Intelligent Internet Technologies

Lectures 5-6.

XML Schema

Alexandra V. Vitko

KNURE, AI Department, alexandra_vitko@yahoo.com

Well formed XML vs. Valid XML

- **Well formed** XML document obeys all XML syntax rules.
- **Valid** XML obeys all XML syntax rules + corresponds to its structure definition (DTD or schema)

DTD and XML Schema

- **Document Type Definition (DTD)**:
  A DTD defines the elements that can appear in an XML document, the order in which they can appear, how they can be nested inside each other, and other basic details of XML document structure. DTDs are part of the original XML specification.

- **XML Schema**: A schema can define all of the document structures that you can put in a DTD, and it can also define data types and more complicated rules than a DTD can. The W3C developed the XML Schema specification.

XML Schemas vs. DTD

```
<xsd:element name="paper" type="papertype"/>
<xsd:complexType name="papertype">
    <xsd:sequence>
         <xsd:element name="title" type="xsd:string"/>
         <xsd:element name="author" minOccurs="0"/>
         <xsd:element name="year"/>
         <xsd:choice>  <xsd:element name="journal"/>
                       <xsd:element name="conference"/>
         </xsd:choice>
    </xsd:sequence>
</xsd:complexType>
```

DTD: <!ELEMENT paper (title,author*,year, (journal|conference))>

XML Schema

- Provides power of DTDs with addition of data types and constraints
- Schema is an XML format, and can be manipulated with XML tools
- Can specify numbers, dates, etc.
- Can specify minimum and maximum values
- Can specify string format, yyyy-mm-dd

XML Schema is

- A grammar definition language
  - Like DTDs but better
    - Uses XML syntax
    - although more complex!
  - Defined by W3C (W3C Recommendation)

- Primary features
  - Datatypes
    - e.g. integer, float, date, etc...
  - More powerful content models
    - e.g. type derivation, etc...

Sample Schema

```
<?xml version="1.0“?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <xsd:element name="age">
        <xsd:simpleType>
        <xsd:restriction base="xsd:positiveInteger" >
        <xsd:maxInclusive value="120"/>
        </xsd:restriction>
</xsd:simpleType>
   </xsd:element>
   <xsd:element name="name">
        <xsd:complexType>
             <xsd:attribute name=“first" type="xsd:string"
   use="required"/>
             <xsd:attribute name=“last" type="xsd:string" use="required"/>
        </xsd:complexType>
   </xsd:element>
   <xsd:element name="person">
        <xsd:complexType>
             <xsd:sequence>
                 <xsd:element ref="name"/>
                 <xsd:element ref="age"/>
             </xsd:sequence>
        </xsd:complexType>
   </xsd:element>
</xsd:schema>
```

Schema Components

- Simple type definitions - Complex type definitions - Element declarations	Primary components
- Attribute Group definitions - Identity constraint definitions - Model Group definitions - Notation declarations	Secondary components
- Annotations - Attribute declarations - Model groups - Particles - Wildcards	Helper components

XML Schema - Structure

- The **<schema>** element: The root element of an XML Schema.
  - Has many attributes to specify general information about the schema such as default values for attributes, and target namespace.

- Customary to use **xs** or **xsd** as the namespace prefix in an XML Schema.

Schema Wrapper

```
<xsd:schema
    xmlns:xsd=“http://www.w3.org/2001/XMLSchema”>
….
</xsd:schema>
```

Declaration vs. Definition

- declarations - things that will be used in an XML instance document.
  - element declarations
  - attribute declarations
- definitions - things that are just used in the schema document; a definition creates a new type
  - simple type definitions
  - complex type definitions
  - attribute group, model group definitions

Element Declarations

- Created with the <element> tag.
- Child elements can be <simpleType>, <complexType>, <simpleContent>, and <complexContent> to specify specify details about the content model for that element.
- Can specify default or fixed values.

