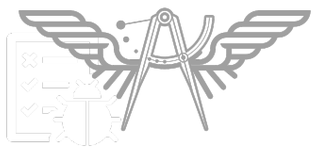
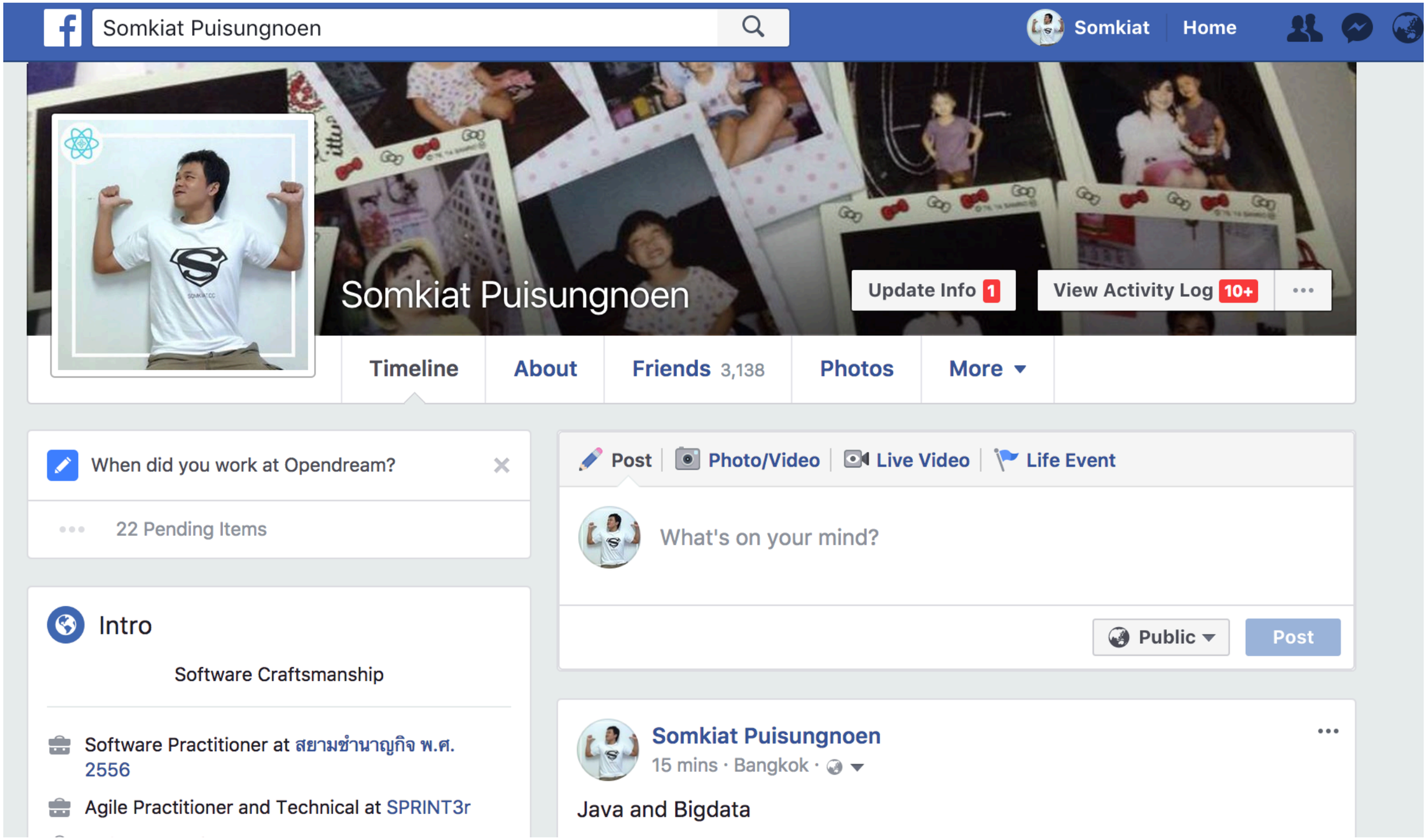
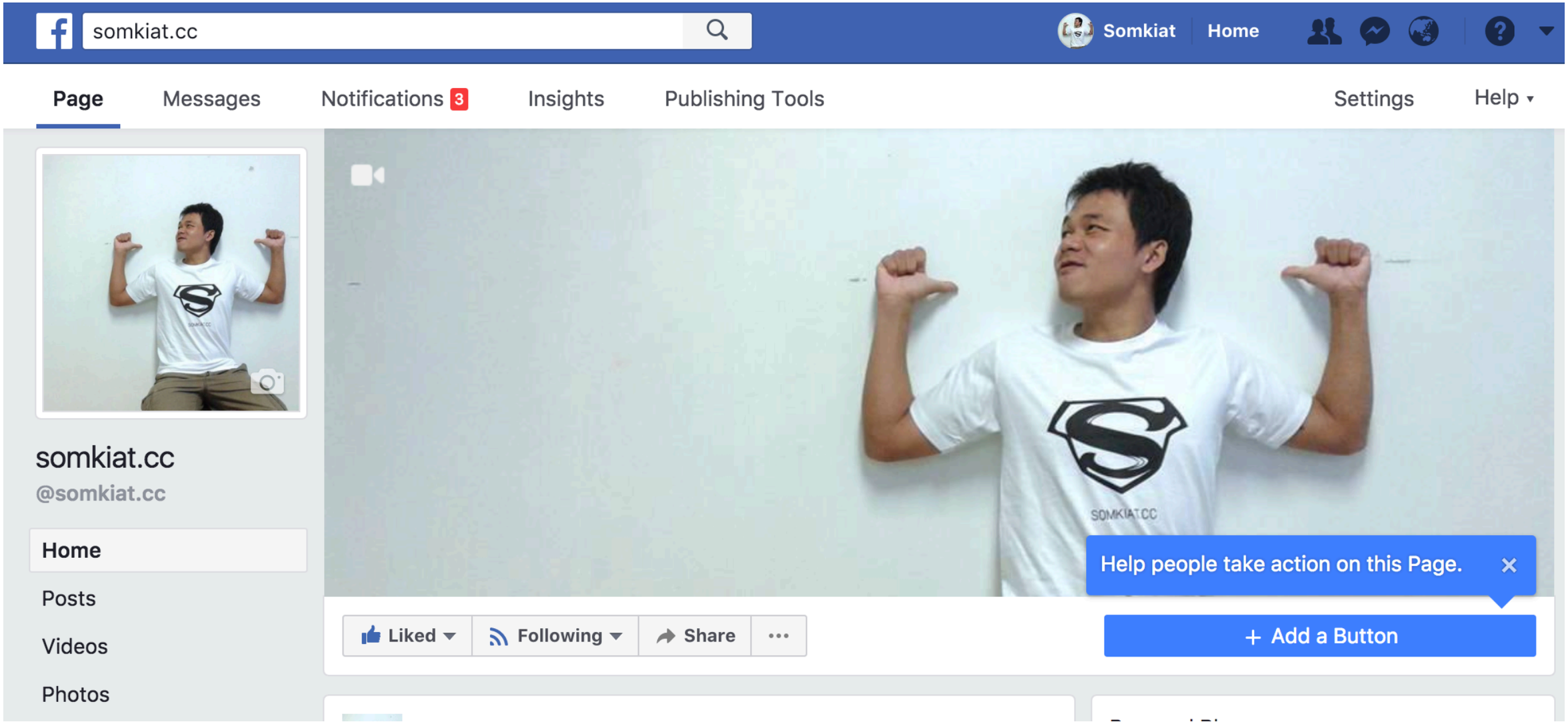
สยามชำนาญกิจ
SIAM CHAMNAN KIT

Quality Practice for Developers







Quality Practice for Developers

KBTG
KASIKORN
BUSINESS-TECHNOLOGY GROUP

Class Overview

Testing in general

- What kind of test should we write?
- Testing Pyramid
- Testing Techniques
- Automate Tests
- **Activity 01**

Automate Unit Tests

- Unit Tests Benefits
- Unit Tests - Simple Steps
- Test-driven development (TDD) - Concept
- Unit Tests And Mocks - Overview
- **Workshop 01 – (Unit Tests in Practices)**

Software Engineering Practices

- Simple Design
- Design Patterns – Overview
- Clean Code
- **Workshop 02 (Code Refactor)**

Forming Team Method of Working

- Code Smell/Code Review/Merge Request
- Code Review Tool – SonarQube
- Time Of Improvement
- **Workshop 03 (Quality Practices)**

Module #1
Testing in general

Technical excellence

TECHNICAL
EXCELLENCE

https://less.works/less/technical-excellence/index

SPECIFICATION BY EXAMPLE

CONTINUOUS INTEGRATION

CONTINUOUS DELIVERY

TEST AUTOMATION

ACCEPTANCE TESTING

TECHNICAL EXCELLENCE

ARCHITECTURE & DESIGN

CLEAN CODE

THINKING ABOUT TESTING

TEST-DRIVEN DEVELOPMENT

UNIT TESTING

SPECIFICATION BY EXAMPLE

CONTINUOUS INTEGRATION

CONTINUOUS DELIVERY

TEST AUTOMATION

TECHNICAL EXCELLENCE

ARCHITECTURE & DESIGN

ACCEPTANCE TESTING

CLEAN CODE

THINKING ABOUT TESTING

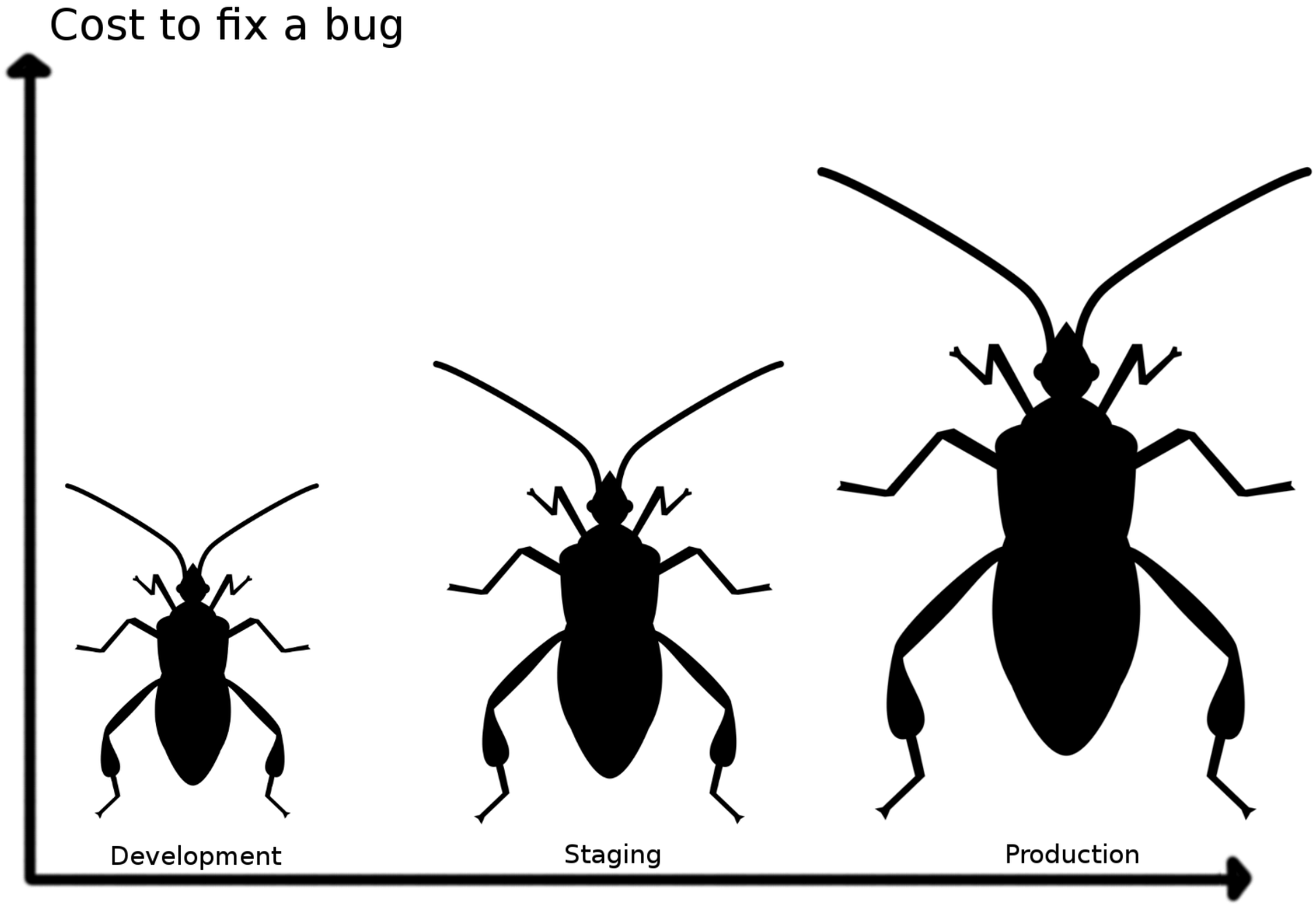
TEST-DRIVEN DEVELOPMENT

UNIT TESTING

"Program testing can be used to show the presence of bugs, but never their absence."

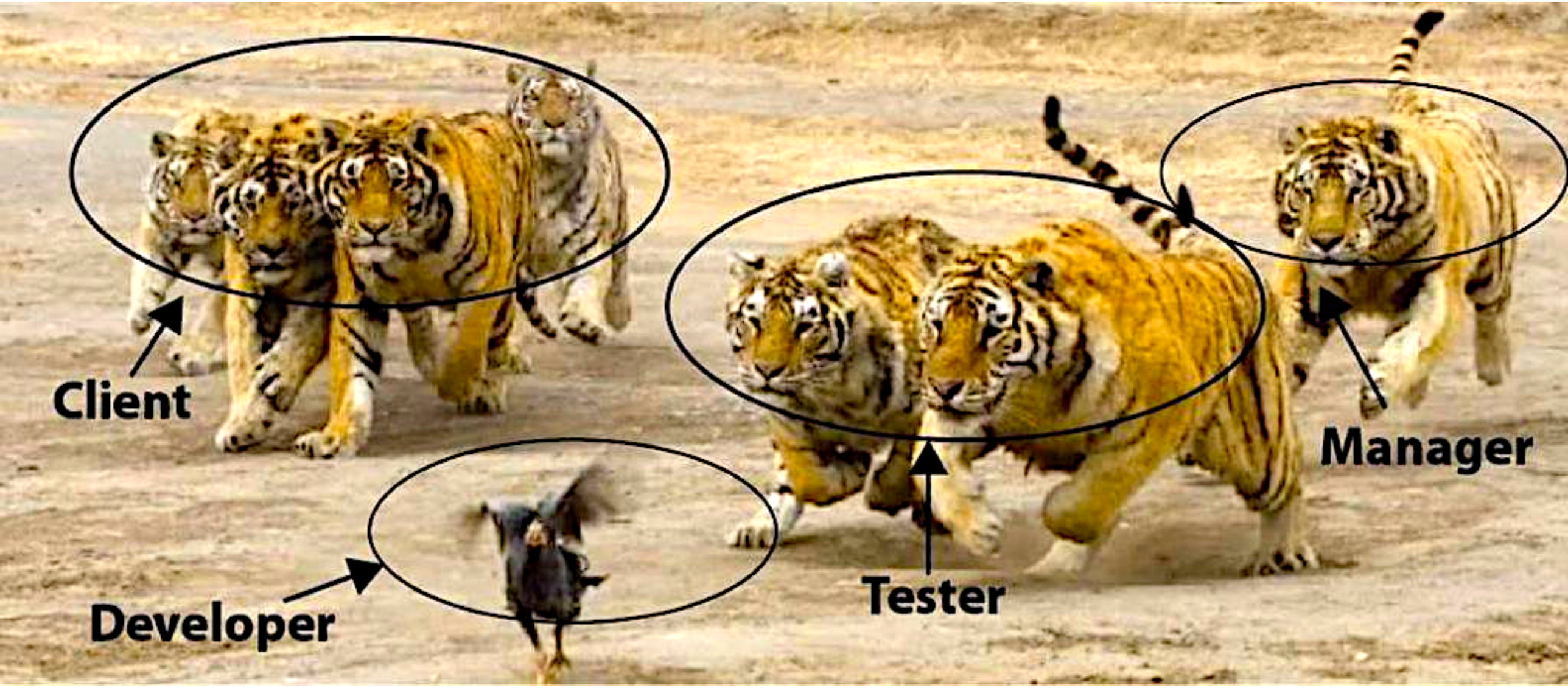
Edsger W. Dijkstra, 1970, Notes on Structured Programming

Start with Why …

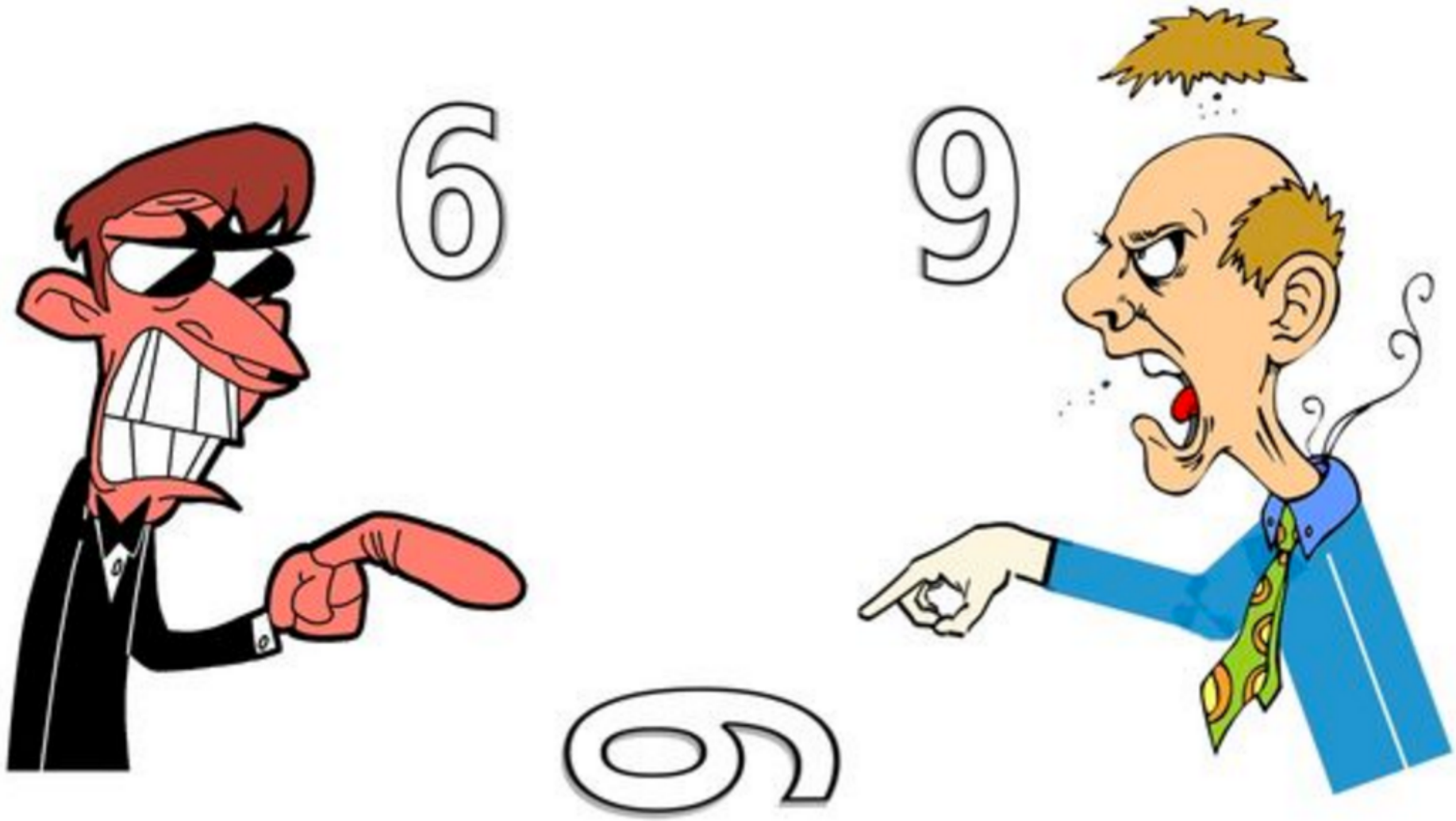


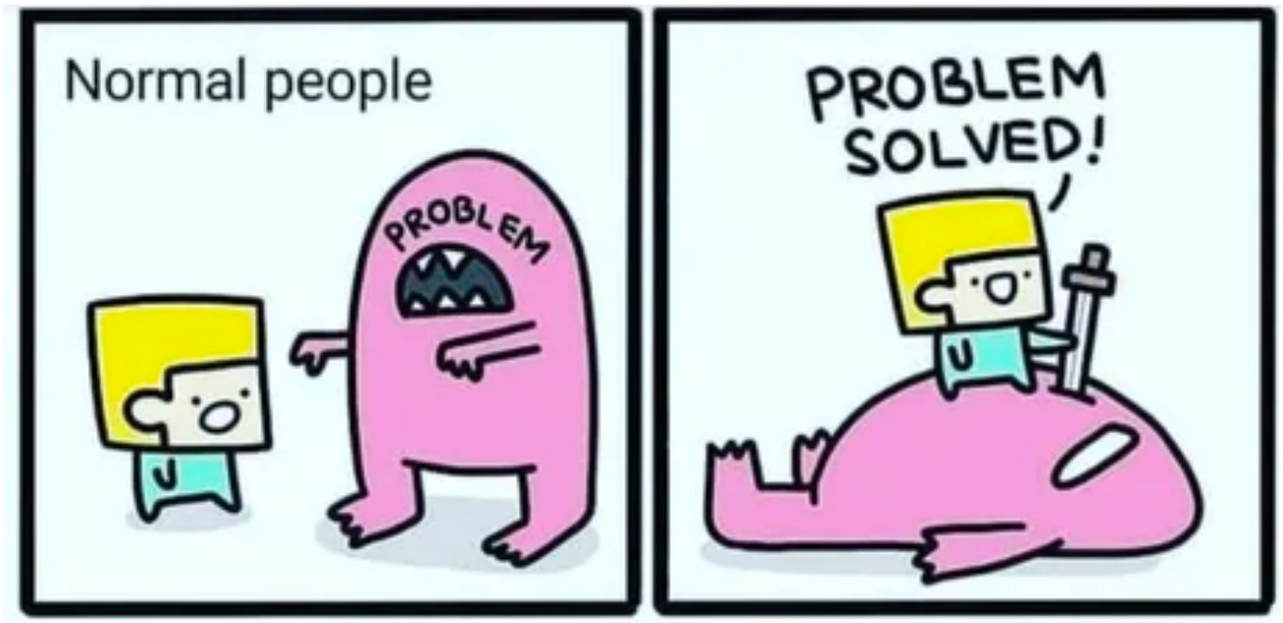
DDD

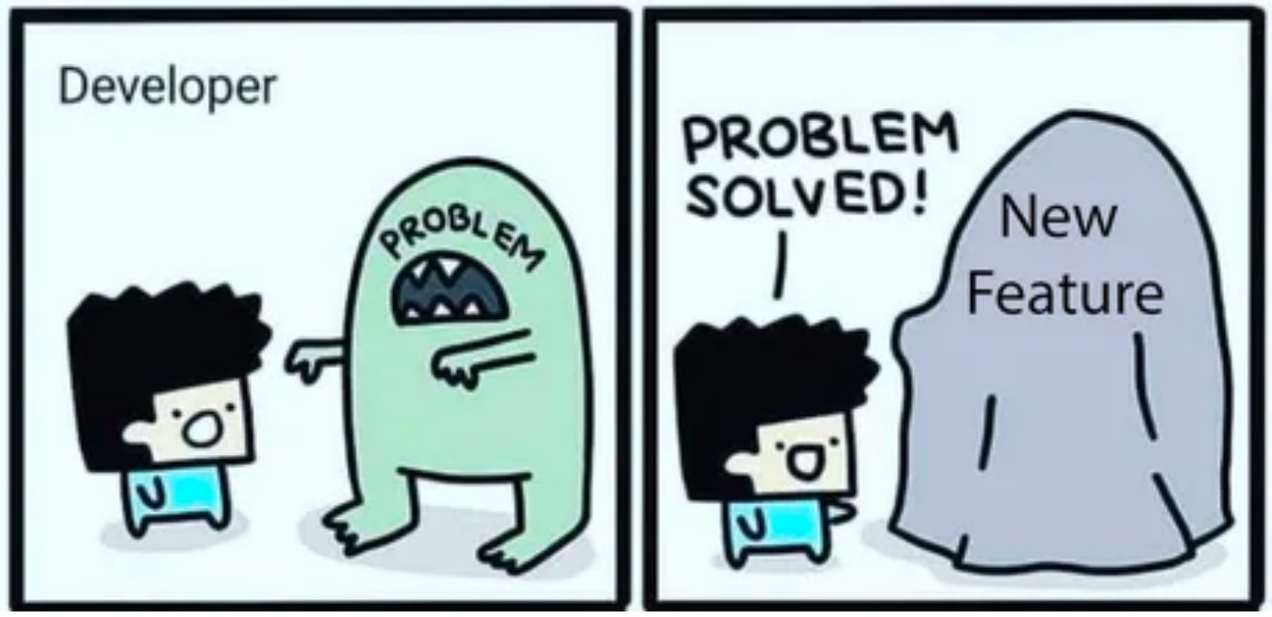
Deadline Driven Development



Developer vs Tester







Every time a developer changing the code !!

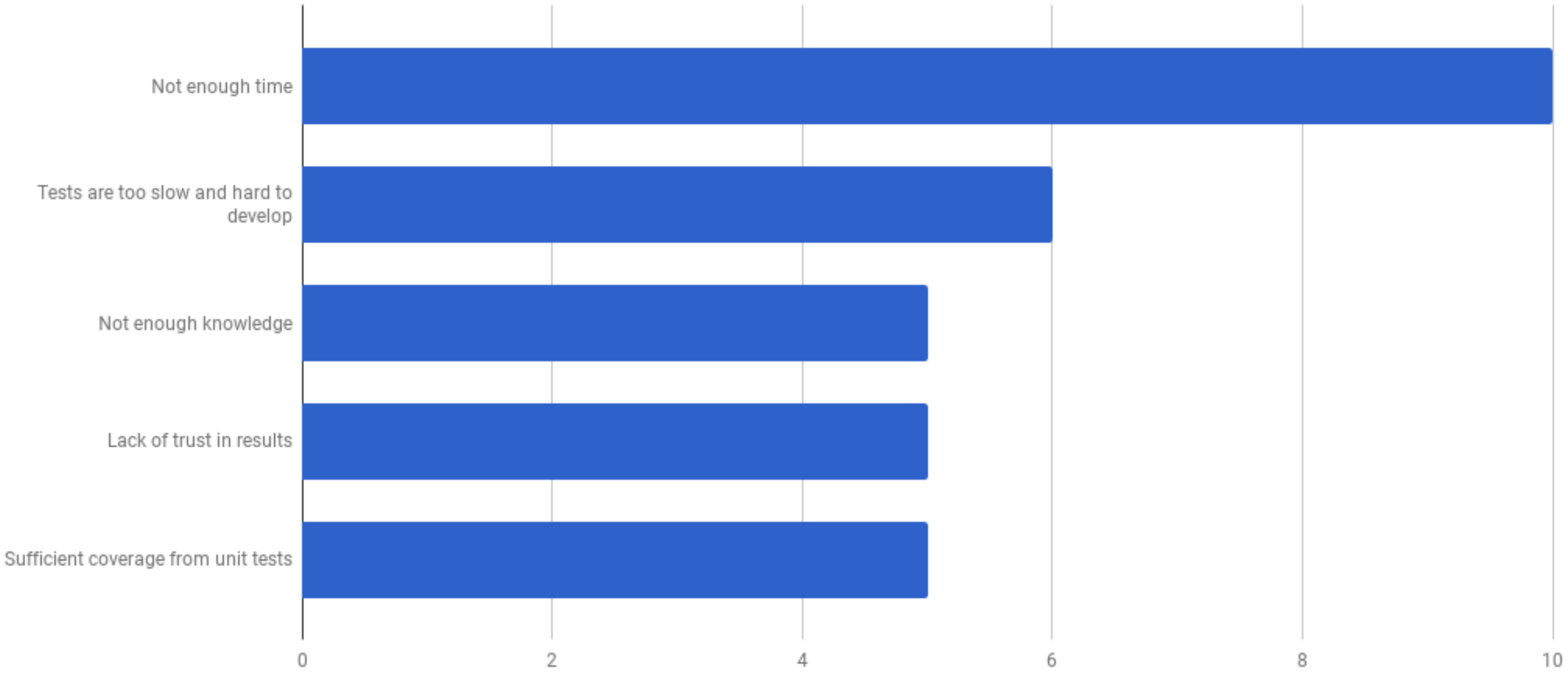
Why we need to test ?

Help you to **catch bugs**
Boosted confidence
Quality code
Enforce **modularity** of your project
Develop features **faster**
Better documentation

But,

It’s take time to learning and practice !!

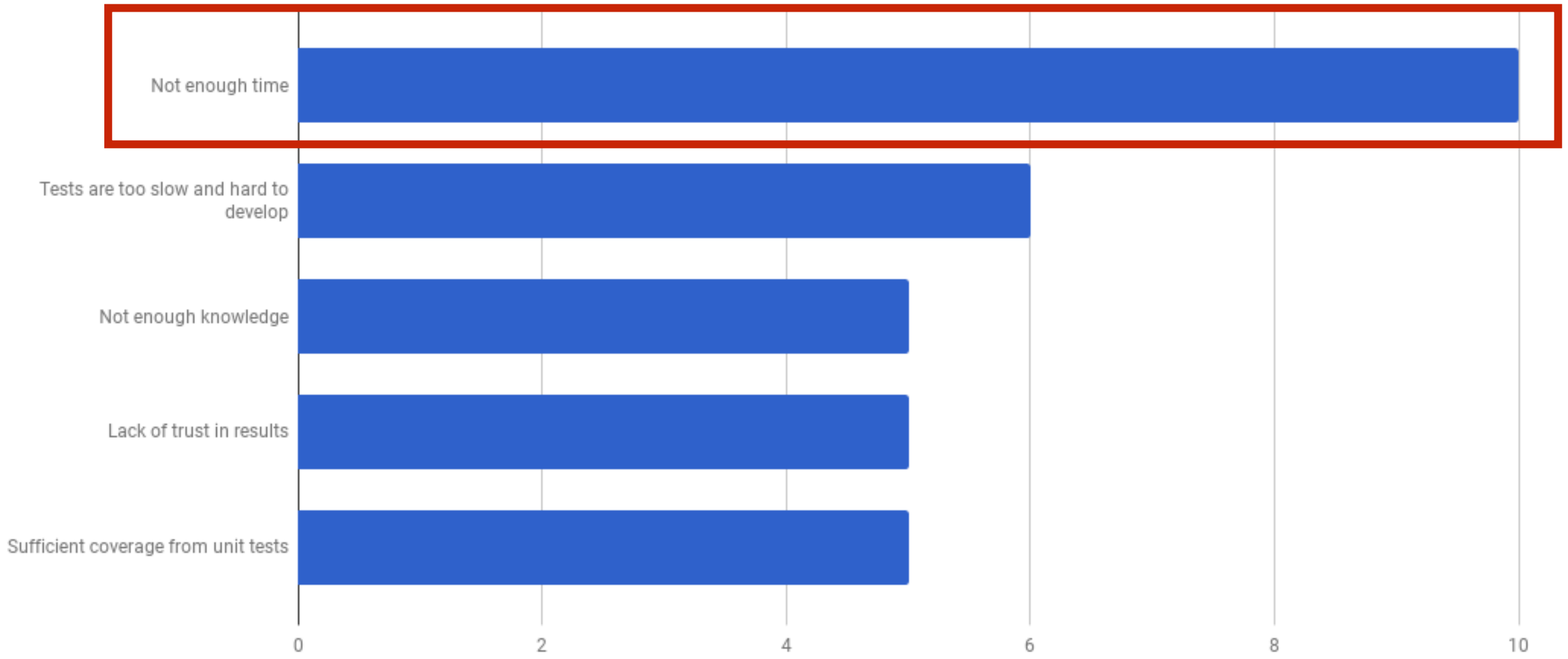
Why not write automated tests ?



https://slack.engineering/android-ui-automation-part-1-building-trust-de3deb1c5995

Why not write automated tests ?

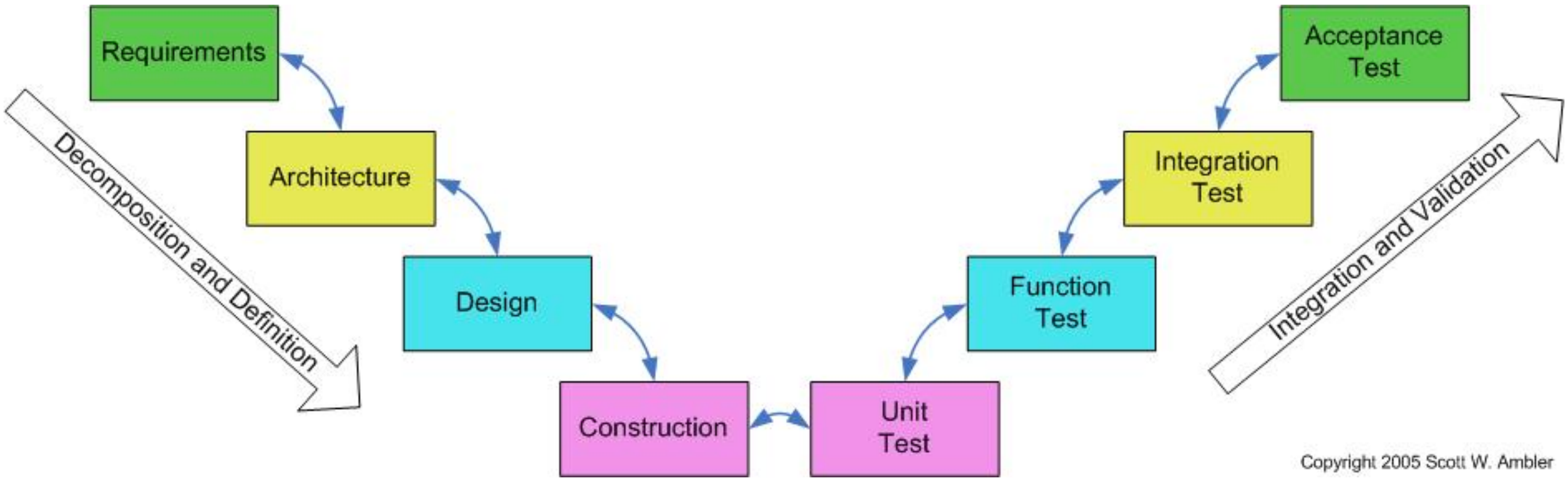
Not enough time !!!



