TEI Customization with Roma and ODD

April 2015

Contents

1 Part 1: Customizing the TEI with Roma

1.1 Objectives

The aim of this part of the exercise is to make you familiar with customizing the TEI, by using the web-based Roma application and the oXygen editor. You may not have time to do both parts of this exercise, but if not you can do the second part at another time. You will

- choose appropriate TEI modules
- combine TEI modules to make a customization
- select a subset of elements from a TEI module
- set up and use a customised schema in oXygen
- generate documentation and schemas using oXygen

1.2 Getting starting

First, let's consider the initial situation you find yourself in when you simply use all of the TEI:

- Start up oXygen.
- Click on the New icon, top left (or select New from the File menu, or type CTRL-N) to open the New dialogue
- Choose Framework Templates, then TEI P5, then All to select a schema which permits every TEI element in existence.
- Click the Create button below. oXygen displays a TEI document for you to complete.

Remembering that oXygen helps you by indicating which elements are available at any point in your document:

- Put the cursor inside the `<p>` element in your document.
- Type '<'. oXygen gives you a menu of all the elements available according to the TEI All schema.
- Scroll down the list. A pop-up containing a brief description of each appears as you do so, which is better than nothing. Do you feel confident that you'd know how to use, for example, the `<camera>`, `<incident>`, `<metamark>`, or `<notatedMusic>` elements though?
- Type ESC to leave the menu and delete the < you just added.
- You are spoilt for choice... but in any particular project it's hardly likely you will need all these elements; the same thing applies to attributes, of course. Worse, with so many possibilities (some of which overlap) it's easy to make inconsistent choices. And any software which wants to process your documents must be prepared for all eventualities, which makes the whole thing needlessly complicated.

Can we do better?

1.3 Making a schema with Roma

Roma is a web application developed for the TEI which can be used to create your own TEI customization. Let's use it to make a schema using just the elements that we need for the markup of a digital edition with a traditional critical apparatus.

- Open a web browser and visit the site http://www.tei-c.org/Roma/.
- The initial screen allows you to choose from a number of starting points:
  1. start from a mimimal number of TEI modules, to which you can add new elements or modules and remove existing elements;
  2. start from the TEI All schema you used at the start of this exercise and remove things you don't want
  3. start from one of a small number of pre-defined templates:
  4. start from a pre-existing TEI customization file such as the popular TEI Lite
  5. start from a customization you (or someone else) has already made

We suggest you start from TEI Bare, and build up, adding just what is needed

- Click the third radio button (next to ‘Create a new customization...’) and make sure that template TEI Absolutely Bare is selected in the dropdown menu.

- I press the red start button to begin!

On the next screen, set the parameters as follows:

- *Title:* Change this to "TEI for Critical Apparatus".
- *Filename:* Change this to (for example) appcrit (this is an XML identifier and so may not contain spaces).
- Leave the Namespace and Prefix fields unchanged
- *Language:* You can continue to work in English, but if you prefer German, Italian, Spanish, French, Portuguese, Russian, Swedish, Chinese or Japanese, feel free to select the appropriate radio button.
- *Author name:* Type in your name
- Change the description to something like "A minimal TEI tagset for the encoding of traditional critical apparatus"
- Click the Save button at the foot of the page.

If you chose a language other than English, you'll see that the Roma interface language has changed accordingly. You should also visit the Language (or equivalent) page to set the language used by your generated schema

1.4 Select your modules

A *module* is a group of TEI elements. Every TEI element is declared in a particular module. Some modules are very general and have many components; others are more specialised. For example, if you are encoding a dictionary, or a speech transcript, you will need to choose the module for Dictionaries, or that for Transcribed Speech respectively, but the elements provided by these modules are of less interest in other types of document.

