**Abstract**

Stand-off markup is widely considered as a possible solution for overcoming the limitation of inline XML markup, primarily dealing with multiple overlapping hierarchies. Considering previous contributions on the subject and implementations of stand-off markup, we propose a new TEI-based model for encoding, that still uses the regular TEI elements, but in a stand-off manner. Our light notation moves the bulk of markup into a separate `<standoff>` element, grouping layers of related textual features encoded via existing TEI elements (e.g. `<name>` or `<corr>`) into individual `<stf>` elements; furthermore, our proposed notation provides a schema for referencing the transcription using the `xml:id` attribute. This approach is illustrated through a variety of examples. Our proof-of-concept transformation package works directly on the stand-off markup, without the necessity of reducing it back to inline TEI for parsing, querying and visualizing.

0. Introduction. Stand-off markup.

"Markup is said to be stand-off, or external, when the markup data is placed outside of the text it is meant to tag".2 One of the most widely recognized limitations of inline XML markup is its inability to cope with multiple overlapping hierarchies. However, as Bański points out, overlap and discontinuity are inherent in our theoretical constructs of texts, thus we need to embrace such constructs rather than attempt to outwit them. Stand-off markup has been considered a possible solution to this problem.

Moreover, on a theoretical level inline markup embeds disparate interpretations of the text into an already interpretative transcription; stand-off markup on the other hand clearly separates these layers of interpretation. On a very practical level, stand-off reduces distractions and ambiguity for the encoder, for example avoiding the necessity to chose a wrapper and an inner tag in the case of a name with a correction `<corr><persName>Charles</persName></corr>` versus
It also facilitates further enrichment of existing digital texts thanks to a modular, decentralized method of introducing and storing the markup.

The notation proposed here is to be considered as a point of departure for further developments. It moves the bulk of markup into a separate `<standoff>` element, grouping "layers" of related textual features encoded via existing TEI elements (e.g. `<name>` or `<corr>`), into individual `<stf>` elements, and proposes a schema for addressing fragments of the transcription using `@xml:id`'s.

In this paper we will outline our proposal first, providing examples and discussing key issues such as the granularity of the text and the division into layers. Secondly, we will present a proof-of-concept transformation package for conversion from inline to stand-off markup and for visualization of stand-off layers, together with some examples of direct queries on the stand-off document.


While several successful attempts have already been made to overcome the problem of overlap, either by introducing stand-off markup in XML (Berrie; Ide and Suderman; Stührenbergs) or by using other, non-XML markup schemes (Ide and Suderman; Piez), our intention is to concentrate on a TEI-based solution. Though, within the wider TEI ecosystem there exist isolated projects applying stand-off markup for particular needs (e.g. Cummings, Folger Digital Texts), little attempt has been made to model stand-off markup more generally. Nevertheless, a recent paper (Pose, Lopez and Romary) has informed our work; it is worth recalling and testing here the ideas and the model presented in that contribution.

In A Generic Formalism for Encoding Stand-off annotations the authors introduce three new TEI elements: `<standoff>`, which contains all the stand-off annotations; `<stf>`, containing one stand-off annotation; `<stfGrp>`, for a group of stand-off annotations, i.e. a group of `<stf>`. Each stand-off annotation `<stf>` consists of the content of the annotation and a reference to the portion of the text being annotated: the references are encoded using pointing mechanisms such as `<ptr>`, `<link>` and `<linkGrp>`, while the content of the annotations is encoded using `<label>` or `<fs>`. The complexity of this model is especially visible in the latter. The powerful Feature Structures module of the TEI Guidelines is generally used by encoders that need to annotate in a structured way features not possible to cover using existing TEI elements (cf. Boot; Stegmann and Witt) and the linguistic community represents a significant part of the users of this module. A feature is first defined in the `<encodingDesc>` section of the `<teiHeader>` and then used throughout the `<text>`.

The example (Example 1) demonstrates how the Pose-Lopez-Romary model might be implemented.

