



Introduction to Test Driven Development



Dario Campagna

Let's start with
Development



Now add **Test** **Driven**

Software development practice

Clean code that works

Test first

Small steps, fast feedback



Clean code

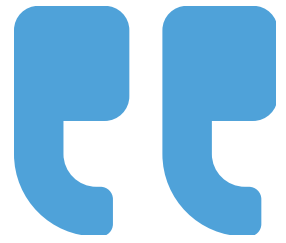
Easy to understand

Easy to evolve

Easy to maintain

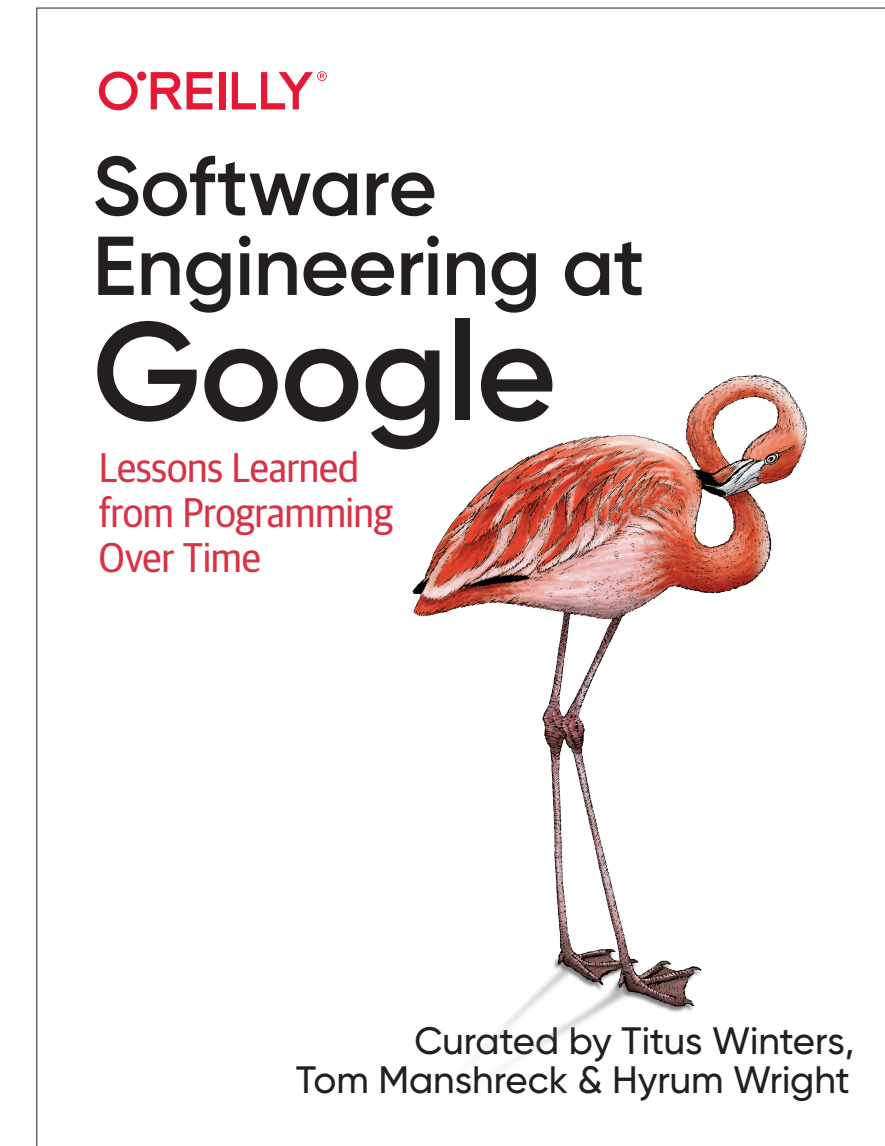
Sustains delivery pace





Your project is **sustainable** if, for the expected life span of your software, you are capable of reacting to whatever **valuable change** comes along, for either technical or business reasons.

Software Engineering at Google



[Free digital version](#)



Example of Ugly Code

TerrariaClone class from the GitHub repository [TerrariaClone](#).

