Three Questions and One Experiment

On Data Modeling in the Humanities

Wendell Piez

Workshop on Knowledge Organization
and Data Modeling in the Humanities

March 14-16, 2012

Brown University, Providence RI

Sponsors:

Center for Digital Editions, University of Würzburg

Center for Digital Scholarship, Brown University

Women Writers Project, Brown University

Funded by:

National Endowment for the Humanities

Deutsche Forschungsgemeinschaft

Brown University Library

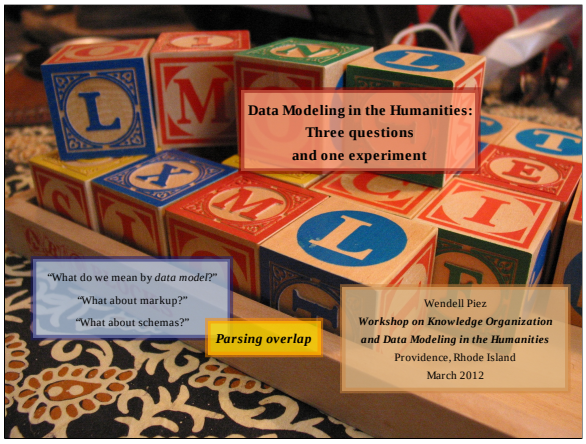
1 Three Questions

It is interesting to note, as we reflect on three days of conversation on Data Modeling and Knowledge Organization (attendants at the event will soon see how far these notes diverge from the remarks I gave *ex tempore*), how the topic of "data modeling" is coming, at length, to be a common reference point for the many disparate approaches to work in the digital humanities. Nearly twenty years ago, when I started work in this field, the term was hardly used, although it was already apparent how computing humanists (as we called ourselves) could and did readily find something in common between what, for example, markup and text encoding specialists did in order to formalize document description (we called it "document analysis"), and the analysis and modeling activity pursued by others who built, for example, resources in the form of relational databases, textual concordances, or what have you. So it doesn't come as much surprise that we now find this central activity is being recognized and, perhaps, in some way formalized ("modeled"). One of my themes today, indeed, is that we might consider this development and even consider resisting it, at least if we find it has any tendency to calcify and restrict our practices and methodologies, which are simultaneously undergoing so much change and growth. Yet at the same time, I think it is true that there is a "there" there. Although we live in its branches, this tree has a trunk, and its trunk is data modeling.

So I am after three questions:

- What is a data model, and in particular what do we mean by "data model" in the context of research in the (digital) humanites?
- What about markup?
- And what about schemas?

Readers who do not consider themselves practitioners of digital text-based markup technologies (even if they

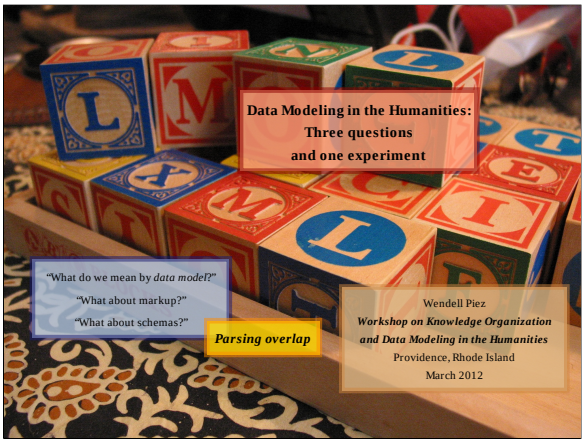


routinely read and author documents encoded in XML or HTML) may wonder why I pose the latter two questions, and what they have to do with the first. As an XML practitioner myself, I am bound to see each of these questions as nested inside the previous one. In particular, the second question is motivated precisely because as the digital humanities expands and we discover "data modeling" as a common concern and area of interest, we inevitably meet up with **markup languages**, an important technology for us where it has long had a central place. Yet it is also clear (and has been all along) that not all data modeling in the humanities uses, or must use, markup as an instrument. The place of markup itself, as a particular kind of text-based modeling technology, is therefore at issue at least implicitly whenever we take up the topic of data modeling in the humanities. Being convinced (for reasons that will become clear) that, for the foreseeable and perhaps indefinite future, markup will remain vitally important for us, I will consider markup languages in the hope that even if you do not agree with me on particulars, you find that it can serve as a useful standpoint from which to pose questions regarding "data modeling" in the large. Which leads us further in, since it also becomes ever more painfully apparent that in their current form, markup technologies have certain problems in application that make them unsuitable even for some of the modeling work for which they are intended – and indeed, as I will also explain, this has to do with a particular relation between currently dominant markup technologies (namely XML and its related specifications) and a particular form of data model.

One way of examining this question is to set out deliberately to design and implement a model for text on a different basis from XML. This is what has motivated my experiment with LMNL, a model (represented by a syntax, much in the way that XML gives us a syntax that implies a model) based on an interpretation of markup following quite different principles from XML's, enforcing different rules and so enabling the construction of a different sort of representation of text – a different sort of model. Which raises the third question, inasmuch as such a radical departure from the way we have used and designed document markup then poses significant practical and theoretical challenges, all of which lead to that abstraction we call the **schema**, as a modeling technology. In brief, without out fashioning our model of a document or text as a single hierarchical arrangement (of the sort that XML

imposes but LMNL leaves aside), we can't have a schema; yet we find we need one even more than ever. So, what about it?

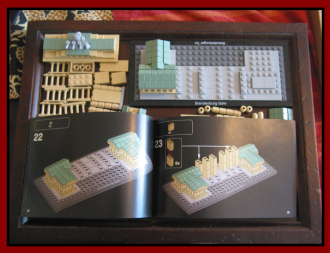
This nested-doll arrangement of questions provides opportunities for engagement at several levels. I am talking about data modeling in the abstract, as a topic of interest to everyone doing digital humanities, who is therefore involved in data modeling, or in using or exploiting data models or (indeed) models or media conceived generally. I am also talking about markup technologies, and hence about XML and its limitations. And I am talking about new approaches to conceptualizing models of text and information generally that may enable us, going forward, to use markup – and thus to model information – in new ways.



What do we mean by

data model?

LEGO® Architecture *Brandenberg Gate*
model designed by Adam Reed Tucker



Mauer 1985 diorama by the author

For LEGO® see LEGO.com

2 What do we mean by "data model"?

We might best address our first question, "What is a data model?", not by way of formal definition (since that will prove to be part of the problem) but by way of examples, or even (as depicted in the illustrations here) by analogy. We might loosely say that a Lego model of the Brandenburg Gate (designed by Adam Reed Tucker for the Lego company and sold in the Lego Architecture® series) is a **data model**, at least inasmuch as the Brandenburg Gate itself (built in its present form in 1788-1791) is an historical artifact of interest to scholars. And I refer to a *particular* Lego construction, a particular instance of this model. Yet at the same time, those of whose work entails data modeling might respond that such an instance does not actually map very well to what we call a "data model", which we consider to be an abstraction capable of supporting a set of such instances: we might think here of the "XML model", by which we probably mean XML's tree of elements and attributes, or of the "RDF" model, by which we mean triples representing nodes and arcs, linked together in a network or graph. Here it may be that the closer analogy is to Lego itself, that is the specification of a set of pieces or bricks, designed to fit together in such a way as to allow us to build whatever particular model we like – the Brandenburg Gate, the US Capitol, a vacation home, the Death Star from *Star Wars*. We could talk about the "Lego model", in order to refer to the particular way Lego pieces are designed to stack and interlock, as an underlying model for the models we build out of Lego.

