Defining XML vocabularies

DTDs and XML Schemas

What is an XML vocabulary?

- Synonyms
  - ‘Application of XML’
  - XML Language
- Set of elements and attributes for representing domain-specific information
- “Instance” of a Mark Up Language
- Some are approved by standard organisations
  - E.g. ebXML, MathML, XSL etc.

Remember: XML is syntax!

Why have a definition of a vocab?

- Validate data in order to find inconsistencies
- Indicates what structures and names can be used in the data
- Data constructed and named in a conformant manner leads to
  - Easier construction by supplier (provides structure)
  - Easier parsing by consumer

Well formed XML

- XML Declaration required
- At least one element
  - Exactly one root element
- Empty elements are written in one of two ways:
  - Closing tag (e.g. "<br></br>")
  - Special start tag (e.g. "<br />")
- For non-empty elements, closing tags are required
- Attribute values must always be quoted
- Start tag must match closing tag (name & case)
- Correct nesting of elements
  - Example incorrect nesting and incorrect case

```
<full_name>
<first_name>
John </Full_name>
</first_name>
```

Valid XML

- Well-formed plus conforms to DTD or XML Schema
- All elements and attributes are declared within a DTD/XML Schema
- Elements and attributes match the declarations in the DTD/XML Schema

What is a DTD?

- Document Type Definition,
- Defines structure/model of XML documents
  - Elements and Cardinality
  - Attributes
  - Aggregation
- Defines default ATTRIBUTE values
- Defines ENTITIES
- Stored in a plain text file and referenced by an XML document (external)
- Alternatively a DTD can be placed in the XML document itself (internal)

Element Type Declaration

- Define grouping of elements
  - "(", ")"
- Define sequence of elements
  - ",": followed-by (Sequence)
  - "|": logical or (Choice)

```
<!ELEMENT doc
   (title, author, editor,
   chapter, appendix)>

<!ELEMENT title (#PCDATA)>

<!ELEMENT author
   (name | synonym)>

<!ELEMENT image EMPTY>

<!ELEMENT paragraph
   (#PCDATA | bold | italic)*>
```

Element Type Declaration

- Define occurrences of elements
  - ?: zero-or-one
  - +: one-or-more
  - *: zero-or-more

```
<!ELEMENT doc
   (title, author+, editor?,
  chapter+, appendix*)>

<!ELEMENT chapter
   (title,
    (section+ | paragraph+))>

<!ELEMENT list
   (item?, item?, item)>

<!ENTITY % list "ordered |
  unordered | definition">

<!ELEMENT paragraph
   (#PCDATA | %list;)*>
```

Entity Declaration

- Internal entities
  - Built-in

```
<!ENTITY author
   "Norman Walsh, Sun Corp.">
```

- External entities
  - References to a file (text, images etc.)

```
<!ENTITY copyright
   SYSTEM "copyright.xml">
```

- Parameter entities
  - Used inside DTDs

```
<!ENTITY % part
   "(title?, (paragraph |
   section)*)">
```

Attribute List Declaration

- Define type of attribute
  - CDATA
  - ID
  - IDREF
  - ENTITY
  - NMTOKEN
  - NOTATION
- Define default values of attributes
  - #REQUIRED
  - #IMPLIED
  - #FIXED
  - A list of values with default selection

```
<!ATTLIST person
   ssn ID #IMPLIED>


<!ATTLIST adult
   age CDATA #REQUIRED>


<!ATTLIST mml
   version ‘1.0’ #FIXED>


<!ATTLIST person
    sex (m | f) #REQUIRED>


<!ATTLIST day
   temperature (l | m | h) "l">
```

Simple DTD Example

```
<!DOCTYPE doc[
<!ENTITY % part "(title?, (paragraph | section)*)">

<!ELEMENT doc (title, author+, chapter+, appendix*)>
<!ATTLIST doc type (book | article) "book“
              isbn CDATA #REQUIRED>

<!ELEMENT title (#PCDATA)>
<!ELEMENT author (#PCDATA)>
<!ELEMENT chapter %part;>
<!ELEMENT appendix %part;>
<!ELEMENT section %part;>
<!ELEMENT paragraph (#PCDATA | url | ol)*>
<!ATTLIST paragraph type CDATA #IMPLIED>
<!ELEMENT ol (item+)>
<!ELEMENT item (paragraph+)>
<!ELEMENT url (#PCDATA)>
]>
```

Over to you…

- Possible DTD for following?

```
<database>
<person age='34'>
      <name>
                <title> Mr </title>
                <firstname> John </firstname>
                <firstname> Paul </firstname>
                <surname> Murphy </surname>
      </name>
      <hobby> Football </hobby>
      <hobby> Racing </hobby>
</person>

<person >
      <name>
                <firstname> Mary </firstname>
                <surname> Donnelly </surname>
      </name>
</person>
</database>
```

Over to you…

```
<database>
<person age='34'>
      <name>
            <title> Mr </title>
            <firstname> John </firstname>
            <firstname> Paul </firstname>
            <surname> Murphy </surname>
      </name>
      <hobby> Football </hobby>
      <hobby> Racing </hobby>
</person>

<person >
      <name>
            <firstname> Mary </firstname>
            <surname> Donnelly </surname>
      </name>
</person>
</database>
```

```
<!DOCTYPE database [

<!ELEMENT database (person*)>

<!ELEMENT person (name,hobby*)>
<!ATTLIST person age CDATA #IMPLIED>

<!ELEMENT name (title?, firstname+,
   surname)>

<!ELEMENT hobby (#PCDATA)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT firstname (#PCDATA)>
<!ELEMENT surname (#PCDATA)>

]>
```

```
<?xml version="1.0"!>
<!DOCTYPE catalog SYSTEM "books.dtd">
<catalog>
   <book id='bk101' type='softback'>
      <author>Gambardella, Matthew</author>
      <title>XML Developer's Guide</title>
      <genre>Computer</genre>
<price>44.95</price>
      <publish_date>2000-10-01</publish_date>
      <description>An in-depth look at creating
applications with XML.
</description>
</book>
<book id='bk102' type='hardback'>
      <author nationality='irish'>Jenkins,
Fred</author>
      <title>XML Technology Guide</title>
         <price>50.00</price>
      <publish_date>2000-10-01</publish_date>
      <description>An in-depth look at using XML
technologies.</description>
         <stocked_by>Easons</stocked_by>
         <stocked_by>Amazon</stocked_by>
   </book >
</catalog>
```

