The Clause Model Pattern

A Survey of Existing DTDs

jharrop@speedlegal.com
Draft 1: Tuesday, July 08, 2003

Introduction

The eContracts TC needs a grammar which captures clause structure: see http://www.oasis-open.org/committees/download.php/2390/ClauseModel-Requirements-1.pdf

The Clause Model Requirements document includes an example of the type of material we need to be able to represent. Essentially, we need a model which provides a "numbered object" comprising:

- a heading
- paragraphs of text (including lists)
- nested numbered objects

It is the regularity of this ordering which creates the structure which it is valuable to model.

For example, the content model for this numbered object can and should say that if a heading appears, the heading must come first.

Ideally, we would re-use an appropriate model from an existing DTD. Re-use would confer several benefits:

- less effort in design process
- higher quality, since existing DTDs have already been through a release process
- easier to identify shortcomings in the model
- existing tool support (editing environments, stylesheets etc)
- existing body of skilled users

Needless to say, the benefits of reuse should not be overstated. One of the reasons there are so many XML DTD's and schemas is that the benefits of having a model which is a great fit for a particular problem domain often outweigh the inconvenience of an inconvenient fit provided by some pre-existing DTD.

In particular, the benefits of existing tool support are easily overstated, since provided we use common patterns in our DTD, many existing tools can be readily customised using their built in facilities for doing that.

This document examines various document types, seeking to identify models we might re-use.

- XHTML
- OpenOffice
- WordML
- Docbook
- UBL
- 5629A
- LSSA XML Schema
- Europe Lex
- DITA
- NLM Journal Publishing DTD

A key question for consideration is whether we can use a strict subset of a given DTD, or, at least, fit entirely within the extension mechanisms it provides. If we can't do this, our documents will not be valid against the original DTD, and some interoperability will be lost.

This document is a work in progress. Comments are welcome.

For most of these DTDs, my observations are based on a quick examination of the DTD itself – something in the order of 1-2 hours. I do not work with any of them regularly, and therefore might well have missed something. Corrections therefore will not come as a surprise.

Summary

None of the DTDs examined provide a complete "off-the-shelf" solution to our needs.

The rest of this section categorises them broadly, and attempts to identify what they might offer us.

DTD's with a parameter entity based extension/customisation mechanism:
- XHTML Modularization
- DocBook
- NLM Journal Publishing

With the exception of DocBook (which offer both recursive and explicitly numbered sections), these DTDs provide recursive models.

These DTDs offer some candidate names for our numbered object:

- div (xhtml)
- sec1-5, or section (DocBook)
- sec (NLM Journal Publishing)

but, except for DocBook, can't quite model the attachment.

There is little to be said for adopting the whole of one of these DTDs, and then customising it down to the small subset which we need. Better to start with the core piece we need (perhaps drawing on these DTDs for naming inspiration), and consider whether an extension/customization mechanism could/should be bolted on (note that the Clause Model Requirements Document contains no such requirement).

Moving on, IBM's Darwin Information Typing Architecture (DITA) stands out because of its unique specialization mechanism. Although DITA as it stands makes title compulsory, it appears to be worthy of more detailed examination. Potentially, it offers an alternative approach to a "one size fits all" name for our numbered object.

5629A does not have an extension/customisation mechanism, and given its specialized target audience, is not suitable for our purposes.

Word processor DTD/XSD:
- OpenOffice
- WordML

These don't offer anything of particular value for the basic clause model, although they may be of assistance for requirement 11. In particular, their styling/numbering mechanisms are to be avoided.

Finally, several of the efforts surveyed do not provide input which is relevant to our clause modeling exercise:
- UBL
- LSSA XML Schema

- Europe LexData / LexML

Details

XHTML

XHTML Abstract Modules

The modularization of XHTML defines XHTML modules and a general modularization methodology in order to ease the development of document types that are based upon XHTML.

Appendix E explains how to develop a DTD from the modules: http://www.w3.org/TR/xhtml-modularization/dtd_developing.html#s_developingdtds

The modularization includes Text and List modules:

http://www.w3.org/TR/xhtml-modularization/abstract_modules.html#s_textmodule

http://www.w3.org/TR/xhtml-modularization/abstract_modules.html#s_listmodule

The Text module defines:

- Block Structural (div, p)
- Block Phrasal (h1-h6, blockquote, pre)
- Inline Structural (br, span)
- Inline Phrasal

The List module defines:

- ol, ul: (li)+
- li
- dl: (dt | dd)+

(see appendix F: http://www.w3.org/TR/xhtml-modularization/dtd_module_defs.html#a_dtdsupport

Of interest:

- div, li, dt: (#PCDATA | %Flow.mix;)*
- h1, p, span, dd: (#PCDATA | %Inline.mix;)*

As a guide, the Document Model Modules for XHTML Basic 1.0 and XHTML 1.1 (xhtml-basic10-model-1.mod, xhtml11-model-1.mod) defines:
- Flow.mix: (Heading, List, Block Structural, Block Phrasal, Table, Inline Structural, Inline Phrasal)
- Inline.mix: (Inline Structural, Inline Phrasal)

The element which comes closest to modeling our numbered object is <li> or <div>.

