HOWARD PODESWA

The Agile Guide to Business Analysis and Planning

From Strategic Plan to Continuous Value Delivery

Foreword by ALAIN ARSENEAULT, former IIBA Acting President & CEO

FREE SAMPLE CHAPTER

SHARE WITH OTHERS

The Agile Guide to Business Analysis and Planning

This page intentionally left blank

The Agile Guide to Business Analysis and Planning

From Strategic Plan to Continuous Value Delivery

Howard Podeswa

Addison-Wesley

Boston • Columbus • New York • San Francisco • Amsterdam • Cape Town
Dubai • London • Madrid • Milan • Munich • Paris • Montreal • Toronto • Delhi • Mexico City
São Paulo • Sydney • Hong Kong • Seoul • Singapore • Taipei • Tokyo

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.

For information about buying this title in bulk quantities, or for special sales opportunities (which may include electronic versions; custom cover designs; and content particular to your business, training goals, marketing focus, or branding interests), please contact our corporate sales department at corpsales@pearsoned.com or (800) 382-3419.

For government sales inquiries, please contact governmentsales@pearsoned.com.

For questions about sales outside the U.S., please contact intlcs@pearson.com.

Visit us on the Web: informit.com/aw

Library of Congress Control Number: 2020952174

Copyright © 2021 Pearson Education, Inc.

All rights reserved. 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. For information regarding permissions, request forms and the appropriate contacts within the Pearson Education Global Rights & Permissions Department, please visit www.pearson.com/permissions/.

Cover image: Kornn / Shutterstock
Lightbulb icon: Irina Adamovich / Shutterstock

ISBN-13: 978-0-13-419112-6
ISBN-10: 0-13-419112-9

ScoutAutomatedPrintCode

The book is dedicated to my parents: my late father, Yidel Podeswa, a professional artist whose creative talent and life force have been an everlasting inspiration to me, and my mother, Ruth Podeswa, who, through her encouragement and example, instilled in me the confidence to take on new challenges.

This page intentionally left blank

Contents

Foreword xxvii

Preface xxxi

About the Author xlvii

Chapter 1 The Art of Agile Analysis and Planning 1

1.1 Objectives 1

1.2 On Art and Agile Analysis 1

1.3 I Work for a Mainstream Company! What's This Got to Do with Me? 5

1.4 Story 1: It's Not My Problem 8

1.4.1 Conclusions 9

1.5 Story 2: The Cantankerous Customer 10

1.5.1 Conclusions 11

1.6 Chapter Summary 11

1.7 What's Next? 11

Chapter 2 Agile Analysis and Planning: The Value Proposition 13

2.1 Objectives 13

2.2 What Is Agile Analysis and Planning? 13

2.3 Who Is a Business Analyst? 14

2.4 Why Agile Analysis and Planning? 15

2.5 The Parallel Histories of Agile and Business Analysis 16

2.5.1 A Brief History of Business Analysis 17

2.5.2 A Brief History of Agile Development 17

2.6 Two Diagnoses for the Same Problem 18

2.7 The Business Analysis Diagnosis 19

2.8 The Business Analysis Track Record 19

2.9 The Agile Diagnosis 22

2.10 The Agile Track Record 22

2.11 Why Agile Teams Should Include an Effective BA Competency 24

2.12 Chapter Summary 25

2.13 What's Next? 25

Chapter 3 Fundamentals of Agile Analysis and Planning 27
3.1 Objectives 27
3.2 What the Agile Manifesto Means for Business Analysis 28
3.2.1 Agile Manifesto 28
3.2.2 The Impact of the First Value on Analysis 28
3.2.3 The Impact of the Second Value on Analysis 28
3.2.4 The Impact of the Third Value on Analysis 29
3.2.5 The Impact of the Fourth Value on Analysis 29
3.3 What the Twelve Principles Mean for Business Analysis 29
3.4 Practices, Standards, and Frameworks 31
3.4.1 Business Analysis Standards 31
3.4.2 Requirements-Related Terminology 32
3.4.3 Agile Planning 42
3.4.4 Agile Frameworks 43
3.5 Overview of Agile Roles and the Business Analyst 58
3.5.1 The Product Owner's BA Responsibilities 59
3.5.2 The Agile Team Analyst 60
3.5.3 The ScrumMaster's BA Responsibilities 60
3.5.4 Proxy User 60
3.5.5 BA Responsibilities of the Product Champion (Director) 61
3.5.6 Coach 61
3.5.7 When Are Dedicated Business Analysts Advised? 61
3.5.8 Business Analysts Provide Requirements Leadership 62
3.5.9 The Distinction between Business Analysts and Business Systems Analysts 63
3.6 Soft Skills of the Agile Business Analyst 63
3.6.1 Making the Unconscious Conscious 63
3.6.2 Curiosity 64
3.6.3 Agent of Change. 64
3.6.4 Political Intelligence 64
3.6.5 Works Well with Difficult People 64
3.6.6 Negotiation Skills 64
3.6.7 Facilitation 64
3.6.8 Adaptability 65
3.6.9 Not Afraid to Ask Questions 65
3.6.10 Sense of Humor 65
3.7 13 Key Practices of Agile Analysis and How They Differ from Waterfall . . 65
3.7.1 A Competency, Not a Role 65

3.7.2 A Facilitator, Not a Messenger 65
3.7.3 Changes to Requirements Are Welcomed 66
3.7.4 Collaboration with Developers vs. Contractual Relationship 66
3.7.5 Just-In-Time Requirements Analysis 66
3.7.6 Conversation versus Documentation 66
3.7.7 Specification by Example: Acceptance Test–Driven Development 66
3.7.8 Small Requirements Units 66
3.7.9 Vertical Slices of Functionality 67
3.7.10 Lightweight Tools 67
3.7.11 Business Analyst and Business Stakeholder Engagement across the Complete Development Lifecycle 67
3.7.12 Mix of BA Classic and Agile BA Tools 67
3.7.13 Meet Them Where They Are 67
3.8 Agile Business Analysis Rules of Thumb 68
3.9 Chapter Summary 68
3.10 What's Next? 68

Chapter 4 Analysis and Planning Activities across the Agile Development Lifecycle 69

4.1 Objectives 69
4.2 Overview of the Agile Analysis and Planning Map 72
4.3 The Zones 72
4.4 The Lanes 73
4.5 A Story in Three Acts 74
4.6 Act 1: The Short Lane 74
4.6.1 Initial Preparation and Planning 75
4.6.2 Seeding the Backlog 75
4.6.3 Daily Activities 77
4.6.4 Feature Closeout: Prepare for GA 78
4.6.5 Quarterly Inception, Iteration Inception 78
4.6.6 Iteration Closeout 78
4.6.7 Quarterly Closeout 78
4.7 Act 2: The Long Lane 79
4.8 Act 3: The Grand Lane 79
4.8.1 Scale the Organization 79
4.8.2 Scaled Quarterly Planning 80
4.8.3 Scaled Iteration Planning 80
4.8.4 Daily Planning and Analysis 80

4.8.5 Iteration Closeout 80
4.8.6 Quarterly Closeout 81
4.9 Chapter Summary 81
4.10 What's Next? 81

Chapter 5 Preparing the Organization 83
5.1 Objectives 83
5.2 This Chapter on the Map 86
5.3 What Is Initiation and Planning? 86
5.4 How Long Should You Spend Up Front on Initiation and Planning? 87
5.4.1 The Greater the Anticipated Risks, the Greater the Need for Upfront Planning 87
5.4.2 What's Past Is Prologue 87
5.5 The Purpose Alignment Model 88
5.5.1 Differentiating Quadrant (Top Right) 89
5.5.2 Parity Quadrant (Bottom Right) 89
5.5.3 Partner Quadrant (Top Left) 90
5.5.4 Who Cares? Quadrant (Bottom Left) 90
5.6 Preparing the Infrastructure 90
5.6.1 Transitioning from Manual to Automated Testing 91
5.6.2 Timing the Automation of the Build and Distribution Processes 93
5.7 Organizing Development Teams 93
5.7.1 Guidelines for Forming Agile Teams 94
5.7.2 Organize around Value 95
5.7.3 Feature Teams versus Generic Teams 96
5.7.4 The Extended Team 97
5.7.5 Why Organizing by Competency Is Bad for the Business 98
5.8 Managing Stakeholder Expectations about Agile Development 99
5.8.1 The Negative Expectation That Requirements Delayed Are Requirements Denied 99
5.8.2 Productivity Expectations 100
5.9 Preparing the Customer–Developer Relationship 101
5.9.1 Customer's Bill of Rights and Responsibilities 101
5.9.2 Developers' Bill of Rights and Responsibilities 102
5.10 Agile Financial Planning 102
5.10.1 Measuring Success 103
5.10.2 Discovery-Driven Financial Planning 103
5.11 Preparing the Marketing and Distribution Teams 103
5.12 Preparing Channels and Supply Chains 104

5.13 Preparing Governance and Compliance 104
5.13.1 Challenge Compliance Assumptions 105
5.13.2 Do Compliance *After* Process Design 105
5.13.3 Focus on Goals, Not Means 105
5.13.4 One-Time Experiments 106
5.14 Preparing for Increased Demand on Resources 106
5.15 Preparing an Enterprise for Agile Development 107
5.15.1 Agile Fluency Model 107
5.15.2 Transitioning the Team 108
5.15.3 Transition Activities at the Enterprise Level 109
5.15.4 Transition Timeline 111
5.15.5 Communications Plan 111
5.15.6 Agile Enterprise Transition Team 112
5.16 Determine Organizational Readiness 112
5.16.1 Organizational Readiness Checklist 113
5.17 Chapter Summary 113
5.18 What's Next? 114

Chapter 6 Preparing the Process 115

6.1 Objectives 115
6.2 This Chapter on the Map 115
6.3 Process Preparation 118
6.4 Tailoring the Agile Practice to the Context 118
6.4.1 Costs of Agile Development 118
6.4.2 Benefits of Agile Development 119
6.4.3 Finding the Best Trade-Off of Costs and Benefits 119
6.4.4 Determining the Framework 121
6.5 Tuning the Process 122
6.5.1 Business Analysis Information Artifacts and Events 122
6.5.2 Checklist of Agile BA Information Artifacts 122
6.5.3 Defining Requirements Types 123
6.5.4 Tuning the Backlog 124
6.5.5 Determining Requirements Granularity Levels 127
6.5.6 Tracing Requirements and Other Configuration Items 129
6.5.7 Setting Process Parameters 134
6.6 Optimizing the Process Using Value Stream Mapping 145
6.7 Determining Process Readiness 145
6.8 Chapter Summary 146
6.9 What's Next? 146

Chapter 7 Visioning . . . 147
7.1 Objectives . . . 150
7.2 This Chapter on the Map . . . 150
7.3 Overview of Product Visioning and Epic Preparation . . . 150
7.3.1 An Example of Product Visioning and Why It's Important . . . 151
7.3.2 Visioning Checklist . . . 152
7.3.3 Initial Stakeholder Identification . . . 152
7.3.4 Facilitation Tips . . . 152
7.4 Root-Cause Analysis . . . 152
7.4.1 Five Whys . . . 153
7.4.2 Cause–Effect Diagrams . . . 157
7.4.3 Cause–Effect Trees . . . 161
7.5 Specifying a Product or Epic . . . 166
7.6 The Problem or Opportunity Statement . . . 167
7.7 The Product Portrait . . . 169
7.7.1 The Product Portrait Template . . . 170
7.8 Crafting the Product and Epic Vision Statements . . . 172
7.8.1 The Product Vision Statement . . . 172
7.8.2 The Epic Vision Statement . . . 172
7.8.3 Properties of Well-Crafted Product and Epic Vision Statements . . . 172
7.8.4 Vision versus Mission Statements . . . 173
7.9 Stakeholder Analysis and Engagement . . . 175
7.9.1 Identify and Analyze Stakeholders . . . 176
7.9.2 Plan Stakeholder Collaboration . . . 176
7.9.3 Plan Stakeholder Communication . . . 178
7.9.4 Facilitate and Conduct Ongoing Engagement and Analysis . . . 179
7.10 Analyzing Goals and Objectives . . . 182
7.10.1 Use Circumstance-Based Market Segmentation as a Basis for Goals and Objectives . . . 182
7.10.2 Representing Goals and Objectives within the Story Paradigm . . . 183
7.11 Analyze Leap of Faith Hypotheses . . . 185
7.11.1 What Is a Lean Startup? . . . 185
7.11.2 What Are Leap of Faith Hypotheses? . . . 185
7.11.3 Value Hypotheses . . . 186
7.11.4 Growth Hypotheses . . . 186
7.11.5 Specifying Metrics . . . 187
7.11.6 Hypotheses in Discovery-Driven Planning . . . 189
7.11.7 Assumption Checklist . . . 189
7.11.8 Using a Milestone Planning Chart to Plan Assumption Testing . . . 190

7.12 Chapter Summary 192
7.13 What's Next? 192

Chapter 8 Seeding the Backlog—Discovering and Grading Features 193

8.1 Objectives 193
8.2 This Chapter on the Map 196
8.3 Overview: Seeding the Backlog 196
8.3.1 Definitions: Epics and Stories 196
8.3.2 How Many Features Should You Seed Up Front? 196
8.3.3 Whom to Invite to Backlog Seeding 197
8.4 Circumstance-Based Market Segmentation for Feature Discovery 198
8.5 Other Ways to Discover Initial Features 198
8.6 Feature Independence 199
8.7 Using the Role-Feature-Reason Template to Represent Epics and Features 199
8.8 Specifying Emergent Features 200
8.9 Physical Representation of Features 200
8.10 Feature Attributes 201
8.11 Determining Customer and User Value with Kano Analysis 202
8.11.1 Select the Target Features 202
8.11.2 Select the Customers 203
8.11.3 Create the Questions 203
8.11.4 Create Prototypes 204
8.11.5 Test the Questionnaire Internally 204
8.11.6 Conduct the Survey 204
8.11.7 Grade the Features 204
8.11.8 Interpreting the Kano Grades 206
8.11.9 Satisfaction versus Fulfillment Graph 207
8.11.10 The Natural Decay of Delight (and Its Opposite) 208
8.11.11 Continuous Analysis 208
8.12 Sequencing Epics and Features in the Backlog 212
8.12.1 Determining Cost of Delay 212
8.12.2 Determining WSJF 213
8.12.3 Prioritization Tips 213
8.13 Writing Feature Acceptance Criteria 215
8.14 Analyzing Nonfunctional Requirements and Constraints 216
8.14.1 Do NFRs Go in the Backlog? 217
8.14.2 NFRs and Constraints Checklist 217

8.15 Chapter Summary 220
8.16 What's Next? 220

Chapter 9 Long-Term Agile Planning 221
9.1 Objectives 221
9.2 This Chapter on the Map 224
9.3 Overview of Long-Term Planning, Epic Planning, and MVP 224
9.4 The Full-Potential Plan 225
9.4.1 Phase 1: Define Bold Targets 225
9.4.2 Phase 2: Create a Detailed Plan 226
9.4.3 Phase 3: Deliver Quick Wins 226
9.4.4 The Business Analyst's Contribution to a Successful Full-Potential Plan 227
9.5 Using MVPs to Validate the Assumptions behind the Plan 228
9.5.1 Overview 228
9.5.2 What Is an MVP? 229
9.5.3 The MVP Process 229
9.6 Capabilities for Effective MVP Implementation 231
9.6.1 Technical Capabilities 231
9.6.2 Deployment and Delivery Approach 232
9.6.3 Deployment Options and Potential Issues 234
9.7 Overview of the Product Roadmap 240
9.8 Planning the Interim Periods 241
9.8.1 Specify the Interim Timeline 242
9.8.2 Craft Interim Goals and Objectives 242
9.8.3 Specify Assumptions and Metrics 243
9.8.4 Specify Events and Milestones 243
9.8.5 Specify Features 243
9.9 Using the Product Roadmap for Shorter Planning Horizons 248
9.10 Chapter Summary 248
9.11 What's Next? 249

Chapter 10 Quarterly and Feature Preparation 251
10.1 Objectives 251
10.2 This Chapter on the Map 254
10.3 Overview of Features 254
10.3.1 Examples of Feature-Length Change Initiatives 254
10.4 Benefits of Feature Preparation 256
10.5 Feature Preparation Activities 256

10.6 Timing of Feature Preparation 257
10.7 Assessing Readiness 258
10.7.1 Using the Feature Definition of Ready (Feature DoR) 258
10.8 Accounting for Preparation Work: Tasks and Spikes 258
10.9 Specifying Features and Their Acceptance Criteria 259
10.9.1 Specifying Epic Acceptance Criteria 260
10.9.2 Specifying Feature Acceptance Criteria 261
10.9.3 The Analyst Contribution 261
10.9.4 Analyze AC During Triad Meetings 262
10.9.5 Specifying AC in the BDD Gherkin Syntax 262
10.9.6 Specifying UAT for End-to-End Workflows 263
10.10 Context Analysis 263
10.11 Stakeholder Analysis 264
10.12 Persona Analysis 264
10.12.1 History of Personas 265
10.12.2 Persona Examples 266
10.12.3 Creating Personas 267
10.12.4 Documenting Personas 268
10.12.5 Working with Personas 269
10.13 Overview of Journey, Process, and Value Stream Maps 272
10.14 Journey Mapping 272
10.14.1 Overview of the Customer Journey Map 273
10.14.2 Customer Journey Map: Mortgage Example 273
10.14.3 Components of a Journey Map 274
10.14.4 Using the Journey Map 278
10.14.5 More on Journey Maps 283
10.15 Value Stream Mapping 283
10.15.1 Developing a Value Stream Map 284
10.16 Business Process Modeling 285
10.16.1 Bring Process Participants Together 285
10.16.2 What Situations Call for Process Modeling? 286
10.16.3 Screenshots Do Not a Process Model Make 286
10.16.4 Do Just Enough Analysis for Your Purposes 287
10.16.5 Models with and without Swimlanes 287
10.16.6 BPMN 287
10.17 Use-Case Modeling 298
10.17.1 Use-Case Modeling Example: Claims 298
10.17.2 Use-Case Modeling Elements 299

10.18 User-Role Modeling Workshops 300
10.18.1 Agenda 300
10.19 Review the Architecture 307
10.19.1 Context Diagram 307
10.19.2 UML Communication Diagram 308
10.19.3 Data Flow Diagrams 308
10.19.4 Architecture (Block) Diagrams 310
10.20 Chapter Summary 312
10.21 What's Next? 313

Chapter 11 Quarterly and Feature Planning 315
11.1 Objectives 315
11.2 This Chapter on the Map 318
11.3 Overview of Quarterly Planning 318
11.4 Overview of Flow-Based Feature Planning 318
11.5 When Is Planning at This Level Advised and Not Advised? 319
11.6 When to Use Quarterly Planning versus Flow-Based Feature Planning 319
11.7 How to Conduct Quarterly Planning with Agility 320
11.7.1 Create a Culture of Change 321
11.7.2 Use Data-Informed Decisioning 321
11.7.3 Specify Outcomes, Not Outputs 321
11.7.4 View the Plan as a Hypothesis, Not a Contract 321
11.8 XP's Planning Game Guidelines 322
11.8.1 Overview of the Planning Game 322
11.8.2 Overview of Roles 323
11.8.3 Overview of Planning Principles 323
11.9 Quarterly Planning: Timing Considerations 325
11.10 Preparing for the Planning Event 325
11.10.1 Verify Entry Conditions 325
11.10.2 Prepare Invitation List 326
11.10.3 Determine the Planning Horizon 326
11.10.4 Prepare Inputs and Deliverables 326
11.10.5 Refine Features and Acceptance Criteria Incrementally 327
11.11 Planning Topics (Agenda) 328
11.11.1 Overview 329
11.11.2 Exploration 332
11.11.3 Commitment 341
11.11.4 Planning Retrospective 348

11.12 Reviewing the Quarterly Plan, Once the Quarter Is Underway 351
11.12.1 Start of an Iteration 351
11.12.2 Velocity Corrections 351
11.12.3 New Features 352
11.12.4 The Plan Becomes Obsolete 352
11.13 Chapter Summary 352
11.14 What's Next? 352

Chapter 12 MVPs and Story Maps 353

12.1 Objectives 353
12.2 This Chapter on the Map 356
12.3 MVPs and Story Mapping: How the Tools Complement Each Other . . . 356
12.4 MVP Planning 356
12.4.1 What Is an MVP? 357
12.4.2 MVP Case Study: Trint 357
12.4.3 Venues for MVP Experiments 358
12.4.4 MVP Types 359
12.4.5 MVP's Iterative Process 363
12.4.6 The Pivot 363
12.4.7 Incrementally Scaling the MVP 364
12.4.8 Using MVPs to Establish the MMP 365
12.5 Story Mapping 366
12.5.1 Jeff Patton's Story Map 366
12.5.2 Benefits of a Story Map 367
12.5.3 The Anatomy of a Story Map 368
12.5.4 Dependency Relationships on the Map 370
12.5.5 Story Map Example 370
12.5.6 Tips for Writing Stories on the Map 372
12.5.7 Constructing the Backbone 372
12.5.8 Constructing the Ribs 379
12.5.9 Other Forms of Story Maps 386
12.6 Chapter Summary 388
12.7 What's Next? 388

Chapter 13 Story Preparation 391

13.1 Objectives 391
13.2 This Chapter on the Map 394
13.3 Overview of Story Preparation 394

13.4 Story Fundamentals 394
13.4.1 What Is a Story? 395
13.4.2 Alternative Terminology 395
13.4.3 Size Taxonomy 395
13.4.4 What's in a Name? 396
13.4.5 User Story Examples 397
13.5 The Three Cs of Stories 397
13.5.1 Card 397
13.5.2 Conversation 398
13.5.3 Confirmation 398
13.6 Who Is Responsible for User Stories? 398
13.6.1 Who Writes Stories? 398
13.6.2 The Analyst Value Added 399
13.6.3 The Triad 400
13.7 Physical versus Electronic Stories 403
13.8 Specifying Values for Story Attributes 404
13.9 Writing the Story Description 404
13.9.1 When to Use a Story Template (and When Not To) 404
13.9.2 Role-Feature-Reason (Connextra) Template 405
13.10 Specifying Story Acceptance Criteria 407
13.10.1 Examples of Story Acceptance Criteria 408
13.10.2 Who Writes Acceptance Criteria? 408
13.10.3 When to Create and Update Acceptance Criteria 409
13.10.4 Specification by Example 409
13.10.5 How Extensive Should the Acceptance Criteria Be? 411
13.10.6 How Many Acceptance Criteria per Story? 411
13.10.7 Characteristics of Well-Formed Acceptance Criteria 411
13.10.8 Emergent Acceptance Criteria 413
13.10.9 Using the Behavior-Driven Development (BDD) Gherkin Format . . . 413
13.10.10 Who Tests Acceptance Criteria and When? 414
13.11 Stories That Aren't User Stories 414
13.11.1 What Is a Spike or Enabler Story? 415
13.11.2 Functional Spike 416
13.11.3 Technical Spike 418
13.11.4 Compliance Story 420
13.11.5 Bug-Repair Stories 420
13.12 Guidelines for Writing High-Quality Stories 420
13.12.1 INVEST 421
13.12.2 INVEST IN CRUD 421

13.13 Patterns for Splitting Stories 422
13.13.1 How to Use the Patterns 422
13.13.2 Tie-Breakers 422
13.13.3 The Patterns 423
13.14 Analyzing Business Rules and AC with Decision Tables 433
13.14.1 Behavioral Business Rules 434
13.14.2 Decision Table Example 435
13.14.3 Benefits of a Decision Table 436
13.14.4 How to Elicit Rules Using the Table 436
13.15 Chapter Summary 440
13.16 What's Next? 440

Chapter 14 Iteration and Story Planning 441
14.1 Objectives 441
14.2 This Chapter on the Map 444
14.3 Overview of Iteration and Story Planning 444
14.4 Attendees 445
14.5 Duration 445
14.6 Inputs for Iteration Planning 445
14.7 Deliverables of Iteration Planning 446
14.7.1 The Iteration Goal and Iteration Backlog 446
14.7.2 The Developer Task Board 446
14.7.3 The Increment 446
14.8 Planning Rules 447
14.9 Part 1: Forecast What Will Be Accomplished 447
14.9.1 Update 448
14.9.2 Forecast Capacity 448
14.9.3 Review Ready and Done Definitions 449
14.9.4 Craft the Iteration Goal 449
14.9.5 Discuss Stories 450
14.9.6 Forecast the Stories That Will Be Delivered 450
14.10 Part 2: Plan the Implementation 451
14.10.1 Should You Invite the PO to Part 2? 451
14.10.2 Overview of Part 2 451
14.10.3 Part 2 Steps 452
14.11 Setting Up the Kanban Board 458
14.11.1 Columns on the Kanban Board 459
14.12 Scaling Iteration Planning 462

14.13 Feature Preview Meeting 462
14.13.1 Feature Preview Objectives 462
14.13.2 Timing Considerations 462
14.13.3 Why *Two* Iterations Ahead? 463
14.14 Chapter Summary 463
14.15 What's Next? 463

Chapter 15 Rolling Analysis and Preparation—Day-to-Day Activities 465
15.1 Objectives 465
15.2 This Chapter on the Map 468
15.3 Overview of Rolling Analysis 468
15.3.1 A Day in the Life of the Agile Analyst 468
15.3.2 Overview of Analysis Tasks 470
15.4 Updating Task Progress 470
15.5 Triad Guideline 470
15.6 Actions That May Be Taken against a Developer Task 471
15.7 Monitoring Progress 471
15.7.1 The Daily Standup 471
15.7.2 Follow-Up Meeting 474
15.7.3 Updating the Developer Task Board 475
15.7.4 Updating the Kanban Board 476
15.7.5 Monitoring Progress with a Daily Burndown Chart 479
15.7.6 Burnup Charts 486
15.7.7 What Should You Use: Burndown or Burnup Charts? 486
15.7.8 Cumulative Flow Diagrams 487
15.8 Story Testing and Inspection (Analyze-Code-Build-Test) 490
15.8.1 Overview of the Analyze-Code-Build-Test Cycle 490
15.8.2 Validating Value 491
15.9 Managing Scope Change during the Iteration 495
15.9.1 When Progress Is Lower or Higher than Expected 495
15.9.2 When the PO Wants to Add Stories After the Iteration Begins 495
15.10 Updating Business Analysis Documentation 496
15.10.1 Persisting Stories 496
15.10.2 Feature Documentation: Organize by Features, Not Stories 497
15.10.3 Updating the Use-Case Model 497
15.10.4 Other Analysis Documentation 506
15.10.5 Tracing Analysis Artifacts 506
15.11 Ongoing Analysis of Upcoming Epics, Features, and Stories 509
15.11.1 How Long Should You Spend on Preparation? 509

15.11.2 Overview of Rolling Preparatory Analysis 509
15.11.3 Feature Preparation 510
15.11.4 Story Preparation 510
15.11.5 Pruning and Ordering 512
15.12 Accounting for Progress at the End of the Iteration 513
15.12.1 Accounting for Stories That Are Not Done 513
15.12.2 Accounting for Progress When an Iteration Is Canceled 513
15.13 The Iteration Review 514
15.13.1 Inputs and Deliverable 514
15.13.2 Topics/Agenda 515
15.13.3 Iteration Review—Artifacts for Forecasting and Tracking Progress 516
15.14 The Iteration Retrospective 517
15.14.1 Timing Considerations 517
15.14.2 Attendees 518
15.14.3 Inputs and Deliverables 518
15.14.4 Topics 518
15.14.5 Iteration Retrospective Games 520
15.15 Chapter Summary 524
15.16 What's Next? 525

Chapter 16 Releasing the Product 527

16.1 Objectives 527
16.2 This Chapter on the Map 530
16.3 Getting Stories to Done 530
16.4 Releasing to the Market: Timing Considerations 530
16.4.1 Should You Reserve a Hardening Iteration for Prerelease Activities? 531
16.5 Staging the Release 532
16.5.1 Pre-Alpha 533
16.5.2 Alpha Testing 533
16.5.3 Beta Testing 533
16.5.4 General Availability 535
16.6 Quarterly (Release) Retrospective 539
16.6.1 Facilitation Guidelines 539
16.6.2 Preparing the Timeline 542
16.6.3 Walkthrough of a Quarterly Retrospective 543
16.7 Pivot-or-Persevere Meeting 544
16.7.1 Data-Informed—Not Data-Driven 545

16.7.2 Timing Considerations 545
16.7.3 Attendees 545
16.7.4 Walkthrough of a Pivot-or-Persevere Meeting 545
16.8 Chapter Summary 547
16.9 What's Next? 548

Chapter 17 Scaling Agility 549
17.1 Objectives 552
17.2 This Chapter on the Map 552
17.3 Why Do We Need a *Scaled* Agile Approach? 552
17.3.1 Why Scaled Agile Teams Are Interdependent 553
17.3.2 Product Complexity 554
17.3.3 Shared Components 554
17.4 Planning: Choosing an Approach That Supports Inter-team Collaboration 554
17.4.1 Review of the Two Approaches 555
17.4.2 Which Approach Should You Use at the Frontend? 555
17.4.3 Overview of the Analyst Contribution to Scaled Planning and Implementation 557
17.5 Continuous Delivery: Delivering Software Continuously, Safely, and Sustainably at Scale 558
17.5.1 Overview of Automation in the Test-Build-Deploy Steps 558
17.5.2 DevOps, CI/CD 559
17.5.3 Test-Driven Development 562
17.5.4 ATDD and BDD 563
17.6 Scaled Agile Culture: Creating a Culture That Supports Innovation at Scale 564
17.6.1 Effective Agile Leadership 564
17.6.2 Prioritize Quality 566
17.6.3 Remove Silos; Foster Collaboration 566
17.6.4 Foster a Culture of Rapid Learning 566
17.7 Scaling the Backlog 566
17.7.1 Overview 567
17.7.2 One Top-Level Product 568
17.7.3 Multiple Subproducts 568
17.7.4 One Product-Level PO 568
17.7.5 One Backlog at the Whole-Product Level 568
17.7.6 Multiple Team Backlogs 569

17.7.7 Feature Teams 569
17.7.8 Component Teams 569
17.7.9 One Definition of Done (DoD) 569
17.8 Scaling the Agile Organization 570
17.8.1 Scaling by Subproduct and Product Area: MyChatBot Case Study 570
17.8.2 Scaling the PO Role 571
17.8.3 Portfolio and Program Structure 572
17.8.4 Forming the Feature Teams 575
17.8.5 The Extended Team 576
17.8.6 Component Teams 577
17.8.7 Competency Groups 577
17.8.8 The Product Owner Council 579
17.8.9 User Task Force 581
17.8.10 Release Management Team 581
17.9 Scaling the Agile Process 581
17.9.1 Scaled Agile Frameworks 582
17.9.2 Overview of Scaled Activities and Events 583
17.9.3 Initial Preparation 585
17.9.4 Scaled Quarterly and Feature Planning 586
17.9.5 Scaled Iteration (Sprint) Planning Meetings 595
17.9.6 Big Room Iteration Planning 598
17.9.7 Feature Preview 599
17.9.8 Integration Meetings 599
17.9.9 Daily Standup 600
17.9.10 Scrum of Scrums (SoS) 600
17.9.11 Product Owner Council Meeting 601
17.9.12 Scaled (Quarterly) Feature Preparation (Multiple Teams) 602
17.9.13 Team-Level Story Preparation 605
17.9.14 User Task Force Meetings 606
17.9.15 Scaled Iteration Review or Feature Review 606
17.9.16 Scaled Iteration Retrospective 607
17.9.17 Scaled Quarterly/Feature Retrospective 610
17.9.18 Open Space 611
17.9.19 Triad 614
17.10 Agile Requirements Management Software Tools 615
17.10.1 Requirements Management Tool Checklist 615
17.10.2 Overview of Agile Requirements Management Tools 615

