Breakdowns in writing intentions when simultaneously deploying SGML-marked texts in hard copy and electronic copy

D. G. HENDRY
Human–Computer Interaction Design Laboratory, Department of Computing and Information Science, University of Guelph, Guelph, Ontario, Canada N1G 2W1

Abstract. Many writers of technical documentation must consider two different presentation media, namely traditional printed books and electronic forms. This appears to be a long-term situation, not a transitional phase: for some reading tasks, hard copy will be preferred, but for others, electronic copy will be preferred. In some settings, it is thus necessary to prepare material that is of high quality in both media, often with the constraint that a single source file be used. The problem is to specify the structure of a text so that whether it is printed or deployed electronically, neither version contains textual problems caused by its dual role. Several examples are presented to show how a writer’s structuring intentions can be effective in hard copy but not in electronic copy. The difficulty of preserving structuring intentions in both media stems from declarative markup languages that are rhetorically impoverished. While standard markup languages can be used to specify what text elements comprise a text, they cannot be used to specify the intended roles of the text elements. To preserve structuring intentions, it is proposed that a rhetorical markup language is needed. Two potential advantages of such a language are improved media-transferability and improved visibility of text structure.

1. Writing simultaneously for print and electronic forms

1.1. Interactive computing systems for reading and writing

As in the period of incunabula¹ when literates shifted from reading scirial manuscripts to reading printed books, those who read and write electronic materials are participating in an enormous change. That the nature of reading and writing is changing is readily seen on a cursory look at the many research papers and books on hypertext (e.g., Barrett 1988, Bolter 1991, Conklin 1987, Delany and Landow 1991). This paper examines one phenomenon that is forcing additional burdens on some writers of technical documentation, namely that of preparing material in a single source file that is to be both printed on paper and deployed electronically.

In this paper hard copy refers to reading material that is printed on paper. In contrast, electronic copy refers to reading material that is deployed in an interactive computing system. Two examples of such systems are SuperBook (Egan et al. 1990) and BookManager (IBM 1990). These systems are designed for users of technical documentation and they illustrate the kinds of adjustments that readers and writers will face. In fact, the examples given later are based on material deployed in BookManager, although they are illustrative of any computer system that deploys SGML-marked texts.

SuperBook radically changes how readers search and read material because the primary method of access is through a technique called context-guided search, a novel combination of full-text search and the table of contents. Traditional books have provided tables of contents and indices as the primary access aids. In SuperBook, however, readers most often find what they want by first performing a full-text search. The results of the search, in the form of hit counts, are posted in the table of contents on a per-section basis. Readers then scan the table of contents with a fish-eye view, looking for relevant sections. The fish-eye view emphasises those sections with high hit counts, and helps readers disambiguate section headings and choose relevant material. The synergistic combination of these electronic access methods is claimed to be superior to the traditional methods. Some empirical research supports this claim, although little is known about its generalizability (Egan et al. 1990).

For example, in a direct comparison of SuperBook and hard copy, subjects were required to answer a question by

¹Author’s present address: David Hendry, School of Computer & Mathematical Sciences, The Robert Gordon University, Aberdeen AB9 1FR, Scotland.
writing a short essay. The results showed that those who used SuperBook wrote better essays than those who used hard copy. This performance difference was accounted for by the fact that SuperBook highlights target search words thereby encouraging subjects to focus on key parts of a section (Landauer 1991).

BookManager is an interactive computing system for reading technical documentation that is available in versions for several different operating systems, including VM/CMS, DOS, and OS/2. BookManager deploys a common source file, which is prepared with a declarative markup language, in any of these versions. (Of course, the source file can also be printed.) Along with providing full-text search and a scheme for ranking sections containing target search-words, each of the versions provides a table of contents, an index, and other provisions for moving around the document. On first glance these electronic access methods appear merely to mimic their hard copy ancestors; however, they can in fact be quite different. For example, the section headings in the table of contents, which indicate the hierarchical organization of the subject matter, can be expanded and collapsed to show more or less detailed headings. Although not as radical as SuperBook’s fish-eye view, the expandable/collapsible table of contents is certainly operationally and visually different from a hard-copy table of contents.

Empirical research on how people use BookManager (Hendry et al. 1991a, Hendry et al. 1991b, Nonnecke 1992, Carey et al. 1993) suggests that it is used differently than hard copy, although a direct comparison was not carried out. For example, we believe that scrolling through the table of contents is, at times, used as a substitute for leafing through pages when an overview of the book is sought.

These systems for reading technical documentation are changing the manner in which reading and writing tasks are carried out. Understanding the details of what these systems offer, how they differ from hard copy, and how they affect our strategies for reading and finding material will help both writers, who are preparing electronic copy, and designers of new systems. Examining the problem of preserving writing intentions in both hard copy and electronic copy, given that a single source file is used, is one way of addressing these issues.

1.2. Selective use of hard copy and electronic copy for reading tasks

Anecdotal evidence suggests that people prefer hard copy for some tasks, but electronic copy for others. For example, when reading a printed book, have you ever thought ‘I wish this book was available electronically so I could search for such and such a word’? Or, when reading electronic copy, have you ever thought ‘I wish this was in hard copy so I could get a better sense of its size and organisation’?

