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?

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

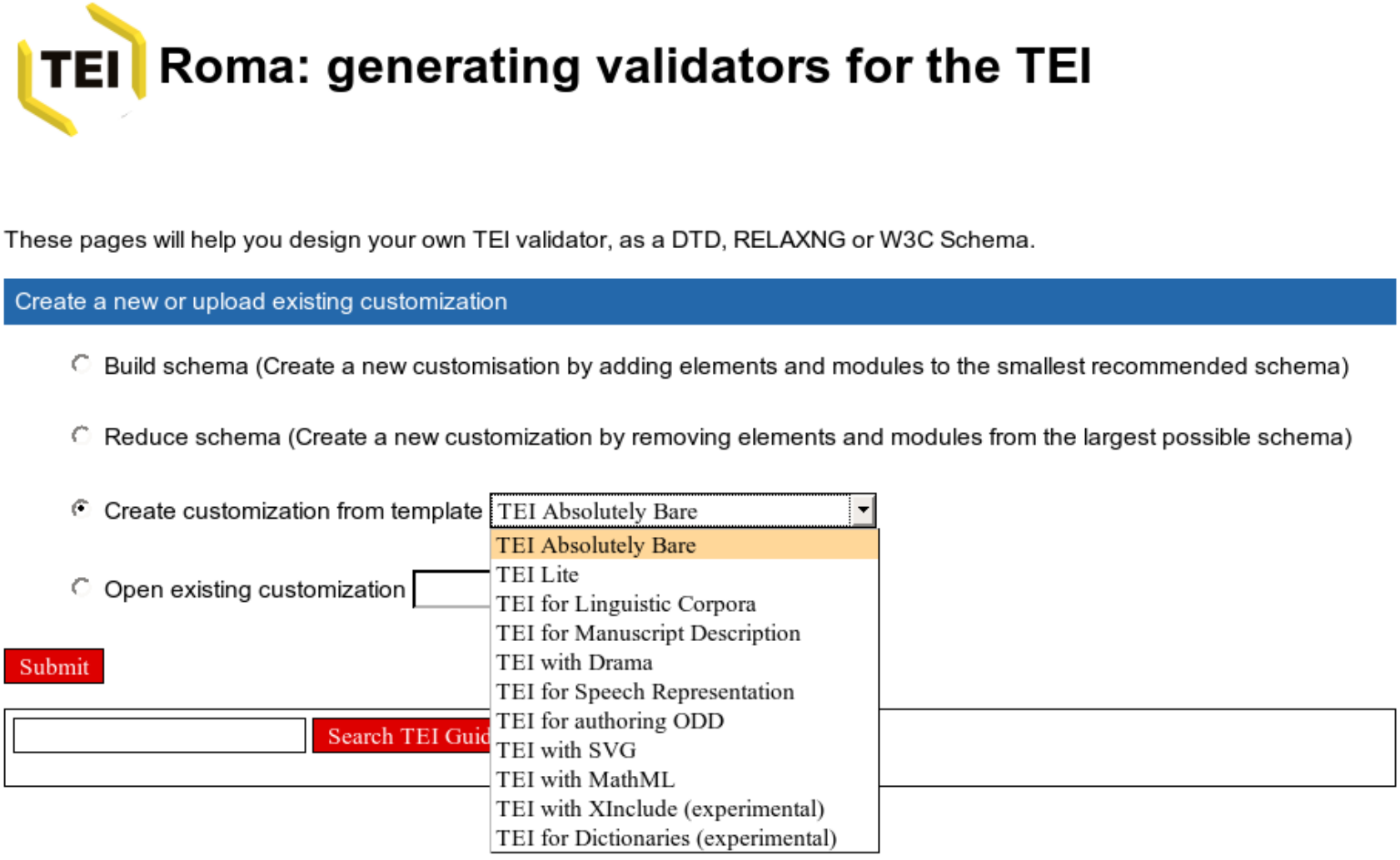
How do you choose?

