ELECTRONIC PUBLISHING, VOL. 5(1), 19–44 (MARCH 1992)

Important papers in the history of document preparation systems: basic sources

RICHARD FURUTA[1]

National Institute of Standards and Technology[2]
and
University of Maryland[3]

SUMMARY
This report provides a narrative description of influential papers that discuss computer-based document preparation systems. The report's focus is on the systems actually used to prepare documents—editors and formatters, and the goal is to provide an introduction to the papers that have been influential on the community of researchers who investigate such systems.

KEY WORDS Document preparation Text processing Document manipulation Formatting

1 INTRODUCTION

Document preparation has been an increasingly important application of computers for over twenty-five years. When first developed, document preparation systems were only one of many applications supported on a computing system (and, indeed, an application that system administrators often tried to suppress rather than support). In the modern computing environment, it is common to find computers that are used only for document preparation. The sophistication of the systems has increased over the years, evolving from systems that were intended to produce text on a particular output device into device-independent systems in which it is possible to specify text, and also equations, tables, line drawings, and other document components. Modern systems enhance the ease with which the document can be specified, perhaps through an interactive interface, and enhance the reusability of the document and its components, perhaps through generic coding (generic coding is specification of the document's logical structure rather than its physical appearance).

In addition to being an important target for development projects, document preparation systems have become the focus of academic research, carried out both in universities and in industrial research centers. Indeed, the research is wide ranging as a successful document preparation system draws ideas from a broad collection of areas. Document

[1] Address correspondence to Department of Computer Science, University of Maryland, College Park, MD 20742. Electronic mail to `furuta@cs.umd.edu` on the Internet.
[2] Affiliated with the Office Systems Engineering Group, Systems and Software Technology Division, Computer Systems Laboratory. Contribution of the National Institute of Standards and Technology. Not subject to copyright.
[3] Affiliated with the Department of Computer Science and the Institute for Advanced Computer Studies at the University of Maryland, College Park, MD. The portion of this work carried out at the University of Maryland is based upon work supported by the National Science Foundation under Grant No. CCR–8810312.

CCC 0894–3982/92/010019–26 *Received 4 August 1989*
Revised 9 June 1992

preparation systems have also been the target of standardization, and international standards organizations have issued standards for some parts of the document preparation process.

As one might expect with an area of study of tools dedicated to producing written words, many of the developments in the area of document preparation systems have been charted in published reports, books, journal papers, conference papers, user manuals, and internal documents. The newcomer to the area faces the daunting task of finding and evaluating these reports. Unfortunately, one finds systems that have been created in a vacuum: systems that reinvent and reimplement previously developed solutions.

It is the purpose of this report to identify those projects that have been especially influential on the thinking of the community of researchers who have investigated these systems. In many cases, a particularly successful project will generate many publications about itself. In such cases, I have tried to select the one or two of those publications that best describe the project as a whole. I intentionally have limited the reports that I have included on this list to “papers”—I have not included books or reference manuals describing the software package. The limitation to papers is because, by their nature, papers tend to be more focused, more concise, and more widely disseminated than other types of publications. Additionally, it is easier to obtain copies of individual papers than larger books. For example, the United States’ copyright law (and perhaps that of other countries as well) includes provisions permitting personal “fair use” copying of individual papers for “purposes of scholarship or research”. I believe that, in all cases, the selected papers are openly available (although a few may be difficult to find), and, when possible, I have selected published reports (journal and conference papers) rather than internal reports.

The papers listed in this report mostly are limited to those appearing in 1988 or earlier—it is too early to determine whether or not work appearing since that time will be “influential.” I scanned the reference lists of the papers presented at the significant scholarly conferences in the area (see Sections 1.1.2 and 1.1.3, below), the reference lists of those papers that have appeared in the first two volumes of the journal *Electronic Publishing: Origination, Dissemination, and Design* (the issues that were available at the time this report was prepared), and the major bibliographic lists that have been collected (see Section 1.1.4, below), to obtain suggestions for papers to be added to my own initial list of references. Additionally, I am grateful to my colleagues who have commented on earlier drafts of this report (as noted in the acknowledgements at the end of the report).

It is important to note, however, that the selected list of papers is my own, and that by design the list is neither comprehensive nor reflective of the complete set of “good” work in the area. It seemed neither appropriate nor practical to produce a comprehensive list the large volume of publications in the area. For example, as part of the Text Encoding Initiative, an attempt is being made to produce a comprehensive bibliography in the overlapping area of “structured text.” A current draft of that bibliography [1] includes 832 references; there is no reason to expect that a comprehensive bibliography covering document preparation systems would be smaller.

It also did not seem appropriate to me to include a paper solely because I or the reviewers felt that it represented “good” work. I wish to emphasize that exclusion of a paper from this list is not intended to serve as an indication of its quality. The purpose of this report is to identify papers that have become influential on other researchers in the area, particularly as reflected by citation in those other researchers’ papers. Good work has appeared in forums that have not received wide dissemination among researchers in this area—for example in specialized conferences or in peripherally relevant journals. However, since I have tried to to eliminate papers that do not seem to be commonly known and referenced by researchers in the area, it is quite possible that such good and innovative papers are not found in this report.

It may also be useful to this report's readers to reemphasize that this report's focus on document preparation systems covers but a part of the broader area of "electronic publishing." Although some of the related areas are discussed in Section 6, no attempt has been made to cover the broader area comprehensively. Similarly, this report does not attempt to cover the practices associated with traditional publishing, except as they are implemented in the computer tools that result from the primary work described here.

1.1 Information sources

Since the list of papers given in this report is, by design and by necessity, partial, the reader performing a more complete literature search may wish to consult the sources listed below.

1.1.1 Journals and periodicals

The results of work in document preparation systems have appeared in a wide range of journals, as may be noted when inspecting this paper's literature list. Certain journals bear special attention in performing literature searches.

Many influential papers may be found in *Software—Practice and Experience*, published by John Wiley and Sons. I expect that significant papers will continue to appear there as well as in the newer Wiley journal, *Electronic Publishing: Origination, Dissemination, and Design*. Important papers have also appeared in the Association for Computing Machinery's *Transactions on Office Information Systems* (renamed in 1989 to *Transactions on Information Systems*). Occasionally, more general papers have appeared in *IEEE Computer*, *IEEE Software*, *IEEE Transactions on Software Engineering*, *Communications of the ACM*, and the *Computer Journal*.

A more specialized source is the *TUGBoat*, the publication of the TEX Users Group, which focuses primarily on uses of Knuth's TEX typesetting system [2] and related programs.[4] Significant papers have occasionally appeared in less directly related sources, for example *Byte* magazine. The reports published by Seybold Publications provide timely descriptions of relevant systems, although they are generally not available in the academic environment.

1.1.2 Scholarly conferences

Conference proceedings have contained the only reports of much of the influential work. The International Conference on Research and Trends in Document Preparation Systems, held in Lausanne, Switzerland, in February of 1981 is acknowledged to be the first scholarly conference devoted to document preparation systems. (This conference is commonly called the "Lausanne Conference.") The papers submitted to the conference were distributed to attendees as extended abstracts. The conference organizers commissioned full papers from selected extended abstracts, and collected them together into a book.

[4] TEX is discussed later in this report.

- *Document Preparation Systems: A Collection of Survey Articles*, eds., Jurg Nievergelt, Giovanni Coray, Jean-Daniel Nicoud, and Alan C. Shaw, North-Holland Publishing Company, 1982.

The chapters in the book provide broad-ranging reviews of the state of the art in several areas of document preparation systems.

The first conference of significant visibility was the ACM SIGPLAN SIGOA Symposium on Text Manipulation, held in Portland, Oregon, in June of 1981.

- *Proceedings of the ACM SIGPLAN SIGOA Symposium on Text Manipulation.* Available as *SIGPLAN Notices*, **16**(6):61–67, June 1981, and *SIGOA Newsletter* 2(1&2), Spring/Summer 1981.

The papers in the proceedings of this conference are strongly oriented to systems-based descriptions of research and commercial systems, and almost all report on systems of significance.

