10-1-1992

SGML-based publishing

Erinne G. Cheney

School of Printing Management and Sciences
Rochester Institute of Technology
Rochester, New York

Certificate of Approval

Master's Thesis

This is to certify that the Master's Thesis of

ERINNE CHENEY
name of student

With a major in _____
has been approved by the Thesis Committee as satisfactory
for the thesis requirement for the Master of Science degree
at the convocation of

OCTOBER 1992
date

Thesis Committee:

Frank J. Cost
Thesis Advisor

Marie Freckleton
Graduate Program Coordinator

George H. Ryan
Director or Designate

SGML-Based Publishing

by

Erinne G. Cheney

A thesis project submitted in partial fulfillment of the requirements for the degree of Master of Science in the School of Printing Management and Sciences in the College of Graphic Arts and Photography of the Rochester Institute of Technology

October 1992

Thesis Advisor: Professor Frank Cost

SAMPLE statements to **authorize** or **deny** permission to reproduce an RIT thesis.

1. Title of thesis_____

_____

I_____ hereby **grant permission** to the Wallace Memorial Library of RIT to reproduce my thesis in whole or in part. Any reproduction will not be for commercial use or profit.

Date_____

2. Title of thesis SGML-Based Publishing

_____

I Erinne Cheney **prefer to be contacted** each time a request for reproduction is made. I can be reached at the following address.

83 Sidney Street
Rochester, New York 14609

Date_____

3. Title of Thesis_____

_____

I_____ hereby **deny** permission to the Wallace Memorial Library of RIT to reproduce my thesis in whole or in part.

Date_____

rev. 1/89

TABLE OF CONTENTS

1. INTRODUCTION....1
1.1. PURPOSE....2
1.2. PROCEDURE....3
2. REVIEW OF THE LITERATURE....3
2.1. THE NEW PARADIGM....4
2.1.1. THE LOGICAL APPROACH....5
2.1.2. THE PROCEDURAL APPROACH....6
2.2. WHAT IS SGML?....7
2.2.1. DOCUMENT INTERCHANGE STANDARD....9
2.2.2. META-LANGUAGE....10
2.2.3. DATABASE CREATION TOOL....10
3. SGML-BASED PUBLISHING SYSTEM....11
3.1. DOCUMENT PREPARATION....12
3.1.1. DOCUMENT TYPES AND STRUCTURES....13
3.1.2. DOCUMENT ANALYSIS....16
3.1.3. ELEMENT LIST....18
3.1.4. CREATING A DOCUMENT TYPE DEFINITION....20
3.2. INPUT....21
3.2.1. CONTEXT-SENSITIVE EDITOR....21
3.2.2. ASCII EDITOR....22
3.2.3. AUTOTAGGING....23
3.2.4. STORAGE....24
3.3. VALIDATION....25
3.4. CREATING OUTPUT....25
3.4.1. BATCH FORMATTER....27
3.4.2. INTERACTIVE FORMATTER....28
3.4.3 DSSSL....29
3.4.4. OUTPUT TYPES....30
4. CONCLUSION....31

TABLE OF CONTENTS

FIGURE #1. 33
FIGURE #2. 34
FIGURE #3. 35
FIGURE #4. 36
FIGURE #5. 39
FIGURE #6. 43
FIGURE #7. 45
FIGURE #8. 47

BIBLIOGRAPHY. 48

1. INTRODUCTION

The 1990's will be noted as the start of true electronic publishing. The requirements for electronic publishing lie in its definition: a system that has the ability to create, manage, manipulate, integrate text and/or graphical information for subsequent output to a variety of output devices. With the introduction of The Standard Generalized Markup Language (ISO 8879) in 1986, a new era in electronic publishing began. SGML is a meta-language that can be used to create descriptive markup language definitions for document processing and document interchange. SGML is proving to be a tool that will enable electronic publishing systems to expand and embrace the future.

The future holds the promise of an ideal publishing system that: would handle information in a way that allows for the most flexibility in data retrieval; could provide the easiest editability; could achieve high productivity; should have the ability to encode data so that it can be interchanged across platforms and among applications.

The issue of productivity has been the most problematic for electronic publishing systems. The current methods for managing, manipulating, and integrating information before formatting for final output are often slow and extremely complex. To prepare information to be output in a variety of forms drastically decreases productivity

because it is often necessary to re-code documents with new formatting instructions for each type of output. The Standard Generalized Markup Language can be used to prepare data only once for storage so that it can be re-used and assembled into a variety of formats. SGML may be the answer for some electronic publishing applications. Enterprises that are currently attempting to produce a variety of publishing products such as: CD's, books, manuals, on-demand printing, customized documents, multi-media, and electronic documents face many bewildering options that must be examined to see if it can be applied to their own publishing needs. They need answers about the potential of an SGML-based system that will not be too difficult or expensive to obtain.

1.1. *PURPOSE*

The purpose of this research project is to investigate how data can be captured once and then output in several forms using the Standard Generalized Markup Language. This topic represents the definition of true electronic publishing: a system that has the ability to create, manage, manipulate, integrate text and/or graphical information for subsequent output to a variety of output devices. Publishers interested in finding out if SGML-based publishing will benefit their particular operation will be interested in this

project. People who have an interest in electronic publishing issues will be able to gain an understanding of one of the most important developments in this industry, SGML.

1.2. *PROCEDURE*

This investigation will be done by outlining a detailed model for an SGML-based publishing system (Figure #1) using SGML-based software, and by studying SGML-based publishing systems that are currently being used in commercial publishing. A representative document will be used to illustrate how data can be prepared so that it can be entered into an SGML-based system once for subsequent output in a variety of forms.

This project will be evaluated by its ability to give a clear and complete description of an SGML-based publishing system.

2. REVIEW OF THE LITERATURE

The available information on SGML can be seen as pieces of a complex puzzle. Figuring out how the pieces connect is not easy. An inquiring mind wishing to know what SGML is, and how it should be used, faces an Olympic-sized challenge. The best approach to understanding SGML is to first understand the issues and problems

involved in electronic publishing or computer information systems. After this investigation into the literature it will be possible to understand SGML and how it can be used to help create a productive publishing system. The proceeding is a topical literature review outlining these issues.

2.1. *THE NEW PARADIGM*

There is a new paradigm in the industry regarding document processing and electronic publishing. Instead of dealing with documents based upon physical formats, we can now choose to take a logical approach:

> The variety of electronic representation can be viewed as a continuum from the physical to the logical. At one extreme a simple document image serves as surrogate for the paper version. At the other extreme is an electronic document consisting of descriptive tags and content. This latter form allows many kinds of interaction including editing, full-text retrieval, and re-formatting, i.e., electronic publishing (Handley & Weibel, 1990).

Traditionally, the job of document designing was done by editors and graphic designers after the content was created. The author would write the content and the editor would act as the liaison between the author and the graphic designer. Due to the increasing amount of new technologies that require documentation, there has been a significant increase in the amount of published information. The content of documents also has become more complex. Thus publishing has become more complex when attempting to

use traditional document production methods. Publishers began to employ a logical approach to document designing by separating the form from the content before the data is processed for final output. Now many publishers view documents as having two dimensions: logical and physical.

2.1.1. *The Logical Approach*

The logical approach to document processing frees the document analyst/designer from having to worry about how the document will look. This person is specifically interested in the content structure. Where a graphic designer must consider the length of the document, the size of the image area, and other physical constraints; the document analyst must consider the ability of the information to conform to the specific structure of a document type. By viewing a document from a logical perspective, a document analyst analyzes and identifies what document is by its type: a textbook, technical manual, or a novel, and then uses descriptive markup to prepare the data.

Markup is the term used to describe codes added to electronically prepared text to define the structure of the text or the format in which it is to appear (Bryan, p.5). A descriptive markup language provides a means for coding information

based upon the structure of the content. This method uses nouns and adjectives to describe the different elements within a document (Davis & Waldt, 1987).

2.1.2. *The Procedural Approach*

Publishers who deal with ephemeral information, or information that does not require constant editing or updating after it is published, do not need to follow the logical paradigm. If a publishing operation meets the criteria above and also has a document production environment that produces many graphical images and merges them with text on a desktop or workstation system, then the traditional procedural methods of document production are best. Preparing the data to be published in this environment is different. The methods most often used are very production oriented. They facilitate immediate publishing needs, which are to get the information out as soon as possible, by any means necessary. The editor in this environment may use two types of markup: proofreader's marks to convey instructions to the author, and procedural copy markup to convey instructions to the artist. A procedural markup language is used to encode text with instructions to the output device on how the document should appear (Figure #2).

