Software Maintenance: The Need for Standardization

NORMAN F. SCHNEIDEWIND, SENIOR MEMBER, IEEE

Invited Paper

Hardware and software maintenance are contrasted. The key difference between the two—the ease with which software can be changed—leads to the need for managing software change. Standardization of software maintenance is proposed as the method for managing software change. A model of software maintenance is advanced as the foundation for standardizing software maintenance.

I. INTRODUCTION

As an introduction to the subject of software maintenance, we provide some definitions followed by an explanation of the importance of the subject.

A. Definitions

Software Maintenance: Modification of a software product after delivery to correct faults, to improve performance or other attributes, or to adapt the product to a changed environment [1].

This definition is the conventional one and is useful if our interest in modification to software is limited to changes that are made after the software is delivered. However, it is a fact that changes are not confined to the post-delivery phase; they are made during all life cycle phases. In some cases, changes are made in significant numbers prior to delivery.

Maintainability: The ease with which a software can be maintained [1].

Change Management: The process of making changes to software and controlling their effects during the entire life of the software.

This definition recognizes the fact that modifications to software must be managed effectively during the entire life of the software. It is the definition used here.

According to various sources, software maintenance accounts for a significant amount of the total time and cost of running a data processing organization. For example, one study reports the following: about half of applications staff time spent on maintenance, over 40 percent of the effort in supporting an operational application system spent on user enhancements and extensions, and about half a man-year of effort allocated annually to maintain the average system [2]. In another report the same authors list the factors which cause the significant maintenance effort: system age, system size, relative amount of routine debugging, and the relative development experience of the maintainers [3]. System age drives the other factors: with increased system age, system size increases, leading to greater effort allocated to routine debugging, and with increased system age, the relative development experience of the maintainers declines due to organizational turnover and change. All of these factors tend to increase the time and cost of performing maintenance. Thus maintenance is an area that deserves a lot of attention. Improvements in maintenance practices should result in reduced costs and increased effectiveness of performing maintenance.

However there is a limit to reducing cost and increasing effectiveness through improved practices, because the maintainability of the software has largely been determined by the developer before it ever reaches the maintainer. The maintainer can only influence quality during the maintenance phase of the software life cycle. The quality of the software as designed is determined, in part, by whether the software development methodology assists the developer in producing maintainable software. Consequently, maintenance practices, which maintainers control, and development methodology, which developers control, are candidates for standardization.

The objective of standardization is to improve the maintainability of both existing and future software. Contrariwise, there are certain aspects of the "maintenance problem" that the above approach does not address. These are the following: 1) Much of the software that is maintained was developed without benefit of any methodology; consequently, methodology is not an issue in these cases; 2) Methodology is only an issue for future software; thus improvements in maintenance practices are only applicable to existing software; 3) An important determinant of the
maintainability of software is the knowledge and skill of the developer and maintainer; 4) There are other aspects of a development methodology, such as expressiveness, that are important when evaluating it for use in addition to its usefulness as an aid for producing maintainable software. These aspects are beyond the scope of the paper as are the areas of software engineering environments and tools, which can contribute significantly to the quality of both development and maintenance.

The paper consists of the following sections: Purposes; Objectives of Maintenance; Model of Maintenance; Standardization of Change Documentation; Software Communication Mechanisms and Maintenance; Standardization Through Examination of Development Methodologies; Example; Further Research; Summary.

II. Purposes

The purposes of the paper are the following: 1) Provide a brief introduction to software maintenance by describing its objectives, processes and tasks, contrasting it with hardware maintenance for the benefit of readers who may be more familiar with hardware maintenance, and 2) present the case for standardizing software maintenance practices and those aspects of software development methodology that affect the maintainability of the delivered software. Purpose 2 is derived from 1 on the basis that the kind of discipline and rigor that exists in hardware maintenance should be an objective of software maintenance.

Notice that we do not contend that identical methodologies or procedures should be used for software maintenance because there are differences in characteristics and complexities between the two; these differences are described in the paper. Rather, we propose that software maintenance should be supported by a model of maintenance and a minimum set of standardized practices, which would be augmented or tailored according to the needs of individual organizations or applications. The maintenance model includes characteristics of development methodologies because, as stated previously, these characteristics affect maintainability.

III. Objectives of Maintenance

The objective of maintenance is to make required changes in software in such a way that its value to users is increased. Required changes can result from either the need to correct errors or to increase the functionality of the software.

A. Maintenance Process

In the broad view of maintenance, it is not limited to making post-delivery changes[4]. Rather, it is a process that starts with user requirements and never ends [5]. Even the installation of and changes to a replacement system can be considered part of the maintenance process. Our approach to identifying the maintenance functions which should be standardized is to 1) adopt the view that maintenance is a process of change management and 2) identify tasks in maintenance that are concerned with making changes to software, including changes to documentation (e.g., specification, design, listing, test plan, etc.).

B. Maintenance Tasks

Using the concept of change management, the following maintenance tasks can be identified:

- Identify need for change
- Determine whether change should be made, based on benefit-cost analysis
- Evaluate the effects of change, including possible side effects
- Determine whether change can be made without creating an incompatibility with the rest of the software
- Make the change, if warranted, and only if it can be done in a standard way

C. Differences Between Hardware and Software Maintenance

Whereas failures in hardware are true failure events, which are caused by physical phenomena—wearout, burnout, malfunction, or stress—software “failures” are error discovery events, which are caused by errors made by humans. Software errors are caused by the following: inadequate or misunderstood specifications, incorrect program logic, misuse of programming language, and mistakes in clerical operations. These errors exist in software prior to its execution and are only discovered by virtue of an input forcing the software through an execution path that contains an error.

The ability to understand the nature of errors when maintaining software has been reported to be related to the quality of documentation [6]. Therefore the characteristics of documentation that affect maintenance should be a part of any plan to improve maintenance. Documentation for maintenance is discussed in Section VI.

1) Spare Parts: For software, there are no spare parts for replacing a module that has an error. The error must be fixed before the operation can continue. This is an inherent factor which makes software less reliable than hardware.

Repair times and down times can be very long. This situation demands easy maintainability. In particular, traceability must be achieved: the ability to easily trace through all relevant documents, organizations and personnel for the purpose of locating information which will assist the maintainer in correcting the error in such a way that the change will not damage another part of the program that is working (ripple effect).

2) Prototyping: Prototyping of software is similar to the hardware engineer’s test bench and development systems (e.g., in circuit emulation systems). With the software prototype we want to obtain a quick and inexpensive test of a development idea before committing a lot of time, personnel, and money to the production system. Another objective is to test design approaches in a simplified and controlled environment without the confounding interactions of a large system present. If the ideas will not work in the prototype, there is no hope of them working in the production system. One use of a prototype seldom mentioned is to test for flexibility of making changes to the software. For example, is the software constructed so that the effects of making changes are highly visible?

