Documenting TEI Customisations

James Cummings

April 2009

TEI Infrastructure

- The TEI encoding scheme consists of a number of modules
- These declare XML elements and their attributes
- An element's declaration assigns it to one (or more) model classes
- Another part declares its possible content and attributes with reference to these classes
- This indirection allows strength and flexibility
- It makes it easy to add/exclude new elements by referencing existing classes

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 focus on elements drawn from a single module, which that chapter then defines
- A TEI Schema is created by selecting modules and add/removing elements from them as needed

Modules

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

Support for many schema languages

The TEI uses a subset of itself called TEI ODD as a base to generate both project documentation and schemas:

- TEI schemas can be generated for
  - ISO RELAX NG language
  - W3C Schema Language
  - XML DTD language
- Internally, content models are defined using 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

Coping with partially-baked ideas

In a TEI ODD, you can ...

- constrain the domain of a value list
- enforce schematron rules about e.g. codependency
- provide new elements in your own namespace
- remove (non-mandatory) child elements

From the single TEI ODD you can then generate the required schemas, as well as your project documentation.

New elements

A schema is a grammar. How can you add new terminals to an existing syntax?

- Content models are expressed indirectly, by reference to element classes rather than elements
- Hence adding a new element is simply a matter of saying which class(es) it belongs to

The TEI schema is also enriched with semantics. How can you explain what a new element means?

- Class membership also conveys some semantics
- ODD includes detailed documentation

Do not re-invent the wheel

- TEI P5 has extensive I18N features for translation of ...
  - schema objects
  - schema documentation
- TEI is hospitable to other namespaces:
  - You can use SVG for graphics, MathML for math, or any other markup if you like
- TEI ODD also includes an `<equiv>` element for mapping to external ontologies

For example

Embedding SVG within TEI:

```
<figure>
 <svg xmlns="http://www.w3.org/2000/svg"
  width="6cm" height="5cm" viewBox="6 3 6 5">
  <ellipse xmlns="http://www.w3.org/2000/svg"
   style="fill:
#ffffff" cx="9.75" cy="6.35" rx="2.75" ry="2.35"/>
 </svg>
</figure>
```

A user-defined attribute:

```
<div
   xmlns:my="http://www.example.org/ns/nonTEI">
 <p n="12" my:topic="rabbits">Flopsy, Mopsy, Cottontail,
and Peter...</p>
</div>
```

NVDL processors validate against multiple namespace schemas, so you can validate each part individually

The TEI Class System

- 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 often are structurally or semantically related)
- Classes may contain other classes
- Elements inherit the properties from any classes of which they are members

Attribute Classes

- Attribute classes are given (usually adjectival) names beginning with att.; e.g. members of the att.naming class get a *@key* attribute rather than have them define it individually
- If another element needs a *@key* attribute then the easiest way to provide it is to add it to the att.naming class
- Classes can be grouped together into a super classes

att.global

All elements are a member of att.global; this includes, 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.

att.global also contains att.global.linking so if the `linking` module is loaded it provides attributes:

- *@corresp* points to elements that correspond to the current element in some way
- *@copyOf* points to an element of which the current element is a copy
- *@next* points to the next element of a virtual aggregate of which the current element is part.
- *@prev* points to the previous element of a virtual aggregate of which the current element is part

Model Classes

- Model classes contain groups of elements 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:
  - model.divLike: structural class grouping elements for divisions
  - model.divPart: structural class grouping elements used inside divisions
  - model.nameLike: semantic class grouping name elements
  - model.persNamePart: semantic sub-class grouping elements that are part of a personal name

Macros

Macros are short-hand names for common patterns:

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

Datatype Macros

A special set of macros which provide common datatypes, mostly used for attributes:

- data.key a coded value
- data.word a single word or token
- data.name an XML Name
- data.enumerated a single XML name taken from a documented list
- data.duration.w3c a W3C duration
- data.temporal.w3c a W3C date
- data.truthValue a truth value (true/false)
- data.language a language
- data.sex human or animal sex

Basic Model Class Structure

The TEI class system makes a threefold division of elements:

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

The TEI identifies the following groupings from these 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

Classes for divisions

The TEI architecture defines five classes, all of which are populated by this module:

- model.divTop groups elements appearing at the beginning of a text division.
- model.divTopPart groups elements which can occur only at the beginning of a text division.
- model.divBottom groups elements appearing at the end of a text division.
- model.divBottomPart groups elements which can occur only at the end of a text division.
- model.divWrapper groups elements which can appear at either top or bottom of a textual division.

model.divWrapper members

<argument> A formal list or prose description of the topics addressed by a subdivision of a text.
<byline> contains the primary statement of responsibility given for a work on its title page or at the head or end of the work.
<dateline> contains a brief description of the place, date, time, etc. of production of a letter, newspaper story, or other work, prefixed or suffixed to it as a kind of heading or trailer.
<docAuthor> (document author) contains the name of the author of the document, as given on the title page (often but not always contained in a byline).
<docDate> (document date) contains the date of a document, as given (usually) on a title page.
<epigraph> contains a quotation, anonymous or attributed, appearing at the start of a section or chapter, or on a title page.

model.divTopPart members

<head> (heading) contains any type of heading, for example the title of a section, or the heading of a list, glossary, manuscript description, etc.
<salute> (salutation) contains a salutation or greeting prefixed to a foreword, dedicatory epistle, or other division of a text, or the salutation in the closing of a letter, preface, etc.
<opener> groups together dateline, byline, salutation, and similar phrases appearing as a preliminary group at the start of a division, especially of a letter.

model.divTop = model.divTopPart + model.divWrapper

model.divBottomPart members

<closer> groups together salutations, datelines, and similar phrases appearing as a final group at the end of a division, especially of a letter.
<signed> (signature) contains the closing salutation, etc., appended to a foreword, dedicatory epistle, or other division of a text.
<trailer> contains a closing title or footer appearing at the end of a division of a text.
<postscript> contains a postscript, e.g. to a letter.

