The Addison-Wesley Signature Series

A Mike Cohn Signature Book

Succeeding with Agile

Software Development Using Scrum

Mike Cohn

Foreword by Tim Lister

Praise for *Succeeding with Agile*

"Understanding the mechanics of an agile process is just not enough. Mike Cohn has compiled a superb and comprehensive collection of advice that will help individuals and teams with the intricate task of adopting and adapting agile processes to fit their specific challenges. This book will become the definitive handbook for agile teams."
—**Colin Bird**, Global Head of Agile, EMC Consulting

"Mike Cohn's experience working with so many different organizations in the adoption of agile methods shines through with practical approaches and valuable insights. If you really want agile methods to stick, this is the book to read."
—**Jeff Honious**, Vice President, Innovation, Reed Elsevier

"Mike Cohn has done it again. *Succeeding with Agile* is based on his experience, and all of our experience, with agile to date. He covers from the earliest days of the project up to maturity and offers advice for the individual, the team, and the enterprise. No matter where you are in the agile cycle, this book has something for you!"
—**Ron Jeffries**, www.XProgramming.com

"If you want to start or take the next step in agile software development, this book is for you. It discusses issues, great solutions, and helpful guidelines when scaling up in agile projects. We used the guidelines from this book extensively when we introduced agile in a large, FDA-regulated department."
—**Christ Vriens**, Department Head of MiPlaza, part of Philips Research

"If making the move to agile has always baffled you, then this book will unlock its mysteries. Mike Cohn gives us all the definitive, no-nonsense guide to transforming your organization into a high-powered, innovative, and competitive success."
—**Steve Greene**, Senior Director, Program Management and Agile Development, www.salesforce.com

"Mike Cohn is a great advisor for transforming your software organization. This book is a distillation of everything Mike has learned over the years working with companies that are trying to become more agile. If you are thinking of going agile, pick up this book."
—**Christopher Fry**, Ph.D., Vice President Development, Platform, www.salesforce.com

"Whether you're just starting out or have some Scrum experience under your belt, in *Succeeding with Agile*, Mike Cohn provides a wealth of information to guide you in your quest toward continuous improvement. Throughout the book, concepts are reinforced with practical everyday advice, including how to handle objections and thought-provoking 'things to try now.' An extensive list of recommended readings round this out to be a must have book."
—**Nikki Rohm**, Studio Director Project and Resource Management, Electronic Arts

"The first steps along the path of improving your software process with Scrum are hard, and every step reveals new challenges. In *Succeeding with Agile*, Mike Cohn shows how other organizations have followed this path, how you can learn from them to have a successful implementation of Scrum, and put your organization on the path of constant improvement and delivery of value."

—**Johanes Brodwall**, Chief Scientist, Steria Norway

"I began to recommend Mike Cohn's new book as soon as I began to review it. It seems that as soon as someone asked me a question about some corner of agile development, I would realize that I had just read something excellent in one of Mike's chapters. I am so glad the book is finally out so I can stop saying, 'Mike Cohn has a great new book coming out soon that will talk about this problem.' Now I can say, 'Mike's book is out! Get it!'"

—**Linda Rising**, Coauthor with Mary Lynn Manns of *Fearless Change: Patterns for Introducing New Ideas*

"The title says it all; this is an astonishingly insightful and pragmatic guide to succeeding with agile software development. If you only read one agile book, this is the one. I want to give it to all my clients now!"

—**Henrik Kniberg**, Agile Coach, Agile Alliance Board Member, Author of *Scrum and XP from the Trenches*

"Mike Cohn blends thorough theoretical knowledge with practical hands-on techniques. This is another great agile book from Mike. It will help your team, your department, or your whole organization *Succeed with Agile*."

—**Matt Truxaw**, Application Delivery Manager, Kaiser Permanente IT, Certified Scrum Master

"Mike Cohn's new book is the definitive guide for companies transitioning to Scrum. Its contents are practical and easily accessible. Get it, read it, and apply it!"

—**Roman Pichler**, Author of *Agile Product Management with Scrum*

"*Succeeding with Agile* is at once enormously practical, deeply insightful, and a pleasure to read. It combines great ideas with stories and examples from around the software industry and will appeal to a wide range of readers, from those looking to adopt a new company-wide agile process to developers who just need to improve the way a team is running a single project."

—**Andrew Stellman**, Developer, Project Manager, and Author of *Head First PMP*, *Beautiful Teams*, *Applied Software Project Management*

"Adopting agile methods is hard enough on a greenfield web app in a small company. Transforming an enterprise is another matter. This book captures challenges like the ones we faced and offers insight and, more importantly, practical approaches."

—**Michael Wollin**, Senior Development Manager, Broadcast Production Systems, CNN

"Mike Cohn has put together a fantastic book of guidelines to not only start the Scrum implementation, but to turn your entire corporation into an agile community. I have already implemented many of the recommendations included in this text and have seen a positive influence on the support for Scrum within our organization."
—**James Tischart**, CSM, CSP, CTFL, Vice President, Product Delivery, Mx Logic, Inc

"In *Succeeding with Agile*, Mike Cohn has scoured and sifted through the collective experience and lessons of not only scores of different projects, teams, and organizations from his own agile experience, but also from the experience of countless others. He provides real-world stories from the trenches, useful data and studies, and invaluable insights into what has and hasn't worked well when adopting, adapting, and scaling Scrum. What I like best about the book is where Mike provides wisdom on several different alternatives and approaches and the circumstances in which each is most suitable."
—**Brad Appleton**, Internal Agile Consultant at a Fortune 100 telecommunications company

"I believe Mike Cohn's book will answer many questions and issues that people and teams struggle with in terms of how to improve collaboration, communication, quality, and team productivity. I especially appreciate and agree with Mike's statement that 'there can be no end state in a process that calls for continuous improvement.' This is hard work and it requires persistence, teamwork, and good people. I plan to make *Succeeding with Agile* mandatory reading within my organization, just like we did with his book on *Agile Estimating and Planning*."
—**Scott Spencer**, Vice President Engineering, First American CoreLogic, Inc.

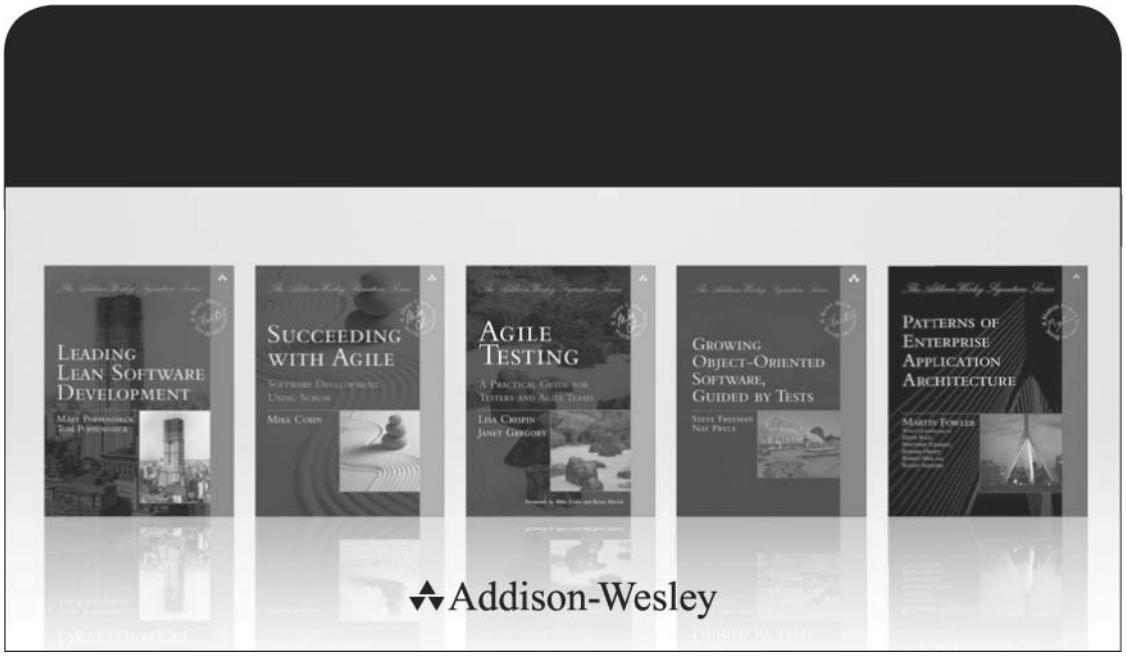
"Mike Cohn has done it again. This comprehensive study of agile software development provides numerous techniques and methodologies to achieve success. I enthusiastically recommend this book to anyone who wants to start using agile or wants to improve their software development process."
—**Benoit Houle**, Senior Development Manager, BioWare (a Division of Electronic Arts)

