LEARNING MADE EASY

Enterprise Agility

for dummies®

A Wiley Brand

Use tools to identify your organization's culture

Find a matching enterprise agile framework

Start a successful organizational change

Doug Rose

CSP-SM, PMI-ACP, PMP, SAFe SPC

Enterprise Agility

by Doug Rose

for dummies®
A Wiley Brand

Enterprise Agility For Dummies®

Published by: **John Wiley & Sons, Inc.,** 111 River Street, Hoboken, NJ 07030-5774, www.wiley.com

Copyright © 2018 by John Wiley & Sons, Inc., Hoboken, New Jersey

Published simultaneously in Canada

No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, except as permitted under Sections 107 or 108 of the 1976 United States Copyright Act, without the prior written permission of the Publisher. Requests to the Publisher for permission should be addressed to the Permissions Department, John Wiley & Sons, Inc., 111 River Street, Hoboken, NJ 07030, (201) 748-6011, fax (201) 748-6008, or online at http://www.wiley.com/go/permissions.

Trademarks: Wiley, For Dummies, the Dummies Man logo, Dummies.com, Making Everything Easier, and related trade dress are trademarks or registered trademarks of John Wiley & Sons, Inc. and may not be used without written permission. SAFe and Scaled Agile Framework are registered trademarks of Scaled Agile, Inc. All other trademarks are the property of their respective owners. John Wiley & Sons, Inc. is not associated with any product or vendor mentioned in this book.

LIMIT OF LIABILITY/DISCLAIMER OF WARRANTY: THE PUBLISHER AND THE AUTHOR MAKE NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE ACCURACY OR COMPLETENESS OF THE CONTENTS OF THIS WORK AND SPECIFICALLY DISCLAIM ALL WARRANTIES, INCLUDING WITHOUT LIMITATION WARRANTIES OF FITNESS FOR A PARTICULAR PURPOSE. NO WARRANTY MAY BE CREATED OR EXTENDED BY SALES OR PROMOTIONAL MATERIALS. THE ADVICE AND STRATEGIES CONTAINED HEREIN MAY NOT BE SUITABLE FOR EVERY SITUATION. THIS WORK IS SOLD WITH THE UNDERSTANDING THAT THE PUBLISHER IS NOT ENGAGED IN RENDERING LEGAL, ACCOUNTING, OR OTHER PROFESSIONAL SERVICES. IF PROFESSIONAL ASSISTANCE IS REQUIRED, THE SERVICES OF A COMPETENT PROFESSIONAL PERSON SHOULD BE SOUGHT. NEITHER THE PUBLISHER NOR THE AUTHOR SHALL BE LIABLE FOR DAMAGES ARISING HEREFROM. THE FACT THAT AN ORGANIZATION OR WEBSITE IS REFERRED TO IN THIS WORK AS A CITATION AND/OR A POTENTIAL SOURCE OF FURTHER INFORMATION DOES NOT MEAN THAT THE AUTHOR OR THE PUBLISHER ENDORSES THE INFORMATION THE ORGANIZATION OR WEBSITE MAY PROVIDE OR RECOMMENDATIONS IT MAY MAKE. FURTHER, READERS SHOULD BE AWARE THAT INTERNET WEBSITES LISTED IN THIS WORK MAY HAVE CHANGED OR DISAPPEARED BETWEEN WHEN THIS WORK WAS WRITTEN AND WHEN IT IS READ.

For general information on our other products and services, please contact our Customer Care Department within the U.S. at 877-762-2974, outside the U.S. at 317-572-3993, or fax 317-572-4002. For technical support, please visit https://hub.wiley.com/community/support/dummies.

Wiley publishes in a variety of print and electronic formats and by print-on-demand. Some material included with standard print versions of this book may not be included in e-books or in print-on-demand. If this book refers to media such as a CD or DVD that is not included in the version you purchased, you may download this material at http://booksupport.wiley.com. For more information about Wiley products, visit www.wiley.com.

Library of Congress Control Number: 2018930061

ISBN 978-1-119-44613-2 (pbk); ISBN 978-1-119-44610-1 (ebk); ISBN 978-1-119-44609-5 (ebk)

Manufactured in the United States of America

10 9 8 7 6 5 4 3 2 1

Contents at a Glance

Introduction 1

Part 1: Getting Started with Enterprise Agility 5

CHAPTER 1: Taking It All In: The Big Picture 7

CHAPTER 2: Reviewing Agile Team Practices and Frameworks 27

CHAPTER 3: Simplifying Lean Agility with SLAM 59

Part 2: Reviewing the Top Enterprise Agile Frameworks 81

CHAPTER 4: Joining the Big Leagues with the Scaled Agile Framework. 83

CHAPTER 5: Growing Scrum with Large-Scale Scrum. 131

CHAPTER 6: Making Process Decisions with Disciplined Agile Delivery. 181

CHAPTER 7: Working in Tribes with the Spotify Engineering Culture 209

CHAPTER 8: Improving Workflow and Eliminating Waste with Kanban and Lean 229

Part 3: Leading a Large-Scale Organizational Change. 261

CHAPTER 9: Sizing Up Your Organization 263

CHAPTER 10: Driving Organizational Change. 285

CHAPTER 11: Putting It All Together: Ten Steps to an Agile Enterprise 309

Part 4: The Part of Tens 329

CHAPTER 12: Ten Reasons Enterprise Agile Transformations Fail 331

CHAPTER 13: Ten Tips for Overcoming Common Obstacles. 341

CHAPTER 14: Ten Ways Enterprise Agility Improves Product Delivery 349

Index 359

Table of Contents

INTRODUCTION 1
About This Book. 1
Foolish Assumptions 2
Icons Used in This Book 3
Beyond the Book 4
Where to Go from Here 4

PART 1: GETTING STARTED WITH ENTERPRISE AGILITY 5

CHAPTER 1: Taking It All In: The Big Picture 7
Defining Agile and Enterprise Agility 7
Understanding agile product delivery 8
Defining "enterprise agility" 12
Checking out popular enterprise agile frameworks 15
Practicing as much agile as your organization can tolerate 16
Achieving Enterprise Agility in Three Not-So-Easy Steps 18
Step 1: Review the top enterprise agile frameworks 18
Step 2: Identify your organization's existing culture 19
Step 3: Create a strategy for making big changes 21

CHAPTER 2: Reviewing Agile Team Practices and Frameworks 27
Exploring Common Agile Practices 28
Starting with user stories, epics, and themes 28
Estimating with story points 30
Queuing up work 33
Conducting stand-up meetings 34
Shifting to test-driven development 34
Developing product iterations through continuous integration . . . 34
Delivering Products with Scrum 36
Wrapping your brain around Scrum theory 37
Getting up to speed on Scrum values 38
Working in a Scrum team 39
Stepping through Scrum events 41
Producing Scrum artifacts 46
Growing Scrum 48

Developing Better Software with Extreme Programming 50
Checking out XP's values 50
Following XP's software engineering practices 51
Noting the similarities between XP and Scrum 52
Mixing and matching agile frameworks 53
Learning from Manufacturing with Lean Software Development 54
Managing Workflow with Kanban 56
Innovating Quickly with Lean Startup 56
Using a build-measure-learn feedback loop 57
Focusing on the one metric that matters 58
Mixing Lean Startup with enterprise agile frameworks 58

