OVERVIEW: THE EMERGENCE OF ELECTRONIC PUBLICATION TECHNOLOGY

Sharon L. Murrel

Sharon L. Murrel *is a member of technical staff in the Computer-Aided Information Systems Research Department at AT&T Bell Laboratories in Murray Hill, New Jersey. Ms. Murrel's research interests include creating advanced systems for electronic textual communication, particularly focusing on publishing and computer conferencing. She is currently building an interactive system for designing tabular information using "layout-by-example" as well as developing new languages and tools for automated production of large publications, like this journal. She joined the company in 1977 and has a B.S. in computer science from New York University, and an M.A. and Ph.D. in experimental psychology from the State University of New York at Stony Brook.*

The physical object you are now holding in your hand—the new issue of the *AT&T Technical Journal*—was born directly from this issue's theme: electronic renderings of language. The integration of the ideas, tools and techniques described in these papers provided the platform used to electronically produce this issue.

Traditionally, written language is associated with paper; today, it also lives in various electronic forms. Paper output, complete with high quality color plates, can be taken to the beach without an extension cord, battery packs, or a network. However, rigorous scholarly research benefits from the cross-indexing and browsing facilities available when the same information is available in large on-line libraries. Thus, these two forms of communication—physical and electronic—provide radically different approaches to the same—related—objective: the user's ability to access preserved data and information. Our goal is to maximize the synergy between the two output forms by tying together the different renderings.

This issue of the *AT&T Technical Journal* is, in fact, what this issue is all about: it is self-referential. That is, it is composed of articles written by authors using a variety of typesetting tools, transmitted across our electronic network, edited by journal staff, translated into a generic markup language, and formatted according to new style sheets developed specifically for the publication's special requirements. The pages of this issue have been produced using the `troff` text formatter driven by a stylesheet-based preprocessor named `monk`, and by a series of "little languages" that create holes in the text, place captioned figures, draw holding lines for art and photos, generate citations and cross references, and address numerous other crucial production details. This issue, successfully integrating all these tools, marks a new beginning for the *AT&T Technical Journal*.

Kernighan opens the issue with a philosophical survey of the documentation tools created by the research group at AT&T Bell Laboratories, where the UNIX® operating system was born. Naturally, his focus is `troff`, the typesetting language pioneered at Bell Laboratories around 1973, and the rich collection of special purpose tools subsequently built upon its power and capabilities. Kernighan stresses that

these tools are formally defined *languages* rather than programs; and he explains how this group of separate, specialized languages together exemplify the UNIX system tool-based tradition.

Bentley explains the significance of little languages in extending the capabilities of these general purpose tools. Before our eyes he builds a little language that draws data diagrams, then sketches such languages in three other domains. The dotchart language generates input for `grap`, a graph drawing language, which in turn writes only `pic`, a picture drawing language. `pic` then is the owner of all the knowledge necessary for generating correct `troff`. Not only are these languages layered upon each other, but also all are based on a family of software development tools. `grap` and `pic` are defined as "context-free grammars" that use the `yacc` and `lex` tools to create their parser and lexical analysis routines. The ease with which these languages are written and augmented as separate tools underlines the value of the research philosophy, and is a stunning demonstration of functional composition.

Ansen's summary of the Office Document Architecture (ODA) standard offers one approach to generic markup and document access, topics further developed in the papers which follow. The goal of ODA is to enable electronic document exchange between dissimilar systems. This is accomplished by providing a shared specification for compound document structuring and encoding. ODA's hierarchical structure organizes document input into logical structures based upon meaning—e.g., paragraphs and figures—and content-independent layout structures, such as pages and blocks. Analogously, it separates rules for processing content from rules for processing structure and styles. These distinctions form the basis for the future evolution of ODA to encompass new content types and media.

Murrel and Kowalski describe and illustrate `monk`, a stylesheet-driven preprocessor for `troff`. `Monk` provides primitives for programming document style sheets, and an extensible set of generic markup codes for use by authors. As a transitional tool bridging purely procedural and purely generic markup, `monk` continues to allow precise procedural formatting commands. However, it introduces many generic elements whose format is completely determined by the requested style. `Monk` adheres to the UNIX system philosophy of separate languages—each undertaking small parts of the job—by exploiting the power of hierarchical specifications. By following the style sheets, it translates author requests into a set of primitives, which are in turn automatically compiled into raw `troff` input. Like the dotchart example above, the style sheets dictate the transformation of author requests. The primitives produced encapsulate detailed knowledge of `troff`. `Monk` therefore frees the author from needing to understand style and layout. Moreover, it allows the document designer to *design documents* rather than debug troff.

Petrea and Taylor trace the migration toward both generic markup languages and multimedia production systems at an organization responsible for over half a million documents. Beginning as a `troff` shop in which authors specified indentation and font size, they are now adopting generic markup codes to improve semantic capture, standardize the look of their documents, and render them in other electronic forms. While a human can identify the surface elements of a document, a computer cannot, even when reading a text file. A great deal of semantic information crucial for document reprocessing is difficult or impossible to extract ex post facto. Therefore, it is crucial that authors specify the *type* of information instead of *how* to format it. For example, rather than centering five lines, writers must identify the first two lines as a *title* and the last three as *author information*. This progression from procedural to generic markup is an enabling technology. When authors contribute this information, new tools can provide advanced capabilities dependent on the desired style, output media, or user purpose.

Research users at Bell Labs can retrieve and display full text documents, including graphics, from a

large on-line database while sitting in their own offices. Cherry and Waldstein, in describing this retrieval and display capability, address a range of issues from document capture to user interfaces for browsing. Much of their work focuses on handling papers written in multiple typesetting languages, including several `troff` macro packages, `monk` and TEX, preprocessing the tables, equations, pictures and bitmaps, and displaying them on a variety of terminals. (TEX is a trademark of the American Mathematical Society.) Given a mouse and a bitmap, `reader` provides a dynamic, non-sequential browsing facility. This suite of tools allows users to check facts and references, read papers, or simply preview them before requesting paper copies.

Crenshaw presents some data evaluating the usefulness of documentation, and gently reminds us not to neglect document *content* as well as formats, however elegant and automatic. Her results are based on answers to questionnaires—completed by both document producers and readers—evaluating the process and management of document production, document content, and the delivery method. The most interesting results rate the importance of a variety of documentation attributes. Technical *accuracy* and *currency* are rated highest. This emphasizes the importance of progress toward the availability of on-line, living documents.

In the final article, Nelson and L'Hommedieu describe the electronic publishing platform that is developing based on the tools described in all the other papers, and on a network which reaches to all parts of AT&T. Their work illustrates the effort to make our rich family of tools more widely available, and to incorporate new graphical and WYSIWYG (What you see is what you get) tools.

A final thought: electronic publishing as we understand it is *not* word processing taken to a somewhat more sophisticated level. It is a wholly new set of complex, interwoven technologies encompassing word processing, certainly, but also graphical and other non-print forms of representation. The best we can hope for in this forum is to suggest the breadth of the field, take you on a brief guided tour, and recommend to our readers that you take the opportunity to explore it in greater detail. For the field of electronic publishing technology is bigger than what is going on in any one of these papers.

(Manuscript received July 25, 1989)