AN OPEN ARCHITECTURE CORPORATE ELECTRONIC PUBLISHING PLATFORM

Nils-Peter Nelson and Carmela L'Hommedieu

Nils-Peter Nelson *is a supervisor, and* ***Carmela L'Hommedieu*** *is a programmer in the Computing Technology Department of AT&T Bell Laboratories, Murray Hill, New Jersey. Mr. Nelson is responsible for supervising the development and maintenance of computer graphics and electronic publishing software for internal R&D users. He holds a B.S. in mathematics from City College of the City University of New York, and an M.S. in mathematics and computer science from Stevens Institute of Technology, Hoboken, New Jersey. He joined AT&T in 1966. Ms. L'Hommedieu joined AT&T Bell Laboratories in 1973 directly out of high school, first as a typist and then as a documentation specialist for the Computer Science Research Center. There she learned UNIX® text formatting and typesetting skills and developed the* `eqnchar`
(continued on page 110)

Corporate Electronic Publishing (CEP) encompasses more than the primitive capabilities of desktop publishing systems. It also includes networking and, in its optimal form, provides a platform for addressing a wide-ranging set of corporate documentation requirements. At AT&T Bell Laboratories we have evolved a platform for CEP consisting of AT&T's UNIX® operating system, the Datakit® virtual circuit switch, the Documenter's Workbench® and Graphics Workbench software, and the Adobe System, Inc. PostScript™ page description language. We cite real-life examples in which authors and printing facilities at geographically dispersed locations participated in cooperative efforts using the platform. We describe new work in progress that builds on this platform and addresses needs in illustration, photographic reproduction and plotting.

Introduction

Nearly every office worker in a large corporation is in the publishing business. Business letters, interoffice memoranda, news bulletins, meeting notices, technical papers, training manuals—all these can properly be considered part of the publishing world. Whether the medium of delivery is a computer terminal, personal computer, optical disk, sheet of paper, or bound book, the problem common to them all is the efficient distribution of visual information.

Computing has dramatically redefined the publishing world. Much of the recent publicity around this revolution concerns "desktop publishing." For a few thousand dollars—the price of a PC, laser printer and formatting software—anyone can go into the publishing business. Less talked about are the more complex needs of large corporations, needs that cannot be satisfied by standalone desktop systems. By Corporate Electronic Publishing (CEP) we mean not only the input and output devices and software necessary to produce documents, but also the *networking* that allows people and devices at

Panel 1. Terms and Acronyms in This Paper

CEP	Corporate Electronic Publishing
CGM	Computer Graphics Metafile
GIF	Graphics Interchange Format, a standard defining a mechanism for storing and transmitting raster-based graphics information
GWB	Graphics Workbench
LAN	local-area network
MIS	Management Information Systems
PSDN	Packet Switched Data Network
RS232	Electronic Industries Association (EIA) standard interface between a computer and peripheral device
VCS	Virtual Circuit Switch
WAN	wide-area network
WYSIWYG	What you see is what you get

different physical locations to work together.

One approach to CEP is the proprietary system. After thorough study, a single vendor is selected to satisfy the corporation's electronic publishing needs. There are many such offerings in the commercial market, but it is not our goal to examine or even list them here. The advantage of such systems is that responsibility for them is clearly defined: i.e., whatever goes wrong is the vendor's fault. But finding "the one system" may prove impossible. In the real world, one vendor may have an excellent scanning device, but a poor printer. Another has a user-friendly formatter but no way to tie into your network. Another is excellent for business letters, but cannot handle complex jobs involving equations, tables, or graphics. Experience proves that no one package can solve every publishing need. Thus, we want a platform that not only supports a variety of devices and software packages but also includes corporate networking capabilities.

At AT&T Bell Laboratories, we have evolved such a platform. It is based on the UNIX operating system, the Datakit VCS, the Documenter's Workbench and Graphics Workbench software, and the PostScript page description language. Since most structural elements of the platform serve our general computing environment as well, meeting our publishing needs becomes a small, incremental cost to our standard operation.

