XML, XSLT and e-Commerce

F. Dignum

Utrecht University
Contents

• XML as central formatting tool
• XSLT for conversions
• Business processes in e-Commerce
• ebXML: XML for business processes
XML order

Integrate
in XML

Product data

Supplier data
Compose document in XML

XML document

Prolog

XML Declaration

Document Type Declaration

Internal subset

Text + Markup

Parsing path

External subset
Concise XML Syntax

Example.xml

```xml
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<!DOCTYPE example SYSTEM "Example.dtd" [ 
<!ENTITY XML "eXtensible Markup Language"> 
<!ENTITY history SYSTEM "History.XML"> 
<!ENTITY wheelchair SYSTEM "c:/Wheelchair.tif" > 
<!ENTITY % figs "INCLUDE"> ]>
<example>
  <par>The &XML; format is a very important move
to bringing the benefits of structured markup
to the masses.</par>
&history;
  <par>The following figure shows a wheelchair:</par>
  <fig filename="wheelchair" />
  <par>The tags <![CDATA[<example>, <par> and
<fig../> are used in this document]].</par>
</example>
```

History.xml

```xml
<par>Superficially it looks like HTML because the
tags have the same delimiters, &#60; and &#62;
</par>
<par xml:space='preserve' xml:lang="en.gb">
--- XML ---
|                  |
SGML    HTML
</par>
```

Example.dtd

```xml
<!-- The example DTD -->
<!NOTATION TIFF SYSTEM "Showtiff.exe" >
<!ENTITY % figs "IGNORE" >
<![%figs[ <!ENTITY %ExampleContent "par | fig"> ]]> 
<!ENTITY % ExampleContent "par"> 
<!element example (%ExampleContent;)+>
<!element par (#PCDATA)> 
<!element fig EMPTY> 
<!attlist fig filename ENTITY #REQUIRED>
```

With thanks to Neil Bradley: “The XML Companion”,
Entities

Syntax:

for DTD: `<!ENTITY % nwc "note | warning | caution" >`

for DOC: `<!ENTITY XML "Extensible Markup Language" >`

Replacement

Keyword

Entity name
Entities

• Entity references are requests for data to be imbedded at the point of reference

• In a Document:
  – Internal text entities: simple text replacement
  – External text entities: inclusion of an external document
  – Binary entities: reference to multimedia files
  – Character defining entities: for characters outside the default characterset
  – Built-in entities: for characters used in markup
  – Character entities: the number of a character in the default characterset

• In a DTD:
  – Parameter entities: simple text replacement
Internal text entities

• Purpose: simple text replacement; text stored in entity
• DTD:
  – `<!ENTITY gca "Graphics Communications Association" >`
• DOC:
  – `... the &gca; sponsor meetings ...`
  – `==> ... the Graphics Communications Association sponsor meetings ...`
External text entities

- **Purpose**: inclusion of an external document; reference stored in entity

- **DTD**:
  - `<!ENTITY ch1 SYSTEM "http://www.../ch1.xml">`

- **DOC**:
  - `<book>a book about xml &ch1; ... more content ... </book>`
Binary entities

- Purpose: reference to multimedia files ("Non-XML data")
- Syntax in DTD:
  - `<!NOTATION Name PUBLIC Datatype>`
  - `<!ENTITY Name SYSTEM URL NDATA Datatype>`
- DTD
  - `<!ENTITY figure1 SYSTEM "c:\graphics\figure1.pic" NDATA EPS>`
  - `<!ELEMENT graphics EMPTY>`
  - `<!ATTLIST graphics filename ENTITY #IMPLIED>`
  - `<!ELEMENT p (#PCDATA | graphics)+ >`
- DOC
  - `<p>As is shown in the following diagram: <graphics filename="figure1"/></p>`
  - **wrong**: `<p> As is shown in the following diagram: &figure1;</p>`
Parameter entities

• Purpose: simple text replacement in a DTD
  
  - DTD: <!ENTITY % subelems "(para | list | table | note)" >
  
  - DTD: <!ELEMENT  body (things, %subelems;) >
  
  => DTD: <!ELEMENT  body (things, (para | list | table | note)) >

• Purpose: to keep text literally
  
  - DOC: <!ENTITY % subelems "(para | list | table | note)" >
ATTRIBUTE DECLARATION

Why: to associate information with an Element:
metadata, hypertext, multimedia, layout (!?), ...

