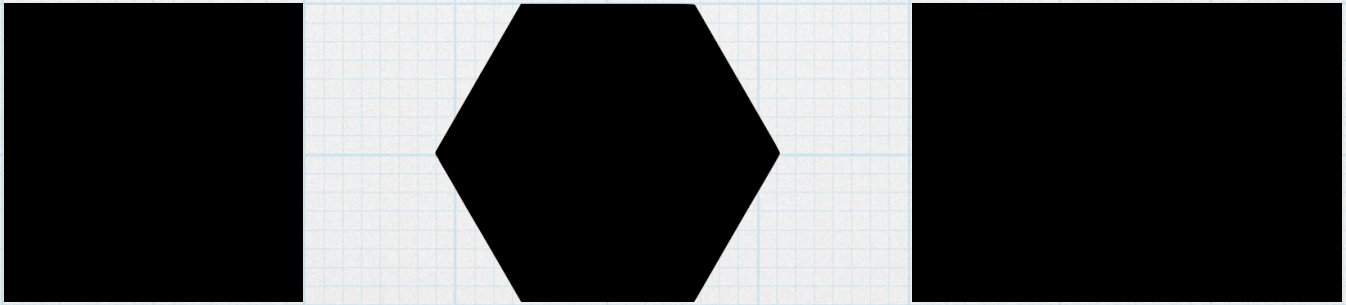
# Want To Succeed With TDD?

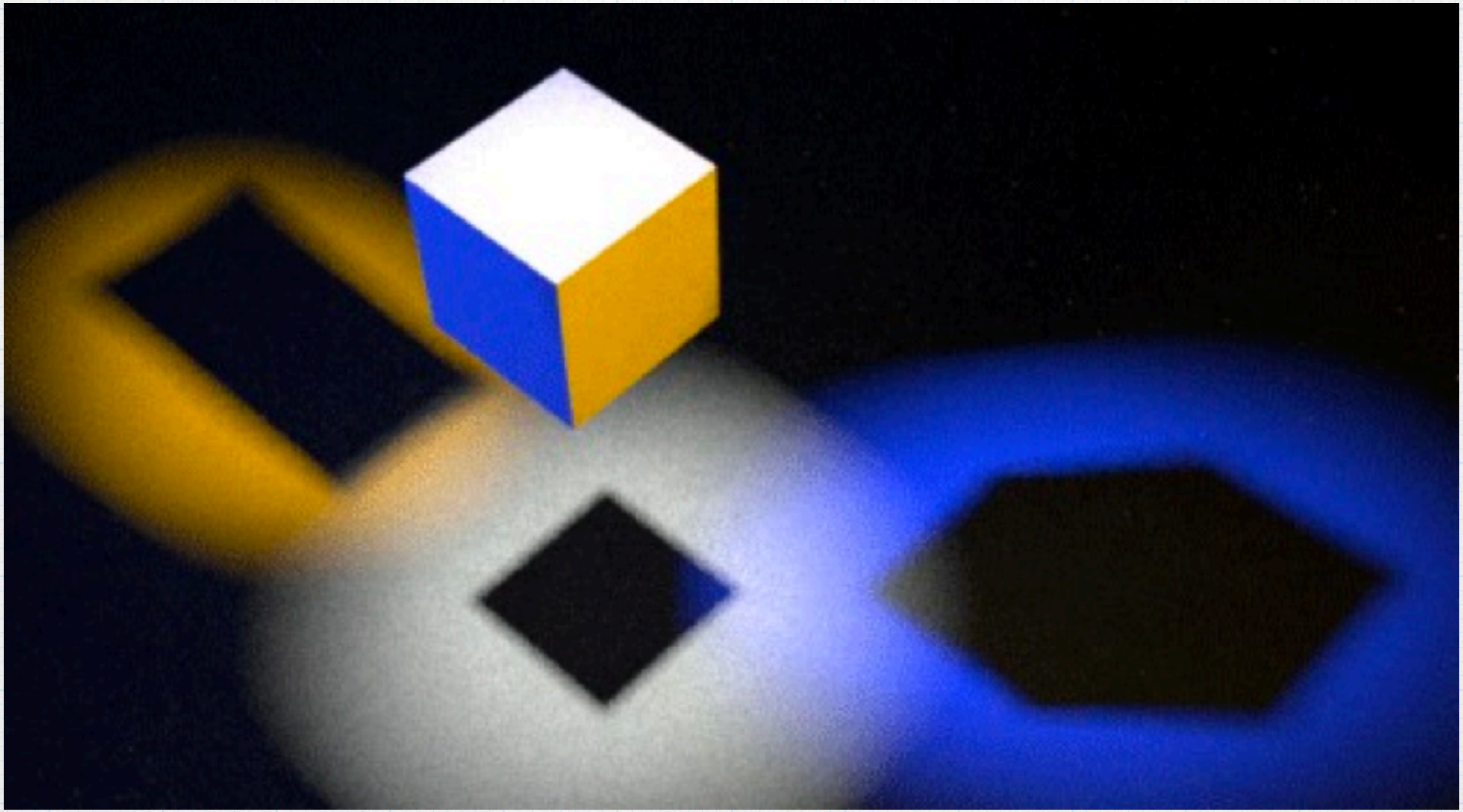
## Stop Writing Tests!

Burk Hufnagel - Solution Architect
Daugherty Business Solutions

# Thought Experiment

What 3-D shape could cast these shadows?

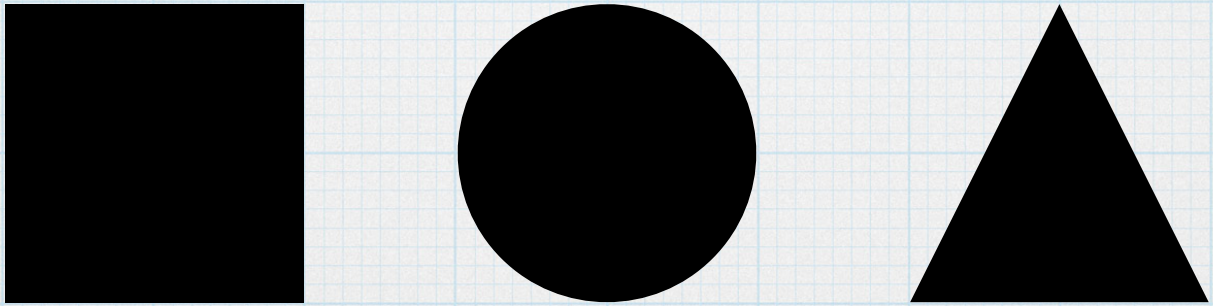


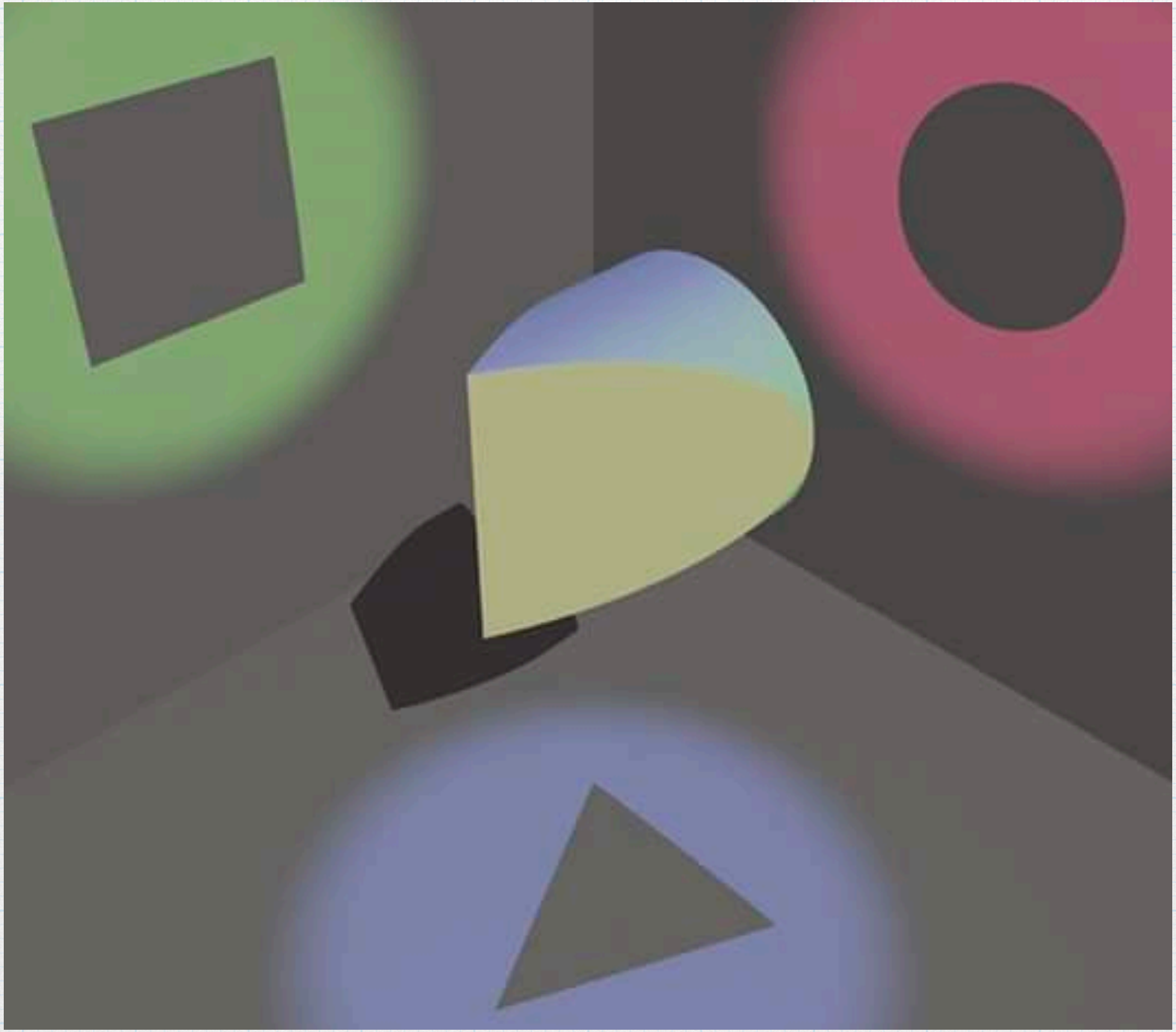


http://m759.net/wordpress/?p=49811

# Thought Experiment

## What about these shadows?





https://halilibrahimak.wordpress.com/tag/aile-danismani/page/10

Changing how you look at, or think about, something can also change how you perceive it...

