The Addison Wesley Signature Series

A Mike Cohn Signature Book

Essential Scrum

A Practical Guide to the Most Popular Agile Process

Kenneth S. Rubin

Forewords by Mike Cohn and Ron Jeffries

FREE SAMPLE CHAPTER

SHARE WITH OTHERS

Praise for *Essential Scrum*

"Agile coaches, you're gonna be happy with this book. Kenny Rubin has created an indispensable resource for us. Do you have a manager who just doesn't 'get it'? Hand them this book and ask them to flip to Chapter 3 for a complete explanation of how Scrum is less risky than plan-driven management. It's written just for them—in management-speak. Want to help the team come to a common understanding of Scrum? The visual icon language used throughout this book will help you help them. These are just two ways this book can aid you to coach Scrum teams. Use it well."

—Lyssa Adkins, Coach of Agile Coaches, Agile Coaching Institute; author, *Coaching Agile Teams*

"One of the best, most comprehensive descriptions of the core Scrum framework out there! *Essential Scrum* is for anyone—new to or experienced with Scrum—who's interested in the most important aspects of the process. Kenny does an excellent job of distilling the key tenets of the Scrum framework into a simple format with compelling visuals. As a Scrum coach for many teams, I continually reference the material for new ways to help teams that are learning and practicing the framework. I've seen Scrum continually misinterpreted and poorly implemented by big companies and tool vendors for more than ten years. Reading this book will help you get back to the basics and focus on what's important."

—Joe Balistrieri, Process Development Manager, Rockwell Automation

"Corporate IT leadership, which has been slow to embrace agile methods, would benefit immensely from giving a copy of this book to all of their project and delivery managers. Kenny Rubin has laid out in this book all the pragmatic business case and process materials needed for any corporate IT shop to successfully implement Scrum."

—John F. Bauer III, veteran of technical solution delivery in large corporate IT shops

"Kenny's extensive experience as a consultant, trainer, and past managing director of the Scrum Alliance is evident in this book. Along with providing the basics and introduction to Scrum, this book addresses the questions of masses—what happens to project managers? *Essential Scrum* helps us understand the big picture and guides how organization leaders can support and be involved with their Scrum teams for successful agile transformations."

—Sameer S. Bendre, CSM, PMP, Senior Consultant, 3i Infotech Inc.

“If you’re new to agile development or to Scrum, this book will give you a flying start. The examples and descriptions are clear and vivid, and you’ll often find yourself asking a question just before the book addresses that very topic.”

—Johannes Brodwall, Principal Solution Architect, Steria Norway

“Kenny’s well-structured explanations have a clarity to them that echoes the sensibilities of Smalltalk—the development environment with which he worked for years and from which both Scrum and Extreme Programming were born. This book pulls together a thorough set of agile management principles that really hit the mark and will no doubt guide you toward a more effective agile approach.”

—Rowan Bunning, Founder, Scrum WithStyle

“There are lots of books on Scrum these days, but this book takes a new angle: a reality check for software practitioners. Kenny uses real-world examples and clear illustrations to show what makes a solid foundation for successful agile development. Readers will understand the value of building quality in, and the reality that we can’t get everything right up front; we must work incrementally and learn as we go. It might have ‘Scrum’ in the title, but the book leverages effective practices from the larger agile universe to help managers and their teams succeed.”

—Lisa Crispin, coauthor, *Agile Testing*

“Kenny Rubin managed to write the book that I want everyone associated with Scrum development to read! He covers everything you’ll need to know about Scrum and more!”

—Martine Devos, European Scrum Pioneer and Certified Scrum Trainer

“I’ve reviewed a number of agile books in the past few years, so the question of ‘Do we really need another one?’ always comes to my mind. In the case of Kenny’s book, I very much believe the answer is ‘yes.’ Getting the benefit of different, experienced perspectives on commonly encountered and needed material is valuable. Kenny has one of those valuable perspectives. One unique aspect of the book is an interesting ‘iconography’—a new icon language for Scrum and agile that Kenny has created. I believe you’ll find value-added material in this book to expand your ideas for how Scrum can be applied.”

—Scott Duncan, Agile/Scrum coach and trainer

“Anyone who has had Scrum training or has been part of a Scrum team will find *Essential Scrum* to be a great follow-up read. It dives into the details of how to become more agile through implementing Scrum processes, and it explains exactly how to break down complex projects into manageable initiatives (or ‘sprints’). Kenny Rubin provides a wealth of relevant case studies on what worked—or what didn’t—in a variety of organizations. The simple layout and businesslike graphics make it easy to scan quickly and find specific topics. Any organization that is seeking to evolve from a traditional waterfall approach toward a more agile methodology will find *Essential Scrum* a definitive guidebook for the journey."

—Julia Frazier, product manager

"Developing software is hard. Adopting a new way of working while in a project is even harder. This book offers a bypass of many of the pitfalls and will accelerate a team's ability to produce business value and become successful with Scrum. I wish I had this kind of book when I started using Scrum."

—Geir Hedemark, Development Manager, Basefarm AS

"I am convinced that *Essential Scrum* will become the foundation reference for the next generation of Scrum practitioners. Not only is it the most comprehensive introduction to Scrum available today, but it is also extremely well written and easy on the eye with its fantastic new visual Scrum language. If that isn't enough, Kenny shares a range of his valuable personal insights and experiences that we can all certainly learn from."

—Ilan Goldstein, Agile Solutions Manager, Reed Elsevier

"Scrum is elegantly simple, yet deceptively complex. In *Essential Scrum,* Kenny Rubin provides us with a step-by-step guide to those complexities while retaining the essential simplicity. Real-world experiences coupled with enlightening illustrations make Scrum come to life. For senior managers and team members alike, this is a must-read book if you are starting or considering whether to implement Scrum in your organization. This will certainly be a book recommended to my students."

—John Hebley, Hebley & Associates

"Kenny unpacks a wealth of wisdom and knowledge in *Essential Scrum,* providing valuable and comprehensive insights to the practical application of agile/Scrum. Whether you're new to agile or are looking to reach a greater maturity of continuous improvement in your organization, this is a definitive handbook for your toolbox."

—David Luzquiños, Head of Agile Enablement, Agile Coach, Betfair

"Kenny Rubin continues to provide clarity and insight into adopting agile in a pragmatic way. In one hand he holds the formal or ideal Scrum definition, and in the other, the pragmatic application of it. He brings the wisdom of his workshops and years of experience to the table for you to read in his latest book. If you are about to start out on your agile adoption journey or are seeking guidance midcourse, grab a copy."

—Cuan Mulligan, freelance coactive Agile coach

“A decade after publication of the first Scrum books, it is time to combine the essential aspects of the Scrum framework with the practical experiences and approaches of the last ten years. Kenny Rubin does so in a satisfying and nondogmatic way. The reader gets a pragmatic look at Scrum and learns when and how to best apply Scrum to achieve business benefits.”

—Yves Stalgies, Ph.D., Director IT, www.etracker.com

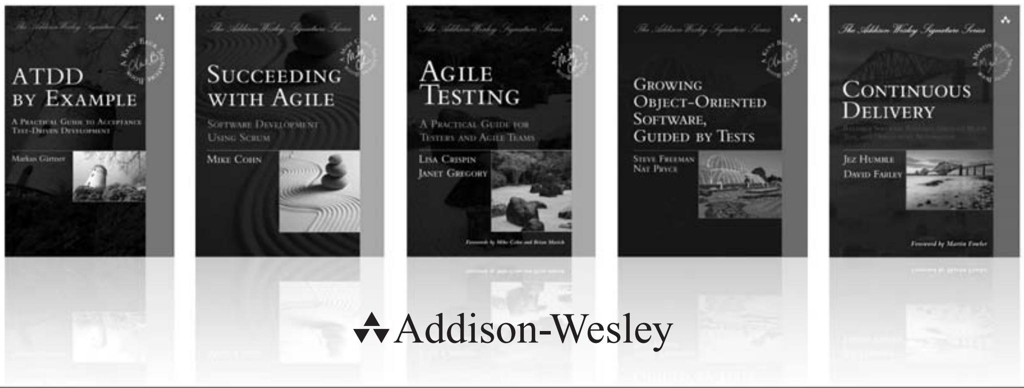
“Adoption of Scrum is most successful when everyone involved—even peripherally—with product development has a good understanding of the fundamentals. *Essential Scrum* provides an ideal overview of both the big picture and the details in an accessible style. It is sure to become a standard reference.”

—Kevin Tureski, Principal, Kevin Tureski Consulting

ESSENTIAL SCRUM

The Addison-Wesley Signature Series

Kent Beck, Mike Cohn, and Martin Fowler, Consulting Editors



Visit **informit.com/awss** for a complete list of available products.

The Addison-Wesley Signature Series provides readers with practical and authoritative information on the latest trends in modern technology for computer professionals. The series is based on one simple premise: Great books come from great authors. Titles in the series are personally chosen by expert advisors, world-class authors in their own right. These experts are proud to put their signatures on the covers, and their signatures ensure that these thought leaders have worked closely with authors to define topic coverage, book scope, critical content, and overall uniqueness. The expert signatures also symbolize a promise to our readers: You are reading a future classic.

Make sure to connect with us!
informit.com/socialconnect

Addison Wesley | informIT.com THE TRUSTED TECHNOLOGY LEARNING SOURCE | Safari Books Online

ALWAYS LEARNING PEARSON

Essential Scrum

A Practical Guide to the Most Popular Agile Process

Kenneth S. Rubin

Addison-Wesley

Upper Saddle River, NJ • Boston • Indianapolis • San Francisco
New York • Toronto • Montreal • London • Munich • Paris • Madrid
Capetown • Sydney • Tokyo • Singapore • Mexico City

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and the publisher was aware of a trademark claim, the designations have been printed with initial capital letters or in all capitals.

The author and publisher have taken care in the preparation of this book, but make no expressed or implied warranty of any kind and assume no responsibility for errors or omissions. No liability is assumed for incidental or consequential damages in connection with or arising out of the use of the information or programs contained herein.

The publisher offers excellent discounts on this book when ordered in quantity for bulk purchases or special sales, which may include electronic versions and/or custom covers and content particular to your business, training goals, marketing focus, and branding interests. For more information, please contact:

U.S. Corporate and Government Sales
(800) 382-3419
corpsales@pearsontechgroup.com

For sales outside the United States, please contact:

International Sales
international@pearson.com

Visit us on the Web: informit.com/aw

Library of Congress Cataloging-in-Publication Data

Rubin, Kenneth S.
Essential Scrum : a practical guide to the most popular agile process / Kenneth S. Rubin.
p. cm.
Includes bibliographical references and index.
ISBN 978-0-13-704329-3 (pbk. : alk. paper)—ISBN 0-13-704329-5 (pbk. : alk. paper)
1. Scrum (Computer software development) 2. Agile software development. 3. Project management. I. Title.
QA76.76.D47R824 2012
005.1—dc23

2012010892

Copyright © 2013 Pearson Education, Inc.

Agile visual icon language copyright © Kenneth S. Rubin and used with permission.

All rights reserved. Printed in the United States of America. This publication is protected by copyright, and permission must be obtained from the publisher prior to any prohibited reproduction, storage in a retrieval system, or transmission in any form or by any means, electronic, mechanical, photocopying, recording, or likewise. To obtain permission to use material from this work, please submit a written request to Pearson Education, Inc., Permissions Department, One Lake Street, Upper Saddle River, New Jersey 07458, or you may fax your request to (201) 236-3290.

ISBN-13: 978-0-13-704329-3
ISBN-10: 0-13-704329-5
Text printed in the United States on recycled paper at Edwards Brothers Malloy in Ann Arbor, Michigan.
First printing, July 2012

To my wife, Jenine, for all your loving support
To my sons, Jonah and Asher, for inspiring me
To my father, Manny, for teaching me the value of hard work
To my mother, Joyce, for showing me what real courage looks like
(may her memory be a blessing)

This page intentionally left blank

Contents

List of Figures xxv

Foreword by Mike Cohn xxxi

Foreword by Ron Jeffries xxxiii

Preface xxxv

Acknowledgments xxxix

About the Author xliii

Chapter 1 Introduction 1

- What Is Scrum? 1
- Scrum Origins 3
- Why Scrum? 4
- Genomica Results 4
- Can Scrum Help You? 5
  - Complex Domain 8
  - Complicated Domain 8
  - Simple Domain 8
  - Chaotic Domain 9
  - Disorder 9
  - Interrupt-Driven Work 9
- Closing 10

PART I Core Concepts 11

Chapter 2 Scrum Framework 13

- Overview 13
- Scrum Roles 14
  - Product Owner 15
  - ScrumMaster 16
  - Development Team 16
- Scrum Activities and Artifacts 16
  - Product Backlog 18

Sprints 20
Sprint Planning 21
Sprint Execution 23
Daily Scrum 23
Done 25
Sprint Review 26
Sprint Retrospective 27
Closing 28

Chapter 3 Agile Principles 29
Overview 29
Variability and Uncertainty 32
Embrace Helpful Variability 32
Employ Iterative and Incremental Development 33
Leverage Variability through Inspection, Adaptation, and Transparency 35
Reduce All Forms of Uncertainty Simultaneously 36
Prediction and Adaptation 37
Keep Options Open 37
Accept That You Can’t Get It Right Up Front 38
Favor an Adaptive, Exploratory Approach 39
Embrace Change in an Economically Sensible Way 40
Balance Predictive Up-Front Work with Adaptive Just-in-Time Work 43
Validated Learning 44
Validate Important Assumptions Fast 45
Leverage Multiple Concurrent Learning Loops 45
Organize Workflow for Fast Feedback 46
Work in Process (WIP) 48
Use Economically Sensible Batch Sizes 48
Recognize Inventory and Manage It for Good Flow 49
Focus on Idle Work, Not Idle Workers 51
Consider Cost of Delay 52
Progress 54
Adapt to Real-Time Information and Replan 54
Measure Progress by Validating Working Assets 54
Focus on Value-Centric Delivery 55
Performance 56
Go Fast but Never Hurry 56
Build In Quality 56
Employ Minimally Sufficient Ceremony 57
Closing 58

Chapter 4 Sprints 61
Overview 61
Timeboxed 62
Establishes a WIP Limit 62
Forces Prioritization 62
Demonstrates Progress 62
Avoids Unnecessary Perfectionism 63
Motivates Closure 63
Improves Predictability 64
Short Duration 64
Ease of Planning 64
Fast Feedback 64
Improved Return on Investment 65
Bounded Error 65
Rejuvenated Excitement 65
Frequent Checkpoints 66
Consistent Duration 67
Cadence Benefits 67
Simplifies Planning 68
No Goal-Altering Changes 69
What Is a Sprint Goal? 69
Mutual Commitment 69
Change versus Clarification 69
Consequences of Change 70
Being Pragmatic 72
Abnormal Termination 72
Definition of Done 74
What Is the Definition of Done? 74
Definition of Done Can Evolve Over Time 76
Definition of Done versus Acceptance Criteria 77
Done versus Done-Done 77
Closing 78

Chapter 5 Requirements and User Stories 79
Overview 79
Using Conversations 81
Progressive Refinement 82
What Are User Stories? 83
Card 83
Conversation 84
Confirmation 85

Level of Detail 86
INVEST in Good Stories 88
Independent 88
Negotiable 89
Valuable 90
Estimatable 91
Sized Appropriately (Small) 92
Testable 92
Nonfunctional Requirements 93
Knowledge-Acquisition Stories 93
Gathering Stories 95
User-Story-Writing Workshop 95
Story Mapping 96
Closing 98

Chapter 6 Product Backlog 99
Overview 99
Product Backlog Items 100
Good Product Backlog Characteristics 101
Detailed Appropriately 101
Emergent 102
Estimated 102
Prioritized 103
Grooming 104
What Is Grooming? 104
Who Does the Grooming? 105
When Does Grooming Take Place? 106
Definition of Ready 108
Flow Management 110
Release Flow Management 110
Sprint Flow Management 111
Which and How Many Product Backlogs? 112
What Is a Product? 113
Large Products—Hierarchical Backlogs 114
Multiple Teams—One Product Backlog 115
One Team—Multiple Products 117
Closing 118

Chapter 7 Estimation and Velocity 119
Overview 119
What and When We Estimate 120
Portfolio Backlog Item Estimates 121

Product Backlog Estimates 121
Task Estimates 122
PBI Estimation Concepts 123
Estimate as a Team 123
Estimates Are Not Commitments 124
Accuracy versus Precision 125
Relative Size Estimation 125
PBI Estimation Units 128
Story Points 128
Ideal Days 128
Planning Poker 129
Estimation Scale 130
How to Play 131
Benefits 133
What Is Velocity? 133
Calculate a Velocity Range 134
Forecasting Velocity 135
Affecting Velocity 135
Misusing Velocity 137
Closing 138

Chapter 8 Technical Debt 139
Overview 139
Consequences of Technical Debt 141
Unpredictable Tipping Point 142
Increased Time to Delivery 142
Significant Number of Defects 142
Rising Development and Support Costs 142
Product Atrophy 143
Decreased Predictability 143
Underperformance 143
Universal Frustration 144
Decreased Customer Satisfaction 144
Causes of Technical Debt 144
Pressure to Meet a Deadline 144
Attempting to Falsely Accelerate Velocity 145
Myth: Less Testing Can Accelerate Velocity 145
Debt Builds on Debt 147
Technical Debt Must Be Managed 148
Managing the Accrual of Technical Debt 149
Use Good Technical Practices 149
Use a Strong Definition of Done 149
Properly Understand Technical Debt Economics 150

Making Technical Debt Visible 153
Make Technical Debt Visible at the Business Level 153
Make Technical Debt Visible at the Technical Level 154
Servicing the Technical Debt 155
Not All Technical Debt Should Be Repaid 157
Apply the Boy Scout Rule (Service Debt When You Happen Upon It) 158
Repay Technical Debt Incrementally 159
Repay the High-Interest Technical Debt First 160
Repay Technical Debt While Performing Customer-Valuable Work 160
Closing 162

PART II Roles 163

Chapter 9 Product Owner 165
Overview 165
Principal Responsibilities 166
Manage Economics 167
Participate in Planning 168
Groom the Product Backlog 169
Define Acceptance Criteria and Verify That They Are Met 169
Collaborate with the Development Team 170
Collaborate with the Stakeholders 171
Characteristics/Skills 171
Domain Skills 171
People Skills 172
Decision Making 173
Accountability 173
A Day in the Life 174
Who Should Be a Product Owner? 176
Internal Development 176
Commercial Development 177
Outsourced Development Project 180
Component Development 180
Product Owner Combined with Other Roles 181
Product Owner Team 182
Product Owner Proxy 183
Chief Product Owner 183
Closing 184

Chapter 10 ScrumMaster 185
Overview 185
Principal Responsibilities 185
Coach 185
Servant Leader 186
Process Authority 186
Interference Shield 187
Impediment Remover 187
Change Agent 187
Characteristics/Skills 188
Knowledgeable 188
Questioning 188
Patient 189
Collaborative 189
Protective 189
Transparent 189
A Day in the Life 190
Fulfilling the Role 191
Who Should Be a ScrumMaster? 191
Is ScrumMaster a Full-Time Job? 192
ScrumMaster Combined with Other Roles 192
Closing 193

Chapter 11 Development Team 195
Overview 195
Role-Specific Teams 195
Principal Responsibilities 196
Perform Sprint Execution 196
Inspect and Adapt Each Day 197
Groom the Product Backlog 197
Plan the Sprint 197
Inspect and Adapt the Product and Process 197
Characteristics/Skills 198
Self-Organizing 198
Cross-Functionally Diverse and Sufficient 200
T-Shaped Skills 201
Musketeer Attitude 203
High-Bandwidth Communications 204
Transparent Communication 205
Right-Sized 206
Focused and Committed 207