CHAPTER 3: **Simplifying Lean Agility with SLAM** 59
Introducing the Simple Lean-Agile Mindset (SLAM) 60
Starting at the Bottom with System-Level Optimization 62
Shortening the cycle time 63
Clearing communication channels 63
Budgeting for value streams 64
Encouraging transparency 65
Timeboxing 66
Working in cross-functional teams 67
Respecting people 68
Removing fear of failure 68
Setting and Executing a Strategic Vision 69
Breaking it down 71
Prioritizing the work 72
Pulling work into the teams 73
Taking an Empirical Approach to Products, Operations, and Innovation 74
Pulling in ideas, features, and tasks 76
Getting real-time feedback by delivering in small batches 77
Building Toward Business Agility 78

PART 2: REVIEWING THE TOP ENTERPRISE AGILE FRAMEWORKS 81

CHAPTER 4: **Joining the Big Leagues with the Scaled Agile Framework** 83
Getting Your Head in the SAFe Game 84
Meeting the SAFe framework and principles 84
Picking and choosing what to use 88
Approaching SAFe as a practical compromise 90
Avoiding water-agile-fall 92
Differentiating doing agile from being agile 94
Wait a minute: Is SAFe the right solution? 94

Stepping Through the SAFe Management Layers and Levels 96
Bottom up: Starting at the foundation 96
Heading to the top: The enterprise 98
Visioning at the Portfolio level 98
Tackling big jobs at the Large Solution level 107
Making things happen at the Program level 111
Getting to work at the Team level 120
Filling in the background with the SAFe spanning palette 122
Making Your Organization Agile with SAFe 124
Plugging the gaps 125
Making your organization agile instead of fitting agile into your organization 126
Starting communities of practice 127

CHAPTER 5: **Growing Scrum with Large-Scale Scrum** 131

Taking a Quick Tour of the LeSS Framework 132
Tracing the product development process 133
Brushing up on LeSS principles 136
Getting up to speed on LeSS structure 139
Understanding the importance of technical excellence 143
Meeting LeSS management 145
Supersizing your delivery with LeSS Huge 147
Descaling Enterprise Agility 150
Embracing the spirit of LeSS 151
Experimenting to create your own approach 152
Seeing software developers as craftsmen/women 153
LeSS is Scrum 154
Getting to Know the Key Players 155
Starting at the top: The head of product 156
Meeting the LeSS product owner 156
Getting to know the LeSS Scrum Master 159
Moving Up to Scrum at Scale: Adoption 160
Getting started: Laying the foundation 162
Committing to delivery in sprint planning 166
Coordinating efforts through communication 168
Refining the product backlog 170
Learning from achievements and mistakes: Continuous improvement 173
Avoiding common LeSS pitfalls 176

CHAPTER 6: **Making Process Decisions with Disciplined Agile Delivery** 181

Understanding What Makes DA Tick 182
Brushing up on the principles of effective process frameworks 183
Exploring the DA process decision framework 185

Seeing DAD as a goal-driven, hybrid approach 187
Looking at DAD as a group of process blades 187
Delivering in Lifecycles 191
Navigating the three-phase delivery lifecycle 193
Choosing a DAD delivery lifecycle 198
Striving to become enterprise aware and to consider the risk-value lifecycle 200
Governing the lifecycle with milestones 200
Getting to Know the Cast: Roles 201
Meeting the lead actors: Primary roles 202
Stepping behind the scenes with the supporting cast: Secondary roles 204
Deciding whether DA is worth the trouble 206
Appreciating the value in simplicity 207

CHAPTER 7: **Working in Tribes with the Spotify Engineering Culture** 209

Building Your Spotify Community 210
Starting with a squad 211
Forming tribes of squads 216
Setting up chapters 218
Sharing interests and knowledge in guilds 218
Embracing a Creative, Failure-Friendly Culture 221
Driving agility and innovation through culture and values 221
Reducing the negative consequences of failure 223
Encouraging innovation 223
Developing an aversion to waste 224
Engaging in continuous improvement 224
Strengthening community and culture overall 225
Understanding Spotify's Approach to Product Development/Planning 225
Changing the game plan for larger products 226
Having a system owner 227
Deciding Whether the Spotify Approach Is Right for You 227

CHAPTER 8: **Improving Workflow and Eliminating Waste with Kanban and Lean** 229

Grasping Kanban Principles and Practices 230
Brushing up on Kanban principles 231
Embracing Kanban properties 231
Pulling rather than pushing work 233
Working in small batches 234
Making Systems Lean 235
Getting up to speed on Lean's core values 236
Connecting Lean manufacturing to software development 236

Implementing Kanban and Lean 239
Mapping your value stream 239
Eliminating waste 242
Identifying potential bottlenecks 242
Creating Kanban boards 243
Creating Kanban cards 247
Using Kanban boards to track workflow 252
Improving workflow 253
Using Kanban to reduce management meddling 257
Is Lean Kanban a Viable Enterprise Agile Framework? 258

PART 3: LEADING A LARGE-SCALE ORGANIZATIONAL CHANGE 261

CHAPTER 9: Sizing Up Your Organization 263
Agilebots Transform! Committing to Radical Change 264
Understanding What Culture Is and Why It's So Difficult to Change 265
Understanding why culture is so entrenched 266
Avoiding the common mistake of trying to make agile fit your organization 267
Identifying Your Organization's Culture Type 268
Running with the wolf pack in a control culture 270
Rising with your ability in a competence culture 272
Nurturing your interns in a cultivation culture 274
Working it out together in a collaboration culture 276
Laying the Groundwork for a Successful Transformation 279
Appreciating the value of an agile organization 279
Clarifying your vision 281
Planning for your transformation 282

CHAPTER 10: Driving Organizational Change 285
Choosing an Approach: Top-Down or Bottom-Up 286
Driving Change from Top to Bottom with the Kotter Approach 287
Step 1: Create a sense of urgency around a Big Opportunity 288
Step 2: Build and evolve a guiding coalition 289
Step 3: Form a change vision and strategic initiatives 290
Step 4: Enlist a volunteer army 291
Step 5: Enable action by removing barriers 292
Step 6: Generate (and celebrate) short-term wins 293
Step 7: Sustain acceleration 293
Step 8: Institute change 294
Improving your odds of success 295

Driving a Grass-Roots Change: A Fearless Approach 295
Recruiting a change evangelist 296
Changing without top-down authority 298
Making change a self-fulfilling prophecy 298
Looking for change patterns 300
Recruiting innovators and early adopters 301
Tailoring your message. 301
Steering clear of change myths 301
Overcoming Obstacles Related to Your Organization's Culture 304
Seeing how culture can sink agile 304
Acknowledging the challenge 305
Prioritizing the challenge 305
Gaining insight into motivation 306

