

Eighth Edition

Software Engineering

A PRACTITIONER'S APPROACH

Roger S.
PRESSMAN

Bruce R.
MAXIM



Software Engineering

A PRACTITIONER'S APPROACH

EIGHTH EDITION

Roger S. Pressman, Ph.D.
Bruce R. Maxim, Ph.D.

**Mc
Graw
Hill**
Education



SOFTWARE ENGINEERING: A PRACTITIONER'S APPROACH, EIGHTH EDITION

Published by McGraw-Hill Education, 2 Penn Plaza, New York, NY 10121. Copyright © 2015 by McGraw-Hill Education. All rights reserved. Printed in the United States of America. Previous editions © 2010, 2005, and 2001. 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 McGraw-Hill Education, 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 1 0 9 8 7 6 5 4

ISBN 978-0-07-802212-8

MHID 0-07-802212-6

Senior Vice President, Products & Markets:

Kurt L. Strand

Vice President, General Manager: *Marty Lange*

Vice President, Content Production & Technology

Services: *Kimberly Meriwether David*

Managing Director: *Thomas Timp*

Publisher: *Raghu Srinivasan*

Developmental Editor: *Vincent Bradshaw*

Marketing Manager: *Heather Wagner*

Director, Content Production: *Terri Schiesl*

Project Manager: *Heather Ervolino*

Buyer: *Sandy Ludovissy*

Cover Designer: *Studio Montage, St. Louis, MO.*

Cover Image: *Farinaz Taghavi/Getty images*

Compositor: *MPS Limited*

Typeface: *8.5/13.5 Impressum Std*

Printer: *R. R. Donnelley*

All credits appearing on page or at the end of the book are considered to be an extension of the copyright page.

Library of Congress Cataloging-in-Publication Data

Pressman, Roger S.

Software engineering : a practitioner's approach / Roger S. Pressman,
Ph.D. — Eighth edition.

pages cm

Includes bibliographical references and index.

ISBN-13: 978-0-07-802212-8 (alk. paper)

ISBN-10: 0-07-802212-6 (alk. paper)

1. Software engineering. I. Title.

QA76.758.P75 2015

005.1—dc23

2013035493

The Internet addresses listed in the text were accurate at the time of publication. The inclusion of a website does not indicate an endorsement by the authors or McGraw-Hill Education, and McGraw-Hill Education does not guarantee the accuracy of the information presented at these sites.

Software Engineering

A PRACTITIONER'S APPROACH

*To my granddaughters
Lily and Maya, who already
understand the importance
of software, even though they're
still in preschool.*

—Roger S. Pressman

*In loving memory of
my parents, who taught
me from an early age that
pursuing a good education
was far more important
than pursuing money.*

—Bruce R. Maxim

ABOUT THE AUTHORS

Roger S. Pressman is an internationally recognized consultant and author in software engineering. For more than four decades, he has worked as a software engineer, a manager, a professor, an author, a consultant, and an entrepreneur.

Dr. Pressman is president of R. S. Pressman & Associates, Inc., a consulting firm that specializes in helping companies establish effective software engineering practices. Over the years he has developed a set of techniques and tools that improve software engineering practice. He is also the founder of Teslaccessories, LLC, a start-up manufacturing company that specializes in custom products for the Tesla Model S electric vehicle.

Dr. Pressman is the author of nine books, including two novels, and many technical and management papers. He has been on the editorial boards of *IEEE Software* and *The Cutter IT Journal* and was editor of the “Manager” column in *IEEE Software*.

Dr. Pressman is a well-known speaker, keynoting a number of major industry conferences. He has presented tutorials at the International Conference on Software Engineering and at many other industry meetings. He has been a member of the ACM, IEEE, and Tau Beta Pi, Phi Kappa Phi, Eta Kappa Nu, and Pi Tau Sigma.

Bruce R. Maxim has worked as a software engineer, project manager, professor, author, and consultant for more than thirty years. His research interests include software engineering, human computer interaction, game design, social media, artificial intelligence, and computer science education.

