



ESSENTIAL SKILLS FOR THE AGILE DEVELOPER

A Guide to Better Programming and Design

*Net*bjectives

Lean-Agile Series

ALAN SHALLOWAY
SCOTT BAIN
KEN PUGH
AMIR KOLSKY

Praise for *Essential Skills for the Agile Developer*

“I tell teams that the lean and agile practices should be treated like a buffet: Don’t try and take everything, or it will make you ill—try the things that make sense for your project. In this book the authors have succinctly described the ‘why’ and the ‘how’ of some of the most effective practices, enabling all software engineers to write quality code for short iterations in an efficient manner.”

—**Kay Johnson**

Software Development Effectiveness Consultant, IBM

“Successful agile development requires much more than simply mastering a computer language. It requires a deeper understanding of agile development methodologies and best practices. *Essential Skills for the Agile Developer* provides the perfect foundation for not only learning but truly understanding the methods and motivations behind agile development.”

—**R.L. Bogetti**

www.RLBogetti.com,

Lead System Designer, Baxter Healthcare

“*Essential Skills for the Agile Developer* is an excellent resource filled with practical coding examples that demonstrate key agile practices.”

—**Dave Hendricksen**

Software Architect, Thomson Reuters

Essential Skills for the Agile Developer: A Guide to Better Programming and Design

Table of Contents

Contents

Series Foreword

Preface

Acknowledgments

About the Authors

Part I: The Core Trim Tabs

Chapter 1 Programming by Intention

Advantages

Summary

Chapter 2 Separate Use from Construction

An Important Question to Ask

Perspectives

Timing Your Decisions

Overloading and C++

Validating This for Yourself

Summary

Chapter 3 Define Tests Up Front

A Trim Tab: Testing and Testability

What Is Testing?

Testability and Code Quality

Case Study: Testability

Table of Contents

A Reflection on Up-Front Testing

Summary

Chapter 4 Shalloways Law and Shalloways Principle

Types of Redundancy

Redefining Redundancy

Other Types of Redundancy

The Role of Design Patterns in Reducing Redundancy

Few Developers Spend a Lot of Time Fixing Bugs

Redundancy and Other Code Qualities

Summary

Chapter 5 Encapsulate That!

Unencapsulated Code: The Sabotage of the Global Variable

Encapsulation of Member Identity

Self-Encapsulating Members

Preventing Changes

The Difficulty of Encapsulating Reference Objects

Breaking Encapsulation with Get()

Encapsulation of Object Type

Encapsulation of Design

Encapsulation on All Levels

Practical Advice: Encapsulate Your Impediments

Summary

Chapter 6 Interface-Oriented Design

Design to Interfaces

Definition of Interface

Interface Contracts

Separating Perspectives

Mock Implementations of Interfaces

Keep Interfaces Simple

Table of Contents

- Avoids Premature Hierarchies
- Interfaces and Abstract Classes
- Dependency Inversion Principle
- Polymorphism in General
- Not for Every Class
- Summary

Chapter 7 Acceptance TestDriven Development (ATDD)

- Two Flows for Development
- Acceptance Tests
- An Example Test
- Implementing the Acceptance Tests
- An Exercise
- What to Do If the Customer Wont Tell You
- Summary

Part II: General Attitudes

Chapter 8 Avoid Over- and Under-Design

- A Mantra for Development
- The Pathologies of Code Qualities
- Avoid Over- and Under-Design
- Minimize Complexity and Rework
- Never Make Your Code Worse/Only Degrade Your Code Intentionally
- Keep Your Code Easy to Change, Robust, and Safe to Change
- A Strategy for Writing Modifiable Code in a Non-Object-Oriented
or Legacy System
- Summary

Chapter 9 Continuous Integration

- Branching the Source Code
- The Merge-Back
- Test-Driven Development and Merge Cost

Table of Contents

Continuous Integration

Continuous Integration Servers

Summary

Part III: Design Issues

Chapter 10 Commonality and Variability Analysis

Using Nouns and Verbs as a Guide: Warning, Danger Ahead!

What Is the Real Problem?

What We Need to Know

Commonality and Variability Analysis

A New Paradigm for Finding Objects

The Analysis Matrix: A Case Study

Summary

Chapter 11 Refactor to the Open-Closed

The Open-Closed Principle

Refactoring

Summary

Chapter 12 Needs versus Capabilities Interfaces

The Law of Demeter

Coupling, Damned Coupling, and Dependencies

The Ideal Separation: Needs Interfaces and Capabilities Interfaces

Back to the Law of Demeter

Summary

Chapter 13 When and How to Use Inheritance

The Gang of Four

Initial Vectors, Eventual Results

Favoring Delegation

The Use of Inheritance versus Delegation

Uses of Inheritance

Scalability

Table of Contents

Applying the Lessons from the Gang of Four to Agile Development

Testing Issues

Theres More

Part IV: Appendixes

Appendix A: Overview of the Unified Modeling Language (UML)

What Is the UML?

The Class Diagram

Sequence Diagram

Summary

Appendix B: Code Qualities

Christmas-Tree Lights: An Analogy

Cohesion

Coupling

Redundancy

Encapsulation

Appendix C: Encapsulating Primitives

Encapsulating Primitives in Abstract Data Types

Principles

Narrow the Contract

Expanding Abstract Data Types

Use Text as External Values

Enumerations Instead of Magic Values

Disadvantages

Summary

Index