Optimizing XML for Comparison and Change

Nigel Whitaker & Robin La Fontaine
DeltaXML
About DeltaXML

• Software company based in Worcestershire, UK
• First XML comparison product: 2002
• Comparison engine, toolkit, format specific products, n-way merge
• Primarily a product company, provide support & consultancy
About the talk/paper

- Based on support experiences; Why didn’t they do it this way...
- Audience: XML developers, schema designers, XML users/authors
- Covering XML documents and data
- Discuss our software, but applicable to other comparison and XML processing more generally
Overview

1. Whitespace
2. Mixing Ordered and Unordered Content
3. Representing Change
4. Format Flattening
We see more issues with data than documents

“DTDs aren’t needed because we’re reading and writing” - private communication

“Tools should strip out whitespace”, yes, but it requires user intervention and isn’t as good as a parser
Whitespace 2

<contact>

... 

<phone type="work">
  <countryCode>44</countryCode>
  <areaCode>020</areaCode>
  <local>7234 5678</local>
</phone>

</contact>
<!DOCTYPE contact SYSTEM "contact.dtd">
<contact>

...</phone type="work">
        <countryCode>44</countryCode>
        <areaCode>020</areaCode>
        <local>7234 5678</local>
    </phone>
</contact>
Whitespace 4

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Value</th>
<th>Type</th>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>attr</td>
<td>type</td>
<td>work</td>
<td>attr</td>
<td>type</td>
<td>'work'</td>
</tr>
<tr>
<td>elem</td>
<td>countryCode</td>
<td>44</td>
<td>text</td>
<td>type</td>
<td>'^J'</td>
</tr>
<tr>
<td>elem</td>
<td>areaCode</td>
<td>020</td>
<td>elem</td>
<td>countryCode</td>
<td>'44'</td>
</tr>
<tr>
<td>elem</td>
<td>local</td>
<td>7234 5678</td>
<td>elem</td>
<td>areaCode</td>
<td>'020'</td>
</tr>
</tbody>
</table>
<pre><code>                           | elem | local           | '7234 5678' |
                           |      | text             | '^J'        |
</code></pre>
Whitespace: Suggestions

• Create a DTD for your data
• Relate it to the instance files where possible
• xml:space - useful for post parsing space control
Overview

1. Whitespace
2. Mixing Ordered and Unordered Content
3. Representing Change
4. Format Flattening
Mixed Order 1

- Documents are usually ordered
- Data can be ‘orderless’
- We use different algorithms:
  - Ordered data uses LCS dynamic programming techniques
  - Orderless uses hashing and maps
Mixed Order 2

<contact>
  <name>John Smith</name>
  <addressLine>25 Malet Street</addressLine>
  <addressLine>Bloomsbury</addressLine>
  <addressLine>London</addressLine>
  <addressLine>UK</addressLine>
  <postcode>W1A 2AA</postcode>
  <phone type="office">+44 20 1234 5678</phone>
  <phone type="fax">+44 20 1234 5680</phone>
  <phone type="mobile">+44 7123 123456</phone>
</contact>
<contact>
    <name>John Smith</name>
    <addressLine>25 Malet Street</addressLine>
    <addressLine>Bloomsbury</addressLine>
    <addressLine>London</addressLine>
    <addressLine>UK</addressLine>
    <postcode>W1A 2AA</postcode>
    <phones deltaxml:ordered="false">
        <phone type="office">+44 20 1234 5678</phone>
        <phone type="fax">+44 20 1234 5680</phone>
        <phone type="mobile">+44 7123 123456</phone>
    </phones>
</contact>
Mixed Order: Suggestions

• Don’t mix as siblings
• Add some wrappers, useful for other purposes
• Document processing expectations for orderless
Overview

1. Whitespace
2. Mixing Ordered and Unordered Content
3. Representing Change
4. Format Flattening
xml:lang

• Defined in XML Spec, but why should you use it?
• Use cases: profiling/filtering, but others too
• Seen in HTML, Docbook, DITA, XSLT, SVG
Segmenting text

<p>Hello World</p>

<p>
  <word>Hello</word>
  <space></space>
  <word>world</word>
  <punctuation>!</punctuation>
</p>
Segmenting: implementation

- Naive assumption: words are separated by spaces
- Simple implementation: tokenizer, regexp
- Only works for latin/western alphabets
- Unicode Annex 29 is the proper way
- Implemented by International Components for Unicode (icu4j.jar)
- But which BreakIterator? Now we need xml:lang!