# Want To Succeed With TDD?

## Stop Writing Tests!

Burk Hufnagel - Solution Architect
Daugherty Business Solutions

# Who is Burk Hufnagel?

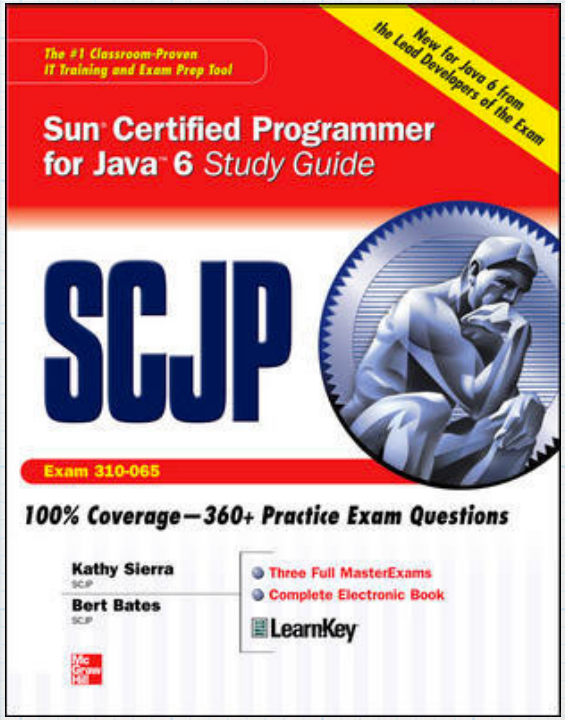
## Programmer and Solution Architect with

## Daugherty Business Solutions

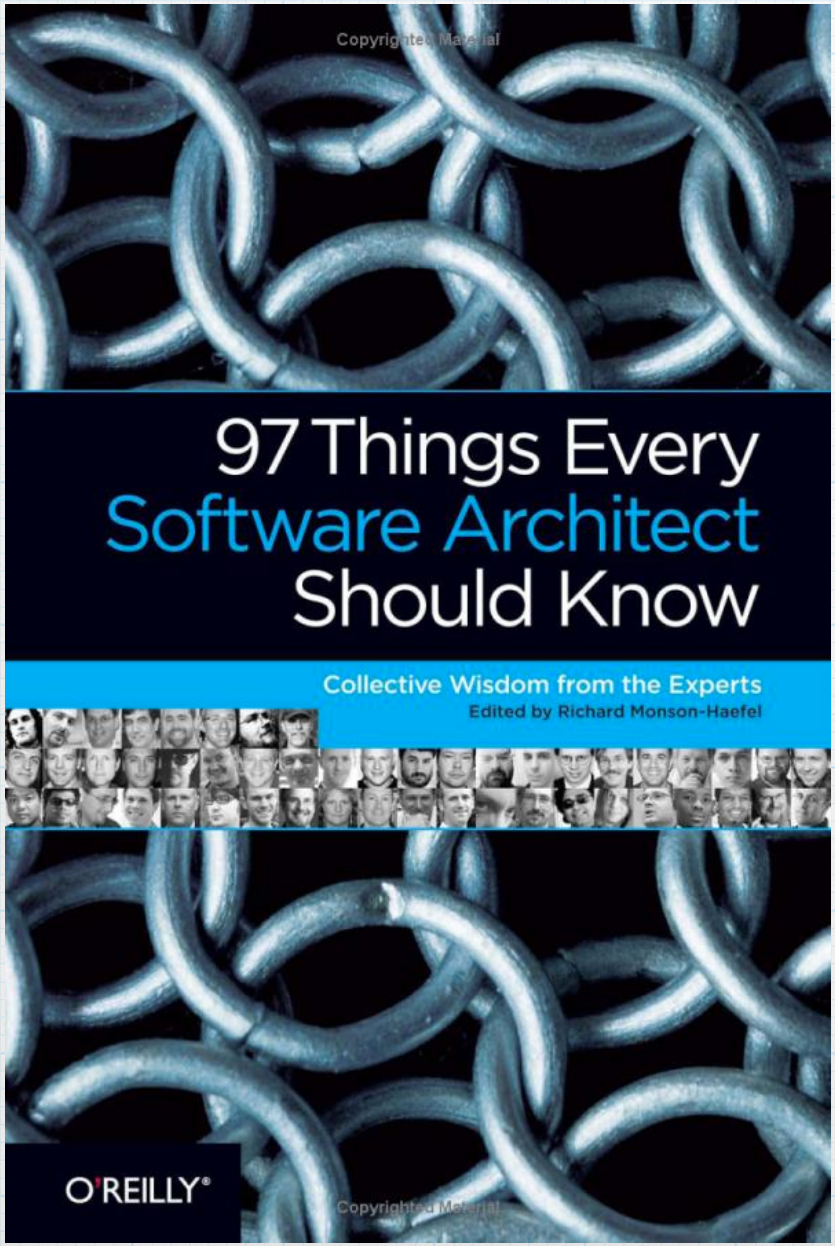
Tech Reviewer: SCJP6 Study Guide

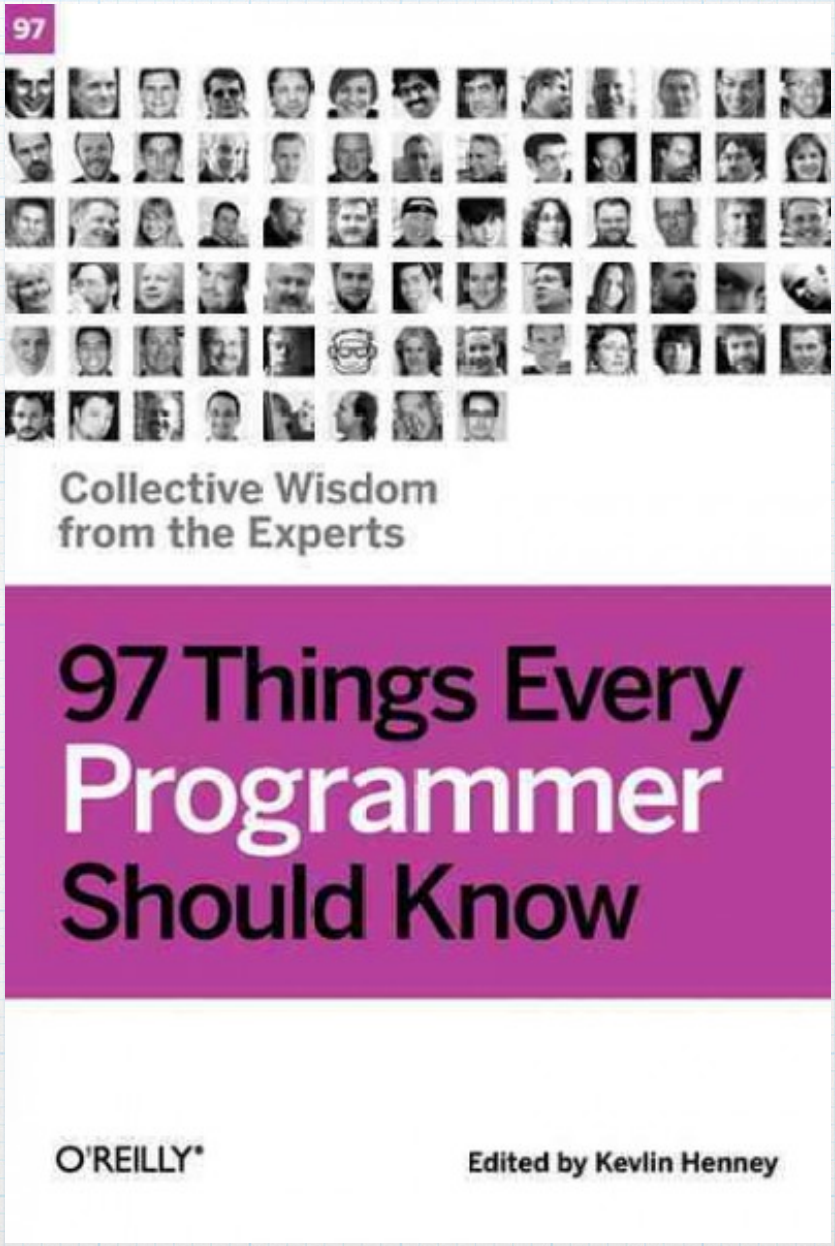
"Burk fixed more code than we care to admit."

Kathy Sierra & Bert Bates



# Who is Burk Hufnagel?



# Why are you here?

# Deliver Better Code, Faster

# Why are you here?

## Deliver

Writing code doesn't create business value

## Better Code

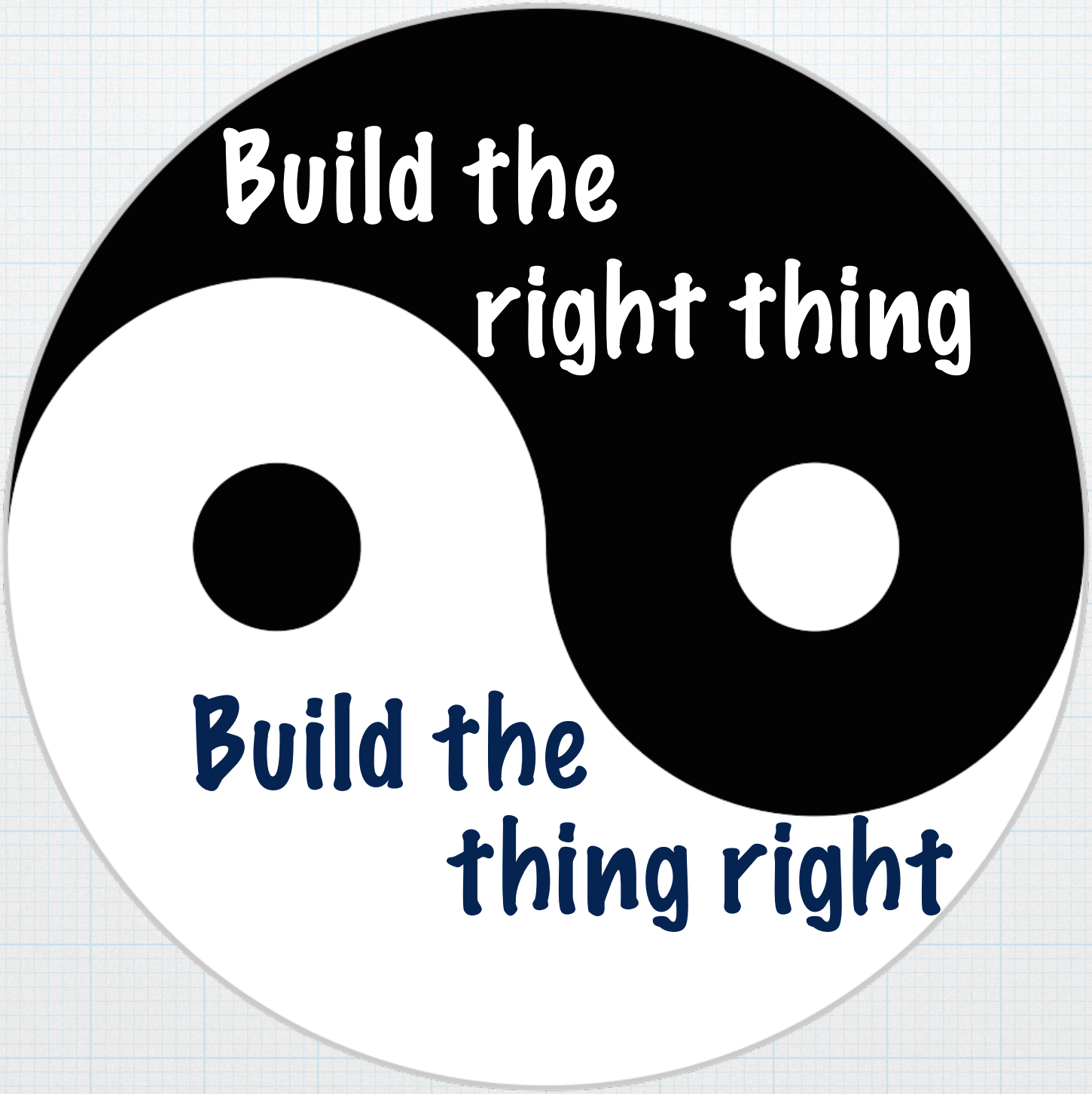
Solving the business problem is not enough

