Introduction to Customising the TEI with Roma

TEI@Oxford

July 2009
Some terminology

• The TEI encoding scheme consists of a number of *modules*

• Each module contains a number of *element specifications* (marked up in TEI using the `<elementSpec>` element)

• Each element specification contains:
  • a canonical name (`<gi>`) for the element, and optionally other names in other languages
  • a canonical description (also possibly translated) of its function
  • a declaration of the *classes* to which it belongs
  • a definition for each of its *attributes*
  • a definition of its *content model*
  • usage examples and notes

• a TEI *schema* specification (`<schemaSpec>`) is made by selecting modules and (optionally) modifying their contents

• a TEI document containing a schema specification is called an *ODD* (One Document Does it all)
What is a module?

- A convenient way of grouping together a number of element declarations
- These are usually on a related topic or specific application
- Most chapters of P5 focus on elements drawn from a single module, which that chapter then defines
- A TEI Schema is created by selecting modules and adding or removing elements from them as needed
### Which modules exist?

<table>
<thead>
<tr>
<th>Module name</th>
<th>Chapter</th>
</tr>
</thead>
<tbody>
<tr>
<td>analysis</td>
<td>Simple Analytic Mechanisms</td>
</tr>
<tr>
<td>certainty</td>
<td>Certainty and Responsibility</td>
</tr>
<tr>
<td>core</td>
<td>Elements Available in All TEI Documents</td>
</tr>
<tr>
<td>corpus</td>
<td>Language Corpora</td>
</tr>
<tr>
<td>dictionaries</td>
<td>Dictionaries</td>
</tr>
<tr>
<td>drama</td>
<td>Performance Texts</td>
</tr>
<tr>
<td>figures</td>
<td>Tables, Formulae, and Graphics</td>
</tr>
<tr>
<td>gaiji</td>
<td>Representation of Non-standard Characters and Glyphs</td>
</tr>
<tr>
<td>header</td>
<td>The TEI Header</td>
</tr>
<tr>
<td>iso-fs</td>
<td>Feature Structures</td>
</tr>
<tr>
<td>linking</td>
<td>Linking, Segmentation, and Alignment</td>
</tr>
<tr>
<td>msdescription</td>
<td>Manuscript Description</td>
</tr>
<tr>
<td>namesdates</td>
<td>Names, Dates, People, and Places</td>
</tr>
<tr>
<td>nets</td>
<td>Graphs, Networks, and Trees</td>
</tr>
<tr>
<td>spoken</td>
<td>Transcriptions of Speech</td>
</tr>
<tr>
<td>tagdocs</td>
<td>Documentation Elements</td>
</tr>
<tr>
<td>tei</td>
<td>The TEI Infrastructure</td>
</tr>
<tr>
<td>textcrit</td>
<td>Critical Apparatus</td>
</tr>
<tr>
<td>textstructure</td>
<td>Default Text Structure</td>
</tr>
<tr>
<td>transcr</td>
<td>Representation of Primary Sources</td>
</tr>
<tr>
<td>verse</td>
<td>Verse</td>
</tr>
</tbody>
</table>
How do you choose?

- Just choose everything (not really a good idea)
- The TEI provides a small set of predefined combinations (TEI Lite, TEI Bare...)
- Or you could roll your own (but then you need to know what you're choosing)

**Roma** a command line script, with a web front end, designed to make this process much easier

http://www.tei-c.org/Roma/
These pages will help you design your own TEI validator, as a DTD, RELAXNG or W3C Schema.