EXAMPLE SYNTAX

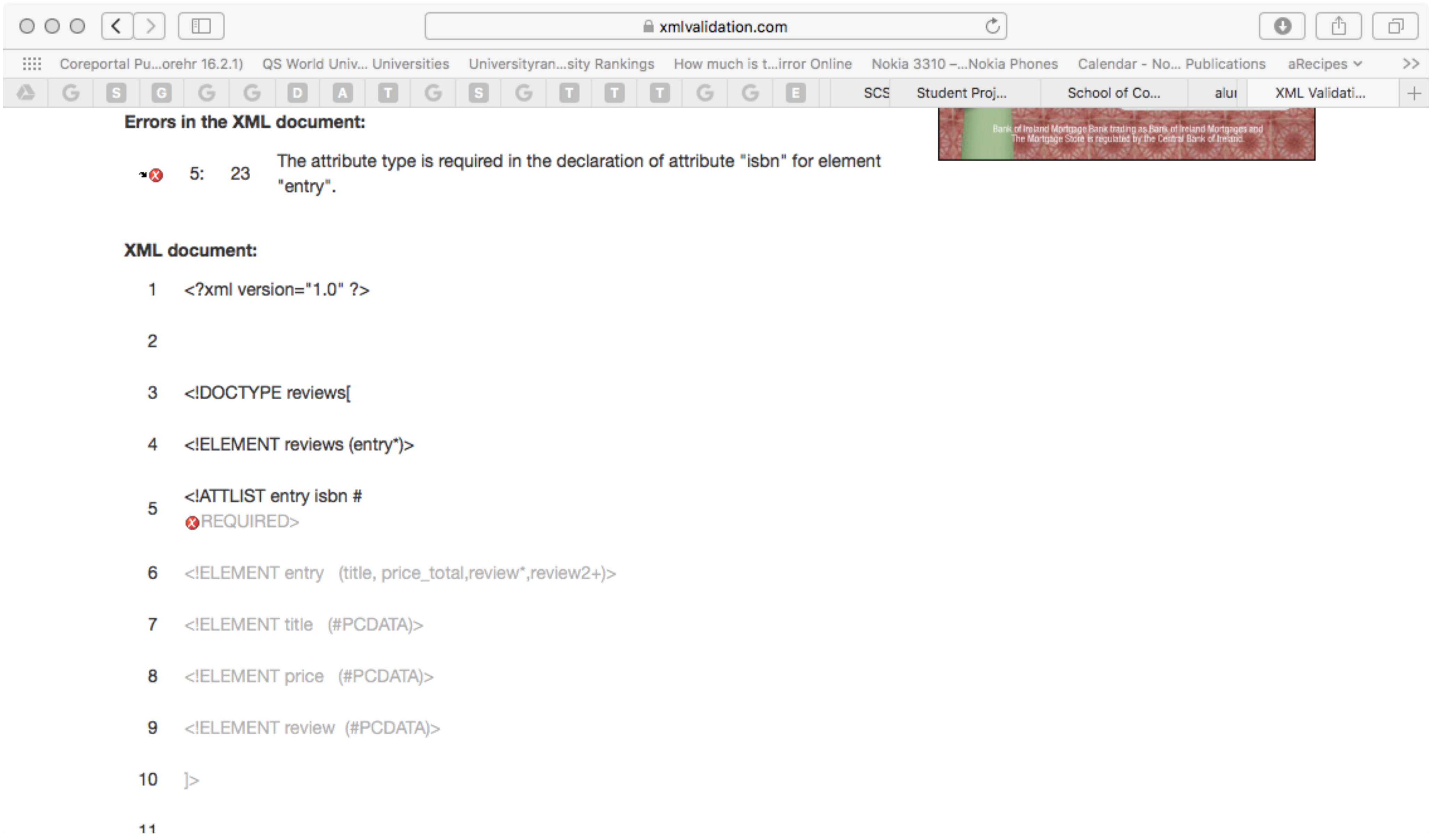
```
<!DOCTYPE NEWSPAPER [

<!ELEMENT NEWSPAPER (ARTICLE+)>
<!ELEMENT ARTICLE
      (HEADLINE,BYLINE+,LEAD?,BODY,NOTES*)>
<!ELEMENT HEADLINE (#PCDATA)>
<!ELEMENT BYLINE (#PCDATA)>
<!ELEMENT LEAD (#PCDATA)>
<!ELEMENT BODY (#PCDATA)>
<!ELEMENT NOTES (#PCDATA)>

<!ATTLIST ARTICLE AUTHOR CDATA #REQUIRED>
<!ATTLIST ARTICLE EDITOR CDATA #IMPLIED>
<!ATTLIST ARTICLE DATE CDATA #IMPLIED>
<!ATTLIST ARTICLE EDITION CDATA #IMPLIED>

<!ENTITY NEWSPAPER "Trinity Times">
<!ENTITY PUBLISHER "Trinity Press">
<!ENTITY COPYRIGHT "Copyright 1998 TCD Press">

]>
```

SUGGEST A DTD

DTD

```
<!DOCTYPE catalog [
<!ELEMENT catalog    (book+) >
<!ELEMENT book       (author, title, genre?,
price, publish_date, description,stocked_by*) >
<!ATTLIST book   id ID #REQUIRED >
<!ATTLIST book   type (hardback|softback)
#REQUIRED >

<!ELEMENT author            (#PCDATA)   >
<!ATTLIST author nationality CDATA #IMPLIED >
    <!ELEMENT title             (#PCDATA)   >
    <!ELEMENT genre             (#PCDATA)   >
    <!ELEMENT price             (#PCDATA)   >
    <!ELEMENT publish_date      (#PCDATA)   >
    <!ELEMENT description       (#PCDATA)   >
<!ELEMENT stocked_by    (#PCDATA)   >
]>
```

XMLVALIDATION.COM



xmlvalidation.com
Coreportal Pu...orehr 16.2.1) QS World Univ... Universities Universityran...sity Rankings How much is t...irror Online Nokia 3310 –...Nokia Phones Calendar - No... Publications aRecipes
SCS Student Proj... School of Co... alu XML Validati...

Errors in the XML document:

5: 23 The attribute type is required in the declaration of attribute "isbn" for element "entry".

XML document:

