Refactoring

Jason Gorman

© Codemanship Ltd 2025

Schedule

09:45	Zoom Meeting Starts
10:00	Training Begins (prompt)
11:30	Break
11:45	Training resumes
13:00	Lunch
14:00	Training resumes (prompt)
15:30	Break
15:45	Training resumes
17:00	End

© Codemanship Ltd 2025

What To Expect Today

- Introduction to Refactoring
- The Refactoring Discipline
- Code Smells
    - Long Methods
    - Comments
    - Duplicate Code
    - Divergent Change & Large Classes
    - Message Chains
    - Long Parameter Lists
    - Data Clumps

© Codemanship Ltd 2025

codemanship

What To Expect Tomorrow

- More Code Smells
  - Primitive Obsession
  - Feature Envy & Data Classes
  - Switch Statements
- Legacy Code
- Refactoring Metrics
  - Code Quality Metrics
  - Software Delivery metrics
- Long-Form Refactoring
  - Refactoring Golf

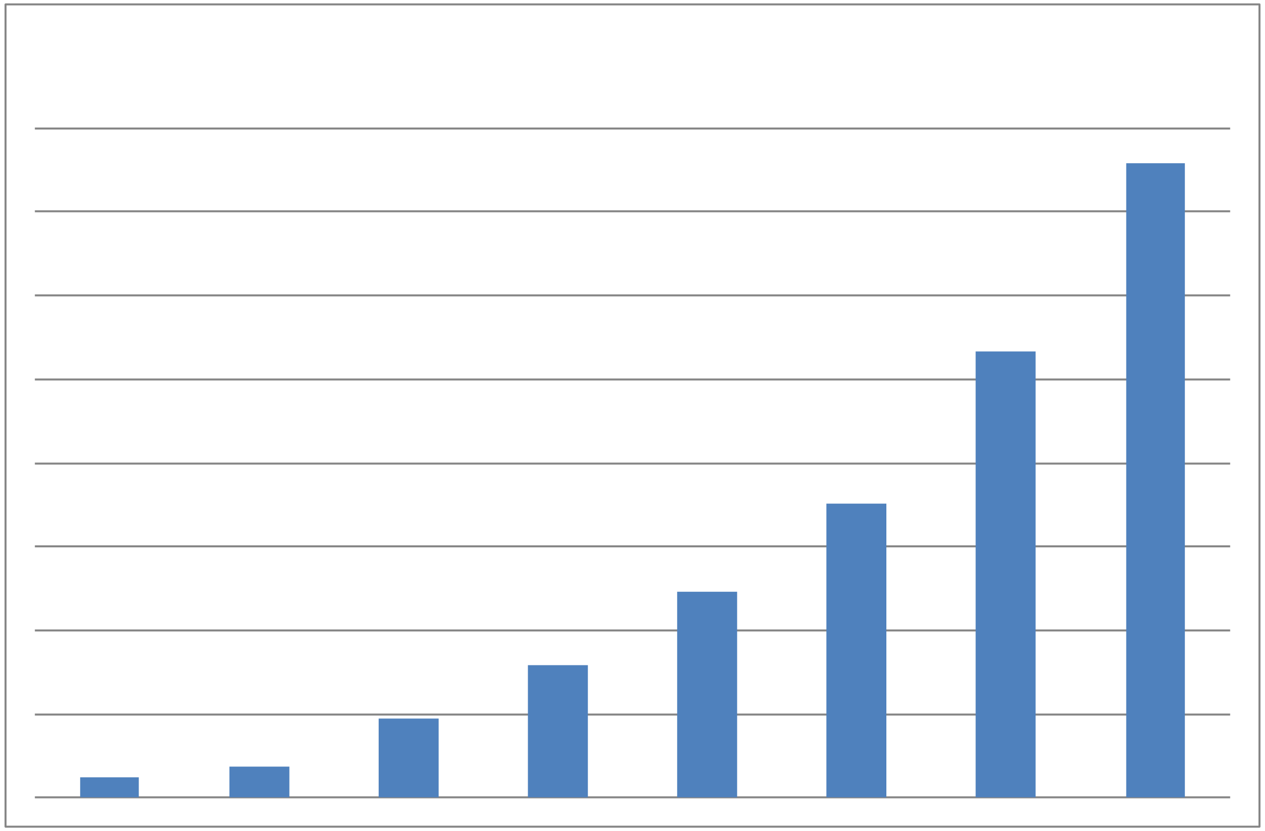
© Codemanship Ltd 2025

Introduction To Refactoring

© Codemanship Ltd 2025

codemanship

Delivery Lead Times



© Codemanship Ltd 2025

codemanship

What Makes Code Harder To Change?

