



# Executable Specifications *with* **Scrum**

A Practical Guide to Agile Requirements Discovery



**Mario Cardinal**

---

## Praise for *Executable Specifications with Scrum*

“This is a great book that demonstrates the value of putting effort behind requirements in an Agile environment, including both the business and technical value. The book is well-written and flows nicely, approachable for both the manager and the developer. I am recommending this book to all Scrum teams who need to integrate business analysts and architects as active teammates.”

—**Stephen Forte**, Chief Strategy Officer at Telerik and  
Board Member at the Scrum Alliance

“Cardinal’s book brings to light one of the most important and neglected aspects of Scrum: Having user stories that are ready to sprint. Teams often complain about this, and the author offers practical advice on how to get it done right!”

—**Steffan Surdek**, co-author of *A Practical Guide to Distributed Scrum*

“*Executable Specifications with Scrum* doesn’t shine through its depth but its breadth. This compendium of proven agile practices describes an overarching process spike touching important aspects of product development in a cohesive way. In this compact book, Mario Cardinal clearly explains how he achieves a validated value stream by applying agile practices around executable specifications.”

—**Ralph Jocham**, Founder of agile consulting company effective agile. and  
Europe’s first Professional Scrum Master Trainer for Scrum.org

“Cardinal provides deep insights into techniques and practices that drive effective agile teams. As a practitioner of the craft Cardinal describes, I now have a written guide to share with those who ask, ‘What is this [ATDD/BDD/TDD/Executable Specification/etc] thing all about?’ Regardless of the name de jour, Cardinal gives us what works.”

—**David Starr**, Senior Program Manager, Microsoft Visual Studio

“Scrum is barely a process, only a framework. It is a tool, and you have to provide many complementary practices to reach true business agility. This book is perfect for teams that are using Scrum and want to learn about or get started with executable specifications.”

—**Vincent Tencé and François Beauregard**, Scrum Trainers at Pyxis Technologies

“This book maps out the important place of specifications in an agile landscape to the benefit of agilists of all roles.”

—**Erik LeBel**, Technology and Development Consultant at Pyxis Technologies

# Executable Specifications with Scrum

*This page intentionally left blank*

---

---

# Executable Specifications with Scrum

A Practical Guide to  
Agile Requirements Discovery

Mario Cardinal

◆ Addison-Wesley

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

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

Visit us on the Web: [informit.com/aw](http://informit.com/aw)

*Library of Congress Control Number: 2013939927*

*Copyright © 2014 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. To obtain permission to use material from this work, please submit a written request to Pearson Education, Inc., Permissions Department, One Lake Street, Upper Saddle River, New Jersey 07458, or you may fax your request to (201) 236-3290.

ISBN-13: 978-0-32-178413-1

ISBN-10: 0-32-178413-8

Text printed in the United States on recycled paper at Courier in Westford, Massachusetts.

First printing, July 2013

**Editor-in-Chief**  
Mark Taub

**Executive Editor**  
Chris Guzikowski

**Senior Development Editor**  
Chris Zahn

**Marketing Manager**  
Stephane Nakib

**Managing Editor**  
Kristy Hart

**Senior Project Editor**  
Lori Lyons

**Copy Editor**  
Apostrophe Editing Services

**Senior Indexer**  
Cheryl Lenser

**Proofreader**  
Paula Lowell

**Editorial Assistant**  
Olivia Basegio

**Cover Designer**  
Chuti Prasertsith

**Senior Compositor**  
Gloria Schurick

*To my four outstanding children:  
Dominic, Lea-Marie, Romane, and Michael.*



*This page intentionally left blank*

# Contents

Preface	xvi
Chapter 1: Solving the Right Problem	1
Chapter 2: Relying on a Stable Foundation	13
Chapter 3: Discovering Through Short Feedback Loops and Stakeholders' Desirements	25
Chapter 4: Expressing Desirements with User Stories	35
Chapter 5: Refining User Stories by Grooming the Product Backlog	45
Chapter 6: Confirming User Stories with Scenarios	73
Chapter 7: Automating Confirmation with Acceptance Tests	97
Chapter 8: Addressing Nonfunctional Requirements	123
Chapter 9: Conclusion	145
Glossary	153
Index	159