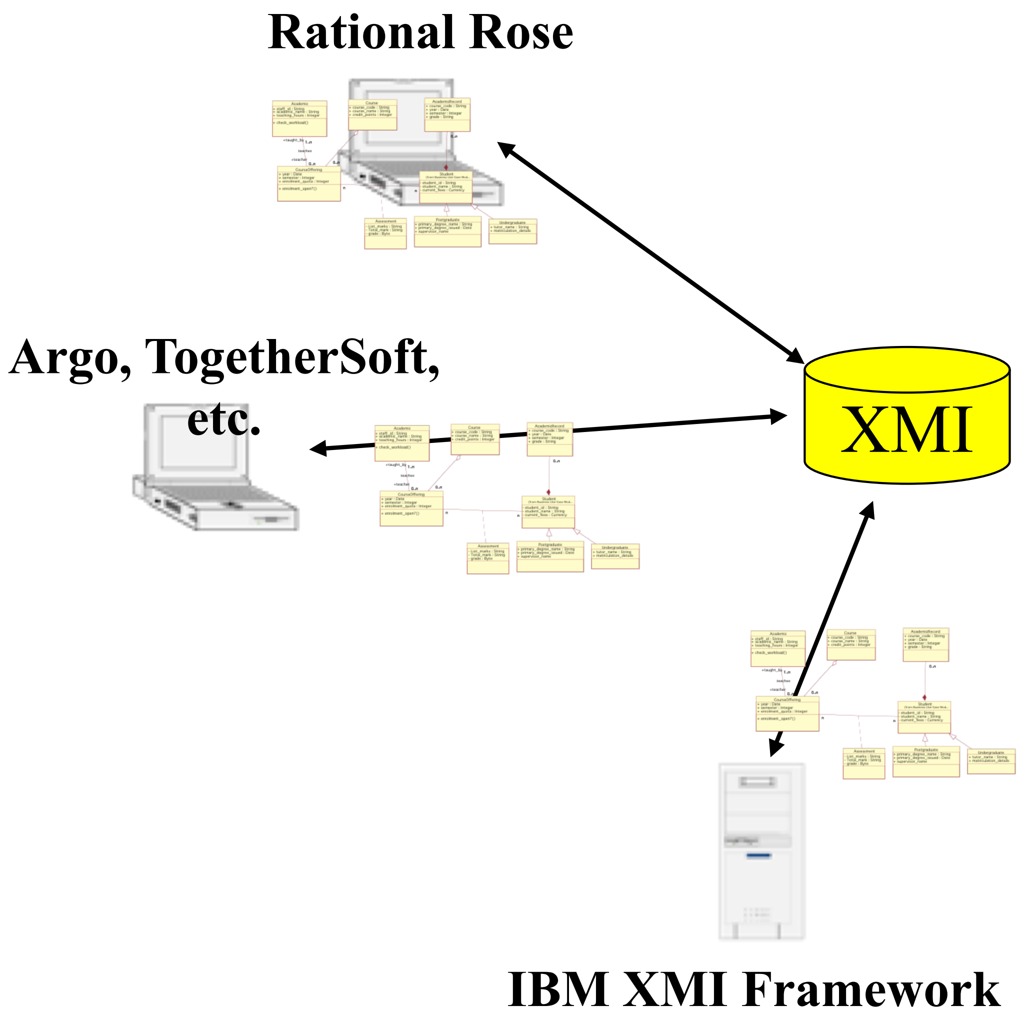
```
<?xml version="1.0" ?>

<!DOCTYPE reviews[
<!ELEMENT reviews (entry*)>
<!ATTLIST entry isbn #
REQUIRED>
<!ELEMENT entry  (title, price_total,review*,review2+)>
<!ELEMENT title  (#PCDATA)>
<!ELEMENT price  (#PCDATA)>
<!ELEMENT review  (#PCDATA)>
]>
```

Generating XML Documents from UML Instances

Through Example

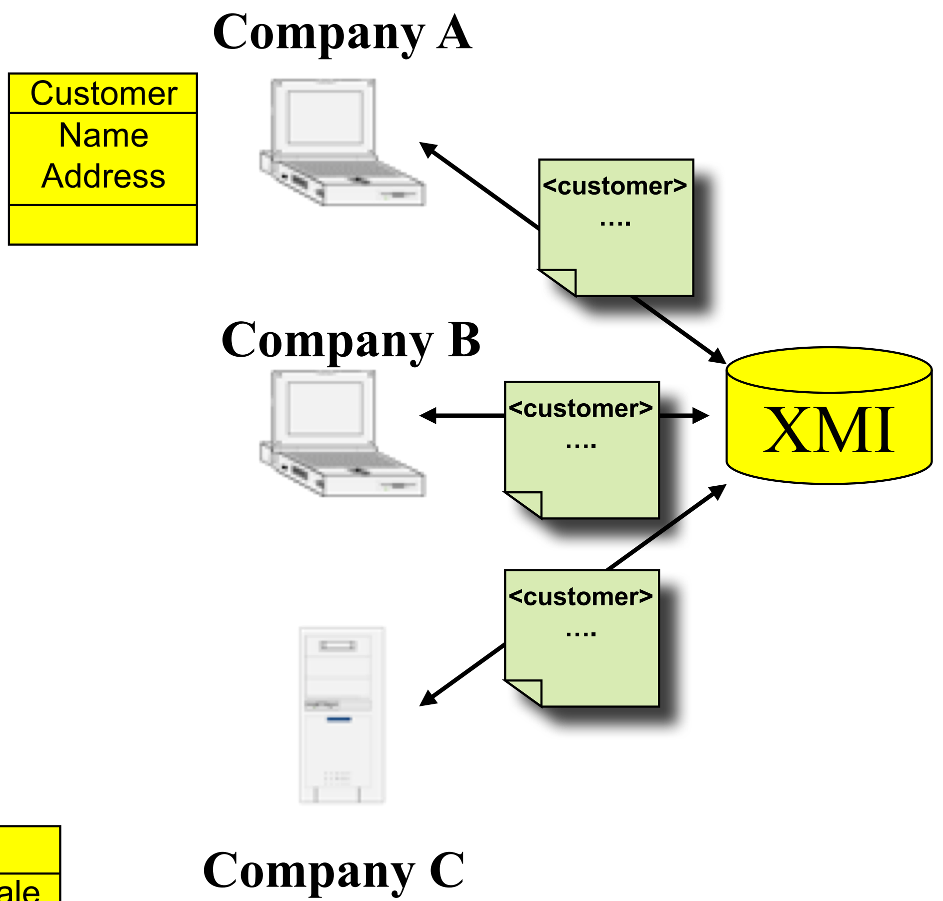
XML Data Interchange: XMI

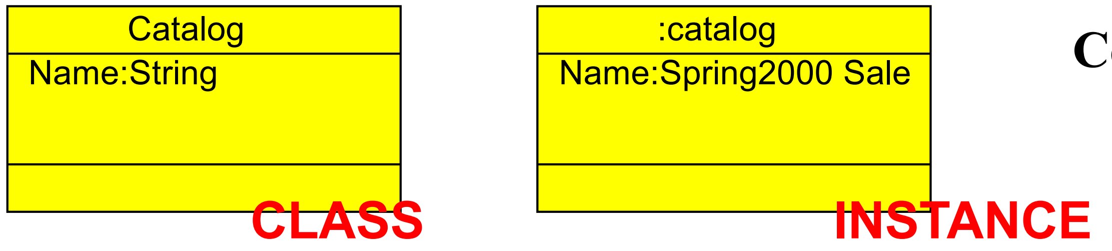
- Standard sponsored by the OMG
- Originally for allowing interchange of UML models between UML editors



XML Data Interchange: XMI

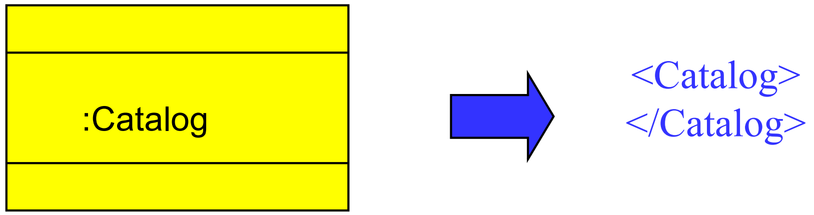
- Now seen as sensible XML representation of UML for other purposes
  - E.g. XML representation of entities specified using UML
- Want to generate
  - XML document instance from UML instance model
  - Validating Schema/DTD from corresponding UML class model





UML Class mapping

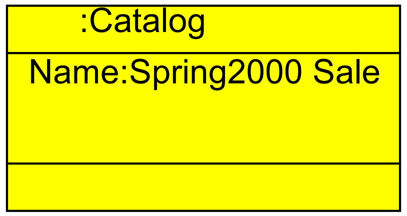
- Each instance of a UML class produces one XML element



- UML class name translates to XML tag name
- Be careful in naming your UML classes as XML has restrictions on valid tag names
  - Cannot contain spaces
  - Alpha or Number characters but also full stop, dash or underscore (. - _)
  - Can begin with letter or _
  - CANNOT begin with letters XML !!

