Uncovering the TEI and ODD

A pedagogical strip-tease

Laurent Romary - Max Planck Digital Library

Objectives

- Using Roma as a pedagogical tool
  - A user oriented approach
  - A coherent picture of the TEI
- Understanding the core principles of the TEI
  - Modules, classes, and main elements
- Getting acquainted to ODD
  - Showing why ODD is an essential part in the TEI architecture
- Providing an intuitive view about *conformance*

Prerequisites

- What the learner knows
  - Principles of tagging (cf. HTML)
  - XML (may have already drafted his own DTD for encoding his stuff)
  - What he wants to do

    Some ideas about the data he would like to encode

    …and the kind of information he would like to see in his "schema"
- What he may not necessarily know
  - Principles of data interchange and/or long-term archiving
  - Hence, what *standardization* may mean…

The TEI spirit

- Conformance?
  - Sharing a common text encoding culture
  - Sharing the same vocabulary (when applicable)
  - Allowing user autonomy in defining modifications (extensions, customization), but sharing the mechanisms to do so

Main concepts

- Literary programming
  - One Document Does it all
    - Schema specification
    - User oriented documentation
- The founding principles of ODD
  - Modularity: all specifications pertaining to a coherent sub-domain of the TEI
  - Classes: identifying shared behaviours or semantics
- Extensibility: a consequence of the above mechanisms

Specifying a schema

- What is a schema?
  - Elements, attributes and constraints between these
- Which schema language should be used?
  - Agnosticism of the TEI
    - DTD, Relaxng, W3C
  - Still, RelaxNG fragments in ODD

ODD and the TEI

- ODD for the TEI
  - The TEI is defined as an ODD specification
    - Main application: Roma
- ODD is part of the TEI
  - The ODD language contains a set of additional elements dedicated to schema specification
    - TD - Documentation elements
- ODD beyond the TEI
  - ODD can be used by any community to specify their own schema

A journey in five steps

1. *Easy TEI*
   - *Simple access to the TEI through Roma*
2. *Subsetting the TEI*
   - *Making the TEI even easier to use*
3. *Enlarging the application profile*
   - *Using modules*
4. *Modifying the TEI objects*
   - *First insights into extensibility*
5. *Behind the scene - ODD*
   - *Starting to use the actual specification language*

Step 1

Easy TEI

The scenario

- Quick and simple access to the TEI environment
  - I have seen my colleague next door doing some encoding with the TEI and I want to do the same at once
    - I have just downloaded Oxygen…

Main action points

- Go to Roma
  - http://tei.oucs.ox.ac.uk/Roma
- Toy with user profile [*Customize*]
- Overview of interface tabs
  - Without going into them proper
- Generate Schema [Schema]
- Make a try on Oxygen
  - Create a simple document
  - Get back to Roma to have a basic documentation
    - E.g. the never ending <fileDesc>

Discussion points

- But that's already too much!
  - The TEI as a market place (LB="framework")
    - See Step 2
- The things I actually want are missing!
  - Are you sure?
  - Identify what's missing
    - Already defined components in the TEI structure - See Step 3 ("Modules")
    - Missing objects - See Step 4 ("Subsetting")
    - Uncovered textual genres (*Cuneiform tablets encoding*) - See Step 5 ("Odd as a specification language")

Discussion points (cont.)

- Doing this does not necessarily means *conformance*
  - Schemas (a fortiori DTDs) only provide a syntactic validation of your document
  - Understanding the meaning of tags
    - Cf. tag misuse or abuse

      <emph>a priori</emph>
  - Several solutions for the same problem
    - Numbered vs. unnumbered s

Lessons to be learnt

- The TEI is XML-based
- Getting started with the TEI is easy
- A full documentation comes with it

Step 2

Subsetting the TEI

The scenario

- Defining a project oriented TEI Lite
  - I just want a set of useful tags to toy with my text
- Understanding the basics of the TEI architecture
  - Modules
  - Main organisation of a TEI text
  - Getting things documented

Main action points

- Go to Roma…
  - Look at [Modules]
    - Default modules + additional modules
    - Explore default modules by pointing to main elements (by order of pedagogical interest)
      - textstructure: TEI - text - body - div
      - core: p - q - list - pb - head
      - header
  - Start checking out elements
    - Preserving the skeleton (TEI/text/body)
    - Making editorial choices (numbered vs. unnumbered div’s)

Main action points (cont.)

- Check the result on XML
  - Look at the resulting schema
  - Start editing a document
- Back to Roma
  - Generate a documentation
    - Customizing everything at once!

Discussion points

- Introducing the concept of production vs. Exploration schema
  - Limit the elements to those that the encoder may use
  - But how can I reuse what I have just done on Roma?
- Conformance (cont.): First risk of divergence across applications
  - Two application profiles are not necessarily compatible with one another
  - Still, they share the same semantics for the same element

Lessons to be learnt

- The TEI is not a monolithic environment
  - Very few things are really mandatory
- The TEI is more than just a market place
  - Basic document structure must be preserved
- The TEI is based on a powerful environment for working with elements and producing a documentation

Step 3

Enlarging the application profile

The scenario

- Completing the default module selection
  - It was nice to encode my text, but I would like to create a dictionary…
- Understanding the magic behind the scene
  - Classes: to connect modules to one another
  - ODD: to save a user's profile

Main action points

- Go to Roma…
  - [Modules] Add dictionary modules
- Test the result on Oxygen
  - Create a first dictionary entry
- Back to Roma
  - [Save customization]
  - Look at resulting file
    - First ODD declarations
  - Reload the file under Roma
    - Carry on work…