© Codemanship Ltd 2025

codemanship

Readability

Complexity

Duplication

codemanship

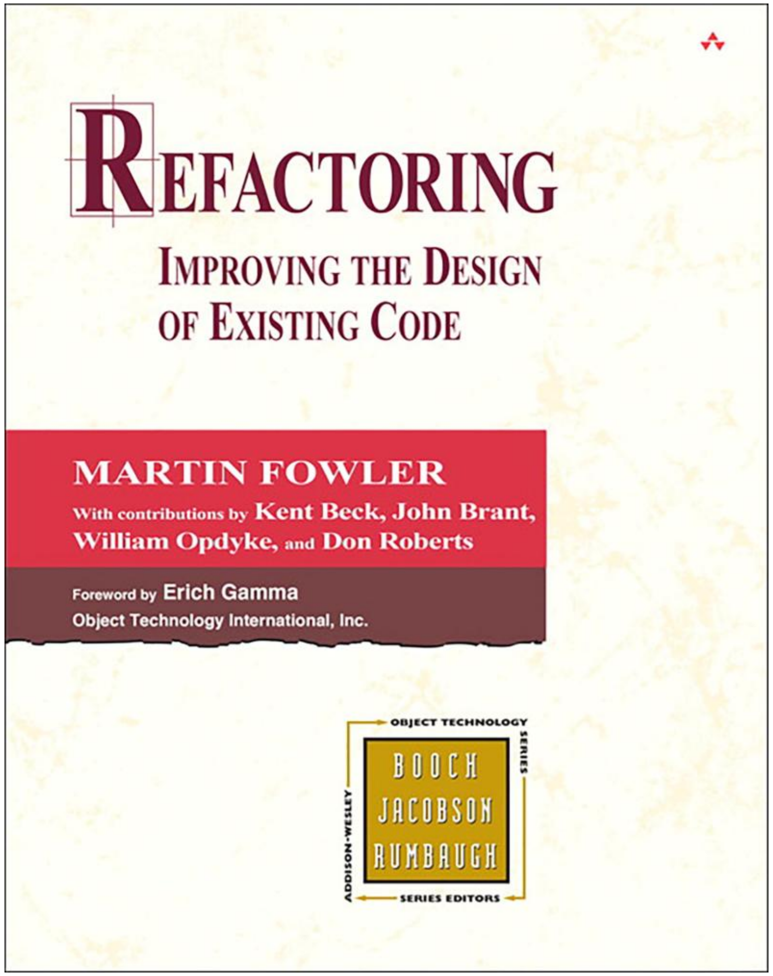
Dependencies & The “Ripple Effect”

Refactoring

Improving The Internal Design Of Software To Make It Easier To Change

© Codemanship Ltd 2025

codemanship



REFACTORING

IMPROVING THE DESIGN OF EXISTING CODE

MARTIN FOWLER

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

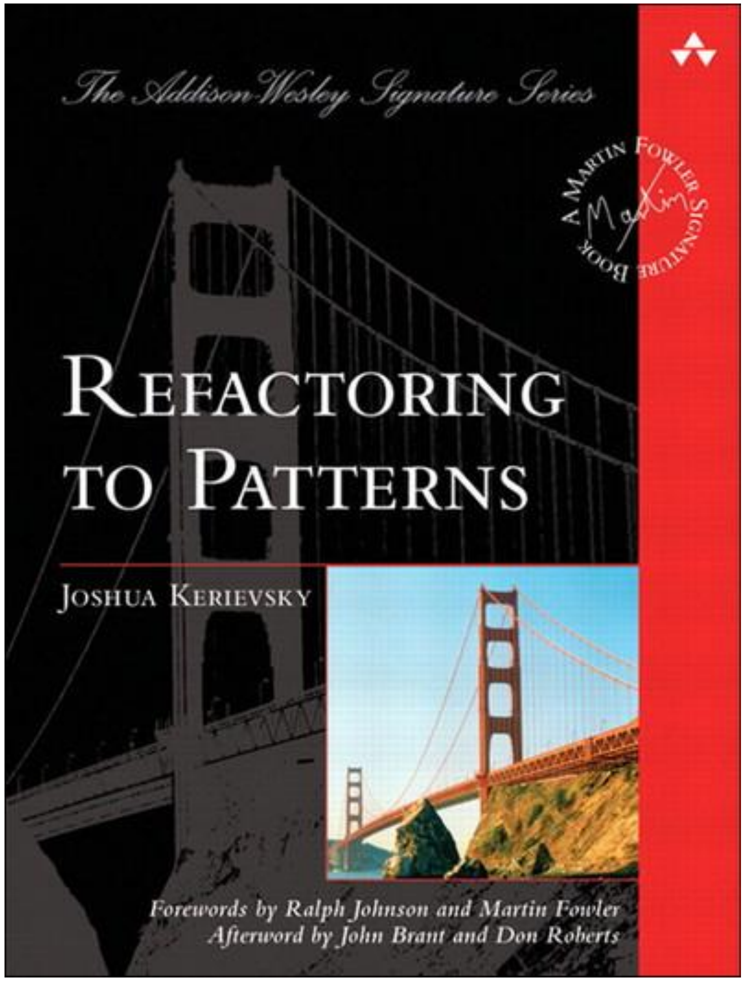
Foreword by Erich Gamma
Object Technology International, Inc.