In many cases the prototype is treated as throwaway code. It is used for the purposes described and an improved version, based on the lessons learned, is coded as the next pro-
IV. Metrics for Maintenance

In order to manage software change it is desirable to measure the effects of change. This is accomplished with quality metrics. A quality metric is defined as follows: a quantitative measure of the degree to which software possesses a given attribute that affects its quality [1]. Ideally, there would be agreement on a set of application-independent, language-independent, software structure-independent metrics ("universal metrics"). Agreement does not exist in the software engineering community on a universal set. Lacking this agreement, metrics which are known to be related to the effectiveness and efficiency of the software development process are used during development to measure and improve the development process; these are called process metrics [7]. It is assumed that their use will result in maintainable software. However, process metrics, like traceability, have little to do with measuring whether the system achieves its quality requirements. For that we need product metrics like reliability, accuracy, response time, throughput, etc. The two types of metrics are related in the sense that high process metric values will contribute to high product metric values. Product metrics are beyond the scope of this paper.

The role of metrics in maintenance can be demonstrated by posing the following question: When a maintenance action is taken, how are the relevant metrics values affected?

1. What are the relevant metrics?
2. What were the original values?
3. What are the new values?
4. Examine incremental changes
   a. Are they in the right direction (e.g., reduced complexity)?
   b. Are they approximately the right values (e.g., within the bounds of experience with respect to the maintenance action)?

V. Model of Maintenance

To explain the dynamic interaction between development and maintenance as exemplified by the changes in metrics values as a result of development and maintenance actions, the model in Fig. 1 is provided. A model of the maintenance process is essential for standardization to be achieved. Different organizations may want to use different metrics, depending on the relevance of the metrics to their maintenance environments and projects.

This model may be understood and applied as follows:

1) Evaluate: Estimate the incremental change in metric value of a proposed maintenance action. If the software change is made, measure its effect after the change is made. To the extent feasible, quantify the effect of the change. The following questions are relevant when considering a change to software: Given the development methodology and a maintenance action, how will the metrics values be affected (magnitude and sign)? Will they change in a direction to indicate the software will be (or has been) improved? Or will the change indicate that the software will be (or has been) degraded?

This model would assist the maintenance organization to 1) determine whether a change should be made, 2) determine whether a change improved maintainability, if it was made, and 3) document the history of the project and the change so that this information can be used when making future change decisions.

Fig. 1. Model of the interaction between development, maintenance, and metrics.

VI. Standardization of Change Documentation

Because there is a great difference in applications, programming environments, etc., in various organizations, the maintenance standard should accommodate those differences and specify only a minimum set of requirements and procedures.

Standardization can be viewed as a process of posing questions prior to a maintenance action and having the maintainer answer them. The purpose of this is to ensure that the maintainer has thought about the consequences of proposed changes and is alerted to potential pitfalls. Maintenance decisions and actions should be recorded in a database for use in making future maintenance decisions. The entities which are subject to change are software components (an element of a software system such as a module or unit). For the sake of brevity, "software component" will hereafter be called "component."

A. Documenting the Effects of Change

It should be a standard procedure of maintenance to document a proposed change in the following format (or sim-
Table 1 Example Input-Output Change Relationship

<table>
<thead>
<tr>
<th>Input (Name)</th>
<th>Type</th>
<th>Format</th>
<th>Value</th>
<th>Range</th>
<th>Precision</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Type</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>Format</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Value</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>Precision</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Accuracy</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

ilar format) and, if the change is made, to fill in as much detail as possible about the change. The items to be considered in deciding on a change are more important than the specific format used to document the change. The Xs in the matrix in Table 1 indicate a relationship between an input item and an output item.

Change an Input

<table>
<thead>
<tr>
<th>Type</th>
<th>Format</th>
<th>Value</th>
<th>Range</th>
<th>Precision</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(How are outliers handled?)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Range</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Value</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Accuracy</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

B. Documentation Requirements

As a minimum the following should be standard documentation for supporting maintenance: requirements specification, design specification, program listing, test plan, and test results, as summarized below.

Phase                     Documentation
Requirements analysis      Requirements specification
Design                    Design specification
Coding                    Listing
All                       Test plan, Test results

VII. SOFTWARE COMMUNICATION MECHANISMS AND MAINTENANCE

Mechanisms which are available for communicating between components are an important aspect to maintenance because of the serious consequences of making an error in adding or changing a linkage. As opposed to other types of software changes, a change in a communication mechanism affects more than one component. This is particularly important for networks where a defective mechanism can adversely affect the operation of computers at remote sites.

A. Types of Communication Mechanisms

- Data linkages (for the transfer of data)
- Control linkages (for the transfer of control)
- Subroutine call
- Procedure call
- Message passing
- Remote procedure call (RPC)

- Transaction (e.g., update in a data base management system)

B. Characteristics of Communication Between Software Components

1) Explicit: There is an actual transfer or exchange of data or passing of parameters or an output from one component is the input to another component.

2) Implicit: Based on the position of the given component within a sequence of components (e.g., instructions in a program)

Before components are added, deleted or modified, it should be standard procedure to ascertain and document the effects of making the change on inter component communication. Furthermore, if the change is made, as much detail as possible should be documented about the change, as suggested by the questions below.

3) ADD a component:

- What other components will the given component communicate with once it is added?
- What are the communication linkages? (parameter passing, message exchange, RPC, etc.)?
- What existing communication linkages will be affected by the change?

4) DELETE a component:

- What communication linkage will be broken by the deletion?
- What are the new communication linkages that result from the deletion?

5) MODIFY a component:

- What is the existing communication linkage which involves this component?
- How will this communication linkage be modified by the change in the component?

VIII. STANDARDIZATION THROUGH EXAMINATION OF DEVELOPMENT METHODOLOGIES

There is evidence that the characteristics of development methodologies [8] and the characteristics of programming languages [9] can influence maintainability.

A. Characteristics of Development Methodology

When we maintain software we may not be cognizant of the development methodology which was used to produce the software, but it will affect our ability to maintain the software. The evaluation hinges on a single criterion: Does the methodology support the creation of software which is easy to change without inducing side effects (an unexpected and undesirable result of making a change)? This objective will be achieved if the methodology forces the designer to formally consider the consequences of making a change once the software has to be maintained. It follows that in order to capitalize on a methodology that supports maintenance, it is necessary to use that methodology to maintain the software. The following is a standard procedure for evaluating a methodology with respect to its capability to support maintenance.
Does the methodology assist to:

1) Prevent side effects when performing maintenance
2) Provide ability to make selective change (i.e., does not change or destroy another part of the software when making a change)
3) Reduce dependencies between inputs, processes and outputs (dependencies make it difficult to change the software without affecting something else which was working correctly prior to the change)
4) Determine whether change can be made without creating an incompatibility with the rest of the software
5) Support a rational change policy:
   - Make a change, if warranted, and only if it can be done in a standard way, a “standard way” being defined as being in conformance with the above procedure for assessing the impact of change.
   - Keep changes small.
   - Make changes in small, controlled increments.
   - If there is a big change to make, break the changes into manageable pieces.

IX. Example

A. Characteristics of Development Methodology

The process of identifying and evaluating development methodology principles that are conducive to maintenance is illustrated with real examples from personal computer network operating system software (IBM PC DOS V3.2 [10] and PC LAN Program V1.11[11] and the state diagram method of specifying software logic [12].

