XML Technologien und ihre Bedeutung für LZA und I

XML Technologien und ihre Bedeutung für LZA und I

Sebastian Rahtz
Information Manager, Oxford University Computing Services
Text Encoding Initiative Board of Directors
Text Encoding Initiative Technical Council

Berlin, October 2007

Outline

On the table

- Markup and digital text
- XML basics
- Schemas
- XSLT
- Vocabularies
- The Text Encoding Initiative
- Interchange

Markup and digital text

A real world object

What shall we record?

- Where is it?
- What shape is it? what are its decorative features?
- What do we know about Molly Cotton?
- Text of the inscription
- When did we make this record?

Some of this is standard metadata, but we probably want to go deeper.

Before XML

- In the beginning there was *procedural* markup `RED INK ON; print balance; RED INK OFF`
- which being generalised became *descriptive* markup `<balance type='overdrawn'>some numbers</balance>`
- also known as encoding or annotation

Descriptive markup allows for re-use of data

In more detail

Markup:

- makes explicit the distinctions we want to make when processing a string of bytes
- works better when we record what we think things *are* than what they *look like*
- adds value by supplying multiple annotations
- facilitates re-use of the same material
  - in different formats
  - in different contexts
  - by different users

It works for text and data

What is in a text?

Upon Julia's *Clothes*

WHEN as in silks my *Julia* goes,
Then, then (me thinks) how sweetly flowes
That liquefaction of her clothes.

Next, when I cast mine eyes and see
That brave Vibration each way free;
O how that glittering taketh me!

What can markup capture?

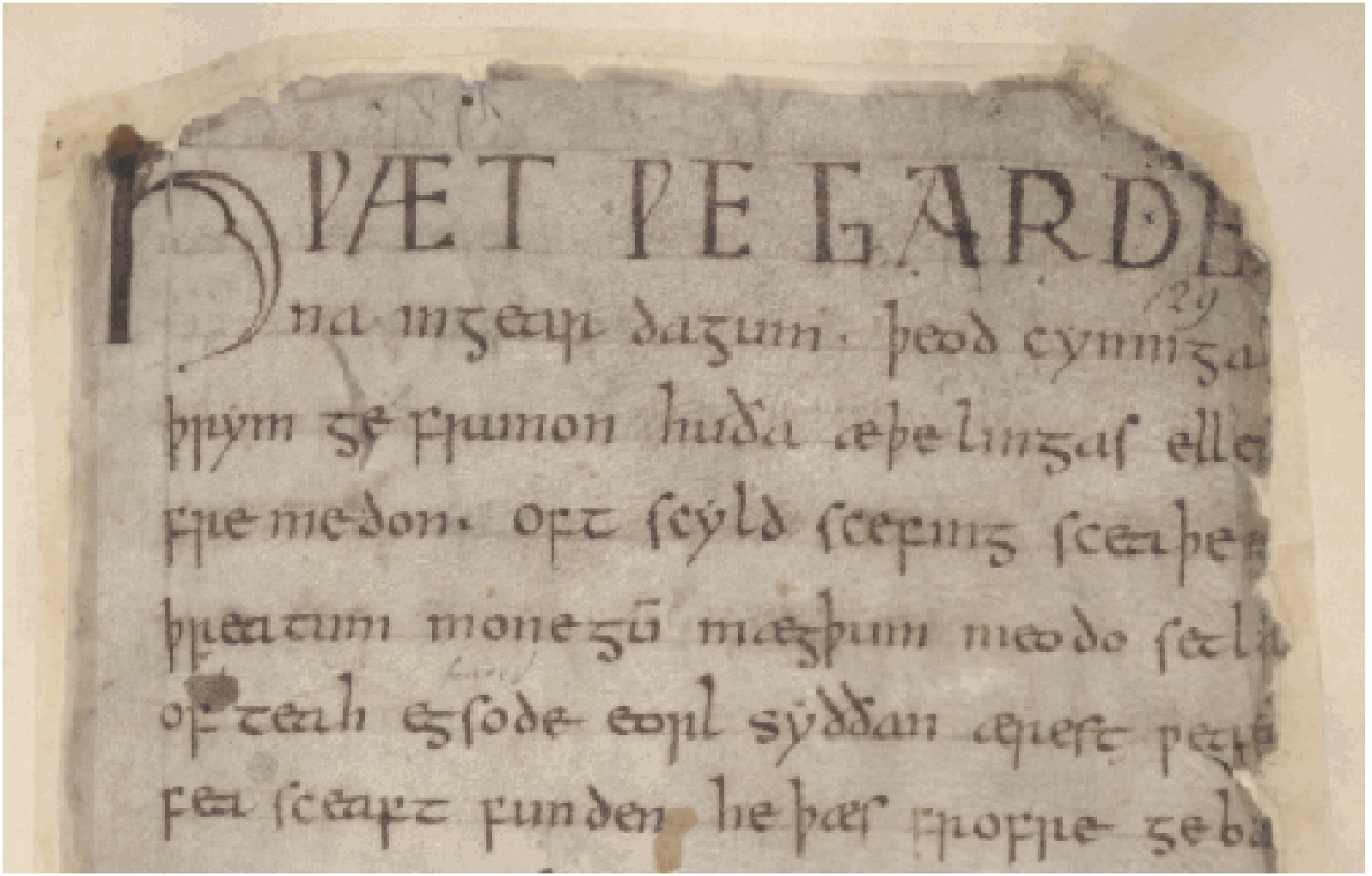
Compare

```
<head>Upon Julia's Clothes</head>
<lg>
  <l>Whenas in silks my <hi>Julia</hi> goes,</l>
  <l>Then, then (me thinks) how sweetly flowes</l>
  <l>That liquefaction of her clothes.</l>
</lg>
```