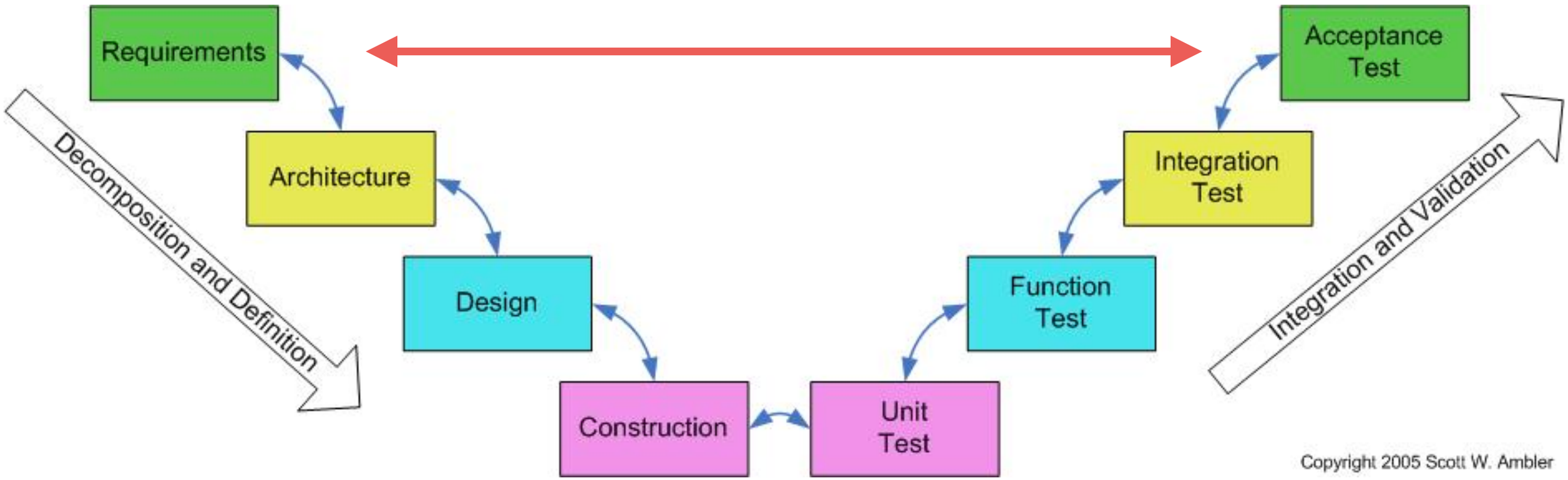
https://slack.engineering/android-ui-automation-part-1-building-trust-de3deb1c5995

What kind of test should we write ?

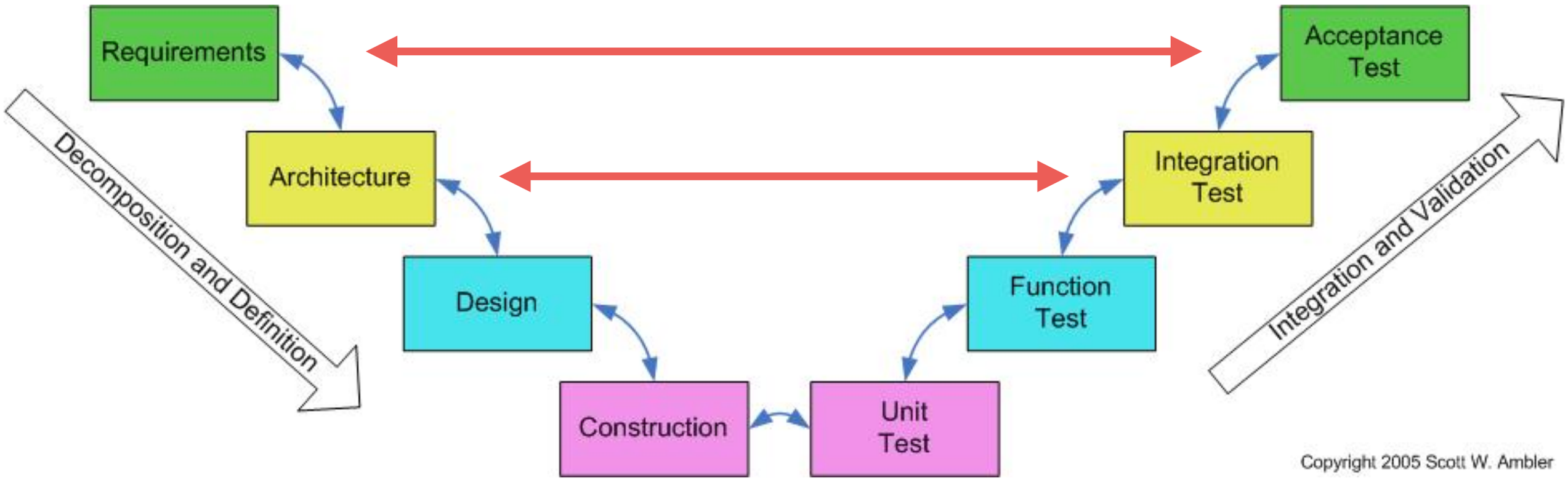
V Model



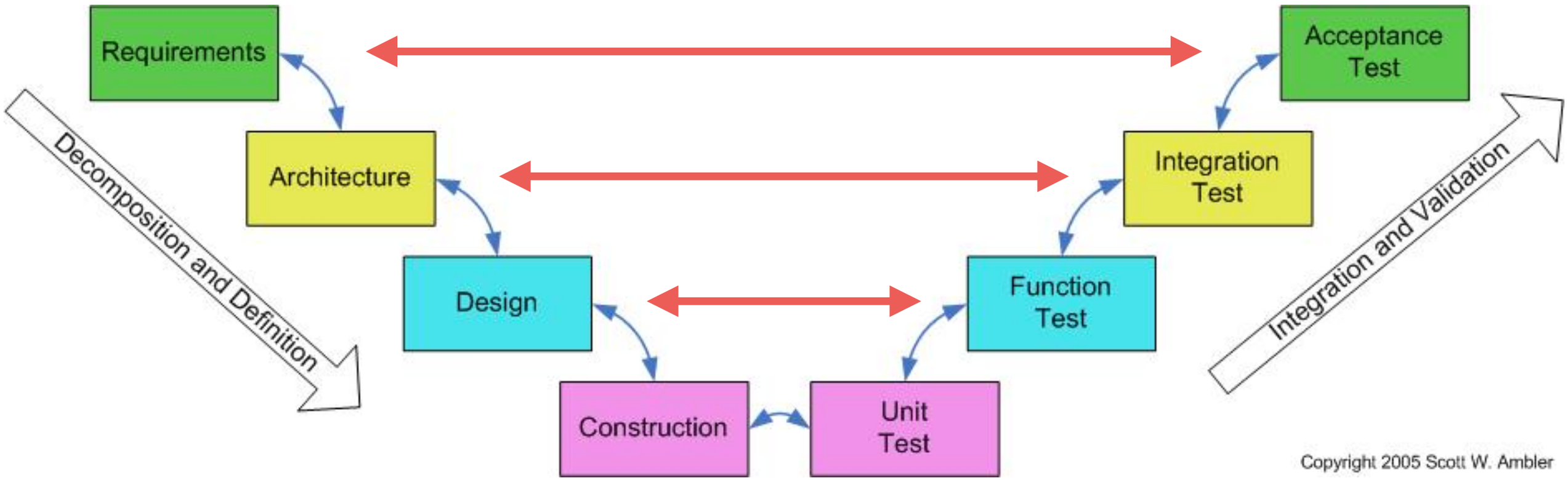
V Model



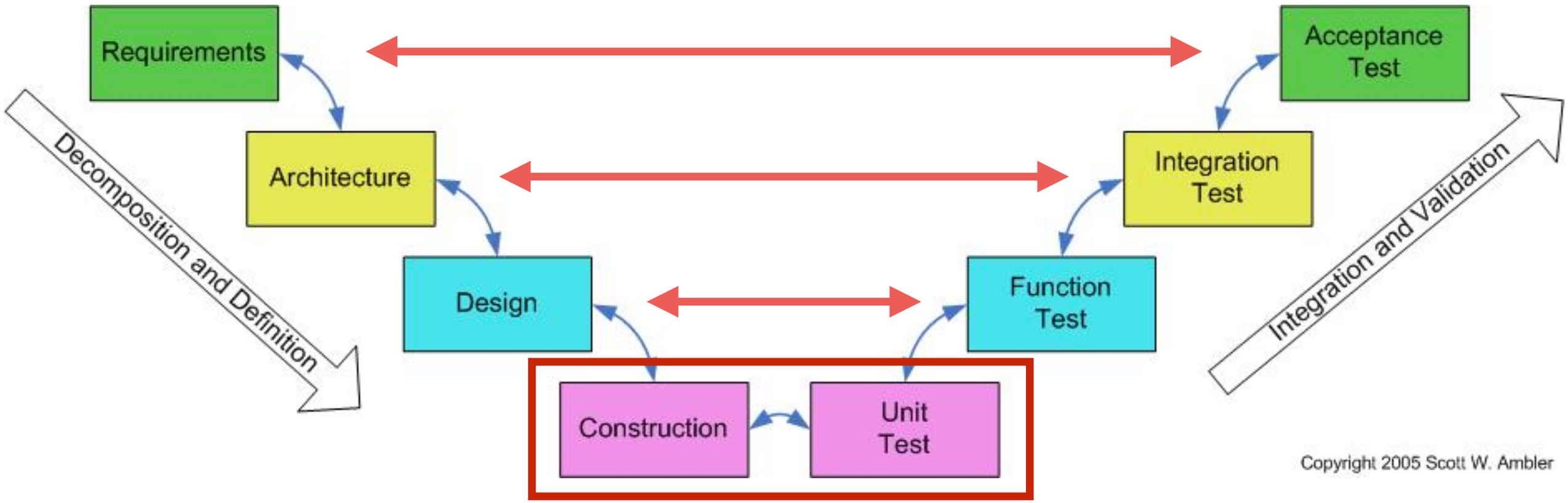
V Model



V Model

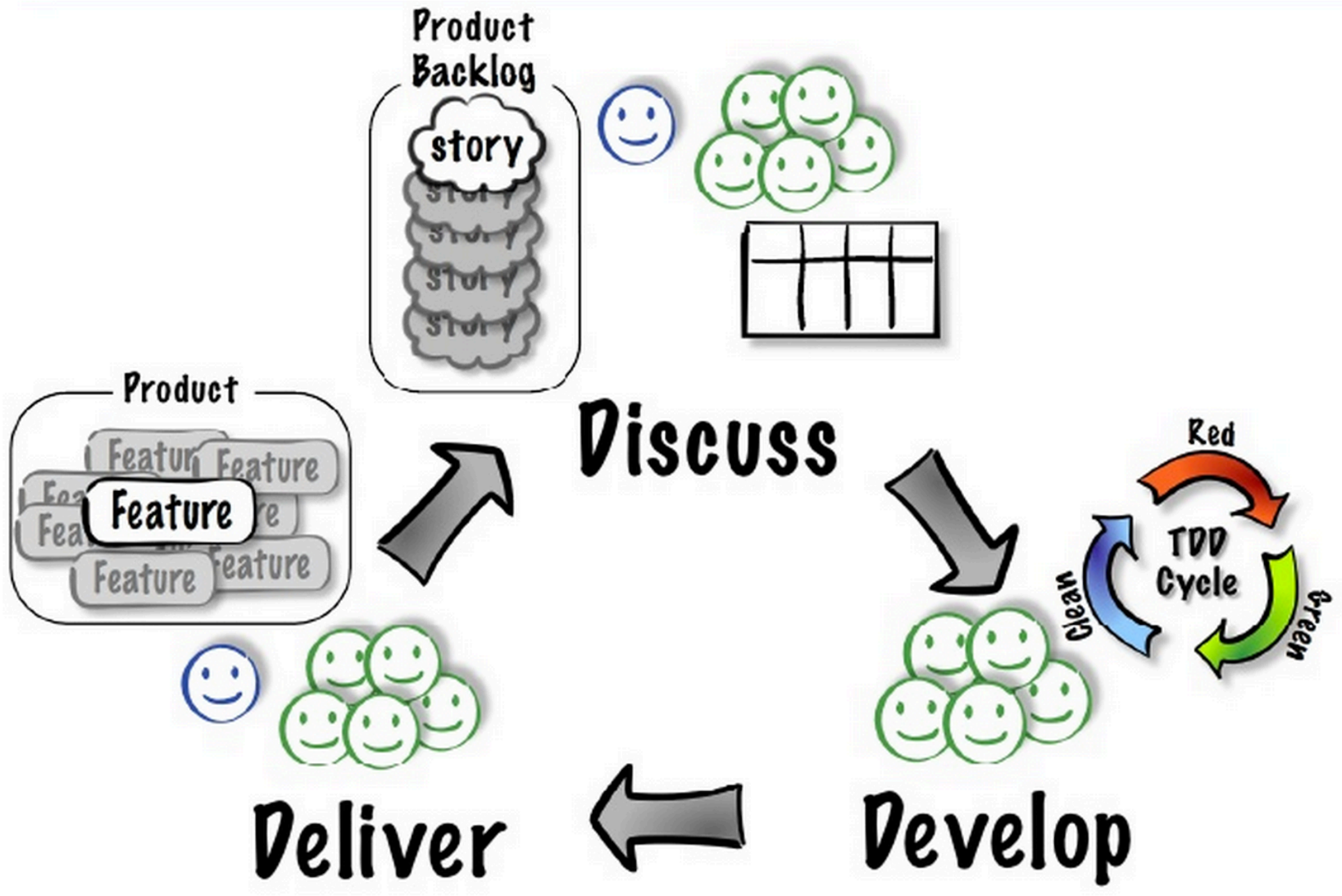


V Model



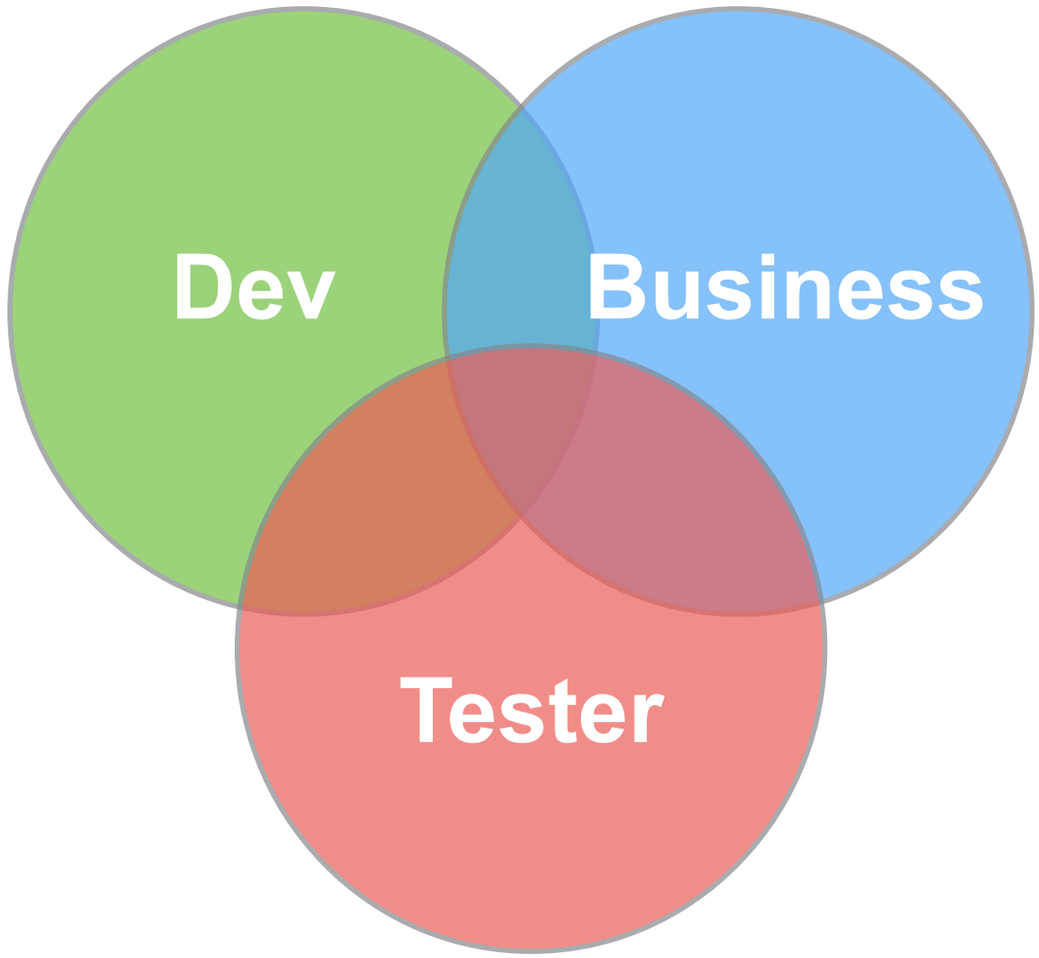
THINK before coding

Acceptance Test-Driven Development



(Model developed with Pekka Klärck, Bas Vodde, and Craig Larman.)

Acceptance Test-Driven Development



Acceptance Tests

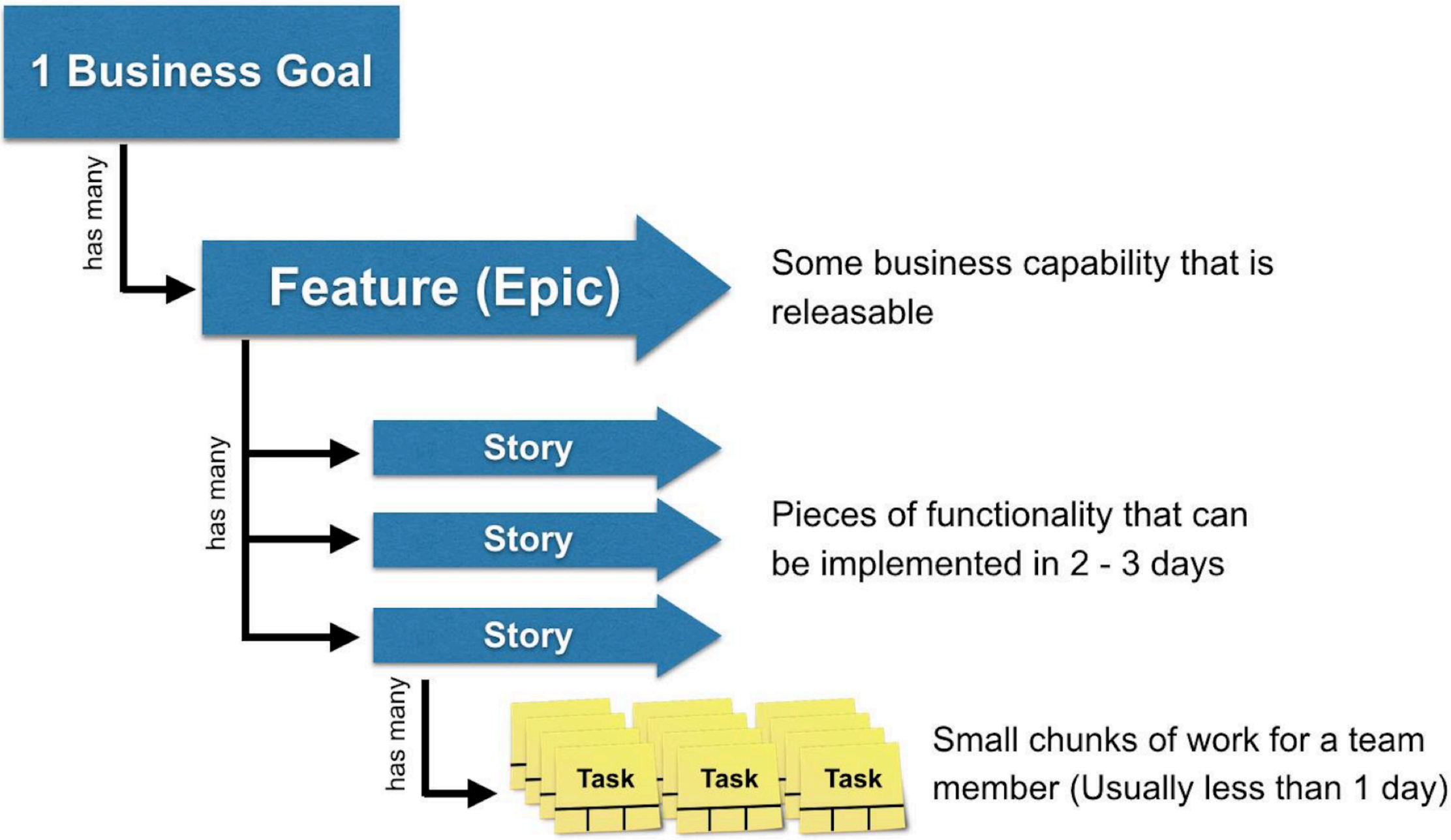
=

Business Criteria

+

Examples (data)

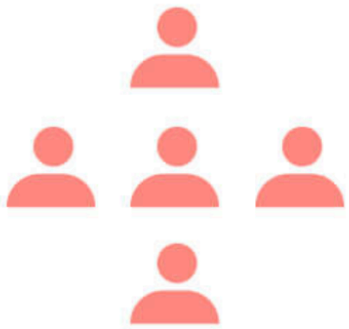
Work break down



Whole team approach

Functional

Common functional expertise



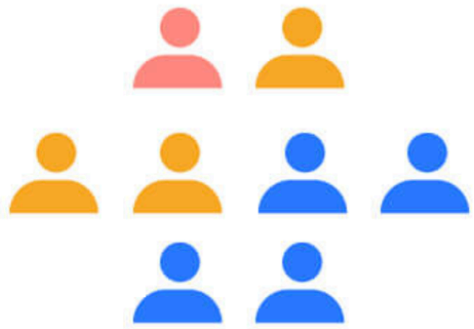
System analysts

Developers

Testers

Cross - Functional

Representatives from the various functions

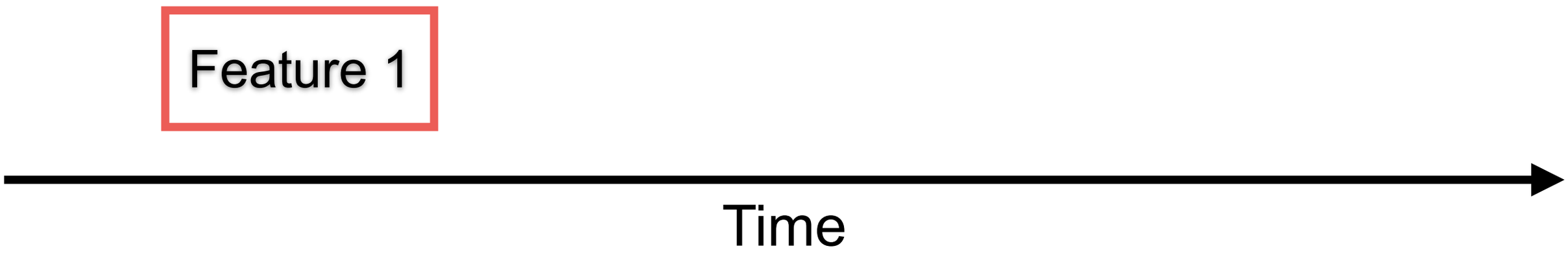


Development Team

Key success factors

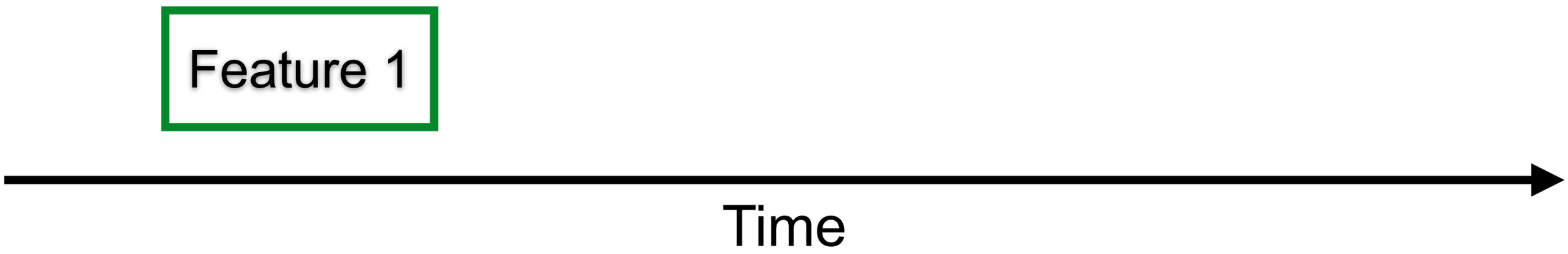
Whole team solve problems
Whole team thinks about **testing**
Whole team **committed to quality**
Everyone **collaborates**