- > 6500 lines of code
- > 1300 lines of code for init() method
- Deeply nested if and for statements
- Many other “issues”

```
3056     if (ic != null) {
3057         if (ic.type.equals("workbench")) {
3058             for (ux=0; ux<3; ux++) {
3059                 for (uy=0; uy<3; uy++) {
3060                     if (mousePos[0] >= ux*40+6 && mousePos[0] < ux*40+46 &&
3061                         mousePos[1] >= uy*40+inventory.image.getHeight()+46 &&
3062                         mousePos[1] < uy*40+inventory.image.getHeight()+86) {
3063                         checkBlocks = false;
3064                         if (mouseClicked) {
3065                             mouseNoLongerClicked = true;
3066                             moveItemTemp = ic.ids[uy*3+ux];
3067                             moveNumTemp = ic.nums[uy*3+ux];
3068                             if (moveItem == ic.ids[uy*3+ux]) {
3069                                 moveNum = (short)inventory.addLocationIC(ic, uy*3+ux, moveItem, moveNum, moveDur);
3070                                 if (moveNum == 0) {
3071                                     moveItem = 0;
3072                                 }
3073                             }
3074                             else {
3075                                 inventory.removeLocationIC(ic, uy*3+ux, ic.nums[uy*3+ux]);
3076                                 if (moveItem != 0) {
3077                                     inventory.addLocationIC(ic, uy*3+ux, moveItem, moveNum, moveDur);
3078                                 }
3079                                 moveItem = moveItemTemp;
3080                                 moveNum = moveNumTemp;
3081                             }
3082                         }
3083                     }
3084                 }
3085             }
3086         if (mousePos[0] >= 4*40+6 && mousePos[0] < 4*40+46 &&
3087             mousePos[1] >= 1*40+inventory.image.getHeight()+46 &&
3088             mousePos[1] < 1*40+inventory.image.getHeight()+86) {
3089             checkBlocks = false;
3090             if (mouseClicked) {
3091                 if (moveItem == ic.ids[9] && moveNum + ic.nums[9] <= MAXSTACKS.get(ic.ids[9])) {
3092                     moveNum += ic.nums[9];
3093                     inventory.useRecipeWorkbench(ic);
3094                 }
3095                 if (moveItem == 0) {
3096                     moveItem = ic.ids[9];
3097                     moveNum = ic.nums[9];
3098                     if (TOOLDURS.get(moveItem) != null) {
3099                         moveDur = TOOLDURS.get(moveItem);
3100                     }
3101                     inventory.useRecipeWorkbench(ic);
3102                 }
3103             }
3104         }
3105     }
3106 }
```



Ugly code is

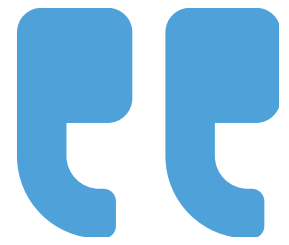
Rigid

Fragile

Inseparable

Opaque





The nature of code is to grow ugly.

Bas Vodde



Why does code **grow ugly?**

We have no time to clean it

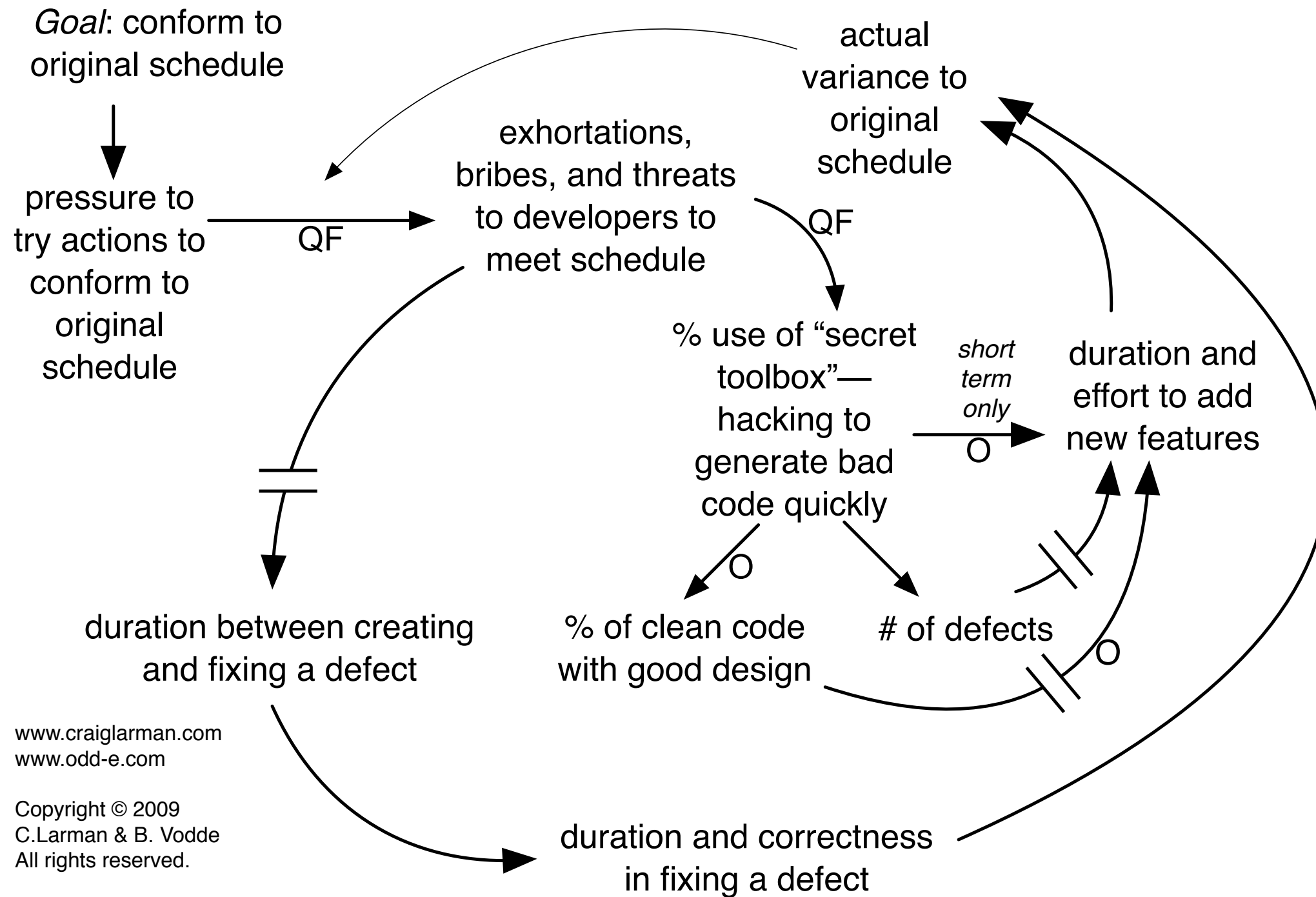
We need to go “faster”

