Want To Succeed With TDD?

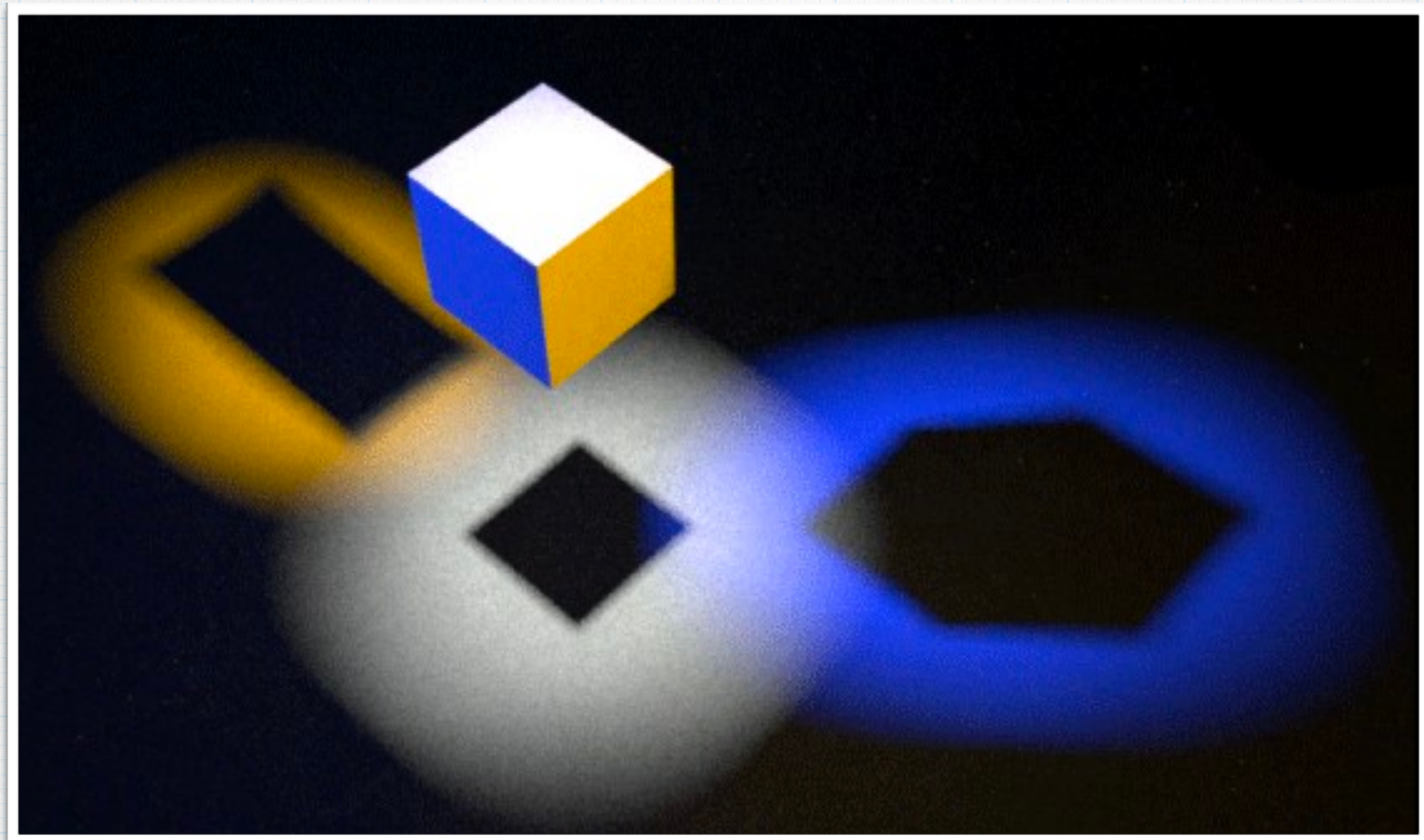
Stop Writing Tests!

Burk Hufnagel - Solution Architect
Daugherty Business Solutions

Thought Experiment

What would an object have to look like if its shadow could be a square, a hexagon, or a rectangle?

