Structure and Writers' Productivity

Advantages of Hierarchical Editing

by Lynne A. Price, Ph.D.
Text Structure Consulting, Inc.
lprice@txstruct.com

Lynne A. Price is president of Text Structure Consulting, Inc., a consulting company that specializes in XML and SGML, particularly training and development of structured FrameMaker and FrameMaker+SGML applications. Prior to founding Text Structure Consulting in 1996, Lynne was a software engineer at Adobe, which she joined through Frame Technology as one of the original developers of the tool that eventually became FrameMaker+SGML. Lynne has been active in the SGML community since 1985. She participates in both US and international SGML standards committees. Lynne's interest in structured documentation began in graduate school. She completed a Ph.D. in Computer Sciences at the University of Wisconsin-Madison in 1978. Her dissertation was titled *Representing Text Structure for Automatic Processing*.

The Hierarchical Model

Use of structured documents in FrameMaker is often motivated by a requirement for working with XML or SGML. This paper shows that the structured editing environment provides numerous features that allow writers to be even more productive than they can be working directly with FrameMaker paragraph and character formats and other formatting catalogs. It therefore recommends that writers seriously consider structured documents even in situations where there is no need to format existing XML or SGML material or to produce XML or SGML versions of final results.

Both structured and the familiar unstructured documents use the same formatting engine. Formatting basics—including paragraph, character, and table formats, variables, master pages, cross-references, tables, anchored frames, generated files, text insets, flows, and so forth—are common to both. The difference between the two types of documents is the basic organization of data within a flow. In unstructured documents, a flow consists of a sequence of tagged paragraphs. Although a paragraph format may specify a "Next Paragraph Tag" to help a writer create an appropriately tagged new paragraph, the formats used in successive paragraphs are independent. The software does not prevent nonsensical sequences such as the first paragraph in a document being tagged **2 Bullet**, the next being tagged **Heading3**, and the third one **Main Title**.

While the data model underlying structured documents is more complex, it is quite intuitive. Here, a document consists of a hierarchy of elements, with each element in turn consisting of smaller elements, text, or some combination. For example, an entire document may be an element tagged **Chapter**. The **Chapter** element may begin with a subelement tagged **Title** that is followed by a sequence of **Paragraph**, **List**, **Table**, and **Figure** elements and that sequence by some **Section** elements. Not surprisingly, each **List** consists of **Item**s, each **Item** of **Paragraph**s, subelements of a **Table** include **Row** and **Cell**, and the structure of each **Section** is similar to that of the **Chapter** itself. Furthermore, subelements may occur within paragraph-level elements. A **Paragraph** may contain **Quote**s, **EmphasizedPhrase**s, **Citation**s, **Cross-Reference**s and the like. While character formats may be associated with such **text-range** elements, a change in appearance is not necessary.

As shown by several examples below, the advantage of this structured model comes from the author's ability to manipulate entire structures of elements instead of working with one paragraph at a time.

Context-Sensitive Formatting: An Example

A simple reorganization of a document illustrates how the element hierarchy assists a writer. Suppose a biologist is describing measurements of the shells of various sea snails. In a rough draft, a description of the sample animals may be part of a section describing the materials and methods of his research:

Materials and Methods

Sample
The diversity of specimens affords phylogenetic breadth. Quantitative data were taken from 30 specimens in 7 species with columellar folds and 5 species without columellar folds. Qualitative observations on an additional 9 species supplement these data…

Dissections
The outside of each shell was marked along the union between two whorls every quarter…

Figure 1 Organization of a rough draft

Upon review, the researcher may decide to move the description of the sample before the "Materials and Methods" section, so that the document is changed as follows:

Sample

The diversity of specimens affords phylogenetic breadth. Quantitative data were taken from 30 specimens in 7 species with columellar folds and 5 species without columellar folds. Qualitative observations on an additional 9 species supplement these data…

Materials and Methods

Dissections
The outside of each shell was marked along the union between two whorls every quarter…

Figure 2 After reorganization

To achieve this result in an unstructured environment, the paper's author must perform the following individual steps:

1. Set an insertion point at the beginning (or the end) of the material to be moved.
2. Extend the selection to the opposite end of the material to be moved. (While only the title and part of a single paragraph are illustrated here, it is quite possible that the section occupies several pages. If so, the writer must be careful to select the entire section but no surrounding material.)
3. Cut the selection to the clipboard.
4. Set an insertion point at the desired location.
5. Paste the clipboard contents to the new location.
6. Change the paragraph format of the moved title from that of a subsection to that of a primary section.