CHAPTER 11: **Putting It All Together: Ten Steps to an Agile Enterprise** 309
Step 1: Identifying Your Organization's Culture 310
Step 2: Listing the Strengths and Challenges with Changing Your Culture 312
Step 3: Selecting the Best Approach to Organizational Change Management 315
Step 4: Training Managers on Lean Thinking 316
Step 5: Starting a Lean-Agile Center of Excellence (LACE) 318
Step 6: Choosing a High-Level Value Stream. 319
Step 7: Assigning a Budget to the Value Stream 320
Step 8: Selecting an Enterprise Agile Framework 322
Step 9: Shifting from Detailed Plans to Epics. 324
Step 10: Respecting and Trusting Your People 325

PART 4: THE PART OF TENS 329

CHAPTER 12: **Ten Reasons Enterprise Agile Transformations Fail** 331
The Organization's Culture Clashes with Agile Values 331
Teams Aren't Interested in Making Changes. 332
Executive Support Is Lacking 333
The Proposed Change Is Too Radical 334
The Customer Won't Cooperate 334
Leadership Refuses to Invest in Training 335
The Developers Insist on Requirements 335
Each Team Wants to Do Its Own Thing 336
Nobody Has a Plan to Measure Improvements 337
The Functional Areas Are Too Deeply Entrenched 339

CHAPTER 13: **Ten Tips for Overcoming Common Obstacles** 341
Develop a Clear Roadmap 341
Find Support at the Top 343
Set Realistic Expectations. 343
Compensate Employees for Their Investment 344
Change Minds as Well as Systems 344
Be Objective When Assessing Your Organization's Culture 345
Build Broad Consensus on the Reason for the Change. 345
Don't Rely Solely on Outside Consultants to Drive Change. 346
Encourage Reluctant Executives and Managers to Embrace the Change. 347
Listen to the Skeptics 348

CHAPTER 14: **Ten Ways Enterprise Agility Improves Product Delivery** 349
Increasing Agility 349
Boosting Innovation 350
Enhancing Transparency 350
Boosting Productivity 351
Making Product Development More Fun and Rewarding 352
Strengthening Customer Relationships 353
Enhancing Product Quality. 354
Making Product Delivery More Predictable. 355
Reducing the Risk of Failure. 355
Improving Developer Discipline 356

INDEX 359

Introduction

To survive and thrive in a fast-moving economy, enterprises must work to improve their agility; they need to be able to pivot quickly to respond to new technologies, emerging opportunities and threats, and ever-evolving customer demands. However, many organizations are built more like cruise ships than jet skis. They're designed to command and control, making decisions at the top and passing them along the chain of command to the employees who do the work. Even when these organizations manage to change direction, they're either too late to market or too far off course to stay ahead of the competition.

An agile enterprise is lean and nimble. Product developers collaborate closely with the organization's leaders and management and with customers to optimize value. Decision-making is distributed throughout the organization, and employees are encouraged to take the initiative, experiment and innovate, and continuously learn and improve. Agile organizations ride the waves of change instead of being tossed and turned by external factors beyond their control.

However, a large-scale agile transformation is no small feat, especially when it develops complex products that traditionally involve a great deal of up-front planning. How do you transform a large organization with deeply entrenched functional areas into a collection of small, closely aligned teams without sinking the ship? In this book, I answer that question.

About This Book

Over the past ten years, I've helped a number of large organizations become agile enterprises. Most organizations that succeed follow the same three-step approach:

1. Review the top enterprise agile frameworks.
2. Identify the organization's existing culture.
3. Create and execute a strategy for making big changes.

Those that fail never do so from a lack of trying. They fail from *doing* agile instead *being* agile. They create teams that do everything agile teams are supposed to do, but they continue to function as they always did — making decisions at the top, issuing commands, and expecting employees to follow orders. They just don't try

to change their *mindset*. As a result, they fall short of creating a culture of mutual trust and respect in which employees and customers collaborate closely to deliver innovative products. These organizations look like agile enterprises, but they never reap the full benefits of agility.

In this book, I take a three-pronged approach to transforming organizations into agile enterprises so they can both *be* agile and *do* agile:

- **Being agile:** To achieve enterprise agility, everyone in your organization must have a shared understanding of what it is and its purpose. If your teams understand agility, but your executives and managers don't, you'll end up with teams that merely do what they're told instead of coming up with creative solutions. In Part 1, I bring you up to speed on the agile mindset, describe agile at the team level, and present the Simple Lean-Agile Mindset (SLAM), which provides a high-level understanding of enterprise agility.
- **Doing agile:** In Part 2, I introduce the top enterprise agile frameworks — Scaled Agile Framework® (SAFe®), Large-Scale Scrum (LeSS), Disciplined Agile Delivery (DAD), the Spotify Engineering Culture, Kanban, and Lean. Most organizations begin their agile transformations by choosing one of these frameworks and then tailoring it to meet their needs. Others may use the frameworks to generate ideas for their own custom enterprise agile framework.
- **Making the transformation:** In Part 3, I lead you through the process of transforming your organization to improve your enterprise agility. Here, you evaluate your organization's existing culture, choose a top-down or bottom-up approach to executing the transformation, and then follow my detailed ten-step transformation process. (Keep in mind that enterprise agility is an ongoing process of continuous improvement, not a one-time event. Your organization will evolve over time.)

Foolish Assumptions

A key component of enterprise agility is empirical process control. As such, its practitioners frown upon making detailed plans. Instead, teams are encouraged to "think, build, release, and tweak," through empirical, data-driven decisions. However, because I don't know you personally (although you seem nice), I had to make several assumptions about you when writing this book:

- You're probably an executive or manager in a large organization who has heard of enterprise agility and you want to learn more about it. Or you have decided already that you want to make your organization an agile enterprise.

(Or you may be an employee who sees the value of enterprise agility and you want to enlighten others in your organization.)

- Your knowledge of agility ranges from knowing nothing about it to actually working as a part of an agile team. In other words, you can benefit from this book whether you know a lot or a little about enterprise agility.
- You're interested primarily in *enterprise* agility, not *business* agility. Enterprise agility pertains to product development, while business agility has a much broader scope that permeates an organization. While I touch on business agility in several chapters, my focus in this book is on using enterprise agility to optimize *product delivery*.
- You're committed to adopting an enterprise agile mindset. That is, you're not just interested in making your organization more agile, but also you want everyone in your organization to collaborate as autonomous, closely aligned teams to deliver awesome new products.

Icons Used in This Book

Throughout this book, icons in the margins highlight different types of information that call out for your attention. Here are the icons you'll see and a brief description of each.

REMEMBER

I want you to remember everything you read in this book, but if you can't quite do that, then remember the important points flagged with this icon.

TECHNICAL STUFF

Throughout this book, I stick to the bare essentials — what you need to know to conduct a successful enterprise agile transformation. If I dig any deeper into a topic, I warn you with this icon. If you're looking for an in-depth discussion, dig in; otherwise, you can safely skip ahead.

TIP

Tips provide insider insight. When you're looking for a better, faster way to do something, check out these tips.

WARNING

"Whoa!" Although enterprise agility encourages learning through experimentation and failure, learning without the failure is always preferred. When you see the warning icon, proceed with caution. I've seen many organizations make critical mistakes that have slowed or derailed their attempts at becoming more agile. Learn from *their* mistakes.

