Clause Model Solution Proposal – “The Underlying Pattern”

Submitted by jharrop@speedlegal.com
Monday, July 07, 2003

Introduction

This proposal is based on concepts underlying the DTD which SpeedLegal (www.speedlegal.com) has been shipping for the past 3 years to its clients in various geographies to mark up their contracts.

The key contribution provided by this proposal is the insight that the clause structures as exemplified in the attachment to the Clause Model Requirements Document can be modeled by the following simple pattern:

> a "numbered object" comprising:
>
> - a heading
> - paragraphs of text (including lists)
> - nested numbered objects

This proposal demonstrates that that simple pattern works, and highlights some of the variations we might consider, while remaining true to the basic pattern

At its most basic, the pattern could be implemented as:

```
<!ELEMENT NumberedObject (Heading?, ClauseBodyPara*, NumberedObject*)>
```

I will be submitting a DTD survey shortly which examines a variety of DTDs to see whether they implement this pattern, and if so, how.

Element names

I'm relaxed as to what the element names end up being called. The element names contained in this proposal are not an essential part of it. Indeed, they don't meet requirement 6.

I expect the DTD survey to offer some suggestions in this regard.

Note: in parts of this document, and in particular in the DTD and sample markup attached, I use “Clause” in place of “NumberedObject”.

ClauseBodyPara

In this model, ClauseBodyPara is mixed content. It can contain the usual inline stuff, plus List (and, in a full implementation, table). Put another way, List always lives inside a paragraph, not between paragraphs.

It is worth considering whether the first ClauseBodyPara in a numbered object should have a different name.

Constraints

With the simple pattern:

<!ELEMENT NumberedObject (Heading?, ClauseBodyPara*, NumberedObject*)>

it is possible to have an empty NumberedObject, or one which contains nothing at this level (ie just a nested NumberedObject).

Neither of those pathological cases appears in the attachment to the Clause Model Requirements document, or for that matter, in the contracts and other business documents which I have reviewed.

For this reason, it is preferable to identify the sensible sub-models:

- Heading and nested NumberedObject
- ClauseBodyPara, with or without Heading and/or nested NumberedObject

That would look something like:

```
<!ELEMENT NumberedObject (
        (Heading, NumberedObject+)
        | (Heading?, ClauseBodyPara*, NumberedObject*)
)>
```

Recursive versus non recursive

The proposed model is recursive.

As I've said, I can see arguments for moving to a non-recursive model. Namely:
- ease of writing stylesheets
- were it to prove to be the case that authors have other preferred terminology

Having said that, a recursive model assists in meeting requirement 9, and also has application development benefits (requirement 8).

Container for nested objects

This proposal puts the nested numbered objects in a container called "Subclauses".

<!ELEMENT Clause (Heading?, ClauseBodyPara*, Subclauses?)>
<!ATTLIST Clause ID ID>

Equally, one could simplify/relax to something like:

<!ELEMENT Clause (Heading?, ClauseBodyPara*, Clause*)>

ie omit the extra container.

Lists versus Subclauses

There would be benefits in removing the distinction between List and Subclauses, and instead using a

common label (eg “SubStructure”):

- people sometimes argue about whether something is a List or Subclauses, and the distinction can be fuzzy (particularly if the list items have headings);
- re-use and cut/paste would be easier.

Against this is the idea that Lists are a distinct concept, and sometimes are "and", sometimes "or" (which could be captured as an attribute).

Clause Numbers

We'd expect a ClauseNumber element to also be available (probably optional), with a definition like:

<!ELEMENT ClauseNumber (#PCDATA)>
<!ATTLIST ClauseNumber FullForm CDATA #REQUIRED>

where @FullForm might be "21.4.2", and the #PCDATA might be "(ii)" ie what is to appear in the document.

So you'd get something like:

<!ELEMENT Clause (ClauseNumber?, Heading?, ClauseBodyPara*, Subclauses*)>

Fit to Requirements

Summary of Requirement	*Fit*
1. markup the core structures found in documents like Attachment 1.	**Meets**
2. represent the structured hierarchy of the content	**Meets.**
3. represent the benchmark contracts	**Meets*.** Note that an inline heading (eg in http://www.oasis-open.org/apps/org/workgroup/legalxml-econtracts/download.php/2092/2.doc) is modelled in a heading element outside the paragraph. * The TC’s collection of benchmark contracts needs to be completed and cleaned up.
4. define clause objects .. as self contained objects	**Meets.**
5. self contained markup of content so that [you can display the] text file to determine the terms of the contract	**Meets,** though an assessment of numbering and cross references is deferred until requirement 11 is fully specified.
6. must not use the following terms in element markup	**Does Not Meet.** Uses the word "clause". Easy to fix.
7. permit the markup of contract terms without inclusion of any	**Meets.**

legal semantic markup or annotation	
8. as simple as practicable to facilitate user training, support and application development	User training/support: **Meets – largely** Query whether the distinction between <list> and <subclauses> should be retained or not**.**
	Application development: **Meets**
9. re-use content in different levels of the hierarchy, without having to change the names of the elements	**Meets** except where you wish to change content from a list to subclauses or vice versa. This probably should be fixed.
10. allow clauses or other content to be incorporated into a document by reference	**Outside the scope of this proposal.** We have a mechanism for doing this in our application, but this proposal does not include it.
11. Once specific requirements for these features are determined....	[Deferred until requirements fully developed]

DTD and XML

Please see valid document, using a doctype with an internal subset only, attached (also provided as a separate .xml file).

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE clausemodel_jharrop_proposal [
    <!-- root element -->
    <!ELEMENT clausemodel_jharrop_proposal (Clause)+>
    <!-- the model proper -->
    <!ELEMENT Clause ((ClauseNumber?, Heading, Subclauses) | (ClauseNumber?, Heading?, ClauseBodyPara*, Subclauses?))>
    <!ATTLIST Clause
    ID ID #REQUIRED
>
    <!ELEMENT ClauseNumber (#PCDATA)>
    <!ATTLIST ClauseNumber
    FullForm CDATA #REQUIRED
>
    <!ELEMENT Heading (#PCDATA)>
    <!ELEMENT Subclauses (Clause)+>
    <!ELEMENT ClauseBodyPara (#PCDATA | List)*>
    <!-- Lists  -->
    <!ELEMENT List (ListEntry)+>
    <!ELEMENT ListEntry (ListNumber?, Heading?, ListBodyPara*)>
    <!ATTLIST ListEntry
    ID ID #REQUIRED
>
    <!ELEMENT ListNumber (#PCDATA)>
    <!ATTLIST ListNumber
    FullForm CDATA #REQUIRED
>
    <!ELEMENT ListBodyPara (#PCDATA | List)*>
]>
<clausemodel_jharrop_proposal>
    <!-- 1.    Provisions about the specification of colours in contracts -->
    <Clause ID="colour_specs">
        <ClauseNumber FullForm="1">1.</ClauseNumber>
        <Heading>Provisions about the specification of colours in contracts</Heading>
        <Subclauses>
            <!--1.1   Spectrum colours-->
            <Clause ID="colour_spectrum">
                <ClauseNumber FullForm="1.1">1.1</ClauseNumber>
                <Heading>Spectrum colours</Heading>
                <ClauseBodyPara>Here is a contrived, complex list structure using the spectrum colours and one or two others:
                    <List>
                        <ListEntry ID="colours_inc_red">
                            <ListNumber FullForm="1.1.1">(a)</ListNumber>
                            <ListBodyPara>red,</ListBodyPara>
                        </ListEntry>
                        <ListEntry ID="colours_inc_orange">
                            <ListNumber FullForm="1.1.2">(b)</ListNumber>
                            <ListBodyPara>orange,</ListBodyPara>
                        </ListEntry>
                        <ListEntry ID="colours_inc_yellow">
                            <ListNumber FullForm="1.1.3">(c)</ListNumber>
                            <ListBodyPara>yellow,</ListBodyPara>
                        </ListEntry>
                        <ListEntry ID="colours_inc_green">
                            <ListNumber FullForm="1.1.4">(d)</ListNumber>
                            <ListBodyPara>green,</ListBodyPara>
                        </ListEntry>
                        <ListEntry ID="colours_inc_blue">
                            <ListNumber FullForm="1.1.5">(e)</ListNumber>
                            <ListBodyPara>blue, including:
                                <List>
                                    <ListEntry ID="colours_inc_blue_pale">
                                        <ListNumber FullForm="1.1.5.1">(i)</ListNumber>
                                        <ListBodyPara>pale blue,</ListBodyPara>
                                    </ListEntry>
                                    <ListEntry ID="colours_inc_blue_dark">
                                        <ListNumber FullForm="1.1.5.2">(ii)</ListNumber>
                                        <ListBodyPara>dark blue,</ListBodyPara>
                                    </ListEntry>
                                </List>
                                but excluding violet,
                            </ListBodyPara>
                        </ListEntry>
                        <ListEntry ID="colours_inc_indigo">
                            <ListNumber FullForm="1.1.6">(f)</ListNumber>
                            <ListBodyPara>indigo, and </ListBodyPara>
                        </ListEntry>
                        <ListEntry ID="colours_inc_violet">
                            <ListNumber FullForm="1.1.7">(g)</ListNumber>
                            <ListBodyPara>violet,</ListBodyPara>
                        </ListEntry>
```

