

# Testing, Testing... 1, 2, 3

---

Dave Dribin



# Testing, Testing... 1, 2, 3

---

Dave Dribin – @ddribin 



# Spreading the Test Infection

---

“Every programmer knows they should write tests for their code. Few do.”



```
HANDLEP*** Address 8016a950 has base at 80100000
```

```
.6.2 irq1:if SYSVER 0xf0000565
```

Name	Dll Base	DateStmp	Name
ntoskrnl.exe	80010000	33247f88	al.dll
atapi.sys	80007000	3324804	SIPORT.
Disk.sys	801db000	3360156	A382.SY
Ntfs.sys	80237000	344eeb4	wvid.sy
NTice.sys	f1f48000	31ec6c8d	loppy.SY
Cdrom.SYS	f228e000	31ec6c9	ull.SYS
KSecDD.SYS	f2290000	335a	.SYS
win32k.sys	fe0c2000	34	dll
Cdfs.SYS	fdca2000	3	
nbfs.sys	fdc35000		ys
netbt.sys	f1f68000		.s
afd.sys	f2008000		s
Parport.SYS	fdc14000		y.
	f14d0000		y





# Software Testing

---

- Find bugs
- Meets user expectations
- Provide confidence in software

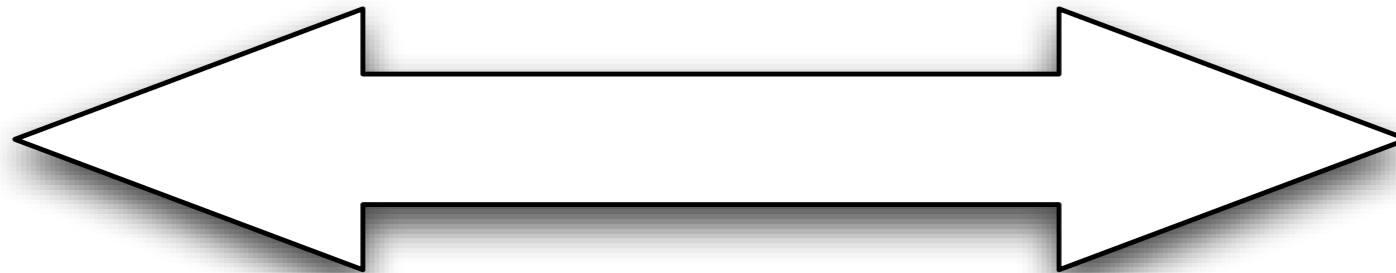
# Categories of Tests

---

## Functional Tests

System Tests

Unit Tests



Whole  
Application

Single Class

# System Tests: Benefits

---

- Tests the application as it's shipping
- Often gets customer involved
- Useful for usability and performance testing



# System Tests: Disadvantages

---

- More difficult to automate
- More difficult to test edge cases
- Errors harder to debug
- Slow feedback cycle

# What is a Unit Test?

---

- Tests a single class
- Very fast
- Automated
  - Repeatability
  - Self-checking

# A test is not a unit test if:

---

- It talks to the database
- It communicates across the network
- It touches the file system
- It can't run at the same time as any of your other unit tests
- You have to do special things to your environment

Michael Feathers,  
idealized unit test

```
#import <Foundation/Foundation.h>
```

```
@interface Rectangle : NSObject
```

```
{
```

```
    float _leftX;
```

```
    float _bottomY;
```

```
    float _width;
```

```
    float _height;
```

```
}
```

```
@property (readonly) float perimeter;
```

```
@property (readonly) float area;
```

```
- (id) initWithLeftX:(float)leftX  
                bottomY:(float)bottomY  
                width:(float)width  
                height:(float)height;
```

```
@end
```

```
#import <Foundation/Foundation.h>
#import <SenTestingKit/SenTestingKit.h>
#import "Rectangle.h"
```

```
@interface RectangleTest : SenTestCase
@end
```

```
@implementation RectangleTest
```

```
- (void)testPerimeter
{
    // ...
}
```

```
- (void)testArea
{
    // ...
}
```

```
@end
```

```
- (void)testPerimeter
{
    // Setup
    Rectangle * rectangle =
        [[[Rectangle alloc] initWithLeftX:5
                                     bottomY:3
                                     width:8
                                     height:9] autorelease];

    // Exercise
    float actualPerimeter = rectangle.perimeter;

    // Verify
    STAssertEqualsWithAccuracy(actualPerimeter,
                               34.0f, 0.01f, nil);
}
```

# Unit Tests: Benefits

---

- Lots of regression tests catch when changes break existing code
- Allows safe refactoring
- Cleaner, easier to maintain code
- Test hard-to-run conditions (network errors, disk errors, date related)
- Locality of errors
- Tests as sample code
- Faster development

# Refactoring

---



# Refactoring

---

“You keep using that word. I do not think it means what you think it means.”

