Journal of the Text Encoding Initiative

Issue 8 | December 2014 - December 2015
Selected Papers from the 2013 TEI Conference

TEI in LMNL: Implications for Modeling

Wendell Piez

Electronic version
URL: http://journals.openedition.org/jtei/1337
DOI: 10.4000/jtei.1337
ISSN: 2162-5603

Publisher
TEI Consortium

Electronic reference
Wendell Piez, « TEI in LMNL: Implications for Modeling », *Journal of the Text Encoding Initiative* [Online], Issue 8 | December 2014 - December 2015, Online since 22 October 2015, connection on 01 May 2019. URL : http://journals.openedition.org/jtei/1337 ; DOI : 10.4000/jtei.1337

TEI in LMNL: Implications for Modeling

Wendell Piez

1. TEI in LMNL

1 LMNL, the Layered Markup and Annotation Language, describes a data model and syntax for an experimental prototype in document description, modeling, and markup. Properly a meta-markup language, LMNL has no native semantics beyond the idea of annotated ranges over texts. Proposed over ten years ago (Tennison and Piez 2002) and developed intermittently since then, LMNL is not so much an alternative to XML as a foil for it, substituting the information contained in XML hierarchies with "a more rudimentary data model that would capture the information we needed, while not itself trying to assert containment or sibling relations" (Tennison and Piez 2002). For years we have had only partial or prototype implementations of LMNL-based processing systems (see Czmiel 2004, Caton 2005, and Piez 2012), or mockups of what LMNL would be like (Piez 2004) —tantalizingly capable even while fragile and short-lived—and we have had more and more LMNL-like solutions (both within and outside XML) to the problems of representation addressed by LMNL (including multiple concurrent hierarchies and "arbitrary overlap," that is, the overlap between ranges of the same type such as texts marked for indexing for annotation). However, it has become more and more apparent that what Jeni Tennison and I dreamed up then was, in a certain sense,

"obvious," and (as such) one of those things that will continually be rediscovered. As if it were a territory only now being mapped, LMNL was always there, and indeed visited, before Jeni and I discussed it. One might say that the LMNL we proposed is nothing more than a particular, peculiar way of generalizing some solutions that are increasingly all around us every day. Nonetheless, with a LMNL processor in hand, it turns out one can start doing some interesting things, as it does indeed move the boundaries of what is possible. Both the analysis and the representation of literary texts (just to take the examples with which I have worked) are enabled in new ways by being able to overlap freely in the model. We find LMNL gives us a way of approaching modeling and markup that sheds an oblique light on systems (such as XML-based systems or the HTML-based web) with which we are more familiar. Indeed this is, perhaps, what has stimulated such interest as it has attracted, despite its evident weaknesses and immaturity.

2 In one important respect, LMNL is similar to XML. While it projects a model of a text in a hierarchy of *elements*, whose properties (names, contents, organization) are regular enough to support higher-level application semantics to be described for a particular process (and/or a generalized set of processes), XML imposes no "native semantics" in the sense that it is left to an application designer to define element types, permissible relations among element types, and their behaviors and "meaning" in application. This is done in XML by designing and deploying a controlled vocabulary—a tag set—along with rules for its use, which together describe an XML document type along with an abstract document model. Of course, TEI is an example of this. The semantics of TEI element types are defined not by XML but by TEI; TEI-conformant applications can and do take advantage of the categories and relations described by TEI (not just those that might be inferred directly from the XML structure) in doing useful things with TEI data.

3 Similarly, LMNL does not preclude any naming or any expression of constraints, instead leaving the assignment of range names and semantics to its user; and this (as it is in XML) may be a significant burden. TEI gives us a good place to start whenever we have to consider naming and typing problems. (While largely unappreciated, this is one aspect of TEI that is invaluable.) Then, too, if we could pull TEI data easily into LMNL-based systems, we would be able to take advantage of the prior investment in TEI and perhaps save ourselves steps (at least in areas where the TEI markup remains serviceable).

4 This leads to the intriguing idea that we might build on TEI semantics when working with data in LMNL. The tagging could be TEI XML, or it could be a LMNL rendition of the same structures, but the data model would be LMNL and could represent any overlapping ranges in the data directly, rather than implicitly by means of stand-off or pointer mechanisms, as has to be done in XML.

