How Practical is Practical SGML?

Robert J. Glushko, Passage Systems, Inc.

Five years ago the first edition of Eric van Herwijnen's *Practical SGML* was for me and many other people a Rosetta Stone that enabled us to understand SGML syntax. I eagerly looked forward to the second edition because today my business is "practical SGML" — helping organizations use SGML as a means for managing information in non-proprietary ways so that the same content can be delivered in multiple formats or channels, such as CD-ROM, the World Wide Web, print, or Braille. But there is a paradox to my review — the second edition is significantly better than the first in many respects, but feels much less practical than the first edition did. The world has changed more dramatically than the author's perspective on SGML has, and *Practical SGML*, with its strong focus on SGML syntax, no longer is the best book to start with for someone new to SGML.

An Introduction to SGML

SGML, the Standard Generalized Markup Language, is an international standard (ISO 8879) that defines a syntax for the character representation of structured information, in contrast to the proprietary binary formats of word processors. An SGML document contains the textual content (the words) along with syntactic markers (the "markup" or "tags") that indicate the boundaries and relationships among the information structures (the "elements") that contain the content (chapters, titles, paragraphs, glossary terms, cross references, and so on).

Because different organizations produce different types of publications, no single set of elements can describe what is important in all of them. Product manuals, catalogs, policies and procedures, directories, and textbooks are examples of publication types; each has distinctive kinds of information. SGML defines the syntax for specifying whatever set of elements and their relationships is needed to meet any requirements for describing information. SGML is thus not a specific markup language, but a Standard and Generalized way to create Markup Languages.

Each markup language defined using SGML is called a document type, whose formal definition in SGML syntax is the Document Type Definition or DTD. One of the best known document types is HyperText Markup Language or HTML, the set of elements used for creating documents for the Internet's World Wide Web. Other well known SGML document types are DocBook, used by many computer hardware and software companies for technical documentation, and Pinnacles, used by semiconductor firms for component datasheets. The formal definition of document types in SGML allows documents to be interpreted and validated according to the formal definition; thus, the specification of a document type not only tells you how to put together a document, it also allows you verify that you have done so correctly.

SGML is a powerful way of thinking about information management and publishing. SGML information can be read by people, sent to any computer, and used by any application that can use a DTD to identify the elements. As a result, SGML is being widely adopted by companies seeking more flexible use and reuse of the information they create.

A Look Back at *Practical SGML*, 1st Edition

In 1990 the "SGML business" was far different and much less mature than it is today. Compliance with the Defense Department's CALS initiative, which required the use of a specific SGML document type for information interchange, was a major driver for SGML adoption. HTML and the World Wide Web did not exist, and no computer hardware or software vendor had yet delivered SGML-based online documentation. Pioneers in these

early days tried to learn SGML from the *SGML Handbook* (Goldfarb, 1990).

From the minute it was published, the *Handbook* was the authoritative book on the SGML standard. It is largely the text of the SGML standard (ISO 8879) annotated by its editor, Charles Goldfarb. Every serious SGML project must have someone who is comfortable with the *Handbook*. But just as the Oxford English Dictionary is the definitive dictionary of the English language and may not be very useful as a guide for new speakers of English, the Handbook is an intimidating starting point for most people trying to learn SGML.

In this context, when the first edition of Eric van Herwijnen's *Practical SGML* (1990) appeared, it seemed that I'd found the Rosetta Stone for SGML. The book was primarily a feature-by-feature exposition of SGML syntax, but its presentation was far less formal and abstract than the *Handbook*'s. *Practical SGML* also briefly addressed authoring, document analysis, developing Document Type Definitions (DTDs), and other topics outside the scope of the ISO 8879 standard but which I needed to understand to make sense of SGML. Thousands of other people must have felt the same way about the first edition. Within a few years there were 10,000 copies in print, the book had been translated into several languages, and it had become the most recommended "starter text" for SGML.

