



SOFTWARE ENGINEERING
A PRACTITIONER'S APPROACH
THIRD EDITION

ROGER S. PRESSMAN

SOFTWARE ENGINEERING

A PRACTITIONER'S APPROACH

THIRD EDITION

Roger S. Pressman, Ph.D.

McGraw-Hill, Inc.

New York St. Louis San Francisco Auckland Bogotá
Caracas Lisbon London Madrid Mexico City Milan
Montreal New Delhi San Juan Singapore
Sydney Tokyo Toronto

To my parents

SOFTWARE ENGINEERING **A PRACTITIONER'S APPROACH**

Copyright © 1992, 1987, 1982 by McGraw-Hill, Inc. All rights reserved. Printed in the United States of America. Except as permitted under the United States Copyright Act of 1976, no part of this publication may be reproduced or distributed in any form or by any means, or stored in a data base or retrieval system, without the prior written permission of the publisher.

90 DOC/DOC 99876

ISBN 0-07-050814-3

This book is printed on acid-free paper.

This book was set in New Century Schoolbook
by Beacon Graphics Corporation.

The editors were Eric M. Munson and Bernadette Boylan;
the designer was Jo Jones.

New drawings were done by Fine Line Illustrations, Inc.

Cover painting by Joseph Gillians.

R. R. Donnelley & Sons Company was printer and binder.

Library of Congress Cataloging-in-Publication Data

Pressman, Roger S.

Software engineering: a practitioner's approach / by Roger S.
Pressman

p. cm.

Includes bibliographical references and index.

ISBN 0-07-050814-3

1. Software engineering. I. Title.

QA76.758.P75 1992

91-11321

005.1—dc20

Roger S. Pressman is an internationally recognized consultant and author in software engineering. He received a B.S.E. (cum laude) from the University of Connecticut, an M.S. from the University of Bridgeport, and a Ph.D. in engineering from the University of Connecticut, and has over two decades of industry experience, holding both technical and management positions with responsibility for the development of software for engineered products and systems.

As an industry practitioner and manager, Dr. Pressman worked on the development of CAD/CAM systems for advanced engineering and manufacturing in aerospace applications. He has also held positions with responsibility for scientific and systems programming.

In addition to his industry experience, Dr. Pressman was Bullard Associate Professor of Computer Engineering at the University of Bridgeport and Director of the University's Computer-Aided Design and Manufacturing Center. His research interests included software engineering methods and tools.

Dr. Pressman is President of R.S. Pressman & Associates, Inc., a consulting firm specializing in software engineering methods and training. He serves as principal consultant, specializing in helping companies establish effective software engineering practices. In addition to consulting services rendered to many Fortune 500 clients, the company markets a wide variety of software engineering training products and services. Dr. Pressman has authored three videotape training series, *Software Engineering Training Curriculum*, *A CASE Curriculum*, and *Essential Software Engineering*, that are distributed worldwide.

Dr. Pressman has written many technical papers, is a regular contributor to industry newsletters, and is the author of five books. In addition to *Software Engineering: A Practitioner's Approach*, he has written *Making Software Engineering Happen* (Prentice-Hall), a book that addresses the management problems associated with implementing software engineering technology, *Software Engineering: A Beginner's Guide* (McGraw-Hill), an introductory text, and *Software Shock* (Dorset House), a book that focuses on software and its impact on business and society. Dr. Pressman is a member of the ACM, IEEE and Tau Beta Pi, Phi Kappa Phi, Pi Tau Sigma, and Eta Kappa Nu.

Over the past two decades, software engineering has come of age. Today, it is recognized as a legitimate discipline, one worthy of serious research, conscientious study, and tumultuous debate. Throughout the industry, “software engineer” has replaced “programmer” as the job title of preference. Software engineering methods, procedures, and tools have been adopted successfully across a broad spectrum of industry applications. Managers and practitioners alike recognize the need for a more disciplined approach to software development.

