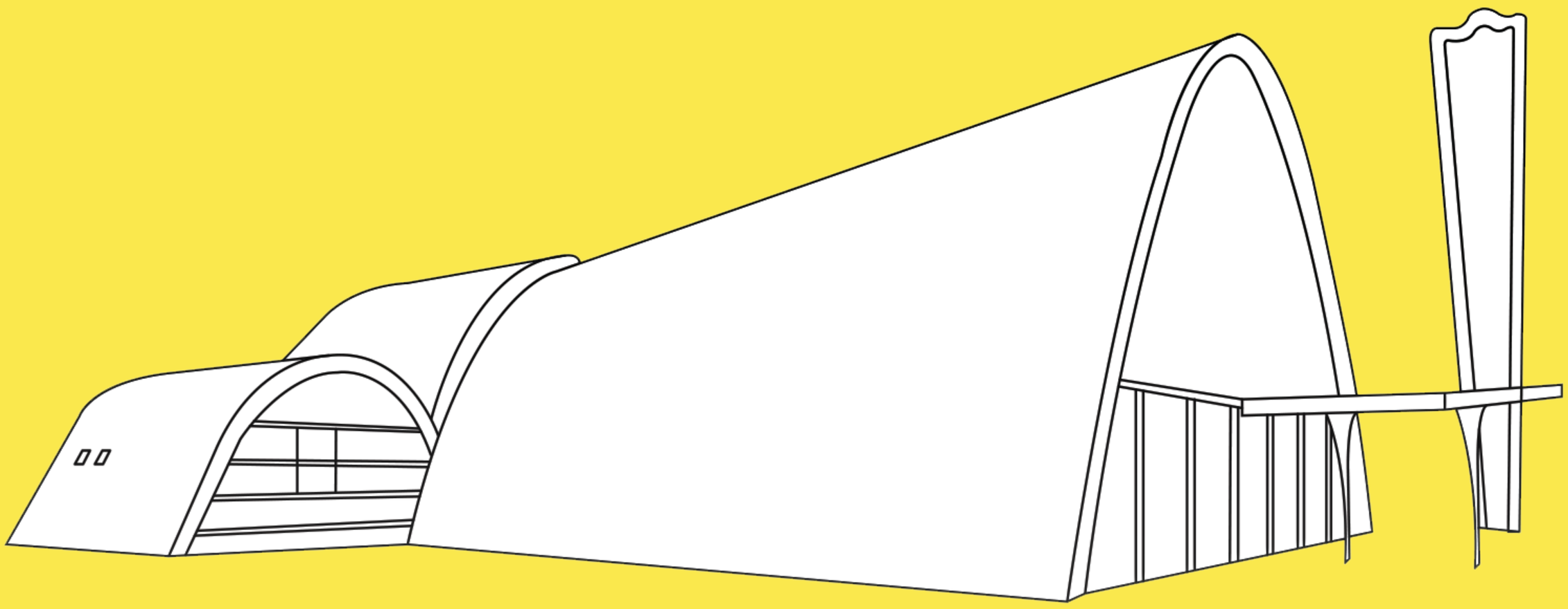
SOFTWARE ENGINEERING

A Modern Approach



MARCO TULIO VALENTE

Sample Version

Full text available at softengbook.org

Software Engineering:

A Modern Approach

Marco Tulio Valente

Version 1.3.2

Copyright © 2024 by Marco Tulio Valente.

All rights reserved. No part of this book may be reproduced in any form or by any means without written permission from the author.

The author assumes no responsibility for errors, omissions, or damages resulting from the use of the information contained in this book.