- **Build schema** (Create a new customisation by adding elements and modules to the smallest recommended schema)
- **Reduce schema** (Create a new customization by removing elements and modules from the largest possible schema)

**Create a new or upload existing customization**

- **Create customization from template**
  - TEI Absolutely Bare
  - TEI Lite
  - TEI for Linguistic Corpora
  - TEI for Manuscript Description
  - TEI with Drama
  - TEI for Speech Representation
  - TEI for authoring ODD
  - TEI with SVG
  - TEI with MathML
  - TEI with XInclude (experimental)
  - TEI for Dictionaries (experimental)

**Open existing customization**

**Submit**
Roma: Customize

Roma: generating validators for the TEI

Set your parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>My TEI Extension</td>
</tr>
<tr>
<td>Filename</td>
<td>myTei</td>
</tr>
<tr>
<td>Prefix for TEI pattern names in schema</td>
<td></td>
</tr>
<tr>
<td>Language</td>
<td>English, Deutsch, Italiano, Español, Français, Portugues, Russian, Svenska, 中文</td>
</tr>
<tr>
<td>Author name</td>
<td>generated by Roma 3.0</td>
</tr>
<tr>
<td>Description</td>
<td>My TEI Customization starts with modules tei, core, header, and textstructure</td>
</tr>
</tbody>
</table>

Submit

Search TEI database
Roma: Schema

Roma: generating validators for the TEI

Time to give you a schema

Creating a schema

Which format do you prefer?

- Relax NG schema (compact syntax)
- Relax NG schema (XML syntax)
- W3C schema
- DTD

Submit

Search TEI database

Roma was written by Arno Mittelbach and is maintained by Sebastian Rahtz. Sanity check written by Iohan Bernevig. Please direct queries to the TEI @ Oxford project. This is Roma version 3.0, last updated 2007-10-21.
Roma: generating validators for the TEI

Getting some nice documentation

Which output would you prefer?

- html
- pdf
- TEI Lite
- Tei

Search TEI database

Roma was written by Arno Mittelbach and is maintained by Sebastian Rahtz. Sanity check written by Ioan Bernevig. Please direct queries to the TEI@Oxford project. This is Roma version 3.0, last updated 2007-10-21.
What did we just do?
We processed a pre-existing ODD file which contained (as well as some discursive prose) the following schema specification:

```xml
<schemaSpec ident="tei_bare" start="TEI">
   <moduleRef key="core"/>
   <moduleRef key="tei"/>
   <moduleRef key="header"/>
   <moduleRef key="textstructure"/>
   <elementSpec ident="abbr" mode="delete" module="core"/>
   <elementSpec ident="add" mode="delete" module="core"/>
   <!-- ... -->
   <elementSpec ident="trailer" mode="delete" module="textstructure"/>
   <elementSpec ident="title" mode="change" module="core">
      <attList>
         <attDef ident="level" mode="delete"/>
      </attList>
   </elementSpec>
   <!-- ... -->
</schemaSpec>
```

We selected four modules, deleted loads of elements, and also deleted an attribute
Roma provides an interface to the detail

- The [Modules] tab shows the modules available
- Selecting a module from it shows the elements within that module, and gives you the choice to
  - include all of them (and then remove some)
  - exclude all of them (and then put back the ones you want)
- You can also change an element's attribute list, and the values they permit
### List of TEI Modules

<table>
<thead>
<tr>
<th>Module name</th>
<th>A short description</th>
</tr>
</thead>
<tbody>
<tr>
<td>add analysis</td>
<td>Simple analytic mechanisms</td>
</tr>
<tr>
<td>add certainty</td>
<td>Certainty and uncertainty</td>
</tr>
<tr>
<td>add core</td>
<td>Elements common to all TEI documents</td>
</tr>
<tr>
<td>add corpus</td>
<td>Header extensions for corpus texts</td>
</tr>
<tr>
<td>add declarfs</td>
<td>Feature system declarations</td>
</tr>
<tr>
<td>add dictionaries</td>
<td></td>
</tr>
<tr>
<td>add drama</td>
<td>Performance texts</td>
</tr>
<tr>
<td>add figures</td>
<td>Tables, formulae, and figures</td>
</tr>
<tr>
<td>add gaiii</td>
<td>Character and glyph documentation</td>
</tr>
<tr>
<td>add header</td>
<td>The TEI Header</td>
</tr>
<tr>
<td>add iso-fs</td>
<td>Feature structures</td>
</tr>
<tr>
<td>add linking</td>
<td>Linking, segmentation and alignment</td>
</tr>
<tr>
<td>add msdescription</td>
<td></td>
</tr>
<tr>
<td>add namesdates</td>
<td>Names and dates</td>
</tr>
<tr>
<td>add nets</td>
<td>Graphs, networks and trees</td>
</tr>
<tr>
<td>add spoken</td>
<td>Transcribed Speech</td>
</tr>
<tr>
<td>add tagdocs</td>
<td>Documentation of TEI modules</td>
</tr>
<tr>
<td>add textcrit</td>
<td>Text criticism</td>
</tr>
<tr>
<td>add textstructure</td>
<td>Default text structure</td>
</tr>
</tbody>
</table>

