สยามชำนาญกิจ
SIAM CHAMNAN KIT

# Automated Testing

# Automated Testing

# Software Delivery


https://martinfowler.com/articles/practical-test-pyramid.html

# Technical excellence








TECHNICAL
EXCELLENCE




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

SPECIFICATION BY EXAMPLE

CONTINUOUS INTEGRATION

CONTINUOUS DELIVERY

TEST AUTOMATION

TECHNICAL EXCELLENCE

ACCEPTANCE TESTING

ARCHITECTURE & DESIGN


CLEAN CODE

THINKING ABOUT TESTING


TEST-DRIVEN DEVELOPMENT

UNIT TESTING

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

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

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

SPECIFICATION BY EXAMPLE

CONTINUOUS INTEGRATION

CONTINUOUS DELIVERY

TEST AUTOMATION

TECHNICAL EXCELLENCE

ARCHITECTURE & DESIGN

ACCEPTANCE TESTING

THINKING ABOUT TESTING


TEST-DRIVEN DEVELOPMENT

CODE

CLEAN CODE

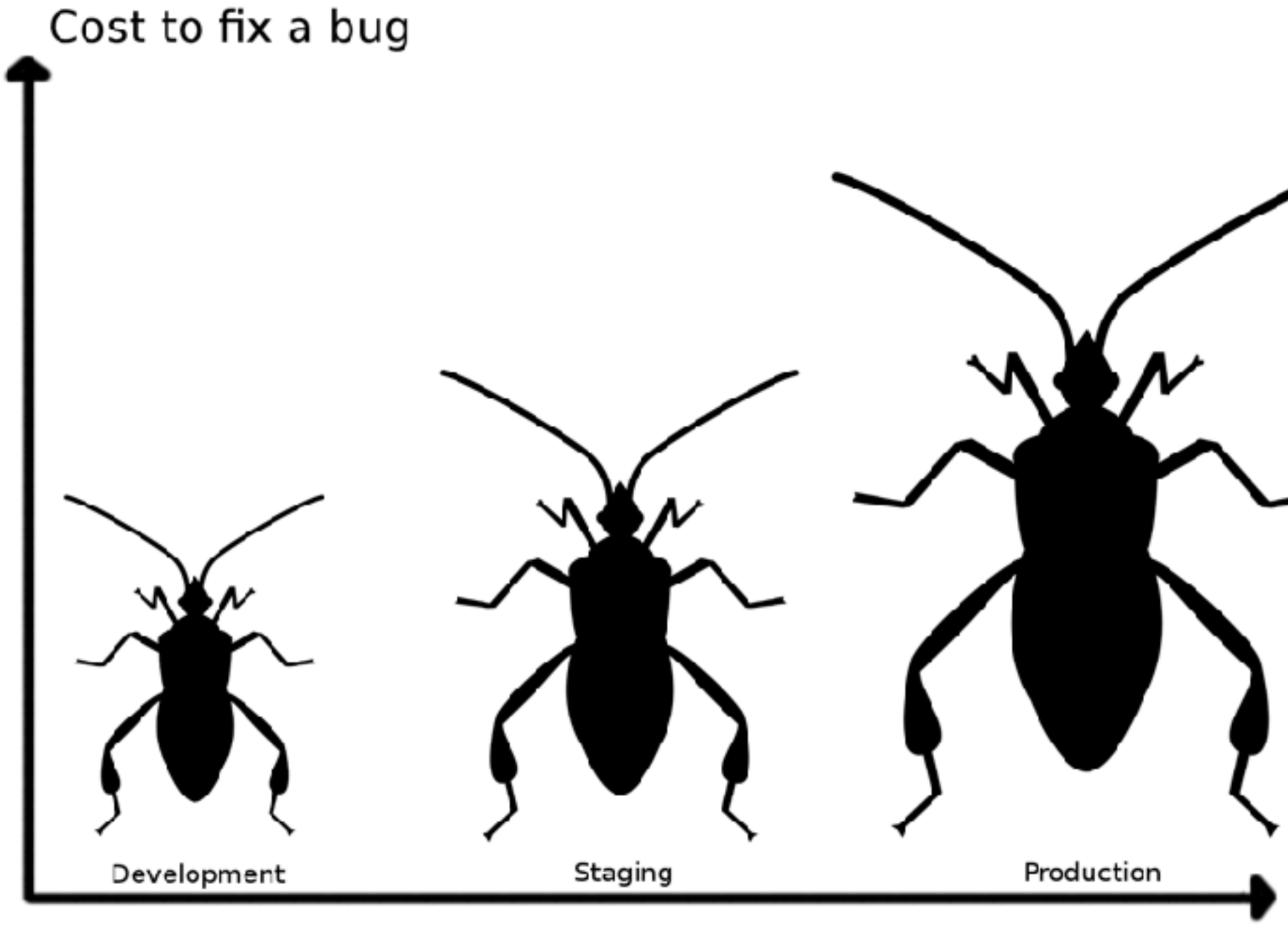
UNIT TESTING

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

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

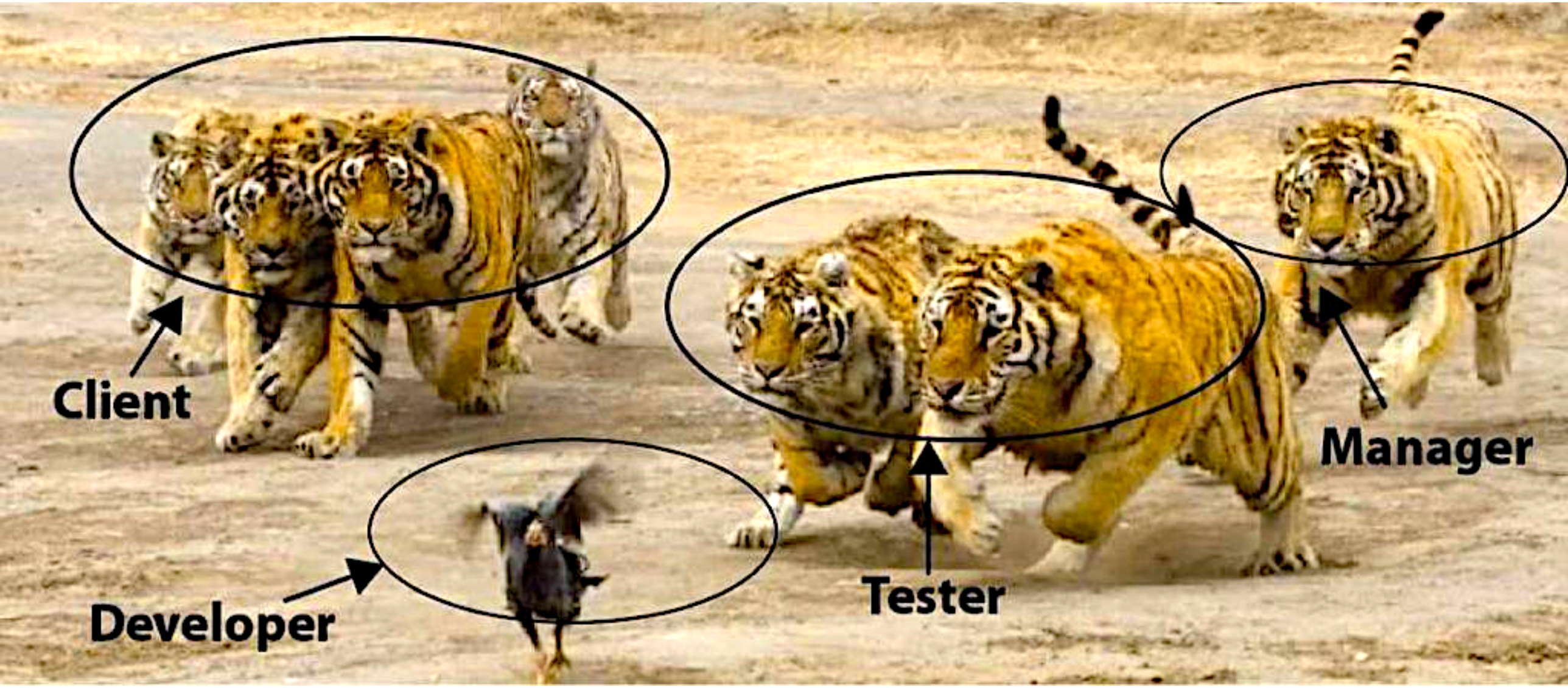
Edsger W. Dijkstra, 1970, Notes on Structured Programming