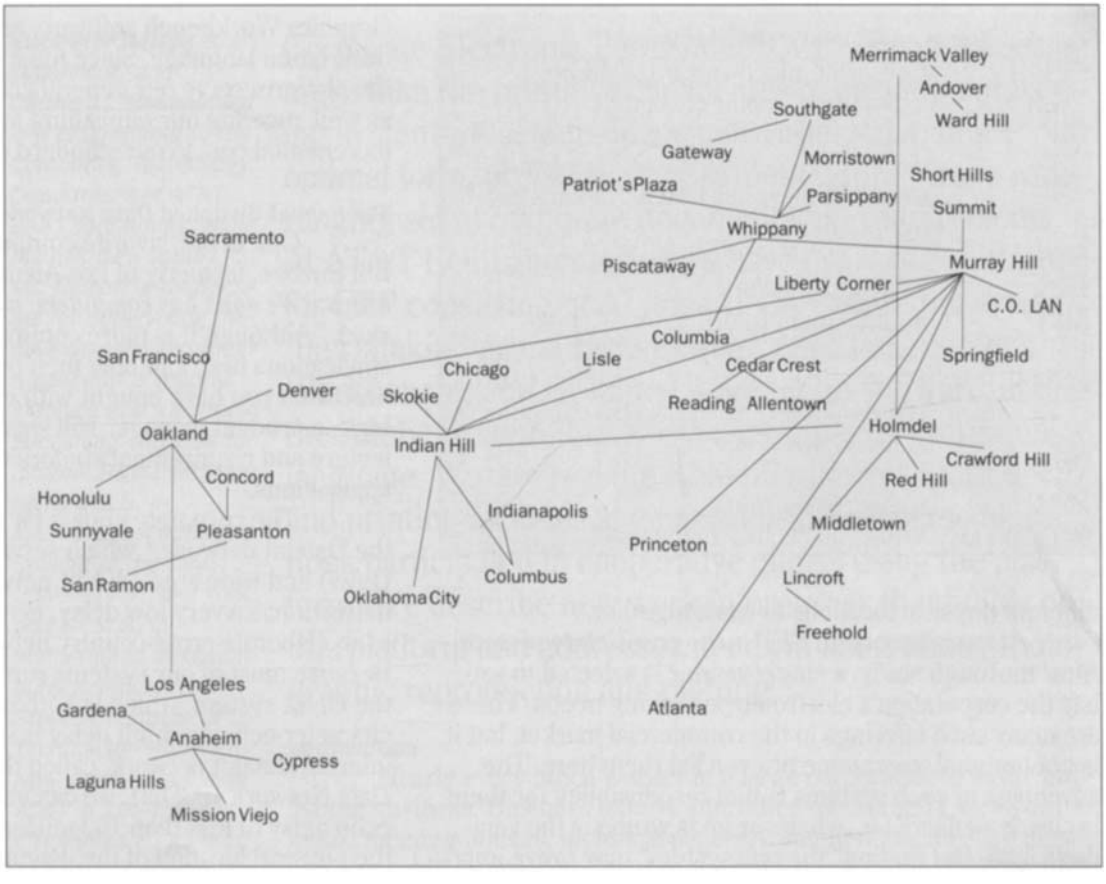
The Packet Switched Data Network

One of our favorite computing aphorisms is from Bill Buzbee, formerly of Los Alamos National Laboratories: *"We don't buy computers, we buy nodes on our network."* Although it is more common to select systems or applications first, and only then puzzle out how to connect what you have bought with everything else you have, a prudent manager will specify the network architecture and requirements before evaluating systems and applications.

The corporate choice for Bell Laboratories was the Datakit network,[1] which serves both local-area (LAN) and wide-area (WAN) networking needs. This network has a very low delay, even when daisy-chained in an elaborate cross-country network (see Figure 1). Because most of our systems run under various forms of the UNIX system, which is both interactive and character-echoing, high delay is intolerable. With our internal Datakit network, called the Packet Switched Data Network (PSDN), we can guarantee per character echo delay of less than 100 milliseconds, regardless of the physical location of the user and system. At each of our many physically separated locations there are one or more Datakit switch nodes, interconnected by dedicated phone lines, operating generally at 56 kilobits per second or by T1 (1.544 megabits per second).

PSDN is easily able to support high-speed host-to-host file transfers while simultaneously handling thousands of lower speed terminal-oriented virtual circuits which are created on an as-needed basis, but otherwise consume no bandwidth. The PSDN network is ubiquitous in Bell Laboratories, and supports the RS232 interface as well as higher speed multiplexors. Thus, in addition to attaching all our terminals, PCs, and work-

Figure 1. The packet switched data network.



stations to PSDN, we can install printers in any office, lab or hallway. This is an essential ingredient of our electronic publishing platform, since convenient access to hardcopy printers is a desirable feature of any publishing system. Within a given location, terminals, PCs, workstations, hosts and printers are all equal members of the network. Although this might be considered a "one tier" architecture, we usually describe it as "two-tiered" (see Figure 2) to distinguish user-owned systems from centralized services.

