Introduction to XML

CS 317/387

Agenda – Introduction to XML

1. What is it?
2. What's it good for?
3. How does it work?
4. The infrastructure of XML
5. Using XML on the Web
6. Implementation issues & costs

2

1. What is it?

Discussion points:

- First principles: OHCO
- Example: A simple XML fragment
- Compare/contrast: SGML, HTML, XHTML
  - A different XML for every community
- Terminology

3

Ordered hierarchies of content objects

- Premise: A text is the sum of its component parts
  - A <Book> could be defined as containing: <FrontMatter>, <Chapter>s, <BackMatter>
  - <FrontMatter> could contain: <BookTitle> <Author>s <PubInfo>
  - A <Chapter> could contain: <ChapterTitle> <Paragraph>s
  - A <Paragraph> could contain: <Sentence>s or <Table>s or <Figure>s …
- Components chosen should reflect anticipated use

4

Ordered hierarchies of content objects

- OHCO is a useful, albeit imperfect, model
  - Exposes an object's intellectual structure
  - Supports reuse & abstraction of components
  - Better than a bit-mapped page image
  - Better than a model of text as a stream of characters plus formatting instructions
  - Data management system for document-like objects
  - Does not allow overlapping content objects
  - Incomplete; requires infrastructure

5

Content objects in a book

- **Book**
  - **FrontMatter**
    - **BookTitle**
    - **Author(s)**
    - **PubInfo**
  - **Chapter(s)**
    - **ChapterTitle**
    - **Paragraph(s)**
  - **BackMatter**
    - **References**
    - **Index**

6

Content objects in a catalog card

- **Card**
  - **CallNumber**
  - **MainEntry**
  - **TitleStatement**
    - **TitleProper**
    - **StatementOfResponsibility**
  - **Imprint**
  - **SummaryNote**
  - **AddedEntrySubject(s)**
  - **Added EntryPersonalName(s)**

7

Semistructured Data

- Another data model, based on trees.
- Motivation: flexible representation of data.
  - Often, data comes from multiple sources with differences in notation, meaning, etc.
- Motivation: sharing of *documents* among systems and databases.

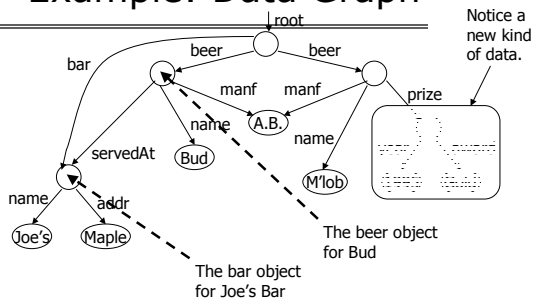
8

Graphs of Semistructured Data

- Nodes = objects.
- Labels on arcs (attributes, relationships).
- Atomic values at leaf nodes (nodes with no arcs out).
- Flexibility: no restriction on:
  - Labels out of a node.
  - Number of successors with a given label.

9

Example: Data Graph



XML

- XML = *Extensible Markup Language*.
- While HTML uses tags for formatting (e.g., "italic"), XML uses tags for semantics (e.g., "this is an address").
- Key idea: create tag sets for a domain (e.g., genomics), and translate all data into properly tagged XML documents.

A simple XML fragment

```
<Book>
  <FrontMatter>
    <BookTitle>XML Is Easy</BookTitle>
    <Author>Tim Cole</Author>
    <Author>Tom Habing</Author>
    <PubInfo>CDP Press, 2002</PubInfo>
  </FrontMatter>
  <Chapter>
    <ChapterTitle>First Was SGML</ChapterTitle>
    <Paragraph>Once upon a time …</Paragraph>
  </Chapter>
</Book>
```

This is NOT XML

```
<PoemFragment>
  <Stanza>
    <Line><Sentence>It was six men of
   Indostan</Line>
    <Line>To learning much inclined,</Line>
    <Line>Who went to see the Elephant</Line>
    <Line>(Though all of them were blind),</Line>
    <Line>That each by observation</Line>
    <Line>Might satisfy his
   mind.</Sentence></Line>
  </Stanza>
</PoemFragment>
```