CASE STUDY

Throughout this book, you'll find plenty of real-life case studies that provide valuable insight into enterprise agile transformations (successes and failures), so if you're the type of person who commonly skips sidebars, I strongly encourage you to break that nasty habit — at least for this book.

Beyond the Book

In addition to the abundance of information and guidance on enterprise agility that I provide in this book, you get access to even more help and information online at `Dummies.com`. There you can find a free, access-anywhere Cheat Sheet that gives you even more pointers on how to embark on an enterprise agile transformation. To get this Cheat Sheet, simply go to `www.dummies.com` and search for "*Enterprise Agility For Dummies* Cheat Sheet" in the Search box.

Where to Go from Here

You're certainly welcome to read this book from cover to cover, but I wrote it in a way that facilitates skipping around. If you're new to agile, I recommend you read Chapters 1 and 2 to get up to speed on the topic. Chapter 3 is also essential reading, but you *could* hold off on reading Chapter 3 until you review the different enterprise agile frameworks in Part 2. Chapter 3 provides a conceptual understanding of enterprise agility that highlights common themes among all the frameworks.

In Part 2, I cover the top enterprise agile frameworks, so feel free to skip around in that part — the chapters aren't sequential. I describe each of the frameworks, so you can make a well-informed choice of which framework to start with.

When you're ready to embark on your enterprise agile transformation, turn to Part 3. In this part, the chapters are sequential, so read Chapters 9, 10, and 11 in that order. Chapter 11 is most important, because it outlines a specific ten-step process for transforming your organization into an agile enterprise.

With enterprise agility, failing is okay, as long as you learn from it and persevere. The danger is that failing often leads to discouragement. When an agile transformation doesn't meet expectations, organizations often conclude that greater agility isn't the right solution and they give up. In nearly all cases, improving agility *is* the right solution — it's the transformation process that fails. Approach enterprise agility with the conviction that it's the right solution as long as everyone in your organization adopts an agile mindset. If you're struggling to overcome obstacles, look for and address issues in the transformation process, which can almost always be traced back to pockets of resistance in the organization — people who haven't accepted the agile mindset.

1 Getting Started with Enterprise Agility

IN THIS PART . . .

Get up to speed on what's involved in making your organization an agile enterprise and begin to appreciate the crucial role that culture plays in any enterprise agile transformation.

Explore the key differences between agile at the team level, enterprise agility, and business agility, and recognize the importance of starting on a smaller scale.

Start to gauge just how receptive or resistant your organization will be to the big changes you're about to implement.

Take a quick look at the 1-2-3 process of transitioning your organization's product delivery to enterprise agility, and start thinking about the approach you will take to transform your organization.

Brush up on agile basics at the team level, so you have a fundamental understanding of various agile frameworks, such as Scrum, Extreme Programming, Lean Software Development, and Kanban.

Get to know the principles that drive agile product development and the common practices many agile teams use in the product delivery process.

Understand the challenges you're likely to face as you scale agile to develop and deliver enterprise-level products, and start thinking about ways to meet these challenges.

IN THIS CHAPTER

- Getting up to speed on the agile mindset
- Defining enterprise agility
- Distinguishing enterprise agility from business agility
- Transforming your organization with the 1-2-3 approach

Chapter 1

Taking It All In: The Big Picture

When you're getting ready to tackle a complex topic, such as enterprise agility, having a general understanding of the topic and what it entails is a great place to start. In this chapter, I give you that eye-in-the-sky view of enterprise agility. Here you develop a general understanding of agile and enterprise agility and the key distinction between the two. You discover how to build an agile enterprise without making the common mistake of trying merely to scale up agile frameworks to your entire organization. And I introduce you to some commonly used agile frameworks that I cover in greater detail in Part 2.

Defining Agile and Enterprise Agility

Because you're reading a book about enterprise agility, I assume you're familiar with the topic, but readers may have different levels of understanding and different ideas about what "agile" and "enterprise agility" mean. In this section, I define the two terms and explain the key differences between them.

Understanding agile product delivery

According to the Agile Alliance, *agile* is "the ability to create and respond to change in order to succeed in an uncertain and turbulent environment." Instead of relying on extensive up-front planning, "solutions evolve through collaboration between self-organizing, cross-functional teams utilizing the appropriate practices for their context." (*Self-organizing* means the teams manage themselves. *Cross-functional* means each team has all the expertise and skills required to complete its work.)

Small teams (typically fewer than nine people) are empowered to collaborate and make decisions as opposed to being subject to intensive planning, rigid processes, and consulting management for direction and approval. The goal is to remove the management obstacles that commonly get in the way of competent people doing their jobs.

REMEMBER

Agile frameworks originated in the context of software development, an area subject to rapid change — changes in end-user needs, technologies, and even the tools and processes used to develop software. To be effective, developers needed to be agile. They had to be able to make decisions locally instead of having to wade through the bureaucracy of traditional management matrixes.

The Agile Manifesto

In 2001, 17 software developers gathered at The Lodge at Snowbird ski resort in the Wasatch mountains of Utah and talked about why companies were having difficulty developing software. They represented some of the newer methods in software development — Scrum, Extreme Programming, the Crystal Methods, and continuous integration. After some discussion, they identified what was common among all these approaches: They were all lightweight compared to the complexities of the popular software development approaches at the time, including IBM's Rational Unified Process (RUP) and the manufacturing-inspired waterfall approach. They didn't want to become known as a bunch of "lightweights," so they settled on calling their approach "agile." Together they formed the Agile Alliance.

The word "agile" implied that software developers needed to be quick and flexible and able to change course quickly to take advantage of new ideas, changing customer needs, and emerging technologies. Many of the first articles and books on the topic included drawings of cheetahs.

After they settled on a name for their workgroup, a few of the members drafted the *Manifesto for Agile Software Development*. The Agile Manifesto, as it has come to be called, provides insight into the mindset agile embraces (from `agilemanifesto.org`):

> We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:
>
> - **Individuals and interactions** over *processes and tools*
> - **Working software** over *comprehensive documentation*
> - **Customer collaboration** over *contract negotiation*
> - **Responding to change** over *following a plan*
>
> That is, while there is value in the italicized items on the right, we value the bolded items on the left more.

After the group came down from the mountain, they decided to continue to work together. In the weeks and months following their return, they added 12 agile principles they deemed to be consistent with the Agile Manifesto's four values and exemplary of the kinds of operating principles one could expect to observe in an agile group.

Agile principles

Agile is based on the following 12 guiding principles:

- Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
- Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
- Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
- Business people and developers must work together daily throughout the project.
- Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
- The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
- Working software is the primary measure of progress.
- Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
- Continuous attention to technical excellence and good design enhances agility.
- Simplicity — the art of maximizing the amount of work not done — is essential.

- The best architectures, requirements, and designs emerge from self-organizing teams.
- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

WHAT DOES IT MEAN TO BE "AGILE"?

