Extreme Markup Languages 2007®

Montréal, Québec
August 7-10, 2007

Exploring intertextual semantics
A reflection on attributes and optionality

Yves Marcoux
GRDS - EBSI

Élias Rizkallah
GRDS - EBSI

Abstract

At Extreme 2006 [Marcoux 2006], we introduced an approach to structured-document modeling based on natural language, with a strong preoccupation for the facilitation of modeler-author communication. The goal was to develop a modeling framework which would (1) facilitate the communication of the "semantic intentions of the modeler" to (human) authors of valid XML documents; and (2) encourage modelers to develop models amenable to plain understanding and interpretation by authors and other readers of the documents. A brief sketch of the framework, called *intertextual semantics*, was given, and its potential benefits and drawbacks were discussed theoretically. Application of the framework was limited to a few simple examples. In particular, attributes were not addressed directly, nor the possibility of optional "containers" (elements or attributes). In the present article, we consider how these two features of XML can be handled in intertextual semantics. We also state explicitly the two hypotheses which underpin the intertextual semantics vision of modeling.

Exploring intertextual semantics

A reflection on attributes and optionality

Table of Contents

Extreme Markup Languages®

Exploring intertextual semantics
A reflection on attributes and optionality

Yves Marcoux and Élias Rizkallah

§ Introduction

NL-based modeling and intertextual semantics

In [Marcoux 2006], we tackled the question of modeler-author communication in the context of authoring valid XML documents. We advocated that modelers (document *engineers*, *architects*, etc.) should document their models (DTDs, schemas, etc.) with artefacts allowing a "reference interpretation" to be generated automatically from any conforming document. We proposed that the reference interpretation be in natural language (NL); essentially, a character string making up a passage in NL (a sentence, a paragraph, or a whole text). The underlying idea was that, with such an apparatus, the semantic intentions of the modelers would be easy to communicate to human authors, right in the editing window, at creation time of valid XML documents. This idea might be expressed bluntly as: "showing authors the immediate textual context in which their content is going to be interpreted, is the best way to get pretty much exactly what you want from them."

Associating a "reference interpretation" to any valid XML document can be seen as defining a *semantic framework*, which assigns a "meaning"—the reference interpretation—to raw marked-up documents. Note that such a semantic framework has a quite unusual trait: it uses natural language as its *semantic* domain, not as its *syntactic* domain. In many semantic frameworks—specially those used for NLP—, natural language is used as the syntactic domain, and the semantic domain is some artificial formalism, such as first-order logic. In our framework, the *range* of the semantic function is uninterpreted NL expressions, ready, though, for interpretation by humans belonging to some target community (whose definition likely involves the mastery of some natural language).

We suggested the phrase *intertextual semantics* (IS)—after "intertextuality", coined in 1966 by Bulgarian philosopher Julia Kristeva—for our semantic framework. The phrase "intertextual semantics" was chosen to reflect the idea that meaning is given to a document fragment (or piece of data) by placing it in a network of other texts or text segments ("interrelated texts", hence the name).

The range of our semantic function is in fact not only character strings, but rather strings of *colored* characters, with the underlying alphabet comprising characters in two different colors, each character existing in the two versions (a blue `"a"` and a red `"a"`, for example). This allows us to distinguish segments supplied by the author from the ones supplied by the modeler in the reference interpretation of documents. We were not very specific about which characters the alphabet contained, but we did use a few non-graphic characters, such as spaces and paragraph breaks. [1]

Another twist (and this is what allows us to properly speak of *networks* of texts) is that segments supplied by the modeler (i.e., made up of characters in one specific color) can contain *hyperlinks*. Hyperlinks are simply addresses (or URIs) encoded in some recognizable way; in [Marcoux 2006], we enclosed them [in square brackets].

So much for the *range* of the semantic function. For the function itself, we advocated starting with a minimalistic approach. In [Marcoux 2006], we explored the use of *affixes*, or *peritexts*, [2] more precisely, associating a "text-before" and a "text-after" segment to each element type in the model. It is the modeler who specifies those peritexts, which constitute the artefacts, referred to earlier, from which the "reference interpretation" of any conforming document can be generated automatically. The generation process is most easily understood as a very simple styling mechanism, which replaces start- and end-tags by text-before and text-after segments. However, it must be emphasized that the process is *not* proposed, nor intended to be used, as a styling mechanism.

Since our initial goal was to facilitate the communication of the semantic intentions of the modeler to the authors, we envisioned an authoring environment in which the different *views* of the document that an author can select from would include an "intertextual semantics" view, in which the *intended meaning*—or reference interpretation—of the document (as established by the modeler) would be given explicitly in NL. Alternatively, it could be possible to "expand", on demand, only a fragment of a document into its

Figure 1

```
<billing>
  <amount-burial>1205.47</amount-burial>
  <payable-burial>D</payable-burial>
  <amount-cremation>788.00</amount-cremation>
  <payable-cremation>F</payable-cremation>
</billing>
```

XML fragment

intertextual semantics equivalent. We also envisioned a mechanism by which the "dialog" between an author and the editor would actually be a *semantic conversation* between the author and the modeler. This would be accomplished by having the author choose, at any decision point in the document, not from the possible *containers* insertable at that point, but from the possible *meanings* (represented by the peritexts) of those containers.

Finally, we suggested that thinking in terms of IS might lead to a new approach to modeling, in which the modeler—roughly—would start with NL, then work out the peritexts, then the markup, the latter being thought of as "abbreviations" of the peritexts.

An example

This example uses a fragment of an actual prototype XML model proposed for the transfer of information between funeral homes and cemeteries in a project in which the authors are involved. The fragment corresponds to the billing section of a request sent by a funeral home to a cemetery. The example has been translated from French for inclusion in the present article.

