The Addison-Wesley Signature Series

A Mike Cohn Signature Book

Agile Testing

A Practical Guide for Testers and Agile Teams

Lisa Crispin
Janet Gregory

Forewords by Mike Cohn and Brian Marick

Library of Congress Cataloging-in-Publication Data:

Crispin, Lisa.
Agile testing : a practical guide for testers and agile teams /
Lisa Crispin, Janet Gregory. — 1st ed.
p. cm.
Includes bibliographical references and index.
ISBN-13: 978-0-321-53446-0 (pbk. : alk. paper)
ISBN-10: 0-321-53446-8 (pbk. : alk. paper) 1. Computer software—
Testing. 2. Agile software development. I. Gregory, Janet. II. Title.

QA76.76.T48C75 2009
005.1—dc22
2008042444

ISBN-13: 978-0-321-53446-0
ISBN-10: 0-321-53446-8
Text printed in the United States on recycled paper at R.R. Donnelley in Crawfordsville, Indiana.
First printing, December 2008

Foreword

By Mike Cohn

"Quality is baked in," the programmers kept telling me. As part of a proposed acquisition, my boss had asked me to perform some final due diligence on the development team and its product. We'd already established that the company's recently launched product was doing well in the market, but I was to make sure we were not about to buy more trouble than benefit. So I spent my time with the development team. I was looking for problems that might arise from having rushed the product into release. I wondered, "Was the code clean? Were there modules that could only be worked on by one developer? Were there hundreds or thousands of defects waiting to be discovered?" And when I asked about the team's approach to testing, "Quality is baked in" was the answer I got.

Because this rather unusual colloquialism could have meant just about anything, I pressed further. What I found was that this was the company founder's shorthand for expressing one of quality pioneer W. Edwards Deming's famous fourteen points: Build quality into the product rather than trying to test it in later.

The idea of building quality into their products is at the heart of how agile teams work. Agile teams work in short iterations in part to ensure that the application remains at a known state of quality. Agile teams are highly cross-functional, with programmers, testers, and others working side by side throughout each iteration so that quality can be baked into products through techniques such as acceptance-test driven development, a heavy emphasis on automated testing, and whole-team thinking. Good agile teams bake quality in by building their products continuously, integrating new work within minutes of its being completed. Agile teams utilize techniques such as refactoring and a preference for simplicity in order to prevent technical debt from accumulating.

Learning how to do these things is difficult, and especially so for testers, whose role has been given scant attention in previous books. Fortunately, the book you now hold in your hands answers questions on the mind of every tester who's beginning to work on an agile project, such as:

- What are my roles and responsibilities?
- How do I work more closely with programmers?
- How much do we automate, and how do we start automating?

The experience of Lisa and Janet shines through on every page of the book. However, this book is not just their story. Within this book, they incorporate dozens of stories from real-world agile testers. These stories form the heart of the book and are what makes it so unique. It's one thing to shout from the ivory tower, "Here's how to do agile testing." It's another to tell the stories of the teams that have struggled and then emerged agile and victorious over challenges such as usability testing, legacy code that resists automation, transitioning testers used to traditional phase-gate development, testing that "keeps up" with short iterations, and knowing when a feature is "done."

Lisa and Janet were there at the beginning, learning how to do agile testing back when the prevailing wisdom was that agile teams didn't need testers and that programmers could bake quality in by themselves. Over the years and through articles, conference presentations, and working with their clients and teams, Lisa and Janet have helped us see the rich role to be filled by testers on agile projects. In this book, Lisa and Janet use a test automation pyramid, the agile testing quadrants of Brian Marick (himself another world-class agile tester), and other techniques to show how much was missing from a mind-set that said testing is necessary but testers aren't.

If you want to learn how to bake quality into your products or are an aspiring agile tester seeking to understand your role, I can think of no better guides than Lisa and Janet.

Foreword

By Brian Marick

Imagine yourself skimming over a landscape thousands of years ago, looking at the people below. They're barely scraping out a living in a hostile territory, doing some hunting, some fishing, and a little planting. Off in the distance, you see the glitter of a glacier. Moving closer, you see that it's melting fast and that it's barely damming a huge lake. As you watch, the lake breaks through, sweeping down a riverbed, carving it deeper, splashing up against cliffs on the far side of the landscape—some of which collapse.

As you watch, the dazed inhabitants begin to explore the opening. On the other side, there's a lush landscape, teaming with bigger animals than they've ever seen before, some grazing on grass with huge seed heads, some squabbling over mounds of fallen fruit.

People move in. Almost immediately, they begin to live better. But as the years fly past, you see them adapt. They begin to use nets to fish in the fast-running streams. They learn the teamwork needed to bring down the larger animals, though not without a few deaths along the way. They find ever-better ways to cultivate this new grass they've come to call "wheat."

As you watch, the mad burst of innovation gives way to a stable solution, a good way to live in this new land, a way that's taught to each new generation. Although just over there, you spy someone inventing the wheel . . .

▪ ▪ ▪

In the early years of this century, the adoption of Agile methods sometimes seemed like a vast dam breaking, opening up a way to a better—more productive, more joyful—way of developing software. Many early adopters saw benefits right away, even though they barely knew what they were doing.

Some had an easier time of it than others. Programmers were like the hunters in the fable above. Yes, they had to learn new skills in order to hunt bison, but they knew how to hunt rabbits, more or less, and there were plenty of rabbits around. Testers were more like spear-fishers in a land where spear-fishing wouldn't work. Going from spear-fishing to net-fishing is a much bigger conceptual jump than going from rabbit to bison. And, while some of the skills—cleaning fish, for example—were the same in the new land, the testers had to invent new skills of net-weaving before they could truly pull their weight.

So testing lagged behind. Fortunately, we had early adopters like Lisa and Janet, people who dove right in alongside the programmers, testers who were not jealous of their role or their independence, downright *pleasant* people who could figure out the biggest change of all in Agile testing: the tester's new social role.

As a result, we have this book. It's the stable solution, the good way for testers to live in this new Agile land of ours. It's not the final word—we *could* use the wheel, and I myself am eager for someone to invent antibiotics—but what's taught here will serve you well until someone, perhaps Lisa and Janet, brings the next big change.

