A HYPERTEXT-BASED DOCUMENTATION WORKBENCH FOR Ada-LANGUAGE SYSTEMS

BARRY P. FINKELSTEIN

PLANNING ANALYSIS CORPORATION

1 INTRODUCTION

1.1 Purpose

The purpose of this paper is to describe an approach to developing on-line technical documentation of Ada-language systems using hypertext technology. The use of hypertext enables the creation of documentation which is easily accessible by analysts, designers, and programmers. Since hypertext supports multiple access paths or user views, a single integrated set of documentation can serve a variety of purposes and audiences. It can support system enhancement and maintenance, training of technical support staff, and reuse of system components.

1.2 Contents

This paper describes the experience of Planning Analysis Corporation (PAC) in developing hypertext-based documentation for Ada-language systems. The paper addresses the following:

- Background discussion on documentation needs and challenges, and hypertext and its applicability.
- PAC's experience in developing a hypertext-based documentation Workbench for an Ada-language system and other software systems.
- Future directions in hypertext-based documentation for Ada-language systems.

2 BACKGROUND

2.1 Documentation Needs and Challenges

As developers of Ada-language systems prepare for the 21st century, they face several related challenges, all associated with effective documentation of increasingly complex systems. Technical documentation is required to support development of new systems, maintenance and enhancement of existing systems, and reuse of code.

Quality technical documentation is a vehicle for communication among system users, developers, and maintainers. It also aids communication among people working on separate projects, and thus can facilitate reuse of software components. As software systems become larger, more complex, and increasingly integrated, the need for effective documentation increases. Comprehensive documentation enables organizations to obtain maximum value from their information systems. It extends the life span of systems, and provides a basis for more effective maintenance and enhancement. Just as one would

not attempt to use or maintain a complex item of machinery, such as a car, without effective technical documentation, so too our complex information systems require quality technical documentation.

Maintenance activities typically consume over half, and as much as 80%, of the software budgets of organizations. One of the objectives of Ada has been to reduce total life cycle costs by reducing the costs associated with system maintenance and enhancement. Research has shown that more time is spent trying to understand the software than any other maintenance task. Effective, easily accessible documentation is intended to reduce the effort and cost associated with understanding the software.

Effective documentation reduces total life cycle costs by improving the maintenance and enhancement processes. Much like up-front analysis and design, documentation efforts require up-front investments in order to yield efficiencies later in the life cycle.

In many organizations, technical documentation of computer software is inadequate. Organizations committed to Ada are no exception in this area. Experience has demonstrated numerous obstacles to producing high quality technical documentation, including:

- The additional cost and time to produce
- The large volumes of paper required to document modern large-scale systems
- The difficulty accessing complex documentation and locating key specifications
- The difficulty maintaining documentation as the software evolves

As a result, many organizations are no more than minimally committed to trying to produce and maintain documentation.

Accurate, accessible, and up-to-date technical documentation is critical for Ada-language systems, to support full life cycle maintenance and software reuse, a key potential benefit of Ada.

2.2 Hypertext and Documentation

As described in the next section, Planning Analysis Corporation (PAC) has used hypertext technology to increase the effectiveness of documentation. Hypertext-based documentation can overcome, in whole or in part, the obstacles to successful documentation listed above.

Hypertext is a technology that enables us to link textual and graphical material to improve access to complex information sets. A hypertext system of documentation consists of items of information, called nodes, linked together with preprogrammed links. The nodes may consist of any type of information which can be represented as standard ASCII files. They can be specifications for computer programs, diagrams used to document software, data dictionary entries, screen and report layouts, and computer code itself. The links are key words, phrases, or ideas which tie the nodes together.

Using a single keystroke, a user can invoke a link and jump from one item of information to a related item. The user need not know anything about the organization of the material to access it easily and quickly. No complex search processes are required. Each user can follow his or her own access path through the material.

Hypertext is especially effective at supporting nonlinear access to information. Paper documentation, by contrast, is best suited to sequential access, although limited indexing or random access is available. When using a hypertext system, a user may read sequentially, but also may follow logical branches or jumps to related items of interest. The system keeps track of the branches taken, thus allowing the user to re-

trace his or her steps and not get lost.