<li> and <div> are close since they use Flow.mix which can contain a heading, paragraphs of text, and nest themselves.

<li> and <div> are not close enough:

- they can directly contain #PCDATA (#PCDATA should only be allowed in the heading and the paragraphs of text).

- they do not mandate that if a heading is to appear, it appears first.

These objections could be overcome by redefining them. At the very least, if we were using <div> for the numbered object, we'd want to redefine <div> to disallow #PCDATA. ie div (%Flow.mix;)*

We could use <p> for the paragraphs of text.

XHTML is clean because it clearly distinguishes between block and inline - inline can never contain block. However, this means <p> can't represent a paragraph which contains a sentence which includes a list (which is bad).

In addition, there is still an uncomfortable mixture of recursive and non-recursive here.

If we favoured a recursive model (that is, the container for the numbered object has the same name (eg <div>) at each level), then a heading (if one appears) in each level should also have the same name (eg <heading>). Instead of defining <heading>, we could use <h1> irrespective of level (ie and remove <h2> - <h6>).

Alternatively, if we favoured a non-recursive model, we'd keep h1-h6, and replace div with div1-div6.

Summary:

Pattern	**Candidate Elements**	**Comments**
Numbered object container	<div>	- #PCDATA allowed - Doesn't enforce order (eg heading first) - could use <li> instead, since its exactly the same
Heading?	<h1>	- Use <h1> at each level if we want a recursive model. Delete <h2>-<h6>
Paragraph*	<p>	- **Doesn't meet requirement 2**: Needs to be redefined to allow a list, unless we can live without lists inside sentences.
List*	<ol> <li>	- better than <div>, since <ol> provides a container
Container for nested numbered objects?	[none]	- Unless we used <li> for the numbered object, in which case the container would be <ol>
Numbered object*	<div>	See comments above

Consideration	**For**	**Against**
Additional infrastructure provided by DTD	○ images, tables ○ CSS (though this can be used with other XML)	○
End user skill set		No familiarity. Unless we redefine <div> to specify order (h1?, p*, div*) , things are hard for the user.
Developer skill set	○ widespread familiarity with HTML ○	○ low familiarity with XHTML modularization
Tool support – editing environment	○ widespread tool support	○ catalog file support desirable in editing tool
Tool support – stylesheets	○ widespread tool support ○ display in web browsers	
Extension mechanism	This DTD includes a well conceived extension mechanism. ○ A document type based on XHTML modules could be defined.	However, not quite a strict subset of XHTML, since <p> needs to allow a list inside it (and then there's the material which comes before and after the clause model – the so-called top and tail).
Is DTD being actively developed under an open process?	○ Yes, W3C	
Position in Lifecyle	○ W3C Recommendation 10 April 2001	
Licence terms	○ [to be completed]	

We want to make our content model easy to understand so as to fuel acceptance, which suggests we'd want to present it as a single file (rather than a complex abstraction). In other words, presenting as our primary deliverable a DTD which uses XHTML modules would be a mistake. That is not to say that that could not be a secondary deliverable. To the extent that this comment is valid, it applies to certain of the other DTDs (eg DocBook, DITA, and the Journal Publishing DTD).

Key Contribution	Value
CSS	low, since this can work with other DTDs as well
easy to transform to HTML	low, since: - this is becoming less necessary since

	browsers become able to display XML - a transform to HTML is simple enough anyway

OpenOffice

office.mod:

<!ENTITY % body includes "...text:tracked-changes,%text-decls;,text:h|text:p|text:list|table:table|draw:page|text:section|text:table-of-content| ..."

<!ELEMENT office:body %body;>

text.mod
http://xml.openoffice.org/source/browse/xml/xmloff/dtd/text.mod?rev=1.54&content-type=text/x-cvsweb-markup

All text is included in the paragraph elements, which may be either a plain paragraph text:p or a header text:h. I understand a third paragraph element numbered-paragraph will also be introduced following work in OASIS Open Office TC.

<!ENTITY % inline-text "(#PCDATA | %inline-text-elements;)*">

<!ELEMENT text:p %inline-text;> , so lists aren't allowed in a text:p paragraph (violating requirement 2).

<!ELEMENT text:h %inline-text;>
<!ATTLIST text:h text:level %positiveInteger; "1"> (yuck)

First, consider text:section as a candidate for our numbered object. Failing that, text:list (see below).

```
<!ENTITY % sectionAttr "text:name CDATA #REQUIRED
            text:style-name %styleName; #IMPLIED
            text:protected %boolean; 'false' ">
```

<!ELEMENT text:section ((text:section-source|office:dde-source)?, %sectionText;) >

where %sectionText; is something like:

(text:h|text:p|text:list| table:table|text:section|text:table-of-content|text:illustration-index|text:table-index|text:object-index|text:user-index|text:alphabetical-index|text:bibliography|text:index-title|text:change | text:change-start | text:change-end)*