Preface

We were early adopters of Extreme Programming (XP), testing on XP teams that weren't at all sure where testers or their brand of testing fit in. At the time, there wasn't much in the agile (which wasn't called agile yet) literature about acceptance testing, or how professional testers might contribute. We learned not only from our own experiences but from others in the small agile community. In 2002, Lisa co-wrote *Testing Extreme Programming* with Tip House, with lots of help from Janet. Since then, agile development has evolved, and the agile testing community has flourished. With so many people contributing ideas, we've learned a whole lot more about agile testing.

Individually and together, we've helped teams transition to agile, helped testers learn how to contribute on agile teams, and worked with others in the agile community to explore ways that agile teams can be more successful at testing. Our experiences differ. Lisa has spent most of her time as an agile tester on stable teams working for years at a time on web applications in the retail, telephony, and financial industries. Janet has worked with software organizations developing enterprise systems in a variety of industries. These agile projects have included developing a message-handling system, an environmental-tracking system, a remote data management system (including an embedded application, with a communication network as well as the application), an oil and gas production accounting application, and applications in the airline transportation industry. She has played different roles—sometimes tester, sometimes coach—but has always worked to better integrate the testers with the rest of the team. She has been with teams from as little as six months to as long as one-and-a-half years.

With these different points of view, we have learned to work together and complement each other's skill sets, and we have given many presentations and tutorials together.

WHY WE WROTE THIS BOOK

Several excellent books oriented toward agile development on testing and test patterns have been published (see our bibliography). These books are generally focused on helping the developer. We decided to write a book aimed at helping agile teams be more successful at delivering business value using tests that the business can understand. We want to help testers and quality assurance (QA) professionals who have worked in more traditional development methodologies make the transition to agile development.

We've figured out how to apply—on a practical, day-to-day level—the fruits of our own experience working with teams of all sizes and a variety of ideas from other agile practitioners. We've put all this together in this book to help testers, quality assurance managers, developers, development managers, product owners, and anyone else with a stake in effective testing on agile projects to deliver the software their customers need. However, we've focused on the role of the tester, a role that may be adopted by a variety of professionals.

Agile testing practices aren't limited to members of agile teams. They can be used to improve testing on projects using traditional development methodologies as well. This book is also intended to help testers working on projects using any type of development methodology.

Agile development isn't the only way to successfully deliver software. However, all of the successful teams we've been on, agile or waterfall, have had several critical commonalities. The programmers write and automate unit and integration tests that provide good code coverage. They are disciplined in the use of source code control and code integration. Skilled testers are involved from the start of the development cycle and are given time and resources to do an adequate job of all necessary forms of testing. An automated regression suite that covers the system functionality at a higher level is run and checked regularly. The development team understands the customers' jobs and their needs, and works closely together with the business experts.

People, not methodologies or tools, make projects successful. We enjoy agile development because its values, principles, and core practices enable people to do their best work, and testing and quality are central to agile development. In this book, we explain how to apply agile values and principles to your unique testing situation and enable your teams to succeed. We have more about that in Chapter 1, "What Is Agile Testing, Anyway?" and in Chapter 2, "Ten Principles for Agile Testers."

How We Wrote This Book

Having experienced the benefits of agile development, we used agile practices to produce this book. As we began work on the book, we talked to agile testers and teams from around the globe to find out what problems they encountered and how they addressed them. We planned how we would cover these areas in the book.

We made a release plan based on two-week iterations. Every two weeks, we delivered two rough-draft chapters to our book website. Because we aren't co-located, we found tools to use to communicate, provide "source code control" for our chapters, deliver the product to our customers, and get their feedback. We couldn't "pair" much real-time, but we traded chapters back and forth for review and revision, and had informal "stand-ups" daily via instant message.

Our "customers" were the generous people in the agile community who volunteered to review draft chapters. They provided feedback by email or (if we were lucky) in person. We used the feedback to guide us as we continued writing and revising. After all the rough drafts were done, we made a new plan to complete the revisions, incorporating all the helpful ideas from our "customers."

Our most important tool was mind maps. We started out by creating a mind map of how we envisioned the whole book. We then created mind maps for each section of the book. Before writing each chapter, we brainstormed with a mind map. As we revised, we revisited the mind maps, which helped us think of ideas we may have missed.

Because we think the mind maps added so much value, we've included the mind map as part of the opening of each chapter. We hope they'll help you get an overview of all the information included in the chapter, and inspire you to try using mind maps yourself.

Our Audience

This book will help you if you've ever asked any of the following excellent questions, which we've heard many times:

- If developers are writing tests, what do the testers do?
- I'm a QA manager, and our company is implementing agile development (Scrum, XP, DSDM, name your flavor). What's my role now?

- I've worked as a tester on a traditional waterfall team, and I'm really excited by what I've read about agile. What do I need to know to work on an agile team?
- What's an "agile tester"?
- I'm a developer on an agile team. We're writing code test-first, but our customers still aren't happy with what we deliver. What are we missing?
- I'm a developer on an agile team. We're writing our code test-first. We make sure we have tests for all our code. Why do we need testers?
- I coach an agile development team. Our QA team can't keep up with us, and testing always lags behind. Should we just plan to test an iteration behind development?
- I'm a software development manager. We recently transitioned to agile, but all our testers quit. Why?
- I'm a tester on a team that's going agile. I don't have any programming or automation skills. Is there any place for me on an agile team?
- How can testing possibly keep up with two-week iterations?
- What about load testing, performance testing, usability testing, all the other "ilities"? Where do these fit in?
- We have audit requirements. How does agile development and testing address these?

If you have similar questions and you're looking for practical advice about how testers contribute to agile teams and how agile teams can do an effective job of testing, you've picked up the right book.

There are many "flavors" of agile development, but they all have much in common. We support the Agile Manifesto, which we explain in Chapter 1, "What Is Agile Testing, Anyway?" Whether you're practicing Scrum, Extreme Programming, Crystal, DSDM, or your own variation of agile development, you'll find information here to help with your testing efforts.

A User Story for an Agile Testing Book

When Robin Dymond, a managing consultant and trainer who has helped many teams adopt lean and agile, heard we were writing this book, he sent us the user story he'd like to have fulfilled. It encapsulates many of the requirements we planned to deliver.