5 A small example (the first quatrain of a sonnet of Petrarch, in TEI XML) appears below. Two hierarchies are encoded: the poetic structure (prioritized in the XML structure) and the linguistic (sentences and phrases), with the components of the latter linked using the TEI `@part` attribute. (Of course this is a vast simplication of what one might actually wish to do.)

```
  <lg type="quatrain">
   <l><seg type="s" part="I"><seg type="phr" part="N">L’arbor gentil che forte
amai molt’anni,</seg></seg></l>
   <l><seg type="s" part="M"><seg type="phr" part="I">mentre i bei rami non
m’ebber a sdegno</seg></seg></l>
   <l><seg type="s" part="M"><seg type="phr" part="M">fiorir faceva il mio debile
ingegno</seg></seg></l>
   <l><seg type="s" part="F"><seg type="phr" part="F">e la sua ombra,</seg> <seg
type="phr">et crescer negli affanni.</seg></seg></l>
  </lg>
```

6 Transposed into LMNL “naïvely” using a generic transformation:

Example 1. LMNL representation of a quatrain (see Piez 2012 for a detailed explanation of LMNL syntax).

```
[lg [type}quatrain{]}
  [l}[seg [type}s{][part}I{]}[seg [type}phr{][part}N{]}L’arbor gentil che forte
 amai molt’anni,{seg]{seg]{l]
  [l}[seg [type}s{][part}M{]}[seg [type}phr{][part}I{]}mentre i bei rami non
 m’ebber a sdegno{seg]{seg]{l]
  [l}[seg [type}s{][part}M{]}[seg [type}phr{][part}M{]}fiorir faceva il mio debile
 ingegno{seg]{seg]{l]
  [l}[seg [type}s{][part}F{]} [seg [type}phr{][part}F{]}e la sua ombra,{seg][seg
 [type}phr{][part}N{]}et crescer negli affanni.{seg]{seg]{l]
{lg]
```

7 Or, using a transformation amended to recognize the semantics of TEI `@part`:

Example 2. LMNL representation of a quatrain including `@part` attribute information.

```
[lg [type}quatrain{]}
  [l}[seg [type}s{]}[seg [type}phr{]}L’arbor gentil che forte amai molt’anni,{seg]
{l]
  [l}[seg [type}phr{]}mentre i bei rami non m’ebber a sdegno{l]
  [l}fiorir faceva il mio debile ingegno{l]
  [l}e la sua ombra,{seg] [seg [type}phr{]}et crescer negli affanni.{seg]{seg]{l]
{lg]
```

8 In this case, ranges mapped from elements marked as `@part='I'` get start tags but no end tags, since they are taken not to end their ranges. Similarly, elements marked as `@part="F"` get no start tags, while those marked as `@part="M"` get no tags at all.

9 Note that even this transformation is not lossy, since a reverse transformation that induced an element hierarchy from the ranges would be able to mark elements that had to be segmented with their appropriate `@part` attributes. (To be sure, a processor that wrote such segmented elements would have to respect the intended semantics of TEI `@part` in doing so—`"I"` for the first segment in the `seg` range, `"F"` for the last, and `"M"` for all the others. Yet since the sequencing is determinable from the data, the process can be automated.)

10 Finally, one is tempted to promote `@type` values here into range names. This is no longer TEI, but nonetheless corresponds semantically to the TEI we began with:

Example 3. LNML representation including information from TEI `@type`.

```
[quatrain}
  [l}[s}[phr}L’arbor gentil che forte amai molt’anni,{phr]{l]
  [l}[phr}mentre i bei rami non m’ebber a sdegno{l]
  [l}fiorir faceva il mio debile ingegno{l]
  [l}e la sua ombra,{phr] [phr}et crescer negli affanni.{phr]{s]{l]
{quatrain]
```

11 All of these LMNL instances are ready for processing by a LMNL processor.

12 This capability of manipulating the data directly raises many questions for modeling documentary data using markup, as outlined below.

2. Implications for Modeling

13 The foregoing examples show how LMNL's ability to represent overlapping ranges opens modeling opportunities. But there is more to it than this.

