

# **Software Engineering**

## **A Practitioner's Approach**

**Seventh Edition**

An abstract collage artwork featuring a large, textured blue square in the center. To its left is a semi-circular shape made of layered wood and dark material. Below the blue square is a dark blue rectangle, and to its right is a maroon rectangle. The background is a light, textured surface. The entire cover has a warm orange-brown background.

**Roger S. Pressman**

# Software Engineering

**A PRACTITIONER'S APPROACH**

**SEVENTH EDITION**

**Roger S. Pressman, Ph.D.**



**Higher Education**

Boston Burr Ridge, IL Dubuque, IA New York San Francisco St. Louis  
Bangkok Bogotá Caracas Kuala Lumpur Lisbon London Madrid Mexico City  
Milan Montreal New Delhi Santiago Seoul Singapore Sydney Taipei Toronto



# Higher Education

SOFTWARE ENGINEERING: A PRACTITIONER'S APPROACH, SEVENTH EDITION

Published by McGraw-Hill, a business unit of The McGraw-Hill Companies, Inc., 1221 Avenue of the Americas, New York, NY 10020. Copyright © 2010 by The McGraw-Hill Companies, Inc. All rights reserved. Previous editions © 2005, 2001, and 1997. No part of this publication may be reproduced or distributed in any form or by any means, or stored in a database or retrieval system, without the prior written consent of The McGraw-Hill Companies, Inc., including, but not limited to, in any network or other electronic storage or transmission, or broadcast for distance learning.

Some ancillaries, including electronic and print components, may not be available to customers outside the United States.

This book is printed on acid-free paper.

1 2 3 4 5 6 7 8 9 0 DOC/DOC 0 9

ISBN 978-0-07-337597-7

MHID 0-07-337597-7

Global Publisher: *Raghothaman Srinivasan*  
 Director of Development: *Kristine Tibbetts*  
 Senior Marketing Manager: *Curt Reynolds*  
 Senior Managing Editor: *Faye M. Schilling*  
 Lead Production Supervisor: *Sandy Ludovissy*  
 Senior Media Project Manager: *Sandra M. Schnee*  
 Associate Design Coordinator: *Brenda A. Rolwes*  
 Cover Designer: *Studio Montage, St. Louis, Missouri*  
 (USE) Cover Image: © *The Studio Dog/Getty Images*  
 Compositor: *Macmillan Publishing Solutions*  
 Typeface: *8.5/13.5 Leawood*  
 Printer: *R. R. Donnelley Crawfordsville, IN*

## Library of Congress Cataloging-in-Publication Data

Pressman, Roger S.  
 Software engineering : a practitioner's approach / Roger S. Pressman. — 7th ed.  
 p. cm.  
 Includes index.  
 ISBN 978-0-07-337597-7 — ISBN 0-07-337597-7 (hard copy : alk. paper)  
 1. Software engineering. I. Title.  
 QA76.758.P75 2010  
 005.1—dc22

2008048802



# Software Engineering

**A PRACTITIONER'S APPROACH**

***In loving memory of my  
father who lived 94 years  
and taught me, above all,  
that honesty and integrity  
were the best guides for  
my journey through life.***

## ABOUT THE AUTHOR

**R**oger S. Pressman is an internationally recognized authority in software process improvement and software engineering technologies. For almost four decades, he has worked as a software engineer, a manager, a professor, an author, and a consultant, focusing on software engineering issues.

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

After receiving a Ph.D. in engineering from the University of Connecticut, Dr. Pressman moved to academia where he became Bullard Associate Professor of Computer Engineering at the University of Bridgeport and director of the university's Computer-Aided Design and Manufacturing Center.

Dr. Pressman is currently president of R.S. Pressman & Associates, Inc., a consulting firm specializing in software engineering methods and training. He serves as principal consultant and has designed and developed *Essential Software Engineering*, a complete video curriculum in software engineering, and *Process Advisor*, a self-directed system for software process improvement. Both products are used by thousands of companies worldwide. More recently, he has worked in collaboration with *EdistaLearning* in India to develop comprehensive Internet-based training in software engineering.