The manipulation is simpler using the structured approach:

1. Select the section to be moved. As shown under Using the Structure View below, there is no need to extend the selection from one end of the section to the other, it is treated as a single data object.
2. Drag the section to the new location where it is automatically reformatted.

A key part of what makes this operation convenient for the writer is the automatic formatting. Built into a structured template are **format rules** that specify how an element looks in different contexts. When the context changes, because the user moves the element or adds new content adjacent to it, FrameMaker automatically reformats the element appropriately. Thus, the author can concentrate on creating the words and organization of his document and need not bother about explicitly applying formats. The success of such operations does, of course, depend on a well-designed template that provides element types and format rules appropriate to the content just as the usability of an unstructured template depends on the usefulness of the formatting tags it provides.

Techniques of Structured Editing

The user interface for working with structured documents is essentially the same as that for editing unstructured documents. There are a few additional tools and operations that allow the author to manipulate elements.

Using the Structure View

One tool used with structured documents is the **Structure View**. Figure 3 illustrates the Structure View for the document just discussed:

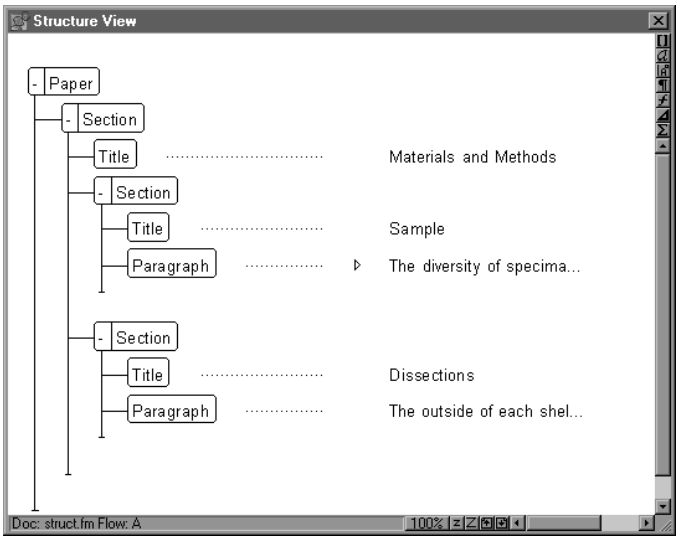

Figure 3 The Structure View

As shown, this optional window represents each element in a document with a "bubble" and indents bubbles to show the nesting of elements. Text next to the bubbles allows the user to match the portion of a document that appears in the Structure View with that in the document window.

Note that Figure 3 includes three bubbles tagged **Section** and three called **Title**. While these **Section** elements are containers that hold text only within subelements, the **Title** elements are individual FrameMaker paragraphs. The first **Title** is that of a major section; its formatting in the document window is quite different from that of the other **Title**s which are within subsections.

This use of the same tag in multiple contexts is another advantage of the structured model. An element like **Title** can occur as the title of a chapter or appendix, as well as the head of a section at assorted nesting levels, or the title of a list or procedure. Its formatting in these contexts varies considerably. In an unstructured document, different paragraph tags would be used for each of these purposes, and the author would need to remember the various names. In the structured document, the author need only remember one tag: **Title**, **T**, **Head**, or whatever name is chosen. The document's format rules change formatting properties appropriately depending on the context in which this element occurs.

The Structure View does more than simply display the structure of the current document. The user can click in it to set an insertion point or make a selection. These operations can also be made in the document window, but the Structure View provides an especially simple way to select an entire element—clicking on a bubble selects the corresponding element, including all its text and subelements. For example, the user can click on the second **Section** bubble here to select the entire "Sample" section, including its **Title** and **Paragraph** subelements:

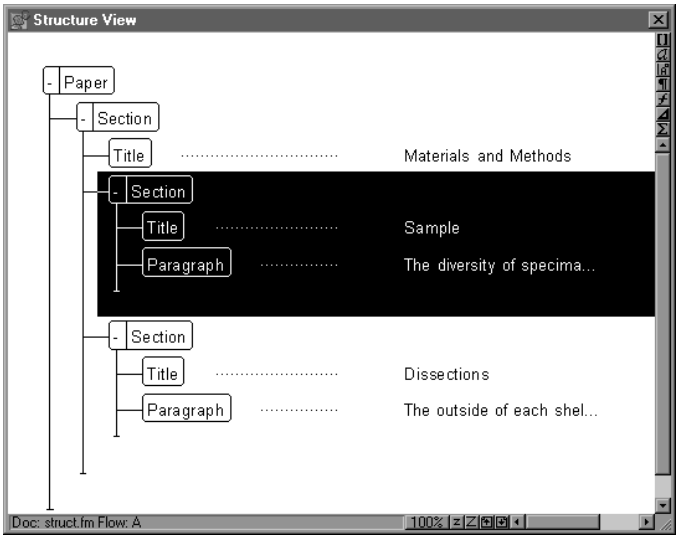

Figure 4 Selection in the Structure View

The Structure View also makes it easy for writers to rearrange material, since the user need only drag a bubble to a new location to move the element. Some users prefer to open the Structure View only when doing this type of editing.

Navigation

Often during editing, a user needs to find the end of the current major division of a document, or the start of the next one. In an unstructured document, an author who remembers to do so can search for the tag of a paragraph that might follow the current section. Within a subsection, thus, he might look for a subsection head, section head, or chapter title. Too often, though, the author relies on his own ability to recognize the start of a section as he scans through the document, hoping that when he finds one section head he has not overlooked another.

In a structured document, navigation is easy. By pressing a platform-dependent modifier key (the meta key on UNIX, the control key on the Macintosh, and the control and alt keys together on Windows) along with one of the four arrow keys on the keyboard, the user can easily move from element to element. In particular, the down arrow moves the insertion point past the next element whether that element is a range of text within a paragraph, or a complex structure such as an entire paragraph; up arrow moves the insertion point in the opposite direction. The left arrow moves into the current element. For example, before a **List**, the left arrow might move the insertion point into the **List** but before its first **Item** from where repeating the left arrow would move the insertion point into the **Item**. The right arrow moves the insertion point into progressively less deeply nested contexts: from inside an **Item** to the containing **List** to the containing **Section** to the containing **Chapter**.

The Context-Sensitive Element Catalog

Navigation, selection, and formatting depend solely on the hierarchy of a document's element structure. Another aspect of FrameMaker's hierarchical model is that there are **content rules** governing the contexts in which various elements are permitted. A structured template, for instance, may require that every **Section** begin with a **Title** and prohibit **Title**s elsewhere in a **Section**. Somewhat similar to the familiar Paragraph and Character Catalogs, the structured user interface includes an optional **Element Catalog** window. Unlike the Paragraph and Character Catalogs, though, the Element Catalog is context-sensitive and changes as the user navigates through the document. Figure 5 shows how the Element Catalog might appear:

- Between **Paragraph** elements, where **Figure**s, **List**s, additional **Paragraph**s, and **Table**s are permitted and a new subsection can start,
- Within a **Paragraph**, where text **Citation**s to the scientific literature and names of **Species** are permitted, and
- Within a **List** where only **Item**s can occur.

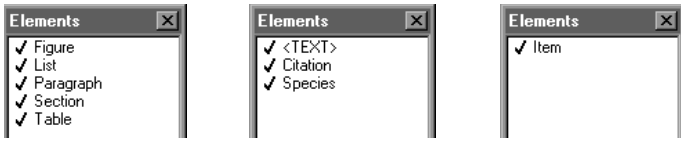

Figure 5 The Element Catalog

Since the Element Catalog changes with the context, at any one time it usually contains a relatively short list. This concise list of available elements makes it relatively easy for the author to find a needed element type. In most applications, the number of items displayed in the Element Catalog is significantly smaller than the number of paragraph tags that would appear in the Paragraph Catalog of an unstructured version of the same document because:

1. Since one element type may be formatted in different ways, fewer element types than paragraph formats are needed

2. Only those element types permitted in the current context are usually displayed

Validation

In addition to supporting the context-sensitive nature of the Element Catalog, a structured document's content rules also support the notion of **validity** of a document, that is, whether or not a document's element structure matches the content rules. A validation dialog box, parallel in many ways to the spelling checker, allows the user to view and correct successive structure errors. Furthermore, the Structure View uses various conventions to indicate validation errors. The "hole" under the second **Section** bubble in Figure 6 indicates that this **Section** does not contain the required **Title**.