OBJECT TECHNOLOGY SERIES

BOOCH JACOBSON RUMBAUGH

SERIES EDITORS

ADDISON-WESLEY



The Addison Wesley Signature Series

A MARTIN FOWLER SIGNATURE BOOK

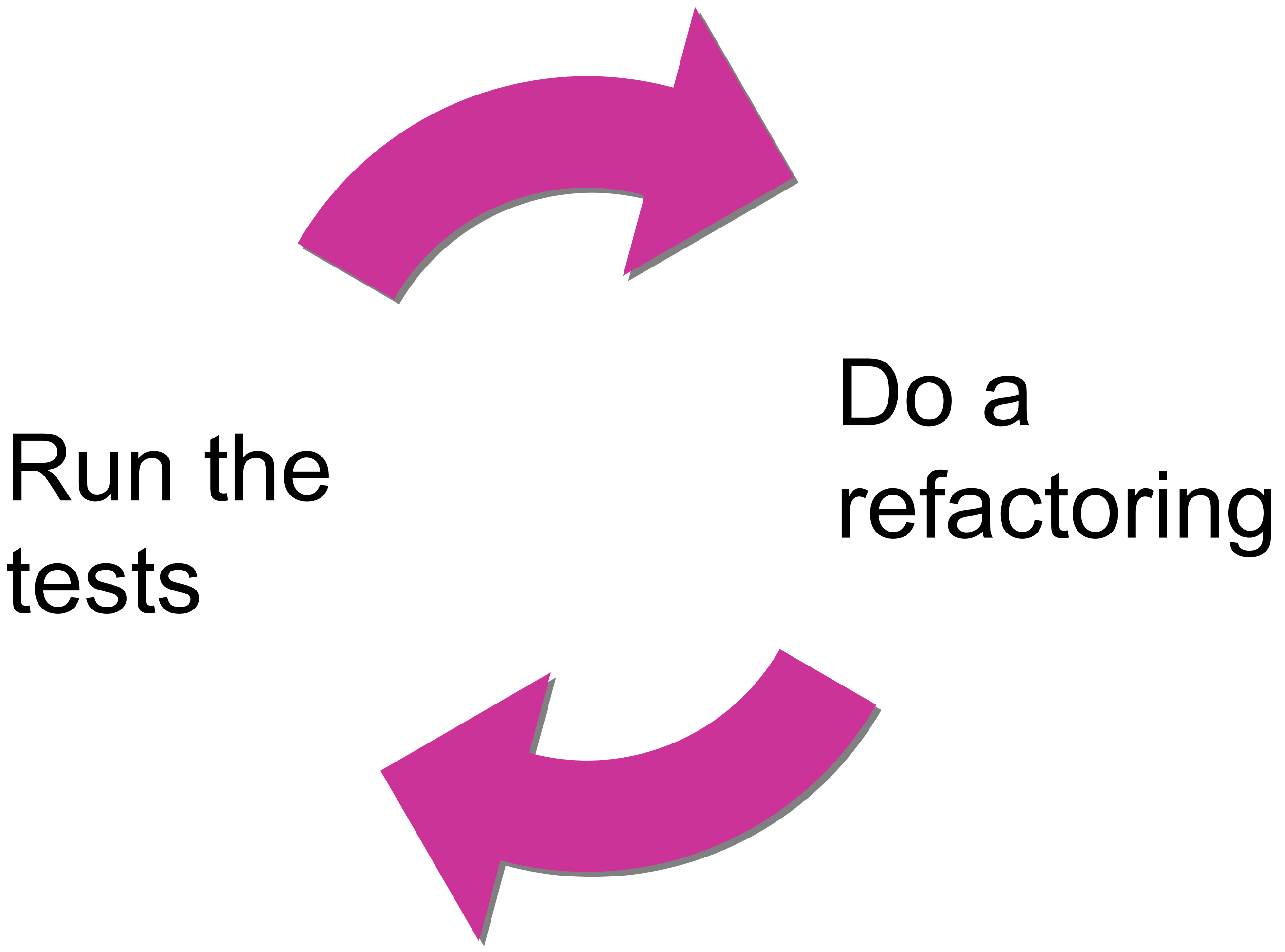
REFACTORING TO PATTERNS

JOSHUA KERIEVSKY

Forewords by Ralph Johnson and Martin Fowler
Afterword by John Brant and Don Roberts

© Codemanship Ltd 2025

Refactoring Process



codemanship

© Codemanship Ltd 2025

Refactoring Habits

- Do one refactoring at a time
- Run the tests after every refactoring
- Refactor directly to well-defined goals
- Commit on green, revert on red
- **Never** refactor on a red light
- Use *predictable* automated refactorings whenever possible
- If sufficient tests don't exist, write them before refactoring

© Codemanship Ltd 2025

codemanship

© Codemanship Ltd 2025

{ }
codemanship

test && commit || revert

```
git add .
dotnet test && git commit -m "Working commit" || git reset --hard
```

© Codemanship Ltd 2025

codemanship

Code Smells

- …are indications of *increasing entropy* in your code
  - Increasing complexity
  - Increasing duplication
  - Increasing dependency issues
  - Decreasing comprehensibility
- As time goes on, code can become *rigid* and *brittle*

© Codemanship Ltd 2025

Classes Of Code Smell

Complexity

- **Long Method**
- **Large Class**
- **Primitive Obsession**
- Data Clumps
- **Long Parameter Lists**

Responsibility Problems

- **Divergent Change**
- Shotgun Surgery
- **Data Class**

Couplers

- **Feature Envy**
- **Message Chains**
- Inappropriate Intimacy
- Middle Man

OO Abuses

- **Switch Statements**
- Temporary Field
- Refused Bequest
- Alternative Classes With Different Interfaces
- Parallel Inheritance Heirarchies

Redundancy

- Lazy Class
- **Duplicate Code**
- Dead Code
- Speculative Generality

© Codemanship Ltd 2025

codemanship

Primitive Refactorings ("Short-Form")

