Why SGML? Why Now?

Yves Marcoux and Martin Sévigny
Groupe Départemental de Recherche sur les Documents Structurés (GRDS), École de Bibliothéconomie et des Sciences de l'Information (EBSI), Université de Montréal, Montréal, Quebec, Canada.
E-mail: marcoux@ere.umontreal.ca; msevigny@sympatico.ca

1. Information Sciences, Documentary Information, and SGML

Let us define ''documentary information'' as any *public* information that has a significant *value* for society. This value could be, for example, legal, historical, or simply a reference value. Because of its value to society, documentary information deserves to be preserved and made accessible to the public in an efficient manner. In our opinion, the fundamental preoccupation of library and information sciences is, and has always been, the design and implementation of efficient systems for the transfer of documentary information within society.

While this preoccupation has not changed in essence since early times, its manifestations have evolved dramatically in the last two or three decades. Probably the most important change that library science has undergone during this period is the ever-increasing use of various technologies for producing and processing documentary information. Librarians and other information professionals must now set up systems capable of processing, disseminating, and preserving documentary information in electronic form. However, because of the plethora of incompatible and short-lived formats that are presently being used, information in electronic form does not readily satisfy the needs of accessibility, dissemination, and preservation that characterize documentary information. Thus, special attention must be paid to these issues in the development of transfer systems for electronic documentary information.

This is where SGML becomes interesting. SGML is one of the first technologies—if not *the* first—whose judicious use can provide documentary information in electronic form with the required qualities for ensuring its efficient accessibility, dissemination, and preservation within society. It is thus no surprise that librarians, archivists, and other information professionals have shown a great deal of interest in this technology. At EBSI, for instance, SGML is now included in research and teaching activities in the fields of information retrieval, analysis, and communication, as well as in archival studies.

In this article, we wish to present the nature of SGML, justify its existence, and explain why today, more than ever, it is possible to tap into the potential of this standard for representing electronic information by making it the cornerstone of documentary information systems. The approach is purposely theoretical. We want to show that SGML is neither a temporary fad of the software industry, nor another proprietary format that a handful of product developers is trying to push onto the market, but rather a robust and general technology, which is based on sound theoretical foundations, and which can offer system designers the same level of data independence for documentary information as the relational model offers for traditional databases. We will also see how SGML differs from ODA (*Open Document Architecture*), its main competitor.

The goal of this article is not to present SGML in detail, nor do we wish to describe its history or use worldwide. The reader interested in these aspects is invited to consult the references listed in the bibliography.

2. The Nature of SGML

SGML is an acronym for *Standard Generalized Markup Language.* Let us examine the meaning of each of these words.

Language

SGML is a format for electronic documents, that is to say, a computer language for describing documents.

A preliminary version of this article was presented, in French, at the conference ''SGML et Inforoutes—pour la diffusion optimale de l'information gouvernementale et juridique'' organized by CRDP and EBSI at Le Musée du Québec on September 27, 1995.

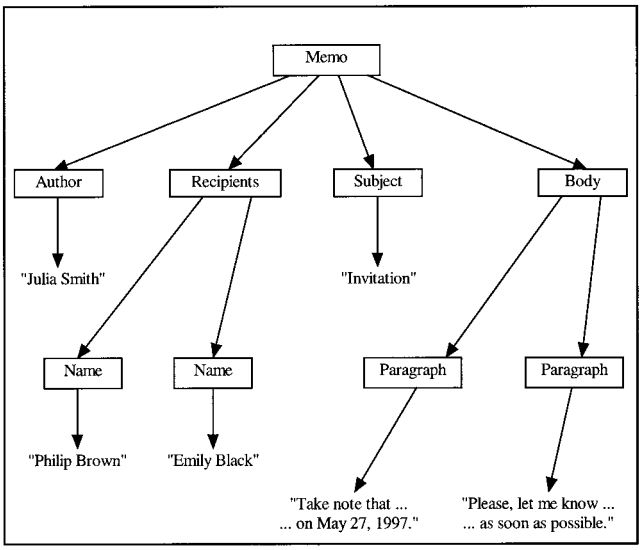

FIG. 1. A memo seen as a hierarchical document.

SGML's notion of a ''document'' is extremely broad, and theoretically encompasses all forms of electronic information. Thus, all of the following are representable in SGML: Word processor documents, spreadsheets, multimedia documents, hypertexts, database tables, and even computer programs.