Given the striking differences between hard copy and electronic copy, the enormous variety of reading materials, and the many varied reasons for reading, selective use of the two forms should be expected. In fact, in a field study (Granda, Halstead-Nusslock & Winters, 1990), experienced computer professionals judged hard copy to be more effective than electronic copy for learning how to do something new. On the other hand, electronic copy was judged more effective for refreshing one’s memory on something already learned. This research supports anecdotal evidence that for some tasks hard copy is better suited than electronic copy, but for other tasks the reverse is true.

The following three scenarios illustrate selective use of hard copy and electronic copy.

1.2.1. Electronic copy for reference: When working on a computer (e.g., programming, graphing data, editing sounds, etc.) material often must be consulted for particular detailed facts. You may need to be reminded about the syntax of a given command, to find the name of a command that provides such and such a function, to learn how to change the attributes of a particular object, or to learn the meaning of something you see. Common to these so-called referencing tasks is that you know something for which you need to know more detail. For referencing tasks, you have a very clear notion of what information will help, if not the exact vocabulary of the material you seek. In these cases being able to retrieve information directly from your working environment and where reading is minimal, electronic copy is likely to be preferred by many.

1.2.2. Hard copy for comprehensive reading: Texts are often consulted with only a vague notion of what is needed or when a new body of knowledge is to be learned in depth. One example is finding material that bears on a problem that arises because of a fundamental misconception. Another is assimilating material that is dispersed throughout a large text. Characteristic of these so called comprehensive reading tasks is often an uncertainty for what material is needed and an emphasis on needing to make sense of the structure of the material. For tasks like these many people may prefer hard copy because of its familiarity, sharpness of typography, and tactility which aids orientation and flexible access.

1.2.3. Hard copy to facilitate access to electronic copy: In the future it is likely that many workers will have access to technical documentation that is distributed electronically, but not in print. The reason for this is that to save money management may buy electronic documentation only. In cases like these, users may wish to print out the table of contents or the index and use the hard-copy versions of these access methods for accessing material in electronic form. (In day-long usability studies of electronic documentation, we observed several participants print out the table of contents.
for this purpose, Hendry et al. 1991b). Of course, workers may also decide to print out relevant sections or complete chapters and later use these fragments as a basis for further reading of the electronic documentation.

Incidentally, in past shifts in writing technology the older technology was quickly superseded, a result being that there was little selective use of the older and newer technologies (Eisenstein 1983, McArthur 1986). Today, hard copy has not been marginalized because it generally has many properties that electronic copy does not have, including portability, familiarity, standardization, archival longevity, typographical sharpness, visual stability, tactility, and so on. Undoubtedly, hard copy and electronic copy will coexist for some time to come (see foreword in Delany and Landow 1991).

1.3. Multiform writing using a single source file

Simply stated, multiform writing refers to the activity of preparing material in a single source file that is to be deployed in several forms, including, for example, in print and in an interactive computing system. Multiform writing can be much more complex than writing for a single presentation format because, instead of attending to the properties of a single form, writers must juggle the constraints of two output forms when making writing and design trade-offs. The likelihood that people will continue to use hard copy and electronic copy selectively, together with the need for organizations to strive for cost effectiveness, necessitates this kind of writing.

Underpinning the enterprise of developing and popularizing systems that support multiform writing are two principal criteria that developers should measure their systems against:

- **Effectiveness.** The long term goal is that the electronic form of a source file is at least as effective as the hard-copy form. That is, the electronic form becomes a true substitute, not merely a supplement.
- **Economy.** The writing effort required to create the source file is reasonable. The writing effort may potentially be more than writing for a single medium, but the goal is that it should be substantially less than writing separately for two media.

Rarely are systems measured against these criteria and although there is some empirical data addressing the first (e.g., Egan et al 1990), there is none that bears on the second.

One approach for improving the effectiveness of systems that support multiform writing is to find examples where a writer’s intentions for structuring material are represented accurately in one medium but not in the other. Such examples can be used to isolate and fix weaknesses in the notation used to mark the text, in the procedures for printing or deploying the text, in writing practice, or in all of these. The goal of this paper is to present examples where a writer’s structuring intentions are not preserved in both media and to argue that the underlying cause is that the notation used to markup the text does not capture the writer’s intentions well enough (thus, only the first of the above criteria is addressed).

2. Using a declarative markup language to specify structure

Before it is possible to discuss some textual problems that may result from multiform writing and to build a case for their prevention, it is first necessary to consider how the elements comprising a text, and their relationships to each other, are specified in a source file. This section first examines how a set of core tags, which are likely to be found in all declarative markup languages for technical writing, are used to specify the structure of a text. Below, the term *structure* refers to *syntactical* structure; that is, what elements in a text are marked and what the markings specify about the hierarchical relationships between elements. This notion of structure contrasts with *rhetorical* structure, where it is possible for readers to perceive the units of material and their relationships, as the writer expected, but differently from what is actually coded in the text. Once the use of the tags is examined, some weaknesses of them are discussed and a rhetorical markup language is sketched. The goal here is to show that while these core tags allow the writer to specify what text elements comprise a text, they do not allow the writer to specify their rhetorical roles. Section 3 presents examples that illustrate a variety of textual problems which occur because of this deficiency.

