Agile Adoption Patterns

A Roadmap to Organizational Success

Amr Elssamadisy

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

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

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

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

For sales outside the United States please contact:

International Sales
international@pearsoned.com

This Book Is Safari Enabled

The Safari® Enabled icon on the cover of your favorite technology book means the book is available through Safari Bookshelf. When you buy this book, you get free access to the online edition for 45 days.

Safari Bookshelf is an electronic reference library that lets you easily search thousands of technical books, find code samples, download chapters, and access technical information whenever and wherever you need it.

To gain 45-day Safari Enabled access to this book:

- Go to www.awprofessional.com/safarienabled.
- Complete the brief registration form.
- Enter the coupon code XPNZ-JWYL-ZBBW-RFLT-3MXJ.

If you have difficulty registering on Safari Bookshelf or accessing the online edition, please e-mail customer-service@safaribooksonline.com.

Visit us on the Web: www.awprofessional.com

Library of Congress Cataloging-in-Publication Data:

Elssamadisy, Amr, 1970-

Agile adoption patterns : a roadmap to organizational success / Amr Elssamadisy. -- 1st ed.

p. cm.

Includes bibliographical references and index.

ISBN 0-321-51452-1 (pbk. : alk. paper) 1. Computer software—Development. 2. Agile software development—Management. I. Title.

QA76.76.D47E42 2008

005.1--dc22

2008015486

Copyright © 2009 Pearson Education, Inc.

All rights reserved. Printed in the United States of America. This publication is protected by copyright, and permission must be obtained from the publisher prior to any prohibited reproduction, storage in a retrieval system, or transmission in any form or by any means, electronic, mechanical, photocopying, recording, or likewise. For information regarding permissions, write to:

Pearson Education, Inc.
Rights and Contracts Department
501 Boylston Street, Suite 900
Boston, MA 02116
Fax (617) 671 3447

ISBN-13: 978-032-151452-3
ISBN-10: 0-321-51452-1

Text printed in the United States on recycled paper at RR Donnelly in Crawfordsville, Indiana.
First printing: July 2008

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
Karen A. Gill

Indexer
Erika Millen

Proofreader
Language Logistics, LLC

Publishing Coordinator
Raina Chrobak

Cover Designer
Alan Clements

Compositor
Gloria Schurick

FOREWORD

By Linda Rising*

I was on yet another flight recently—a plane so small that my seat was both on an aisle and a window! So it was easy for me to see my two neighbors across the aisle—both young girls who looked like sisters. They were alone and had obviously made this trip from Houston to Richmond before. They settled in as we started to taxi, and as the small plane popped up and became airborne, they reached in their carry-on bags and pulled out—could it be—yes, books! I was amazed at how amazed I was! Young folks reading! And they read the entire two-hour flight. It completely restored my sometimes-shaky faith in humanity. There is hope! One of the few good things about flying is the chance to have a little time to read and to notice that others are also reading—even youngsters.

There's something about patterns and books. They go together. I remember commenting about this in an early talk about patterns. Afterward, one of the participants came up and started looking through the stack of patterns books I had brought. He said, "Some of my happiest moments have been with books." Well said. Patterns people do like books. They buy books, and they read them. In fact, entire conferences are devoted to reading and talking about patterns, and many of the products of these conferences become books. But nowadays, you might feel there are so many books, and we have so little time. Agreed. My stack of books waiting for their plane trip is growing, and I find myself torn between attractive alternatives each time I assemble my own carry-on bag. So while I'm always happy to look at a book—especially a book about patterns—I always wonder if people will have the time to read it.

There are several reasons why I think Amr Elssamadisy's book will not face a life buried in someone's stack of books-to-be-read-sometime-on-some-flight-somewhere. It's not just because its a book of patterns. It's not just because it's a book about Agile development. I'm a believer in patterns. I'm also a believer in Agile development. Like many other believers, this faith in "a better way" is not enough to convince decision makers who are looking for costs

* Coauthor, *Fearless Change*, with Mary Lynn Manns

and benefits. Even though I earned a Ph.D. in computer science, working in the area of design metrics, and an M.S. in mathematics, I often find it hard to measure these costs and benefits satisfactorily. As the famous British writer Lewis Carroll cautions, "If you dont know where you're going, any road will get you there." [1]

Victor Basili had some recommendations in 1994 in his classic paper "The Goal, Question, Metric Approach" [2].

- Develop a set of corporate, division, and project business goals and associated measurement goals for productivity and quality.
- Generate questions (based on models) that define those goals as completely as possible in a quantifiable way.
- Specify the measures needed to be collected to answer those questions and track process and product conformance to the goals.
- Develop mechanisms for data collection.
- Collect, validate, and analyze the data in real time to provide feedback to projects for corrective action.
- Analyze the data in a postmortem fashion to assess conformance to the goals and to make recommendations for future improvements.

I guess what I'm trying to say is that this book is useful for practitioners who want to follow the advice in the Basili paper and apply this methodology not to metrics but in moving to Agile development. Yes, this book is written as a collection of patterns, but it doesn't get so wound up in being about patterns. Yes, it's about Agile, but it's not evangelizing Agile. The book is practical and readable, and the focus is on business value. The book makes the very wise observation that no one path to Agile (or any other worthwhile goal) is achieved in the same way by all seekers. There are no easy answers here. Amr takes the approach, often found in other Agile books that discuss code, by examining a list of "smells." This might be helpful, not only for pointing out applicable patterns but also for supporting your struggle to determine what your business truly values. This is a worthwhile exercise regardless of whether you are contemplating Agile practices or not.

Having said all that, I want to report that there are no silver bullets in this book. Sorry! Patterns revolve around a context that says when its appropriate to apply the solution. Authors of good patterns also include a signpost that warns the users of the consequences of applying the solution that, even if the context is appropriate, there are no guarantees that we will all live happily ever after. With patterns, as in life, the best advice needs to be considered carefully before rushing in. I'm happy to say that Amr adheres to these pattern guidelines. Even the book itself has a context. I always appreciate that when I'm browsing the pages. This author has done us all a favor by saying what audience he addresses. A quick look at the section "Is This Book for You?" will help you decide whether to buy the book or not.

I'm hoping that if you decide you are part of the target audience, you will buy the book, and, more than that, you will move it to the top of your book stack and read it. I believe that if you are able to do all this, you will find these patterns useful for moving your business to Agile. And that means I'll get to see you at the next Agile conference! Enjoy!

[1] Carroll, L., *Alice's Adventures in Wonderland*, originally published in 1865.

[2] Basili, V., Caldiera, G., and Rombach, H. D., "Goal Question Metric Paradigm," *Encyclopedia of Software Engineering*, pp. 528–532, John Wiley & Sons, Inc., 1994. www.cs.umd.edu/~basili/publications/technical/T87.pdf.

Foreword

By J.B. (Joe) Rainsberger*

Change campaigns are hard, and adopting an Agile approach to delivering software is no different. The processes are stressful for participants, for leaders, and for those who can only stand aside and watch. They're so hard, in fact, that I find it hopelessly optimistic to think that a single book could help people navigate through a successful Agile adoption—until now.

In the pages of this book, you will find concise, practical advice for all facets of adopting the Agile mindset. There is advice on which practices to adopt, how, and when. There is advice on how to foster the level of community involvement that effective software delivery demands. There is even specific advice on how to learn what you need. It is as comprehensive a manual as I have ever seen on the topic: specific enough for you to follow, but not so prescriptive that you forget to think for yourself. It is the closest I have ever seen to the One Book You Need to begin adopting an Agile approach to delivering software.

As someone who has written a "recipes" book, I am partial to the format that the author has employed here. I find that once I have gone past the manifesto or "why question" stage of exploring a new concept, I look for concrete practices to try, described concisely, with several thought-provoking points to consider as I begin my practice. That is what the author has provided here, and it couldn't suit me better. If you are serious about succeeding at delivering software, and you believe an Agile approach is a path to success, then start with *Agile Adoption Patterns: A Roadmap to Organizational Success*. Read it thoroughly, and examine its bibliography. There are considerable riches in these pages, even to mine for years. Get digging!

* (http://www.diasparsoftware.com)
Your guide to software craftsmanship
JUnit Recipes: Practical Methods for Programmer Testing
2005 Gordon Pask Award for contributions to Agile Software Practice

Preface

In this book, you and I focus on adoption of Agile practices. I help you answer basic questions that are on your mind:

- Where do I start?
- What practice(s) are best for my particular environment?
- How can I adopt these practices incrementally?
- What pitfalls should I watch out for?

Is This Book for You?

Are you adopting one or more Agile practices or seriously thinking about trying out one or more practices on your team? Have you read any of the Agile methodology books on Extreme Programming, Scrum, or Test-Driven Development, and are you theoretically convinced about at least trying the practices?

Or perhaps you're coming off your first project and you've been asked to join another team to help them succeed as you did previously. Of course, every project is different. Are the same practices you used the last time going to be as effective on the next project? It depends! This book helps you get past "It depends!" to determine what practices should be adopted and give you some hints how they may need to be adapted.

Or maybe you are unlucky enough to have been part of a failing Agile project (or possibly are still on one). Read this book to get an idea why the practices you are using may not be applicable. Be agile about your Agile practices.

If any of these scenarios fit, this book is for you. It helps you look at the individual practices and their relationships and gives you a strategy that has been used successfully several times on multiple projects by multiple companies. It also provides you with warnings concerning how practices have gone wrong before and how you can recognize and respond to the problems that occur. This is not just one person's opinion or an untried method. All the patterns you will read about here come from *real-world* project experience.

Finally, let me say a few words about who this book isn't for:

- Advanced practitioners who already get Agile practices and are looking for new theories or practices. All the information in this book is collected from the experience of multiple projects, so chances are you've already heard about everything here.
- Beginners who want to start from zero. This book does not adequately describe the practices from ground zero. However, this book will be a good companion to other works that delve more deeply into full Agile practices.

THE PLAN

I give you even more questions that you should consider and answer on your journey toward adopting Agile practices. Does this sound too good to be true? It isn't really. Many of us who have been in the Agile community for several years have figured this out the hard way—by trial and error. This book shares those experiences. Here is an overview of what you will be able to accomplish by reading this book.

