endava | be more

AN INTRODUCTION TO TEST-DRIVEN DEVELOPMENT WITH JAVA

IYD – ROSARIO JAVA COMMUNITY

MARTIN ARNESI – 06/2020

OUTLINE

1- Software development costs and factors

2- What is Test-Driven Development ?

3- Why Practice TDD ?

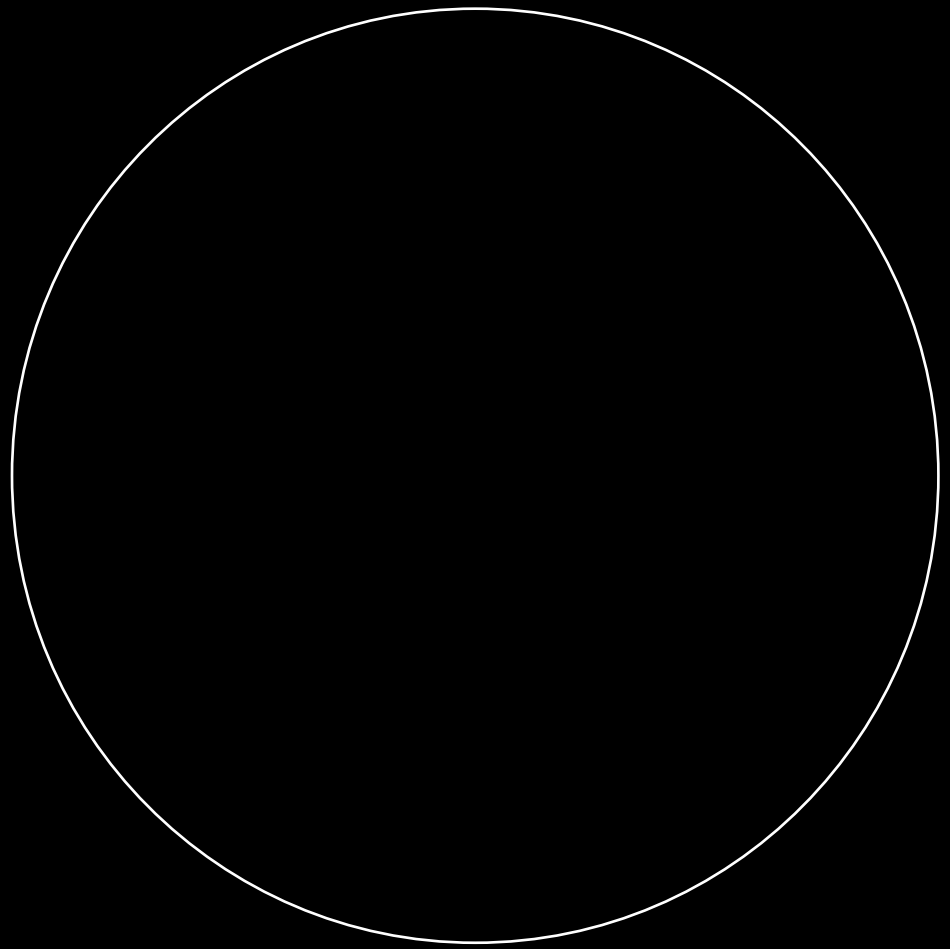
4- Fizz-Buzz TDD Demo

SOFTWARE DEVELOPMENT COSTS AND FACTORS

To understand why Test-Driven Development exists and what value it provides to software developers and businesses, we first need to look at the challenges making software today.

