TEI P5 Progress Report

Sofia, October 2005

TEI, a new phase

The P5 release of the TEI Guidelines has three aims:

Interoperability taking advantage of the work done by others

Expansion addressing areas as yet untamed

Internal audit cleaning up the accretions of a decade

... all without losing touch with its core constituency

Are we nearly there yet?

- Infrastructural developments
- What's new so far?
- Customization and Modularity
- Internationalization

Infrastructural developments

- The TEI editors' toolkit:
  - more than one XML editor
  - a library of XSLT scripts
  - a version control system
  - a test suite
  - self-validating source and examples
- Working practices:
  - the workgroup model
  - role of the council
  - snapshot releases
    - Feb 2005
    - Aug 2005
    - ... and Oct 2005
- Opening the TEI: moving the source to sourceForge

What's new so far?

- New modules for gaiji, msDescription
- The war on attributes
- Linking mechanisms
- Attribute datatypes
- The class struggle

But first... what's in the draft?

New and forthcoming content in TEI P5

New

- schema documentation and generation
- manuscript description
- `<choice>`, `<index>`, `<graphic>` etc.
- feature structures (now ISO 24610)

Forthcoming

- structure chapter
- "personography"
- handling of overlap
- dictionaries
- terminologies
- relation of header to other metadata standards

Gaiji: is your journey really necessary?

- Getting rid of `&wibble;` in favour of the actual character (or `&#xxxx;`) is highly recommended
- If you *really* need to use non-Unicode characters...
  - wherever text is possible as content, <g> can be used, either as a pointer, or to hold any convenient representation
  - nonstandard characters and glyphs can now be defined in the header
- we now use xml:lang (just as we now use xml:id)

Documenting your use of the private use area


```
<charDesc>
<glyph xml:id="z103">
<glyphName>LATIN LETTER Z WITH TWO STROKES</glyphName>
<mapping type="standardized">Z</mapping>
<mapping type="PUA">&#E304;</mapping>
</glyph>
</charDesc>
```

We may now refer to

```
<g ref="#z103"/>
```

and expect the processing application to work out what to do

Character documentation for glyph variants


```
<charDesc>
  <glyph xml:id="r1">
  <glyphName>LATIN R WITH ONE FUNNY STROKE</glyphName>
   <charProp>
      <localName>entity</localName>
       <value>r1</value>
   </charProp>
   <graphic url="r1img.png"/>
 </glyph>
  <glyph xml:id="r2">
  <glyphName>LATIN R WITH TWO FUNNY STROKES</glyphName>
   <charProp>
      <localName>entity</localName>
       <value>r2</value>
   </charProp>
   <graphic url="r2img.png"/>
 </glyph>
</charDesc>
```

The war on attributes

- an attribute value cannot contain markup
- the language of an element's content and its attributes must be the same

Work started with the `<choice>` element to replace "mirror" tags; now complete-ish:

```
<sic corr="what!?">whaaa</sic>
<choice><sic>whaaa</sic><corr>what!?</corr></choice>
```

```
<gap desc="transcriber dozes off"/>
<gap><desc>transcriber dozes off</desc>
 <desc lang="fr">transcripteur s'endort</desc></gap>
```

Linking mechanisms

- P4 had two different ways of linking:

  internal `<ptr>`: using ID/IDREF
  external `<xptr>` : using TEI-invented syntax

- But the world has moved on!
- In P5, all pointing is done in the same way, using URI
- A URI may be absolute...

```
<ptr target="http://www.tei-c.org/P5/Guidelines/SA.html"/>
```

- .. relative (the base is value of xml:base)...

```
<list xml:base="http://www.tei-c.org/Members/">
  <item><ref target="2005-Sofia">this meeting</ref></item>
  <item><ref target="2004-Baltimore">last year's</ref></item>
</list>
```

- .. or you may use a "bare name"

```
<sp who="#Macbeth"><speaker>Mac.</speaker> ...
```

- and other XPointer framework schemes may be used

Other XPointer framework schemes

Six new XPointer schemes defined:

- xpath()
- left(), right()
- range()
- string-range()
- match()

```
<ref xml:base="http://www.tei-c.org/Talks/2005/Sofia/"
  target="p5report.xml#range(xpath(//div[12]/list/item[1]),
          xpath(//div[12]]/list/item[5])">
the six added schemes
</ref>
```