UML Attribute mapping

- Each attribute of a UML class produces a child XML element
- Element name is made unique by prepending with the class name

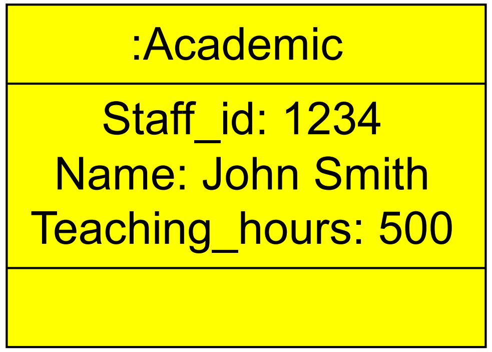


```
<Catalog>
   <Catalog.name>
      Spring 2000 Sale
   </Catalog.name>
</Catalog>
```

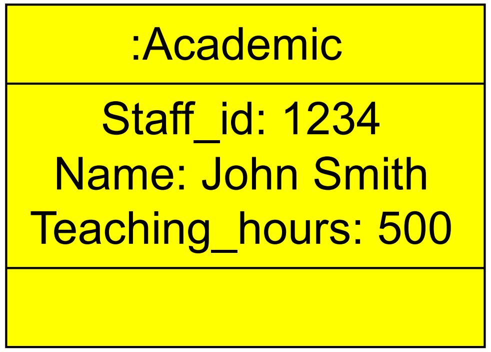
- XML has no representation for multivalued attributes of UML so these attributes are translated into individual XML elements
  - E.g. keyword[0..*]:String

```
<Catalogitem.keyword> Personal Computer </Catalogitem.keyword>
<Catalogitem.keyword> Windows 2000 </Catalogitem.keyword>
<Catalogitem.keyword> Notebook </Catalogitem.keyword>
```

Over to you…



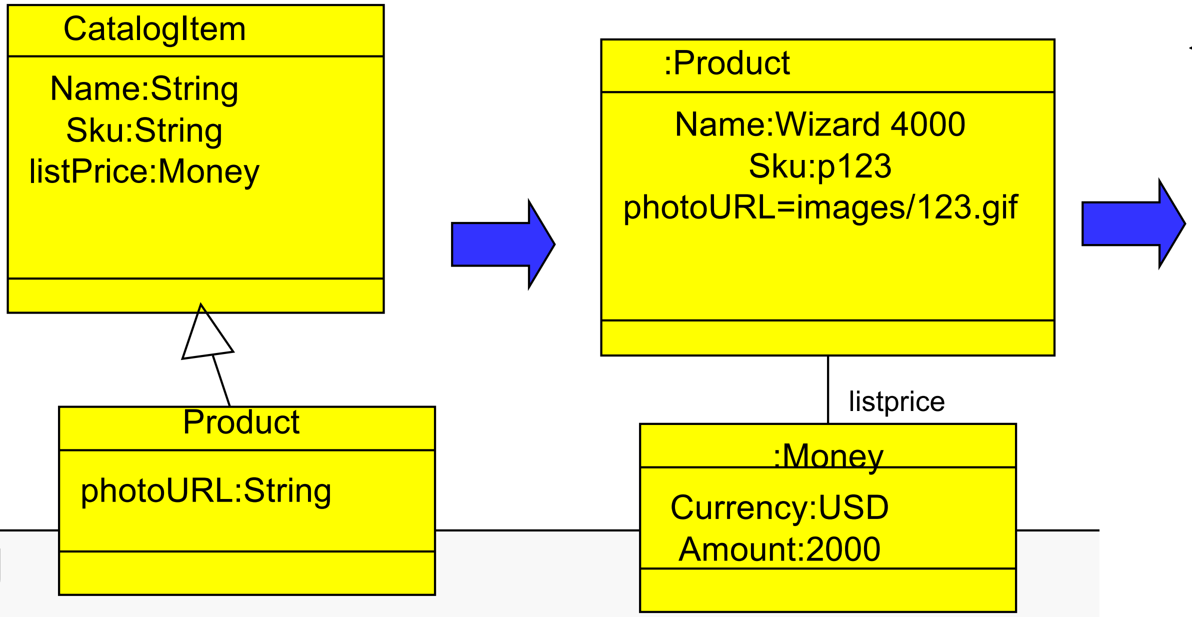
Over to you…



```
<Academic>
   <Academic.staff_id> 1234 </Academic.staff_id>
   <Academic.name> John Smith </Academic.name>
   <Academic.teaching_hours> 500 </Academic.teaching_hours>
</Academic>
```

UML Inheritance mapping

- Current XML standards do not have built in mechanism for representation of inheritance
- The 'XMI standard' specifies use of "copy down" approach for generalisations, attributes, association refs and compositions
  - That is definitions from all superclasses are copied down to the class being translated into XML