# Contents

<b>Preface</b>	<b>xvi</b>
<b>Chapter 1 Solving the Right Problem</b>	<b>1</b>
Distinguishing the Requirements from the Solution	4
Recognizing the Impact of Uncertainty	5
Tackling Uncertainty	7
Summary	10
References	10
<b>Chapter 2 Relying on a Stable Foundation</b>	<b>13</b>
Defining What Will Hardly Change	14
Creating a Healthy Team	14
Requiring the Involvement of All Stakeholders	16
Expressing a Shared Vision	17
Distinguishing a Meaningful Common Goal	20
Identifying a Set of High-Level Features	21
Validating the “Can-Exist” Assumption	22
Summary	23
References	23
<b>Chapter 3 Discovering Through Short Feedback Loops and Stakeholders’ Desirements</b>	<b>25</b>
Applying the Trial-and-Error Method	25
Using Short Feedback Loops	29
Targeting Feedback Along the Expected Benefits	31
Focusing on the Stakeholders’ Desirements	31
Summary	34
References	34
<b>Chapter 4 Expressing Desirements with User Stories</b>	<b>35</b>
Describing Desirements by Using User Stories	35
Discovering Desirements by Exploring Roles and Benefits	38
Establishing a Ubiquitous Language	40
Recording Desirements by Using a Product Backlog	41

Summary	43
References	44
<b>Chapter 5 Refining User Stories by Grooming the Product Backlog</b>	<b>45</b>
Managing the Product Backlog	46
Collaborating to Groom the Product Backlog	48
Ranking User Stories with a Dot Voting Method	49
Illustrating User Stories with Storyboards	52
Sizing User Stories Using Comparison	56
Splitting User Stories Along Business Values	60
Tracking User Stories with a Collaboration Board	62
Delivering a Coherent Set of User Stories	68
Planning Work with User Stories	70
Summary	71
References	72
<b>Chapter 6 Confirming User Stories with Scenarios</b>	<b>73</b>
Scripting User Stories with Scenarios	74
Expressing Scenarios with Formality	76
Scripting Scenarios Using the FIT Tabular Format	77
Scripting Scenarios Using Given-When-Then Syntax	79
Choosing Between FIT Tabular Format or Given-When-Then Syntax	80
Formalizing a Ubiquitous Language	81
Splitting Scenarios into Commands or Queries	83
Confirming Collaboratively in a Two-Step Process	85
Removing Technical Considerations from Scenarios	89
Evolving Scenarios from Sprint to Sprint	91
Organizing Scenarios by Feature	92
Documenting Scenarios by Feature	93
Avoiding Duplication and Merging Conflicts	94
Summary	95
References	96
<b>Chapter 7 Automating Confirmation with Acceptance Tests</b>	<b>97</b>
Evolving Scenarios into Acceptance Tests	98
Automating Scenarios Using the Red-Green-Refactor Cycle	101

Translating the Scenario into an Acceptance Test	104
Transposing Using an Internal DSL	104
Creating a Test	107
Coding the DSL into the Newly Created Test	108
Connecting the Newly Created Test with the Interface	110
Exercising the Interface	112
Chaining Context Between the Steps of the Scenario	113
Making the Test Fail	114
Implementing the Interface	115
Replacing Unit Testing with Context-Specification	
Testing	116
Making the Test Pass	117
Evolving the Acceptance Test	117
Running Acceptance Tests Side-by-Side with Continuous	
Integration	118
Enhancing Scenarios with Test Results	119
Summary	121
References	122
<b>Chapter 8 Addressing Nonfunctional Requirements</b>	<b>123</b>
Improving External Quality Using Restrictions	125
Translating Nonfunctional Requirements into	
Restrictions	127
Reducing the Functional Scope to a Single Scenario	129
Setting Measurable Quality Objectives	131
Testing Restrictions with Proven Practices	135
Ensuring Internal Quality Using Sound Engineering Practices	137
Improving Software Construction with Explicit Practices	137
Mastering Practices with Collaborative Construction	140
Summary	142
References	143
<b>Chapter 9 Conclusion</b>	<b>145</b>
Recapitulating the Book	146
Summarizing the Process	148
Drawing Attention to Individual Roles	149
<b>Glossary</b>	<b>153</b>
<b>Index</b>	<b>159</b>

