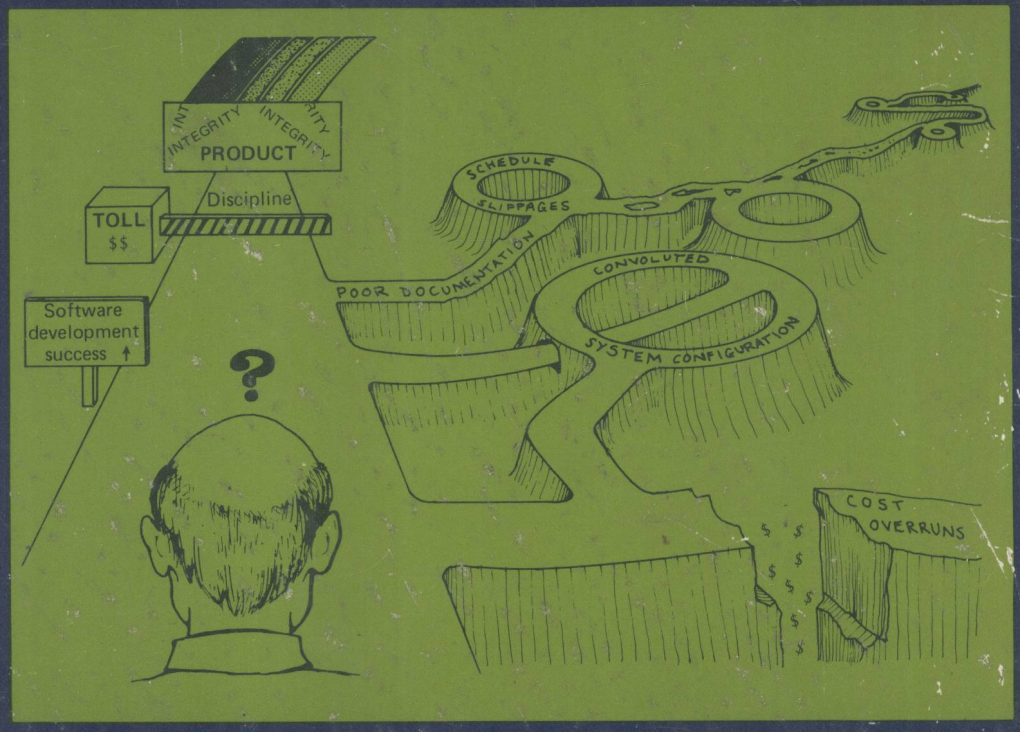
Software Configuration Management

An Investment in Product Integrity



EDWARD H. BERSOFF
VILAS D. HENDERSON
STANLEY G. SIEGEL

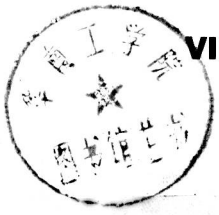
Software Configuration Management

An Investment in Product Integrity

EDWARD H. BERSOFF
CTEC, Inc.

VILAS D. HENDERSON
Desiderum, Inc.

STANLEY G. SIEGEL
CTEC, Inc.



PRENTICE-HALL, INC., Englewood Cliffs, N.J. 07632

Library of Congress Cataloging in Publication Data

BERSOFF, EDWARD H.
Software configuration management, an investment in product integrity.

Bibliography: p. 352
Includes index.
1. Computer programming management. 2. Computer programs—Verification. I. Henderson, Vilas D., joint author. II. Siegel, Stanley G., joint author. III. Title.
QA76.6.B473 658' .054'25 79-26678
ISBN 0-13-821769-6

Editorial/production supervision and interior design by Gary Samartino
Cover design by Edsal Enterprises, Inc.
Manufacturing buyer: Joyce Levatino

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

PRENTICE-HALL INTERNATIONAL, INC., *London*
PRENTICE-HALL OF AUSTRALIA PTY. LIMITED, *Sydney*
PRENTICE-HALL OF CANADA, LTD., *Toronto*
PRENTICE-HALL OF INDIA PRIVATE LIMITED, *New Delhi*
PRENTICE-HALL OF JAPAN, INC., *Tokyo*
PRENTICE-HALL OF SOUTHEAST ASIA PTE. LTD., *Singapore*
WHITEHALL BOOKS LIMITED, *Wellington, New Zealand*