In 1992 I co-founded Passage Systems, a consulting, software, and data conversion services company that helps companies make the "passage" from print to online publishing. Passage Systems specializes in "practical" SGML — using SGML, not as an end in itself, but as a means for managing information in non-proprietary ways so that the same content can be delivered in multiple formats or channels, such as CD-ROM, the World Wide Web, print, or Braille. Our trainers and consultants have introduced SGML to hundreds, maybe thousands of people, including most of our own employees. We own many copies of the *SGML Handbook* and many copies of *Practical SGML*.

Review of the 2nd Edition

In 1995 I eagerly purchased the second edition of *Practical SGML*, expecting it to be as important a book for the new era of SGML as the first edition had been in the initial era. In the preface van Herwijnen sets the unequivocal target for *Practical SGML* "to withstand the test of time and make it the definitive introductory book about SGML" (p. xvi).

The second edition has many strengths. It represents a substantial rewriting of the first edition. While it remains largely a simplified introduction to SGML syntax, the second edition is kinder and gentler than the first edition. It contains dozens of exercises that help readers verify their understanding as they work through the book. Icons in the margin, labeled as "opinions," "advice," or "warnings," highlight information to which the author wants the reader to pay special attention.

Some topics critical for newcomers to SGML are covered more thoroughly, especially document analysis and DTD construction. Some advanced topics from the first edition have been treated more lightly or eliminated. The book's thorough glossary and index make it an excellent quick reference.

The book, of course, is not flawless. *Practical SGML* still has a distinct bias toward programmers, which the author intends, but which makes it less readable for writers and managers. One manifestation of this bias is the recommendation to explain the meaning of each element as comments in the computer-readable part of the DTD (p. 108-109). This puts important information where it would be found only by SGML syntax experts. It would be more useful to explain the document type in a separate document written for authors, trainers, managers, or others with a stake in the analysis. Mixing this design rationale and usage information with its syntactic encoding makes it less accessible.

Practical SGML's assertion that "document analysis and writing DTDs are very akin to programming" (p. xvi) overemphasizes the last stage of translating a document type into SGML syntax. It glosses over the effort and iteration involved in eliciting information requirements, reaching a consensus among the stakeholders, and making the DTD compatible with the capabilities and processes of the organization that will use it. This more complex perspective is comprehensively treated in Maler and El Andaloussi's *Developing SGML DTDs* (1996), which is likely to become the standard reference.

Another significant weakness of the book is its lack

of discussion of how to use SGML to represent tables. Tables occur frequently in publications and there are numerous tables in *Practical SGML*, but there is absolutely no discussion of how to use SGML to model tables. SGML does not prescribe any particular way to represent tables, and while the CALS table model is often used, its two-dimensional geometric approach is too weak to represent many types of commonly occurring tables (Thompson, 1995). It would be useful pedagogically in an introductory book to contrast geometric markup for tables with more content-based alternatives.

The author has an occasional bad habit of letting his expert knowledge of SGML syntax overcome his responsibility not to lead SGML beginners into temptation:

- "Although SGML technically allows you to make these changes [to the reserved names used in the reference concrete syntax], I have personally never had the need to do so. But if you want to... (p. 142)"
- "My recommendation is not to use minimization, but if you absolutely want to, this section tells you how to do it (p. 152)".

"The context in which any treatment of SGML must be evaluated today has profoundly changed."

Perhaps because they have a certain cleverness to them that appeals to the programmer in van Herwijnen, *Practical SGML* has lengthy descriptions of SGML minimization features (OMITTAG, SHORTTAG, and DATATAG). These were invented to reduce typing when SGML markup was entered "by hand", but SHORTTAG and DATATAG are almost never used today. OMITTAG is used by SGML experts (so they can quickly modify SGML documents with text editors like vi, for example), but OMITTAG should never be used by SGML beginners. Beginners should create SGML using a "native" SGML editor or by running a program that converts word processor source files.

A Paradox

When I itemize the strengths and weaknesses of the second edition and compare it to the first, I can convince myself that the second is significantly better. But the second edition feels much less practical than the first one did for me five years ago, and it does not achieve the ambitious goals the author sets for it. While I recommend *Practical SGML* for someone who needs to learn SGML syntax, I can no longer recommend it as the book from which to learn about SGML.