	Applies To				
Rename	Any identifier				
Safe Delete	Any declaration or file				
Move	Method/Function	Field	Type	File	
Extract Method	Expression	Statement/Block			
Extract Class	Class				
Extract Superclass	Class				
Extract Interface	Class				
Change Signature	Method/Function				
Introduce Parameter	Expression				
Introduce Parameter Object*	Method				
Introduce Variable	Expression				
Introduce Field	Expression				
Inline	Class	Field	Method	Variable	Parameter

© Codemanship Ltd 2025

* *Transform Parameters* in Rider

Composite Refactorings ("Long-Form")

- Replace Method with Method Object
- Replace Conditional With Polymorphism
  - Replace Type Code With Strategy
- Replace Magic Literals With Enum
- Replace Parameter with Method
- Replace Data Value with Object
- Replace Array With Object
- Substitute Collaborator
- Hide Delegate
- Decompose Conditional
- Preserve Whole Object

© Codemanship Ltd 2025

* *Transform Parameters* in Rider

Get The Source Code

From GitHub: https://github.com/jasongorman/csharp_code_smells

As ZIP: https://codemanship.co.uk/files/csharp_code_smells.zip

© Codemanship Ltd 2025

Long Methods

- Extract Method
- Decompose Conditional
- Introduce Field

© Codemanship Ltd 2025

Comments

- Rename Method/Variable/Class etc
- Extract Method, Class
- Introduce Variable, Field (inc. constant)
- Replace Magic Literals With Enum

© Codemanship Ltd 2025

codemanship

Duplicate Code

- Extract Method
- Extract Class
- Extract Superclass

© Codemanship Ltd 2025

Divergent Change/Large Class

- Extract Class
- Replace Method With Method Object

Message Chains

- Hide Delegate
  - Extract Method
  - Move Method
- Substitute Collaborator

© Codemanship Ltd 2025

Long Parameter Lists

- Replace Parameter with Method
- Introduce Parameter Object
- Preserve Whole Object