and

```
<s n="1" role="head">
  <w type="pp">Upon</w>
  <w type="np">Julia</w>
  <w type="pos">'s </w>
  <w type="nn2">Clothes</w>
</s>
<s n="2" role="line">
  <w type="adv">Whenas</w>
  <w type="pp">in</w>
  <w type="nn2">silks</w> ...
</s>
```

A manuscript



Choose again what to record. This:


```
<hi rend="dropcap">H</hi>&amp;amp;WYN;ÆT WE GARDE <lb/>na in
gear-dagum þeod-cyninga <lb/>þrym gefrunon,
hu ða æþelingas <lb/>ellen fremedon. oft scyld scefing sceaþe
<add>na</add>
<lb/>þreatum, moneg<expan>um</expan> mægþum meodo-setl
<add>a</add>
<lb/>of<damage desc="blot"/>teah egsode <sic>eorl</sic> syððan
ærest wear
<add>þ</add>
<lb/>fea sceaft funden...
```

Or this:


```
<lg>
  <l>Hwæt!  we Gar-dena in gear-dagum</l>
  <l>þeod-cyninga þrym gefrunon,</l>
  <l>hu ða æþelingas ellen fremedon,</l>
</lg>
<lg>
  <l>Oft Scyld Scefing sceaþena þreatum,</l>
  <l>monegum mægþum meodo-setla ofteah;</l>
  <l>egsode Eorle, syððan ærest wearþ</l>
  <l>feasceaft funden...</l>
</lg>
```

From text to data


```
<biblStruct xml:id="Simons1999">
 <analytic>
  <author>
   <forename>Gary</forename>
   <forename>F.</forename>
   <surname>Simons</surname>
  </author>
  <title level="a">Using Architectural Forms to Map
      TEI Data into an
      Object-Oriented Database</title>
 </analytic>
 <monogr>
  <title level="j">Computers and the Humanities</title>
  <imprint>
   <biblScope type="vol">33</biblScope>
   <biblScope type="issue">1-2</biblScope>
   <date>1999</date>
   <biblScope type="pp">85-101</biblScope>
  </imprint>
 </monogr>
</biblStruct>
```

Or mimicking a computer program


```
<match pattern="**/*.htm">
 <select type="resource-exists">
  <when test="1/2.xml">
   <redirect-to uri="cocoon:/1/2.xml"/>
  </when>
  <otherwise>
   <read src="1/2.htm" mime-type="text/html"/>
  </otherwise>
 </select>
</match>
```

And back to text again


```
<body>
 <div xml:lang="en">
  <ab>DR MOLLY COTTON</ab>
  <ab>O.B.E. F.S.A.</ab>
  <ab>MARY AYLWIN COTTON</ab>
  <ab>NÉE MARSHALL</ab>
  <ab>*1. 8. 1902 +31. 5. 1984</ab>
 </div>
</body>
```

but also data


```
<listPerson>
  <person sex="2">
    <persName>
      <forename>Mary</forename>
      <forename>Aylwin</forename>
      <surname>Cotton</surname>
    </persName>
    <birth when="1902-08-01"/>
    <death when="1984-05-31"/>
    <nationality reg="UK"/>
  </person>
</listPerson>
```

