Terminology Clarification

- **Schema**
  - Refers to the W3C XML Schema
- **Schemas**
  - Refers to one or more industry standards
- **schema (lowercase)**
  - Refers to an information model
  - Can be a DTD or Schema
  - May also be a relational database schema
General Concepts

DTD and XML Schema

Document Type Definition

- An information modeling syntax

```
<!ELEMENT x (#PCDATA)>
<!ELEMENT y (#PCDATA)>
```

x and y are always interpreted as String, regardless of how they are used.
XML Schema

- An information modeling syntax in XML

```
<element name="x" type="boolean" />
<element name="y" type="integer" />
```

x and y now have enforceable data types

Schema Validation

- Schemas can be used to validate a document in two ways
- **Content Model Validation**
  - Checks order and nesting of elements
  - Similar to DTD validation
- **Data Type Validation**
  - Checks the element content for valid type and range
  
  Example: month element is an integer between 1 and 12
  
  ```
  <month>15</month> INVALID
  <month>5</month>  VALID
  ```
Comparison with DTDs (1)

- DTDs use their own unique syntax (EBNF)
  - XML Schemas use XML syntax

- DTDs are concise
  - XML Schemas are verbose

Comparison with DTDs (2)

- XML Schemas can be parsed and manipulated programmatically like any other XML document
  - DTDs cannot

Note: Custom DTD parsers are available, but the ability to parse a DTD is not inherent in most XML parsers.
Comparison with DTDs (3)

XML Schemas support a number of datatypes
(int, float, boolean, date, etc.)

leftrightarrow DTDs treat all data as strings

Many tools exist for validating documents against DTDs

leftrightarrow Not many tools exist to do so with XML Schemas
- Tools emerging
  • XML Authority, XML Writer, and others

Internet Software Lab., Univ. of Incheon

Comparison with DTDs (4)

XML Schemas allow open-ended data models
- vocabulary extension and inheritance

leftrightarrow DTDs support only a closed model

XML Schemas support attribute groups

leftrightarrow DTDs offer only limited attribute group support

Internet Software Lab., Univ. of Incheon
Comparison with DTDs (5)

XML Schemas support **namespace** integration
- Allow the association of individual nodes of a
document with type declarations in a schema

⇔ DTDs allow only one association
- between the document and the DTD

---

**XML Schema Features**

- Rich datatypes
  - integer, float, date, time, boolean, ...
- User-defined types
- Extendable types
- Open, closed or refinable content models
- Grouping
- Namespace support
XML Schemas vs. DTDs

<table>
<thead>
<tr>
<th>XML Schemas</th>
<th>DTDs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Support namespaces</td>
<td>n/a</td>
</tr>
<tr>
<td>Written in XML syntax</td>
<td>n/a</td>
</tr>
<tr>
<td>Extensive datatype support</td>
<td>Very limited</td>
</tr>
<tr>
<td>Full, object-oriented extensibility</td>
<td>Extended via string substitutions</td>
</tr>
<tr>
<td>Open, closed or refinable content models</td>
<td>Closed only</td>
</tr>
</tbody>
</table>

Save $$$ using XML Schemas (1)

Code to actually do the work

Code to check the structure and content of the data

In a typical program, up to 60% of the code is spent checking the data!
Save $$$ using XML Schemas (2)

If your data is structured as XML and there is a Schema, then you can hand the data-checking task off to a Schema Validator.

Thus, your code is reduced by up to 60%!!!

Big $$$ savings!

End of DTDs?

NO!!!

- DTDs have
  - Widespread use and support
  - Many legacy applications and documents
  - Too much time and money invested
  - Experienced programmers and consultants
W3C XML Schema

Basic Syntax

Sources

- XML Schema Part 0: Primer
  - David C. Fallside (IBM)
- XML Schema Part 1: Structures
  - Henry S. Thompson (University of Edinburgh)
  - David Beech (Oracle Corp.)
  - Murray Maloney (for Commerce One)
  - Noah Mendelsohn (Lotus Development Corporation)
- XML Schema Part 2: Datatypes
  - Paul V. Biron (Kaiser Permanente, for Health Level Seven)
  - Ashok Malhotra (IBM)
Basic Components

- Declarations
  - These are used by instance documents.

- Types
  - Each declaration has an associated type
  - Type can be ANY

- Definitions
  - Type definitions

Element Types

- Simple Types
  - No element children
  - No attributes

- Complex Types
  - Allow element children
  - Attributes allowed
Anonymous vs. Named Types

- **Anonymous Types**
  - Used within one element only
  - Have no name, so they cannot be referenced
  - Inline declarations

- **Named Types**
  - Have a name
  - Can be referenced and used in other parts of the schema
  - User-defined types
  - Out-of-line declarations

---

**Named Type Example**

```xml
<element name="person">
  <complexType>
    <element ref="name"/>
    <element ref="age" minOccurs='0' maxOccurs='1'/>
    <element ref="hobby" minOccurs='1' maxOccurs='unbounded'/>
  </complexType>
</element>
```

