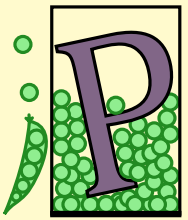
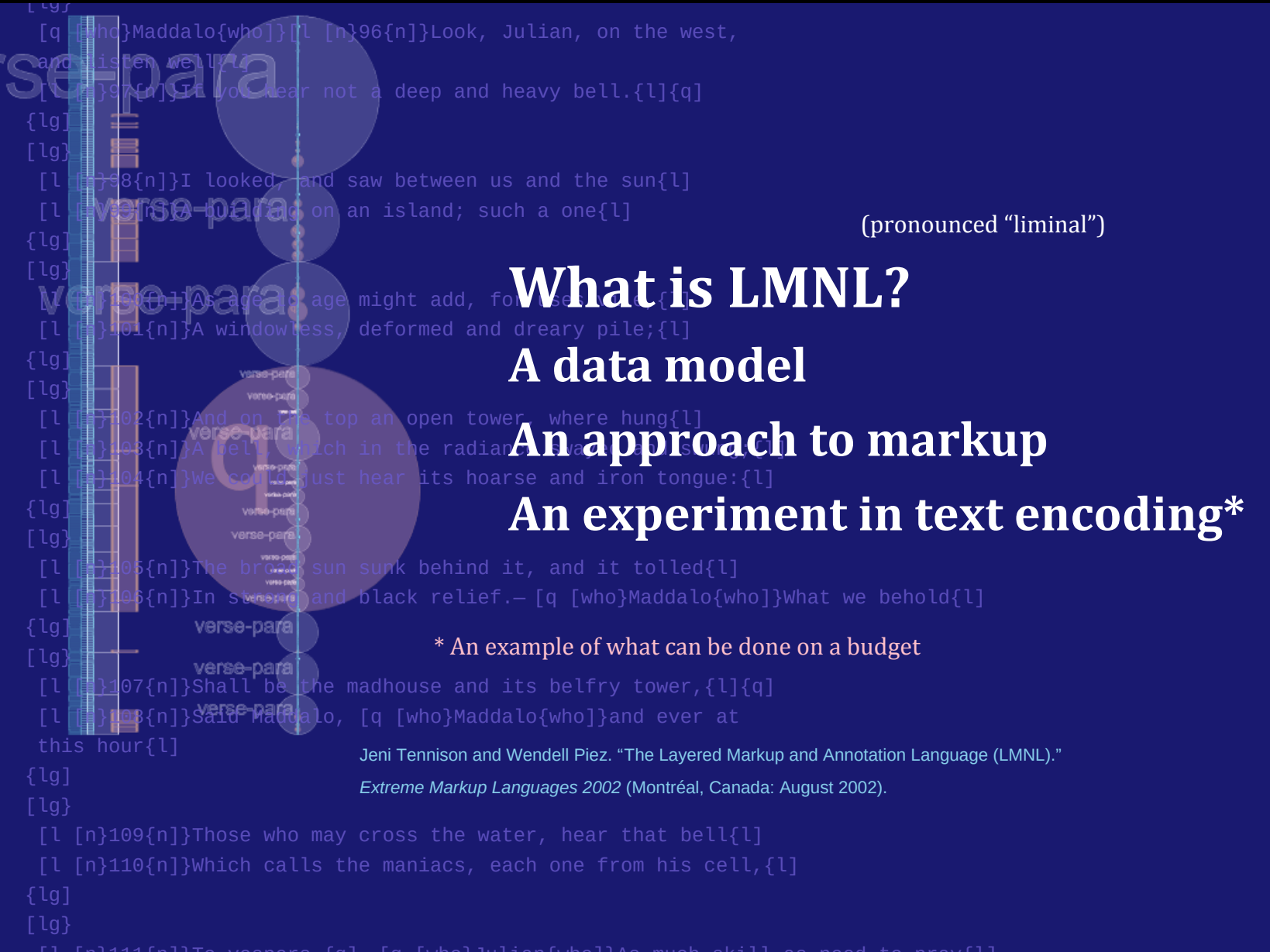
[title}

Markup Beyond XML

{title [alt}A LMNL Progress Report{]]



Wendell Piez
www.wendellpiez.com
Digital Humanities 2013
Lincoln, Nebraska
July 19 2013
Eat Your Vegetables



(pronounced “liminal”)

What is LMNL?

