XML, DTD and XML Schema

Charlie Abela
Department of Artificial Intelligence
charlie.abela@um.edu.mt
Last Lecture

- Limitations of the syntactic Web
  - Information retrieval, extraction and integration

- Semantic Web
  - Objectives
    - Machine processable information to ease information seeking tasks
  - Technologies
    - Metadata
    - Ontologies
    - Logic
    - Software agents
Lecture Outline

- Introduction
- XML: eXtensible Markup Language
- DTD: Document Type Definition
- XML Schema
- Exercise
- Suggested Reading list

Note: slides are based on Nicola Henze, Daniel Krause: Semantic Web available at http://www.kbs.uni-hannover.de/Lehre/semweb06/index.xml and on slides related to the Semantic Web Primer textbook
What is XML?

- XML is not a tag set (like HTML), but a meta language
- Enables users to define their own language
- Tags not concerned with how to render data, but instead define content
  - (HTML) `<b>` indicates that text is bold
  - (XML) `<employee>` indicates that data is about an employee
- **An agreed-upon textual format for representing tree-structured data**
- Definition is either implicit (deduced from document structure) or is explicit (defined in Document Type Definition or DTD)
<html>
<head> </head>
<body>
<h2>Student List</h2>
<ul>
  <li>9906789</li>
  <li>Adam</li>
  <li>adam@um.edu.mt</li>
  <li>yes - final</li>
</ul>
<ul>
  <li>9806791</li>
  <li>Adrian</li>
  <li>adrian@um.edu.mt</li>
  <li>no</li>
</ul>
</body>
</html>

<?xml version = "1.0"?>
@student_list>
  <student>
    <id>9906789</id>
    <name>Adam</name>
    <email>adam@um.edu.mt</email>
    <bsc>yes</bsc>
  </student>
  <student>
    <id>9806791</id>
    <name>Adrian</name>
    <email>adrian@um.edu.mt</email>
    <bsc>no</bsc>
  </student>
</student_list>

Specifies presentation  
Specifies data structure
An XML document contains

- A prolog:
  - an XML declaration and
  - an optional reference to external structuring documents

- A number of elements:
  - The “things” the XML document talks about
  - May contain Attributes
<?xml version="1.0" encoding="UTF-16"?>
<!DOCTYPE email SYSTEM "email.dtd">
<email>
  <head>
    <from name="Charlie Abela" address="charlie.abela@um.edu.mt"/>
    <to name="CSAI-4" address="csai-4@cs.um.edu.mt"/>
    <subject>CSA4110 Assignment</subject>
  </head>
  <body>
    Assignment proposal will be uploaded on the course site, in due time
  </body>
</email>
XML Prolog

- XML declaration `<xml version="1.0"?>`
- optional: character encoding
  `<xml version="1.0" encoding="ISO-8859-1"?>`
  possible encodings: UTF-8, UTF-16 (UTF: Unicode Transformation Format), UTF-16 is the ISO standard
- optional: whether the document is self-contained or refers to external structuring documents
  `<xml version="1.0" encoding="ISO-8859-1" standalone="no"?>`
- such a reference looks e.g. like
  `<!DOCTYPE email SYSTEM "email.dtd">`
  - structure is defined in the document type definition-file "email.dtd",
  - SYSTEM : only locally recognized name / file
  - otherwise: PUBLIC
XML Elements

- main building blocks of the XML document
- elements consist of a **begin-tag**, **content**, **end-tag**
  e.g. `<subject>CSA3210</subject>`
- some restrictions on naming tags:
  - **first character**: letter | underscore | colon
    e.g. `myname`, `_myname`, `:myname`
  - no element may begin with `[x|X][m|M][l|L]`
  - xml is case sensitive
- content of an element: `[text | other elements ]⁺ | nothing`
  `<author>Jim Born
    <date-of-birth> 28. August 1958 </date-of-birth>
    <place-of-birth> Valletta </place-of-birth>
  </author>`
- empty elements `<author></author>` can be abbreviated as `<author/>`
XML Attributes