"There's no doubt that Mike Cohn's new book will become the reference on how to run software projects with Scrum. The book is very carefully crafted and avoids the trap of giving you the one, simple recipe to all your problems. Though mainly centered on Scrum, Mike draws on various other techniques to produce a handbook that is thorough and complete. This is not a hasty mash-up supported by just an act of faith or a single experience. The examples are credible and are a testimony of Mike's vast personal experience of the topic."
—**Philippe Kruchten**, Professor of Software Engineering at University of British Columbia

"This book is packed with useful advice on how your organization can become agile. It's a practical handbook for coaches and change agents who face real-world challenges, such as scaling agile for distributed teams, and who seek to engage with the wider organization. I love the way that Mike Cohn brings the book to life with stories from situations he's faced in the industry and follows up with data and insights from research. I learned something new from every chapter, and I bet you will too."
—**Rachel Davies**, Coauthor of *Agile Coaching*

This page intentionally left blank

SUCCEEDING WITH AGILE



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

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

SUCCEEDING WITH AGILE

Software Development Using Scrum

MIKE COHN

Addison-Wesley

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

The Library of Congress Cataloging-in-Publication data is on file.

ISBN-13: 978-0-321-57936-2
ISBN-10: 0-321-57936-4
Text printed in the United States on recycled paper at Edwards Brothers in Ann Arbor, Michigan.
Seventh Printing: October 2013

Editor-in-Chief
Karen Gettman

Executive Editor
Chris Guzikowski

Senior Development Editor
Chris Zahn

Managing Editor
Kristy Hart

Project Editor
Jovana San Nicolas-Shirley

Copy Editor
San Dee Phillips

Indexer
Lisa Stumpf

Proofreader
Karen Gill

Publishing Coordinator
Raina Chrobak

Cover Designer
Alan Clements

Compositors
Jake McFarland
Bumpy Design

To Laura, Savannah, and Delaney
for making me the one who knows.

This page intentionally left blank

Contents

This page intentionally left blank

Foreword

All the time I hear people talking about software projects as journeys, and I think they are implying that software projects are not just journeys, but they are journeys into the unknown. We start with funding from a sponsor, muster together a stout-hearted crew, head out in what we guess might be a useful direction, and the rest is The Odyssey. We live the tales of the brave Odysseus: tales of Lotus Eaters, the Cyclops, Circe, the Sirens, Scylla, and Calypso. We succeed or fail only with the help or rage of the gods. How wonderfully romantic, and how perfectly silly.

I think that the more appropriate analogy along this line is the project as an expedition. We have a goal or a short list of goals. We have some well-proven maps; we have some vaguer ones, too. We have the advice and journals from those who have been out there and made it back to tell their stories.

We don't walk out the door and face the unknown; but on the other hand, there are some big question marks, and these bring us into a high-risk position. We accept these risks, because if the expedition can succeed there are surely significant rewards. We have skills, but there are uncertainties.

How do we deal with this? I recommend that we look back, oh, about 300 years, to the York Factory on Hudson Bay in Canada. At that time this was the headquarters of the Hudson Bay Company. The Hudson Bay Company's main line of business was to be the supplier of all necessary provisions for fur traders going out on, you guessed it, expeditions, from Hudson Bay. The fur traders developed a great way to start an expedition, and it was called "The Hudson Bay Start." Having done their one-stop shopping at The Company, the fur traders would go out of Hudson Bay only a mile or two and set up camp. Why? Certainly not to set up traps; they wanted to discover what they forgot to bring while they were less than an hour's hike back into town! Being the excellent project person that you are, you know that for the vast majority of time the leather-faced expert fur trader would reappear for another shopping trip.

What the heck does all this have to do with the book in your hands right now? With *Succeeding with Agile,* Mike Cohn has delivered The Hudson Bay Start for agile development. This is it. This is a weather-beaten experienced fur trapper giving you *the* checklist to work through before you begin your expedition. By reading this book, you will find that Mike brings up issues that you never thought of, offers advice on how you might handle situations, and helps you define new roles on your team.

Don't be the only person on your team to read this book; with self-organizing teams anyone can be expedition leader at any given time. This book is going to lead to many very interesting discussions; I guarantee it.

I worry a bit that I am saying that Mike has handed you a book without choices for you. He points out early and often that you must make your choices on individual, team, and organizational issues.

Succeeding with Agile is not about having a single successful project; it is about how agility can transform an organization. I guess in Hudson Bay terms, it's about how to have a great career as Voyageurs.

If you have any lingering doubts about Mike as an experienced expedition leader, notice that his company is Mountain Goat Software.

Tim Lister
Principal, The Atlantic Systems Guild, Inc.
New York City

Acknowledgments

I owe a tremendous debt to my official reviewers: Brad Appleton, Johannes Brodwall, Rachel Davies, Ron Jeffries, Brian Marick, and Linda Rising. They read and commented on the entire manuscript, sometimes multiple times. Each offered tremendously valuable insights that have immeasurably improved the book.

Special thanks also to Tod Golding, Kenny Rubin, Rebecca Traeger, and my wife, Laura, who spent hours discussing the table of contents with me. There were times we thought those conversations would never end.

There's no way to thank Rebecca Traeger enough. She is a miracle worker as an editor, adviser, and sounding board. As she is the former editor for the Agile Alliance and the Scrum Alliance, I contend that she is the best-read person in the agile world. She's also the world's greatest editor. She worked wonders with this book, doing more slicing and dicing than a Veg-O-Matic on a late-night infomercial. This book is significantly better for her involvement in it.

Wow. A foreword by Tim Lister. I'm incredibly honored. I've known Tim for a handful of years, and so I e-mailed him to ask if he'd write the Foreword. I didn't know it, but he was vacationing at the time I e-mailed him and so he replied a week later. I saw the e-mail reply first on my phone, which only displayed the first two lines. Before I tapped the message to see the full e-mail, I had flashbacks of getting college admission letters—would it be good news or bad news? I was ecstatic when he said yes. I was then doubly thrilled when he had such nice things to say in his Foreword. Thank you, Tim.

My assistant, Jennifer Rai, provided invaluable help throughout this project. From tracking down references, to getting permissions, to keeping my research organized, she did it all. I appreciate her dedication, professionalism, and the consistent thoroughness of her work. I couldn't ask for more in an assistant.

For the past two years I have been posting chapters to this book's website at www.SucceedingWithAgile.com. I have been fortunate to have had a wonderful group of people download, review chapters, and provide comments to me. I would like to thank the following individuals for reading draft chapters posted on that site or for providing anecdotes that made their way into the book: Fridtjof Ahlswede, Peter Alfvin, Ole Andersen, Joshua Boelter, Mikael Boman, Rowan Bunning, Butterscotch, Bill Campbell, Mun-Wai Chung, Scott Collins, Jay Conne, John Cornell, Lisa Crispin, Alan Dayley, Ken DeLong, Scott Duncan, Sigfrid Dusci, Mike Dwyer, Pablo Rodriguez Facal, Abby Fichtner, Hillel Glazer, Karen Greaves, Janet Gregory, Ratha Grimes, Geir Hedemark, Fredrik Hedman, Ben Hogan, Matt Holmes, Sue Holstad, Benoit Houle, Eric Jimmink, Quinn Jones,

Martin Kearns, Jeff Langr, Paul Lear, Lowell Lindstrom, Catherine Louis, Rune Mai, Artem Marchenko, Kent McDonald, Susan McIntosh, Alicia McLain, Ulla Merz, Ralph Miner, Brian Lewis Pate, Trond Pedersen, David Peterson, Roman Pichler, Walter Ries, Adam Rogers, René Rosendahl, Kenny Rubin, Mike Russell, Michael Sahota, George Schlitz, Lori Schubring, Raffi Simonian, Jamie Tischart, Ryan Toone, Matt Truxaw, J. F. Unson, Srinivas Vadhri, Stefan van den Oord, Bas Vodde, Bill Wake, Daniel Wildt, Trond Wingård, Rüdiger Wolf, Elizabeth Woodward, Nick Xidis, Alicia Yanik, and Mauricio Zamora.

