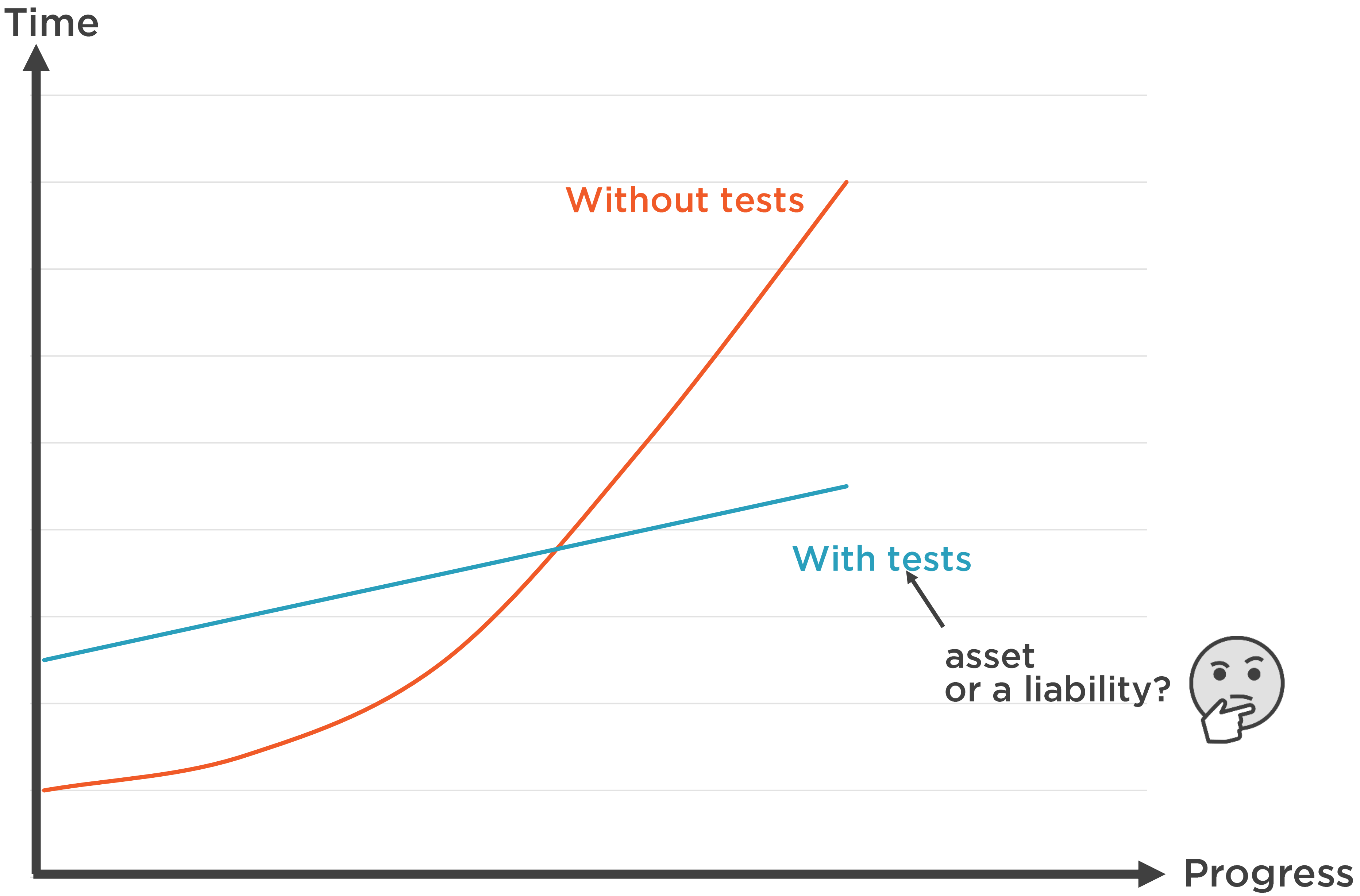
Fundamentals of Test Automation Principles in Java