x
xml:lang recommendation

- Use it whenever possible:
  - DTD designers - put it on the root element at least
  - Developers please write the attributes
- Remember icu4j.jar
- Finding your locale:
  
  ```xml
  <xsl:variable name="locale" as="xs:string" select="ancestor-or-self::*[@xml:lang][1]/@xml:lang"/>
  ```
xml:lang

<?xml-model href="http://docbook.org/xml.dtd">
<?xml-model href="http://docbook.org/xml.dtd">
<article xmlns="http://docbook.org/ns/docbook" version="5.0" xml:lang="en_GB">
  <info>
    <title>Optimizing XML for Compa</title>
  </info>
</article>
Representing Change 1

• XML Generic formats
  • Comparator specific formats: deltaV2
  • Track-changes: editor specific, W3C Community Group

• Language Specific:
  • HTML: `<ins/>`, `<del/>`
  • DITA: `@status`, `@rev`
  • DocBook: `@revisionflag`
Representing Change 2

<p status="new">This topic demonstrates how status can be used</p>

<title>DITA <ph status="new">Topic</ph> title</title>
Representing Change 3

The <xref href="http://www.w3.org/TR/2006/REC-xml-20060816/">XML Specification</xref> allows ...

The <xref href="http://www.w3.org/TR/xml/">XML Specification</xref> allows...

The <xref status="new" href="http://www.w3.org/TR/2006/REC-xml-20060816/">XML Specification</xref><xref status="deleted" href="http://www.w3.org/TR/xml/">XML Specification</xref> allows...
Representing Change 4

<image href="bike.gif" placement="break"><alt>Two-wheeled bicycle</alt></image>

<image href="bike.gif" placement="break"><alt>Two-wheeled <ph status="deleted">bicycle</ph> <ph status="new">cycle</ph></alt></image>

<image href="bike.gif" placement="break"> <alt status="deleted">Two-wheeled bicycle</alt> <alt status="new">Two-wheeled cycle</alt> </image>

<image status="deleted" href="bike.gif" placement="break"><alt>Two-wheeled bicycle</alt></image> <alt>Two-wheeled cycle</alt></image>

<image status="new" href="bike.gif" placement="break"><alt>Two-wheeled cycle</alt></image>
Representing Change: Suggestions

• Built-in support for change - consider using it
• Ideally provide consistency for text() also allow a simple wrapper element
• Use repetition (*, +) unless good reason not to
Overview

1. Whitespace
2. Mixing Ordered and Unordered Content
3. Representing Change
4. Format Flattening
Format flattening I

- Document users care about words, not XML centric view
- But formatting has semantics too

<p>Hello XML London attendees!</p>
<p><b>Hello</b> XML London attendees!</p>
Format Flattening 2

<p>
<b-start/>
<word>Hello</word>
<b-end/>
<space/>
<word>XMLLondon</word>
<space/>
<word>attendees</word>
</p>
Format Flattening 3

• Removability: can you remove the formatting element leaving a valid result? The content model of a `<span>` is the same as that of the places where a `<span>` is used.

• Nestability: can the formatting element contain an immediate child of the same type? A `<span>` can directly contain another `<span>`. `<b><i>word</i></b>` vs `<i><b>word</b></i>`
Format Flattening 4

For example: <p><i>CD</i> is three and, therefore,</p>

<p>FNP = <i>DD</i> + 2 kGy</p>

= 3,4 kGy + 2 kGy

= 5,4 kGy

NOTE FNP shall not exceed 5,5 kGy.

For example: FNP = <i>DD</i> + 2 kGy

= 3,4 kGy + 2 kGy

= 5,4 kGy

NOTE FNP shall not exceed 5,5 kGy.
For example:

\[ \text{CD} \times 3 \text{ and, therefore, } \]
\[ \frac{\text{FNP}}{\text{DD}} \times 2 \text{ kGy} \]
\[ \text{FNP} = 3.4 \text{ kGy} + 2 \text{ kGy} \]

**Model Description**

Any combination of:

- Text, numbers, or special characters
- \(<\text{email}>\text{ Email Address}\rangle\)
- \(<\text{ext-link}>\text{ External Link}\rangle\)
- \(<\text{uri}>\text{ Uniform Resource Indicator (URI)}\rangle\)
- \(<\text{hr}>\text{ Horizontal Rule}\rangle\)
- . . . .
- \(<\text{p}>\text{ Paragraph}\rangle\)
Format Flattening: Suggestions

- Our informal *removability* and *nestability* guidance is probably a good idea

- Avoid *pernicious mixed content*