J. TECHNICAL WRITING AND COMMUNICATION, Vol. 32(2) 155-170, 2002

DOES BEING TECHNICAL MATTER? XML, SINGLE SOURCE, AND TECHNICAL COMMUNICATION

FILIPP SAPIENZA
University of Colorado–Denver

ABSTRACT

XML is a recent Web design language that will enable technical communicators to produce documentation that can reuse information and present it across multiple types of media for diverse audiences. However, little is understood about how XML will impact technical communication in terms of theory, academic research, and pedagogy. In this article, I argue that XML requires more interdisciplinary approaches toward the teaching and research of technical communication, particularly with respect to the integration of technical and rhetorical knowledge.

One day a major university hospital calls your office and asks you to recommend graduates who can develop a content management system for surgery dictations. Under the current system, surgery dictations are handled manually by typists transcribing tape recordings into Microsoft Word documents. Patient registration data must be accessed via a separate text-only terminal and manually keyed into these Word documents for each "opnote." Patients with more than one procedure currently require additional opnotes to be created and filed. Hospital administrators want a system that links dictations to patient admissions records—that is, one complete record per patient—as well as an improved document structure to facilitate transcription. They believe the best way to accomplish this is to have transcriptionists enter dictations via fill-in forms with fixed fields over the Web. They want the information saved in a format compatible with a wide variety of applications, including spreadsheets, databases,

and word processors. They also want the information accessible not only to the Web but to wireless media. This way, surgeons on-call may rapidly access data from remote locations to advise attending staff at the hospital in emergency situations.

Perhaps you are thinking this kind of task does not seem likely for a technical communicator. You may consider sending the caller to the computer science or information systems department. I myself once developed an opnote system for a university hospital in Michigan. There was no Web, only a MacIntosh network and a host of applications that I had to configure to work together. At that time, I was a starving college student whose job description was computer programmer, *not* technical communicator. But why not ask the technical communicator? To pose the question differently, what specific skills does the technical communicator uniquely offer the situation that cannot be furnished by a computer programmer?

While one can appreciate the technical wizardry necessary to put together this system, I claim that this kind of task should and likely will require the involvement of technical communicators. I say "should" because surgery opnote management is not only a logistical problem requiring computer skills, but also involves matters of genre, audience, and rhetorical effectiveness in emergency situations. Information must be timely—the notion of *kairos* comes to mind—and stored in one source so that updates are consistent and accurate. I say "will" because current trends in electronic media technology are moving toward systems that will more tightly integrate "data" with "prose" in single-source format. The XML markup language has been designed specifically for this purpose. XML is a self-describing data and document format permitting rapid data exchange, mixed media integration, platform independent design, and reusability of content. These technologies did not exist when I designed my system. (In fact, the system I ended up designing could not directly interface with the main hospital database due to severe incompatibility issues that would have required a large capital investment to improve.) The coordination of different applications to put together the system required someone who had a lot of technical skill, but this meant that the "rhetorical" half of the project was largely neglected.

In this article, I claim that technical communicators are uniquely poised to seize upon opportunities that integrate rhetorical craft with technical wizardry. Indeed, they will probably have to integrate the two areas of knowledge. Technical communicators will probably face a day when all organizational documents are saved in XML format because they can be easily integrated across different departments (such as accounting and volunteer services) and different programs (such as word processors and spreadsheets). Even today, many HTML Web pages are "transformed" XML documents. Every manual, online help screen, spreadsheet, and database will need to communicate with one another and be accessible through the Web, wireless devices, text-only terminals, print, and a variety of other media.

