Microsoft

Adaptive Code

Agile coding with design patterns and SOLID principles

Second Edition

Best practices

Gary McLean Hall

Microsoft

Adaptive Code: Agile coding with design patterns and SOLID principles

Second Edition

Gary McLean Hall

Adaptive Code: Agile coding with design patterns and SOLID principles

Table of Contents

Cover

Title Page

Copyright Page

Acknowledgments

Contents

Introduction

PART I: AGILE DEVELOPMENT FRAMEWORKS

- Chapter 1 Introduction to Scrum
  - Scrum versus waterfall
  - Roles and responsibilities
    - Product owner
    - Scrum master
    - Development team
  - Artifacts
    - The Scrum board
    - Charts and metrics
    - Backlogs
  - The sprint
    - Release planning
    - Sprint planning
    - Daily Scrum
    - Sprint demo
    - Sprint retrospective

Table of Contents

Scrum calendar

Agile in the real world

Rigidity

Untestability

Metrics

Conclusion

Chapter 2 Introduction to Kanban

Kanban quickstart

The information radiator

Limiting work in progress

Protecting against change

Defining done

Event-driven ceremonies

Classes of service

Service level agreements

Class WIP limits

People as a class of service

Analysis

Lead time and cycle time

Cumulative flow diagrams

Conclusion

PART II: FOUNDATIONS OF ADAPTIVE CODE

Chapter 3 Dependencies and layering

Dependencies

A simple example

Framework dependencies

Third-party dependencies

Modeling dependencies in a directed graph

Managing dependencies

Implementations versus interfaces

The new code smell

Pearson

Table of Contents

Alternatives to object construction

Resolving dependencies

Dependency management with NuGet

Layering

Common layering patterns

Cross-cutting concerns

Asymmetric layering

Conclusion

Chapter 4 Interfaces and design patterns

What is an interface?

Syntax

Explicit implementation

Polymorphism

Adaptive design patterns

The Null Object pattern

The Adapter pattern

The Strategy pattern

Further versatility

Duck-typing

Mixins

Fluent interfaces

Conclusion

Chapter 5 Testing

Unit testing

Arrange, Act, Assert

Test-driven development

More complex tests

Unit-testing patterns

Writing maintainable tests

The Builder pattern for tests

The Builder pattern

Table of Contents

Clarifying unit test intent

Writing tests first

What is TDD?

Test-driven design

Test-first development

Further testing

The testing pyramid

Testing pyramid anti-patterns

The testing quadrant

Testing for prevention and cure

How do you decrease MTTR?

Conclusion

Chapter 6 Refactoring

Introduction to refactoring

Changing existing code

A new account type

Aggressive refactoring

Red, green, refactorredesign

Making legacy code adaptive

The golden master technique

Conclusion

PART III: SOLID CODE

Chapter 7 The single responsibility principle

Problem statement

Refactoring for clarity

Refactoring for abstraction

SRP and the Decorator pattern

The Composite pattern

Predicate decorators

Branching decorators

Lazy decorators

Pearson

Table of Contents

Logging decorators

Profiling decorators

Decorating properties and events

Conclusion

Chapter 8 The open/closed principle

Introduction to the open/closed principle

The Meyer definition

The Martin definition

Bug fixes

Client awareness

Extension points

Code without extension points

Virtual methods

Abstract methods

Interface inheritance

Design for inheritance or prohibit it

Protected variation

Predicted variation

A stable interface

Just enough adaptability

Predicted variation versus speculative generality

Do you need so many interfaces?

Conclusion

Chapter 9 The Liskov substitution principle

Introduction to the Liskov substitution principle

Formal definition

LSP rules

Contracts

Preconditions

Postconditions

Data invariants

Pearson

Table of Contents

Liskov contract rules

Code contracts

Covariance and contravariance

Definitions

Liskov type system rules

Conclusion

Chapter 10 Interface segregation

A segregation example

A simple CRUD interface

Caching

Multiple interface decoration

Client construction

Multiple implementations, multiple instances

Single implementation, single instance

The Interface Soup anti-pattern

Splitting interfaces

Client need

Architectural need

Single-method interfaces

Conclusion

Chapter 11 Dependency inversion

Structuring dependencies

The Entourage anti-pattern

The Stairway pattern

An example of abstraction design

Abstractions

Concretions

Abstracting capabilities

The improved client

Abstracting queries

Further abstraction

Pearson

Table of Contents

Conclusion

PART IV: APPLYING ADAPTIVE CODE

Chapter 12 Dependency injection

Humble beginnings

The Task List application

Constructing the object graph

Beyond simple injection

The Service Locator anti-pattern

Illegitimate Injection

The composition root

Convention over configuration

Conclusion

Chapter 13 Coupling, cohesion, and connascence

Coupling and cohesion

Coupling

Cohesion

Connascence

Name

Type

Meaning

Algorithm

Position

Execution order

Timing

Value

Identity

Measuring connascence

Locality

Unofficial connascence

Static vs. dynamic connascence

Conclusion

Pearson

Table of Contents

Appendix: Adaptive tools
Index
About the author

Pearson