

Relax NG with Son of ODD, or What the TEI did Next

Lou Burnard and Sebastian Rahtz
Oxford University
Text Encoding Initiative
Extreme Markup Languages, Montréal,
August 2004



Topics

- The T E what??
- Literate programming ODD-style
- DTD vs Relax NG vs W3C Schema
- Hooks for other ontologies
- Customization to the max



Real-world problems

- ☞ keeping the documentation in step with the design
- ☞ customizing the interface to match the design
- ☞ taking advantage of generic tools for Relax NG editing and validating
- ☞ taking advantage of other people's schemas and vocabularies



The T E What?

- The **Text Encoding Initiative** was set up in the late 80s as a research project to define Guidelines for the mark up of (largely) literary and linguistic material.
 1. first release (P1) in 1990 (SGML), revised 1993, 1994
 2. fourth release (P4) in 2000, converted to XML
- The Guidelines describe nearly 500 textual elements grouped into classes and modules, and are maintained as a single XML document.
- The TEI has been and remains a major influence in the digital library and in linguistic computing generally.
- The TEI is now a membership consortium (and a sourceforge project): please join/in.



TEI, a new start

The next release of the TEI Guidelines (P5) 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



Interoperability

A lot of other people have been working in this area since 1987!

TEI P5 must fit into a joined-up digital world, along with

- W3C standards (XLink, schema, etc)
- Unicode character encoding
- Specialized markup vocabularies (MathML, SVG, DocBook, etc)
- Other metadata schemas (METS, EAD, etc)
- Other conceptual models and ontologies



Expansion and Cleanup

- ☞ ‘Fitting in with’ the TEI risks becoming a new orthodoxy: we need to promote evolutionary change.
- ☞ Some parts of TEI P4 were successfully experimental (e.g. the extended pointer syntax, corpus metadata)...
- ☞ ... some were influentially experimental and have become FAQs (frequently answered questions) e.g. synchronization and standoff
- ☞ ... others were just experimental, and have been overtaken by events (e.g. writing system declaration, feature structures, terminology...)



Literate programming ODD-style

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.



ODD processors

- ☛ We supply a library of XSLT scripts that can generate
 - ☛ The book in canonical TEI XML format
 - ☛ The book in HTML or PDF
 - ☛ RelaxNG, DTD, or W3C schema fragments
- ☛ The same library is used by the new customization layer to generate
 - ☛ project-specific documentation
 - ☛ project-specific schemas
 - ☛ translations into other (human) languages
- ☛ We are using **Perforce** to manage our CMS, and experimenting with **eXist** as a better back end than the file system



The TEI abstract model

- The TEI abstract model sees a markup scheme (a *schema*) as consisting of a number of discrete *modules*, which can be combined more or less as required.
- A schema is made by combining references to modules and optional element over-rides.
- Each *element* declares the module it belongs to: elements cannot appear in more than one module.
- Each module extends the range of elements and attributes available by adding new members to existing *classes of elements*, or by defining new classes.



The TEI class system

- Class membership can do two distinct things for an element:
 1. give it some attributes
 2. allow it to join a club
- Content models reference clubs rather than specific elements (wherever possible)
- Content models are named patterns, distinct from element names
- (There are also special named patterns for common content models such as `macro.phraseSeq`)



Expression of TEI content models

Beyond the class system, TEI elements have to be defined. How? (This is also known as the Durand Conundrum)

1. continue (as in P4) to use raw XML DTD language
2. maintain in DTD language but transform to some other schema language at the point of delivery
3. transform to some other schema language for maintenance and delivery
4. invent an entirely new abstract language for later transformation to some schema language

We chose a combination of 3 and 4 — revise our abstract language to use RelaxNG for content modelling (only).



DTD vs Relax NG vs W3C Schema

- DTDs are not XML, and need specialist software
- W3C schema is not consistently implemented, is poorly documented, and looks over-complex
- Relax NG on the other hand...
 - uncluttered design
 - good documentation
 - multiple open source 100%-complete implementations
 - ISO standard
 - useful features for multipurpose structural validation
 - Compelling leadership (can James Clark do wrong?)