2.1. Specifying text structure

Markup formatters can be categorized, based on the function of their tags, into two types: procedural and declarative. With a procedural markup language, writers use the ‘tags’ (‘commands’ is the more appropriate term) to specify how all text elements are to be formatted. With declarative markup, however, writers use tags to specify *what* the text elements are, but now *how* they are to be formatted. The commands for formatting the elements are stored separately in what can informally be called a style sheet. One style sheet might contain the typographic rules for printing marked texts on A4 sized paper, another might contain the procedures for laying them out in resizable windows and for creating an expandable/collapsible table of contents of selectable headings, like the one in BookManager. This separation of a text’s logical structure from its physical realisation allows for the preparation of structured documents that are independent of a particular output form (Andre et al. 1989). Thus in theory, the same text can be printed or deployed in a variety of physical forms.
Often, declarative markup languages are tailored to particular types of writing. The tags needed for marking up a business letter are different from those needed for a departmental memo or for a school textbook. For letters, an ‘inside-address’ tag is needed. For memos, ‘to’ and ‘from’ tags are needed. For textbooks, a ‘title-page’ tag is needed. Of course, there is often some overlap in what tags are needed for different types of writing. For memos and letters, a ‘closing’ tag is needed, and all three of these document types will likely share a set of core tags, such as the ‘paragraph’ tag for marking paragraphs; a ‘list’ tag for creating, for example, enumerated and bulleted lists; and ‘heading’ tags for organizing the text into topics and subtopics. Creating a declarative markup language requires a very good understanding of what text elements comprise a class of documents and how the documents are likely to be processed—yes, printed, but also how they are to be used by other applications.

Suppose a writer wishes to create a markup language for a class of documents, say a genre of poem. How does he or she define the tags and the grammar of how the tags can be combined so that potential any system could process the poems? This is the raison d’être for the Standard Generalized Markup Language (SGML). SGML is an international standard that provides a framework and rules for specifying declarative markup languages (see Goldfarb 1991 for an annotated description of the ISO 8879 standard). A writer, likely in collaboration with a software technician, could follow the rules to define a set of tags and a BNF grammar (the details are complex but we can ignore them). The tags, grammar, and some other necessary information are kept in what is called a Document Type Definition (DTD). When a tagged poem is bundled together with its DTD any system that adheres to SGML could potentially process it.

Figure 1 illustrates what a technical writer sees when marking a text, in this case one chapter from a manual on a complex interactive database application. The chapter explains several commands for formatting an on-screen report. It begins with a chapter heading and some introductory material, which includes two paragraphs, an example, another paragraph, and an unordered list. Following this are three task-oriented sections, each of which explains a single command for formatting a different aspect of a report. The chapter ends with a summary of the commands that were explained in the task oriented sections. When the chapter is printed, the paragraphs consist of plain text with a blank line before and after; the examples consist of an image of the on-screen report, a command, and an image of the report after the command is applied; and the headings are similar to the ones in this very paper.

This source file is a simplification of an actual one, created by a technical writer in a commercial setting. For ease of presentation, it has been simplified by reducing the number of task-oriented sections from six to three, by shortening the original headings, and by reducing the number of paragraphs within the section headings.

To code the chapter six different tags were used: Example, Head_1, Head_2, Item, Paragraph, and Unordered_List. Here, these tags (plus Head_3 which was not used in figure 1 but is used later) are referred to as the core-set to reflect the fact that they are available in all but the most specialised SGML document type definitions. A very high percentage of the text elements that comprise any document, but particularly technical documents, are paragraphs, headings, and lists. The tags in the core-set are assumed to be representative of existing markup languages because they are available in the following document type definitions: ‘general’, an exemplary DTD found in Annex E of the ISO 8879 standard (Goldfarb 1991); ‘textbook’, a DTD suitable for producing a series of computer science textbooks (Bryan 1988); and ‘Ericbook’, a DTD used to publish a textbook on SGML (van Herwijne 1990). They are also available in BookMaster, a

1. **Head_1**: `<How to Format a Report>`
2. **Paragraph**: `..........` `..........` `..........`
4. **Example**: `..........`
5. **Paragraph**: `..........` `..........`
6. **Unordered_List**:
7. **Item**: `<Changing column titles>`
8. **Item**: `<Changing column widths>`
9. **Item**: `<Changing report footers>`
10. **Item**: `<Summary of commands>`
11. **Head_2**: `<Changing column titles>`
12. **Paragraph**: `..........`
13. **Example**: `..........`
14. **Head_2**: `<Changing column widths>`
15. **Paragraph**: `..........`
16. **Example**: `..........`
17. **Paragraph**: `..........`
18. **Head_2**: `<Changing report footers>`
19. **Paragraph**: `..........`
20. **Paragraph**: `..........`
21. **Example**: `..........`
22. **Head_2**: `<Summary of commands>`
23. **Paragraph**: `..........`
24. **Example**: `..........`
25. **Paragraph**: `..........`

Figure 1. Conventional declarative markup showing how the core-set of tags can be used to structure a chapter on how to format an on-screen report. Tags in the core-set are: Example, Head_1, Head_2, Head_3 (not used), Item, Paragraph, and Unordered_List. The chapter begins with an introductory section, then continues with three task-oriented sections, and ends with a summary section.
commercial declarative markup language (IBM 1989). Throughout this paper these tags, as actually used by technical writers in their work, will be used to illustrate some textual problems that can occur in multiform writing.