17.11 Lightweight Tools for Supporting Inter-team Collaboration 615
17.11.1 Team Structure 616
17.11.2 Visualization 616
17.11.3 "Just Talk" 616
17.11.4 Scouts 616
17.11.5 Roamers 616
17.11.6 Shared Team Members 617
17.11.7 Implement Work Items Sequentially, Not Concurrently 617
17.11.8 Enforce a Definition of Ready 617
17.12 Potential Issues and Challenges in Scaling Agility 617
17.12.1 Guidelines for Non-colocated Teams 617
17.12.2 Guidelines for Working with Waterfall Teams 619
17.12.3 Inability to Deploy Frequently and Reliably 620
17.12.4 Recurring Integration Errors and Dependency Issues 620
17.12.5 Conflicting Priorities 620
17.12.6 Insufficient Business Resources 621
17.13 Chapter Summary 622
17.14 What's Next? 622

Chapter 18 Achieving Enterprise Agility 623
18.1 Objectives 623
18.2 This Chapter on the Map 626
18.3 Overview of Enterprise Agility 627
18.3.1 Definition of an Agile Enterprise 627
18.3.2 Why It Matters 627
18.3.3 The Business Analysis Contribution 628
18.3.4 Drivers for Enterprise Agility 628
18.3.5 Agility in Heavily Regulated Sectors 629
18.4 Foundational Practices 629
18.4.1 Lean Startup/MVP 630
18.4.2 Full-Potential Plan 630
18.4.3 Circumstance-Based Market Segmentation 630
18.4.4 Disruptive Innovation 631
18.5 Overview of the Agile Process for Developing Innovative Products 631
18.6 Agile Corporate Culture 632
18.6.1 Definition of Corporate Culture 633
18.6.2 Definition of Agile Corporate Culture 633
18.7 Overview of Principles and Practices for an Agile Corporate Culture 634

18.8 Three Principles for Applying Agile Practices 635
18.8.1 Tailor the Approach to the Circumstance 635
18.8.2 Protect Islands of Innovation 644
18.8.3 Invest Aggressively in Enterprise Agility 648
18.9 The Thirteen Practices for an Agile Corporate Culture 650
18.9.1 Iterative Experimentation (Fail Fast) 650
18.9.2 Embrace Change 652
18.9.3 Acceleration 653
18.9.4 Empathy 655
18.9.5 Responsible Procrastination (Last Responsible Moment) 659
18.9.6 Distributed Authority 659
18.9.7 Let Those Who Do the Work Estimate the Effort 663
18.9.8 Collaboration 663
18.9.9 Commit to Outcomes, Not Outputs 666
18.9.10 Transparency 666
18.9.11 Bust Silos 667
18.9.12 Data-Informed Innovation 672
18.9.13 Monitor Adjacent and Low-End Markets 673
18.10 Agile Financial Planning 675
18.10.1 Real Options 675
18.10.2 Discovery-Driven Planning 675
18.11 Chapter Summary 676

Appendix A Additional Resources and Checklists 677
A.1 Mapping of Book Chapters to IIBA and PMI Guides 677
A.2 Rules of Thumb in Agile Analysis and Planning 682
A.3 Facilitation Tips 684
A.4 Visioning Checklist 686
A.5 Stakeholder Checklist 687
A.6 NFRs and Constraints Checklist 689
A.7 Readiness Checklist for Quarterly Planning 690
A.7.1 Analysis Readiness 690
A.7.2 Logistics Readiness 690
A.8 Checklist of Invitees for Quarterly Planning 692
A.9 Checklist of Quarterly and Feature Planning Inputs 693
A.10 Checklist of Quarterly and Feature Planning Deliverables 694
A.11 Checklist of Quarterly (Release) Retrospective Questions 695
A.11.1 DevOps and Supporting Practices Perspective 695
A.11.2 Technology Perspective 695

A.11.3 Productivity Perspective 695
A.11.4 Quality Assurance (Testing) Perspective 696
A.11.5 Program/Portfolio Perspective 697
A.11.6 Marketplace Perspective 697
A.12 Checklist of Invitees for Scaled Quarterly and Feature Planning 698
A.13 Overview of Agile Requirements Management Tools 699
A.13.1 JIRA 699
A.13.2 Blueprint 699
A.13.3 JAMA Software 699
A.13.4 Other Requirements Management and Collaboration Tools 699

Appendix B Discovery-Driven Planning Case Study: BestBots 701

B.1 Background: BestBots Case Study 701
B.2 Initial Market Analysis 702
B.2.1 Market Estimates (Past and Future) 702
B.2.2 Compound Annual Growth Rate 702
B.2.3 Spreadsheet Fix 703
B.3 Determine Constraints (Required Outcomes) 703
B.3.1 Constraints 704
B.4 Create Draft of Reverse Income Statement 705
B.4.1 Conclusions from the Reverse Income Statement Draft 706
B.5 Create Pro Forma Operations Specifications 706
B.6 Create Assumptions Checklist 708
B.7 Revise Reverse Income Statement 709
B.8 Create Milestone Planning Chart 710

Bibliography 713

Index 715

Foreword

There are three things Howard and I have in common: our passion for business analysis, our enthusiasm for painting, and our love of good food and conversation.

Several years ago, I worked at one of the largest banks in Canada as the center of excellence (CoE) lead for requirement management/business analysis. I held the responsibility for advancing the requirement management capabilities for the organization's IT & Operations unit, including the training curriculum for business analysis. It is there that I met Howard as we collaborated, mapping the bank's business analysis competencies in the development of a new training curriculum for the bank's business analysts. I was immediately impressed by Howard's ability to understand what I was trying to achieve. He understood well the role of the business analyst and the knowledge and experience business analysts must have to be efficient in their position. His recommendations to augment the curriculum's quality were to the point, and his willingness to collaborate and adjust his course offerings to fit my needs was essential to me.

We subsequently met several times, through formal business meetings, discussing how his courses were performing for us. These were also excellent opportunities to discuss how we could collaborate to advance the training curriculum further. I eventually moved to a different position. Howard and I stayed in contact. We met regularly on a casual basis, catching up, and often ran into each other at industry conferences where Howard presented.

We collaborated through the International Institute of Business Analysis (IIBA). I served in various capacities for fifteen years, initially as a volunteer in multiple roles, including chair of the board of directors. I also led the association as interim president and CEO in 2013–2014. I covered various roles and functions afterward, including director of business and corporate development, where I established multiple strategic alliances with other professional associations.

I established a formal relationship with the Agile Alliance. I negotiated with them a collaboration to develop the second edition of the *Agile Extension to the BABOK Guide, v3.0*, which successfully launched in August 2017. It is an excellent publication. The book tells you what you need to know about agile analysis—it lays out the land, if you wish; it describes the concepts and techniques practitioners should know. Howard has mapped them all out for you in this publication plus many others. However, in my opinion, the real value this book provides, and the reason I don't hesitate telling you to invest in it, is the way Howard interlaces, using a running case study, dozens of scenario-based examples, tools, and techniques. Furthermore, Howard describes them all across the product development lifecycle and how they apply to the most common agile industry frameworks.

Over the last twenty-plus years in business analysis covering various functions, I saw firsthand how difficult it has been for many organizations to transition from a waterfall or some form of iterative development approach to agile. To my chagrin, I saw many organizations debating whether the role of the business analyst still had a place in an agile environment. I witnessed how challenging it has been for many seasoned business analysts to upskill their agile competencies to continue to bring value to their organization. There has been much progress since then, and business analysts have emerged as essential contributors to agile initiatives. Today, organizations with a high level of maturity in product development understand the critical importance that business analysts bring to their agile practices. But for many other organizations, there are still significant challenges as organizations try to fit bits and parcels of two or three agile frameworks to meet their internal processes and ways to manage projects. And this is where the real value of this book comes in. Howard has laid out more than 175 tools and techniques, examples, and guidelines that product owners and business analysis practitioners can readily apply.

Howard's involvement with IIBA is also important to note, as an overall supporter of the association, as a contributor in the review of *BABOK v3: A Guide to the Business Analysis Body of Knowledge*, and sometimes as a gadfly, provoking the organization toward continual improvement. During my tenure at IIBA, I had the privilege to cochair the IIBA official annual Building Better Capability global conference over the course of five years. I had the opportunity to see Howard present in person. The subjects of his presentations were always pertinent, and the delivery always professional, valuable, and enthusiastically well received.

Throughout the years, the relationship I had with Howard evolved into a friendship, as we shared similar passions and interests. I have the highest respect and admiration for Howard. As this book demonstrates, he is the consummate business analysis professional and a recognized leader in the field. He is also an accomplished artist, having exhibited his work in several galleries throughout the world. And he certainly knows how to pick the perfect restaurant for a great meal and conversation.

Howard is a pioneer in the field of business analysis. His first book, *UML for the IT Business Analyst: A Practical Guide to Object-Oriented Requirements Gathering*, was published in 2005 as business analysis fully emerged as a profession. His second book, *The Business Analyst's Handbook*, published in 2009, has become a business analysis staple for both seasoned and aspiring business analysts throughout the world. *The Agile Guide to Business Analysis and Planning* represents a culmination of his vast experience in both agile and business analysis.

What is unique about this book is how Howard treats the subject. It is also how he presents himself. The book has a personal feel to it. It's rather uncommon for a business publication to include several pages dedicated to the author's particular interest, in this case, Howard's passion for painting. But by doing so, Howard connects with the reader on a more personal level, demonstrating how his artistic capabilities add to the richness of his persona and how creativity can be a catalyst for problem solving and innovation, which Howard describes across the book.

With his vast experience in the field, Howard demonstrates how business analysis and agile practitioners can apply fundamental business analysis practices and techniques across the most widely used agile frameworks—including Scrum, Kanban, SAFe, DevOps, XP, lean software development, lean startup, and continuous delivery (CD)—and across all the product development lifecycle activities.

Whether you are new to agile practices or a seasoned business analyst transitioning from traditional business analysis to agile analysis, you will learn which tools to use and when to use them. Howard provides step-by-step guidance for performing your analysis work across the entire product development lifecycle, advice and guidance you can use immediately to be more confident and productive from day one on an agile project. Product owners will gain confidence in interacting with agile teams as they carry out the high-level agile planning analysis activities. Furthermore, they will be able to leverage Howard's guidance to manage stakeholder expectations and keep them involved and engaged throughout the product development process.

I don't pretend to be an expert in agile analysis and planning. I know enough about it to understand how valuable this book is for anyone involved in an agile initiative. I have seen the challenges many practitioners are facing when embarking on a new agile initiative. This book will become a staple reference that both product owners and business analysis practitioners should have by their side.

I am grateful that Howard asked me to write this foreword and thankful for the trust he put in me in helping him wherever I can to bring this publication to fruition. I know you will enjoy reading it and get great value from it.

Happy reading.

—Alain Arseneault
Former IIBA Acting President & CEO, and
President & CEO of TheBAExecutive™

This page intentionally left blank

Preface

"The green reed which bends in the wind is stronger than the mighty oak which breaks in a storm."
—Confucius

This book aims to help enterprises become nimbler and more effective in responding to a rapidly changing environment by assisting them in establishing a reliable, agile analysis, and planning competency. **Agile analysis and planning** is defined in this book as an organizational competency concerned with the *examination of a business* or any aspect of it (including culture, organizational structure, processes, and products) to *learn what needs to change and when* in order to achieve a desired outcome, in a context that places a high premium on *adaptability*, *resilience,* and *continuous innovation and value delivery*. Key activities within the competency include analyzing who the product is for (the stakeholders), defining their requirements, determining when the capabilities will be delivered, and estimating costs and resources.

Why I Wrote This Book

In my many years of consulting with IT organizations, I've seen practitioners of agile analysis and planning struggle to find a hands-on book that provides guidance they could readily use on the job. Current books on the subject lay down the framework for the competency. International Institute of Business Analysis (IIBA)'s *Agile Extension to the BABOK Guide*, published in association with the Agile Alliance, provides a foundation that describes, in broad terms, how to apply techniques and principles at different planning horizons. Project Management Institute (PMI)'s *Agile Practice Guide* provides a valuable overview, from the perspective of project leaders and project teams. There are also essential books that provide detailed guidance on specific aspects of the discipline, such as Humble's excellent books on DevOps, Cohn's books on user stories, and books devoted to specific frameworks, such as *The Scrum Guide*. I saw a gap in the market, though, for something built on the foundation of those books but that goes further. I realized there were hardly any publications that *connected the dots across these essential techniques* while providing guidance specific enough for the practitioner to adapt and apply them on the job. I wrote this book to fill that gap. It offers actionable advice backed up by specific examples that illustrate how to use and adapt agile practices in different scenarios.

The guidance in the book is supported by more than 175 tools, techniques, examples, diagrams, templates, checklists, and other job aids, making it an essential tool kit for any business analysis practitioner or product owner. It synthesizes the analysis and planning guidance of the most widely used agile frameworks and distills the lessons I've learned from the last twenty to thirty years working with agile teams. Over time, I've made my share of mistakes—failing, trying again, and failing better (to paraphrase Samuel Beckett). Along the way, I've learned what works and what doesn't. This book incorporates the lessons learned from those mistakes so that you don't have to learn them the hard way.

The guidance you'll find in this book draws from the collective wisdom of those I've worked with over the years: my colleagues and clients at REI Co-op, Covance, LabCorp, US Food and Drug Administration (FDA), Intact Insurance, TD Bank, BMO Bank of Montreal, Rogers Corp, TELUS, Canada Mortgage, Housing, True Innovation Inc., and many others. I am grateful to them for trusting me to work with them and sharing their lessons learned with me so that I could pay it forward and share them with you.

Agile analysis and planning focuses on improving communication with customers and users so that the business can anticipate and respond effectively to changes in customers' habits and behaviors *even under extreme uncertainty*. At no time in my memory has this felt as important as today. As I complete this book, a pandemic is raging across the globe, and the world is facing a long-overdue reckoning with the consequences of racial and economic disparity. Everything at this moment seems uncertain, from the profound to the mundane. What will society be like at the end of these changes? Will we come together or be further divided? Will the shift from real-world engagement toward online life be permanent? Will remote work become the norm? What about distance learning and online shopping? It's a time of great challenge but also an opportunity for reinvention. It is my wish that this book will help you and the organizations you work for navigate these changes, adapt, and even thrive in these incredibly uncertain times—and in the "new normal" that is to follow.

State-of-the-Art Guidance across Agile Frameworks

This is my third book on business analysis. My earlier books, *UML for the IT Business Analyst* (2005, 2009) and *The Business Analyst's Handbook* (2008), described how to carry out the business analysis function within an iterative-development lifecycle. It's been very gratifying to witness the international success enjoyed by those books, including a Spanish and Portuguese edition and a second release of the UML book. If you liked those books, I am confident you will enjoy this new publication as well. Much has changed, though, since my first publication. This book returns to similar ground but with a refreshed perspective on today's most successful and widely used agile and analysis frameworks and practices. These include:

- DevOps
- SAFe
- Kanban

- Scrum
- Lean software development
- Lean startup and minimum viable product (MVP)
- User stories, Extreme Programming (XP)
- Continuous integration/ continuous delivery (CI/CD)
- Test-driven development (TDD), acceptance test–driven development (ATDD), and behavior-driven development (BDD)
- Full-potential plan
- Discovery-driven planning
- Circumstance-based market segmentation
- Agile Fluency model

In addition, the book is aligned with the following professional certification guides:

- PMI: *Agile Practice Guide*
- IIBA: *Agile Extension to the BABOK Guide v2*
- PMI: *Business Analysis Practice Guide*
- IIBA: *BABOK v3: A Guide to Business Analysis Body of Knowledge*

What Makes This Book Unique?

Unlike many other guides, this book contains everything you need *in one place* to practice effective agile analysis and planning:

- **Detailed guidance:** It's a practical manual that tells you what to do *and* shows you how to do it.
- **Integration with business analysis:** Most books on agile analysis focus solely on agile techniques, overshadowing the use of valuable business analysis techniques such as business rules analysis and process modeling. This book shows you how to insert legacy analysis techniques into an agile process to increase an agile team's productivity.
- **Broad coverage of agile approaches and frameworks:** The book incorporates best practices from today's most widely used agile frameworks, including lean, SAFe, Kanban, and Scrum, enabling you to be effective in any agile environment.
- **Experience-based guidance:** This book is based on years of experience working with companies and teams on improving agile analysis and planning in their organizations, learning what works and when. It's *informed* by today's most effective agile frameworks but *is beholden to none*.

- **Context-based just-in-time learning:** The book presents you with techniques and guidelines in the context in which you'll be using them across the development lifecycle. You learn what you need to know and when you need to apply it.
- **Extensive job aids:** The book includes more than 175 valuable job aids to increase your understanding and effectiveness. These include:
  - Concrete examples and templates that you can use to create analysis and planning artifacts, such as the product vision statement, product roadmaps, story maps, epics, features, spikes, stories, and acceptance criteria
  - Sample diagrams and diagram legends
  - Meeting agendas and other facilitation aids
  - Checklists
- **Contiguous end-to-end case study:** An end-to-end case study runs through the book, enabling you to see exactly how the steps and artifacts feed into each other over the course of an agile development lifecycle.

Furthermore, the book provides clear evidence of the value of business analysis in an agile organization—demonstrating how traditional business analysis combined with agile analysis and planning techniques can produce higher-performing agile teams.

Why Agile Analysis and Planning Is Important for the Enterprise

We know that organizations that adopt an agile approach experience significant benefits. For example, their projects are 37 percent faster to market than the industry average (QSM),[1] and their productivity increases by 16 percent.[2] But we also know that an agile organization can dramatically *improve* its success rates by enhancing its level of competency in analysis and planning.[3] The "Business Analysis Benchmark"[4] showed that project success rates for agile organizations *more than doubled* from 42 percent at the lowest maturity level (level 1) for the competency to 91 percent at the highest maturity level (level 4). Moreover, it found that even modest increases in maturity levels could have a significant impact. For example, a half-step increase from level 2 to 2.5 led to a rise in success rates from 62 percent to 74 percent for agile organizations. (More on this research is presented in Chapter 2.)

1. Quantitative Software Management Associates (QSMA), "The Agile Impact Report. Proven Performance Metrics from the Agile Enterprise," QSMA for Rally Software Development Corp., 2009, 1.
2. QSMA, "Agile Impact Report," 1.
3. The report correlated success to the maturity level of the requirements process, roughly equivalent to what I refer to as *analysis and planning* in the book. The report looked at the impact of maturity level on success rates for different development approaches, including agile.
4. Keith Ellis, "Business Analysis Benchmark—The Impact of Business Requirements on the Success of Technology Projects," IAG Consulting, 2009.

Problems that can be addressed by having effective agile analysis and planning capabilities in your organization include the following:

- Added costs for rework because requirements were sufficiently understood up front
- Delays due to poor team planning and coordination
- Reduced team productivity because work is not being well prioritized across the product
- Poorly managed stakeholder expectations
- Underresourced, overworked product owners
- Challenges scaling agile development because cultural issues within the organization are not appropriately addressed

Today, agile analysis and planning is recognized as an effective approach for addressing these issues and more. Organizations who already have business analysts experienced in traditional business analysis are upskilling them with agile competencies and embracing them as valuable contributors. At the same time, startup technology companies that began their agile journey without a strong business analysis competency are now adding it to their organizations. As they mature, they're finding that the skillset is becoming more relevant to them because of the increased levels of complexity in the business domains they address and in their products' underlying architecture.

The benefits to the business of establishing an effective agile analysis and planning competency include the following:

- *Enhanced ability to anticipate customer need*: Agile analysts use a wealth of techniques to gain a deep understanding of the customer. Root-cause analysis and circumstance-based market segmentation identify the underlying needs of customers and the root causes of the problems they are experiencing. Kano analysis helps the business forecast the capabilities customers would embrace. MVP testing reveals which proposed features are most valuable to customers and validates hypotheses in order to direct development resources.
- *Improved ability to manage change*: Agile analysis increases the ability of teams to *sense and respond to change* and make the appropriate adjustments along the way.
- *Ability to plan effectively:* The competency enables an organization to plan effectively for the short term and long term, whether under conditions of extreme uncertainty or when conditions are well known.
- *Reduced time to market:* Time to market is reduced because agile analysis focuses development effort on a minimal set of high-value features that are further evaluated and enhanced over time.
- *Data-informed decisions:* Agile analysis and planning practices enhance the ability to make *data-informed decisions* by using the lean startup MVP process, A/B testing, and actionable metrics.

- *Reduced rework and delays:* Agile analysis reduces rework and unnecessary delays because the right amount of analysis is performed at the right time.
- *Improved team productivity:* Productivity improves because the team is always working on items of the highest value across the product.
- *Improved stakeholder engagement:* Stakeholders are more engaged due to an incremental, rolling analysis process that involves them throughout the lifecycle.
- *Product owner support:* With a well-developed agile analysis competency, product owners are provided with the support they need to be effective in their jobs. Agile analysis and planning practitioners take on requirements and day-to-day communication with the team so that product owners can focus on the outward-facing aspects of the role.
- *Ability to leverage the business analysis (BA) experience*: By upskilling their existing business analysts and incorporating them into agile organizations, companies can *leverage the experience of seasoned business analysts* to improve team performance on agile initiatives.

Who Should Read This Book

The intended readers for this book can be broadly grouped as follows:

- Business analysis practitioners and product owners
- IT directors and leaders of centers of excellence (CoEs) in business analysis, agile practice, and DevOps
- Educators

The benefits for each type of reader are as follows.

Business Analysis Practitioners and Product Owners

The primary reader for this book is the working professional—a person responsible for the analysis, planning activities, or both, in an agile software development organization. The job titles of those who perform this work vary widely among organizations, as does the distribution of responsibilities between those titles. They include business analysts, team analysts, product owners, proxy product owners, and product managers. This book is for *anyone* responsible for this work in an agile organization—regardless of job title.

If you are a product owner, you can use the knowledge in this book to learn how to

- Organize and coordinate agile teams for peak effectiveness.
- Analyze the market for the product.
- Develop a compelling product vision statement.

- Plan and estimate requirements implementation at all planning horizons.
- Plan MVPs to test hypotheses for the product and make data-informed decisions.
- Prioritize epics and features across the product.

If you're a business analyst, you can use this book to communicate the product vision to the team and help them translate that vision down to smaller requirements units and specifications (e.g., features, stories, and their acceptance criteria). Within these pages, you'll also find detailed guidance on maintaining the product backlog, tracking the progress of stories, story preparation, and estimation. Senior business analysts will learn how to prepare and tailor the agile analysis process for their situation—including setting up the product backlog, gaining consensus on the definition of ready, setting Kanban work-in-progress limits, and determining capacity.

If you're responsible for analysis and planning at any level in your organization, the information in this book will provide you with the confidence and skills to work effectively within any of the popular agile frameworks and practices in use today. If you're an entry-level business analyst or team analyst, you'll appreciate the chapter on fundamentals, the detailed guidance on feature and story preparation, and the wealth of job aids in the book. If you're a product or higher-level business analyst, you'll benefit from the book's strategic guidance dealing with culture, stakeholder analysis business objectives, strategic planning, and scaling considerations.

IT Directors and Leaders of CoEs in Business Analysis and Agile Practice

IT directors and CoE leaders in business analysis and agile practice can leverage the information contained in this book to

- Develop and customize an agile analysis and planning framework that's right for the organization.
- Build a library of CoE resources for analysts and planners using the book's templates, checklists, and examples.
- Craft a strong value proposition to communicate the benefits of agile analysis and planning competency in the organization.

Educators: College or Corporate Trainers or Learning Directors

If you're an educator, you can use this book as a basis for building a curriculum in agile analysis and agile development that incorporates today's most popular proven concepts, tools, and techniques. Each chapter describes clearly defined objectives and summaries, leveraging a running case study with sample solutions that you can use for group workshops.

If you are interested in using the book to build a training curriculum, please contact me for additional content and services, including PowerPoint presentations, eLearning offerings, and in-house training. Send email inquiries to info@nobleinc.ca or check online at https://www.nobleinc.ca.

How This Book Works

Think of this book as the voice of a coach in your ear as you walk through the agile analysis and planning process. Each chapter guides you through the activities performed at that point in the agile development cycle. The steps are illustrated with a running case study so that you can see how analysis and planning artifacts evolve the course of development and how they connect to each other. Additional examples are provided so you can see how to apply the techniques to other situations.

I should note that the sequencing of analysis activities in the chapters is only a rough guide because agile analysis and planning is not a sequential process. You rarely complete a planning or analysis activity in one step; more typically, you perform some of it up front and the rest of it in a rolling fashion. Moreover, activities are often carried out concurrently. For the most part, the chapters are sequenced based on the order in which activities are *first* performed.

How to Read the Book

There are two ways to read this book:

1. The traditional way, front to back. That's what I'd advise if you're new to agile or business analysis.
2. By skipping to the parts that are most important to you. You may prefer to read the book this way if you have some agile experience and want to fill in your knowledge gaps. In that case, I'd recommend you
   - First scan Chapter 3 to fill gaps you may have in fundamental concepts.
   - Next, read Chapter 4 to gain a bird's-eye view of the agile analysis and planning activities covered in this book.
   - Then go to the chapters that deal with the activities that interest you. Each chapter is self-contained, dealing with one or more analysis or planning activities. When it refers to a topic that was introduced earlier in the book, I've included a cross-reference in case you're reading the book in a nonsequential manner.

Overview of Chapters

The following is a brief description of each chapter:

Chapter 1, The Art of Agile Analysis and Planning	Presents a brief, personalized look at the art of agile analysis and planning based on lessons learned from my life both as an artist and as an analyst. It explains why I believe the agile approach is conducive to the creative process.
Chapter 2, Agile Analysis and Planning: The Value Proposition	Presents the value proposition for developing an effective competency in agile analysis and planning in an organization.
Chapter 3, Fundamentals of Agile Analysis and Planning	Explains the principles, frameworks, concepts, and practices that underlie the agile analysis and planning competency and the rest of this book, such as lean, Kanban, Scrum, DevOps, and user stories.
Chapter 4, Analysis and Planning Activities across the Agile Development Lifecycle	Provides an overview of planning and analysis activities across the agile product development lifecycle. Three scenarios are covered: short-term initiatives with planning horizons up to three months, long-term initiatives up to five years, and scaled agile initiatives. The Agile Analysis and Planning Map in this chapter provides a bird's-eye view of the process. This map is referenced in later chapters so that you can see where you are in the development process as you progress through the book.
Chapter 5, Preparing the Organization	Explains how to prepare an organization for agile software development, including guidance on forming effective agile teams, managing stakeholders' expectations, and guidelines for governance, finance, and marketing groups. (Please note that guidelines specific to scaled organizations are covered in Chapter 17.)
Chapter 6, Preparing the Process	Describes how to prepare the agile analysis and planning process. Senior analysts and CoE leads will learn how to customize the right agile framework and practices for their situation and how to fine-tune process parameters like work-in-progress limits and the definition of ready to optimize team productivity.
Chapter 7, Visioning	Covers early analysis activities to envision a new product or significant enhancement. Product owners can use the information in this chapter to craft effective product and epic vision statements and specify objectives. Analysts will learn to communicate the product vision to the team and continue the visioning process through root-cause and stakeholder analysis. The chapter also covers the specification of "leap of faith" hypotheses in preparation for MVP planning.

Chapter 8, Seeding the Backlog—Discovering and Grading Features	Focuses on the discovery and specification of the initial items in the product backlog. Analysts and product owners should read this chapter to learn how to prioritize and specify features and nonfunctional requirements for the product or release backlog. Prioritization tools covered in this chapter include Kano analysis, cost of delay, and weighted shortest job first (WSJF).
Chapter 9, Long-Term Agile Planning	Explains how to perform long-term planning for horizons of six months to five years. Product owners and business analysts can use the information in this chapter to create a long-term product roadmap, specify goals, objectives, assumptions, and metrics for the planning period. The chapter explains the full-potential plan—an approach for planning transformative change over a three- to five-year period. It describes the agile approach to planning using MVPs to test assumptions and determine what to include in the product. The chapter also explores deployment strategies and options for the long-term implementation plan, including guidelines for when to use narrow and deep versus wide and shallow approaches.
Chapter 10, Quarterly and Feature Preparation	Describes how to prepare upcoming features. When the team is using a Kanban approach, this preparation occurs on a rolling basis. When a timeboxed planning approach is used, it occurs before quarterly planning for the group of features lined up for the quarter. This chapter applies to both approaches. The chapter includes both agile and legacy tools, including the feature definition of ready, ATDD, specification of feature acceptance criteria using BDD, value stream mapping, journey mapping, and process modeling.
Chapter 11, Quarterly and Feature Planning	Describes how to plan an upcoming feature or quarter. The chapter applies to teams that use timeboxed planning approaches (in which case all features for the quarter are planned together) and those that use a single-item flow-based approach (in which case a single feature is planned). The chapter begins with guidance on when to use which approach. It explains how to plan and estimate features using methods and approaches such as the Planning Game, Planning Poker, Delphi estimation, story points, ideal developer days, as well as the no-estimating approach.

Chapter 12, MVPs and Story Maps	Demonstrates how to use MVPs and story maps to plan the delivery of learning and value within short timeframes. MVPs are minimal versions of the product that enable the product owner to test hypotheses and make data-informed decisions about development investment and resource allocation. Story maps are visual representations of the plan that indicate the operational and implementation sequencing of stories.
Chapter 13, Story Preparation	Covers the analysis of stories before implementation. This preparatory work occurs on a rolling basis if the team is using Kanban. It is performed before iteration planning when a timeboxed approach such as Scrum is used. This chapter covers both contexts. Tools covered include the INVEST story-writing guidelines, patterns for splitting stories, and the specification of story acceptance criteria using BDD and the Gherkin syntax.
Chapter 14, Iteration and Story Planning	Covers planning for a short-term horizon of one week to one month. The chapter explains how to determine team capacity and how to forecast which stories will be done. Planning tools covered in this chapter include the iteration backlog, developer task board, and Kanban board.
Chapter 15, Rolling Analysis and Preparation—Day-to-Day Activities	Describes day-to-day rolling analysis and planning activities. The chapter includes guidance on ongoing story and feature preparation, the daily stand-up, updating the developer task board, burndown chart, cumulative flow diagrams, and more.
Chapter 16, Releasing the Product	Covers the final preparations for general availability (GA), also known as production release. The chapter includes guidance on operational preparations, value validation, alpha testing, and beta testing. It also examines the pros and cons of using a hardening iteration before GA.
Chapter 17, Scaling Agility	Describes the analysis and planning challenges faced by large agile organizations. It provides actionable guidance for scaling the agile organization, the process, and the product backlog. This chapter explains and incorporates best practices for scaled agile development, including DevOps, CI/CD, ATDD, BDD, and SAFe.
Chapter 18, Achieving Enterprise Agility	Explores agile analysis, planning, and product development from the enterprise perspective—beyond the IT context that has been the main focus of the rest of this book. The chapter includes thirteen practices for optimizing an enterprise's responsiveness to change.