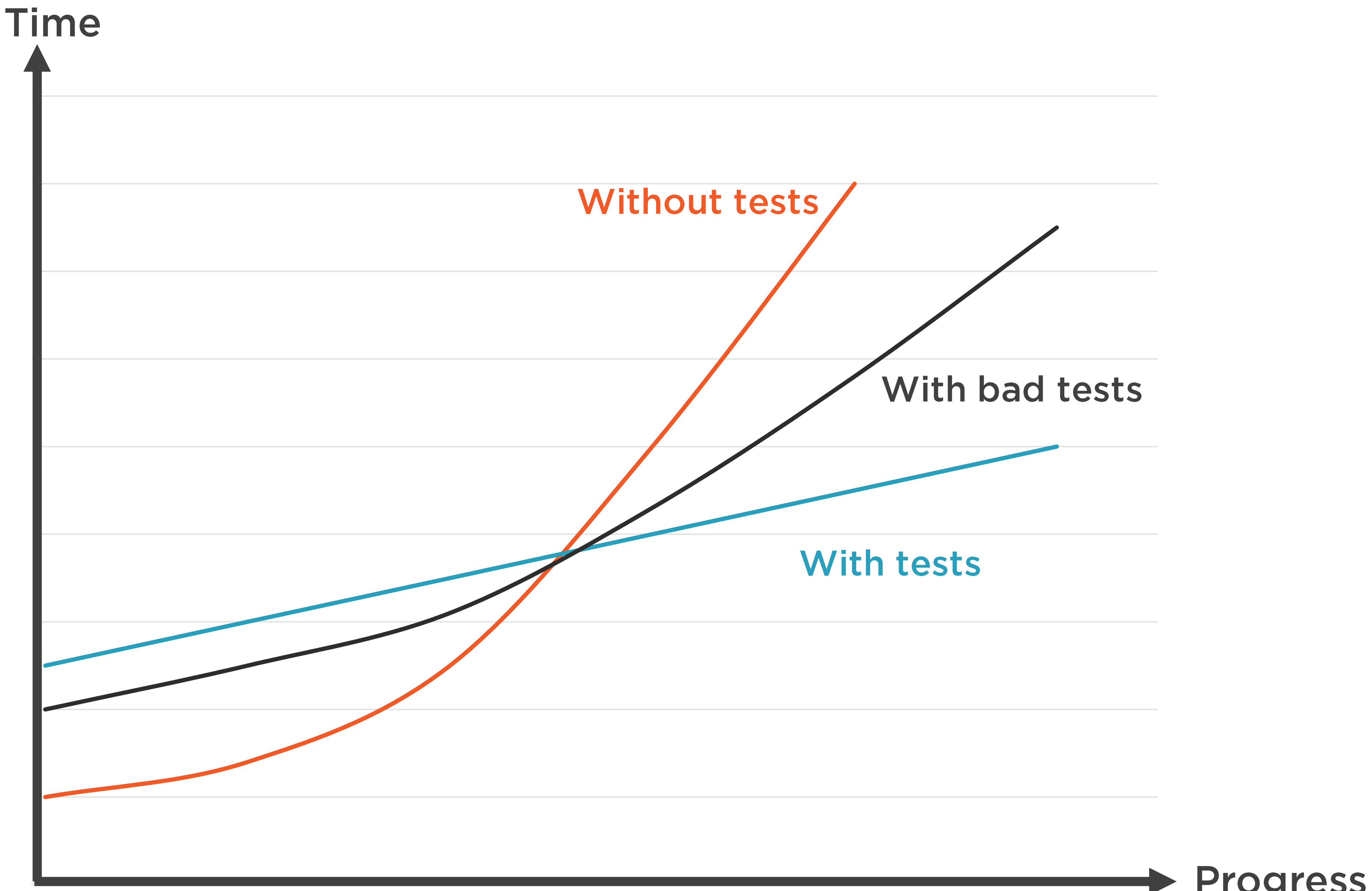
TESTS THAT BRING LITTLE VALUE

Andrejs Doronins
TEST AUTOMATION ENGINEER

Should we automate tests?

Yes!

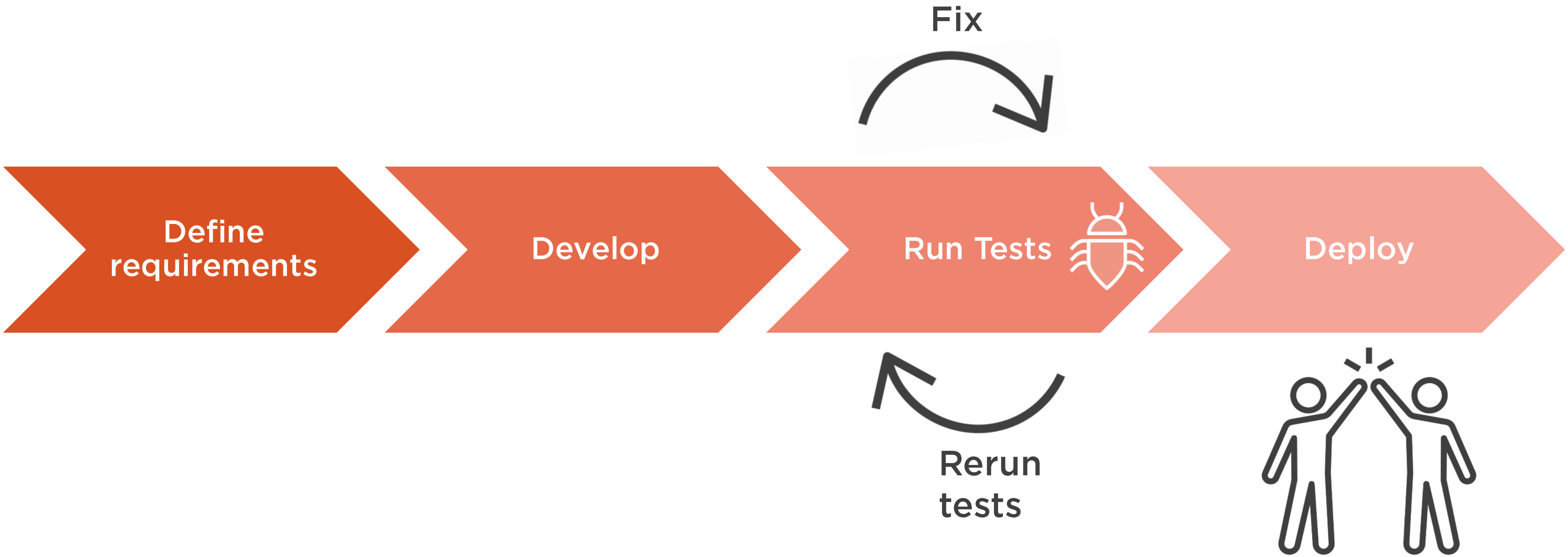

Time
Without tests
With tests
asset
or a liability?
Progress


Time
Without tests
With bad tests
With tests
Progress

Regression Testing

Regression (Bug)

Happens when an existing feature stops working correctly after a certain event, typically a code change