- Click the Modules button on the toolbar to see the modules from which your schema is derived
- The list of selected modules displayed on the right of the screen, contains just three modules.
- Obviously, we will need to add some elements from the module for Critical Apparatus. **Click on the word Add preceding the module name textcrit in the list on the left.**
- The module is added to the list on the right, which now contains the modules: tei (Roma doesn't allow you to remove or modify this infrastructural module), core, header, textcrit, and textstructure.
- For the exercise, you will need a few more elements, taken from other modules. **Add the modules transcr and linking to the list.**

1.5 Including and excluding elements

Selecting a module by default includes all the elements defined by that module, which may not be what we want.

- Click on the word 'core' in the right hand list (nb. not the word 'remove' but the name of the module). A table listing all the elements provided by this module is displayed.
- Each row of the table contains:
  - the canonical name of the element
  - an indication of its Inclusion or Exclusion in the current schema

  - the name of this element in the current schema (normally this is the canonical name, but Roma allows you to rename elements, for example if you are working in a language other than English, but generally it is a better idea to change the language of the description, but leave the element names the same)
  - a question mark link to the full reference information for this element
  - a brief description of the element
  - a link which allows you to change the element's attributes

- This interface allows you to explore in detail all the elements provided. Click on the question mark link for any of the elements which interest you to read more about them.

- The interface also controls your selection of elements. Use the Include or Exclude button in the heading of the table to include or exclude by default all the elements defined by a module. Then click on the button beside any element whose status you wish to change in order to include (or exclude) it. Usually it is more convenient to start by excluding all the elements and then adding back the ones you actually want.

Here are the elements you need to include in your schema:

- From the core module you will need the elements `<add>`, `<choice>`, `<corr>`, `<del>`, `<head>`, `<item>`, `<lb>`, `<list>`,`<p>`, `<sic>` and `<title>`.

- **When you've finished, don't forget to click the red Save button at the foot of the page!**

- From the textstructure module you will need the elements `<TEI>`, `<body>`, `<div>` and `<text>`

- From the header module you will need the elements `<fileDesc>`, `<handNote>`, `<profileDesc>`, `<publicationStmt>`, `<sourceDesc>`, `<teiHeader>` and `<titleStmt>`.

- From the textcrit module you will need the elements `<listWit>`, `<witness>`, `<app>`, `<lem>`, and `<rdg>`.

- From the transcr module you will need the elements `<handNotes>`, `<subst>`, and `<supplied>`.

- And finally, from the linking module you will need the element `<anchor>`.

Note that in this exercise we only control the presence or absence of elements in a schema. Selecting from the possible attributes for those elements is also possible using Roma, but we do not explore it yet.

1.6 Creating a schema

- Click the Schema tab. You can choose amongst several schema languages because the TEI system is defined (as far as possible) independently of any particular language.

- We recommend you to generate your schema in RELAXNG either in compact or in XML syntax.

- Click on the red Generate button and save the schema file which Roma sends you in your working folder.

- **Do not shut down your web browser, leave Roma open!**

1.7 A document that does NOT allow everything

- Start up oXygen, or switch to it if it is already running.
- Click on the New icon, top left (or select New from the File menu, or type CTRL-N) to open the New dialogue.
- Choose New Document, then XML Document.
- Click the Customize button below. oXygen displays the Customize Editor dialog box.
- From the dropdown menu at the far right of the Schema URL window choose Browse for local file
- Navigate to your working folder, select the schema file you have just created, and click the Open button
- Information about your schema is displayed. Click the Create button to create an empty document which will use your schema.
- Notice what oXygen adds to the top of the document. This tells it where to get the schema it needs to validate.
- You may like to check what elements are available within a `<p>` element now... it should be less than with the full TEI schema.
- Another way to do this is to associate the schema with an already open document. (Using the Document, Schema, then Associate Schema menus.)

1.8 Generating documentation

Every project needs some internal documentation a bit more discursive and explanatory than a RELAXNG schema. Roma can be used to generate such documentation automatically.