As XML technologies gain a stronger presence in electronic content development, we who research and teach technical communication must reflect on how this technology will transform what we do. Writing in XML is not as linear as word processing. It is similar to the "lexias" and "nodes" of hypertext [1, pp. 3-4], although XML utilizes a "tree-like" structure. Presently in our field, one can ignore Web development as a specialization and focus on traditional paper-based documentation. With XML, it may not be possible to make this distinction because it is used as a data formatting language rather than limited to presentation markup. In other words, technical communicators will need to not only consider presentation but also data management, a domain currently reserved for database and computer specialists. What new theoretical problems does the modularized writing process of XML pose? What kinds of knowledge domains will students in technical communication need to master in order to competently address XML technical communication tasks in the working world? How might XML transform not just how we teach technical communication but how we research it? In this article, I address these questions by analyzing the potential impact of XML and single sourcing on theory, research, and teaching. Before addressing this impact, let us first examine what XML and single source technology are. (Note: a glossary has been created to clarify technical terms.)

WHAT IS SINGLE SOURCE?

Single source means creating content once and using it many times. It does not refer to a process of cutting and pasting the same content into multiple documents [2, pp. 189]. Instead, document "modules" are referenced from a database or document management system and combined and reused as necessary. The document management system may be updated by multiple content developers from various locations. When the reader retrieves a document, the system combines modules according to profile and preference information that the reader supplies. Many e-commerce Web sites single source product information across the Internet.

With single source documentation, writers create one document that can be output to any number of format displays and customized for specific users. Single source benefits technical communicators who need to produce both print and Web documents, who create documents to cover related products (for example, a surgery instrument Model B with all the features of Model A plus some advancements), and who need to quickly convert and update documents across a wide range of organizational divisions and platforms (display surgery data for doctors, medication profile for pharmacists, etc.). Single source allows developers to write documentation that is device independent. In other words, there is no longer a need to develop separate print, help, HTML, PDF, and other versions of the document. A document is updated one time and then "transformed" rapidly and dynamically into multiple formats as needed.

SINGLE SOURCE AND THE WORLD WIDE WEB

A popular procedure for single source documentation is to work with Adobe FrameMaker and Quadralay WebWorks. These applications function very well, capable of producing documents in PDF, HTML, WinHelp, and other formats. But they are expensive to purchase and upgrade and constrained to a proprietary interface for development. Furthermore, they have limited real-time data and database management capabilities. Another method to produce single source documents on the Web bypasses single source software altogether and instead combines a database server with a programming language. The documentation specialist places all of the Web pages into a relational database and produce the output using either Java, a CGI script, or a server embedded programming language such as PHP. For example, a specific database field may contain text to include in a syllabus, while another database may have fields containing the HTML markup tags. Using the scripting language, the content of the different databases are retrieved, combined and then presented to the Web browser. Moreover, PHP and other scripting languages contain functions and add-on modules that facilitate export of data into PDF and other formats, so that the information may be transported to different media.

LIMITATIONS OF HTML, SCRIPTING LANGUAGES, AND DATABASES

Combining a scripting language with a database compensates for the limitations of HTML, currently the Web page markup language of choice for most designers. The popularity of HTML can be attributed to cost (it is a non-proprietary code requiring nothing more than a text editor) and ease of use. But HTML is simply a document formatting language that by itself works only on a World Wide Web browser. Nothing in the document structure identifies anything useful about the kind of data contained: whether a particular part of text is a number, a word, a book chapter or section. For example, using HTML code tags to specify boldface only says "whatever exists between the and tags, make it bold." HTML allows no direct way to say "whenever a book chapter begins, do X with the information." Because of this limitation, it is impossible to "template" HTML files such that headings and footers can be written once and applied to many pages on a site. Instead, designers must manually cut and paste codes across pages to ensure consistency of layout, import pieces of code using something called "server side includes" or develop pages in Dynamic HTML, a language that is not fully compatible with every Web browser. By itself, HTML is not very good at allowing for data exchange across different media, such as providing printer-friendly PDF formatted files or documents suited for wireless devices.

Those who wish to transport data from HTML documents to other platforms must manually create a separate HTML document customized for each platform or write scripts to parse out data and format for different outputs.

