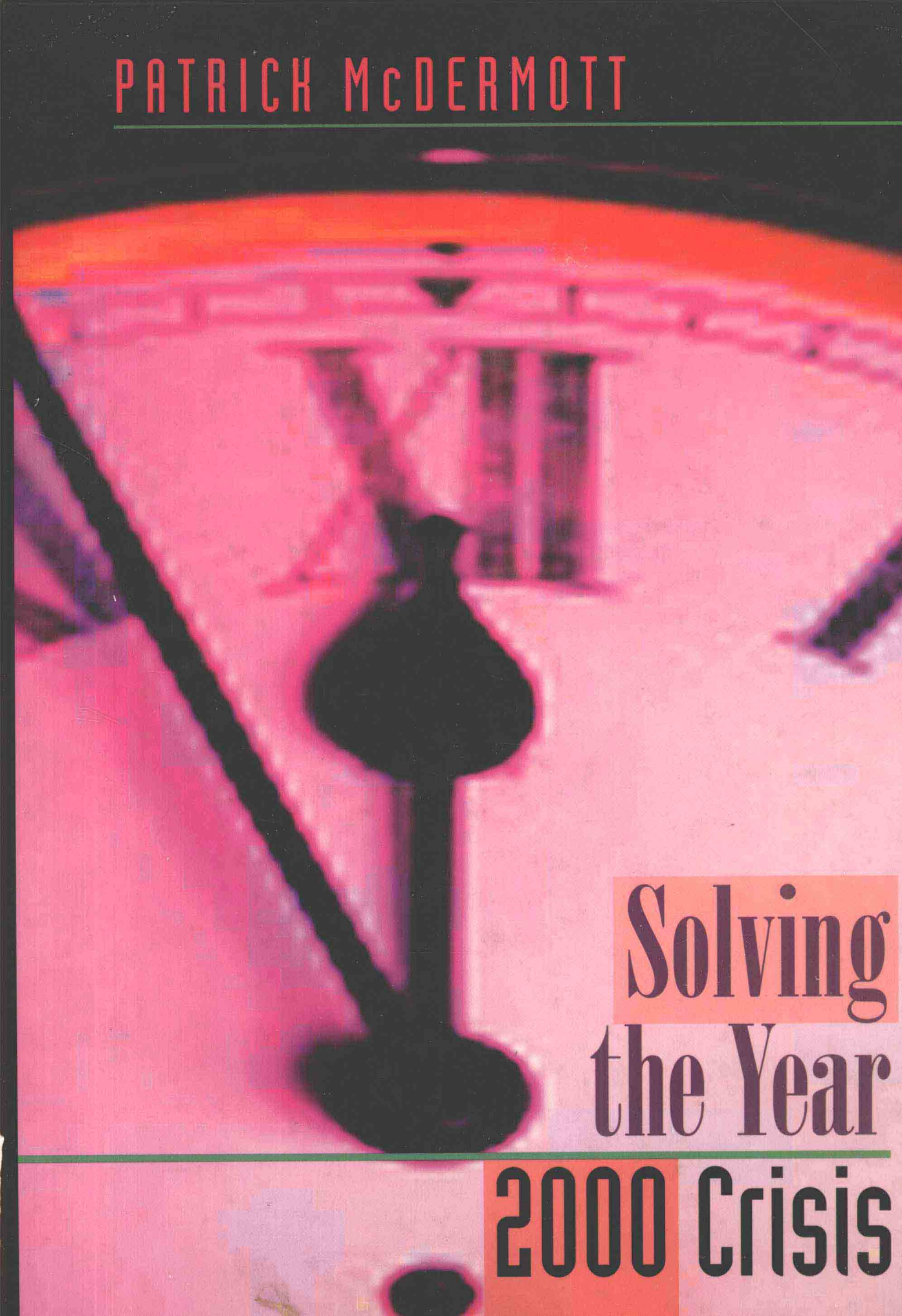
PATRICK McDERMOTT



# Solving the Year 2000 Crisis

# Solving the Year 2000 Crisis

Patrick McDermott

Artech House
Boston • London

**Library of Congress Cataloging-in-Publication Data**
McDermott, Patrick.
Solving the Year 2000 crisis / Patrick McDermott.
p. cm.— (Artech House computer science library)
Includes bibliographical references and index.
ISBN 0-89006-725-2 (alk. paper)
1. Year 2000 date conversion (Computer systems) I. Title.
II. Series
QA76.76.S64M37 1998
005.1'6—dc21 98-2923
CIP

**British Library Cataloguing in Publication Data**
McDermott, Patrick
Solving the year 2000 crisis.—(Artech House computer science library)
1. Year 2000 date conversion (Computer systems)
I. Title
005.1'6