model.divBottom = model.divBottomPart + model.divWrapper

Defining a TEI Schema

- A schema helps you know a document is valid in addition to being well-formed
- A TEI schema is a combination of TEI modules, optionally including customizations of the elements/attributes/classes that they contain
- This schema is defined in an application-independent manner with a TEI ODD (One Document Does it all) file which allows for:
  - creation of a schemas such as DTD, RELAX NG or W3C Schema
  - internationalized documentation which reflects your customization of the TEI
  - documentation of how your schema differs from tei_all that is suitable for long-term preservation

Important ODD concepts

The TEI's literary programming with ODD (One Document Does it all) provides:

- Schema specification
- User oriented documentation
- Modularity: all specifications pertaining to a coherent sub-domain of the TEI
- Classes: identifying shared behaviours or semantics
- Extensibility: a consequence of the above mechanisms

The TEI ODD in practice

The TEI Guidelines, its schema, 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

Possibilities of 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 each module
- Probably add local datatype constraints
- Possibly add new elements/attributes in other namespaces
- Possibly localize the names of elements

Real life TEI customization

We aim to support a range of interactions with the TEI:

- **Easy TEI** Simple access to the TEI through Roma
- **Subsetting the TEI** Making the full TEI even easier to use
- **Enlarging the application profile** Using modules
- **Modifying the TEI objects** First insights into extensibility
- **Behind the scene - ODD** Starting to use the actual specification language

Roma

The TEI knows you don't want to necessarily have to write TEI code in order to customize the TEI. So it has provided Roma, which is a command-line script, and corresponding web front-end to help you do this.

The people behind Roma are:

Arno Mittelbach Initial programming

Sebastian Rahtz Maintenance and frequent improvements

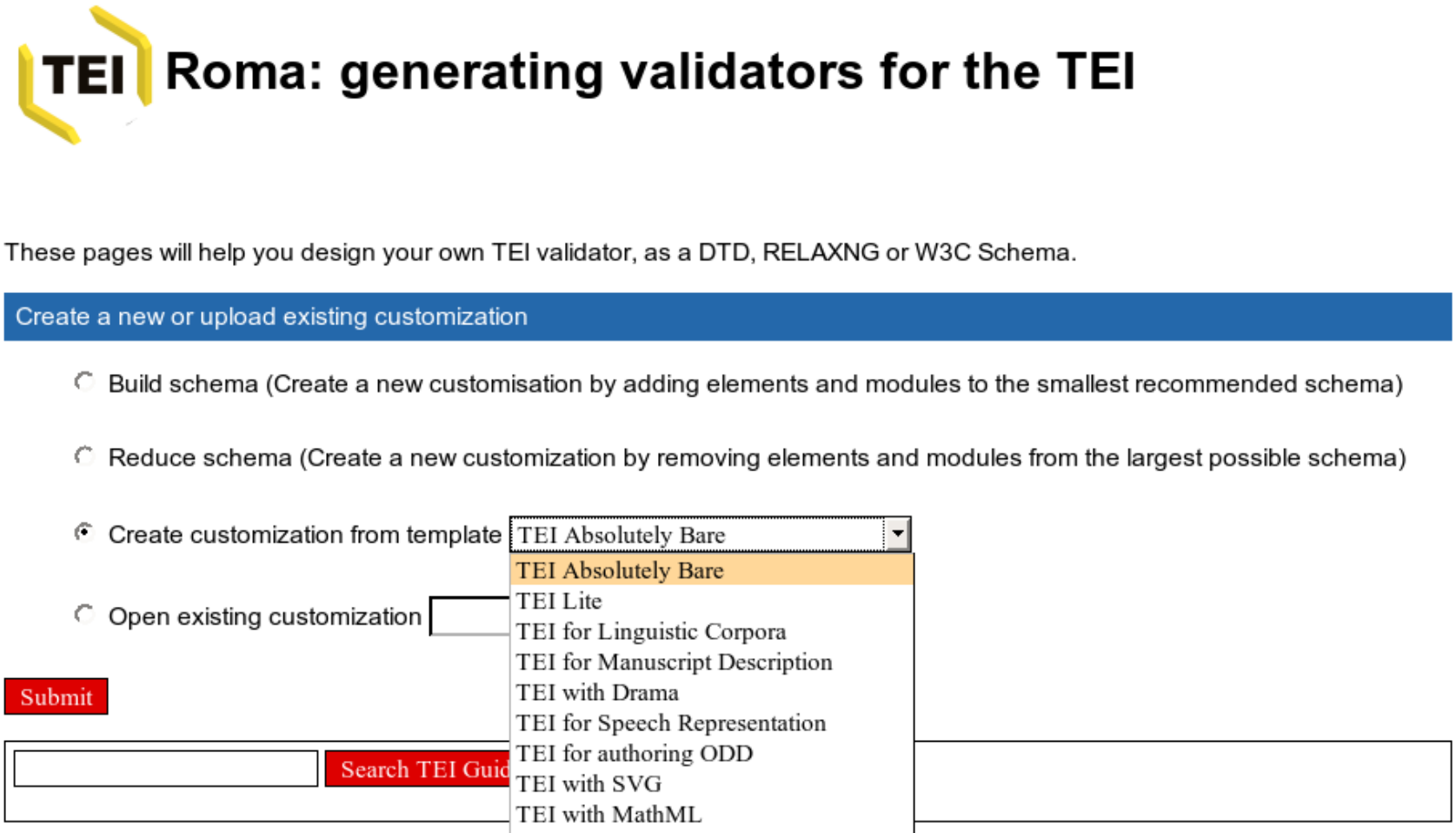
Ioan Bernevig A 'Sanity Checker' addition

How to use the TEI

Imagine that you have seen your colleague next door doing some encoding with the TEI and want to do the same thing:

- Go to Roma at `http://tei.oucs.ox.ac.uk/Roma/`
- Toy with the user profile [Customize]
- Generate a schema [Schema]
- Make a trial with the editor, creating a simple document
- Get back to Roma and make basic documentation

Roma: New


Roma: generating validators for the TEI
These pages will help you design your own TEI validator, as a DTD, RELAXNG or W3C Schema.
Create a new or upload existing customization
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 customization from template
TEI Absolutely Bare
Open existing customization
Submit
Search TEI Guid
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)