Returning to figure 1, the writer made several design decisions that were intended to make the chapter effective for readers. These decisions, though, cannot be inferred from the markup shown in figure 1—to infer them, the actual content must be studied. The following are four decisions:

1. **Decision 1**—Write an introductory paragraph that summarizes the purpose of the entire chapter. This is the role of the first paragraph in figure 1, but nothing in the markup specifies that the second paragraph serves a different role. In general, within a series of heading levels, the relations among text elements are unknown, except that elements follow each other.

2. **Decision 2**—Write an initial example that will be used as a basis for explaining the material in each of subsequent task-oriented sections. This is the role of the second paragraph and the example in figure 1. This role, however, is not specified with the declarative markup. In general, the relations among text elements that are within different heading levels are unknown.

3. **Decision 3**—Use a bulleted list of the subsequent section headings to signal what topics are to be covered (created with the Unordered List tag). The paragraph tag marks a topic sentence for what material is to be covered and the items are each of the subsequent headings of the chapter. The role of the list, however, is not specified in the markup. The items in the unordered list could just as easily be subject material that elaborates the preceding example. (By writing convention, when bulleted lists are used the reader is signalled that there is no inherent order to the items. In contrast, when enumerated lists are used—for example ‘i’, ‘ii’, etc.—the reader is signalled to read the items in order.)

4. **Decision 4**—Write a summary section that explains the short-cuts for entering the commands that were explained in the preceding sections. This is the role of the final section (a heading level 2), but this special role is not distinguished from the other sections; the last section could just as easily be another task-oriented section on another aspect of formatting on-screen reports.

In sum, the structure of the chapter is the following: it begins with a brief introduction to the whole chapter (lines 1–2 in figure 1); then a base-example (3–4), which is referred to in each of the task-oriented sections, is presented; then the task-oriented sections are signalled with a bulleted list to suggest that they can be read in any order (5–10); then the task-oriented sections follow (11–13, 14–17, 18–21); finally a summary section of command short-cuts concludes the chapter (22–25). This structure is shown diagrammatically in figure 2a.

These decisions are somewhat independent of the tags used to structure the chapter and some choices of tags, particularly headings, may better reflect the decisions than others. An alternative structure that may represent the writer’s rhetorical intentions more accurately is shown in figure 2b. Two changes have been made. First, the original introduction has been divided into two sections: ‘Intro.’, which introduces the entire chapter (lines 1–2), and ‘base-example’, which presents the base-example and explains that the task-oriented sections will use it (lines 3–10). Second, the task-oriented sections have been made subordinate to the section where the base-example is found (using a heading level 3s). These structural changes to the heading levels may better convey the writer’s decisions because the task-oriented sections are coupled to only the material on the base-example, not also to the introduction to the entire chapter; and the summary is at the same level as the base-example, suggesting that it is more general than the task-oriented sections. In fact, readers who are familiar with other similar database applications might be able to read the summary only and glean all the information given in the task-oriented sections.

The crucial point here is that although the tags from the core-set can be used in different ways to structure the chapter, they cannot be used explicitly to code the rhetorical roles that the writer had in mind for the sections. Nothing in the markup specifies that the first heading level 2 introduces a base-example, that the second heading level 2 summarizes the task-oriented sections, or that each of the task-oriented sections uses an example given earlier. It could be that each lower section (i.e., the material under heading level 3s), instead of describing how to complete a single task, actually describes a different component of an object, which is first
introduced in the higher section. In this case the higher section may not need to be consulted to understand one of the lower sections. Or, it could be that to fully understand any one component of the object, one must be at least familiar with the other components; that is, the material is interdependent. Or, it could be that the sections outline a sequence of subtasks that must be followed in order to complete a complicated higher-level task. The tags in the core-set only allow the writer to say what the text elements are, not how they work together to achieve a communication goal.

2.2. Modelling rhetorical design decisions

When readers do not recognize the dependencies between units of material, a symptom of being unaware of the rhetorical structure, they can encounter difficulties, and this seems to be especially true in electronic copy. In one laboratory study (Hendry et al. 1991b), the chapter, as structured in figure 2a, was presented in BookManager and experienced programmers were asked to complete a set of database querying tasks. One task required participants to change the format of an on-screen report by applying the commands that were explained in one of the task-oriented sections. One participant quickly found the relevant material with a full-text search; read it, without scanning the neighbouring material; and thinking he understood it tried to complete the task at hand. After two or three unsuccessful attempts at entering a series of commands he seemed to realise that although he was in the correct section, he didn’t understand the material. Unfortunately, he seemed unsure of the source of his misunderstanding. Eventually, after considerable unproductive scanning of the document, he came to the section with the base-example, corrected his misconception, and then completed the task successfully, after which he reflected on his performance, saying in part:

If I had a paper thing, I would flip [turn the page] but because it is on-line I don’t want to lose the place I am at . . . I didn’t want to get out of where I was because it was too much of a bother . . . that’s quite a negative thing.

Tentatively, two conclusions can be drawn from this reading episode. First, the person indicates a reluctance to consult neighbouring material because he seems to fear disorientation, a pervasive problem in electronic copy (Nielsen 1990). But, if the manual was presented in hard copy he seems to believe that he would have consulted it, perhaps because of the affordances provided by the tactility of a book. Second, if the dependency between the task-oriented section and the base-example were signalled in BookManager, perhaps he would have gone directly to the base-example, even before reading the section on how to complete the task.

