FOURTH EDITION

MANAGING THE STRUCTURED TECHNIQUES

EDWARD YOURDON

YOURDON PRESS COMPUTING SERIES

MANAGING THE STRUCTURED TECHNIQUES

Fourth Edition

EDWARD YOURDON

Prentice Hall
Englewood Cliffs, New Jersey 07632

Library of Congress Cataloging-in-Publication Data

```
Yourdon, Edward.
   Managing the structured techniques / Edward Yourdon. -- 4th ed.
      p.   cm.
   Bibliography: p.
   Includes index.
   ISBN 0-13-551680-3
   1. Electronic data processing--Structured techniques.  2. Computer
  software--Development--Management.   I. Title.
  QA76.9.S84Y68 1988
  005.1'13--dc19                                                88-31643
                                                                    CIP
```

Editorial/production supervision: Jacqueline A. Jeglinski
Cover design: Bruce Kenselaar
Manufacturing buyer: Mary Ann Gloriande

Printed in the United States of America
10 9 8 7 6 5 4 3 2 1

ISBN 0-13-551680-3

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., *Toyko*
Simon & Schuster Asia Pte. Ltd., *Singapore*
Editora Prentice-Hall do Brasil, Ltda., *Rio de Janeiro*

MANAGING THE STRUCTURED TECHNIQUES

To my son,
David Nash Yourdon,
who is already a better writer than his father.

Preface

Structured analysis is obsolete. Structured design is irrelevant. And structured programming is passé.

If you believe these gloomy pronouncements, here are a few more: Capitalism is doomed; greed is out. Organized religion is on the wane; soon, there will be no more churches. And Congress is going to vote next year to abolish alcohol and tobacco. They're even going to balance the budget.

No, the structured techniques are not dead. They are very much alive, and they continue to evolve to adapt to new technologies and to new *paradigms*. One important paradigm that emerged in the 1980s and that has become widely accepted is prototyping. Its proponents argue that there is no point trying to "pre-specify" the requirements for a system because users don't really know what they want, don't really know what possibilities and alternatives are open to them, and simply cannot understand abstract models such as data flow diagrams.

Thomas Kuhn discusses the concept of "paradigm shifts" eloquently in his book, *The Structure of Scientific Revolutions* ([Kuhn, 1962]); serious students of the software engineering revolution must read this book to put current (and future) developments in perspective. But the systems development field differs from some other scientific disciplines in at least one important way: New paradigms do not necessarily replace old paradigms, but rather enlarge and refine the older ones. When Copernicus proposed a rather radical paradigm shift in the field of astronomy, he presented his audience with a binary choice: People either had to believe that the earth revolved around the sun, or that the sun revolved around the earth. You couldn't have it both ways.

But in the systems development field, you *can* have it both ways. There is no need to make a binary choice between the prespecified (structured analysis) paradigm and the prototyping paradigm. They both work. Sometimes it makes more sense to build a formal, abstract system model on paper and explore its characteristics; sometimes it makes more sense to build a prototype and let the user kick the tires and drive the system around the block to get a feeling for it.

The *really* significant change in the past ten years is the recognition, in the "savvy" MIS organizations, that there is an enormous spectrum of information systems; different paradigms (and methodologies, guidelines, techniques, programming languages, etc.) are required, depending on the kind of system being built, the users who want the system, the environment in which the system will operate, etc.

From this perspective, I now view *structured systems development* as a "metaparadigm": an attempt to bring together several independent paradigms that have heretofore been advertised as competitive and mutually exclusive. I use the word "advertise" deliberately. It is unfortunate that both the academicians and the consultants in this industry have been driven by greed and ego to promote their ideas in "brand name" form. But this has happened throughout history. As Kuhn points out, the fiery, young revolutionary eventually becomes a conservative, old fussbudget, desperately trying to defend his old paradigm against the onslaught of new ideas.

In the systems development field, practical programmers and analysts won't tolerate this any longer. Any single-minded paradigm becomes quickly tarnished and discredited if it is applied in a maniacal fashion to *all* systems and *all* projects. Thus, there is more and more of a movement underway to take the best concepts and ideas from each competing brand-name methodology— the best ideas of the Gane-Sarson approach, the Warnier-Orr-Jackson approach, the Yourdon-DeMarco-Constantine approach, the Martin approach, and the concepts of Dijkstra, Wirth, Parnas, Brooks, and dozens of others— into a metaparadigm that "works" within the framework of a specific MIS organization. As this happens, brand names will disappear.