What does an ODD look like?

```
<elementSpec module="spoken" ident="pause">
  <classes>
    <memberOf key="tei.comp.spoken"/>
    <memberOf key="tei.timed"/>
    <memberOf key="tei.typed"/>
  </classes>
  <content>
    <rng:empty xmlns:rng="..."/>
  </content>
  <attList>
    <attDef ident="who" usage="opt">
      <datatype><rng:data type="IDREF"/></datatype>
      <valDesc>A unique identifier</valDesc>
      <desc>supplies the identifier of the
        person or group pausing.
        Its value is the identifier of a <gi>person</gi>
        or <gi>persGrp</gi> element in the TEI header.
      </desc>
    </attDef>
  </attList>
  <desc>a pause either between or within utterances.</desc>
</elementSpec>
```



... from which we generate

```
pause = element pause { pause.content }
pause.content =
  empty,
  tei.global.attributes,
  tei.comp.spoken.attributes,
  tei.timed.attributes,
  tei.typed.attributes,
  pause.attributes.who,
  pause.newattributes,
  [ a:defaultValue = "pause" ] attribute TEIform { text }?

pause.newattributes |= empty
tei.comp.spoken |= pause
tei.timed |= pause
pause.attributes.who =
  attribute who { pause.attributes.who.content }?
pause.attributes.who.content = xsd:IDREF
```



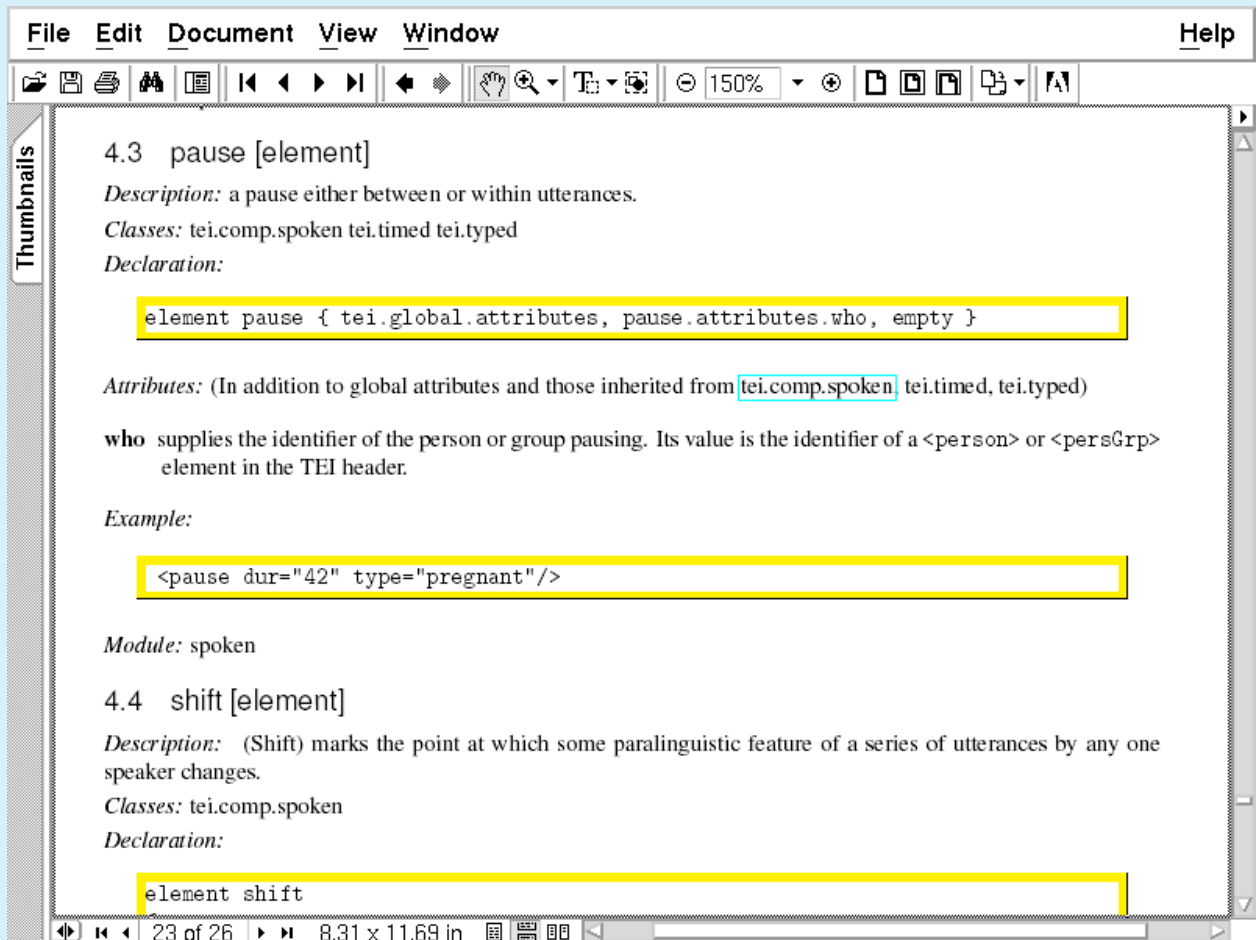
.. which translates to

```
<!ENTITY % pause 'INCLUDE' >
<![ %pause; [
<!ELEMENT %n.pause; %om.RR; EMPTY>
<!ATTLIST %n.pause;
    %tei.global.attributes;
    %tei.timed.attributes;
    %tei.typed.attributes;
    who IDREF #IMPLIED
    TEIform CDATA 'pause' >

<!ENTITY % tei.comp.spoken "%x.tei.comp.spoken;
%n.event; | %n.kinesic; | %n.pause; | %n.shift;
| %n.u; | %n.vocal; | %n.writing;">
```



