Markup Languages & XML

Grid Computing (M)
Lecture 5
Olufemi Komolafe (femi@dcs.gla.ac.uk)
18 January 2006

UNIVERSITY of GLASGOW

Overview

- XML Background
  - History & evolution
- XML Document Structure
  - Well-formed XML documents
  - Valid XML documents
- Using XML
  - "Friends" of XML

XML Background

History & Evolution

XML History

- Markup language
  - Combines text and extra information about text
  - Extra information expressed using markup
- SGML (Standard Generalized Markup Language)
  - ISO Standard (ISO 8879:1986) for data storage & exchange
  - Meta-language for defining languages (through Document Type Definitions and Schemas)
  - HTML: famous SGML language
  - Separation of content & display
  - Verbose: standards 600+ pages long

XML History

Mid 1990s

- SGML experts convinced markup for growing web inadequately supported
  - SGML too complex
  - HTML too limited

1996

- W3C launched "SGML on the web activity"
  - XML = "e**X**tensible **M**arkup **L**anguage"
  - XML working group launched
    - ***Is stripped down version of SGML optimised for web possible?***

1998

- XML 1.0 becomes W3C Recommendation
  - XML Version 1.0 2nd Edition (2000)
  - XML Version 1.0 3rd Edition (2004)

XML Original Design Goals

http://www.w3.org/TR/1998/REC-xml-19980210

1. XML shall be straightforwardly usable over the Internet.
2. XML shall support a wide variety of applications.
3. XML shall be compatible with SGML.
4. It shall be easy to write programs which process XML documents.
5. The number of optional features in XML is to be kept to the absolute minimum, ideally zero.
6. XML documents should be human-legible and reasonably clear.
7. The XML design should be prepared quickly.
8. The design of XML shall be formal and concise.
9. XML documents shall be easy to create.
10. Terseness in XML markup is of minimal importance.

XML Standardisation Process

World Wide Web Consortium (W3C) *http://www.w3.org/*

- 350+ members (major vendors etc.)
- Controls XML standards (as well as HTML, CSS etc.)
- Work needed organised into related activities
  - working groups
- Key stages
  1. Working Draft
     - no consensus yet
  2. Candidate Recommendation
     - soliciting implementation experience & feedback
  3. Proposed Recommendation
     - sent for final approval by advisory committee
  4. W3C Recommendation
     - approved by W3C

XML & HTML

HTML	XML
Designed to ***display*** data & focus on how data ***looks***	Designed to ***describe*** data & focus on what data ***is***
About ***displaying*** information	About ***describing*** information
Predefined tags (e.g.<p>,<h1>)	Developers define own tags
Tags mix format and structure	Tags define only structure
Sloppy "syntax" often OK	Strict "syntax"

Some uses of XML

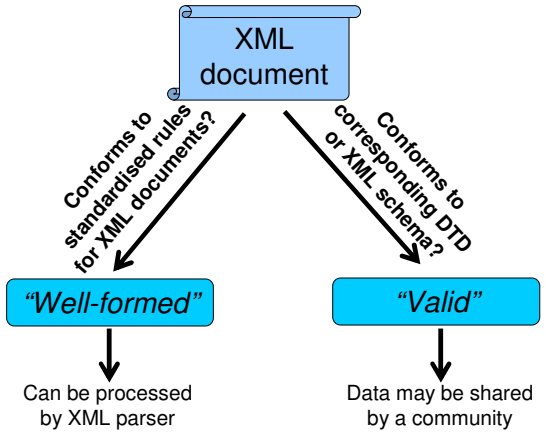
- Exchanging data
  - Data may be exchanged between incompatible systems by converting to XML
- Storing data
  - Data stored in files or databases
  - XML databases & XML-based query languages exist
- Describing meta-data
  - Information about structure & meaning of data
  - Enables more intelligent web searches
- Supporting operations invocations
  - Information received in XML document parsed & used to invoke services

XML Document Structure

Well-formed XML Documents

Valid XML Documents

Well-formed & Valid XML Documents



Exemplar Well-formed XML Document

```
<? xml version = "1.0" encoding = "UTF-8" ?>
<memo>
    <to> Femi </to>
    <from> Colin </from>
    <heading> Reminder </heading>
    <content>
        <message> Remember you're giving the GC lecture today </message>
    </content>
</memo>
```

Remember, nothing special about XML….