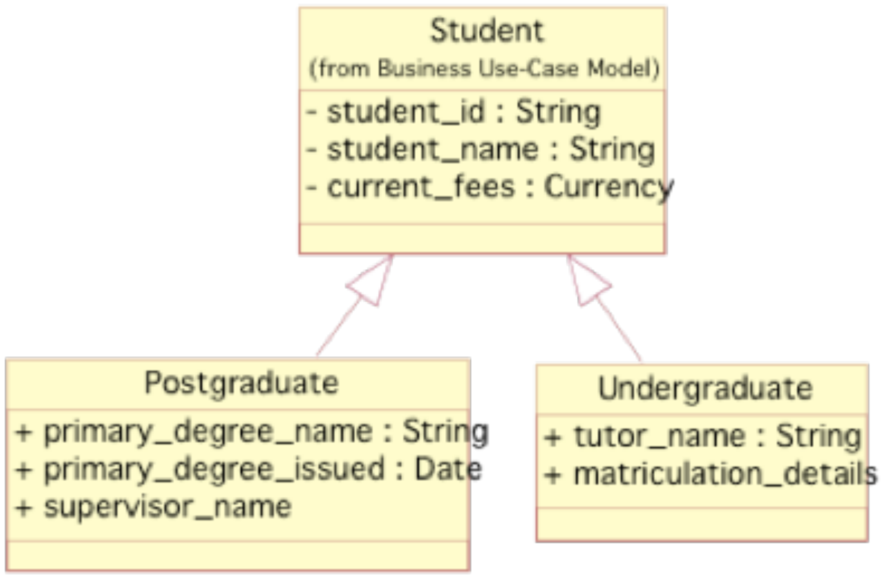


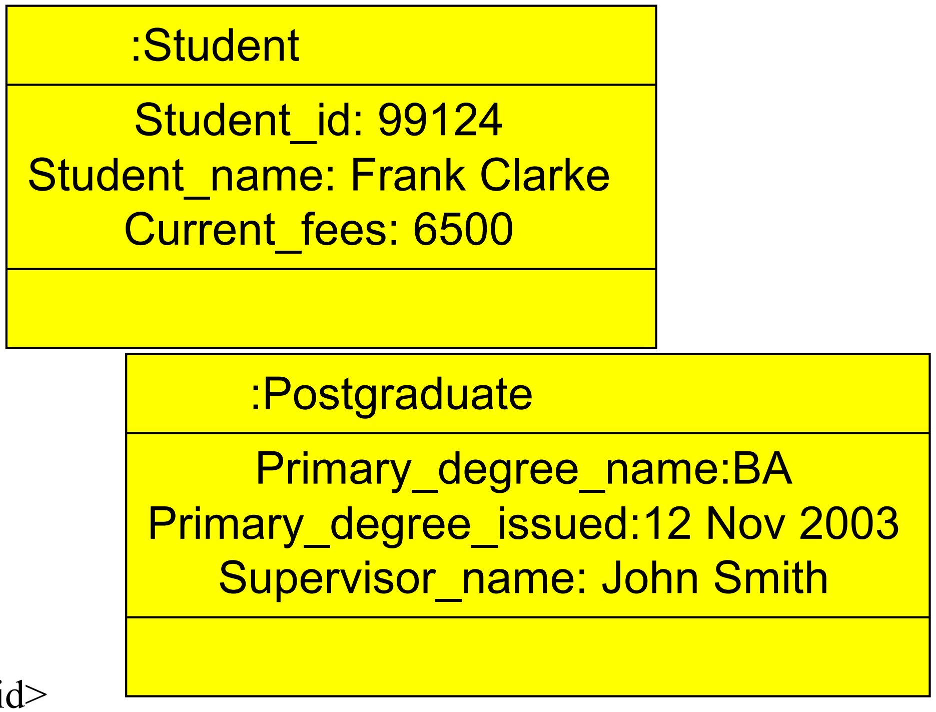
Note: element name due to copy down from superclass CatalogItem

Money is a class so its attributes copied in

```
<Product>
    <CatalogItem.name>
        Wizard 4000
    </CatalogItem.name>
    <CatalogItem.sku>  p123
    </CatalogItem.sku>
     <CatalogItem.listPrice>
          <Money>
          <Money.currency>USD</Money.currency>
          <Money.amount>2000</Money.amount>
           </Money> </CatalogItem.listPrice>
     <Product.photoURL> images/123.gif
      </Product.photoURL>
  </Product>
```

Over to you…

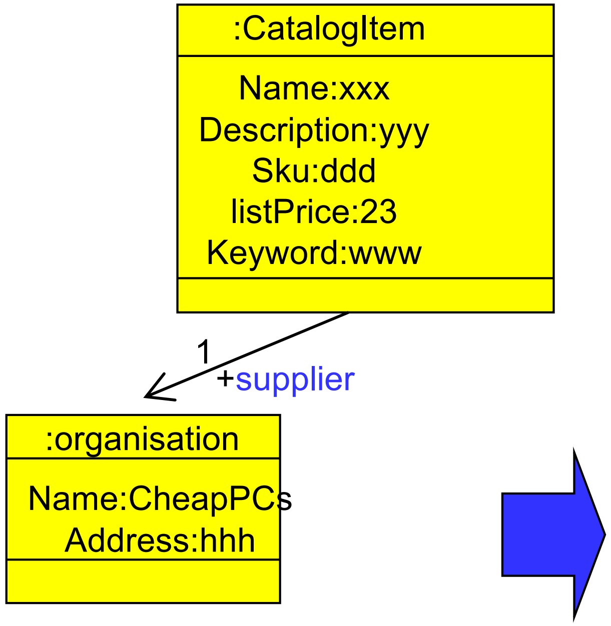



:Student
Student_id: 99124
Student_name: Frank Clarke
Current_fees: 6500
:Postgraduate
Primary_degree_name:BA
Primary_degree_issued:12 Nov 2003
Supervisor_name: John Smith

```
<Postgraduate>
   <Student.student_id> 99124 </Student.student_id>
   <Student.student_name> Frank Clarke </Student.student_name>
   <Student.current_fees> 6500 </Student.current_fees>
   <Postgraduate.primary_degree_name> BA </Postgraduate.primary_degree_name>
   <Postgraduate.primary_degree_issued > 12 November 2003 </Postgraduate.primary_degree_issued>
   <Postgraduate.supervisor_name > John Smith </Postgraduate.supervisor_name>
</Postgraduate>
```

UML Associations
Simple approach

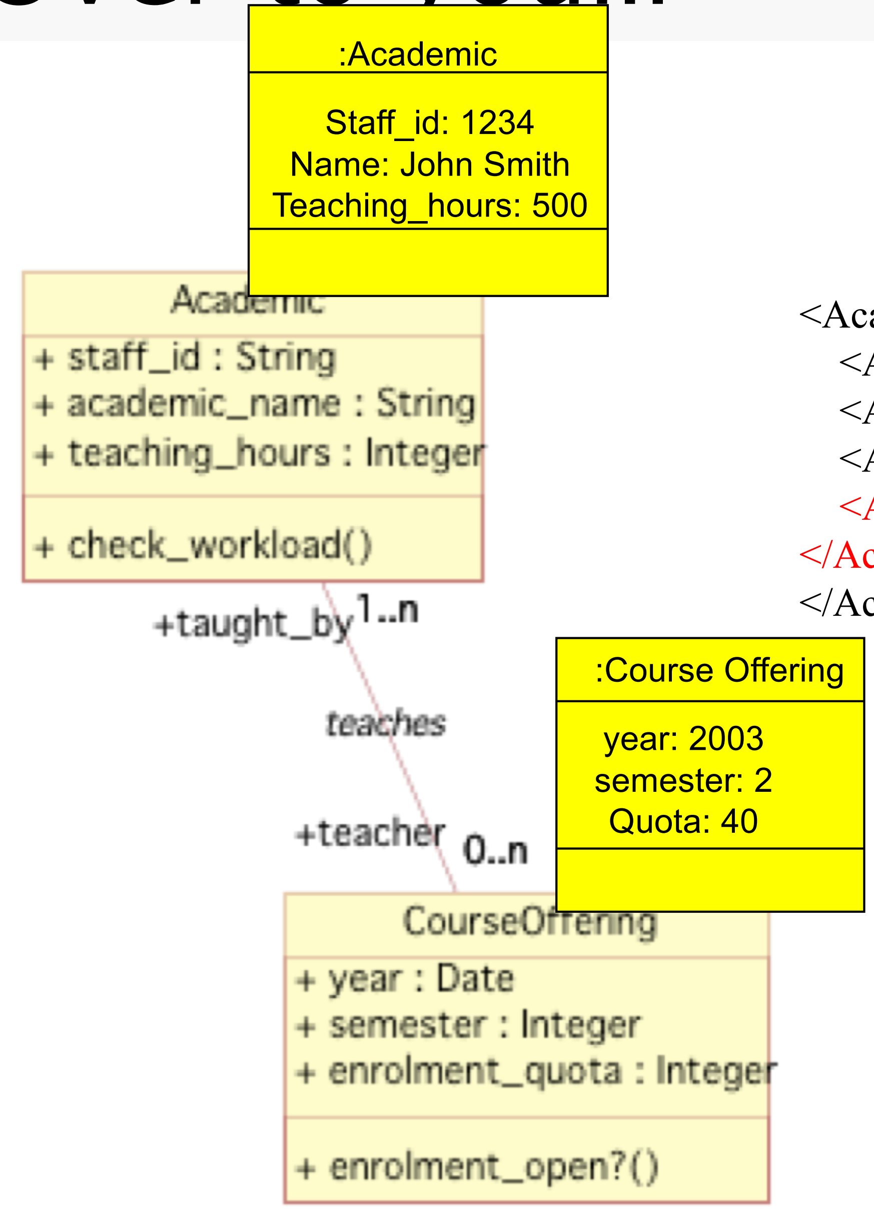
- A reference to the class of the associated class is included in the definition using the xmi.idref attribute
  - xmi.id then used to label definition of class