This process has just begun, and it has a long way to go; along the way, there will still be onslaughts from new paradigms that seek to obliterate existing paradigms by forcing yet another binary choice. As this edition was being written, *object-oriented design* was playing that aggressive role in MIS organizations around the country. So we will have several years of "two steps forward and one step back" before we have a seamless integration of paradigms in even the best of MIS organizations.

Indeed, it is more likely to be an ongoing, never-ending process, because as long as we continue to have quantum-leap improvements in hardware technology, we will continue to change the way we think about computers and systems. I think we will be well into the next century before advances in hardware technology begin to level off—and that's a conservative view at best.

The primary purpose of this book is not to show how the technical aspects of various paradigms mesh together; that would require *several* books. However, I have incorporated some of the changes that have taken place in the well-defined field of structured analysis and structured design; specifically, this book eliminates the classical concept of building a "current physical" and "current logical" model in structured analysis. Instead, it presents the "event partitioning" approach discussed in such books as [McMenamin and Palmer, 1984], [Ward and Mellor, 1985], and [Yourdon, 1989]. And the discussion of CASE tools has been updated to reflect current developments; however, I have omitted references to specific vendors and products in this area, because they would surely be obsolete by the time this book is published.

Most of the book, though, is not concerned with the technical issues per se, but rather with the *management* and *cultural* issues surrounding structured techniques. Anticipating and preventing paradigm wars is probably the most important thing you an MIS manager can do as we end this turbulent decade of the 1980s; helping your staff grow and accommodate new paradigms, thus enlarging their ability to build systems, is the most important long-term investment you can make.

A few other changes in this edition require brief comment. The chapter on program librarians in the previous edition has been dropped. CASE tools and other productivity aids have rendered the notion of a human assistant obsolete in most MIS organizations. The chapter on chief programmer teams still takes a gloomy view of its practicality in large MIS organizations, but acknowledges its stunning success in many of the smaller PC-based software development organzations. And, of course, the bibliography has been enlarged and brought up-to-date; it is by no means an exhaustive list of everything that has been written on software engineering and structured techniques, but it certainly represents a basic library for the professional systems developer.

It is a pleasure to acknowledge the assistance of many people who helped me produce this new edition of *Managing the Structured Techniques*. Bob Spurgeon and Adrian Bowles read the manuscript and made many helpful suggestions. Ed Moura, the managing editor of Yourdon Press, encouraged the project from the beginning and never complained as deadline

after deadline was missed. And last (though probably first given my verbosity), my editor, Jackie Jeglinski, for her invaluable help and patience throughout the project.

Edward Yourdon
New York City
January, 1989

Contents

Chapter 1
INTRODUCTION

1.1 THE STRUCTURED REVOLUTION

The issue here is not technology, but the *management* of technology. I don't care whether you've ever heard of structured analysis, or whether you know what a leveled data flow diagram is. It's not important whether you are familiar with the structured design concepts of coupling and cohesion, or whether you believe that structured programming is more practical in Pascal than in COBOL.

What *does* matter is the way you introduce these and other new software development technologies into your organization, and how you manage their use. With no management involvement, structured techniques and software engineering will attract some followers at the grass-roots level, but will not have any significant impact. *Programmer* productivity might be increased for the few individuals who decided to follow an organized, disciplined approach to the development of systems, but *project* productivity would not necessarily increase.

And *project* productivity is not even enough. As we enter the 1990s, it is vitally important that we begin focusing on *enterprise* productivity—because other enterprises around the world are focusing on this level of productivity, and it promises to have an ever-increasing impact on the profitability and very survival of many organizations.

Indeed, we must eventually focus on *national* productivity, for information systems are becoming a larger and larger part of our overall economy. In 1985, the information processing industry represented approximately 8 percent of the Gross National Product in the United States; in 1990, it will be 15 percent. Thus, the *quality* of our information systems and the *productivity* of the people building those systems are now every bit as important as quality and productivity were in the steel industry in the 1960s and in the automobile industry in the 1970s. Just as American dominance in many of the traditional "smokestack" industries came under heavy attack from Europe, Japan, Asia, and various Third World countries, so the American