Host Systems and Software

It might seem obvious that AT&T Bell Laboratories employs the UNIX system for most of its internal computing needs, since the system originated there in the 1970s. But the reason for the popularity of the UNIX system within AT&T itself is not company loyalty or an ignorance of alternatives. Rather, the system is unique in that it runs on PCs, workstations, minicomputers, mainframes and even our one supercomputer, the CRAY X-MP. This compatibility between different systems with

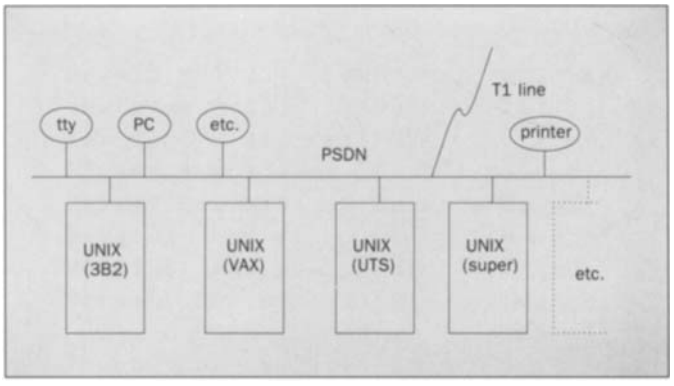


Figure 2. A two-tiered architecture.

varying degrees of power is the most distinguishing feature of our computing/publishing environment when compared to typical corporate MIS or desktop publishing enterprises. If an article intended for a magazine grows into a full length book, the source can be migrated from the UNIX system-networked PC or small multi-user system where it started to a large mainframe; the software tools, file formats and output device accessibility will be identical. This is in stark contrast to non-UNIX PC users who may find themselves waiting hours while an underpowered engine labors under the weight of 300 pages of text and graphics. Should this seem like an extraordinary example, Hodel[2] reports that the 5ESS® Switch is shipped to customers with some *120,000 pages* of documentation: the equivalent, if stacked on end, to a five-story building.

The formatting software on which we rely is chiefly the collection of tools in the Documenter's Workbench® software package.[3] But at this point, let's look at the publishing process from the user's point of view. Text can be entered from any terminal, including PCs, which are coming into increasing use as slightly intelligent terminals, using any editor of choice. Typical editors at AT&T Bell Laboratories are `ed`, `vi`, `emacs`, `jim`, and `sam`. All produce the one and only UNIX system byte stream format, so file conversion is never an issue. The formatting software in Documenter's Workbench includes:

- `troff`[4] for basic pagination, hyphenation, justification of text, font and size control, and macro definition facilities
- `tbl`[5] for tables of data and text
- `eqn`[6] for mathematical equations
- `pic`[7] for simple pictures, such as flow diagrams (Figures 1 and 2 are examples)
- `grap`[8] for scatter and line plots.

In general, the Documenter's Workbench tools use the `troff` text formatter as a filter to produce a device-independent metalanguage;[9] a postprocessor then converts this metalanguage into the PostScript page description language that in turn drives the output device selected. (Note that although there are postprocessors for other page description languages, our current development at AT&T Bell Laboratories is oriented to the PostScript standard.)

A recent addition to the `troff` toolkit is a picture inclusion facility that allows inclusion and placement of PostScript files anywhere in a document (see Figures 1 and 2); `troff` need only be told the file name, and size and placement of the "hole" on the page. The PostScript language performs scaling in the likely event the image is a different size than the hole. This "picture inclusion" approach has two advantages over the filter model:

- It is faster because graphic data does not have to be streamed through `troff`.
- It allows access from high-level applications to PostScript language capabilities that `troff` does not support, e.g., shading, color, line widths and line styles.

With this approach, including line art, graphs, and pictures of any kind becomes a matter of creating the appropriate PostScript file. This can be done using a number of commercially available Microsoft MS-DOS™ or Apple Macintosh™ software packages. Because some of these packages produce metalanguages other than the