Iterative and incremental process



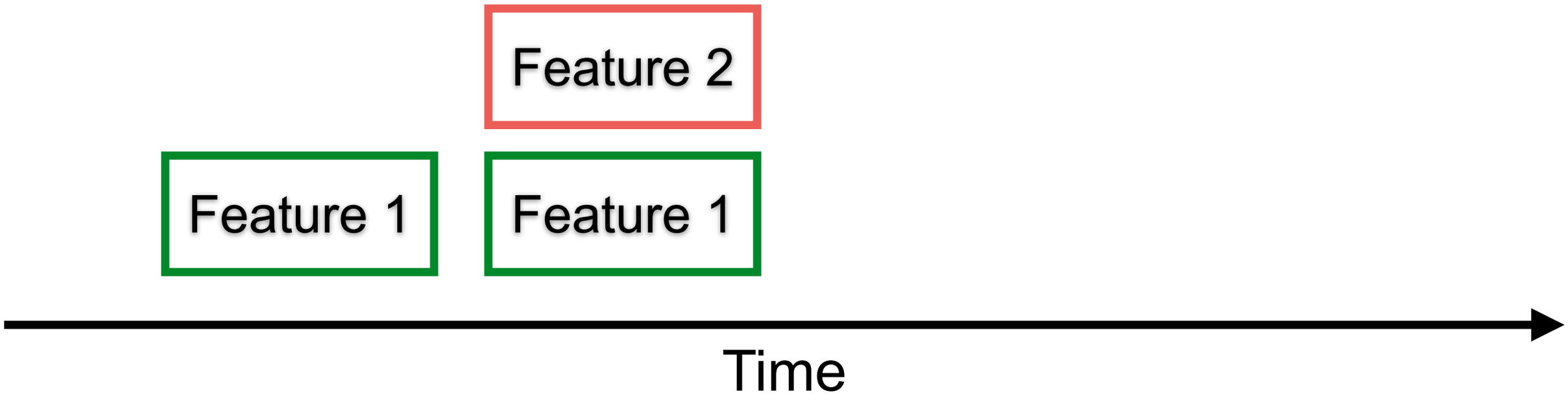
Iterative and incremental process

Done = coded and tested



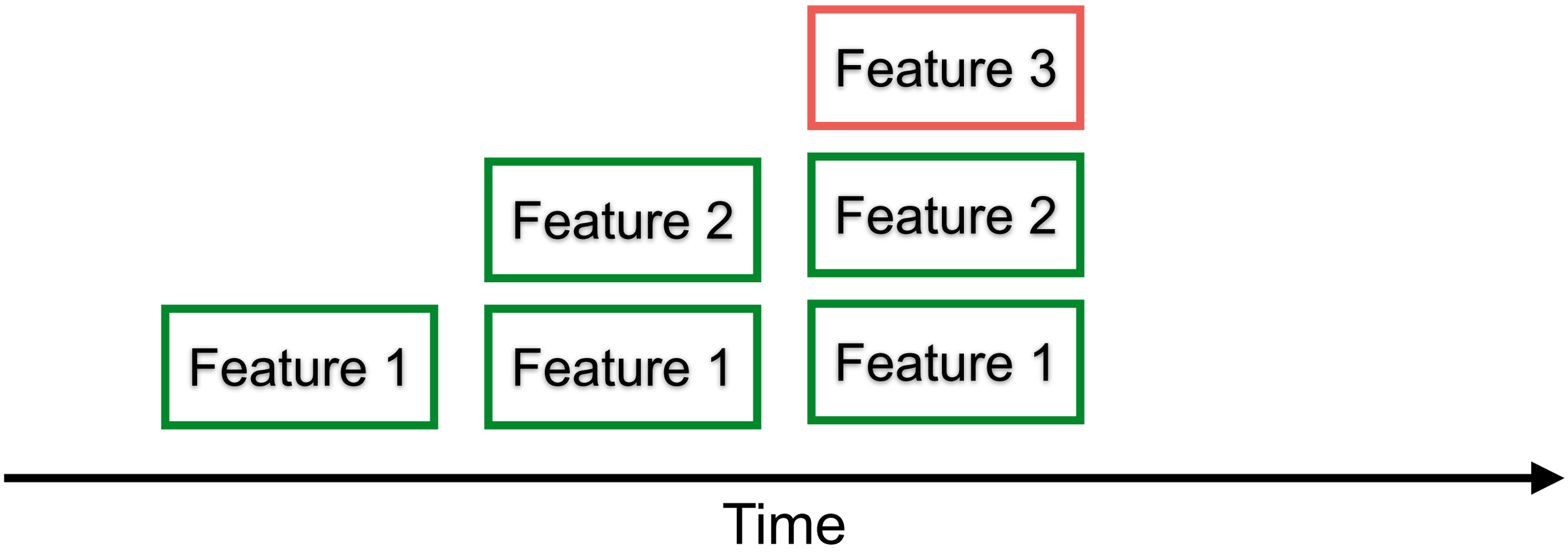
Iterative and incremental process

Done = coded and tested



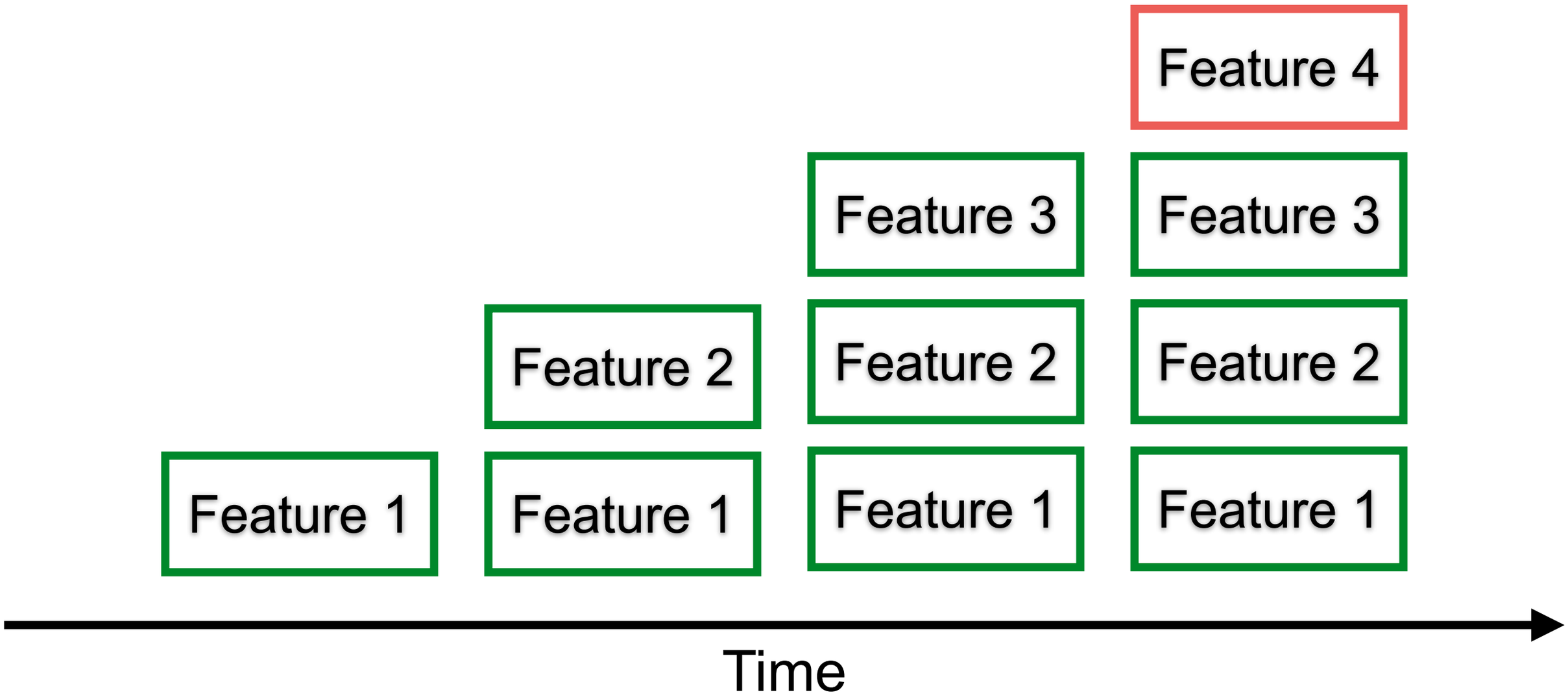
Iterative and incremental process

Done = coded and tested



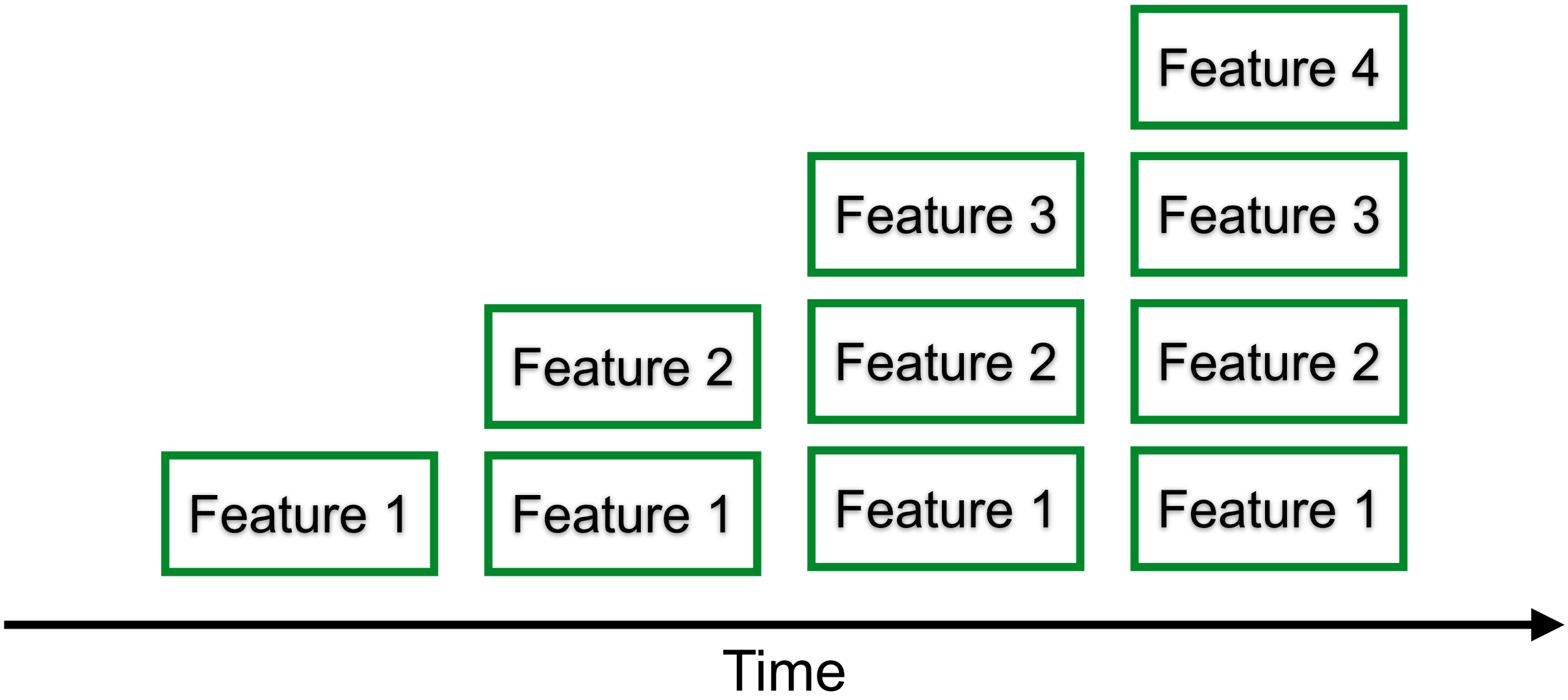
Iterative and incremental process

Done = coded and tested



Iterative and incremental process

Done = coded and tested



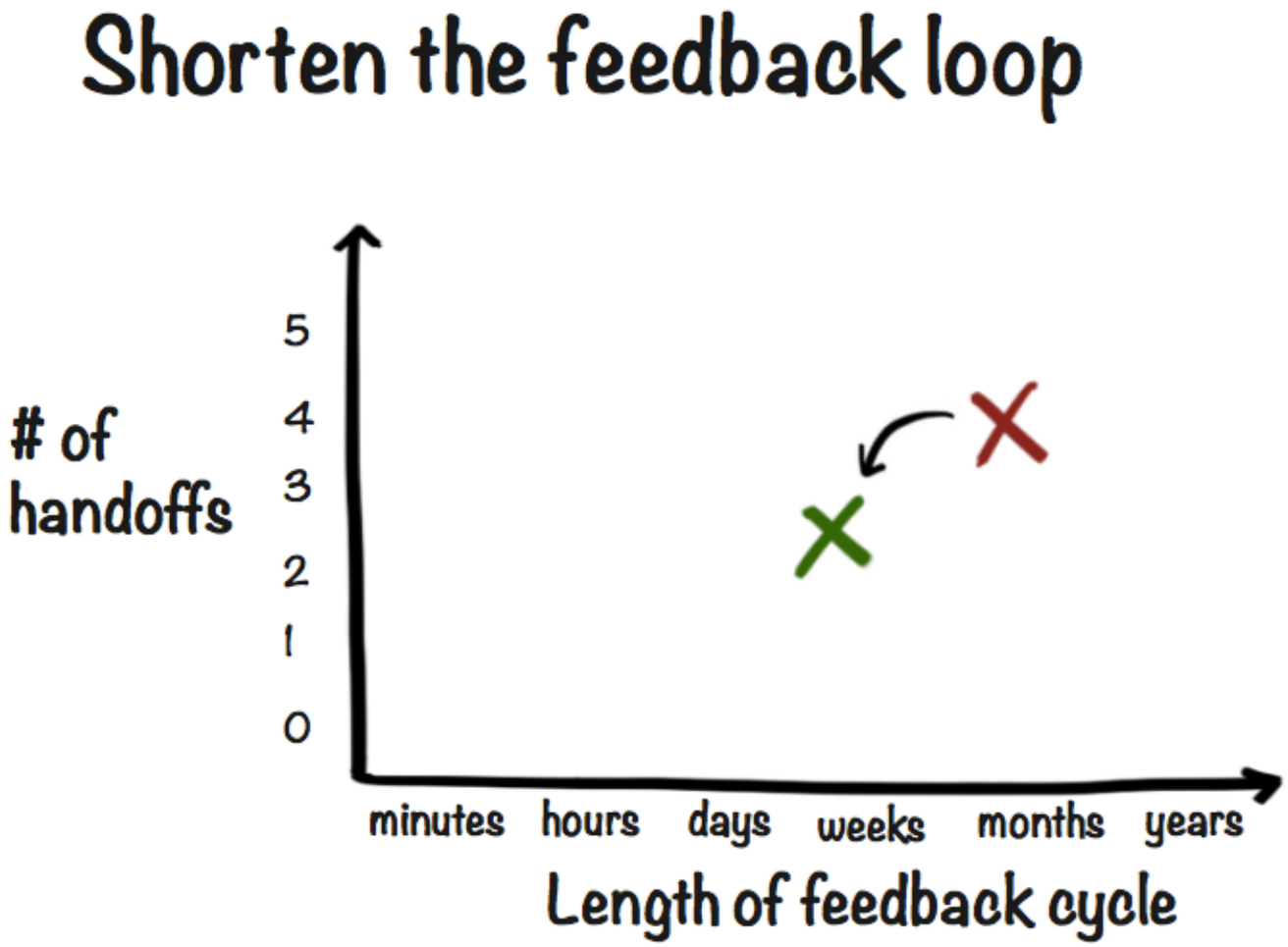
Testing is activity

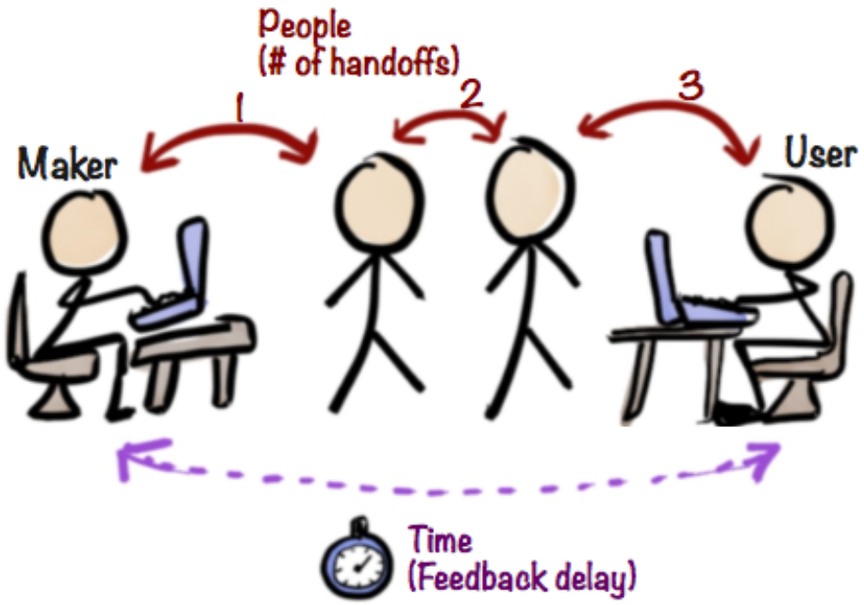
~~Test phase~~

~~Test team~~

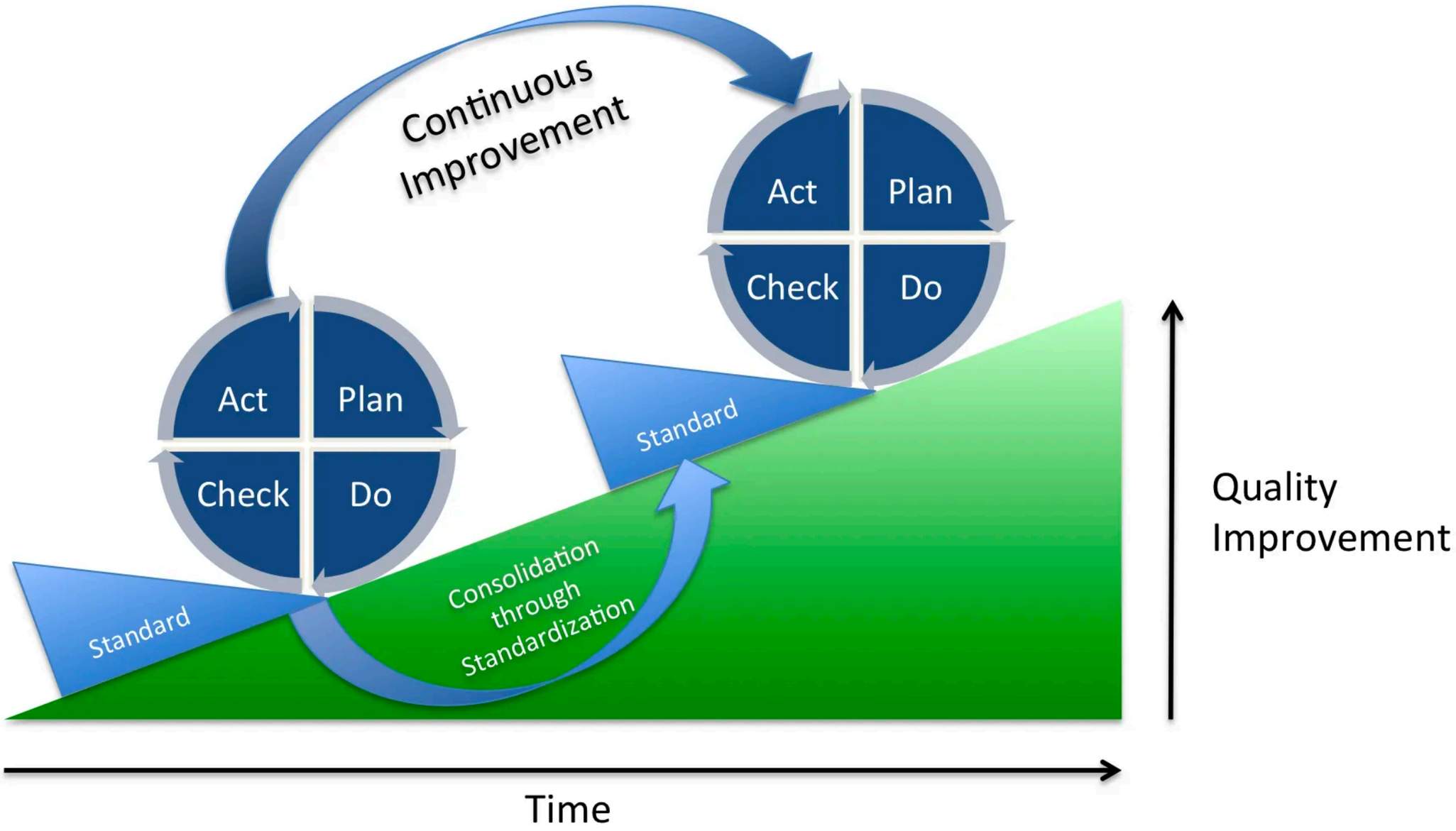
~~Tester role~~

Fast feedback loop

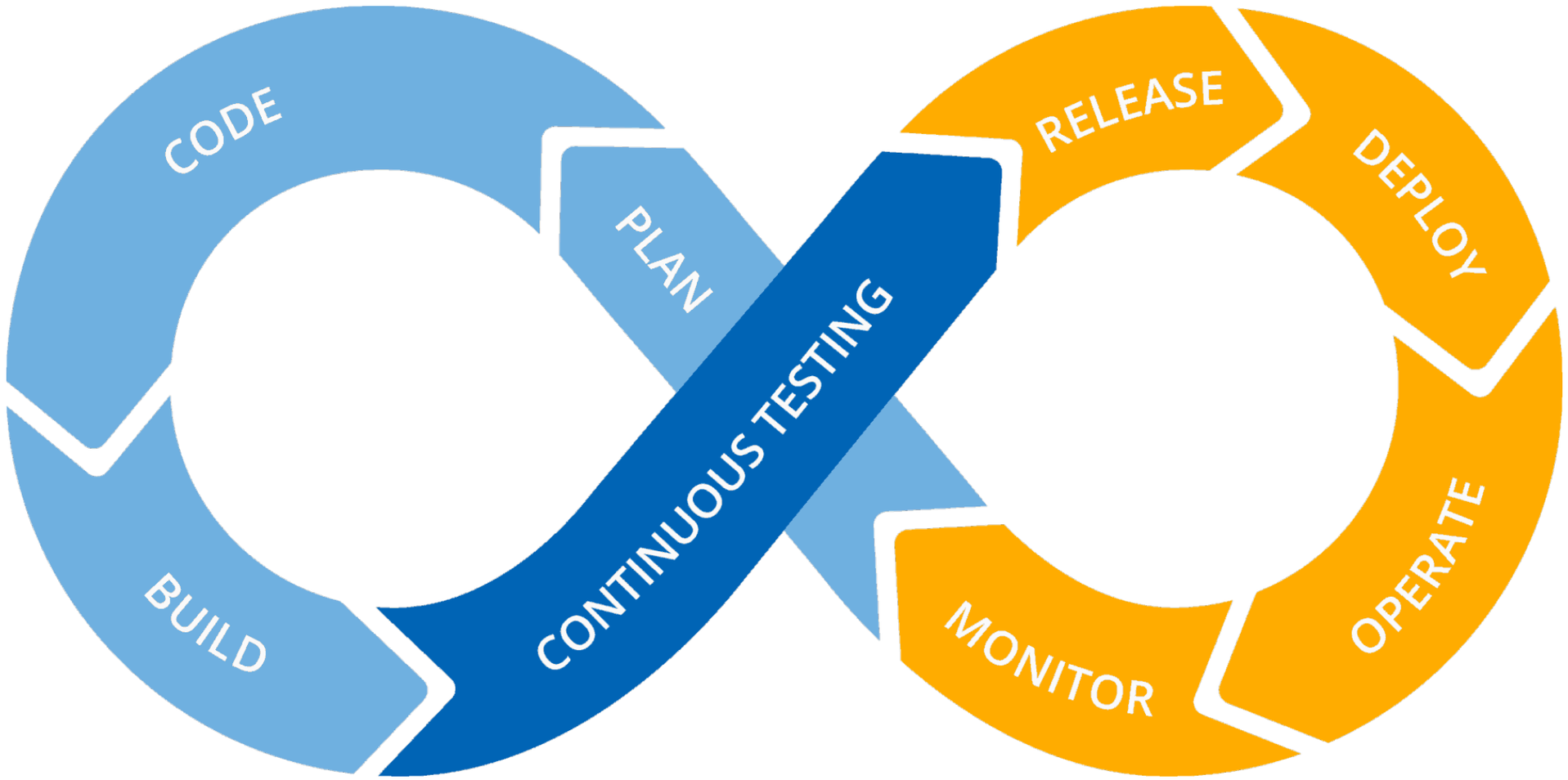




Continuous improvement

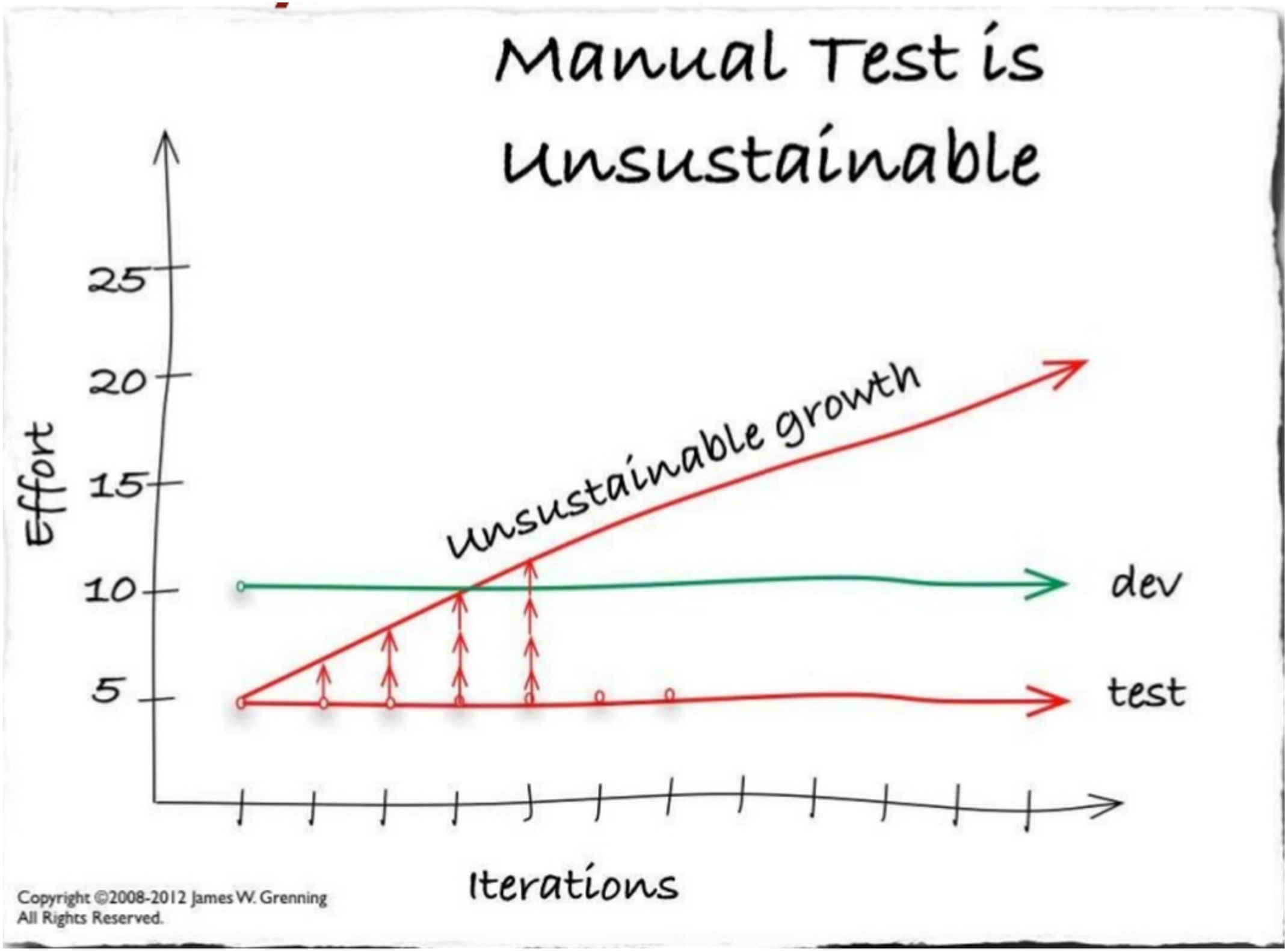


Continuous testing

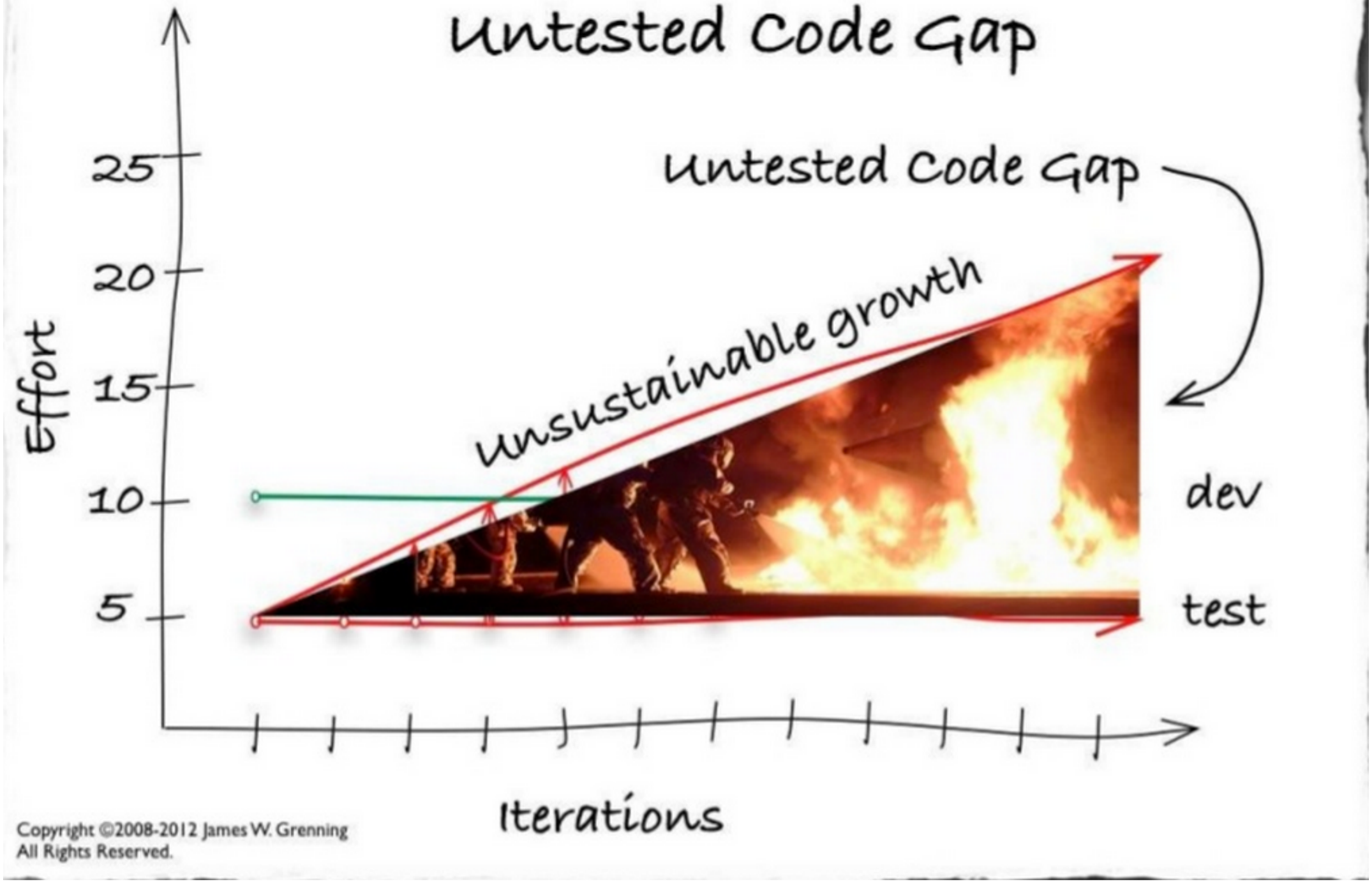


But …

Manual testing ?



Risk Accumulates in the Untested Code Gap



We need automation !!

Why should you automate ?

Manual checking take too long
Manual checks are error prone
Free people to do their best work
Provides living document
Repeatable
Save time