The element declarations are straightforward (all elements are mandatory and non-repeatable), and thus, omitted. We present:

1. the raw XML source of a fragment, valid—by hypothesis—with respect to some DTD or schema (Figure 1);
2. the "text-before" and "text-after" segments (peritexts) for each element type, as would be established by the modeler (Table "Table: Peritexts");
3. the IS rendering of the XML fragment, as it would appear in an authoring environment, should the author choose the IS view (Figure 2).

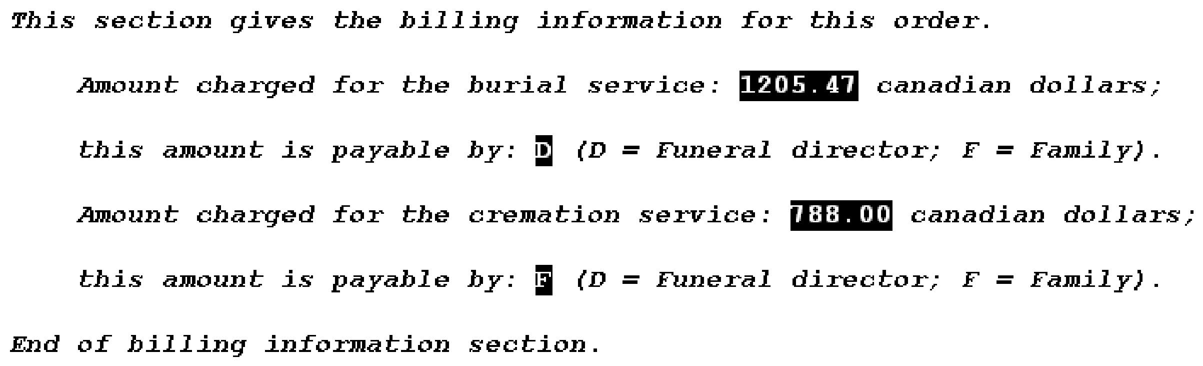
Table 1: Table: Peritexts

Element	text-before	text-after
`billing`	"This section gives the billing information for this order. "	" End of billing information section."
`amount-burial`	"Amount charged for the burial service: "	" canadian dollars; "
`payable-burial`	"this amount is payable by: "	" (D = Funeral director; F = Family)."
`amount-cremation`	"Amount charged for the cremation service: "	" canadian dollars; "
`payable-cremation`	"this amount is payable by: "	" (D = Funeral director; F = Family)."

Note that the IS view shown was generated automatically by a generic mechanism (based on XSLT) which, for clarity, also does some indentation of the elements according to the XML structure, in addition to inserting the peritexts. The author-contributed characters are typeset in roman on black background, and the modeler-contributed ones (forming the peritexts) are in italics. Please bear in mind, however, that IS is *not* a styling mechanism.

Other examples can be found in [Marcoux 2006].

Figure 2



```
This section gives the billing information for this order.
    Amount charged for the burial service: 1205.47 canadian dollars;
    this amount is payable by: D (D = Funeral director; F = Family).
    Amount charged for the cremation service: 788.00 canadian dollars;
    this amount is payable by: F (D = Funeral director; F = Family).
End of billing information section.
```

IS rendering of the XML fragment

Development strategy for the framework

In the framework sketched last year, many structural possibilities present in XML were not discussed. For example, attributes were not addressed directly, nor the possibility of optional “containers” (elements and attributes).

Among the possibilities of future work mentioned, were:

1. Enrich the underlying mechanism to include more than peritext insertion, including support for attributes.
2. Integrate the consultation/insertion of valid contents examples, as a way to facilitate document creation.
3. Implement the first sketch of the framework with some styling mechanism (CSS, XSLT) in a real XML editor (in “dialog” mode, not just “application preview” mode, which is possible right now with no development in editors like XMetal, but still would not demonstrate the full possibilities of intertextual semantics).
4. Work out intertextual semantics for some existing models.
5. Experiment in actual authoring situations.

While implementation would be very compelling, we deemed it necessary to start by trying to work out the semantics of a few existing models. Two kinds of benefits were anticipated *a priori* from such an endeavor. First, it could indicate how the definition of intertextual semantics should be enhanced, compared to the minimalist approach of the first sketch. Second, it could reinforce, or else decrease, one's intuition that intertextual semantics is indeed a good counselor in modeling. Of course, only user studies could actually “prove”, in any scientific meaning of the term, how models and documents created in the intertextual semantics framework stand in relationship to others in terms of understandability and usability, but real-life confrontation as described above could at least give intuitive insight into the value of the approach.

We intended to work on *Really Simple Syndication* (RSS) [RSS 2007], because of the popularity of the format, and also because of the general feeling of uneasiness associated with its deployment (at least in structured document circles; see for example [Bray 2006]). We wanted to see whether an intertextual analysis could shed some light on the reasons of those alleged problems. We also intended to work on Atom [Atom 2005], a competing format to RSS for Web syndication, which we wanted to use as a point of comparison, and which, perhaps, intertextual analysis could reveal superior to RSS in some respect. [3]

Very quickly, we met with basically two interrelated types of problems. The first one is the handling of attributes, which the framework sketch did not address at all. The second one is omissible “containers” (element or attribute). The problems are interrelated because attributes can be omissible (i.e., in DTD's, `#IMPLIED`).

Reflecting on attributes lead us to develop an idiosyncratic mechanism to handle them in IS. We present here this mechanism, which also accounts for omissible attributes. In the case of omissible *elements*, our reflections lead us not to extend IS, but rather suggest an approach for dealing with the difficulty within IS.

