

Fundamentals of Software Engineering

Third Edition

软件工程基础 (第3版)



Rajib Mall 著

清华大学出版社

大学计算机教育国外著名教材系列（影印版）

Fundamentals of Software Engineering

Third Edition

软件工程基础

（第3版）

Rajib Mall 著



清华大学出版社
北 京

Rajib Mall

Fundamentals of Software Engineering, Third Edition

EISBN: 978-81-203-3819-7

Copyright © 2011 by PHI Learning Private Limited, New Delhi.

Original language published by The McGraw-Hill Companies, Inc. All Rights reserved. No part of this publication may be reproduced or distributed by any means, or stored in a database or retrieval system, without the prior written permission of the publisher.

Authorized English language edition jointly published by McGraw-Hill Education (Asia) Co. and Tsinghua University Press. This edition is authorized for sale only to the educational and training institutions, and within the territory of the People's Republic of China (excluding Hong Kong, Macao SAR and Taiwan). Unauthorized export of this edition is a violation of the Copyright Act. Violation of this Law is subject to Civil and Criminal Penalties.

本书英文影印版由清华大学出版社和美国麦格劳-希尔教育出版(亚洲)公司合作出版。此版本仅限在中华人民共和国境内(不包括中国香港、澳门特别行政区及中国台湾地区)针对教育及培训机构之销售。未经许可之出口,视为违反著作权法,将受法律之制裁。

未经出版者预先书面许可,不得以任何方式复制或抄袭本书的任何部分。

北京市版权局著作权合同登记号 图字: 01-2011-7646

本书封面贴有 McGraw-Hill 公司防伪标签,无标签者不得销售。

版权所有,侵权必究。侵权举报电话: 010-62782989 13701121933

图书在版编目(CIP)数据

软件工程基础: 第3版 = Fundamentals of Software Engineering, Third Edition: 英文 / (印) 马尔 (Mall, R.)

著. --影印本. --北京: 清华大学出版社, 2012.1

大学计算机教育国外著名教材系列: 影印版

ISBN 978-7-302-27488-9

I. ①软… II. ①马… III. ①软件工程—高等学校—教材—英文 IV. ①TP311.5

中国版本图书馆 CIP 数据核字 (2011) 第 259706 号

责任编辑: 龙敬铭

责任印制: 王秀菊

出版发行: 清华大学出版社

<http://www.tup.com.cn>

社 总 机: 010-62770175

投稿与读者服务: 010-62776969, c-service@tup.tsinghua.edu.cn

质 量 反 馈: 010-62772015, zhiliang@tup.tsinghua.edu.cn

地 址: 北京清华大学学研大厦 A 座

邮 编: 100084

邮 购: 010-62786544

印 刷 者: 清华大学印刷厂

装 订 者: 北京市密云县京文制本装订厂

发 行 者: 全国新华书店

开 本: 185×230 印张: 29

版 次: 2012 年 1 月第 1 版

印 次: 2012 年 1 月第 1 次印刷

印 数: 1~3000

定 价: 49.00 元

产品编号: 044596-01

出版说明

进入 21 世纪, 世界各国的经济、科技以及综合国力的竞争将更加激烈。竞争的中心无疑是对人才的竞争。谁拥有大量高素质的人才, 谁就能在竞争中取得优势。高等教育, 作为培养高素质人才的事业, 必然受到高度重视。目前我国高等教育的教材更新较慢, 为了加快教材的更新频率, 教育部正在大力促进我国高校采用国外原版教材。

清华大学出版社从 1996 年开始, 与国外著名出版公司合作, 影印出版了“大学计算机教育丛书(影印版)”等一系列引进图书, 受到国内读者的欢迎和支持。跨入 21 世纪, 我们本着为我国高等教育教材建设服务的初衷, 在已有的基础上, 进一步扩大选题内容, 改变图书开本尺寸, 一如既往地请有关专家挑选适用于我国高等本科及研究生计算机教育的国外经典教材或著名教材, 组成本套“大学计算机教育国外著名教材系列(影印版)”, 以飨读者。深切期盼读者及时将使用本系列教材的效果和意见反馈给我们。更希望国内专家、教授积极向我们推荐国外计算机教育的优秀教材, 以利我们把“大学计算机教育国外著名教材系列(影印版)”做得更好, 更适合高校师生的需要。

