

# Building Uncle Bob's Button

Burk Hufnagel - Technical Architect  
Daugherty Business Solutions



# Who is Burk Hufnagel?





# Who is Burk Hufnagel?

Technical reviewer for SCJP 6 Study Guide.

“Burk fixed more code than we care to admit.”

— Kathy Sierra & Bert Bates

Sun Certified Java Programmer, Developer and Enterprise Architect.

Programmer and Architect with Daugherty Business Solutions.

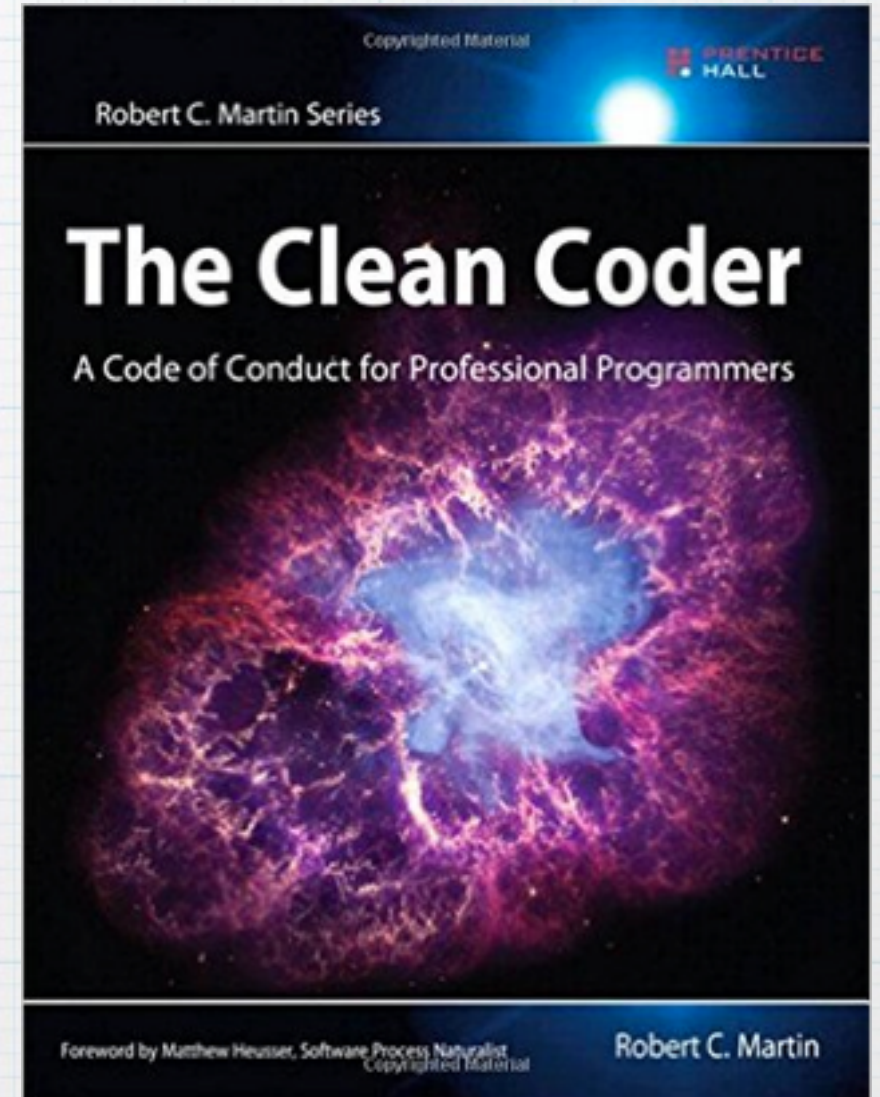
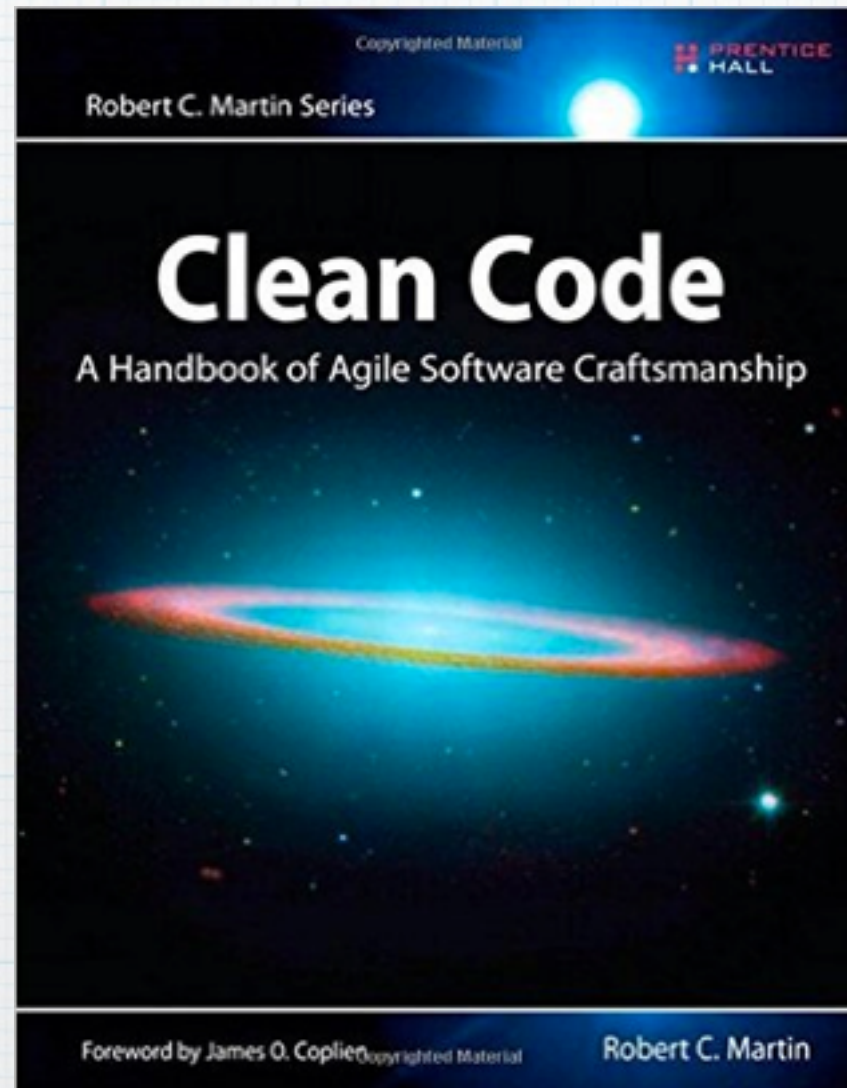