- 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/

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

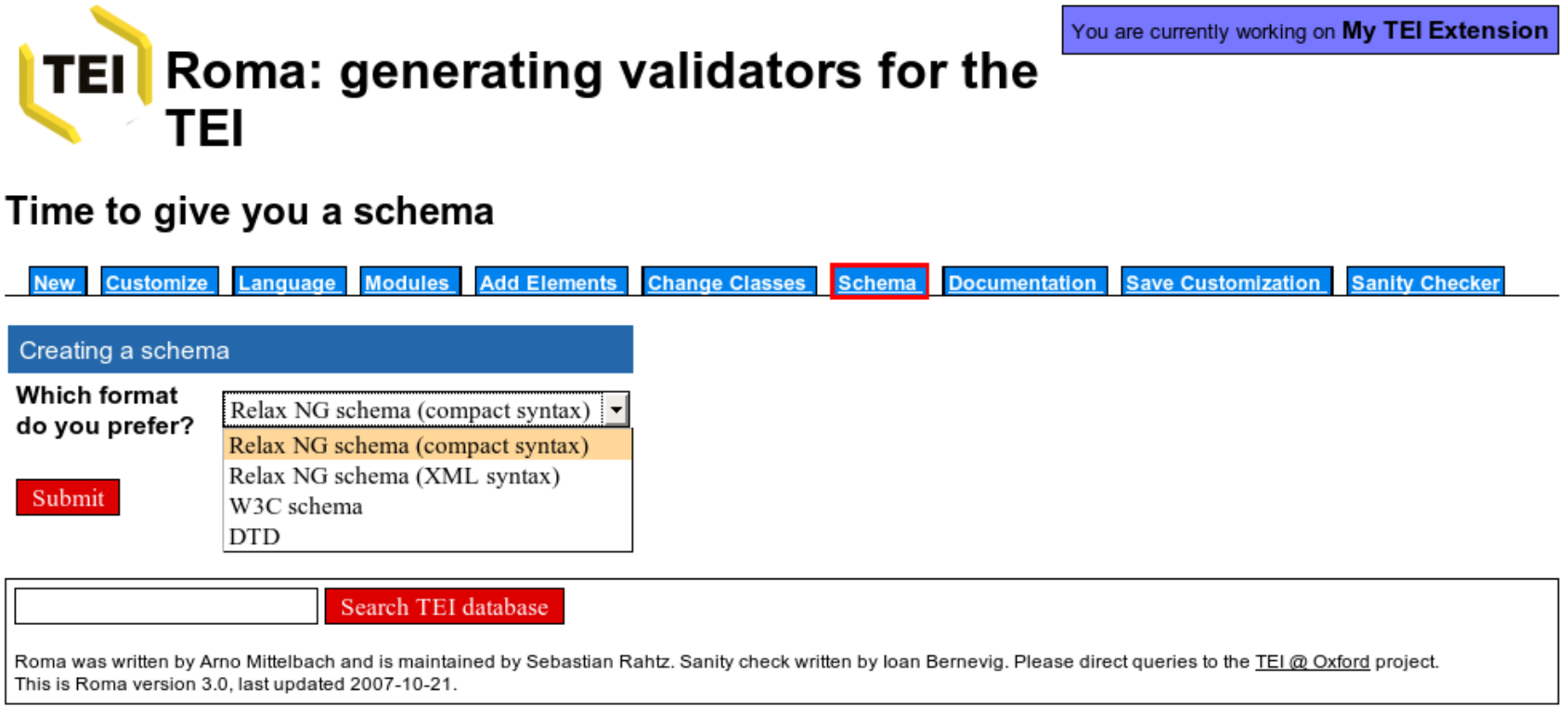
Title	My TEI Extension
Filename	myTei
Prefix for TEI pattern names in schema	
Language	(•) English ( ) Deutsch ( ) Italiano ( ) Español ( ) Français ( ) Portugues ( ) Russian ( ) Svenska ( ) 日本語 ( ) 中文
Author name	generated by Roma 3.0
Description	My TEI Customization starts with modules tei, core, header,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