All our reflections are based on two different hypotheses, which we call *intertextual semantics hypothesis* (ISH) -1 and -2, and which we realized had to be stated explicitly. ISH-1, essentially, is that IS can be considered a valid representation of the writing/reading process of a document. ISH-2, in a nutshell, is that a raw marked-up document should be an “abbreviation” of its intertextual semantics. It is related to the question of information perennity and document autonomy. ISH-1 and -2 were not fully explicited in last year's paper.

About this paper

In this paper, we:

1. Present ISH-1, which states that the intertextual semantics of a document can be considered to be a valid representation of the writing/reading process of that document. (Section “Making sense of a document, intended interpretation: ISH-1”.)
2. Present ISH-2, which states that it makes sense to orient markup choices so that a raw marked-up document turns out to be more or less an “abbreviation” of its intertextual semantics. (Section “Document autonomy, information perennity: ISH-2”.)
3. Present two difficulties encountered in the process of trying to specify an intertextual semantics for existing models (RSS and Atom): handling attributes and handling omissible containers. For the first one, we give an extension of the IS framework mechanism to treat attributes. For the second, we explain why extending IS would not be the most appropriate solution, and suggest an approach for dealing with the difficulty within IS. (Sections “Attributes in the light of IS” and “Optionality”.)

We conclude with general comments and indications of future work.

Related work

Standard modeling methodologies (for example, [Travis & Waldt 1995] [Maler & El Andaloussi 1996] [Glushko & McGrath 2005]) do not in general include a full-fledged formal semantic framework. The semantic aspects of modeling are rather treated in pragmatical terms through discussion of *human-readable documentation* and *application development*. We call those two approaches the *pragmatical* semantic frameworks of structured documents. Semantic properties and usability of models and of conforming documents, though recognized as crucial, are side-effects of those pragmatical considerations.

Some formal semantic approaches have been proposed in the past to provide semantics to structured documents. Renear, Dubin, and Sperberg-McQueen [Renear et al. 2002] provide a historical background and a description of a specific project: BECHAMEL. Judging from their paper, the general premise is that natural-language descriptions are insufficient and must be complemented by a separate formal apparatus:

> What is needed is a mechanism that would allow the markup language designer to rigorously and formally specify semantic relationships; these specifications could then be read by processing applications which would configure themselves accordingly, without case by case human intervention.
>
> [Renear et al. 2002], p. 122.

While we share that point of view, the “processing applications” we have in mind are very specific: document authoring. That is why we do not follow the same track as [Renear et al. 2002]. In a way, we could describe our approach as trying to operationalize natural-language descriptions is such a way that they can support document creation. We do not replace natural-language, we frame it with mechanisms that make it supportive of the document creation process.

The semantic approach that seems closest to ours has been introduced by Wrightson [Wrightson 2001] [Wrightson 2005]. Wrightson used *situation semantics* [Devlin 1991] to analyse, among other things, human legibility of XML *well-formed* documents. Although we are interested here only in valid documents, there seems to be a lot in common between Wrightson's preoccupations and ours. However, Wrightson's approach seems to be more *explanatory*, whereas ours aims to be more *prescriptive*, in the sense of being deployable right at document-authoring time, and of suggesting modeling practices more directly. We concentrate on the *writing* of *valid* XML documents, hence on the communication between modeler and author at document-creation time, direct human legibility of documents being only a possible side-effect.

Another approach with which we have a lot in common is that of Sperberg-McQueen, Huitfeldt, and Renear, presented at Extreme 2000 [Sperberg-McQueen et al. 2000]. The authors develop a framework for structured-document semantics based on *sentence skeletons* and *deictic expressions* (a generic concept of which XPath relative expressions are a specific case). Although similar concepts can be found in our framework (or in possible extensions), there are two important differences between our approaches: First, although the possibility of using natural-language sentence skeletons is mentioned in [Sperberg-McQueen et al. 2000], the bulk of the discussion—and all examples—use Prolog predicates. Second, in [Sperberg-McQueen et al. 2000], the primary focus is not on document authoring, but rather on inferences ("licensed"—or legitimized—by the markup) that can be made by the readers of a document. Note that the set of inferences licensed by a piece of markup can be considered to be a description of its semantics (a point of view explicitly adopted by Sperberg-McQueen et al., on p. 233 of their paper). As such, one could hope to use this set for providing semantic support to authors at authoring time. However, a set of inferences might not be the most appropriate semantic description in this context: for one thing, the set might be infinite; even if it is not, it might be hard to compute and/or understand. In a way, the IS of an XML fragment can be viewed as conveying the *facts*—in a very general sense—licensed by that fragment, facts from which readers will later be able to make their own inferences.

The idea of using text-related techniques to improve systems design is not new: Smith [Smith 1994], for example, wrote that "[t]alk, theorized as conversation and analyzed as discourse, may provide the models of interaction that we need, in order to improve the design of hypertext systems and to extend the reach of its applications" (p. 281). Well-known examples of text-related techniques for systems development are Donald Knuth's WEB system and *Literate Programming* in general [Knuth 1984], TEI's ODD (*One Document Does it all*; see for instance [Cover 2005]), and Sperberg-McQueen's SWEB [Sperberg-McQueen 1996]. Using semiotics in general—not just textual—in interface design has also been explored, for example by De Souza [De Souza 2005]. *Personas* [Pruitt & Adlin 2006] and *storytelling* [Erickson 1996] also fall in that category of techniques.

§ Making sense of a document, intended interpretation: ISH-1

Suppose that looking at a document fragment through some viewing or styling interface yields the following display:

```
Price: 10000 MNT
```

The underlying raw document fragment, supposing it is in XML, might look like this:

```
<price currency="MNT">10000</price>
```

Suppose we want the user (reader) to have a feeling of how much money that amount represents, maybe because we are trying to sell him something, and we want him to be able to compare prices. Unless the user has some very specific knowledge about currencies, he does not get such a feeling just by looking at the display. There *is* a message that he immediately derives from the display, because of his familiarity with the particular communication genre which deals with presenting amounts of money: it is that he needs to access resources external to this display in order to make sense of it. First, he needs to "resolve" the currency code, then consult a current conversion rate table for that currency. But the display itself gives no clue as to how to resolve those links, in fact the resources needed are not even mentioned, let alone how to access them. One could suspect that a standard code is used for currencies, but whether there are many such codes, and where they are must be discovered. Ditto for currency conversion tables.

