Items with a Simple Paragraph Model

(In Reply to "Structural markup – Basic clause model proposal and specification")

Jason Harrop, jharrop@speedlegal.com

17 November 2003

Summary

It is the thesis of this document that everything turns on whether we insist on an element which represents a grammatical paragraph, so that a list (block or inline) in a sentence can be encapsulated in that element:

```
<Item>
   <GrammaticalPara>
            Here is a list of colours:
              <Items>
                 <Item><Text>red,</Text></Item>
                 <Item><Text>green,</Text></Item>
                 <Item><Text>blue</Text></Item>
              </Items>
   </GrammaticalPara>
<Item>
```

In the models the TC has been considering, the GrammaticalPara element has been labeled Para or Block. I call these **Grammatical Paragraph models,** and argue that these models have certain inevitable consequences.   The consequnces flow from the fact that you can also do:

```
<Item>
   <GrammaticalPara>
            Here is a list of colours:
   </GrammaticalPara>
   <Items>
             <Item><Text>red,</Text></Item>
             <Item><Text>green,</Text></Item>
             <Item><Text>blue</Text></Item>
   </Items>
<Item>
```

On the other hand, if we decide that we do not need to model a grammatical paragraph, but rather, are content with capturing blocks of text (a typographical view), then a simpler model is available:

```
<Item>
            <Text>Here is a list of colours:</Text>
              <Items>
                 <Item><Text>red,</Text></Item>
                 <Item><Text>green,</Text></Item>
```

```
        <Item><Text>blue</Text></Item>
      </Items>
</Item>
```

I call models with this feature a **Simple Paragraph Model** (after DocBook's "simpara" element)[1].

I suggest that in view of the difficulties sometimes encountered in asking lawyers and other contract authors to choose between block-lists and sub-clauses, **<u>the TC should adopt a Simple Paragraph Model.</u>**

It completely avoids some of the technical difficulties which have dogged us so far, and is considerably easier for lawyers and other contract authors.

By way of introduction, this document explains in more detail the outstanding issues with lists and subclauses which we have been grappling with.

It is these issues, and the insight that they are inherent in Grammatical Paragraph Models[2], which leads us to a Simple Paragraph model.

The model contained in Peter's "Structural markup – Basic clause model proposal and specification" document (draft 1.0 of 11 November 2003) is a Grammatical Paragraph model. Hereinafter I refer to it (and that document as a whole) as GP1.

In my view, a Simple Paragraph Model offers us very considerable advantages over any Grammatical Paragraph Model.

It is for this reason that the body of this document does not discuss the specifics of GP1.

If, following a comparison of Simple and Grammatical Paragraph Models, and a discussion of their relative merits, the TC expresses a preference for a Grammatical Paragraph Model, GP1 would need to be considered further. For this reason, a critique of GP1 and recommendations for improving it are contained in Appendix 2.

The block-list/sub-clause continuum

In the clause model requirements document and GP1 topic 9.4, a series of examples of structures are presented, with the objective of demonstrating that there is no clear distinction between a block list and a sub-clause.

GP1 page 20 observes (correctly) that there is

> a continuum of structures that range from clauses to lists

<u>The problem is that any Grammatical Paragraph Model (GP1 is no exception) forces an author to choose whether a particular structure is a list or a sub-clause.</u>

That's the whole point of a Grammatical Paragraph Model – a list in a sentence, is encapsulated in the grammatical paragraph container which represents the paragraph the sentence is part of. For example,

> The following meals are included in the daily tariff:
>
> (a) breakfast
> (b) lunch
> (c) dinner,
>
> except on weekdays, where lunch is excluded.

would be captured within

```
  <GrammaticalPara>
        The following meals are included in the daily tariff:

        (a)     breakfast
        (b)     lunch
```

1 In his article "Modelling Paragraphs in XML", Alex Brown calls something similar a "Sibling Blocks Model".

2 Assuming they also allow list-like structures *outside* the grammatical paragraph element.

(c) dinner,

except on weekdays, where lunch is excluded.

</GrammaticalPara>

Where the list is clearly part of a sentence as in that example, things are easy. What if the list isn't embedded in a sentence? It could still be part of a paragraph. The Grammatical Paragraph Model forces the author to decide between **two ways** of marking up the structure.

For example, given the text:

Next we consider motive and opportunity.

(a) Motive is blagh blagh.

(b) Opportunity is blagh blagh.

Is that to be marked up as:

```
<GrammaticalPara>
        Next we consider motive and opportunity.

        <List>
        (a)     Motive is blagh blagh.

        (b)     Opportunity is blagh blagh.
        </List>

</GrammaticalPara>
```

or

```
<GrammaticalPara>
        Next we consider motive and opportunity.

</GrammaticalPara>

<List>
        (a)     Motive is blagh blagh.

        (b)     Opportunity is blagh blagh.
</List>
```

How do they decide? An author might consider whether the following indicia of listness are present:

- is the list embedded in a sentence
  - is there a preceding colon?
  - presence of and/or
  - continuation of sentence after list
- whether each list item makes sense as a standalone sentence
- where the document numbers lists and sub-clauses differently, whether list numbering is used
- presence of a heading
- potentially, whether you want the headings to appear in a table of contents

GP1 topic 9.3 introduces a distinction between document outline and narrative content. Read carefully, that distinction is not employed to help distinguish lists from sub-clauses. Indeed, document outline and narrative content are said to be "overlapping hierarchies" – the reason being that in the middle of the block-list/sub-clause continuum (where you have sub-clauses without headings, and list items with headings), it is not obvious which hierarchy the structure belongs to (GP1 p19, penultimate para).