Appendixes	Provide a collection of useful tools for the agile analyst and planner, including checklists, templates, and agendas for easy reference on-the-job or during training. Also included is a detailed case study illustrating discovery-driven planning—the financial planning counterpart to the data-driven development approach described in the rest of this book.

Repeating Book Features

This book contains several repeating features to make it easier to find what you need. They are identified with icons as follows:

Checklist: Useful lists for the practitioner (e.g., a checklist of stakeholders)

Example: A concrete example of an artifact

Template: A template for creating an artifact (text or diagram)

Tips and Guidelines: Useful tips, guidelines, and formulae for the practitioner

Cross-reference: Cross-reference to another book section, where you can learn more about a topic

Introducing the BLInK Case Study

This book follows one case study through the product development lifecycle, from visioning to continuous value delivery. The case study is included so that you can immediately see how to apply the techniques and to connect them over the course of product development. (If you're not a fan of case studies, you can skip or quickly scan those sections. I won't be offended, and you won't miss any new concepts.)

Many people learn best by doing. I am one of them. If that describes you, I urge you to actively work through the case study sections yourself, comparing your deliverables with those I've provided in the book. It's perfectly okay for your deliverables to be different from those in the book or for you to come up with different results. The outputs will depend on the conversation you have (or *imagine* having) with stakeholders and how you choose to document them. What's important is that you can justify any decisions you've made.

The example I've chosen for this book revolves around a fictionalized insurance company called Better Living (BL) Inc. As the case study opens, BL is looking to develop a usage-based insurance (UBI) product that uses data from Internet of Things devices to personalize health insurance costs and benefits. The product is to be named BLInK —Better Living through Insurance Knowledge.

One reason I chose this case study is that it's current: as I started work on this book, I was working with an insurance client on a similar product. But the main reason I chose it is that it involves the analysis of an innovative product within a mainstream business—just the type of initiative where one is most likely to find an agile business analyst. As the case study opens in Chapter 7, the product is in its early visioning phase. Throughout the rest of the book, we follow the agile analysis and planning of this product through to implementation and delivery.

Certification Information

This book is mapped to the following professional certification guides:

- *BABOK v3: A Guide to the Business Analysis Body of Knowledge*
- *Agile Extension to the BABOK Guide v2*
- *The PMI Guide to Business Analysis*
- *The Agile Practice Guide*

For a detailed mapping of chapters to the guides, please see Appendix A.2.

Register your copy of *The Agile Guide to Business Analysis and Planning* on the InformIT site for convenient access to electronic templates, updates, and/or corrections as they become available. To start the registration process, go to informit.com/register and log in or create an account. Enter the product ISBN (9780134191126) and click Submit. Look on the Registered Products tab for an Access Bonus Content link next to this product and follow that link to access any available bonus materials. If you would like to be notified of exclusive offers on new editions and updates, please check the box to receive email from us.

Thanks

No person gets anywhere on their own; we all do it with the help and mentoring of others. First and foremost, I want to thank the many colleagues and mentors who have generously shared their knowledge throughout my career. A special thanks to Alain Arseneault, with whom I worked closely at BMO Financial Group and in many other contexts. He has been enormously instrumental in the development and success of business analysis internationally through his pioneering work developing the bank's competency and later through his involvement with IIBA in multiple capacities, including acting CEO. Alain has been incredibly generous with support and guidance over the years, and he has gone beyond-the-beyond with this assistance on this book. I can't thank him enough.

Often, transformative change is the result of a change *agent*—an individual with vision and a strategy for executing it. I've met these talented individuals in many organizations, and they've often wielded influence far beyond their formal titles, largely as a result of the respect in which they are held by their peers. In this regard, I want to thank Abhijeet Mukherjee, with whom I worked at UST Global to raise the maturity level of business analysis across the corporation. Thanks, too, to Saurabh Ranjan, who was UST's COO at the time and a champion and primary sponsor for Global BA and Strategic Consulting CoE-related programs and initiatives. I also want to thank three other leaders of change in their organizations—Trenton Allen at REI Co-op; Andre Franklin at Covance; and Dana Mitchell, agile practice lead for agile transformation at TD Bank Securities—for trusting me to work with their teams and for sharing their insights about agile analysis and planning practices.

A big shoutout as well to the *early agile adopters*, clients who saw the promise of iterative, incremental development right from the beginning and were true pioneers in business domains that were not particularly open to agile development and analysis at the time. Foremost among these was John Beattie, former VP at TELUS—an agile visionary and someone with whom I had the immense pleasure of working. I'd also like to thank Tim Lloyd from True Innovation for his helpful encouragement and collaboration over many years.

Special thanks to Karl Wiegers, whose early writing on requirements spurred my interest in business analysis, for sharing his experience and guidance as a writer and analyst. He is a living example of the principle of paying it forward. Thanks also to Christopher Edwards for his valuable input and detailed notes on the last chapters. Without all of these people, and many others too numerous to name, this book would not exist in its current form.

Thanks also to my technical editors, Ron Healy, for the care he took to consider the guidance in this book against his own experience, and to Clifford Berg, who encouraged me to expand the coverage of DevOps practices and challenge my own assumptions, and helped me find the most useful guidance to highlight in several of the book's key chapters. Both editors gave me precisely what I was looking for—*a hard time*—and the book is much better for their efforts. Thanks also to Tracy Brown, my development editor, for her support and guidance. A huge shoutout to Haze Humbert, executive editor at Pearson, for cajoling, encouraging, and generally kicking my ass to get this book done, and to everyone else on the Pearson team, including Rachel Paul, Menka Mehta, Julie Nahil, and Carol

Lallier. Thanks, as well, to Christopher Guzikowski, my first editor at Pearson during the early days of the book, for believing in the book and supporting it when it counted most.

This book is especially indebted to the almost weekly telephone calls about its themes over the four years of its making with a lead developer at Hootsuite, one of Canada's most innovative agile companies. His input and insights are so interwoven into this book that he is very much a collaborator. It is an added pleasure that he is also my son, Yasha Podeswa.

This page intentionally left blank

About the Author

Howard Podeswa is an established author, professional artist, and sought-after speaker at international conferences. His paintings have been shown in numerous exhibitions across Canada and internationally, including the United States, Italy, and South Africa. His work is held in numerous private and public collections.

Podeswa's career in software development began when an academic background in nuclear physics led to a job working on a nuclear-accident simulation program for Atomic Energy of Canada Ltd. Since then, he has been enthralled by software development and often found himself on the cusp of change as a developer of innovative systems in transportation, laboratory automation, and communications. From the 1990s onward, he has been helping large organizations transition their planning, analysis, and requirements engineering (RE) processes to agile practices across a broad range of sectors, including telecommunications, banking, government services, insurance, and healthcare.

He plays a leading role in the industry as a designer of agile and business analysis (BA) training programs for companies and higher education institutions, including Boston University Corporate Education Center and Humber College; as a reviewer of the BA profession's standard books of best practices (*BABOK* [IIBA] and *Business Analysis for Practitioners—A Practice Guide* [PMI]); and as an author whose books have become staples in many BA libraries: *The Business Analyst's Handbook* and *UML for the IT Business Analyst*.

Podeswa, through his role as director for Noble Inc., has provided agile and BA training programs and consulting services to clients across the globe in the private and public sectors. Companies that have benefitted from his services include the International Standards Organization (ISO), Moody's, the Mayo Clinic, TELUS, UST Global, BMO, TD Bank, Intact Insurance, Labcorp, the US Food and Drug Administration (FDA), Canada Mortgage and Housing Corporation (CMHC), Bell Nexia, and Thomson Reuters.

This page intentionally left blank

10.2 This Chapter on the Map

As indicated in Figure 10.1, we'll be examining the following items in the Seeding the Backlog zone: epic preparation, feature preparation, acceptance test–driven development (ATDD) / behavior-driven development (BDD), persona analysis, journey mapping, value stream mapping, and process mapping.

10.3 Overview of Features

Since we're about to focus on features, let's quickly review some fundamental concepts about them.

A **feature** is a product-level work item that can be completed by one or more teams within one quarter or release cycle. The feature may be expressed in the Connextra format—for example, "As a member, I want to receive messages and notifications so that I can respond to issues that require my immediate attention."

A feature is bigger than a story but smaller than an epic. The relationships can be summarized as follows:

Epic > Feature > Story

Features often begin as epics. As we learned earlier, in Chapter 7, "Visioning," an **epic** is a product-level work item that may require multiple teams over multiple quarters and may span product areas, business areas, and value streams. An example of an epic is the introduction of home delivery across a product line to increase sales revenues by 20 percent. Chapter 7 explains how to prepare an epic by articulating the epic vision and leap of faith hypotheses. It also explores the MVP process for determining the minimum marketable features (MMFs)—the high-value features to develop. The next step is to prepare the upcoming features. This chapter focuses on that preparation.

10.3.1 Examples of Feature-Length Change Initiatives

As discussed in previous chapters, you decompose large work items into stories—small work items that deliver value but require no more than a few days' work—in order to shorten the feedback cycle and smooth the flow of work. If that's the case, why not dispense with epics and features and treat all requirements as stories? You can, if the team is exclusively tasked with small enhancements and bug fixes. Frequently, though, teams are asked to work on items that exceed the maximum size for a user story. A larger container—an epic or feature—is required to encapsulate the high-level functionality and objectives that it will deliver. Epics and features also include acceptance criteria (AC) that describe the product's behavior when stories are strung together in an end-to-end workflow. Examples of work items larger than a story include the following:

10.3.1.1 Deliver a New or Improved Value Stream or Process

A work item to create a new process or value stream—or reengineer an existing one—typically exceeds the maximum size of a user story and must be managed as a feature. Feature preparation activities may include value stream mapping and modeling of the current and future processes.

10.3.1.2 Nontrivial Change to a Mature Product

When a product is young, it's relatively easy to add a new capability because there aren't too many existing ones that the new capability might affect. However, as the product matures and accumulates a broader range of capabilities and components, it becomes harder to add or change a capability because it can affect so many existing parts. As a result, the change request must be classified as a feature.

Consider Customer Engagement One (CEO), the app being developed by our example company, CEO Inc.[1] Suppose the first version of the product allows customer support agents to view messages from two sources—each with its own format and rules. If the product owner (PO) wants to add a third message source, such as email, doing so affects only one function—viewing. This requirement is achievable within a few days, so you manage it as a user story.

Now suppose CEO has grown into a mature product with features to ingest, view, triage, tag, respond to, assign, and resolve messages. It's much more difficult to add a new message source because all of the existing features have to be adapted. A change of this type now takes weeks to implement and involves multiple teams. Consequently, you treat it as a feature (or epic, if it spans quarters), not a user story.

10.3.1.3 Implementing a Use Case

A use case is a usage of the product or system, typically sized to deliver a goal a user can accomplish through a single interaction with the product. Examples of use cases include the following:

- Submit a college application.
- Open an account.
- Place an order.

Each use case represents all the ways the interaction can play out, including successful and unsuccessful scenarios that the solution must support. The effort to implement all the scenarios of a use case typically exceeds the maximum story size. Consequently, you manage the use case as a feature and each scenario or set of related scenarios as a story. For example, you represent the *Place an order* use case as a feature. The user stories for the feature include the following, expressed in an informal format:

- Place an order (basic flow: no options).

1. This example was adapted from one provided by Yasha Podeswa in a conversation with the author, August 2019.

- Place an express order.
- Place a backorder.

10.4 Benefits of Feature Preparation

Sometimes I have to convince teams that feature preparation is not only allowed in agile development but should be encouraged and included in the plan. By preparing features before quarterly planning sessions begin, you facilitate improved capacity planning: developers can provide better estimates because they have a clear understanding of what's being requested. Furthermore, by preparing features before their implementation, you enable hyperproductive teams.[2] Developers can begin work on the solution without having to wait for key information or technical preparations. Collaborating teams can work in parallel with confidence because the feature's acceptance criteria (AC) and process models specify how the pieces must fit together when assembled. If integration errors show up, they're caught quickly because the feature AC are also used as the basis for specifying and executing automated high-level integration tests.

10.5 Feature Preparation Activities

This chapter focuses on *preparation*, while the next chapter focuses on *planning*. There is no strict line between the two, but in general, **planning** is about commitment—determining what features and goals will be delivered and gaining the commitment of collaborating teams to do the work. **Preparation** is the work to make an item ready for planning and implementation.

The outcome of feature preparation is a **ready** feature—one that is suitable for quarterly planning and able to be implemented without undue delay or rework. For example, a ready feature is prioritized and can be accomplished in three months or less by one or more teams.

Feature preparation activities include analysis and technical preparation. The analysis activities may include the items summarized in the following checklist.

✓ **Checklist of Feature Preparation Analysis Activities**

- ☐ Specification of features and AC
- ☐ Context analysis
- ☐ Stakeholder analysis
- ☐ Persona analysis

2. Jeff Sutherland, "Scrum: What Does It Mean to Be Ready-Ready?" (OpenViewVenture, 2011), https://www.youtube.com/watch?time_continue=3&v=XkhJDbaW0j0

- ☐ Journey mapping
- ☐ Value stream mapping
- ☐ Process modeling
- ☐ Use-case modeling
- ☐ User-role modeling workshops
- ☐ Initial splitting into stories

This chapter covers all of the items in the preceding list except for the last. The decomposition of features into stories (aka story splitting) is covered in Chapter 13, "Story Preparation."

To be clear, you don't perform *all* of the preparatory activities in the preceding checklist for every feature. The chapter provides guidance on activities to *consider* doing—but only do what's necessary for the situation.

Guidelines for splitting features into user stories are provided in Chapter 13, section 13.13. Additional guidelines for preparing features on a scaled initiative can be found in Chapter 17, section 17.9.12.

Technical preparation involves the drafting of a solution design, creation and testing of proofs of concept and prototypes, and readying the **architectural runaway**—a task that includes the specification of service communication protocols, identification of components, and creation of infrastructure. While this book focuses on analysis issues, we do review some of the models used in technical preparation that you should be familiar with as an analyst. These include the following:

- Context diagrams
- Communication diagrams
- Data-flow diagrams
- Block diagrams

10.6 Timing of Feature Preparation

When do you begin the preparation of features? The lean guideline is to wait until the last responsible moment (LRM)—the point at which any further delay would result in unacceptable costs. How you apply this principle depends on the planning approach you're using.

In a Kanban system, you prepare each feature as it approaches the top of the backlog, with a lead time of about six weeks for large features and two to four weeks for smaller ones.

If the teams are using the alternative planning approach—timeboxing—you prepare the group of features lined up for the upcoming quarter starting about halfway (six weeks) into the prior quarter. Some organizations prepare these features in a reserved iteration (e.g., SAFe's Innovation and Planning [IP] Iteration),[3] but this is generally not advised. We look at arguments for and against reserved iterations (aka hardening iterations) in Chapter 17.

10.7 Assessing Readiness

Use the checklist in Appendix A.7 to assess whether or not teams are ready for quarterly planning. Conditions in the checklist include that a vision, roadmap, and impacted users have been specified and that sufficient features (about ten to twenty) are ready.

10.7.1 Using the Feature Definition of Ready (Feature DoR)

Use the **feature definition of ready (DoR)** to determine if a feature is ready to be included in the quarterly plan or (in Kanban) to advance on to development.

The following are examples of the feature DoR conditions we saw in Chapter 6, "Preparing the Process."

- The feature is right-sized: The feature is small enough to be implemented within a quarter by one or more teams.
- The feature has no (or minimal) dependencies on other features.
- The feature is valuable.
- All teams are committed.
- The feature is estimable: The feature is understood well enough to be estimated.

For more on the feature DoR, see Chapter 6, section 6.5.7.6.

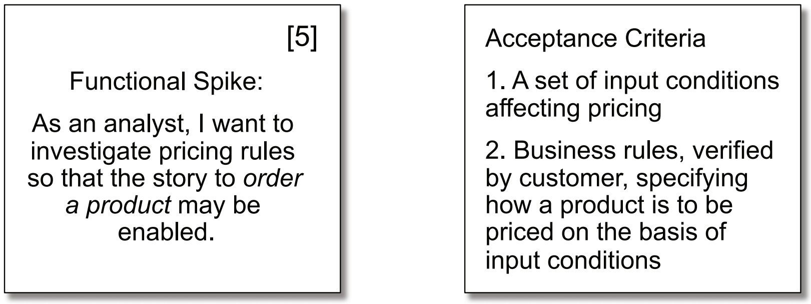
10.8 Accounting for Preparation Work: Tasks and Spikes

Once you've flagged the need for preparatory analysis, how do you account for that work in your plans? If the analysis will be performed during the iteration in which it's flagged, represent it as a developer task. A **developer task** is a work item carried out by an individual team member. (The term *developer* is deceiving. Analysis, design, testing, and coding are all treated as developer tasks.) Developer tasks are posted on a developer task board.

We look at developer tasks and developer task boards in Chapter 15, sections 15.4, 15.6, 15.7.3, and 15.7.5.

3. Richard Kastner and Dean Leffingwell, *SAFe 5.0 Distilled: Achieving Business Agility with the Scaled Agile Framework* (Boston: Addison-Wesley, 2020), 262.

If you plan to defer the analysis work to a future iteration, you'll have to add it to the product backlog. However, you can't represent it as a user story because it doesn't result in working code. Instead, you manage the analysis as a **functional spike**, also known as an **enabler story**. We'll look at functional spikes in Chapter 13. Figure 10.2 is an example of one.



[5]

Functional Spike:

As an analyst, I want to investigate pricing rules so that the story to *order a product* may be enabled.

Acceptance Criteria

1. A set of input conditions affecting pricing

2. Business rules, verified by customer, specifying how a product is to be priced on the basis of input conditions

Figure 10.2 *Example of a functional spike*

Figure 10.2 illustrates the functional spike to investigate pricing rules. The value that it delivers is expressed in the "so that" clause: the spike enables a future story to order a product. The spike is assigned five story points, indicating the estimate and time limit for the analysis.

Once you've identified the analysis activities required to prepare the feature, the next step is to perform them. The following sections provide guidelines for performing feature AC specification, persona analysis, journey and value stream mapping, and process and use-case modeling.

10.9 Specifying Features and Their Acceptance Criteria

Meet with business representatives, developers, and testers (sometimes called "the Triad") to describe the feature in a way that clearly communicates the requirement. Chapter 8, "Seeding the Backlog—Discovering and Grading Features," section 8.7, provides guidelines for specifying features using the Role-Feature-Reason (Connextra) template. Coach stakeholders and the team to use the template, but don't force its use where the resulting wording is unnatural and impedes understanding.

Then, specify feature AC. AC play a central role in agile analysis: they serve as requirements and as the basis for user acceptance testing (UAT). For the first release of the feature, specify just enough AC to define an MMF—the minimum functionality required to deliver value that the customer would view as significant.

As an analyst, you support feature AC specification. You support ATDD guidance by ensuring AC are specified before work on the feature begins so that they can serve as **specifications by example**. The AC tell the developers how much functionality must be delivered for the item to be releasable—providing them with the information they need to estimate the feature for capacity planning. The AC also serve as test scenarios to validate the solution. These scenarios describe how the product must behave when user stories are strung together in a larger workflow or value stream. A common approach is to specify the AC in a feature file in the Gherkin syntax so they can be interpreted by a test automation tool such as Cucumber.

AC and estimates are so intertwined that you should encourage stakeholders to discuss them at the same time with developers and QA professionals so trade-offs can be explored. This is the principle behind the Triad approach, discussed in Chapter 13.

For more on the Triad, see Chapter 13, section 13.6.3.

10.9.1 Specifying Epic Acceptance Criteria

Specify epic AC that communicate, at a high level, the minimum requirements for completion. In Chapter 7, we saw the following epic example. Its AC expresses the epic's business objective, "legacy system can be retired."

e.g.

Epic: Modernize customer loyalty program.

Acceptance Criteria: Implementation of this epic means that the legacy system can be retired.

The following AC examples specify minimum capabilities for an epic.

Epic: As a planner, I want to introduce dropship capability to increase top-line sales without the inventory ownership expense.

Acceptance Criteria:

Provide the ability to identify dropship-eligible product.

Enable financial reporting (sales $/units, sell-through %, inventory ownership) for all dropship-eligible products.

Identify when dropship-eligible product is no longer available for sale.

Provide the ability to execute a clearance (markdown) price change for dropship-eligible product.

Epic: Implement payment platform.

Acceptance Criteria: Completing this epic allows multiple payment types to be used interchangeably.

10.9.2 Specifying Feature Acceptance Criteria

Like epic AC, feature AC do not have to cover all possible scenarios. Instead, begin by specifying an MMF that includes only the minimum level of functionality needed for the feature to be seen as valuable by customers.

Following is an example of feature belonging to the epic we saw earlier: "As a planner I want to introduce dropship capability to increase top-line sales without the inventory ownership expense." Its AC are specified in brief descriptive text, also known as scenario titles.

Feature: Enable dropship product identification in assortment planning.

Acceptance Criteria:

Scenario: Specify a dropshipped product. (success)

Scenario: Specify a product ineligible for dropshipping. (failure)

Scenario: Search for dropshipped products satisfying search attributes.

Following is an example we saw in Chapter 8.

Feature:

As an incident manager, I want to manage incidents from a single interface so that I can view and prioritize issues across all sources.

Acceptance Criteria:

I can view and manage scheduling delays.

I can view and manage nonemergency incidents.

I can filter/sort/rank all incidents by defined attributes.

10.9.3 The Analyst Contribution

As an agile analyst, you support ATDD by facilitating Triad conversations between stakeholders, QA, and developers about AC and by specifying AC, as discussed earlier. However, you should review and adjust your contribution over time based on experience. Options for your involvement in feature AC include the following:[4]

- You own the feature files—or the team as a whole owns them.
- You write the AC, scenario titles, and Gherkin given/when/then specifications—or you write AC and scenario titles, and QA professionals write the given/when/then specs.

4. Ian Tidmarsh, "BDD—An Introduction to Feature Files," Modern Analyst, https://www.modernanalyst.com/Resources/Articles/tabid/115/ID/3871/BDD-An-introduction-to-feature-files.aspx

10.9.4 Analyze AC During Triad Meetings

Analyze AC for epics and features incrementally, through collaborative sessions with business stakeholders (representing the customer), testers, and developers—the Triad.

Before committing a feature to development, facilitate Triad discussions to specify high-level AC in the language of the business. The AC and conversations clarify the requirements to stakeholders, testers, and developers. Continue to meet with the Triad to refine the AC with more specific test scenarios.

See Chapter 13, section 13.6.3, for more on the Triad.

This chapter focuses on feature preparation, but you also need to prepare stories and their AC. Story preparation and AC are discussed in Chapter 13.

10.9.5 Specifying AC in the BDD Gherkin Syntax

The Gherkin syntax is widely used because it can be easily interpreted by stakeholders, testers, and test automation tools. Typically, you begin by writing story AC informally; then, as the story approaches development, you specify test scenarios in Gherkin feature files. Gherkin includes keywords such as *given*, *when*, and *then* to identify standardized aspects of test scenarios.

Gherkin Template

Scenario: <<scenario title>>
  Given <<precondition>>
  When <<trigger>>
  Then <<postcondition>>

For example, you create the following feature to introduce dropship capabilities.

Feature: Introduce Dropship Capability

As a planner, I want to introduce dropship capability for the company to increase top-line sales without the inventory ownership expense.

Acceptance Criteria

* Provide the ability to identify dropship-eligible product.
* Provide the ability to execute a clearance (markdown) price change for dropship-eligible product.
* Enable financial reporting (sales $/units, sell-thru %, inventory ownership) for all dropship-eligible products.
* Identify when dropship-eligible product is no longer available for sale.

- Indicate operational workflow on a story map backbone.
- Indicate how feature implementation will be sequenced in the story map ribs.

12.2 This Chapter on the Map

As shown in Figure 12.1, the chapter examines story mapping and MVP in the Quarterly Inception/Feature Inception zone.

12.3 MVPs and Story Mapping: How the Tools Complement Each Other

The primary objective of quarterly/feature planning (the subject of the last chapter) is to develop a plan indicating how goals and capabilities will be delivered over the planning horizon. That much is true for both agile and traditional planning. What makes an agile plan different is that its goals—especially at the start of new product development—are often *learning* goals, validated through MVPs, experimental versions of the product or feature designed to test hypotheses and deliver learning. The learning that is derived from this process is fed back into the agile plan—impacting subsequent goals and features that will be delivered.

MVPs and quick wins often require workarounds for steps that have not yet been implemented. Story maps provide a convenient way to view an end-to-end workflow at each time interval so that stakeholders and the team can visualize gaps where workarounds are required. Beyond their use for MVP planning, story maps are useful tools for planning features so that workflows are supported and meaningful value is delivered to the customer on a regular basis (e.g., at least every iteration or one- to two-week period).

Both tools are covered in this chapter. We begin with MVP planning.

12.4 MVP Planning

When a product is a new-market innovation, you can't prioritize features reliably upfront because customers themselves often won't know what they want until they see it. The lean startup approach,[2] introduced earlier in this book, addresses this problem by running experiments on customers—short-circuiting "the ramp by killing things that don't make sense fast and doubling down on the ones that do."[3]

2. Eric Ries, *The Lean Startup* (New York: Random House, 2011).
3. Brad Smith (CEO, Intuit), as quoted in Ries, *The Lean Startup*, 35.

12.4.1 What Is an MVP?

A minimum viable product (MVP) is a low-cost, experimental version of the product or feature used to test hypotheses and determine if it's worth fully investing in it. According to Eric Ries, the inventor of lean startup, an MVP is "that version of the product that enables a full turn of the Build-Measure-Learn loop with a minimum of effort and the least amount of development."[4] MVP is not (as often thought) the first version of the product released to the market. It's a version meant for *learning*—a means to test hypotheses and to determine the minimum set of features to include in a market-ready product. The minimal *releasable* version of the product is referred to as the **minimum marketable product** (MMP).

12.4.2 MVP Case Study: Trint

You only really understand why MVPs are so crucial to the success of innovative product development when you see a real example of the process. That was the case as I followed the story of Trint, a company founded by Emmy-winning reporter, foreign and war correspondent (and good friend) Jeffrey Kofman. Like many late-stage entrepreneurs, Kofman set out to solve a problem he understood intimately because it had bothered him throughout much of his previous professional life: every time Kofman had to transcribe an interview by hitting PLAY, STOP, TRANSCRIBE, and REWIND, he couldn't understand why he was still using a process that had remained virtually unchanged since the 1960s and 1970s. Why wasn't artificial intelligence (AI) being used to automate the speech-to-text transcription? He knew the reason: journalists can't risk inaccuracies. Since AI makes mistakes, journalists wouldn't use an AI-based product unless there was a way to verify the content. The real problem, then, was how to leverage automated speech-to-text in order to get to 100 percent accuracy.

Kofman knew that if he could solve that problem, he would have a winning product. Furthermore, he knew that if his team could solve it for journalists—whom he knew to be unforgiving—they could solve it for anybody. He concluded, therefore, that the most important leap of faith hypothesis for the product was that the company could find a way for users to correct errors in place in order to deliver transcripts that could be verified and trusted. As Kofman saw it, his team needed to create a layer on top of AI (the automated speech-to-text component) so that the AI part would do the heavy lifting of transcription, allowing the user to focus on quicker tasks: search, verify, and correct. He believed that by using this approach, he could reduce the time to perform a task that would normally take hours to complete down to minutes or even seconds. From earlier chapters of this book, you'll recognize Kofman's steps as the beginning of the MVP process: the articulation of the problem, vision, and leap of faith hypotheses for the product.

To create the MVP, Kofman gathered a team of developers with experience in audio-to-text alignment using manually entered text. He challenged them to hack together an MVP version that would automatically transcribe speech to text and allow a user to edit it.

The company's first MVP was built in just three months. Kofman decided to use some of his limited seed funding to invest in user lab testing. He brought in a group of journalists

4. Ries, 77.

for the testing day. Interestingly (as is often the case), the first MVP was "wrong." While the journalists liked the concept, they struggled to use the product, finding it annoying to switch back and forth between editing and playback modes. (The original design used the space bar as a toggle between modes *and* as the text space character during editing, confusing users.) As Kofman told me, "Good innovative products should solve workflow problems; this was creating new ones." And so, using feedback from the MVP, he asked the developers to build a new user experience with a better workflow.

MVP isn't just about one test; it's a process. Fifteen months into the project, in early 2016, the company developed a more refined version of the MVP. Kofman was ready to prove his hypothesis that there was a strong market for the product. At this point, the product provided much of the core functionality needed by users, such as the ability to search for text to locate key portions of an interview. However, it still lacked key components required to make it fully ready for the market. For example, there were no mechanisms for payments or pricing.

Through his extensive network of journalistic colleagues, Kofman let it be known that they would be opening up the product for free usage during one week of beta testing. When the testing began, things proceeded normally until an influential journalist at National Public Radio sent out a highly enthusiastic tweet, causing usage to soar. At ten thousand users, the system crashed. It took the company two days to get back online, but the test proved beyond a doubt that there was a market for the product.

Today, Kofman views that one day of MVP lab testing as perhaps the most important action taken by the company in its early days because it caused developers to change direction *before* spending a lot of time and money on a failed solution. The lesson, as Kofman tells it, is this: "You *have* to test your ideas out on *real* people"—the people who will actually use your product.

In previous chapters, we examined how to identify the leap of faith hypotheses that must be tested and validated for the product to be viable. Now, we focus on the next step: planning the MVPs that will test those hypotheses.

12.4.3 Venues for MVP Experiments

Since an MVP is only a test version, one of the first things to consider is where to run the test and who the MVP's testers will be. Let's explore some options.

12.4.3.1 Testing in a Lab

A user testing lab may be internal or independently operated by a third party. Testing labs provide the safest venue for testing, making them appropriate for testing in highly regulated mainstream business sectors, such as banking or insurance, where there is minimal tolerance for errors. Because the lab setting provides an opportunity to gain deep insight into users' experience of the product, it's also an ideal venue for MVP testing at the beginning of innovative product development when it's critical to understand customer motivations and the ways they use the product.

The testers should be real users. However, in cases where the requirements are stable, proxies may be used (e.g., product managers with a strong familiarity with the market).

Include testers familiar with regulations governing the product, such as legal and compliance professionals, to identify potential regulatory issues.

12.4.3.2 Testing MVPs Directly in the Market

The most reliable feedback comes from MVP-testing in the marketplace to a targeted group of real customers. Consider this option for *new-market* disruptions, where first adopters are often willing to overlook missing features for novelty. This option is also advised for *low-end* disruptions, where customers are willing to accept reduced quality in return for a lower price or greater convenience.