Yet also there is an in-between, namely a generalized model of the Brandenburg Gate – the particular set of pieces, along with the instructions on how to combine them, that underlies a particular instance. (Were one so inclined, one could acquire several of these and have several Brandenburg Gates, all the "same model".) Or, if we step back again, if we conceive of something that Lego does not actually offer (but which we might fairly

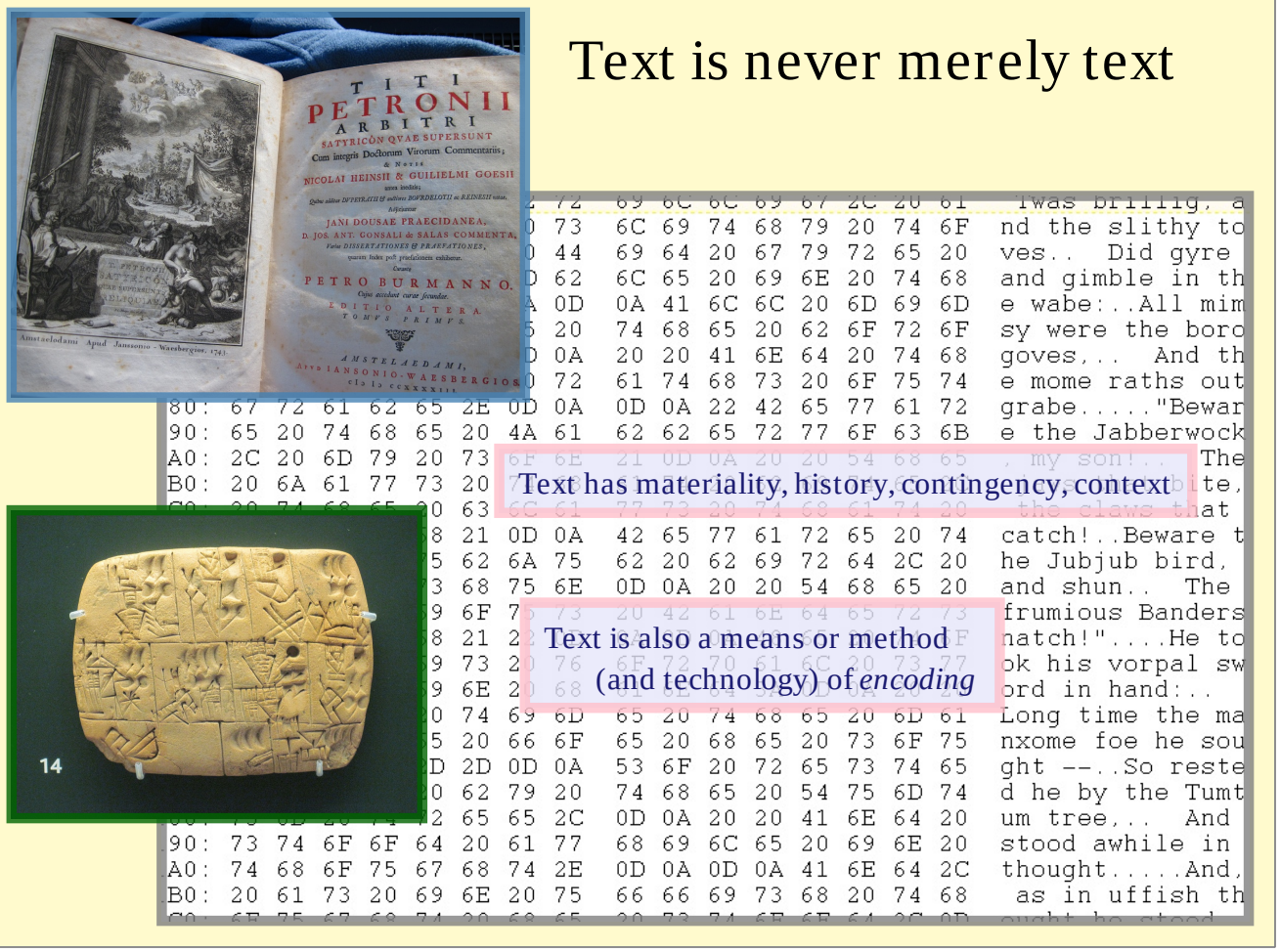
invent for ourselves, if we chose), we can imagine a set of rules for building “Lego models of late eighteenth-century neoclassical architecture”, such as the Brandenburg Gate and others of its type (whatever we define that to be), real or imagined. Such a type is a kind of model. And indeed we all know of the way we do this in the digital realm, with (for example) schemas that constrain and codify families of XML documents intended to be (at least for purposes of processing) interchangeable.

Finally, all the way further on the concrete end of the spectrum, we can allow that there may also be particular models representing not only particular artifacts like the Brandenburg Gate, but representing them in particular states or times and places. (A TEI document might represent a particular copy of a text.) My diorama pictured here, *Mauer 1985*, represents the Brandenburg Gate in Berlin, but not as it is today.

Of particular note also is how, in order to build this last item, I had to go outside the constraints of the Lego model, and supplement it. (Since I didn’t have enough Lego pieces to build a wall, I used alphabet blocks. And being unable to make a sign out of Lego, I printed one on an index card. If you look at pictures of the Berlin Wall with the Brandenburg Gate behind it, you’ll see my sign doesn’t look much like the original, even apart from the reduction in scale. This is because I had to do the best I could with the modeling technologies available to me. In this case, the whimsy of the model’s departure from the historical fact is itself part of the model’s representation – an important fact, of which we should take note in passing. This Berlin Wall is neither so serious nor so formidable as was the original, and this difference is as important to the “meaning” of my model as the similarity.) And this too, finally, is characteristic of the models that we build when pursuing research in the humanities: there can also be an improvisational aspect to a model, a kind of “bricolage”, as we assemble and bend technology to our purposes, and there can moreover be a difference from the original that is significant and purposeful.

Moreover, it is important to understand how improvisation at this higher level is simultaneously enabled, and frustrated, by rigor and regularity at the lower lever. The system at the lower level inhibits me from working outside its capabilities; but it is also what permits me to build a model at all.

I submit that all these cases of representation, from concrete to abstract, are fairly called **models**, although they have a logical relation of dependency between them, as each sort of model depends on the ability to specify and build a model of a more generalized kind. Similarly, models also imply higher-level models, insofar as they participate, as members of classes, in higher-level significations. So every XML instance, as an encoded document, both depends on character encoding and the rules of XML syntax, and it implies a schema, more or less coherent, even if it does not call one explicitly. Every schema, in turn, implies an organization and set of relationships among more elemental particles (in XML, elements, attributes and string values), just as schemas themselves can be classed according to the kinds of models they describe and the modes of document or data description they deploy. And the XML model itself is predicated on certain modeling capabilities, namely the representation (in serialized formats and in a computer's working memory) of labeled and typed data, along with algorithms that process directed graph structures. The same sort of generalization is possible for other kinds of data models, whether graphs, relational tables, or text formats. *It is modeling all the way down.*

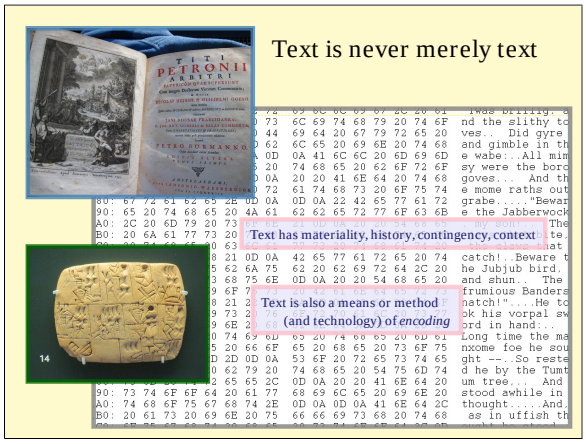


3 Text and “plain text”

All this is complicated greatly by the peculiar nature of what is arguably the core technology of humanistic study, namely **text** itself. I say “core” because even when we do not study texts (as art historians, archeologists, musicologists or others), we use texts as an instrument of study, and we represent, manage and share our studies with texts. That is, even when we are not making models *of* text, we are frequently making models (or at any rate representations) *with* text.

Here we run headlong in something humanists all know very well: *text is never merely text*. Even if we define text as, or at any rate identify it with, a technology of **encoding** (which means, in effect, that it in its essence, it is an abstraction, a model of a kind of information, and that any particular text is only a model of a text) nonetheless, as students and scholars, we cannot escape the fact that text is also always material and embedded in historical contingency. Thus, it always brings with it a kind of penumbra of associations, including the associations due to it because we know something about how it was (or is) expected to be used. So, *what we know about text is bound with its processing, use and reception*, and text always testifies to and gives evidence about **intentionality**, in the scholastic and larger sense that **Allen Renear** has used. Moreover, in order to construe the text in its fullness we sometimes (usually or always) have to bring more to it than mere knowledge of it *qua* text, that is as an encoding methodology.

