
SOFTWARE
CRAFTSMANSHIP
FOR NEW
DEVELOPERS
LOUISVILLE
.NET MEETUP

Who is Chad Green

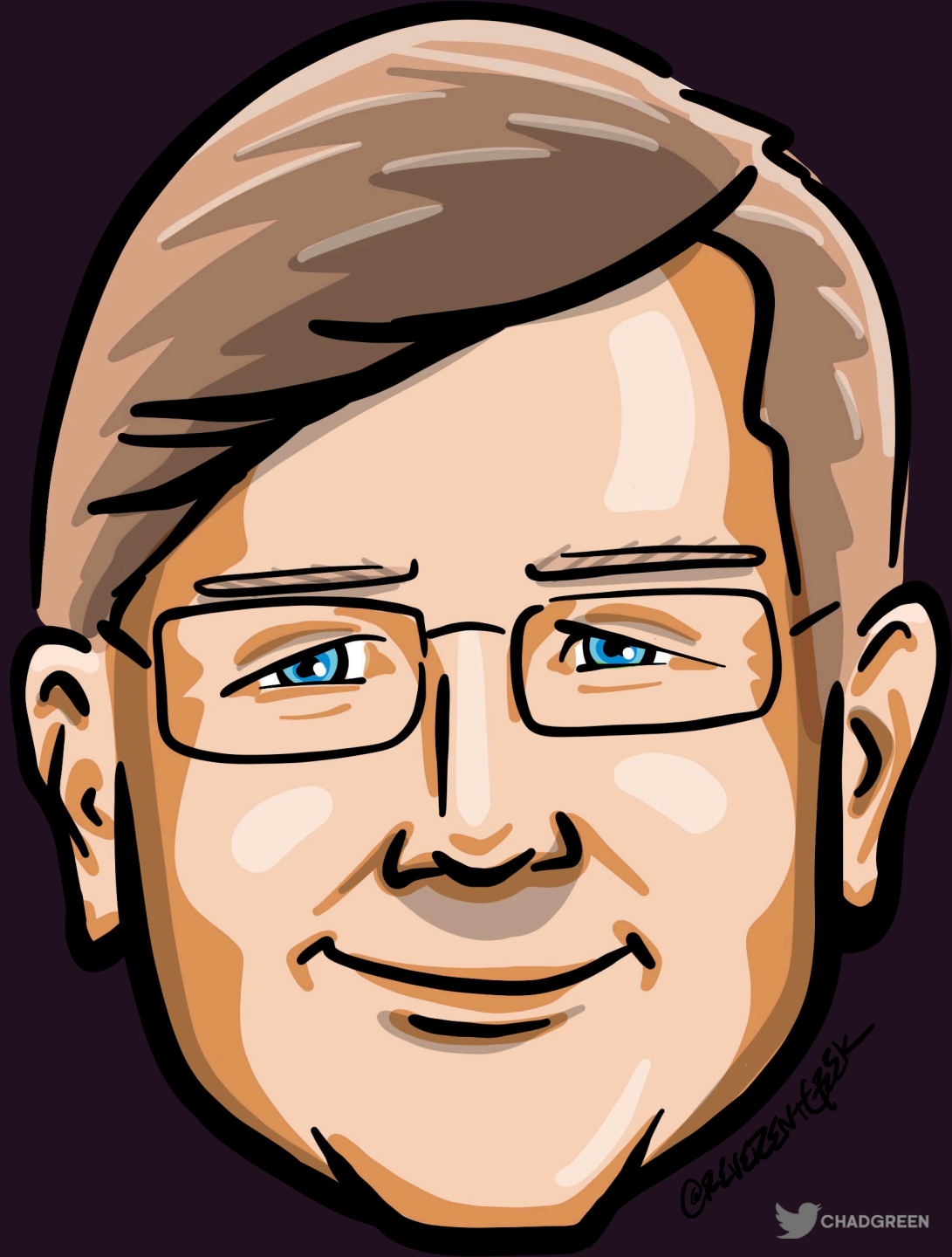
chadgreen@chadgreen.com

TaleLearnCode

ChadGreen.com

ChadGreen & TaleLearnCode

ChadwickEGreen



What is Software Craftsmanship

Software Craftsmanship for New Developers

What Software Craftsmanship is not

Beautiful Code

What Software Craftsmanship is not

Beautiful Code

What Software Craftsmanship is not

Beautiful Cod

Test-Driven Development

What Software Craftsmanship is not

Beautiful Cod

Test-Driven Development

What Software Craftsmanship is not

Beautiful Cod

Self-Selected Group of People

What Software Craftsmanship is not

Beautiful Cod

Self-Selected Group of People

What Software Craftsmanship is not

Beautiful Cod

Specific Technologies or Methodologies

Self-Selected Group of People

What Software Craftsmanship is not

Beautiful Cod

Specific Technologies or Methodologies

Self-Selected Group of People

What Software Craftsmanship is not

Beautiful Cod

Self-Selected Group of People

Certifications

Specific Technologies or Methodologies

What Software Craftsmanship is not

Beautiful Cod

Self-Selected Group of People

Certifications

Specific Technologies or Methodologies

What Software Craftsmanship is not

Beautiful Cod

Self-Selected Group of People

Religion

Specific Technologies or Methodologies

Certifications

What Software Craftsmanship is not

Beautiful Cod

Self-Selected Group of People

Religion

Specific Technologies or Methodologies

Certifications

What Software Craftsmanship is not

Beautiful Code

Test-Driven Development

Self-Selected Group of People

Specific Technologies or Methodologies

Certifications

Religion

What Software Craftsmanship is not

Beautiful Code

Test-Driven Development

Self-Selected Group of People

Specific Technologies or Methodologies

Certifications

Religion

Why Software Craftsmanship

- Software developers have had a hard time defining themselves

Scientific Approach

Engineering Approach

What is Software Development

Craft/Trade

Engineering

Science

Art

Agile Manifesto Ignites a Spark

Individuals and interactions

over processes and tools

History of Software Craftsmanship

1992: What is Software Deign

History of Software Craftsmanship

1997: *The Pragmatic Programmer*

History of Software Craftsmanship

2001: *Software Craftsmanship*

History of Software Craftsmanship

2002: Software Apprenticeship Summit

History of Software Craftsmanship

2006: 8th Light Founded

History of Software Craftsmanship

2008: Craftsmanship over Crap

History of Software Craftsmanship

2008: *Clean Code*

History of Software Craftsmanship

2008: Software Craftsmanship Summit

History of Software Craftsmanship

2009: Manifesto for Software Craftsmanship

History of Software Craftsmanship

2011: *Clean Coder*

Manifesto for Software Craftsmanship

