


---

# SOFTWARE ENGINEERING ECONOMICS

---

**Barry W. Boehm**

*Director, Software Research and Technology  
TRW, Inc.*

A large, stylized handwritten signature, likely of Barry W. Boehm, is positioned in the upper right corner of the page. The signature is composed of several fluid, overlapping strokes.

# **SOFTWARE ENGINEERING ECONOMICS**

**Prentice-Hall, Inc., Englewood Cliffs, New Jersey 07632**

*Library of Congress Cataloging in Publication Data*

BOEHM, BARRY W. (date)

Software engineering economics.

(Prentice-Hall advances in computing science and technology series)

Bibliography: p.

Includes index.

1. Electronic digital computers—Programming—Economic aspects. 2. Electronic digital computers—Programming—Economic aspects—Case studies. I. Title. II. Series.

QA76.6.B618 001.64'25'0681 81-13889  
ISBN 0-13-822122-7 AACR2

© 1981 by Prentice-Hall, Inc., Englewood Cliffs, N.J. 07632

All rights reserved. No part of this book  
may be reproduced in any form or by any means  
without permission in writing from the publisher.

Editorial/Production Supervision  
and Interior Design: *Lynn S. Frankel*  
Cover Design: *Carol Zawislak*  
Manufacturing Buyer: *Gordon Osbourne*

**Prentice-Hall Advances  
in Computing Science and Technology Series**  
*Raymond T. Yeh, editor*

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

ISBN 0-13-822122-7

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*

---

# PREFACE

A course in engineering economics has become a fairly standard component of the hardware engineer's education. So far, the opportunities for software engineers to take a similar course tailored to software engineering economics have been rare. As a result, I think most software engineers miss out on a chance to acquire and use a number of significant economic concepts, techniques, and facts which can play a vital part in their future careers—and a vital part in making our software easier to live with and more worthwhile.

Not surprisingly, then, the major objective of this book is to provide a basis for a software engineering economics course, intended to be taken at the college senior/first-year graduate level. This objective has led to two subsidiary objectives:

1. To make the book easy for students to learn from;
2. To make the book easy for professors to teach from.

I have also tried to make the book serve a third objective:

3. To provide help for working professionals in the field.

Since these aims are sometimes at variance with each other, I have added notes to the student, professor, and practicing software engineer as a starting point for dealing with the contents of the book.

The basic structure of the book is shown in Figure A. Part I contains introductory material which provides a context, motivation, and framework of software engineering goals for the material to follow. Parts II and III cover two complementary topics: a quantitative model of the software life-cycle in Part II, and the fundamentals of engineering economics as they apply to software projects in Part III. Part IV then provides the detailed techniques for software life-cycle cost estimation which underlie the simpler cost models in Part II, and which further support the software engineering economic analysis techniques in Part III.

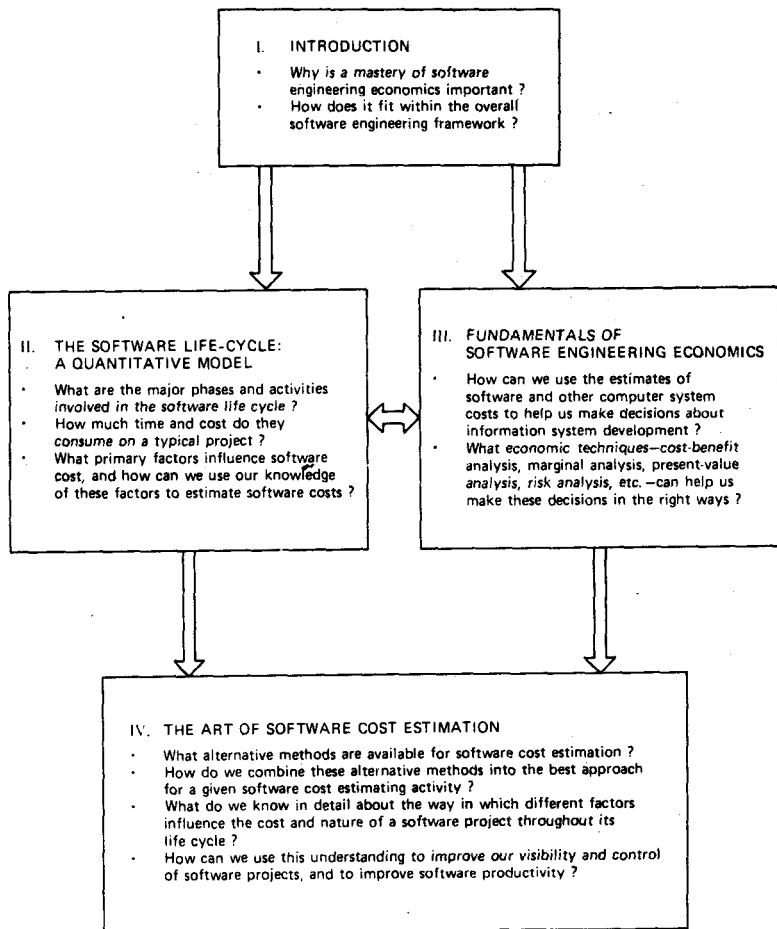
Figure A also shows the primary questions addressed in each part of the book. Thus, for example, Part IV addresses not only questions of software cost estimation and understanding software cost-influence factors, but also such questions as, "How can we use this understanding to improve our visibility and control of software projects, and to improve software productivity?"