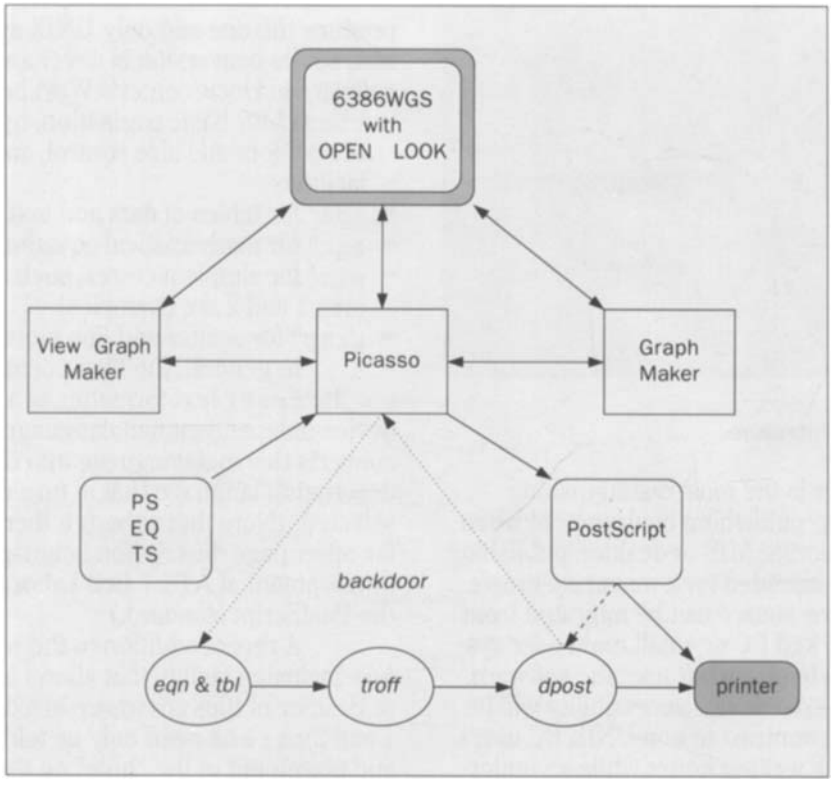


Figure 3. GWB architecture.

PostScript standard (for example, CompuServe's GIF™ and Computer Graphics Metafile [CGM]), the members of the Computer Graphics Group of the AT&T Bell Laboratories Computing Technology Center have written or acquired several translators for PostScript code.

To aid simple picture drawing (see Figure 3), we have designed and are currently developing a suite of UNIX system graphics tools, collectively called the Graphics Workbench (GWB), that will generate PostScript code directly from a high-level, user-oriented specification. The first tool, called `Picasso`, currently is in prototype. Based on Kernighan's `pic`, `Picasso` will serve not only as a directly usable picture-drawing language, but also as a base for higher level languages. When used directly, input can be either a formal, programmable language (see Figure 4), or via mouse and menu as specified by the OPEN LOOK™ Graphical User Interface[10] standard (see Figure 5). In GWB, high level applications built over `Picasso` might include graphing, viewgraphs, flow charts, state diagram generators and project trackers.

The philosophy of both Documenter's Workbench and GWB is to take advantage of the building-block approach the UNIX system encourages. Each tool does a specific job. Each fits with the other like parts in a machine, or a set of articulated gears. Improved tools or new tools are designed from the beginning to rest on the platform formed by existing tools. Users can then

```
.PS 4 4
boxwid = 1.25;  boxht = 1; ellipsewid = 1; ellipseht = 0.75
textsize = 15; textspace = 18; lineweight = 0.01; grey = 0.5
textfont = setfont("H")
Picasso: box at 4.25, 3.5 "Picasso"
Newvg: box with .e at Picasso.w - (1.5,0) "View Graph" "Maker"
Newgrap: box with .w at Picasso.e + (1.5,0) "Graph" "Maker"
WGS6386: box ht 1.5 wid 1.75 rad 0.15 filled grey with .s at Picasso.n + (0,1)
Screen: box ht 1.25 wid 1.5 rad 0.15 filled white "6386WGS" "with" \
        "OPEN LOOK" at WGS6386.c
Input: box rad 0.15 "PS" "EQ" "TS" with .ne at Picasso.sw - (1,0.625)
Output: box rad 0.15 "PostScript" with .nw at Picasso.se + (1,-0.625)
textfont = setfont("HI")
Troff: ellipse "troff " with .n at Picasso.s - (0,2.25)
EqnTbl: ellipse "eqn \fH&\f(HI tbl " with .e at Troff.w - (0.75,0)
Dpost: ellipse "dpost" with .w at Troff.e + (0.75,0)
Printer: box ht 0.5 wid 1 rad 0.15 filled grey font "H" "printer" \
        with .w at Dpost.e + (0.75,0)
"backdoor" at (Troff.x, Troff.y + 1)
arrowwid = 0.1; arrowht = 0.15; arrowfill = 1
arrow <-> from Picasso.n to WGS6386.s
arrow <-> from Newvg.ne to WGS6386.sw
arrow <-> from Newgrap.nw to WGS6386.se
arrow <-> from Newvg.e to Picasso.w
arrow <-> from Picasso.e to Newgrap.w
arrow from Input.ne to Picasso.sw
arrow from Picasso.se to Output.nw
arrow from EqnTbl.e to Troff.w
arrow from Troff.e to Dpost.w
arrow dotted from Picasso.s to EqnTbl.n
arrow dotted from Dpost.n to Picasso.s
arrow from Dpost.e to Printer.w
arrow from Output.s to Printer.nw
arrow dashed from Output.s to Dpost.n
.PE
```

Figure 4. `Picasso` sample command language.