Not only working software, but also

well-crafted software

Manifesto for Software Craftsmanship

Not only responding to change, but
also **steadily adding value**

Try and leave this world a little better than you found it, and when you turn comes to dies you can die happy in the feeling that at any rate you have not wasted your time but have done your best.

Robert Stephenson Smyth Bader-Powell,
founder of The Scot Association

Manifesto for Software Craftsmanship

Not only individuals and interactions,
but also **a community of professionals**

Manifesto for Software Craftsmanship

Not only customer collaboration, but also **productive partnerships**

Software Craftsmanship is about professionalism in software development.

Technical Debt

Software Craftsmanship for New Developers

What is Technical Debt

Implied cost of additional rework caused by choosing an easy solution

Example of Technical Debt

No User Roles

Example of Technical Debt

No User Roles

Permission for Specific Requirement

Example of Technical Debt

No User Roles

Permission for Specific Requirement

Differentiation of Users

Example of Technical Debt

No User Roles

Permission for Specific Requirement

Differentiation of Users

Yet Another Permission Change

Common Causes of Technical Debt

Insufficient up-front definition

Common Causes of Technical Debt

Business pressures

Common Causes of Technical Debt

Lack of process or understanding

Common Causes of Technical Debt

Tightly-coupled components

Common Causes of Technical Debt

Lack of a test suite

Common Causes of Technical Debt

Lack of documentation

Common Causes of Technical Debt

Lack of collaboration

Common Causes of Technical Debt

Parallel development

Common Causes of Technical Debt

Delayed refactoring

Common Causes of Technical Debt

Lack of alignment to standards

Common Causes of Technical Debt

Lack of knowledge

Common Causes of Technical Debt

Lack of ownership

Common Causes of Technical Debt

Poor technical leadership

Common Causes of Technical Debt

Last minute specification changes

Common Causes of Technical Debt

- Insufficient up-front definition
- Business pressures
- Lack of process or understanding
- Tightly-coupled components
- Lack of a test suite
- Lack of documentation
- Lack of collaboration
- Parallel development
- Delayed refactoring
- Lack of alignment to standards
- Lack of knowledge
- Lack of ownership
- Poor technical leadership
- Last minute specification changes

SOLID Principles

Software Craftsmanship for New Developers

S.O.L.I.D.

- First five object-oriented design principles
  - **S** – Single-responsibility principle
  - **O** – Open-closed principle
  - **L** – Liskov substitution principle
  - **I** – Interface segregation principle
  - **D** – Dependency Inversion Principle

Single Responsibility Principle (SRP)