Figure B shows how each part of the book is organized into components and individual chapters. This figure is reproduced at the beginning of each major component (Part openings) of the book. For example, Figure B indicates the successive levels of detail provided in the hierarchical software cost estimation model called COCOMO, for COConstructive COSt Model. The top level of the hierarchy is Basic COCOMO, a simple formula estimating the cost of a software project solely as a function of its size in delivered source instructions, presented in Chapters 5, 6, and 7. The next level of the hierarchy is Intermediate COCOMO, presented in Chapters 8 and 9. It estimates the cost of a software project as a function of size and a number of other software cost driver attributes, such as personnel experience and capabilities, computer hardware constraints, and degree of use of modern programming practices. The most accurate and detailed level of the hierarchy is Detailed COCOMO, presented in Chapter 23 with elaborations in Chapters 24 through 27. It uses the cost driver attributes to estimate the software product's costs by individual phase, subsystem, and module.

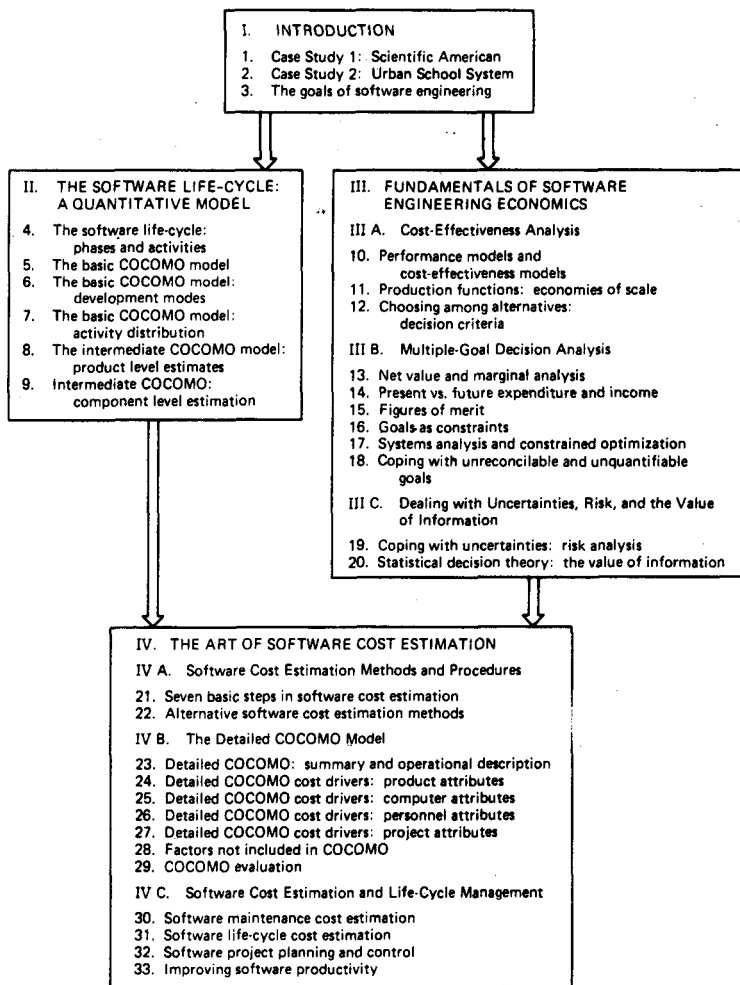
The term "constructive" used to describe COCOMO derives from the detailed explanations in Chapters 24-27 of how the various software cost driver attributes influence the amount of effort required to complete each phase of the software life-cycle. The model not only provides estimating formulas, but it also provides the best explanation possible for why the model gives the results it does. The detailed material in Chapters 24-31 also discusses the frontiers of our knowledge of software life-cycle cost estimation, and provides an extensive agenda of suggestions for further research which can extend our knowledge of the software life-cycle and its economic properties.