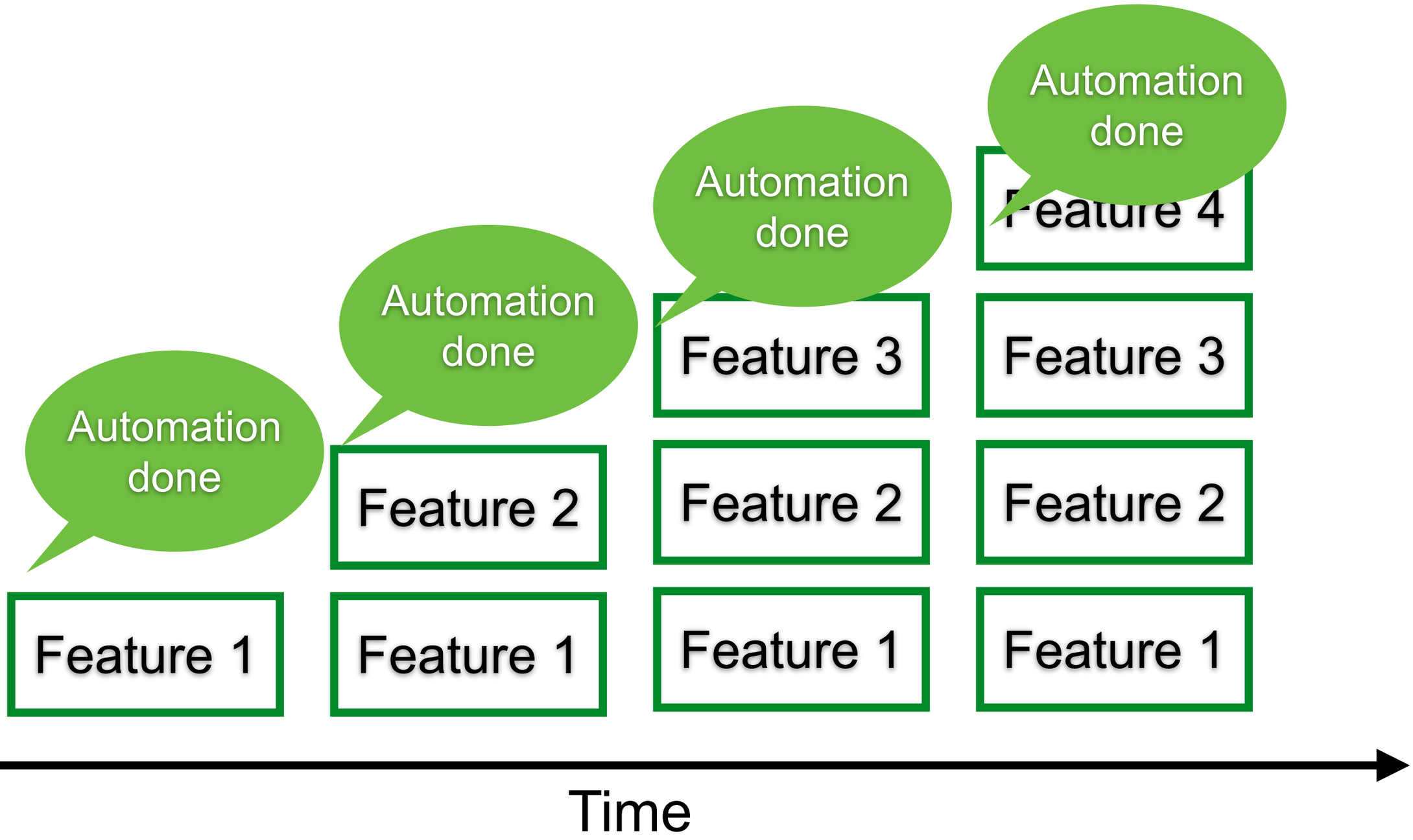
Workshop

Why aren’t you automate ?
Share obstacles in your group

https://bit.ly/2XSPmAH

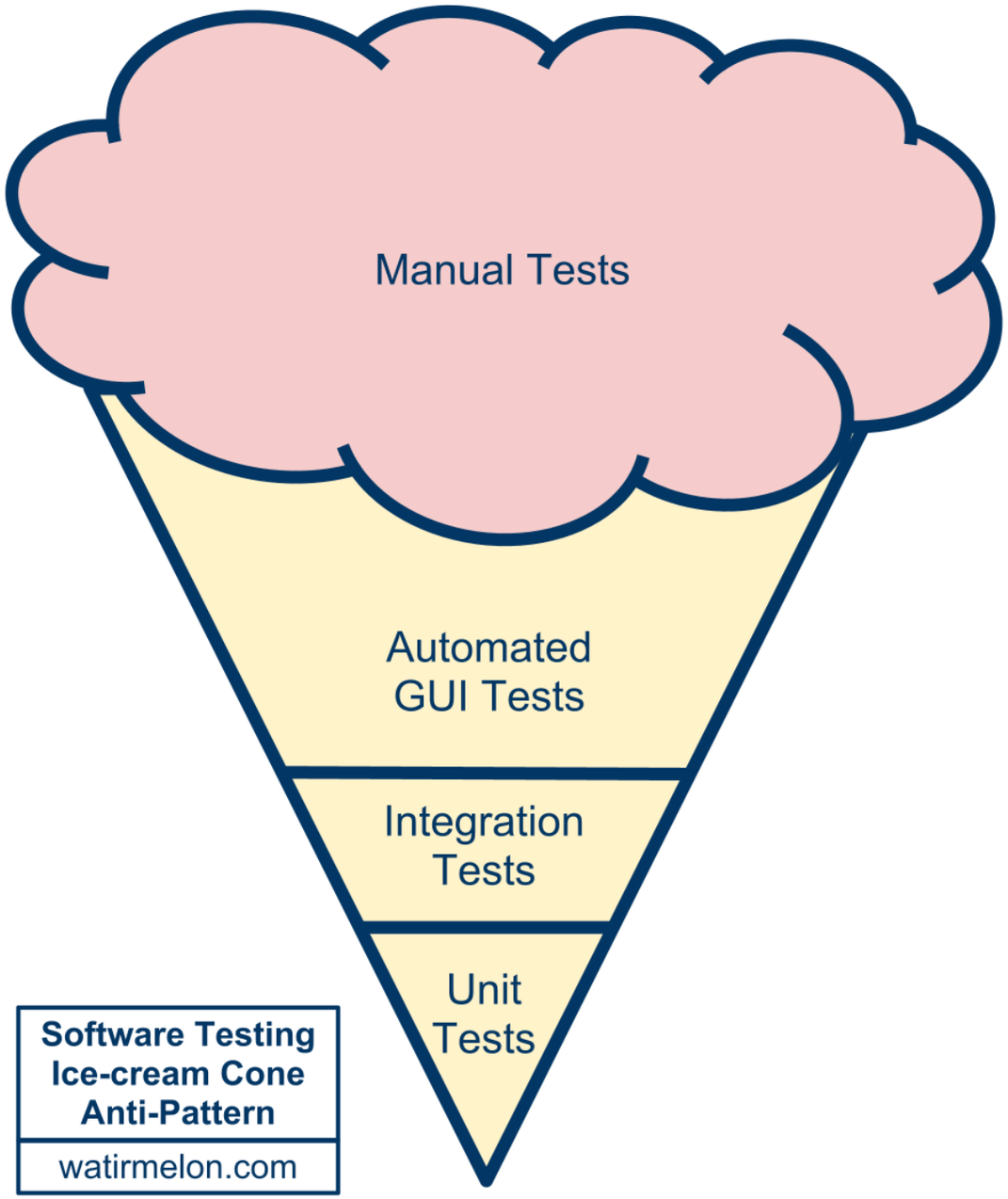
Iterative and incremental process

Done = coded and tested

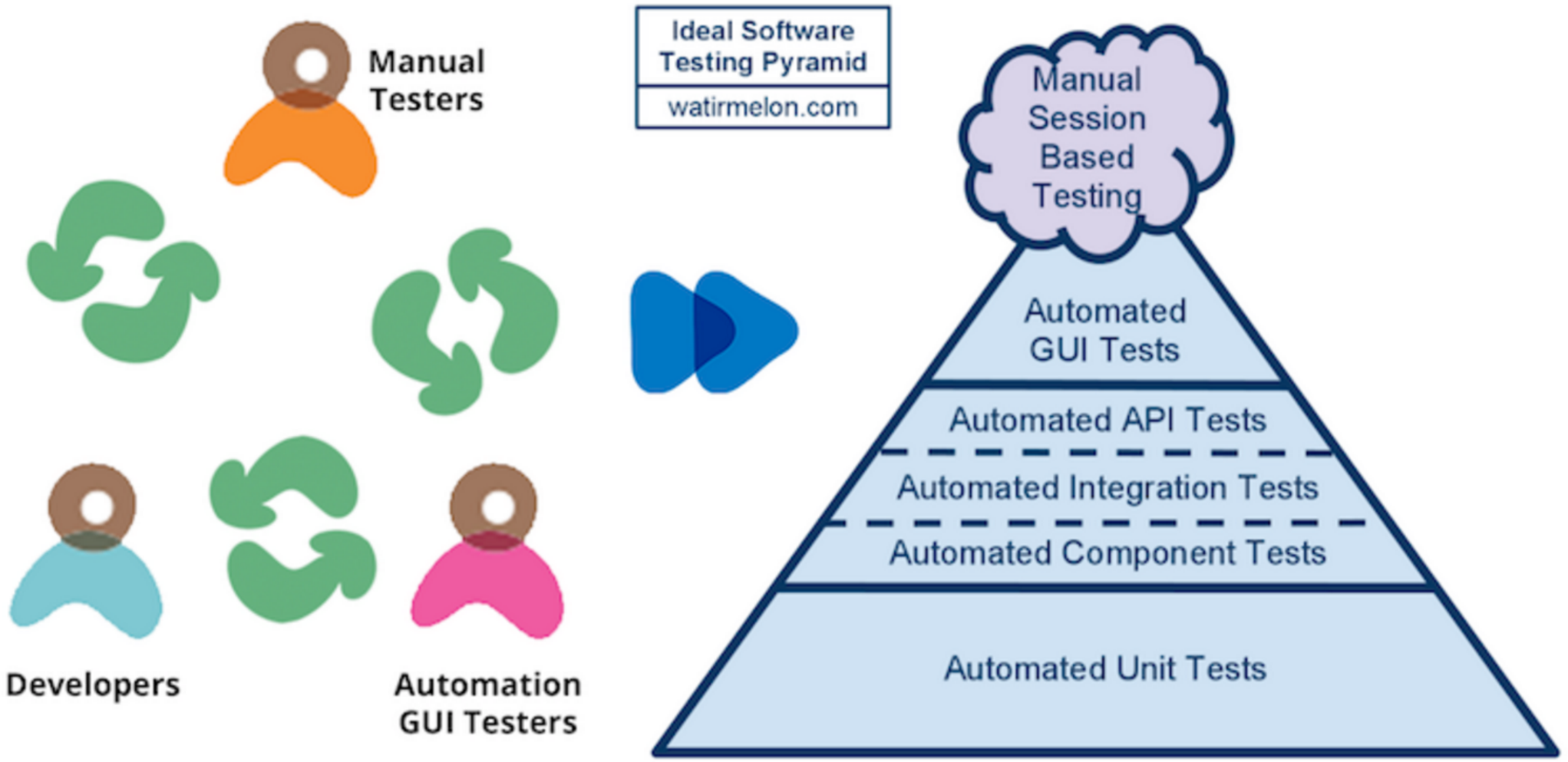


Testing pyramid

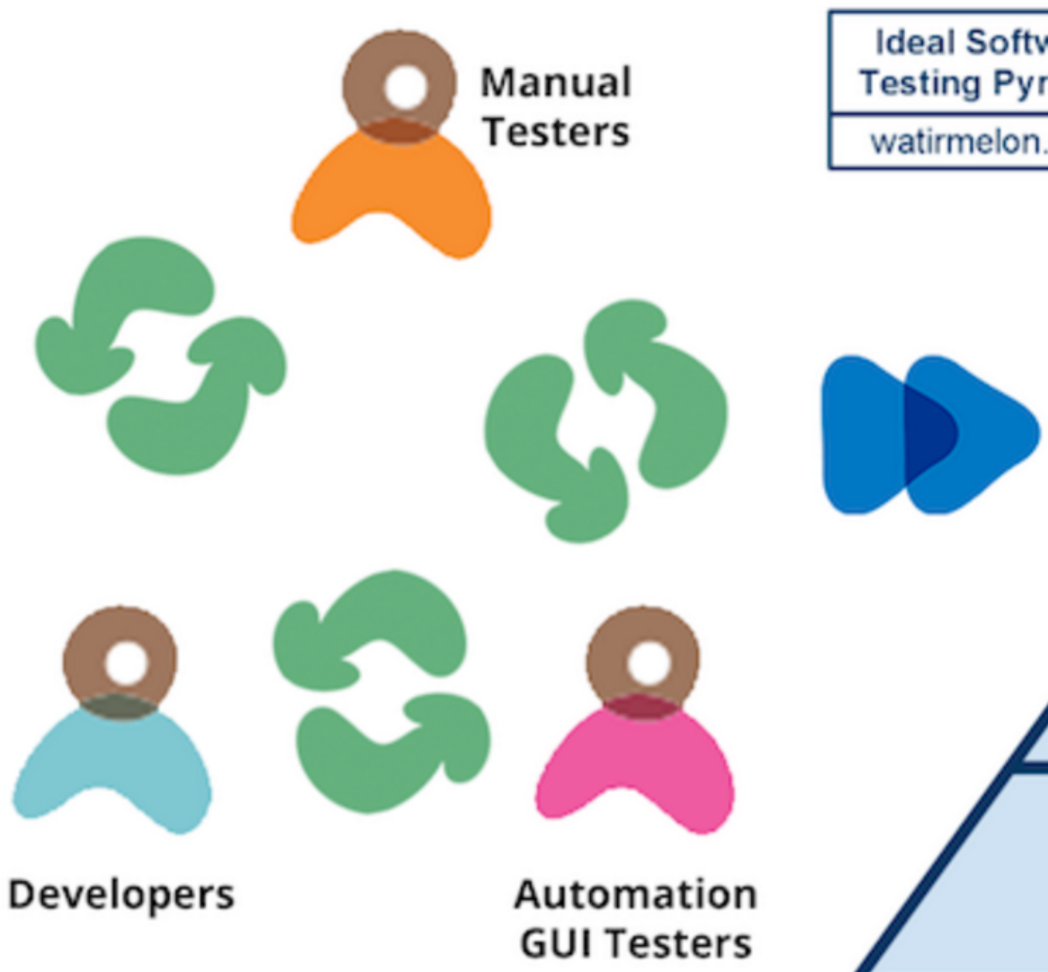
Ice-cream testing

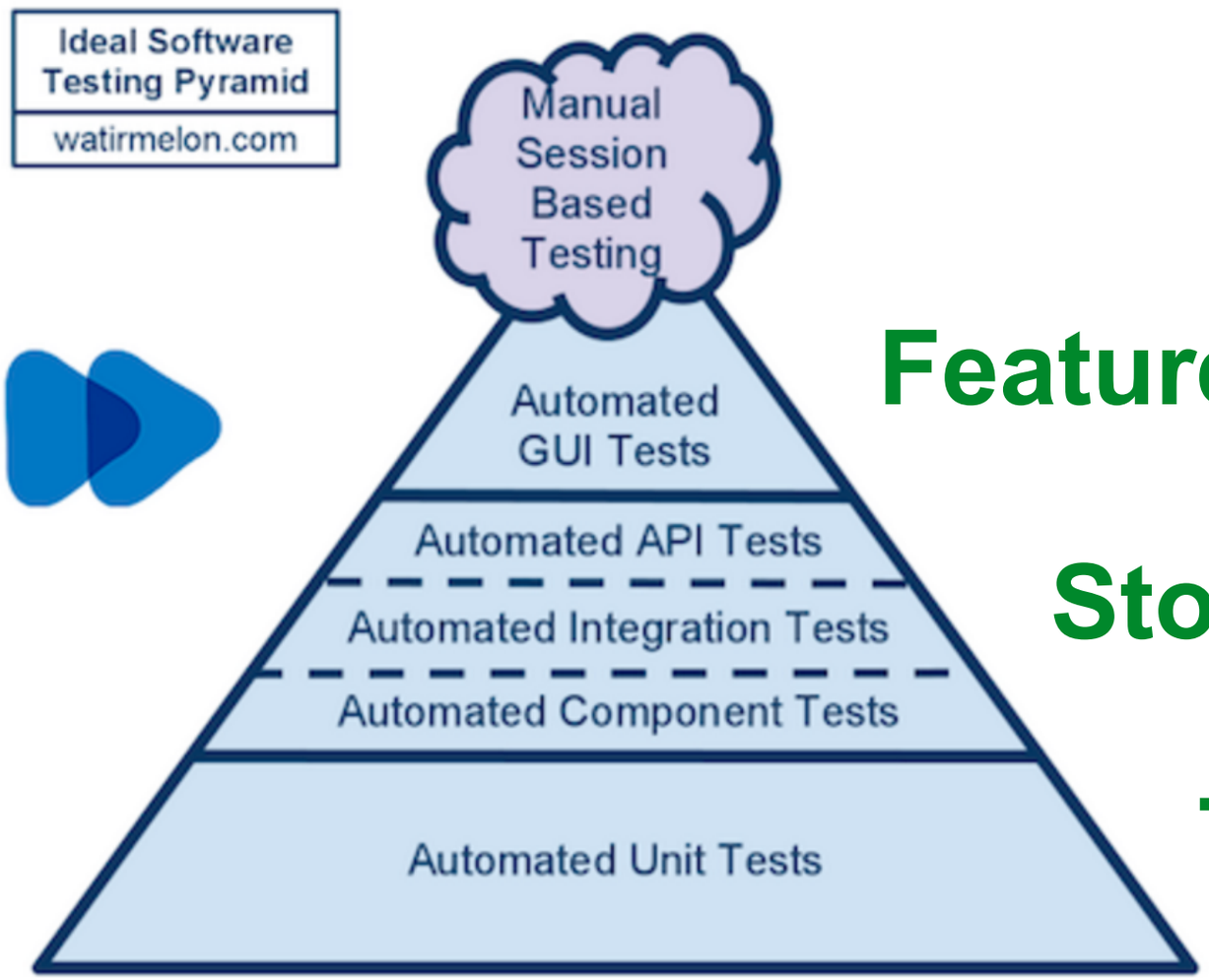


Testing Pyramid

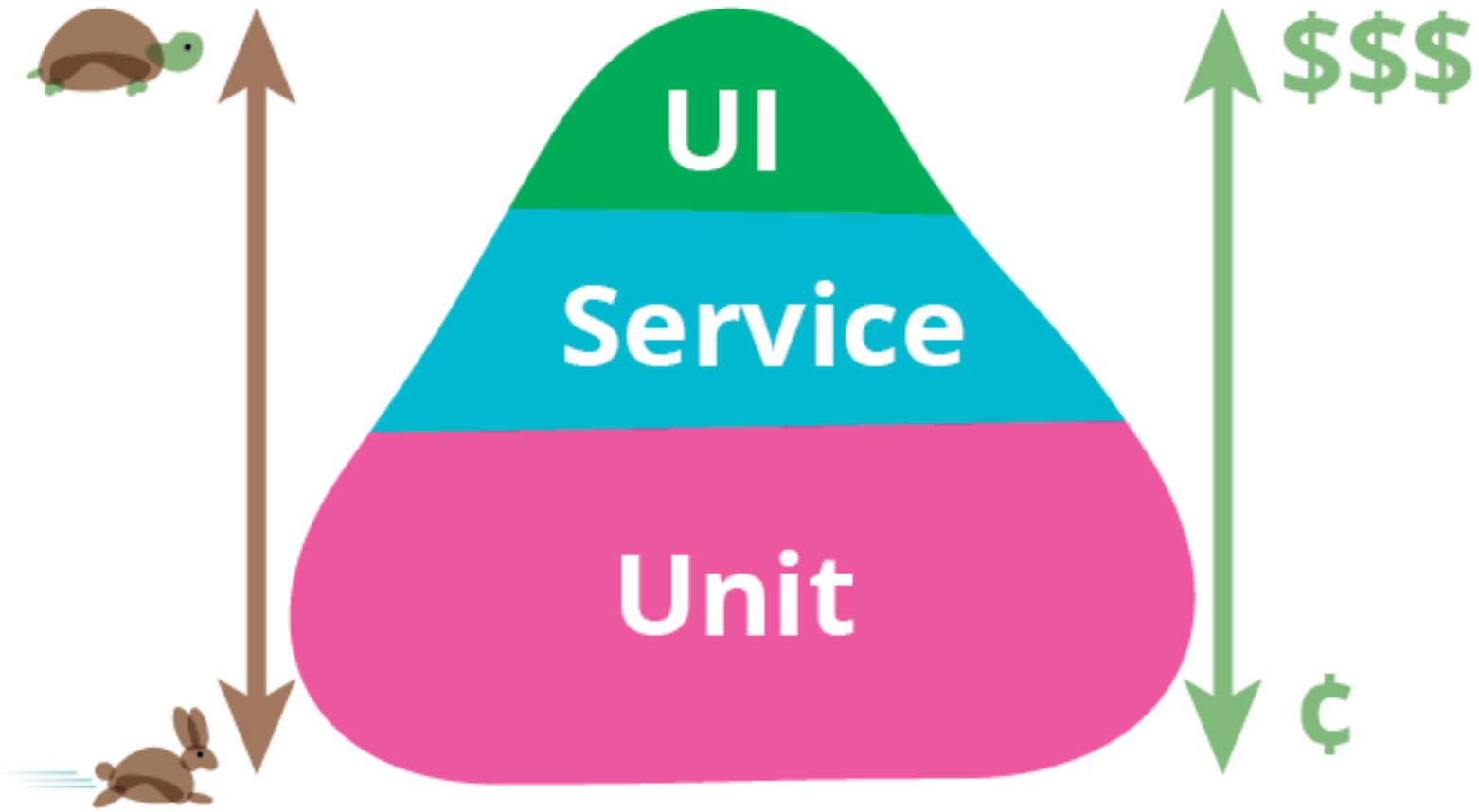


Testing Pyramid

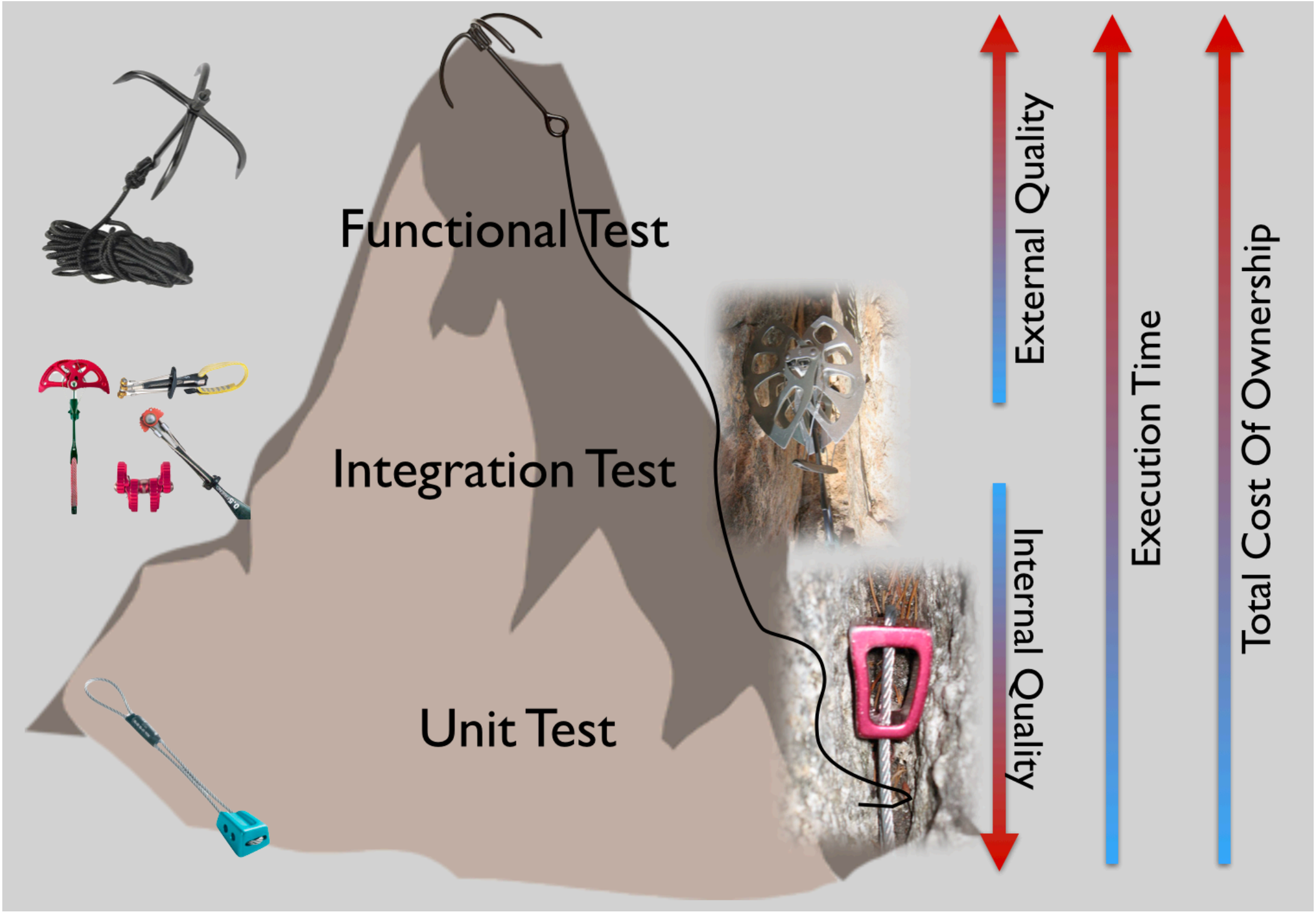




Testing Pyramid

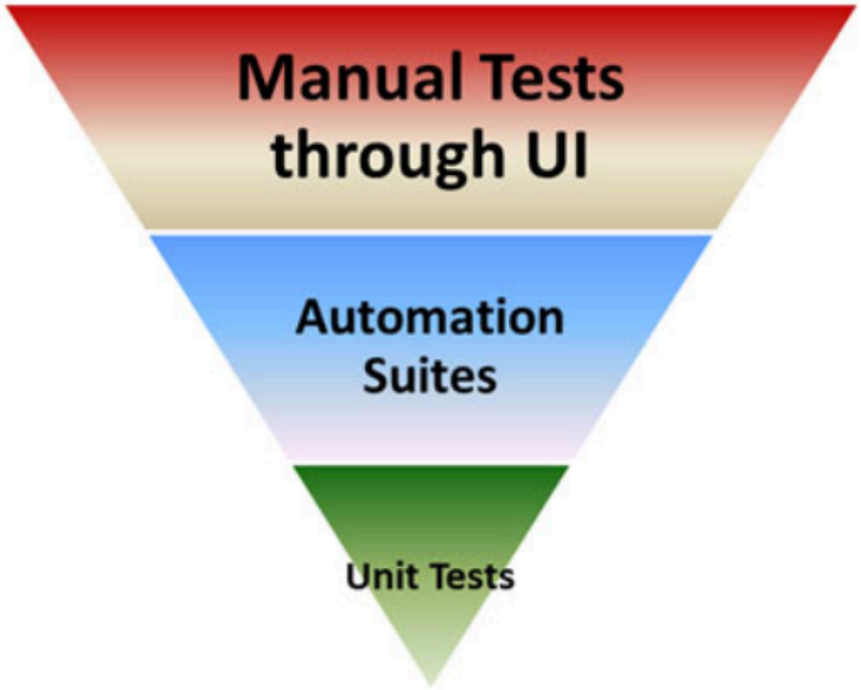


https://martinfowler.com/bliki/TestPyramid.html

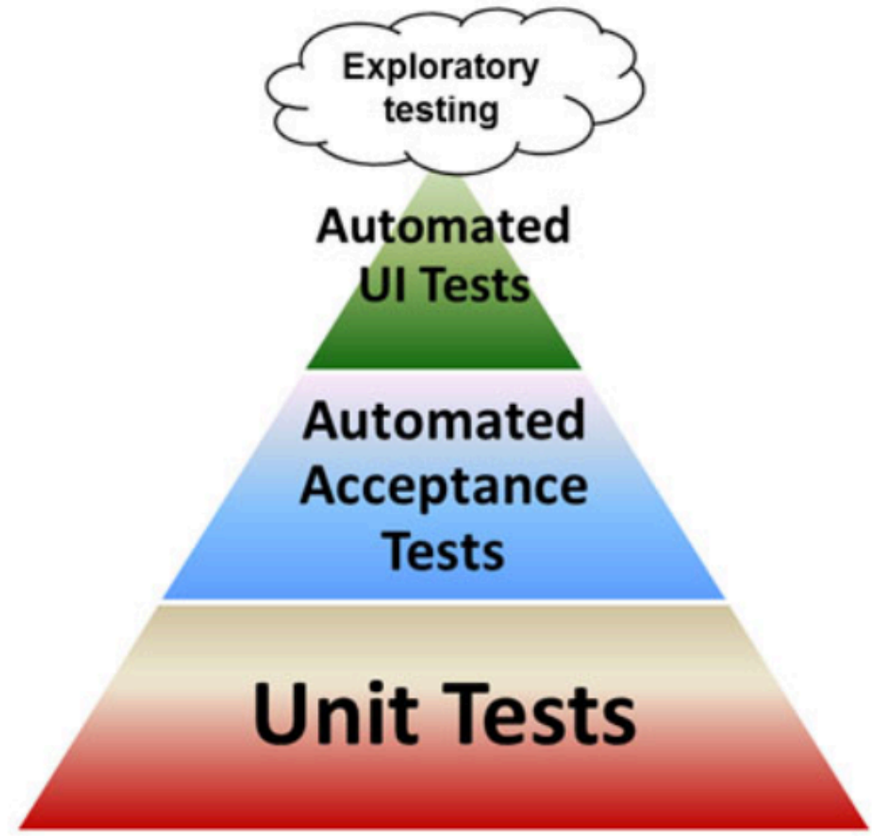


https://less.works/less/technical-excellence/unit-testing.html

Find vs Prevent



Traditional
(find bugs)



Agile
(prevent bugs)

https://martinfowler.com/bliki/TestPyramid.html

Mind-set switch

Instead of

We are here to find bug

We are here to ensure requirement are met

We are here to break the software

Mind-set switch

Instead of

We are here to find bug

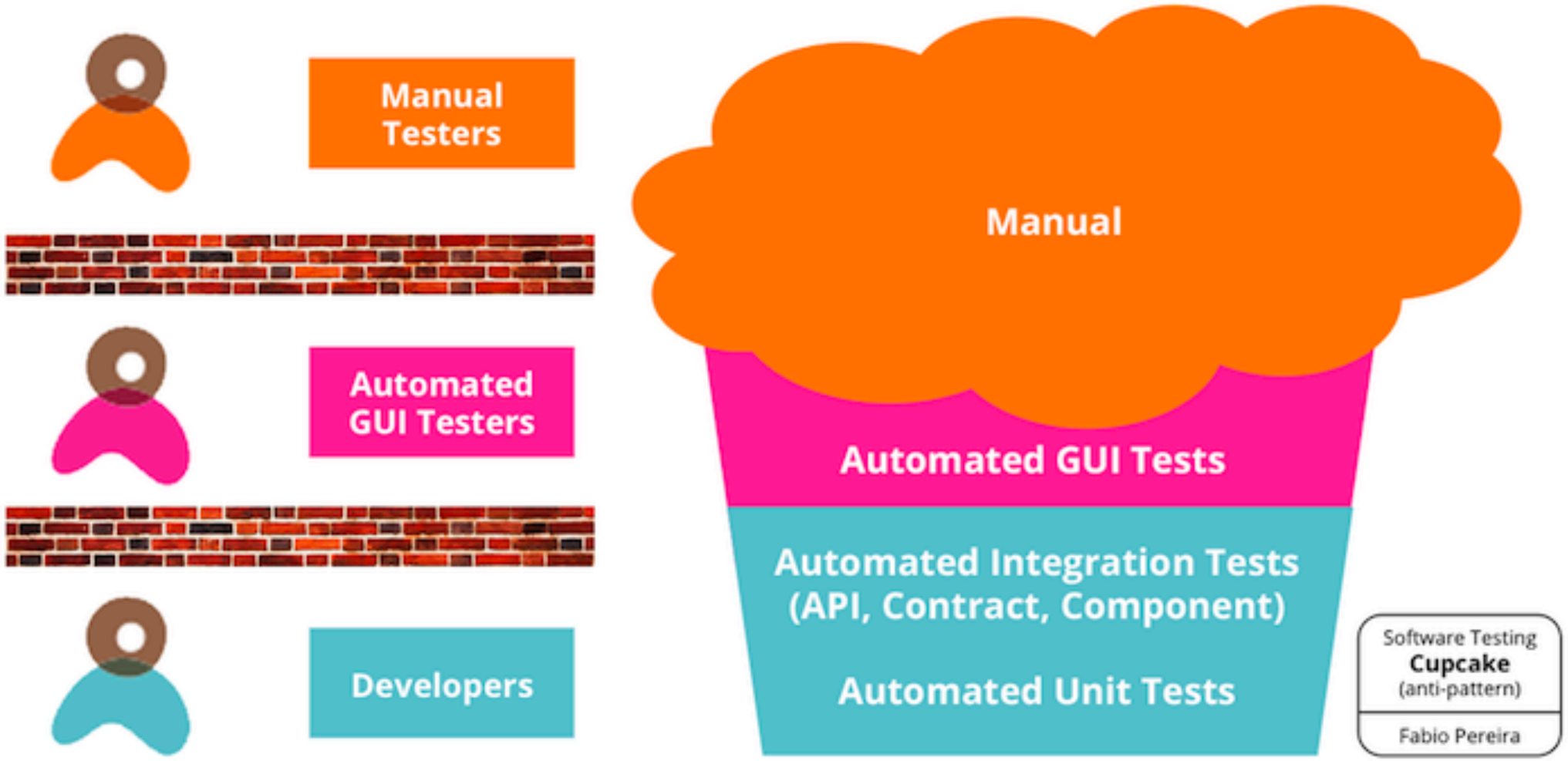
We are here to ensure requirement are met

We are here to break the software

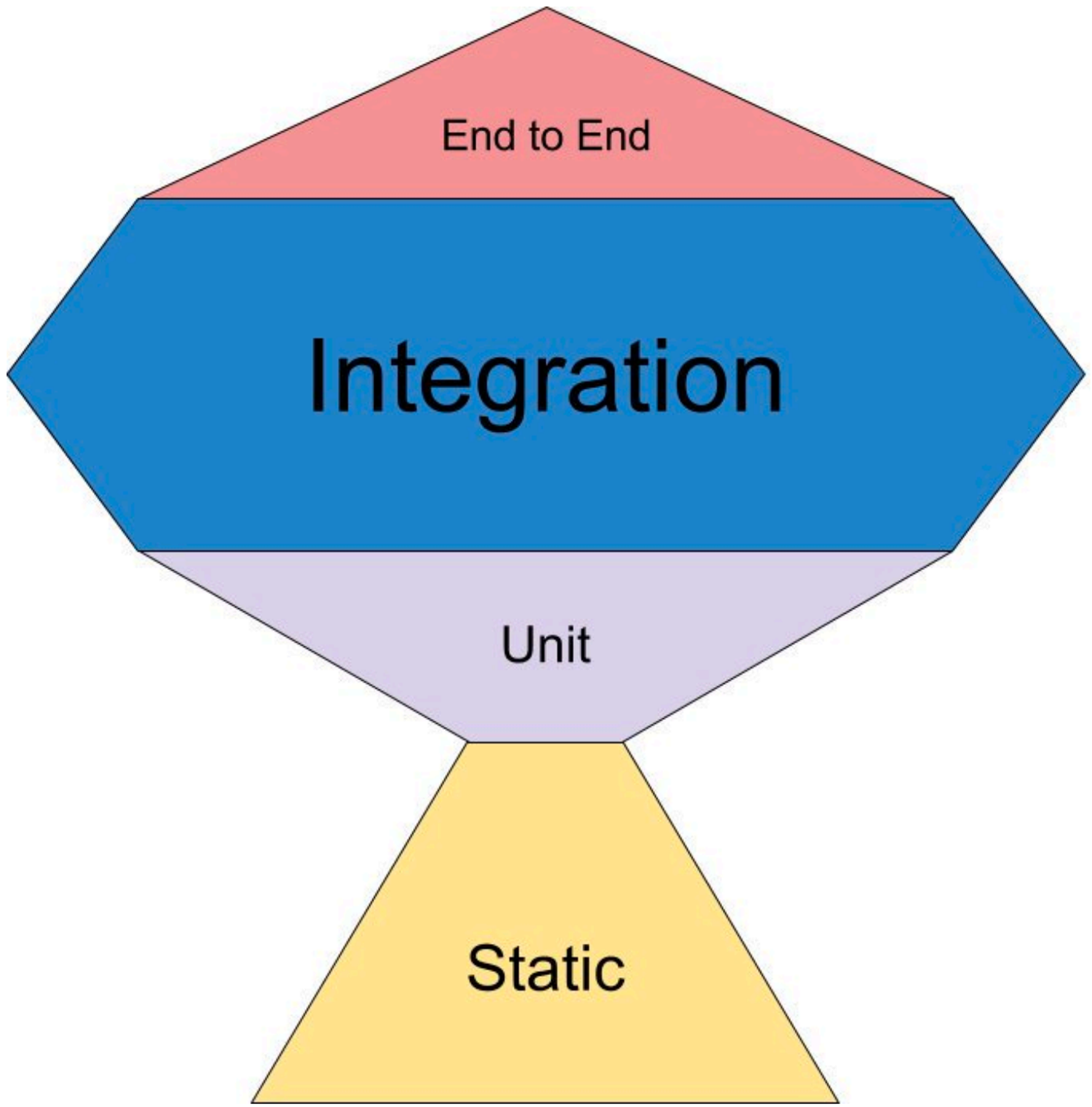
Think

What can I do to help deliver the software successfully !!

Cupcake testing



Trophy testing



Workshop Test Pyramid

Test login

User name	Password	Expected result	Comment
Somkiat	PasswordSomKiat	Access to system as Somkiat	Valid
Som kiat	PasswordSomKiat	Error	Space in user
Somkiat	Password SomKiat	Error	Space in password
Somkiat	1234	Error	Password too short

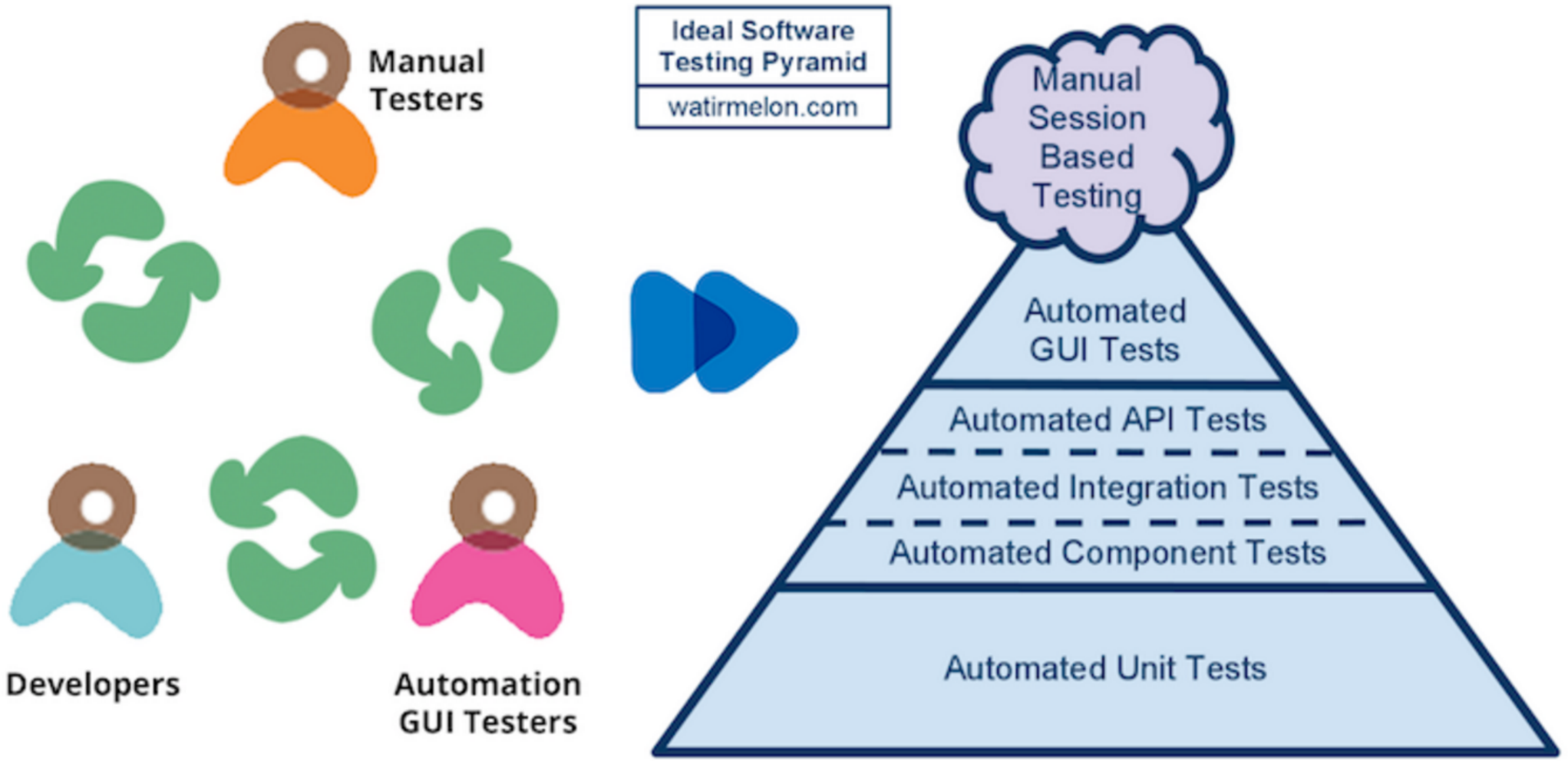
Workshop

1. Draw a test pyramid
2. Identify tests at each level
3. Think about value (business and customer)
4. What tests should be manual ?

https://bit.ly/2XPAjrl

https://bit.ly/3aofbLD

https://bit.ly/2Vp52dv



Understand What and How to test ?

Discussion is very important

Remember !!

Test pyramid is a tool
To talk about automation tasks
How to prioritise and help to do automation ?
Way to make **visible to the whole team**

Where to start ?

What are the **biggest obstacles** ?

Time/Tools/System/People

What should be careful ?