ISBN 0-89006-725-2

**Cover and text design by Darrell Judd**

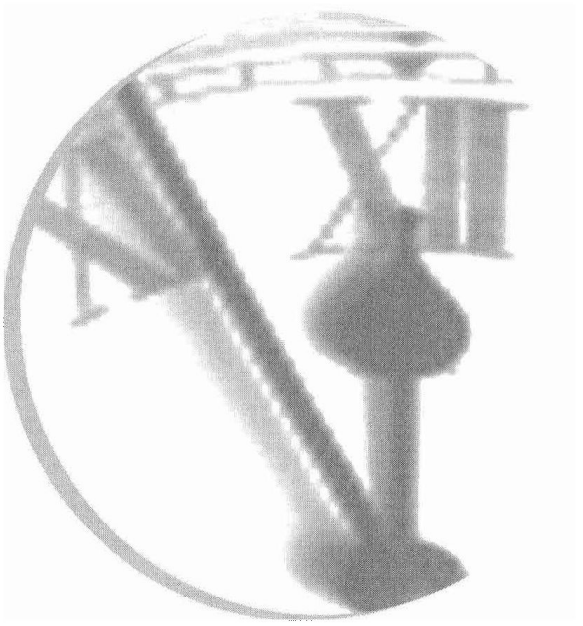

**685 Canton Street**
**Norwood, MA 02062**



International Standard Book Number: 0-89006-725-2
Library of Congress Catalog Card Number: 98-2923

10 9 8 7 6 5 4 3 2 1

*To Lilian*



# Foreword

OVER THE YEARS, Patrick's path and mine have crossed on the winding road to creating software quality. He has taken many of my seminars, learning the various methods and models that advocate the motto of Logical Conclusions, Inc.: "No Bugs."

From the name we have given them, I think of bugs as "innocent" little things that crawl into a system when the original computer programmer isn't looking. It is ironic that my mother, who has never seen a line of computer code in her life, knows what a bug is. What an awful situation the systems profession has gotten itself into when the layperson has a well-known word for such an unprofessional aspect of a business.

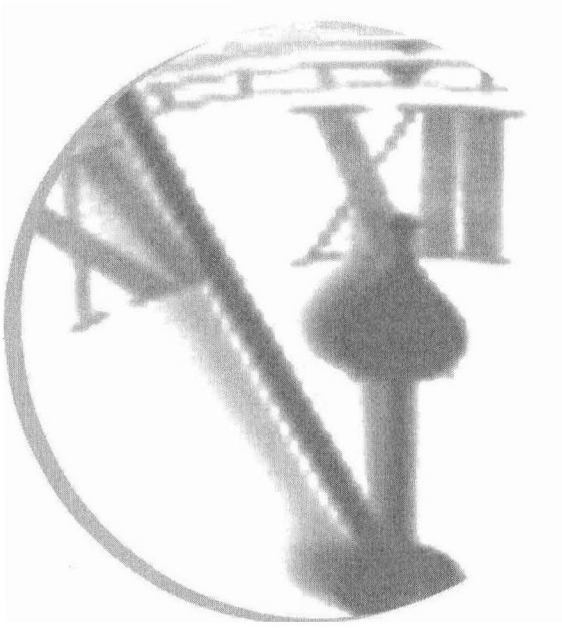
Patrick has dedicated a good part of his recent professional life to perhaps the biggest bug known to the layperson. The Y2K problem definitely is not something that accidentally crawled into systems but something that was deliberately set and discovered about as far into systems' life cycles as a bug can get.

The Y2K problem reminds me of a quote I often used in my programming seminars. Edsgar Dykstra, one of the granddaddies of structured programming, said (as near as I can remember), "More programming sins have been committed under the guise of efficiency than any other reason, including blind stupidity."

The Y2K problem stems from an efficiency issue. Ironically, the effort to clear up the problem appears to be costing most companies far more than the original systems' development effort and the savings in storage space over the years.

If only we would learn from such messes and resolve to do better in the future. The way is clear; it only requires following the discipline already available to do it right the first time. Meanwhile, I believe Patrick's book should be of great help in eradicating this especially nasty bug before it metamorphoses into a disastrous epidemic.

*Brian Dickinson*
*Logical Conclusions, Inc.*
*Lake Tahoe, California*
*November 1997*



# Preface

When I was in the fifth grade, I lived at Sewart Air Force Base near Smyrna, Tennessee. I delighted in observing the many varieties of insects that inhabited the streams and fields near my home and decided I would be an entomologist when I grew up. That was a momentous decision, since it required abandoning my long-held (since early in the fourth grade) plan to be a Nobel prize–winning chemist. Then I read that chemical companies were one of the largest employers of entomologists, and my two passions were complementary: I would be an entomologist for a chemical company. It was some months before the awful realization struck me that the tie between chemistry and entomology is insecticides and that the goal of most entomologists is to kill as many insects as possible. As a consequence, I abandoned my plans for both chemistry and entomology and eventually found a career that involved exterminating bugs without harming any insects: computer programming. So it would seem natural that I would be drawn to what has been called "the mother of all computer bugs" [1]. The problem of how to deal with

the year 2000, or Y2K, in computer software may seem deceptively simple, but writing this book has drawn on all my quarter century of experience as an economist, a programmer, an analyst, and a manager.