Point of View Changes Things

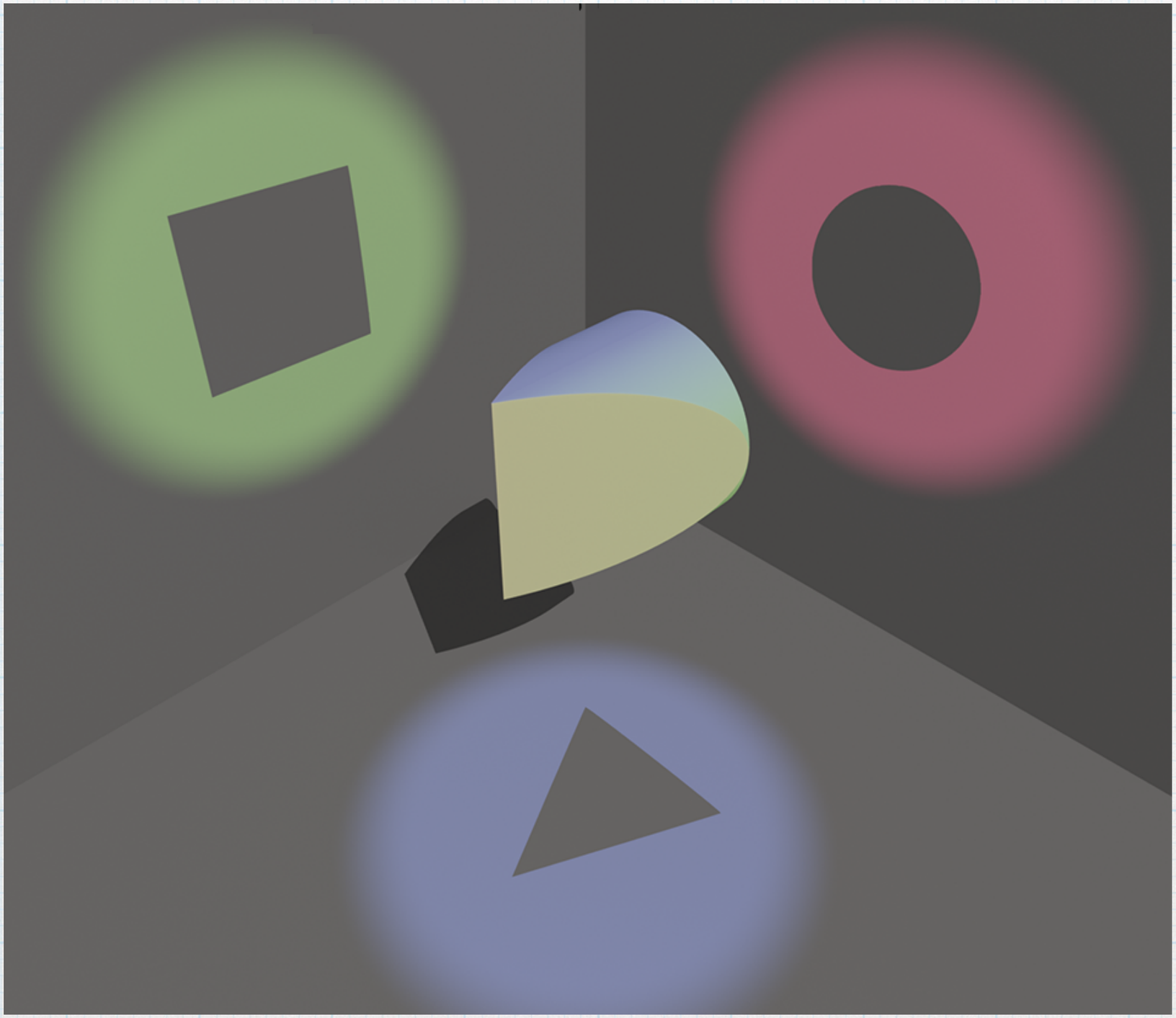


http://m759.net/wordpress/?p=49811

Imagination Time

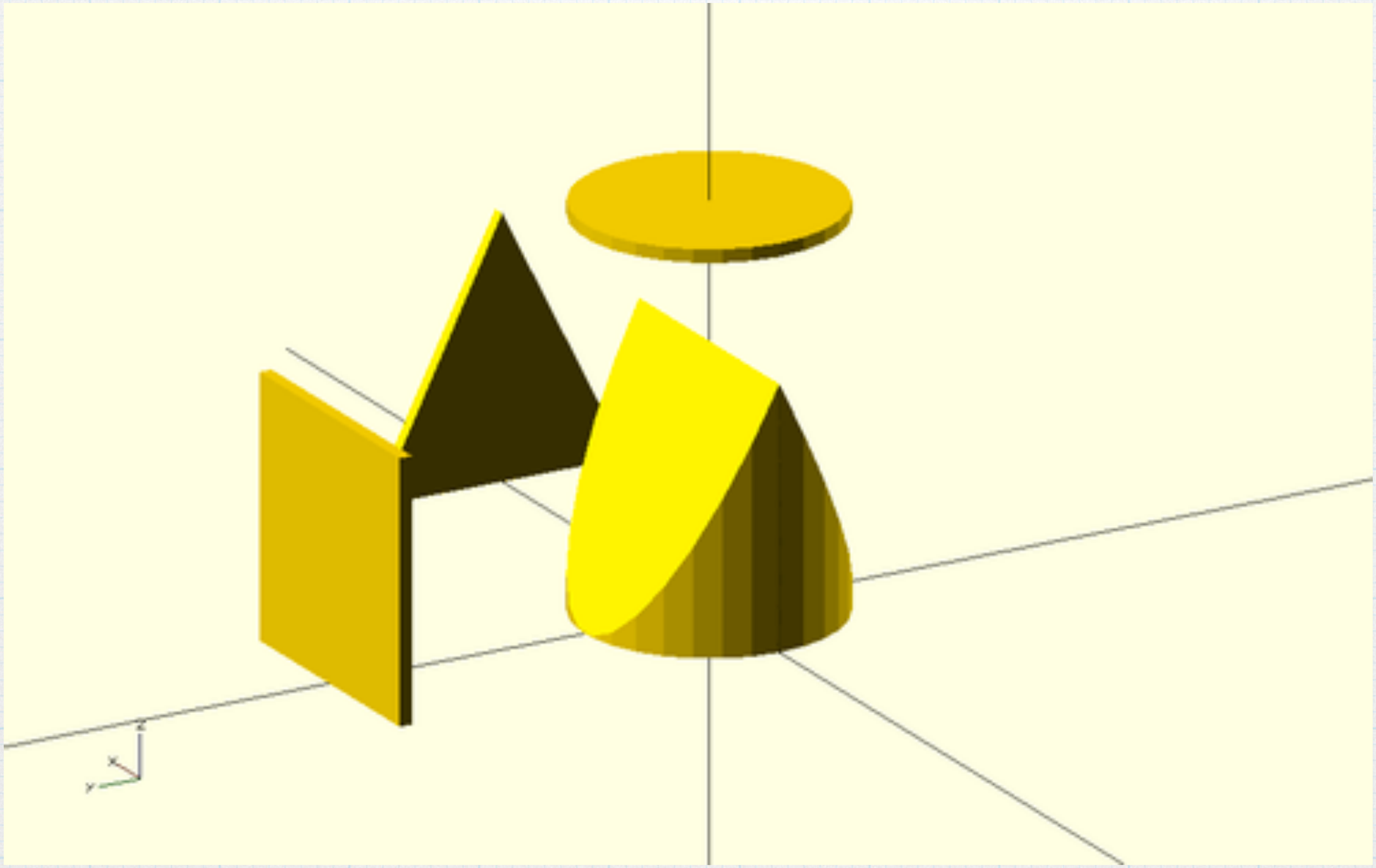
What about an object whose shadow could look like a square, a circle, or a triangle.

Point of View Changes Things



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

View from a Different Angle



http://kitwallace.tumblr.com/post/103975175234/george-harts-circle-triangle-square-puzzle

Reasonable people
may see the same thing
in different ways,
based on
their perspective.

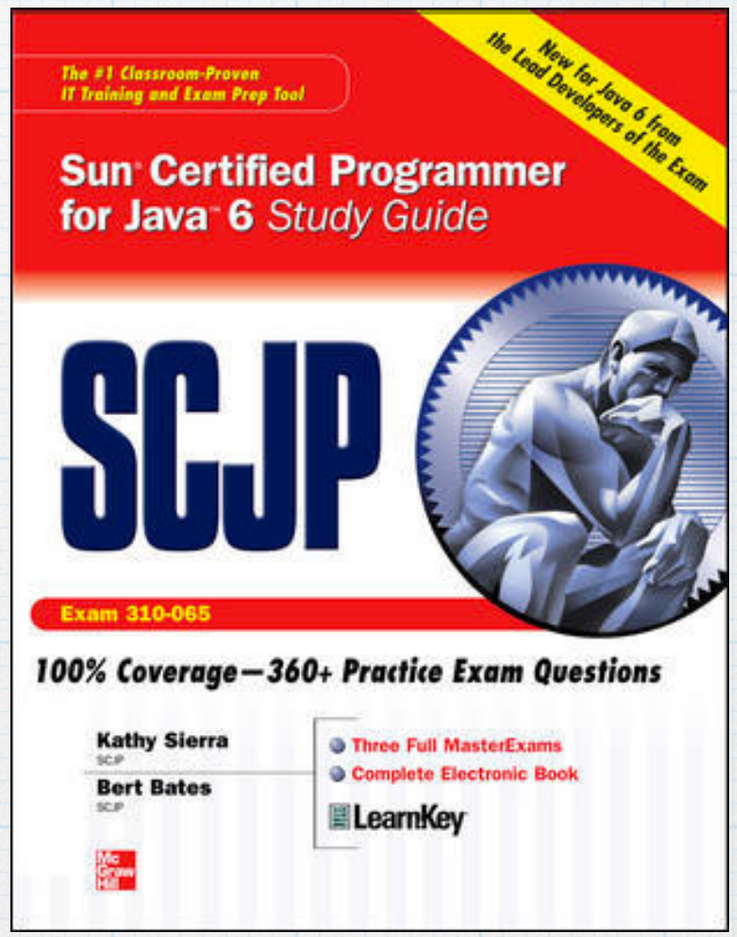
Want To Succeed With TDD?

Stop Writing Tests!

Burk Hufnagel - Solution Architect
Daugherty Business Solutions

Who is Burk Hufnagel?