- Express properties of elements.
- An attribute is a name-value pair inside the opening tag of an element:
  <author name="Jim Born" date-of-birth="28 August 1958"/>

- Attributes vs. elements:
  - attributes: properties of an element
  - elements can be nested, attributes not
  - order of elements is important but order of attributes is not
More XML

- Comments
  - A piece of text that is to be ignored by parser
    <!-- This is a comment -->

- Processing Instructions (PIs)
  - Define procedural attachments
    <?stylesheet type="text/css" href="mystyle.css"?>
XML well-formed

- An XML document is well-formed if it is syntactically correct
  - only one root element is possible
  - each element consists of an opening and a corresponding closing tag, however abbreviations possible e.g. `<author/>`
  - tags may not overlap, correct nesting:
    - for example this is not correct:
      ```xml
      <author>Jim Brown
      <date-of-birth> 28. August 1949 </author>
      </date-of-birth>
      ```
  - attributes of an element have to be unique
  - naming restrictions for elements
XML Tree Model

```
Root
  email
    head
      from
        name: charlie
        address: abela@um.edu.mt
      to
        name: CSAl-4
        address: csai-4@cs.um.edu.mt
    subject
      Assignment proposal will be uploaded on the course site, in due time
```

Assignment proposal will be uploaded on the course site, in due time.
The tree representation of an XML document is an ordered labeled tree:

- There is exactly one root
- There are no cycles
- Each non-root node has exactly one parent
- Each node has a label.
- The order of elements is important but the order of attributes is not
Valid XML

- An XML file is considered as valid if
  - it is well-formed
  - uses and abides to **structuring information**, defined in either a:
    - DTD: Document Type Definition
    - XML Schema
Structuring Information

- Structuring documents is important
  - Define all the element and attribute names that may be used
  - Define the structure
    - what values an attribute may take
    - which elements may or must occur within other elements, etc.

- If such structuring information exists, the document can be validated
Let’s consider an XML excerpt:

```xml
<lecturer>
  <name>Jim Brown</name>
  <room>230, Computer Building</room>
</lecturer>
```

DTD for above element (and all lecturer elements):

```xml
<!ELEMENT lecturer (name, room)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT room (#PCDATA)>
```
The element types **lecturer**, **name**, and **room** may be used in the document.

A **lecturer** element contains a **name** element and a **room** element, in that order (**sequence**).

A **name** element and a **room** element may have any content.

In DTDs, **#PCDATA** (parsed character data) is the only **atomic type** for element.
Adding disjunction

- Disjunction can be defined between element components

  ```xml
  <!Element contact (email | phone)>
  ```

- This defines a `contact` element to contain *either* an `email` element *or* a `phone` element

- One can also specify whether order between these elements is important or not. Any idea?

  ```xml
  <!Element lecturer ((name, phone) | (phone, name))>
  ```
Example of an XML Element/DTD

```xml
<order orderNo="_23456" customer="John Smith" date="October 15, 2002">
  <item itemNo="a528" quantity="1"/>
  <item itemNo="c817" quantity="3"/>
</order>

<!ELEMENT order (item+)>
<!ATTLIST order
  orderNo ID #REQUIRED
  customer CDATA #REQUIRED
  date CDATA #REQUIRED>

<!ELEMENT item EMPTY>
<!ATTLIST item
  itemNo ID #REQUIRED
  quantity CDATA #REQUIRED
  comments CDATA #IMPLIED>
```
Comments on the DTD

- The `item` element type is defined to be empty
- `+` (after `item`) is a **cardinality operator**:
  - `?`: appears zero times or once
  - `*`: appears zero or more times
  - `+`: appears one or more times
  - No cardinality operator means exactly once
Comments on the DTD (2)