13

XML comes from SGML

- Standard Generalized Markup Language
  - Based on IBM's GML (Goldfarb, et al.)
  - ISO standard since 1989
  - Used for large-scale document management (Boeing 747 user's manual)
  - Expensive, complex to implement
  - Not Web-friendly (no "well-formed" SGML)
  - Too many options (e.g., tag minimization)

14

XML, HTML, & XHTML

- HTML—display-oriented, SGML-based scheme for making Web pages
  - Syntax & allowed elements (semantics) are fixed
- XML—set of rules for defining markup schemes
  - Element set is fully extensible
  - Syntax is fixed
- XHTML—HTML modified to be XML-compliant (not just SGML-compliant)

15

Markup languages compared

- XML syntax is stricter than HTML or SGML
  - Must explicitly close all elements
  - Attributes must be enclosed in quotes
  - All markup is case-sensitive
- XML & SGML: no fixed tags, no predefined style
- XML & SGML are extensible
  - Fixed elements (HTML) vs. rules (XML, SGML)
  - HTML elements describe how to present content
  - XML elements can describe the content itself

16

A different XML for every community

- XML is a set of rules used for defining & encoding intellectual structures
- XML is extensible & customizable
  - Its greatest strength
  - Its greatest weakness
- HTML was invented by physicists
  - What if it had been lawyers, or teachers, or bureaucrats, or librarians, or …?

17

Terminology

- Document instance
- Document class
- Document Type Definition (DTD), or schema
- Well-formed XML
- Valid XML
- Stylesheets
- XML Transformations
- Document Object Model (DOM)

18

2. What's it good for?

- Smarter documents
- Full text
- Metadata
- Machine-to-machine interactions

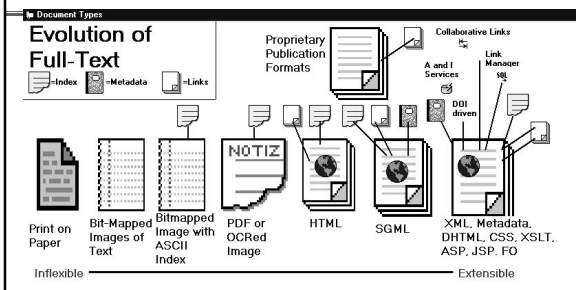
19

Smarter documents

- Standards-based
- Facilitates…
  - Search & discovery
    - Precise, field-specific searching
  - Interoperability & normalization
  - Complex transformations
  - Linking between and within texts
  - Reuse of documents and fragments

20

Smarter documents



21

Full text

- Electronic Text Center (U of VA Library)
  - Originally SGML, now also XML, eBooks
  - 70,000 texts; 350,000 related images
  - 37,000 visits to collection per day
  - http://etext.lib.virginia.edu/
- Open eBook Forum
  - International trade & standards organization
  - Goal: establish specs & stds for epublishing
  - http://www.openebook.org/

22

Using XML for full text

- No inherent presentation information
  - Requires…
    - CSS in XML-aware browsers, or
    - XSLT to transform to XHTML, or
    - XSL-FO to reformat for presentation
  - Techniques for including non-text content vary by application
- XML can be verbose
- Most standard full-text schemas are complex

23

Metadata

- XML schemas exist for a range of metadata standards
  - Encoded Archival Description (EAD)
  - MARC 21 XML (also MODS)
  - Metadata Encoding & Transmission Standard (METS)
  - Dublin Core Variants
    - Open Archives Initiative (OAI)
    - National Science Digital Library (NSDL)
  - Resource Description Framework (RDF)

24

Using XML for metadata

- Consistency in applying schema
  - Optional versus required elements
  - Consistent use of elements
  - Granularity & depth of information
- XML schemas still evolving
  - Attributes versus elements
  - Mixing namespaces
  - Schema languages
  - Philosophical issues

25

Machine-to-machine interactions

- Web services
  - Facilitating machine-to-machine communications via XML
  - Simple Object Access Protocol (SOAP)
  - XML Protocol Working Group
- Semantic Web
  - Abstract representation of data on the Web
- XML and Databases

26

3. How does it work?

In XML, there's content and there's markup.