- Return to Roma and select the Documentation tab.
- Choose 'HTML web page' and click the Generate button.
- Roma sends you an HTML file, which you can save in your working folder
- Click on the file to open it. It begins with a list of elements you chose, each one with a link to its full documentation
- You can also generate this documentation in PDF format, if you prefer.

You can modify the documentation, for example to include examples taken from your own material, or to exclude irrelevant commentary. This cannot however be easily done via the Roma interface: you need to access the underlying source code, the ODD, which Roma is editing for you. (We will do this in part 2 of this exercise.)

1.9 Optional: working with attributes

Our schema now includes only the elements we want, but we would like to constrain it further. For example, we have the `type` attribute on the `<div>` element to categorize sections by means of a code. It would be useful to make sure that this code is always present, and also to make sure the values used all come from the same fixed list.

Go back to Roma. (If you have closed the browser, you will need to restart Roma and reload the session you saved earlier). Go to the *Modules* tab and click on `textstructure` in the right-hand column. Find `<div>` and click on `Change attributes` on the right-hand side. This will show you all the attributes of `<div>`. Click on `type`, and you will be able to change its properties:

1. Change the `Is it optional` radio button to make it compulsory
2. Change the radio button for `Closed list?` to make it a closed list
3. In the box for `List of values`, type (for example)

```
cartoon,verse,prose,drama
```

   (ie a list of possible values, separated by commas, but *without any spaces*).

4. You can change the description if you like.
5. Click the red `Save` button

Now Generate a new schema, just as you did as before, and save it, over-writing the file that you created before. Reload your file in oXygen. Try creating a `<div>` and supplying an illegal value to check that your list of legal values is being respected.

You can return to Roma and explore changing other attributes, including those defined by attribute classes, if you have time. For example, if you wanted to make the *@type* attribute compulsory *everywhere* (probably not a good idea...), you would edit the class att.typed.

1.10 Limitations of Roma

Roma is a web-based interface to a set of XSLT stylesheets supported by the TEI. These same stylesheets are also usable within oXygen, so much of the functionality of Roma is accessible via oXygen, and also by other TEI tools which use the same stylesheets. Using Roma you can select elements and modules, add constraints on attribute values, and (within limits) add new elements to existing TEI classes. It thus provides a good starting point to explore the world of TEI customisation. For more advanced use, however, it is better to manipulate the XML TEI format of your ODD directly with a tool such as oXygen.

2 Part 2: TEI Customisation Markup

2.1 Objectives

The aim of this part of exercise is to get you to understand a TEI ODD customzation created by Roma, and create one from scratch using the newer features of ODD. You should be able to

- add documentation to a TEI ODD customization
- make modifications not possible in Roma

2.2 Saving a customization

A customization file (an ODD) is just another TEI document like all the others. You can save it and re-edit it using any XML editor, not only within Roma.

- Return to Roma, and click the Save Customization tab.
- The browser should download an XML with the same name as your schema, e.g. appcrit.xml. Save this file in your working folder.
- Start oXygen again, open the file appcrit.xml, and have a look at it.

As you can see, the file includes material inherited from the schema from which it was derived. If you built it up from TEI Bare there will be very little; if you started from another customization there will be rather more. In either case, you are free to edit this TEI document, like any other, adding discussion of your house rules and practices, your own examples etc, in order to produce a traditional Users Manual for your TEI project.

Look at the `<schemaSpec>` element at the end of the ODD. This is where your customized schema is defined. As you see, it contains a `<moduleRef>` for each module chosen, together with an indication of the elements selected from that module.

2.3 Adding documentation to a TEI ODD customization

If you have time, use oXygen to enrich your ODD with some discussion of the elements you have added to the schema.. You can reload this file in Roma to regenerate the documentation, but it is quicker to do this directly within oXygen.