A module should have one, and only one, reason to change

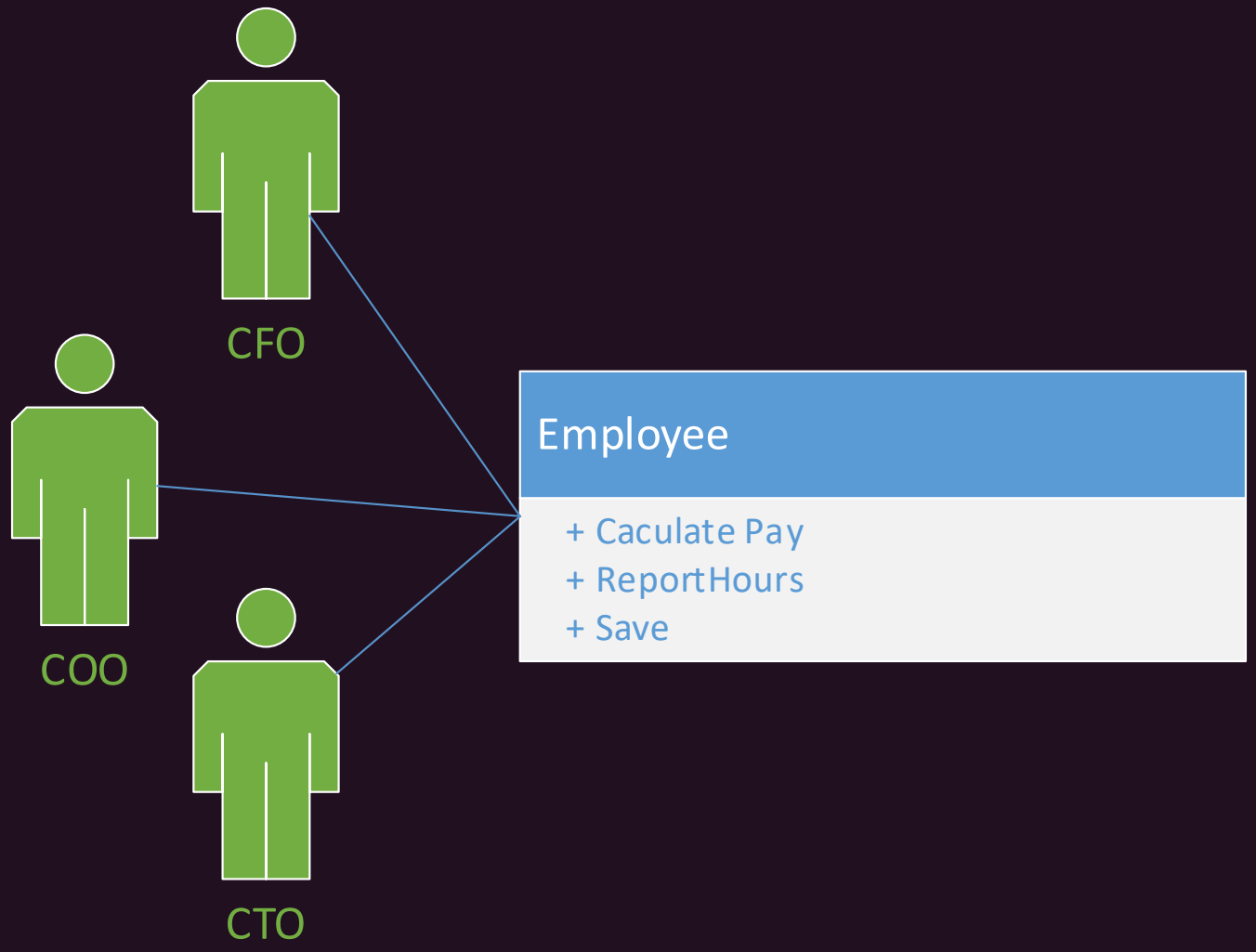
Single Responsibility Principle (SRP)

A module should have one, and only one, reason to change

Single Responsibility Principle (SRP)

A module should have one, and only one, reason to change

Single Responsibility Principle (SRP)



Open-Closed Principle (OCP)

A software artifact should be open for extension but closed for modification

Liskov Substitution Principle (LSP)

Let **q(x)** be a property provable about objects of **x** of type **T**. Then **q(y)** should be provable for objects **y** of type **y** where **S** is a subtype of **T**.

Liskov Substitution Principle (LSP)

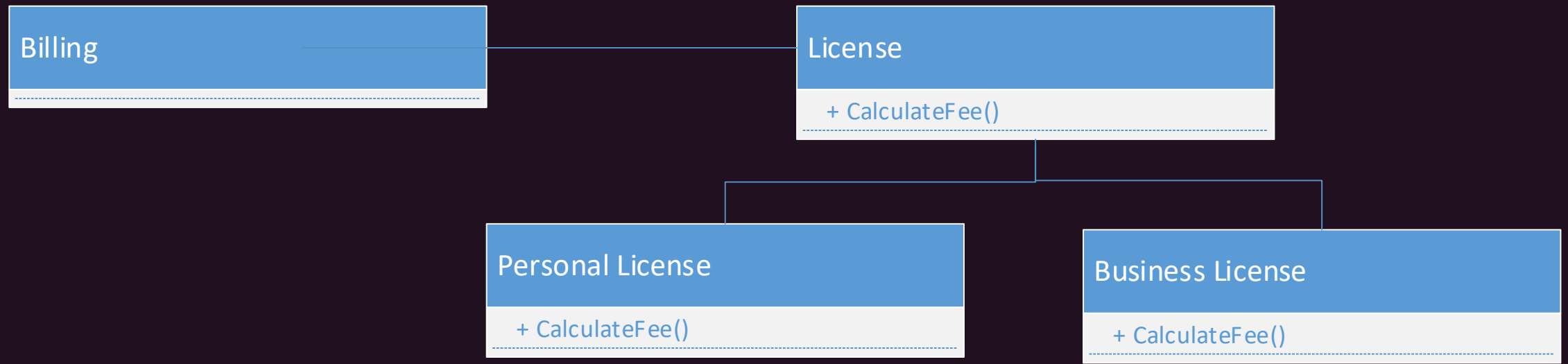
Let **q(x)** be a property provable about objects of **x** of type **T**. Then **q(y)** should be provable for objects **y** of type **y** where **S** is a subtype of **T**.

