#### **Digital Texts with XML and the TEI**

*Lou Burnard* February 2004



# Questions we will try to answer on this course

- 1. What is text mark-up for?
- 2. What is XML?
- 3. How is the TEI system organized and what is it for?
- 4. How do I customize the TEI system to create digital texts the way I want them?
- 5. How do I do cool stuff with my digital texts?



# 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?



# **Today's topics**

- What do we mean by a digital text?
- What do we mean by markup?
- What is the TEI?

. . .

We will try to provide answers to these questions. You will also explore them, both theoretically and practically. And you will meet a powerful XML editor



#### 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!

#### Is this:

#### 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 mel

#### the same as this:



#### Digital Texts with XML and the TEI

# 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 then 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

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



#### **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

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

#### and



# 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



#### **First exercise**

Imagine you are going to markup several thousand pages like this.

- 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

SGML Standard Generalized Markup Language

HTML Hypertext Markup Language

W3C World Wide Web Consortium

XML eXtensible Markup Language

- **DTD** Document Type Definition (or Declaration)
- **CSS** Cascading Style Sheet
- **XSLT** eXtensible Stylesheet Language -Transformations
- RelaxNG Not an acronym, as far as we know

Oh, and then there's also

**TEI** Text Encoding Initiative



# 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 vendorindependent
- XML empowers the content provider and facilitates data integration



#### An example XML document

```
<?xml version="1.0" encoding="utf-8" ?>
   <cookBook>
     <recipe n="1">
      <head>Nail Soup</head>
      <ingredientList> .... </ingredientList>
      <procedure> .... </procedure>
     </recipe>
    <recipe n="2">
     <!-- contents of second recipe here -->
     </recipe>
&recipe3;
   <!-- hic desunt multa -->
   </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 reportoire 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

<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 this course, we will be using Relax NG





#### Parts of an XML document

```
<?xml version="1.0" ?>
<!DOCTYPE hello >
<hello xmlns="http://www.greetings.org">
hello world
</hello>
```

- The XML declaration
- Namespace declarations
- The Doctype declaration
- The root element of the document itself



#### **Namespace declarations**

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

- unless otherwise stated, every element in a document comes from the same *default* namespace
- 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>

Multiple schemas may be needed to validate such a document Digital Texts with XML and the TEI 25

#### The XML declaration

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 version="1.0" ?>

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

The default, and recommended, encoding is UTF-8



### **The Doctype Declaration**

An XML document may contain a *doctype declaration* (DTD), which may include a *DTD subset* clause in square brackets.

- 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



#### An example DTD file

```
<!ELEMENT div (head*, (lg|l|lb)*, signed*)>
<!ATTLIST div n CDATA #IMPLIED
type CDATA #IMPLIED>
<!ELEMENT lg (head?, (l|lb)+) >
<!ATTLIST lg n CDATA #IMPLIED>
<!ELEMENT head (#PCDATA) >
<!ATTLIST head type CDATA #IMPLIED>
<!ELEMENT l (#PCDATA|emph|q|lb)* >
<!ELEMENT lb EMPTY >
<!ATTLIST lb n CDATA #IMPLIED>
<!ELEMENT q (#PCDATA|lb)*>
<!ELEMENT signed (#PCDATA)>
<!ENTITY mdash "&#x2014;">
```

This might be invoked using a DOCTYPE statement like the following

<!DOCTYPE div SYSTEM "filename.dtd" []>



#### (start of) Relax NG equivalent

```
<?xml version="1.0" encoding="UTF-8"?>
   <grammar xmlns="http://relaxng.org/ns/structure/1.0">
     <define name="div">
       <element name="div">
         <ref name="attlist.div"/>
         <zeroOrMore>
           <ref name="head"/>
         </zeroOrMore>
         <zeroOrMore>
            <choice>
              <ref name="lq"/>
              <ref name="l"/>
              <ref name="lb"/>
           </choice>
         </zeroOrMore>
         <zeroOrMore>
           <ref name="signed"/>
         </zeroOrMore>
       </element>
     </define>
     <define name="attlist.div" combine="interleave">
       <optional>
         <attribute name="n"/>
       </optional>
       <optional>
Dig
         <attribute name="type"/>
       </optional>
       / -
```



## (start of) Relax NG equivalent

```
div =
  element div { attlist.div, head*, (lg | l | lb)*, signed* }
attlist.div &=
  attribute n { text }?,
attribute type { text }?
lg = element lg \{ attlist.lg, head?, (l | lb) + \}
attlist.lq &= attribute n { text }?
head = element head { attlist.head, text }
attlist.head &= attribute type { text }?
l = element l \{ attlist.l, (text | emph | q | lb)^* \}
attlist.l &= empty
lb = element lb { attlist.lb, empty }
attlist.lb &=
attribute n { text }?}
q = element q { attlist.q, (text | lb) * }
attlist.g &= empty
emph = element emph { attlist.emph, (text | lb)* }
attlist.emph &= empty
signed = element signed { attlist.signed, text }
attlist.signed &= empty
start = \langle div \rangle
```



#### **Second exercise**

Using a schema to markup The Fly poem.

- The script is in your handouts
- This exercise uses the same toy schema we described above
- How useful do you think this schema is?



#### What is a schema for?

- To get the best out of XML, you need two kinds of :
  - document type declaration: elements, attributes, entities, notations (syntactic constraints)
  - document type definition: usage and meaning constraints on the foregoing
- Published specifications (if you can find them) for XML DTDs usually combine the two, hence they lack modularity



# **Some typical scenarios**

1. Make up your own DTD

- starting from scratch
- by combining components from one or more pre-existing conceptual frameworks (aka architecture or namespace)
- 2. Customize a pre-existing DTD
  - definitions should be meaningful within a given user community
  - declarations should be appropriate to a given set of applications

The TEI is a good candidate for the second approach



#### The T E what?

- 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
  - Ianguage 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 for DTD customization.



#### **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 combined into a very loose document type declaration
- A mechanism for creating multiple views (DTDs) of the foregoing
- One such view and associated tutorial: TEI Lite (http://www.tei-c.org/TEI/Lite/)

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)