# Who is Uncle Bob?

Co-signer of the Agile Manifesto



Robert C. Martin  
aka Uncle Bob



Advocate of Test Driven Development.



# What is Uncle Bob's Button?

Sometime before 2010,  
a question was posted on Quora.com:

“Does test-driven development (TDD)  
really improve software quality?”

On June 18, 2015, Robert C. Martin  
(Uncle Bob), posted his answer...



# What is Uncle Bob's Button?

Guns don't kill people.  
People kill people.

TDD doesn't improve software quality.  
People improve software quality.

But TDD, used responsibly,  
with the right training, is an assault rifle.



# What is Uncle Bob's Button?

Imagine having a push button that could tell you, in a few seconds, whether your whole system was working as expected or not.

Push it and it lights up green if everything is working, or red if it isn't...

What could you do, if you had that button?

How fast could you go?

(Paraphrased from the original by me)



# Why Do You Want Uncle Bob's Button?

What could you do  
if you had that button?

How fast could you go  
if you had that button?



# Buy vs Build

Why not find and buy a software package or subscribe to a service?

Turns out it's a custom job that has to happen while the software is being written...



# Real World Version

For Uncle Bob's Button you would need:

- \* Button - press to initiate process
- \* Sensors - (custom) examine and report
- \* Red and Green Lights - visual feedback
- \* Wire - connect everything
- \* Power Source - used by sensors and lights



# Software Version

In software, it's simpler but not easier:

- \* Test framework - wires and power source
- \* Unit tests - sensors and lights
- \* Follow Test Driven Development to create the unit tests