ISBN: 978-65-01-26467-7

Contents

Preface **1**

Preface to the Brazilian Edition **3**
Audience 5
Prerequisites 5
Author 6
Acknowledgments 6

1 Introduction **7**
1.1 Definition and Historical Context 7
1.2 Topics of Study 11
1.3 Classification of Software Systems 24
1.4 Upcoming Chapters 25
Bibliography 27
Exercises 27

2 Processes **29**
2.1 Introduction 29
2.2 Agile Manifesto 30
2.3 Extreme Programming 35
2.4 Scrum 49
2.5 Kanban 57
2.6 When Not to Use Agile Methods? 63
2.7 Other Iterative Methods 64
Bibliography 67
Exercises 68

3 Requirements **73**
3.1 Introduction 73

3.2 Requirements Engineering . . . 75
3.3 User Stories . . . 79
3.4 Use Cases . . . 85
3.5 Minimum Viable Product (MVP) . . . 90
3.6 A/B Testing . . . 96
Bibliography . . . 100
Exercises . . . 100

4 Models **103**
4.1 Introduction . . . 103
4.2 UML for Creating Sketches . . . 105
4.3 Class Diagrams . . . 109
4.4 Package Diagrams . . . 117
4.5 Sequence Diagrams . . . 118
4.6 Activity Diagrams . . . 121
Bibliography . . . 124
Exercises . . . 124

5 Design Principles **127**
5.1 Introduction . . . 127
5.2 Conceptual Integrity . . . 130
5.3 Information Hiding . . . 132
5.4 Cohesion . . . 137
5.5 Coupling . . . 138
5.6 SOLID and Other Design Principles . . . 143
5.7 Source Code Metrics . . . 157
Bibliography . . . 161
Exercises . . . 162

6 Design Patterns **167**
6.1 Introduction . . . 167
6.2 Factory . . . 170
6.3 Singleton . . . 172
6.4 Proxy . . . 174
6.5 Adapter . . . 177
6.6 Facade . . . 179
6.7 Decorator . . . 180
6.8 Strategy . . . 184
6.9 Observer . . . 185
6.10 Template Method . . . 188

6.11 Visitor . . . 189
6.12 Other Design Patterns . . . 192
6.13 When Not to Use Design Patterns? . . . 193
Bibliography . . . 195
Exercises . . . 195

7 Architecture **199**
7.1 Introduction . . . 199
7.2 Layered Architecture . . . 202
7.3 MVC Architecture . . . 204
7.4 Microservices . . . 210
7.5 Message-Oriented Architectures . . . 216
7.6 Publish/Subscribe Architecture . . . 218
7.7 Other Architectural Patterns . . . 220
7.8 Architectural Anti-Patterns . . . 221
Bibliography . . . 222
Exercises . . . 223

8 Testing **225**
8.1 Introduction . . . 225
8.2 Unit Testing . . . 228
8.3 Principles and Smells . . . 236
8.4 Test Coverage . . . 240
8.5 Testability . . . 244
8.6 Mocks . . . 247
8.7 Test-Driven Development . . . 253
8.8 Integration Tests . . . 258
8.9 End-to-End Tests . . . 260
8.10 Other Types of Testing . . . 261
Bibliography . . . 264
Exercises . . . 265

9 Refactoring **269**
9.1 Introduction . . . 269
9.2 Examples of Refactorings . . . 271
9.3 Refactoring Practice . . . 285
9.4 Automated Refactorings . . . 286
9.5 Code Smells . . . 289
Bibliography . . . 298
Exercises . . . 298

10 DevOps **303**
10.1 Introduction . . . 303
10.2 Version Control . . . 307
10.3 Continuous Integration . . . 312
10.4 Continuous Deployment . . . 317
Bibliography . . . 322
Exercises . . . 323

A Git **325**
A.1 Init & Clone . . . 325
A.2 Commit . . . 326
A.3 Add . . . 327
A.4 Status, Diff & Log . . . 328
A.5 Push & Pull . . . 329
A.6 Merge Conflicts . . . 331
A.7 Branches . . . 332
A.8 Remote Branches . . . 336
A.9 Pull Requests . . . 337
A.10 Squash . . . 339
A.11 Forks . . . 340
Bibliography . . . 341
Exercises . . . 341

Index **343**

Preface

This book is an English translation of a work originally published in early 2020. Initially intended for Brazilian students, we chose to write it in Portuguese to maximize accessibility and make an impact on software engineering education throughout our country.