Just plain text with some XML tags enclosed in angle brackets

BUT when combined with software to process tags & contents, numerous uses

Exemplar XML Document

```
<? xml version = "1.0" encoding = "UTF-8" ?>
<memo>
    <to> Femi </to>
    <from> Colin </from>
    <heading> Reminder </heading>
    <content>
        <message> Remember you're giving the GC lecture today </message>
    </content>
</memo>
```

Root element

XML declaration (version, character encoding)

Child element

End of root element

Some Basic XML Rules

```
<? xml version = "1.0" encoding = "UTF-8" ?>
<memo>
    <to> Femi </to>
    <from> Colin </from>
    <heading> Reminder </heading>
    <content>
        <message> Remember you're giving the GC lecture today </message>
    </content>
</memo>
```

Single root element

All elements must have closing tag

Tags are case sensitive e.g. </From> incorrect

Proper nesting of tags mandatory e.g. <content><message>…</content></message> incorrect

Some Basic XML Rules

```
<? xml version = "1.0" encoding = "UTF-8" ?>
<memo date = "18/1/2005">
    <to> Femi </to>
    <from> Colin </from>
    <heading> Reminder </heading>
    <content>
        <message/>
        <!-- This is a comment -->
    </content>
</memo>
```

Attributes may be used to give extra information
•quotation marks mandatory

Empty element
i.e. <message/> == <message></message>

XML syntax for comments similar to HTML

Badly-formed XML Document: Spot the Errors?

```
<? xml version = "1.0" encoding = "UTF-8" ?>
<overratedbands>
    <band>
        <name> Franz Ferdinand </name>
        <album releasedate = "4/10/2005" tracks = "14">
            <name>You could have it so much better <name>
        </album>
    </band>
    <band>
        <name> Oasis </name>
        <album releasedate = 31/5/2005>
            <NAME>Don't believe the truth </name>
        </band>
    </album>
<overratedBands>
```

Elements vs Attributes?

Should document content be put in element OR attribute

<car><colour>red</colour></car>

<car colour="red" />

Provides logical structuring of information

Easier extensibility

Good if attribute itself has attributes

May simplify parsing

Provides description of characteristics of information

Good for default values

Reduces document size

Valid XML Documents

- Users define own tags
  - Users must agree on what tags mean to be able to exchange XML document meaningfully
- Valid structure must be defined
  - Defines legal grammars/format of content
  - Documents checked against this pre-defined structure
    - If compliant, document said to be "valid"
- Two alternative approaches
  - Document Type Definition (DTD)
  - XML Schemas

Document Type Definition

- Pre-defined structure that facilitates data portability
  - Defines correct document structure
- Appears either at top of file or separately
- Describes
  - Permitted elements
    - *Element X composed of*
      - *Exactly one Y1*
      - *One or more Y2s*
      - *Optionally a Y3*
      - *Any number of Y4s*
      - *Either a Y5 or Y6*
  - Structural relationships between elements
    - *Element Y1 is composed of a Z1 or a Z2*
  - Attributes of elements
    - *Element Z1 is a text string*

Different communities creating relevant DTDs

WML: for Wireless Application Protocol

MathML

Chemical Markup Language

…….

DTD Syntax

<!keyword keyword-name keyword-name-descriptor>

keyword	Comment
ATTLIST	Defines one or more attributes of a given element
DOCTYPE	Consists of internal DTD or references to external DTD
ENTITY	Defines entity to be used either in document itself or elsewhere in DTD
ELEMENT	Defines an element that may be part of document's element tree
NOTATION	Defines document content that is outside scope of XML standard

Defining Elements in a DTD

Template into which content of all occurrences of element will fit

Specifies element "type": pure data, compound type etc

<!ELEMENT element-name content-model>

EMPTY, #PCDATA, ANY *or children defined as* ***model group***

Examples

<!ELEMENT junk ANY> ⇨anything can appear in element called *junk*; not checked by parser

<!ELEMENT Text #PCDATA> ⇨ parsed character data; to be handled normally by parser

<!ELEMENT foo EMPTY> ⇨there is no content, *foo* is empty

Defining Elements in a DTD

<!ELEMENT element-name content-model>

EMPTY, #PCDATA, ANY *or children defined as* ***model group***

List of components

Components either another element type or #PCDATA

Components may be
- **sequence (comp1, comp2, comp3)**
- **alternatives (comp1 | comp2 | comp3)**