Dr. Maxim is associate professor of computer and information science at the University of Michigan—Dearborn. He established the GAME Lab in the College of Engineering and Computer Science. He has published a number of papers on computer algorithm animation, game development, and engineering education. He is coauthor of a best-selling introductory computer science text. Dr. Maxim has supervised several hundred industry-based software development projects as part of his work at UM-Dearborn.

Dr. Maxim’s professional experience includes managing research information systems at a medical school, directing instructional computing for a medical campus, and working as a statistical programmer. Dr. Maxim served as the chief technology officer for a game development company.

Dr. Maxim was the recipient of several distinguished teaching awards and a distinguished community service award. He is a member of Sigma Xi, Upsilon Pi Epsilon, Pi Mu Epsilon, Association of Computing Machinery, IEEE Computer Society, American Society for Engineering Education, Society of Women Engineers, and International Game Developers Association.

CONTENTS AT A GLANCE

CHAPTER 1	The Nature of Software	1
CHAPTER 2	Software Engineering	14

PART ONE THE SOFTWARE PROCESS 29

CHAPTER 3	Software Process Structure	30
CHAPTER 4	Process Models	40
CHAPTER 5	Agile Development	66
CHAPTER 6	Human Aspects of Software Engineering	87

PART TWO MODELING 103

CHAPTER 7	Principles That Guide Practice	104
CHAPTER 8	Understanding Requirements	131
CHAPTER 9	Requirements Modeling: Scenario-Based Methods	166
CHAPTER 10	Requirements Modeling: Class-Based Methods	184
CHAPTER 11	Requirements Modeling: Behavior, Patterns, and Web/Mobile Apps	202
CHAPTER 12	Design Concepts	224
CHAPTER 13	Architectural Design	252
CHAPTER 14	Component-Level Design	285
CHAPTER 15	User Interface Design	317
CHAPTER 16	Pattern-Based Design	347
CHAPTER 17	WebApp Design	371
CHAPTER 18	MobileApp Design	391

PART THREE QUALITY MANAGEMENT 411

CHAPTER 19	Quality Concepts	412
CHAPTER 20	Review Techniques	431
CHAPTER 21	Software Quality Assurance	448
CHAPTER 22	Software Testing Strategies	466
CHAPTER 23	Testing Conventional Applications	496
CHAPTER 24	Testing Object-Oriented Applications	523
CHAPTER 25	Testing Web Applications	540
CHAPTER 26	Testing MobileApps	567

CHAPTER 27	Security Engineering	584
CHAPTER 28	Formal Modeling and Verification	601
CHAPTER 29	Software Configuration Management	623
CHAPTER 30	Product Metrics	653

PART FOUR MANAGING SOFTWARE PROJECTS 683

CHAPTER 31	Project Management Concepts	684
CHAPTER 32	Process and Project Metrics	703
CHAPTER 33	Estimation for Software Projects	727
CHAPTER 34	Project Scheduling	754
CHAPTER 35	Risk Management	777
CHAPTER 36	Maintenance and Reengineering	795

PART FIVE ADVANCED TOPICS 817

CHAPTER 37	Software Process Improvement	818
CHAPTER 38	Emerging Trends in Software Engineering	839
CHAPTER 39	Concluding Comments	860
APPENDIX 1	An Introduction to UML	869
APPENDIX 2	Object-Oriented Concepts	891
APPENDIX 3	Formal Methods	899
REFERENCES		909
INDEX		933

When 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 and a half 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 eighth edition of *Software Engineering: A Practitioner's Approach* is intended to serve as a guide to a maturing engineering discipline. The eighth edition, like the seven editions that preceded it, 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 eighth 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, we have further enhanced the popular "support system" for the book, providing 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) specifically designed for *Software Engineering: A Practitioner's Approach*.

The Eighth Edition. The 39 chapters of the eighth edition are organized into five parts. This organization better compartmentalizes topics and assists instructors who may not have the time to complete the entire book in one term.

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 and mobile 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.