Working at a Sustainable Pace 208
Long-Lived 209
Closing 211

Chapter 12 Scrum Team Structures 213
Overview 213
Feature Teams versus Component Teams 213
Multiple-Team Coordination 218
Scrum of Scrums 218
Release Train 220
Closing 223

Chapter 13 Managers 225
Overview 225
Fashioning Teams 227
Define Boundaries 227
Provide a Clear Elevating Goal 228
Form Teams 228
Change Team Composition 229
Empower Teams 230
Nurturing Teams 231
Energize People 231
Develop Competence 231
Provide Functional-Area Leadership 232
Maintain Team Integrity 233
Aligning and Adapting the Environment 233
Promote Agile Values 233
Remove Organizational Impediments 234
Align Internal Groups 234
Align Partners 234
Managing Value-Creation Flow 235
Take a Systems Perspective 235
Manage Economics 236
Monitor Measures and Reports 236
Project Managers 237
Project Management Responsibilities on a Scrum Team 237
Retaining a Separate Project Manager Role 239
Closing 243

PART III Planning 245

Chapter 14 Scrum Planning Principles 247
- Overview 247
- Don't Assume We Can Get the Plans Right Up Front 248
- Up-Front Planning Should Be Helpful without Being Excessive 248
- Keep Planning Options Open Until the Last Responsible Moment 249
- Focus More on Adapting and Replanning Than on Conforming to a Plan 249
- Correctly Manage the Planning Inventory 251
- Favor Smaller and More Frequent Releases 252
- Plan to Learn Fast and Pivot When Necessary 254
- Closing 255

Chapter 15 Multilevel Planning 257
- Overview 257
- Portfolio Planning 259
- Product Planning (Envisioning) 259
  - Vision 259
  - High-Level Product Backlog 259
  - Product Roadmap 260
- Release Planning 261
- Sprint Planning 264
- Daily Planning 264
- Closing 265

Chapter 16 Portfolio Planning 267
- Overview 267
  - Timing 267
  - Participants 268
  - Process 268
- Scheduling Strategies 270
  - Optimize for Lifecycle Profits 270
  - Calculate Cost of Delay 271
  - Estimate for Accuracy, Not Precision 274
- Inflow Strategies 275
  - Apply the Economic Filter 275
  - Balance the Arrival Rate with the Departure Rate 276
  - Quickly Embrace Emergent Opportunities 278
  - Plan for Smaller, More Frequent Releases 279

Outflow Strategies 280
Focus on Idle Work, Not Idle Workers 281
Establish a WIP Limit 281
Wait for a Complete Team 282
In-Process Strategies 283
Use Marginal Economics 283
Closing 285

Chapter 17 Envisioning (Product Planning) 287
Overview 287
Timing 287
Participants 288
Process 290
SR4U Example 290
Visioning 291
High-Level Product Backlog Creation 294
Product Roadmap Definition 295
Other Activities 298
Economically Sensible Envisioning 299
Target a Realistic Confidence Threshold 300
Focus on a Short Horizon 302
Act Quickly 302
Pay for Validated Learning 303
Use Incremental/Provisional Funding 304
Learn Fast and Pivot (aka Fail Fast) 305
Closing 306

Chapter 18 Release Planning (Longer-Term Planning) 307
Overview 307
Timing 308
Participants 308
Process 309
Release Constraints 311
Fixed Everything 311
Fixed Scope and Date 312
Fixed Scope 313
Fixed Date 313
Variable Quality 314
Updating Constraints 314
Grooming the Product Backlog 315
Refine Minimum Releasable Features (MRFs) 316

Sprint Mapping (PBI Slotting) 316
Fixed-Date Release Planning 318
Fixed-Scope Release Planning 323
Calculating Cost 325
Communicating 326
    Communicating Progress on a Fixed-Scope Release 327
    Communicating Progress on a Fixed-Date Release 329
Closing 330

PART IV Sprinting 333

Chapter 19 Sprint Planning 335
Overview 335
    Timing 335
    Participants 335
    Process 336
Approaches to Sprint Planning 338
    Two-Part Sprint Planning 338
    One-Part Sprint Planning 339
Determining Capacity 340
    What Is Capacity? 340
    Capacity in Story Points 342
    Capacity in Effort-Hours 342
Selecting Product Backlog Items 343
Acquiring Confidence 344
Refine the Sprint Goal 346
Finalize the Commitment 346
Closing 346

Chapter 20 Sprint Execution 347
Overview 347
    Timing 347
    Participants 348
    Process 348
Sprint Execution Planning 349
Flow Management 349
    Parallel Work and Swarming 350
    Which Work to Start 352
    How to Organize Task Work 352

What Work Needs to Be Done? 353
Who Does the Work? 354
Daily Scrum 354
Task Performance—Technical Practices 355
Communicating 356
Task Board 356
Sprint Burndown Chart 357
Sprint Burnup Chart 359
Closing 360

Chapter 21 Sprint Review 363
Overview 363
Participants 364
Prework 365
Determine Whom to Invite 366
Schedule the Activity 366
Confirm That the Sprint Work Is Done 367
Prepare for the Demonstration 368
Determine Who Does What 368
Approach 368
Summarize 369
Demonstrate 370
Discuss 371
Adapt 371
Sprint Review Issues 372
Sign-offs 372
Sporadic Attendance 372
Large Development Efforts 373
Closing 373

Chapter 22 Sprint Retrospective 375
Overview 375
Participants 377
Prework 378
Define the Retrospective Focus 378
Select the Exercises 379
Gather Objective Data 379
Structure the Retrospective 380
Approach 380
Set the Atmosphere 382
Share Context 382

Identify Insights 385
Determine Actions 387
Close the Retrospective 390
Follow Through 391
Sprint Retrospective Issues 392
Closing 393

Chapter 23 The Path Forward 395
There Is No End State 395
Discover Your Own Path 396
Sharing Best Practices 396
Using Scrum to Discover the Path Forward 397
Get Going! 398

Glossary 401

References 423

Index 427

This page intentionally left blank

LIST OF FIGURES

Figure 1.1	Agile development overview	2
Figure 1.2	Scrum benefits	6
Figure 1.3	Cynefin framework	7
Figure 2.1	Scrum practices	14
Figure 2.2	Scrum roles	15
Figure 2.3	Scrum framework	17
Figure 2.4	Product backlog	19
Figure 2.5	Product backlog grooming	19
Figure 2.6	Product backlog item sizes	20
Figure 2.7	Sprint characteristics	21
Figure 2.8	Sprint planning	21
Figure 2.9	Sprint backlog	22
Figure 2.10	Sprint execution	23
Figure 2.11	Daily scrum	24
Figure 2.12	Sprint results (potentially shippable product increment)	25
Figure 2.13	Sprint review	27
Figure 2.14	Sprint retrospective	27
Figure 3.1	Waterfall process	30
Figure 3.2	Categorization of principles	31
Figure 3.3	Defined process	32
Figure 3.4	Scrum uses iterative and incremental development.	34
Figure 3.5	Scrum process model	36
Figure 3.6	Make decisions at the last responsible moment.	38
Figure 3.7	Plan-driven requirements acquisition relative to product knowledge	39
Figure 3.8	Historical cost of exploration	40
Figure 3.9	Significant late cost of change with sequential development	41
Figure 3.10	Self-fulfilling prophecy	42
Figure 3.11	Flattening the cost-of-change curve	43
Figure 3.12	Balancing predictive and adaptive work	44
Figure 3.13	Learning loop pattern	46
Figure 3.14	Component integration	47
Figure 3.15	How utilization affects queue size (delay)	52

Figure 3.16	Deliver high-value features sooner.	55
Figure 3.17	Ceremony scale	58
Figure 4.1	Sprints are the skeleton of the Scrum framework.	61
Figure 4.2	The benefits of timeboxing	63
Figure 4.3	The benefits of short-duration sprints	64
Figure 4.4	Excitement over time	65
Figure 4.5	Checkpoint comparison	66
Figure 4.6	Cumulative investment at different states	71
Figure 4.7	Deciding on the next sprint length after sprint termination	73
Figure 5.1	Scrum uses placeholders for requirements.	81
Figure 5.2	A user story template and card	83
Figure 5.3	User story with additional data attached	84
Figure 5.4	User story conditions of satisfaction	85
Figure 5.5	User story abstraction hierarchy	87
Figure 5.6	Example epic	87
Figure 5.7	Example theme	88
Figure 5.8	Highly dependent stories	89
Figure 5.9	Example technical story	90
Figure 5.10	Undesirable technical story	91
Figure 5.11	Nonfunctional requirements	93
Figure 5.12	Knowledge-acquisition story	94
Figure 5.13	Story map	97
Figure 6.1	The product backlog is at the heart of the Scrum framework.	99
Figure 6.2	Product backlog items	100
Figure 6.3	Product backlog items are different sizes.	102
Figure 6.4	Product backlog items are estimated.	103
Figure 6.5	Product backlog items are prioritized.	104
Figure 6.6	Grooming reshapes the product backlog.	105
Figure 6.7	Grooming is a collaborative effort.	106
Figure 6.8	Outside-of-primary-flow grooming with sequential projects	107
Figure 6.9	When grooming happens	108
Figure 6.10	Definition of ready	109
Figure 6.11	Release-level view of the product backlog	111
Figure 6.12	The product backlog as a pipeline of requirements	112
Figure 6.13	The product backlog is associated with the product.	113
Figure 6.14	Hierarchical product backlogs	115
Figure 6.15	Team-specific view of the product backlog	116
Figure 6.16	Scenarios for multiple product backlogs	117

Figure 7.1	The relationship among size, velocity, and duration	120
Figure 7.2	What and when we estimate	121
Figure 7.3	Product backlog item estimating concepts	123
Figure 7.4	The full Scrum team participates in estimation.	124
Figure 7.5	Effect of committing on estimates	124
Figure 7.6	Effort versus accuracy when estimating	126
Figure 7.7	Relative size estimation	126
Figure 7.8	Absolute versus relative size estimation	127
Figure 7.9	Planning Poker concepts	129
Figure 7.10	Planning Poker uses binning.	130
Figure 7.11	Innolution Planning Poker cards	131
Figure 7.12	Calculating and using a velocity range	134
Figure 7.13	A team's velocity over time	136
Figure 7.14	The effect of overtime on velocity (based on a figure from Cook 2008)	137
Figure 8.1	Consequences of technical debt	141
Figure 8.2	Cost-of-change curve affected by technical debt	143
Figure 8.3	Pressure to meet a deadline can lead to technical debt.	145
Figure 8.4	Accruing technical debt to meet unreasonable fixed scope and date	146
Figure 8.5	The myth, reality, and good practice of how testing affects velocity	146
Figure 8.6	As technical debt increases, velocity decreases.	147
Figure 8.7	Activities for managing technical debt	148
Figure 8.8	Example technical debt economic analysis	150
Figure 8.9	Ways to make technical debt visible at the technical level	154
Figure 8.10	Approaches for servicing technical debt	156
Figure 8.11	A technique for managing technical debt when using Scrum	161
Figure 9.1	The product owner faces two directions simultaneously.	165
Figure 9.2	Principal product owner responsibilities	166
Figure 9.3	The product owner manages economics.	167
Figure 9.4	Comparison of customer or business engagement over time	170
Figure 9.5	Product owner characteristics	172
Figure 9.6	A day in the life of a product owner	174
Figure 9.7	Example of a product owner on internal development	177
Figure 9.8	Example of a product owner on commercial development	178
Figure 9.9	Pragmatic Marketing framework	179
Figure 9.10	Example of a product owner on outsourced development	180
Figure 9.11	Example of a product owner on component development	181
Figure 9.12	Same person as product owner of more than one Scrum team	182
Figure 9.13	Hierarchical product owner role	184

Figure 10.1 Principal ScrumMaster responsibilities 186
Figure 10.2 ScrumMaster characteristics 188
Figure 10.3 A day in the life of a ScrumMaster 190
Figure 10.4 Same person as ScrumMaster of more than one team 193

Figure 11.1 Development team responsibilities with respect to Scrum activities 196
Figure 11.2 Development team characteristics 198
Figure 11.3 Flocking isn't the result of top-down planning. 199
Figure 11.4 Flocking: simple rules and frequent feedback 200
Figure 11.5 Team diversity 201
Figure 11.6 T-shaped skills 202
Figure 11.7 Team members must act as if they are all in the same boat. 204
Figure 11.8 The cost of multitasking 208
Figure 11.9 Sustainable pace over time 209

Figure 12.1 One product and multiple component teams 214
Figure 12.2 Two products and multiple component teams 215
Figure 12.3 Combined feature team and component teams 217
Figure 12.4 Scrum of scrums 219
Figure 12.5 Release train structure 221

Figure 13.1 Greatest concerns about adopting agile 225
Figure 13.2 Functional manager responsibilities in a Scrum organization 226
Figure 13.3 Managers define the boundaries. 227
Figure 13.4 Functional managers collectively create Scrum teams. 228
Figure 13.5 Teams rarely have fully connected communication channels. 240
Figure 13.6 Teams frequently form collaboration clusters. 241
Figure 13.7 Funneling coordination through a project or program manager 242
Figure 13.8 Project manager on complex, multiparty development 243

Figure 14.1 Scrum planning principles 247
Figure 14.2 Big up-front Gantt chart 250
Figure 14.3 When the map and the terrain don't agree, believe the terrain. 251
Figure 14.4 Single-release economics 253
Figure 14.5 Multi-release economics 253

Figure 15.1 Different levels of planning 257
Figure 15.2 Scrum Alliance website product roadmap 261
Figure 15.3 A release line in the product backlog 262

Figure 15.4 Product roadmap releases mapped to the product backlog 263
Figure 15.5 A release can encompass one or more sprints. 263
Figure 15.6 Each sprint has a sprint backlog. 264
Figure 15.7 Hierarchical Scrum planning 266

Figure 16.1 Portfolio-planning activity 268
Figure 16.2 Portfolio-planning strategies 269
Figure 16.3 Cost-of-delay profiles 273
Figure 16.4 Applying the economic filter 276
Figure 16.5 Balancing inflow and outflow in the portfolio backlog 277
Figure 16.6 The value of many emergent opportunities decays rapidly. 279
Figure 16.7 Large products in the portfolio backlog create a convoy. 280
Figure 16.8 Teams are the unit of capacity for establishing the product WIP limit. 282
Figure 16.9 In-process product decision flow based on marginal economics 284

Figure 17.1 Envisioning is an ongoing activity. 288
Figure 17.2 Envisioning (product-planning) activity 289
Figure 17.3 Areas of stakeholder value 292
Figure 17.4 Fixed, periodic releases 296
Figure 17.5 SmartReview4You product roadmap 297
Figure 17.6 SR4U knowledge-acquisition sprint storyboard 298
Figure 17.7 Guidelines for economically sensible envisioning 300
Figure 17.8 Consequences of setting the confidence threshold bar too high 301
Figure 17.9 Decision making under the illusion of certainty 303
Figure 17.10 Incremental/provisional funding 304

Figure 18.1 Different release cadences 307
Figure 18.2 When release planning happens 309
Figure 18.3 Release-planning activity 310
Figure 18.4 Fixed date and fixed scope playing a game of chicken 312
Figure 18.5 Mapping product backlog items to sprints 317
Figure 18.6 Sprint calendar for SR4U Release 1.0 319
Figure 18.7 Product backlog ready for release planning 321
Figure 18.8 Determining the range of features on a fixed-date release 322
Figure 18.9 Location of must-have features relative to the range of deliverable features 322
Figure 18.10 Results of fixed-scope planning 325
Figure 18.11 Fixed-scope-release burndown chart 327
Figure 18.12 Fixed-scope-release burnup chart 328

Figure 18.13	Variable-scope-release burnup chart	329
Figure 18.14	Fixed-date-release burnup chart (with inverted product backlog)	330
Figure 19.1	When sprint planning happens	336
Figure 19.2	Sprint-planning activity	337
Figure 19.3	Two-part sprint-planning approach	339
Figure 19.4	One-part sprint-planning approach	340
Figure 19.5	Development team capacity in a sprint	341
Figure 19.6	Sprint backlog showing PBIs and task plan	345
Figure 20.1	When sprint execution happens	347
Figure 20.2	Sprint execution activity	348
Figure 20.3	Cost of multitasking	350
Figure 20.4	Mini waterfall during sprint execution—a bad idea	352
Figure 20.5	Subset of Extreme Programming technical practices	355
Figure 20.6	Example task board	356
Figure 20.7	Sprint burndown chart	358
Figure 20.8	Sprint burndown chart with trend lines	359
Figure 20.9	Sprint burnup chart	360
Figure 21.1	When the sprint review happens	363
Figure 21.2	Sprint review prework	366
Figure 21.3	Sprint review activity	369
Figure 22.1	Edward Bear illustrating the need for a retrospective	376
Figure 22.2	When the sprint retrospective happens	376
Figure 22.3	Sprint retrospective prework	378
Figure 22.4	Sprint retrospective activity	381
Figure 22.5	Aligning perspectives to create a shared context	383
Figure 22.6	Sprint event timeline	384
Figure 22.7	Emotions seismograph	385
Figure 22.8	Retrospective insight card wall	386
Figure 22.9	Insight cards clustered into similarity groups	386
Figure 22.10	Insight cards placed into predetermined groups	387
Figure 22.11	Example of dot voting	388
Figure 22.12	Sprint retrospective issues	391

FOREWORD BY MIKE COHN

I had lunch today at a Burger King. A sign on the wall proclaimed the restaurant the "Home of the Whopper" and then proceeded to tell me there were over a million different ways to order a Whopper. If various combinations of extra or no pickles, tomatoes, lettuce, cheese, and so on can lead to over a million ways to make a hamburger, there must be billions of possible ways to implement Scrum. And while there is no single right way, there are better and worse ways to implement Scrum.

In *Essential Scrum*, Kenny Rubin helps readers find the better ways. His isn't a prescriptive book—he doesn't say, "You must do this." Instead, he teaches the essential principles underlying success with Scrum and then gives us choices in how we live up to those principles. For example, there is no one right way for all teams to plan a sprint. What works in one company or project will fail in another. And so Kenny gives us choices. He describes an overall structure for why Scrum teams plan sprints and what must result from sprint planning, and he gives us a couple of alternative approaches that will work. But ultimately the decision belongs to each team. Fortunately for those teams, they now have this book to help them.

An unexpected benefit of *Essential Scrum* is the visual language Kenny introduces for communicating about Scrum. I found these images very helpful in following along with the text, and I suspect they will become commonplace in future discussions of Scrum.

The world has needed this book for a long time. Scrum started as a small concept. The first book to talk about it—*Wicked Problems, Righteous Solutions* in 1990 by DeGrace and Stahl—did so in six pages. But in the more than 20 years since that book appeared, Scrum has expanded. New roles, meetings, and artifacts have been introduced and refined. With each new piece that was added, we were at risk of losing the heart of Scrum, that part of it that is about a team planning how to do something, doing some small part of it, and then reflecting on what the team members did and how well they did it together.

With *Essential Scrum*, Kenny brings us back to the heart of Scrum. And from there teams can begin to make the decisions necessary to implement Scrum, making it their own. This book serves as an indispensable guide, helping teams choose among the billions of possible ways of implementing Scrum and finding one that leads to success.