To

Carol
Susan
Rochelle

Jan

Bena
Gary
Deborah
Rachel

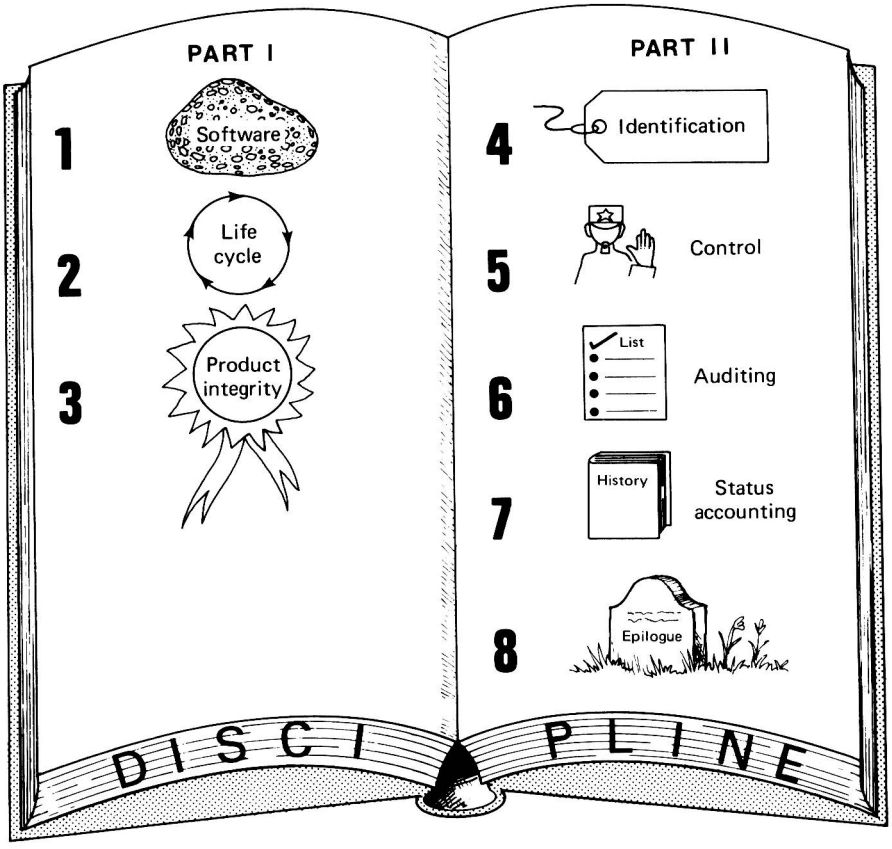


Figure P-1 What This Book Is About.

Preface

Does any of the following sound familiar or conjure up painful memories:

> "I thought the computer system I paid for was supposed to draw pie charts—not draw charts of pi!"
>
> "Sir, before I hand in my resignation, I would like you to hear my side of the story as to why the software we promised the customer two years ago has not been delivered."
>
> "What do you mean the money budgeted for this computer system development has already been expended, and no software has been produced?"
>
> "Bob Bitbuster—the whiz programmer who left the company last week—is the only guy who knows what this computer program is supposed to do?"
>
> "Does anybody know why we have the same subroutine called by three different names?"

This book is about discipline. It is about discipline that managers should apply to software development. Why is such discipline needed? Quite simply, because the software industry has traditionally behaved in an undisciplined manner—doing its own thing. The industry has typically turned out products that have, among other things,

Contained other than what was expected (usually less, rather than more);

Been delivered much later than scheduled;

Cost more than anticipated;

Been poorly designed;

Been poorly documented.

If you can empathize with any of the individuals quoted above, then you should find it beneficial to read this book. If you have been exposed to the practices referred to above (even through hearsay), then you should find it beneficial to read this book. In short, if you are now, or intend to be, a software seller, buyer, or user, then you should find it beneficial to read this book.

Lest you think that you are not now or ever will be a software user, buyer, or seller—keep in mind that the technological explosion of the 1970s in electronic component miniaturization ushered in the era of personalized computing. Without too much exaggeration, it can be asserted that nearly everyone is a potential software seller, buyer, or user.

This book is about a discipline called *software configuration management* (SCM). As the book's subtitle suggests, the theme of the book is that an investment in the discipline of SCM represents a step along the road to product integrity. The objective of SCM is to assist the software seller in achieving product integrity and to assist the software buyer and user in obtaining a product that has integrity.