For providing me with encouragement, insights, and data, I feel deeply indebted to many people. I wish I could name them all.

At TRW, I have benefited from a great deal of management insight and support from Simon Ramo, C. W. (Bill) Besserer, Bob Williams, and Ed Goldberg, and a wealth of technical information and insight from Tom Bauer, Mike Cozzens, Myron Lipow, Fred Manthey, Nancy Mikula, Eldred Nelson, Ron Osborne, and Tom Thayer; from



**FIGURE A** Book Structure—Major questions addressed



**FIGURE B** Book structure—Parts and chapters

ex-TRW'ers Bert Abramson, Tom Bell, John Brown, Kurt Fischer, Bob Page, and Win Royce; and particularly from Ray Wolverton.

Within the field of quantitative software analysis, I have had many enjoyable, stimulating, and valuable exchanges with Prof. Vic Basili of the University of Maryland, Dr. Les Belady of IBM, Tom DeMarco, Tom Gilb, the late Prof. Maurice Halstead of Purdue University, Capers Jones of ITT, Prof. Manny Lehman of Imperial College, London, Dick Nelson of RADC, Bob Park of RCA, Dr. Montgomery Phister, Jr., Larry Putnam of Quantitative Software Management, Inc., and Claude Walston of IBM.

In other areas, I feel fortunate to have learned much from discussions with Dr. Gerald Weinberg of Ethnotech, on software psychology; with Dr. Dave Parnas of the Naval Research Labs and IBM, Dr. Harlan Mills of IBM, and Prof. Tony Hoare of Oxford University on software methodology; and from numerous informal tennis-court seminars on economics and statistics from Dr. Charles Wolf of Rand and Prof. Carl Morris of the University of Texas.

In preparing this book, I must first acknowledge the essential contributions of Karl Karlstrom of Prentice-Hall and particularly Prof. Richard Hamming of the U.S. Navy Postgraduate School, who more or less goaded me into writing it. My secretaries, Marilyn Gripenwaldt and Kay Clyne, have been exceptionally supportive and helpful. Ms. Lynn Frankel of Prentice-Hall has made the book a great deal better as its editor. I received many valuable suggestions from the reviewers of the manuscript, particularly Prof. Lee Cooper of USC, Prof. Richard Fairley of Colorado State University, Prof. Ellis Horowitz of USC, Brian Kernighan of Bell Labs, Prof. Tim Standish of University of California, Irvine, and David Weiss of the Naval Research Labs.

Finally, for the many necessarily anonymous contributors of software data, a special note of thanks and a wish that they could be recognized directly as well. And I owe my family more than I could ever express in words.

## **A NOTE TO THE STUDENT**

There is a good chance that, within a few years, you will find yourself together in a room with a group of people who will be deciding how much time and money you should get to do a significant new software job. Perhaps one or two of the people in the room will know software well, but most of them will not. They will be higher-level managers, business analysts, marketing specialists, product-line planners, and the like. Generally, they will discuss issues and make decisions in terms of such concepts as marginal return on investment, cost-benefit ratio, present value, and risk exposure.

There will also be a number of highly interested people who won't be in the room. These include the people who will be working for you or with you on the software job, and the people who will have to use the software that your team is going to produce. Whether they know it or not, their fate over the next few months or years will depend a good deal on how well you and the non-software people in the room can produce a realistic decision on the appropriate scope, budget, and schedule for your software job.



The non-software people in the room won't be able to do this by themselves; they won't have the feel for the technical software tradeoffs that you do. So it will be extremely important for you to be able to communicate with them, and understand the economic concepts that underlie the ways they have learned to think and to make decisions. If you can do this, you will have a chance to change what is often an adversary relationship between software people and business-oriented people into a relationship of mutual understanding, commitment, and trust.