As with any computer language, SGML has its syntax and semantics. Those familiar with programming usually feel at home with SGML's syntax. It is of course a descriptive language, not a procedural one, thus, it does not use the imperative style. SGML could be described as a language for defining constants, the constants being the documents to be processed, exchanged, preserved. The semantics of SGML is that of tree-structured hierarchical documents. Figure 1 shows an example of the kind of document that can be defined in SGML: A memorandum, such as might circulate in a large corporation. Figure 2 shows how this same document can be represented syntactically in SGML.

SGML's syntax is based on ASCII (*American Standard Code for Information Interchange*) and is thus directly ''eye-readable.''[1] It does not involve any special codes that cannot be displayed or printed directly.

We should point out that SGML standardizes only the expression of a document's structure, and not of any processing or operation to be done on this document—not even formatting. This feature may seem to be a serious limitation at first glance; however, it is in fact a powerful advantage of SGML, which we will discuss shortly.

Let us note finally that SGML's rather daunting syntax (in fact no more so than that of any other computer language) is not necessarily visible to the end users. Indeed, it is quite possible to design an SGML-based document processing chain in which the users (and even the authors) are unaware of the format the documents are stored in (in the same way, for example, as the HTML format is unnoticed by many users of the World Wide Web).

Standard

In the world of electronic document formats, the following levels of standardization can be identified.

- Secret proprietary format: A format defined by a specific hardware or software producer, and whose specifications are unpublished. The original software or hardware must be used to access the documents. Under users' pressure, this kind of format is starting to disappear.
- Public proprietary format: A format defined by a specific producer, but whose specifications are publicly available. Other parties can develop import/export converters for the format, if they want, e.g., RTF (Microsoft's Rich Text Format), WordPerfect.
- De facto standard: A public proprietary format that has become very popular and is recognized by a large number of producers, e.g., GIF (Graphics Interchange Format, introduced by CompuServe), WordPerfect.
- Official standard: A standard defined and adopted by an official standardization body. Standardization bodies include a number of non-profit organizations or consortia, various national and international associations, as well as organizations entirely devoted to standardization on a national or international level, e.g., SGML, ODA.

The *International Organization for Standardization* (ISO), which has its headquarters in Geneva, Switzerland, is possibly the most important standardization body in the world. In particular, the *General Agreement on Tariffs and Trade* (GATT) recognizes the precedence of ISO standards over others. SGML is an ISO standard, ISO 8879, adopted by the ISO in 1986. It is, therefore, a first-rate international standard.

Standardization is the very basis of open systems. An open system is a computer system whose input-output specifications are not only public, but are based on established standards. In their purest form, open systems can easily be combined into processing chains independent of any specific hardware, platform, or software, and whose links are also independent of each other. Confor-

```
<memo>
<author> Julia Smith </author>
<recip>
<name> Philip Brown </name>
<name> Emily Black </name>
</recip>
<subject> Invitation </subject>
<body>
<par> Take note that ... on May 27, 1997. </par>
<par> Please, let me know ... as soon as possible. </par>
</body>
</memo>
```

FIG. 2. SGML text for the document in Figure 1.

[1] SGML's ''concrete reference syntax'' is based on ASCII. However, the role of each character can be redefined and adapted to any character set.

mance to standards improves the interoperability of systems, which in turn increases the users' return on their investments in hardware and software.

The benefits of standardization will of course vary with the popularity of the standards used. The status of official standard provides a guarantee of functionality resulting from serious study. On this point, SGML carries the highest possible credentials, being an ISO standard. In practice, however, the number of products and systems conforming to a standard is at least as important as its official status. In this respect, SGML is now a well-established standard, with a large user base and numerous commercial products. Moreover, the number of users and applications is increasing all the time.

Note that the status of official standard has a particular significance for an electronic document format: It increases the potential *perennity* of documents. Indeed, even a very popular proprietary format can rapidly become extinct as a result of technological breakthroughs or market movements. When this happens, the documents have to be quickly converted into an ''equivalent'' format, otherwise, they would simply become unreadable. This is, however, unlikely to happen in the case of an official standard. The potential perennity, then, that accompanies an official standard is of prime importance for documentary information.

Markup

An electronic document, which can be thought of as a file, has no inherent structure other than that of a linear character (or byte) string. If any parts of the document are to be identifiable, basic conventions must be established. For example, fixed locations within the electronic document can be designated to contain certain specific information, or else a system of pointers and/or separators can be developed. *Markup* (or *tagging*) is another technique; it is the one on which SGML is based.