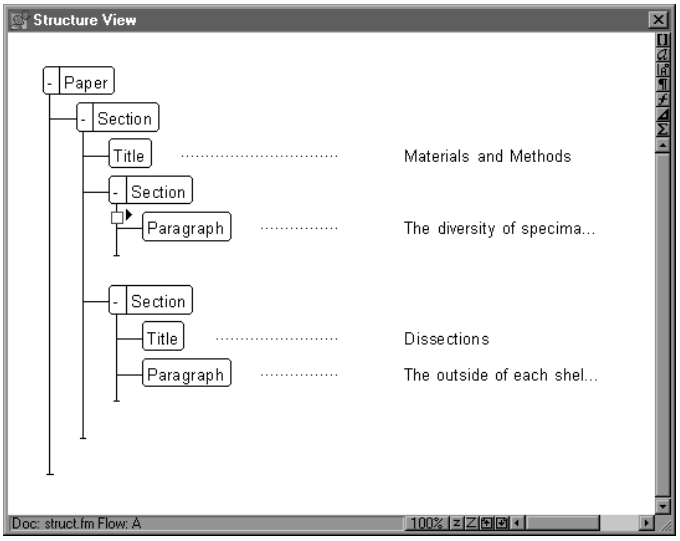

Figure 6 A hole indicates a missing element

A document's content rules define the element types it may contain and their permitted arrangement. They can also express some of the conventions writers in an organization are expected to follow. For example, they may require that every figure be preceded by an introductory paragraph or that there be at least two sections at every level.

Additional Scenarios

Once an author understands the basics of structured editing, there are numerous situations in which the structured model makes editing easier.

Working with Lists

Many documents contain numbered or bulleted lists, and much editing involves making changes to them. As shown in Figure 7, the biology paper used in previous examples may contain a list of hypotheses about the purpose of columellar folds in snail shells:

1 **Columellar folds enhance a snail's ability to maneuver its shell.**
Animals with folds maneuver better because they have a larger surface area of attachment.
2 **Columellar folds enhance a snail's ability to escape predation.**
Snails with folds retract deeper into their shells, making it harder for a predator to reach its prey.

Figure 7 A numbered list

As in an unstructured document, the structured version uses FrameMaker's autonumbering properties to number the list items. In this particular document, the autonumber for the first item is "o:<n=1>\t" and that for succeeding items is "o:<n+>\t". Thus, the first item sets the counter to 1, and the remaining items increment the counter.

A New First Item

To add a new item at the beginning of an unstructured document, the author must remember to use two different paragraph formats. He might insert a new item using the paragraph format for starting a new list. The result is shown in Figure 8.

1 **Columellar folds guide the columellar muscle as the animal moves in and out of its shell.**
Essentially, the folds act as a railroad track guiding a train, preventing the animal from slipping in its shell.
1 **Columellar folds enhance a snail's ability to maneuver its shell.**
Animals with folds maneuver better because they have a larger surface area of attachment.
2 **Columellar folds enhance a snail's ability to escape predation.**
Snails with folds retract deeper into their shells, making it harder for a predator to reach its prey.

Figure 8 A new item at the beginning of a list

The first two items are both numbered 1 because, in the unstructured environment, inserting a new item at the beginning of the list does not affect the second item. The format of the latter still indicates that the counter should be initialized. In a structured document, though, the format rules account for an item's position in the list. When the user inserts a new item at the beginning, the new item's counter is

reset, but the format of the former first (and now second) item also changes automatically. The second item is correctly numbered 2, because its counter is incremented.

Moving Items

Similar processing occurs when the writer rearranges items in the list. If the fist two items are reversed in an unstructured document, the first item in the resulting list will initially have a number that is one higher than the list item in the previous list in the document, and the second item will be numbered 1. In the structured document, whether the counter is incremented or initialized depends solely on the item's position in the current list and numbering is correct without user intervention.

Changing Between Bulleted and Numbered Lists

Sometimes a user wants to change a numbered list to a bulleted list. In an unstructured document, it is fairly easy to make such changes as long as each list item consists of only a single paragraph.

