



ESSENTIALS OF

software engineering

THIRD EDITION

Frank Tsui
Orlando Karam
Barbara Bernal



ESSENTIALS OF

software engineering

THIRD EDITION

Frank Tsui
Orlando Karam
Barbara Bernal

All of Southern Polytechnic State University



JONES & BARTLETT
LEARNING

World Headquarters
Jones & Bartlett Learning
5 Wall Street
Burlington, MA 01803
978-443-5000
info@jblearning.com
www.jblearning.com

Jones & Bartlett Learning books and products are available through most bookstores and online booksellers. To contact Jones & Bartlett Learning directly, call 800-832-0034, fax 978-443-8000, or visit our website, www.jblearning.com.

Substantial discounts on bulk quantities of Jones & Bartlett Learning publications are available to corporations, professional associations, and other qualified organizations. For details and specific discount information, contact the special sales department at Jones & Bartlett Learning via the above contact information or send an email to specialsales@jblearning.com.

Copyright © 2014 by Jones & Bartlett Learning, LLC, an Ascend Learning Company

All rights reserved. No part of the material protected by this copyright may be reproduced or utilized in any form, electronic or mechanical, including photocopying, recording, or by any information storage and retrieval system, without written permission from the copyright owner.

Essentials of Software Engineering, Third Edition is an independent publication and has not been authorized, sponsored, or otherwise approved by the owners of the trademarks or service marks referenced in this product.

The screenshots in this product are for educational and instructive purposes only. All trademarks displayed are the trademarks of the parties noted therein. Such use of trademarks is not an endorsement by said parties of Jones & Bartlett Learning, its products, or its services, nor should such use be deemed an endorsement by Jones & Bartlett Learning of said third party's products or services.

Production Credits

Executive Publisher: Kevin Sullivan
Senior Developmental Editor: Amy Bloom
Director of Production: Amy Rose
Production Assistant: Eileen Worthley
Marketing Manager: Lindsay White
V.P., Manufacturing and Inventory Control: Therese Connell
Composition: Northeast Compositors, Inc.
Cover and Title Page Design: Kristin E. Parker
Cover and Title Page Image: © Kushch Dmitry/Shutterstock, Inc.
Printing and Binding: Edwards Brothers Malloy
Cover Printing: Edwards Brothers Malloy

Library of Congress Cataloging-in-Publication Data

Tsui, Frank F.

Essentials of software engineering. -- Third edition / Frank Tsui, Orlando Karam, Barbara Bernal.
pages cm

Includes bibliographical references and index.

ISBN 978-1-4496-9199-8 (pbk.) -- ISBN 1-4496-9199-4 (pbk.)

1. Software engineering. I. Karam, Orlando. II. Bernal, Barbara. III. Title.

QA76.758.T78 2014

005.1--dc23

2012029913

6048

Printed in the United States of America

17 16 15 14 10 9 8 7 6 5 4 3



Preface

Essentials of Software Engineering was born from our experiences in teaching introductory material on software engineering. Although there are many books on this topic available in the market, few serve the purpose of introducing only the core material for a one-semester course that meets approximately three hours a week for 16 weeks. With the proliferation of small web applications, many new information technology personnel have entered the field of software engineering without fully understanding what it entails. This book is intended to serve both new students with limited experience as well as experienced information technology professionals who are contemplating a new career in the software engineering discipline. The complete life cycle of a software system is covered in this book, from inception to release and through support.

The content of this book has also been shaped by our personal experiences and backgrounds—one of us with more than 25 years in building, supporting, and managing large and complex mission-critical software with companies such as IBM®, BlueCross BlueShield, MARCAM, and RCA and another with extensive expertise in constructing smaller software with Agile methods.

Although new ideas and technology will continue to emerge and some of the principles introduced in this book may have to be updated, we believe that the underlying and fundamental concepts we present here will remain.

Preface to the Third Edition

For this third edition, our goal is, again, to improve the text without growing it beyond the original intent, which was to include only the essential topics such that they can be covered within a one-semester introduction to software engineering course. The flow of the text has also been kept constant throughout the different editions.

Thanks to feedback from many readers and students, we have made numerous corrections and small commentary changes. We have proactively solicited input from those who have used this as a textbook in their classes and have incorporated many of their suggestions. As such, we have been joined by a third author, Barbara Bernal, who has used this book as the text in her introduction to software engineering classes for several years.