Databases offer searching and sorting of data as well as rapid access, but the organizations who design them usually encode the record structures in a specific order that makes data difficult to exchange with other organizations. Furthermore, databases require "strict" integrity of data fields, meaning that it may be difficult to have one record contain multiple data structures or even mixed types (such as text and numbers). Mixed types can also slow down the performance of a database. In fact, databases are not really suited to handle dynamic Web content at all: they are best used to organize, store, and rapidly index large masses of similarly structured data. When it comes to mixes of prose, graphics, tables, numbers, and other data that commonly characterize technical documents, the capabilities of databases must be stretched.

Finally, databases and scripting languages generally make the Web server work hard. Any sorting or indexing of information takes place on the Web server instead of using the resources of the client (browser's) own machine. Each time a request is made to process data, the overall performance of the system suffers time lags and even data lockups. In a critical care facility, this problem is simply unacceptable.

THE PROMISE OF XML

XML (Extensible Markup Language), developed and endorsed by the World Wide Web Consortium (W3C) in February 1998, offers Web designers a language that modularizes content for information exchange in ways that make it easier to distribute the same content across multiple platforms. Like HTML, XML marks up the content of a document with tags, but XML tags identify "metainformation" about the data they enclose rather than simply describe presentation formatting. A surgery dictation might contain HTML code that resembles the following:

```
<b>NAME</b>: Jane Smith<br>
<b>REGISTRATION NUMBER</b>: 123456789<br>
<b>ATTENDING SURGEONS</b>: Dr. Sally Jones<p>
<b>PROCEDURE</b>: . . .
```

In this example, the HTML tags don't say "change Name, Registration Number, Attending Surgeons, and Procedures to bold" (using the) tag. They say, "change whatever data is contained by and into boldface, but I don't care what that data is. Conversely, one could render the opnote in XML as follows:

```
<opnote>

<name>Jane Smith</name>
<reg>123456789</reg>
<attending>Dr. Sally Jones</attending>
<procedure> . . . </procedure>

</opnote>
```

We can then code an XML stylesheet (XSL) to say, "whenever you find a name, a registration number, attending surgeon, or procedure, change it to boldface." In other words, XML knows that there exists a name, registration number, and so forth, and that it should do something according to those labels. XML tags differ from HTML tags in that XML tags do not intrinsically specify how a unit of text should be presented: they merely provide metadata that is used by an XSL style sheet and other applications to determine the proper formatting and distribution of content.

XML saves system resources because most of the data processing takes place on the client machine, freeing resources on the server. Because the server is a shared computer, speed goes down the more users access its resources. A client computer (such as the one on my desktop) does not share resources with anyone. Additionally, XML data can be transformed with XSL into any kind of presentable markup language, including HTML or even another XML document with a different tree structure. It is often used to produce wireless markup (WML) and other online help markup languages. XML allows developers to combine different documents (manuals, procedures, demographics) into one document and transform parts of the document for specific users and contexts. A list of patient names and registration numbers and surgeons could be transformed into a spreadsheet format, while a set of documents containing only name and procedure descriptions could be transformed into a word processing format and sent to a head surgeon. Updates to the original document would only have to be performed one time on one file, not cut and pasted across many documents. The content "modules" of an XML document are not re-pasted but re-used (re-sourced) from the same file. XML also allows templating of documents so that designers may ensure a similar look and feel across an entire site. This template need be developed only one time, and applied across multiple content pages. In sum, XML allows for greater consistency, in and across documents, cross-media publishing, reusability of information modules across documents, easier information sorting and retrieval, and customization of information [3, p. 59].

XML AND TECHNICAL COMMUNICATION THEORY

An important theoretical aspect of XML in technical documentation has to do with the way that XML designers establish contextual relationships among

documents. In interpersonal communication, any interaction involves contextual understandings or "cues" that are largely metalinguistic and serve to give meaning to the words, sentences, and paragraphs exchanged between and among parties [4, p. 131]. When context is unclear, the message is often misunderstood or ambiguous. In HTML, this context is not as important to codify because HTML merely specifies presentation formatting, the contextual understandings are implied through the text (as opposed to structure) of the document, not in the markup tags themselves. But because XML is best suited to information exchange, it is necessary to embed within an XML document the necessary contexts for its reception by another machine.