As it turns out, GP1 concludes (emphasis is added):

> "In practice, it is expected that the author's choice will depend on the way in which **document numbering** is applied by the author's applications and the degree of control given to the author to specify **numbering patterns** at particular parts of the document." (GP1 pp19-20)
>
> "In marginal cases, the examples show that authors may need to deliberately markup a list as either within the document hierarchy or within the narrative content to achieve a particular **automatic numbering outcome where the application treats numbering differently** in the document outline to numbering in the narrative content."

It should be noted that the distinction which GP1 introduces between document outline and narrative content is an abstract and perhaps artificial concept. As GP1 9.3.2 says (emphasis added):

> "Items [in the document outline] usually have titles and may be numbered. The document outline is usually revealed by the contents listing for the document, although this is not always so. .."

The quote about "automatic numbering outcome(s)" above suggests that numbering depends on whether something is part of the "document outline" or not. I'm not sure that that is how contract authors think – particularly where the contract doesn't have a table of contents.

Technical aside: In Microsoft Word, whether something appears in the table of contents or not doesn't depend on how it is numbered, but rather, on whether it has an outline level.

Put simply, where the application numbers lists and sub-clauses differently, you'll make something in the middle of the continuum a list or a sub-clause, depending on how you want it numbered.

GP1 p21 says (emphasis added):

> The approach taken by the clause model is that authors should not have to care as to whether something is correctly characterised as a clause or a list. This is achievable when there are only two ways in which the hierarchy can be represented. An item is either part of the document outline (Items occur within a parent Item element) or it is part of the narrative content (Items occur within a parent Block element).

Unfortunately, the problem is that **if there are "two ways in which the hierarchy can be represented", then they "do have to care"**.

It is the Grammatical Paragraph Model which introduces these two ways, and with them, all the confusion in trying to explain to authors, which way to do it, and how to change from one to the other.

Further, it is the Grammatical Paragraph Model and its "two ways" which introduces the problem in GP1 10.7, for which one solution is to redefine Item according to context, using RelaxNG or W3C XML Schema. With a Simple Paragraph Model, the problem simply never arises.

In contrast, a Simple Paragraph Model provides only **one way** to represent structures on the block-list/sub-clause continuum:

```
<SimplePara>Next we consider motive and opportunity.</SimplePara>

<List>
    (a)      Motive is blagh blagh.

    (b)      Opportunity is blagh blagh.
</List>
```

That is, block-rendered Items never occur in a paragraph.

GP1 suggests that the List container could be dispensed with (GP1 11.6). However, in doing so:

- the real but somewhat problematic concept of a list is replaced with the unfamiliar and equally problematic distinction between document outline and narrative content.
- attributes which could naturally be placed on the list container have to be relocated to somewhere unexpected
    - in particular, it becomes difficult to support multiple lists in a clause (not just from a numbering perspective – what if the user wants to capture the fact that the first list is an "or"-list, and the second an "and"-list?)

- the convenience offered by the list container for XSLT and other processing is lacking

You don't have to have a list container as part and parcel of a Simple Paragraph Model. However, a Simple Paragraph Model offers you the opportunity to use a single container for any structure on the block-list/sub-clause continuum, and so the temptation to dispense with it is vanquished. Advantageously, that container can also be used for inline lists (see below). The best name for that container would seem to be "Items".

Inline-list/block-list/sub-clause continuum

What is an "inline list"? By way of example, the following paragraph contains two:

> Subject to the further provisions of this Section 1.4, Tenant shall have the option to terminate this Lease effective as of **(i)** September 30, 2004, **(ii)** September 30, 2006, **(iii)** June 30, 2008 or **(iv)** June 30, 2013 (each of the dates set forth in the preceding clauses (i) - (iv) is called a "Termination Date"), by giving an exercise notice to Landlord on or before the date that is **(A)** one year prior to the applicable Termination Date in the case of a termination as of September 30, 2004 or September 30, 2006 or **(B)** 18 months prior to the applicable Termination Date in the case of a termination as of June 30, 2008 or June 30, 2013.

Other examples nest them.

The key difference between block and inline lists is that in a block list, each item is a separate block (ie separated by carriage returns). Headings are less common on inline list items, but they could occur.

Given that the main difference is presentation, authors may wish to flip between representing something as an inline or a block list. Conceivably, they might take an inline list, convert it to a block list, add headings, and want it to be numbered as a sub-clause.

For this reason I've added inline-list to the continuum[3].