This third edition includes the following main modifications and additions:

- Addition of Scrum method and elimination of some lesser-used processes in Chapter 5
- Expanded UI design discussion that includes an example of HTML-Script-SQL design and implementation in Chapter 7
- Inclusion of “essential samples” for Team Plan, Software Development Plan, Requirements Specification, Design Plan, and Test Plan, presented in new appendices
- Retitled Chapter 14 from “Epilogue” to “Epilogue and Some Contemporary Issues” to briefly relate some current issues within software engineering

The first and second editions of this book have been used by numerous colleges and universities, and we thank them for their patience and input. We have learned a lot in the process. We hope the third edition will prove to be a better one for all future readers.

Organization of the Book

Chapters 1 and 2 demonstrate the difference between a small programming project and the effort required to construct a mission-critical software system. We purposely took two chapters to demonstrate this concept, highlighting the difference between a single-person “garage” operation and a team project required to construct a large “professional” system. The discussion in these two chapters delineates the rationale for studying and understanding software engineering. Chapter 3 is the first place where software engineering is discussed more formally. Included in this chapter is an introduction to the profession of software engineering and its code of ethics.

The traditional topics of software processes, process models, and methodologies are covered in Chapters 4 and 5. Reflecting the vast amount of progress made in this area, these chapters explain in extensive detail how to evaluate the processes through the Capability Maturity Models from the Software Engineering Institute (SEI).

Chapters 6, 7, 9, 10, and 11 cover the sequence of development activities from requirements through product release at a macro level. Chapter 8, following the chapter on software design, steps back and discusses design characteristics and metrics utilized

in evaluating high-level and detail designs. Chapter 11 discusses not only product release, but the general concept of configuration management.

Chapter 12 explores the support and maintenance activities related to a software system after it is released to customers and users. Topics covered include call management, problem fixes, and feature releases. The need for configuration management is further emphasized in this chapter. Chapter 13 summarizes the phases of project management, along with some specific project planning and monitoring techniques. It is only a summary, and some topics, such as team building and leadership qualities, are not included. The software project management process is contrasted from the development and support processes. Chapter 14 concludes the book and provides a view of some of the future topics in our field.

The new appendices for this third edition give readers and students insight into possible results from major activities in software development. An often-asked question is what a requirements document or a test plan should look like. To help answer this question and provide a starting point, we have included sample formats of possible documents resulting from the four activities of Planning, Requirements, Design, and Test Plan. These are provided as follows:

- Appendix A Essential Software Development Plan (SDP)
- Appendix B Essential Software Requirements Specifications (SRS)
 - Example 1: Essential SRS—Descriptive
 - Example 2: Essential SRS—Object Oriented
 - Example 3: Essential SRS—IEEE Standard
 - Example 4: Essential SRS— Narrative Approach
- Appendix C Essential Software Design
 - Example 1: Essential Software Design—UML
 - Example 2: Essential Software Design—Structural
- Appendix D Essential Test Plan

Many times in the development of team projects by novice software engineers there is a need for specific direction on how to document the process. The four new appendices were developed to give the reader concrete examples of the possible essential outlines. Each of the appendices gives an outline with explanations. This provides the instructor with concrete material to supplement class activities, team project assignments, and/or independent work.

The topical coverage in this book reflects those emphasized by the IEEE Computer Society-sponsored *Software Engineering Body of Knowledge (SWEBOK)* and by the *Software Engineering 2004 Curriculum Guidelines for Undergraduate Degree Program in Software Engineering*. The one topic that is not highlighted but is discussed throughout the book concerns quality—a topic that needs to be addressed and integrated into all activities. It is not just a concern of the testers. Quality is discussed in multiple chapters to reflect its broad implications and cross activities.

Suggested Teaching Plan

All the chapters in this book can be covered within one semester. However, some instructors may prefer different emphasis:

- Those who want to focus on direct development activities should spend more time on Chapters 6 through 11.
- Those who want to focus more on indirect and general activities should spend more time on Chapters 1, 12, and 13.

It should be pointed out that both the direct development and the indirect support activities are important. The combined set forms the software engineering discipline.

There are two sets of questions at the end of each chapter. For the Review Questions, students can find answers directly in the chapter. The Exercises are meant to be used for potential class discussion, homework, or small projects.