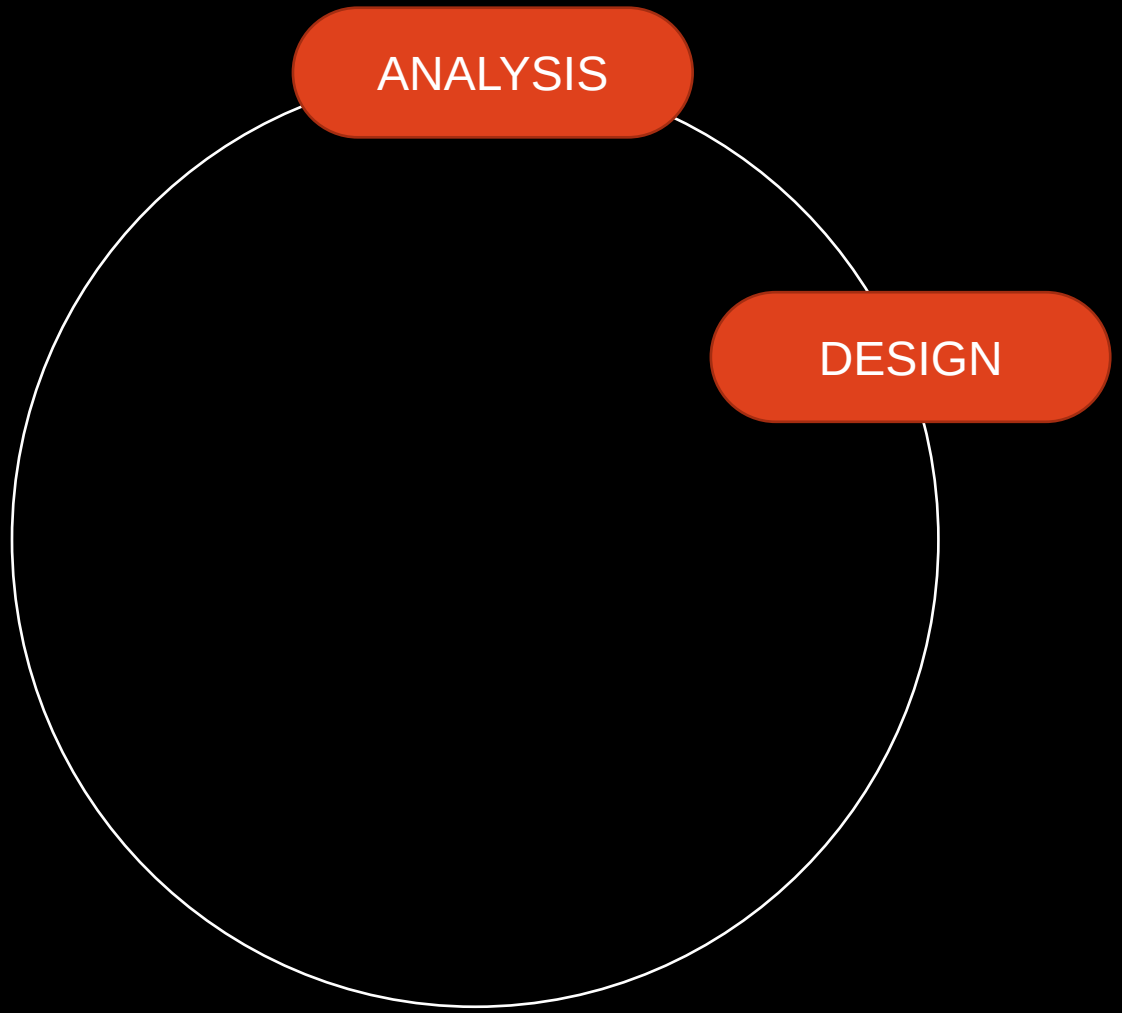
SOFTWARE DEVELOPMENT COSTS AND FACTORS

Phases to creating & delivering software



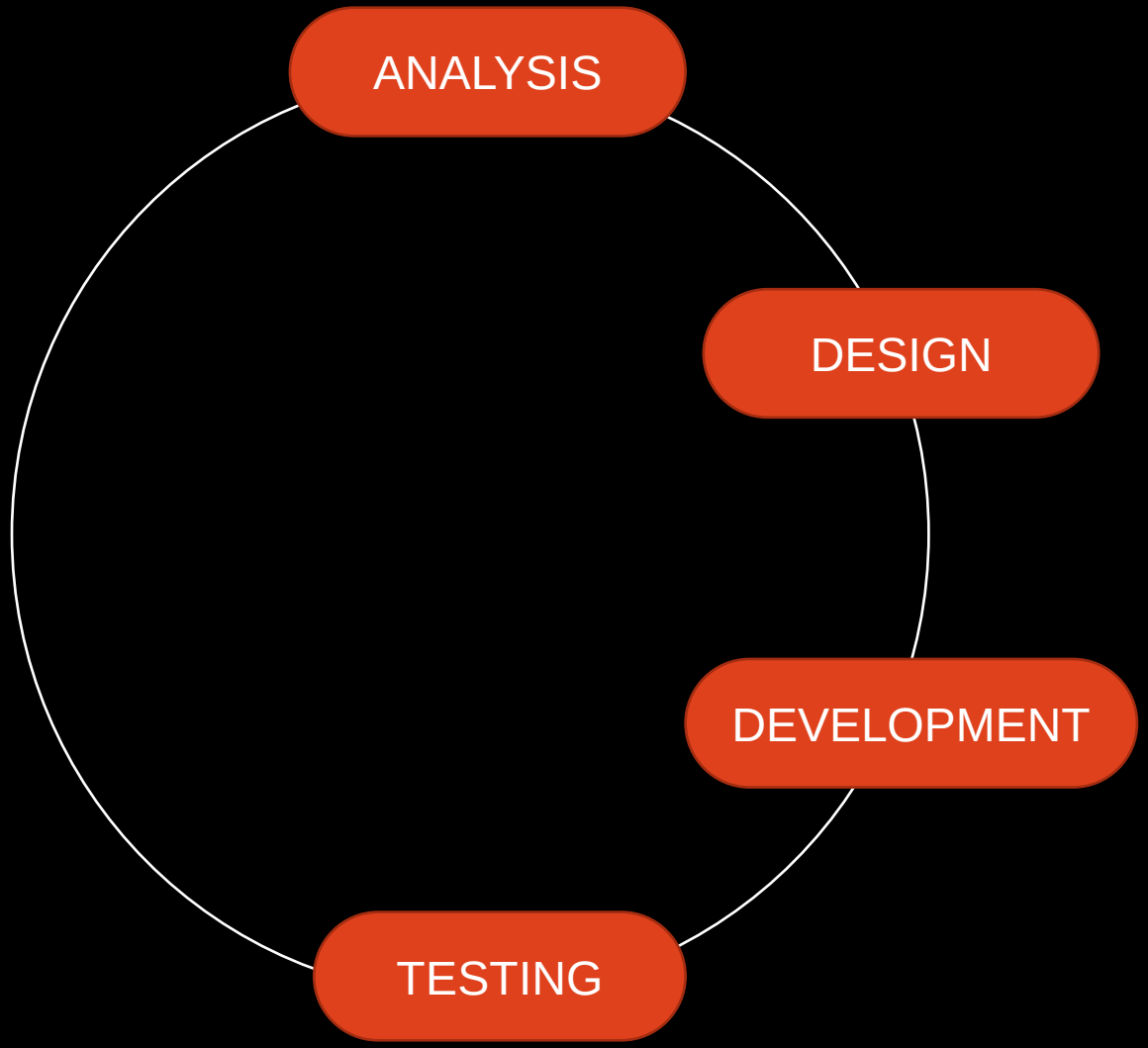
SOFTWARE DEVELOPMENT COSTS AND FACTORS

Phases to creating & delivering software



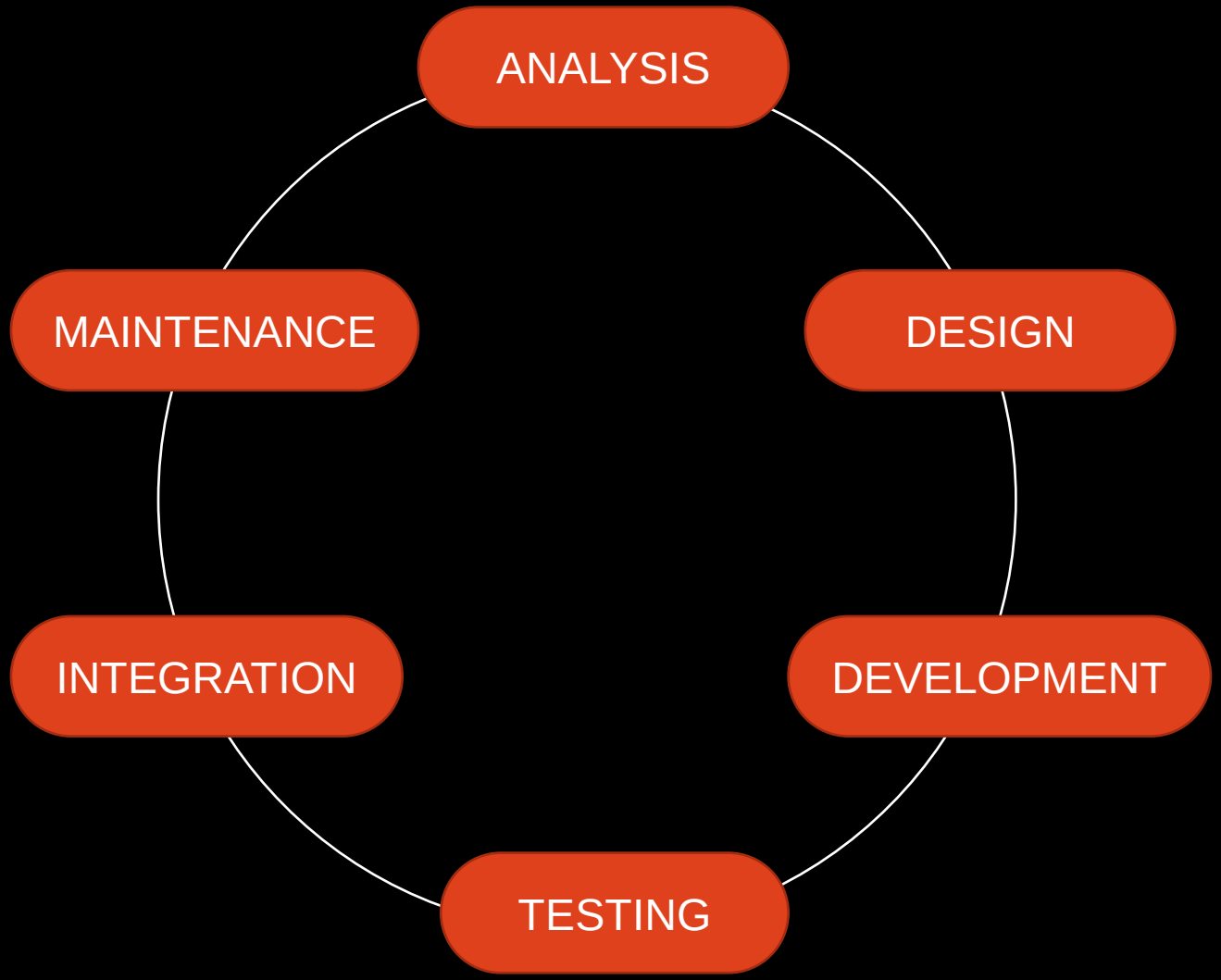
