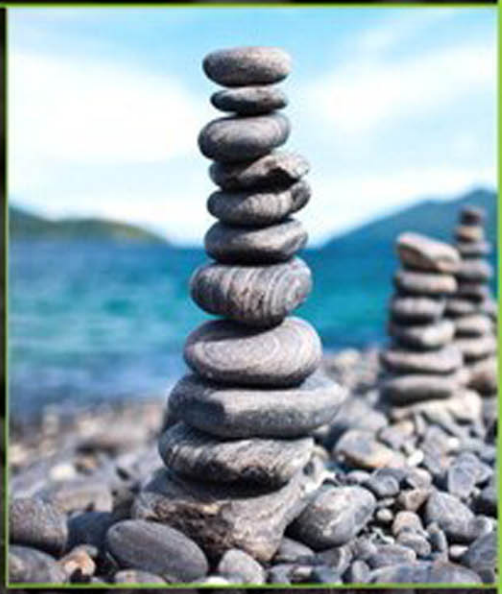
The Addison Wesley Signature Series

A Mike Cohn Signature Book

More Agile Testing

Learning Journeys for the Whole Team

Janet Gregory
Lisa Crispin



Forewords by Elisabeth Hendrickson and Johanna Rothman

Praise for *More Agile Testing*

"I love this book. It will help to create really great testers. That's a good thing, since anyone who reads this will want to have one on their team."

—*Liz Keogh, agile coach, Lunivore Limited*

"This book will change your thinking and move your focus from *tests* to *testing*. Yes, it is not about the result, but about the activity!"

—*Kenji Hiranabe, cofounder of Astah and CEO, Change Vision, Inc.*

"To my mind, agile development is about learning—that one word captures the true spirit of what agile is all about. When I had the chance to read through their new book, I could only say, 'Wow! Janet and Lisa have done themselves proud.' This is not a book about testing; this is a book about learning. Their clear explanations are accompanied by great true stories and an impressive list of books, articles, and other resources. Those of us who like learning, who love to dig for more information, can rejoice! I know you're always looking for something interesting and useful; I can guarantee that you will find it here!"

—*Linda Rising, coauthor of* Fearless Change: Patterns for Introducing New Ideas

"Janet and Lisa's first book, *Agile Testing*, drew some general principles that are still important today but left me wondering, 'how?' In this second book, they adapt those principles to today's development landscape—with mobile, DevOps, and cloud-based applications delivered in increasingly compressed release cycles. Readers get specific testing tools for the mind along with new practices and commentary to accelerate learning. Read it today."

—*Matt Heusser, Managing Principal, Excelon Development*

"An excellent guide for your team's agile journey, full of resources to help you with every kind of testing challenge you might meet along the way. Janet and Lisa share a wealth of experience with personal stories about how they helped agile teams figure out how to get value from testing. I really like how the book is filled with techniques explained by leading industry practitioners who've pioneered them in their own organizations."

—*Rachel Davies, agile coach, unruly and coauthor of* Agile Coaching

"Let me net this out for you: agile quality and testing is hard to get right. It's nuanced, context-based, and not for the faint of heart. In order to effectively balance it, you need hard-earned, pragmatic, real-world advice. This book has it—not only from Janet and Lisa, but also from forty additional expert agile practitioners. Get it and learn how to effectively drive quality into your agile products and across your entire organization."

—*Bob Galen, Principal Consultant, R Galen Consulting Group, and Author of* Agile Reflections *and* Scrum Product Ownership

Components are "Firmware," "Printing," and "File System" for an operating system project, or "Database," "Cart," and "Product Browser" for an online shopping site.

- **Capabilities** (verbs of the system) describe the abilities of a particular Component to satisfy the Attributes of the system. An example Capability for a shopping site could be "Processes monetary transactions using HTTPS." You can see that this could be a Capability of the "Cart" component when trying to meet the "Secure" Attribute. The most important aspect of Capabilities is that they are testable.

Creating a high-level matrix using this model can be a simple way to visualize your system. Figure 8-6 shows an example of what such a matrix might look like. Gojko Adzic agrees that exposing system characteristics and providing more visibility is definitely a good idea (Adzic, 2010a), though he cautions that while we can learn from other fields, we should be careful about using them as a metaphor for software development.

Use heuristics such as Elisabeth Hendrickson's "Test Heuristics Cheat Sheet" (Hendrickson, 2011) or tried-and-true techniques such as state diagrams or truth tables to think of new ideas for attributes. Combine these ideas with models like the Quadrants so that the conversations about the system constraints or usability can extract clear examples. Using all the tools in your toolbox can only help increase the quality of the product.

Components			Capabilities	Attributes		
Mobile App	Firmware	Printing		Fast	Secure	Stable
			Manage profile			
			Send messages			
			Update network			
INFLUENCE AREA				RISK / IMPORTANCE		

Figure 8-6 ACC example

PLANNING FOR TEST AUTOMATION

Since Mike Cohn came up with his test automation pyramid in 2003, many teams have found it a useful model to plan their test automation. To take advantage of fast feedback, we need to consider at what level our automation tests should be. When we look at the standard pyramid, Figure 8-7, we see three levels.

