Namespaces and XML Schema

References

- XML Schema Part 0: Primer Second Edition (http://www.w3.org/TR/xmlschema-0/)
- Semantic Web Primer (Chapter 2)

Ideas

- XML documents – no precise semantics ➔ not *machine-readable*
- DTD – the first attempt to *constraint,* to *provide* meanings and relationships between components of a XML documents – but *with many limitations* (next slide, previous class)

Inadequacy of DTD as a Data Definition Language

- Designed for a single document (combination of dtd?)
- Syntax is very different than that of XML
- Limited basic types
- Limited means for expressing data consistency constrains
- Enforcing referential integrity for attributes but not elements
- XML data is ordered; not database data
- Element definitions are global to the entire document

Goals

- XML schema language: to define, describe and catalogue XML vocabularies for classes of XML documents
- A schema: to define and describe a class of XML documents (constraints, meaning, relationships, …)

What's good about schema?

- Documents (contents) conform to a schema can be easily
  - exchanged between agents
  - discovered by agents
  - advertised by agents

  who understand the schema
- Richer language comparing to DTD and …

XML Schema – Main Features

- Same syntax as XML
- Integration with the namespace mechanism (different schemas can be imported from different namespaces and integrated into one)
- Built-in types (similar to SQL)
- Mechanism for defining complex types from simple types
- Support keys and referential integrity constraints
- Better mechanism for specifying documents where the order of element types does not matter

XML Schema and Instance

Example

The Purchase Order, po.xml

Instance: A normal XML document
- root element: purchaseOrder
- elements: shipTo, billTo name, items, item, …

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

XML schema for the PO.XML document

The Purchase Order Schema, po.xsd

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

<xsd:element name="purchaseOrder" type="PurchaseOrderType"/>
<xsd:element name="comment" type="xsd:string"/>

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

The "US Address" type

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

The "Items" type

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

The "SKU" type

```
<!-- Stock Keeping Unit, a code for identifying products -->

<xsd:simpleType name="SKU">
        <xsd:restriction base="xsd:string">
                <xsd:pattern value="\d{3}-[A-Z]{2}"/>
        </xsd:restriction>
</xsd:simpleType>

</xsd:schema>
```

in PO.XSD

- defines types
  - complex type (e.g., items)
  - built-in types (e.g., integer, string, date)
- defines constraints
  - pattern
  - number of occurrences
- references to something like
  - xsd:schema
  - xsd:element

NAMESPACE

Namespaces

- Reusing of vocabularies from multiple sources
- For avoiding naming conflicts
- Name of every XML tag must have two parts:
  - **namespace**: a string in the form of a **uniform resource identifier** (**URI**) or a **uniform resource locator** (**URL**)
  - **local name**: as regular XML tag but cannot contain ':'
- Structure of an XML tag:

namespace:local_name

Namespaces

- An XML namespace is a collection of names, identified by a URI reference, which are used in XML documents as element types and attribute names. XML namespaces differ from the "namespaces" conventionally used in computing disciplines in that the XML version has internal structure and is not, mathematically speaking, a set.

Source: www.w3c.org

Uniform Resource Identifier

- URI references which identify namespaces are considered identical when they are exactly the same character-for-character. Note that URI references which are not identical in this sense may in fact be functionally equivalent. Examples include URI references which differ only in case, or which are in external entities which have different effective base URIs.

Source: www.w3c.org

Namespace declaration

- Defined by

  xml : prefix = declaration
- Tags belonging to a namespace should be prefixed with "*prefix:*"
- Tags belonging to the default namespace do not need to have the prefix
- Have its own scope

XML Document and Schema

A document conforms to a schema is called an *instance* of this schema and is said to be *schema valid*.

XML processor does not check for schema validity

XML Schema and Namespaces

- Describes the structure of other XML documents
- Begins with a declaration of the namespaces to be used in the schema, including
  - http://www.w3.org/2001/XMLSchema
  - http://www.w3.org/2001/XMLSchema-instance
  - **targetnamespace** (user-defined namespace)

Target namespace

- identifies the set of names defined by a particular schema document
- is an attribute of the *schema* element (**targetNamespace**) whose value is the name space containing all the names defines by the schema *which is usually an URI and can be referred/reused by other schema*

Purchase Order Schema with Target Namespace, po1.xsd

```
<schema xmlns="http://www.w3.org/2001/XMLSchema"        → Default namespace
        xmlns:po="http://www.example.com/PO1"
        targetNamespace="http://www.example.com/PO1"
        elementFormDefault="unqualified"    → Instances do not need to have the prefixes
        attributeFormDefault="unqualified">
<element name="purchaseOrder" type="po:PurchaseOrderType"/>
<element name="comment" type="string"/>
<complexType name="PurchaseOrderType">
        <sequence>
                <element name="shipTo" type="po:USAddress"/>
                <!-- etc. -->
        </sequence>
        <!-- etc. -->
</complexType>
<complexType name="USAddress">
        <sequence>
                <element name="name" type="string"/>
                <element name="street" type="string"/>
                <!-- etc. -->
        </sequence>
</complexType>
<!-- etc. -->
</schema>
```