<!ATTLIST text:section %sectionAttr;>

<!ATTLIST text:section text:display (true|none|condition) "true">

<!ATTLIST text:section text:condition %formula; #IMPLIED>

<!ATTLIST text:section text:protection-key CDATA #IMPLIED>

```
<!ATTLIST text:section text:is-hidden %boolean; #IMPLIED>
```

```
<!ELEMENT text:section-source EMPTY>
<!ATTLIST text:section-source xlink:href %string; #IMPLIED>
<!ATTLIST text:section-source xlink:type (simple) #FIXED "simple">
<!ATTLIST text:section-source xlink:show (embed) #FIXED "embed">
<!ATTLIST text:section-source text:section-name %string; #IMPLIED>
<!ATTLIST text:section-source text:filter-name %string; #IMPLIED>
```

So note in passing that text:section-source addresses requirement 10 (incorporate content by reference).

From the OASIS Open Office TC work, I understand 2 models for lists will be supported.

The first approach is <text:list> - the number of <list> ancestor elements determines the paragraph's level.

Something like:

```
<!ENTITY % list-items "((text:list-header,text:list-item*)|text:list-item+)">
<!ELEMENT text:list %list-items;>

<!ELEMENT text:list-header (text:p|text:h)+>
<!ELEMENT text:list-item (text:p|text:h|text:list)+>
```

So, the way to represent clause structure in Open Office might be to use <text:list>:

- list-header is not useful, since it is a header on the list, not each list item

- list-item permits a text:p or a text:list to come before a text:h ! It also permits more than one text:h

- since tables and images don't appear in the content model for lists or text:p or text:h, the model would need to be extended.

As such, text:list is not a particularly good fit for our needs.

The second approach to lists (noted for completeness) is a <numbered-paragraph> that has an attribute specifying the level of the numbering. This approach is not useful to us, since the original document structure is not maintained (requirement 2).

OpenOffice may have something to contribute for footnotes/endnotes, table of contents, change tracking etc, but that can be considered later.

Summary:

Pattern	**Candidate Elements**	**Comments**
Numbered object container	<text:section>	- also partially addresses requirement 10 (incorporate content by reference) - Doesn’t enforce order (eg heading first) - better than text:list though, since it would allow tables

Heading?	<text:h>	
Paragraph*	<text:p>	- **Doesn't meet requirement 2**: Needs to be redefined to allow a list, unless we can live without lists inside sentences.
List*	<text:list> <text:list-item>	
Container for nested numbered objects?	[none]	
Numbered object*	<text:section>	

Conclusion: a document type based on OpenOffice modules could be defined. However, not quite a strict subset, since, as similar to XHTML, <text:p> needs to allow a list inside it.

Consideration	**For**	**Against**
Additional infrastructure provided by DTD	○ images, tables ○ style and numbering mechanism ○ footnotes/endnotes, table of contents, change tracking ○ content by reference	○
End user skill set		No familiarity. Unless we redefine <text:section> to specify order (text:h?, text:p*, text:section*) , things are hard for the user.
Developer skill set		○ little familiarity
Tool support – editing environment	○ widespread support in generic tools (albeit not WYSIWYG)	○ catalog file support desirable in editing tool
Tool support – stylesheets		
Extension mechanism	○ To be determined.	
Is DTD being actively developed under an open process?	○ Yes, Oasis	

Position in Lifecyle	○ Pre-existing specification submitted to Oasis ○ TC making relatively minor alterations to that specification	
Licence terms	Can freely modify under: ○ LGPL, or ○ Sun Industry Standards Source License Version 1.1	

Key Contribution	Value
?	?
	-

WordML and friends

WordML, is part of the XML format used in Word 2003 where no customer defined schema has been specified. The structure and design of the Microsoft Word XML document schema is similar to RTF.

The schema http://schemas.microsoft.com/office/word/2003/2/wordml (not DTD) can be found in the publicly available Microsoft Word XML Content Development Kit Beta 2 which can be downloaded from http://download.microsoft.com/download/4/9/7/49799b71-5502-40c6-b7ce-c791f87f65cd/xmlcdkb2.msi

Sample documents contained in the Content Development Kit (eg DocLibrary\OfficeML\Contract.xml) declare various namespaces:

```
xmlns:w="http://schemas.microsoft.com/office/word/2003/2/wordml"
xmlns:v="urn:schemas-microsoft-com:vml"
xmlns:w10="urn:schemas-microsoft-com:office:word"
xmlns:SL="http://schemas.microsoft.com/schemaLibrary/2003/2/core"
xmlns:aml="http://schemas.microsoft.com/aml/2001/core"
xmlns:wx="http://schemas.microsoft.com/office/word/2003/2/auxHint"
xmlns:o="urn:schemas-microsoft-com:office:office"
```

but the CDK only includes a schema and documentation for wordml.

The paragraph element p contains a paragraph properties element pPr, and elements for “runs of text” r. The text itself appears in a t element in the r.