*is equivalent to*

```xml
<element name="person" type="personType"/>
<complexType name="personType">
  <element ref="name"/>
  <element ref="age" minOccurs='0' maxOccurs='1'/>
  <element ref="hobby" minOccurs='1' maxOccurs='unbounded'/>
</complexType>
```
Constraint

- An element can have
  
  type attribute  or  complexType child element

- But not Both!

```xml
<element name="person" type="personType">
  <complexType>
    ...
  </complexType>
</element>
```

Occurrences of Element (1)

- For elements:
  - The default value of minOccurs is 1
  - `unbounded` is used to indicate that there is no maximum number of occurrences
  - There is no default value for maxOccurs
    - 1 when minOccurs is 0 or 1
    - equals minOccurs when minOccurs is anything other than 0 or 1
    - If no maxOccurs, then it is equal to minOccurs
    - If no minOccurs, then the element must appear exactly once
### Occurrences of Element (2)

<table>
<thead>
<tr>
<th>Element</th>
<th>minOccurs</th>
<th>maxOccurs</th>
<th>DTD</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>A</td>
<td>0</td>
<td>1</td>
<td>A?</td>
</tr>
<tr>
<td>A</td>
<td>1</td>
<td>unbounded</td>
<td>A+</td>
</tr>
<tr>
<td>A</td>
<td>0</td>
<td>unbounded</td>
<td>A*</td>
</tr>
</tbody>
</table>

### Occurrences of Attribute

- For attributes:
  - Attributes may appear once or not at all
  - A *use* attribute is used in an attribute declaration to indicate whether the attribute’s value is *required* or *optional*
  - If *optional*, whether the attribute’s value is *fixed* or whether there is a *default* value can also be specified
  - An attribute, *value*, provides any value that is called for
Content Types

- empty
- elementOnly
- textOnly
- mixed

Empty Type

DTD:

```xml
<ELEMENT image EMPTY>
<ATTLIST image href CDATA #REQUIRED>
```

Schema:

```xml
<element name="image" minOccurs="0" maxOccurs="unbounded"/>
<complexType content="empty">
  <attribute name="href" type="unReference" use="required" />
</complexType>
</element>
```

XML:

```xml
<image href="http://www.example.com/sample.gif"/>
```
**Mixed Type**

**DTD:**
```
<ELEMENT salutation (#PCDATA, name)>
  <element name="salutation"/>
  <complexType content="mixed">
    <element name="name" type="string"/>
  </complexType>
</element>
```

**Schema:**
```
<element name="salutation"/>
<complexType name="name" type="string"/>
</element>
```

**XML:**
```
<salutation>Dear Mr. Gildong Hong</salutation>
```

**<sequence> vs. <all>**

```
<element name="name">
  <complexType>
    <sequence>
      <element ref="family"/>
      <element ref="given"/>
    </sequence>
  </complexType>
</element>
```

```
<element name="name">
  <complexType>
    <all>
      <element ref="family"/>
      <element ref="given"/>
    </all>
  </complexType>
</element>
```

```
<ELEMENT name (family, given)>
  <ELEMENT name (family, given) | (given, family)>
```
XML Document

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

<person xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
        xsi:noNamespaceSchemaLocation="person.xsd">
    <name>Gildong Hong</name>
    <age>30</age>
    <hobby>reading</hobby>
</person>
```

Equivalent DTD

```xml
<!ELEMENT person (name, age?, hobby+)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT name (#PCDATA)>
```
Sample XML Schema

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

<schema>
  <element name="person">
    <complexType>
      <element ref="name" />
      <element ref="age" minOccurs='0' maxOccurs='1'/>
      <element ref="hobby" minOccurs='1' maxOccurs='unbounded'/>
    </complexType>
  </element>
</schema>
```

Simple (Built-In) Types (1)

- All built-in types are SimpleTypes

  - string
  - boolean
  - float 1.34 (single precision)
  - double 1.343 (double precision)
  - decimal 0, -12.3, 1000
  - timeInstant 2001-02-08T13:20:00.000
  - timeDuration (1 year, 2 months, 3 days, ...)
  - recurringInstant (Feb 25, every year)
  - binary 01001000
  - uri-reference http://www.w3.org
  - XML 1.0 attribute types (ID, IDEF, ENTITY, NOTATION, ...)
### Simple (Built-In) Types (2)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>billTo</td>
<td>XML 1.0 Name</td>
</tr>
<tr>
<td>QName</td>
<td>Namespace qualified name</td>
</tr>
<tr>
<td>NCName</td>
<td>QName without prefix</td>
</tr>
<tr>
<td>integer, negative-integer, ...</td>
<td>123, -123</td>
</tr>
<tr>
<td>long, int, short, byte</td>
<td>5000303030, 50000, -1, 123</td>
</tr>
<tr>
<td>unsigned-long, unsigned-int, ...</td>
<td>0, 0</td>
</tr>
<tr>
<td>date</td>
<td>2001-02-08</td>
</tr>
<tr>
<td>time</td>
<td>13:20:00</td>
</tr>
</tbody>
</table>