© Codemanship Ltd 2010

codemanship

Primitive Obsession

- Replace Data Value With Object
- Replace Array With Object

© Codemanship Ltd 2025

Feature Envy

- Extract Method
- Move Method
- Move Field

© Codemanship Ltd 2010

Data Classes

- Move Method
- Move Field

© Codemanship Ltd 2010

Refactoring To Patterns

© Codemanship Ltd 2010

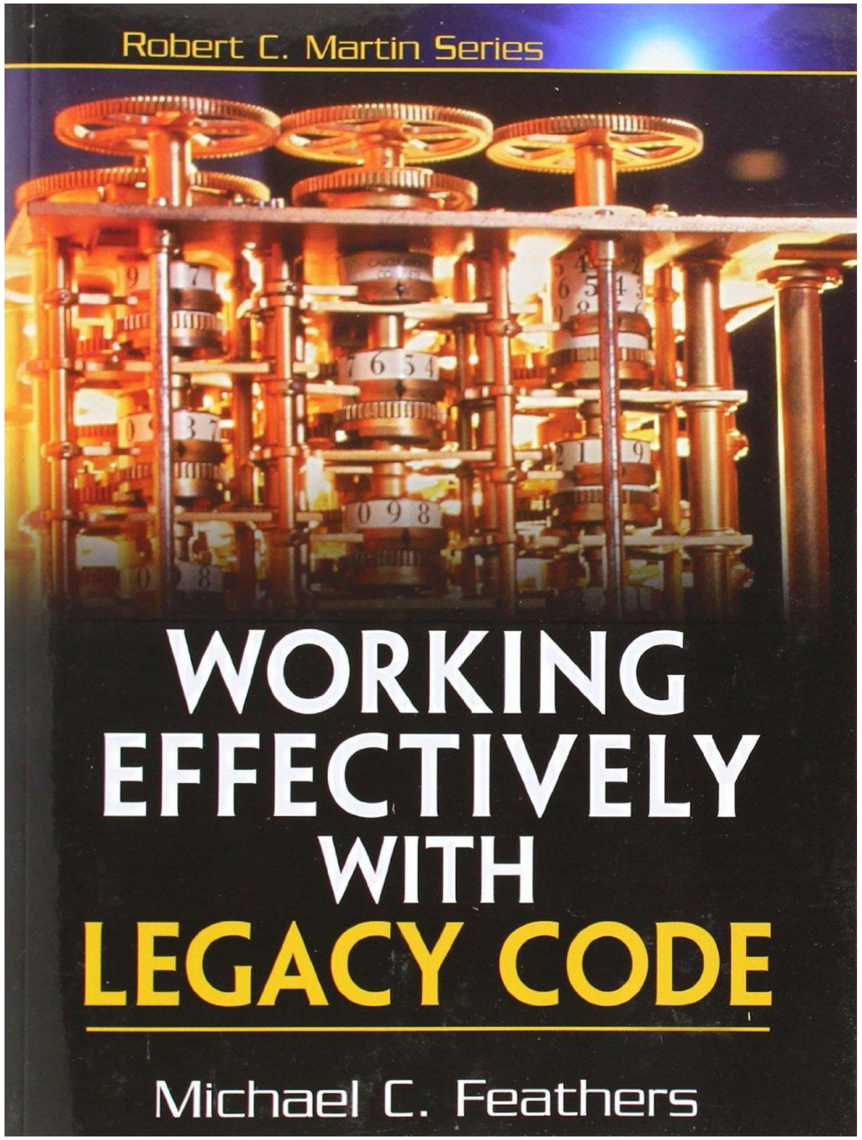
codemanship

Switch Statements

- Replace Type Code With Strategy

© Codemanship Ltd 2010

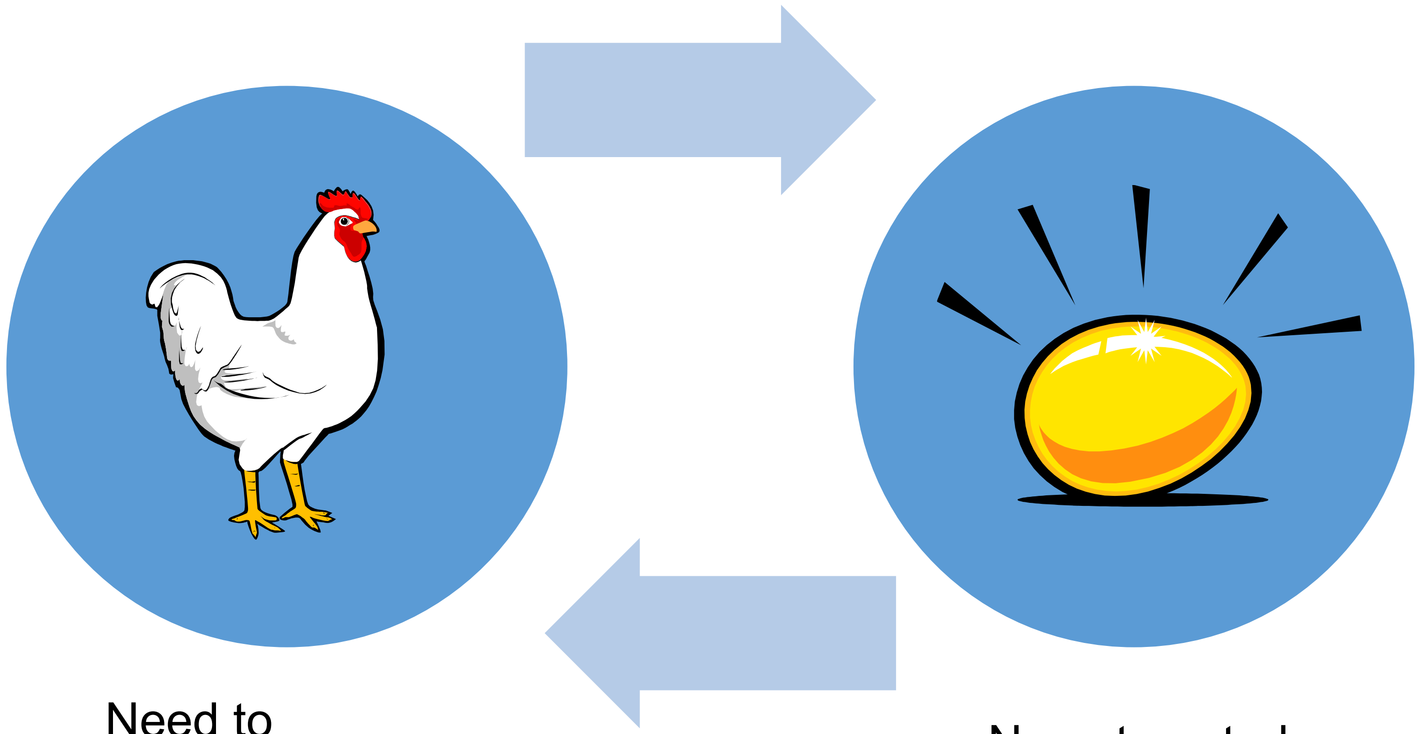
TDD & Legacy Code



© Codemanship Ltd 2025

codemanship

Catch 22



Need to refactor to make code unit testable

No automated tests to support refactoring

© Codemanship Ltd 2025

codemanship

5-Step Process

1. Identify What Code Needs To Change
2. Refactor Code To Make It Unit-Testable
3. Write Unit Tests
4. Make The Change
5. Apply The “Boy Scout Rule”