But the problems discussed in the first and second editions of this book remain with us. Many individuals and companies still develop software haphazardly. Many professionals and students are unaware of modern methods. And as a result, the quality of the software that we produce suffers. In addition, debate and controversy about the true nature of the software engineering approach continue. The status of software engineering is a study in contrasts. Attitudes have changed, progress has been made, but much remains to be done before the discipline reaches full maturity.

The third edition of *Software Engineering: A Practitioner’s Approach* is intended to provide one element of a foundation from which a bridge from adolescence to maturity can be constructed. The third edition, like the two editions that have preceded it, is intended for both students and practitioners, and maintains the same format and style as its predecessors. The book retains its appeal as a guide to the industry professional and a comprehensive introduction to the student at the upper-level undergraduate or first-year graduate level.

Like in the earlier editions, software engineering methods are presented in the chronological sequence that they are applied during software development. However, the third edition is more than a simple update. The book has been restructured to accommodate the dramatic growth in the field and to emphasize new and important software engineering methods and tools. Rather than maintaining a strict life-cycle view, this edition presents generic activities that are performed regardless of the software engineering paradigm that has been chosen.

Chapters that have been retained from earlier editions have been revised and updated to reflect current trends and techniques. Major new sections have been added to chapters on computer system engineering, requirements analysis fundamentals, data flow-oriented design, object-oriented design, real-time design, software quality assurance, software testing techniques, and maintenance. In addition to these revisions, eight new chapters have been added to the third edition.

The original chapter on software project management has been removed and replaced by three new chapters on software metrics, estimation, and project planning. A new chapter on structured analysis presents the notation and approach for both conventional and real-time applications. A chapter on object-oriented analysis and data modeling provides a detailed treatment of these new and important modeling techniques.

The five existing chapters on software design have been further bolstered with a new chapter on user interface design. Software configuration management—a topic that has become pivotal to successful software development—is now treated in a separate chapter. The role of automation in software engineering is considered in two new chapters on computer-aided software engineering (CASE). One chapter emphasizes software tools and their application and the other discusses integrated CASE environments and the repository. The final chapter (also new) looks toward the twenty-first century and examines changes that will affect our approach to software engineering.

Many new examples, problems, and points to ponder have been added, and the “Further Readings” sections (one of the more popular tidbits in earlier editions) have been expanded and updated for every chapter.

The 24 chapters of the third edition have been divided into five parts. This has been done to compartmentalize topics and assist instructors who may not have the time to complete the entire book in one term. Part I—“Software—the Process and Its Management”—presents a thorough treatment of software project management issues. Part II—“System and Software Requirements Analysis”—contains five chapters that cover analysis fundamentals and requirements modeling methods and notation. Part III—“The Design and Implementation of Software”—presents a thorough treatment of software design, emphasizing fundamental design criteria that lead to high-quality systems and design methods that translate an analysis model into a software solution. Part IV—“Ensuring, Verifying, and Maintaining Software Integrity”—emphasizes the activities that are applied to

ensure quality throughout the software engineering process. Part V—“The Role of Automation”—discusses the impact of CASE on the software development process.

The five-part organization of the third edition enables an instructor to “cluster” topics based on available time and student need. An entire one-term course can be built around one or more of the five parts. For example, a “design course” might emphasize only part III; a “methods course” might present selected chapters in parts II, III, IV, and V; and a “management course” would stress parts I and IV. By organizing the third edition in this way, I have attempted to provide an instructor with a number of teaching options.

An *Instructor’s Guide* for the third edition of *Software Engineering: A Practitioner’s Approach* is available from McGraw-Hill. The *Instructor’s Guide* presents suggestions for conducting various types of software engineering courses, recommendations for a variety of software projects to be conducted in conjunction with a course, solutions to selected problems, and reference to a variety of complementary teaching materials that form a “system” for teaching software engineering.

