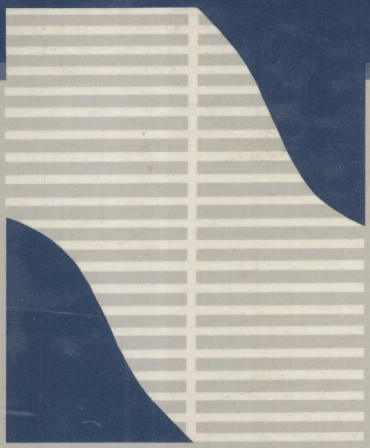
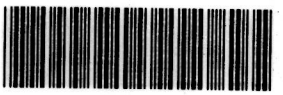


Information Systems Development

Principles of Computer-aided Software Engineering

ALBERT F. CASE, JR.

INFORMATION SYSTEMS DEVELOPMENT: PRINCIPLES OF COMPUTER-AIDED SOFTWARE ENGINEERING



ALBERT F. CASE, JR.

Nastec Corporation

PRENTICE-HALL

Englewood Cliffs, New Jersey 07632

Library of Congress Cataloging-in-Publication Data

Case, Albert F.
Information systems development.

Bibliography: p.
Includes index.
1. System design. 2. Computer software—
Development. I. Title.
QA76.9.S88C39 1986 005.1 85-25567
ISBN 0-13-464520-0

Editorial/production supervision and
interior design: *Carol L. Atkins*
Cover design: *Lundgren Graphics, Ltd.*
Manufacturing buyer: *Gordon Osbourne*

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

ISBN 0-13-464520-0 025

Prentice-Hall International (UK) Limited, *London*
Prentice-Hall of Australia Pty. Limited, *Sydney*
Prentice-Hall Canada Inc., *Toronto*
Prentice-Hall Hispanoamericana, S.A., *Mexico*
Prentice-Hall of India Private Limited, *New Delhi*
Prentice-Hall of Japan, Inc., *Tokyo*
Prentice-Hall of Southeast Asia Pte. Ltd., *Singapore*
Editora Prentice-Hall do Brasil, Ltda., *Rio de Janeiro*
Whitehall Books Limited, *Wellington, New Zealand*

To Deb, Kim, Chris, Mom, and Dad

Preface

OBJECTIVES OF THIS BOOK

One can rarely pick up a trade journal or systems management text without finding an article on the subject of managing the systems development process, ensuring the satisfaction of systems users, or keeping project budgets under control. Now the subject of the "software challenge," the challenge of producing computer software systems of high quality and low cost in a timely fashion, is becoming a topic of interest to the popular and business press.

There are, today, tools, techniques, methodologies, and strategies available to software development management to meet the software challenge. The intention of *Information Systems Development: Principles of Computer-Aided Software Engineering* is to outline an integrated strategy and methodology for improving information systems development productivity. This strategy calls for:

1. A structured, well-defined planning process that drives systems development
2. An engineering approach to developing software and managing projects
3. Specifications for a computer-aided software engineering (CASE) system, an automated tool kit to support managers and software developers

Although the strategy and procedures outlined in this book are oriented to the process of developing business-oriented information systems, the techniques and processes outlined support engineered products/embedded systems and

large-scale special-purpose systems (such as those of the Department of Defense and the aerospace industry) as well.

AUDIENCE

Information Systems Development presents a comprehensive management and technical overview of the process of developing information systems. Its management and planning procedures and implementation recommendations would be of great value to a director or vice president of management information systems (MIS). Since the specific technical processes for designing software are covered, this book would also be an excellent guide for managers, analysts, or consultants charged with selecting and implementing development techniques.

Project and systems development managers will find the discussions of the systems development life cycle and project management techniques most helpful in managing their projects. Some of the techniques can be implemented immediately with visible, short-term results.

Since this book examines and defines the software development process from planning to implementation, it could be used as either a primary or supplementary textbook for advanced courses in MIS or project management.

WHY THIS BOOK WAS WRITTEN: THE SOFTWARE CHALLENGE

The "software challenge" refers to the conflict between the desire on the part of data processing users to have new, high-quality* information systems, and the ability of the systems development organization or department to deliver them in a timely, cost-effective manner.

The software challenge is really two problems. The first problem is immediate. It is relatively common knowledge that development projects rarely come in on target. Systems installed typically have extensive "bug-fixing" maintenance periods after installation, frequently disrupting the normal operations of the enterprise. Software development is expensive and unreliable.