Syntax:

DTD:  <!ELEMENT  el_name (.............)                         >
      <!ATTLIST    el_name  att_name1  type1   default1
               att_name2  type2   default2     >

Spelling of attribute name: as an XML name (~ element name)

Allowed: more than one <!ATTLIST for an element
<table>
<thead>
<tr>
<th>Type</th>
<th>Attribute value is</th>
</tr>
</thead>
<tbody>
<tr>
<td>CDATA</td>
<td>SGML character data</td>
</tr>
<tr>
<td>ENTIT(Y)(IES)</td>
<td>(list of) subdocument(s) entity name(s)</td>
</tr>
<tr>
<td>ID</td>
<td>Unique identifier for element</td>
</tr>
<tr>
<td>IDREF(S)</td>
<td>(list of) (a) reference(s) to a previously ID</td>
</tr>
<tr>
<td>NMTOKEN(S)</td>
<td>(list of) name token(s)</td>
</tr>
<tr>
<td>NOTATION</td>
<td>member of a list of notations</td>
</tr>
<tr>
<td>Name group</td>
<td>one of a finite set</td>
</tr>
</tbody>
</table>

*TYPES FOR ATTRIBUTE DECLARED VALUES*
ATTRIBUTE DECLARATION, declared values 1/3

DTD:

<!ELEMENT memo ( idinfo, body ) >
<!ATTLIST memo
  rev CDATA #REQUIRED
  size NMTOKEN #REQUIRED
  projects NMTOKENS #REQUIRED

DOC:

<memo rev="27/1/96 - 3.2a"
  size=".17-.19" projects="2-a 3-b" >..... </memo>
ATTRIBUTE DECLARATION, declared values 2/3

DTD:
<!NOTATION tex PUBLIC "-//local//NOTATION TeX Formula//EN"
 "c:\programs\show_tex" >
<!ENTITY pic1 SYSTEM "c:\proj3\file12" NDATA tex >
<!ENTITY pic2 SYSTEM "c:\proj4\file15" NDATA tex >
<!ELEMENT fig empty >         <!ELEMENT figr empty >
<!ELEMENT figrs empty >

<!ATTLIST fig    id     ID    #REQUIRED
    file    ENTITY #REQUIRED
<!ATTLIST figr refid    IDREF #REQUIRED >
<!ATTLIST figrs refids    IDREFS #REQUIRED >

DOC:
<fig id="oor" file="pic1" > <fig id="neus" file="pic2" >
<figr refid="neus"> <figrs refids="neus oor">
ATTRIBUTE DECLARATION, declared values 3/3

DTD:

```xml
<!NOTATION eqn SYSTEM "c:\eqn.exe">
<!NOTATION tex SYSTEM "c:\tex.exe">
<!ELEMENT memo ( idinfo, body ) >
<!ELEMENT formula CDATA >
<!ATTLIST memo security ( ts | sec | unc ) #REQUIRED >
<!ATTLIST formula data NOTATION #REQUIRED >
```

DOC:

```xml
<memo security="sec">...</memo>
<formula data="eqn"> 3 over 4 </formula>
```
DEFAULT VALUES FOR ATTRIBUTE DECLARATIONS

Reserved Words:

FIXED - used for attributes with constant values

REQUIRED - demands a user-entered value (always the case when there is no DTD)

IMPLIED - value supplied by application if not entered explicitly

Example default value in DOC:

```xml
<!ATTLIST memo security ( ts | sec | unc ) “unc” >
```
An order message in XML
The use of XSLT

XML → XSLT → HTML

XML → XSLT → XML

XML → XSLT → XML

XML → XSLT → XML
XSL-FO: Multilingual document

<?xml version="1.0"?>
<!DOCTYPE 書籍カタログ [
  <!ELEMENT 書籍カタログ (書籍)+ >
  <!ELEMENT 書籍 (書名, 著者, 出版社, (定価 | 在庫数)) >
  <!ATTLIST 書籍 xml:lang CDATA #REQUIRED >
  <!ELEMENT 書名 (#PCDATA) >
  <!ELEMENT 著者 (#PCDATA) >
  <!ELEMENT 出版社 (#PCDATA) >
  <!ELEMENT 定価 (#PCDATA) >
]>

<書籍カタログ>
  <書籍 xml:lang="JP">
    <書名>XML入門</書名>
    <著者>村田、門馬、荒井</著者>
    <出版社>日本経済新聞社</出版社>
    <定価>2800</定価>
  </書籍>
  <書籍 xml:lang="EN">
    <書名>Developing SCML DTDs</書名>
    <著者>E. Maler and J. el Andaloussi</著者>
    <出版社>Prentice Hall</出版社>
    <定価>50</定価>
  </書籍>
</書籍カタログ>
XML入門

著者 村田、門馬、荒井
出版社 日本経済新報社
定価 2800円

Developing SGML DTDs

著者 E. Maler and J. el Andaloussi
出版社 Prentice Hall
定価 50ドル
XML入門

Author: 村田、門馬、荒井
Publisher: 日本経済新聞社
Price: ¥2800

Developing SGML DTDs

Author: E. Maler and J. el Andaloussi
Publisher: Prentice Hall
Price: $50
XSL-FO features 1/2

• an XSL stylesheet defines the presentation of an XML document

• a style sheet can control the look and feel of a single document or a whole website

• incorporates XSLT: rule-based translation of XML documents into other type of documents, like HTML

• incorporates the ECMAScript language (standardized JavaScript) for data manipulation, allowing the translation of XML objects into any desired output
XSL-FO features 2/2

• Different and personalized style sheets can be applied to the same XML document for different purposes and different output media, such as display, print, handheld devices, etc.