Suppose that the currency codes used by the system conform to standard *ISO 4217 Currency names and code elements*. If we want to make things easier for the user, we could add, in the interface, indications about the required resource:

```
Price: 10000 MNT (currency code as per ISO 4217)
```

That would at least tell the user that he needs to consult a specific list to resolve the currency code. Now, we could make things even easier for him. We could add a live link to the ISO 4217 table of currency codes:

```
Price: 10000 MNT (currency code as per ISO 4217,
which you can consult by clicking "here")
```

The word "here" would be a link to <http://www.iso.org/iso/en/prods-services/popstds/currencycodeslist.html>, the list of ISO 4217 codes. That would save the user the burden of locating the mentioned standard.

Of course, there is much more that can be done. We could for example program in the interface a call to a currency-conversion Web service, and show only the amount in a currency chosen by the user.

Now, suppose the system can save documents so they can be recalled at a later time (and maybe displayed using a different currency, for example), or simply for record-keeping purposes. The most sensible approach is of course for the system to save the raw documents, not the display.

Suppose after some time you want to consult some of the saved documents, and either the original viewing application is no more available, or you are temporarily disconnected from it. How, then, will you make sense of the raw documents?

Now, if the modeler did his job correctly, there *should* be comments in the document model (DTD, for example) explaining all the “interpretation paths” that allow making usable sense of the raw documents. But how are they formulated, and where exactly are they? Did the information disappear with the application? We are in a kind of “blind” lookup problem, as we had encountered with the original display, knowing we must look for something, but without any explicit name for it or pointer to it. All we have for sure is a pointer (`DOCTYPE` or `schemaLocation`) to a model, which is only guaranteed to have declarations in it (content models, etc.), not necessarily comments in some definite shape and form.

Now, the reason why you might need to know the meaning of the various names in a model, is not necessarily that you want to look at raw documents; it might be that you are developing a new application for exploiting the documents in a way different from the viewing application (compute statistics on a collection of documents, for instance). So the need to give meaning to a model is not just to be able to interpret raw documents. But this task certainly gives a fair measure of how well we “understand” the model.

In essence, the idea behind intertextual semantics (IS) is to make sure the interpretation of raw documents is always possible, and as easy as possible, by urging the modeler to:

1. Identify explicitly all the explanations and resources needed to perform the interpretation *over and above what is assumed to be known by the members of the target community*, and provide live pointers to those resources, whenever possible.
2. Anchor those explanations and pointers to resources in peritexts associated with each element type of the model.
3. Limit as much as possible the number of link traversals—from peritext to resources and from one resource to another—needed to perform the interpretation.

The modeler specifies, for each element type of the model, text-before and text-after segments (peritexts) that will be inserted before and after each element of that type in a document instance, to form what is called the *intertextual semantics* of that element. In the worst case, those text segments will contain only a pointer to a resource explaining how to interpret the content of the element. But that link traversal could sometimes be eliminated by “wording” the segments in such a way that they make up a sensible NL passage, giving sense directly to the element content, by placing it a an appropriate (textual) context.

The idea is that the peritexts (the IS specification for the model) are an integral part of the model; we assume that they will always be available to assist in the interpretation of documents, that you will never be stuck with only the XML declarations of the model. So the “recipes” to make sense out of raw documents will not go away with an application.

Of course, the “network” of resources needed to interpret raw documents of a given model need only extend until it reaches resources directly understandable (or assumed to be so) by the members of the target readers community. Once such directly understandable resources are reached, there is no need to provide links to further explanations of those resources.

ISH-1 *Intertextual semantics hypothesis 1* (ISH-1) states that for any given document model and any given target community, the intuitive notion of an “intended interpretation” (of the raw documents of the model by the members of the community) is adequately represented by an IS specification for the model, together with the network of resources referenced in this specification.

So, according to ISH-1, there is no more accurate way to specify an intended interpretation for the documents of a given model, than to give an IS specification for that model. This might seem very strong, but think of it the other way: can we legitimately say that an intended interpretation is specified if we cannot give a collection of resources and a path through them that suffice to reach that interpretation? Recall that IS, being based on NL, can be as precise or vague as deemed appropriate by the modeler. So, on the one hand, the vagueness or fuzziness of an envisioned interpretation does not prevent it from being expressible in IS. On the other hand, interpretations in terms of extremely precise, mathematical-like formalisms can also be expressed, as long as there exists a resource (textbook or other) explaining the formalism *in NL*, resource which we can refer or link to from a peritext.

It is important to realize that the IS of a document *suggests* an interpretation path for a document, but that path is by no means unique, nor mandatory. Imagine, for instance, that a user whose "home currency" is the Mongolian Tugrik (ISO 4217 code MNT) is confronted to the raw document shown above, then there is no reason for him to consult any external resources in order to get a feeling of how much money the amount represents. The path suggested by IS is somewhat like footsteps painted on the floor in public places, showing how to get from place X to place Y; nobody is expected to put their feet exactly on the painted spots, yet, they are useful to indicate a general direction, and an exact destination. And some people know shortcuts, and will use them.

It is also important to realize that a particular reading of a document may yield much more information than is present in the "intended interpretation" given by IS. IS corresponds to a kind of "minimal" meaning that everybody in the target community is expected to get.

