

## About this book...

This book is a must read for anyone involved in creating a product; management, testers and developers. Use it to change the way you think projects should be run and close those communication gaps.

— Toby Henderson

Have you ever wanted a better way to communicate, clarify and satisfy business requirements? Wouldn't it be great if those requirements evolved along with the software, always consistent and clear? And those requirements helped drive development so that we knew when we were done? With clarity, Gojko describes an elegance and effective way of achieving this with the whole team: Inventing, thinking and communicating with specific, insightful examples that also serve as acceptance tests.

— Rick Mugridge, [www.rimuresearch.com](http://www.rimuresearch.com), and lead author of 'Fit for Developing Software', Prentice Hall, 2005.

Effective software development is all about good communication and this book explains the toolkit that allows us to do this effectively. Worth reading whatever your role is.

— Mark Needham, <http://www.markneedham.com/blog/>

Whether you're new to testing, new to agile, or an old pro at one or both, you'll experience "aha" moments that will inspire your team as you read. This book will challenge some of your preconceived notions and make you think. It paves the way for people in different roles, such as business analysts, QA engineers and developers, to adapt to a more productive agile approach. From practical ways to improve communication with customers, to helpful examples of useful test tools, this book is a major addition to our agile testing knowledge base.

— Lisa Crispin, co-author, 'Agile Testing: A Practical Guide for Testers and Agile Teams', Addison-Wesley Professional 2009

Gojko addresses an underrated point: that Test-Driven Requirements, or Executable Requirements, are not about tools, automated tests, or even professionalism. They are about communication. I wish each of my colleagues and clients had a copy of his book, and maybe that fact will be made just a little bit clearer to them.

— Eric Lefevre-Ardant, Agile Coach and Developer,  
<http://ericlefevre.net/>

I wish that the book had been available a few years ago when the company I was at (and myself) were trying out agile. Could have been a lot easier and more successful if we'd read it.

— Philip Kirkham

If you've tried agile acceptance testing you'll know that as well as being a really exciting it's also incredibly difficult. Luckily we now have a book that helps guide us through the many tricky choices that we face, practical and pragmatic advice that even the most experienced agile developer should be aware of.

— Colin Jack, Senior Software Developer, FNZ

As a tester, I welcome any opportunity to increase shared understanding of requirements and expectations - our team will be relying on this book to guide us as we begin our journey with agile acceptance testing.

— Marisa Seal

You would be at least 6 months ahead of the game in Agile QA by just reading Gojko's book.

— Gennady Kaganer, QA Manager at Standard and Poor's

Gojko applies his experience to the practice of producing software that is useful to end users. This is an important work in extending the test-driven specification of software beyond individual units and into the sum of the parts.

— Bob Clancy, <http://www.agiletester.net>

Bridging the Communication Gap will not only bring you up-to-date with the latest thinking about agile acceptance testing, but also guide you as you put the ideas into practice. This book is packed with insights from Adzic's experience in the field.

— David Peterson, creator of the Concordion acceptance-testing framework

I'm convinced that the practice of agile acceptance testing, done properly, can make a dramatic improvement to both the communication with the customer and the quality of the final product. This book is a solid introduction to the subject, and represents the first attempt I've seen to survey the practice as a whole without focusing on a single tool or technology.

— Geoff Bache

# Bridging the Communication Gap

Specification by Example and Agile  
Acceptance Testing

Gojko Adzic

# Bridging the Communication Gap: Specification by Example and Agile Acceptance Testing

Gojko Adzic

Copy-editor: Marjory Bisset

Cover design: Boris Marcetic

Published 5 Jan 2009

Copyright © 2009 Neuri Limited

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where these 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 has taken care in the preparation of this book, but makes no expressed or implied warranty of any kind and assumes 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.

All rights reserved. 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:

Neuri Limited  
25 Southampton Buildings  
London WC2A 1AL  
United Kingdom

You can also contact us by e-mail: [contact@neuri.com](mailto:contact@neuri.com)



*Register your book online*

Visit <http://www.acceptancetesting.info> and register your book online to get free PDF updates and notifications about corrections or future editions of this book.