The procedural method views a document as being the same as its physical format. The format and content are not separate (Barkes, 1991). In many publishing operations the rules for creating the structure of the content by the editor and artist are procedural; based upon the final physical appearance. The primary considerations during production are based upon the desired format. The content of the document must be made to fit this format, which is usually the secondary consideration during production.

The problem with this method is that in each phase, a document is tied to sets of physical formatting requirements for specific output types. If there is a need to re-use the same content of a document, and produce output in a different form, then the information must be re-entered into an application that produces the desired output. This has been a productivity problem with some publishing systems because documents, or output types, are stored at this level of the production cycle with the formatting instructions intertwined with the content. It can be difficult to edit documents at this stage without interfering with the final physical appearance. It is also difficult to exchange the same information for different output types, a flexibility that is needed in this age of multimedia (Barkes, 1991).

2.2. *WHAT IS SGML?*

As a culture, we have only recently begun exploring electronic documents; we have no collective notion yet of how they ought to look or behave (Walter, 1992). The objective of SGML is to free markup from the dependencies of the processing to be performed, and to regularize its use for modern text processing applications (Goldfarb, p.125).

ISO 8879 is the formal specification created and published by the International Standards Organization for the use of the Standard Generalized Markup Language. SGML facilitates the logical approach to publishing information. SGML is a meta-language that creates descriptive markup languages that will prepare data by providing a standard way of coding structured information. The data that is marked up using SGML is independent of the input and output device(s) being used. Data that is prepared with SGML reduces the need to recode the data at every production step. This gives more control over the use of information.

SGML looks at the role of the different elements of a document type and identifies them with tags (Figure #3). The language can be used to define or identify other types of elements besides text that may be a part of a document. SGML identifies all of these various elements, their functions and relationships to each other and to the document

as a whole. This means that it would be possible to treat every element of the content as a separate unit, and structure the information of a document in almost any way. SGML is not simply a set of markup tags. It provides the language, or more specifically the syntax, for creating the language of the tags (Alexander & Walter, 1990). This is called a document type definition (DTD).

> A DTD is an ASCII text file that describes the structure of a class of documents, what elements they must or may contain and the order and frequency of their occurrence in a document. A DTD is often thought of as a structural outline of a document, but it may also be thought of as a language, a set of tokens (tags), and a set of rules that govern how those tokens may be used together in a document (Alexander & Walter, 1990).

2.2.1. *Document Interchange Standard*

With SGML, documents can preserve their inherent information regardless of what changes might be made in their presentation (Castro, 1990). SGML was designed to be a standard for document interchange. Document interchange is the moving of data across different systems. Today, publishing systems are often comprised of several different types of platforms and operating systems. The users are members of workgroups that share data through networks, or database management systems. Or in other cases data is received from different sources in various formats, and integrated within a publishing system. These heterogeneous environments make the production process difficult.

The conversion process that the data types must go through from system to system can be time consuming, or in some cases, impossible.

> The key to the successful interchange of data within an enterprise or between larger entities is standards. Standards are the key to allowing diverse systems to work together to provide a single corporate information flow. Standards allow for the design of systems that evolve without the loss of knowledge and information (Boyd, 1989).

Because of these difficulties electronic publishing systems require document interchange standards that can provide transparent movement of information between all types of systems and platforms without diminishing the integrity of the content and structure of a document. SGML meets these requirements by separating the form from the content of documents.

2.2.2. Meta-Language

SGML is a meta-language notation that can be used to create languages that describe documents, databases, and other structured information (Alexander & Walter, 1990). The language is an internationally recognized standard used as the basis for several other related document processing standards such as Document Style Semantics and Specification Language (DSSSL), Standard Page Description Language (SPDL), Standard Music Description Language (SMDL), and Hypermedia/Time-based Subset (Hytime).

2.2.3. *Database Creation Tool*

Using a database to help publish information is not a new process. With recent developments in network configurations it is possible to give database systems a larger role in publishing data. SGML can be used to create a structured document database. The concepts that form the foundation of SGML are derived from such traditional database concepts as content encoding, system and application independence, and rigorous definition of information (Gangemi, 1987). A structured database includes fields of information and their definitions. A document type definition (DTD) can be used to express the structure of a proposed system.

A DTD can be presented in written form which makes it easier to communicate the proposed design structure and other characteristics of a database before extensive and costly programming begins (Gangemi, 1987). Because DTD's created with SGML are device-independent, databases do not have to be created based upon a system that already exists. This is an advantage because a database that is designed with the help of SGML can grow with an organization, and the users of the system do not have to be concerned about the information contained in the database being made obsolete by future developments.

3. SGML-BASED PUBLISHING SYSTEM

This outline of an SGML-based publishing system corresponds to the model illustrated in Figure #1.

The task of analyzing documents and creating an SGML document type definition is the first step in using an SGML-based publishing system. Context-sensitive editors help writers create documents that conform to a DTD. The tagged document is then compiled and validated through an SGML parser. Most SGML software application packages have the ability to create and interactively parse documents.

3.1. DOCUMENT PREPARATION

An SGML-based publishing system: receives input from a variety of sources in many different forms; enters the data into a database as SGML structured data; validates the data so that it is correctly organized according to a DTD; and translates the SGML tagged data for formatted output. An SGML-based publishing system consists of: workstations linked through a database management system, or to each other through

a network/server system; SGML context-sensitive editors; conforming SGML/ISO 8879 validating parsers; and an SGML/formatter interface.

Before data can be entered into an SGML-based system the information must be tagged. How the document is tagged is based upon the particular document type definition that will be used. It is the job of the document analyst to create DTDs. Creating a document type definition involves analyzing a representative document type because one DTD will be used for a whole class of documents.

The analysis involves creating a structured model of the information. The model is examined with the help of the authors and editors of a particular document in order to determine what elements will be included in the DTD. The model becomes the basis for the DTD, there is an identifiable correlation between the two.

3.1.1. *Document Types and Structures*

A document is one possible representation of a body of information in a structured or hierarchical form. It can, for instance, be an entire database of information. Most document types (novel, manual, electronic book, etc.) have a general structure that a reader can identify because of recognizable text elements such as: parts, chapters, sections, and

paragraphs that are organized in a hierarchical structure. Different hierarchical models can be used to convey the relationships between the text elements of various document types.

Document model structures are used to describe the representation of a document. The "actual" document is what the readers see and use. The visual structure of a document should reflect the logical structure of the text that the document realizes (Southall, 1989). Structure editing is based on a single structure or set of rules for a class of documents. This structure is formally described and allows a consistent structure of the documents to be maintained according to certain data models such as a tree structure (Vercoustre, 1990).

A document that can conform to a tree-structure has these characteristics:

- *Hierarchy*. Ranks can be assigned to the elements of the structure.
- *Containment*. Higher-ranking elements contain lower ranking elements.
- *Sequence*. Elements of equal rank follow one another in the text.

These relationships can be visually expressed by a model (Figure #5) in order to fully realize the document structure (Southall, 1989).

The tree-structure is the visual representation of the elements of a document. Making a model of a document can clarify many of the complexities of the relationships between the elements of a structured document. Figure #5 is an example of a tree-structure model. The complexities of the relationships between the various elements are identified. Many of the components of the elements can be repeated in different places.

A structure model can convey information about the document type. Different document types vary in the number of elements and the complex links between the elements. For example: the document characteristics of a textbook are more numerous than those of a novel. A textbook contains lists, charts, references, an index, and maybe footnotes. A novel usually does not contain these elements. A textbook or a technical manual can have long and painfully complex tree-structures that are difficult to follow.

Creating a model helps the document analyst know what will be the best way to process the information without threatening any important document characteristics. Some documents cannot conform to a logical structure very easily or they cannot conform at all. Analyzing information by using a tree model can provide the designer with valuable information about how to prepare the data and apply the descriptive markup before entering it into the publishing system.

These advantages and disadvantages of using the logical approach to creating documents are based upon those identified by Nanard (1987) and Furuta (1988).

Advantages:

- Identifying and creating structure increases the flexibility and editability of a document.
- Authors and editors are relieved of any formatting of a document.
- Maintaining the consistency of a document type that has an identified structure is easier than one without an identified structure.
- Links between various elements can be maintained more easily.
- More control can be administered during the processing of documents.