- Understand some of the basic drivers or principles and values that underlie all Agile practices and make them successful.
- Focus on business value to the customer. List important areas of value to many customers. An example of a business value would be Reduce Cost.
- Understand symptoms that occur when business value is not being delivered. I'll call these symptoms smells. An example of a smell related to the Reduce Cost business value is Customer Asks for Everything Including the Kitchen Sink.
- Tie these business values and smells to individual Agile practices.
- Use the information in the first four items to decide which practices to adopt to increase your business value and remove the smells present at your company. At this point, you will be able to come up with a coarse-grained adoption strategy for your environment.
- Provide a detailed description of each practice in pattern format and include adoption information for each practice.
- Call out practices that work well together as clusters. Relate these clusters to business values and smells. Describe the clusters and adoption strategies as done for the practices.

STRUCTURE AND CONTENT

This book is organized into several parts and subparts, so a quick overview of the structure and content of those parts is in order. I recommend that you read Chapters 1 through 8 straight through and do all the exercises where applicable. This will give you essential context concerning software development so that you and I are on the same page, take you step by step through the creation of an Agile adoption strategy tailored to your organization's context, and introduce you to the pattern format in which all the practices are presented in the pattern catalog.

After you have finished these eight chapters, use Part 3 of the book as a reference to implement the strategy you've created. Skip around or read straight through. Both will work; you can read the patterns presented independently. Each chapter will help you adopt a particular practice, warn you of pitfalls, and give you references for further reading.

Read the case studies to get a feel for how this approach has translated to other organizations, but beware—it gets messy in real-life situations. Finally, the appendices are chapters that were too useful not to include but didn't quite fit in with the flow of the book. They are short, so feel free to take a look at them at any time throughout your reading.

Part 1: Thoughts about Software Development

Part 1 covers some basic issues of software development and sets the context for the rest of the book. I examine reasons why software development is so difficult. I also look at why adoption of new practices—any practices, not just Agile practices—are difficult and depend on your personal involvement and commitment. Read the chapters in this section and keep the ideas in the back of your mind as you go through the rest of the book.

- Chapter 1, "Learning Is the Bottleneck"
- Chapter 2, "Personal Agility for Potent Agile Adoption"

Part 2: Crafting an Agile Adoption Strategy

Part 2 starts to get into the meat of the problem—picking and choosing Agile practices for your particular context. By the time you are done with these chapters and have completed the exercises, you will have an initial set of practices that your team should start to adopt. Be aware that for the purposes of creating an adoption strategy, I will refer to many practices that are described in the remainder of the book. So don't worry if you have a set of practices to adopt that you do not completely understand yet. Their descriptions are in the later sections.

- Chapter 3, "Business Value"
- Chapter 4, "Smells"
- Chapter 5, "Adopting Agile Practices"

Part 3: The Pattern Catalog

Part 3 is the pattern catalog. The pattern catalog details how to successfully adopt and adapt the practices that you've determined in Part 2 meet your organization's business goals. This section should be used as a reference to put your adoption plan to practice. Read Chapters 6, 7, and 8 and then use the rest on an as-needed basis to execute your adoption strategy. Note that the practices are organized into subparts as well.

- Chapter 6, "The Patterns of Agile Practice Adoption"
- Chapter 7, "Goal"
- Chapter 8, "Cycle"

Part 3.1 Feedback Practices

The feedback practices are predominantly concerned with working as a team and planning functions. They are practices that help you and your team "solve the right problem" by iteratively building your software system and consistently checking whether the system solves the needs of the customer.

- Chapter 9, "Iteration"
- Chapter 10, "Kickoff Meeting"
- Chapter 11, "Backlog"
- Chapter 12, "Planning Poker"
- Chapter 13, "Stand-Up Meeting"
- Chapter 14, "Done State"
- Chapter 15, "Demo"
- Chapter 16, "Retrospective"
- Chapter 17, "Release Often"
- Chapter 18, "Co-Located Team"
- Chapter 19, "Self-Organizing Team"
- Chapter 20, "Cross-Functional Team"
- Chapter 21, "Customer Part of Team"
- Chapter 22, "Evocative Document"
- Chapter 23, "User Story"
- Chapter 24, "Use Case"
- Chapter 25, "Information Radiator"

Part 3.2 Technical Practices

The technical practices are concerned with "solving the problem right" by creating and maintaining the code of your software system. They are the bit-head practices that your team will use to build and evolve the software system.

- Chapter 26, "Automated Developer Tests"
- Chapter 27, "Test-Last Development"
- Chapter 28, "Test-First Development"
- Chapter 29, "Refactoring"
- Chapter 30, "Continuous Integration"
- Chapter 31, "Simple Design"
- Chapter 32, "Functional Tests"
- Chapter 33, "Collective Code Ownership"
- Chapter 34, "Pair Programming"

Part 3.3 Supporting Practices

These are not Agile practices per se, but they are practices that you can use to support your team's adoption and introduce change into your organization.

- Chapter 35, "Coach"
- Chapter 36, "Engage the Community"
- Chapter 37, "Reading Circle"
- Chapter 38, "Workshop"
- Chapter 39, "Classroom Training"

Part 3.4 The Clusters

The clusters of practices are sets of Agile practice patterns that work especially well together. The first two clusters are focused on people, interactions, and teamwork. The practices that make up these clusters enable a team to recognize change as it happens and provide a working process for responding to those changes. The last three clusters are technical in nature and give the team the technical ability to respond to changes as they occur.

- Chapter 40, "Agile Iteration"
- Chapter 41, "Communication Cluster"
- Chapter 42, "Evolutionary Design"
- Chapter 43, "Test-Driven Development"
- Chapter 44, "Test-Driven Requirements"

Part 4: Case Studies

These are reports of two different adoption efforts. They get beyond the theory and show how two organizations are working through their Agile adoptions. One has been very successful, and the other is still struggling. Both are real companies, with real people and real politics. It gets messy. But, in the end, so will your adoption effort before you succeed.

- Chapter 45, "BabyCenter"
- Chapter 46, "Company X"

Appendices

The appendices contain material that does not quite fit in the main flow of the book but that could be useful to you.

- Appendix A, "Pattern to Business Value Mappings"
- Appendix B, "Pattern to Smell Mappings"
- Appendix C, "Getting the Most from Agile Practice Patterns"
- Appendix D, "Further Reading"

How to Read this Book

Okay. So enough about what you are going to do. How do you do it? The first thing you have to do is come up with a set of Agile development practices for you and your team. You can do that by reading Part 1 and taking the time to do the exercises at the end of Chapters 3, 4, and 5. It is important that you spend the time to work through the exercises. After completing these chapters, you will have a list of prioritized practices to consider.

At that point, you can start with the third part of the book—the patterns. You will use the list of practices on your list to "dig deep" by reading each pattern and deciding if it is *really* applicable to your environment. When you find a practice that matches, you and your team will start adopting it incrementally using the guidance in that pattern. You'll also watch out for symptoms of that practice going bad by using the guidance in the "smells" documented in each pattern.

Finally, you'll continuously evaluate the effectiveness of the practices you've adopted and adapt them to get greater value from them for your business. Start right now by turning to the next chapter.

Chapter 5

Adopting Agile Practices

So far you have read about business value and smells. You have also done the exercises at the end of each chapter and come up with a prioritized list of business values and a prioritized list of smells that need fixing. If you have not done so yet, please stop now and go back and do so. Armed with an understanding of your customers' priorities and the main pains your company is experiencing, you are ready to determine what practices you should consider adopting to alleviate those pains and get the most value for your efforts.

In this chapter, I will give you direction on how to go about successfully choosing which practices to consider adopting. I'll also ask you to benchmark your work—even if you just do it subjectively—so you can be "agile" about your adoption. This is, however, only advice on how to come up with your own priorities and your own list of practices to adopt. If you are looking for a prescription—do practice A, then B, but not C—you won't find it here. (And if you do find it elsewhere, my advice to you is not to trust it.)

The Practices

The bulk of this book contains patterns of Agile practice adoption—that is, Agile practices written up in pattern format with a focus on adoption. In this chapter, your goal is to choose the practices that fit your organization's context. That comes down to relying on the work you've done in the previous chapter in prioritizing your business values and smells and using it to choose which practices to adopt to improve your business values and reduce your smells.

PATTERNS OF AGILE PRACTICE TO BUSINESS VALUE MAPPINGS

Let's start with the meat of the chapter. Figures 5-1 through 5-7 provide diagrams for each business value and the practices that improve that business value. These mappings, like all the patterns in this book, are built by aggregating experiences from several Agile adoption efforts. Each of these practices corresponds to a pattern that is documented later in this book. Don't worry if you don't know exactly what some of these practices are at this point in time.

Lets examine Figure 5-1 to understand how to read these business value charts. Arrows between practices indicate dependencies; therefore, Refactoring depends on Automated Developer Tests. Also vertical ordering is important; the higher up a practice is, the more effective it is for the business value. Therefore, Iterations are more effective than Automated Developer Tests, and Test-First Development is more effective than Test-Last Development with respect to decreasing the time to market. Use these diagrams to determine what practices to consider adopting. Take the suggestions accompanying each diagram as just that—suggestions. All the practices in each diagram positively affect that business value, and you may discover upon reading the details of the suggested practices that they do not apply in your particular context.

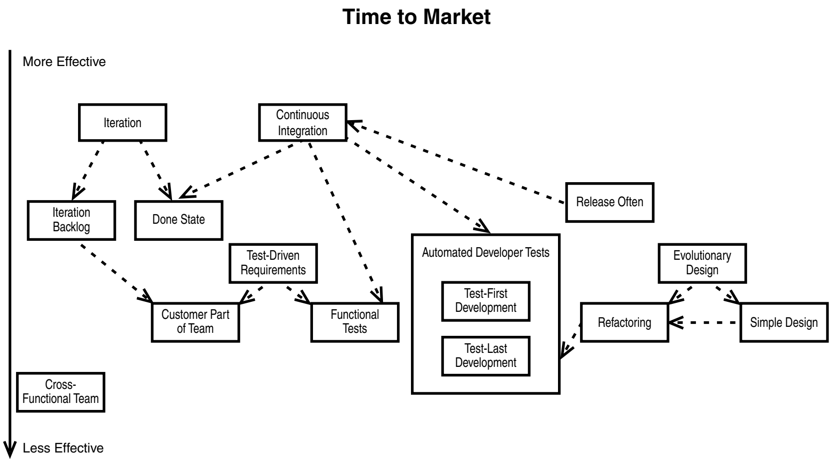

Figure 5–1 Time to Market practices

Small steps and failing fast are the most effective methods to get things out quickly. Weed out defects early because the earlier you find them, the less they will cost, and you won't be building on a crumbling foundation. That is why Iteration and Continuous Integration lead the practices that most positively affect time to market. They are both, however, dependent on other practices. Consider starting with automated tests and the Iteration trio—Iteration, Done State, and Iteration Backlog—when you want to improve time to market.