Thank you to Jeff Schaich who did a wonderful job creating the illustrations for this book. When I was first introduced to Jeff, I was told he might be as much of a perfectionist as I am. He may be, and his drawings show it.

Stephen Wilbers, author of *Keys to Great Writing*, provided some much needed editing and advice early on. I am thankful for his suggestions and encouragement.

As always, the staff at Pearson was wonderful to work with. Chris Guzikowski showed tremendous patience with me, especially early on when I refused to commit to a deadline of any sort. Chris Zahn provided excellent guidance during those early days when I was working to organize what I wanted to say. Jake McFarland designed the interior of the book and did a wonderful job. Jake also showed tremendous patience with my endless barrage of InDesign questions, for which I am extremely thankful. Raina Chrobak was extremely helpful throughout the project, but especially down the home stretch, which is always a frantic period.

Jovana San-Nicolas Shirley was fantastic as this book's project editor. She kept everything moving smoothly, coordinating each of us involved in the final months of the project. I appreciate her willing replies to my e-mails at all hours of the day and night. San Dee Phillips did a top-notch (or is it top notch?) job for the final copy edit. I thank her for going over the manuscript at exactly the right level and for so carefully finding all the last little errors that really polished the text.

Thank you as well to cover designer Alan Clements. What a beautiful cover! Can you judge a book by its cover? I hope so based on the number of people who have already told me they love this one. Lisa Stumpf did a marvelous job with our indexing. She herself should be indexed under thorough and meticulous. Karen Gill did the final proofreading and was fantastic at finding all the little inconsistencies and problems. Kim Scott of Bumpy Design took care of the final page composition. I appreciate her joining at the end to help all of us make the deadline.

I would also like to thank Chris Guzikowski and Karen Gettman of Pearson for offering me the opportunity to edit a Signature Series of books for Addison-Wesley. I can still clearly remember sitting at Ken Kaplan's place in Ben Lomond in the woods of California in 1985 reading *C Primer Plus*. It was written by Stephen Prata but was part of a series by Mitchell Waite. I didn't know what a series editor did, but it sounded important and cool. Now I'm learning what a series editor does and am incredibly honored by their confidence in me.

My thanks also go to Lyssa Adkins, Lisa Crispin, Janet Gregory, Clinton Keith, Roman Pichler, and Kenny Rubin. Each has written or is writing a book that will be part of this series. We have had many discussions about writing, agile, how to make certain points, and more. Through these discussions, each has improved this book.

A special thank you to all of my clients and to everyone who has ever attended one of my classes. I'm not smart enough to sit around, think big thoughts, and come up with great ideas on my own. Everything I know I've learned from working with teams and observing what worked or from talking with participants in classes. This book would be four pages long if not for you. Thank you.

Thank you to Ken Schwaber, Jeff Sutherland, Mike Beedle, Jeff McKenna, Martine Devos, and others who were there in the earliest days of Scrum. Without them writing about Scrum, presenting about it at early conferences, and talking about it, Scrum wouldn't be what it is today. Thank you as well to all of the trainers and coaches in the Scrum community who push so hard to improve how we do Scrum while pushing just as hard to keep Scrum from becoming more than the simple framework it is. My conversations with you so many of you have influenced me in more ways than you know.

There's no way to thank my family enough for all the sacrifices they made while allowing me the time to work on this book. I couldn't ask for a more wonderful and loving wife than I have in Laura. Our daughters, Savannah and Delaney, remain my practically perfect precious princesses. I cherish every moment with them. And with this book finally done, I promise them many more hours and days doing all the things we haven't done enough of lately—now it's my turn to make you the ones who know how far love goes.

This page intentionally left blank

About the Author

Mike Cohn is the founder of Mountain Goat Software, through which he provides training and consulting on Scrum and agile software development. Mike specializes in helping companies adopt Scrum and become more agile as a way of building extremely high performance development organizations. In addition to this book, he is the author of *User Stories Applied for Agile Software Development, Agile Estimating and Planning,* and books on Java and C++ programming.

With more than 25 years of experience, Mike has previously been a technology executive in companies of various sizes, from start-up to Fortune 40. He has also written articles for *Better Software*, *IEEE Computer*, *Cutter IT Journal*, *Software Test and Quality Engineering*, *Agile Times*, and the *C/C++ Users Journal*. Mike is a frequent speaker at industry conferences and is a founding member of the Agile Alliance and Scrum Alliance. He is also a Certified Scrum Trainer, having co-taught the first Certified ScrumMaster class with Ken Schwaber in May 2003.

For more information, visit www.mountaingoatsoftware.com. Mike maintains a popular blog at blog.mountaingoatsoftware.com. He can also be found on Twitter as mikewcohn and by e-mail at mike@mountaingoatsoftware.com.

This page intentionally left blank

Introduction

This is not a book for those who are completely new to Scrum or agile. There are other books, classes, and even websites for that. If you are completely new to Scrum, start with one of those.[1] Nor is this a book for purists. They can find many blogs that will argue the one, true way of agile or Scrum. This is a book for pragmatists. It is for those who have started with Scrum and then encountered problems or for those who have not yet started with Scrum but who know they want to. They don't need to read again about how to draw a burndown chart or what three answers each person gives at the daily scrum. They need advice on the harder stuff—how to introduce and spread Scrum, how to get people to let go of doing a big design at the start of the project, how to deliver software that works by the end of each sprint, what managers do, and more. If these concerns sound familiar, this is a book for you.

To answer these questions, this book draws on my experience with Scrum over the past 15 years, but especially over the last 4. For the last 4 years, every evening after I spent the day with one of my clients, I would go back to my hotel room and make notes about the problems they were facing, the questions they asked, and the advice I gave. I then followed up, either with return visits or e-mails. I wanted to know for sure what advice was working to solve which problems.

As I collected the questions, problems, and advice, I was able to look for common themes. Some obstacles were completely unique to one client or one team. Others were more prevalent and repeated across many teams and organizations. It is these more universal problems—and my advice on overcoming them—that form the basis of this book. This advice is particularly evident in two ways: First, most chapters include boxes labeled *Things to Try Now.* These re-create the advice I found myself giving most often or that was most helpful in particular situations. Second, most chapters also include boxes labeled *Objection*. I have tried in these boxes to reproduce a typical conversation in which someone disagreed with the point I was making at the time. As you read these objections, try to hear the voice of some of your coworkers. I suspect you have heard many of the same objections. In these boxes, you will see how I've sought to overcome them.

1 A good starting point is www.mountaingoatsoftware.com/scrum.

What Else I've Assumed About You

Beyond assuming that you understand the basics of Scrum and now want to either introduce it into your organization or get good at it, I assume that you have some influence within the organization. That doesn't mean I have aimed this book at directors, vice presidents, and the CEO. The type of influence I am assuming is just as likely to come from your personality and individual credibility with your coworkers as it is to come from whatever job title is on your business card. Sure, having a fancy title can help. But as we'll see, the type of influence needed to succeed with Scrum more often comes from opinion leaders.

How This Book Is Organized

When I began this book four years ago, my working subtitle was *Getting Started and Getting Good,* as those were the two things I really wanted to help with. In collecting anecdotes and giving advice, I realized that getting started and getting good at Scrum are the same thing. There are not separate techniques we apply to start and then different techniques we use to get good at it.

Part I is about getting started—it includes advice on whether to start small or convert everyone at once, how to help people move from being aware that a new process is needed to desiring change to having the ability to do it, and how to select initial projects and teams. You will use the basic mechanisms introduced in this section not only to get started but also to get good. Among these are the improvement communities and improvement backlogs of Chapter 4, "Iterating Toward Agility."

In Part II, I focus on individuals and the changes each needs to make as part of the process of adopting Scrum. Chapter 6, "Overcoming Resistance," describes the type of resistance some individuals may exhibit. In it, I offer advice for thinking about why someone is resistant and then provide guidance on how to help the person get past the resistance. Chapters 7 and 8 describe the new roles that exist on a Scrum project and the changes necessary in the traditional roles, such as programmer, tester, project manager, and so on. Chapter 9, "Technical Practices," describes some of the technical practices (continuous integration, pair programming, test-driven development, and so on) that should be used or at least experimented with and that can change much of how individuals approach their day-to-day work.