ISBN: 978-0-9556836-1-9

---

# Table of Contents

|  |       |
|--|-------|
| About the PDF edition .....                                  | ix    |
| Acknowledgements .....                                       | xi    |
| About the author .....                                       | xiii  |
| Introduction .....   | xv    |
| Why should you care? .....                                   | xvi   |
| Who is this book for? .....                                  | xvii  |
| What will you get out of this book? .....                    | xviii |
| What's inside? .....   | xix   |
| Giving credit where credit is due .....                      | xxi   |
| How this book is organised .....                             | xxii  |
| I. The Communication Gap .....                               | 1     |
| 1. The next bottleneck in software projects .....            | 3     |
| The telephone game .....                                     | 5     |
| Imperative requirements are very easy to misunderstand ..... | 8     |
| Are obvious things really obvious? .....                     | 9     |
| A small misunderstanding can cost a lot of money .....       | 14    |
| Fulfilling specifications does not guarantee success .....   | 15    |
| Requirements are often already a solution .....              | 18    |
| Cognitive diversity is very important .....                  | 20    |
| Breaking The Spirit of Kansas .....                          | 21    |
| 2. Finding ways to communicate better .....                  | 25    |
| Challenging requirements .....                               | 25    |
| We really communicate with examples .....                    | 26    |
| Working together, we find better solutions .....             | 28    |
| Communicating intent .....                                   | 29    |
| Agile acceptance testing in a nutshell .....                 | 31    |
| So what does testing have to do with this? .....             | 34    |
| Better names .....   | 38    |
| II. Building and Maintaining a Shared Understanding .....    | 41    |
| 3. Specifying with examples .....                            | 43    |
| How do you brush your teeth? .....                           | 43    |
| A practical example .....                                    | 45    |

|   |     |
|---|-----|
| Realistic examples make us think harder .....       | 48  |
| Identifying important examples .....                | 50  |
| Dealing with processing workflows .....             | 54  |
| Working with business workflows .....               | 57  |
| 4. Specification workshops .....                    | 59  |
| Running a workshop .....                            | 60  |
| Workshop output .....                               | 63  |
| Sharing domain knowledge .....                      | 64  |
| What specification workshops are not .....          | 68  |
| How to keep the workshop focused .....              | 70  |
| Dealing with open issues .....                      | 72  |
| 5. Choosing acceptance criteria .....               | 75  |
| Acceptance tests are the specification .....        | 75  |
| Choosing the right set of examples .....            | 76  |
| Distilling the specifications .....                 | 78  |
| Specifying business workflows .....                 | 80  |
| Converting examples to tests .....                  | 82  |
| Automating tests .....                              | 86  |
| Dealing with user interfaces .....                  | 91  |
| Who should write acceptance tests? .....            | 95  |
| What acceptance tests are not .....                 | 97  |
| 6. Focusing development .....                       | 103 |
| Building software to satisfy acceptance tests ..... | 103 |
| Suggesting new features .....                       | 105 |
| Designing software based on acceptance tests .....  | 106 |
| Developing the glue between code and tests .....    | 110 |
| Use the project language consistently .....         | 111 |
| Unit tests vs acceptance tests .....                | 113 |
| Running acceptance tests .....                      | 115 |
| Running manual tests .....                          | 117 |
| 7. Facilitating change .....                        | 119 |
| A live specification .....                          | 119 |
| Keeping it live .....                               | 120 |
| Introducing changes .....                           | 123 |
| Solving problems .....                              | 124 |
| Keeping the software flexible .....                 | 125 |
| Keeping the specification flexible .....            | 126 |
| III. Implementing agile acceptance testing .....    | 139 |