Disadvantages:

- Hierarchically structured documents do not correspond to the authoring of a document. Authors do not write in this way so trying to adhere to a structure can be restrictive.
- Document models are often too rigid. It is difficult to switch from model to model or use different contents for the same models.

3.1.2. *Document Analysis*

The best way to understand this process is by looking at an example. A very simple DTD will be created based upon a document type of which a hypothetical example is a manual entitled *The Handbook of Print Production for Desktop Publishers.* It is to be a reference guide for desktop publishers to use for print production. This document is also meant to be an "active publication" meaning that the information will be constantly updated for the intended users. What form that it will eventually be output in has not been determined at this time. The proposed outline is contained in Figure #4.

It is necessary for the document analyst to work with the authors and the editors in the painstaking process of creating a tree model. The document analyst can also find out things about the information from the authors such as the intended audience. The intended use of the information by the editors and the end users is perhaps the most important piece of knowledge that the document analyst needs to know. The intended use is important because the document analyst must know how flexible the DTD needs to be. Here are some possible results of the document analysis of *The Handbook of Print Production for Desktop Publishers*:

1. *Can the the document conform to a hierarchical structure?*
   Yes. The content can fit the general structure of a textbook.

2. *Who is the Audience?*
   Anyone who uses a personal computer to assemble and prepare information that will be output in some published form. This will include a majority of the people in the graphic arts and publishing industries. The level of information that is required by the audience will vary based upon the segment. Artists may need to know more than a color separator. A publishing manager may only need to have introductory information. Individuals may not need all of the information that is outlined.

3. *What are the flexibility requirements?*
   Based on the diversity of needs of the audience, the DTD must have a high degree of flexibility in relation to information retrieval. Because the document will be an "active publication", It is necessary to to have flexibility for the maximum amount of editability as well. With the possible exception of chapters 2, 3, 4, and 8 in the content outline, the document will constantly be updated to include new information.

Based upon these answers, a tree-model structure of the document type can be created (Figure #5). This structure represents all of the elements identified by the document analysis, that can be a part of *The Handbook of Print Production for Desktop Publishers*.

3.1.3. *Element List*

An element description list (Figure #6) will provide the document analyst with a clearer description about the relationships and element structure of a document type. It identifies all necessary and optional elements, and the occurrence

and sequence for each one. The creation of an element description list is not a necessary step. More experienced document analysts can be successful without using one.

Element lists are useful aids to the document analyst when creating a DTD. An element description list is a less formal DTD. Document type definitions are not as redundant, and require a specific written format. The lists show the elements, and embedded sub-elements that are defined as a model group. Like the elements themselves, the model group consists of one or more element names that are connected by SGML group sequence connectors (Figure #6) which also occur in the DTD (Bryan, 1988, p.110).

As with a DTD, the element list also qualifies the use of each embedded sub-element. These are called occurrence indicators (Figure #6). Connectors and indicators can be used together. However, when determining the effect of occurrence indicators and group connectors it is important to realize that occurrence indicators have a higher precedence than connectors (Bryan, p.111). The occurrence indicators and operators represent all of the document possibilities. They determine what can and cannot be a part of a document that conforms to a particular DTD. Element lists are a good thing to give to authors and editors to check off on before the actual DTD is written.

3.1.4. *Creating A Document Type Definition*

Once a document type is identified, figuring out the possible elements that make up the document type and which should be included in the DTD is not trivial. Creating a DTD involves taking the information accumulated from the document analysis and then applying the rules for constructing an SGML DTD as mandated by ISO 8879. The DTD in this example (Fig #7) contains shortened versions of the names of the elements in the tree-model and element list. The names are not minimized to the extent that they couldn't be understood by a human.

The DTD that was created using the hypothetical manual *The Handbook of Print Production for Desktop Publishers* as a representative document type is a simple DTD, and does not contain all of the features that are allowed by ISO 8879. A DTD that is successfully parsed should not be confused with a good DTD. A DTD that is well constructed and successfully parsed must also have the ability to provide a way for the

intended uses of the information to be realized. If the DTD is not flexible enough to produce a variety of document instances, then it is not a very good DTD.

3.2. *INPUT*

Data can be input into the system by either keying text in through a context-sensitive editor, by inserting tags into an ASCII file that was either keyed or scanned, or by translating an incompatible file into SGML (autotagging). Manual re-keying or optically scanning text from hard copy is not uncommon. This introduces a large amount of human and machine error. A lot of valuable time must be spent proofing the results. If a publisher were to provide the means for authors to submit text as SGML tagged files via an easy to use, WYSIWYG, context-sensitive editor, then one step in the input process could be eliminated, thus increasing productivity.

When first starting up an SGML-based system it may be necessary to translate information that exists in another file format or language into SGML. There are an increasing number of these sophisticated software applications that can do this translation process.

3.2.1. *Context-Sensitive Editor*

The best way to create an SGML tagged document is to use an SGML context-sensitive editor. An SGML context-sensitive editor is a software application that allows the user to create documents that conform to a DTD. They are similar to word processing applications except that the emphasis in creating documents is placed upon the content structure rather than physical appearance. The application creates the tags based upon the DTD chosen by the user, and then inserts the tags. Many of these software applications can validate documents as they are created. As well as being used to create documents, SGML editors can be used to insert tags into a document that was not originally created with an SGML editor. This is done by flowing in the text from a document created by an ASCII editor into an SGML editor.

3.2.2. *ASCII Editor*

Most computer platforms have the ability to create an ASCII file. The word processing software applications that run on various operating systems can output an ASCII file. Creating SGML documents with an ASCII editor is very tedious because the tags are inserted by the user character-by-character. This increases the possibility of errors to occur when the document is parsed. Using an ASCII editor can be a complex undertak-

ing because the user needs to understand the structure of a DTD in order to create an SGML tagged document. If a DTD is extremely complex, then it would be virtually impossible to use an ASCII editor to structure and tag documents.

The only reason to use an ASCII editor to create SGML tagged documents is because they are widely available and used. Using an ASCII editor instead of an SGML context-sensitive editor to structure and tag documents, is similar to writing a postscript program to layout a page instead of using a page layout software application.

3.2.3. *Autotagging*

Using a translation software package is the best way to convert a document that was not prepared through an SGML editor. This procedure is called autotagging. Most of these conversion software applications do what is called a "context translation". The file that was originally created using a word processing application is scanned for patterns that represent the elements that correspond to a DTD. The application then generates and inserts the tags into the document based upon the first assessment. The newly tagged document is then passed through a parser that is a part of the conversion program. Not all documents that were created with a word processor can be converted in this way. The structure of incompatible documents must correspond to a DTD

because the autotagging that takes place is based upon the documents ability to conform to a structure. If a document created in a word processing application does not conform to a structure, then it must be edited and the conversion program must be employed with a certain amount of human intervention. So in many cases autotagging is not completely automatic.

3.2.4. *Storage*

The document database is at the center of an SGML-based publishing system (Figure#8). As with many large enterprises a database management system is used to help facilitate the productivity of business operations. By integrating management operations with publishing operations into one database system, the processof publishing information is helped tremendously. SGML can be used to create the structure of the integrated database system which will provide security for private files, manage protocol, and track the progress of documents.

An integrated database management system can be used to manage all of the steps in the creation of SGML tagged documents. It can be used to direct all operations: from the collaboration of the document analysis and the authoring of documents, to the final preparation for output.

3.3. *VALIDATION*

An SGML parser is a software program used to validate DTD's and document instances (SGML document). SGML parsers can help document analysts create DTDs that conform to ISO 8879. The parser works by examining the logical elements of a document instance and makes certain that the elements correspond to the associated DTD. They create a report on any design flaws, or the misuse of any features of SGML that are considered errors by ISO 8879. There are many parsers available that can be used to work interactively with an SGML editor to create and validate documents and DTDs. Interactive parsers help create a more productive input process.

SGML parsers also gives information to the formatter which prepares the SGML documents for output. How this is done is an implementation decision made by the publishing system designer.

3.4. *CREATING OUTPUT*

A DTD is a list of ingredients for one type of a document. A formatter can be thought of as something that assembles the various ingredients together to create a document. A formatter is needed because SGML documents do not contain any information about

how they should be processed, this information must be added. A formatter is a program that prepares documents created in SGML to be output by any device. The formatter gathers the data that it is sent from the parser, and converts that information into the format required by the output device. The parser tells the formatter which DTD the document conforms to, and therefore what tags to look for in the conversion process. The parser also supplies the formatter with important structural information about the document that must be maintained.

