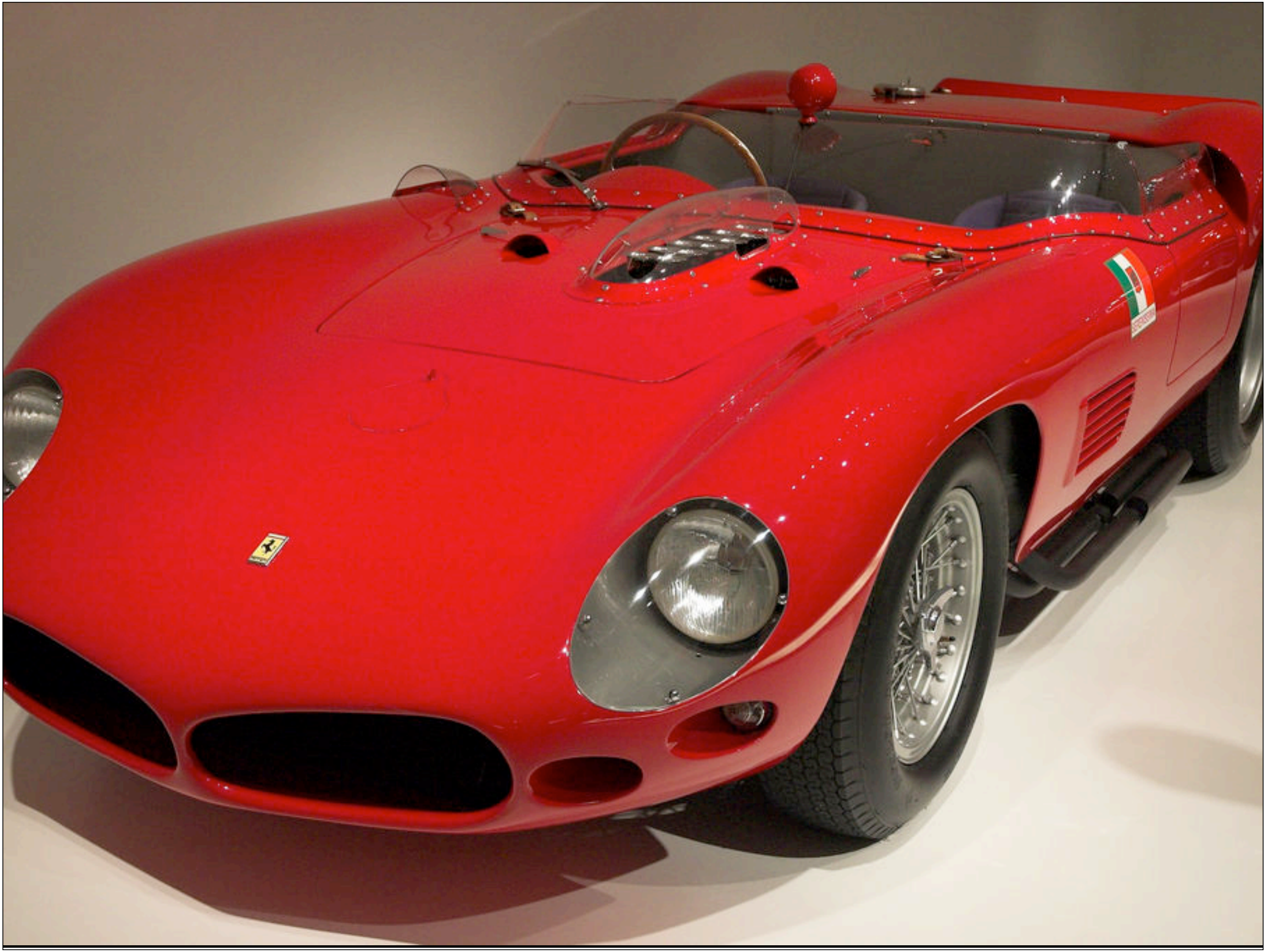
Refactoring

What is **Refactoring**?

Refactoring is a disciplined technique for restructuring an existing body of code, altering its internal structure without changing its external behavior.

