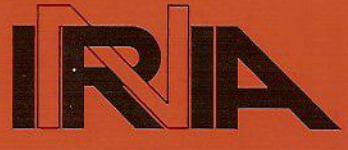
STRUCTURED DOCUMENTS

Edited by
J. André, R. Furuta & V. Quint



THE CAMBRIDGE SERIES ON ELECTRONIC PUBLISHING

STRUCTURED DOCUMENTS

Edited by

J. ANDRÉ

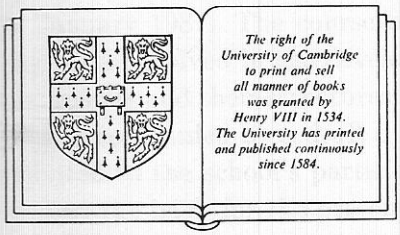
INRIA, Rennes

R. FURUTA

University of Maryland

V. QUINT

INRIA, Grenoble



CAMBRIDGE UNIVERSITY PRESS

Cambridge

New York New Rochelle

Melbourne Sydney

Preface

A document may be described as a collection of objects with higher-level objects formed from more primitive objects. The object relationships represent the logical relationships between components of the document. For example, the present document is described as a book at the highest level. The book is subdivided into chapters, each chapter into sections, subsections, paragraphs, and so forth. Such a document organization has come to be known as the *structured document* representation.

This book describes the principles underlying this logical structuring of documents and the advantages that can be drawn from it.

The book originated with a series of lectures given during a winter school on document structures organized by INRIA.[1] The school, directed by Jacques André and Vincent Quint, took place at Aussois (Savoy, France) in January 1987. The course brought together sixty or so researchers and engineers involved in the development or use of computer systems handling documents and their structures. Most of the lectures presented at this school were accompanied by a draft document, collected together in a booklet distributed to the school's participants [AQ87].[2] These draft documents have formed the basis for the present book.[3]

[1] The *Institut National de Recherche en Informatique et Automatique* is the French national laboratory for research in Computer Science. For more information on INRIA and its schools, write to *INRIA, Service des relations extérieures, B.P. 105, F-78153 Le Chesnay Cedex, France.*

[2] Bibliographic citations, shown between square brackets, found in this chapter and through the book refer to the collected bibliography located at the end of the book.

[3] Several papers were given during the course which could not, for a variety of reasons, be included in this book. In particular, these included a paper by Philippe Maurice from the Caen SEPT on ODA, a paper by Jacques André on document systems compared to programming

The book reflects the structure and contents of the course: four survey papers giving a fairly complete overview of the field, accompanied by shorter papers, each of which gives a particular viewpoint on the problem. The contributions as a whole, and therefore the corresponding chapters of this book, were specifically designed for the course. We will describe the organization of the book in more detail in the first chapter.

Our heartfelt thanks are due to David Beeson, ably assisted by Janet Macke, who translated those texts that were originally produced in French.[4] We are heavily indebted to the INRIA for financing this project, and to Agnès Couaillier and Valéry Brucker for their help in the Aussois school organization. Paul Gatewood prepared software at the University of Maryland that assisted in creating the book's merged bibliography and Philippe Louarn of IRISA helped us produce the camera-ready output in Rennes. We thank members of the staff in the Systems and Software Technology Division of the (United States) National Bureau of Standards for their time in discussion of relevant issues. And of course, we are indebted to the participants in the Document Structure course, whose efforts gave the course its value and to the lecturers who gave this book its contents.

Jacques André
Richard Furuta
Vincent Quint

languages, three notes respectively from Jacques André, Pat Norrish and Richard Southall on tabular typography, and finally one by Isabelle Levy (Italsoft, Paris) on *Text and logical markup for professional publishers.*

[4]The course was bilingual—presentations and papers were presented in English or French at the preference of the lecturer.

Table of Contents

By Way of an Introduction. Structured Documents: What and why?