Supplements

PowerPoint Lecture Outlines, Answers to End-of-Chapter Exercises, and sample Test Questions are available for free instructor download. To request access, please visit go.jblearning.com/Tsui3e or contact your account representative.

Acknowledgments

We would first like to thank our families, especially our wives, Lina Colli and Teresa Tsui. They provided constant encouragement and understanding when we spent more time with the manuscript than with them. Our children—Colleen and Nicholas; Orlando and Michelle; and Victoria, Liz, and Alex—enthusiastically supported our efforts as well. We would also like to thank Han Reichgelt, Dean of the School of Computing and Software Engineering at Southern Polytechnic State University, and Lisa Rossbacher, President of Southern Polytechnic State University, for providing us with a supportive and conducive environment for manuscript research and writing.

In addition, we would like to thank the reviewers who have improved the book in many ways. We would like to specifically thank the following individuals:

- Brent Auernheimer, California State University–Fresno
- Ayad Boudiab, Georgia Perimeter College
- Kai Chang, Auburn University
- David Gustafson, Kansas State University
- Theresa Jefferson, George Washington University
- Dar-Biau Liu, California State University–Long Beach
- Bruce Logan, Lesley University
- Jeanna Matthews, Clarkson University

- Michael Oudshoorn, Montana State University
- Frank Ackerman, Montana Tech
- Mark Hall, Hastings College
- Dimitris Papamichail, University of Miami
- Dr. Jody Paul, Metro State Denver

We continue to appreciate the help from Tim Anderson, Amy Bloom, Eileen Worthley, and others at Jones & Bartlett Learning.

Any remaining error is solely the mistake of the authors.

—*Frank Tsui*

—*Orlando Karam*

—*Barbara Bernal*



Contents

Preface iii

Chapter 1 Writing a Program 1

- 1.1 A Simple Problem 2
- 1.2 Decisions, Decisions 2
 - 1.2.1 Functional Requirements 3
 - 1.2.2 Nonfunctional Requirements 4
 - 1.2.3 Design Constraints 5
 - 1.2.4 Design Decisions 6
- 1.3 Testing 6
- 1.4 Estimating Effort 7
- 1.5 Implementations 8
 - 1.5.1 A Few Pointers on Implementation 8
 - 1.5.2 Basic Design 10
 - 1.5.3 Unit Testing with JUnit 10
 - 1.5.4 Implementation of StringSorter 10
 - 1.5.5 User Interfaces 16

- 1.6 Summary 19
- 1.7 Review Questions 19
- 1.8 Exercises 20
- 1.9 Suggested Readings 20

Chapter 2 Building a System 23

- 2.1 Characteristics of Building a System 24
 - 2.1.1 Size and Complexity 24
 - 2.1.2 Technical Considerations of Development and Support 25
 - 2.1.3 Nontechnical Considerations of Development and Support 29
- 2.2 Building a Hypothetical System 30
 - 2.2.1 Requirements of the Payroll System 30
 - 2.2.2 Designing the Payroll System 32
 - 2.2.3 Code and Unit Testing the Payroll System 34
 - 2.2.4 Integration and Functionally Testing the Payroll System 35
 - 2.2.5 Release of the Payroll System 36
 - 2.2.6 Support and Maintenance 36
- 2.3 Coordination Efforts 37
 - 2.3.1 Process 37
 - 2.3.2 Product 38
 - 2.3.3 People 38
- 2.4 Summary 39
- 2.5 Review Questions 39
- 2.6 Exercises 39
- 2.7 Suggested Readings 40

Chapter 3 Engineering of Software 41

- 3.1 Examples and Characteristics of Software Failures 42
 - 3.1.1 Project Failures 42
 - 3.1.2 Software Product Failures 43
 - 3.1.3 Coordination and Other Concerns 44
- 3.2 Software Engineering 45
 - 3.2.1 What Is Software Engineering? 45

3.2.2	Definitions of Software Engineering	45
3.2.3	Relevancy of Software Engineering and Software	46
3.3	Software Engineering Profession and Ethics	47
3.3.1	Software Engineering Code of Ethics	47
3.3.2	Professional Behavior	49
3.4	Principles of Software Engineering	49
3.4.1	Davis's Early Principles of Software Engineering	50
3.4.2	Royce's More Modern Principles	52
3.4.3	Wasserman's Fundamental Software Engineering Concepts	52
3.5	Summary	54
3.6	Review Questions	54
3.7	Exercises	54
3.8	Suggested Readings	55