SOFTWARE DEVELOPMENT COSTS AND FACTORS

Phases to creating & delivering software



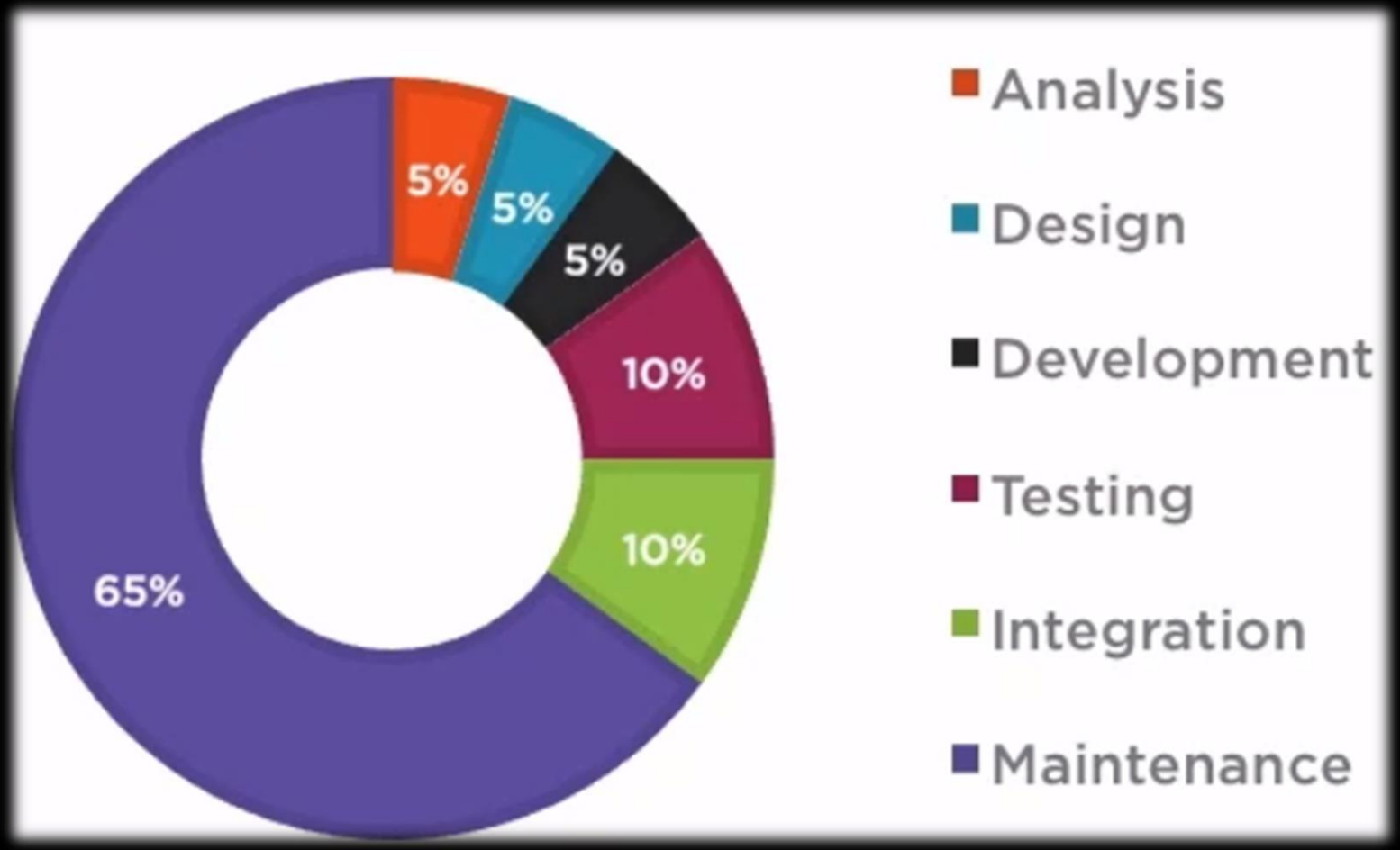
SOFTWARE DEVELOPMENT COSTS AND FACTORS

Phases to creating & delivering software



SOFTWARE DEVELOPMENT COSTS AND FACTORS