Tagging consists in inserting into an electronic document short character strings, called *tags,* which indicate the start or end of a part of the document that is to be identified. The tags found in an electronic document are collectively referred to as *markup.* In SGML's concrete reference syntax, the start-tags are usually of the form ⟨gen-id⟩ and the end-tags, of the form ⟨/gen-id⟩. The names (represented here by gen-id) that can appear in a tag are called *generic identifiers,* and in some sense, they indicate the ''type'' of information located between a start-tag and a matching end-tag (matching tags have the same generic identifier). Note that, in general, a start-tag can contain more information than just a generic identifier (for example, *attributes*).

It is important to stress the fact that an author will seldom have to manually enter the tags in an SGML document. Indeed, there are specialized SGML editors that handle much of the tagging automatically. To insert a tag at any point in the document, the author need only choose from the list of tags valid at that particular point. SGML also provides mechanisms to reduce the amount of markup required; for instance, tags can sometimes be omitted when the context excludes any ambiguity.

```
<!DOCTYPE memo [
<!ELEMENT memo - -
     ((author & (date?) & subject & recip & (cc?)), body)>
<!ELEMENT (recip | cc) - - (name+)>
<!ELEMENT body - - (par*)>
<!ELEMENT (author | date | subject | name | par) - -
(#PCDATA)>
]>
```

FIG. 3. A prologue including a DTD for memoranda.

Generalized

In SGML, it is the ''markup'' that is generalized, not the language. Thus, as was once jokingly remarked, SGML should be parsed S((GM)L), and not S(G(ML)).

The tags in SGML identify the ''type'' of information found in the various parts of a document. Naturally, the types of information found in a document will vary with the nature of the document. For a memorandum, the tags presented in Figure 2 would be adequate. For a corporate annual report, however, a totally different set of tags would be required.

Now, SGML must be able to represent all kinds of documents. So, does the standard have to include a generic identifier for every possible ''type'' of information? This would be impossible, of course, since anybody can come up with a new ''type'' of information and document. The solution is to make SGML a *metalanguage,* in which tag sets, as well as usage rules for these tags, can be defined.[2] This mechanism is called *generalized markup.*

In SGML, the definition of a set of tags and of the rules governing their use is called a *DTD,* for Document Type Definition.[3] The language in which DTDs are expressed is an integral part of SGML, and it also uses markup, though of a special kind. Every SGML document has a prologue that identifies the DTD to which the document conforms. (The DTD can be totally embedded in the prologue; alternatively, part or all of the DTD can be located in a file external to the document. In this case, the external part is identified in the prologue by a unique name, such as a file name.) Figure 3 illustrates a prologue including a DTD to which the document of Figure 2 conforms.

[2] These rules determine the acceptable relative positioning of tags. For example, ⟨name⟩ elements (i.e., portions of text delimited by matching ⟨name⟩ and ⟨/name⟩ tags) could be restricted to appear only within ⟨recip⟩ elements.

[3] As Travis and Waldt point out (Travis and Waldt, 1995, The SGML Implementation Guide: A Blueprint for SGML Migration, New York: Springer-Verlag, p. 218), this definition of a DTD is not entirely accurate. However, it is by far the most popular one, and we use it here for simplicity.

3. Justification for, and Necessity of, SGML

The idea of markup far predates SGML. In fact, it can be traced all the way back to the beginning of the computer industry, in the form of ''special characters'' used to control peripherals. From this primitive form, markup evolved into *descriptive markup*, which is the form found in SGML. Let us examine this evolution, and see why it was necessary.

Hardware-Oriented Markup

On some early printing terminals, the character set did not distinguish between lower-case and upper-case letters. However, two control characters, called *shift-in* and *shift-out*, were used to move the printing head from lower-case to upper-case position and back again. In terms of markup, these characters acted as tags that modified the meaning of the intervening text.

Another example of primitive markup is the use of ''escape sequences'' to control printers. The sequence-opening (or ''escape'') character can be seen as a start-tag, and the sequence-closing character as an end-tag. The meaning of this pair of tags is that the intervening text must not be printed, but rather interpreted as a command in the printer's control language. At a higher level, whole escape sequences can be viewed as tags, delimiting portions of text that need to be printed in a special way (boldface, italics, underlined, etc.).

These examples illustrate what we call *hardware-oriented* markup, in which the form and meaning of the tags are dictated by the equipment used.