© Codemanship Ltd 2025

Refactoring Metrics

© Codemanship Ltd 2025

Code Metrics

Indicators of Maintainability

© Codemanship Ltd 2025

Dimensions of Maintainability

- Complexity
- Coupling & Cohesion
- Duplication
- Readability

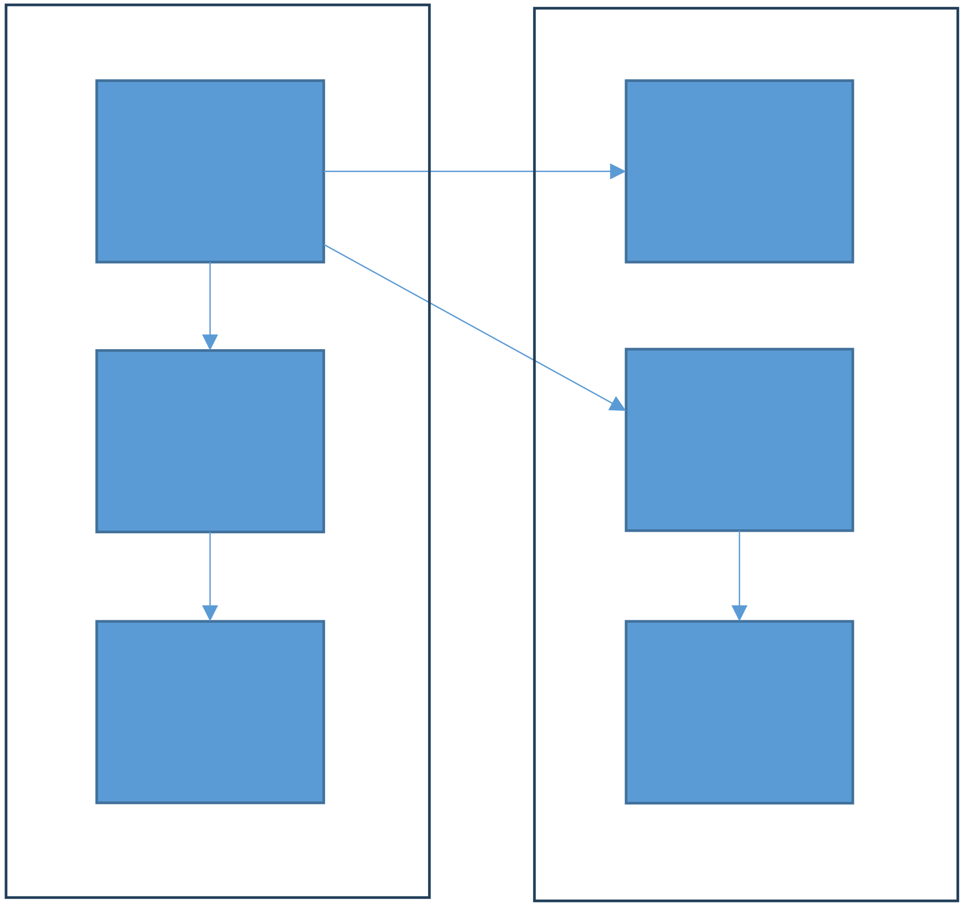
© Codemanship Ltd 2025

Code Quality Metrics - Complexity

- Lines of code
- Cyclomatic complexity
- No. of Methods, Classes etc.
- Halstead metrics
- Maintainability Index

© Codemanship Ltd 2025

Code Quality Metrics – Coupling & Cohesion



Various Static Analysis Tools e.g., NDepend

© Codemanship Ltd 2025

codemanship

Code Quality Metrics - Duplication

```
public void Credit(float amount)
{
    balance += amount;
    transactions.Add(new Transaction(true, amount));
    DateTime now = DateTime.Now;
    lastTransactionDate = now.Date + "/" +
                          now.Month + "/" +
                          now.Year;
}

public void Debit(float amount)
{
    balance -= amount;
    transactions.Add(new Transaction(true, -amount));
    DateTime now = DateTime.Now;
    lastTransactionDate = now.Date + "/" +
                          now.Month + "/" +
                          now.Year;
}
```

Code Similarity Finder
https://simian.quandarypeak.com/

© Codemanship Ltd 2025

codemanship

Code Quality Metrics – Readability Testing

```
int[] numbers = {2,0,7,13,3,2,9,15,4}
var min = Int32.MaxValue;
     for (var i = 0; i < numbers.length; i++) {
          int number = numbers[i];
          if (number < min)
               min = number;
  }
  Console.WriteLine(min);
```

What will the output of this code be?