# REFACTORING

IMPROVING THE DESIGN  
OF EXISTING CODE

**MARTIN FOWLER**

With Contributions by **Kent Beck, John Brant,  
William Opdyke, and Don Roberts**

Foreword by **Erich Gamma**  
Object Technology International Inc.



# Refactoring

---

- Refactoring (noun): a **small incremental change** made to the internal structure of software to make it **easier to understand** and cheaper to modify **without changing its observable behavior**.

# Why Refactor

---

- Improves the design after it has been written
- Makes software easier to understand
- Helps you find bugs
- Helps you program faster

# Refactoring

---

- “It is essential for refactoring that you have good tests.”
- “The tests are essential because even though I follow refactorings [...] I’m still human and still make mistakes.”
- “Refactoring requires tests. If you want to refactor, you have to write tests.”

# Writing Tests After Code

---

- Takes a lot of discipline
- Often skipped due to lack of time
- Hard to retrofit tests on existing code

# Writing Tests Before Code

---

- Write failing tests first
- Implement code so tests pass
- Refactor to make it clean

# Test Driven Development

---

- Make it fail
- Make it work
- Make it clean
  
- “An acquired taste”



# Resistance to Unit Testing

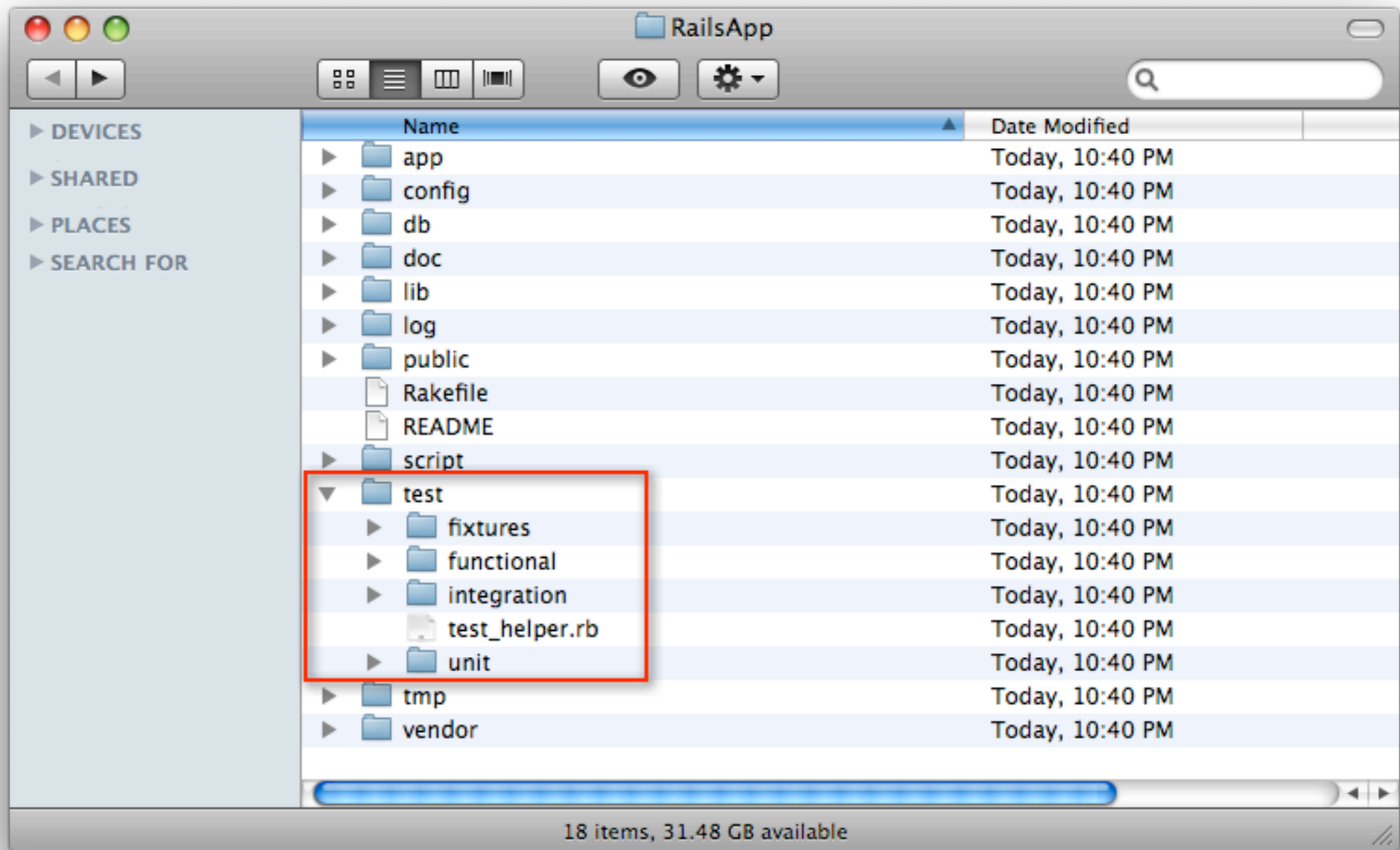
---

- Writing tests is a bunch of extra code.