Automating end-to-end tests

User Interface are slow

Working with database

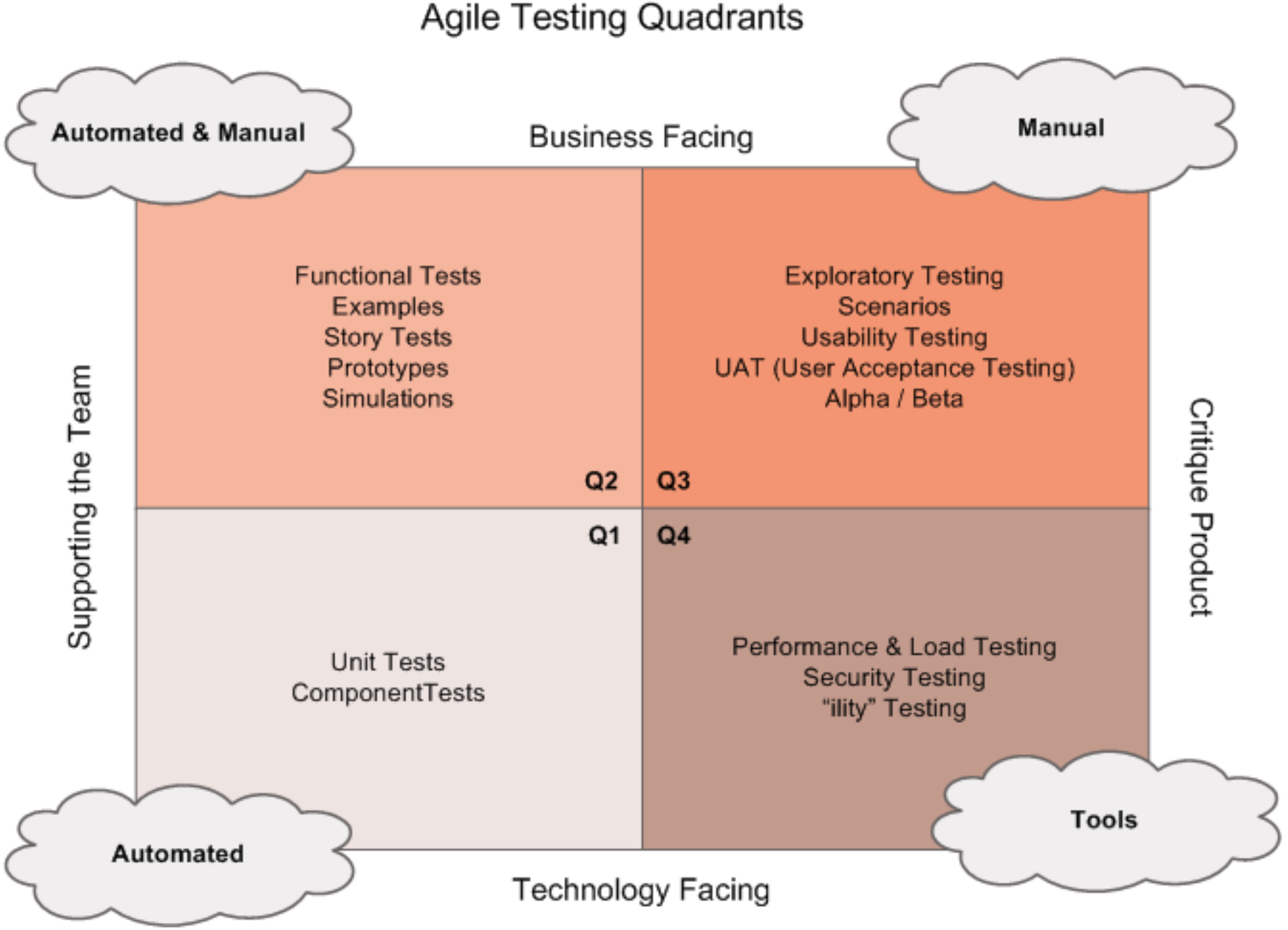
Working with external system

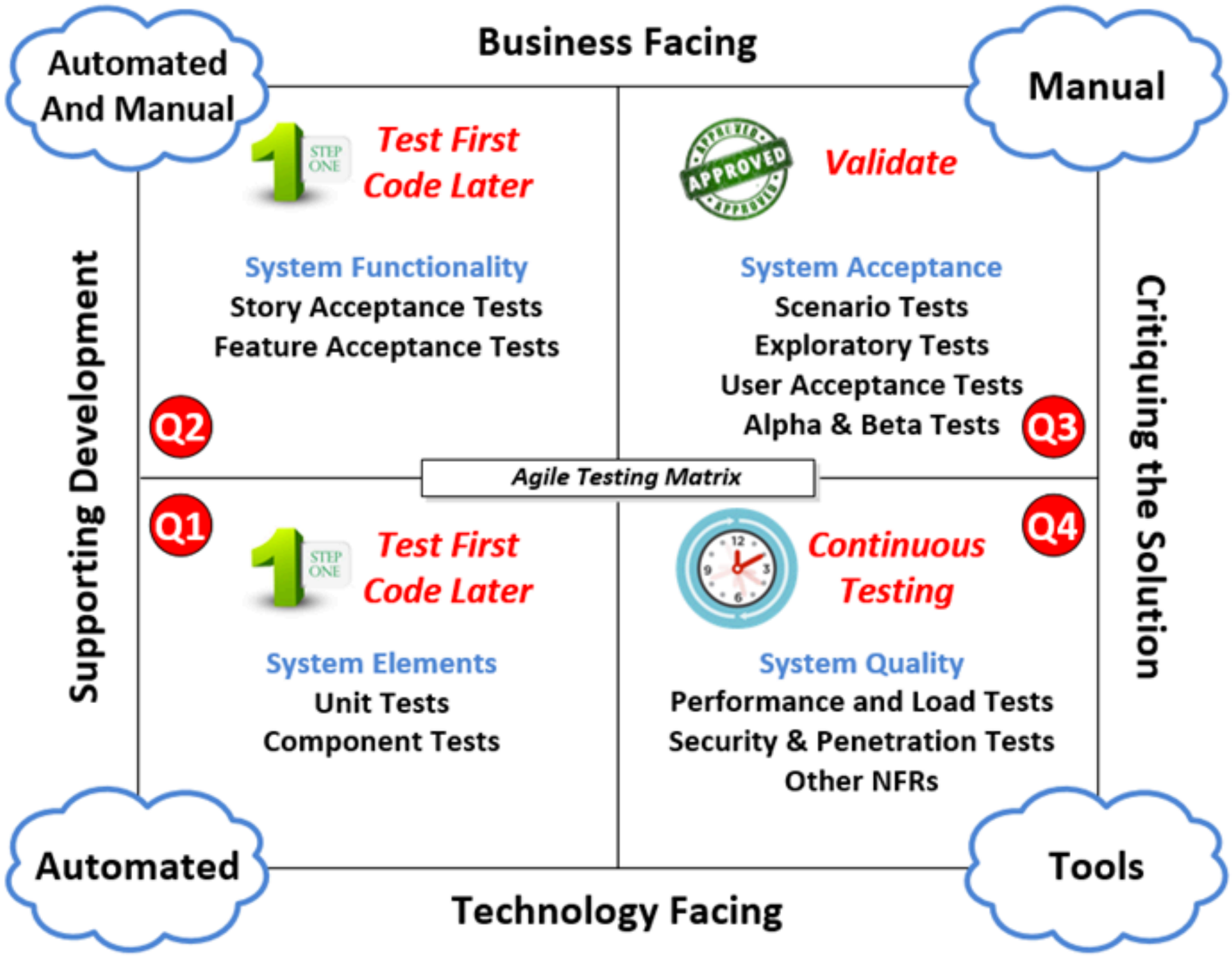
Automating every paths

Testing Quadrant

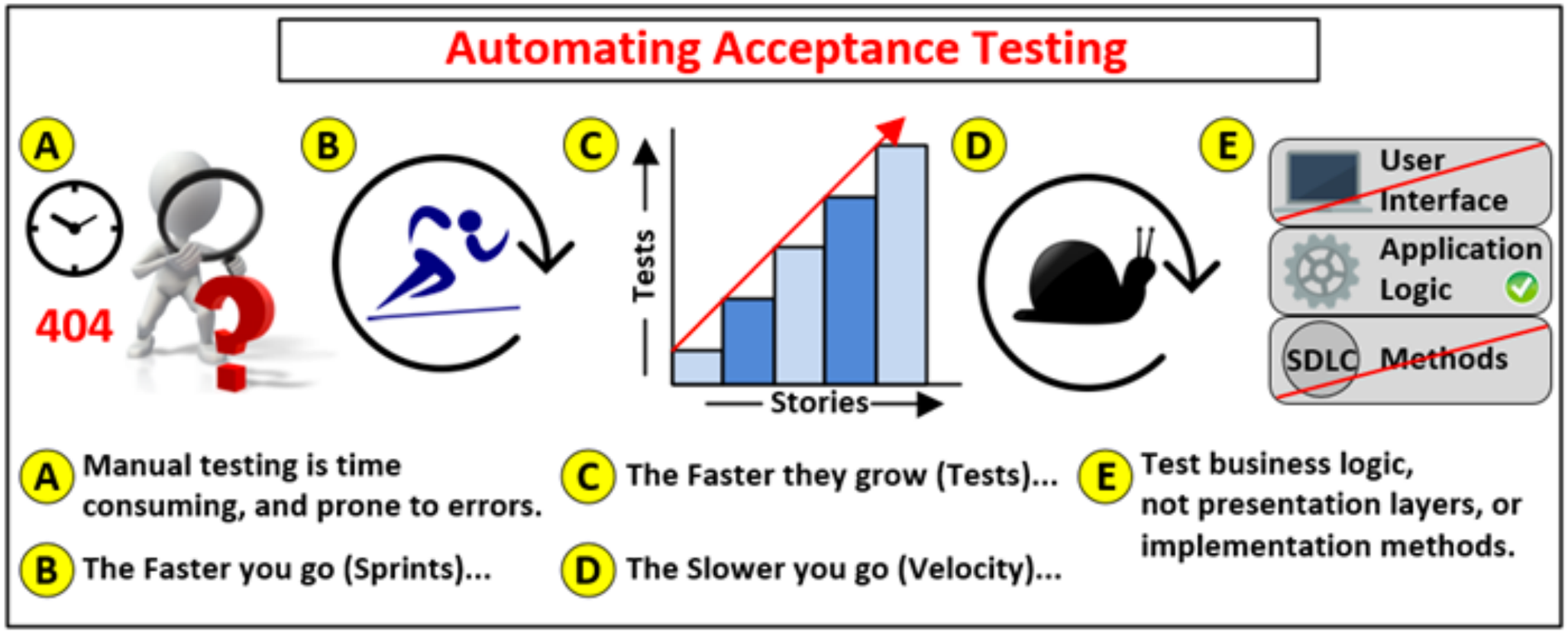
Testing Quadrants

Scope of tests
Classify your tests
Visible your tests





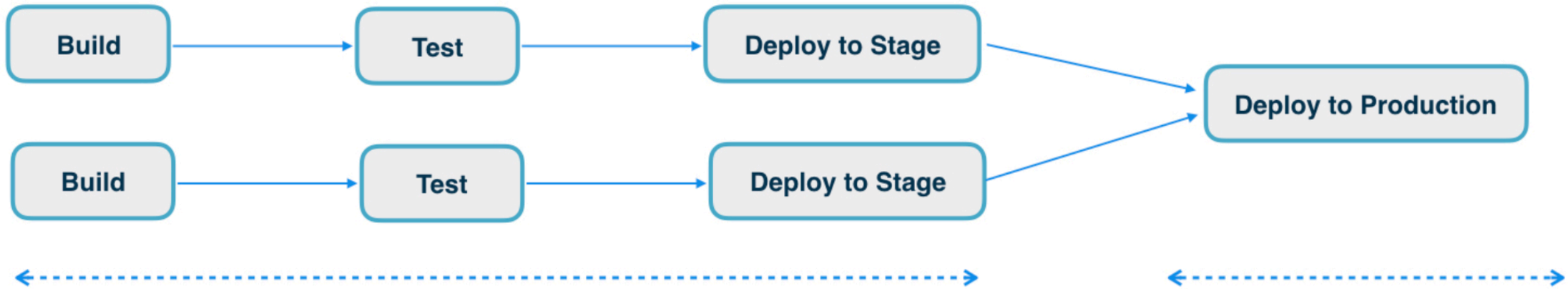
Test automation is essential

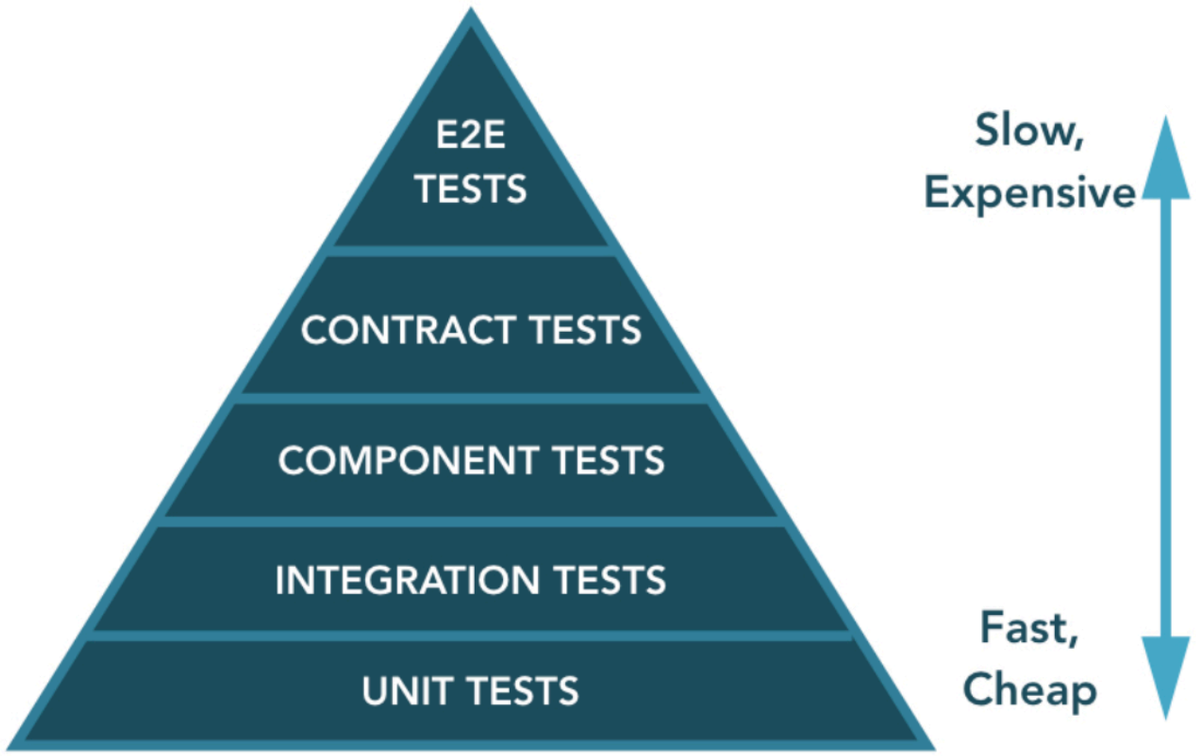


Test strategy

https://martinfowler.com/articles/microservice-testing/

Test strategy

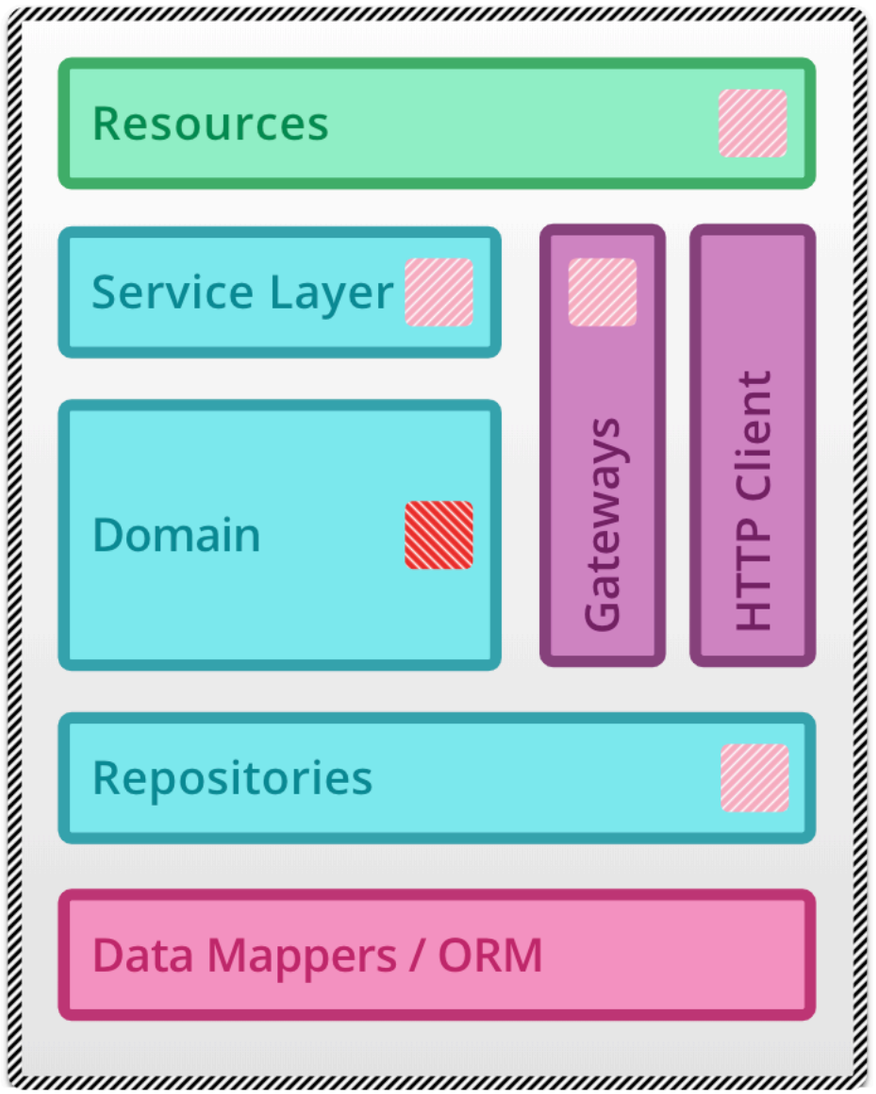




Monitoring
Distributed Tracing
Fault-injection Testing

Canary Deployments
Blue-Green Deployments
A/B Testing

Unit testing

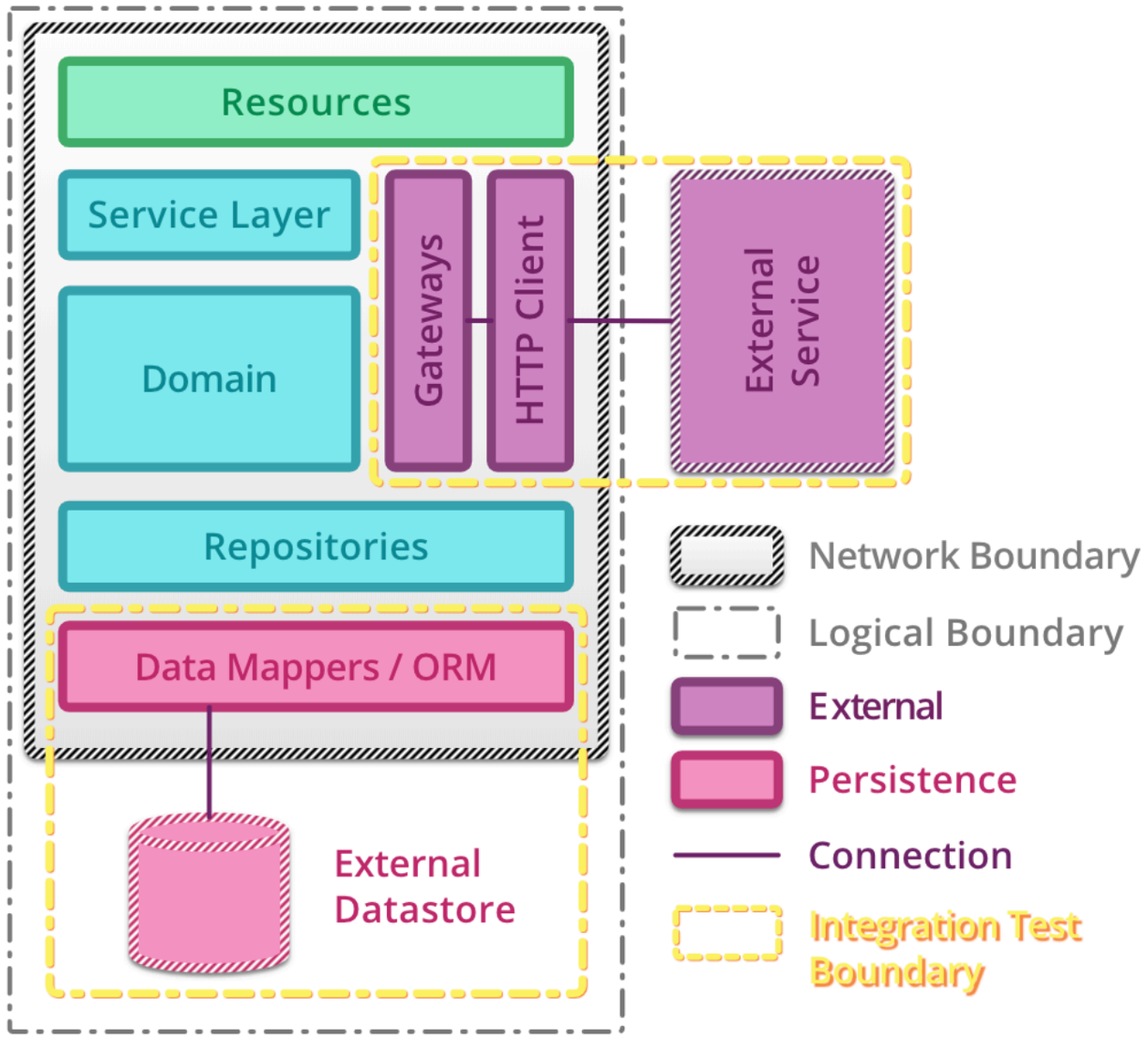


Services are commonly a
coordination code.

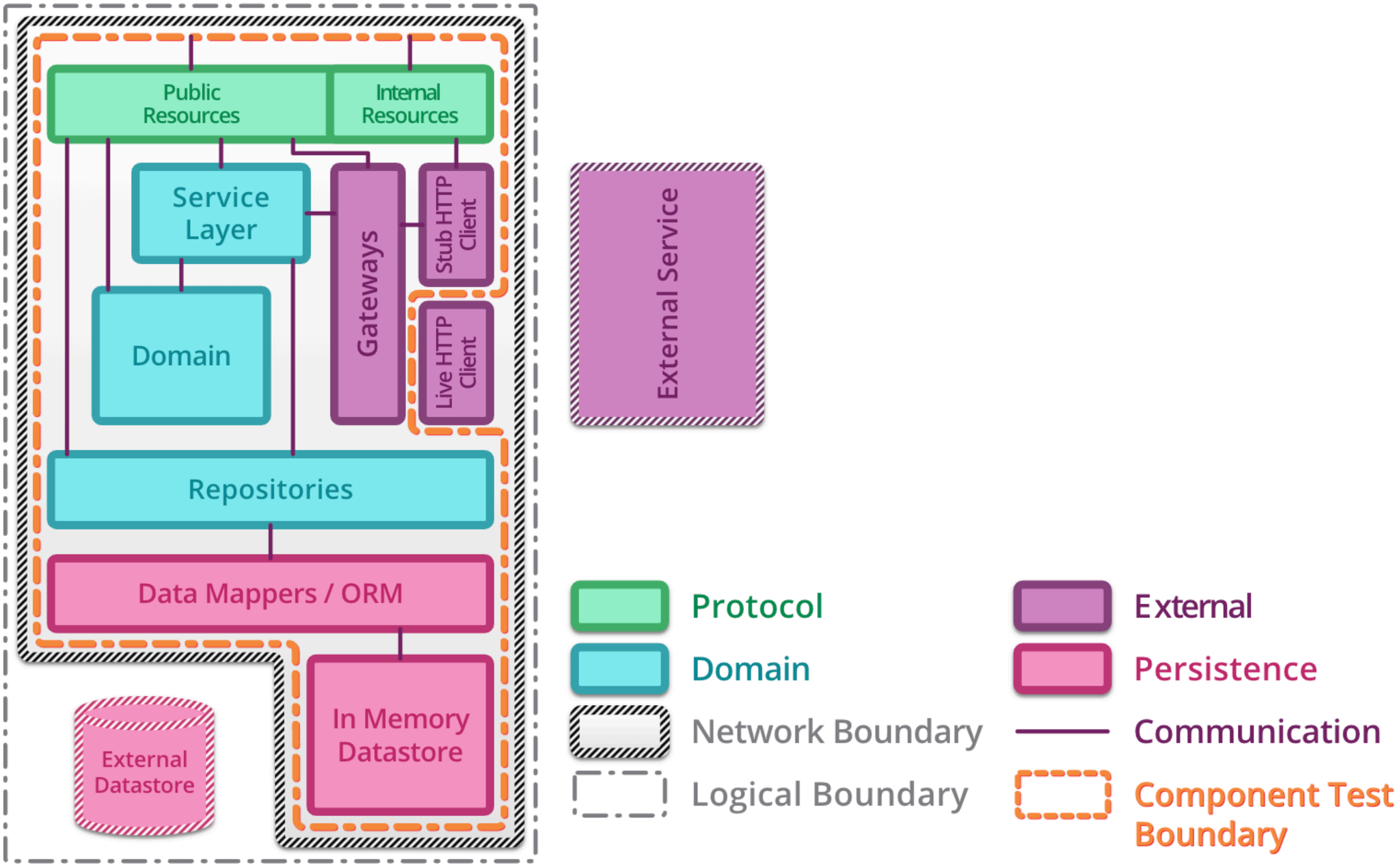
Unit - Solitary

Unit - Sociable

Integration testing



Component testing



Test design

Economy of Test Design

Easy to understand
Easy to maintain
Readable by the business
One purpose per test
Re-runnable

Economy of Test Design

Easy to understand
Easy to maintain
Readable by the business
One purpose per test
Re-runnable

Poor test practices reduce the benefits

Respect your tests

Don’t ignore it if it fails

Fix the code or fix the test

100% of regression tests must pass all the time

Always refactor or improve

Test data !!

Avoid database/external system access

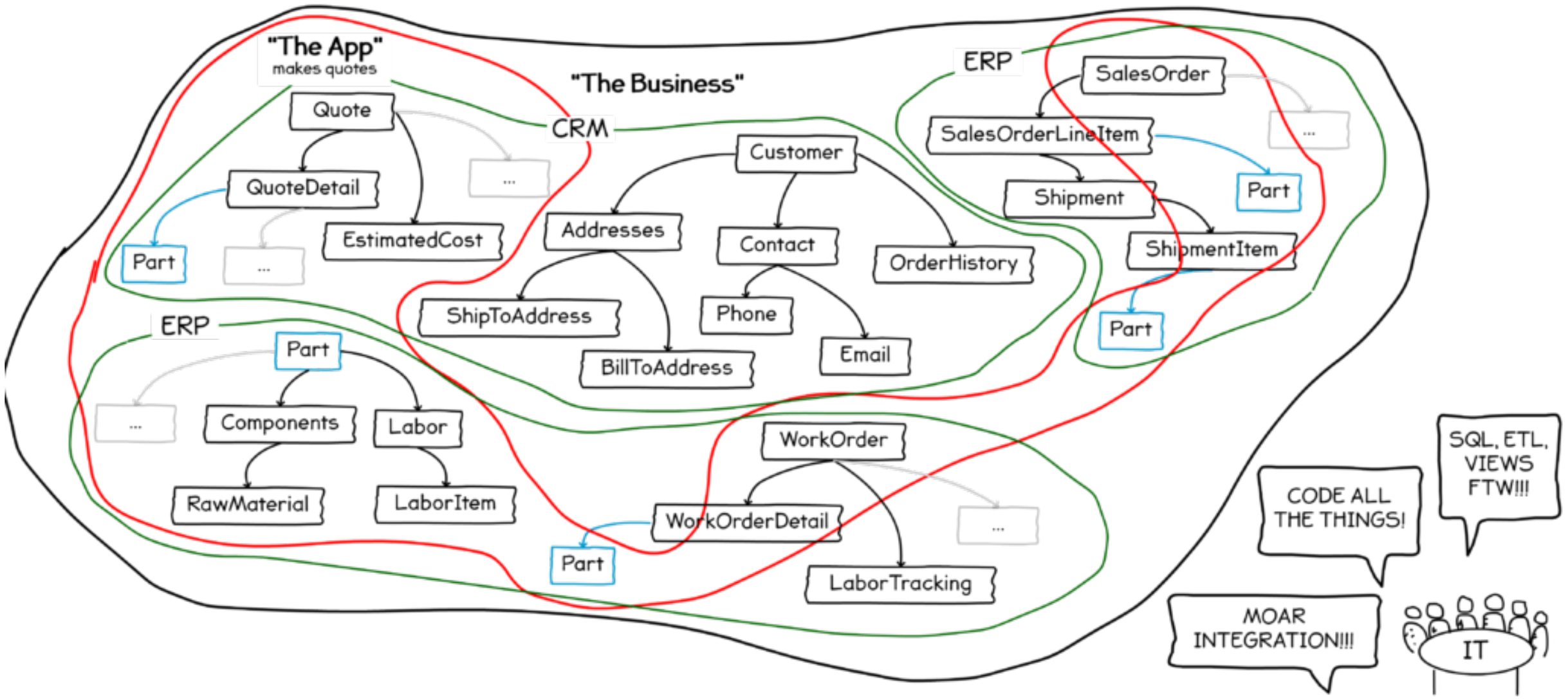
Setup/ tear down test data

Use production-like data

Need to control your data test

System impacts !!

Know your systems

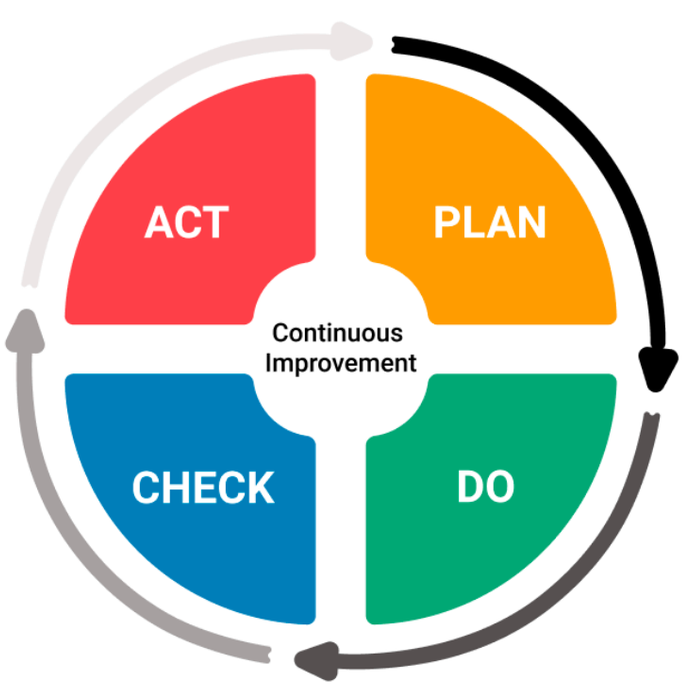


Automation feedback

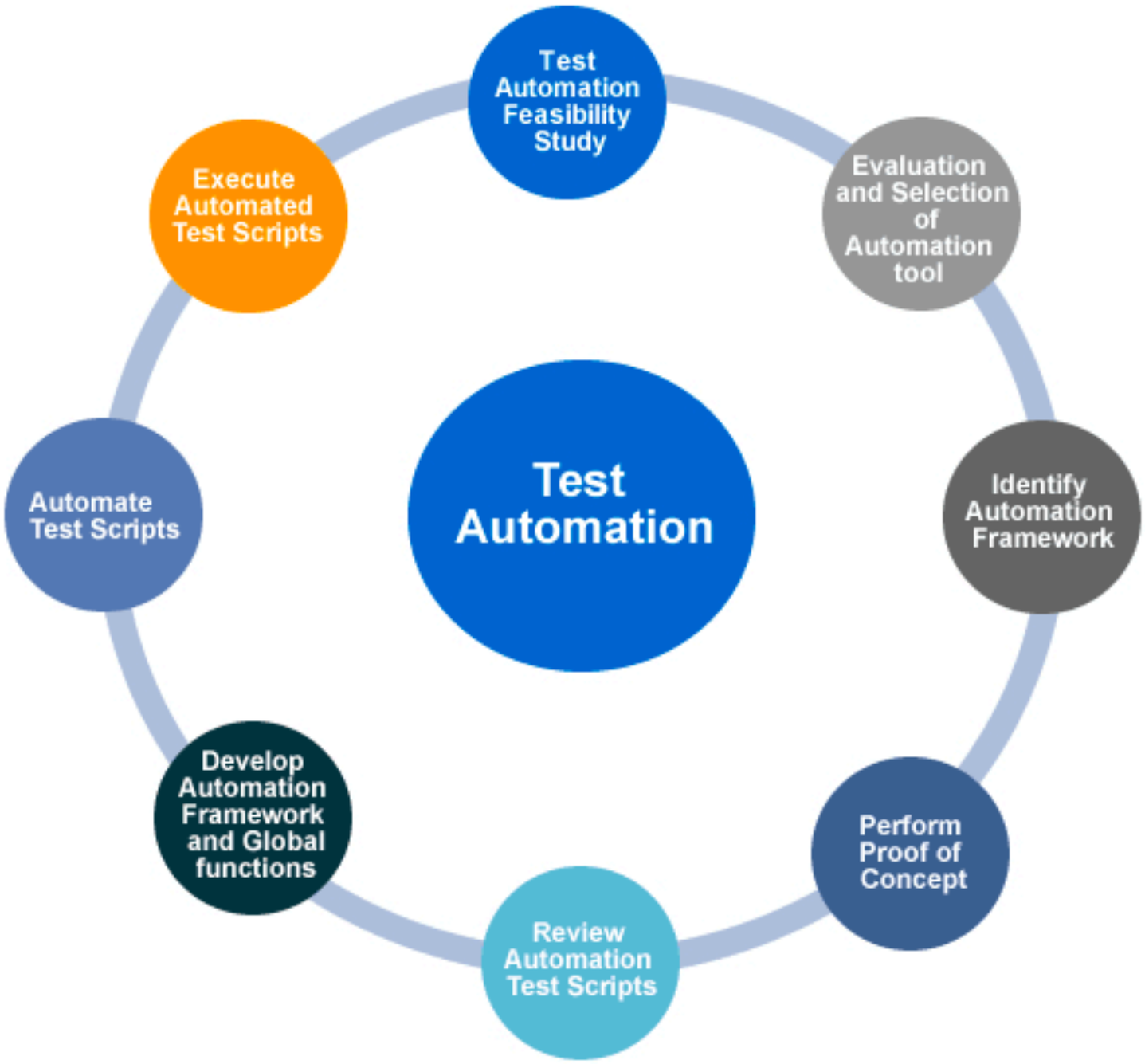
Easier over time ?

Time spent on maintenance ?

Test find regression bugs ?