# What's a Unit Test

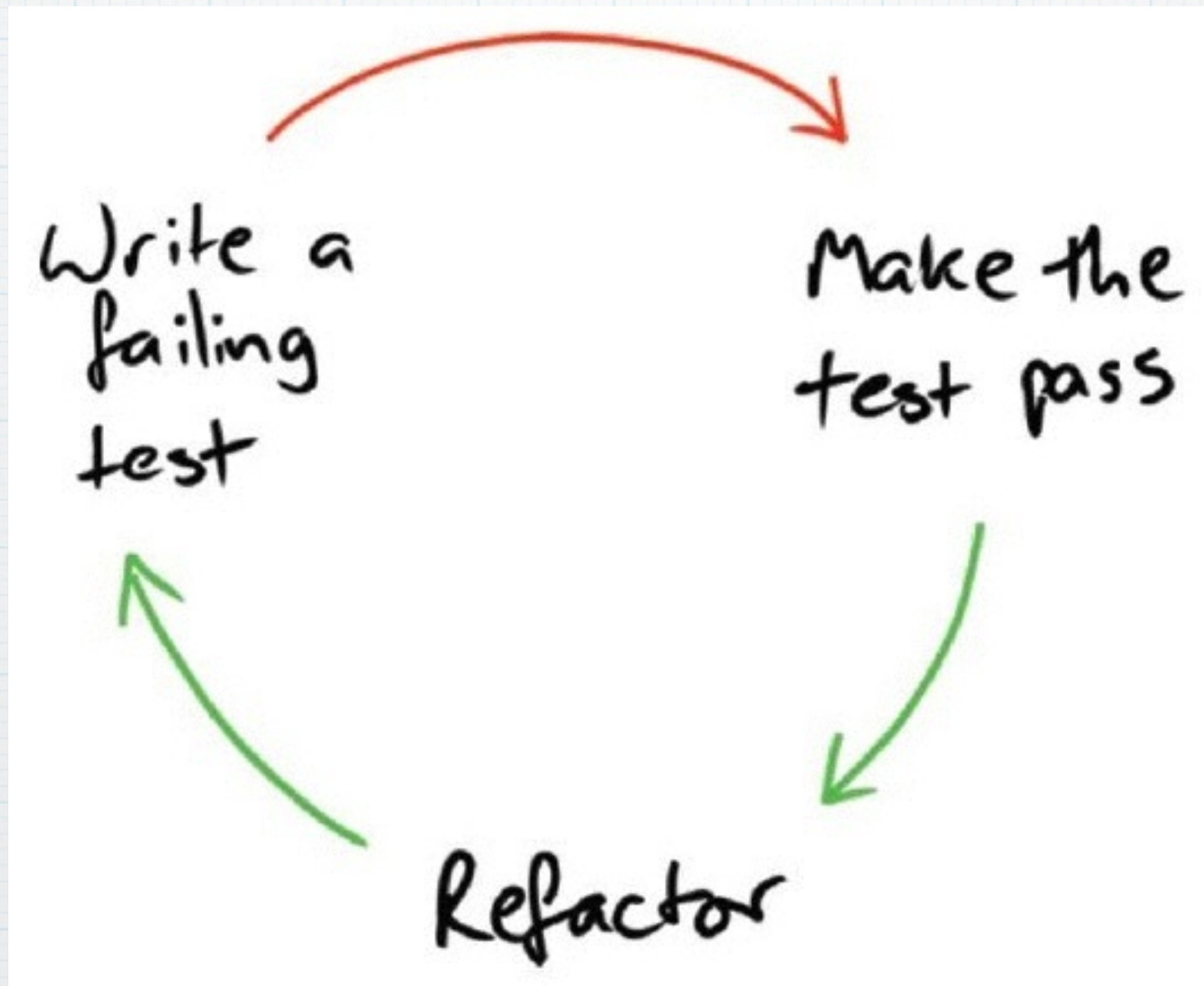
A unit test is an automated test that focuses on one piece of code; for Java this is normally a single class.

It's not a unit test if it:

- \* Talks to the database
- \* Communicates across the network
- \* Touches the file system
- \* Can't run while any other unit tests are running
- \* Requires changing the environment  
(like editing config files) for it to run.



# What is TDD?





# Building Uncle Bob's Button is a Do It Yourself project

- \* Unit Tests are written as you design and build your application.
- \* Unit Tests return value **as** you're writing them.
- \* However, I see two problems...