A batch (command file) for starting a user personal computer on a local area network (LAN) and assigning resources provided by a server is shown in Fig. 2: the corresponding state diagram is shown in Fig. 3. This batch file was modified to provide some additional network capabilities as shown in Fig. 2; the corresponding modification is shown in Fig. 3 with dotted boxes. The boxes represent states and the arrows represent state transitions. The numbers on the left side of the commands in the batch file correspond to the numbers on the state boxes on Fig. 3. The convention for labeling state transition arrows is: Event/Action. In some cases in Fig. 3 there is no event; in these cases “NE” is used to indicate this. The DOS and PC LAN Program handle transfers of control implicitly (e.g., a transfer of control occurs automatically from PC LAN Program to DOS under certain error conditions). There is no capability in the batch file language for describing error conditions explicitly, although they are shown in the state diagram to clarify the operation.

Asterisks in the batch file identify comments. Unfortunately, the comment concerning accessing the D drive was not changed with the modification. This comment is no longer applicable and caused confusion in trying to understand the program logic. With the modification, neither the D drive nor the directory program 1DIR are accessed at this point in the program. The comment should have been changed to refer to the E drive and the PROFILE program.

This affects the transitions from states 5 to 6 and 6 to 7. For the sake of brevity, the error events and actions associated with states 6’ and 7’ are not shown in Fig. 3; they are similar to those for states 6 and 7.

Neither a state diagram nor another type of methodology that would show the consequences of making a change was used in creating the batch program. The use of such a methodology would have helped to avoid this kind of error by:

- preventing side effects (erroneous comment)
- providing ability to make selective change (replace commands 6 and 7 with 6’ and 7’ correctly), and
- identifying existing communication linkages (communication between commands 6 and 7 and the D drive and its directories) and by identifying changed communication linkages (communication between commands 6’ and 7’ and the E drive and its directories).

It was mentioned previously that metrics are part of the maintenance model—they assist in evaluating the effects of change. When used over hundreds of components, the metrics can assume numerical values (e.g., for Completeness: ratio of completed components to total number of components in the system). For a single component, as in the example, a qualitative interpretation is appropriate. This is done in Table 2 for the example, using typical metrics. Although the modification has improved functionality, it has degraded maintainability.

B. Characteristics of Programming Language

Characteristics of the programming language can also significantly influence the ability to maintain [9]. Two brief examples from the DOS language [10] will be given:

- PATH command: If this command appears once and is repeated, the most recent occurrence of the command is the only one in effect. This means that any paths used to
CAN'T LOAD START FILE
FILE/ERROR MSG.
START DIRECTORIES/ERROR MSG.
FILE LOADED
BACK AT DOS
NE/LOAD TOKEN-RING PROGRAMS
CAN'T START NETWORK/ERROR MSG.
NE/START NETWORK
RESOURCES NOT AVAILABLE/ERROR MSG.
NETWORK STARTED
NE/REQUEST RESOURCES
RESOURCES ASSIGNED
NE/ACCESS D
DRIVE D
NE/LOAD DIR/ERROR MSG.
AT D PROMPT
6'
DRIVE NOT DEFINED/ERROR MSG.
AT NET MENU
7'
NE/ACCESS E
DRIVE E
NE/LOAD PROFILE
PROFILE PROGRAM LOADED

Fig. 3. State diagram of a token-ring LAN user computer start program.

establish directories in a previous occurrence are lost unless they are repeated in the new PATH command. In effect, this means that a new path must be a superset of the previous path, if all original directory information is to be retained. However, this could result in long path commands and, without writing complicated logic, commands are limited to a single line! Thus the maintenance principle of being able to make a selective change (i.e., one wants to just add or delete parts of the PATH command, not write a new one) cannot be achieved with this command.

- **IF command:** The IF command has the format: IF string1 == = string2 command. The requirement for the second "==" is unexpected. This nuance of the language has caused several errors in implementing network batch files. This seemingly minor item can cause havoc in maintenance because a frequent change to batch files occurs as the result of adding capabilities to the network that are conditioned on the availability of certain resources. The IF command is key to specifying these conditions.

X. FURTHER RESEARCH

Further research is necessary to examine development methodologies in more detail with respect to their influence on maintainability, for example the object oriented approach[9]. The objectives of this paper have been to make a start towards the goal of standardizing maintenance by proposing that a change management methodology is the key to standardization, and to begin a dialogue with the software engineering community concerning approaches for standardizing maintenance. The objective has not been to solve the whole problem, which is complex.

XI. SUMMARY

We have contrasted software maintenance with hardware maintenance. Although there are similarities, the major difference—the ease of changing software—causes unique software maintenance problems. We have proposed that maintenance can be improved through standardization. The elements of the proposed standardization process are the following:

- Metrics
- Model of maintenance
- Change documentation
- Software communication mechanisms
- Development methodology supportive of maintenance

<table>
<thead>
<tr>
<th>Table 2</th>
<th>Metrics Applied to Example Program</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Metric</strong></td>
<td><strong>Original Program</strong></td>
</tr>
<tr>
<td>Completeness: Are all required program parts present?</td>
<td>Yes</td>
</tr>
<tr>
<td>Consistency: Are the code and documentation uniform and free of contradiction?</td>
<td>Yes</td>
</tr>
<tr>
<td>Modularity: Is the structure cohesive and self-contained?</td>
<td>No</td>
</tr>
<tr>
<td>Traceability: Can the program parts be traced from one to another?</td>
<td>Yes</td>
</tr>
<tr>
<td>Verifiability: Can the correct operation and performance of the program be verified?</td>
<td>Yes</td>
</tr>
</tbody>
</table>

SCHNEIDEWIND: SOFTWARE MAINTENANCE STANDARDIZATION 623
An example was presented of the application of one development methodology—state diagrams—to illustrate how proposed and accomplished changes can be illuminated so that errors can be avoided and maintainability improved.

Finally, we stated that because the maintenance problem is so complex, more research must be done—particularly on the relationship between development methodologies and maintainability—before maintenance can be standardized.

REFERENCES


Norman F. Schneidewind (Senior Member, IEEE) received the B.S. degree in electrical engineering from the University of California at Berkeley, the M.S. degree in computer science from San Jose State University, San Jose, CA, and the M.S. and Ph.D. degrees in operations research from University of Southern California, Los Angeles. He is a Professor of Computer Science at the Naval Postgraduate School, Monterey, CA, where he teaches and does research in software engineering and computer networks. He is Technical Director of Computing in his department responsible for computer laboratories. Previously, he held various management and technical positions in system and software development in the computer industry.

Dr. Schneidewind is a member of ACM, Eta Kappa Nu, Tau Beta Pi, and Sigma Xi, and holds the Certificate in Data Processing. He is an editor of IEEE TRANSACTIONS ON SOFTWARE ENGINEERING and is Editor-in-Chief of the IEEE/ACM Fundamental Concepts in Software Engineering Series. He originated the first technical society conference on software maintenance with the Software Maintenance Workshop. He is Program Chairman for the Ninth International Conference on Distributed Computing Systems, and is Chairman of the IEEE Standard for Software Quality Metrics Working Group, which is developing the first standard in this area. He received the Computer Society Certificate of Appreciation for chairing the TC on Simulation and the Magazine Advisory Committee.
**Book Reviews**

The following reviews were selected from those recently published in various IEEE journals, magazines, and newsletters. They are reprinted here to make them conveniently available to the many readers who otherwise might not have ready access to them. Each review is followed by an identification of its original source.