# Figure List

Figure 1.1: Usage of features in a typical system	3
Figure 1.2: Uncertainty diagram	5
Figure 1.3: Traditional engineering and uncertainty	7
Figure 1.4: R&D and uncertainty	8
Figure 1.5: Agile and uncertainty	9
Figure 3.1: Sprint	29
Figure 4.1: Product backlog is the list of desirements sorted by importance	41
Figure 4.2: Product backlog is like an iceberg	43
Figure 5.1: Grooming the backlog	49
Figure 5.2: An example of a storyboard for an animated film	53
Figure 5.3: An example of a paper prototype	54
Figure 5.4: A computerized low-fidelity storyboard	55
Figure 5.5: Deck of Fibonacci cards	59
Figure 5.6: The backlog grooming workflow	62
Figure 5.7: A collaboration board is a two-dimensional grid	63
Figure 5.8: A task board is a well-known example of a collaboration board	64
Figure 5.9: A collaboration board with no signals	65
Figure 5.10: A collaboration board with “Done” signals	65

Figure 5.11: A collaboration board with “Ready” signals	66
Figure 5.12: A collaboration sticker has nine display areas	66
Figure 5.13: A collaboration sticker representing a user story	67
Figure 5.14: Planning sprints with story mapping	69
Figure 6.1: A state transition	75
Figure 6.2: A FIT table	77
Figure 6.3: A FIT table is a state transition	78
Figure 6.4: A FIT table is a test	78
Figure 6.5: Describing concepts using precondition and consequence states	82
Figure 6.6: Formalizing a ubiquitous language	83
Figure 6.7: Differentiating between command and query	84
Figure 6.8: Querying a list of items	84
Figure 6.9: Confirming collaboratively using a two-step process	86
Figure 6.10: Specifying the scenarios	87
Figure 6.11: Scenarios work at many levels	90
Figure 6.12: Organizing the scenarios by feature	92
Figure 6.13: A scenario validates only one feature	93
Figure 6.14: Generating the specification with computer-based tools	94
Figure 7.1: The acceptance test is a copy of a scenario in a format suitable for execution on a computer	98

Figure 7.2: Turning scenarios into acceptance tests using a three-stage process	102
Figure 7.3: Turning scenarios into acceptance tests is how an increment is built	103
Figure 7.4: Coding the internal DSL inside the SpecFlow automation framework	108
Figure 7.5: Coding the internal DSL inside the StoryQ automation framework	109
Figure 7.6: Connecting the steps with the interface	112
Figure 7.7: Chaining context between the steps	114
Figure 7.8: Implementing the interface using TDD	116
Figure 7.9: Visualizing specifications conformance by identifying failing tests	120
Figure 7.10: Tracing work completeness by measuring passing tests	121
Figure 8.1: Imposing restrictions using a concrete and specific functional scope	129
Figure 8.2: Addressing a restriction side by side with its linked functional scope	130
Figure 8.3: Avoid linking restrictions with a user story	130
Figure 8.4: Linking restrictions with scenarios is a process repeated story after story	131
Figure 8.5: Enhancing a scenario with a restriction	132
Figure 8.6: Querying a list of items in a scenario	133
Figure 9.1: Summarizing the process	149



# Preface

There is a wide range of books that have been written about specifications. Unfortunately, most of them are not useful for software development teams. These books rely on traditional engineering practices. They assume requirements are known upfront and, once specified, will not change for the duration of the project. And if changes happen, they presume they will be minor, so they could be tracked with a change management process. They promote a sequential process starting with a distinct requirements phase that delivers a detailed requirements specification before starting to design and build the product.

---

## Goal of This Book

It is my belief that traditional engineering practices are not suitable for software development. Central to the process of software specification is a high level of uncertainty, which is not the case with traditional engineering. Fortunately, with the growth of the agile community in the past decade, a body of knowledge more suited to the reality of software development has emerged. Many books explaining agility have become must-read books for anyone interested in software development. A large majority of them contain at least a chapter or two on requirements, some almost totally dedicated to this topic. Because I believe these texts are important, I will include citations from them and reference them throughout this book.

I wrote this book to add to this body of knowledge. It is a compendium of the agile practices related to executable specifications. Executable specifications enable us to easily test the behavior of the software against the requirements. Throughout this book, I will explain how you can specify software when prerequisites are not clearly defined and when requirements are both difficult to grasp and constantly evolving. Software development practitioners will learn how to trawl requirements incrementally, step-by-step, using a vision-centric and an emergent iterative practice. They will also learn how to specify as you go while writing small chunks of requirements.