Fortunately, the book has been a considerable success, surpassing our expectations by a wide margin. It is now adopted by numerous Brazilian universities in different states across the country. Although estimating exact figures is difficult, we have evidence that over 400 instructors have used or are currently using the book. We frequently receive messages with positive feedback from students, instructors, and industry professionals who are using the book to improve their skills in the field.

Thus, with the book firmly established in Brazil by late 2023, we decided to embark on an English translation project to reach a global audience. This decision was also motivated by requests from Brazilian colleagues working abroad. Furthermore, the Portuguese edition remains relevant and doesn't require significant updates.

For a deeper insight into the motivations behind writing the book, its organization, and features, we recommend that readers refer to the original preface, which we have reproduced in the following pages.

Website

The book has an open access version, available at https://softengbook.org. To report any errors, inconsistencies, or typos, please use the form at (link[1]).

Marco Tulio Valente

Belo Horizonte, March 19, 2024

[1] https://forms.gle/urrjFgGYgmKuJrfb9

Preface to the Brazilian Edition

> *The uselessness of prefaces is a commonplace in the history of prefaces, so I will be brief.* – Eduardo Giannetti

The idea of writing this book emerged in early 2019 when I was assigned to teach the software engineering course for the bachelor's degree in computer science at UFMG, Brazil. To prepare for the course, I began by analyzing well-known software engineering books. To my surprise, I found they had changed little since I had taken the course during my undergraduate studies more than 25 years earlier.

My goal was to find a book that would allow me, at the beginning of a class, to inform the students: "Today, we will study this topic, which corresponds to this chapter in the book." At the end of the class, I wanted to suggest to the students: "To better comprehend the topics we just covered, I suggest you do these exercises." Unfortunately, I couldn't find such a book. Instead, I had to conduct extensive research and read at least 15 books.

As a result, I created 600 slides covering the essential material for an undergraduate course in software engineering, particularly for programs that have only one course in the field. However, studying solely from slides does not provide the comprehensive learning experience obtained from reading a full and contextualized text.

Thus, I decided to transform the slides into my ideal textbook. It could also help other instructors who might face similar challenges when teaching an introductory software engineering course.

I would like to highlight the following points about the content, organization, and writing strategy of the book:

- It was written to be a modern book, covering techniques and principles widely used in software engineering today.

- Additionally, the book covers traditional techniques and principles. We believe it's important to know the history of the field, highlighting what has and has not worked. This perspective is relevant to the students' training, in our opinion.

- The book includes numerous examples and discussions of real cases. For this purpose, we created the "Real World" sections, where topics are illustrated with recent and interesting examples, from large software companies and from the scientific literature.

- The book was written in Portuguese, primarily because we aimed to contribute to Brazilian courses and students. However, we chose not to translate certain terms, such as refactoring, branches, and sprints, as we believe Brazilian developers use them in their day-to-day work without translation.

- Despite addressing modern topics, it is intended to be a timeless book. Therefore, techniques and technologies that have not yet stood the test of time are either not addressed or are addressed briefly. An example is the most recent frameworks and libraries for constructing web interfaces, which tend to change rapidly. We also avoided strongly coupling the book to any specific programming language. For instance, most code examples are presented in Java, but use a neutral syntax without using sophisticated or Java-specific resources.

- It is not an excessively long book with numerous chapters that, in practice, discuss topics not pertinent to undergraduate courses.

- Increasingly, software engineers have to write code. Today, there's no room for saying "I don't need to program because I'm an architect or analyst." Therefore, in the software design, testing, and refactoring chapters, we attempted to follow the recommendation of Linus Torvalds (creator of Linux): "Talk is cheap, but show me the code." Thus, these chapters include several source code examples illustrating problems and solutions in real systems.

- In writing this book, we followed the principles we advocate. Thus, it was written following several agile principles and practices. Each chapter was treated as a sprint; once completed, it was made available for use, critique, and suggestions. Similar to software development, we believe it's risky in

today's world to spend years writing a manuscript while locked in an office before making it public.

Audience

This book is intended for undergraduate students and was designed for curricula with a single 60-hour software engineering course. However, in conjunction with other books, it can be used in programs with two or more courses in the field.