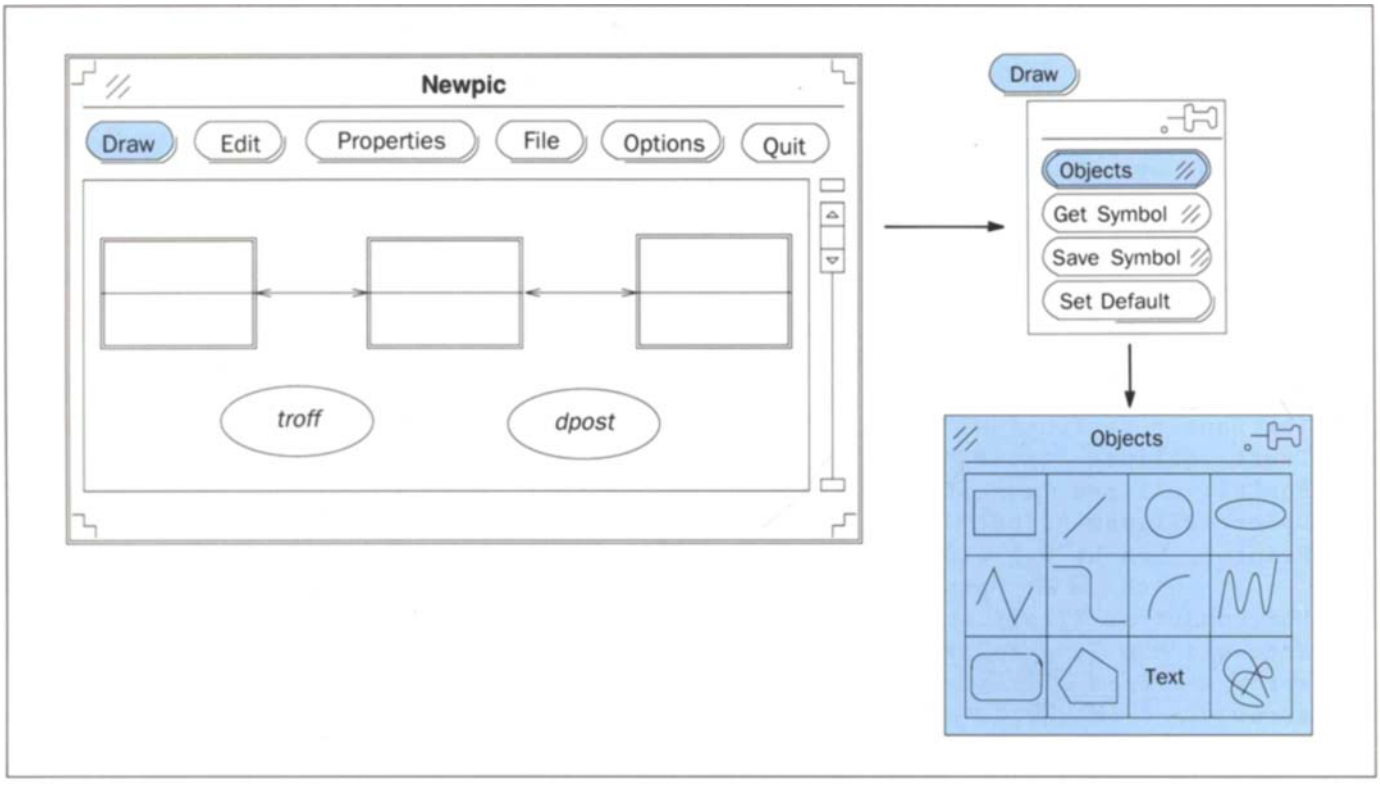


Figure 5. Sample `Picasso` graphical user interface.

construct tool combinations, via the UNIX `shell`, that exactly match their need. For example, rather than build indexing or bibliographic referencing capabilities into `troff`, those capabilities can be offered through packaged scripts, invoked only by those who need them. One such package, `refer`, converts symbolic references to citations and generates full references from a bibliographic database.