—Mike Cohn
Author of *Succeeding with Agile*, *Agile Estimating and Planning*, and *User Stories Applied*
www.mountaingoatsoftware.com

This page intentionally left blank

Foreword by Ron Jeffries

When Kenny asked me to write a foreword for *Essential Scrum,* I was thinking, "This will be quick and easy; it must be a short book going straight to a simple description of what Scrum is." I knew Kenny's work, so I knew it would be a good read, and short, too. What could be better!

Imagine my surprise and delight when I found that this book covers just about everything you'll need to know about Scrum, on the first day or years into your use of Scrum. And Kenny doesn't stop there. He starts with the central ideas, including the agile principles that underlie all the agile methods, and a quick view of the Scrum framework. Then he drills in, deeper and deeper. It's still a good read, and it's quite comprehensive as well.

Kenny covers planning in good detail, looking at requirements, stories, the backlog, estimation, velocity. Then he takes us deeper into the principles and helps us deal with all the levels of planning and all the time horizons. He describes how sprints are planned, executed, reviewed, and improved. And throughout, he gives us more than the basics, highlighting key issues that you may encounter as you go along.

My own focus in Scrum and agile is on the necessary developer skills to ensure that teams can deliver real, running, business-focused software, sprint after sprint. Kenny helps us understand how to use ideas like velocity and technical debt safely and well. Both of these are critical topics, and I commend them to your attention.

Velocity tells us how much the team is delivering over time. We can use it to get a sense of how much we're getting done and whether we're improving. Kenny warns us, however, that using velocity as a performance measure is damaging to our business results, and he helps us understand why.

Technical debt has become a very broad term, referring to almost everything that could go wrong in the code. Kenny helps us tease apart all the various meanings and helps us understand why we care about these seemingly technical details. In particular, I like his description of how putting a team under pressure will inevitably damage our prospects of getting a good product on time.

Scrum, like all agile methods, relies on an exploratory approach with rapid feedback. Kenny tells a story of his brief use of punch cards, and it reminded me of my earliest experience with computing, many years before Kenny saw his first punch card.

As a college student, I was lucky enough to get a job as a sort of intern at Strategic Air Command headquarters in Omaha. In those days all computing was on cards. My cards got sent down several floors underground at SAC HQ and run on the computer that would run the war, if we ever had one. I was lucky to get one or two runs a day.

As soon as my security clearance came through, I would go down to the computer room in the middle of the night. I would sweet-talk Sergeant Whittaker into letting me run my own programs, sitting at the console of the machine—yes, the machine whose main job was to launch a nuclear attack. Rest easy, though: The red button was not in that room.

Working hands-on with the machine, I got ten times as much work done as when I had to wait for my cards to be taken down and my listings to be brought back up. Feedback came faster, I learned faster, and my programs worked sooner.

That's what Scrum is about. Instead of waiting months or even years to find out what the programmers are doing, in Scrum we find out every couple of weeks. A Scrum product owner with a really good team will be seeing actual features taking shape every few days!

And that is what Kenny's book is about. If you're new to Scrum, read it through from beginning to end. Then keep it nearby. If you've been doing Scrum for a while, scan it, then keep it nearby.

When you find yourself thinking about something that's happening to your team, or wondering about different things to try, pick up this book and look around. Chances are you'll find something of value.

—Ron Jeffries

Preface

This book discusses Essential Scrum—the things you have to know if you're going to be successful when using Scrum to develop innovative products and services.

What Is Essential Scrum?

Scrum is based on a small set of core **values**, **principles**, and **practices** (collectively the **Scrum framework**). Organizations using Scrum should embrace the Scrum framework in its entirety, perhaps not through the entire organization all at once, but certainly within the initial teams that will use Scrum. Embracing all of Scrum does not mean, however, that organizations must implement Scrum according to some cookie-cutter, one-size-fits-all formula. Rather, it means that organizations should always stay true to the Scrum framework while choosing an appropriate blend of **approaches** for their Scrum implementations.

Essential Scrum combines the values, principles, and practices of Scrum with a set of tried-and-true approaches that are consistent with, but not mandated by, the Scrum framework. Some of these approaches will be appropriate to your situation; others will not. Any approach will need to be inspected and adapted to your unique circumstances.

Origins of This Book

As an agile/Scrum coach and trainer, I am frequently asked for a reference book for Scrum—one that provides a comprehensive overview of the Scrum framework and also presents the most popular approaches for applying Scrum. Because I have been unable to find a single book that covers these topics at a level deep enough to be useful to today's practitioners, I found myself recommending a collection of books: a few that discuss the Scrum framework but are out of date or incomplete; several highly regarded agile books that do not focus solely on Scrum; and a handful that are focused on a specific aspect of Scrum or a specific approach but do not cover the full Scrum framework in depth. That's a lot of books for someone who just wants a single, stand-alone resource that covers the essentials of Scrum!

The originators of Scrum (Jeff Sutherland and Ken Schwaber) do have a Scrum-specific publication called "The Scrum Guide." This short document (about 15 pages) is described by its authors as the "definitive rule book of Scrum and the documentation of Scrum itself" (Schwaber and Sutherland 2011). They equate their document to the rules of the game of chess, "describing how the pieces move, how turns are taken, what is a win, and so on." Although useful as a Scrum overview or rule book, "The Scrum Guide" is by design not intended to be a comprehensive source of essential Scrum knowledge. Extending the authors' analogy, giving a new Scrum team just "The Scrum Guide" and expecting good results would be like giving a new chess player a 15-page description of the rules of chess and expecting her to be able to play a reasonable game of chess after reading it. It just isn't a stand-alone resource.

This book, *Essential Scrum,* is an attempt to be the missing single source for essential Scrum knowledge. It includes an in-depth discussion of Scrum's principles, values, and practices—one that in most cases agrees with other agile thought leaders and "The Scrum Guide." (Where this book offers a different perspective from what is widely promoted elsewhere, I point it out and explain why.) This book also describes approaches that are consistent with the Scrum framework and that have been used successfully by me and teams I have coached. I did not intend for this book to replace other books that provide a deep vertical treatment of a given Scrum practice or approach. Such books are complementary to and extend this book. Rather, think of *Essential Scrum* as the starting point on the journey of using Scrum to delight customers.

Intended Audience

For the many thousands of people who have taken my Working on a Scrum Team, Certified ScrumMaster, and Certified Scrum Product Owner classes, and the many teams I have coached, this book will refresh and perhaps even clarify topics we have already discussed. And for the even larger number of people with whom I have not yet had the pleasure of working, this book will either be your first introduction to Scrum and agile or it will be a chance to look at Scrum in a different light and perhaps even improve how you perform Scrum.

I did not write this book for any one specific role—this is not a book specifically for product owners, or ScrumMasters, or members of the development team. Instead, it is a book intended to give everyone involved with Scrum, from all the members of the Scrum team to those with whom they interact in the organization, a common understanding of Scrum based on a core set of concepts with a clear vocabulary for discussing them. With this shared foundation my hope is that your organization will be in a better position to successfully use Scrum to deliver business value.

I imagine that every Scrum team member would have a copy of this book on her desk open to a chapter relevant to the work at hand. I also envision managers at all levels of the organization reading it to understand why Scrum can be an effective approach for managing work and to understand the type of organizational change that may be necessary to successfully implement Scrum. Organizations using or planning to use an agile approach other than Scrum will also find the information relevant and helpful to their specific agile adoption.

Organization of This Book

This book begins with a brief introduction to Scrum (Chapter 1) and concludes with a discussion of where you might go next (Chapter 23). The remaining chapters are organized into four parts:

- Part I—Core Concepts (Chapters 2–8): Scrum framework, agile principles, sprints, requirements and user stories, product backlog, estimating and velocity, and technical debt
- Part II—Roles (Chapters 9–13): product owner, ScrumMaster, development team, Scrum team structures, and managers
- Part III—Planning (Chapters 14–18): Scrum planning principles, multilevel planning, portfolio planning, envisioning/product planning, and release planning
- Part IV—Sprinting (Chapters 19–22): sprint planning, sprint execution, sprint review, and sprint retrospective

How to Use This Book

As you would expect, I wrote the book assuming that most people would read it linearly from front to back. If you are new or newer to Scrum, you should take this approach because the chapters do tend to build on one another. That being said, if you are looking for one place to get an end-to-end overview of the Scrum framework (a highly visual Scrum primer), read and reference Chapter 2.

For those who are more familiar with Scrum, you can use this book as a Scrum reference guide. If you're interested in sprint retrospectives, jump directly to Chapter 22. If you are interested in exploring the nuances of the product backlog, jump directly to Chapter 6. I highly recommend, however, that everyone, even those familiar with Scrum, read Chapter 3 in its entirety. The principles laid out there form the foundation of the Scrum framework and the rest of the book. It is not simply a restatement of the values and principles of the Agile Manifesto (Beck et al. 2001) that is common in many other written descriptions of Scrum.

Visual Icon Language

I am proud to include in this book a new visual language for describing Scrum. This language is composed from a vocabulary of icons that have been designed to capture essential Scrum roles, artifacts, and activities. This visual Scrum language is an effective way to communicate concepts and improves the overall shared understandability of Scrum. If you are interested in obtaining and using the new full-color visual Scrum language art (this book is printed in two colors), visit www.innolution.com for details. This website will also host a variety of resources and discussions related to the book.

Let's Get Started

So, whatever your role, whatever your situation, you have picked up this book for a reason. Spend a little time getting to know Scrum. In the pages that follow you just might find a powerful framework that you can make your own, allowing you to substantially improve the way you develop and deliver products and services to delight your customers.

Acknowledgments

This book would not have been possible without the input of many people, including the thousands of people who have attended my agile-related classes and coaching sessions. By mentioning some people by name, I run the risk of failing to mention others. To those whose names I fail to mention, please know that all of our discussions and email exchanges have been invaluable to me and have definitely influenced this book!

There are three people in particular I would like to thank: Mike Cohn, Rebecca Traeger, and Jeff Schaich. Without the unique involvement of each, this book would be a mere shadow of itself.

Mike Cohn has been a friend and colleague since we first worked together at Genomica in 2000. He was gracious enough to include my book in the Mike Cohn Signature Series; by being affiliated with Mike and the other prestigious authors in that book series, "I look good by the company that I keep," as my parents would say. Mike was my go-to person whenever I wanted to bounce around ideas or discuss book strategies. He always made time in his insane schedule to review each chapter and give me his thoughtful feedback. Working with Mike all these years has been a very rewarding experience—one that I hope will continue long into the future.

Rebecca Traeger has been my personal editor on this book. We have worked together since my days as managing director of the Scrum Alliance in 2007. At that time Rebecca was the editor of the Scrum Alliance website and through that work (and much more since) became the industry's foremost editor on agile-related materials. Early on in writing this book I reached out to Rebecca and asked if she would work with me again, and to my good fortune, she agreed. Nobody saw any chapter unless Rebecca had seen it first. At times her feedback would make me blush, because she frequently improved how I said something, making it sound both more understandable and approachable. If you just love a section of this book, you can be sure Rebecca had her hands in it. If you don't, I probably foolishly chose to ignore her recommendations.

Jeff Schaich is an artist/designer extraordinaire. We have worked on so many different art projects that I can't recall them all. Early on in the formulation of this book I wanted to create an agile/Scrum icon vocabulary to use as the basis for my training presentations and many of the over 200 figures in the book. I knew that I needed a great designer to pull off this feat. Jeff agreed to take on the challenge. There are times when this book seemed like two different projects—writing the content and creating the artistic concepts. I'm honestly not sure which took more time. I am sure, however, that without Jeff's artistic input, this book would have suffered immeasurably.

I am deeply honored to have both Mike Cohn and Ron Jeffries, two luminaries in the agile community, write forewords for the book! In their own unique ways each has done a great job of properly placing the book in context and opening the door for a discussion of Essential Scrum. Also, Mike, stop eating at Burger King, and Ron, thanks for not pushing the red button!

I'd also like to thank the many people who took time out of their busy schedules to review chapters and send me their feedback. Let me start by mentioning reviewers who provided extensive feedback: Joe Balistrieri, Johannes Brodwall, Leyna Cotran, Martine Devos, Scott Duncan, Ilan Goldstein, John Hebley, Geir Hedemark, James Kovacs, Lauri Mackinnon, Robert Maksimchuk, and Kevin Tureski.

In addition, I would like to thank other reviewers who provided excellent feedback on select chapters: Lyssa Adkins, John Bauer, Sameer Bendre, Susan Briscoe, Pawel Brodzinski, Rowan Bunning, Josh Chappell, Lisa Crispin, Ward Cunningham, Cornelius Engelbrecht, Julia Frazier, Brindusa Gabur, Caroline Gordon, Drew Jemilo, Mike Klimkosky, Tom Langerhorst, Bjarne Larsen, Dean Leffingwell, Maurice le Rutte, David Luzquiños, Lv Yi, Shay McAulay, Armond Mehrabian, Sheriff Mohamed, Cuan Mulligan, Greg Pease, Roman Pichler, Jacopo Romei, Jens Schauder, Bill Schroeder, Yves Stalgies, Branko Stojaković, Howard Sublett, Julie Sylvain, Kevin Tambascio, Stephen Wolfram, and Michael Wollin.

I would also like to thank the staff at Pearson who were great partners in this project. They tolerated my delays with patience and always offered encouragement. Special thanks to Chris Guzikowski, who oversaw the whole thing from soup to nuts. He was there from my first Pearson meeting at a pub in Lexington, MA, through the final production. I would also like to thank Olivia Basegio for adeptly handling logistics and Julie Nahil who did a fantastic job overseeing the project. In addition, thanks to Barbara Wood for the great job of helping polish the manuscript and Gail Cocker for pulling all of the art together into a coherent and beautiful whole.

I am also grateful to my assistant, Lindsey Kalicki, to whom I was able to offload many important tasks so that I could stay focused on book development. I am lucky to be able to work with such a skilled professional.

Most of all, I would like to acknowledge my family—Jenine, Jonah, and Asher—and the critical role that they played. I have asked so very much from them during the long effort of creating this book. No amount of gratitude can make up for the family pressure it caused and our lost time together.

Jenine is my loving soulmate and has stuck by me through all of the ups and downs of writing this book. The sacrifices she made so that I could write would double the size of this book if I tried to list them all. I couldn't have done it without her!

Funny thing is, a year after we were married in 1993, I published my first book, *Succeeding with Objects*. At that time Jenine made me promise that I would never write another book again. Luckily for me, after 15 years memories fade and the crushing workload doesn't seem as bad in hindsight, so when she urged me to write this one I was surprised to say the least! She hasn't yet told me I can't do book number three, but I suspect it might be 15 more years before the memory of this one fades enough for either of us to want me to write another one!

I also deeply appreciate the loving support from my sons, Jonah and Asher. They gave up time with their dad so that I could write. They were always there to bounce around ideas and to give input on the book. A number of their content and art suggestions have made their way into the book—and it's better because of them! I hope they learned the value of perseverance and that even the most daunting work can be completed if you take it a step at a time and don't give up.

Finally, I would like to acknowledge my mom, Joyce Rubin (Genesha Esther bat Avrahm), for all of the love and support she gave me. Without her influence this book would never have been possible. Sadly, she did not survive to see its publication. Her passing in January 2012 left a void in my life and the lives of her family that can never be filled. She was a very special person to the many whose lives she touched. Mom, I miss you more than I can possibly express.

This page intentionally left blank

About the Author

Kenny Rubin provides Scrum and agile training and coaching to help companies develop products in an effective and economically sensible way. A Certified Scrum Trainer, Kenny has trained over 18,000 people on agile and Scrum, Smalltalk development, managing object-oriented projects, and transition management. He has coached over 200 companies, ranging from start-ups to Fortune 10.

Kenny was the first Managing Director of the worldwide Scrum Alliance, a nonprofit organization focused on the successful adoption of Scrum. In addition to this book, Kenny is also the coauthor of the 1995 book *Succeeding with Objects: Decision Frameworks for Project Management*. He received his B.S. in Information and Computer Science from the Georgia Institute of Technology and his M.S. in Computer Science from Stanford University.

Kenny's background is rooted in the object-oriented technology community. He started as a Smalltalk developer on a NASA-funded project back in 1985 and developed the first blackboard expert system outside of LISP. In 1988 he was fortunate to join ParcPlace Systems, a start-up company formed as a Xerox PARC spin-off, whose charter was to bring object-oriented technology out of the research labs and release it to the world. As a Smalltalk development consultant with many different organizations in the late 1980s and throughout the 1990s, Kenny was an early adopter of agile practices. His first use of Scrum was in 2000 for developing bioinformatics software.

In the course of his career, Kenny has held many roles, including successful stints as a Scrum product owner, ScrumMaster, and member of development teams. In addition, he has held numerous executive management roles: CEO, COO, VP of Engineering, VP of Product Management, and VP of Professional Services. He has also overseen the development of five commercial software product suites, generating over $200M in aggregate revenue. In addition, he has been directly involved in raising over $150M in venture capital funding and assisted in taking two companies public on the NASDAQ.

His multifaceted background gives Kenny the ability to understand (and explain) Scrum and its implications equally well from multiple perspectives: from the development team to the executive board.

This page intentionally left blank

Chapter 2
Scrum Framework

This chapter provides an overview of the Scrum framework with a primary focus on its practices, including roles, activities, and artifacts. Subsequent chapters will provide a deeper treatment of each of these practices, including an in-depth look at the principles that underlie the practices.

Overview

Scrum is not a standardized process where you methodically follow a series of sequential steps that are guaranteed to produce, on time and on budget, a high-quality product that delights customers. Instead, Scrum is a **framework** for organizing and managing work. The Scrum framework is based on a set of values, principles, and practices that provide the foundation to which your organization will add its unique implementation of relevant engineering practices and your specific approaches for realizing the Scrum practices. The result will be a version of Scrum that is uniquely yours.

To better grasp the framework concept, imagine that the Scrum framework is like the foundation and walls of a building. The Scrum values, principles, and practices would be the key structural components. You can't ignore or fundamentally change a value, principle, or practice without risking collapse. What you can do, however, is customize inside the structure of Scrum, adding fixtures and features until you have a process that works for you.

Scrum is a refreshingly simple, people-centric framework based on the values of honesty, openness, courage, respect, focus, trust, empowerment, and collaboration. Chapter 3 will describe the Scrum principles in depth; subsequent chapters will highlight how specific practices and approaches are rooted in these principles and values.

The Scrum practices themselves are embodied in specific roles, activities, artifacts, and their associated rules (see Figure 2.1).

The remainder of this chapter will focus on Scrum practices.