In this book, I have tried to provide you with the essential concepts and techniques you will need to be able to think in economic terms as well as to think in programming terms. Besides the practical utility of these concepts, I hope you will find them as stimulating as I have in providing a new perspective on our field of computers and information processing. I've found them very helpful in illuminating such questions as:

- Why does information have value?
- Why do people commission software products?
- How do people decide what information processing products they want?
- Why is the software life-cycle organized the way it is?

And, as with other pursuits, the better we can understand why the software engineering field exists, the better we will become in practicing it.

## **A NOTE TO THE PROFESSOR**

In this note—well, actually, with this note and the material in the book—I hope to convince you of three things:

1. Software engineering economics is a stimulating and satisfying topic to teach and study.
2. This book can be used successfully either for a one-quarter or one-semester course on software engineering economics, or as a secondary text for more general software engineering courses.
3. Software engineering economics is a significant and fruitful research area.

First, I think you will find software engineering economics an enjoyable and rewarding subject to teach. The basic relationships in microeconomics form a nice, clean, mathematical discipline. The material on risk and the value of information provides a stimulating perspective on why so many people feel they need the computers, software, and processed information our field produces. And the material on factors influencing software costs helps to explain a good many current software engineering guidelines and their impact on the software life-cycle.

Further, I don't think that one has to immerse oneself in industrial practice and jargon to find relevant examples and applications of software engineering economics. When I was at USC, I was impressed with the wide variety of computer and software applications being developed around the university, and—particularly in

these days of tight university budgets—the degree of concern with computer and software costs. Accordingly, I have tried to keep the book free of industrial jargon, and to include a good many university-oriented questions and examples in order to keep the material in a familiar context.

The fundamental material in the book can be covered fairly well in a one-quarter or one-semester course. The primary learning objectives I have used in teaching such a course are to enable the student to:

- Identify the factors most strongly influencing software costs, and use these to determine the estimated costs of a software project.
- Understand the fundamental concepts of microeconomics as they apply to software engineering.
- Apply economic analysis techniques to software engineering decision situations.

An outline for a fairly ambitious one-quarter course I have given using the book is provided below:

Week	Book Chapters	Topics
1	1-4	The Software Life-Cycle: An Economic Perspective
2	5-6	Simple Software Cost Models
3,4	7-9	Intermediate-Level Software Cost Models: Factors Influencing Software Costs
5	10-12	Cost Effectiveness Analysis: Production Functions, Economies of Scale, Choosing Among Alternatives
6	—	Review, Midterm Exam
7	13-15	Multiple-Goal Decision Analysis: Net Value, Present Value, Figures of Merit
8	16-18	Multiple-Goal Decision Analysis: Constraints, Systems Analysis, Unquantifiable Goals
9	19-20	Risk, Uncertainty, and the Value of Information
10	21-22	Practical Software Cost Estimation Techniques
11	31-32	Case Study: Software Life-Cycle Cost Analysis and Control
12	—	Final Exam

For the first time teaching a software engineering economics course, the above volume of material is probably better suited to a one-semester course. A satisfactory one-quarter course could cover only the material through Chapter 18, and still satisfy the basic learning objectives reasonably well.

Such a course can be taught at either an upper-level undergraduate or a first-year graduate level. The only prerequisites are a general familiarity with the programming process (the equivalent of about two years' worth of computer science courses) and a familiarity with the basics of differential calculus. For exercising the software cost estimation models, a hand calculator with exponentials (an  $X^y$  key) is strongly recommended, although I have included curves which allow the student to work the models without a calculator, but with much less accuracy and facility.

Finally, I hope you'll get far enough into the subject of software engineering economics to become intrigued with some of the fundamental research questions it raises about the nature of the software development process, such as:

- Why does software development cost as much as it does?
- What factors make the cost of software go up or down, and how do they interact?
- What activities consume most of the cost?
- How can new software techniques reduce software cost?

In Part IV of this book, I have presented and analyzed a data base representing the costs and development attributes of 63 software projects, in an attempt to answer the question:

*"How can we explain this project data in a way that will help future projects estimate and understand their software costs?"*

The resulting set of cost models presented in the book represents a first step toward answering this question, but a tremendous amount of valuable research still remains to be done. A number of significant new insights can be achieved simply by further analysis of the existing 63-project data base. And a great deal more insight can be achieved through collection and analysis of further observational and experimental data. Most of the chapters in Part IV contain a final section on "Topics for Further Research" indicating some of the most promising directions we can go in illuminating the fundamental questions above. I hope you or your students will give them a try.