My audience is the workers, both business and technical, and front-line managers actually dealing with the nitty-gritty of the problem in large corporations and government agencies. A number of excellent books are out there that cover planning and initiating Y2K projects, but this book is intended as a view from the trenches covering the practical aspects of actually getting the systems fixed. Perhaps because my background is both computer science and economics, I take a pragmatic approach. The purpose of the book is threefold: (1) to give you the *knowledge* you need to carry out the job, (2) to give you the *sense of urgency* you need to keep going through a tough project, and (3) to impart the *confidence* you need to tackle it.

First, the knowledge. This book is divided into five parts designed to provide Y2K teams with the facts needed to solve the problem. Part I describes the problem in detail; Part II covers the mechanical solutions available to fix the problem; Part III discusses the people issues surrounding the project; Part IV explains how to organize the project; and Part V takes a look at the business aspects of the problem.

Second, I want to give you a sense of urgency. If you have not started your Y2K project, you must do so *now*. As the following table shows, time is running out rapidly. The table shows the number of calendar days, normal workdays, and weekends to January 1, 2000. For many Y2K teams, days and workdays will be the same, since they'll be working through the weekends!

Countdown to January 1, 2000

| As of | Days | Workdays | Weekends |
|---|---|---|---|
| 1/1/98 | 730 | 501 | 104 |
| 2/1/98 | 699 | 480 | 99 |
| 3/1/98 | 671 | 461 | 95 |
| 4/1/98 | 640 | 439 | 91 |
| 5/1/98 | 610 | 418 | 87 |
| 6/1/98 | 579 | 398 | 82 |
| 8/1/98 | 518 | 354 | 74 |
| 7/1/98 | 549 | 376 | 78 |

Countdown to January 1, 2000 (continued)

| As of | Days | Workdays | Weekends |
|---|---|---|---|
| 9/1/98 | 487 | 333 | 69 |
| 10/1/98 | 457 | 312 | 65 |
| 11/1/98 | 426 | 291 | 60 |
| 12/1/98 | 396 | 272 | 56 |
| 1/1/99 | 365 | 251 | 52 |
| 2/1/99 | 334 | 231 | 47 |
| 3/1/99 | 306 | 212 | 43 |
| 4/1/99 | 275 | 189 | 39 |
| 5/1/99 | 245 | 167 | 35 |
| 6/1/99 | 214 | 147 | 30 |
| 7/1/99 | 184 | 125 | 26 |
| 8/1/99 | 153 | 104 | 21 |
| 9/1/99 | 122 | 82 | 17 |
| 10/1/99 | 92 | 61 | 13 |
| 11/1/99 | 61 | 41 | 8 |
| 12/1/99 | 31 | 21 | 4 |

Last but not least, I hope to impart confidence. Fixing your Y2K bugs will be a major project and one that will have a planned return on investment (ROI) of zero (some of your projects may have had no ROI, but this is the first time it has been planned that way). But the sky is not going to fall, and with some planning, foresight, common sense, and plain hard work, we'll get through this crisis, although not without some problems. If managed properly, the cost of the project should be in the same order of magnitude as many major development efforts your organization has successfully completed in the past.

The following table lists some possible assessments of the Y2K problem and its eventual effects on the world economy. My assessment is about a 5. I think this is a tough job that will take hard work, but one I am confident we can handle if we get serious about it.

*Let's just do it!*

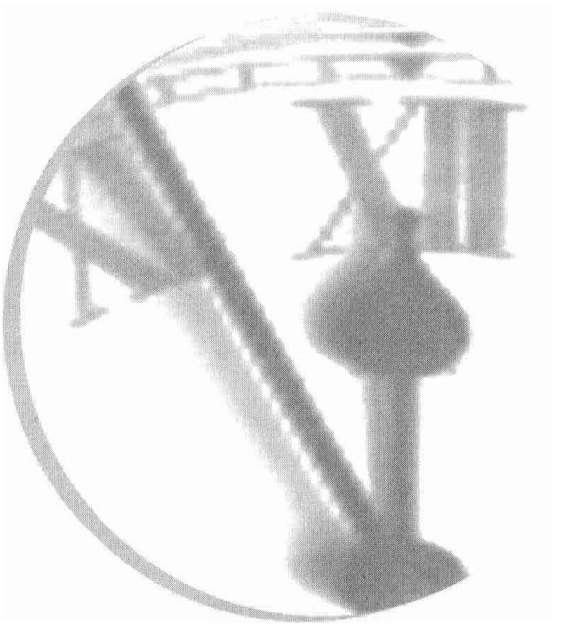
How bad is it?