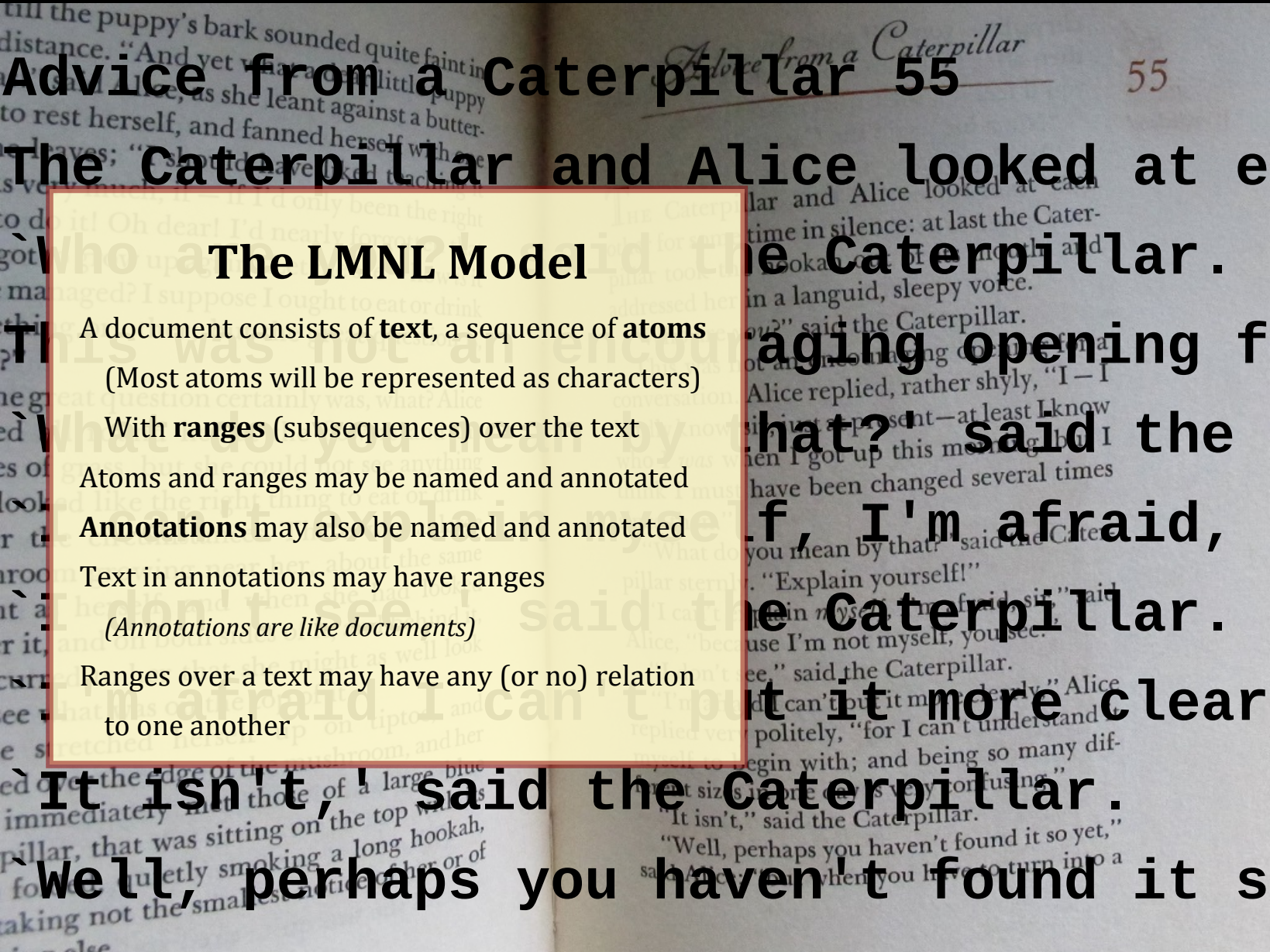
A data model

An approach to markup

An experiment in text encoding*

* An example of what can be done on a budget

Jeni Tennison and Wendell Piez. “The Layered Markup and Annotation Language (LMNL).” *Extreme Markup Languages 2002* (Montréal, Canada: August 2002).