Chapter 4 Software Process Models 57

4.1	Software Processes	58
4.1.1	Goal of Software Process Models	58
4.1.2	The "Simplest" Process Model	58
4.2	Traditional Process Models	59
4.2.1	Waterfall Model	59
4.2.2	Chief Programmer Team Approach	61
4.2.3	Incremental Model	61
4.2.4	Spiral Model	63
4.3	A More Modern Process	64
4.3.1	General Foundations of Rational Unified Process Framework	64
4.3.2	The Phases of RUP	65
4.4	Entry and Exit Criteria	68
4.4.1	Entry Criteria	69
4.4.2	Exit Criteria	69
4.5	Process Assessment Models	70
4.5.1	SEI's Capability Maturity Model	70
4.5.2	SEI's Capability Maturity Model Integrated	72
4.6	Process Definition and Communication	78
4.7	Summary	79

- 4.8 Review Questions 80
- 4.9 Exercises 80
- 4.10 Suggested Readings 81

Chapter 5 New and Emerging Process Methodologies 83

- 5.1 What Are Agile Processes? 84
- 5.2 Why Agile Processes? 85
- 5.3 Some Process Methodologies 86
 - 5.3.1 Extreme Programming (XP) 86
 - 5.3.2 The Crystal Family of Methodologies 91
 - 5.3.3 The Unified Process as Agile 94
 - 5.3.4 Scrum 94
 - 5.3.5 Open Source Software Development 96
 - 5.3.6 Summary of Processes 98
- 5.4 Choosing a Process 98
 - 5.4.1 Projects and Environments Better Suited for Each Kind of Process 99
 - 5.4.2 Main Risks and Disadvantages of Agile Processes 99
 - 5.4.3 Main Advantages of Agile Processes 100
- 5.5 Summary 101
- 5.6 Review Questions 101
- 5.7 Exercises 101
- 5.8 Suggested Readings 102

Chapter 6 Requirements Engineering 103

- 6.1 Requirements Processing 104
 - 6.1.1 Preparing for Requirements Processing 104
 - 6.1.2 Requirements Engineering Process 105
- 6.2 Requirements Elicitation and Gathering 107
 - 6.2.1 Eliciting High-Level Requirements 108
 - 6.2.2 Eliciting Detailed Requirements 110
- 6.3 Requirements Analysis 112
 - 6.3.1 Requirements Analysis and Clustering by Business Flow 112
 - 6.3.2 Requirements Analysis and Clustering with Object-Oriented Use Cases 114

- 6.3.3 Requirements Analysis and Clustering by Viewpoint-Oriented Requirements Definition 116
- 6.3.4 Requirements Analysis and Prioritization 117
- 6.3.5 Requirements Traceability 120
- 6.4 Requirements Definition, Prototyping, and Reviews 120
- 6.5 Requirements Specification and Requirements Agreement 124
- 6.6 Summary 125
- 6.7 Review Questions 126
- 6.8 Exercises 127
- 6.9 Suggested Readings 127

Chapter 7 Design: Architecture and Methodology 129

- 7.1 Introduction to Design 130
- 7.2 Architectural Design 131
 - 7.2.1 What Is Software Architecture? 131
 - 7.2.2 Views and Viewpoints 131
 - 7.2.3 Meta-Architectural Knowledge: Styles, Patterns, Tactics, and Reference Architectures 133
- 7.3 Detailed Design 139
 - 7.3.1 Functional Decomposition 139
 - 7.3.2 Relational Database Design 141
 - 7.3.3 Object-Oriented Design and UML 146
 - 7.3.4 User-Interface Design 152
 - 7.3.5 Some Further Design Concerns 157
- 7.4 HTML-Script-SQL Design Example 158
- 7.5 Summary 161
- 7.6 Review Questions 161
- 7.7 Exercises 162
- 7.8 Suggested Readings 162

Chapter 8 Design Characteristics and Metrics 165

- 8.1 Characterizing Design 166
- 8.2 Some Legacy Characterizations of Design Attributes 166
 - 8.2.1 Halstead Complexity Metric 166
 - 8.2.2 McCabe's Cyclomatic Complexity 168