It is also suitable for technical and technological degree programs. We have deliberately employed clear, accessible language to facilitate comprehension and widespread adoption. We also wrote the book with industry professionals in mind, especially those looking to improve their knowledge of modern software engineering topics.

Prerequisites

Readers are expected to have knowledge of basic programming concepts, algorithms, and data structures. Knowledge of object-oriented programming concepts is also recommended. Assuming a four-year undergraduate degree, the book is best suited for students in the latter half of their studies.

Marco Tulio Valente

Belo Horizonte, February 10, 2020

Author

Marco Tulio Valente holds a Ph.D. in Computer Science from the Federal University of Minas Gerais, Brazil, where he is an Associate Professor in the Department of Computer Science since 2009. Previously, he was an adjunct professor at PUC Minas for 11 years and a senior systems analyst at a telecommunication company for 4 years. He is the author of over 150 scientific papers and has supervised more than 30 master's theses and 15 doctoral dissertations.



Acknowledgments

I would like to thank the 30 students who have been part of the ASERG research group over the last decade. A substantial part of the book's content has its origin in the work of our group. Special thanks to Professors Eduardo Figueiredo and André Hora, my colleagues at DCC/UFMG, who reviewed some chapters of the book. Also, to Prof. José Luis Braga, who gave valuable suggestions to improve the text. I'd like to thank the students of the postgraduate course we offered in the second semester of 2019 who helped us to calibrate and refine the book's content—especially Ulisses Fernandes and Fábio Ferreira, for their careful review. Thanks also to Aline Brito, who helped convert the text into the final format, and to João Eduardo Montandon and Hudson Borges, for reviewing the appendix on Git. Lastly, thanks to Osvaldo and Cynthia Valente, for proofreading parts of the manuscript.

Index

A/B Testing, 96
    Control Version, 96
    Treatment Version, 96
A/B Tests, 321
ABC Systems, 24
Abstraction, 128
Adapter, 177
Agile Manifesto, 19, 30
Agile Methods, 19, 32, 63
Alexander, Christopher, 167
Amazon, 135
Architectural Anti-Patterns, 221
    Big Ball of Mud, 221
Architectural Patterns, 200
Automated Build, 47

Backlog
    Product, 51
    Sprint, 51
Bad Smells, 289
Batch Processing, 218
Beck, Kent, 206, 229, 262, 286, 314
Behavioral Test, 251
Bezos, Jeff, 135
Big Ball of Mud, 221
Big Design Upfront, 44
BitKeeper, 310
Blob, 292
Boehm, Barry, 65
Boundary Value Analysis, 263
Box, George, 104
BPMN (Business Process Model and Notation), 124
Branches, 332
    Master Branch, 332
Brechner, Eric, 60
Brooks' Law, 17
Brooks, Frederick, 9, 17, 74, 130
Builder, 192
Burndown Chart, 52

Canary Release, 321
CASE (Computer-Aided Software Engineering), 105
Casual Systems, 24
CBO (Coupling Between Objects), 160
CHAOS Report, 31
Chrome, browser, 319, 321
Chrysler, 38
Client/Server, 220
Cloud Computing, 213
Code Clones, 289
    Classification, 289
Code Duplication, 272
Code Review, 21, 46
Code Smells, 289
    Blob, 292
    Code Clones, 289
    Comments, 296
    Data Classes, 296
    Duplicate Code, 289
    Feature Envy, 292
    Global Variables, 293

God Class, 292
Large Classes, 291
Long Methods, 291
Methods With Many Parameters, 292
Mutable Objects, 294
Primitive Obsession, 294
Cohesion, 137
Collective Code Ownership, 46
Comments, 296
Commit Graphs, 334
Common Lisp, 190
Composition, 149
Conceptual Integrity, 130
Configuration Management, 17
Continuous Delivery, 319
Feature Flags, 320
Feature Toggles, 320
Continuous Deployment, 317
Continuous Integration, 47, 312
Best Practices, 314
Continuous Integration Servers, 315
Definition, 314
Trunk-Based Development, 316
When not to Use, 317
Continuous Integration Servers, 315
Conway, Melvin, 213
Conway's Law, 213
Coupling, 138
Evolutionary, 141
Logical, 141
Structural, 141
CVS, 308
Cyclomatic Complexity, 161