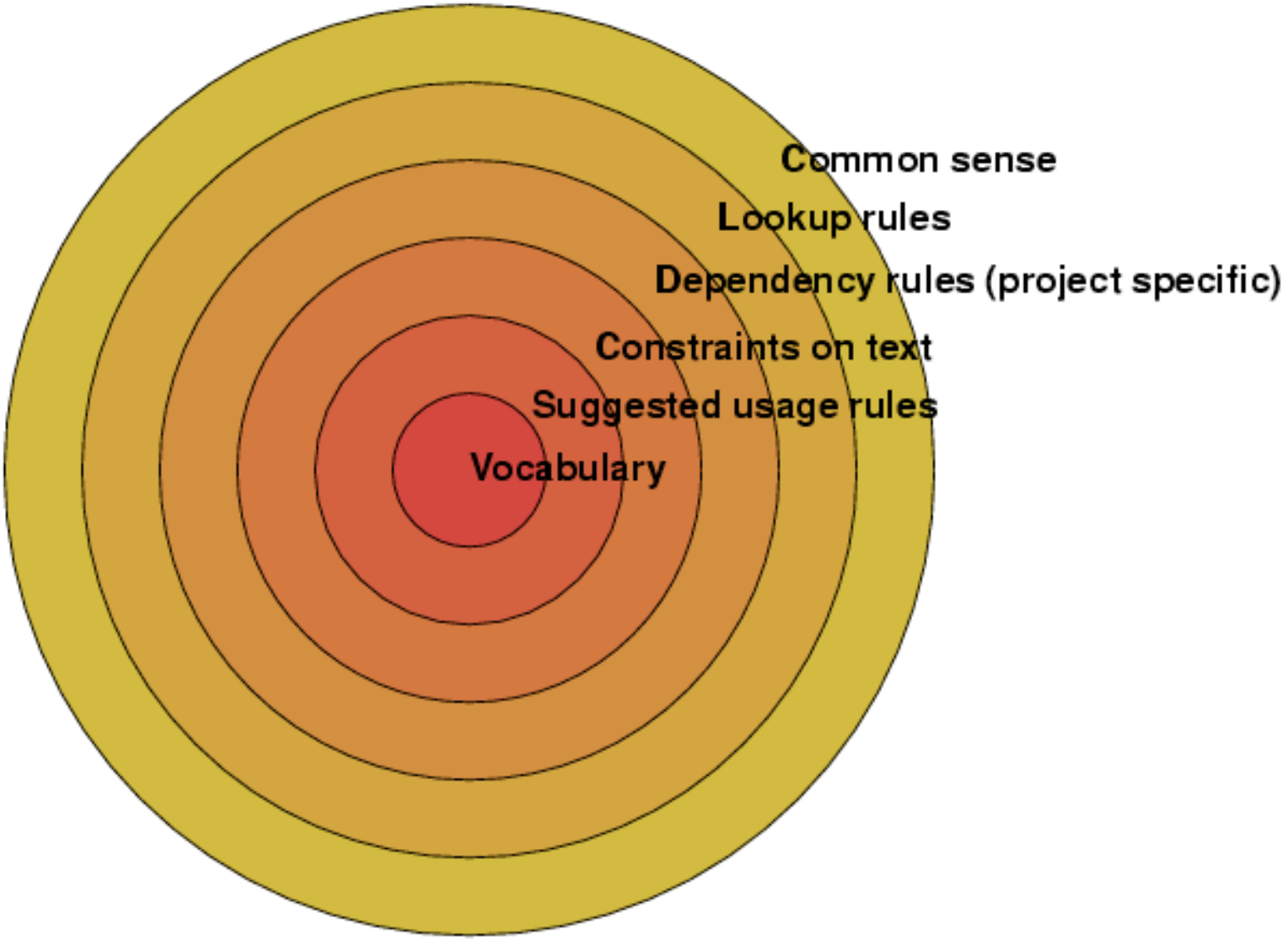
Validation and schemas

XML has a fixed syntax, and you define the vocabulary rules.
Syntax and vocabulary checking schemas can be written in:

- The W3C schema language
- RELAX NG schema language
- XML DTD language

But there is more to life than checking syntax checking.

What kinds of validation do we need?



Simple schemas

DTD, XSD and RELAX NG schemas specify:

- the name of the root element
- names for all elements used
- names and datatypes and (occasionally) default values for their attributes
- rules about how elements can nest
- and a few other things, depending on the schema language

A schema does *not* specify anything about what elements ‘mean’

ISO Schematron can add data-specific rules

Before:

```
<element name="identifier">
  <text/>
</element>
```

After:

```
<element name="identifier">
  <text/>
</element>
<pattern name="identifier-content">
  <rule context="x:identifier">
    <report test="string-length(normalize-space(.))=0">
Identifier cannot be empty
    </report>
  </rule>
</pattern>
```

Worthy but dull


```
<define name="xcri-description">
 <element name="description">
  <ref name="xcri-model-description"/>
  <attribute name="type">
   <text/>
  </attribute>
 </element>
</define>
<pattern name="description-type">
 <rule context="x:presentation/x:description">
  <report
     test="@type and not(@type='Study Hours'
or @type='Contact Pattern' or @type='Contact Hours')"> The
type attribute on the description of a presentation is a fixed
vocabulary;
      this does not match. </report>
 </rule>
</pattern>
```