The conversion is based on pattern matching. An example of how this may work is as follows: The SGML document is scanned to locate the patterns of the tags and find a match to the tags that are contained in a look-up table. The matched tags are replaced by the formatting codes that correspond to a particular type of output devices. The look-up table corresponds to only one document type definition. The table contains tags in one column and corresponding formatting codes in an adjacent column. This means that for every DTD there is a unique look-up table for each output type.

The ways that formatters are built generally depends on the types of output that are required of a publishing system.

3.4.1. *Batch Formatter*

Many publishing operations only produce one type of document and output, such as: printed manuals, journals, or textbooks. A publishing operation that produces one type of output would only use a few DTDs. A batch composition system would be used to create pages for mass reproduction. Documents are retrieved from some type of storage, parsed, formatted, composed and paginated for the output device. A formatter built for this process is virtually hard-wired to a particular DTD which affords no flexibility.

One benefit of building a formatter in this way is that the composition is very fast because there are no intermediate steps in the conversion process. There is no human intervention to get through the various processing steps because there is only one final physical format for all documents. This is appropriate for publishers that produce one type of output such as printed books, and on-demand publishing. However, if a publishing system contained a formatter built in this way and wanted to expand and process documents created from different DTDs, then a new formatter would have to be configured.

3.4.2. *Interactive Formatter*

In Figure #1 the SGML-based publishing model shows a possible publishing system that produces a variety of output with a formatter for every type of output. This section of the model represents a different conversion for each DTD for each type of output, even if the output is similar. For example: Two printed versions of a document having different styles would require unique conversions. As stated previously the conversion is via a look-up table for each DTD for each type of output. In order to have more flexibility and control over the information, it is necessary to create more control between the SGML document and the conversion process. This control is usually in the form of another software program that can be used to give additional formatting information to an SGML document. Instead of hard wiring a DTD to a particular set of formatting instructions, this type of software program allows for the creation of "on-the-fly" look-up tables for SGML documents that go through a formatter. This has proven to be necessary because each instance of a DTD may require special processing that is not covered by the hard-wired look-up table. These software programs enhance the conversion process, and can make a publishing system that produces many different types of output more productive than creating a hard-wired formatter for each type of output.

3.4.3. *DSSSL*

The Document Style Semantic and Specification Language is a draft ISO standard that will allow formatting information to be interchanged between various output systems. The objectives of the DSSSL standard is to provide a formal and precise means of expressing the full range of document production specifications required by the graphic arts industry (Adler, 1989). Instead of using a conversion software application that does "on-the-fly" conversions, DSSSL can be used to attach formatting instructions to SGML documents so that they can still be output by any formatter and output device without altering the results.

In the SGML-based publishing system model in Figure #1, there are types of output that produce the same result: printed pages. There is a different conversion for each of these outputs even though they will all produce printed pages. DSSSL could be used to attach processing information to an SGML document so that it could be formatted and output through a copier, a postscript device or a typesetter and still produce a document that looks the same. DSSSL used in this way would drive the formatter, but it could be used to give information to another language that will format the SGML document, for instance, in the creation of a hypertext or multimedia document. DSSSL would fit well into a publishing system that produces a variety of published products.

3.4.4. *Output Types*

Figure #1 contains the types of output that can be possible for an SGML-based publishing system. They represent the most popular options used by commercial and corporate publishers today. Creating multimedia and hypertext documents involves working with diverse datatypes and formats. People that create these types of documents need to have data sources that can be integrated and output from one source. On-demand and custom publishing products are being produced at a rate as fast as traditionally printed products. The increasing number of publishers that provide on-line information services reflect the desire of consumers to get information as the need for it arises.

As previously stated SGML can be used to create structured databases. One of the ways that SGML is currently being used is in the creation of CD-ROM products. These large databases contained in a small package can be structured using SGML. The value of information stored on a CD-ROM is that it allows the end-user to search through the data in a hypertext manner.

SGML creates a generic source of information that will outlast the output system. As new output systems are developed the same source of SGML tagged data can be used to produce new output types. So when a revolutionary new type of output is made

available, for example voice synthesis, a publishing system that stores its data in SGML will be able to produce a voice synthesis without doing costly conversions.

4. CONCLUSION

An enterprise that publishes a large amount of information that has a long life and must be constantly edited should seriously consider using an SGML-based system. SGML can be used for operations that receive input from many different sources and in different formats.

Publishers have a larger number of publishing products to offer to their customers. How they go about this operation can be very difficult and complex. The most economical and productive way would be to have one generic source of data that can be used by all of the output systems that a publishing operation may have. Having one source of data shared by many connected users, on divergent operating systems, and at different locations through a network database system can increase the productivity of the editorial functions of a publishing operation.

The key to using SGML to output one source of information in a variety of forms is to spend the time and effort to prepare the input data. Taking the logical approach to publishing instead of a procedural approach can open up new possibilities for a

publishing enterprise. The logical approach frees a publishing system from being dependent on any one particular type of output that is produced. A publishing enterprise that wants to be productive and competitive can no longer use a system that is dependent on one type of output. A system of this type cannot be easily adapted for future file formats and output. SGML facilitates the logical approach by preparing data in a way that will allow it to be productively edited, retrieved, and published in a variety of forms.

A practical way to view SGML is from the point of view of the problem to which it is applied (Alexander & Walter, 1990). SGML can be used to implement a new methodology and solve the problems that exist in publishing: heterogeneous environments, data interchange and integration, and producing different types of output from a single source of information. These problems were the force behind the creation of SGML, not to bring about the often predicted paperless society, or to answer all of the problems in electronic publishing. SGML is not a magical solution to integrate all of the wonders of technology that can be a part of an ideal publishing system. It is an option that an organization can choose to help create productive publishing and information systems.