Book Story 1

As a QA professional, I can understand the main difference between traditional QA professionals and agile team members with a QA background, so that I can begin internalizing my new responsibilities and deliver value to the customer sooner and with less difficulty.

Acceptance conditions:

- My concerns and fears about losing control of testing are addressed.
- My concerns and fears about having to write code (never done it) are addressed.
- As a tester I understand my new value to the team.
- As a tester new to Agile, I can easily read about things that are most important to my new role.
- As a tester new to Agile, I can easily ignore things that are less important to my new role.
- As a tester new to Agile, I can easily get further detail about agile testing that is important to MY context.

Were I to suggest a solution to this problem, I think of Scrum versus XP. With Scrum you get a simple view that enables people to quickly adopt Agile. However, Scrum is the tip of the iceberg for successful agile teams. For testers who are new, I would love to see agile testing ideas expressed in layers of detail. What do I need to know today, what should I know tomorrow, and what context-sensitive things should I consider for continuous improvement?

We've tried to provide these layers of detail in this book. We'll approach agile testing from a few different perspectives: transitioning into agile development, using an agile testing matrix to guide testing efforts, and explaining all the different testing activities that take place throughout the agile development cycle.

How to Use This Book

If you aren't sure where to start in this book, or you just want a quick overview, we suggest you read the last chapter, Chapter 22, "Key Success Factors," and follow wherever it leads you.

Part I: Introduction

If you want quick answers to questions such as "Is agile testing different than testing on waterfall projects?" or "What's the difference between a tester on a traditional team and an agile tester?," start with Part I, which includes the following chapters:

- Chapter 1: What Is Agile Testing, Anyway?
- Chapter 2: Ten Principles for Agile Testers

These chapters are the "tip of the iceberg" that Robin requested in his user story. They include an overview of how agile differs from a traditional phased approach and explore the "whole team" approach to quality and testing.

In this part of the book we define the "agile testing mind-set" and what makes testers successful on agile teams. We explain how testers apply agile values and principles to contribute their particular expertise.

Part II: Organizational Challenges

If you're a tester or manager on a traditional QA team, or you're coaching a team that's moving to agile, Part II will help you with the organizational challenges faced by teams in transition. The "whole team" attitude represents a lot of cultural changes to team members, but it helps overcome the fear testers have when they wonder how much control they'll have or whether they'll be expected to write code.

Some of the questions answered in Part II are:

- How can we engage the QA team?
- What about management's expectations?
- How should we structure our agile team, and where do the testers fit?
- What do we look for when hiring an agile tester?
- How do we cope with a team distributed across the globe?

Part II also introduces some topics we don't always enjoy talking about. We explore ideas about how to transition processes and models, such as audits or SOX compliance, that are common in traditional environments.

Metrics and how they're applied can be a controversial issue, but there are positive ways to use them to benefit the team. Defect tracking easily becomes a point of contention for teams, with questions such as "Do we use a defect-tracking system?" or "When do we log bugs?"

Two common questions about agile testing from people with traditional test team experience are "What about test plans?" and "Is it true there's no documentation on agile projects?" Part II clears up these mysteries.

The chapters in Part II are as follows:

- Chapter 3: Cultural Challenges
- Chapter 4: Team Logistics
- Chapter 5: Transitioning Typical Processes

Part III: The Agile Testing Quadrants

Do you want more details on what types of testing are done on agile projects? Are you wondering who does what testing? How do you know whether you've done all the testing that's needed? How do you decide what practices, techniques, and tools fit your particular situation? If these are your concerns, check out Part III.

We use Brian Marick's Agile Testing Quadrants to explain the purpose of testing. The quadrants help you define all the different areas your testing should address, from unit level tests to reliability and other "ilities," and everything in between. This is where we get down into the nitty-gritty of how to deliver a high-quality product. We explain techniques that can help you to communicate well with your customers and better understand their requirements. This part of the book shows how tests drive development at multiple levels. It also provides tools for your toolkit that can help you to effectively define, design, and execute tests that support the team and critique the product. The chapters include the following:

- Chapter 6: The Purpose of Testing
- Chapter 7: Technology-Facing Tests that Support the Team

- Chapter 8: Business-Facing Tests that Support the Team
- Chapter 9: Toolkit for Business-Facing Tests that Support the Team
- Chapter 10: Business-Facing Tests that Critique the Product
- Chapter 11: Critiquing the Product Using Technology-Facing Tests
- Chapter 12: Summary of Testing Quadrants

Part IV: Automation

Test automation is a central focus of successful agile teams, and it's a scary topic for lots of people (we know, because it's had us running scared before!). How do you squeeze test automation into short iterations and still get all the stories completed?

Part IV gets into the details of when and why to automate, how to overcome barriers to test automation, and how to develop and implement a test automation strategy that works for your team. Because test automation tools change and evolve so rapidly, our aim is not to explain how to use specific tools, but to help you select and use the right tools for your situation. Our agile test automation tips will help you with difficult challenges such as testing legacy code.

The chapters are as follows:

- Chapter 13: Why We Want to Automate Tests and What Holds Us Back
- Chapter 14: An Agile Test Automation Strategy

Part V: An Iteration in the Life of a Tester

If you just want to get a feel for what testers do throughout the agile development cycle, or you need help putting together all the information in this book, go to Part V. Here we chronicle an iteration, and more, in the life of an agile tester. Testers contribute enormous value throughout the agile software development cycles. In Part V, we explain the activities that testers do on a daily basis. We start with planning releases and iterations to get each iteration off to a good start, and move through the iteration—collaborating with the customer and development teams, testing, and writing code. We end the iteration by delivering new features and finding ways for the team to improve the process.

The chapters break down this way:

- Chapter 15: Tester Activities in Release or Theme Planning
- Chapter 16: Hit the Ground Running

- Chapter 17: Iteration Kickoff
- Chapter 18: Coding and Testing
- Chapter 19: Wrap Up the Iteration
- Chapter 20: Successful Delivery

Part VI: Summary

In Chapter 21, "Key Success Factors," we present seven key factors agile teams can use for successful testing. If you're having trouble deciding where to start with agile testing, or how to work on improving what you're doing now, these success factors will give you some direction.

