Processing of structured documents

Spring 2003, Part 3
Helena Ahonen-Myka

Building content models

- <xsd:sequence>: fixed order
- <xsd:choice>: (1) choice of alternatives
- <xsd:group>: grouping (also named)
- <xsd:all>: no order specified

Building content models

- a simplified view of the allowed structure of a complex type:
  - complexType -> annotations?, (simpleContent | complexContent | ((all | choice | sequence | group)? , attrDecls))

Nested choice and sequence groups

```
<xsd:complexType name="PurchaseOrderType">
  <xsd:sequence>
    <xsd:choice>
      <xsd:group     ref="shipAndBill" />
      <xsd:element   name="singleUSAddress"
                     type="USAddress" />
    </xsd:choice>
    <xsd:element name="items"    type="Items" />
  </xsd:sequence>
```


Nested choice and sequence groups

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


An 'all' group

- An 'all' group: all the elements in the group may appear once or not at all, and they may appear in any order
  - minOccurrs and maxOccurs can be 0 or 1
- limited to the top-level of any content model
- has to be the only child at the top
- group's children must all be individual elements (no groups)

An '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>
```

7

Occurrence constraints

- Groups represented by 'group', 'choice', 'sequence' and 'all' may carry minOccurs and maxOccurs attributes
- by combining and nesting the various groups, and by setting the values of minOccurs and maxOccurs, it is possible to represent any content model expressible with an XML 1.0 DTD
  - 'all' group provides additional expressive power

8

Attribute groups

- Also attribute definitions can be grouped and named

```
<xsd:element name="item">
  <xsd:complexType>
    <xsd:sequence> … </xsd:sequence>
    <xsd:attributeGroup ref="ItemDelivery" />
  </xsd:complexType></xsd:element>

<xsd:attributeGroup name="ItemDelivery">
  <xsd:attribute name="partNum" type="SKU" />
  …
</xsd:attributeGroup>
```

9

Namespaces and XML Schema

- An XML Schema document contains declarations of namespaces that are used in the document
  - xmlns:xsd="http://www.w3.org/2001/XMLSchema" for the elements and types with special XML Schema semantics
  - target namespace
  - namespaces for included or imported schema components (types, elements, attributes)

10

Target namespace

- namespace = a collection of names
- every top-level (global) schema component is added to the target namespace
- if the target namespace is not defined, the global schema components are explicitly without any namespace
- declaration, e.g.: targetNamespace="uri:mywork"

11

Qualified and unqualified locals

- global elements and attributes always have the prefix of their namespace in an instance document
- the prefix of local elements and attributes can be hidden or exposed
  - in a schema: elementFormDefault = "qualified" or "unqualified" (attributeFormDefault similarly)

12

Modularization of schema definitions

- as schemas become larger, it is often desirable to divide their content among several schema documents
- components of other schema documents can be referred using 'include' or 'import'

13

Modularization of schema definitions: include

- 'include':
  - <include schemaLocation="http://www..."/>
  - all the global schema components from the referred schema are available
  - only components with the same namespace or no-namespace components allowed
  - the included no-namespace components are added to the target namespace

14

Modularization of schema definitions: import

- 'import':
  - <import namespace="http://www..."/>
  - namespace has to be declared
  - all the global schema components from the referred schema are available
  - imported components may refer to a different namespace

15

Import

```
<schema xmlns="http://www.w3.org/2001/XMLSchema"
        xmlns:html="http://www.w3.org/1999/xhtml"
        targetNamespace="uri:mywork"
        xmlns:my="uri:mywork">

<import namespace="http://www.w3.org/1999/xhtml">
…
<complexType name="myType">
  <sequence>
    <element ref="html:p" minOccurs="0"/>
  </sequence>
  …
</complexType>
<element name="myElt" type="my:myType"/>
</schema>
```

16

Type libraries

- As XML schemas become more widespread, schema authors will want to create simple and complex types that can be shared and used as the basic building blocks for building new schemas
- XML Schemas already provide types that play this role: the simple types
- other examples: currency, units of measurement, business addresses

17

Example: currencies