# Start with Why …

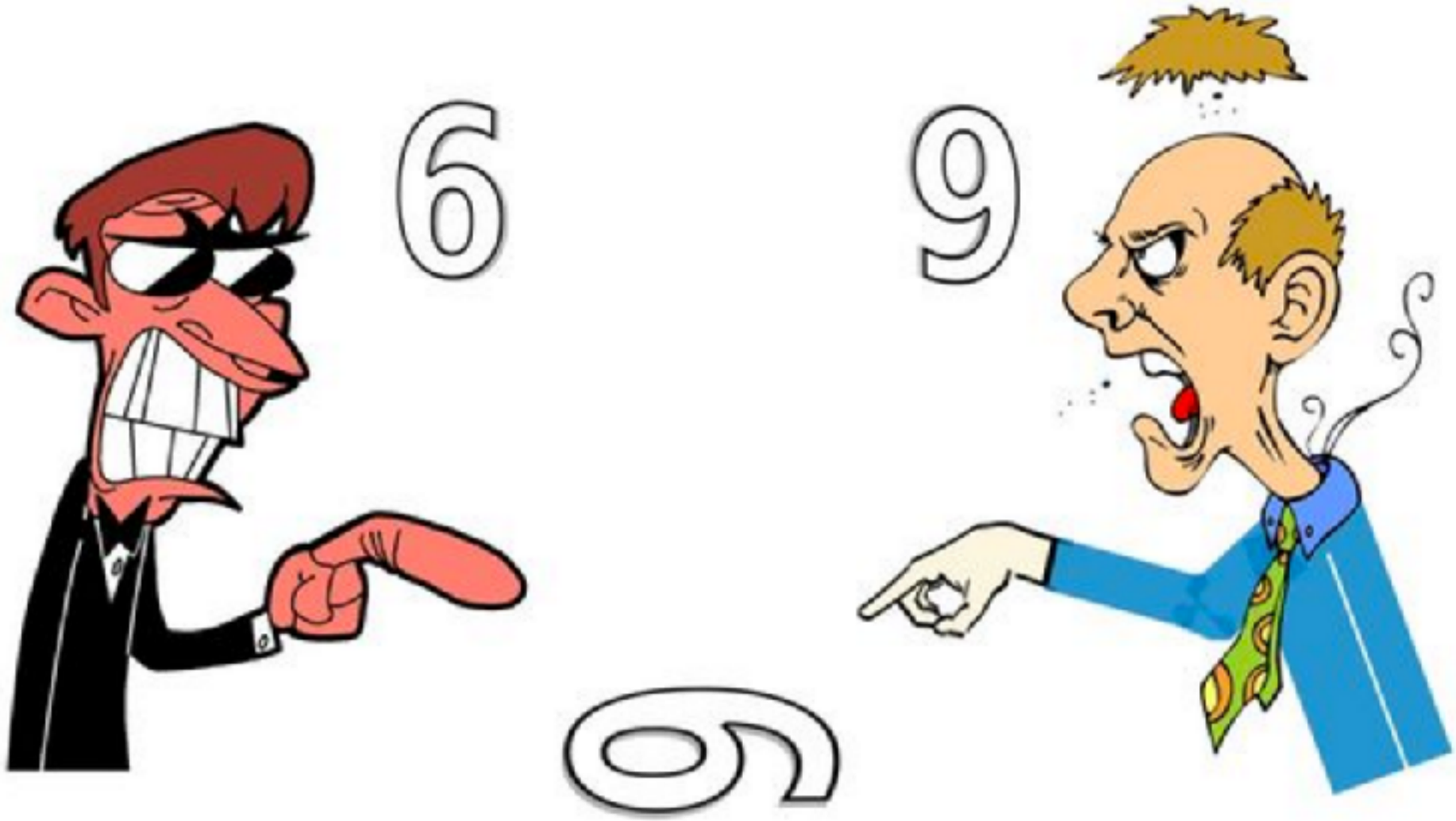


# DDD