Discussion points

- But the structure I get is not exactly what I want
  - See next Step…
- Conformance (cont.)
  - Not all TEI application use the same set of modules
  - Extra subsetting is still possible (¬<entryFree>)

Lessons to be learnt

- The TEI offers a very easy way to build a complex application
- The TEI is based on a specification language - ODD

Step 4

Modifying the TEI objects

The scenario

- Tuning a TEI profile
  - I want to describe my own view on what a dictionary entry is
    - Adding an element <caseFrame>
    - Restricting the content model of <gen> (m + f)
- Getting acquainted with classes

Main action points

- Go to Roma
  - Start with a default TEI + dictionary structure
  - Explain what classes are (see next slides)
  - observe <gen>
    - Classes and content model
  - [Add elements] create <caseFrame>
- Test result on Oxygen
- [Save customization] Look at resulting file

Main action points (cont.)

- Go to Roma
  - Change content model of <gen> with

```
<valList type="closed">
  <valItem ident="m"/>
  <valItem ident="f"/>
</valList>
```

- Check result on Oxygen
- [Save customization] Look at resulting file

Classes in the TEI

- Groups together all elements with the same role in the TEI architecture
  - Same syntactic behaviour
    - The elements in the class will appear in the same content models
  - Semantic similarity
    - The class defines a group of elements belonging to the same family of concepts
- Principle:
  - elements declare themselves as belonging to a class

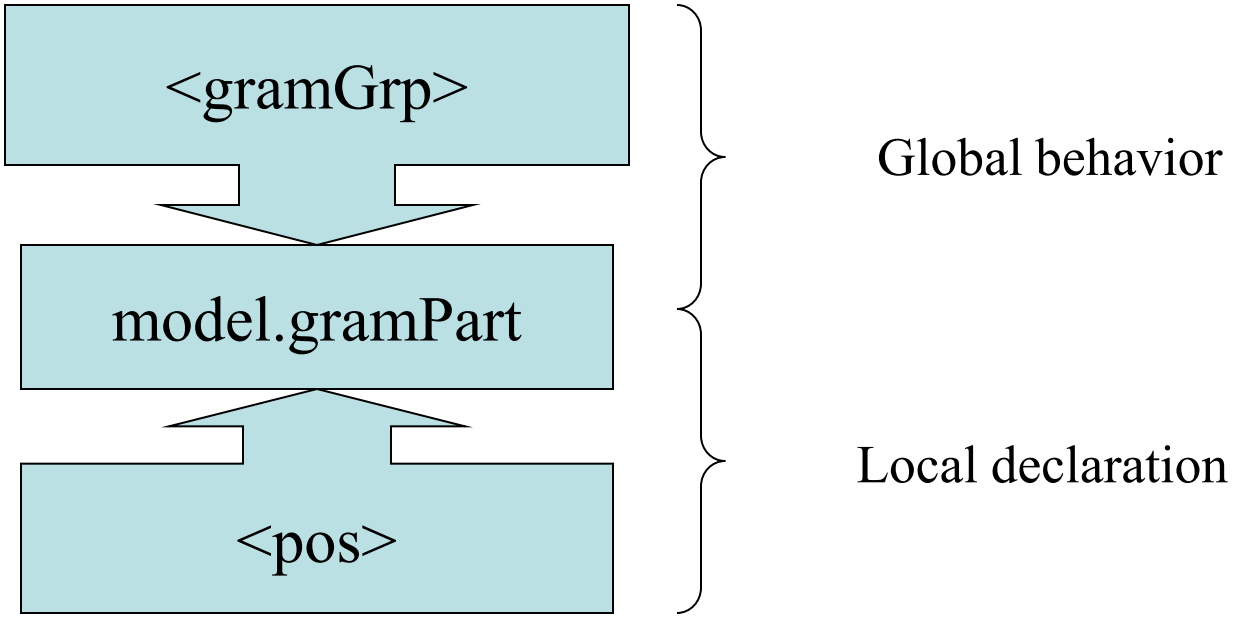
A simple example: model.gramPart

- Grammatical information in a dictionary entry
  - E.g.:

```
<entry>
        <form>
                <orth>luire</orth>
        </form>
        <gramGrp>
                <pos>verb</pos>
                <subc>intransitive</subc>
        </gramGrp>
</entry>
```

  - Rather homogeneous set of elements
    - <pos>, <gen>, <number>, <case>, etc.
  - May also appear in <form>

The decoupling principle



Discussion points

- How far can I go in making change to the TEI
  - See next step
- Conformance
  - How can I be conformant to the TEI when I actually change the specifications?
    - Adding objects and documenting them
    - Subsetting values

Lessons to be learnt

- The TEI allows you to tune an application profile and generate its documentation

Step 5

Behind the scene — ODD

The scenario

- Going beyond Roma
  - Should I really go through the interface to create my modifications
- Editing ODD files

Main action points

- Take a previous configuration file (step 4)
- Toy with the various possibilities under Oxygen
  - Delete elements manually
  - Restrict the value of another element
- Go back to Roma
  - Present the documentation elements module
  - Create an application profile with TD
- Validate (!) the configuration file

Discussion points

- What is a schema
  - RelagNG vs. ODD specification
- Can I do more than that
  - Describing a schema from scratch

- Conformance
  - Am I still conformant I just do not care about existing TEI elements?

Lesson to be learnt

- Feeling the urge to know and do more…

Conclusion

- Roma contains all the necessary entry points into the TEI principles

- What remains to be done
  - More homogeneous documentation with P5
  - Graphical presentation of element dependences
  - Better integration of RelaxNg