The "Electronic Publishing" series of international conferences have become the focal point for presentation of scholarly research into document preparation systems. The conferences have been run by members of the research community and the papers in them have concentrated on presenting descriptions of prototype systems and on giving details of the underlying models and implementation techniques. The first of these conferences, called "EP86," was sponsored by the British Computer Society in 1986.

- *Text Processing and Document Manipulation*, ed., J. C. van Vliet, Cambridge University Press, April 1986. Proceedings of the international conference, University of Nottingham, 14–16 April 1986.

The EP86 papers included some which were primarily designs, unvalidated at the time of the conference by implementation. Taken as a whole, the systems and techniques discussed at the conference represent a widely chosen selection of research into the entire range of electronic publishing systems.

The second in the series of "Electronic Publishing" conferences (EP88) was sponsored by INRIA, the French national research organization, in 1988.

- *Document Manipulation and Typography*, ed., J. C. van Vliet, Cambridge University Press, April 1988. Proceedings of the International Conference on Electronic Publishing, Document Manipulation, and Typography, Nice (France), 20–22 April, 1988.

When compared to EP86, EP88 represents a significant strengthening of the standards applied to accepted papers. The papers presented were required to be technically sound and interesting, but also validated by experience.

These standards were carried forward into the third conference in the series (EP90), which was sponsored by the United States' National Institute of Standards and Technology in 1990.

- *EP90*, ed., Richard Furuta, Cambridge University Press, September 1990. Proceedings of the International Conference on Electronic Publishing, Document Manipulation, and Typography, Gaithersburg, Maryland, September 1990.

Perhaps the most significant new trend reflected in EP90 was the increasing importance of hypertext, hypermedia, and multimedia research within the context of electronic publishing. Hypertext and hypermedia will be revisited later in this report's discussion of associated technologies.

The most recent in this series of conferences was EP92, held in Lausanne, Switzerland, and sponsored by the Ecole Polytechnic Fédérale de Lausanne (Swiss Federal Institute of Technology).

- *EP92: Proceedings of Electronic Publishing, 1992*, eds., C. Vanoirbeek and G. Coray, Cambridge University Press, April 1992.

In December of 1988, ACM sponsored a second conference in the area of document processing.

- *Proceedings of ACM Conference on Document Processing Systems* ACM, New York, December 1988.

DocProc '88 shared a core set of topics in common with EP88, but adopted a broader definition of the areas of interest within the realm of document processing. In particular, invited papers and panels permitted more direct consideration of the workplace environments in which documents were created and used, thereby broadening the the focus from only the *means* by which document manipulation was achieved.

1.1.3 Scholarly conferences—specialized audiences

In addition to the highly visible international conferences just described, the proceedings of some smaller and more specialized conferences are worth noting.

The PROTEXT series of conferences was held in 1984, 1985, 1986, and 1987. Proceedings from these conferences are available from Boole Press, and are edited by J. J. H. Miller. A tutorial session was associated with each of the conferences, and the commissioned papers are also available in book form from Boole Press, again with Dr Miller as editor. Although the conferences were not refereed until the 1987 session, a number of interesting systems were first described in the conferences' proceedings.

Proceedings from the three European TEX conferences are available in book form. While the papers are loosely focused around the TEX formatting system, many of the concepts discussed are applicable to document preparation systems in general.

- *Proceedings of the First European Conference on TEX for Scientific Documentation*, ed., Dario Lucarella, Addison-Wesley, 1985.
- *TEX for Scientific Documentation*, ed., Jacques Désarménien, Springer-Verlag, 1986. Lecture notes in Computer Science, No. 236.
- *TEX: Applications, Uses, Methods*, ed., Malcolm Clark, Ellis Horwood, 1990.

In addition, proceedings of the annual TEX Users Group (TUG) conferences held in the United States are available from TUG. The focus here is often more TEX-specific than the books just mentioned.

Papers commissioned for a course on structured documents held in January of 1987 in Aussois, France, and sponsored by INRIA were published in 1989.

- *Structured Documents*, eds., Jacques André, Richard Furuta, and Vincent Quint, Cambridge University Press, 1989.

The papers in this book provide an overview of a particular class of document, the structured document, which describes the document's logical structure (e.g., chapter formed from sections formed from subsections, etc.) rather than its physical structure (e.g., its physical display on paper). The book also describes the current status of specific areas of research into the structured document.

1.1.4 Bibliographic lists

The proceedings for the Symposium on Text Manipulation and EP86 included annotated bibliographies. The interested reader may also wish to consult these bibliographies.

- Brian K. Reid and David Hanson, 'An annotated bibliography of background material on text manipulation', *Proceedings of the ACM SIGPLAN SIGOA Symposium on Text Manipulation*, *SIGPLAN Notices*, **16**(6), 157–160, (June 1981). (Also available as *SIGOA Newsletter* 2(1&2), Spring/Summer 1981.)
- J. C. van Vliet and J. B. Warmer, 'An annotated bibliography on document processing', in *Text Processing and Document Manipulation*, ed., J. C. van Vliet, 260–276, Cambridge University Press, (April 1986). Proceedings of the international conference, University of Nottingham, 14–16 April 1986.

Two wide-ranging bibliographies collected by André provide overviews of the area and related topics.

- Jacques André, 'Analytical bibliography on text manipulation', *T.S.I.—Technology and Science of Informatics*, **1**(5), 369–378, (1983).
- J. André, 'Manipulation de documents: Bibliographie', *T.S.I.—Technique et Science Informatiques*, **5**(4), 363–365, (1986).

André's bibliographies also provide an excellent introduction to work carried out in France and other European countries.

Rubinstein's 1988 book *Digital Typography* includes an annotated bibliography with entries of relevance (although, as suggested by the book's title, more strongly focused on the related area of digital typography and font design). The *Structured Documents* book, mentioned above, includes a large bibliography of collected citations obtained from the individual articles making up the content, and as noted above a comprehensive bibliography on structured text is being collected by the Text Encoding Initiative.

2 SURVEYS OF DOCUMENT PREPARATION SYSTEMS

Three survey papers provide a good overview. Van Dam and Rice's 1971 survey of editors gives a snapshot of early development in text editors, introducing techniques that continue to influence current systems.

- Andries van Dam and David E. Rice, 'On-line text editing: A survey', *ACM Computing Surveys*, **3**(3), 93–114, (September 1971).

The survey of editing systems is updated in Meyrowitz and van Dam's 1982 survey (this 1982 survey, and the formatter survey discussed next, were originally prepared for the 1981 Lausanne conference, mentioned above). In addition to reviewing text editors, this survey also describes some of the early, influential integrated editor/formatters (editors that permit both the editing of the document and also the display of the resulting formatted document).

- Norman Meyrowitz and Andries van Dam, 'Interactive editing systems: Parts I and II', *ACM Computing Surveys*, **14**(3), 321–415, (September 1982).

The same issue of *Computing Surveys* also contains a survey of document formatting systems. While many of the integrated systems discussed in the formatting survey are also discussed in the editing survey, the two articles differ in approach.

- Richard Furuta, Jeffrey Scofield, and Alan Shaw, 'Document formatting systems: Survey, concepts, and issues', *ACM Computing Surveys*, **14**(3), 417–472, (September 1982).

The overall model of document preparation used to categorize the papers described in this report is developed from that described in the Furuta, Scofield, and Shaw survey. We identify three representations of the document: abstract representation, physical representation, and page representation. The document's abstract representation describes the document's content and structure, and is specified by an author in a markup language. The physical representation of the document maps the document into the context of a particular physical representation (precisely specifying, for example, the positionings of the document's elements within the display space). The page representation is in the format expected by a specific output device.

Document preparation functions are defined as mappings among the representations. *Editing* modifies the abstract representation; in essence it is a mapping from abstract representation to abstract representation.[5] *Formatting* transforms an abstract representation into a physical representation. The physical representation is converted into a page representation through a *viewing* transformation.[6]

- Pehong Chen and Michael A. Harrison, 'Multiple representation document development', *Computer*, **21**(1), 15–31, (January 1988).

