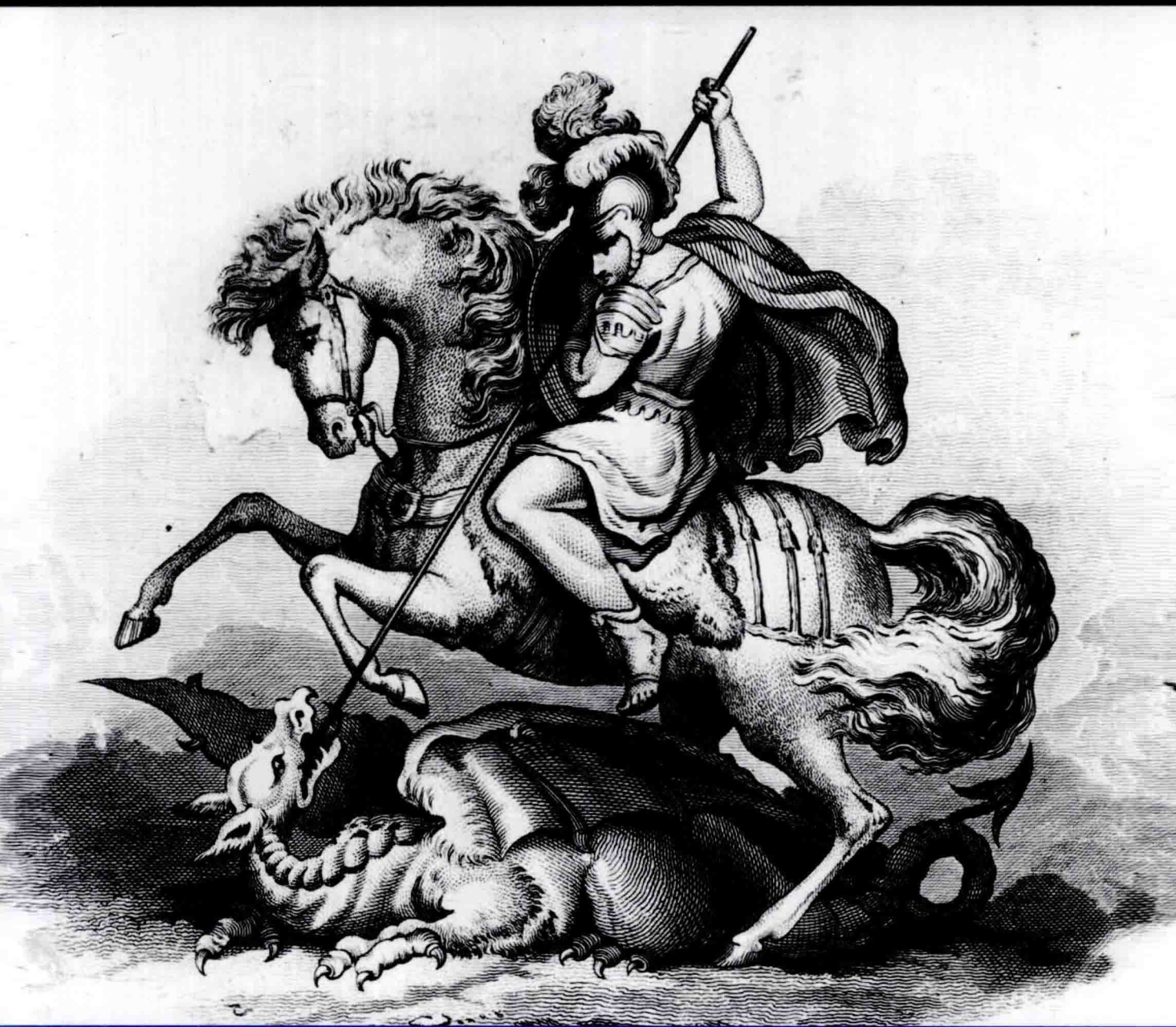
Second Edition

Software Engineering

Modern Approaches



Eric J. Braude • Michael E. Bernstein

Software Engineering

Modern Approaches

SECOND EDITION

Eric J. Braude
Boston University, Metropolitan College

Michael E. Bernstein
Boston University, Metropolitan College

WILEY

JOHN WILEY & SONS, INC.

