Digital Texts, XML, and TEI

Lou Burnard, Matthew Driscoll, and Sebastian Rahtz
Questions we will try to answer on this course

1. What is mark-up for?
2. What is XML?
3. How do I do cool stuff with my digital texts?
4. How is the TEI system organized and what is it for?
5. How do I customize the TEI system to create digital texts the way I want them?
Questions we will (probably) not try to answer on this course

- Who can I get to do all this for me?
- How would I do all this using Word?
- How would I do all this using a database?
- How would I do all this using some other XML scheme?
- What is a digital text for anyway?
What's 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’s in a text (2)?
What’s in a text (3)?

Hwæt wē Gār-Dena in gēar-dagum
bēod-cyninga þrym gefrūnon,
hū ōa æþelingas ellen fremedon.

Oft Scyld Scēfing sceapena þreatum,
5 mōnegum mǣþum meodo-setla oftēah;
ægsode Eorl[æ], syðdan ārest weard
fēascaft funden; hē þæs frōfre gebād:
wēox under wolcnum, weorð-myndum þāh,
oðþæt him Æghwylc þāra ymb-sittendra
10 ofer hron-rāde hŷran scolde,
The ontology of text

Where is the text?

- in the shape of letters and their layout?
- in the original from which this copy derives?
- in the ideas it brings forth? in their format, or their intentions?

Texts are abstractions conjured up by readers. Markup encodes those abstractions.
Encoding of texts

- Texts are more than sequences of encoded glyphs
  - They have structure and content
  - They also have multiple readings
- Encoding, or markup, is a way of making these things explicit
- Only that which is explicit can be reliably processed
Styles of markup

- In the beginning there was *procedural* markup
  
  ```plaintext
  RED INK ON; print balance; RED INK OFF
  ```

- which being generalised became *descriptive* markup
  
  ```xml
  <balance type='overdrawn'>some numbers</balance>
  ```

- also known as *encoding* or *annotation*

  descriptive markup allows for re-use of data
Some more definitions

- Markup makes explicit the distinctions we want to make when processing a string of bytes.
- Markup is a way of naming and characterizing the parts of a text in a formalized way.
- It’s (usually) more useful to markup what things are than what they look like.
What does markup capture?

Compare

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

and

```xml
<s n="1" role="head">
    <w type="pp">Upon</w>
    <w type="np">Julia</w><w type="pos">’</w>s <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>
```
Likewise..

Compare

<hi rend="dropcap">H</hi>&WYNYÆT WE GARDE
<lb/>na in gear-dagum þeod-cyninga
<lb/>þrym gefrunon, hu ða æþelingas
<lb/>ellen fremedon. oft scyld scæping sceapæ<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 sceæft funden...

and

<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 Scæping sceapæna þreatum,</l>
  <l>monegum mægþum meodo-setla ofteah;</l>
  <l>egsode Eorle, syððan ærest wearþ</l>
What’s the point of markup?

- To make explicit (to a machine) what is implicit (to a person)
- To add value by supplying multiple annotations
- To facilitate re-use of the same material
  - in different formats
  - in different contexts
  - for different users
A useful mental exercise

Imagine you are going to markup several thousand pages of complex material....

- Which features are you going to markup?
- Why are you choosing to markup this feature?
- How reliably and consistently can you do this?

Now, imagine your budget has been halved. Repeat the exercise!
Some alphabet soup

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SGML</td>
<td>Standard Generalized Markup Language</td>
</tr>
<tr>
<td>HTML</td>
<td>Hypertext Markup Language</td>
</tr>
<tr>
<td>W3C</td>
<td>World Wide Web Consortium</td>
</tr>
<tr>
<td>XML</td>
<td>eXtensible Markup Language</td>
</tr>
<tr>
<td>DTD</td>
<td>Document Type Definition (or Declaration)</td>
</tr>
<tr>
<td>CSS</td>
<td>Cascading Style Sheet</td>
</tr>
<tr>
<td>Xpath</td>
<td>XML Path Language</td>
</tr>
<tr>
<td>XSLT</td>
<td>eXtensible Stylesheet Language - Transformations</td>
</tr>
<tr>
<td>RelaxNG</td>
<td>Regular Expression Language for XML (New Generation)</td>
</tr>
</tbody>
</table>