In the remainder of this report, we will focus on influential systems, as partitioned by the just-described document representations and functions. The next section will consider systems that perform only the editing function. This will be followed by systems that implement the formatting mapping in isolation. Interactive systems that simultaneously carry out the editing and the formatting functions are then considered. Finally, we turn our attention to a brief overview of some affiliated issues.

[5] The complete model also applies the editing function to the document's physical representation—for example, the result of fine-tuning the output by turning individual pixels on or off. This mapping is not needed in the context of the present paper.

[6] Chen and Harrison have also presented a framework for characterizing document development, identifying a non-exhaustive list of tasks that might be carried out. Their set of tasks covers a broader range of activities than those associated only with document preparation. For each of the tasks, Chen and Harrison analyze the tradeoffs for each task between interactive specification and textual language-based specification, showing how the appropriateness of the style of specification is tied to the system user's task domain.

3 INFLUENTIAL SYSTEMS: EDITING

One useful characterization of an editing system results from examining the class of objects that the system is intended to manipulate. Such an examination is most useful when it focuses not only on the list of objects supported, but also on the degree to which the system encapsulates knowledge of each of the object representation's syntax and semantics.

In the most general category of editors, objects to be edited are treated simply as a stream of bytes with the addition of only the additional grouping of sequences of characters into lines. Because of the generality of the representation, a wide variety of kinds of objects can be edited—program text, documentation, and indeed compiled object code are frequent targets. However, because of this generality, such editors are usually not very helpful in ensuring the accurate specification of an object—syntactic errors, for example, are not flagged until a later processing step, after the editing session has ended. For this reason, editors have been developed that incorporate information of the structure of the objects being specified—computer programs and structured documents are the two examples considered here.

The systems discussed in this section handle only editing functions. A later section will consider the incorporation of formatting functions to create an interactive integrated editing and formatting system.

3.1 General-purpose editors

General-purpose editors permit editing of a wide range of objects by reducing all to a common denominator—as a file of lines of characters. User interfaces vary in character from those that are oriented to a command line to those that are oriented to a full-screen display of the file undergoing editing. Examples of the interaction styles for each of these kinds of editors have been given in the van Dam and Meyrowitz survey paper mentioned earlier [3]. See, for example, the descriptions of TECO and SOS for examples of line-oriented editors and the discussion of XEDIT for an example of a screen-oriented editor.

A screen-oriented editor that has been particularly influential, with strong supporters and also strong detractors, is named EMACS. EMACS is notable both for its reconfigurability. It also provides a user interface designed primarily for expert users, instead of a more general one designed for a wider group of users with more diverse skills.

- Richard M. Stallman, 'EMACS, the extensible, customizable self-documenting display editor', *Proceedings of the ACM SIGPLAN SIGOA Symposium on Text Manipulation, SIGPLAN Notices*, **16**(6), 147–156, (June 1981). (Also available as *SIGOA Newsletter* 2(1&2), Spring/Summer 1981.)

3.2 Syntax-directed program editors

A wide variety of editors have been developed that incorporate knowledge of the syntax of a particular programming language. During program generation, this knowledge is used to insure that the specified program is syntactically correct. Techniques used in design and implementation of syntax-directed program editors are similar to techniques used in design and implementation of document editors, and lessons learned in one area often can be applied in the other. However, direct application of a system designed primarily for editing programs to the task of editing documents seems less appropriate, primarily because an implementation based on tree-walking primitives closely parallels the structure of computer programs but tends to artificially subdivide the components in a document. As experience with Mentor suggests (see below), extension of syntax-directed editors to contain features specialized for the document domain seems to provide a more appropriate interface.

One of the earliest examples of such a syntax-directed system is Hansen's Emily.

- Wilfred J. Hansen, 'User engineering principles for interactive systems', *Proceedings, AFIPS Fall Joint Computer Conference*, **39**, 523–532, (1971).

Emily permitted the interactive expansion of a specification written in BNF, and was intended for use in construction and modification of programs written in a grammatically specified higher-level language.

One of the best-known syntax-directed program editors is the Cornell Program Synthesizer.

- Tim Teitelbaum and Thomas Reps, 'The Cornell Program Synthesizer: A syntax-directed programming environment', *Communications of the ACM*, **24**(9), 563–573, (September 1981).

The Cornell Program Synthesizer was implemented specifically for the separate block structured languages that it supported. The more general approach to design and implementation of such program editors is to through a "generator" that can create such specialized program editors from a description of the language that is to be manipulated.

- Thomas Reps and Tim Teitelbaum, 'The Synthesizer Generator', in *Proceedings of the ACM SIGSOFT/SIGPLAN Software Engineering Symposium on Practical Software Development Environments*, pp. 42–48. Association for Computing Machinery, (April 1984). (Available as *Software Engineering Notes*, **9**(3), May 1984, and *SIGPLAN Notices*, **19**(5), May 1984.)

This approach is also relevant to development of interactive document editing systems; see particularly the discussion of Grif in the later section on influential interactive integrated systems.

Some techniques developed for the Mentor syntax-directed program manipulation environment are interesting because they seem to have been added to a more basic syntax-directed program editor to assist in the editing of documents. For this reason, examining the Mentor design shows the similarities and differences between functions specialized for program editing and functions specialized for document editing.

- V. Donzeau-Gouge, B. Lang, and B. Mélèse, 'Practical applications of a syntax directed program manipulation environment', in *Proceedings—7th International Conference on Software Engineering*, pp. 346–354. IEEE Computer Society, (1984).
- V. Donzeau-Gouge, G. Kahn, B. Lang, and B. Mélèse, 'Document structure and modularity in Mentor', *Proceedings of the ACM SIGSOFT/SIGPLAN Software En-*

gineering Symposium on Practical Software Development Environments, Software Engineering Notes, **9**(3), 141–148, (May 1984). (Also issued as *SIGPLAN Notices* 19(3), May 1984.)

Many other syntax-directed program editors have been developed, often embedded within a more comprehensive program development environment. Surveying the state of this area is beyond the scope of the present report.

3.3 Document editors

QUIDS is an early editor, intended for use on line-oriented displays, that encoded specific knowledge about a simple logically oriented document structure.

- G. F. Coulouris, I. Durham, J. R. Hutchinson, M. H. Patel, T. Reeves, and D. G. Winderbank, ‘The design and implementation of an interactive document editor’, *Software—Practice and Experience*, **6**(2), 271–279, (April–June 1976).

QUIDS represented the document as a sequence of paragraphs, not as a sequence of lines as do general-purpose editors. QUIDS included specific commands to get particular effects, such as paragraphs that were not indented, headings, etc. QUIDS also included the ability to specify limited symbolic referencing, providing a specification of a string that replaced corresponding symbolic references in the text. With the exception of symbolic referencing, the representation of the text is a linear sequence of paragraphs. In particular there is no notion of hierarchical relationships among document objects.

Borkin and Prager were among the first to present a hierarchical tree-based document model and discuss the issues in designing an editor-formatter for it.

- S. A. Borkin and J. M. Prager, ‘Some issues in the design of an editor-formatter for structured documents’, Technical report, IBM Cambridge Scientific Center, (November 1980).

We will return to the issues involved in editing structured documents again later in this report when discussing interactive integrated editing and formatting systems.

4 INFLUENTIAL SYSTEMS: FORMATTING

In this section, we consider systems that implement the formatting function. The following section considers systems that implement both formatting and also editing in an integrated fashion. In the systems described in this section (*formatters*), the editing has been performed as a separate step (for example, by one of the systems described in the preceding section) producing a representation of the document called the “markup”—the content of the document interspersed with commands that will be interpreted by the formatter. The formatter’s translation of markup into physical representation is usually carried out as a separate step (hence the common use of the term “batch formatters” to describe these systems).

We first focus on the markup languages themselves. We then turn our attention to describing some document components that have received specific attention in formatting systems.