The software engineering literature continues to expand at an explosive rate. Once again, my thanks to the many authors of books, papers, and articles who have provided me with additional insight, ideas, and commentary over the past decade. Many have been referenced within the pages of each chapter. All deserve credit for their contribution to this rapidly evolving field. I also wish to thank the reviewers of the third edition, James Cross, Auburn University; Mahesh Dodani, University of Iowa; William S. Junk, University of Idaho; and Laurie Werth, University of Texas. Their comments and criticism have been invaluable.

The content of the third edition of *Software Engineering: A Practitioner’s Approach* has been shaped by hundreds of industry professionals, university professors, and students who have used the first and second editions of the book and have taken the time to communicate their suggestions, criticisms, and ideas. In addition, my personal thanks go to our many industry clients throughout North America and Europe, who certainly teach me as much or more than I can teach them.

Finally, to Barbara, Mathew, and Michael, my love and thanks for tolerating my travel schedule, understanding the evenings at the office, and encouraging still another edition of “the book.”

Roger S. Pressman

C O N T E N T S

Preface

xix

PART ONE SOFTWARE—THE PROCESS AND ITS MANAGEMENT

CHAPTER 1: SOFTWARE AND SOFTWARE ENGINEERING	3
1.1 THE IMPORTANCE OF SOFTWARE	4
1.1.1 The Evolving Role of Software	4
1.1.2 An Industry Perspective	7
1.1.3 An Aging Software Plant	8
1.2 SOFTWARE	10
1.2.1 Software Characteristics	10
1.2.2 Software Components	13
1.2.3 Software Applications	15
1.3 SOFTWARE: A CRISIS ON THE HORIZON	17
1.3.1 Problems	18
1.3.2 Causes	19
1.4 SOFTWARE MYTHS	20
1.5 SOFTWARE ENGINEERING PARADIGMS	22
1.5.1 Software Engineering: A Definition	23
1.5.2 The Classic Life Cycle	24
1.5.3 Prototyping	26
1.5.4 The Spiral Model	29
1.5.5 Fourth-Generation Techniques	30
1.5.6 Combining Paradigms	33
1.6 A GENERIC VIEW OF SOFTWARE ENGINEERING	34

1.7 SUMMARY	36
REFERENCES	37
PROBLEMS AND POINTS TO PONDER	37
FURTHER READINGS	38
 CHAPTER 2: PROJECT MANAGEMENT: SOFTWARE METRICS	 41
2.1 THE PROJECT MANAGEMENT PROCESS	42
2.1.1 Beginning a Software Project	42
2.1.2 Measures and Metrics	43
2.1.3 Estimation	43
2.1.4 Risk Analysis	44
2.1.5 Scheduling	44
2.1.6 Tracking and Control	44
2.2 METRICS FOR SOFTWARE PRODUCTIVITY AND QUALITY	45
2.3 MEASURING SOFTWARE	45
2.3.1 Size-Oriented Metrics	47
2.3.2 Function-Oriented Metrics	48
2.4 METRICS FOR SOFTWARE QUALITY	51
2.4.1 An Overview of Factors that Affect Quality	52
2.4.2 Measuring Quality	52
2.5 RECONCILING DIFFERENT METRICS APPROACHES	54
2.6 INTEGRATING METRICS WITHIN THE SOFTWARE ENGINEERING PROCESS	56
2.6.1 Arguments for Software Metrics	56
2.6.2 Establishing a Baseline	57
2.6.3 Metrics Collection, Computation, and Evaluation	58
2.7 SUMMARY	61
REFERENCES	61
PROBLEMS AND POINTS TO PONDER	62
FURTHER READINGS	63
 CHAPTER 3: PROJECT MANAGEMENT: ESTIMATION	 65
3.1 OBSERVATIONS ON ESTIMATING	65
3.2 PROJECT PLANNING OBJECTIVES	67
3.3 SOFTWARE SCOPE	67
3.4 RESOURCES	70
3.4.1 Human Resources	71
3.4.2 Hardware Resources	71
3.4.3 Software Resources	71
3.4.4 Reusability	74
3.5 SOFTWARE PROJECT ESTIMATION	75
3.6 DECOMPOSITION TECHNIQUES	76
3.6.1 LOC and FP Estimation	76
3.6.2 An Example	78
3.6.3 Effort Estimation	81
3.6.4 An Example	82