Generic Markup

Hardware-oriented markup has an obvious disadvantage: It is dependent on the type of equipment used. If, for some reason, the equipment is changed, the markup must be changed, too. For this reason, hardware-oriented markup is almost never stored in electronic documents. Instead, the authoring software usually stores its own codes, which are independent of the hardware used. This is called *generic markup*. The generic markup can be converted into hardware-oriented markup by an appropriate *driver* when needed—for example, to print a document. Then, should different equipment be used, only the driver need be changed; the electronic documents themselves require no modification.

The formatting codes used by popular word processors are examples of generic markup. These codes are generated by the software and stored in the electronic documents. When the time comes to print a document, a driver (supplied by the software or printer producer) specific to the type of printer used translates the codes into control instructions for the printer.

Generic markup is a definite improvement over hardware-oriented markup, but it does not solve all the problems of document production. For example, suppose we have prepared a series of technical manuals with a word processor (without using a style sheet) and we want to modify the size and horizontal position of the titles. Since the formatting instructions are stored within the electronic documents (in the form of generic markup tags), we obviously have no choice but to modify each one of the documents. This, in itself, is quite a chore. However, the real problem is that in order to modify the formatting of the titles, we must be able to *locate* them in the documents. If the titles already have some distinctive formatting not used anywhere else in the document (for instance, centered and in boldface), then this might be done automatically. Otherwise, it will have to be done manually.

There is an additional problem if we want to continue producing the manuals in the original style. The source documents will have to be duplicated, and each copy maintained separately. This makes updates much more costly.

Descriptive Markup

These are some of the problems associated with generic markup, but there are others, for example, problems related to the conversion of documents from one format to another, or to the exchange of documents. The cause of these problems is that generic markup is *procedural;* it expresses one particular operation to be performed on the documents, namely, formatting. When another operation must be performed, the markup does not help at all, in fact, it can sometimes become an obstacle. The solution is to expunge from the markup all references to specific operations, which is precisely the idea behind *descriptive* markup, the kind proposed by SGML.

The fundamental hypothesis behind descriptive markup is that a document is best described by its inherent *structure*, rather than by the operations to be performed on it. With descriptive markup, tags identify all the structural elements in a document and associate a ''type'' to each of them, but do not provide any indication as to how these elements are to be formatted or otherwise processed. It is the *applications* processing the documents that determine how each type of structural element is treated.[4]

The Benefits of Descriptive Markup

By removing processing instructions from the electronic documents, descriptive markup introduces a natural and beneficial separation between contents and pro-

[4] SGML permits and encourages descriptive markup, but it cannot impose it. The meaning of the tags is defined outside SGML, by the designers of a system, and nothing can prevent it from being formatting instructions (which is largely the case, for example, in HTML). There is also an explicit syntactical construct in SGML for ''processing instructions.'' However, this use of markup is not really in keeping with SGML's underlying philosophy.

cessing. Authors can concentrate on the contents of the documents, while typographers and computer professionals make layout decisions, and define more generally how the whole collection of documents is to be processed. We call this separation the ''work factorization'' of document production.

One benefit of this factorization is that writing documents and deciding how to process them can be done at two different times. For example, documents can be entered and stored before any application has been developed to process them. It is also possible to write new applications to process existing documents without having to modify them in any way.

Descriptive markup can also greatly improve automatic or computer-aided indexing of documents, as well as information retrieval. Indeed, with descriptive markup, it is possible to know in which structural element of the text (title, caption, quotation, footnote, etc.) each word occurrence is located. This aspect of descriptive markup is of particular interest in documentary information.

The Added Value of SGML

What we have said above argues in favor of descriptive markup in general, not just SGML. A simple way of implementing descriptive markup would be to fully use the style-sheet functions of a word processor. These functions are sometimes sophisticated enough to permit descriptive markup. Does SGML offer any advantage over this approach?

The answer to this, is that SGML offers *standardization,* and the *robustness of generalized markup.* We have already indicated the importance of standardization. Generalized markup, for its part, allows a system designer to ensure that the markup is always used in a consistent manner, by defining rules governing the use of tags. This is a form of *validation* of documents at creation-time.

Of course, in more detailed matters, SGML offers many possibilities that make it especially interesting for document-processing systems. This subject, however, is beyond the scope of the present article.

The Costs of SGML

The benefits of SGML are not without cost. The main expense associated with SGML is its formal complexity. In order to design and implement an SGML document-processing system, it is necessary to have a fairly thorough understanding of the standard, its underlying principles, and the details of its application. Thus, when an organization decides to adopt SGML, it must anticipate costs involved in training its personnel, even if part or all of the design, development, and implementation work is given to specialized consultants. Tools, products, and services will probably need to be evaluated.