## Faster

TDD is a key step towards CI/CD or DevOps

# Solving the Business Problem

* Build the right thing
* Build the thing right


Build the
right thing
Build the
thing right

# Why are you here?

## Delivering

Writing code doesn't create business value

## Better Code

Solving the business problem is not enough

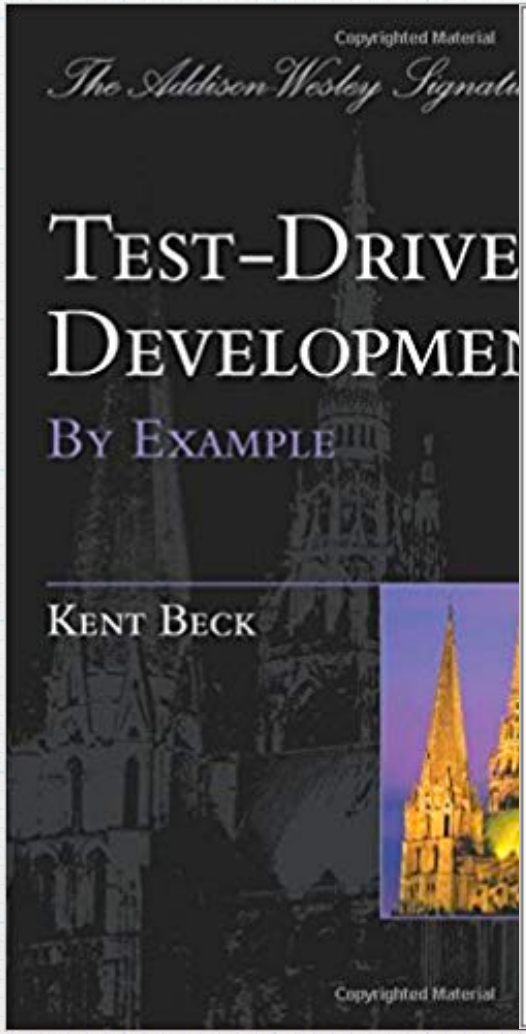
## Faster

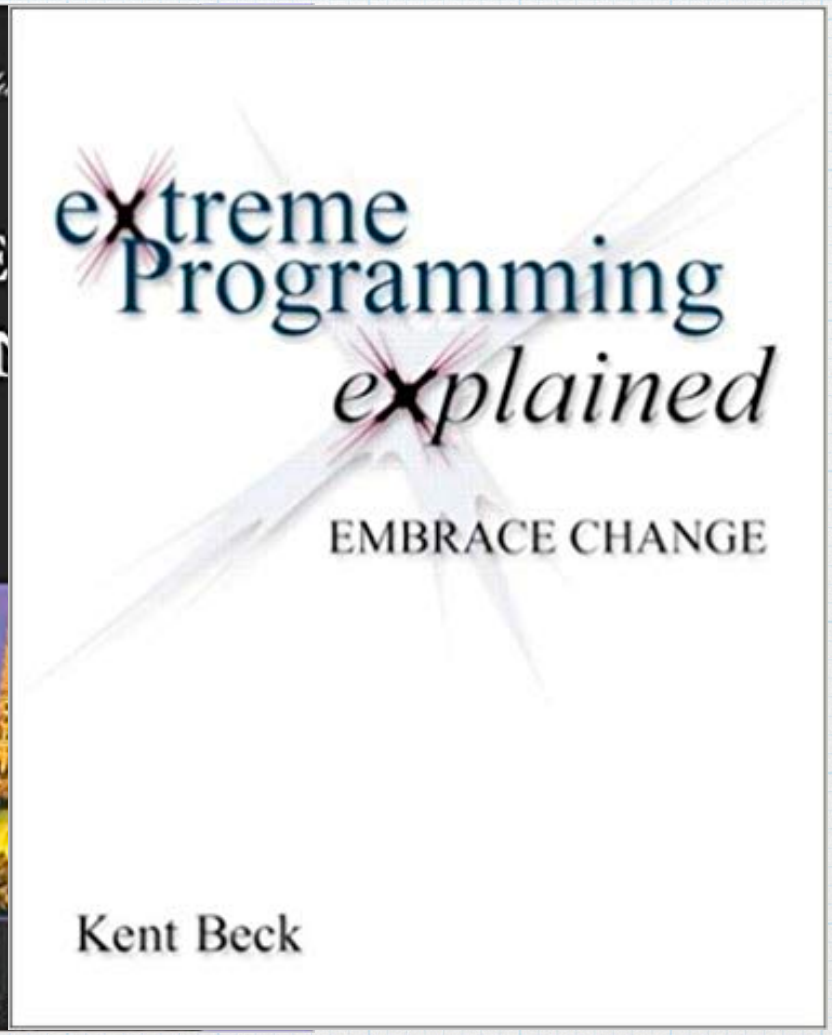
TDD is a key step towards CI/CD or DevOps

# Agenda

* ~~Introduction~~
* What is TDD?
* Why should you care?
* Successfully adopting TDD
* Summary

# What is TDD?


Copyrighted Material
The Addison-Wesley Signatu
TEST-DRIVE
DEVELOPMEN
BY EXAMPLE
KENT BECK
Copyrighted Material


extreme
Programming
explained
EMBRACE CHANGE
Kent Beck

# What is TDD?

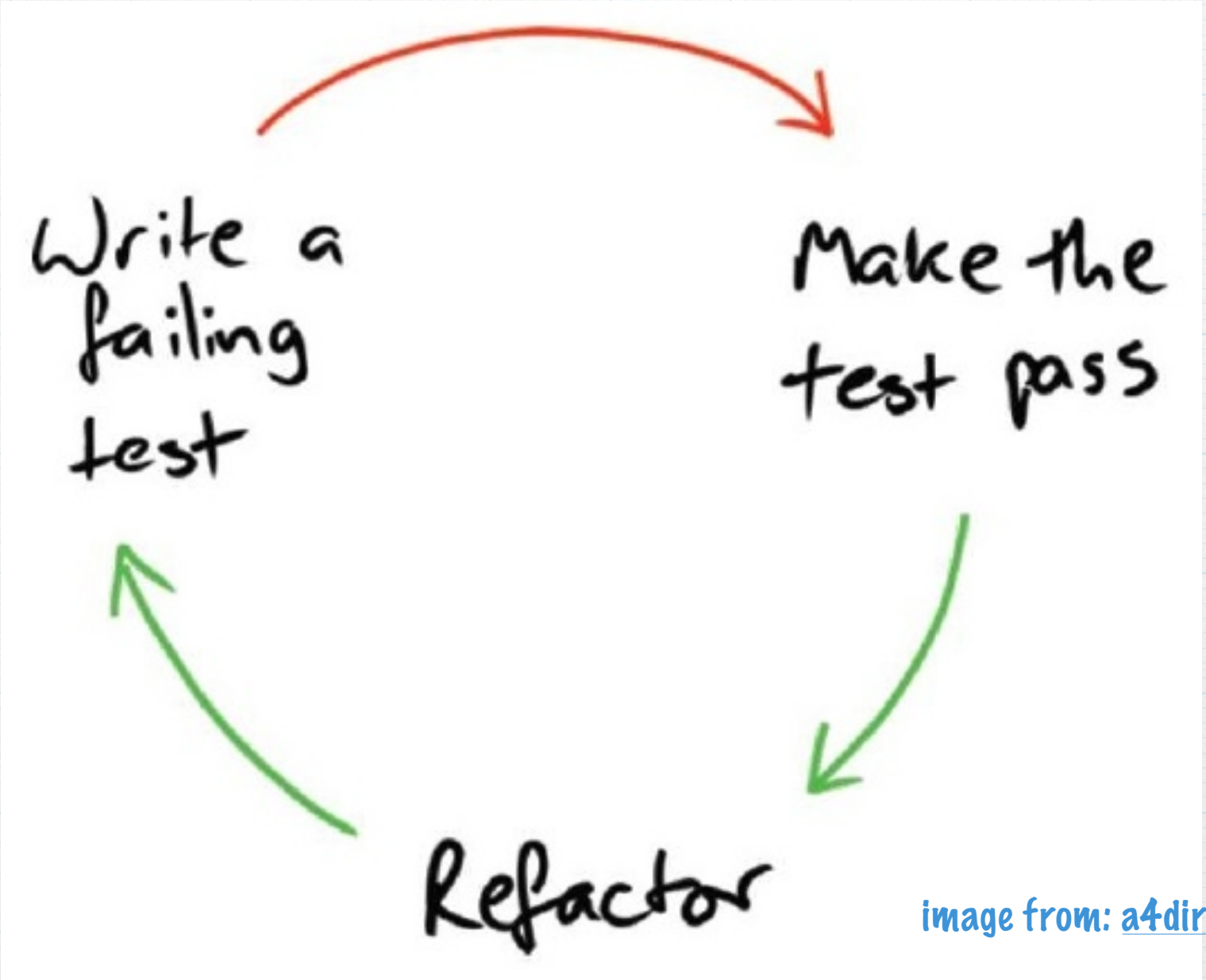
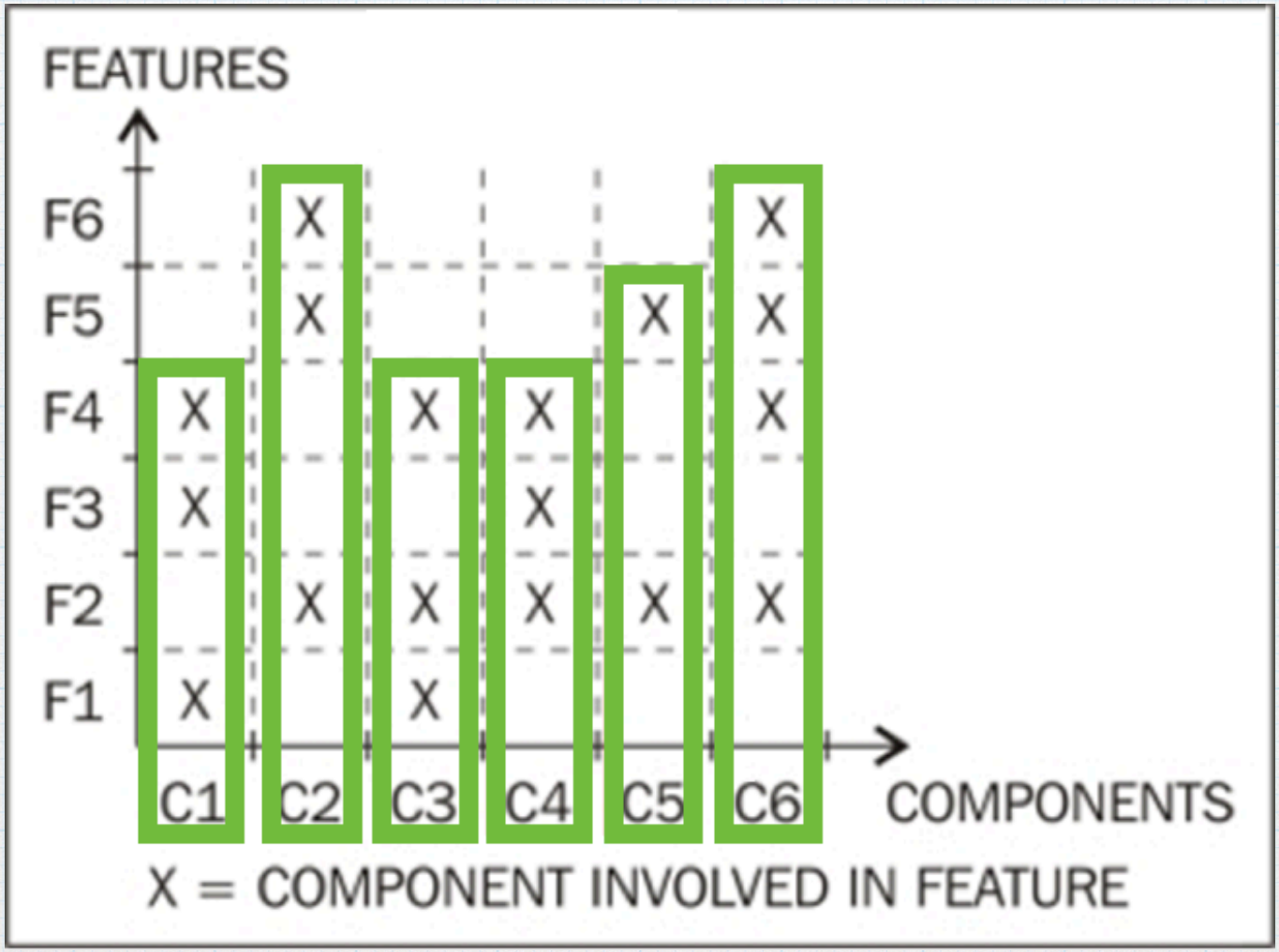