Advice from a Caterpillar 55
The Caterpillar and Alice looked at e
`Who are you?' said the Caterpillar.
This was not an encouraging opening f
`What do you mean by that?' said the
I don't explain myself, I'm afraid,
`I don't see,' said the Caterpillar.
I'm afraid I can't put it more clear
`It isn't,' said the Caterpillar.
`Well, perhaps you haven't found it s
The LMNL Model
A document consists of **text**, a sequence of **atoms**
(Most atoms will be represented as characters)
With **ranges** (subsequences) over the text
Atoms and ranges may be named and annotated
Annotations may also be named and annotated
Text in annotations may have ranges
(Annotations are like documents)
Ranges over a text may have any (or no) relation to one another

Implementation

Implicit

XML may be mapped directly to LMNL
- Elements cover ranges
- (or segments of ranges, or “milestone” markers at range start/end points)

Or, LMNL may be represented using standoff markup
- Elements* maintained out of line indicate and represent ranges
- Pointers indicate range locations and extent
- Range or standoff properties (elements, attributes) become annotations
- * Or some other representation or notation (yes JSON)

Out of line

Or, LMNL may be maintained in a database or object structure
- represented in a UI but never serialized

Object model

But — range models have an Achilles heel:
- Defining, deploying, controlling and maintaining the pointers ...
- Plus, we will soon want a serialization (external format)

... well ...

(Markup syntax?)

Markup

... LMNL has a syntax ...

```
[poem [title}Paradise Lost{] [author}John Milton{]}
[book [n}1{n]}
[verse-paragraph}
[l [n}1.1{]}[s}[phr}Of Man's first disobedience,{phr] [phr}and the fruit{l]
[l [n}1.2{]}Of that forbidden tree whose mortal taste{l]
[l [n}1.3{]}Brought death into the World,{phr] [phr}and all our woe,{phr]{l]
[l [n}1.4{]}[phr}With loss of Eden,{phr] [phr}till one greater Man{l]
[l [n}1.5{]}Restore us,{phr] [phr}and regain the blissful seat,{phr]{l]
[l [n}1.6{]}[phr}Sing,{phr] [phr}Heavenly Muse,{phr] [phr}that,{phr] [phr}on the secret top
[l [n}1.7{]}Of Oreb,{phr] [phr}or of Sinai,{phr] [phr}didst inspire{l]
[l [n}1.8{]}That Shepherd who first taught the chosen seed{l]
[l [n}1.9{]}In the beginning how the heavens and earth{l]
[l [n}1.10{]}Rose out of Chaos:{phr] [phr}or,{phr] [phr}if Sion hill{l]
[l [n}1.11{]}Delight thee more,{phr] [phr}and Siloa's brook that flowed{l]
[l [n}1.12{]}Fast by the oracle of God,{phr] [phr}I thence{l]
[l [n}1.13{]}Invoke thy aid to my adventurous song,{phr]{l]
[l [n}1.14{]}[phr}That with no middle flight intends to soar{l]
[l [n}1.15{]}Above th' Aonian mount,{phr] [phr}while it pursues{l]
[l [n}1.16{]}Things unattempted yet in prose or rhyme {phr]{s]{l]
[l [n}1.17{]}[s}[phr}And chiefly thou {phr] [phr}O Spirit {phr] [phr}that dost prefer{l]
[l [n}1.18{]}Before all temples th' upright heart and pure,{phr]{l]
[l [n}1.19{]}[phr}Instruct me,{phr] [phr}for Thou know'st;{phr] [phr}Thou from the first{l]
[l [n}1.20{]}Wast present,{phr] [phr}and,{phr] [phr}with mighty wings outspread,{phr]{l]
[l [n}1.21{]}[phr}Dove-like sat'st brooding on the vast Abyss,{phr]{l]
```

Much like XML except

1. Relax well-formedness constraint on element typing (naming) in tags (“last in, first out”) while retaining tag types (start, end, empty)
2. Refactor and generalize attributes into annotations
3. Do without DTD (or top-down tree-based) validation/parsing for now
4. Tweak delimiters

   Recognizably not XML any more

   Does not conflict with XML if either is embedded in the other

Note: this is nothing against XML!