© Codemanship Ltd 2025

Complexity As A (Rough) Indicator Of Readability

- Halstead Difficulty
- Cyclomatic Complexity

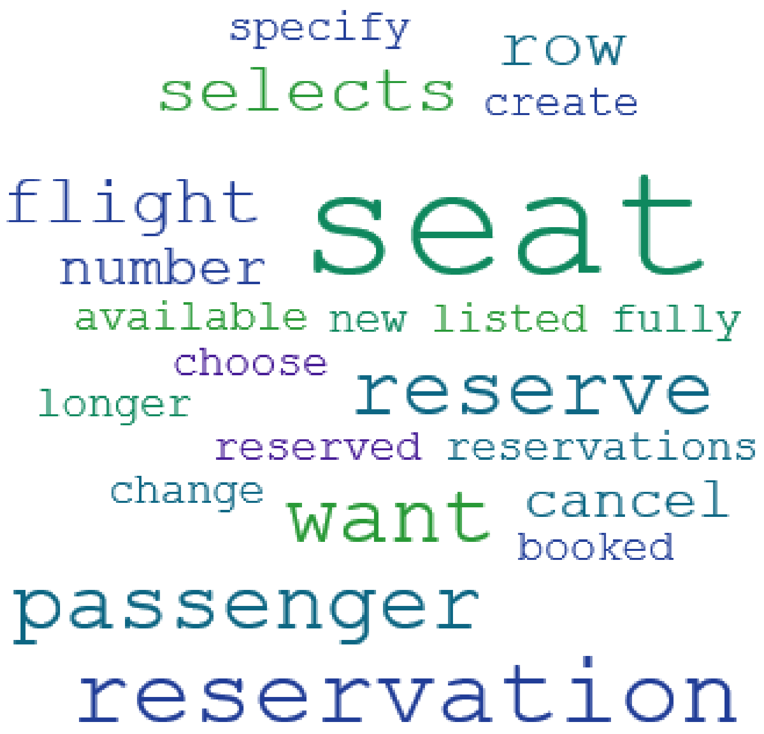
© Codemanship Ltd 2025

codemanship

Conceptual Correlation

```
public class PlaceRepositoryTests {

  @Test
  public void allocateFlagsPlaceForUser() {
    PlaceRepository placeRepository =
                        new PlaceRepository();
    User user = new User();
    Place place =
          placeRepository.allocate("A", 1, user);
    assertEquals(user, place.flaggedFor());
  }

}
```

```
public class FlightSeatingTests {

  @Test
  public void seatIsReservedForPassenger() {
    FlightSeating seating = new FlightSeating();
    Passenger passenger = new Passenger();
    SeatReservation reservation
            = seating.reserve("A", 1, passenger);
    assertEquals(passenger,
                     reservation.getPassenger());
  }

}
```



.NET Standard POC
https://github.com/jasongorman/Conceptual

© Codemanship Ltd 2025

Automated Code Smell Detection

- Roslyn Code Analysis
- SonarQube (C# plugin)
- NDepend

© Codemanship Ltd 2025

codemanship

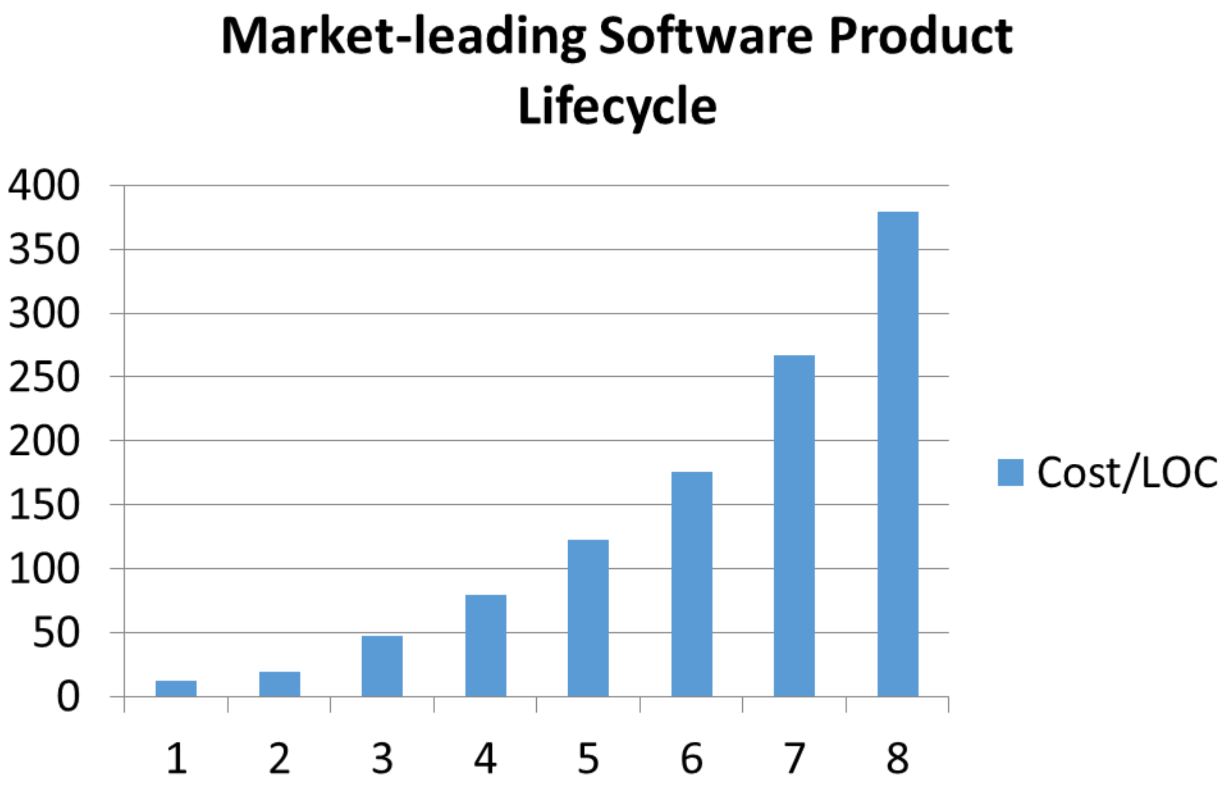
Software Delivery Metrics

The Pay-Off For Refactoring

© Codemanship Ltd 2025

Cost of Change

- Cost of adding, modifying or deleting a line of code
  - Total development cost (e.g., per week) / code churn (e.g., per week)