In Part III, we expand outward from individuals to teams. We look first at how to structure teams to best achieve the benefits of Scrum. Next, in Chapter 11, "Teamwork," I cover the nature of teamwork on a Scrum project. In Chapter 12, "Leading a Self-Organizing Team," we look at what it means to lead a self-organizing Scrum team. In that chapter, I provide specific advice for what ScrumMasters, functional

managers, and other leaders can do to help a team self-organize for success. Chapters 13–15 round out Part Three with a discussion of sprints, planning, and quality.

Part IV expands our focus outward once more, this time to the organization. In Chapter 17, "Scaling Scrum," we take an extended look at what is necessary to scale Scrum up to work on large, multi-team projects. In Chapter 18, "Distributed Teams," we consider the additional complexities of distributed teams. Then, in Chapter 19, "Coexisting with Other Approaches," we add yet more complexity by discussing how to make Scrum work when part of the project uses a sequential process or when there are compliance or governance requirements. Part IV concludes with Chapter 20, "Human Resources, Facilities, and the PMO," focusing on special considerations of the impact of Scrum on an organization's human resources, facilities, and project management office groups.

Part V contains two chapters. Chapter 21, "Seeing How Far You've Come," summarizes various approaches to measuring how far an organization has progressed in becoming agile. Chapter 22, "You're Not Done Yet," concludes the book with the reminder that being agile requires continuous improvement. It doesn't matter how good you are today; to be agile you must be better next month.

A Note on Some Terms

As with most things, writing about Scrum is harder than talking about it. It is too easy to misinterpret a sentence or take one sentence out of context. To avoid these problems, I have tried to be careful and precise in my use of certain terms. I use the word *developer*, for example, to refer to anyone on the development side of the project. This includes programmers, testers, analysts, user experience designers, database administrators, and so on.

The word *team* poses its own challenges. It, of course, includes the developers, but does *team* include the ScrumMaster and product owner? Naturally, this depends on the context. When I have wanted to be especially clear, I use *whole team* to refer to everyone: developers, product owner, and ScrumMaster. However, slavish use of *whole team* would have reduced the readability of the book. So you will encounter *team* as well, but usually in places where the context makes it sufficiently clear which group I'm referring to.

In referring to Scrum and agile teams, I have also needed a term to refer to those teams that are neither. In various places, I have used *sequential, traditional,* and even *non-agile*. Each conveys a slightly different meaning and is used appropriately.

How to Use This Book

Many books have a heading like the one above this sentence. But those headings usually say How to *Read* This Book. The best way to read this book is to use it. Don't just read it. When you encounter a *Things to Try Now* section, try some of them. Or note them and try them at your next retrospective or planning meeting, if that is what I recommended.

It is not necessary to read the book in order. In fact, there could well be entire chapters you do not need to read. If in your organization's quest to become good at Scrum, you have no significant problems with planning and no distributed teams, then skip or skim those chapters. I do, however, recommend that everyone read at least the first four chapters and read them in order. They lay the foundation for much of what follows.

In Chapter 4 you will be introduced to the idea of improvement communities and improvement backlogs. An improvement community is a group of like-minded individuals who are passionate about driving improvements in a particular area. One improvement community could form when three people passionate about the product backlog decide to collect best practices and advice to share across teams. Another improvement community could include hundreds of people interested in improving how your organization tests its applications. An improvement backlog is exactly what it sounds like—a prioritized list of things that an improvement community would like to help the organization get better at.

One of my hopes is that improvement communities—including the Enterprise Transition Community that guides and energizes the transition effort—will use this book to load their improvement backlogs. In fact, many of the top-level section headings have been deliberately worded so that those headings can go right onto an improvement backlog. As examples, consider "Shift from Documents to Discussions" in Chapter 13, "Prepare in This Sprint for the Next" in Chapter 14, and "Automate at Different Levels" in Chapter 16.

As a long-time Scrum trainer and consultant, I have worked with hundreds of teams and organizations, and I've come to believe that success with Scrum is possible for every organization. Some will have a harder time than others. Some will be challenged by a rigid corporate culture. Others will confront entrenched, difficult personalities facing personal loss. The lucky ones will have supportive leadership and passionately engaged employees. What each of these organizations will have in common, though, is the need for pragmatic and proven advice. I have written this book with the hope of providing it.

Part I

Getting Started

Willingness to change is a strength
even if it means plunging part of the company
into total confusion for a while.

—Jack Welch

This page intentionally left blank

Chapter 1

Why Becoming Agile Is Hard (But Worth It)

Many software development organizations are striving to become more agile. And who can blame them? Successful agile teams are producing higher-quality software that better meets user needs more quickly and at a lower cost than are traditional teams. Besides, who wouldn't want to be more agile? It just plain sounds good, doesn't it? It is almost as though one cannot be too thin, too rich, or too agile. But beyond the buzzword and hype, organizations that take becoming agile seriously by adopting a process such as Scrum are seeing dramatic benefits.

They are seeing significant gains in productivity with corresponding decreases in cost. They are able to bring products to market much faster and with a greater degree of customer satisfaction. They are experiencing greater visibility into the development process, leading to greater predictability. And for them, out-of-control, will-it-ever-be-done projects have become a thing of the past.

One company to realize these benefits by adopting Scrum is Salesforce.com. Founded in 1999 in a San Francisco apartment, Salesforce.com is one of the true, lasting dot-com-era success stories. With revenue of more than $450 million and 2,000 employees in 2006, Salesforce.com had noticed the frequency of its releases had dwindled from four a year to one a year. Customers were getting less and waiting longer to get it; something needed to be done. The company decided to transition to Scrum. During the first year of making the switch, Salesforce.com released 94% more features, delivered 38% more features per developer, and delivered over 500% more value to its customers compared to the previous year (Greene and Fry 2008). In the ensuing two years, revenue more than doubled to more than $1 billion. With results like these, it is not surprising that so many organizations have transitioned to Scrum. Or at least tried to.

I say "tried to" because transitioning to Scrum and other agile methods is hard—much harder than many companies anticipate. The changes required to reap all of the rewards being agile can bring are far reaching. These changes demand a great deal from not only the developers but the rest of the organization as well. Changing practices is one thing; changing minds is quite another. It is my aim in this book to show not only how to transition well but also how to succeed long term.

I've personally witnessed several failed agile adoptions that could have been prevented. The first was in a company that had spent more than a million dollars on its transition effort. Executives brought in outside trainers and coaches and hired five people into an "Agile Office" to which new Scrum teams could turn for advice. The company's failure was the result of thinking that the implications of adopting Scrum would be restricted to only the development organization. The executives who initiated this transition thought that educating and supporting developers would be sufficient. They failed to consider how Scrum would touch the work of salespeople, the marketing group, and even the finance department. Without changes to these areas, organizational gravity pulled the company back where it had started.

For completely different reasons, Josef ultimately failed at introducing Scrum to his company. A newly promoted and first-time project manager, Josef was instantly attracted to Scrum because it fit his natural management style. Josef easily convinced his team—who had all been his peers as little as one month before—to try Scrum on their new project. The project was wildly successful, earning accolades for the team and winning Josef the chance at a much larger project. Josef introduced the new project team to Scrum, and most members were willing to try the new approach. Although those working on the project were happy to use Scrum, some of the functional managers to whom they reported got nervous about what Scrum might mean to their careers. Josef's luck ran out. The functional managers—in particular the directors of quality assurance and database development—banded together and convinced the vice president of engineering that Scrum was inappropriate for projects of the complexity and importance being done in their company.

Caroline fared a little better. A vice president of development in a large data management company, Caroline had more than 200 developers in her organization. After seeing the benefits of Scrum on one project, she excitedly launched an initiative to introduce Scrum across her division. All employees were provided with training or coaching. Within a few months nearly all teams were producing working software at the end of each two-week sprint. This was great progress. When I visited this company a year later, though, the employees had failed to make any additional headway. To be sure, teams were producing higher-quality software and doing it a bit faster than they had before starting with Scrum, but her company's gains were only a fraction of what they could have been. Caroline's company had forgotten that continuous improvement is part of Scrum.

Frightening, isn't it? Each of these failures was a well-intentioned effort to transition to Scrum. Yet all the good intentions in the world could not keep them from failing. Don't worry, though. Transitioning to Scrum may be hard, but it's entirely possible with the right approach. In this chapter we examine why transitioning to any agile development process, including Scrum, is especially difficult.

We detail some of the challenges that derailed the companies I've mentioned. Most important, though, we look at the reasons why the benefits of becoming an agile organization are more than worth the effort.