Local and Global Elements

- Local element:

```
<xsd:complexType name="ttt">
    <xsd:sequence>
        <xsd:element name="address" type="..."/>...
    </xsd:sequence>
</xsd:complexType>
```

- Global element:

```
<xsd:element name="address" type="..."/>

<xsd:complexType name="ttt">
    <xsd:sequence>
        <xsd:element ref="address"/>  ...
    </xsd:sequence>
</xsd:complexType>
```

ref

Element Declarations

```
<xsd:element name=“myglobalelt1” type=“mySimpleType”/>
<xsd:element name=“myglobalelt2” type=“myComplexType”/>
<xsd:element name=“myglobalelt3”>
      <xsd:complexType>
        <xsd:sequence>
           <xsd:element name=“mylocalelt” type=“otherType”/>
           <xsd:element ref=“myglobalelt2”/>
        </xsd:sequence>
        <xsd:attribute name=“mylocalattr” type=“xsd:date”/>
     </xsd:complexType>
</xsd:element>
```

Constraining Attributes of <element>

- default – specify a default value for the element
- fixed – forces the instance document to include information
- maxOccurs, minOccurs– specifies how many times an element may appear in the document or content model. 0,1,...n or unbounded

Attribute Declarations

- Attributes are declared with the **<attribute>** tag.
- Can be global or local.
- Can have a default or fixed value.
- Attribute groups can be created using the <attributeGroup> tag.
- Enumerations can be created using the <enumeration> tag as a restriction.

Attributes in XML Schema

```
<xsd:element name=“paper” type=“papertype”/>
<xsd:complexType name=“papertype”>
    <xsd:sequence>
          <xsd:element name=“title” type=“xsd:string”/>
          . . . . . .
</xsd:sequence>
 <xsd:attribute name=“language" type="xsd:language" fixed=“en-GB"/>
</xsd:complexType>
```

- Attributes are associated to the *type*, not to the element
- Only to *complex types*.

Occurrence Constraints for Elements and Attributes

Elements (minOccurs, maxOccurs) fixed, default	Attributes use, fixed, default	Notes
(1, 1) -, -	required, -, -	element/attribute must appear once, it may have any value
(1, 1) 37, -	required, 37, -	element/attribute must appear once, its value must be 37
(2, unbounded) 37, -	n/a	element must appear twice or more, its value must be 37; in general, **minOccurs** and **maxOccurs** values may be positive integers, and **maxOccurs** value may also be "unbounded"
(0, 1) -, -	optional, -, -	element/attribute may appear once, it may have any value

Occurrence Constraints for Elements and Attributes (2)

Elements (minOccurs, maxOccurs) fixed, default	Attributes use, fixed, default	Notes
(0, 1) 37, -	optional, 37, -	element/attribute may appear once, if it does appear its value must be 37, if it does not appear its value is 37
(0, 1) -, 37	optional, -, 37	element/attribute may appear once; if it does not appear its value is 37, otherwise its value is that given
(0, 2) -, 37	n/a	element may appear once, twice, or not at all; if the element does not appear it is not provided; if it does appear and it is empty, its value is 37; otherwise its value is that given;
(0, 0) -, -	prohibited, -, -	element/attribute must not appear

Note

- Important Note:

  neither **minOccurs**, **maxOccurs**, nor **use** may appear in the declarations of global elements and attributes.

Group Element

- The group element enables you to group together element declarations.
- the group element is only for grouping together element declarations, no attribute declarations allowed.
- compositors – *all* / *sequence* / *choice*

"All" Group

```
<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>
```

- A restricted form of &
- Restrictions:
  - Only at top level
  - Has only elements
  - Each element occurs at most once
- E.g. "comment" occurs 0 or 1 times

“Sequence” and “Choice” Group Example

```
<xsd:group name=“myModelGroup”>
      <xsd:sequence>
           <xsd:element name=“firstThing” type=“type1”/>
           <xsd:element name=“secondThing” type=“type2”/>
      </xsd:sequence>
</xsd:group>

<xsd:complexType name=“newType”>
    <xsd:choice>
        <xsd:group ref=“myModelGroup”/>
        <xsd:element ref=“anotherThing”/>
   </xsd:choice>

</xsd:complexType>
```

Attribute Groups

```
<xsd:attributeGroup name=“myAttrGroup”>
        <xsd:attribute name=“myD1” type=“xsd:string”/>
        <xsd:attribute name=“myD2” type=“xsd:integer”/>
        <xsd:attribute name=“myD3” type=“xsd:date”/>
</xsd:attributeGroup>
<xsd:complexType name=“myDS”>
      <xsd:element name=“myelement” type=“myType”/>
      <xsd:attributeGroup ref=“myAttrGroup”/>
</xsd:complexType>
```

note: attribute declarations always come last after the element declarations

XML Schema – Type Definitions

- There are 2 types of type definitions: simple and complex.
- They are used to define datatypes which may act as global type definitions or be applied to other components, such as element declarations and attribute declarations.
- All types are derived from *anyType.*

