

# Agile Firestarter



## Introduction to TDD



Alex Hung  
CTO of Ballywoo

# The Rationale for Unit Tests

## Debatable Software Engineering 'Facts'

(Agile/XP/SCRUM/etc.) is better than (Agile/XP/SCRUM/etc.)

TDD (test-first) is better than Test-After

## Non-Debatable Software Engineering Facts:

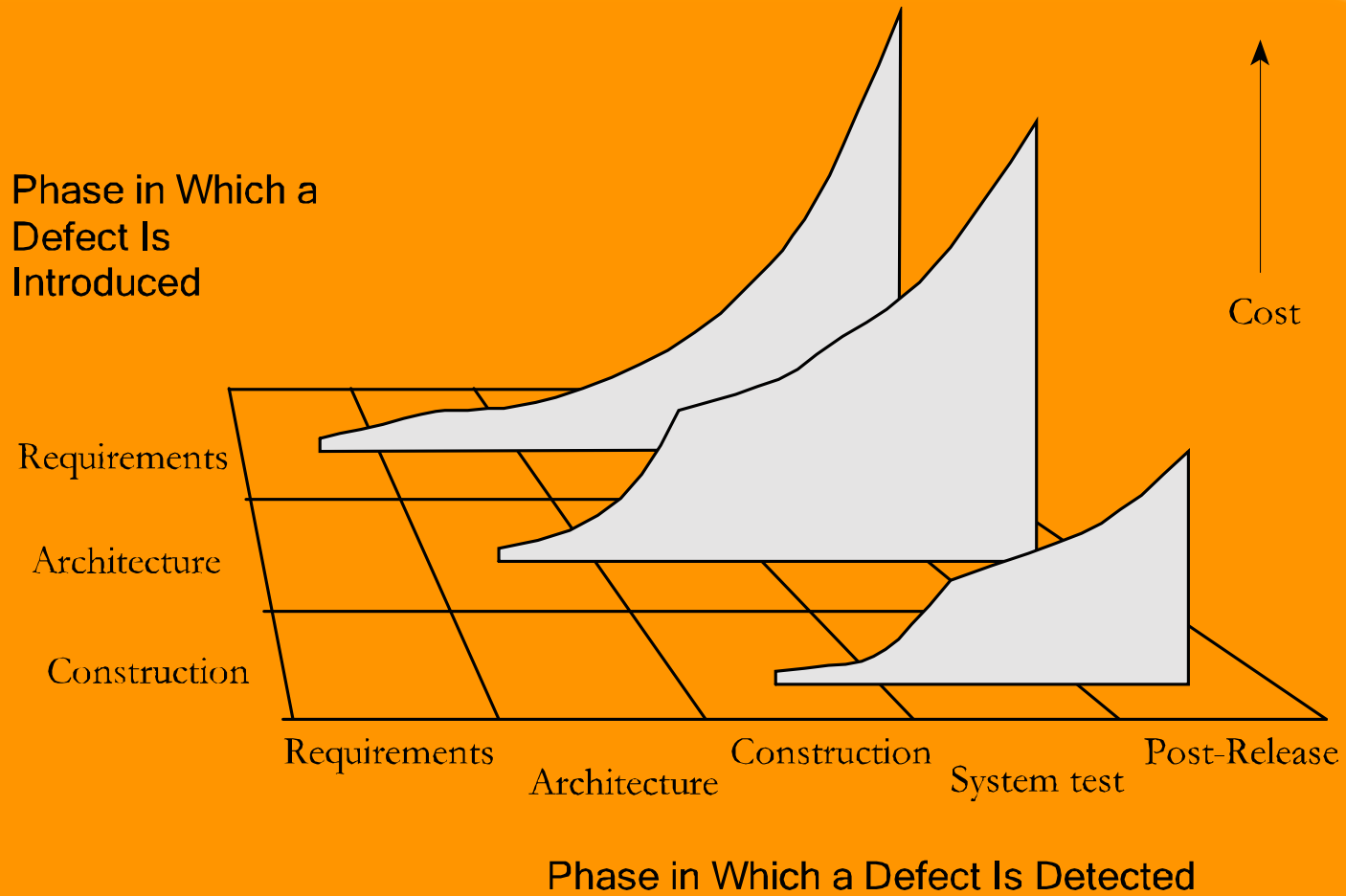
There will always be bugs

Complex programs have more bugs than simple programs

Code is more maintainable when its divided into bite-sized chunks

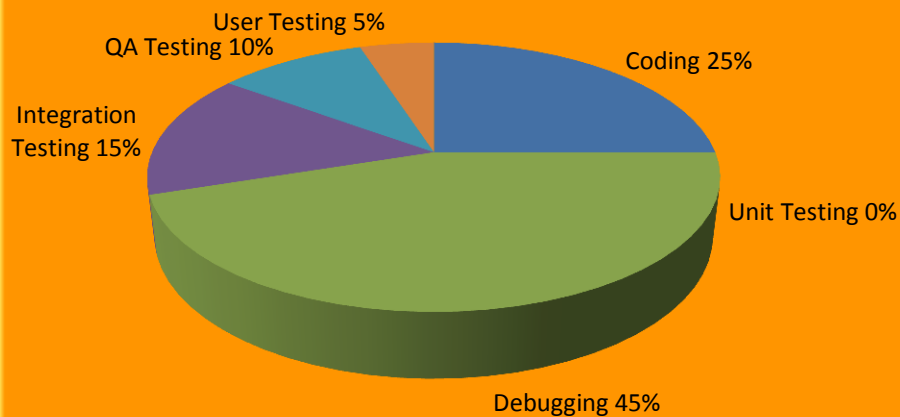
**The cost of fixing a bug escalates non-linearly over time as the project progresses**

# 'Code Complete' Defect Cost Graph

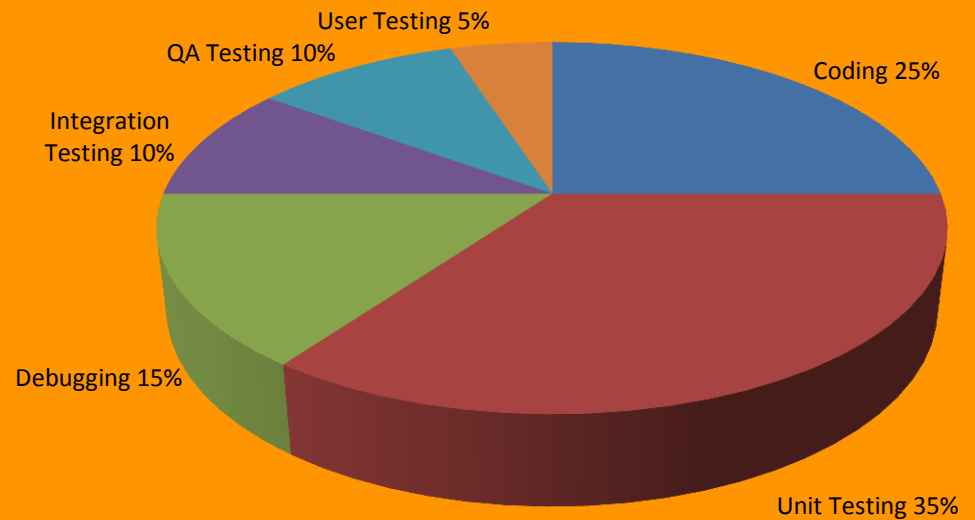


# Allocation of Developer Effort

## Effort Allocation without Unit Tests



## Effort Allocation with Unit Tests



# Types of Tests

## Human-Based Testing

Load the app, click the buttons

Time-consuming

Error-prone

Difficult to reliably reproduce results

Different inputs → Different outputs

# Types of Tests

## Automated Testing

Computer does what its good at (mass-repetition)

High-speed

Hundreds (1000s?) of automated tests in the same time to execute a single human-based test

Reproducible

Same inputs → Same outputs

GIGO (Garbage In, Garbage Out)

# Types of Automated Tests

## User Interface Tests

Primary purpose is to test user interaction with the application as a whole

Typically run via a user-interface-runner

(NUnitForms, NUnitASP, Selenium, WatiN, WatiR, White, etc.)

# Types of Automated Tests

## Integration Tests

Primary purpose is to test interaction between components

Often mistaken for unit tests

Just as valuable as unit tests, every bit as automatable as unit tests

Typically run via a unit test framework but with more complex pre-test setup and post-test teardown steps

No point in running until unit tests pass!