Daily Stand-Up, 53
Decorator, 180
Demeter, 150
Dependency Inversion, 146
Design for Change, 169
Design Patterns, 167
Adapter, 177
Behavioral, 169
Builder, 192
Creational, 169
Decorator, 180
Facade, 179
Factory, 170
Iterator, 192
Null Object, 198
Observer, 185
Proxy, 174
Singleton, 172
Strategy, 184
Structural, 169
Template Method, 188
Visitor, 189
Design Principles, 127, 129, 143
Demeter, 150
Dependency Inversion, 146
Interface Segregation, 145
Liskov Substitution, 154
Open-Closed, 152
Prefer Composition Over Inheritance, 148
Prefer Interfaces to Classes, 146
Single Responsibility, 144
SOLID, 143
Design Properties, 129
Design Sprint, 95
DevOps, 303
DevOpsDay, 306
Dijkstra, Edsger W., 202
Double Dispatch, 190
Dropbox, 93
Duplicate Code, 289

End-to-End Tests, 260
Selenium, 260
Equivalence Classes, 262
Eric Ries, 90
Events, 219
Extreme Programming, 35
Customer Representative, 38
Negotiated Scope Contracts, 48
Practices, 38
Principles, 37

Process Metrics, 49
Releases, 40
Values, 36
Working environment, 48

Facade, 179
Facebook, 99, 142, 236, 316, 318
Factory, 170
Farley, David, 306
Feature Envy, 292
Feature Flags, 320
Feature Toggles, 320
Flaky Tests, 237
Flowcharts, 124
Foote, Brian, 221
Formal Methods, 105
Forward Engineering, 106
Fowler, Martin, 88, 106, 109, 206, 271, 286, 314
Frameworks, 189

Gamma, Erich, 168, 229
Gang of Four, 148, 168
Getters, 135
Git, 310, 325
Add, 327
Branches, 332
Clone, 325
Commit, 326
Commit Graphs, 334
Diff, 329
Fork, 340
GitHub, 337, 340
HEAD, 335
Index, 327
Init, 325
Log, 329
Master, 332, 335
Merge, 334
Merge Conflicts, 331
Pull, 329
Pull Requests, 337
Push, 329
Remote Branches, 336
Squash, 339
Stage, 327
Status, 328
Tracked Files, 327
Working Directory, 327
GitHub, 310, 337, 340
Global Variables, 293
GMail, 208
Go, 150
Google, 10, 236, 243, 311, 316

Helm, Richard, 168
Humble, Jez, 306

IBM, 65
inception, 67
Incremental Design, 44
Information Hiding, 132
InfoSys, 221
Inheritance
of Classes, 148
of Interfaces, 148
Integration Conflicts, 312
Integration Hell, 313
Integration Tests, 258
Interface Segregation, 145
Inversion of Control, 189
Iterator, 192

JavaScript, 208
Johnson, Ralph, 168
JUnit, 229, 231

Kanban, 57
Little's Law, 62
WIP Calculation, 60
WIP Limits, 59
Work in Progress (WIP), 59
Kent Beck, 142
Knuth, Donald, 29

Larman, Craig, 88
Latency, 215
Laws of Software Evolution, 269
LCOM (Lack of Cohesion Between Methods), 158

Lead Time, 62
Lean Development, 90
Lean Startup, 90
Left-pad, 142
Left-pad episode, 142
Legacy Systems, 16
Lehman, Meir, 269
Lehman's Laws, 269
Lines of Code (LOC), 157
Linux, 10, 29, 201
Liskov Substitution, 154
Liskov, Barbara, 154
Little's Law, 62