---

|   |     |
|---|-----|
| 8. Starting with agile acceptance testing .....                               | 141 |
| Agile acceptance testing in the context of the develop-<br>ment process ..... | 141 |
| Fitting into iterations .....   | 144 |
| Igniting the spark .....  | 148 |
| Don't mention testing .....   | 150 |
| Adopting in phases .....  | 151 |
| Choose a facilitator for the workshops .....                                  | 152 |
| Hire a mentor .....   | 154 |
| Avoid the cargo cult mentality .....  | 155 |
| 9. Planning with user stories .....   | 159 |
| User stories in a nutshell .....  | 160 |
| How are stories different from more traditional<br>methods? .....             | 161 |
| Benefits of user stories .....  | 162 |
| Three Cs of user stories .....  | 164 |
| Project planning .....  | 165 |
| User stories and acceptance tests .....                                       | 169 |
| 10. Tools of today .....  | 173 |
| FIT .....   | 173 |
| Concordion .....  | 179 |
| JBehave .....   | 181 |
| TextTest .....  | 183 |
| Selenium .....  | 187 |
| 11. Tools of tomorrow .....   | 193 |
| Domain-specific languages .....   | 193 |
| Different user interfaces for different roles .....                           | 195 |
| Propagating the effects of changes .....                                      | 196 |
| Direct domain mapping .....   | 198 |
| Better editors .....  | 199 |
| Better test organisation .....  | 200 |
| Visual workflow descriptions .....  | 201 |
| IV. Effects on all of us .....  | 203 |
| 12. Effects on business analysts .....  | 205 |
| Benefits for business analysts .....  | 206 |
| Challenges for business analysts .....  | 209 |
| 13. Effects on testers .....  | 217 |
| The new role of the tester .....  | 217 |

|   |     |
|---|-----|
| Benefits for testers .....                | 218 |
| Challenges for testers .....              | 222 |
| 14. Effects on developers .....           | 229 |
| New responsibilities for developers ..... | 229 |
| Benefits for developers .....             | 230 |
| Challenges for developers .....           | 232 |
| A. Resources .....                        | 239 |
| Online resources .....                    | 241 |
| Talks and videos .....                    | 241 |
| Presentations .....                       | 242 |
| Articles .....                            | 242 |
| Tools .....                               | 243 |
| Mailing Lists .....                       | 243 |
| Index .....                               | 245 |

---

# About the PDF edition

---

If you are reading this book in a print edition, you should know that there is a PDF version available, and you might find it useful to get an electronic copy as well. I've decided to make this book available as a PDF so that the readers can get the most out of it. Judging from the sales of my previous book, many people today prefer reading technical texts in an electronic form so that they can search through the content, copy and paste text and archive the material more easily.

If you are reading this as a PDF, please note that this material is copyrighted and it is not for free distribution. I have no intention of chasing people for redistributing the book, but if you have received this PDF from someone and you found it useful, consider actually purchasing your own copy. The PDF edition is relatively cheap and your purchase will support similar efforts in the future.

Buying a PDF or a printed version entitles you to free PDF updates. Register your copy and you'll receive a free PDF when the book is updated in the future. For more information on the PDF version of this book, and to register your copy, see:

<http://www.acceptancetesting.info>

---

---

# Acknowledgements

---

This book is a result of a small independent publishing effort, and as such would not be possible without the help of many people.

I'd like to thank Antony Marcano, Bob Clancy, Colin Jack, David Peterson, David Vydra, Eric Lefevre-Ardant, Gennady Kaganer, Geoff Bache, Jennitta Andrea, Lisa Crispin, Marisa Seal, Mark Needham, Melissa Tan, Mike Scott, Phil Kirkham and Rick Mugridge for all the excellent suggestions, helping me keep this book focused and providing insight into their views and experiences. Without you, this book just would not be possible.

Marjory Bisset from Pearl Words again did a great job of copy-editing this book and ensuring that readers have a much more enjoyable experience with it.

Finally, I'd like to thank Boris Marcetic from Popular for designing the covers.

---

---

# About the author

---

Gojko Adzic runs Neuri Ltd, a UK-based consultancy that helps companies build better software by introducing agile practices and tools and improving communication between software teams, stakeholders and clients. His programming story so far includes equity and energy trading, mobile positioning, e-commerce, betting and gaming and complex configuration management.

Gojko is the author of several popular printed and online guides on acceptance testing, including *Test Driven .NET Development with Fitnesse* and *Getting Fit with .NET*, and more than 200 articles about programming, operating systems, the Internet and new technologies published in various online and print magazines. He is the primary contributor to the DbFIT database testing library which is used by banks, insurance companies and bookmakers worldwide.

To get in touch, write to [gojko@neuri.com](mailto:gojko@neuri.com) or visit <http://gojko.net>.