Agile is an organizing concept for orchestrating software development or other work. It has never been or tried to be a unified engineering system for developing software. Since its birth, it has become more like an empty truck bed gathering new ideas (after being accepted as common practice) as it travels through time. Soon after the Agile Manifesto was written, several books and articles were added to the truck bed of ideas collectively regarded as "agile":

- In 2002, *Test-Driven Development: By Example* by Kent Beck, encouraged developers to think about what the software would accomplish before starting to code.
- Around the same time, James Grenning published an article entitled "Planning Poker or how to avoid analysis paralysis while release planning," to help agile teams create group estimates for the time they thought the work would take to complete.
- In 2003, *Lean Software Development: An Agile Toolkit*, by Mary and Tom Poppendieck, argued that software was pulling the wrong ideas from manufacturing. Instead of using a one-phase-at-a-time waterfall approach, agile teams should work to maximize value to the customer by making their processes leaner — a concept inspired by Toyota's manufacturing process, the Toyota Production System (TPS).
- In 2006, *Agile Retrospectives: Making Good Teams Great*, by Esther Derby and Diana Larsen, introduced the concept and practice of team retrospectives.
- In 2010, *Kanban: Successful Evolutionary Change for Your Technology Business*, by David J. Anderson, explained how to use Lean principles to visualize work and maximize workflow.

The ideas presented in those books and others like them were not written into the original Agile Manifesto, although many of those ideas represent practices known to the Manifesto's authors. Today most agile teams consider them to be a core part of their work. The takeaway message here is that even agile is agile — subject to change "in an uncertain and turbulent environment." To become agile, you need to understand the accepted values, principles, and practices and then apply them in a way that works for you and adapt whenever necessary. What works for one organization may not work for another, and the agile of tomorrow may not look anything like what the writers of the Agile Manifesto had envisioned in 2001.

Agile frameworks

To facilitate their product development process, agile teams use different methodologies, referred to as "frameworks," such as the following:

- **Extreme Programming (XP):** A team of contributors, formed around a business representative called "the customer," operates according to certain basic values including simplicity, communication, feedback, courage, and respect. Through high customer involvement, close teamwork, rapid feedback loops, and continuous planning and testing, teams strive to deliver working software at frequent intervals (generally one to three weeks).
- **Kanban:** A team uses a "Kanban board" to track and visualize workflow. The board divides product development stages into columns, such as To Do, In Progress, and Done. Each work item is described on a "Kanban card" (index card or sticky note) and cards are arranged in the To Do column in order of priority. As team members are able, they pull work items from the To Do column and perform the work required. When they're done, the card is moved to the Done column. It gets more complicated, and the Kanban board can have many columns, but that's the general idea. Kanban strives to minimize work in progress (WIP), eliminate bottlenecks, and minimize waste (increase efficiency).
- **Lean Startup:** The Lean methodology follows a "Think it, build it, ship it, tweak it" approach with data driving ideas that lead to the development of code. The framework calls for a close connection with customers and frequent tests that drive a never-ending cycle of improvement.
- **Scrum:** A *product owner* provides a prioritized wish list of features, fixes, and so on, called a *product backlog.* A *development team* draws from the top of that list (a *sprint backlog*), decides how to implement those items, and estimates the amount of time it will take to complete that work in the form of a potentially shippable product (typically 30 days or fewer). The development team meets daily to assess progress and discuss issues. A *Scrum Master* functions as the servant-leader for the Scrum team — more in the capacity of facilitator than project manager. There is a clear separation of concerns as the product owner prioritizes *what* must be done next, and the development team figures out *how* to get those things done.

See Chapter 2 for more about these team-level agile frameworks.

Agile practices

Agile practices are specific applications of agile, as opposed to more general theories and principles. Here are just a few of the many agile practices:

- **Planning poker:** A game for estimating product backlogs. The product owner describes a product feature or function, and each player (team member) draws a card from her own deck with a value, such as 1, 2, 3, 5, 8, 20, 40, or 100 to estimate the time or work required. After all players have chosen their cards, they flip their cards over at the same time. If everyone's estimate is the same, that becomes the estimate; otherwise, players discuss the reasons for their estimates until consensus is reached or the team determines that more information is needed.
- **Product backlog:** A prioritized list of work items that must be completed to deliver a product.
- **Stand-up meetings:** Daily meetings during which everyone stands as a clear message that the meetings cannot extend past 15 minutes.
- **User story:** A description of a product feature from the user's perspective such as, "Customers can pay with credit cards, debit cards, or PayPal."
- **Work-in-progress (WIP) pull board:** Kanban uses a WIP pull board designed to limit WIP and encourage collaboration among team members. Seeing a WIP item on the board, the team can address the issue and remove the item. The notion of "pull" is key; instead of having work pushed on them, which often produces traffic jams and delays, team members pull work items from the board as they're able to do the work.

WARNING

Don't equate agile with a framework or a set of agile practices. Agile is more of a culture or shared mindset among team members that influences the way team members think about their work and impacts the way they work individually and together as a team. Having a shared understanding and appreciation of the agile concept is far more important than having shared practices. For example, mutual respect, trust, and a spirit of innovation are far more important than user stories and stand-up meetings.

Defining "enterprise agility"

Enterprise agility is agile for big products — typically one that requires many different teams throughout the organization that coordinate with many different departments and stakeholders.

While agile involves one or two teams working on a *part* of a product, enterprise agility may involve dozens or even hundreds of teams working on a *whole* enterprise solution. When you have that many teams working on a single enterprise

solution, you start running into alignment issues and creating a lot of dependencies. Although you may want to remain agile, you need to start with at least a unified vision and have a system in place that enables the teams to communicate, coordinate, and collaborate efficiently and effectively to bring the vision to fruition and improve on the vision through innovation.

While agile team frameworks, including Scrum and Extreme Programming, work well on a small scale, they can lead to chaos when you attempt to scale up. To resolve this issue, the agile community has developed a number of enterprise agile frameworks — systems to help align the efforts of teams working together on a big product and reduce the number of dependencies.

WARNING

Don't confuse enterprise agility with business agility. *Business agility* applies the agile mindset to the entire organization, which is sometimes referred to as "diffusion of IT-based innovations." Business agility deals with all domains, including those outside of product development, such as adaptive leadership, organizational design, human resources (HR) or personnel, and budgeting. This book's focus is on enterprise agility, *not* business agility (but I do include a brief section on business agility near the end of this chapter).

However, for enterprise agility to work in your organization, everyone in the organization must adopt an agile mindset. Otherwise, the traditional management practices that are common in a culture that values predictability and failure avoidance will clash with the agile values of experimentation and innovation. You won't get the full benefit of agile if agile teams are merely doing what they're told.

REMEMBER

Few organizations that consider themselves agile enterprises have the culture and mindset to make that claim. What typically happens is that an organization will have five or six agile teams that practice Scrum, Extreme Programming, Kanban, or Lean Startup. The teams may achieve some degree of success — the organization may produce higher-quality software and the developers may be happier — but until the agile mindset permeates the entire organization, it's not an agile enterprise and will not reap the full benefits of enterprise agility.

TRACING THE RISE OF "ENTERPRISE AGILITY"

After witnessing the success of agile software development teams, people in the agile community began to wonder whether the concept could be scaled to large organizations that develop enterprise solutions. After all, what organization would not want to be more *agile?*

(continued)

(continued)

But large organizations aren't designed to be nimble. As much as everybody celebrates disruptive entrepreneurship, being big has its rewards. Large organizations do a lot of interesting work, and there are real advantages to their size, scale, and deliberation. Most of these organizations focus on steady incremental improvements. The challenge is to help such organizations reap the benefits of agile without losing the benefits of being big.