Security checks


```
<define name="xcri-model-description">
<!-- ... -->
 <zeroOrMore>
  <choice>
   <ref name="htmlDiv"/>
   <text/>
  </choice>
 </zeroOrMore>
 <pattern name="description-safe">
  <rule context="x:description">
   <report test=".//xhtml:script"> Do NOT allow HTML
        script elements in your description </report>
  </rule>
 </pattern>
</define>
```

Tricksier


```
<define name="xcri-postcode">
 <element name="postcode">
  <text/>
  <ns prefix="x" uri="http://xcri.org/profiles/catalog"/>
  <pattern name="postcode-type">
   <rule context="x:postcode">
    <report test="string-length(normalize-
space(.))=0">Postcode
           element cannot be empty</report>
    <assert
      test="document('postcodes.xml')/postcodes/postcode/@ident=
before(.,' ')"> First half of postcode must match known list
</assert>
   </rule>
  </pattern>
 </element>
</define>
```

Mixed vocabulary


```
<course xmlns="http://xcri.org/profiles/catalog">
 <identifier
xmlns="http://xcri.org/profiles/catalog">O07P158AHJ</identifier>
 <title xmlns="http://xcri.org/profiles/catalog">After the Ice
Ages - New Research on Mesolithic Britain</title>
 <description
xmlns="http://xcri.org/profiles/catalog" type="Structure">
  <xhtml:div>
   <xhtml:br/>9.45am Mesolithic Britain: setting
     the scene<xhtml:br/>PROFESSOR NICK BARTON<xhtml:br/>
     ...</xhtml:div>
 </description>
 <subject
xmlns="http://purl.org/dc/elements/1.1/">Archaeology</subject>
</course>
```

ISO NVDL can specify validation of mixed namespaces in stages


```
<namespace ns="http://xcri.org/profiles/catalog">
 <validate schema="oxcap.rng">
  <mode>
   <namespace ns="http://purl.org/dc/elements/1.1/">
    <validate schema="dc.rng"/>
   </namespace>
   <namespace ns="http://www.w3.org/1999/xhtml">
    <validate schema="xhtml.rng">
     <mode>
      <anyNamespace>
       <allow/>
      </anyNamespace>
     </mode>
    </validate>
   </namespace>
  </mode>
 </validate>
</namespace>
```

Using XML: transformation

The XSLT language is

- Expressed in XML; uses namespaces to distinguish output from instructions
- Purely functional
- Reads and writes XML trees
- Designed to generate XSL FO, but now widely used to generate HTML

How is XSLT used? (1)

- With a command-line program to transform XML (eg to HTML)
  - Downside: no dynamic content, user sees HTML
  - Upside: no server overhead, understood by all clients
- In a web server *servlet*, eg serving up HTML from XML (eg Cocoon, Axkit)
  - Downside: user sees HTML, server overhead
  - Upside: understood by all clients, allows for dynamic changes

How is XSLT used? (2)

- In a web browser, displaying HTML from XML on the fly
  - Downside: many clients do not understand it
  - Upside: user gets XML displayed as HTML
- Embedded in specialized program
- As part of a chain of production processes, performing arbitrary transformations

XSLT implementations

MSXML	Built into Microsoft Internet Explorer
Saxon	Java-based, standards leader, implements XSLT 2.0 (basic version free)
Xerces	Java-based, widely used in servlets (open source)
libxslt	C-based, fast and efficient (open source); used from Perl, PHP, C etc
transformiix	C-based, used in Mozilla (open source)

What do you mean, 'transformation'?

Take this

```
<recipe>
  <title>Pasta for beginners</title>
  <ingredients>
    <item>Pasta</item>
    <item>Grated cheese</item>
  </ingredients>
  <cook>Cook the pasta and mix with the cheese</cook>
</recipe>
```

and make this


```
<html>
  <h1>Pasta for beginners</h1>
  <p>Ingredients: Pasta Grated cheese</p>
  <p>Cook the pasta and mix with the cheese</p>
</html>
```

How do you express that in XSL?


```
<xsl:stylesheet version="1.0"
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
 <xsl:template match="recipe">
  <html>
   <h1>
    <xsl:value-of select="title"/>
   </h1>
   <p>Ingredients:
   <xsl:apply-templates select="ingredients/item"/>
   </p>
   <p>
    <xsl:value-of select="cook"/>
   </p>
  </html>
 </xsl:template>
</xsl:stylesheet>
```

Standardized vocabularies

- XHTML: web pages
- MathML: maths
- SVG: vector graphics
- RSS/Atom: news feeds
- Docbook/DITA: technical writing
- TEI: digital texts
- MODS/METS: library finding aids

The list goes on and on for each specialism ...