3.7	EMPIRICAL ESTIMATION MODELS	83
3.7.1	COCOMO	84
3.7.2	Putnam Estimation Model	87
3.7.3	Function-Point Models	89
3.8	AUTOMATED ESTIMATION TOOLS	89
3.9	SUMMARY	91
	REFERENCES	92
	PROBLEMS AND POINTS TO PONDER	93
	FURTHER READINGS	94
CHAPTER 4: PROJECT MANAGEMENT: PLANNING		95
4.1	PROJECT PLANNING—REVISITED	96
4.2	RISK ANALYSIS	96
4.2.1	Risk Identification	97
4.2.2	Risk Projection	98
4.2.3	Risk Assessment	99
4.2.4	Risk Management and Monitoring	101
4.3	SOFTWARE PROJECT SCHEDULING	102
4.3.1	People-Work Relationships	104
4.3.2	Task Definition and Parallelism	105
4.3.3	Effort Distribution	106
4.3.4	Scheduling Methods	107
4.3.5	A Scheduling Example	108
4.3.6	Project Tracking and Control	114
4.4	SOFTWARE ACQUISITION	117
4.5	SOFTWARE RE-ENGINEERING	119
4.6	ORGANIZATIONAL PLANNING	120
4.7	THE SOFTWARE PROJECT PLAN	122
4.8	SUMMARY	124
	REFERENCES	124
	PROBLEMS AND POINTS TO PONDER	125
	FURTHER READINGS	127

PART TWO SYSTEM AND SOFTWARE REQUIREMENTS ANALYSIS

CHAPTER 5: COMPUTER SYSTEM ENGINEERING		131
5.1	COMPUTER-BASED SYSTEMS	132
5.2	COMPUTER SYSTEMS ENGINEERING	134
5.2.1	Hardware and Hardware Engineering	137
5.2.2	Software and Software Engineering	140
5.2.3	Human Factors and Human Engineering	144
5.2.4	Databases and Database Engineering	145
5.3	SYSTEM ANALYSIS	146
5.3.1	Identification of Need	147
5.3.2	Feasibility Study	148
5.3.3	Economic Analysis	149

5.3.4	Technical Analysis	154
5.3.5	Allocation and Trade-offs	156
5.4	MODELING THE SYSTEM ARCHITECTURE	159
5.4.1	Architecture Diagrams	159
5.4.2	Specification of the System Architecture	162
5.5	SYSTEM MODELING AND SIMULATION	164
5.6	SYSTEM SPECIFICATION	165
5.7	SYSTEM SPECIFICATION REVIEW	166
5.8	SUMMARY	168
	REFERENCES	168
	PROBLEMS AND POINTS TO PONDER	169
	FURTHER READINGS	170
CHAPTER 6: REQUIREMENTS ANALYSIS FUNDAMENTALS		173
6.1	REQUIREMENTS ANALYSIS	174
6.1.1	Analysis Tasks	174
6.1.2	The Analyst	176
6.2	PROBLEM AREAS	177
6.3	COMMUNICATION TECHNIQUES	178
6.3.1	Initiating the Process	179
6.3.2	Facilitated Application Specification Techniques	180
6.4	ANALYSIS PRINCIPLES	184
6.4.1	The Information Domain	185
6.4.2	Modeling	186
6.4.3	Partitioning	187
6.4.4	Essential and Implementation Views	189
6.5	SOFTWARE PROTOTYPING	190
6.5.1	A Prototyping Scenario	190
6.5.2	Prototyping Methods and Tools	192
6.6	SPECIFICATION	194
6.6.1	Specification Principles	194
6.6.2	Representation	197
6.6.3	The Software Requirements Specification	198
6.7	SPECIFICATION REVIEW	200
6.8	SUMMARY	202
	REFERENCES	203
	PROBLEMS AND POINTS TO PONDER	203
	FURTHER READINGS	204
CHAPTER 7: STRUCTURED ANALYSIS AND ITS EXTENSIONS		207
7.1	A BRIEF HISTORY	208
7.2	BASIC NOTATION AND ITS EXTENSIONS	208
7.2.1	Data Flow Diagrams	209
7.2.2	Extensions for Real-Time Systems	212
7.2.3	Ward and Mellor Extensions	212
7.2.4	Hatley and Pirbhai Extensions	215
7.2.5	Behavioral Modeling	218
7.2.6	Extensions for Data-Intensive Applications	220