### Creating New Datatypes

- Creating new datatypes from an existing datatype (called the "base" type)
- specifying values for one or more of the optional facets for that type
Specifying Facet Values

```xml
<simpleType name="name" base="source">
  <facet value="value" />
  <facet value="value" />
</simpleType>
```

Facets:
- minInclusive
- maxInclusive
- minExclusive
- maxExclusive
- length
- minLength
- maxLength
- pattern
- enumeration

Sources:
- string
- boolean
- float
- double
- decimal
- timeDuration
- recurringDuration
- uriReference

---

Example of the Pattern Facet

```xml
<simpleType name="TelephoneNumber" base="string">
  <length value="8"/>
  <pattern value="\d{3}-\d{4}"/>
</simpleType>
```

- Creating a new datatype called **TelephoneNumber**
  - Elements of this type can hold string values
  - String length must be exactly 8 characters long
  - String must follow the pattern: `ddd-dddd`
  - `d` represents a digit
- The regular expression makes the length facet redundant
Example of the Enumeration Facet

```xml
<element name="airline">
  <complexType>
    <attribute name="carrierName" type="NMTOKEN" use="default" value="KAL">
      <simpleType base="string">
        <enumeration value="KAL"/>
        <enumeration value="Asiana"/>
        <enumeration value="Delta"/>
      </simpleType>
    </attribute>
  </complexType>
</element>
```

```xml
<ATTLIST airline
  carrierName (KAL | Asiana | Delta) "KAL">
```

---

Derived Types

```xml
<schema xmlns="http://www.w3.org/1999/XMLSchema">
  <simpleType name="Sku" base="string">
    <pattern value="\d(3)-(A-Z)(2)="/>
  </simpleType>
  <element name="sku" type="Sku"/>
</schema>
```

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

<sku>345-AB</sku>
```
Guidelines for implementing Dublin Core in XML

UKOLN
University of Bath

URLs

- Guidelines for implementing Dublin Core in XML
  - http://www.ukoln.ac.uk/metadata/dcmi/dc-xml-guidelines/

- Example Dublin Core XML Schemas
  - http://www.ukoln.ac.uk/metadata/dcmi/dcxml/examples.html
Recommendation 1

- Implementors should base their XML applications on XML Schemas rather than XML DTDs

Recommendation 2

- Implementors should use XML Namespaces to uniquely identify DC elements, element refinements and encoding schemes

- XML Namespaces
  - dc="http://purl.org/dc/elements/1.1/"
  - dcterms="http://purl.org/dc/terms/"
  - dcxml="http://purl.org/dc/xml/"
Recommendation 3

- Implementors should encode properties as XML elements and values as the content of those elements

  <dc:title>Dublin Core in XML</dc:title>

  rather than

  <dc:title value="Dublin Core in XML"/>

Recommendation 4

- The property names for the 15 DC elements should be all lower-case

  <dc:title>Dublin Core in XML</dc:title>

  rather than

  <dc:Title>Dublin Core in XML</dc:Title>
**Recommendation 5**

- Multiple *property values* should be encoded by repeating the XML element for that *property*.

  `<dc:title>First title</dc:title>`
  `<dc:title>Second title</dc:title>`

---

**Recommendation 6**

- *Element refinements* should be treated in the same way as other *properties*.

  `<dcterms:available>2002-06</dcterms:available>`
  
rather than
  `<dc:date refinement="available">2002-06</dc:date>`
  or
  `<dc:date type="available">2002-06</dc:date>`
  or
  `<dc:date>
    <dcterms:available>2002-06</dcterms:available>
  </dc:date>`
Recommendation 7

- *Encoding schemes* should be implemented using the 'xsi:type' attribute of the XML element for the property

  <dc:identifier xsi:type="dcterms:URI">
  http://www.ukoln.ac.uk/
  </dc:identifier>

Recommendation 8

- *Element refinements and encoding schemes* should use the names specified in the DC Qualifiers recommendation

  <dcterms:isPartOf xsi:type="dcterms:URI">
  http://www.bbc.co.uk/
  </dcterms:isPartOf>

  <dcterms:temporal xsi:type="dcterms:Period">
  name=The Great Depression; start=1929; end=1939;
  </dcterms:temporal>
Recommendation 9

- Where the language of the value is indicated, it should be encoded using the 'xml:lang' attribute.

  <dc:subject xml:lang="en">seafood</dc:subject>
  <dc:subject xml:lang="fr">fruits de mer</dc:subject>

Example DC Record

```xml
<?xml version="1.0"?>
<metadata xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:dcterms="http://purl.org/dc/terms/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="schema.xsd">
  <dc:title>UKOLN</dc:title>
  <dcterms:alternative>
    UK Office for Library and Information Networking
  </dcterms:alternative>
  <dc:subject xsi:type="dcterms:DDC">062</dc:subject>
  <dc:publisher>UKOLN, University of Bath</dc:publisher>
  <dcterms:isPartOf xsi:type="dcterms:URI">
    http://www.bath.ac.uk
  </dcterms:isPartOf>
  <dc:identifier>http://www.ukoln.ac.uk/</dc:identifier>
</metadata>
```
<closing>
Thank You!!!
</closing>

<email>jschae@incheon.ac.kr</email>