Normally, a hypertext document is based on a hierarchical structure, with the material organized into major sections, much like the chapters of a book. Each section may be further subdivided as required. The lowest level subdivisions become the nodes of the hypertext network. Nodes are linked with other nodes to form a logical network within the overall hierarchy. The hierarchy provides a high level menu structure for user access, while the links provide users with the power to navigate throughout the material, following their own perspectives and pursuing their specific information needs.

The hypertext software used in PAC's projects operates on standard IBM-compatible microcomputers under the DOS operating system. The software can access any standard ASCII files and selected types of graphics. The software provides powerful tools for creating and organizing large numbers of links, thus allowing development of large hypertext networks.

Hypertext is well-suited to software documentation, offering the following benefits:

- It enables a user to move rapidly among the various items of documentation, e.g., from the data element dictionary to module or report specifications, following links based on a specific data element.
- It enables a maintenance programmer or analyst to quickly determine the components of a system potentially affected by a planned change.
- It allows linking of the documentation to the code itself, thus facilitating integrated maintenance of both code and documentation.

3 A CASE STUDY: A HYPERTEXT-BASED DOCUMENTATION WORKBENCH

3.1 Overview

Planning Analysis Corporation has developed a Hypertext-based Documentation Workbench to facilitate the creation and use of effective technical documentation. The Workbench has been applied to an Ada-language management information system as well as several other computer software systems. The product can be easily tailored to an organization's technical environment and specific needs for documentation. It accommodates virtually any types of documentation, including automatically produced documentation. It also conforms to any documentation and systems development life cycle and standards.

The Hypertext-based Workbench addresses several of the challenges associated with producing and using technical documentation listed above. By integrating the documentation and the computer code into a single on-line system, it eases access to the documentation. By providing all technical documentation in automated form, it eliminates the need for bulky, voluminous paper documentation. Through use of hypertext technology, the Workbench allows a user to follow links among specifications of all system components, along with the Ada source code. The Workbench accommodates Ada constructs, providing special documentation of software components in packages, and specifying the compilation order.

PAC's initial experience with hypertext documentation involved documenting a system for the U.S. Marine Corps. The project involved the Unit Diary System, a new system developed in Ada for IBM-compatible microcomputers, Zenith Z248s under DOS. The Ada system was developed largely by converting an older system written for IBM Series 1 computers. PAC was engaged to reverse engineer the system to produce effective technical documentation to support long-term maintenance. The system contains over 650 Ada modules and 40,000 lines of Ada source code. The bulk of the material in this pa-

per refers to the creation of hypertext documentation for this system.

3.2 Organization and Contents of the Hypertext Documentation

Figure 1 is a schematic representation of the organization and contents of the Documentation Workbench from the perspective of the user interface. As indicated, the Workbench integrates complete technical documentation in a single hypertext-based network. In addition, the system documentation is linked directly to the source code. Thus, a designer or programmer can peruse the specifications and then browse through the source code itself. By linking the Workbench with a standard PC-based text editor, the facility can provide a complete configuration management (CM) system that supports maintenance of software and documentation in a single system.

The user interfaces with the system through a hypertext browser, called PC-Hypertext. The browser is part of the complete set of software used to develop the hypertext network. Other elements of the hypertext software include a high level hypertext outliner, called Houdini, which is used to create and keep track of the links; and a series of utilities that link the material together and perform various other functions. The software, produced by MaxThink, Inc., is relatively inexpensive and is needed only by the developer of the documentation. The user needs access only to the browser which can be distributed freely.

Figure 2 is a view of the high level menu structure of the documentation workbench. Each of the eight choices shown on the right side of the menu represent the highest level topics or chapters of the documentation. Following from this high level topic menu, a user may explore system instructions, module descriptions, record definitions, screen definitions, disk file specifications, the data dictionary, packages, and the compilation order. In addition, the Find command shown at the bottom of the menu screen provides a comprehensive alphabetic index to all files or nodes in the system.