Oh, and then there’s also

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TEI</td>
<td>Text Encoding Initiative</td>
</tr>
</tbody>
</table>
XML: what it is and why you should care

- XML is **structured data** represented as strings of text
- XML looks like HTML, except that:-
  - XML is **extensible**
  - XML must be **well-formed**
  - XML can be **validated**
- XML is application-, platform-, and vendor- independent
- XML empowers the **content provider** and facilitates data integration
An example XML document

```xml
<?xml version="1.0" encoding="utf-8" ?>
<cookBook>
  <recipe n="1">
    <head>Nail Soup</head>
    <ingredientList>
      <ingredient>an onion</ingredient>
      <ingredient>two carrots</ingredient>
      <ingredient>water</ingredient>
      ...
      <ingredient>a nail</ingredient>
      <ingredient>some gullible peasants</ingredient>
    </ingredientList>
    <procedure>
      <step>put the water on to boil</step>
      ....
      <step>take out the nail and serve</step>
    </procedure>
  </recipe>
  <recipe n="2">
    <!-- contents of second recipe here -->
  </recipe>
  <recipe n="2">
    <!-- hic desunt multa -->
  </recipe>
</cookBook>
```
XML terminology

An XML document may contain:-

- elements, possibly bearing attributes
- processing instructions
- comments
- entity references
- marked sections (CDATA, IGNORE, INCLUDE)

An XML document must be well-formed and may be valid
XML is an international standard

- XML requires use of ISO 10646
  - a 31 bit character repertoire including most human writing systems
  - encoded as UTF8 or UTF16
- other encodings may be specified at the document level
- language may be specified at the element level using xml:lang
The rules of the XML Game

- An XML document represents a (kind of) tree
- It has a single root and many nodes
- Each node can be
  - a subtree
  - a single element (possibly bearing some attributes)
  - a string of character data
- Each element has a type or generic identifier
- Attribute names are predefined for a given element; values can also be constrained
Representing an XML tree

- An XML document is encoded as a linear string of characters.
- It begins with a special **processing instruction**.
- Element occurrences are marked by **start- and end-tags**.
- The characters `<` and `&` are Magic and must always be "escaped".
- **Comments** are delimited by `<!- - and - ->`.
- **CDATA sections** are delimited by `<![CDATA[ and ]]>`.
- Attribute name/value pairs are supplied on the start-tag and may be given in any order.
- **Entity references** are delimited by `&` and `;`.
XML syntax: the small print

What does it mean to be well-formed?

1. there is a single root node containing the whole of an XML document
2. each subtree is properly nested within the root node
3. names are always case sensitive
4. start-tags and end-tags are always mandatory (except that a combined start-and-end tag may be used for empty nodes)
5. attribute values are always quoted
Splot the mistake

```xml
<greeting>Hello world!</greeting>
<greeting>Hello world!</Greeting>

<greeting><grunt>Ho</grunt> world!</greeting>
<grunt>Ho <greeting>world!</greeting></grunt>
<greeting><grunt>Ho world!</greeting></grunt>

<grunt type=loud>Ho</grunt>
<grunt type="loud"></grunt>

<grunt type= "loud">
<grunt type ="loud"/>
```
Defining the rules

A **valid** XML document conforms to rules which are stated in an external **schema** of some sort.

A schema specifies:

- 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

n.b. A schema does *not* specify anything about what elements "mean"
Schema languages

Schemas can be written in:

- The W3C schema language
- Relax NG schema language
- XML DTD Language

In the TEI, we mostly use Relax NG
Parts of an XML document

- The XML declaration
- Namespace declarations
- The root element of the document itself
An XML document must begin with an XML declaration which does two things:

- specifies that this is an XML document, and which version of the XML standard it follows
- specifies which character encoding the document uses

```xml
<?xml version="1.0" ?>
<?xml version="1.0" encoding="iso-8859-1" ?>
```

The default, and recommended, encoding is UTF-8
Namespace declarations

An XML document can use elements declared in different name spaces.