Martin Fowler
Refactoring, improving design of existing code



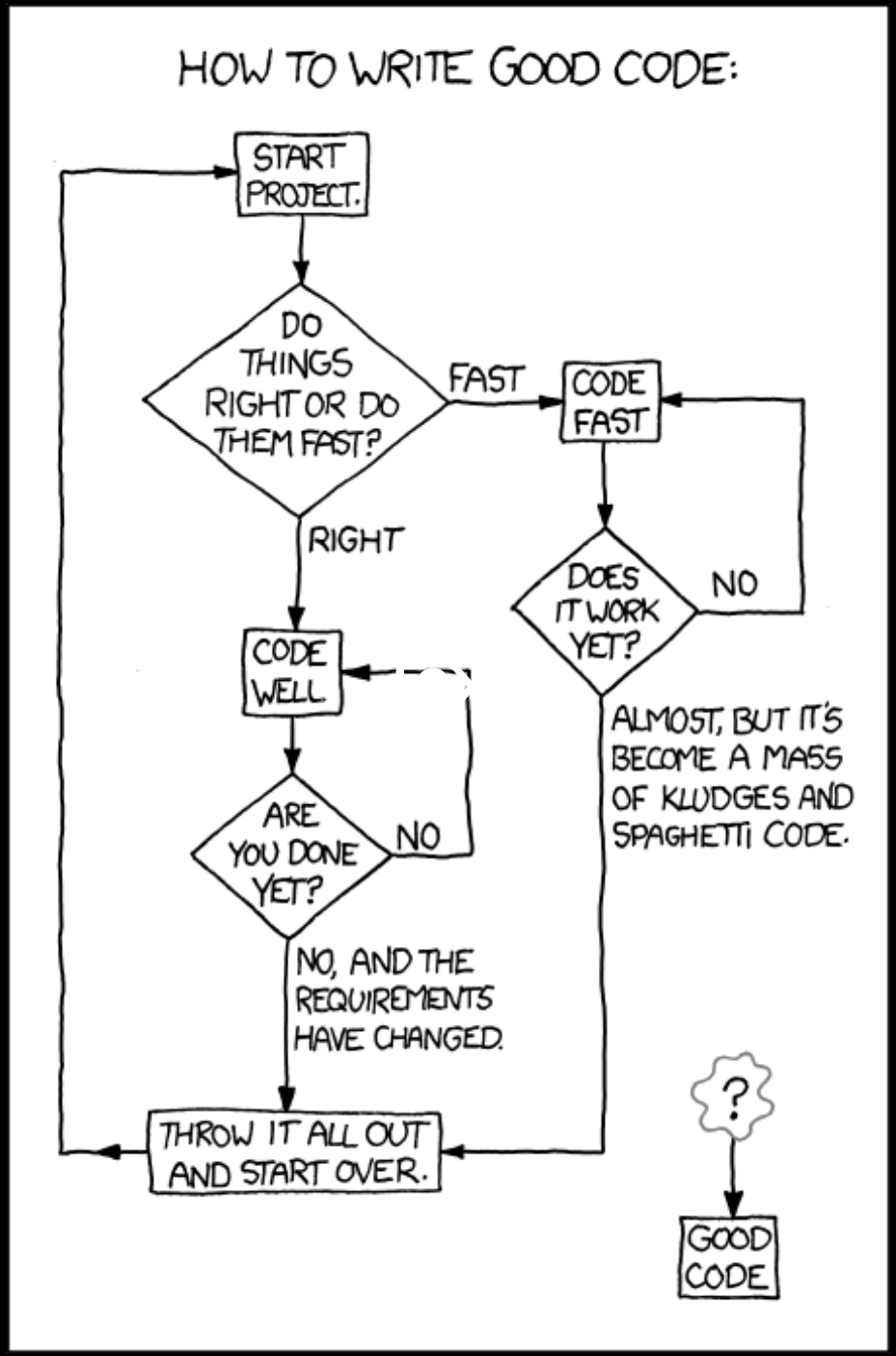
What happened?

Sharepoint?

Lack of time?

Lack of skill?

Poor management?


HOW TO WRITE GOOD CODE:
START PROJECT.
DO THINGS RIGHT OR DO THEM FAST?
FAST
CODE FAST
RIGHT
DOES IT WORK YET?
NO
CODE WELL.
ARE YOU DONE YET?
NO
ALMOST, BUT IT'S BECOME A MASS OF KLUDGES AND SPAGHETTI CODE.
NO, AND THE REQUIREMENTS HAVE CHANGED.
?
THROW IT ALL OUT AND START OVER.
GOOD CODE

Why **Refactor**?

Any fool can write code that a computer can understand. Good programmers write code that humans can understand

Martin Fowler

Remove duplication
Remove duplication
Remove duplication

Make code
maintainable

Make code readable (by humans)

Spaghetti Code

Technical Debt

Improve Design

Reduce BLOC
(beers x LOC)

When to **Refactor**?

Fixing bugs

Adding features

Code Review

Manager on Vacation?

100% sure the new code is better

Cost **Not to** Refactor

greater than

Cost to Refactor

Cost = Doing the change

\+ Testing it

\+ Documentation

Risk of
Introducing **bugs**


I DON'T ALWAYS TEST MY CODE
BUT WHEN I DO I DO IT IN PRODUCTION
memegenerator.net


STOP

Only refactor if you are confident

(it works as before, no side effects)

Unit Tests

One class
One method

No dependencies
(mocks)

