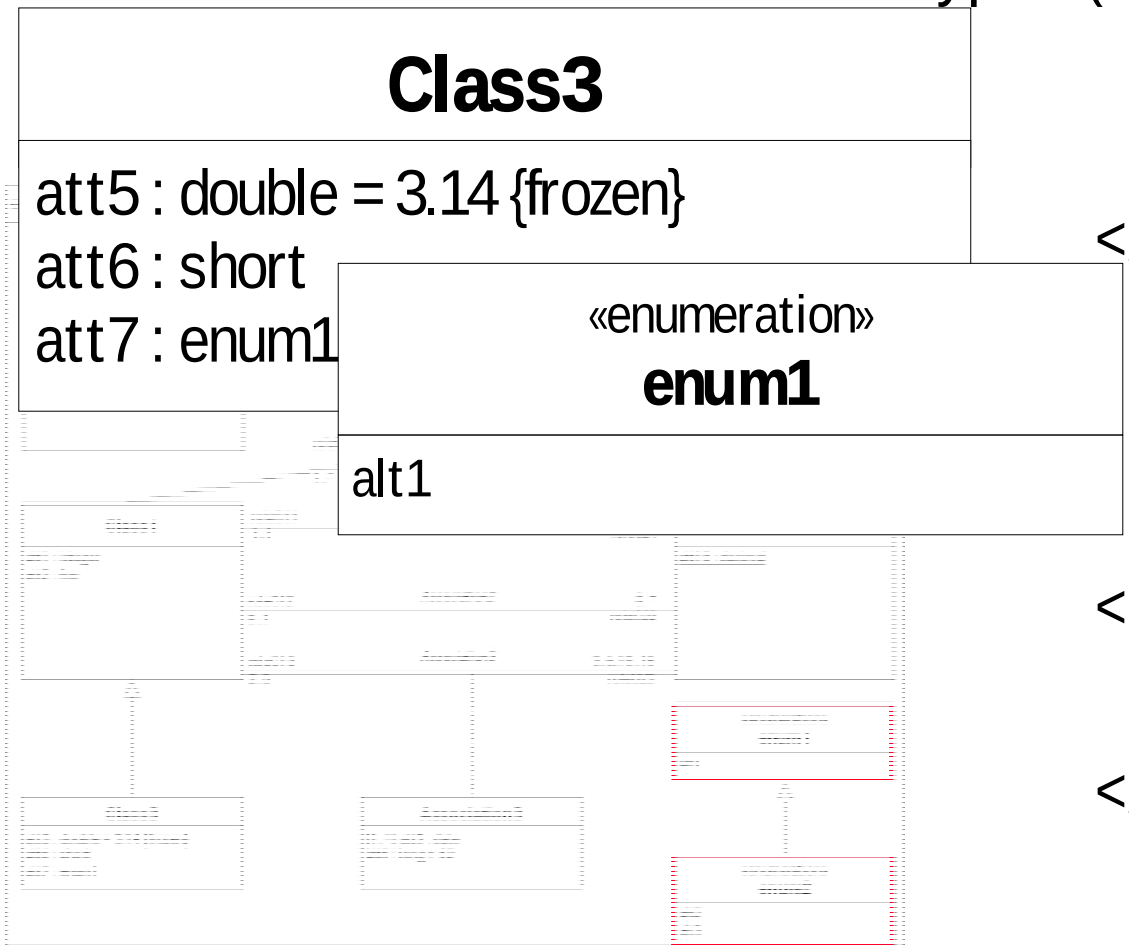


Time ⟶

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

Derivation Patterns -- Attributes

Purpose: Identify the *matchingType* ...
II. used defined data types (e.g. *enumeration types*)