... and, indeed, to



The screenshot shows a web browser window with a menu bar (File, Edit, Document, View, Window, Help) and a toolbar. The main content area displays the TEI element definition for 'pause'. The 'Declaration' section is highlighted in yellow, showing the XML schema: `element pause { tei.global.attributes, pause.attributes.who, empty }`. Below this, the 'Attributes' section is highlighted in cyan, showing: (In addition to global attributes and those inherited from `tei.comp.spoken`, `tei.timed`, `tei.typed`) **who** supplies the identifier of the person or group pausing. Its value is the identifier of a `<person>` or `<persGrp>` element in the TEI header. An 'Example' section is highlighted in yellow, showing: `<pause dur="42" type="pregnant"/>`. The 'Module' section is highlighted in yellow, showing: spoken. Below this, the 'shift' element definition is partially visible, with its 'Declaration' section highlighted in yellow, showing: `element shift`. The browser status bar at the bottom shows '23 of 26', '8.31 x 11.69 in', and a scrollbar.

File Edit Document View Window Help

150%

Thumbnails

4.3 pause [element]

Description: a pause either between or within utterances.

Classes: tei.comp.spoken tei.timed tei.typed

Declaration:

```
element pause { tei.global.attributes, pause.attributes.who, empty }
```

Attributes: (In addition to global attributes and those inherited from `tei.comp.spoken`, `tei.timed`, `tei.typed`)

who supplies the identifier of the person or group pausing. Its value is the identifier of a `<person>` or `<persGrp>` element in the TEI header.

Example:

```
<pause dur="42" type="pregnant"/>
```

Module: spoken

4.4 shift [element]

Description: (Shift) marks the point at which some paralinguistic feature of a series of utterances by any one speaker changes.

Classes: tei.comp.spoken

Declaration:

```
element shift
```

23 of 26 8.31 x 11.69 in

Generation of alternate outputs

1. Relax NG schema fragments are generated by an XSLT transform
2. ... and progressively flattened and simplified by a further set of XSLT transforms
3. DTDs, compact Relax NG, and W3C Schema are all generated using James Clark's `trang` (but not necessarily from the same inputs)

Vocabularies like MathML and SVG inclusion are managed by simply `<include>`ing the relevant RelaxNG grammars, each in their own namespace.



Customizing the TEI

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 ODD

```
<schema>  
<moduleRef name="tei"/>  
<moduleRef name="header"/>  
<moduleRef name="textstructure"/>  
<moduleref name="linking"/>  
</schema>
```



From which we can generate...

```
<grammar ns="http://www.tei-c.org/P5/"
  xmlns="http://relaxng.org/ns/structure/1.0"
  datatypeLibrary=
    "http://www.w3.org/2001/XMLSchema-datatypes">
<include href="Schema/tei.rng"/>
<include href="Schema/header.rng"/>
<include href="Schema/textstructure.rng"/>
<include href="Schema/linking.rng"/>
</grammar>
```



More interestingly..

```
<schema>
<moduleRef name="teiheader"/>
<moduleref name="verse"/>
<!-- add a new element -->
<elementSpec ident="soundClip">
<classes memberOf="tei.data"/>
  <attList>
    <attDef ident="location">
      <datatype><rng:data type="URI"/></datatype>
      <valDesc>A location path</valDesc>
      <desc>supplies the location of the clip</desc>
    </attDef>
  </attList>
  <desc>includes an audio object in a document.</desc>
</elementSpec>
<!-- change an existing element -->
<elementSpec ident="head" mode="change">
<content><rng:text/></content>
</elementSpec>
</schema>
```



