Books

Distributed Computing Environments, Daniel Cerutti and Donna Pierson, Editors, McGraw-Hill, Inc., New York, 1993. 398 pp. (ISBN 0-07-010516-2).

This book provides a comprehensive one-volume survey of current developments in distributed computing, which includes such *au courant* topics as client/server, open systems, technology trends, and standards activities. Targeted primarily for users involved in strategic planning, system analysis, and purchasing decisions, it would be helpful to anyone who wants detailed background or a broader understanding of the feature topics and product announcements that appear regularly in today's trade press.

The contributors include 15 experts who hold influential jobs in the computer industry, many of whom have been deeply involved in shaping the topics discussed.

The biggest problem in preparing a survey of fast-changing developments is where to begin and where to end. One contributor, Hal Lorin, chooses to begin in the Paleolithic Era—not the Paleolithic Era (Old Stone Age) of human social and economic development, but the similar era of computing, which Lorin defines as the period from 1955–1970, the period of centralization. Later eras include Mesolithic Computing, 1971–1980, the period of time-sharing interactive systems; the Neolithic Period, 1981–1990, the era of VLSI (very large scale integration) and changing economics; and the Early Bronze Age, 1991–2000, our current era of accelerating change.

This witty style is mirrored in some of the other chapters, including, surprisingly, a chapter on standards by Gary Leikam, who outlines not only the history of current standards groups but also the major issues involved. Although it is often claimed that standards suppress innovation, Leikam feels they actually serve to accelerate the introduction of new technology by providing a stable common foundation to build upon. Where effective standards originate is sometimes uncertain, as shown by contrasting the OSI (open systems interconnection) network model with the Internet. "OSI is the quintessential example of a clean-slate, prescriptive approach to standardization by committee in advance of proven feasibility," he writes, "versus the Internet's pragmatic, engineering-oriented style of controlled experimentation and demonstrated workability and compatibility prior to formalization."

Like Caesar's Gaul, this volume is divided into three parts.

Part 1, "The Environment," consists of sections on the rise of distributed computing and open systems, and on the business of distributed computing.

Part 2, "The Technology of Network Computing," begins with an essay on models of computing, then moves on to issues of connectivity, communications, network operating systems, client/server computing, distributed systems management, and distributed transaction systems.

Part 3, "The Distributed Computing Industry," covers technology trends and directions, architecture, standards activities, research efforts, and leading-edge environments.

A volume with 15 contributors represents varied viewpoints, but a poll of the authors would probably support the following views:

1. Changing computer models result from changing economics.
2. Technology, rather than user needs, drives these changes.
3. Standards and open systems are good, but
4. Users will not wait for standards before adopting new technology.

Of course, even a volume created to interpret changing technology is always threatened by further changes. For example, Bob Blake in his chapter on leading-edge environments writes "when applications reside on different systems, either the data must move to the program, or the program must move to the data," thus implicitly denying the existence of the object paradigm. Ted Hanss' chapter on research directions surveys ten major efforts, ranging alphabetically from the Amoeba project at the Free University in The Netherlands to TRON in Japan, and including Andrew, Athena, Chorus**, Institutional File System, LOCUS**, Mach, Plan 9, and Sprite.

Covering as it does current topics, this book would be particularly useful to anyone writing reports or preparing presentations on any of these new technologies, and it could be particularly useful to anyone involved in purchasing new technology, for, as the book argues, the rate of change has been dramatic, but we are promised a greater rate of change still. It becomes tremendously difficult to know when to commit to a technology because of the danger that the technology will be rendered obsolete before it has begun to pay for itself. Editor Cerutti recommends a "half-step ahead strategy," which can be achieved by being the first to innovate in a certain area. "Once a product or technology begins to be a leader in the marketplace, it must be made available to other parties for adoption. Success depends on the decisions of how and when to make the technology available." And, I expect, success for the user would lie in correctly determining when to adopt it.

George McQuilken
Demeter International, Inc.
Marblehead
Massachusetts

**Trademark or registered trademark of Chorus, Inc., or Locus Computing Corp.

Enterprise Computing, Alan R. Simon, Bantam Books, New York, 1992. 303 pp. (ISBN 0-553-08953-6).