Unqualified elements: defined in the default namespace

Qualified elements: defined in the namespace specified by the value of the prefix

A Purchase Order with Unqualified Locals, po1.xml

```
<?xml version="1.0"?>
<apo:purchaseOrder xmlns:apo="http://www.example.com/PO1"
        orderDate="1999-10-20">
<shipTo country="US">
        <name>Alice Smith</name> <street>123 Maple Street</street>
        <!-- etc. -->
</shipTo>
<billTo country="US">
        <name>Robert Smith</name> <street>8 Oak Avenue</street>
        <!-- etc. -->
</billTo>
<apo:comment>Hurry, my lawn is going wild<!/apo:comment>
        <!-- etc. -->
</apo:purchaseOrder>
```

Purchase Order Schema with Target Namespace, po2.xsd

```
<schema xmlns="http://www.w3.org/2001/XMLSchema"        → Default namespace
        xmlns:po="http://www.example.com/PO2"
        targetNamespace="http://www.example.com/PO2"
        elementFormDefault="qualified"      → Instances do need to have the prefixes
        attributeFormDefault="unqualified">
<element name="purchaseOrder" type="po:PurchaseOrderType"/>
<element name="comment" type="string"/>
<complexType name="PurchaseOrderType">
        <sequence>
                <element name="shipTo" type="po:USAddress"/>
                <!-- etc. -->
        </sequence>
        <!-- etc. -->
</complexType>
<complexType name="USAddress">
        <sequence>
                <element name="name" type="string"/>
                <element name="street" type="string"/>
                <!-- etc. -->
        </sequence>
</complexType>
<!-- etc. -->
</schema>
```

A Purchase Order with Explicitly Qualified Locals, po2.xml

```
<?xml version="1.0"?>
<apo:purchaseOrder xmlns:apo="http://www.example.com/PO2"
        orderDate="1999-10-20">
<apo:shipTo country="US">
        <apo:name>Alice Smith</apo:name>
        <apo:street>123 Maple Street</apo:street>
        <!-- etc. -->
</apo:shipTo>
<apo:billTo country="US">
        <apo:name>Robert Smith</apo:name>
        <apo:street>8 Oak Avenue</apo:street>
        <!-- etc. -->
</apo:billTo>
<apo:comment>Hurry, my lawn is going wild<!/apo:comment>
        <!-- etc. -->
</apo:purchaseOrder>
```

Undeclared target namespace

- Elements and attributes from other namespaces should have prefix (see po.xsd, page 11, where the prefix *xsd* is used)
- Validate unqualified elements in the document instances (see po.xml, page 9, e.g. *purchaseOrder*, *items*)

Schema on multiple documents

The International Purchase Order Schema, ipo.xsd

```
<schema targetNamespace="http://www.example.com/IPO"
        xmlns="http://www.w3.org/2001/XMLSchema"
        xmlns:ipo="http://www.example.com/IPO">

<annotation>
        <documentation xml:lang="en">
            International Purchase order schema for Example.com
            Copyright 2000 Example.com. All rights reserved.
        </documentation>
 </annotation>

<!-- include address constructs -->
<include schemaLocation="http://www.example.com/schemas/address.xsd"/>

<element name="purchaseOrder" type="ipo:PurchaseOrderType"/>
<element name="comment" type="string"/>
```

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

Address: not defined in this document but in "address.xsd" – see "include statement"

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

```
  <simpleType name="SKU">
    <restriction base="string">
      <pattern value="\d{3}-[A-Z]{2}"/>
    </restriction>
  </simpleType>

</schema>
```

File address.xsd

```
<schema targetNamespace="http://www.example.com/IPO"
        xmlns="http://www.w3.org/2001/XMLSchema"
        xmlns:ipo="http://www.example.com/IPO">

  <annotation>
    <documentation xml:lang="en">
      Addresses for International Purchase order schema
      Copyright 2000 Example.com. All rights reserved.
    </documentation>
  </annotation>

  <complexType name="Address">
    <sequence>
      <element name="name"   type="string"/>
      <element name="street" type="string"/>
      <element name="city"   type="string"/>
    </sequence>
  </complexType>
  etc.
</schema>
```

same

Target namespace of the including schema (page 29)

Types

- Simple types
  - Primitive
  - Deriving simple types
- Complex types

Simple Types (Built-In)