The strengths of this approach are the possibility of encoding features not covered by the TEI tagset and, most of all, the validation mechanism provided by the `<fsdDecl>` declaration. Nevertheless, an attempt to implement the module has revealed that using Feature
Structures is valuable for declaring new features, but mapping the complete set of TEI elements and attributes to Feature Structures could be a painstakingly laborious exercise. The example above, the sole purpose of which is to define the TEI element `<said>`, illustrates that the framework of Feature Structures requires the explicit re-declarations of the elements that are already a part of the TEI vocabulary. This obscures or even cuts the link between declared feature structures and corresponding elements from TEI schema. Therefore encoders cannot rely on their understanding of TEI vocabulary while marking up the text. Furthermore, the stand-off markup produced bears little resemblance to standard TEI.

2. A new model

The model that we present here makes use of the `<standoff>` and `<stf>` elements proposed by Pose-Lopez-Romary. Into that model, we introduce a pointing mechanism and a straightforward way to encode the content of the annotation making use of existing TEI tags like `<corr>` or `<persName>`.

The TEI document with stand-off markup is composed of the root `<TEI>` element, which contains three compulsory elements: `<teiHeader>`, `<standoff>` and `<text>` (see Example 2).

Declaration of the element TEI

```plaintext
element TEI
{
  att.global.attributes,
  att.global.rendition.attributes,
  att.global.linking.attributes,
  att.global.analytic.attributes,
  att.global.facs.attributes,
  att.global.change.attributes,
  att.global.responsibility.attributes,
  attribute version { data.version }?,
  (teiHeader, standoff, text)
}
```

The element `<standoff>` contains all the stand-off annotations, grouped into `<stf>` elements. Each `<stf>` element contains the stand-off annotations that belong to a single layer (see below) and it may also carry metadata about the annotation (author, date, type, etc.).

The content of the annotation is expressed through existing TEI elements and their attributes. The elements can be empty or contain text; the latter is true in the case of text added by the editor, such as in notes, bibliographical references, and normalized text. Three new attributes, `@stf_target`, `@stf_from` and `@stf_to`, are introduced specifically for the purpose of referencing the annotated text using the same `teidata.pointer` datatype as the familiar `@target` attribute (see Example 3).
Declaration of elements standoff and stf

<table>
<thead>
<tr>
<th>element standoff</th>
<th>element stf</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>{</td>
<td>{</td>
</tr>
<tr>
<td>att.global.attributes</td>
<td>att.global.attributes,</td>
</tr>
<tr>
<td></td>
<td>att.typed.attributes,</td>
</tr>
<tr>
<td></td>
<td>att.ascribed.attributes,</td>
</tr>
<tr>
<td></td>
<td>att.datable.w3c.attributes</td>
</tr>
<tr>
<td>(stf+)</td>
<td>(model.phrase, model.inter, model.global,</td>
</tr>
<tr>
<td></td>
<td>model.common, model.divPart, model.divLike,</td>
</tr>
<tr>
<td></td>
<td>model.divTop, model.divBottom)*</td>
</tr>
<tr>
<td></td>
<td>}</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The <text> contains the reference text for the stand-off markup, which is tokenized and each token is encoded as a <seg>, with a unique identifier represented as @xml:id. In our scenario, the text is tokenized at a word level, distinguishing words and punctuation, though it would be equally justifiable to tokenize with a different granularity, e.g. at character level. The @xml:ids reflect this choice, with values starting with ‘w’ or ‘pc’, accordingly (see Example 4). For entities in between <seg>s, as gaps or images, anchors can be set. As nothing below the token level can be addressed, it is important to take this into account at tokenization stage and choose suitable granularity. In our example, for instance, it is not possible to point to a chunk of text smaller than a word. This issue would not be as prominent if pointing mechanisms proposed by XPointer specification were already available. XPointer proposes to look at the text as a stream and a graph, not only as a hierarchy, making it possible to address unmarked portions of a document, e.g. parts of a token or target locations before or after a certain element. As these options unfortunately remain unimplemented, we propose, when necessary, to create segments shorter than the default level of granularity: e.g. in case of word tokenization, when a page (or a column or a line) starts (or ends) in the middle of a word it would be divided into two <seg>s (see Example 5).