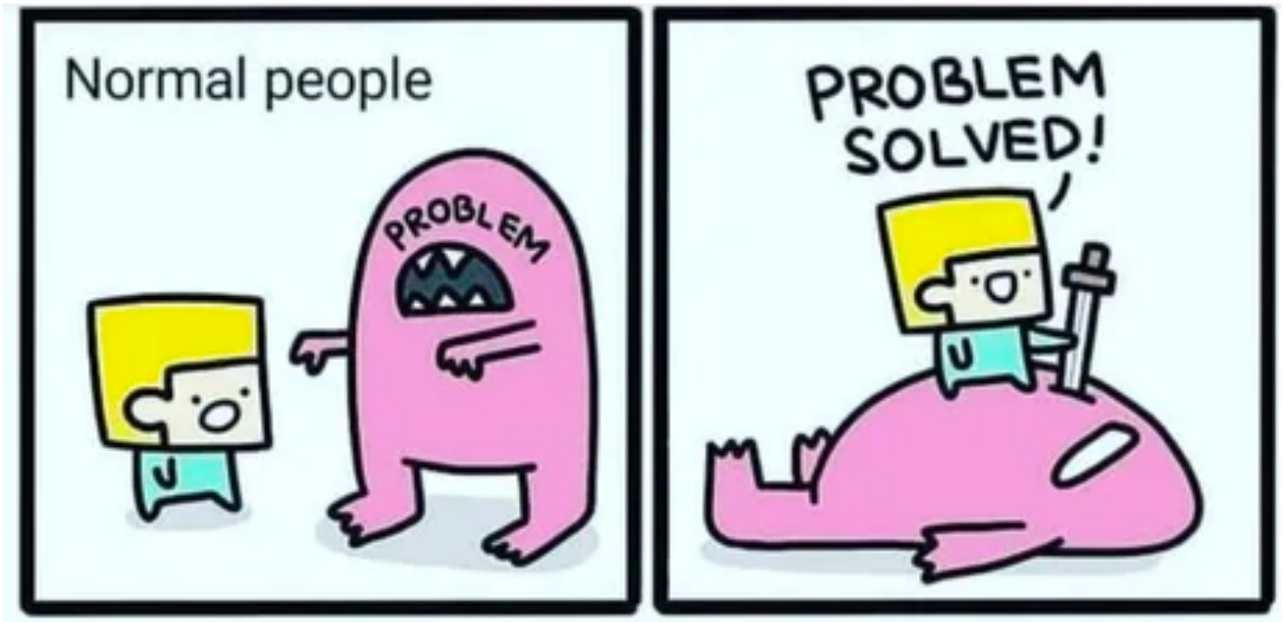
# **D**eadline **D**riven **D**evelopment

