John McClure

To: 'Legalxml-Econtracts'
Subject: RE: [legalxml-econtracts] Structural Markup – "Basic" Clause Model Recommendations Report

Jason,
Thanks for taking the arrows this time! Long note ahead!
Regards,
John

Here's my response to Jason's 5 points made in his evaluation of the usefulness the RDF proposal.

1. Advantages of adopting RDF were outlined in my proposal. Also, if we are to use RDF for 'semantic markup', the prudent course is to adopt RDF from the outset so that it *can be* adopted for 'semantic markup'. If we don't put rdf:ID's onto the clauses for instance, then it is hellishly a pain to reference those clauses when making statements "about" them.

2. It's likely a matter of perspective here. Mine is that it's relatively foolish for us to pretend that the vast vast majority of attorneys will ever want to enter XML directly -- they'll use products that'll choose the correct XML elements for them, that will then write the markup for them. For those that want to create a stanrdard for those who do want to write markup directly, I say: let us define how to create a useable XHTML file, and let people use the XHTML tools.... SURELY Word 200x is able to competently write XHTML -- really all that needs to be done is to define an attribute to stick onto the XHTML elements, and define "reserved" style and semantic names, and then be done with it. We don't need to get all bogged down in whether to have listitem elements and all that.

3. My proposal is misrepresented here. It uses rdf:Bag just for the captions found within the clause, not for the caption for the clause. It uses rdf:Seq for the clauses' paragraph content. Such explicitness about what content is ordered, and what is not, is deemed desirable even by the XML spec itself, which notes that one may not imply that the order of elements within an entity is equivalent to its order of presentation. The <rdf:Seq> element makes this fact explicit for the content of a Block; it is a best practice for XML markup.

4. I'm sure what is meant to be said here is that in the RDF proposal, it is *possible* for more than one caption to point at a given clause, thereby giving rise to an application-defined formatting decision about which caption to render. This doesnt trouble me in the slightest. Again, this all gets back to the perspective that we bring to table.

5. My proposal has been misunderstood. There is no "requirement" to specify formatting directives. In the absence of any directive, it defaults to a simple paragraph. Formatting considerations certainly run aplenty in your proposal, plus formatting is clearly the key issue addressed in the citation concerning a "pure" paragraph. I say, the exchanged markup should be able to simply say WHAT formatting is to be applied to the clause at the time it is layed out for presentation. The use of the <rdf:type> element accomplishes this; our job is merely to define a controlled vocabulary for those layouts that encompass 97% of formatting requirements.
==========

What are the real problems had with the RDF proposal? My guess is:

1. It uses 5 ! elements from a non-LegalXML established namespace -- the Seq/Alt/Bag, type, and li elements. Again, it's a matter of perspective.... I support creating a standard that provides clear guidelines for using elements in a document that are defined by other namespaces.

2. It places all Blocks into a list element, always as children of a <rdf:Seq>, meaning that authors have to specify both <rdf:Seq> and <rdf:li> in order to enter a (child) Block. But let's be real here. The content model in the RDF Proposal establishes "introducer" elements, such as <contains> - this is a natural element to choose when entering tags by hand. But then what? If we make the <rdf:Seq> element required content for the <contains> element, then most XML editors will *automatically* create the <rdf:Seq} element, so that's not a problem. The user is then shown a list with 2 elements: rdf:type and rdf:li. This is not a big deal !

3. It uses <rdf:type> to convey key information about an element, cleanly separating presentation from content. The value of the rdf:type points to a dictionary where we provide a formal definition. I really scratch my head trying to know what the meaning of "Item" is -- it's name is so overloaded ! In fact, I've only seen <item> elements used within container elements (such as <group>, from the , never as a stand-alone thing, because it means *nothing*

.... may as well say <gorp>.

4. In addition to the <rdf:li> element, the RDF approach uses a mixed-content element named <text> to hold PCDATA. The element name is lower-camel-case because text strings *are* attributes of the Block object.

5. It uses 3 elements as property introducers for a Block: <contains> for introducing the list of child Blocks; <for> for identifying the id of a Block to which a Caption pertains; and <using> for list bullets. The added clarity of these elements is apparently seen to be a burden, however when other information about the clause is to be exchanged, it will likely be necessary to add "extra" qualifying elements then -- why not have a consistent style of markup? If the answer is that no other information beyond the content of the Item itself is to be exchanged, then one has to ask, where is your architecture for exchanging all the OTHER information that everyone has requested, such as date clause inserted, who inserted, comments about the clause, relations to previous versions of the clause, and on and on -- how are these to be accommodated? For me, that's pretty easy -- use different introducer elements ! For instance, want to cite the <Block> that this block was copied from? Easy as pie: <is><Created><from rdf:resource='uri'/>.... want to know who inserted the clause? Easy as pie: <is><Inserted><by rdf:resource='uri'/>..... and so on.... the point being, taking the blinders off just a little to consider the larger picture of the data to be exchanged shows the shortcomings of a model that hasn't introducers such as <contains> to accommodate distinguishing metdata from content data.