12.4.3.3 Dark Launch

Another way to limit negative impacts during MVP feature testing is to **dark-launch** it—to stealthily make it available to a small group of selected users before broadening its release. If the feature is not well received initially, it can be pulled back before it impacts the product's reputation; if customers like it, it is developed fully, incorporated in the product, and supported.

12.4.3.4 Beta Testing

A beta version is an "almost-ready-for-prime-time" version—one that is mostly complete but may still be missing features planned for the market-ready version. **Beta testing** is *real-world testing* of a beta version by a *wide range of customers* performing *real tasks*. Its purpose is to uncover bugs and issues, such as usability, scalability, and performance issues, before wide release.

Feedback and analytics from beta testing are used as inputs to fix remaining glitches and address user complaints before releasing the product or change to the market. Split testing may also be performed at this time—whereby one cohort of users is exposed to the beta version while a control group is not.

For more on split testing, see Chapter 7, section 7.11.5.2.

Beta testing is not just for MVPs; it should be a final testing step after internal alpha testing for all new features and major changes before they are widely released.

For more on beta testing, see Chapter 16, section 16.5.3.

12.4.4 MVP Types

When planning an MVP, the objective is to hack together a version of the product or feature that delivers the desired learning goals as quickly and inexpensively as possible. The following are strategies for achieving that. One MVP might incorporate any number of these strategies.

- Differentiator MVP
- Smoke-and-Mirrors MVP

- Walking Skeleton
- Value Stream Skeleton
- Concierge MVP
- Operational MVP
- Preorders MVP

These MVPs are described in the following sections.

12.4.4.1 Differentiator MVP

At the start of new product development, the most common strategy is to develop a low-cost version that focuses on the product's differentiators. This was the approach we saw taken earlier by Trint. Using existing components, the company was able to piece together an MVP demonstrating the differentiating features of its product (speech-to-text auto-transcription plus editing) and validating its value in just three months.

Another example is Google Docs, which began as Writely. Writely was an experiment by Sam Schillace to see what kind of editor could be created by combining AJAX's (JavaScript in the browser) content-editable functionality with word-processing technology.[5] Early versions focused on the product's key differentiators—its speed, convenience, and collaborative capabilities—while leaving out many other word-processing features, such as rich formatting and pagination. The hypothesis was that users would be excited enough about the differentiators to ignore the lack of richness in other areas. Interestingly, real-time collaboration on documents—which became a differentiating feature—was not seen as a primary one at the time; it was included because it seemed like the most natural way to solve the problem of documents worked on by multiple people.

The first version of the original product was pulled together quickly, using the browser for most of the editing capabilities and JavaScript to merge the local user's changes with those of other users. The client-side JavaScript amounted only to about ten pages of code.[6] Over time, the company added more word-processing features when it became apparent that they were essential to users and in order to open up new markets. Just one year after Writely was introduced, it was acquired by Google. Within the first month of its adoption, about 90 percent of Google was using it.

12.4.4.2 Smoke-and-Mirrors MVP (or Swivel Chair)

A **Smoke-and-Mirrors** MVP approach provides the user with an experience that is a close facsimile of the real thing but is, in fact, an illusion—like the one created by the magician pulling strings behind the curtain in the movie *The Wizard of Oz*.

5. Ellis Hamburger, "Google Docs Began as a Hacked-Together Experiment, Says Creator," *The Verge*, July 3, 2013, https://www.theverge.com/2013/7/3/4484000/sam-schillace-interview-google-docs-creator-box

6. Hamburger, "Google Docs."

One of my clients, a cable company, used this approach to provide an MVP frontend for customers to configure their own plans. The site operated in a sandbox, disconnected from operational systems. Behind the scenes, an internal support agent viewed the inputs and **swivel-chaired** to an existing internal system to process the request. The customer was unaware of the subterfuge. The MVP allowed the company to test the hypothesis that customers would *want* to customize their own plans before investing in developing the capability.

12.4.4.3 Walking Skeleton

A **Walking Skeleton**, or **spanning application**, validates *technical (architectural)* hypotheses by implementing a low-cost end-to-end scenario—a thin vertical slice that cuts through the architectural layers of the proposed solution. If the Walking Skeleton is successful, the business will invest in building the real product according to the proposed solution. If it is unsuccessful, the technical team goes back to the drawing board and pivots to a new technical hypothesis.

For example, in the Customer Engagement One (CEO) case study, the organization plans an end-to-end scenario for ingesting text messages from a social-network application, saving the messages using the proposed database solution, retrieving them, and viewing them as a list. Another example is Trint, whose first MVP incorporated the end-to-end scenario from speech to text to editing in order to validate the architectural design for the product.

12.4.4.4 Value Stream Skeleton

A **Value Stream Skeleton** implements a thin scenario that spans an operational value stream—an end-to-end workflow that ends with value delivery. It's similar to a technical Walking Skeleton except that it validates market instead of technical hypotheses. It covers an end-to-end *business* flow but does not necessarily use the proposed architectural solution.

The intuitive sequence for delivering features is according to the order in which they're used. For example, you might begin by delivering a feature to add new products to the product line for an online store and follow with features to receive inventory, place an order and fulfill an order. Not only does this sequence minimize dependency issues, but it also enables users to perform valuable work while waiting for the rest of the system to be delivered. I usually took this approach in my early programming days. The problem with it, though, is that it results in a long lag until an end customer receives value (e.g., a fulfilled order). In a business environment where there is a strong advantage in being fast to market, that kind of lag is unacceptable. Another problem is that it can delay the time until a company can begin receiving revenue from customers.

A Value-Stream Skeleton avoids these problems by delivering quick wins that implement thin versions of the end-to-end value stream, often with reduced functionality.

The first version of a Value-Stream Skeleton focuses on the value stream's endpoints—the *entry point* where the customer makes a request and the *endpoint* where the customer receives value. Workarounds are often used for the missing steps. For example, the first MVP for an online store allows a customer to purchase a few select products. The product descriptions and prices are hardcoded into the interface instead of being pulled from a database. This lowers development costs. The products are offered only in a single geographic region—simplifying the business rules and delivery mechanisms that the MVP implements. Despite the thinness of the MVP, it provides learning value to the business and real value to an end customer, who can already order and receive the products with this early version. As the business grows, the MVP evolves to handle more products and a broader geographical region.

12.4.4.5 Concierge MVP

The Concierge MVP[7] is based on the idea that it's better to build for the few than the many. Early versions are aimed at a small submarket that is *very* enthusiastic about the product, and the learning gained from the experience is used to scale the product. One example of a Concierge MVP is Food on the Table,[8] an Austin, Texas, company that began with a customer base of one parent. The company met with the parent once a week in a café to learn the parent's needs and take orders. The orders were filled manually. The process was repeated for a few other customers until the company learned enough to build the product.

As the example illustrates, you begin the Concierge MVP approach by selecting a single, real customer. The first customer can be found through market research, using analytics to determine the desired customer profile and inviting a customer who fits the profile to act as an MVP tester. Alternatively, you can select the first customer from among individuals who have previously indicated an interest in the product. This customer is given the "concierge treatment"—served by a high-ranking executive (e.g., vice president of product development) who works very closely with the customer, adding and adjusting features as more is learned.

At this stage, internal processes are often mostly manual. A company might spend a few weeks working with the first customer in this way, learning what that person does and does not want, and then select the next customer. The process is repeated until the necessary learning has been obtained and manual operations are no longer viable—at which point the product is built and deployed.

12.4.4.6 Operational MVP

An MVP isn't always created to validate software hypotheses and features; it can also be used to test operational hypotheses and changes. In a real-life example (which I'll keep anonymous to protect the company), a company created an MVP to test the impact of a price hike on sales. The MVP displayed the higher price to a select group of customers, but behind the scenes, the customers were still being charged the regular, lower price. Once the learning objective was achieved, customers received an email notifying them that they had been part of a test group and that no extra charges were actually applied.

7. Eric Ries, *The Lean Startup* (New York: Random House, 2011), 180.

8. Eric Ries in Lee Clifford and Julie Schlosse, "Testing Your Product the Lean Startup Way," *Inc.*, July 17, 2012, https://www.inc.com/lee-clifford-julie-schlosser/lean-startup-eric-ries-testing-your-product.html

12.4.4.7 Preorders MVP

The most reliable and cost-effective way to test a value hypothesis that customers will pay for an innovative product is to offer a means to order it before it's actually ready. The MVP can be something as simple as a promotional video or demonstration prototype. It may employ a stripped-down ordering process, such as order by email attachment, order by phone, or an online ordering site with hardcoded options. An MVP of this type might not require any stories—or it might need a few small stories (e.g., to set up a simple frontend for placing orders).

My own company, Noble Inc., used this approach when we were considering developing a product to provide a 360-degree evaluation of the business analysis practice in an organization. For the MVP, we developed a facsimile of the product and demonstrated it to our clients in an attempt to generate presales. What we learned was that there wasn't enough interest to justify building the real thing. Despite the failure of the test, I consider it money well spent. Imagine if we had learned it only after a large investment!

Dropbox's version of this MVP strategy played out much better. Dropbox posted a video of its product,[9] illustrating its main features. The video received enthusiastic and voluminous feedback from potential customers—making the case for the product and generating important suggestions about features and potential issues that were incorporated into the first marketed version.

12.4.5 MVP's Iterative Process

You don't just create an MVP and test it once. The MVP process is iterative. Its steps are as follows:

1. **Establish an MVP to test hypotheses.**

   Specify an MVP to test one or more leap of faith hypotheses (e.g., using any of the MVP types discussed in the prior section).

2. **Tune the engine.**

   Make incremental adjustments to fine-tune the product on the basis of feedback from customers as they use the product.

3. **Decision point: persevere or pivot.**

   After tuning for a while, decide whether to persevere with the business model or pivot to a different hypothesis.

12.4.6 The Pivot

A **pivot** is a switch to a different hypothesis based on a failure of the original premise. A company may decide to pivot near the start of a product's development due to the MVP process described previously. Alternatively, the pivot may occur at any time in a product's life if it becomes apparent there is no market for the product, and the product should be

9. Drew Houston, "Dropbox Original MVP Explainer Video," 2007, https://www.youtube.com/watch?time_continue=12&v=iAnJjXriIcw

reoriented toward a new market or usage.[10] An example of a pivot to an established product is Ryanair, once Europe's largest airline (based on passenger numbers).[11] Back in 1987, when the company realized it was failing financially, it pivoted to a low-end, disruptive revenue model based on the hypothesis that customers would be willing to pay for meals and other perks in return for cheap fares. The hypothesis was borne out when customers flocked to the airline.[12] More recently, in response to Brexit, the company has again pivoted—this time away from the United Kingdom to a business model based on growth outside of it.[13]

12.4.6.1 Constructive Failures

A pivot represents a failed premise, but, as the Ryanair example shows, the failure can often be constructive. In fact, many of today's successful companies are a result of such failures. For example, Flickr resulted from the failure of a previous offering—Game Neverending.[14] When the original product failed, the company pivoted by turning it into a successful photo-sharing app, leveraging the lessons it had learned about the value of community and the social features it had developed for the game (such as tagging and sharing). Groupon is another example. Conceived initially as an idealistic platform for social change, it then pivoted to become a platform for those seeking a bargain.

12.4.7 Incrementally Scaling the MVP

An effective way to develop a product is to start with a manual MVP and automate and scale it incrementally as the product grows. This approach was used by Zappos, an online shoe store.

Here's how the process played out, as described by the company's founder: "My Dad told me . . . I think the one you should focus on is the shoe thing. . . . So, I said okay, . . . went to a couple of stores, took some pictures of the shoes, made a website, put them up and told the shoe store, if I sell anything, I'll come here and pay full price. They said okay, knock yourself out. So, I did that, made a couple of sales."[15] In 1999, the company

10. Clif Gilley, "Do You Have to Build an MVP to Pivot?" [blog post], Quora, December 16, 2013, https://www.quora.com/Do-you-have-to-build-a-MVP-to-pivot

11. Thanks to my editor, Ron Healy, for informing me of this example.

12. Geoff Daigle, "Case Studies from Amazon, Yahoo, and Ryanair Reveal How Growth Teams Should Use Data + Feedback," Thinkgrowth.org, August 21, 2017, https://thinkgrowth.org/case-studies-from-amazon-yahoo-and-ryanair-reveal-how-growth-teams-should-use-data-feedback-d7b410a005f8

13. Alistair Smout and Kate Holton, "UPDATE 2—As Brexit Bites, Ryanair to Pivot Growth Away from UK for Next 2 Years," Reuters, April 6, 2017, https://www.reuters.com/article/britain-eu-ryanair-hldgs/update-2-as-brexit-bites-ryanair-to-pivot-growth-away-from-uk-for-next-2-years-idUSL5N1HE1YQ

14. Reid Hoffman, "The Big Pivot—with Slack's Stewart Butterfield," *Masters of Scale with Reid Hoffman* [podcast], November 14, 2017. https://player.fm/series/masters-of-scale-with-reid-hoffman/the-big-pivot-wslacks-stewart-butterfield

15. Jay Yarow, "The Zappos Founder Just Told Us All Kinds of Crazy Stories—Here's the Surprisingly Candid Interview," *Business Insider*, November 28, 2011, https://www.businessinsider.com/nick-swinmurn-zappos-rnkd-2011-11?op=1

signed on a dozen brands—all men's brown comfort shoes. As they added more respected brands, such as Doc Martens, the company and market grew and, in tandem, Zappos automated and scaled its business systems and processes.

12.4.8 Using MVPs to Establish the MMP

Using the MVP process, a company can quickly and inexpensively validate through experimentation which features will make the most difference. These features are referred to as the minimal marketable features (MMFs). An MMF is the smallest version of a feature (the least functionality) that would be viewed as valuable by customers if released to the market. MMFs may deliver value in various ways, such as through competitive differentiation, revenue generation, or cost savings. Collectively, the MMFs define the minimum marketable product (MMP)—the "product with the smallest feature set that still addresses the user needs and creates the right user experience."[16]

BLINK CASE STUDY PART 20

Create an MVP

Background

You convene stakeholders and developers to specify the BLInK hypotheses that will be tested during the first quarter and plan the MVPs that will be used to validate them.

The Ask

The deliverables of the workshop will be

- **Deliverable 1:** Hypothesis—Leap of faith hypothesis (or hypotheses) critical to the business case for the product
- **Deliverable 2**: MVP—High-level description of the MVP that will be used to test the hypothesis.

Inputs

- Chapter 7, Case Study Part 8, Deliverable 1: Assumptions Checklist

What Transpires

The group discusses assumptions that are most critical to the product's business case. They agree that the most urgent leap of faith hypothesis is that reluctance to sharing data can be overcome when a benefit is shown immediately (A7). Business stakeholders and developers brainstorm ways to test the hypothesis quickly and inexpensively.

16. Roman Pichler, "The Minimum Viable Product and the Minimum Marketable Product," October 9, 2013, https://www.romanpichler.com/blog/minimum-viable-product-and-minimal-marketable-product

scaled iteration retrospective. The chapter provides guidelines for selecting software tools to support collaboration among teams. It also offers lightweight solutions, such as using roamers and scouts.

The chapter concludes with guidance for addressing potential problems and challenges when scaling an agile organization, such as coordinating with waterfall teams.

17.1 Objectives

This chapter will help you

- Understand how DevOps, CI, CD, and ATDD enable frequent, reliable delivery of value to the end user.
- Understand how to structure a scaled development organization into portfolios, programs, product areas, feature teams, and component teams.
- Know when to use timeboxed and when to use flow-based planning approaches.
- Conduct scaled agile events, such as scaled quarterly and iteration planning meetings.
- Conduct rolling analysis (feature and story preparation) on a scaled agile initiative.

17.2 This Chapter on the Map

As indicated in Figure 17.1, the chapter focuses on the Grand Lane of the planning and analysis map, cutting across all activity zones from Initiation and Planning to Quarterly Closeout.

17.3 Why Do We Need a *Scaled* Agile Approach?

It's common, in agile circles, to hear that a scaled agile organization should be composed of self-sufficient, independent teams.[1,2] If agile teams were, in fact, totally independent at scale, there would be no need for scaled agile frameworks (or this chapter); you would simply follow team-level agile practices and multiply them across the organization without any additional processes or roles. (As we'll see, this is roughly the approach of

1. For example, the Scrum Guide declares that "members have all the skills necessary to create value each Sprint" and are "self-managing." Ken Schwaber and Jeff Sutherland, "The Scrum Team," in *The Scrum Guide: The Definitive Guide to Scrum—The Rules of the Game*, 2020, 5, https://www.scrumguides.org

2. As another example, Ron Jeffries writes, "Much of the work of any company can be done by single cross-functional teams." See Ron Jeffries, "Issues with SAFe," April 2, 2014, http://ronjeffries.com/xprog/articles/issues-with-safe

the Large Scale Scrum [LeSS] framework.) Yet, in practice, dependencies among teams are the norm, not the exception, in scaled agile organizations. These persistent dependencies aren't a bug. They're a feature of a well-scaled organization, and it is neither possible nor *desirable* to eliminate them. Because agile teams in scaled organizations are interdependent—not independent—we need effective solutions for coordinating and integrating their work at scale.

First, we examine why teams are interdependent in a scaled agile organization. Then, we look at the following strategies for addressing that interdependence:

- **Planning:** Choosing an agile planning approach that supports inter-team collaboration
- **Continuous Delivery:** Integrating, testing, and delivering software continuously, safely, and sustainably at scale (DevOps/CI/CD)
- **Scaled Agile Culture:** Creating a culture that supports innovation at scale
- **Scaling the Backlog:** How to structure the product backlog in a scaled agile environment
- **Scaling the Organization:** How to structure a scaled agile organization
- **Scaling the Process:** Scaling the agile process to promote collaboration across teams
- **Scaling Tools:** Tools and techniques for supporting scaled agile development and team coordination
- **Potential Issues in Scaling Agility:** How to address challenges scaling agility, such as non-colocated teams and coordination with waterfall developers

17.3.1 Why Scaled Agile Teams Are Interdependent

Scaled agile teams tend to be dependent on each other because of the interconnectedness of a product's features, technical complexity, and shared components. Let's explore these issues.

17.3.1.1 Interconnected Features

Consider a mobile phone and the subproducts—or high-level features—it encompasses, such as a camera, photo-editing, messaging, and social-network capabilities. In a scaled agile organization, each of these subproducts is maintained by a feature team or team of teams.

The user can use each subproduct on its own, but the product's full value lies in how all its subproducts work with *each other*. For example, customers can access photo-editing and messaging directly from the camera—enabling them to shoot, edit, and send images seamlessly. Because subproducts are *designed* to work together this way, rather than as standalones, they will inevitably have dependencies on each other—and so will the teams that develop and maintain them.

The same applies when the product is not a physical object but a software system. Consider Z-News, a fictional, digital news service. Z-News's teams are organized by business areas (e.g., an order-processing team, a service-delivery team, a billings team). Now suppose that stakeholders have requested a new subscription service to deliver personalized news hourly to readers. This single request will require numerous teams working in concert with each other. The order-processing team will add the capability to order the new subscription, the service-delivery team will implement the delivery of customized news each hour, and the billings team will implement the monthly subscription charges for the new service. Across the value stream—from the subscription order to service delivery—each team relies on data produced by other teams. For example, the order-processing team captures subscription details, such as topics and sources, and the service-delivery team uses that information to determine what news items to deliver. Because the teams are interdependent, they need to coordinate their plans at the frontend of the development cycle, collaborate throughout development, and integrate and test their work continually as stories are done. *How* they do that effectively is the subject of this chapter.

17.3.2 Product Complexity

Another reason for team dependencies is that the competencies required to implement a feature for a complex product are usually too numerous to be accommodated in a small agile team of no more than ten members. Expertise is typically needed in UI design and coding, cloud services, the deployment framework, automated testing, the application stack, the software stack (infrastructure), open-source tools, database management, and business domain knowledge. Since a small team usually can't cover all these competencies, the competencies are typically distributed among a group of interdependent teams.

17.3.3 Shared Components

Another reason that team dependencies can't, and shouldn't, be eliminated is that multiple teams often share software components and are dependent on the team that manages them. As we'll explore later in this chapter, if we let feature teams change a component as they see fit, the result will be inconsistency in design and quality across the component. To ensure this doesn't happen, a **component team** takes primary responsibility for it. However, component teams introduce dependencies—because if a feature team requires a change to a component, it's dependent on the component team to implement it. Similarly, if the component team changes a component, the feature teams that depend on it are potentially impacted.

17.4 Planning: Choosing an Approach That Supports Inter-team Collaboration

There are two necessary but distinct coordination issues to address in a scaled organization: What approach will the organization use to plan work across multiple teams, and how will it time the integration and delivery of software across multiple teams? In answering those questions, it's essential to realize that the solutions to the two problems are not necessarily the same. In fact, it's usually best to use a mixed approach—a timeboxed or hybrid approach to plan large features at the frontend and a flow-based approach at the back end to continuously implement, integrate, and deliver improvements to the customer. We addressed the issue of flow-based versus timeboxed approaches earlier in this book. Let's revisit it now with a focus on scaled agile organizations.

17.4.1 Review of the Two Approaches

In a **flow-based** approach, each work item moves from step to step in the development lifecycle at its own pace, provided that work-in-progress (WIP) limits at each step are not exceeded. The aim is to achieve a continuous flow of each item without bottlenecks—from initiation through delivery. This is the approach used by the Kanban framework.

In contrast, with **timeboxed** planning, teams commit to *all* of the work items for a specified period (the timebox) at the start of the period. Two common timeboxes are the quarter and the iteration. A quarter refers to three months, but (as noted elsewhere) I use the term in this book as a shorthand for a release cycle, a SAFe program increment (PI), or any period of two to six months. An **iteration** is a shorter timebox, typically one or two weeks. Frameworks that incorporate iterations include Scrum, Extreme Programming (XP), LeSS, and SAFe. In Scrum, this period is referred to as a *sprint*. The maximum duration of a sprint is one month.

17.4.2 Which Approach Should You Use at the Frontend?

As a general guideline, feature teams benefit most from a mixed planning approach at the frontend, using flow-based (Kanban-style) planning for customer-driven features and quarterly (timeboxed) planning for large, strategic initiatives.

17.4.2.1 When to Use a Flow-Based Approach to Accept Requirements into Development

The flow-based portion of the budget enables teams to respond quickly to learning, rather than waiting a quarter or more to apply newly gained knowledge. This part of the budget should be set aside for small efforts that can be handled by a single team with minimal help from others. For example, the team might be exploring options to improve the conversion rate of browsers to subscribers or looking at different ways for a user to filter or sort content. To do so, they try out different options with customers and adapt them based on customer feedback. Since customers' responses drive each inspect-and-adapt cycle, there is no sense in trying to predict and prioritize their preferences too far in advance. Consequently, a flow-based approach is advised.

17.4.2.2 The Pitfalls of Relying Solely on a Flow-Based Approach

However, many organizations with which I work have discovered that when they rely *solely* on flow-based planning, the product becomes fractured because the approach

17.8 Scaling the Agile Organization

As noted earlier in this chapter, an organization developing a complex product will inevitably require multiple interdependent teams in order to cover all the necessary competencies for all of its subproducts and components. For example, the top-level product for a large company might easily include more than twenty subproducts. Each of these, in turn, might be delivered over multiple channels (e.g., Web, mobile), each of which requires specialized technical competencies. For a company such as SAP (a vendor of enterprise resource planning software), this can require, in total, more than two thousand agile teams.[23] In this section, we explore how to structure agile organizations of that size.

17.8.1 Scaling by Subproduct and Product Area: MyChatBot Case Study

The solution is to structure the organization by subproducts, also known as product areas. Let's look at a fictional example, MyChatBot. MyChatBot is an innovative company and product based on the hypothesis that customers will want to use chatbots for common customer-engagement tasks in order to increase sales and customer outreach at minimal cost. The company has identified ten primary high-level tasks customers would use MyChatBot for, including Sales, Marketing, Customer Support and Engagement, Analytics. In circumstance-based market segmentation, these are identified as the **jobs** customers hire the product to do.

See Chapter 8, section 8.4, for more on circumstance-based market segmentation.

Figure 17.4 depicts how the MyChatBot organization is structured into levels of subproducts. For illustration purposes, I've included only four of its subproducts.

As indicated in Figure 17.4, MyChatBot is the top-level product. Below are its subproducts—one for each primary usage of the product. Four of these usages are highlighted: Sales, Marketing, Customer Support and Engagement, and Analytics.

Each of these subproducts has numerous sub-subproducts, referred to as *product areas*. For example, the Customer Support and Engagement subproduct includes a product area for each of the following sub-subproducts:

- **Collaboration Tool Automation:** To facilitate the collaboration of support staff
- **Ingest Content:** To load Chatbot messages originating on social media and elsewhere
- **User Efficiency:** To optimize the efficiency of customer-support users

Each product area is divided up into **feature sets—**groups of related product features. For example, Collaboration Tool Automation has one team for each of the following feature sets: tagging, triaging, and assigning messages using automation. In a larger organization, there might be multiple teams devoted to each feature set.

23. Darrell K. Rigby, Jeff Sutherland, and Andy Noble, "Change Management: Agile at Scale," *Harvard Business Review* (May–June 2018), https://hbr.org/2018/05/agile-at-scale

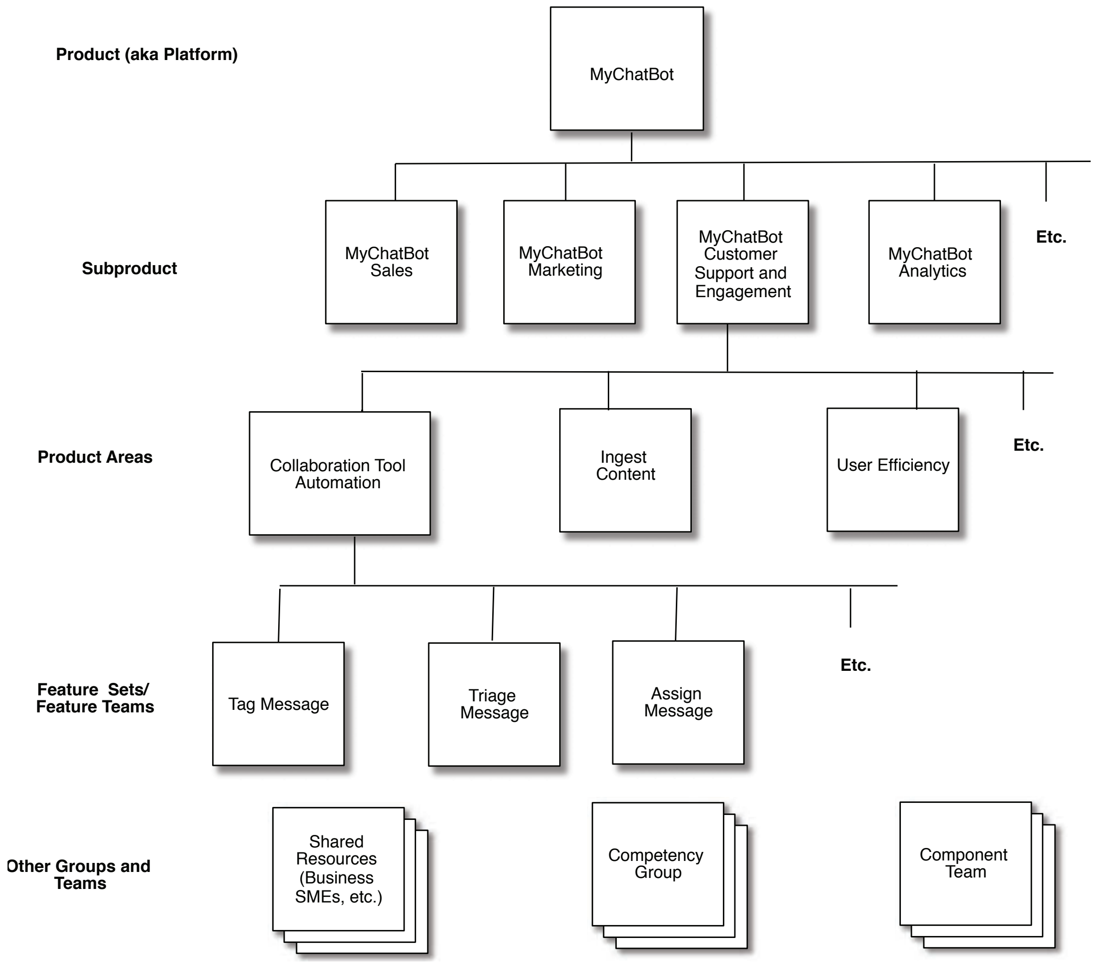


Figure 17.4 *MyChatBot organization*

In addition to the feature teams, Figure 17.4 indicates component teams dedicated to commonly used components. For example, MyChatBot might have a component team dedicated to an API that manages outgoing messages to third-party products, such as social networks. Figure 17.4 also indicates **competency groups**—associations that supply the teams with members, shared resources, and support within a particular area of expertise, such as UX design.

17.8.2 Scaling the PO Role

As mentioned earlier in this chapter, high-performing organizations require leadership at every level. A product-level PO is responsible for the whole product, while area POs are assigned at all the intermediate subproduct levels down to the individual team. Each of these teams is led by a team PO or proxy PO. We've discussed the product-level PO. Let's examine the other roles.

17.8.2.1 Area POs

An **area PO** should be assigned to each subproduct or sub-subproduct down to the level above the team level. (At the team level, a team PO or proxy PO is assigned, as described shortly.) Each area PO is responsible for a subproduct—a high-level use case, or job, customers hire the product to do. The role may be filled by a portfolio manager, program manager, product manager, or SAFe Release Train Engineer (RTE). Area POs have ultimate responsibility for prioritization decisions in their area—though (as noted earlier) other stakeholders are typically required for signoffs and approvals, and local decision-making should be devolved to lower-level POs. An area PO may also act as a PO for one of the lower levels.

17.8.2.2 Team POs

Each team is led by a team PO or proxy PO (described in the next section). The PO's outward-facing activities include speaking with business executives to understand strategic objectives, interacting with salespeople and customers, attending trade shows, conducting surveys to understand the market, and talking to data analysts to understand how people are using the product. Inward-facing duties involve close day-to-day interactions with the team—requiring about ten hours or more per week.

The full complement of PO-related responsibilities is often too excessive for a single person, so the work is often distributed among roles. If there is a team-level PO, the team PO focuses on outward-facing activities, while the team analyst focuses on inward-facing responsibilities. If the team is led by a proxy PO, the area PO focuses outward, and the proxy PO takes on inward-facing tasks.

17.8.2.3 Proxy PO and Business Analyst

It's hard enough for a PO to find sufficient time to work day-to-day with *one* team while fulfilling external-facing responsibilities. In practice, a PO is often required to support *more* than one team because of a scarcity of resources. An effective solution in this case is to use a **proxy PO** or business analyst at the team level to take on some of the PO's responsibilities. The proxy PO or business analyst works full time with the team to answer detailed questions about the requirements and communicate higher-level goals to the team so that the PO can focus on external responsibilities.

Formally, this can play out in several ways. An area PO may be assigned to preside over a group of teams, with proxy POs at the team level. Alternatively, a team-level PO may be shared by a few teams, with team analysts taking on inward-facing responsibilities at the team level.