Other Elements

We've also included a glossary we hope you will find useful, as well as references to books, articles, websites, and blogs in the bibliography.

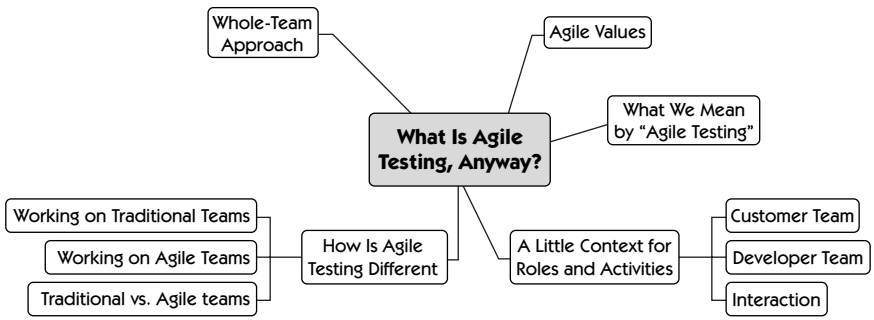
JUST START DOING IT—TODAY!

Agile development is all about doing your best work. Every team has unique challenges. We've tried to present all the information that we think may help agile testers, their teams, managers, and customers. Apply the techniques that you think are appropriate for your situation. Experiment constantly, evaluate the results, and come back to this book to see what might help you improve. Our goal is to help testers and agile teams enjoy delivering the best and most valuable product they can.

When we asked Dierk König, founder and project manager of Canoo WebTest, what he thought was the number one success factor for agile testing, he answered: "Start doing it—today!" You can take a baby step to improve your team's testing right now. Go get started!

Chapter 1

What Is Agile Testing, Anyway?



Like a lot of terminology, "agile development" and "agile testing" mean different things to different people. In this chapter, we explain our view of agile, which reflects the Agile Manifesto and general principles and values shared by different agile methods. We want to share a common language with you, the reader, so we'll go over some of our vocabulary. We compare and contrast agile development and testing with the more traditional phased approach. The "whole team" approach promoted by agile development is central to our attitude toward quality and testing, so we also talk about that here.

Agile Values

"Agile" is a buzzword that will probably fall out of use someday and make this book seem obsolete. It's loaded with different meanings that apply in different circumstances. One way to define "agile development" is to look at the Agile Manifesto (see Figure 1-1).

Using the values from the Manifesto to guide us, we strive to deliver small chunks of business value in extremely short release cycles.

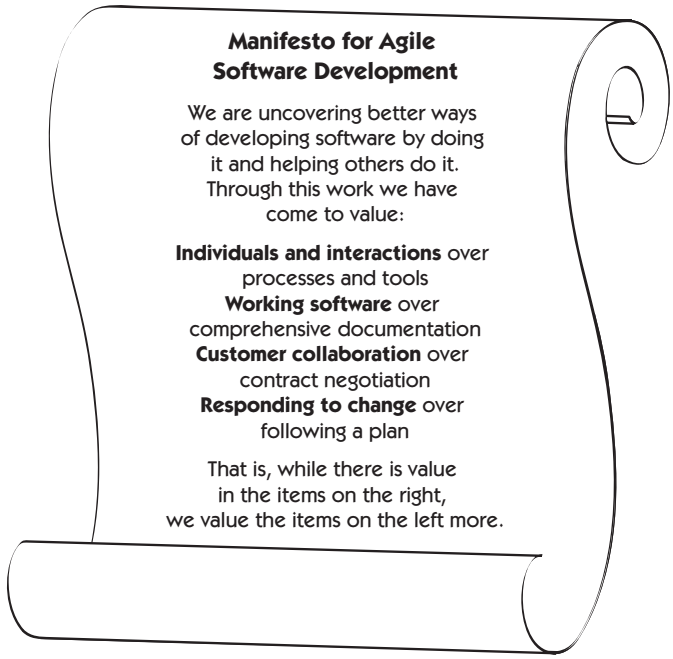


Figure 1-1 Agile Manifesto

Chapter 21, "Key Success Factors," lists key success factors for agile testing.

We use the word "agile" in this book in a broad sense. Whether your team is practicing a particular agile method, such as Scrum, XP, Crystal, DSDM, or FDD, to name a few, or just adopting whatever principles and practices make sense for your situation, you should be able to apply the ideas in this book. If you're delivering value to the business in a timely manner with high-quality software, and your team continually strives to improve, you'll find useful information here. At the same time, there are particular agile practices we feel are crucial to any team's success. We'll talk about these throughout the book.

WHAT DO WE MEAN BY "AGILE TESTING"?

You might have noticed that we use the term "tester" to describe a person whose main activities revolve around testing and quality assurance. You'll also see that we often use the word "programmer" to describe a person whose main activities revolve around writing production code. We don't intend that these terms sound narrow or insignificant. Programmers do more than turn a specification into a program. We don't call them "developers," because ev-

eryone involved in delivering software is a developer. Testers do more than perform "testing tasks." Each agile team member is focused on delivering a high-quality product that provides business value. Agile testers work to ensure that their team delivers the quality their customers need. We use the terms "programmer" and "tester" for convenience.

Several core practices used by agile teams relate to testing. Agile programmers use test-driven development (TDD), also called test-driven design, to write quality production code. With TDD, the programmer writes a test for a tiny bit of functionality, sees it fail, writes the code that makes it pass, and then moves on to the next tiny bit of functionality. Programmers also write code integration tests to make sure the small units of code work together as intended. This essential practice has been adopted by many teams, even those that don't call themselves "agile," because it's just a smart way to think through your software design and prevent defects. Figure 1-2 shows a sample unit test result that a programmer might see.

This book isn't about unit-level or component-level testing, but these types of tests are critical to a successful project. Brian Marick [2003] describes these types of tests as "supporting the team," helping the programmers know what code to write next. Brian also coined the term "technology-facing tests," tests that fall into the programmer's domain and are described using programmer terms and jargon. In Part II, we introduce the Agile Testing Quadrants and examine the different categories of agile testing. If you want to learn more about writing unit and component tests, and TDD, the bibliography will steer you to some good resources.