Dr. Pressman has written many technical papers, is a regular contributor to industry periodicals, and is author of seven technical books. In addition to *Software Engineering: A Practitioner's Approach*, he has co-authored *Web Engineering* (McGraw-Hill), one of the first books to apply a tailored set of software engineering principles and practices to the development of Web-based systems and applications. He has also written the award-winning *A Manager's Guide to Software Engineering* (McGraw-Hill); *Making Software Engineering Happen* (Prentice Hall), the first book to address the critical management problems associated with software process improvement; and *Software Shock* (Dorset House), a treatment that focuses on software and its impact on business and society. Dr. Pressman has been on the editorial boards of a number of industry journals, and for many years, was editor of the "Manager" column in *IEEE Software*.

Dr. Pressman is a well-known speaker, keynoting a number of major industry conferences. He is a member of the IEEE, and Tau Beta Pi, Phi Kappa Phi, Eta Kappa Nu, and Pi Tau Sigma.

On the personal side, Dr. Pressman lives in South Florida with his wife, Barbara. An athlete for most of his life, he remains a serious tennis player (NTRP 4.5) and a single-digit handicap golfer. In his spare time, he has written two novels, *The Aymara Bridge* and *The Puppeteer*, and plans to begin work on another.

**W**hen computer software succeeds—when it meets the needs of the people who use it, when it performs flawlessly over a long period of time, when it is easy to modify and even easier to use—it can and does change things for the better. But when software fails—when its users are dissatisfied, when it is error prone, when it is difficult to change and even harder to use—bad things can and do happen. We all want to build software that makes things better, avoiding the bad things that lurk in the shadow of failed efforts. To succeed, we need discipline when software is designed and built. We need an engineering approach.

It has been almost three decades since the first edition of this book was written. During that time, software engineering has evolved from an obscure idea practiced by a relatively small number of zealots to a legitimate engineering discipline. Today, it is recognized as a subject worthy of serious research, conscientious study, and tumultuous debate. Throughout the industry, software engineer has replaced programmer as the job title of preference. Software process models, software engineering methods, and software tools have been adopted successfully across a broad spectrum of industry segments.

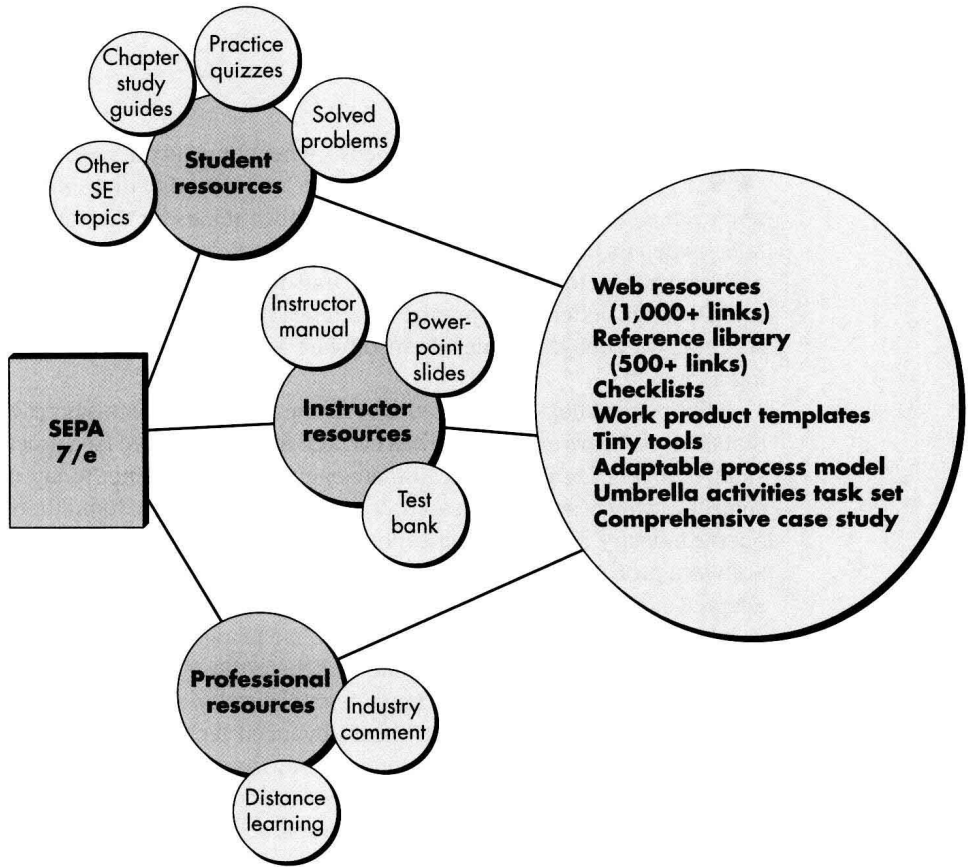
Although managers and practitioners alike recognize the need for a more disciplined approach to software, they continue to debate the manner in which discipline is to be applied. Many individuals and companies still develop software haphazardly, even as they build systems to service today's most advanced technologies. Many professionals and students are unaware of modern methods. And as a result, the quality of the software that we produce suffers, and bad things happen. 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 seventh edition of *Software Engineering: A Practitioner's Approach* is intended to serve as a guide to a maturing engineering discipline. Like the six editions that preceded it, the seventh edition is intended for both students and practitioners, retaining 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.