Executive Editor	Beth Lang Golub
Editor	Jonathan Shipley
Assistant Editor	Georgia King
Editorial Assistant	Mike Berlin
Marketing Manager	Christopher Ruel
Designer	RDC Publishing Group Sdn Bhd
Production Manager	Janis Soo
Assistant Production Editor	Yee Lyn Song

Cover Credit: © Hulton Archive/Getty Images

This book was set in 10/12 Point Weiss by Thomson Digital, and printed and bound by R.R. Donnelley. The cover was printed by R.R. Donnelley.

This book is printed on acid free paper. ∞

Library of Congress Cataloging-in-Publication Data

Braude, Eric J.
Software engineering : modern approaches / Eric J. Braude, Michael E. Bernstein. — 2nd ed.
p. cm.
Includes bibliographical references and index.
ISBN 978-0-471-69208-9 (cloth)
1. Software engineering. 2. Object-oriented programming (Computer science) I. Bernstein, Michael E. II. Title.
QA76.758.B74 2011
005.1—dc22

2009051247

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

For Judy (Eric J. Braude)
To Bambi, Garrett and Reid,
for all their love and support (Michael E. Bernstein)

Preface

Much of the modern world runs on software. As a result, software engineers are entrusted with significant responsibility. Although it is a biomedical engineer, for example, who designs health monitoring systems, it is a software engineer who creates its actual control functions. A marketing professional develops ways to reach customers online but it is a software engineer who makes the system a reality.

Today's software engineer must be able to participate in more than one kind of software process, work in agile teams, deal with customers, express requirements clearly, create modular designs, utilize legacy and open source projects, monitor quality, incorporate security, and apply many types of tests.

THE ISSUE OF SCALE

A software application consists of tens, hundreds, even thousands of classes. This is very different from managing three or four of them, and results in the dragon of complexity suggested by this book's cover. As also suggested there, however, this dragon can be subdued. Indeed, to deal with numerous and complex classes, software engineers have at their disposal a wide variety of tools and techniques. These range from the waterfall process to agile methodologies, from highly integrated tool suites to refactoring and loosely coupled tool sets. Underlying this variety is continuing research into reliable approaches, and an acknowledgment of the fact that one size does not fit all projects.

THIS EDITION COMPARED WITH THE FIRST

The first edition of this book emphasized the object-oriented approach, which has subsequently become widespread. It was also designed to help student teams carry out hands-on term projects through theory, examples, case studies, and practical steps. Object-orientation and hands-on continue to be major features of this edition. However, we have widened the scope of the first edition, especially by including extensive coverage of agile methods and refactoring, together with deeper coverage of quality and software design.

Readers of the first edition made extensive use of the complete video game case study—an example that they could follow "from soup to nuts" but which was significantly more comprehensive than a toy. This edition retains and updates that case study, but it adds the exploration of a simpler example on one hand (a DVD rental store) and large, real, open source case studies on the other. In particular, to provide students a feeling for the scope and complexity of real-world applications, this book leads them through selected requirements, design, implementation, and maintenance of the Eclipse and OpenOffice open source projects. The size, complexity, and transparency of these projects provide students a window into software engineering on a realistic scale.

Every book on software engineering faces a dilemma: how to reconcile the organization of the *topics* with the organization of actual software project *phases*. An organization of chapters into process/project management/requirements analysis/design/implementation/test/maintenance is straightforward but is liable to be misinterpreted as promoting the waterfall development process at the expense others. Our approach has been to use this organization in the seven parts of the book but to demonstrate throughout that each phase

typically belongs to a cycle rather than to a single waterfall sequence. In particular, our approach integrates agile methodologies consistently.

This edition also introduces somewhat advanced influential ideas, including model-driven architectures and aspect-oriented programming. Nowadays, formal methods are mandated by government agencies for the highest levels of security, and this book aims to educate readers in their possibilities. Due to print space limitations, some of this material is to be found in the online extension of this book.

In summary, specific features of this edition compared with the first are as follows:

- A sharpening and standardization of the material from the first edition
- A strong agile thread throughout, including a chapter on agility alone and one devoted to refactoring.
- A separate chapter on quality in six of the book's seven parts
- Real-world case studies, taken from the Eclipse and OpenOffice open source projects
- Greatly expanded coverage of software design and design patterns
- New chapters on advanced, influential software engineering ideas
- An organization of many of the book's seven parts as follows:
  - Principles
  - Details
  - Quality
  - Advanced Methods

HOW INSTRUCTORS CAN USE THIS BOOK

This book has been designed to accommodate multiple approaches to the learning and teaching of software engineering. Most instructors teach the fundamentals of software process, project management, requirements analysis, design, testing, implementation, and maintenance. Beyond this common ground, however, instructors employ a wide variety of styles and emphases. The following are major approaches, together with the sequence of chapters that support each of them.

