

# **A unified model for text markup: TEI, Docbook, and beyond**

*Sebastian Rahtz (Oxford University),  
Norman Walsh (Sun), Lou Burnard (Oxford  
University)*

XML Europe, Amsterdam, April 2004



# Summary

Let's have a decent markup model for interchanging literate documents.

- ☞ DTDs/Schemas for text markup
- ☞ principles of mixing models
- ☞ using Relax NG
- ☞ mixing and interleaving Docbook and TEI
- ☞ linking to ISO and other conceptual frameworks



# What is the context of this work?

The *Text Encoding Initiative Consortium* Technical Council has a *Metalanguage Task Force*, tasked with developing a new generation of the TEI literate programming scheme, specifically to encompass the good work of schemas. We are all members of that task force.

In parallel, Norm is developing an experimental new version of Docbook based entirely on Relax NG schemas.

Sorry, Norm is not here!



# Which schema for textual material?

Perhaps surprisingly, few public schemas or DTDs are widely used in creating textual material:

1. HTML (and XHTML)
2. Docbook
3. Text Encoding Initiative
4. (and the proprietary schemas used by some publishers)

The first three schemes cover the areas of web pages, technical documentation, and literary and linguistic material.



# HTML, Docbook or TEI?

**HTML** *presentation*: simple layout markup, combining digital resources

**Docbook** publishing scheme, for converting between digital and print versions of newly authored documents

**TEI** representing complex existing paper documents

Distinctions are blurring. Manuscripts for publication may be authored in HTML or TEI; textual analysis of Docbook documents is feasible. Where once we had programming language wars, are we now doomed to have document schema wars?



# Specialized addons, we understand

SVG and MathML come cocooned in their own namespaces, and are (largely) designed to describe terminal nodes. we embed them in other languages, using XML namespaces:

and so we can conclude that

```
<math xmlns="http://www.w3.org/1998/Math/MathML" overflow="scroll">
```

```
  <msub>
```

```
    <mi>E</mi>
```

```
    <mrow>
```

```
      <mtext>max</mtext>
```

```
    </mrow>
```

```
  </msub>
```

```
</math>
```

measures the contribution of the collisions  
with energy transfer close to...



# Interleaving is harder

If we want to combine HTML, TEI, and Docbook it is much harder to

1. know at what level markup can be incorporated
2. at what points one can return to the outer language.

e.g. using TEI elements to mark up a play inside a Docbook document, or using Docbook elements to describe a GUI inside a TEI document.

**Not just do this, but describe it using a schema!**



# Classes are the key: (1) Docbook

In Docbook `<guibutton>` is in the class `gui.inlines`, which is in `domain.inlines`, which is in `inlines`.

```
inlines =
text
  | general.inlines
  | domain.inlines
  | extension.inlines
domain.inlines =
  technical.inlines
  | error.inlines
  | gui.inlines
...
gui.inlines =
  db.guiicon
  | db.guibutton
...
db.guibutton =
  element guibutton
  { guibutton.attlist, (docbook.text | db.accel)* }
```





# Classes are the key: (2) TEI

The TEI uses a Literate Programming scheme:

`<date>` is defined as being a member of the `tei.data` and `tei.date` classes, and it can contain elements from the `macro.phraseSeq` collection

```
<elementSpec ident="date" usage="opt"
  xmlns="http://www.tei-c.org/ns/1.0" >
  <classes>
    <memberOf key="tei.data"/>
    <memberOf key="tei.date"/><
  </classes>
  <content>
    <rng:ref name="macro.phraseSeq"/>
  </content>
  . . . .
</elementSpec>
```

which expands into a schema comparable to Docbook



# Can DTDs express interleaving?

The principle customization feature of XML DTDs is the parameter entity. But

1. Parameter entities are just strings and once defined can never be extended, adjusted, or changed in any way.
2. Parameter entities must be defined before they are used.
3. You can't mix several choices of elements into a content model that has mixed content.
4. There's no indirection. You can't indicate that a particular element simply isn't allowed.