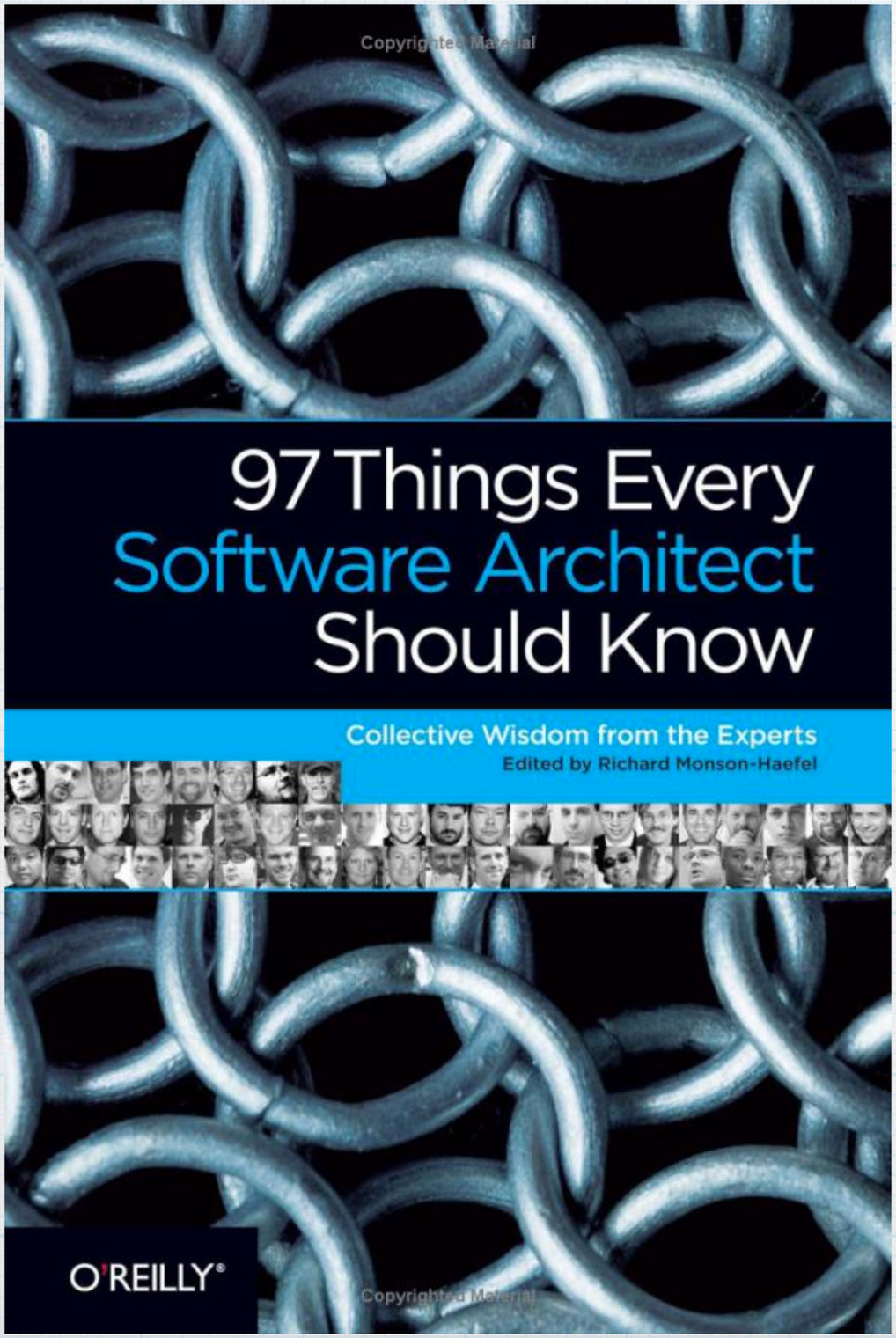
"Burk fixed more code than we care to admit."
— Kathy Sierra & Bert Bates,
Authors: SCJP6 Study Guide

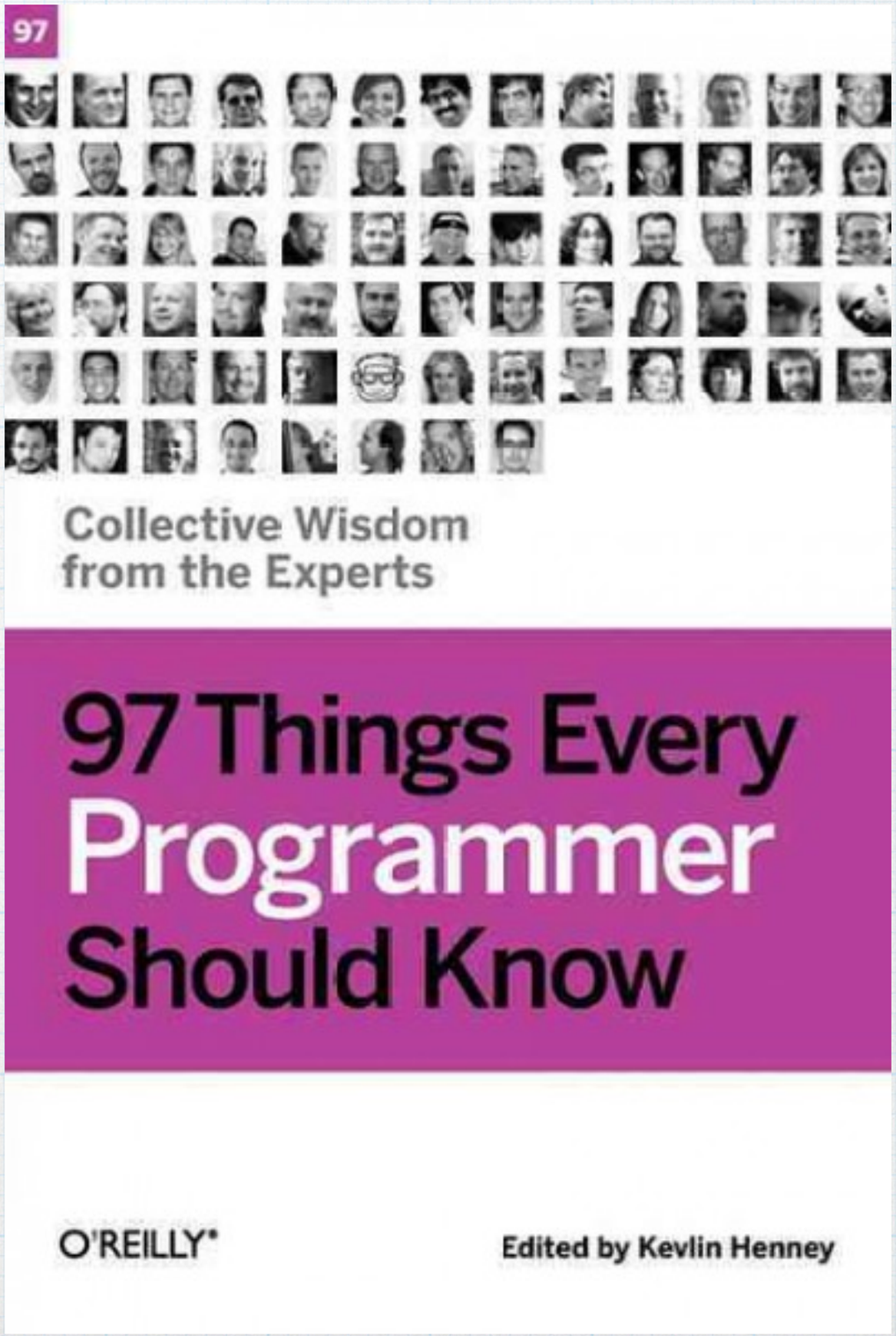


Sun Certified Java Programmer,
Developer, Enterprise Architect

Programmer and Architect with
Daugherty Business Solutions

Who is Burk Hufnagel?