---

---

# Introduction

---

I am getting more and more convinced every day that communication is, in fact, what makes or breaks software projects. Programming tools, practices and methods are definitely important, but if the communication fails then the rest is just painting the corpse. Complex projects simply have no chance of success without effective communication.

This is a book about improving communication between customers, business analysts, developers and testers on software projects, especially by using specification by example and agile acceptance testing. Although these two practices are not yet popular, I consider them as key emerging software development practices because they can significantly improve the chances of success of a software project. (At the same time, agile acceptance testing is one of the worst named practices ever. For the time being, just forget that it has the word testing in the name.) Agile acceptance testing and specification by example essentially help us to close the communication gap between different participants in a software project, ensure that they speak the same language and build a truly shared and consistent understanding of the domain. This leads to better specifications – the participants flush out incorrect assumptions and discover functional gaps before the development starts and build software that is genuinely fit for purpose.

Ward Cunningham and Ken Auer used the basic ideas behind agile acceptance testing to nail down what their users wanted in 1999.<sup>1</sup> Almost a decade later, the practice is still used only by a small group of early adopters. However, it has matured considerably. Judging from recent conferences, magazine articles and blog posts, it seems to me that interest is growing and the time is right for a wider group to learn about and adopt it. You probably agree, which is why you picked up this book.

---

<sup>1</sup> See *Don't just break software, make software*[1]

# Why should you care?

Agile acceptance testing and specification by example help business people, software developers and testers communicate better and understand each other. Each of these groups has different needs and problems in a software project, so improved communication has different benefits for them. In this section, I briefly list the the benefits agile acceptance testing and specification by example will deliver for each group. The rest of this book explains how to achieve these benefits.

These are the most important benefits for product owners, business analysts and project managers:

- Developers will actually read the specifications that you write.
- You will be sure that developers and testers understand the specifications correctly.
- You will be sure that they do not skip parts of the specifications.
- You can track development progress easily.
- You can easily identify conflicts in business rules and requirements caused by later change requests.
- You'll save time on acceptance and smoke testing.

From a developer's perspective, these are the most important advantages of agile acceptance testing and specification by example:

- Most functional gaps and inconsistencies in the requirements and specifications will be flushed out before the development starts.
- You will be sure that business analysts actually understand special cases that you want to discuss with them.
- You will have automated tests as targets to help you focus the development.
- It will be easier to share, hand over and take over code.

From a tester's perspective, these are the most important benefits of agile acceptance testing and specification by example:

- You can influence the development process and stop developers from making the same mistakes over and over.
- You will have a much better understanding of the domain.
- You'll delegate a lot of dull work to developers, who will collaborate with you on automating the verifications.
- You can build in quality from the start by raising concerns about possible problems before the development starts.
- You'll be able to verify business rules with a touch of a button.
- You will have a lot more time for exploratory testing.
- You will be able to build better relationships with developers and business people and get their respect.

## Who is this book for?

This book is primarily intended for product owners, business analysts, software developers and testers who want to learn about how to implement agile acceptance testing and how to improve communication on their projects. It should also prove to be interesting to project managers working on agile software projects or in the process of migrating to an agile methodology. I wrote this book both for people working in implementation teams and for people working on the customer side, and for teams that practice agile development as well as those who are migrating to an agile process. I use the term *implementation team* in this book as a name for the entire group of people working on a project, from business analysts to developers, testers and anyone else involved.

Although agile teams generally strive for less role-naming than those practising traditional processes, I use the role names *product owner*, *business analyst*, *software developer* and *tester* in this book and discuss how each of these roles is affected by agile acceptance testing and how they participate in writing specifications by example. Companies in transition with teams coming from a more traditional background will have all these roles clearly defined, sometimes isolated into different departments, buildings or even countries. Most agile programming literature is written by software developers for software

developers, leaving testers and business analysts to fend for themselves and try to work out what they should be doing. Agile acceptance testing requires active participation from all project implementation team members, including business analysts and testers, not just developers. This is why I want to use these roles and names to emphasise their involvement in the process. Contrary to common belief, it is not unusual for larger agile projects to have dedicated business analysts and testers. ThoughtWorks is generally accepted as one of the leading agile software development companies and their process has these roles clearly defined and uses these names.<sup>2</sup>

The roles of business analysts and testers on agile projects differ somewhat from their roles on projects that use more traditional formal methods, but this does not mean that the people behind these roles do not exist. Roles are blurred a bit, but specialists still have to focus on particular areas. If you are a business analyst or a tester, this book should help you define more accurately what your job is and how to make a more useful contribution to building software the agile way.