§ Document autonomy, information perennity: ISH-2

ISH-2 *Intertextual semantics hypothesis 2* (ISH-2) states that it is desirable that a raw XML document be a sort of "abbreviation" of its IS, as given by the IS specification of its model. In particular, this means that the model names should be as close to "abbreviations" of their peritexts as possible.

We will not discuss at length ISH-2, but simply point out that it agrees with the idea of information perennity. If we strive for a raw XML document to be as close as possible to its IS, then we increase the chances that it still be interpretable, should it by any chance survive its model (and thus, its IS specification). It also agrees with the simple no-nonsense argument that, unless deliberate obfuscation is seeked, there is no reason to choose model names that do not indicate as much as possible the meaning of the corresponding containers (elements, attributes).

Now, there are limits to what you can put in an XML name; for example, there is no way to embed a hyperlink in it, at least without resorting to horrible escaping conventions. But still, a lot can be done. For example, applying ISH-2 to the example above would yield a raw document that might look like this:

```
<price currency-ISO4217="MNT">10000</price>
```

If, for any reason, someone has to interpret that raw document without the corresponding IS, chances are they will understand currency codes are as per ISO 4217.

§ Attributes in the light of IS

Attributes enjoy a special status in structured documents. Various people have diverging points of view on them. Mostly and generally, though, they are recognized as breaking the purely hierarchical nature of a document. They are often presented using the "Post-it" metaphor, as a structure adjunct to a document element, but aside from it. Attributes are viewed as notes written on a Post-it stuck to an element node, or on a label tied to the node with a string.

Another evidence of the "aside" nature of attributes is the fact that some editors (including for example XMetal), never show attributes in the main editing window. Instead, they show them in a separate pane, and for just one element at a time. Additionally, attributes appear in tool-tip pop-ups (with yellow background!) displayed whenever the mouse cursor hovers above a start- or end-tag, which increases the "Post-it" impression.

Another peculiar aspect of attributes is the lack of order among them. Although many XML tools have options to order attribute specifications within a start-tag (for example, the oXygen editor can order them alphabetically by attribute name), it is impossible, by modeling, to *impose* any particular ordering among the attributes of an element (even in RELAX NG, where the absence of order is rather artificial [Clark & Murata 2001]).

In IS, the question of how to handle attributes is of great importance, because we want the semantics of any element to be more or less perfectly sequential.

It has sometimes been said that attributes should be used only to convey layout information, or at least information that can be ignored in a sequential human-reading of the document (see for instance [Castro 2000]). Obviously, however, they can contain information crucial for the interpretation of the element (and of the whole document, for that matter); just consider how a `status="withdrawn"` attribute specification might impact the reading of an element. Thus, it is not possible to simply ignore them in a semantic framework.

In previous work, attributes have sometimes been treated as subelements. For instance, in [Marcoux & Sévigny 1997] and [Le Maitre et al. 2000], attributes were conceptually transformed into text-only (i.e., `#PCDATA`) subelements with a generic ID equal to the attribute name prefixed by `"@"`, and with a textual content equal to the normalized value of the attribute. Thus, for example, an HTML element `<a href="dog.jpg">Click here</a>` would conceptually be transformed into `<a><@href>dog.jpg</@href>Click here</a>`.

The question of attributes in IS has two sides: (1) how can we assign IS to an existing model that uses attributes; and (2) what do we suggest for the creation of new models to take up the role of attributes.

As far as question (1) goes, treating attributes as subelements—in a way similar to the one presented above —would result in conflicting `@`-names for attributes with the same name for different elements. This problem, however, could be solved by making each `@`-element local to its parent element (in the sense of W3C schemas local elements). Other problems—with no obvious solution, this time—would be: (1) The total separation of the peritexts of an attribute from those of the element; there would be no possible "interplay" between the two, corresponding to the idea that an attribute simply tweaks or modulates the element's meaning. (2) The absence of order; the peritexts of all `@`-elements would have to be formulated in a way general enough to make sense whatever their order of appearance, which is a significant constraint.

As far as question (2) goes, treating attributes as subelements would be tantamount to eliminating them altogether from the modeler's toolkit. Thus, we would say "there is no such thing as unordered containers, existing on the side of the purely hierarchical structure of a document; just be happy with subelements". This sounds like a big loss.

So it does seem that attributes require idiosyncratic treatment.

Idiosyncratic treatment of attributes

We first present an extension of IS that supports attributes, but not omissible attributes. In the next section on optionality, we will modify it to allow omissible attributes.

Existing devices for "sideway" comments in natural language provide hints as to how attributes might be handled: for example, comment and parenthetical clauses—clauses presented (as this one) in parentheses or between long dashes—are used for this purpose.

The treatment we suggest for attributes, however, is not directly based on those devices. We permit *attribute place-holders* in the peritexts associated with elements. An attribute place-holder has the form `@`*`attribute-name`*. Whenever such a place-holder is found in text-before and text-after segment, it is replaced by the (normalized) value of the attribute named *`attribute-name`*. Of course, it is *possible* to use the NL devices for sideway comments *around* the place-holders in the peritexts of an element, but it is not mandatory.

For example, an `amount` element might have a `currency` attribute, and the text-before segment for `amount` might be defined as `"Amount (in the currency whose alphabetic code in ISO 4217 is “@currency”): "`.

Thus, the fragment `<amount currency="CAD">1024</amount>` would have as an intertextual semantics the string:

```
Amount (in the currency whose alphabetic code in ISO 4217 is
“CAD”): 1024
```

Note that here, one of the mentioned NL devices for sideway comments is used (parentheses).

A place-holder is allowed to occur *exactly once* in the text-before *or* text-after (not both) of an element, for each possible attribute of that element. This restriction is introduced to avoid the possible confusion that might result, in authoring environments, if the value of an attribute could be edited in two or more different places.