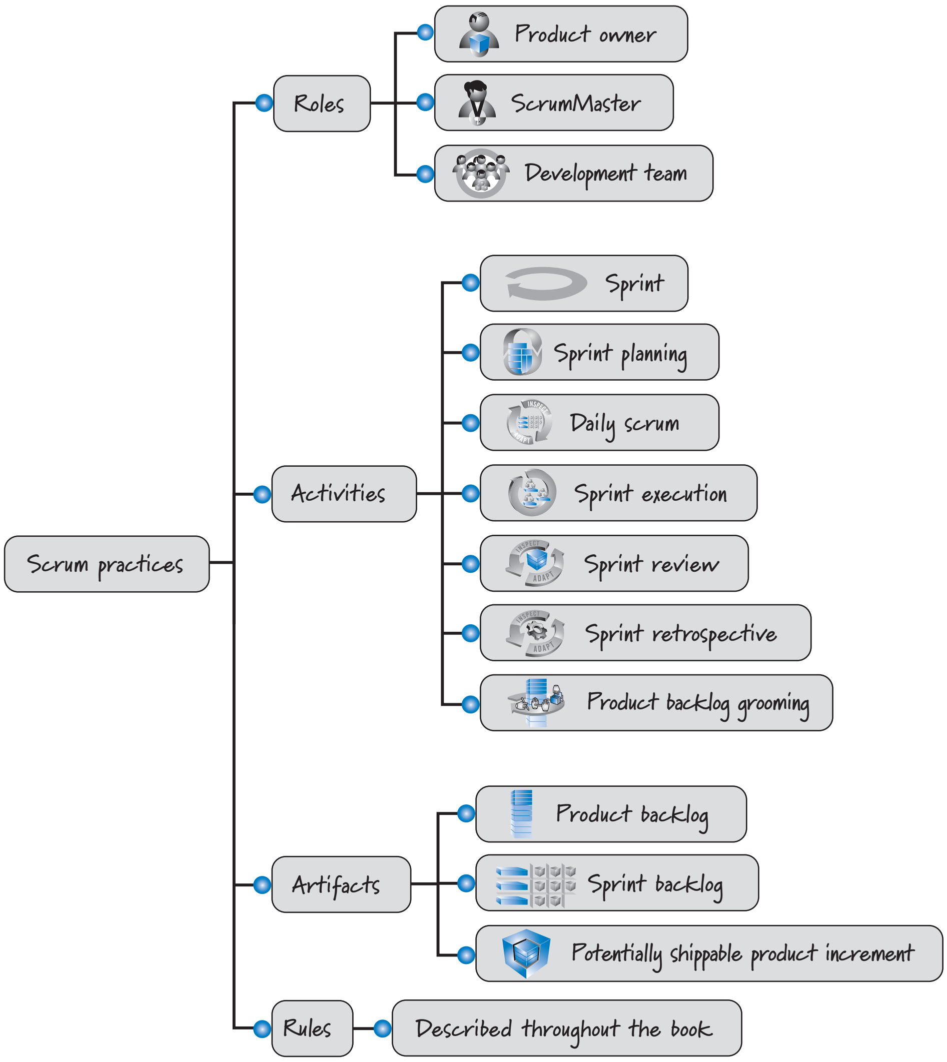


FIGURE 2.1 Scrum practices

Scrum Roles

Scrum development efforts consist of one or more **Scrum teams**, each made up of three Scrum roles: **product owner**, **ScrumMaster**, and the **development team** (see Figure 2.2). There can be other roles when using Scrum, but the Scrum framework requires only the three listed here.

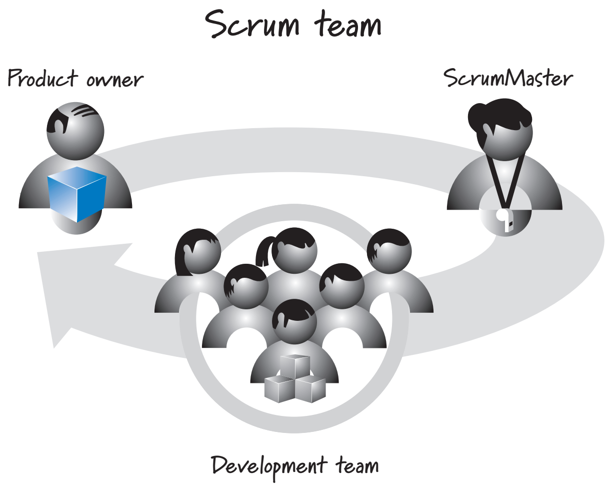


FIGURE 2.2 Scrum roles

The product owner is responsible for what will be developed and in what order. The ScrumMaster is responsible for guiding the team in creating and following its own process based on the broader Scrum framework. The development team is responsible for determining how to deliver what the product owner has asked for.

If you are a manager, don't be concerned that "manager" doesn't appear as a role in Figure 2.2; managers still have an important role in organizations that use Scrum (see Chapter 13). The Scrum framework defines just the roles that are specific to Scrum, not all of the roles that can and should exist within an organization that uses Scrum.

Product Owner

The product owner is the empowered central point of product leadership. He[1] is the single authority responsible for deciding which features and functionality to build and the order in which to build them. The product owner maintains and communicates to all other participants a clear vision of what the Scrum team is trying to achieve. As such, the product owner is responsible for the overall success of the solution being developed or maintained.

It doesn't matter if the focus is on an external product or an internal application; the product owner still has the obligation to make sure that the most valuable work possible, which can include technically focused work, is always performed. To

1. In this book the product owner will always be referred to as "he" or "him" and the ScrumMaster as "she" or "her." This is consistent with the visual representation of each role within the figures.

ensure that the team rapidly builds what the product owner wants, the product owner actively collaborates with the ScrumMaster and development team and must be available to answer questions soon after they are posed. See Chapter 9 for a detailed description of the product owner role.

ScrumMaster

The ScrumMaster helps everyone involved understand and embrace the Scrum values, principles, and practices. She acts as a coach, providing process leadership and helping the Scrum team and the rest of the organization develop their own high-performance, organization-specific Scrum approach. At the same time, the ScrumMaster helps the organization through the challenging change management process that can occur during a Scrum adoption.

As a facilitator, the ScrumMaster helps the team resolve issues and make improvements to its use of Scrum. She is also responsible for protecting the team from outside interference and takes a leadership role in removing **impediments** that inhibit team productivity (when the individuals themselves cannot reasonably resolve them). The ScrumMaster has no authority to exert control over the team, so this role is not the same as the traditional role of project manager or development manager. The ScrumMaster functions as a leader, not a manager. I will discuss the roles of functional manager and project manager in Chapter 13. See Chapter 10 for more details on the ScrumMaster role.

Development Team

Traditional software development approaches discuss various job types, such as architect, programmer, tester, database administrator, UI designer, and so on. Scrum defines the role of a development team, which is simply a diverse, cross-functional collection of these types of people who are responsible for designing, building, and testing the desired product.

The development team self-organizes to determine the best way to accomplish the goal set out by the product owner. The development team is typically five to nine people in size; its members must collectively have all of the skills needed to produce good-quality, working software. Of course, Scrum can be used on development efforts that require much larger teams. However, rather than having one Scrum team with, say, 35 people, there would more likely be four or more Scrum teams, each with a development team of nine or fewer people. See Chapter 11 for more details on the development team role and Chapter 12 for more details on coordinating multiple teams.

Scrum Activities and Artifacts

Figure 2.3 illustrates most of the Scrum activities and artifacts and how they fit together.

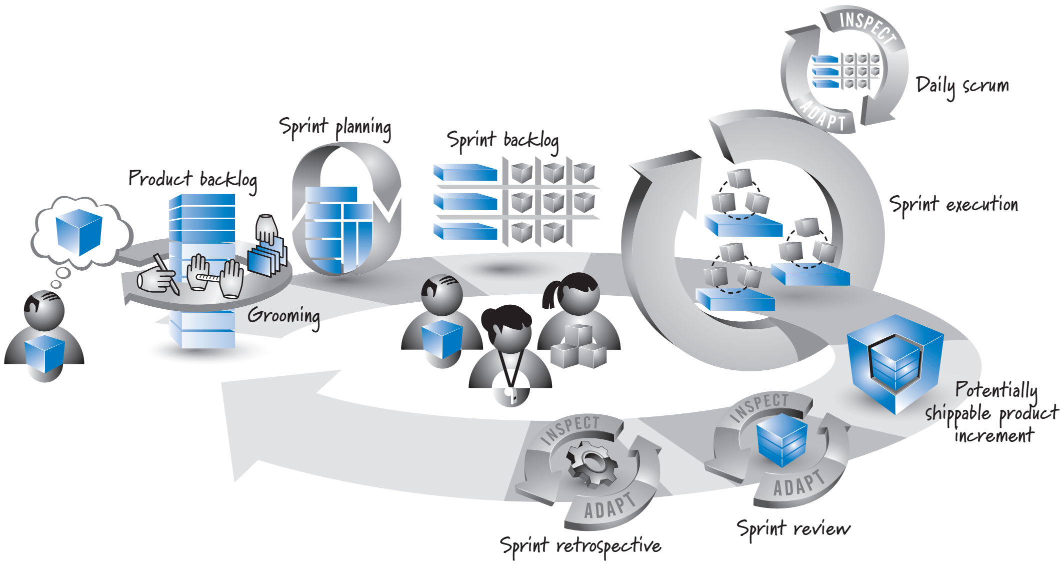


FIGURE 2.3 Scrum framework

Let's summarize the diagram, starting on the left side of the figure and working clockwise around the main looping arrow (the sprint).

The product owner has a vision of what he wants to create (the big cube). Because the cube can be large, through an activity called **grooming** it is broken down into a set of features that are collected into a prioritized list called the product backlog.

A sprint starts with sprint planning, encompasses the development work during the sprint (called sprint execution), and ends with the review and retrospective. The sprint is represented by the large, looping arrow that dominates the center of the figure. The number of items in the product backlog is likely to be more than a development team can complete in a short-duration sprint. For that reason, at the beginning of each sprint, the development team must determine a subset of the product backlog items it believes it can complete—an activity called sprint planning, shown just to the right of the large product backlog cube.

As a brief aside, in 2011 a change in "The Scrum Guide" (Schwaber and Sutherland 2011) generated debate about whether the appropriate term for describing the result of sprint planning is a **forecast** or a **commitment**. Advocates of the word *forecast* like it because they feel that although the development team is making the best estimate that it can at the time, the estimate might change as more information becomes known during the course of the sprint. Some also believe that a commitment on the part of the team will cause the team to sacrifice quality to meet the commitment or will cause the team to "under-commit" to guarantee that the commitment is met.

I agree that all development teams should generate a forecast (estimate) of what they can deliver each sprint. However, many development teams would benefit from using the forecast to derive a commitment. Commitments support mutual trust between the product owner and the development team as well as within the development team. Also, commitments support reasonable short-term planning and decision making within an organization. And, when performing multiteam product development, commitments support synchronized planning—one team can make decisions based on what another team has committed to do. In this book, I favor the term *commitment*; however, I occasionally use *forecast* if it seems correct in context.

To acquire confidence that the development team has made a reasonable commitment, the team members create a second backlog during sprint planning, called the sprint backlog. The sprint backlog describes, through a set of detailed **tasks**, how the team plans to design, build, integrate, and test the selected subset of features from the product backlog during that particular sprint.

Next is sprint execution, where the development team performs the tasks necessary to realize the selected features. Each day during sprint execution, the team members help manage the flow of work by conducting a synchronization, inspection, and adaptive planning activity known as the daily scrum. At the end of sprint execution the team has produced a potentially shippable product increment that represents some, but not all, of the product owner's vision.

The Scrum team completes the sprint by performing two inspect-and-adapt activities. In the first, called the sprint review, the stakeholders and Scrum team inspect the product being built. In the second, called the sprint retrospective, the Scrum team inspects the Scrum process being used to create the product. The outcome of these activities might be adaptations that will make their way into the product backlog or be included as part of the team's development process.

At this point the Scrum sprint cycle repeats, beginning anew with the development team determining the next most important set of product backlog items it can complete. After an appropriate number of sprints have been completed, the product owner's vision will be realized and the solution can be released.

In the remainder of this chapter I will discuss each of these activities and artifacts in greater detail.

Product Backlog

Using Scrum, we always do the most valuable work first. The product owner, with input from the rest of the Scrum team and stakeholders, is ultimately responsible for determining and managing the sequence of this work and communicating it in the form of a prioritized (or ordered) list known as the **product backlog** (see Figure 2.4). On new-product development the product backlog items initially are features required to meet the product owner's vision. For ongoing product development, the product backlog might also contain new features, changes to existing features, defects needing repair, technical improvements, and so on.

The product owner collaborates with internal and external stakeholders to gather and define the product backlog items. He then ensures that product backlog items

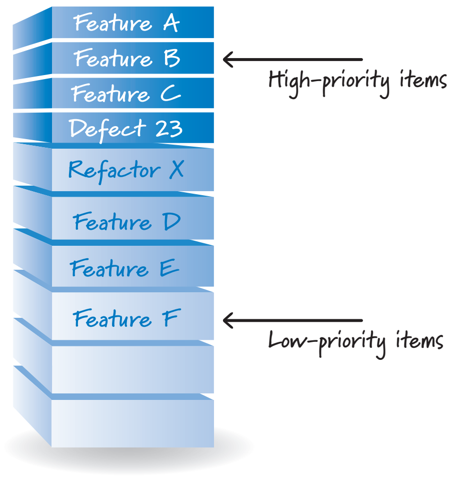


FIGURE 2.4 Product backlog

are placed in the correct sequence (using factors such as value, cost, knowledge, and risk) so that the high-value items appear at the top of the product backlog and the lower-value items appear toward the bottom. The product backlog is a constantly evolving artifact. Items can be added, deleted, and revised by the product owner as business conditions change, or as the Scrum team's understanding of the product grows (through feedback on the software produced during each sprint).

Overall the activity of creating and refining product backlog items, estimating them, and prioritizing them is known as grooming (see Figure 2.5).

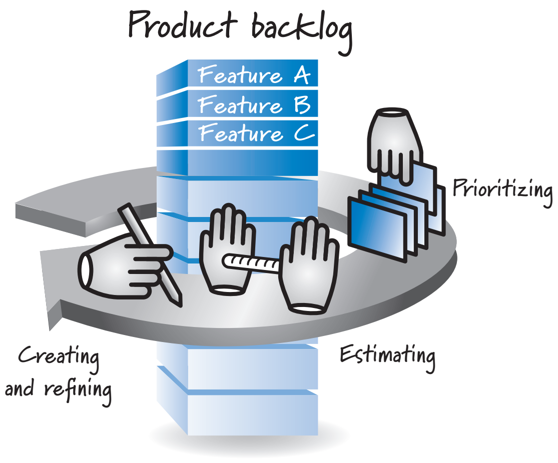


FIGURE 2.5 Product backlog grooming

As a second brief aside, in 2011 there was another debate as to whether the appropriate term for describing the sequence of items in the product backlog should be *prioritized* (the original term) or *ordered*, the term used in "The Scrum Guide" (Schwaber and Sutherland 2011). The argument was that prioritizing is simply one form of ordering (and, according to some, not even the most appropriate form of ordering). The issue of how to best sequence items in the product backlog, however, is influenced by many factors, and a single word may never capture the full breadth and depth of the concept. Although there may be theoretical merit to the ordered-versus-prioritized debate, most people (including me) use the terms interchangeably when discussing the items in the product backlog.

Before we finalize prioritizing, ordering, or otherwise arranging the product backlog, we need to know the size of each item in the product backlog (see Figure 2.6).

Size equates to cost, and product owners need to know an item's cost to properly determine its priority. Scrum does not dictate which, if any, size measure to use with product backlog items. In practice, many teams use a **relative size measure** such as **story points** or **ideal days**. A relative size measure expresses the overall size of an item in such a way that the absolute value is not considered, but the relative size of an item compared to other items is considered. For example, in Figure 2.6, feature C is size 2 and feature E is size 8. What we can conclude is that feature E is about four times larger than feature C. I will discuss these measures further in Chapter 7.

Sprints

In Scrum, work is performed in iterations or cycles of up to a calendar month called **sprints** (see Figure 2.7). The work completed in each sprint should create something of tangible value to the customer or user.

Sprints are **timeboxed** so they always have a fixed start and end date, and generally they should all be of the same duration. A new sprint immediately follows the completion of the previous sprint. As a rule we do not permit any goal-altering changes in scope or personnel during a sprint; however, business needs sometimes make adherence to this rule impossible. I will describe sprints in more detail in Chapter 4.

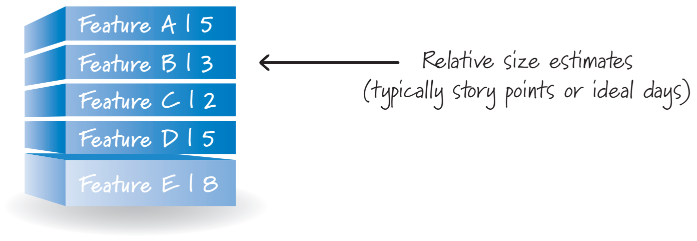


FIGURE 2.6 Product backlog item sizes

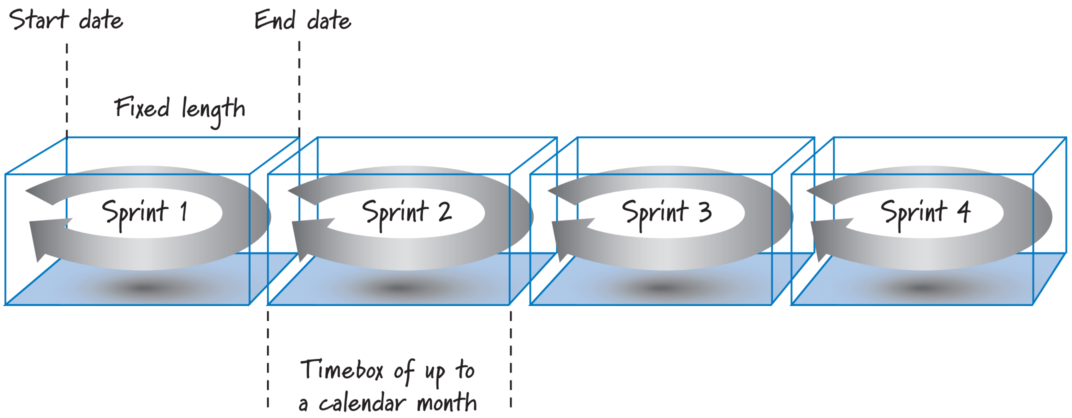


FIGURE 2.7 Sprint characteristics

Sprint Planning

A product backlog may represent many weeks or months of work, which is much more than can be completed in a single, short sprint. To determine the most important subset of product backlog items to build in the next sprint, the product owner, development team, and ScrumMaster perform **sprint planning** (see Figure 2.8).

During sprint planning, the product owner and development team agree on a **sprint goal** that defines what the upcoming sprint is supposed to achieve. Using this

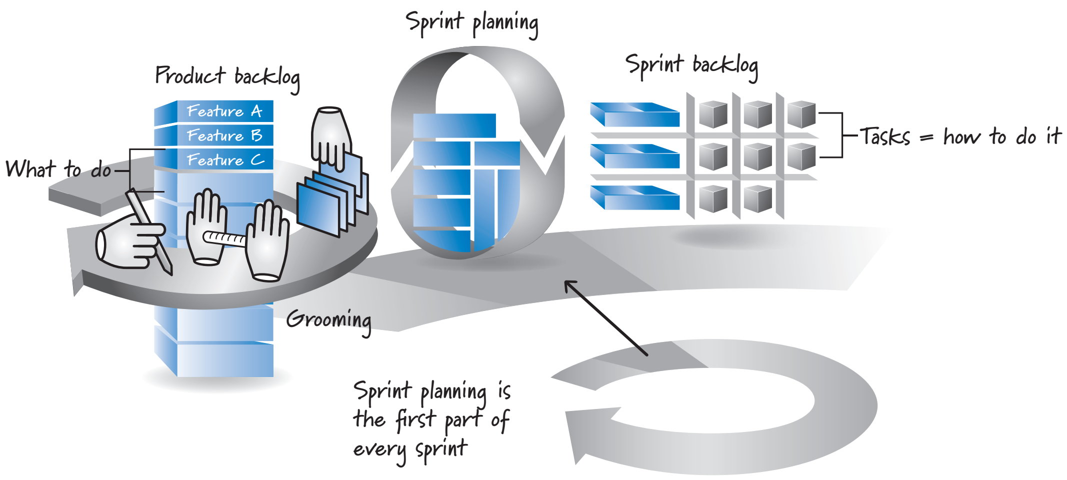


FIGURE 2.8 Sprint planning

goal, the development team reviews the product backlog and determines the high-priority items that the team can realistically accomplish in the upcoming sprint while working at a **sustainable pace**—a pace at which the development team can comfortably work for an extended period of time.