## A NOTE TO THE PRACTICING SOFTWARE ENGINEER

During your software engineering experience, I would imagine that you have evolved a number of personal guidelines for estimating software costs and for dealing with software product and project decisions. I think you'll find this book helpful in calibrating your own rules of thumb with other people's experiences, and in providing you with some additional useful techniques for dealing with software cost estimation and software engineering decisions. I hope, also, as you go through the book, you can enjoy an experience as stimulating and rewarding as mine has been, as I began to see how various, seemingly unrelated techniques and decision guidelines I had been using in practice were actually parts of a unified framework of economic principles.

Depending on your primary interests and needs for information, you may wish to concentrate on selected portions of this book rather than read it from cover to cover. For some of the likely interests you may have, here are the most appropriate parts of the book to read.

- If you are primarily interested in improving your (organization's) ability to estimate software development costs, your best bet is to begin with Chapters 21 and 22

on software cost estimation techniques, followed by Chapters 4-9 on the software life-cycle and on the Basic and Intermediate COCOMO models.

- \* If you are further interested in estimating maintenance and other software-related costs, read Chapters 30 and 31.
- \* If you are further interested in implementing a detailed software cost estimation model and tailoring it to your organization's experience, read Chapters 23 and 29.
- If you are primarily interested in the effect of a particular software attribute (such as project personnel capability, use of modern programming practices, or language level) on software costs, read the appropriate section in Chapters 24-28.
- If you are primarily interested in improving your ability to perform software economic decision analyses, read Chapters 10-18.
- If you are primarily interested in software project planning and control techniques, read Section 31.6 and Chapter 32.

However, even if you are primarily interested in a particular topic, I would especially recommend your reading the introductory material in Chapters 1-3 and Chapter 33 on improving software productivity. These chapters provide a context and an approach for realizing a more effective, satisfying, and productive environment within which to practice your software engineering activities.

## I. INTRODUCTION

1. Case Study 1: Scientific American
2. Case Study 2: Urban School System
3. The goals of software engineering

## II. THE SOFTWARE LIFE-CYCLE: A QUANTITATIVE MODEL

4. The software life-cycle:  
phases and activities
5. The basic COCOMO model
6. The basic COCOMO model:  
development modes
7. The basic COCOMO model:  
activity distribution
8. The intermediate COCOMO model:  
product level estimates
9. Intermediate COCOMO:  
component level estimation

## III. FUNDAMENTALS OF SOFTWARE ENGINEERING ECONOMICS

### III A. Cost-Effectiveness Analysis

10. Performance models and  
cost-effectiveness models
11. Production functions: economies of scale
12. Choosing among alternatives:  
decision criteria

### III B. Multiple-Goal Decision Analysis

13. Net value and marginal analysis
14. Present vs. future expenditure and income
15. Figures of merit
16. Goals as constraints
17. Systems analysis and constrained optimization
18. Coping with unreconcilable and unquantifiable  
goals

### III C. Dealing with Uncertainties, Risk, and the Value of Information

19. Coping with uncertainties: risk analysis
20. Statistical decision theory: the value of information

## IV. THE ART OF SOFTWARE COST ESTIMATION

### IV A. Software Cost Estimation Methods and Procedures

21. Seven basic steps in software cost estimation
22. Alternative software cost estimation methods

### IV B. The Detailed COCOMO Model

23. Detailed COCOMO: summary and operational description
24. Detailed COCOMO cost drivers: product attributes
25. Detailed COCOMO cost drivers: computer attributes
26. Detailed COCOMO cost drivers: personnel attributes
27. Detailed COCOMO cost drivers: project attributes
28. Factors not included in COCOMO
29. COCOMO evaluation

### IV C. Software Cost Estimation and Life-Cycle Management

30. Software maintenance cost estimation
31. Software life-cycle cost estimation
32. Software project planning and control
33. Improving software productivity

---

# CONTENTS

<b>PREFACE</b>	<b>xix</b>
<b>PART I INTRODUCTION: MOTIVATION AND CONTEXT</b>	<b>1</b>
<b>Chapter 1 Case Study 1: <i>Scientific American</i> Subscription Processing</b>	<b>3</b>
1.1 The Old System	3
1.2 The Programming Solution: Top-Down Stepwise Refinement	4
1.3 The Programming Solution: Results	5
1.4 The Economic-Programming Approach	6