The Text Encoding Initiative

- defines and names several hundred useful textual distinctions
- provides a set of modules that can be used to define schemas making those distinctions
- provides a customization mechanism for modifying and combining those definitions with new ones using the same conceptual model

The TEI provides a framework for the definition of multiple schemas

TEI Deliverables

- A set of recommendations for text encoding, covering both generic text structures and some highly specific areas based on (but not limited by) existing practice
- A very large collection (about 530) of element definitions with associated declarations for various schema languages
- a modular system for creating personalized schemas or DTDs from the foregoing

Legacy of the TEI

- a way of looking at what ‘text’ really *is*
- a codification of current scholarly practice
- (crucially) a set of shared assumptions and priorities about the digital agenda:
  - focus on content and function (rather than presentation)
  - identify generic solutions (rather than application-specific ones)

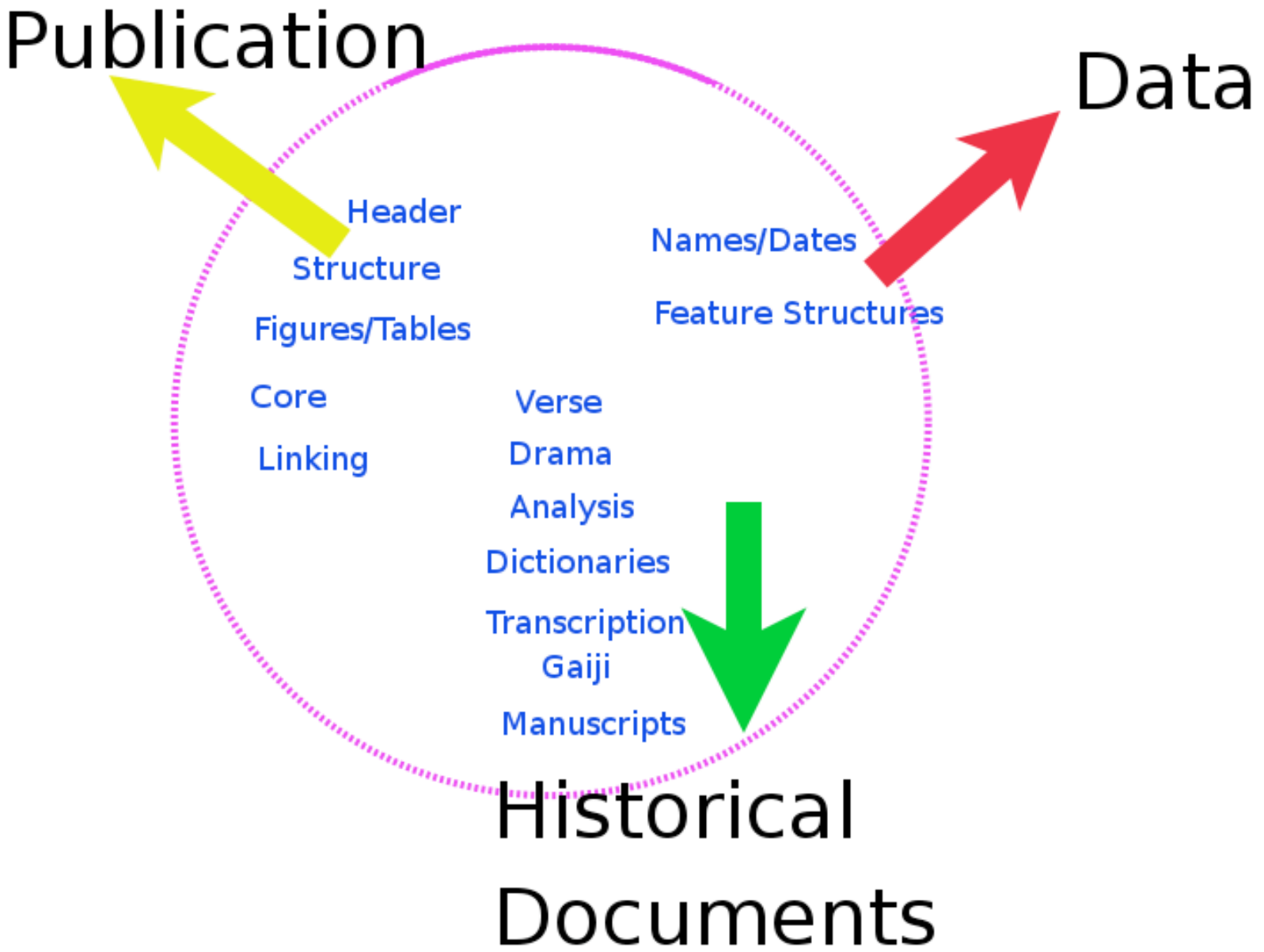
Some things the TEI is famous for

- Prose, verse and drama
- Editorial markup
- Linguistic markup
- Complex linking
- Describing manuscripts