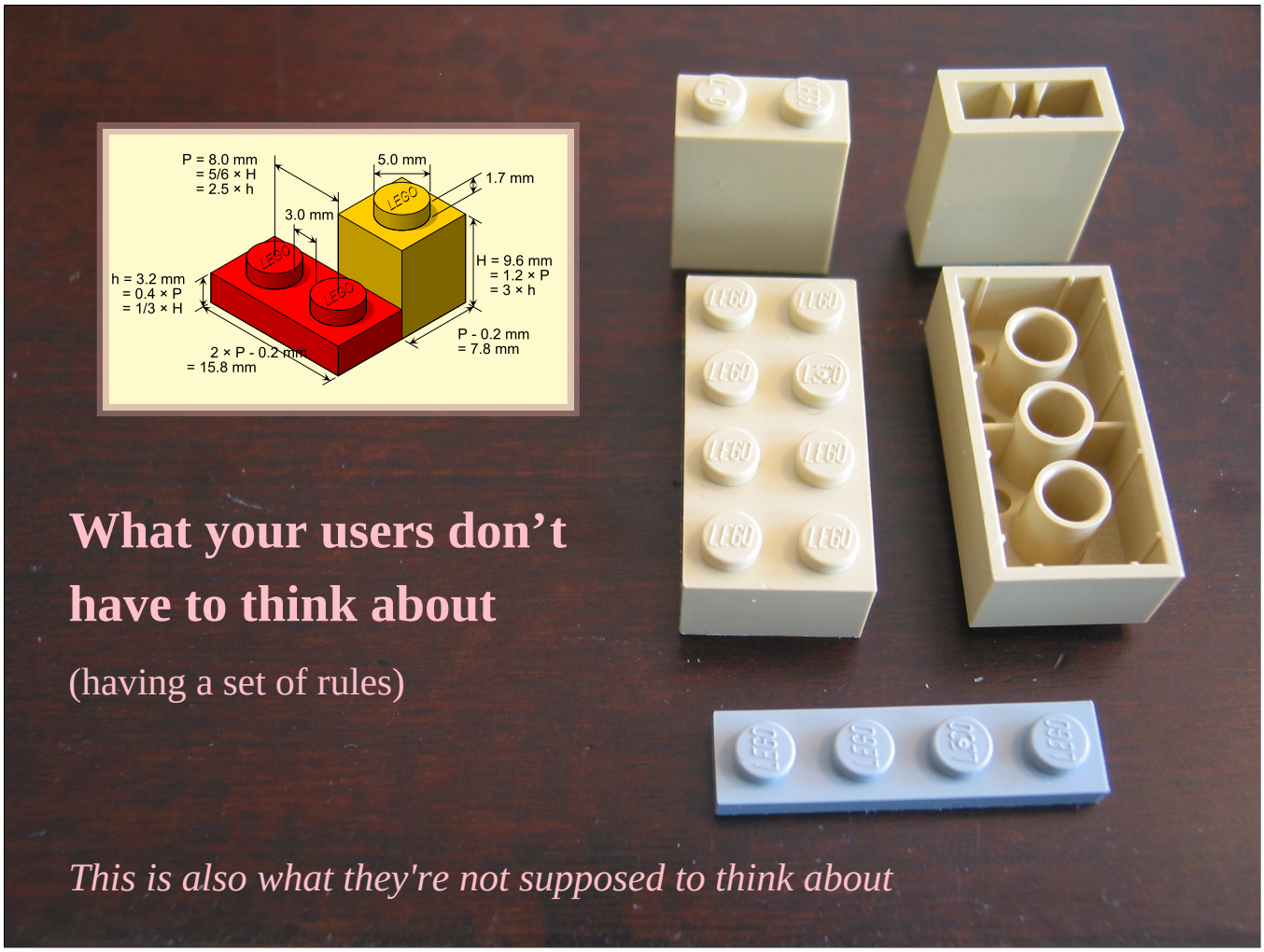
Yet at the same time, this aspect of text is always subject to loss and decay, and to deal with this, text itself deploys its own ameliorative methods, and so certain aspects or features of this larger context are themselves also frequently encoded in and by a text – so we learn that even the earliest cuneiform tablets, for example, present both “text” and “markup”, at least in the sense that many of the markings are evidently there to refer directly to



other markings, serving to gloss, qualify, contextualize and describe them.

Similarly, as media develop, this layering increases, even while the enabling technologies grow more complex, systematic, and capable, so that by the time we get to print culture, we get texts glossing texts glossing texts. For example, in the frontispiece of an 18th century edition of Petronius, we have not only a typographical elaboration that indicates, in both implicit and more or less explicit ways, all kinds of information about the text – its status, culture, associations – going well beyond the facts of it and its publication (what we might today call this book's metadata). In this particular case, in a maneuver we might have called "post-modern" were this not an edition of 1743, we even have a depiction, in the form of an engraved print, of a fanciful scene of a ruin (just as the text of Petronius is a ruin), in the midst of which is a slab on which is inscribed – a text.

Nor is this particularly uncommon – which fact offers its own context when we look at comparatively spare modern renditions of text (which are designed – modeled – in the context of these more elaborate early examples). The history of the reception of texts becomes part of the text: this is not a new idea. And even as we move into the digital realm, we can leave none of this completely behind. If anything, *analog technologies anticipate and foreshadow data modeling in the digital*, which is only different in that it is even more formalized and more explicit.

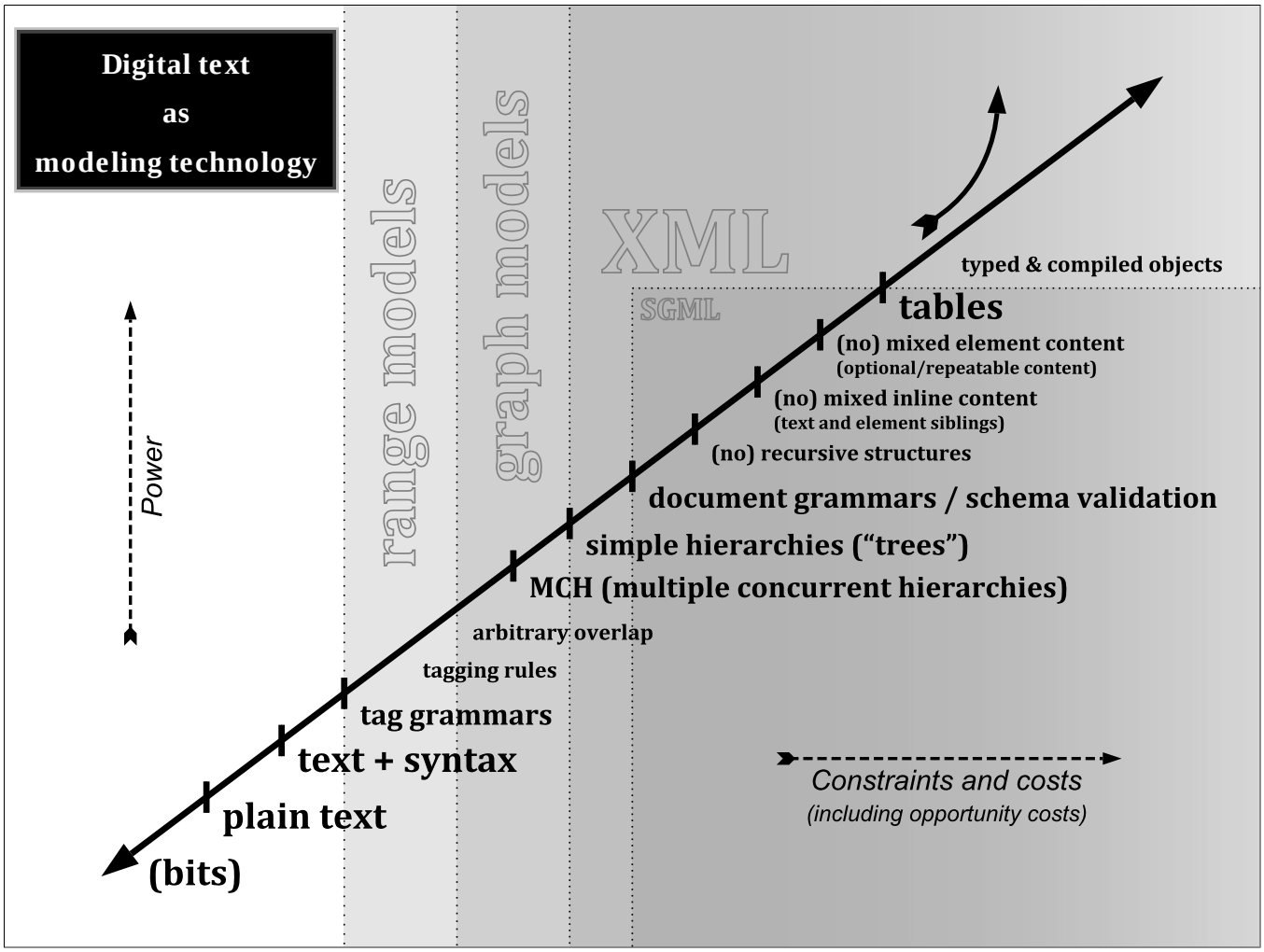


4 Implicit capabilities

So the modeling technology we design, in order for us to enable others to build models, is itself a model. And if it is well designed, it will be simple enough to be easy to use, versatile and adaptable, yet also provide capabilities for the particular modeling purposes for which it is intended. Complexity then emerges from the application of simple principles of relation and connection among components, which start as independent entities but are soon related as constituent parts in a larger whole. In use, these *capabilities are implicit*: when building with Lego, we neither need, nor want, to be thinking of the exact dimensions of the parts of a Lego brick, with its dots and crannies, and how they work to make it possible to connect the pieces. (Not being very well informed about Lego, I do not even know the technical terms for the operating parts of a Lego brick, although I do not doubt they are known to the designers and manufacturers of Lego. Yet I can still build with Lego.)