SGML-BASED PUBLISHING SYSTEM

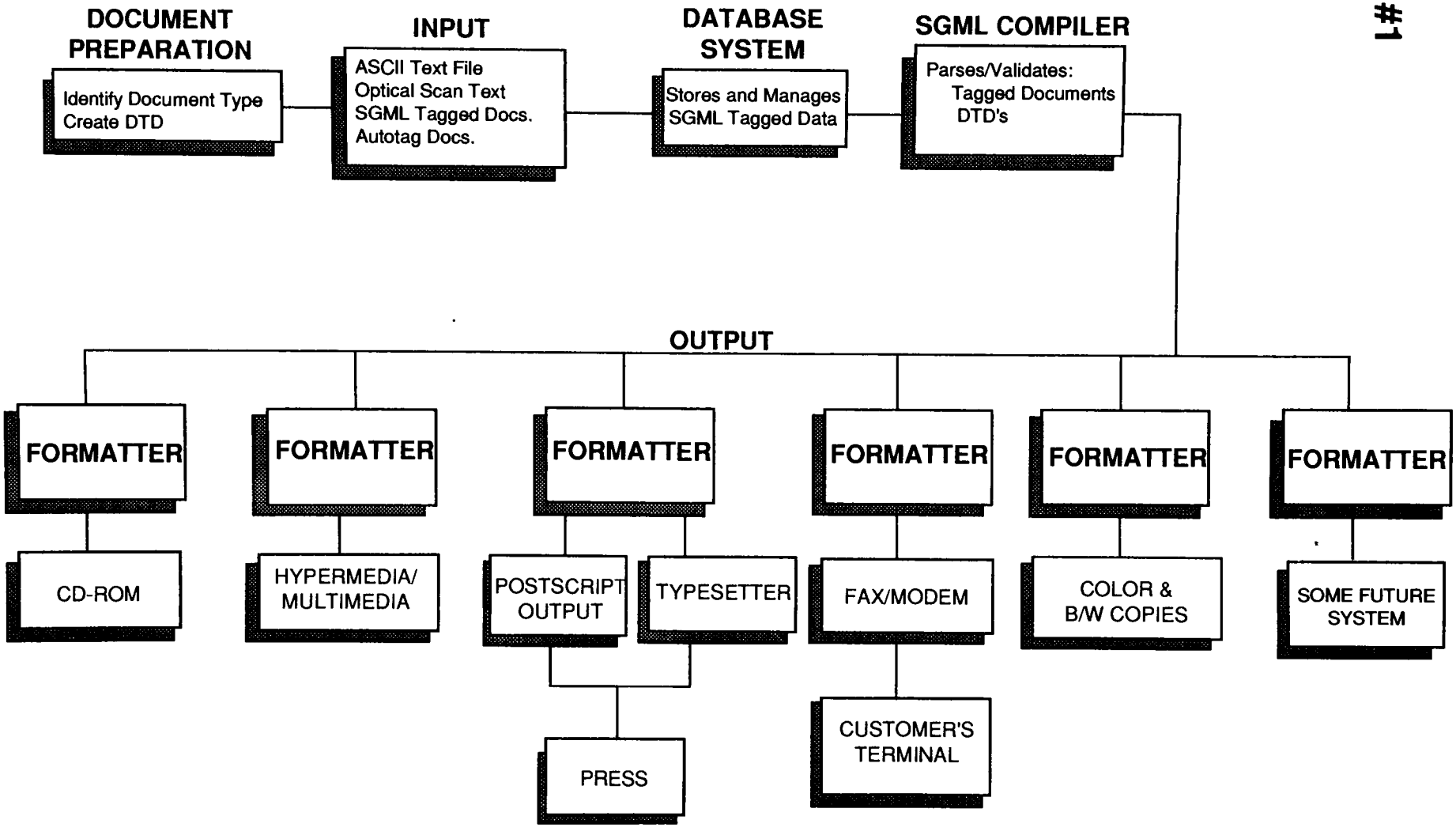


FIGURE #1

FIGURE #2

PROCEDERAL MARKUP

SGML PUBLISHING SYSTEM — 16/18 Palatino Bold Caps

1. INTRODUCTION — 1416 Palatino Bold Caps

The 1990's will be noted as the start of true electronic publishing. The requirements for electronic publishing lie in its definition: a system that has the ability to create, manage, manipulate, integrate text and/or graphical information for subsequent output to a variety of output devices. The ability to encode data so that it can be interchanged across platforms and between applications. The issue of productivity has been the most problematic for electronic publishing systems. The current methods for managing, manipulating, and integrating information

10/12 Palatino Romanupper/lower

- List Item
- List Item
- List Item
- List Item

10/12 Palatino Roman Caps

FIGURE #3

SGML/ DESCRIPTIVE MARKUP

<Chapter Title> **SGML PUBLISHING SYSTEM**

<Section Title> **1. INTRODUCTION**

<Paragraph> The 1990's will be noted as the start of true electronic publishing. The requirements for electronic publishing lie in its definition: a system that has the ability to create, manage, manipulate, integrate text and/or graphical information for subsequent output to a variety of output devices. The ability to encode data so that it can be interchanged across platforms and between applications. The issue of productivity has been the most problematic for electronic publishing systems. The current methods for managing, manipulating, and integrating information

<List Item>
- List Item
- List Item
- List Item
- List Item

FIGURE #4

THE HANDBOOK OF PRINT PRODUCTION FOR DESKTOP PUBLISHERS (OUTLINE)

1. Models for desktop publishing-printing systems
   - A. Postscript
   - B. System Components, including hardware and software
   - C. Network Architectures
   - D. Workflows and material flows

2. What printing process will be used?
   - A. Run length, image quality requirements, sizes, colors
   - B. Lithography, flexography, gravure, screen
   - C. Process specific requirements
   - D. Electronic processes
   - E. Decision tree

3. Planning for the bindery
   - A. Binding methods
   - B. Page sizes
   - C. Signatures
   - D. Sheetfed, half web, full web signature sizes
   - E. Impositions
   - F. Compensation for material behavior

4. Paper and Ink
   - A. Materials used in different processes
   - B. How design must accommodate choice of materials
   - C. Recycling and printing on recycled materials
   - D. Health and safety

5. Film and Plates
   - A. Film requirements for plate making
   - B. Reading orientation
   - C. Tonal orientation
   - D. Spreads, chokes, and trapping
   - E. Stripping requirements
   - F. Imagesetters
   - G. Basic principles of scan-conversion
   - H. Computer-to-plate

6. Type
   - A. Adobe Type 1 format
   - B. TrueType
   - C. Other type formats
   - D. Effects of printing process characteristics on type reproduction

7. Input scanning
   - A. Types of scanners
   - B. Linear spatial resolution
   - C. Pixel depth and noise (tonal resolution)
   - D. Scanning resolution requirements for given output res.
   - E. Ideal copy for scanning
   - F. Raster to vector conversion for line art
   - G. Data compression

8. Tone and color reproduction
   - A. Synthetic color
   - B Photographic color
   - C. Spot and process color
   - D. Color gamuts for printing processes
   - E. RGB to CMYK
   - F. HSB color model
   - G. Pantone color
   - H. Focoltone color
   - I. Halftone reproduction
   - J. Color separations
   - K. Screening technologies
   - L. DPI of imagesetter vs. LPI of halftone
   - M. Proofing technologies

9. The right tools for the job
   - A. Print quality capabilities of processes
   - B. Register, image structure, tone & color res. capabilities
   - C. Imagesetter requirements determined by print expectations
   - D. Imagesetter types
   - E. Page layout programs
   - F. Hardware/software platforms
   - G. Storage options and requirements
   - H. Network structure requirements

10. Working with a service bureau
    - A. File structures and interchange formats
    - B. Production processes that work
    - C. Standard procedures
    - D. Checklists

11. Training issues
    - A. Teaching desktop publishers about printing
    - B. Information resources
    - C. Methods
    - D. Communication
    - E. Software-based self-training

Annotated index

Appendices
1. Guide to file formats

* FIGURE #5

*Figure based on textbook tree-structure in *SGML: An Author's Guide to the Standard Generalized Markup Language*

FIGURE #5 CON'T

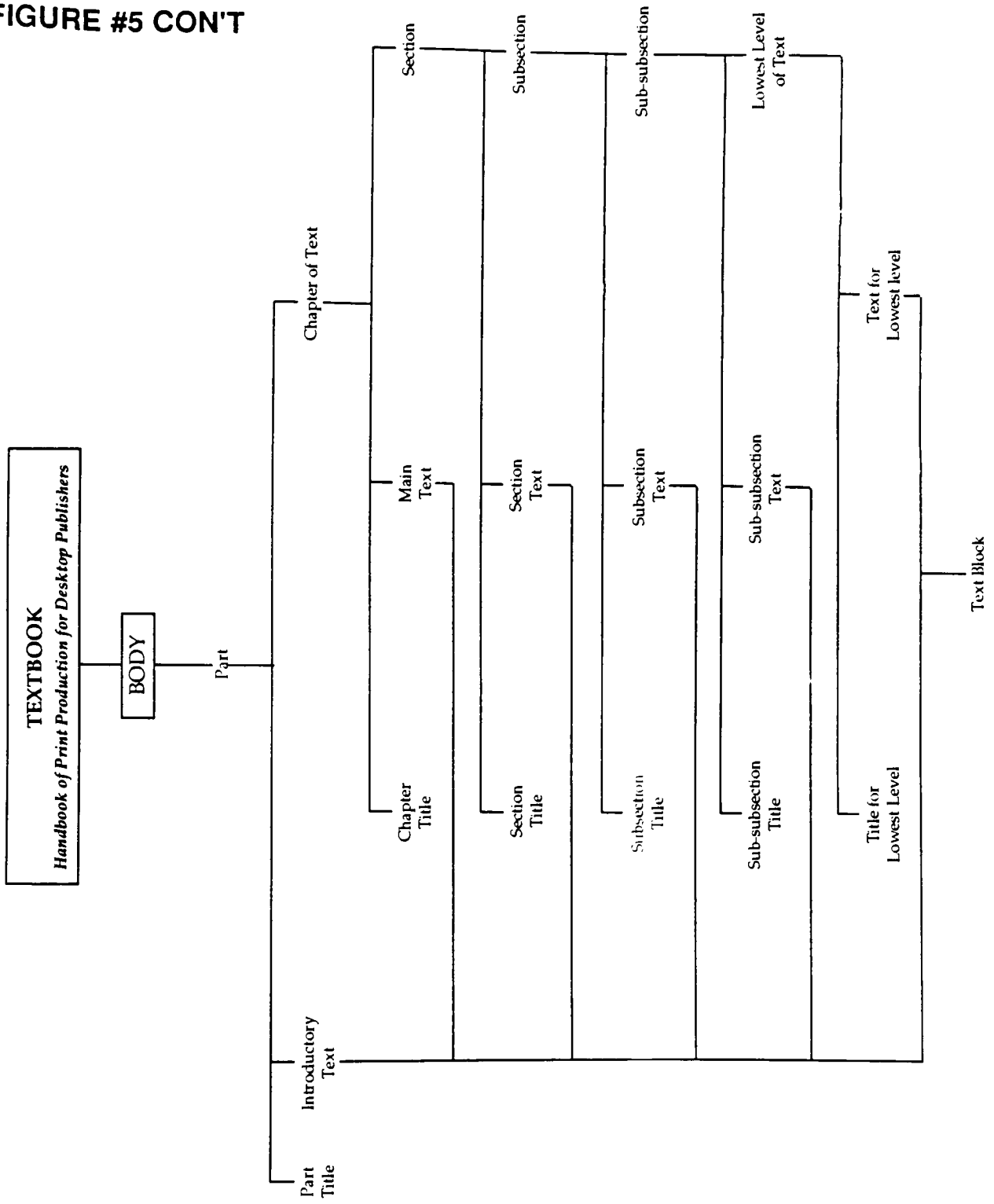


*Figure based on textbook tree-structure in *SGML: An Author's Guide to the Standard Generalized Markup Language*