The high level menu shown in Figure 2 is intended to look like a book, with the title page on the left and the table of contents on the right. This high level topic menu may be further divided into additional high level menus. Figure 3 shows a menu

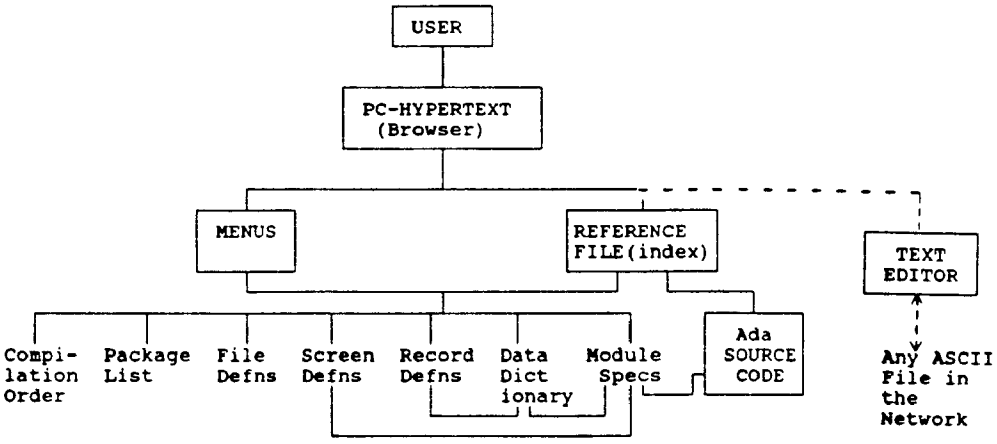

Figure 1.

USER INTERFACE TO Ada DOCUMENTATION WORKBENCH

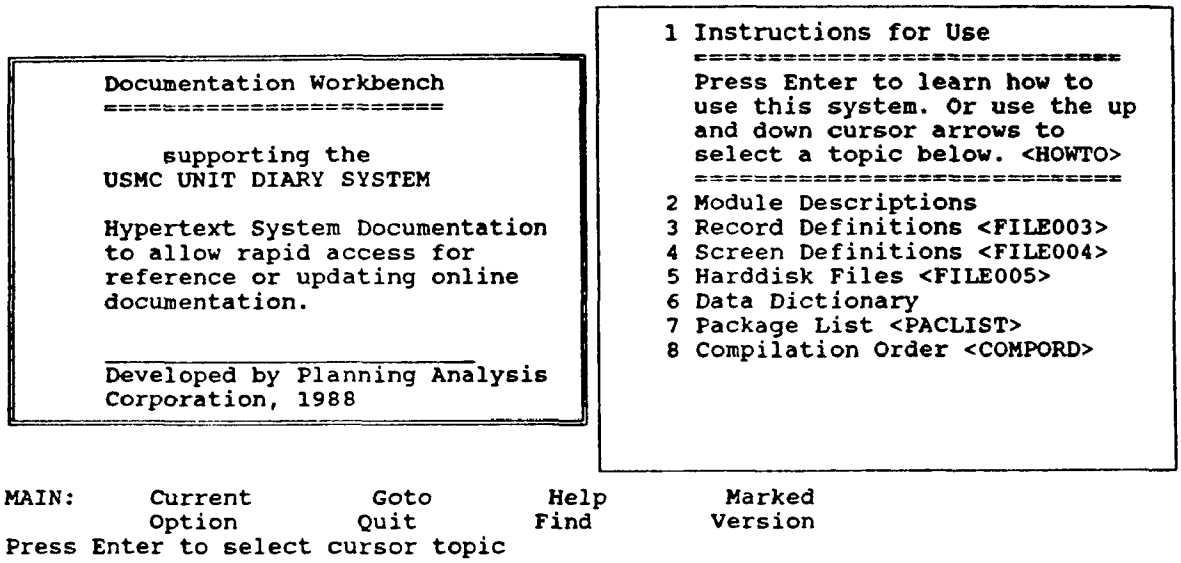

Figure 2.

High Level Menu Structure

at the next level. As shown, the higher level topic is presented on the left, with the breakdown into subtopics shown on the right. This gives readers a perspective on where they are in the hierarchy. Continuing to choose options on the menus, simply using the arrow keys, quickly provides access to specific files or nodes of the hypertext network. Each module specification, screen definition, record definition, etc., is a node.