The host-based toolkit approach also allows users to install pieces experimentally, or broadcast them to hundreds of hosts on the network. Thus, there is no need to mail and install thousands of floppy disks for each iteration of the package.

Output Devices and PostScript Code

Although electronic publishing is almost 20 years old, one of the field's most significant advances—the emergence of the PostScript page description language[11] as an industry standard for laser printers—has occurred only in the last two years. Prior to the PostScript standard, high level software had to be aware of the peculiarities of every vendor's output device. This awareness extended to resolution in dots per inch (dpi), page size and orientation, font selection, special characters, and many other device

attributes. The PostScript language provides not only these primitive capabilities, but it also is programmable. For example, it is possible to add network line protocol checking into a printer by downloading the appropriate PostScript program into the printer.

We have our own PostScript interpreter, so we can make use of the language on our internal devices that do not otherwise offer it. The interpreter allows us to send PostScript code to a variety of intelligent terminals and workstations for in-the-office proofreading before committing to paper. In this way, we have turned the AT&T 630 MTG terminal[12] into a PostScript previewing device. For devices for which the interpreter cannot be run (for example, in a printer that does not support program downloading), we can run the interpreter on the host and produce the appropriate device-specific language there. This configuration can also be used to produce bitmaps for use in other non-PostScript standard environments.

Another strategy is to put the interpreter into a network-connected PC that acts as a device controller. Alternative media, such as 35mm slides, can be supported in this way without having to tamper with high-level applications or burden the network with large bitmaps.

An exciting prospect in recent months is the emergence of the first color PostScript printers. We have already installed several such devices and use them both for end-user graphics and for proofing documents intended for high-quality magazines.

Input Devices

Most of our internal publications at present consist of straight text, probably due to the inadequacy and awkwardness of picture-producing hardware and software rather than to lack of need. We have designed a Public Scanning Station, consisting of a commercially available 300-dpi optical scanner, a high resolution monitor and an AT&T 6386 WGS. The intention is to deploy the station in central locations at each site, such as Computer Centers, so users can digitize and crop photographs or artwork off-line. The resulting digital image can then be sent via PSDN as a PostScript file to the user's home system, where it can be inserted into a document by the `troff` picture inclusion mechanism described above.

A second systems project in our group is the Artist's Work Station, based again on the AT&T 6386 WGS, with a Truevision, Inc. VISTA™ graphics board and software from AT&T's Graphic Software Laboratory. It will allow the trained artist to produce the high quality color illustrations normally associated with large and expensive proprietary systems. Unlike proprietary systems, however, the device will have the advantage of being accessible from PSDN. Thus, it will be able to exchange electronic "soft finals" with customers at remote sites. The Artist's Work Station has already been used by professional artists in AT&T Bell Laboratories' Murray Hill Art Group to turn out production-quality artwork.

Some Real Life Examples

Let us now look at our CEP platform in operation, as it exists today.

One of the authors recently edited a paper written jointly with fourteen other people, spread over nine locations in five states. Each author composed a section on his or her home UNIX system using predefined `troff` macros. The files were all sent to the editor electronically via PSDN. Using the `sam` editor on an AT&T 630 MTG terminal, the editor "cut and pasted" the sections into one document. Standard UNIX system tools—known collectively as the Writer's Workbench®—were used to check for spelling and grammatical errors. The editor then made style and tense consistent across the segments. This laborious final step is unlikely to be fully automated by an expert system anytime soon.

Although `troff` has been maligned for the "sin" of not being WYSIWYG ("What you see is what you get"), by editing `troff` source in one window of the 630 MTG and displaying in another the formatted output via

the `proof` program or PostScript interpreter, we could immediately see the results of our editing effort. The edited document was then formatted and printed on a local PostScript printer. For some readers, printing illustrates the ineffectiveness of terminal proofing. Others may scorn it as a psychological "security blanket" for "paper addicts." In fact, the higher resolution of the printer (300-dpi versus about 100-dpi for the terminal) indeed reveals errors that might otherwise go undetected.

After final cleanup, the source document was sent electronically to the other 14 authors. The fact that *source* was sent, rather than finished output, is critical. First, since the Documenter's Workbench package and PostScript printers can be found at every location, each author could easily print his or her own paper copy with confidence that it would appear as the editor intended. In this mode, the platform is serving as a high quality fac-

simile service, offering cheap, rapid transfer of documents to many readers simultaneously. Second, by having source, the authors could reword a section if they thought the editor had done violence to his or her ideas. The rewritten section alone would then be electronically mailed back to the editor. Through this iterative process, 15 authors in different parts of the country produced a polished twenty-page document in a few weeks. Final copy was made on a centrally located 600-dpi printer to satisfy corporate requirement for an official "File Copy," to add the required hand-written signatures, and to serve as master for reproduced paper copies for additional readers.

