# All roads lead to .. TEI

*Sebastian Rahtz and Lou Burnard*

## 1 Before you start

In this exercise, you will use **Roma**, a web tool available from the TEI web site and also included on your TEI Knoppix CD, to make an XML schema. If you are not using the TEI Knoppix CD, you will need access to the net and a web browser such as IE, Mozilla, or Opera. Once you have your schema you will also need an XML-aware editor to use it (in our case, Gnu Emacs).

Using the Knoppix CD, you may want to choose a different keyboard. This version of Knoppix starts up with a UK keyboard. If you want a different one, click on the national flag in the bottom right corner. If the flag you want doesn't appear for selection, choose the **Configure** option to find it, and then select it.

Our goal is to make a very very simple schema, which we can use to mark up a multimedia document. We don't need anything like the full complexity of TEI Lite, much less the full TEI. We just want to mark up headings, dates, lists, paragraphs, figures and ... sound clips. Unfortunately, the TEI Guidelines don't seem to have an element specifically for marking up sound clips, so we need to invent it. We'll also have to say what kind of element it is and where it is allowed to appear.

## 2 Making your own schema

1. Open the Roma application. If you are online, you can point your favourite web browser at `http://tei.oucs.ox.ac.uk/Roma/`, or, if you are working from the CD, at `http://localhost/Roma/` (on the TEI Knoppix CD the recommended web browser is Mozilla Firefox, available on the toolbar; but you can also use Konqueror or Amaya or even Lynx if you prefer).

2. The Roma start screen allows you to create a new customization, or to upload an existing customization for further work. We will start from scratch, which is the default option. Press the `Submit` button at bottom right of the screen to continue.

The next and subsequent screens show you a row of buttons for acting on your customization (`Save`, `Customize`, `New`, and `Help`), and a row of buttons for each of the major stages or tasks making up a customization (`Modules`, `Add elements`, `Change classes`, `Language`, `Schema`, and `Documentation`. We won't explore all of these in this exercise. By default the Customize your Customization screen is displayed. This allows you to specify a file name and other details for the schema, and also to change the interface language if you wish (**do not try this! the German translation is not complete yet**). For now, we will accept the defaults. Press the `Modules` button to proceed.

The modules screen shows two lists: on the left are all available TEI modules; on the right are the modules currently selected for your schema. You can add modules from the list on the left, and remove modules from the list on the right, by clicking the appropriate word next to the module you wish to operate on.

1. For this exercise, we will add the `figures` module to the default modules already selected. Click the word `add` next to "Tables, Formulae and Figures".

2. The modules chosen contain many more elements than we need, so we now need to remove some of them. Click the name of a module in the list of modules in your schema (the right-most column) to see a list of the elements this module defines. Start by clicking on the the `textstructure`, and `figures` modules (leave the others alone for now).

Each element listed has a name, a radio button indicating whether it is to be included or excluded, a tag name, a description, and a link to a further screen where its attributes are specified. You can toggle inclusion or exclusion of all elements in the list by clicking the appropriate column heading. Click on `Exclude` to remove all elements from the module.

Now work down the list clicking the `Include` button to restore the elements needed for this exercise. Remember to press the `Submit` button when you have finished with each module. Press the `Modules` or `back` links to go back to the list of modules.

**from the textstructure module** add `<back>`, `<body>`, `<front>`, `<div>`,and `<text>`

**from the figures module** add `<figure>`, and `<figDesc>`

We now need to define our new element. Click the `Add Elements` button at the top of the screen. The Defining a new element form is displayed. which allows you to enter a name for your element, to specify the classes of which it is a member, its content model, and its description. Unless you feel confident about working out what the possibilities are for yourself, we suggest that you proceed as follows:

1. name the element 'soundClip'
2. make it a member of the class 'model.global'
3. give it the content model 'TEXT'
4. supply a description such as 'references an external audio resource'

Remember to press `Submit` when you have finished (you may have to scroll the screen to the right to see the button). Now we have to add an attribute to point to a sound file:

1. click on `Change attributes`
2. add an attribute called `url`
3. and set the "Contents" to `data.pointer`
4. give a suitable description
5. Press Submit as usual

We are now ready to generate a schema. Click the *Schema* button, and then press Submit. Your browser will ask whether you want to save or open the generated file: you should save it into your home directory (the default). Look at the result, if you feel strong, or experiment with other options of the web application.

We've prepared a little test file which you can use to check that you've made your schema correctly. When you open it, however, you will have to tell emacs to use your newly made schema. Proceed as follows.

- Open the file `edison.xml` with emacs. As this is an XML file emacs will open it in NXML mode, by default.
- Choose Set Schema from the XML menu, and select `File...`, the last option on the submenu this opens.
- Supply the name and location of your schema file (`myTEI.rnc`, probably) in the minibuffer at the bottom of the screen. Press Return.
- Emacs will ask whether it should save this information to the `schemas.xml` file. Type yes, and press return.

Now check the mode line—does it say Valid? If it does, try introducing an error in the document. If it does not, click on the word Invalid, and try to see what went wrong.

## 3 Making a real file

Your next challenge is to transform the test document into one which includes a picture and a soundclip, and then to transform it to HTML for display on a website.

We have provided a couple of example objects for your use in the sample directory, called `edison.jpg` and `edison.mp3` respectively. One shows a photograph of Edison annotated by the great man himself: the story goes that this was found only slightly charred after a fire which destroyed Edison's original factory in New Jersey. The sound clip is the famous Mary had a little lamb recording, as recounted by Edison in a recording made in 1927.

With your new schema, you can now directly reference these objects in your document, using a `<figure>` and a `<soundClip>` element respectively. Here is what they might look like:

```
<p>...<figure><graphic url="edison.jpg"/>
    <head>The Great Man</head>
    <figDesc>Photograph of Edison
     annotated by himself in <date>1887</date></figDesc>
     </figure>
...
    <soundClip  url="edison.mp3">
      Edison reminisces about his first
      phonograph recording
      </soundClip>
     </p>
```

Once your document is valid, you can transform it into an HTML web page by using an XSLT stylesheet. We have prepared a suitable stylesheet for this purpose in the file `display.xsl`. We have also prepared a simple CSS stylesheet (`display.css`) which can be used to display your file without first converting it.

To see what CSS can do, add a stylesheet reference like the following:

```
<?xml-stylesheet type="text/css" href="display.css"?>
```

at the start of your file, and then try opening the XML file with Firefox.

To use XSLT, which is probably the most reliable method, you run a free-standing XSLT processor such as xsltproc and generate a static HTML page from your document. Try typing

```
xsltproc -o edison.html display.xsl edison.xml
```

at the command prompt (Tools/Shell Command in Emacs). This will generate an HTML file called `edison.html` which any web browser should be able to display.