The five-part organization of the eighth 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 eighth edition in this way, we have attempted to provide an instructor with a number of teaching options. In every case the content of the eighth edition is complemented by the following elements of the *SEPA, 8/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 1,000 categorized *Web References* allow a student to explore software engineering in greater detail and a *Reference Library* with links to more than 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 eighth edition. These include a complete online *Instructor's Guide* (also downloadable) and supplementary teaching materials including a complete set of more than 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 tools, a comprehensive collection of Web-based resources, and an “adaptable process model” that provides a detailed task breakdown of the software engineering process.



McGraw-Hill Connect® Computer Science provides online presentation, assignment, and assessment solutions. It connects your students with the tools and

resources they'll need to achieve success. With Connect **Computer Science** you can deliver assignments, quizzes, and tests online. A robust set of questions and activities are presented and aligned with the textbook's learning outcomes. As an instructor, you can edit existing questions and author entirely new problems. Integrate grade reports easily with Learning Management Systems (LMS), such as WebCT and Blackboard—and much more. ConnectPlus® **Computer Science** provides students with all the advantages of Connect **Computer Science**, plus 24/7 online access to a media-rich eBook, allowing seamless integration of text, media, and assessments. To learn more, visit www.mcgrawhillconnect.com



LEARNSMART® McGraw-Hill LearnSmart® is available as a standalone product or

an integrated feature of McGraw-Hill Connect **Computer Science**. It is an adaptive learning system designed to help students learn faster, study more efficiently, and retain more knowledge for greater success. LearnSmart assesses a student's knowledge of course content through a series of adaptive questions. It pinpoints concepts the student does not understand and maps out a personalized study plan for success. This innovative study tool also has features that allow instructors to see exactly what students have accomplished and a built-in assessment tool for graded assignments. Visit the following site for a demonstration. www.mhlearnsmart.com



SMARTBOOK™ Powered by the intelligent and adaptive LearnSmart engine, **SmartBook™**

is the first and only continuously adaptive reading experience available today. Distinguishing what students know from what they don't, and honing in on concepts they are most likely to forget, SmartBook personalizes content for each student. Reading is no longer a passive and linear experience but an engaging and dynamic one, where students are more likely to master and retain important concepts, coming to class better prepared. SmartBook includes powerful reports that identify specific topics and learning objectives students need to study.

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

With this edition of *Software Engineering: A Practitioner's Approach*, Bruce Maxim joins me (Roger Pressman) as a coauthor of the book. Bruce brought copious software engineering knowledge to the project and has added new content and insight that will be invaluable to readers of this edition.

Acknowledgments. Special thanks go to Tim Lethbridge of the University of Ottawa who assisted us 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.

In addition, we'd like to thank Austin Krauss, Senior Software Engineer at Treyarch, for providing insight into software development in the video game industry. We also wish to thank the reviewers of the eighth edition: Manuel E. Bermudez, University of Florida; Scott DeLoach, Kansas State University; Alex Liu, Michigan State University; and Dean Mathias, Utah State University. Their in-depth comments and thoughtful criticism have helped us make this a much better book.

Special Thanks. BRM: I am grateful to have had the opportunity to work with Roger on the eighth edition of this book. During the time I have been working on this book my son Benjamin shipped his first MobileApp and my daughter Katherine launched her interior design career. I am quite pleased to see the adults they have become. I am very grateful to my wife, Norma, for the enthusiastic support she has given me as I filled my free time with working on this book.

RSP: 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. They now have children of their own, Maya and Lily, who start still another generation. Both girls are already wizards on mobile computing devices. Finally, to my wife 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

Bruce R. Maxim

TABLE OF CONTENTS

Preface xxvii

CHAPTER 1 THE NATURE OF SOFTWARE 1

1.1	The Nature of Software	3
1.1.1	Defining Software	4
1.1.2	Software Application Domains	6
1.1.3	Legacy Software	7
1.2	The Changing Nature of Software	9
1.2.1	WebApps	9
1.2.2	Mobile Applications	9
1.2.3	Cloud Computing	10
1.2.4	Product Line Software	11
1.3	Summary	11
	PROBLEMS AND POINTS TO PONDER	12
	FURTHER READINGS AND INFORMATION SOURCES	12