image from: a4direct.com

# What is TDD?




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

# Agenda

* ~~Introduction~~
* ~~What is TDD?~~
* Why should you care?
* Successfully adopting TDD
* Summary

# Why Should You Care?

* Fewer bugs in your code
* Spend less time debugging
* Deliver to production faster
* Higher code quality
* A key step towards CI/DI & DevOps
* More opportunities to think

# Why Should You Care?

## Programming is thinking, not typing

From: https://www.indiatoday.in/education-today/gk-current-affairs/story/brain-reacts-to-words-320501-2016-04-29

# Agenda

* ~~Introduction~~
* ~~What is TDD?~~
* ~~Why should you care?~~
* Successfully adopting TDD
* Summary

# Successfully Adopting TDD

Looks easy...

Lots of benefits...

So why isn't everyone doing it?

# Unreasonable Expectations

* There's only three steps...
  * Requires changing your habits
* Sounds too good to be true
* Lack of social proof
* Unintentional bad examples

# Some Truths about TDD

* Thinking is required
* You'll still write bugs, but finding & fixing them is easier
* Changing habits is **hard**, but having a good **reason** to change makes it easier and having a **partner** makes it more likely

# TDD is like riding a bike

From: http://thismamamakesstuff.com/wp-content/uploads/2010/05/IMG_0513-900x600.jpg

# TDD is like riding a bike

From: https://www.wsj.com/video/how-to-ride-a-bike-for-adults/AFA3015A-FCD1-413B-A9B3-00CAB866E566.html

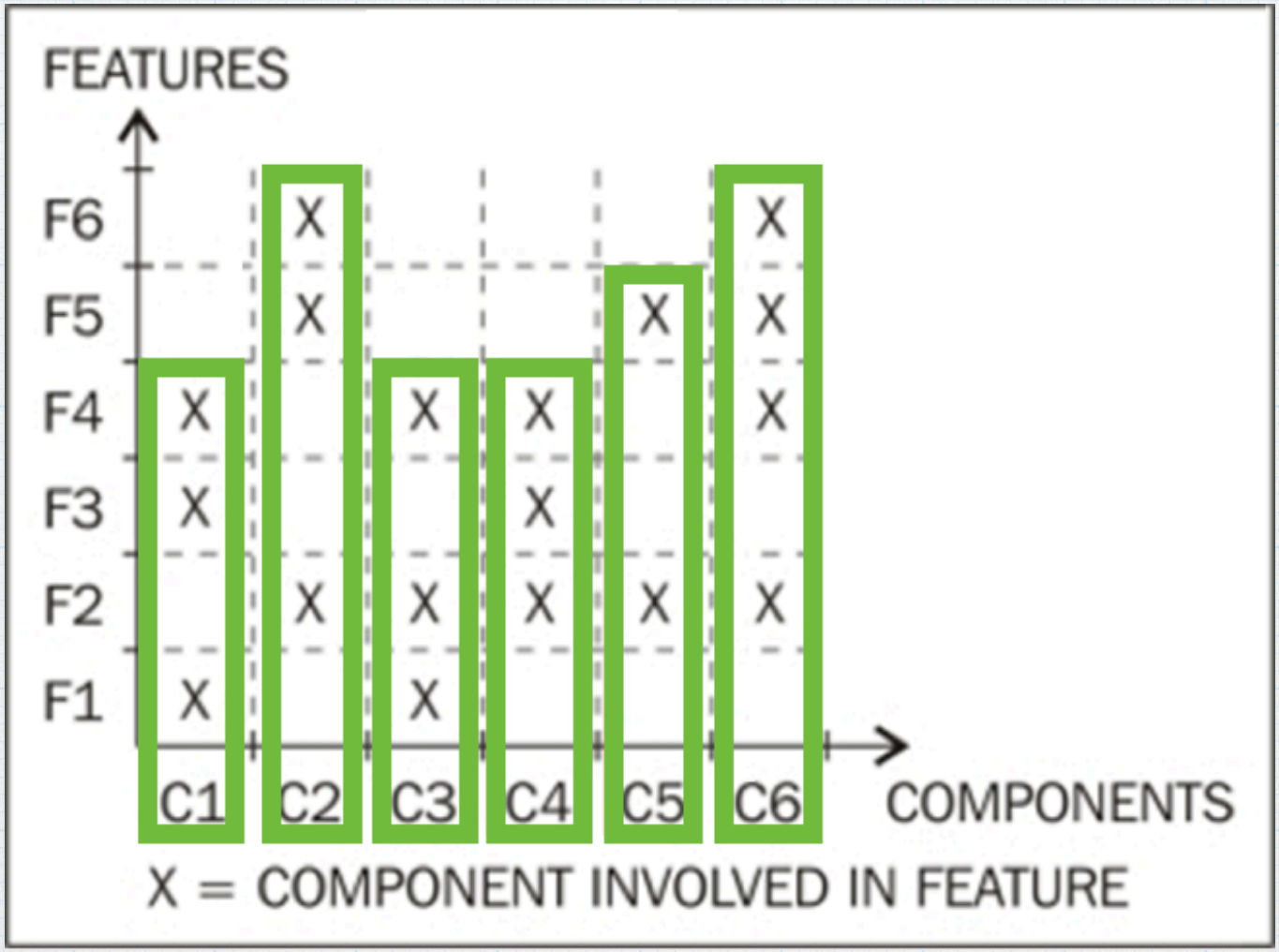
# TDD is like riding a bike

From: https://i0.wp.com/farm4.staticflickr.com/3844/14468892960_a22582ca66.jpg

# What's wrong with "Test"

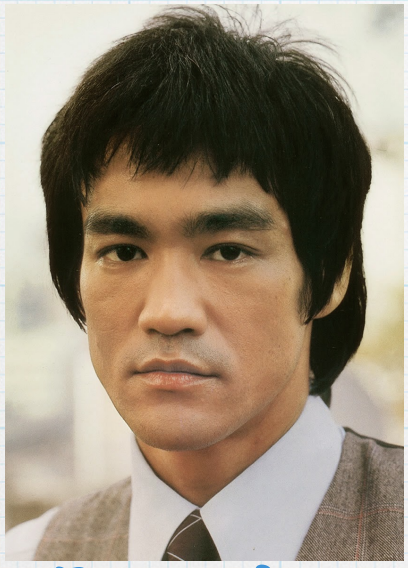
* Writing tests first is illogical
  * But, writing them last is hard
* Tests are limited to pass/fail so they provide little information
* TDD is focused on unit tests
  * Necessary but not sufficient to deliver better software faster

# TDD Focus on Units




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

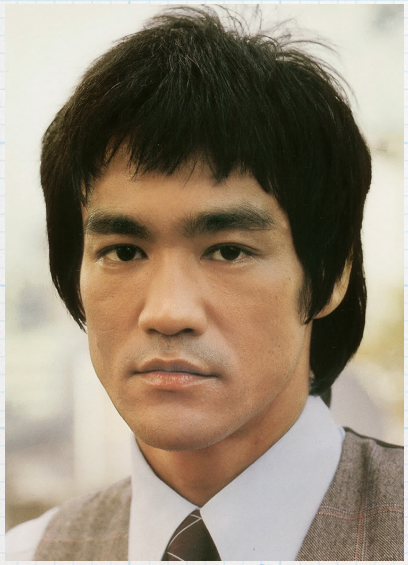
# What about tests?