### List of selected Modules

- remove **core**
- remove **tei**
- remove **header**
- remove **textstructure**
# Roma: Change Module

**TEI**

## Roma: generating validators for the TEI

### Change module

<table>
<thead>
<tr>
<th>List of elements in module: core</th>
<th>Tag name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>abbr</td>
<td>abbr</td>
<td>contains an abbreviation of any sort.</td>
</tr>
<tr>
<td>addr</td>
<td>add</td>
<td>contains letters, words, or phrases inserted in the text by an author, scribe, annotator, or corrector.</td>
</tr>
<tr>
<td>addrLine</td>
<td>addrLine</td>
<td>contains one line of a postal address.</td>
</tr>
<tr>
<td>address</td>
<td>address</td>
<td>contains a postal address, for example of a publisher, an organization, or an individual.</td>
</tr>
<tr>
<td>altident</td>
<td>altident</td>
<td>supplies the recommended XML name for an element, class, attribute, etc. in some language.</td>
</tr>
<tr>
<td>analytic</td>
<td>analytic</td>
<td>contains bibliographic elements describing an item (e.g. an article or poem) published within a monograph or journal and not as an independent publication.</td>
</tr>
<tr>
<td>author</td>
<td>author</td>
<td>in a bibliographic reference, contains the name of the author(s), personal or corporate, of a work; the primary statement of responsibility for any bibliographic item.</td>
</tr>
<tr>
<td>bibl</td>
<td>bibl</td>
<td>contains a loosely-structured bibliographic citation of which the sub-components may or may not be explicitly tagged.</td>
</tr>
<tr>
<td>biblScope</td>
<td>biblScope</td>
<td>defines the scope of a bibliographic reference, for example, the scope of a journal or monograph.</td>
</tr>
</tbody>
</table>
What does the Punch Project need?

A simple selection of elements, but also

- we want to allow only certain values for `@type` on `<div>`
- we want a new element to wrap the combination of a `<cit>` and a comment on it: we will call it a `<citCom>` (you might like to think of a better name)

Other constraints are possible -- we might want to insist that a `<div type="cartoon">` contains a graphic, for example.
The ODD advantage

We can express these constraints in our ODD, and then generate a formal schema to enforce them using whichever schema language we like

- TEI schemas can be generated in
  - ISO RELAX NG language
  - W3C Schema Language
  - XML DTD language

- ODD itself defines an element's content models using a subset of RELAX NG syntax

- Datatypes are defined in terms of W3C datatypes

- Some facilities (e.g. alternation, namespaces) cannot be expressed in DTDs -- RELAX NG schema is recommended

- Additional constraints can be expressed in Schematron
### Roma: selecting attributes