清华大学出版社

PREFACE

The third edition of this book was necessary to trace on the rapid advancements taking place in the field of software engineering. The techniques and practices discussed, are revamped according to the current trends since the first edition was written.

This book has taken its shape while teaching the subject of **Software Engineering** to the undergraduate and postgraduate students at IIT, Kharagpur over the last one-and-a-half decades. The book treats all important topics of software engineering, including the recent advancements made in a coherent framework. At the same time, the book deals with the topics from the perspective of the practising engineers. A large portion of the text is based on my own practical experience gained while working on software development projects in several organizations.

Designed to serve as a textbook for one semester course in software engineering for the undergraduate students, this book can be used as a reference book for postgraduate students as well. As the topics on Halstead's software science, Software reuse and formal specification covers the topics taught at the postgraduate level.

The book is also intended to the students who are familiar with at least one high level programming and one low level programming language. The students expected are to possess basic ideas about operating systems, systems programming, compiler writing, and computer architecture issues. Experience in writing large-sized programs would be very helpful in grasping some of the important concepts discussed in this book.

The emphasis of this book is to illustrate the important concepts through small examples rather than a single large running example. The former approach has been opted as this would help to illustrate several subtle and important concepts through appropriate small examples, as illustrating these concepts through a single running example becomes a difficult task.

The layout of the chapters has been guided by the sequence of activities undertaken during the life of a software product. However, since the project management activity is spread over all phases, it was considered necessary to discuss these as early in the book as possible. Software project management has been discussed in Chapter 3. However, while teaching this book, one should teach the project management topic after Chapter 11, as teachers would be able to give the design assignments to the students early, and they would get sufficient time to complete them.

In the text, I have taken the liberty to use he/his to actually mean both the genders. This has been done only to increase the readability of the writing rather than with intent of any bias.

The PowerPoint slides to teach the book as well as the solutions manual can be obtained either from the publisher or by sending an e-mail to rajib@cse.iitkgp.ernet.in

We welcome readers to send their suggestions or queries.

RAJIB MALL

PREFACE TO THE FIRST EDITION

This book is designed as a textbook on software engineering for undergraduate students in computer science. Software engineering is a fast developing field. While teaching the subject at the Indian Institute of Technology Kharagpur, I felt the need for organizing a textbook that gives a coherent account of all the state-of-the-art topics and at the same time presents these topics from the viewpoint of practising engineers. A portion of the text is, therefore, based on my own practical experience, gained while working on software development projects in several industries.

The book starts with a comprehensive introduction to software engineering, including some important life cycle models. Chapter 2 presents and discusses techniques and concepts of software project management. This chapter encompasses all phases of software development that are considered crucial to the success of software projects. Chapter 3 focuses on requirements analysis and specification. In this chapter, different approaches to formal requirements specification and essential features of algebraic specifications as a formal specification technique are explored. Chapter 4 highlights some important facets of software design. In Chapter 5, the methodology of Structured Analysis/Structured Design (SA/SD) in relation to traditional function-oriented design. Chapter 7 brings out some basic aspects, techniques and methods pertaining to user interface design. Significant progress has been made in this field and it is important for students to know the various issues involved in a good user interface design. Chapter 8 discusses coding and unit testing techniques. Integration and system testing techniques are elaborately described in Chapter 9. These are the main quality control activities. Chapter 10 is, therefore, exclusively devoted to software quality assurance aspects, ISO 9000 and software reliability models, as these are considered necessary to expose students to basic quality concepts as part of a software engineering course. Finally, in Chapter 11, the student has been introduced to general concepts to CASE tools, without going into specifics of any particular CASE tool.