- In addition to defining elements, we define attributes.
- This is done in an **attribute list** containing:
  - Name of the element type to which the list applies.
  - A list of triplets of **attribute name**, **attribute type**, and **value type**.
- **Attribute name**: A name that may be used in an XML document using a DTD.
DTD: Attribute Types

- Similar to predefined data types, but limited selection
- The most important types are
  - **CDATA**, a string (sequence of characters)
  - **ID**, a name that is unique across the entire XML document
  - **IDREF**, a reference to another element with an ID attribute carrying the same value as the IDREF attribute
  - **IDREFS**, a series of IDREFs
  - **(v1)| . . . |(vn)**, an enumeration of all possible values
- Limitations: no dates, number ranges etc.
DTD: Attribute Value Types

- **#REQUIRED**
  - Attribute must appear in every occurrence of the element type in the XML document

- **#IMPLIED**
  - The appearance of the attribute is optional

- **#FIXED "value"**
  - Every element must have this attribute

- **"value"**
  - This specifies the default value for the attribute
Referencing with IDREF and IDREFS

<!ELEMENT family (person+)>
<!ELEMENT person (name)>
<!ELEMENT name (#PCDATA)>
<!ATTLIST person id ID #REQUIRED
mother IDREF #IMPLIED
father IDREF #IMPLIED
children IDREFS #IMPLIED>
An XML Document Respecting the DTD

```
<family>
  <person id="bob" mother="mary" father="peter">
    <name>Bob Marley</name>
  </person>
  <person id="bridget" mother="mary">
    <name>Bridget Jones</name>
  </person>
  <person id="mary" children="bob bridget">
    <name>Mary Poppins</name>
  </person>
  <person id="peter" children="bob">
    <name>Peter Marley</name>
  </person>
</family>
```
DTD for the Email Example

<!ELEMENT email (head, body)>  
<!ELEMENT head (from, to+, cc*, subject)>  
<!ATTLIST from   name        CDATA #IMPLIED
                        address       CDATA #REQUIRED>  
<!ATTLIST to     name       CDATA #IMPLIED
                        address       CDATA #REQUIRED>  
<!ATTLIST cc     name       CDATA #IMPLIED
                        address       CDATA #REQUIRED>  
<!ELEMENT subject (#PCDATA)>  
<!ELEMENT body (text, attachment*)>  
<!ELEMENT text (#PCDATA)>
XML Schema

- Significantly richer language for defining the structure of XML documents
- Its syntax is based on XML itself
  - not necessary to write separate tools
- Reuse and refinement of schemas
  - Expand or delete already existent schemas
- Sophisticated set of data types, compared to DTDs (which only supports strings)
An XML schema is an element with an opening tag like

```xml
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2000/10/XMLSchema">
```

Structure of schema elements

- Element and attribute types using data types
Data Types

- There is a variety of **built-in data types**
  - Numerical data types: `integer`, `decimal` etc.
  - String types: `string`, `ID`, `IDREF`, `CDATA` etc.
  - Date and time data types: `time`, `Month` etc.
- There are also **user-defined data types**
  - *simple data types*, which cannot use other elements or attributes
  - *complex data types*, which can use a combination of these
Element Types

- A simple element is an XML element that can contain only text. It cannot contain any other elements or attributes.

\[
<\text{xs:element name="email" type="xs:string"/>}
<\text{xs:element name="head" type="xs:string"/>}
<\text{xs:element name="to" type="xs:string"/>}
\]
Attribute Types

- Attributes are optional by default. To specify that the attribute is required, use the "use" attribute:
  
  ```xml
  <xs:attribute name="id" type="ID" use="required"/>
  ```

- Default or Fixed value:
  - **Default** assigns a value automatically to the attribute when no other value is specified.
  - **Fixed** also assigns a value automatically to the attribute, but it's not possible to specify another value.

  ```xml
  <xs:attribute name="speaks" type="Language" default="en"/>
  ```
Restrictions

- XML elements or attributes with a defined data type, can have restrictions or facets on the element's or attribute's content.