Define requirements

Develop

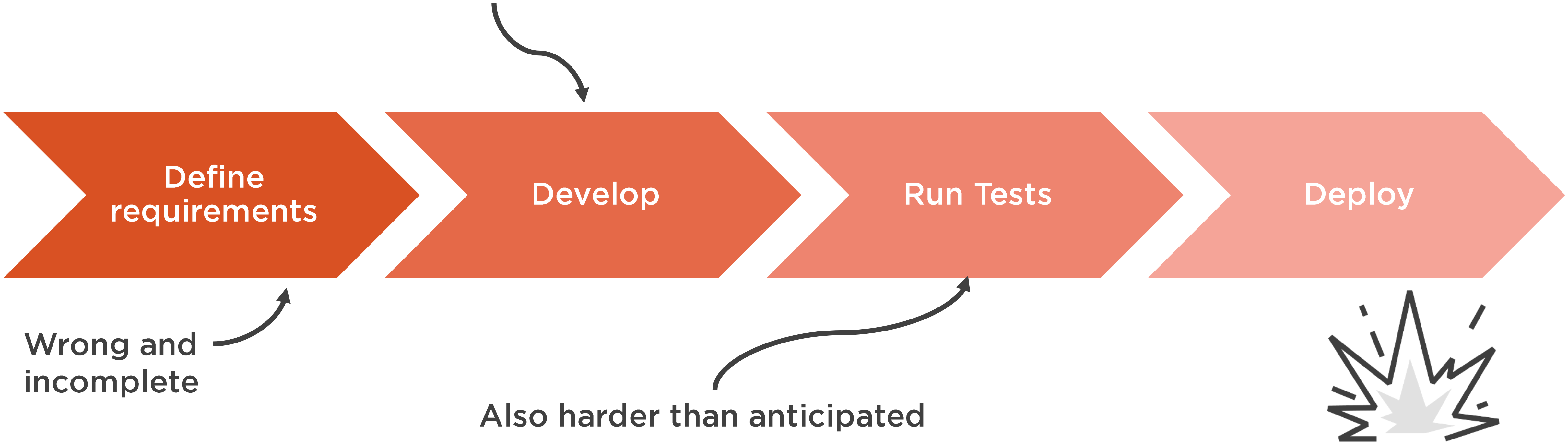
Run Tests

Deploy

Fix

Rerun tests

Harder than anticipated



Define requirements

Develop

Run Tests

Deploy

Wrong and incomplete

Also harder than anticipated

- What?
- Why?
- What can we do about it?

Kaboom!

Issues with Automated Tests

Slow

- everyone’s waiting…

Flaky

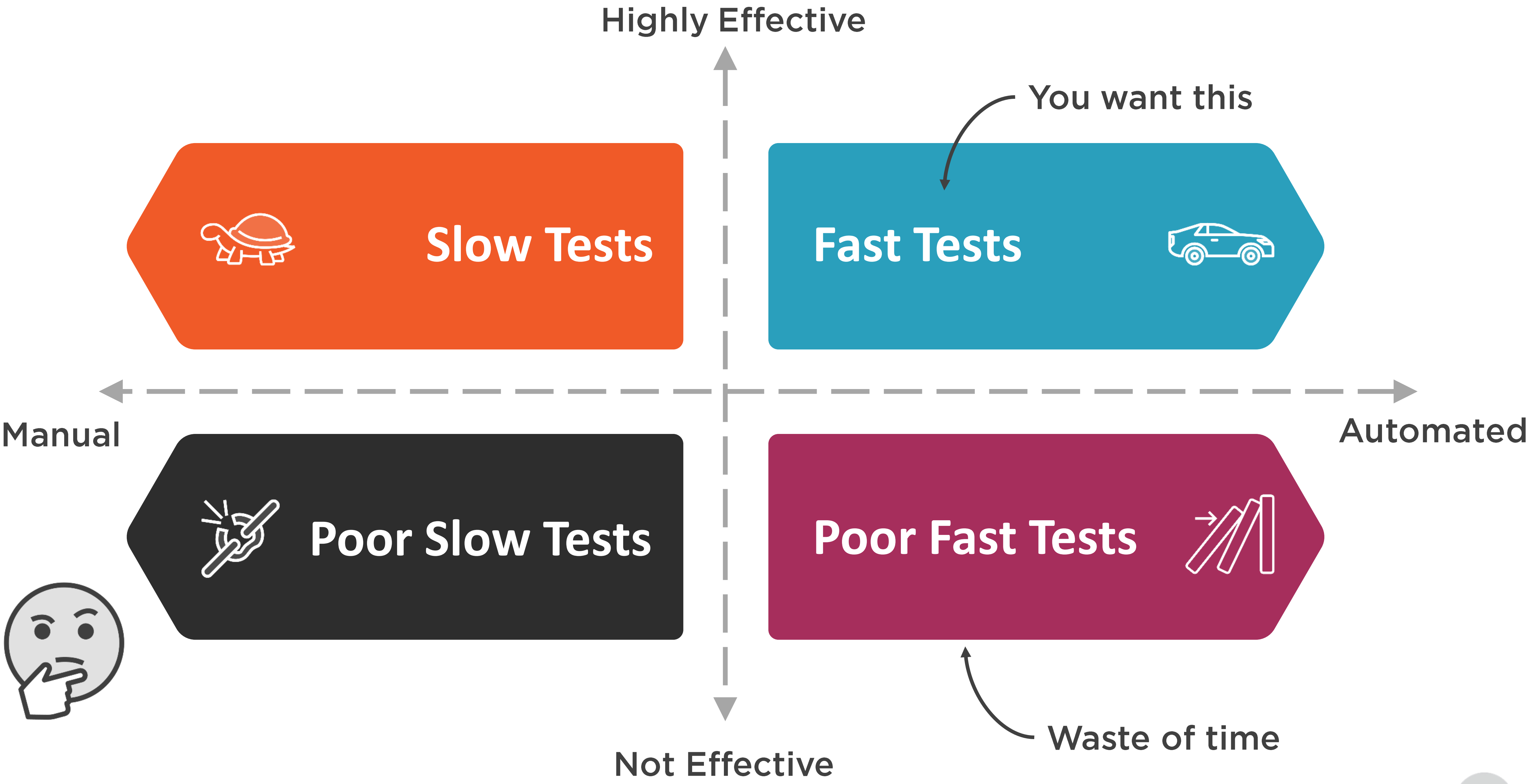
- fail sporadically, needs constant investigation

