

Developing Information for Multiple Formats: You CAN Get There from Here

Patricia G. Moell and Helen F. Weeks

This paper describes the experiences of SAS Institute Inc. in developing single-source software documentation for presentation in multiple formats. The project is an ongoing team effort from all areas of the Publications Division. Our main goal is to develop online and hardcopy reference documentation. Toward this end, we set goals of using single-source files, reusing information, and tracking all information chunks and the relationships among them. To accomplish these goals we had to make decisions about the tools we are going to use, what information we are going to include, how we will design and present the modular information, linking and indexing strategies, and testing. This paper discusses the choices we made in light of our goals.

GOALS FOR DEVELOPING INFORMATION FOR MULTIPLE FORMATS

In the next major release of the SAS[®] System, we plan to deliver reference information in both hardcopy and online formats. For the online format, we have determined that our customers need fully hyperlinked, modular information in native help viewing environments such as the Windows Help system. SAS software runs on many operating systems including Windows, OS/2, Macintosh, a variety of UNIX systems, and (the real challenge!) mainframe operating systems such as MVS and VMS.

To meet the needs of our product and our users and to continue operating without doubling our staff, we had to develop a method for authoring information once and reusing it. From a single source of information, we plan to produce

- online help information targeted for topic-based viewers such as Windows Help
- online information for book delivery systems similar to IBM's BookManager
- online information on the World Wide Web
- hardcopy reference books
- hardcopy quick reference books that contain selected portions of the full reference.

In order to produce so much from a single source, we quickly determined that we need to author information in a portable format such as SGML, store small fragments of information in an SGML-intelligent document management system, and develop automated processes to assemble information sets (help libraries and books) and convert these information sets to the final delivery formats.

Also, we determined that we need to author information in such a way that it could serve dual purposes: hardcopy and online documentation. This is a primary goal as we develop information, and it requires a lot of planning.

DECIDING ON A TOOLSET

At the start of our evaluation project, we developed a list of criteria that served as the minimum requirements for the tools that we evaluated. The tools must

- run on the HP-UX operating system

- be in a production state (that is, not beta)
- recognize and support SGML
- allow for interface customizations
- support batch processing environments
- handle multiple file formats.

We then organized teams to review certain functional capabilities in the various products that met our initial criteria. We attended demos by the vendors to see how the products worked and how they could fit into the needs of our division. The teams evaluated products in the following areas:

- authoring and editing features
- formatting capabilities
- conversion methods
- performance
- customization features
- document management.

Selecting a Tool for Authoring and Formatting

With the dual objective to deliver hardcopy and online documentation, we required a mark-up language that could easily convert to the various host viewer languages where the SAS System is available. We decided on SGML because of its ability of handling multiple file formats — its tags are based on content rather than formatting. Another benefit is that SGML is nonproprietary.

In SGML, a DTD (document type definition) defines tags to use within a document. A writer must follow the rules defined in the DTD for a document to parse correctly. It would be cumbersome for writers to keep up with these rules in an ASCII file. There are few authoring tools on the market that have become SGML-friendly, allowing writers to select from a list of valid tags (based on the cursor location) so that they do not create context errors. We selected Arbortext's ADEPT Editor for that reason as well as for the following other advantages:

- We can customize the interface by writing programs with the command language.
- We are satisfied with the formatting capabilities and performance that we could achieve via a FOSI (formatting output specification instance) through a product add-on for hardcopy documents.
- We are impressed by the interface-driven development environment for our DTD and FOSI developers.
- We are confident that our needs for online editing and review will be considered at Arbortext as a future enhancement to their product.

Finding a Conversion Tool

Because we have to convert our SGML source files to the various host viewer languages (for example, RTF and IPF), we needed a conversion tool that would simplify this process. We chose Exoterica's Omnimark language to handle the conversions. Omnimark recognizes the structure of SGML and allows you to write programs that have awareness of the current element's context. It also runs the files through its built-in SGML parser to make sure that the document is valid before it is converted.

Another conversion challenge is translating all of our existing documentation (hereafter referred to as "legacy information") from a formatting language into SGML. Using Omnimark and scripts, we have the capability to complete these translation tasks.