Another cost factor is the very high level of generality that SGML tools have, by definition. This generality increases the cost of developing the products and, inevitably, is reflected in their selling price.

Yet another cost factor for the organization is the transformation SGML brings to work habits. Earlier on, we mentioned the factorization of work between authors on one hand, and typographers and computer professionals on the other. Another consideration is the constraints that a DTD imposes on the authors. They no longer have absolute freedom in the authoring process; the structure of their work must comply to the rules set by the system's designers. Even worse, perhaps, the authors no longer control the exact appearance of their printed documents. Such changes may be something of a ''culture shock'' to writers. If, on the other hand, writers continue to work with a word processor, and the documents are subsequently converted to SGML, then the cost of traumatizing the authors is replaced by the perhaps equally high cost of the conversion.

Since the DTDs are of paramount importance in an SGML system, their design must be based on a careful documentary analysis, and represents an additional cost. It is often given to specialized consultants.

4. Why SGML?

A descriptively tagged document contains information on its own *logical structure,* and is thus called a *structured document.* A format which allows the representation (whether by markup or otherwise) of structured documents is called a *structured document format.* SGML is thus a *standard structured document format.*

Why would SGML be any better than other standard structured document formats? Indeed, the first question should be: Are there any others? The answer is yes, though very few. In fact, the only format that can compete with SGML is ODA, *Open Document Architecture* (originally named *Office Document Architecture*). It is, like SGML, a standard structured document format; therefore, it has many possibilities similar to those in SGML, in particular, it provides the equivalent of descriptive markup.[5] Moreover, ODA is also an ISO standard: ISO 8613, adopted by the ISO in 1989. Thus, ODA is a potential alternative to SGML. Although a detailed description of ODA is beyond the scope of this article, we will now outline some of its characteristics, by way of comparison with SGML.

Differences between SGML and ODA

An ODA document is composed of both a logical and a physical structure, as well as contents elements shared by the two. Style-sheets (in the form of structural-compo-

[5] Although ODA allows structuring documents in a way equivalent to markup, it does not actually use this technique. We use this terminology for simplicity.

nent attributes) establish the correspondence between both structures. The logical structure is similar to the structuring offered by SGML. The physical structure corresponds to the document as it appears on paper. Unlike SGML, then, ODA can specify the formatting of documents. This constitutes one of the main differences between SGML and ODA.

This difference can be understood in terms of the design objectives of ODA, which was first and foremost intended for office automation applications. In that context, the operations of authoring and formatting documents are usually close to each other, both from a time and administrative point of view. It was thus natural that formatting instructions be included directly in the electronic documents, rather than placed in separate stylesheets.

Another difference between SGML and ODA is that ODA documents are not directly eye-readable. This is because the data structure on which an ODA document is defined is *not*, as in SGML, the character string, but rather an object-oriented structure.[6] Even if ODA explicitly provides an interchange format (ODIF, or *Open Document Interchange Format*), none of the forms it can take (there are three such forms) is intended to be directly readable. In this respect, ODA is not as close as SGML to the universe of traditional documents.

SGML and ODA also differ in the way they accommodate non-textual contents (images, sound, etc.) and foreign-language character sets. In SGML, any character set can be defined and used, and an arbitrary number of ''notations'' can be declared for non-textual contents. A natural consequence of this is that the reader of a document must have the necessary tools to correctly interpret these elements. If two parties wish to exchange documents incorporating such elements, they must agree beforehand on which character sets and notations are going to be used.

One of the design objectives of ODA was to support the ''blind exchange'' of documents, i.e., without any prior agreement between the parties. To meet this objective, there was no other solution than to restrict the allowed character sets and notations *in the standard itself*, and this is essentially what ODA does.[7] Presently, standard representations for text, raster graphics and vector graphics are included; future extensions are expected to add new types of contents.

Having to standardize representations for three types of contents and their associated formatting functionality makes ODA a voluminous standard, that would be difficult to implement in its totality. To allow for subsets of the standard to be defined and used in specific applications or contexts, the notion of *Document Application Profile*, or DAP, is included in ODA. A DAP essentially defines a subset of ODA, on which groups of users agree.[8] The development of DAPs is not done by ISO; presently, it is coordinated by a group called PAGODA (*Profile Alignment Group for ODA*), linked to the CCITT (*International Telegraphic and Telephone Consultative Committee*). Three DAPs have been defined so far: Q111 (or FOD11), Q112 (or FOD26), and Q113 (or FOD36). Their levels of functionality range from simple text files to desktop-publishing documents.

