

The Addison-Wesley Signature Series



GROWING OBJECT-ORIENTED SOFTWARE, GUIDED BY TESTS

STEVE FREEMAN
NAT PRYCE



Praise for *Growing Object-Oriented Software, Guided by Tests*

“The authors of this book have led a revolution in the craft of programming by controlling the environment in which software grows. Their Petri dish is the mock object, and their microscope is the unit test. This book can show you how these tools introduce a repeatability to your work that would be the envy of any scientist.”

—Ward Cunningham

“At last a book, suffused with code, that exposes the deep symbiosis between TDD and OOD. The authors, pioneers in test-driven development, have packed it with principles, practices, heuristics, and (best of all) anecdotes drawn from their decades of professional experience. Every software craftsman will want to pore over the chapters of worked examples and study the advanced testing and design principles. This one’s a keeper.”

—Robert C. Martin

“Design is often discussed in depth, but without empiricism. Testing is often promoted, but within the narrow definition of quality that relates only to the presence or absence of defects. Both of these perspectives are valuable, but each on its own offers little more than the sound of one hand clapping. Steve and Nat bring the two hands together in what deserves—and can best be described as—applause. With clarity, reason, and humour, their tour de force reveals a view of design, testing, code, objects, practice, and process that is compelling, practical, and overflowing with insight.”

—Kevlin Henney, co-author of *Pattern-Oriented Software Architecture*
and *97 Things Every Programmer Should Know*

“Steve and Nat have written a wonderful book that shares their software craftsmanship with the rest of the world. This is a book that should be studied rather than read, and those who invest sufficient time and energy into this effort will be rewarded with superior development skills.”

—David Vydra, publisher, testdriven.com

“This book presents a unique vision of test-driven development. It describes the mature form of an alternative strain of TDD that sprang up in London in the early 2000s, characterized by a totally end-to-end approach and a deep emphasis on the messaging aspect of objects. If you want to be an expert in the state of the art in TDD, you need to understand the ideas in this book.”

—Michael Feathers

“With this book you’ll learn the rhythms, nuances in thinking, and effective programming practices for growing tested, well-designed object-oriented applications from the masters.”

—Rebecca Wirfs-Brock

Growing Object-Oriented Software, Guided by Tests

Table of Contents

Contents

Foreword

Preface

Acknowledgments

About the Authors

Part I: Introduction

Chapter 1: What Is the Point of Test-Driven Development?

Software Development as a Learning Process

Feedback Is the Fundamental Tool

Practices That Support Change

Test-Driven Development in a Nutshell

The Bigger Picture

Testing End-to-End

Levels of Testing

External and Internal Quality

Chapter 2: Test-Driven Development with Objects

A Web of Objects

Values and Objects

Follow the Messages

Tell, Don't Ask

But Sometimes Ask

Unit-Testing the Collaborating Objects

Table of Contents

Support for TDD with Mock Objects

Chapter 3: An Introduction to the Tools

Stop Me If You've Heard This One Before

A Minimal Introduction to JUnit 4

Hamcrest Matchers and `assertThat()`

jMock2: Mock Objects

Part II: The Process of Test-Driven Development

Chapter 4: Kick-Starting the Test-Driven Cycle

Introduction

First, Test a Walking Skeleton

Deciding the Shape of the Walking Skeleton

Build Sources of Feedback

Expose Uncertainty Early

Chapter 5: Maintaining the Test-Driven Cycle

Introduction

Start Each Feature with an Acceptance Test

Separate Tests That Measure Progress from Those That Catch
Regressions

Start Testing with the Simplest Success Case

Write the Test That You'd Want to Read

Watch the Test Fail

Develop from the Inputs to the Outputs

Unit-Test Behavior, Not Methods

Listen to the Tests

Tuning the Cycle

Chapter 6: Object-Oriented Style

Introduction

Designing for Maintainability

Table of Contents

Internals vs. Peers

No And's, Or's, or But's

Object Peer Stereotypes

Composite Simpler Than the Sum of Its Parts

Context Independence

Hiding the Right Information

An Opinionated View

Chapter 7: Achieving Object-Oriented Design

How Writing a Test First Helps the Design

Communication over Classification

Value Types

Where Do Objects Come From?