The students using this textbook should be proficient at least in one high level and low level programming language each. They should also possess basic knowledge of operating systems, systems programming, compiler writing, and computer architecture.

The emphasis in this book is to illustrate the important concepts through several small examples rather than a single large running example. The book also contains many exercises

at the end of each chapter aimed at reinforcing the knowledge of principles and techniques of software engineering.

I do hope fervently that the students will find this text both stimulating and useful.

Acknowledgements

Many persons have contributed to make this book a reality. I would especially like to express my appreciation to Prof. L.M. Patnaik for his unstinted support and encouragement. I would also like to thank Prof. Sunil Sarangi, Dean (CEP) for his guidance throughout the preparation of the manuscript. Thanks are also due to Prof. Ajit Pal, the present Head of the Department, and all my colleagues at the Computer Science and Engineering Department of IIT Kharagpur for their helpful comments and suggestions. I express my special thanks to Prof. P.K.J. Mahapatra of IEM Department for his help during the final preparation of the manuscript.

I acknowledge the help and cooperation received from all the staff members of the Computer Science and Engineering Department of IIT Kharagpur.

I would like to acknowledge the financial assistance provided by IIT Kharagpur for the preparation of the manuscript and I wish to thank the numerous B.Tech and M.Tech students whose enthusiastic participation in classroom discussions helped me to present many ideas and concepts, as discussed in this book, with greater clarity.

Finally, I wish to express my sincere thanks to all my family members for their moral support. In particular, I thank my parents, Sanjib, Kanika Bhabhi, Sudip, Shivani, Sonali and Amitabha, my parents-in-law and GUGLOO. I am grateful to my wife Prabina for her constant encouragement.

RAJIB MALL

Contents

<i>Preface</i>	<i>xv</i>
<i>Preface to the First Edition</i>	<i>xvii</i>
<i>List of Figures</i>	<i>xix</i>
1. INTRODUCTION	1–29
1.1 The Software Engineering Discipline—Its Evolution and Impact	3
1.1.1 Evolution of an Art into an Engineering Discipline	3
1.1.2 A Solution to the Software Crisis	4
1.2 Software Development Projects	6
1.2.1 Programs versus Products	6
1.2.2 Types of Software Development Projects	7
1.2.3 Software Projects being Undertaken by Indian Companies	7
1.3 What is Wrong with the Exploratory Style of Software Development? ...	8
1.3.1 Perceived Problem Complexity: An Interpretation Based on Human Cognition Mechanism	9
1.3.2 Principles Deployed by Software Engineering to Overcome Human Cognitive Limitations	12
1.3.3 Why Study Software Engineering?	15
1.4 Emergence of Software Engineering	15
1.4.1 Early Computer Programming	15
1.4.2 High-Level Language Programming	16
1.4.3 Control Flow-Based Design	16
1.4.4 Data Structure-Oriented Design	20
1.4.5 Data Flow-Oriented Design	20
1.4.6 Object-Oriented Design	21
1.4.7 What Next?	22
1.4.8 Other Developments	23
1.5 Notable Changes in Software Development Practices	23
1.6 Computer Systems Engineering	25
<i>Summary</i>	<i>26</i>
<i>Exercises</i>	<i>27</i>