Figure 5-2 gives the practices that increase product utility. By far, the most effective practice is Customers Part of Team. Go there first. Then consider functional tests if you are already doing automated developer tests or an iteration ending with a demo.

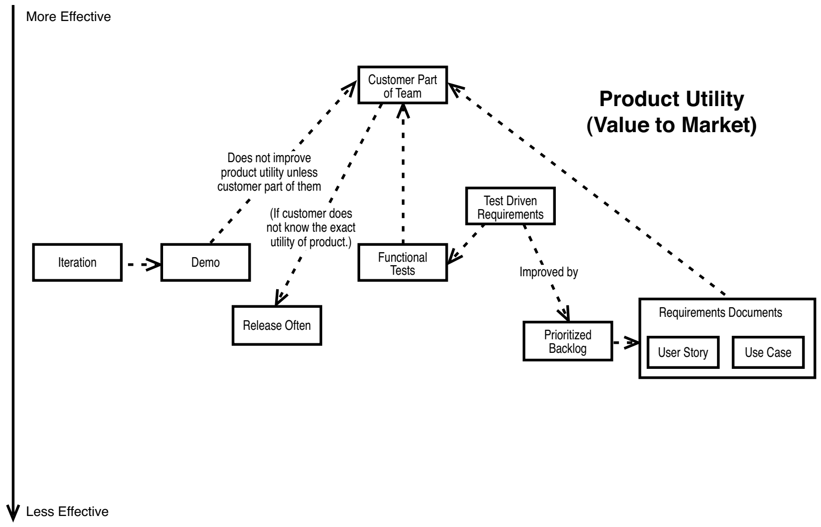

Figure 5–2 Product Utility practices

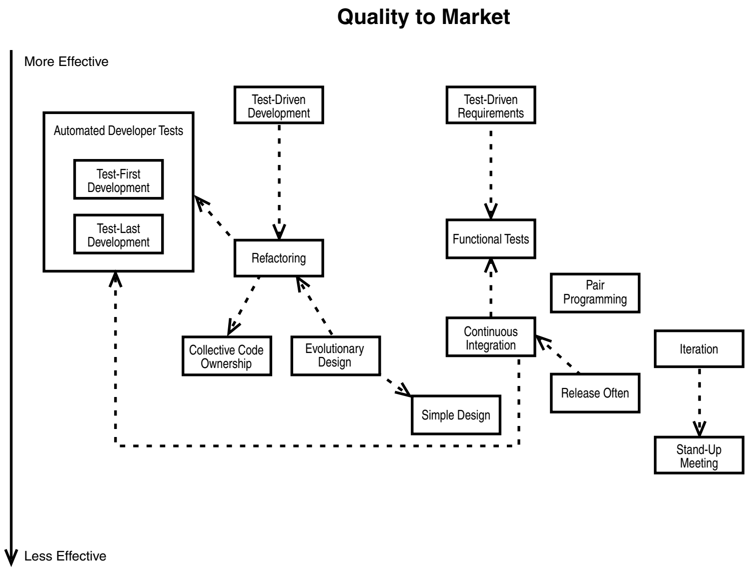

Figure 5–3 Quality to Market practices

Although quality to market test-driven development and test-driven requirements are king, of course, they both depend on other practices. So consider starting with one of the Automated Developer tests (preferably test-first development) and Pair Programming, closely followed by Refactoring. Pair programming helps you come up to speed with these particularly difficult practices. Once you are comfortable with automated developer tests, aim for full-fledged test-driven development and consider functional tests.

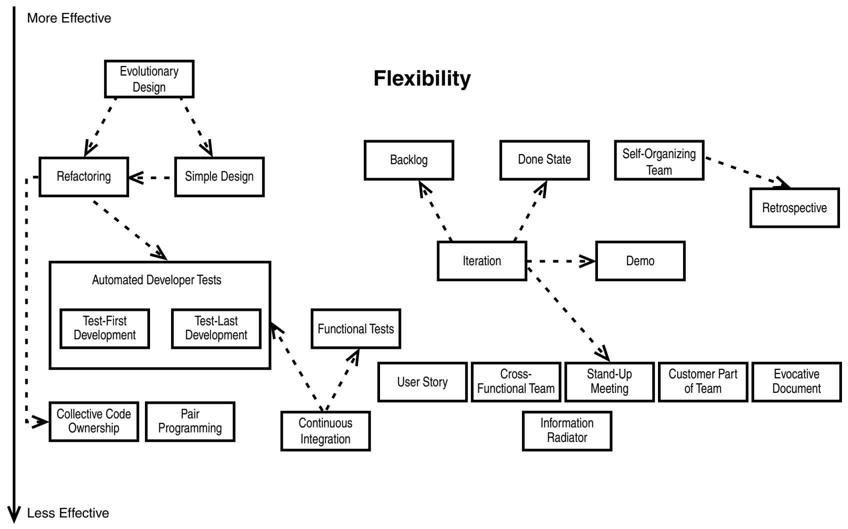

Figure 5–4 Flexibility practices

There are two general types of flexibility in software development: team flexibility and technical flexibility. Team flexibility is the team's ability to recognize and respond to changes that happen. For a team to respond to changes by changing the software, there needs to be technical flexibility. Therefore, you need both team flexibility and technical flexibility. Start with Automated Developer Tests, a self-organizing team, and the trio of Iteration, Done State, and Backlog. The testing gets you on your way to technical flexibility, and the remaining practices enable your team's flexibility.

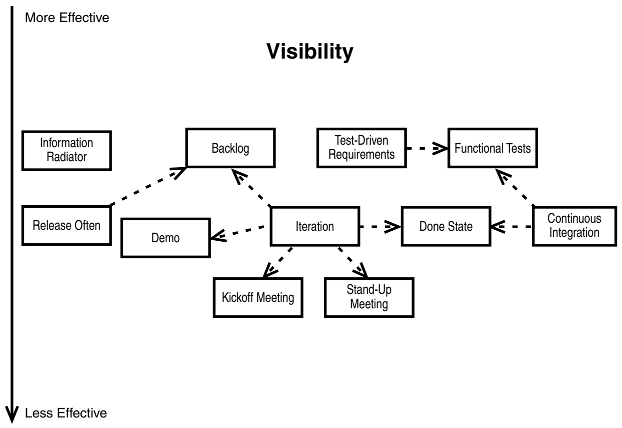

Figure 5–5 Visibility practices

The backlog and information radiators are your first easy steps toward increased visibility. Depending on your need for increasing visibility, you can take an easy route and consider iterations with a done state and a demo or a more difficult but effective route with functional tests and test-driven requirements.

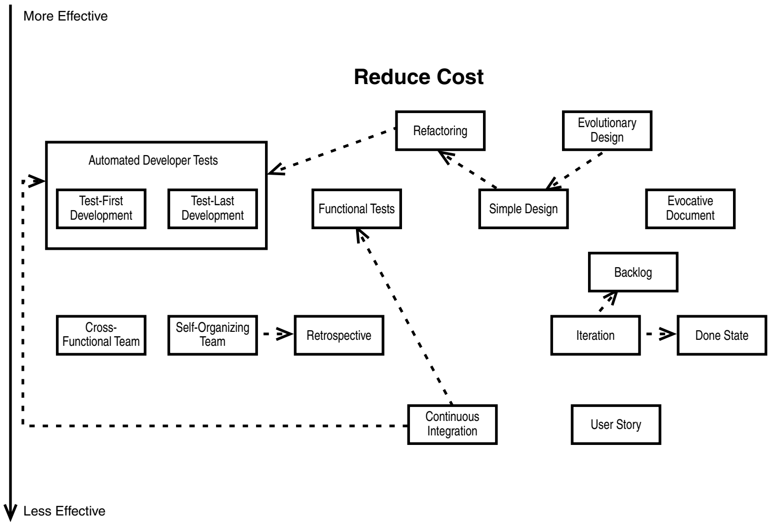

Figure 5–6 Reduce Cost practices

You can reduce cost in two ways: make the code easier to maintain and write less code—that is, code for the most important features first. Automated tests, followed by refactoring, simple design, and evolutionary design, are your path toward reducing the cost of maintenance. A backlog, iteration, and done state reduce the amount of code written.

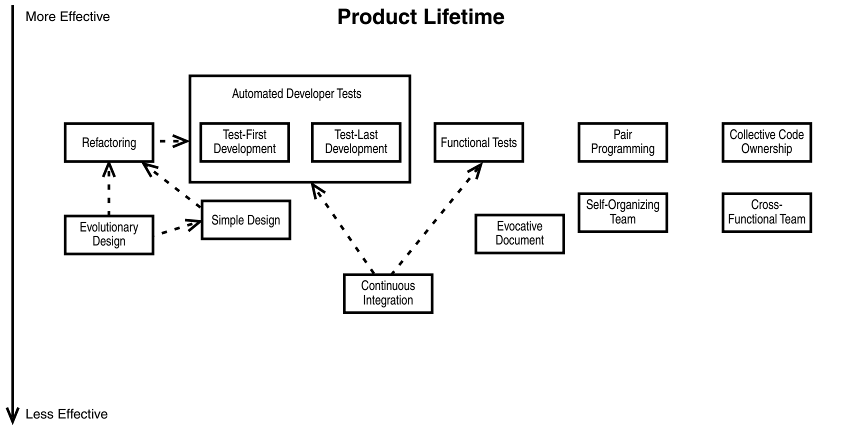

Figure 5–7 Product Lifetime practices

Product lifetime is inversely proportional to the cost of software maintenance. There are two ways that we know how to reduce maintenance costs: 1) build a safety net of tests that allow changes to the software system and reduce the cost of change and 2) spread the knowledge of the design of the software system. Automated developer tests are your key to (1), while pair programming and collective code ownership are good starts for (2).

