the art of UNIT TESTING

with Examples in .NET

MANNING

ROY OSHEROVE

Early Praise for *The Art of Unit Testing*

"*The Art of Unit Testing* provides a thorough, incremental introduction to writing tests as part of the programming process. Programmers looking to write their first test will find easy-to-follow instructions, while those who have been testing for a while will find ideas for refining their technique"
—Kent Beck, Three Rivers Institute

"Beautifully crafted, detailed unit testing masterpiece. Bravo, Bravo, Bravo!"
—Mohammad Azam, Microsoft MVP, HighOnCoding

"This book tells all the truth about unit testing, even the unpleasant side of it."
—Franco Lombardo, Molteni Informatica

"Roy knows testing."
—Wendy Friedlander, Agile Solutions

"Unit testing, directly from theory to practice."
—Francesco Goggi, Software Engineer

The Art of Unit Testing

with Examples in .NET

Roy Osherove

MANNING
Greenwich
(74° w. long.)

For online information and ordering of this and other Manning books, please visit www.manning.com. The publisher offers discounts on this book when ordered in quantity. For more information, please contact:

Special Sales Department
Manning Publications Co.
Sound View Court 3B
Greenwich, CT 06830

fax: (609) 877-8256
email: manning@manning.com

Manning Publications Co.
209 Bruce Park Avenue
Greenwich, CT 06830

Development Editor: Nermina Miller
Copyeditor: Andy Carroll
Proofreader: Anna Welles
Typesetter: Marija Tudor
Cover designer: Leslie Haimes

ISBN: 978-1-933988-27-6

Printed in the United States of America
1 2 3 4 5 6 7 8 9 10 – MAL – 14 13 12 11 10 09

To my wife, Tal, and my sons,
Itamar and Aviv—my family

Brief contents

Contents

Foreword

When Roy Osherove told me that he was working on a book about unit testing, I was very happy to hear it. The testing meme has been rising in the industry for years, but there has been a relative dearth of material available about unit testing. When I look at my bookshelf, I see books that are about test-driven development specifically and books about testing in general, but until now there has been no comprehensive reference for unit testing—no book that introduces the topic and guides the reader from first steps to widely accepted best practices. The fact that this is true is stunning. Unit testing isn't a new practice. How did we get to this point?

It's almost a cliché to say that we work in a very young industry, but it's true. Mathematicians laid the foundations of our work less than 100 years ago, but we've only had hardware fast enough to exploit their insights for the last 60 years. There was an initial gap between theory and practice in our industry, and we're only now discovering how it has impacted our field.

In the early days, machine cycles were expensive. We ran programs in batches. Programmers had a scheduled time slot, and they had to punch their programs into decks of cards and walk them to the machine room. If your program wasn't right, you lost your time, so you desk-checked your program with pencil and paper, mentally working out all of the scenarios, all of the edge cases. I doubt the notion of automated unit testing was even imaginable. Why use the machine for testing when you could use it to solve the problems it was meant to solve? Scarcity kept us in the dark.

Later, machines became faster and we became intoxicated with interactive computing. We could just type in code and change it on a whim. The

idea of desk-checking code faded away, and we lost some of the discipline of the early years. We knew programming was hard, but that just meant that we had to spend more time at the computer, changing lines and symbols until we found the magical incantation that worked.

We went from scarcity to surplus and missed the middle ground, but now we're regaining it. Automated unit testing marries the discipline of desk-checking with a newfound appreciation for the computer as a development resource. We can write automated tests, in the language we develop in, to check our work—not just once, but as often as we're able to run them. I don't think there is any other practice that's quite as powerful in software development.

As I write this, in 2009, I'm happy to see Roy's book come into print. It's a practical guide that will help you get started and also serve as a great reference as you go about your testing tasks. *The Art of Unit Testing* isn't a book about idealized scenarios. It teaches you how to test code as it exists in the field, how to take advantage of widely used frameworks, and, most importantly, how to write code that's far easier to test.

The Art of Unit Testing is an important title that should have been written years ago, but we weren't ready then. We are ready now. Enjoy.

MICHAEL FEATHERS
SENIOR CONSULTANT
OBJECT MENTOR

Preface

One of the biggest failed projects I worked on had unit tests. Or so I thought. I was leading a group of programmers creating a billing application, and we were doing it in a fully test-driven manner—writing the test, then writing the code, seeing the test fail, making the test pass, refactoring, and starting all over again.

The first few months of the project were great. Things were going well, and we had tests that proved that our code worked. But as time went by, requirements changed. We were forced to change our code to fit those new requirements, and when we did, tests broke and had to be fixed. The code still worked, but the tests we wrote were so brittle that any little change in our code broke them, even though the code was working fine. It became a daunting task to change code in a class or method because we also had to fix all the related unit tests.

Worse yet, some tests became unusable because the people who wrote them left the project and no one knew how to maintain the tests, or what they were testing. The names we gave our unit-testing methods were not clear enough, and we had tests relying on other tests. We ended up throwing out most of the tests less than six months into the project.

The project was a miserable failure because we let the tests we wrote do more harm than good. They took more time to maintain and understand than they saved us in the long run, so we stopped using them. I moved on to other projects, where we did a better job writing our unit tests, and we had some great successes using them, saving huge amounts of debugging and integration time. Ever since that first failed project, I've been compil-

ing best practices for unit tests and using them on subsequent projects. I find a few more best practices with every project I work on.