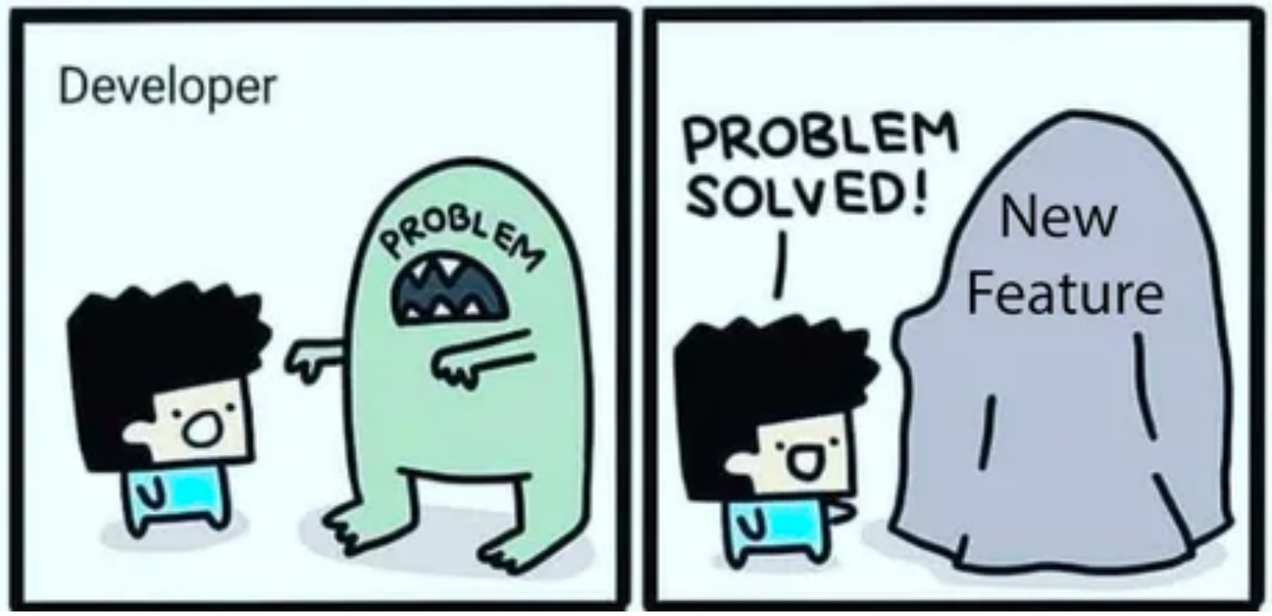


# Developer vs Tester



# Every time a developer changing the code !!



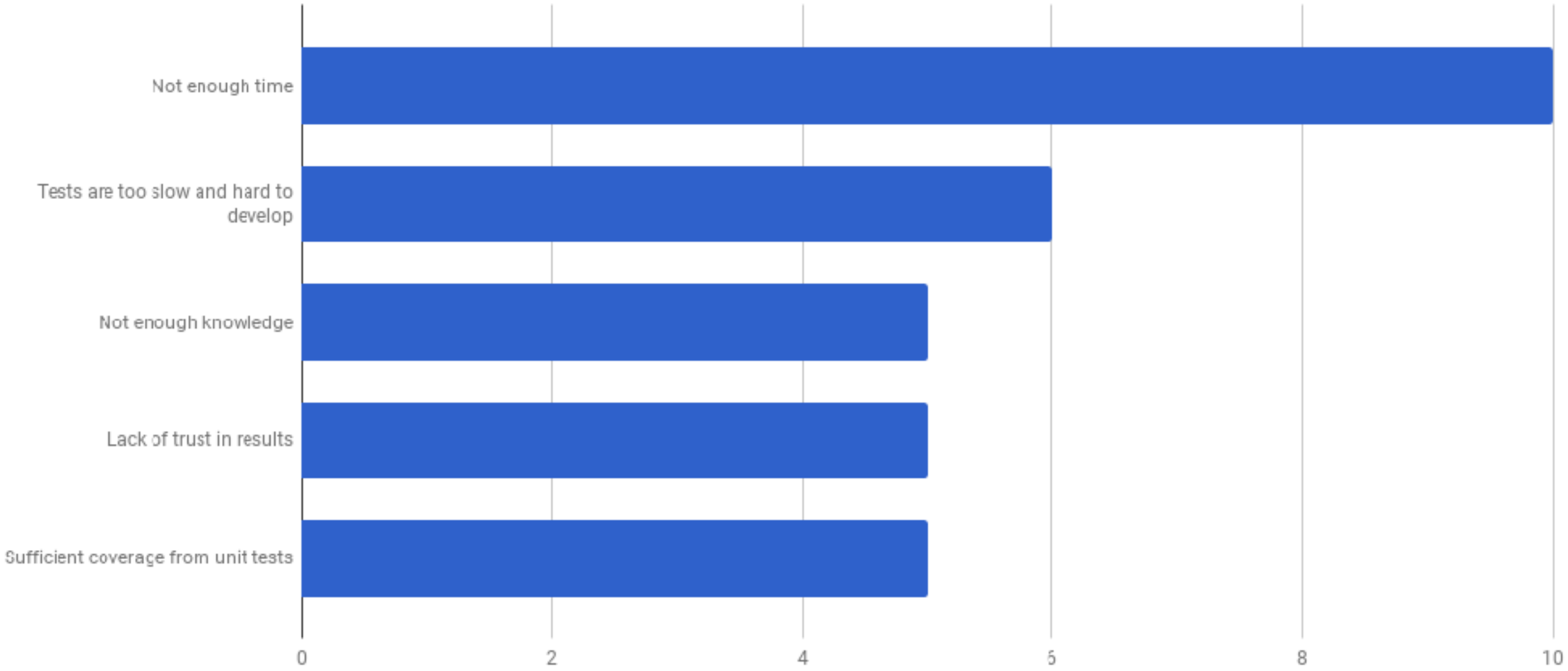
# Why we need to test ?

