XML schemas

Ariane Pinche

Warsaw, May 20–24, 2019.

1. TEI project

- TEI all is like a framework that help editors in there’s structuration of the information.
- It consists of a set of recommendations that must be adapted to the specificities inherent to each editorial project.
- **But** how to find a way to ensure **homogeneity** and **sustainability**.

Solutions ?

–Project documentation;

–Creating validation rules that restrict the possibilities of the encoding. It involves making choices between several admissible solutions for the following questions:

- Why edit?
- For whom ?
- What is the size of the corpus?
- In what form will the text be edited?
- What will be the use of data?
- How long and what resources?
- What will be the workflow of the project?
- **Where to stop?**

2. How to regulate its encoding, good pratices ?

- TEI-compliant XML documents must be:
  - well formed
  - valid
- TEI-conformant encoding, what does it mean ?
  - XML need to be well formed;
  - The proposed encoding must be validated with a TEI_all schema;
  - The encoding must conform to the TEI abstract model;
  - Encoding must make good use of the TEI namespace (and other namespaces if needed);
  - Encoding must be documented.

3. XML schemas

3.1 The different types of schemas

- DTD;
- Relax NG;
- Schematron;
- XML schema;
- ODD ‘One Document Does it all’;
    - [TEIguidelines] (http://www.tei-c.org/release/doc/tei-p5-doc/en/html/USE.html);
    - [ODD documentation] (http://www.tei-c.org/guidelines/customization/getting-started-with-p5-odds/)

3.2 How to make an ODD ?

- From scratch, which can be very time-consuming;
- From *Roma* by selecting its own modules: https://roma2.tei-c.org
- From an XML file in Oxygen, with the scenario ODD by example.

3.3 Roma manipulations

try it yourself

- TEI modules
- TEI elements
- TEI attributes

4. Documentation

The ODD includes:

- a TEI root element;
- a teiHeader;
- a text element;

The body element can contain divs with level (div1, div2, etc.)
We can structure its documentation in a first div1 and place its specifications in another div1.

4.1 Documentation tags

- `<att>` (attribute) contains the name of an attribute that appears in the current of the text.
- `<gi>` (generic identifier) contains the name of an element.
- `<tag>` (tag) the contents of an opening or closing tag, possibly with attribute specifications, but excluding the characters marking the opening and closing of the tag.
- `<val>` (value) contains only one attribute value.

Example

```
<p>The corpus presents three scenarios.
In the first case, the original punctuation
is removed in the standard edition.
Adding the <att>type</ att> attribute
with value <val>orig</val> on the <gi>pc</gi>
element indicates that the sign comes from the punctuation
of the manuscript and
that it must not  appear in the standard version.
<egXML xmlns=" http://www.tei-c.org/ns/Examples">
car <lb/>adonc estoit costume en
<placeName ref="#france"><choice>
  <orig>f</orig>
  <reg>F</reg>
</choice>rance</placeName><pc type=" orig">.</pc>
qe li vilein <lb/>
de la contree prenoient les ymages de lor <lb/>deables
</egXML>
</p>
```

5. Advanced personalisation with ODD

5.1 oddbyexample tutorial

- Open your XML file
- Configure a transformation scenario: Document / Transformation / Configure transformation scenario
- Create a new scenario:
    - XML transformation with XSLT;
    - Inform the path of the XSL ${frameworks}/tei/XML/tei/stylesheet/tools/oddbyexample.xsl;
    - Select Saxon 9.xX processor
    - Advanced options, *template (-it): main*;

- Parameters: corpus $ {cfdu} (i.e. current directory)
- Configure Output (Output tab): Set a name and location for the future ODD.
- Save your configuration
- Apply to your XML file

5.2 customisation

there are 4 main manipulations:

- Delete elements;
- Customise elements;
- Customise the attributes and attribute values of an element.
- Add elements;

5.2.1 Deleting an element

- Simple way

```xml
<elementSpec ident="rdgGrp" mode="delete"/>
```

- Second way

```xml
<moduleRef key="textcrit" except="rdgGrp"/>
```

- Third way

```xml
<moduleRef key="textcrit" include="app lem rdg"/>
```

5.2.2 Customise an element

5.2.2.1 Changing description

```
<elementSpec ident=" lem" mode=" change">
 <gloss>Lemme</gloss>
 <desc>permet de signaler
   la leçon choisie dans le texte édité</desc>
 <attList>
   <attDef ident=" type" mode=" change">
         <valItem ident="viz"/>
      </valList>
  </attDef>
 </attList>
</elementSpec>
```

5.2.2.2 Ruling a sequence of elements

- Use element `<content>` and `<sequence>` and customise your parameters with attributes:
    - *@minOccurs* Indicates the smallest number of times this component may occur.
    - *@maxOccurs* Indicates the largest number of times this component may occur.
    - *@PreserveOrder* if the value is *true*, indicates that the elements order must correspond to the content model order

Example

```xml
<elementSpec ident="div1" mode="change">
<content>
<sequence preserveOrder="true">
<elementRef key="head" minOccurs="1" maxOccurs="1"/>
<elementRef key="div2" minOccurs="0" maxOccurs="unbounded"/>
</sequence>
</content>
</elementSpec>
```

If you want to add text in the element use the element `<textNode/>`

You can also use the element `<alternate>`.

It indicates that the construct referenced by its children is an alternation

For example, in the element `<choice>`:

```
<content>
 <alternate>
  <sequence>
   <elementRef key="sic"/>
   <elementRef key="corr"/>
  </sequence>
  <sequence>
   <elementRef key="orig"/>
   <elementRef key="reg"/>
  </sequence>
  <sequence>
   <elementRef key="abbr"/>
   <elementRef key="expan"/>
  </sequence>
 </alternate>
</content>
```

5.2.3 Customising attributes and attribute values of an element.

5.2.3.1 Modifying attributes on an element

```xml
<elementSpec ident="div" mode="change">
<attList>
<attDef ident="part" mode="delete"/>
<attDef ident="type" mode="change">
  <valList mode="add" type="closed">
    <valItem ident="section"/>
    <valItem ident="chapter"/>
</valList>
</attDef>
</attList>
</elementSpec>
```

5.2.3.2 Mandatory attributes

```
<elementSpec ident="div" mode="change">
  <attList>
    <attDef ident="part" mode="delete"/>
    <attDef ident="n" mode="change" usage="req"/>
  </attList>
</elementSpec>
```