14 LMNL also supports structured annotations on ranges. In a way analogous to XML attributes attached to XML ranges, LMNL annotations can be associated with LMNL ranges. However, they also offer several features that XML attributes do not:

1. LMNL annotations may be marked up. Thus, they may be structured in the same way as documents may be structured (in other words, LMNL annotations are isomorphic to documents).
2. The order of annotations assigned to a range is preserved in the model.
3. Annotations may have the same name as other annotations on the same range. And (like ranges) they may be anonymous.
4. Since ranges may also be anonymous, all the properties assigned to a range may be represented in its annotations.
5. Annotations may be annotated. Accordingly, a range may have a "tree" of annotations, each annotation further annotated. This also makes LMNL annotations composable.[1]

15 In LMNL syntax, annotations are shown using the same syntax as ranges; the only differences are that they are placed in tags (start or end tags for marked ranges, or in tags representing empty ranges or atoms) and that they may not, unlike ranges, overlap with one another. (Every annotation is discrete even though its contents may include overlapping ranges.) Finally, in order to reduce tagging overhead, end tags for simple annotations (those with no internal markup) may be represented using anonymous end-tag notation, as `{]`.

16 So this is a (single) legal LMNL start tag: `[sonnet [author [name}Francesco Petrarca{] [born [year}1304{][place}Arezzo{]] [deceased [year}1374{][place}Arquà{]]]}` The range tagged as `sonnet` (whose start tag is shown) has an annotation named `author` with three annotations of its own: `name`, `born`, and `deceased`. The annotations `born` and `deceased` each have a `year` and `place` annotation. So we have four levels of annotations annotating annotations; for example, `sonnet/author/born/place` here is *Arezzo*. Thus LMNL annotations present an XML-like capability

of structure and hierarchy, presenting navigable, structured data sets in much the way XML does —even while ranges can overlap within a single layer (whether in the document itself, or in an annotation's value or "text layer").

17 Below is the same code, except using indentation to indicate where annotations have annotations. The key is to understand that `[born [year}1304{]]` represents an empty annotation (which uses the same tagging syntax as an empty range), itself with a single annotation (named "year," with a value of "1304"). The tagging syntax is recursive—range markup is used inside tags to indicate annotations on the ranges indicated by those tags. In this example, `{]` serves as an abbreviation for `{year]` (an end-tag for the "year" annotation).

Example 4. A range in LMNL, with embedded annotations.

```
[sonnet
   [author [name}Francesco Petrarca{]
           [born [year}1304{][place}Arezzo{]]
           [deceased [year}1374{][place}Arquà{]]]}
```

18 Overlapping ranges and structured annotations together give LMNL capabilities for modeling very different from those of XML. Just as a transposition from XML syntax to LMNL syntax can expose whatever overlap is indicated by TEI XML directly (by showing start and end tags only where the entire range starts and ends, not every place an XML element boundary appears), it can also promote XML elements into LMNL annotation structures instead of LMNL ranges. Nothing need be flattened: because LMNL annotations can support structure and markup, annotations are first-class objects. For example, it might be considered natural to present a TEI header as an annotation (with its own annotations for `<fileDesc>`, `<sourceDesc>`, etc.) in the start tag of a TEI instance. In such a LMNL document, the header would not be considered part of the content of the document, but would be considered an annotation of it—an arrangement which, arguably, better reflects its status as metadata.

19 Modeling in LMNL is thus more flexible and adaptable than modeling in XML. If subordinate information such as alternative readings, analytical descriptions, or component-level metadata can be represented in annotations, the main text itself can maintain its integrity as a distinct artifact—unlike in XML, in which a transcription's editorial apparatus, as soon as it gets too complicated to represent in attributes (i.e., name/value pairs), must be (if it is to be stored

in the same document) promoted into element contents within the same conceptual "tree" as the transcription itself. Similarly, when a text must be organized into multiple concurrent hierarchies (such as grammatical, rhetorical, narratological, and prosodic hierarchies, or a hierarchy representing the physical organization of text in volumes and pages), these can simply be tagged in LMNL and processed appropriately according to rules that respect their semantics.