Jacques André*
Richard Furuta†
Vincent Quint‡

Later in this chapter, we will describe the organization of the book in more detail and introduce the individual chapters that follow. First, in order to give a clearer definition of the subject, we briefly will summarize the developments that led to the concepts and tools with which we are concerned here.

1 A little history

Document production systems started timidly in the 1960s. The first systems were simple extensions to program editing tools, for example formatters such as ROFF or RUNOFF. They then grew rapidly and now form a class of very varied tools.

The first systems were low-level formatters, handling text input through the editor used for writing programs. The role of these formatters was very simple. Essentially it was to construct lines of equal length and to produce justified pages on the basis of a *ribbon* of text. A few commands could be introduced into the text to control page layout: these were elementary commands, very close to those understood by the printer (spaces, linefeeds, formfeeds, etc.). The facilities provided by printers of the period necessarily limited the potential for formatting: a limited character set, fixed pitch and only rough positioning of characters.

The language in which commands were expressed showed little formalism and was at a low level, similar to an assembly language. These lan-

*INRIA/IRISA, Campus de Beaulieu, F-35042 Rennes Cedex, France
†Department of Computer Science, University of Maryland, College Park, MD 20742, U.S.A.
‡INRIA/IMAG-LGI, BP 53X F-38041 Grenoble Cedex, France

guages have advanced in line with developments in programming languages. To start with, macro-instructions were added, making it possible to adapt the formatter and to provide the user with higher-level commands. The commands too were enhanced to take greater advantage of the possibilities provided by new printers, or even phototypesetters, as in TROFF [KLO78].

Formatters were also enhanced to handle non-text elements. The formatting tools available under UNIX [Rit78] provide a good example of this development. A set of preprocessors handles various types of objects within documents intended for TROFF: Tbl [Les76] handles tables, Eqn [KC75] handles formulas, Pic [Ker82] handles drawings. Other tools assist the author by dealing with bibliographic references or by detecting errors in spelling or even in style [Che81]. It was also at this time that Donald Knuth [Knu84] defined TEX. TEX's principal concern was typographic quality, obtained at the time by means of electrostatic printers such as Versatec, even though their resolution was only 200 dots per inch.

Alongside these tools, which were principally attractive to academic computer scientists, there began to appear professional publishing systems, which were subject to confidentiality constraints and very specific to computer or typographic hardware. The high point of this development was the IBM *Generalized Markup Language.*

Another approach was launched at the end of the 70s, in particular by B. Reid [Rei80a], who regarded the language of formatters as a high-level language describing a document in logical terms rather than as a function of the desired presentation. The role of the formatter became more important since it had to be decided, from the logical description of the document, on the page layout to adopt before producing an image on a printer. At the same period, the emergence of laser printers also stimulated the development of formatters of greater typographic quality.

An indispensable complement to a formatter is an editor allowing input and updating of the document description to be submitted to the formatter. Certain editors, originally limited to this role, later took on board formatting functions, to become editor-formatters, producing documents that are immediately printable, since they are already laid out to page. But these systems have not replaced formatters, because the formatting functions they provide correspond to those in the earliest formatters, i.e. they are low-level functions. Nonetheless, these editor-formatters enjoyed a great success, firstly on specialist word processing machines, and then as application software on micro-computers.

The latest stage in development of this type of system was based on the use of graphics screens to display an image corresponding to what would be produced on a laser printer. This is the *what you see is what you get* (WYSIWYG) approach, first of all developed on the Alto with Bravo [Lam78] and then adopted in products like the Star [HN82] or the Macintosh [Wil84]. It allows advantage to be taken of all the potential of the hardware (numerous character fonts and sizes, integrated graphics) and provides a high-level user interface.