How could this reading problem be prevented? One possibility would be to eliminate the dependency. The writer could repeat the base-example in each of the task-oriented sections; that is, the writer could make the task-oriented sections completely modular. A consequence of this, however, is that a redundancy would be introduced into the chapter: the base-example would appear three times instead of just once. Here, the writer faces a design trade-off. On the one hand, she could make each of the sections modular and accept the chapter-wide redundancy. On the other, she could eliminate the redundancy but create a hidden dependency. The writer chose the latter trade-off and may have sound rhetorical or practical reasons for doing so. A rhetorical reason could be something like: most readers will read the entire chapter and thus will read the base-example first, should not be forced to plough through the base-example in each of the task-oriented sections, and, in any case, the base-example is only three pages away (one page flip). A practical reason could be something like: keep the chapter-wide redundancy low so page counts are low. Importantly, the best trade-off may depend on where the chapter is presented: in hard copy, low redundancy but a hidden dependency may be optimal; in electronic copy, highly modular sections with a chapter-wide redundancy may be the most effective.

To examine this trade-off in some detail, it can be modelled using a formalism called Entity–Relationship Modelling for Information Artefacts (ERMIA) (Green 1991). Briefly, the formalism is used to represent the entities and relationships that comprise an information artefact, potentially any type of structured information (e.g., bar graphs, music notation, train timetables, controls on electronic devices, etc.). Once represented, rather different information artefacts can be compared in a common notation and structural similarities can be discerned. ERMIA provides a flexible notation for discussing the strengths and weaknesses of information artefacts by highlighting structural properties.

Figure 3 shows three ERMIA diagrams. Taking a technical view, Model-A represents the structure of the chapter that is shown in figure 2a. Here all that is known is that the chapter consists of one introduction which introduces several sections. The entities (‘Chapter’, ‘Intro’, and ‘Section’) are represented by boxes and the relationships (‘consists’ and ‘introduces’) by lines. Model-A is read: ‘one chapter consists of one ‘intro’ and one ‘intro’ introduces several “sections”’. This model highlights the fact that the markup used in figure 1 does not specify the distinctions between the base-example, the task-oriented sections, and the summary.

Taking a writer-centred view, the other two models represent the design trade-offs. Model-B shows that the writer views the chapter as consisting of one introduction (‘Intro’), one summary (‘Summary’), and one introduction to the tasks (‘Intro. to tasks’). Then, one ‘Intro. to tasks’ has one ‘Base-example’ and it introduces several ‘Task-oriented’
sections and each of these has a 'Task-example'. Finally, the dependency between the task-examples and the base-example is represented by the 'makes use of' relationship. Model-C shows the writer's view if she had decided to include the base-example in each of the task-oriented sections, thereby creating modular sections but a chapter-wide redundancy.

Another approach that might alleviate the reading problem would be to change the design of BookManager so that such semantic dependencies in the text are coded and shown to readers. One possibility, of many, for visually representing the dependency would be to place a special icon next to the headings of the task-oriented sections. But, how could the text in the source file be marked to specify that dependencies exist? Certainty, the tags from the core-set could not be used for this purpose. To illustrate one approach, figure 4 sketches a markup language for representing structures like those shown in Model-B (for brevity, a SGML document type definition is not defined). In the figure, rhetorical tags are used to mark the text elements according to the writer's rhetorical intentions.

Inspection of figure 4 shows that several changes have been made to the source file shown in figure 1. The tags 'Chapter', 'Topic', and 'Subtopic' are similar to the heading levels 1–3, respectively, but the topic and subtopic tags use attributes, a provision of SGML. The 'Type' attribute is used to specify the role of a topic; here, a topic is either an introduction ('intro'), a summary ('summary'), or simply a unit of material that plays some other role ('plain'). Two (optional) attributes have also been added to the 'Example' tag for specifying dependencies: 'ID' is used to name examples so that they can be referred to elsewhere in the source file, and 'BasedOnRef' is used to specify that one example is based on another. So, for example, the base-example has been named 'bExp' in line 6, and in lines 14, 17, and 22 the task-examples are coded to specify that they are based on it. 'Unordered Preview' is another new tag, which uses attributes in a similar manner. A writer can use it to signal what topics are to be presented (its role contrasts to that of the 'Unordered List' tag, used in figure 1, but we shall not discuss this point until section 3.1).

These tags illustrate how writers might specify rhetorical structure. Doing so would allow for the design of systems that provided readers greater flexibility in access and more accurate signalling of structure than what is possible with the core-set. As another example, introductions and summaries could be given special treatment—a reader could request that all summaries be listed in a window so they could be quickly scanned, or the introductions could be made readily available from a list of topic headings, perhaps generated by a full-text

---

![Diagram](image-url)
search, so the reader could disambiguate them. The tags in
the core-set are deficient because texts marked with them
cannot be processed in simple ways that accord with how
readers and writers view them.