There is a natural way to mark up the above example with a Simple Paragraph Model:

> <SimplePara>Subject to the further provisions of this Section 1.4, Tenant shall have the option to terminate this Lease effective as of <Items>**(i)** September 30, 2004, **(ii)** September 30, 2006, **(iii)** June 30, 2008 or **(iv)** June 30, 2013</Items> (each of the dates set forth in the preceding clauses (i) - (iv) is called a "Termination Date"), by giving an exercise notice to Landlord on or before the date that is <Items>**(A)** one year prior to the applicable Termination Date in the case of a termination as of September 30, 2004 or September 30, 2006 or **(B)** 18 months prior to the applicable Termination Date in the case of a termination as of June 30, 2008 or June 30, 2013.</Items></SimplePara>

Note that here I've used "Items" as the name for the list container – whatever it is called, it should be the same name as for the block structures.

A Simple Paragraph model

Introduction

The problems posed by the block-list/sub-clauses continuum where you are trying to use a Grammatical Paragraph Model simply do not arise if you use a Simple Paragraph Model instead.

The Proposal

<!ELEMENT Items (Item)*>

<!ATTLIST Items UseListNumbers (true|false) false>

<!ENTITY % TextSiblings "Text | Items | Table" >

<!ELEMENT Table (EMPTY)>

3Although obviously there is no grey area or continuum between inline- and block-list.

```
<!ELEMENT Item  (Num?, Heading?, (%TextSiblings;)* )>
<!ATTLIST Item ID ID #REQUIRED>


<!ELEMENT Num (#PCDATA)>
<!ELEMENT Heading (Text)*>


<!ENTITY % InlineContextContent "Items" >
<!ELEMENT Text (#PCDATA | %InlineContextContent; )*>
```

Example

```
<Item ID="example">
        <Num>1</Num><Text>Next we consider motive and opportunity.</Text>
        <Items UseListNumbers="true">
                <Item ID="li1"><Num>(a)</Num><Text>Motive is blagh blagh.</Text></Item>
                <Item ID="li2"><Num>(b)</Num><Text>Opportunity is blagh blagh.</Text></Item>
        </Items>
</Item>
```

Text and the Simple Paragraph

In this model the "Text" element plays two roles, one of which is the simple paragraph element.

Its other role is in an inline list, where Item is re-used.

The Text element is always block for block level Items, and inline for inline level Items.

Item

Item is something which gets numbered, irrespective of whether it represents a section or clause, or some sub-structure on the inline-list/block-list/sub-clauses continuum.

It can have a heading, and may then contain any number of simple paragraphs and substructures in any order.

Note that the content model (%TextSiblings;)* is looser than the loose model refered to in GP1. The reason for this is to accomodate content which would otherwise have to be regarded as legacy content. See Appendix 1 for more details.

The Items container

The Items container occurs in two places[4]:

- in an Item, where it represents a structure on the block-list/sub-clauses continuum
- in Text, where it represents an inline list

To convert an inline list to a block list, you would move the Items container out of the Text element, to next to it.

This container is convenient:

- it is a natural place for attributes pertaining to the list
- it is convenient for XSLT and other processing

<u>The Continuum Attribute</u>

Since the Items go in the same place irrespective of where the structure falls on the block-list/sub-clause continuum, you can't distinguish between a list and a sub-clause by looking at the element and its position alone (that's the idea, after all).

4 An alternative model would be to only allow the Items container in one place (ie next to Text), whether it was intended to represent and inline or block list (or sub-clauses). However, this gets messy (eg what if the following Text is supposed to be a new paragraph?).

However, you may need to know for numbering purposes (ie in documents where list items and sub-clauses are numbered differently).

So, an attribute is required. It can naturally be placed on the Items container.

There are various alternatives:

- @list="true|false"
- @UseListNumbers="true|false"
- @hierarchy="outline|narrative"
- @numberformat="list|subclause"
- @numberformat="[some complex number format specification]"

For the moment, I've used @UseListNumbers, but the alternatives may merit further discussion.

Ideally, this attribute would only be available in a block context, since in an inline context, the Items would always be numbered as a list.

Local content models for Item?

The question arises as to whether for an Item in an inline list (ie in an inline context), Tables and other blocks available in the block context should be prohibited.

Note that there is no need to do this in order to address the problem documented in GP1 10.7, since that problem is particular to Grammatical Paragraph Models.

Here the purpose would be to prevent what is ordinarily thought of as block level content from appearing in an inline context.

If enforced, this would mean that a document would become invalid if a block-list containing a table were made into an inline list by moving it to the inline context.

It is useful to distinguish between a continuously validating XML editor, and one which does not do so.

- A continuously validating XML editor simply would not allow such a block-list to be moved to the inline context.
- One which did not continuously validate would only flag that error at some later point when the document was validated. The user would then have to correct the error (by deleting the table, or undoing their move).

I tend to think that we should not redefine Item in an inline context to prevent what would ordinarily be thought of as block level content (eg a table) from appearing. If it does happen, then the rendering application should handle it.

Nevertheless, it may be worth exploring whether a local content model would be possible (assuming that in each case the parent element is Items).

"*Titles on Items in the document outline*"

GP1 10.6 proposes 4 ways to address the problem it identifies (I discuss GP1 10.6 further in Appendix 2)

It suffices here to note that, irrespective of whether you use a Grammatical Paragraph Model or a Simple Paragraph Model:

- the problem GP1 10.6 identifies occurs;
- the 4 options presented in GP1 10.6 (and the further options presented in Part 3 below) work equally well (or badly) to address the problem.