XML developers must specify not only tags for how to structure information but additional codes that dictate how the tags are supposed to be interpreted by other organizations receiving the data. The way that this communicative context is encoded in XML is through the development of "schemas" and "DTDs" (Document Type Definitions) [5, p. 13]. Schemas and DTDs create a shared context identified by formal rules and sequences that XML tags must follow. For example, a DTD might state that everything encapsulated by <reg> and </reg> (a REGISTRATION "record") must only contain a series of numbers nine digits in length. One can also specify the sequence of elements (name, followed by registration number, followed by procedure, etc.). A DTD constrains the XML document so that anyone wanting to create future opnotes must conform to a specific valid layout.

The slippery issue concerning schemas and DTDs is what exactly should constitute a valid document structure, grammar, and syntax, and who should develop it? This latter question is not simply a bureaucratic issue but a political one, because in effect the person developing a DTD will be asked to write a new linguistic context, and perhaps a new language, that an entire organization or several organizations must be willing to adopt and share. This responsibility is implicit in XML development because the efficiency of XML-based single-source documentation best works if the scope of the project is adopted by everyone involved. In a hospital, for example, any redundancy of critical data costs time and negatively impacts the quality of care given to patients. If even one major department cuts and pastes surgery data into their own specialized Word files, it makes the XML/single source system redundant and possibly out of date. In short, the technical communicator designing a DTD acquires a position of significant power, potentially shaping how the organization structures knowledge about products and processes.

THE GENRE OF XML

Let me frame the problem in a rhetorical context by suggesting that the DTD designer uses computer codes to create a *genre* for documentation exchange. Genres rhetorically gives shape and recognizability to speech acts that are evoked

by specific contexts. According to Carolyn Miller, genres are not static forms but rather ways of seeing the world and therefore forms of social knowledge [6, p. 157]. Miller suggests that genres arise from the thick of discursive activity embedded in rhetorical situations that are always contextual and that result "not from 'perception' but from 'definition.' Because human action is based on and guided by meaning, not by material causes, at the center of action is a process of interpretation" [6, p. 156]. Rhetorical practice makes public both the interpretation of a specific issue and the intentions of those who articulate that interpretation, and the fitting responses over time to addressing an issue therefore become highly contextualized speech acts implicated as much by culture and community as by the given exigence that a rhetorical act is supposed to resolve.

According to Catherine Schryer,[1] the medical record is somewhat unique in that it is a genre that not only shapes the "socialization within the [medical] community but also the subsequent interaction of the communities with the rest of society" [7, p. 204]. Schryer argues that any kind of record keeping system socializes people into a certain kind of interaction. Schryer studied a veterinary school in which professors complained that students had what they called a "green grad" problem, "a staccato style of writing and speaking with both other veterinarians and clients" [7, p. 228]. The professors saw the literacy problem as primarily connected to writing exams, but Schryer traced it to recent changes in how the school kept medical records. The new system "socialized" students into treating cases as single word "problem lists" [7, p. 221]. A medical problem only became recognizable and treatable when encoded in this format. Veterinary professors emphasized this rhetorical structure in their examinations, penalizing students who may have gotten the "right" answer but produced a response in a different format. The list emphasis meant that students started producing documents that were "coherent" but not "cohesive." Instead of explicit cohesion, the single word reader must read "a great deal of tacit information into these accounts. . . . The new formatting of records were also organized in such a way that writers do not have to include many of the cohesive ties needed for readers outside the narrow circle of students and instructors at the college" [7, p. 224]. According to course outlines, behavior viewed through the animal owner's eyes was regarded as "subjective" data [7, p. 218].

Given the predominantly restrictive rules of logic that guide computer programming and hypertext markup, it would appear that XML schema would exemplify a formalistic genre process not unlike the single word format intended to clarify patient histories at the veterinary college. Everyone wants a more logical genre, correct? Yet Schryer found that not everyone at the college embraced the new system: older clinicians either left the clinic or "refused in less overt ways to cooperate with the implementation of the new system" [7, p. 215].

[1] I thank John Killoran for relating Dr. Schryer's name to me.