4.1 Markup languages

Perhaps the earliest influential document formatting program was RUNOFF, developed for MIT's CTSS operating system in the early 1960s.[7] RUNOFF commands were defined in terms of the physical characteristics of the printed document. Elements of RUNOFF's syntax and semantics continue to influence the design of document markup languages; see particularly **troff**.

- J. Saltzer, 'Manuscript typing and editing: TYPSET, RUNOFF', in *The Compatible Time-Sharing System: A programmer's guide,* Second Edition, ed., P. A. Crisman, section AH.9.01, The MIT Press, (1965).

In Unix, the basic component of document formatter has been imbedded into a larger comprehensive document-preparation environment. In addition to document preparation tools, the Unix environment includes tools that to some degree assist the author in evaluating and improving writing quality.

- B. W. Kernighan, M. E. Lesk, and J. F. Ossanna, Jr., 'UNIX time-sharing system: Document preparation', *The Bell System Technical Journal*, **57**(6), 2115–2135, (July–August 1978).
- Lorinda Cherry, 'Computer aids for writers', *Proceedings of the ACM SIGPLAN SIGOA Symposium on Text Manipulation, SIGPLAN Notices*, **16**(6), 61–67, (June 1981). (Also available as *SIGOA Newsletter* 2(1&2), Spring/Summer 1981.)

As will be discussed later in this report, the Unix document preparation system's tools represent the different components of the document's representation in separately defined languages, specialized to the characteristics of the component. These representations have, in many cases, provided the standard model that other researchers have adopted in their own systems.

Representing the document as a collection of logically related objects is the basis for "generic markup" (also called "generic coding"), introduced by IBM's Generalized Markup Language (GML).

- C. F. Goldfarb, 'A generalized approach to document markup', *Proceedings of the ACM SIGPLAN SIGOA Symposium on Text Manipulation, SIGPLAN Notices,* **16**(6), 68–73, (June 1981). (Also available as *SIGOA Newsletter* 2(1&2), Spring/Summer 1981.)

In such markup languages, the author specifies the document's components: for example a book as a sequence of chapters, each chapter as a sequence of sections, each section as a sequence of subsections, and continuing until the document's content is completely specified. Unlike RUNOFF-based languages, the author does not focus on the *presentational* aspects of the document's description (i.e., how the document will be mapped to the printed page). However, many such generic markup languages have been implemented as macro packages for such a physically oriented description language. GML is implemented as a

[7] RUNOFF was not, however, the first computer-based document formatting system. Contenders for this honor, dating back to the late 1950s and early 1960s, include "Colossal Typewriter," written by John McCarthy, and TJ-1, written by Peter Samson. On CTSS, RUNOFF itself was preceded by "memo, modify, and ditto."

set of macros for the RUNOFF-like SCRIPT language and macro packages such as Unix's `-ms` and `-me` are based on **troff**.

Scribe brought generic markup to the attention of the academic community, and provided a practical demonstration that separating the description of the document's content from the description of its appearance improved the portability and reusability of the document's markup. In implementation, Scribe incorporates two languages: a markup language used by the author, and a "database" language that controls the transformation from markup to printed form. The separation between content specification and format specification, therefore, is stronger in Scribe than in its predecessors. Indeed, Scribe's expectation is that a trained designer will provide the "style" for the document, describing the printed appearance, and that the author will concentrate on the document's content.

- Brian K. Reid, 'A high-level approach to computer document formatting', *Conference Record of the Seventh Annual ACM Symposium on Principles of Programming Languages*, (January 1980).
- B. K. Reid, 'The Scribe document specification language and its compiler', *Abstracts of the Presented Papers, International Conference on Research and Trends in Document Preparation Systems,* Lausanne, Switzerland, 59–62, (February 1981).

Scribe also incorporates other powerful abstraction mechanisms, for example separate compilation of pre-identified parts of the document, symbolic identification of cross-references and of bibliographic citations, and automatic numbering of sections and list components.[8]

Coombs *et al*., provide a relatively recent essay discussing the characteristics and advantages of generic coding.

- James H. Coombs, Allen H. Renear, and Steven J. DeRose, 'Markup systems and the future of scholarly text processing', *Communications of the ACM*, **30**(11), 933–947, (November 1987).

The focus in TEX is not as directly on the authoring process, but is most strongly on achieving a high quality of typeset output.

- Donald E. Knuth and Michael F. Plass, 'Breaking paragraphs into lines', *Software—Practice and Experience,* **11**(11), 1119–1184, (November 1981).

As will be discussed in the next section, TEX's standards for quality are applied not only to textual but also to mathematical material.

LATEX [4], a generic markup language implemented using TEX's macro facilities is one of the most commonly used languages in this category. LATEX markup is similar to Scribe's in syntax, although the LATEX error messages are not as related to the implementation as are Scribe's (this is a general characteristic of those markup languages that are implemented as macros in another language). "Styles" are easier to define in Scribe than in LATEX, again because Scribe's notations are specialized for that task.

Also of increasing importance as a markup notation is that of the SGML international standard; see the later section on "Standards."

[8] Generic coding and hierarchically based document representations are the basis for current standards work; see the discussion of standards in the subsequent section covering affiliated subjects.

4.2 Component objects

One view of a document is as a set of primitive component objects, collected together into a document. The prevailing component is usually textual—paragraphs of text shaped in varying ways. However, other objects are often distinguished, for example mathematical material, tabular material, and line drawings.

4.2.1 Mathematical material

The intricate issues in mathematical typesetting are described by Knuth. This report also gives an overview of the design issues behind TEX, mentioned above, and its related components.

- Donald E. Knuth, 'Mathematical typography', in *TEX and* METAFONT*: New Directions in Typesetting*, part 1, Digital Press and the American Mathematical Society, (December 1979). (Reprinted from the *Bulletin* (New Series) *of the American Mathematical Society* 1(2), 337–372, March 1979. Josiah Willard Gibbs lectures, presented 4 January, 1978.)

The predominating model of equations is that incorporated into Unix's **eqn**, which describes equations in positional terms rather than by describing the equation's mathematical meaning.

- Brian W. Kernighan and Lorinda L. Cherry, 'A system for typesetting mathematics', *Communications of the ACM*, **18**(3), 151–157, (March 1975).

4.2.2 Tabular material

Beach discusses stylistic considerations in tabular formatting.

- Richard J. Beach, 'Tabular typography', in *Text Processing and Document Manipulation*, ed., J. C. van Vliet, 18–33, Cambridge University Press, (April 1986).

The Unix **tbl** representation of tables is the most commonly found. In essence, this representation imbeds the content of the table into a description of the table's structure, which in the case of **tbl** is biased towards describing the table as a series of rows.

- M. E. Lesk, 'Tbl—a program to format tables', Computing Science Technical Report 49, Bell Laboratories, Murray Hill, NJ, (September 1976).

Biggerstaff *et al.*, describe an interactive system that manipulates **tbl**-like tables.

- Ted J. Biggerstaff, D. Mac Endres, and Ira R. Forman, 'TABLE: Object oriented editing of complex structures', in *Proceedings—7th International Conference on Software Engineering*, pp. 334–345. IEEE Computer Society, (1984).

4.2.3 Line drawings

Two line drawing specification languages have been implemented as Unix document preparation tools. The most commonly used, **pic**, specifies the drawing by specifying the placement of component objects.

- Brian W. Kernighan, 'PIC—A language for typesetting graphics', *Software—Practice and Experience*, **12**(1), 1–21, (January 1982).

A second language, IDEAL, describes the figure through a system of simultaneous equations, which constrain the placement relations among the points of the component objects.

- Christopher J. Van Wyk, 'A high-level language for specifying pictures', *ACM Transactions on Graphics*, **1**(2), 163–182, (April 1982).

Interactive tools for creating line drawings are discussed in the following section on interactive integrated systems.

4.2.4 Bibliographic references

Bibliographic lists can be maintained in an external database, with citation of entries from the database specified through symbolic means. In the Unix system's REFER, the citation is specified by a list of enough words taken from the entry to identify a single unambiguous match.

- M. E. Lesk, 'Some applications of inverted indexes on the UNIX system', Computing Science Technical Report 69, Bell Laboratories, Murray Hill, NJ, (June 1978).

