Editorial Work Programme beyond P4

Lou Burnard, Syd Bauman and their rottweiler Sebastian Rahtz
October 2002
The aim of this talk

How do we get from P4 to P5?

What we need to do now?

[Yes, these slides are authored in TEI XML]
P3, P4, P5

☞ P3, aka *The Big Green Books* first published in April 1994 and last revised May 1999: the final deliverable from the ACH-ALLC-ACL project begun in Poughkeepsie in 1990

☞ P4, aka *The XML Version* published April 2002: this SGML/XML version will be frozen

☞ P5, aka *TEI — the next generation* has no publication date yet. It will not necessarily retain compatibility with P4; transition aids will be provided.
P4: The XML Edition

Unlike P3, it is itself expressed (and processed) using XML

Its DTD can be configured as SGML or as XML

Parts now considered obsolescent are flagged as such, but retained

Compatibility with P3 DTDs is retained: a P3-conformant document is *ipso facto* P4-conformant

http://www.tei-kern-z@c.org/Guidelines/ ... and on your CD ... and in print
From P4 to P5

☛ the process: how should we decide what to change?

☛ the content: what topics should we (in fact) change?

☛ the format: in what form should the Guidelines be maintained?

☛ the output: what should the Guidelines be, as well as text?
Producing P5: the revision process

corrigible errors
☛ dealt with ad hoc by the editors; see EDW70

emer gent areas
☛ specific workgroups, set up by and reporting to Council

known issues
☛ RFC process managed by editors, leading to formal proposals to Council
Emergent areas

1. Character encoding issues [workgroup]
2. Manuscript description and transcription
3. Standoff markup; XLink and TEI Xpointers [workgroup]
4. Termbanks, dictionaries, ontologies
5. Schemas and other enhanced content validation
6. Best practice in migrating SGML to XML
7. Best practice in TEI training
8. Authoring of technical documents (Son of ODD)
9. Plus new areas we haven’t thought of
New areas we haven’t thought of

(By definition, I haven’t anything much to say about these, but this is the list I showed last year . . . )

» tags for authoring
» tags for éditions génétiques
» tags for multimodal communicative acts
» develop feature structure annotation
» develop metrical annotation scheme
» develop image annotation scheme
Known issues

➲ Things which have annoyed at least one TEI user enough for them to have proposed a fix

➲ An expandable list, for example:

➲ consistent treatment for dates and times
➲ review/rationalize divtop and divbot elements
➲ review/rationalize the a.edit class
➲ recommended practice for linking page images
➲ verse and drama should be additional modules, not bases
➲ epistles and charters
Son of ODD: the shopping list

An XML vocabulary which can both describe and generate encoding schemes expressed in XML, SGML, RelaxNg, W3Schema...

- which can be used to author technical documentation

- which can be used to deliver searchable versions of P5, in a variety of formats (HTML, PDF, etc)

- and which is easily maintained with free open source software
From P3 to P4

ODD2: One Document Does it all (in XML) . . . .
Schemata and enhanced validation

- In P3, we had DTDs only, with some prose specification of additional constraints (for TEI conformance...)
- In P4, XMLification mostly meant removing such few constraints as were available in DTD content models
- In P5, we could introduce full schematron-like content validation
- This would require modification to the ODD system, but nothing much beyond what is already implicit
- It could also generate multiple schemas, with specific associated documentation
Architectural issues

- how do we integrate other XML vocabularies?
- how do we support customization while retaining the facility for interchange?

The pizza chef does this by assimilating your changes to its own structure. XML namespaces do this at the price of losing structural validation.
Rethinking the Tower of Babel

To keep the tower standing we need

- a metalanguage able to co-ordinate and manage different validation technologies
- an intellectual and technical framework within which different vocabularies can be mutually intelligible
- the will to accept multiple interpretations at the metalanguage level and to avoid assertions of ownership
How do we get there?

 Namespace and TEIform are the best solutions we have so far: we need better ones.

 How viable or well-defined are our work group and modification procedures?