FIGURE #5 CON'T

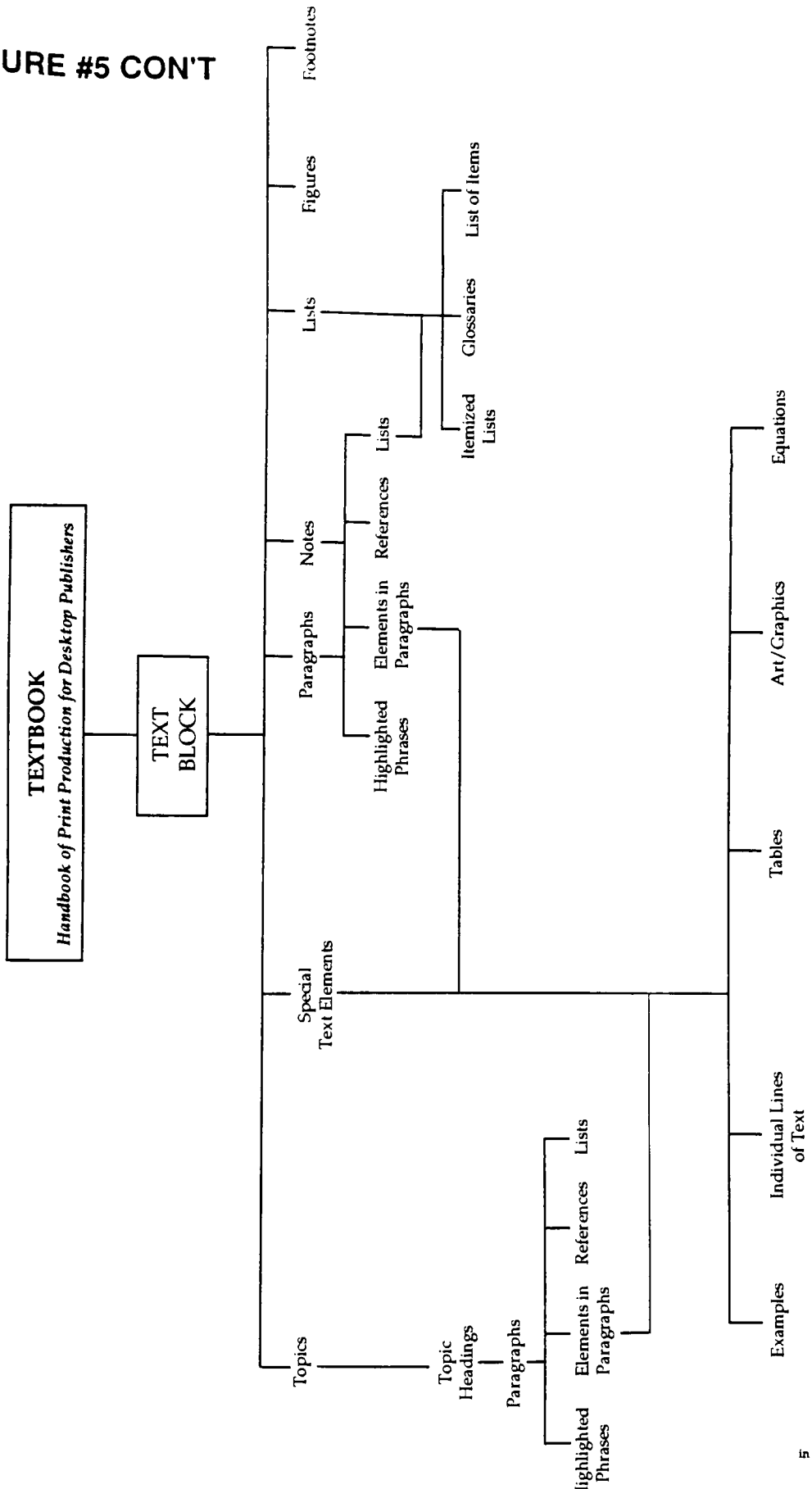


*Figure based on textbook tree-structure in *SGML: An Author's Guide to the Standard Generalized Markup Language*

FIGURE #5 CON'T

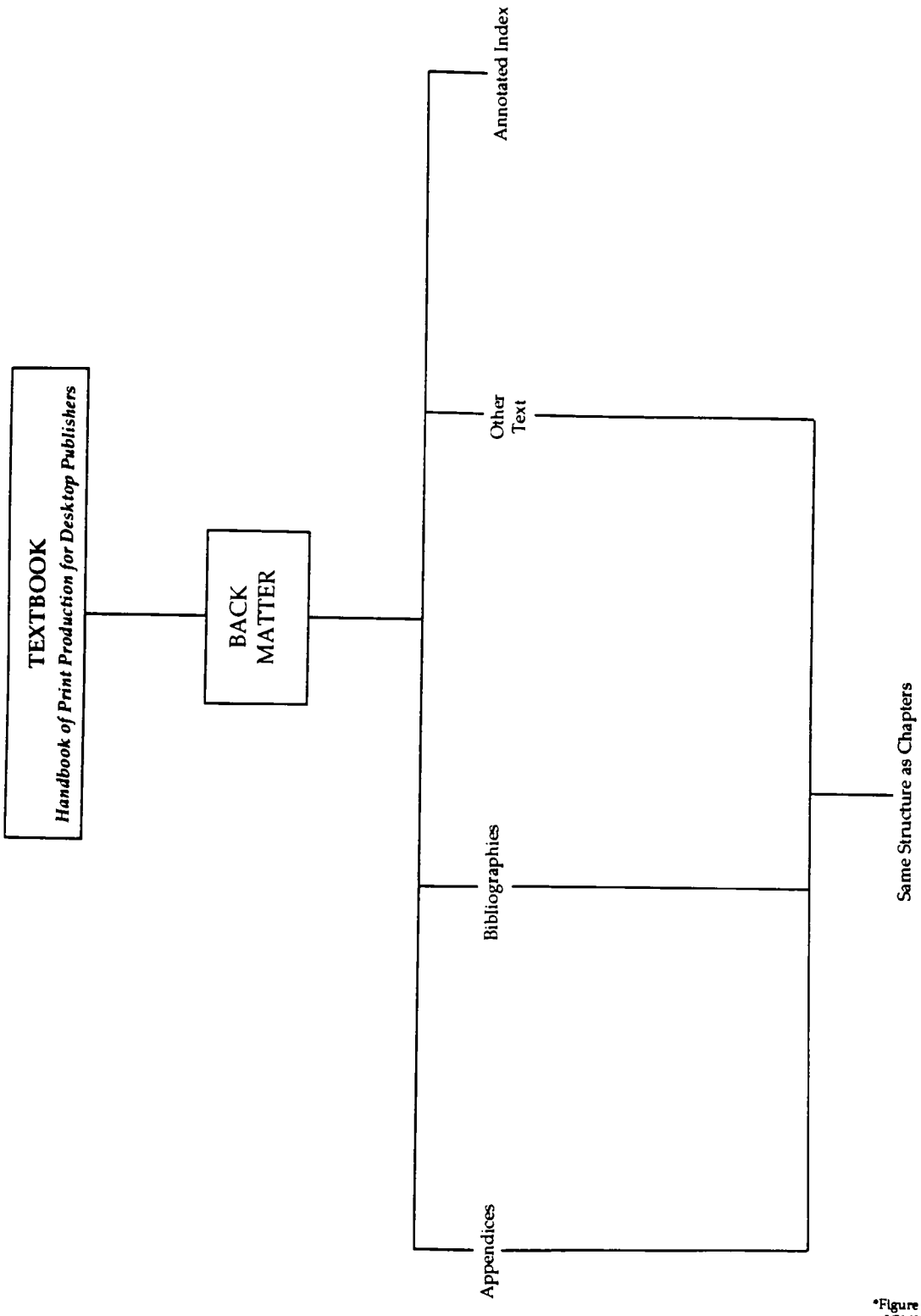


*Figure based on textbook tree-structure in *SGML: An Author's Guide to the Standard Generalized Markup Language*

FIGURE #6

ELEMENT LIST

SGML Group Sequence Connectors:

,	All must occur, in the order specified.
&	All must occur, in any order.
|	One and only one must occur.