Defining the typology, ordering, and arrangement of how an organization's documents are to be structured and managed potentially involves debate and discussion as to documentation standards, organizational identity, bureaucratic roles, and hard to quantify things like "tradition."

I myself encountered this type of resistance when working on the surgery dictation project. One issue involved the job security of an editor who had manually performed tasks that were to essentially be taken over by the new opnote system. Her fears could have been articulated as "I'm not sure how this system will affect my social interaction and status within this organization." To ameliorate these fears, I made every effort to get this editor involved as the expert user in this system, a task that she reluctantly embraced. Not a day passed without her filing strong complaints to supervisors about system glitches and me. Indeed, the system had glitches, yet never once did the administration reconsider implementing the project. Bureaucratic power has a way of imposing systems on people whether they want it or not.

My experience and the work of Catherine Schryer resonates with Langdon Winner's theory of technology and human interaction. Winner writes that "as technologies are being built and put to use, significant alterations in patterns of human activity and human institutions are already taking place. . . . The construction of a technical system that involves human beings as operating parts brings a reconstruction of social roles and relationships" [8, p. 11]. Winner is not a technological determinist: he emphasizes that it is the interaction of human will and technologies that produce widespread effects. XML technologies will affect the developers of documents in profound ways, leading some to question their own social status within organizations. If widely adopted, it will socialize people differently and into different tasks, but not in the way that commentators currently think. Many have stated that technical writers with XML will no longer have to worry about the technical aspects of document design: they need only worry about content [5, pp. 97-99]. Ann Rockley furthers this line of thought when she states:

> In the past, technical communicators tended to create a 'document,' most often working on their own to create that document. Sometimes a team of writers created a document suite. With single sourcing . . . someone may be responsible for writing the core information (the information that is reused) while others are responsible for identifying how the information set needed for a specific solution differs from the core and adding information that covers those differences. Or it may mean that a number of writers work on different aspects of the core and work together to ensure that all the information is integrated [2, p. 192].

But what constitutes "core" information? That any successful writer must be aware of the larger context of information reception has long been a central teaching of technical communication [9, pp. 121-142]. Moreover, this vision of single source documentation suggests that XML can move contextualization

strategies out of the processes of rhetorical invention, arrangement, and style and combine them with the more mechanistic processes of memory and delivery. I suggest that such communication processes cannot be solved simply through mechanistic means. Genre can be a contestable issue for an organization, shaping human interaction as it is in turn shaped by the people who use it [7, pp. 208-209]. XML will pose new theoretical issues in technical communication about genre by foregrounding the integration of genre with technological systems, much as XML will more tightly integrate computational and documentation skills.

EFFECTS ON THE PRACTICE OF SCHOLARSHIP

XML has already shaped scholarly research in the sciences. A number of specialized markup languages for chemistry and physics have already taken a prominent role in helping researchers exchange and validate findings. XML makes it easier to exchange and include bibliographic information that conforms to specific publication formats. XML also allows scholars to embed tabular data into an article that can be exported immediately into a spreadsheet or statistics program so that researchers can validate data and test out alternative possibilities. In fact, XML would enable electronic documents that allow other scholars to perform such operations right on the article itself.

DocBook is a DTD designed for the technical communication field that allows authors to develop content, indexes, glossaries, figures, and other components using a standard framework that facilitates and expedites presentation of content across a variety of mediums [10, p. xviii]. The flexible capabilities for data and prose modeling make XML capable of testing usability methodologies by allowing researchers to code XML documents according to linguistic frameworks in usability theory. For example, one such test might code an XML document according to Hans van der Meij's IST manual information types: <conceptual>, <coordinative>, <constructive>, and <corrective> [11, p. 214].[2] An ISTE manual in XML provides a working model to operationalize these concepts.