17.8.3 Portfolio and Program Structure

Another way to structure a scaled organization is by portfolios and programs. This structure is especially well-suited to initiatives that span departments or entire products. Figure 17.5 depicts the organizational structure for XComm, a fictional company loosely based on a real telecommunications company.

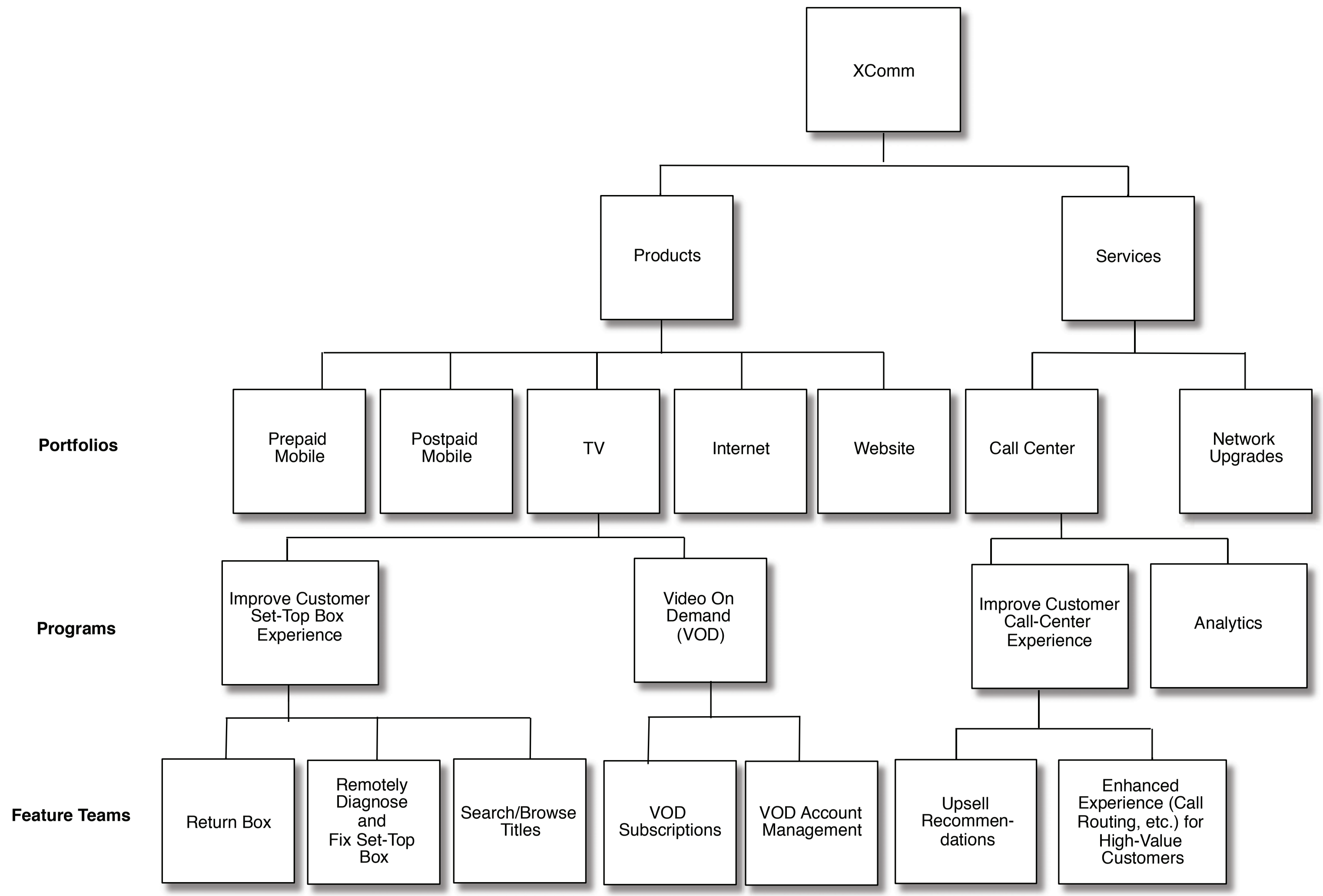


Figure 17.5 *Portfolio and program organizational structure*

As depicted in Figure 17.5, the organization is divided into products and services. The products division focuses on initiatives to improve the products XComm sells to its customers (e.g., mobile and Internet products). The services side focuses on quality improvements to its support services (e.g., call center improvements and network upgrades).

17.8.3.1 Portfolio Level

A **portfolio** is a broad initiative that may span departments, business areas, products, and systems. Figure 17.5 indicates that the products division contains prepaid mobile, TV, and Internet portfolios, each representing a line of business.

The portfolio is the largest organizing unit in SAFe, responsible for strategy and investment. Lean portfolio management (LPM) practices should be used. The focus of LPM is on providing resources to long-lived teams of teams[24] so that they can realize strategic objectives and achieve desired outcomes. This contrasts with the traditional practice of funding one-time projects with specified outputs. LPM includes the lean startup practices covered in this book, such as MVP, pivot or persevere, lean techniques for eliminating waste, and cultural practices such as servant-leadership (discussed in this chapter).

Portfolio Epics

In SAFe, long-lived initiatives at the portfolio level are classified as **portfolio epics**. A portfolio epic can span multiple teams of teams—referred to in SAFe as *Agile Release Trains* (ARTs). The following format may be used to specify the hypothesis statement for a portfolio epic:[25]

e.g.

Epic description: For [customers] who [perform some activity], the [solution] is a [what] that [delivers this value]. Unlike [competition/existing solution or non-existing solution], our solution [does something better].

Business outcomes (measurable benefits)

- <benefit 1>
- <benefit 2>

Leading indicators

- <indicator 1>
- <indicator 2>

24. John May, "Lean Portfolio Management: How to Build a Better Enterprise by Being More Lean," Atlassian, https://www.atlassian.com/agile/agile-at-scale/lean-portfolio-management

25. Richard Kastner and Dean Leffingwell, *SAFe 5.0 Distilled: Achieving Business Agility with the Scaled Agile Framework* (Boston: Addison-Wesley, 2020), 154.

2. **Identify the opportunities:** Ask customers what needs aren't being met well today. What services and products are too costly, too inconvenient, or too inaccessible? What difficulties are customers experiencing *that they don't even think of as problems* because there is currently no alternative? Which of these problems can the company solve through innovation?
3. **Separate customers by needs:** As Theodore Levitt, a professor at Harvard Business School, has said, "People don't want to buy a quarter-inch drill. They want a quarter-inch hole."[13] In other words, it's not the tool or product that counts to the customer; it's the outcome. Divide customers into groups by their needs (also referred to as jobs)—a problem they want to solve or a need they have that is not being met—not by demographics, product, or market size. Then seek to understand and address the needs of each group.
4. **Determine the vision:** Articulate the vision for the product or improvement. If it's a disruptive innovation, specify a vision for performing a job in a way that meets or outperforms expectations in the target group's critical areas of concern (e.g., cost, convenience)—even though initial versions might underperform in areas they care less about.
5. **Identify the leap of faith hypotheses:** Identify the leap of faith hypotheses that must be true for the business model to be successful.
6. **Conduct MVP testing:** Test the leap of faith hypotheses through rounds of MVP experiments with real customers, making adjustments based on feedback and metrics. Use leading indicators to forecast the likely outcome.
7. **Pivot or persevere:** Use the results of MVP testing to identify the Minimum Marketable Product (MMP)—the smallest version of the product that would be viable in the market. Use feedback from MVP testing to determine whether to commit to the vision or make a radical change in direction.
8. **Continuously improve:** Use the results of MVP testing to identify the Minimum Marketable Product (MMP)—the smallest version of the product that would be viable in the market. Use an iterative process to implement the MMP and continuously improve the product. Use data, frequent feedback, and MVP testing to inform decisions.
9. **Accelerate:** If the innovation is disruptive, accelerate rapidly to capture the market before incumbents and competition can respond.

18.6 Agile Corporate Culture

Successful innovation is not just about having a good idea—or even the right processes. It's about culture. Everyone involved in developing a product deemed "innovative" in

13. As quoted in Christensen and Raynor, *The Innovator's Solution: Creating and Sustaining Successful Growth*, chapter 3.

their industries—especially if it's a disruptive innovation—must share an organizational culture that embraces, supports, and encourages innovation. Failing to do so can result in disappointing failure.

Let's begin by defining corporate culture; then we'll look at what it means for that culture to be agile.

18.6.1 Definition of Corporate Culture

Culture is the sum total of beliefs and ideas that guide behavior. Adam Grant defines it as "repeated patterns of behavior that reveal norms and values."[14] Perhaps the most succinct way to explain culture is that it's *"what people do when no one's watching."*[15]

Corporate culture is "the beliefs and ideas that a company has and the way in which they affect how it does business and how its employees behave."[16]

18.6.2 Definition of Agile Corporate Culture

An **agile corporate culture** is a set of behaviors and ideas that guide an organization and its employees *in ways that optimize the organization's ability to anticipate and respond to change*. Agile cultures embed collaboration, empowered decision-making, and cognitive empathy in the organization—elements we explore further in this chapter.

Jeremy Gutsche defines the following prerequisites for an innovative culture:

- **Urgency:** A necessary condition for reinvention and innovation is that people have a sense of urgency about the need for change.
- **Perspective:** When the organization's perspective is based on past accomplishments, the result can be complacency and a loss of urgency. An agile organization's perspective is not focused on the past or exclusively on the present; it's oriented toward future needs and trends.
- **Experimental Failure:** The enterprise must value and nurture a culture of experimentation. People should *expect* failure to occur—as a natural and necessary part of innovation.
- **Customer Obsession:** The company must be obsessed with understanding its customers and creating an emotional, cultural connection with them.
- **Intentional Destruction:** The organization understands that existing hierarchies must be destroyed as a necessary precondition for reinvention, and it supports that process.

14. Adam Grant, "The Science of Leadership" [podcast], *Stay Tuned with Preet*, December 27, 2018.

15. Grant, "The Science of Leadership."

16. "Corporate culture," *Cambridge Dictionary,* http://dictionary.cambridge.org/dictionary/english/corporate-culture

These elements underlie the guidance in the following sections. For more on Jeremy's model, I urge readers to explore *The Innovation Handbook*[17] and *Exploiting Chaos*.[18]

18.7 Overview of Principles and Practices for an Agile Corporate Culture

Many existing agile and agile-adjacent frameworks and practices touch on agile corporate culture, even if they don't always call it out in those terms. These include lean thinking, Six Sigma, lean startup, the GE Beliefs,[19] DevOps, the Agile Manifesto, as well as lessons learned from transitioning companies.[20] The following synthesizes this guidance into a set of principles and practices for an agile culture.

The three principles for applying agile practices are as follows:

- Tailor the approach to the circumstance.
- Protect islands of innovation.
- Invest aggressively in enterprise agility.

The thirteen practices for an agile corporate culture are as follows:

- Iterative experimentation (fail fast)
- Embrace change
- Acceleration
- Empathy
- Responsible procrastination
- Distributed authority
- Let those who do the work estimate the effort
- Collaboration

17. Jeremy Gutsche, *Create the Future + the Innovation Handbook: Tactics for Disruptive Thinking* (New York: Fast Company, 2020).

18. Jeremy Gutsche, *Exploiting Chaos: 150 Ways to Spark Innovation in Times of Change* (New York: Gotham Books, 2009). Available as an ebook at http://cdn.trendhunterstatic.com/EXPLOITING-CHAOS-by-Jeremy-Gutsche-TrendHunter.pdf

19. Jeffrey Immelt, "Letter to Shareholders," in *GE 2014 Annual Report*, 10–11, https://www.annualreports.com/HostedData/AnnualReportArchive/g/NYSE_GE_2014.pdf

20. See, for example, Steve Blank, "Corporate Acquisitions of Startups—Why Do They Fail?" *Forbes*, April 22, 2014, https://www.forbes.com/sites/steveblank/2014/04/22/corporate-acquisitions-of-startups-why-do-they-fail. Also see Peter Nowak, "Video Streaming in Canada," September 27, 2016, http://www.alphabeatic.com/video-streaming

- Commit to outcomes, not outputs
- Transparency
- Bust silos
- Data-informed innovation
- Monitor adjacent and low-end markets

18.8 Three Principles for Applying Agile Practices

Let's begin with the principles for applying the practices.

18.8.1 Tailor the Approach to the Circumstance

The core meaning of agility is *adaptability*, and nowhere is this attribute more apt than for the agile approach itself. As in *Fight Club* (the novel and film by the same name), the first rule of an agile corporate culture is that *there is no agile culture*—or no *single* one for all situations. The agile practices that an organization adopts must be tailored to fit the mission of the enterprise and the values that matter most to it—and those practices should evolve as the mission changes over time. For example, Apple's original mission was "To make a contribution to the world by making tools for the mind that advance humankind."[21] A corporate culture tailored to this mission would embrace most, if not all, of the agile practices discussed in this chapter, such as fail fast. In contrast, Apple's mission today is the more prosaic and product-focused statement that "Apple designs Macs, the best personal computers in the world, along with OS X, iLife, iWork, and professional software. Apple leads the digital music revolution with its iPods and iTunes online store. Apple has reinvented the mobile phone with its revolutionary iPhone and App Store, and is defining the future of mobile media and computing devices with iPad."[22] A corporate culture aligned with the new mission's emphasis on past and current products and successes would lean more toward predictability and reliability and less toward experimentation and transformational change than one aligned with the first. It's not a question of what's right—but what's right *for the organization at that time*.

Culture can also vary *within* an organization. Suppose an established enterprise has created a new business unit to develop an innovative service. Even while the rest of the enterprise adopts a culture that supports predictability, the new business unit would be wise to adopt a highly agile culture that promotes learning and embraces change due to the novelty of the product.

21. "How Apple's Current Mission Differs from Steve Jobs' Ideals," Investopedia, June 22, 2019, https://www.investopedia.com/ask/answers/042315/what-apples-current-mission-statement-and-how-does-it-differ-steve-jobs-original-ideals.asp

22. "How Apple's Current Mission Differs."

Index

A

A/B (split) testing
- actionable metrics and, 187
- staging the release, 539
- value validation with, 491–494

AC. *see* Acceptance criteria (AC)

Acceleration
- agile culture embraces, 653–655
- innovative development and, 632
- sustaining, 110

Acceptance, of risk, 346

Acceptance criteria (AC)
- committing to feature's, 343
- decision tables analyzing, 433–440
- feature change initiatives and, 254
- feature documentation by specifying, 497
- feature preparation and, 216–217, 256
- for functional spikes, 417
- refining incrementally, 327–328
- as requirements-related term, 40
- rules of thumb for, 683
- specifying features and their, 259–263

Acceptance criteria (AC), story
- avoid too many, 411, 429–430
- confirmation of, 398, 407
- examples of, 397, 408
- extensiveness of, 411
- as specification by example, 409–410
- specification of, 407–414
- testing, 8
- of well-formed, 411–412
- when to create and update, 409
- writers of, 408–409

Acceptance test-driven development (ATDD)
- agile analysis vs. waterfall, 66
- and BDD, 563–564
- defined, 56
- history of, 18
- preventing last-minute integration issues, 9
- specifying feature AC, 259, 261

Acceptance testing. *see* User acceptance testing (UAT)

Accountability, vision with, 565

Accuracy
- of estimates, 335
- of risk forecasts, 546–547
- of transparency, 667

Actionable metrics, 187–188

Actions
- against developer tasks/stories, 471
- eliciting business rules with decision tables, 437, 438
- as journey map component, 277, 282

Activities
- BPMN private process model, 293, 295
- BPMN public process model, 288
- postponing until last responsible moment (LRM), 659
- scaled, 583–585
- story map backbone, 374

Activity card, story map, 368–369

Actor card, story map, 368–369

Actors, constructing story map backbone, 374–375

Adaptability
- balancing scope commitment and, 343
- of business analysts, 65
- as core meaning of agility, 635
- of high value in agile analysis, 14

Additional resources and checklists. *see* Resources and checklists, additional

Agenda. *see* Topics/agenda

Agile Alliance, 18

Agile analysis
- key practices versus waterfall, 65–68
- parallel histories of BA and, 16–17
- rules of thumb, 68

Agile analysis and planning
- art of. *see* Art of Agile analysis and planning
- definition, 13
- fundamentals. *see* Fundamentals of agile analysis and planning

Agile Analysis and Planning Map
- activities across development life cycle, 70–71
- enterprise agility, 624–626

Grand Lane, 79–81
introduction to, 72
iteration and story planning, 442–443
lanes, 73–74
Long Lane, 79
long-term agile planning, 222–223
MVPs and story maps, 354–355
overview of, 72–74
preparing the organization, 84–85
preparing the process, 115
quarterly and feature planning, 316–317
quarterly and feature preparation, 252–254
releasing product, 528–529
seeding product backlog, 194–196
Short Lane, 74–78
story in three Acts, 74
story preparation, 392–393
summary of, 81
visioning activities, 147–149
zones, 72–73
Agile analysis and planning, value proposition
agile development history, 17–18
agile diagnosis, 22
agile track record, 22–23
business analysis diagnosis, 18–19
business analysis history, 16–17
business analysis track record, 19–21
defining business analyst, 14–15
mapping to IIBA and PMI guides, 678
objectives, 13
reasons for using, 15–16
summary of, 25
two diagnoses for same problem, 18
understanding, 13–14
why agile should include BA competency, 24–25
Agile corporate culture. *see* Corporate culture
Agile enterprise transition team, 112
Agile Extension to the BABOK Guide, 18
Agile financial planning. *see* Financial planning
The Agile Manifesto, 17–18
Agile Manifesto, 28–31
The Agile Practice Guide, 31
Agile Practice Guide, 18
Agile Release Trains (ARTs)
job-based organization structure, 668–669
quarterly planning event and, 318
Scrum of Scrums meetings for, 600
teams in SAFe organized into, 57, 582
Agile requirements management tools, 699–700
Agile Software Development with Scrum (Schwaber and Beedle), 7
Airbnb, iterative experimentation, 650
Align cycles, waterfall teams, 619
Alpha testing
beta testing after internal, 359
pre-alpha stage, 533
product release, 533
Analysis and Planning activities across development life cycle
map. *see* Agile Analysis and Planning Map, understanding
mapping to IIBA and PMI guides, 678
objectives, 69
rules of thumb, 682–683
summary of, 81
Analysis documentation, product release, 538
Analyze-code-build-test cycle, 490–494
Anonymity, Delphi estimation, 338
Appendix A. *see* Resources and checklists, additional
Appendix B. *see* Discovery-driven planning, BestBots case study
Apple iPhone, as disruptive innovation, 640
Approval, agenda plan, 348, 351
Architect, product owner council, 580
Architectural runway, preparing for scaling, 585
definition, 510
Architecture
collaboration-enhancing, 664
cost-benefit calculation, 120–121
scaled feature preparation, 602–603
Architecture (block) diagrams, 310–312
Architecture review
architecture (block) diagrams, 310–312
context diagrams, 307–308
data flow diagrams (DFDs), 308–310
UML communication diagrams, 308
Area POs, defined, 572
Art of Agile analysis and planning
Cantankerous Customer story, 10–11
example of, 1–5
It's Not My Problem story, 8–9
for mainstream businesses, 5–7
mapping to IIBA and PMI guides, 678
objectives, 1
summary of, 11
Artifacts
iteration review for forecasting/tracking, 516–517
tracing analysis, 506–508
updating BA documentation, 496
in use-case model, 499–502
ARTs. *see* Agile Release Trains (ARTs)
Asana, requirements management tool, 699
Aspirational, vision statement as, 173
Assumptions
checklist, 708–709
defined, 36–37

interim goals to test, 383
leap of faith hypotheses as, 189–191
reviewing in planning agenda, 329
specifying for interim periods, 243
validating using MVPs, 228–230
Assumptions/hypotheses card, story map, 368–369
ATDD. *see* Acceptance test-driven development (ATDD)
Attendees
daily standup, 473
iteration planning, 445
iteration retrospective, 518
iteration review, 514
pivot-or-persevere meetings, 545
product backlog refinement, 512
quarterly planning, 692
quarterly/feature planning event, 326
scaled quarterly and feature planning, 587, 698
Attractive feature, Kano grades, 206–207
Attributes
defining PBI, 125–126
values for story, 404
Automation
computers for repetitive work in DevOps, 561
continuous testing in DevOps, 562
DevOps provisioning, 562
MVP case study for, 357–358
in test-build-deploy, 558–559
testing without, 90–91
timing build and distribution process, 93
transitioning from manual testing to, 91–93
value stream mapping for process, 284
Autonomy
benefits of, 660
for branding disruptive products, 646–647
extending to innovative business units, 645–646
for internal business units, 647
structuring new business as legally separate entity, 647
Availability
defined, 35
determining initial capacity, 136
forecasting iteration capacity, 448–449
as quality of transparency, 667
AWS (Cloud) competency group, 578

B

BABOK v3: A Guide to the Business Analysis Body of Knowledge (BABOK Guide)[IIBA]
BA information artifacts and events, 122
BA practices, 17
BA standards, 31
business rules, 37
context analysis, 263–264
knowledge areas, 31–32
mapping of book chapters to, 677–681
requirement types checklist, 123
Backbone, constructing story map
activities, 374
actors, 374–375
case study, 374–379
inputs, 372–373
narrative, 372
overview of, 370
scope, 373–374
user tasks, 374
Backend systems, cost-benefit calculation, 120
Backlog, iteration
defined, 446
forecast goal and scope, 447–451
forecasting stories that will be delivered, 450–451
Backlog, quarterly, 327
Backlog refinement
as preparation in this book. *see* Preparation
rules of thumb, 683
Scrum, 47
Backward (upward) traceability, 130, 506
Bad news, safe spaces for, 653
Barriers, removing to accelerate change, 110
Basic (normal, or happy-day scenario) flow, 497
Basic grade features, Kano, 206
Batch size, DevOps and small, 561
BDD. *see* Behavior-driven development (BDD)
Beck, Kent, 322–325, 336
Behavior, modeled by leaders, 566
Behavioral business rules, 37, 433–438
Behavior-driven development (BDD)
as agile framework, 56
ATDD and, 563–564
feature acceptance criteria, 262–263
preventing last-minute integration issues, 9
story acceptance criteria, 408, 413–414
Behavior-trended funnel, split testing, 494
Best practices, in estimation, 333–335
BestBots. *see* Discovery-driven planning, BestBots case study
Beta testing
closed (private), 534
deployment to customers, 235–236
experimentation in mainstream via closed, 652
of MVPs and all changes, 359
open (public), 534
product release, 533–534
Big Bets, full potential plan for, 226–227, 630
Big room iteration planning
conversation options during, 132
facilitation skills of agile business analyst, 64
scaled iteration planning, 80
scaling agile process, 598–599

Bill of rights, customer's and programmer's
  cantankerous customer and, 10
  customer's, 101–102
  developer's, 102
  XP's Release Planning Game, 323–325
Blanchard, Ken, 565
BLInK case studies
  architecture diagrams map, 311–312
  BPMN private process model, 296–298
  BPMN public process model, 291–292
  cause-effect analysis, 159–161
  cause-effect tree, 163–166
  defined, 147
  Five Whys, 154–157
  improved outcomes, 166
  introduction to, xlii–xliii
  iteration implementation planning, 456–458
  Kano analysis, 209–212
  long-term agile planning-product roadmap, 243–248
  MVP creation, 365–366
  NFRs/constraints analysis, 217–220
  personas, 270–271
  pivot-or-persevere meetings, 546–547
  prioritizing features/specifying NFRs, 217–220
  problem statement, 168–169
  product vision statement, 174–175
  program board creation, 592–595
  release roadmap creation, 348–350
  Sailboat game, 522–523
  seeding product backlog, 200–201
  specifying stories for MVP, 384–386
  stakeholder identification, 179–181
  story map backbone creation, 375–379
  story map completion, 438–440
  traceability, 507–508
  use-case model creation, 501–502
  user-role modeling workshop, 304–306
Blueprint Storyteller, 699
Bold targets, full potential plan, 225–226, 630
Bottlenecks, cumulative flow diagrams, 489
Boundaries, cumulative flow diagram and, 488
Boundary event, BPMN private process model, 295
BPMN. *see* Business Process Model and Notation (BPMN)
Brainstorming, 511
Branding, autonomous, 646–647
Breakout, Open Space events, 614
Breathing, Open Space events, 612
A Brief History painting series (Podeswa), 3–4
Briefly described level, requirements granularity, 128
BSA (Business systems analysts), 63
Buckets, creating personas via, 267
Budget (capacity), planning agenda, 330–331
Bug fixes
  bug-repair stories for, 420
  deployment of, 236
  quarterly and feature planning estimates for, 341
  releasing immediately to market, 531
Build
  analyze-code-build-test cycle, 492
  and distribution processes, 93
Build-Measure-Learn loop, 228–232
Build-Test-Learn cycle, 232
Bulleted outline level, requirements granularity, 128
Bulletin board, Open Space events, 612
Burndown charts
  burnup charts versus, 486–487
  diagnosing productivity with signatures, 482–486
  forecasting with, 330–331, 472
  main elements, 479–480
  monitoring progress, 479–486
  optional elements, 480–481
  quarterly/release, 516–517
  tracking developer tasks, 480–483
Burndown signatures
  healthy signature, 483–484
  overestimating signature, 485–486
  overview of, 482–486
  underestimating signature, 484–485
Burnup charts, 486–487
Business
  empathy in, 658
  inviting representative to Triad meetings, 401
  prioritizes stories, 324
Business analysis (BA)
  certification, 17
  checklist of information artifacts, 122–123
  competency, and Scrum, 45
  contributing to agile enterprise, 628
  definition, 17
  diagnosis, 18–19
  do not skip in agile development, 8–9
  history of, 16–17
  impact of Agile Manifesto on, 28–29
  overview of, 6
  parallel histories of agile and, 16–17
  practiced in agile context, 6–7
  reasons for agile analysis, 15–16
  rules of thumb, 682–683
  ScrumMaster and, 46
  track record, 19–21
  updating use-case model documentation. *see* Use-case model, updating BA documentation
  what twelve principles mean for, 29–31
  why agile should include competency of, 24–25
Business analysis (BA) documentation, updating
  BLInK use-case model, 501–502

feature, 497
persisting stories, 496
tracing analysis artifacts, 506–508
use-case model, 497–501
use-case specifications, 503–506
Business Analysis Benchmark, 24–25
Business Analysis Practice Guide (PMI)
BA domains, 32
business analysis standards, 31
Business analyst
agile analysis vs. waterfall, 67
business systems analyst (BSA) versus, 63
contribution to feature AC, 261
data-informed innovation role, 672
defining, 14–15
maximizing business value, 7
propagating change, 10–11
proxy POs and, 572
roles and functions, 58–63
scaled planning, implementation and, 557–558
soft skills, 63–65
successful full-potential plan, 227–228
supporting PO, 450
Triad meetings and, 402–404
writing acceptance criteria, 408–409
Business case, reviewing in planning agenda, 329
Business data strategist, 672
Business entity, agile unit as separate, 647
Business goals, 32–33, 181–184
Business model disruptions, 642–643
Business objectives
analyzing, 182–184
BA track record for, 20
defined, 33
interim goals to achieve, 383
translate product/epic vision statement into, 181–184
Business Process Model and Notation (BPMN)
private process model, 293–298
public process model, 288–292
reasons to select, 287–288
and UML, 57–58
Business process modeling, 285–298
Business requirements
BABOK v3, 123
defined, 33
Business resources, insufficient, 621
Business rules
analyzing with decision tables, 433–438
behavioral and definitional, 37, 434
defined, 37
Business Rules pattern, splitting stories, 424
Business systems analysts (BSA), 63
Business use-case model, 498
Business value
business analysts maximize, 7
constructing story map ribs, 381
crafting iteration goal for, 449
of story, 395
testing feature leap of faith hypotheses for, 493