#### Roma: generating validators for the TEI

#### Added Attributes

<table>
<thead>
<tr>
<th>Change attribute</th>
<th>Include</th>
<th>Exclude</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>org</td>
<td></td>
<td></td>
<td>org</td>
<td>specifies how the content of the division is organized.</td>
</tr>
<tr>
<td>sample</td>
<td></td>
<td></td>
<td>sample</td>
<td>indicates whether this division is a sample of the original source and if so, from which part.</td>
</tr>
<tr>
<td>part</td>
<td></td>
<td></td>
<td>part</td>
<td>specifies whether or not the division is fragmented by some other structural element, for example a speech which is divided between two or more verse stanzas.</td>
</tr>
<tr>
<td>type</td>
<td></td>
<td></td>
<td>type</td>
<td>characterizes the element in some sense, using any convenient classification scheme or typology.</td>
</tr>
<tr>
<td>subtype</td>
<td></td>
<td></td>
<td>subtype</td>
<td>provides a sub-categorization of the element, if needed</td>
</tr>
<tr>
<td>decls</td>
<td></td>
<td></td>
<td>decls</td>
<td>identifies one or more declarable elements within the header, which are understood to apply to the element bearing this attribute and its content.</td>
</tr>
<tr>
<td>xml:id</td>
<td></td>
<td></td>
<td>xml:id</td>
<td>provides a unique identifier for the element bearing the attribute.</td>
</tr>
<tr>
<td>n</td>
<td></td>
<td></td>
<td>n</td>
<td>gives a number (or other label) for an element, which is not necessarily unique within the document.</td>
</tr>
<tr>
<td>xml:lang</td>
<td></td>
<td></td>
<td>xml:lang</td>
<td>indicates the language of the element content using a tag generated according to BCP 47</td>
</tr>
<tr>
<td>rend</td>
<td></td>
<td></td>
<td>rend</td>
<td>indicates how the element in question was rendered or presented in the source text.</td>
</tr>
<tr>
<td>rendition</td>
<td></td>
<td></td>
<td>rendition</td>
<td>points to a description of the rendering or presentation used for this element in the source text.</td>
</tr>
<tr>
<td>xml:base</td>
<td></td>
<td></td>
<td>xml:base</td>
<td>provides a base URI reference with which applications can resolve relative URI references into absolute URI references.</td>
</tr>
</tbody>
</table>
Roma: constraining attribute values

Roma: generating validators for the TEI

Add some attributes

Add a new attribute

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class name</td>
<td></td>
</tr>
<tr>
<td>Is it optional?</td>
<td>yes</td>
</tr>
<tr>
<td>Contents</td>
<td>Text</td>
</tr>
<tr>
<td>Default value</td>
<td></td>
</tr>
<tr>
<td>Closed list?</td>
<td>yes</td>
</tr>
<tr>
<td>List of values</td>
<td>cartoon, snippets, snippet, verse, story, review, report,</td>
</tr>
<tr>
<td>Description</td>
<td>characterizes the element in some sense, using any convenient classification scheme or typology.</td>
</tr>
</tbody>
</table>

Save
What did we just do?

Our ODD now includes something like this:

```xml
<elementSpec ident="div" module="textstructure" mode="change">
  <attList>
    <attDef ident="type" mode="change" usage="req">
      <valList type="closed" mode="replace">
        <valItem ident="cartoon"/>
        <valItem ident="snippet"/>
        <valItem ident="verse"/>
      </valList>
    </attDef>
  </attList>
</elementSpec>

Note that we can also add documentation to the ODD:

```xml
<valItem ident="cartoon">
  <gloss>contains a humorous picture, usually with dialogue underneath</gloss>
</valItem>
```
When defining a new element, we need to consider
  • its name and description
  • what attributes it can carry
  • what it can contain
  • where it can appear in a document

The TEI class system helps us answer all these questions (except the first).
• The TEI distinguishes over 500 elements,
• Having these organised into classes aids comprehension, modularity, and modification.
• *Attribute class*: the members share common attributes
• *Model class*: they can appear in the same locations (and are often semantically related)
• Classes may contain other classes
• An element can be a member of any number of classes, irrespective of the module it belongs to.
Attribute Classes

- Attribute classes are given (usually adjectival) names beginning with `att.`; e.g. `att.naming`, `att.typed`
- all members of `att.naming` inherit from it attributes `@key` and `@ref`; all members of `att.typed` inherit from it `@type` and `@subtype`
- If we want an element to carry the `@type` attribute, therefore, we add the element to the `att.typed` class, rather than define those attributes explicitly.
A very important attribute class: att.global

All elements are a member of att.global; this class provides, among others:

- @xml:id  a unique identifier
- @xml:lang  the language of the element content
  - @n  a number or name for an element
- @rend  how the element in question was rendered or presented in the source text.

All new elements are members of this class by default.
Model Classes

- Model classes contain groups of elements which are allowed in the same place. e.g. if you are adding an element which is wanted wherever the `<bibl>` is allowed, add it to the `model.biblLike` class

- Model classes are usually named with a **Like** or **Part** suffix:
  - members of `model.pLike` are all things that ‘behave like’ paragraphs, and are permitted in the same places as paragraphs
  - members of `model.pPart` are all things which can appear within paragraphs. This class is subdivided into
    - `model.pPart.edit` elements for simple editorial intervention such as `<corr>`, `<del>` etc.
    - `model.pPart.data` ‘data-like’ elements such as `<name>`, `<num>`, `<date>` etc.
    - `model.pPart.msdesc` extra elements for manuscript description such as `<seal>` or `<origPlace>`
Basic Model Class Structure

Simplifying wildly, one may say that the TEI recognises three kinds of element:

- **divisions**  high level major divisions of texts
- **chunks**  elements such as paragraphs appearing within texts or divisions, but not other chunks
- **phrase-level elements**  elements such as highlighted phrases which can occur only within chunks

There are ‘base model classes’ corresponding with each of these, and also with the following groupings: three:

- **inter-level elements**  elements such as lists which can appear either in or between chunks
- **components**  elements which can appear directly within texts or text divisions

And yes, there is a class `model.global` for elements that can appear anywhere -- at any hierarchic level.
Defining our new element `<citCom>`

**What other elements is it like?** It's like a paragraph or quotation. It's not a phrase level element, because it must contain more than just unstructured text.

**What other elements can contain it?** It can only appear within a division, like a paragraph.

**What can it contain?** It must contain a citation (i.e. a quote optionally associated with a bibliographic reference) or something like that, followed by at least one paragraph of commentary.

**Conclusions:**

- we make it a member of `model.divPart`
- we will have to define a special content model for it
Roma: Defining a new element

Roma: generating validators for the TEI

Add Element

New Customize Language Modules Add Elements Change Classes Schema Documentation Save Customization

go back to list

Defining a new element:

Name: citCom
Namespace: http://www.example.org/ns/nonTEI
Description: contains a citation followed by some commentary on it

Model classes:
- model.addPart
- model.applicationLike
- model.bibPart
- model.catDescPart
- model.common
- model.dimLike
- model.div2Like
- model.div4Like
- model.div6Like
- model.divBottom
- model.divGenLike
- model.divPart
- model.addressLike
- model.bibLike
- model.castItemPart
- model.choicePart
- model.dateLike
- model.div1Like
- model.div3Like
- model.div5Like
- model.div7Like
Defining a content model

• A typical TEI element defines its content by referencing *classes* of element which it can contain, rather than using specific elements.
• Content models are defined using the RELAXNG vocabulary
• Here are some very common predefined content models:
  - `macro.paraContent` content of paragraphs and similar elements
  - `macro.limitedContent` content of prose elements that are not used for transcription of extant materials
  - `macro.phraseSeq` a sequence of character data and phrase-level elements
  - `macro.phraseSeq.limited` a sequence of character data and those phrase-level elements that are not typically used for transcribing extant documents
  - `macro.specialPara` the content model of elements which either contain a series of component-level elements or else contain a series of phrase-level and inter-level elements
### Roma: Defining a new element 2

<table>
<thead>
<tr>
<th>Attribute classes</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>att.ascribed</td>
<td>att.canonical</td>
</tr>
<tr>
<td>att.coordinated</td>
<td>att.damaged</td>
</tr>
<tr>
<td>att.datablable</td>
<td>att.databindable.w3c</td>
</tr>
<tr>
<td>att.datablable.iso</td>
<td>att.declarable</td>
</tr>
<tr>
<td>att.declaring</td>
<td>att.dimensions</td>
</tr>
<tr>
<td>att.divLike</td>
<td>att.duration</td>
</tr>
<tr>
<td>att.duration.iso</td>
<td>att.duration.w3c</td>
</tr>
<tr>
<td>att.editLike</td>
<td>att.enjamb</td>
</tr>
<tr>
<td>att.entryLike</td>
<td>att.handFeatures</td>
</tr>
<tr>
<td>att.identified</td>
<td>att.internetMedia</td>
</tr>
<tr>
<td>att.interpLike</td>
<td>att.lexicographic</td>
</tr>
<tr>
<td>att.measurement</td>
<td>att.metrical</td>
</tr>
<tr>
<td>att.msExcerpt</td>
<td>att.naming</td>
</tr>
<tr>
<td>att.personal</td>
<td>att.placement</td>
</tr>
<tr>
<td>att.pointing</td>
<td>att.pointing.group</td>
</tr>
<tr>
<td>att.ptrLike.form</td>
<td>att.ranging</td>
</tr>
<tr>
<td>att.rdgPart</td>
<td>att.segLike</td>
</tr>
<tr>
<td>att.sourced</td>
<td>att.spanning</td>
</tr>
<tr>
<td>att.tableDecoration</td>
<td>att.textCritical</td>
</tr>
<tr>
<td>att.timed</td>
<td>att.transcriptional</td>
</tr>
<tr>
<td>att.translatable</td>
<td>att.xmlspace</td>
</tr>
<tr>
<td>att.typed</td>
<td></td>
</tr>
</tbody>
</table>

**Contents**

User content

---
What did we just do?

We added a new element specification to our ODD, like this:

```xml
<elementSpec
    ident="citCom"
    ns="http://www.example.org/ns/nonTEI"
    mode="add">
    <desc> contains a citation followed by some commentary on it. </desc>
    <classes>
        <memberOf key="model.divLike"/>
        <memberOf key="att.typed"/>
    </classes>
    <content>
        <rng:ref name="cit"/>
        <rng:oneOrMore>
            <rng:ref name="model.pLike"/>
        </rng:oneOrMore>
    </content>
</elementSpec>
```

Note that this new element is not in the TEI namespace. It belongs to the IPP project only!
Other kinds of constraints

• You can also constrain the content of an element or the value of an attribute to be of a particular *datatype* (for example, to insist that the element `<date>` contains only a date)

• This can be done by using one of a set of predefined *macros* to define the content. Examples include
  
  data.word  a single word or token
  data.name  an XML Name
  data.enumerated  a single XML name taken from a documented list
  data.temporal.w3c  a W3C date
  data.truthValue  a truth value (true/false)
  data.language  a human language
  data.sex  human or animal sex

• Or you can define a more complex constraint, e.g. using Schematron
Schematron constraints

• (New at P5 release 1.4)
• An element specification can also contain a
  `<constraintSpec>` element which contains rules about its
  content expressed as ISO Schematron *constraints*

```xml
<elementSpec ident="div" module="teiststructure" mode="change"
  xmlns:s="http://purl.oclc.org/dsdl/schematron">
  <constraintSpec ident="cartoon" scheme="isoschematron">
    <constraint>
      <s:assert test="@type='cartoon' and .//tei:graphic">
        a cartoon must include a graphic
      </s:assert>
    </constraint>
  </constraintSpec>
</elementSpec>
```

However...

• You can only add such rules by editing your ODD file: Roma
  doesn't know about them.
• Not all schema languages can implement these constraints