- Most of the commonly used types (string, integer, int, long,decimal, float, …)

  Table 2 in http://www.w3.org/TR/xmlschema-0/

Simple Type Constructors (1)

- Pattern (Restriction)

Name of Type

```
<xsd:simpleType name="SKU">
    <xsd:restriction base="xsd:string">
    <xsd:pattern value="\d{3}-[A-Z]{2}"/>
    </xsd:restriction>
</xsd:simpleType>
```

Simple Type Constructors (2)

- Pattern (Enumeration)

Name of Type

```
<xsd:simpleType name="USState">
    <xsd:restriction base="xsd:string">
        <xsd:enumeration value="AK"/>
        <xsd:enumeration value="AL"/>
        <xsd:enumeration value="AR"/>
        <!-- and so on ... -->
    </xsd:restriction>
</xsd:simpleType>
```

Simple Type Constructors (3)

- Range

Name of Type

```
<xsd:simpleType name="myInteger">
    <xsd:restriction base="xsd:integer">
        <xsd:minInclusive value="10000"/>
        <xsd:maxInclusive value="99999"/>
    </xsd:restriction>
</xsd:simpleType>
```

List Type (1)

Name of Type

```
<xsd:simpleType name="listOfMyIntType">
  <xsd:list itemType="myInteger"/>
</xsd:simpleType>
```

instance document can contain something like the following

```
<listOfMyInt>20003 15037 95977 95945</listOfMyInt>
```

List Type (2)

Name of Type

```
<xsd:simpleType name="USStateList">
  <xsd:list itemType="USState"/>
</xsd:simpleType>
<xsd:simpleType name="SixUSStates">
  <xsd:restriction base="USStateList">
  <xsd:length value="6"/>
</xsd:restriction> </xsd:simpleType>
```

ONLY SIX STATES

Union Type

Name of Type

```
<xsd:simpleType name="zipUnion">
   <xsd:union
       memberTypes="USState listOfMyIntType"/>
</xsd:simpleType>
```

Name of sub Type

```
<zips>CA</zips>
<zips>95630 95977 95945</zips>
<zips>AK</zips>
```

Anonymous Type Definitions

- Items (page 13)

```
<xsd:complexType name="Items">
  <xsd:sequence>
      …
  </xsd:sequence>
< xsd:complexType>
```

Type Declaration for Elements &Attributes

- Type declaration for simple elements and attributes

<element name="CrsName" type="string"/>

Specify that CrsName has value of type *string*

Type Declaration for Elements &Attributes

- Type declaration for simple elements and attributes

<element name="status" type="adm:studentStatus"/>

Specify that status has value of type *studentStatus* that will be defined in the namespace referred to as *adm*

Complex Types

- Use to specify the type of elements with children or attributes
- Opening tag: *complexType*
- Can be associated to a name in the same way a simple type is associated to a name

Complex Types

- Special Case: element with simple content and some attributes/no child with some attributes (EMPTY COMPLEX TYPY)

```
<xsd:element name="internationalPrice">
  <xsd:complexType>
      <xsd:attribute name="currency" type="xsd:string"/>
      <xsd:attribute name="value" type="xsd:decimal"/>
  </xsd:complexType>
</xsd:element>
```

Complex Types

- Combining elements into group -- <all>

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

The four elements can appear in arbitrary order! (NOTE: <all> requires special care: top level, all elements, no group)

Complex Types

- Combining elements into group – <sequence>

```
<complexType name="NameType">
<sequence>
  <element name="First" type="string">
  <element name="Last" type="string">
</sequence>
</complexType>
```

The two elements must appear in order

Complex Types

- Combining elements into group – <choice>

```
<xsd:complexType name="PurchaseOrderType">
    <xsd:sequence>
          <xsd:choice>
            <xsd:group   ref="shipAndBill"/>
            <xsd:element name="singleUSAddress" type="USAddress"/>
          </xsd:choice>
          <xsd:element ref="comment" minOccurs="0"/>
          <xsd:element name="items"  type="Items"/>
    </xsd:sequence>
    <xsd:attribute name="orderDate" type="xsd:date"/>
</xsd:complexType>

<xsd:group id="shipAndBill">
 <xsd:sequence>
   <xsd:element name="shipTo" type="USAddress"/>
   <xsd:element name="billTo" type="USAddress"/>
 </xsd:sequence>
</xsd:group>
```

Complex Types

- Can also refer to local type like – allowing different elements to have children with the same name (next slides)