Other approaches would have been possible, including assigning separate text-before and text-after segments to each attribute. But then, the lack of ordering would have to be addressed separately. We feel our proposal respects the “asideness” aspect of attributes and gives a satisfactory solution to the ordering problem.

Does our proposal provide a “natural” criterion for judging the quality of attribute names? Recall that, for element names (genID's), the natural “quality” criterion is that they be more or less abbreviations of their peritexts. The attribute names do not appear directly in the peritexts *after* replacement of the place-holders by their respective attribute values; however, they do appear in the “raw” peritext, as supplied by the modeler. This suggests a “natural” quality criterion for attribute names: each should be an “abbreviation” of the neighborhood in which the corresponding place-holder appears in the element's peritexts.

It should be noted that our introduction of attributes slightly modifies the role of peritexts for elements that do have attributes: they now play a somewhat joint role of peritext for the element itself and for its attributes.

Finally, we mention one possible benefit of using the suggested approach in authoring environments: the attributes, being integrated to the peritexts of elements, could now naturally be displayed in the main editing window, without disrupting the sequentiality of the document. Having them right under his eyes, the author would be constantly reminded of their existence and meaning, which might help avoid errors.

§ Optionality

We will start our discussion of optionality with the case of omissible attributes.

The problem with the mechanism for attributes just presented is that it more or less breaks down in the case of omitted attributes. If an attribute is omissible, and omitted in an instance, how is the peritext handled? One option would be to do as if the attribute value were the empty string. Then, with the above example, the IS of the element would become:

```
Amount (in the currency whose alphabetic code in ISO 4217 is “”): 1024
```

One could argue here that this particular attribute should not be omissible, but rather have a default value. The point is, even with a legitimately omissible attribute, the peritext would be badly ill-formed.

There are many possible solutions. One is to have *sections*, instead of place-holders, for attributes in the element's peritexts. The insertion of a section corresponding to an attribute is conditional to the presence of the attribute in the start-tag of the element.

The sections would be marked using the notation:

```
@attribute-name[...section...]
```

The `section` itself is allowed to contain a single occurrence of the character `"@"`, which will indicate where the attribute value is to be inserted in the `section`. (Of course, appropriate escaping conventions would be needed to allow for characters `"@"` and `"]"` to occur as data within the sections.)

The peritext of the previous example would then be rewritten:

```
Amount@currency[ (in the currency whose alphabetic code in ISO 4217 is
“@”)]: 1024
```

This, with an attribute value of `CAD`, the peritext becomes:

```
Amount (in the currency whose alphabetic code in ISO 4217 is
“CAD”): 1024
```

With the attribute omitted, it becomes:

```
Amount: 1024
```

The important thing, at this point, is not the exact mechanism used, but the fact that there are fairly simple ways to handle omissible attributes adequately.

Meaningful omissions

IS was introduced with facilitation of document authoring in mind, trying to make modeler and author communicate better. However, the following discussion is more relevant if we adopt the point of view of a reader. So we will look at how readers make sense of a document, how they can make *all the sense* there is in a document.

Suppose the high-level structure of a memo is defined by:

```
<!ELEMENT memo (author, date, public?, body)>
```

where `public` is an `EMPTY` element, which is thus either present or absent in any specific memo. In the IS framework, the semantics of element `public` would be made precise by defining appropriate text-before and/or -after segments, for example a text-before of “Viewing of this memo is not restricted to members of its author's organization.”

For an author, this IS specification probably suffices as semantic support. Indeed, come to the point between `date` and `body`, a semantic conversation fragment will take place between the editor and the author, whereby the author will be informed of the possibility of adding “Viewing of this memo is not restricted to members of its author's organization” to the meaning of his document.

Even so, one might argue that, depending on the particular authoring environment, the cursor may never actually end up on its own at this particular point in the document, and the author may never quite realize that this addition is possible. Thus, even from an author's point of view, this IS specification *may* be somewhat incomplete. From a *reader*'s point of view, however, the problem is more obvious.

Recall that we would like the IS of a document to serve as its “reference interpretation”, the one that would be put forward in court should the document ever be involved in a dispute. One would expect such a reference interpretation to convey *all* the sense that was purposely meant to be in the document.

For a reader to get all the sense there is in a memo, it is important for him to know that there *could* have been a `public` subelement, even when he is looking at a memo that doesn't have that subelement; in other words, that the absence is meaningful. To find this out, the reader could traverse either a `DOCTYPE` or a `schemaLocation` reference, or the equivalent, read and understand the content model (or the equivalent in documentation), *then* conclude that there is a deliberate absence in the memo. This is a longer interpretation path than if the information were right in the IS of the memo.

A second problem would be for the reader to know *exactly what it means* for a memo not to have a `public` subelement. With just the content models, he could not be sure exactly what the modelers thought it should mean. The way the IS of `public` is worded, though, he should be entitled to think that its absence means the *opposite* of what the presence means, namely that “Viewing of this memo *is* restricted to members of its author's organization.” But had `public` meant “This memo must be sent to the media,” what exactly, then, would the opposite be? A strict logical negation (“need not be sent”) or a more colloquial negation (“may not be sent”)? Hard to tell for sure without more information.

A solution to both problems is that there be an indication, *in the IS of the memo*, to the effect that, “unless otherwise stated, viewing of the memo *is restricted* to members of its author's organization.” Clearly, that indication cannot be in a peritext associated with `public` (because it is relevant exactly in those cases where `public` is omitted), however, it could be in the peritext of the containing element `memo`.