Note: association takes form of attribute with **role name** as name

```
<CatalogItem.supplier>
        <Organisation xmi.idref="s1"/>
 <CatalogItem.supplier>

<Organisation xmi.id="s1">

    <Organisation.name> Cheap PCs

    </Organisation.name>

</Organisaion>
```

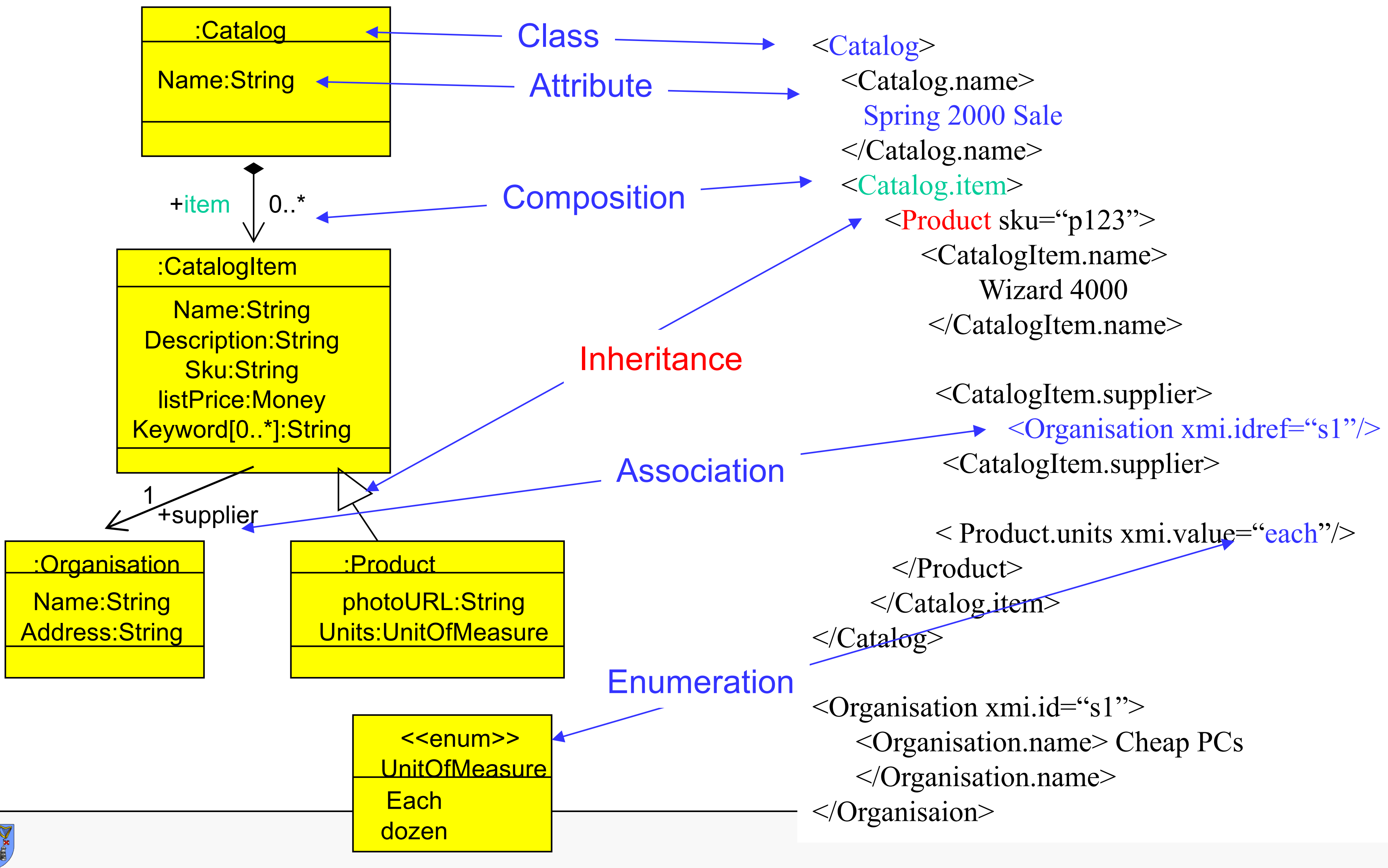
Over to you…



```
<Academic xmi.id="22">
   <Academic.staff_id> 1234 </Academic.staff_id>
   <Academic.name> John Smith </Academic.name>
   <Academic.teaching_hours> 500 </Academic.teaching_hours>
   <Academic.teacher> <CourseOffering xmi.idref="4ba5"/>
</Academic.teacher>
</Academic>
```

```
<CourseOffering xmi.id="4ba5">
   <CourseOffering.year> 2003 </CourseOffering.year>
   <CourseOffering.semester> 2 </CourseOffering.semester>
   <CourseOffering.enrolment_quota> 40 </CourseOffering.enrolment_quota
   <CourseOffering.taught_by> <Academic xmi.idref="22"/>
</CourseOffering.taught_by>
</CourseOffering>
```

Summary Example



```
<Catalog>
  <Catalog.name>
    Spring 2000 Sale
  </Catalog.name>
  <Catalog.item>
    <Product sku=“p123”>
      <CatalogItem.name>
          Wizard 4000
       </CatalogItem.name>

        <CatalogItem.supplier>
            <Organisation xmi.idref=“s1”/>
         <CatalogItem.supplier>

        < Product.units xmi.value=“each”/>
     </Product>
   </Catalog.item>
</Catalog>

<Organisation xmi.id=“s1”>
    <Organisation.name> Cheap PCs
    </Organisation.name>
</Organisaion>
```

Reminder - Part 1: Deliverables

1. Follow the detailed spec for UML design at

https://www.scss.tcd.ie/CourseModules/CS2041/materials/slides/GroupProjectSpec17.pdf

1. A printed hard-copy report for the UML design from the group including:
   - Introduction to system, your background research, how you went about researching the domain and how you went about undertaking the task
   - UML use case diagrams and detailed scenario descriptions
   - UML Class diagram and description of design decisions made
   - UML activity diagrams and description
   - Ethics Canvas and description
   - Listing of who did what
   - Discussion of Strengths and Weaknesses of the overall UML Design
   - ***** ALL GROUPS To sign in report at LAB session 10am on Monday 13th November 2017*****

Reminder:

Prepare next Group 5 minute presentation to present Thursday 16th November

- UML Use Case diagram (as reference)
- 2 Activity Diagrams (1 per selected use case/oval)
- Ethics Canvas
- strengths & weaknesses of design

Deadline for presentation material

- Email by Wednesday 15th November 5pm
- PDF version to be of presented
- PDF version that includes presentation with speaker notes
- **You MUST Include Group Number in Subject Line of Email "CS2041 Group XXX:.... "**

XML NAMESPACES & XML SCHEMA

What are XML Namespaces?