This is an expanded and revised edition of the popular introductory communications book first published in 1971. This second edition accurately reflects the current trend towards digital communications and much recent work in this direction is included in this book. This revised edition has 17 chapters distributed over 759 pages and is meant for use in a two-semester introductory communications course.

The first two chapters provide the necessary mathematical background in spectral analysis, random variables and random processes. These chapters contained (in addition to the usual topics) some often neglected topics such as: expansions in orthogonal functions, Gram-Schmidt procedure and vector space representation of signals. While I felt that the topic of random processes was not discussed in sufficient detail compared to random variables, I was impressed by the clarity with which the concept of power spectral density was discussed.

The next two chapters discussed AM and FM. These chapters provide the usual topics such as DSB, SSB, VSB modulation techniques, FM signal spectrum, FM generation, FM demodulation, etc. I really enjoyed reading the five pages devoted to the stereophonic communication system example illustrating the SNR calculation for a total system.

Chapter 5 entitled "Analog-to-Digital Conversion" deals with sampling theorem, quantization, PAM, PCM, DPCM and ADM among other topics. Sampling theorem is presented in a clear manner, without unnecessary sophistication, for both low-pass and band-pass signals. As with many other chapters in this book, the authors provide the practical touch for this chapter by including a brief discussion about the T1 digital system. It also considers the concept of channel vocoders. However, I was surprised to see no discussion on Hilbert transforms when single sideband modulation techniques were introduced.

Chapter 6 discusses digital modulation techniques such as BPSK, DPSK, DQPSK, OOK, M-ary PSK, QAM, BPSK, M-ary FSK and MSK. It also includes partial response signaling schemes such as duobinary. This chapter includes some of the most recent techniques, making this book very attractive. This discussion of minimum shift keying (MSK) generation and reception are particularly appealing.

Chapter 7 considers the mathematical representation of noise. In addition to the usual topics of filtering effects, noise mixing, noise bandwidth and quadrature noise components, the authors provide the probability density functions of the noise components and their time derivatives. I also liked the use of example filters such as RC low-pass filters, integrators, etc., to illustrate the effects of linear filtering on noise.

Chapters 8 through 10 deal with the effects of noise in AM and FM systems and the important threshold effect in FM. Chapter 8 considers the effects of noise in various AM systems. Threshold effects in nonlinear demodulators such as square-law demodulator and envelope demodulator are clearly explained here. The next chapter provides a similar discussion for the effects of noise in FM. Preemphasis and deemphasis employed in commercial FM broadcasting is clearly explained in this chapter. Chapter 10 is devoted to the important concept of the threshold in FM. Particularly appealing are the sections on spike noise, spike suppression, phase-locked loops (PLL), the PLL output SNR and carrier recovery. These three chapters (8-10) are placed next to each other which helps in comparing the noise effects on AM and FM.

Chapters 11 and 12 discuss data transmission techniques and the effects of noise on these. After introducing the optimal filter theory, several signaling schemes such as PSK, FSK, non-coherent FSK, DPSK and QPSK are discussed. Signal space approach is used to calculate the probability of error for some of the digital signaling schemes. Chapter 12 introduces the effects of various noise (such as thermal noise, quantization noise, etc.) on pulse-code modulation and delta modulation. Once again, the authors provide an example application of the adaptive delta modulation in the Space Shuttle and thus reassure the readers that the various abstruse concepts being introduced are being applied.

Chapter 13 was the surprise in this book. Unlike many other introductory communications books, this one devotes 99 pages to the topics of information theory and coding. In my opinion, this is a welcome change from the usual practice of relegating these important topics to the last chapter. This chapter introduces the ideas of average information, entropy, information rate, source coding, channel capacity, bandwidth-SNR tradeoff, orthogonal signal transmission, block coding, many examples of algebraic coding, Reed-Solomon codes and convolutional coding. Powerful decoding algorithms such as the Viterbi algorithm are introduced in a simple manner so that their basic idea is easily understood. An indication of the up-to-date nature of this book is that almost ten pages are devoted to the topic of trellis-decoded modulation, which is only recently making its way into even more advanced communication theory books. This chapter is perhaps the strongest chapter in this book.

Chapter 14 attempts to describe the noise contributions from various physical sources such as resistors, reactive elements, two-ports, antennas, etc., in a communication system. Somehow, this chapter feels hollow (probably because it follows the very rigorous discussion on information theory and coding in Chapter 13) and the only redeeming feature was the simple satellite-to-earth communication system example illustrating the SNR calculation for a total system.

The last three chapters consider three applications, namely, telephone switching, computer communications systems and spread-spectrum systems. I was particularly pleased by the inclusion of telephone switching since most introductory communications books choose to ignore this topic and most undergraduates do not seem to receive formal teaching in this important area. The chapter on communications networks discusses various network types, provides examples such as Arpanet, Tymnet, etc. There is a brief, but accurate discussion of computer communication networks to get the uninformed interested in this important application. The last chapter provides a brief discussion of the advantages of spread-spectrum systems as well as some of the methods used to spread the signal spectrum. It probably would have improved this chapter had the authors provided some real examples of the application of spread-spectrum ideas.

This book has many things to recommend. The authors introduce various topics in an organized and clear fashion. They provide
discussion of much new material and provide many state of the art example applications. However, there were a few aspects of this book with which I was disappointed. While there are a total of 580 problems, a significant fraction of them are one-line problems asking for proofs of various equations in the text. Instead, I would have preferred to see a smaller number of problems that brought together a number of concepts. I was also surprised to see only 38 references (some of them repeated) for 17 chapters. In fact, three chapters (Digital Modulation Techniques, Noise Calculation in Communication Systems and Spread Spectrum Modulation) had no references at all. Even with the admirable job the authors have done in including various new topics, they do not have the space to include everything and I feel that they should have provided more references. In spite of these minor drawbacks, this is one of the best books in introductory communications and I recommend it highly for classroom use as well as outside the classroom.


Object-oriented programming has gained acceptance as a research topic and applied field in the last few years. Several annual conferences and symposia have been established on this subject—for example, the Object-Oriented Programming Systems, Languages and Applications (OOPSLA) Conference, started in 1986 by ACM. New object-oriented languages and commercial environments continue to appear and a few of them are now becoming available for personal computers.

Object-based systems have had an anomalous history. They have their roots in Smalltalk-80, the Xerox-supported environment. Smalltalk-80 was developed and promoted as an internal vehicle for research at Xerox PARC and was only disclosed as a complete environment after ten years of development effort. During this time, no attempt was made to stimulate external discussion on its basic concepts. Other object-oriented languages, such as Ada and Prolog, benefited from a more open research process.

As a consequence of their disparate origins, objects and their properties are not clearly defined. Their definitions must be inferred from the existing prototypes of object systems. As a result, there is some confusion over what objects and their properties are.

In Object-Oriented Programming: An Evolutionary Approach, Cox defines objects and their properties, and at the same time presents his own point of view concerning their possible applications. Cox argues that objects are the unescapable product of the evolution of software systems design toward the practice of reusing software components rather than designing software systems from scratch. In fact, Cox uses "object" as a synonym for software integrated circuit (IC). He argues that the production of software components is the long-term goal of any effort in software engineering. Cox envisions the realization of this goal as a collection of software integrated circuits.