In writing this book, we have adopted a textbook approach. We feel that this approach is appropriate because, to our knowledge, a book that addresses the subject of SCM in depth has heretofore not been written. Many concepts may therefore be new to the general readership. Consequently, these concepts will require reinforcement. We have found the standard textbook devices of detailed examples as well as exercises to be solved by the reader to be effective vehicles for achieving this reinforcement. The purpose of many of the exercises is to provoke classroom discussion and debate; some of the exercises extend concepts introduced or touched upon in the main text.

We asserted earlier that nearly everyone is a potential software user, buyer, or seller. Therefore, our aim in this book is to appeal to a broad audience. To achieve this aim, we have pursued the following course:

The style is expository to accommodate classroom use. However, we have slanted the style toward the informal to avoid pedantry.

The text is reasonably self-contained to accommodate self-study. Complex ideas and elaborate extensions of basic concepts are left for exercises.

To ease into the presentation of technical material, we begin each chapter except the last with examples taken from everyday experience. We thus appeal to commonly understood notions to motivate the discussion of technical concepts.

Mathematical symbology is not used extensively. Other than a knowledge of some concepts taught in high school algebra, a background in mathematics is neither

assumed nor required. However, for those of you with such a background, we have included some material that should provoke your interest. This material, which may be skipped without loss of continuity, has been placed in footnotes, exercises, and (in Chapter 4) a separate section.

A knowledge of (or even exposure to) computer programming is neither assumed nor required. Such knowledge may even be a disadvantage in some cases (those of you with programming experience may be compelled to so some rethinking). However, we have included material that does assume software development experience. In order not to disrupt continuity, this material has, for the most part, been placed in footnotes and exercises.

We therefore believe that the material in this book should be accessible to individuals such as:

1. Undergraduate students in the computer science, management science, and operations research disciplines (the material should be more than adequate for a one-semester course).
2. System development managers and contract managers in industry and government.
3. Computer system* designers, analysts, and programmers.
4. Software vendors.
5. Anybody who wants to learn something about software and the management of its development.

This text is not a cookbook. It does not prescribe in detail the steps required to perform SCM on a particular software development project. For this reason, this book does not contain pages of blank forms that you can reproduce and fill in to accomplish SCM on a particular project. Rather, this book sets forth SCM *principles* that you can apply to your particular needs. We supplement the presentation of these principles with detailed examples so that you can see how to perform this application. Even if we believed that it were possible to write an SCM cookbook, we surmise that such a treatise would suppress creativity in software development management.

A comment is in order concerning our use of examples. To maintain continuity of presentation, and to strengthen the credibility of the theoretical discussions, we have threaded a case study throughout the final five chapters of the text. This case study represents a composite of the authors' software development experiences. It is used to explicitly illustrate the utility of applying the discipline of SCM throughout

*Unless otherwise indicated in this book, the term *computer system* encompasses entities ranging from pocket and desktop programmable calculators (such as the TI-59 or HP-41C) to minicomputers (such as the PDP-11) to maxicomputers (such as the IBM/370 series or the CRAY-1 array processor). The SCM discipline is applicable in some degree to computer systems of any size.

the entire software/system life cycle. We also explicitly illustrate some of the difficulties that typically result from not applying this discipline. In addition to the case study, we have also incorporated other examples into the text that illustrate particular points of SCM methodology that could not be conveniently or adequately addressed in the case study.

We have also made liberal use of figures and diagrams. We feel that pictures help considerably in visualizing many of the SCM concepts that may be difficult to grasp through the written word alone. Also, we feel that the text would be unduly ponderous were it not liberally interspersed with figures.

The book is divided into two parts (see Figure P-1). Part I, entitled "The Pursuit of Product Integrity" and consisting of the first three chapters, builds a case for the need for SCM. Those of you having an interest in the software development process should find these three chapters informative. Part II, entitled "The Principles" and consisting of the remaining five chapters, presents and illustrates the principles of SCM.