Patterns of Agile Practice to Smell Mappings

There are two types of smells: business smells and process smells. The business smells are inverses of business values, and their patterns of Agile practice mappings are identical:

- Quality Delivered to Customer Is Unacceptable: Quality to Market
- Delivering New Features to Customer Takes Too Long: Time to Market
- Features Are Not Used by Customer: Product Utility
- Software Is Not Useful to Customer: Product Utility
- Software Is Too Expensive: Reduce Cost

The remaining smells have their own mappings to patterns of Agile practices (see Figures 5-8 through 5-15).

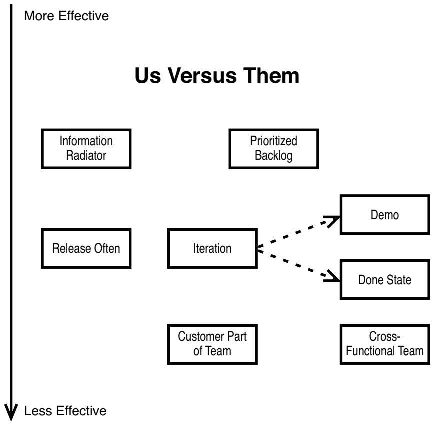

Figure 5–8 Us Versus Them practices

The Us Versus Them smell can best be alleviated by having frequent conversations about the true nature of the project. Start with increasing visibility by creating information radiators that show the key points in your development. Create a prioritized backlog by involving the whole development team—including the customers. Use these practices to increase visibility and build trust. When you are ready, take it further and build more trust by delivering often by adopting the iteration, demo, and done state trio.

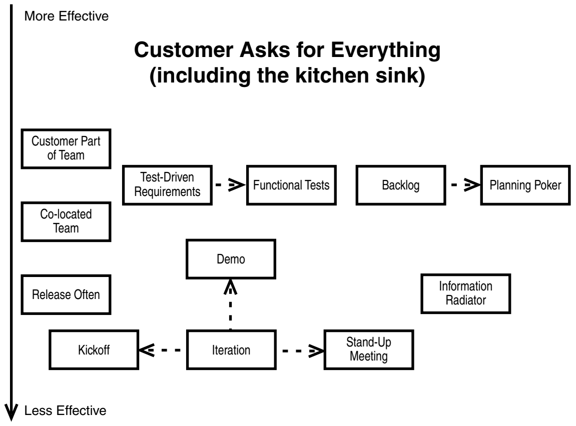

Figure 5–9 Customer Asks for Everything practices

Understand that when the customer asks for everything, it is a symptom of lack of trust from the customer that the features will be delivered promptly and a legacy of change-management barriers in traditional development. The best way to address this issue is to bring the customers in as part of the development team. Have them build the Backlog with the teams input and be responsible for its prioritization.

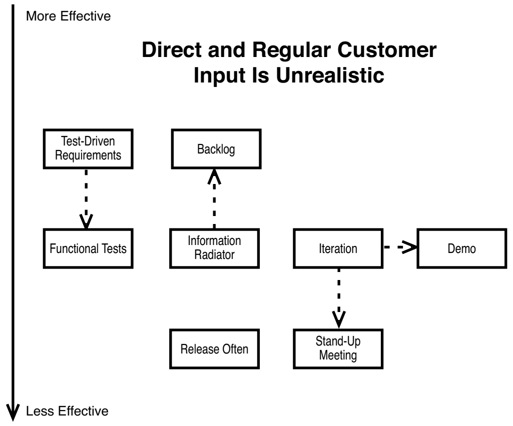

Figure 5–10 Direct and Regular Customer Input Is Unrealistic practices

If direct input from the customer is not possible, mitigate this problem by reducing the number of communication errors. You can do this by building functional tests and slowly working toward test-driven requirements, where the customer's requirements document becomes an executable specification. This particular practice will take a long time to adopt correctly, so start early and be patient. In the meantime, create a backlog and start delivering work incrementally, with iterations ending with a demo.

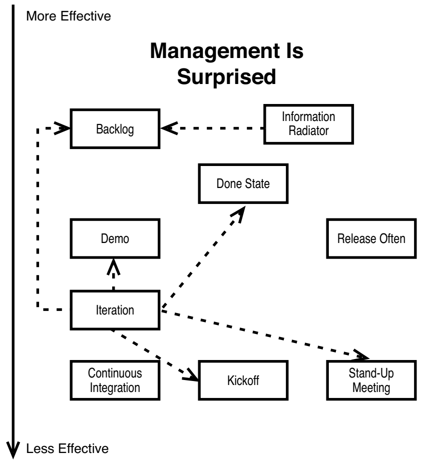

Figure 5–11 Management Is Surprised—Lack of Visibility practices

To keep management from being surprised, you need to do two things: 1) build your application incrementally from end to end and 2) communicate your true progress. Address 1) by defining a done state that is as close to deployment as possible and then working in iterations. Communicate your true progress by working through information radiators showing your true progress and using a demo at the end of every iteration.

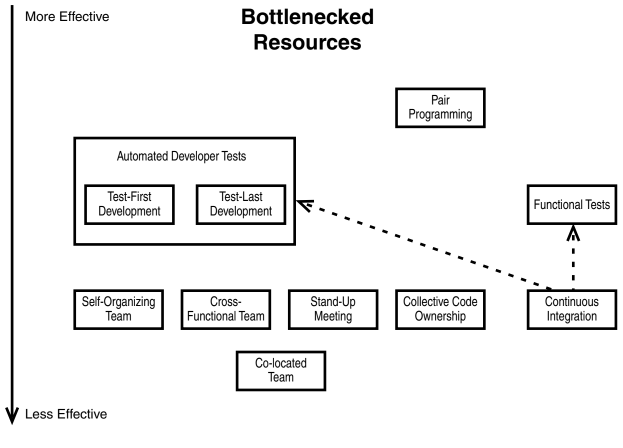

Figure 5–12 Bottlenecked Resources—*Software Practitioners Are Members of Multiple Teams Concurrently* practices

Bottlenecked resources happen because of specialization. Pair programming is your single most effective method to share knowledge and spread the specialization. This, in turn, allows more than one person to address issues that were previously the domain of a single expert. Automated developer tests help this in another way—they allow people to work in code they do not know well and rely on a safety net of tests to tell them if they have broken any previously working code. This should be your second step toward alleviating resource bottlenecks.

Projects churn when there is no clear prioritization. Prioritize requirements by creating and maintaining a backlog. To make sure that the backlog is an accurate reflection of customer needs, make your customers part of your team and put together a cross-functional team that can build those requirements end to end. This will give you and your customers a better understanding of requirements, their priorities, and a feedback loop to make course corrections quickly.

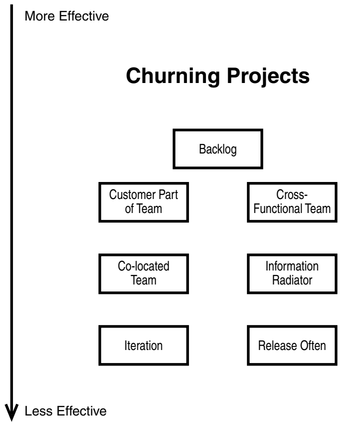

Figure 5–13 Churning Projects practices

Hundreds of bugs need to be reduced. Start with automated developer tests supported by pair programming to reduce the number of bugs you are introducing and start building a safety net of tests. Then work with iterations with a done state to find as many bugs as possible. Don't put off painful issues such as integration. Fix things early.

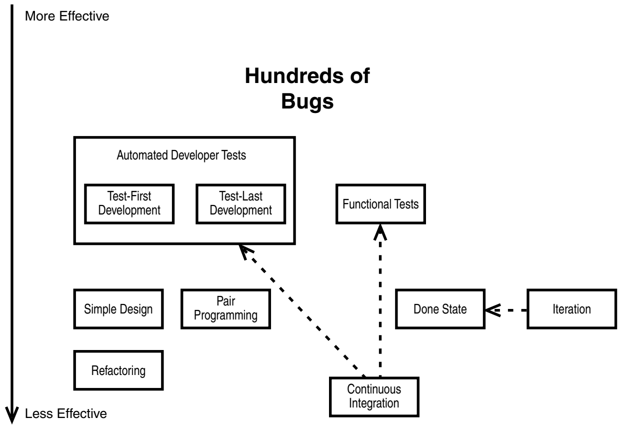

Figure 5–14 Hundreds (Possibly Thousands) of Bugs *in Bug Tracker* practices

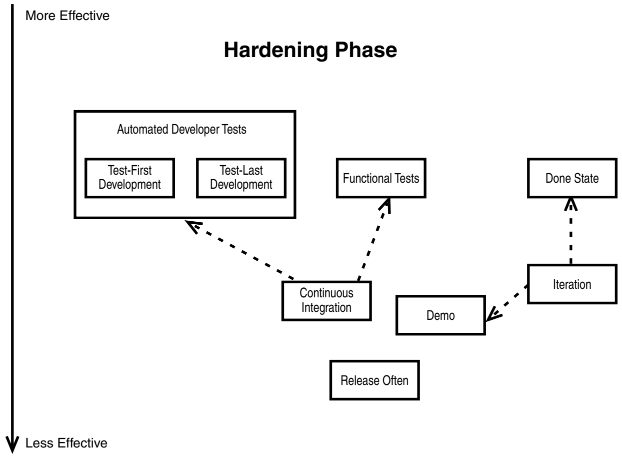

Figure 5–15 Hardening Phase *Needed at End of Release Cycle* practices

If you have a hardening phase, you've let a significant number of defects accumulate. Stop doing what you've been doing and add tests as you develop code via automated developer tests. Choose a good done state—one that takes you as close to deployment in every iteration as possible—to weed out those difficult-to-find bugs early.

CRAFTING YOUR AGILE ADOPTION STRATEGY

You can use the information you have gathered so far about business value and smells to determine which practices you should consider adopting.