A. *Process emphasis*, concentrating on how applications are developed
   All of Parts I through IV; and Chapters 15, 22, and 25 (the remaining principles and introduction chapters)

B. *Design emphasis*, which teaches software engineering primarily as a design activity
   Principles and introduction: Chapters 1, 3, 7, and 10; all of Part V; and Chapters 22 and 25 (principles and introduction)

C. *Programming and agile emphasis*, which emphasizes software engineering as a code-oriented activity that satisfies requirements, emphasizing agile approaches
   Principles and introduction: Chapters 1, 3, 7, 10, and 15; all of Part VI; and Chapters 25 and 26

D. *Two-semester course*, which enables the instructor to cover most topics and assign a substantial hands-on project

D1. All of the chapters in the book, either in sequence from beginning to end
or

D2. In two passes as follows:

(i) Principles and introduction chapters in the first semester: Chapters 1, 3, 7, 15, 22, and 25
(ii) The remaining chapters in the second semester

E. *Emphasis on a hands-on projects and case studies*, which relies mostly on an active team or individual project as the vehicle for learning theory and principles
Principles and introduction chapters: Chapters 1, 3, 7, 15, 22, 25, and 26, and all case study sections in the remaining chapters

F. *Theory and principles emphasis*, concentrating on what one can learn about software engineering and its underpinnings
Principles and introduction chapters: Chapters 1, 2, 3, 7, 15, 22, and 25, followed, as time allows, by Chapters 14 and 21 (emerging topics)

G. *Quality assurance and testing emphasis*
Principles and introduction: Chapters 1, 3, 7, and 10; Chapters 2, 5, 9, 13, 20, 23 (quality); and Chapters 25, 26, 27, and 28 (testing).

The web site for this book, including review questions and the Encounter game case study, is www.wiley.com/college/braude.

Eric Braude
Michael Bernstein
Boston, MA
January 2010

Acknowledgments

We owe a debt of gratitude to our students at Boston University's Metropolitan College. Working in myriad industries and businesses, they have given us invaluable feedback. The College itself has provided a model place for the teaching and learning software engineering. Our thanks go to Dick Bostwick and Tom VanCourt, much of whose work in the first edition carries over to this one. We are grateful to the people of Wiley for working with us through the painstaking process of writing and publishing this book. We are particularly appreciative of the help from our editors, Dan Sayre and Jonathan Shipley; from Georgia King, Yee Lyn Song, and the indefatigable staff. We thank the reviewers of our manuscript, whose feedback has been invaluable:

Arvin Agah, University of Kansas
Steven C. Shaffer, Pennsylvania State University
Stephen M. Thebaut, University of Florida
Aravinda P. Sistla, University of Illinois, Chicago
James P. Purtilo, University of Maryland
Linda M. Ott, Michigan Technological University
Jianwei Niu, University of Texas, San Antonio
William Lively, Texas A&M University
Chung Lee, California State University, Pomona
Sudipto Ghosh, Colorado State University
Max I. Fomitchev, Pennsylvania State University
Lawrence Bernstein, Stevens Institute of Technology
John Dalbey, California Polytechnic University
Len Fisk, California State University, Chico
Ahmed M. Salem, California State University, Sacramento
Fred Strauss, New York University
Kai H. Chang, Auburn University
Andre van der Hoek, University of California, Irvine
Saeed Monemi, California Polytechnic University
Robert M. Cubert, University of Florida
Chris Tseng, San Jose State University
Michael James Payne, Purdue University
Carol A. Wellington, Shippensburg University
Yifei Dong, University of Oklahoma
Peter Blanchfield, Nottingham University
Desmond Greer, Queen's University Belfast
WeiQi Yan, Queen's University Belfast
Zaigham Mahmood, Derby University
Karel Pieterson, Hogeschool Van Amsterdam

This book would not have been possible without the constant love, patience, and encouragement of our families.

Brief Contents

Contents