Some less obvious areas the TEI is in

- Names, places, persons, organisations and dates
- Digital facsimiles
- Non-hierarchical markup
- Dictionaries
- Figures and tables
- Writing new documents (and these slides ...)
- Describing XML schemas

The Text Encoding Initiative meets historical data



Example: some report writing


```
<div>
 <head>The Fabrics</head>
 <p> The fabrics identified are described below. Proportions
   of each fabric are listed as percentage figures. Following
   each description is a generalized list of vessels commonly
   found in that fabric at Middleton Stoney.
 <list type="gloss">
   <label>Ra</label>
   <item> Reduced ware; coarse; major inclusions: grog. The
type
        is a`catch-all' for grog-tempered fabrics, and
different sherds
        mayincorporate inclusions of quartz, quartzite, mica,
chalk,
        and red or black iron ore in varying proportions. The
colour
        ranges from grey or black through to pale pinkish or
yellowish
        brown. </item>
  </list>
 </p>
</div>
```

Example: a more complex record from classical archaeology


```
<person xml:id="V2-45386">
  <sex value="1"/>
  <persName type="full" xml:lang="el-
grc">Θεμιστοκλ□ς</persName>
  <persName xml:lang="el-grc-x-lgpn">Qemistoklh3s</persName>
  <persName xml:lang="el-grc-x-
perseus">*qemistoklh=s</persName>
  <birth notAfter="-0075" notBefore="-0125">
    <placeName key="Hagnous">Hagnous</placeName>
  </birth>
  <floruit>ii/i BC</floruit>
  <state key="#relationship">
    <label>s. <persName xml:lang="el-grc">Θεόφραστος
I</persName>
    </label>
  </state>
  <state key="#relationship">
    <label>f. <persName xml:lang="el-grc">Θεόφραστος
II</persName>
    </label>
  </state>
  <state key="#relationship">
    <label>f. <persName xml:lang="el-grc">Ξενοκλ□ς
II</persName>
```

Hang on (1), why don't you use an SQL database, mapping software, GIS etc?

1. All the data is in a single archival format independent of proprietary software
2. All the complexity can be modelled in one place
3. If you do have deep recursive text structures, modelling them in an RDB is hard
4. Making everything an XML transformation is very powerful

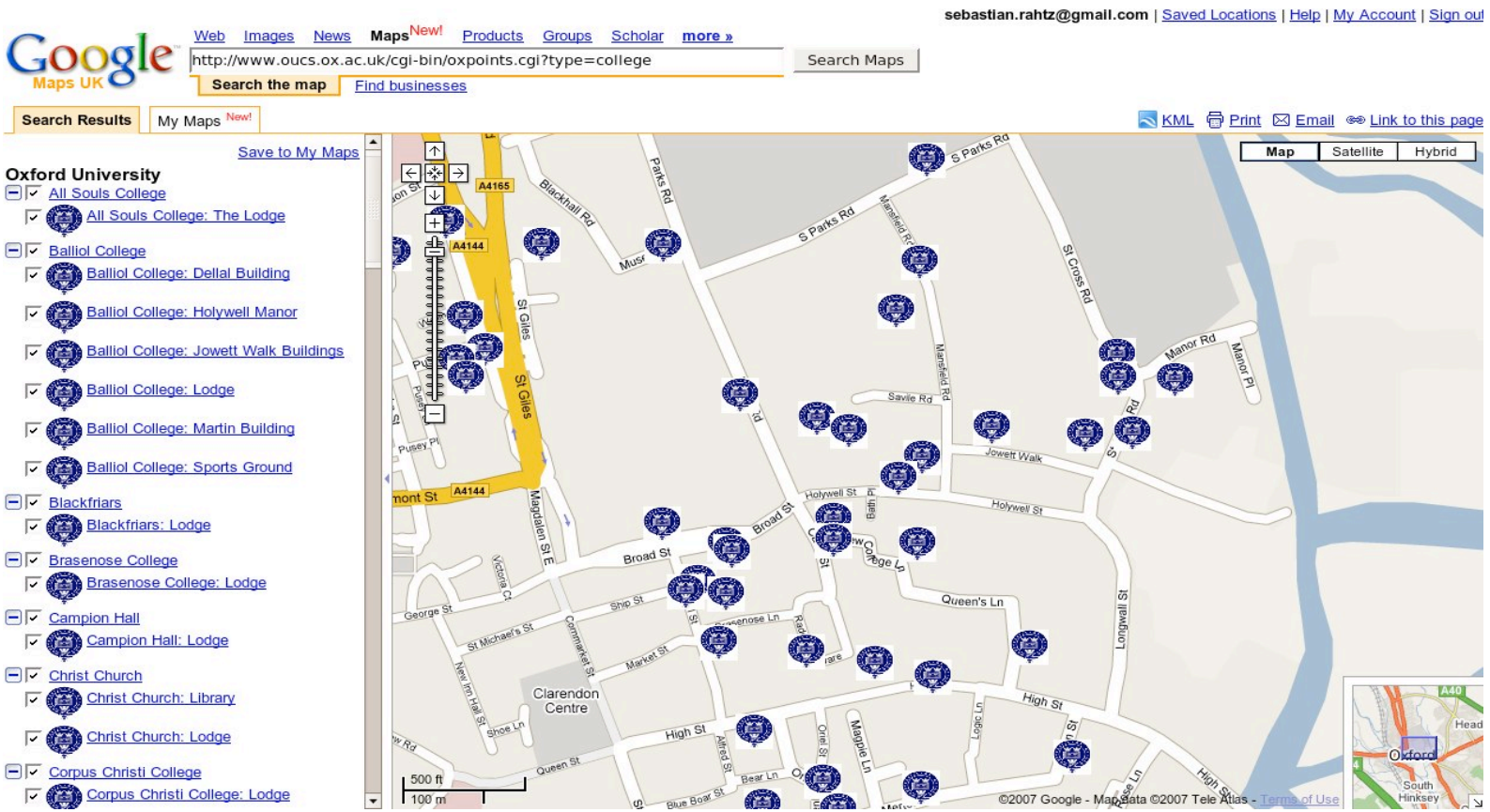
Hang on (2): why aren't you using an ontology in RDF?