- **Choose practices based solely on business value delivered.** In this scenario, you are not suffering from any severe pains. You just want to improve your software development process by increasing the business value that your team delivers. Use the Business Value to Practice mapping in Figures 5-1 through 5-7 to choose practices that most strongly affect your organization's business values.
- **Choose practices to alleviate smells that have been prioritized by business value.** This technique focuses on alleviating pains that you have while keeping business value in mind. Smells are prioritized according to your customers business values. Then, from the prioritized smell list, you choose the appropriate practices to adopt with the help of the Smell to Practice mappings shown in Figures 5-8 through 5-15.
- **Choose practices to address the most visible smells.** This is common, although I wouldn't recommend it. It's plain and simple "firefighting"—trying to get rid of the biggest pain regardless of the business value it delivers. This is all too common when the technical team determines the priority without customer input. (I've often been guilty of this.)

The information found in the figures at the beginning of this chapter is prioritized by effectiveness. Therefore, the first practice in the figure is the most effective practice for increasing the business value or alleviating the smell. Get your feet wet with the first practice, and after you've successfully adopted that, come back and take another look at the remaining practices and clusters related to your business value or smell.

No matter how you prioritize your list of practices to adopt, you should adopt those practices as iteratively as possible. Armed with the list of practices, here is how you can successfully adopt the Agile practices on your list.

1. Start with an evaluation of the status quo. Take readings (even if they are subjective) of the current business value(s) you want to improve and the smell(s) you want to alleviate.
2. Set goals that you want to reach. How much do you want to increase the business value? How much do you want to reduce the smell? What is the time frame? Take a guess initially and modify it as you know more through experience.
3. Pull the first practice or cluster off the list you created.

4. Read the pattern that is related to that cluster or practice. Decide if it is applicable or not by matching the context and forces to your working environment (more details on what patterns are and their different sections in Part 3: The Pattern Catalog). If the practice is not applicable in your environment, go back and pick the next one off the business value/smells table.
5. Once you have determined that the pattern is applicable in your environment, read the pattern thoroughly. Follow the advice in the "Adoption" section in the pattern to get started.
6. Periodically evaluate whether the business value you are addressing is improving or that the smell you are addressing is being resolved. If it is not, adapt your practice for your environment using hints from the "Variations" and "But" sections in the pattern. (You might want to take a quick read of Chapter 6, "The Patterns of Agile Practice Adoption," at this point to get an understanding of what an Agile adoption pattern looks like.)
7. Go back to step 1 and re-evaluate your business value or smell. If it needs more improvement (that is, you still have not met your goal set in 2), consider adding another practice or an entire cluster to resolve the issue. If it has met your goals, move on to the next one.

So where is the test-driven part of this approach? Your tests are your goal values that you set in step 2. In step 6, you check your readings after adopting a practice. This is a test of how effectively the practice(s) you adopted has already met the goal set earlier. This loop—set a goal, adopt a practice, and then validate the practice against the expected goal—is a test-driven adoption strategy.[1]

WHERE NEXT?

With the completion of this chapter, you are ready to create your own Agile adoption strategy that focuses on the business values and smells of your organization. This is not a one-shot deal—remember to be agile about your adoption. Measure your progress against your goals and revisit them regularly. Modify your strategy as you learn more about each practice and your own environment. It is natural to make a few wrong turns, but fail fast and quickly recover.

1. In management practices, this is commonly referred to as the PDCA cycle (Plan, Do, Check, Act), originally developed by Walter Shewhart at Bell Laboratories in the 1930s and promoted effectively in 1950s by the quality management guru W. Edward Deming.

Index

Note: Page numbers followed by *n* refer to footnotes

A

adoption patterns
  Agile Iteration
    adoption, 260-261
    business value, 257
    common problems, 261
    context, 258
    description, 259-260
    forces, 258-259
    overview, 257
    references, 262
    sketch, 258
    variations, 261-262
  agile practice to business value mappings, 38-43, 323
  agile practice to smell mappings, 43-48, 325
  Automated Developer Tests
    adoption, 167-170
    business value, 164
    common problems, 170-171
    context, 165
    definition, 163
    description, 166-167
    forces, 165-166
    references, 172
    sketch, 164-165
    variations, 171-172
  Backlog
    adoption, 84
    business value, 81
    common problems, 85
    context, 82
    description, 83
    forces, 82-83
    iteration backlogs, 82
    overview, 81
    product backlogs, 82
    references, 86
    sketch, 81-82
    variations, 86
  character roles, 58-59
  Classroom Training
    adoption, 250-251
    business value, 249
    common problems, 252
    context, 250
    description, 250
    forces, 250
    overview, 249
    sketch, 249
    variations, 252-253
  Coach
    adoption, 232-233
    business value, 231
    common problems, 233
    context, 232
    definition, 231
    description, 232

forces, 232
references, 234
sketch, 231
variations, 234
Collective Code Ownership pattern
adoption, 221
business value, 219
common problems, 222
context, 220
description, 221
forces, 220
overview, 219
references, 222
sketch, 219
variations, 222
Co-Located Team
adoption, 121-122
business value, 119
common problems, 122
context, 120
description, 121
forces, 120
overview, 119
references, 123
sketch, 119-120
variations, 122
Communication cluster
adoption, 266
business value, 264
common problems, 267
context, 264
description, 265-266
forces, 265
overview, 263
references, 268
sketch, 264
variations, 267
Continuous Integration
adoption, 191-194
business, 189
common problems, 194-195
context, 190
definition, 189
description, 191
forces, 190
references, 196
sketch, 189-190
variations, 195-196
Cross-Functional Team
adoption, 134
business value, 131
common problems, 134-135
context, 132
description, 133
forces, 133
overview, 131
references, 135
sketch, 132
variations, 135
Customer Part of Team
adoption, 139-140
business value, 137
common problems, 140-141
context, 138
description, 139
forces, 138
overview, 137
references, 142
sketch, 137-138
variations, 142
Cycle
adoption, 67
business value, 65
common problems, 67
context, 66
description, 66
forces, 66
overview, 65
references, 68
sketch, 65
variations, 68
definition, 55
Demo
adoption, 105
business value, 103
common problems, 105-106

context, 104
definition, 103
description, 104-105
forces, 104
references, 107
sketch, 103
variations, 106
Done State
adoption, 101
business value, 99
common problems, 101-102
context, 100
definition, 99
description, 100
forces, 100
references, 102
sketch, 99-100
variations, 102
Engage the Community
adoption, 236-238
business values, 235
common problems, 238
context, 236
description, 236
forces, 236
overview, 235
references, 238
sketches, 235
variations, 238
Evocative Document
adoption, 145-146
business value, 143
common problems, 146
context, 144
definition, 143
description, 145
forces, 144-145
references, 147
sketch, 143-144
variations, 146-147
Evolutionary Design
adoption, 273-274
business value, 269
common problems, 274-275
context, 271
description, 272-273
forces, 271-272
overview, 269
references, 275
sketch, 269-270
variations, 275
format, 55-57
Functional Tests
adoption, 210-211
architecture smells, 214-215
benefits of, 208-210
business value, 203
context, 204
definition, 203
description, 206
forces, 204-205
implementation smells, 211-213
Item Inventory, 206-208
references, 217
sketch, 203-204
variations, 215-217
Goal, 61-63
references, 64
variations, 63-64
Information Radiator
adoption, 158-159
business value, 157
common problems, 159
context, 158
definition, 157
description, 158
forces, 158
references, 160
sketch, 157
variations, 160
Iteration
adoption, 74
business value, 71
common problems, 75-76
context, 72
description, 73

forces, 72-73
overview, 71
references, 76
sketch, 72
variations, 76
Kickoff Meeting
adoption, 79-80
business value, 77
context, 78
description, 78
forces, 78
overview, 77
references, 80
sketch, 77-78
variations, 80
overview, 37
Pair Programming
adoption, 225-226
business value, 223
common problems, 226
context, 224
definition, 223
description, 224-225
forces, 224
references, 227
sketch, 223
variations, 226-227
Planning Poker
adoption, 89-90
business value, 87
common problems, 90-91
context, 88
definition, 87
description, 88-89
forces, 88
references, 91
sketch, 87-88
Reading Circle
adoption, 241-242
business value, 239
common problems, 242
context, 240
definition, 239
description, 240-241
forces, 240
references, 243
sketch, 239-240
variations, 242-243
reasons for adopting agile practices, 13
Refactoring
adoption, 185
business value, 183
common problems, 186
context, 184
definition, 183
description, 184-185
forces, 184
references, 187
sketch, 183-184
variations, 186-187
Release Often
adoption, 117
business value, 115
common problems, 118
context, 116
description, 117
forces, 117
overview, 115
references, 118
sketch, 116
variations, 118
Retrospective
adoption, 112
business value, 109
common problems, 112-113
context, 110
definition, 109
description, 111-112
forces, 110-111
references, 113
sketch, 109-110
variations, 113
Self-Organizing Team
adoption, 127-128
business value, 125

common problems, 128
context, 126
description, 127
forces, 126-127
overview, 125
references, 129
sketch, 125-126
variations, 129
Simple Design
adoption, 200
business value, 197
common problems, 200-201
context, 198
definition, 197
description, 199
forces, 198
references, 201
sketch, 197-198
variations, 201
Stand-Up Meeting
adoption, 95-96
business value, 93
common problems, 96
context, 93
definition, 93
description, 94-95
forces, 94
references, 98
sketch, 93
variations, 97
successful adoptions, 14
Test-Driven Development
adoption, 280-281
business value, 277
common problems, 281-282
context, 278
description, 279-280
forces, 278-279
overview, 277
references, 282-283
sketch, 277-278
variations, 282
Test-Driven Requirements
adoption, 288
business value, 285
common problems, 289-290
context, 286-287
description, 287-288
forces, 287
overview, 285
references, 290-291
sketch, 286
variations, 290
Test-First Development
adoption, 180-181
business value, 177
common problems, 181
context, 178
definition, 177
description, 179-180
forces, 178-179
references, 182
sketch, 177-178
variations, 181
Test-Last Development
adoption, 175
business value, 173
common problems, 175-176
context, 174
definition, 173
description, 175
forces, 174
references, 176
sketch, 173-174
tips for use, 57-58
unsuccessful adoptions, 14
Use Case
adoption, 155
business value, 153
common problems, 155-156
context, 154
definition, 153
description, 154
forces, 154