Choosing a Document Manager

Our search for a document management product stemmed from a need to create a document database library and then be able to individually access small chunks of text from any document. These chunks could be accessed and updated by many writers and also pulled into other documents for reuse. A document management system would also support version control and file security so that multiple writers could work on a document without overwriting each other's work.

We have not yet selected a document management system because most have lacked our minimum requirement of being a production product on our platform. We have looked at many document managers and plan to purchase one in the near future.

DECIDING WHAT INFORMATION TO INCLUDE

One of the first decisions we had to make was which kinds of information need to go online. Because part of our goal is to deliver reference information in an online format, we did a document analysis of our primary reference documentation. We found that most of our reference documentation contained more than dictionary-style reference entries; we found overview information, tutorial-like information, examples, and more. The resulting specifications were used to develop a series of DTDs with compatible components for a variety of information.

We decided to base the DTDs on a representative sample of our reference books. We used a top-down data modeling approach to deliver a set of limited, standard structural and content elements and a list of rules describing the behavior, sequence, and restrictions of each element.

We are also working on a basic approach to the documentation for each software product that we document. We are sketching rough plans for product documentation and are encouraging prototyping. For each product documented, we are developing an information specification, which includes

- Ž what types of information are needed (dictionary of features, concepts, quick reference, tasks, tutorials, examples, syntax, and so on)
- Ž for which platforms the information needs to be available (for example, UNIX, Windows, and so on)
- Ž what user assistance methods are available for each platform (which viewers, for example, provide balloon help, pop-up windows, bookmarks, animation, and so on) and which new user assistance methods we will need to develop
- Ž what components are needed for each information type

(copyright, TOC, browse sequence, keywords, index, and so on)

Ž whether legacy information needs to be converted to SGML

Ž how legacy information may need to be restructured.

Writers and editors are preparing prototypes for reference documentation for our next version of the software. As we test our prototypes against our DTDs, we will work to refine the DTDs to meet the needs of our information modules. We see this as an iterative process.

And because one of our multiple formats is hardcopy, we also are examining

Ž which kinds of information we need to develop for hardcopy only and how the DTDs can support these modules of information and the book framework

Ž how we can weave modular information together in a way that makes sense for a hardcopy book.

Once we knew the types of information that we wanted to present, we worked to design the specifics of this modular documentation.

DESIGNING MODULAR INFORMATION

Part of the challenge of porting the same information into multiple formats is arranging the information so that it is presentable and usable on each target platform. We found that the best way to do this is to break the information into self-contained modules—chunks of information that can stand on their own yet when presented together in sequence, work just as well. We realized that we no longer can predict the context in which a user will access the information we provide.

As part of creating modular information, we also tried to reduce the volume of information without detracting from its usability. By implementing documentation downsizing principles, we were able to cut the length of some topics considerably, an important objective for online presentation. Linking within hypertext documents also enables chunks to be smaller by eliminating the need to repeat information. You can link to background or detailed information as often as needed.

We designed our information on the premise that readers are using the documentation to help them accomplish a task. To that end, we separated the steps for completing a task from the background information surrounding the task.

PRESENTING THE INFORMATION

Based on document analysis, information specification analysis, prototyping, and usability testing, we considered the various methods for presenting our information. Although we have one source of information, we had to consider many different output methods: hardcopy books, several different online viewers, and other user assistance methods (for example, wizards and context-sensitive help such as tool tips and balloon help).

Design of the hardcopy and the online formats took place about the same time. We tried to maintain consistency in the designs wherever possible. However, this does not mean that we wanted the online information to simply be the hardcopy book represented online.

Online Presentation

We revisited our information specification to determine what user assistance methods were available to us for each information type and each platform. Methods varied from online viewers to wizards to context-sensitive help. The level of granularity of the information (that is, the size of the chunk) also had an impact on its presentation.

Viewer paradigms range from discrete topic-based viewers to viewers based on a book model. We followed each viewer's guidelines (when available) and did our own research to decide on the formatting for fonts, type size and leading, color, white space, body and list styles, and tables. We also decided how we are going to use pop-up windows, secondary windows, and task topic windows.

We considered which types of information work best in which type of user assistance method. For example, how-to information may work best in a task topic window on Windows and as a section with an ordered list in which each item is clearly delineated in BookManager. Information that appears as a table in hardcopy might need to surface online as a table; or it may be converted into a list or a series of links. The decision we made depended on the complexity of the information and what information the user ultimately needs to receive.