20 This can be demonstrated in a LMNL representation of a Shakespeare play (converted, like the sonnet example given above, from TEI source data, in this case from the Folger Digital Editions of Shakespeare). Here, the dramatic and dialogic form is represented directly along with the verse structure (where the plays are in verse). Tagged in LMNL, a line of verse is always a line, and does not have to be split between two speeches when it spans across them:

Example 5. An extract from Act 1, Scene 2 of *The Tempest*.

```
[sp [speaker}ARIEL{speaker]}
[line [n}1.2.231{]}To every article.{line]
[line [n}1.2.232{]}I boarded the King's ship; now on the beak,{line]

[...]

[line [n}1.2.242{]}Yea, his dread trident shake. {sp]
[sp [speaker}PROSPERO{speaker]}My brave spirit!{line]
[line [n}1.2.244{]}Who was so firm, so constant, that this coil{line]
[line [n}1.2.245{]}Would not infect his reason? {sp]
```

21 Interestingly, it becomes possible to represent settings, stage directions, and metadata (such as act and scene headers) in the play in the form of annotations, thus "hiding" them away from the base text, while making them readily available should we want them. Since annotations can be annotated, they provide a convenient way of presenting data (such as complex metadata) that truly is hierarchical, while markup in the text flow can freely describe whatever the researcher deems to be important, irrespective of any structural relations (or lack thereof) among features of interest.

3. Current Capabilities, Potentials, and Work to be Done

22 As shown above, LMNL markup is not difficult to generate programmatically from TEI data; moreover, where TEI markup can be taken to indicate range relations rather than structure (either by marking with milestones or by element alignment), a little logic can translate it into LMNL start/end tag pairs, thus representing the range directly (as in the third quatrain example above). Otherwise, any hierarchies that happen to be present in the XML, will persist, albeit expressed as sets of "enclosing" and "enclosed" ranges (and no longer "parents" and "children").

23 In order to process LMNL syntax inputs, however, we need a parser and some sort of processing stack. Since 2011 I have been prototyping such a LMNL framework in the form of Luminescent (see Piez 2012). Luminescent is implemented as an XSLT pipeline capable of "compiling" LMNL markup into a database-like representation, expressed as XML, which presents all the same information as stand-off pointers into a sequence of segments of the text. This same technique is used internally by several systems dealing with overlap in XML (see in particular CATMA[2] and XStandoff[3]), though details differ. As it is XML-based, this format can in turn be queried, processed, and transformed using XML technologies including XPath and XSLT to generate outputs in XML, HTML, SVG, or plain text for display or any other application.

24 One such target format of LMNL processing is a hierarchical "view" of a LMNL document—an XML hierarchy interpolated on request, promoting any range types to (nesting) elements. Given a set of LMNL ranges, in other words, it is possible to stipulate that you want a structured document promoting whichever ranges you wish into first-class element structures in an XML document—which becomes the so-called "sacred" hierarchy as described by Marinelli, Vitali, and Zacchiroli (2008), while "profane" hierarchies remain implicit.[4] This might be a convenient form to have on the way to a presentational format such as HTML. Then, for another purpose, you can return to the LMNL source (or its compiled XML representation) and request a different hierarchy. XML, in other words, remains available and useful for transitional formats between LMNL and the results one ultimately wishes to produce.

25 Some of the outputs offered by Luminescent are on display in the supplementary materials provided with this article.

26 However intriguing some of these may be, in many cases the TEI practitioner may ask what is the point of working so hard not to use XML? Or more closely: if the data is TEI going in, and the first thing that happens to the LMNL is to convert it back into XML, then why not skip the LMNL syntax? Both of these are fair questions. I would in fact agree that everything being done with LMNL here might actually be more straightforwardly and easily done by processing straight from TEI into the various target formats. After all, we actually have ways of dealing with overlap now in XSLT 2.0, as I have reported (Piez 2011).

27 However, it can be argued that efficiency and ease of use are not actually among the primary goals of practitioners of TEI, who have had various research agendas. The intent here is not to demonstrate anything as superior or inferior, either in general or for particular purposes. LMNL may be useful, or it may merely be interesting for the light it casts. To be sure, it supports, in a generalized way, certain operations that are difficult or onerous to engineer in XML natively (and have to be done case by case), such as hierarchical transposition (for example, converting a document representing pages, such as a TEI Genetic Editions instance, into a "logical" view of the same document), or filtering and extraction of ranges of text associated with structures not directly represented in the primary hierarchy of an XML document, such as narrative or dialogic structures. It is not that these operations cannot be done in TEI—they can be and have been done (typically using advanced features in XSLT or XQuery). However, when it comes to a certain kind of work with markup (especially an application of markup one might call exploratory as well as descriptive [Piez 2001]), there is something liberating about LMNL's simplicity and flexibility. In particular, when multiple concurrent hierarchies (MCH) are of special interest, to be unburdened of the chore of assigning and maintaining pointers or references (something without which XML simply cannot approach these problems) comes as a considerable relief: even in a plain text editor, one can simply *tag,* focusing on the tasks of tagging. Then too, in LMNL one can simply start tagging, both without worrying about the startup development that inevitably goes with an XML application of this sort,[5] and with confidence that certain very generic processes (including processes that can convert LMNL ranges into XML element hierarchies) can be applied to LMNL at any time without any commitment at all to specialized application code for any particular application semantic. This ease of use has an important consequence: essentially, LMNL allows