Test Automation selection



Manage automated tests

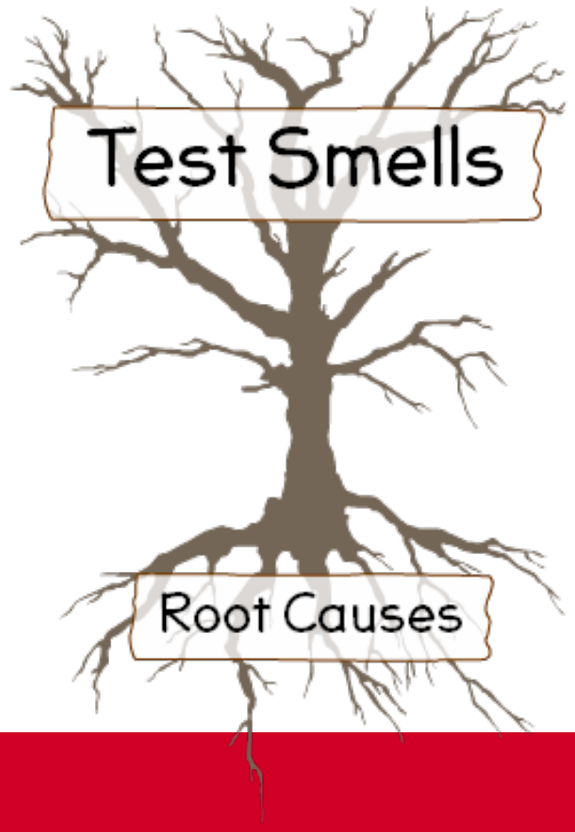
Use source control

Continuous integration to run tests on every change

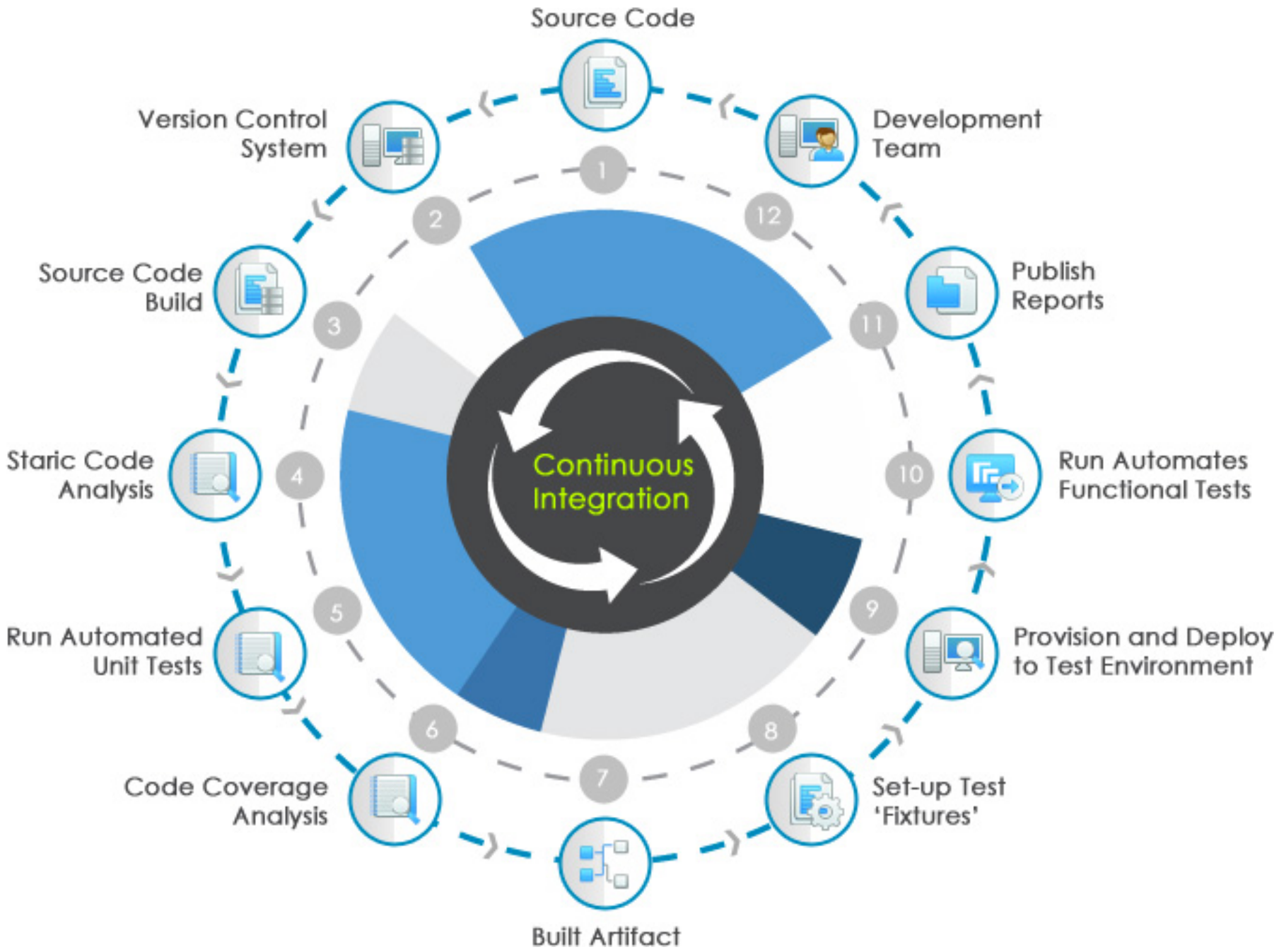
Keep all tests passed

Analyse failures tests

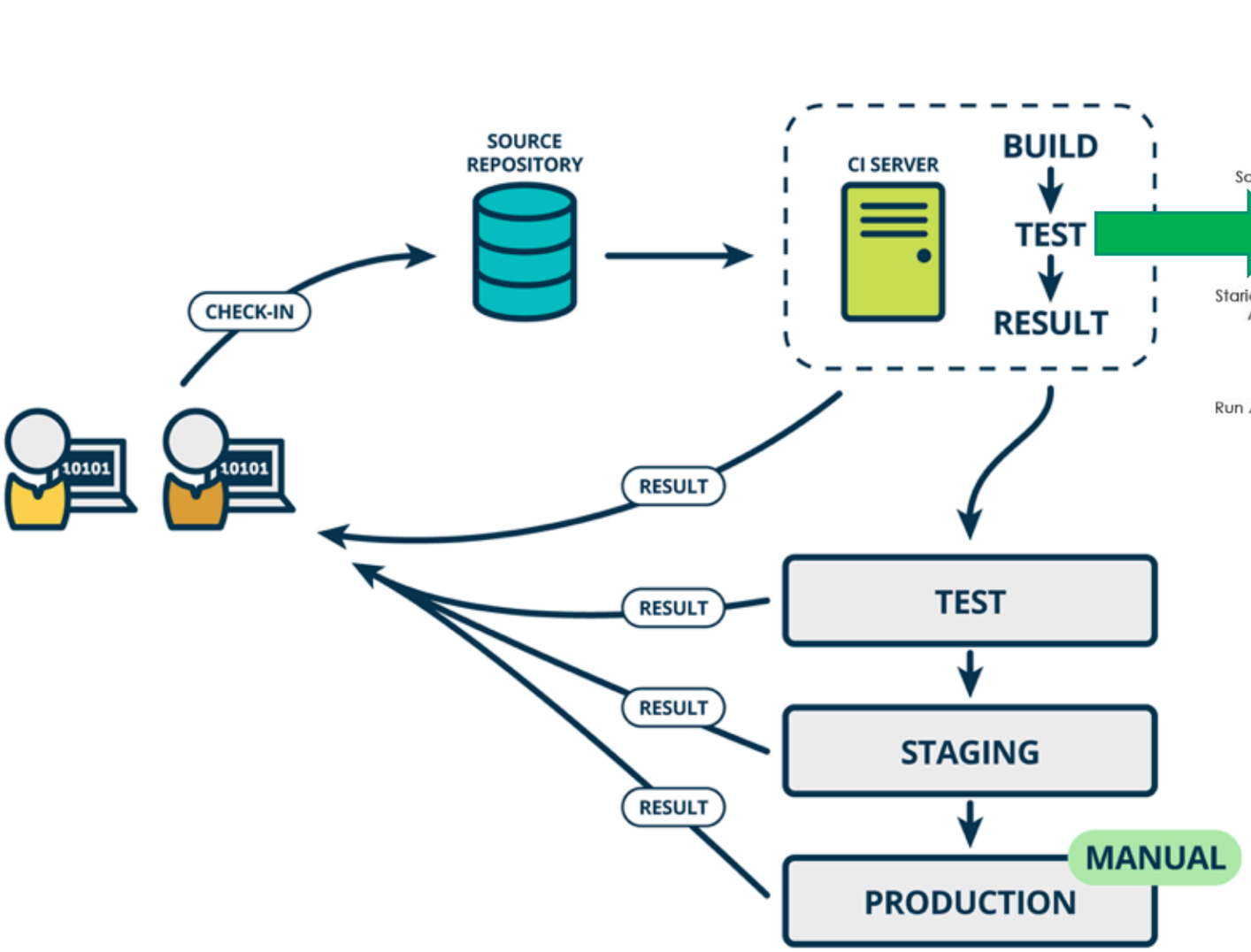
Create test standard

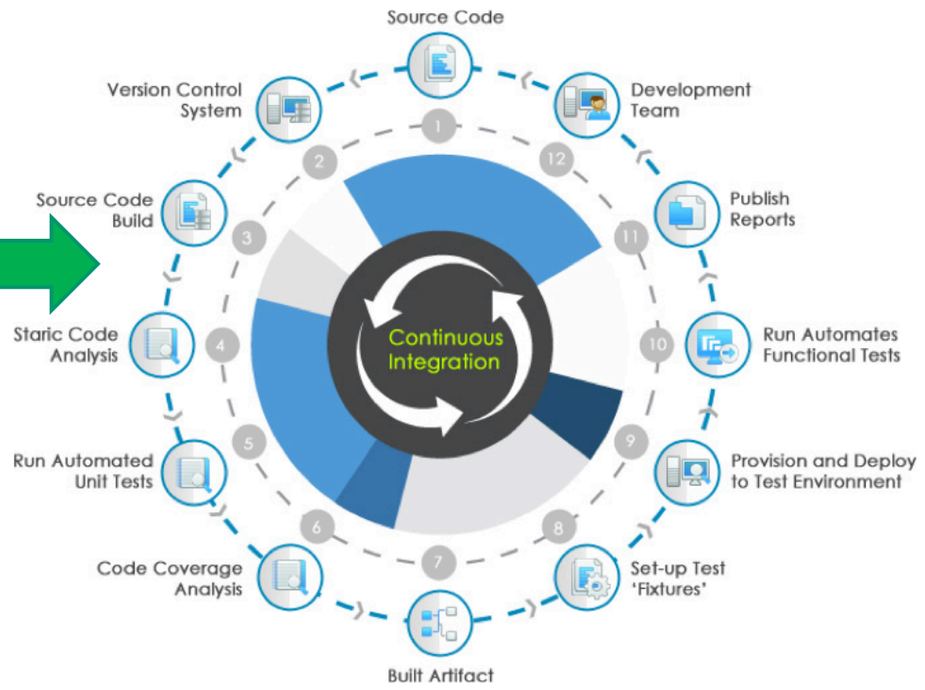


Continuous integration

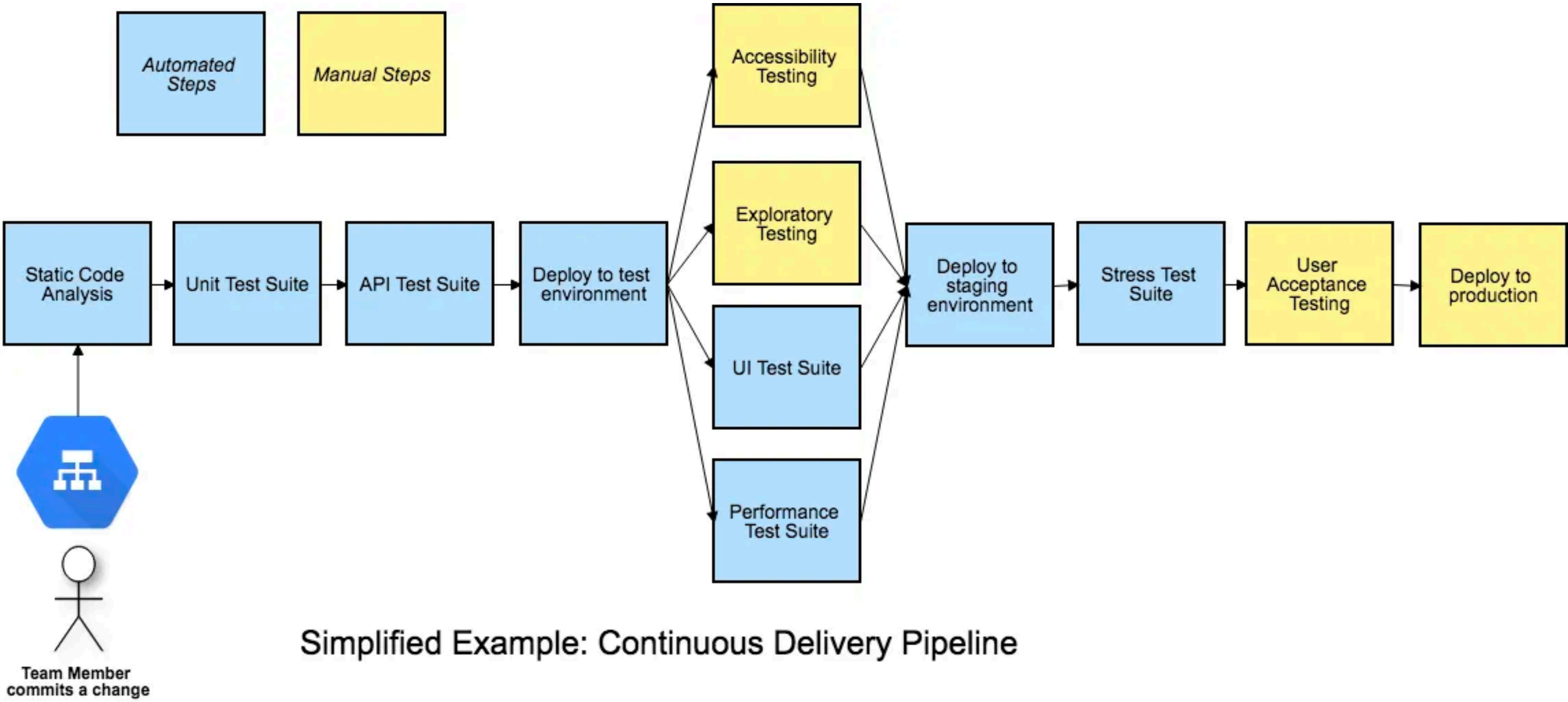


Continuous integration



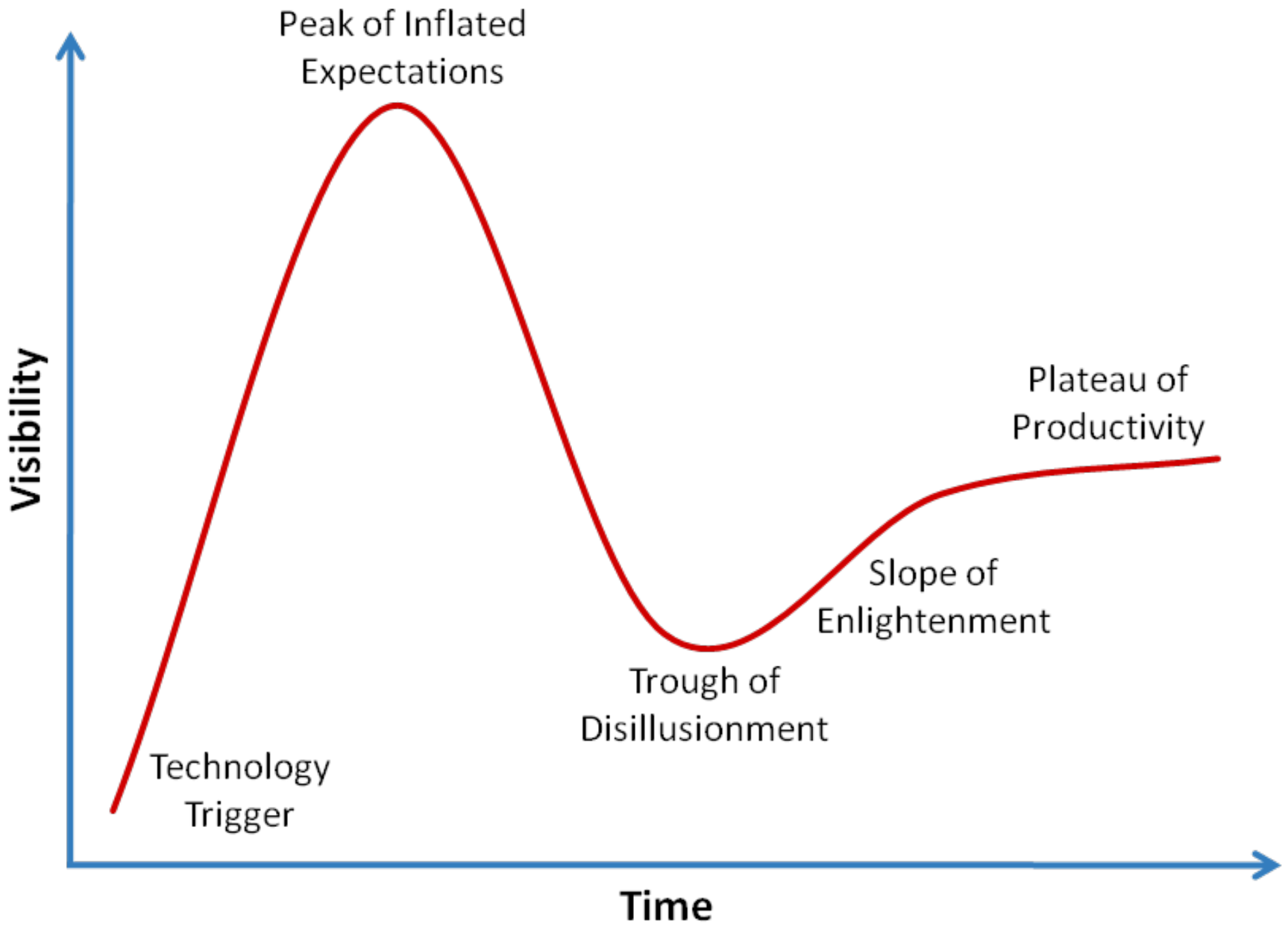


Design your pipeline/process



Simplified Example: Continuous Delivery Pipeline

Hype cycle



Start with simple

Use feedback to improve

Module #2
Automate Unit Tests

What Unit test ?

Software programs written to exercise other software programs with **specific preconditions** and verify the **expected behaviours**

Usually written in the **same programming language**

Test code

Production code

What Unit test ?

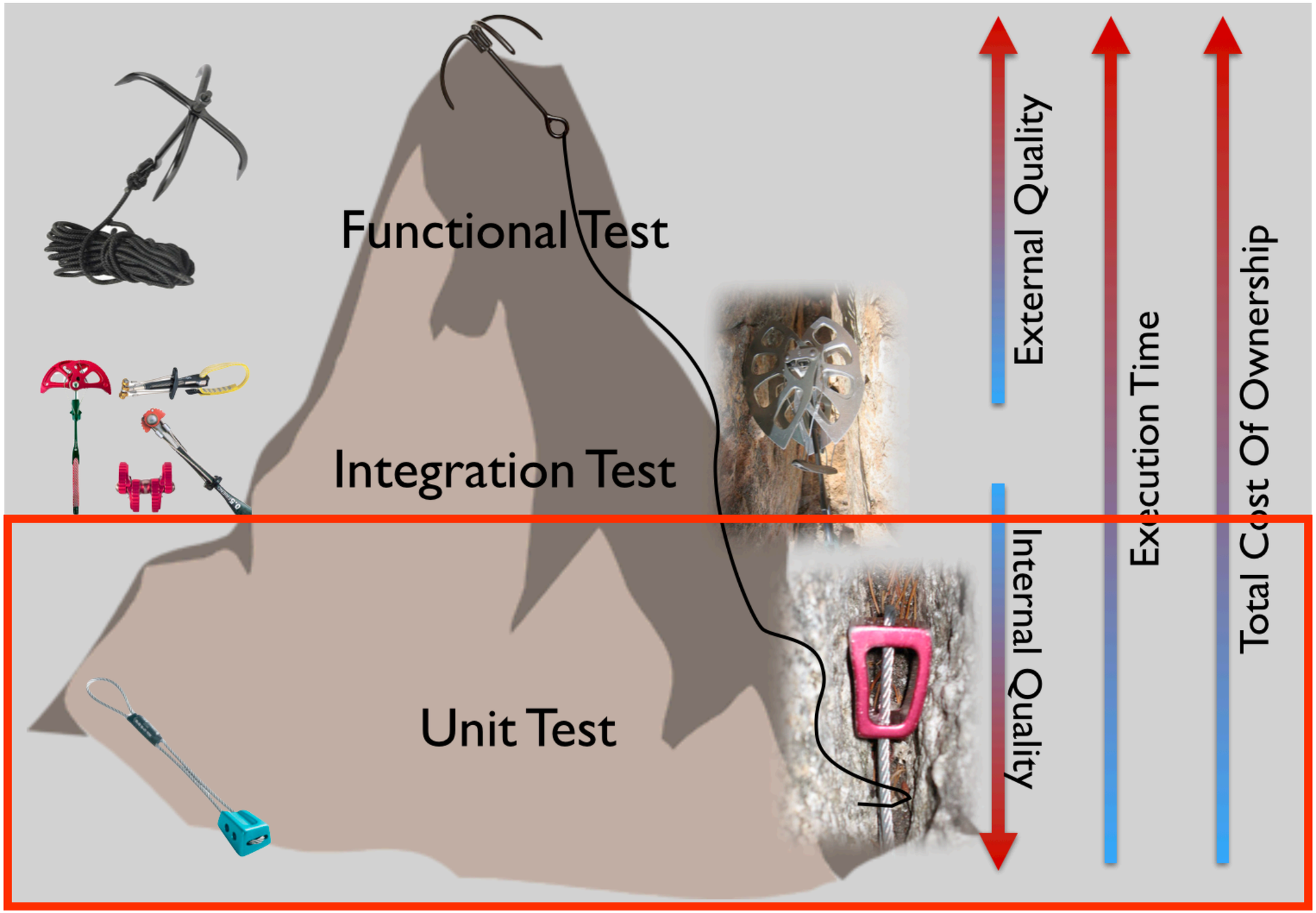
Small and test only limited pine of code functionality

Run very fast (100+ within a few seconds)

Golden rule of Unit test

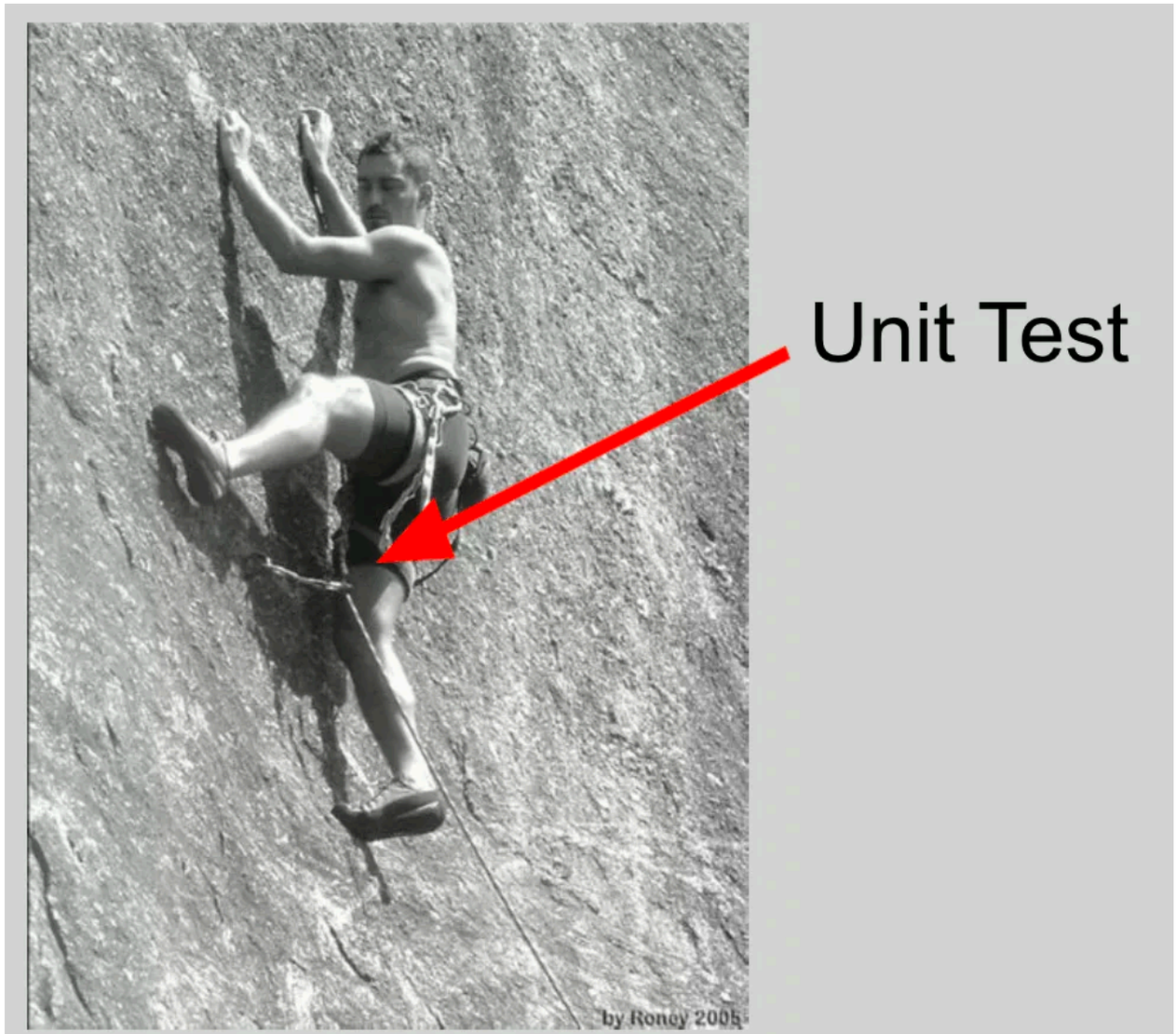
Each unit test case should be
very limited in scope

Why Unit test ?



Why Unit test ?

Protect what we have implemented than to find any defects



Purpose of Unit test

Facilitates changes

Simplifies integration

Documentation

Design tool

Good Unit Tests (F.I.R.S.T)

Fast

Independent/Isolate

Repeat

Self-validate

Thorough/Timely

We need testable code and easy to test

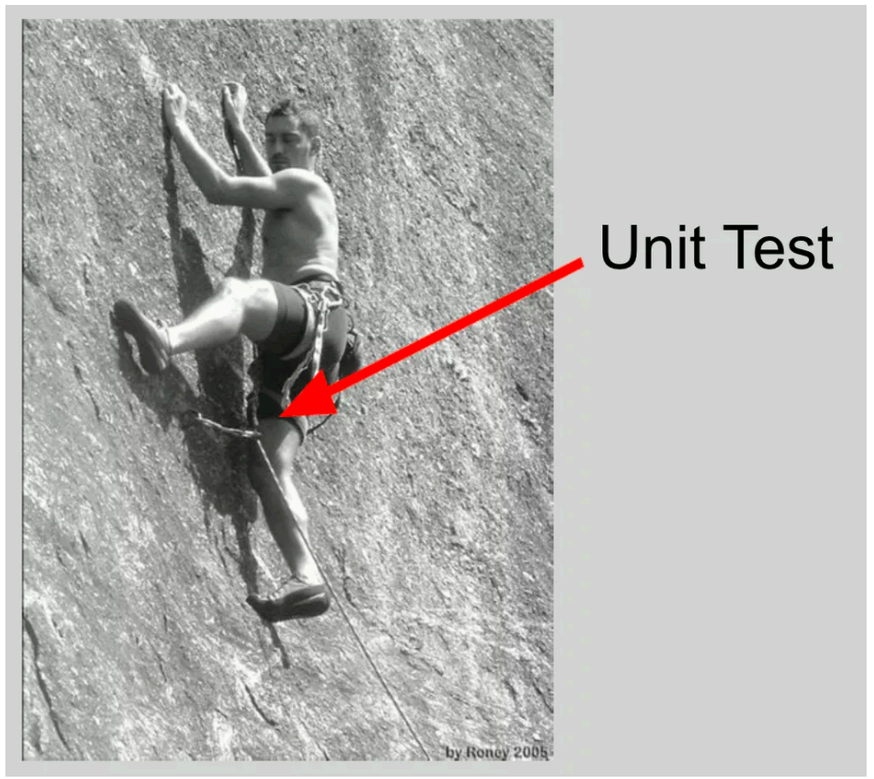
Misconception of Unit test

Not as important as the production code

Done by testing engineers

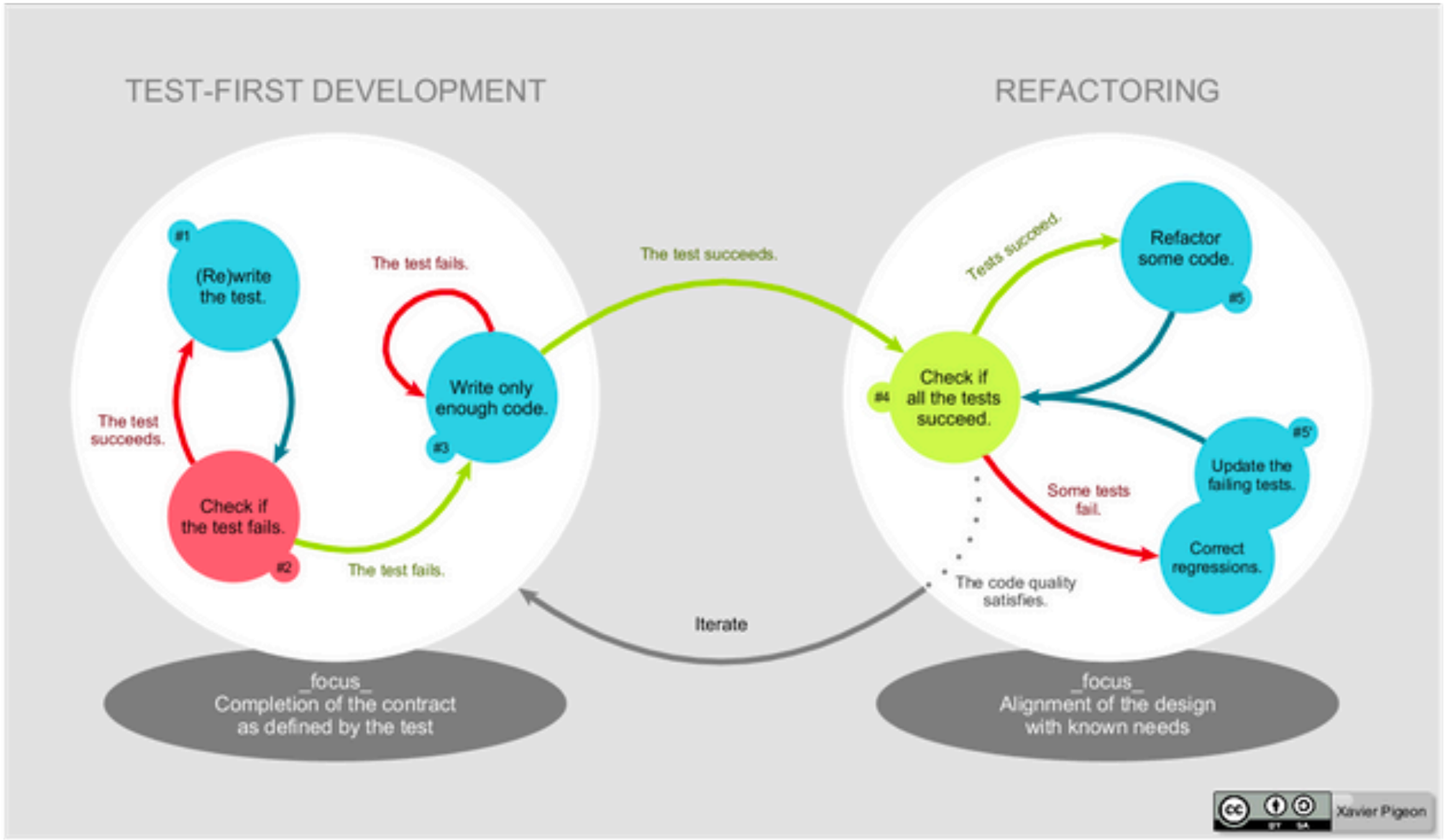
Write unit test without changing production code