# Relax NG does it better

A few features of Relax NG patterns that enable a design that will allow us to mix TEI and DocBook in sensible ways:

- ➡ Patterns can be extended:

```
class.hqphrase = emph | tag | gloss  
class.hqphrase |= acronym
```

- ➡ There's a “null” pattern: the `notAllowed` pattern matches nothing. You can define extension points that are explicitly empty.
- ➡ Patterns are not required to be deterministic. Relaxing this constraint allows us to add elements to patterns without concern for whether those patterns appear in choice groups elsewhere.



# Docbook/TEI Intersections

Relax NG wrapper schemas can load patterns from both Docbook and TEI, and redefine element classes to allow controlled interleaving of markup. But where does it makes semantic sense to move between languages? This is plainly silly:

```
<dbk:article>
  <tei:date>13th February
    <dbk:guimenu>...</dbk:guimenu>
  </tei:date>
</dbk:article>
```

The languages must define interchange points, and then consider how they relate



# Typical Docbook classes

**inlines** ubiq.inlines general.inlines domain.inlines  
extension.inlines

**general.inlines** publishing.inlines product.inlines  
bibliography.inlines graphic.inlines indexing.inlines  
link.inlines glossary.inlines

**domain.inlines** technical.inlines error.inlines os.inlines  
programming.inlines markup.inlines math.inlines  
gui.inlines keyboard.inlines



# Typical TEI public classes

**tei.addrPart** groups elements which may constitute a postal or other form of address.

**tei.agent** groups elements which contain names of individuals or corporate bodies.

**tei.bibl** groups elements containing a bibliographic description.

**tei.biblPart** groups elements which can appear only within bibliographic citation elements.

**tei.complexVal** groups elements which express complex feature values in feature structures.

**tei.data** groups phrase-level elements containing names, dates, numbers, measures, and similar data.

**tei.date** groups elements containing a date specifications.



# Docbook block-level classes

- ☞ admonition.blocks
- ☞ extension.blocks
- ☞ formal.blocks
- ☞ graphic.blocks
- ☞ informal.blocks
- ☞ list.blocks
- ☞ para.blocks
- ☞ publishing.blocks
- ☞ synopsis.blocks
- ☞ technical.blocks
- ☞ verbatim.blocks



# TEI blocks according to structural properties

**tei.common** This class defines the set of chunk- and inter-level elements available in all bases.

**tei.dramafront** groups elements which appear at the level of divisions within front or back matter of performance texts only.

**tei.fmchunk** groups elements which can occur as direct constituents of front matter, when a full title page is not given.

**tei.front** groups elements which appear at the level of divisions within front or back matter.

**tei.teiText** Main element covering the body of a TEI document





# Doh! so what?

The TEI and Docbook both work by grouping elements in a hierarchy of classes:

- ➡ Docbook inline elements can be used in any of the TEI 'phrase-like' situations, and the TEI technical classes can be used alongside Docbook inlines
- ➡ the main block-level classes are comparable (eg TEI `tei.lists` and `list.blocks`)
- ➡ The highest level Docbook info classes should be interchangeable with TEI's `<teiHeader>`.

Now let's make some schemas!



# Practical examples

Five scenarios to model using a mixture of Docbook and TEI; not to demonstrate real-life situations, but to cover four basic operations:

1. adding an individual element from one schema to a class in another scheme
2. replacing a class content model from one schema with that from another
3. extending a class content model with alternatives from another scheme
4. interleaving classes of elements from both schemes



# How it works in RelaxNG

The basic technique used is to extend or replace an existing class; in RelaxNG, a class is implemented as a pattern containing a choice. Thus we can group `<para>`, `<quote>` and `<list>` by defining a pattern `blocks` as follows:

```
blocks = para | quote | lists
```

We can then add another choice to the class by writing

```
blocks |= speech
```

So any reference to `blocks` now means `para | quote | lists | speech`.