Roma: Customize

Roma: generating validators for the TEI

You are currently working on **My TEI Extension**

Set your parameters

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

Set your parameters

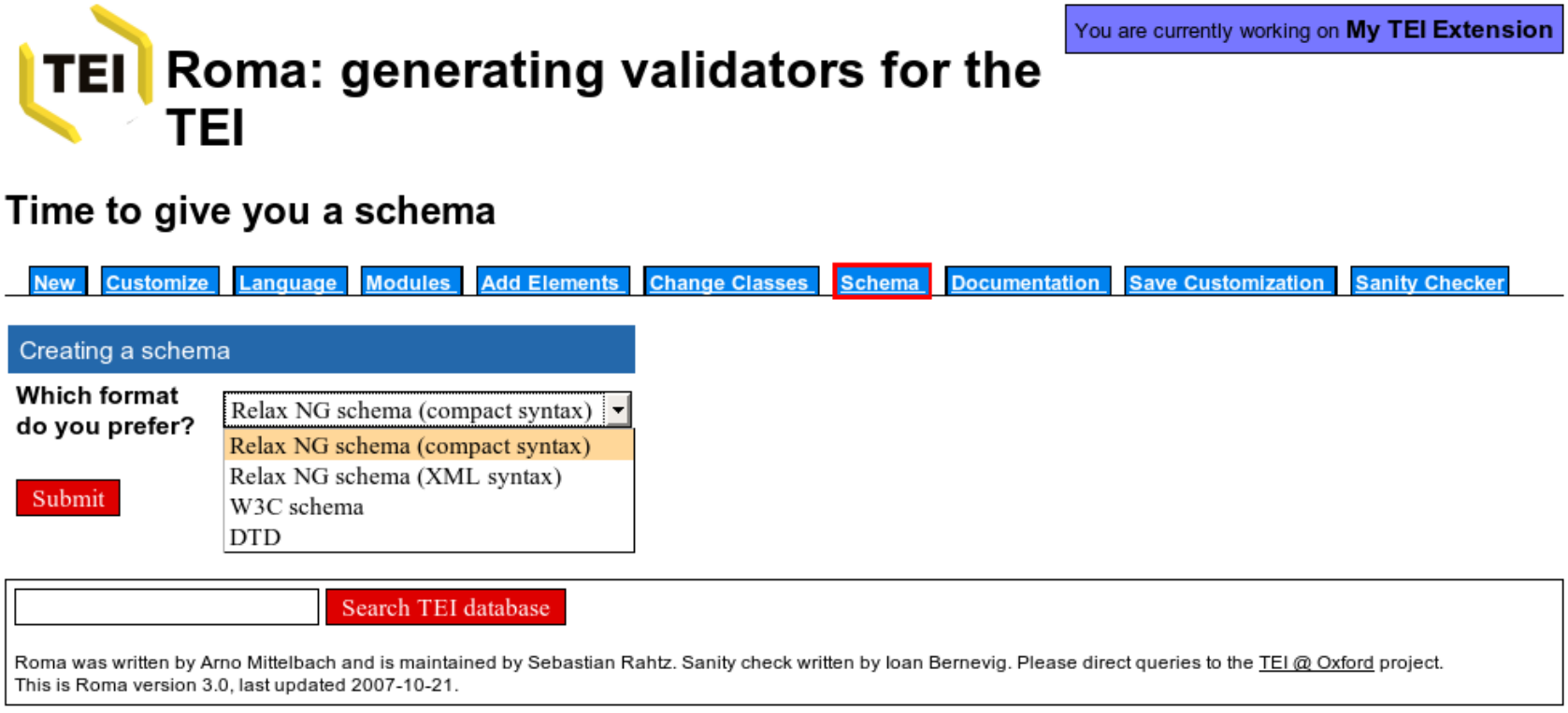
Title	My TEI Extension
Filename	myTei
Prefix for TEI pattern names in schema	
Language	(•) English ( ) Deutsch ( ) Italiano ( ) Español ( ) Français ( ) Portugues ( ) Russian ( ) Svenska ( ) 日本語 ( ) 中文
Author name	generated by Roma 3.0
Description	My TEI Customization starts with modules tei, core, header, and textstructure