We are afraid of breaking it

Fear prevents us to clean it



The "Faster is Slower" Dynamic

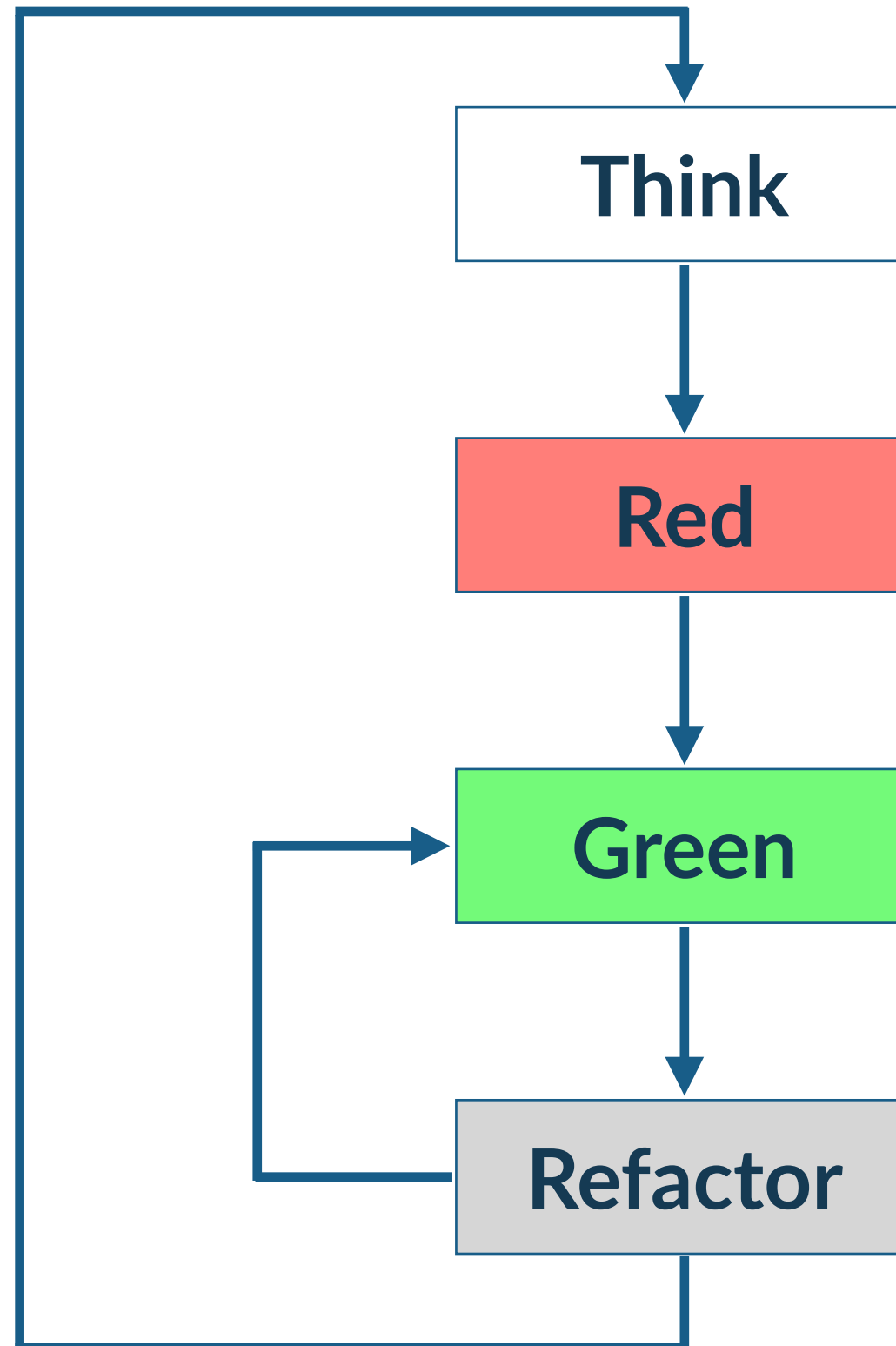