Add unit test later



Let’s start with Test-First

Test-First

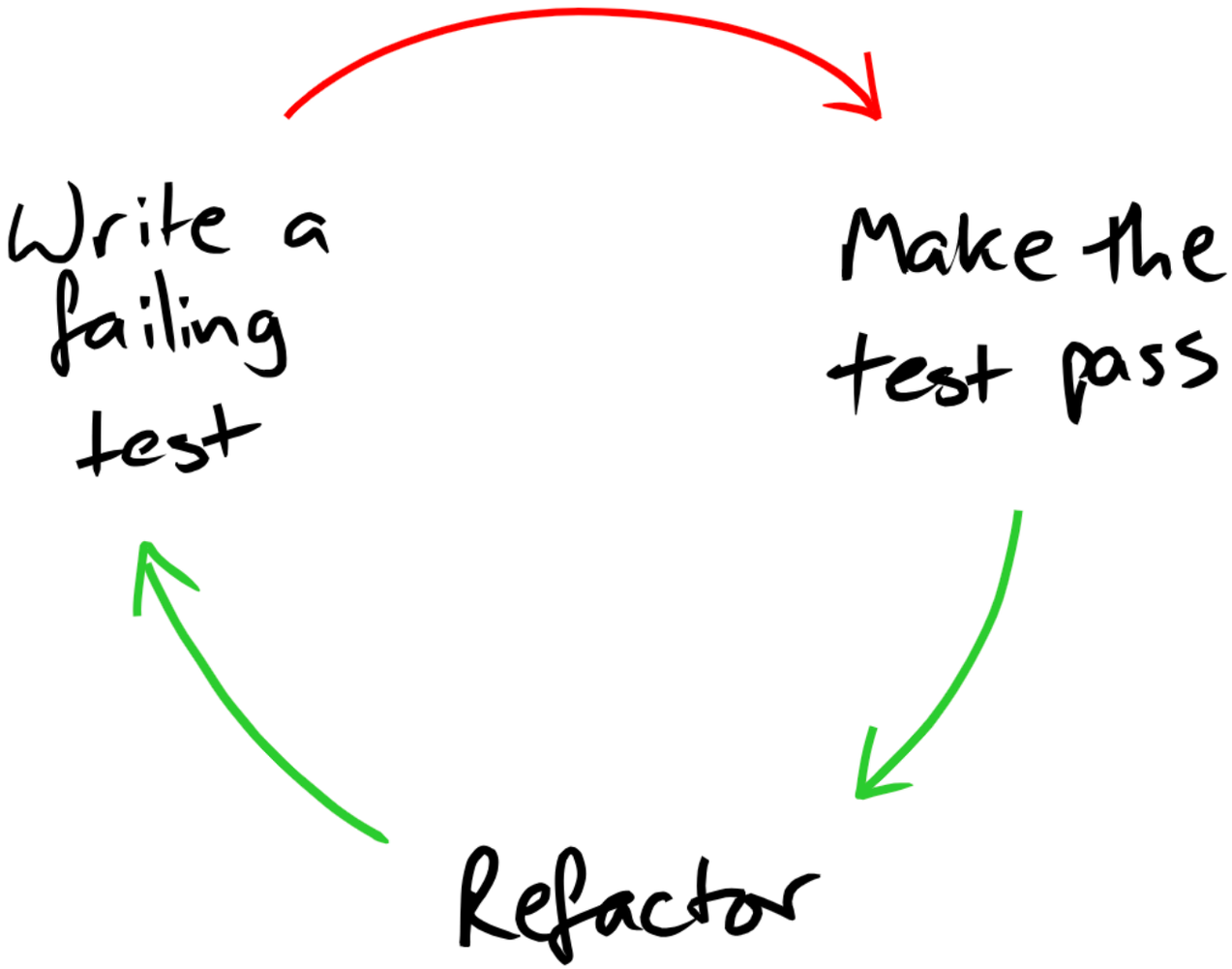


TDD

ALL CODE IS GUILTY UNTIL PROVEN INNOCENT

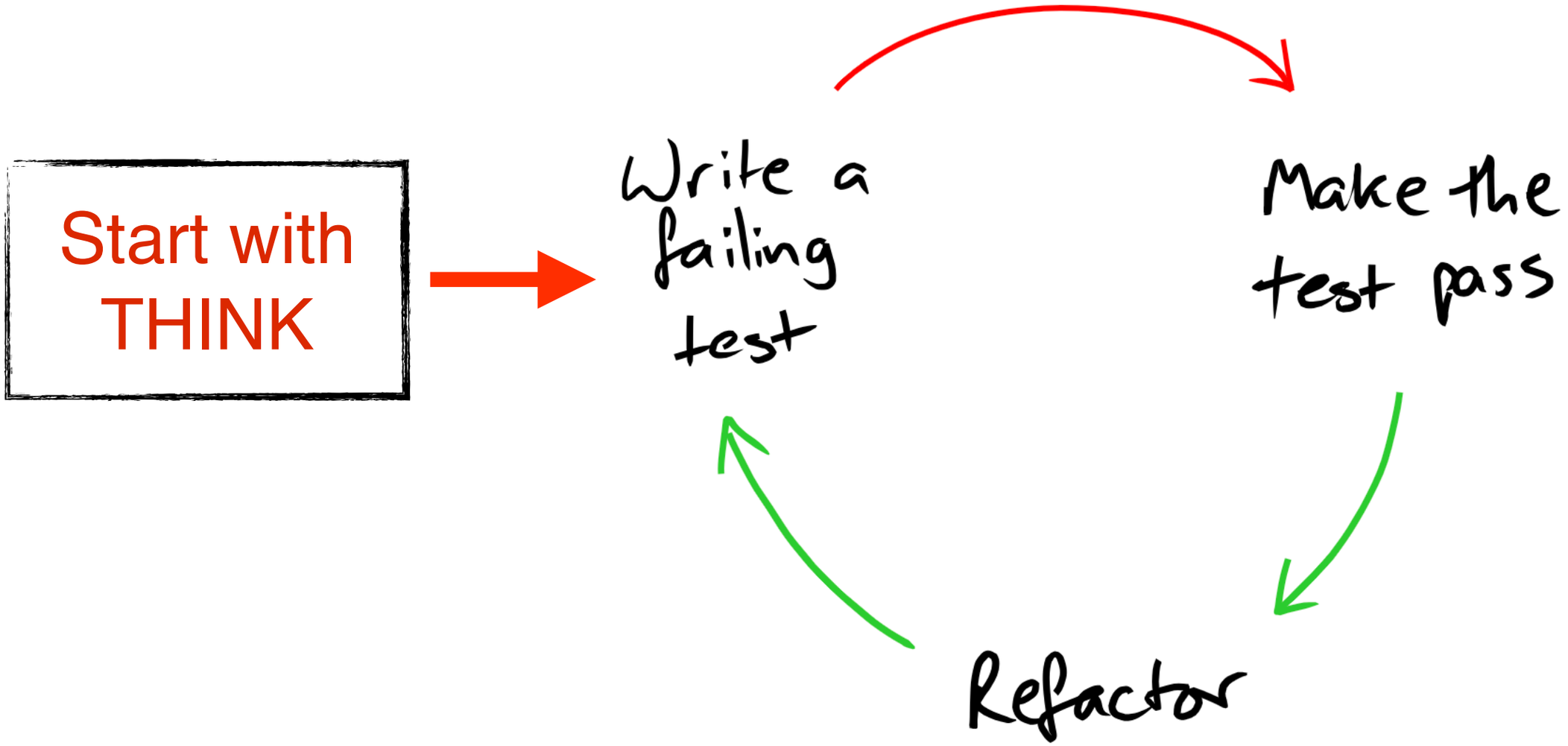
TDD === ทำดีดี

TDD Cycle

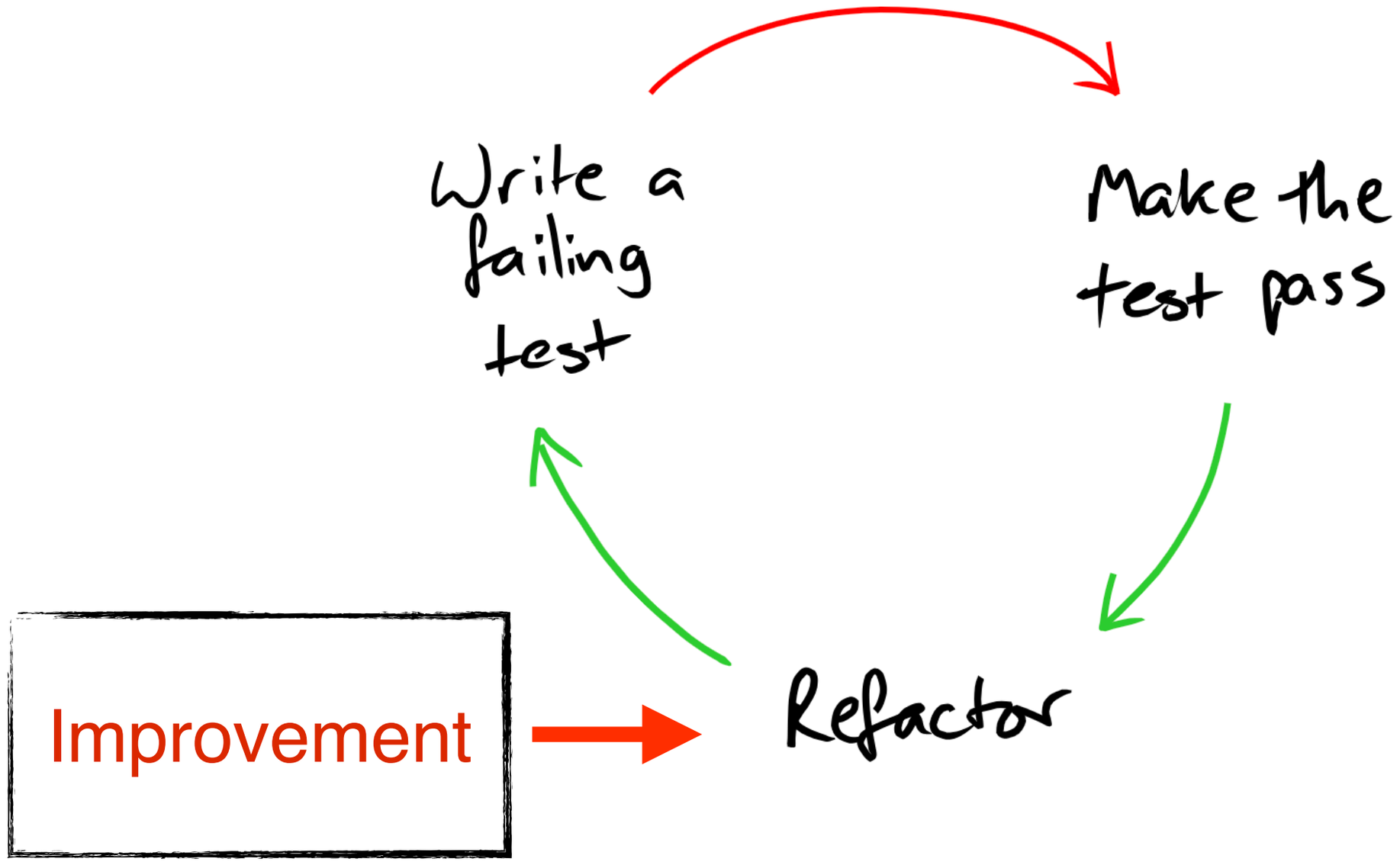


Red Green Refactor

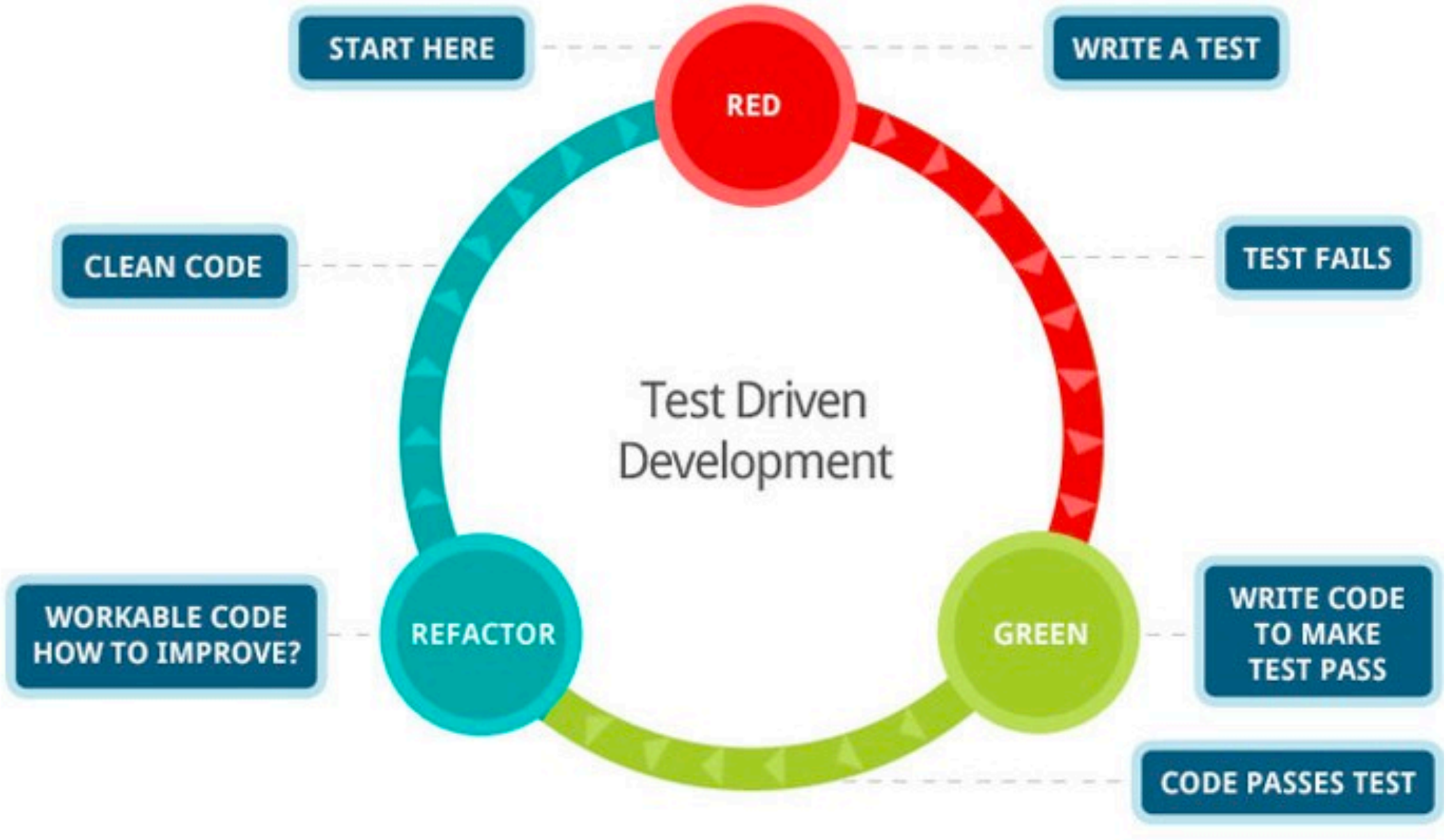
Improve TDD Cycle



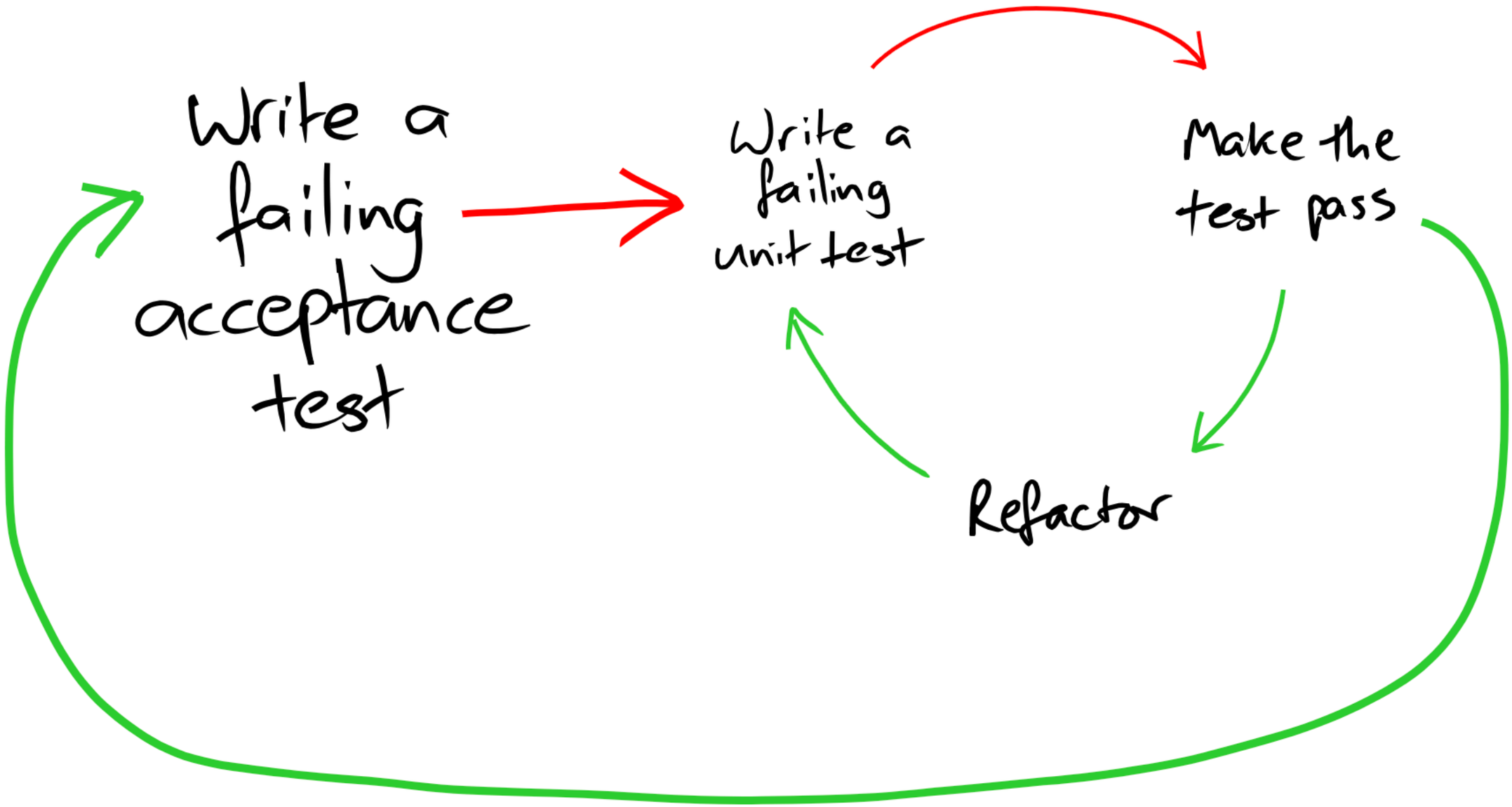
Improve TDD Cycle



Improve TDD Cycle



Acceptance tests



The Golden rule of TDD

"Never write a line of code without a failing test"

- Kent Beck -

Workshop

Input	Expected result
[1, 5]	1,2,3,4,5
[1, 5)	1,2,3,4
(1, 5]	2,3,4,5
(1, 5)	2,3,4

http://codingdojo.org/kata/Range/

Organize your tests

How to make your tests visually **consistent** ?

Keeping tests maintainable by testing **behavior**

The importance of **test naming**

Using **test's life cycle**

Good test structure

1. Setup the test data
2. Call your method under test
3. Assert that the expected results are returns

Good test structure

AAA (Arrange Act Assert)
Given When Then from BDD style

https://xp123.com/articles/3a-arrange-act-assert/
https://www.martinfowler.com/bliki/GivenWhenThen.html

Code coverage

"Code coverage can show the high risk areas in a program, but never the risk-free."

Paul Reilly, 2018, Kotlin TDD with Code Coverage

Code coverage

A tool to measure how much of your code is covered by tests that break down into classes, methods and lines.

Code coverage

```
1.  public class MyRange {
2.      public boolean startWithInclude(String input) {
3.          return input.startsWith("[");
4.      }
5.
6.      public boolean endWithInclude(String input) {
7.          if(input == null) {
8.              return false;
9.          }
10.         return input.endsWith("]");
11.     }
12.
13.     public boolean startWithInclude2(String input) {
14.         return input.startsWith("[");
15.     }
16. }
```

Code coverage

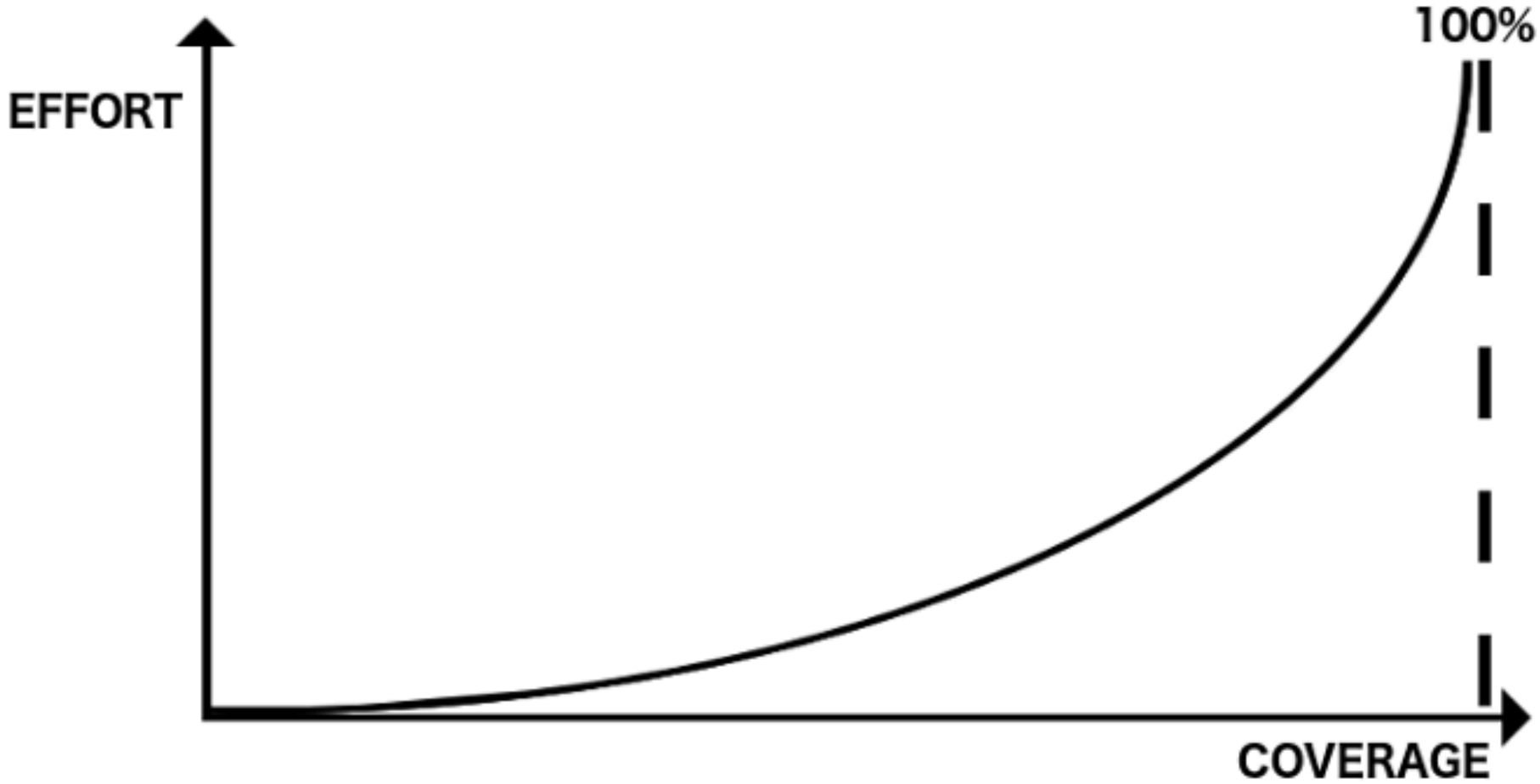
But 100% of code coverage **does not mean** that your code is 100% correct

Code coverage

Powerful tool to improve the quality of your code

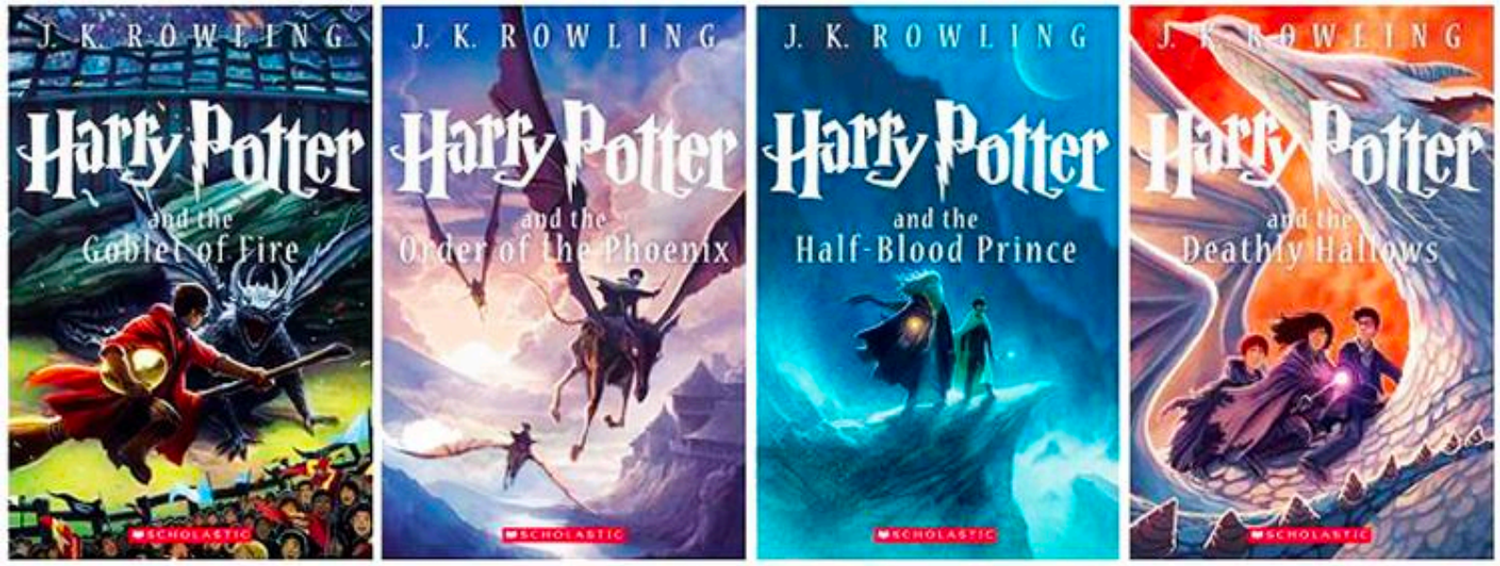
Code coverage != quality of tests

Code coverage 100% ?



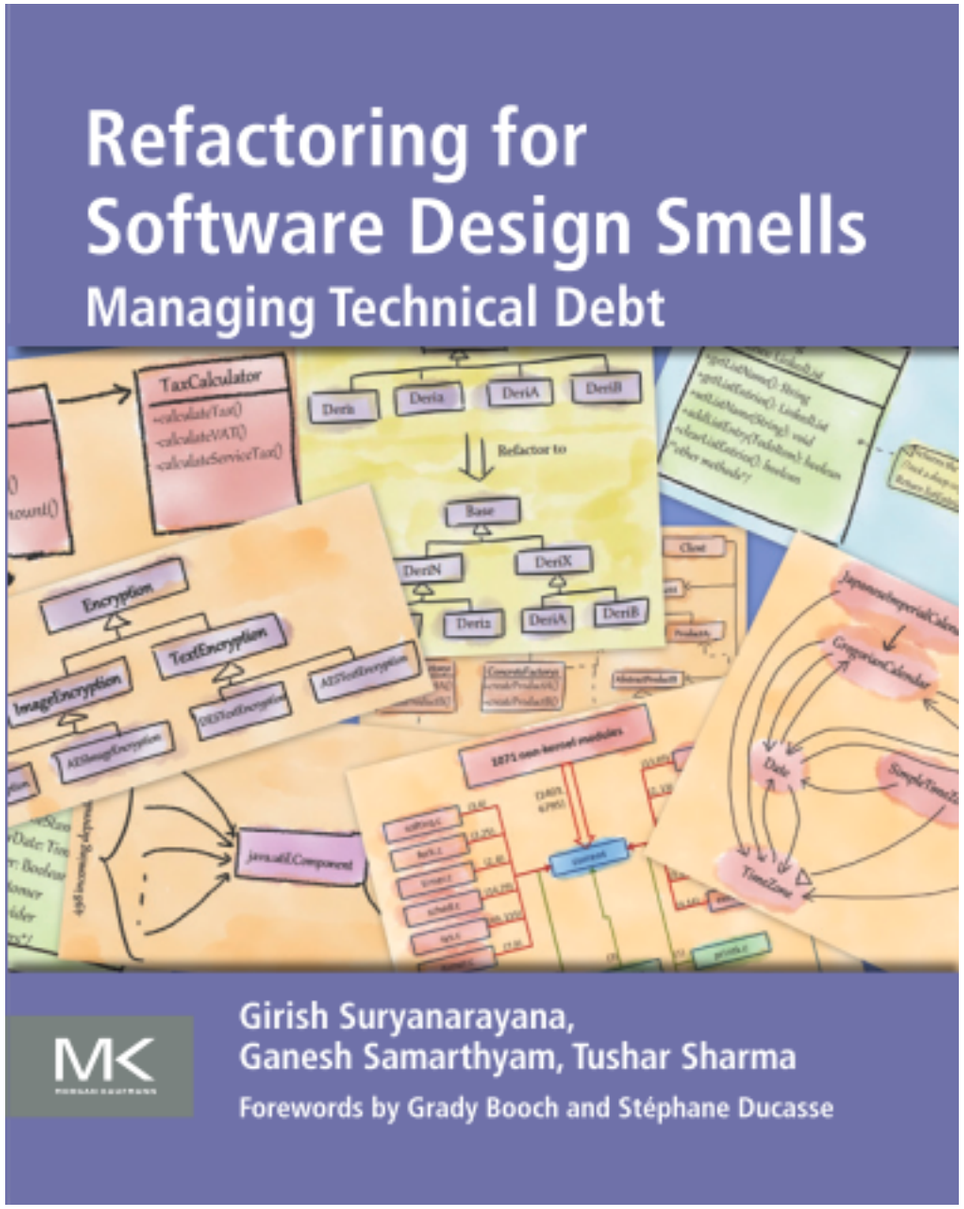
Test as Design

Workshop

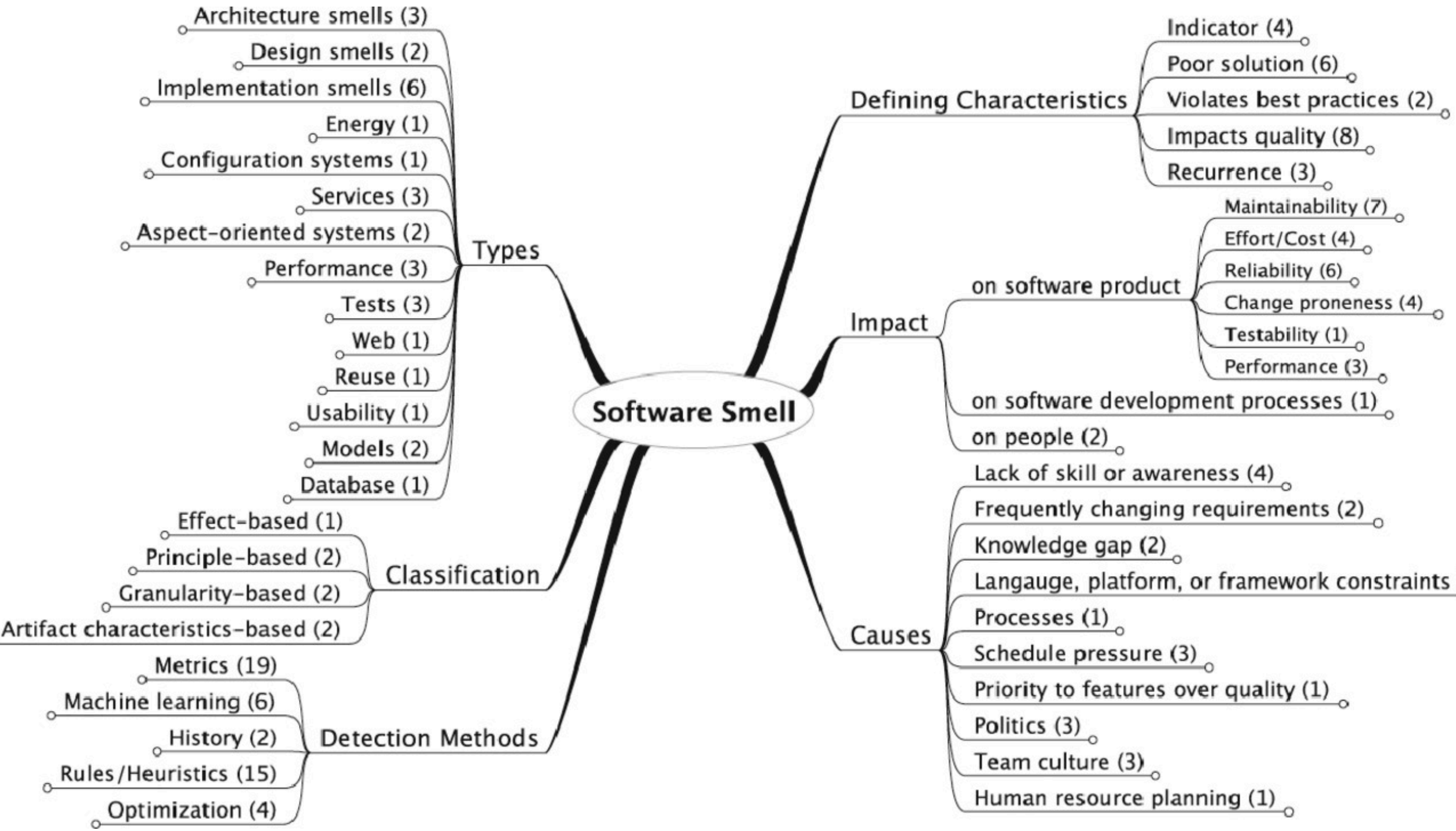


http://codingdojo.org/kata/Potter/

Don’t forgot Refactoring



http://www.tusharma.in/smells/



https://www.sciencedirect.com/science/article/pii/S0164121217303114

Code Smell

Code Smells

– What? How can code "smell"??

– Well it doesn't have a nose... but it definitely can stink!

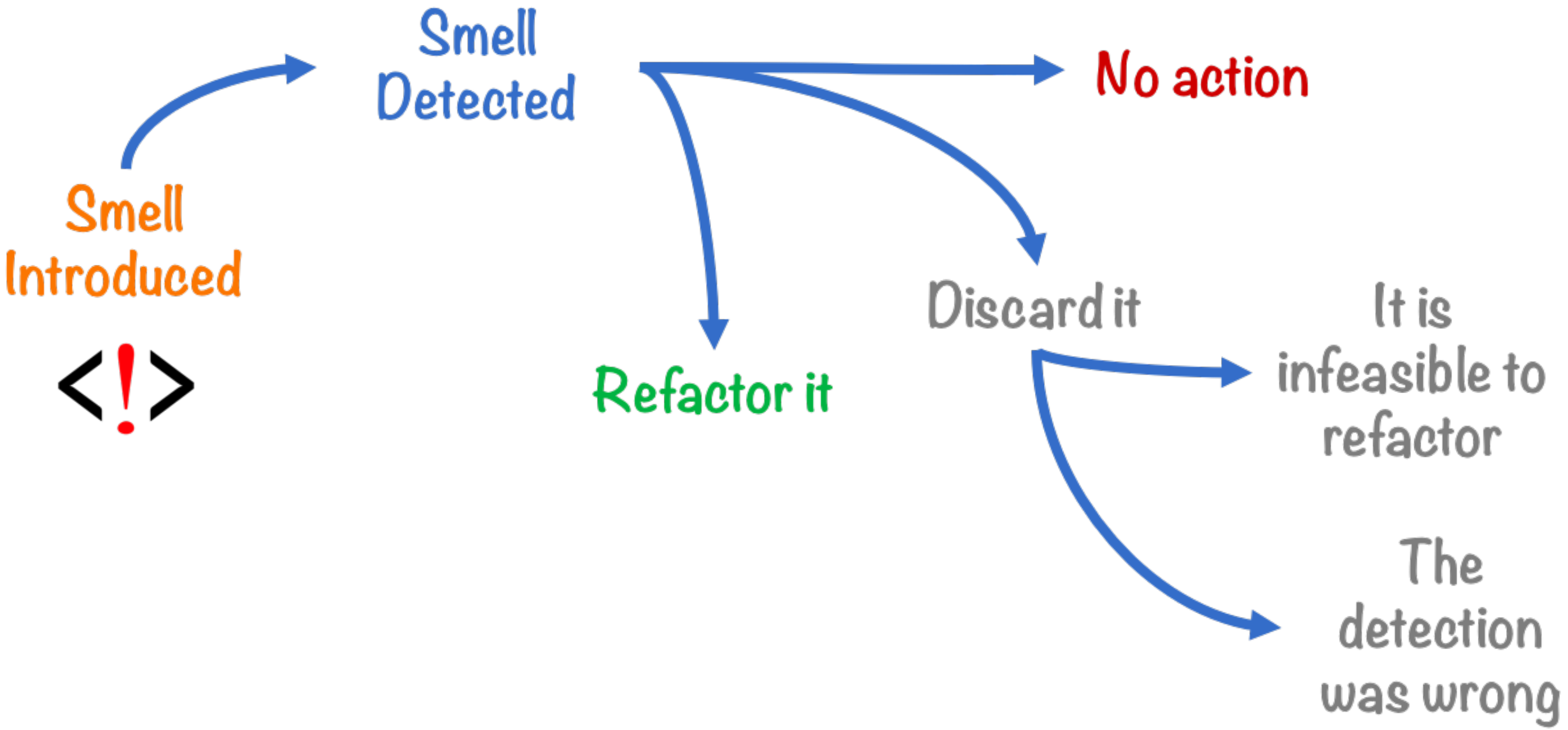
Bloaters

Bloaters are code, methods and classes that have increased to such gargantuan proportions that they are hard to work with. Usually these smells do not crop up right away, rather they accumulate over time as the program evolves (and especially when nobody makes an effort to eradicate them).

- Long Method
- Large Class
- Primitive Obsession
- Long Parameter List
- Data Clumps

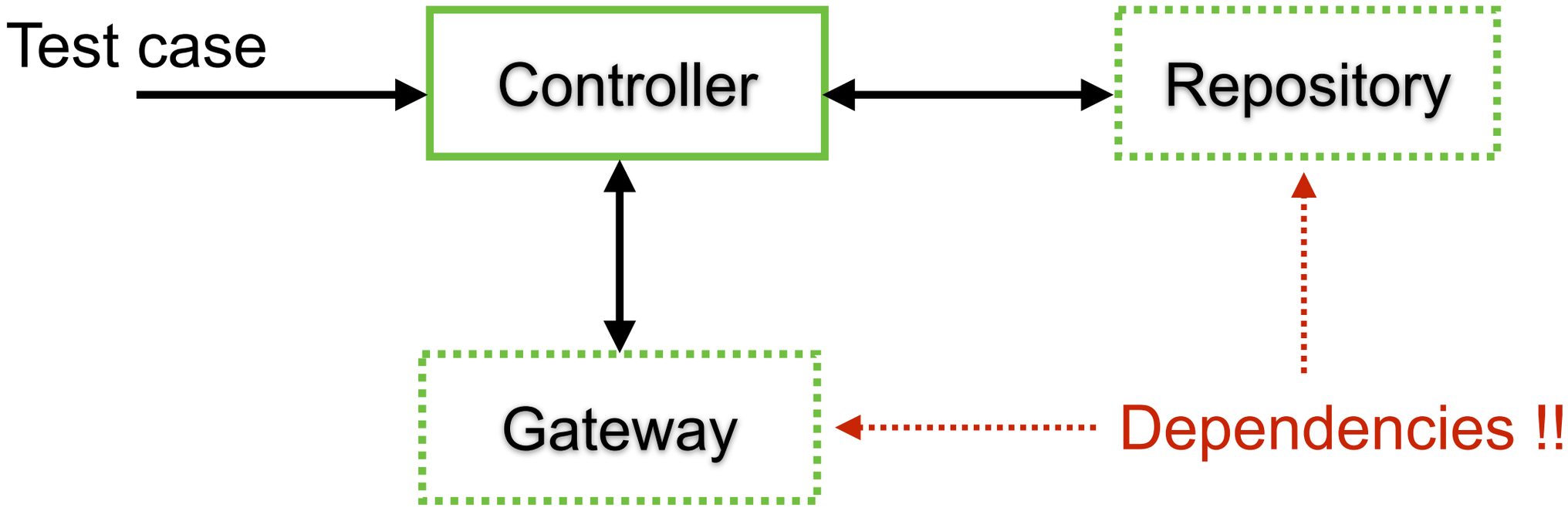
https://sourcemaking.com/refactoring/smells

Life-cycle of a smell

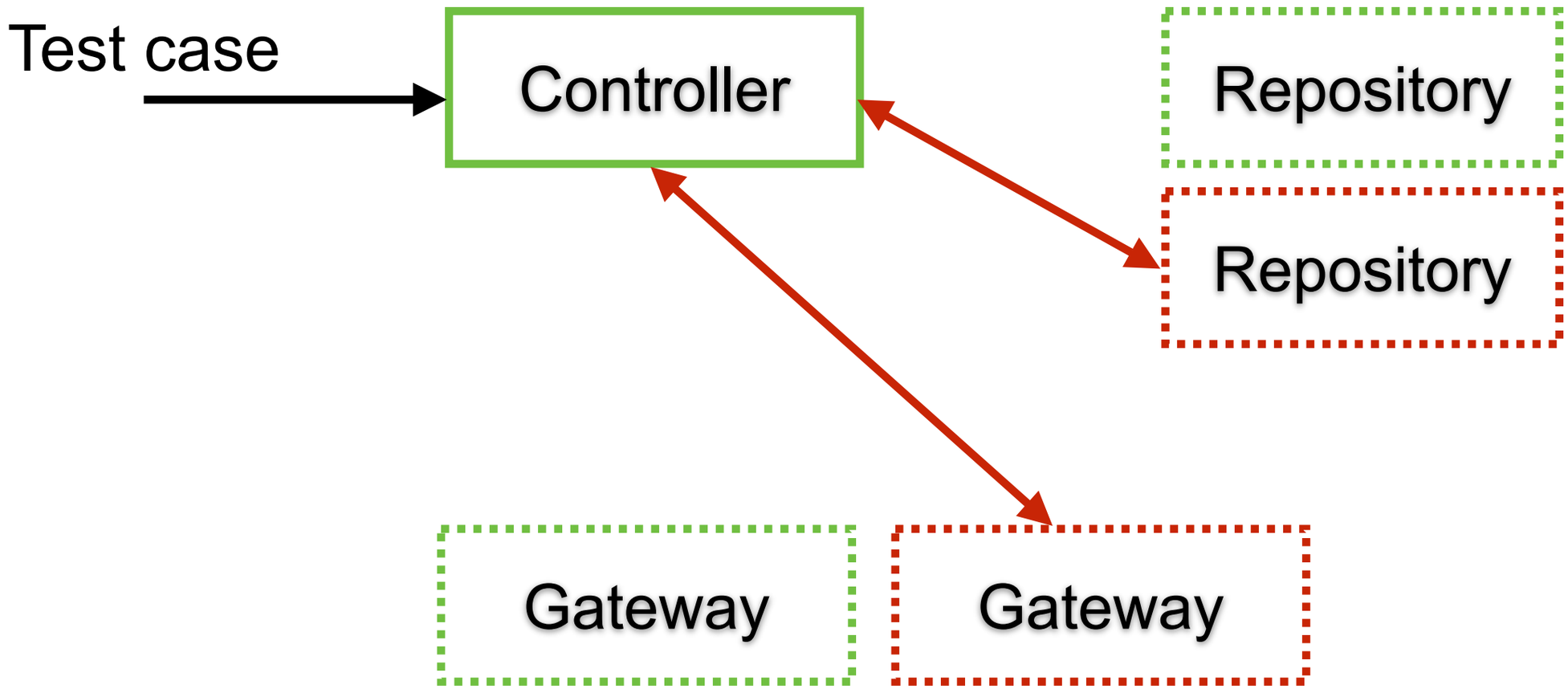


Manage your dependencies

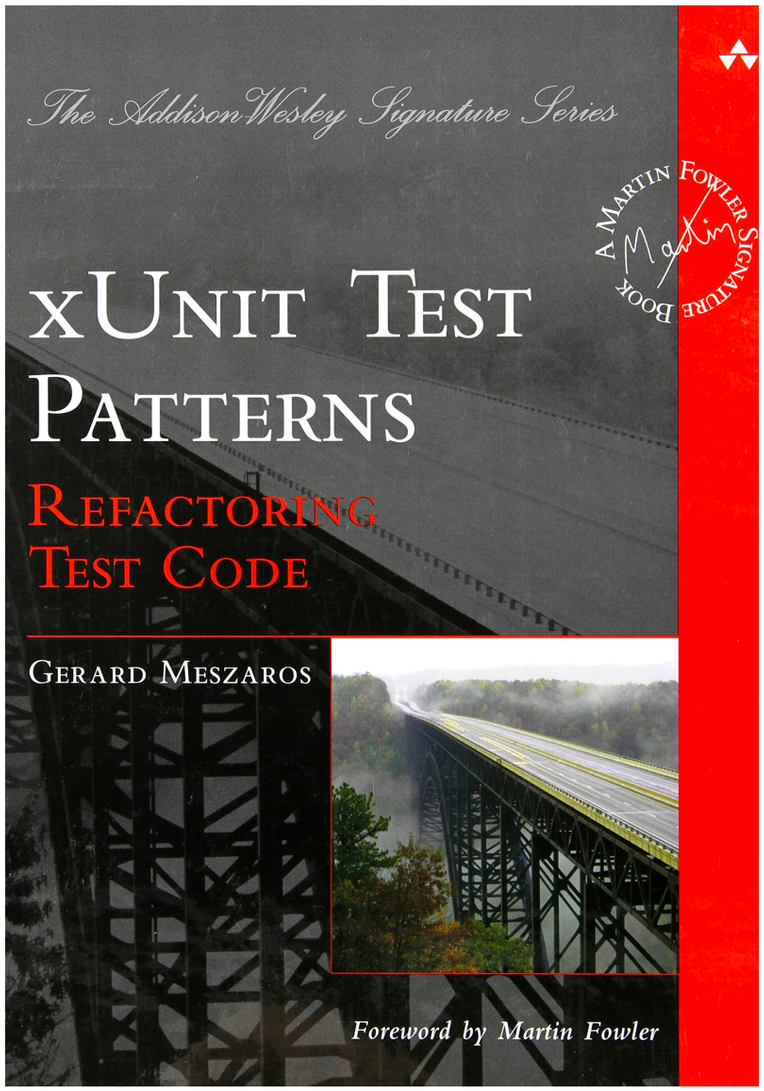
How to test Controller ?