Enter, enterprise agility. As agile software development was hitting its stride around 2007, the agile community started talking about how to put fast-moving agile teams into larger, more established organizations by "scaling agile." Two early books on the topic were *Scaling Software Agility: Best Practices for Large Enterprises* by Dean Leffingwell (2007) and *Scaling Lean & Agile Development: Thinking and Organizational Tools for Large-Scale Scrum* by Craig Larman and Bas Vodde (2008). At about the same time, Scott Ambler had introduced his Agile Unified Process, but he has since stopped working on it to work on Disciplined Agile Delivery (DAD).

However, the notion of scaling was never accurate. Scaling an agile team would turn the team into a lumbering hippopotamus instead of an agile cheetah. A more effective and realistic solution is to find the sweet spot between fast-moving teams and the slow, deliberate enterprise. At the same time organizations were looking to become more agile, the role of enterprise software in an organization's success was growing, so large organizations needed their software development teams to become more agile. Yet, they needed a buffer zone between agile and the rest of the enterprise.

Enterprise agile transformations created a whole new genre of articles, books, and consultants. In a few short years, the number of people who changed their LinkedIn profile to "agile coach" went from hundreds to tens of thousands as the demand for experts who could help large enterprises navigate their transformation to enterprise agility soared. Many of the authors of these scaling agile ideas started to create their own enterprise agile frameworks. These frameworks proliferated like diet and exercise programs, and large organizations couldn't get enough of these pre-packaged solutions.

These frameworks were so enticing that by 2016 nearly half of all enterprise agile transformations were using (or actively considering) an enterprise agile framework. Just a little over a quarter were considering building their own. The little over a half that were using an enterprise agile framework generally settled for one of the top five frameworks I cover in this book: Scaled Agile Framework® (SAFe®), Large-Scale Scrum (LeSS), Disciplined Agile Delivery (DAD), the Spotify Engineering Culture, or Kanban and Lean.

Even when organizations try to build their own enterprise agile frameworks, they often rely on one of these pre-packaged frameworks as a template. So, while there is no standard enterprise agile framework, a consensus is forming around a standard set of ideas.

Checking out popular enterprise agile frameworks

Just as agile has several different frameworks for structuring the way teams function, enterprise agility has a selection of frameworks that provide direction for how teams work together on enterprise solutions. Currently, about a dozen well-established frameworks are available, and each one takes a different approach. Collectively, these methodologies form a cafeteria of ideas from which organizations can choose based on the organization's existing culture and the culture it wants to establish moving forward.

Following are five of the most popular frameworks:

- **Disciplined Agile Delivery (DAD):** A *process decision framework*, DAD encourages you to make certain choices at different points in product delivery, but doesn't prescribe any specific process to follow to make your organization agile. Instead of prescribing a process, it offers general guidance such as, "Here are the goals, and here are a few approaches for meeting each of those goals, and here's some guidance to help you choose the best approach." You're free to choose any framework and practices to mix and match, or create your own. (See Chapter 6 for details.)
- **Large-Scale Scrum (LeSS):** A framework that contains many of the elements familiar in Scrum at the team level, including sprint planning, backlogs (prioritized lists of work items), sprints (the basic unit of development that results in an iteration of the product), daily sprint meetings, and a sprint retrospective (a sort of post-mortem meeting). The primary distinction between LeSS and Scrum is that with LeSS, you have several teams working in different "lanes" on different sprints, sometimes coordinating and collaborating between lanes. (See Chapter 5 for details.)
- **Lean Product Delivery:** A system for reducing waste in products and processes by eliminating anything that's unnecessary, including excessive steps (in a process) and functionality (in a product) that don't bring value to a customer. The focus is on minimizing waste and maximizing value. (See Chapter 8 for details.)
- **Kanban:** A system in which team members pull work items from a list of prioritized items on a Kanban board to work on them as their capacity allows. Kanban (signal) cards are used to indicate when a work item is ready for the next stage in the process. A buildup of Kanban cards in any stage of the process signals a bottleneck that must be addressed. The emphasis is on maintaining a smooth and continuous workflow. (See Chapter 8 for details.)

- **Scaled Agile Framework (SAFe):** A collection of frameworks, principles, and practices that attempts to combine the best of top-down management with the best of agile. Teams work together as teams of teams (called "agile release trains," or ARTs) and as teams of teams of teams (called "solution trains") to achieve the enterprise's vision. SAFe is one of the more complex frameworks, adding numerous processes, layers, roles, and tools to solution delivery. (See Chapter 4 for details.)
- **Spotify Engineering Culture:** A mashup or composite of agile frameworks and practices that's anchored by a strong culture of mutual respect, trust, and innovation. Teams (called "squads") and teams of teams (called "tribes") are encouraged to experiment freely, release products frequently, and tweak their products and processes for continuous improvement. Failure is not punished, and learning from failure is revered to encourage squads to experiment. (See Chapter 7 for details.)

Practicing as much agile as your organization can tolerate

The downside of some of these enterprise agile frameworks, with the exception perhaps of DAD and the Spotify Engineering Culture, is that they try to "productize" your transformation. (To *productize* is to take a concept like agile and turn it into a pre-packaged solution.) It's like getting a suit off the rack when you really need something that's tailored to your organization.

The suit off the rack isn't really how most enterprise agile transformations are done. There won't be a day when you cross the agile finish line. Your organization will never reach an agile end state. Instead, much like a fitness program, you try to integrate these new ideas into the way you already work. It is a long process of small adjustments and continuous improvement, which is why you should think of your enterprise agile transformation as your organization accepting as much agile as it can tolerate. It's about how well your organization accommodates change.

TIP

Before you even think about where you want to be on the agile scale, look at where you are. How much change can your organization tolerate? Think of it almost like a room in which you can only put so much furniture. If your organization can tolerate only small changes, then think of the highest priority agile practices that you can try to implement.

WARNING

Don't try to go too big too soon. Many enterprise agile frameworks require that you make several big changes simultaneously. The hope is that if your organization can tolerate big changes, you can quickly reap the benefits of your transformation. However, your organization will likely snap back if you try to make too

many big changes too quickly, especially if your organization has a low change tolerance. Consider a more gradual approach — starting with a few teams, reviewing the results, and then building on your success. Build the desired culture in one small corner of your organization and, if successful, it will spread, as long as you remove any obstacles.

REMEMBER

Large organizations usually have a change tolerance — how much change they can stomach without too much grumbling. If you exhaust everybody's ability to change, transformation is likely to grind to a halt. Everyone will go on working, but don't expect any more progress in the change department.

CASE STUDY

A FAILED ATTEMPT

I once worked for one of the largest retailers in the United States. Its business was centered around home improvement and construction. The way it worked made it one of the fastest-growing companies in the world. It wasn't a high-tech business, but it needed technology as a way to improve the organization.

The solution it decided on consisted of replacing all managers in the information technology (IT) department with managers from high-tech businesses. One of the first changes these managers wanted to make was to transform the retailer into an agile enterprise.