Mainframes, 203
Martin, Robert, 143
McCabe's Complexity, 161
Mercurial, 308
Merge Conflicts, 312, 331
Merge Hell, 313
Message Oriented Architecture, 216
    Space decoupling, 216
    Temporal decoupling, 216
Meszaros, Gerard, 251
Meta, 142
Meyer, Bertrand, 24, 152
Microkernel, 201
Microservices, 210
    Conway's Law, 213
    Data Management, 214
    Monoliths, 210
    Scalability, 212
    Two-Phase Commit, 215
    When Not to Use, 215
Microsoft, 57, 99
Microsoft Research, 46
Millennium Bug, 16
Minimum Viable Product (MVP), 90
    Actionable Metrics, 92
    Design Sprint, 95
    Dropbox, 93
    Funnel Metrics, 92
    Prototyping, 94
    Vanity Metrics, 92
    Zappos, 92
Mission-Critical Systems, 24
Mockito, 249
Mocks, 247
    Dummy Objects, 251
    Fake Objects, 251
    Frameworks, 249
    Mockito, 249
Monoliths, 201, 210
MVC (Model-View-Controller), 204
MVP, 90

NATO Conference, 8
Negotiated Scope Contracts, 48
Netflix, 99
Null Object, 198

Object-Oriented Design, 129
Observer, 185
Opdyke, William, 270
Open-Closed, 152
Ousterhout, John, 127, 136, 162, 194, 285
Over-engineering, 24

Pair Programming, 45
Parnas, David, 132
Partial Failures, 212
Peer-to-Peer, 221
Petri Nets, 124
Pipes and Filters, 220
Planning Poker, 39
Prefer Composition Over Inheritance, 148
Prefer Interfaces to Classes, 146
Principle of Least Privilege, 150
Product Owner, 50
Professional Ethics, 22
Professional Practice, 22
Project Management, 17
    Brooks' Law, 17
Prototyping, 94
Proxy, 174
Publish/Subscribe, 218
    Broker, 219

Events, 219
Topics, 219

Rational Software, 65
Rational Unified Process (RUP), 65
Refactoring, 269
Automated, 286
Definition, 270
Extract Class, 281
Extract Method, 271
Flag Removal, 283
Method Inline, 275
Move Method, 276
Opportunistic, 285
Planned, 286
Preconditions, 287
Pull Up Method, 279
Push Down Method, 279
Remove Dead Code, 285
Renaming, 282
Replace Conditional with Polymorphism, 284
Variable Extraction, 283
Requirements, 73
Elicitation, 75
Functional, 12, 73
Non-Functional, 12, 73
Properties, 76
Requirements Engineering, 12, 75
Requirements Engineering, 12
Reuse
Black-Box, 150
White-Box, 150
Reverse Engineering, 107
Rust, 150

Scalability, 212
Horizontal, 212
Schwaber, Ken, 49
Scrum, 49
Burndown Chart, 52
Daily Stand-Up, 53
Definition of Done, 52
Product Backlog, 51
Product Owner, 50
Retrospective, 54
Roles, 50
Scrum Board, 52
Scrum Master, 50, 56
Sprint, 51
Sprint Backlog, 51
Sprint Planning, 51
Sprint Review, 53
Time-box, 54
Scrum Board, 52
Scrum Master, 50
Selenium, 260
Semantic Versioning, 17
Separation of Concerns, 137
Setters, 135
Single Dispatch, 191
Single Page Applications, 208
Single Responsibility, 144
Singleton, 172
Slacks, 42
Smalltalk, 204, 229
Software Architecture, 13, 199
Anti-Patterns, 221
Client/Server, 220
Layers, 202
Message Oriented, 216
Microservices, 210
MVC (Model-View-Controller), 204
Peer-to-Peer, 221
Pipes and Filters, 220
Publish/Subscribe, 218
Single Page Applications, 208
Styles, 200
Tanenbaum-Torvalds Debate, 201
Three Tiers, 203
Two Tiers, 204
Software Design, 12, 127
Software Economics, 23
Software Maintenance, 15
Adaptive, 16
Corrective, 16
Evolutionary, 16
Preventive, 16
Software Models, 20, 103