- W3C recommendation (January 1999)
- Each XML vocabulary is considered to own a namespace in which all elements (and attributes) are unique
- A single document can use elements and attributes from multiple namespaces
  - A prefix is declared for each namespace used within a document.
  - The namespace is identified using a URI (Uniform Resource Identifier)
- An element or attribute can be associated with a namespace by placing the namespace prefix before its name (i.e. '*prefix*:*name*')
  - Elements (and attributes) belonging to the default namespace do not require a prefix

Example: XML Namespaces

St. James's Hospital

```
<!ELEMENT Patient (Name, DOB)>

<!ELEMENT Name   (First, Last)>
<!ELEMENT First  (#PCDATA)>
<!ELEMENT Last   (#PCDATA)>
<!ELEMENT DOB    (#PCDATA)>
```

Airport Pharmacy

```
<!ELEMENT Drug
     ((Name|Substance), Code)>

<!ELEMENT Name      (#PCDATA)>
<!ELEMENT Substance (#PCDATA)>
<!ELEMENT Code      (#PCDATA)>
```

```
<?xml version='1.0'?>

<Accident Report
  xmlns:sjh="http://hospital/sjh"
  xmlns:dub=http://airport/dub >

<sjh:Patient>
 <sjh:Name>
  <sjh:First>Mike</sjh:First>
  <sjh:Last>Murphy</sjh:Last>
 </sjh:Name>
 <sjh:DOB>12/12/1950</sjh:DOB>
</sjh:Patient>

<dub:Drug>
 <dub:Name>Nurofen</dub:Name>
 <dub:Code>IE-975-2</dub:Code>
</dub:Drug>

  [...]
</Accident Report>
```

What are XML Schemas?

- W3C Recommendation, 2 May 2001
  - Part 0: Primer
  - Part 1: Structures
  - Part 2: Datatypes
- DTDs use a non-XML syntax and have a number of limitations
  - no namespace support
  - lack of data-types
- XML Schemas are an alternative to DTDs
- Used to formally specify a "class" of XML documents (↔ "instance document")
- Supports simple/complex data-types

Why use XML Schemas?

- Uses an XML syntax
- Supports simple and complex data-types such as user-defined types
- An XML document and its contents can be validated against a Schema
- Can validate documents containing multiple namespaces
- Schemas are more powerful than DTDs and will eventually replace DTDs

Named Types – simple

DTD

```
<!ELEMENT birthday(#PCDATA)>
```

XML Schema

```
<xsd:element name=”birthday" type="xsd:date"/>
```

XML doc. Instance

```
<birthday>01 March 2001</birthday>
```

Named Types – complex

DTD

```
<!ELEMENT student_name (firstname, lastname)>
```

XML Schema

```
<xsd:complexType name="namePerson">
   <xsd:sequence>
     <xsd:element name="firstname" type="xsd:string"/>
     <xsd:element name="lastname" type="xsd:string/>
   </xsd:sequence>
</xsd:complexType>
<xsd:element name="student_name" type="namePerson"/>
```

XML doc. Instance

```
<student_name>
   <firstname>Michael</firstname>
   <lastname>Porter</lastname>
</student_name>
```

Primitive Datatypes

- string
- boolean
- decimal
- float
- double
- duration
- dateTime
- time
- date
- gYearMonth
- gYear
- gMonthDay
- gDay
- gMonth
- hexBinary
- base64Binary
- anyURI
- QName
- NOTATION

`http://www.w3.org/TR/xmlschema-2/`

Simple Type - Restriction

XML Schema

```
<simpleType name='celsiusBodyTemp'>
  <restriction base='decimal'>
    <totalDigits value='4'/>
    <fractionDigits value='1'/>
    <minInclusive value='36.4'/>
    <maxInclusive value='40.5'/>
  </restriction>
</simpleType>
<xsd:element name="temp" type="celsiusBodyTemp"/>
```

XML doc. Instance

```
<temp>37.2</temp>
```

Simple Type - Enumeration

XML Schema

```
<xsd:simpleType name="weekday">
   <xsd:restriction base="xsd:string">
     <xsd:enumeration value="Sunday"/>
     <xsd:enumeration value="Monday"/>
     <xsd:enumeration value="Tuesday"/>
     [...]
   </xsd:restriction>
</xsd:simpleType>
<xsd:element name="delivery" type="weekday"/>
```

XML doc. Instance

```
<delivery>Tuesday</delivery>
```

Complex Type - Cardinalities

DTD

```
<!ENTITY % fullname "title?, firstname*, lastname">
<!ELEMENT student_name (%fullname;)>
```

XML Schema

```
<xsd:complexType name="fullname">
  <xsd:sequence>
    <xsd:element name="title" minOccurs="0"/>
    <xsd:element name="firstname" minOccurs="0"
                 maxOccurs="unbounded"/>
    <xsd:element name="lastname"/>
  </xsd:sequence>
</xsd:complexType>
<xsd:element name="student_name" type="fullname"/>
```

XML doc. Instance

```
<student_name>
  <firstname>Michael</firstname>
  <firstname>Jason</firstname>
  <lastname>Porter</lastname>
</student_name>
```

Complex Type – Derived Type by extension

DTD

```
<!ENTITY % name "title?, firstname*, lastname">
<!ELEMENT student_name (%name;, maidenname?)>
```

XML Schema

```
<xsd:complexType name="fullnameExt">
   <xsd:complexContent>
     <xsd:extension base="fullname">
       <xsd:sequence>
         <xsd:element name="maidenname" minOccurs="0"/>
       </xsd:sequence>
     </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
<xsd:element name=”student_name" type="fullnameExt"/>
```

XML doc. Instance

```
<student_name>
   <firstname>Jane</firstname>
   <lastname>Porter</lastname>
   <maidenname>Hughes</maidenname>
</student_name>
```

Complex Type – Derived Type by Restriction

XML Schema

```
<xsd:complexType name="simpleName">
   <xsd:complexContent>
     <xsd:restriction base="fullname">
       <xsd:sequence>
         <xsd:element name="title" maxOccurs="0"/>
         <xsd:element name="firstname" minOccurs="1"/>
         <xsd:element name="lastname"/>
       </xsd:sequence>
     </xsd:restriction>
   </xsd:complexContent>
</xsd:complexType>
```

XML doc. Instance

```
<name>
   <firstname>Jane</firstname>
   <lastname>Porter</lastname>
</name>
```

Structure - Sequence

DTD

```
<!ELEMENT student_name (title?, firstname*, lastname)>
```

XML Schema

```
<xsd:complexType name="fullname">
  <xsd:sequence>
    <xsd:element name="title" minOccurs="0"/>
    <xsd:element name="firstname" minOccurs="0"
                 maxOccurs="unbounded"/>
    <xsd:element name="lastname"/>
  </xsd:sequence>
</xsd:complexType>
<xsd:element name="student_name" type="fullname"/>
```

XML doc. Instance

```
<student_name>
  <firstname>Michael</firstname>
  <firstname>Jason</firstname>
  <lastname>Porter</lastname>
</student_name>
```

Structure - Choice

DTD

```
<!ELEMENT pay (product, number, (cash | cheque))>
```

XML Schema

```
<xsd:complexType name="payment">
   <xsd:sequence>
     <xsd:element ref="product"/>
     <xsd:element ref="number"/>
     <xsd:choice>
       <xsd:element ref="cash"/>
       <xsd:element ref="cheque"/>
     </xsd:choice>
   </xsd:sequence>
</xsd:complexType>
<xsd:element name="pay" type="payment"/>
```