6, The RDF proposal is keyed to a substantial *technical architecture* that is beyond that proffered as "Guidelines". The RDF proposal does not allow clause numbers for instance because those are in effect, required to be inserted by software (eg XSL) when laying out the document for presentation. The architecture I propose is one that established *different* standards for non-presentation material vs presentation versions of the contract. The architecture I propose *specifically* states that non-presentation material may be construed as neither a "contract offer" nor an "executed contract". Given that, the basic strategy for the design of our markup is to exchange information necessary for an XSL program to create XHTML, XSL-FO, or SVG/PDF material that could be construed as legally-signficant material. This means that the markup states only WHAT a formatting program is to do, not HOW to do it. To me, that's the major difference here: the RDF proposal by its nature is ACTIVELY, UNAMBIGUOUSLY, and CLEARLY stating that it is not presentation material that could be considered legally significant, while Jason's states that the "clause model can be rendered easily using CSS".

Both Jason's and Peter's proposals, however, create more questions architecturally than they answer. Is the XML document they are standardizing able to function as a presentation artifact or not? The answer is "maybe" -- maybe if there is no pagination, or page-specific information, maybe if there is no running header or footer, maybe if there is proper element ordering for their presentation, maybe if style attributes (class and style) are used, maybe if there is no content generation, maybe if there is a signature block, and so on.

SO what does that mean to us? It means that our most (imo) fundamental requirement as a LegalXML TC is not yet satisfied: creation of a standard that yields legal *contracts* whose clauses can be referenced by others, eg a judicial opinion. From a practical view, a judge or an attorney or a party needs to reference the final presentation material. They cannot reference pieces of a presentation datastream that is dynamically created by software and presented to a browser for presentation to parties. So, the material that is exchanged between the parties MUST be the datastream presented for presentation to a browser. Jason's and Peter's proposals are in great denial about this entire, crucial, topic. I, on the other hand, treat it squarely by saying that (1) the datastream in this proposal is for a contract proposal at most (2) the contract offer and executed contracts are different datastreams entirely, not a superset of the contract proposal.

==========

Though the number of elements in the Appendix Markup are the same (mine is 4% fewer), there are major areas that are yet to be undertaken in Jason's and Peter's proposals that are already covered by mine: (1) All kinds of lists that one finds in HTML -- dl, ul, sl (2) Boilerplate inclusions -- easy, use normal rdf mechanisms, as outlined; and (3) Unique identification of Blocks -- uses an IEEE & W3C standard URIs. Now, I'd certainly agree that my proposal can be sharpened (eg, allow Caption to be anywhere in a document)? I don't understand how this proposal from Jason and Peter are meant to build "consensus". However, in order to pursue consensus, let's ask, where do the proposals match?

1. <Item> and <Block> are the same functional element.
2. <List> and <rdf:Seq> are the same functional element
3. <ListItem> and <rdf:li> are close to the same functional element
4. <TextBlock> and <text> are close to the same functional element
5. <Title> and <Caption> are the same functional element
6. ID and rdf:ID are close to the same functional attribute

Differences are

1. <Block> is used, not <Item>, because we are fundamentally concerned with representing blocks of text, not items of ....what? Further, the term "item" is so heavily overloaded in our society, that it's rendered practically meaningless.
2. <rdf:Seq>/Bag/Alt are used, not <List>, because real benefits are had by adopting the simple RDF namespace.
3. <rdf:li> is used, not <ListItem>, because real benefits are had by adopting the simple RDF namespace. It also is named in a manner to (correctly) connote that the listitem is an attribute of its container.
4. I have no need for <Para>, since I already provide a <Block>. I don't see any gain from having a <Para> inside a <Block>, when they mean the same thing. That approach could cause formatting issues.
5. <text> is used, not <TextBlock>, because text content is a property (not an object) of a containing block (hence its lower camel case), and because it is shorter, less formal, and lacks the presentation notion that accompanies anything called a Block.
6. <Caption> is used, not <Title> for a name because the definition of "caption" is closely and unambiguously related to document publishing, while "title" is way overloaded. Further, I believe that according to the recommended rules about element naming, it would suggest that it should be called ItemTitle, right, since "Title" is applicable to many other contexts.
7. <XXX rdf:ID=''> is used, not <XXX ID=''> because noone likes html ids anymore, primarily because there is no operational requirement for it to be unique in any context but its parent element. We absolutely need to adopt URIs for clauses for citation purposes.