 Where will the software development take place?

 What new technologies can we leverage to facilitate this?

 We start by thinking about the Son of ODD, which can also serve as a candidate for the new authoring module.
TEI Literate Programming

The TEI is not a set of DTDs, but an abstract markup model. We formalize the markup constraints in a small XML language \textit{in the documentation}, to generate:

1. XML user documentation containing embedded DTD fragments as needed
2. XML documentation for the elements and attributes in an alphabetic reference manual
3. the actual DTD modules defining the markup language

Following Knuth’s Tangle and Weave, we have ODD: One Document Does it all
ODD basics

<tagDoc> a special-purpose element for documentation of a single TEI element
<classDoc> an element which documents a single TEI class.

Elements in the TEI scheme are assigned to one or more classes of three distinct kinds:

*model classes* elements with similar structural properties, i.e. they can appear at the same point in the document hierarchy

*attribute classes* elements with common attributes

*semantic classes* elements with similar semantic properties
<bibl> in ODD

The <tagdoc> for the <bibl> element:

```xml
<tagDoc id="BIBL" usage="opt">
  <gi>bibl</gi>
  <name>bibliographic citation</name>
  <desc>contains a loosely-structured bibliographic citation of which the components may or may not be explicitly tagged.</desc>
  <attList/>
  <exemplum>
    <bibl>Blain, Clements and Grundy: Feminist Companion to Literature in English (Yale, 1990)</bibl>
  </exemplum>
  <part type="base" name="core"/>
  <classes names="CLBIBL DECLABL"/>
  <elemDecl> %om.RO; (#PCDATA | %m.phrase; | %m.biblPart; | %m.Incl;)*</elemDecl>
</tagDoc>
```
When run through the DTD generator, the following declaration is produced from this tagdoc element:

```xml
<!ELEMENT %n.bibl; %om.RO; (#PCDATA | %m.phrase; | %m.biblPart; | %m.Incl;)>  
<!ATTLIST %n.bibl;  
  %a.global;  
  %a.declarable;  
  TEIform CDATA 'bibl' >
```
Why is that a bad thing?

The guts of the content model are PCDATA and cannot be taken apart as XML. So

- We cannot validate the content internally
- We cannot validate it against other content models
- We cannot (easily) convert it to other constraint languages
- It does not (currently) represent the target DTD properly
<bibl> in Relax NG

<elemDecl omit="RR">
  <rng:group xmlns:rng="http://relaxng.org/ns/structure/1.0">
    <optional>
      <ref name="head"/>
    </optional>
    <oneOrMore>
      <choice>
        <ref name="bibl"/>
        <ref name="biblStruct"/>
        <ref name="biblFull"/>
      </choice>
    </oneOrMore>
    <optional>
      <ref name="trailer"/>
    </optional>
  </rng:group>
</elemDecl>
Constraint 1: datatyping

The `<birth>` element currently has a promising attribute ‘date’, defined as

```xml
<attribute name="date">
  <ref name="ISO-date"/>
</attribute>
```

but ISO-date turns out to be simply defined as

```xml
<define name="ISO-date">
  <text/>
</define>
```

ie plain text. If we change that to

```xml
<define name="ISO-date">
  <data type="date"
    datatypeLibrary="http://www.w3.org/2001/XMLSchema-datatypes"/>
</define>
```

the schema processor will check our birth dates to see if they are valid.
Constraint 2: element content

Kohsuke Kawaguchi’s Sun Multi-Schema XML Validator supports Schematron constraints on element content:

```xml
<define name="email">
  <element name="email">
    <ref name="a.global"/>
    <ref name="paraContent"/>
    <s:assert test="contains(.,@')">
      email address must contain an @ sign
    </s:assert>
  </element>
</define>
```

The `<s:assert>` adds a test beyond the normal element content with a test (expressed in XPath) which must be passed. More complex examples can clearly go beyond regexp-like tests which could have been done as a schema datatype.
Constraint 3: either/or