The book is tailored to appeal to two rather distinct groups of readers: (1) those who have no previous experience with object systems and (2) those who are knowledgeable about object systems and have the necessary experience in system design to take advantage of the book's suggestions.

Each of these types of readers can take advantage of the book since it strikes a clever balance between chapters that are full of instructive examples and beautifully drawn pictures, and chapters that are rigorously theoretical and less descriptive.

The book can be divided into three parts: an assessment of the object paradigm and its motivation; a detailed description of the proposed object model and system—Objective-C; and examples of its use, a discussion of its limitations, and a few suggestions on how to overcome them.

In Chapter 1, Cox uses the example of design needs of a software system architect to argue for the reusability of software components. Even the contrived name of the software company, OSI, suggests in a subliminal way the necessity for standardization as a "dead end machine" solution. All the analogies throughout the book are effective and suggestive. In this chapter, the comparison of World War II defense techniques—the Maginot Line defense, the Swiss defense, and a hybrid one—reinforces Cox's conclusion that the best tactic for software development is a hybrid approach which aims for compatibility with existing software systems.

Chapter 2 categorizes several solutions to the problems of developing large software systems: the object-oriented approach is compared to the Unix pipeline approach and the traditional subroutine library approach. The issue of binding names to their physical counterparts is discussed in detail in later chapters of the book. C++ is a modification of C language. It introduces a construct for classes—a data-type apparatus. However, it does have some incompatibilities with C.

Chapter 4 and 5 describe Objective-C language, a language that is fully compatible with C. It has Objective-C as a simple front end between the usual C-preprocessor and the C-compiler. Chapter 4 introduces the key concepts of Objective-C language and presents examples of how objects and the message exchange between them are accomplished. Chapter 5 focuses on the idea of reusing information via inheritance and related concepts. The programming process is described exhaustively. The only weak link in this chapter is the discussion of multiple inheritance: its motivation is crudely outlined and only a quick glimpse of its implementation it given. I feel this is one of the book's few neglected topics.

Chapter 6 presents a simple example of Objective-C language—a step-by-step description with the development of the solution. The resulting performance is compared with the performance of a hand-coded solution at several levels. Finally, costs are carefully evaluated.

Chapters 7 and 8 describe the characteristics of some basic classes: object, array, and collection. Chapter 9 explores a more complete example, the design of a user interface based on icons for the Unix "make" commands. This chapter not only serves as a guide to the use of Objective-C but also provides an original explanation of the successive layers of a graphical interface. The example is completed by a cost performance comparison.

Chapter 10 is of particular interest to the programmer who is familiar with object-oriented systems. The constraints of the proposed object language are stated and the possibility of their release is proposed. This chapter deserves a careful reading. I regard it as one of the most important in the book—sufficiently important to be read as an independent paper. It deals with the most important issues to be solved in any object system support: memory allocation/deallocation strategies; the need for virtual memory interaction and the problem of interchange of objects between systems; and concurrency and object execution capacities, with the aim of proposing objects in distributed and coordination systems.

Cox presents a persuasive argument that object methodology is a sound solution to software engineering problems. In the process, he contributes in an essential way to the clarification of the object paradigm. His book provides a useful perspective on object programming for either the uninitiated or the initiated programmer. It could even be adopted as a textbook in a graduate course on software engineering environments.


Nowhere is the “information explosion” more pronounced than in the field of remote sensing. Spaceborne imaging instruments scan the whole of the earth’s surface at high resolution (tens of
Remote sensing and advanced computer analysis techniques are inevitable partners. Richards’ Remote Sensing Digital Image Analysis endeavors to assist the marriage. The book draws together “the range of digital image processing procedures and methodologies commonly used in this field into a single treatment . . . at a level meaningful to the non specialist digital image analyst, but in sufficient detail that algorithm limitations, alternative procedures, and current trends can be appreciated.”

The book grew out of a graduate course and manages to combine clear and orderly development of ideas with comprehensive scope. The presentation is succinct, but the orderly development and the exercises at the end of each chapter make the book suitable as a text. On the other hand, it is sufficiently comprehensive in scope and detailed in treatment to serve as a basic reference source.

Richards has allowed for the varied background of his readers. Chapter 1 identifies major sources of remote sensing image data. It provides supporting material for subsequent chapters and, at the same time, is an excellent compendium in its own right. Chapters 2 and 3 introduce the principles of error correction, registration, and interpretation of digital image data. Subsequent chapters discuss various forms of enhancement (both radiometric and geometric), data transformations (multispectral and Fourier), supervised and unsupervised classification, and feature reduction. In the final chapter, the preceding chapters are drawn together and, with the aid of a set of case studies, placed in the context of overall classification methodologies. Appendices provide much of the necessary mathematical background. A final appendix summarizes principles of real-time image processing in interactive image display systems, providing a useful, albeit brief, introduction for users of such systems.

There is much to praise in both the content and the presentation of the book. The publishers have certainly done the material justice with high quality glossy paper and many photographs, including some in color.

As is inevitable in any book attempting to cover as broad a scope as this one does, the content occasionally lacks depth and some issues are barely touched on. For example, the treatment of instrument defects does not take into account nonlinearity and long- range noise; and context-dependent classification is barely alluded to. However, references at the end of each chapter direct readers to further information.

Although largely a book about image analysis techniques, the author’s knowledge of application background is obvious. Application-specific issues are addressed and techniques which relate to specific applications (for example, the Kauth-Thomas Tasseled Cap Transformation) are given their due place. At the same time, Richards does not lose sight of computing issues such as computational cost of the algorithms.

In general, the level of mathematical complexity is appropriate to the likely readership. The one significant exception, which Richards himself acknowledges, is Chapter 7 (Fourier Transformation of Image Data) which, because of the mathematical complexities, some readers may wish to pass over.

On the negative side, there is a pronounced bias for agricultural and land-use applications. As a result, techniques from other fields, such as geology and bathymetry, receive less emphasis or are omitted altogether. One detected error of technical detail is the obscuring of distinction between active versus passive sensing and microwave versus nonmicrowave sensing.

Overall, however, this is a well-written, excellently presented book, with few weaknesses. It is suitable as a graduate text or as a reference book, and successfully bridges the gap between digital image analysis and application remote sensing.


This text provides a readable introduction to the basic theories of fuzzy mathematics and a comprehensive exposition of some of the main applications. The text is particularly recommended for specialists in applied mathematics and researchers in artificial intelligence. In AI implementation, topics such as common sense reasoning have eluded scientific treatment and formalization within the framework of classical logic and probability theory. Researchers in this field are encouraged by the ability of the theory of fuzzy sets to provide a formal basis for dealing with uncertainty in expert systems. The application of the theory of fuzzy sets to this problem may be the acid test that will determine the theory’s practical usefulness.

Fuzzy set theory has its origins in a now classic 1965 article by Lotfi Zadeh of UC Berkeley (L. Zadeh, “Fuzzy Sets,” Information and Control, vol. 8, 1965, pp. 338-353). This article attracted wide interest and resulted in the rapid development of this unorthodox and controversial field. The theory of fuzzy subsets is based upon multivalued logic and allows us to entertain conflicting propositions. A key result, the Goguen representation theorem (J. A. Goguen, “L-Fuzzy Sets,” I. Math. Analysis and Applications, vol. 18, 1967, pp. 145-174), states that any system satisfying certain axioms is equivalent to a system of fuzzy sets. The Goguen representation theorem is a precise mathematical result in the theory of categories. It allows us to conclude that inexact concepts can be represented by fuzzy sets.