# Only Part of the Picture





# TDD: It's Not That Simple

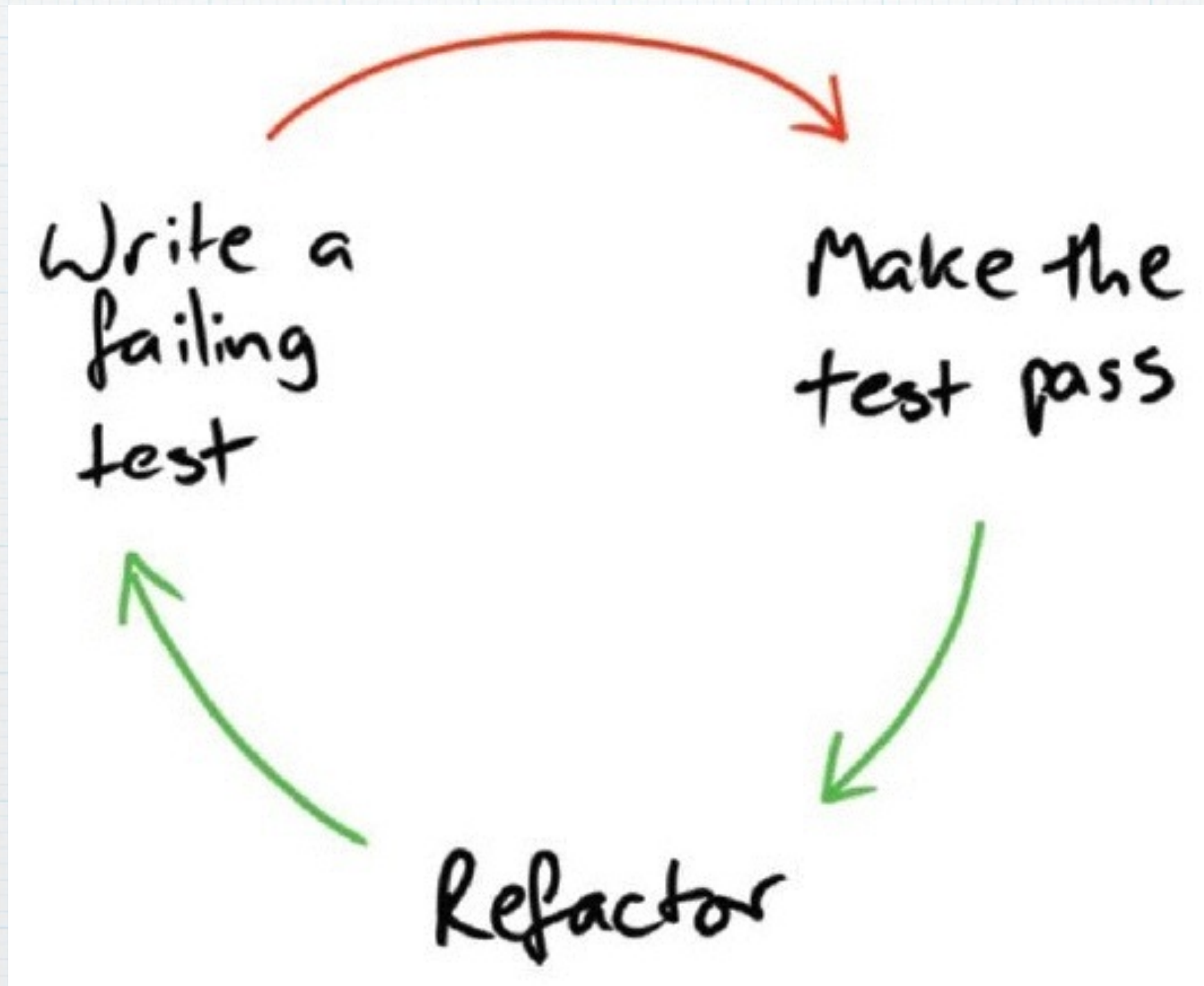


By Gil Zilberfeld

*“TDD is really easy to explain. The rules are easy to follow. Why then so few people actually do it, and when they do, so few actually do it well?”*



# Know What Tests to Write





# Uncle Bob's Button 2.0

Uncle Bob's Button is based on using Test Driven Development. This is Good News because TDD helps you design and build good code.

However, I believe that TDD is not enough precisely because it is focused on units.