2.3. Summary

In this section an example was used to illustrate how a set
of core tags, found in declarative markup languages for
technical writing, can be used to specify the structure of a
text. The example shows that writers may have definite
rhetorical reasons for structuring, but these cannot be
specified with the tags. The markup is thus impoverished in
a rhetorical sense. That is, the dependencies among text
elements, created to achieve a communication goal, have not
been specified explicitly. A consequence of this impoverish-
ment is that interactive computing systems often cannot
effectively portray the rhetorical structure and guide readers.
One case was examined where the dependency between a
base-example and some task-oriented examples was not
signalled and this caused trouble for a reader. In the next
section we shall examine several more cases.

Emerging from this analysis is the following question: if,
in addition to specifying what text elements comprise a
document, a specification of their actual roles was given,
what benefits could be gained? After all, a wide variety of
documents have been successfully published without this
further specification, and writing, even with pencil and paper,
is difficult enough as it is. The main benefit is for readers,
particularly readers of electronic copy. Rhetorical tags offer
the potential for presenting material such that its arrangement
is visible and understandable in both print and online.
Material that is highly role expressive would alleviate the
reading problem described above where the reader was not
inclined to scan neighbouring material. If the person was
informed about the dependency between the base-example
and the task-oriented example, he would be more likely to
consult the prerequisite material. Furthermore, and more
importantly, role expressive material should heighten read-
ers’ general awareness for the rhetorical structure of the text,
and this should result in better comprehension (Meyer et al.

3. Deploying texts in electronic form: breakdowns in
writing intentions

As a first step for addressing the effectiveness criterion,
which was mentioned earlier, systems for multiform writing
should deploy texts in hard copy and electronic copy without
introducing textual errors. In the previous section, it was
argued that to do this the rhetorical roles of text elements need
to be specified. This section will elaborate on both these
issues by turning to examples that illustrate breakdowns in
writing intentions that can be attributed to markup tags which
are rhetorically deficient. The examples show that if the tags
specified the roles that the text elements are intended to play,
than certain textual problems could be prevented.

The examples are drawn from technical documentation,
prepared by professionals for print (see Hendry 1991a, for the
details). The technical documentation was written without
knowledge that it would eventually be deployed electroni-
cally; thus, the examples did not emerge from a true process
of multiform writing where writers would balance the
constraints of both media. Nevertheless, the examples show
textual problems that can be expected to occur when written
material must work well in both hard copy and electronic
copy. The examples are intended to capture textual problems
in a succinct form so that rhetorical deficiencies with the tags
can be illustrated. They do not address particular writing
decisions, nor the task of measuring the seriousness of the
problems which result.

Each example consists of two schematic pictures: a printed
and electronic version of the same source file. The printed
versions are assumed to accurately represent the structures
the writer intended to convey, whereas the electronic versions
contain what are called textual anomalies. This term is used
to denote that the structure of the material in the electronic
form drifts from the printed exemplar. The source file, which
is not shown in the examples, is assumed to consist of a text
that has been marked with tags from the core-set (see section
2.1). Importantly, other procedural commands, which might
fix some of the anomalies, are not allowed because that would
violate the main tenet of declarative markup languages: one
specifies what something is not what it is to look like.

Of the many rules for formatting text electronically in
BookManager, only four are relevant. The first rule is that
material contained within heading levels 1 and 2 is treated
as a unit, which is assigned an unique topic number. These
units, hereafter called topics, are analogous to pages: they
provide a means for readers to move through the material
topically by topic-heading, instead of physically by page. The
rationale for prefixing headings with topic numbers is that
they provide readers with information about the location of
topics, both distance from the title page and level of detail
(e.g., 7.2 vs. 4.3.6). This can be extremely valuable when, for
example, a reader needs to decide which topic from a list of
search hits to pursue. The second rule is that heading level
3s are not treated as topics; instead, they are shown in italic
on a separate line within a topic and they are not prefixed with
a topic number. They provide a method for dividing material
within a topic. The third rule is that the system automatically
inserts a list of subtopics after the material in a heading level
1. The subtopics are the level heading 2s for the immediate
subordinate topics and are selectable. There are two good
reasons for inserting the subtopics: 1) they signal what
material is to come and 2) they provide rapid access to
material. Finally, paragraphs, unordered lists, and examples
The electronic copy is susceptible to incoherencies. Although not seen here, the juxtaposition of the subtopic list with the author-created list can, besides producing a redundancy, create incoherencies when items in the two lists do not correspond. This can happen when, for example, the writer summarizes (not signals) forthcoming material in general terms with a list and when the headings for the material contain considerably more detail. In cases like these readers are invited to match items in the general summary with the headings, which is likely not what the writer intended.

- The electronic copy subtly alters the writer’s intentions regarding the ordering of sections. The author’s choice of the bulleted list (seen in the hard copy), indicates that the sections are not ordered. The addition of the topic numbers to the section headings in the electronic copy, however, suggests an inherent ordering to the material (i.e., Eyelet follows Washer and Ring follows Eyelet).
- In the electronic copy, the title subtopics is not really appropriate, since what follows is more precisely signalled as a list of grommets.

The main problem illustrated in figure 5 is that the author-created list signals what material is to come but this important fact is unknown to the system. One solution is to introduce the ‘Unordered Preview’ tag, which was sketched in section 2.2 (see figure 4). With it a writer could distinguish between lists for presenting subject matter and lists for signalling forthcoming material. If the role of the list was specified, the system could suspend the presentation of the subtopic list and present just the preview.