To acquire confidence in what it can get done, many development teams break down each targeted feature into a set of tasks. The collection of these tasks, along with their associated product backlog items, forms a second backlog called the **sprint backlog** (see Figure 2.9).

The development team then provides an estimate (typically in hours) of the effort required to complete each task. Breaking product backlog items into tasks is a form of design and **just-in-time** planning for how to get the features done.

Most Scrum teams performing sprints of two weeks to a month in duration try to complete sprint planning in about four to eight hours. A one-week sprint should take no more than a couple of hours to plan (and probably less). During this time there are several approaches that can be used. The approach I use most often follows a simple cycle: Select a product backlog item (whenever possible, the next-most-important item as defined by the product owner), break the item down into tasks, and determine if the selected item will reasonably fit within the sprint (in combination with other items targeted for the same sprint). If it does fit and there is more capacity to complete work, repeat the cycle until the team is out of capacity to do any more work.

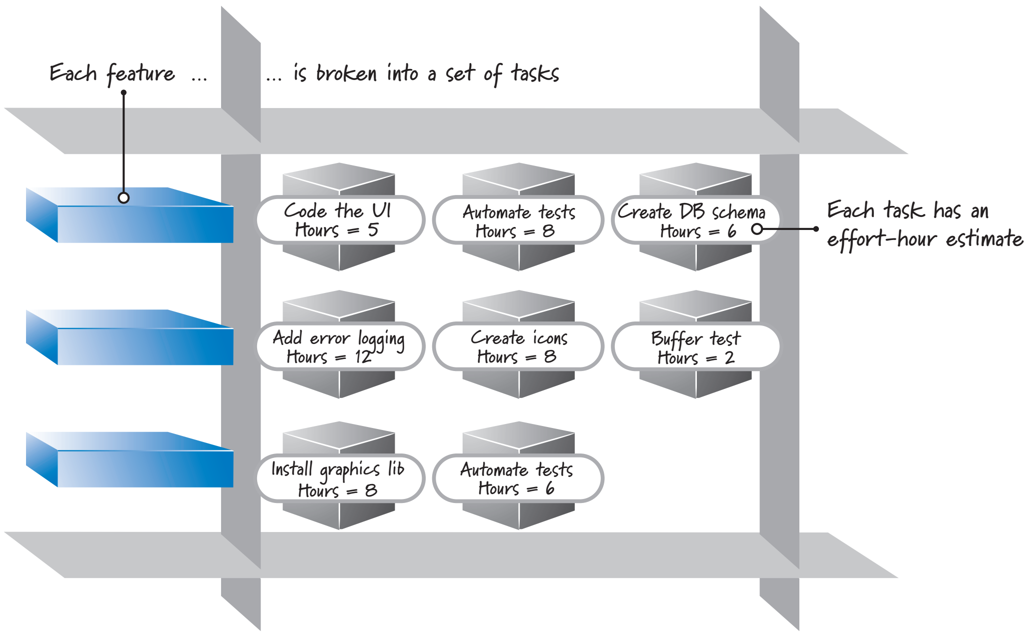

Each feature ...
... is broken into a set of tasks
Code the UI Hours = 5
Automate tests Hours = 8
Create DB schema Hours = 6
Each task has an effort-hour estimate
Add error logging Hours = 12
Create icons Hours = 8
Buffer test Hours = 2
Install graphics lib Hours = 8
Automate tests Hours = 6

FIGURE 2.9 Sprint backlog

An alternative approach would be for the product owner and team to select all of the target product backlog items at one time. The development team alone does the task breakdowns to confirm that it really can deliver all of the selected product backlog items. I will describe each approach in more detail in Chapter 19.

Sprint Execution

Once the Scrum team finishes sprint planning and agrees on the content of the next sprint, the development team, guided by the ScrumMaster's coaching, performs all of the task-level work necessary to get the features done (see Figure 2.10), where "done" means there is a high degree of confidence that all of the work necessary for producing good-quality features has been completed.

Exactly what tasks the team performs depends of course on the nature of the work (for example, are we building software and what type of software, or are we building hardware, or is this marketing work?).

Nobody tells the development team in what order or how to do the task-level work in the sprint backlog. Instead, team members define their own task-level work and then self-organize in any manner they feel is best for achieving the sprint goal. See Chapter 20 for more details on sprint execution.

Daily Scrum

Each day of the sprint, ideally at the same time, the development team members hold a timeboxed (15 minutes or less) **daily scrum** (see Figure 2.11). This inspect-and-adapt activity is sometimes referred to as the **daily stand-up** because of the common practice of everyone standing up during the meeting to help promote brevity.

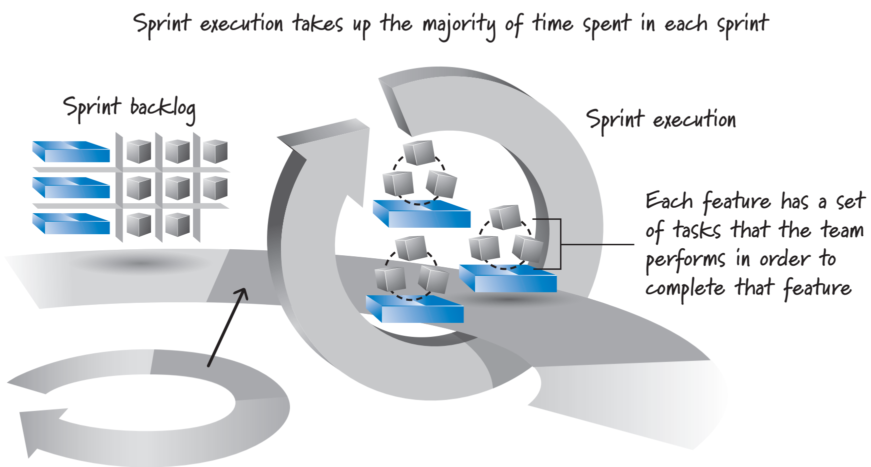

Sprint execution takes up the majority of time spent in each sprint
Sprint backlog
Sprint execution
Each feature has a set of tasks that the team performs in order to complete that feature

FIGURE 2.10 Sprint execution

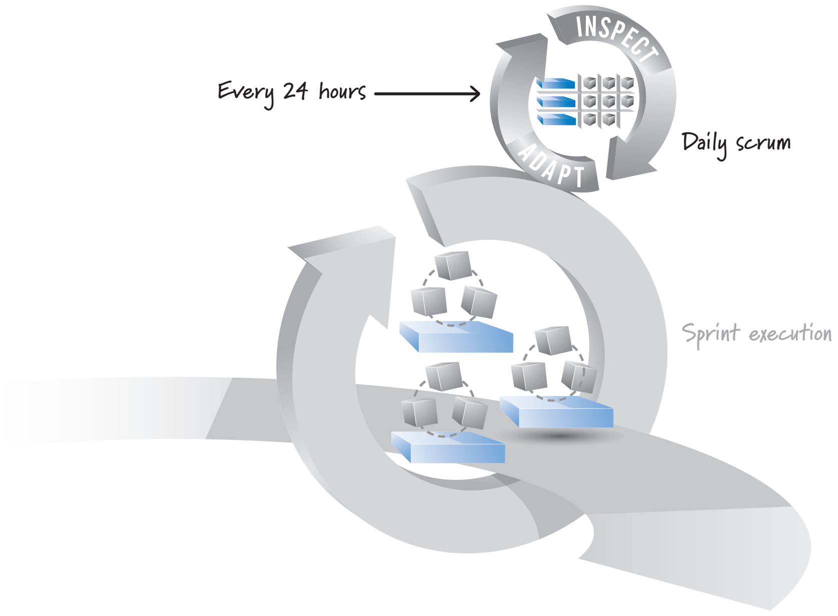


FIGURE 2.11 Daily scrum

A common approach to performing the daily scrum has the ScrumMaster facilitating and each team member taking turns answering three questions for the benefit of the other team members:

- What did I accomplish since the last daily scrum?
- What do I plan to work on by the next daily scrum?
- What are the obstacles or impediments that are preventing me from making progress?

By answering these questions, everyone understands the big picture of what is occurring, how they are progressing toward the sprint goal, any modifications they want to make to their plans for the upcoming day's work, and what issues need to be addressed. The daily scrum is essential for helping the development team manage the fast, flexible flow of work within a sprint.

The daily scrum is not a problem-solving activity. Rather, many teams decide to talk about problems after the daily scrum and do so with a small group of interested people. The daily scrum also is not a traditional status meeting, especially the kind historically called by project managers so that they can get an update on the project's status. A daily scrum, however, can be useful to communicate the status of sprint backlog items among the development team members. Mainly, the daily scrum is an inspection, synchronization, and adaptive daily planning activity that helps a self-organizing team do its job better.

Although their use has fallen out of favor, Scrum has used the terms "**pigs**" and "**chickens**" to distinguish who should participate during the daily scrum versus who simply observes. The farm animals were borrowed from an old joke (which has several variants): "In a ham-and-eggs breakfast, the chicken is involved, but the pig is committed." Obviously the intent of using these terms in Scrum is to distinguish between those who are involved (the chickens) and those who are committed to meeting the sprint goal (the pigs). At the daily scrum, only the pigs should talk; the chickens, if any, should attend as observers.

I have found it most useful to consider everyone on the Scrum team a pig and anyone who isn't, a chicken. Not everyone agrees. For example, the product owner is not required to be at the daily scrum, so some consider him to be a chicken (the logic being, how can you be "committed" if you aren't required to attend?). This seems wrong to me, because I can't imagine how the product owner, as a member of the Scrum team, is any less committed to the outcome of a sprint than the development team. The metaphor of pigs and chickens breaks down if you try to apply it within a Scrum team.

Done

In Scrum, we refer to the sprint results as a **potentially shippable product increment** (see Figure 2.12), meaning that whatever the Scrum team agreed to do is really done according to its agreed-upon definition of done. This definition specifies the degree

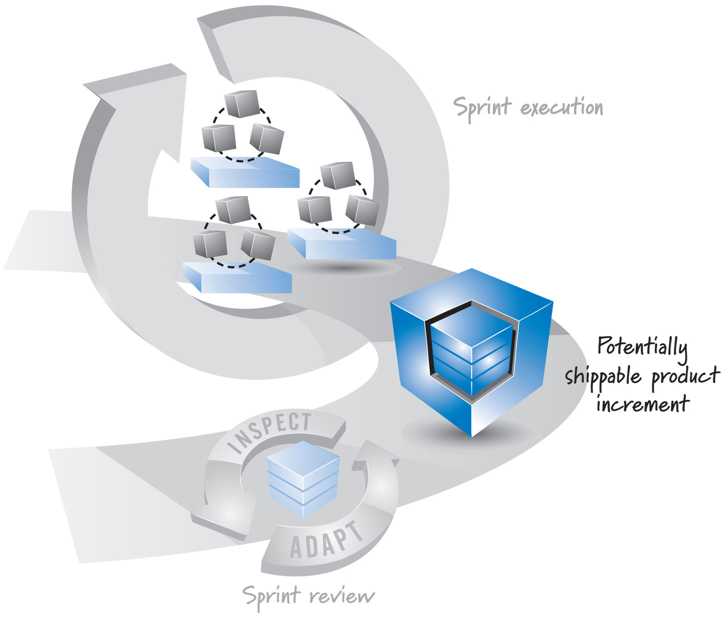


FIGURE 2.12 Sprint results (potentially shippable product increment)

of confidence that the work completed is of good quality and is potentially shippable. For example, when developing software, a bare-minimum definition of done should yield a complete slice of product functionality that is designed, built, integrated, tested, and documented.

An aggressive definition of done enables the business to decide each sprint if it wants to ship (or deploy or release) what got built to internal or external customers.

To be clear, "potentially shippable" does not mean that what got built must actually be shipped. Shipping is a business decision, which is frequently influenced by things such as "Do we have enough features or enough of a customer workflow to justify a customer deployment?" or "Can our customers absorb another change given that we just gave them a release two weeks ago?"

Potentially shippable is better thought of as a state of confidence that what got built in the sprint is actually done, meaning that there isn't materially important undone work (such as important testing or integration and so on) that needs to be completed before we can ship the results from the sprint, if shipping is our business desire.

As a practical matter, over time some teams may vary the definition of done. For example, in the early stages of game development, having features that are potentially shippable might not be economically feasible or desirable (given the exploratory nature of early game development). In these situations, an appropriate definition of done might be a slice of product functionality that is sufficiently functional and usable to generate feedback that enables the team to decide what work should be done next or how to do it. See Chapter 4 for more details on the definition of done.

Sprint Review

At the end of the sprint there are two additional inspect-and-adapt activities. One is called the **sprint review** (see Figure 2.13).

The goal of this activity is to inspect and adapt the product that is being built. Critical to this activity is the conversation that takes place among its participants, which include the Scrum team, stakeholders, sponsors, customers, and interested members of other teams. The conversation is focused on reviewing the just-completed features in the context of the overall development effort. Everyone in attendance gets clear visibility into what is occurring and has an opportunity to help guide the forthcoming development to ensure that the most business-appropriate solution is created.

A successful review results in bidirectional information flow. The people who aren't on the Scrum team get to sync up on the development effort and help guide its direction. At the same time, the Scrum team members gain a deeper appreciation for the business and marketing side of their product by getting frequent feedback on the convergence of the product toward delighted customers or users. The sprint review therefore represents a scheduled opportunity to inspect and adapt the product. As a

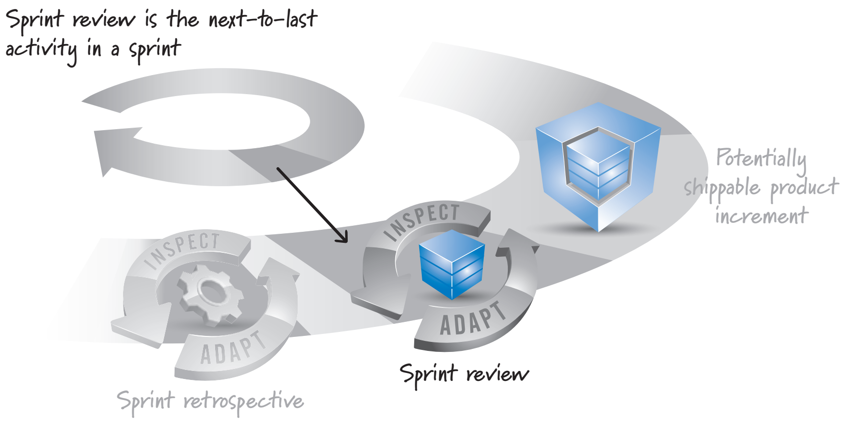

Sprint review is the next-to-last activity in a sprint
INSPECT
ADAPT
Potentially shippable product increment
INSPECT
ADAPT
Sprint review
Sprint retrospective

FIGURE 2.13 Sprint review

matter of practice, people outside the Scrum team can perform intra-sprint feature reviews and provide feedback to help the Scrum team better achieve its sprint goal. See Chapter 21 for more details on the sprint review.

Sprint Retrospective

The second inspect-and-adapt activity at the end of the sprint is the **sprint retrospective** (see Figure 2.14). This activity frequently occurs after the sprint review and before the next sprint planning.

Whereas the sprint review is a time to inspect and adapt the product, the sprint retrospective is an opportunity to inspect and adapt the process. During the sprint retrospective the development team, ScrumMaster, and product owner come together

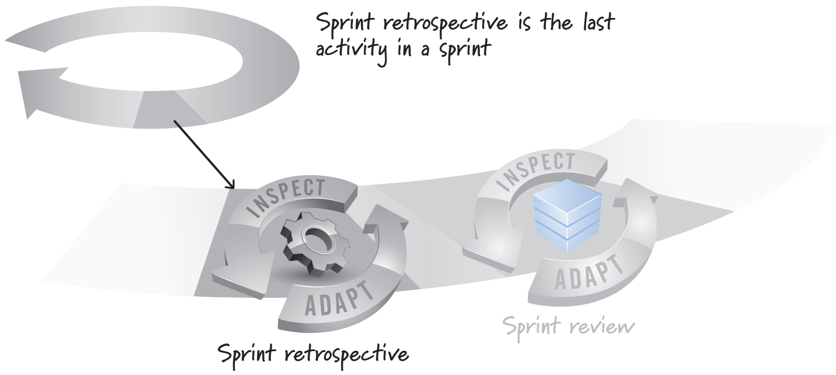


FIGURE 2.14 Sprint retrospective

to discuss what is and is not working with Scrum and associated technical practices. The focus is on the continuous process improvement necessary to help a good Scrum team become great. At the end of a sprint retrospective the Scrum team should have identified and committed to a practical number of process improvement actions that will be undertaken by the Scrum team in the next sprint. See Chapter 22 for details on the sprint retrospective.

After the sprint retrospective is completed, the whole cycle is repeated again—starting with the next sprint-planning session, held to determine the current highest-value set of work for the team to focus on.

Closing

This chapter described core Scrum practices, focusing on an end-to-end description of the Scrum framework's roles, activities, and artifacts. There are other practices, such as higher-level planning and progress-tracking practices, that many Scrum teams use. These will be described in subsequent chapters. In the next chapter, I will provide a description of the core principles on which Scrum is based. This will facilitate the deeper exploration of the Scrum framework in subsequent chapters.