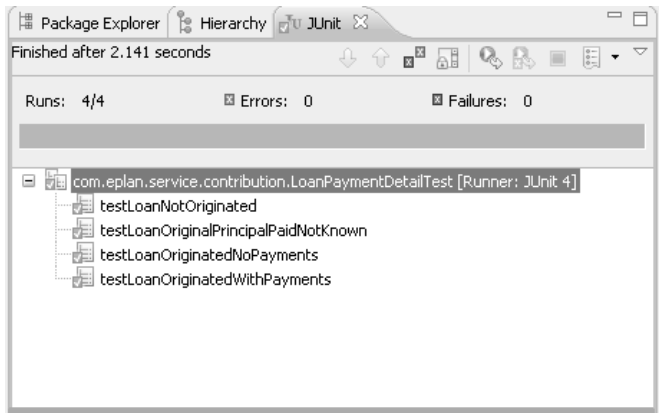


Figure 1-2 Sample unit test output

If you want to know how agile values, principles, and practices applied to testing can help you, as a tester, do your best work, and help your team deliver more business value, please keep reading. If you've bothered to pick up this book, you're probably the kind of professional who continually strives to grow and learn. You're likely to have the mind-set that a good agile team needs to succeed. This book will show you ways to improve your organization's product, provide the most value possible to your team, and enjoy your job.

Lisa's Story

During a break from working on this chapter, I talked to a friend who works in quality assurance for a large company. It was a busy time of year, and management expected everyone to work extra hours. He said, "If I thought working 100 extra hours would solve our problems, I'd work 'til 7 every night until that was done. But the truth was, it might take 4,000 extra hours to solve our problems, so working extra feels pointless." Does this sound familiar?

—Lisa

If you've worked in the software industry long, you've probably had the opportunity to feel like Lisa's friend. Working harder and longer doesn't help when your task is impossible to achieve. Agile development acknowledges the reality that we only have so many good productive hours in a day or week, and that we can't plan away the inevitability of change.

Agile development encourages us to solve our problems as a team. Business people, programmers, testers, analysts—everyone involved in software development—decides together how best to improve their product. Best of all, as testers, we're working together with a team of people who all feel responsible for delivering the best possible quality, and who are all focused on testing. We love doing this work, and you will too.

When we say "agile testing" in this book, we're usually talking about business-facing tests, tests that define the business experts' desired features and functionality. We consider "customer-facing" a synonym for "business-facing." "Testing" in this book also includes tests that critique the product and focus on discovering what might be lacking in the finished product so that we can improve it. It includes just about everything beyond unit and component level testing: functional, system, load, performance, security, stress, usability, exploratory, end-to-end, and user acceptance. All these types of tests might be appropriate to any given project, whether it's an agile project or one using more traditional methodologies.

Agile testing doesn't just mean testing on an agile project. Some testing approaches, such as exploratory testing, are inherently agile, whether it's done an agile project or not. Testing an application with a plan to learn about it as you go, and letting that information guide your testing, is in line with valuing working software and responding to change. Later chapters discuss agile forms of testing as well as "agile testing" practices.

A Little Context for Roles and Activities on an Agile Team

We'll talk a lot in this book about the "customer team" and the "developer team." The difference between them is the skills they bring to delivering a product.

Customer Team

The customer team includes business experts, product owners, domain experts, product managers, business analysts, subject matter experts—everyone on the "business" side of a project. The customer team writes the stories or feature sets that the developer team delivers. They provide the examples that will drive coding in the form of business-facing tests. They communicate and collaborate with the developer team throughout each iteration, answering questions, drawing examples on the whiteboard, and reviewing finished stories or parts of stories.

Testers are integral members of the customer team, helping elicit requirements and examples and helping the customers express their requirements as tests.

Developer Team

Everyone involved with delivering code is a developer, and is part of the developer team. Agile principles encourage team members to take on multiple activities; any team member can take on any type of task. Many agile practitioners discourage specialized roles on teams and encourage all team members to transfer their skills to others as much as possible. Nevertheless, each team needs to decide what expertise their projects require. Programmers, system administrators, architects, database administrators, technical writers, security specialists, and people who wear more than one of these hats might be part of the team, physically or virtually.

Testers are also on the developer team, because testing is a central component of agile software development. Testers advocate for quality on behalf of the customer and assist the development team in delivering the maximum business value.

Interaction between Customer and Developer Teams

The customer and developer teams work closely together at all times. Ideally, they're just one team with a common goal. That goal is to deliver value to the organization. Agile projects progress in iterations, which are small development cycles that typically last from one to four weeks. The customer team, with input from the developers, will prioritize stories to be developed, and the developer team will determine how much work they can take on. They'll work together to define requirements with tests and examples, and write the code that makes the tests pass. Testers have a foot in each world, understanding the customer viewpoint as well as the complexities of the technical implementation (see Figure 1-3).

Some agile teams don't have any members who define themselves as "testers." However, they all need someone to help the customer team write business-facing tests for the iteration's stories, make sure the tests pass, and make sure that adequate regression tests are automated. Even if a team does have testers, the entire agile team is responsible for these testing tasks. Our experience with agile teams has shown that testing skills and experience are vital to project success and that testers do add value to agile teams.

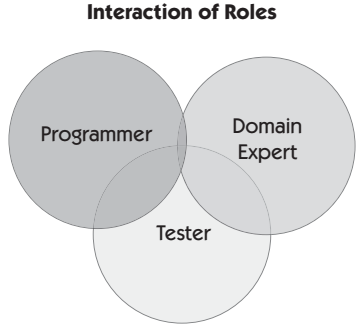


Figure 1-3 Interaction of roles

How Is Agile Testing Different?

We both started working on agile teams at the turn of the millennium. Like a lot of testers who are new to agile, we didn't know what to expect at first. Together with our respective agile teams, we've worked on we've learned a lot about testing on agile projects. We've also implemented ideas and practices suggested by other agile testers and teams. Over the years, we've shared our experiences with other agile testers as well. We've facilitated workshops and led tutorials at agile and testing conferences, talked with local user groups, and joined countless discussions on agile testing mailing lists. Through these experiences, we've identified differences between testing on agile teams and testing on traditional waterfall development projects. Agile development has transformed the testing profession in many ways.