Components labelled
? ⇨ 0 or 1 occurrence
+ ⇨ 1 or more occurrences
*** ⇨ 0 or more occurrences**
unlabelled ⇨ exactly 1 occurrence

Examples

<!ELEMENT Form (Heading, Field+, Signature)>

<!ELEMENT Field (Textbox | Checkbox | Menu)>

<!ELEMENT Heading (#PCDATA)>

Some DTD & XML Examples

DTD	XML
<!ELEMENT Message EMPTY>	<Message/>
<!ELEMENT name (#PCDATA)>	<name>Femi</name>
<!ELEMENT Car(Make,Model,Age?)> <!ELEMENT Make (#PCDATA)> <!ELEMENT Model (#PCDATA)> <!ELEMENT Age (#PCDATA)>	<Car> <Make> Audi </Make> <Model>TT</Model> </Car>
<!ELEMENT Winner (Labour|Tory)> <!ELEMENT Labour (#PCDATA)> <!ELEMENT Tory (#PCDATA)>	<Winner><Labour>Brown</Labour></Winner> **OR** <Winner><Tory>Cameron</Tory></Winner>

Defining Attributes in a DTD

- Element can also have attributes defined
- Attributes for each element type declared separately

  <!ATTLIST element-name attribute-name type modifier>
- Each attribute declared as ***name, type*** & ***modifier***
  - ***modifier:*** constrains value expected for attribute
    - #REQUIRED ⇨no default so value must be supplied with all instances of element type
    - #IMPLIED ⇨no default but attribute is optional
    - #FIXED ⇨only value an element of this type can take
    - defaultValue ⇨ assumed if no value given for particular element

Attribute Types

Attribute type determines what will appear as attribute values

Attribute Types	Comment
CDATA	A character string
ID	Unique for each element (c.f. a primary key)
IDREF	Refers to an ID of some other element (c.f. a foreign key)
IDREFS	Refers to one or more IDs
ENTITY	Refers to an entity
ENTITIES	Refers to one or more entity
NOTATION	A notation
NMTOKEN	A name treated as a token by a parser
NMTOKENS	A series of names
(enumerated list)	Contains only values drawn from that list

Example: DTD

MEMO contains exactly one ***TO***, ***FROM***, ***SUBJECT***, ***BODY*** & ***SIGN*** element

MEMO has an attribute ***importance*** which can be ***HIGH***, ***MEDIUM*** or ***LOW*** (default)

```
<!DOCTYPE MEMO[
  <!ELEMENT MEMO (TO,FROM,SUBJECT,BODY,SIGN)>
  <!ATTLIST MEMO importance ("HIGH"|"MEDIUM"|"LOW") "LOW">
  <!ELEMENT TO (#PCDATA)>
  <!ELEMENT FROM (#PCDATA)>
  <!ELEMENT SUBJECT (#PCDATA)>
  <!ELEMENT BODY (P+)>
  <!ELEMENT P (#PCDATA)>
  <!ELEMENT SIGN (#PCDATA)>
  <!ATTLIST SIGN DoB CDATA #IMPLIED email CDATA #REQUIRED>
]>
```

TO, ***FROM*** & ***SUBJECT*** are parsed character data

BODY contains 1 or more ***P*** elements

P & ***SIGN*** are parsed character data

SIGN has an optional attribute ***DoB*** & a mandatory attribute ***email***; both are character data

Example: Corresponding Valid XML

```
<MEMO importance="HIGH">
  <TO>Joe Bloggs</TO>
  <FROM>Jane Doe</FROM>
  <SUBJECT>Request</SUBJECT>
  <BODY>
      <P>Can you please lend me some money?</P>
      <P> Will pay you back ASAP </P>
      <P> This is a really uninteresting memo, isn't  it?! </P>
      <P> In that case, goodbye </P>
  </BODY>
  <SIGN email="a@b.com">JD</SIGN>
</MEMO>
```

Using DTDs

Internal DTD

```
<?xml version="1.0"?>
<!DOCTYPE note [
    <!ELEMENT note
    (to,from,heading,body)>
    <!ELEMENT to (#PCDATA)>
    <!ELEMENT from (#PCDATA)>
    <!ELEMENT heading (#PCDATA)>
    <!ELEMENT body (#PCDATA)>
]>
<note>
    <to>Jane</to>
    <from>Gail</from>
    <heading>Reminder</heading>
    <body> blah blah blah …</body>
</note>
```

