EMERGENT DESIGN

The Evolutionary Nature of Professional Software Development

Net Objectives

Product Development Series

SCOTT L. BAIN

Emergent Design

Emergent Design: The Evolutionary Nature of Professional Software Development

Table of Contents

Contents

Series Foreword

Preface

Acknowledgments

About the Author

Chapter 1 Software as a Profession

- How Long Have Human Beings Been Making Software?
- What Sort of Activity Is Software Development?
- What Is Missing?
- Who Is Responsible?
- Uniqueness

Chapter 2 Out of the Closet, Off to the Moon

- Patterns and Professionalism in Software Development
- Andrea's Closet
- Off to the Moon
  - Forces Lead to Forces
  - Different Forces, Different Design
  - And There Are More Contextual Forces
  - The Costs and Benefits
  - On to Mars

Pearson

Table of Contents

The Value of Patterns
Summary
Chapter 3 The Nature of Software Development
We Fail Too Much
Definitions of Success
The Standish Group
Doing the Wrong Things
Doing the Things Wrong
Time Goes By, Things Improve
One Reason: The Civil Engineering Analogy
Giving Up Hope
Ignoring Your Mother
Bridges Are Hard, Software Is Soft
We Swim in an Ocean of Change
Accept Change
Embrace Change
Capitalize on Change
A Better Analogy: Evolving Systems
Summary
Chapter 4 Evolution in Code: Stage 1
Procedural Logic Replaced with Object Structure
The Origins of Object Orientations and Patterns
An Example: Simple Conditionals and the Proxy Pattern
The Next Step: Either This or That
Why Bother?
One Among Many

Pearson

Table of Contents

Summary

Chapter 5 Using and Discovering Patterns

Design from Context: More Carpentry from Scott

Patterns Lead to Another Cognitive Perspective

Patterns Help Give Us a Language for Discussing Design

Patterns in This Book

Summary

Chapter 6 Building a Pyramid

Elements of the Profession

A Visual Representation

Summary

Chapter 7 Paying Attention to Qualities and Pathologies

Encapsulation

Cohesion

Method Cohesion

Cohesion of Perspective Level

Class Cohesion

How Cohesive Is Cohesive Enough?

Coupling

Intentional Coupling Versus Accidental Coupling

Types of Coupling

Redundancy

Redundancy and Coupling

Testability

Readability

Pathologies

Indicators of Weak Cohesion

Pearson

Table of Contents

Indicators of Accidental or Illogical Coupling

Indicators of Redundancy

Summary

Chapter 8 Paying Attention to Principles and Wisdom

Separating Use from Creation

Fowler's Perspectives

Another Kind of Perspective

The Perspective of Use

A Separate Perspective: Creation

Considering Construction Details Last

The Real World

The Open-Closed Principle

Open-Closed at the Class Level

Open-Closed at the Method Level

The Dependency Inversion Principle

Advice from the Gang of Four

Designing to the Interface of a Method

Designing to the Interface of a Class

GoF: Favor Object Aggregation Over Class Inheritance

GoF: Consider What Should Be Variable in Your Design and Encapsulate the Concept That Varies

Summary

Chapter 9 Paying Attention to Practices

Consistent Coding Style

Comments

Naming Classes, Methods, and Variables

Virtues of Coding Standards

Programming by Intention

Pearson

Table of Contents

Are They Really All Private?

Encapsulating the Constructor

Principles Versus Practices

Making the Decision

Commonality-Variability Analysis

Practices and Freedom

Summary

Chapter 10 Paying Attention to Disciplines: Unit Testing

Economies of Testing

Unit Testing

Up-Front Testing

JUnit Framework

JUnit Basics

JUnit Examples

Rule.java: Code First, Then Test

RuleContainer.java: Test First, Then Code

Eliminating Redundancy: @Before and @After

Automating Tests in Batches

Exceptions and Unit Testing

Mock Objects

MockObject Frameworks

Faking It

Dependency Injection and the Endo-Testing Technique

Endo-Testing

Summary

Chapter 11 Paying Attention to Disciplines: Refactoring

Refactoring Bad Code

Refactoring Good Code

Pearson

Table of Contents

Structural Changes Versus Functional Changes
Refactoring Helps You Choose Your Battles
Patterns Can Be Targets of Refactoring
Avoiding Refactoring: Prefactoring
The Mechanics of Refactoring
Refactoring Legacy Code
Summary
Chapter 12 Test-Driven Development
What Makes Development Test-Driven?
Test-Driven Versus Test-First
Designing from the Perspective of the Unit Test
Testing and Quality
Testing and Cohesion
Testing and Coupling
Testing and Redundancy
Test-Driven Development and Patterns
The Strategy Pattern
Turtles All the Way Down
Mock Object/Mock Turtles
Mock Objects
Mock Turtles
Testing the Decorator Pattern
Summary
Chapter 13 Patterns and Forces
Making Decisions in an Evolving Design
Christopher Alexander and Forces
The Signal Processor Example

Pearson

Table of Contents

The PKZip Example

Testing and Forces

More Choices, More Forces

Summary

Chapter 14 Emergent Design: A Case Study

The Problem Domain: The MWave Corporation

The Teams

The Simplest Thing That Could Possibly Work

A New Requirement: Complex Machines

Oh, By the Way

More Good News

Summary: What a Long, Strange Trip It Has Been

Chapter 15 A Conclusion: 2020

Appendix A: Evolutionary Paths

Encapsulated Constructor to Singleton

Programming-by-Intention to Encapsulated Constructor to Strategy (Varying on an Issue Extrinsic to the Client)

Programming-by-Intention to Encapsulated Constructor to Strategy (Varying on an Issue Intrinsic to the Client)

Programming-by-Intention to Encapsulated Constructor to Chain of Responsibility (Varying on an Issue Extrinsic to the Client)

Programming-by-Intention to Encapsulated Constructor to Chain of Responsibility (Varying on an Issue Intrinsic to the Client)

Encapsulated Construction to Strategy to Singleton (Object Factory) to Adapter/Façade

Encapsulated Constructor to Proxy to Decorator with Singleton (Object

Table of Contents

Factory)

Appendix B: Overview of Patterns Used in the Examples

The Abstract Factory Pattern

Contextual Forces

Implementation Forces

Consequent Forces

The Adapter Pattern

Contextual Forces

Implementation Forces

Consequent Forces

The Bridge Pattern

Contextual Forces

Implementation Forces

Consequent Forces

The Chain of Responsibility Pattern

Contextual Forces

Implementation Forces

Consequent Forces

Chain of Responsibility: The Poker Example

The Composite Pattern

Contextual Forces

Implementation Forces

Consequent Forces

The Decorator Pattern

Contextual Forces

Implementation Forces

Consequent Forces

The Façade Pattern

Pearson

Table of Contents

- Contextual Forces
- Implementation Forces
- Consequent Forces

The Proxy Pattern
- Contextual Forces
- Implementation Forces
- Consequent Forces

The Singleton Pattern
- Contextual Forces
- Implementation Forces
- Consequent Forces

The Strategy Pattern
- Contextual Forces
- Implementation Forces
- Consequent Forces

The Template Method
- Contextual Forces
- Implementation Forces
- Consequent Forces

Appendix C: The Principle of the Useful Illusion
- My View
- An Alternate View

Bibliography

Index

Pearson