I can, it is true, now look up both terminology and specifications on line. Indeed, Lego enthusiasts may and do want to know the facts of this lower level, should they seek (for example) to extend Lego and achieve **interoperability** with Lego "out of the box". Yet for ordinary purposes, we should keep in mind that it is a sign of success when we are able to work oblivious to the underlying design of our modeling technology.

Thus when designing a modeling technology, we want our users not to notice (not to have to pay attention to) how it works. That is, we want our audience not to see the model as a model, but to see *through* the model to the thing being modeled. (They are studying the cuneiform tablet, not its electronic surrogate.) Or is this so? Perhaps we should also consider that this is precisely what might differentiate data modeling in general, which seeks only utilitarian purposes, and data modeling for the purposes of representation, criticism and consciousness.



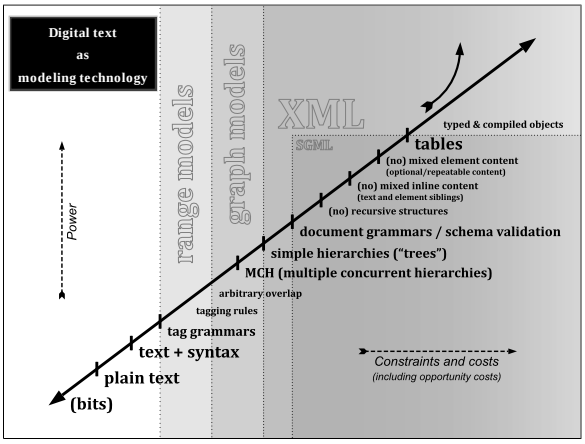
5 Digital text, plain and otherwise

Given this idea, we can turn to digital text technology and consider how applications of digital text can thus be seen on a spectrum **raw** to **cooked**, unconstrained to constrained. "Rich", on the low end, maybe, to "accessible" and "refined" on the high end. By **digital text** I mean a sequence of characters represented in some binary encoding in the computer (and thus, we should not fail to remember, already built on layers of abstraction, including the map or table of the characters themselves, and the particular encoding by which that mapping is realized, e.g. Unicode and UTF-8).

Why do I have **power** at the high end, rather than down at the bottom, where weak (or no) bindings to processes, we might think, makes us more expressive and therefore more powerful? Because I think we can and must distinguish between "potential power", in an "uncommitted", not fully formalized system of semantic labeling, and the actual power we get once we bind a digital text – be it input into a program, or indeed a program's own source code – into a processing framework (its **operational semantics**). By **bind** here I mean the action of mapping information and its parts dynamically into in-memory structures capable of manipulation and rewriting by means of algorithmic processes. Of course, even to call the potential power of free, unformed text "uncommitted" may be to say too much, since it may well (and we hope it does) be committed to a semantics or "meaning" in its linguistic denotation and connotation, absent any application of automated processes. This much meaning we get for free, as it were, simply by virtue of a text's being text: arguably, if it fails to encode anything at all, it is not a text but only a random string of characters. Yet it is only when we formalize and systematize the syntax of digital text, and moreover wire up vocabularies or sets of keywords that in turn can be "compiled" into lower-level operations, that it becomes tractable to automated processes. At this point, digital text's capacity

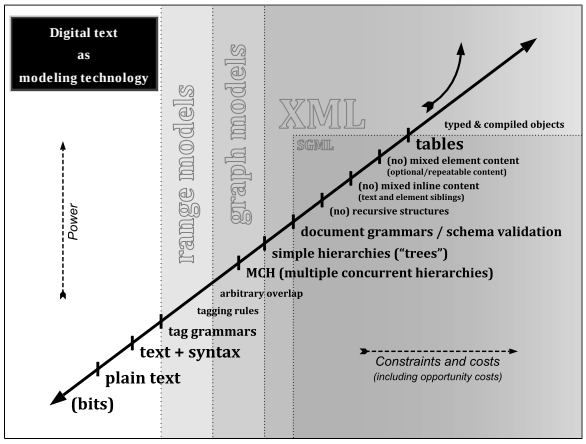
for modeling moves from implicit and potential to explicit and capable, as *constraints are enabling*; and the more elaborated and complex these formalisms are, the more specific the constraints, the more complex and dependable the processing can become.

Yet the paradox is that the more we wish to actualize or drive processing, the more we have to make commitments that forestall some possible interpretations of a text in favor of others. So we see here the same need for balance, in a capable text-based encoding system, between a simple formalism on the one hand, and on the other, the complexity, potential and actual, of its applications. Moreover, we can see that we have a need here for **standards**, inasmuch as if we wish to share our texts and our applications of them, we need to encode them in ways others are prepared to use and recognize – in ways they use themselves.



Extant repositories of digital text show exactly this distinction between plain text, with no particular syntax or markup protocol, at the low end, through forms of text and markup observing more and more rigorous (and rigorously enforced) sets of rules at the high end. Information locked up in databases and in proprietary word and document processing software show, moreover, the price for living at those heights – the data is not portable, and goes stale after a few years, as the infrastructure required to process them evolves. For purposes of longevity and platform independence, we discover, it is often better to export the data into simpler and more legible forms – lose power, trading it off for greater potential.

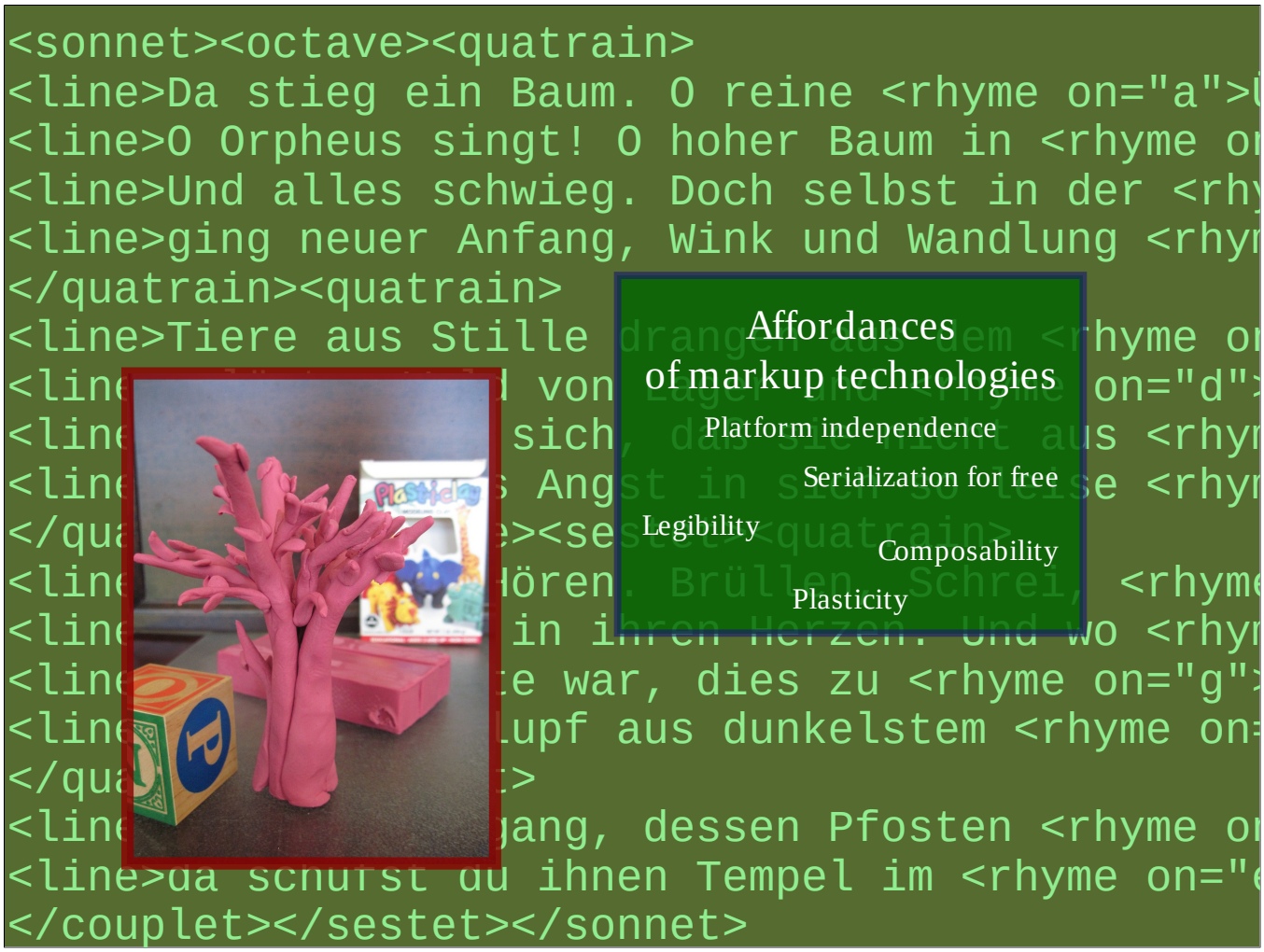
This gives perspective to the importance of the emergence of the XML standard in the closing years of the twentieth century. Not only did it unlock data from proprietary applications, it also eased the requirement made by its parent technology, SGML, that every instance (document) be bound explicitly to a schema (document type). That is, we could not even have an SGML document without knowing not only what syntax, but what markup vocabulary we would be limited to: and this imposed costs that were in many cases prohibitive. In contrast, **well-formed XML** (that is, data marked up using XML syntax considered even without its particular vocabulary) is more generally expressive, and requires less overhead to produce, than does XML-plus-schema. Yet at the same time, its commitment to nested tags at the syntax layer limits an XML text to a single unitary organization, i.e. a tree hierarchy – deliberately and by design,