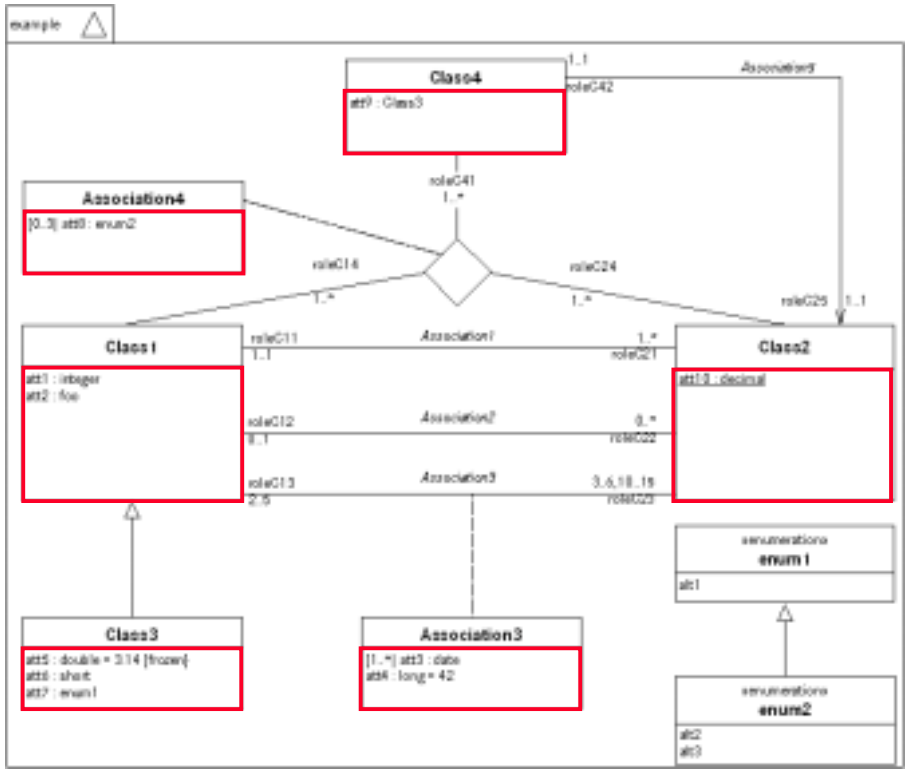



```
<xsd:simpleType name="enum1Type">
<xsd:restriction base="xsd:string">
  <xsd:enumeration value="alt1"/>
</xsd:restriction>
</xsd:simpleType>

<xsd:element name="att7Type"
                type="enum1Type"/>
```

Derivation Patterns -- Attributes

Purpose: Identify the *matchingType* ...
III. (direct) usage of XSD's pre-defined types

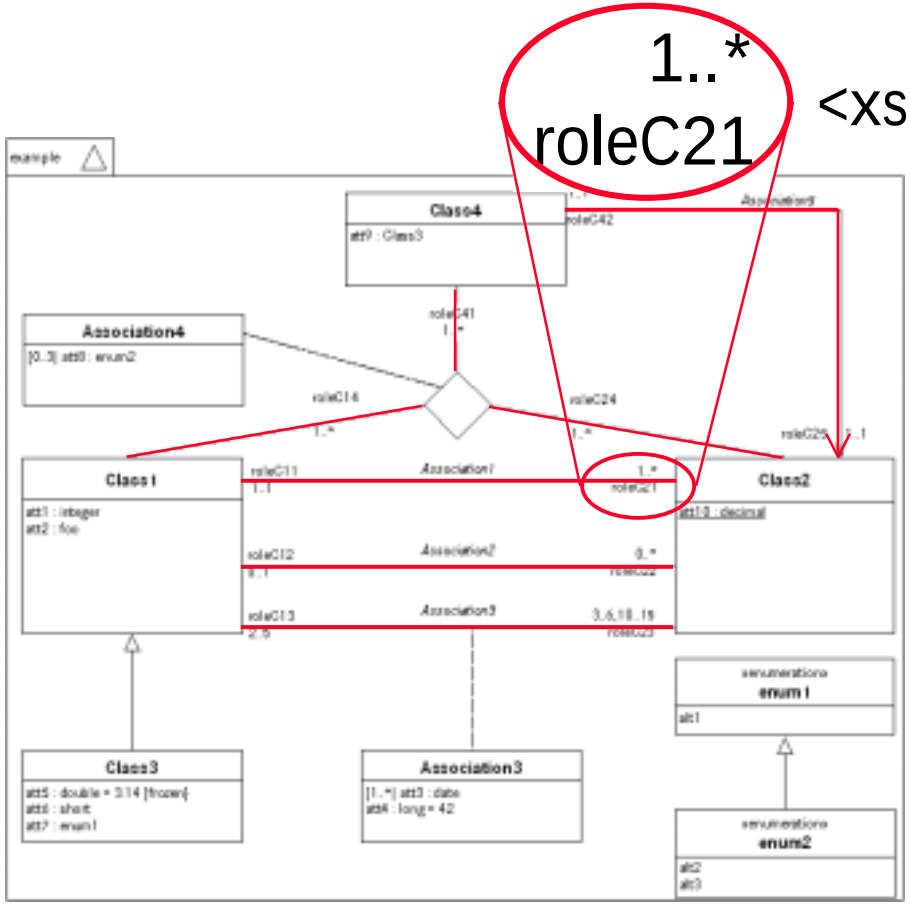


normalizedString	integer
token	nonPositiveInteger
language	negativeInteger
IDREFS	long
ENTITIES	int
NMTOKEN	short
NMTOKENS	byte
Name	nonNegativeInteger
NCName	unsignedLong
ID	unsignedInt
IDREF	unsignedShort
ENTITY	unsignedByte
gYearMonth	positiveInteger
gYear	string
gMonthDay	boolean
gDay	decimal
gMonth	float
hexBinary	double
base64Binary	duration
anyURI	dateTime
QName	time
NOTATION	date