Maintenance is the most expensive cost in developing software.

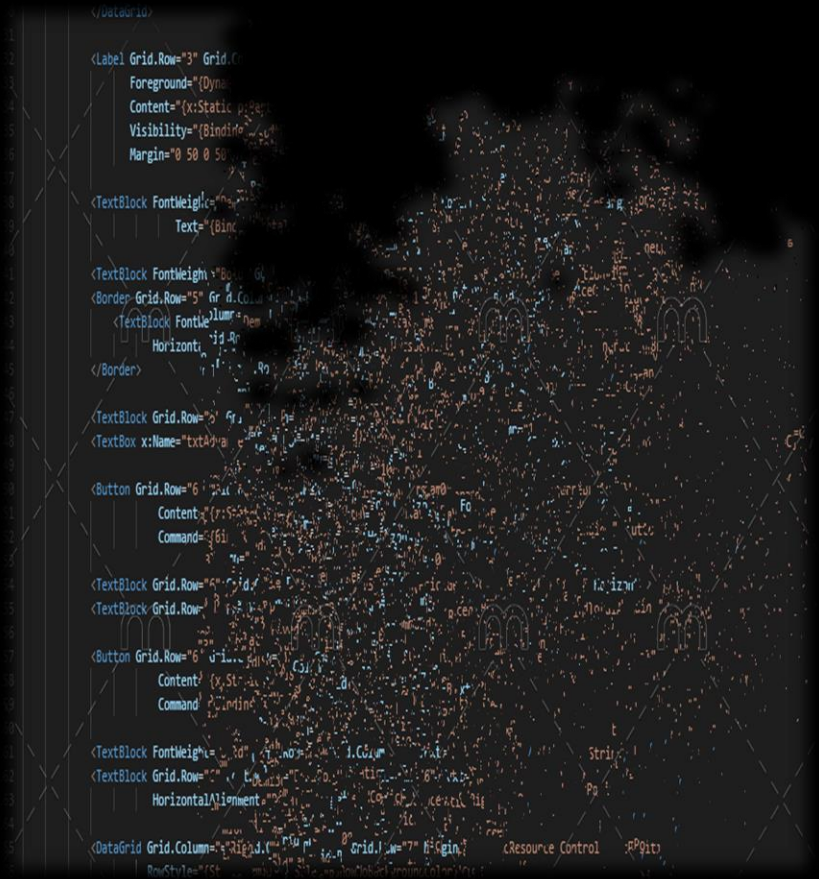


YES….

MAINTENANCE ACCOUNTS 65% OF ALL SOFTWARE DEVELOPMENT COST!

SOFTWARE DEVELOPMENT COSTS AND FACTORS

Why maintaining the software we build is so expensive?