- Cardinality constraints:
  - `minOccurs="x"` (default value 1)
  - `maxOccurs="x"` (default value 1)
  - Generalizations of *, ?, + offered by DTDs

```xml
<xsd:element name="email" type="xsd:string"/>
<xsd:element name="head" minOccurs="1" maxOccurs="1"/>
<xsd:element name="to" type="xsd:string" minOccurs="1"/>
```
Complex Data Types

- **Complex data types** are defined from already existing data types by defining some attributes (if any) and using:
  - `sequence`, a sequence of existing data type elements (order is important)
  - `all`, a collection of elements that must appear (order is not important)
  - `choice`, a collection of elements, of which one will be chosen
Complex Data Type Example

- Defines a lecturer type which has an optional title attribute and two string elements, firstname and lastname.

```xml
<xs:complexType name="lecturerType">
  <xs:sequence>
    <xs:element name="firstname" type="xs:string"
      minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="lastname" type="xs:string"/>
  </xs:sequence>
  <xs:attribute name="title" type="xs:string" use="optional"/>
</xs:complexType>
```
Data Type Extension

Already existing data types can be extended by new elements or attributes. Example:

```xml
<complexType name="extendedLecturerType">
  <extension base="lecturerType">
    <sequence>
      <element name="email" type="string" minOccurs="0" maxOccurs="1"/>
    </sequence>
    <attribute name="rank" type="string" use="required"/>
  </extension>
</complexType>
```
Resulting Data Type

```xml
<complexType name="extendedLecturerType">
  <sequence>
    <element name="firstname" type="string"
             minOccurs="0" maxOccurs="unbounded"/>
    <element name="lastname" type="string"/>
    <element name="email" type="string"
             minOccurs="0" maxOccurs="1"/>
  </sequence>
  <attribute name="title" type="string" use="optional"/>
  <attribute name="rank" type="string" use="required"/>
</complexType>
```
Data Type Extension (2)

- A **hierarchical relationship** exists between the original and the extended type
  - Instances of the extended type are also instances of the original type
  - They may contain additional information, but neither less information, nor information of the wrong type
Data Type Restriction

- An existing data type may be restricted by adding constraints on certain values
- Restriction is not the opposite of extension
  - Restriction is not achieved by deleting elements or attributes
- The following hierarchical relationship still holds:
  - Instances of the restricted type are also instances of the original type
  - They satisfy at least the constraints of the original type
Example of Data Type Restriction

```xml
<complexType name="restrictedLecturerType">
  <restriction base="lecturerType">
    <sequence>
      <element name="firstname" type="string" minOccurs="1" maxOccurs="2"/>
    </sequence>
    <attribute name="title" type="string" use="required"/>
  </restriction>
</complexType>
```
Restriction of Simple Data Types

```xml
<simpleType name="dayOfMonth">
  <restriction base="integer">
    <minInclusive value="1"/>
    <maxInclusive value="31"/>
  </restriction>
</simpleType>
```
Data Type Restriction: Enumeration

```
<simpleType name="dayOfWeek">
    <restriction base="string">
        <enumeration value="Mon"/>
        <enumeration value="Tue"/>
        <enumeration value="Wed"/>
        <enumeration value="Thu"/>
        <enumeration value="Fri"/>
        <enumeration value="Sat"/>
        <enumeration value="Sun"/>
    </restriction>
</simpleType>
```
<schema xmlns=http://www.w3.org/2000/10/XMLSchema version="1.0">
  <element name="email" type="emailType"/>
  <complexType name="emailType">
    <sequence>
      <element name="head" type="headType"/>
      <element name="body" type="bodyType"/>
    </sequence>
  </complexType>
</schema>

<complexType name="headType">
  <sequence>
    <element name="from" type="nameAddress"/>
    <element name="to" type="nameAddress" minOccurs="1"/>
    <element name="cc" type="nameAddress" minOccurs="0"/>
    <element name="subject" type="string"/>
  </sequence>
</complexType>
DTDs VS XML Schema