Alternatively, XML allows researchers to provide real-time updates of projects in progress. With XML, the traditional collaborative article or book would not be produced from cutting and pasting large passages of text. Nor would it involve one or more HTML links to another document. Instead, an article could be modularized and produced dynamically according to the particular items that a reader wants to search. Rather than read one entire paper, academics will quickly be able to compile a "hybrid" paper drawing chunks of text from multiple articles. But who authors a hybrid paper? Because hybrid texts complicate notions of authorship, they may also complicate the current system for rewards and incentives in academic publishing. XML makes it easier to embed another author's

[2] I am indebted to James Stratman for this suggestion.

text onto a page not as something cut and pasted but as "real" data. DTDs and XSL style sheets further complicate matters because they are not content in and of themselves, but what exactly they are in legal terms is still not clear. These XML documents take a lot of time and energy to develop and often embed programming logic with content. With the exception of open source projects, most source code is protected by copyright law, which is why Microsoft doesn't go around giving away its source code for Windows. But the XML stylesheet language XSL straddles the line between a "declarative" and "programming" language because in addition to formatting the document for presentation, it also permits the user to perform queries on data and map database fields from one format to another. According to Michael Floyd:

> Declarative languages let you describe something using a set of rules. The rules can be placed anywhere, but they usually say little about how actions should be executed, or in what order. Cascading Style Sheets is one example of a declarative language. XSL isn't a declarative language in the truest sense. An XSL style sheet is an XML document that conforms to a schema, and is therefore a well-defined markup language. The markup is used to describe programming constructs for loops, if/then tests, and so on [12, p. 52].

To what extent are XSL codes intellectual property? Can people charge money for their use? It would certainly be preposterous to argue that the first novelist owns that genre, and that today readers worldwide pay the descendents royalties. But can an organization own a schema, a DTD, and/or a stylesheet? These are the kinds of unresolved matters that could impact how technical communication scholars conduct their research. They are also worthy research issues for technical communication scholars in their own right.

TEACHING TECHNICAL COMMUNICATION

If XML technologies become commonplace, then they could impact not only Web communication but all forms of technical communication. This possibility may require a significant restructuring of technical communication curricula. The "digital divide" in technical communication classrooms between producing for print and producing for screen will have to be reconceptualized. Faculty will have to learn different domains of technical communication: those working with text will have to learn something about electronic media, and vice versa. That is because XML may become a standard data format for all technical documents in the future.

In addition, new courses will need to more directly address issues of accessibility and internationalization. Because XML facilitates content creation for multiple audiences, every rhetorical task will involve a conceptualization of many different groups with different capabilities. Any major Web design task will involve producing content for international audiences. XML allows one to provide

translations of content in one document that can be accessed according to country. XML also facilitates the development of documentation that can work equally well with those who have and do not have physical limitations. A fertile area of both pedagogical and theoretical engagement is how XML standards can be developed to meet the needs of disabled audiences. New kinds of usability measures and tests will need to be developed to assess these needs, both from pedagogical and theoretical perspectives.

Some of the new skills students must learn will draw not only from communication theory but from computer and information science. Programming concepts such as parent-child relationships, nodes, trees, objects, abstractions, classes, inheritance, and recursion will likely become part of the vocabulary of \ technical communicators working with XML. Lars Johnsen argues that the production of technical documents with XML will be based on "object-oriented" paradigms that have long been a part of computer programming [3, p. 59]. According to Johnsen, XML "inherently takes an object-oriented view of information. In XML, a document is a hierarchically structured tree of labeled content objects to which further information values are assigned" [3, p. 60]. For example, one aspect of object-oriented data is in the transportation of information from a single-plane or "flat" orientation to vertical "parent-child" relationships among data values. In the opnote example, <opnote> is a parent or "root" object, <name> is a child of <opnote>, and if we wanted, we could include <first>, <last>, and <middle> as child objects of <name>. Conversely, the HTML rendition is a flat view of the information, with no intrinsic hierarchical relation implied among the different data elements.