Hardcopy Presentation

Layout for the hardcopy version was based on document analysis, downsizing efforts, legibility research, and previous experience. We were given a management directive to present the information in a two-column format.

Examples

Figures 1 and 2 are prototype examples of the same information presented in WinHelp and in our hardcopy format:

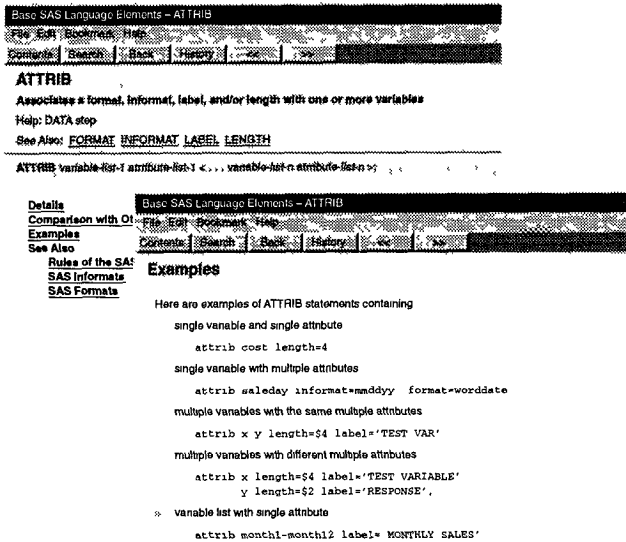


Figure 1 Reference Information in WinHelp

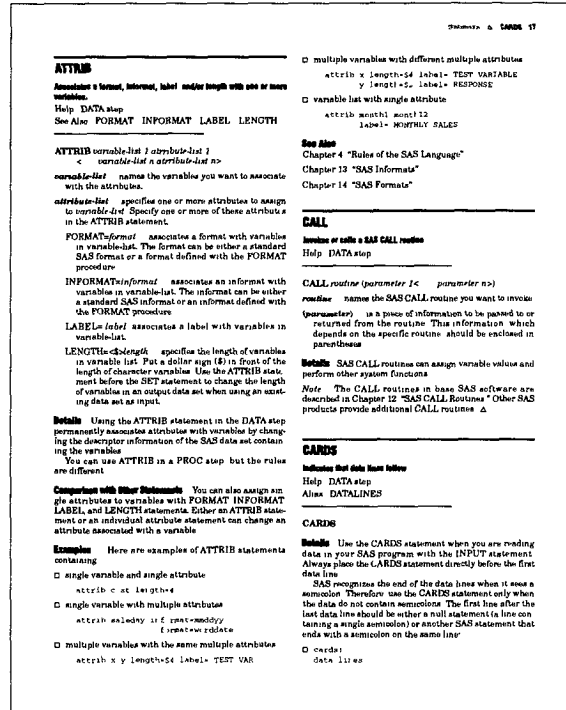


Figure 2 Reference Information in Hardcopy Format

Although the presentations are different, you will notice some consistency

- ⌘ Elements that are scannable and easy to pick out in the hardcopy are at the top level of the topic in WinHelp.
- ⌘ We used the same font to present code online and in the hardcopy.
- ⌘ We also used the same type styles for syntax elements in both versions.

CROSS-REFERENCING, LINKING, AND INDEXING

Generating Linked Cross-References

Cross-references are references to an item in text that is already identified with an SGML tag. These items include figures, screens, examples, headings, and tables. We are working on an application that will be running as the writer is authoring text and will compile lists of these items as they are entered into the text. Using the authoring interface, the writer can generate a linked cross-reference to an item. The software automatically enters link information into a link database. The link database contains an entry for every item that is either the launching point or the landing point for a link. This database is maintained separately from the document.

The actual text of a cross-reference depends on the type of item being referenced (figure, table, heading, and so on) and the medium for which the documentation is being generated (online or hardcopy). For example, a cross-reference in the hardcopy may say "See Figure 2.1 on page 53." The online cross-reference would be a link from the reference to the actual figure, and the page number would not be included.

Generating Subjective Links

Subjective links are links to a point in text that is not already associated with an SGML tag (such as a phrase in the middle of a paragraph). When the writer creates a subjective link, the software assigns identifiers to the launching and landing points for the link and updates the link database.