Understanding how to write unit tests—and how to make them maintainable, readable, and trustworthy—is what this book is about, no matter what language or integrated development environment (IDE) you work with. This book covers the basics of writing a unit test, moves on to the basics of interaction testing, and then introduces best practices for writing, managing, and maintaining unit tests in the real world.

Acknowledgments

A big thank you to Michael Stephens and Nermina Miller at Manning, who were patient with me every step of the long way in writing this book.

Thank you Jim Newkirk, Michael Feathers, Gerard Meszaros, and many others, who provided me with inspiration and the ideas that made this book what it is. And a special thank you to Michael for agreeing to write the foreword to the book.

The following reviewers read the manuscript at various stages during its development. I'd like to thank them for providing valuable feedback: Svetlana Christopher, Wendy Friedlander, Jay Flowers, Jean-Paul S. Boodhoo, Armand du Plessis, James Kovacs, Carlo Bottiglieri, Ken DeLong, Dusty Jewett, Lester Lobo, Alessandro Gallo, Gabor Paller, Eric Raymond, David Laribee, Christian Siegers, Phil Hanna, Josh Cronemeyer, Mark Seemann, Francesco Goggi, Franco Lambardo, Dave Nicolette, and Mohammad Azam. Thanks also to Rod Coffin, who did a technical proofread of the final manuscript shortly before it went to press.

A final word of thanks to the early readers of the book in Manning's Early Access Program for their comments in the online forum. You helped shape the book.

About this book

How to use this book

If you've never written unit tests before, this book is best read from start to finish so you get the full picture. If you already have experience, you should feel comfortable jumping into the chapters as you see fit.

Who should read this book

The book is for anyone who writes code and is interested in learning best practices for unit testing. All the examples are written in C# using Visual Studio 2008, so .NET developers will find the examples particularly useful. But the lessons I teach apply equally to most, if not all, statically typed object-oriented languages (VB.NET, Java, and C++, to name a few). If you're a developer, team lead, QA engineer (who writes code), or novice programmer, this book should suit you well.

Roadmap

The book is divided into four parts.

Part 1 takes you from zero to sixty in writing unit tests. Chapters 1 and 2 cover the basics, such as how to use a testing framework (NUnit), and introduce the basic automated test attributes, such as `[SetUp]` and `[TearDown]`. They also introduce the ideas of asserts, ignoring tests, and state-based testing.

Part 2 discusses advanced techniques for breaking dependencies: mock objects, stubs, mock frameworks, and patterns for refactoring your code to use them. Chapter 3 introduces the idea of stubs and shows how to

manually create and use them. Chapter 4 introduces interaction testing with handwritten mock objects. Chapter 5 merges these two concepts and shows how isolation (mock) frameworks combine these two ideas and allow them to be automated.

Part 3 talks about ways to organize test code, patterns for running and refactoring its structure, and best practices when writing tests. Chapter 6 discusses test hierarchies, how to use test infrastructure APIs, and how to combine tests in the automated build process. Chapter 7 discusses best practices in unit testing for creating maintainable, readable, and trustworthy tests.

Part 4 talks about how to implement change in an organization and how to work on existing code. Chapter 8 discusses problems and solutions you would encounter when trying to introduce unit testing into an organization. It also identifies and answers some questions you might be asked. Chapter 9 talks about introducing unit testing into existing code. It identifies a couple of ways to determine where to begin testing and discusses some tools for testing untestable code.

Finally, there are two appendixes. Appendix A discusses the loaded topic of designing for testability and the other alternatives that exist today. Appendix B has a list of tools you might find useful in your testing efforts.

Code conventions and downloads

You can download the source code for this book from the book's site at www.ArtOfUnitTesting.com, as well as from the publisher's website at www.manning.com/TheArtofUnitTesting.

All source code in listings or in the text is in a `fixed-width font like this` to separate it from ordinary text. In listings, **`bold code`** indicates code that has changed from the previous example, or that will change in the next example.

Code annotations accompany some of the listings, highlighting important concepts. In some cases, numbered bullets link to explanations that follow in the text.

Software requirements

To use the code in this book, you need at least Visual Studio C# Express (which is free), or a more advanced version of it (Professional or Team Suite, for example). You'll also need NUnit (an open source and free framework) and other tools that will be referenced where they're relevant. All the tools mentioned are either free, open source, or have trial versions you can use freely as you read this book.

Author Online

The purchase of *The Art of Unit Testing* includes free access to a private forum run by Manning Publications where you can make comments about the book, ask technical questions, and receive help from the authors and other users. To access and subscribe to the forum, point your browser to www.manning.com/TheArtofUnitTesting. This page provides information on how to get on the forum once you are registered, what kind of help is available, and the rules of conduct in the forum.

Manning's commitment to our readers is to provide a venue where a meaningful dialogue between individual readers and between readers and the author can take place. It's not a commitment to any specific amount of participation on the part of the author, whose contribution to the book's forum remains voluntary (and unpaid). We suggest you try asking him some challenging questions, lest his interest stray!

The Author Online forum and the archives of previous discussions will be accessible from the publisher's website as long as the book is in print.