- Luckily, the TEI semantics map neatly and reliably to CIDOC CRM concepts, which means that RDF representation in CIDOC terms is just a transformation away.
- The TEI is a way of instantiating your material for CIDOC CRM material
- The TEI scheme has explicit recording of CIDOC mapping.

A TEI XML geo record

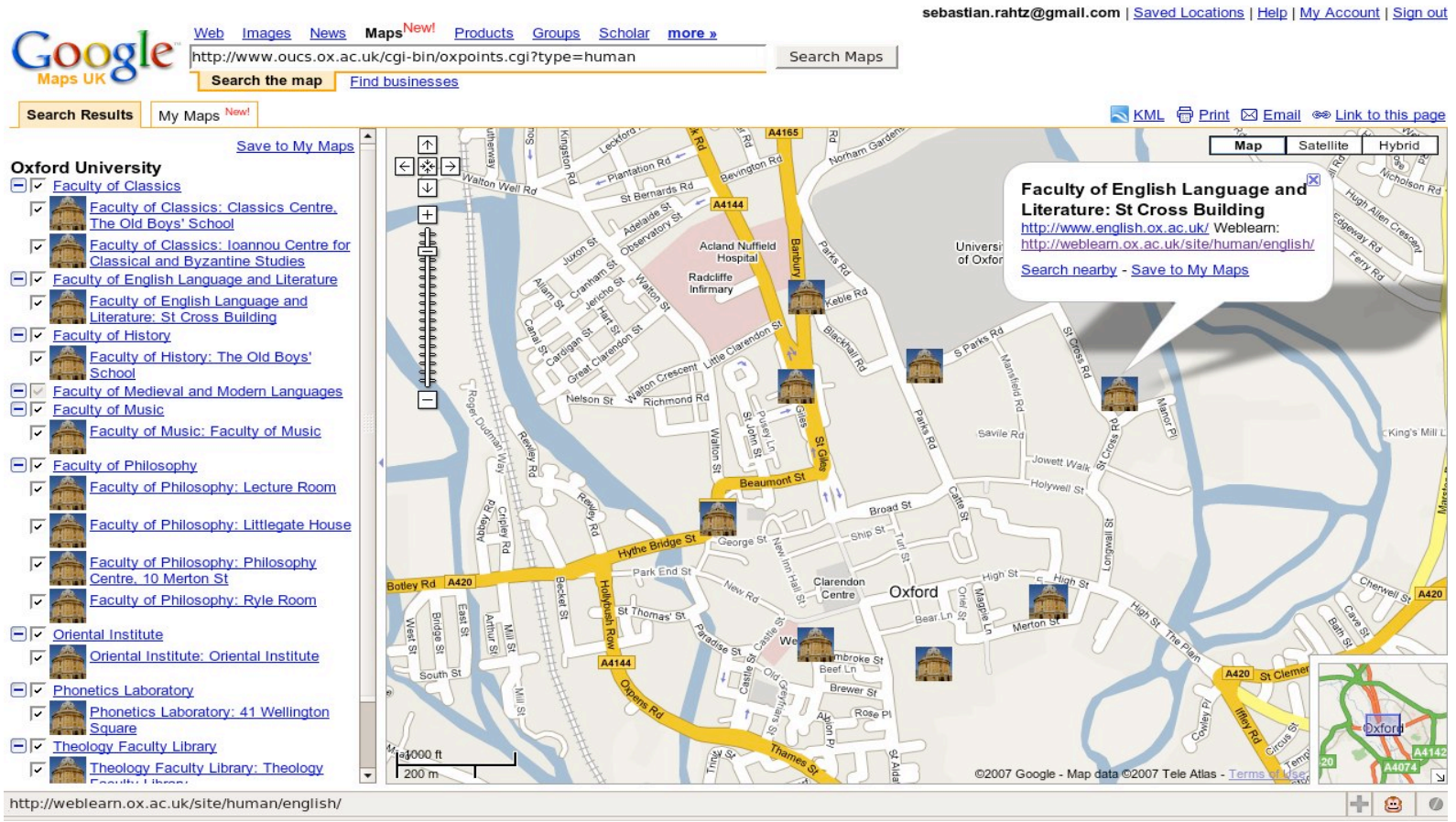

```
<place type="college" xml:id="mert">
 <label>Merton College</label>
 <location type="address">
  <address>
   <addrLine>Oxford</addrLine>
   <postCode>OX1 4JD</postCode>
  </address>
 </location>
 <place type="building">
  <label>Lodge</label>
  <location when="2007-01-20T21:26:32.601Z">
   <geo>-1.252216100692749 51.751291136684 88</geo>
   <note> recorded by Sebastian Rahtz</note>
  </location>
 </place>
</place>
```