Liskov Substitution Principle (LSP)

Every derived class should be substitutable for their base class

Liskov Substitution Principle (LSP)

Every derived class should be substitutable for their base class



Interface Segregation Principle (ISP)

A client should never be forced to implement an interface that it does not use

Clients should not be forced to depend on methods they do not use

Dependency Inversion Principle (DIP)

Entities must depend on abstractions not on concretions.

Other Key Principles

Software Craftsmanship for New Developers

DRY – Don't Repeat Yourself

Every piece of knowledge must have a single, unambiguous, authoritative representation within a system

- If you write it once, think about encapsulating it.
- If you write it twice, you have to encapsulate it.
- If your write it three times, programming isn't for you.

Phil Japikse, Microsoft MVP, ASP Insider, MCSD, MCDBA, PSM II, PSD, CSM, Consultant, Coach, Author, Trainer

KISS – Keep it Simple Stupid

The simplest explanation tends to be the right one

YANGI – You Aren't Going to Need It

Implement things when you actually need them

Key Practices

Software Craftsmanship for New Developers

TDD – Test Driven Development

Repetition of very short development cycle

Requirements turned into very specific test cases

Software is written only to pass new tests

Three Laws of TDD

You are not allowed to write any production code until you have first written a failing unit test

Three Laws of TDD

You are not allowed to write more of a unit test than is sufficient to fail – and not compiling is failing

Three Laws of TDD

You are not allowed to write more code that is sufficient to pass the currently failing unit test