## What will you get out of this book?

Primarily, this book will help you to get started and implement agile acceptance testing and specification by example effectively. It explains the principles and ideas behind the practice and helps you put it into the wider context of your development process.

As with any other new and relatively radical idea, learning how to apply it effectively in your own role is only part of the challenge. Agile acceptance testing requires active participation and affects the jobs of programmers, business analysts, testers and to some extent project managers. Because of this, it also raises a lot of concerns and it is unfortunately subject to many misconceptions. These problems are typically fuelled by partial information, misunderstandings and fears of change. One of my primary goals with this book is to dispel these

---

<sup>2</sup><http://download.microsoft.com/documents/uk/msdn/architecture/architectinsight/2007/Life-cycle/LIF02-The-Agile-Template-for-VSTS.ppt>.

misconceptions and address fears and issues that people often have about agile acceptance testing.

It is also my intention with this book to challenge some established ways of thinking in the industry. Agile acceptance testing breaks down traditional boundaries around testing, requirements and specification processes in a way that significantly improves communication on a project. Specification by example is an approach to the writing of specifications and requirements radically different from the established industry process. I will explain this in a lot more detail throughout the book, but please note that if you come from a more traditional software process background, you may need to put aside the stereotypes that you are familiar with in order to grasp the ideas and gain the full benefits.

This book will help you to discover how agile acceptance testing affects your work whether you are a programmer, business analyst or a tester. It will help you gain an understanding of this technique and offer ideas on how to convince other team members and stakeholders to use it. I hope that this book takes into account the various perspectives of these different roles. It is intentionally not too technical or suited only for programmers, because agile acceptance testing is not a programming technique: it is a communication technique that brings people involved in a software project closer. I consider myself primarily a programmer and my personal biases and experiences will surely show. I have, however, found a much broader perspective on agile acceptance testing as a result of working with different teams, consulting, mentoring and giving public talks. Discussing fears such as losing jobs with testers over coffee, participating in requirements gathering, working as a product owner and even as a business domain expert on several projects has hopefully given me the understanding necessary to represent the needs and views of the other roles.

## What's inside?

I am very proud of the fact that I have helped several companies adopt agile acceptance testing and specification by example and deliver

successful software over the last few years. This book is a summary of my experiences, learnings and ideas from this journey. It is based on projects that are now happily running in production and a series of public and private seminars, talks and workshops that I organised during this period.

This book contains a description of the process that I use today to bridge the communication gap between business people and software implementation teams. More importantly, it describes ideas, principles and practices behind the process and their effects on people participating in software projects.

This book is not a detailed user manual for any acceptance testing tool. Although I will describe briefly some of the most popular tools in Chapter 10, I intentionally want to keep this book relatively non-technical and absolutely not tool or technology specific. Often I hear from managers that their teams looked into this or that tool and that they did not like it a bit, so they rejected the whole idea of agile acceptance testing. One of the main messages I want to convey with this book is that the biggest benefit of agile acceptance testing is improved communication and mutual understanding, not test automation. There is too much focus on tools today and in my opinion this is quite wrong.

While you are reading this book, it's important to focus on the ideas and principles. The process and the tools described in the book are there just to assist. If you do not like a particular tool, find a better one or automate things yourself in a way that makes sense for your team. If you do not like some part of the process, adjust it or replace it with something that makes more sense for your team. Once you understand the underlying ideas, principles and practices, applying them in your environment should be easy. Use the process described in this book just as a guide, not as a prescription.