XML Schema
- More readable: since based on XML
- Reuse of technology: reuse/refine schemas
- No need for separate parsers
- Allows user to define types by extending and/or restricting existing ones
- Provides a set of data types that can be used in XML docs (DTDs limited to strings)
Namespaces

- An XML document may use more than one DTD or schema. To avoid name clashes
- The solution is to use a different prefix for each DTD or schema
  - prefix:name
- Namespaces are declared within an element and can be used in that element and any of its children (elements and attributes)
- A namespace declaration has the form:
  - xmlns:prefix="location"
  - location is the address of the DTD or schema
- If a prefix is not specified: xmlns="location" then the location is used by default
Example

```xml
<?xml version="1.0" encoding="UTF-16"?>
<vu:instructors
    xmlns:vu="http://www.vu.com/empschema"
    xmlns:gu="http://www.gu.au/empschema"
    xmlns:uky="http://www.uky.edu/empschema">
  <uky:faculty uky:title="assistant professor"
               uky:name="John Smith"
               uky:department="Computer Science"/>
  <gu:academicStaff
    gu:title="lecturer"
    gu:name="Mate Jones"
    gu:school="Information Technology"/>
</vu:instructors>
```
Exercise

- A person’s profile can be described through a vcard. A typical description may contain:
  - an id: referring to the vcard id
  - one title e.g. Dr. or Mr.
  - one role e.g. lecturer
  - one name: first, surname
  - one or more contact telephone numbers: each consists of a number and description (e.g. work)
  - zero or more email addresses: each consists of address
  - zero or more contact addresses: each consists of street, locality, post code and country
  - one or more office details: each consists of a room, department, building

- Create an XML representation for this vcard. Create also a DTD and an XML schema for the created vcard XML representation
Suggested Reading

- Textbook: Semantic Web Primer Chapter 2
- Extensible Markup Language, T.Bray, J.Paoli, http://www.w3.org/TR/REC-xml
- XML tutorial, http://www.w3schools.com/xml/default.asp
- DTD tutorial, http://www.w3schools.com/dtd/default.asp
Next lecture

- Introduction to XPath and XQuery
  - languages that allow for node selection and extraction from an XML document
Extra XML Example

```xml
<?xml version="1.0" standalone="no"?>
<!DOCTYPE Bookstore SYSTEM "bookstore.dtd">
<Bookstore>

    <Title>A First Course in Database Systems</Title>
</Book>

    <Title>Database Systems: The Complete Book</Title>
    <Remark>
        Amazon.com says: Buy this book bundled with
        It's a great deal!
    </Remark>
</Book>

<Author Ident="HG">
    <First_Name>Hector</First_Name>
    <Last_Name>Garcia-Molina</Last_Name>
</Author>

<Author Ident="JU">
    <First_Name>Jeffrey</First_Name>
    <Last_Name>Ullman</Last_Name>
</Author>

<Author Ident="JW">
    <First_Name>Jennifer</First_Name>
    <Last_Name>Widom</Last_Name>
</Author>

</Bookstore>
```
Extra DTD Example

<!ELEMENT Bookstore (Book*, Author*)>
<!ELEMENT Book (Title, Remark?)>
<!ATTLIST Book ISBN ID #REQUIRED
  Price CDATA #REQUIRED
  Edition CDATA #IMPLIED
  Authors IDREFS #REQUIRED>
<!ELEMENT Title (#PCDATA)>
<!ELEMENT Remark (#PCDATA | BookRef)>
<!ELEMENT BookRef EMPTY>
<!ATTLIST BookRef book IDREF #REQUIRED>
<!ELEMENT Author (First_Name, Last_Name)>
<!ATTLIST Author Ident ID #REQUIRED>
<!ELEMENT First_Name (#PCDATA)>
<!ELEMENT Last_Name (#PCDATA)>