Pair Programing

Two programmers work together at one workstation

Practicing – Coding Katas

Practice, Practice, Practice

Practice on *how* to solve the problem

Practicing – Coding Katas

Practice, Practice, Practice

Practice on *how* to solve the problem

Practicing – Coding Katas

Practice, Practice, Practice

Practice on *how* to solve the problem

- codingdojo.org/kata
- Codekata.com
- Codewars.com


Code Smells
Software Craftsmanship for New Developers

Code Smells

- Comments
- Environment
- Functions
- General
- Names
- Tests

Code Smells – Comments

Inappropriate Information

Code Smells – Comments

Obsolete Comment

Code Smells – Comments

Redundant Comment

```
i++ // increment i
```

Code Smells – Comments

Poorly Written Comment

Code Smells – Comments

Commented-Out Code

Code Smells – Environment

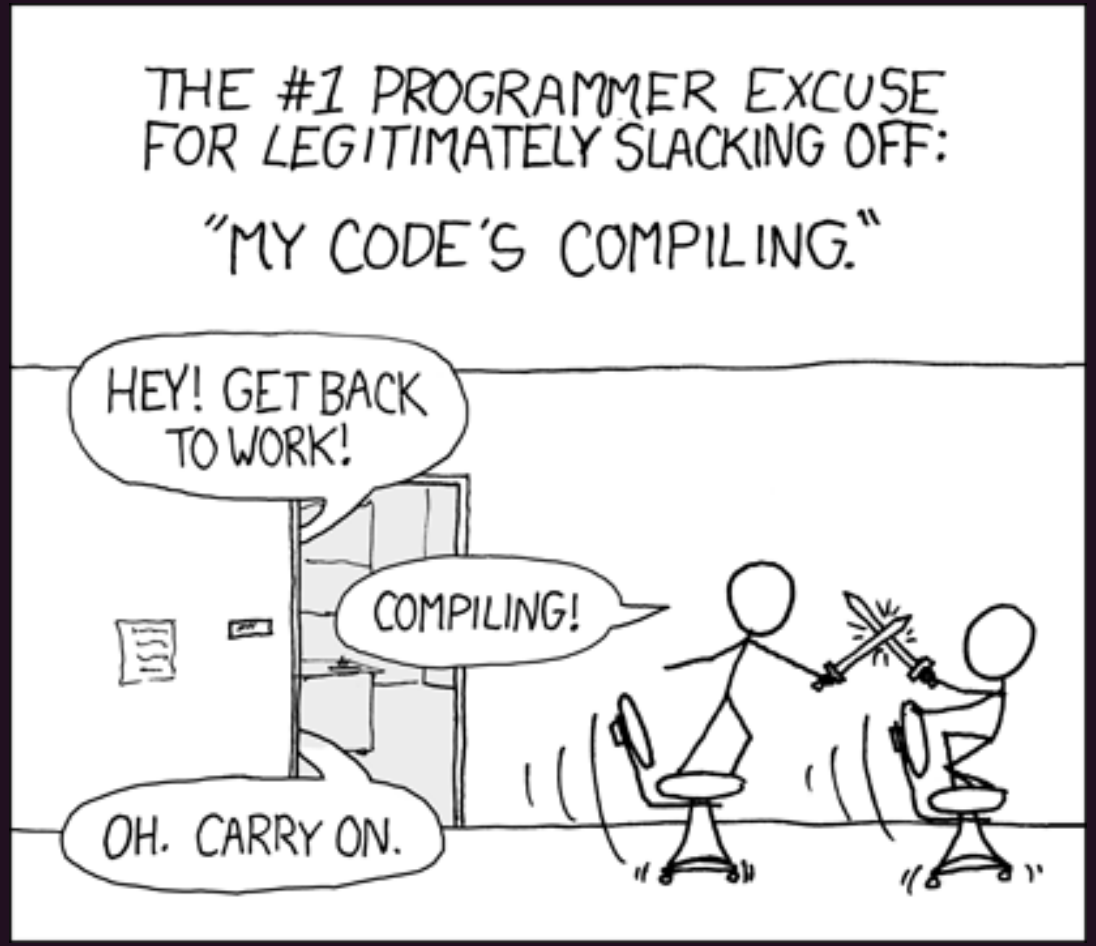
Build Requires More Than One Step

Code Smells – Environment

Build Requires More Than One Step

Code Smells – Environment

Build Requires More Than One Step



Code Smells – Environment

Tests Require More Than One Step

Code Smells – Function

Dead Function

Code Smells – General

Obvious Behavior is Unimplemented

Code Smells – General

Incorrect Behavior at the Boundaries

Code Smells – General

Overridden Safeties

Code Smells – General

Duplication (DRY)

Code Smells – General

Dead Code

Code Smells – General

Inconsistency

Code Smells – General

Clutter

Code Smells – General

Misplaced Responsibility

Code Smells – General

Function Names Should Say What They Do

```
DateTime newDate = date.add(5)
DateTime newDate = date.AddDays(5)
```

Code Smells – General

Not Following Standard Conventions

Code Smells – General

Replace Magic Numbers with Named Constants

3.14159265358979**3**

3.14159**7**53589793

Code Smells – General

Functions Doing More Than One Thing

Code Smells – Names

Undescriptive Names

```
CREATE PROCEDURE dbo.HII_Mobile_Cond_Workout_Activities_Log_View
  @ID INT = 0,
  @cond_workout_ID INT = 0
AS
BEGIN

  SELECT al.ID,
         al.cond_workout_ID,
         al.activity,
         al.mins,
         al.cal_burn,
         a.Category,
         ai.ID AS intensity_id,
         ai.Intensity
    FROM HII_Mobile_Cond_Workout_Activities_Log al
   INNER JOIN HII_Cond_Activities a            ON a.ID = al.activity
    LEFT JOIN HII_Cond_Activities_Intensity ai ON ai.Activity_ID = a.ID AND ai.ID=al.intensity
   WHERE al.active=1
     AND (cond_workout_ID = @cond_workout_ID OR al.Id = @ID)
   ORDER BY al.created_date

END
```

```
CREATE PROCEDURE dbo.HII_Mobile_Cond_Workout_Activities_Log_View
  @ID INT = 0,
  @cond_workout_ID INT = 0
AS
BEGIN

  SELECT al.ID,