  - It'll slow me down
- I'm working with an existing code base, and it's hard to add tests
- I don't know how to write unit tests
- TDD sounds ridiculous

# Other communities are “Test Infected”

---

- Java
  - JUnit
- Ruby and Ruby on Rails
  - New Rails app creates functional, integration, and unit tests directories



# Lack of Testing in Objective-C Community

---

- Why?
  - “Mac OS X/iPhone apps are more UI centric”
- Plenty of non-drawing code to test
- Model-View-Controller actually makes code *easier* to test
  - As do delegates
  - But... tendency towards large controller classes

# REFACTORING

IMPROVING THE DESIGN  
OF EXISTING CODE

**MARTIN FOWLER**

With Contributions by **Kent Beck, John Brant,  
William Opdyke, and Don Roberts**

Foreword by **Erich Gamma**  
Object Technology International Inc.

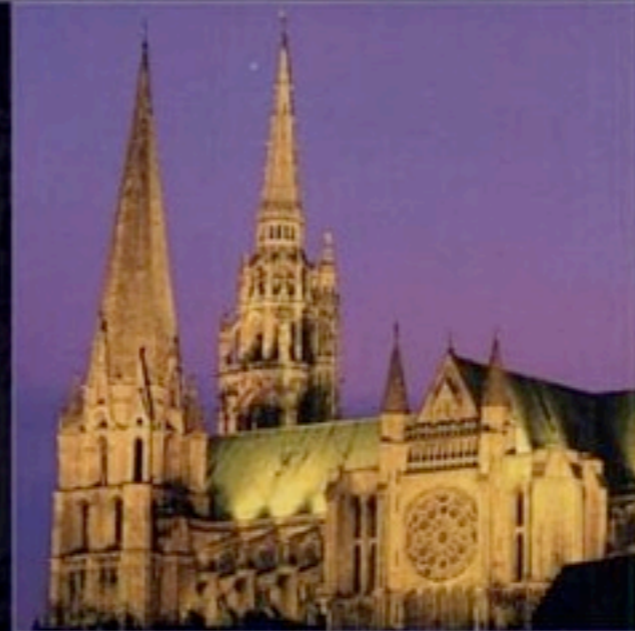


*The Addison-Wesley Signature Series*

# TEST-DRIVEN DEVELOPMENT

BY EXAMPLE

KENT BECK



A KENT BECK SIGNATURE  
BOOK

# test-driven development

*A Practical Guide*



- ▶ The relentlessly practical Test-Driven Development guide: real problems, real solutions, real code
- ▶ Includes a start-to-finish project written in Java™ and using JUnit
- ▶ Introduces TDD frameworks for C++, C#.NET, Python, VB6, and more
- ▶ For every developer and project manager interested in TDD