CHAPTER 2 SOFTWARE ENGINEERING 14

2.1	Defining the Discipline	15
2.2	The Software Process	16
2.2.1	The Process Framework	17
2.2.2	Umbrella Activities	18
2.2.3	Process Adaptation	18
2.3	Software Engineering Practice	19
2.3.1	The Essence of Practice	19
2.3.2	General Principles	21
2.4	Software Development Myths	23
2.5	How It All Starts	26
2.6	Summary	27
	PROBLEMS AND POINTS TO PONDER	27
	FURTHER READINGS AND INFORMATION SOURCES	27

PART ONE THE SOFTWARE PROCESS 29

CHAPTER 3 SOFTWARE PROCESS STRUCTURE 30

3.1	A Generic Process Model	31
3.2	Defining a Framework Activity	32
3.3	Identifying a Task Set	34
3.4	Process Patterns	35
3.5	Process Assessment and Improvement	37
3.6	Summary	38
	PROBLEMS AND POINTS TO PONDER	38
	FURTHER READINGS AND INFORMATION SOURCES	39

CHAPTER 4 PROCESS MODELS 40

-
- 4.1 Prescriptive Process Models 41
 - 4.1.1 The Waterfall Model 41
 - 4.1.2 Incremental Process Models 43
 - 4.1.3 Evolutionary Process Models 45
 - 4.1.4 Concurrent Models 49
 - 4.1.5 A Final Word on Evolutionary Processes 51
 - 4.2 Specialized Process Models 52
 - 4.2.1 Component-Based Development 53
 - 4.2.2 The Formal Methods Model 53
 - 4.2.3 Aspect-Oriented Software Development 54
 - 4.3 The Unified Process 55
 - 4.3.1 A Brief History 56
 - 4.3.2 Phases of the Unified Process 56
 - 4.4 Personal and Team Process Models 59
 - 4.4.1 Personal Software Process 59
 - 4.4.2 Team Software Process 60
 - 4.5 Process Technology 61
 - 4.6 Product and Process 62
 - 4.7 Summary 64

PROBLEMS AND POINTS TO PONDER 64

FURTHER READINGS AND INFORMATION SOURCES 65

CHAPTER 5 AGILE DEVELOPMENT 66

-
- 5.1 What Is Agility? 68
 - 5.2 Agility and the Cost of Change 68
 - 5.3 What Is an Agile Process? 69
 - 5.3.1 Agility Principles 70
 - 5.3.2 The Politics of Agile Development 71
 - 5.4 Extreme Programming 72
 - 5.4.1 The XP Process 72
 - 5.4.2 Industrial XP 75
 - 5.5 Other Agile Process Models 77
 - 5.5.1 Scrum 78
 - 5.5.2 Dynamic Systems Development Method 79
 - 5.5.3 Agile Modeling 80
 - 5.5.4 Agile Unified Process 82
 - 5.6 A Tool Set for the Agile Process 83
 - 5.7 Summary 84

PROBLEMS AND POINTS TO PONDER 85

FURTHER READINGS AND INFORMATION SOURCES 85

CHAPTER 6 HUMAN ASPECTS OF SOFTWARE ENGINEERING 87

-
- 6.1 Characteristics of a Software Engineer 88
 - 6.2 The Psychology of Software Engineering 89
 - 6.3 The Software Team 90
 - 6.4 Team Structures 92
 - 6.5 Agile Teams 93
 - 6.5.1 The Generic Agile Team 93
 - 6.5.2 The XP Team 94

6.6	The Impact of Social Media	95
6.7	Software Engineering Using the Cloud	97
6.8	Collaboration Tools	98
6.9	Global Teams	99
6.10	Summary	100