Hard to do
with **Legacy Code**

Core of TDD

Integration Tests

More than one class

Communication between components

Acceptance tests

Black box testing

End to end

Given **input**
When doing **YYY**
Expect **output**

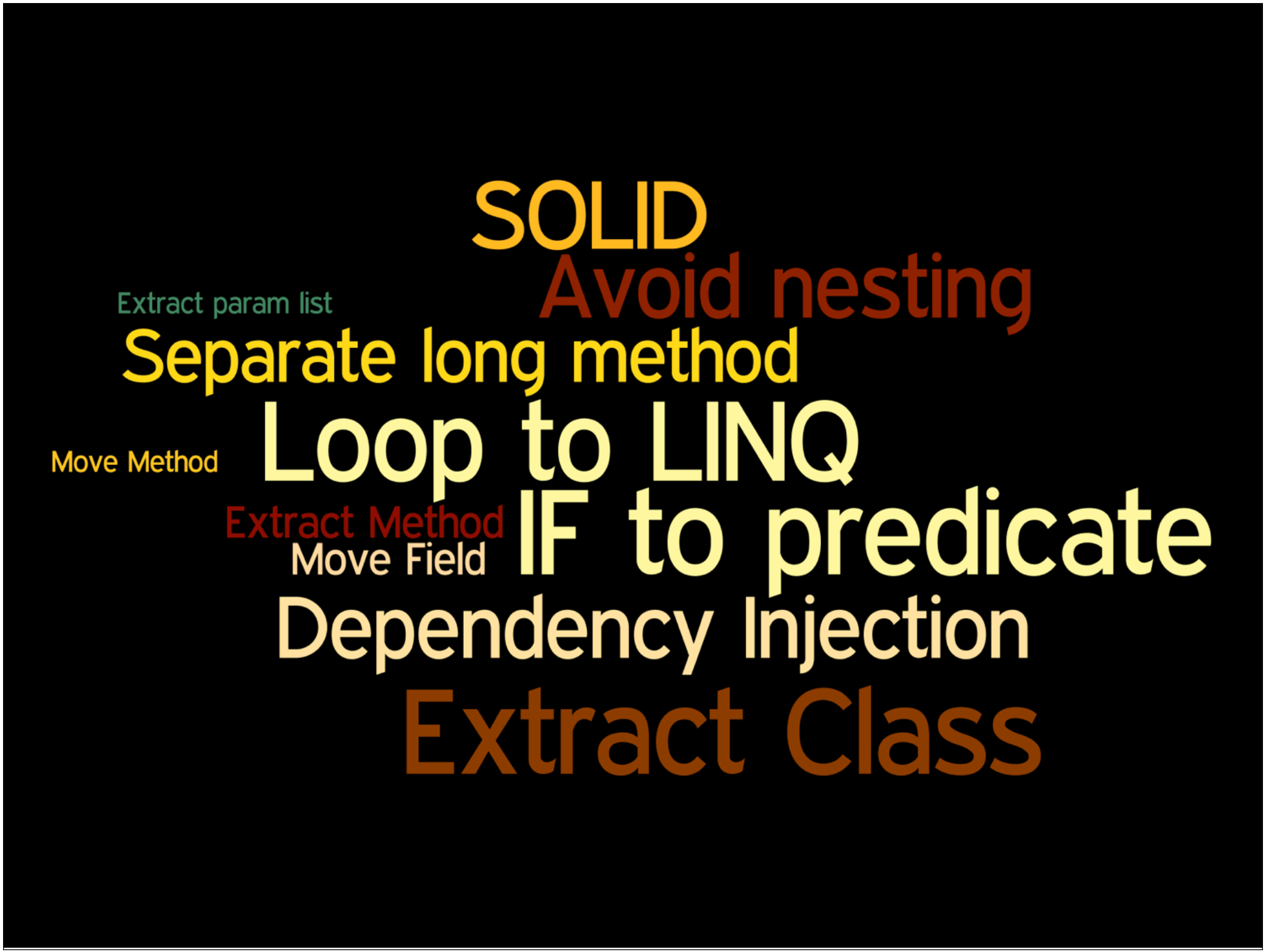
Works with **Legacy Code**