Writing with XML will also require modifications in how we teach students to conceptualize rhetorical issues of audience, exigence, usability, relationships between pictoral and textual information, navigability, and context. Much of these changes arise from the structural difference between XML and traditional documents produced electronically. Technical documents produced with MS-Word and QuarkXPress are typically used in ways that treat content development as a process of figuring out how to balance prose and graphical elements in a way best valued by an end-user of a product. Designers think about who their potential audiences are for the text, where to place navigational elements, and so forth. Management of documents is often left to programmers or database specialists who can write scripts to move content around different servers.

With XML, technical communicators must think more explicitly in data management terms. Content *granularity,* or the smallest usable unit of information [13, online], is a key concern of technical communication with XML. Granularity in prose documents is typically conceived in terms of headings, sections, paragraphs, and sentences. But XML introduces smaller granular types or "fields" (such as author, title, year of publication) that have fixed lengths and strict integrity checks that are required for many of XML's transformative capabilities. Again, XML essentially combines elements of prose documents with database records.

This mixture of technical communication information and data records is not really new with XML. Technical communicators have often been confronted with documents that contain mixtures of data records and prose content. It is just that their skills have been applied primarily to the presentation aspects of documentation rather than infrastructural management. However, technical communicators cannot easily ignore infrastructural factors with XML development. Take, for example, the document management system for surgery opnotes. Surgery dictations exemplify the kind of document best suited for XML codification. They contain record data (such as patient registration number, procedure name, patient name, etc.) that can easily be placed in a database, but they also contain prose text that may consist of subsections describing in narrative format the surgery procedure performed on the patient. A class assignment might require students to do the following:

- Link opnotes to patient admission forms by registration number;
- Format output for attending surgeons using the Web at the hospital;
- Format the same output for surgeons on-call using cellular telephones or medical personnel in emergency rescue helicopters;
- Develop a schema for validating opnotes that can be used to exchange patient data with other hospitals;
- Develop documentation in XML for utilization of the schema; and
- Develop a user interface for creating new opnotes from the Web.

What kinds of knowledge areas must students learn to complete this project? They must be able to master computing markup skills and probably some Perl or PHP programming as well. They must also learn something about the infrastructural issues of the Internet because they will need to resolve complications quickly. (In my opinion, too many technical communication programs shield students from the computing environment with third party development programs such as Dreamweaver and FrontPage. My experience is that students limited to these platforms confuse the actual operating environment with the tool itself.) As technical communicators, they must also learn issues related to audience (which specific surgeons, nurses, and residents will read the documents), context (what navigational elements will medical professionals need to easily search and access the documents across multiple media formats), usability (what kind of file naming conventions are best suited for the medical transcriptionists to locate and update dictations quickly), genre (how to create a DTD that everyone can accept), and so forth. The technical communication principle of this project is that these documents must be rhetorically effective, accessible, and delivered on-time if patient lives are to be saved. Students are using data management and computer programming skills to solve a *rhetorical* problem, not a *computer* problem, and it is also for this reason that technical communicators may be the best ones prepared for such projects.

CONCLUSION: WORDS OF CAUTION

I have argued that XML will require technical communicators to become more intimately knowledgeable about computing skills and environments because XML integrates different types of information for multiple audiences. I have suggested that many of the literacy skills that are germane to technical communication are equally central to the more technical skills that are part of XML development. While I am optimistic about the strengths of XML and how it might improve technical communication, I suggest that the field must proceed cautiously in considering this technology. One cause of concern is that XML has a fairly high learning curve. The data transformation codes at the heart of XSL are very similar to a programming language. The popularity of HTML is that it is easy, not to mention free and accessible worldwide. A technology that is hard to learn may go unused by the technical communication community.

Additionally, many of the standards for XSL, particularly the formatting capabilities, have not been formalized by the W3C. As such, not all browsers support XML processing. As of May 2002, IEG and NSG.2 support XML and XSL with varying results. Browser incompatibilities and bugs still exist. The most reliable way to write XML requires that all XSL transformations be done on the server to be universally accessible, taxing the server and negating XML's ability to distribute computing processes to the client machine. Finally, XML still lacks affordable development tools that are flexible and easy to learn. The complexity of organizing XML documents requires good development tools particularly for adoption by the technical communication community.