Implementing the Enterprise, Alan R. Simon, Bantam Books, New York, 1993. 396 pp. (ISBN 0-553-09152-2).

The first of these two books by Alan Simon, *Enterprise Computing*, is an ambitious book. It tries to bring together a large number of the elements of what the author (and others) call enterprise computing. The author characterizes his topic as the search for the integration of enterprise systems resources. Despite the fuzzy edges of this area, most of us would agree with the author's general structure and his selection of particular aspects.

The book addresses the present and emerging technologies of "open" "distributed" systems, covering many key standards and the relevant state of the art in communications, database, applications development, mail, and client/server. The discussions are informal, accessible, forthright, and occasionally even amiable. I believe it is a quite useful book for a broad range of professional and management needs.

One must read the preface. It reveals the author's concerns with the material and some of his reasons for treating certain areas as he does. The preface convinces the reader that the author is an honest and diligent man, anticipating the objections of some readers (including this reviewer), and struggling with the vast amorphousness of the topic area.

Part One introduces topics about motivation, some key standards (POSIX**, ISO GOSIP [International Organization for Standardization government OSI profile], X.400, X.500), and takes on some of the CASE (computer-aided software engineering) and API (application programming interface) issues in a context of interoperable and portable distributed systems. The treatment is competent and informed.

Part Two provides a full description of IBM Systems Application Architecture* (SAA*) and Digital Equipment Network Application Support** (NAS) as exemplars of a set of interfaces and protocols constituting a "systems architecture." Something of the nature of the architectures of

some other key vendors is also discussed. Also in Part Two we have a reasonable discussion of communications issues across SNA, DNA, and TCP/IP (Systems Network Architecture, digital network architecture, and Transmission Control Protocol/Internet Protocol). There is an interesting first discussion of emerging higher performance telecommunications technologies. Perhaps most useful in Part Two is the calm and professional way the author discusses issues of migration. He introduces considerations in a deliberate way, avoiding the hysterical hype of words like "legacy," "downsizing," "rightsizing." Not quite as satisfactory, because it lacks some necessary concepts, is the honest try at a discussion of portable applications.

There are many things right with this book. In particular, the intelligence and professionalism of the author. The book deserves reading by strategists in using organizations as well as by marketing staffs of systems vendors. Consultants should browse it to see how topics are organized and stressed.

Unhappily, by the nature of its ambition, and because of rapidly accelerating rates of changes, the book already seems incomplete and, in parts, unsatisfying. Every reviewer, of course, will have his or her own biases, and I am necessarily revealing my own. My purpose is not to diminish the value of the book, but to suggest it cannot be the only book that is read in this area, and that shifts in perspective since its publication leave certain of its assessments off focus.

The broad coverage necessarily leads to a certain amount of inconsistency in detail in the way that various technologies are covered. For example, in Chapter Three, we have some actual SQL (structured query language) examples, but we have no corresponding level of detail for program-to-program communications, remote procedure calls, or any of the other key software technologies. Some technology is treated almost "in passing," and it is not clear whether all of the decisions about detail and level seem proper in the spring of 1994.

One great difficulty of the book is the treatment of vendor-sponsored architectures like SAA and NAS. The discussions of these seem very much influenced by vendor material and vendor consultation, so that they lack some of the professional judgment that the author provides in other sections. SAA and NAS, like other architectures of the mid-1980s, seem to some to lack the direct relevance they once had. The book does not address the forces operating on vendor architectures as new technologies and cultures enter the market. The dramatically increasing role of X/Open as a certifier of standards, within and without the UNIX** community, is not at all mentioned.

On balance the book is useful. It is honest, competent, and like all books in rapidly changing disciplines, flawed. Whether some of the conditions of 1994 might have been foreseen in 1991 is a judgment for the reader to make.

Implementing the Enterprise, the second volume by Simon, is a companion to *Enterprise Computing*, written with the intention of exploring the topics of that book in somewhat greater detail. The subject of both books is the technology for achieving what some call enterprise computing, and others cooperative computing or distributed open computing. All these phrases represent the notion that the resources of computing can be arrayed across geographical boundaries and system differences to form an apparently coherent computing resource. This is something many of us hope is true, but few of us are disciplining ourselves to achieve.