Why Not ODA?

ODA provides the equivalent of descriptive markup, and it is an ISO standard; so, is it a viable alternative to SGML for documentary information systems? As argued previously, the value of a standard is highly dependent on its popularity. At the present time, SGML is far ahead of ODA, both in the number of commercial products available and the number of users.

The family of software products conforming to SGML is very rich (see Appendix). In contrast, software products conforming to ODA are relatively scarce, and all have appeared fairly recently.[9]

As far as the number of users is concerned, many important projects dealing with electronic documents have adopted SGML, whereas, to our knowledge, ODA is just starting to be used, and only in the context of office automation.

In a few years, we will see whether the arrival of further ODA products will have made it a real competitor to SGML. It is still entirely possible that ODA will ultimately find its share of the market. Nevertheless, today, a valid element in the answer to the question: ''Why SGML, and not ODA?'' is ''Because it is more popular.''

Reasons for the Success of SGML

Some of the factors explaining SGML's popularity are purely circumstantial. For instance, part of SGML's success is certainly due to the fact that it came out 3 years before

[6] For readers familiar with object-oriented systems, it must be added that this structure does not include any method.

[7] The character sets are not really restricted in ODA itself, however, only predefined standard sets can be used.

[8] Theoretically, a DAP could be used to *extend* ODA rather than limit it. However, all references to DAPs in the literature present them as a means of defining subsets of ODA, and this is true of all DAPs developed so far.

[9] We are aware of the following products: The *ODA Viewer*, from the French producer Bull, is a Microsoft Windows application for viewing and annotating ODA documents (FOD26 level). A limited version, called *FreeView* is available free of charge on the Internet. Two other products handle conversions between ODA and various formats: One is *ConvertPerfect*, from Novell, a stand-alone package offering two-way conversion between ODA (FOD26) and many popular word-processing formats. The other is *WinWord 6.0 ODA Converter Kit*, from Microsoft, a ODA (FOD26) conversion filter for Microsoft Word. With this filter, ODA becomes yet another format accepted by Word. An ODA Toolkit was developed by the *ODA Consortium*, a group of computer companies interested in ODA. This toolkit has been available since March 1993, and supports FOD36 since March 1994. To our knowledge, however, no commercial product supports this level yet.

ODA, and that it was adopted very early on by the CALS program (*Computer-aided Acquisition and Logistics Support*) of the U.S. Department of Defense (DoD).[10] However, we believe some credit for SGML's success is due to its inherent features, in particular the following two:

The first feature is ''eye-readability.'' This is of course a psychological factor. But we believe it has an non-negligible impact on the acceptance of a format, nevertheless. Someone who is used to working with paper documents will probably find using eye-readable electronic documents less daunting; the shock of the new technology is thus reduced. For example, publishing houses, accustomed, as they have always been, to paper, were nevertheless among the first users of SGML. The importance of eye-readability must really not be underestimated.

The second feature is the fact that SGML does not standardize formatting instructions. As we have seen, ODA is different in this respect, since it allows standardized formatting instructions (in addition to descriptive markup) to be included in the documents. This characteristic of ODA permits the blind-exchange of formatted documents, but at the cost of having to standardize all possible formatting functionality—which is quite an ambitious task. In practice, restricted DAPs were developed first, causing a ''slow start'' which might have discouraged potential supporters, who were asked to give up functionality they had come to rely on. The more than 5 years that elapsed between the adoption of ODA as a standard and the arrival of the first end-user products illustrates, in our opinion, the deterrent effect of this slow start.

SGML, for its part, did not have to wait for the standardization of formatting functionality, and was ready to take off when the users were. Ironically, a proposed standard for formatting functionality has recently passed the last stage before adoption as an international standard by the ISO. It is DSSSL, or *Document Style Semantics and Specification Language.* Long awaited by the SGML community, DSSSL is the ideal complement of SGML for system designers wishing to integrate standardized formatting instructions into their systems. However, since SGML and DSSSL are two distinct standards, the 10 years or so that elapsed between their respective adoption did not keep SGML from gaining initial acceptance by users.

HyTime and Other Formats for Hypermedia

Hypertext structuring increases in popularity every day; thus, no document format can afford to ignore it. SGML has always allowed the representation of hypertext links; for example, one possible way to do it is to decide that certain tags are used to identify the link anchors (origin and end points). Any application aware of the meaning of these tags is then able to recognize the links, and act accordingly if and when the user activates them (for example, by clicking on them).