In an `<xref>` you can have a url or a target attribute, but not both.

```xml
<define name="al.xref" combine="interleave">
  <ref name="a.global"/>
  <ref name="a.xPointer"/>
  <choice>
    <attribute name="url"><text/></attribute>
    <attribute name="target"><data type="IDREF"/></attribute>
  </choice>
  <optional>
    <attribute name="TEIform" a:defaultValue="xref">
      <text/>
    </attribute>
  </optional>
</define>
```
Constraint 4: vocabularies in other namespaces

```xml
<xsp:page
  language="Perl"
  xmlns:xsp="http://apache.org/xsp/core/v1">
  <TEI.2>
    <teiHeader>...
    </teiHeader>
    <text>
      <body>
        <p>Page delivered
        <xsp:expr>scalar localtime</xsp:expr>.
        For further details, see
        <xptr url="http://www.oucs.ox.ac.uk/">.
        </p>
      </body>
    </text>
  </TEI.2>
</xsp:page>
```
Schema for alternate namespaces

```xml
<grammar
   xmlns="http://relaxng.org/ns/structure/1.0"
   xmlns:xsp="http://apache.org/xsp/core/v1"

  <include href="rng/tei2.dtd.rng">
    <define name="TEI.prose">
      <ref name="INCLUDE"/>
    </define>
    <start combine="choice">
      <choice>
        <ref name="xsp"/>
        <ref name="TEI.2"/>
      </choice>
    </start>
  </include>
</grammar>
```
What did ‘xsp’ mean there?

```xml
<define name="xsp">
   <element name="xsp:page">
      <oneOrMore>
         <attribute><anyName/> <text/> </attribute>
      </oneOrMore>
      <ref name="anyXSP"/>
      <ref name="TEI.2"/>
   </element>
</define>

<define name="anyXSP">
   <element>
      <nsName ns="http://apache.org/xsp/core/v1"/>
      <zeroOrMore>
         <choice>
            <attribute>
               <anyName/>
            </attribute>
            <text/>
            <ref name="anyXSP"/>
         </choice>
         </zeroOrMore>
   </element>
</define>
```
Constraint 5: domains attribute of `<ptrGroup>`

The text says

If this attribute is supplied every element specified as a target must be contained within the element or elements specified by it. An application may choose whether or not to report failures to satisfy this constraint as errors, but may not access an element of the right identifier but in the wrong context. If this attribute is not supplied, then target elements may appear anywhere within the current document.

Can I turn this into a formal constraint?
Relax NG for TEI: Resources

TEI www.tei-c.org

TEI Schemas www.tei-c.org/Schemas/RelaxNG

Relax NG www.relaxng.org

Multi-Schema XML Validator
www.sun.com/software/xml/developers/multischema/,
with Schematron addon at
www.sun.com/software/xml/developers/schematronaddon/
Pizza Problems

The underlying program, *carthago*, is written in Lex, common implementations of which do not support Unicode.

The input to *carthago* must be written in SGML. Although it can emit XML, it will also accept SGML models which cannot be translated into XML.

The web interchange is showing its age and needs an even dummier interface.

And how shall we express constraints for a schema-based TEI?
Towards Roma

Do we still need a flattener if we move to schema? The benefits of flattening are

☛ The P4 DTDs take SGML/XML DTD to the limit, and beyond the capability of some processors

☛ The flattened DTD is (usually) smaller and makes less load on the processing application

☛ Removing the parameterisation makes it impossible for individual documents to tinker with the scheme

Are all these still valid if we use schema?
Ways to Roma

❖ Update *carthago* to support Unicode, and carry on as we are
❖ Allow for alternative input forms (XML, SGML, RelaxNG, etc) and convert them to an internal format for a new *carthago*
❖ Switch the input format to eg RelaxNG and make everyone learn that
❖ Do not support upload of modification files but do everything with wizard Q & A
❖ Abandon the Pizza project entirely
The timetable

.errr

Son of ODD: promised by end 2003

Documentation and beta testing must before then for work groups to use it