The basic idea underlying fuzzy sets is that a subset $A$ of the universe of discourse $X$ (for example, the set $A$ of beautiful persons in a town $X$) is such that the transition between full membership and no membership is gradual rather than abrupt. The more an object $x$ belongs to $A$, the closer it is its grade of membership $\mu_A(x)$, where the grade of membership is assigned to those objects that fully belong to $A$ while $0$ is assigned to those objects that do not belong to $A at all. A fuzzy set is thus a “class” with a continuum of grades of membership, and is a generalization of abstract non- fuzzy (“crisp”) set theory where the grades of membership can only take the binary values $0$ or $1$. If $X = \{x\}$ denotes a space of objects, then a fuzzy set $A$ in $X$ is a set of ordered pairs $A = \{(x, \mu_A(x))\}$ where $\mu_A(x)$ is termed “the grade of membership of $x$ in $A$” and is a number in the interval $[0, 1]$. The grades of membership reflect an “ordering” of the objects in the universe and can be interpreted as the degree of compatibility of the predicate associated with $A$ and the object $x$.

The author observes that there are “three kinds of inexactness: generality, that a concept applies to a variety of situations; ambiguity, that it describes more than one distinguishable subconcept; and vagueness, that precise boundaries are not defined.” He explains that each of the three types of inexactness can be represented by a fuzzy set and provides examples of how this is done.

The text consists of seven chapters and an extensive bibliography of some 1000 references. The chapter topics are “Fuzzy Sets,” “Possibility Theory and Fuzzy Quantification,” “Fuzzy Functions,” “Fuzzy Events and Fuzzy Statistics,” “Fuzzy Relations,” “Fuzzy Logics,” and “Some Applications.”

The author of Fuzzy Mathematical Techniques with Applications conjectures that “Eventually . . . , the theory of fuzzy sets is likely to be recognized as a natural development in the evolution of scientific thinking; and the skepticism about its usefulness will be viewed, in retrospect, as a manifestation of the human attachment to tradition and resistance to innovation.” I concur with this conjecture and believe that his text does much to realize that end.


Batteries for Implantable Biomedical Devices—B. B. Owens, Ed. (New York, NY: Plenum Press, 1986, 358 pp., $35.00) Reviewed by Paul H. King, Department of Electrical and Biomedical Engineering, Vanderbilt University.

This text of 11 chapters by 19 authors is a unique treatment of the design of and the use of batteries in implantable biomedical devices. It stresses lithium-iodide battery technology (rightfully) with a fair section on nuclear batteries, and mention of other types. The editor states “The purpose of the present technical monograph is to summarize the technology of batteries that have been developed and applied to implantable medical devices.”

The text begins with a brief review of the types of implant instru-
ments, telemetry, stimulation, manipulation, and control) and follows with a discussion of the basics of several devices (pacemakers, defibrillators, bone regeneration, drug delivery systems, telemetry, etc.). In each case a brief mention is made of the physiology involved, followed by a discussion of the device parameters. The chapter concludes with a brief mention of the business aspects of implants ($2 billion in 1984) and a mention of future directions foreseen.

Chapter 2 is a unique chapter in that it is simply the edited text of a discussion with Dr. Samuel Ruben (mercury-zinc battery development) and a similar discussion with Dr. Wilson Greatbach (lithium/iodide battery development). The presentation of their biographical information relating to battery development and use is far more readable as an interview than it would have been in a simply historical review.

Chapters 3 through 8 cover various discussions relating to lithium batteries, notably battery design, evaluation, and performance modeling, and lithium halogen, lithium/liquid oxidant, and lithium/solid cathode systems. While some sections repeat others, this section of the text is an extremely good tutorial on the major considerations in the chemistry, design, construction, use, and testing of lithium based battery systems. It is in the thoroughness of subject coverage that one finds that the text is best used after one has had courses in at least statistics, material science, and physical chemistry; that is to say that the text is at least sophomore, perhaps junior level.

Two very short chapters cover the background, chemistry, and use of mercury and rechargeable batteries. While not currently relevant, these chapters, with the one following, round out the discussion of battery types.

The concluding chapter covers nuclear batteries, with detailed discussions of isotope selection, battery design, and thermal, electrical, radiation, and safety considerations. The depth of coverage, especially on thermal design, would allow parts of this chapter to be used as case studies in a design course.

In general, this is an excellent text, suitable for junior or senior use. Portions of the text could easily be extracted for use in a senior level design course, in a senior/graduate level advanced instrumentation course, or in heat transfer courses. The text is in general very well referenced, additional reading is likely easily available. As mentioned in the foreword, the text is "very helpful for power source researchers, medical students and physicians, and evaluators of present and future generations of implantable power sources and devices." It contains a wealth of information for all of the above.


According to the authors of Design of VLSI Circuits Based on VENUS, designing VLSI chips is rapidly becoming a necessary skill for every digital designer. Their concern is how to make it simple for nonspecialist engineers to produce working chips. Their solution is VENUS, a Siemens CAD system, which supports a wide variety of chip design methodologies. The book has good coverage of the issues and details needed to design ICs. Many design alternatives are covered well, including gate array, standard cell, and macrocell methodologies. While the tools and examples are all Siemens-based, the concepts are valid in most design environments. The descriptions of device physics and manual IC design are weak. In addition, the quality of the writing is sometimes poor; the content is sometimes hidden. Some of this may be due to its translation from German, though this is generally good. The writing quality varies from chapter to chapter, perhaps depending on additional coauthors.

Chapter 1 is an overview of the design process, defining all the specifications, verification, and testing steps, which must be included to design working chips. This book recognizes the great importance of testing and makes this clear early on, along with the importance of verification of the functional equivalence between different levels of a design. With all this detail included, the chip design process looks very complicated indeed. Chapter 1 then introduces standard design, e.g., gate arrays and standard cells. It shows quite clearly how standard design automates many implementation and verification steps in the design process and guarantees first pass correct chips in exchange for layout constraints. While the content of Chap. 1 is excellent, the writing is unsatisfying. Many simple concepts are unclear on the first reading.

Chapter 2 discusses semiconductor technology and transistor circuits, both bipolar and MOS. The treatment of device behavior is so brief as to be essentially useless. Fabrication is acceptable. Most circuits are presented without much explanation, although MOS memory devices are explained quite well. The chapter is generally weak.

Chapter 3 discusses the layout methods for standard design as well as manual design. At this point, we start learning about the capabilities of VENUS, with many figures from VENUS-based designs. VENUS supports automatic layout for three methodologies: gate arrays, standard cells, and macrocells, which do not need to be in rows or columns. Cells can be library cells, parameterized, or manually designed. The treatment of manual design and geometric design rules for layout is very brief and unsatisfying—you have to know it already to understand it. The chapter has good coverage of the design trade-offs of the various methods. The capabilities of VENUS are quite impressive as described here. Some of the layout figures in the chapter are very hard to follow due to lack of labels.

Chapter 4 is devoted to testing, including design for testability. The authors do a fine job of covering testing issues including fault models, parametric testing, logic testing, scan paths, BILBO, electron beam probe, etc. Test engineering is at least half the effort in designing a chip; this chapter shows why.