For example:

```
<w:p>
    <w:pPr>properties go here</w:pPr>
    <w:r>
        <w:t>This clause is a single paragraph</w:t>
    </w:r>
</w:p>
```

According to XMLCDK.doc, “A Paragraph element can also contain another Paragraph element — for example, if a paragraph contains an inline textbox, that textbox may be made up of multiple paragraphs.”. However, it appeared to me from the WordML schema that neither w:p nor w:r can contain a nested w:p.

Interestingly, DocLibrary\OfficeML\Contract.xml uses <wx:sect> and <wx:sub-section>, but there is no documentation I could find for the auxHint schema..Nevertheless, the document at http://www.xmlw.ie/aboutxml/wordml.htm suggests that <wx:sub-section> can be nested, which is good for our purposes.

.

According to XMLCDK.doc, “Lists are actually just Paragraphs. A list element is an individual paragraph that references a particular list style and the list level.”

Summary:

Pattern	**Candidate Elements**	**Comments**
Numbered object container	<wx:sub-section>	- no documentation so this is somewhat speculative
Heading?	<w:p> <w:pPr>properties go here</w:pPr> <w:r> <w:t>My Heading</w:t>	- no special element for heading?
Paragraph*	<w:p> <w:pPr>properties go here</w:pPr> <w:r> <w:t>My Paragraph</w:t>	- **Doesn’t meet requirement 2**: unless the list items were assumed to be block level, and the paragraph containing the list was represented by wx:sub-section; which is not good enough (the same sort of ugliness could address the deficiency in XHTML and OpenOffice)
List*	<w:p> <w:pPr>properties go here</w:pPr> <w:r> <w:t>My list item</w:t>	
Container for nested numbered objects?	[none]	
Numbered object*	<wx:sub-section>	

Considerations:

Consideration	**For**	**Against**
Strict subset of DTD	○	○ Not possible

Additional infrastructure provided by DTD	o images, tables o style and numbering mechanism o footnotes/endnotes, table of contents, change tracking o content by reference	o
End user skill set		No familiarity. Hard for the user, since the format is verbose and there is no distinction between heading and paragraph..
Developer skill set		o little/no familiarity
Tool support – editing environment		o limited support (other than in certain versions of Word 2003)
Tool support – stylesheets		
Extension mechanism	o To be determined	
Is DTD being actively developed under an open process?	o	No
Position in Lifecyle	o Will be implemented in Office 2003	
Licence terms	o	Unknown

Key Contribution	Value
?	?
	-

Conclusion: WordML is not a good basis for our clause model. Depending on clarification of the licence terms, it may be worth consulting for some of the items in requirement 11 (eg footnotes).

Docbook

http://www.oreilly.com/catalog/docbook/chapter/book/section.html summarises the position in the following terms:

"Section is one of the top-level sectioning elements in a component. There are three types of sectioning elements in DocBook:

* Explicitly numbered sections, Sect1…Sect5, which must be properly nested and can only be five levels deep.

* Recursive Sections, which are an alternative to the numbered sections and have unbounded depth.

* SimpleSects, which are terminal. SimpleSects can occur as the "leaf" sections in either recursive sections or any of the numbered sections, or directly in components.

Sections may be more convenient than numbered sections in some authoring environments because they can be moved around in the document hierarchy without renaming."

So let us examine "explicitly numbered sections" and "recursive sections" in turn. I don't consider simplesect any further than to note its definition:

```
        <!ELEMENT simplesect %ho; ((%sect.title.content;), (%divcomponent.mix;)+)
            %ubiq.inclusion;>
```

<u>Explicitly Numbered Sections</u>

<!ELEMENT sect1 %ho; (sect1info?, (%sect.title.content;), (%nav.class;)*,
    (((%**divcomponent.mix**;)+,
    ((%refentry.class;)* | **sect2* | simplesect***))
    | (%refentry.class;)+ | sect2+ | simplesect+), (%nav.class;)*)

and similar for sect2-sect5.

where:

<!ENTITY % divcomponent.mix
    "%list.class; |%admon.class;
    |%linespecific.class; |%synop.class;
    |%**para.class**; |%**informal.class**;
    |%formal.class; |%compound.class;
    |%genobj.class; |%descobj.class;
    |%ndxterm.class; |beginpage
        %forms.hook;
    %local.divcomponent.mix;">

```
<!ENTITY % local.informal.class "">
<!ENTITY % informal.class
    "address|blockquote
        |graphic|graphicco|mediaobject|mediaobjectco
        |informalequation
    |informalexample
        |informalfigure
        |informaltable %local.informal.class;">
```

```
<!ENTITY % local.para.class "">
<!ENTITY % para.class
    "formalpara|para|simpara %local.para.class;">
```

The para.class entity defines three types of paragraph: formalpara, para, and simpara.

```
<!ELEMENT formalpara %ho; (title, (%ndxterm.class;)*, para)>

<!ELEMENT para %ho; (%para.char.mix; | %para.mix;)*>

<!ELEMENT simpara %ho; (%para.char.mix;)*>
```

Para is a normal paragraph, and can include lists since it uses %para.mix:

```
<!ENTITY % local.para.mix "">
<!ENTITY % para.mix
        "%list.class;           |%admon.class;
        |%linespecific.class;
                    |%informal.class;
        |%formal.class;
        %local.para.mix;">
```

A formalpara can have a title, but we probably don’t need that, since a section provides the title.