The seventh edition is considerably more than a simple update. The book has been revised and restructured to improve pedagogical flow and emphasize new and important software engineering processes and practices. In addition, a revised and updated “support system,” illustrated in the figure, provides a comprehensive set of student, instructor, and professional resources to complement the content of the book. These resources are presented as part of a website ([www.mhhe.com/pressman](http://www.mhhe.com/pressman)) specifically designed for *Software Engineering: A Practitioner's Approach*.

**The Seventh Edition.** The 32 chapters of the seventh edition have been reorganized into five parts. This organization, which differs considerably from the sixth edition, has been done to better compartmentalize topics and assist instructors who may not have the time to complete the entire book in one term.

**Support  
System for  
SEPA, 7/e**



Part 1, *The Process*, presents a variety of different views of software process, considering all important process models and addressing the debate between prescriptive and agile process philosophies. Part 2, *Modeling*, presents analysis and design methods with an emphasis on object-oriented techniques and UML modeling. Pattern-based design and design for Web applications are also considered. Part 3, *Quality Management*, presents the concepts, procedures, techniques, and methods that enable a software team to assess software quality, review software engineering work products, conduct SQA procedures, and apply an effective testing strategy and tactics. In addition, formal modeling and verification methods are also considered. Part 4, *Managing Software Projects*, presents topics that are relevant to those who plan, manage, and control a software development project. Part 5, *Advanced Topics*, considers software process improvement and software engineering trends. Continuing in the tradition of past editions, a series of sidebars is used throughout the book to present the trials and tribulations of a (fictional) software team and to provide supplementary materials about methods and tools that are relevant to chapter topics. Two new appendices provide brief tutorials on UML and object-oriented thinking for those who may be unfamiliar with these important topics.



The five-part organization of the seventh 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. A software engineering survey course would select chapters from all five parts. A software engineering course that emphasizes analysis and design would select topics from Parts 1 and 2. A testing-oriented software engineering course would select topics from Parts 1 and 3, with a brief foray into Part 2. A “management course” would stress Parts 1 and 4. By organizing the seventh edition in this way, I have attempted to provide an instructor with a number of teaching options. In every case, the content of the seventh edition is complemented by the following elements of the *SEPA, 7/e Support System*.

**Student Resources.** A wide variety of student resources includes an extensive online learning center encompassing chapter-by-chapter study guides, practice quizzes, problem solutions, and a variety of Web-based resources including software engineering checklists, an evolving collection of “tiny tools,” a comprehensive case study, work product templates, and many other resources. In addition, over 1000 categorized *Web References* allow a student to explore software engineering in greater detail and a *Reference Library* with links to over 500 downloadable papers provides an in-depth source of advanced software engineering information.