Why are you here?

Delivering Better Code, Faster

What does that mean?

Delivering
Writing code does not create business value

Better Code
Solving the business problem is not enough

Faster
Thing change faster and faster

Solving the Business Problem

* Build the right thing
* Build the thing right

Build the
right thing

Build the
thing right

What does that mean?

Delivering
Writing code does not create business value

Better Code
Solving the business problem is not enough

Faster
Thing change faster and faster

Agenda

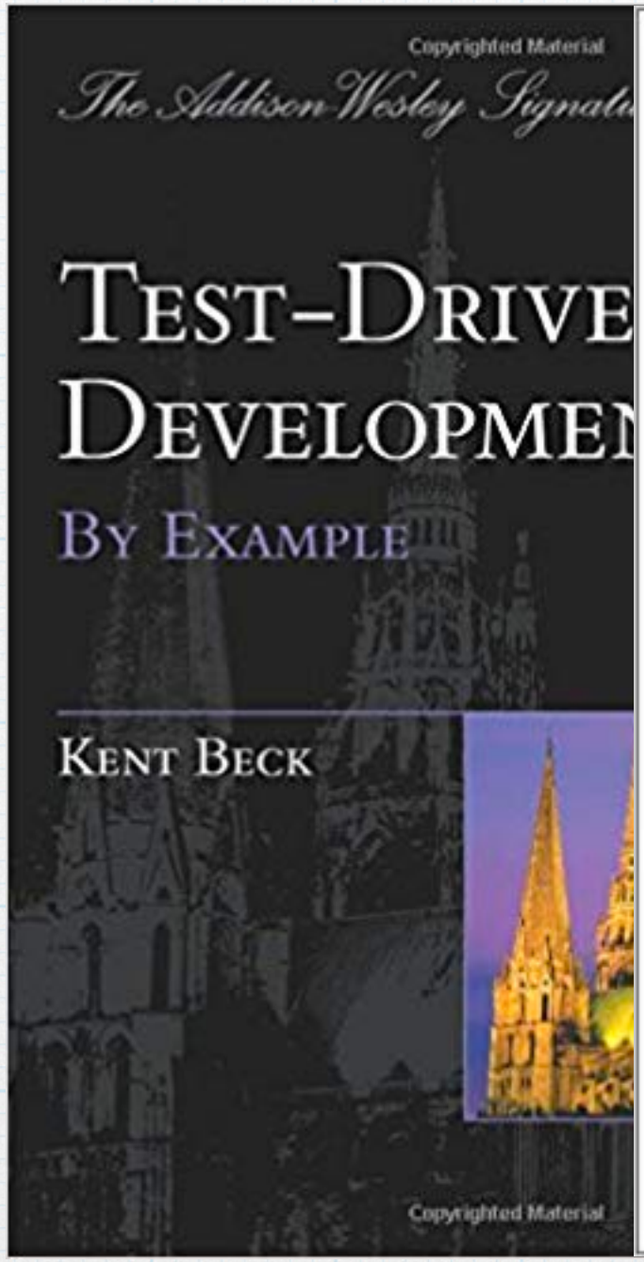
* Introduction
* What is TDD?
* Why should you care?
* Successfully adopting TDD
* Summary

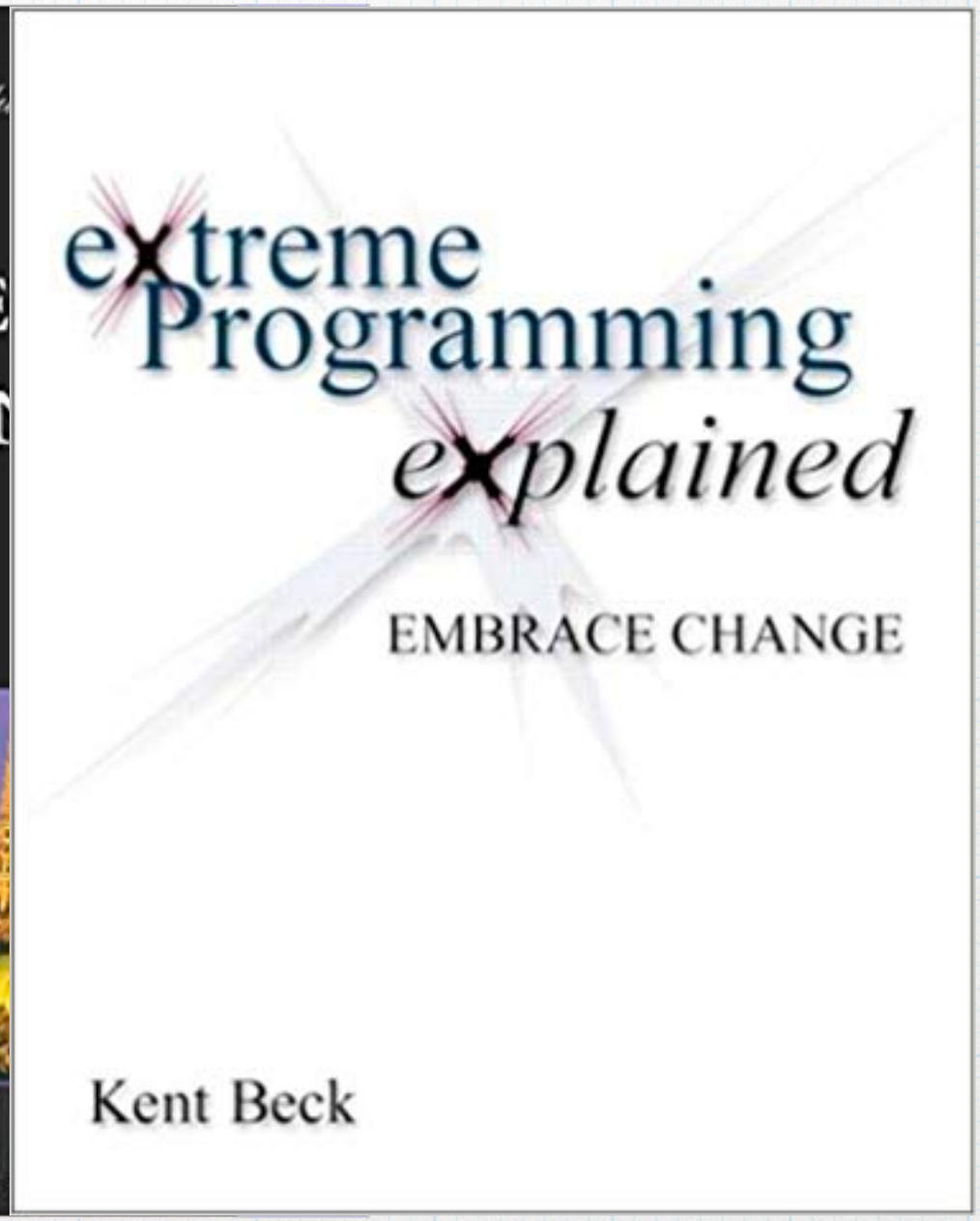
Questions are a Good Thing.