1 **Columellar folds guide the columellar muscle as the animal moves in and out of its shell.**
2 **Columellar folds enhance a snail's ability to maneuver its shell.**
3 **Columellar folds enhance a snail's ability to escape predation.**

Figure 9 A list of one-paragraph items

Given the list in Figure 9 in an unstructured document, the user can select the entire list and then apply a new bulleted format to it to produce the version in Figure 10.

- **Columellar folds guide the columellar muscle as the animal moves in and out of its shell.**
- **Columellar folds enhance a snail's ability to maneuver its shell.**
- **Columellar folds enhance a snail's ability to escape predation.**

Figure 10 The list is now bulleted

More work is necessary if the items have multiple paragraphs as shown in Figure 11.

1 **Columellar folds guide the columellar muscle as the animal moves in and out of its shell.**
Essentially, the folds act as a railroad track guiding a train, preventing the animal from slipping in its shell.
2 **Columellar folds enhance a snail's ability to maneuver its shell.**
Animals with folds maneuver better because they have a larger surface area of attachment.
3 **Columellar folds enhance a snail's ability to escape predation.**
Snails with folds retract deeper into their shells, making it harder for a predator to reach its prey.

Figure 11 Multi-paragraph list items

Here, the author must apply the bulleted format separately to the first paragraph in each item. If there are numerous items, this tedious task can be time-consuming. In an appropriately-constructed structured document, though, it is a one-step operation. The Structure View for the list appears in Figure 12.

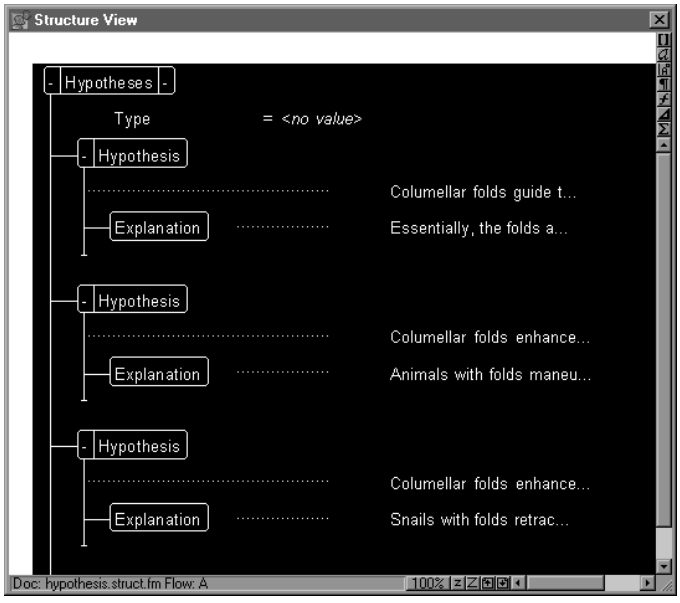

Figure 12 Structure View for a list of hypotheses

The list is coded as a **Hypotheses** element with each item as a **Hypothesis**. The second paragraph in the item is an **Explanation**. The template has been designed to format a **Hypotheses** element as a numbered list. There are two ways to change it to a bulleted list. If the template provides an alternative element for a bulleted version, something like **UnorderedHypotheses**, the author can reformat the entire list by simply changing the **Hypotheses** element to **UnorderedHypotheses**.

The template designer is unlikely to provide two such similar element types, however. There is an alternative. Note the legend "Type = <no value>" under the **Hypotheses** bubble in Figure 12. One or more named strings called **attributes** can be associated with each element in a structured document. Attributes (like FrameMaker **Comment** markers) can simply be non-printing strings associated with a point in a document. However, a document's format rules can be based on attribute values. In this document, the **Type** attribute has been defined to have two possible values, **Number** and **Bullet**. The user can simply set the value to **Bullet** to reformat the list as a bulleted list.

Text-range Elements

The advantages of structured editing apply to text-range elements within a paragraph as well as to larger elements.

Prefixes and Suffixes

Format rules can apply character formats automatically to text-ranges. Furthermore, they can specify a prefix or suffix to insert automatically before or after the element. For example, prefixes and suffixes can surround a **Quotation** element with directional quotation marks or place square brackets around a **Citation** element. Prefixes and suffixes help ensure consistency throughout a document and save some typing (thus reducing the possibility of typographical error).