External DTD

```
<?xml version="1.0"?>
<!ELEMENT note (to,from,heading,body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
```

note.dtd

```
<?xml version="1.0"?>
<!DOCTYPE note SYSTEM "note.dtd">
<note>
    <to>Jane</to>
    <from>Gail</from>
    <heading>Reminder</heading>
    <body> blah blah blah …</body>
</note>
```

<!DOCTYPE root-element SYSTEM "filename">

DTD vs. XML Schema

Advantages offered by XML Schema

- Support for data types
- Support for namespaces
- XML document
- Greater descriptive detail
- Newer
- Extensible
  - to import elements from other schemas,
  - reference multiple schemas from same document
  - derive own data types from standard types

BUT

- XML Schema definition more verbose & complex than DTDs
- Being an XML document, structure of the XML Schema document must be defined
  - Ironically DTD used

XML Schema

- Official W3C recommendation (May 2001)
- XML Schemas are successors of DTDs
- Describe & constrain content of XML document
- Defines
  - elements that can appear in a document
  - attributes that can appear in a document
  - which elements are child elements
  - the order of child elements
  - the number of child elements
  - whether an element is empty or can include text
  - data types for elements & attributes
  - default & fixed values for elements & attributes

Namespaces

```
<album>
   <title>Leftism</title>
</album>
```

```
<song>
   <title>Melt</title>
</song>
```

```
<album>
   <title>Leftism</title>
   <tracks>
        <song>
              <title>Melt</title>
        </song>
   </tracks>
</album>
```

Conflict!

Parser cannot tell difference between ***title*** elements unless instructed

Namespaces

```
<album>
   <a:title xmlns:a="URI_album">Leftism</a:title>
   <tracks>
        <song>
              <s:title xmlns:s="URI_song">Melt</s:title>
        <song>
   </tracks>
</album>
```

*Informs parser that all elements prefixed with **s** belong to namespace associated with given URI for the **title** element and contents (essentially **s** is shorthand notation for "URI_SONG")*

xmlns:namespace_prefix="namespace"

URI (uniform resource identifier)

For namespace identification

Not looked up by parser: simply system for creating unique identifiers

Can be URL or uniform resource name, URN (urn:namespace:string)

Declaring Namespaces

Explicit declaration

```
<c:customer xmlns:c="URI_customer">
    <c:name> John</c:name>
    <c:product>
        <c:shirt>
            <c:size>Medium</c:size>
            <c:colour>Black</c:colour>
        </c:shirt>
    </c:product>
</c:customer>
```

Prefix mandatory for children to be associated with namespace

Normally used for external schema usage

Default declaration

```
<customer xmlns="URI_customer">
    <name>John</name>
    <product>
        <shirt>
            <size>Medium</size>
            <colour>Black</black>
        </shirt>
    </product>
</customer>
```

URI_customer default namespace for customer
- *this namespace associated with all children*
- *no need for prefixes within children*

Normally used for local schema usage

Using Namespaces

Can have elements from different namespaces within the same document

Multiple namespace declarations can be in scope simultaneously → convention is to put all namespace declarations in root element

```
<A xmlns:foo="http://foo.org" xmlns:bar="http://bar.org" xmlns="http://default.org">
    <foo:B> abcd</foo:B>
    <bar:C>efgh</bar:C>
    <foo:D>mnop</foo:D>
    <E>rstu</E>
</A>
```

In http://foo.org namespace

In http://bar.org namespace

No prefix → in default (http://default.org) namespace

Some Common Namespaces

Typical prefix	Comment
xsi	*http://www.w3.org/2001/XMLSchema-instance* Namespace governing XMLSchema instances
xsd	*http://www.w3.org/2001/XMLSchema* Namespace governing XMLSchema (*.xsd) files
wsdl	*http://www.schemas.xmlsoap.org/wsdl/* WSDL namespace
soap	*http://www.schemas.xmlsoap.org/wsdl/soap* WSDL SOAP binding namespace
tns	Conventionally refers to current XML document

XML Schema

<schema> element

- root of every XML Schema
- typically contains a number namespace declarations

Typical Schema Declaration

```
<?xml version="1.0"?>

<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"

targetNamespace="http://www.dcs.gla.ac.uk/DCSSchema"

xmlns="http://www.anotherURL.com" >

...

</xsd:schema>
```

Namespace being constrained by schema (i.e. elements defined by this schema come from http://www.dcs.gla.ac.uk/DCSSchema namespace)

Specifies prefix & namespace for schema elements & data types

Indicates default namespace (for elements without prefix)

Description of the valid XML document structure goes here

Defining Elements & Attributes

Defining Element

Name of element in XML document being constrained

e.g. minOccurs="x" maxOccurs="y" (default x=y=1)

<element name="*elementName*" type="*elementType*" *Options*/>

Type of element
- Predefined XML Schema data type?
- User defined type?