2. SOFTWARE LIFE CYCLE MODELS	30–56
2.1 Why Use a Life Cycle Model?	31
2.1.1 Why Document a Life Cycle Model?	32
2.1.2 Phase Entry and Exit Criteria	33
2.2 Classical Waterfall Model	33
2.2.1 Phases of Classical Waterfall Model	34
2.2.2 Shortcomings of the Classical Waterfall Model	40
2.2.3 Is the Classical Waterfall Model Useful at All?	40
2.3 Iterative Waterfall Model	41
2.3.1 Phase Containment of Errors	42
2.3.2 Shortcomings of the Iterative Waterfall Model	42
2.4 Prototyping Model	43
2.5 Evolutionary Model	45
2.5.1 Life Cycle Activities	46
2.6 Spiral Model	48
2.6.1 Risk Handling in Spiral Model	49
2.6.2 Phases of the Spiral Model	49
2.6.3 Pros and Cons of the Spiral Model	49
2.6.4 Spiral Model as a Meta Model	50
2.7 Comparison of Different Life Cycle Models	50
2.7.1 Selecting an Appropriate Life Cycle Model for a Project	51
<i>Summary</i>	52
<i>Exercises</i>	53
3. SOFTWARE PROJECT MANAGEMENT	57–107
3.1 Responsibilities of a Software Project Manager	57
3.1.1 Job Responsibilities of a Software Project Manager	57
3.1.2 Skills Necessary for Software Project Management	58
3.2 Project Planning	58
3.2.1 The SPMP Document	60
3.3 Metrics for Project Size Estimation	61
3.3.1 Lines of Code (LOC)	61
3.3.2 Function Point Metric	63
3.4 Project Estimation Techniques	66
3.4.1 Empirical Estimation Techniques	66
3.4.2 Heuristic Techniques	66
3.4.3 Analytical Estimation Techniques	67
3.5 Empirical Estimation Techniques	67
3.5.1 Expert Judgement Technique	67
3.5.2 Delphi Cost Estimation	68
3.6 COCOMO—A Heuristic Estimation Technique	68
3.6.1 Basic COCOMO Model	70
3.6.2 Intermediate COCOMO	73
3.6.3 Complete COCOMO	74
3.6.4 COCOMO 2	74

3.7	Halstead's Software Science—An Analytical Technique	76
3.7.1	Length and Vocabulary	78
3.7.2	Program Volume	78
3.7.3	Potential Minimum Volume	78
3.7.4	Effort and Time	79
3.7.5	Length Estimation	79
3.8	Staffing Level Estimation	80
3.8.1	Norden's Work	80
3.8.2	Putnam's Work	81
3.8.3	Effect of Schedule Change on Cost	82
3.8.4	Jensen's Model	83
3.9	Scheduling	83
3.9.1	Work Breakdown Structure	84
3.9.2	Activity Networks and Critical Path Method	85
3.9.3	Gantt Charts	86
3.9.4	PERT Charts	86
3.9.5	Project Monitoring and Control	88
3.10	Organization and Team Structures	89
3.10.1	Organization Structure	89
3.10.2	Team Structure	91
3.11	Staffing	93
3.11.1	Who is a Good Software Engineer?	94
3.12	Risk Management	95
3.12.1	Risk Identification	95
3.12.2	Risk Assessment	96
3.12.3	Risk Containment	97
3.13	Software Configuration Management	98
3.13.1	Necessity of Software Configuration Management	98
3.13.2	Configuration Management Activities	99
3.13.3	Source Code Control System (SCCS) and RCS	101
3.14	Miscellaneous Plans	102
	<i>Summary</i>	102
	<i>Exercises</i>	103
4.	REQUIREMENTS ANALYSIS AND SPECIFICATION	108–148
4.1	Requirements Gathering and Analysis	109
4.1.1	Requirements Gathering	109
4.1.2	Requirements Analysis	112
4.2	Software Requirements Specification (SRS)	114
4.2.1	Characteristics of a Good SRS Document	115
4.2.2	Examples of Bad SRS Documents	117
4.2.3	Important Categories of Customer Requirements	117
4.2.4	Functional Requirements	119
4.2.5	How to Identify the Functional Requirements?	121
4.2.6	How to Document the Functional Requirements?	122
4.2.7	Traceability	124