Software Process, 29
Software Processes, 18
Software Quality, 20
    External, 21
    Internal, 21
Software Testing, 46, 225
    Acceptance Tests, 263
    Alpha Tests, 264
    Behavioral, 251
    Beta Tests, 264
    Black-Box, 261
    Boundary Value Analysis, 263
    Coverage, 240
    End-to-End Tests, 260
    Equivalence Classes, 262
    Functional, 261
    Integration Tests, 258
    Manual Tests, 263
    Mocks, 247
    Structural, 261
    System Tests, 260
    Test Pyramid, 226
    Testability, 244
    Testing Non-Functional Requirements, 264
    Types of Tests, 226
    Unit Testing, 228
    White-Box, 261
Sommerville, Ian, 75
Source Code Metrics, 157
    CBO (Coupling Between Objects), 160
    Cohesion, 158
    Coupling, 160
    Cyclomatic Complexity, 161
    LCOM (Lack of Cohesion in Methods), 158
    Lines of Code (LOC), 157
    McCabe's Complexity, 161
    Size, 157
Spiral Model, 65
Sprint, 51
Sprint Planning, 51
Sprint Retrospective, 54
Sprint Review, 53
Squad, 55
Stack Overflow survey, 35
Stakeholders, 17
Startups, 90
Strategy, 184
Stubs, 177, 247, 251
Subversion, 308
Sutherland, Jeff, 49
SWEBOK, 11
System Administrators, 304
System Under Test (SUT), 235

Tanenbaum, Andrew, 201
Tanenbaum-Torvalds Debate, 201
Technical Debt, 297
Template Method, 188
Test Coverage, 240
    Branches, 243
    Function Calls, 243
    Functions, 243
    statements, 240
Test Driven Design (TDD), 46
Test Pyramid, 226
Testability, 244
Test-Driven Development (TDD), 46, 253
TeX, 29
THE, Operating System, 202
Thompson, Ken, 157
Throughput, 62
Time-box, 54
Torvalds, Linus, 29, 201, 310, 325
Toyota, 57, 90
Trunk-Based Development, 316
Two-Phase Commit, 215
Two-way Data Binding, 210

UML, 20, 105
    Activity Diagram, 121
    as a Blueprint, 106
    as a Programming Language, 106
    as a Sketch, 106
    Associations, 110
    Class Diagrams, 109

Dependencies, 115
Dynamic Diagrams, 108
Forward Engineering, 106
Inheritance, 115
Multiplicity, 111
Package Diagrams, 117
Reverse Engineering, 107
Sequence Diagram, 118
Static Diagrams, 108
Types of Diagrams, 108
Use Case Diagram, 88
UML Diagrams
Activities, 121
Classes, 109
Packages, 117
Sequence, 118
Use Cases, 88
Unified Process (UP), 65
Unit Testing, 228
assert, 231
Benefits, 235
Definitions, 234
Fixture, 231, 234
JUnit, 229, 231
Regressions, 235
Test Case, 234
Test Suite, 234
When to Write, 235
xUnit, 229
Unit Tests
FIRST Principles, 237
Flaky, 237
Number of Asserts, 239
Smells, 238
Unix, 308
Use Cases, 85
Actor, 85
Extensions, 85
Main Flow, 85
User Stories, 38, 79
Epic, 83
Personas, 82
Planning Poker, 39
Properties, 81
User Roles, 82
Velocity, 43
Writing Workshop, 82

Velocity, 43
Version Control, 307, 308
Version Control Systems, 307
Distributed, 308
Monorepos, 310
Multirepos, 310
Visitor, 189
Vlissides, John, 168
Vue.js, 208

Waterfall, 18, 33
Web System Tests, 260
WIP Limits, 59
Work in Progress (WIP), 59
Workspace, 48
Wrapper, 178

Yoder, Joseph, 221

This book was formatted using Pandoc to convert Markdown to LaTeX, which was then used to generate a PDF file. The fonts used are Bitstream Charter for text and `Beramono` for code, both at 11pt. The EPUB and HTML versions were also generated from the same Markdown files using Pandoc.