Agenda

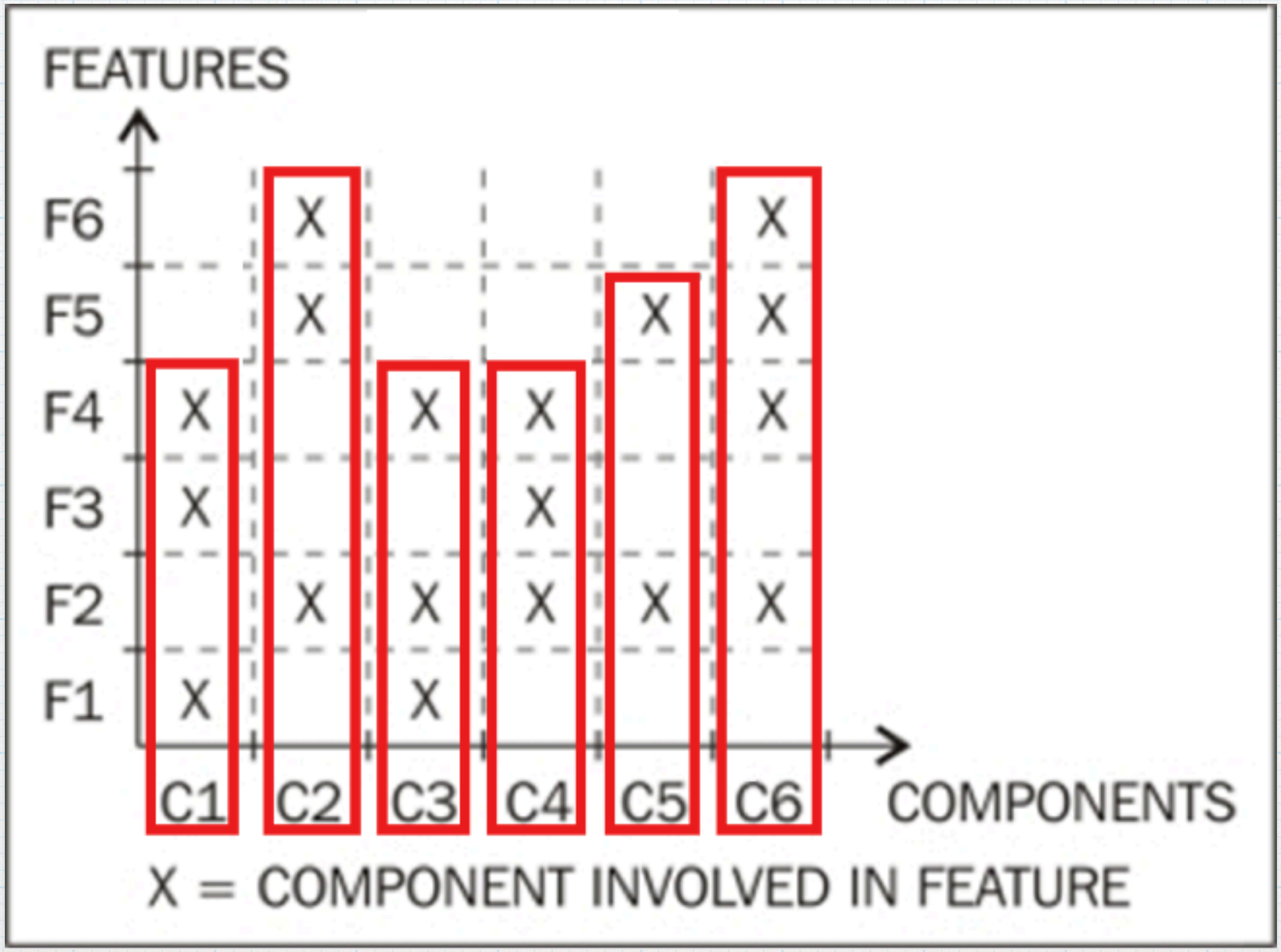
* ~~Introduction~~
* What is TDD?
* Why should you care?
* Successfully adopting TDD
* Summary

What Is TDD?


Copyrighted Material
The Addison Wesley Signature
TEST-DRIVEN
DEVELOPMENT
BY EXAMPLE
KENT BECK
Copyrighted Material


extreme
Programming
explained
EMBRACE CHANGE
Kent Beck

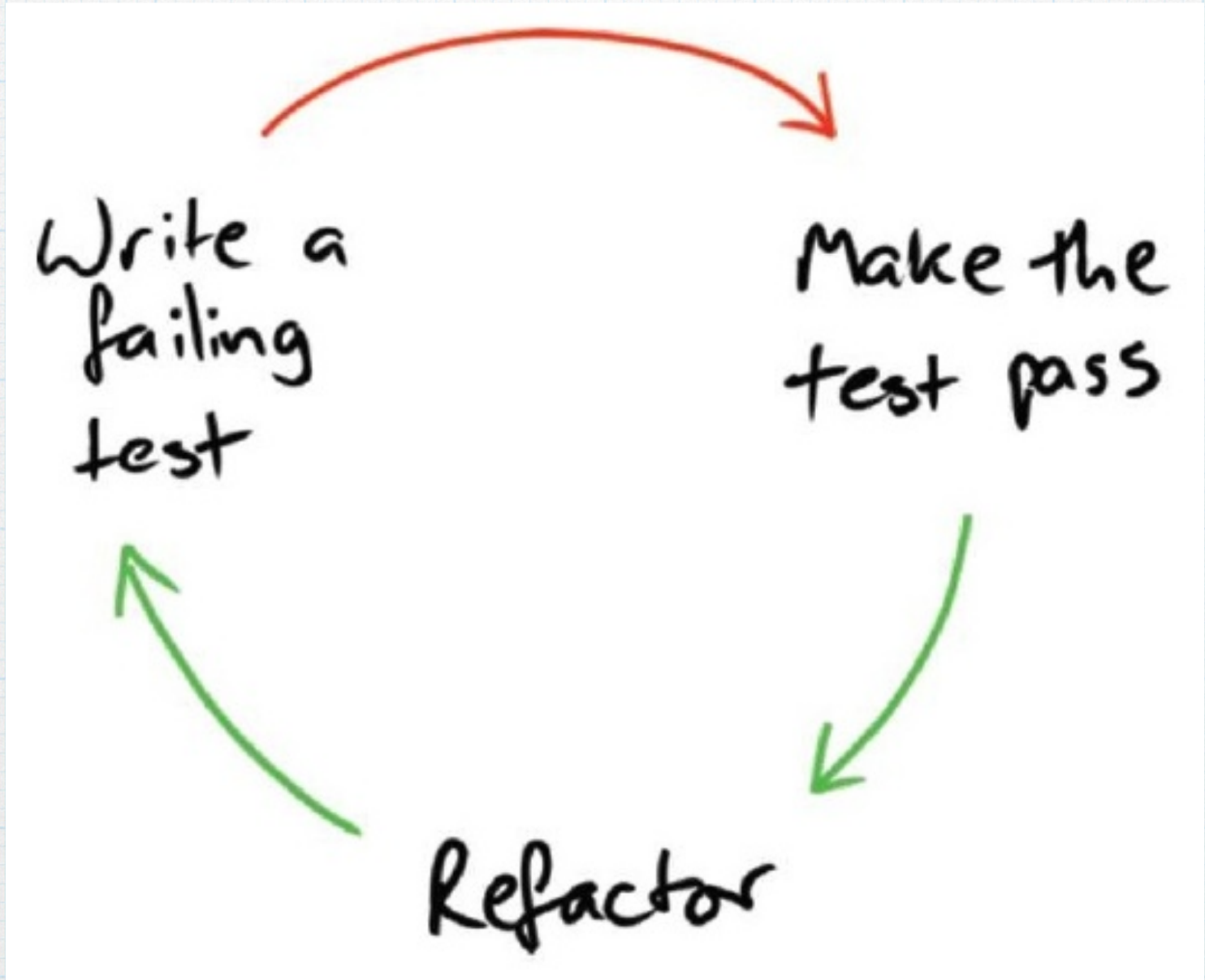
What Is TDD?