Scribe, mentioned earlier, and BibTEX, associated with the TEX macro package LATEX, require manual specification of a unique identifier for each bibliographic entry. This identifier is then included in the citation specification. While this insures that citations are stable over time (citations will continue to identify uniquely a specific citation, even if more entries are added to the database) selecting and remembering the unique identifier is a burden on the database maintainer.

4.2.5 Other objects

The Unix document preparation system includes a variety of other languages specialized for description of further types of document objects. Examples include graphs and chemical diagrams. The system can be augmented with routines that aid in the generation of indexes, when the terms to be indexed are manually identified. However, the automatic generation of indexes without manual assistance remains a research issue.

- Jon L. Bentley and Brian W. Kernighan, 'GRAP—A language for typesetting graphs', *Communications of the ACM*, **29**(8), 782–792, (August 1986).
- Jon L. Bentley, Lynn W. Jelinski, and Brian W. Kernighan, 'CHEM—A program for typesetting chemical diagrams (user manual)', Computing Science Technical Report 122, AT&T Bell Laboratories, Murray Hill, NJ, (April 1986).
- Jon L. Bentley and Brian W. Kernighan, 'Tools for printing indexes', *Electronic Publishing: Origination, Dissemination, and Design*, **1**(1), 3–17, (April 1988).

The Unix document preparation system is not unique in providing these document objects. However, the division of document preparation into specification of a collection of component objects, each with its own specialized description language is a characteristic of this system. The strengths and weaknesses of such an architecture is itself an interesting topic of study.

5 INFLUENTIAL SYSTEMS: INTERACTIVE INTEGRATED SYSTEMS

In this section, we consider systems in which the editing and formatting functions have been merged into a seamless whole. The author's editing activities appear to take place on a printed version of the page. Indeed, in many cases, there is but a single displayed representation of the document, which is identical to the printed version; these systems are called "What You See Is What You Get," or WYSIWYG for short.

Central to many of these systems is the notion that manipulations take place directly on the displayed objects (e.g., there are no hidden tree walking operations). This characteristic of systems has been identified by Shneiderman as "direct manipulation."

- Ben Shneiderman, 'Direct manipulation: A step beyond programming languages', *Computer*, **16**(8), 57–69, (August 1983).

5.1 Systems

Many of the characteristics of existing interactive document preparation systems can be traced back to the pioneering applications developed for the Xerox Alto [5]. The Alto, a personal computer designed in 1973, included a bitmapped screen and a mouse. The Alto's document preparation environment included Bravo, a WYSIWYG editor, Markup, a bitmap picture editor, and Draw, an object-oriented drawing editor.[9]

- Butler W. Lampson, 'Bravo manual', in *Alto User's Handbook*, eds., B. W. Lampson and E. A. Taft, Computer Science Laboratory, Xerox Palo Alto Research Center, (November 1978).
- William M. Newman, 'Markup user's manual', in *Alto User's Handbook*, eds., B. W. Lampson and E. A. Taft, Computer Science Laboratory, Xerox Palo Alto Research Center, (November 1978).
- Patrick C. Baudelaire, 'Draw manual', in *Alto User's Handbook*, eds., B. W. Lampson and E. A. Taft, Computer Science Laboratory, Xerox Palo Alto Research Center, (November 1978).

The Alto was never sold commercially. The Xerox Star presented many of the Alto's ideas in the commercial market.[10]

- David Canfield Smith, Charles Irby, Ralph Kimball, and Bill Verplank, 'Designing the Star user interface', *Byte*, **7**(4), 242–282, (April 1982).
- Jeff Johnson, Teresa L. Roberts, William Verplank, David C. Smith, Charles H. Irby, Marian Beard, and Kevin Mackey, 'The Xerox Star: A retrospective', *Computer*, **22**(9), 11–29, (September 1989).

The second paper in this list, the Xerox Star retrospective paper by Johnson *et al.*, also includes a brief description of some of the systems that were developed for the Alto.

[9] As the *Alto User's Handbook* is not generally available, these papers are listed in recognition of their historical importance. However, some of the later citations in this section contain brief descriptions of these tools, as will be mentioned.

[10] The Star is now known as "Viewpoint."

This paper also includes an interesting diagram that describes how the systems described here influenced each other as well as how the systems developed at the Xerox research laboratories have influenced the design of systems such as the Apple Macintosh.

Xerox's Cedar environment also traces its roots to the Alto. As with the Alto, Cedar is a research environment, and is not commercially available. The editor for Cedar is named Tioga and is based on a tree representation of documents.

- Warren Teitelman, 'A tour through Cedar', *IEEE Software*, **1**(2), 44–73, (April 1984).

TiogaArtwork encodes illustrations into the Tioga data representation.

- Richard Beach and Maureen Stone, 'Graphical style: Towards high quality illustrations', *SIGGRAPH '83 Conference Proceedings, Computer Graphics*, **17**(3), 127–135, (July 1983).

Gargoyle represents a tool that permits design of high-quality illustrations.

- Ken Pier, Eric Bier, and Maureen Stone, 'An introduction to Gargoyle: An interactive illustration tool', in *Document Manipulation and Typography*, ed., J. C. van Vliet, 223–238, Cambridge University Press, (April 1988). Proceedings of the International Conference on Electronic Publishing, Document Manipulation, and Typography, Nice (France), April 20–22, 1988.

Bravo, the Star, and Tioga, have introduced and refined the concept of describing a document's appearance through the use of *styles*, which control the appearance of document components and document objects. As the style descriptions are separated from the document's content, modifications to the appearance of the document can be accomplished by modification of the style definitions, obtaining a similar effect to Scribe's generic markup. Johnson and Beach discuss the issues in the use of styles:

- Jeff Johnson and Richard J. Beach, 'Styles in document editing systems', *Computer*, **21**(1), 32–43, (January 1988).

Styles are one means of combining flexibility similar to that of generic markup with a WYSIWYG form of document manipulation.

A number of other research systems also have attempted to incorporate a Scribe-like representation of the document's logical structure. Etude presented a WYSIWYG-like display of the document, augmented with information about the defined structure.

- Michael Hammer, Richard Ilson, Timothy Anderson, Edward J. Gilbert, Michael Good, Bahram Niamir, Larry Rosenstein, and Sandor Schoichet, 'The implementation of Etude, an integrated and interactive document production system', *Proceedings of the ACM SIGPLAN SIGOA Symposium on Text Manipulation, SIGPLAN Notices*, **16**(6), 137–146, (June 1981). (Also available as *SIGOA Newsletter* 2(1&2), Spring/Summer 1981.)

Symbolics' Concordia system, now incorporated into a commercially available product, is intended to support the creation of documentation by teams of technical writers. It provides an editing interface for a Scribe-like document markup that incorporates hypertext-like mechanisms for accessing related parts of the document set.

- Janet H. Walker, 'Supporting document development with Concordia', *Computer*, **21**(1), 48–59, (January 1988).

Janus took a combined approach, simultaneously presenting two representations of the document: a WYSIWYG view and a GML-like marked-up view.[11]

- Donald C. Chamberlin, James C. King, Donald R. Slutz, Stephen J. P. Todd, and Bradford W. Wade, 'JANUS: An interactive system for document composition', *Proceedings of the ACM SIGPLAN SIGOA Symposium on Text Manipulation, SIGPLAN Notices*, **16**(6), 82–91, (June 1981). (Also available as *SIGOA Newsletter* 2(1&2), Spring/Summer 1981.)
- Donald C. Chamberlin, James C. King, Donald R. Slutz, Stephen J. P. Todd, and Bradford W. Wade, 'JANUS: An interactive document formatter based on declarative tags', *IBM Systems Journal*, **21**(3), 250–271, (1982).

It is interesting to compare Etude with the commercially available Interleaf system, as many of the same designers worked on both systems. One significant difference is that the initial Interleaf document representation was based on a simpler, more linear, data structure than was Etude. This is possibly as an attempt to reduce the amount of complexity with which the system's user must deal. (The Interleaf document representation continues to evolve as further capabilities are added to the system.)

