Digital Texts with XML and the TEI
Part 2: Taming the TEI Tiger

Lou Burnard / Sebastian Rahtz

February 2004

Today's topics

- The TEI Header
- The TEI and its architecture
- Working with the schema generator

In today's exercises, you'll create a TEI header with emacs, and also build your very own schema.

XML : a licence for ill?

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

- ☛ The XML concept is dangerously powerful:
  - ☛ SGML (and 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 still too good to ignore
  - ☛ mainstream software development
  - ☛ proliferation of tools
  - ☛ the future of the web

DTD : what does it mean?

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

Some answers

- Rolling your own schema
  - ... starting from scratch
  - ... by combining snippets, preferably from an existing conceptual framework (aka **architecture**)
- customizing someone else's schema
  - **definitions** should be meaningful within a given user community
  - **declarations** should be appropriate to a given set of applications
- The TEI provides a good candidate architecture

Designing a schema for the TEI

- How can a single mark-up scheme handle a large variety of requirements ?
  - all texts are alike
  - every text is different
- Learn from the database designers
  - one construct, many views
  - each view a selection from the whole

How many schemas do we need?

- one (the Corporate or WKWBFY approach)
- none (the Anarchic or NWEUMP approach)
- as many as it takes (the Mixed Economy or XML approach)

or a single main schema with many faces (a British schema)

The core tagsets

- detailed metadata provision: the TEI Header
- tags for a large set of common textual requirements:
  - paragraphs
  - highlighted phrases
  - names, dates, number, abbreviations...
  - editorial tags
  - notes, cross-references, bibliography
  - verse and drama

The base tagsets

- define basic high-level structure of document
- one must be chosen from:
  - prose, verse, or drama
  - transcribed speech
  - dictionaries
  - terminology
- or combine two or more using either of
  - the general base (anything anywhere)
  - the mixed base (homogenous divisions)

TEI additional tagsets

- sets of elements for specialised application areas
- can be mixed and matched ad lib
- currently provided:
  - linking and alignment; analysis; feature structures;
  - certainty; physical transcription; textual criticism,
  - names and dates; graphs and trees; figures and tables;
  - language corpora....
- in preparation...
  - manuscript description

The Chicago Pizza Model

A useful metaphor for expressing modularity. To build a TEI pizza, take...

- the core tagsets
- the base of your choice
- the toppings of your choice
- your own extensions

How does this model work?

- Use of modular sections within the schema
- declarations for each element are enclosed in a pattern which can be redefined
- the status of patterns can be over-ridden in your own schema
- Use of parameterised class system

An example

In a schema we write

```
include "tei.rnc" {
  p = element parágrafo { content.p }
}
include "general.rnc"
include "figures.rnc"
include "linking.rnc" {
   ab = notAllowed
}
```

which includes two modules; does one renaming; and excludes on element.

Element Classes

- Most TEI elements are assigned to one or more
  - **element classes**, identifying their syntactic properties, or
  - **attribute classes**, identifying their attributes
- In the schema, each class is represented by a *pattern*
- This provides a (relatively) simple way of
  - documenting and understanding the schema
  - modifying content models
  - facilitating customization
- An alternative way of doing *architectural forms*

Some TEI model classes

- `divn`: structural elements like divisions (`<div>`, `<div>`, `<div2>`...)
- `divtop`: elements which can appear at the start of a `divn` element (`<head>`, `<epigraph>`, `<byLine>`...)
- `chunk`: paragraph-like elements (`<sp><p><lg>`...)
- `phrase`: elements which appear within chunks (`<hi>`,`<foreign>`, `<date>` ...)

TEI attribute classes

- `global`: attributes which are available to every element (n, lang, id, TEIform)
- `linking`: attributes for elements which have linking semantics (targType, targOrder, evaluate)

The TEIFORM attribute

Two main usages...

- protect applications from the effect of element renaming

```
<titolo TEIform="title">...</titolo>
```

- protect applications from the effect of syntactic sugar

```
<tag type="xyz">
```

can be rewritten as

```
<xyz TEIform="tag">
```

A case study: the Lampeter corpus

See http://www.tu-chemnitz.de/phil/english/real/lampeter/lamphome.htm (or look in the Oxford Text Archive)

- Fairly typical requirements for language corpora
  - light presentational tagging
  - structural markup for access
  - demographic information about text production
  - small number of tags to ease data capture and validation
- Implementation
  - tagsets: prose base, and tags from four additional sets
  - some extensions, many exclusions

The Lampeter corpus view of the TEI

```
include "tei.rnc"
include "general.rnc"
include "corpus.rnc"
include "figures.rnc"
include "transcr.rnc"
include "linking.rnc"
```