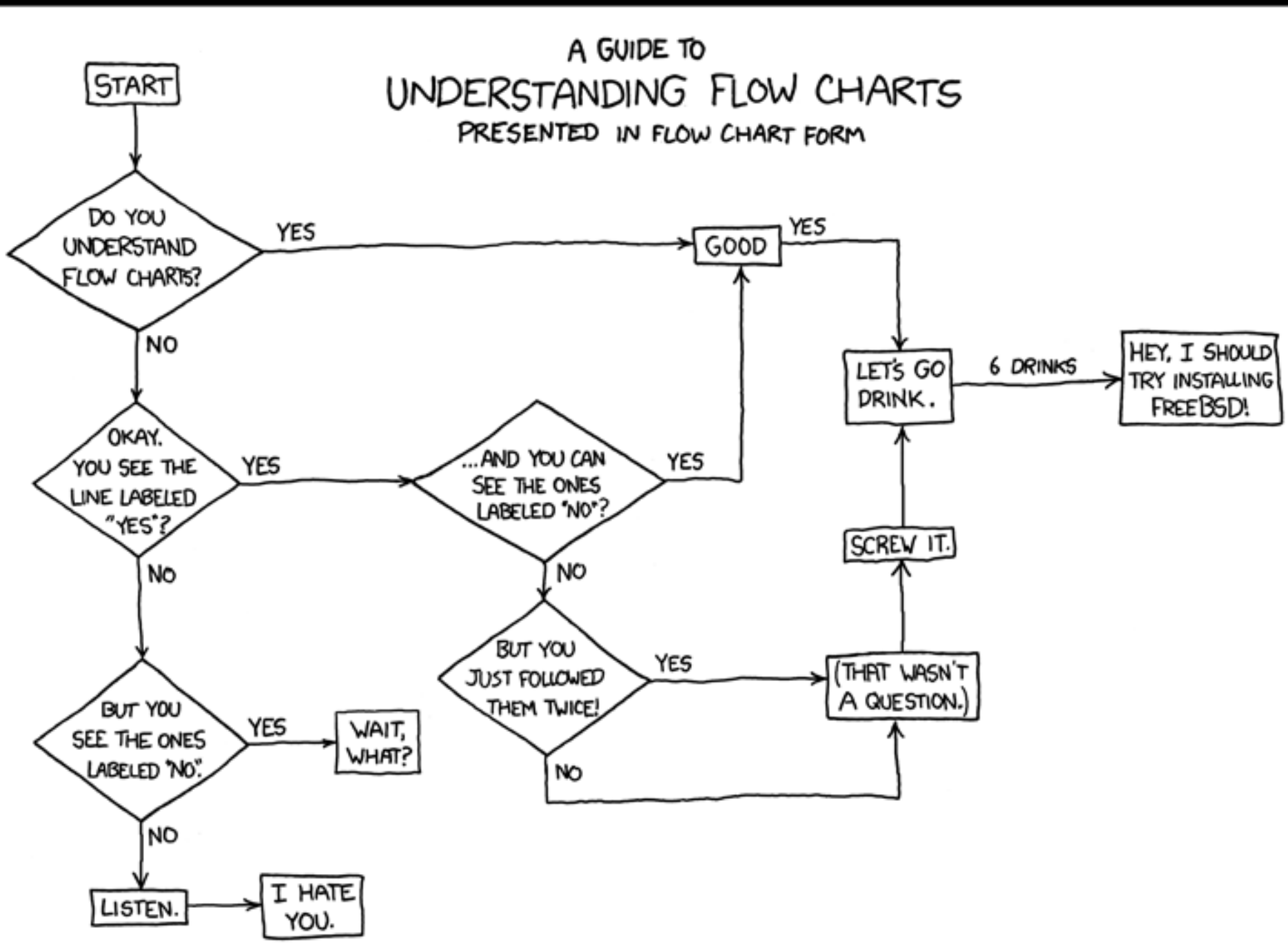
Works with **New Code**

Works with
-40!!!

Core of BDD

How to **Refactor**?

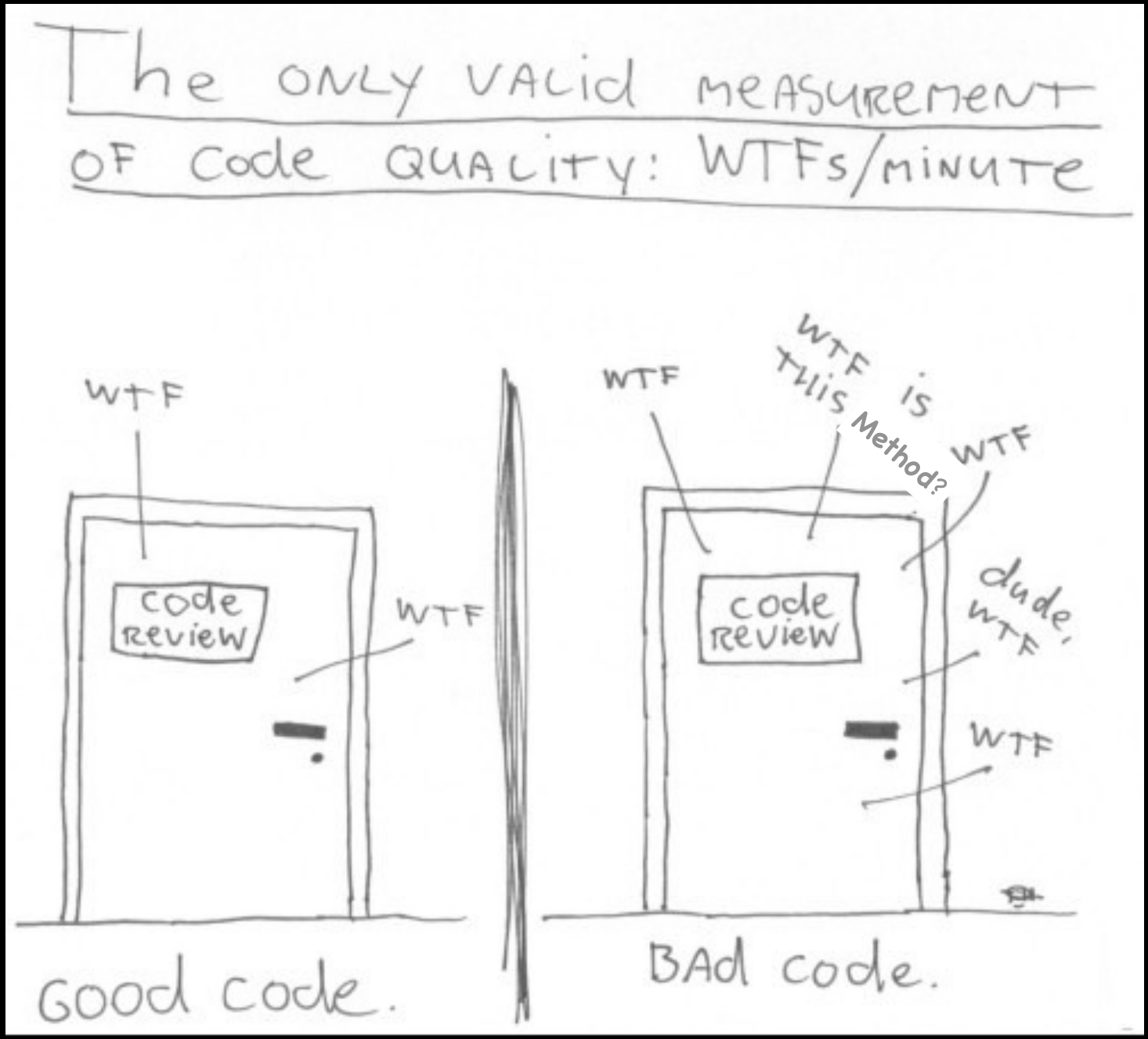

SOLID
Avoid nesting
Extract param list
Separate long method
Move Method
Loop to LINQ
Extract Method
Move Field
IF to predicate
Dependency Injection
Extract Class