# Behavior Driven Development

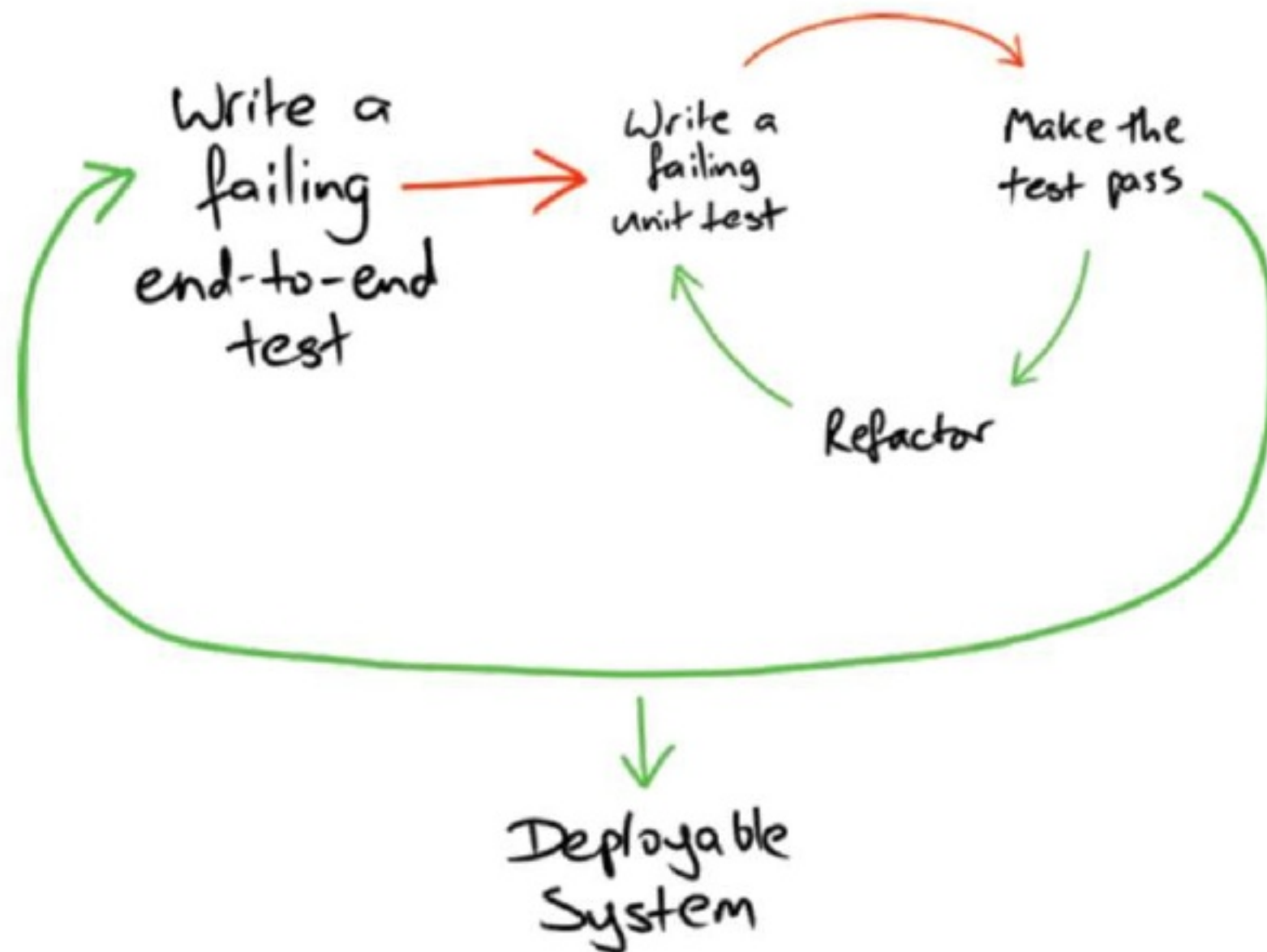


Dan North  
Father of BDD

“BDD is about implementing an application by describing its behavior from the perspective of its stakeholder.”



# Behavior Driven Development





# TDD vs BDD

TDD is about building things the right way

- \* High quality code
- \* Easy to understand and maintain

BDD is about building the right thing.

- \* Meet business needs/requirements.