Derivation Patterns -- Associations

Purpose: Transformation from the net to structure of trees

Basic Idea: Modification of all minimal multiplicities to zero




```
<xsd:element name="Class1" type="Class1Type">
  <xsd:complexType>
    <xsd:all>
      <!-- attributes -->
      <xsd:element name="Class2.role21"
                   ref="Class2Type"
                   minOccurs="0"
                   maxOccurs="unbounded"/>
      ...
    </xsd:all>
</xsd:element>
```

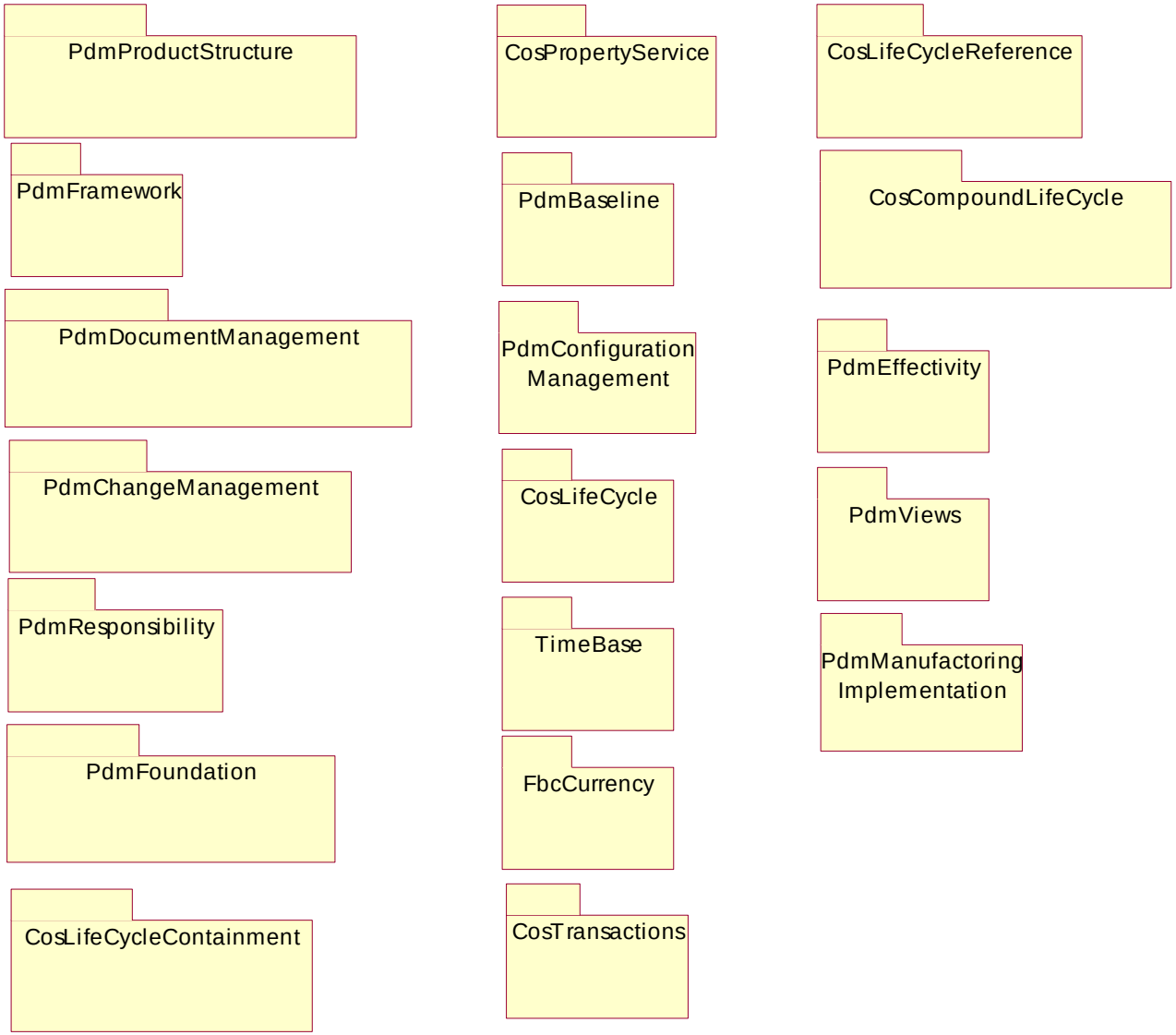
Derivation Patterns -- Handling Redundancy