# Docbook adding to TEI

A TEI document uses the Docbook `<qandaset>` element inside sections to ask a basic question:

```
<div xmlns="http://www.tei-c.org/ns/1.0">
<head>In normal TEI mode </head>
<p>Marley was dead: to begin with. There is no doubt
whatever about that.....Old Marley was as
dead as a door-nail.</p>
</div>

<div rend="slide">
<head>Docbook QandA example</head>
<qandaset xmlns="http://docbook.org/docbook-ng" >
<qandaentry>
<question>
<para>To be, or not to be?</para>
</question>
<answer>
<para>That is the question.</para>
</answer>
</qandaentry>
</qandaset>
```



# Schema

The solution here is to explicitly add the Docbook `<qandaset>` element to the TEI `lists` class. We load the default TEI core tagsets, and all of Docbook. It is also necessary to override the Docbook default starting elements, as the master scheme here is TEI.

```
default namespace = "http://www.tei-c.org/ns/1.0"
include "tei.rnc" {
include "general.rnc" {
  tei.lists |= db.qandaset
}
include "docbook.rnc" {
  start |= TEI
}
```



# TEI adding to Docbook

A Docbook document uses the TEI play markup to embed portions of Hamlet:

```
<article
  xmlns:tei="http://www.tei-c.org/ns/1.0"
  xmlns="http://docbook.org/docbook-ng"
  version="bourbon">
  ...
  <section><info><title>And now for something completely
    different</title></info>
    <tei:stage type="setting"> Elsinore. A platform before
      the castle. FRANCISCO at his post.</tei:stage>

    <tei:stage type="entrance"> Enter to him BERNARDO. </tei:stage>

    <tei:sp who="Barnardo"><tei:speaker>Bernardo</tei:speaker>
      <tei:l part="Y">Who's there? </tei:l>
    </tei:sp>
    ...
  </section>
</article>
```



# Schema

Here the master scheme is Docbook; there is no need to override the default start pattern for TEI, as we load a view of the TEI (`teilib.rnc`) which is intended for interchange, and has no `<start>` pattern at all. The TEI class `common` is added to the Docbook `technical.blocks` class:

```
include "docbook.rnc"  
include "teilib.rnc"  
technical.blocks |= tei.common
```



# TEI class replacing Docbook class

A Docbook document in which the normal metadata header is replaced by a TEI `<teiHeader>`:

```
<article
  xmlns="http://docbook.org/docbook-ng"
  version="bourbon">
  <teiHeader xmlns="http://www.tei-c.org/ns/1.0">
    <fileDesc>
      <titleStmt>
        <title>Testing TEI headers inside Docbook</title>
        <author>Sebastian Rahtz and Norman Walsh</author>
      </titleStmt>
      <publicationStmt>
        <p>published by OUCS</Screen>
      </publicationStmt>
      <sourceDesc>
        <p>written from scratch</p>
      </sourceDesc>
    </fileDesc>
  </teiHeader>
  <section>
    <info><title>Introduction</title></info>
    . . .
  </article>
```





# Schema

Again, we simply load Docbook and TEI interchange core; but this time we do not extend an existing class, but replace it:

```
include "docbook.rnc" {  
  article.info = tei.teiHeader  
}  
include "teilib.rnc"
```



# Docbook class replacing TEI class

A TEI document in which the normal metadata header is replaced by a Docbook `<info>`, and MathML is used in the body of the TEI document:

```
<TEI xmlns="http://www.tei-c.org/ns/1.0">
<info xmlns="http://docbook.org/docbook-ng">
  <title>Simulation of Energy Loss Stragglng</title>
  <author><personname>
    <surname>Physicist</surname>
    <firstname>Maria</firstname></personname></author>
  <pubdate>2004-03-11</pubdate>
</info>
<text>
<body>
<p>The description of ionisation fluctuations is
characterised by the significance parameter
<formula notation="MathML">
  <math xmlns="http://www.w3.org/1998/Math/MathML">
    <mi> $\kappa$ </mi>
  </math>
</formula>,
....
  </body>
</text>
</TEI>
```

A



# Schema (1)

This is a much more complex example. First, we declare some namespaces, and load MathML 2 and Xlink modules (the latter is used internally by MathML):

```
namespace a = "http://relaxng.org/ns/compatibility/annotations/1.0"

namespace xlink = "http://www.w3.org/1999/xlink"
default namespace = "http://www.tei-c.org/ns/1.0"

include "Schema/xlink.rnc" {}
include "Schema/mathml2-main.rnc"
```

Next we load Docbook, and override its start pattern:

```
include "docbook.rnc" {
  start |= TEI
}
```



## Schema (2)

For the TEI, overload the existing class for headers with that from Docbook, and in the module which covers figures and formula, overload the datatype pattern for formula content with the root MathML pattern:

```
include "teilib.rnc" {
  tei.teiHeader = article.info
}
include "linking.rnc"
include "figures.rnc" {
  datatype.Formula = mathml.math
}
```



# TEI and Docbook interleaved

A TEI document which allows for Docbook elements for describing GUIs in inline contexts, and for TEI structured inline elements to then 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 on it:
<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

This final example is deceptively simple, but very powerful. We tell the TEI that it can use the Docbook `gui.inlines` class in with its normal phrase-level elements, and tell Docbook that it can use the TEI class `data` in *its* phrase-level class.