The consensus in the technical communication community is to take a wait-and-see attitude toward XML. In the business world, however, XML is frequently used for e-commerce, which might mean that whether technical communicators want it or not, they may have to develop documents compatible with it. Organizations moving to XML environments must be prepared to invest heavily in time, money, and education, with the expectation that over time the move will pay for itself. Adopting an XML based single source strategy is an undertaking that requires considerable planning and a willingness to sacrifice short-term gain for long-term benefits. Whatever the case, the technical communication and computing fields are becoming more intertwined, presenting new challenges and opportunities for practitioners and scholars alike, and further making apparent that a person who calls himself or herself a technical communicator must also be technical.

GLOSSARY OF TERMS

Java: A programming language expressly designed for use on the Internet. Java is typically used to build large scale interactive Web sites that require much data management.

CGI script: CGI (Common Gateway Interface) is a standardized specification for connecting external applications with Web servers. A CGI script "talks" to a Web server using this protocol.

PHP server side script: A script language and interpreter that typically processes information through the Web server, as opposed to JavaScript, which processes information in the Web browser.

lexia and nodes: Roland Barthes (1970) used this term to define the "units of reading" in a multivocal text. George Landow (1992) used the term to refer to "text chunks" in a hypertext document.

JavaScript: A language from Netscape that is coded in a Web page to perform image rollovers, drop down menus, cookie setting, and other functions.

W3C: World Wide Web Consortium, an organization that promotes standards for the Web.

SGML: (Standard Generalized Markup Language) is a standard for specifying structural and other semantic elements of a document. XML is derived from SGML.

object oriented: A programming style guided by describing objects and their properties rather than specifying procedures for objects to perform.

PDF: (Portable Document Format) preserves all the elements of a printed document and makes it available for viewing on the Web.

REFERENCES

1. G. Landow, *Hypertext: The Convergence of Contemporary Theory and Technology,* Johns Hopkins University Press, Baltimore, pp. 1-21, 1992.
2. A. Rockley, The Impact of Single Sourcing and Technology, *Technical Communication, 48*:2, pp. 189-193, 2001.
3. L. Johnsen, Object-orientation, Visual Language, and XML, *Technical Communication, 48*:1, pp. 59-65, 2001.
4. J. Gumperz, *Discourse Strategies,* Addison-Wesley, Reading, Massachusetts, pp. 130-152, 1982.
5. K. Dick, *XML: A Manager's Guide,* Addison-Wesley, Reading, Massachusetts, pp. 97-99, 2000.
6. C. Miller, Genre as a Social Action, *Quarterly Journal of Speech, 70,* pp. 151-167, 1984.
7. C. Schryer, Records as Genre, *Written Communication, 10*:2, pp. 200-234, 1993.
8. L. Winner, *The Whale and the Reactor,* University of Chicago Press, Chicago, pp. 3-18, 1986.
9. L. Flower, Designing for a Reader, *Problem Solving Strategies for Writing,* Harcourt Brace, New York, pp. 121-142, 1986.
10. J. Brockmeier and K. Pritchard, *DocBook Publishing,* Prima Publishing, Roseville, California, pp. xvi-xxiv, 2001.
11. H. van der Meij, The ISTE Approach to Usability Testing, *IEEE Transactions on Professional Communication, 40*:3, pp. 203-223, 1997.
12. M. Floyd, Trouble with Transformations, *webtechniques,* pp. 52-55, July 2001.

13. R. Bourret, online at http://www.rpbourret.com/xml/XMLAndDatabases.htm (date accessed: 6-13-2001).

Other Articles On Communication By This Author

Sapienza, F., Nurturing Translocal Communication: Russian Immigrants on the World Wide Web, *Technical Communication, 48*:4, 2001.

Sapienza, F., Communal Ethos on a Russian Emigre Web Site, *Javnost: Journal of the European Institute for Communication and Culture, 6*:4, pp. 39-52, 1999.

Direct reprint requests to:

Filipp Sapienza
Department of English
102B Plaza Building
University of Colorado–Denver
Denver, CO 80217