Elements vs. Types

```
<xsd:element name="person">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="name"
                              type="xsd:string"/>
      <xsd:element name="address"
                              type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

```
<xsd:element name="person" type="ttt"/>
<xsd:complexType name="ttt">
  <xsd:sequence>
    <xsd:element name="name"
                              type="xsd:string"/>
    <xsd:element name="address"
                              type="xsd:string"/>
  </xsd:sequence>
</xsd:complexType>
```

Local and Global Types in XML Schema

- Local type:

```
<xsd:element name="person">
        [define locally the person's type]
</xsd:element>
```

- Global type:

```
<xsd:element name="person" type="ttt"/>

<xsd:complexType name="ttt">
        [define here the type ttt]
</xsd:complexType>
```

Global types: can be reused in other elements

XML Schema Types

- Simple types
  - Basic datatypes
  - Can be used for attributes *and* element text
  - Extendable
- Complex types
  - Defines structure of elements
  - Extendable
- Types can be named or “anonymous”

Simple Types

- A simple type is a restriction on a primitive type or another user-defined simple type.
- Can be defined anywhere in the schema (named), or used anonymously within an element declaration.
- Use the <simpleType> tag.

Some Simple Types

xsd: prefix

Simple Type	Examples (delimited by commas)
byte	-1, 126
unsignedByte	0, 126
short / unsignedShort	-1, 12678 / 0, 12678
integer	-126789, -1, 0, 1, 126789
positiveInteger	1, 126789
negativeInteger	-126789, -1
nonNegativeInteger	0, 1, 126789
nonPositiveInteger	-126789, -1, 0
int / unsignedInt	-1, 1267896754 / 0, 1267896754
decimal	-1.23, 0, 123.4, 1000.00
float / double	-INF, -1E4, -0, 0, 12.78E-2, 12, INF

Some Simple Types (2)

xsd: prefix

Simple Type	Examples (delimited by commas)
string	Some string
boolean	true, false , 1 , 0
date	-1999-05-31
gMonth	--05--
gYear	1999
gYearMonth	1999-02
gDay	---31
gMonthDay	--05-31
Name	shipTo
anyURI	http://www.example.com/doc.html#ID5
language	en-GB, en-US, fr

Deriving Simple Types

- Apply facets
  - Specify enumerated values
  - Add restrictions to data
  - Restrict lexical space
    - Allowed length, pattern, etc...
  - Restrict value space
    - Minimum/maximum values, etc...
- Extend by list or union

Facets of Simple Types

•Facets = additional properties restricting a simple type

•15 facets defined by XML Schema

Examples:

- length
- minLength
- maxLength
- enumeration
- totalDigits
- maxInclusive
- maxExclusive
- minInclusive
- minExclusive
- fractionDigits

Using the Enumeration Facet

```
<xsd:attribute name=“gender”>
     <xsd:simpleType>
          <xsd:restriction base=“xsd:string”>
               <xsd:enumeration value=“M”/>
               <xsd:enumeration value=“F”/>
          </xsd:restriction>
     </xsd:simpleType>
</xsd:attribute>
```

Anonymous Simple Type (local)

without name attribute

```
<xsd:attribute name="myAttribute" default="42">
     <xsd:simpleType>
          <xsd:restriction base="xsd:integer">
               <xsd:minExclusive value="0"/>
          </xsd:restriction>
     </xsd:simpleType>
</xsd:attribute>
```

A Simple Type Example (1)

- Integer with value (1234, 5678]

```
<xsd:simpleType name="MyInteger">
  <xsd:restriction base="xsd:integer">
    <xsd:minExclusive value="1234"/>
    <xsd:maxInclusive value="5678"/>
  </xsd:restriction>
</xsd:simpleType>
```

A Simple Type Example (2)

- Validating integer with value (1234, 5678]

01	`<data></data>`	INVALID
02	`<data>Andy</data>`	INVALID
03	`<data>-32</data>`	INVALID
04	`<data>1233</data>`	INVALID
05	`<data>1234</data>`	INVALID
06	`<data>1235</data>`	
07	`<data>5678</data>`	
08	`<data>5679</data>`	INVALID

Not so Simple Types

- List types:

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