Obscure

- they make little sense

Little value

- mostly don’t prove anything worthwhile



Highly Effective

You want this

Slow Tests

Fast Tests

Manual

Automated

Poor Slow Tests

Poor Fast Tests

Waste of time

Not Effective

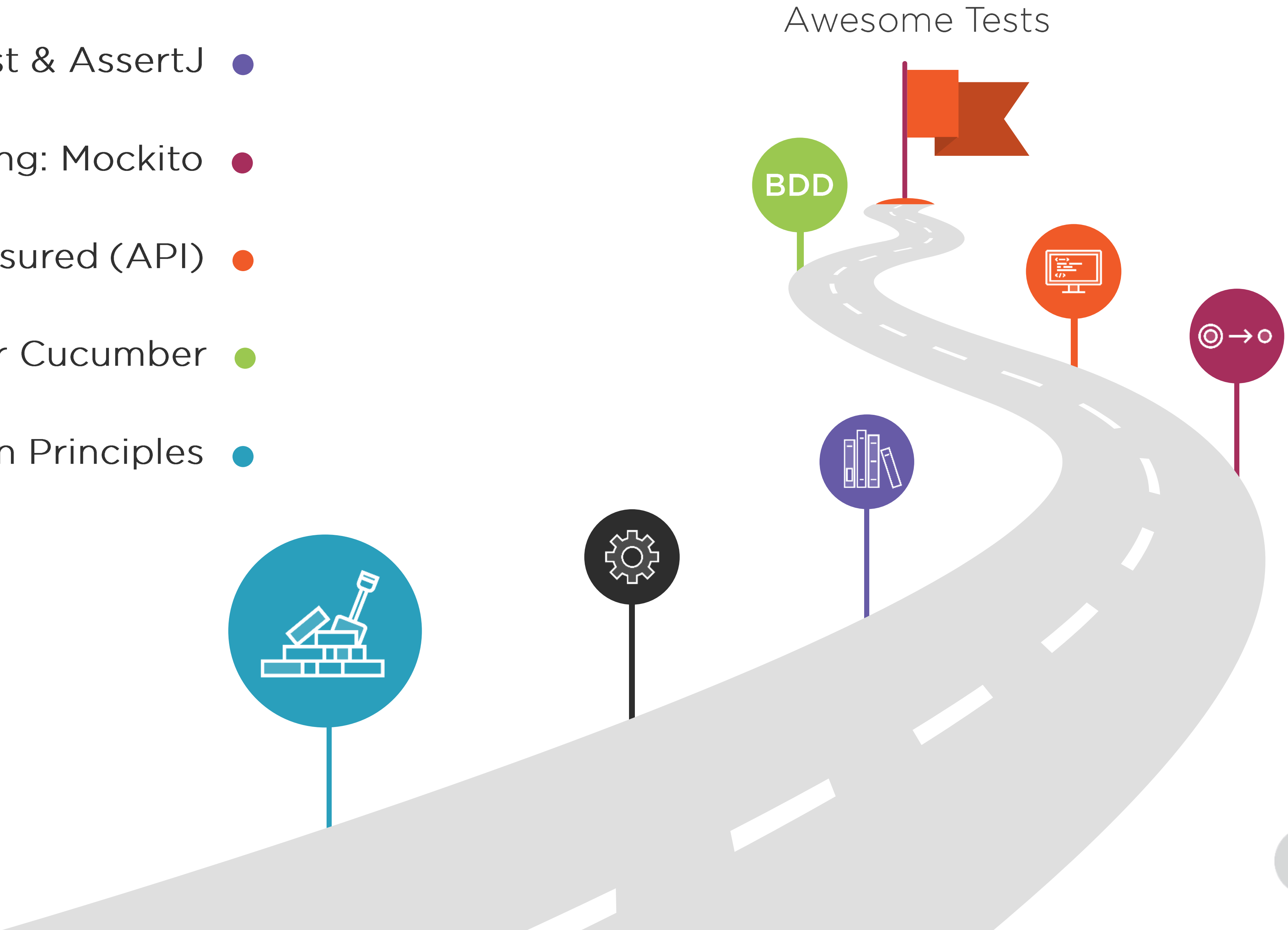
Why do we write slow, flaky
and unreliable tests?