         al.cond_workout_ID,
         al.activity,
         al.mins,
         al.cal_burn,
         a.Category,
         ai.ID AS intensity_id,
         ai.Intensity
    FROM HII_Mobile_Cond_Workout_Activities_Log al
   INNER JOIN HII_Cond_Activities a            ON a.ID = al.activity
    LEFT JOIN HII_Cond_Activities_Intensity ai ON ai.Activity_ID = a.ID AND ai.ID=al.intensity
   WHERE al.active=1
     AND (cond_workout_ID = @cond_workout_ID OR al.Id = @ID)
   ORDER BY al.created_date

END
```

```
CREATE PROCEDURE dbo.HII_Mobile_Cond_Workout_Activities_Log_View
  @ID INT = 0,
  @cond_workout_ID INT = 0
AS
BEGIN

  SELECT al.ID,
         al.cond_workout_ID,
         al.activity,
         al.mins,
         al.cal_burn,
         a.Category,
         ai.ID AS intensity_id,
         ai.Intensity
    FROM HII_Mobile_Cond_Workout_Activities_Log al
   INNER JOIN HII_Cond_Activities a            ON a.ID = al.activity
    LEFT JOIN HII_Cond_Activities_Intensity ai ON ai.Activity_ID = a.ID AND ai.ID=al.intensity
   WHERE al.active=1
     AND (cond_workout_ID = @cond_workout_ID OR al.Id = @ID)
   ORDER BY al.created_date

END
```

```
CREATE PROCEDURE dbo.GetWorkActivitiesLog
  @Id INT = 0,
  @WorkoutId INT = 0
AS
BEGIN

  SELECT ActivityLog.ID,
         ActivityLog.cond_workout_ID,
         ActivityLog.activity,
         ActivityLog.mins,
         ActivityLog.cal_burn,
         Activities.Category,
         Intensity.ID AS intensity_id,
         Intensity.Intensity
    FROM HII_Mobile_Cond_Workout_Activities_Log AS ActivityLog
   INNER JOIN HII_Cond_Activities AS Activities ON a.ID = ActivityLog.activity
    LEFT JOIN HII_Cond_Activities_Intensity Intensity
         ON Intensity.Activity_ID = Activities.ID AND Intensity.ID=ActivityLog.intensity
   WHERE ActivityLog.active=1
     AND (ActivityLog.cond_workout_ID = @WorkoutId OR ActivtyLog.Id = @ID)
   ORDER BY ActivityLog.created_date

END
```

```
CREATE PROCEDURE dbo.GetWorkActivitiesLog
  @Id INT = 0,
  @WorkoutId INT = 0
AS
BEGIN

  SELECT ActivityLog.ID,
         ActivityLog.cond_workout_ID,
         ActivityLog.activity,
         ActivityLog.mins,
         ActivityLog.cal_burn,
         Activities.Category,
         Intensity.ID AS intensity_id,
         Intensity.Intensity
    FROM HII_Mobile_Cond_Workout_Activities_Log AS ActivityLog
   INNER JOIN HII_Cond_Activities AS Activities ON a.ID = ActivityLog.activity
    LEFT JOIN HII_Cond_Activities_Intensity Intensity
         ON Intensity.Activity_ID = Activities.ID AND Intensity.ID=ActivityLog.intensity
   WHERE ActivityLog.active=1
     AND (ActivityLog.cond_workout_ID = @WorkoutId OR ActivtyLog.Id = @ID)
   ORDER BY ActivityLog.created_date

END
```