- Robert A. Morris, 'Is what you see enough to get? A description of the Interleaf publishing system', in *PROTEXT II: Proceedings of the Second International Conference on Text Processing Systems*, ed., J. J. H. Miller, 56–81, Boole Press, (October 1985).

Quint's Grif system, on the other hand, attempts to apply direct manipulation principles in a system incorporating what in essence is a grammatically constrained document representation—i.e., a representation with even stronger constraints on object interrelationships than the Scribe model.

- Vincent Quint and Irène Vatton, 'Grif: An interactive system for structured document manipulation', in *Text Processing and Document Manipulation*, ed., J. C. van Vliet, pp. 200–213. Cambridge University Press, (April 1986). Proceedings of the international conference, University of Nottingham, 14–16 April 1986.

Strongly constrained document representations raise implementation challenges, particularly if the system's designer wishes to provide the naturalness of interface afforded by direct manipulation.

- Richard Furuta, Vincent Quint, and Jacques André, 'Interactively editing structured documents', *Electronic Publishing: Origination, Dissemination, and Design,* **1**(1), 19–44, (April 1988).

The Quill system, under development by the same lab that developed Janus, has many

[11] Janus is now called ICEF2 and is marketed by IBM.

of the same goals as does Grif—namely the WYSIWYG-like editing of a structured document representation.

- Donald D. Chamberlin, Helmut F. Hasselmeier, Allen W. Luniewski, Dieter P. Paris, Bradford W. Wade, and Mitch L. Zolliker, 'Quill: An extensible system for editing documents of mixed type', in *Proceedings of the 21st Hawaii International Conference on System Sciences*, pp. 317–325, (January 1988).

While many of the general issues addressed in the Quill project have been encountered earlier in a research setting, a characteristic of the Quill project is the care with which it is being engineered. Published reports describe components of Quill's architecture in detail.

- Donald D. Chamberlin, Helmut F. Hasselmeier, and Dieter P. Paris, 'Defining document styles for WYSIWYG processing', in *Document Manipulation and Typography*, ed., J. C. van Vliet, 121–137, Cambridge University Press, (April 1988). Proceedings of the International Conference on Electronic Publishing, Document Manipulation, and Typography, Nice (France), April 20–22, 1988.
- Allen W. Luniewski, 'Intent-based page modelling using blocks in the Quill document editor', in *Document Manipulation and Typography*, ed., J. C. van Vliet, 205–221, Cambridge University Press, (April 1988). Proceedings of the International Conference on Electronic Publishing, Document Manipulation, and Typography, Nice (France), 20–22 April, 1988.
- Donald D. Chamberlin, 'An adaptation of dataflow methods for WYSIWYG document processing', in *Proceedings of ACM Conference on Document Processing Systems* (5–9 December, 1988, Santa Fe, New Mexico), pp. 101–109. ACM, New York, (December 1988).
- Donald D. Chamberlin, 'Managing properties in a system of cooperating editors', in *EP90*, ed., Richard Furuta, 31–46, Cambridge University Press, (September 1990). Proceedings of the International Conference on Electronic Publishing, Document Manipulation, and Typography, Gaithersburg, Maryland, September 1990.

A general description of the Quill system as a whole had not been produced at the time of this report. However, details of the system as a whole can be extracted from the descriptions of its individual parts.

6 AFFILIATED SUBJECTS

The previous sections have presented references that describe topics at the core of work into document preparation systems. In this section, we redirect our focus to consider topics that are ancillary to this core—topics that both build from and also affect the direction of research into document preparation systems.

The metric used to select papers in this section differs from that used so far. Rather than look for influential papers as in previous sections, here I have tried instead to find papers that give a comprehensive overview of the topic issues. Further, I have not attempted to place these papers in the historical context of research in the topic area, as I have in previous sections.

6.1 Standards

Two standards of current interest to document preparation researchers are the "Standard Generalized Markup Language" (SGML) [6], which describes a generic markup language for documents, and the "Office Document Architecture" (ODA) [7], which describes a broader scheme for describing and interchanging the logical and physical representations of office documents. Barron presents an introduction to SGML.

- David Barron, 'Why use SGML?', *Electronic Publishing: Origination, Dissemination, and Design*, **2**(1), 3–24, (April 1989).

An introduction to ODA may be found in the paper by Horak. The paper by Hunter *et al.*, gives a further introduction to the present form of the standard.

- W. Horak, 'Office document architecture and office document interchange formats: Current status of international standardization', *Computer*, **18**(10), 50–60, (October 1985).
- Roy Hunter, Per Kaijser, and Frances Nielsen, 'ODA: A document architecture for open systems', *Computer Communications*, **12**(2), 69–79, (April 1989).

SGML provides a means for identifying the component objects of a document and their logical relationships through grammatical specification of a generic markup language. It does not describe how those objects are to be displayed on the printed page. A separate standard named DSSSL [8] is being developed for that purpose. ODA describes both the logical relationships among component objects of the document (the logical structure) and also how the objects are to be displayed (the layout structure).

An additional effort seeks to standardize a page description language (see the following section).

6.2 Page description languages

Page description languages describe the positioning of graphical marks on a printed page. Many of the current page description languages are based on the pioneering model of Warnock and Wyatt.

- John Warnock and Douglas K. Wyatt, 'A device independent graphics imaging model for use with raster devices', *SIGGRAPH '82 Conference Proceedings, Computer Graphics*, **16**(3), 313–319, (July 1982).

In this model, an image is created from a source and a stencil. The source is masked by the stencil to create the image. Morris and Reid review two currently available page description languages, Interpress and PostScript, which are based on this model.

- Robert A. Morris, 'Page description languages', in *An Introduction to Text Processing Systems: Current Problems and Solutions*, ed., J. J. H. Miller, 67–85, Boole Press, (October 1985). Lecture Notes of a Workshop held in association with PROTEXT II the Second International Conference on Text Processing Systems.
- Brian K. Reid, 'Procedural page description languages', in *Text Processing and Document Manipulation*, ed., J. C. van Vliet, 214–223, Cambridge University Press, (April 1986). Proceedings of the international conference, University of Nottingham, 14–16 April 1986.

PostScript is perhaps the most influential of the page description languages. It is defined in the three reference manuals from its creator, Adobe Systems Incorporated [9,10,11]; I am not aware of a shorter paper defining the language. Interpress, however, has been defined in article form:

- Abhay Bhushan and Michael Plass, ‘The Interpress page and document description language’, *Computer*, **19**(6), 72–77, (June 1986).

As noted above, a standard page description language (SPDL) is being defined [12]. Robinson and Strasen give an overview of this process.

- Peter J. Robinson and Stephen M. Strasen, ‘Standard page description language’, *Computer Communications*, **12**(2), 85–92, (April 1989).

6.3 Font design

Documents prepared through the assistance of computer-based editing and formatting systems frequently are printed on digital raster devices—laser printers at a typical resolution of 300 dots/inch, phototypesetters at typical resolutions of greater than 1000 dots/inch, etc. Bigelow and Day introduce issues in design and storage of fonts for such devices.

- Charles Bigelow and Donald Day, ‘Digital typography’, *Scientific American*, **249**(2), 106–119, (August 1983).

Bigelow also describes the issues in designing fonts for display on computer screens.

- Charles Bigelow, ‘Principles of type design for the personal workstation’, in *Gutenberg-Jahrbuch 1986*, ed., Hans-Joachim Koppitz, 253–270, Gutenberg-Gesellschaft, Mainz, (1986).

Knuth has developed an approach to digital type design, that of the “meta” font in which a family of fonts shares a common parameterized description.

- Donald E. Knuth, ‘Lessons learned from METAFONT’, *Visible Language*, **19**(1), 35–53, (1985).

The benefits and drawbacks of the approach and of the METAFONT implementation have generated debate among font designers. METAFONT is used to generate fonts for TEX-produced output.

Research issues in the general field of typography, which includes font design, has generated the “RIDT” series of conferences. The proceedings of the 1988 and 1989 conferences are available as a single volume.