Bruce Lee
Early TDD adopter

No books on Agile or Automated Testing. However, we do have a recording...

# What about tests?

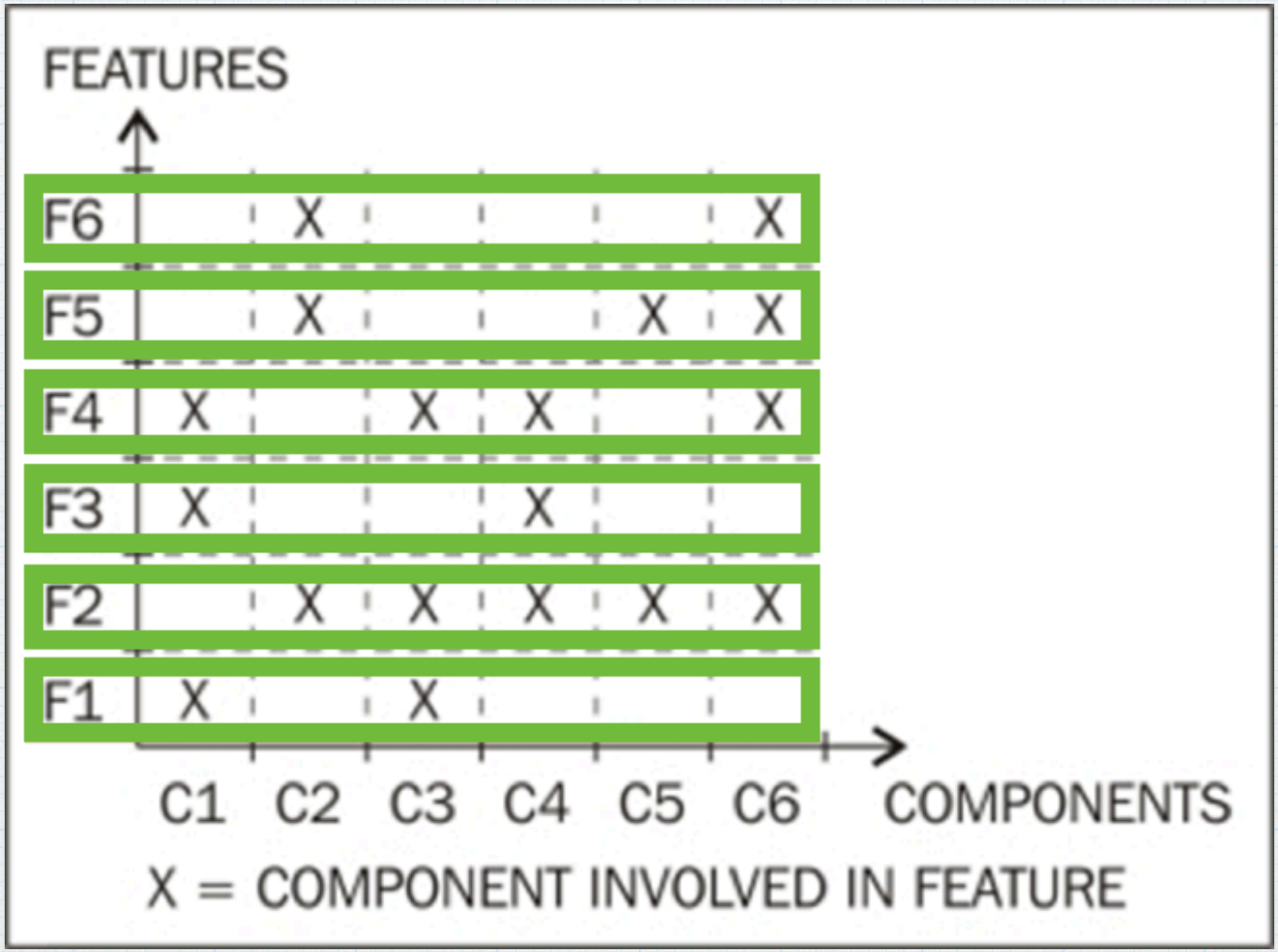


Bruce Lee
Early BDD adopter

"It is like a finger,
pointing the way
to the moon...

Don't concentrate
on the finger,
or you will miss
all that heavenly glory."

# Should Focus on Features




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

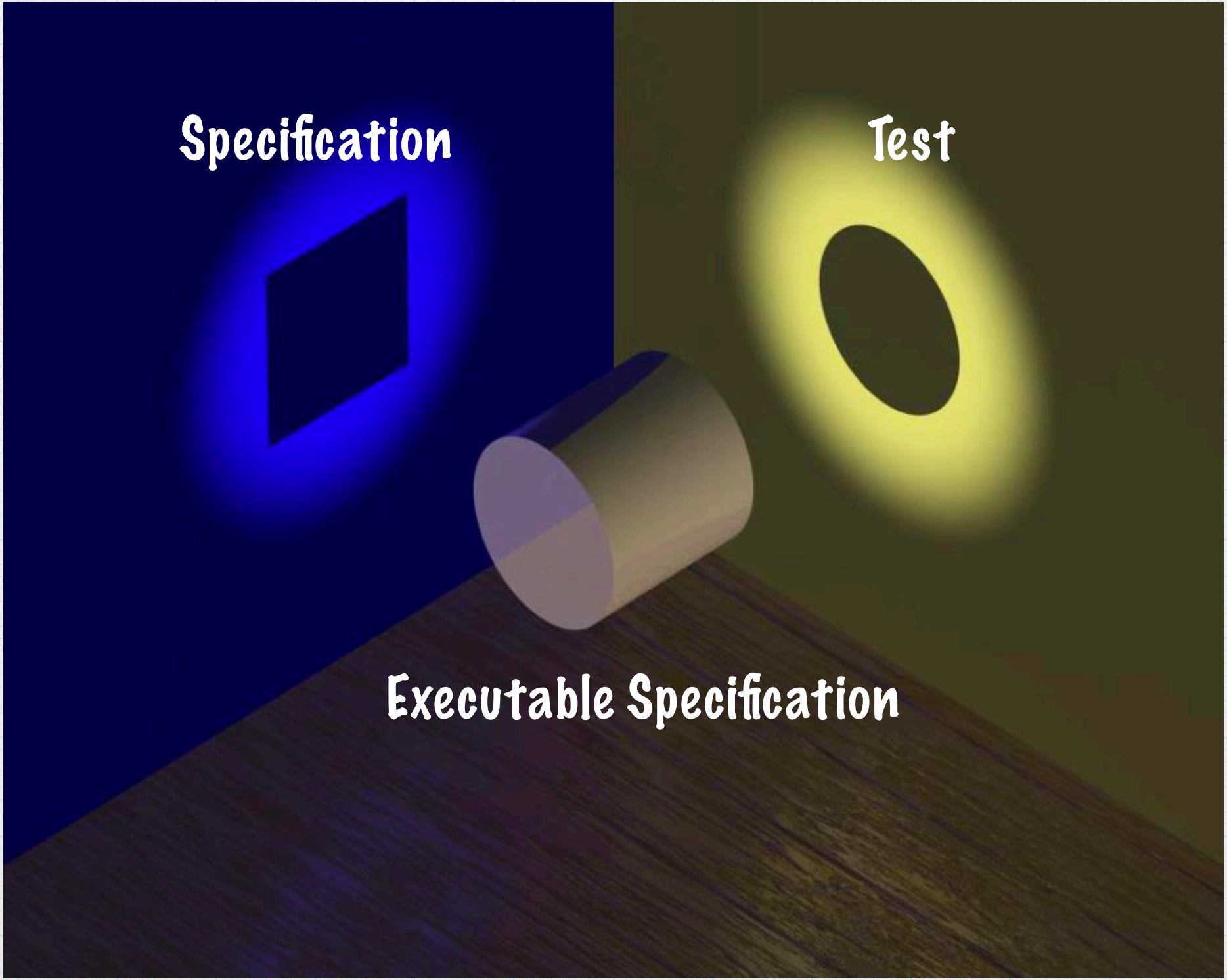
# Write Specifications

* Based on requirements
* Describe what should happen
* You must understand the details: no guessing, ask business folks
* Lets you separate design from implementation

But don't stop here...

# Executable Specifications

* Adds code to the spec so it can be run, like a test, against production code
* Can generate living documentation for business & technical folks
* Bridges the gap between designing and implementing solutions