**Instructor Resources.** A broad array of instructor resources has been developed to supplement the seventh edition. These include a complete online *Instructor's Guide* (also downloadable) and supplementary teaching materials including a complete set of over 700 *PowerPoint Slides* that may be used for lectures, and a test bank. Of course, all resources available for students (e.g., tiny tools, the Web References, the downloadable Reference Library) and professionals are also available.

The *Instructor's Guide for Software Engineering: A Practitioner's Approach* 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 a number of useful teaching aids.

**Professional Resources.** A collection of resources available to industry practitioners (as well as students and faculty) includes outlines and samples of software engineering documents and other work products, a useful set of software engineering checklists, a catalog of software engineering (CASE) tools, a comprehensive collection of Web-based resources, and an “adaptable process model” that provides a detailed task breakdown of the software engineering process.

When coupled with its online support system, the seventh edition of *Software Engineering: A Practitioner's Approach*, provides flexibility and depth of content that cannot be achieved by a textbook alone.

**Acknowledgments.** My work on the seven editions of *Software Engineering: A Practitioner's Approach* has been the longest continuing technical project of my life. Even when the writing stops, information extracted from the technical literature continues to be assimilated and organized, and criticism and suggestions from readers worldwide is evaluated and cataloged. For this reason, my thanks to the many authors of books, papers, and articles (in both hardcopy and electronic media) who have provided me with additional insight, ideas, and commentary over nearly 30 years.

Special thanks go to Tim Lethbridge of the University of Ottawa, who assisted me in the development of UML and OCL examples and developed the case study that accompanies this book, and Dale Skrien of Colby College, who developed the UML tutorial in

Appendix 1. Their assistance and comments were invaluable. Special thanks also go to Bruce Maxim of the University of Michigan–Dearborn, who assisted me in developing much of the pedagogical website content that accompanies this book. Finally, I wish to thank the reviewers of the seventh edition: Their in-depth comments and thoughtful criticism have been invaluable.

Osman Balci,  
*Virginia Tech University*

Max Fomitchev,  
*Penn State University*

Jerry (Zeyu) Gao,  
*San Jose State University*

Guillermo Garcia,  
*Universidad Alfonso X Madrid*

Pablo Gervas,  
*Universidad Complutense de Madrid*

SK Jain,  
*National Institute of Technology Hamirpur*

Saeed Monemi,  
*Cal Poly Pomona*

Ahmed Salem,  
*California State University*

Vasudeva Varma,  
*IIIT Hyderabad*

The content of the seventh edition of *Software Engineering: A Practitioner's Approach* has been shaped by industry professionals, university professors, and students who have used earlier editions of the book and have taken the time to communicate their suggestions, criticisms, and ideas. My thanks to each of you. In addition, my personal thanks go to our many industry clients worldwide, who certainly have taught me as much or more than I could ever teach them.

As the editions of this book have evolved, my sons, Mathew and Michael, have grown from boys to men. Their maturity, character, and success in the real world have been an inspiration to me. Nothing has filled me with more pride. And finally, to Barbara, my love and thanks for tolerating the many, many hours in the office and encouraging still another edition of “the book.”

*Roger S. Pressman*

# CONTENTS AT A GLANCE

CHAPTER 1      Software and Software Engineering    1

---

## **PART ONE      THE SOFTWARE PROCESS    29**

CHAPTER 2      Process Models    30

CHAPTER 3      Agile Development    65

---

## **PART TWO      MODELING    95**

CHAPTER 4      Principles that Guide Practice    96

CHAPTER 5      Understanding Requirements    119

CHAPTER 6      Requirements Modeling: Scenarios, Information, and Analysis Classes    148

CHAPTER 7      Requirements Modeling: Flow, Behavior, Patterns, and WebApps    186