Purpose: Coping with possibly multiple referenced elements

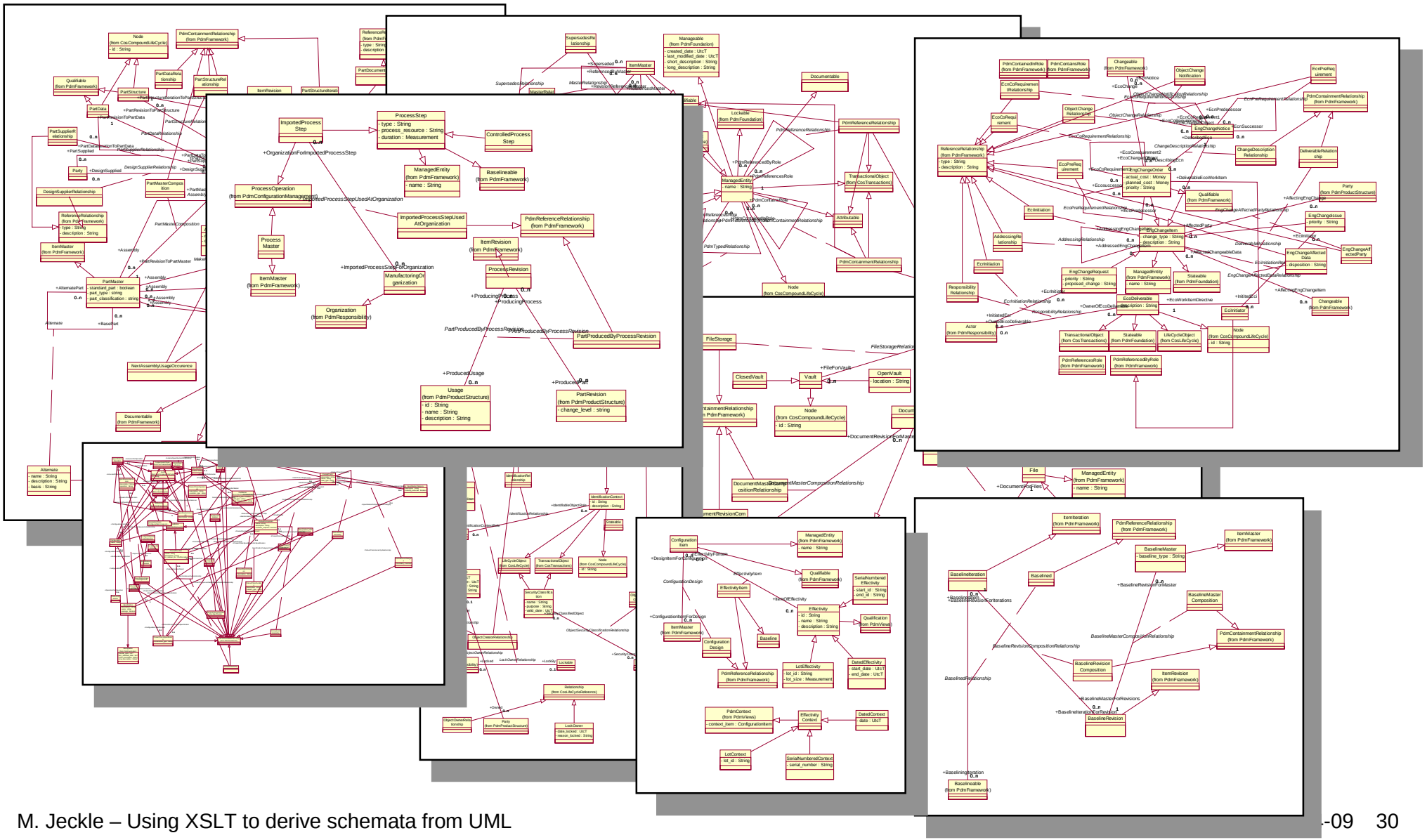
Basic Idea: Definition of unique identifier at class level,
separating defining occurence from pure reference

```
<xsd:element name="Class1" type="Class1Type">
  <xsd:all>
    <xsd:element name="Class1.att1" type="xsd:integer"
                 minOccurs="0"/>
    ...
  </xsd:all>
   <xsd:attribute name="xmi.id" type="ID"/>
   <xsd:attribute name="xmi.idref" type="IDREF"/>
...
</xsd:element>
```

OMG's PDM-Enablers



OMG's PDM-Enablers



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 craftsmen's approach
- Coherent, built-in synchronization between data model and XML voc.
- 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