references, 156
sketch, 153
variations, 156
User Story
adoption, 151
business value, 149
common problems, 151-152
context, 150
definition, 149
description, 150-151
forces, 150
references, 152
sketch, 149-150
variations, 152
Workshop
adoption, 247
business value, 245
common problems, 247
context, 246
description, 246-247
force, 246
overview, 245
references, 248
sketch, 245-246
variations, 247
adoption strategies
building, 51
test-driven adoption strategies, 49-50
agile adoption patterns. *See* adoption patterns
Agile Alliance, 237
Agile conferences, 237
Agile Estimating and Planning, 89
Agile iterations
adopting, 260-261
business value, 257
common problems, 261
context, 258
description, 259-260
forces, 258-259
overview, 257
references, 262
sample adoption scenario, 258
variations, 261-262
Agile Journal, 238
Agile Project Leadership Network (APLN), 237
agile software development
agile adoptions. *See* adoption patterns
business values. *See* business
daily cycles, 6
demos, 6
development of, 5
iterations, 6
learning
communication, 7-9
impact on time spent on projects, 3-4
importance of, 3
learning bottlenecks, 9-11
recognizing and responding to change, 5-7
management tests, 7
personal agility, 18-19
releases, 7
Responsibility Process model
adopting, 16
illustration, 15
individual responsibility, 18
overview, 15-17
and response to change, 18
responsibility of agile team members, 17
retrospectives, 6
scrum of scrums, 7
smells
business smells, 30-32
definition, 29
overview, 29
process smells, 32-35
success of, 5
TDR (test-driven requirements), 6
test-first development, 6
analysis paralysis, 85
APLN (Agile Project Leadership Network), 237
architecture smells (functional tests), 214-215
automated developer tests
adopting, 167-170

business value, 164
common problems, 170-171
context, 165
definition, 163
description, 166-167
forces, 165-166
references, 172
sample adoption scenario, 164-165
variations, 171-172

B

BabyCenter case study
overview, 295-296
Q1 2007, 296-301
applying software development practices, 300-301
determining business value, 296-297
selecting software development practices, 297-298
Q1 2008, 302-303
results, 302-303
backlogs
adoption, 84
business value, 81
common problems, 85
context, 82
description, 83
forces, 82-83
iteration backlogs, 82
overview, 81
product backlogs, 82
references, 86
sample adoption scenario, 81-82
variations, 86
Basili, Victor, xiv
Better Software, 238
bibliography, 333-337
bottlenecks
bottlenecked resources, 33-34, 47, 325
learning, 9-11
bugs, 34, 47, 325
build monkeys, 160
building adoption strategies, 51
burn-down charts, 160
business smells
customer asks for everything, 32, 45
definition, 30
delivering new features to customer takes, 30
features are not used by customer, 30-31
quality delivered to customer is, 30
software is to expensive to build, 31
software not useful to customer, 31
us versus them, 31-32, 44
business values
agile practice to business value mappings, 38-43, 323
determining, 26-27, 296-297
driving processes/technologies with, 297-298
increased flexibility, 24-25
increased product lifetime, 26
increased product utility, 24
increased quality to market, 24
increased visibility, 25
mapping to patterns, 323
as organizational goals, 26
overview, 23
reduced costs, 25
reduced time to market, 23

C

cargo cults, 12
case studies
BabyCenter
applying software development practices, 300-301
determining business value, 296-297
overview, 295-296
Q1 2007, 296-301
Q1 2008, 302-303
results, 302-303
selecting software development practices, 297-298
Company X
business goals, 306
context, 306
current problems, 311-312

    Digital Output team, 309-310
    Imaging team, 310
    OS team, 308-309
    overview, 305
    QA team, 306-307
    Remote Output team, 308
    results, 316-319
    Scheduler team, 310
    suggested practices, 312-316
    Systems team, 307
    what currently works, 311
change, recognizing and responding to, 5-7
character roles, 58-59
chickens, 96
churning projects, 34, 47, 324-325
Classroom Training pattern
  adoption, 250-251
  business value, 249
  common problems, 252
  context, 250
  description, 250
  forces, 250
  overview, 249
  sketch, 249
  variations, 252-253
clusters
  Agile Iteration
    adoption, 260-261
    business value, 257
    common problems, 261
    context, 258
    description, 259-260
    forces, 258-259
    overview, 257
    references, 262
    sketch, 258
    variations, 261-262
  Communication
    adopting, 266
    business value, 264
    common problems, 267
    context, 264
    description, 265-266
    forces, 265
    overview, 263
    references, 268
    sample adoption scenario, 264
    variations, 267
  Evolutionary Design
    adoption, 273-274
    business value, 269
    common problems, 274-275
    context, 271
    description, 272-273
    forces, 271-272
    overview, 269
    references, 275
    sketch, 269-270
    variations, 275
  mapping to business value, 323
  mapping to smells, 325
  Test-Driven Development
    adoption, 280-281
    business value, 277
    common problems, 281-282
    context, 278
    description, 279-280
    forces, 278-279
    overview, 277
    references, 282-283
    sketch, 277-278
    variations, 282
  Test-Driven Requirements
    adoption, 288
    business value, 285
    common problems, 289-290
    context, 286-287
    description, 287-288
    forces, 287
    overview, 285
    references, 290-291
    sketch, 286
    variations, 290
coaches
  adoption, 232-233
  business value, 231
  common problems, 233
  context, 232

definition, 231
description, 232
forces, 232
references, 234
sample adoption scenario, 231
variations, 234
Cohn, Mike, 89
collective code ownership
adopting, 221
business value, 219
common problems, 222
context, 220
description, 221
forces, 220
overview, 219
references, 222
sample adoption scenario, 219
variations, 222
co-located teams
adoption, 121-122
business value, 119
common problems, 122
context, 120
definition, 8
description, 121
forces, 120
overview, 119
references, 123
sketch, 119-120
variations, 122
communication, 7-9
Communication cluster pattern
adoption, 266
business value, 264
common problems, 267
context, 264
description, 265-266
forces, 265
overview, 263
references, 268
sketch, 264
variations, 267
community, engaging
adoption, 236-238
business values, 235
common problems, 238
context, 236
description, 236
forces, 236
overview, 235
references, 238
sketches, 235
variations, 238
Company X case study
business goals, 306
context, 306
current problems, 311-312
Digital Output team, 309-310
Imaging team, 310
OS team, 308-309
overview, 305
QA team, 306-307
Remote Output team, 308
results, 316-319
Scheduler team, 310
suggested practices
Go Faster practices, 312-313
Go Slower practices, 313-314
Increase Quality practices, 314
long term recommendations, 316
proofs and concerns, 315
recommendations for moving forward, 314-315
Systems team, 307
what currently works, 311
conferences, 237
constraints, 331
continuous integration
adopting, 191-194
business definition, 189
common problems, 194-195
context, 190
definition, 189
description, 191
forces, 190
references, 196
sample adoption scenario, 189-190
status, 160
variations, 195-196

Core System teams, 309-310
costs
  expense to build software, 31
  reducing, 25, 42, 324
cross-functional teams
  adopting, 134
  business value, 131
  common problems, 134-135
  context, 132
  description, 133
  forces, 133
  overview, 131
  references, 135
  sample adoption scenario, 132
  variations, 135
Customer Part of Team pattern
  adoption, 139-140
  business value, 137
  common problems, 140-141
  context, 138
  description, 139
  forces, 138
  overview, 137
  references, 142
  sketch, 137-138
  variations, 142
customers
  including on teams
    adoption, 139-140
    business value, 137
    common problems, 140-141
    context, 138
    description, 139
    forces, 138
    overview, 137
    references, 142
    sample adoption scenario, 137-138
    variations, 142
  input, 32-33
cycles
  adopting, 67
  business value, 65
  common problems, 67
  context, 66
  daily cycles, 6
  definition, 65
  description, 66
  forces, 66
  references, 68
  sample adoption scenario, 65
  variations, 68

D

daily cycles, 6
delivering new features, 30
demos
  adopting, 105
  business value, 103
  common problems, 105-106
  context, 104
  definition, 103
  description, 104-105
  forces, 104
  overview, 6
  references, 107
  sample adoption scenario, 103
  variations, 106
design
  evolutionary design
    adopting, 273-274
    business value, 269
    common problems, 274-275
    context, 271
    description, 272-273
    forces, 271-272
    overview, 269
    references, 275
    sample adoption scenario, 269-271
    variations, 275
  simple design
    adopting, 200
    business value, 197
    common problems, 200-201
    context, 198
    definition, 197
    description, 199
    forces, 198
    references, 201
    sample adoption scenario, 197-198
    variations, 201

Design Patterns: Elements of Reusable Object-Oriented Software, 187
determining business values, 26-27
Digital Output teams, 309-310
documents
  evocative documents
    adoption, 145-146
    business value, 143
    common problems, 146
    context, 144
    definition, 143
    description, 145
    forces, 144-145
    references, 147
    sample adoption scenario, 143-144
    variations, 146-147
  information radiators
    adopting, 158-159
    build monkeys, 160
    burn-down charts, 160
    business value, 157
    common problems, 159
    context, 158
    continuous integration, 160
    definition, 157
    description, 158
    forces, 158
    happy face/sad face, 160
    impediment charts, 160
    references, 160
    sample adoption scenario, 157
    storyboards, 160
    variations, 160
  use cases
    adopting, 155
    business value, 153
    common problems, 155-156
    context, 154
    definition, 153
    description, 154
    forces, 154
    references, 156
    sample adoption scenario, 153
    variations, 156
  user stories
    adopting, 151
    business value, 149
    common problems, 151-152
    context, 150
    definition, 149
    description, 150-151
    forces, 150
    references, 152
    sample adoption scenario, 149-150
    variations, 152
Domain-Driven Design, 211
done states
  adoption, 101
  business value, 99
  common problems, 101-102
  context, 100
  definition, 99
  description, 100
  forces, 100
  references, 102
  sketch, 99-100
  variations, 102