Submit

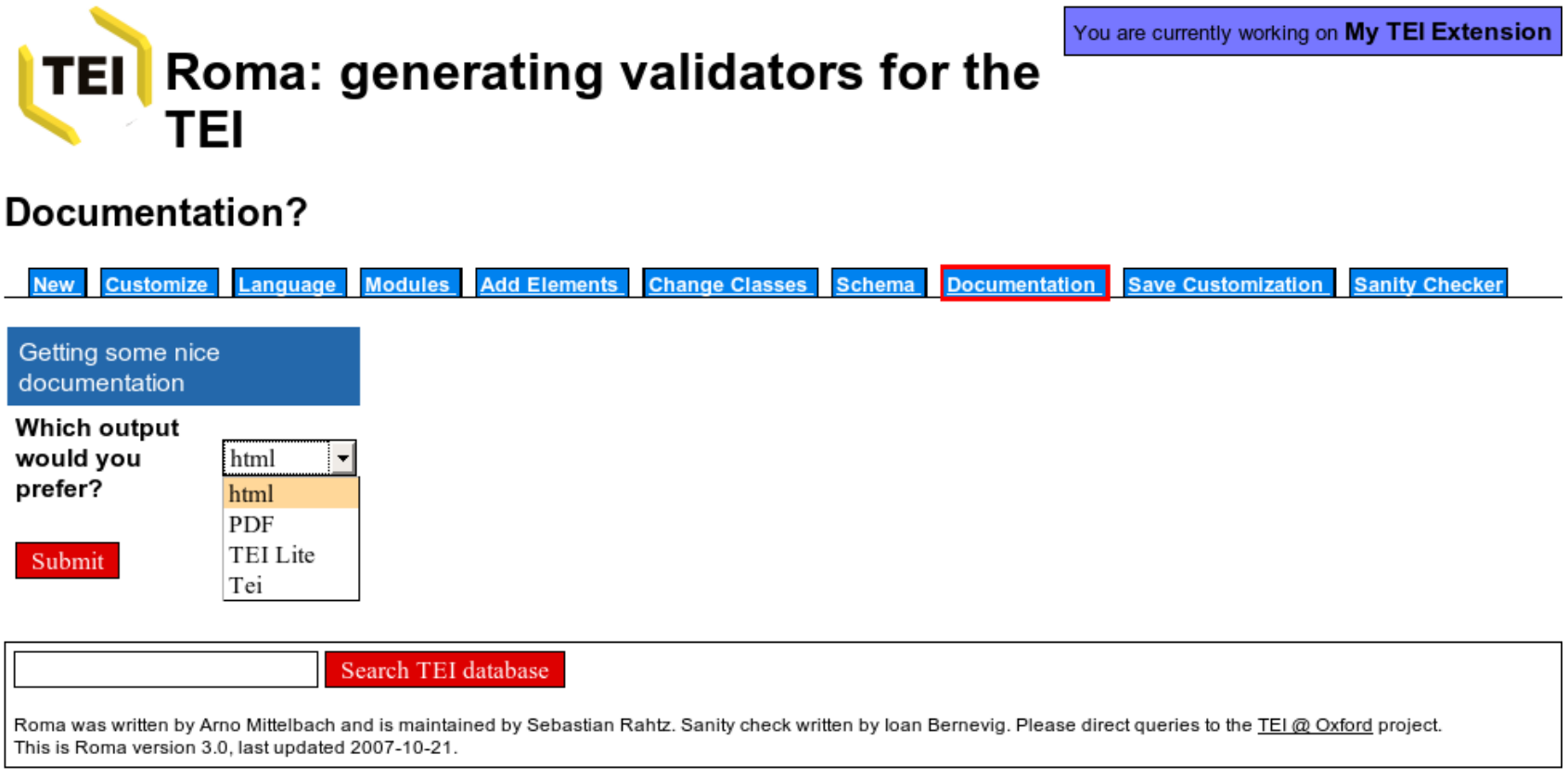
Search TEI database

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



What did we just do?

