Ending the myth of presentation in digital editions

Magdalena Turska, Sebastian Rahtz and James Cummings

September 2014 @magdatureska @jamescummings @rahtz
The point

What ensures the best chance of long-term preservation and reuse is the concentration on openly licensed, quality encoding conforming as best as possible to standards.

To achieve this it is necessary to lower the barriers in transition from the annotated source to published edition.
Can we keep our promises?

- scholarly use
- and RE-use
- interchange
- long-term preservation
Available != reusable
...but how is it going to look?

... BUT HOW IS IT GOING TO LOOK LIKE ???

- creators: prevalent textual editing background; perceive the encoding only as the necessary (and unavoidable) step towards the real goal - the published edition
- developers: the encoding is the most important and resource hungry; the rest is almost trivial
- creators: How is it going to look like?
- developers: You tell me!
...and how to make it look like this?

„How do I encode this to make it look so-and-so?“
Ultimately honesty is the most important quality of the editor
Where's the lasting value?

- data modelling: a schema and encoding policy
- annotation
- presentation
- research, re-use
- long-term preservation of knowledge made explicit in the encoding

Presentation by its very nature is ephemeral: bound to be modified over time or multiplied for other use scenarios
The long haul

Computer stuff gets obsolete pretty quickly: a decade or two or maybe 2 years if you're unlucky
Need to refurbish the interfaces, migrate the underlying software and perhaps data as well is one constant in long-term preservation
It’s the data source, the encoding that is of long-lasting value and not the presentation
Good, formal, well documented data model will lend itself to fairly easy conversion into any new solution that comes
How to standardize individual research?

- will to follow project’s own path is always strong
- counterbalanced by the urge for standardization and limits on the available resources

TEI is particularly successful, perhaps because it tries to marry both. Yet TEI's weak constraints and customizability mean processing and publishing of TEI-encoded files is mostly left to the users. For now we are without the tools that are as flexible and customizable and user-friendly as the encoding vocabulary.
Andrews (2012) points out: ‘consensus is indeed lacking on what exactly a digital critical edition should be. As long as there is no agreement on the end result of digital philology, there can be none on its methods; as long as there is no consensus on method, there will not be widely applicable computational tools available to help produce digital critical texts.’
It's still hard to get from point A to point B (even if your mathematician says it's trivial)

Only when the editors are as familiar with processing and publication tools as they are with text processors they can start to care more about fidelity of encoding. Is solution really so simple and we just need more tools?
Final solution is a myth

However good the infrastructure it will never solve all the challenges of individual research.
Still, even innovative projects probably could fit 80% within the ‘standard’ approach and perhaps only the last few steps need some special treatment during both encoding and processing phases.
Going off-road? Take your army knife

Dealing with the unforeseen might seem to require magic wand, but what if similar powers could be gained from rudimentary understanding of 4 not-so-foreign languages?
Why?

- XML vocabularies (e.g. TEI) are the language we use: we should take responsibility and make informed decisions about representation of the source material;
- general XML concepts of well-formedness, validity and schemas also a must
- XPath: finding answers to research questions, validating working hypotheses or just checking the consistency of our encoding
- CSS: for expressing precisely the rendition we record or the effect we want to achieve
- HTML: just because there's 95% chance we will output it at some point

No one says it's easy, but it has immediate rewards.
Perils

- over-reliance on imperfect graphic tools
- tag abuse
- output-driven encoding
- inadequate encoding based on what is possible with the tools
- decision not made independently but delegated to external authority or tech support
- introduction of delays and distortions due to imperfect communication
- increase of the costs
Community projects

What is yet to happen is for a universally usable and customizable system to emerge that is user-friendly, easy to run and customize as needed. Before it happens, there needs to be some compromise on what are the goals, the recommended workflows and also what the acceptable levels of complexity and steepness of learning curve are.

TAPAS, DTA, DARIAH, TextGrid are some of the projects and networks crossing the institutional and national boundaries that aim to create a lasting infrastructure for publication, research and long-term preservation.
Meanwhile in TEI...

The ambivalence continues

- every project is different and we delight in flexibility and customization
- projects are not that different and we aim at standardization & interoperability

‘(...) documents worth encoding in TEI are very different from customer letters. But not that different, and eight out of ten probably will benefit from staying within the confines of a well thought-out standard schema and its surrounding processing rules. And even the two that don’t may benefit from staying within that standard schema as far as possible. ’ - M. Mueller
The TEI Simple project aims to produce a complete 'out of the box' customization suited to the representation of early modern and modern books which meets the needs of the many users for whom the task of creating a customization is daunting or seems irrelevant.
What ‘Simple’ tries to achieve

- remove the ambiguity for the encoder
- provide default processing scenario
- define formal means of expressing intended outputs and software to transform it into usable XSLT stylesheets
A Simple Processing Model (SPM)

It might be possible to express a broad range of processing scenarios in the ODD

- **What it should do**: create an extension of ODD metalanguage, adding new elements into TEI to prescribe the intended processings for every TEI SIMPLE element in various circumstances using just a few simple attributes
  - create an automated transformation generator - a software suite to automatically process the ODD files creating customized XSLT stylesheets
- **What it won't do**: TEI Simple processing does not aim to substitute the XSLT language and cannot be expected to reach exactly the same level of flexibility.