- Click the spanner/wrench icon, select Transformation->Configure Transformation Scenario from the Document menu, or type CTRL-SHIFT-C
- Oxygen shows you the list of transformations available for an ODD file: you can convert it to HTML, DOCX, EPUB, and various schema formats.
- Check the tick box for the conversions that interest you (we suggest XHTML and RNC) and press the Apply associated button
- Each transformation requested is run for you and the results displayed in a separate window
- Once you have configured the transformations you want, you can run the process again simply by clicking on the big red triangle button, selecting Transformation->Apply Transformation Scenario from the Document menu or typing CTRL-SHIFT-T.
- You can also use Roma to generate this documentation.

2.4 Modifications not possible in Roma

One of the things you cannot do it Roma is provide documentation for value lists.

1. In your ODD file, change the *@place* attribute of `<add>` to give it a closed `<valList>` containing just the values you want to support (eg just ‘above’ and `<below>`). A bit like this:

```
<elementSpec mode="change" ident="add">
 <attList>
  <attDef ident="place" mode="change">
   <valList type="closed" mode="replace">
    <valItem ident="above">
     <desc>ABOVE</desc>
    </valItem>
    <valItem ident="below">
     <desc>BELOW</desc>
    </valItem>
   </valList>
  </attDef>
 </attList>
</elementSpec>
```

   Look carefully here at the use of *@mode* and make sure you understand why the `<valList>` is in ‘replace’ mode.

2. Generate documentation and check you can see your changes.
3. Generate a RELAX NG schema, and use it in oXygen to edit a file. Do you see the documentation values you specified?
4. do the same thing, providing a `<valList>` on another element. You will have to decide whether to use 'replace' or 'add' mode.

Suppose in your prose section you want to discuss a particular group of elements? add this to your ODD:

```
<specList>
 <specDesc key="p"/>
 <specDesc key="p" atts="+"/>
 <specDesc key="p" atts="xml:id"/>
 <specDesc key="valList"/>
 <specDesc key="valList" atts="type"/>
</specList>
```

make documentation and see the effect of each of these `<specDesc>` elements.

Do you need to include examples of the use of XML elements in your documentation? To do this use the `<egXML>` element which is in a different namespace "http://www.tei-c.org/ns/Examples". This allows you to include any well-formed XML fragments as documentation.

```
<egXML xmlns="http://www.tei-c.org/ns/Examples">

<person xml:id="Ovi01" sex="1" role="poet">
 <persName xml:lang="en">Ovid</persName>
 <persName xml:lang="la">Publius Ovidius Naso</persName>
 <birth when="-0044-03-20"> 20 March 43 BC <placeName>
   <settlement type="city">Sulmona</settlement>
   <country key="IT">Italy</country>
  </placeName>
 </birth>
 <death notBefore="0017" notAfter="0018">17 or 18 AD <placeName>
   <settlement type="city">Tomis (Constanta)</settlement>
   <country key="RO">Romania</country>
  </placeName>
 </death>
</person>

</egXML>
```

3 Application to your own project

In order to properly design and construct a schema you must have a very good understanding of the documents you will be encoding. Moreover, you must have a sense of the research that will be done on the documents and what should be encoded, and how, in order to enable this research.

3.1 Document Analysis

Think about the documents of your own project:

- What textual features does it have?

- Given limited resources, which are you going to encode?
- What edge cases or unusual items might there be?
- Is there anything that might be problematic to encode?

3.2 Schema Design

Once you have analysed the documents you must design the schema. This is often more difficult to get right than one might expect. Think about the encoding needs of your own project:

- What modules of the TEI will in need?
- What elements does it need?
- What attributes do these need?
- How should their values be limited?
- How should the values of the *@type* attribute be constrained? For which elements?
- What about 'global' attributes like *@rend*? Should these be constrained globally or for individual elements?
- How should these be documented in a local encoding manual stored in your ODD file?
- What sort of examples should be included in such a manual?

Spend some time thinking about your own project and what your ODD should consist of? Make a list of the modules and elements you believe you need, and any attributes for which you would constrain the value lists.