# Types of Automated Tests

## Unit Tests

Primary purpose is to test public interface of one or more classes

As Isolated as possible (hence 'unit')

Independent of other system components

If your test touches something else you didn't develop but your app depends on to run (database, web service, etc.) then it's not a unit test

Sandboxed

No side-effects to the environment allowed

Typically run via a unit test framework (NUnit, MbUnit, etc.)

# How is TDD different from Other Types of Testing

How is TDD different from

**Other Types of Testing**

Exercise at Code level vs. Application or UI

Feedback loop is very short

Lead straight to code

How is TDD different from

Other Types of Testing

Easy to setup – code vs. scripting/clicking

Repeatable and Consistent

# What is Test-Driven Development?

Testing *while* coding

not *before* or *after*

# What is Test-Driven Development?

Think “Test-Driven Design”

or

“Specification-Driven-Development”

Consider your design from the perspective of the consumers of your methods, classes, etc.

Outside-in design instead of Inside-Out design

# What is Test-Driven Development?

“How-Do-I-Use-This-Driven-Development”

For every line of code you write, ask yourself a simple (but powerful) question:

*“HOW WILL I TEST THAT?”*

# What about...

## Costs

Additional time spend on code that isn't part of delivery.

# What about...

## Costs

We are programmers, not testers! We should concentrate on developing features, not testing!

# What about...

## Benefits

Every class and method immediately has at least **TWO** consumers:

Your **APP** and your **TEST!**

# What about...

## Benefits

Better-isolated code leads to easier to extend, enhance, replace, maintain (lifecycle costs)

# What about...

## Benefits

Waiting until code is written to write the tests often means hard (impossible?) to test code!

# What about...

## Benefits

Less likely to write code that you eventually don't need

Write nothing that you *think* you will need

(YAGNI – You Ain't Gonna Need It!)

Now its your turn!

# CALCULATING PRIMES

# Problem Statement

Calculate Prime Numbers between 1-10,000

A positive integer divisible only by 1 and itself

By definition, 0 and 1 are *non-prime*

Write results to the console as follows:

Multiple lines of 5 comma-separated values each

Every 10 lines, insert a line indicating the count of the primes output thus far

n,n,n,n,n

n,n,n,n,n

n,n,n,n,n

n,n,n,n,n

n,n,n,n,n

n,n,n,n,n

n,n,n,n,n

n,n,n,n,n

n,n,n,n,n

n,n,n,n,n

Count: 50

n,n,n,n,n

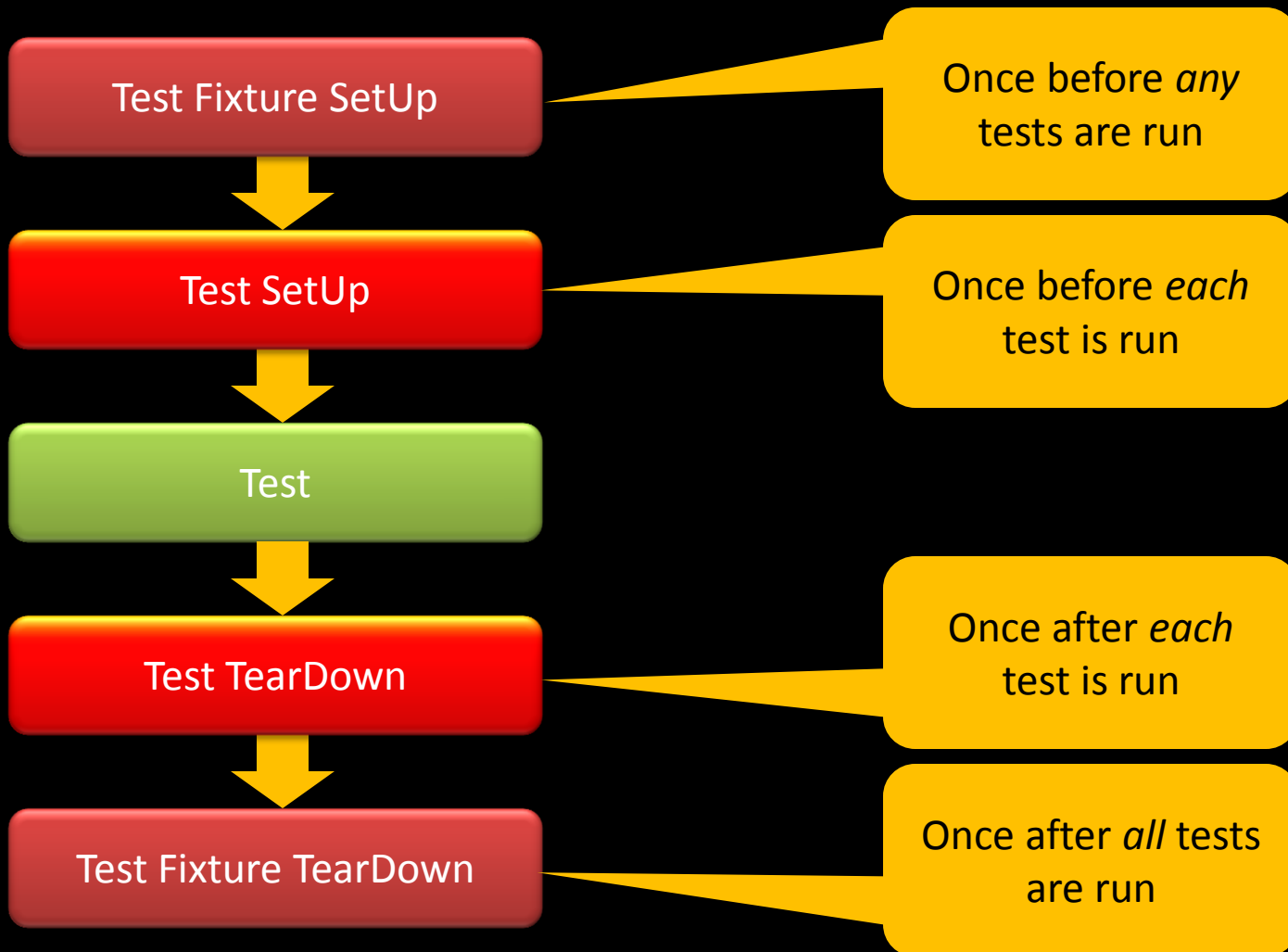
n,n,n,n,n

....

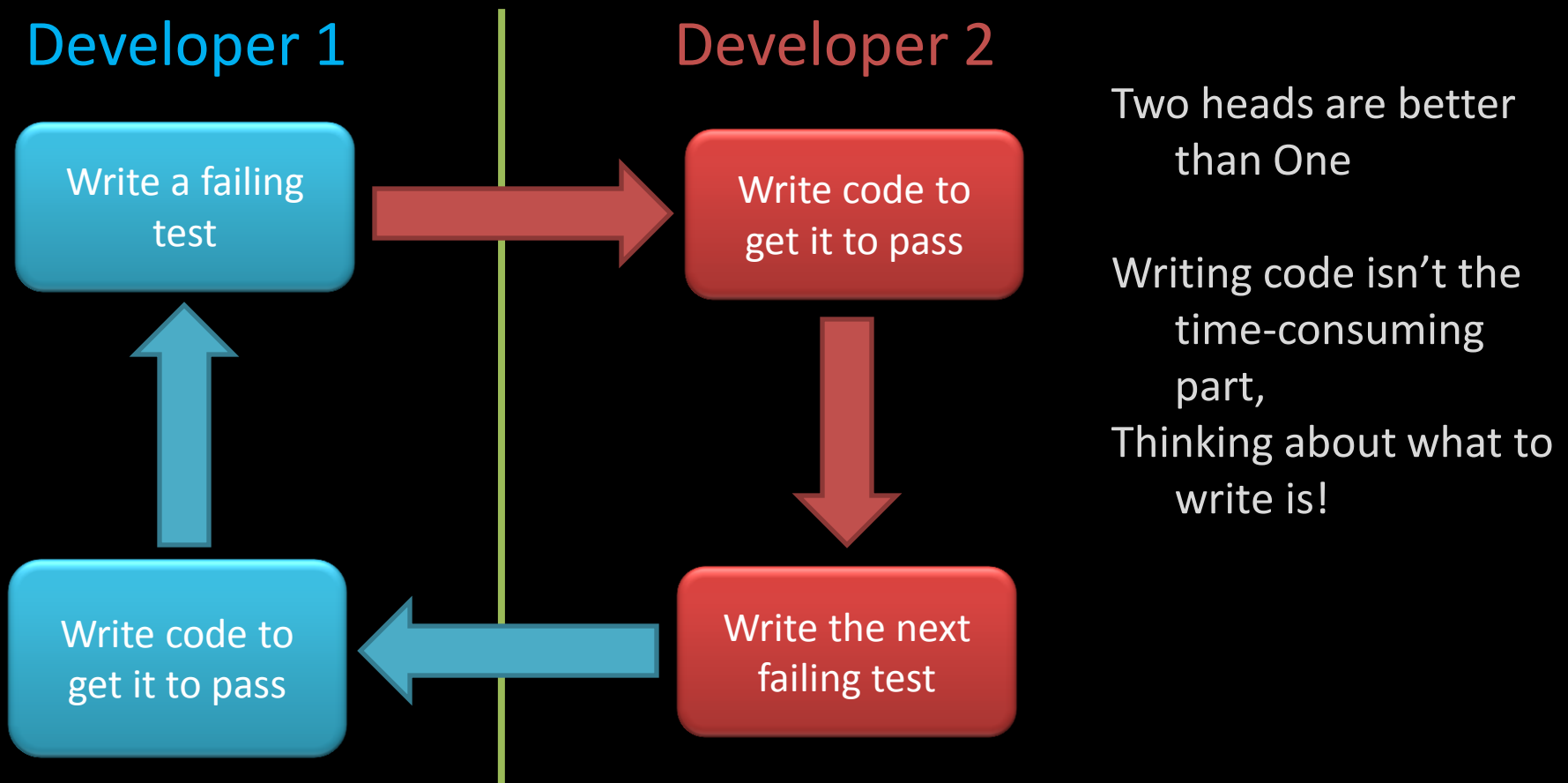
Enough talking, let's write some code!