Identify Relationships with Interfaces

Refactor Interfaces Too

Compose Objects to Describe System Behavior

Building Up to Higher-Level Programming

And What about Classes?

Chapter 8: Building on Third-Party Code

Introduction

Only Mock Types That You Own

Mock Application Objects in Integration Tests

Part III: A Worked Example

Chapter 9: Commissioning an Auction Sniper

To Begin at the Beginning

Communicating with an Auction

Getting There Safely

This Isn't Real

Chapter 10: The Walking Skeleton

Table of Contents

Get the Skeleton out of the Closet

Our Very First Test

Some Initial Choices

Chapter 11: Passing the First Test

Building the Test Rig

Failing and Passing the Test

The Necessary Minimum

Chapter 12: Getting Ready to Bid

An Introduction to the Market

A Test for Bidding

The AuctionMessageTranslator

Unpacking a Price Message

Finish the Job

Chapter 13: The Sniper Makes a Bid

Introducing AuctionSniper

Sending a Bid

Tidying Up the Implementation

Defer Decisions

Emergent Design

Chapter 14: The Sniper Wins the Auction

First, a Failing Test

Who Knows about Bidders?

The Sniper Has More to Say

The Sniper Acquires Some State

The Sniper Wins

Making Steady Progress

Chapter 15: Towards a Real User Interface

A More Realistic Implementation

Table of Contents

- Displaying Price Details
- Simplifying Sniper Events
- Follow Through
- Final Polish
- Observations

Chapter 16: Sniping for Multiple Items

- Testing for Multiple Items
- Adding Items through the User Interface
- Observations

Chapter 17: Teasing Apart Main

- Finding a Role
- Extracting the Chat
- Extracting the Connection
- Extracting the SnipersTableModel
- Observations

Chapter 18: Filling In the Details

- A More Useful Application
- Stop When We've Had Enough
- Observations

Chapter 19: Handling Failure

- What If It Doesn't Work?
- Detecting the Failure
- Displaying the Failure
- Disconnecting the Sniper
- Recording the Failure
- Observations

Part IV: Sustainable Test-Driven Development

Chapter 20: Listening to the Tests

Table of Contents

Introduction

I Need to Mock an Object I Can't Replace (without Magic)

Logging Is a Feature

Mocking Concrete Classes

Don't Mock Values

Bloated Constructor

Confused Object

Too Many Dependencies

Too Many Expectations

What the Tests Will Tell Us (If We're Listening)

Chapter 21: Test Readability

Introduction

Test Names Describe Features

Canonical Test Structure

Streamline the Test Code

Assertions and Expectations

Literals and Variables

Chapter 22: Constructing Complex Test Data

Introduction

Test Data Builders

Creating Similar Objects

Combining Builders

Emphasizing the Domain Model with Factory Methods

Removing Duplication at the Point of Use

Communication First

Chapter 23: Test Diagnostics

Design to Fail

Small, Focused, Well-Named Tests

Table of Contents

- Explanatory Assertion Messages
- Highlight Detail with Matchers
- Self-Describing Value
- Obviously Canned Value
- Tracer Object
- Explicitly Assert That Expectations Were Satisfied
- Diagnostics Are a First-Class Feature

Chapter 24: Test Flexibility

- Introduction
- Test for Information, Not Representation
- Precise Assertions
- Precise Expectations
- "Guinea Pig" Objects

Part V: Advanced Topics

Chapter 25: Testing Persistence

- Introduction
- Isolate Tests That Affect Persistent State
- Make Tests Transaction Boundaries Explicit
- Testing an Object That Performs Persistence Operations
- Testing That Objects Can Be Persisted
- But Database Tests Are S-l-o-w!

Chapter 26: Unit Testing and Threads

- Introduction
- Separating Functionality and Concurrency Policy
- Unit-Testing Synchronization
- Stress-Testing Passive Objects
- Synchronizing the Test Thread with Background Threads
- The Limitations of Unit Stress Tests

Table of Contents

Chapter 27: Testing Asynchronous Code

Introduction

Sampling or Listening

Two Implementations

Runaway Tests

Lost Updates

Testing That an Action Has No Effect

Distinguish Synchronizations and Assertions

Externalize Event Sources

Afterword: A Brief History of Mock Objects

Appendix A: jMock2 Cheat Sheet

Appendix B: Writing a Hamcrest Matcher

Bibliography

Index