as it enables the operation of efficient processes, on all XML documents, that navigate, interact with and manipulate tree-shaped (simple graph) structures.

Noticing the tradeoffs here, we can posit that for any particular problem, we need to be able to choose the appropriate place along this scale – assembling raw texts uncommitted to particular formats, and yet, when needed, operationalizing our information by committing it to formats and to the models they imply. Yet I submit that this is only part of the capability we need. The problem with texts at the high end is that they are inevitably unsuited for operations other than those for which their formats are designed. The commitment comes at a cost, not only in the trouble and expense required to produce, validate and maintain our data sets, but also in what we have decided we cannot do with them. Consequently, to be able to commit texts to particular formats, more or less constrained – and so to use text as a modeling technology with greater or lesser affordances of particular kinds – is not enough. We also need to be able to move up and down the scale easily: to process raw data, by combinations of means including automated, semi-automated, and hand methods, into more highly controlled forms; but also to convert data that has been formatted for one set of operations easily into formats for others, or for no particular operation at all.

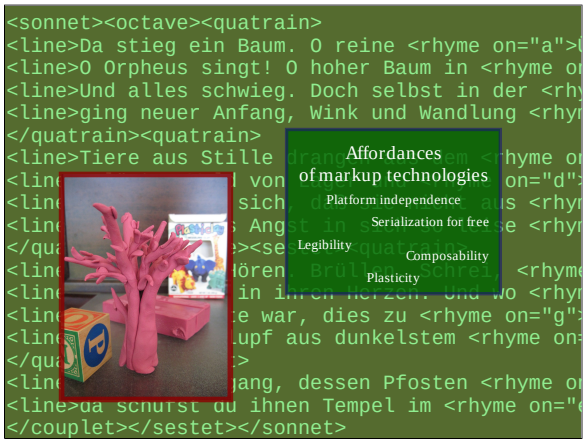
It should be clear from this how important I consider markup technologies to be, inasmuch as they constitute an application of digital text between more highly constrained forms at the high end, and relatively unconstrained forms at the low end. Moreover, as I suggested earlier, the line between text and markup is actually surprisingly difficult to draw, and (at least in some loose sense of the term) markup may (and generally will) be present all the way at the bottom of the scale from implicit to explicit structure – while at the high end, even executable code is written in codified and constrained text-based formats before it is compiled, and the plain text editor is a basic tool for every programmer.



6 Affordances of markup technologies

Nevertheless the role of markup technologies in particular – by which we mean, specifically, the use of codes embedded within the text stream in order to represent meta-textual information, whether it be declarative and **descriptive** or process-oriented and **procedural**, remains controversial in some quarters. Given this, it is probably worth considering not only what distinguishes markup as a technology, but what its particular affordances are, and why we like it.

Markup technologies combine certain virtues. Since they are text-based, they are in principle (and generally in practice) more accessible than opaque binary formats, which are illegible without specialized software for reading and interpreting them. (Being codified in ubiquitous standards such as US-ASCII, ISO 8859 and Unicode, plain text processing is supported by a commodity market for tools.) So text-based markup is **legible**. Insofar as a markup syntax is also systematic and formalized in such a way that it maps straightforwardly to a data structure (as XML does), this means that along with a legible and accessible layer for our proposed **semantics**, we get a **serialization** of that data structure for free: something very useful for purposes of storage, maintenance, development and exchange. Moreover, due to the way markup syntaxes are deployed and chunks of information embedded in files, texts using markup are, or at least can be, **composable**, in the sense that coherent chunks of markup and marked-up texts can be readily be copied from one resource to another. So it lends itself to hand-editing, when necessary. And perhaps most importantly, since markup is essentially a specialized application of text itself, its principles and application are already apparent, without much special instruction, to the literate user, at least as long as she or he is not intimidated by code itself. To read and write is already, in some sense, to know how to use markup.

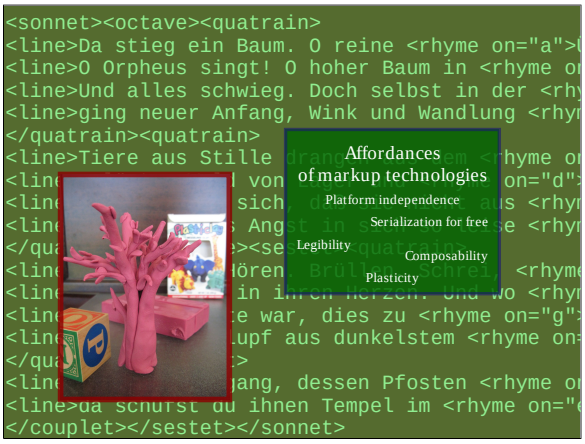


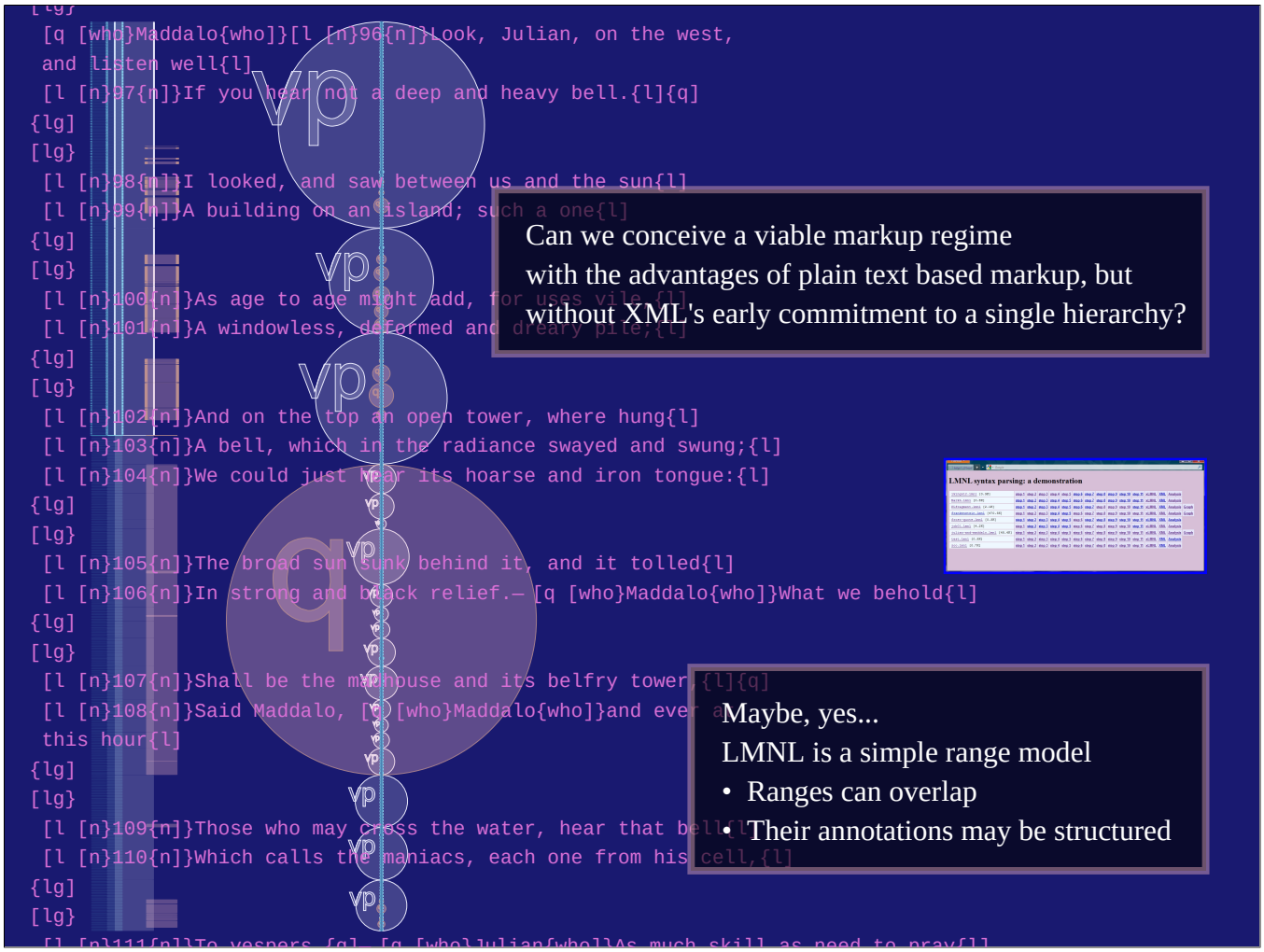
All of this makes markup an especially useful and versatile modeling technology, a kind of material neither too rigid nor too soft. We can think of it as a kind of clay: for some things, perhaps, ungainly, and not necessarily, in itself (that is, without the support of further infrastructure), very useful for modeling tasks that are very large or complex. Yet clay is used by artists not only to model works to be made of clay, but also works that will be produced in other forms, as a clay model may be cast in bronze. And in fact markup technologies are used in exactly this way, to model data structures and information organizations that will not be processed by operating directly on the marked-up text itself, but only at a step removed. (This is what XSLT and XQuery do when they process an in-memeory tree structure, rather than the raw markup itself.) In this architecture the markup is **parsed** into a more optimal internal form (tables or graphs) to support manipulations more efficiently than plain-text processing can be.