We processed a pre-existing ODD file which contained (as well as some discursive prose) the following schema specification:

```
<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" mod-
ule="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

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

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

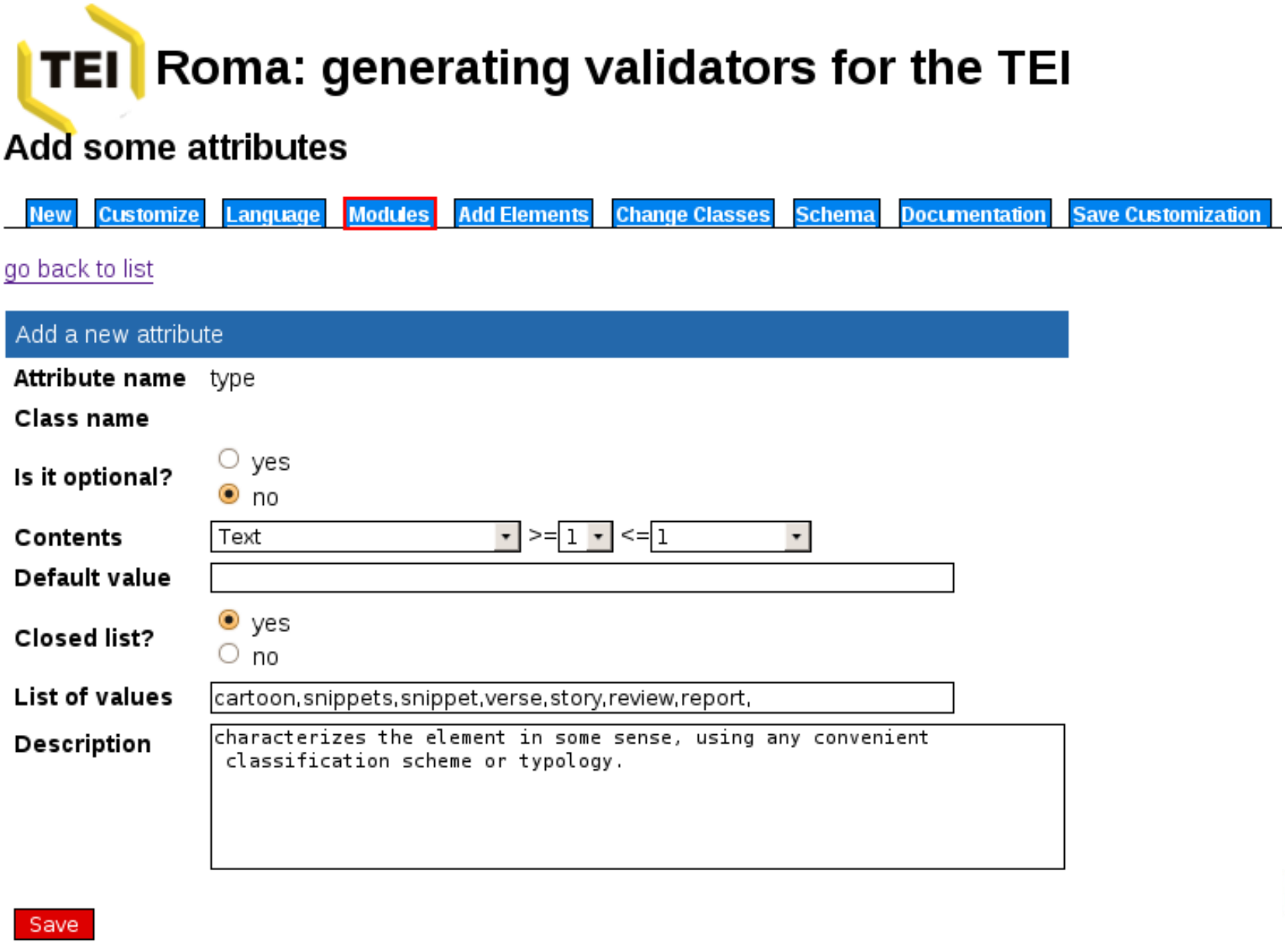
New | Customize | Language | Modules | Add Elements | Change Classes | Schema | Documentation | Save Customization | Sanity Checker

List of attributes: div

Add new attributes

Change attribute	Include	Exclude	Name	Description	Delete
org	○	●	org	specifies how the content of the division is organized.	
sample	○	●	sample	indicates whether this division is a sample of the original source and if so, from which part.	
part	○	●	part	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.	
type	●	○	type	characterizes the element in some sense, using any convenient classification scheme or typology.	
subtype	○	●	subtype	provides a sub-categorization of the element, if needed	
decls	○	●	decls	identifies one or more declarable elements within the header, which are understood to apply to the element bearing this attribute and its content.	
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	

Roma: constraining attribute values



What did we just do?

Our ODD now includes something like this:

```
<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:

```
<valItem ident="cartoon">
 <gloss>contains a humorous picture, usually with
   dialogue underneath</gloss>
</valItem>
```

Defining a new element

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 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 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.