A simpara can’t have a list in it, since it doesn’t use %para.mix.

So, we’d probably use <para>.

```
<!ENTITY % local.list.class "">
<!ENTITY % list.class
        "calloutlist|glosslist|itemizedlist|orderedlist|segmentedlist
        |simplelist|variablelist %local.list.class;">
```

An itemized list:

```
<!ELEMENT itemizedlist %ho; (blockinfo?, (%formalobject.title.content;)?,
                (%listpreamble.mix;)*, listitem+)>
```

is sufficient for our purposes.

The content model for listitem includes the three types of paragraph, ItemizedList and Table. To put a heading on a list item, we’d use formalpara?

Summary for “explicitly numbered sections”:

Pattern	**Candidate Elements**	**Comments**
Numbered object container	sec(n)	
Heading?	title	
Paragraph*	para	
List*	itemizedlist	Need to use formalpara to put a heading on a list item.
Container for nested numbered objects?	[none]	
Numbered object*	sec(n+1)	

Recursive sections

This is the other model DocBook offers us.

```
<!ENTITY % local.section.class "">
<!ENTITY % section.class    "section %local.section.class;">

<!ELEMENT section %ho; (
                                        sectioninfo?,
                                        (%sect.title.content;),
                                        (%nav.class;)*,
                                        (
                                            ( (%divcomponent.mix;)+, … )
                                        …|(%section.class;)+
                                              ), ..)
                                         %ubiq.inclusion;>
```

so we can do:

```
  <section>
     <title>....</title>
     <para>...</para>
     <para>...</para>
     <section>....</section>
     <section>....</section>
  </section>
```

Summary for “recursive sections”:

Pattern	**Candidate Elements**	**Comments**
Numbered object container	section	
Heading?	title	
Paragraph*	para	
List*	itemizedlist	Need to use formalpara to put a heading on a list item.
Container for nested numbered objects?	[none]	
Numbered object*	section	

Considerations applicable to both sectioning models:

Consideration	**For**	**Against**
Strict subset of DTD	○ Possible	○

Additional infrastructure provided by DTD	○ images, tables ○ (style and numbering mechanism is left to stylesheets) ○ footnotes/endnotes, table of contents	○ no change tracking?
End user skill set		No familiarity in our target user group.
Developer skill set	Well documented DTD	○
Tool support – editing environment		○
Tool support – stylesheets	A set of stylesheets exists	
Extension mechanism	DocBook is designed to be easy to modify	
Is DTD being actively developed under an open process?	○ Yes, Oasis: http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=docbook	
Position in Lifecyle	○	Reaching end of lifecycle. Refactoring/re-thinking is likely to commence soon: http://www.xmlhack.com/read.php?item=1990
Licence terms	○	

Key Contribution	Value
?	?
	-

UBL

UBL is about creating standard payloads for ebXML. "concrete standard XML syntax for business"

- The 20% of documents and business objects actually used by 80% of electronic business partners:
  - Procurement
    - Purchase Order, P.O. Response, P.O. Change
  - Materials management
    - Advance Ship Notice, Planning Schedule, Goods Receipt
  - Payment
    - Commercial Invoice, Remittance Advice

Transport/logistics
  Consignment Status Request, Consignment Status Report,
  Bill of Lading
Catalogs
  Price Catalog, Product Catalog
Statistical reports
Accounting Report

- Library of standard XML business information entities (BIEs)

- Context methodology to make the standard documents interoperate across industries

Library Content SC: Defines business documents and a library of XML and ebXML CCTS based building blocks (chair: Tim McGrath <tmcgrath@portcomm.com.au>)

The 0p70 Library Content public review (http://www.oasis-open.org/committees/ubl/lcsc/0p70/) addresses the trading cycle from Order to Invoice between Buyer and Seller. It includes specifications for:

- Order
- Order Response (simple)
- Order Response (complex)
- Order Cancellation
- Despatch Advice
- Receipt Advice
- Invoice

Only passing reference is made to legal contract/trading partner agreement (section 5.3.4, 5.3.5). However, the Order.xsd references cat:Contract, cat:SalesConditions and cat:DeliveryTerms, where cat is UBL's CommonAggregateTypes namespace.

I found Java classes representing these types at www.softml.net: they are not useful for present purposes. [Note: some of the other types will worth exploring further for our semantic/non-structural work eg PaymentTerms, Item, DeliverySchedule.]

In conclusion, UBL has nothing to offer us by way of assistance with our clause model.

Interestingly, Annex D.4 "Formatting specifications and stylesheets" says:

"This section contains examples of formatting specifications and "stylesheets" that can be used to display instances of UBL schemas in human-readable form. Presentational semantics have not been formalized in this version of the UBL schema library, and they may never be formalized due to differing international requirements and conventions for the presentation of information found in business documents."

The sample stylesheets use XSLT and XSL-FO.