Another approach would have been to enrich the IS framework to allow the injection of text in the IS of a document at the point where some omissible component is omitted. For those elements that can be omitted, we would add a third peritext saying what should be inserted at the point of omission in the IS (a similar device could also be devised for omissible attributes). For example, we could have:

Table 2

Element	**text-before**	**text-after**	**if-omitted**
`public`	"Viewing of this memo is not restricted to members of its author's organization. "	empty	"Viewing of this memo is restricted to members of its author's organization. "

However, we are not inclined to go that way, essentially because of ISH-2. Recall that ISH-2 says that a raw XML document should as much as possible be an abbreviation of its IS. If a segment in the IS of a document were generated by the absence of something (the empty string) in the raw document, then, that something (the empty string) could hardly be considered an abbreviation of the segment. Another problem might be to determine exactly when an element is to be considered omitted.

An approach for optionality and choices

A datatype assigned to any “field” (programming language variable, database column, etc.) restricts the values that can be stored in it. As such, it can also contribute to the interpretation of the values stored in the field. For example, in a *genre* field of enumerated type with the three possible values "jazz", "pop", and "classical", a value of "pop" does not have quite the same meaning as if there were 200 possible genres. Likewise, protocols governing the entry of data in a field affect its interpretation: a three-line article summary is not interpreted the same way if summaries are limited to five lines, and if they are limited to 100 lines. So, knowing what the *possible* contents are can be important for the interpretation of their *actual* contents.

Essentially, the implication for IS is that sometimes, something about the content model, possible attributes, and writing rules of an element might have to be mentioned in the IS of that element (either directly in a peritext, or indirectly, in a resource pointed to by a hyperlink in a peritext). Mandatory subelements and attributes can “take care of themselves,” that is, contribute to the IS by their own peritexts (or through their contribution to the element's peritexts for an attribute). However, if we want to mention the possible presence of omissible subelements and attributes, *this must be done in the IS of the parent element* (or element, for an attribute).

This turns out to be the solution to omissible elements that we presented above. But it also concerns *subelements that might be absent as a result of a choice*. For example, suppose memos are defined by:

```
<!ELEMENT memo (author, (internal | external), body)>
```

then, it might be relevant to mention, in the IS of `memo`, that `internal` and `external` are the two (and only two) possible followers of author. Because knowing that there are two possibilites, and what they are, helps interpret better the actual value. It is a question that the modeler must consider.

§ Conclusion

In this article, we addressed the question of how attributes and omissible containers (elements and attributes) can be handled in the framework of intertextual semantics. Our conclusion for attributes was that they require an extension of the framework, and we presented one. For omissible containers, we concluded it is preferable not to extend the framework for them, and suggested a possible approach to deal with them within the framework.

These two questions showed up when we started trying to work out IS for two existing models: RSS and Atom. One of the goals of the endeavor was exactly to raise such questions. Future work includes pursuing the confrontation to “real life” models, to further enrich and refine the framework.

There is little doubt in our mind that IS can *theoretically* be applied to define the semantics of any structured document model. Being based on natural language, it shares its versatility, extremely high affordance, and self-definability. Actually, we would not be too surprised if IS turned out to be an interesting theoretical setting for studying the whole span of *information architecture* activities [Dillon & Turnbull 2005], from need analysis to user interface design, or at least some of those activities besides modeling.

Whether IS-based modeling can be done in a practical and useful fashion, however, is still an open question, which we look forward to investigate through experimentation in authoring situations, once the framework has reached a fairly stable state.

Notes

1. Non-graphic characters cannot be colored, but they could have, say, a background color. The important thing is that *some* notational convention allow distinguishing modeler-contributed from author-contributed characters.

2. The term “peritext” was coined almost at the same time (in 1987) with two different meanings, by Gérard Genette (“Seuils” – Paris : Seuil, 1987) and Marcel De Grève (“Texte et péritexte”, in : Degrés, Vol. 15, no. 49-50, Spring-summer, 1987, pp. 1-21). Our use of the term is closer to Genette's, for whom it designates segments around a text, in the space of a single document, like a title, a foreword, or even footnotes “inserted in the chinks of the text”. For us, however,

these segments must also have sequential compositionality with the text around which they appear.

3. Most RSS and Atom documents are generated automatically by computer applications, without a direct human author. But the semantics of the model must still be communicated to someone: the application developer, who, in fact, indirectly assumes the role of author. We believe a framework focused on the modeler-author communication, such as IS, is still relevant in this context. Instead of looking at the IS of a specific document being created (like an author would do), the developer would look at the global IS specification of the model to infer the "meaning" of each element type.

Bibliography

[Atom 2005] *The Atom Syndication Format.* IETF RFC 4287, December 2005. **http://www.ietf.org/rfc/rfc4287**

[Bray 2006] *Tim Bray on Rails, REST, XML, Java, and More.* InfoQ, Oct 11, 2006. **http://www.infoq.com/interviews/tim_bray_rails_and_more**

[Castro 2000] Castro, Elizabeth. *XML for the World Wide Web: Visual QuickStart Guide*, 2001, Peachpit Press, 272 pages.

[Clark & Murata 2001] CLARK, James; MURATA Makoto. *RELAX NG Specification.* 2001. **http://www.oasis-open.org/committees/relax-ng/spec-20011203.html**

[Cover 2005] Cover, Robin. *SGML/XML and Literate Programming.* **http://xml.coverpages.org/xmlLitProg.html**

[De Souza 2005] De Souza, C. S. *The semiotic engineering of human-computer interaction.* MIT Press, 2005.

[Devlin 1991] Devlin, Keith. *Logic and Information.* Cambridge University Press, 1991.

[Dillon & Turnbull 2005] Dillon, A.; Turnbull, D. *Information Architecture*, in *Encyclopedia of Library and Information Science.* New York: Marcel-Dekker, 2005.

[Erickson 1996] Erickson, T. "Design as storytelling." *interactions* 3, 4 (Jul. 1996), 30-35. **http://doi.acm.org/10.1145/234813.234817**

[Glushko & McGrath 2005] Glushko, Robert J.; McGrath, Tim. *Document engineering: analyzing and designing documents for business informatics and Web services.* MIT Press, 2005.