CHAPTER 8      Design Concepts    215

CHAPTER 9      Architectural Design    242

CHAPTER 10      Component-Level Design    276

CHAPTER 11      User Interface Design    312

CHAPTER 12      Pattern-Based Design    347

CHAPTER 13      WebApp Design    373

---

## **PART THREE      QUALITY MANAGEMENT    397**

CHAPTER 14      Quality Concepts    398

CHAPTER 15      Review Techniques    416

CHAPTER 16      Software Quality Assurance    432

CHAPTER 17      Software Testing Strategies    449

CHAPTER 18      Testing Conventional Applications    481

CHAPTER 19      Testing Object-Oriented Applications    511

CHAPTER 20      Testing Web Applications    529

CHAPTER 21      Formal Modeling and Verification    557

CHAPTER 22      Software Configuration Management    584

CHAPTER 23      Product Metrics    613

---

## **PART FOUR      MANAGING SOFTWARE PROJECTS    645**

CHAPTER 24      Project Management Concepts    646

CHAPTER 25      Process and Project Metrics    666

CHAPTER 26	Estimation for Software Projects	691
CHAPTER 27	Project Scheduling	721
CHAPTER 28	Risk Management	744
CHAPTER 29	Maintenance and Reengineering	761

---

**PART FIVE      **ADVANCED TOPICS**      785**

CHAPTER 30	Software Process Improvement	786
CHAPTER 31	Emerging Trends in Software Engineering	808
CHAPTER 32	Concluding Comments	833
APPENDIX 1	An Introduction to UML	841
APPENDIX 2	Object-Oriented Concepts	863
REFERENCES		871
INDEX		889



# TABLE OF CONTENTS

*Preface* xxv

---

## **CHAPTER 1 SOFTWARE AND SOFTWARE ENGINEERING 1**

---

1.1	The Nature of Software	3
1.1.1	Defining Software	4
1.1.2	Software Application Domains	7
1.1.3	Legacy Software	9
1.2	The Unique Nature of WebApps	10
1.3	Software Engineering	12
1.4	The Software Process	14
1.5	Software Engineering Practice	17
1.5.1	The Essence of Practice	17
1.5.2	General Principles	19
1.6	Software Myths	21
1.7	How It All Starts	24
1.8	Summary	25
	PROBLEMS AND POINTS TO PONDER	25
	FURTHER READINGS AND INFORMATION SOURCES	26

---

## **PART ONE THE SOFTWARE PROCESS 29**

---

### **CHAPTER 2 PROCESS MODELS 30**

---

2.1	A Generic Process Model	31
2.1.1	Defining a Framework Activity	32
2.1.2	Identifying a Task Set	34
2.1.3	Process Patterns	35
2.2	Process Assessment and Improvement	37
2.3	Prescriptive Process Models	38
2.3.1	The Waterfall Model	39
2.3.2	Incremental Process Models	41
2.3.3	Evolutionary Process Models	42
2.3.4	Concurrent Models	48
2.3.5	A Final Word on Evolutionary Processes	49
2.4	Specialized Process Models	50
2.4.1	Component-Based Development	50
2.4.2	The Formal Methods Model	51
2.4.3	Aspect-Oriented Software Development	52
2.5	The Unified Process	53
2.5.1	A Brief History	54
2.5.2	Phases of the Unified Process	54
2.6	Personal and Team Process Models	56
2.6.1	Personal Software Process (PSP)	57
2.6.2	Team Software Process (TSP)	58
2.7	Process Technology	59
2.8	Product and Process	60

2.9	Summary	61
	PROBLEMS AND POINTS TO PONDER	62
	FURTHER READINGS AND INFORMATION SOURCES	63

## **CHAPTER 3      AGILE DEVELOPMENT    65**

---

3.1	What Is Agility?	67
3.2	Agility and the Cost of Change	67
3.3	What Is an Agile Process?	68
3.3.1	Agility Principles	69
3.3.2	The Politics of Agile Development	70
3.3.3	Human Factors	71
3.4	Extreme Programming (XP)	72
3.4.1	XP Values	72
3.4.2	The XP Process	73
3.4.3	Industrial XP	77
3.4.4	The XP Debate	78
3.5	Other Agile Process Models	80
3.5.1	Adaptive Software Development (ASD)	81
3.5.2	Scrum	82
3.5.3	Dynamic Systems Development Method (DSDM)	84
3.5.4	Crystal	85
3.5.5	Feature Driven Development (FDD)	86
3.5.6	Lean Software Development (LSD)	87
3.5.7	Agile Modeling (AM)	88
3.5.8	Agile Unified Process (AUP)	89
3.6	A Tool Set for the Agile Process	91
3.7	Summary	91
	PROBLEMS AND POINTS TO PONDER	92
	FURTHER READINGS AND INFORMATION SOURCES	93