PROBLEMS AND POINTS TO PONDER	101
-------------------------------	-----

FURTHER READINGS AND INFORMATION SOURCES	102
--	-----

PART TWO MODELING 103

CHAPTER 7 PRINCIPLES THAT GUIDE PRACTICE 104

7.1	Software Engineering Knowledge	105
7.2	Core Principles	106
7.2.1	Principles That Guide Process	106
7.2.2	Principles That Guide Practice	107
7.3	Principles That Guide Each Framework Activity	109
7.3.1	Communication Principles	110
7.3.2	Planning Principles	112
7.3.3	Modeling Principles	114
7.3.4	Construction Principles	121
7.3.5	Deployment Principles	125
7.4	Work Practices	126
7.5	Summary	127

PROBLEMS AND POINTS TO PONDER	128
-------------------------------	-----

FURTHER READINGS AND INFORMATION SOURCES	129
--	-----

CHAPTER 8 UNDERSTANDING REQUIREMENTS 131

8.1	Requirements Engineering	132
8.2	Establishing the Groundwork	138
8.2.1	Identifying Stakeholders	139
8.2.2	Recognizing Multiple Viewpoints	139
8.2.3	Working toward Collaboration	140
8.2.4	Asking the First Questions	140
8.2.5	Nonfunctional Requirements	141
8.2.6	Traceability	142
8.3	Eliciting Requirements	142
8.3.1	Collaborative Requirements Gathering	143
8.3.2	Quality Function Deployment	146
8.3.3	Usage Scenarios	146
8.3.4	Elicitation Work Products	147
8.3.5	Agile Requirements Elicitation	148
8.3.6	Service-Oriented Methods	148
8.4	Developing Use Cases	149
8.5	Building the Analysis Model	154
8.5.1	Elements of the Analysis Model	154
8.5.2	Analysis Patterns	157
8.5.3	Agile Requirements Engineering	158
8.5.4	Requirements for Self-Adaptive Systems	158
8.6	Negotiating Requirements	159

8.7	Requirements Monitoring	160
8.8	Validating Requirements	161
8.9	Avoiding Common Mistakes	162
8.10	Summary	162
	PROBLEMS AND POINTS TO PONDER	163
	FURTHER READINGS AND OTHER INFORMATION SOURCES	164

CHAPTER 9 REQUIREMENTS MODELING: SCENARIO-BASED METHODS 166

9.1	Requirements Analysis	167
9.1.1	Overall Objectives and Philosophy	168
9.1.2	Analysis Rules of Thumb	169
9.1.3	Domain Analysis	170
9.1.4	Requirements Modeling Approaches	171
9.2	Scenario-Based Modeling	173
9.2.1	Creating a Preliminary Use Case	173
9.2.2	Refining a Preliminary Use Case	176
9.2.3	Writing a Formal Use Case	177
9.3	UML Models That Supplement the Use Case	179
9.3.1	Developing an Activity Diagram	180
9.3.2	Swimlane Diagrams	181
9.4	Summary	182
	PROBLEMS AND POINTS TO PONDER	182
	FURTHER READINGS AND INFORMATION SOURCES	183

CHAPTER 10 REQUIREMENTS MODELING: CLASS-BASED METHODS 184

10.1	Identifying Analysis Classes	185
10.2	Specifying Attributes	188
10.3	Defining Operations	189
10.4	Class-Responsibility-Collaborator Modeling	192
10.5	Associations and Dependencies	198
10.6	Analysis Packages	199
10.7	Summary	200
	PROBLEMS AND POINTS TO PONDER	201
	FURTHER READINGS AND INFORMATION SOURCES	201

CHAPTER 11 REQUIREMENTS MODELING: BEHAVIOR, PATTERNS, AND WEB/MOBILE APPS 202

11.1	Creating a Behavioral Model	203
11.2	Identifying Events with the Use Case	203
11.3	State Representations	204
11.4	Patterns for Requirements Modeling	207
11.4.1	Discovering Analysis Patterns	208
11.4.2	A Requirements Pattern Example: Actuator-Sensor	209
11.5	Requirements Modeling for Web and Mobile Apps	213
11.5.1	How Much Analysis Is Enough?	214
11.5.2	Requirements Modeling Input	214
11.5.3	Requirements Modeling Output	215
11.5.4	Content Model	216