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

Whitespace 1

- 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>
```

Whitespace 3

```
<!DOCTYPE contact SYSTEM “contact.dtd”>
<contact>
   ...
   <phone type="work">
     <countryCode>44</countryCode>
     <areaCode>020</areaCode>
     <local>7234 5678</local>
   </phone>
</contact>
```

Whitespace 4

Type	Name	Value
attr	type	work
elem	countryCode	44
elem	areaCode	020
elem	local	7234 5678

Type	Name	Value
attr	type	'work'
text		'^J '
elem	countryCode	'44'
text		'^J '
elem	areaCode	'020'
text		'^J '
elem	local	'7234 5678'
text		'^J '

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>
```

Mixed order

```
<contact deltaxml:ordered='false' >
  <name>John Smith</name>
  <address deltaxml:ordered='true'>
    <addressLine>25 Malet Street</addressLine>
    <addressLine>Bloomsbury</addressLine>
    <addressLine>London</addressLine>
    <addressLine>UK</addressLine>
  </address>
  <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>
```

x

Mixed Order 3

```
<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!

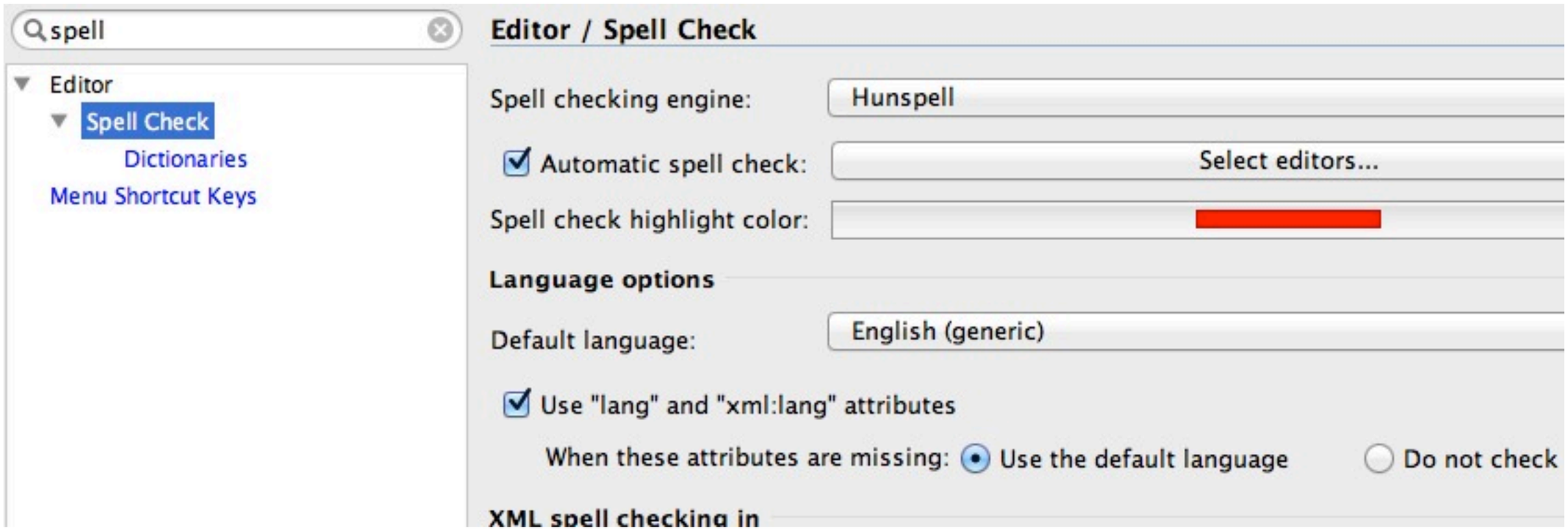
x

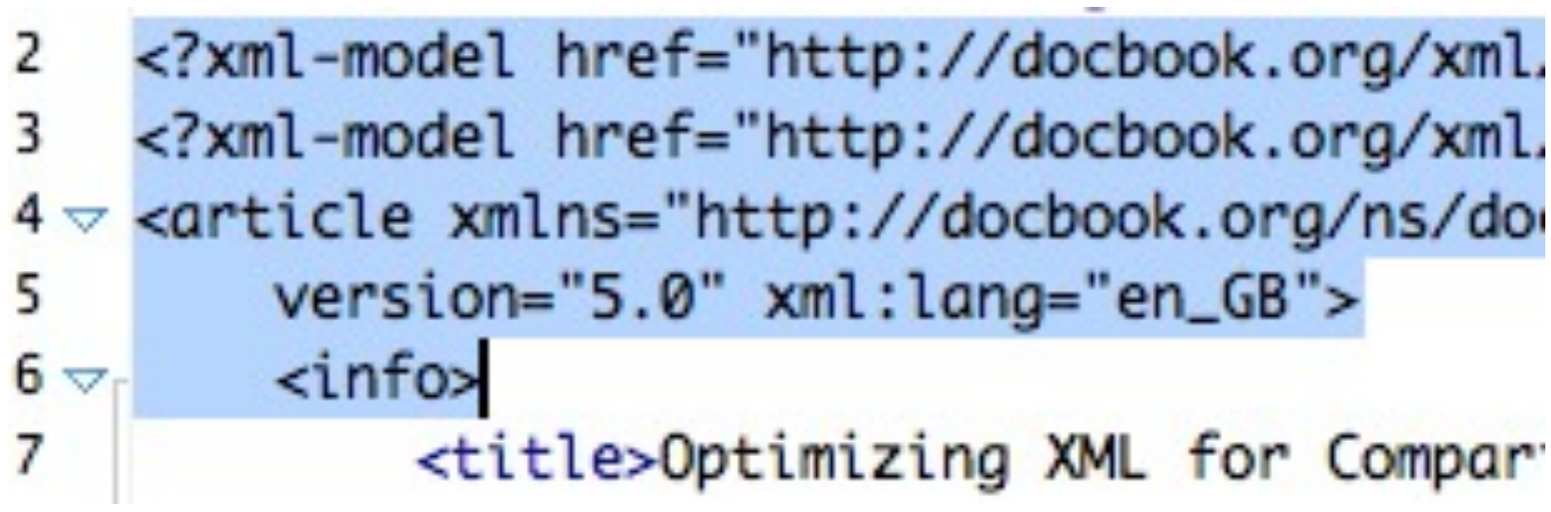
xml:lang recommendation

- 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:

```
<xsl:variable name="locale" as="xs:string" select="ancestor-or-self::*[@xml:lang][1]/@xml:lang"/>
```

xml:lang


spell
Editor
Spell Check
Dictionaries
Menu Shortcut Keys
Editor / Spell Check
Spell checking engine:
Hunspell
Automatic spell check:
Select editors...
Spell check highlight color:
Language options
Default language:
English (generic)
Use "lang" and "xml:lang" attributes
When these attributes are missing:
Use the default language
Do not check
XML spell checking in


2 <?xml-model href="http://docbook.org/xml.
3 <?xml-model href="http://docbook.org/xml.
4 <article xmlns="http://docbook.org/ns/do
5 version="5.0" xml:lang="en_GB">
6 <info>
7 <title>Optimizing XML for Compar

X

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

This topic demonstrates how status can be used

<title>DITA <ph status="new">Topic</ph> title</title>

Representing Change 3

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

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

<p>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...</p>
```

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>
<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 1

- 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> </space>
  <word>XMLLondon</word>
  <space> </space>
  <word>attendees</word>
</p>
```

Format Flattening 3

- Removability: can you remove the formatting element leaving a valid result? The content model of a is the same as that of the places where a is used.
- Nestability: can the formatting element contain an immediate child of the same type? A can directly contain another .
word vs word

Format Flattening 4

```
<td>For example:<p><italic>CD</italic>* is three and, therefore,</p>
   <p>FNP = <italic>DD</italic>* + 2 kGy</p>
   <p>   = 3,4 kGy + 2 kGy</p><p>   = 5,4 kGy</p>
   <p>NOTE   FNP shall not exceed 5,5 kGy.</p></td>
```

```
<td>For example:<break/><italic>CD</italic>* is three and, therefore,<break/>
    FNP = <italic>DD</italic>* + 2 kGy
   <break/>   = 3,4 kGy + 2 kGy<break/>
   = 5,4 kGy<break/>
   NOTE    FNP shall not exceed 5,5 kGy.</td>
```

Format Flattening 5

```
<td deltaxml:deltaV2="A!=B">For example:
    <p deltaxml:deltaV2="A"><italic>CD</italic>* is three and, therefore,</p>
    <break deltaxml:deltaV2="B"/>
    <p deltaxml:deltaV2="A">FNP = <italic>DD</italic>* + 2 kGy</p>
<italic deltaxml:deltaV2="A!=B">
    <p deltaxml:deltaV2="A">   = 3,4 kGy + 2 kGy</p>
    ....
```

Model Description

Any combination of:

- Text, numbers, or special characters
- `<email>` **Email Address**
- `<ext-link>` **External Link**
- `<uri>` **Uniform Resource Indicator (URI)**
- `<hr>` **Horizontal Rule**
- `....`
- `<p>` **Paragraph**

Format Flattening: Suggestions

- Our informal *removability* and *nestability* guidance is probably a good idea
- Avoid *pernicious mixed content*