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

What Is TDD?

Red, Green, Refactor



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.

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 in the debugger
* Deliver to production faster
* Better code quality
* Highly cohesive & loosely coupled

Benefits of TDD

* TDD is not a magic wand
* TDD is a highly efficient way to do the work you have to do to get into production.

https://www.planetsuperheroes.com/product/harry-potter-prop-elder-wand

TANSTAAFL

There Ain't No Such Thing As A Free Lunch

* Clean code doesn't automatically happen
* You still have to think about how to design your API and code
* You'll still write bugs, but finding and fixing them should be quicker and easier
* Changing your habits is **hard**, but having a good **reason** to change makes it easier.

Agenda

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

Successfully Adopting TDD

* Common problems with TDD
* Solutions to the problems

Common Problems with TDD

* Unreasonable expectations
* Tests

Unreasonable Expectations

* Sounds too easy and feels too hard

Unreasonable Expectations

* There's only three steps...
  * Depends on changing your habits
  * You have to do the work

Tests

* Writing tests first is illogical
  * How do you know what to test?
  * How do you know when you're done?
* "Test" is the wrong word.

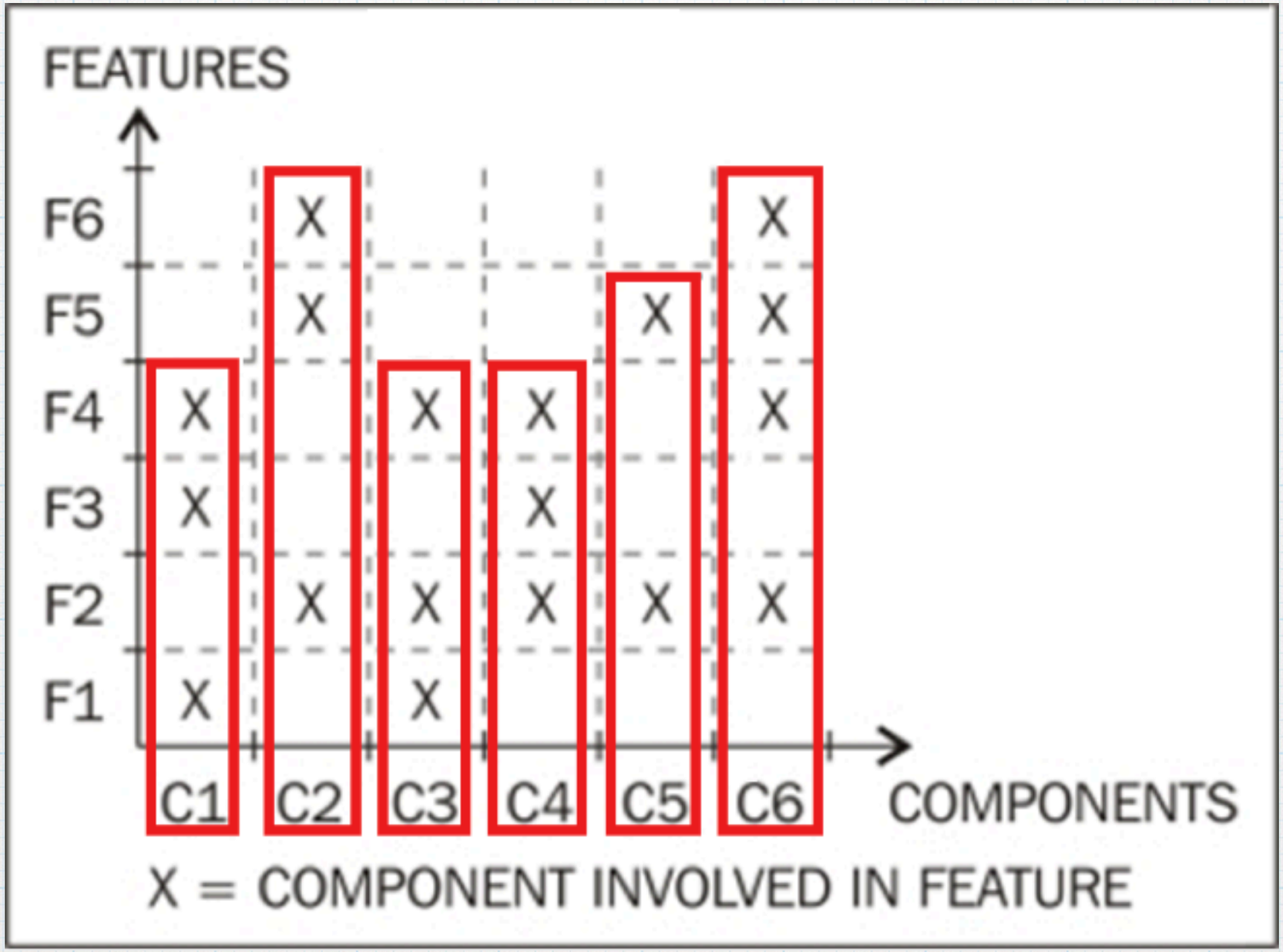
Specifications

* Describe what should happen.
* You need to know this in order to satisfy the business requirements.