Yet there is a complication to this story: currently dominant markup technologies, including not only XML and everything based on it, but also other families of text-based representation of structured data (wiki markup or “markdown”, JSON, what have you), all work by mapping implicitly to a single directed graph or “tree” structure. (Nor are serializations such as RDFa, or range models of documents represented using standoff markup, exceptions to this, inasmuch as they model simple tree structures in order to model other structures.) And while tree structures are well understood and capable of supporting very complex manipulations, not everything we wish to model naturally takes the shape of a tree or can even be gracefully represented by a single branching organization. We know (and within the digital humanities this has been recognized at least since 1991, when **Michael Sperberg-McQueen** wrote about it), texts (however defined in whatever view of it) can have more than one hierarchical organization at a time. By optimizing our data structures (our models) as tree structures, we must make a commitment, observe a constraint: this (and not that) is the (tree) structure we will resolve and recognize in our processing, and everything else must be represented by means of it. Thus we have systems which can straightforwardly present a text’s grammatical structure, but not its meter and prosody, or its logical organization, but not its page structure or its narrative organization. And the cumbersome workarounds that let us do more are difficult to engineer and use.

So there remains a significant gap along the spectrum between free-form, unconstrainted text at one end, and markup technologies representing (first) trees, and (then) more elaborate or generalized data structures representable by trees, at the other. We might well have a markup syntax that did not impose XML's rule that only the most recently open tag can be matched with a corresponding close tag. Such a markup syntax would implicitly permit the description of arbitrary structures without their having to be accommodated to a single tree or even any tree or trees at all: this constraint, at least, would have no hold on them. Yet it would be regular enough to support processes that could be used (much in the way XML uses its trees) to expose structures they described to other, higher-order processes. Such a markup or modeling technology would be able to represent both what we call **multiple concurrent hierarchies** (MCH), and **arbitrary overlap**. (MCH is actually a subset of the arbitrary overlap problem, as the latter also comprehends overlap between structures that relate to no others in any sort of hierarchy, single or multiple.)

One interesting effect of a markup methodology like this is that it could counterbalance, perhaps, the tendency in the XML community to consider the definition and codification of markup schemas as its primary work, above even the development, design and application of markup itself. XML (and SGML before it) has taught us much about schemas, document manipulations, workflow issues, and the rest. But texts are richer and stranger than XML has been able to comprehend, and it might be that by turning our attention back to modeling text with text – rather than simply fitting our texts to the models of it we know how to make – we will not only come to understand it better, but also build better tools for doing so.



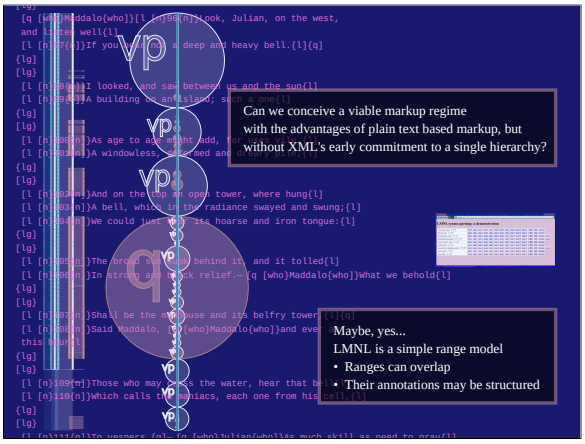


7 LMNL: a demonstration

Thus, in my own research, I haven't been quite willing to forsake the idea of markup considered generally, even as I have had to contend with XML's limitations. In 2001, **Jeni Tennison** and I proposed using markup in much the way I have just described, mapping it to a **range model**, which permits multiple hierarchies in effect by permitting any hierarchy, or none. In LMNL (**Layered Markup and Annotation Language**), textual information is conceptualized as presenting a set of ranges to be marked, named, classed and annotated, each range relating to the text directly but not necessarily to one another. (Whether ranges or families of ranges should relate to each other is left to a process to define, rather than being considered an inherent part of the model.) While it has proven challenging to develop parsing technology for this syntax – since parsers traditionally work top-down, and must assume a grammar, i.e. a single, unitary organization, not only for tags and text but in effect governing the entities or elements parsed – recently I was able to start doing this by considering parsing markup as an **upconversion**, the term of art that describes the interpolation of structures based on implicit information. In this case, as I am an XML developer, the interpolated structure is, indeed, a tree: yet this tree does not represent the document "itself" but only the marked-up text, that is, the text with its tags. A subsequent process then reads the tags out of this structure to identify the ranges being identified. Ironically, the range model itself, in my implementation, is represented in XML (it is, in fact, an application of what we call "standoff markup", which is to say markup that does not "mark up" directly but at a remove), which is perfectly suitable for me since I know how to process it using XSLT.

And – this sort of markup works, at least at the modest levels of scale (up to the length of a short novel) I have used it for. To date, I have the following applications working with arbitrary LMNL markup as input:

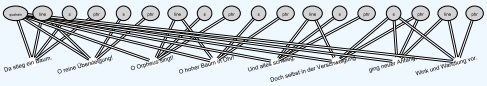
- An analytic routine is capable of detecting and reporting on instances of overlapping ranges
- Given a set of range names, XML can be extracted in which the named ranges are represented as (nested) XML elements
- With the help of libraries, we can display, render, convert into HTML, SVG, XML, or any format we can generate from XML....
- Well-formedness checking can report errors in tagging syntax to a user.



While these applications are being developed on an XML platform, they do not depend on it. (And I am interested in encouraging others to take the technology in different directions.)

If nothing else, these demonstrations show that overlapping phenomena in literary and historical texts are very interesting and well worth the trouble of representing. For example, I have made graphic illustrations of the prosodic features of poems (the way verse structure and grammatical structure interact, overlapping, to help define the rhetorical form of the poem), and of the narrative features of novels and poetry (for example, where discourse appears and how nested narratives are deployed). These applications speak for themselves. And yet, while doing so, I think they give only the barest hint of the possibilities for markup once we are able to use it to describe many hierarchies at once or no hierarchy at all.