www.craiglarman.com
www.odd-e.com

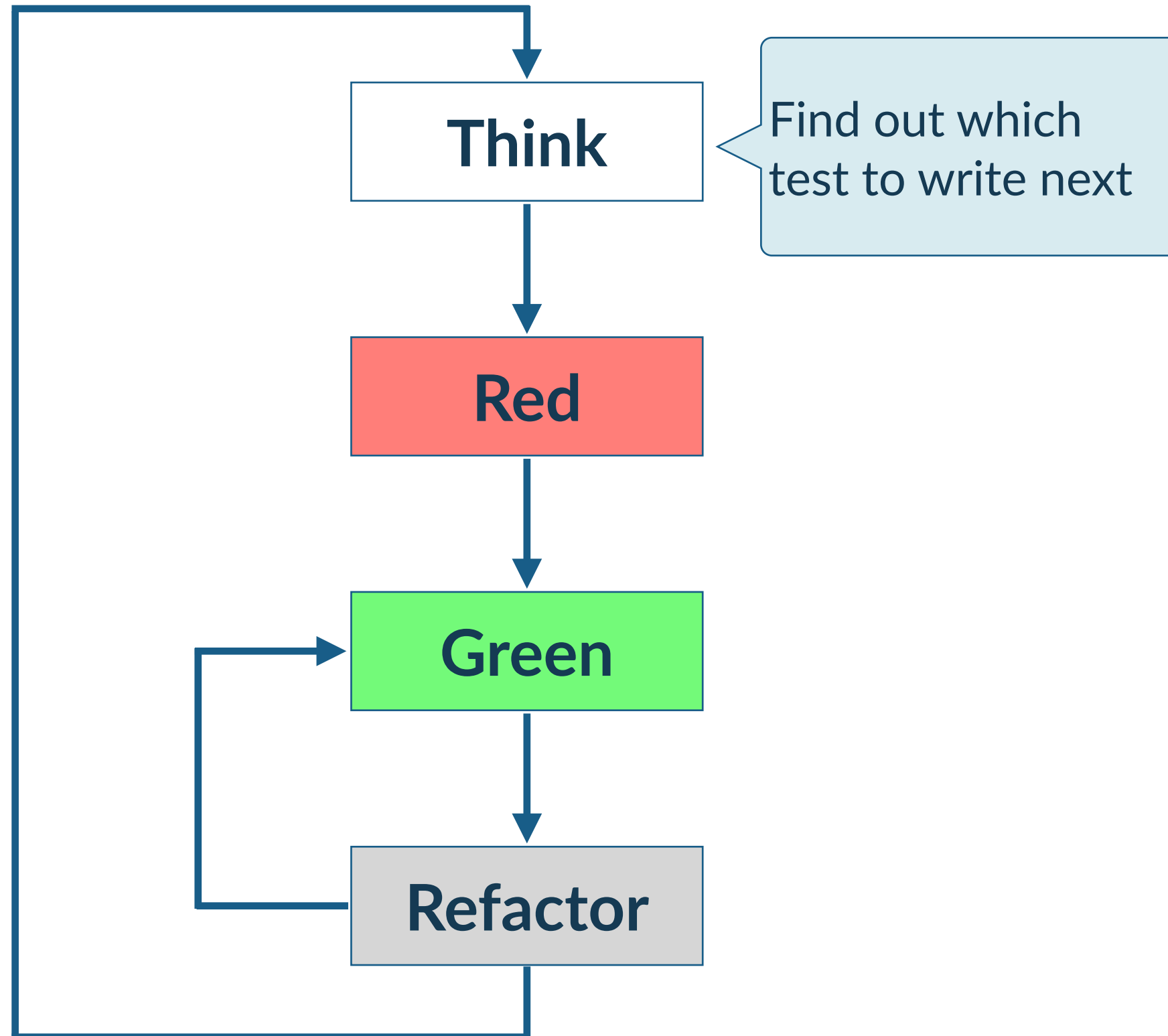
Copyright © 2009
C.Larman & B. Vodde
All rights reserved.