A GUIDE TO
UNDERSTANDING FLOW CHARTS
PRESENTED IN FLOW CHART FORM
START
DO YOU UNDERSTAND FLOW CHARTS?
YES
GOOD
YES
NO
LET'S GO DRINK.
6 DRINKS
HEY, I SHOULD TRY INSTALLING FREEBSD!
OKAY. YOU SEE THE LINE LABELED "YES"?
YES
...AND YOU CAN SEE THE ONES LABELED "NO"?
YES
NO
SCREW IT.
BUT YOU JUST FOLLOWED THEM TWICE!
YES
(THAT WASN'T A QUESTION.)
NO
BUT YOU SEE THE ONES LABELED "NO".
YES
WAIT, WHAT?
NO
LISTEN.
I HATE YOU.

Code should be clear

Like telling a story

Refactoring algorithm



Second

Write a test for it

Third

Make it better

Fourth

Run the tests

Repeat until
out of coffee

Nesting Conditionals

C#

```
public double SomeMethod()
{
    var result = 0d;

    if (_firstGuard)
    {
        result = FirstCalculation();
        if (_secondGuard)
        {
            result = SecondCalculation();
        }
    }
    return result;
}
```

C#


```
public double BetterMethod()
{
    if (!_firstGuard)
    {
        return 0;
    }

    if (!_secondGuard)
    {
        return FirstCalculation();
    }

    return SecondCalculation();
}
```

Ruby


```ruby
class NestedCalculation
  def awesome_method
    first_calculation   ||
    second_calculation  ||
    default_calculation
  end

  def first_calculation
    @first_guard && some_calc_here
  end

  def second_calculation
    # etc...
  end
end
```

C#

```
public double SomeMethod()
{
    var result = 0d;

    if (_guard1)
    {
        if (_guard2)
        {
            if (_guard3)
            {
                result = Calc1() + Calc2();
            }
        }
    }
    return result;
}
```

C#


```
public double BetterMethod()
{
    if (_guard1 && _guard2 && _guard3)
    {
        return Calc1() + Calc2();
    }

    return 0;
}
```

C#

```
public bool SomeMethod()
{
    var result = false;

    if (_firstGuard)
    {
        if (_secondGuard)
            result = true;
    }
    else
        result = true;

    return result;
}
```

C#


```
public bool BetterMethod()
{
    return !_firstGuard || _secondGuard;
}
```

Functional Inspiration

DRY

Stop writing custom loops

Meaning rulez

Java

```java
public Iterable<String> deploy(
  Iterable<String> collection) {

  Collection<String> result = new ArrayList<>...;

  Iterator<String> cursor = collection.iterator();

  while(cursor.hasNext()) {
    result.add("Deployed to " + cursor.next());
  }

  return result;
}
```

Java


```java
public Iterable<String> betterDeploy(
  Iterable<String> environments) {

  return with(environments)
          .convert(new DeployConverter());
}

class DeployConverter
      implements Converter<String, String> {

    public String convert(String env) {
        return "Deployed to " + env;
    }
}
```

Scala


```scala
def betterDeploy(environments: Iterable[String])
  : Iterable[String] {

  environment.map env => s"Deploy to $env"
}
```

Java

```java
public class Movie {

  private String title;
  private int review;

  public Movie(String title, int review) {
    this.title = title;
    this.review = review;
  }

  public String getTitle() {...}

  public int getReview() {...}

}
```

Java


```java
@Test
public void whereAreMyPostIt() {

  // arrange
  Iterable<Movie> movies = asList(
    new Movie("Blazing Saddles", 5), new Movie("Terminator"),
    new Movie("Canadian Bacon", 8)
  );

  // act
  Iterable<Movie> reviewed =
       filter(having(on(Movie.class).getReview(), greaterThan(-1))
       , movies);

  // assert
  assertThat(joinFrom(reviewed).getTitle(),
             equalTo("Blazing Saddles, Canadian Bacon"));
}
```

Java


```java
@Test
public void wheresMyGanttChart() {

  // arrange
  Iterable<Movie> movies = asList(new Movie("Blazing Saddles"),
    new Movie("Terminator"), new Movie("Curator"));

  // act
  Matcher<Movie> endsWithAtor = new Predicate<Movie>() {
    public boolean apply(Movie item) {
      return item.getTitle().endsWith("ator");
    }
  };

  Iterable<Movie> actual = filter(endsWithAtor, movies);

  // assert
  assertThat(joinFrom(actual).getTitle(),
             equalTo("Terminator, Curator"));
}
```

C#

```
public int Mysterious(IEnumerable<int> collection)
{
    return collection.Aggregate((a, b) => a + b);
}
```