So, my proposal still stands for the group's consideration. Rather than creating a DTD in 2003 that, quite frankly, appears to be headed for an almost universal yawn, I suggest that we take advantage of a technology that is the bedrock of the Semantic Web, that does play well to our needs for semantic markup, and that is not nearly as onerous or useless as some people want to make it out to be. Here are Jason's example markups, in the RDF. Judge for yourself:

==========
Jason's Example 1:
==========

```
<Item ID="discretions">
        <Num>1.</Num>
        <Title><TextBlock>Exercise of Discretions</TextBlock></Title>
        <Para><TextBlock> ... </TextBlock></Para>
</Item>
```

RDF Treatment
==========

```
<Block rdf:ID='discretions'>
        <rdf:type rdf:resource='&dictionary;#FullyCaptionedParagraph'/>
        <text> ... </text>
        <is><Captioned><text>Exercise of Discretions</text></Captioned></is>
</Block>
```

==========
Jason's Example 2:
==========

```
<Item ID="ip">
        <Num>2.</Num><Title><TextBlock>Intellectual Property</TextBlock></Title>
        <Item ID="ip-own">
                <Num>2.1</Num>
                <Title><TextBlock>Ownership</TextBlock></Title>
                <Para><TextBlock> ... </TextBlock></Para>
        </Item>
        <Item ID="ip-ass">
                <Num>2.2</Num>
                <Title><TextBlock>Assignment</TextBlock></Title>
                <Para><TextBlock> ... </TextBlock></Para>
        </Item>
</Item>
```

RDF Treatment
==========

```
<Block rdf:ID='ip'>
        <rdf:type rdf:resource='&dictionary;#FullyCaptionedParagraph'/>
        <is><Captioned><text>Intellectual Property</text></Captioned></is>
        <contains>
                <rdf:Seq>
                <rdf:li> <Block rdf:ID='ip-own">
                                <rdf:type rdf:resource='&dictionary;#FullyCaptionedParagraph'/>
                                <text> ... </text>
                                <is><Captioned><text>Ownership</text></Captioned></is>
                         </Block>
                </rdf:li>
                <rdf:li> <Block rdf:ID="ip-ass">
                                <rdf:type rdf:resource='&dictionary;#FullyCaptionedParagraph'/>
                                <text> ... </text>
                                <is><Captioned><text>Assignment</text></Captioned></is>
                        </Block>
                </rdf:li>
                </rdf:Seq>
        </contains>
</Block>
```

==========
Jason's Example 3:
==========

```
<Item ID="termination">
        <Num>3.</Num>
        <Title><TextBlock>Termination </Title>
        <Para>
                <TextBlock> ... </TextBlock>
                <List>
                        <ListItem ID="term-a">
                                <Num>(a)</Num>
                                <Para><TextBlock> ... </TextBlock></Para>
                        </ListItem>
                        <ListItem ID="term-b">
                                <Num>(b)</Num>
                                <Para><TextBlock> ... </TextBlock></Para>
                        </ListItem>
                        <ListItem ID="term-c">
                                <Num>(c)</Num>
                                <Para><TextBlock> ... </TextBlock></Para>
                        </ListItem>
                </List>
        </Para>
</Item>
```

RDF Treatment
==========

```
<Block rdf:ID='termination'>
        <rdf:type rdf:resource='&dictionary;#FullyCaptionedParagraph'/>
        <is><Captioned><text>Termination</text></Captioned></is>
        <contains>
                <rdf:Seq>
                <rdf:li> <Block><text> ... </text></Block></rdf:li>
                <rdf:li>
                    <rdf:Seq>
                        <rdf:type rdf:resource='&dictionary;#ParenthesizedList'/>
                        <rdf:type rdf:resource='&dictionary;#LowerCasedList'/>
                        <rdf:li> ... </rdf:li>
                        <rdf:li> ... </rdf:li>
```

```
                    <rdf:li> ... </rdf:li>
                </rdf:Seq>
            </rdf:li>
            </rdf:Seq>
        </contains>
</Block>

================================================================================
Examples of block and inline quotations in RDF.
================================================================================
<Block rdf:ID='idfragment1'>
        <contains>
            <rdf:Seq>
            <rdf:li> <Quotation>
                        <rdf:type rdf:resource='&dictionary;#Block/>
                        <contains>
                            <rdf:Seq>
                            <rdf:li><Block><text> ... </text></Block></rdf:li>
                            <rdf:li><Block><text> ... </text></Block></rdf:li>
                            <rdf:li><Block><text> ... </text></Block></rdf:li>
                            </rdf:Seq>
                        </contains>
                        <is><Said><by rdf:resource='&whois;RobertFrost'/>
                    </Quotation>
            </rdf:li>
            <rdf:li> <Block><text>jason's trailing text part for a block quote</text></Block>
            </rdf:li>
            <rdf:li> <Block>
                        <contains>
                            <rdf:Seq>
                            <rdf:li> <Quotation>
                                        <rdf:type rdf:resource='&dictionary;#IncompleteQuote'/>
                                        <text> ... </text>
                                        <is><Said><by rdf:resource='&whois;RobertFrost'/>
                                    </Quotation>
                            </rdf:li>
                            <rdf:li> jason's trailing text for an inline quote, presented inline.</rdf:li>
                            <rdf:li> <Block>
                                        <text>trailing text for inline quote, shown as a block.</text>
                                    </Block>
                            </rdf:li>
                            </rdf:Seq>
                        </contains>
                    </Block>
            </rdf:li>
            </rdf:Seq>
        </contains>
</Block>
```