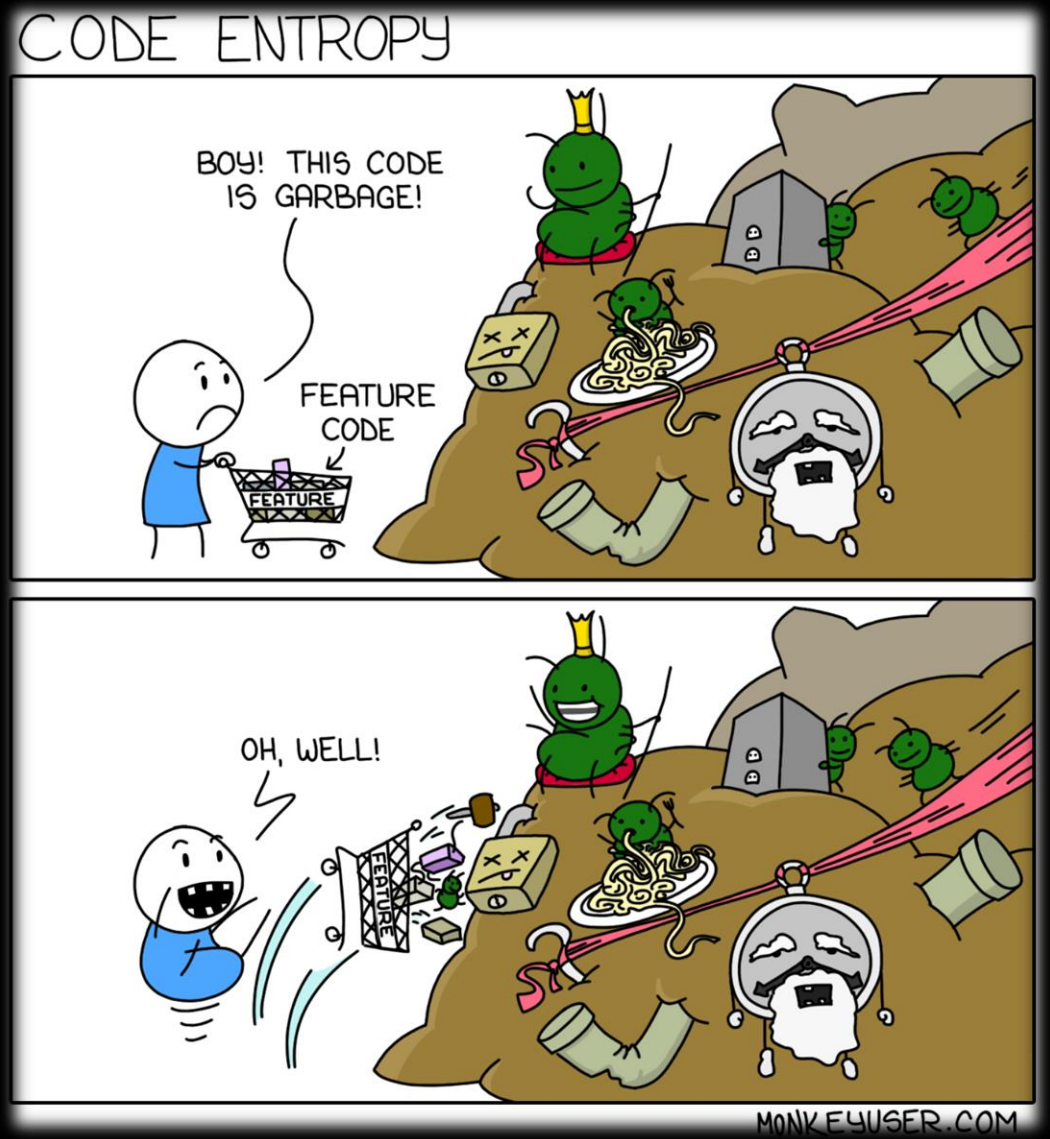
Software entropy

The chaos: The *law of entropy* is driving the whole universe. The term means that all things strive to chaos and disorder

Unfortunately the same principles are valid for the software development. Software entropy is the *measure of software* that reflects the *complexity of its maintenance*.

When a program *is modified*, its (disorder or entropy) *complexity will increase*.

SOFTWARE DEVELOPMENT COSTS AND FACTORS



SOFTWARE DEVELOPMENT COSTS AND FACTORS

Software Rot

Is a slow deterioration of software quality over time or its diminishing responsiveness that will eventually lead to software becoming faulty, unusable, or in need of upgrade.

Software that is not currently being used gradually becomes unusable as the remainder of the application changes. Changes in user requirements and the software environment also contribute to the deterioration.