Why Transitioning Is Hard

All change is hard. I've seen employees in an uproar over something so small as a change in their company's healthcare plan. Larger changes can be even more painful. But there are certain attributes of transitioning to Scrum that make it more difficult than most other changes. They are as follows:

- Successful change is not entirely top-down or bottom-up.
- The end state is unpredictable.
- Scrum is pervasive.
- Scrum is dramatically different.
- Change is coming more quickly than ever before.
- Best practices are dangerous.

Successful Change Is Not Entirely Top-Down or Bottom-Up

Successful organizational change cannot be fully top-down or bottom-up. In a top-down change, a powerful leader shares a vision of the future and the organization follows the leader toward that vision. Imagine a charismatic, respected, and powerful leader such as Steve Jobs telling his Apple employees that they are moving beyond computer hardware and software to dominate digital music. His reputation and style might have pointed the company in a new direction, but that alone would not have been enough to pull off such a monumental feat. Change management expert John Kotter agrees.

> No one individual, even a monarch-like CEO, is ever able to develop the right vision, communicate it to large numbers of people, eliminate all the key obstacles, generate short-term wins, lead and manage dozens of change projects, and anchor new approaches deep in the organization's culture. (1996, 51–52)

By contrast, in a bottom-up change, a team or some individuals decide that a change is needed and they set about making it happen. Some teams undertake a bottom-up change with an "ask for forgiveness later" attitude. Others flaunt that they are breaking the rules. Still others attempt to fly under the corporate radar as long as possible.

Most successful changes, and especially a change to an agile process like Scrum, must include elements of both top-down and bottom-up change. Mary

Lynn Manns and Linda Rising agree, writing in *Fearless Change*, "We believe that change is best introduced bottom-up with support at appropriate points from management—both local and at a higher level" (2004, 7). An organization attempting to transition to Scrum without support from the top will encounter resistance that cannot be overcome from below. This usually occurs as soon as the new Scrum process begins to affect how areas outside the original team do their work. In response, middle managers protect their departments by striking out against changes created by Scrum. Top-down support will be needed to remove these kinds of impediments and obstacles.

Similarly, without bottom-up engagement, the transition will feel like sitting under a ceiling fan in an open-air restaurant in Mexico: just a bunch of hot air blowing down from above. When this happens, individuals resist being told what to do. Bottom-up participation will be needed because it will be the individual team members who work through the issues of discovering how Scrum will work best within their organization.

Key to any successful adoption of Scrum will be combining elements of both bottom-up and top-down change.

The End State Is Unpredictable

Perhaps you've read a book on Extreme Programming and have decided that is the right approach for your company. Or maybe you attended a Certified ScrumMaster training course and think Scrum sounds good. Or maybe you read a book on a different agile process, and it sounds perfect for your organization.

In all likelihood, you're wrong.

None of these processes as described by their originators is perfect for your organization. Any may be a good starting point, but you will need to tailor the process to more precisely fit the unique circumstances of your organization, individuals, and industry. Alistair Cockburn concurs: "Having a chance to change or personalize a process to fit themselves seems to be a critical success factor for a team to adopt a process. It's the act of creation that seems to bind teams to 'their own' process."[1]

You may have a clear vision of what "doing Scrum" means to you, and you may get others to buy into exactly that same vision, but where the organization ends up is likely to be somewhat different. In fact, to even refer to end states in a Scrum transition is incorrect; there can be no end state in a process that calls for continuous improvement.

This creates a problem for an organization that wants to transition to Scrum through a traditional change approach that relies on gap analysis and then on closing the identified gaps. If we cannot anticipate the end state of a Scrum

1 This and all other uncited references are personal communications between the speaker and me.

transition, we cannot identify all of the gaps between there and the current state. So, a gap analysis–driven change approach will not work. The closest we can come is to identify gaps between where we are now and an improved, intermediate state.

After identifying these smaller gaps, though, we are still left with the problem of how to close them. It is difficult (and often impossible) to predict exactly how people will respond to the many small changes that will be needed on the way to becoming agile. Teamwork expert Christopher Avery views organizations as living systems.

> We can never direct a living system, only disturb it and wait to see the response…. We can't know all the forces shaping an organization we wish to change, so all we can do is provoke the system in some way by experimenting with a force we think might have some impact, then watch to see what happens. (2005, 22–23)

So, a transition to Scrum cannot be a process that "articulates and defines the entire change process required to bridge the gap between 'as is' and 'to be' and creates tactical plans," as I read in a traditional change management book (Carr, Hard, and Trahant 1996, 144–5). Creating such a plan would require leaping two impossible hurdles: first, knowing exactly where we'll want to end up; and second, knowing exactly the steps to get there. Because we cannot overcome these impossibilities, the best we can do is adopt a "provoke and observe" approach (Avery 2005, 23) in which we try something, see if it moves us closer to an intermediate, improved state, and if so do more of it. These pokings and proddings of the organization are not random. They are carefully selected based on experience, wisdom, and intuition to drive a successful transition to Scrum.

SEE ALSO

Chapter 4, "Iterating Toward Agility," will describe the overall process I recommend when adopting Scrum.

Scrum Is Pervasive

When a change is isolated, when it doesn't affect everything a person does, that change is often easier to introduce into an organization. Consider the case of an organization using a non-agile process that decides to introduce a mandatory operational readiness review before an application is deployed onto the company's web servers. This is a relatively isolated change. Sure, there will be some developers who will hate the new procedure and will complain, perhaps loudly. But, when it comes down to it, this is not a pervasive change. Even if they don't like this change, they can still continue doing the majority of their work unscathed.

Consider now the case of a developer transitioning to Scrum. This developer has to work on smaller pieces of work at a time to complete something by the end of each timeboxed sprint. The developer might have to write automated tests to go with each new bit of code. She might even alternate short bouts of testing and coding in something called test-driven development. And she might need to do all this with her headphones off while pair programming. These are fundamental

changes. They aren't something relegated to a few hours a day or week, as code inspections might be. This type of fundamental change is difficult because it pervades everything about a developer's workday. Resistance will be greater because the impact is greater.

Adopting Scrum is pervasive in a second way as well. Being agile will have implications to the organization that reach far outside the software development department. Introducing the operational readiness review would almost certainly not impact finance, sales, or other departments. But each of those departments can be impacted by Scrum. Finance groups will have to reconcile company policies on capitalizing or expensing with the way Scrum projects run. Sales will want to consider altering how they communicate date and scope commitments and may change how they structure contracts. With more groups affected by a move to Scrum, there is more chance for resistance and certainly more chance for misunderstandings. These add up to make transitioning to Scrum harder than other changes.

SEE ALSO

The impact of Scrum on other groups, such as finance, operations, human resources, and others is discussed in Chapter 20, "Human Resources, Facilities, and the PMO."

Scrum Is Dramatically Different

Not only do the changes created by adopting Scrum pervade everything development team members do, but also many of the changes go against much of their past training. Many testers, for example, have learned that their job is testing for compliance to a specification. Programmers have been trained that a problem is to be analyzed in depth and a perfect solution designed before any coding begins. On a Scrum project, testers and programmers need to unlearn these behaviors. Testers learn that testing is also about conformance with user needs. Programmers learn that a fully considered design is not always necessary (and sometimes not even desirable) before coding begins. Abby Fichtner, who shares her thoughts on her Hacker Chick blog, has told me she agrees with how hard this adjustment can be for programmers.

> Getting used to emergent design is hard because it feels like you're going to be just hacking! And if you've prided yourself on being a very good developer and always doing well-thought-out designs, it turns your whole world upside down and says "no, all those things you thought made you great, now those same things actually make you a bad developer." Very world-rocking stuff.

Because transitioning to Scrum involves asking people to work in ways that are unfamiliar and run counter to training and experience, people are often hesitant, if not outright resistant, to the change. Consider, for example, the case of Terry, a senior and respected programmer in his company. Terry had participated in a hands-on full-day class on test-driven development and was convinced of its benefits. An enthusiastic Terry returned to the office expecting to stop doing big,

SEE ALSO

Emergent design and test-driven development are discussed in Chapter 9, "Technical Practices."

up-front designs and allow design to emerge through the use of test-driven development. It didn't go as smoothly as he thought it would. He wrote me an e-mail describing his deflating experience.

> Getting the other programmers to even try test-driven development was much harder than I thought. I tried pushing it as a way to skip the long up-front design phases we'd become accustomed to, but failed miserably. After a few months I got the other developers to start writing tests first, but only because it was a good idea on its own. They still wouldn't abandon the lengthy up-front design phase. It took me another year to make much progress shortening that, and we could still go much shorter.