Uniformity of description

- modules, elements, attributes, value-lists are treated uniformly
- each has an identifier, a gloss, a description, and one or more equivalents
- each can be added, changed, replaced, deleted within a given context
- for example, membership in the `tei.typed` class gives you a generic TYPE, which can be over-riden for specific class members



Overriding a value-list

```
<elementDecl ident="list" module="core">
  <classes>
    <memberOf key="tei.typed"/>
  </classes>
  <!--... -->
  <attDef ident="type" mode="replace">
    <valList>
      <valItem ident="ordered">Items are ordered</valItem>
      <valItem ident="bulleted">Items are bulleted</valItem>
      <valItem ident="frabjous">Items are frabjous</valItem>
    </valList>
  </attDef>
</elementDecl>
```

... not as easy as it looks (lazy evaluation rules)



Our gesture towards ontological mapping

The `<equiv>` element supplies 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



Using other vocabularies

- ➡ Namespaces help with the obvious cases (e.g. mathML, SVG...)
- ➡ But they don't help where there is overlap (e.g. HEML)
- ➡ And they enforce an Us and Them mentality
- ➡ Can we do better?



TEI and DocBook interleaved

We want a TEI document which allows DocBook elements to appear, e.g. for describing GUIs in inline contexts

We also want TEI structured inline elements to appear inside the DocBook GUI elements:

```
<TEI xmlns="http://www.tei-c.org/ns/1.0"
      xmlns:dbk="http://docbook.org/docbook-ng">
  ....
  <text>
    <body>
<p>The button on our web page has the current date:
<dbk:guibutton>
<date
  calendar="Julian" value="1732-02-22">
Feb. 11, 1731/32, O.S.</date>
</dbk:guibutton>
or at least the date on which we last updated it.</p>
    </body>
  </text>
</TEI>
```



Schema for the above

We have to be able to tell the TEI that it can use the DocBook `gui.inlines` class along with its normal phrase-level elements, and tell DocBook that it can use the TEI class `data` in *its* phrase-level class.

```
include "tei.rnc" {
  tei.hqphrase |= gui.inlines
}
include "docbook.rnc" {
  docbook.text |= tei.data
  start = TEI
}
```

We know how to do this in RelaxNG, but not yet in ODD... &winita;



Roma: a customization application

The Roma web-based service allows users to

- Choose and customize modules
- Add new elements
- Change or add class attributes
- Generate element and attributes names in various languages
- Generate reference documentation for chosen subset

Roma communicates with the TEI via XQuery calls to an eXist database of the TEI sources



Roma in action (1)

Roma: generating validators for the TEI

Te>

Customize Customization Change Modules Add Elements Change Classes Customize Language Create Schema Create Documentation Save Custom

Change module

[back](#)

List of elements in module: iso-fs

	Include	Exclude	Tag name	Description	Attributes
binary	<input checked="" type="radio"/>	<input type="radio"/>	binary	represents the value for a feature-value specification which can take either of a pair of values.	Change attributes
coll	<input checked="" type="radio"/>	<input type="radio"/>	coll	contains (or points to) several feature structures which together provide a value for their parent feature element, organized as indicated by the value of the org attribute.	Change attributes
default	<input checked="" type="radio"/>	<input type="radio"/>	default	provides default value for a feature.	Change attributes
f	<input checked="" type="radio"/>	<input type="radio"/>	f	represents a feature value specification, that is, the association of a name with a value of any of several different types.	Change attributes
fLib	<input checked="" type="radio"/>	<input type="radio"/>	fLib	assembles library of feature elements.	Change attributes
fs	<input checked="" type="radio"/>	<input type="radio"/>	fs	represents a feature structure, that is, a collection of feature-value pairs organized as a structural unit.	Change attributes
fvLib	<input checked="" type="radio"/>	<input type="radio"/>	fvLib	assembles a library of reusable feature value elements (including complete feature structures).	Change attributes
numeric	<input checked="" type="radio"/>	<input type="radio"/>	numeric	represents a numeric value or range of values for a feature.	Change attributes
string	<input checked="" type="radio"/>	<input type="radio"/>	string	provides a string value for a feature.	Change attributes
symbol	<input checked="" type="radio"/>	<input type="radio"/>	symbol	provides a symbolic feature value.	Change attributes
vAlt	<input checked="" type="radio"/>	<input type="radio"/>	vAlt	represents a feature value which is the disjunction of the values it contains.	Change attributes
				represents a feature value which is the result of	