Submit

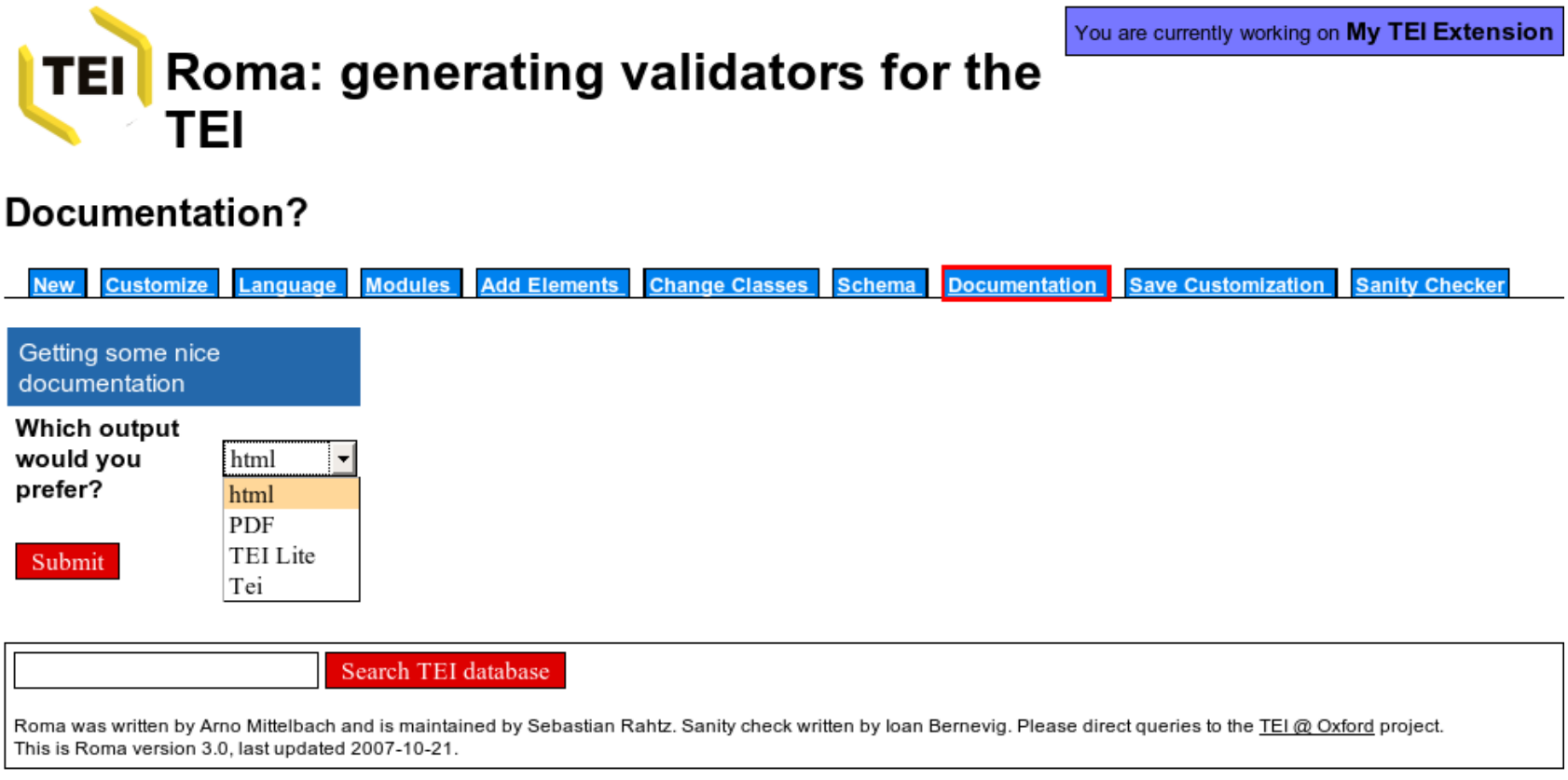
Search TEI database

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.

Roma: Schema


Roma: generating validators for the TEI
You are currently working on My TEI Extension
Time to give you a schema
New
Customize
Language
Modules
Add Elements
Change Classes
Schema
Documentation
Save Customization
Sanity Checker
Creating a schema
Which format do you prefer?
Relax NG schema (compact syntax)
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 Ioan Bernevig. Please direct queries to the TEI @ Oxford project.
This is Roma version 3.0, last updated 2007-10-21.

Roma: Documentation



Subsetting the TEI

Suppose you now feel you want to use some more of the TEI, but not all of it

- Go to Roma…
- Look at [Modules]
- Explore default modules by pointing to main elements (by order of interest). You can throw away most things, but
  - In textstructure, you should really keep <TEI>, <text>, <body> and
  - In core, most people need <p>, <q>, <list>, <pb/> and <head>
  - From header, keep everything unless you really understand the details
- Start checking out elements
- Make editorial choices (numbered vs. unnumbered divs)