## **PART TWO      MODELING    95**

---

### **CHAPTER 4      PRINCIPLES THAT GUIDE PRACTICE    96**

---

4.1	Software Engineering Knowledge	97
4.2	Core Principles	98
4.2.1	Principles That Guide Process	98
4.2.2	Principles That Guide Practice	99
4.3	Principles That Guide Each Framework Activity	101
4.3.1	Communication Principles	101
4.3.2	Planning Principles	103
4.3.3	Modeling Principles	105
4.3.4	Construction Principles	111
4.3.5	Deployment Principles	113
4.4	Summary	115
	PROBLEMS AND POINTS TO PONDER	116
	FURTHER READINGS AND INFORMATION SOURCES	116

### **CHAPTER 5      UNDERSTANDING REQUIREMENTS    119**

---

5.1	Requirements Engineering	120
5.2	Establishing the Groundwork	125
5.2.1	Identifying Stakeholders	125

5.2.2	Recognizing Multiple Viewpoints	126
5.2.3	Working toward Collaboration	126
5.2.4	Asking the First Questions	127
5.3	Eliciting Requirements	128
5.3.1	Collaborative Requirements Gathering	128
5.3.2	Quality Function Deployment	131
5.3.3	Usage Scenarios	132
5.3.4	Elicitation Work Products	133
5.4	Developing Use Cases	133
5.5	Building the Requirements Model	138
5.5.1	Elements of the Requirements Model	139
5.5.2	Analysis Patterns	142
5.6	Negotiating Requirements	142
5.7	Validating Requirements	144
5.8	Summary	145
	PROBLEMS AND POINTS TO PONDER	145
	FURTHER READINGS AND INFORMATION SOURCES	146

## **CHAPTER 6      REQUIREMENTS MODELING: SCENARIOS, INFORMATION, AND ANALYSIS CLASSES    148**

---

6.1	Requirements Analysis	149
6.1.1	Overall Objectives and Philosophy	150
6.1.2	Analysis Rules of Thumb	151
6.1.3	Domain Analysis	151
6.1.4	Requirements Modeling Approaches	153
6.2	Scenario-Based Modeling	154
6.2.1	Creating a Preliminary Use Case	155
6.2.2	Refining a Preliminary Use Case	158
6.2.3	Writing a Formal Use Case	159
6.3	UML Models That Supplement the Use Case	161
6.3.1	Developing an Activity Diagram	161
6.3.2	Swimlane Diagrams	162
6.4	Data Modeling Concepts	164
6.4.1	Data Objects	164
6.4.2	Data Attributes	164
6.4.3	Relationships	165
6.5	Class-Based Modeling	167
6.5.1	Identifying Analysis Classes	167
6.5.2	Specifying Attributes	171
6.5.3	Defining Operations	171
6.5.4	Class-Responsibility-Collaborator (CRC) Modeling	173
6.5.5	Associations and Dependencies	180
6.5.6	Analysis Packages	182
6.6	Summary	183
	PROBLEMS AND POINTS TO PONDER	183
	FURTHER READINGS AND INFORMATION SOURCES	184