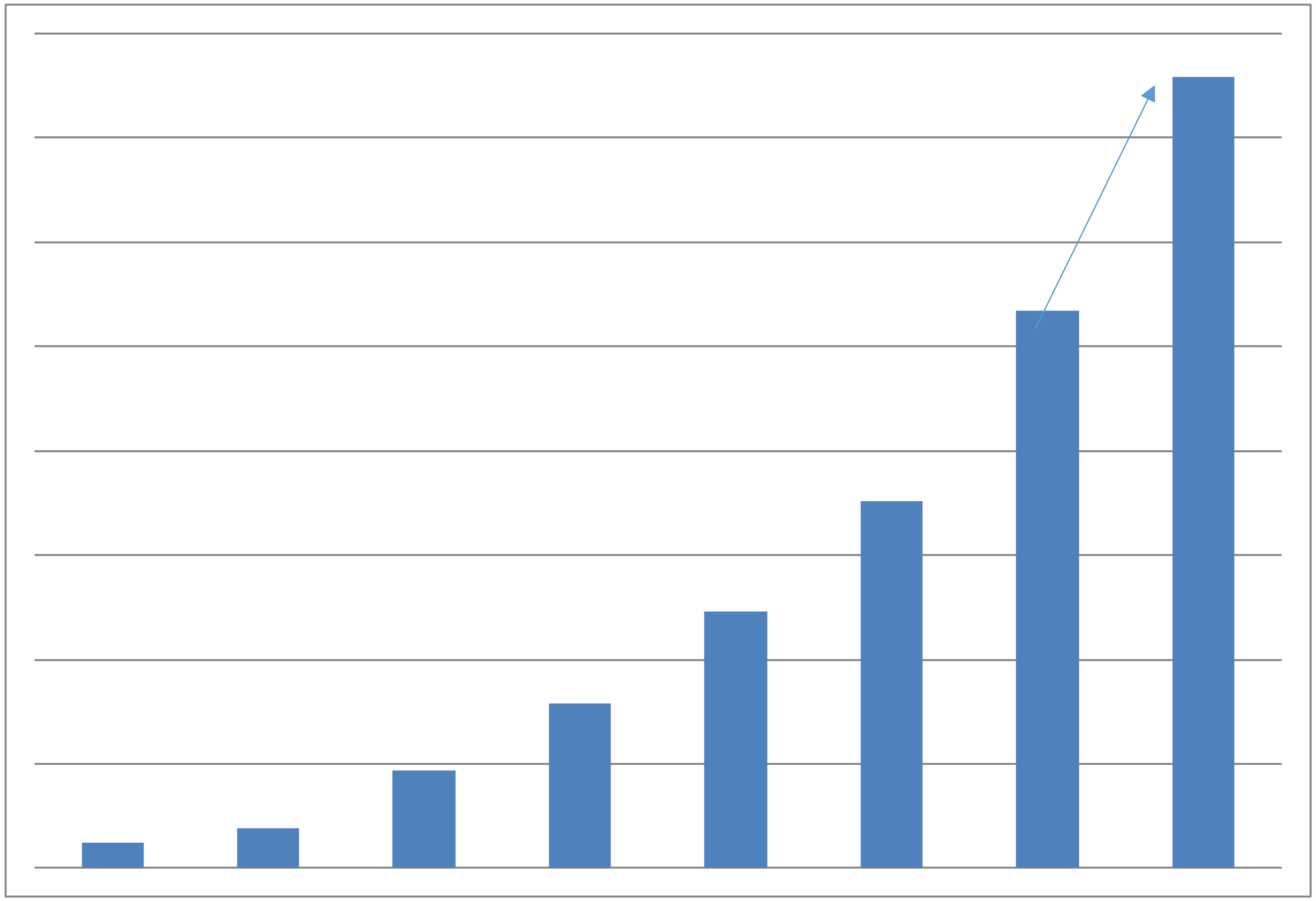
© Codemanship Ltd 2025

codemanship

Sustainability of Delivery

- Power Law distribution of lead time over time

$$\frac{52}{44} = 1.18$$



codemanship

Business Metrics – Delivery Lead Time

- Elapsed time from a feature being requested to a working implementation available to end users
  - Agile teams (Continuous Delivery) – hours -> weeks
  - Waterfall teams (“Big bang” rollouts) – months -> years

© Codemanship Ltd 2025

codemanship

Business Metrics – Delivery Lead Time

- Elapsed time from a feature being requested to a working implementation available to end users
  - Agile teams (Continuous Delivery) – hours -> weeks
  - Waterfall teams (“Big bang” rollouts) – months -> years

© Codemanship Ltd 2025

{} codemanship

Refactoring Golf

codemanship

The Game

```
public void debit(float amount) {

      // deduct amount from balance
      balance -= amount;

      // record transaction
      transactions.add(new Transaction(true,
          amount));

      // update last debit date
      Calendar calendar = Calendar.getInstance();

      lastDebitDate = calendar.get(Calendar.DATE)
          + "/" +
      calendar.get(Calendar.MONTH) + "/" +
      calendar.get(Calendar.YEAR);
}
```

```
public void debit(float amount) {
      deductAmountFromBalance(amount);
      recordTransaction(amount);
      updateLastDebitDate();
}
```

© Codemanship Ltd 2015

codemanship

Scoring (Per Move)

- 1 point – for each automated refactoring
- 1 point – for each cut & paste
- 1 point – for any code edit done using shortcuts (e.g. CTRL+F)
- 0 points – for code formatting (e.g., deleting a blank line)

PENALTIES

- 2 points – for every change to a line of code done manually
- x2 – for any move made while the code can't pass the tests
- 2 points – for copying and pasting code

© Codemanship Ltd 2015

Get The Code

https://github.com/jasongorman/CSharp_RefactoringGolf

© Codemanship Ltd 2025

codemanship

codemanship

www.codemanship.co.uk

© Codemanship Ltd 2025