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



# Validation of testing

We have tested our five examples with a variety of the available software, as follows:

**Jing** <http://www.thaiopensource.com/relaxng/jing.html>

**RNV** <http://davidashen.net/rnv.html>

**Emacs nxml mode** <http://www.thaiopensource.com/download/>

**xmlint** <http://xmlsoft.org/>

**<oXygen/>** <http://www.oxygenxml.com>

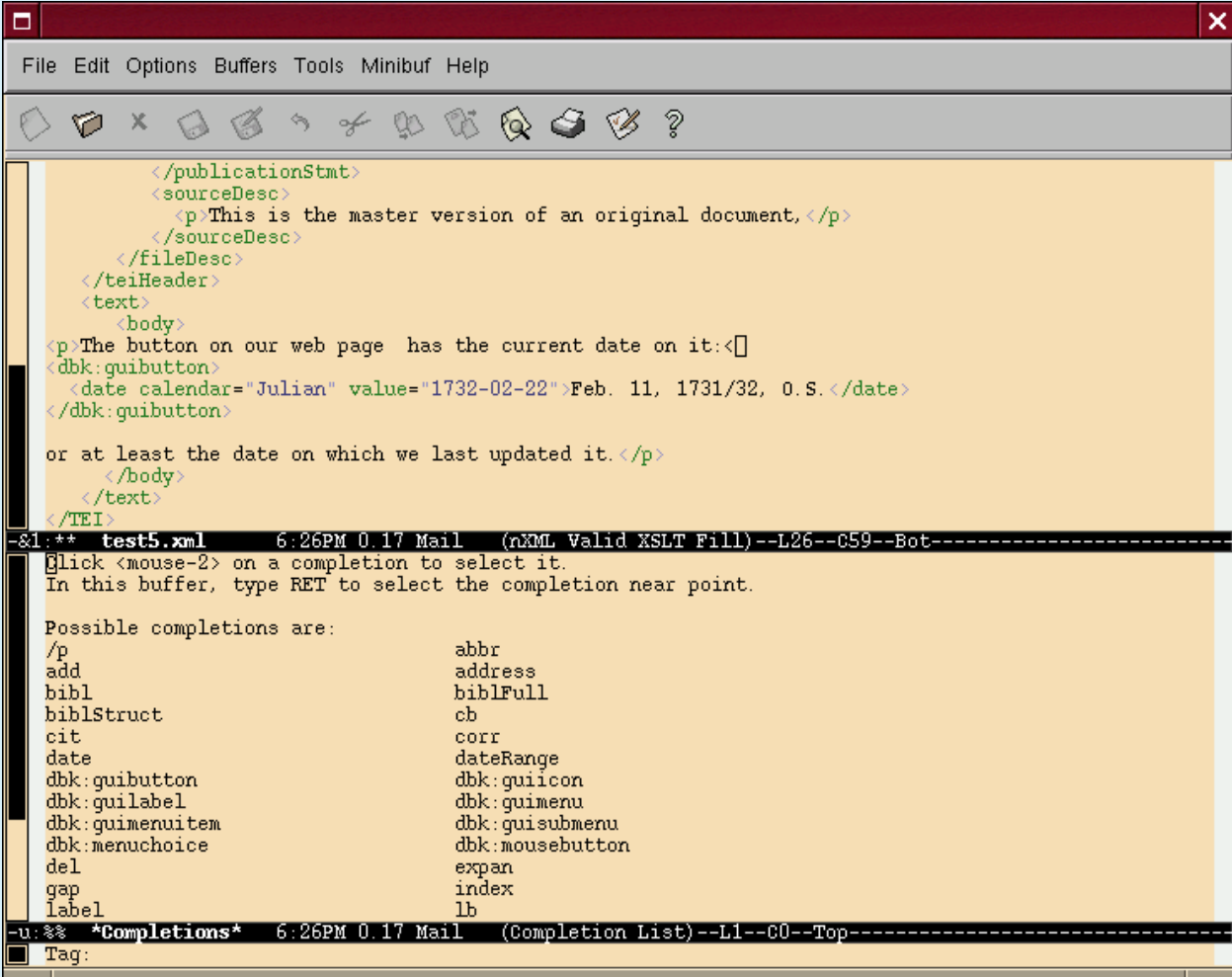
**MSV** <http://www.sun.com/software/xml/developers/multischema/>