```
                    </List>
                    from which all colours can be derived.
                </ClauseBodyPara>
            </Clause>
            <!-- 1.2  CMYK colours -->
            <Clause ID="colours_cmyk">
                <ClauseNumber FullForm="1.2">1.2</ClauseNumber>
                <Heading>CMYK colours</Heading>
                <ClauseBodyPara>CMYK colours (cyan, magenta, yellow and black) are normally specified for inputs to colour printing processes.
                </ClauseBodyPara>
            </Clause>
            <!-- 1.3  RGB colours -->
            <Clause ID="colours_rgb">
                <ClauseNumber FullForm="1.3">1.3</ClauseNumber>
                <Heading>RGB colours</Heading>
                <Subclauses>
                    <Clause ID="colours_rgb1">
                        <ClauseNumber FullForm="1.3.1">1.3.1</ClauseNumber>
                        <ClauseBodyPara>RGB colour (red, green, brown) specifications are used for computer screen displays.</ClauseBodyPara>
                    </Clause>
                    <Clause ID="colours_rgb2">
                        <ClauseNumber FullForm="1.3.2">1.3.2</ClauseNumber>
                        <ClauseBodyPara>Using only these 3 colours, you can specify any colour.</ClauseBodyPara>
                    </Clause>
                    <Clause ID="colours_rgb3">
                        <ClauseNumber FullForm="1.3.3">1.3.3</ClauseNumber>
                        <ClauseBodyPara>The number of colours you can specify depends on the colour depth available. For example:
                            <List>
                                <ListEntry ID="colours_rgb31">
                                    <ListNumber FullForm="1.3.3.1">(a)</ListNumber>
                                    <ListBodyPara>8 bit colour can render 256 colours</ListBodyPara>
                                </ListEntry>
                                <ListEntry ID="colours_rgb32">
                                    <ListNumber FullForm="1.3.3.2">(b)</ListNumber>
                                    <ListBodyPara>16 bit colour can render 65,536 colours.</ListBodyPara>
                                </ListEntry>
                            </List>
                        </ClauseBodyPara>
                    </Clause>
                </Subclauses>
            </Clause>
            <!-- 1.4  Using black and white -->
            <Clause ID="colours_blackwhite">
                <ClauseNumber FullForm="1.4">1.4</ClauseNumber>
                <Heading>Using black and white</Heading>
                <Subclauses>
                    <Clause ID="colours_greyscale">
                        <ClauseNumber FullForm="1.4.1">1.4.1</ClauseNumber>
                        <Heading>Greyscale</Heading>
                        <ClauseBodyPara>The number of greys depends on the available colour depth, as for other colours.
                    </ClauseBodyPara>
                    </Clause>
                    <Clause ID="colours_mono">
                        <ClauseNumber FullForm="1.4.2">1.4.2</ClauseNumber>
                        <Heading>Black and white</Heading>
                        <ClauseBodyPara>This is really called monochrome. You can specify either:
                            <List>
                                <ListEntry ID="colours_mono_black">
                                    <ListBodyPara>black, or</ListBodyPara>
                                </ListEntry>
                                <ListEntry ID="colours_mono_white">
                                    <ListBodyPara>white.</ListBodyPara>
                                </ListEntry>
                            </List>
                        </ClauseBodyPara>
                    </Clause>
                </Subclauses>
            </Clause>
        </Subclauses>
    </Clause>
    <!-- 2.    Colour profiles-->
    <Clause ID="colour_profiles">
        <ClauseNumber FullForm="2">2.</ClauseNumber>
        <Heading>Colour profiles</Heading>
        <ClauseBodyPara>One thing to remember is that when working with colours, always use a colour profile that is available for your display or
output device.          This will ensure you achieve the most consistent results.
        </ClauseBodyPara>
    </Clause>
</clausemodel_jharrop_proposal>
```