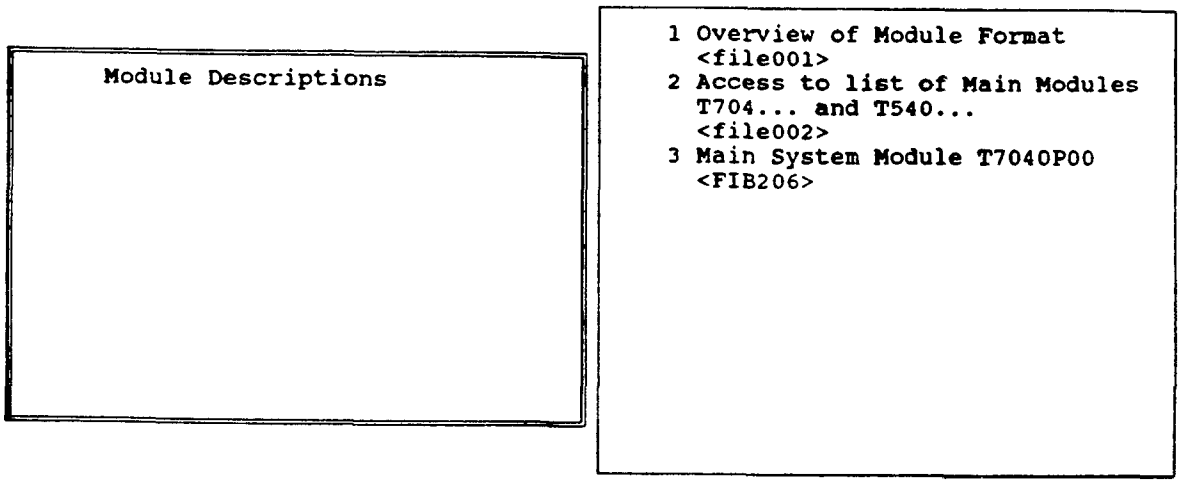

Figure 3.

Next Level Menu Structure

For the Marine Corps Unit Diary System, the technical documentation is structured around detailed module specifications. The system includes over 650 distinct modules of Ada code. A critical task in creating and assembling the module specifications involved organizing the module specifications in an orderly fashion. PAC used several automated tools, including an Ada code parser, to devise and maintain a logical "call tree", which tracks the structure of which modules invoke other modules. The module specifications are organized in the hypertext network following the logical call structure of the system. Thus, the system shows an analyst or programmer the modules that call or are called by each other module. Given the high level of modularity and reuse of Ada modules within the system, the ability to discern the call structure is critical to effective maintenance.

The Workbench provides several diverse access paths to the technical specifications, including:

- Direct access through the menu to all items except the source code
- Access to the source code through the module specifications
- Access to screen specifications from module specifications
- Access to record definitions and module specifications through the data dictionary
- Access to all files through an alphabetic index
- Access to lower level modules through the call tree structure of module specifications

3.3 Design Considerations for Hypertext Documentation

In creating hypertext documentation for Ada-language or other systems, several factors must be considered.

First, the audience and purpose for the documentation must be analyzed. For the Marine Corps system discussed above, the primary audience was maintenance programmers who would enhance and maintain the programs. Accordingly, the hypertext documentation was structured around the module specifications. In another project we are beginning for the Environmental Protection Agency, the audience is systems analysts who will tailor the system to specific users through manipulation of data base parameters. In this application, the hypertext documentation will focus on the data base as the highest level organizing element.

Second, the level of detail for the documentation must be determined. To some extent, the appropriate level of detail is a function of the purpose of the documentation. To support program maintenance, detailed program specifications are normally required. To support analysis only, the documentation need not be so detailed. The required level of detail is also affected by other considerations, such as the extent to which the programs are self-documented through comments, logical structure, and naming conventions. The module specifications need not be so detailed if the source code communicates well.

Third, the documentation must be designed to support ongoing maintenance. As discussed in Section 4, an important consideration for the future is how to enable easy and cost-effective maintenance of technical documentation as software systems evolve. Some features of the Documentation Workbench aid in documentation maintenance. For example, the system can be linked online to an editor, so that with a single command, a user can invoke an editor and modify any elements in the system. Of course, this raises important issues about configuration control. The hypertext network also helps identify components of documentation which must be changed to reflect software changes.