the text encoder to delay commitments to one or another hierarchy as primary until later in the process. This makes it an appealing "counter case" for the use of text-based markup as well as for the tree-shaped model an XML document must normally take.

28 It should not need to be stressed that LMNL (as a technology, not simply a conception) has a long way to go to be dependable; the gain in expressiveness when tagged structures can overlap, and annotations can be annotated, comes at a price. At this point the greatest difficulty is obviously the fact that the processing stack is entirely custom-built and "bespoke," using XML technologies to implement both the parsing of LMNL syntax and processing the range-based data model it was designed to represent. Yet even if this were not an impediment for users other than myself, there are nonetheless important caveats. Without anything strictly analogous to schemas in XML, LMNL is unsuitable for many of the tasks for which XML is so good, such as the preparation of documents for publishing—a task for which validation to a schema, more or less explicit, is essential. Yet even this contrast is instructive, as much of what makes LMNL so interesting is how it provokes thinking about such questions as what a schema language would look like that could account for overlap and MCH (see Sperberg-McQueen 2006 and Tennison 2007), as well as prompting experiments such as an XQuery function library that supports querying a LMNL data model (including overlaps), something I have more recently been doing in Luminescent.

29 Yet the considerable difference between a mature, robust set of standards-based technologies and a small research project (however ambitious in its conception) may also shed light on the greatest strengths of TEI and XML even while it prompts us to think about what is interesting about LMNL, especially in the context of TEI. The primary goal of text encoding in the humanities should not be to conform to standards, although that may be a necessary means to important ends. Rather, we encode texts and represent them digitally in order to present, examine, study, and reflect on the rich heritage of knowledge and expression presented to us in our cultural legacy. TEI markup does this, and the practicability of getting the information out of a TEI XML file and into something quite different shows us how much we do *not* have to make up, from scratch, even while we grab arbitrary data sets from the Internet. LMNL is both possible and interesting largely because it already has XML technologies, including TEI, to do so much of the heavy lifting. Yet, while it has been evident from early days that texts in the humanities pose special problems for digital processing—because they are characterized by complex structures including concurrent hierarchies (see, especially,

Sperberg-McQueen 1991; Renear, Mylonas, and Durand 1993; and Huitfeldt 1994–95)—we have also been hampered by the fact that the best technologies available have been optimized for one structure at a time. Experimenting with a markup and modeling technology that does not force the false choice between one structure and another, we discover not only new capabilities for the digital processing of humanities texts—simultaneously building on TEI and extending it to new uses—but also something about the way our tools condition our view of our texts.