This book also does not have all the answers about agile acceptance testing and specification by example. These two practices are currently gaining a lot of momentum and generating a lot of new ideas. Because there is very little published material on them at the moment, you

will see many more references to conference talks, blog posts and online video clips than to books or articles. Apart from the practices as I use them today, this book also describes some promising ideas that I want to try out in the near future. It even contains some interesting views of other people that I do not agree with completely. Some of them might prove to be interesting in the future with better tools or be applicable in a different environment. In Chapter 11 I speculate how the tools for agile acceptance testing tools might evolve. Blogs, mailing lists and sites listed in Appendix A will help you continue the journey and keep up-to-date with new advances.

## Giving credit where credit is due

I make no claim and take no credit for inventing any of the ideas or techniques that are described in this book. The techniques I describe are based on ideas I have picked up on the way from books, conferences, blogs, discussion groups and colleagues. This book is an attempt to collect different techniques that have emerged in the community, show how they relate to each other and organise them in a one coherent practice.

Agile acceptance testing is an evolution of the idea of *customer tests* proposed by Kent Beck in which automated tests are used to verify what the customers expect out of a software system. It also draws on ideas developed by many others. Gerald Weinberg and Donald Gause used the *black-box requirements testing* process to redefine the role of tests in a project and relate them directly to requirements. Brian Marick has done great work on bringing tests closer to the business by clearly defining *business-facing tests*. He is also one of the key promoters of the idea of *specification by example*, where realistic examples are used instead of abstract requirements. Dan North's ideas of *behaviour-driven development* bring acceptance testing closer to business people. Ron Jeffries described the *card-conversation-confirmation* concept nailing down the basic ideas behind specification workshops and linking acceptance tests to user stories. Joshua Kerievsky's *storytest-driven development* expands on this to fit acceptance testing into agile planning techniques. Jim Shore defined

the *describe-demonstrate-develop* process as a way to use acceptance tests iteratively to focus development. Lisa Crispin and Antony Marcano were instrumental to promoting and improving *agile testing* practices, tearing down the walls between testers and developers and business people. Eric Evans came up with the idea of *ubiquitous language*, used to avoid confusion caused by different jargons on a project.

These ideas are supplemented by the practices and knowledge embodied in Ward Cunningham's tool FIT and Object Mentor's tool FitNesse, along with many other similar tools and extension libraries, especially Rick Mugridge's FitLibrary and the learning that came out of using these tools in commercial projects. David Peterson's focus on *specifications over scripting* with Concordion is instrumental to getting the most out of agile acceptance testing. A lot of the underlying principles come from *test-driven development*, which in turn borrows a lot of ideas from the *zero quality control* idea of Shigeo Shingo and the *Toyota Production System* of Taichi Ono along with their counterparts in the software world, the *lean software development* ideas of Mary and Tom Poppendieck. Effective application of agile acceptance testing depends on various practices such as short iterations from Scrum, Extreme Programming and other agile processes and the work on *user stories* by Mike Cohn and others. Some ideas are even borrowed from Prussian military tactics from the nineteenth century and their current implementation in the US Army.

Agile acceptance testing is more than the sum of all these ideas and practices. It is an approach to software development characterised by a focus on communication and the creation of an understanding of the domain shared by all implementation team members and customers.

## How this book is organised

In Part I, I present causes of the communication problems in software projects, explain why the traditional model of building requirements and specifications does not work and how other agile programming

practices do not really solve the problem but only provide a work-around. Then I introduce agile acceptance testing as the solution to these problems.

In Part II, I introduce the techniques and principles of agile acceptance testing and explain how they work together to help us facilitate communication and build better software.

In Part III, I talk about implementing agile acceptance testing in organisations. I explain how this practice fits into the wider software development process and how to start using it in your organisation. I also briefly describe current popular tools for agile acceptance testing and discuss what can we expect from future tools. This part also includes a chapter on user stories, another agile technique that makes the implementation of agile acceptance testing much easier.

In Part IV, I deal with the human side of this practice, explaining how it affects our jobs and the way we work. I analyse the effects on business analysts, testers and developers. The chapter on the effects on business analysts is also applicable to customers or other business people involved in software projects. In this part we also revisit the benefits listed in section *Why should you care?* on page xvi and see how the principles and practices described this book deliver them.

---

# Part I. The Communication Gap

Effective communication is the key to successful software projects. In this part, we set the stage by looking at common communication problems on software projects, analysing what happens when communication is impeded and how we can improve the flow of information.