Chapter 5 gives an overview of the kinds of cells one can expect in a standard design system, in this case, VENUS. Complete lists of cells are given, along with data sheets for a few of them. In addition, the chapter describes the process for adding a new cell to the library and the many verification, testing, and fabrication steps, which must be completed for the cell to be of sufficient quality for others to use. Both parts are very instructive.

Chapter 6 follows the standard design procedure "for real," taking a design through each of the steps using the VENUS design system. It starts with high-level decisions, such as "which design methodology?" and "which design methodology?" It then proceeds through the specification, verification, and layout steps, showing actual listings and plots. The result is a correct chip design, including the test patterns needed to test the fabricated chips. The chapter is very detailed, but quite instructive.

Chapter 7 looks at the future. It touches on several topics, including integrating higher tools such as logic synthesis and system-level tools with the chip-oriented tools in VENUS. Expert systems and design data bases are mentioned briefly.

Design of VLSI Circuits Based on VENUS is a good and useful treatment of standard design, especially the design steps and tools. It is not a good book for Mead-Conway style custom VLSI; it has very brief treatment of devices and manual layout. However, this is precisely the point of standard design: to get working chips and avoid "thinking in silicon."


Book Alert

The following descriptions of recent books were prepared by the staff of the Engineering Societies Library, 345 East 47th Street, New York, NY 10017-2394. These books are available in the Library for loan or reference use. The prospective buyer should contact the listed publishers or a local technical book store.

This book balances theory and applications applicable to monolithic microwave integrated circuits and hybrid circuits. The early chapters are devoted to textbook basics. They also include a comprehensive treatment of GaAs MESFET parameter and noise measurement. Later chapters treat power GaAs MESFET amplifiers, oscillators, GaAs digital integrated circuits, microwave monolithic integrated circuits, and an aspect of GaAs MESFET reliability—surface-induced degradation mechanisms.


This book bridges the gap between classical and modern methods of signal processing. The author emphasizes random signal types, including digital signal processing methods using the Fast Fourier Transform (FFT), through modern parametric and adaptive signal processing techniques, up to the sophisticated model-based processing methods. The book presents digital signal processing using the basic premise that the signal processor is provided with a set of data records and asked to extract the required signal information. Also included are computer simulated examples using SIG* and SSAPACK* in order to give readers a feel for the "practice" of signal processing.


Chapters include: Principles of operation, Ground antenna systems, Interrogators, Sliding window plot extraction, Monopulse plot extraction, Transponders, Multipath and interference, System design, Performance measurement and monitoring, Mode S, and Traffic advisory and collision avoidance system.


Technological developments in composite materials, nondestructive testing, and signal processing as well as biomedical applications, have stimulated wide-ranging engineering investigations of heterogeneous anisotropic media and surface waves of different types. Wave propagation in solids is now of considerable importance in a variety of applications.

The book presents many of the key results in this field and interprets them from a united engineering viewpoint. The conceptual importance and relevance for applications were the prevailing criteria in selecting the topics. Included are body and surface waves in elastic, viscoelastic, and piezoelectric media and waveguides, with emphasis on the effects of inhomogeneity and anisotropy. The book focuses on physically meaningful theoretical models, a broad spectrum of which is covered. Both exact and approximate approaches are discussed.


This book presents a thorough introduction to structural vibrations (structure-borne sound), with emphasis on those at audio frequencies, and the attendant radiation of sound. It presents in-depth discussions of fundamental principles and basic problems, in order to enable the reader to understand and solve his own problems.

It includes chapters dealing with: measurement and generation of vibrations and sound, the various types of structural wave motions, structural damping and its effects, impedances and vibration responses of the important types of structures, attenuation of vibrations, and sound radiation from structures.


This book treats theoretical and experimental aspects of the physics of quasi-one-dimensional conductors, such as metal-chain compounds, organic conductors, and conducting polymers. The properties and physical concepts associated with electronically one-dimensional systems are clearly and fully described. New topics such as high-transition-temperature organic superconductivity are also included.


The field of electron and ion optics is based on the analogy between geometrical light optics and the motion of charged particles in electromagnetic fields. The spectacular development of the electron microscope clearly shows the possibilities of image formation by charged particles of wavelength much shorter than that of visible light.

This book is intended to be a self-contained, systematic and up-to-date introduction to the field. The reader can find the derivation of the most essential relationships, the understanding of which is absolutely necessary to do any meaningful work with particle beams and optics. Modern computer methods are especially emphasized. The book can serve as a textbook for engineers, scientists, and graduate students who wish to understand the basic principles of electron and ion optics and apply them to the design and operation of beam-type devices and instruments.

The treatment is restricted to the presentation of the basics and the most recent results of research, including the author's own. Throughout the book the relevance of the presented material to practical applications is emphasized, but no attempt has been made to review the applications themselves.


This book introduces multiplicative complexity theory and applies it to computation of discrete convolutions and the discrete Fourier transform. A mathematical framework is established that enables the multiplicative complexity of many common bilinear systems to be determined. Semilinear systems are introduced as a means of determining the multiplicative complexity of the discrete Fourier transform. Complexity results are derived for other discrete transforms such as the cosine and Hartley transforms.

Complete proofs of all the major results are included—the book is essentially self-contained. Many examples are provided to demonstrate the application of the theory.


This book provides fundamental concepts for analyzing and diagnosing electronic circuits and establishes recent, efficient analog techniques in areas that cannot be handled by standard and established models.

Offering a practical treatment of the theories, algorithms, and techniques of circuit analysis and diagnosis, this reference focuses on computational approaches.

Chapters cover computationally efficient modelling of transistors, LU decomposition algorithms for parallel and vector computation, piecewise linear analysis, decomposition techniques in circuit simulation, waveform relaxation algorithms, computer methods for analyzing switched-capacitor networks, techniques for finding the periodic steady-state response of circuits, symbolic network analysis, diagnosability of analog circuits, symbolic network analysis, diagnosability of analog circuits, fault location by nodal equations, table approach to fault diagnosis, computa-
tional approaches to fault dictionary decomposition approaches to fault location, and optimization techniques for modeling, diagnosis, and tuning.

Fifth Generation Wafer Architecture—Malcolm J. Shute. (Englewood Cliffs, NJ: Prentice-Hall, 1988, 316 pp., bound, price not given, ISBN 0-13-314238-8.) This volume explores the possible realization of fifth generation computer architectures through wafer scale integration. It looks at architectural considerations, and designs for fifth generation, non-von Neumann computer architectures. It also considers fault tolerance, failure tolerance, reconfiguration, and wafer scale integration. It describes a number of the techniques and problems of building the largest single chip computers possible, namely those which occupy an entire wafer of semiconductor. It necessarily spans the many subject areas which influence computer design, in particular those of computer science, suggesting that which is desirable in a computer, and those of microelectronics, indicating that which is possible in a computer.

Intuitive Operational Amplifiers; From Basics to Useful Applications. (McGraw-Hill Series in Intuitive IC Electronic,—Thomas M. Fredrikson. (New York, NY: McGraw-Hill, 1988, 322 pp., revised ed., bound, $29.95, ISBN 0-07-021966-4.) Containing real-world problems and their solutions, this practical book provides a solid understanding of the op amp, demonstrates just what linear circuits and systems are all about, and presents the design equations for many useful application circuits (along with a handy collection of application circuits to use as a reference). Covering such important topics as frequency stability and feedback control theory, it spans the entire subject from background information on the op amp to new developments and future directions.