*Build it Right*

Business Failure

Business Success

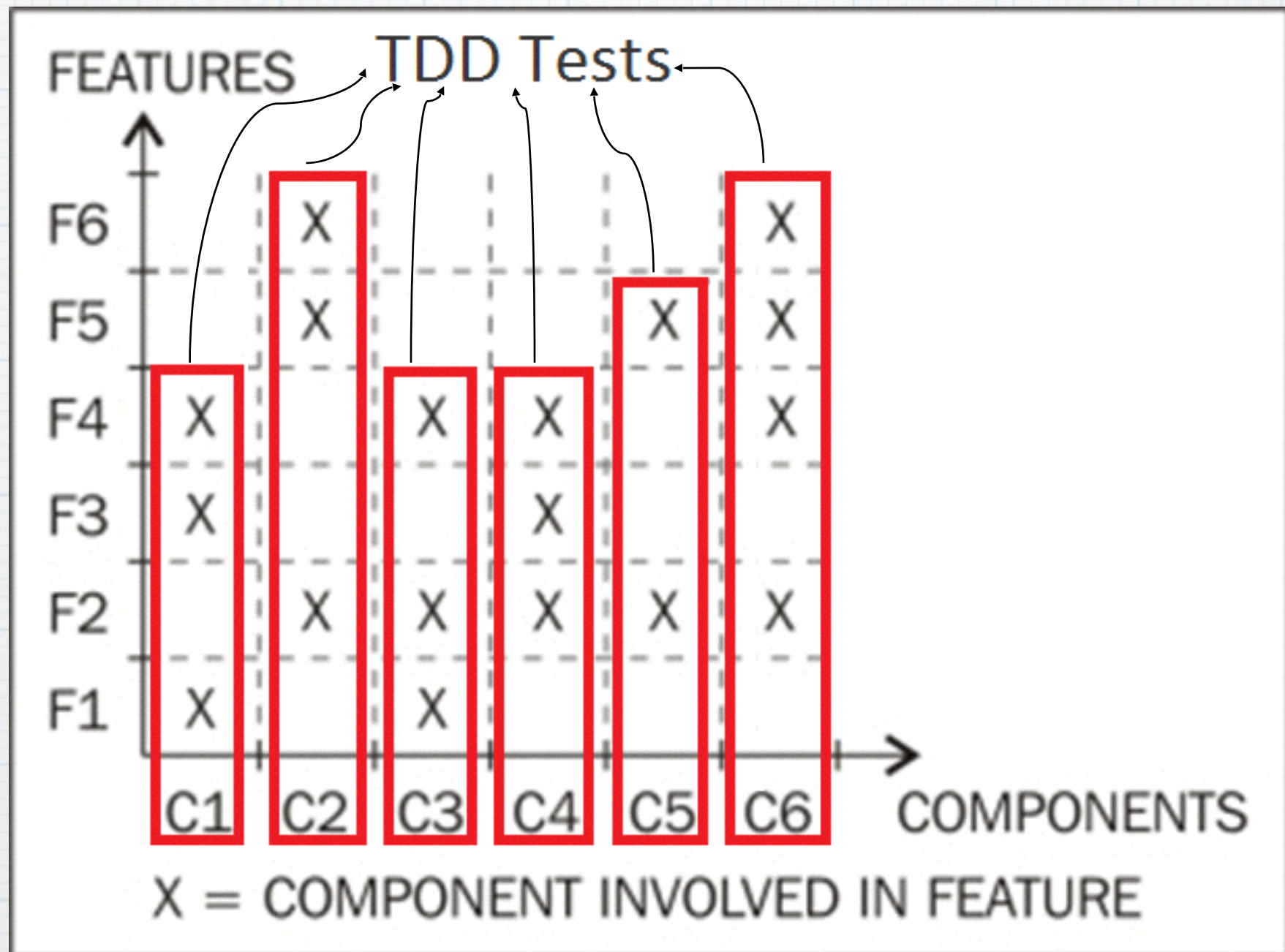
*Build the Right Thing*

Useless Crap

Maintenance Nightmare



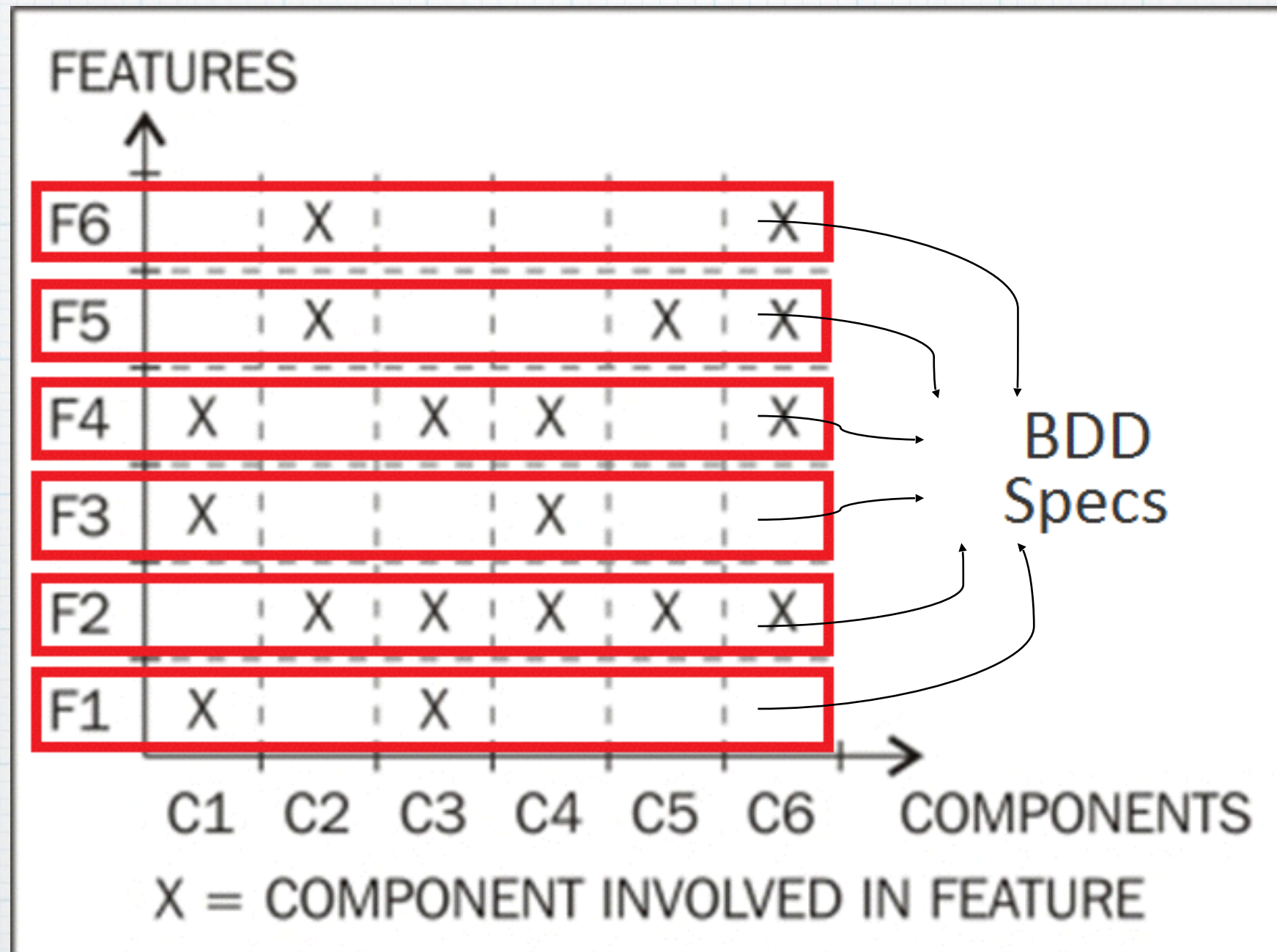
# TDD's Focus



Modified diagram from “Learning Behaviour-driven Development with JavaScript”



# BDD's Focus



Modified diagram from “Learning Behaviour-driven Development with JavaScript”



# My beliefs

Based on my experience,  
I believe that Test Automation,  
when used properly,  
results in delivering better quality code  
in less time.



# Let's Get Started Now...

## To Do List Tracker

Enter description, then click the Add Task button

Add Task

### To Do List

☐ Write a test

### Completed Tasks

☒ Shave yak



# An Outside-In Approach

- \* Specification By Example/  
Behavior Driven Development  
(Cucumber JS)
- \* Test Driven Development  
(JUnit)



# To Do List Goal

**As a** human with too much on my mind

**I want** an online To Do List tracker

**So that** I don't have to remember it all



# Create a Feature File

Feature: Add items to my To Do List

As a human with too much on my mind

I want to add items to my To Do List

So that I don't have to remember them all



# Create a Feature File

Scenario: Adding item

Given an empty list

When I add "Write a test" to the list

Then it should be displayed in the To Do area

When I add "Shave yak" to the list

Then it should be displayed in the To Do area

And it should be under "Write a test"



# Cucumber-JS Results

```
bth0624$ ./node_modules/.bin/cucumber.js
```

```
Feature: Add items to my To Do List
```

```
As a human with too much on my mind  
I want to add items to my To Do List  
So that I don't have to remember them all
```

```
Scenario: Adding items
```

```
? Given an empty list  
? When I add "Write a test" to the list  
? Then it should be displayed in the To Do area  
? When I add "Shave yak" to the list  