The Lampeter corpus extensions

```
analytic = notAllowed
biblStruct = notAllowed
# hic desunt multa
supplied = notAllowed
class.phrase |= it
class.phrase |= ro
class.phrase |= sc
class.phrase |= su
class.phrase |= bo
class.phrase |= go
class.biblPart |= printer
class.biblPart |= pubFormat
class.biblPart |= bookSeller
class.demographic |= socecstatusPat
class.demographic |= biogNote
```

The Lampeter corpus extensions (2)

```
it =
   element it {
      attributes.class.global, macro.phraseSeq
      }
#Similar definitions for :
# ro sc su bo go
# printer pubFormat
# bookSeller biogNote sococstatusPat
```

Three types of customization

1. Kill an element

```
 ab = notAllowed
```

2. Add a new element to a class

```
MyList = element MyList {
  attributes.class.global, (item)+
}
```

3. Rename an element

```
 p = element parágrafo { content.p }
```

Moral: using the TEI for authoring

A DTD for authoring should be

- prescriptive rather than descriptive
- closely tied to current authoring practice
- *very* easy to use

This suggests that we need

- contentfull tagging
- only the tags we need
- all the tags we need

Contentfull tagging

Which is better for the authorer:

```
<list type='steps'>
<item n="1">Log in to the network
with your course username and
password.</item>
<item n="2">Start Netscape by
double-clicking on its icon.</item>
<!-- ... -->
</list>
```

or

```
<stepList>
<step n="1">Log in to the network
with your course username and
password.</item>
<step n="2">Start Netscape by
double-clicking on its icon.</item>
<!-- ... -->
</stepList>
```

?

All and only

Unmodified TEI offers authorers too many choices:

- four different types of bibliographic citation
- three (or four) different tags for proper names
- an indigestably rich choice of text editing tags

At the same time, unmodified TEI lacks

- detailed table model
- detailed tags for mathematical and other formulae
- front matter for modern publications
- tags for multimedia objects

all this can be addressed by TEI customization

Where are extensions needed for authoring?

Tables the TEI's minimalist model sweeps all the complexity into an already over-loaded `rend` attribute

Maths and other scientific notations (TEI assumes you will use an external notation)

Algorithmic graphics (The Death Of The Embedded Graphic)

Front matter for documents other than early printed books, e.g. STM articles

Office documents and other things 'born digital'

Two office documents

```
<! - a memorandum marked up in TEI -->
<!ENTITY u-shortsight SYSTEM "http://www.ourcompany.com/policy/" ND

<text>
<front>
 <opener type="from"><name>Ty Coon</name></opener>
 <opener type="to"><name>Ev Angelist</name></opener>
 <date>Today</date>
</front>
<body><div>
<p>Re your memorandum of <date>July 21st</date>, I
think that the chance of us switching to XML in
this company is minimal. See <xptr doc="u-shortsight"/>.
</p>
</div></body>
</text>
```

```
<!-- a business letter marked up in TEI-->
<text><body>
<p><address><!-- ... -->
<salute>Dear Ty</salute>
<p>Do you realize that the word-processor stored
your memo to me marked up in XML?</p>
<signature>Ev</signature>
</body></text>
```

Possible practical answers

We may need to do some or all of:

- Define extensive additional tagsets, possibly containing much syntactic sugar, for new domains
- Suck in external DTDs, like MathML, SVG, and XHTML tables and forms (but we will need to address name clashes and universal namespace support may be a while coming)
- Use all and only those parts of the TEI we need to avoid tag overload for authors
- Add convenience attributes (eg to bypass purist XLink markup for URLs)

The author vs the editor?

Hold on: do we need to use the same DTD for authoring, for archive, for editing, for production? The TEI philosophy allows us:

1. Develop sample documents for a new domain using generic tools like `<div>` and `type` attributes
2. Generate a private *authoring* DTD which uses domain-specific language:

```
<! – memorandum marked up in TEIMEMO -->
<memo>
<front>
 <from_opener>Ty Coon</from_opener>
 <to_opener>Ev Angelist</to_opener>
 <date>Today</date>
</front>
<body>
<div>
<p>Re your memorandum of <date>July 21st</date>, I think that

the chance of us switching to XML in this company is minimal.

See <xptr url="http://www.ourcompany.com/policy/"/>.
</p>
</div>
```

Why bother?

- The TEI is a well-known reference point
- Using the TEI enables
  - sharing of data and resources
  - shared modular software development
  - lower learning curve and reduced training costs
- The TEI is stable, rigorous, and well-documented
- The TEI is also flexible, customizable, and extensible *in documented ways*
- The architectural approach offers the best compromise for practical work.