INDEX

A

Absolute sizes, vs. relative sizes in estimation, 125–128
Acceptance criteria
    conditions of satisfaction related to product backlog, 77
    defined, 401
    definition of ready and, 110
    product owner defining and verifying, 169–170
    user stories containing confirmation information, 85–86
Acceptance-test-driven development (ATTD), 85–86, 402
Acceptance tests
    conditions of satisfaction expressed via, 85
    defined, 401
    product owner responsibilities and, 169
    verifying conditions of satisfaction, 77
Accountability, of product owner, 173
Accrual of technical debt, managing, 149–152
Accuracy
    defined, 402
    vs. precision in estimation, 125, 274–275
Actions, resulting from retrospective
    deciding what action to take, 389–390
    determining possible actions, 387–388
    follow through on, 391–392
    as output of sprint retrospective, 381
    selecting insights to act on, 388
Activities
    defined, 402
    overview of, 16–18
Adaptation. *See also* Prediction and adaptation principle, in agile development
    balancing predictive work with adaptive work, 43–44
    based on product review, 371
    daily scrum as inspect-and-adapt activity, 354
    defined, 402
    discovering your own path forward, 396
    and exploration in approach to development, 39–40
    as focus of planning rather than conformance, 249–251
    leveraging variability, 35–36
    plan-driven development compared with agile development, 59
    responsibilities of development team, 197–198
    sprint retrospective and, 375
    sprint review and, 363
Agile development
    concerns about adopting, 225
    defined, 402
    managers promoting agile values, 233–234
    no end state in, 395
    overview of, 1–3
    plan-driven approach compared with, 59–60
    product backlog in, 1
    sharing best practices, 396–397
The Agile Manifesto (Beck), xxxi, 30, 204–205, 210
Agile principles
    accepting that you can't get it right up front, 38–39
    adapting to real-time information and replanning based on, 54
    adaptive, exploratory approach, 39–40
    balancing predictive work with adaptive work, 43–44
    batch sizes in, 48–49
    cost of change and, 40–43
    cost of delays and, 52–54

Agile principles (*continued*)
  embracing helpful variability, 32–33
  focusing on idle work, 51–52
  inspection, adaptation, and transparency, 35–36
  inventory management, 49–50
  iterative and incremental approach to development, 33–35
  keeping options open, 37–38
  learning loops in, 45–46
  measuring progress by asset validation, 54–55
  minimizing unnecessary formality, 57–58
  organizing workflow for fast feedback, 46–47
  overview of, 29–32
  prediction and adaptation, 37
  quality built-in to development process, 56–57
  reducing uncertainty, 36–37
  sustainable pace in performance of work, 56
  validated learning in, 44–45
  value-centric delivery in, 55
  variability and uncertainty and, 32
  work in process (WIP) and, 48
Agile Retrospectives (Derby and Larsen), 379
All-at-once product development
  defined, 402
  in origins of Scrum, 3
All-before-any approach
  defined, 402
  to work in process, 48
Anticipatory process. *See* plan-driven development
Appelo, Jurgen, 230
Approach
  defined, 402
  essential scrum includes, xxix
  realizing Scrum practices, 13
Artifacts
  defined, 402
  just-in-time approach to creating work products, 43
  managing inventory of planning artifacts, 251–252
  potentially shippable product increment as, 25–26
  product backlog as, 18–19
  sprint backlog as, 18
Asset teams, 214. *See* also Component teams
Assets
  measuring progress by asset validation, 54–55
  monitoring and reports focusing on asset validation, 236
Assumptions
  calculating release costs and, 325–326
  defined, 402
  replanning based on validation of, 251
  validated learning and, 45, 304
Atmosphere, setting for sprint retrospective, 382
ATTD (Acceptance-test-driven development), 85–86, 402
Attendance
  sprint retrospective issues, 392
  sprint review issues, 372–373
Authority, levels of (Appelo), 230
Automated testing, 149, 355–356

B

Batch size
  in agile development, 48–49
  comparing plan-driven development with agile development, 60
  defined, 403
Benefits of Scrum, 4–5
Best practices, 396–397
Blame
  creating blame-free atmosphere for sprint retrospective, 382
  sprint retrospective issues, 393
"Boil-the-ocean" projects, 65
Boy Scout rule
  defined, 403
  servicing technical debt when you happen upon it, 158–159
Budget constraint
  fixed date approach, 313–314
  fixed everything approach, 311–312
  fixed scope and date approach, 312–313
  fixed scope approach, 313
  in release planning, 311

Burndown chart
defined, 403
for fixed-scope release, 327–328
sprint, 357–359
Burnup charts
defined, 403
for fixed-scope release, 328–329
sprint, 359–360
Business
engagement pattern with, 170
making technical debt visible at business level, 153–154
ScrumMaster skills related to business domain, 188

C

Cadence
benefits of consistent duration of sprints, 67–68
defined, 403
Capacity
defined, 403
in Kanban, 10
measuring in effort-hours, 342–343
measuring in story points, 342
sprint planning, 22, 340–342
underutilization of, 351
Card format, for user stories, 83–84
Ceremony
defined, 403
minimizing unnecessary, 57–58, 368
plan-driven development compared with agile development, 60
Change
consequences of, 70–71
handling cost of, 40–43
maintaining sprint goals despite, 69–73
managing, 79
overcoming the status quo, 398–399
as product backlog item, 101
Change agent, ScrumMaster as, 187, 191
Chaotic domain
in Cynefin framework, 6–7, 9
defined, 403
Checkpoints, short duration sprints providing frequent, 66–67
Chickens and pigs, 25, 403
Chief product owner, 183–184, 404
Clarification, of sprint goals, 69–70
Closing retrospectives, 390–391
Closure, timeboxing enforcing, 63
CMMI maturity model, 395
Coach
a day in the life of ScrumMaster, 190
ScrumMaster as, 16, 185–186
Code refactoring. *See* Refactoring code
Cohn, Mike, xxv, xxxiii–xxxiv, 129–130, 206, 395, 397–398
Collaboration
benefits of face-to-face communication, 205
cross-cluster, 240–241
funneling through project manager, 242–243
of product owner with development team, 170–171
of product owner with stakeholders, 171
ScrumMaster skills, 189
in sprint review, 370
Commercial development projects, 177–179
Commercial-off-the-shelf (COTS), 8
Commitment
as basis of sprint goals, 69
change and, 71
checking if realistic, 344–345
defined, 404
of development team, 207–208
estimates contrasted with, 124–125
sporadic attendance and, 372–373
sprint planning outcomes and, 17–18, 346
Communication
channels between teams, 240–241
development team skills/characteristics, 204–205
facilitating shared understanding, 81–82
product owner skills, 172–173
of progress in fixed-date release, 329–330
of progress in fixed-date release planning, 327–329
ScrumMaster skills, 189–190
of sprint execution progress, 356
transparency of, 205–206
Competence, managers developing team members, 231–232

Complaints, sprint retrospective issues, 393
Complex adaptive systems
 defined, 404
 flight pattern of geese illustrating, 199
 Scrum origins and, 3
Complex domain
 in Cynefin framework, 6–8
 defined, 404
Complexity, Scrum providing confidence in handling, 6
Complicated domain
 in Cynefin framework, 6–8
 defined, 404
Component teams
 combining with feature teams, 217–218
 defined, 404
 feature teams compared with, 213–216
 product owner for, 177, 180–181
 when defining a product, 116
 when to use, 216
Components
 development of, 213–214
 development projects, 180–181
 integration and testing, 46–47
Conditions of satisfaction, 404. *See also* Acceptance criteria
Confidence, acquiring in sprint planning, 344–346
Confidence threshold
 defined, 404
 for product planning, 290, 298
 targeting realistic, 300–302
Confirmation information, in user stories, 85–86
Conformance to plans, in plan-driven development, 54, 60, 250
Constraints, in release planning
 fixed date, 313–314
 fixed everything approach, 311–312
 fixed scope, 313
 fixed scope and date, 312–313
 inputs to sprint planning, 338
 overview of, 311
 updating, 314
 variable quality and, 313–314
Continuous deployment (delivery)
 defined, 404
 planning release of features to customers, 308
 product roadmap and, 260
Continuous improvement
 no end state in Scrum, 395
 sprint retrospective and, 375
 while applying iterative and incremental development, 34
Continuous integration
 defined, 404
 helping work at a sustainable pace, 209
 technical practice, 355
 use of good practices prevents accrual of technical debt, 149
Contracts, limitations of fixed-price contracts, 235
Conversations
 in development of user stories, 84–85
 facilitating shared understanding, 81–82
Coordination. *See also* Collaboration
 cross-cluster, 240–241
 funneling through project manager, 242–243
Cost of delay
 Agile principles and, 52–54
 comparing plan-driven development with agile development, 60
 defined, 404
 portfolio planning and, 271–274
 for properly quantifying technical debt economics, 150–152
Costs
 calculating in release planning process, 325–326
 handling cost of change, 40–43
 of idle work, 52–53
 Scrum reducing, 6
 technical debt impacting development costs, 142–143
COTS (Commercial-off-the-shelf), 8
Cross-cluster collaboration, 240–241
Cross-functional diversity and sufficiency, of development team, 200–201
Cross-functional teams
 in agile development, 2
 defined, 405
 feature teams, 213

high-bandwidth communication and, 205
managers forming, 228–229
quality built-in to development process, 56–57
vs. role-specific teams, 195–196
Cunningham, Ward
on refactoring, 149
on technical debt, 139–140
Customer satisfaction
Scrum benefits, 6
technical debt decreasing, 144
Customer uncertainty
defined, 405
reducing, 36
Customers
engagement pattern with, 170–171
planning release of features to, 307
product owner understanding needs of, 166
repaying technical debt while performing customer-valuable work, 160–162
value-centric delivery focused on needs of, 55
value of user stories to, 90
Cynefin framework
defined, 405
for situation-appropriate decision making, 6–7

D

Daily planning, 258, 264–265
Daily scrum
approaches to, 397
daily planning during, 264–265
defined, 405
sprint execution and, 354
sprints and, 23–25
when grooming occurs and, 108
Daily stand-up. *See* Daily scrum
Date constraint
fixed date approach, 313–314
fixed everything approach, 311–312
fixed scope and date approach, 312–313
fixed scope approach, 313
in release planning, 311
Deadlines, resulting in technical debt, 144–145
Decision making
economic filter for go/no-go decision making, 275–276
illusion of certainty and, 303
incremental/provisional approach to funding, 304–305
keeping options open, 37–38, 249
plan-driven development compared with agile development, 59
by product owner, 173
which work needs to be done, 353–354
which work to start, 352
DEEP (Detailed appropriately, Emergent, Estimated, and Prioritized)
appropriate detail, 101–102
characteristics of good product backlog, 101
defined, 405
emergent nature of, 102
prioritization in, 103–104
size estimates in, 102–103
Defects
compounding, 142
defining when sprint is complete or done and, 75
as product backlog item, 100–101
as technical debt, 139
Defined process, in plan-driven development, 32
Definition of done
checklist, 74–76
confidence threshold as envisioning, 290
defined, 405
development team needs skills to meet, 200–201
evolving over time, 76–77
for managing technical debt, 149–150
no end state in Scrum, 395
nonfunctional requirements for inclusion in, 93
overview of, 25–26
preventing accrual of technical debt, 150
versus acceptance criteria, 77
what work needs to be done, 353–354
Definition of ready
acceptance criteria, 169
checklist, 109–110
defined, 405

Definition of ready (*continued*)
- overview, 108–110
- product backlog items for sprint planning, 336
- providing boundaries for work at the task level, 353
- selecting product backlog items and, 344
- understanding how to demonstrate items at sprint review, 370

Delays, cost of. *See* Cost of delay

Delegation, as means of empowering teams, 230

Deliverables. *See also* Potentially shippable product increments (PSIs)
- short duration sprints and, 65–66
- technical debt increasing time to delivery, 142

Demonstration aspect, of sprint review, 368, 370

Design flaws, as technical debt, 139

Detail
- in product backlog, 101–102
- in user stories, 86–88

Development team
- accountability of product owner to, 173
- communication skills of, 204–205
- cross-functional diversity and sufficiency of, 200–201
- daily scrum for, 23
- defined, 405
- focus and commitment of, 207–208
- grooming product backlog, 105–106
- long-lived nature of, 209–211
- multiple teams with one product backlog, 115–116
- Musketeer attitude (all for one, one for all), 203–204
- one team with multiple product backlogs, 117–118
- overview of, 195
- participant in product planning, 288–289
- participant in release planning, 308
- participant in requirements conversation, 84
- participant in sprint execution, 348
- participant in sprint planning, 335
- participant in sprint retrospective, 377
- participant in sprint review, 364–365
- PBI estimation, 123–124
- product owner collaborating with, 166, 170–171
- responsibilities of, 196–197
- role of, 16
- role-specific teams, 195–196
- rules of Planning Poker and, 132
- as Scrum role, 16
- self-organizing nature of, 198–200
- small size of, 206
- sprint planning, 21–23
- sustainable pace in performance of work, 208–209
- T-shaped skills, 201–203
- technical practices for task performance, 355–356
- transparency of communication, 205–206
- when grooming occurs and, 107–108

Discussions
- in sprint review, 371
- when estimating, 121–122
- when writing user stories, 81–82, 85

Disorder domain
- in Cynefin framework, 6–7, 9
- defined, 406

Diversity, of development team, 200–201

Documentation
- conversations compared with, 81–82
- cost of delay example involving, 53–54
- in definition of done, 74
- lack of in VersionOne 2011 survey, 225
- plan-driven development as document-centric process, 57
- in Scrum development, 57–58
- supplementing user stories, 84

Domain skills, of product owner, 171–172

Done
- acceptance criteria compared with, 77
- checklist, 74–76
- defined, 406
- done vs. done done, 77–78
- evolution of definition of done over time, 76–77
- no end state in Scrum, 395
- sprint review confirming, 367–368
- value of strong definition of done in preventing accrual of technical debt, 149–150

Dot voting
defined, 406
selecting which insights to act on, 388
Duration, of sprints
calculating from estimated size and measured velocity, 119–120
consistency of, 67–68
short duration preference, 64–67
story points in calculation of, 128

E

Economic filter
defined, 406
for go/no-go decision making, 275–276
Economics
of abnormal sprint termination, 72–73
of aligning all teams to a single product backlog, 115–116
of change, 70–71
of component teams, 218
of developing a project plan per sprint, 349
focusing on short time horizon in product planning, 302
improved by fast feedback, 65
incremental/provisional funding, 304–305
learning fast and pivoting as necessary, 305
of long-lived teams, 210
managing, 167–168, 236
marginal economics applied to in-process products, 283–285
of product planning, 299–300
of release approach, 253
of rising development costs, 142
single versus multiple release, 252–253
of smaller, more frequent releases, 279–280
speed and efficiency and, 302–303
targeting realistic confidence threshold, 300–302
of technical debt, 150–152
validated learning, 303–304
Economies of scale, manufacturing vs. product development, 48
Effort/cost, scheduling portfolio backlog items and, 274
Effort hours
capacity in, 342–343
checking if commitment is realistic, 344–345
tasks in, 122
Emergent opportunities
defined, 406
embracing quickly, 278–279
Emotions seismograph
defined, 406
mining for insights, 385
sprint retrospective and, 384–385
Empirical process control (Schwaber and Beedle), 35, 406
End of life, not repaying technical debt for products approaching, 157
End uncertainty
defined, 406
reducing uncertainty, 36
Enjoyment, Scrum benefits, 6
Enterprise Transition Community (ETC), 398
Environment, managers' responsibilities
aligning internal groups, 234
aligning partners, 234–235
promoting agile values, 233–234
removing organizational impediments, 234
Envisioning. *See* Product planning
Epics
defined, 406
estimating, 103
in release train, 221
representing product backlog items, 294–295
size of user stories and, 86–88
story mapping technique and, 96
Errors, setting limit or bound on, 65
Estimable criteria, INVEST, 91–92
Estimation
accuracy vs. precision, 125, 274–275
commitments contrasted with, 124–125
defined, 407
development team in PBI estimation, 123–124
ideal days for measurements in, 128–129
overview of, 119–120
of PBIs, 121
Planning Poker approach, 129–133
of product backlog, 121–122

Estimation (*continued*)
product owner and, 175
relative sizes vs. absolute sizes, 125–128
scale in, 130
story points for measurements in, 128
of tasks, 122
units for, 128
what and when of, 120–121
ETC (Enterprise Transition Community), 398
Event timeline
defined, 407
mining for insights, 385
sprint retrospective and, 384
Excitement/enthusiasm, short duration sprints rejuvenating, 65–66
Exercises
inputs for sprint retrospective, 380
selecting for use in sprint retrospective, 379
Experiments, knowledge-acquisition user stories, 93
Exploitation, 59, 407
Exploration
defined, 407
knowledge-acquisition user stories, 93–95
plan-driven development compared with agile development, 39–40, 59
External stakeholders
defined, 407
product owner collaborating with, 171
Extreme Programming (Beck and Andres), 355, 407

F

Face-to-face communication, 205
Facilitator
ScrumMaster as, 16
for sprint execution, 348
for sprint planning, 335–336
for sprint retrospective, 393
for sprint review, 368
Fail fast, 305, 407
Fast feedback. *See also* Feedback
defined, 407
early review and, 367
monitoring and reports aligned to, 236
organizing workflow for, 46–47
plan-driven development compared with agile development, 59
short duration sprints aiding, 64–65
technical debt and, 139
Fast pace, of development
fast delivery as Scrum benefit, 6
go fast but never hurry, 56
Feature teams
combining with component teams, 217–218
comparing with component teams, 213–216
defined, 407
product owner, 180-181
Features
defined, 407
as product backlog item, 100–101
release flow management and, 110–111
user stories and, 87–88
Feedback. *See also* Fast feedback
in iterative and incremental development, 34–35
learning fast and pivoting as necessary, 305
learning loops and, 45–46
organizing workflow for, 46–47
performance feedback given by managers, 232
plan-driven development compared with agile development, 59
in prioritization of iterations, 2
short duration sprints aiding, 64–65
sprint review and, 364–365
Fixed date constraint, in release planning, 313–314
Fixed-date release
calculating costs in, 325–326
communicating progress of, 329–330
defined, 408
overview of, 318–323
planning and, 67–68
Fixed everything constraint, in release planning, 311–312
Fixed-price contracts, limitations of, 235
Fixed scope and date constraints, in release planning, 312–313
Fixed scope constraint, in release planning, 313

Fixed-scope release
  calculating costs in, 325–326
  communicating progress of, 327–329
  defined, 408
  planning, 323–325
  product roadmap and, 260–261
Flow
  daily scrum in management of, 354
  deciding which work needs to be done, 353–354
  deciding which work to start, 352
  defined, 408
  managing in sprint execution, 349–350
  organizing task work, 352–353
  organizing workflow for fast feedback, 46–47
  parallel work and swarming, 350–352
  release flow management, 110–111
  Scrum used in organizing work flow, 3
  sprint flow management, 111–112, 349–350
  who does the work, 354
Focus
  of development team, 207–208
  of sprint retrospective, 378–379
Forecasts
  defined, 408
  terminology choices for sprint planning outcomes, 17–18
  velocity, 135
  vs. commitments, 346
Formality
  minimizing unnecessary, 57–58
  plan-driven development compared with agile development, 60
  unnecessary formality defined, 420
Framework, for Scrum
  activities and artifacts, 16–18
  closing review, 28
  core values, principles, and practices in, xxix
  daily scrum, 23–25
  defined, 408, 416
  overview of, 13
  practices, 14
  product backlog, 18–20
  roles, 14–16
  sprint execution, 23
  sprint planning, 21–23
  sprint results, 25–26
  sprint retrospective, 27–28
  sprint review, 26–27
  sprints, 20–21, 61
Frustration, technical debt resulting in, 144
Functional managers. *See* Managers
Funding, incremental/provisional approach to, 304–305

G

Gantt chart
  sprint execution and, 349
  of up-front plan, 250
Go/no-go decision making
  economic filter for, 275–276
  funding decisions, 299
Goals. *See also* Sprint goal
  managers providing team goals, 228
  no end state for, 395
Grenning, James, 129
Grooming
  defined, 408
  insight backlog, 390
  overview of, 104
  product backlog, 19, 315
  responsibilities of development team, 197
  responsibility of product owner, 169
  in Scrum framework, 17
  ScrumMaster working with product owner on, 190
  what it is, 104–105
  when does it occur, 106–108
  who does it, 105–106
Groups
  compared with teams, 209–210
  defined, 408
  managers role in aligning internal, 234

H

Happened-upon technical debt, 155, 158–159, 408
Harvesters (Goldberg and Rubin), 217
Hidden agendas, transparency and, 189
Hierarchical product backlogs, for large products, 114–115

High-bandwidth communication, development team skills/characteristics, 204–205
Hiring/firing authority, of managers, 229

I

Ideal days
- defined, 408
- measuring magnitude of PBI, 128–129
- as relative size measures, 20

Ideal hours
- defined, 408
- task estimated in, 122

Ideation, 288
Idle work
- comparing plan-driven development with agile development, 60
- defined, 409
- focusing on idle work not idle workers, 51–52, 281
- monitoring and reports focusing on, 236

Idle workers, 51–52, 281, 409
Impediments
- defined, 409
- managers removing, 234
- ScrumMaster removing, 187, 191

Implementable stories
- defined, 409
- size of user stories and, 87
- story mapping technique and, 97

In-process products
- defined, 409
- marginal economics applied to, 283–285
- overview of, 283
- portfolio planning and, 268

Incremental approach, to servicing technical debt, 159
Incremental development
- agile principles underlying Scrum, 33–35
- defined, 409
- short duration sprints rejuvenating participant excitement, 65