Generating Linked Indexes and Keyword Lists

In most cases, an online information database has an associated keyword list that allows the reader to search for topics using a specified keyword or string. In hardcopy information, this same function is performed by the index. Because we are using a single source for all of our hardcopy and online information, we need a way to produce both keyword lists and indexes, preferably without entering all of the index entries and keywords twice.

Some considerations that complicated our indexing decisions include

- reuse of modules. Because a module may be picked up and moved anywhere, it cannot be indexed according to information that may come before or after the module (the "context"). All index entries must deal only with the information in the module.
- granularity of modules. If the modules are chapter-sized, then you can start a page range at the beginning and end it later in the same module. Any particular paragraph can be indexed with regard to its context. If your modules are paragraph-sized and you want to reuse modules in different contexts, page ranges and context-relative index entries are not feasible.
- use of page ranges. SGML index tags cannot cross container boundaries. We cannot start a page range entry in one module and end it in another. Therefore, any implementation of page ranges must use nested modules. However, using page ranges results in a module that may not be completely reusable. But do we want a long string of numbers following an index entry in the hardcopy index? Do we want to give up page ranges in the hardcopy index?

Our in-house indexer will be entering index terms into modules as those modules are finished. We will index all modules independently, without regard to possible context, and we will not use page ranges. Our software will use these index entries to generate hardcopy indexes and online keyword lists.

TESTING

Because we are providing the information online as part of the software package, we need to provide testing at several levels: for the information module, at the concatenation of the information, at the integration of the document into the software, and across platforms at the viewer level.

At the module level, we are testing each information module to make sure that it meets the user's needs and that it conforms to our company standards for quality. Each module is edited according to a set of quality-check guidelines. For each module, we strive to make sure that

- the topic can stand alone
- the content is accurate
- the content is expressed concisely and logically with an appropriate combination of text and figures
- links to other modules make sense
- grammar and spelling are correct
- legal requirements, if any, are met
- user interface standards are met

At the concatenation level, we are testing the information modules in relation to one another. Do they work together in the way they were intended? For each set of modules, we strive to make sure that

- the links jump to the correct content (including the index)
- the information is where the user expects it to be
- all modules are in the correct relationship to one another
- all appropriate information is available and can be retrieved for the type of help or type of online documentation available.

At the integration level, we are testing to see whether the combination of components or modules that we already know work well together can work when combined with already existing online documentation and with the software it accompanies. Integration testing is specifically aimed at exposing the problems that occur when components are combined.

After the integration test, we are doing regression testing to verify that modifications have not caused unintended side effects or to verify that a modified system still meets requirements. Then we perform acceptance testing to verify that the integration works with the software as designed.

At the cross-platform level, we are testing to see whether there are problems with the conversion from SGML to the language for the native viewer. We test using a viewer on each platform for which our software ships to see

- what problems occur in the conversion results
- whether the help or online documentation functions as planned for the specific viewer
- what idiosyncrasies arise for each platform viewer and/or language that will require modifying the original writing of the information modules.

Because the purpose of testing is to uncover problems that render the online documentation less than useful, we have resources on hand at the various stages of testing, including editors, writers, and programmers, to fix the problems that testing uncovers.

CONCLUSION

The job is not over for us. We are just beginning to see the results of all our planning and prototyping. We are still learning to use our new tools, deciding how to chunk and present certain types of information, and refining our testing processes. However, because of the effort that we made up front in setting goals, listing requirements for our toolset, and developing working models of our information set, we are now better prepared to provide our users with the information they want in the format they want.

SAS is a registered trademark of SAS Institute Inc. in the USA and other countries. " " indicates USA registration.

Other brand and product names are registered trademarks or trademarks of their respective companies.

The following people from SAS Institute's Publications Division contributed to this paper

- Jeanne Ferneyhough, technical writer
- Chris Hemedinger, technical writer
- Ginny Matsey, graphic designer
- Patricia Moell, editor
- Cindy Roposh, applications developer
- Helen Weeks, editor
- Holly Whittle, manager of Publications Technology Development Department

Many others throughout the Publications Division reviewed this paper for technical accuracy and consistency.

Publications Division
SAS Institute Inc.
SAS Campus Drive
Cary, NC 27513
919-677-8000