Roma: Modules

Roma: generating validators for the TEI

You are currently working on **My TEI Extension**

Modules

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

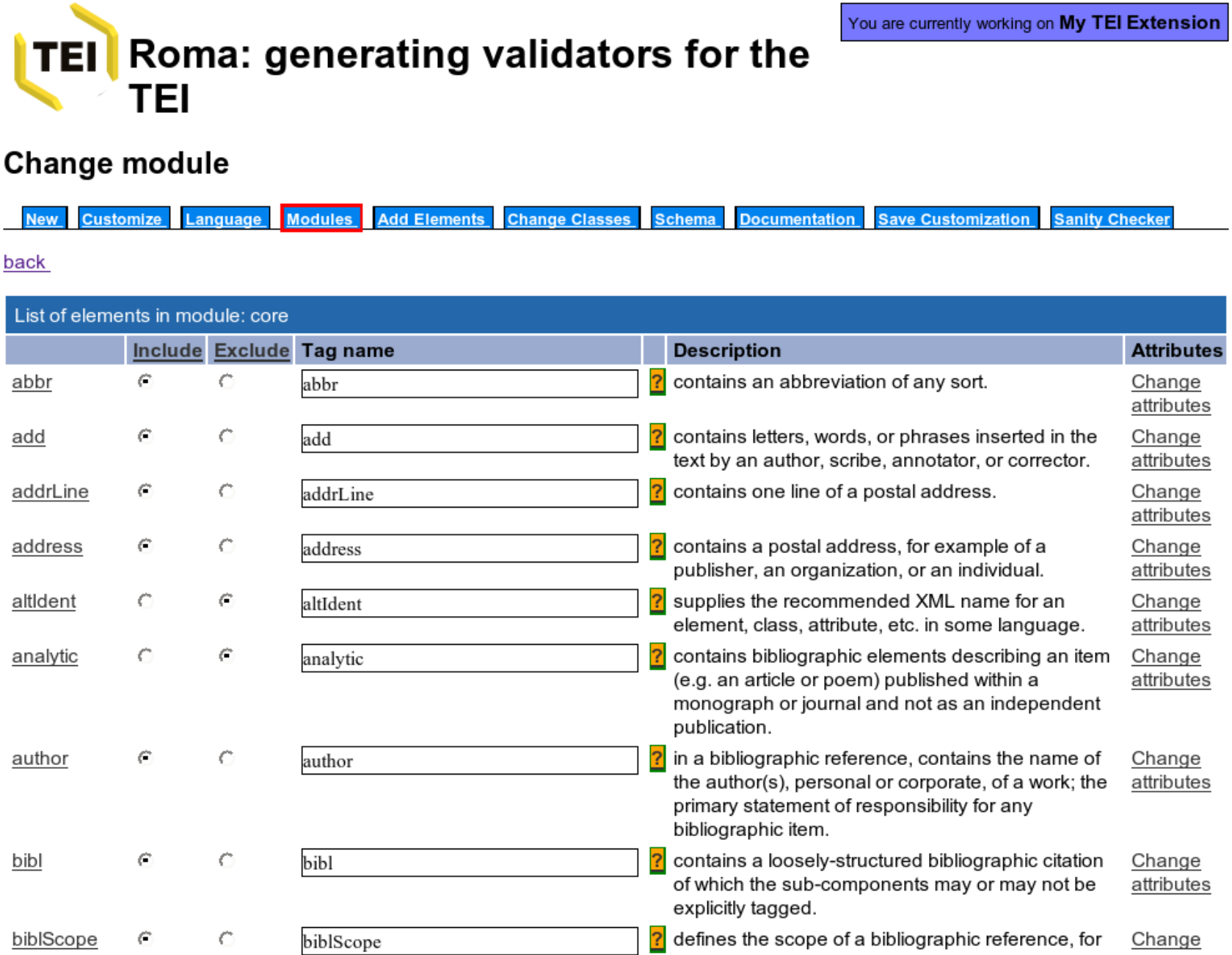
List of TEI Modules

	Module name		A short description	Changes
add	analysis	?	Simple analytic mechanisms	
add	certainty	?	Certainty and uncertainty	
add	core	?	Elements common to all TEI documents	
add	corpus	?	Header extensions for corpus texts	
add	declarefs	?	Feature system declarations	
add	dictionaries	?	Printed dictionaries	
add	drama	?	Performance texts	
add	figures	?	Tables, formulae, and figures	
add	gaiji	?	Character and glyph documentation	
add	header	?	The TEI Header	
add	iso-fs	?	Feature structures	
add	linking	?	Linking, segmentation and alignment	
add	msdescription	?	Manuscript Description	
add	namesdates	?	Names and dates	
add	nets	?	Graphs, networks and trees	
add	spoken	?	Transcribed Speech	
add	tagdocs	?	Documentation of TEI modules	
add	textcrit	?	Text criticism	
add	textstructure	?	Default text structure	

List of selected Modules

remove	core
	tei
remove	header
remove	textstructure

Roma: Change Module



Roma: generating validators for the TEI

You are currently working on **My TEI Extension**

Change module

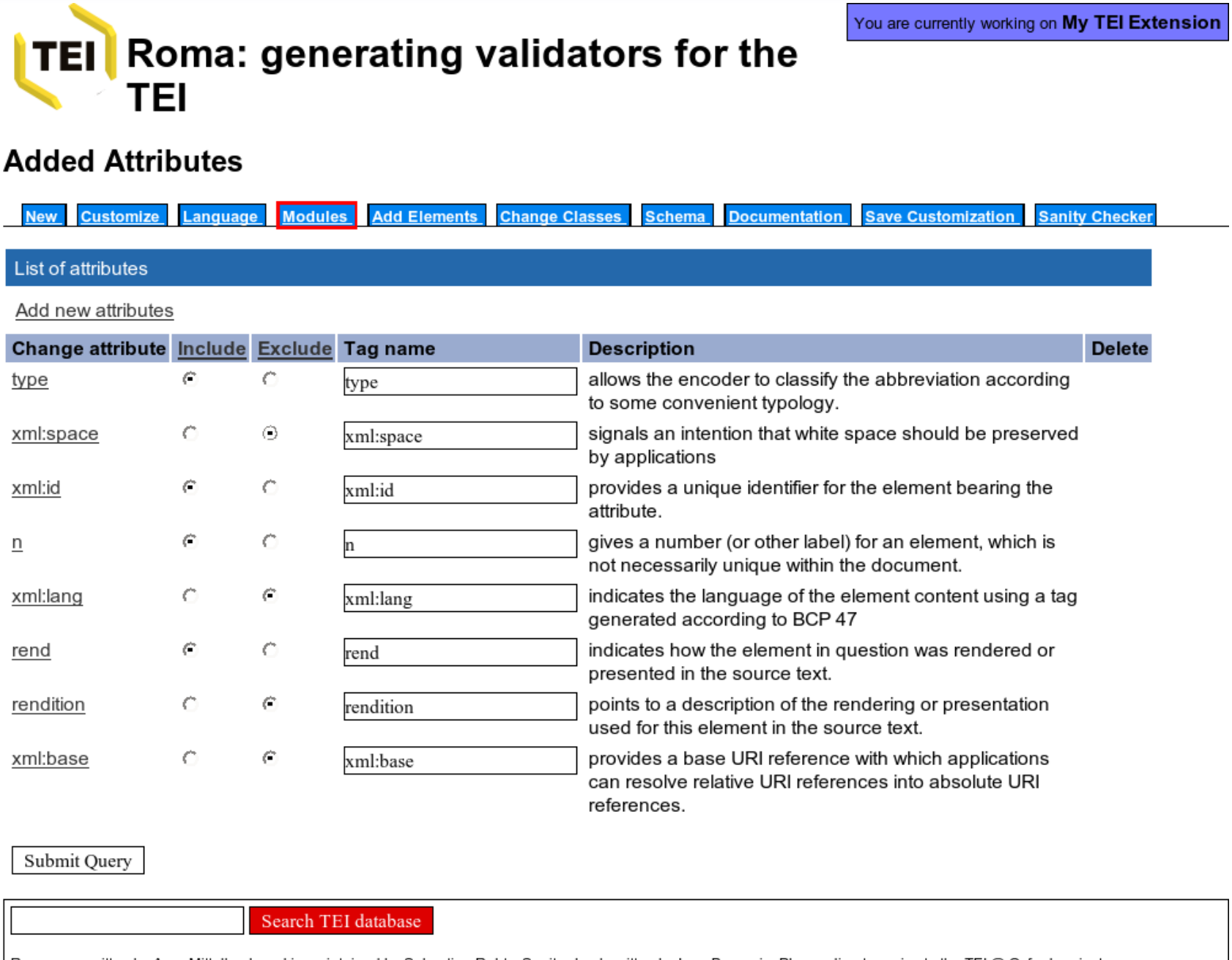
New | Customize | Language | Modules | Add Elements | Change Classes | Schema | Documentation | Save Customization | Sanity Checker