Code Smells – Names

Encoded Names

intRepeat

repeatCount

Code Smells – Names

Names Not Describing Side-Effects

```
public Foo GetFoo() { }
public Foo CreateAndGetFoo() { }
public Foo Create () { }
```

Code Smells – Tests

Insufficient Tests

Code Smells – Tests

Not Using a Test Coverage Tool

Code Smells – Tests

Skipping Trivial Tests

Code Smells – Tests

Not Testing Boundary Conditions

Code Smells – Tests

Exhaustively Test Near Bugs

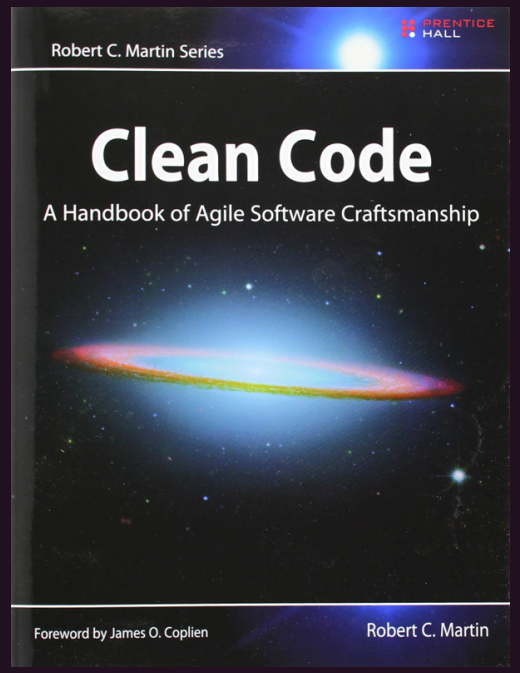
Code Smells – Tests

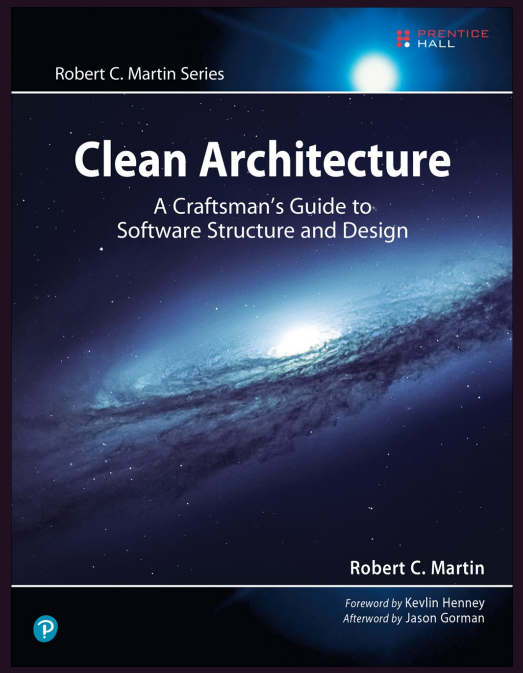
Tests Should be Fast

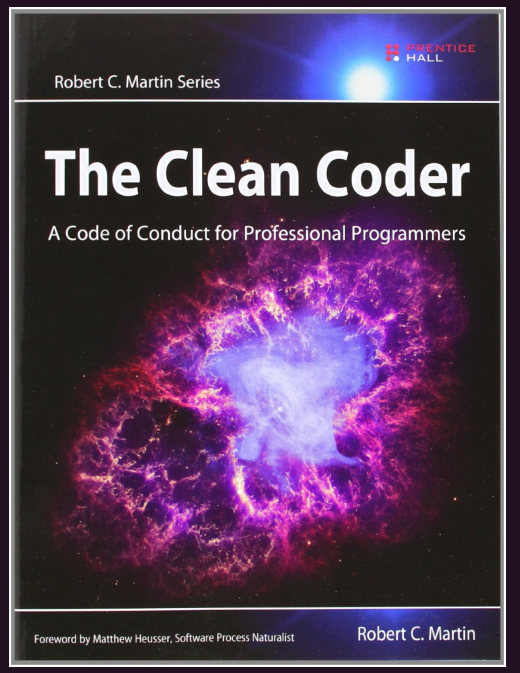
Getting More

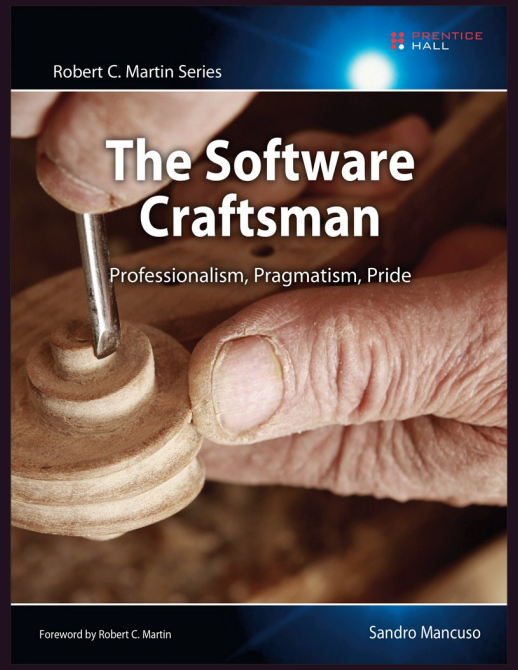
Software Craftsmanship for New Developers

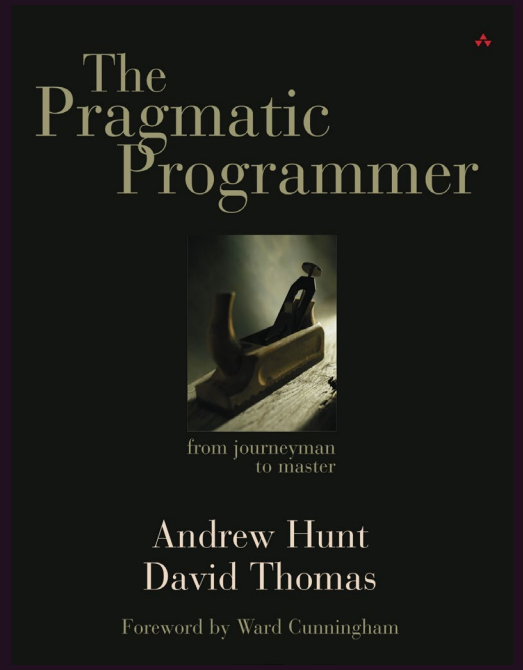
Books

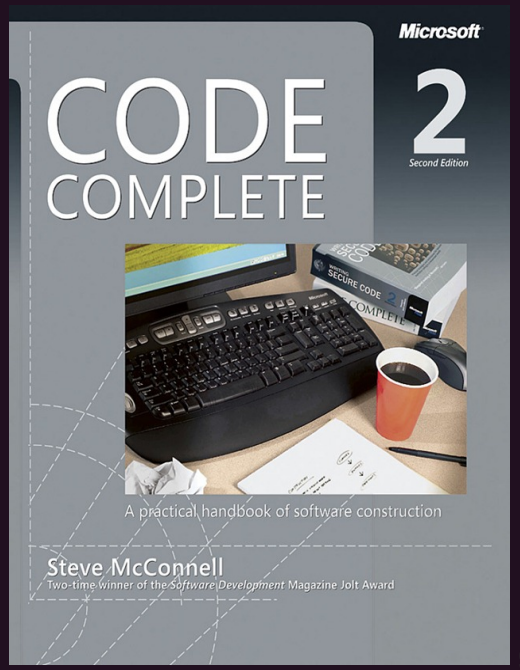

Robert C. Martin Series
PRENTICE HALL
Clean Code
A Handbook of Agile Software Craftsmanship
Foreword by James O. Coplien
Robert C. Martin