Change Is Coming More Quickly Than Ever Before

Back in 1970 Alvin Toffler coined the term *future shock*, saying that it is the disorientation people feel when confronted with "too much change in too short a period of time" (1970, 4). Human, and therefore organizational, capacity to change is limited—ask people to change too many things at the same time and they shut down; the shattering stress and disorientation of future shock kicks in.

In many organizations, employees have been suffering from future shock for years. Teams are asked to do more with fewer people. Outsourcing and distributed teams have become increasingly common. These adjustments were preceded by the rush to move applications to a client/server model, then onto the web, and then into services. Add to these the constant, and constantly accelerating, rate of change in technology itself—new languages, new tools, new platforms—and future shock is now. It should not be surprising that transitioning to Scrum can often be the change that pushes people into future shock. The pervasive nature of adopting Scrum and the fundamental changes it causes in how people work and interact have a higher risk of triggering the future shock effect.

Best Practices Are Dangerous

With most organizational change, after someone figures out the right or best way to do something, that way of doing it is captured as a "best practice" and shared with everyone else. For some types of work, collecting and reusing best practices is a tremendous aid to the change effort. An organization that is selling a product to a new type of customer may, for example, capture best practices for overcoming objections from potential customers. When transitioning to Scrum, however, collecting best practices can be dangerous.

Like sirens singing to us from the rocks, best practices tempt us to relax and stop the effort of continuous improvement that is essential to Scrum. Taiichi Ohno, originator of the Toyota Production System, has written that "there is something

called standard work, but standards should be changed constantly. Instead, if you think of the standard as the best you can do, it's all over." Ohno goes on to say that if we establish something as the "best possible way, the motivation for kaizen [continuous incremental improvement] will be gone" (1982).

Although team members should always look to share with one another their newly discovered good ways of working, they should resist the urge to codify them into a set of best practices. One example of a best practice gone awry is the company that decided that all daily scrums needed to be held no later than 10:00 a.m. I found this an extremely unnecessary dictate. I'm not entirely sure what purpose the dictate served. But many employees took the rule to be further proof that "Scrum is all about micro-management."

THINGS TO TRY NOW

- ❑ Think about your current transition to Scrum. Are you just getting started, in the middle, or feeling like you're nearing the end of the transition push? No matter where you are, identify the primary obstacle you think may be holding you back from the next level of success.

Why It's Worth the Effort

Despite all the reasons why transitioning to Scrum can be particularly difficult, stakeholders in companies that have made the transition are happy they've done so. One reason stakeholders are so satisfied is that time-to-market is reduced when using an agile process like Scrum. This faster time-to-market is enabled by the higher productivity of agile teams, which is in turn the result of the higher quality seen on agile projects. Because employees are freed up to do high-quality work and because they see their work delivered sooner into the hands of waiting users, job satisfaction goes up. With higher job satisfaction comes more engaged employees, which leads to more productivity gains, initiating a virtuous cycle of continued improvement.

The rest of this chapter looks in more depth at these claims. In doing so I present evidence in support of each. Some of the evidence is anecdotal and drawn from my experience, experiences of my clients and colleagues, or experiences reported in magazines or at conferences. Additionally, though, the claims are supported by data from the following sources:

- A rigorous comparison of 26 agile projects against a baseline database of 7,500 primarily traditional development projects. This study was conducted by Michael Mah, managing partner of QSM Associates (QSMA), which has been collecting productivity, quality, and other metrics on projects for more than 15 years. The agile projects Mah studied ranged in size from 60 to 1,000 people (Mah 2008).

- Various academic and research papers, including aggregate research by David Rico, Ph.D., who surveyed 51 published studies of agile projects (2008).
- An online survey of more than 3,000 people conducted by agile tool vendor, VersionOne (2008), and another of 642 people conducted by *Dr. Dobb's Journal* (Ambler 2008a), a popular development magazine. Each survey was conducted in 2008. Industry surveys such as these cannot, of course, be taken as definitive. Individuals opting to take such surveys are probably predisposed toward favorable views of agile. Results from these surveys are presented because they are more representative than conclusive. These surveys will be referenced as VersionOne and DDJ in the sections that follow.

SEE ALSO

Data from this chapter is summarized in Microsoft PowerPoint and Apple Keynote presentations available at www.succeeding-withagile.com.

In the following sections we look at these reasons why transitioning to an agile process like Scrum is worthwhile:

- Higher productivity and lower costs
- Improved employee engagement and job satisfaction
- Faster time to market
- Higher quality
- Improved stakeholder satisfaction
- What we've been doing no longer works

Higher Productivity and Lower Costs

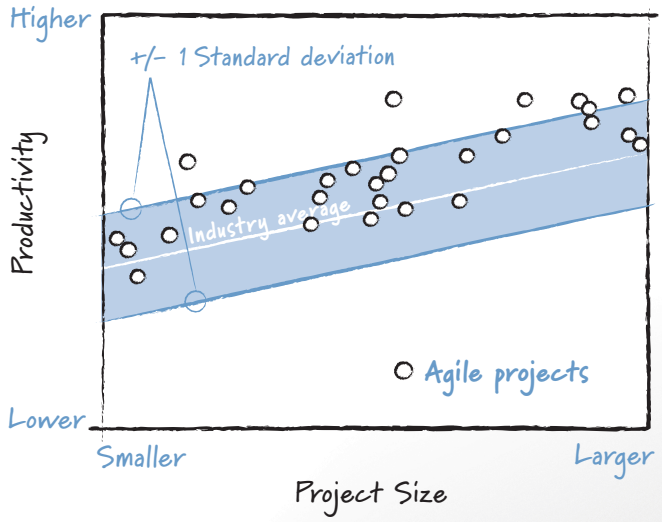
There is unfortunately no universally agreed-upon measure of productivity. Martin Fowler has gone so far as to say that measuring productivity of developers is impossible (2003). And although I agree with Fowler, I do think it is possible to measure proxies or stand-ins for productivity. Some teams use the number of lines of code as a proxy for productivity. Others use as a proxy the number of function points delivered or simply the number of features delivered, ignoring that not all features are the same size. Are there problems with these proxies? Absolutely. But I think the usefulness of proxy productivity measures is justified if we can reasonably make the assumption that data has not been gamed by teams fabricating lines of code or function points by duplicating code, failing to take advantage of reuse, or so on. In many cases, especially those involving large data sets as the QSMA study does, I think this is a reasonable assumption.

QSMA calculates a productivity index for the projects in its database. This index takes into consideration effort, schedule, technical difficulty, and more and is an attempt to help make cross-team comparisons more meaningful. In his comparison between agile and traditional projects, Mah found agile projects to be 16%

more productive, an increase that he found to be statistically significant. Figure 1.1 shows the agile projects (as dots) compared to the average productivity and one standard deviation around it in the QSMA database. As you can see, most of the dots are above the industry average line, with a handful of projects more than one standard deviation more productive than the industry average.

FIGURE 1.1

Agile teams are significantly more productive than the industry average. Source: Mah 2008.



The QSMA results are corroborated by both the DDJ and VersionOne surveys. Eighty-two percent of participants in the DDJ survey felt that productivity was somewhat or much higher when using agile methods like Scrum than it was before. Only 5% felt productivity was somewhat or much lower. Seventy-three percent of the VersionOne respondents believed that being agile had significantly improved (23%) or improved (50%) productivity.

It stands to reason that if people are productive, costs will be lower. The VersionOne and DDJ studies both bear this out, as can be seen in Table 1.1.[2]

David Rico's survey of case studies of agile teams published through 2008 is shown in Table 1.2. Rico found that the median reported productivity increase was 88% and the median cost savings was 26%. These indicate solid evidence that agile teams are more productive, which leads to cost savings to their projects.

2 The VersionOne survey asked respondents to answer on a scale that included Significantly Improved, Improved, No Benefit, Worse, and Much Worse. The DDJ survey used a similar scale but used Much Higher, Somewhat Higher, No Change, Somewhat Lower, and Much Lower. For improved readability, all tables in this chapter use the labels from the VersionOne survey.

Development Cost	DDJ	VersionOne
Improved	32%	30%
Significantly Improved	5%	8%

TABLE 1.1
A significant number of survey respondents report that agile improved development costs.