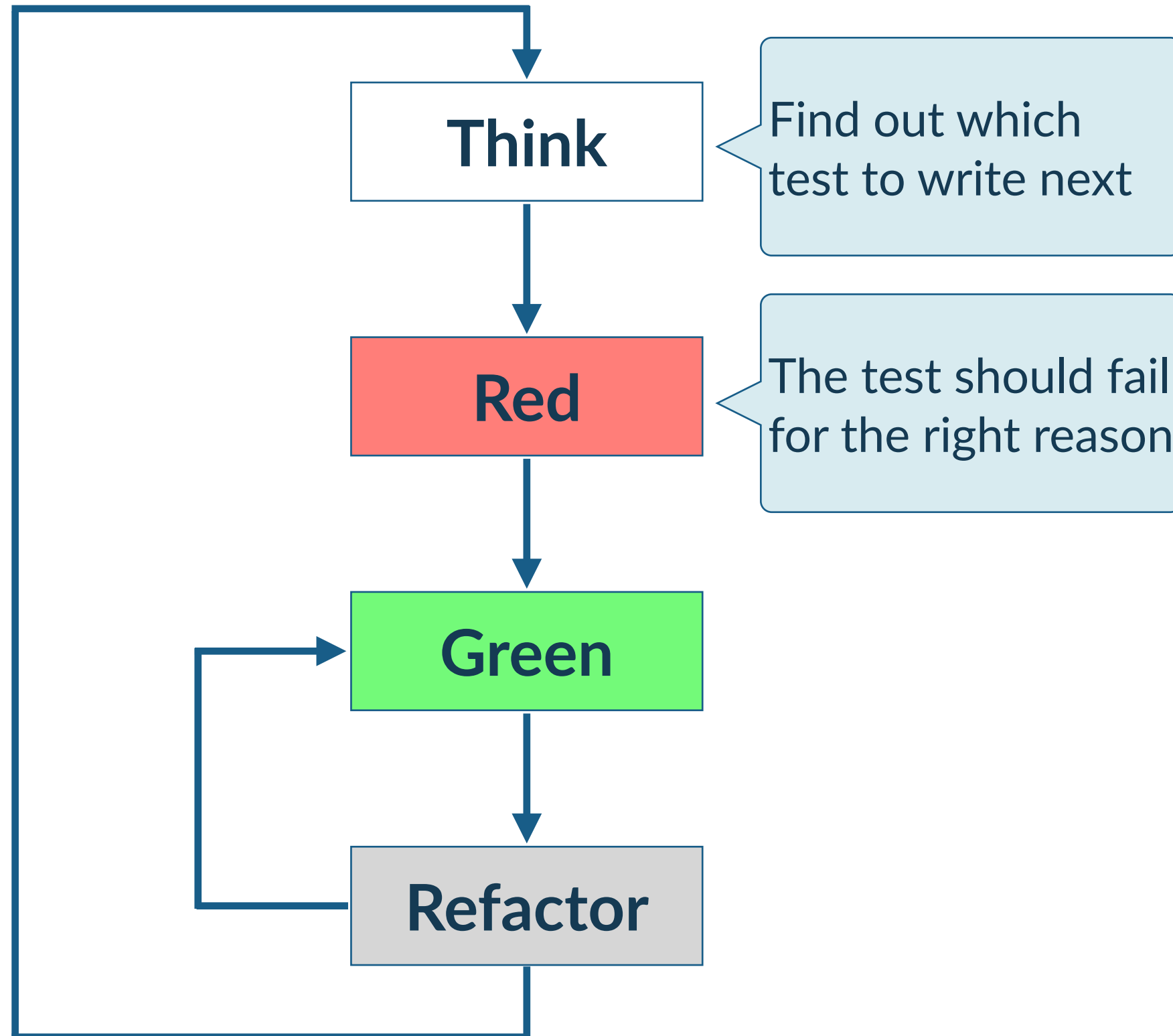
Test Driven Development (TDD) Cycle



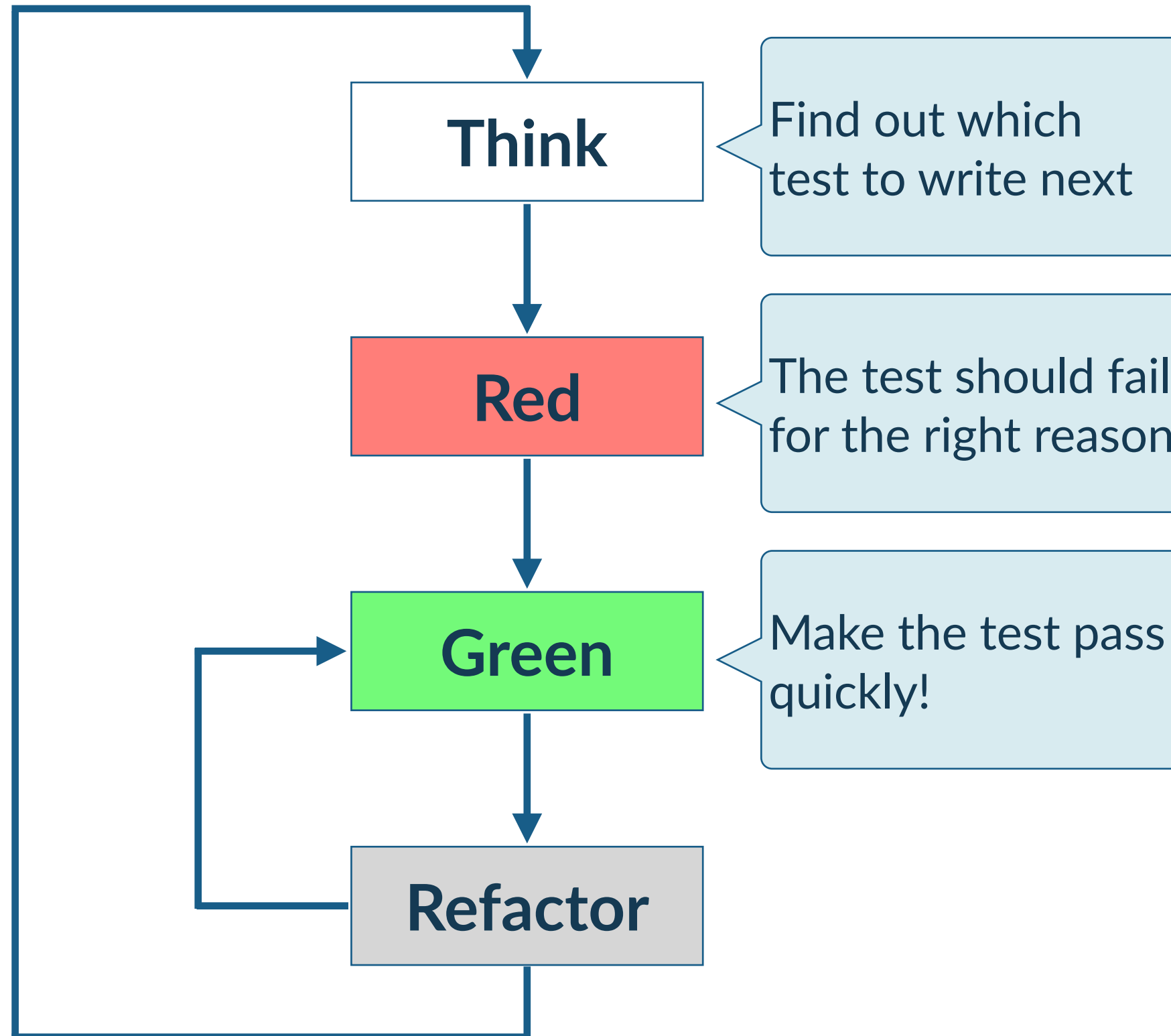
Test Driven Development (TDD) Cycle



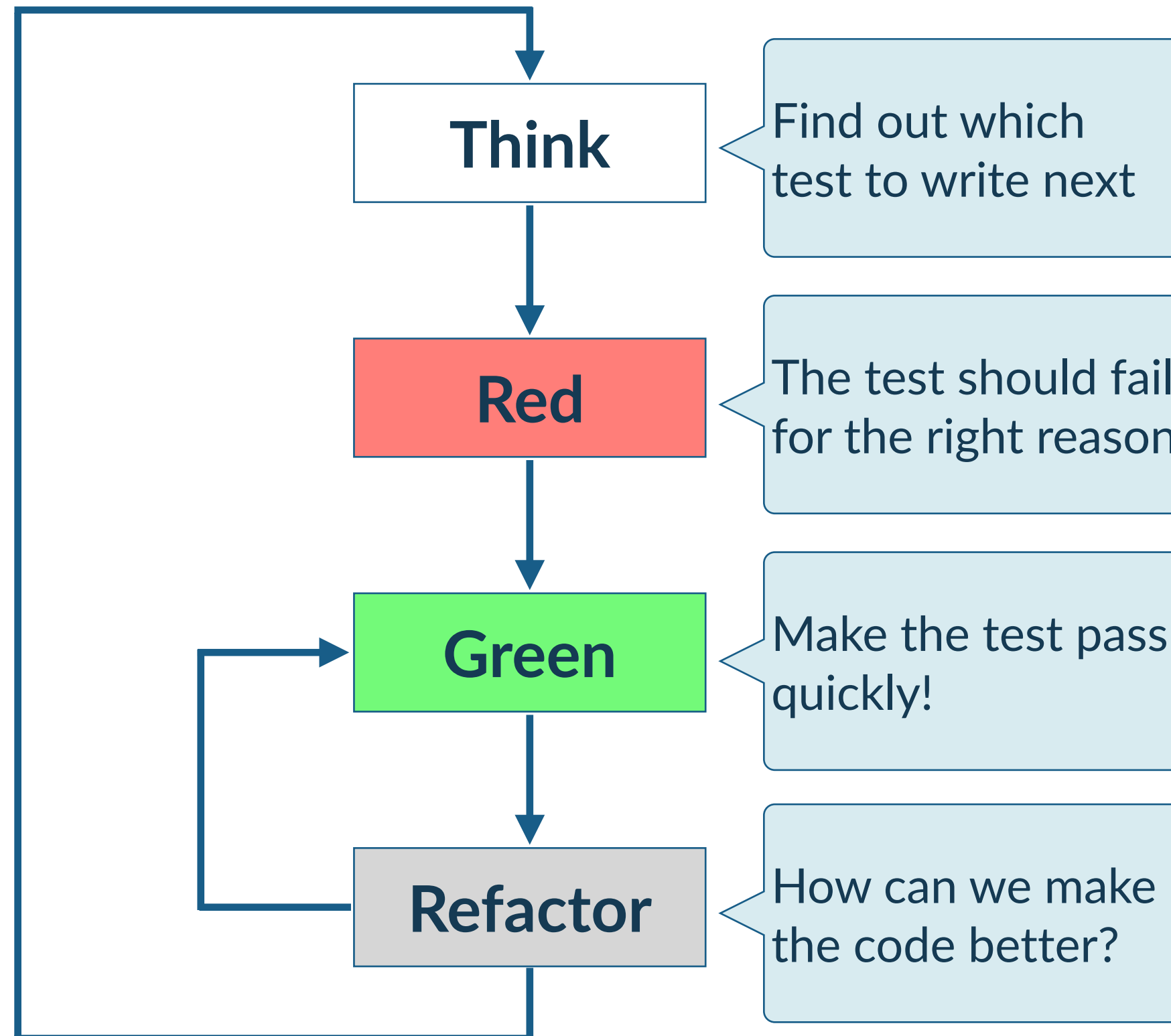
Test Driven Development (TDD) Cycle



Test Driven Development (TDD) Cycle



Test Driven Development (TDD) Cycle



Test Infrastructure

Things we need to practice TDD

Automated build

Test framework

Assertion library

Arrange/Act/Assert

Why?