- Markup
  - Elements
  - Attributes
  - Comments
  - Processing instructions
- Content
  - Entities
  - Encoded (Unicode) characters

27

Well-Formed and Valid XML

- *Well-Formed XML* allows you to invent your own tags.
  - Similar to labels in semistructured data.
- *Valid XML* involves a DTD (*Document Type Definition*), a grammar for tags.

28

Well-Formed XML

- Start the document with a *declaration*, surrounded by <?xml … ?> .
- Normal declaration is:

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

  - "Standalone" = "no DTD provided."
- Balance of document is a *root tag* surrounding nested tags.

29

Tags

- Tags, as in HTML, are normally matched pairs, as <FOO> … </FOO> .
- Tags may be nested arbitrarily.
- XML tags are case sensitive.

30

Example: Well-Formed XML

```
<?xml version = "1.0" standalone = "yes"?>
<BARS>
  <BAR><NAME>Joe's Bar</NAME>
    <BEER><NAME>Bud</NAME>
      <PRICE>2.50</PRICE></BEER>
    <BEER><NAME>Miller</NAME>
      <PRICE>3.00</PRICE></BEER></BAR>
  <BAR> …
</BARS>
```

A NAME subobject

A BEER subobject

31

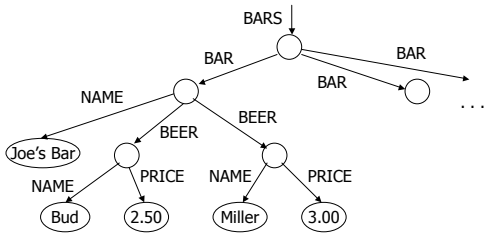
XML and Semistructured Data

- Well-Formed XML with nested tags is exactly the same idea as trees of semistructured data.
- We shall see that XML also enables nontree structures, as does the semistructured data model.

32

Example

- The <BARS> XML document is:



33

DTD Structure

```
<!DOCTYPE <root tag> [
  <!ELEMENT <name>(<components>)>
  . . . more elements . . .
]>
```

34

DTD Elements

- The description of an element consists of its name (tag), and a parenthesized description of any nested tags.
  - Includes order of subtags and their multiplicity.
- Leaves (text elements) have #PCDATA (*Parsed Character DATA*) in place of nested tags.

35

Example: DTD

```
<!DOCTYPE BARS [
  <!ELEMENT BARS (BAR*)>
  <!ELEMENT BAR (NAME , BEER+)>
  <!ELEMENT NAME (#PCDATA)>
  <!ELEMENT BEER (NAME, PRICE)>
  <!ELEMENT PRICE (#PCDATA)>
]>
```

A BARS object has zero or more BAR's nested within.

A BAR has one NAME and one or more BEER subobjects.

A BEER has a NAME and a PRICE.

NAME and PRICE are text.

36

Element Descriptions

- Subtags must appear in order shown.
- A tag may be followed by a symbol to indicate its multiplicity.
  - * = zero or more.
  - + = one or more.
  - ? = zero or one.
- Symbol | can connect alternative sequences of tags.

37

Example: Element Description

- A name is an optional title (e.g., "Prof."), a first name, and a last name, in that order, or it is an IP address:

```
<!ELEMENT NAME (
  (TITLE?, FIRST, LAST) | IPADDR
)>
```

38

Use of DTD's

1. Set standalone = "no".
2. Either:
   a) Include the DTD as a preamble of the XML document, or
   b) Follow DOCTYPE and the <root tag> by SYSTEM and a path to the file where the DTD can be found.

39

Example (a)

```
<?xml version = "1.0" standalone = "no" ?>
[illegible]
  [illegible]
  [illegible]
  [illegible]
  [illegible]
  [illegible]
]>
[illegible]
  [illegible]
    [illegible]
    [illegible]
  [illegible]
    [illegible]
  [illegible]
  [illegible]
  <BAR> …
</BARS>
```

The DTD

The document

40

Example (b)

- Assume the BARS DTD is in file bar.dtd.

```
<?xml version = "1.0" standalone = "no" ?>
<!DOCTYPE BARS SYSTEM "bar.dtd">
<BARS>
   <BAR><NAME>Joe's Bar</NAME>
        <BEER><NAME>Bud</NAME>
                <PRICE>2.50</PRICE></BEER>
        <BEER><NAME>Miller</NAME>
                <PRICE>3.00</PRICE></BEER>
   </BAR>
   <BAR> …
</BARS>
```

Get the DTD from the file bar.dtd

41

Attributes

- Opening tags in XML can have *attributes*.
- In a DTD,

`<!ATTLIST` *E* `. . . >`

declares an attribute for element *E*, along with its datatype.