The second problem is more insidious and less visible. There are limited software development resources. A recent U.S. Department of Defense (DoD) study entitled "Software Technology for Adaptable Reliable Systems" indicated that demand for computer software is increasing at a 12% compound rate annually, while the supply of expert computer software professionals is growing

*Throughout this book, the term "high-quality" will represent both absence of defect *and* adherence to the user's needs.

at only a 4% compound rate. Present productivity tools can be expected to increase productivity only 4%. This study concludes, therefore, that by 1990, there could be a shortage of qualified software development personnel of as many as 1 millon persons. This means that while demand for software is growing, the resources that can provide it are not. This translates directly into higher labor costs, hence higher software costs. Even if the developers are available, they may not be affordable.

FUELING THE GROWTH IN SOFTWARE DEMAND

What is fueling this tremendous growth in demand for computer software?

First, tremendous advances in hardware engineering and technology, combined with the plummeting cost of computers (mainframe, mini, micro, and home), are fueling demand for application software to make these computers operate.

Second, during the 1970s, quality of product and productivity of the American work force threatened the very survival of the U.S. economic system. Executive management recognized that the computerization of America was the key to regaining American industrial and economic supremacy. The latest software applications to support this computerization are significantly more complex than the applications developed in the 1960s and 1970s.

Third, the world of consumer products has discovered the microchip. Everything from automobiles, dishwashers, and microwave ovens to industrial photocopiers and robotics systems are now computer controlled. This new world of embedded systems, or engineered computer products, represents a whole new vista of software applications.

Fourth, a growing awareness of the "Information Age," spurred by such works as Alvin Toffler's *Third Wave* and John Naisbitt's *Megatrends,* has made mainstream America more computer literate. Hence they are more demanding of organizations that provide computer software.

WHAT DOES THIS MEAN TO SOFTWARE DEVELOPMENT MANAGEMENT AND PROFESSIONALS?

It means primarily that we are being challenged to produce an increasing number of more complex information systems. And we are being challenged to do so within the constraints of limited resources, compounded by the fact that each new system built requires an increase in maintenance or support resources, thus detracting from the resources available to build new systems.

HOW CAN WE MEET THIS CHALLENGE?

The engineering and manufacturing industries were faced with a similar problem—to produce more, higher-quality products within limiting resource constraints. Their solution was to provide a systematic, engineering discipline to the development effort, controlled by advanced management techniques and supported by new technologies such as computer-aided design and manufacturing (CAD/CAM) and computer-aided engineering (CAE). Engineers and manufacturers became *smarter* about how they did their work. Systems development managers and professionals must now follow suit. Software developers must be able to keep pace, in productivity and quality, with hardware developers.

There has been much work done in this area already. The purveyors of methodologies and design techniques, the vendors of programming and generation tools, and the suppliers of project control systems all offer partial solutions. To be effective, however, these solutions must be *INTEGRATED.* Some effort toward integration has already been made in the government arena. For example, the DoD STARS (Software Technology for Adaptable Reliable Systems) program is endeavoring to specify a complete environment for developing software in a controlled, managed environment. What is needed is an identification of these advances and a translation from the research environment to the real-world business of information systems development.

Information Systems Development is a comprehensive, practical manual for evaluating software engineering technology and applying it to the development of information systems. It examines the planning, management, control, and development processes involved in systems development, and provides a blueprint for a software engineering management information system to integrate these processes.

USERS' GUIDE FOR INFORMATION SYSTEMS DEVELOPMENT

This book, itself a system, is a set of specifications for a system to build information systems. A system to build systems is relatively complex and contains recursion and iteration of various processes. This is not bad if you are describing the system to a computer in a programming language that contains *DO-WHILE, PERFORM,* and/or *CALL* constructs. English, on the other hand, is not so structured. Because people cannot read in a parallel processing mode, each of the four parts contains enough information to stand alone. This helps to eliminate the need to constantly cross-reference to other parts of the text.

Although every author hopes and dreams that readers will hang on his or her every word, reality indicates that this is not true. Since each part is capable of being read independently, the reader can random-access the part of the book that

addresses a specific informational need. This is a by-product of the high cohesion/low coupling of the individual parts of the book.

Since every good system needs a users' guide, such a guide for this text is included below. For each part, the general content and major audience are included.

PART I: Introduction.

The introduction provides the student, professional, and manager with a historical perspective of the challenges (both immediate and long term) facing the software development industry, and defines software engineering. This section is important to an understanding of the other sections.

PART II: Systems Planning Process.

A prime contention of the management aspects of the software engineering approach to systems development is that management is a function of planning. This is "must" reading for managers, but could be skipped by project managers and software development professionals if Chapter 6 were read.

PART III: Software Engineering Transformation Process.

This section deals primarily with the technical aspects of managing systems projects. Life cycles and development techniques are described. This is an important section for the professional and the project manager. Senior management would benefit from reading Chapter 11.

PART IV: Computer-Aided Software Engineering.