2 What is a document?

We may define a document, circularly and somewhat factiously, to be the product of a document manipulation system. Actually the common use of the term "documents" corresponds not only to the traditional production of printers, publishers etc., but also to the papers, memos, and notes produced in offices. Indeed, in some authors' definitions, computer program code is also considered to be a special form of document. "Document" is a rather general word, and rather than attempting to provide a formal definition, that certainly would fail to fit some uses of the word, we will rely on the reader's intuitive understanding of the word.[1]

3 Why structured documents?

Of all these systems electronically manipulating documents, those with which we will be most concerned here are those which consider the document not as a stream of characters throughout which presentation information has been scattered, but as an organized structure.

In the first case (corresponding to what is called "procedural programming languages" in the program domain), we tell the system how to print the text. For example: *use character font no 3 in 16 point, move to the top of the page, right justify, print 'CHAPTER 3', advance two lines, print the chapter title, etc.* In batch systems, these instructions, or attributes, are mixed into the text itself. It may not be obvious that the same is true for many interactive systems where, although the author can say (and even see on screen) that a particular word is in italics or that a particular title (or rather the words forming the title) are in bold and right justified, he nevertheless has no automatic way of renumbering sections, of putting foreign words into italics or of handling page references.

[1] As example, the American Heritage Dictionary defines "document" as "a paper bearing evidence, proof, or information."

In the second case (corresponding to "declarative programming languages"), we no longer tell the system how to do things, but simply tell it what we want done. This means working at three levels: in the first place we have to describe the *logical structure* of the document (e.g. *a book is formed of a preface, chapters, etc. A chapter is formed of a title, sections, sub-sections, etc.*). Next we have to move from this logical structure to the corresponding *physical structure* (the printed text) (*a chapter title is right justified, in 16 point, using font times bold, preceded by the chapter number, etc.*). Systems often provide a method to handle these first two tasks. Finally, we have to apply this structure, and its page layout, to the document being written, (*this is a chapter title, here is a new section, here is a footnote, etc.*) and only this job remains to be done by the author.

Writing a structured document can then be handled, as in structured programming, in a top-down way: we start off by sketching the outline, and we postpone the typographic details of page layout for as long as possible (or even forever).

But this structural viewpoint does not only allow us to lay out documents to page and print them, but also provides real processing possibilities: given the logical structure of a document, we can build up a table of contents or an index, we can automatically number sections or notes, we can scan through the document and therefore order it in different ways, we can set up different styles of page layout without changing the text itself, we can code the document by means of different formalisms, we can send it to other people, etc. and this is all possible without the tiresome use of *search and replace* operations that generally have to be carried out in non-structured texts.

4 Contents of this book

It is the principles underlying this logical structuring of documents and the advantages that can be drawn from it that are described in this book.

1. The first two papers are in a sense a survey of what can be called document structuring:
   - in the first paper, Richard Furuta examines the concepts underlying document structures and classifies the various document models implemented in the typical systems on the basis of the degree of organization they use. He also considers models developed for the representation of various components such as mathematical expressions, tables and graphs.

- in the second paper, Vincent Quint describes the various types of software implementing structured models. The two main classes of systems are structured formatters and editors. Although the first are now well known (Scribe dates from the end of the 70s), the latter are far more recent and still raise unresolved questions. Quint considers these issues, and examines the solutions that have been proposed in various experiments. The chapter also describes the types of processing carried out by all systems, whether they are interactive or not.

2. The documents produced by these tools are not intended for purely local processing. They can be shared by many authors working on different machines, or they may be broadcast by all sorts of data transmission methods. This leads at once to the problem of standardization. We have to define a representation of documents which is comprehensible to the systems used by the different users of a single document.
   - Brian Reid analyses the problem of document communication and storage in the light of the type of structure adopted. He draws on the Kiosk system for illustrations of the points he makes.
   - Vania Joloboff describes various representations that can be used to exchange documents between systems. He is interested both in representations describing the graphics structure of a document and in those representing its logical organization, where the former allow a document to be printed or displayed while the latter also make re-processing possible. He describes the main standards used, whether *de facto* standards or standards proposed by official organizations.
3. Over and above these four basic papers, this book also includes a series of papers on more specific points, corresponding to the various talks given during this course:
   - a great deal of work has been done on the development of systems for the direct creation of documents, but much remains to be done on the retrieval of printed documents, in particular if we want to go beyond the simple recognition of characters and actually build up the structure of the documents input. Rolf Ingold describes an approach to the problem.
   - a text is more than its structural form. It also has a certain semantics, induced by the elements of its physical form (we might

call this its semiology if this term did not have certain pejorative connotations). This is why we asked a number of typographers on the one hand, and a number of linguists on the other, to describe their viewpoint on this aspect of structuring:

- Richard Southall considers, as a typographer, the various stages through which a text goes from an author to production (in its final form). Basing his presentation on the concept of *meta-design* in METAFONT for the creation of character fonts, he demonstrates the value of a *meta-design* system for typography.
- Pat Norrish studies the semantic structures of texts according to their typographic representation. She analyses more deeply two of such units: paragraphs and tables.
- Jacques Virbel starts by describing a series of hypotheses allowing us to tackle the issue of the abstract representation of textual structures, from a linguistic point of view. This allows us to handle text at a high structural level and to control its physical production.
- Saïd Tazi, after showing the limits of the hierarchical model, proposes a formalism which establishes a correspondence between the formally perceptible aspects of texts and their interpretation in terms of performative discourse acts.

Finally, the bibliographic references for all these papers have been collected together at the end of the book, followed by an index of the main systems and concepts mentioned.

By Way of a Colophon

This book is one of the first to be published in "The Cambridge Series on Electronic Publishing," from Cambridge University Press. The topic of the book is an aspect of that area, and further the details of its preparation and formatting provide a snapshot of the existing capabilities and potentials for electronic preparation and transmission of documents.

The book originated with a series of lectures given during an INRIA winter school held at Aussois, France, in January 1987. The draft documents written for that school were prepared with many different formatters, such as LaTeX, MacAuthor, Mint, Scribe, Word, or even by typewriter, and each was prepared in its own style. Furthermore, the language used in the papers reflected the differing languages of the lecturers—half of the papers were written in French and the rest in English.

Our work, as editors, began when we decided to merge the individual papers' bibliographical references into a single list. As possible, the individual authors' lists have not been retyped—the electronic form of the original lists were collected, converted into BibTeX, and merged together, in part using software we developed to allow selection of parts of BibTeX source files.[1] Transport of files was over networks such as the EARN, BITNET, the Internet (composed of the Arpanet, Milnet, and associated local area networks), and in large part over UUCP links.

Secondly, we translated French-language papers into English. The translations have been made by David Beeson, the translator of *TSI* (a French magazine dedicated on Computer Science published both in French by Dunod, and in English by John Wiley). Beeson delivered English transla-

[1] Further information about this work can be obtained from Richard Furuta.

tions prepared with Word on an IBM-PC. After the individual authors examined and updated their chapters, we converted the Word version into LaTeX, primarily by hand. The papers were then transferred over UUCP/Internet from INRIA to the University of Maryland. The chapters' bibliographic citations were converted from textual strings to LaTeX `\cite` commands at this time, and other cross-references were converted to symbolic form as well. Index terms were identified manually and were encoded into the LaTeX source files using the LaTeX `\index` command. Postprocessing necessary to form the index was performed by Scott Simpson's `index` program.

The figures have been prepared in a number of ways: LaTeX's `picture` mode, with Apple Macintosh products such as MacDraw and MacPaint, with the Xerox ViewPoint workstation, and others. Consequently, some of the figures have been stripped into the camera-ready form.

The document style used in this book is modified from that specified for the papers presented at EP88, the International Conference on Electronic Publishing, Document Manipulation and Typography [vV88]. All papers were formatted to that style and adjusted as needed at the University of Maryland, and then transferred back to INRIA. Then, the `DVI` file were processed on a Linotronic 300 typesetter by Imprimerie Louis Jean at Gap (France), using the TeX Computer Modern outline fonts redesigned with Metafont by Victor Ostromoukhov, to be compatible with PostScript.