E

engaging community (Engage the Community pattern)
  adoption, 236-238
  business values, 235
  common problems, 238
  context, 236
  description, 236
  forces, 236
  overview, 235
  references, 238
  sketches, 235
  variations, 238
estimation
  estimation paralysis, 85
  Planning Poker pattern
    adoption, 89-90
    business value, 87
    common problems, 90-91
    context, 88

definition, 87
description, 88-89
forces, 88
references, 91
sketch, 87-88
Evan, Eric, 211
evocative documents
adoption, 145-146
business value, 143
common problems, 146
context, 144
definition, 8, 143
description, 145
forces, 144-145
references, 147
sample adoption scenario, 143-144
variations, 146-147
evolutionary design
adopting, 273-274
business value, 269
common problems, 274-275
context, 271
description, 272-273
forces, 271-272
overview, 269
references, 275
sample adoption scenario, 269-270
variations, 275

F
Feathers, Michael, 171
features
delivering, 30
features not used by customer, 30-31
feedback practices
Backlog pattern
adoption, 84-85
business value, 81
common problems, 85
context, 82
description, 83
forces, 82-83
iteration backlogs, 82
overview, 81
product backlogs, 82
references, 86
sketch, 81-82
variations, 86
Co-Located Team pattern
adoption, 121-122
business value, 119
common problems, 122
context, 120
description, 121
forces, 120
overview, 119
references, 123
sketch, 119-120
variations, 122
Cross-Functional Team pattern
adoption, 134
business value, 131
common problems, 134-135
context, 132
description, 133
forces, 133
overview, 131
references, 135
sketch, 132
variations, 135
Customer Part of Team pattern
adoption, 139-140
business value, 137
common problems, 140-141
context, 138
description, 139
forces, 138
overview, 137
references, 142
sketch, 137-138
variations, 142
Demo pattern
adoption, 105
business value, 103
common problems, 105-106
context, 104

definition, 103
description, 104-105
forces, 104
references, 107
sketch, 103
variations, 106
Done State pattern
adoption, 101
business value, 99
common problems, 101-102
context, 100
definition, 99
description, 100
forces, 100
references, 102
sketch, 99-100
variations, 102
Evocative Document pattern
adoption, 145-146
business value, 143
common problems, 146
context, 144
definition, 143
description, 145
forces, 144-145
references, 147
sketch, 143-144
variations, 146-147
Information Radiator pattern
adoption, 158-159
business value, 157
common problems, 159
context, 158
definition, 157
description, 158
forces, 158
references, 160
sketch, 157
variations, 160
Iteration pattern
adoption, 74
business value, 71
common problems, 75-76
context, 72
description, 73
forces, 72-73
overview, 71
references, 76
sketch, 72
variations, 76
Kickoff Meeting pattern
adoption, 79-80
business value, 77
context, 78
description, 78
forces, 78
overview, 77
references, 80
sketch, 77-78
variations, 80
Planning Poker pattern
adoption, 89-90
business value, 87
common problems, 90-91
context, 88
definition, 87
description, 88-89
forces, 88
references, 91
sketch, 87-88
Release Often pattern
adoption, 117
business value, 115
common problems, 118
context, 116
description, 117
forces, 117
overview, 115
references, 118
sketch, 116
variations, 118
Retrospective pattern
adoption, 112
business value, 109
common problems, 112-113
context, 110

definition, 109
description, 111-112
forces, 110-111
references, 113
sketch, 109-110
variations, 113
Self-Organizing Team pattern
adoption, 127-128
business value, 125
common problems, 128
context, 126
description, 127
forces, 126-127
overview, 125
references, 129
sketch, 125-126
variations, 129
Stand-Up Meeting pattern
adoption, 95-96
business value, 93
common problems, 96
context, 93
definition, 93
description, 94-95
forces, 94
references, 98
sketch, 93
variations, 97
Use Case pattern
adoption, 155
business value, 153
common problems, 155-156
context, 154
definition, 153
description, 154
forces, 154
references, 156
sketch, 153
variations, 156
User Story pattern
adoption, 151
business value, 149
common problems, 151-152
context, 150
definition, 149
description, 150-151
forces, 150
references, 152
sketch, 149-150
variations, 152
finding smells, 36
FIT, 206
Fit for Developing Software, 210
FITNesse, 206
flexibility
increasing, 24-25
practices, 41
Fowler, Martin, 169
functional tests
adopting, 210-211
benefits of, 208-210
business value, 203
common problems
architecture smells, 214-215
implementation smells, 211-213
references, 217
variations, 215-217
context, 204
definition, 203
description, 206
forces, 204-205
Item Inventory Management Test example, 206-208
sample adoption scenario, 203-204

G

Gamma, Erich, 187
"The Goal, Question, Metric Approach" (paper), xiv
goals
adopting, 63
business values, 26, 61
common problems, 63
context, 62

description, 62
forces, 62
references, 64
S.M.A.R.T. goals, 62
sample adoption scenario, 61
variations, 63-64
grok, 97

H

happy face/sad face radiators, 160
hardening phase, 34-35, 48, 325
Highsmith, Jim, 5

I

Imaging teams, 310
impediment charts, 160
implementation smells (functional tests), 211-214
incentives, pain as, 35
increasing
flexibility, 24-25
product lifetime, 26
product utility, 24
quality to market, 24, 323
visibility, 25
individual responsibility, 18
InfoQ, 238
information radiators
adopting, 158-159
business value, 157
common problems, 159
context, 158
definition, 8, 157
description, 158
forces, 158
references, 160
sample adoption scenario, 157
variations, 160
input from customers, 32-33
integration, continuous, 35
adopting, 191-194
business definition, 189
common problems, 194-195
context, 190
definition, 189
description, 191
forces, 190
references, 196
sample adoption scenario, 189-190
variations, 195-196
Item Inventory Management Test example, 206-208
iteration backlogs, 82
Iteration pattern
adoption, 74
business value, 71
common problems, 75-76
context, 72
description, 73
forces, 72-73
overview, 71
references, 76
sketch, 72
variations, 76
iterations
Agile Iteration pattern
adopting, 260-261
business value, 257
common problems, 261
context, 258
description, 259-260
forces, 258-259
overview, 257
references, 262
sample adoption scenario, 258
variations, 261-262
Iteration pattern
adoption, 74
business value, 71
common problems, 75-76
context, 72
description, 73
forces, 72-73
overview, 71
references, 76
sketch, 72
variations, 76

K

Kerievsky, Joshua, 187
kickoff meetings
  adopting, 79-80
  business value, 77
  context, 78
  description, 78
  forces, 78
  overview, 77
  references, 80
  sample adoption scenario, 77-78
  variations, 80

L

lean manufacturing, 332
learning
  communication, 7-9
  impact on time spent on projects, 3-4
  importance of, 3
  learning bottlenecks, 9-11
  recognizing and responding to change, 5-7
lifetime of products
  increasing, 26
  practices, 43

M

mailing lists, 237
management tests, 7
mappings
  patterns to business values, 323
  patterns to smells, 325
  agile practice to business values, 38-43, 323
  agile practice to smells, 43-49, 325
Martin, Robert, 5
meetings
  kickoff meetings
    adopting, 79-80
    business value, 77
    context, 78
    description, 78
    forces, 78
    overview, 77
    references, 80
    sample adoption scenario, 77-78
    variations, 80
  reading circles
    adopting, 241-242
    business value, 239
    common problems, 242
    context, 240
    definition, 239
    description, 240-241
    forces, 240
    references, 243
    sketch, 239-240
    variations, 242-243
  retrospectives
    adopting, 112
    business value, 109
    common problems, 112-113
    context, 110
    definition, 109
    description, 111-112
    forces, 110-111
    references, 113
    sketch, 109-110
    variations, 113
  stand-up meetings
    adopting, 95-96
    business value, 93
    common problems, 96
    definition, 93
    description, 94-95
    references, 98
    variations, 97
"Mock Roles, Not Objects" (paper), 169
models, Responsibility Process model, 18
  adopting, 16
  illustration, 15
  individual responsibility, 18
  overview, 15-17
  and response to change, 18
  responsibility of agile team members, 17

O
ObjectMother, 169
OO analysis and design, 332
organizational goals, 26
OS teams, 308-309

P
pair programming
  adopting, 225-226
  business value, 223
  common problems, 226
  context, 224
  definition, 8, 223
  description, 224-225
  forces, 224
  references, 227
  sample adoption scenario, 223
  variations, 226-227
patterns. *See* adoption patterns
personal agility, 18-19
pigs, 96
Planning Poker pattern
  adoption, 89-90
  business value, 87
  common problems, 90-91
  context, 88
  definition, 87
  description, 88-89
  forces, 88
  references, 91
  sketch, 87-88
practices. *See specific practices*
process smells
  bottlenecked resources, 33-34, 47
  churning projects, 34, 47
  definition, 32
  direct and regular customer input is unrealistic, 32-33, 45
  hardening phase needed at end of release cycle, 34-35, 48
  hundreds of bugs in bug tracker, 34, 47
  integration is infrequent, 35
  lack of visibility, 33, 46
product backlogs, 82
products
  lifetime
    increasing, 26
    practices, 43
  utility
    increasing, 24
    practices, 39
projects, churning, 34, 47, 324-325

Q
QA teams, 306-307
quality
  to market, 24, 40, 323
  unacceptable, 30

R
reading patterns, 328-329
Reading Circles
  adopting, 241-242
  business value, 239
  common problems, 242
  context, 240
  definition, 239
  description, 240-241
  forces, 240
  references, 243
  sketch, 239-240
  variations, 242-243
recognizing change, 5-7
reducing
  costs, 25, 42
  time to market, 23
refactoring
  adopting, 185
  business value, 183
  common problems, 186
  context, 184
  definition, 183

description, 184-185
forces, 184
references, 187
sample adoption scenario, 183-184
variations, 186-187
Refactoring to Patterns, 187
release cycle, 34-35, 48, 325
releases, 7
releasing often
adopting, 117
business value, 115
common problems, 118
context, 116
description, 117
forces, 117
overview, 115
references, 118
sketch, 116
variations, 118
Remote Output teams, 308
resources
bottlenecked resources, 33-34, 47, 325
on lean manufacturing, 332
on OO analysis and design, 332
on theory of constraints, 331
on working with people, 331
responding to change, 5-7
Responsibility Process model
adopting, 16
illustration, 15
individual responsibility, 18
overview, 15-17
and response to change, 18
responsibility of agile team members, 17
retrospectives
adopting, 112
business value, 109
common problems, 112-113
context, 110
definition, 109
description, 111-112
forces, 110-111
overview, 6
references, 113
sketch, 109-110
variations, 113