- *Raster Imaging and Digital Typography: Proceedings of the International Conference, Ecole Polytechnic Fédérale Lausanne, October 1989*, eds., Jacques André and Roger D. Hersch, Cambridge University Press, 1989.

The next conference in this series was held in Boston, Massachusetts in 1991.

- *Raster Imaging and Digital Typography II*, eds., Robert A. Morris and Jacques André, Cambridge University Press, 1991.

6.4 Multilingual processing

To a large degree, the systems cited to this point have been centered around documents written in the Roman alphabet (indeed, around documents written in the English language). The papers discussed in this section describe solutions which permit use of other languages and alphabets, and indeed which permit mixing of languages and alphabets within a single document.

J. Becker describes the issues involved in processing documents in non-Roman alphabets and in mixing languages.

- Joseph D. Becker, 'Multilingual word processing', *Scientific American*, **251**(1), 96–107, (July 1984).

Arabic has received specific attention. MacKay describes an early system that was used to typeset Arabic manuscripts.

- Pierre A. MacKay, 'Setting Arabic with a computer', *Scholarly Publishing*, **8**(2), 142–150, (January 1977).

J. Becker describes more recent solutions in the area.

- Joseph D. Becker, 'Arabic word processing', *Communications of the ACM*, **30**(7), 600–610, (July 1987).

Z. Becker and Berry describe a system built around **troff** to handle tri-directional text—left to right for languages such as English, right to left for languages such as Hebrew, and top to bottom for languages such as Chinese.

- Zeev Becker and Daniel Berry, 'triroff, an adaptation of the device-independent troff for formatting tri-directional text', *Electronic Publishing: Origination, Dissemination, and Design*, **2**(3), 119–142, (October 1989).

Moon and Shin also provide a good overview of the issues in digital type design for Chinese-language fonts.

- Y. S. Moon and T. Y. Shin, 'Chinese fonts and their digitization', in *EP90*, ed., Richard Furuta, 235–248, Cambridge University Press, (September 1990). Proceedings of the International Conference on Electronic Publishing, Document Manipulation, and Typography, Gaithersburg, Maryland, September 1990.

6.5 Human factors assessments

Components of document preparation systems, particularly text editors, have been the study of empirical evaluation. One of the better-known studies, based on Roberts' Ph.D. investigations, collected and evaluated keystroke-level use data from nine text editors, and used the resulting information in comparing the editors.

- Teresa L. Roberts and Thomas P. Moran, 'The evaluation of text editors: methodology and empirical results', *Communications of the ACM*, **26**(4), 265–283, (April 1983).

A broader discussion can also be found in book form:

- Stuart K. Card, Thomas P. Moran, and Allen Newell, *The Psychology of Human–Computer Interaction*, Lawrence Erlbaum Associates, 1983.

The Roberts and Moran paper and other related papers can also be found as a small part of a general collection that gives a tutorial introduction to the study of human–computer interaction:

- Ronald M. Baecker and William A. S. Buxton, *Readings in Human–Computer Interaction: A Multidisciplinary Approach*, Morgan Kaufmann, 1987.

Other components of document preparation systems have been studied as well. One active topic has been an examination of the interaction between typeface and readability and a comparison of the relative readability of screen and paper. This work has been reviewed by Mills and Weldon:

- Carol Bergfeld Mills and Linda J. Weldon, 'Reading text from computer screens', *ACM Computing Surveys*, **19**(4), 329–358, (December 1987).

A general resource for beginning further investigation of these topics is a 2000-page book generated by the Association for Computing Machinery. This collection provides a bibliography of the recent literature, various indexes into this bibliography, and reviews of selected papers.

- ACM Press, *Resources in Human–Computer Interaction*, ACM Press, 1990.

6.6 Hypertext systems

Hypertext defines an information organization scheme favoring division of a document into a collection of fragmentary pieces that are related together by links. A hypertext may exist only in electronic form; hypertexts are most naturally read on the display of an interactive computer by following the links from one fragment to another. Although the document may never exist in paper form, it is readily apparent that hypertext systems share representations, applications, and goals in common with the paper-based document preparation systems already described.

Vannevar Bush is credited with originating the concept of a hypertext with his 1945 proposal for a machine called the memex. The memex stores all of an individual's literature as well as the trails that the person takes through the literature.

- Vannevar Bush, 'As we may think', *The Atlantic Monthly*, **176**(1), 101–108, (July 1945).

Although computer-based hypertext systems have existed since the 1960s (see, for example, van Dam and Rice's 1971 survey listed earlier [13]), recent years have seen a great increase in interest in hypertext as a computer application. A recent survey article catalogues much of the ongoing development, and has become the standard reference point for subsequent investigations.

- Jeff Conklin, 'Hypertext: An introduction and survey', *Computer*, **20**(9), 17–41, (September 1987).

Conklin's article also summarizes the contributions of other early researchers such as Engelbart and Nelson, as well as describing many in the second wave of hypertext systems that existed at the time of the report.

An initial, small-scale workshop on hypertext was held in 1987. The Hypertext '87 proceedings became generally available in 1989, and contain more detailed descriptions of many of the systems discussed by Conklin.

- Association for Computing Machinery. *Proceedings of Hypertext '87*, New York, 1989. Proceedings of the conference held 13–15 November, 1987, Chapel Hill, North Carolina.

Selected papers from Hypertext '87 appeared in revised and edited form in the July 1988 issue of the *Communications of the ACM* (volume 31, number 7).

Hypertext is an extremely rapidly developing area of research. Hypertext '89, the successor to the 1987 workshop, presented not only later developments of previously existing projects, but also showed an increasingly wide range of applications incorporating the hypertextual metaphor.

- Association for Computing Machinery. *Proceedings of Hypertext '89*, New York, 1989. Proceedings of the conference held 5–8 November, 1989, Pittsburgh, Pennsylvania.

The third conference in the series, Hypertext '91, was held in San Antonio, Texas, in December 1991.

- Association for Computing Machinery. *Hypertext '91 Proceedings*, New York, 1991. Proceedings of the conference held 15–18 December, 1991, San Antonio, Texas.

A related series of hypertext conferences are being held in Europe. The first in this series, ECHT 90, was held in France.

- *Hypertexts: Concepts, systems, and application*, eds., A. Rizk, N. Streitz, and J. André, Cambridge University Press, 1990.

ECHT 90 had a similar mix of papers to those found at Hypertext '89. ECHT 90 did seem to be a bit more open to studies that were more purely theoretical in scope, i.e., not necessarily bound to existing system implementations. After ECHT 90, the ACM Hypertext conference series and the ECHT conference series were merged. ECHT 92 is scheduled to be held in Milan, Italy, in late 1992 and Hypertext '93 in Seattle, Washington, in late 1993.

7 CONCLUDING REMARKS

To the casual observer it may appear that document preparation has become such a central, well-developed, and stable computer application that little additional innovation is possible or desirable. Investigations continue to be carried out, however, particularly into those topics that extend the scope of applicability for document preparation systems.

An issue of practical concern is interchangeability between different markup representations of a document. Conversion of one representation to another is especially difficult when the representations fall into different classes; for example the conversion between a generic-coding-based representation and one based on the physical page appearance. For one approach see [14]. Also of research interest is recognition of the structure of raw text such as that generated by a scanner. Since any markup representation of such text has been lost, conversion must proceed in the absence of the clues to structure that the markup provides.

A special case of document conversion involves transformation of document objects specified using one set of constraints to instances that conform to a different set of constraints. Such situations arise in generic coding markup on converting from one class of document to another (for example from an article class into a book class), on tracking changes to the definition of a particular class's definition (for correction of errors or in response to changes in externally imposed requirements, for example), and in conversion from one type of object to another (from itemized to enumerated list, for example). For further discussion, see [15] and [16].

A clearly specified formal description of the model underlying a document's representation provides a basis for understanding the representation's characteristics, for verifying the correctness of the representations and their use, and for comparing different representations. (For further illustration see, for example, the work by Brown *et al*. [17].)