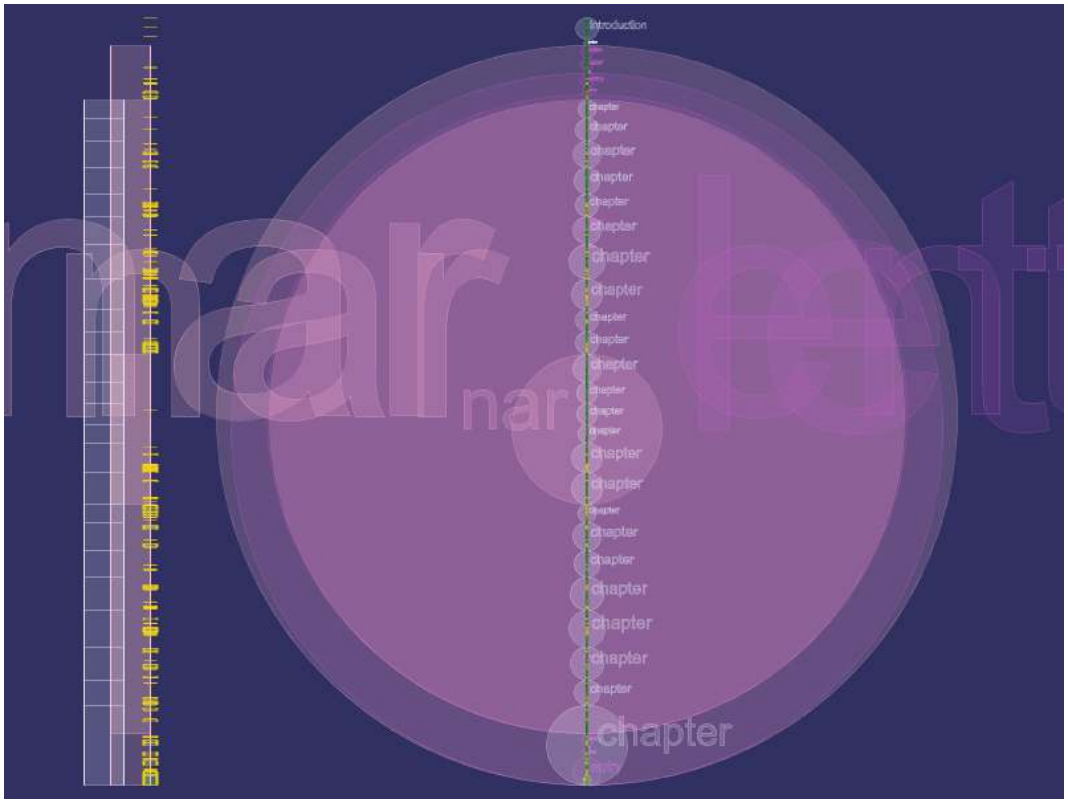
4. Supporting Materials

30 This article includes a number of small demonstrations of LMNL-tagged documents and of processing outputs from Luminescent. These outputs vary in quality (and in some cases, large file sizes may stress available system resources), but should be self-explanatory. Any of the HTML or SVG files may be opened in a browser, although for best results, an SVG viewer which supports zoom to arbitrary levelssuch as Apache Batik (available alone or as the SVG Viewer in the oXygen XML editor) should be used. LMNL text files can be viewed in any browser or text editor.

- Sonnet examples: Several sonnets, including the Petrarch example above, are given in LMNL, along with HTML/SVG-formatted display versions, with diagrams showing their structure, including overlap between verse and grammatical form.
- Longer poems: Percy Shelley's "Julian and Maddalo" and William Butler Yeats's "Easter 1916" are provided with similar treatments.
- Folger Digital Texts, converted into LMNL: These editions of several of Shakespeare's plays are converted from TEI inputs and show TEI semantic usage in LMNL data, with some structural adjustments. Also included are SVG "maps" of the plays, showing their structure by means of an illustration of the extent of acts, scenes, and speeches.
- A novel: Mary Shelley's epistolary novel *Frankenstein* is marked up in LMNL syntax. Again, for the most part TEI element and attribute names and semantics are used. Three concurrent hierarchies are encoded: pages in the source edition; narrative and dialogic structures; and the logical structure of letters, chapters, and paragraphs.

31 LMNL is documented at http://www.lmnl-markup.org/; Luminescent is available on Github, at https://github.com/wendellpiez/Luminescent.

Figure 1. Mary Shelley's *Frankenstein*.



32 Depicted in figure 1 is a diagram generated programmatically from a data instance, marked up in LMNL, representing the disposition of ranges within the nineteenth-century novel.The size and placement of "bubbles" indicate the extent of ranges within the text, as identified by markup. At http://wendellpiez.com/FrankensteinTransformed, a more highly developed and perhaps intelligible version is available.

BIBLIOGRAPHY

Caton, Paul. 2005. "LMNL Matters?" Paper presented at Extreme Markup Languages 2005, Montréal, Quebec, August 1–5. Abstract: http://conferences.idealliance.org/extreme/html/2005/Caton01/EML2005Caton01.html.

Czmiel, Alexander. 2004. "XML for Overlapping Structures (XfOS) Using a Non XML Data Model." Paper presented at ALLC/ACH 2004, Gothenburg, Sweden, June 11–16.

Huitfeldt, Claus. 1994–95. "Multi-Dimensional Texts in a One-Dimensional Medium." In "Humanities Computing in Norway," special issue, *Computers and the Humanities* 28 (4/5): 235–41. doi:10.1007/BF01830270.