The second example of our CEP platform required less networking, but more complex device technology. Becker, Chambers, and Wilks of the Information Sciences Research Division of AT&T Bell Laboratories wanted to publish a new version of their popular book on the `S` statistics package for the UNIX system. A key requirement was that all examples in the book be machine-tested; that is, the examples in print had to be the very text tested, not a typesetter's interpretation of it. The entire book was formatted using `troff`, and examples included using its source-file reading capability. An article from *The New York Times*, used as an illustration, was scanned and digitized, then included as a PostScript image file. Finally, the graphs in the book were generated by `S` itself, and also were included as PostScript files. Drafts were produced on a local 300-dpi PostScript printer, although fine-tuning of half-toned images required a 600-dpi printer for proofing. The final output was sent electronically to a commercial typesetting service as a PostScript file, and printed there on a 1270-dpi typesetter.

In checking the final proofs the authors discovered two problems that had not surfaced in draft copies. The finer resolution of the typesetter made gray-filled areas appear much lighter than they had on the local laser printers. Also, the printer's constant-width font, used by the authors to represent terminal input, was excessively light. These were resolved promptly, and *The New S Language* was printed, sent to a publisher for binding, and distributed to bookstores only a few weeks after the authors' final draft.

As a third, and perhaps most challenging task, this entire edition of the *AT&T Technical Journal* was produced on general-purpose UNIX systems in Bell Laboratories, using software discussed in this article.

Work To Be Done

The platform we have described has evolved over 20 years, and evolves still. For example, some of the pieces, such as the Graphics Workbench and the Public Scanning Station, still are in development.

A major impediment to ubiquity of the platform is the slow speed of some of the elements of our internal network. A scanned image in black and white might generate four megabytes of data; a device connected at 9600 bits per second would take an intolerable two hours to transmit. Of course, current LAN technology permits much higher bandwidth (both the Datakit network and Xerox Corporation's Ethernet® network can handle

several megabits per second with the proper interface) so this is a problem of deployment rather than technology.

A frequent complaint about `troff` is its awkwardness as a language and its difficulty for novice users. It is our position, however, that `troff` is best viewed as an *assembler language* that understands arcane features of printer hardware, but is best written by another program rather than by a human. The `troff` preprocessors provide a reasonable interface for specific formatting tasks (e.g., `tbl` for tables and `eqn` for equations), and each has its own language suited to the task at hand. Traditionally, macro packages serve as the main interface to `troff` for defining the document format and preparing the text within it. The macro packages are programmed in the `troff` formatting language, but the user-level macro calls are inherently as terse and cumbersome as the `troff` language itself.

There is much work in progress towards improving the user-level interface. One alternative to the traditional macro package is `monk`,[13] a database-driven interface that offers a mnemonic command language and predefined style sheets to generate specific document types. With `monk`, users need only change the information in the template file initially, then add the body text marked with the appropriate English-like command to define the particular element in the document (e.g., section, paragraph, list item).

The popularity of graphical user interfaces (the WYSIWYG approach) has prompted a good deal of work geared towards the "friendly" user interface. One example is `snaz`, a visual editor for `monk` that displays formatted output to reflect the commands and text entered using menu and mouse functions. In addition, there is work in progress to develop an OPEN LOOK graphical user interface to some of the Documenter's Workbench tools and all of the GWB tools. This easy-to-use mouse-and-menu approach should provide a friendlier front-end without sacrificing the raw power of `troff` as a full typesetting language.

Recent features of UNIX System V, particularly Remote File Sharing, will make it unnecessary to electronically mail copies of co-authored documents around the network. Instead, a single shared copy could be maintained, so all authors always see the same document. Through the windows mechanism, authors could exchange comments or add margin notes in real time while reading the shared copy.

Much exciting work is left to do on other manual, but theoretically automatable, steps in publishing—e.g., layout, the arrangement of pictures, diagrams, columns, etc. on the page. We currently are integrating state-of-the-art optical character recognition software from our Research area into the Public Scanning Station. Output from the scanning station could thus be reconstructed text as well as bitmaps.