42

Example: Attributes

- Bars can have an attribute `kind`, a character string describing the bar.

```
<!ELEMENT BAR (NAME BEER*)>
  <!ATTLIST BAR kind CDATA
    #IMPLIED>
```

Attribute is optional
opposite: #REQUIRED

Character string type; no tags

43

Example: Attribute Use

- In a document that allows BAR tags, we might see:

```
<BAR kind = "Sumiko"
  <NAME>Akasaka</NAME>
  <BEER><NAME>Sapporo</NAME>
        <PRICE>5.00</PRICE></BEER>
  ...
</BAR>
```

Note attribute values are quoted

44

ID's and IDREF's

- Attributes can be pointers from one object to another.
  - Compare to HTML's NAME = "foo" and HREF = "#foo".
- Allows the structure of an XML document to be a general graph, rather than just a tree.

45

Creating ID's

- Give an element *E* an attribute *A* of type ID.
- When using tag <*E* > in an XML document, give its attribute *A* a unique value.
- Example:

```
<E  A = "xyz">
```

46

Creating IDREF's

- To allow objects of type *F* to refer to another object with an ID attribute, give *F* an attribute of type IDREF.
- Or, let the attribute have type IDREFS, so the *F* –object can refer to any number of other objects.

47

Example: ID's and IDREF's

- Let's redesign our BARS DTD to include both BAR and BEER subelements.
- Both bars and beers will have ID attributes called `name`.
- Bars have SELLS subobjects, consisting of a number (the price of one beer) and an IDREF `theBeer` leading to that beer.
- Beers have attribute `soldBy`, which is an IDREFS leading to all the bars that sell it.

48

The DTD

```
<!DOCTYPE BARS [
  <!ELEMENT BARS (BAR*, BEER*)>
  <!ELEMENT BAR (SELLS*)>
      <!ATTLIST BAR name ID #REQUIRED>
  <!ELEMENT SELLS (#PCDATA)>
      <!ATTLIST SELLS theBeer IDREF
  #REQUIRED>
  <!ELEMENT BEER EMPTY>
      <!ATTLIST BEER name ID #REQUIRED>
      <!ATTLIST BEER soldBy IDREFS
  #IMPLIED>
]>
```

Bar elements have name as an ID attribute and have one or more SELLS subelements.

SELLS elements have a number (the price) and one reference to a beer.

Explained next

Beer elements have an ID attribute called name, and a soldBy attribute that is a set of Bar names.

49

Example Document

```
<BARS>
  <BAR name = "JoesBar">
      <SELLS theBeer =
  "Bud">2.50</SELLS>
      <SELLS theBeer =
  "Miller">3.00</SELLS>
  </BAR> …
  <BEER name = "Bud" soldBy = "JoesBar
      SuesBar …"/> …
  </BARS>
```

50

Empty Elements

- We can do all the work of an element in its attributes.
  - Like BEER in previous example.
- Another example: SELLS elements could have attribute `price` rather than a value that is a price.

51

Example: Empty Element

- In the DTD, declare:

<!ELEMENT SELLS EMPTY>
<!ATTLIST SELLS theBeer IDREF #REQUIRED>
<!ATTLIST SELLS price CDATA #REQUIRED>

- Example use:

<SELLS theBeer = "Bud" price = "2.50"/>

Note exception to "matching tags" rule

52

Elements

Elements are markup that enclose content

- <element_name>…</element_name> or <element_name />
- Content models
  - Parsed Character Data Only
  - Child Elements Only
  - Mixed

```
<author>Narayan, RK</author>
```

53

Attributes

Associate a name-value pair with an element

- <tag name1="value1" name2='value2'>…</tag>
  - Can be used to embellish content…
  - or to associate added content to an element

```
<author order='1'>Narayan,
  RK</author>
<author name= 'Desai, Kiran' />
```

