Lou Burnard 2016-12-11

#### 1 What is this for?

This little guide is intended to simplify and clarify the task of maintaining or updating an existing TEI customization (an ODD), in particular an ODD which was written before release 3.0 of TEI P5. That release introduced several new features intended to remove the system's reliance on other schema languages. These changes are colloquially known as 'purification' because their motivation was to ensure that all aspects of a TEI specification should be encoded using TEI and nothing else (1); they affect only the way that the content model of an element or the datatype of an attribute are defined, but for completeness we have included reminders about some other customization features that you may wish to review.

## 2 It ain't broke: why fix it?

Since release 2.0 it has been possible to tie any ODD file to a particular release of the Guidelines. If you know approximately the date of the release of TEI P5 against which your ODD was first compiled, you can use the @source attribute on the <schemaSpec> element it contains to ensure that the schema generated from it continues to use the TEI definitions contained in that particular source. This has the obvious advantage that you don't have to do any work to maintain it, but some equally obvious disadvantages:

- You can't use any of the new features, corrections, or other improvements introduced in TEI P5 after the date your ODD was created
- Your TEI use risks becoming increasingly divergent from the mainstream, so that interchange with others will become increasingly difficult

Nothing in digital form is ever really finished. It's more than likely that the requirements of your project will have changed a little since your ODD was first designed. It's almost inevitable that as your project evolves, you'll have come across things you'd do differently if you could start all over again. Reviewing your ODD is a very good way of thinking about doing that.

## 3 Good things to review

Here's a little checklist of things you might want to review in the light of experience:

- Which elements are you actually using? Beginners often think that it's better to allow almost any kind of content in their schema: an extreme case of this misapprehension leads people to use TEI All for everything. It may well be that your project started out a bit uncertain about the kind of data it would have to be able to handle. But now you've encoded umpteen gazillion pages of Whatever, you surely know a bit better the kind of things that crop up. And every element you allow for in your schema is another element you need to explain to your encoders, document, find examples for, and check that it is being used consistently (if it is used at all). It's also another element that the poor benighted software developer you finally got funding for has to be prepared to handle in that swish new interface you've been promising yourself forever.
- Reducing the number of elements permitted by your schema makes it easier for you to concentrate on the quality of your documentation, for example by introducing examples more appropriate to your project than those provided by vanilla TEI (which somehow manage to be both general and very specialised).

- The same considerations apply to attributes, and in particular their range of values. At the outset you may not have been sure what values to permit for the @foo attribute on your <br/>bar> elements, so you allowed anything. Now you have discovered that some of your encoders gave this attribute the value 'centre', others used 'center', and yet others used 'middle', all meaning (probably) the same thing. Now that you know what values you want, add a <vallist> to your ODD to insist on them, and do some data cleanup.
- Has the circus moved on? You may have spent a lot of time and effort defining new elements or attributes not available in the TEI at the time, only to find that the TEI (independently or as a result of your good work) has subsequently decided to implement exactly what you did, or something like it. Maybe you can remove your old modifications and be pure TEI again. Even if the TEI decided to do things differently from you, it might be worth looking at the TEI version of your smart idea to see if it can be adapted to your needs!

And yes, it's quite likely that as a result of this review, you may well need to change all your data as well as your schema. But that's a simple matter of XSLT programming ...

### 4 Content Model Revision

In days of yore, (i.e. releases P1 to P3 inclusive) TEI Content Models were expressed in the SGML language. The first XML release TEI P4, and all releases of P5 prior to 3.0 used the ISO standard RELAX NG language for this purpose. But from the start, a design goal of the TEI was to define an encoding scheme independent of any implementation metalanguage. The TEI has always used its own 'ODD' language to define components which are then processed to produce documentation in a variety of formats (HTML, PDF, ePub etc.) and schemas in a number of different languages (RELAX NG, W3C Schema, and XML DTD). At release 3.0 this principle was extended to the definition of content models, which previously had been expressed using the RELAX NG language, but which are now expressed using some special purpose TEI elements (<sup>2</sup>)

If your customization changed the content model of any element, it will have done so using expressions in RELAX NG, typically using a <rng:ref> element to provide the name of a pattern. For example, suppose your customization defined a new element with a content model of macro.phraseSeq. In the <elementSpec> defining your new element, there will be a <content> element containing something like <rng:ref name="macro.phraseSeq"/>. To keep in step with the current TEI P5 release, you need to change this RELAX NG content to its equivalent in the ODD language which is <macroRef key="macro.phraseSeq"/>.