• completely internationalized -left-to-right, right-to-left, and top-to-bottom scripts can occur mixed in the same document

• equipped with professional page layout facilities such as multiple column sets, rotated text areas, float zones, etc.

• history: subset of DSSSL with alternative syntax, plus key concepts from CSS
XSLT stylesheet

• XSLT “Stylesheet” actually: “Transformationsheet”

• Is itself an XML document

• Standard XML heading

• Standard XSLT heading

• Rules (“templates”)

• Geared towards a tree structure
Template rule

<xsl:template match="paragraph">
  <p>
    <xsl:apply-templates/>
  </p>
</xsl:template>

<xsl:template match="/">
  <html><body>
    <xsl:apply-templates/>
  </body></html>
</xsl:template>
Xpath

• Sub language, a.o. used within XSLT

• Xpath expression:
  – Identification of document parts
  – Boolean conditions
  – Numerical calculations
  – String manipulation
Nodes in an XML tree

- Root node
- Element node
- Text node
- Attribute node
- Comment node
- Processing instruction node
- Namespace node
All personnel are entitled to one personal day per year.

If you have any questions, please forward them to my office.
Transformation methods 1/2

• Recursive treatment of document ("recursive descent")
  – For each element a template rule
  – Each element produces desired output
  – Each element contains `<xsl:apply-templates>` call
  – Output data have more or less the same structure and order as the input
  – Also called push processing
Example Push Processing (XML->XML)

**XML in:**
<employee>
  <name>J. Smith</name>
  <number>123</number>
</employee>

**XML out:**
<PERSON><NUM>123</NUM><NM>J. Smith</NM></PERSON>

**XSLT:**
<xsl:template match="employee">
  <PERSON>
    <NUM><xsl:apply-templates select="number"/></NUM>
    <NM><xsl:apply-templates select="name"/></NM>
  </PERSON>
</xsl:template>
XSLT examples

Look at the webpage under the link “Literature on XSLT etc.”

It leads to a webpage with some tutorial material and a zip file with a number of examples.

You can open the examples using XMLSpy (opening the *.spp projects is sufficient):

- Booklist.(spp, xml, xsl) This example illustrates the generation of an HTML table from an XML-list.

- Brief.(spp, xml, xsl) This example illustrates the presentation of an XML document in HTML.
ebXML scenario

1. Request Business Details
2. Build Local System Implementation
3. Register Implementation Details
4. Register COMPANY A Profile
5. Agree on Business Arrangements
6. Do business transactions

Company A

Company B

ebXML Registry

Scenarios and Profiles
1. Company A decides to trade electronically.
2. Company A then implements an ebXML compliant application.
3. Company A submits, to an ebXML registry, their business profile information which describes their new ebXML related functionality.
4. ebXML compliant Company B discovers company A, who is also ebXML compliant, and also identifies the business scenarios supported by company A.
5. Company B then sends a request to company A requesting that they engage in a business scenario using ebXML and submits a proposed business arrangement. Upon negotiation of this proposal, a collaboration agreement is made between company A and B which forms part of the trading partner agreement.
6. Company A and B are now ready to engage in eBusiness using an ebXML conformant messaging.
ebXML business model

4 easy steps to ebXML

1. Design and register process

2. Implement and register profile

3. Optionally negotiate agreement

4. Conduct ebXML business
Business and Info in XML
Business transactions

Business Process

Process Composition

Partner Types

Business Collaboration

Agreement

Economic Event

Choreography

Transition

Guard

Roles

Business Transaction

Request Document

Response Document

Business Transaction Execution Patterns
Transactions in XML/DTD

<!ELEMENT EbXmlProcessSpecification ((package|include|documentation)*)>
<!ATTLIST EbXmlProcessSpecification name CDATA #REQUIRED uuid CDATA #IMPLIED version CDATA #IMPLIED >
...
<!ELEMENT business-transaction (documentation?, request, response*)>
<!ATTLIST business-transaction name CDATA #REQUIRED isNonRepudiationReceiptRequired (true | false) #IMPLIED isIntelligibleCheckRequired (true | false) #IMPLIED isAuthorizationRequired (true | false) #IMPLIED isSecureTransportRequired (true | false) #IMPLIED isReliableTransportRequired (true | false) #IMPLIED isNonRepudiationRequired (true | false) #IMPLIED isNonRepudiationReceiptRequired (true | false) #IMPLIED timeToAcknowledge CDATA #IMPLIED