Working with dependencies

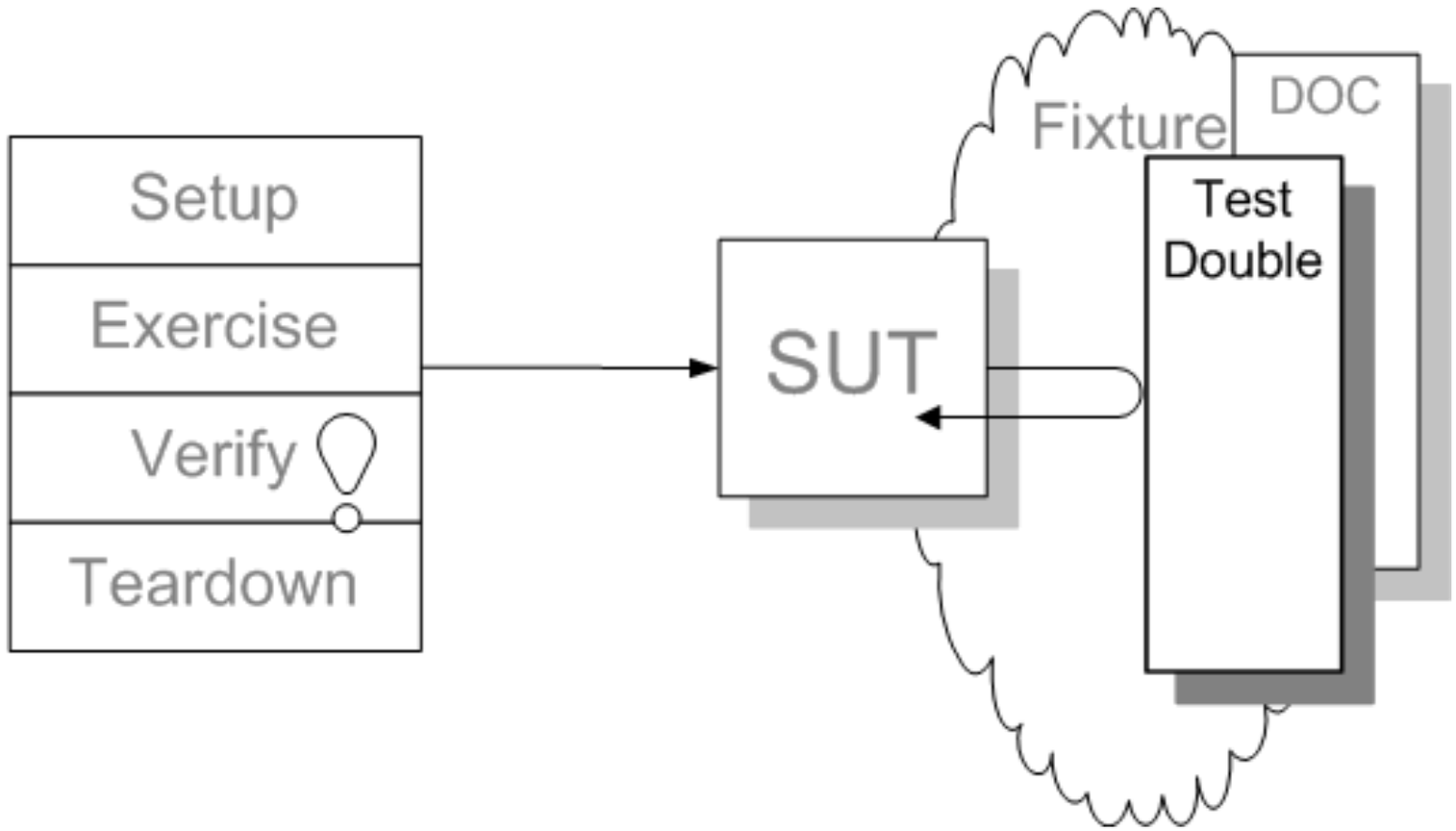


Test double ?



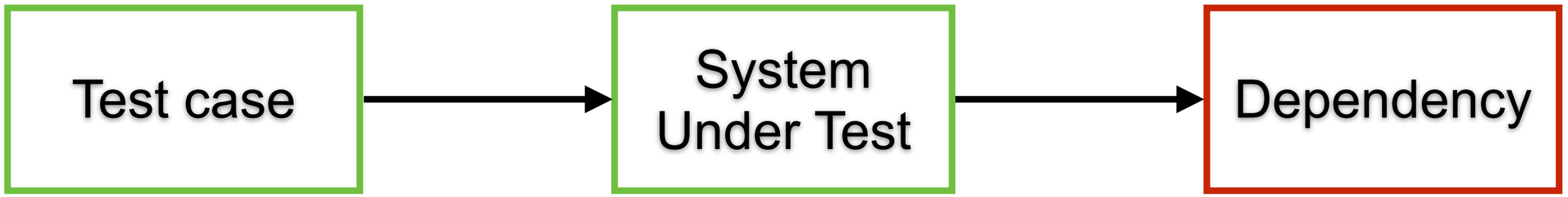
http://xunitpatterns.com

Test double ?

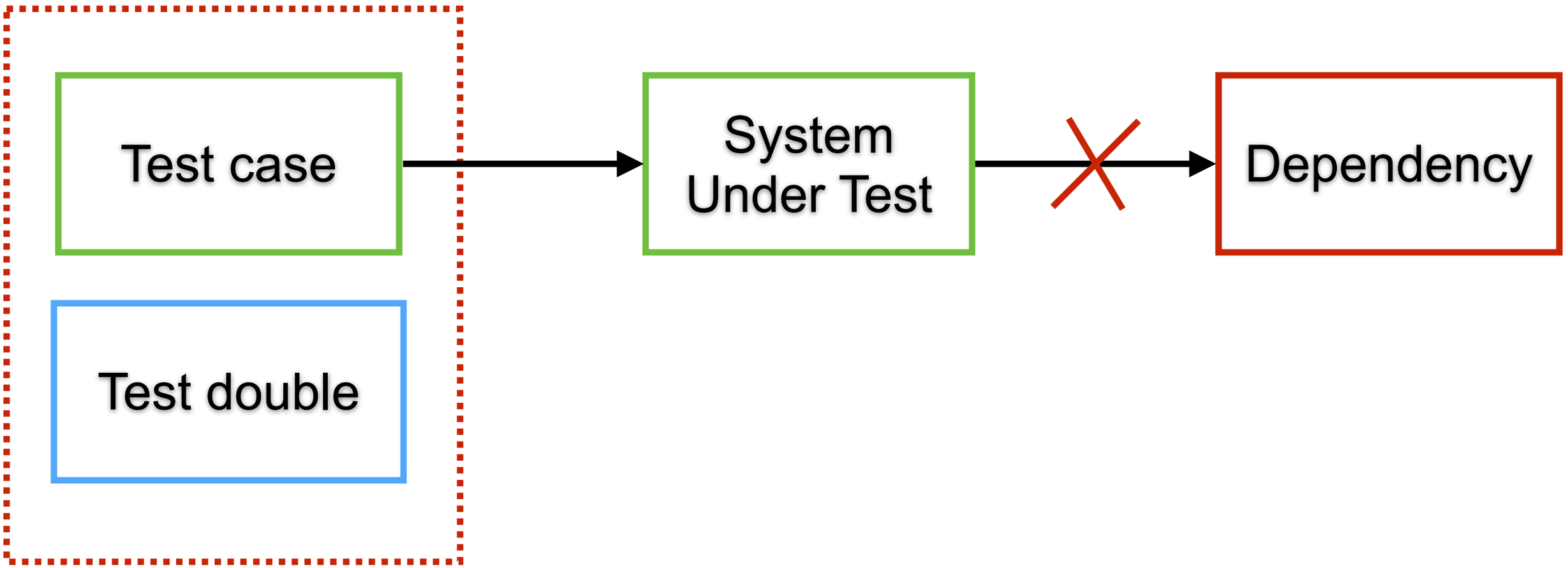


http://xunitpatterns.com/Test%20Double.html

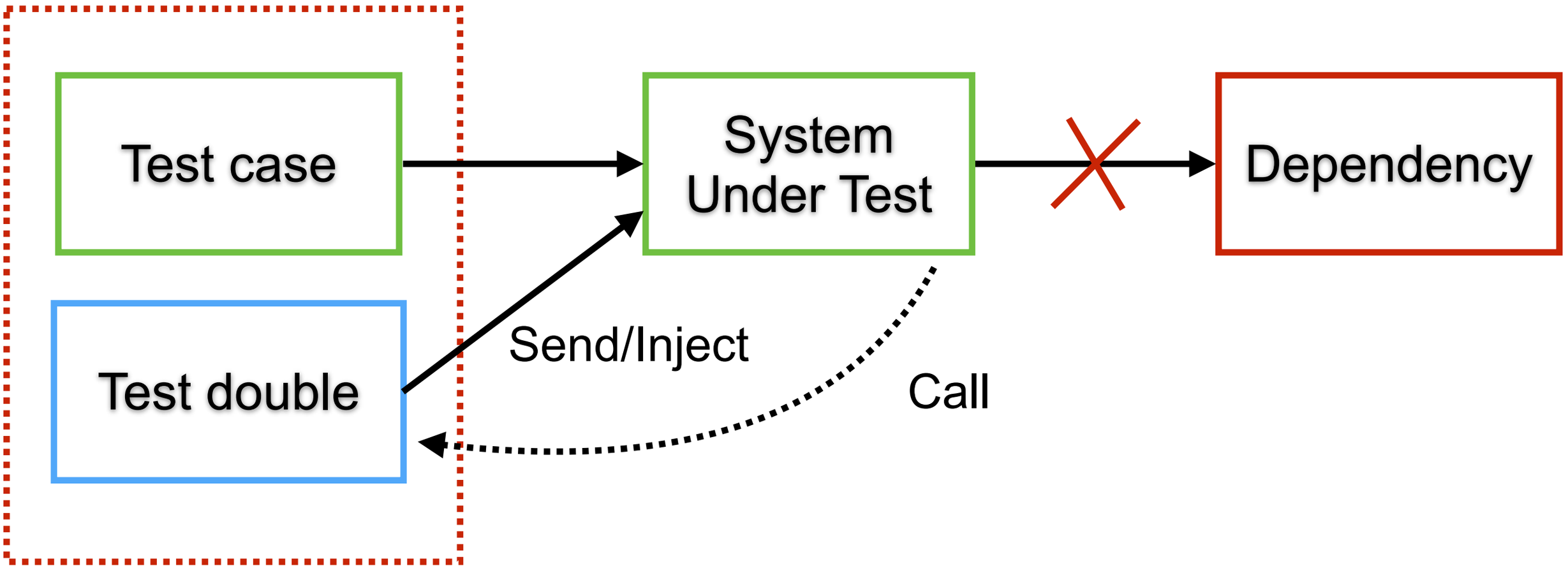
Test double ?



Create test double



Send/inject test double to SUT



Test double ?

Dummy object	Stub	Spy	Mock object	Fake object

Dummy object

Passed around but never actually used
Used to fill parameter lists

Fake

Working with implementation but take some shortcut
Not suitable for production

E.g. In-memory database, Fake API server

Stub

Provide answers to calls made during the test

Spy

Like stub
Record some information based on how its called

E.g. how many message it was sent via email service

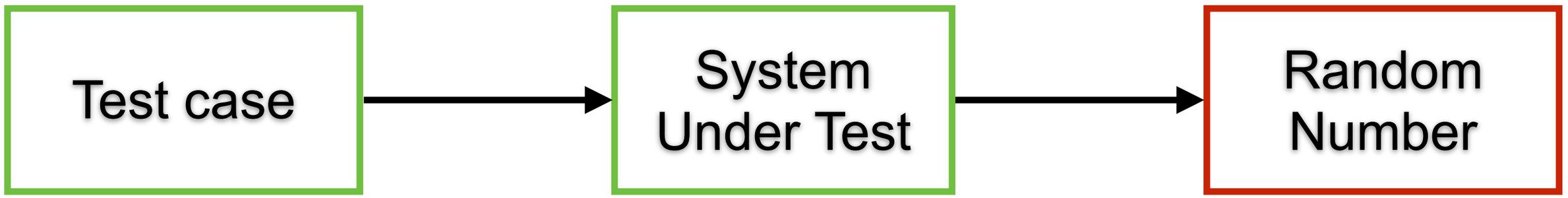
Mock object

Pre-programmed with expectations with spec

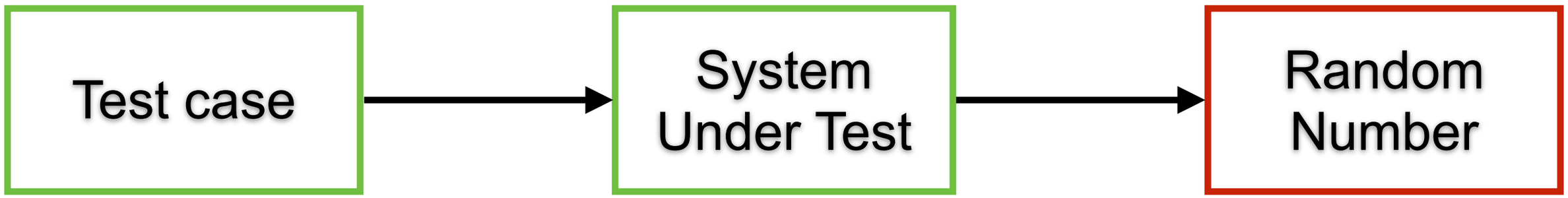
Mock object can throw an exception if receive a call that don’t expect

Workshop

Workshop

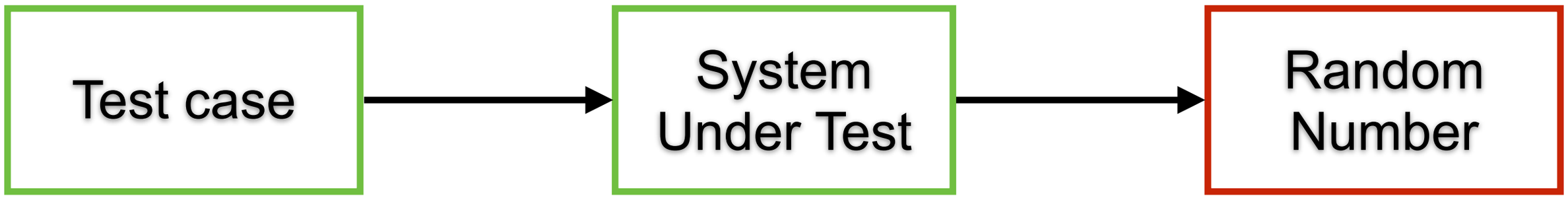


Workshop



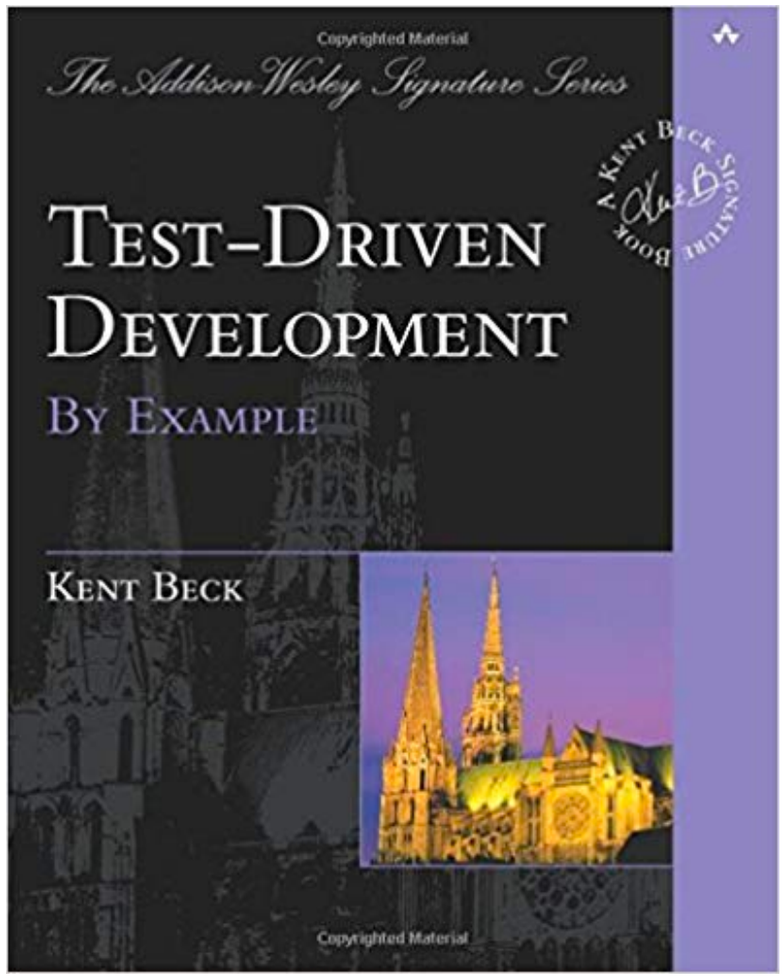
Test random number = 5 ?

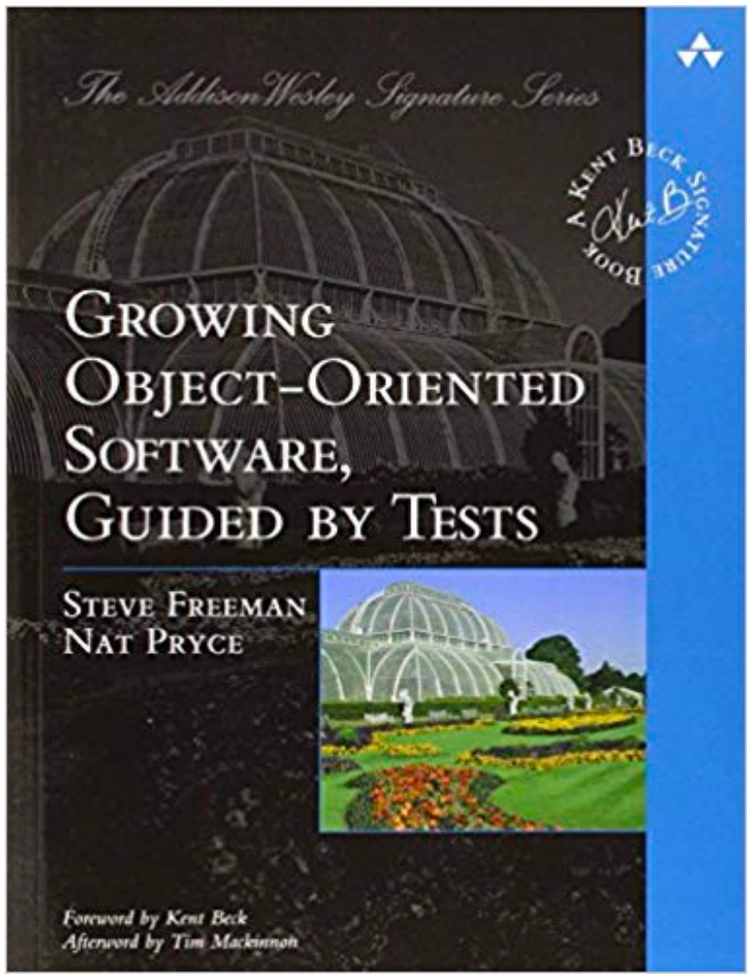
Workshop

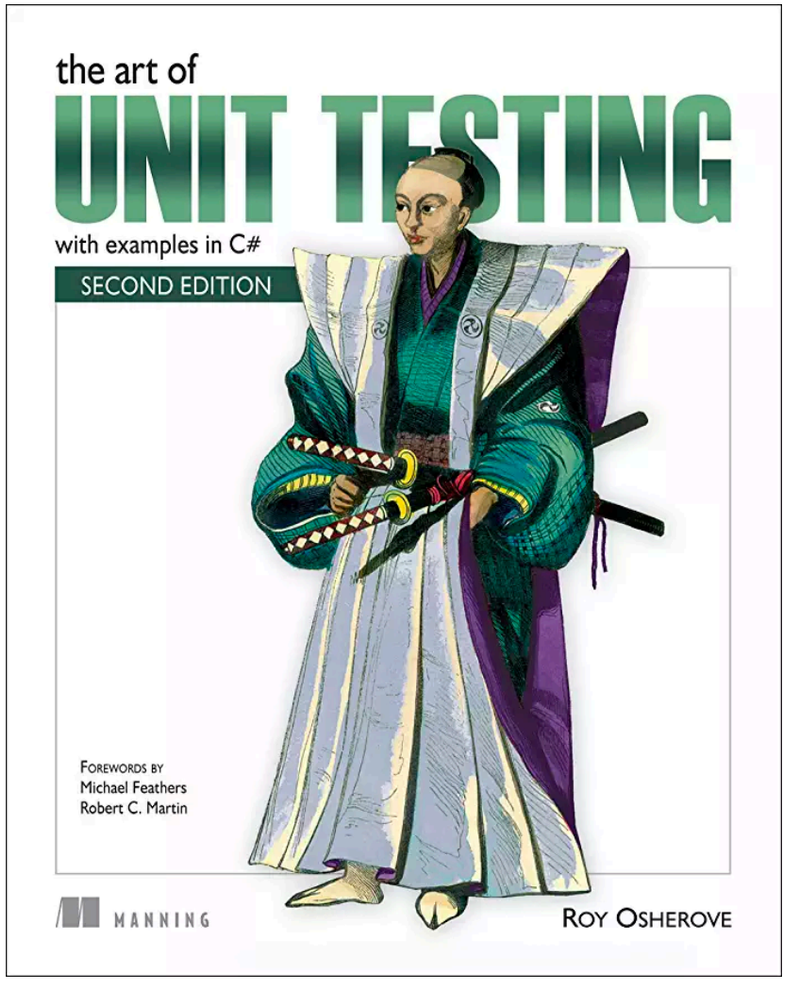


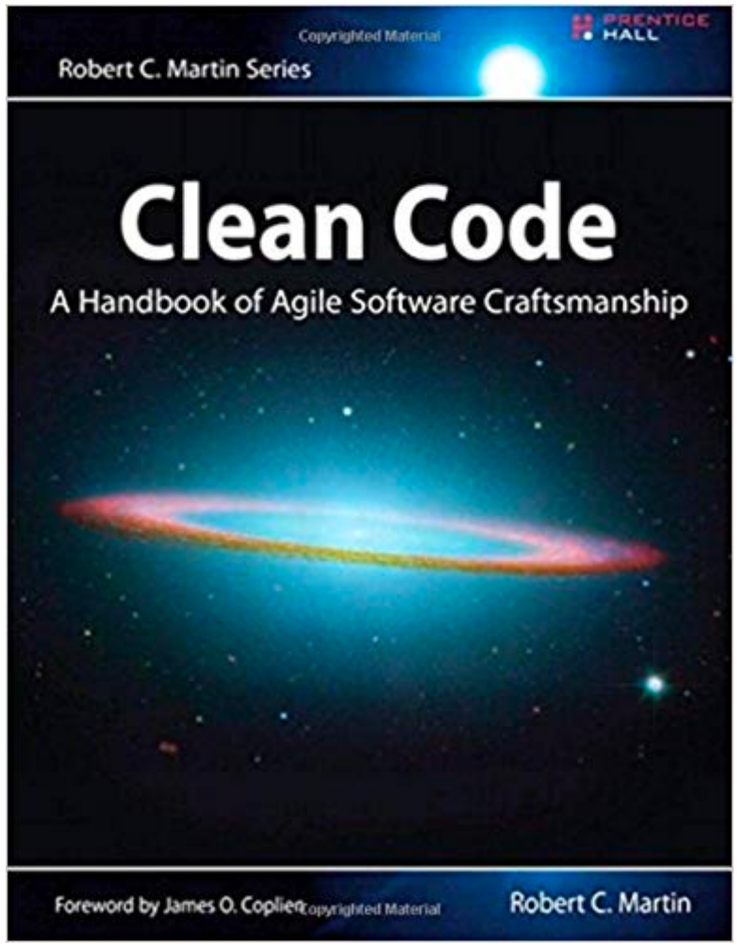
Test random number must called 1 time ?

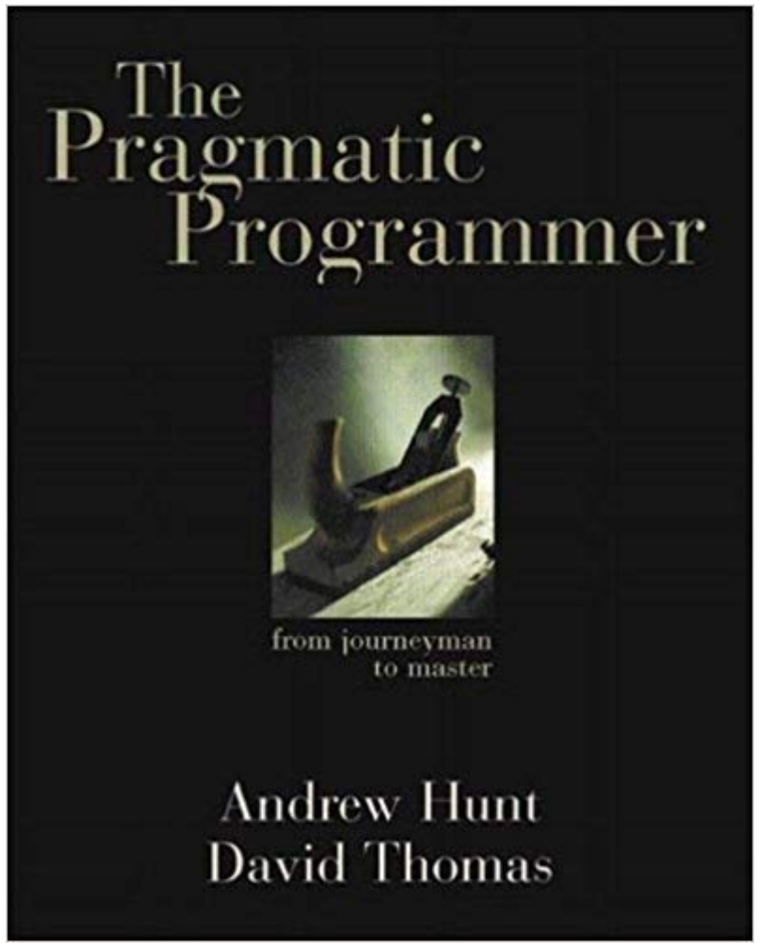
Books

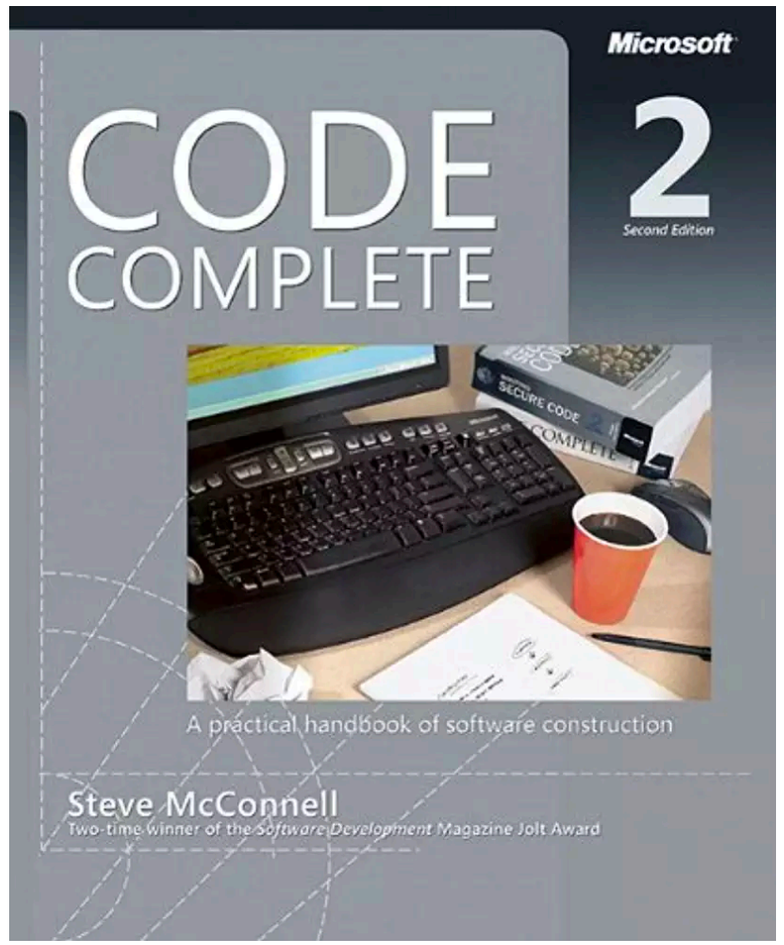


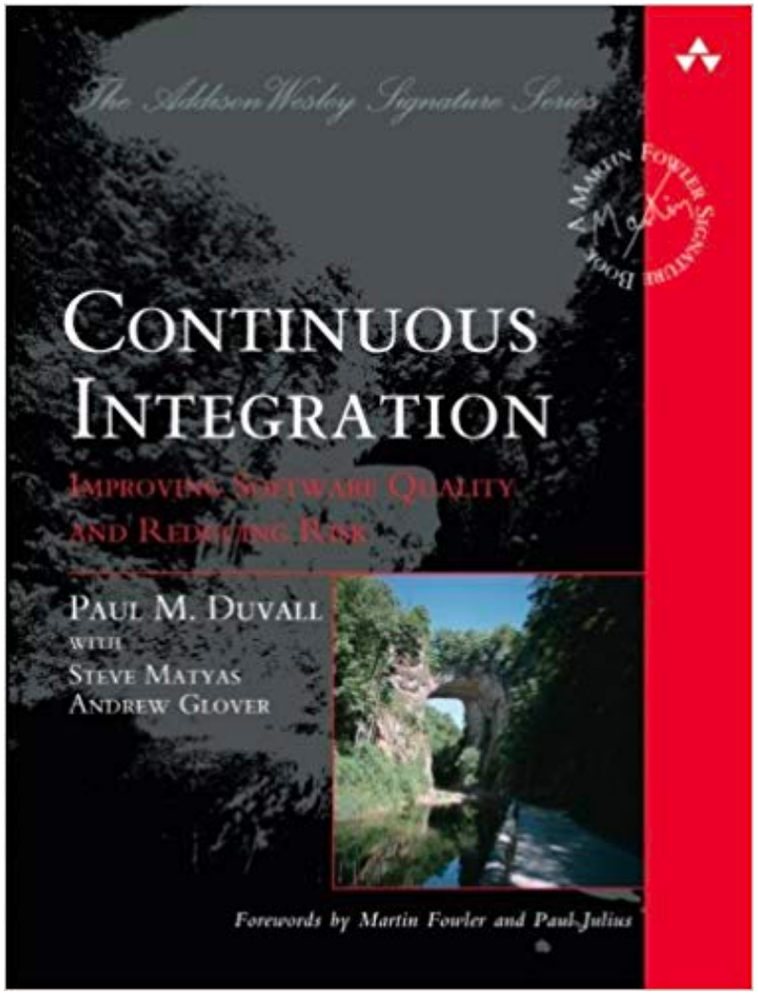


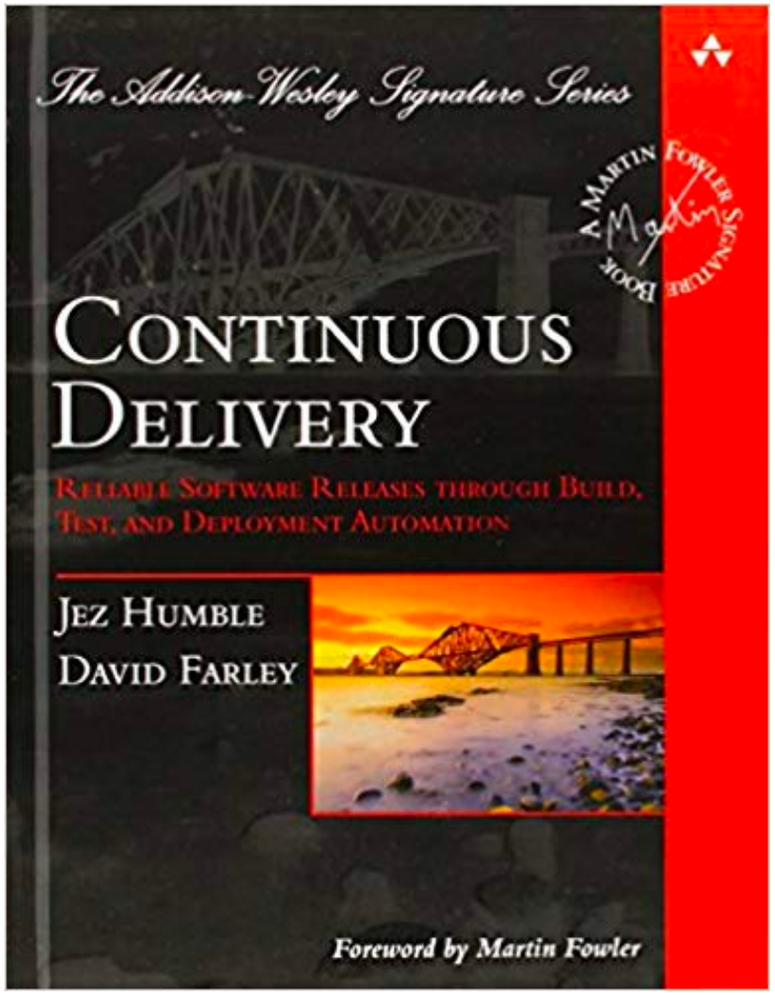












Module #3
Software Engineering Practices