To run the tests, as fast as possible

To build the test suite

To check test status

A way to write tests



Code Kata

A system of coding practice incorporating techniques and notions that have been cultivated and polished for decades.

[Dave Nicolette](#)

- The purpose is to practice and internalize programming *techniques*
- (Some are) Designed to reflect programming problems that have particular *shapes*



Good tests

Describe

Tests should have names that describe a business feature or behavior.

Avoid

Technical names and leaking implementation details.

Communicate

Tests should clearly express required functionalities to the reader.



Ways to move forward

Fake it

Just return the exact value you need.

Something that works is better than something that doesn't work!

Obvious implementation

When you are sure of the code you need to write, write it, and see the test go green!

Triangulation

Write a new and more specific test that forces the code to be more generic.



Refactor to remove duplication

Types of duplication

Code, data, knowledge.

Wait

Avoid removing duplication too soon, as this may lead you to extract the wrong abstractions.

Rule of Three

Extract duplication only when you see it for the third time.



Duplication of knowledge

```
public class Cylinder {  
  
    private final double radius;  
    private final double height;  
  
    public Cylinder(double radius, double height) {  
        this.radius = radius;  
        this.height = height;  
    }  
  
    public double volume() {  
        return Math.PI * Math.pow(radius, 2) * height;  
    }  
  
    public double surface() {  
        return 2 * Math.PI * Math.pow(radius, 2) + 2 * Math.PI * radius * height;  
    }  
}
```



Duplication of knowledge

```
public class Cylinder {  
  
    private final double radius;  
    private final double height;  
  
    public Cylinder(double radius, double height) {  
        this.radius = radius;  
        this.height = height;  
    }  
  
    public double volume() {  
        return Math.PI * Math.pow(radius, 2) * height;  
    }  
  
    public double surface() {  
        return 2 * Math.PI * Math.pow(radius, 2) + 2 * Math.PI * radius * height;  
    }  
}
```



Duplication of knowledge

Extract method.

```
public class Cylinder {  
  
    private final double radius;  
    private final double height;  
  
    public Cylinder(double radius, double height) {  
        this.radius = radius;  
        this.height = height;  
    }  
  
    public double volume() {  
        return baseSurface() * height;  
    }  
  
    public double surface() {  
        return 2 * baseSurface() + 2 * Math.PI * radius * height;  
    }  
  
    private double baseSurface() {  
        return Math.PI * Math.pow(radius, 2);  
    }  
}
```



Duplication of knowledge

Extract method.

```
public class Cylinder {  
  
    private final double radius;  
    private final double height;  
  
    public Cylinder(double radius, double height) {  
        this.radius = radius;  
        this.height = height;  
    }  
  
    public double volume() {  
        return baseSurface() * height;  
    }  
  
    public double surface() {  
        return 2 * baseSurface() + 2 * Math.PI * radius * height;  
    }  
  
    private double baseSurface() {  
        return Math.PI * Math.pow(radius, 2);  
    }  
}
```



Duplication of hard coded data

```
@Test
public void productNotFound() throws Exception {
    Display display = new Display();
    Sale sale = new Sale(display);

    sale.onBarcode("99999");

    assertEquals("Product not found for 99999", display.getText());
}
```

```
public class Sale {

    private Display display;

    public Sale(Display display) {
        this.display = display;
    }

    public void onBarcode(String barcode) {
        display.setText("Product not found for 99999");
    }
}
```



Duplication of hard coded data

```
@Test
public void productNotFound() throws Exception {
    Display display = new Display();
    Sale sale = new Sale(display);

    sale.onBarcode("99999");

    assertEquals("Product not found for 99999", display.getText());
}
```

```
public class Sale {

    private Display display;

    public Sale(Display display) {
        this.display = display;
    }

    public void onBarcode(String barcode) {
        display.setText("Product not found for 99999");
    }
}
```



Duplication of hard coded data

Replace literal value with variable.

```
@Test
public void productNotFound() throws Exception {
    Display display = new Display();
    Sale sale = new Sale(display);

    sale.onBarcode("99999");

    assertEquals("Product not found for 99999", display.getText());
}
```

```
public class Sale {

    private Display display;

    public Sale(Display display) {
        this.display = display;
    }

    public void onBarcode(String barcode) {
        display.setText("Product not found for " +
            barcode);
    }
}
```



Duplication of hard coded data

Replace literal value with variable.

```
@Test
public void productNotFound() throws Exception {
    Display display = new Display();
    Sale sale = new Sale(display);

    sale.onBarcode("99999");

    assertEquals("Product not found for 99999", display.getText());
}
```

```
public class Sale {

    private Display display;

    public Sale(Display display) {
        this.display = display;
    }

    public void onBarcode(String barcode) {
        display.setText("Product not found for " +
            barcode);
    }
}
```



Tests in TDD

Should be...