Analytical Techniques for Thin Films; Treatise on Materials Science and Technology, Volume 27—K. N. Tu and R. Rosenberg, Eds. (Orlando, FL: Academic Press, 1988, 493 pp., bound, $89.95, ISBN 0-12-607740-1.) This volume is intended to introduce to the readers a selected set of analytical techniques developed for thin films and interfaces. These techniques are all based on scattering and excitation phenomena and theories; they are different from those used for bulk materials, because it is the near-surface region that is emphasized and sensitivity is an issue. Also, they are not quite the same as those used exclusively for surface analysis since it is the interior of a film that is of concern, and the lateral resolution is important. In-depth resolution; in particular, the analysis of submicron-size thin films is challenging.

Electromagnetic Inverse Profiling: Theory and Numerical Implementation—A. G. Tijhuis. (The Netherlands: VNU Science Press/VSP, 1987, 465 pp., bound, $72.00 (approx.), ISBN 90-6764-693-X.) This monograph overviews the author's research on the direct scattering of electromagnetic waves by one- and two-dimensional objects and its use for one-dimensional inverse profiling. Having investigated these problems for a number of years, the author has obtained a series of results both in method and in application to specific problems. Several techniques are presented for the solution of transient electromagnetic direct-scattering problems. These problems are solved indirectly via a Fourier or Laplace transformation to the real- or complex-frequency domain, as well as directly in the time domain. Both the analytical and the computational aspects of each solution technique are discussed. For each technique, representative numerical results of its application to at least one example problem are presented and discussed. Special attention is devoted to the physical interpretation of the theoretical and numerical results. A more detailed survey of the direct- and inverse-scattering problems that have been resolved, and of the algorithms applied to arrive at the solutions, is given in Chapter 1.

The Industrial Laser Annual Handbook (SPIE Volume 919)—David Beltorte and Morris Levitt, Eds. (Tulsa, OK and Bellingham, WA: PennWell Publishing Co. and SPIE-International Society for Optical Engineering, 1987, 259 pp., bound, $99.00 members, $125.00 non-members, ISBN 0-87814-320-3.) The most important change in this year's edition of the Handbook is the increase in the size of Section I, Laser material processing data and guidelines. Subjects include, among others, laser cutting data tables and curves; laser drilling, heat treating, surface hardening, and cladding; data tables and curves; and laser versus non-laser process comparison. Section II contains twelve papers concerning current aspects of laser technology, Section III is a company and product directory, and Section IV is concerned with related products and services.


Full coverage is devoted to the structural, mechanical, electronic, electrical, and software design elements involved. Also investigated in detail are such topics as manipulator simulation, actuators, transmissions and joints, and grippers. An extended chapter deals with computations and derivations (robot mathematics), and the final chapter suggests research thrusts for the robots of the future.

<table>
<thead>
<tr>
<th>Title</th>
<th>Issue</th>
<th>Guest Editor(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Radar Cross Sections of Complex Objects</td>
<td>May '89</td>
<td>W. Ross Stone 1446 Vista Claridad La Jolla, CA 92037</td>
</tr>
<tr>
<td>Electric Utility Systems Planning Issues and Methods (Special Section)</td>
<td>June '89</td>
<td>Robert J. Ringlee Power Technologies, Inc. P.O. Box 1058 Schenectady, NY 12301</td>
</tr>
<tr>
<td>Large Problems and the Requirements They Place on Supercomputers (Special Section)</td>
<td>Mid '89</td>
<td>Paul B. Schneck Supercomputing Research Center Institute for Defense Analysis 17100 Science Drive Bowie, MD 20715-4300</td>
</tr>
<tr>
<td>Optical Satellite Communication (Special Section)</td>
<td>Mid '89</td>
<td>Vincent Chan C-270 MIT Lincoln Laboratory 224 Wood Street Lexington, MA 02173</td>
</tr>
<tr>
<td>High-Speed Communication Networks (Special Section)</td>
<td>Mid '89</td>
<td>Imrich Chlamtac Dept. Electrical and Computer Engineering University of Massachusetts Amherst, MA 01003</td>
</tr>
<tr>
<td>Collaborative Semiconductor Research</td>
<td>Late '89</td>
<td>William C. Holton Microstructure Sciences Semiconductor Research Corporation P.O. Box 12053 Research Triangle Park, NC 27709</td>
</tr>
<tr>
<td>Superconductivity</td>
<td>Late '89</td>
<td>Theodore Van Duzer Dept. Electrical Engineering and Computer Sciences University of California Berkeley, CA 94720</td>
</tr>
<tr>
<td>Optical Signal Processing Devices (Special Section)</td>
<td>Late '89</td>
<td>Joseph L. Horner Solid State Science Directorate Rome Air Development Center Hanscom AFB, MA 01731-5000</td>
</tr>
<tr>
<td>Future of Computer-Aided Design</td>
<td>Late '89</td>
<td>Stephen W. Director Dept. Electrical and Computer Engineering Carnegie Mellon University Pittsburgh, PA 15213</td>
</tr>
<tr>
<td>Fast-Packet-Switching Networks (Special Section)</td>
<td>Late '89</td>
<td>Gottfried W. Luderer Room 6e-342 AT&amp;T Bell Laboratories Naperville Wheaton Road Naperville, IL 60566</td>
</tr>
<tr>
<td>Title</td>
<td>Issue</td>
<td>Guest Editor(s)</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>-------</td>
<td>---------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| Supercomputer Technology                  | Early '90 | Tse-yun Feng  
              Dept. Electrical Engineering  
              Pennsylvania State University  
              EE East Bldg.  
              University Park, PA 16802  
              Ali R. Hurson  
              Dept. Electrical Engineering  
              Pennsylvania State University  
              EE East Bldg.  
              University Park, PA 16802 |
| Air Traffic Control                       | Early '90 | Martin T. Pozesky  
              ADL  
              NAS Programs  
              Federal Aviation Administration  
              800 Independence Ave., S. W.  
              Washington, DC 20591  
              Robert M. White  
              Research and Engineering  
              Control Data Corporation  
              8100 34th Ave. S.  
              Minneapolis, MN 55440  
              N. K. Bose  
              Dept. Electrical Engineering  
              Pennsylvania State University  
              121 EE East  
              University Park, PA 16802  
              Clifford Lau  
              Code 1114SE  
              Office of Naval Research  
              800 N. Quincy Street  
              Arlington, VA 22217-5000  
              William Streifer  
              Spectra Diode Laboratories  
              80 Rose Orchard Way  
              San Jose, CA 95134-1995  
              William W. Wu  
              INTELSAT  
              3400 International Drive  
              Washington, DC 20008  
              Geoffrey Hyde  
              Room 0209  
              COMSAT Laboratories  
              22300 Comsat Drive  
              Clarksburg, MD 20871  
              Sajjad H. Durrani  
              Code TP  
              NASA Headquarters  
              Washington, DC 20546  
              R. Fabian Pease  
              Stanford Electronics Laboratories  
              Stanford University  
              Stanford, CA 94305-4055 |