Occurrence Indicators:

+	One or more can occur, but at least one must occur.
?	Zero or one can occur.
*	Zero or many may occur.

Textbook=Front matter, body, back matter
Front Matter=Half title page, title page, title verso, table of contents, preliminary text, illustration and photo list?
Half Title Page=Title&authors&
Title=#PCDATA
Author=#PCDATA
Title Page=title&authors&edition&editor&publisher
Authors=#PCDATA
Edition=#PCDATA
Editor=#PCDATA
Publisher=#PCDATA
Title Verso=Publisher details, copyright details
Publisher Details=Name, location
Name=#PCDATA
Location=#PCDATA
Copyright Details=Date of publication, copyright holder, copyright declaration, colophon?, cataloging details, ISBN #
Date of Publication=#PCDATA
Copyright Holder=#PCDATA
Copyright Declaration=#PCDATA
Colophon=Printing details, other contributors
Printing Details=Name, location
Other Contributors=typesetter, designer, illustrator, researchers
Typesetter=#PCDATA
Designer=#PCDATA
Illustrator=#PCDATA
Researchers=#PCDATA
Cataloging Details=Library of Congress #
Library of Congress=#PCDATA
ISBN #=#PCDATA
Table of Contents=#PCDATA
Preliminary Text=Forward?, Preface?, Introduction, Acknowledgments?
Forward=#PCDATA
Preface=#PCDATA

FIGURE #6 continued

Introduction=#PCDATA
Acknowledgments=#PCDATA
Illustration and Photo List=#PCDATA

Body=Part*, chapter of text +
Part=Part title, introductory text
Part Title=#PCDATA
Introductory Text=Topics*, (paragraph+| notes*), lists*, figures*, footnotes*
Topics=Topic headings
Topic Headings=Paragraph+
Paragraph=#PCDATA|Highlighted phrases*, elements in paragraphs*|references| lists*
Highlighted Phrases=#PCDATA
Elements in Paragraphs=Examples*|individual lines of text*|tables*| graphics*| equations*
Examples=#PCDATA
Individual Lines of Text=#PCDATA
Tables=#PCDATA
Graphics=EMPTY
Equations=#PCDATA
Notes=#PCDATA
Lists=itemized lists*| glossaries*| list of items*
Figures=EMPTY
Footnotes=#PCDATA
Chapter of Text=Chapter title, main text, section+
Chapter Title=#PCDATA
Main Text=Topics*, special text elements*, (paragraph+| notes*), lists*, figures*, footnotes*
Section=Section title, section text, subsection
Section Title=#PCDATA
Section Text=Topics*, special text elements*, (paragraph+| notes*), lists*, figures*, footnotes*
Subsection=Subsection title*, subsection text*, sub-subsection*
Subsection Title=#PCDATA
Subsection Text=Topics*, special text elements*, (paragraph+| notes*), lists*, figures*, footnotes*
Sub-subsection=Title for lowest level*, text for lowest level*
Title for Lowest Level=#PCDATA
Text for Lowest Level=Topics*, special text elements*, (paragraph+| notes*), lists*, figures*, footnotes*

Back Matter=Appendices*, bibliographies+, index?
Appendices=Topics*, special text elements*, (paragraph+| notes*), lists*, figures*, footnotes*
Bibliographies=#PCDATA|Highlighted phrases*, elements in paragraphs*|references| lists*
Annotated Index=#PCDATA|Highlighted phrases*, elements in paragraphs*|references| lists*

FIGURE #7

DOCUMENT TYPE DEFINITION

```
<!DOCTYPE textbook SYSTEM 'dtd of The Handbook of Print Production
For Desktop Publishers'>
<!ELEMENT textbook                    -- (front, body, back)>
<!ELEMENT front                       -- (htitlep, titlep, titlev, toc, prelimtxt,
                                         illusli?, photoli?>
<!ELEMENT htitlep                     -- (title+ & authors?)>
<!ELEMENT title                       -- (#PCDATA)>
<!ELEMENT authors                     -- (#PCDATA)>
<!ELEMENT titlep                      -- (title & authors & edition & editor* &
                                         pub)>
<!ELEMENT edition                     -- (#PCDATA)>
<!ELEMENT editor                      -- (#PCDATA)>
<!ELEMENT pub                         -- (pubdet, copydet)>
<!ELEMENT pubdet                      -- (name+, location+)>
<!ELEMENT name                        -- (#PCDATA)>
<!ELEMENT location                    -- (#PCDATA)>
<!ELEMENT copydet                     -- (date, holder, decl, colophon?,
                                         catalogdet, ISBN)>
<!ELEMENT date                        -- (#PCDATA)>
<!ELEMENT holder                      -- (#PCDATA)>
<!ELEMENT decl                        -- (#PCDATA)>
<!ELEMENT colophon                    -- (printdet, othercon?)>
<!ELEMENT printdet                    -- (name, location)>
<!ELEMENT othercon                    -- (typesetter & designer & illustrator &
                                         researcher*)>
<!ELEMENT typesetter                        -- (#PCDATA)>
<!ELEMENT designer                    -- (#PCDATA)>
<!ELEMENT illustrator                 -- (#PCDATA)>
<!ELEMENT researcher                  -- (#PCDATA)>
<!ELEMENT catalogdet                  -- (loc)>
<!ELEMENT loc                         -- (#PCDATA)>
<!ELEMENT ISBN                        -- (#PCDATA)>
<!ELEMENT toc                         -- (#PCDATA)>
<!ELEMENT prelimtxt                   -- (forward?, preface?, introduc,
                                         acknowledg?)>
<!ELEMENT forward                     -- (#PCDATA)>
<!ELEMENT preface                     -- (#PCDATA)>
<!ELEMENT introduc                    -- (#PCDATA)>
<!ELEMENT acknowledg                  -- (#PCDATA)>
<!ELEMENT illusli                     -- (#PCDATA)>
<!ELEMENT photoli                     -- (#PCDATA)>
<!ELEMENT body                        -- (part*, chapte*)>
<!ELEMENT part                        -- (partitle, introtxt)>
<!ELEMENT partitle                    -- (#PCDATA)>
<!ELEMENT (introtxt | text | sectxt | subtxt | subsubtxt) -- (topics*, para+ | (notes, list,
                                                            fig, fnotes)*>
<!ELEMENT topics                      -- (topichead*, para+)>
```