Getting there from here



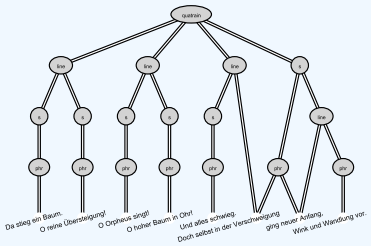
Small demonstrations are ... small

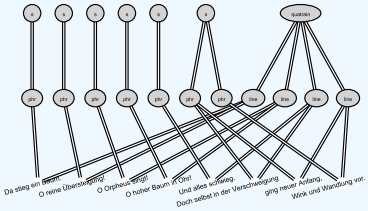
Range markup is very free form

Range processing may not scale up *(but why not?)*

How do we bridge to more powerful abstractions (graphs, Goddag?)

```
[quatrain}[line}[s}[phr}Da stieg ein Baum.{phr]{s]
 [s}[phr}O reine Übersteigung!{phr]{s]{line]
[line}[s}[phr}O Orpheus singt!{phr]{s]
 [s}[phr}O hoher Baum in Ohr!{phr]{s]{line]
[line}[s}[phr}Und alles schwieg.{phr]{s]
 [s}[phr}Doch selbst in der Verschweigung{line]
[line}ging neuer Anfang,{phr]
 [phr}Wink und Wandlung vor.{phr]{line]{s]{quatrain]
```





8 Supporting complexity

Yet, in keeping with themes we have considered, LMNL may yet prove to be difficult to process for some kinds of problems. (Different developers with whom I have spoken have different ideas about this.) LMNL works by using tags (or other indicators, embedded or not) to designate ranges, but ranges recognize no inherent relations between them except by virtue of the text they range over. And while we know documents or texts of interest for humanistic study are not single hierarchies, we also know that they are more than piles of ranges: they contain multiple hierarchies and even more intricate structures; to process these algorithmically, we will have to represent them in our models. A number of variants on graph structures, including the Goddag structure of **Claus Huitfeld** and **Michael Sperberg-McQueen**, have been proposed. It is to be hoped that a viable serialization format for such complex documents, such as LMNL syntax, might enable such work to move forward. But a great deal of work is still to be done.

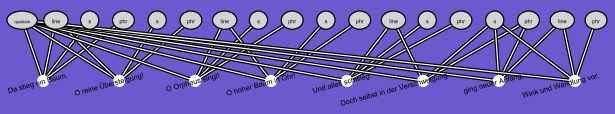
One difficulty is that once we allow free-form tagging of ranges, a markup syntax alone is not sufficient to indicate what the proper hierarchy of a document might be. In the example shown, where the sentence/phrase structure (marked as **s** and **phr**) overlaps the verse structure (marked as **quatrain** and **line**), a "naive processor" might try and infer a structure in which no proper "sibling" relationships can be inferred between sentences or between lines, since sentences are inside lines in some cases, and lines inside sentences in others. To reconcile this and model the document in the way we properly want (in which all sentences are together on a level, while all lines are together on a different level) will call for additional information regarding the proper relations among these ranges, which is not given in the markup as such.

What are schemas for?

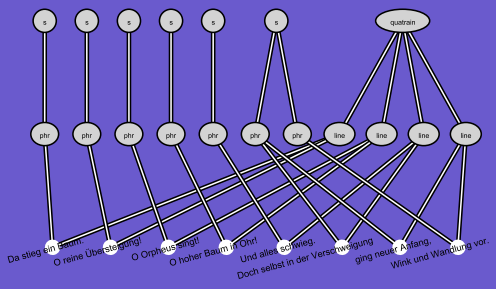
Three different kinds of *transformation*

- Validation (returns a set of errors/ warnings)
- Data annotation/ enhancement (returns an amended document)
- Configuration of tools, workbench

All these can help solve our problem



```
define poem (s+ ^ (quatrain|tercet|couplet|line)+)
define quatrain (line{4})
define tercet (line{3})
define couplet (line{2})
define s (phr+)
```



XML offers a platform

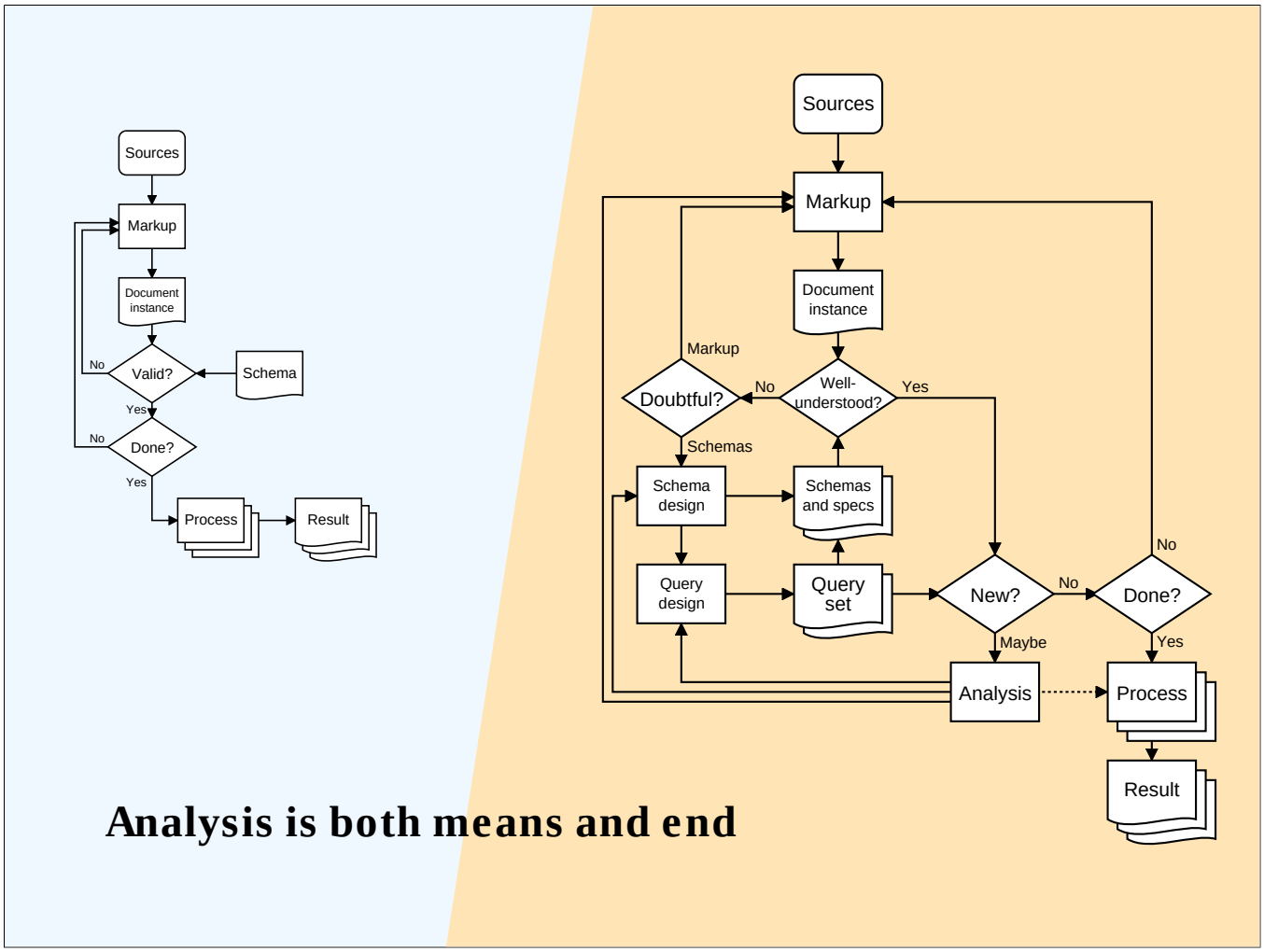
- Document grammars and constraint languages
- Datatype specifications
- Transformation technologies
  - Conversion in/ out
  - Diagnostics
  - Display
  - Proof-copy and testing

International Space Station, in orbit

9 Markup schema as mediation

Fortunately, our experience with analogous problems in XML, even with its single hierarchies, points a way forward. In XML, schemas have a number of important functions. As a codification or formalization of an abstract model, a schema provides an input to processes that effect special kinds of **transformation** over documents. Schema validation, for example, answers the question whether a document conforms to the schema. (Is this particular XML model an instance of a more general model?) XML schemas can provide data typing information; applying a schema to a document yields an enhanced document in which the component parts of the model are assigned particular properties and can be processed in particular ways. Finally, schemas have always been used in markup systems to define constraints for tools and user interfaces, optimizing the production and modification of markup instances, inasmuch as particular models can be manipulated by particular tools. And these are precisely the sorts of operation that we will need over range models in order to use them more efficiently and optimally. In other words, schema technologies are just what we need to exploit information in a LMNL document that is only implicit – to push it up the richness-accessibility slope.