In the ODD language, references to different kinds of object (elements, classes, macros, etc.) in a content model are represented by different element types, specifically <elementRef>, <classRef>, <macroRef>. Hence, a RELAX NG content model in the form <rng:ref name="x"> will become <elementRef key="x"/> if x names an element, <classRef key="x"/> if it names a class, and <macroRef key="x"/> if it names a macro.

```
<rng:choice>
  <rng:ref name="bar"/>
  <rng:ref name="baz"/>
  </rng:choice>
```

Its equivalent in pure ODD would be

```
<alternate>
  <elementRef key="bar"/>
  <elementRef key="baz"/>
  </alternate>
```

If some components of your content model are optional or repeatable, you will have used the RELAX NG elements <rng:oneOrMore> or <rng:zeroOrMore>. In ODD optionality and repeatability are expressed using attributes @minOccurs and @maxOccurs, which can be supplied for any of the elements discussed so far. For example, a RELAX NG content model such as

```
<rng:oneOrMore>
  <rng:choice>
    <rng:ref name="bar"/>
    <rng:ref name="baz"/>
    </rng:choice>
</rng:oneOrMore>
```

has the following equivalent in pure ODD

```
<alternate minOccurs="1"
maxOccurs="unbounded">
  <elementRef key="bar"/>
  <elementRef key="baz"/>
  </alternate>
```

An empty element is indicated in the RELAX NG language by a special pattern called <rng:empty>. In the ODD language, however, we indicate that an element has no content by supplying an empty <content> element in its specification.

The ODD language also provides a few more special-purpose component elements for content models: <textNode>, , and <anyElement>.

The <textNode> element is provided as a replacement for the built-in RELAX NG pattern <rng:text>. There are no restrictions on where it may be placed in an ODD content model, although existing schema languages mostly permit it to appear only in mixed content models like the following:

```
<alternate minOccurs="0"
maxOccurs="unbounded">
  <textNode/>
  <elementRef key="foo"/>
  </alternate>
```

The RELAX NG language allows a number of other components within a content model, some of which are difficult to convert, but few of which are likely to appear in a pre-existing TEI ODD. If your ODD used the RELAX NG element <rng:value> to specify content explicitly, this must be expressed in an ODD as a <vallitem> within a <vallist>. If your ODD used the RELAX NG element <rng:element> to specify that any element is permitted at this point in a content model, you can do something similar with the ODD <anyElement> element. In general, however, the presence of any of the following in your old content model will require manual intervention: <rng:anyName> <rng:attribute> <rng:data> <rng:element>

Here are some example **<content>** elements taken from the content models of existing TEI elements:

#### RELAX NG Specification ODD equivalent

```
<content>
                                        <content>
                                          <alternate minOccurs="0"
 <rng:zeroOrMore>
                                          max0ccurs="unbounded">
  <rng:choice>
   <rng:text/>
                                           <textNode/>
   <rng:ref name="model.limitedPhrase</pre>
                                           <classRef key="model.limitedPhrase"/>
                                          <classRef key="model.global"/>
   <rng:ref name="model.global"/>
  </rng:choice>
                                         </alternate>
                                        </content>
 </rng:zeroOrMore>
</content>
```

```
<content>
                                        <content>
 <rng:group>
                                          <sequence>
  <rng:zeroOrMore>
                                           <elementRef key="analytic"</pre>
   <rng:ref name="analytic"/>
                                            minOccurs="0"
                                            max0ccurs="unbounded"/>
  </rng:zeroOrMore>
  <rng:oneOrMore>
                                           <sequence min0ccurs="1"</pre>
   <rng:ref name="monogr"/>
                                            max0ccurs="unbounded">
                                            <elementRef key="monogr"/>
   <rng:zeroOrMore>
                                            <elementRef key="series"</pre>
    <rng:ref name="series"/>
   </rng:zeroOrMore>
                                             minOccurs="0"
  </rng:oneOrMore>
                                             max0ccurs="unbounded"/>
  <rng:zeroOrMore>
                                           </sequence>
   <rng:choice>
                                           <alternate min0ccurs="0"
    <rng:ref name="model.noteLike"/>
                                            max0ccurs="unbounded">
                                            <classRef key="model.noteLike"/>
    <rng:ref name="model.ptrLike"/>
                                            <classRef key="model.ptrLike"/>
    <rng:ref name="relatedItem"/>
    <rng:ref name="citedRange"/>
                                            <elementRef key="relatedItem"/>
                                            <elementRef key="citedRange"/>
   </rng:choice>
                                           </alternate>
  </rng:zeroOrMore>
 </rng:group>
                                          </sequence>
</content>
                                         </content>
```

# 5 Attribute values and datatypes

As noted above, you may wish to change your existing ODD to be more precise in the way that attribute values are specified. If you decide to introduce a <valList> (semi-closed or closed) to constrain the possible values of an attribute you will also need to change its <datatype> to reference teidata.enumerated.