It is of course possible to establish *standard* ways of representing hypertext links—as well as many other hypermedia functionality—in SGML documents. The advantages of such a standardization are obvious. An extension to SGML was designed for this purpose: HyTime, or *Hypermedia/Time-based Document Structuring Language* (ISO 10744, adopted by the ISO in 1994). HyTime provides, among other things, standard ways of representing hypertext links in SGML documents. Any HyTime-compliant application is able to recognize these links and obey them when they are activated by the user.

Note that HyTime is a compatible extension of SGML; it is therefore not an alternative to SGML, simply a way to go further with certain types of documents. To our knowledge, there are no end-user products that support HyTime in its entirety. There is, however, a HyTime ''engine'' available to software developers. Also note that the World Wide Web browser for SGML documents, *Panorama*, recognizes a subset of the links representable in HyTime (see Appendix).

The very existence of HyTime is an argument in favor of SGML. Indeed, it guarantees the possibility of evolving towards more functionally sophisticated documents. Note that an extension to ODA for hypertext structuring is also expected, a ''HyperODA,'' which has apparently been on the drawing board for some years. But, since it is only a proposal, and not an actual standard, it may be several more years before compliant products appear on the market.

While on the subject of hypertext documents, we cannot avoid mentioning the biggest hyperdocument of all, the World Wide Web (WWW). Though deprecated by purists for not respecting SGML's philosophy of descriptive markup, the WWW is nevertheless the single most important SGML application ever developed. Indeed, the language of WWW documents, HTML (for *Hypertext Markup Language*), is essentially an SGML DTD.

5. Why Now?

In spite of its conceptual simplicity, SGML is a powerful and complex formal system which requires sophisticated tools and brings about important changes in any organization that adopts it. Because of the costs associated with it, SGML has always been a technology that is embraced by necessity, when the value of the information involved is so high, and the benefits of its universal accessibility and preservation are so great, that they outweigh the expenditure required in converting to SGML. In our opinion, this necessity component will always be a con-

[10] The name was later changed to *Continuous Acquisition and Lifecycle Support.* The original DoD program was aimed at reducing the cost of producing, storing, consulting, and updating technical documentation. For CALS, SGML represented a tool for the rational treatment of documentation, and it was adopted not only in principle, but also in the form of specific DTDs. The CALS program affected not only the direct suppliers of the DoD, but also its sub-contractors; it thus created a genuine ''wave'' in the industry, that later expanded into the European and Canadian markets. Hence, thanks to CALS, SGML enjoyed substantial visibility and promotion.

sideration in any decision regarding a changeover to SGML. But it is a consideration whose importance is diminishing, thanks to a number of factors.

One such factor is that the selling price of many SGML products is now steadily decreasing, partly because development costs have been absorbed, and partly because new competing products are always appearing. Also, most SGML products are now available on microcomputer platforms and have reached a high level of usability.

Another factor is the level of knowledge that information professionals have of SGML. The intrinsic value of any technology is qualified by the availability and number of designers, analysts, programmers, and technicians who master it. Today, SGML has made its way into many commercial and academic curricula. Here, too, SGML is ahead of ODA.

In the light of all these factors, we believe that the cost/benefit ratio of SGML solutions has never been better. In our opinion, any organization involved in documentary information management should seriously consider the options SGML has to offer.

6. Conclusion

In this article, we have tried to present the nature of SGML, to explain its existence, and to show why this standard format for electronic documents is the natural cornerstone of today's documentary information systems. We have tried to show that SGML is neither a temporary fad of the industry, nor just another format among many, but rather a robust technology, based on solid theoretical foundations. We have also made some comparisons between SGML and ODA, its closest competitor. Of course, there are many facets of SGML we have not covered, such as the notion of sub-document, and the parallel tagging of the same document according to two or more DTDs. Yet, we hope to have given the reader an idea of the principles and philosophy underlying SGML.

We have little doubt that descriptive markup is the only viable solution for the efficient management of documentary information in electronic form. The independence from authoring, research, and processing tools that it provides, is an essential condition for the proper and adequate accessing, dissemination, and preserving of documentary information. Of all the possible approaches to descriptive markup, SGML seems to us to be the most promising one, partly because of its great popularity, which maximizes the benefits associated with standardization, and partly because of intrinsic qualities, such as eye-readability and the clear separation it imposes between document contents and their processing.