```
<listOfMyInt>2003 3037 4977 5455</listOfMyInt>
```

Not so Simple Types

- List types :

```
<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>
```

```
<sixStates>PA NY CA NY LA AK</sixStates>
```

Not so Simple Types

- Union types :

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

When we define a union type, the memberTypes attribute value is a list of all the types in the union.

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

Complex Types

- XML Schema Complex Types define structure of elements
- Complex types can be restrictions or extensions.
  - Can be a restriction on a simple type.
  - Can be a restriction on some other complex type.
  - Can be an extension of a simple type.
  - Can be an extension of another complex type.
- Declared either globally or locally.

Deriving Types by Extension

- extend a base type's content by adding elements and/or attributes

```
<xsd:complexType name="newType" >
<xsd:complexContent>
              <xsd:extension base="baseType">
              <xsd:sequence>
       <xsd:element name="newelt" type="xsd:string"/>
      </xsd:sequence>
       <xsd:attribute name="newattr" type="xsd:integer"/>
              </xsd:extension>
</xsd:complexContent>
</xsd:complexType>
```

Deriving Types by Restriction

- Narrows the ranges or reduces alternatives of elements and/or attributes

```
<xsd:complexType name="myType">
<xsd:sequence>
      <xsd:element name="elt1" type="xsd:integer" minOccurs="0"/>
  </xsd:sequence>
      <xsd:attribute name="attr1" type="xsd:string"/>
</xsd:complexType>

<xsd:complexType name="restrictedType">
  <xsd:complexContent>
    <xsd:restriction base="myType">
      <xsd:sequence>
       <xsd:element name="elt1" type="xsd:integer" minOccurs="1"/>
       </xsd:sequence>
       <xsd:attribute name="attr1" type="xsd:string" fixed="sausage"/>
    </xsd:restriction>
  </xsd:complexContent>
</xsd:complexType>
```

Anonymous Complex Type (local)

without name attribute

```
<xsd:element name=“personName”>
    <xsd:complexType>
     <xsd:sequence>
        <xsd:element name=“title”/>
        <xsd:element name=“forename”/>
        <xsd:element name=“surname”/>
     </xsd:sequence>
        <xsd:attribute name=“age” type=“xsd:integer”/>
   </xsd:complexType>
</xsd:element>
```

A Complex Type Example (1)

- Mixed content that allows <b>, <i>, and <u> elements and text

```
<xsd:complexType name=‘RichText’ mixed=‘true’>
  <xsd:choice minOccurs=‘0’ maxOccurs=‘unbounded’>
    <xsd:element name=‘b’ type=‘RichText’/>
    <xsd:element name=‘i’ type=‘RichText’/>
    <xsd:element name=‘u’ type=‘RichText’/>
  </xsd:choice>
</xsd:complexType>
```

A Complex Type Example (2)

- Validation of RichText