R



Roma in action (2)

Roma: generating validators for the TEI

Customize Customization Change Modules Add Elements Change Classes Customize Language Create Schema Create Documentation S

Add Element

[go back to list](#)

Defining a new element

Name	MyElement
Model classes	<input type="checkbox"/> tei.Incl <input type="checkbox"/> tei.comp.dictionaries <input type="checkbox"/> tei.dictionaryTopLevel <input type="checkbox"/> tei.formInfo <input type="checkbox"/> tei.lists <input type="checkbox"/> tei.addrPart <input type="checkbox"/> tei.comp.drama <input type="checkbox"/> tei.divbot <input type="checkbox"/> tei.formPointers <input type="checkbox"/> tei.loc <input type="checkbox"/> tei.agent <input checked="" type="checkbox"/> tei.comp.spoken <input type="checkbox"/> tei.divtop <input type="checkbox"/> tei.fragmentary <input type="checkbox"/> tei.metad <input type="checkbox"/> tei.bibl <input type="checkbox"/> tei.comp.verse <input type="checkbox"/> tei.dramafront <input type="checkbox"/> tei.front <input type="checkbox"/> tei.morph <input type="checkbox"/> tei.biblPart <input type="checkbox"/> tei.complexVal <input type="checkbox"/> tei.edit <input type="checkbox"/> tei.gramInfo <input type="checkbox"/> tei.notes <input type="checkbox"/> tei.categorize <input type="checkbox"/> tei.data <input type="checkbox"/> groups phrase-level elements <input type="checkbox"/> tei.chunk <input type="checkbox"/> tei.date <input checked="" type="checkbox"/> for simple editorial correction <input type="checkbox"/> tei.common <input type="checkbox"/> tei.demographic <input type="checkbox"/> and transcription. <input type="checkbox"/> tei.oddDe <input type="checkbox"/> tei.oddPh <input type="checkbox"/> tei.oddRe
Attribute classes	<input type="checkbox"/> tei.analysis <input type="checkbox"/> tei.dictionaries <input type="checkbox"/> tei.interpret <input type="checkbox"/> tei.pointer <input type="checkbox"/> tei.dateable <input type="checkbox"/> tei.divn <input type="checkbox"/> tei.linking <input type="checkbox"/> tei.pointerGroup <input type="checkbox"/> tei.xPointer <input type="checkbox"/> tei.declarable <input type="checkbox"/> tei.enjamb <input type="checkbox"/> tei.measured <input type="checkbox"/> tei.readings <input type="checkbox"/> tei.declaring <input type="checkbox"/> tei.entries <input type="checkbox"/> tei.metrical <input type="checkbox"/> tei.timed
Contents	macro.component <content></content>

Ri



Roma in action (3)

Roma: generating validators for the TEI

Customize Customization Change Modules Add Elements Change Classes Customize Language Create Schema Create

Time to give you a schema

Createing a schema	
Which format do you prefer?	<input type="text" value="Relax NG schema (compact syntax)"/>
	<ul style="list-style-type: none">Relax NG schema (compact syntax)Relax NG schema (XML syntax)W3C schemaDTDWrapper Relax NG schema

CVS date: \$Date: 2004/06/21 11:34:11 \$, \$Revision: 1.9 \$

Arno Mittelbach



Things we have not mentioned

- Embedding schematron or other constraints
- A separate namespace for embedded examples (no more CDATA)
- Experimenting with Namespace Routing
Language to manage normal grammar-based validation, + example validation, +Schematron constraints



Challenges and worries

- ➡ Can we wean the TEI community from DTDs?
- ➡ What does a multi-namespace, highly configurable, modern schema do for interchange and stability?
- ➡ Will the industry take Relax NG to its heart?



Conclusions

- Constraint languages (whatever the syntax) are always going to be a minority interest
- The additional abstraction layer provided by the ODD language should help designers, users, and system developers alike:
 - by using a single vocabulary for definition and for customization
 - by applying that vocabulary to both structure and content



Resource pointers

TEI <http://www.tei-c.org/>

Experimental P5 release <http://www.tei-c.org/P5/>

Roma customization tool

<http://tei.oucs.ox.ac.uk/Roma/>

Other talks about the TEI <http://www.tei-c.org/Talks/>

ODD NG

<http://www.tei-c.org/Activities/META/FASC-td.pdf>