Yet the particulars of how this is to be done remain to be explored. With less constraints over our markup – no single hierarchy, and perhaps no hierarchy at all – we can no longer apply schemas as grammars, imposing single hierarchies of elements: they will have to be something different. Neither a mathematician or computer programmer, I am not qualified to design and implement these solutions, or do more than guess which approaches may be computationally tractable. In other words, we have here an area of research not only in data modeling in the humanities, but in computer science. And the problem set is rich enough to convince me we will make progress in it.

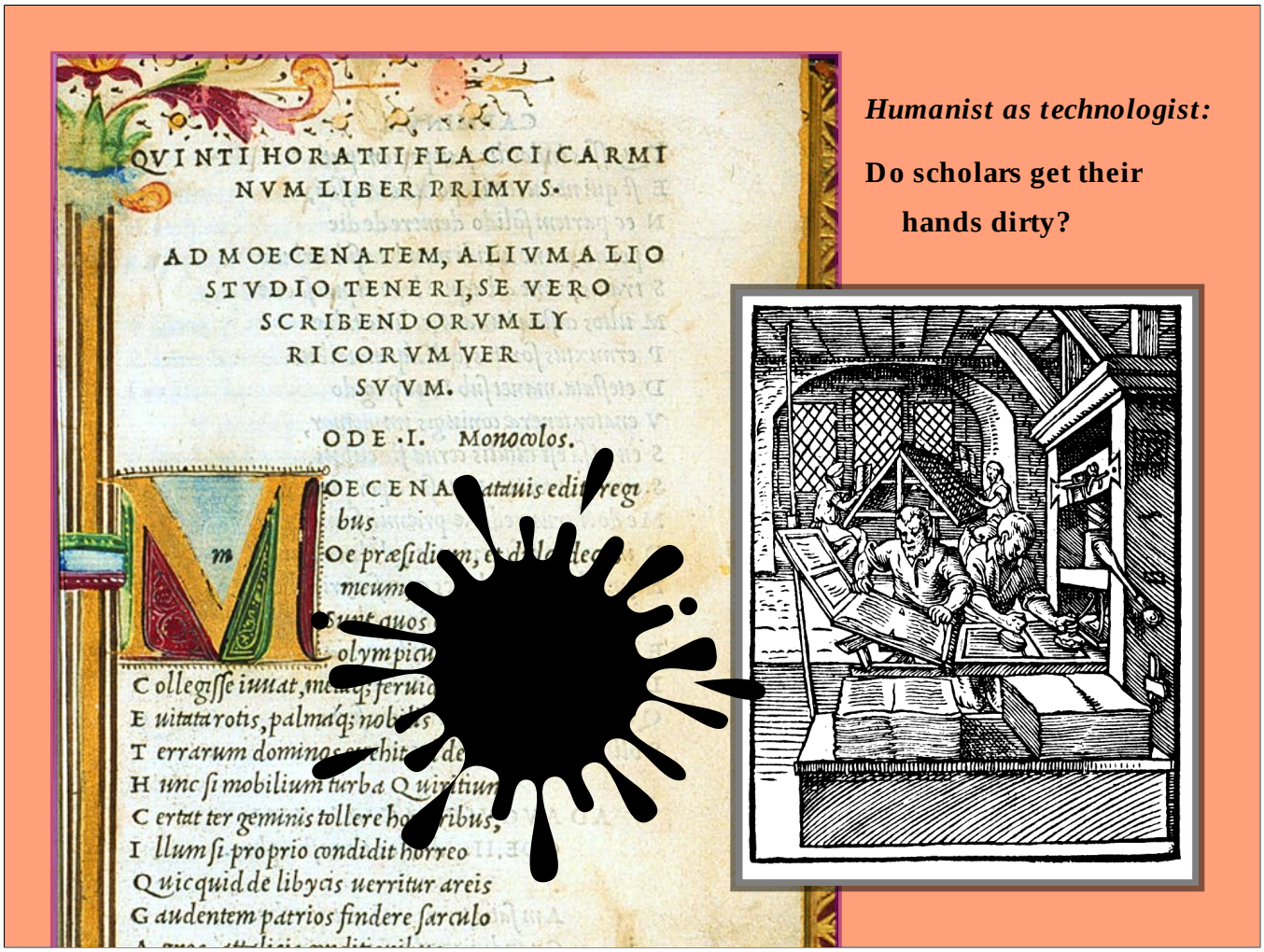


10 Towards hermeneutic markup

Ultimately this means that for modeling in markup, both schemas and instances need to be **plastic**: as representations and as abstract models of representations they should be objects of study, emendation, alteration, critique and contest even while they may also serve as models of pre-existing artifacts, the texts or other things of which they serve as models, which we are thus able to describe, document and study, even if not (or not in all their particulars) to define. So a particular markup language, description or schema can no longer be taken as a given.

This is not, to be sure, the way markup systems, as document processing systems, have been traditionally designed. Their design reflects the functional requirements of document production processes organized from the top down, for purposes of automation – processes for which tree structures (for the most part) have proven well suited. Outside certain niches, these systems have had no special need to address thir design to serve the needs of research and discovery in themselves, except insofar as requirements for “search and retrieval” can be codified in advance.

In contrast, the proper organization of tasks and operations for humanistic research (or for any open-ended research) must be one in which the models themselves are a subject of scrutiny, and the processes to be performed on the data are not fixed in advance, but evolve. In other words, opportunities for critique and evaluation not only of work products but also of the technological means, are essential to the proper functioning of the system as a whole. And this means, moreover, the process itself can be dynamic and receptive to new inputs and new ideas, which in turn makes it more robust. Automated and machined operations are then much more than production tools: they also provide a framework and a set of opportunities for understanding and expression. *Data modeling is both a means, and an end in itself.*



11 Getting our hands dirty

In summary, we are faced with something of a dilemma. We wish to enable the work of others, and we wish to do the work ourselves. But when can the work begin, when there is so much work on the work to be done? As scholars and students of the humanities, do we also have to be technologists, computer programmers, designers and architects not only of arguments and theories, but of the media in which they are presented and of machines in which they may applied and demonstrated?

The paradox here, I think, is that the answer is yes, we do have to get our hands dirty; yet among the important purposes of the work is so that we should not have to – or that we should not have *only* to. In fact, this problem is as old as scholarship itself, inasmuch as academic study has always been implicated in questions of class, wealth and leisure, social status, ideologies of "the life worth living" and how it is achieved and shared. Yet it is worth recalling that the term **humanist** itself, construed narrowly, refers to a group of technological innovators, who catalyzed and facilitated the European Renaissance through innovations in print media, developing presses and type faces, methods of printing and book distribution, and international networks of collaborators. Neither should we forget, more generally, that the power of technology itself comes from the way it codifies and tends to enforce specialization and division of labor. Certainly part of the point of my dirty hands is that yours do not have to be so ink-stained, and much of the effort of the humanities has always been devoted to creating works (now including digital works) that are clean, complete and beautiful in themselves, like Aldus's Renaissance edition of Horace. Yet at the same time, the projects, systems and institutions in which this work is done are most successful and most *humanistic* when they are not simply top-down organizations, each part directed by a master plan to an alien end, but instead have a more organic unity and direction, each role responding to others and all in communication. And so the productions of these

networks – be they data models or what we make of them – each with its own integrity and *performing in its own way*, also reflect and communicate, reaching out to both past and future. In this respect, the work of the digital humanities is much as the work of the humanities always has been, *only more so*: personal and private, yet also collaborative and collegial. Despite any noble aspiration to completeness, my work is not properly done unless I can also show how it is done, staining our hands again.

12 Colophon

These slides were authored in an XML format of the author's invention and formatted in SVG and PDF using XSLT and XSL-FO.

All images in the slides are the author's, with the following exceptions:

- Lego specifications (SVG) from Wikimedia Commons: http://commons.wikimedia.org/wiki/File:Lego_dimensions.svg
- International Space Station: http://apod.nasa.gov/apod/image/0607/iss_sts121.jpg
- Aldus Manutius edition of Horace from Wikimedia Commons: http://commons.wikimedia.org/wiki/File:Aldus-Horace3.jpeg
- Woodcut of printers at work: http://en.wikipedia.org/wiki/File:Printer_in_1568-ce.png
- SVG ink splash: http://openclipart.org/people/voyeg3r/ink-splash.svg

Within the text, particular phrases and terms of interest are marked up: **persons**, **terms** and *themes*.

Two formats have been produced for publication: XHTML and PDF. The HTML version presents the slides themselves in PNG, for portability and layout fidelity across browsers and display devices. The PDF presents the slides in high resolution (which accounts for its size): so please, if you are using a PDF reader or ebook application, zoom in to resolve detail.