### 3.1. Unordered lists to signal material to come

Figure 5 shows a chapter that describes three kinds of grommets. The chapter is structured simply as an unordered collection of items, each of which is described. The introductory material does nothing more than signal what is to come, and then the material is presented in three major sections (i.e., under level 2 headings). In the electronic copy, the list of subtopics, which is automatically inserted by the system, follows the author-created list. By comparing the hard copy and electronic copy, four anomalies can be seen:

- The electronic copy is redundant, as the list of subtopics is shown twice. One list was created by the author and the other was automatically inserted by the system.
3.2. Cascaded headings to signal two levels of structure

Figure 6 shows a chapter that consists of three major sections on how to maintain (i.e., replace, clean, and paint) widgets and grommets. Here, there is a two level structure: one level divides the material by tasks and the other level divides the material by objects. The power of this structure lies in its parallelism across tasks and that it allows the reader to compare the tasks in terms of the type of object (e.g., either grommets or widgets). This structure is portrayed visually through what can be called cascaded headings, where the second heading completes the message that is begun in the first (e.g., Replacing Widgets, Painting Grommets, etc.). The structure is easy to grasp because the beginning and ending headings are visible across more than one task. Comparing the hard copy and electronic copy, two anomalies can be seen:

- The list of subtopics does not preserve the effect that is achieved with the cascaded headings. So, when looking at the list of subtopics, nothing says that the task replacing concerns both grommets and widgets, although this is mentioned in the introductory section. Only the first of the two level structure is conveyed in the list of subtopics.
- Like the example given in section 2.0, there are hidden dependencies here. Suppose a reader accesses the beginning of topic 2.2, which covers cleaning widgets. At this point, the reader is given no signals of the existence of additional information on cleaning grommets or that there are other tasks for maintaining widgets and grommets. Thus, learning about the two kinds of objects, the maintenance tasks, and the differences between maintaining widgets and grommets, is likely to be difficult.

In this example the system has disrupted the intentions of the writer for how the headings are to work together visually. If there were no compare/contrast structure across tasks, then the electronic copy would be adequate. However, because the only way to learn of the entire structure is by putting together bits and pieces of it, readers can be expected to experience trouble.

Figure 7 shows the structural information that should be shown to readers in one way or another. This is the structure that is conveyed when the headings work together visually in print. In the electronic copy this whole picture is never available at a glance, but of course it could be made available if the intended roles of the text elements were specified to the system.

Incidentally, this example illustrates an invertible structure, which refers to material that could naturally be presented in one of several alternative structures, according to reader preference. For example, some readers may prefer to arrange the material so that the tasks are subordinate to the objects, instead of the objects being subordinate to the tasks (as is shown in figure 6). Invertible structures have been explored using a system that presents multiple tables of contents (Carey et al. 1990) and the specification of invertible structures in technical documentation would allow users more flexible access to material.

3.3 Skipped level headings to signal relative importance

Figure 8 shows a chapter that describes the three steps to follow when recycling widgets. It is structured as an enumerated list of topics and this fact is indicated through sequence numbers in the headings. However, a single special section on terminology preceedes the major sections of the chapter. This special section, while of lesser importance than the major sections, must nevertheless be presented first because it contains a summary of the terminology that is to be used throughout the chapter. To indicate its special role, the writer decided to demote its heading level, using a heading level 3 instead of a heading level 2. This writing decision breaks accepted practice on proper use of a declarative markup language, which dictates that one should organise all sections according to the text hierarchy provided by the tags (e.g., Head_1, Head_2, Head_3; not Head_1, Head_2, Head_3, Head_2). By comparing the hard copy and electronic copy, two anomalies can be seen:

- Designing the system to effectively handle demoted heading levels is difficult because it is not obvious
where to place the subtopics, which signal subsequent material. The difficulty arises because the underlying hierarchical model of the declarative markup language is broken when heading levels are skipped. There are two obvious solutions. First, the subtopics can be inserted after the section on terminology. However, an undesirable consequence of this solution is that the subtopics are distanced from the introductory section. Second, the demoted heading level can be promoted to that of the major sections (as shown in figure 8). This solution, however, eliminates the special role attached to the section on terminology. Unfortunately, neither solution preserves the author’s original intention for how the material is to be structured. That is, readers are to know of both the terminology section and the major sections, but additionally are to recognize the distinction between these two kinds of sections. (See Tazi 1989 and Barnard et al. 1988 for further writing examples that do not conform to a hierarchy of nested text elements.)

- The topic numbers clash with the headings, which contain numbers to indicate that the material follows a sequence.

To prevent the anomalies exhibited in this example, the declarative notation must be enhanced to allow the specification of sections that are sequenced. That way a system could show the difference between a collection of items that is enumerated and one that is not. One possibility would be to introduce an ‘Ordered__Preview’ tag, which would work in a similar fashion to the ‘Unordered__Preview’ tag (see figure 4). Additionally, the notation must be enhanced so that sections that play a special role (e.g., give terminology, summarise all steps, etc.) can be distinguished from the primary sections of a chapter. This could be achieved with a ‘Type’ attribute as was done in figure 4, where topics were typed as either ‘intro.’, ‘summary’, or ‘plain’.