4.2.8	Organization of the SRS Document	125
4.2.9	Techniques for Representing Complex Logic	129
4.3	Formal System Specification	132
4.3.1	What is a Formal Technique?	132
4.3.2	Model versus Property-Oriented Methods	133
4.3.3	Operational Semantics	133
4.3.4	Merits and Limitations of Formal Methods	134
4.4	Axiomatic Specification	136
4.5	Algebraic Specification	137
4.5.1	Properties of Algebraic Specifications	139
4.5.2	Auxiliary Functions	140
4.5.3	Structured Specification	141
4.5.4	Pros and Cons of Algebraic Specifications	142
4.6	Executable Specification and 4GL	142
	<i>Summary</i>	142
	<i>Exercises</i>	143
5.	SOFTWARE DESIGN	149–169
5.1	Outcome of a Design Process	149
5.1.1	Classification of Design Activities	150
5.1.2	Classification of Design Methodologies	151
5.1.3	Analysis versus Design	151
5.2	How can We Characterize a Good Software Design?	152
5.3	Cohesion and Coupling	155
5.3.1	Coupling	155
5.3.2	Cohesion	156
5.3.3	Functional Independence	156
5.3.4	Classification of Cohesiveness	157
5.3.5	Classification of Coupling	159
5.4	Layered Arrangement of Modules	160
5.5	Approaches to Software Design	162
5.5.1	Function-Oriented Design	162
5.5.2	Object-Oriented Design	163
5.6	Object-Oriented versus Function-Oriented Design Approaches	165
5.6.1	Automated Fire-Alarm System—Customer Requirements	165
5.6.2	Function-Oriented Approach	166
5.6.3	Object-Oriented Approach	166
	<i>Summary</i>	167
	<i>Exercises</i>	168
6.	FUNCTION-ORIENTED SOFTWARE DESIGN	170–216
6.1	Overview of SA/SD Methodology	171
6.2	Structured Analysis	172
6.3	Data Flow Diagrams (DFDs)	172
6.3.1	Primitive Symbols Used for Constructing DFDs	173

6.3.2	Some Important Concepts Associated with Constructing DFD Models	174
6.3.3	Developing the DFD Model of a System	177
6.3.4	Shortcomings of the DFD Model	193
6.4	Extending DFD Technique to Make it Applicable to Real-Time Systems	194
6.5	Structured Design	194
6.5.1	Flow Chart versus Structure Chart	196
6.5.2	Transformation of a DFD Model into Structure Chart	196
6.5.3	Transform Analysis	196
6.5.4	Transaction Analysis	197
6.6	Detailed Design	200
6.7	Design Review	201
	<i>Summary</i>	201
	<i>Exercises</i>	202
7.	OBJECT MODELLING USING UML	217–263
7.1	Overview of Basic Object-Orientation Concepts	217
7.1.1	Basic Mechanisms	218
7.1.2	Key Concepts	227
7.1.3	Related Technical Terms	232
7.1.4	Advantages of OOD	232
7.2	Unified Modelling Language (UML)	233
7.3	UML Diagrams	236
7.4	Use Case Model	238
7.4.1	Representation of Use Cases	240
7.4.2	Why Develop the Use Case Diagram?	243
7.4.3	How to Identify the Use Cases of a System?	244
7.4.4	Essential versus Real Use Case	244
7.4.5	Factoring of Commonality among Use Cases	244
7.4.6	Use Case Packaging	247
7.5	Class Diagrams	248
7.6	Interaction Diagrams	254
7.7	Activity Diagrams	256
7.8	State Chart Diagram	257
7.9	Postscript	258
7.9.1	Package and Deployment Diagrams	258
7.9.2	UML 2.0	259
	<i>Summary</i>	260
	<i>Exercises</i>	260
8.	OBJECT-ORIENTED SOFTWARE DEVELOPMENT	264–299
8.1	Patterns	265
8.1.1	Basic Pattern Concepts	266
8.1.2	Types of Patterns	267
8.1.3	More Pattern Concepts	268