```
<schema targetNamespace="http://www.example.com/Currency"
        xmlns:c="http://www.example.com/Currency"
        xmlns="http://www.w3.org/2000/08/XMLSchema">
<complexType name="Currency">
  <simpleContent>
    <extension base="decimal">
      <attribute name="name">
        <simpleType>
          <restriction base="string">
            <enumeration value="AED"/>
            <enumeration value="AFA" />
            <enumeration value="ALL" />
              …
```

18

Extending content models

- Mixed content models
  - an element can contain, in addition to subelements, also arbitrary character data
- import
  - an element can contain elements whose types are imported from external namespaces
  - e.g. this element may contain an HTML 'p' element here
- more flexible way:
  - 'any' element, 'any' attribute

19

Example

```
<purchaseReport
        xmlns="http://www.example.com/Report">
<regions> <!-- part sales  by regions --> </regions>
<parts>  <!-- part descriptions  --> </parts>
<htmlExample>
  <table xmlns="http://www.w3.org/1999/xhtml"
      border="0" width="100%">
    <tr>
      <th align="left">Zip Code</th>
      <th align="left">Part Number </th>
      <th align="left">Quantity</th>
    </tr>
    <tr><td>95819</td><td> </td> <td> </td></tr>
    <tr><td> </td><td>872-AAA</td><td>1</td></tr>
    ...
```

20

Including an HTML table

- To permit the appearance of HTML in the instance document we modify the report schema by declaring the content of the element 'htmlExample' by the 'any' element
- in general, an 'any' element specifies that any well-formed XML is permissible in a type's content model
- in the example, we require the XML to belong to the namespace http://www.w3.org/1999/xhtml
  - -> the XML should be XHTML

21

Schema declaration with any

```
<element name="purchaseReport">
 <complexType>
  <sequence>
   <element name="regions" type="r:RegionsType"/>
   <element name="parts" type="r:PartsType"/>
   <element name="htmlExample">
      <complexType>
        <sequence>
          <any namespace="http://www.w3.org/1999/xhtml"
            minOccurs="1" maxOccurs="unbounded"
            processContents="skip"/>
        </sequence>
      ...
```

22

Schema validation

- The attribute 'processContents'
  - 'skip': no validation
  - 'strict': an XML processor is obliged to obtain the schema associated with the required namespace and validate the HTML appearing within the htmlExample element

23

anyAttribute

```
<element name="htmlExample">
  <complexType>
    <sequence>
      <any namespace="http://www.w3.org/1999/xhtml"
            minOccurs="1" maxOccurs="unbounded"
            processContents="skip"/>
    </sequence>
    <anyAttribute namespace="http://www.w3.org/1999/xhtml"/>
  </complexType>
</element>
```

24

Other features in XML Schema

- deriving complex types by extension and restriction
- redefining types and groups
- substitution groups
- abstract elements and types
- keys and references

25

XML Schema best practices?

- design decisions, e.g.
  - Element or type?
  - Global vs. local?
  - How to use namespaces (0 vs 1 vs many)?
  - Hide vs expose namespaces in instances?
- XML Schema Best Practices web site
  - See a link on our material page

26

Other schema languages

- XDR
- SOX
- Schematron
- DSD
- RELAX (NG), TREX

27

Example 1: DTD

```
<!DOCTYPE addressBook [
<!ELEMENT addressBook (card*)>
<!ELEMENT card (name, email)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT email (#PCDATA)>
]>
```

28

Example 1: RELAX NG

```
<element name="addressBook"
xmlns=http://relaxng.org/ns/structure/1.0>
  <zeroOrMore>
      <element name="card">
          <element name="name">
              <text />
          </element>
          <element name="email">
              <text />
          </element>
      </element>
  </zeroOrMore>
</element>
```

29

Example 2: DTD

```
<!DOCTYPE addressBook [
<!ELEMENT addressBook (card*)>
<!ELEMENT card EMPTY>
<!ATTLIST card
      name CDATA #REQUIRED
      email CDATA #REQUIRED>
]>
```

30

Example 2: RELAX NG

```
<element name="addressBook"
xmlns=http://relaxng.org/ns/structure/1.0>
  <zeroOrMore>
      <element name="card">
          <attribute name="name">
              <text />
          </attribute>
          <attribute name="email">
              <text />
          </attribute>
      </element>
  </zeroOrMore>
</element>
```

31