# OUR FIRST TEST

# Unit Test Flow



# Ping-Pong Pair Programming



# The Process

REFACTOR

...mercilessly!

Write a Failing Test

Get it to Pass

Refactor!

Finally some code!

# Now GO!

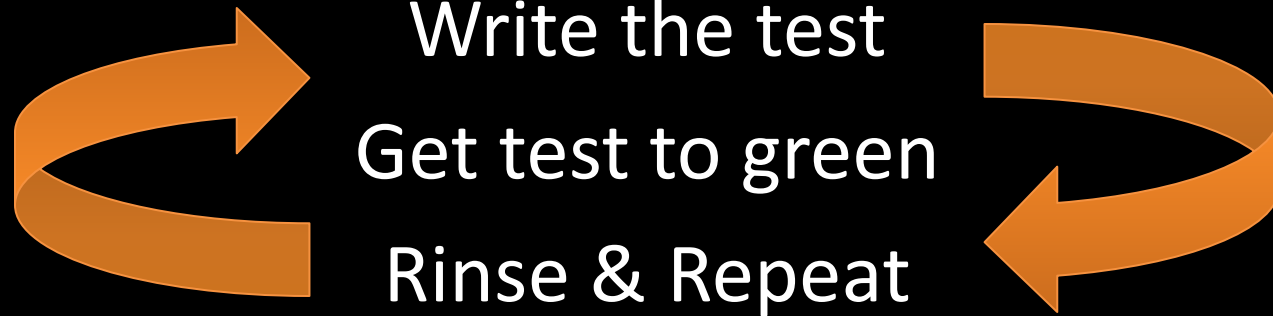
Find someone to pair with

Decide on features to implement

Write the test

Get test to green

Rinse & Repeat



Remember: **Failure** *IS* an option!

# Ways to think about Unit Testing

## Unit Testing is...

An alternative to spending your life in the debugger

A way to validate that your code is aligned with your intent

‘it compiles’ is a validation of the syntax of your code

‘it passes the unit tests’ is a validation of the behavior of your code

An hedge against future bugs

Prevents regression bugs (oops, I broke it!)

A safety-net that allows you to experiment with your design

‘What-if’ can be safely explored

No code is ‘hands-off for fear of something breaking’

# Ways to think about Unit Testing

## Unit Testing is not...

Going to save you any time in the project

...until someone changes the project requirements





Agile  
Firestarter

Questions?

Twitter: AlexHung  
alex.y.hung@gmail.com