FIGURE #7 continued

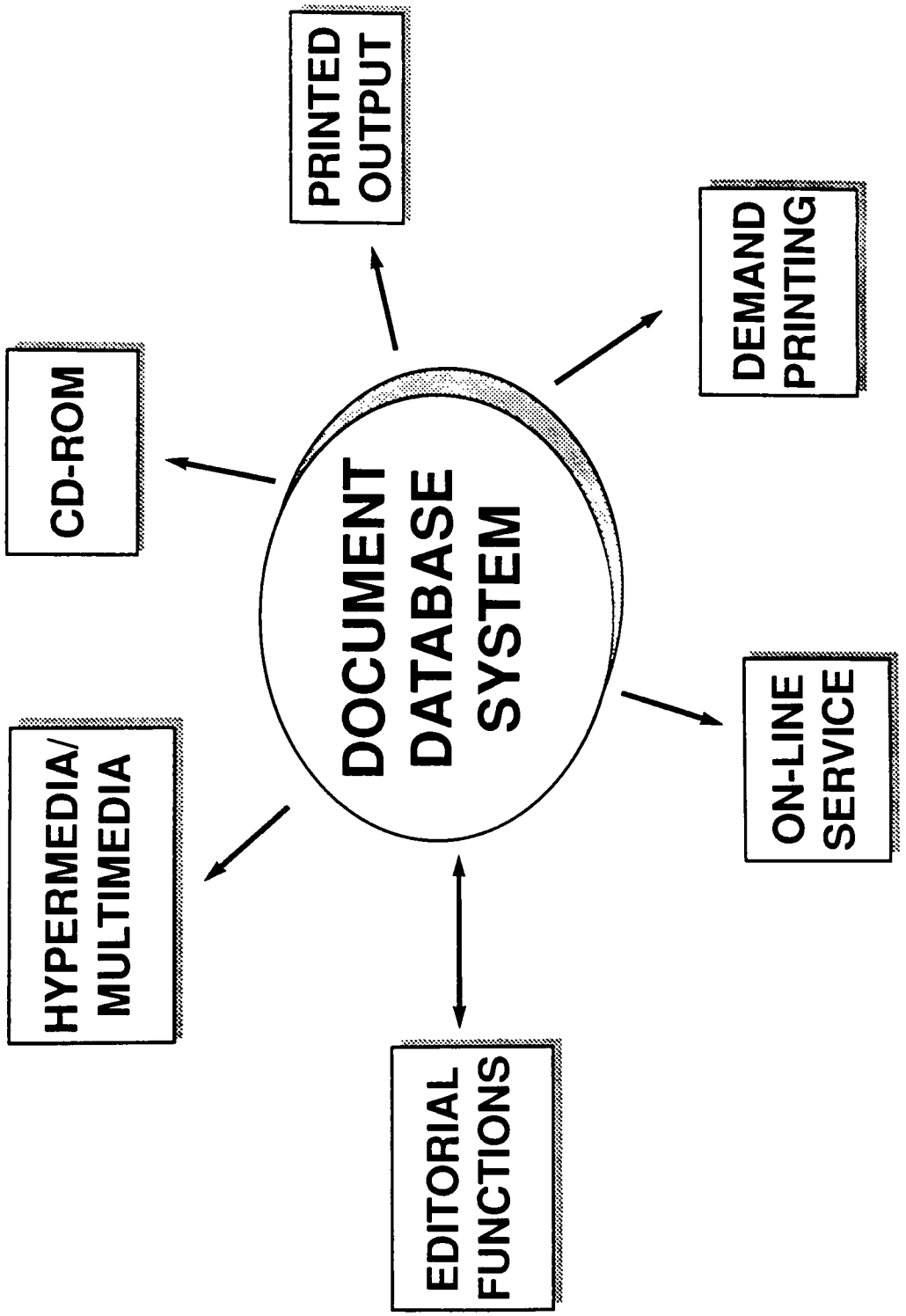
```
<!ELEMENT topichead       -- (#PCDATA)>
<!ELEMENT para            -- (#PCDATA | (phrases, elements,
                              quotes)* | ref* | list*)>

<!ELEMENT phrases         -- (#PCDATA)>
<!ELEMENT elements        -- (examples | txtlines | graphic
                              |equations)* | table?)>

<!ELEMENT examples        -- (#PCDATA)>
<!ELEMENT txtlines        -- (#PCDATA)>
<!ELEMENT graphic         -- EMPYT)>
<!ELEMENT equations       -- (#PCDATA)>
<!ELEMENT table           -- EMPTY)>
<!ELEMENT quotes          -- (#PCDATA)>
<!ELEMENT ref             -- (#PCDATA)>
<!ELEMENT list            -- (itemli | glossary | listi)*)>
<!ELEMENT itemli          -- (#PCDATA)>
<!ELEMENT glossary        -- (#PCDATA & phrases)>
<!ELEMENT listi           -- (#PCDATA)>
<!ELEMENT notes           -- (#PCDATA)>
<!ELEMENT fig             -- EMPTY)>
<!ELEMENT fnotes          -- (#PCDATA)>
<!ELEMENT chapter         -- (chaptitle, text , section)+>
<!ELEMENT chaptitle       -- (#PCDATA)>
<!ELEMENT section         -- (sectitle+, sectxt+, subsec*)>
<!ELEMENT subsec          -- (subtitle+, subtxt+, subsubsect*)>
<!ELEMENT subtitle        -- (#PCDATA)>
<!ELEMENT subsubsec       -- (subsubtitle+, subsubtxt+)>
<!ELEMENT subsubtitle     -- (#PCDATA)>
<!ELEMENT backmatter      -- (appendices*, bibliography+, index?)>
<!ELEMENT appendices      -- (topics*, para+ | (notes, list, fig,
                              fnotes)*>

<!ELEMENT bibliography    -- (para+)>
<!ELEMENT index           -- (list | annotated)>
<!ELEMENT annotated       -- (#PCDATA)>
```

FIGURE #8


PRINTED OUTPUT
DEMAND PRINTING
CD-ROM
DOCUMENT DATABASE SYSTEM
ON-LINE SERVICE
HYPERMEDIA/ MULTIMEDIA
EDITORIAL FUNCTIONS

BIBLIOGRAPHY

Adler, Sharon C., (1989). DSSSL: Document style semantics and specification languange. *TAG The SGML Newsletter, 8*, pp. 8-9.

Alexander, G. & Walter, M., (1990). A fresh look at SGML: the conventional wisdom changes. *The Seybold Report on Publishing Systems, 20*(7), pp. 3-16.

Barkes, Kevin G., (1991). Substance over form: breaking down the barriers between the printed page and electronic media. *DEC Professional, 10*(10,) pp. 34-38.

Boyd, Barbara, (1989, July). DEC's technical authoring environment. *TAG The SGML Newsletter, 10*, pp. 5-7.

Bryan, Martin, (1988). *SGML: An Authors Guide to the Standard Generalized Markup Language*. Cheltenham, UK: Addison Wesley Publishing Co., Inc.

Castro, Christine, (1991, April). The recipe for document interchange. *Computer Publishing Magazine*, pp. 26-31.

Cruz, G.C. & Judd, T.H., (1990). The role of a descriptive markup language in the creation of interactive multimedia documents for customized electronic delivery. *EP90:Proceedings of the International Conference on Electronic Publishing*, R. Furuta (Ed.) V110. New York: Cambridge University Press.

Date, C.J., (1986). *An introduction to database systems*, Volume 1, 4th Ed. Reading: Addison-Wesley Publishing Company.

Davis, W. & Waldt, D., (1987, May/June). An SGML overview. *TAG The SGML Newsletter, 1*(1), pp. 4-5.

Exoterica Corporation. (1991). *XGML OminMark Sampler*. Ottawa, Ontario: Exoterica Corporation

Furuta, Richard, (1989). Concepts and models for structured documents. *Structured Documents*, Andre, Furuta & Quint (Eds.) Cambridge: Press Syndicate.

Furuta, R., Quint, V., Andre, J., (1989, April). Interactively editing structured document. *Electronic Publishing, 1*(1), pp.65-76.

Gangemi, Joseph V., (1987, July/Agust). SGML and structured database design. *TAG The SGML Newsletter, 1*(2), p.7.

Goldfarb, Charles, (1990). *The SGML Handbook.* Yurib Rubinsky (Ed.) Oxford: Clarendon Press.

Graf, John, (1987). Where do I start? Beginning document analysis. *TAG The SGML Newsletter, 1*(5), pp. 4-7.

Handley, J. & Weibel, S., (1990). ADAPT: automated document analysis processing and tagging. *EP90: Proceedings of the International Conference on Electronic Publishing,* R. Furuta (Ed.) v110. New York: Cambridge University Press.

James, Geoffrey. (1985). *Document databases,* New York: Van Nostrand Reinhold Company.

Joss, Molly W. (1992, January). Database publishing: streamlined formatting of repetitive records. *Seybold Report on Desktop Publishing, 6*(5), pp.3-14.

Nanard, M., Nanard, L., (1987, October). Interactive manipulation of structured documents. *PROTEXT IV.*

Southall, Richard, (1989). Interfaces between the designer and the document. *Structured Documents,* Andre, Furuta & Quint (Eds.) Cambridge: Press Syndicate.

Vercoustre, Ann-Marie, (1990). Structured editing - hypertext approach, *EP90: Proceedings of the International Conference on Electronic Publishing,* R. Furuta (Ed.) V110. New York: Cambridge University Press.

Walter, Mark, (1992). A look inside JCALS: integrated approach to technical manuals. *The Seybold Report on Publishing Systems,* 21(11), pp.13-26.

(1992). Scholarly journals: Case studies of SGML for electronic publishing. *The Seybold Report on Publishing Systems, 21*(18) pp. 3-19.

Wright, Haviland, (1989). When does "autotagging" become intelligent markup. *TAG The SGML Newsletter, 8,* pp. 10-12.