| Score | Assessment |
|---|---|
| 0 | It is a hoax perpetrated by vendors, consultants, and lawyers to make a lot of money. |
| 1 | It is a minor problem that easily can be fixed. |
| 2 | It is no big deal. |
| 3 | The problem should be addressed, but it is not serious. |
| 4 | It is a serious problem if not fixed, but it is well within the realm of what can be done. |
| 5 | It is a serious problem that will require hard work, but it can be fixed without disaster. |
| 6 | The problem can be fixed but will be very expensive. |
| 7 | It is a serious problem that will have huge costs and cripple some large companies. |
| 8 | The costs will be so severe as to cause major economic repercussions. |
| 9 | The problem will cause severe disruptions to our way of life for at least several months. |
| 10 | The problem will be the end of civilization as we know it, and we never will recover. |

*Patrick McDermott*
*Oakland, California*
*May 1998*

## REFERENCE

1. Keogh, J., *Solving the Year 2000 Problem*, Boston: AP Professional, 1997, p. 7.



# Acknowledgments

ONE OF THE NICEST THINGS about the Y2K problem is that I keep running into people I haven't seen in years. Thanks to two friends from the "old days" at the California Division of Labor Statistics for their ideas and encouragement both then and now, Jerry Freeman and Colt Rymer.

Brian Dickinson of Logical Conclusions, Inc., was a source of advice and inspiration, both in reviewing material for this book and in classes I have taken from him over the years. Brian's ideas and teaching have improved the quality of systems developed for the State of California and American President Lines, so I know there is an approach to system development that leads to a logical conclusion.

Y2K also has introduced me to many interesting experiences and acquaintances and some new friends. Thanks to Anthony M. Peeters of San Francisco Computer Consultants, Inc. He has been a source of insight and wisdom, and his selfless work in forming and running the San

Francisco Bay Year 2000 User Group is helping to alleviate the impact of Y2K in the Bay area.

Thanks to the staff and students at the University of California extensions at UC Santa Cruz, UC Davis, UCLA, and UC Berkeley for their help, questions, and comments, which have improved the book tremendously. Special thanks to Nancy Bruss at UC Davis for suggesting a general seminar on the subject that got me started teaching at UC; to Dan Clarke and Sandra Clark for their ideas and support on the Y2K COBOL course at UC Santa Cruz; and to Sandra for her help with turning the concept into a reality.

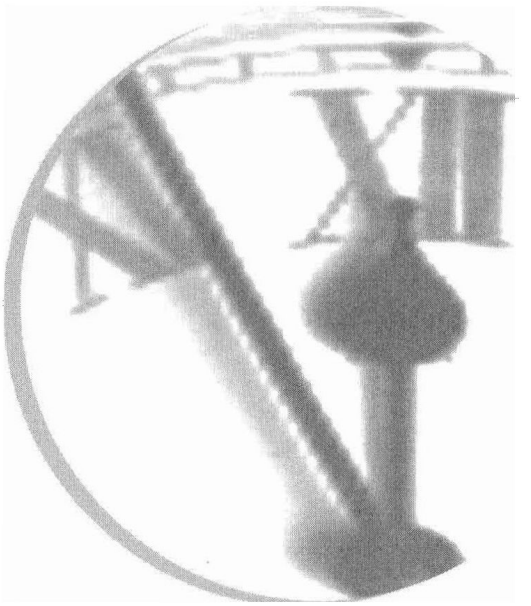
Bill Payson of Senior Staff 2000 originally suggested the concept of a Y2K COBOL course at UC Santa Cruz that led to my involvement in a project that helped shape this book. His spirit of determination no doubt will continue to inspire us to tackle the Year 2000 challenge.

Mike Gee was born in Okinawa, Japan, where I went to high school. The Japanese word *omoshiroi* means both "fun" and "interesting." It was both fun and interesting being featured in USoft's Year 2000 seminar series in the San Francisco Bay area. Thanks to Mike for including me and for his insights on replacement as a solution to Y2K.

Thanks to Jonathan Plant at Artech House for acquiring the book and his many acts of kindness along the way. It has been a pleasure to work with him and the other fine people at Artech House.

Many thanks to Peter de Jager and the subscribers of his Maillist for many interesting discussions and observations that were both informative and amusing and the spirit of helping others that he has championed.

Words can never adequately thank Lilian Roberts for her support and understanding. Without her encouragement, I might never have completed the project. Her help in crafting sentences about topics she didn't understand and acting interested in what is to most an inherently uninteresting topic significantly contributed to any merit this book might have.



# Contents