7.3	THE MECHANICS OF STRUCTURED ANALYSIS	221
7.3.1	Creating a Data Flow Model	221
7.3.2	Creating a Control Flow Model	224
7.3.3	The Control Specification	227
7.3.4	The Process Specification	228
7.4	THE REQUIREMENTS DICTIONARY	229
7.5	STRUCTURED ANALYSIS AND CASE	234
7.6	SUMMARY	234
	REFERENCES	235
	PROBLEMS AND POINTS TO PONDER	235
	FURTHER READINGS	237
CHAPTER 8: OBJECT-ORIENTED ANALYSIS AND DATA MODELING		239
8.1	OBJECT-ORIENTED CONCEPTS	240
8.1.1	Identifying Objects	242
8.1.2	Specifying Attributes	245
8.1.3	Defining Operations	247
8.1.4	Interobject Communication	247
8.1.5	Finalizing the Object Definition	249
8.2	OBJECT-ORIENTED ANALYSIS MODELING	250
8.2.1	Classification and Assembly Structures	250
8.2.2	Defining Subjects	252
8.2.3	Instance Connections and Message Paths	254
8.2.4	OOA and Prototyping	254
8.3	DATA MODELING	256
8.3.1	Data Objects, Attributes, and Relationships	257
8.3.2	Entity-Relationship Diagrams	260
8.4	SUMMARY	262
	REFERENCES	263
	PROBLEMS AND POINTS TO PONDER	263
	FURTHER READINGS	264
CHAPTER 9: ALTERNATIVE ANALYSIS TECHNIQUES AND FORMAL METHODS		267
9.1	REQUIREMENTS ANALYSIS METHODS	268
9.1.1	Common Characteristics	268
9.1.2	Differences in Analysis Methods	269
9.2	DATA STRUCTURE-ORIENTED METHODS	270
9.3	DATA STRUCTURED SYSTEMS DEVELOPMENT	270
9.3.1	Warnier Diagrams	270
9.3.2	The DSSD Approach	272
9.3.3	Application Context	273
9.3.4	Application Functions	273
9.3.5	Application Results	276
9.4	JACKSON SYSTEM DEVELOPMENT	277
9.4.1	The Entity Action Step	278
9.4.2	The Entity Structure Step	279
9.4.3	The Initial Model Step	280

9.5	SADT	283
9.6	FORMAL SPECIFICATION TECHNIQUES	287
9.6.1	Current Status of Formal Methods	287
9.6.2	The Attributes of Formal Specification Languages	288
9.7	A FORMAL SPECIFICATION IN Z	289
9.7.1	About the Kernel	290
9.7.2	Documentation	291
9.7.3	Kernel States	292
9.7.4	Background Processing	297
9.7.5	Interrupt Handling	300
9.7.6	Formal Methods—The Road Ahead	303
9.8	AUTOMATED TECHNIQUES FOR REQUIREMENTS ANALYSIS	304
9.8.1	Software Requirements Engineering Methodology	305
9.8.2	PSL/PSA	306
9.8.3	TAGS	306
9.8.4	Specification Environments	307
9.8.5	Tools for Formal Methods	307
9.8.6	Automated Techniques—A Summary	308
9.9	SUMMARY	309
	REFERENCES	309
	PROBLEMS AND POINTS TO PONDER	311
	FURTHER READINGS	311