Specification
Test
Executable Specification

# The Rhythm of TDD

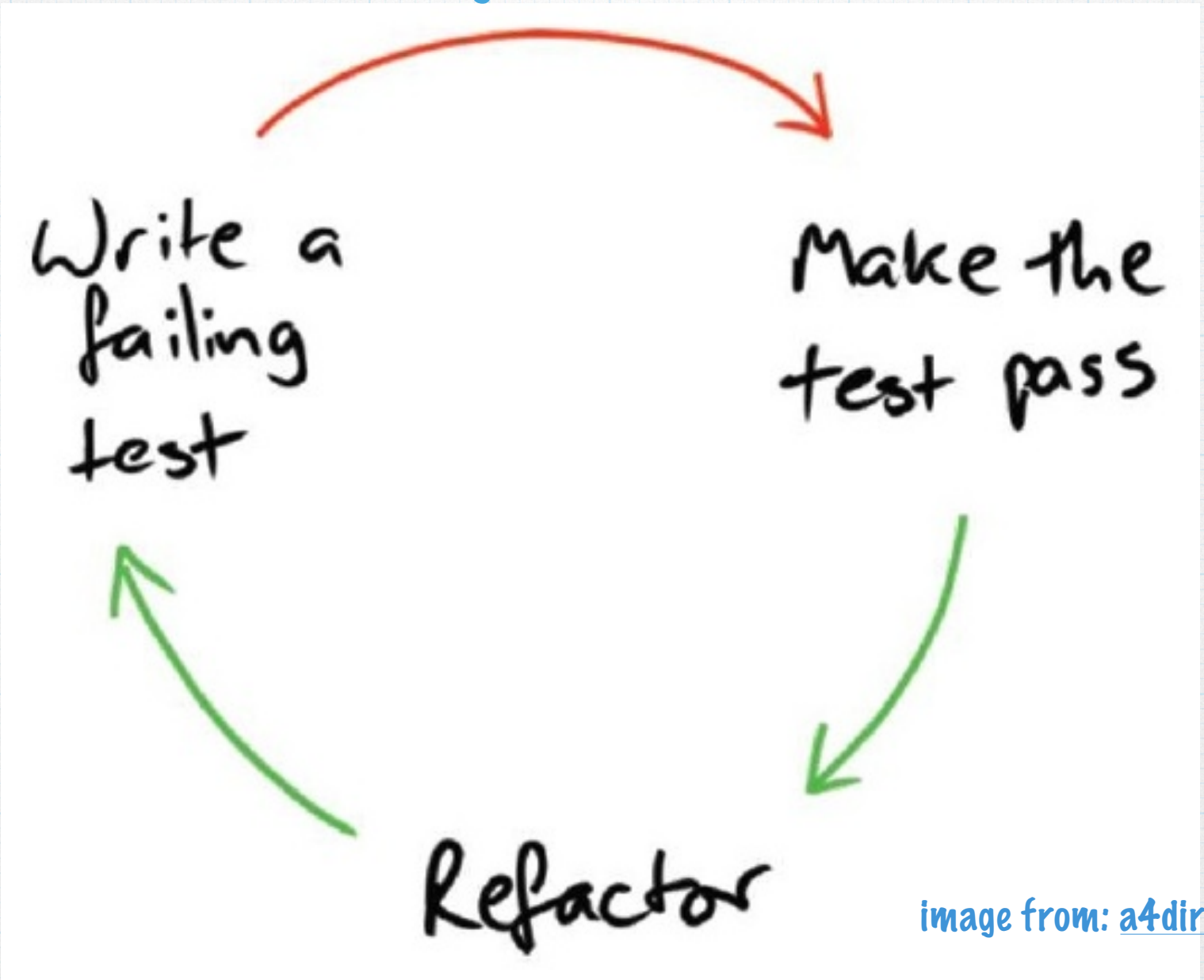



image from: a4direct.com

# 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.

# Behavior Driven Development

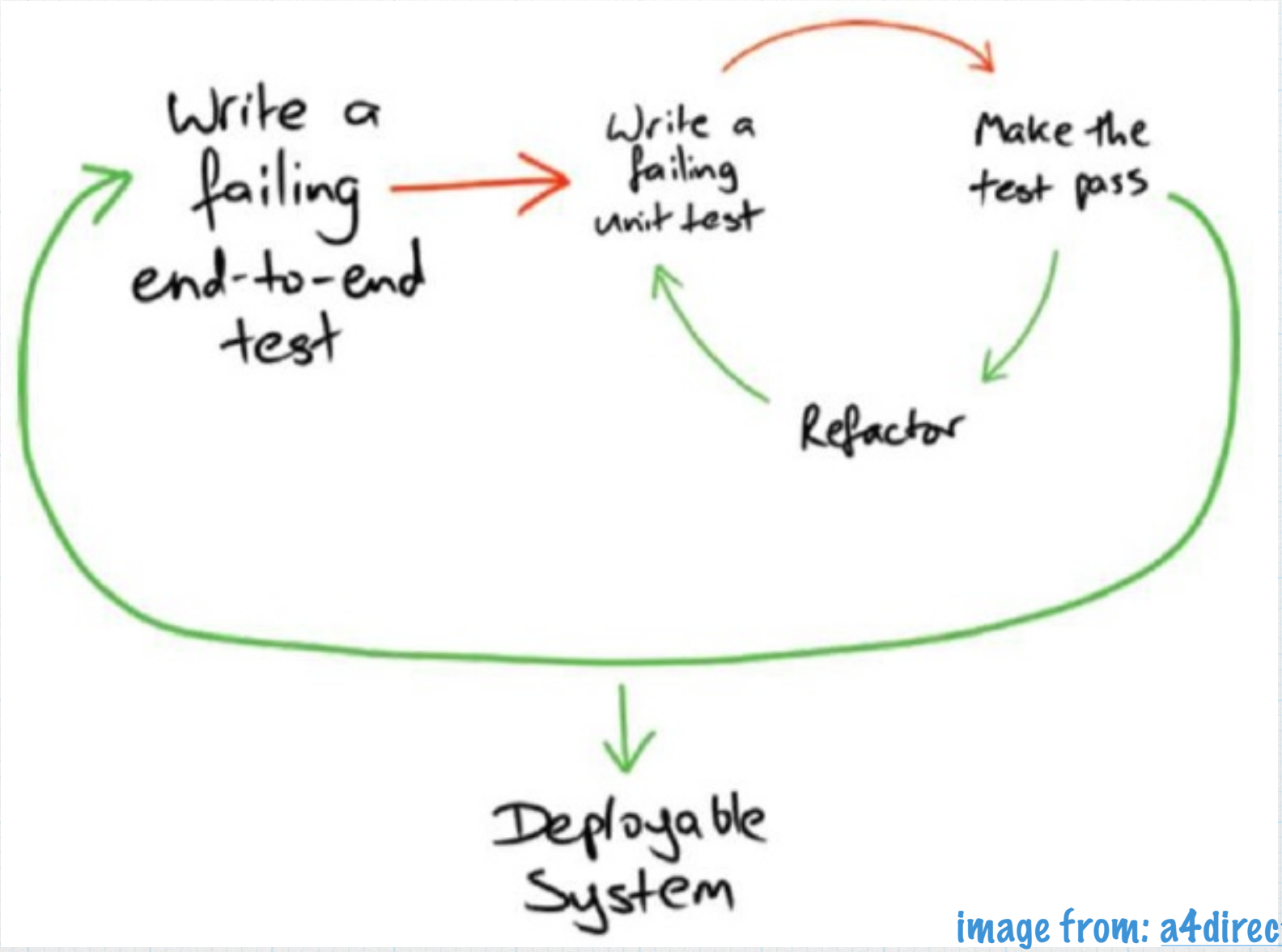



image from: a4direct.com

# To Do List Goal

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

# Create a Scenario

## Scenario: Adding items

Given an empty list
When I add "Shave yak" to the list
Then it should be displayed in the To Do area
When I add "Adopt TDD" to the list
Then it should be displayed in the To Do area
And it should be under "Shave yak"

# Scenarios describe behavior

Scenario: Adding items

Given an empty list

When I add "Shave yak" to the list

Then it should be displayed in the To Do area

## Step definitions describe lower level behavior...

```
this.Given(/^an empty list$/, function () {
    todoList = new ToDoList();
});

this.When(/^I add "Shave yak" to the list$/, function () {
    todoList.addTask( "Shave yak" );
});

this.Then(/^it should be displayed in the To Do area$/, function () {
    todoList.shouldDisplayTask( "Shave yak" );
});
```

...which might suggests a unit test.

```
class ListTrackerSpecification extends Specification {
    def "When a task is added to the list, it should contain the task"() {
        given: "an empty task list"

        when: "a task is added to the list"

        then: "the list should contain the task"

    }
}
```

```
class ListTrackerSpecification extends Specification {
    def "When a task is added to the list, it should contain the task"() {
        given: "an empty task list"
            def listTracker = new ListTracker()

        when: "a task is added to the list"

        then: "the list should contain the task"

    }
}
```

```
class ListTrackerSpecification extends Specification {
    def "When a task is added to the list, it should contain the task"() {
        given: "an empty task list"
            def listTracker = new ListTracker()

        when: "a task is added to the list"

        then: "the list should contain the task"

    }
}
```

```
public class ListTracker {

}
```

```
class ListTrackerSpecification extends Specification {
    def "When a task is added to the list, it should contain the task"() {
        given: "an empty task list"
            def listTracker = new ListTracker()

        when: "a task is added to the list"

        then: "the list should contain the task"

    }
}
```

```
public class ListTracker {

}
```

```
class ListTrackerSpecification extends Specification {
   def "When a task is added to the list, it should contain the task"() {
      given: "an empty task list"
         def listTracker = new ListTracker()

      when: "a task is added to the list"
         listTracker.addTask( "Shave Yak" )

      then: "the list should contain the task"

   }
}
```

```
public class ListTracker {

}
```

```
class ListTrackerSpecification extends Specification {
   def "When a task is added to the list, it should contain the task"() {
      given: "an empty task list"
         def listTracker = new ListTracker()

      when: "a task is added to the list"
         listTracker.addTask( "Shave Yak" )

      then: "the list should contain the task"

   }
}
```

```
public class ListTracker {
   public void addTask( String taskDescription ) {

   }
}
```

```
class ListTrackerSpecification extends Specification {
    def "When a task is added to the list, it should contain the task"() {
        given: "an empty task list"
            def listTracker = new ListTracker()

        when: "a task is added to the list"
            listTracker.addTask( "Shave Yak" )

        then: "the list should contain the task"

    }
}
```

```
public class ListTracker {
    public void addTask( String taskDescription ) {
        <= Does nothing. This should cause a failure later…
    }
}
```

```
class ListTrackerSpecification extends Specification {
   def "When a task is added to the list, it should contain the task"() {
      given: "an empty task list"
         def listTracker = new ListTracker()

      when: "a task is added to the list"
         listTracker.addTask( "Shave Yak" )

      then: "the list should contain the task"
         def tasks = listTracker.getTasks();
   }
}
```

```
public class ListTracker {
   public void addTask( String taskDescription ) {

   }
}
```

```
class ListTrackerSpecification extends Specification {
    def "When a task is added to the list, it should contain the task"() {
        given: "an empty task list"
            def listTracker = new ListTracker()

        when: "a task is added to the list"
            listTracker.addTask( "Shave Yak" )

        then: "the list should contain the task"
            def tasks = listTracker.getTasks()
    }
}
```

```
public class ListTracker {
    public void addTask( String taskDescription ) {

    }

    public List<String> getTasks() {

    }
}
```

```
class ListTrackerSpecification extends Specification {
    def "When a task is added to the list, it should contain the task"() {
        given: "an empty task list"
            def listTracker = new ListTracker()

        when: "a task is added to the list"
            listTracker.addTask( "Shave Yak" )

        then: "the list should contain the task"
            def tasks = listTracker.getTasks()
    }
}
```

```
public class ListTracker {
    public void addTask( String taskDescription ) {

    }

    public List<String> getTasks() {

    }
}
```

```
class ListTrackerSpecification extends Specification {
   def "When a task is added to the list, it should contain the task"() {
      given: "an empty task list"
         def listTracker = new ListTracker()

      when: "a task is added to the list"
         listTracker.addTask( "Shave Yak" )

      then: "the list should contain the task"
         def tasks = listTracker.getTasks()
   }
}
```

```
public class ListTracker {
   public void addTask( String taskDescription ) {

   }

   public List<String> getTasks() {
      return null; <== Explicitly invalid result
   }
}
```

```
class ListTrackerSpecification extends Specification {
   def "When a task is added to the list, it should contain the task"() {
      given: "an empty task list"
         def listTracker = new ListTracker()

      when: "a task is added to the list”
         listTracker.addTask( "Shave Yak" )

      then: "the list should contain the task"
         def tasks = listTracker.getTasks()
   }
}
```

```
public class ListTracker {
   public void addTask( String taskDescription ) {

   }

   public List<String> getTasks() {
      return null;
   }
}
```

```
class ListTrackerSpecification extends Specification {
   def "When a task is added to the list, it should contain the task"() {
      given: "an empty task list"
         def listTracker = new ListTracker()

      when: "a task is added to the list”
         listTracker.addTask( "Shave Yak" )

      then: "the list should contain the task"
         def tasks = listTracker.getTasks()
         tasks.contains( "Shave Yak" )
   }
}
```

```
public class ListTracker {
   public void addTask( String taskDescription ) {

   }

   public List<String> getTasks() {
      return null;
   }
}
```

```
class ListTrackerSpecification extends Specification {
    def "When a task is added to the list, it should contain the task"() {
        given: "an empty task list"
            def listTracker = new ListTracker()

        when: "a task is added to the list”
            listTracker.addTask( "Shave Yak" )

        then: "the list should contain the task"
            def tasks = listTracker.getTasks()
            tasks.contains( "Shave Yak" )
    }
}
```