XML doc. Inst.

```
<pay>
   <product>Ericsson Telefon MD110</product>
   <number>1544-198-J</number>
   <cash>EUR150</cash>
</pay>
```

Attributes

DTD

```
<!ELEMENT greeting (#PCDATA)>
<!ATTLIST greeting language CDATA "English">
```

XML Schema

```
<xsd:element name="greeting">
   <xsd:complexType>
     <xsd:simpleContent>
       <xsd:extension base="xsd:string">
         <xsd:attribute name="language" type="xsd:string"/>
       </xsd:extension>
     </xsd:simpleContent>
   </xsd:complexType>
</xsd:element>
```

XML doc. Instance

```
<greeting language="German">Hello!</greeting>
```

Attribute Groups

DTD

```
<!ELEMENT img EMPTY>
<!ATTLIST img src CDATA #REQUIRED
          width CDATA #IMPLIED
          height CDATA #IMPLIED>
```

XML Schema

```
<xsd:attributeGroup name="imgAttributes">
   <xsd:attribute name="src" type="xsd:string" use="required"/>
   <xsd:attribute name="width" type="xsd:integer"/>
   <xsd:attribute name="height" type="xsd:integer"/>
</xsd:attributeGroup>


<xsd:element name="img">
   <xsd:complexType>
     <xsd:attributeGroup ref="imgAttributes"/>
   <xsd:complexType>
</xsd:element>
```

XML Inst.

```
<img src="XMLmanager.gif" width="60"/>
```

Mixed Content

DTD

```
<!ELEMENT p (#PCDATA | b | i)*>
<!ELEMENT b (#PCDATA)>
```

XML Schema

```
<xsd:complexType name="bolditalicText" mixed="true">
   <xsd:choice minOccurs="0" maxOccurs="unbounded"/>
     <xsd:element ref="b" />
     <xsd:element ref="i" />
   </xsd:choice>
</xsd:complexType>


<xsd:element name="p" type="bolditalicText"/>
```

XML doc. Instance

```
<p>This is <b>bold</b> and <i>italic</i> text</p>
```

Empty Element

DTD

```
<!ELEMENT img EMPTY>
<!ATTLIST src CDATA #REQUIRED>
```

XML Schema

```
<xsd:element name="img">
   <xsd:complexType>
     <xsd:attribute name="src" type="xsd:string"/>
   </xsd:complexType>
</xsd:element>
```

XML doc. Instance

```
<img src="XMLmanager.gif"/>
```

XML Schema Example

```
<?xml version="1.0" encoding="utf-8"?>

<xsd:schema xmlns:xsd="http://www.w3.org/2000/10/XMLSchema">
  <xsd:element name="book">

    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="title" type="xsd:string"/>
        <xsd:element name="author" type="xsd:string"/>
        <xsd:element name="character” type="xsd:string"
                     minOccurs="0" maxOccurs="unbounded">
        </xsd:element>
      </xsd:sequence>

      <xsd:attribute name="isbn" type="xsd:string"/>
    </xsd:complexType>

  </xsd:element>
</xsd:schema>
```

Create an XSD for

```
<?xml version="1.0"?>
<purchaseOrder xmlns="http://tempuri.org/po.xsd" orderDate="1999-10-20">
   <shipTo country="US">
      <name>Alice Smith</name> <street>123 Maple Street</street>
      <city>Mill Valley</city> <state>CA</state> <zip>90952</zip>
   </shipTo>
   <billTo country="US">
      <name>Robert Smith</name><street>8 Oak Avenue</street>
      <city>Old Town</city> <state>PA</state> <zip>95819</zip>
   </billTo>
   <comment>Hurry, my lawn is going wild!</comment>
<items> <item partNum="872-AA">
         <productName>Lawnmower</productName>
         <quantity>1</quantity> <USPrice>148.95</USPrice>
         <comment>Confirm this is electric</comment>
      </item>
      <item partNum="926-AA">
         <productName>Baby Monitor</productName>
         <quantity>1</quantity><USPrice>39.98</USPrice>
         <shipDate>1999-05-21</shipDate>
      </item>
   </items> </purchaseOrder>
```

Possible Solution

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" targetNamespace="http://tempuri.org/po.xsd"
xmlns="http://tempuri.org/po.xsd" elementFormDefault="qualified">
 <xs:annotation>
  <xs:documentation xml:lang="en">
   Purchase order schema for Example.com.
   Copyright 2000 Example.com. All rights reserved.
  </xs:documentation>
 </xs:annotation>

 <xs:element name="purchaseOrder" type="PurchaseOrderType"/>

 <xs:element name="comment" type="xs:string"/>

 <xs:complexType name="PurchaseOrderType">
  <xs:sequence>
   <xs:element name="shipTo" type="USAddress"/>
   <xs:element name="billTo" type="USAddress"/>
   <xs:element ref="comment" minOccurs="0"/>
   <xs:element name="items"  type="Items"/>
  </xs:sequence>
  <xs:attribute name="orderDate" type="xs:date"/>
 </xs:complexType>
```

```
<xs:complexType name="USAddress">
      <xs:annotation>
      <xs:documentation>
       Purchase order schema for Example.Microsoft.com.
       Copyright 2001 Example.Microsoft.com. All rights
reserved.
      </xs:documentation>
      <xs:appinfo>
        Application info.
      </xs:appinfo>
     </xs:annotation>

  <xs:sequence>
   <xs:element name="name"   type="xs:string"/>
   <xs:element name="street" type="xs:string"/>
   <xs:element name="city"   type="xs:string"/>
   <xs:element name="state"  type="xs:string"/>
   <xs:element name="zip"    type="xs:decimal"/>
  </xs:sequence>
  <xs:attribute name="country" type="xs:NMTOKEN"
    fixed="US"/>
 </xs:complexType>
```

```
<xs:complexType name="Items">
 <xs:sequence>
  <xs:element name="item" minOccurs="0" maxOccurs="unbounded">
   <xs:complexType>
    <xs:sequence>
     <xs:element name="productName" type="xs:string"/>
     <xs:element name="quantity">
      <xs:simpleType>
       <xs:restriction base="xs:positiveInteger">
        <xs:maxExclusive value="100"/>
       </xs:restriction>
      </xs:simpleType>
     </xs:element>
     <xs:element name="USPrice"    type="xs:decimal"/>
     <xs:element ref="comment"   minOccurs="0"/>
     <xs:element name="shipDate" type="xs:date" minOccurs="0"/>
    </xs:sequence>
    <xs:attribute name="partNum" type="SKU" use="required"/>
   </xs:complexType>
  </xs:element>
 </xs:sequence>
</xs:complexType>

<!-- Stock Keeping Unit, a code for identifying products -->
<xs:simpleType name="SKU">
 <xs:restriction base="xs:string">
  <xs:pattern value="\d{3}-[A-Z]{2}"/>
 </xs:restriction>
</xs:simpleType>

</xs:schema>
```

Summary

- XML Vocabularies are defined using
  - DTD
  - XSD
- DTDs/XSDs used to validate XML documents
- XSD – more powerful than DTDs
  - Supports simple and complex data-types such as user-defined types
  - Can validate documents containing multiple namespaces