PART THREE THE DESIGN AND IMPLEMENTATION OF SOFTWARE

CHAPTER 10:	SOFTWARE DESIGN FUNDAMENTALS	315
10.1	SOFTWARE DESIGN AND SOFTWARE ENGINEERING	316
10.2	THE DESIGN PROCESS	317
10.2.1	Design and Software Quality	318
10.2.2	The Evolution of Software Design	319
10.3	DESIGN FUNDAMENTALS	319
10.3.1	Abstraction	320
10.3.2	Refinement	323
10.3.3	Modularity	323
10.3.4	Software Architecture	325
10.3.5	Control Hierarchy	326
10.3.6	Data Structure	328
10.3.7	Software Procedure	330
10.3.8	Information Hiding	331
10.4	EFFECTIVE MODULAR DESIGN	332
10.4.1	Module Types	332
10.4.2	Functional Independence	333
10.4.3	Cohesion	334
10.4.4	Coupling	336
10.5	DATA DESIGN	338
10.6	ARCHITECTURAL DESIGN	340

10.7	PROCEDURAL DESIGN	341
10.7.1	Structured Programming	341
10.7.2	Graphical Design Notation	342
10.7.3	Tabular Design Notation	347
10.7.4	Program Design Language	349
10.7.5	A PDL Example	355
10.7.6	Comparison of Design Notation	357
10.8	DESIGN DOCUMENTATION	359
10.9	SUMMARY	362
	REFERENCES	362
	PROBLEMS AND POINTS TO PONDER	364
	FURTHER READINGS	365
CHAPTER 11: DATA FLOW-ORIENTED DESIGN		367
11.1	DESIGN AND INFORMATION FLOW	367
11.1.1	Contributors	368
11.1.2	Areas of Application	368
11.2	DESIGN PROCESS CONSIDERATIONS	369
11.2.1	Transform Flow	369
11.2.2	Transaction Flow	370
11.2.3	A Process Abstract	371
11.3	TRANSFORM ANALYSIS	372
11.3.1	An Example	372
11.3.2	Design Steps	373
11.4	TRANSACTION ANALYSIS	382
11.4.1	An Example	382
11.4.2	Design Steps	383
11.5	DESIGN HEURISTICS	387
11.6	DESIGN POSTPROCESSING	389
11.7	DESIGN OPTIMIZATION	390
11.8	SUMMARY	391
	REFERENCES	391
	PROBLEMS AND POINTS TO PONDER	392
	FURTHER READINGS	394
CHAPTER 12: OBJECT-ORIENTED DESIGN		395
12.1	ORIGINS OF OBJECT-ORIENTED DESIGN	396
12.2	OBJECT-ORIENTED DESIGN CONCEPTS	397
12.2.1	<i>Objects, Operations, and Messages</i>	397
12.2.2	Design Issues	398
12.2.3	Classes, Instances, and Inheritance	399
12.2.4	Object Descriptions	401
12.3	OBJECT-ORIENTED DESIGN METHODS	402
12.4	CLASS AND OBJECT DEFINITION	404
12.5	REFINING OPERATIONS	407
12.6	PROGRAM COMPONENTS AND INTERFACES	408