The following is an overview of the book's eight chapters:

- In Chapter 1, we introduce the concept of *software.* Our definition is perhaps more comprehensive than conventional definitions and is fundamental to the proper understanding of the SCM problem. The similarities and differences between software and other types of systems are discussed. In this chapter, we also provide motivation for attempting to solve the SCM problem. The chapter concludes with the introduction of the functions of configuration management.
- In Chapter 2, we focus on the concept of *system.* We describe the life cycle of a system. We indicate the relationship between the life cycle of a system and the life cycle of the software components of this system. The life cycle concept is used to introduce the fundamental SCM concept of *baseline.* Management of the system life cycle is discussed in terms of this concept.
- In Chapter 3, we discuss the disciplines needed to attain and maintain product integrity. We divide these disciplines into three groups—management, "doing," and product assurance. We indicate the role of SCM—a product assurance discipline—vis-à-vis the role of other disciplines in the pursuit of product integrity. To prepare for the detailed treatment of SCM in subsequent chapters, we introduce the notions of *software design object* and *software configuration item.*
- In Chapter 4, we begin our detailed study of the SCM discipline. We examine the first of the four classical configuration management functions—configuration identification—as it applies to a system consisting in part of software. Compared to existing literature, our discussion of this function is perhaps innovative. The concept of baseline introduced in Chapter 2 provides the basis for developing configuration identification techniques. A generalized approach to labeling software configurations is presented.

Through the case study, we illustrate how this generalized approach can be tailored to specific project needs. To lay the groundwork for Chapter 5, the concept of *baseline change* is introduced.

- In Chapter 5, we examine the second classical configuration management function—configuration control—as it applies to a system consisting in part of software. We show how management of configuration changes is the key to disciplining software life cycle evolution. For this purpose, we focus on the concept of *configuration control board.* We introduce the concepts of *evolutionary change* and *revolutionary change.* We discuss the documentation (i.e., forms) needed to support change processing. We also address the subject of the ingredients of a configuration management plan. Again, through the use of the case study and other specific examples, we illustrate the application of configuration control in specific terms.
- In Chapter 6, we examine the third classical configuration management function—configuration auditing—as it applies to a system consisting in part of software. Auditing is the ultimate payoff of applying the SCM discipline. Through the auditing function, the software seller is able to determine whether or not what the buyer/user asked him to build has indeed been built, and the software buyer/user is able to determine whether what he is paying for is what he asked the seller to build. Auditing checklists are presented for each of the baselines introduced in Chapter 2. Tools and techniques for conducting SCM audits are described. We use the case study and other specific examples to illustrate the accomplishment of the configuration auditing function.
- In Chapter 7, we examine the fourth and final classical configuration management function—configuration status accounting—as it applies to a system consisting in part of software. We discuss the need for configuration status accounting and discuss techniques for implementing this function. The case study is used to illustrate specific applications of the configuration status accounting function.
- In the final chapter, we provide perspective to the concepts and principles described and illustrated in the preceding chapters. We shift our attention from the microscopic aspects of how each of the four SCM functions is accomplished to macroscopic considerations. We explore, using the case study, the manner in which these four functions interact with one another. We examine whether the application of SCM really results in a better product. We offer some guidelines for determining how much discipline is needed on a particular software project. We conclude the book by offering some suggestions regarding how you may wish to apply the material in the book to your work and other endeavors.

With the preceding as a guide, let us begin our pursuit of software product integrity.

ACKNOWLEDGMENTS

Several organizations and individuals provided moral and/or physical support for the development of this book. To CTEC, Inc. goes our appreciation for the facilities without which this book could not have been produced. To the CTEC technical staff goes our appreciation for stimulating ideas and constructive criticism. We particularly wish to thank Marilee J. Layman and Dr. William L. Bryan. They willingly set aside evenings and weekends to review the manuscript and galleys and assist in the preparation of the index. To the CTEC administrative staff goes our appreciation for their willingness to put up with our seemingly endless demands for text revisions and our different colored pens. In particular, we wish to thank Martha Begle for her diligent and tireless assistance during the preparation of the manuscript.

To Marilynn Weidner, we owe a special debt of gratitude that can never be repaid. Her assistance during all phases of this project has been extraordinary. She has brought order to what at times was chaos and has been a source of strength during our moments of despair.

To Paul Becker, we wish to express our appreciation for his guidance and encouragement. From this project's inception, he enthusiastically supported our efforts.

Finally, to our wives and families go special thanks. Their understanding of our need to work early in the morning, late at night, and on weekends is appreciated. Their moral support and willingness to let us do our thing was more than we probably deserved.