3. Stand-off layers

Each <stf> element, as mentioned above, contains the stand-off annotations that belong to a single layer. A text can be conceived as a set of disjointed layers, each representing a logical hierarchy dealing with a single characteristic of a text, e.g. references to places or physical organization of source document into pages. Some phenomena, though represented with
different TEI elements may still be considered to belong to the same layer, as they form a clear aggregation structure. Obvious examples of such structures are verse lines and stanzas (<l> and <lg>) or logical division of the text into volumes, chapters and sections. Ultimately division into layers should be informed by particular research goals of the project and specificity of the source material, for example: named entities such as persons and places may end up in a single layer or in two different ones, depending on the focus of the project; markup of the same features created by different encoders will likely be kept separated. From this perspective a unique formal prerequisite for a layer is that its components remain in whole/part or aggregation relationship, which assures that an overlap will never occur between ancestor/descendants in a single layer. Thus, division into pages and paragraphs clearly forms different layers, since even though a page may be an aggregation of complete paragraphs, paragraphs equally well may span over the page boundary. However, there is no requirement for targets of sibling elements in a layer not to overlap.

As follows from above, all annotations for a single layer can be represented in a nested structure of elements. Some examples of nesting hierarchies that can be treated as layers are:

- page/line
- gb/pb/cb/lb (see Example 6)
- div type='chapter'/div type='section'/p
- lg/l (see Example 7)
- app/rdg (see Example 3)

Annotations using the same TEI element are usually grouped together, as is the case of <persName>s (see Example 3). Exceptions are the elements that have different functions in TEI depending on context or values of their attributes and thus belong to separate layers, as is the case of a <note> element: used both for marginal notes in the source and commentary notes from the editor.

Variant readings usually represented inside a <choice> element (e.g. <expan> for the expanded abbreviation, <reg> for the regularized form, etc.) are encoded as separate stand-off layers, while the base reading forms a part of the reference text (see Example 8).

4. Inline to stand-off transformation

We describe here a transformation scenario from inline to stand-off. The XSLT can be used to transform into stand-off markup the already existing inline markup of an encoded text.
The transformation includes several steps:
- the text is tokenized and each token is encoded as a `<seg>` (see example 9)
- each `<seg>` is given a @xml:id
- all the reference text, consisting of `<seg>`s, is enclosed within the `<text>`
- `<stf>` elements are created for each layer (see example 10)

The example below (Example 9) illustrates an implementation of the tokenization step in XSLT.

Next example demonstrates a way to create a stand-off layer for references to people previously encoded inline as `<persName>`s. This example could be generalized into XSLT function that takes element name and desired stand-off layer identifier to produce stand-off annotation, but for the sake of readability here we would like to concentrate on a concrete case.

5. Querying stand-off

The stand-off encoding can be still queried with widely available XML tools, as XPath and XQuery processors. Some examples of such queries are presented below. The first example (Example 11) aims to retrieve all names of people, i.e. the contents of `<persName>` elements in TEI encoding.

Example 11.

**inline markup:**
```
//persName/text()
```

**stand-off markup:**
```
//seg[@xml:id = //persName/@stf_target/translate(.,'#','')]""
```

Another example (Example 12) retrieves the text content of the fourth line

Example 12.

**inline markup:**
```
//l[4]/text()
```

**stand-off markup:**
```
//seg[following-sibling::seg[1][preceding-sibling::seg[@xml:id =
//div[@type='line'][4]/@stf_from/translate(.,'#','')] and
preceding-sibling::seg[1][following-sibling::seg[@xml:id =
//div[@type='line'][4]/@stf_to/translate(.,'#','')]])"
```
It is immediately evident that querying stand-off markup that way, though possible, can get very cumbersome, even for conceptually simple searches that would have very concise equivalent for inline TEI markup. Similarly performance of such XPath expressions on larger corpora might be extremely poor. Users, in particular, cannot be expected to produce such complex expressions by hand. This difficulty calls for a layer of abstraction that facilitates query building, possibly through enhanced XPath notation, that makes room for categories such as ‘being part of the same range as’, in addition to usual XPath axes of ancestry, descendancy etc. It is necessary to note that implementation of such concepts poses a great challenge in itself though and we consider it to be beyond the scope of this paper.