Defining Attribute

Name of attribute in XML document being constrained

e.g. can specify if mandatory (minOccurs="1"), default values etc.

<attribute name="*attributeName*" type="*attributeType*" *Options*/>

Same types available as for elements

XML Schema Data Types

Type	Purpose
string	Character strings
boolean	Binary valued logic (true or false)
float/double	32/64-bit floating type
decimal, integer	Standard decimal notation, positive & negative
timeInstant	Combination of date & time representing single instant of time
timeDuration	Duration of time
date, time	Specific time that recurs over timeDuration
binary	Binary data
uri	URI

XML Schema Data Types

Support for data types significant advantage w.r.t. DTD

Makes it easier to

- Describe permissible document content
  - <xsd:element name="age" type="xsd:integer"/>
  - Hence, <age>young</age> incorrect
- Validate the correctness of data
  - <xsd:element name="birthday" type="xsd:date"/>
  - Hence, <birthday>2006-10-12</birthday> unambiguous
    - Schema defines format for date as YYYY-MM-DD
- Define restrictions on data
  - <xsd:element name="car" type="xsd:string" fixed="BMW"/>
  - Hence, <car/> or <car>BMW</car> correct
  - but <car>Audi</car> incorrect
- Define data patterns/formats
  - <xsd:element name="car" type="xsd:string" default="BMW"/>
  - Hence, <car/> == <car>BMW</car>

XML Schema Example

```
<?xml version="1.0"?>
<book isbn="014404173">
   <title> War and Peace </title>
   <author>Leo Tolstoy</author>
   <character>
        <name>Pierre Buzekhov</name>
        <spouse>Helene</spouse>
        <personality>genial, aspirational, confused</personality>
        <DoB>1803-10-04</DoB>
   </character>
   <character>
        <name>Andrew Bolkonski</name>
        <spouse>Anna</spouse>
        <personality>patriotic, distant, genuine</personality>
        <DoB>1811-06-11</DoB>
   </character>
</book>
```

What is the XML schema for this document?