(On the contrary it will show how useful and powerful it is)

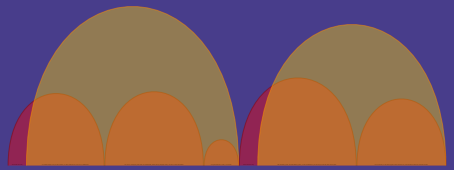
If XML didn't exist we'd have to invent it

In particular, XML turns out to be

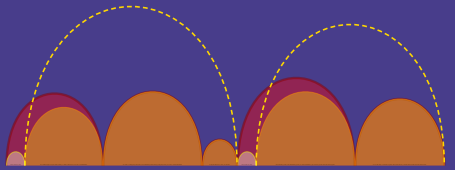
a very capable platform

for implementing LMNL

LMNL supports overlapping ranges



In XML we fake it



This is the guy who once told Mike Kay that XSLT is fun

Neither is it a pitch (except in a general sense)

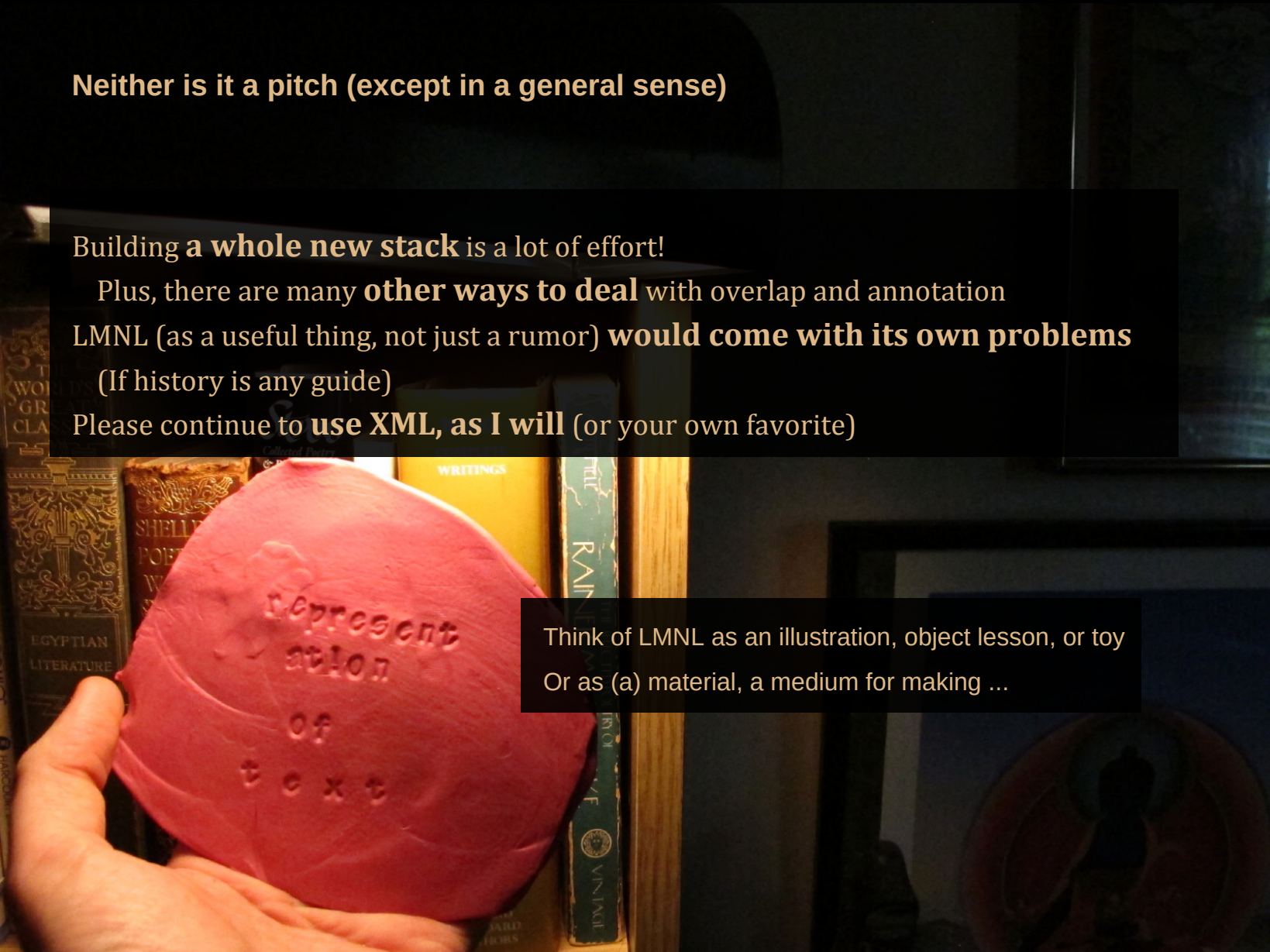
Building **a whole new stack** is a lot of effort!