5629A

5629A Core DTD

from http://www.defence.gov.au/dps/dpstoc.html

SUBGRP (Sub Group)

"Identifies a sub group of numbered paragraphs [PARA0] within the body of the document. A Sub Group is the structural container analogous to a Small Shoulder Paragraph Heading, a upper-lower case left aligned shoulder heading. The reason a separate container element is used for small shoulder headings is that according to 5629A numbered paragraphs may have in-line or small shoulder headings, and that small shoulder headings apply until the next heading of equal or greater weight. Thus, several numbered paragraphs may occur within a subgroup, and a separate container is required in order to avoid an infinitely recursive structure. A subgroup is used to introduce a number of paragraphs relating to the same subject."

```
<!ELEMENT SUBGRP (TITLE,SHORTTITLE?,PARA0+)>
```

Note that any SUBGRP doesn’t directly allow nested objects after PARA0, so any nesting will need to be inside PARA0.

PARA0 (Primary Paragraph)

"Identifies a primary numbered paragraph in the document's structure. According to the standard: The paragraph is essentially a unit of thought. It shall be homogenous in subject matter and sequential in treatment. Where desirable, paragraphs may be broken down into sub-paragraphs which themselves may be broken down into sub-sub-paragraphs. This structural container envelopes all the components of the numbered paragraph including: warnings, cautions, notes, the title (if applicable) and text of the paragraph, and any subparagraphs or procedural steps. The [TITLE] of the numbered paragraph is generally intended for inline paragraph titles and will generally appear after any warnings, cautions or notes that may need to precede the title and/or text of numbered this paragraph. When the applicable publishing standard allows the use of Small Shoulder Paragraph Headings these are to apply until the next heading of equal or greater level. Thus, in effect a paragraph heading may apply to more than one numbered paragraph - so Small Shoulder Paragraph Headings have been implemented as Sub Group [SUBGRP] structural containers that may hold a number of (numbered) paragraphs, ie [PARA0]s."

```
<!ELEMENT PARA0
(
   (
      (
         (TITLE,SHORTTITLE?,WARNING*,CAUTION*,NOTE*)
        |(
            (
               (WARNING+,CAUTION*, NOTE*)
               |(CAUTION+,NOTE*)
               |NOTE+
             )
            ,(TITLE,SHORTTITLE?)?
             )
         )?
        ,PARA+,NOTE*
        , (STEP1,STEP1+)?
      ),
      (SUBPARA1.GRP|SUBPARA1)*
   )>
```

Note the hierarchy is non recursive - SUBPARA1-SUBPARA3 are the respective levels of subordinate paragraph, and a .GRP container is available but optional (!).

PARA (Paragraph Text)

"Identifies a basic unit of paragraph text. This is a low level textual container that basically corresponds to a blob of text. It may occasionally be preceded by inline items such as paragraph or figure numbers and/or inline titles."

```
<!ELEMENT PARA
(#PCDATA|CHANGE|XREF|INDXFLAG|VERBATIM|EMPHASIS|APPLICABIL|GRAPHIC|
                    EXTREF|FIGURE|TABLE)*>
```

So tables and graphics appear inside a paragraph.

5629a seems to say:

"1. Clause Title

Clause Body text paragraph of words of words of words of words of words of words of words of words of words of words of words of words of words of words of words"

should be modelled as:

```
<SUBGROUP>
   <TITLE>Clause Title</TITLE>

   <PARA0>
   <PARA>Clause Body text paragraph of words  of words of words of words of words of words of words
of words of words of words of words of words of words of words of words</PARA>
</PARA0>
</SUBGROUP>
```

as compared to:

"1. [Clause Title] Clause Body text paragraph of words of words of words of words of words of words of words of words of words of words of words of words of words of words of words"

which should be modelled as:

```
<PARA0>
   <TITLE>ClauseTitle</TITLE>
   <PARA>Clause Body text paragraph of words  of words of words of words of words of words of words
of words of words of words of words of words of words of words of words</PARA>
</PARA0>
```

However, it seems a bit odd to use SUBGROUP this way, since the nested objects live inside PARA0. Accordingly, we won't, and we leave the location of the heading to the output stylesheets to determine.

What about lists? STEP1 is about as close as we get.

STEP1 (First Level Procedural Step)

"Identifies a first level procedural step. A step is a single operation in a sequence of events, such as in maintenance instructions. Further breakdowns within the sequence are identified by second level steps, ie [STEP2]. A step is treated as a sub-paragraph [SUBPARA1] for presentation purposes, and second level steps as sub-sub-paragraphs."

STEP2 (Second Level Procedural Step)

"Identifies a second level procedural step. A step is a single operation in a sequence of events, such as in maintenance instructions. Some Australian Defence DTDs (not 5629A) allow further breakdowns within the sequence which are identified by third level steps, ie [STEP3]. A second level step is treated as a sub-sub-paragraph [SUBPARA2] for presentation purposes, and third level steps as sub-sub-sub-paragraphs."

STEP1 appears to be as close as we get to a list:

- again, the hierarchy is non recursive
- paragraph text can't continue after the list (!)

