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

Table of Contents

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.

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