back

List of elements in module: core

	Include	Exclude	Tag name		Description	Attributes
abbr	◉	○	abbr	?	contains an abbreviation of any sort.	Change attributes
add	◉	○	add	?	contains letters, words, or phrases inserted in the text by an author, scribe, annotator, or corrector.	Change attributes
addrLine	◉	○	addrLine	?	contains one line of a postal address.	Change attributes
address	◉	○	address	?	contains a postal address, for example of a publisher, an organization, or an individual.	Change attributes
altIdent	○	◉	altIdent	?	supplies the recommended XML name for an element, class, attribute, etc. in some language.	Change attributes
analytic	○	◉	analytic	?	contains bibliographic elements describing an item (e.g. an article or poem) published within a monograph or journal and not as an independent publication.	Change attributes
author	◉	○	author	?	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.	Change attributes
bibl	◉	○	bibl	?	contains a loosely-structured bibliographic citation of which the sub-components may or may not be explicitly tagged.	Change attributes
biblScope	◉	○	biblScope	?	defines the scope of a bibliographic reference, for	Change

Roma: Change Attributes



TEI Roma: generating validators for the TEI

You are currently working on **My TEI Extension**

Added Attributes

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

List of attributes

Add new attributes

Change attribute	Include	Exclude	Tag name	Description	Delete
type	●	○	type	allows the encoder to classify the abbreviation according to some convenient typology.	
xml:space	○	●	xml:space	signals an intention that white space should be preserved by applications	
xml:id	●	○	xml:id	provides a unique identifier for the element bearing the attribute.	
n	●	○	n	gives a number (or other label) for an element, which is not necessarily unique within the document.	
xml:lang	○	●	xml:lang	indicates the language of the element content using a tag generated according to BCP 47	
rend	●	○	rend	indicates how the element in question was rendered or presented in the source text.	
rendition	○	●	rendition	points to a description of the rendering or presentation used for this element in the source text.	
xml:base	○	●	xml:base	provides a base URI reference with which applications can resolve relative URI references into absolute URI references.	

Submit Query

Search TEI database

Roma was written by Arno Mittelbach and is maintained by Sebastian Rahtz. Sanity check written by Ioan Romascu. Please direct queries to the TEI @ Oxford project.

Roma: Change Attribute Values

Roma: generating validators for the TEI

You are currently working on **My TEI Extension**

Add some attributes

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

go back to list

Add a new attribute

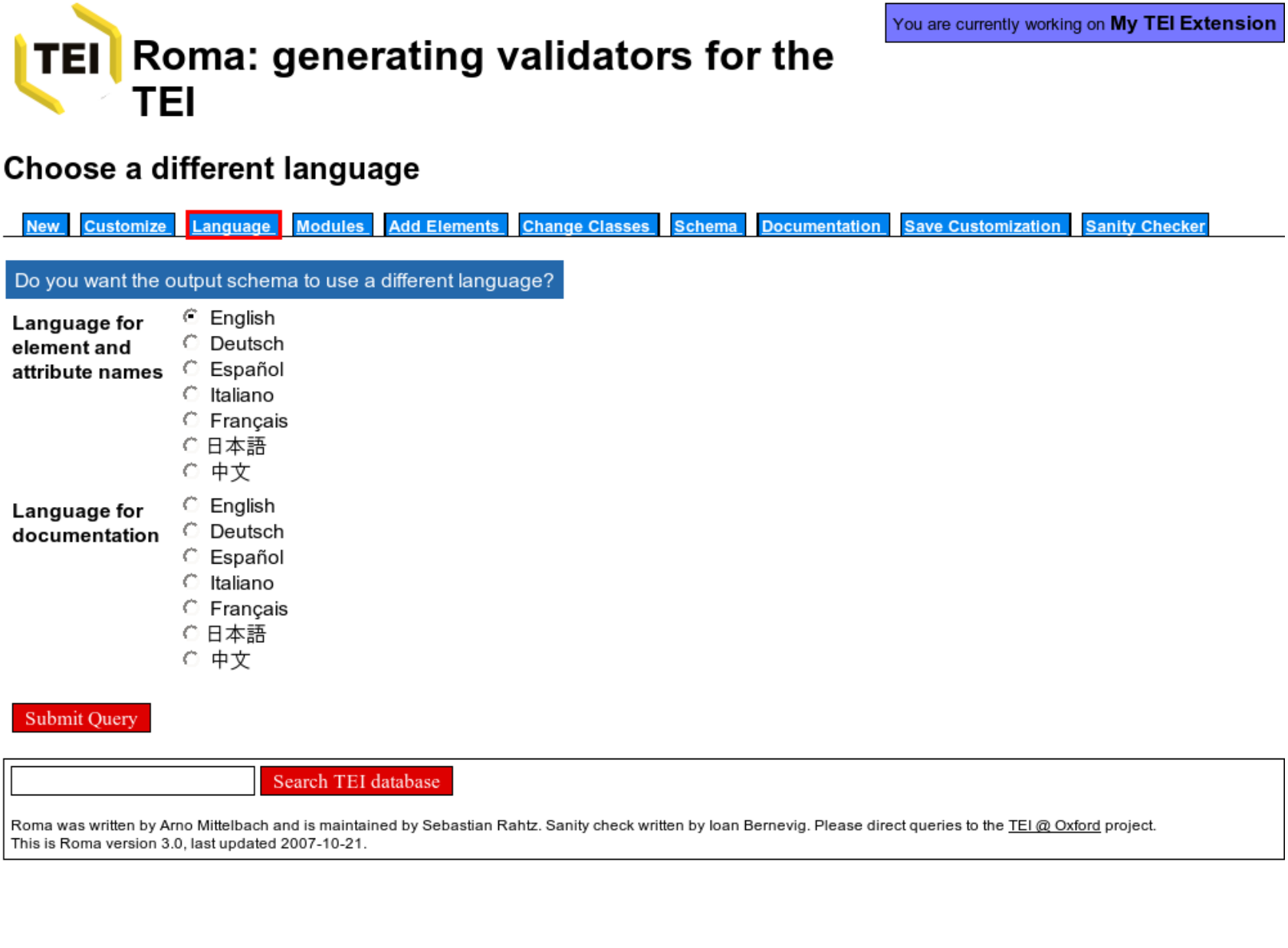
Attribute name	type
Class name	
Is it optional?	○ yes ◉ no
Contents	data.enumerated >= 1 <= 1
Default value	
Closed list?	◉ yes ○ no
List of values	red, blue, green, purple, pink, yellow, other
Description	allows the encoder to classify the abbreviation according to some convenient typology.