SOFTWARE DEVELOPMENT COSTS AND FACTORS

A lot of these problems are typically associated with what we call

Legacy Code

SOFTWARE DEVELOPMENT COSTS AND FACTORS

A lot of these problems are typically associated with what we call
Legacy Code

1) ***"Is source code inherited from somebody else…"***

SOFTWARE DEVELOPMENT COSTS AND FACTORS

A lot of these problems are typically associated with what we call
Legacy Code

1) ***"source code inherited from somebody else…"***

2) ***"source code inherited from an older version of the software…"***

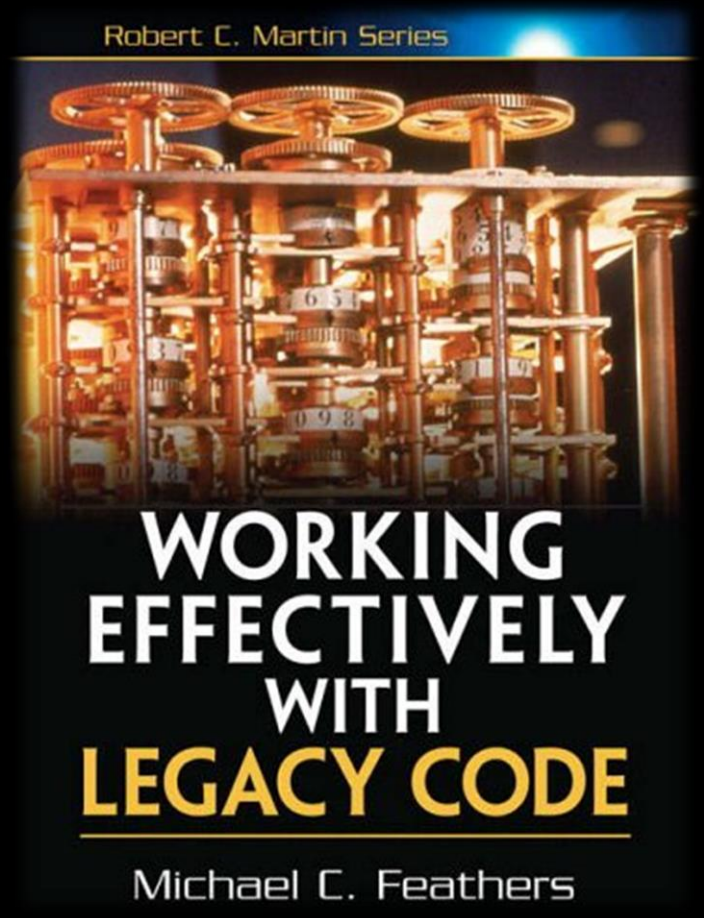
SOFTWARE DEVELOPMENT COSTS AND FACTORS

A lot of these problems are typically associated with what we call
Legacy Code

1) ***"source code inherited from somebody else…"***

2) ***"source code inherited from an older version of the software…"***

3) ***"source code without tests." M. Feathers***

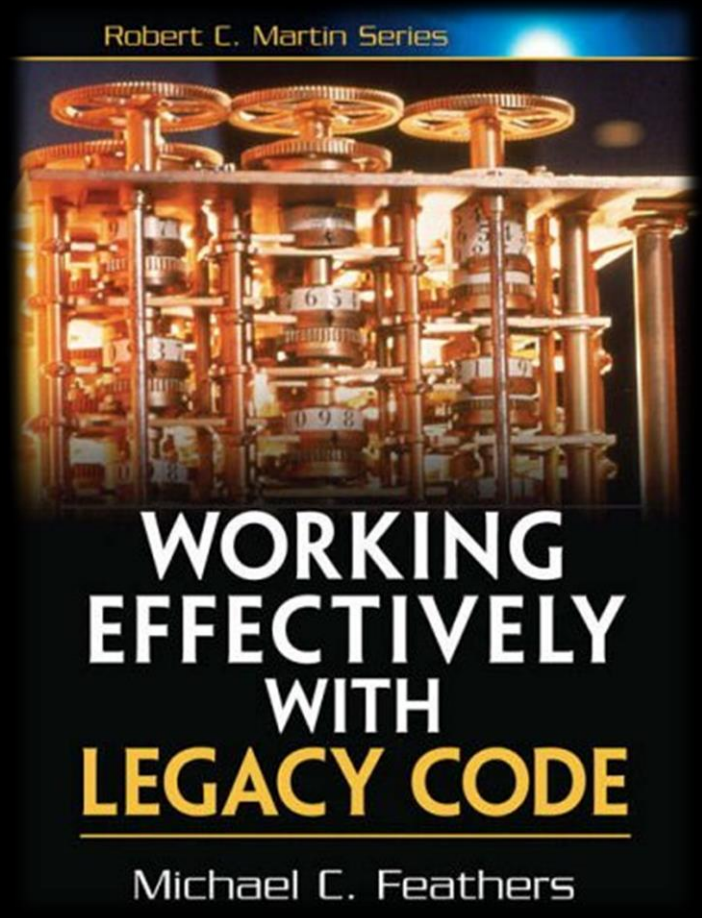


SOFTWARE DEVELOPMENT COSTS AND FACTORS

Source Code without tests

Source Code without automated tests is often code that is more brittle, more error prone to breaking when changed, and is likely validated very infrequently because it requires manual test suites, or perhaps no test suites at all.

Code without automated tests suffers from most of the shortcomings that make maintaining software much more expensive than it really needs to be.



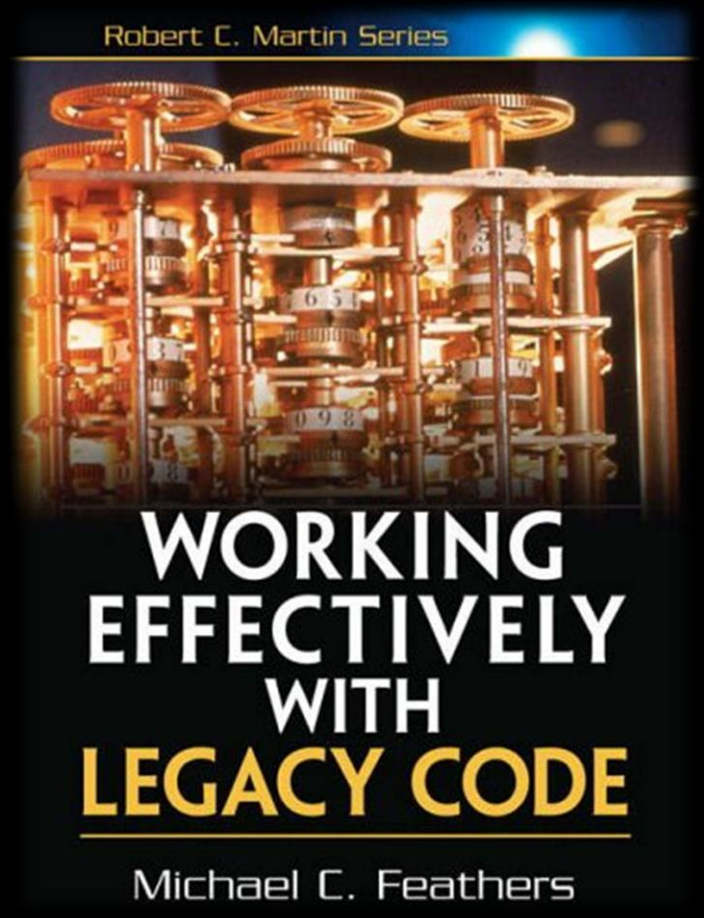
SOFTWARE DEVELOPMENT COSTS AND FACTORS

Source Code without tests

To me, *legacy code* is simply code without tests. I've gotten some grief for this definition. What do tests have to do with whether code is bad? To me, the answer is straightforward, and it is a point that I elaborate throughout the book:

> Code without tests is bad code. It doesn't matter how well written it is; it doesn't matter how pretty or object-oriented or well-encapsulated it is. With tests, we can change the behavior of our code quickly and verifiably. Without them, we really don't know if our code is getting better or worse.

You might think that this is severe. What about clean code? If a code base is very clean and well structured, isn't that enough? Well, make no mistake. I love



WHAT IS TEST-DRIVEN DEVELOPMENT?