Pattern	**Candidate Elements**	**Comments**
Numbered object container	PARA0	
Heading?	TITLE	
Paragraph*	PARA	
List*	STEP1	Doesn't meet requirement 2, since not only can a list not go inside a paragraph, but paragraph level text can't follow the list.
Container for nested numbered objects?	SUBPARA1.GRP	Optional, but to be avoided since its HEADING element is mandatory.
Numbered object*	SUBPARA1	

I don't consider 5629A any further.

LSSA XML Schema

The (UK) Legal Software Suppliers Association (LSSA)'s XML Schema for the Legal Profession (Ver 2.1) defines common data components (eg address, person, organisation, party). It does not purport to include a clause model.

http://www.legaltechnologyinsider.com/lssa/xmlstuff.htm

Europe Lex

LexData / LexML

Different philosophy, apparently not proposing to develop a clause model:

"Having observed the great difficulties in the USA in achieving general standard structures, it has been concluded that in Europe it would be irrealistic [sic] to strive for one structure per document type. The European legal landscape is .. too diverse to undertake, with any chance of success, a similar attempt as in the USA. Rather the view is, that it is better to allow and encourage a greater number of standard structures to be created. Structures which are created for very specific, actual and practical purposes. So more of a bottom up approach. On the one hand, each of these structures may cover a smaller amount of legal data, on the other hand each structure is likely to be of more direct practical use. From an organisational point of view an added advantage is that smaller communities, with similar interests, can agree on a standard structure more quickly. This approach is promoted by LEXML, the European Forum for XML in the legal domain. Other parts of the world will take their policy decisions concerning the standardisation process in the legal domain. Beyond doubt we will be faced with a wide variety of structures for legal data."

http://www.lexdata.org/general.htm

Perhaps DITA (referenced below) would be a way forward for them.

NLM Journal Publishing

http://dtd.nlm.nih.gov/publishing/

The American National Center for Biotechnology Information (NCBI), a center of the National Library of Medicine (NLM), created the Journal Publishing Document Type Definition (DTD) with the intent of providing a common format for the creation of journal content in XML.

```
"-//NLM//DTD Journal Publishing DTD v1.0 20030210//EN"
    Delivered as file "journalpublishing.dtd"

<!ENTITY % body-model   "(%para-level;)*, (%sec-level;)*"        >
<!ELEMENT  body         (%body-model;)                           >



<!ELEMENT  sec          (%sec-model;)                            >
where

            <!ENTITY % sec-model    "title, (%para-level;)*,
                                     (%sec-level;)*,
                                     (%sec-back-matter-elements;)*"          >

            <!ENTITY % sec-opt-title-model
                                    "title?, (%para-level;)*,
                                     (%sec-level;)*,
                                     (%sec-back-matter-elements;)*"          >

            <!ENTITY % sec-level    "%sec.class;"                            >


            <!ENTITY % sec.class    "sec"                                    >
```

So <sec> has a title, para-level stuff, and then nested <sec>.

The title is not optional unless the alternative model is employed

```
            <!ENTITY % sec-opt-title-model
                                    "label?, title?, (%para-level;)*,
                                     (%sec-level;)*,
                                     (%sec-back-matter-elements;)*"          >
```

And sec-opt-title-model is not used in the Journal Publishing DTD (unlike the Archiving and Interchange DTD) therefore we'd need to use some other subset of the Archive and Interchange DTD in order to get optional title.

```
<!ELEMENT  title        (#PCDATA %struct-title-elements;)*          >
where:

            <!ENTITY % struct-title-elements
                                    "| %break.class; | %simple-phrase;"      >

<!--                PARAGRAPH-LEVEL ELEMENTS                 -->
```

```
<!--                Elements that may be used at the same   structural level as a paragraph, for   example
inside a Section   Note: There a major overlap between this   parameter entity and that for the elements
that are at the same level as a paragraph.   Inline elements appear only inside a   paragraph, but block
elements such as quotes   and lists may appear either within a   paragraph or at the same level as a
paragraph. This serves a requirement in a   repository DTD, since some incoming material   will have
restricted such elements to only   inside a paragraph,  some incoming material   will have restricted them to
only outside a   paragraph and some may allow them in both   places. Thus the DTD must allow for them to
be in either or both.                -->
<!ENTITY % para-level   "%block-display.class; | %block-math; |
                   %list.class; | %math.class; |
                   %para.class;"                    >


<!--                PARAGRAPH CLASS                      -->
<!--                Information for the reader that is at the   same structural level as a Paragraph      -->
<!ENTITY % para.class   "p | %rest-of-para.class;"               >
<!ENTITY % rest-of-para.class
                  " disp-quote | speech | statement |
                   verse-group"                       >


<!ELEMENT  p          (#PCDATA | %inside-para;)*               >  per "-//NLM//DTD Archiving and
Interchange DTD Suite Paragraph-Like Elements v1.0 20021201//EN"
    Delivered as file "para.ent"



<!ENTITY % inside-para  "%block-display.class; | %block-math; |
                   %citation.class; | %emphasis.class; |
                   %inline-display.class; |
                   %inline-math; |  %inpara-address; |
                   %link.class; | %list.class; |
                   %math.class; | %rest-of-para.class; |
                   %phrase.class; | %subsup.class;"         >



"-//NLM//DTD Archiving and Interchange DTD Suite List Class Elements v1.0 20021201//EN"
    Delivered as file "list.ent"

<!ENTITY % list.class   "def-list | list"                  >

<!ENTITY % def-list-model
                  "label?, title?, term-head?, def-head?,
                   def-item*, def-list*"                >

                     <!ELEMENT  def-list     (%def-list-model;)             >
<!ELEMENT  def-item     (term, def*)                        >


<!ENTITY % list-model   "label?, title?, list-item+"            >
<!ELEMENT  list        (%list-model;)                        >
<!ELEMENT  list-item    (p | %list.class;)+                    >
```