Incremental funding
- defined, 409
- economics of product planning, 304–305

Independent criteria, INVEST, 88–89
Inflow strategies, portfolio planning
- balancing product flow into/out of portfolio backlog, 276–278
- economic filter for go/no-go decision making, 275–276
- embracing emergent opportunities, 278–279
- overview of, 275
- small, frequent releases, 279–280

Information radiator. *See also* Communication
- defined, 409
- elements of, 356

Innovation accounting
- defined, 409
- metrics in, 236–237

Innovation waste, 90, 409
Insight backlog
- defined, 409
- grooming, 390
- inputs for sprint retrospective, 381
- as source of insights, 386

Insight cards, 386–387
Insights, in sprint retrospective
- identifying, 385–387
- inputs for sprint retrospective, 381
- insight backlog, 390
- selecting among, 388–389

Inspection
- daily scrum as inspect-and-adapt activity, 354
- defined, 409–410
- discovering your own path forward, 396
- leveraging variability, 35–36
- planning based in inspection and adaptation, 248
- responsibilities of development team, 197–198
- sprint retrospective and, 375
- sprint review and, 363

Integration
- of components, 46–47
- continuous integration practice, 149, 404
- defined, 410
- of improvement actions, 391
- release train approach (Leffingwell) and, 222

Integration management, technical debt and, 140
Integration tests, 75
Interference shield, ScrumMaster as, 187
Internal development projects, choosing product owner for, 176–177
Internal stakeholders
  defined, 410
  product owner collaborating with, 171
Internationalization, testing in definition of done, 75
Interrupt-driven work, Scrum not suited for, 9–10
Inventory
  defined, 410
  managing in agile development, 49–50
  managing planning artifacts, 251–252
  plan-driven development compared with agile development, 60
INVEST criteria, for user stories
  defined, 410
  estimable, 91–92
  independent, 88–89
  negotiable, 89–90
  overview of, 88
  sized appropriately, 92
  testable, 92
  valuable, 90–91
Investment, change impacting, 70–71
Iterative development
  in agile development, 2–3
  agile principles underlying Scrum, 33–35
  defined, 410

J

Jeffries, Ron, xxvii–xxviii, xxxiv, 83
JIT (just in time). *See* Just in time (JIT)
Just enough
  appropriate detail in product backlog, 101
  of predictive planning, 300
  requirements and, 79
Just in time (JIT)
  appropriate detail in product backlog, 101
  balancing predictive work with adaptive work, 43–44
  balancing up-front planning with just-in-time planning, 248
  creating work products, 43
  defined, 410
  keeping options open, 249
  requirements and, 79
  sprint planning, 335

K

Kanban
  defined, 410
  development process suited for interrupt driven work, 9–10
Katz, Ralph, 210
Kerth, Norm, 375, 379
Knowledge acquisition
  as product backlog item, 100–101
  sprint for, 298
  user stories, 93–95
Knowledgeable, ScrumMaster skills, 188
Known technical debt
  defined, 410
  repaying incrementally, 159
  repaying while performing customer-valuable work, 160–161
  servicing, 155–156

L

Last responsible moment (LRM) (Poppendieck and Poppendieck)
  defined, 411
  keeping options open, 37
Leadership
  managers providing in functional areas, 232–233
  product owner role, 15
Learning
  discovering your own path forward, 396
  economics of product planning, 305
  fast learning combined with pivoting, 254–255
  managers role in development of competence, 231–232
Learning loops
  aligning performance feedback to, 232
  concurrency of, 45–46
  defined, 411
Leffingwell, Dean, 272

Lifecycle profits
defined, 411
impact of cost of delay on, 54
optimizing scheduling for lifecycle profitability, 270–271
Longer-term planning. *See* Release planning
LRM (last responsible moment) (Poppendieck and Poppendieck)
defined, 411
keeping options open, 37

M

Man-hours, task estimated in, 122
Managers
aligning internal groups, 234
aligning partners, 234–235
changing team composition, 229
defining team boundaries, 227
developing team member competence, 231–232
empowering teams, 230–231
energizing team members, 231
fashioning teams, 226
forming teams, 228–229
maintaining team integrity, 233
managing economics, 236
monitoring measures and reports, 236–237
overview of, 225–226
participating in sprint retrospective, 377
project management responsibilities, 237–239
promoting agile values, 233–234
providing leadership in functional areas, 232–233
providing team goals, 228
removing organizational impediments, 234
systems perspective of, 235
when to retain separate project manager role, 239–243
Manufacturing. *See* Product manufacturing
Marginal economics, applied to in-process products, 283–285
Maturity models, not part of Scrum, 395
Means uncertainty
defined, 411
reducing uncertainty, 36
Measures (metrics)
of capacity, 342–343
managers monitoring, 236–237
Mess (Martin), terminology for technical debt, 140
Milestone-driven planning. *See* Release planning
Milestones, in short duration sprints, 66–67
Minimum marketable features (MMFs). *See* Minimum releasable features (MRFs)
Minimum releasable features (MRFs)
baseline values for actionable metrics, 2 37
defined, 411
defining product roadmap and, 295–296
determining in release planning, 309–310, 320
marginal economics applied to in-process products, 284
refining, 316
Minimum viable product (MVP). *See* Minimum releasable features (MRFs)
MMFs (minimum marketable features). *See* Minimum releasable features (MRFs)
Monitoring measures and reports, managers, 236–237
Motivation
managers role in energizing people, 231
product owner skills, 173
MRFs (Minimum releasable features). *See* Minimum releasable features (MRFs)
Multilevel planning
daily planning, 264–265
overview of, 257–258
portfolio planning, 259
product planning, 259–261
release planning, 261–263
sprint planning, 263
Multiple teams
coordinating using release train approach, 220–223
coordinating using scrum of scrums, 218–220
Multitasking, cost of, 350–351
Musketeer attitude (all for one, one for all)
defined, 411
development team skills/characteristics, 203–204

Must-have features
    defined, 411
    defining product roadmap and, 295
    determining in release planning, 320
    in focusing on short time horizon, 302
    release flow management and, 110–111
MVP (Minimum viable product). *See* Minimum releasable features (MRFs)

N

Naive technical debt, 140, 412
Negotiable criteria, INVEST, 89–90
New products
    portfolio planning. *See* Portfolio planning
    product planning. *See* Product planning
Nice-to-have features
    defined, 412
    release flow management and, 110–111
    release planning, 314, 320
Nonaka, Ikujiro, 3
Nonfunctional requirements, 93, 412

O

Objective data
    gathering for sprint retrospective, 379
    inputs for sprint retrospective, 380
One-part approach, to sprint planning, 339–340
One-product-one-product-backlog rule
    large products and, 114–115
    multiple teams and, 115–116
    what is a product and, 113–114
Opportunities, embracing emergent opportunities quickly, 278–279
Options, keeping options open, 37–38, 249
Ordered, terminology for product backlog sequences, 20
Organizational impediments. *See* Impediments
Outflow strategies, portfolio planning
    establishing WIP limits, 281–282
    focusing on idle work not idle workers, 281
    overview of, 280
    waiting until entire team is in place, 282–283
Outsourcing
    choosing product owner for outsourced projects, 180
    limitations of fixed-price contracts, 235
Overtime, impact on quality and velocity, 136–137

P

Parallel work, sprint execution and, 350–352
Participants
    in portfolio planning, 268
    in product planning, 288–289
    in release planning, 308
    in sprint execution, 348
    in sprint planning, 335–336
    in sprint retrospective, 377–378
    in sprint review, 364–365
Partners, managers aligning, 234–235
Path forward
    discovering, 396
    no end state in Scrum, 395
    overcoming the status quo, 398–399
    sharing best practices, 396–397
    using Scrum to discover, 397–398
Patience, ScrumMaster skills, 189
Patton, Jeff, 96
PBI estimation
    accuracy vs. precision in, 125
    contrasting estimates with commitments, 124–125
    development team in, 123–124
    overview of, 121–122
    Planning Poker approach to, 129–130
    relative sizes vs. absolute sizes, 125–128
    units for, 128–129
PBIs (product backlog items). *See* Product backlog items (PBIs)
People skills, of product owner, 172–173
Perfectionism, avoiding unnecessary, 63
Performance
    definition of done and, 75
    definition of ready and, 110
    feedback given by managers, 232
    technical debt resulting in underperformance, 143
Performance principle, in agile development
    minimizing unnecessary formality, 57–58

Performance principle, in agile development (*continued*)
- overview of, 56
- quality built-in to development process, 56–57
- sustainable pace in performance of work, 56

Person-hours, task estimated in, 122
Personas (roles)
- defined, 412
- in user stories, 96

Pichler, Roman, 101
Pigs and chickens, 25, 412
Pipeline of requirements, product backlog as, 112
Pivoting
- defined, 412
- economics of product planning, 305
- envisioning, 288–289
- innovation accounting, 237
- marginal economics applied to in-process products, 283–284
- planning and, 254–255

Placeholders
- product backlog items (PBIs) as requirements placeholder, 80–81
- user stories marking exploration work, 94

Plan-driven development
- agile principles compared with, 59–60
- all-before-any approach to work in process, 48
- assumptions in, 45
- beliefs regarding, 30
- costs of change in, 43
- defined, 412
- defined process in, 32
- as high-ceremony approach, 57
- integration and testing components in, 46–47
- limitations regarding re-planning, 54
- linear approach to uncertainty in, 36
- phase orientation vs. customer expectations, 54–55
- requirements in, 79
- risks related to up front planning, 38–39
- sequential approach compared with agile's exploratory approach, 39–40
- types of, 29
- variability not accounted for in, 35

Planning
- accepting that you can't get it right up front, 38–39
- adapting to real-time information, 54
- consistent duration of sprints simplifying, 67–68
- daily planning, 264–265
- a day in the life of product owner, 175
- multilevel approach to, 257–258
- portfolio planning. *See* Portfolio planning
- product owner participating in, 168–169
- product planning. *See* Product planning
- release planning. *See* Release planning
- short duration sprints aiding, 64
- sprint execution, 349
- sprint planning. *See* Sprint planning
- sprints, 21–23

Planning Poker
- defined, 412
- how to play, 131–133
- overview of, 129–130
- scale in assigning estimates, 130

Planning principles
- emphasis on small, frequent releases, 252–254
- focus on adapting and replanning rather than conforming, 249–251
- keeping options open, 249
- learning fast and pivoting as necessary, 254–255
- managing inventory of planning artifacts, 251–252
- not assuming up-front plans are right, 248
- overview of, 247–248
- up-front planning should be helpful not excessive, 248–249

Platforms
- lack of experience resulting in technical debt, 140
- testing in definition of done, 75

PMI (Project Management Institute), 237–239
Point inflation, 138, 412
Pollinators (Goldberg and Rubin), 217
Portfolio backlog
- defined, 413

estimating, 121
inflow strategies, 275–280
outflow strategies, 280–283
portfolio planning and, 267, 269
in-process strategies, 283–285
release train approach (Leffingwell), 221
Portfolio planning
balancing product flow into/out of portfolio backlog, 276–278
calculating cost of delays, 271–274
defined, 413
economic filter for go/no-go decision making, 275–276
embracing emergent opportunities, 278–279
establishing WIP limits, 281–282
estimating for accuracy not precision, 274–275
focusing on idle work not idle workers, 281
managing economics of, 236
marginal economics applied to in-process products, 283–285
in multilevel planning, 259
optimizing scheduling for lifecycle profitability, 270–271
overview of, 267
participants in, 268
planning level details for, 258
process of, 268–270
product owner participating in, 168
small, frequent releases in, 279–280
strategies for in-process products, 283
strategies for inflow, 275
strategies for outflow, 280
strategies for sequence of products, 270
timing of, 267
waiting until entire team is in place, 282–283
Potentially shippable product increments (PSIs)
defined, 413
defining when sprint is complete or done, 74–78
as input to sprint review, 368–369
inspecting and adapting during sprint review, 363
as outcome of iterative process, 2–3
planning release of features to customers, 307
release train approach (Leffingwell) and, 220, 222–223
sprint results, 25–26
Practices
activities. *See* Activities
artifacts. *See* Artifacts
defined, 413
roles. *See* Roles
rules. *See* Rules
in Scrum framework, 14
Pragmatism
no-goal-altering-change rule and, 72
Pragmatic Marketing Framework, 178–179
Precision
defined, 413
vs. accuracy in estimating, 125, 274–275
Prediction
balancing predictive work with adaptive work, 43–44
just enough predictive planning, 300
plan-driven development compared with agile development, 59
technical debt decreasing predictability, 143
timeboxing improving predictability, 64
Prediction and adaptation principle, in agile development
accepting that you can't get it right up front, 38–39
adaptive, exploratory approach in, 39–40
balancing predictive work with adaptive work, 43–44
handling cost of change, 40–43
keeping options open, 37–38
overview of, 37
pivoting and, 254–255
Predictive process. *See* Plan-driven development
Prescriptive process. *See* Plan-driven development
Principle of least astonishment
defined, 413
transparency of communication and, 206
Principles. *See* Agile principles
Prioritization
in product backlog, 103–104
sporadic attendance and, 372–373

Prioritization (*continued*)
  terminology choices for product backlog sequences, 20
  timeboxing enforcing, 62
Process authority, ScrumMaster as, 186–187
Process-centric development, 60
Process structure, 59
Product backlog
  in agile development, 1–2
  appropriate detail in, 101–102
  conditions of satisfaction, 77
  creating high-level list in product planning process, 294–295
  deciding which and how many to form, 112–113
  defined, 413
  definition of ready, 108–110
  determining what is a product, 113–114
  economics, 168
  emergent nature of, 102
  estimating, 121–122
  grooming, 104–108, 369, 413
  as input to sprint planning, 337
  large products with hierarchical backlogs, 114–115
  mapping to releases, 263
  multiple teams with one product backlog, 115–116
  one team with multiple product backlogs, 117–118
  overview of, 99
  PBIs in, 100–101
  prioritization in, 103–104
  product owner responsible for grooming, 169
  product planning and, 259–260
  release flow management, 110–111
  release planning and, 320–321
  representing technical debt, 155
  in Scrum framework, 18–20
  size estimates in, 102–103
  sprint flow management, 111–112
  sprint planning and, 17, 21–23
Product backlog items (PBIs)
  appropriate detail, 101–102
  creating high-level list in product planning process, 294–295
  deciding which work to start, 352
  defined, 413
  definition of ready, 109–110
  emergent nature of, 102
  estimating. *See* PBI estimation
  grooming tasks related to, 104–105
  mapping to sprints, 316–318
  measuring velocity and, 133
  organizing task work, 352–353
  overview of, 100–101
  parallel work and swarming, 350
  as placeholders for requirements, 80–81
  prioritizing, 103–104
  representing technical debt, 155
  selecting in sprint planning, 343–344
  sign-offs and, 372
  size estimates, 102–103
  user stories adding detailed items, 315
Product development
  benefits of Scrum for, 10
  calculating duration from estimated size and measure velocity, 119–120
  defined, 413
  economies of scale, 48
  focusing on idle work not idle workers, 51–52
  inventory management, 50
  vs. product manufacturing, 32–33
Product manufacturing
  comparing plan-driven development with agile development, 59
  economies of scale, 48
  inventory management, 49–50
  vs. product development, 32–33
Product owner
  accountability of, 173
  chief product owner, 183–184
  collaborating with development team, 170–171
  collaborating with stakeholders, 171
  combining with other roles, 181–182
  for commercial development, 177–179
  for component development, 180–181
  creating/verifying acceptance criteria, 169–170
  a day in the life of, 174–176
  deciding if work is done, 367
  decision making by, 173
  defined, 414

domain skills of, 171–172
function relative to estimation process, 123
grooming product backlog, 105–106, 169
for internal development, 176–177
managing economics, 167–168
for outsourced development, 180
overview of, 165–166
in overview of Scrum roles, 15–16
participant in product planning, 288–289
participant in product portfolio, 268
participant in requirements conversation, 84
participant in sprint execution, 348
participant in sprint planning, 335
participant in sprint retrospective, 377
participant in sprint review, 364–365
people skills of, 172–173
planning functions of, 168–169
principal responsibilities of, 166
proxy product owner, 183
rules of Planning Poker, 132
in sprint planning, 21–22
team approach to, 182–183
understanding value of technical stories, 90–91
who should fill this role, 176
Product owner proxy, 183, 414
Product planning
creating product backlog, 294–295
a day in the life of product owner, 175
defined, 414
defining product roadmap, 295–297
economic filter for go/no-go decision making, 275–276
economic sensibility in, 299–300
incremental/provisional funding in, 304–305
learning fast and pivoting as necessary, 305
in multilevel planning, 259
new product example, 290–291
other types of work in, 298–299
overview of, 287
participants in, 288–289
planning level details for, 258
process of, 290
product backlog and, 259–260
product owner participating in, 168–169
product roadmap and, 260–261
product vision, 259, 291–294
short time horizon as focus of, 302
speed and efficiency of, 302–303
targeting realistic confidence threshold, 300–302
timing of, 287–288
validated learning in, 303–304
Product roadmap
defining, 295–297
definition of, 414
product planning and, 260–261
release planning and, 262–263
Product vision. *See* Vision
Productivity, multiple projects and, 207
Products
atrophy of appeal due to technical debt, 143
defined, 413
determining what is a product, 113–114
development team responsible to inspect and adapt, 197
large products with hierarchical backlogs, 114–115
not repaying technical debt for products nearing end of life, 157
not repaying technical debt for products with short life, 157–158
planning new. *See* Product planning
portfolio of new. *See* Portfolio planning
Program backlog, 221
Progress
communicating in fixed-date release, 329–330
communicating in fixed-scope release, 327
comparing plan-driven development with agile development, 60
of sprint execution, 356
timeboxing demonstrating, 62–63
Progress principle, in agile development
adapting to real-time information and replanning based on, 54
measuring progress by validating working assets, 54–55
overview of, 54
value-centric delivery in, 55
Progressive refinement strategy
applying to requirements, 82
defined, 414
level of detail, 86

Project chartering, 299, 414. *See also* Product planning
Project inception, 299. *See also* Product planning
Project initiation, 299. *See also* Product planning
Project Management Institute (PMI), 237–239
Project managers. *See also* Managers
responsibilities of, 237–239
when to retain separate project manager role, 239–243
Project Retrospectives (Kerth), 375, 379
Proof of concept, 93
Prototypes
knowledge-acquisition user stories, 93
not repaying technical debt for throwaway prototypes, 157
PSIs. *See* Potentially shippable product increments (PSIs)

Q

Quality
building in to development process, 56–57
comparing plan-driven development with agile development, 60
influenced by long-lived teams, 210
overtime and, 137
pressure to meet a deadline affects, 144–148
reduced due to working on too many items in parallel, 350–351
release constraints, 311
team diversity leads to, 201
traditional project management responsibility, 238
variability due to constraints, 313–314
Questioning ability, ScrumMaster skills, 188–189
Queue
defined, 414
impact of utilization on queue size (delay), 52–53
portfolio backlog and, 280