Isolated and composable

Fast and automated

Behavioral and structure-insensitive

Specific and deterministic

Inspiring and predictive

Writable and readable

Beware of

Databases

Network communications

File system

Other shared fixtures

Configurations



Should you **always** practice
Test Driven Development?



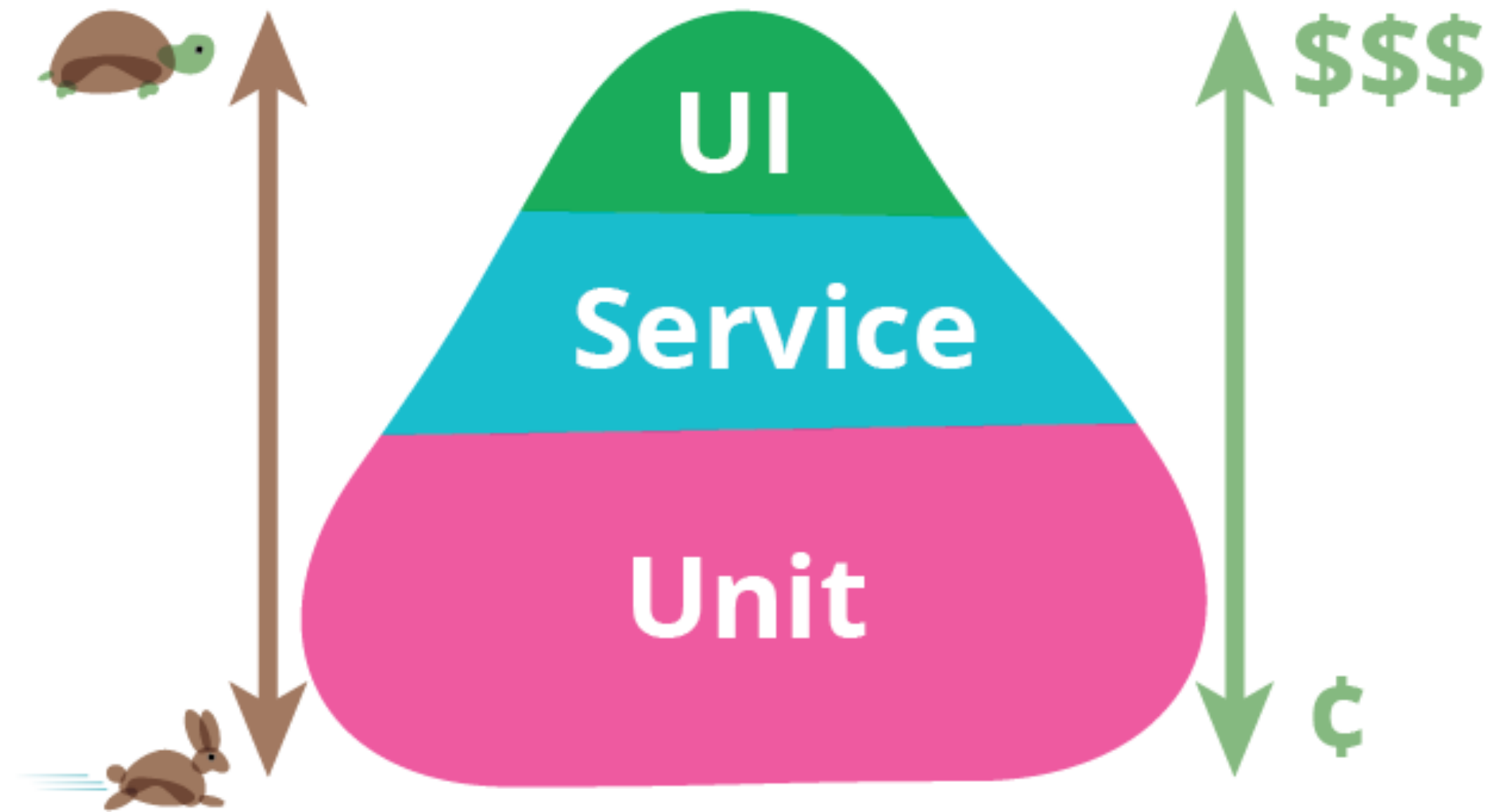
TDD is a **tool**

To get value from a tool, it's necessary to:

1. Choose the right tool for the job.
2. Use the tool properly.



Test Pyramid



<https://martinfowler.com/bliki/TestPyramid.html>
Copyright © 2012 Martin Fowler



References



Test Driven Development by Example

Kent Beck

Agile Technical Practices Distilled

Pedro Moreira Santos, Marco Consolaro,
Alessandro Di Gioia

Extreme Programming Explained

Kent Beck