## **CHAPTER 7      REQUIREMENTS MODELING: FLOW, BEHAVIOR, PATTERNS, AND WEBAPPS    186**

---

7.1	Requirements Modeling Strategies	186
7.2	Flow-Oriented Modeling	187

7.2.1	Creating a Data Flow Model	188
7.2.2	Creating a Control Flow Model	191
7.2.3	The Control Specification	191
7.2.4	The Process Specification	192
7.3	Creating a Behavioral Model	195
7.3.1	Identifying Events with the Use Case	195
7.3.2	State Representations	196
7.4	Patterns for Requirements Modeling	199
7.4.1	Discovering Analysis Patterns	200
7.4.2	A Requirements Pattern Example: Actuator-Sensor	200
7.5	Requirements Modeling for WebApps	205
7.5.1	How Much Analysis Is Enough?	205
7.5.2	Requirements Modeling Input	206
7.5.3	Requirements Modeling Output	207
7.5.4	Content Model for WebApps	207
7.5.5	Interaction Model for WebApps	209
7.5.6	Functional Model for WebApps	210
7.5.7	Configuration Models for WebApps	211
7.5.8	Navigation Modeling	212
7.6	Summary	213
	PROBLEMS AND POINTS TO PONDER	213
	FURTHER READINGS AND INFORMATION SOURCES	214

## CHAPTER 8 DESIGN CONCEPTS 215

---

8.1	Design within the Context of Software Engineering	216
8.2	The Design Process	219
8.2.1	Software Quality Guidelines and Attributes	219
8.2.2	The Evolution of Software Design	221
8.3	Design Concepts	222
8.3.1	Abstraction	223
8.3.2	Architecture	223
8.3.3	Patterns	224
8.3.4	Separation of Concerns	225
8.3.5	Modularity	225
8.3.6	Information Hiding	226
8.3.7	Functional Independence	227
8.3.8	Refinement	228
8.3.9	Aspects	228
8.3.10	Refactoring	229
8.3.11	Object-Oriented Design Concepts	230
8.3.12	Design Classes	230
8.4	The Design Model	233
8.4.1	Data Design Elements	234
8.4.2	Architectural Design Elements	234
8.4.3	Interface Design Elements	235
8.4.4	Component-Level Design Elements	237
8.4.5	Deployment-Level Design Elements	237
8.5	Summary	239
	PROBLEMS AND POINTS TO PONDER	240
	FURTHER READINGS AND INFORMATION SOURCES	240



**CHAPTER 9     ARCHITECTURAL DESIGN     242**


---

9.1	Software Architecture	243
9.1.1	What Is Architecture?	243
9.1.2	Why Is Architecture Important?	245
9.1.3	Architectural Descriptions	245
9.1.4	Architectural Decisions	246
9.2	Architectural Genres	246
9.3	Architectural Styles	249
9.3.1	A Brief Taxonomy of Architectural Styles	250
9.3.2	Architectural Patterns	253
9.3.3	Organization and Refinement	255
9.4	Architectural Design	255
9.4.1	Representing the System in Context	256
9.4.2	Defining Archetypes	257
9.4.3	Refining the Architecture into Components	258
9.4.4	Describing Instantiations of the System	260
9.5	Assessing Alternative Architectural Designs	261
9.5.1	An Architecture Trade-Off Analysis Method	262
9.5.2	Architectural Complexity	263
9.5.3	Architectural Description Languages	264
9.6	Architectural Mapping Using Data Flow	265
9.6.1	Transform Mapping	265
9.6.2	Refining the Architectural Design	272
9.7	Summary	273
	PROBLEMS AND POINTS TO PONDER	274
	FURTHER READINGS AND INFORMATION SOURCES	274

**CHAPTER 10     COMPONENT-LEVEL DESIGN     276**


---

10.1	What Is a Component?	277
10.1.1	An Object-Oriented View	277
10.1.2	The Traditional View	279
10.1.3	A Process-Related View	281
10.2	Designing Class-Based Components	282
10.2.1	Basic Design Principles	282
10.2.2	Component-Level Design Guidelines	285
10.2.3	Cohesion	286
10.2.4	Coupling	288
10.3	Conducting Component-Level Design	290
10.4	Component-Level Design for WebApps	296
10.4.1	Content Design at the Component Level	297
10.4.2	Functional Design at the Component Level	297
10.5	Designing Traditional Components	298
10.5.1	Graphical Design Notation	299
10.5.2	Tabular Design Notation	300
10.5.3	Program Design Language	301
10.6	Component-Based Development	303
10.6.1	Domain Engineering	303
10.6.2	Component Qualification, Adaptation, and Composition	304
10.6.3	Analysis and Design for Reuse	306
10.6.4	Classifying and Retrieving Components	307