No unexplained errors were encountered during this informal testing.



# Useability in editors (1)

## Test 5 in Emacs:



The screenshot shows the Emacs editor interface. The main window displays XML code with a completion list overlaid. The completion list is titled "Possible completions are:" and lists various XML tags and attributes. The status bar at the bottom shows the current buffer is "test5.xml" and the completion list is active.

```
</publicationStmt>
<sourceDesc>
  <p>This is the master version of an original document,</p>
</sourceDesc>
</fileDesc>
</teiHeader>
<text>
  <body>
<p>The button on our web page has the current date on it:<input type="text" value="" />
<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>
```

Possible completions are:

/p	abbr
add	address
bibl	biblFull
biblStruct	cb
cit	corr
date	dateRange
dbk:guibutton	dbk:guiicon
dbk:guilabel	dbk:guimenu
dbk:guimenuitem	dbk:guisubmenu
dbk:menuchoice	dbk:mousebutton
del	expan
gap	index
label	lb

u:⌘ \*Completions\* 6:26PM 0.17 Mail (Completion List)--L1--C0--Top

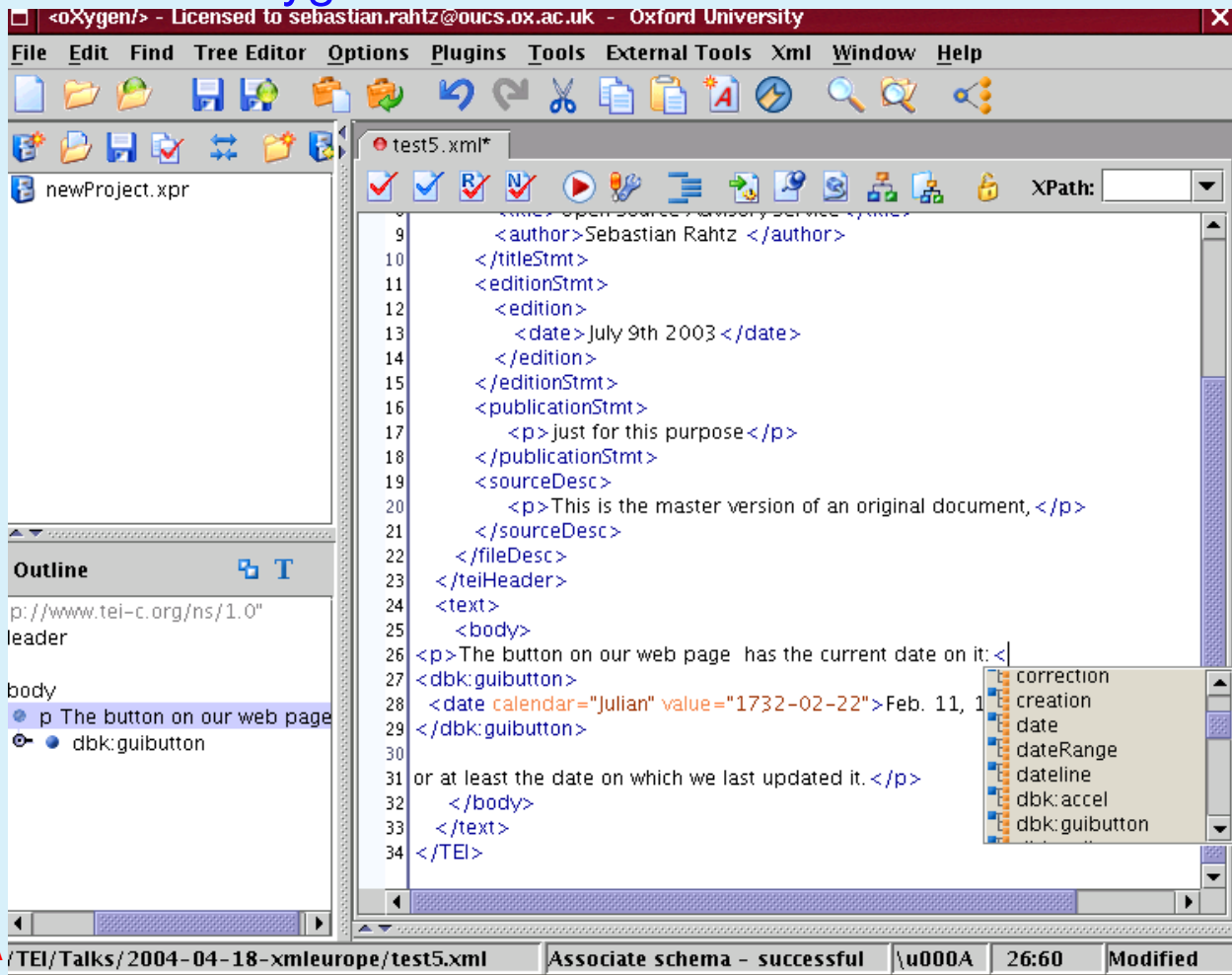
A





# Useability in editors (2)

## Test 5 in oXygen:



The screenshot shows the oXygen XML editor window. The title bar reads "<oxygen/> - Licensed to sebastian.rahtz@oucs.ox.ac.uk - Oxford University". The menu bar includes File, Edit, Find, Tree Editor, Options, Plugins, Tools, External Tools, Xml, Window, and Help. The toolbar contains various icons for file operations and editing. The main editor area displays the XML code for "test5.xml":

```
9 <author>Sebastian Rahtz </author>
10 </titleStmt>
11 <editionStmt>
12 <edition>
13 <date>July 9th 2003 </date>
14 </edition>
15 </editionStmt>
16 <publicationStmt>
17 <p>just for this purpose</p>
18 </publicationStmt>
19 <sourceDesc>
20 <p>This is the master version of an original document,</p>
21 </sourceDesc>
22 </fileDesc>
23 </teiHeader>
24 <text>
25 <body>
26 <p>The button on our web page has the current date on it:<
