Methods and Tools for Management Information Systems

Lecture 3

8. November 2010


- Set of Recommendations that define a language to describe XML vocabularies precisely using an object-oriented approach:
  - XML Schema Part 0: Primer (*non-normative*)
  - XML Schema Part 1: Structures
  - XML Schema Part 2: Datatypes
- Formalization of the constraints, expressed as rules or a model of structure, that apply to a class of XML documents

- DTDs provide the capability to do basic validation of:
  - Element nesting
  - Element occurrence constraints
  - Permitted attributes
  - Attribute types and default values
- DTDs do not provide fine control over the length, type, or format of elements and attributes values declared to contain character data


- Design goals of XML schema:
  - More precision in describing document structures and their contents
  - Support of XML namespaces
  - Usage of XML vocabulary to describe XML vocabularies
  - Extensible type system
  - Reuse of existing XML vocabularies
- W3C XML Schema standard includes the following features:
  - Simple and complex data types
  - Type derivation and inheritance
  - Element occurrence constraints
  - Namespace-aware element and attribute declarations

- XML schemas are defined using XSD, the *XML Schema Description Language*, a context-free, regular grammar:
  - XSD Part 1 covers the definition of content models for reusable element and attribute structures
  - XSD Part 2 establishes a type system
  - Every XML schema is an XML document of its own, i. e., no internal subsets
  - All primitives for schema creation are part of the namespace `http://www.w3.org/2001/XMLSchema`
  - All primitives for schema usage are part of the namespace `http://www.w3.org/2001/XMLSchema-instance`


- Root element of a schema definition is `schema`:

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="rulebase">
    <!-- ... -->
  </xs:element>
</xs:schema>
```

or, if namespaces are used:

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://somewhere.org/rulebase"
    xmlns:rulebase="http://somewhere.org/rulebase"
    elementFormDefault="qualified"
    attributeFormDefault="qualified">
  <xs:element name="rulebase">
    <!-- ... -->
  </xs:element>
</xs:schema>
```

- Schema documents are referenced either through `schemaLocation` or `noNameSpaceSchemaLocation`, depending on whether namespaces are used:

```
<?xml version="1.0" encoding="UTF-8"?>
<mining xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://somewhere.org/mining mining.xsd"
    xmlns="http://somewhere.org/mining">
  <!-- ... -->
</mining>
```

or not:

```
<?xml version="1.0" encoding="UTF-8"?>
<rulebase xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNameSpaceSchemaLocation="rulebase.xsd">
  <!-- ... -->
</rulebase>
```

- *Example:*
  - A simple address book entry:

```
<?xml version="1.0"?>
<fullName>Scott Means</fullName>
```

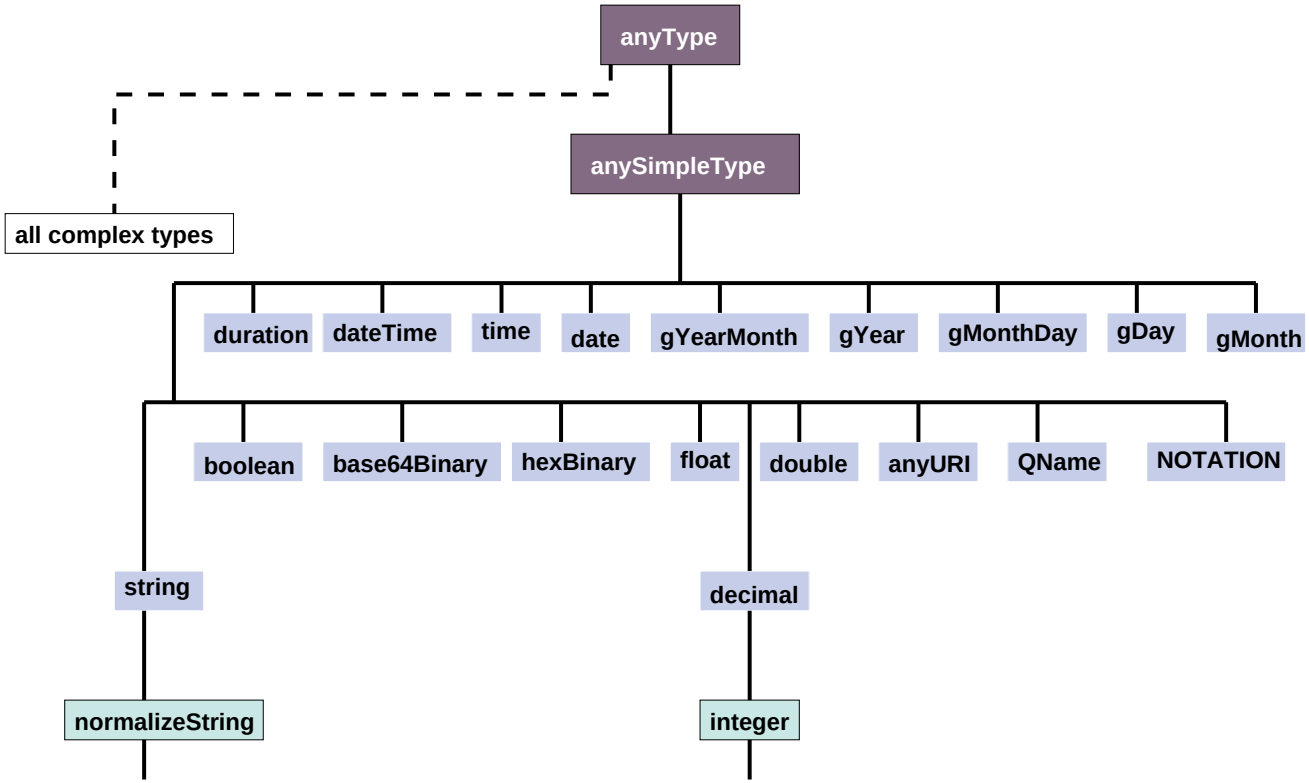
  - Corresponding schema document:

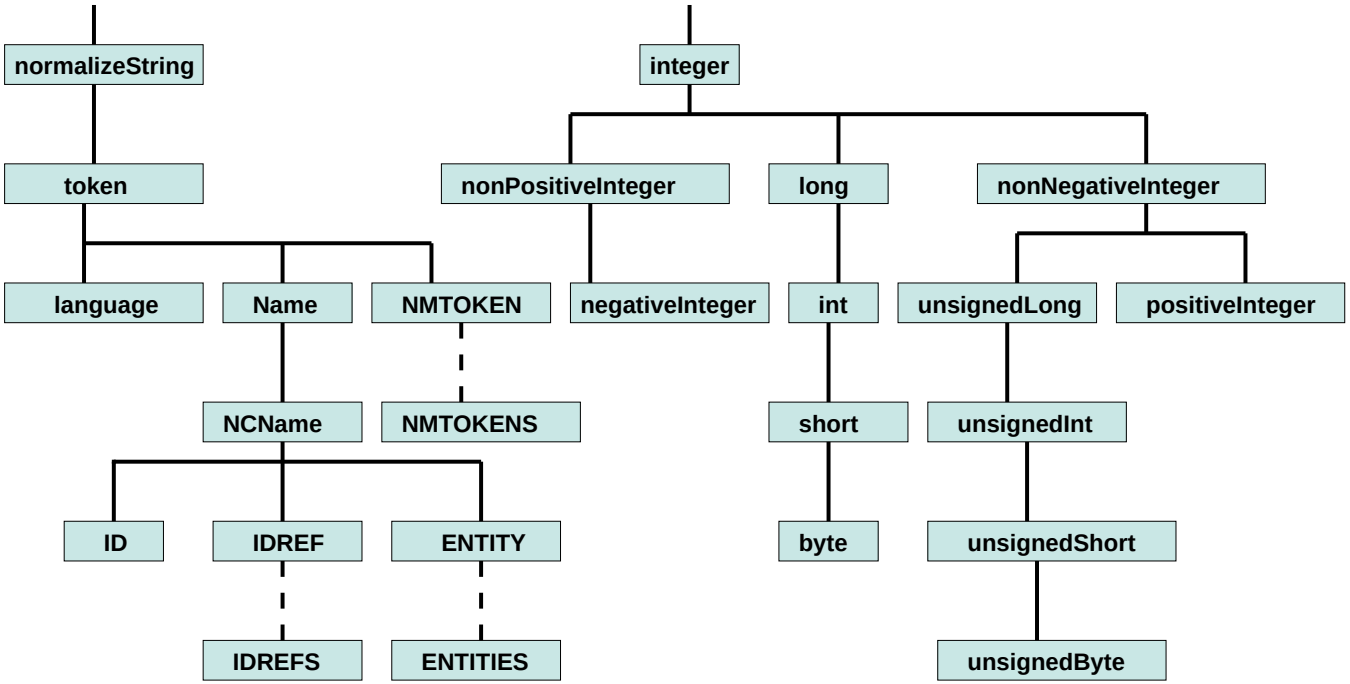
```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="fullName" type="xs:string"/>
</xs:schema>
```

