Writing and Analyzing Effective Computer System Documentation


Two years ago, my department purchased computer software that sits unused because nobody has time or patience to interpret the poorly written manual that accompanied it. We learned the hard way—the expensive way—to take a critical look at documentation before making purchases. Analyzing existing documentation is just one area addressed in Ann Stuart’s book.

The book covers three traditional stages of computer system development and presents representative documents for each stage: analysis (proposals and functional specifications), design (system specifications and program specifications), and implementation (user manuals). There is a section on informal writings (memoranda, status reports, minutes, summaries), as well as one on general rules that apply to all computer documents and a section demonstrating nine different approaches one may take to present material—approaches like compare and contrast, cause and effect, definition, and persuasion.

Stuart identifies a rather broad reading audience that includes computer scientists who design and implement systems, technical writers who document systems, supervisors who are responsible for delivering systems, and clients who must know whether systems are work-documented. The text is also appropriate for college students and can be used for reference or for self-instruction.

A major strength of Writing and Analyzing Effective Computer System Documentation is the wealth of both good and bad examples—more than 100 of them. I like the way the author actively involves the reader by presenting a short document or segment and then asks the reader to identify its strengths and weaknesses. The reader can compare notes with the author’s comments and then, in some instances, look at corrected or rewritten versions.

One very nice feature of this book is that the reader can start just about anywhere. Have to write a system specification? Fine. Turn to page 129 for general comments and advice followed by a wide variety of annotated examples. It’s not really necessary to read each section in the order presented. This is a particularly welcome feature for those who are on the job and who need to learn about a special kind of document quickly. A detailed checklist at the end of each chapter further enhances the value of the book as a reference tool.

Now for some bad news: To use the book as a reference, you need a good index. This book doesn’t have one. For example, “flowcharts” appears, not under “F,” but under “chart systems,” which is under “S” for “systems specifications.” The lone index entry under “B” is “be responsible for what you write.” “Budgets” and “benefits” are indexed under “outline,” which is a subset of “proposal.” It’s a strange index.

Plenty of headings, subheads, and lists throughout the book make it easy to read, but I was disappointed to see inconsistencies in the way material is physically presented. A gray screen usually (but not always) highlights sections of example documents. Some times the author’s comments about examples fall inside the screened area, sometimes not. And different typefaces are used in a way that confuses rather than clarifies.

A quick scan of a dozen other technical writing books reveals that a section on common grammar or word usage problems is standard. Here, this information is scattered throughout the book. A chapter bringing together grammar hints would have been helpful for those readers whose main job is not writing.

Despite minor weaknesses, Writing and Analyzing Effective Computer System Documentation is a useful book for novices who have to write or analyze computer system documentation or for experienced folks who have been writing documents that could use improvement.

I like the way the author actively involves the reader by asking him to identify a segment’s strengths and weaknesses.

S. Modlin
Department of Instructional Technology
Rochester Institute of Technology
Rochester, New York

---

The Journal of Systems and Software

The Journal of Systems and Software (JSS) is an interdisciplinary journal focusing on managerial and technical problems in the computing systems domain. Innovative in scope and editorial direction, this timely journal applies theory developed from both research and real-world experience to actual computer systems and software life cycle situations.

Each issue contains important papers written by experts from academia, government, and industry. JSS combines current information with historical facts to examine the following: theoretical and empirical advances in engineering methodologies, management behavior, software structure, computer programming techniques, testing and maintenance strategies, and related computer hardware design issues.

1985 Subscription Information
ISSN: 0164-1212
Volume 4 (4 issues) Institutional Rate: $33.00

For air delivery in the U.S., Canada, or Mexico, add $2.00. Subscribers outside the U.S. should add $9.00 for surface delivery (not available to Europe), $5.00 for surface air lift delivery to Europe, $14.50 for surface air lift to Japan, or $26.00 for air delivery to the rest of the world.

For sample copies, send to:
in North and South America:
Elsevier Science Publishing Co., Inc.
P.O. Box 1063, Grand Central Station, New York, NY 10016

in the rest of the world:
Elsevier Science Publishers
Direct Mail Department, P.O. Box 211, 1000 AE Amsterdam,
The Netherlands.

Note: Send subscription orders to either your usual supplier or the New York address.

North-Holland

Reader Service Number 1

IEEE-EXPERT

Intelligent Systems and their Applications

a new periodical from the IEEE Computer Society

For more information, call or write
Circulation Department
IEEE Computer Society
10662 Los Vaqueros Circle
Los Alamitos, CA 90720
(714) 821-8380

September 1985