C
CA Technologies with Rally Software, 700
Cadence, setting process parameters, 134
Caliber, requirements management tool, 700
Canadian Imperial Bank of Commerce (CIBC), 6
Cancelled iterations, progress in, 513
Cantankerous Customer story, 10–11
Capabilities. *see* Minimum viable product (MVP), capabilities
Capacity (velocity)
adjusting after implementation begins, 137
defined, 35
determining, 135–136
forecasting iteration, 448–449
for multiple teams, 137
Cards
developer task, 454
physical versus electronic stories and, 403–404
on story maps, 368–369
Three Cs of stories and, 397–398
Case studies
BestBots. *see* Discovery-driven planning, BestBots case study
communities of practice (guilds), 669–670
throughout this book. *see* BLInK case studies
Catch event, BPMN private process model, 295
Cause-effect diagrams, root-cause analysis, 157–161
Cause-effect tree, root-cause analysis, 161–166
CD. *see* Continuous delivery (CD)
CDs, as sustaining innovation, 637
CEO (Customer Engagement One) case study
agile analysis and planning, 15–16
feature does not deliver sufficient value, 234–237
nontrivial change to mature product, 255
persona examples, 266
story in three Acts, 74–79
wide and shallow development approach, 239
Certification
in business analysis (BA), 17
in Business Data Analytics, 672
Certified Professional for Requirements Engineering (CPRE), 17, 31, 69–71
Change
Agile corporate culture embraces, 652–653
as agile development benefit, 119
Agile Manifesto on, 28–29
articulating vision of, 652

best trade-off of costs and benefits, 119–121
DevOps continuous delivery and, 559–561
DevOps lightweight management of, 560
feature initiatives of, 254–255
indicating on daily burndown chart, 481
for mature product, 255
specifying interim timeline, 242
in waterfall vs. agile, 14, 66
welcoming, 29
Change agents, business analysts as, 64
Change culture, 320–321
Change management
communications plan, 111–112
for organizations with no agile experience, 110–111
preparing enterprise for agile development, 107
transitioning activities at enterprise level, 109–111
Channels, preparing, 104
Charts, product portrait for, 170–171
Checklist. *see also* Resources and checklists, additional
agile BA information artifacts, 122–123
attendees for quarterly planning, 69
attendees for scaled quarterly and feature planning, 698
for general availability, 535–538
for NFRs and constraints, 689
quarterly and feature planning deliverables, 694
quarterly and feature planning inputs, 693
quarterly release retrospective, 541
quarterly release retrospective questions, 695–697
readiness. *see* Readiness checklist
requirements management tool, 615
stakeholder, 176, 687–688
visioning readiness, 152
Choreography, successful full-potential plan via, 228
Christensen, Clayton, 631, 638–639
CI. *see* Continuous integration (CI)
CIBC (Canadian Imperial Bank of Commerce), 6
Circle, Open Space events, 612
Circles and Soup game, 520, 523–524, 544, 609
Circumstance-based market segmentation
as basis for goals/objectives, 182–184
enterprise agility practice, 630
for feature discovery, 193
incorporating empathy, 657–658
other ways to discover initial features, 198–199
overview of, 193
in Short Lane analysis and planning, 75–77
as source of information for personas, 267
CJA (Customer journey analytics), 658–659
Claims
BPMN private process model, 293–294
BPMN public process model, 288–289
use-case model, 298–299
Closed (private) beta testing, 534, 652
Cloud (AWS) competency group, 578
CMS (Configuration management system), 131–133
Coach
BA responsibilities of, 61
leader as, 564
leader provides vision as, 660
Cockburn, Alistair, 331
Code-test-learn, split testing with funnel metrics, 493
Cognitive empathy (perspective taking), 656, 658
Colbert, Steven, 7
Collaboration
agile analysis vs. waterfall, 66
agile corporate culture practice of, 663
Delphi estimation, 339–340
DevOps culture of, 559–560
Dutch polder model of, 665–666
internal (within enterprise), 663–665
iteration review and, 516
outside enterprise, 665
planning stakeholder, 176–178
Collaborative culture
of business people and developers, 29
nurturing, 10–11
Colocation, big room iteration planning, 598
Columns
determining Kanban states, 144
Kanban board, 459–462
Commitment
agile planning, 352
balancing adaptability versus scope, 342
for committed vs. targeted features, 343
to goals and objectives over features, 329
to iteration goal, 449
to outcomes, not output, 666
to planning implementation, 455–456
to quarterly and feature planning agenda, 348–350
to quarterly goals and objectives, 341
to scope forecast, 341–342
to sprint planning meeting, 597
stories to avoid overcommitment, 324
why quarterly plan is sometimes a promise, 321–322
Commitment phase, XP Quarterly/Release Planning Game, 322
Communication
business analysis diagnosis and, 19
of non-colocated teams, 618
Communications plan, 111–112, 178–179
Communities of practice (CoPs), or guilds, 668–682
Compassionate empathy (empathic concern), 657
Competency
agile analysis vs. waterfall, 65
forming guilds around, 669–671

groups, 577–578
organize teams around value, not, 95–96, 98
Complete, well-formed AC as, 412
Complex UI pattern, splitting stories, 426
Complexity
point estimates measuring, 335
team dependencies due to product, 554
Compliance
defined, 104
preparing, 104–106
story, 420
value stream mapping for, 284
Component teams
overview of, 569–570
scaling agile organization, 571
supporting extended team, 577
Comprehensive
configuration management in DevOps as, 561
product backlog as, 125
transparency must be, 667
Computers, for repetitive work in DevOps, 561
Comstock, Beth, 652
Concierge MVPs, 362
Concise, well-formed AC as, 412
Condition-response table, business rules, 437
Conditions, specifying business rules, 436–437
Configuration management database (CMDB), 131–132
Configuration management, DevOps comprehensive, 561
Configuration management system (CMS), 131–133
Confirmation
specifying story AC. *see* Acceptance criteria (AC), story
Three Cs of stories, 398
Conflict, story point estimates reduce, 336–337
Confluence, requirements management tool, 700
Connextra template
defined, 40
describing/estimate features with, 332
overview of, 193
representing epics/features, 193
specifying features, 259
specifying products or epics, 166–167
summarizing root-cause analysis, 166
tuning user stories, 139, 141
when to write/not write story with, 404–405
writing stories with, 395
writing story description with, 405–407
Consensus-based decision making, 662
Constraints
checklist for NFRs or, 689
defined, 37
in discovery-driven planning case study, 703–705
regulatory, 236
seeding product backlog, 216–217
Construction phase, RUP lifecycle, 49
Constructive failures, 364
Context
diagrams, 307–308
feature preparation via, 263–264
specifying for personas, 269
tailoring agile practice to, 118–121
Continuous analysis, Kano, 208
Continuous delivery (CD)
ATDD and BDD, 563–564
automation in test-build-deploy steps, 558–559
cadence of, 43
CD and CI, 561–562
DevOps practices, 559–562
of high value in agile analysis, 14
history of agile development, 18
how it works, 233–234
MVP, 233–234
overview of, 558
quarterly release retrospective, 540
in rolling analysis, 469
of valuable software, 29–30
Continuous integration (CI)
delivery cadence and, 43
DevOps practice of, 561
how it works, 233–234
innovative development and, 632
MVP deployment, 233–234
quarterly release retrospective, 540
in rolling analysis, 469
and trunk-based development, DevOps, 562
Contract, avoid viewing quarterly plan as, 321
Convergence, definition of, 639
Conversation
agile analysis vs. waterfall, 66
managing dependencies, 132
Three Cs of stories, 398
user story as reminder to have, 39
Conversion rate metrics, 492–494
Cooper, Alan, 265
CoPs (communities of practice), or guilds, 668–682
Core values, Agile Manifesto, 28–29
Corporate culture
achieving enterprise agility, 632–633
definition of, 633
Corporate culture, Agile practices
acceleration, 653–655
bust silos, 667–672
collaboration, 663–666
commit to outcomes, not output, 666
data-informed innovation, 672–673
distributed authority, 659–663

embrace change, 652–653
empathy, 655–659
iterative experimentation, 650–652
let those who do the work estimate the effort, 663
monitor adjacent and low-end markets, 673–675
overview, 634–635
responsible procrastination (last responsible moment), 659
transparency, 666–667
Corporate culture, Agile principles
invest aggressively in enterprise agility, 648–650
overview, 634–635
protect islands of innovation, 644–647
tailor approach to circumstance, 635–643
Cost of delay
determining, 126–127
Lean software development and, 53
responsible procrastination and, 659
sequencing epics and features in backlog, 212
Costs
of agile development, 118
agile development reduces, 119
assessing for backlog items, 193
finding best trade-off of benefits and, 119–121
impact of BA on, 20–21
operational, 215
COVID-19 pandemic, as a trigger for business model disruptions, 642–643
CPRE (Certified Professional for Requirements Engineering), 17, 31, 69–71
Crest Whitestrips, disruptive innovation, 636, 638, 640
Cretaceous-Tertiary (K-T), extinction event, 638–639
Critical events, on daily burndown chart, 481
Cross-functional teams, organize around value, 668
Cross-stream (horizontal) traceability, 130
Cross-team dependencies, 600
CRUD acronym, writing high-quality user stories, 421
Cultivating Communities of Practice (McDermott), 672
Culture. *see also* Corporate culture
defined, 633
fostering rapid learning, 566
nurturing collaborative, 10–11
organizing development teams, 94
Cumulative flow diagrams, 487–490
Curiosity of business analyst, 64
Customer
bill of rights, 101–102
collaboration, Agile Manifesto, 28–29
journey map, 273–278
story of cantankerous, 10–11
Customer Engagement One (CEO) case study, 15–16
Customer journey analytics (CJA), 658–659
Customer-developer relationship, 101–102
Customers
balancing user features and technical debt, 341
beta testing by end, 533–534
concerns about deployment frequency, 237
corporate culture obsession with, 633
deployment to beta, 235–236
fostering cognitive empathy with feedback from, 658
identifying opportunities via, 632
journey map for improved experience of, 278–279
needs of, 667
provide proposed features to, 330
selecting in Kano analysis, 203
successful full-potential plan via, 227
Customer's and Programmer's Bill of Rights, 323–325

D

DAD, hybrid framework, 583
Daily Activities zone
defined, 73
rolling analysis in. *see* Rolling analysis and preparation (day-to-day activities)
Short Lane analysis and planning in, 77–78
Daily burndown charts. *see* Burndown charts
Daily planning and analysis, 80
Daily Scrum. *see* Daily standup
Daily standup
attendees, 473
day in the life of agile analysis, 468–469
defined, 47–48
facilitation tips, 472
forecasting, 474
monitoring progress, 471–474
objectives, 472
overview of, 471
scaling agile process, 600
self-organization, 472
status updates to team, 473–474
timing considerations, 472
Dark launch, testing MVPs, 359
Data analyst, 672
Data Complexity pattern, splitting stories, 425
Data engineer, 672
Data flow diagrams (DFDs), 308–310
Data integrity, defined, 35
Data scientist, 672
Data-informed decision-making, 321, 545
Data-informed innovation, 672–673
Day to day activities, 524
DDD (Domain-driven design), 57–58
Debriefing, Open Space events, 614
Decision tables, business rules/AC, 433–438
Decisions
delaying in Agile planning, 42
distributed authority approach to, 659–663

Dutch polder model of making, 666–667
postponing until last responsible moment (LRM), 659
Decomposing stories, 136
Defects, as waste, 52
Definition of done (DoD)
examples of, 138
reviewing in iteration planning, 449
reviewing in sprint planning meeting, 597
in rolling analysis, 469
in Scrum, 46
specify only one across whole product, 569
tuning, 137–138
Definition of ready (DoR)
example of, 138–142
feature, 142–144, 258
inter-team collaboration, 617
reviewing in iteration planning, 449
reviewing in sprint planning meeting, 597
in Scrum, 46
team-level story preparation, 606
tuning story, 138
Definitional business rules, 37, 434
Delayed requirements, managing stakeholder expectations, 99
Delighters
documenting personas for, 269
Kano grades, 206–207
Deliverables
decision table, 434
iteration planning, 446–447
iteration retrospective, 518
iteration review, 514–515
quarterly and feature planning checklist for, 694
quarterly release retrospective, 539
quarterly/feature planning event for, 326–327
scaled quarterly and feature planning, 588
sprint planning meeting, 596
Triad meetings on user stories, 401
Delivery
agile fluency model, 108
cadence, 43, 54
MVP deployment and, 232–234
Delphi estimation, 338–340
Demo, iteration review, 515–516
Dependencies
analyst role in preventing, 557
indicated on program board, 589
managing/identifying, 344–347
prioritizing story to prevent, 381
recurring issues with, 620
resolving waterfall, 619
SoS meetings addressing cross-team, 600
story map relationship, 370
story preparation, 511
threatening delivery of upcoming feature(s), 605
Deployment
to beta customers, 235
of bug fixes and minor changes, 236
customer concerns on frequency of, 237
deferred, due breaking a flow, 236–237
deferred, due to technical limitations, 236
deferred versus immediate, 234–237
delivery versus, 233
environment types, 231–232
impact of regulatory constraints, 236
inability to achieve frequent and reliable, 620
of major features and enhancements, 236
MVP delivery approach and, 232–234
options and potential issues of, 234
to sacrificial product, 235
Design group, scaling agile organization, 577
Developer
basing estimate on capable, 333
has final say on estimates, 325
in product owner council, 580
relationship with customer, 101–102
role in Triad meetings, 402–403
Developer task board
forecasting using, 474
iteration implementation with, 453–454
as iteration planning deliverable, 446
updating, 475
Developer task cards, iteration planning, 454
Developer tasks
daily burndown chart, 479–480
feature preparation, 258–259
identifying iteration development, 452–453
measuring progress on burnup chart, 486
signatures on burndown charts, 483–486
sprint planning, 47
tracking, versus stories on burndown, 482
Development
agile diagnosis and, 22
BA competency and, 24–25
empathy in product improvement and, 658
history of, 17–18
infrastructure spike (or story), 419
investment in technology to accelerate product, 648–649
journey map for investment in, 278
managers in product owner council, 580
managing stakeholder expectations about, 99–101
MVP environment for, 231–232
preparing infrastructure, 90–93
role, XP Quarterly/Release Planning Game, 323
story estimation by, 324
track record, 22–23

of value stream map, 284–285
value streams, 283
Development teams
attending backlog seeding, 197
business can lead technical/engineering, 668
collaboration between customers and, 10–11
extended, 97
feature vs. generic teams, 96–97
forming cross-functional teams around value, 668
organizing around value, 93–96
overview of, 93
pre-alpha testing by, 533
transitioning to agile development, 108–109
DevOps
as agile framework, 56
benefits of, 560
collaborative culture, 559
defined, 559
delivery cadence and, 43
determining traceability, 130
history of, 18
Microfocus ALM Octane tool integration with, 699
MVP deployment, 232
practices, 560–561
preparing testing infrastructure, 91–93
quarterly release retrospective, 540
quarterly release retrospective checklist, 695
resources on, 562
scaling agile organization, 578
DFDs (data flow diagrams), 308–310
Diagnostics, with burndown signatures, 482–486
Differentiating quadrant, purpose alignment model, 88–89
Differentiator MVPs, 360
Difficult people, business analysts work with, 64
Digital camera, as low-end disruption, 641
Discovery-driven financial planning
agile financial planning, 103
hypotheses in, 189
overview of, 675–676
Discovery-driven planning, BestBots case study
background, 701–702
create assumptions checklist, 708–709
create milestone planning chart, 710–711
create pro forma operations specifications, 706–708
determine constraints (required outcomes), 703–705
draft reverse income statement, 705–706
initial market analysis, 702–703
overview of, 701
revise reverse income statement, 709–710
Discussed, well formed stories as, 421
Disruptive innovation
adapting culture for, 644
adapting to sustaining innovations versus, 644
business model disruptions, 642–643
cost-benefit calculation, 121
determining enterprise agility, 636–644
does not have to be of low quality, 641
as enterprise agility principle, 631
as evolutionary leap, 638–639
litmus test for identifying disruptions, 643–644
low-end disruptions, 641–642
mainstream disruptions, 642
new-market disruptions, 642
protecting islands of, 644–647
Uber and, 640–641
understanding, 637–638
updates to Christensen's model of, 639–640
Dissatisfiers, Kano grades, 206
Distributed authority
agile corporate culture practice of, 659
be like the octopus, 661
benefits of, 660
elements that must be present for, 660–661
holacratic approach to, 662–663
localized and individualized, 661–662
Distribution team, preparing, 103–104
Documentation
agile analysis vs. waterfall, 66
analysis, 538
feature, 497
focus on compliance goals, not means, 105–106
increasing for non-colocated teams, 619
of information on personas, 268–269
tracing analysis artifacts, 506–508
updating BA. *see* Business analysis (BA) documentation, updating
DoD. *see* Definition of done (DoD)
Domain-driven design (DDD), 57–58
DoR. *see* Definition of ready (DoR)
Downstream (forward) traceability, 129–130, 507
Downward traceability, 507
Drivers, for agile organization, 628–629
Dropbox, Preorders MVP, 363
Duration, iteration planning, 445
Dutch polder model, collaboration, 665–666
Dynamic, product backlog as, 125

E
Effort, measured by point estimates, 335
Elaboration phase, RUP lifecycle, 49
Electronic stories, versus physical, 403–404
Elements, daily burndown chart, 479–480

Emergent AC, specifying, 413
Emergent features
  agile vision statement leaves room for, 173
  specifying, 200
Emotional empathy, 656–657
Emotional marketing, 657
Empathic concern (compassionate empathy), 657
Empathy
  agile corporate culture practice of, 655–656
  benefits of, 656
  cognitive, 656
  compassionate, 657
  defined, 655
  emotional, 656–657
  practical tools, 657–659
Enabler story. *see* Spikes, SAFe
End customers, beta testing by, 533–534
End event, BPMN private process model, 293
End user, user story must deliver to, 395
End-to-end process, agile development, 8–9
End-to-end UAT, 263, 286
Engineering and component group, 577
Engineering teams, business can lead product, 668
Enhancements
  actionable metrics for, 188
  deployment of, 236
Enterprise, achieving agility
  agile financial planning, 675–676
  culture. *see* Corporate culture
  foundational practices, 629–631
  innovative product development process, 631–632
  introduction, 623
  on the map, 624–626
  mapping to IIBA and PMI guides, 681
  objectives, 623, 626
  overview of, 627–629
  summary of, 676
Enterprise, preparing for agile development
  agile enterprise transition team, 112
  agile fluency model, 107–108
  communications plan, 111–112
  overview of, 107
  transition activities, 109–111
  transition team, 108–109
  transition timeline, 111
Enterprise and strategy analysis, as BA focus, 17
Entry conditions, quarterly/feature planning, 325–326
Environment types, deployment, 231–232
Epic vision statement, 172–173, 181–184
Epics
  definition of, 196
  to features and stories from, 38
  features often begin as, 254
  long-term planning requirements and, 224
  nontrivial change to mature products as, 255
  ongoing analysis of upcoming, 509–512
  overview of, 38–39
  planning long-term. *see* Product roadmap
  portfolio, 574–575
  program, 575
  represent user capabilities as, 183
  requirements-related terminology, 38–39
  Role-feature-Reason template representing, 199
  rules of thumb for estimating/splitting, 682–683
  scaling backlog, 566–567
  sequencing in backlog, 212–215
  specifying acceptance criteria, 260–261
  taxonomy of story size, 395–396
  visioning. *see* Visioning
Estimates
  development creates story, 324
  development has final say on, 325
  done by those who do the work, 663
  feature preparation walkthrough, 605
  feature prioritization using broad, 214
  setting standards for, 134–135
  for splitting stories and epics, 682–683
  story, 40
  for time to complete developer tasks, 454–455
  using for functional spikes, 418
  for well-formed stories, 421
Estimates, quarterly and feature planning agenda
  bug fixes, 341
  Delphi estimation, 338–340
  describe and estimate features, 332
  iteration planning, 341
  no-estimating approach, 338
  objectives of, 332
  spikes, 340
  technical stories and nonfunctional requirements, 341
  time spent on, 332
  units and methods for, 334–338
  work included in, 333
Evans, Eric, 57–58
Events
  developing collaborative culture by holding, 664
  overview of scaled, 583–585
  quarterly release retrospective, 542–543
  specifying for interim periods, 243
Evolutionary leap, disruption as, 638–639
Excitement feature, Kano grades, 206–207
Exclusive gateway, BPMN private process model, 293
Executive support, for organizations with no agile experience, 110
Expected features, Kano grades, 206

Experimental failure, agile corporate culture values, 633
Exploration phase, XP Quarterly/Release Planning Game, 322
Extended team, 97
External view of process, BPMN public process model, 288–292
Extreme environmental stressor, evolutionary disruption from, 638–639
Extreme Programming Explained (Beck), 48, 322–325
Extreme Programming (XP)
  as agile framework, 48–49
  history of agile development, 17
  requirements units as stories in, 38
  term "story" originates in, 395
  timeboxed planning in, 121, 555

F

Facilitation
  additional tips, 684–685
  consensus-based decision making, 662
  daily standup tips for, 472
  quarterly release retrospective, 539–542
  scaled quarterly and feature planning, 589
  skills of business analysts, 64
  stakeholder engagement and analysis, 179–181
  tips for cause-effect diagrams, 157–158
  tips for stakeholder events, 152
  using product portrait as visioning tool, 170–171
Facilitator, agile analysis vs. waterfall, 65
Fail fast (iterative experimentation) practice, Agile corporate culture, 650–652
Failure, iterative experimentation and, 651–652
Feature card, story map, 368–369
Feature Closeout, Short Lane analysis and planning, 78
Feature definition of ready (DoR)
  definition of, 9
  preparing for planning event, 326
  quarterly and feature planning preconditions, 586
  quarterly and feature preparation using, 258
  readiness checklist for quarterly planning, 690
  scaled planning and implementation, 557
  tuning, 142–144
  for upcoming requirements item, 9
Feature demo, 606
Feature ready, rolling analysis, 469
Feature set, 318, 570–571
Feature teams
  defined, 575
  with extended team, 576
  forming, 575–576
  versus generic teams, 96–97
  scaling backlog, 569
  structure of, 97
Features
  acceptance criteria, 40, 215–216
  during alpha testing, 533
  attributes of, 201–202
  during beta testing, 534
  deferred vs. immediate deployment of, 234–237
  defining independent, 199
  definition of, 196
  deployment of major, 236
  discovering via circumstance-based market segmentation, 198
  documenting, 497
  from epics to, 38
  grading in Kano analysis, 204–207
  how many to seed up front, 196–197
  initial preparation for scaled initiative, 585
  lifecycle, 40–41
  narrow/deep versus wide/shallow approach to, 237–240
  ongoing analysis of upcoming, 509–512
  other ways to discover, 198–199
  overview of, 39
  PBI attributes, 201–202
  physical representation of, 200
  planning. *see* Quarterly and feature planning
  preparation activities, 510
  preparation of. *see* Quarterly and feature preparation
  preview meeting, 462–463, 599
  prioritizing in long-term planning, 224–225
  prioritizing to complete product portrait, 217–220
  refining incrementally, 327–328
  Role-feature-Reason template representing, 199
  scaled (quarterly) preparation of, 602–605
  scaled quarterly retrospective and, 609–611
  scaling backlog, 566–567
  sequencing in backlog, 212–215
  specifying emergent, 200
  specifying for interim periods, 242–243
  stakeholder productivity expectations of, 100
  story maps plan coherent set of, 367
  target, in Kano analysis, 202
  taxonomy of story size and, 395–396
  team dependencies due to interconnected, 553–554
  use cases or user tasks sized as, 382
  user capabilities as, 183
  as waste, 52
Feedback
  alpha testing for, 533
  from beta testing, 359
  in Delphi estimation, 339, 340
  fostering cognitive empathy with customer, 658
  Lean software development, 53
  successful full-potential plan via, 227–228
  voice of the customer as, 658

Feelings, as journey map component, 278, 282
Fibonacci sequence, in story estimation, 40, 337–338
Field research, circumstance-based market segmentation, 630–631
Final review, iterations, 516
Financial planning
  achieving enterprise agility, 675–676
  data-informed, 672–673
  preparing organization, 102–103
Fishbone diagrams, root-cause analysis, 157–161
Five W questions
  problem or opportunity statement, 167–168
  product portrait, 170
Five Whys method, root-cause analysis, 153–157, 161–162
Flickr
  as constructive failure, 364, 651
  as disruptive innovation, 638
  empathy in development of, 658
Flow-based feature planning
  overview of, 318
  quarterly planning versus, 315, 319–320
Flow-based planning
  continuous implementation/delivery via, 558
  determining requirements granularity levels, 127
  feature review via, 607
  frameworks supporting, 121
  iteration implementation, 451
  Kanban board columns for, 459–462
  Kanban using, 42
  rolling analysis using, 469
  rolling preparatory analysis using, 509
  story planning via, 444
  story preparation via, 394
  timeboxed planning versus, 121, 555
  using for frontend, 555–556
Flows, use-case
  tracing analysis artifacts, 506–508
  updating specifications, 505–506
  updating use-case model, 497
Fluency model, agile, 107–109
Focusing, agile fluency model, 108
"Focusing on threes," embracing change, 653
Follow-up meeting, monitoring progress, 474
Forecasting
  accomplishments in sprint planning meeting, 595–597
  all developer tasks in backlog, 480
  commitment to scope, 342
  delivery of feature, 330–331
  feature/story delivery time via story points, 335
  goal and scope of iteration, 447–451
  iteration review, artifacts for, 516
  measuring past velocity for, 331
  quarterly plan as, 321
  stories that will be delivered, 450–451
  updates, 474
  using burndown charts for, 482
  without estimating, 338
Foresight, hindsight as best, 333
Forward (downstream) traceability, 129–130, 507
Foundational practices, enterprise agility, 629–631
14th Annual State of Agile Report, 22
Frameworks, agile
  ATDD. *see* Acceptance test-driven development (ATDD)
  BDD. *see* Behavior-driven development (BDD)
  determining, 121
  DAD, 583
  DevOps, 56
  Domain-driven design (DDD), 57–58
  Kanban, 44
  Lean software development, 51–55
  Lean startup, 55
  Lean Thinking, 50–51
  LeSS, 583
  NEXUS, 583
  overview of, 43
  Rational Unified Process (RUP), 49
  SAFe. *see* Scaled Agile Framework (SAFe)
  scaled, 582–583
  Scrum, 44–48
  TDD. *see* Test-driven development (TDD)
  UML and BPMN, 57–58
  use cases, 49–50
  XP. *see* Extreme Programming (XP)
Franklin, Andrea, 105
Frequency
  constructing story map ribs, 381
  of POC meeting, 601–602
  for pruning and ordering meetings, 512
Frontend, flow-based approach to, 555–556
FRs (functional requirements), 34, 380–381
Full-potential plan
  business analyst contribution, 227–228
  create detailed plan, 226
  deliver quick wins, 226–227
  enterprise agility practice, 630
  long-term planning, 225
  MVP implementation. *see* Minimum viable product (MVP), capabilities
  MVPs validate assumptions, 228–230
  product roadmap for. *see* Product roadmap
  set bold targets, 225–226
Full-time membership, development team, 95
Fully described level, requirements granularity, 129

Functional requirements (FRs), 34, 380–381
Functional spikes
  feature preparation, 258–259
  naming standards example, 395–396
  overview of, 416–417
  uncertainty pattern, 427–428
Fundamentals of agile analysis and planning
  agile frameworks, 43–58
  Agile Manifesto, 28–29
  agile planning, 42–43
  agile roles and BA, 58–63
  key practices in agile vs. waterfall, 65–68
  mapping to IIBA and PMI guides, 678
  objectives, 27
  requirements-related terminology, 32–41
  rules of thumb, 68
  soft skills of agile business analysts, 63–66
  standards, 31–32
  summary of, 68
  twelve principles, 29–31
Funnel metrics, split testing outcomes, 492–494
Future, embracing change in, 653

G
Game Neverending
  empathy when developing Flickr from, 657–658
  Flickr born out of failure of, 638, 651
Games, iteration retrospective, 520–524
Gateway, in BPMN public process model, 288
Gating, avoiding DOR, 142
General availability (GA) stage
  analysis documentation, 538
  general availability checklist, 535–538
  monitoring, 538
  product is accessible in, 535
  rules of thumb, 683
  value validation, 539
Generic versus feature teams, 96–97
Gherkin feature files
  continuous development and, 563–564
  feature documentation with, 497
  specifying automated UAT, 93
  specifying end-to-end UAT, 263
  specifying feature AC, 56, 260
  Gherkin syntax, 408
Given-When-Then template, BDD
  feature AC examples using, 262–263
  feature documentation, 497
  seeding product backlog, 76
  using, 413
  syntax, 408
Goals
  BPMN private process model, 294–295
  business, 32–33, 181–184
  commitment to, 329–330, 341
  crafting common iteration, 597
  crafting for planning agenda, 329–330
  crafting interim, 242, 383
  crafting iteration, 449–450
  daily standup supports shared team, 472
  feature prioritization supports strategic, 214
  focus on compliance, 105–106
  forecasting iteration, 447–451
  identifying persona, 268–269
  iteration planning, 446
  as journey map component, 282
  Multiple User Goals pattern for user stories, 426–427
  planning agenda using outcome-based, 329–330
Goals and objectives analysis, visioning, 147
Goldratt, Eliyahu M., 161
Google Docs, 360, 700
Governance, 104–106
Grades, in Kano analysis, 204–207
Grand Lane
  Agile Analysis and Planning Map, 79–81
  defined, 73
  scaling agility focus on. *see* Scaling agility
Grant, Adam, 633
Granularity levels, 125, 127–129
Gravity of past success, sustaining, 645
Greenleaf, Robert K., 564–565
Groupon, as constructive failure, 364
Growth
  accelerating operational capacity for, 654
  hypotheses, 186
  investment in technology for, 649–650
Guesstimate stories delivered, for initial capacity, 136
Guiding coalition, accelerating change, 109
Guilds, as communities of practice, 668–682
Gutsche, Jeremy, 633

H
H&M, purpose brand for, 646
Happy-day scenario flow, 497
Hardening (stabilizing) iterations, 531–532
Healthy signature, burndown chart, 483–484
Heaven painting (Podeswa), 3
Heavily regulated sectors, agile in, 629
Hell painting (Podeswa), 3–4
Hesse, Hermann, 565
High-level functionality, via interim goals, 383
High-level use cases, product-development value, 668–669
High-quality stories, writing guidelines, 420–421
High-risk (fixed-price solution), targeting agility level for, 119–120
Hindsight, as best foresight, 333
Holacratic approach, distributed authority, 662–663

Hoote Suite, 556
Horizontal (cross-stream) traceability, 130, 507
"How," well-formed AC does not describe, 412
Humor, business analysts and, 65
Hybrid approach
  supported by most organizations, 121
  updating use-case model, 498
Hypotheses
  assumptions/hypotheses card, story map, 368–369
  leap of faith. *see* Leap of faith hypotheses
  quarterly plan, 321
  value, 186, 188

I

IBM Doors Next tool, 700
Ideal developer days (IDDs), estimation, 40, 335–336
IIBA (International Institute for Business Analysis), 14–15
IIBA guides, mapping of book chapters to, 677–681
Impact, problem or opportunity statement, 167–169
Impact and influence matrix, stakeholder communication, 178–179
Impediments
  accelerate growth by removing, 655
  story preparation, 511
Implementation
  of developer tasks, 471
  sprint planning meeting, 597–598
Implementation, iteration and story planning
  case study, 456–458
  identifying developer tasks, 452–456
  inviting PO, 451
  overview of, 451–452
  steps, 452
Implementation sequence, story maps, 366–367, 379–380
Improvement plan, iteration retrospective, 519
In Search of Excellence (Peters), 88
Inception phase, RUP lifecycle, 49
Increment
  defined, 45
  as iteration planning deliverable, 446–447
Incremental process, refining features and AC as, 327–328
Incremental scaling, MVPs, 364–365
Incumbent businesses, 654
Independent
  features, 199
  well-formed stories as, 421
Indifferent features, Kano grades, 207
Individual
  as core value of Agile Manifesto, 28
  decision-making authority, 661–662
  well-formed stories as, 421
Information
  checklist of BA artifacts, 122–123
  conveying via face-to-face conversation, 29
  corporate culture commitment to transparency of, 666–667
  hoarding via silos, 667
Information radiators, 52, 54–55
Information security (Infosec) group, 578
Informative workspaces, XP, 49
Infrastructure
  investing in technology, 649–650
  preparing development/testing, 90–93
In-house infrastructure, 649
Initial capacity (velocity), 135–137
Initial market analysis, discovery-driven planning, 702–703
Initial preparation, scaled initiatives, 585–586
Initiation and Planning zone
  defined, 72
  how long to spend upfront on, 87–88
  long-term agile planning, 222–223
  Prepare the Organization in. *see* Organizational preparation
  Prepare the Process in. *see* Process preparation
  scaling agility in. *see* Scaling agility
  Short Lane analysis and planning in, 74–75
  understanding, and resources for, 86–87
  visioning tools. *see* Visioning
Initiatives
  Big Bet, 226
  long-term planning, 224
The Inmates Are Running the Asylum (Cooper), 265
Innovation
  corporate culture practice of data-informed, 672–673
  defining, 637
  disruptive. *see* Disruptive innovation
  protect islands of, 644–647
  sustaining, 637
  types of, 637
Innovation and planning (IP) iteration, SAFe, 582
Innovation Games, agile collaboration tool, 700
Innovative development
  agile process for, 631–632
  approach to, 5
  for mainstream businesses, 5–7
  MVP case study, 357–358
  process for, 631–632
The Innovator's Dilemma (Christensen, Raynor, and McDonald), 631, 644
The Innovator's Solution (Christensen, Raynor, and McDonald), 631, 644
Input artifacts, planning agenda, 331