54

Comments

Human-readable annotations

- Can be inserted anywhere after headers
- Not part of the document structure
- Usually ignored by XML parsers
- Do not have to be passed to application

```
<!-- This is a comment -->
```

55

Processing instructions

Machine-readable & application-specific

- Must be passed through by XML Parsers
- XML Declaration is a special PI
- XML Declaration is <u>always</u> first line in file

```
<?xml version='1.0' encoding='UTF-8' ?>
<?MyApp indent='on' linefeeds='off' ?>
```

56

Entities

- Placeholders for internal or external content
  - Placeholder for a single character…
  - or string of text…
  - or external content (images, audio, etc.)
- Implementation specifics may vary

```
<!ENTITY copyright "&#xA9;" >
&copyright; is replaced by ©
<!ENTITY pic SYSTEM "mugshot.gif" NDATA gif >
&pic; is replaced by graphic image
```

57

Character Encoding Issues

- XML Parsers must accept UTF-8 & UTF-16
- Also must accept &#nnnn; or &#xhhhh;
- MARC-8 encodings must be converted to Unicode for use in XML

http://lcweb.loc.gov/marc/specifications/specchartables.html

58

4. The infrastructure of XML

- Required to make it work…
  - DTDs & schemas: defining document classes
  - Reusing & integrating schemas (using namespaces)
  - Stylesheets for presentation & transformation
  - Standards for linking, querying, & pointing
  - Programming standards

59

Defining Document Classes

- Formal descriptions of document structure
  - Set expectations
  - Maximize reusability
  - Enforce business rules
- DTDs
- XML Schema
- Schematron
- Relax NG

60

Document Type Definitions (DTD)

- Legacy from SGML; part of XML standard

```
<!DOCTYPE Book SYSTEM 'http://…'>
<!ELEMENT Book (Front, Chapter+, Back?)>
<!ATTLIST Book
     type (series|monograph) #REQUIRED>
```

61

XML schema language

- New in XML
  - Uses XML syntax
  - Supports datatyping
  - Richer and more complex

```
<book xsi:noNamespaceSchemaLocation='HTTP://…'>

<xsd:element name='Book'>
  <xsd:complexType>
      <xsd:sequence>
            <xsd:element name='Front' minOccurs='1'
                  maxOccurs='1' type='frontType'/>…
```

62

Alternatives: Schematron & RelaxNG

- Schematron based on XPath (XSLT)
  - Doesn't support datatyping as well
  - Supports additional content models
  - May become an ISO standard
- RelaxNG
  - Returns some of the power of SGML DTDs back to XML (mixed and unordered content)
  - Uses datatyping from the XML Schema spec
  - Does not support inheritance
  - Developed by an OASIS Technical Committee chaired by James Clark

63

Namespaces

- Qualify element and attribute names
- Allows modularization of schemas
  - Mix and match elements from multiple schemas
    in document instances
  - Import or include from one XML Schema into another

```
<oai:metadata xmlns:oai='http:…' xmlns:oai_dc='…'
  xmlns:dc='…'>
  <oai_dc:dc>
      <dc:title>…</dc:title>
      <dc:creator>…</dc:creator>
```

64

XML & Cascading Style Sheets

- Attach styling instructions directly to XML files
  - `<?xml-stylesheet href="http:…" type="text/css" ?>`
  - Supported by newest browsers: IE5+, Mozilla, Opera
- Can style but not rearrange elements
  - Block or inline style
  - Bold, italic, underline, font, color, etc.
  - Margins, positioning
  - Generated content (browser support not good)

```
front author {color:red; font-weight:bold; font-
  family:serif;}
```

65

XSLT — Transforming Stylesheets

Language for transforming XML documents
- Into HTML, Text, or other XML documents
- Supported in new browsers (IE5+, Mozilla; not Opera)
- Usually applied on the server or in batch mode

- Valuable for interoperability or reusability

```
<xsl:template match='//author'>
   <xsl:element name='dc:creator'>
       <xsl:value-of select='lastname'/>
        <xsl:text>, </xsl:text>
        <xsl:value-of   select='firstname'/>
   </xsl:element >
</xsl:template>
```