Console | Markers | Progress | JUnit

Finished after 0.034 seconds

Runs: 1/1 Errors: 1 Failures: 0

com.burkhufnagel.ListTrackerSpec [Runner: JUnit 4] (0.001 s)

taskAddedToListShouldBeRetrievable (0.001 s)

Failure Trace

java.lang.NullPointerException

at com.burkhufnagel.ListTrackerSpec.taskAddedToListShouldBeRetrievabl

```
    public List<String> getTasks() {
        return null;
    }
}
```

```
class ListTrackerSpecification extends Specification {
   def "When a task is added to the list, it should contain the task"() {
      given: "an empty task list"
         def listTracker = new ListTracker()

      when: "a task is added to the list”
         listTracker.addTask( "Shave Yak" )

      then: "the list should contain the task"
         def tasks = listTracker.getTasks()
         tasks.contains( "Shave Yak" )
   }
}
```

```
public class ListTracker {
   public void addTask( String taskDescription ) {

   }

   public List<String> getTasks() {
      return null;
   }
}
```

```
class ListTrackerSpecification extends Specification {
   def "When a task is added to the list, it should contain the task"() {
      given: "an empty task list"
         def listTracker = new ListTracker()

      when: "a task is added to the list”
         listTracker.addTask( "Shave Yak" )

      then: "the list should contain the task"
         def tasks = listTracker.getTasks()
         tasks.contains( "Shave Yak" )
   }
}
```

```
public class ListTracker {
   public void addTask( String taskDescription ) {

   }

   public List<String> getTasks() {
      String[] todos = {”Shave Yak”};
      return Arrays.asList( todos );
   }
}
```

```
class ListTrackerSpecification extends Specification {
   def "When a task is added to the list, it should contain the task"() {
      given: "an empty task list"
         def listTracker = new ListTracker()

      when: "a task is added to the list”
         listTracker.addTask( "Shave Yak" )

      then: "the list should contain the task"
         def tasks = listTracker.getTasks()
         tasks.contains( "Shave Yak" )
   }
}
```

```
public class ListTracker {
   public void addTask( String taskDescription ) {

   }

   public List<String> getTasks() {
      return new ArrayList<String>();
   }
}
```

```
class ListTrackerSpecification extends Specification {
    def "When a task is added to the list, it should contain the task"() {
        given: "an empty task list"
            def listTracker = new ListTracker()

        when: "a task is added to the list”
            listTracker.addTask( "Shave Yak" )

        then: "the list should contain the task"
            def tasks = listTracker.getTasks()
            tasks.contains( "Shave Yak" )
    }
}
```

Console | Markers | Progress | JUnit

Finished after 0.019 seconds

Runs: 1/1 Errors: 0 Failures: 1

com.burkhufnagel.ListTrackerSpec [Runner: JUnit 4] (0.000 s)
taskAddedToListShouldBeRetrievable (0.000 s)

Failure Trace
java.lang.AssertionError
at com.burkhufnagel.ListTrackerSpec.taskAddedToListShouldBeRetrievab

```
    public List<String> getTasks() {
        return new ArrayList<String>();
    }
}
```

```
class ListTrackerSpecification extends Specification {
   def "When a task is added to the list, it should contain the task"() {
      given: "an empty task list"
         def listTracker = new ListTracker()

      when: "a task is added to the list”
         listTracker.addTask( "Shave Yak" )

      then: "the list should contain the task"
         def tasks = listTracker.getTasks()
         tasks.contains( "Shave Yak" )
   }
}
```

```
public class ListTracker {
   public void addTask( String taskDescription ) {

   }

   public List<String> getTasks() {
      return new ArrayList<String>();
   }
}
```

```
class ListTrackerSpecification extends Specification {
   def "When a task is added to the list, it should contain the task"() {
      given: "an empty task list"
         def listTracker = new ListTracker()

      when: "a task is added to the list”
         listTracker.addTask( "Shave Yak" )

      then: "the list should contain the task"
         def tasks = listTracker.getTasks()
         tasks.contains( "Shave Yak" )
   }
}
```

```
public class ListTracker {
   private List<String> tasks = new ArrayList<>();

   public void addTask( String taskDescription ) {

   }

   public List<String> getTasks() {
      return tasks;
   }
}
```

```
class ListTrackerSpecification extends Specification {
   def "When a task is added to the list, it should contain the task"() {
      given: "an empty task list"
         def listTracker = new ListTracker()

      when: "a task is added to the list”
         listTracker.addTask( "Shave Yak" )

      then: "the list should contain the task"
         def tasks = listTracker.getTasks()
         tasks.contains( "Shave Yak" )
   }
}
```

```
public class ListTracker {
   private List<String> tasks = new ArrayList<>();

   public void addTask( String taskDescription ) {
      this.tasks.add( taskDescription );
   }

   public List<String> getTasks() {
      return tasks;
   }
}
```

```
class ListTrackerSpecification extends Specification {
    def "When a task is added to the list, it should contain the task"() {
        given: "an empty task list"
            def listTracker = new ListTracker()

        when: "a task is added to the list”
            listTracker.addTask( "Shave Yak" )

        then: "the list should contain the task"
            def tasks = listTracker.getTasks()
            tasks.contains( "Shave Yak" )
    }
}
```

Console | Markers | Progress | JUnit

Finished after 0.022 seconds

Runs: 1/1 | Errors: 0 | Failures: 0

com.burkhufnagel.ListTrackerSpec [Runner: JUnit 4] (0.001 s)

Failure Trace

```
    public void addTask( String taskDescription ) {
        this.tasks.add( taskDescription );
    }

    public List<String> getTasks() {
        return tasks;
    }
}
```

```
class ListTrackerSpecification extends Specification {
   def "When a task is added to the list, it should contain the task"() {
      given: "an empty task list"
         def listTracker = new ListTracker()

      when: "a task is added to the list”
         listTracker.addTask( "Shave Yak" )

      then: "the list should contain the task"
         def tasks = listTracker.getTasks()
         tasks.contains( "Shave Yak" )
   }
}
```

```
public class ListTracker {
   private List<String> tasks = new ArrayList<>();

   public void addTask( String taskDescription ) {
      this.tasks.add( taskDescription );
   }

   public List<String> getTasks() {
      return tasks;
   }
}
```

```
class ListTrackerSpecification extends Specification {
    def "When a task is added to the list, it should contain the task"() {
        given: "an empty task list"
            def listTracker = new ListTracker()

        when: "a task is added to the list”
            listTracker.addTask( "Shave Yak" )

        then: "the list should contain the task"
            def tasks = listTracker.getTasks()
            tasks.contains( "Shave Yak" )
    }
}
```

```
public class ListTracker {
    private List<String> tasks = new ArrayList<>();

    public void addTask( String taskDescription ) {
        this.tasks.add( taskDescription );
    }

    public List<String> getTasks() {
        return tasks;
    }
}
```

```
class ListTrackerSpecification extends Specification {
    def "When a task is added to the list, it should contain the task"() {
        given: "an empty task list"
            def listTracker = new ListTracker()

        when: "a task is added to the list”
            listTracker.addTask( "Shave Yak" )

        then: "the list should contain the task"
            def tasks = listTracker.getTasks()
            tasks.contains( "Shave Yaks" )
    }
}
```

```
public class ListTracker {
    private List<String> tasks = new ArrayList<>();

    public void addTask( String taskDescription ) {
        this.tasks.add( taskDescription );
    }

    public List<String> getTasks() {
        return tasks;
    }
}
```

```
class ListTrackerSpecification extends Specification {
    def "When a task is added to the list, it should contain the task"() {
        given: "an empty task list"
            def listTracker = new ListTracker()

        when: "a task is added to the list”
            listTracker.addTask( "Shave Yak" )

        then: "the list should contain the task"
            def tasks = listTracker.getTasks()
            tasks.contains( "Shave Yaks" )
    }
}
```

Console | Markers | Progress | JUnit

Finished after 0.322 seconds

Runs: 1/1 Errors: 0 Failures: 1

com.burkhufnagel.Li
When a task is ad

Failure Trace

Condition not satisfied:

```
tasks.contains("Shave yaks")
|     |
|     false
[Shave yak]
```

at com.burkhufnagel.ListTrackerSpecification.When a task is added to the list, it should be retrievable(ListTrackerSpecification.groovy:73)

```
        return tasks;
    }
}
```

```
class ListTrackerSpecification extends Specification {
    def "When a task is added to the list, it should contain the task"() {
        given: "an empty task list"
            def listTracker = new ListTracker()

        when: "a task is added to the list”
            def taskDescription = "Shave Yak"
            listTracker.addTask( "Shave Yak" )

        then: "the list should contain the task"
            def tasks = listTracker.getTasks()
            tasks.contains( "Shave Yaks" )
    }
}
```

```
public class ListTracker {
    private List<String> tasks = new ArrayList<>();

    public void addTask( String taskDescription ) {
        this.tasks.add( taskDescription );
    }

    public List<String> getTasks() {
        return tasks;
    }
}
```

```
class ListTrackerSpecification extends Specification {
    def "When a task is added to the list, it should contain the task"() {
        given: "an empty task list"
            def listTracker = new ListTracker()

        when: "a task is added to the list”
            def taskDescription = "Shave Yak"
            listTracker.addTask( taskDescription )

        then: "the list should contain the task"
            def tasks = listTracker.getTasks()
            tasks.contains( taskDescription )
    }
}
```

```
public class ListTracker {
    private List<String> tasks = new ArrayList<>();

    public void addTask( String taskDescription ) {
        this.tasks.add( taskDescription );
    }

    public List<String> getTasks() {
        return tasks;
    }
}
```

```
class ListTrackerSpecification extends Specification {
    def "When a task is added to the list, it should contain the task"() {
        given: "an empty task list"
            def listTracker = new ListTracker()

        when: "a task is added to the list”
            def taskDescription = "Shave Yak"
            listTracker.addTask( taskDescription )

        then: "the list should contain the task"
            def tasks = listTracker.getTasks()
            tasks.contains( taskDescription )
    }
```