S

S.M.A.R.T. goals, 62
Scheduler teams, 310
scrum of scrums, 7
self-organizing teams
adopting, 127-128
business value, 125
common problems, 128
context, 126
definition, 8
description, 127
forces, 126-127
overview, 125
references, 129
sketch, 125-126
variations, 129
setting goals. *See* goals
simple design
adopting, 200
business value, 197
common problems, 200-201
context, 198
definition, 197
description, 199
forces, 198-199
references, 201
sample adoption scenario, 197-198
variations, 201
smells
agile practice to smell mappings, 43-48, 325
business smells
customer asks for everything, 32, 45
definition, 30
delivering new features to customer, 30
features are not used by customer, 30-31
quality delivered to customer is, 30

software is to expensive to build, 31
software not useful to customer, 31
us versus them, 31-32, 44
definition, 29
finding, 36
functional tests
architecture smells, 214-215
implementation smells, 211-213
references, 217
variations, 215-217
as incentives, 35
mapping to patterns, 325
overview, 29
process smells
bottlenecked resources, 33-34, 47
churning projects, 34, 47
definition, 32
direct and regular customer input is unrealistic, 32-33, 45
hardening phase needed at end of release cycle, 34-35, 48
hundreds of bugs in bug tracker, 34, 47
integration is infrequent, 35
lack of visibility, 33, 46
software, releasing often
adopting, 117
business value, 115
common problems, 118
context, 116
description, 117
forces, 117
overview, 115
references, 118
sketch, 116
variations, 118
stand-up meetings
adopting, 95-96
business value, 93
common problems, 96
context, 93
definition, 8, 93
description, 94-95
forces, 94
references, 98
sample adoption scenario, 93
variations, 97
states, done. *See* done state
storyboards, 160
Stranger in a Strange Land, 97*n*
strategies for adoption
building, 51
test-driven adoption strategies, 49-50
supporting practices
Classroom Training pattern
adoption, 250-251
business value, 249
common problems, 252
context, 250
description, 250
forces, 250
overview, 249
sketch, 249
variations, 252-253
Coach pattern
adoption, 232-233
business value, 231
common problems, 233
context, 232
definition, 231
description, 232
forces, 232
references, 234
sketch, 231
variations, 234
Engage the Community pattern
adoption, 236-238
business values, 235
common problems, 238
context, 236
description, 236
forces, 236
overview, 235

references, 238
sketches, 235
variations, 238
Reading Circle pattern
adoption, 241-242
business value, 239
common problems, 242
context, 240
definition, 239
description, 240-241
forces, 240
references, 243
sketch, 239-240
variations, 242-243
Workshop pattern
adoption, 247
business value, 245
common problems, 247
context, 246
description, 246-247
force, 246
overview, 245
references, 248
sketch, 245-246
variations, 247
Systems teams, 307

T

TDD. *See* test-driven development
TDR. *See* test-driven requirements
teams
character roles, 58-59
coaches
adoption, 232-233
business value, 231
common problems, 233
context, 232
definition, 231
description, 232
forces, 232
references, 234
sample adoption scenario, 231
variations, 234
collective code ownership
adopting, 221
business value, 219
common problems, 222
context, 220
description, 221
forces, 220
overview, 219
references, 222
sample adoption scenario, 219
variations, 222
co-located teams
adoption, 121-122
business value, 119
common problems, 122
context, 120
definition, 8
description, 121
forces, 120
overview, 119
references, 123
sketch, 119-120
variations, 122
Company X case study
Digital Output team, 309-310
Imaging team, 310
OS team, 308-309
QA team, 306-307
Remote Output team, 308
Scheduler team, 310
Systems team, 307
concurrent membership on multiple teams, 33-34
cross-functional teams
adopting, 134
business value, 131
common problems, 134-135
context, 132
description, 133

forces, 133
overview, 131
references, 135
sample adoption scenario, 132
variations, 135
customers, including on teams
adoption, 139-140
business value, 137
common problems, 140-141
context, 138
description, 139
forces, 138
overview, 137
references, 142
sample adoption, 137-138
variations, 142
pair programming
adopting, 225-226
business value, 223
common problems, 226
context, 224
definition, 8, 223
description, 224-225
forces, 224
references, 227
sample adoption scenario, 223
variations, 226-227
personal agility, 18-19
questions to ask, 327
references and further reading, 331
responsibility, 17
self-organizing teams
adopting, 127-128
business value, 125
common problems, 128
context, 126
definition, 8
description, 127
forces, 126-127
overview, 125
references, 129
sketch, 125-126
variations, 129
technical practices
Automated Developer Tests pattern
adoption, 167-170
business, 164
common, 170-171
context, 165
definition, 163
description, 166-167
forces, 165-166
references, 172
sketch, 164-165
variations, 171-172
Collective Code Ownership pattern
adoption, 221
business value, 219
common problems, 222
context, 220
description, 221
forces, 220
overview, 219
references, 222
sketch, 219
variations, 222
Continuous Integration pattern
adoption, 191-194
business, 189
common problems, 194-195
context, 190
definition, 189
description, 191
forces, 190
references, 196
sketch, 189-190
variations, 195-196
Functional Tests pattern
adoption, 210-211
architecture smells, 214-215
benefits of, 208-210
business value, 203
context, 204
definition, 203

description, 206
forces, 204-205
implementation smells, 211-213
Item Inventory, 206-208
references, 217
sketch, 203-204
variations, 215-217
Pair Programming pattern
adoption, 225-226
business value, 223
common problems, 226
context, 224
definition, 223
description, 224-225
forces, 224
references, 227
sketch, 223
variations, 226-227
Refactoring pattern
adoption, 185
business value, 183
common problems, 186
context, 184
definition, 183
description, 184-185
forces, 184
references, 187
sketch, 183-184
variations, 186-187
Simple Design pattern
adoption, 200
business value, 197
common problems, 200-201
context, 198
definition, 197
description, 199
forces, 198
references, 201
sketch, 197-198
variations, 201
Test-First Development pattern
adoption, 180-181
business value, 177
common problems, 181
context, 178
definition, 177
description, 179-180
forces, 178-179
references, 182
sketch, 177-178
variations, 181
Test-Last Development pattern
adoption, 175
business value, 173
common problems, 175-176
context, 174
definition, 173
description, 175
forces, 174
references, 176
sketch, 173-174
test-driven adoption strategies, 49-50
test-driven development
adoption, 280-281
business value, 277
common problems, 281-282
context, 278
description, 279-280
forces, 278-279
overview, 277
references, 282-283
sample adoption scenario, 277-278
variations, 282
test-driven requirements, 6
adoption, 288
business value, 285
common problems, 289-290
context, 286-287
description, 287-288
forces, 287
overview, 285
references, 290-291
sample business scenario, 286
variations, 290

test-first development, 62
  adopting, 180-181
  business value, 177
  common problems, 181
  context, 178
  definition, 177
  description, 179-180
  forces, 178-179
  overview, 6
  references, 182
  sample adoption scenario, 177-178
  variations, 181
testing
  automated developer tests
    adopting, 167-170
    business value, 164
    common problems, 170-171
    context, 165
    definition, 163
    description, 166-167
    forces, 165-166
    references, 172
    sample adoption scenario, 164-165
    variations, 171-172
  functional tests
    adopting, 210-211
    architecture smells, 214-215
    benefits of, 208-210
    business value, 203
    context, 204
    definition, 203
    description, 206
    forces, 204-205
    implementation smells, 211-213
    Item Inventory Management Test, 206-208
    references, 217
    sample adoption scenario, 203-204
    variations, 215-217
  management tests, 7
  test-driven development
    adoption, 280-281
    business value, 277
    common problems, 281-282
    context, 278
    description, 279-280
    forces, 278-279
    overview, 277
    references, 282-283
    sample adoption scenario, 277-278
    variations, 282
  test-driven requirements, 6
    adoption, 288
    business value, 285
    common problems, 289-290
    context, 286-287
    description, 287-288
    forces, 287
    overview, 285
    references, 290-291
    sample business scenario, 286
    variations, 290
  test-first development
    adopting, 180-181
    business value, 177
    common problems, 181
    context, 178
    definition, 177
    description, 179-180
    forces, 178-179
    overview, 6
    references, 182
    sample adoption scenario, 177-178
    variations, 181
  test-last development
    adopting, 175
    business value, 173
    common problems, 175-176
    context, 174
    definition, 173
    description, 175
    forces, 174

references, 176
sample adoption scenario, 173-174
test-last development
adopting, 175
business value, 173
common problems, 175-176
context, 174
definition, 173
description, 175
forces, 174
references, 176
sample adoption scenario, 173-174
theory of constraints, 331
time to market
practices, 38
reducing, 23, 323
training, classroom training
adopting, 250-251
business value, 249
common problems, 252
context, 250
description, 250
forces, 250
overview, 249
sample adoption scenario, 249
variations, 252-253
Tuckman, Bruce, 3*n*

U

unsuccessful adoptions, 14
us versus them (business smell), 31-32, 44, 323-325
use cases
adopting, 155
business value, 153
common problems, 155-156
context, 154
definition, 153
description, 154
forces, 154
references, 156
sample adoption scenario, 153
variations, 156
user groups, 237
user stories
adopting, 151
business value, 149
common problems, 151-152
context, 150
definition, 149
description, 150-151
forces, 150
references, 152
sample adoption scenario, 149-150
variations, 152
utility of products, increasing, 24

V

value to market, increasing, 323
visibility
increasing, 25
lack of, 33, 46
practices, 42

W-X-Y-Z

Working Effectively with Legacy Code, 171
workshops
adopting, 247
business value, 245
common problems, 247
context, 246
description, 246-247
force, 246
overview, 245
references, 248
sample adoption scenario, 245-246
variations, 247
Writing Effective Use Cases, 155

XP Days, 237