XML Schema Example

```
<?xml version="1.0" encoding="utf-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="book">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="title" type="xsd:string"/>
        <xsd:element name="author" type="xsd:string"/>
        <xsd:element name="character" minOccurs="0" maxOccurs="unbounded">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="name" type="xsd:string"/>
              <xsd:element name="spouse" type="xsd:string" minOccurs="0"
                                                  maxOccurs="unbounded"/>
              <xsd:element name="DoB" type="xsd:date"/>
              <xsd:element name="personality" type="xsd:string"/>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
      <xsd:attribute name="isbn" type="xsd:string"/>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

*To match start tag for **book** element*

book has attributes & non-text children ⇨ ***complexType***

title *of data type* ***string***

Specifying cardinality of an element

sequence *defines ordered sequence of sub-elements. (c.f.* ***choice*** *and* ***all****.)*

book *contains* ***title, author & character(s)***

character *contains* ***name, spouse(s), DoB & personality***

Attribute for ***book*** *, defined after elements*

Corresponding DTD?

Using XML

"Friends" of XML

XML APIs

Tree-based APIs

Map XML document into internal tree structure, navigable by application

W3C DOM WG maintains recommended tree-based API for XML & HTML documents

Event-based APIs

Report parsing events (e.g. start & end of elements) to application through callbacks

Application implements handlers to deal with the different events

DOM

Builds in-memory hierarchical model of the XML elements

Good if whole document needed
•manipulating data

SAX

Provides event-driven framework for parsing XML documents

Good for using parts of data
•simple extraction tasks

XLink, XPointer & XPath

- **Two major components to linking in XML**
  - **XLink**
    - XML Linking Language
    - a generalisation of the HTML link concept
    - Elements inserted into XML document to create & describe links between resources
    - XML syntax used to create structures
      - Like simple uni-directional hyperlinks
      - More sophisticated links
    - uses **XPointer** to locate resources
    - W3C Recommendation (June 2001)
  - **XPointer**
    - XML Pointer Language
    - Allows hyperlinks to point to specific parts of XML document
    - **XPath** expression used to navigate XML document
    - W3C Recommendation (March 2003)
- **XPath**
  - a declarative language for locating nodes and fragments in XML trees
  - important and widely used
    - used in **XPointer** (for addressing), **XSL** (for pattern matching), **XML Schema** (for uniqueness and scope descriptions), and **XQuery** (for selection and iteration)
  - W3C Recommendation (November 1999)

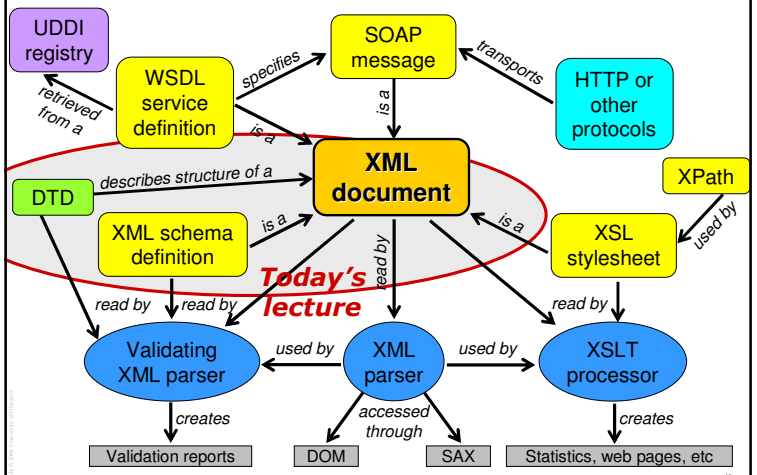
XSL

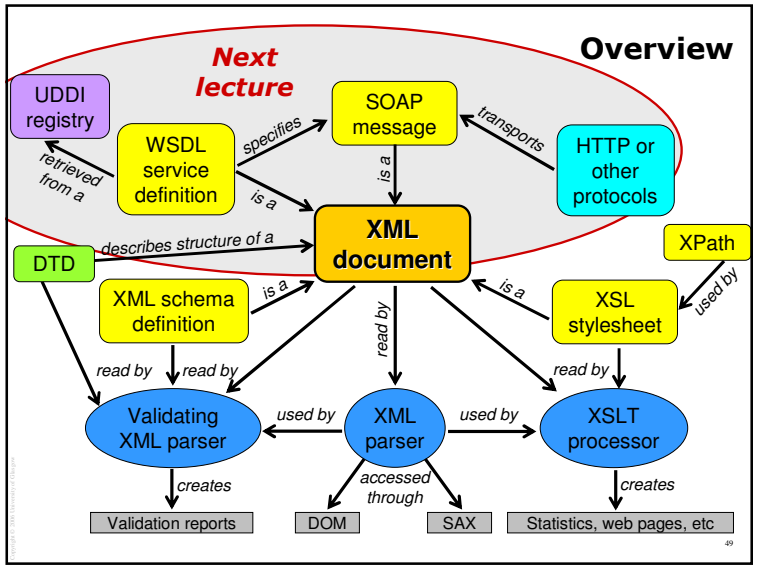
- eXtensible Stylesheet Language
- Developed by W3C due to need for an XML-based Stylesheet Language.
- Family of W3C recommendations defining XML document transformation and presentation
- 3 parts
  - XPath ⇨ for accessing/referencing parts of XML document
  - XSL-FO ⇨ for formatting XML documents
  - XSLT ⇨ for transforming XML documents
- W3C Recommendation (October 2001)

XSLT

- XSL Transformations
- Language for transforming an XML document
  - into another XML document, HTML, XHTML, ……
- Changes the structure not the data
- In transformation process
  - XSLT uses XPath to define parts of the source document that match one or more predefined templates.
  - When a match is found, XSLT will transform the matching part of the source document into the result document.
  - The parts of the source document that do not match a template will end up unmodified in the result document.
- W3C Recommendation (November 1999)

Overview




Overview
Next lecture
UDDI registry
WSDL service definition
SOAP message
HTTP or other protocols
retrieved from a
specifies
is a
transports
is a
XML document
DTD
describes structure of a
XPath
XML schema definition
is a
XSL stylesheet
is a
used by
read by
read by
read by
read by
Validating XML parser
used by
XML parser
used by
XSLT processor
creates
accessed through
creates
Validation reports
DOM
SAX
Statistics, web pages, etc
49