8.2	Some Common Design Patterns	270
8.2.1	Expert	270
8.2.2	Creator	271
8.2.3	Facade Pattern	271
8.2.4	Model View Separation Patterns	272
8.2.5	Observer Pattern	273
8.2.6	Model-View-Controller (MVC) Pattern	274
8.2.7	Publish-Subscribe Pattern	275
8.2.8	Intermediary (or Proxy) Pattern	277
8.3	An Object-Oriented Analysis and Design Methodology	277
8.3.1	The Unified Process	278
8.3.2	Overview of the OOAD Methodology	278
8.3.3	Use Case Model Development	279
8.3.4	Domain Modelling	281
8.3.5	Identification of Entity Objects	284
8.3.6	Booch's Object Identification Method	285
8.4	Interaction Modelling	287
8.4.1	CRC Cards	288
8.5	Applications of the Analysis and Design Process	289
8.6	OOD Goodness Criteria	294
	<i>Summary</i>	296
	<i>Exercises</i>	297
9.	USER INTERFACE DESIGN	300–322
9.1	Characteristics of a Good User Interface	301
9.2	Basic Concepts	303
9.2.1	User Guidance and Online Help	303
9.2.2	Mode-based versus Modeless Interface	304
9.2.3	Graphical User Interface (GUI) vs. Text-based User Interface	304
9.3	Types of User Interfaces	305
9.3.1	Command Language-based Interface	305
9.3.2	Menu-based Interface	306
9.3.3	Direct Manipulation Interfaces	307
9.4	Fundamentals of Component-based GUI Development	308
9.4.1	Window System	309
9.4.2	Types of Widgets	311
9.4.3	An Overview of X-Window/MOTIF	312
9.4.4	X Architecture	313
9.4.5	Visual Programming	314
9.4.6	Size Measurement of a Component-based GUI	315
9.5	A User Interface Design Methodology	315
9.5.1	Implications of Human Cognition Capabilities on User Interface Design	316
9.5.2	A GUI Design Methodology	316
9.5.3	Task and Object Modelling	317

9.5.4	Selecting a Metaphor	318
9.5.5	Interaction Design and Rough Layout	319
9.5.6	User Interface Inspection	319
<i>Summary</i>		320
<i>Exercises</i>		321
10.	CODING AND TESTING	323-369
10.1	Coding	324
10.1.1	Coding Standards and Guidelines	324
10.2	Code Review	326
10.2.1	Code Walkthrough	327
10.2.2	Code Inspection	327
10.2.3	Clean Room Testing	328
10.3	Software Documentation	329
10.3.1	Internal Documentation	329
10.3.2	External Documentation	330
10.3.3	Gunning's Fog Index	330
10.4	Testing	331
10.4.1	Basic Concepts and Terminologies	331
10.4.2	Why Design Test Cases?	332
10.5	Testing in the Large versus Testing in the Small	334
10.6	Unit Testing	334
10.6.1	Driver and Stub Modules	335
10.7	Black-Box Testing	336
10.7.1	Equivalence Class Partitioning	336
10.7.2	Boundary Value Analysis	337
10.7.3	Summary of the Black-Box Test Suite Design Approach	338
10.8	White-Box Testing	338
10.8.1	Basic Concepts	338
10.8.2	Statement Coverage	340
10.8.3	Branch Coverage	341
10.8.4	Condition Coverage	341
10.8.5	Path Coverage	342
10.8.6	McCabe's Cyclomatic Complexity Metric	344
10.8.7	Data Flow-based Testing	347
10.8.8	Mutation Testing	347
10.9	Debugging	348
10.9.1	Debugging Approaches	348
10.9.2	Debugging Guidelines	349
10.10	Program Analysis Tools	349
10.10.1	Static Analysis Tools	349
10.10.2	Dynamic Analysis Tools	350
10.11	Integration Testing	351
10.11.1	Phased versus Incremental Integration Testing	352