**David Astels**

Foreword by Ron Jeffries

THE COAD SERIES

*The Addison-Wesley Signature Series*

# xUNIT TEST PATTERNS

REFACTORING  
TEST CODE

GERARD MESZAROS



*Foreword by Martin Fowler*





Robert C. Martin Series



**WORKING  
EFFECTIVELY  
WITH  
LEGACY CODE**

---

Michael C. Feathers

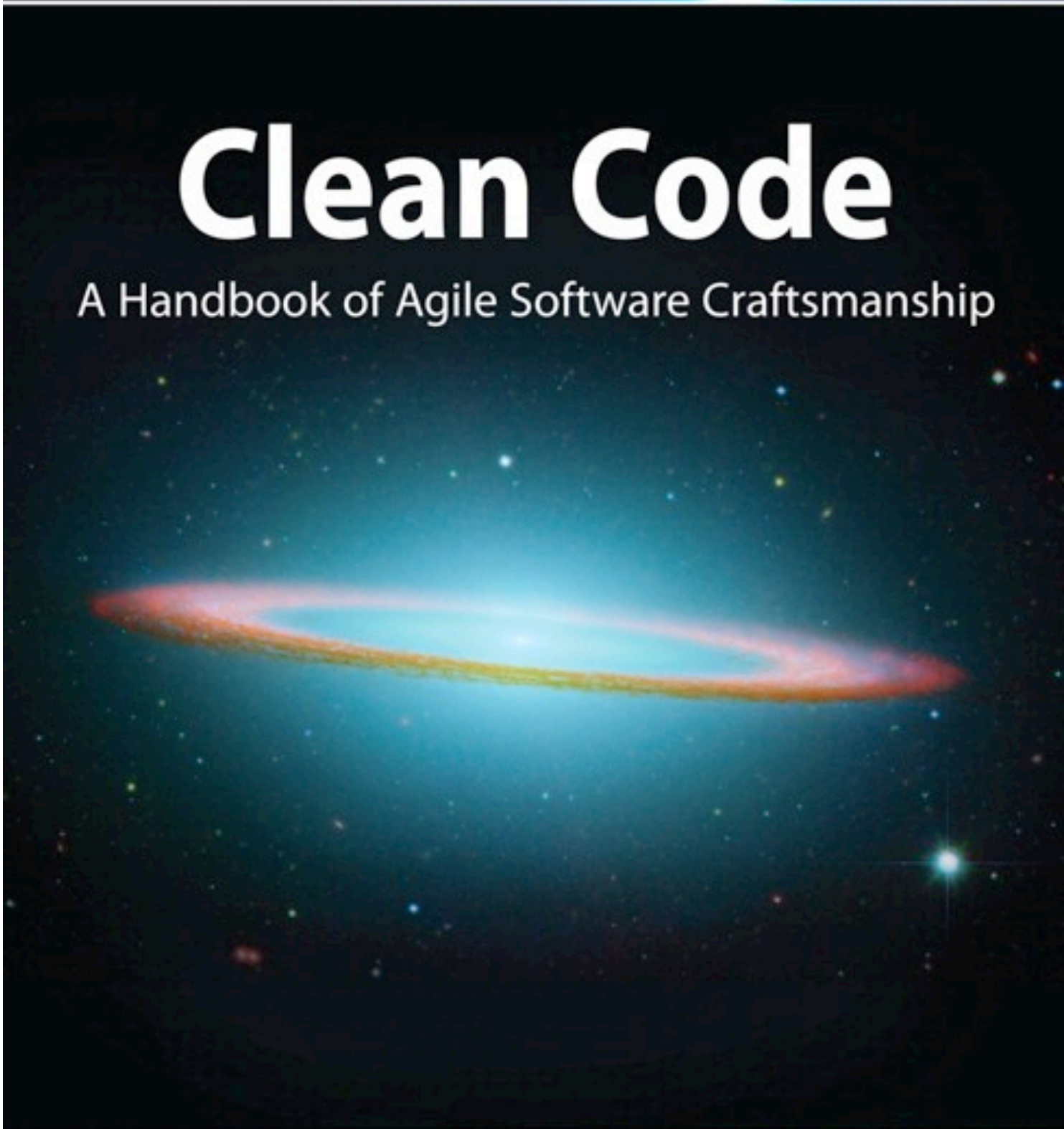
The logo for Prentice Hall, consisting of a red grid of dots to the left of the text "PRENTICE HALL" in red capital letters.

PRENTICE  
HALL

Robert C. Martin Series

# Clean Code

A Handbook of Agile Software Craftsmanship

The background of the book cover is a dark space scene featuring a prominent edge-on spiral galaxy with a bright yellow and orange core and reddish-pink outer rings. The galaxy is set against a deep blue and black background filled with numerous small, distant stars.

Foreword by James O. Coplien

Robert C. Martin



# TESTING

I FIND YOUR LACK OF TESTS DISTURBING.