Working on Traditional Teams

Neither working closely with programmers nor getting involved with a project from the earliest phases was new to us. However, we were used to strictly enforced gated phases of a narrowly defined software development life cycle, starting with release planning and requirements definition and usually ending with a rushed testing phase and a delayed release. In fact, we often were thrust into a gatekeeper role, telling business managers, "Sorry, the requirements are frozen; we can add that feature in the next release."

As leaders of quality assurance teams, we were also often expected to act as gatekeepers of quality. We couldn't control how the code was written, or even if any programmers tested their code, other than by our personal efforts at collaboration. Our post-development testing phases were expected to boost quality after code was complete. We had the illusion of control. We usually had the keys to production, and sometimes we had the power to postpone releases or stop them from going forward. Lisa even had the title of "Quality Boss," when in fact she was merely the manager of the QA team.

Our development cycles were generally long. Projects at a company that produced database software might last for a year. The six-month release cycles Lisa experienced at an Internet start-up seemed short at the time, although it was still a long time to have frozen requirements. In spite of much process and discipline, diligently completing one phase before moving on to the next, it was plenty of time for the competition to come out ahead, and the applications were not always what the customers expected.

Traditional teams are focused on making sure all the specified requirements are delivered in the final product. If everything isn't ready by the original target release date, the release is usually postponed. The development teams don't usually have input about what features are in the release, or how they should work. Individual programmers tend to specialize in a particular area of the code. Testers study the requirements documents to write their test plans, and then they wait for work to be delivered to them for testing.

Working on Agile Teams

Transitioning to the short iterations of an agile project might produce initial shock and awe. How can we possibly define requirements and then test and deliver production-ready code in one, two, three, or four weeks? This is particularly tough for larger organizations with separate teams for different functions and even harder for teams that are geographically dispersed. Where do all these various programmers, testers, analysts, project managers, and countless specialties fit in a new agile project? How can we possibly code and test so quickly? Where would we find time for difficult efforts such as automating tests? What control do we have over bad code getting delivered to production?

We'll share our stories from our first agile experiences to show you that everyone has to start somewhere.

Lisa's Story

My first agile team embraced Extreme Programming (XP), not without some "learning experiences." Serving as the only professional tester on a team of eight programmers who hadn't learned how to automate unit tests was disheartening. The first two-week iteration felt like jumping off a cliff.

Fortunately, we had a good coach, excellent training, a supportive community of agile practitioners with ideas to share, and time to learn. Together we figured out some ins and outs of how to integrate testing into an agile project—indeed, how to drive the project with tests. I learned how I could use my testing skills and experience to add real value to an agile team.

The toughest thing for me (the former Quality Boss) to learn was that the customers, not I, decided on quality criteria for the product. I was horrified after the first iteration to find that the code crashed easily when two users logged in concurrently. My coach patiently explained, over my strident objections, that our customer, a start-up company, wanted to be able to show features to potential customers. Reliability and robustness were not yet the issue.

I learned that my job was to help the customers tell us what was valuable to them during each iteration, and to write tests to ensure that's what they got.

—Lisa

Janet's Story

My first foray into the agile world was also an Extreme Programming (XP) engagement. I had just come from an organization that practiced waterfall with some extremely bad practices, including giving the test team a day or so to test six months of code. In my next job as QA manager, the development manager and I were both learning what XP really meant. We successfully created a team that worked well together and managed to automate most of the tests for the functionality. When the organization downsized during the dot-com bust, I found myself in a new position at another organization as the lone tester with about ten developers on an XP project.

On my first day of the project, Jonathan Rasmusson, one of the developers, came up to me and asked me why I was there. The team was practicing XP, and the programmers were practicing test-first and automating all their own tests. Participating in that was a challenge I couldn't resist. The team didn't know what value I could add, but I knew I had unique abilities that could help the team. That experience changed my life forever, because I gained an understanding of the nuances of an agile project and determined then that my life's work was to make the tester role a more fulfilling one.

—Janet

Read Jonathan's Story

Jonathan Rasmusson, now an Agile Coach at Rasmusson Software Consulting, but Janet's coworker on her second agile team, explains how he learned how agile testers add value.

> So there I was, a young hotshot J2EE developer excited and pumped to be developing software the way it should be developed—using XP. Until one day, in walks a new team member—a tester. It seems management thought it would be good to have a QA resource on the team.
>
> That's fine. Then it occurred to me that this poor tester would have nothing to do. I mean, as a developer on an XP project, I was writing the tests. There was no role for QA here as far as I could see.
>
> So of course I went up and introduced myself and asked quite pointedly what she was going to do on the project, because the developers were writing all the tests. While I can't remember exactly how Janet responded, the next six months made it very clear what testers can do on agile projects.
>
> With the automation of the tedious, low-level boundary condition test cases, Janet as a tester was now free to focus on much greater value-add areas like exploratory testing, usability, and testing the app in ways developers hadn't originally anticipated. She worked with the

customer to help write test cases that defined success for upcoming stories. She paired with developers looking for gaps in tests.

But perhaps most importantly, she helped reinforce an ethos of quality and culture, dispensing happy-face stickers to those developers who had done an exceptional job (these became much sought-after badges of honor displayed prominently on laptops).

Working with Janet taught me a great deal about the role testers play on agile projects, and their importance to the team.

Agile teams work closely with the business and have a detailed understanding of the requirements. They're focused on the value they can deliver, and they might have a great deal of input into prioritizing features. Testers don't sit and wait for work; they get up and look for ways to contribute throughout the development cycle and beyond.

If testing on an agile project felt just like testing on a traditional project, we wouldn't feel the need to write a book. Let's compare and contrast these testing methods.

Traditional vs. Agile Testing

It helps to start by looking at similarities between agile testing and testing in traditional software development. Consider Figure 1-4.

In the phased approach diagram, it is clear that testing happens at the end, right before release. The diagram is idealistic, because it gives the impression there is as much time for testing as there is for coding. In many projects, this is not the case. The testing gets "squished" because coding takes longer than expected, and because teams get into a code-and-fix cycle at the end.