Executable Specifications

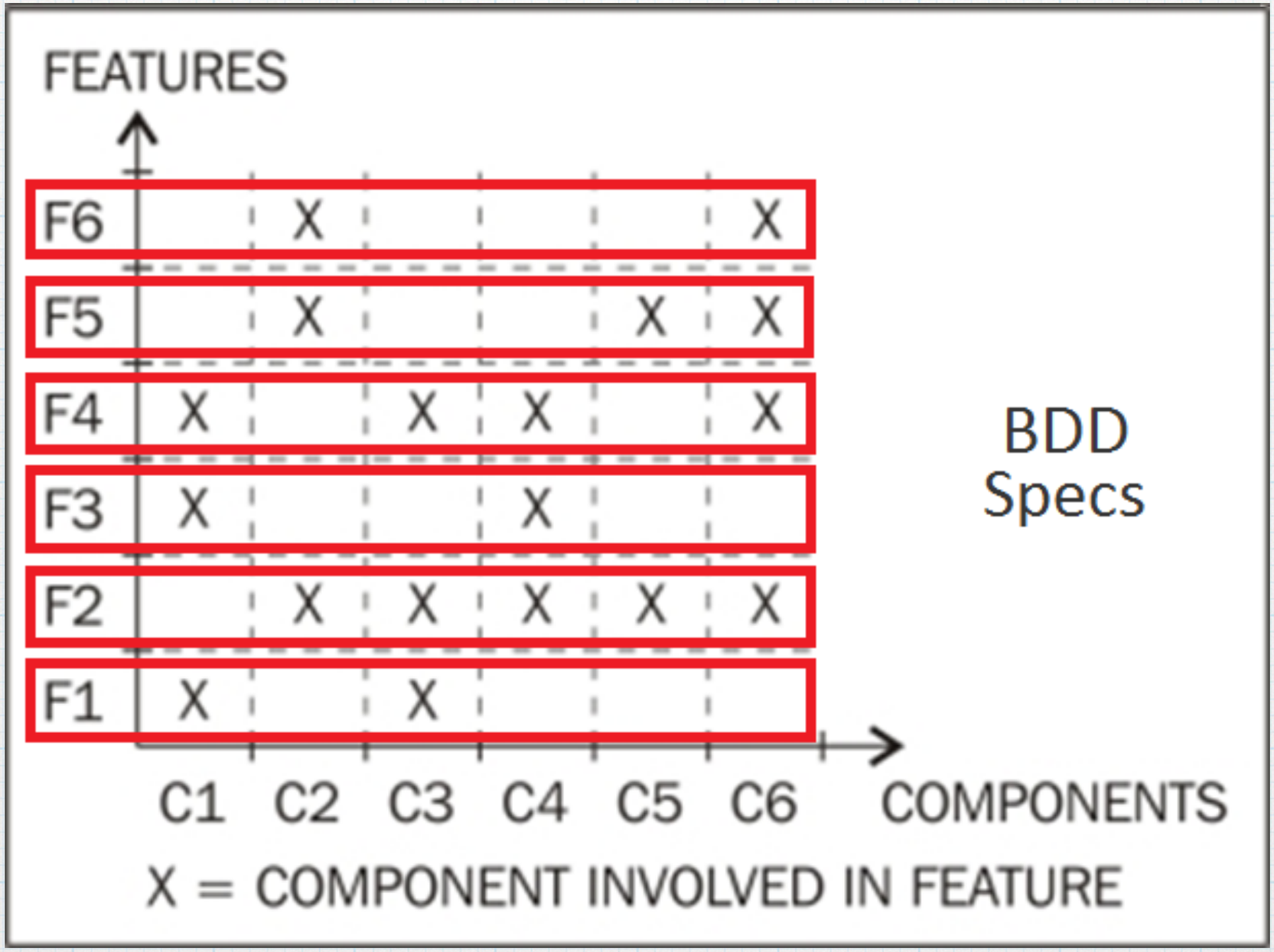
* Blend of user story and code.
* Include the Given, When, Then and the how.

TDD Focus on Units



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

Focus on Features



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

TDD?

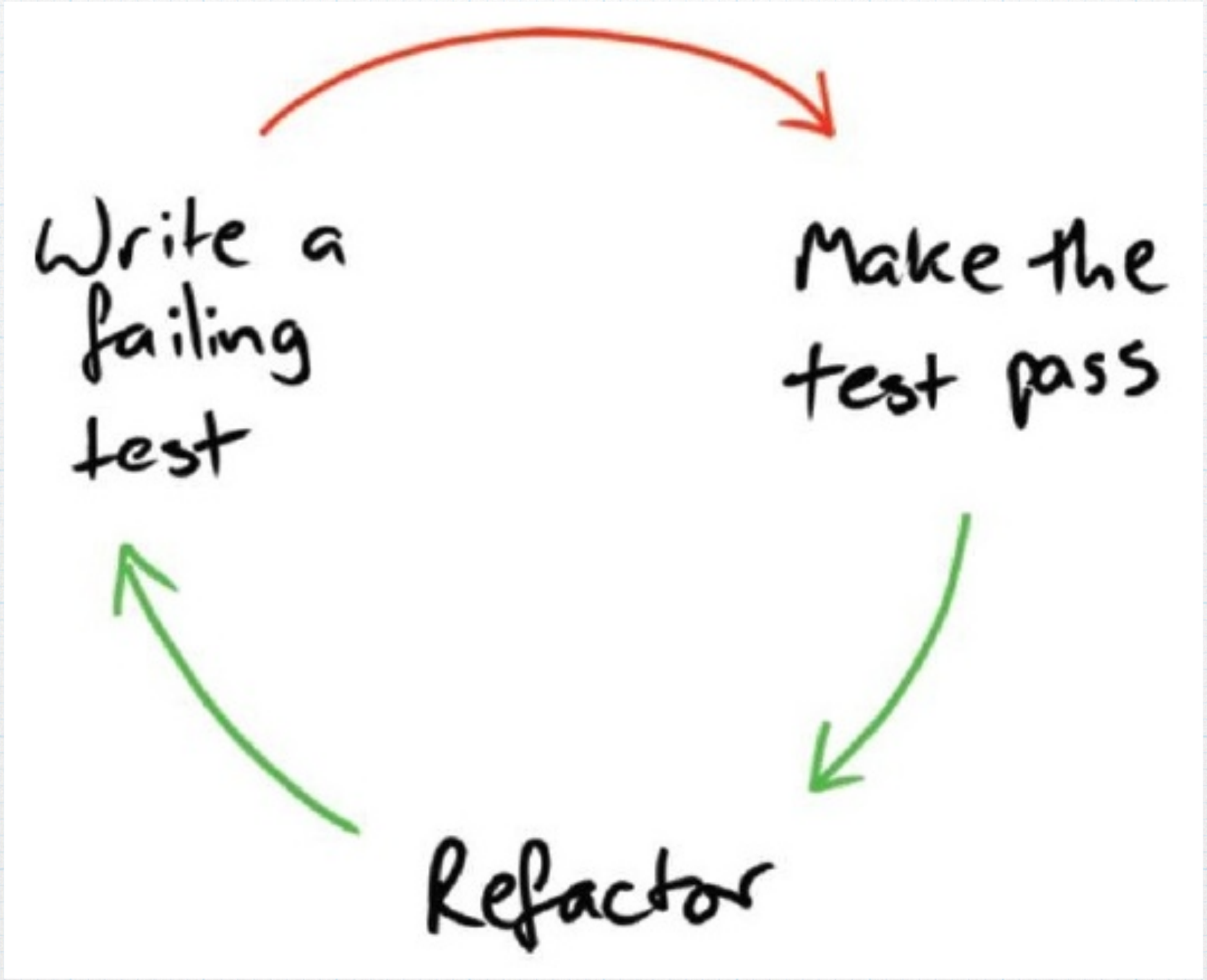
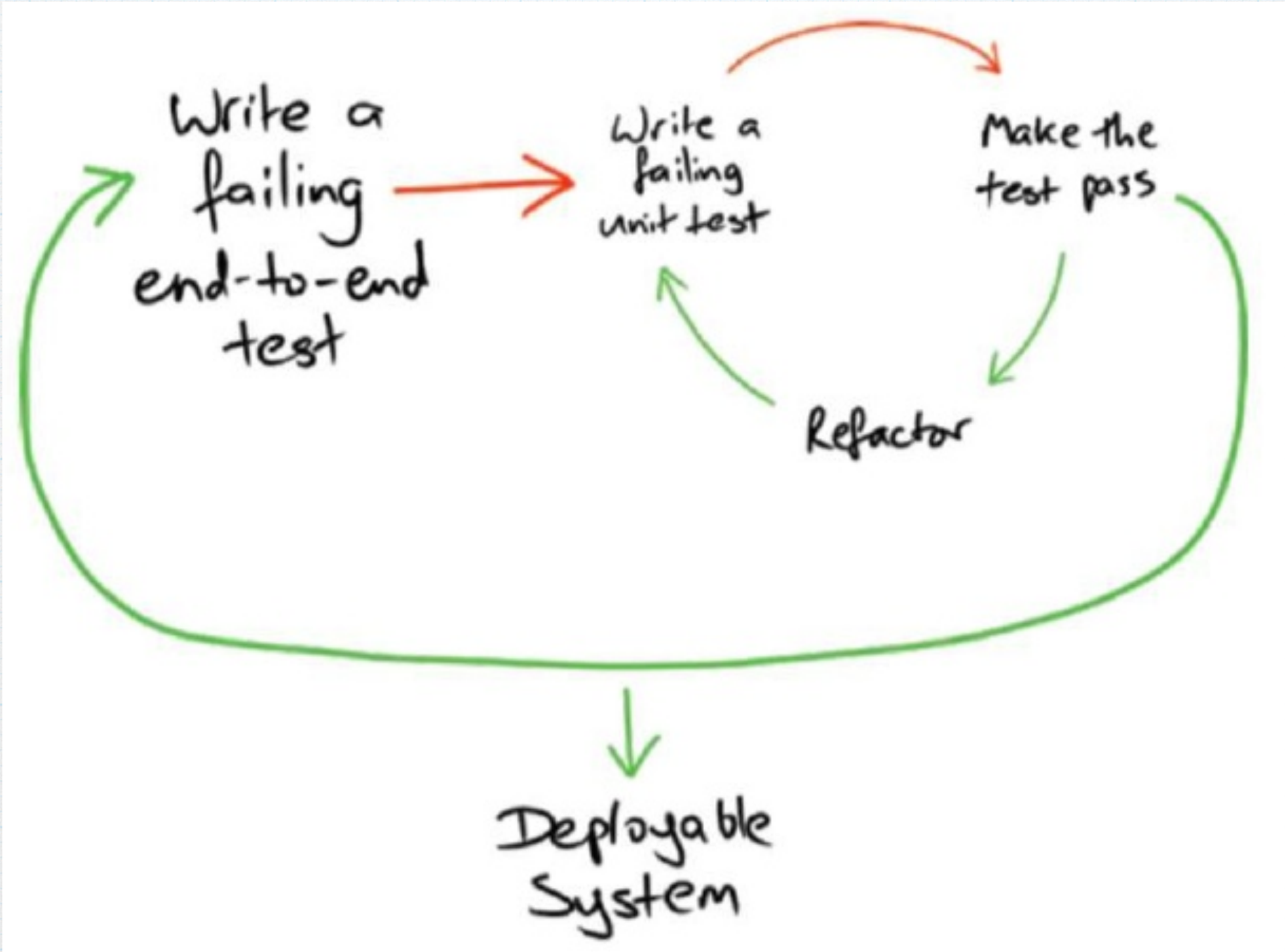