Robert C. Martin Series
PRENTICE HALL
Clean Architecture
A Craftsman's Guide to
Software Structure and Design
Robert C. Martin
Foreword by Kevlin Henney
Afterword by Jason Gorman

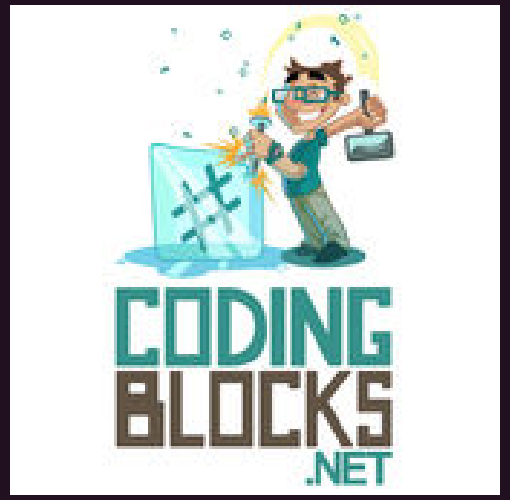

Robert C. Martin Series
PRENTICE HALL
The Clean Coder
A Code of Conduct for Professional Programmers
Foreword by Matthew Heusser, Software Process Naturalist
Robert C. Martin

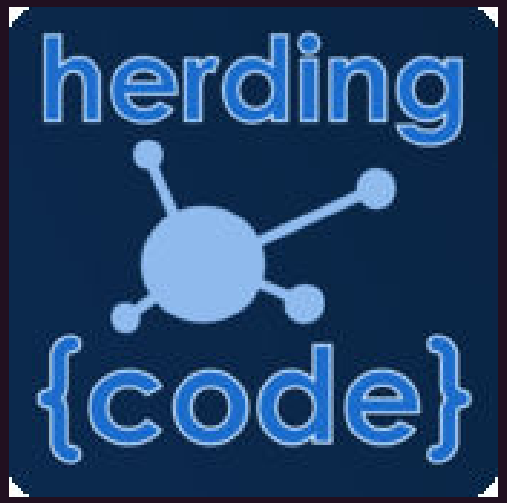

Robert C. Martin Series
PRENTICE HALL
The Software Craftsman
Professionalism, Pragmatism, Pride
Foreword by Robert C. Martin
Sandro Mancuso


The Pragmatic Programmer
from journeyman to master
Andrew Hunt
David Thomas
Foreword by Ward Cunningham


Microsoft
CODE COMPLETE
2
Second Edition
A practical handbook of software construction
Steve McConnell
Two-time winner of the Software Development Magazine Jolt Award

Podcasts


CODING
BLOCKS
.NET


herding
{code}


THE
InfoQ
PODCAST


developer
tea


.NET Rocks!

Live Coding

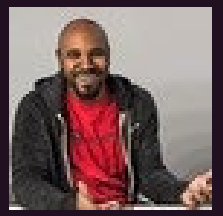
MicrosoftDeveloper
VlsualStudio

425show

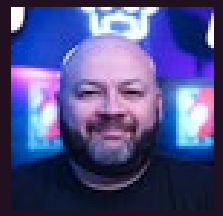
CodeItLive

THE LIVE CODERS


ardalis


CLDubya

CodingGarden


BaldBeardedBuilder


CSharpFritz


TaleLearnCode

Meetups


Microsoft Data Platform Continuity Virtual Group
51 Members


agileLNK
1,045 Agilists


Ruby and Open Source Meetup, Lincoln, Nebraska, USA
200 Members


Omaha Agile Development
1,969 Members


GAAD
Nebraska Digital Accessibility Meetup
189 Members

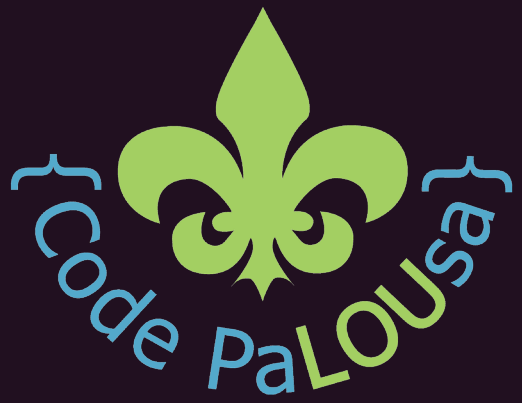

Audacious Agile Conversations With Lean Coffee in Omaha
785 Members

Virtual/Hybrid Meetups

Conferences

Nebraska.Code() — July 13-15, 2021

Prairie.Code() — September 23-24, 2021



Code PaLOUsa — August 18 – 20, 2021

Software Craftsmanship is about professionalism in software development.

Thank You

chadgreen@chadgreen.com

TaleLearnCode

ChadGreen.com

ChadGreen & TaleLearnCode

ChadwickEGreen