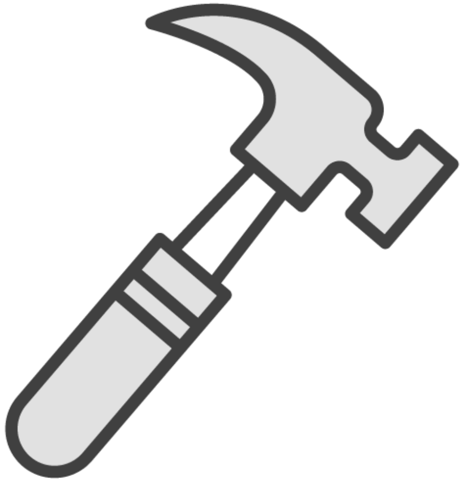
> “[…] Automation applied to an efficient operation will magnify the efficiency […] Automation applied to an inefficient operation will magnify the inefficiency”

Bill Gates

Automation simply magnifies whatever it automates

- Runners: JUnit or TestNG ●
- Helpers: Hamcrest & AssertJ ●
- Mocking: Mockito ●
- Selenium (UI) & RestAssured (API) ●
- BDD: JBehave or Cucumber ●
- Test Automation Principles ●



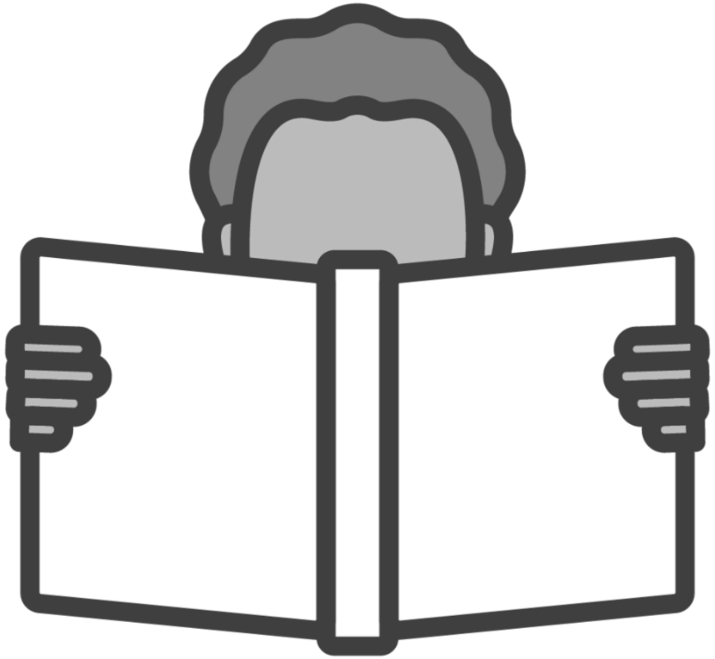


Using a shiny new tool

!=

Test Automation Engineer Pro

Resources on Testing

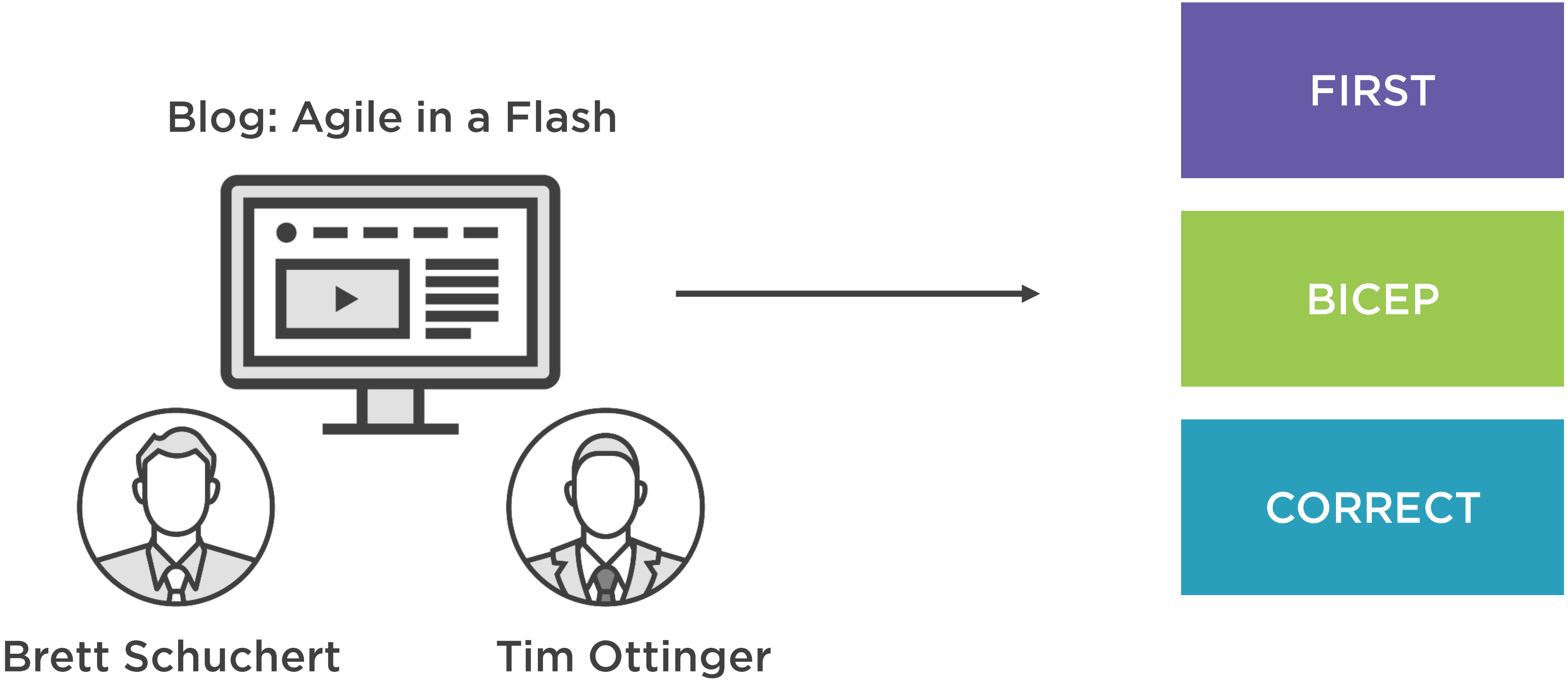


Books:

- The Art of Software Testing
- Lessons Learned in Software Testing
- A Practitioner's Guide to Software Test Design

Website:

- https://www.guru99.com/software-testing.html



Blog: Agile in a Flash

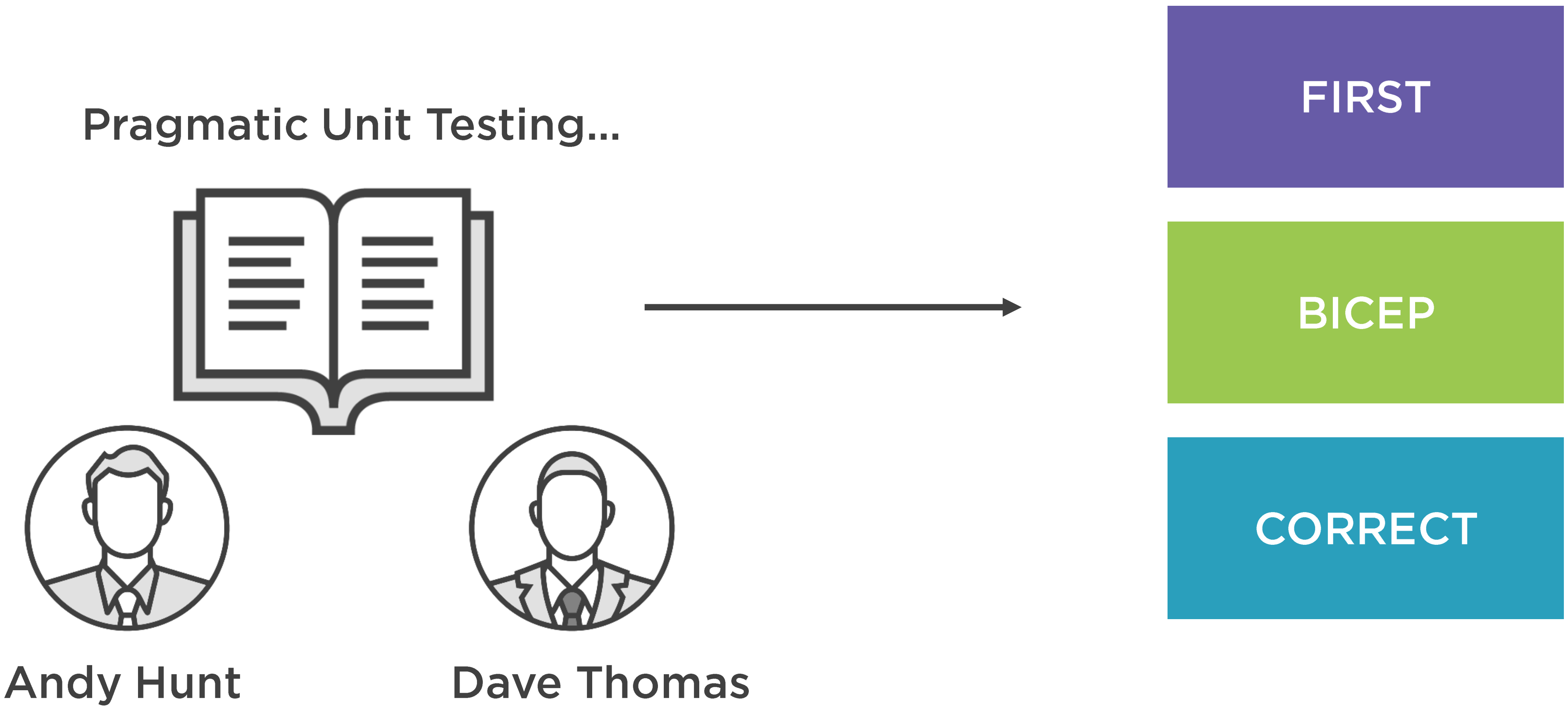
Brett Schuchert

Tim Ottinger

FIRST

BICEP

CORRECT



Pragmatic Unit Testing...

Andy Hunt

Dave Thomas

FIRST

BICEP

CORRECT

This course applies these principles to a variety of functional tests

Who This Course Is For

Software Developer

(Writing any kind of functional tests)

SDET

(Writing any kind functional tests)

Manual tester

(with programming basics)

Prerequisites

Programming experience (ideally Java)

- Any OOP language (e.g. C#) will do too

Basic experience writing or maintaining any kind of automated tests

Test Runner:

- Java: JUnit, TestNG; C#: nUnit, xUnit…

IDE:

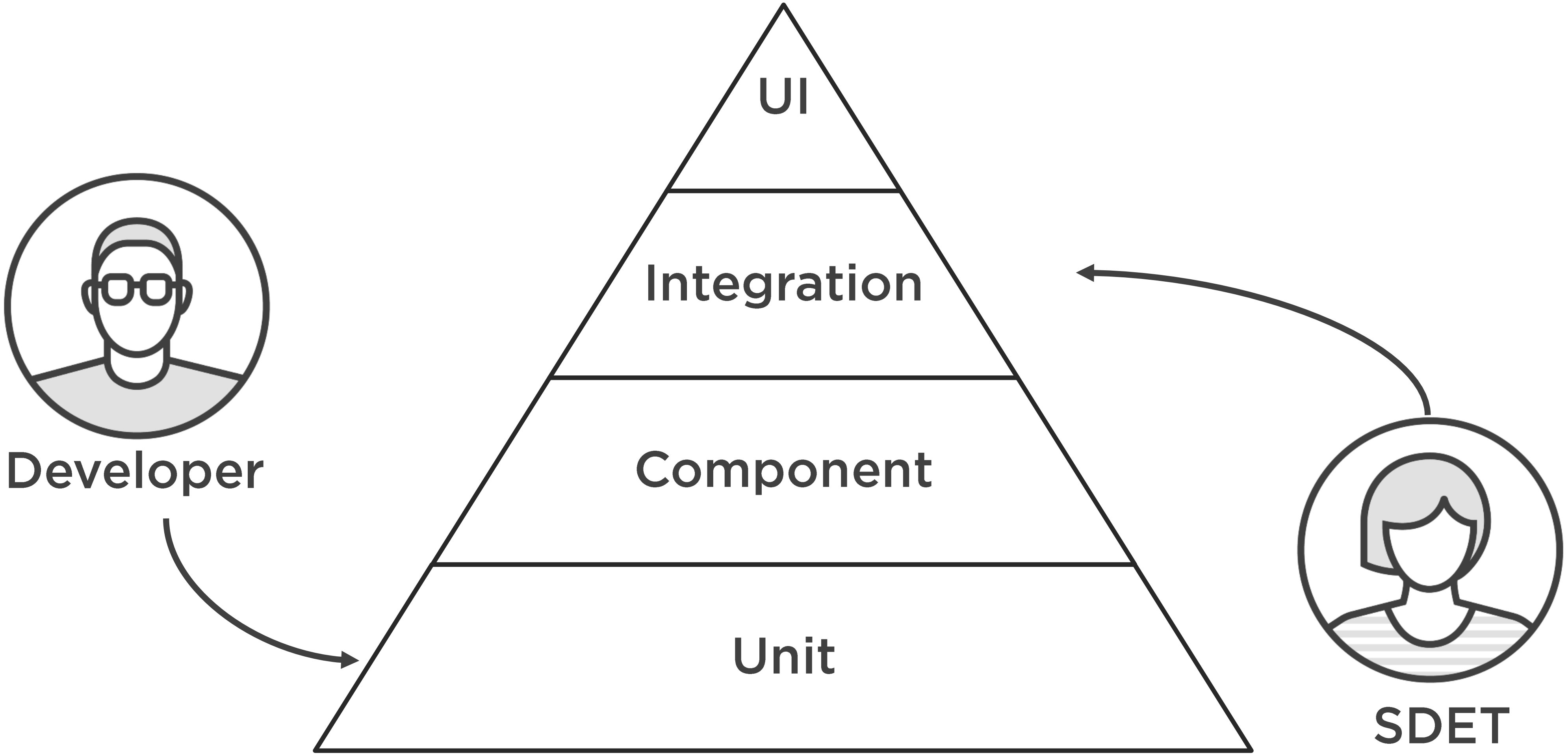
- IntelliJ, Eclipse, VisualStudio, etc.