```
<content></content>
<content>XML</content>
<content>XML is <i>good</i></content>
<content><B>bold</B></content>            INVALID
<content><foo/></content>                 INVALID
<content>XML is <b>good</b></content>
```

Corresponding Schema to XML Document

- In the XML document:

xsi namespace

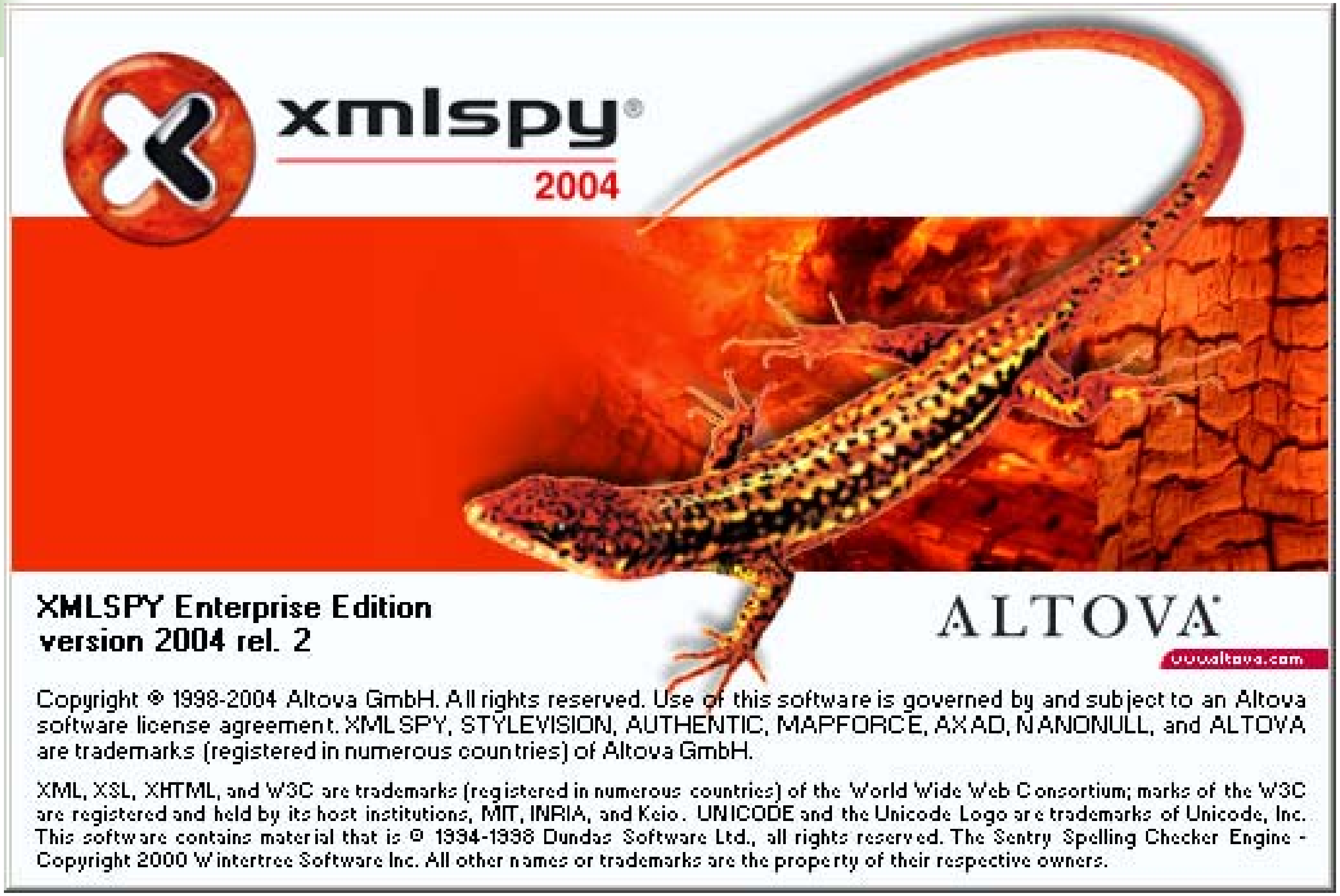
```
<course
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:noNamespaceSchemaLocation="course.xsd">
...
</course>
```

to use schema elements without namespace

XML Schema Validation

- XML Schema Validator (W3C): XSV - online!
- XML Editors and Parsers with validation service
- Hard to recommend any one XML Schema tool. XML Spy is most popular by far, but has flaws. XSV or Xerces (XML Schema-aware Parser) more reliable but all lack GUIs.

XML Spy (www.altova.com)



XMLSPY Enterprise Edition
version 2004 rel. 2

Copyright © 1998-2004 Altova GmbH. All rights reserved. Use of this software is governed by and subject to an Altova software license agreement. XMLSPY, STYLEVISION, AUTHENTIC, MAPFORCE, AXAD, NANONULL, and ALTOVA are trademarks (registered in numerous countries) of Altova GmbH.

XML, XSL, XHTML, and W3C are trademarks (registered in numerous countries) of the World Wide Web Consortium; marks of the W3C are registered and held by its host institutions, MIT, INRIA, and Keio. UNICODE and the Unicode Logo are trademarks of Unicode, Inc. This software contains material that is © 1994-1998 Dundas Software Ltd., all rights reserved. The Sentry Spelling Checker Engine - Copyright 2000 Wintertree Software Inc. All other names or trademarks are the property of their respective owners.

Read More in

- World Wide Web Consortium. XML Schema. W3C Recommendation. Available at http://www.w3.org/XML/Schema

- XML.com articles
  - http://www.xml.com/pub/a/2001/06/06/schemasimple.html