Plus, there are many **other ways to deal** with overlap and annotation

LMNL (as a useful thing, not just a rumor) **would come with its own problems**

(If history is any guide)

Please continue to **use XML, as I will** (or your own favorite)



Think of LMNL as an illustration, object lesson, or toy

Or as (a) material, a medium for making ...

LMNL Applications

Query Analysis Transformation

In theory

Because ranges in LMNL documents have no hierarchy, any (all) hierarchies are implicit

Any hierarchy may be interpolated when needed (and expressed in XML)

Arbitrary overlap (ranges overlapping ranges of the same type or family) is fine

Heterogeneous document description and annotation is possible

So is dynamic (auto-) tagging (prior markup does not interfere)

Visualizations and heuristics can expose structures or lack thereof

In practice

This is essentially a solo research project ...

(so progress is fitful)

... serving mainly as an opportunity for reflection ...

... *and yet* ...

Finally, pudding

Until 2011, my explorations all used XML (milestones) to encode LMNL

This is cumbersome and difficult, even while you learn a lot about processing XML

In the meantime I learned not to worry about “overlap” so much (as XML tools have improved)

Eventually (after others had done the same) I figured out how I could process the syntax

(LMNL syntax parsers or processors had already been implemented

by Gavin Thos. Nichol, Jeni Tennison, and Matt Palmer at least)

By thinking of parsing as not a series of events

But a sequence (pipeline) of mappings from syntax to (emerging) object model

Effectively, using a series of XSLT “refinements” to “compile” LMNL syntax

Building an XML-based (standoff) representation of a LMNL document

(Suitable for further processing)

Luminescent

XML into LMNL

LMNL to XML

LMNL to HTML/SVG

Heuristics / Analytics

XSLT, under (your choice)

Cocoon

XProc (Calabash)

XQuery (BaseX)

Download at github.com/wendellpiez/Luminescent

Viewable at cocoon.lis.illinois.edu:8080/lis590dpl/wapiez/Luminescent/lmnl

Lessons of LMNL

XML excels when you need a high degree of consistency,
especially over large amounts of information
with the disadvantage of having to define everything up front

In comparison to XML, LMNL is flexible and permissive
and correspondingly touchy in production (lacking validation guardrails)

But well-suited for small- and medium-size experiments including one-offs
Markup first, then model

Potentially at scale too (given better tools and interfaces)

Plus, LMNL offers a variety of interesting possibilities
still unexplored

Things difficult in XML are easy in LMNL

Things easy in XML are difficult in LMNL

A solution: use both together?

Complexity of the problem domain does not disappear, but it moves

Markup can be simple and spare, or more complex, ornate and promiscuous

— For complex tagging we need a structured editing UI

Some applications become truly easy

The cost has been in the time to build tooling

This presents the developer with a dilemma:

Build generic utilities, or focus on particular demonstrations?

References and resources

Layered Markup and Annotation Language

(Historical) specs at `www.lmnl-markup.org`

With grateful acknowledgement to Jeni Tennison, Gavin Thomas Nichol,
John Cowan, Steve DeRose, Alex Czmiel, Paul Caton, Matt Palmer, Gregor Middell
(and owed to many others) for all their efforts

Along with the collegial support of Sperberg-McQueen, Huitfeld, Witt, Durusau, Durand, and many more

Luminescent

XSLT pipeline for parsing LMNL syntax into xLMNL, plus processing (in XSLT)

Capable of filtering, querying, XML extraction, display (HTML and SVG)

Requires XSLT 2.0; runs inside XProc (Calabash), Apache Cocoon, or BaseX (so far)

Open source at github.com/wendellpiez/Luminescent

Running on Cocoon at cocoon.lis.illinois.edu:8080/lis590dpl/wapiez/Luminescent/lmnl