Falls Church, Virginia

EDWARD H. BERSOFF

VILAS D. HENDERSON

STANLEY G. SIEGEL

Contents

Preface ix

PART I The Pursuit of Product Integrity

Chapter 1 The Problem of Software 3

1-1 Introduction *3*
1-2 The System in Perspective *6*
1-3 Concept of Software *10*
1-4 Software Development—Art or Science? *18*
1-5 What Is SCM? *19*
Exercises *22*
Footnotes *25*

Chapter 2 Managing the System Development 29

2-1 Introduction *29*
2-2 The System Life Cycle *41*
2-3 Managing the System Life Cycle *46*
2-4 Unfinished Business *49*
Exercises *52*
Footnotes *55*

Chapter 3 Attaining and Maintaining Product Integrity 57

3-1 The Requisites *57*
3-2 Organizing for Product Integrity *61*
3-3 Management's Role in Attaining and Maintaining Product Integrity *66*
3-4 Product Assurance—The Supporting Disciplines *72*
3-5 The "Doing" Disciplines *77*
3-6 System Architectures for Attaining and Maintaining Product Integrity *82*
3-7 And Now! *88*
Exercises *90*
Footnotes *93*

PART II The Principles

Chapter 4 Configuration Identification 97

4-1 The Problem *97*
4-2 The Notion of Software Configuration *107*
4-3 Principles of Software Configuration Identification *113*
4-4 A Guided Tour through the Software Life Cycle from the Configuration Identification Viewpoint *130*
4-5 Software Configuration Identification Formalism *151*
4-6 Summary *157*
Exercises *157*
Footnotes *161*

Chapter 5 Configuration Control 172

5-1 The Problem *172*
5-2 The Configuration Control Board *178*
5-3 The "Forms" of Software Configuration Control *199*
5-4 The Configuration Management Plan *208*
5-5 The Configuration Control Tour through the Software Life Cycle *213*
5-6 Summary *220*
Exercises *220*
Footnotes *223*

Chapter 6 Configuration Auditing 226

6-1 The Problem *226*
6-2 Auditing Principles *237*
6-3 Auditing Applied to the Case Study—A Guided Tour *255*
6-4 Techniques and Tools *270*
6-5 Summary *273*
Exercises *273*
Footnotes *278*

Chapter 7 Configuration Status Accounting 284

7-1 The Problem *284*
7-2 Principles of Configuration Status Accounting *289*
7-3 Configuration Status Accounting Tours the Software Life Cycle *303*
7-4 Summary *310*
Exercises *310*
Footnotes *312*

Chapter 8 Epilogue 315

8-1 Where Have We Been? *315*
8-2 SCM Function Interactions *318*
8-3 What Does the Toll Really Buy? *331*
8-4 How Much Discipline Is Enough? *336*
8-5 Where Do You Go from Here? *340*
8-6 A Final Thought *343*
Exercises *344*
Footnotes *349*

Bibliography 352

Index 363

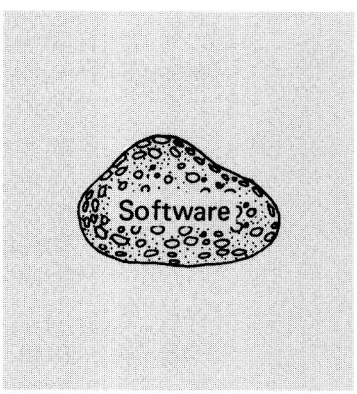
The Pursuit of Product Integrity

PART I

The Problem of Software

1

1-1 INTRODUCTION



This book is about a management methodology applicable to the development of systems comprised partially or totally of software.[1]* This methodology—or, more precisely, discipline—we call *software configuration management,* or SCM.

The theme of this book is discipline—discipline as it applies to the development of software products. Many of the ideas in this book are not complex; in fact, they are almost alarmingly simple! You may therefore ask: Why have we been motivated to write a book about simple ideas? The answer is probably more complex than some of the ideas. We will not dwell on all the aspects of the answer, but we will offer some brief, partial replies to the question. Some appreciation of why this book was written is, we feel, important to becoming acclimated to its spirit.

We begin with a simple illustration. Let us consider what is involved in the writing of a letter (say, a letter of apology from a president of a corporation to a ruffled client). We have all written letters, so if we think about the processes involved,

*Footnotes to each chapter appear after the exercise section at the end of the chapter.