Effective design and implementation of document preparations can require application of a broad range of traditional computer science techniques. Although the particular characteristics of the document processing application mean that it is not always appropriate to blindly apply techniques developed for other domains, examination of document processing systems from the viewpoint of traditional computer science can be productive. Indeed, in many cases, document preparation systems provide an alternate domain for examination of techniques developed for other computer science applications. In one recent example, Kaebling compares the characteristics of SGML grammars to those identified for compiler generation [18]. As another example, the issues in preparation and maintenance of very large documents (such as those accompanying commercial aircraft) resemble those in developing and supporting very large computer programs. Application of software engineering tools developed for programming can be productive in these cases.

The topics mentioned so far have been motivated by examination of the capabilities of systems for preparation of paper-based documents. Extending the document metaphor to apply to dynamic objects is also a topic for investigation. The content associated with such objects can result from computation, from information derived from the enclosing environment (for example, information obtained from a database), and from other changing sources. However, the existence of dynamic objects does not imply that the document cannot exist in paper form, as a printed version of the document represents its state as frozen at a point in time. As an example of an implementation of such active document components, English, *et al*., have described a Lisp-based implementation in Interleaf [19].

Application of document-related concepts to purely interactive applications is also an important area of research, as suggested by the related work in hypertext systems. One such topic is the use of document authoring techniques to structure interactive applications. A document is designed to convey knowledge to its reader. The author of such a document manipulates both the content of the document but also its presentation (its structure), taking into account the characteristics of the display medium (e.g., paper). Similarly, an interactive application can be viewed as intended to achieve some effect when carried out. Proposals for specifying the structure of the application have included specification of a path through a sequence of invocations of separate computer applications (see, for example, Zellweger's scripted documents [20,21]), invocation of processes as a side effect of executing an automaton (for example [22,23]), and associating actions with objects in a traditionally displayed document (see [24] and [25], for example).

In summary then, research into document preparation applications remains important in improving the capabilities of systems for specifying paper-based documents. In addition, the structuring of interactive applications using the document metaphor suggests that such results will continue to be relevant even should the long-predicted paperless society materialize.

ACKNOWLEDGEMENTS

I would like to thank the following people for supplying source material used in this report, for helping me locate references, and for making helpful comments on drafts (and my apologies if I have neglected to mention someone): Debra Adams, Jacques André, John Barkley, Rick Beach, Barbara Beeton, Charles Bigelow, Don Chamberlin, Don Gillies, Pierre MacKay, Robert A. Morris, Charles Nicholas, Fran Nielsen, Jerry Saltzer, Peter Samson, Richard Southall, P. David Stotts, Larry Welsch, J. C. van Vliet, Vincent Quint, and the anonymous referees.

REFERENCES

1. Robin Cover, Nicholas Duncan, and David T. Barnard, 'Bibliography on structured text', Technical Report 90-281, Queen's University at Kingston Department of Computing and Information Science, (June 1990).
2. Donald E. Knuth, *The TEXbook*, Addison-Wesley, 1984.
3. Norman Meyrowitz and Andries van Dam, 'Interactive editing systems: Parts I and II', *ACM Computing Surveys*, **14**(3), 321–415, (September 1982).
4. Leslie Lamport, *LATEX: A Document Preparation System*, Addison-Wesley, 1985.
5. C. P. Thacker, E. M. McCreight, B. W. Lampson, R. F. Sproull, and D. R. Boggs, 'Alto: A personal computer', Technical Report CSL-79-11, Xerox Palo Alto Research Center, Palo Alto, CA, (August 1979). Also contained in Siewiorek, Bell, and Newell, *Computer Structures: Readings and Examples*, second edition.
6. ISO, *Text and Office Systems—Standard Generalized Markup Language*, October 1986. Document Number: ISO 8879–1986(E).
7. International Standards Organization, *Text and Office Systems—Office Document Architecture (ODA) and Interchange Format*, 1989. International Standard 8613.
8. International Standards Organization, *Final Text, ISO/IEC CD 10179, Information Technology—Text and Office Systems—Document Style Semantics and Specification Language (DSSSL)*, 1991. ISO/IEC Draft International Standard 10179.
9. Adobe Systems Incorporated, *PostScript Language: Reference Manual*, Addison-Wesley, 1985.
10. Adobe Systems Incorporated, *PostScript Language: Tutorial and Cookbook*, Addison-Wesley, 1985.
11. Adobe Systems Incorporated, *PostScript Language: Program Design*, Addison-Wesley, 1988.
12. International Standards Organization, *ISO/IEC DIS 10180, Information Processing—Text Communication—Standard Page Description Language*, 1991. ISO/IEC Draft International Standard 10180.
13. Andries van Dam and David E. Rice, 'On-line text editing: A survey', *ACM Computing Surveys*, **3**(3), 93–114, (September 1971).
14. Sandra A. Mamrak, Michael J. Kaelbling, Charles K. Nicholas, and Michael Share, 'Chameleon: a system for solving the data-translation problem', *IEEE Transactions on Software Engineering*, **15**(9), 1090–1108, (September 1989).
15. Richard Furuta and P. David Stotts, 'Specifying structured document transformations', in *Docu-*

ment Manipulation and Typography, ed., J. C. van Vliet, 109–120, Cambridge University Press, (April 1988). Proceedings of the International Conference on Electronic Publishing, Document Manipulation, and Typography, Nice (France), 20–22 April, 1988.

16. Extase Akpotsui and Vincent Quint, ‘Type transformation in structured editing systems’, in *EP92: Proceedings of Electronic Publishing, 1992*, eds., C. Vanoirbeek and G. Coray, 27–41, Cambridge University Press, (April 1992).
17. Allen L. Brown, Jr., Toshiro Wakayama, and Howard A. Blair, ‘A reconstruction of context-dependent document processing in SGML’, in *EP92: Proceedings of Electronic Publishing, 1992*, eds., C. Vanoirbeek and G. Coray, 1–25, Cambridge University Press, (April 1992).
18. Michael J. Kaelbling, ‘On improving SGML’, *Electronic Publishing: Origination, Dissemination, and Design*, **3**(2), 93–98, (May 1990).
19. Paul M. English, Ethan S. Jacobson, Robert A. Morris, Kimbo B. Mundy, Stephen D. Pelletier, Thomas A. Polucci, and H. David Scarbro, ‘An extensible, object-oriented system for active documents’, in *EP90*, ed., Richard Furuta, 263–276, Cambridge University Press, (September 1990). Proceedings of the International Conference on Electronic Publishing, Document Manipulation, and Typography, Gaithersburg, Maryland, September 1990.
20. Polle T. Zellweger, ‘Active paths through multimedia documents’, in *Document Manipulation and Typography*, ed., J. C. van Vliet, 19–34, Cambridge University Press, (April 1988). Proceedings of the International Conference on Electronic Publishing, Document Manipulation, and Typography, Nice (France), April 20–22, 1988.
21. Polle T. Zellweger, ‘Scripted documents: A hypertext path mechanism’, in *Hypertext ’89 Proceedings*, pp. 1–26. ACM, New York, (November 1989).
22. Richard Furuta and P. David Stotts, ‘Programmable browsing semantics in Trellis’, in *Hypertext ’89 Proceedings*, pp. 27–42. ACM, New York, (November 1989).
23. P. David Stotts and Richard Furuta, ‘Dynamic adaptation of hypertext structure’, in *Third ACM Conference on Hypertext Proceedings*, pp. 219–231. ACM, New York, (December 1991).
24. Douglas B. Terry and Donald G. Baker, ‘Active Tioga documents: An exploration of two paradigms’, *Electronic Publishing: Origination, Dissemination, and Design*, **3**(2), 105–122, (May 1990).
25. Eric A. Bier and Aaron Goodisman, ‘Documents as user interfaces’, in *EP90*, ed., Richard Furuta, 249–262, Cambridge University Press, (September 1990). Proceedings of the International Conference on Electronic Publishing, Document Manipulation, and Typography, Gaithersburg, Maryland, September 1990.