Displaying TEI data



The map replacement — more information

Replacing the text browser?



Mapping the world

`http://www.oucs.ox.ac.uk/oxpoints/test.html`

Representation of a schema


```
<elementSpec module="spoken" ident="pause">
 <classes>
  <memberOf key="model.divPart.spoken"/>
  <memberOf key="att.timed"/>
  <memberOf key="att.typed"/>
 </classes>
 <content>
  <rng:empty/>
 </content>
 <attList>
  <attDef ident="who" usage="opt">
   <gloss>A unique identifier</gloss>
   <desc>supplies the identifier of the
        person or group pausing.
   </desc>
   <datatype>
    <rng:ref name="data.pointer"/>
   </datatype>
  </attDef>
 </attList>
 <desc>a pause either between or within utterances.</desc>
</elementSpec>
```

Roma: building ad-hoc TEI schemas

TEI Roma: generating validators for the TEI

Modules

New | Customize | Language | Modules | Add Elements | Change Classes | Schema | Documentation | Save Customization

List of TEI Modules

	Module name		A short description	Changes
add	analysis	?	Simple analytic mechanisms	
add	certainty	?	Certainty and uncertainty	
add	core	?	Elements common to all TEI documents	
add	corpus	?	Header extensions for corpus texts	
add	declarefs	?	Feature system declarations	
add	dictionaries	?	Printed dictionaries	
add	drama	?	Performance texts	
add	figures	?	Tables, formulae, and figures	
add	gaiji	?	Character and glyph documentation	
add	header	?	The TEI Header	
add	iso-fs	?	Feature structures	
add	linking	?	Linking, segmentation and alignment	
add	msdescription	?	Manuscript Description	
add	namesdates	?	Names and dates	
add	nets	?	Graphs, networks and trees	

List of selected Modules

remove	core
remove	tei
remove	header
remove	textstructure

Conclusions

What XML buys you:

- No need to invent a markup language for textual data
- Plenty of vocabularies to choose from
- Standardized processing

Great XML success stories

- XSLT: simple language for transforming XML
- XPath: powerful language for addressing XML trees
- XHTML: taming HTML
- XML DOM: standardized API
- RSS: news feeds
- KML: overlays on maps
- AJAX: dynamic web pages reading XML
- TEI: of course...

What XML gets right

- Mandatory Unicode
- Stream-based extensible markup
- Relative ease of writing a parser
- Namespaces
- Transformations from tree to tree
- Community acceptance of a simple standard

What XML gets wrong

- Legacy of SGML makes for a few awkwardnesses (eg entities)
- Namespaces were a slight afterthought
- Not amenable to old-style line-based tools
- Only models hierarchies

Final lesson

Whatever else you do, make your data available to other machines.

XML is at its best for inter-process communication