As encouraging as these numbers are, they tell only part of the story. A significant benefit to being agile—but one not reflected here—is that agile teams are less likely to build functionality that is no longer needed. A common criticism of a sequential development process is that by the time the software is delivered, the users no longer need the functionality being provided. Because of the frequent feedback, timeboxed sprints, and ability to reprioritize each sprint, a Scrum team is more likely to work only on features users really need. Were we to include this in our measurement of productivity, we would see even more dramatic results.

Category	Lowest Reported Improvement	Median Improvement	Highest Reported Improvement
Productivity	14%	88%	384%
Cost	10%	26%	70%

TABLE 1.2
Impact of agile on productivity and cost. Source: Rico 2008.

Improved Employee Engagement and Job Satisfaction

One factor contributing to the higher productivity and lower costs on agile projects may be that employees enjoy their jobs more. Fifteen months after adopting Scrum, Salesforce.com surveyed its employees and found that 86% were having a "good time" or the "best time" working at the company. Prior to adopting Scrum, only 40% said the same thing. Further, 92% of employees said they would recommend an agile approach to others. Results such as these are common; many of my clients have done employee satisfaction surveys and always with similar results. In its industrywide survey, VersionOne found that 74% of those surveyed reported morale was improved (44%) or significantly improved (30%).

One reason why employees may enjoy their jobs more is because of the sustainable pace promoted by agile processes. Chris Mann and Frank Maurer of the University of Calgary studied the amount of overtime worked by one team for the year before becoming agile and the first year after (2005). They found that before implementing agile practices, team members worked an average of 19% overtime. After adopting an agile process, that dropped by nearly two-thirds to an average of 7% overtime. Further, even though overtime was occasionally needed even after adopting agile practices, there was less variability in the amount

required, as measured by the standard deviations of the team before and after moving to agile. Johannes Brodwall, an agile software architect, says, "Overtime seems to be much less common after we started with agile. Testers are especially noticing the effect. They used to have extremely chunky workloads."

A lack of overtime is likely just one factor contributing to higher job satisfaction among people working on agile teams. There are also the benefits of having more control over your day-to-day work, seeing the results of your work get used sooner, working more closely with coworkers, creating products that are more likely to meet customer and user expectations, and so on. Employees who are happier with their jobs and with their employers will be more engaged in the work they do. Greater employee engagement will result in numerous benefits to the organization.

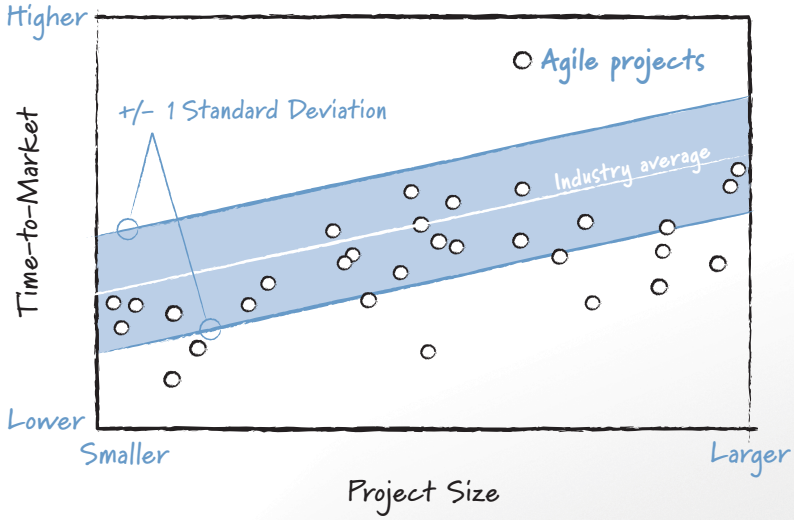
Faster Time to Market

Agile teams tend to release their products more quickly than do traditional teams. According to the VersionOne study, 64% of participants report that time to market has been improved (41%) or significantly improved (23%). The QSMA study comparing 26 agile projects to a database of 7,500 mostly traditional projects found that agile projects have a 37% faster time to market, as shown in Figure 1.2.

Agile teams have faster times to market for two reasons. First, the higher productivity of an agile team allows them to produce functionality more quickly. Second, agile teams are more likely to release incrementally. When stakeholders realize that a team can produce valuable functionality every sprint, they often decide that they do not need to wait for one big-bang delivery of all functionality.

FIGURE 1.2

Agile projects have a 37% faster time to market compared to the industry average. Source: Mah 2008.



Salesforce.com noticed the benefit of this immediately after its rapid transition to Scrum (Greene and Fry 2008). Figure 1.3 shows the cumulative number of features delivered to customers in 2006 (before adopting Scrum) and 2007 (after initiating the transition around the start of the year). This figure shows a simple metric: the raw number of features delivered and when they were delivered and a powerful view of the additional value provided to customers in the first year of using Scrum.

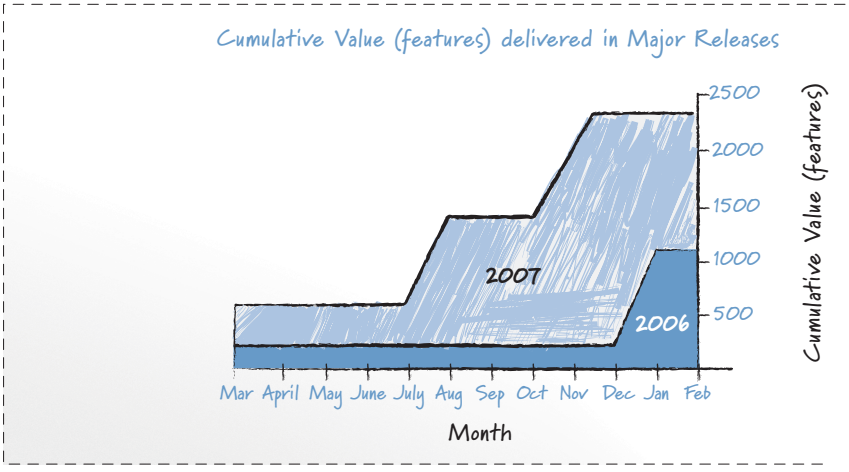


FIGURE 1.3
The cumulative value of features delivered by Salesforce.com in 2006 (pre-Scrum) and 2007 (Scrum).

Higher Quality

If you ask a Scrum team what enables them to be more productive than in their pre-Scrum days, most will say that at least part of their success is that they are consistently producing higher-quality work. Without bugs left behind to drag the team down, they can move quickly and consistently forward. Quality is improved because working at a sustainable pace prevents sloppiness. Quality is also improved through many of the engineering practices such as pair programming, refactoring, and a strong emphasis on early and automated testing.

David Rico's research bears out the claim that agile teams produce higher-quality products. In his survey of 51 published studies of agile projects, he found a minimum quality improvement of 10% and a median improvement of 63%. Rico's research matches my experience at clients where I've been able to measure and report on quality. For example, ePlanServices provides retirement plans to medium-sized businesses. The service is provided largely through a powerful web application. In the first nine months after initiating a Scrum transition, their defect rate per thousand lines of code dropped by 70%.

The VersionOne survey also bears out the claim for higher quality with agile processes such as Scrum. Sixty-eight percent of participants answered that agile had improved (44%) or significantly improved (24%) software quality. Further,

84% of respondents felt that agile had reduced the number of software defects by 10% or more; 30% felt agile had reduced the number of defects by 25% or more. The DDJ survey reported similar results, with 48% saying quality was somewhat higher and 29% saying it was much higher.

Improved Stakeholder Satisfaction

Given all of the benefits of agile processes thus far, it is not surprising that they lead to improved stakeholder satisfaction. The DDJ survey found that 78% of survey participants believe that using an agile process has led to somewhat higher (47%) or much higher (31%) stakeholder satisfaction.

One reason stakeholders are more satisfied by agile processes is because their practices are more friendly toward the shifting priorities that are a fact of life in today's fast-paced, competitive organizations. In the VersionOne study, 92% of participants felt that agile improved the ability to manage changing priorities. Additionally, along with gaining the ability to more easily change priorities, stakeholders on agile projects learn the impact of change. A stakeholder at PetroSleuth, a small development company in the oil and gas industry, found that to be true.