Java 11

TestNG

IntelliJ

Testing Pyramid



UI

Integration

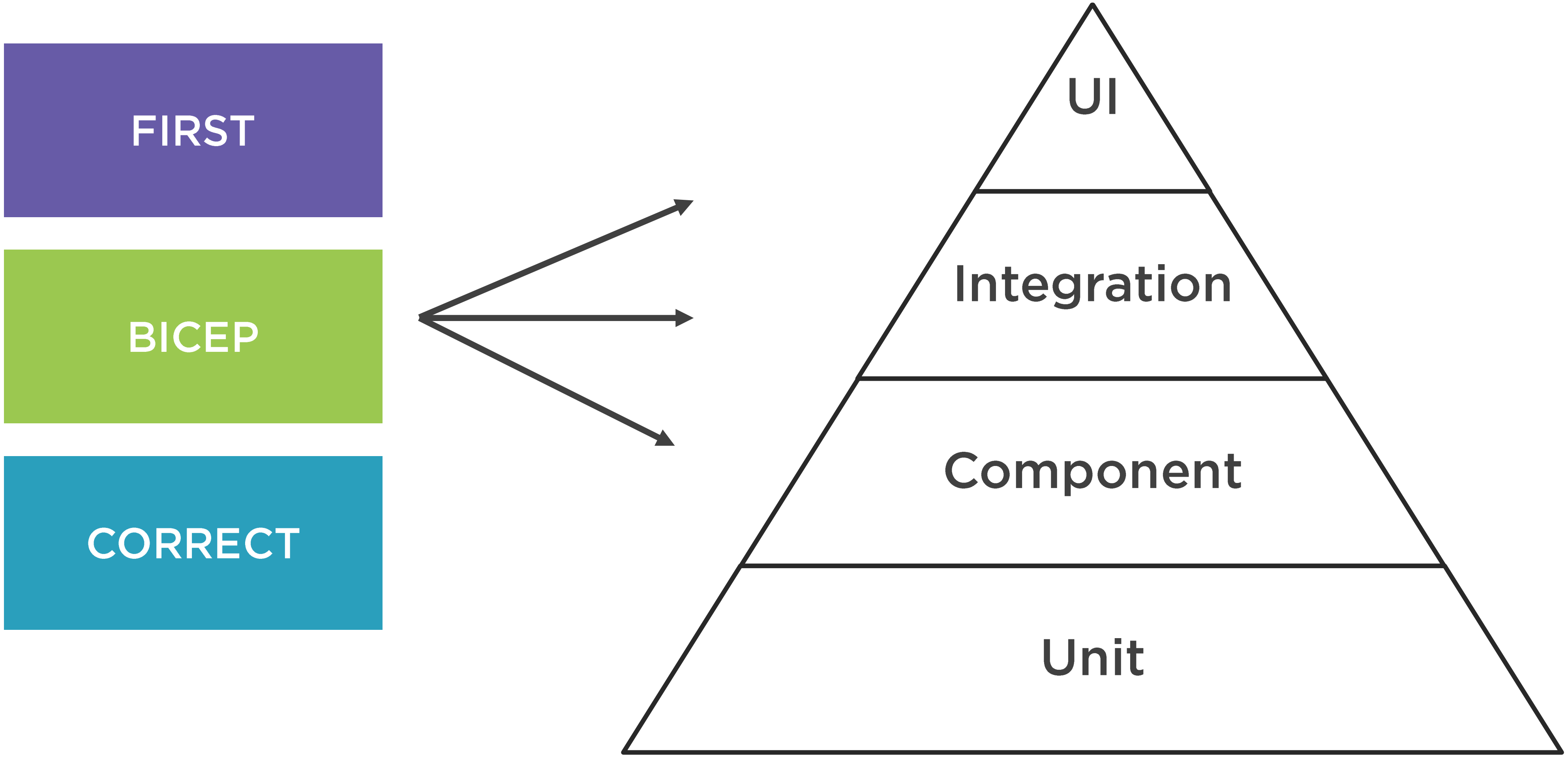
Component

Unit

Developer

SDET

Testing Pyramid



FIRST

BICEP

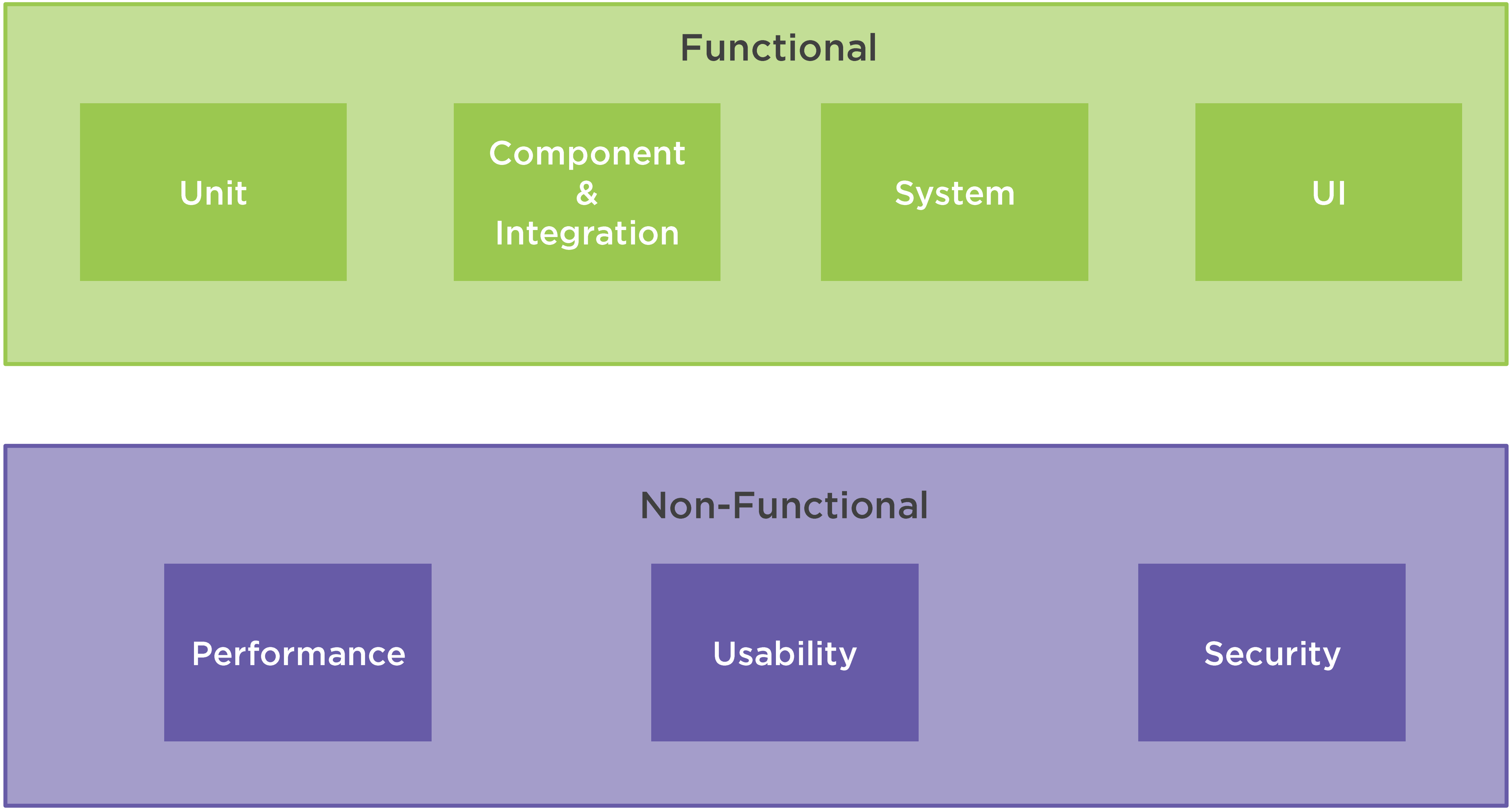
CORRECT

UI

Integration

Component

Unit



Functional

Unit

Component
&
Integration

System

UI

Non-Functional

Performance

Usability

Security

Overview

Course Purpose: writing tests of highest possible value

1. Recognize valuable tests (BICEP, CORRECT)
2. Write valuable tests (FIRST)

Tests should be FIRST

Leveraging BICEP Principles

Making tests CORRECT

Fix and improve what you already have first

You can watch the modules in any order