Fourth, the documentation team must

determine the extent to which automated tools will be used and/or developed to support the documentation process. For the USMC Unit Diary System, PAC developed several automated tools to examine the Ada code and develop the structure of the call tree and also to generate shells of module specifications. We also used several utilities to keep track of files and data linkages. These tools supplemented the powerful hypertext tools used for the overall Workbench system. For other documentation projects, we have relied on automated documentation tools which were used by the software developers. Automated documentation was extracted and restructured to fit into the hypertext structure.

Fifth, the extent to which the software developers are to be involved in the documentation process must be determined. There is great value in involving the designers and programmers in the documentation process. They become committed to creating and using effective documentation. In addition, they know the system best, and can most efficiently develop documentation. However, it is a challenge to divert their time from the development effort to support documentation.

We have been able to form teams consisting of documentation analysts working with system developers. Through the use of hypertext, we have been able to provide easy mechanisms for developers to participate in reviewing and testing documentation as they continue the development effort. By constructing hypertext shells for documentation, we provide easy structures for programmers to create simple documentation which can then be edited by the documentation team. The designers and programmers do not have to worry about document structure or detailed editing; they simply develop technical notes which are then refined and linked into the entire system.

Sixth, the documentation must be designed to conform to applicable standards, e.g., under DoD-STD-2167A. In general, the Workbench can accommodate any standards, since any textual documentation can be represented in a file format accessible by the hypertext software used.

3.4 Applicability to Ada

Although PAC's Documentation Workbench is applicable to software written in any language, it is particularly useful for Ada, because of two inherent characteristics of Ada software engineering. The first is that Ada strongly encourages modularity. Typical Ada systems are created using numerous small modules packaged in a coherent structure. The 40,000 line system we documented using the Workbench contained over 650 modules. The hypertext-based documentation process allows us to effectively link together large numbers of small modules, typically organized around a logical call tree.

The second characteristic of Ada systems is the emphasis on reuse. The hypertext-based documentation facilitates identification of modules which are candidates for reuse. Such documentation tools will become an increasingly important element of reuse libraries. In addition, the hypertext documentation can easily be distributed in automated form, thus providing a vehicle for automated, easy-to-use catalogs of reusable software.

Although the Workbench can be used for existing or new systems, it is most effectively implemented in conjunction with the development of a new system. Procedures are then established so that documentation is created as the software is designed and developed. The documentation is automatically loaded into the Workbench as the software development proceeds. In fact, the Workbench is designed to draw as much as possible on any documentation automatically generated by today's software systems. For example, we have used the Workbench to capture automatically generated documentation produced by Documentor and IBM's CSP software products. In the future, the Work-

bench will be integrated with other facilities for automatic generation of documentation.

Through the use of hypertext technology, the Workbench provides easy access to technical documentation. Tailored access paths are provided for systems analysts, designers, and coders. Additional access paths are provided for helping identify modules for reuse.

4 FUTURE CONSIDERATIONS AND CHALLENGES

Perhaps the greatest challenge associated with high quality technical documentation involves maintenance and control of the documentation as the system evolves. Our Documentation Workbench aids in documentation maintenance to some extent, but more work is required in this area. Closely related is the integration of documentation and the Workbench into an effective configuration control process. PAC is exploring mechanisms for incorporating configuration control procedures and systems to track software and documentation changes. Mechanisms for version control are also important.

A second important consideration for the future is reducing the extensive manual effort associated with creating effective documentation. Our initial experiences in producing technical documentation were highly labor-intensive. In subsequent projects, the documentation effort has been reduced somewhat by integrating the documentation and development processes, relying on automated tools and documentors, and enforcing design and programming standards. However, it remains a labor-intensive process, and requires an organizational commitment beyond that of many development organizations.

Plans to expand and enhance the Documentation Workbench include:

- Increased integration with CASE tools and Ada-language software engineering environments.
- Application of the Workbench to software reuse libraries.
- Development of standard procedures for cost-effective integration of software documentation and development processes.
- Development of automated tools for cost-effective maintenance of technical documentation as software is modified.
- Use of the Workbench to capture metrics on how the documentation is used to support specific tasks, such as program maintenance.

PAC is currently experimenting with application of the Workbench to a COBOL-based system containing approximately 400,000 lines of code. This will demonstrate the viability of this concept for large-scale application software systems.

Our expectation is that the documentation tools and procedures we are now developing and perfecting will be increasingly valuable for Ada-language systems of the 21st century.