The author makes two important points in the preface: there can be no canned philosophy for achieving enterprise computing and it is necessary to deeply research the literature of ideas and available products in order to act as one's own systems integrator. Both these points are valid and suggest an intelligent framework within which to explore the technologies.

The book begins with an overview of enterprise computing that should relieve the reader of the need to read the 1992 work. It then follows with material relevant to communications protocols, messaging and directory protocols (X.400 and X.500), network management, object-oriented interfaces, database management, user interface standards, and client/server. It concludes with a (very) short chapter on methodology for implementing the enterprise and a model from Com-

puter Associates about how computing will look in the 1990s.

There is much valuable information in this volume. The author's style is readable, even charming at times, and he makes the material accessible to the professional mind-set. The material is current and provides coverage for most of the key issues now concerning information technology staffs.

However, the book falls short of its intent to explore the topics of the earlier book in more detail. There seems to be some confusion in the author about the intent of this book, and whether he expects its readers to be familiar with the 1992 work. In addition, in a book called "Implementing the Enterprise," where point of view is of key importance, the tone is often set by those interviewed at various vendors or by the writer of the literature. Topics are frequently discussed in detail only within the structure of a particular approach. In some areas, key technologies are not mentioned.

My clients concerned with implementing the enterprise are anxious to know the standards and emerging technologies; they are also, by way of "implementing," anxious for reality tests. They need guidance as to what is solidly available product, which of various competing technologies is more solid, over what time frames. What are the pitfalls and the dangers. This is crude and vulgar stuff, but it is the natural stuff of "implementing" the enterprise.

It is a daunting task to synthesize the vast jungle of technologies and assess them in a consistent and informed manner. But it is surely the task of a book called "Implementing the Enterprise." Although this book had much that was of interest, it cannot be considered entirely successful.

Harold Lorin
The Manticore Consultancy
and Senior Adjunct Professor,
Hofstra University
New York

*Trademark or registered trademark of International Business Machines Corporation.

**Trademark or registered trademark of the Institute of Electrical and Electronic Engineers (IEEE), Digital Equipment Corporation, or X/Open Company Ltd.

Centralized and Distributed Operating Systems, Gary J. Nutt, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1992. 418 pp. (ISBN 0-13-122326-7).

This book is a sophisticated and elegant presentation of the fundamental technologies of modern operating systems. It revisits basic principles in the light of contemporary systems structures and provides an integrated framework for topics as diverse as multiprocessor scheduling and the client/server model. It is clearly written for the serious student of these topics, containing minimum hype and a narrative style that assumes a professionalism on the part of the reader. This reviewer has adapted it for a graduate course in operating systems design.

The book begins with a useful review of the viewpoints we have about what operating systems are. It reminds us of our notions of a system as resource manager and as the creator of an abstract machine whose characteristics are more convenient for programmers than the raw hardware architecture. Then we are given a fast trip through history, a characterization of contemporary systems structures and operating systems, and we are sent out on our journey over the technologies. For the most part, the technologies are discussed with insight and precision in a clean and nicely flowing narrative style.

The order in which the technologies are discussed moves outward from a center. Each discussion contains the accumulated wisdom of our long experience in operating systems design and, where appropriate, some new insights. The early chapters explore basic process concepts, process scheduling, process synchronization, protection, and communications.

The discussions are supported by formalized programming notations where useful. The algorithmic presentations are clean and useful extensions of the discussions. Beyond small algorithmic sections, the author uses a wide range of standard representations to clarify and support his ideas. Process precedence is shown by Petri net and Precedence graph representations of process activities over time. The formalisms are not at all heavy, and serve to give quick visual impressions of ideas that would be difficult to illustrate in only narrative form. Only rarely, although it does happen, does the author speak particularly to

the logician or computer scientist. But despite occasional arguments beyond the notational background of most of us, the book is still useful for the graduate student and general reader of the industry with some degree of software experience.