Agile is iterative and incremental. This means that the testers test each increment of coding as soon as it is finished. An iteration might be as short as one week, or as long as a month. The team builds and tests a little bit of code, making sure it works correctly, and then moves on to next piece that needs to be built. Programmers never get ahead of the testers, because a story is not "done" until it has been tested. We'll talk much more about this throughout the book.

There's tremendous variety in the approaches to projects that agile teams take. One team might be dedicated to a single project or might be part of another

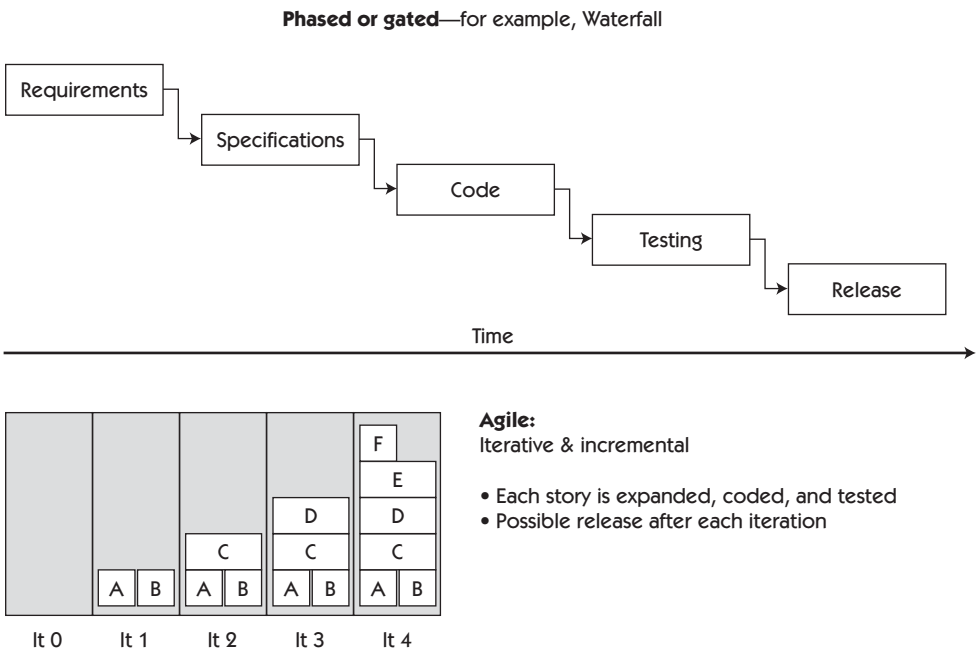


Figure 1-4 Traditional testing vs. agile testing

bigger project. No matter how big your project is, you still have to start somewhere. Your team might take on an epic or feature, a set of related stories at an estimating meeting, or you might meet to plan the release. Regardless of how a project or subset of a project gets started, you'll need to get a high-level understanding of it. You might come up with a plan or strategy for testing as you prepare for a release, but it will probably look quite different from any test plan you've done before.

Every project, every team, and sometimes every iteration is different. How your team solves problems should depend on the problem, the people, and the tools you have available. As an agile team member, you will need to be adaptive to the team's needs.

Rather than creating tests from a requirements document that was created by business analysts before anyone ever thought of writing a line of code, someone will need to write tests that illustrate the requirements for each story days or hours before coding begins. This is often a collaborative effort between a

business or domain expert and a tester, analyst, or some other development team member. Detailed functional test cases, ideally based on examples provided by business experts, flesh out the requirements. Testers will conduct manual exploratory testing to find important bugs that defined test cases might miss. Testers might pair with other developers to automate and execute test cases as coding on each story proceeds. Automated functional tests are added to the regression test suite. When tests demonstrating minimum functionality are complete, the team can consider the story finished.

If you attended agile conferences and seminars in the early part of this decade, you heard a lot about TDD and acceptance testing but not so much about other critical types of testing, such as load, performance, security, usability, and other "ility" testing. As testers, we thought that was a little weird, because all these types of testing are just as vital on agile projects as they are on projects using any other development methodology. The real difference is that we like to do these tests as early in the development process as we can so that they can also drive design and coding.

If the team actually releases each iteration, as Lisa's team does, the last day or two of each iteration is the "end game," the time when user acceptance testing, training, bug fixing, and deployments to staging environments can occur. Other teams, such as Janet's, release every few iterations, and might even have an entire iteration's worth of "end game" activities to verify release readiness. The difference here is that all the testing is not left until the end.

As a tester on an agile team, you're a key player in releasing code to production, just as you might have been in a more traditional environment. You might run scripts or do manual testing to verify all elements of a release, such as database update scripts, are in place. All team members participate in retrospectives or other process improvement activities that might occur for every iteration or every release. The whole team brainstorms ways to solve problems and improve processes and practices.

Agile projects have a variety of flavors. Is your team starting with a clean slate, in a greenfield (new) development project? If so, you might have fewer challenges than a team faced with rewriting or building on a legacy system that has no automated regression suite. Working with a third party brings additional testing challenges to any team.

Whatever flavor of development you're using, pretty much the same elements of a software development life cycle need to happen. The difference

with agile is that time frames are greatly shortened, and activities happen concurrently. Participants, tests, and tools need to be adaptive.

The most critical difference for testers in an agile project is the quick feedback from testing. It drives the project forward, and there are no gatekeepers ready to block project progress if certain milestones aren't met.

We've encountered testers who resist the transition to agile development, fearing that "agile development" equates with chaos, lack of discipline, lack of documentation, and an environment that is hostile to testers. While some teams do seem to use the "agile" buzzword to justify simply doing whatever they want, true agile teams are all about repeatable quality as well as efficiency. In our experience, an agile team is a wonderful place to be a tester.

WHOLE-TEAM APPROACH

One of the biggest differences in agile development versus traditional development is the agile "whole-team" approach. With agile, it's not only the testers or a quality assurance team who feel responsible for quality. We don't think of "departments," we just think of the skills and resources we need to deliver the best possible product. The focus of agile development is producing high-quality software in a time frame that maximizes its value to the business. This is the job of the whole team, not just testers or designated quality assurance professionals. Everyone on an agile team gets "test-infected." Tests, from the unit level on up, drive the coding, help the team learn how the application should work, and let us know when we're "done" with a task or story.