Submit Query

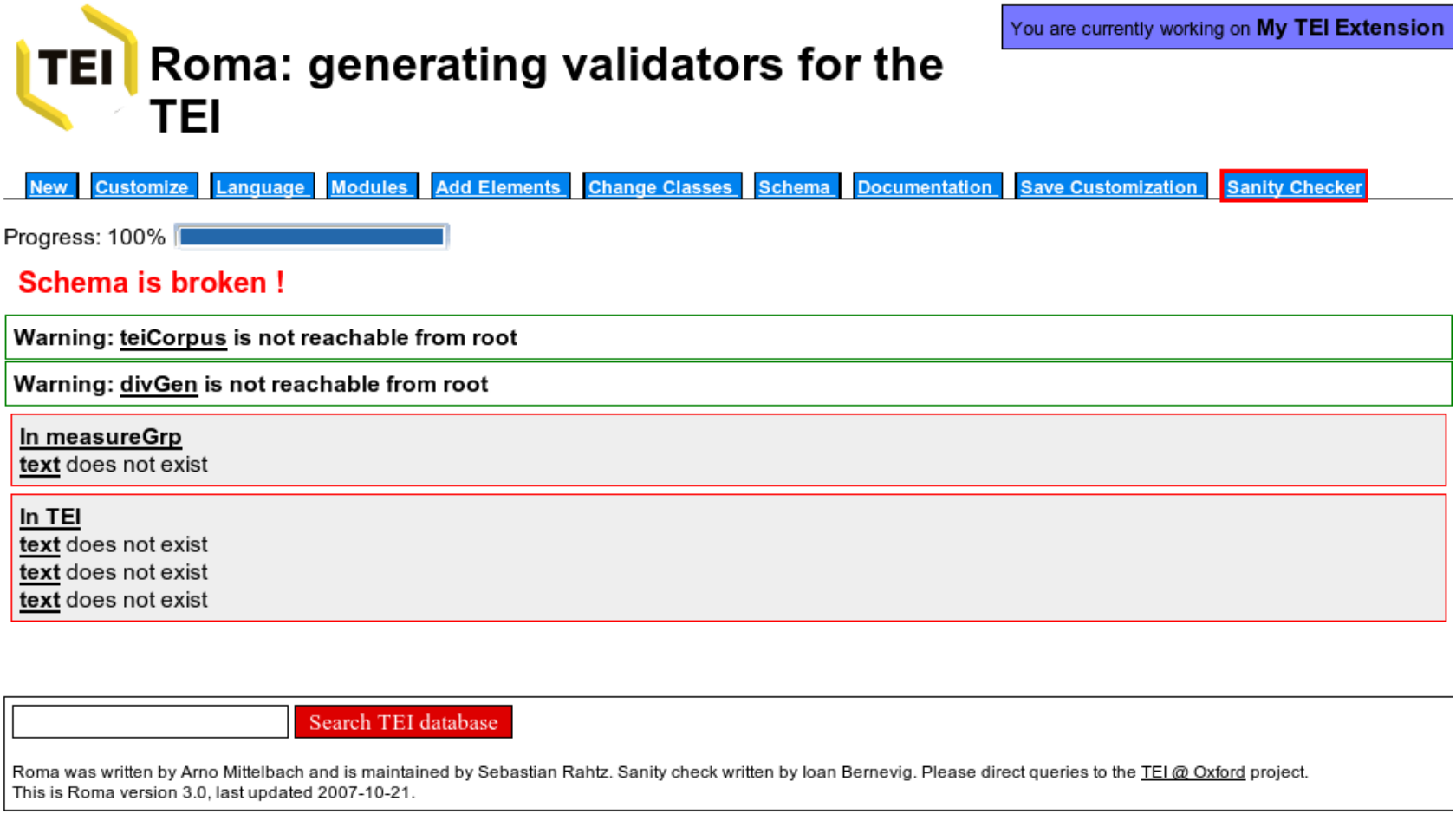
Search TEI database

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.

Roma: Change Language



Roma: Sanity Checker



Understanding ODD

A TEI ODD file can contain as much discursive prose as you want, but as a minimum, it needs a <schemaSpec> element to define the schema it documents

Even more customisation

```
<schemaSpec ident="Chaucer-MoL" start="TEI">
 <moduleRef key="tei"/>
 <moduleRef key="header"/>
 <moduleRef key="core"/>
 <moduleRef key="textstructure"/>
 <moduleRef key="namesdates"/>
 <moduleRef key="transcr"/>
<!-- We don't need these drama elements: -->
 <elementSpec ident="sp" mode="delete" module="core"/>
 <elementSpec ident="speaker" mode="delete" mod-
ule="core"/>
 <elementSpec ident="stage" mode="delete" mod-
ule="core"/>
</schemaSpec>
```

What is happening here?

TEI customizations are themselves expressed in TEI XML, using elements from the tagdocs module.
For example:

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

produces something like TEI Lite, with a slight change

ODD processors

- The TEI maintains a library of XSLT scripts that can generate
  - The TEI Guidelines in canonical TEI XML format
  - The Guidelines in HTML or PDF
  - RELAXNG, DTD, or W3C schema fragments
- The same library is used by the customization layer to generate
  - project-specific documentation
  - project-specific schemas
  - translations into other (human) languages
- We use eXist as a database for extracting material from the P5 sources

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 or additions
- 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.

Expression of TEI content models

Within the class system, TEI elements have to be defined using some language notation; choices include:

1. using XML DTD language (as in older versions of the TEI)
2. using W3C Schema language
3. using the RELAXNG schema language
4. inventing an entirely new abstract language for later transformation to specific schema language

We chose a combination of 3 and 4 — using our abstract language, but switching to RELAXNG for content modelling.