6. Visualizing

The challenge of stand-off annotation, especially where multiple overlapping hierarchies are involved, lies not only in proposing an adequate notation, but in the actual processing of documents encoded in a stand-off manner - for visualization, publishing and efficient querying. The bulk of publishing and processing tools relies on a single hierarchical structure. As an exercise in direct visualization of stand-off markup we present here not just a notation but also a proof-of-concept publication scenario with corresponding application. As our starting point we take the reading version of the reference text with additional markup layers imposed on it through graphical overlays with additional interactive content. We believe such a representation can be useful for publishing purposes, enables interactive engagement with the text, and also illustrates well the underlying concept of separate markup layers.

We consider a roughly grained structural layer of markup to be necessary for formatted presentation of the text on a screen, thus encoding this layer is highly recommended. We recognize, though, that for some purposes (eg. an edition based on documents, with floating commentaries for complicated manuscripts) representation in linear (reading) form is of
secondary or no concern, thus structural markup may not be present or may be based on arbitrary visual or physical groupings like page or zone.

7. Further challenges

The document containing the stand-off annotations can be validated through a customized schema. It contains declarations for the new elements `<standoff>` and `<stf>` and for the new attributes `@stf_target`, `@stf_from` and `@stf_to`. The customized schema validates the syntax presented in the above examples.9 Nevertheless, the strength of a TEI schema is lost here: the elements, floating inside `<stf>` tags, lose their mutual relationships, hence there is nothing preventing, for instance, a `<hi>` to include a `<p>` (and not vice versa). For this reason, the validation of stand-off annotation calls for more than just checking syntactical correctness. One way of testing if stand-off markup is conformant with the TEI model could be transforming it back to inline markup in order to be validated. Two approaches could be pursued here: moving the whole markup inline, introducing milestone elements in the case of an overlap, or validating one layer at the time (again, transforming the stand-off annotation to inline markup), since by design a document with stand-off annotations may not fit the TEI content model. The latter seems to be more in line with the understanding of layers as disparate logical hierarchies.

8. Conclusion

The TEI community can be considered as a growing group of practitioners, who are more or less familiar with the facilities provided by the technology and with the conceptual model behind the XML implementation of the TEI Guidelines. At the same time they are often, though not always, more interested in investigating texts through the TEI, than they are in investigating the TEI as a technical infrastructure itself. A stand-off markup proposal that hopes to be successful with such a large community could therefore follow two paths: provide a model that diverges as little as possible from current practices of encoding and processing TEI documents or, alternatively supply a complete work environment that can significantly differ from them. The latter though should cover not only the encoding but also the processing - for instance querying and visualizing - of the documents produced.

This paper aims to make a start along the first path, raising critical issues related to modelling and technical requirements for the implementation of a "TEI-flavored" stand-off markup. Not claiming to provide definitive solutions, our paper sketches main requirements and points out some key problems, such as the concept of layers, the existence and the granularity of the reference-text, pointing mechanisms and challenges in direct querying. As such, it can be considered a point of departure for re-entering the discussion on stand-off approaches to encoding and moving more steps beyond.

Notes

1 A previous version of this paper has been presented as a poster at the annual TEI Conference, Lyon, 2015. The research leading to these results has received funding from the People Programme (Marie Curie Actions) of the European Union's Seventh Framework Programme FP7/2007-2013/ under REA grant agreement n° 317436.
4 The person who create the `<stf>` can be indicated using the attribute @who, from att.ascribed.
5 We use the term aggregation as understood in Object-Oriented Programming, cf e.g.
6 The occurrences of overlapping sibling elements in a single layer should be rare; some examples might be: overlapping notes, deletions or physical damages. In the case where overlapping sibling elements carries semantically different information, they should be organized into separate layers. This is the case, for example for overlapping *apparatus*.

The commonly raised problem of overlapping *apparatus* can be tackled as an issue of granularity: organizing the readings in smaller units (e.g., words) would avoid overlapping. If this granularity is not suitable, overlapping *apparatus* should be organized in separate layers in the stand-off markup, because they contain different and semantically disjoint arguments about the text, as in Example 13.

8 The stylesheets made for this proof-of-concept do not provide for the transformation of all TEI elements. Nevertheless the mechanism for one element (as in example 10) can be replicated for the others with no or little customization.

---

**Works cited**

<http://www.balisage.net/Proceedings/vol5/html/Banski01/BalisageVol5-Banski01.html>