This book aims to explain the technical mechanisms needed to obtain the benefits of executable specifications. It not only provides a sound case for iterative discovery of requirements, it also goes one step further by teaching you how to connect the specifications with the software under construction. This whole process leads to the building of executable specifications.

It is important to recognize that even with the best intentions you cannot force agreement upon stakeholders. The following African proverb explains this succinctly: “You can’t make grass grow faster by pulling on it.” When knowledge is incomplete and needs are constantly changing, we cannot rely on approaches based on traditional engineering. Instead, it is critical that you emphasize empirical techniques based on the iterative discovery of the requirements. The objective sought is not only to solve the problem right, but also to solve the right problem—this is the paramount challenge of software construction.

This book is unique in that it teaches you how to connect requirements and architecture using executable specifications. You learn how to specify requirements as well as how to automate the requirements verification with a Scrum framework. As a result of reading this book, you can select a tool and start using executable specifications in future agile projects. Here are five advantages to reading this book:

- You can understand how the work of business analysts changes when transitioning from traditional to agile practices.
- You learn how to groom emergent requirements within the Scrum framework.
- You get insight about storyboarding and paper prototyping to improve conversations with stakeholders.
- You discover how to build an emergent design while ensuring implementation correctness at all times
- You can understand that software architects who are adopting agile practices are designing incrementally and concurrently with software development.

## Who Should Read This Book?

Readers of this book have already adopted the Scrum framework or are transitioning to agile practices. They understand the fundamentals of agility but are unfamiliar with executable specifications. They want to understand why the executable specifications are useful and most important how to start with this new practice.

With the massive adoption of Scrum framework, the next major challenge facing agile teams is to integrate business analysts and architects as active teammates. Anyone who is a Scrum master, manager or decision maker who faces this challenge should read this book. In addition, all team members involved in agile projects will benefit from this book. It goes without saying that business analysts and software architects will be happy to find a book that directly addresses their concerns.

Advanced or expert agilists will be interested in the book's concise overview of executable specifications. They could use this book to successfully guide their teammates down this path. In addition, the terminology used throughout the book can help leaders to communicate effectively with their peers.

---

## Road Map for This Book

Executable specifications require a change in mindset. This book focuses on this issue. Executable specifications help reduce the gap between what stakeholders want the software to do (the “What”), and what the software really does (the “How”). Executable specifications address requirements in a way that makes it easy for the development team to verify the software against the specifications and this as often as requirement changes occur.

To facilitate this change in mindset, this book offers a unique approach to the process that spans nine chapters:

- **Chapter 1: Solving the Right Problem**

This chapter explains the need to respond efficiently to the constantly changing requirements using iterative discovery and executable specifications.

- **Chapter 2: Relying on a Stable Foundation**

This chapter explains how to identify what will hardly change: the core certainties on which the team should rely. Those certainties are not requirements. They are high-level guardrails that ensure a solution can be built. They create a stable foundation to ensure that an iterative requirements discovery is possible.

- **Chapter 3: Discovering Through Short Feedback Loops and Stakeholders' Desirements**

This chapter shows that to tackle uncertainties, teams must discover stakeholders' desires and requirements (desirements) through short feedback loops.

- **Chapter 4: Expressing Desirements with User Stories**

This chapter teaches you how to express desirements with user stories and how to record them using the product backlog.

- **Chapter 5: Refining User Stories by Grooming the Product Backlog**

This chapter explains how to groom the product backlog so that you can plan sprints that can increase the likelihood of success of the feedback loops.

- **Chapter 6: Confirming User Stories with Scenarios**

This chapter demonstrates how to confirm user stories by scripting behaviors with scenarios.

- **Chapter 7: Automating Confirmation with Tests**

This chapter explains how to turn scenarios into automated tests so that you can easily confirm the expected behavior of the software against the evolving specifications.

- **Chapter 8: Addressing Nonfunctional Requirements**

This chapter teaches you how to ensure quality software by specifying nonfunctional requirements.

- **Chapter 9: Conclusion**

This last chapter summarizes the key elements of the book.