27 <dbk:guibutton>
28 <date calendar="Julian" value="1732-02-22">Feb. 11, 1
29 </dbk:guibutton>
30
31 or at least the date on which we last updated it.</p>
32 </body>
33 </text>
34 </TEI>
```

The sidebar on the left shows an "Outline" view with the following structure:

- p: //www.tei-c.org/ns/1.0"
- leader
- body
  - p The button on our web page
    - dbk:guibutton

A

We have not yet attempted to use W3C Schema

# Linking to the wide world

Technically, it is not hard to combine vocabularies in different namespaces. Now comes the problem:

- ☞ how are the meanings of technical terms in different languages related?
- ☞ how do new meanings emerge?
- ☞ how are meanings transferred between different languages?

These are not new problems in the language research community, and it is reasonable therefore to look to that community for insight into their solution.



# Technical terminology standards

If we map TEI and Docbook terminologies to a standard conceptual model, 'round-tripping' between documents in either scheme would be feasible. We look to

- ISO/TC 37/SC4's definition of a Linguistic Annotation Framework, comprising a number of related standards, one of which, ISO DIS 12620/1, relates to the definition of Data Category Registries for language resources
- The CIDOC Conceptual Reference Model (CRM) (<http://cidoc.ics.forth.gr/>) which has been developed as a way of defining a common and extensible semantic framework within which complex cultural heritage information can be expressed and interchanged.



# Conclusions

Has this work gone beyond onanism? Hopefully.

- We have two vocabularies which cover different specialities
- Both languages classify elements into semantic classes
- We can model the semantic classes in Relax NG
- We can express interleaving of the classes in Relax NG
- It is practical to author in a combination of the vocabularies and maintain semantic structure
- We have the possibility of formalizing the semantic joins