It does promise to make life easier for those who want to travel that path as far as it will take you, which for quite a few projects will be far enough.
Multitude of outputs

- Processing of any given element may vary significantly depending on a type of output.
- A note element: footnote for print and a pop-up for web
- The same element may need to be processed more than once for the same output.
- A head elements containing chapter titles: processed to be displayed at the chapter start and once more to create table of contents; following different algorithms and producing different visual outputs each time

Conclusion: we need to define multiple processing entries for any element.
How to determine which processing rules apply?

Two factors:

- the output mode
- the contextual constraints (when processing should differ depending on the context in which element is found e.g. different rule for heads on chapter level and different for the heading on a list)
And the actual processing

- **Idea**: allow to select the contextual constraints and the data to apply a TEI Simple function on to tackle typical tasks
- Functions are based on relatively abstract but commonly understandable ideas: make a list, note, new line, apply some particular formatting in-line
- Functions are customizable through parameters that allow to select the data to work on and the formatting class to use for the output

Hopefully this gives the editors enough power and ease of expression to let them concentrate on the encoding.
Working syntax - ODD extension

The `<process>` instruction available for `<elementSpec>` will define a way of processing this element.

Multiple processing instructions may occur to define expected behaviour in various contexts or output formats.
Working syntax - `<process>` element

- **context**: XPath expression defining a context in which this processing instruction is applicable.
- **name**: Name of the function from TEI Simple function library to be applied; input content for the function supplied as function parameter.
- **mode**: Output mode for which this processing instruction is applicable.
- **class**: CSS class or simple:classname name of formatting instruction to be applied to the output.
New ODD - <choice>

<elementSpec ident="choice">
  <process context="ancestor::front and corr and sic"
    name="makeInline(corr)" mode="render"/>
  <process context="corr and sic"
    name="makeMarginalNote(corr)" mode="render"/>
  <process context="corr and sic"
    name="makeInline(corr)" mode="textextract"/>
</elementSpec>
New ODD - `<speaker>` and `<name>`

```xml
<elementSpec ident="speaker">
  <process name="makeInline(.)"
           mode="render"/>
</elementSpec>

<elementSpec ident="name">
  <process name="makeInline(.)"/>
  <process name="makeMarginalNote(.)"/>
</elementSpec>
```
Multiple outputs - <app>

<elementSpec ident="app">
<process context="not(ancestor::app) and (lem)"
  name="makeMarginalNote(.)" mode="render" class="note"/>
<process name="makeInline(lem)"
  mode="render"/>
<process name="makeInline(lem)"
  mode="textextract"/>
</elementSpec>
<lb> and <fw>

```
<elementSpec ident="lb">
  <process name="makeNewline()"
    mode="diplomatic"/>

<!-- no specific process instruction for default "render"
mode means to output textual content (in this case: no output)-->  
</elementSpec>

<elementSpec ident="fw">
  <process name="omit()"
    mode="render"/>
  <process name="makeMarginalNote(.)"
    mode="diplomatic"/>

<!-- no specific process instruction for default "render"
mode means to output textual content (in this case: no output)-->  
</elementSpec>
```
Multiple outputs - <titlePage>

<elementSpec ident="titlePage">
  <process name="makeBlock(.)"
    mode="render"/>
  <process name="omit()"
    mode="textextract"/>
</elementSpec>
Prototype run

Let's apply this approach to a simple thing like Treasure Island by R. L. Stevenson

- take TEI encoded Treasure Island from OTA and run teibyexample on it to create ODD
- add new TEI Simple style `<process>` instructions dealing with headings, paragraphs etc
- run simpleoddtoxsl.xsl that outputs XSLT file
- transform original treasure Island TEI XML with generated file
- cherish the fact that it looks more or less like OTA HTML version
- tweak the ODD and run the process again to make it different without touching any XSLT
Is it simple? Is it powerful enough?

- Does this building blocks approach have enough appeal?
- Will it be powerful enough to really deal with 80% of printed material?
- Won't the requirements, especially the necessary understanding of XPath and CSS be too much of an obstacle?
- Should we even ask the editors to consider acquiring the minimal technical skill set proposed?
Few requirements for SPM architecture

SPM architecture should consist of tools for editing new ODD files based on project’s regular ODD customization and XSLT stylesheets suite for automated generation of (possibly multiple) transform stylesheets based on processing definitions from new ODD

- library of core functions dealing with possible* scenarios (eg. makeInline(), makeNote(), etc.)
- room for extensions and customizations of the output stylesheets as well as core functions library
- library of core CSS stylesheet (skins) as well as inclusion of bespoke CSSs
- graphic interface (Roma-like) for interactive creation of the new ODDs

*Possible within the scope of TEI Simple, eg. something that may occur in early-modern and modern printed book.
What if TEI Simple takes off?

- long-term positives in case it’s successful:
- lowering of the barriers for the starting/unfunded/no-background projects
- better project documentation
- better long-term preservation perspectives greater flexibility
- community-created tool library potential