Coffee

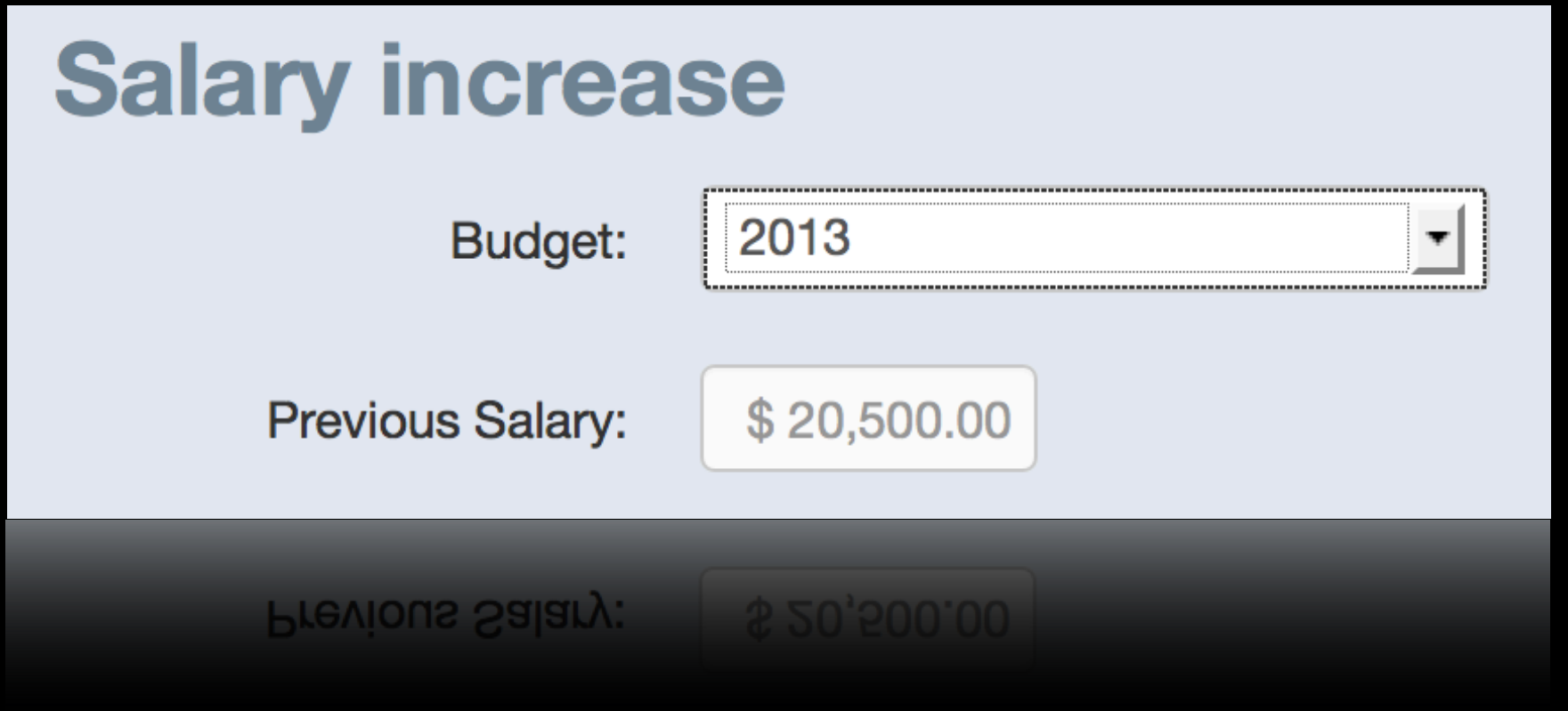
```coffee
[1..1000].reduce (t, s) -> t + s
```

What about
MONADS?

Just kidding :)

Don’t forget **OOP**

Abstraction is
KEY


Salary increase
Budget:
2013
Previous Salary:
$ 20,500.00

Coffee

```coffee
class BudgetViewModel

  constructor: (json) ->
    @budgets     = [2013, 2012, 2011]
    @budgetIndex = 0

  salary: ->
    return 5000 if @budgetIndex == 0
    return 2000 if @budgetIndex == 1
    1000
```

Coffee


```coffee
class BudgetViewModel

  constructor: ->
    @budgets = [
      new BudgetModel(2013, 5000),
      new BudgetModel(2012, 2000),
      new BudgetModel(2011, 1000)
    ]
    @budget  = @budgets[0]

  salary: => @budget.salary
```

Coffee
KO


```coffee
class BudgetViewModel

  constructor: ->
    @budgets = ko.observableArray [
      new BudgetModel(2013, 5000),
      new BudgetModel(2012, 2000),
      new BudgetModel(2011, 1000)
    ]

    @budget = ko.observable()

    @salary = ko.computed => @budget().salary
```

Language is your friend

(or it should be)

The right tool
for the job

Coffee

```coffee
class window.NewsViewModel

    constructor: (@limit = -1) ->
        @news = ko.observableArray()
        @title = ko.observable()
        $.getJSON '../api/news', @loadNews

    loadNews: (data) =>
        max = (if @limit == -1 then -1 else @limit - 1)
        @news(@createNewsItem(e) for e in data[0..max])
        @title @news()[0]?.Title

    createNewsItem: (e) =>
        newsItem =
            Title: e.Title
            Date: @parseDate(e.Date)
            Body: e.Body
```