If *Practical SGML* does not fully live up to its title, it is because **the context in which any treatment of SGML must be evaluated today has profoundly changed.** The first edition appeared when many people were learning SGML because they had to, in order to comply with CALS or a similar contractual mandate to create SGML documents for interchange, not for online delivery. Many of these people were programmers or had a programming or text processing background. In this context, learning SGML meant learning its syntax, so *Practical SGML*'s more accessible treatment of SGML syntax could be heralded as a practical contribution.

Today, most people who hear about SGML have more discretion about whether to use it, most of them are not programmers, and the ubiquitous hype surrounding the World Wide Web has already exposed them to text markup via HTML. For a book about SGML to be practical today, it is no longer enough that it be easier to read than the *SGML Handbook*. A practical book must situate and justify SGML with respect to other formats for information management and delivery, especially HTML on the World Wide Web. It must do so for writers and production personnel rather than for programmers, and it must do so in an end-to-end organizational context. These are harder challenges and they impose a higher threshold for making the claim of "practical" in a book's title.

SGML {and, vs} HTML

The first chapter of *Practical SGML* makes the case for SGML, but I would have made a stronger one. The relentless marketing pace of new releases of word processing software, adding new features at the cost of format incompatibility, poses a continuous threat to the longevity of information. Many alternatives exist for delivering information electronically on CD-ROM, on the Internet, and in embed-

ded applications, but each may require different markup formats. Even if you don't author using SGML, "thinking SGML" by authoring in a structure-aware manner to facilitate conversion of your most important information to SGML is a good defense against these proprietary pressures. It is with respect to HTML and the World Wide Web that these pressures are most intense. It is a measure of the explosive growth of the Web in the last year that *Practical SGML*, published late in 1994, completely fails to mention it. (Of course, this means that *Practical SGML* was being written even earlier, so I do not intend it as a criticism of the author that his crystal ball didn't predict the Web phenomenon).

Much of the increased attention to SGML in the past few years is directly due to the hype surrounding HTML and the Web, but the relationship between HTML and SGML is often misunderstood. HTML is properly described as an "application" of SGML, not a "subset" or "descendant" and certainly not a "replacement" for it, although I've heard all three claims. Much of the great appeal of HTML is its simplicity as a small set of elements for headings, lists, and links, which has made it easy for many software firms to create programs that convert word processing files to HTML or "browsers" that interpret the HTML elements as formatting instructions. The HTML elements and the rules that specify their valid use are documented in a DTD maintained by the World Wide Web Consortium.

Unfortunately, with the exception of programs customized for handling HTML developed by vendors of SGML technology, most HTML technology doesn't treat it as an application of SGML. These tools do not parse their HTML input or output to ensure it is valid with respect to the HTML DTD. Worse yet, some vendors have unilaterally added elements to HTML that only their browsers understand. This is a well-motivated but short-sighted attempt to give Web publishers more flexibility and control over document appearance. It would be far better if all Web browsing technology made use of arbitrary SGML so that any document could use any DTD and the browser could interpret the elements appropriately.

Until this happens, however, the net result of unvalidated HTML and proprietary extensions is to limit the interoperability and technology independence of Web documents. Users can get locked into whichever "dialect" of HTML is most successfully promoted by a browser vendor.

It is easy to get on the Web using HTML in a way that misses the point of its SGML foundations. I believe, however, that except for individuals who publish a small number of Web pages, only an SGML-centric strategy is sustainable after the initial excitement and heroic "hand-crafting" of the first Web publication. An organization publishing large amounts of information on the Web, and for whom timeliness, consistency, and reuse are important goals, needs an SGML-centric approach. Such an approach treats HTML, in whatever proprietary dialects are necessary to exploit the formatting characteristics of different browsers, as output formats created by "down-translation" from SGML. In the future revealed by my crystal ball, when all Web browsers can make full use of arbitrary SGML, no huge legacy of HTML-tagged information would have to be converted.