[studentType – courseType] both have the "Name" element

[studentType – personNameType] both have the "Name" element

```
<complexType name="studentType">
   <sequence>
        <element name="Name" type="…">
        <element name="Status" type="…">
         <element name="CrsTaken" type="…">
   </sequence>
   <attribute name="StudId" type="…">
</complexType>
<complexType name="courseType">
   <sequence>
        <element name="Name" type="…">
   </sequence>
   <attribute name="CrsCode" type="…">
</complexType>
```

Complex Types

- Importing schema: like *include* but does not require schemaLocation

instead of

```
<include
  schemaLocation="http://www.example.com/PO1"/>
```

we can use

```
<import namespace="http://www.example.com/PO1"/>
```

Complex Types

- Deriving new complex types by extension and restriction (for modifying imported schema)

```
….
<import namespace="http://www.example.com/PO1"/>
…..
<complexType name="">
<complexContent> <extension base="..">
        <element name="syllabus" type="string"/>
                </extension>
</complexContent></complexType>
```

The type that is going to be extended

Purchase Order with Order ID

```
….
<import namespace="http://www.example.com/PO1"/>
…..
<complexType name="poWithID">
   <complexContent>
   <extension base="po:PurchaseOrderType">
        <element name="orderNumber" type="myInteger"/>
   </extension>
</complexContent>
</complexType>
```

Additional element of an order

Report of purchase orders

- Use “poWithID” similar to “Item”
- Define “LPO” similar to “Items”
- Define additional schema and instance for inventory (consists of items with descriptions)

==> require that each order listed only once
==> require that SKU are valid

Integrity Constraints

- ID, IDREF, IDREFS can still be used
- Specified using the attribute xpath (next)
- XML keys, foreign keys
- Keys are associated with collection of objects not with types

Integrity Constraints - Keys

```
<key name="PrimaryKeyForLPO">
  <selector xpath="/purchaseOrder/orderNumber"/>
  <field    xpath="orderNumber"/>
</key>
```

Collection of elements which are associated with the key

The key comprises of one element (orderNumber) – which is a child of **poWithID**

Integrity Constraints - Foreign key

```
<keyref name="XXX" refer="PrimaryKeyItems">
  <selector xpath="Item"/>
  <field    xpath="@partNum"/>
</keyref>
```

Source Collection: where the elements should satisfy the key specified by the “PrimmaryKeyItems”

Complete Exampe (report.xsd)

The Report Schema, report.xsd

```
<schema targetNamespace="http://www.example.com/Report"
        xmlns="http://www.w3.org/2001/XMLSchema"
        xmlns:r="http://www.example.com/Report"
        xmlns:xipo="http://www.example.com/IPO"
        elementFormDefault="qualified">

        <!-- for SKU -->
        <import namespace="http://www.example.com/IPO"/>
                                        Import types defined in IPO.xsd
        <annotation>
          <documentation xml:lang="en">
              Report schema for Example.com Copyright 2000
              Example.com. All rights reserved.
          </documentation>
        </annotation>
```

```
<element name="purchaseReport">
    <complexType>
        <sequence>
            <element name="regions" type="r:RegionsType"/>
            <element name="parts" type="r:PartsType"/>
        </sequence>
        <attribute name="period" type="duration"/>
        <attribute name="periodEnding" type="date"/>
    </complexType>

    <unique name="dummy1">
        <selector xpath="r:regions/r:zip"/>
        <field xpath="@code"/>
    </unique>

    <key name="pNumKey">
        <selector xpath="r:parts/r:part"/>
        <field xpath="@number"/>
    </key>
    <keyref name="dummy2" refer="r:pNumKey">
        <selector xpath="r:regions/r:zip/r:part"/>
        <field xpath="@number"/>
    </keyref>
</element>
```

UNIQUE CONSTRAINT

r:zip defined in RegionsType

KEY parts - PartsType

FOREIGN KEY

```
<complexType name="RegionsType">
  <sequence>
   <element name="zip" maxOccurs="unbounded">
    <complexType>
      <sequence>
        <element name="part" maxOccurs="unbounded">
          <complexType>
            <complexContent>
              <restriction base="anyType">
              <attribute name="number" type="xipo:SKU"/>
              <attribute name="quantity" type="positiveInteger"/>
              </restriction>
            </complexContent>
          </complexType>
        </element>
      </sequence>
      <attribute name="code" type="positiveInteger"/>
    </complexType>
   </element>
  </sequence>
</complexType>
```

UNIQUE

FOREIGN KEY

```
<complexType name="PartsType">
    <sequence>
      <element name="part" maxOccurs="unbounded">
        <complexType>
          <simpleContent>
            <extension base="string">
              <attribute name="number" type="xipo:SKU"/>
            </extension>
          </simpleContent>
        </complexType>
      </element>
    </sequence>
</complexType>

</schema>
```

KEY

Homework 3

- Develop a XML schema for the document you have developed for homework 2