In our opinion, the question is really no longer ''SGML, yes or no?'' but rather ''which DTDs?'' Thus, the next level of standardization we should be seeking for documentary information, is that of DTDs. Already, many organizations are working in this direction, for example, the Association of American Publishers (AAP) and the Text Encoding Initiative (TEI). A couple of proposals for USMARC DTDs have circulated on the Internet.

Naturally, any DTD standardization will affect the way information is managed in various places around the world. These changes will sometimes be difficult and costly, but they would be even more so without the prior basic standardization that SGML will have brought (after all, changing a DTD is simpler than adopting SGML in the first place; for one thing, the same tools can still be used to process information). In return, these changes will bring us a little closer to a better world, in which documentary information flows freely, liberated from the captivity of proprietary formats.

7. Appendix: Examples of SGML-Compliant Software Products

Many SGML software products are aimed at the production of documents; they can perform all major functions of the document production chain. Document authoring is handled by SGML editors, such as *Author/Editor* from SoftQuad, and *Adept Editor* from Arbortext, or by add-ins for popular word processors, such as *Near&Far Author for Word* from Microstar, and *WordPerfect SGML Edition* from Novell. *FrameMaker+SGML* from Adobe will produce paper documents from SGML source documents. The conversion of SGML documents to and from other formats is handled by general SGML parsers/taggers/converters, such as *OmniMark* from Exoterica, and *FastTag* from Avalanche.

Some packages can publish SGML documents in electronic form. The *Dynatext* application suite from Electronic Book Technologies will let a publisher assemble, manage, and publish a collection of documents as an electronic book; it includes a hypertext browser to read the electronic books. The *Explorer* package from SoftQuad offers similar functions. These packages are particularly well suited to publishing information on CD-ROM.

It is now possible to publish full-fledged SGML documents (any DTD) on the WWW without prior conversion to HTML. *Panorama* from SoftQuad, is an SGML document browser that can be used in conjunction with a WWW browser. It will display any SGML document based on one or more style-sheets; it recognizes a subset of HyTime for hypertext links. SGML documents can also be converted to HTML on-the-fly by *DynaWeb*, a WWW server from Electronic Book Technologies.

Finally, there are database management systems specially designed to handle SGML documents, for example: *BasisPlus SGMLserver* from Information Dimensions, and *Ful/Text* from Fulcrum Technologies.

8. Bibliography

Barron, D. (1989). Why use SGML? *Electronic Publishing, 2*(1), 3–24.
Campbell-Grant, I. R. (1991). Introducing ODA. *Computer Standards & Interfaces, 11,* 149–157.

Cover, R. The SGML Web Page. ⟨URL:http://www.sil.org/sgml/sgml.html⟩

Danish Standards Association. (1991). *SGML—ODA: Présentation des concepts et comparaison fonctionnelle.* Paris: AFNOR.

DeRose, S. J., & Durand, D. G. (1994). *Making hypermedia work. A user's guide to HyTime.* Boston: Kluwer Academic Publishers.

Goldfarb, C. F. (1991). *The SGML Handbook.* New York: Oxford University Press.

Kuikka, E., & Nikunen, E. Systems for structured text. ⟨URL:http://www.cs.uku.fi/~kuikka/systems.html⟩

Marcoux, Y. (1994, Spring). Les formats normalisés de documents électroniques. *ICO Québec, 6*(1,2), 56–65.

Marcoux, Y. (1994, Summer–Fall). Les formats de documents électroniques en archivistique: La solution au problème des archives électroniques passe-t-elle obligatoirement par les formats normalisés de documents structurés? *Archives, 26,*(1,2), 85–100.

ODA Consortium. ODA Consortium Home Page. ⟨URL:http://odac.novell.com/⟩

Pepper, S. The whirlwind guide to SGML tools and vendors. ⟨URL:http://www.falch.no/people/pepper/sgmltool/⟩

Role, F. (1991). La norme SGML: Pour décrire la structure logique des documents. *Documentaliste-Sciences de l'information, 28*(4–5), 187–192.

Travis, B., & Waldt, D. (1995). *The SGML implementation guide: A blueprint for SGML migration.* New York: Springer-Verlag.

Van Herwijnen, E. (1994). *Practical SGML* (2nd ed.). Norwell, MA: Kluwer Academic Publishers.

Watson, B. C., & Davis, R. J. (1991). ODA and SGML: Assessment of co-existence possibilities. *Computer Standards & Interfaces, 11,* 169–176.

Wright, H. (1992). SGML frees information. *Byte, 17*(6), 279–286.