12.7	A NOTATION FOR OOD	410
12.7.1	Representing Class and Object Relationships	410
12.7.2	Modularizing the Design	411
12.8	IMPLEMENTATION DETAIL DESIGN	414
12.9	AN ALTERNATIVE OBJECT-ORIENTED DESIGN STRATEGY	416
12.9.1	Design Steps	416
12.9.2	A Design Example	418
12.10	INTEGRATING OOD WITH SA/SD	422
12.11	SUMMARY	424
	REFERENCES	425
	PROBLEMS AND POINTS TO PONDER	426
	FURTHER READINGS	427
CHAPTER 13: DATA-ORIENTED DESIGN METHODS		429
13.1	DESIGN AND DATA STRUCTURE	429
13.1.1	Contributors	430
13.1.2	Areas of Application	431
13.1.3	Data Structure versus Data Flow Techniques	431
13.1.4	Data Structure versus Object-Oriented Design	431
13.2	DESIGN PROCESS CONSIDERATIONS	432
13.3	JACKSON SYSTEM DEVELOPMENT	432
13.3.1	JSD Design Steps	434
13.3.2	The Function Step	437
13.3.3	System Timing Step	442
13.3.4	The Implementation Step	443
13.3.5	Procedural Representation	445
13.4	DATA STRUCTURED SYSTEMS DEVELOPMENT	446
13.4.1	A Simplified Design Approach	447
13.4.2	Derivation of Logical Output Structure	448
13.4.3	Derivation of Logical Process Structure	449
13.4.4	Complex Process Logic	451
13.5	SUMMARY	453
	REFERENCES	454
	PROBLEMS AND POINTS TO PONDER	455
	FURTHER READINGS	456
CHAPTER 14: USER INTERFACE DESIGN		457
14.1	HUMAN FACTORS	458
14.1.1	Fundamentals of Human Perception	458
14.1.2	Human Skill Level and Behavior	459
14.1.3	Tasks and Human Factors	460
14.2	STYLES OF HUMAN-COMPUTER INTERACTION	461
14.3	HUMAN-COMPUTER INTERFACE DESIGN	463
14.3.1	Interface Design Models	464
14.3.2	Task Analysis and Modeling	465
14.3.3	Design Issues	467
14.3.4	Implementation Tools	470
14.3.5	Design Evaluation	471

14.4	INTERFACE DESIGN GUIDELINES	473
14.4.1	General Interaction	473
14.4.2	Information Display	474
14.4.3	Data Input	475
14.5	INTERFACE STANDARDS	476
14.6	SUMMARY	477
	REFERENCES	477
	PROBLEMS AND POINTS TO PONDER	478
	FURTHER READINGS	479
CHAPTER 15: REAL-TIME DESIGN		481
15.1	SYSTEM CONSIDERATIONS	482
15.2	REAL-TIME SYSTEMS	482
15.2.1	Integration and Performance Issues	483
15.2.2	Interrupt Handling	484
15.2.3	Real-Time Databases	486
15.2.4	Real-Time Operating Systems	486
15.2.5	Real-Time Languages	488
15.2.6	Task Synchronization and Communication	488
15.3	ANALYSIS AND SIMULATION OF REAL-TIME SYSTEMS	489
15.3.1	Mathematical Tools for Real-Time System Analysis	490
15.3.2	Simulation and Modeling Techniques for Real-Time Systems	494
15.4	DESIGN METHODS	500
15.5	A DATA FLOW-ORIENTED DESIGN METHOD	500
15.5.1	Requirements of a Real-Time Systems Design Method	501
15.5.2	DARTS	502
15.5.3	Task Design	504
15.5.4	Example of the DARTS Design Method	504
15.6	SUMMARY	509
	REFERENCES	509
	PROBLEMS AND POINTS TO PONDER	510
	FURTHER READINGS	511
CHAPTER 16: PROGRAMMING LANGUAGES AND CODING		513
16.1	THE TRANSLATION PROCESS	514
16.2	PROGRAMMING LANGUAGE CHARACTERISTICS	514
16.2.1	A Psychological View	515
16.2.2	A Syntactic/Semantic Model	517
16.2.3	An Engineering View	518
16.2.4	Choosing a Language	519
16.2.5	Programming Languages and Software Engineering	521
16.3	PROGRAMMING LANGUAGE FUNDAMENTALS	522
16.3.1	Data Types and Data Typing	523
16.3.2	Subprograms	524
16.3.3	Control Structures	524
16.3.4	Support for Object-Oriented Approaches	525