- a namespace declaration associates a namespace prefix with an external identifier (which looks like an URL)
- the default namespace *may* be declared using a special `xmlns` attribute
- other name spaces must all use a special prefix, which is also declared

```
<TEI xmlns="http://www.tei-c.org/ns/1.0"> ... </TEI>
```

```
<TEI xmlns="http://www.tei-c.org/ns/1.0"
     xmlns:math="http://www.mathml.org">
  <p>... <math:expr>... </math:expr> ...</p>
</TEI>
```

There is a special xml namespace, used by the TEI for global attributes `xml:id` and `xml:lang`
The Doctype Declaration

In DTD world, an optional "Document Type" declaration may appear:

```xml
<?xml version="1.0" ?>
<!DOCTYPE hello [<!ELEMENT hello (#PCDATA)>]>
<hello xmlns="http://www.greetings.org">
  hello world
</hello>
```

- The DTD is one way of associating the document with its schema (but is not used by W3C or Relax NG for this purpose)
- The DTD subset is used to provide declarations additional to those in the schema
- The DTD subset may be internal, external, or both
In XML a schema is optional!

XML allows you to make up your own tags, and doesn’t require a schema...

- The XML concept is dangerously powerful:
  - XML elements are light in semantics
  - one man’s <p> is another’s <para> (or is it?)
  - the appearance of interchangeability may be worse than its absence

- But XML is too good to ignore
  - mainstream software development
  - proliferation of tools
  - the language of the web
What can a schema (or DTD) do for you?

- ensure that your documents use only predefined elements, attributes, and entities
- enforce structural rules such as ‘every chapter must begin with a heading’ or ‘recipes must include an ingredient list’
- make sure that the same thing is always called by the same name

Schema languages vary in the amount of validation they support
What kinds of validation do we need?

- Vocabulary
- Suggested usage rules
- Constraints on text
- Dependency rules (project specific)
- Lookup rules
- Common sense
What kinds of validation do we need?

- Common sense
- Lookup rules
- Dependency rules (project specific)
- Constraints on text
- Suggested usage rules
- Vocabulary
  
  `<list>`  `<label>`  `<item>`
What kinds of validation do we need?
What kinds of validation do we need?

- Common sense
- Lookup rules
- Dependency rules (project specific)
- Constraints on text
- Suggested usage rules
- Vocabulary

```xml
<list> <label> <item>
  list ((label, item)+ | item+)
</item> <label>
figure.attributes.url = xsd:anyURI
```
What kinds of validation do we need?

- Vocabulary
- Suggested usage rules
- Constraints on text
- Dependency rules (project specific)
- Lookup rules
- Common sense

```xml
<list>
  <label> <item> list ((label, item)+ | item+) </item>
</label>
```

```xml
figure.attributes.url = xsd:anyURI
```

*if list is of type GLOSS, content must include labels*
What kinds of validation do we need?

- Common sense
- Lookup rules
- Dependency rules (project specific)
- Constraints on text
- Suggested usage rules
- Vocabulary

Example XML code:

```xml
<list>
  <label>
    <item/>
    <item/>
  </label>
  <item/>
</list>
```

```xml
figure.attributes.url = xsd:anyURI
```

- if list is of type GLOSS, content must include labels
- persons referenced by key must exist in the persons database
What kinds of validation do we need?

- Common sense
- Lookup rules
- Dependency rules (project specific)
- Constraints on text
- Suggested usage rules
- Vocabulary

```xml
<list> <label> <item>
  list ((label, item)+ | item+)
</item></label></list>
```

```xml
figure.attributes.url = xsd:anyURI
```

- if list is of type GLOSS, content must include labels
- persons referenced by key must exist in the persons database
- don't use the table element to represent glossaries!
What can the TEI do for you?

The TEI provides a framework for the definition of multiple schemas:

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

- Originally, a research project within the humanities
  - Sponsored by three professional associations
  - Funded 1990-1994 by US NEH, EU LE Programme et al

- Major influences
  - digital libraries and text collections
  - language corpora
  - scholarly datasets

- International consortium established June 1999 (see http://www.tei-c.org/)
Goals of the TEI

- better interchange and integration of scholarly data
- support for all texts, in all languages, from all periods
- guidance for the perplexed: what to encode — hence, a user-driven codification of existing best practice
- assistance for the specialist: how to encode — hence, a loose framework into which unpredictable extensions can be fitted

These apparently incompatible goals result in a highly flexible, modular, environment
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 of element definitions with associated declarations for various schema languages
- A modular system for creating personalized schemas or DTDs from the foregoing

for the full picture see http://www.tei-c.org/TEI/Guidelines/
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)