To understand test-driven development, we first need to understand what a test is.

Test:

"is something that verifies your code works as it is expected to. This can be verification of quality, performance, or even reliability, as well as functionality before it is taken into widespread use"

WHAT IS TEST-DRIVEN DEVELOPMENT?

To understand test-driven development, we first need to understand what a test is.

Test:

"is something that verifies your code works as it is expected to. This can be verification of quality, performance, or even reliability, as well as functionality before it is taken into widespread use"

And our tests should verify three things:

WHAT IS TEST-DRIVEN DEVELOPMENT?

To understand test-driven development, we first need to understand what a test is.

Test:

"is something that verifies your code works as it is expected to. This can be verification of quality, performance, or even reliability, as well as functionality before it is taken into widespread use"

And our tests should verify three things:

Does the code satisfy the requirements as specified by the customer?

WHAT IS TEST-DRIVEN DEVELOPMENT?

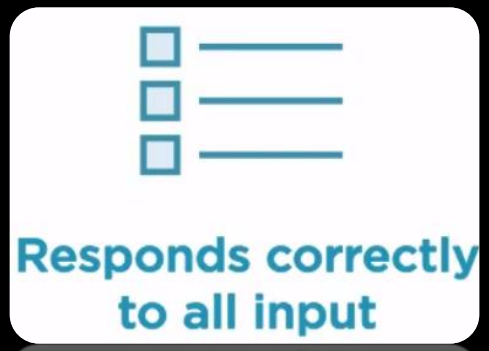
To understand test-driven development, we first need to understand what a test is.

Test:

"is something that verifies your code works as it is expected to. This can be verification of quality, performance, or even reliability, as well as functionality before it is taken into widespread use"

And our tests should verify three things:

Does the code satisfy the requirements as specified by the customer?



Tests also verify that the code responds properly to all inputs…

WHAT IS TEST-DRIVEN DEVELOPMENT?

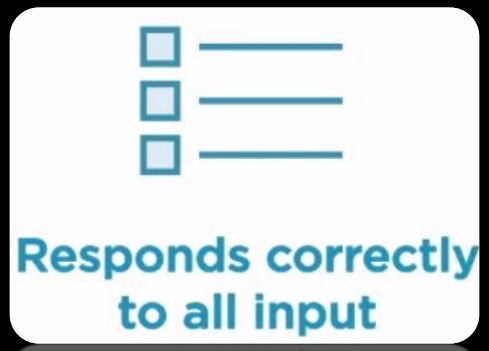
To understand test-driven development, we first need to understand what a test is.

Test:

"is something that verifies your code works as it is expected to. This can be verification of quality, performance, or even reliability, as well as functionality before it is taken into widespread use"

And our tests should verify three things:

Does the code satisfy the requirements as specified by the customer?



Tests also verify that the code responds properly to all inputs…

Tests can verify that the code has acceptable performance

WHAT IS TEST-DRIVEN DEVELOPMENT?

TEST-DRIVEN DEVELOPMENT

"Is a software development process that relies on the repetition of a very short development cycle.
Requirements are turned into very specific test cases, then the code is create and improved to pass that specific tests only." - Wikipedia

WHAT IS TEST-DRIVEN DEVELOPMENT?

TEST-DRIVEN DEVELOPMENT

"Is a software development process that relies on the repetition of a very short development cycle.
Requirements are turned into very specific test cases, then the code is create and improved to pass that specific tests only." - Wikipedia

So, the process is driven by the process of writing tests…

WHAT IS TEST-DRIVEN DEVELOPMENT?

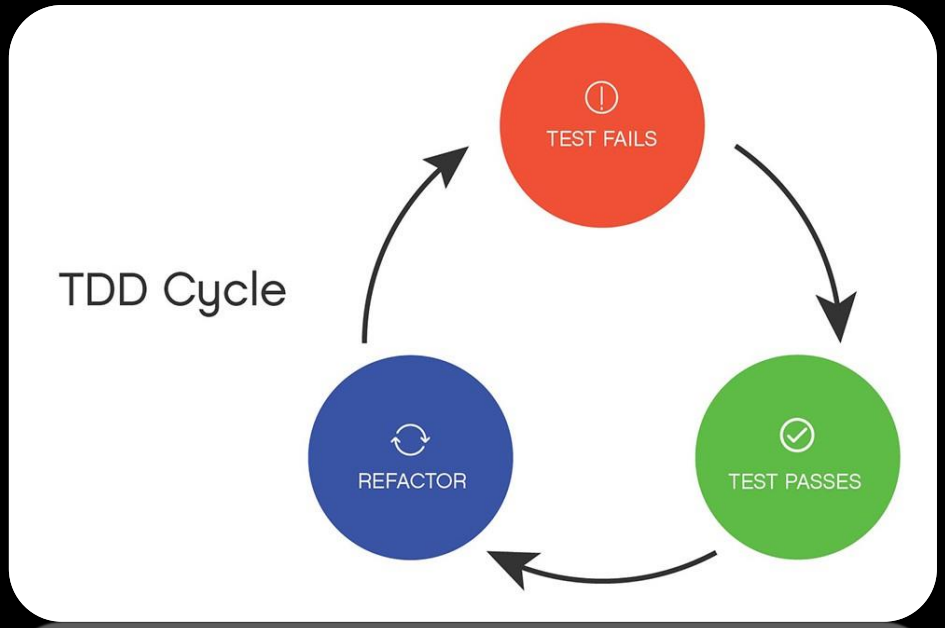
The main mantra that you are likely to hear over and over about test-driven development is….

RED / GREEN / REFACTOR

WHAT IS TEST-DRIVEN DEVELOPMENT?

The main mantra that you are likely to hear over and over about test-driven development is….

RED / GREEN / REFACTOR



WHAT IS TEST-DRIVEN DEVELOPMENT?