8.2.3	Henry-Kafura Information Flow	169
8.2.4	A Higher-Level Complexity Measure	170
8.3	"Good" Design Attributes	171
8.3.1	Cohesion	171
8.3.2	Coupling	175
8.4	Object-Oriented Design Metrics	177
8.4.1	Aspect-Oriented Programming	179
8.4.2	The Law of Demeter	179
8.5	User-Interface Design	180
8.5.1	Good UI Characteristics	180
8.5.2	Usability Evaluation and Testing	181
8.6	Summary	182
8.7	Review Questions	183
8.8	Exercises	184
8.9	Suggested Readings	184
 Chapter 9 Implementation 187		
9.1	Introduction to Implementation	188
9.2	Characteristics of a Good Implementation	188
9.3	Programming Style and Coding Guidelines	189
9.4	Comments	191
9.5	Debugging	193
9.6	Assertions and Defensive Programming	194
9.7	Performance Optimization	195
9.8	Refactoring	196
9.9	Summary	197
9.10	Review Questions	197
9.11	Exercises	197
9.12	Suggested Readings	198
 Chapter 10 Testing and Quality Assurance 199		
10.1	Introduction to Testing and Quality Assurance	200
10.2	Testing	202
10.2.1	The Purposes of Testing	202
10.3	Testing Techniques	203
10.3.1	Equivalence Class Partitioning	205

10.3.2	Boundary Value Analysis	207
10.3.3	Path Analysis	208
10.3.4	Combinations of Conditions	213
10.3.5	Automated Unit Testing and Test-Driven Development	213
10.3.6	An Example of Test-Driven Development	214
10.4	When to Stop Testing	218
10.5	Inspections and Reviews	220
10.6	Formal Methods	222
10.7	Static Analysis	223
10.8	Summary	224
10.9	Review Questions	224
10.10	Exercises	225
10.11	Suggested Readings	226

Chapter 11 Configuration Management, Integration, and Builds 229

11.1	Software Configuration Management	230
11.2	Policy, Process, and Artifacts	230
11.2.1	Business Policy Impact on Configuration Management	234
11.2.2	Process Influence on Configuration Management	235
11.3	Configuration Management Framework	236
11.3.1	Naming Model	236
11.3.2	Storage and Access Model	238
11.4	Build and Integration and Build	240
11.5	Tools for Configuration Management	241
11.6	Managing the Configuration Management Framework	243
11.7	Summary	244
11.8	Review Questions	245
11.9	Exercises	245
11.10	Suggested Readings	246

Chapter 12 Software Support and Maintenance 249

12.1	Customer Support	250
12.1.1	User Problem Arrival Rate	250

12.1.2	Customer Interface and Call Management	252
12.1.3	Technical Problem/Fix	254
12.1.4	Fix Delivery and Fix Installs	256
12.2	Product Maintenance Updates and Release Cycles	258
12.3	Change Control	259
12.4	Summary	261
12.5	Review Questions	261
12.6	Exercises	262
12.7	Suggested Readings	262

Chapter 13 Software Project Management 265

13.1	The Necessity of Project Management	266
13.2	The Project Management Process	266
13.2.1	Planning	267
13.2.2	Organizing	270
13.2.3	Monitoring	271
13.2.4	Adjusting	273
13.3	Some Project Management Techniques	275
13.3.1	Project Effort Estimation	275
13.3.2	Work Breakdown Structure	283
13.3.3	Project Status Tracking with Earned Value	286
13.3.4	Measuring Project Properties and GQM	288
13.4	Summary	290
13.5	Review Questions	291
13.6	Exercises	291
13.7	Suggested Readings	293

Chapter 14 Epilogue and Some Contemporary Issues 295

14.1	Security and Software Engineering	297
14.2	Reverse Engineering and Software Obfuscation	298
14.3	Software Validation and Verification Methodologies and Tools	299
14.4	Suggested Readings	300

Appendix A 303

Essential Software Development Plan (SDP)	303
---	-----

Appendix B 305

- Essential Software Requirements Specifications (SRS) 305
 - Example 1: Essential SRS—Descriptive 305
 - Example 2: Essential SRS—Object Oriented 307
 - Example 3: Essential SRS—IEEE Standard 308
 - Example 4: Essential SRS—Narrative Approach 309

Appendix C 311

- Essential Software Design 311
 - Example 1: Essential Software Design—UML 311
 - Example 2: Essential Software Design—
Structural 312

Appendix D 315

- Essential Test Plan 315

Glossary 317**Index 321**