Inputs
iteration planning, 445
iteration retrospective, 518
iteration review, 514–515
Open Space events, 612
preparing quarterly/feature planning event for, 326
quarterly and feature planning checklist, 693
scaled feature preparation, 604
scaled quarterly and feature planning, 588
sprint planning meeting, 596
story map backbone, 373–374
to Triad meetings on user stories, 401
Inspect-and-adapt event, iteration reviews, 514
Inspect-and-adapt tool, daily standup, 471–474
Institution-as-servant principle, 565
Integration
meetings, scaling agile, 599
recurring issues with, 620
SoS meetings address issues of, 600
Integration Capabilities pattern, splitting stories, 428
Integrity, Lean software development principle, 54
Intentional destruction, in agile corporate culture, 633
Interaction, core value of Agile Manifesto, 28
Interconnected features, and team dependencies, 553–554
Interim goal card, story map, 368–369
Interim goals, Timeline view, 383–387
Interim periods, product roadmap for planning, 241–244, 246–247
Intermediate event, BPMN private process model, 293
Internal collaboration, culture of, 663–665
International Institute for Business Analysis (IIBA), 14–15
International Requirement Engineering Board (IREB), 31
Internet of Things (IoT) development, 5, 286
Interoperability
alpha testing for, 533
defined, 35
Inter-team collaboration
analyst role in, 557–558
choosing approach to, 554–558
DevOps culture for, 559–560
lightweight tools supporting, 615–617
of scaled agile teams, 553–554
Introduce Dropship Capability, UAT for end-to-end workflows, 263
INVEST guidelines, crafting user stories, 421
Investment in enterprise agility, 648–650
Invitees. *see* Attendees
IoT (Internet of Things) development, 5, 286
Ishikawa diagrams, root-cause analysis, 157–161
Iteration
accounting for progress at end of, 513
Lean software development, 53
MVP process of, 363
starting, 351
Iteration and story planning
attendees, 445
duration, 445
feature preview meeting, 462–463
forecast goal and scope, 447–451
implementation planning, 451–458
introduction to, 441
iteration planning deliverables, 446–447
iteration planning inputs, 445
iteration planning parts, 444
Kanban board setup, 458–462
looking two iterations ahead, 463
on the map, 442–443
mapping to IIBA and PMI guides, 680
objectives, 441, 444
overview of, 444–445
planning rules, 447
rules, 447
scaling iteration planning, 462
story planning overview, 444
summary of, 463
Iteration backlog, 446
defined (as sprint backlog), 47
Iteration Closeout zone
Grand Lane analysis and planning in, 80
rolling analysis in. *see* Rolling analysis and preparation (day-to-day activities)
Short Lane analysis and planning in, 78
Iteration demo, 606. *see* Iteration review
Iteration goal, 446
Iteration goal card, story map, 368–369
Iteration Inception zone
defined, 73
Short Lane analysis and planning in, 78
story planning. *see* Iteration and story planning
Iteration planning
quarterly and feature planning estimation, 341
sprint planning as, 47
Iteration retrospective
attendees, 518
games, 520–524
inputs and deliverables, 518
overview of, 517
timing considerations, 517
topics/agenda, 518–520
Iteration review
artifacts for forecasting/tracking progress, 516–517
bring waterfall teams to, 620
inputs and deliverables, 514–515
overview of, 514
topics/agenda, 515–516

Iterative experimentation (fail fast) practice, Agile corporate culture, 650–652
Iterative-incremental development, 6, 17
It's Not My Problem story, 8–9

J
Jacobsen, Ivar, 49
JAMA software tool, 699
Jeffries, Ron, 582
JIRA tool, 699
Jobs
  circumstance-based market segmentation identifies, 630–631
  duration of, 126–127
  organization based on, 668–669
  titles for, 664
Journey mapping
  case study, 279–282
  components, 274–278
  customer journey map, 273–274
  defined, 272
  empathy in business operations and, 658
  feature preparation via. *see* Journey mapping
  more on, 283
  personas guiding, 269
  using, 272–273, 278–279
A Journey to the East (Hesse), 565
"juicy bits" first (user/business value), story map ribs, 381
"Just Talk," inter-team collaboration via, 616
Just-in-time requirements analysis, 66

K
Kanban
  as agile framework, 44
  board setup for iteration planning, 458–462
  cumulative flow diagrams, 487–490
  customer-generated requests, 15–16
  feature planning. *see* Flow-based feature planning
  flow-based planning. *see* Flow-based planning
  as origin of agile development, 17
  requirements granularity levels, 127
  review practices, 347–348
  timing of feature preparation, 257
  tuning workflow parameters, 143–144
  work items, 38
Kanban board
  setting up, 454–462, 474
  updating, 476–479
Kano analysis
  case study, 209–212
  conduct survey, 204
  continuous analysis, 208
  create prototypes, 204
  create questions, 203
  determining customer value of feature, 212
  grade features, 204–206
  grade interpretation, 206–207
  natural decay of delight (and its opposite), 208
  process overview, 202
  satisfaction versus fulfillment graph, 207–208
  select customers, 203
  select target features, 202
  test questionnaire internally, 204
Karl Lagerfeld Pour H&M, 646
KAs (knowledge areas), *BABOK Guide*, 31–32
Kasparov, Garry, 645
Key practices, agile vs. waterfall, 65–68
Knowledge areas (KAs), *BABOK Guide*, 31–32
Kofman, Jeffrey, 357–358
Kotter, John, 109–110, 172–173
K-T (Cretaceous-Tertiary), extinction event, 638–639

L
Lab, MVP testing in, 358–359
Lanes, Agile Analysis and Planning Map
  Grand Lane, 79–81
  graphics of, 69–71
  Long Lane, 79
  Short Lane, 74–78
  summary of, 81
  understanding, 72–73
Lanes, BPMN private process model, 293
Lanes, process modeling and swimlanes, 287
Large Initial Effort pattern, splitting stories, 424
Large Scale Scrum is Scrum, scaled agile framework principle, 582
Large Scale Scrum (LeSS) framework
  "Just Talk" guideline, 616
  scale agile approach of, 553
  timeboxed planning in, 555
Last responsible moment (LRM)
  agile corporate culture practice of, 659
  agile financial planning using Real Options for, 675
  Lean software development, 53
  timing of feature preparation and, 257–258
Latent requirement, Kano grades, 206–207
Law of Two Feet, Open Space events, 613
Lawrence, Richard, 422
Lawrence patterns, 394. *see also* Splitting stories, patterns
Leader as Coach, 564
Leader Who Serves, 565
Leadership
  effective agile, 564–566
  empowering others to make their own decisions, 660

Lean pull mechanism, forecast stories to be delivered, 450–451
Lean software development
  as agile framework, 51
  history of, 628
  information radiators, 54–55
  principles of, 54
  seven wastes, 51–52
  tools of, 52–54
Lean Software Development: An Agile Toolkit (Poppendieck and Poppendieck), 628, 675
The Lean Startup, 628
Lean startup
  actionable metrics in, 186
  as agile framework, 55
  and MVP, 630
  MVP planning in, 356
  understanding, 185
Lean Thinking, 50–51
Leap of faith hypotheses
  analysis of, 185
  assumption analysis, 190–191
  assumption checklist, 189–190
  crafting iteration goal for learning value, 449
  in discovery-driven planning, 189
  growth hypotheses, 186
  identifying in innovative development, 632
  lean startup, 55
  lean startup and, 185, 630
  metrics, 187–188
  milestone planning chart, 190
  MVP approach begins with, 228–229
  MVP case study, 357–358
  MVP process, 230
  pivot-or-persevere meeting to validate, 544–547
  understanding, 185–186
  validating by creating MVP, 630
  value hypotheses, 186
  visioning process, 147
Learn step, MVP process, 230
Learning
  failure as opportunity for, 651
  fostering culture of rapid, 566
  making world your classroom, 653
  MVP is meant for, 357
  planning agenda goals for, 329
Learning value
  constructing story map ribs, 380
  crafting iteration goal for, 449
  of story, 395
Leffingwell, Dean, 56–57
Legend, cause-effect tree, 161–162
Lens, as journey map component, 276, 281
LeSS. *see* Large Scale Scrum (LeSS) framework
Lifecycle
  across states of Kanban board, 476–479
  Agile analysis and planning. *see* Agile Analysis and Planning Map
  feature, 40–41
  Rational Unified Process (RUP), 49
Lightweight tools
  agile analysis vs. waterfall, 67
  for inter-team collaboration, 615–617
Lists, roles, and responsibilities table, stakeholder identification, 176
Loblaw, autonomous branding of, 646
Localized decision-making, 660–662
Long Lane, 73, 79
Longevity, product vision statement design, 173
Long-term agile planning
  capabilities for effective MVP implementation, 231–240
  epic planning, MVP, and overview, 224–225
  full potential plan, 225–228
  on the map, 222–223
  mapping to IIBA and PMI guides, 679
  objectives, 221
  overview of, 221
  planning interim periods, 241–248
  product roadmap, 240–241, 243–248
  summary of, 248
  validating assumptions using MVPs, 228–230
Loose coupling, DevOps, 560
Low-end disruptions
  creating purpose brand for, 646
  litmus test identifies, 643
  overview of, 641–642
Low-fidelity interface story maps, 387–388
Low-level integration tests, automated, 92
LRM. *see* Last responsible moment (LRM)

M
M&As (mergers and acquisitions), 228, 652
The Machine That Changed the World: The Story of Lean Production (Womack, Roos, and Jones), 628
Mainstream businesses
  adopting agile approach, 5–7
  resist experimentation, 651–652
  Scrum popular with, 44
  as source of information for personas, 267
Mainstream disruptions, 642, 643
Maintainability, defined, 35
Manifesto for Agile Software Development, 627
Manual tests, 91–93
Mapping
  agile analysis and planning. *see* Agile Analysis and Planning Map
  of book chapters to IIBA/ PMI guides, 677–681

goals to requirements, 36
MVPs and. *see* Minimum viable product (MVP) and story maps
Market
accelerating time to, 653
checklist for quarterly release retrospective, 697
differentiation, purpose alignment model, 88–90
practice of monitoring adjacent/low-end, 673–675
prioritizing technical risk versus, 214–215
testing MVPs directly in, 359
timing release of product to, 530–532
Marketing
emotional, 657
preparing team for, 103–104
Marketplace, Open Space events, 612
McDonald, Raynor
disruptive innovation and, 631
updates to disruptive model, 639–640
Measure step, MVP process, 230
Mechanisms, determining traceability, 130–131
Merge processes, process modeling to, 286
Mergers and acquisitions (M&As), 228, 652
Message flow, BPMN, 288, 293
Methods, estimation, 334–338
Metrics
actionable, 187–188
goals and objectives, 182–184
lean startup, 55
MVP process determining, 230
quarterly release retrospective, 541
specifying for interim periods, 243
split testing using funnel, 492–494
validating leap of faith hypotheses, 187–188
value stream maps, 283
Microfocus ALM Octane tool, 699
Milestone planning chart, 190, 710–711
Milestones
defined, 37
quarterly release retrospective, 542–543
specifying for interim periods, 243
Minimal quarterly plan, 344
Minimum marketable features (MMFs), 224, 365
Minimum marketable product (MMP)
impact of agile on productivity, 23
stakeholder productivity expectations and, 100
using MVPs to establish, 365
Minimum viable product (MVP)
begins with leap of faith hypotheses, 185–186
full potential plan, 228, 630
hypotheses in discovery-driven financial planning, 189
in lean startup, 55, 630
overview of, 224–225
process of, 229–230
provides PO with proposed features, 330
specifying metrics, 187
split testing, 491
testing for innovative products, 632
understanding, 229
validating long-term plan assumptions, 228–229
Minimum viable product (MVP) and story maps
backbone. *see* Backbone, constructing story map
complementing each other, 356
on the map, 354–355
MVP planning. *see* Minimum viable product (MVP) planning
objectives, 353, 356
overview of, 353
story maps. *see* Story maps
summary of, 388
Minimum viable product (MVP), capabilities
deferred vs. immediate deployment, 234–237
deployment and delivery approach, 232–234
narrow/deep versus wide/shallow, 237–240
overview of, 231
technical capabilities, 231–232
Minimum viable product (MVP) planning. *see also* Story maps
case study, creating MVP, 365–366
case study, Trint, 357–358
establishing MMP, 365
incrementally scaling, 364–365
iterative process, 363
MVP, defined, 357
the pivot, 363–364
summary of, 388
types of MVP, 359–363
venues for experiments, 358–359
Mining timeline, quarterly release retrospective, 543
Mission criticality, purpose alignment model, 88–90
Mission statements, product vision statements vs., 173–174
Mitchell, Dana, 333
Mitigate risk, 346
MMFs (minimum marketable features), 224, 365
Moments of truth
innovative development, 631–632
as journey map component, 278, 282
Monitoring progress
burndown versus burnup charts, 486–487
burnup charts, 486
cumulative flow diagrams, 487–490
daily burndown chart, 479–486
daily standup, 471–474
follow-up meeting, 474
updating developer task board, 475
updating Kanban board, 476–479
Monitoring system, product release and, 538

Motion, reducing though information radiators, 52
Multifunctional flowchart diagram, 287
Multiple Devices, Platforms pattern, 428
Multiple teams
  backlogs, 569
  scaled (quarterly) feature preparation, 602–605
  scaled iteration retrospective, 607–608
  scaled iteration retrospective follow-up, 608
Multiple Use-Case Scenarios pattern, splitting stories, 425
Multiple User Goals pattern, splitting stories, 426–427
Multiple User Roles pattern, splitting stories, 428–429
Must-haves, Kano grades, 206
Mutations, yielding outsize results, 639
MVP. *see* Minimum viable product (MVP)
MyChatBot example, 570–571, 668–669

N

Naming standards, stories, 395–396
Narrative, story map backbone, 372
Narrow and deep strategy, long-term feature implementation, 238–240
Natural decay of delight (and its opposite), Kano analysis, 208
Needs, innovative development and, 631–632
Negotiable, well-formed stories as, 421
Negotiation
  as skill of business analyst, 64
  of time estimates for developer tasks, 455
Netflix, 640, 648
Newell curve (cumulative flow diagrams), 487–490
New-market disruptions, 642, 643
Nexus, scaled agile framework, 582
NFRs. *see* Nonfunctional requirements (NFRs)
Nickolaisen, Niel, 88–90
Noble Inc., 363
No-estimating approach
  forecasting stories, 450–451
  quarterly and feature planning, 338
Non-colocated teams, scaling agility for, 617–619
Nonfunctional requirements (NFRs)
  completing product portrait, 217–220
  constraints checklist and, 689
  constructing story map ribs, 380–381
  defined, 34
  implementation pattern for user stories, 426
  operations infrastructure spike for, 419
  quarterly and feature planning, 341
  seeding product backlog, 216–217
  types of, 35
Noninnovative development, 5
Nonsolutionized, well formed stories as, 421
Normal scenario flow, 497

O

Objectives
  commitment to planning agenda, 329–330
  commitment to quarterly, 341
  crafting interim, 242
  crafting planning agenda, 329
  of estimation, 332
Objectory process, RUP, 49
Obsolete quarterly plans, 352
Octopus, distributed autonomous authority and, 661
Ohno, Taiichi, 50
Omotenashi, Kano grades, 206–207
One-dimensional features, Kano grades, 206
One-time experiments, governance changes as, 106
Open (public) beta testing, product release, 534
Open Space events, 611–614
Operational
  capacity, accelerating growth, 654
  cost, 215
  mission statement as, 173
  MVPs, 362
  sequence, story maps, 366–367
  value, of cross-functional teams, 668
  value streams, 283
Operations infrastructure spike (or story), 419
Opportunities (pain points, moments of truth)
  innovative development, 631–632
  as journey map component, 278, 282
Optimization
  agile fluency model, 108
  BA track record for, 20
  process modeling, 286
  value stream mapping for process, 284
Options thinking, Lean software development, 53
Ordering, product backlog refinement, 509, 512–513
Organization. *see* Scaling agile organization
Organizational preparation
  agile financial planning, 102–103
  channels and supply chains, 104
  customer-developer relationship, 101–102
  determine organizational readiness, 112–113
  of enterprise for agile development, 107–112
  governance and compliance, 104–106
  increased demand on resources, 106
  Initiation and planning, 86–87
  managing stakeholder expectations, 99–101
  on the map, 84–85
  mapping to IIBA and PMI guides, 678
  marketing and distribution teams, 103–104
  objectives, 83
  organizing development teams, 93–98
  overview of, 83
  preparing infrastructure, 90–93

purpose alignment model, 88–90
summary of, 113
time spent upfront on initiation and planning, 87–88
Organizational readiness, determining, 112–113
Otis Elevator Company, 647
Outcomes
agile corporate culture commitment to, 666
planning agenda goals for, 329–330
quarterly planning, 321
Outputs, commit to outcomes not, 666
Outside enterprise, collaborative relationships, 665
Outsize results, mutations yielding, 639
Outsourced infrastructure, technology investment in, 649
Overestimating signature, burndown charts, 485–486
Overextension, remedies to team, 455

P

Pain points
documenting personas for, 269
innovative development, 631–632
as journey map component, 278, 282
Pair programming, 17
Parity quadrant, purpose alignment model, 88–90
Partially done work, as waste, 51
Participants, feature preparation, 603
Partner quadrant, purpose alignment model, 88, 90
Patterns. *see* Splitting stories, patterns
Patton, Jeff, 366–367
PBIs. *see* Product backlog items (PBIs)
PC, as new-market disruption, 642
Performance features, Kano grades, 206
Persevere step, MVP process, 230
Persistent documentation
tracing analysis artifacts, 506–508
updating BA for stories, 496–506
use cases for, 50
Persisting stories, 496
Personas
analysis of, 264–265
case study, 270–271
creating, 267–268
documenting, 268–269
examples of, 266–267
fostering empathy using personalized marketing data, 658
history of, 265
if it feels too contrived, try a real user, 265
as journey map component, 276
story maps based on, 386–387
working with, 269
Perspective, agile corporate culture, 633
Perspective taking (cognitive empathy), 656, 658
Physical form, of product backlog items, 125
Physical representation, of features, 200
Physical stories, versus electronic, 403–404
PI (program increment), SAFe, 57, 582
Pierre Cardin, purpose brand quality, 645–646
Pivot step, MVP process, 230
Pivot-or-persevere meeting, 544–547, 632
Planning
agile financial, 102–103
art of. *see* Art of Agile analysis and planning
do not use story template when actively, 404–405
flow-based. *see* Flow-based planning
fundamentals. *see* Fundamentals of agile analysis and planning
Initiation and Planning zone. *see* Initiation and Planning zone
iteration and story. *see* Iteration and story planning
MVPs. *see* Minimum viable product (MVP) planning
preparation versus, 256
principles of, 323–325
quarterly and feature. *see* Quarterly and feature planning
timeboxed. *see* Timeboxed planning
use story template at end of, 405
value proposition. *see* Agile analysis and planning, value proposition
when to use flow-based vs. timeboxed approach, 555–557
Planning Game rules, 447, 455
The Planning Game, XP, 322–325
Planning Poker, 333, 338–340
PMI (Project Management Institute), mapping of book chapters to, 677–681
PMI guides, mapping book chapters to, 677–681
PMI Professional in Business Analysis (PMI-PBA), 17, 31
PO. *see* Product owner (PO)
POC. *see* Product owner council (POC)
Podeswa, Howard, 2–4
Podeswa, Yasha, xlv, 556
Podeswa, Yidel, 2
Point estimates, complexity versus effort, 335
Political intelligence, of business analysts, 64
Pools, BPMN private process model, 293
Pools, BPMN public process model, 288
Portfolio
checklist for quarterly release retrospective, 697
structure, scaling agile organization, 574–575
Postconditions, scaled feature preparation, 604
Practices, agile corporate culture, 634–635
Pre-alpha stage, product release, 533
Preconditions, scaled quarterly and feature planning, 586

Preorders MVP, 363
Preparation
  backlog refinement as, 47
  organizational. *see* Organizational preparation
  planning versus, 256
  process. *see* Process preparation
  quarterly and feature. *see* Quarterly and feature preparation
  for quarterly/feature planning event, 325–328
  story. *see* Story preparation
Principles, agile practices
  invest aggressively in enterprise agility, 648–650
  overview of, 634
  protect islands of innovation, 644–647
  tailor approach to circumstance, 635–643
Principles, Open Space event, 613
Priorities
  commitment to revise, 341
  conflicting, 620–621
Prioritizing features
  feature preparation walkthrough, 604–605
  managing stakeholder expectations, 99
  quarterly and feature planning agenda, 332
  as right of customer, not developer, 324
  sequencing epics and features in backlog, 212–215
  using personas to determine, 269
Private (closed) beta testing, 534, 652
Private process model, BPMN, 293–298
Pro forma operations, discovery-driven planning, 706–708
Problem or opportunity statement, 167–169
Problem-solving
  integration meeting, 599
  POC meeting, 602
  quarterly retrospective, 540
  sprint planning meeting, 597
Process
  agile innovative development, 631–632
  analysis, 8–9
  compliance after design of, 105
  extra, as waste, 52
  feature change initiatives as new, 254–255
  improvement tasks, 518
  improving with journey maps improving, 279
  innovative product development, 631–632
  scaling. *see* Scaling agile process
  setting parameters, 134
Process modeling
  business, 285–287
  discovering initial features, 198
  feature preparation via, 285–287
  product portrait for, 170–171
  selecting BPMN. *see* Business Process Model and Notation (BPMN)
Process preparation
  BA information artifacts and events, 122–123
  defining requirements types, 123–124
  determining process readiness, 145–146
  determining requirements granularity levels, 127–129
  on the map, 115–117
  mapping to IIBA and PMI guides, 679
  objectives, 115
  overview of, 122
  setting parameters, 134–136
  summary of, 146
  tailoring agile practice to context, 118–121
  tracing requirements/other configuration items, 129–133
  tuning the backlog, 124–127
  understanding, 118
  value stream mapping optimizing, 145
Product
  champion, 61, 580
  distribution, 643
  empathy in developing/improving, 658
  group, 577
  journey map for development investment, 278
  Lean development optimizes whole, 54
Product area
  job-based organization structure, 668–669
  scaling agile organization, 570–571
Product backlog items (PBIs)
  attributes, 125–126
  backlog refinement (preparation), 47
  cost of delay, 126
  definition of done (DoD), 45
  determining WSJF, 126–127
  matching with teams, 597
  physical form of, 125
  quarterly and feature planning estimation, 340
  readiness, 46
  requirements granularity levels, 127–129
  as requirements units, 38
  in rolling analysis, 469
  scaling, 566–569
  Scrum and, 45
  seeding. *see* Seeding the backlog
  setting up, 124–125
  specifying values for story attributes, 404
  sprint planning meeting, 597
  as story, 395
  story preparation, 394
  traceability, 130–133
  transparency, 46

Product backlog refinement
  as essential agile activity, 9
  preparation for, 47
  Scrum, 509
Product owner council (POC)
  BA responsibilities of product champion, 61
  composition of, 580
  frequency and timing, 601–602
  overview of, 579
  scaling agile process, 601
  waterfall approach to, 602
Product owner (PO)
  analyst acting as proxy, 557–558
  attends iteration retrospective, 518
  and BA, 45
  BA responsibilities of, 59–60
  as daily standup attendee, 473
  Grand Lane analysis and planning, 79–80
  insufficient business resources and, 621
  iteration goal proposal by, 449–450
  managing change during iteration, 495
  planning iteration implementation, 451
  product-level, 568
  questions for analyst to ask at Triad event, 402
  responsibility for user stories, 398–403
  role in scaling agile organization, 571–572
  writing acceptance criteria, 408–409
Product portrait, 169–171, 217–220
Product release. *see* Releasing the product
Product roadmap
  constructing implementation plan, 343–345
  creating, 243–248
  long-term planning with, 240–241
  planning agenda for long-term, 329
  planning interim periods, 241–243
  planning shorter horizons, 248
Product vision statement
  case study, 174–175
  crafting, 172–173
  defined, 32
  mission statement vs., 173–174
  translate into goals and objectives, 181–184
Product visioning. *see* Visioning
Production environment, MVP, 232
Production process, business model disruption in, 642
Productivity
  diagnosing with cumulative flow diagrams, 489–490
  impact of agile on, 23
  managing stakeholder expectations, 100–101
  quarterly release retrospective checklist, 695–696
Product-level PO, 568, 571–572
Program
  epics, 575
  quarterly release retrospective checklist, 697
  SAFe, 57
  structure, scaling agile organization, 572–575
Program board, 588–589, 592–595
Program increment (PI), SAFe, 57, 582
Progress
  accounting for at end of iteration, 513
  check, POC meeting, 602
  monitoring team. *see* Monitoring progress
Project Management Institute (PMI), mapping of book chapters to, 677–681
Projects, not used by high-level organizations, 575
Promise, quarterly plan sometimes is a, 321
Proof of concept, technical research spike to create, 419
Prototypes, Kano analysis and, 204
Provisioning, automated DevOps, 562
Proxy PO, 572, 580
Pruning, product backlog refinement, 509, 512–513
Public process model, BPMN, 288–292
Pull systems, Lean software development, 53
Purpose alignment model, 88–90, 213
Purpose brand, low-end disruptions, 646–647

Q

QA subgroup, scaling agile organization, 578
Quadrants, purpose alignment model, 88–90
Qualifiers, adding to spike's AC to avoid waterfall, 417
Quality
  agile prioritizing, 566
  DevOps practice of built-in, 561
  everyone is responsible for, 561
  hardening iterations for, 532
  requirements, 34
  of transparency, 667
Quality assurance (QA)
  acceptance test–driven development and, 56
  as extended team member, 97
  preparing first stories in backlog, 77–78
  quarterly release retrospective and, 78, 696
Quarterly (release) roadmap
  case study, creating, 348–350
  with dependencies, 346–347
  implementing, 344
  on the map, 316
Quarterly and feature planning
  checklist of deliverables, 694
  checklist of inputs, 693
  commitment, 341–348
  flow-based feature planning, 318
  on the map, 316–317
  mapping to IIBA and PMI guides, 680
  objectives, 315, 317
  overview of, 315

preparing for planning event, 325–328
quarterly planning overview, 318
quarterly planning timing, 325
quarterly planning versus flow-based feature, 319–320
quarterly planning, when advised/not advised, 319
quarterly planning, with agility, 320–322
retrospective, 348–350
reviewing once underway, 351–352
summary of, 352
timeboxing pros and cons in, 556–557
topics (agenda), 328–341
XP's planning game guidelines, 322–325
Quarterly and feature planning, scaled
checklist of attendees, 587, 698
creating program board, case study, 592–595
facilitation guidelines, 589
inputs and deliverables, 588
objectives, 586
overview of, 587
preconditions for, 586–587
program board, 588–589
timing considerations, 586
topics/agenda, 589–592
Quarterly and feature preparation
activities, 256–257
architecture review, 307–312
assessing readiness, 258
benefits of feature preparation, 256
BPMN. *see* Business Process Model and Notation (BPMN)
business process modeling, 285–287
context analysis, 263–264
developer tasks and functional spikes, 258–259
feature definition of ready (DoR), 258
journey mapping. *see* Journey mapping
map of, 252–253
mapping to IIBA and PMI guides, 680
objectives, 251
overview of, 251
overview of features, 254–256
persona analysis, 264–271
process modeling, 285–287
in rolling analysis, 469
specification of feature acceptance criteria, 259–263
stakeholder analysis, 264
timing of activities, 257–258
use-case modeling, 298–299
user-role modeling workshops, 300–306
value stream mapping, 283–285
Quarterly backlog, 327
Quarterly Closeout (Epic, Feature Closeout) zone
defined, 73
Grand Lane analysis and planning, 81
product release. *see* Releasing the product
scaling agility. *see* Scaling agility
Short Lane analysis and planning, 78
Quarterly feature retrospective, scaled, 609–611
Quarterly Inception/Feature Inception zone
defined, 69
MVPs/story maps. *see* Minimum viable product (MVP) and story maps
overview of, 72–73
quarterly and feature planning, 318
Short Lane analysis and planning, 78
Quarterly planning
attendee checklist, 692
flow-based feature planning versus, 319–320
readiness checklist, 690–691
rules of thumb, 682
scaled, 80
Quarterly release retrospective
checklist of questions, 695–697
guidelines, 539–542
overview of, 539
preparing timeline, 542–543
recommendations, 544
scaled, 609–611
walkthrough, 543–544
Quarterly/release burndown chart, 516–517
Quarterly/Release Planning Game, XP, 322–325
Questionable features, Kano grades, 207
Questionnaire, Kano analysis, 204
Questions
anyone can ask, 325
business analysts not afraid to ask, 65
in Kano analysis, 202–203
quarterly release retrospective checklist, 695–697
Quick wins, non-colocated teams, 618
Quiz, spotting story-splitting patterns, 431–433

R
R&D (Research and Development), for disruptive services, 648
Rapid learning culture, 566
Rational Team Concert (RTC) tool, 700
Rational Unified Process (RUP)
as agile framework, 49
history of agile development, 17, 18
risk prioritization and, 214–215
use-case model, 298
Raynor, Michael, 631, 639–640
RC (release candidate) stage, product release, 532
Readiness
assessment, features, 258
determining process, 145–146
quarterly planning checklist for, 690–691
Scrum, 46

Readiness checklist
  features, 258
  product visioning, 152
  quarterly planning, 690–691
  visioning, 686
Real Options, agile financial planning, 675
Real-time estimation
  case for, 336
  versus real time/IDD estimates, 335–336
  story points vs., 10
Real-world outcomes, goal and objective metrics, 182–183
Real-world testing, beta testing as, 359
Recommendations, quarterly release retrospective, 543
Recoverability, defined, 35
Recovery actions, developer tasks, 471
Reengineering
  process modeling for process, 286
  value stream mapping for process, 284
Refactoring software, costs of, 118
Refinement
  of estimate over time, 334
  of estimation units for scope, 334–338
  of feature and acceptance criteria, 327–328
  product backlog. *see* Product backlog refinement
Regions
  cumulative flow diagram, 488
  story map, 370
Regulated sectors, agile in heavily, 629
Reinforced, well formed stories as, 421
Reinsertsen, Donald, 651
Relationship
  dependencies, story maps, 370
  persisting between artifacts, 506–508
Release candidate (RC) stage, 532
Release date, in planning agenda, 330–331
Release management
  defined, 581
  team, 581
Release Planning Game, XP, 322–325
Release to manufacturing/market (RTM) stage, 534
Releasing the product
  getting stories to done, 530
  on the map, 528–529
  mapping to IIBA and PMI guides, 681
  to market, deferring, 531
  to market, hardening iterations, 531–532
  to market, timing considerations, 530–531
  objectives, 527
  overview of, 527
  pivot-or-persevere meeting for, 544–547
  quarterly release retrospective, 539–543
  quarterly retrospective walkthrough for, 543–544
  staging, 532–539
  summary of, 547–548
  to users/user representatives prior to, 531
Reliability, defined, 35
Requirements
  alpha testing for gaps in, 533
  BAs for business, 21
  business analysts provide leadership for, 62–63
  communications plan for change in, 112
  costs of agile development and, 118
  defining types of, 123–124
  determining granularity levels for, 127–129
  early BA focus on, 17
  functional versus non-functional, 380–381
  Kano grades for variable, 206
  management tools, 132, 615, 699–700
  managing long-term planning, 224
  managing stakeholder expectations about delayed, 99
  persisting, 496
  tracing, 129–133
  uncertainty pattern and, 427–428
  welcoming change, 29
  why agile should include BA competency, 24–25
Requirements-related terminology
  acceptance criteria, 40
  assumptions, 36–37
  business goal, 32–33
  business objective, 33
  business requirements, 33
  business rule, 37
  constraint, 37
  epics, 38–39
  from epics to features and stories, 38
  feature lifecycle, 40–41
  features, 39
  functional requirements (FRs), 34
  importance and non-importance of, 36
  milestone, 37
  nonfunctional requirements (NFRs), 34–35
  product vision statement, 32
  requirements, defined, 33
  requirements units, 38, 66
  solution requirements, 34
  stakeholder requirements, 34
  stories, 39–40
  story estimation, 40
  themes, 40
  trace goals to requirements, 36
  transition requirements, 35
  user requirements, 34
Research and Development (R&D), for disruptive services, 648
Research users, and personas, 267