image from: a4direct.com

Behavior Driven Development



images from: a4direct.com

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 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"

A scenario describes high level 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.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 be retrievable"() {
        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 be retrievable"() {
        given: "an empty task list"
            def listTracker = new ListTracker()
        when: "a task is added to the list"

        then: "the list should contain the task"

    }
}
```

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

        then: "the list should contain the task"

    }
}
```

```groovy
public class ListTracker {

}
```

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

        then: "the list should contain the task"

    }
}
```

```groovy
public class ListTracker {

}
```

```groovy
class ListTrackerSpecification extends Specification {
    def "When a task is added to the list, it should be retrievable"() {
        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"

    }
}
```

```groovy
public class ListTracker {

}
```

```
class ListTrackerSpecification extends Specification {
    def "When a task is added to the list, it should be retrievable"() {
        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 ) {

    }
}
```

```groovy
class ListTrackerSpecification extends Specification {
    def "When a task is added to the list, it should be retrievable"() {
        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"

    }
}
```

```groovy
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 be retrievable"() {
        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 ) {

    }
}
```

```groovy
class ListTrackerSpecification extends Specification {
    def "When a task is added to the list, it should be retrievable"() {
        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()
    }
}
```

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

    }

    public List<String> getTasks() {

    }
}
```

```groovy
class ListTrackerSpecification extends Specification {
    def "When a task is added to the list, it should be retrievable"() {
        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()
    }
}
```

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

    }

    public List<String> getTasks() {

    }
}
```

```groovy
class ListTrackerSpecification extends Specification {
    def "When a task is added to the list, it should be retrievable"() {
        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()
    }
}
```

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

    }

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

```groovy
class ListTrackerSpecification extends Specification {
    def "When a task is added to the list, it should be retrievable"() {
        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()
    }
}
```

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

    }

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

```groovy
class ListTrackerSpecification extends Specification {
    def "When a task is added to the list, it should be retrievable"() {
        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” )
    }
}
```

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

    }

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

```groovy
class ListTrackerSpecification extends Specification {
    def "When a task is added to the list, it should be retrievable"() {
        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.taskAddedToListShouldBeRetrievab

```groovy
    }

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

```groovy
class ListTrackerSpecification extends Specification {
    def "When a task is added to the list, it should be retrievable"() {
        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” )
    }
}
```

```groovy
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 be retrievable"() {
        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 );
    }
}
```

```groovy
class ListTrackerSpecification extends Specification {
    def "When a task is added to the list, it should be retrievable"() {
        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” )
    }
}
```

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

    }

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

```groovy
class ListTrackerSpecification extends Specification {
    def "When a task is added to the list, it should be retrievable"() {
        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

```groovy
    }

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

```groovy
class ListTrackerSpecification extends Specification {
    def "When a task is added to the list, it should be retrievable"() {
        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” )
    }
}
```

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

    }

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

```groovy
class ListTrackerSpecification extends Specification {
    def "When a task is added to the list, it should be retrievable"() {
        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” )
    }
}
```

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

    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 be retrievable"() {
        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<String>();

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

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

```groovy
class ListTrackerSpecification extends Specification {
    def "When a task is added to the list, it should be retrievable"() {
        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

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

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

```groovy
class ListTrackerSpecification extends Specification {
    def "When a task is added to the list, it should be retrievable"() {
        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” )
    }
}
```

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

    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 be retrievable"() {
        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<String>();

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

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

```groovy
class ListTrackerSpecification extends Specification {
    def "When a task is added to the list, it should be retrievable"() {
        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" )
    }
}
```

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

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

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

```groovy
class ListTrackerSpecification extends Specification {
    def "When a task is added to the list, it should be retrievable"() {
        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 0.334 seconds

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

com.burkhufnagel.
When a task is

Failure Trace

Condition not satisfied:

```
tasks.contains("Shave Yaks")
|     |
|     false
[Shave Yak]
```

at com.burkhufnagel.ListTrackerSpecification.When a task is to the list, it should be retrievable(ListTrackerSpecification.groovy:40)

```
        return tasks;
    }
}
```

```
class ListTrackerSpecification extends Specification {
    def "When a task is added to the list, it should be retrievable"() {
        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<String>();

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

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

```groovy
class ListTrackerSpecification extends Specification {
    def "When a task is added to the list, it should be retrievable"() {
        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 )
    }
}
```

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

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

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

```groovy
class ListTrackerSpecification extends Specification {
    def "When a task is added to the list, it should be retrievable"() {
        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

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

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

Sample Spock Report

Report for com.burkhufnagel.ListTrackerSpecification

Summary:

Created on Mon Sep 25 00:36:16 EDT 2017 by bth0624

Executed features	Failures	Errors	Skipped	Success rate	Time
1	0	0	0	100.0%	0.019 seconds

Features:

- When a task is added to the list, it should be retrievable

When a task is added to the list, it should be retrievable

Return

Given: an empty task list

When: a task is added to the list

Then: the list should contain the task

Agenda

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

Summary

Write Specifications and Run Tests!

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.

Get more info at ajug.org

Final Thoughts

Live Long and Prosper