  - Address book entry with schema reference:

```
<?xml version="1.0"?>
<fullName xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="address-schema.xsd">
  Scott Means</fullName>
```

- XSD defines a total of 44 primitive data types such as `string`, `int`, `double`, `date`, etc., including those used in DTDs
- Most general type is `anyType`, conceptionally the *union* of all data types



anyType
anySimpleType
all complex types
duration
dateTime
time
date
gYearMonth
gYear
gMonthDay
gDay
gMonth
boolean
base64Binary
hexBinary
float
double
anyURI
QName
NOTATION
string
decimal
normalizeString
integer


normalizeString
integer
token
nonPositiveInteger
long
nonNegativeInteger
language
Name
NMTOKEN
negativeInteger
int
unsignedLong
positiveInteger
NCName
NMTOKENS
short
unsignedInt
ID
IDREF
ENTITY
byte
unsignedShort
IDREFS
ENTITIES
unsignedByte

- Defining the structure of XML vocabularies using XSD:
  - Definition of an unstructered, typed element:

    ```
    <xs:element name="element_name" type="type_name"/>
    ```

  - Elements may be further characterized by attributes:

Attribute	Description
`abstract`	element must not be used in an XML document (default `false`)
`block`	allows for control of usage of derived types (`#all` or list of `extension`, `restriction`, and `substitution`)


Attribute	Description
`default`	default value (has to be compatible with type chosen)
`final`	allows for restrictions on type extension and restriction (`#all`, `extension` or `restriction`)
`fixed`	element value is constant and immutable
`form`	element usage with (`qualified`) or without (`unqualified`) namespace prefix
`id`	unique element identifier
`minOccurs`	minimum cardinality (default 1)

Attribute	Description
`maxOccurs`	maximum cardinality (default 1) or the value `unbounded`
`name`	unqualified name of the element (`NCName`)
`nillable`	allows for null values (default `false`)
`ref`	reference to another, globally defined element declaration
`substitutionGroup`	name of a group of elements which may appear in the instance document (instead of the current element)
`type`	attribute type (default `anyType`)


- Rules that apply when using element attributes:
  - ⇒ `default` and `fixed` are mutually exclusive
  - ⇒ default value is applied only if the element is present and empty
  - ⇒ `default` and `fixed` apply only to simple type elements
  - ⇒ usage of `name` and `ref` are mutually exclusive
  - ⇒ `type` is only applicable if no type definition is embedded

Notice: elements defined globally can be used *everywhere* in the instance document, even as the root element


- Definition of empty elements:

```
<xs:element name="phone" minOccurs="0">
  <xs:complexType>
    <xs:attribute name="number" type="xs:string"/>
  </xs:complexType>
</xs:element>
```

- Definition of elements containing complex content only:

```
<xs:element name="phone" minOccurs="0">
  <xs:complexType>
    <xs:complexContent>
      <xs:restriction base="xs:anyType">
        <xs:attribute name="number" type="xs:string"/>
      </xs:restriction>
    </xs:complexContent>
  </xs:complexType>
</xs:element>
```

- *Examples:*

```
<xs:element name="birthdate" type="xs:date"/>

<xs:element name="PI" type="xs:double"
    fixed="3.141592653"
    block="#all"
    final="#all"/>

<xs:element name="firstname" type="xs:token"
    minOccurs="1"
    maxOccurs="unbounded"/>

<xs:element name="productID" type="xs:NCName"
    form="qualified"/>
```


- Complex user-defined types are declared using `complexType`:

```
<xs:element name="element_name">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="sub_element_name"/>
      <!-- ... -->
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

- Enumeration of child elements using `sequence`, selection of elements using `choice`, selection with unbounded cardinality using `all`

  Notice: occurrence may only be 0 or 1 in `xs:all`

- *Named* vs. *anonymous* complex types:

```
<xs:element name="person">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="firstname" type="xs:token"
          maxOccurs="unbounded"/>
      <xs:element name="lastname" type="xs:token"/>
      <xs:element ref="skill_profile" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

- *Named* vs. *anonymous* complex types (cont'd):

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:complexType name="PersonType">
    <xs:sequence>
      <xs:element name="firstname" type="xs:string"
          maxOccurs="unbounded"/>
      <xs:element name="lastname" type="xs:string"/>
      <xs:element ref="skill_profile" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>

  <xs:element name="person" type="PersonType"
      maxOccurs="unbounded"/>
</xs:schema>
```

- Declaration of user-defined types by *restriction*:

```
<xs:complexType name="parentType">
  <xs:sequence>
    <xs:element name="elementA" type="xs:int"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="childType">
  <xs:complexContent>
    <xs:restriction base="parentType">
      <xs:sequence>
        <xs:element name="elementA" type="xs:short"/>
      </xs:sequence>
    </xs:restriction>
  </xs:complexContent>
</xs:complexType>
```


- Declaration of user-defined types by *extension*:

```
<xs:complexType name="parentElement">
  <xs:sequence>
    <xs:element name="elementA"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="childElement">
  <xs:complexContent>
    <xs:extension base="parentElement">
      <xs:sequence>
        <xs:element name="elementB"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```


- Extension of simple types:

```
<xs:schema
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="qualified"
    attributeFormDefault="unqualified">
  <xs:element name="firstname">
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base="xs:string">
          <xs:attribute name="hasChristianName" type="xs:boolean"/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

- Declaration of user-defined (simple) types by restriction or derivation of existing types using `simpleType`:

```
<xs:simpleType name="short" id="short">
  <xs:restriction base="xs:int">
    <xs:minInclusive value="-32768" id="short.minInclusive"/>
    <xs:maxInclusive value="32767" id="short.maxInclusive"/>
  </xs:restriction>
</xs:simpleType>
```

- Union of a non-empty set of (aggregated) base types:

```
<xs:simpleType name="appointment">
  <xs:union memberTypes="xs:date weekdays"/>
</xs:simpleType>
```

- Aggregation (list):

```
<xs:simpleType name="marketBasketElements">
  <xs:list itemType="xs:string"/>
</xs:simpleType>
```

- Constraints for simples types (facets):

Attribute	Description
`length`	restricts the length of a simple type or the number of values in an aggregation
`minlength`	minimum length
`maxlength`	maximum length
`pattern`	content definition using regular expressions

Attribute	Description
`enumeration`	complete list of permissible values
`whitespace`	handling of whitespace (one of `preserve`, `replace` and `collapse`)
`maxInclusive`	permissible values (less than or equal)
`maxExclusive`	permissible values (less than)
`minInclusive`	permissible values (greater than or equal)
`minExclusive`	permissible values (greater than)
`totalDigits`	total number of digits
`fractionDigits`	number of digits to the right of the decimal point


- Definition of attributes on the global or local level:

```
<xs:attribute name="attribute_name" type="type_name"/>
```

- Attributes may be further characterized by attributes as well:

Attribute	Description
`name`	name of the attribute (`NCNAME`)
`id`	unique attribute identifier
`default`	default value
`fixed`	attribute value is constant and immutable
`type`	attribute type (default `simpleType`)
`form`	element usage with (`qualified`) or without (`unqualified`) namespace prefix


Attribute	Description
`ref`	reference to another, globally defined attribute (mutually exclusive with `name`
`use`	attribute is `optional`, `required` or `prohibited` (default `optional`)

Examples:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="qualified">
  <xs:attribute name="myAtt1"/>
  <xs:attribute name="myAtt2" type="xs:decimal"/>
  <xs:attribute name="myAtt3">
    <xs:simpleType>
      <xs:restriction base="xs:int">
        <xs:minInclusive value="10"/>
        <xs:maxInclusive value="20"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
```


```
<xs:simpleType name="myType1">
  <xs:restriction base="xs:string">
    <xs:maxLength value="5"/>
  </xs:restriction>
</xs:simpleType>
<xs:attribute name="myAtt4" type="myType1"/>
<xs:element name="foo">
  <xs:complexType>
    <xs:attribute ref="myAtt1" use="optional"/>
    <xs:attribute ref="myAtt2" use="required"/>
    <xs:attribute ref="myAtt3" use="prohibited"/>
    <xs:attribute ref="myAtt4"/>
    <xs:attribute name="myAtt5" type="xs:date" id="myDate"/>
```

```
      <xs:attribute name="myAtt6">
        <xs:simpleType>
          <xs:restriction base="xs:float">
            <xs:totalDigits value="5"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
    </xs:complexType>
  </xs:element>
</xs:schema>
```


- Declaration of attribute groups:

```
<xs:attributeGroup name="nationality">
  <xs:attribute name="language" type="xs:string"/>
  <xs:attribute name="origin" type="xs:string"/>
  <!-- ... -->
</xs:attributeGroup>

<xs:element name="person">
  <!-- ... -->
  <xs:attributeGroup ref="nationality">
  <!-- ... -->
</xs:element>
```