10.12	Testing Object-Oriented Programs	353
10.12.1	What is a Suitable Unit for Testing Object-Oriented Programs?	353
10.12.2	Do Various Object-Orientation Concepts Make Testing Easy? ...	354
10.12.3	Why are Traditional Techniques Considered Unsatisfactory for Testing Object-Oriented Programs?	355
10.12.4	Grey-Box Testing of Object-Oriented Programs	355
10.12.5	Integration Testing of Object-Oriented Programs	356
10.13	System Testing	356
10.13.1	Performance Testing	357
10.13.2	Error Seeding	359
10.14	Some General Issues Associated with Testing	360
10.14.1	Test Documentation	360
10.14.2	Regression Testing	360
	<i>Summary</i>	360
	<i>Exercises</i>	361
11.	SOFTWARE RELIABILITY AND QUALITY MANAGEMENT	370–395
11.1	Software Reliability	371
11.1.1	Hardware versus Software Reliability	372
11.1.2	Reliability Metrics	373
11.1.3	Reliability Growth Modelling	375
11.2	Statistical Testing	376
11.3	Software Quality	377
11.4	Software Quality Management System	377
11.4.1	Evolution of Quality Systems	378
11.4.2	Product Metrics versus Process Metrics	379
11.5	ISO 9000	379
11.5.1	What is ISO 9000 Certification?	380
11.5.2	ISO 9000 for Software Industry	380
11.5.3	Why Get ISO 9000 Certification?	381
11.5.4	How to Get ISO 9000 Certification?	381
11.5.5	Summary of ISO 9001 Requirements	382
11.5.6	Salient Features of ISO 9001 Requirements	384
11.5.7	ISO 9000-2000	384
11.5.8	Shortcomings of ISO 9000 Certification	384
11.6	SEI Capability Maturity Model	385
11.6.1	Comparison between ISO 9000 Certification and SEI/CMM	388
11.6.2	Is SEI CMM Applicable to Small Organizations?	388
11.6.3	CMMI	389
11.7	Personal Software Process (PSP)	389
11.8	Six Sigma	390
	<i>Summary</i>	391
	<i>Exercises</i>	392

12. COMPUTER AIDED SOFTWARE ENGINEERING	396–403
12.1 Case and its Scope	396
12.2 Case Environment	396
12.2.1 Benefits of CASE	398
12.3 CASE Support in Software Life Cycle	398
12.3.1 Prototyping Support	398
12.3.2 Structured Analysis and Design	399
12.3.3 Code Generation	399
12.3.4 Test Case Generator	400
12.4 Other Characteristics of CASE Tools	400
12.4.1 Hardware and Environmental Requirements	400
12.4.2 Documentation Support	400
12.4.3 Project Management	401
12.4.4 External Interface	401
12.4.5 Reverse Engineering Support	401
12.4.6 Data Dictionary Interface	401
12.4.7 Tutorial and Help	401
12.5 Towards Second Generation CASE Tool	401
12.6 Architecture of a CASE Environment	402
<i>Summary</i>	403
<i>Exercises</i>	403
13. SOFTWARE MAINTENANCE	404–411
13.1 Characteristics of Software Maintenance	404
13.1.1 Types of Software Maintenance	405
13.1.2 Characteristics of Software Evolution	405
13.1.3 Special Problems Associated with Software Maintenance	406
13.2 Software Reverse Engineering	406
13.3 Software Maintenance Process Models	407
13.4 Estimation of Maintenance Cost	410
<i>Summary</i>	411
<i>Exercises</i>	411
14. SOFTWARE REUSE	412–421
14.1 What can be Reused?	412
14.2 Why Almost no Reuse so Far?	413
14.3 Basic Issues in any Reuse Program	413
14.4 A Reuse Approach	414
14.4.1 Domain Analysis	414
14.4.2 Component Classification	415
14.4.3 Searching	416
14.4.4 Repository Maintenance	416
14.4.5 Reuse without Modifications	417
14.5 Reuse at Organization Level	417
14.5.1 Current State of Reuse	418
<i>Summary</i>	419
<i>Exercises</i>	420