This part looks at the management information system (SE/MIS) required to support the software engineering transformation process. The emerging CASE technology as a mechanism to automate the SE/MIS, where it is headed, and what it means to software development are discussed. The premise is that CASE is the mechanism which will enable the software engineering approach to be implemented in a consistent, cost-effective manner. The impact of artificial intelligence on CASE is discussed. This section is highly recommended for all readers and is required reading for senior and middle management.

A NOTE ON CHARTS AND DIAGRAMS

There are numerous diagrams in this book depicting the relationship between the various entities and processes involved in developing information systems.

The representation used in many of these diagrams is process-flow notation. In Part IV, an analysis of the various system development techniques is portrayed in which process-flow development techniques are discussed. It is important to note that process-flow notation is an excellent method of modeling or representing a known environment and is widely recognized and understood by information systems professionals. However, as will be evident from reading Part IV, it is not necessarily the best or the only method for discovery and definition when applied to information systems analysis and design. Process-flow notation was adopted for convenience and should not be construed as an endorsement of process-flow design techniques.

ACKNOWLEDGMENTS

This book is the culmination of more than three years of research, tinkering, and thinking about improving the way information systems are developed. Like any other major project, it cannot be done by a single person. Although my name appears on the cover, numerous people have contributed to the success of this endeavor.

First, I would like to thank the following executives of Nastec Corporation: Ken Hill, Jim McGuire, Dick Ramsdell, Al Connor, Tom Long, and John Manley (who is now the director of the Software Engineering Institute at Carnegie-Mellon University). They not only tolerated my considerable preoccupation with this book, but contributed to the effort with ideas, encouragement, and resources. (The entire original manuscript, both graphics and text, was prepared at Nastec on a CASE 2000 Workstation.) Furthermore, they gave me the opportunity to validate many of the concepts in this book. This would not have been possible under other circumstances.

While most of my professional colleagues contributed in some fashion to this book, some in particular, should be mentioned. Jim Blake applied many of the principles in this book to his projects, and tempered some of my more avant-garde ideas with practical experience. Vaughn Frick is a walking encyclopaedia of information on development techniques and provided considerable insight. Byron Burke assisted me in the area of automating life-cycle methodologies.

Much of the work for this book was done while I was at Ryder System—Automotive Carrier Division. Greg Vogel (then Group Controller for the M&G Convoy subsidiaries) welcomed the use of the concepts in this book for the systems development work in his area of responsibility. David Caswell, also from Ryder, strongly encouraged the project.

There are two other associates who deserve credit for prodding me to completion on this project. Dr. Eric Streiff, Dean of Continuing Education, State University of New York at Buffalo, was enough of a risk-taker to let me develop and teach a new course on data processing project management in the

Millard Fillmore College Division. The class notes for that course ultimately evolved into this book. I would also like to thank Karl Karlstrom, my editor at Prentice-Hall. Karl tolerated three missed deadlines, yet continued to support the project and prodded me to completion.

Last, but certainly not least, I would like to thank my wife, Deborah, my daughter Kimberly Marie and my son, A. F. Christopher Case III who suffered through late nights, missed dinners and absent weekends (while I tried to make up for three missed deadlines). Not once did they waver in their support and encouragement for me to complete this project or complain about my share of the chores which went undone.

In addition to those mentioned there were many others who contributed to this effort: former students, business associates, and friends who gave me ideas, tried out the concepts, and provided valuable critiques of this approach to building systems. I thank them all. The successful and beneficial aspects of this book are the result of all of these contributions; however, any shortcomings, errors, or omissions are my own responsibility—those I will share with no one else.

TRADEMARK ACKNOWLEDGMENTS

Contents

1

The Software Challenge

1.1 INDUSTRY GROWTH MEANS MORE SOFTWARE

- *Point:* In 1983, IBM sold more personal computers (IBM/PC) than it had sold all other types of computers since it was founded. More than 90% of these IBM/PCs were sold to business and government.
- *Point:* In less than five years Apple Computer Company moved from being a garage operation to the Fortune 500.
- *Point:* In the 1970s, the Radio Shack Division of Tandy Company moved into the personal computer business. The TRS-80 computer line now accounts for the majority of Tandy's revenue.
- *Point:* In 1983, *Time* magazine's "Man of the Year" was the computer.
- *Point:* In 1983, President Ronald Reagan announced the Strategic Defense Initiative (SDI), known to the press as the "Star Wars Defense Program," a program of high-technology defense systems predicated on a fewer number of "smart" (i.e., software controlled) weapons rather than larger numbers of troops and conventional (dumb) weapons.

In the last decade, there has been an explosive growth in the rate of proliferation of computers. Daily, new companies are springing up offering faster, cheaper, more powerful devices. As the number of new computers grows, so too grows the need for computer software, which is required to make the computers perform. The demand for software is being driven by the