Resources, increased demand for, 106
Resources and checklists, additional
  agile requirements management tools, 699–700
  checklist of invitees for quarterly planning, 692
  checklist of invitees for scaled quarterly and feature planning, 698
  checklist of quarterly and feature planning deliverables, 694
  checklist of quarterly and feature planning inputs, 693
  checklist of quarterly release retrospective questions, 695–697
  facilitation tips, 684–685
  mapping of book chapters to IIBA/ PMI guides, 677–681
  NFRs and constraints checklist, 689
  readiness checklist for quarterly planning, 690–691
  rules of thumb in agile analysis and planning, 682–683
  stakeholder checklist, 687–688
  visioning readiness checklist, 686
Responsibility, for user stories, 398–403
Responsible procrastination, 659
Retrospective
  planning, 347
  quarterly release. *see* Quarterly release retrospective
  in rolling analysis, 469
  scaled iteration, 607–609
  scaled quarterly/feature, 609–611
Reveal, Delphi estimation, 339
Revenue generation (business value), story map ribs, 381
Revenue streams, business model disruption in, 643
Reverse features, Kano grades, 207
Reverse income statement, case study, 705–706, 709–710
Ribs, constructing story map
  dependencies, 381
  frequency, 381
  implementation sequence, 379–380
  "juicy bits" first, 381
  learning value, 380
  overview of, 370, 379
  revenue generation, 381
  technological risk, 380–381
  timeline view, 383–384
  user task view, 382
  WSJF and cost of delay, 380
Ries, Eric, 54–55, 357
Risk Reduction and Opportunity Enablement Value (RR&OE)
  constructing story map ribs, 379–381
  cost of delay, 212–213
Risks
  full-potential plan for internal, 228
  gravity of past success and, 645
  interim goals for, 383
  managing dependencies and, 345–347
  market prioritization versus technical, 214–215
  upfront planning dependent on, 87
Roadmap. *see* Product roadmap
Roamers, inter-team collaboration, 616–617
Role-Feature-Reason template. *see* Connextra template
Roles
  backlog seeding, 197
  scaling PO, 571–572
  splitting story with multiple user, 428–429
  user-role models. *see* User-role modeling workshop
  XP Quarterly/Release Planning Game, 323
Roles, business analyst and agile
  agile analysis vs. waterfall, 65
  agile team analyst, 60
  business systems analyst (BSA), 63
  coach, 61
  dedicated business analysts, 61–62
  overview of, 58–59
  product champion (director), 61
  product owner (PO), 59–60
  proxy user, 60–61
  requirements leadership, 62–63
Rolling analysis and preparation (day-to day-activities), monitoring progress
  burndown versus burnup charts, 486–487
  burnup charts, 486
  cumulative flow diagrams, 487–490
  daily burndown chart, 479–486
  daily standup, 471–474
  follow-up meeting, 474
  monitoring progress. *see* Monitoring progress
  updating developer task board, 475
  updating Kanban board, 476–479
Rolling analysis and preparation (day-to-day activities)
  accounting for progress at end, 513
  actions against developer tasks, 471
  analysis of upcoming epics, features, and stories, 509–512
  analysis tasks, 470
  day in the life of agile analysis, 468–469
  introduction to, 465
  iteration retrospective, 517–524
  iteration review, 514–517
  managing scope change in iteration, 495
  on the map, 466–467
  mapping to IIBA and PMI guides, 681
  objectives, 465, 468

other analysis documentation, 506–508
overview of, 468
story testing and inspection, 491–494
summary of, 524
Triad meetings, 470
updating documentation. *see* Business analysis (BA) documentation, updating
updating task progress, 470
Rolling lookahead meeting, 462–463
Root-cause analysis
cause-effect diagrams, 157–161
cause-effect tree, 161–166
choosing right tool, 162
Five Whys method, 153–157
at a glance, 153
overview of approach, 152–153
problem or opportunity statement, 167–169
in visioning process, 147
Rose, Willy, 6
RR&OE (Risk Reduction and Opportunity Enablement Value)
constructing story map ribs, 379–381
cost of delay, 212–213
RTC (Rational Team Concert) tool, 700
RTM (release to manufacturing/market) stage, 534
Rules, iteration planning, 447
Rules of thumb
agile analysis and planning, 682–683
agile business analysis, 68
RUP. *see* Rational Unified Process (RUP)
Ryanair, as example of pivot to established product, 364

S
Sacrificial product, deploying to, 235
SAFe. *see* Scaled Agile Framework (SAFe)
Safe spaces, for bad news, 653
Sailboat (or speedboat) game
iteration retrospectives, 520–523
quarterly release retrospective, 544
virtual iteration retrospective, 609
Satisfaction versus fulfillment graph, Kano analysis, 207–208
Satisfiers, Kano grades, 206
Scalabiilty, 35, 649
Scaled (quarterly) feature preparation, 602–605
Scaled Agile Framework (SAFe)
as agile framework, 56–57
history of agile development, 18
overview of, 582
PO Sync in, 601
process analysis via, 9
Scrum of Scrums meetings in, 600–601
terms, 57
and this book, 57
timeboxed planning, 42–43, 121, 555
timing of feature preparation, 257
Scaled iteration
retrospective, 607–609
review, 606–607
scaling agile process, 595
sprint planning meetings, 595–597
Scaled quarterly and feature planning. *see* Quarterly and feature planning, scaled
Scaled quarterly and feature retrospective, 609–611
Scaled quarterly planning, 80
Scaling agile organization
competency groups, 577–578
component teams, 577
extended teams, 576
forming feature teams, 575
organizing teams, 79–80
overview of, 570
PO role in, 571–572
portfolio and program structure, 572–575
product owner council (POC), 579–580
release management team, 581
by subproduct and product area, 570–571
user task force, 581
Scaling agile process
big room iteration planning, 598–599
daily standup, 600
feature preview, 599
initial preparation, 585–586
integration meetings, 599
Open Space event, 611–614
overview of, 581
plan implementation (team level), 597–598
product owner council (POC) meeting, 601–602
scaled (quarterly) feature preparation, 602–605
scaled activities and events, 583–585
scaled agile frameworks, 582–583
scaled iteration (or feature) review, 606–607
scaled iteration retrospective, 607–609
scaled iteration (sprint) planning meetings, 595–598
scaled quarterly and feature planning, 586–595
scaled quarterly/feature retrospective, 609–611
Scrum of Scrums (SOS) meetings, 600–601
team-level story preparation, 605–606
Triad meetings, 614
user task force meetings, 606
Scaling agility
culture, 564–566
interdependency of scaled agile teams, 553–554
inter-team collaboration, light-weight tools, 615–617
inter-team collaboration, planning, 554–558

introduction, 549
iteration planning, 462
on the map, 550–551
mapping to IIBA and PMI guides, 681
MVPs incrementally, 364–365
overview of, 552
potential issues and challenges, 617–621
product backlog, 566–569
reasons for, 552–553
requirements management software tools, 615
summary of, 622
Scaling agility, continuous delivery (CD)
ATDD and BDD, 563–564
automation in test-build-deploy steps, 558–559
CD and CI, 561–562
DevOps practices, 559–562
overview of, 558
Schwaber, Ken, 44
Scope
commitment to forecasting, 341–342
estimation units refine, 334
flow-based feature planning and, 318
forecasting iteration, 447–451
iteration, 446
as journey map component, 274
managing change during iteration, 495
planning iteration implementation, 451–452
Scope line, daily burndown chart, 480
Scouts, 616, 619
Scrum
as agile framework, 44–45
BA competency and, 45
backlog refinement, 47
daily standup (or scrum), 47–48
definition of done (DoD), 46
history of agile development, 17
iterations known as sprints in, 444
product backlog items (PBIs), 38, 45
product backlog refinement, 509–512
product owner (PO) and BA, 45
readiness, 46
ScrumMaster and BA, 46
sprint, 45
sprint (iteration) planning, 47
sprint goal, 47
sprint review and retrospective, 48
timeboxed planning in, 42–43, 121, 555
transparency, 46
using timeboxed planning. *see* Timeboxed planning
The Scrum Guide, 9
Scrum Guide, 44
Scrum of Scrums (SoS) meetings, 600–601
ScrumMaster, 46, 60
Security, defined, 35
Seeding the backlog
analyzing NFRs/constraints, 216–220
attendees, 197
circumstance-based market segmentation for, 75–77, 198
epics and stories, 196
feature attributes, 201–202
feature independence, 199
features to seed upfront, 196–197
Kano analysis, 202–212
on the map, 194–196
mapping to IIBA and PMI guides, 679
objectives, 193
other ways to discover initial features, 198–199
overview of, 193
physical representation of features, 200–201
sequencing epics and features in backlog, 212–215
specifying emergent features, 200
summary of, 220
template for epics and features, 199
writing feature acceptance criteria, 215–216
Seeding the Backlog zone
activities in, 196–197
defined, 72
seeding the backlog, 76
Seeing waste, Lean software development, 52
Segmentation, circumstance-based market, 630–631
Self-organizing
daily standup as, 472
defined, 95
twelve principles for BAs, 30
Self-reflection, twelve principles for BAs, 30
Self-sufficiency, development team, 94–95
Send event, BPMN private process model, 295
Senior development manager, POC, 580
Senior product manager, POC, 580
Sequence flows, BPMN, 288, 293
Sequencing, epics and features in backlog, 212
Servant leadership, agile, 564–565
Servant Leadership (Greenleaf), 564–565
Service delivery, business model disruption in, 643
Service-level requirements (SLRs), 34
Seven wastes, Lean software development, 51–52
70/20/10 rule, change, 653
Shared
components, team dependencies, 554
team members, 617
well-formed AC should be, 412
Shift left, DevOps, 560
Shillace, Sam, 360
Shingo, Shigeo, 50
Short Lane, 73, 74–78
Shorter horizons, product roadmap for, 248
Short-term wins, accelerating change via, 110

Showstopper errors, alpha testing for, 533
Sign up, developer task, 454
Signoffs, iteration review, 514
Silent estimation, Delphi, 339
Silos, busting
  business can lead technical teams, 668
  collaborative culture and, 559, 566
  communities of practice (guilds) for, 669–672
  cross-functional teams organized around value, 668
  everyone works for the business, 667
  job-based organization, 668–669
  overview of, 667
Simplicity, as Lean thinking principle, 30, 51
Single source of truth, product backlog as, 124–125
Site visits, for non-colocated teams, 618
Size
  determining item, in Kanban, 143
  development team should be small, 95
  DevOps practice of small batch, 561
  diagnosing stories that are too big, 490
  splitting stories into patterns. *see* Splitting stories, patterns
  taxonomy of story, 395–396
  varying product backlog, 125
  well-formed stories have small, 421
SLRs (service-level requirements), 34
SMART, 330
SMEs (subject matter experts), attending Triad meetings, 402–403
Smoke tests, analyze-code-build-test cycle, 492
Smoke-and-Mirrors MVPs (or Swivel Chairs), 360–361
Soft skills, agile business analysts, 63–66
Software
  core value of Agile Manifesto, 28–29
  costs of refactoring, 118
  delivering frequently, 29–30
Solitude, developing collaborative culture in, 665
Solution requirements, 34, 123
Spanning application, 361, 383–385
Specification
  by example, 66, 409–410
  updating use-case, 503–506
Spike card, story map, 368–369
Spikes
  functional. *see* Functional spikes
  technical, 418–419
Spikes, SAFe
  feature preparation, 258–259
  process analysis, 9
  quarterly and feature planning estimates, 340
  story preparation, 415–416
Split (A/B) testing
  actionable metrics, 187
  staging release, 539
  value validation, 491–494
Splitting stories, patterns
  Business Rules pattern, 424
  Complex UI pattern, 426
  Data Complexity pattern, 425
  how to use, 422
  Integration Capabilities pattern, 428
  Large Initial Effort pattern, 424
  Multiple Devices, Platforms pattern, 428
  Multiple Use-Case Scenarios pattern, 425
  Multiple User Goals pattern, 426–427
  Multiple User Roles pattern, 428–429
  NFR Implementation pattern, 426
  overview of, 422
  quiz, 431–433
  tie-breaker rules, 422–423
  Too Many Acceptance Criteria pattern, 429–430
  Uncertainty pattern, 427
  Workflow Steps pattern, 423–424
Splitting stories, rules of thumb for estimating, 682–683
Spoken needs, Kano grades, 206
Spreadsheets, traceability, 132
Sprint
  backlog, 47
  goals, 47, 383–387
  planning. *see* Iteration planning
  review, 48, 606
Sprints, Scrum, 44, 45, 444
Stabilization (or IP) iteration, quarterly planning, 325
Stabilizing (hardening) iterations, 531–532
Stages, product release
  alpha testing, 533
  analysis documentation, 538
  beta testing, 533–534
  closed (private) beta testing, 534
  general availability, 535–539
  general availability checklist, 535–538
  as journey map component, 276–277, 281–282
  monitoring, 538
  open (public) beta testing, 534
  overview of, 532–533
  pre-alpha, 533
  release candidate (RC), 534
  release to manufacturing/market (RTM), 534
  value validation, 539
Staging environment, MVP, 231–232
Stakeholder
  agile analysis vs. waterfall, 67
  agile impacts satisfaction of, 23

attending backlog seeding, 197
cause-effect diagram tips for, 157–158
checklist, 687–688
feature preparation, 264
identifying, 152
requirements, 34, 123
visioning of, 147, 150
visioning process of, 147
Stakeholder analysis and engagement
collaboration plan, 176–178
communication plan, 178–179
identify via checklist, 176
list, roles, and responsibilities table, 176
ongoing engagement and analysis, 179–181
overview of, 175–181
Standard operating procedures, process modeling, 286
Standards, business analysis, 30–31
Standish Group, 18–20, 22–23
Start event, BPMN, 288, 293
Startups, must accelerate growth, 654
State-transition diagram, updating Kanban board, 476–479
Statistical group response, Delphi estimation, 339
Status updates, daily standup, 473–474
Steering phase, XP Quarterly/Release Planning Game, 322
Steps outlined level, requirements granularity, 128
Stories
acceptance criteria, 40
acceptance template, 40
actions against developer, 471
avoiding gating, 142
business prioritizing, 324
communicating via, 7
continuous basis activities to get product done, 530
daily burndown chart, 481–482
definition of, 39, 196, 395
development estimates, 324
from epics to features to, 38
estimating, 40
estimating/splitting, 682–683
features are bigger than, 254
goals and objectives are represented in, 183–184
grouping into themes, 40
incomplete, 513
Kanban board setup for iteration planning, 458–462
long-term planning requirements, 224
mapping. *see* Minimum viable product (MVP) and story maps
measuring progress on burnup chart, 486
ongoing analysis of upcoming, 509–512
persisting requirements, 496
preparing for scaled initiative, 585
requirements-related terminology for, 39–40
scaling backlog, 567
testing and inspection, 490–494
themes, 40
tracking developer tasks on burndown versus, 482
tuning, 138–142
updating BA documentation, 496–506
updating developer task board, 475
updating Kanban board, 476–479
use case vs. user, 50
user story, 39–40
writing, 323
as XP contribution to agile, 17
XP functional units as, 38
Stormboard, brainstorming tool, 700
Story maps
anatomy of, 368–370
benefits of, 367
case study, creating backbone, 375–379
case study, stories for MVP, 384–386
completing, 438–440
constructing ribs, 379–384
defined, 366
dependency relationships, 370–375
Jeff Patton's, 366–367
journey maps build, 279
MVPs and, 353–356
MVPs complement, 356
other forms of, 386–388
summary of, 388
Story planning. *see* Iteration and story planning
Story point estimation
case for, 336–337
as estimation unit, 335
Fibonacci sequence for, 337–338
measuring complexity versus effort via, 335
versus real time/IDDs estimates, 335–336
Story points
estimating functional spikes via, 418
estimating other kinds of stories, 340–341
real-time estimates vs., 10
story estimation using, 40
Story preparation
alternative terminology, 395
analyzing business rules/AC, 433–438
case study, complete story map, 438–440
definition of story, 395
introduction to, 391
map of, 392–393
mapping to IIBA and PMI guides, 680
naming standards, 396–397
objectives, 391, 394

overview of, 394
physical versus electronic stories, 403–404
product backlog refinement, 510–512
responsibility for user stories, 398–403
size taxonomy, 395–396
splitting. *see* Splitting stories, patterns
stories that are not user stories, 414–420
story acceptance criteria, 407–414
summary of, 440
team-level, 605–606
Three Cs of stories, 397–398
user story examples, 397
values for story attributes, 404
writing high-quality stories, 420–421
writing story description, 404–407
Story telling, Delphi estimation, 339
Strategic initiatives, accelerating change, 109
Strengthening, agile fluency model, 108
Subconscious requirement, Kano grades, 206–207
Subject matter experts (SMEs), attending Triad meetings, 402–403
Submit claim
developer task board and, 453, 475
tracing analysis artifacts, 506–508
updating use-case specifications, 503–504, 506
Subproducts, 568, 570–571
Success, agile financial planning and, 103
Supplementary requirements, 34
Supply chains, preparing, 104
Survey, Kano analysis, 204
Sustain acceleration, for change, 110
Sustaining innovation, 637, 644
Sutherland, Jeff, 44
Swarm, developer tasks, 471
Swimlanes, process modeling with/without, 287
Swimlane-workflow, 287
Swivel Chairs (Smoke-and-Mirrors MVPs), 360–361
Symbol set, BPMN standard, 287, 295
Systems analysts, many BAs were, 18

T

Tables, product portrait, 170–171
Targeted features, versus committed, 343
Targets
full potential plan defines bold, 225–226, 630
selecting features in Kano analysis, 202
trade-off of costs and benefits, 119–121
visioning as essential, 151
Task switching, as waste, 52
Tasks
actions against developer, 471
analysis, 470
developer task board in iteration planning, 446
feature preparation, 258–259
identifying developer, 452–456
planning iteration implementation, 451–452
product backlog refinement, 509
updating progress of, 470
TDD. *see* Test-driven development (TDD)
Team PO, 572, 580
Teams
cumulative flow diagrams, 608
feature review, 607
interdependency of scaled agile, 553–554
inter-team collaboration via, 616
matching present backlog items with, 597
role of agile analyst, 60
in SAFe, 582
scaling agility for non-colocated, 617–619
self-organizing, 30
story preparation, 605–606
working with waterfall, 619–620
Technical benefits, of story, 395
Technical capabilities, MVP implementation, 231–232
Technical constraints, hardening iterations, 531
Technical debt
balancing user features and, 341
defined, 118
prioritize new development, and payment of, 215
technical debt-payment spike, 419
Technical limitations, deferred deployment due to, 236
Technical research spike (or story), 419
Technical risk, 214–215
Technical spikes (or stories)
business benefits, 324
quarterly and feature planning estimates, 341
types of, 418–419
Technical teams, 668
Technological risk, constructing story map ribs, 380
Technology
invest aggressively in enterprise agility, 648–650
quarterly release retrospective, 695
uncertainty pattern regarding, 427–428
Template
BDD. *see* Behavior-driven development (BDD)
Given-When-Then. *see* Behavior-driven development (BDD)
information persona, 268–269
journey map, 275
minimal quarterly plan, 344
product portrait, 170–171
product roadmap, 241
quarterly roadmap, 344
Role-Feature-Reason. *see* Connextra template
story map, 369
user story, 39–40

Tentative acceptance, feature preparation, 605
Test pyramid, 92–93
Test-build-deploy steps, automation in, 558–559
Test-driven development (TDD)
  as agile framework, 56
  ATDD and BDD, 563–564
  continuous development and, 562–563
  history of agile development, 18
Testing. *see also* Test-driven development (TDD)
  alpha, 359, 533
  beta. *see* Beta testing
  as continuous and automated, 562
  developer task, 471
  focus on compliance goals, 105
  inviting representative to Triad meetings, 401
  Lean software development, 53–54
  MVP, 358–359
  of planning assumptions, 190
  preparing infrastructure for, 90–93
  questions for analyst to ask at Triad event, 403
  in rolling analysis, 469
  split (or A/B), 187–188
  story inspection and, 490–494
  well-formed AC and, 412
  well-formed stories and, 421
Themes, 40, 183
Thoughts, as journey map component, 278, 282
Three Cs of stories, 397–398
Time
  estimates for developer tasks, 454–455
  spent on estimation, 332
  spent on product backlog refinement, 509
Time criticality, 126, 380–381
Time-and-materials contract, 120
Time-based estimates, 335
Timeboxed planning
  feature preparation timing, 257
  feature review via, 607
  flow-based planning versus, 555
  flow-based vs., 121
  frameworks supporting, 121
  for frontend, 556–557
  history of agile development, 17
  increments in, 446–447
  iteration implementation, 451
  iteration planning, 444
  Kanban board columns, 459–462
  overview of, 42–43
  quarterly planning for, 319–320
  requirements granularity levels, 127
  in rolling analysis, 469, 509
  Scrum, 44
  setting process cadence, 134
  setting WIP limits, 144
  starting iteration, 351
Timeline
  constructing story map ribs, 383–384
  interim, 242
  quarterly release retrospective, 542–543
  updating in quarterly release, 543
Timing
  daily standup considerations, 472
  of feature preparation, 257–258, 510
  of feature preview meeting, 462–463
  functional spikes and, 416
  iteration retrospective, 517
  Open Space events, 612
  pivot-or-persevere meetings, 545
  POC meeting, 601–602
  as quality of transparency, 667
  quarterly planning, 325
  releasing product to market, 530–532
  scaled feature preparation, 603
  scaled quarterly and feature planning, 586
  story preparation, 511–512
  of Triad meetings as story develops, 400
Titles, generic job, 664
Too Many Acceptance Criteria pattern, splitting stories, 429–430
Tools
  agile analysis vs. waterfall, 67
  agile requirements management, 699–700
  Lean software development, 52–54
  lightweight, 67, 615–617
  traceability, 130–131
Topics/agenda
  iteration retrospective, 518–520
  iteration review, 515–516
  Open Space events, 613–614
  quarterly and feature planning, 328–331
  quarterly and feature planning, scaled, 589–592
  sprint planning meeting, 596
Top-level product, only one, 568
Total potential person-days, forecasting capacity, 448–449
Touchpoints
  BPMN private process model, 296
  BPMN public process model, 288, 291
  journey map, 278, 282
Toyota Production System, lean software development, 628
Tracing, 129–133, 506–508
Track record, of business analysis, 19–23
Transfer risk, 346
Transition
  phase of RUP lifecycle, 49

requirements, 35, 123
timeline, 111
Transparency
agile corporate culture practice of, 666–667
Scrum control through, 46
siloing versus, 667
Triad meetings
analyst guidelines for, 402–403
analyzing AC during, 262
attendees, 401–402
backlog preparation via, 400–403
benefits of, 400
inputs and deliverables, 401
in rolling analysis, 470
scaling agile, 614
story preparation for, 605–606
timing considerations, 400
Trint
case study, MVP, 357–358
as Differentiator MVP, 360
as disruptive innovation, 638
as mainstream disruption, 642
Trunk-based development, DevOps, 562
T-shirt sizes, as estimation units, 334
Twelve principles, 29–31
12 Principles behind the Manifesto, agile, 18
Two-hands rule, daily standups, 473

U

UAT. *see* User acceptance testing (UAT)
Uber, as disruptor, 640–641
UML
and BPMN, 57–58
communication diagrams, 308
UML for the IT Business Analyst (Podeswa), 6
Unambiguous, well formed story as, 421
Uncertainty pattern, splitting stories, 427
Underestimating signature, burndown charts, 484–485
Undesirable effects (UDEs)
BLInK improved outcomes for, 166
cause-effect trees, 162–165
problem statement, 168–169
product vision statement and, 174–175
Unit tests, most automated tests should be, 92
Units, estimation, 334–338
Updates
acceptance criteria, 409
BA documentation, 496–506
daily standup status, 473–474
developer task board, 475
iteration goal and scope, 448
Kanban board setup, 476–479
task progress in rolling analysis, 470, 471
timeline, in quarterly release retrospective, 543
Upfront planning, 87–88
Upward (backward) traceability, 130, 506
Urgency
accelerating change, 109
of agile corporate culture, 633
Usability
alpha testing for, 533
defined, 35
Use cases
as agile framework, 49–50
discovering initial features, 198–199
implementing change initiative via, 255–256
job-based organization as high-level, 668–669
modeling, 298–299
Multiple Use-Case Scenarios pattern, 425
narrative, 50
scenario, 50
slice, 395
tracing analysis artifacts, 506–508
versus user stories, 50
Use-case brief, 505
Use-case model, updating BA documentation
capturing artifacts, 499–501
case study, 501–502
hybrid approach, 498
overview of, 497
updating use-case specifications, 503–506
use-case-only approach, 498
Use-case only approach, 498
Use-case slice, 50, 497, 498
Use-case specifications
updating, 503–506
updating use-case model, 497
use-case only approach, 498
User acceptance testing (UAT)
in analyze-code-build-test cycle, 490–491
as continuous and automated, 562
do not skip BA in agile development, 8–9
lifecycle across states of Kanban board, 479
specifying feature AC, 259
specifying using Gherkin feature files, 263
test pyramid, 92
validating value with, 493
User feedback, stakeholder expectations and, 100
User interface design, 269, 426
User involvement, BA track record for, 20
User journeys, product portrait for, 170–171
User proxy, 581
User requirements, 34, 123
User stories
analyst value added to, 399–400

decision table example, 435–436
defined, 39–40, 395
examples of, 396
implementing change initiative as, 255
responsibility for, 398
stories that are not, 414–416
templates, 39–40
Triad approach, 400–403
updating use-case model with, 498
using personas, 269
writers of, 398–399
User story card, story map, 368–369
User tasks, 368–369, 374, 581, 606
User value
crafting iteration goal, 449
of story, 395
story mapping, 381
User-role modeling workshop
agenda, 300
case study, 304–306
consolidate user roles, 302
overview of, 300
refine user roles, 303
user roles, 300–301
User-task view, story mapping, 382
UX designers, POC, 580

V

Value
assessing backlog items, 193
business analyst adds to iteration goal, 450
in cost of delay, 126
delivered by iteration goal, 449
focusing estimates on, 334
hypotheses, 186, 188
organize cross-functional teams around, 668
organize teams around, 95–96
specifying story attributes, 402
story is work item that delivers, 39, 395
validation, 493, 539
well-formed stories deliver, 421
Value proposition. *see also* Agile analysis and planning, value proposition
as decisions maximizing business value, 7
for organizations with no agile experience, 110
Value stream analysis, 658
Value stream, defined, 283
Value stream mapping
business process models versus, 285
feature change initiative delivered as, 255
feature preparation via, 283–285
Lean software development, 53
optimizing process, 145
overview of, 283–285
Value Stream Skeleton MVPs, 361–362
Vanity metrics, 187
Variable requirements, Kano grades, 206
Velocity
capacity, in planning agenda, 331
revising in quarterly plan, 351–352
Vertical slices of functionality, agile vs. waterfall, 67
Virtual iteration retrospectives, 608–609
Visio, creating diagrams with, 700
Vision
accelerating change, 109
articulating change, 652
determining, 632
distributed authority equals clear span of, 660
leadership with accountability and, 565
reviewing in planning agenda, 329
Visioning
crafting product or vision statement, 172–175
goals and objectives analysis, 181–184
leap of faith and. *see* Leap of faith hypotheses
on the map, 148–149
mapping to IIBA and PMI guides, 679
objectives, 148–149
overview of, 147, 150–151
problem or opportunity statement, 167–169
product and epic, 150–152
product portrait, 169–171
readiness checklist, 686
root-cause analysis, 152–166
specifying product or epic, 166–167
stakeholder analysis/engagement activities, 175–181
summary of, 192
Visual cues, 132
Visualization, inter-team collaboration via, 616
VoC (voice of the customer), 658
Voice of the customer (VoC), 658
Void risk, 345
Volunteer army, accelerating change, 109
Vulnerabilities, alpha testing for, 533

W

Waiting, as waste, 52
Walk the board, alternative to daily standup, 474
Walking Skeleton (spanning application) MVP, 361
Walkthrough
big room iteration planning, 598–599
Open Space events, 613–614
pivot-or-persevere meetings, 545–546
POC meeting, 602
quarterly release retrospective, 543–544
scaled feature preparation, 604–605
scaled quarterly feature retrospective, 610–611
virtual iteration retrospective, 609

Waste
  acceptance criteria helps drive out, 410
  estimation is, 338
  Lean software development tools and, 52–54
  reducing with responsible procrastination, 659
  seven wastes in Lean software development, 51–52
  stakeholder productivity expectations and, 101
  updating BA documentation to avoid, 496
  value stream mapping highlights/reduces, 283
Waterfall
  agile key practices versus, 65–68
  agile long-term planning versus, 224
  agile replacing, 22
  changes in, 14
  failure of, 22
  step-by-step analysis and planning of, 5
  success of agile vs., 23
  using qualifiers to avoid, 417
  working with teams, 619–620
Weighted shortest job first (WSJF)
  constructing story map ribs, 380
  determining for PBI, 126–127
  for sequencing decisions, 341
  sequencing epics and features in backlog, 202, 213
"What?"
  Connextra template clause, 406, 407
  decision tables, 434
  problem or opportunity statement, 167–169
  product portrait, 170
  well-formed AC describes, 412
"When?"
  problem or opportunity statement, 167–169
  product portrait, 170
"Where?," problem or opportunity statement, 167–169
Whiteboarding, scaled feature preparation, 602–603
"Who?"
  Connextra template clause, 406
  problem or opportunity statement, 167–169
Who cares? quadrant, purpose alignment model, 88, 90
Whole teams, XP, 49
Whole-product level, 568
Whole-team culture, 94
Who-What-Why template. *see* Connextra template
"Why?"
  Connextra template clause, 406–407
  culture embracing change does not ask, 653
  decision tables, 434
  problem or opportunity statements, 167–169
  product portrait, 170
"Why not?," culture embracing change asks, 653
Wide and shallow, long-term implementation of features, 238–240
WIP. *see* Work-in-progress (WIP) limits, Kanban
Wireframes, product portrait for, 170–171
Work, included in estimates, 333
Work items
  inter-team collaboration via sequential, 617
  Kanban, 37, 44, 395
Workaround card, story map, 368–369
Workflow Steps pattern, splitting stories, 423–424
Work-in-progress (WIP) limits, Kanban, 44, 144, 476–479
Wow feature, 206–207, 269
Writely, as Differentiator MVP, 360
WSJF. *see* Weighted shortest job first (WSJF)

X

XP (Extreme Programming)
  The Planning Game guidelines, 322–325
  using timeboxed planning. *see* Timeboxed planning

Y

Y-axis units, cumulative flow diagram, 488

Z

Zones, Agile Analysis and Planning Map, 69–71, 72–73