so list-item can't have a title

Pattern	**Candidate Elements**	**Comments**
Numbered object container	sec	
Heading?	title	Need to employ sec-opt-title-model (from the exchange version) to make this optional.
Paragraph*	p	
List*	list	- list items can't have headings - lists can be nested
Container for nested numbered objects?	[none]	
Numbered object*	sec	

Consideration	**For**	**Against**
Strict subset of DTD	○ Strict subset of exchange model is possible (ie gives us optional title)	○ Except for headings on list items
Additional infrastructure provided by DTD	○ [TBD]	○
End user skill set		No familiarity in our target user group. This DTD is more likely to be adopted in academic/research than in corporate business community
Developer skill set	Well constructed DTD	○
Tool support – editing environment		○
Tool support – stylesheets	A set of stylesheets exists	
Extension mechanism	The Suite has been set up to be extended using a new DTD file and a new DTD-specific customization module to redefine the many Parameter Entities.	
Is DTD being actively developed under an open process?	○ Not really (?), but the site hosts a mailing list.	
Position in Lifecyle	○	v1.0, Feb 2003

Licence terms	○ "These DTDs and the Suite are in the public domain. An organization that wants to create its own DTD from the Suite may do so without permission from NLM."	

Key Contribution	Value
?	?
	-

DITA

Darwin Information Typing Architecture (DITA) is an XML architecture for extensible technical information, created by IBM in order to represent its technical documents.

http://www.ibm.com/developerworks/xml/library/x-dita4/

A document is made up of a number of *topics*.

A ***topic*** is "a chunk of information consisting of a heading and some text, optionally divided into sections". It provides the title, metadata and structure for the content. Happily, a topic may include one or more child topics in its include zone:

http://www-106.ibm.com/developerworks/xml/library/x-dita3/topic_structure.html

So DITA looks attractive in that it appears to model our "numbered object".

However, two potential problems need looking at:
- %title; is required – can it be defined as title?, so that it becomes optional
- %body; is required – can it be defined with optional contents, so that it becomes optional

Title and body are both required (see news://news.software.ibm.com/ibm.software.developerworks.xml.dita Michael Priestly reply to my post entitled "Specialisation, %title and %body")

Nevertheless, it remains interesting in the way that it is designed to be extended. This at least opens the possibility of using terminology suitable for contracts in contracts (eg 'clause') and terminology suitable for judgements in court documents (eg submission, reasoning, finding of fact, finding of law), and terminology suitable for other types of business documents in those business documents. (To be clear, we could just as easily decide to use one container name in all of those types of business documents)

There are two types of specialisation:
- topic specialisation: defining different types of topics

- domain specialisation: essentially, semantics/terminology which can be used in a topic (although emphasis is also handled this way)

Pattern	**Candidate Elements**	**Comments**
Numbered object container	topic	Or, define our own using topic specialization.
Heading?	title	Mandatory ☹.
Paragraph*	body p	Mandatory ☹.
List*	ol li	- list items can't have headings - lists can be nested
Container for nested numbered objects?	[none]	
Numbered object*	topic	

Consideration	**For**	**Against**
Strict subset of DTD	○	○ Title and body are not optional
Additional infrastructure provided by DTD	○ [TBD]	○
End user skill set	No familiarity in our target user group. Nevertheless, the specialization mechanism means that we could use meaningful element names. Also, the DTD is likely to be adopted in the technical writing division of some corporate users of our work.	
Developer skill set	Well constructed DTD	○
Tool support – editing environment		○
Tool support – stylesheets	A set of stylesheets exists	
Extension mechanism	Two notions: - topic specialization - domain specialisation	
Is DTD being actively developed under an open process?	○ Not really (?), but the site hosts a newsgroup.	
Position in Lifecyle	○	Initial release March 2001; Release 1.2, June 2003

Licence terms	○ the IBM Darwin Information Typing Architecture Specification Agreement grants a royalty free licence to modify the Specifications. Distribution is permitted under certain conditions.	

Key Contribution	Value
topic and domain specialization architecture	Worth exploring further.
	-

Colophon

This document was created using OpenOffice 1.1 beta on Linux and transferred to and from Word 2000 as necessary. Published to pdf using http://sourceforge.net/projects/pdfcreator