Marinelli, Paolo, Fabio Vitali, and Stefano Zacchiroli. 2008. "Towards the unification of formats for overlapping markup." *New Review of Hypermedia and Multimedia*,14,1. https://hal.archives-ouvertes.fr/hal-00340578. Doi: 10.1080/13614560802316145.

Piez, Wendell. 2001. "Beyond the 'Descriptive vs. Procedural' Distinction." *Markup Languages: Theory & Practice* 3 (2). http://www.wendellpiez.com/resources/publications/beyonddistinction.pdf.

———. 2004. "Half-steps toward LMNL." In *Proceedings of Extreme Markup Languages 2004*. http://conferences.idealliance.org/extreme/html/2004/Piez01/EML2004Piez01.html.

———. 2011. "TEI Overlap Demonstration: Processing TEI P5, with Overlap, Using XSLT 2.0 in the Client." For the Interedition 2011 Boot Camp, TEI Members Meeting 2011, Würzburg, Lower Franconia. Last updated February 2014. http://piez.org/wendell/projects/Interedition2011/.

———. 2012. "Luminescent: Parsing LMNL by XSLT Upconversion." In *Proceedings of Balisage: The Markup Conference 2012*. Balisage Series on Markup Technologies, vol. 8. doi:10.4242/BalisageVol8.Piez01.

Renear, Allen, Elli Mylonas, and David Durand. 1993. "Refining Our Notion of What Text Really Is: The Problem of Overlapping Hierarchies." Final version, January 6. http://www.stg.brown.edu/resources/stg/monographs/ohco.html.

Sperberg-McQueen, C. M. 1991. "Text in the Electronic Age: Textual Study and Text Encoding, with Examples from Medieval Texts." *Literary and Linguistic Computing* 6 (1): 34–46. doi:10.1093/llc/6.1.34.

———. 2006. "Rabbit/Duck Grammars: A Validation Method for Overlapping Structures." In *Proceedings of Extreme Markup Languages 2006*. http://www.idealliance.org/papers/extreme/proceedings/html/2006/SperbergMcQueen01/EML2006SperbergMcQueen01.html.

Tennison, Jeni, and Wendell Piez. 2002. "The Layered Markup and Annotation Language (LMNL)." Paper presented at Extreme Markup Languages 2002, Montréal, Quebec, August 4–9. Abstract: http://xml.coverpages.org/LMNL-Abstract.html.

Tennison, Jeni. 2007. "Creole: Validating Overlapping Markup." Paper presented at XTech 2007, Paris, France, May 15–18. http://www.princexml.com/howcome/2007/xtech/papers/output/0077-30.pdf

TEI Consortium. 2014. "Non-hierarchical Structures" (chapter 20). In *TEI P5: Guidelines for Electronic Text Encoding and Interchange*. Version 2.6.0. Last updated January 20. N.p.: TEI Consortium. http://www.tei-c.org/Vault/P5/2.6.0/doc/tei-p5-doc/en/html/NH.html.

NOTES

1 Referring specifically to "composability" in the technical sense as described at https://en.wikipedia.org/wiki/Composability.

2 CATMA: Computer Aided Textual Markup and Analysis. See http://www.catma.de/.

3 http://www.xstandoff.net/.

4 Naturally, in mapping from LMNL to XML, one has to stipulate particular sets of ranges that are known to fall into hierarchies, to map to the XML hierarchy, lest the processor breaks elements into segments at points of overlap; fortunately another feature of LMNL is analytic processing that can report which range types overlap with which, and where.

5 Such as the CSS/Schematron apparatus one typically wants, in a structured XML editor, as an aid to editing the very pointers that one has no need for in LMNL.

ABSTRACT

LMNL (the Layered Markup and Annotation Language) is a small, if ambitious, research project in text encoding. Unlike XML, LMNL markup does not represent document components or constituents (as "elements") within a structure, but rather simply labels ranges of text within the document for identification and processing. Avoiding choices between structures, any and all structures can be elucidated; this reveals not only new capabilities for the digital processing of humanities texts—simultaneously building on TEI and extending it to new uses—but also something about the way our tools condition our view of our objects of study.

This paper presents LMNL and suggests some of its strengths in the context of TEI.

INDEX

Keywords: text encoding, visualization, markup, overlap, hierarchy

AUTHOR

WENDELL PIEZ

Wendell Piez is an independent consultant based in Rockville, Maryland, specializing in XML and XSLT.