The lowest level is the base—the unit tests. When we consider testing, we should try to push the tests as low as they can go for the highest return on investment (ROI) and the quickest feedback.

However, when we have business logic where tests need to be visible to the business, we should use collaborative tools that create tests at the service layer (the API) to specify them in a way that documents system behavior. See Chapter 16, "Test Automation Design Patterns and Approaches," for

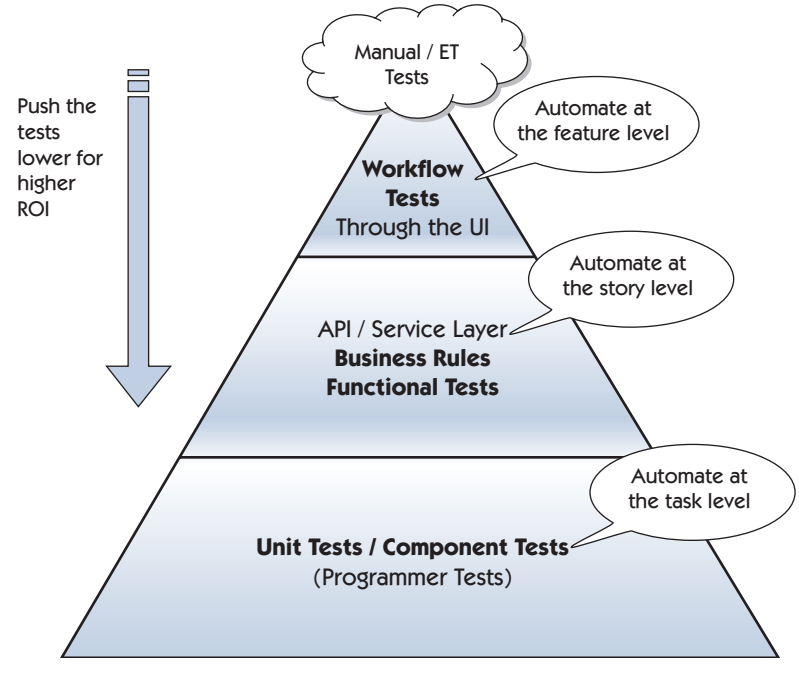


Figure 8-7 Automation pyramid

more details. It is at this layer that we can automate at the story level so that testing and automation can keep up with the coding.

The top layer of the pyramid consists of the workflow tests through the user interface (UI). If we have a high degree of confidence in the unit tests and the service-level or API-level tests, we can keep these slower, more brittle automated tests to a minimum. See Chapter 15, "Pyramids of Automation," for more detail on alternative pyramid models.

Practices such as guiding development with examples can help define what the best level for the test is. A team's cadence can be set by how well they plan and execute their automation and how well they understand the level of detail they need. Consider also how to make your automation test runs visible, whether displayed in the continuous integration environment or on a monitor that is in the open.

SUMMARY

Models are a useful tool for planning. In this chapter, we covered the following points:

- The agile testing quadrants provide a model for thinking about testing in an agile world.
  - The Quadrants help to emphasize the whole-team responsibility for testing.
  - They provide a visible mechanism for talking about the testing needed.
  - The left side is about guiding development, learning what to build, and preventing defects—testing early.
  - The right side is about critiquing the product, finding defects, and learning what capabilities are still missing.
- Gojko Adzic provides an alternative way to think about the Quadrants if you are in a lean startup or continuous delivery environment.
- We also introduced an alternative quadrant diagram from Elisabeth Hendrickson that highlights confirmatory checks versus investigative testing.

- There are already many tools in our agile testing toolbox, and we can combine them with other models such as the Quadrants to make our testing as effective as possible.
- FURPS and ACC are additional examples of models you can use to help plan based on risk and a variety of quality characteristics.
- The automation pyramid is a reminder to think about automation and to plan for it at the different levels.

This page intentionally left blank

Part IV

Testing Business Value

In her book *Explore It!* (Hendrickson, 2013), Elisabeth Hendrickson describes two facets of a test strategy (pp. 5–6). Per Elisabeth, "checks" are tests that verify that "the implementation behaves as intended under supported configurations and conditions." She explains exploring as testing that involves scouting around the areas that the checks don't cover, "designing and executing tiny experiments in rapid succession using the results from the last experiment to inform the next." In Elisabeth's opinion, a comprehensive test strategy needs both checking and exploring.

The distinction between checking and exploring is interesting, but we're not sure that testing definition goes far enough. Testing is more than "just" testing software. It is about testing ideas, helping business experts identify the most valuable features to develop next, finding a common understanding for each feature, preventing defects, and testing for business value.

A whole-team approach to quality means incorporating testing activities from the time a feature is first conceptualized. In this part, we'll cover some tools and techniques that let us help customers focus on features to help achieve their most valuable business goals. Testers make important contributions during these activities, and we'll point out what testers can learn from other specialists to help customers articulate requirements and examples of desired system behavior. We'll also go into detail on guiding development with examples and good ways to obtain those examples.