An agile team must possess all the skills needed to produce quality code that delivers the features required by the organization. While this might mean including specialists on the team, such as expert testers, it doesn't limit particular tasks to particular team members. Any task might be completed by any team member, or a pair of team members. This means that the team takes responsibility for all kinds of testing tasks, such as automating tests and manual exploratory testing. It also means that the whole team thinks constantly about designing code for testability.

The whole-team approach involves constant collaboration. Testers collaborate with programmers, the customer team, and other team specialists—and not just for testing tasks, but other tasks related to testing, such as building infrastructure and designing for testability. Figure 1-5 shows a developer reviewing reports with two customers and a tester (not pictured).

Figure 1-5 A developer discusses an issue with customers

The whole-team approach means everyone takes responsibility for testing tasks. It means team members have a range of skill sets and experience to employ in attacking challenges such as designing for testability by turning examples into tests and into code to make those tests pass. These diverse viewpoints can only mean better tests and test coverage.

Most importantly, on an agile team, anyone can ask for and receive help. The team commits to providing the highest possible business value as a team, and the team does whatever is needed to deliver it. Some folks who are new to agile perceive it as all about speed. The fact is, it's all about quality—and if it's not, we question whether it's really an "agile" team.

Your situation is unique. That's why you need to be aware of the potential testing obstacles your team might face and how you can apply agile values and principles to overcome them.

SUMMARY

Understanding the activities that testers perform on agile teams helps you show your own team the value that testers can add. Learning the core practices of agile testing will help your team deliver software that delights your customers.

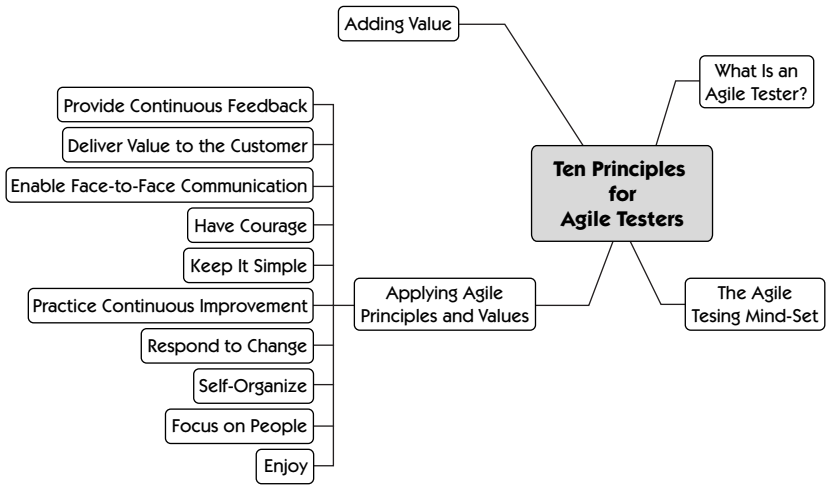
In this chapter, we've explained what we mean when we use the term "agile testing.

- We showed how the Agile Manifesto relates to testing, with its emphasis on individuals and interactions, working software, customer collaboration, and responding to change.
- We provided some context for this book, including some other terms we use such as "tester," "programmer," "customer," and related terms so that we can speak a common language.
- We explained how agile testing, with its focus on business value and delivering the quality customers require, is different from traditional testing, which focuses on conformance to requirements.
- We introduced the "whole-team" approach to agile testing, which means that everyone involved with delivering software is responsible for delivering high-quality software.
- We advised taking a practical approach by applying agile values and principles to overcome agile testing obstacles that arise in your unique situation.

This page intentionally left blank

Chapter 2

Ten Principles for Agile Testers



Everyone on an agile team is a tester. Anyone can pick up testing tasks. If that's true, then what is special about an agile tester? If I define myself as a tester on an agile team, what does that really mean? Do agile testers need different skill sets than testers on traditional teams? What guides them in their daily activities?

In this chapter, we talk about the agile testing mind-set, show how agile values and principles guide testing, and give an overview of how testers add value on agile teams.

What's an Agile Tester?

We define an agile tester this way: a professional tester who embraces change, collaborates well with both technical and business people, and understands the concept of using tests to document requirements and drive development. Agile testers tend to have good technical skills, know how to collaborate with

others to automate tests, and are also experienced exploratory testers. They're willing to learn what customers do so that they can better understand the customers' software requirements.

Who's an agile tester? She's a team member who drives agile testing. We know many agile testers who started out in some other specialization. A developer becomes test-infected and branches out beyond unit testing. An exploratory tester, accustomed to working in an agile manner, is attracted to the idea of an agile team. Professionals in other roles, such as business or functional analysts, might share the same traits and do much of the same work.

Skills are important, but attitude counts more. Janet likes to say, "Without the attitude, the skill is nothing." Having had to hire numerous testers for our agile teams, we've put a lot of thought into this and discussed it with others in the agile community. Testers tend to see the big picture. They look at the application more from a user or customer point of view, which means they're generally customer-focused.

THE AGILE TESTING MIND-SET

What makes a team "agile"? To us, an agile team is one that continually focuses on doing its best work and delivering the best possible product. In our experience, this involves a ton of discipline, learning, time, experimentation, and working together. It's not for everyone, but it's ideal for those of us who like the team dynamic and focus on continual improvement.

Successful projects are a result of good people allowed to do good work. The characteristics that make someone succeed as a tester on an agile team are probably the same characteristics that make a highly valued tester on any team.

An agile tester doesn't see herself as a quality police officer, protecting her customers from inadequate code. She's ready to gather and share information, to work with the customer or product owner in order to help them express their requirements adequately so that they can get the features they need, and to provide feedback on project progress to everyone.

Agile testers, and maybe any tester with the right skills and mind-set, are continually looking for ways the team can do a better job of producing high-quality software. On a personal level, that might mean attending local user group meetings or roundtables to find out what other teams are doing. It

INDEX

C

G

H

I

Q

S

T

U