> The Scrum process has led to our being more involved in the daily review and discussion. This has led to us being more aware, and being held accountable earlier in the process for any changes. (Mann and Maurer 2005, 77)

The VersionOne survey looked deeper into additional factors leading to stakeholder satisfaction. Table 1.3 shows the high percentages of survey participants who reported that agile leads to better alignment between the technology and business groups, reduced project risk, better ability to manage changing priorities, and improved project visibility. Steve Fisher, a senior vice president at Salesforce.com and stakeholder to many of the agile teams there, says adopting Scrum has "delivered total visibility, total transparency, and unbelievable productivity…a complete win" (Greene 2008).

TABLE 1.3
Some of the reasons stakeholders are satisfied with agile.

	Improved	Significantly Improved
Enhanced ability to manage changing priorities	41%	51%
Improved project visibility	42%	41%
Improved alignment between IT and business goals	39%	27%
Reduced project risk	48%	17%

What We've Been Doing No Longer Works

One final reason to consider changing to Scrum is if your current development process is no longer working. When a process that has worked in the past stops working, a common tendency is to do more of it. This was certainly the case at Yahoo!, where chief product officer Pete Deemer was one of the first to recognize the need for change.

> Originally, Yahoo! tried Scrum purely out of desperation—the waterfall approach was clearly not working—and a year-long attempt to do the waterfall "better" through more thorough planning and analysis, more in-depth documents, more sign-offs, and so on was making things worse, not better. For the teams that saw benefits, which were most of the teams that tried Scrum, the benefits were visible almost immediately.

Clinton Keith, former chief technology officer at High Moon Studios, developer of console-based video games, tells a similar story.

> As successful project managers at a well-funded startup, we felt we could "apply more waterfall" to our ambitious new projects. This had the opposite effect of what we hoped for and the projects spiraled out of control. Our assumptions were wrong and forced us to rethink how we were managing projects.

THINGS TO TRY NOW

- ❑ Identify the benefits you have gained from using Scrum so far.
- ❑ If you have not yet gathered metrics on quality, employee morale, stakeholder satisfaction, or so on, select a few factors of interest and measure a baseline you can compare against later.
- ❑ If you gathered baseline measurements earlier and have been doing Scrum for at least three or six months, remeasure and see what progress has been made. Create your own "why Scrum is worth it" charts that you can share with other teams as they begin to transition to Scrum or with existing teams who are having difficulty sticking with it.

Looking Forward

Becoming agile is hard. It is harder than most other organizational change efforts I've witnessed or been part of. I started this chapter by laying out some of the reasons why this is so, including the need to change from the top-down and bottom-up simultaneously, the impossibility of knowing exactly what the end state will look like, the dramatic and pervasive changes caused by Scrum, the difficulty of

adding more change on top of all that is already occurring, and the need to avoid turning Scrum into a list of best practices.

Because you're still reading, I can assume that this list of challenges didn't send you away. That's fortunate because there are tremendous advantages to be had by the organization that overcomes the challenges. These include more productive teams, lower costs, happier employees, reduced time to market, better quality, and improved stakeholder satisfaction.

In the next chapter we look more closely at what is involved in moving you, your team, and your organization from the stage where you know change is necessary and you believe that Scrum is the answer to a point where you can begin making real progress and continuous improvements.

Additional Reading

Ambler, Scott. 2008. Agile adoption rate survey, February. http://www.ambysoft.com/surveys/agileFebruary2008.html.

This article presents the results of a survey conducted in February 2008 and goes beyond the results presented here.

Greene, Steve, and Chris Fry. 2008. Year of living dangerously: How Salesforce.com delivered extraordinary results through a "big bang" enterprise agile revolution. Session presented at Scrum Gathering, Stockholm. http://www.slideshare.net/sgreene/scrum-gathering-2008-stockholm-salesforcecom-presentation.

Greene and Fry led the rollout of Scrum at Salesforce.com. They have shared this entertaining slide deck that covers how they did it, what they learned, and what they'd do differently.

Mah, Michael. 2008. How agile projects measure up, and what this means to you. *Cutter Consortium Agile Product & Project Management Executive Report* 9 (9).

This is Mah's comparison of 26 agile projects to his baseline database of productivity data on over 7,500 mostly traditional projects.

Rico, David F. 2008. What is the ROI of agile vs. traditional methods? An analysis of extreme programming, test-driven development, pair programming, and Scrum (using real options). A downloadable spreadsheet from David Rico's personal website. http://davidfrico.com/agile-benefits.xls.

An extensive survey of the available literature on agile projects that summarizes key percentage improvements in productivity, cost, quality, schedule, customer satisfaction, and return on investment.

VersionOne. 2008. The state of agile development: Third annual survey. Posted as a downloadable PDF in the Library of White Papers on the VersionOne website. http://www.versionone.com/pdf/3rdAnnualStateOfAgile_FullDataReport.pdf.

Every year, agile tool developer VersionOne conducts the largest survey of the state of agile adoption. The survey is international in scope and is the broadest view into the use of agile practices.

This page intentionally left blank

Chapter 2

ADAPTing to Scrum

Lori Schubring was among the first to realize that things had to change. An application development manager for a large manufacturing company, Lori realized that its development process had become "so formalized that we hindered our ability to remain flexible for the business. It got to the point where we weren't turning around project requests fast enough" (2006, 27). Aware of the need to change, Lori attended a free, half-day seminar introducing Scrum. What she saw there was a better way to develop software, a framework she thought might help her organization. As such, Lori developed the desire to change to Scrum. Next, she acquired the ability to do it by participating in a ScrumMaster class, attending an agile conference, and visiting a company that had already adopted Scrum. Lori then promoted Scrum to her boss and team, convincing them of its benefits. Finally, Lori transferred some of the implications of her team using Scrum to the rest of her company so that organizational gravity would not pull the team back to where it had started.

Lori's story encapsulates the five common activities necessary for a successful and lasting Scrum adoption:

- **Awareness** that the current process is not delivering acceptable results
- **Desire** to adopt Scrum as a way to address current problems
- **Ability** to succeed with Scrum
- **Promotion** of Scrum through sharing experiences so that we remember and others can see our successes
- **Transfer** of the implications of using Scrum throughout the company

Conveniently, these five activities—Awareness, Desire, Ability, Promotion, and Transfer—can be remembered by the acronym ADAPT.[1] These activities are also

1 The five activities of ADAPT are based on ADKAR (Hiatt 2006), a general model of change that includes the steps of Awareness, Desire, Knowledge, Ability, and Reinforcement. In practice, I have found separating Knowledge and Ability to be unnecessary. In a field such as software development, knowledge without ability is meaningless. Additionally, the Reinforcement step of ADKAR is replaced in ADAPT with separate Promotion and Transfer steps, emphasizing the importance of these activities to a successful transition.

summarized in Figure 2.1, which shows Awareness, Desire, and Ability as overlapping, whereas Promotion and Transfer repeat and occur throughout the transition effort. After you have transitioned, this cycle will continue as you continuously improve.

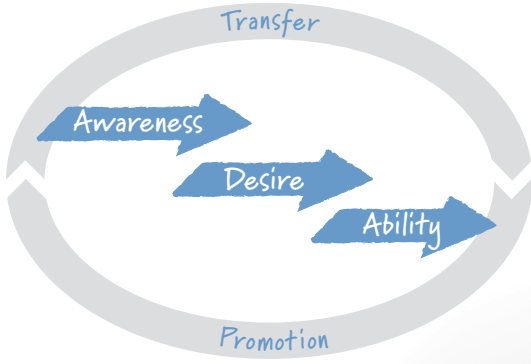


FIGURE 2.1
The five activities of adapting to Scrum.

An organization that successfully adopts Scrum can be thought of as engaging in these activities at multiple levels:

- **Organizationally.** The organization as a whole will engage in these activities. No matter how aware one person or group is, there must be a critical mass of people with a similar awareness before the organization will be able to collectively move forward. In thinking of the ADAPT model at this level, we may speak of a company with an organizational desire to adopt Scrum. Or we may say that our organization currently lacks the ability to do Scrum.
- **As individuals.** Because organizations are made up of individuals, it is important to acknowledge that individuals will progress through the overall transition at different rates. For example, you personally may already have acquired the ability to do Scrum; you've learned some new skills and some new ways of thinking about software development. A colleague, on the other hand, is only starting to become aware that the current approach isn't working.
- **As teams.** Individuals can be aided or hindered in the transition to Scrum by their teams. Teams tend to progress through the ADAPT cycle more or less together. In the same way that studies have shown individuals are more likely to be overweight if their friends are overweight (Thaler and Sunstein 2009), you are more likely to have a desire to do Scrum if the rest of your team does as well.