The SGML can be created using an SGML editor or converted from word processing source files, but in either case it should be SGML according to a document model that better meets the needs of your organization, not HTML, which is too simple and format-oriented to capture the complete structural and content model of your information. The HTML generated from SGML can also be automatically modularized, more consistently and completely linked, indexed by content type, and more efficiently maintained than HTML created by hand or from word processing files without an intermediate SGML representation (Suttor, 1996).

An End-to-end Organizational Perspective on SGML

In addition to emphasizing the syntax of SGML over its implementation, *Practical SGML* is aimed at individuals, which is unfortunate because individuals generally adopt SGML when their organizations do. The ideal result of an SGML implementation is a coherent end-to end system: an integrated set of authoring, conversion, validation, indexing, delivery, and database management software that enables an organization to meet all its requirements for creating, managing, and disseminating information. In such a system information from different authors and sources might be created with different tools, but after coming together as SGML it can be reassembled or linked to create the completed document for delivery to a customer. A technological perspective on

SGML emphasizes the mechanisms for validation at tool boundaries, for shared development, for configuration management, and for version control that enables the numerous pieces of information to come together. But this technological perspective is insufficient to ensure a successful adoption of SGML.

Adopting SGML may require learning new syntax and acquiring new software, but more important to successful adoption is institutionalizing a conceptual shift in focus from fixed printed publications to "information repositories" or "document databases". This new perspective on information management and delivery always involves paying greater attention to the information requirements of customers and users. Changes to authoring and production processes are often required when an organization recognizes the benefits of structural encoding and reuse, partly to overcome years of using proprietary WYSIWYG word processing software to make publications look a particular way for a single purpose. (A case study of SGML adoption written from this organizational and process perspective is described by Glushko and Kershner, 1993).

This broader perspective requires that the organization carefully assess the impact of the end-to-end system on the people who work with it. What are the costs and benefits? Are these costs one time, incurred only during the transition to the new system, or are they recurring? Are the existing skills of the people involved sufficient for them to carry out their new tasks, or do they need additional training? These questions have nothing to do with SGML syntax, but if they are not answered an SGML project will fail.

Conclusion

SGML enables an attractive and sustainable vision of information management, but achieving it takes effort and a long-term perspective. It can be hard to resist the seduction of HTML and the World Wide Web, especially when SGML and HTML share the same syntax and the latter seems so much easier to use than the former. Thus it is essential that someone who wants to learn about SGML does so from a book that isn't so focused on syntax as is *Practical SGML*. A book that fits this requirement is *ABCD...SGML* by Liora Alschuler (1995), which is an excellent first book for managers and writers because it deals with issues and case studies. *Practical SGML* makes a good second book on SGML from which to learn syntax after the idea of SGML is firmly understood. The ideal book would explain the ideas of SGML and its syntax at the same time, but this book does not yet exist.

Two very useful sources of information about SGML on the World Wide Web are the "SGML Web Page" (http://www.sil.org/sgml/sgml/html) and the site developed by SGML Open, a consortium of SGML vendors (http://www.sgmlopen.org).

References

Alschuler, L. (1995). *ABCD...SGML*. London: International Thompson Computer Press.

Glushko, R., and Kershner, K. (1993). Silicon Graphics' IRIS InSight: An SGML success story. *Technical Communication*, 40 (3), 394-402.

Goldfarb, C. (1990). *The SGML Handbook*. Oxford: Clarendon Press.

Maler, E., and El Andaloussi, J. (1996). *Developing SGML DTDs*. Upper Saddle River, NJ: Prentice Hall PTR.

Suttor, J. (1996). HTML and Internet Publishing. Course Notes. Alexandria VA: Graphics Communication Association.

Thompson, M. (1995). A tables manifesto. *Proceedings of SGML'95*, pp. 411-414. alexandria, VA: Graphics Communication Association.

Van Herwijnen, E. (1990). *Practical SGML* (1st Edition). Boston: Kluwer Academic Publishers.

Van Herwijnen, E. (1994). *Practical SGML* (2nd Edition). Boston: Kluwer Academic Publishers.