JS

```
(function() {
  var __bind = function(fn, me){ return function(){ return fn.apply(me, arguments); }; };

  window.NewsViewModel = (function() {

    function NewsViewModel(limit) {
      this.limit = limit != null ? limit : -1;
      this.createNewsItem = __bind(this.createNewsItem, this);

      this.loadNews = __bind(this.loadNews, this);

      this.news = ko.observableArray();
      this.title = ko.observable();
      $.getJSON('../api/news', this.loadNews);
    }

    NewsViewModel.prototype.loadNews = function(data) {
      var e, max, _ref;
      max = (this.limit === -1 ? -1 : this.limit - 1);
      this.news((function() {
        var _i, _len, _ref, _results;
        _ref = data.slice(0, max + 1 || 9e9);
        _results = [];
        for (_i = 0, _len = _ref.length; _i < _len; _i++) {
          e = _ref[_i];
          _results.push(this.createNewsItem(e));
        }
        return _results;
      }).call(this));
      return this.title((_ref = this.news()[0]) != null ? _ref.Title : void 0);
    };

    NewsViewModel.prototype.createNewsItem = function(e) {
      var newsItem;
      return newsItem = {
        Title: e.Title,
        Date: this.parseDate(e.Date),
        Body: e.Body
      };
    };

    return NewsViewModel;

  })();

}).call(this);
```

JVM supports
multiple languages

same for
.net framework

Tests are a great
place to start

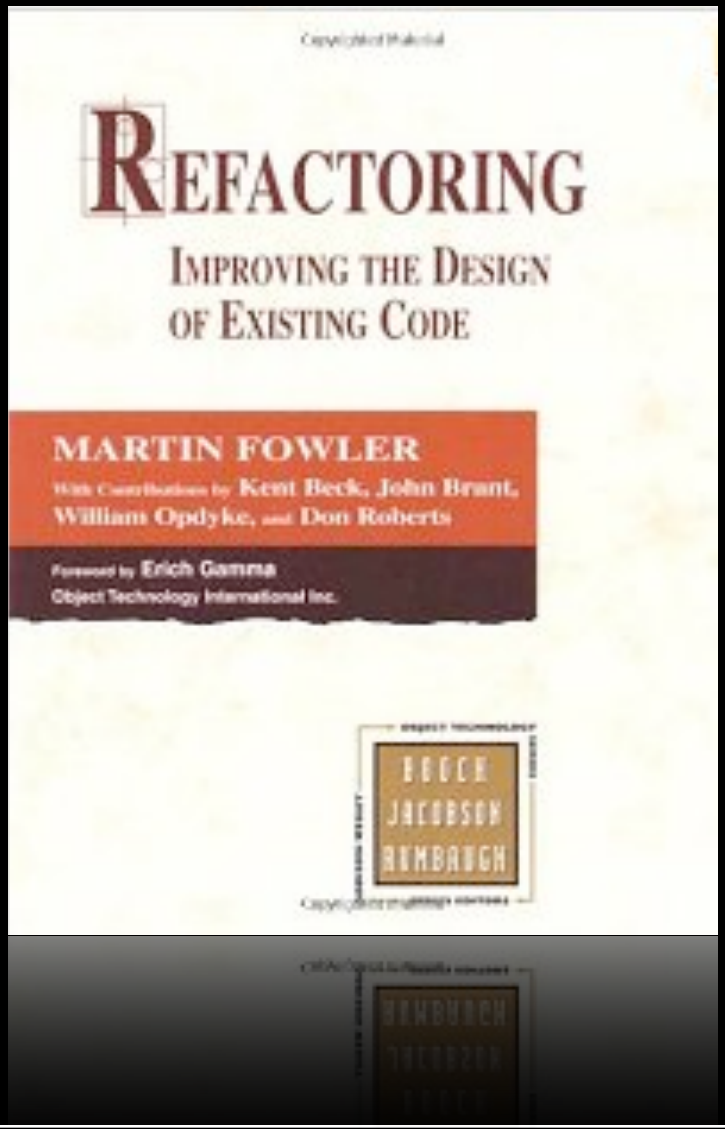

Thank you!