GP1 10.5.1 identifies what it regards as a problem with "Block before Items". As I explain in Appendix 2, I don't see the difficulty. However, if there is a problem, it is likely that the Simple Paragraph Model exacerbates it, by allowing what GP1 might think of as "Block between Items":-

```
<Items>

  <Item ID="main">

    <Heading><Text>My Big Item Heading </Text></Heading>


    <Text>Some Text</Text>
    <Items>
      <Item ID="li_11"><Heading><Text>Sub Heading 1.1..</Text></Heading></Item>
      <Item ID="li_12"><Heading><Text>Sub Heading 1.2..</Text></Heading></Item>
    </Items>

    <Text>Some More Text</Text>

    <Items>
      <Item ID="li_13"><Heading><Text>Sub Heading 1.3..</Text></Heading></Item>
      <Item ID="li_14"><Heading><Text>Sub Heading 1.4..</Text></Heading></Item>
    </Items>
  </Item>
</Items>
```

The fact that this is possible is hardly a fatal flaw. If an application were concerned about it:

- for table of contents purposes, it could raise a warning (note that this structure would be okay if the Items had @UseListNumbers="true", and lists weren't put in the table of contents)
- for chunking purposes, it could either handle it in the natural way, or raise a warning.

Heading element

The content model for the Heading element is one or more Text elements. The intent is to provide a mechanism for a contract author to decide where line breaks fall in long headings.

Because Text can contain an inline list, which in turn re-uses Item, a Heading could currently contain a table. For this reason, it may be better to introduce another element or consider some other approach to handling long headings.

Advantages of this Simple Paragraph Model over Grammatical Paragraph Models

- Major - All structures on the block-list/sub-clause continuum are marked up the same way, the only difference being an attribute value which is significant if numbering is different at each pole of the continuum, so authors may not have to think about it
    - Major – Easier conversion
- Major - All structures on the block-list/sub-clause continuum are placed into the element structure at the same location – the move operation is avoided
- Re-use works along the block-list/sub-clause continuum
- Structures which may flip in the inline-list/block-list dichotomy are marked up the same way, the only difference being whether they appear in a block or inline context.
- The problem documented in GP1 10.7 is avoided entirely.
- Same number of discrete elements as GP1, but fewer tags are used in typical markup[5]

Disadvantages

- Minor - The fact that a block-list is part of a grammatical paragraph is not reflected in the element markup

5 a Grammatical Paragraph Model marks up a grammatical paragraph; GP1 lacks the Items container; grammatical paragraphs typically occur more often than lists.

    - If it is useful to tell where a grammatical paragraph starts (for example, because you want to reflect that in paragraph spacing in the output, or because you start a new paragraph following a list with a tab), an attribute could be added to Text to indicate this (default would be new paragraph).
- Minor - If numbering is different at each pole of the continuum, an attribute is required
- Minor - To move a grammatical paragraph, you have to select multiple objects (similar to todays wordprocessors), rather than just one
- Minor - Harder to use a grammar checker?
- Minor – To move between an inline-list and a block-list, you have to physically move the element from an inline context to a block context.

Other DTDs?

The idea of a Simple Paragraph Model is not new.[6]

XHTML 1.0 is a Simple Paragraph Model, whereas XHTML2 5th WD of May 2003 includes a Grammatical Paragraph Model.

DocBook has both.

OpenOffice and WordML are both Simple Paragraph Models.

<u>XHTML 1.0</u>

XHTML 1.0 is a Simple Paragraph Model.

However, XHTML 1.0 doesn't:

- enforce order (heading first)
- re-use Item in list context
- represent inline lists in <p> - you have to use CSS to render a list inline

<u>DocBook</u>

If we wanted to do this using docbook element names, it would look something like:

```
<!ELEMENT Sections  (section)*>

<!ELEMENT section  (title, (simpara | Sections)* )>

<!ELEMENT simpara  (#PCDATA | Sections)*>
```

but there are three problems:

- there is no "Sections" element
- I think you can't do (simpara | Sections)*, but rather, only (simpara*, Sections*)
- there is no inline list for simpara?

<u>XHTML2 5th WD of May 2003</u>

XHTML2 5th WD of May 2003 includes a Grammatical Paragraph Model, but we might be able to define a subset which amounts to a Simple Paragraph Model.

If we wanted to do this using XHTML2 5th WD of May 2003 element names, it would look something like:

```
<!ELEMENT Sections  (section)*>

<!ELEMENT section  (h, (p | Sections)* )>

<!ELEMENT p (#PCDATA | Sections)*> <!-- Sections is here for inline lists -->
```

6 What may be new is the power of using it within an Item structure, and in particular, one facilitating containerised Item re-use across the entire inline-list/block-list/sub-clause continuum.

but there are problems:

- there is no "Sections" element
- p can contain lists not Sections, and we don't want to distinguish between them

Possible Next Steps

1. First, decide between a Simple Paragraph Model and a Grammatical Paragraph Model.
    - If we choose a Simple Paragraph Model, modify GP1 accordingly.
    - If we choose a Grammatical Paragraph Model, consider the tweaks proposed in Appendix 2, decide which to adopt, and then document each proposed tweak (whether accepted or not) in a new draft of GP1.
2. Articulate appropriate architectural design principles in a new draft of GP1. These should fall out of the decisions the TC will have already made.
3. Onwards with Requirement 11.