Attribute datatypes

- attribute values are now declared by referring to a TEI datatype
- each TEI datatype maps to a W3C XML Schema datatype, and can therefore be validated by regular XML software
- the indirection makes it easier for users to make customizations (and editors to make changes!)
- Currently defined TEI datatypes:

  normalized expressions of quantity certainty, probability, numeric, count

  other normalized values duration, temporal, truthValue, language, sex

  specialized pointers outputMeasurement, namespace, pattern, pointer, pointers

  symbolic names key, word, words, name, names, enumerated, code

Customization

The TEI Guidelines, its DTD, and its schema fragments, are all produced from a single XML resource containing:

1. Descriptive prose (lots of it)
2. Examples of usage (plenty)
3. Formal declarations for components of the TEI Abstract Model:
   - elements and attributes
   - modules
   - classes and macros
4. We call this resource an ODD (One Document Does it all) although the master source is instantiated as a gazillion XML mini-documents.

So what?

The TEI scheme can only be used by customizing it.
Customizations are also expressed in the ODD language
For example:

```
<schemaSpec ident="myTEIlite">
<desc>This is TEI Lite with simplified heads</desc>
  <moduleRef key="core"/>
  <moduleRef key="tei"/>
  <moduleRef key="textstructure"/>
  <moduleRef key="header"/>
  <moduleRef key="linking"/>
  <elementSpec ident="head" mode="change">
    <content><rng:ref name="model.text"/></content>
  </elementSpec>
</schemaSpec>
```

produces the schema for TEI Lite, with a slight change

The TEI abstract model

- Each element declares the module it belongs to: elements cannot appear in more than one module.
- A markup system (a schema) consists of a number of discrete modules, which can be combined more or less as required.
- A schema is made by combining references to modules with other declarations.
- Each module extends the range of elements and attributes available by adding new members to existing classes of elements.

The rise of the class system (1)

- Class membership can do two distinct things for an element:
  1. attribute classes, named att.xxxx, give its members some attributes:
  2. model classes, named model.xxxx, allow its members to join a ‘club’
- Content models reference ‘clubs’ rather than specific elements (wherever possible)
- There are two ways of naming a club:

model.xxxLike elements which are semantically like an xxxx (but fraternize with others)

model.xxxPart sibling elements which constitute an xxxxx

The class struggle

Consider

```
foo (bar|baz|bam|zip)*
```

We could say both

- <foo> contains barLike elements
- <bar> etc. are members of the fooPart class

Either way, we redefine the content model:

```
foo (model.barLike)*
```

The P4 content models offer *a lot* of scope for simplification of this kind...

The rise of the class system (2)

- Classes are easier to understand and remember than elements
- Adding a new element becomes a matter of deciding what it is ‘like’, or what it is a ‘part’ of
- Specialization of the TEI generic structure for specific needs becomes a simple declarative matter

Why the stress on customization?

The TEI has over 20 modules. A working project will:

- Choose the modules they need
- Probably narrow the set of elements within a module
- Probably add local datatype constraints
- Possibly add new elements
- Possibly localize the names of elements

We can do all that in an ODD

Our gestures towards ontological mapping

The `<equiv>` element can supply a URI which identifies an equivalent concept (*not* a name) in some externally-defined ontology, e.g.

- ISO data category registry
- CIDOC conceptual reference model
- Wordnet

It can also be used to specify a stylesheet transformation where syntactic sugar has been applied, for example to specify formally that `<placeName>` is equivalent to `<name type="place">`

Open TEI

- The TEI consortium now releases the Guidelines under a GNU Public license
- All development now takes place in public using CVS on Sourceforge
- Feature requests and bug tracking are also on Sourceforge
- TEI components are available as Debian Linux packages

However, the name TEI remains a trademark, and technical work continues to be authorized by TEI Technical Council, elected by members of the Consortium.

Open TEI: what does it mean?

- The TEI remains a community initiative, driven by the needs of its members and users
- To encourage more devolved development we need to build a larger community of developers
- This means both making entry level development easier and peer approval more visible
- Which means we need more participation from all potential TEI users, as members of SIGs, Workgroups, and Council ...

What's on the horizon?

- I18N and L18N
- Ontological mapping
- FAND
- Resolving the Durand Conundrum
- .... over to you!