? Then it should be displayed in the To Do area  
? And it should be under "Write a test"  
? And the Done area should be empty
```

```
Warnings:
```

```
1) Scenario: Adding items - features/ToDoList.feature:7  
Step: Given an empty list - features/ToDoList.feature:8  
Message:
```

```
Undefined. Implement with the following snippet:
```

```
this.Given(/^an empty list$/, function (callback) {  
  // Write code here that turns the phrase above into concrete actions  
  callback(null, 'pending');  
});
```

```
2) Scenario: Adding items - features/ToDoList.feature:7
```



# Create Steps Defn file

```
module.exports = function() {  
  this.Given(/^an empty list$/, function () {  
    return 'pending';  
  });  
  
  this.When(/^I add "Write a test" to the list$/, function () {  
    return 'pending';  
  });  
  
  this.Then(/^it should be displayed in the To Do area$/, function () {  
    return 'pending';  
  });  
  
  this.When(/^I add "Shave yak" to the list$/, function () {  
    return 'pending';  
  });  
  
  this.Then(/^it should be under "Write a test"$/, function () {  
    return 'pending';  
  });  
  
  this.Then(/^the Done area should be empty$/, function () {  
    return 'pending';  
  });  
};
```



# Create Steps Defn file

```
module.exports = function() {  
  this.Given(/^an empty list$/, function () {  
    return 'pending';  
  });  
  
  this.When(/^I add "Write a test" to the list$/, function () {  
    return 'pending';  
  });  
  
  this.Then(/^it should be displayed in the To Do area$/, function () {  
    return 'pending';  
  });  
  
  this.When(/^I add "Shave yak" to the list$/, function () {  
    return 'pending';  
  });  
  
  this.Then(/^it should be under "Write a test"$/, function () {  
    return 'pending';  
  });  
  
  this.Then(/^the Done area should be empty$/, function () {  
    return 'pending';  
  });  
};
```