R

Range of velocity, calculating, 134–135
Real-time information, adapting to and replanning based on, 54
Reckless debt (Fowler), 140
Refactoring code
defined, 414
as means of paying down technical debt, 141
use of good practices prevents accrual of technical debt, 149
Reinertsen, Donald G.
on batch-size issues, 48–49
on cost of delay, 53
on lifecycle profits, 270
Relative size measures
in cost evaluation, 20
defined, 415
vs. absolute sizes in estimation, 125–128
Release goal
communicating progress using burndown chart, 327–328
communicating progress using burnup chart, 359
defined, 415
economics of, 167
grooming product backlog and, 315
product roadmap and, 296
Release planning
calculating costs in, 325–326
communicating progress in fixed-date release, 329–330
communicating progress in fixed-scope release, 327–329
constraints on release, 311
a day in the life of product owner, 175
defined, 415
defining product roadmap and, 296
emphasis on small, frequent releases, 252–254
fixed date constraint, 313–314
fixed-date release planning, 318–323
fixed everything approach, 311–312
fixed scope and date constraints, 312–313
fixed scope constraint, 313
fixed-scope release planning, 323–325
flow management and, 110–111
grooming product backlog, 315
initial grooming during, 107
managing economics of, 167–168
in multilevel planning, 261–263
overview of, 307–308
participants in, 308

planning level details for, 258
process of, 309–311
refining MRFs, 316
sprint mapping, 316–318
technical debt and, 140
timing of, 308–309
updated plan as output of sprint review, 369
updating constraints, 314
variable quality constraint, 313–314
velocity and, 133
Release train (Leffingwell)
coordinating multiple teams using, 220–223
defined, 415
Releases
defined, 415
small, frequent releases in portfolio planning, 279–280
Replanning, as focus of planning rather than conformance, 249–251
Reports, managers monitoring, 236–237
Requirements
card format for user stories, 83–84
confirmation information in user stories, 85–86
conversations facilitating shared understanding, 81–82
conversations in development of user stories, 84–85
estimatable criteria for user stories, 91–92
gathering user stories, 95
independent criteria for user stories, 88–89
INVEST criteria applied to user stories, 88
knowledge-acquisition user stories, 93–95
level of detail in user stories, 86–88
negotiable criteria for user stories, 89–90
nonfunctional, 93
overview of, 79–80
placeholders for, 80–81
progressive refinement of, 82
sized appropriately criteria for user stories, 92
story mapping technique and, 96–98
testable criteria for user stories, 92
user stories and, 83
valuable criteria for user stories, 90–91
workshop for writing user stories, 95–96
Resource managers, 229. *See also* Managers
Responsibilities, of development team,
groom the product backlog, 197
inspect and adapt each day, 197
inspect and adapt the product and process, 197
perform sprint execution, 196
plan the sprint, 197
Responsibilities, of product owner
collaborating with development team, 170–171
collaborating with stakeholders, 171
creating/verifying acceptance criteria, 169–170
grooming product backlog, 169
managing economics, 167–168
participating in planning, 168–169
Responsibilities, of ScrumMaster,
change agent, 187
coach, 185
impediment remover, 187
interference shield, 187
process authority, 186–187
servant leader, 186
Retrospectives, 375. *See also* Sprint retrospective
Return on investment (ROI)
cost of delays and, 271–272
responsibility of product owner for ensuring, 168
Scrum benefits, 6
short duration sprints improving, 65
small, frequent releases improving, 252, 254
Ries, Eric, 44, 157, 236, 254–255
Risk
associated with setting the confidence threshold, 301
assumptions and, 45
defined, 415
of fixed-price contracts, 180
of misinterpretation using ideal days, 128–129
small batch sizes reduce, 49
traditional project management responsibility, 238
Roadmap. *See* Product roadmap
ROI. *See* Return on investment (ROI)
Role-specific teams, compared with cross-functional teams, 195–196

Roles
combining product owner with other, 181–182
combining ScrumMaster with other, 192–193
defined, 415
development team, 16
overview of, 14–15
product owner, 15–16
ScrumMaster, 16
Roles (personas), in user stories, 96
Rolling lookup-ahead planning (Cohn), 318
Rules
allocate-up-to-ten-percent-capacity-for-grooming rule, 106
avoid-technical-debt-specific-sprints rule, 159
Boy Scout rule, 158–159, 403
consistent-duration sprints rule, 67
defined, 415
development-team-should-be-between-five-and-nine-people rule, 206
development-team-should-be-long-lived rule, 210
involve-all-team-members-in-story-writing rule, 294
no-goal-altering-change rule, 20, 72
one-hour-per-sprint-week rule, 367
one-product-one-product-backlog rule, 114–116
people-who-do-the-work-provide-the-estimates rule, 123
Scrum practices, 14
start-only-what-you-can-finish rule, 344
tasks-should-be-no-more-than-eight-hours rule, 338
teams-should-handle-their-own-coordination rule, 239–240

S

Safety, setting atmosphere for sprint retrospective, 382
Scale
in assigning estimates, 130
multiple small teams vs. single large team, 218
release train, 220–223
scrum-of-scrums, 218–220
Schedules
attendance issues and, 392
benefit of small batch sizes on, 49
predictable Scrum activities, 68
for sprint review, 366–367
Scheduling strategies, portfolio planning
calculating cost of delays, 271–274
estimating for accuracy not precision, 274–275
optimizing for lifecycle profitability, 270–271
overview of, 270
Schwaber, Ken, xxix–xxx, 3
Scope constraint
fixed date approach, 313–314
fixed everything approach, 311–312
fixed scope and date approach, 312–313
fixed scope approach, 313
in release planning, 311
Scrum framework. *See* Framework, for Scrum
"The Scrum Guide" (Sutherland and Schwaber), xxix–xxx
Scrum introduction
benefits to Genomica, 4–5
benefits to organizations, 5–7
Cynefin framework and, 6–10
framework overview, 13–14
origins of, 3
what it is, 1–3
Scrum of scrums (SoS)
for coordinating multiple teams, 206, 218–220
defined, 416
Scrum team
defined, 416
development team. *See* Development team
product owner role. *See* Product owner
roles of, 14–15
ScrumMaster. *See* ScrumMaster
ScrumMaster
combining with other roles, 192–193
a day in the life of, 190–191
defined, 416
facilitating daily scrum, 24
facilitating sprint review, 368
full vs. part time for, 192
function relative to estimation process, 123
grooming product backlog, 105–106
overview of, 185

in overview of Scrum roles, 16
participant in product planning, 288–289
participant in sprint execution, 348
participant in sprint planning, 335–336
participant in sprint retrospective, 377
participant in sprint review, 364–365
responsibilities of, 185–187
scrum of scrums and, 219
skills of, 188–190
in sprint planning, 21–22
sprint retrospective issues, 393
who should fill this role, 191–192
Scrummerfall, 34, 421
Self-fulfilling prophecy, 41–42
Self-organization
defined, 416
by development team, 198–200
sprint execution, 348
undermining, 231
Sequential development. *See* Plan-driven development
Servant leader
defined, 416
ScrumMaster as servant leader of Scrum team, 186
Servicing technical debt
Boy Scout rule for, 158–159
incrementally, 159
overview of, 155–156
paying high-interest debt first, 160
reasons for not repaying, 157–158
while performing customer-valuable work, 160–162
Shared context
creating for sprint retrospective, 382–384
emotions seismograph as aid in creating, 384–385
event timeline as aid in creating, 384
mining for insights, 385
Shippable product. *See* Potentially shippable product increments (PSIs)
Sign-offs, sprint review issues, 372
Silent grouping exercise
for clustering insights, 386
defined, 416
Simple domain
in Cynefin framework, 6–8
defined, 417
Single-piece flow, 48, 417
Six Sigma, 8
Size
in cost evaluation related to product backlog, 20
estimates, 102–103
Skills
inputs to sprint planning, 338
managers role in development of competence, 231–232
of product owner, 171–173
of ScrumMaster, 188–190
technical practices for task performance, 355–356
Small criteria, INVEST, 92
Small teams
favored for Scrum development, 206
high-bandwidth communication and, 205
SMEs (Subject matter experts), 169
Software development, issues related to, 5
Solutions
benefits of Scrum for, 4
defined, 417
faster and better, 201
innovative, 32
Specialists, on development team, 202
Specification by example, 85, 417
Spikes, knowledge-acquisition user stories, 93
Sprint backlog
defined, 417
estimating, 122
as input to sprint review, 368–369
sprint planning and, 264
Sprint burndown chart, 357–359
Sprint burnup chart, 359–360
Sprint demo, 368, 370, 417
Sprint execution
communicating progress of, 356
daily scrum and, 354
deciding which work to start, 352
determining which work needs to be done, 353–354
flow management and, 111–112, 349–350
organizing task work, 352–353
overview of, 23, 347
parallel work and swarming, 350–352
participants in, 348
performed by development team, 196–197
planning, 349
process of, 348–349

Sprint execution (*continued*)
sprint burndown chart and, 357–359
sprint burnup chart and, 359–360
task board and, 356–357
technical practices for task performance, 355–356
timing of, 347
who does the work, 354
Sprint goal
defined, 417
inputs to sprint planning, 338
inputs to sprint review, 368–369
maintaining despite changes, 69–73
refining, 346
selecting product backlog items that align with, 343–344
setting in planning process, 21
Sprint maps, in release planning, 310, 316–318
Sprint planning
acquiring confidence, 344–346
a day in the life of product owner, 175
defined, 417
determining capacity in, 340–343
finalizing commitment, 346
managing economics of, 168
in multilevel planning, 263
one-part approach to, 339–340
overview of, 21–23, 335
participants in, 335–336
planning level details for, 258
process of, 336–338
product owner participating in, 169
refining sprint goal, 346
responsibilities of development team, 197
selecting product backlog items, 343–344
terminology choices for sprint planning outcomes, 17–18
timing of, 335
two-part approach to, 338–339
Sprint results. *See* Potentially shippable product increments (PSIs)
Sprint retrospective
approach to, 380–382
closing the retrospective, 390
creating shared context for, 382–384
deciding among actions, 389–390
defined, 417
defining focus of, 378–379
determining actions, 387–388
emotions seismograph in, 384–385
event timeline in, 384
follow through on, 391–392
gathering objective data, 379
identifying insights, 385–387
insight backlog, 390
issues related to, 392–393
overview of, 27–28, 375–377
participants in, 377–378
prework needed for, 378
responsibilities of development team, 197
selecting among insights, 388–389
selecting exercises for use in, 379
setting atmosphere for, 382
structuring, 380
Sprint review
adapting based on, 371
approach to, 368–369
attendance issues, 372–373
confirming sprint work is done, 367–368
defined, 418
demonstration aspect of, 370
determining facilitator for, 368
determining who to invite, 366
discussions in, 371
for large development projects, 373
overview of, 26–27, 363–364
participants in, 364–365
preparing for demonstration, 368
prework needed for, 365–366
responsibilities of development team, 197
scheduling, 366–367
sign-offs, 372
summarization of sprint goal and sprint results, 369–370
when grooming occurs and, 108
Sprintable stories
defined, 417
size of user stories and, 87
story mapping technique and, 96
Sprints
abnormal termination of, 72–73
consistent duration of, 67–68
daily scrum and, 23–25
defined, 417

defining when complete or done, 74–78
iterative and incremental approach to development, 34
maintaining sprint goals despite changes, 69–73
organizing product planning into, 298
overview of, 20–21, 61–62
in Scrum framework, 17
short duration of, 64–67
timeboxing, 62–64
Staats, Bradley R., 210
Stakeholder value
areas of, 292–294
defined, 418
Stakeholders
accountability of product owner to, 173
defined, 418
defining product backlog, 18
getting feedback in agile development, 2
participant in grooming product backlog, 105–106
participant in product planning, 288–289
participant in portfolio planning, 268
participant in release planning, 308
as participant in requirements conversation, 84
participant in sprint retrospective, 377
participant in sprint review, 364–365
product owner collaborating with, 166, 171
Start/end dates. *See* Timeboxing
Start-only-what-you-can-finish rule, 344
Stories. *See* User stories
Story mapping technique (Patton), 96–98, 418
Story points
defined, 418
measuring capacity in, 342
measuring magnitude of PBI, 128
Planning Poker, 129-133
as relative size measures, 20
Strategic filters
defined, 418
economic filters, 275–276, 406
Strategic technical debt, 140, 418
Strategy planning, 257
Subject matter experts (SMEs), 169
Subjective data, communicating in sprint retrospective, 383
Subsystem teams, 214. *See also* Component teams
Succeeding with Agile (Cohn), xxv, 397
Summarization aspect, of sprint review, 369–370
Sustainable pace
defined, 418
of development team in performance of work, 56, 208–209
Sutherland, Jeff, xxix–xxx, 3
Swarming
defined, 418
sprint execution and, 351–352
T-shaped skills, 201–203
Synchronization
defined, 418
of multiple teams, 220, 222
System
system-level constraints expressed via nonfunctional requirements, 93
system-level focus in sprint retrospective, 385
testing in definition of done, 75
Systems perspective, of managers, 235

T

T-shaped skills
choosing who does the work and, 354
defined, 420
diversity of development team and, 201–203
finding balance in utilization of, 351
Tacit knowledge
defined, 419
of technical debt, 154
Takeuchi, Hirotaka, 3
Targeted technical debt
defined, 419
servicing, 155
Task board
for communicating sprint execution progress, 356–357
defined, 419
Tasks
defined, 419
during sprint planning, 22
estimating sprint backlog, 122
organizing task work, 352–353
technical practices for performance of, 355–356

TDD (test-driven development), 378, 419–420
Team structures
  coordinating multiple teams using release train approach (Leffingwell), 220–223
  coordinating multiple teams using scrum of scrums, 218–220
  feature teams vs. component teams, 213–218
  multiple team coordination, 218
  overview of, 213
Teams
  compared with groups, 209–210
  coordinating multiple, 218–220
  cross-functional. *See* Cross-functional teams
  defined, 419
  development. *See* Development team
  product owner as, 182–183
  swarming, 351
  unit of capacity, 233, 282
  use complete and engaged, 282–283
Teams, fashioning
  changing team composition, 229
  defining team boundaries, 227
  empowering teams, 230–231
  forming teams, 228–229
  overview of, 226
  providing team goals, 228
Teams, nurturing
  developing team member competence, 231–232
  energizing team members, 231
  maintaining team integrity, 233
  providing leadership in functional areas, 232–233
Technical debt
  Boy Scout rule for servicing, 158–159
  causes of, 144–148
  consequences of, 141–144
  defined, 419
  definition of done and, 76
  economics of, 150–152
  making visible at business level, 153–154
  making visible at technical level, 154–155
  making visible with balance sheet, 153-154
  managing, 148
  managing accrual of, 149–150
  overview of, 139–141
  reasons for not repaying, 157–158
  repaying high-interest debt first, 160
  repaying incrementally, 159
  repaying while performing customer-valuable work, 160–162
  servicing, 155–156
  variable quality and, 314
Technical knowledge, ScrumMaster skills, 188
Technical practices
  defined, 419
  for task performance, 355–356
  use of good practices prevents accrual of technical debt, 149
Technical stories
  defined, 419
  value of, 90
Technical work, as product backlog item, 100–101
Test-driven development (TDD), 378, 419–420
Test-first development, 353, 420
Testable criteria, INVEST, 92
Testing
  automated testing, 355–356
  components, 46–47
  excessive manual testing resulting in technical debt, 139
  myth that reduced testing can accelerate velocity, 145–147
  quality built-in to development process, 56–57
  release train approach (Leffingwell) and, 222
  types of tests, 75
Themes
  defined, 420
  story mapping technique and, 96
  user stories and, 87–88
Time-management
  act quickly, 302–303
  focusing on short time horizon in product planning, 302
  timeboxing for, 62
Timeboxing
  benefits of, 62–64
  defined, 420
  sprint retrospective and, 379
  start and end dates, 20–21

Timeline, creating event timeline for sprint retrospective, 384
Timing
 of portfolio planning, 267
 of product planning, 287–288
 of release planning, 308–309
 of sprint execution, 347
 of sprint planning, 335
Traditional development process. *See* Plan-driven development
Training
 a day in the life of ScrumMaster, 190
 managers role in development of competence, 231–232
Transparency
 defined, 420
 of development team, 205–206
 leveraging variability, 35–36
 of ScrumMaster, 189–190
Trust, managers role in establishing, 231
Two-part approach, to sprint planning, 338–339

U

Unavoidable technical debt, 140, 420
Uncertainty. *See also* Variability
 comparing plan-driven development with agile development, 59
 flow management and, 110
 reducing, 36–37
 type of, 36
Underutilization, of capacity, 351
Unintentional debt (McConnell), 140
Unit tests, 75
Units, for estimating product backlog items
 ideal days, 128–129
 story points, 128
Unknown unknowns
 defined, 420
 uncertainty and, 37
Unnecessary formality. *See* Formality
Unpredictable tipping point, characteristics of technical debt, 142
Up-front plans
 accepting that you can't get it right up front, 38–39
 focus on adapting and replanning rather than conforming, 249–251
 focus on making helpful not excessive, 248–249
 just enough predictive planning, 300
 not assuming they are right, 248
User role
 defined, 420
 user stories and, 83, 96
User stories. *See also* Requirements
 benefits of, 79
 card format for, 83–84
 confirmation information in, 85–86
 conversations in development of, 84–85
 defined, 421
 detailed product backlog items resulting from, 315, 320
 estimable criteria for, 91–92
 gathering, 95
 independent criteria for, 88–89
 INVEST criteria applied to, 88
 knowledge-acquisition stories, 93–95
 level of detail in, 86–88
 negotiable criteria for, 89–90
 nonfunctional requirements expressed via, 93
 overview of, 83
 for representing product backlog items, 294–295
 sized appropriately criteria for, 92
 story mapping techniques, 96–98
 testable criteria for, 92
 valuable criteria for, 90–91
 workshop for writing, 95–96
Utilization, relationship to queue size (delay), 52

V

Validated learning
 concurrent learning loops in, 45–46
 defined, 421
 organizing workflow for fast feedback, 46–47
 overview of, 44–45
 product planning and, 303–304
 validating important assumptions, 45

Validation, measuring progress by asset validation, 54–55
Valuable criteria, INVEST, 90–91
Value-centric delivery, 55, 60
Value-creation flow, managers role in
managing economics, 236
monitoring measures and reports, 236–237
systems perspective of, 235
Value-delivery-focused thinking, 353
Values
defined, 421
in Scrum framework, 13
Variability
defined, 421
embracing helpful variability, 32–33
inspection, adaptation, and transparency, 35–36
iterative and incremental approach to development, 33–35
overview of, 32
reducing uncertainty, 36–37
Velocity, of work
affecting, 135–137
calculating range of, 134–135
decreasing as technical debt increases, 147
defined, 421
fixed-scope-release burndown chart, 327
forecasting, 135
inputs to sprint planning, 337
misuse of, 137–138
myth that reduced testing can accelerate velocity, 145–147
overview of, 119–120
pressure to accelerate resulting in technical debt, 145
technical debt increasing time to delivery, 142
using predicted velocity to check if commitment is realistic, 344–345
what it is, 133–134
Vision
basing on areas of stakeholder value, 293–294
creating shared, 291–292
defined, 414
formats for, 292–293
product planning (envisioning) and, 259

W
Waste
defined, 421
innovation waste, 90
Waterfall development. *See also* Plan-driven development
defined, 421
disadvantage of applying to sprint execution, 351–352
error of overlaying Scrum on, 34
Scrum compared with, 5
types of plan-driven approaches, 29
WaterScrum, 34, 422
Weighted shortest job first (WSJF)
defined, 422
scheduling strategies and, 271
Won't-have features
defined, 422
release flow management and, 110–111
Work in process (WIP)
batch sizes in, 48–49
comparing plan-driven development with agile development, 60
considering cost of delays, 52–54
defined, 422
establishing WIP limits, 281–282
inventory management, 49–50
Kanban and, 10
overview of, 48
participants in sprint execution, 51–52
timeboxing setting limit on, 62
Workflow, organizing for fast feedback, 46–47
Workshop, for writing user stories, 95–96
WSJF (Weighted shortest job first)
defined, 422
scheduling strategies and, 271