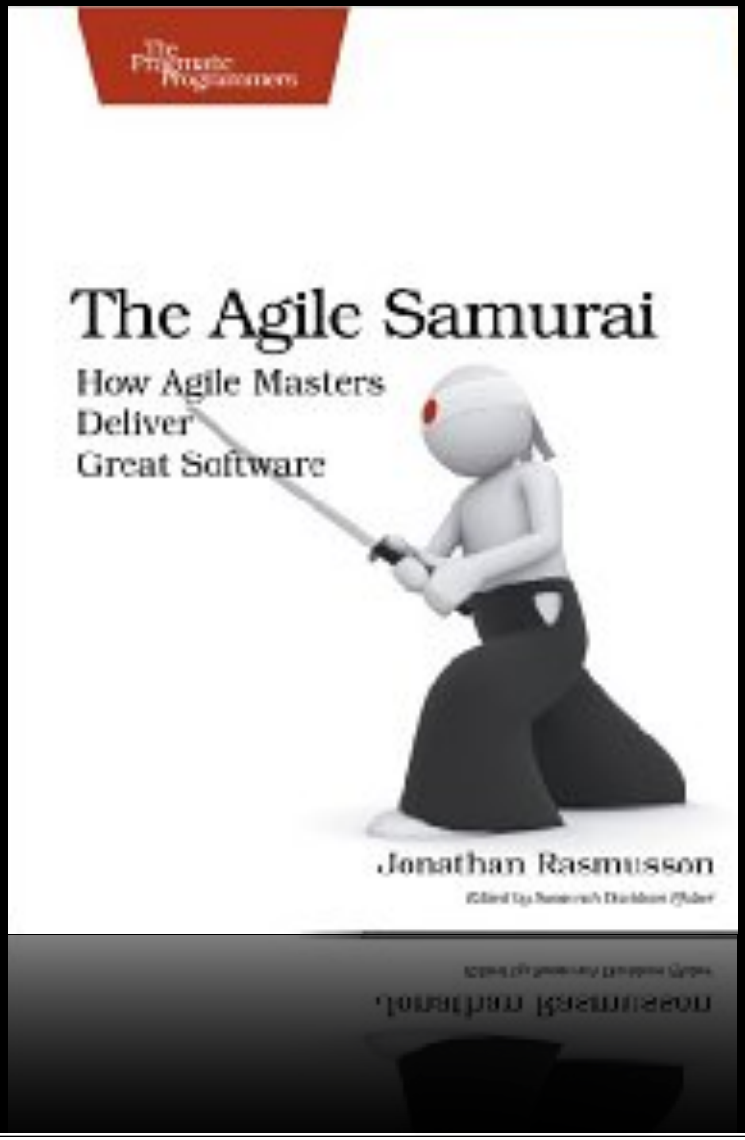
amir@barylko.com


@abarylko

http://bit.ly/**abarylkop**

Books


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.
BECK
JACOBSON
BRANBAUGH


The Pragmatic Programmers
The Agile Samurai
How Agile Masters
Deliver
Great Software
Jonathan Rasmusson

Books


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

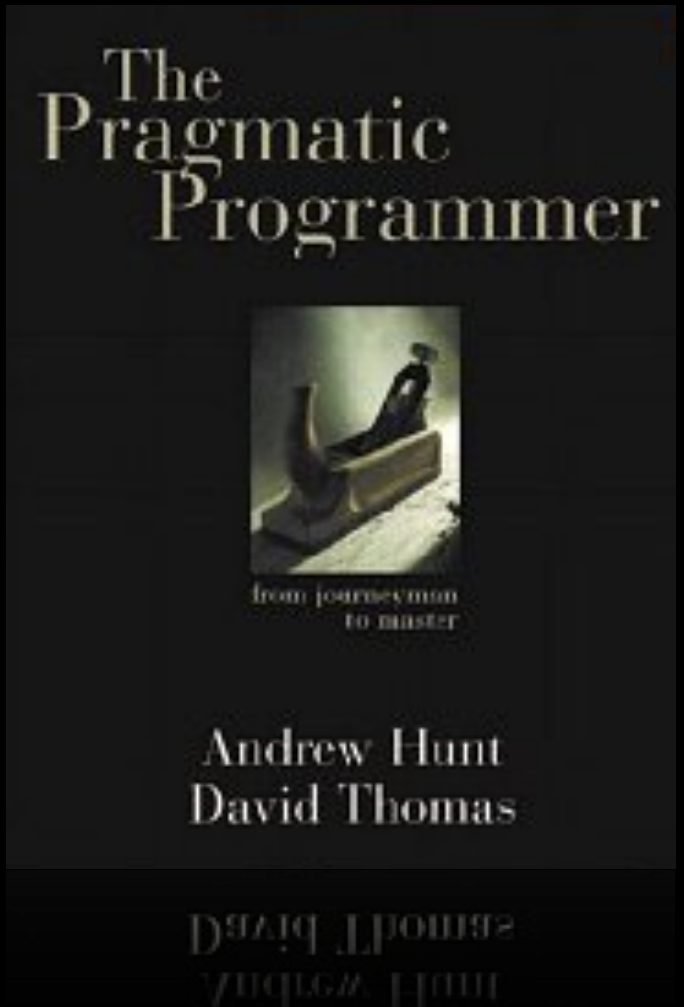

The Pragmatic Programmer
from journeyman to master
Andrew Hunt
David Thomas

Photo Credit

- Under http://creativecommons.org/licenses/by/2.5/
  - Joe Cheng, DSC_7820-01, http://flic.kr/p/2Zt2u
  - Bill Ward, Derek Schin's Trucks 1, http://flic.kr/p/m5L5S
  - Jeremy Keith, Roast beef, http://flic.kr/p/TKUz
  - Rob Campbell, Field of daisies, http://flic.kr/p/6QJjU4
  - Karin Dalziel, The Thinker, http://flic.kr/p/4UYArc
- Under http://creativecommons.org/licenses/by-sa/3.0/us/
  - Derick Bailey, SOLID Motivational Posters, http://bit.ly/17aVaHg

Photo Credit 2

- How to write good code, http://xkcd.com/844/
- Understanding flow charts, http://lifehacker.com/5909501/how-to-choose-the-best-chart-for-your-data

Resources

- http://www.infoq.com/news/2010/06/decision-to-refactor
- http://stackoverflow.com/questions/38635/what-static-analysis-tools-are-available-for-c
- Refactoring Catalog: http://www.refactoring.com/catalog/
- LambdaJ: https://code.google.com/p/lambda
- Coffeescript: http://coffeescript.org/