Console | Markers | Progress | JUnit

Finished after 0.022 seconds

Runs: 1/1 | Errors: 0 | Failures: 0

com.burkhufnagel.ListTrackerSpec [Runner: JUnit 4] (0.001 s)

Failure Trace

```
    public void addTask( String taskDescription ) {
        this.tasks.add( taskDescription );
    }

    public List<String> getTasks() {
        return tasks;
    }
}
```

# Sample Spock Report

## Report for com.bth.bdd.todolist.ToDoListServiceSpec

---

Summary:

*Created on Mon Sep 16 14:28:19 PDT 2019 by bth0624*

| Executed features | Failures | Errors | Skipped | Success rate | Time |
|---|---|---|---|---|---|
| 2 | 0 | 0 | 0 | 100.0% | 0.025 seconds |

Features:

- The to do list should initially be empty
- When an item is added to an empty list, then the list should contain the item

### The to do list should initially be empty

Return

*Given:* a new instance of ToDoListService

*When:* the list is retrieved

*Then:* it should be empty

### When an item is added to an empty list, then the list should contain the item

Return

*Given:* a new instance of ToDoListService

*When:* the task 'Shave yak' is added

*Then:* the list should contain one task

*And:* the task 'Shave yak' should be in the list

---

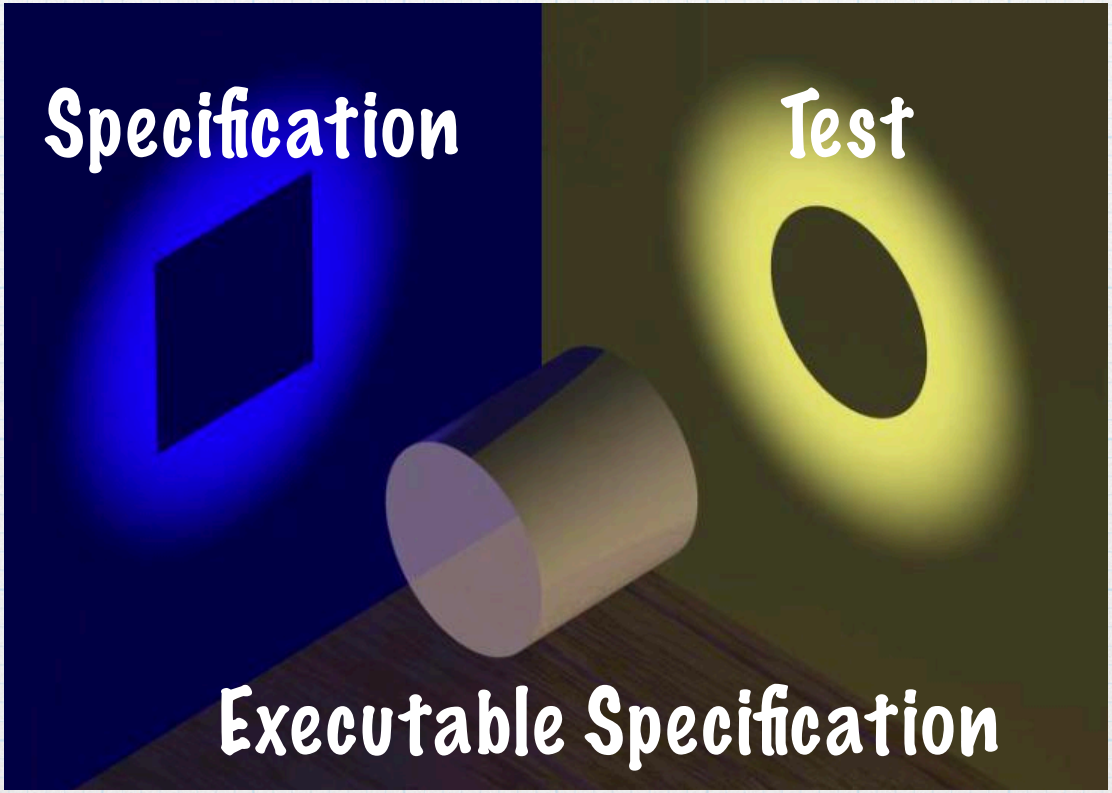
Generated by Athaydes Spock Reports

# Agenda

* ~~Introduction~~
* ~~What is TDD?~~
* ~~Why should you care?~~
* ~~Successfully adopting TDD~~
* Summary

# Summary

## To succeed with TDD, write Executable Specifications and run them as Tests!



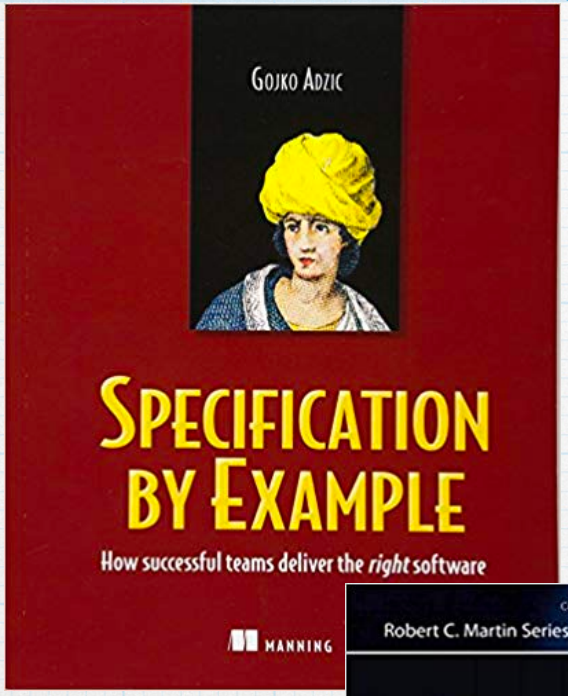
# Thank you!

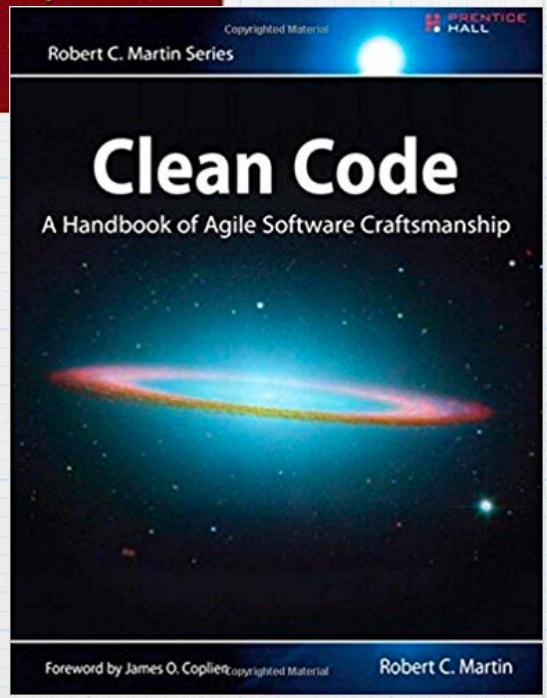
# Live Long and Prosper

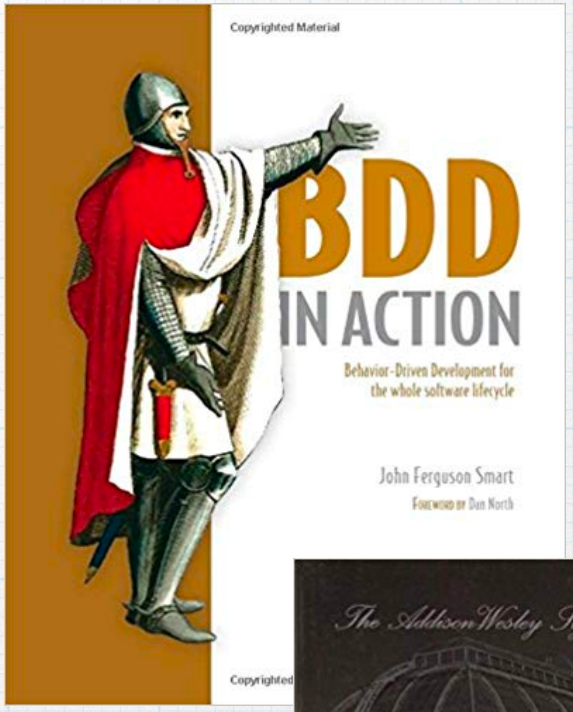
# Please tap the green button on the way out.

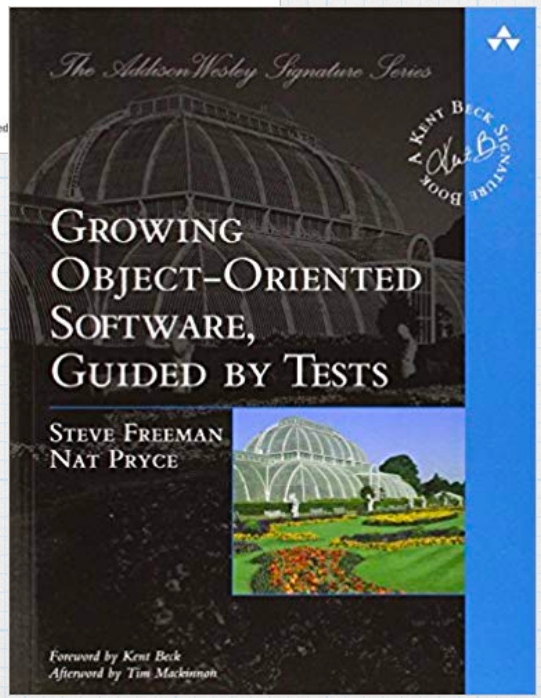
# Thank you!

# Resources


Gojko Adzic
Specification by Example
How successful teams deliver the right software
Manning


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


Copyrighted Material
BDD
IN ACTION
Behavior-Driven Development for the whole software lifecycle
John Ferguson Smart
Foreword by Dan North


The Addison-Wesley Signature Series
A Kent Beck Signature Book
Growing Object-Oriented Software, Guided by Tests
Steve Freeman
Nat Pryce
Foreword by Kent Beck
Afterword by Tim Mackinnon

# Questions & Contact Info

Feedback, questions, comments to:
burkhufnagel@gmail.com

Twitter: @burkhufnagel