1.5	Results of the Economic-Programming Approach	7
1.6	General Discussion	8
1.7	Questions	8

## **Chapter 2 Case Study 2: An Urban School Attendance System**

**10**

2.1	Programming Aspects	10
2.2	Economic Aspects	10
2.3	Human Relations Aspects	11
2.4	Lessons Learned	11
2.5	General Discussion	12
2.6	Questions	13

## **Chapter 3 The Goals of Software Engineering**

**14**

3.1	Introduction	14
3.2	Software Engineering: A Definition	16
3.3	Software Trends: Cost	17
3.4	Software Trends: Social Impact	18
3.5	The Plurality of Goals	20
3.6	An Example: Weinberg's Experiment	20
3.7	The Plurality of Software Engineering Means	21
3.8	The Software Engineering Goal Structure	23
3.9	The GOALS Approach to Software Engineering	23
3.10	Questions	26

## **PART II THE SOFTWARE LIFE-CYCLE: A QUANTITATIVE MODEL**

**29**

## **Chapter 4 The Software Life-Cycle: Phases and Activities**

**35**

4.1	Introduction	35
4.2	The Waterfall Model	35
4.3	Economic Rationale for the Waterfall Model	38
4.4	Refinements of the Waterfall Model	41
4.5	Detailed Life-Cycle Phase Definitions	46

4.6	Detailed Phase/Activity Definitions	46
4.7	The Software Work Breakdown Structure (WBS)	47
4.8	Software Maintenance	54
4.9	Questions	55

## **Chapter 5 The Basic COCOMO Model 57**

5.1	Introduction	57
5.2	Definitions and Assumptions	58
5.3	Development Effort and Schedule	61
5.4	Phase Distribution	64
5.5	Nominal Project Profiles	65
5.6	The Rayleigh Distribution	67
5.7	Interpolation	69
5.8	Basic Software Maintenance Effort Estimation	71
5.9	Questions	71

## **Chapter 6 The Basic COCOMO Model: Development Modes 74**

6.1	Introduction	74
6.2	Basic Effort and Schedule Equations	75
6.3	The Three COCOMO Modes of Software Development	78
6.4	Discussion of the Basic COCOMO Effort and Schedule Equations	83
6.5	Phase Distribution of Effort and Schedule	89
6.6	Questions	94

## **Chapter 7 The Basic COCOMO Model: Activity Distribution 97**

7.1	Introduction	97
7.2	Activity Distribution by Phase	98
7.3	Basic COCOMO Case Study: The Hunt National Bank EFT System	103
7.4	Deriving Basic Project Organization Charts	104
7.5	Discussion of Basic COCOMO Phase and Activity Distributions	110
7.6	Limitations of Basic COCOMO	111
7.7	Questions	111



**Chapter 8 The Intermediate COCOMO Model: Product Level Estimates 114**

- 8.1 Introduction 114
- 8.2 Intermediate COCOMO: Software Development Effort Estimation 117
- 8.3 A Pricing Example: Microprocessor Communications Software 125
- 8.4 A Management Example: Reduced Cost-to-Complete 127
- 8.5 Adjusted Estimate of Annual Maintenance Effort 129
- 8.6 Example: Microprocessor Communications Software Maintenance 130
- 8.7 Interpolation and Extrapolation 132
- 8.8 Estimating the Effects of Adapting Existing Software 133
- 8.9 Discussion of the Intermediate COCOMO Effort Equations 138
- 8.10 Questions 141

**Chapter 9 Intermediate COCOMO: Component Level Estimation 145**

- 9.1 Introduction 145
- 9.2 The Component Level Estimating Form (CLEF) 146
- 9.3 Using the CLEF with Adapted Software 151
- 9.4 Transaction Processing System (TPS) Example: Basic Development Estimate 153
- 9.5 TPS Component Level Maintenance Estimate and Phase Distribution 156
- 9.6 Questions 160

**PART III FUNDAMENTALS OF SOFTWARE ENGINEERING ECONOMICS 165**

**PART IIIA COST-EFFECTIVENESS ANALYSIS 169**

**Chapter 10 Performance Models and Cost-Effectiveness Models 170**

- 10.1 Performance Models 170
- 10.2 Optimal Performance 173
- 10.3 Sensitivity Analysis 176
- 10.4 Cost-Effectiveness Models 178
- 10.5 Questions 181