The book touches on the classical issues, discussing deadlock fully with a section on the Banker's Algorithm, detection, and recovery. It provides nice formalisms for scheduling and useful discussions of various dispatching and queuing models for both uniprocessors and multiprocessors. Memory management is discussed fully with articulate exposition of basic mapping mechanisms and fundamental address abstractions. Multics mechanisms, which should be included as the seminal technology in any book in this area, are explained briefly. I regret that the author chose not to discuss capability-based addressing more completely, but this may be a sentimental view, since it seems to have been so rejected by the industry. On the other hand, there is a nice rich discussion of the behavior of various paging algorithms in various contexts and a convincing review of "working set" principles.

Throughout the early chapters, we encounter material that is the usual material for an operating systems book. The order of the material is sensible, and the sophistication of the discussion is well beyond what we have become used to over an unfortunate period for operating systems books. This book seems to break the trend toward books that are glosses over vendor manuals or that have large letters, many pictures, and simple words.

At Chapter 10 we begin to encounter the material that makes the book an important event in the culture of software. Networking, client/server, and distributed computing issues often excluded from operating systems texts are introduced. The author provides the basics in Chapter 10, introducing the lower layer and middle layer protocols, the attributes of Ethernet and Token Ring LANs, appropriate issues of naming, and naming conventions.

The end of Chapter 10 undertakes the client/server model. The author defines client/server correctly to be an inherently asymmetric model (no peer-to-peer client/server nonsense here) and insists, correctly, that the master is the client, providing a nice distinction between client/server and host-centric systems. Finally, he introduces Berkeley Sockets as the API (application programming interface) for the client/server model.

One wonders, at this point, whether the cleanliness and professionalism of the discussion really requires such a narrow treatment. It might well be so. But unhappily, no matter how "hypey" the environment may be, it is hard to feel absolutely comfortable with a client/server discussion that omits a discussion of remote procedure call (RPC) at this point, only to include it later on in a potpourri chapter on "distributed functionality." Yet this criticism is not crucial.

The book includes introductory but useful discussions of the standard distributed file systems—AT&T Remote File System**, Sun Network File System**, and the Sprite File System. Unhappily, once again, there is an insufficient discussion of the technology leading to the Distributed File System** whose view of the duties of a server are sufficiently different to merit discussion, and whose origin in the Andrew File System is sufficiently respectable not to offend a careful academic. Andrew is referred to very cursively in the text.

Chapter 12 is a general overview of various technologies in network environments. It is good in so far as it sensitizes the reader that there are topics here about which much more should be understood. The chapter contains the literature that will be useful in achieving this understanding.

The later chapters of the book, under the umbrella of "case studies," discuss structural and architectural features of those operating systems that have had the greatest intellectual impact: UNIX**, RC4000, THE, RIG (Rochester Intelligent Gateway), Accent, Mach, and my own favorite, Carnegie Mellon Hydra, which even today looks like a design document for NT**. In the network area, the author has correctly chosen LOCUS** and Eden, the V kernel as the key operating systems. I am personally sorry not to see something about Chorus** and Choices, but one cannot, can one, have everything. The author concludes with some discussion of queuing theory and simulation as methods for understanding operating system performance. I have never thought that this was true, since most queuing models do not reflect actual systems behavior too closely, but it is an

elegant academic pretense, and not for me to be rude.

Each chapter is supported by an extensive list of additional publications on the topic. It is clear that few of the seminal books have not been considered. For a reviewer who was present at much of the creation, and saw ideas evolve over time, it is pleasing to see how all this has been seamed together into a timeless intellectual structure. I like it enough to have used it in my own class. It is a quality book, a serious book, an intelligent book. Some may think it omits some important technologies, but not all would agree. I am pleased to have it available to me.

Harold Lorin
The Manticore Consultancy
and Senior Adjunct Professor,
Hofstra University
New York

**Trademark or registered trademark of Sun Microsystems, Inc., Open Software Foundation, X/Open Co. Ltd., Microsoft Corporation, Locus Computing Corporation, or Chorus, Inc.

Note—The books reviewed are those the Editor thinks might be of interest to our readers. The reviews express the opinions of the reviewers.