66

XSL-FO (formatting objects)

- Another styling language
- Similar to CSS, but includes the power of XSLT to rearrange the document
- Syntax is entirely XML
- Not currently supported in browsers (but there are tools for use on the server or in batch mode)

```
<fo:block font-family="serif" font-weight="bold"
  color="red">
  Author: Cole, T
</fo:block>
```

67

XPath, XPointer, & XLink

- XPath
  - Allows addressing of parts of an XML document
  - Used in XSLT, XPointer, and XQuery
  - /document/front/author/@number
- XPointer (working draft)
  - Used as a fragment id in an XML URI reference
  - http://.../some.xml#xpointer(/document/front/author)
- XLink
  - Creates and describes extended or simple links between resources
  - Used for HTML-style hrefs or imgs, tables of contents, etc.

```
<aulink xlnk:type="simple" xlnk:href="…"
  xlnk:actuate="onRequest">
  Cole, T
</aulink>
```

68

XQuery (XML query language)

- Treat an XML document or collection of documents as a database
- Equivalent to SQL SELECT statements, only for XML
- Some support in XML databases (but working draft only)

69

Programming standards

- "Platform- and language-neutral interfaces that allow programs and scripts to dynamically access and update the content, structure, and style of XML documents."
- Document Object Model (DOM)
  - Object-based
  - Better for complex documents
  - High memory usage, slower
  - Documents can be updated
- Simple API for XML (SAX)
  - Event-based
  - Better for simple documents
  - Low memory usage, faster
  - Documents cannot be updated

70

Other XML-related standards

- XBase
- XForms
- XML Encryption
- XML Signature
- Many more …

71

6. Implementation issues & costs

Discussion Points:

- Tools
  - Range of options
  - Specialized tools expensive
  - Good technical resources on the Web
- Significant upfront investment
  - Initial training is greatest expense
  - Ongoing costs moderate
  - Need to be clear about objectives
  - Similar to what you went through to get on Web

72

XML authoring tools

- XML editors
  - XMetaL (Corel/SoftQuad)
  - Epic Editor (Arbortext)
  - TurboXML (Tibco Extensibility)
- Standard Office Tools
  - WordPerfect 2002 (Corel)
  - Microsoft Office XP
  - OpenOffice
- Plain Text Editors

73

Other XML tools

- Validating parsers & transformation tools
  - MSXML (Microsoft)
  - Xerces, Xalan (Apache Software Foundation)
  - XSV (U. of Edinburgh)
- Document management & database tools
  - Tamino (Software AG)
  - XMLCanon/Developer (Tibco/Extensibility)
  - DLXS/XPAT (U. of Michigan/OpenText)
- XML-aware browsers

74

XML resources on the Web

- World Wide Web Consortium
- OASIS
- Microsoft Developer Network
- Sun Microsystems
- Apache XML Project
- XML.COM (O'Reilly)
- XML.ORG (OASIS)
- ZVON.ORG

75

Need to acquire expertise

- Turnkey XML solutions of limited utility
- Can start with well-formed XML
  - For real utility, need to understand schemas
- Stylesheet expertise required to customize UI
  - CSS if users limited to XML-aware browsers
  - XSLT + CSS for browser neutrality
  - XSLT also required for crosswalk, refresh
- Outsourcing an option for certain applications
- Analogous to WWW & HTML four years ago

76

Ongoing costs

- Underlying technology now reasonably stable
  - Non-proprietary standards, now 4 years old
  - Parsers, validators, & transformation tools stable
  - If initial design meets long-term needs, ongoing maintenance costs will be minimal
- Changes to schemas, presentation layer, workflow can be costly
  - Small schema change can require major retrospective changes in documents & stylesheets
  - Work hard to identify necessary schema changes as quickly as possible

77

Final thoughts

- Core XML technologies stable & mature
  - Ancillary standards are still evolving
- Best way to learn is by doing
  - Start with a small project
- Long-term benefit potential is great
  - Archival refreshes generally easier, less frequent
  - Extensible & powerful
  - Facilitates interoperability now & in the future
- Requires initial investment of time and resources

78