The organization was filled with project managers and business analysts, many of them with construction backgrounds, who approached software as they would any other construction project. They were used to a detailed plan, and it was their job to make sure there were no surprises. They valued predictability over innovation.

When the new managers introduced enterprise agility, the business analysts and project managers thought that it sounded like a plan for total chaos, but they didn't want to be singled out as the resident foot-draggers. After all, the organization just removed a lot of its long-time IT managers. Instead, they made a series of symbolic changes. They changed their titles to make them sound more like agile roles, and they stood during meetings. They didn't make any substantive changes. After a few years, the frustrated IT managers drifted off to jobs at different organizations, and the organization ran as it always had.

The moral of this story is this: If you want to make any major changes to an enterprise organization, you need to change minds first.

Achieving Enterprise Agility in Three Not-So-Easy Steps

The best way to implement enterprise agility in your organization is to take the following three steps:

1. Review the top enterprise agile frameworks.
2. Identify your organization's existing culture.
3. Create a strategy for making big changes.

REMEMBER

This three-step process isn't really unique to enterprise agile transformations; it's pretty standard for making any large-scale organizational changes. You want to better understand the changes you're proposing, then understand the environment in which you're making the changes, and finally figure out how to apply these changes to your environment.

Step 1: Review the top enterprise agile frameworks

The first step toward an enterprise agile transformation is to understand what being agile means and get a sense of what different manifestations of agile look like. The fact is that you can achieve enterprise agility in an infinite number of different ways, just as you can use different health and fitness programs, mix-and-match programs, or develop your own program to become healthy and fit.

A great way to start is to look at the top enterprise agile frameworks I describe in Part 2: SAFe, LeSS, DAD, the Spotify Engineering Culture, Kanban, and Lean. Collectively, they provide several frameworks and include numerous agile principles and practices. Simply by exploring the different frameworks, you will start to develop a more agile mindset and begin to appreciate the full scope of enterprise agility.

TIP

As you explore the enterprise agile frameworks in Part 2, try to look beyond each framework to understand the rationale behind it. If you can understand what the developers of each framework were thinking and the problems they were trying to solve, you will be well on your way to making the right decisions and choices for your organization. Remember, pulling a framework off the shelf may work fine, but be open to the possibility of tailoring it to your organization. No framework is a one-size-fits-all solution.

Step 2: Identify your organization's existing culture

One of the biggest reasons organizations fail in their transformation effort is that they don't take their existing culture into account. The problem is worst when an organization with a firmly embedded traditional management matrix tries to become more agile, because strong management tends to clash with some of agile's emphasis on self-organizing teams.

Organizations don't intentionally ignore culture. They're just so immersed in it that they no longer notice it. Culture is sort of like the air that surrounds us; we don't notice the air until a cold front sweeps in. We don't notice culture until it comes in contact with another culture, at which point cultural differences become readily apparent. You may not notice your organization's culture until you try to change it to something that's very different.

I don't want you to make the common mistake of ignoring your existing culture, so I encourage you to size up your culture before attempting to transform it. Following are four common corporate culture types:

- **Collaboration culture:** Common in schools and professional training organizations, collaboration cultures are run like family businesses, with leaders acting as decision-makers, team builders, and coaches. Managers work closely together like a small group of friends, and the closer you are to the head of the organization, the more authority you have. These organizations are typically more open to change than those with a control or competence culture, so they tend to adopt an enterprise agile mindset more readily. However, in a collaboration culture, leadership may have a difficult time allowing decisions to be made at the team level.
- **Competence culture:** Those with the highest level of expertise rise to the top, become managers, and create and delegate tasks. A meritocracy. The management style is task-driven; it's all about who can do the best job at finishing the work. People in competence cultures often become highly specialized in their areas of expertise, because expertise is what is valued and rewarded. If they excel in more than one area, they're likely to be given too many tasks and become quickly overwhelmed, so they specialize. They also don't like to share their knowledge, because it places them at risk of losing some of their authority.
- **Control culture:** This culture is authoritarian with alpha managers setting the direction and beta managers following close behind. Leadership gives orders and demands compliance. Only a few individuals in the organization have

decision-making powers; others must seek approval or permission, making the organization slow to respond to change. Such organizations favor order and certainty and rely on large management systems that ensure predictable outcomes.

- **Cultivation culture:** Employee growth and development form the cornerstone of the organization. Managers seek to bring someone into the organization, hold them up, and then build them up. Charismatic individuals quickly rise to the top, and generalists commonly do well. These organizations tend to be more democratic and transition more easily to an agile mindset, but decision-making can be slow as consensus is sought among large groups of individuals.

TIP

Consider choosing a framework that's a closer match to your current culture than a match to the culture you want for your organization, so the transformation won't be too much of a stretch. Some frameworks are much more agile than others. For example, Spotify's approach gives teams a lot of autonomy, and that may strike you as the way you want your organization to be. However, Spotify's approach works for Spotify, because it's not a huge organization. Spotify has nurtured a collaborative culture from its inception, and the company redesigned its product's architecture to make it more modular, so a squad can work on one feature without having to integrate its work with a lot of other squads and tribes. If your organization has a strong control culture, making the leap to Spotify's approach may be as challenging as trying to jump across the Grand Canyon on a motorcycle.

Instead, SAFe may be the better choice, because it has more practices for top-down decision-making. It allows for some agility while giving managers deep insight and control over the organization.

REMEMBER

An organization may fit into more than one category; for example, its engineers may be driven more by a competence culture, whereas marketing is run more in line with a cultivation culture.

The famous management consultant Peter Drucker once said that "Culture eats strategy for breakfast." This holds true for enterprise agility. Whatever strategies you pick for your enterprise agile transformation, they won't succeed without the support of a culture that values people, respect, trust, and innovation.

CASE STUDY

POOR CHOICE, POOR OUTCOME

I once worked for an organization with a strong competence culture. Management consisted of engineers who were easily persuaded by other engineers. Everyone worked hard to lock up his own expertise. Individuals knew their competence increased their standing and authority in the organization.

This competence culture made working in teams difficult. Everyone wanted to be a superhero and show that he or she was who carried the rest of the team. No one wanted to collaborate or share information with the other team members. Nobody cross-trained; people simply learned more about what they already knew. On top of that, everyone was working long hours to show his commitment to the product. In the end, leaders had to abandon their agile transformation. They couldn't create the kind of collaborative teams they needed to improve the product.

They would have had a better outcome if they had tried something like Kanban, which would have allowed them to track their workflow and introduce the value of collaboration more subtly. Such an approach also would have helped them visualize their workflow and begin to realize how much time they were wasting by not cross-training people.

Step 3: Create a strategy for making big changes

As you think about your strategy for making big changes, look for the sweet spot between your organization's acceptable and unacceptable change, as shown in Figure 1-1. Finding that sweet spot is more art than science. Identify areas you want to change and areas where you're likely to encounter resistance. Try to understand *why* you may encounter resistance in certain areas. Your organization probably has gravitated toward a particular culture for good reasons, so you can decide whether and how an area needs to be changed. If a certain area is less agile for good reason, you may want to let it be.

After you've found your sweet (and not so sweet) spots, you're ready to start adopting the agile frameworks, processes, and principles you choose.