Attachment 1 - Markup example

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE Items [
  <!ELEMENT Items  (Item)*>
  <!ATTLIST Items UseListNumbers (true|false) "false">
  <!ELEMENT Item  ( Num?, Heading?, (Text | Items | Table)* ) >
  <!ATTLIST Item ID ID #REQUIRED>
  <!ELEMENT Num (#PCDATA)>
  <!ELEMENT Heading (Text)*>
  <!ELEMENT Text (#PCDATA | Items )*>
  <!ELEMENT Table (EMPTY)>
]>
<Items>
  <Item ID="cl_1">
    <Num>1.</Num><Heading><Text>Provisions about the specification of colours in contracts</Text></Heading>
    <Items>
      <Item ID="sc_11">
        <Num>1.1</Num><Heading><Text>Spectrum colours</Text></Heading>
        <Text>Here is a contrived...</Text>
        <Items UseListNumbers="true">
          <Item ID="li_11"><Num>(a)</Num><Text>red,</Text></Item>
          <Item ID="li_12"><Num>(b)</Num><Text>orange,</Text></Item>
          <Item ID="li_13"><Num>(c)</Num><Text>yellow,</Text></Item>
          <Item ID="li_14"><Num>(d)</Num><Text>green,</Text></Item>
          <Item ID="li_15">
            <Num>(e)</Num><Text>blue, including:</Text>
            <Items UseListNumbers="true">
              <Item ID="li_151"><Num>(i)</Num><Text>pale blue,</Text></Item>
              <Item ID="li_152"><Num>(ii)</Num><Text>dark blue,</Text></Item>
            </Items>
            <Text>but excluding violet.</Text>
          </Item>
          <Item ID="li_16"><Num>(f)</Num><Text>indigo, and</Text></Item>
          <Item ID="li_17"><Num>(g)</Num><Text>violet,</Text></Item>
        </Items>
        <Text>from which all colours can be derived.</Text>
      </Item>
      <Item ID="sc_12">
        <Num>1.2</Num><Heading><Text>CMYK colours</Text></Heading>
        <Text>CMYK colours .. are normally specified ...</Text>
      </Item>
      <Item ID="sc_13">
        <Num>1.3</Num><Heading><Text>RGB colours</Text></Heading>
        <Items>
          <Item ID="li_131"><Num>1.3.1</Num><Text>RGB colour .. specifications are used ..</Text></Item>
          <Item ID="li_132"><Num>1.3.2</Num><Text>Using only these three colours ..</Text></Item>
          <Item ID="li_133">
            <Num>1.3.3</Num><Text>The number of colours you can specify depends on the colour depth
available.  For example:</Text>
            <Items  UseListNumbers="true">
              <Item ID="li_1331"><Num>(a)</Num><Text>8 bit colour can render 256 colours</Text></Item>
              <Item ID="li_1332"><Num>(b)</Num><Text>16 bit colour can render 65,536
colours</Text></Item>
            </Items>
          </Item>
        </Items>

      </Item>
      <Item ID="sc_14">
        <Num>1.4</Num><Heading><Text>Using black and white</Text></Heading>
        <Items>
          <Item ID="sc_141">
            <Num>1.4.1</Num><Heading><Text>Greyscale</Text></Heading>
            <Text>The number of greys depends on the available colour depth, as for other colours.</Text>
          </Item>
          <Item ID="sc_142">
            <Num>1.4.2</Num><Heading><Text>Black and white</Text></Heading>
            <Text>This is really called monochrome.  You can specify either:</Text>
            <Items UseListNumbers="true">
              <Item ID="li_1421"><Num>*</Num><Text>black, or</Text></Item>
              <Item ID="li_1422"><Num>*</Num><Text>white</Text></Item>
            </Items>
          </Item>
        </Items>
      </Item>

    </Items>
  </Item>
  <Item ID="cl_2">
    <Num>2.</Num><Heading><Text>Colour profiles</Text></Heading>
    <Text>One thing to remember...</Text>
  </Item>
```

</Items>

Appendix 1 - Legacy documents

- The "tight" model

  ```
  <!ELEMENT Item (Num?, Title?, (Para* | Item*))>
  ```

  can model the most common clause structures found in contracts. Note: I've used Para here, not Block, since I find Block confusing.

- The "loose" model:

  ```
  <!ELEMENT Item (Num?, Title?, Para*, Item*)>
  ```

  is required in order to model certain other structures (see GP1 10.5).

- Neither the tight nor the loose model can model the following structure:

  1. Main Clause

     1.1 First sub clause

     Some text...

     1.2 Second sub clause

     Some text...

     Here is the bit neither the tight nor loose model accommodate. It can be done if 1.1 and 1.2 are modelled as a list, but not if it is "part of the document outline (Items occur within a parent Item element)".

  This is a limitation with the models both Peter and I have proposed to date. It is not an inherent limitation of models in the Grammatical Paragraph Model family. The following ("loosest model") overcomes the limitation:

  ```
  <!ELEMENT Item (Num?, Title?, (Para | Item)* )>
  ```

  The clause model report should either address or acknowledge this limitation.

- The TEN HANOVER LLC lease to GOLDMAN SACHS GROUP, L.P. of August 22, 1997 is full of clauses which contain both block and inline lists (as well as nested inline lists!).

It is submitted that all other things being equal, we should aim to be able to model the widest set of existing clause structures.

Appendix 2 - An Analysis of GP1

This Appendix is included in case the TC favours a Grammatical Paragraph Model over a Simple Paragraph Model such as the one advocated in this document.

If a Grammatical Paragraph Model is the TC's preference for some reason, I believe GP1 can be improved in several respects.

Summary of Points of Concern

Most of the points of concern I have identified relate to the details of how some part of the pattern is modelled. These are contained in the table below.

The points of concern are ordered by their position in the content model hierarchy, from top to bottom.

Point of Concern	***PM Position***	***Significance of Implications***
Topic | Item - Dual hierarchies (10.6.5)	10.6.7 says it "should be left in the draft specification"	High
Element name: "Topic" or "Item" (or "Clause" or "Section") (10.3)	Item, or possibly "Topic" (10.3.2)	Low
Tight or loose model (10.5)	Unclear – preference for tight, but loose should be designated as the standard for exchange (10.5.3)	Medium
Document outline	@StopContentsBelow, but dual hierarchies not ruled out	Medium/Low
Include an element corresponding directly to a grammatical paragraph, encapsulating any list the paragraph contained	Implicitly accepted	Critical
Element name for "grammatical paragraph": "Para" or "Block"	"Block"	High
Para/Block contains <text>, not mixed content	<text>	Medium
Character of <text> element (12.4)	Preference for @text.flow, which determines whether the element is block or inline	Medium
Element name: <Text> or <TextBlock>	<Text>	Low
Content model for Title	#PCDATA	Low
Container for lists and subclauses	No	High
Content model for block list	Re-use item	Medium/High
Content model for inline list	Not addressed yet	High
Mechanism for handling block quotations	@text.flow	Medium

A further concern (not addressed in any detail here) is the limited "design" guidance contained in GP1 topic 7. I believe we ought to articulate architectural design principles as they occur to us, so that our model is internally consistent and does not look like it has been "designed by committee".

The points of concern are discussed below grouped as appropriate, in order of their significance.

Represent Grammatical Paragraph

Point of Concern	***PM Position***	***Significance of Implications***
Include an element corresponding directly to a grammatical paragraph, encapsulating any list the paragraph contained	Block element	Critical

Element name for "grammatical paragraph": "Para" or "Block"	"Block"	High

GP1 is a Grammatical Paragraph Model. GP1 uses an element called "Block" to represent the grammatical paragraph.

It is not the purpose of this section to discuss the advantages and disadvantages of the Grammatical Paragraph Model. That is done elsewhere.

Assuming a Grammatical Paragraph Model, the question is what the element representing it should be called.

I strongly suggest it be called something other than "Block".

To call it a "block" is to mis-use a term which has quite a clear meaning to (X)HTML authors, CSS writers, and users of XSL FO.

The one thing the element clearly isn't is a Block. In GP1 the grammatical paragraph element can't contain mixed content, so it never represents a block of text to be rendered.

Its a container for Text and Item elements forming part of a grammatical paragraph.

Given that it represents a grammatical paragraph, it makes good sense to call it "Para" or something similar, but if that isn't acceptable to the TC for some reason, there must surely be plenty of alternatives to "Block".

Normally I wouldn't give an element name a "high" significance, but this one is, because of its capacity to mislead and confuse.

The Document Outline, Tight or Loose Model, and the prospect of Dual (or Dueling?) Hierarchies

Point of Concern	***PM Position***	***Significance of Implications***
Topic | Item - Dual hierarchies (10.6.5)	10.6.7 says it "should be left in the draft specification"	High
Tight or loose model (10.5)	Unclear – preference for tight, but loose should be designated as the standard for exchange (10.5.3)	Medium
Document outline	@StopContentsBelow, but dual hierarchies not ruled out	Medium/Low

These points of concern relate to GP1 10.5-10.6 (pp25-31).

GP1 p25

I do not understand why having a "Block before Items" presents the problems identified in GP1 10.5.1(a) and (c). For (c), you simply display a page representing that level and containing hyperlinks to child levels.

Note: "Block between Items", as would be allowed by the loosest model and also by the Simple Paragraph Model, could be handled the same way.

Philosophically, I believe we should take this data as we find it, and handle it, rather than promoting a content model which can't deal with it and instead relegates it to the category of legacy data.

GP1 p26 – "Specific Recommendations"

I believe we need to choose between the loose and tight models (and indeed, the even looser model identified in Appendix 1 above).

As soon as you say the "loose model" is the standard for exchange between enterprises, its that which is effectively the standard, since all applications handling extra-enterprise data would need to be able to process it.

Of course, people can define their own subsets for their own private use, but there is no need to give such a subset any status in the standard.

To have a bet both ways as GP1 does will lead to confusion. For example, the "tight" model is presented in GP1 10.1, even though recommendation 5 in 10.5.3 gives the loose model at least equal status, or does it?

Maybe its a matter of saying the "tight" model represents best practice, and the "loose" and/or "looser" model is designed to accomodate content the "tight" model doesn't support. To take this tack, we need to justify why it is best practice, and test whether the notion will be supported in our user community.

<u>GP1 p27</u>

As I understand it, one of the problems of concern to the author of GP1 (4 are identified), is that if you have a series of sub-clauses, some of which have titles and others of which don't, then how do you create a table of contents for them?

Handling this issue is not one of the 10 basic requirements contained in the requirements document, and I believe this issue should be defered to the point at which we have a model for the entire contract and addressed then. There are three reasons for suggesting it be deferred:

- each of the options proposed in GP1 can be included in the clause model at a later time, without requiring fundamental design. So, in the absence of clear requirements accepted by the TC, attempting to handle it now complicates the core clause model work for no good reason.
- we need to see what the entire contract (including parties, recitals, signature blocks, schedules etc) looks like before we have a complete view of the chunking and table of contents problems.
- there are various ways to attack this problem which aren't presented in GP1, for example:
    - Assume the author will notice the problem and correct it (possibly assisted by their application)
    - Explore the possibilities of a table of contents element, which specifies what is to appear in the table of contents and possibly what is to have a heading
    - Leave it out of the standard, to be handled in-house however one likes

Having said that, I would be very concerned if dueling topic and item hierarchies were introduced to address it, for two reasons:
- re-use would be inhibited
- styling and numbering would be harder for people to specify/set-up

These two disadvantages to my mind far outweigh the 5 advantages identified in GP1 p30.

Insofar as the other options presented are concerned, @StopContentsBelow, as suggested in GP1 10.6.4 might be okay, but at this stage it is not clear it is the best solution. Does it mean that conforming applications would have to insist that a Title element be present, until that attribute is explicitly set by the author?

The Block Element (bka Para)

Point of Concern	***PM Position***	***Significance of Implications***
Para/Block contains <text>, not mixed content	<text>	Medium
Character of <text> element (12.4)	Preference for @text.flow, which determines whether the element is block or inline	Medium

This is the grammatical paragraph element – see naming discussion above.

There are really three choices for the content model for the grammatical paragraph:

- mixed content (ie PCDATA and lists etc)
- a grammatical paragraph is made up of block level items
- a grammatical paragraph is made up of elements which can be either block or inline (indeed, some elements could sometimes be block and sometimes inline)

I would be happy with either of the first two – probably most happy with the second – but GP1 12.4.3 raises the prospect of the third, since it suggests that the text element would become an inline one if @text.flow="runon".

The main problem with it is that it is an unncessary expediency which complicates XSLT processing.

For example to create XSL formatting objects, if the text element were always block level, you could have a simple natural form of template:

```
<xsl:template match="Text">
   <fo:block>
      <xsl:apply-templates/>
   </fo:block>
</xsl:template>
```

If it is possible that the following Text element is to be displayed inline, then you have to use some other strategy with the objective of capturing its content in the one fo:block . Without having given it too much thought, here are three possible (but admittedly ugly) approaches one might explore:

1. fo:blocks are inserted by the template which matches econtract:Block, as it for-each's its children
2. The Text template matches only if @text.flow is not present, inserts a fo:block as above, but before closing it, checks if the next sibling is a Text with @text.flow set, and the next one, and the one after that ...
3. Text templates insert opening and closing fo:block tags independently as required, using

```
<xsl:text disable-output-escaping="yes">&lt;fo:block&gt;</xsl:text>
```

However, its not at all clear to me why we'd want to make this problem for ourselves. I don't find the justifications at GP1 p45 convincing.

Re-use across Block-List/Sub-Clauses Continuum, and the List Container

Point of Concern	***PM Position***	***Significance of Implications***
Container for lists and subclauses	No	High
Content model for block list	Re-use item	Medium/High
Content model for inline list	Not addressed yet	High

There are 3 problems here.

GP1 seeks to save authors from having to characterise a structure as either a list or a sub-clause.

As I have shown, this is not possible with a Grammatical Paragraph Model (even if you don't have a list container – howsoever named) – the best that can be done is to introduce some more abstract concepts and recast the problem as one of "changing lists from the document outline to the narrative content".

That's the first problem here.

Problem 2 arises out of omiting a list container (whatever it is called):

- other attributes (and there may be many) which could naturally be placed on the list container have to be relocated to somewhere unexpected
- the convenience offered by the list container for XSLT and other processing is lacking

In particular, without a list container it becomes difficult to adequately support multiple lists in a clause (not just from a numbering perspective – what if the user wants to capture the fact that the first list is an "or"-list, and the second an "and"-list?).

Grammatical Paragraph Models by their nature set quite a high bar in terms of authoring expertise. If you are going to impose a Grammatical Paragraph Model on a contract author, it is entirely consistent to expect him/her to handle the container. If you want to make life easier, use a Simple Paragraph Model (where you can have your cake and eat it as well, since you can have an "items" container without any attendant problems).

Regarding re-use of item in block-lists (see GP1 p32), I think this may be a good idea notwithstanding the problem identified (Problem 3) and the fact that it can't be solved with DTDs. Note that the problem is specific to Grammatical Paragraph Models – it simply does not occur with a Simple Paragraph Model.

Regarding inline lists, I think we ought to aim for re-use between block and inline lists (ie reuse item in an inline list). Admittedly, <text> would become inline in this context.

That's okay, since there is a good reason, and it is easily handled:

```
<xsl:template match="Text[ancestor::econtract:Text]">
   <fo:inline>
      <xsl:apply-templates/>
   </fo:inline>
</xsl:template>


<xsl:template match="Text[not(ancestor::econtract:Text)]">
   <fo:block>
      <xsl:apply-templates/>
   </fo:block>
</xsl:template>
```

The Block Quotation Problem

Point of Concern	***PM Position***	***Significance of Implications***
Character of <text> element (12.4)	Preference for @text.flow, which determines whether the element is block or inline	Medium
Mechanism for handling block quotations	@text.flow	Medium

Problems which are raised by specific artifacts such as block quotations and equations can and should be addressed by point solutions which encapsulate the artifact (ie a QuotationBlock element).

It is acceptable for an artifact that occurs rarely to be tailored specifically to the issues associated with it.

In contrast, it is not a good idea to unnecessarily complicate core elements (ie Text) on account of esoteric issues with rare content.

The name and content model for a block quotation can be addressed as part of requirement 11. However, whether there is support for the broad principles outlined in this section should be decided sooner.

Content Model for Title

Point of Concern	***PM Position***	***Significance of Implications***
Content model for Title	#PCDATA	Low

Sometimes a long title is rendered over multiple lines, and it may be that the last line only of the title is, for example, underlined.

In order to give the author control of where the line breaks occur, one could use <Text> as its content model. I suggest the TC consider this or something similar.

Element Names

Point of Concern	***PM Position***	***Significance of Implications***
Element name: "Topic" or "Item" (or "Clause" or "Section") (10.3)	Item, or possibly "Topic" (10.3.2)	Low

Element name: <Text> or <TextBlock>	<Text>	Low

Given the desire to reuse Item in the context of both block and inline lists, "Item" and "Text" work for me.

If Item is not re-used in a block list, I'd prefer "Clause" to "Item".

If Item is not re-used in an inline list, I may prefer "TextBlock" to "Text".

Note: I presume that GP1 11.3.4 is supposed to appear in 10.3.1.

Appendix 3 – Summary of Models

Point of Concern	***GP1***	***JH Suggestion assuming Grammatical Paragraph Model***	***JH Suggestion assuming Simple Paragraph Model***
Topic | Item - Dual hierarchies (10.6.5)	10.6.7 says it "should be left in the draft specification"	**No**	No
Element name: "Topic" or "Item" (or "Clause" or "Section") (10.3)	Item, or possibly "Topic" (10.3.2)	Preference for "Clause" or "Section" is outweighed by desire to re-use in list context, and unsuitability of those words to that context.	Item
Tight or loose model (10.5)	Unclear – preference for tight, but loose should be designated as the standard for exchange (10.5.3)	Loose model, with list container, which models larger range of content, or even "looser" model.	[Appears looser, but models same range of content as loose model with list container]
Document outline	@StopContentsBelow, but dual hierarchies not ruled out	Defer until after "Requirement 11" phase and explore options	Defer until after "Requirement 11" phase and explore options
Include an element corresponding directly to a grammatical paragraph, encapsulating any list the paragraph contained	"Block"	[Yes, per assumptions]	No
"grammatical paragraph" element name: "Para" or "Block"	"Block"	"Para"	Element not required
Para/Block contains <text>, not mixed content	<text>	Okay	Not applicable
Character of <text> element (12.4)	Preference for @text.flow, which determines whether the element is block or inline	<Text> should be a block level element	<Text> is block level except in an inline list.
Element name: <Text> or <TextBlock>	<Text>	<Text>, given need to re-use in inline list.	Item would have a <Text> child
Impact of author's decision whether something is a block list or a sub-clause	[Author must choose whether to insert the items in <Block>, or after it.]	If its a list, put the List in Para (aka Block). If not, put the items after the block.	The block list / sub-clauses go in the content model in the same place.
Container for block list	No	Yes	Yes, called "items"
Content model for block list	Re-use item	Re-use item (with local definition), in Items container.	Re-use item
Location of inline list in content model	[Logical possibilities seem to be child of "block", or child of "text"]	Preference for placing it within <Text>	In <Text>

Point of Concern	***GP1***	***JH Suggestion assuming Grammatical Paragraph Model***	***JH Suggestion assuming Simple Paragraph Model***
Content model for inline list	Not addressed yet	Re-use item, in Items container.	Re-use Item
Mechanism for handling block quotations	@text.flow	Point solution involving specific element	Point solution involving specific element