[Knuth 1984] Knuth, Donald. "Literate Programming." *The Computer Journal*, 27(2), 1984.

[Le Maitre et al. 2000] LE MAITRE, Jacques; MARCOUX, Yves; MURISASCO, Elisabeth. "SgmlQL + XGQL = Powerful XML Pattern-Matching and Data-Manipulation in a Single Language." In: Mariani, J.; Harman, D. Content-Based Multimedia Information Access: Actes de la conférence RIAO 2000, vol. 2. Paris, Centre de hautes études internationales d'informatique documentaire, 2000, pp. 1346-1362.

[Maler & El Andaloussi 1996] Maler, Eve; El Andaloussi, Jeanne. *Developing SGML DTDs: From Text to Model to Markup.* Prentice Hall PTR, 1996.

[Marcoux & Sévigny 1997] MARCOUX, Yves; SÉVIGNY, Martin. "Querying hierarchical text and acyclic hypertext with generalized context-free grammars." In: Devroye, L. et Chrisment, C. Computer-Assisted Information Searching on Internet: Actes de la conférence RIAO 1997, vol. 1. Paris, Centre de hautes études internationales d'informatique documentaire, 1997, pp. 546-561.

[Marcoux 2006] Marcoux, Yves. "A natural-language approach to modeling: Why is some XML so difficult to write?" *Proceedings of Extreme Markup Languages 2006.* **http://www.idealliance.org/papers/extreme/proceedings/html/2006/Marcoux01/EML2006Marcoux01.html**

[Pruitt & Adlin 2006] Pruitt, John; Adlin, Tamara. *The Persona Lifecycle: Keeping People in Mind Throughout Product Design*. Morgan Kaufmann, 2006.

[Renear et al. 2002] Renear, Allen; Dubin, David; Sperberg-McQueen, C. M. “Towards a Semantics for XML Markup.” *Proceedings of Document Engineering 2002.* **http://portal.acm.org/citation.cfm?doid=585058.585081**

[RSS 2007] RSS Advisory Board. *RSS 2.0 Specification (version 2.0.8).* **http://www.rssboard.org/rss-specification**

[Smith 1994] Smith, C. T. “Hypertextual thinking.” *In:* Selfe, C.; Hilligoss, S. *Literacy and Computers: The Complications of Teaching and Learning with Technology.* New York: MLA, 1994, pp. 264-281.

[Sperberg-McQueen 1996] Sperberg-McQueen, C. M. *SWEB: an SGML Tag Set for Literate Programming.* 1996. **http://www.w3.org/People/cmsmcq/1993/sweb.html**

[Sperberg-McQueen et al. 2000] Sperberg-McQueen, C. M.; Huitfeldt, Claus; Renear, Allen. “Meaning and Interpretation of Markup: Not as Simple as You Think.” *Proceedings of Extreme Markup Languages 2000.*

[Travis & Waldt 1995] Travis, B.; Waldt, D. *The SGML Implementation Guide: A Blueprint for SGML Migration.* Springer, 1995.

[Wrightson 2001] Wrightson, Ann. “Some Semantics for Structured Documents, Topic Maps and Topic Map Queries.” *Proceedings of Extreme Markup Languages 2001.*

[Wrightson 2005] Wrightson, Ann. “Semantics of Well Formed XML as a Human and Machine Readable Language: Why is some XML so difficult to read?” *Proceedings of Extreme Markup Languages 2005.*

The Authors

Yves Marcoux
GRDS - EBSI
Université de Montréal
CP 6128, succ. Centre-ville
Montréal
Québec
Canada
H3C 3J7
yves.marcoux@umontreal.ca
http://mapageweb.umontreal.ca/marcoux/

Yves MARCOUX is a faculty member at EBSI [École de bibliothéconomie et des sciences de l'information], University of Montréal, since 1991. He is involved in teaching, research, standardization, and international cooperation activities in the fields of structured documents, information retrieval, database systems, and digital information management. Prior to his appointment at EBSI, Dr. Marcoux has worked for 10 years in systems maintenance and development, in Canada, the U.S., and Europe. He obtained his Ph.D. in theoretical computer science from Université de Montréal in 1991. His main research interests are document theory, structured document implementation methodologies, and information retrieval in structured documents. He is author of many research reports and scientific articles on various aspects of structured documents. Since 1995, he has led numerous projects related to XML and SGML theory and practice. He is sollicited as an expert on digital information management and structured documents on a regular basis. He has been co-responsible for the Digital Information Management Certificate at EBSI, from its creation in 2000, to 2005. Through GRDS [Groupe départemental de recherche sur les documents structurés], his research group at EBSI, he has been principal architect for the *Governmental Framework for Integrated Document Management*, a project funded by the National Archives of Québec and the Québec Treasury Board.

Élias Rizkallah
GRDS - EBSI
Université de Montréal
CP 6128, succ. Centre-ville
Montréal
Québec
Canada
H3C 3J7
elias.rizkallah@gmail.com
http://grds.ebsi.umontreal.ca/

Élias Rizkallah holds a Master's degree in Library and information science from University of Montréal, where he is currently researcher at GRDS (Groupe départemental de Recherche sur les Documents Structurés). He is also digital information manager for the International Observatory on Financial Services Cooperatives and text-mining methodologist for the Desjardins Centre for Studies in Management of Financial Services Cooperatives, both at HEC Montréal (Hautes Études Commerciales). He is currently completing a doctoral dissertation in the Department of Psychology at Laval University (Québec, Canada), on the methodology and epistemology of research in the field of social representations.

Extreme Markup Languages 2007®
Montréal, Québec, August 7-10, 2007
This paper was formatted from XML source via XSL
by Mulberry Technologies, Inc.