4. Expected implications of a rhetorical markup language

The examples presented in this paper have four common features. They come from published technical documentation prepared by professionals. They were all coded using tags representative of existing markup languages for technical writing. The procedures for deploying the text elements electronically are robust, satisfying many machine-oriented, technical constraints and are, on the whole, very effective. Finally, they illustrate breakdowns, where a writer’s intentions for how the text elements are to be structured are accurately portrayed in hard copy but not in electronic copy.

What is the underlying cause for the breakdowns? One explanation is that the writing practice is inadequate. The breakdowns only occur when certain writing tactics are followed, namely when: base-examples are decoupled from the task-oriented examples, forthcoming material is signalled with a list, material that consists of compare/contrast relations is signalled with cascaded headings, and the special importance of material is signalled by demoting a level heading. This explanation, however, must be rejected on the grounds that subject matter and the communication goals make these tactics compelling from a rhetorical point of view and, beyond that, the tactics work well in hard copy. Why shouldn’t they also work well in electronic copy?

Another potential reason is that the procedures for deploying the text elements electronically are inadequate. Perhaps, for example, the list of subtopics should not be inserted. This would certainly prevent some of the problems but it would make access to the documents significantly more difficult: more paging would be required and orientation would be more difficult with fewer signals about forthcoming material. For expository material that is deployed electronically, more signalling of structure is needed, not less.

Another reason, the one argued for in this paper, is that contemporary markup languages do not allow writers to be explicit about the exact roles of text elements and the relations between them. A consequence of this is that interactive computer systems cannot deploy them as effectively as when they are printed. The examples illustrate this problem and are intended to provoke thought about it.

What are the expected consequences of a rhetorical markup language like the one sketched in figure 4? This work has not progressed to the point where it is possible to go beyond these conjectures:

- Improved media transferability—A rhetorical mark up language would allow for the design of interactive computing systems that do not exhibit textural anomalies akin to the ones illustrated in the previous section. This a necessary but not sufficient step for the design of truly effective systems for multiform writing.
- More strategic reading—A rhetorical markup language would allow interactive computer systems to better show the roles that text elements are intended to play and the relations among text elements. This information invites readers to be more strategic when making decisions about what sections to read and in what order. Furthermore, it informs readers of the dependencies between units of material, something that is crucial in electronic copy where the tactile affordances of a book do not exist.
- More customized access—A rhetorical markup language creates opportunities for readers to customise their views of documents. For instance, readers could show the base-examples along with task-oriented examples or they could remain separated. Readers could show object–task invertible structures either by task or by object. Readers could list all examples within a
A rhetorical markup language opens a variety of new possibilities for deploying texts. For developers there is the challenge to accurately portray a writer’s structuring intentions and to provide effective personalisation features that enable readers to tailor their views of documents. Readers, if they choose, can customise the text and have more responsibility for what it is. Writers can be given a more expressive medium for guiding readers and for specifying texts that can be effectively deployed in several forms. Further research on the effectiveness and economy of using a rhetorical markup language for multiform writing is warranted.

Acknowledgements
Blair Nonnecke, Tom Carey, John Mitterer, and two anonymous referees provided many detailed suggestions for this paper. I’m grateful to Thomas Green, Abdul Halabieh, Dov Lunug, Roger Spall, and Patricia Wright for comments that improved this paper. This work emerged from a joint research project of the University of Guelph (HCI Laboratory) and IBM Canada Laboratory (Toronto Information Development), supported by the University Research Incentive Fund of Ontario and IBM Canada. Facilities of the HCI Laboratory are supported by the Natural Sciences and Engineering Research Council of Canada. I remain responsible for any faults in this paper but I’m grateful for the help of all the members of the research project.

Note

1A book printed at an early date, especially before 1501 (Concise Oxford Dictionary, 8th edition).

References


HENDRY, D. G., NONNECKE, B., LUNUG, D., CAREY, T. T., MITTERER, J. AND SOBIESIAK, R. 1991a, Writing with the BookManager program in mind: examples, summary of use, and writing advice, TR 74.063 Toronto Information Development, IBM Canada Laboratory, North York, Ontario, Canada.


IBM. 1990, SAA BookManager READ/2 (version 1.2.1, SB35-0801-1).

LANDAUER, T. 1991, Let's Get Real: a position paper on the role of
cognitive psychology in the design of humanly useful and usable
systems, in J. Carroll (ed.), Designing Interaction (Cambridge
MCArTHUR, T. 1986, Worlds of Reference: Lexicography, Learn-
ing, and Language from Clay Tablet to Computer (Cambridge
MEYER, B. J. F., BRANDT, D. M. and BLUTH, G. J. 1980, Use of
top-level structure in text: key for reading comprehension in
ninth-grade students. Reading Research Quarterly, 16, 72-103.
NiELSEN, J. 1990, The art of navigating through hypertext,
Communications of the ACM, 33, 296-310.
NONNECKE, B. 1992, Usage monitors as exploratory tools in the
study of online documentation, M.Sc. Thesis, Department of
Computing and Information Science, University of Guelph,
Guelph, Ontario, Canada.
RICHARDS, D. J., McGEH, L. M., LOMAX, R. G. and SHEARD, C. 1987,
Awareness of four text structures: effects on recall for expository
text, Reading Research Quarterly, 22, 177-196.
TAZI, S. 1989, Performing textual structures in documents, in A. J.
R. Furuta and V. Quint (eds), Structured Documents (Cambridge