The Andrew Project[14] at Carnegie-Mellon University and the Media Lab at the Massachussetts Institute of Technology have shown how much more effective documentation can be when it is liberated from paper and allowed the power to speak and move. We expect increasing attention to this hypermedia approach.

Summary

The platform we have described is a combination of both preexisting and newer components built on the base of the UNIX system. Its power lies not in any one piece, but in the synergy resulting from the combined UNIX system, Datakit switch, Documenter's Workbench and Graphics Workbench software, and PostScript language. Both the geographic extent and power of the platform can be extended indefinitely without redesign. Though based in part on the technology of the 1970s, the platform is robust enough to support sophisticated enhancements well into the 1990s.

Acknowledgments

First and foremost, we must mention the pioneering work and continual support and assistance of Brian Kernighan and Jon Bentley of the Computing Science Research Center. Sharon Murrel and Paul DeBra

developed `snaz`, and Marcia Derr developed `prefer`. Rich Drechsler is responsible for the PostScript postprocessor and, along with Alan Wilks, developed the PostScript picture inclusion facility. The Graphics Workbench design and implementation is due to Michael Siemon, Chi Choy and Manijeh Shayegan. Bruce Wallace and Michael Chai did the Artist's Work Station and are currently working on the Public Scanning Station. Vince Fasciano has contributed the CGM-to-PostScript program, among many other efforts. Chris Scussel and Jess Kartus from AT&T Bell Laboratories, Indian Hill, wrote the PostScript interpreter, which has been used by Lorinda Cherry of Research to produce the 630 MTG PostScript viewing program.

References

1. Alan A. Weiss and Debasis Mitra, "A Transient Analysis of a Data Network with a Processor-Sharing Switch," *AT&T Technical Journal*, Volume 67, Issue 5, September/October 1988, pp. 4-16.
2. M. L. Hodel, "Alternate Media for 5ESS® Switch Customer Documentation," *First Annual AT&T Customer Documentation Symposium*, AT&T Bell Laboratories, October 1988, pp. 253-256.
3. *UNIX System V Documenter's Workbench Software Product Overview*, AT&T, 1986.
4. Joseph F. Ossanna, "Nroff/Troff User's Manual," *UNIX Programmer's Manual*, Seventh Edition, Volume 2, Holt, Rinehart and Winston, New York, N. Y., 1983, pp. 196-229.
5. Michael E. Lesk, "TBL—A Program to Format Tables," *UNIX Programmer's Manual*, Seventh Edition, Volume 2, Holt, Rinehart and Winston, New York, N. Y., 1983, pp. 157-174.
6. Brian W. Kernighan and Lorinda L. Cherry, "Typesetting Mathematics—User's Guide (Second Edition)," *UNIX Programmer's Manual*, Seventh Edition, Volume 2, Holt, Rinehart and Winston, New York, N. Y., 1983, pp. 146-156.
7. Brian W. Kernighan, "PIC—A Graphics Language for Typesetting (Revised User Manual)," *Software Practice and Experience*, Volume 12, January 1982, pp. 1-20.
8. Jon L. Bentley and Brian W. Kernighan, "GRAP—A Language for Typesetting Graphs," *Communications of the ACM*, Volume 29, Number 8, August 1986, pp. 782-792.
9. Brian W. Kernighan, "A Typesetter-Independent TROFF," AT&T Bell Laboratories, Computing Science Technical Report No. 97, March 1982.
10. *OPEN LOOK™ Graphical User Interface, Functional Specification (Revision 12)*, Sun Microsystems, Inc., September 1988.
11. *PostScript Language, Reference Manual*, Adobe Systems, Inc., Addison-Wesley, 1985.
12. Yoram Bernet and James M. McCarthy, "The 630 Multi-Tasking Graphics Terminal," *AT&T Technical Journal*, Volume 66, Issue 6, November/December 1987, pp. 3-14.
13. Sharon L. Murrel and Ted J. Kowalski, "Monk: A High-Level Text Compiler," *AT&T Technical Journal*, Volume 68, No. 4, pp. 45-60.
14. James H. Morris, et al., "Andrew—A Distributed Personal Computing Environment," *Communications of the ACM*, Volume 29, Number 3, March 1986.

Biographies (continued)

facility to synthesize graphic symbols for `troff` *typesetting. Since 1980, she has been enhancing UNIX system electronic publishing software, introducing tools from other sources, and developing tools and applications. She integrates new with existing electronic publishing software into a packaged release for AT&T-wide distribution. Currently, she is working with Sharon Murrel on tailoring the technology for specialized applications, such as the AT&T Technical Journal and AT&T customer documentation.*

(Manuscript received May 1, 1989)