# Create Steps Defn file

```
module.exports = function() {  
  this.Given(/^an empty list$/, function () {  
    return 'pending';  
  });  
  
  this.When(/^I add "(.*)" to the list$/, function (task) {  
    return 'pending';  
  });  
  
  this.Then(/^it should be displayed in the To Do area$/, function () {  
    return 'pending';  
  });  
  
  this.Then(/^it should be under "(.*)"$/, function (task) {  
    return 'pending';  
  });  
  
  this.Then(/^the Done area should be empty$/, function () {  
    return 'pending';  
  });  
};
```



# Cucumber-JS Results

```
bth0624$ ./node_modules/.bin/cucumber.js
```

```
Feature: Add items to my To Do List
```

```
As a human with too much on my mind  
I want to add items to my To Do List  
So that I don't have to remember them all
```

```
Scenario: Adding items
```

```
? Given an empty list
```

- When I add "Write a test" to the list
- Then it should be displayed in the To Do area
- When I add "Shave yak" to the list
- Then it should be displayed in the To Do area
- And it should be under "Write a test"
- And the Done area should be empty

```
Warnings:
```

```
1) Scenario: Adding items - features/ToDoList.feature:7  
  Step: Given an empty list - features/ToDoList.feature:8  
  Step Definition: features/step-definitions/ToDoList.step_def.js:9  
  Message:  
    Pending
```

```
1 scenario (1 pending)  
7 steps (1 pending, 6 skipped)  
0m00.001s
```



# Update Steps Defn file

```
module.exports = function() {  
  this.When(/^I add "(.*)" to the list$/, function (taskDescription) {  
    this.taskAdded = taskDescription;  
    return mainPage.addTask(taskDescription);  
  });  
  
  this.Then(/^it should be displayed in the To Do area$/, function () {  
    return expect(mainPage.todoListContains(this.taskAdded)).to.be.true;  
  });  
  
  this.Then(/^it should be under "(.*)"$/, function (task) {  
    let indexOne = mainPage.indexOfTask(task);  
    let indexTwo = mainPage.indexOfTask(this.taskAdded);  
    return expect(indexOne).to.be.(indexTwo-1);  
  });  
};
```



# Server Side

- \* Web Service used by the To Do Client
- \* Methods to add a task, find a task by index, and get the task list
- \* Use TDD to design the classes



# Uncle Bob's Three Laws

1. You may not write any production code until you have first written a failing unit test.
2. You may not write more of a unit test than is sufficient to fail, and not compiling is failing.
3. You may not write more production code than is sufficient to pass the currently failing unit test.



# List Tracker Unit Test

```
public class ListTrackerSpec {  
  
    @Test  
    public void taskAddedToListShouldBeRetrievable() {  
        ListTracker listTracker = new ListTracker(); // Failed - missing class  
        String taskDescription = "Shave Yak";  
  
        listTracker.addTask( taskDescription ); // Fail - missing method  
        List<String> tasks = listTracker.getTasks(); // Fail - missing method  
  
        assertTrue( tasks.contains(taskDescription) ); // Fail for 2 reasons  
        // getTasks() initially returned null and addTasks() did nothing  
    }  
}
```



# List Tracker Source

```
public class ListTracker {  
    private List<String> tasks = new ArrayList<String>();  
  
    public void addTask( String taskDescription ) {  
        this.tasks.add( taskDescription );  
    }  
  
    public List<String> getTasks() {  
        return this.tasks;  
    }  
  
    public String getTasksByIndex(int indexToRetrieve ) {  
        return tasks.get(indexToRetrieve);  
    }  
}
```



# Cucumber-JS Results

```
ToDoListClient$ ./node_modules/.bin/cucumber.js
```

```
Feature: Add items to my To Do List
```

```
As a human with too much on my mind  
I want to add items to my To Do List  
So that I don't have to remember them all
```

```
Scenario: Adding items
```

- ✓ Given an empty list
- ✓ When I add "Write a test" to the list
- ✓ Then it should be displayed in the To Do area
- ✓ When I add "Shave yak" to the list
- ✓ Then it should be displayed in the To Do area
- ✓ And it should be under "Write a test"



# Resources

**“Clean Code: A Handbook of Agile Software Craftsmanship” by Robert C. Martin**

**“Growing Object-Oriented Software, Guided by Tests” by Steve Freeman and Nat Pryce**

**“Specification By Example” by Gojko Adzic**

**“BDD in Action” by John Ferguson Smart**



# Questions & Contact Info

Feedback, questions, comments to:  
burk.hufnagel@daugherty.com

Twitter: @burkhufnagel

Come to the monthly Atlanta Java User  
Group meetings. Info at [ajug.org](http://ajug.org)