Why that combination?

- Expressing constraints in XML language is too attractive to forego
- There is a clamour for better datatyping than DTDs have
- The schema languages are so good, it is silly to reinvent them
- But we like our class system and literate programming

DTD vs RELAXNG vs W3C Schema

- DTDs are not XML, and need specialist software
- W3C schema is not consistently implemented, its documentation is vast and confusing, and it looks over-complex
- RELAXNG on the other hand…
  - uncluttered design
  - good documentation
  - multiple open source 100%-complete implementations
  - ISO standard
  - useful features for multipurpose structural validation

No contest…

An Example ODD

```
<elementSpec module="spoken" ident="pause">
 <classes>
  <memberOf key="model.divPart.spoken"/>
  <memberOf key="att.timed"/>
  <memberOf key="att.typed"/>
 </classes>
 <content>
  <rng:empty/>
 </content>
 <attList>
  <attDef ident="who" usage="opt">
   <gloss>A unique identifier</gloss>
   <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>
   <datatype>
    <rng:ref name="data.pointer"/>
   </datatype>
  </attDef>
 </attList>
</elementSpec>
```

From which we generate: RNC

```
element pause {pause.content, pause.attributes }
pause.content = empty
pause.attributes =
att.global.attributes,
att.timed.attributes,
att.typed.attributes,
att.ascribed.attributes,
model.divPart.spoken |= pause
att.timed |= pause
att.typed |= pause
att.ascribed |= pause
```

Or DTD

```
<!ELEMENT %n.pause; %om.RR; EMPTY>
<!ATTLIST %n.pause;
 %att.global.attributes;
 %att.timed.attributes;
 %att.typed.attributes;
 %att.ascribed.attributes;>
<!ENTITY % model.divPart.spoken
 "%x.model.divPart.spoken; %n.event; | %n.kinesic;
 | %n.pause; | %n.shift; | %n.u;
```

Or documentation

<pause/>

Home | Table of Contents
Element catalogue

<**pause/**> a pause either between or within utterances. 8.3.2 Pausing	
Module	spoken — 8 Transcriptions of Speech
Attributes	[att.timed att.typed att.ascribed]

Declaration	Compact to XML format
	element pause
	{
	att.global.attributes,
	att.timed.attributes,
	att.duration.w3c.attributes,
	att.typed.attributes,
	att.ascribed.attributes,
	empty
	}

Example	<pause dur="PT42S" type="pregnant"/>
Contained by	*model.global.spoken*
May contain	Empty element

Overriding an attribute value-list in a TEI ODD

```
<elementSpec ident="list" module="core">
 <classes>
  <memberOf key="att.typed"/>
 </classes>
 <attList>
  <attDef ident="type" mode="replace">
   <valList type="closed">
    <valItem ident="ordered">
     <gloss>Items are ordered</gloss>
    </valItem>
    <valItem ident="bulleted">
     <gloss>Items are bulleted</gloss>
    </valItem>
    <valItem ident="gloss">
     <gloss>Part of a gloss list</gloss>
    </valItem>
   </valList>
  </attDef>
 </attList>
</elementSpec>
```

Modifying TEI objects

Understanding classes is critical.

- They group together elements with the same role in the TEI architecture
- They group together elements with the same syntactic behaviour
- Classes can provide attributes for groups of like-minded elements
- The elements in the class will appear in the same content models

The class defines a group of elements belonging to the same family of concepts, elements declare themselves as belonging to a class.

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 `att.type` class gives you a generic type attribute, which can be over-ridden for specific class members

Phrase Level Documentation Elements

- <code> (literal code from some formal language)
- <ident> (an identifier for an object of some kind in a formal language)
- <att> (the name of an attribute appearing within running text)
- <val> (a single attribute value)
- <gi> (the name (generic identifier) of an element.)
- <tag> (text of a complete start- or end-tag, possibly including attribute specifications, but excluding the opening and closing markup delimiter characters)
- <specList> (marks where a list of descriptions is to be inserted into the prose documentation)
- <specDesc/> (a description of the specified element or class should be included at this point)

Specification Elements

- `<elementSpec>` (documents the structure, content, and purpose of a single element type)
- `<classSpec>` (reference information for an element class)
- `<macroSpec>` (documents the function and implementation of a pattern)

Common Elements (1)

- Description:
  - <remarks> (any commentary or discussion about the usage of an element, attribute, or class)
  - <listRef> (a list of significant references to places where this element is discussed)
- Examples
  - <exemplum> (a single example demonstrating the use of an element)
  - <eg> (any kind of illustrative example)
  - <egXML> (a single well-formed XML example demonstrating the use of some XML element or attribute)
- Classification
  - <classes> (the classes of which the element or class is a member)
  - <memberOf> (class membership of the parent element or class)

Common Elements (2)

- Element Specifications
  - `<content>` (the text of a content model for the schema)
  - `<attList>` (documentation for all the attributes associated with this element, as a series of `<attDef>` elements)
- Attributes
  - `<attDef>` (definition of a single attribute)
  - `<datatype>` (schema datatype for the attribute value)
  - `<defaultVal>` (default declared attribute value)
  - `<valDesc>` (description of any attribute value)
  - `<valList>` (a list of attribute value items)
  - `<valItem>` (a single attribute value item)

Defining a TEI Schema

- A schema helps you know a document is valid in addition to being well-formed
- A TEI schema is a combination of TEI modules, optionally including customizations of the elements/attributes/classes that they contain
- This schema is defined in an application-independent manner with a TEI ODD (One Document Does it all) file which allows for:
  - creation of a schemas such as DTD, RELAX NG or W3C Schema
  - internationalized documentation which reflects your customization of the TEI
  - documentation of how your schema differs from tei_all that is suitable for long-term preservation

A word of caution

Remember

- The TEI is not a monolithic environment
- Very few things are really mandatory …
- …but the TEI is more than just a market place
- Basic document structure must be preserved

The TEI is a powerful environment for working with elements and producing documentation, but do not abuse it.