Cross-References

In an unstructured document, the source of a cross-reference is always an entire paragraph. In a structured document, cross-references can point to elements. A cross-reference format can hence display the text of a text-range element in another location. Cross-references to text-ranges thus provide an alternative to variables for repeated use of a text string that the author must type only once.

Generated Lists

Generated lists of text-range elements can also be made. For example, a maintenance manual can include a list of all the part numbers mentioned within it. While such a list would have to be created manually in an unstructured environment, FrameMaker can create it automatically in a structured document or book.

Markers and Anchors

Experienced writers who work extensively with FrameMaker often complain about the difficulty of selecting a marker or the anchor for a table or anchored frame. Since every table and anchored frame must be an element in a structured document, and markers can also be elements, structured documents make it much easier to select such non-printing objects.

Furthermore, in an unstructured document, even text symbols provide no visual clue when there are adjacent anchors or markers. The marker symbol in Figure 13 indicates that at least one marker occurs in the displayed paragraph but provides no additional details.

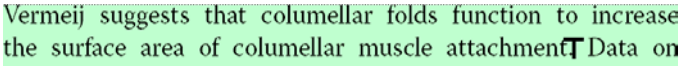

Figure 13 The marker symbol in the document window

However, the Structure View in Figure 14 uses special "square bubbles" for each marker. It not only makes it clear that there are three markers, but displays the marker text as well.

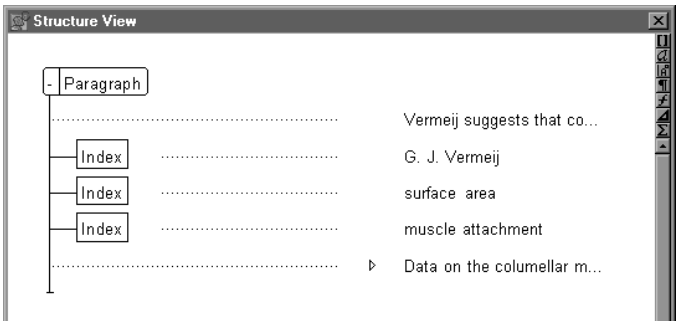

Figure 14 Markers in the Structure View

Changing Cross-References

As a final example of an operation that is easier in the structured world, consider an author who has created numerous cross-references to a single chapter. When he revises the document, he splits this chapter into two. Some of the original cross-references correctly point to the first of the resulting chapters, but the reminder need to be changed to point to the second one. The author therefore wants to search for cross-references to the original chapter and inspect each one so he can decide whether to leave it or change it.

There is no easy way to make the search in an unstructured document. In structured documents, though, two special types of attributes make it easy. A **unique identifier** assigns a string to one element in a document that is used as the unique identifier value for no other element. A cross-reference element then has an **ID reference** attribute whose value is that of the unique identifier of the cross-reference source. (FrameMaker automatically assigns the ID reference and can assign the unique identifiers as well.) With the hypothetical chapter in this example, suppose the chapter's

unique identifier is **ColumellarFold**. Suppose the cross-reference element tag is **Xref** and its ID reference attribute is called **Linkend**. To find cross-references to this particular chapter, the author simply searches for **Xref** elements with the **Linkend** attribute set to **ColumellarFold**.

Concluding Observations

Although FrameMaker's structure model was designed to match that of SGML and XML, users do not need to learn XML or SGML syntax to edit structured documents or even to create structured templates. Like the unstructured environment, FrameMaker's structured environment is WYSISWYG. Editing structured documents does not require looking at angle brackets or reading text interspersed with tags.

Rules for permitted content as well as automatic formatting help enforce consistency within a single document as well as across a set of related documents. This enforcement saves writers effort both when creating material and when reviewing it.

Because formatting is automatic, writers are free to concentrate on content rather than appearance. Automatic formatting saves time in two ways—writers are spared from the need to apply formats manually, and there attention is also not distracted to a formatting task.

These various advantages are not without cost. Development of a structured template requires more sophisticated skills than does development of an unstructured template. Extra development time may be required for analysis and planning of the documentation to be created. Of course, such document analysis on its own can help produce a better organized documentation set.