RED / GREEN / REFACTOR

First, **you write a test that fails**. This test is verifying one part of proper behavior for the system under test.

WHAT IS TEST-DRIVEN DEVELOPMENT?

RED / GREEN / REFACTOR

First, **you write a test that fails**. This test is verifying one part of proper behavior for the system under test.

Next, do the minimal work necessary to make the test pass, the goal here to only write the code necessary to make the system behave as expected.

WHAT IS TEST-DRIVEN DEVELOPMENT?

RED / GREEN / REFACTOR

First, **you write a test that fails**. This test is verifying one part of proper behavior for the system under test.

Next, do the minimal work necessary to make the test pass, the goal here to only write the code necessary to make the system behave as expected.

Finally, refactor the code you wrote to clean it up while having the test continually pass after each refactoring.

WHAT IS TEST-DRIVEN DEVELOPMENT?

RED / GREEN / REFACTOR

You are continually getting feedback on the behavior of the system and iteratively improving the software with each successive test.

WHY PRACTICE TDD?

Busines Benefits

It provides a way to verify the requirements of the software that is being developed by writing tests and adhering to red/green/refactor, requirement verification is built in to the software development process itself

Catch regressions where previously working functionality stops working with a newer release. When the tests verify the requirements of the delivered software, regressions end up causing test failures, catching the problems as early as possible.

Finally, when you factor in this requirements verification and catching failures as soon as possible, it helps lower the maintenance cost of the software for the business

WHY PRACTICE TDD?

Developer Benefits

Using TDD we have a **design-first mentality**. By **writing failing tests**, we need to think about how interactions with our software needs to **happen in order to make our features possible**.

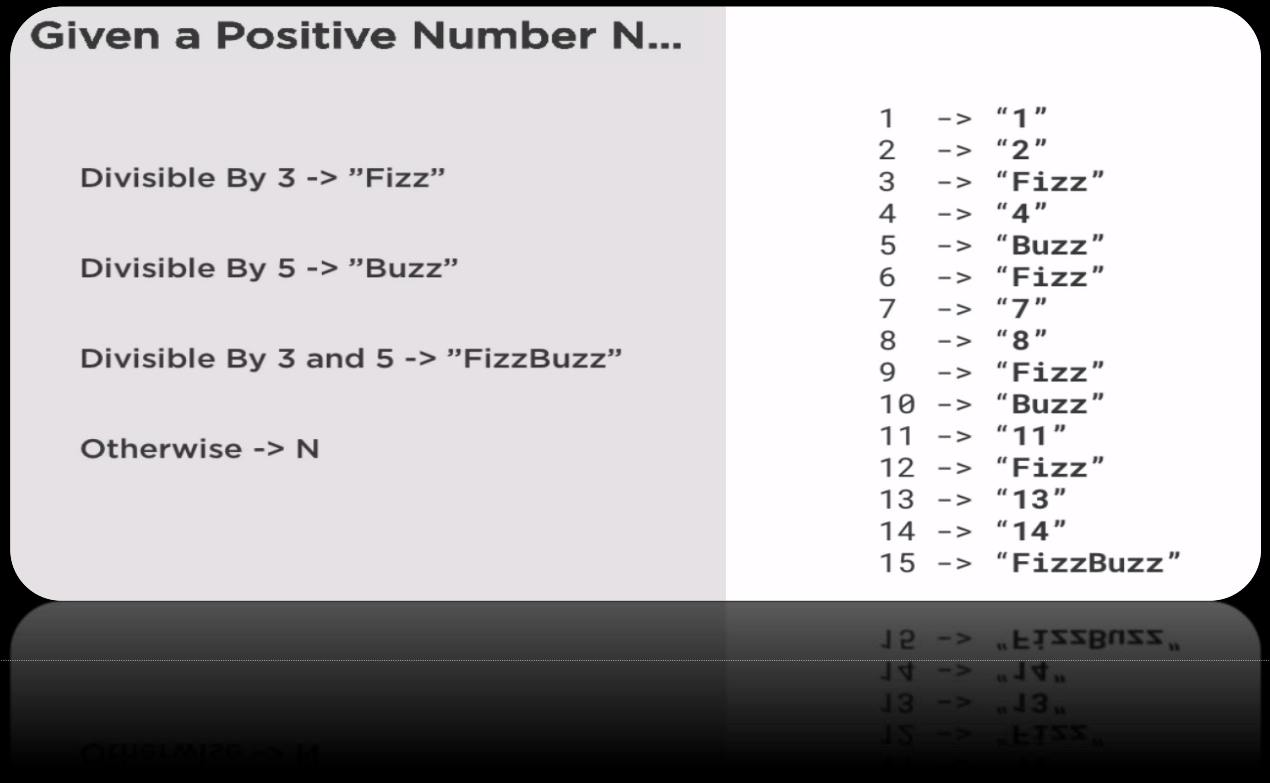
Prevent us from **over-engineering** our code. By focusing on what is needed to make tests pass and to satisfy user expectations, we help **keep ourselves on the rails** and not get lost in architecture design.

Another benefit is increasing your developer momentum. Rather than **struggling for days to implement a single part of a feature**, the encouragement to **break features down to tests that leverage the red/green/refactor flow**, allows you to be making rapid progress and establish a circle of success.

Finally, you gain a lot **more confidence working with code** when you have a **large suite of tests backing you up**. You feel empowered to change code as necessary, even refactoring difficult and complicated parts of the system in order to make it easier to integrate new features.

Implementing FIZZ-BUZZ :

Write an algorithm that take a positive number N,
If the number if multiples of 3, return the String “Fizz”,
If the number if multiples of 5, return "Buzz",
and if number if multiples of both 3 and 5, return “FizzBuzz”,
otherwise return N.



LET’S CODE THIS…..

MARTIN ARNESI
martin.arnesi@endava.com