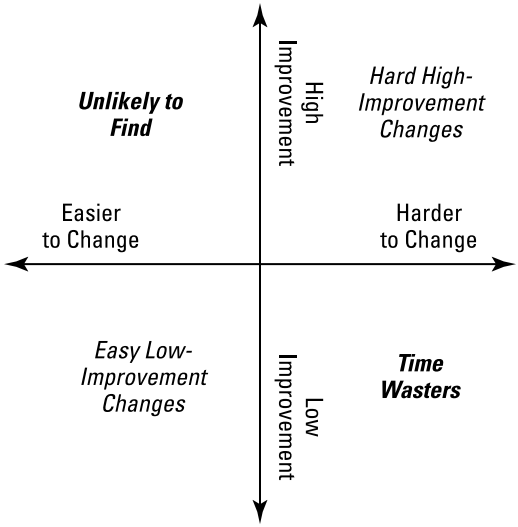

FIGURE 1-1: The change sweet spot.

Choosing a top-down or bottom-up strategy

When you're ready to start your enterprise agility adoption journey, you basically have two big-change strategies from which to choose:

- **Fearless Change:** A bottom-up approach, which can be driven by a few employees. Fearless Change tends to work better in competence, cultivation, and collaboration cultures. Fearless Change may also be effective in smaller, newer organizations that don't yet have a deeply entrenched hierarchy.
- **Kotter approach:** An eight-step, top-down process driven by a change leader, who can be a manager or an outside consultant. The Kotter approach tends to work better in control cultures — the most common culture in large organizations, which typically have a well-defined hierarchy.

See Chapter 10 for more about these two options.

TIP

Whichever strategy you choose, look for opportunities to make smaller, realistic changes. Instead of trying to force change on your organization all at once, win the war gradually, battle by battle. Pick the low-hanging fruit. Giving teams shared workspaces and providing agile training can get the culture ball rolling. Then, build on the momentum of your success.

Mapping out your plan

After you've thought about which approach is likely to work best, map out your plan. As you develop your change management plan, you're likely to end up with

an odd combination of general and specific. You'll have specific deadlines of when to expect real improvement. Maybe you'll have a concrete objective to have all your business analysts sit with your team in a shared workspace. But then you'll have general guidelines on how to reach that objective. You may decide to have everyone in that shared workspace receive coaching on the benefits of sitting together. You could also just make it a simple matter of rearranging desks.

This combination of specific and general guidance gives your plan enough structure to be useful, but enough flexibility to allow teams to adapt. No change management plan will survive implementation; that is, your plan will change as you implement it, and that's okay. The trick is to spend just enough time planning to make your organization more agile, but not so much that you steal away time from the implementation or make your plan so restrictive that it undermines the agile mindset.

Setting the stage for business agility

A growing movement among businesses is to extend enterprise agility from product delivery to the entire organization in order to achieve *business agility.* This movement is really about "agile management" — taking agile ideas that have worked well for product development and using them to run an entire organization. Business agility is about "agilizing" every part of your organization.

The best way to think about the relationship among agile, business agility, and enterprise agility is to look at them as three levels of agile implementation (see Figure 1-2):

- **Agile (at the team level):** You have one-or-two agile teams working on a part of a larger product. In Chapter 2, I introduce you to the various frameworks, principles, and practices that drive agility at the team level, such as Scrum and Extreme Programming.
- **Business agility:** The entire organization adopts an agile mindset and a set of agile principles that guide the way everyone works independently and together.
- **Enterprise agility:** You have dozens or hundreds of agile teams working in concert on a single large product — an enterprise-level product. Some enterprise agile frameworks are simply expansions of team agile approaches; for example, SAFe is Scrum only with more Scrum teams and additional roles and structure to coordinate their work.

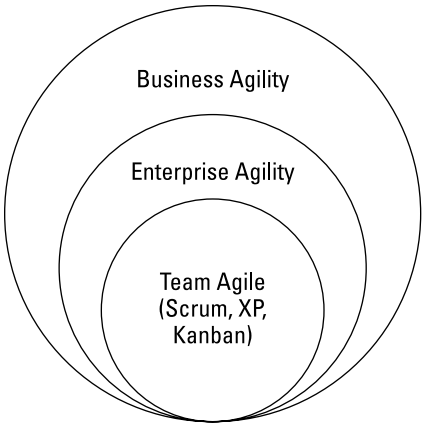

FIGURE 1-2: The three levels of agility.

In general, business agility deals with all domains, including those outside of product development, such as adaptive leadership, organizational design, and budgeting. While the more robust frameworks, including SAFe, touch on these domains, they offer little guidance to help you extend agile into these domains. It's a little like old maps that put dragons in place of uncharted territory with the caption "there be dragons here." They suggest that agility involves changes in other domains, but they don't explicitly describe the changes or offer guidance on how to make those changes.

WARNING

Resist the urge to tackle all three circles at once. Start with a few agile teams. After finding success with those teams, try enterprise agility with a larger product. As you gain success with several teams working together to deliver a whole enterprise solution, you can begin to start thinking about using agile methodologies to rework your entire organization. Don't try to rework your whole organization until you have a proven strategy for delivering enterprise-level products.

REMEMBER

Enterprise agility is *not* business agility. Enterprise agility is about delivering product. All the changes that you make to your organization in terms of frameworks, roles, processes, and practices should sit neatly within the realm of product development. Any changes you make to the overall organization or to organizational leadership will be in the realm of culture and mindset — to make management more receptive to an agile mindset and supportive of the big changes you're introducing to product delivery. Stay focused on delivering better products and not on creating a better organization. Certainly, success in product delivery may lead to an expansion of agile to the entire organization, but start with product delivery and work your way up.

You may find it strange to use practices that were designed for software development to run domains such as human resources, sales, marketing, or legal, but advocates of business agility argue that the accelerating pace of change demands that the entire organization become agile.

Practicing shuhari

Many of the agile and enterprise agile frameworks are influenced by Japanese manufacturing models developed to minimize waste and optimize workflow. Another common agile practice that comes from the Japanese is *shuhari*, a martial arts model of learning and honing one's skills:

- **Shu:** Follow the rules and learn the basics.
- **Ha:** Start to break the rules and put your own learning in context.
- **Ri:** Create your own rules and find your own way.

As you transition your organization's product delivery to enterprise agility, follow the shuhari approach. Here's how:

- **Shu:** Explore the top enterprise agility frameworks, principles, and practices to gain knowledge and wisdom of the commonly accepted approaches to enterprise agility. In other words, learn from the masters.
- **Ha:** Start thinking about how these approaches to enterprise agility would look in your organization. Think about them in the context of what's already in place and in the context of your organization's existing culture. What ideas make sense to you? Where do you think the developers may be wrong? Which ideas are likely to work well (and not so well) in your organization?
- **Ri:** Using all the knowledge and wisdom you've acquired, create your own custom framework tailored to your organization. Adopt the principles and practices that work best for your organization, mix and match, modify, and create your own.

REMEMBER

No two organizations are identical, and none of the enterprise agile frameworks is a one-size-fits-all solution. Use what works, toss what doesn't, and keep your eye on the prize — delivering value to your customers while achieving your business goals. That's what agile is all about.