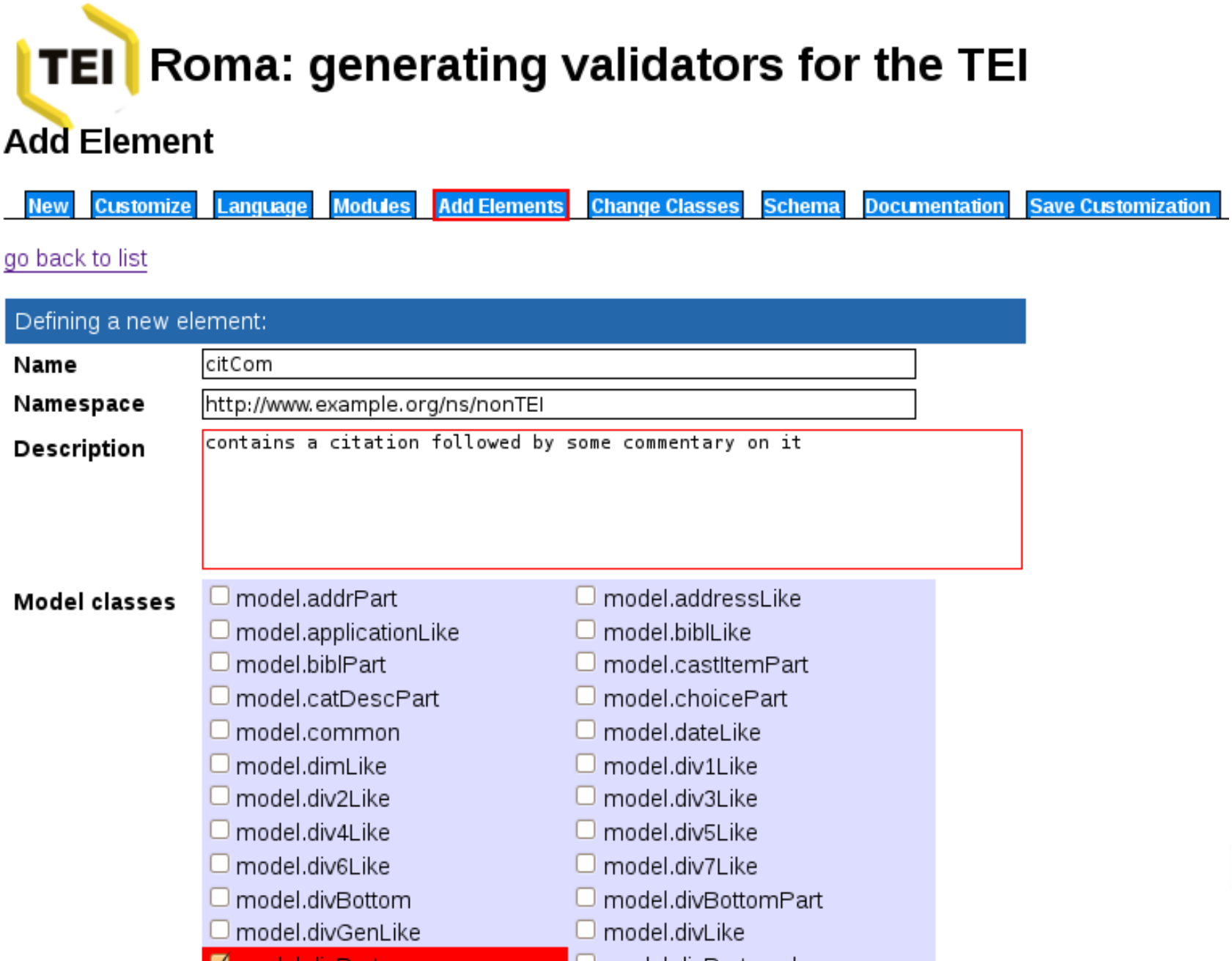
What other elements can contain it? It can only appear within a division, like a paragraph.

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



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

Roma: Defining a new element 2

Attribute classes

- [ ] att.ascribed
- [ ] att.canonical
- [ ] att.coordinated
- [ ] att.damaged
- [ ] att.datable
- [ ] att.datable.iso
- [ ] att.datable.w3c
- [ ] att.declarable
- [ ] att.declaring
- [ ] att.dimensions
- [ ] att.divLike
- [ ] att.duration
- [ ] att.duration.iso
- [ ] att.duration.w3c
- [ ] att.editLike
- [ ] att.enjamb
- [ ] att.entryLike
- [ ] att.handFeatures
- [ ] att.identified
- [ ] att.internetMedia
- [ ] att.interpLike
- [ ] att.lexicographic
- [ ] att.measurement
- [ ] att.metrical
- [ ] att.msExcerpt
- [ ] att.naming
- [ ] att.personal
- [ ] att.placement
- [ ] att.pointing
- [ ] att.pointing.group
- [ ] att.ptrLike.form
- [ ] att.ranging
- [ ] att.rdgPart
- [ ] att.segLike
- [ ] att.sourced
- [ ] att.spanning
- [ ] att.tableDecoration
- [ ] att.textCritical
- [ ] att.timed
- [ ] att.transcriptional
- [ ] att.translatable
- [x] att.typed
- [ ] att.xmlspace

Contents User content

What did we just do?

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

```
<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*

```
<elementSpec ident="div" module="teistructure" 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