If the <datatype> element appears in your ODD already, you will need to change it to use a <dataRef> element pointing to one of the predefined teidata datatype specifications. A set

of predefined data specifications using RELAX NG has been retained for compatibility reasons, but these will probably be withdrawn by the end of 2017.

Proceed as follows:

- if your existing <datatype> contains something like <rng:ref name="data.xxxx"/>, change it to <dataRef key="teidata.xxxx"/>
- if your existing <datatype> contains something like <rng:data type="xxxx"/>,
   change it to <dataRef name="xxxx"/>
- if a component of your existing <datatype> contains something like <rng:param name='pattern'/> you will need to either
  - add an attribute @restriction to the <datatype>, giving as value the content of the <rng:param> element
  - or (if the parameter cannot be expressed as a regular expression, and if the datatype uses the @name attribute) re-express the parameter as a TEI <dataFacet>
- If the datatype being defined permits multiple values, alternation, or anything else other than a single value, you must define it in a <dataSpec> element, using the same components as an element content model (<alternate>, @maxOccurs, etc.) Note that although the <datatype> element permits only a single <dataRef> child, it is itself repeatable, so that you can say (for example) 'this attribute takes at least three pointer values'.

Here are some example **<content>** elements taken from the data specifications used by some existing TEI datatypes:

<content></content>	RELAX NG Specification	ODD equivalent
<pre><rng:data type="NCName"></rng:data></pre>		
<pre><rng:data type="NCName"></rng:data></pre>		
	<rng:data type="NCName"></rng:data>	<pre><dataref name="NCName"></dataref></pre>

```
<content>
  <rng:data type="nonNegativeInteger
  <rng:param name="maxInclusive">1(
  </rng:data>
  </content>
  </content>
  </content>
  </content>
  </content>
```

```
<content>
                                      <content>
 <rng:choice>
                                       <alternate>
  <rng:data type="float">
                                        <dataRef name="float">
   <rng:param name="minExclusive">
                                          <dataFacet name="minExclusive"</pre>
                                           value="0"/>
   <rng:param name="maxExclusive">
  </rng:data>
                                          <dataFacet name="maxExclusive"</pre>
  <rng:value>regular</rng:value>
                                           value="1"/>
  <rng:value>irregular</rng:value>
                                        </dataRef>
  <rng:value>unknown</rng:value>
                                        <valList>
                                          <valItem ident="regular"/>
 </rng:choice>
                                          <valItem ident="irregular"/>
</content>
                                          <valItem ident="unknown"/>
                                        </vallist>
                                       </alternate>
                                      </content>
```

```
<content>
                                     <content>
 <choice>
                                      <valList type="closed">
  <rng:value>high</rng:value>
                                       <valItem ident="high"/>
  <rng:value>medium</rng:value>
                                       <valItem ident="medium"/>
  <rng:value>low</rng:value>
                                       <valItem ident="low"/>
  <rng:value>unknown</rng:value>
                                        <valItem ident="unknown"/>
 </choice>
                                      </vallist>
</content>
                                     </content>
```

# 6 My head hurts: where's the script?

Yes, there is a script or two you can experiment with if you find this all too daunting. They were used to transform the whole of the pre-3.0 TEI Guideline specifications semi-automatically. They deal with 99% of the situations you are likely to encounter, but they're not guaranteed! You can find them on Github at https://github.com/TEIC/TEI/tree/dev/P5/Scripts: use them in good health, but at your own risk.

Don't hesitate to share your experience, good or bad, on the TEI discussion list, by raising a ticket on Github, or by contributing to the TEI wiki. The TEI is a community effort!

#### Notes

 $<sup>^{1}</sup>$ see further Resolving the Durand Conundrum

<sup>&</sup>lt;sup>2</sup>The authoritative source of information on these is of course chapter 22 of the Guidelines *Documentation Elements*. In any disagreement between what is stated there and what is suggested here, the former is correct.