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 !!**

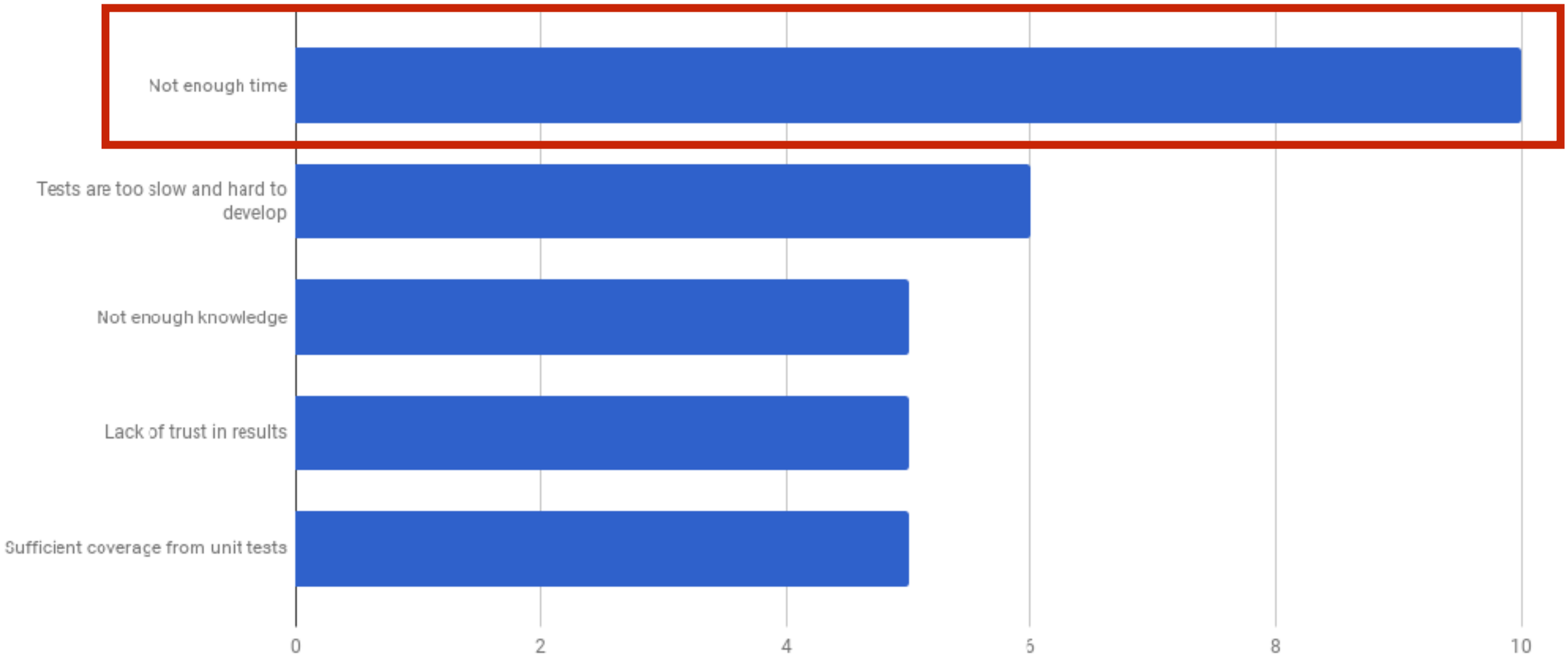
# Why not write automated tests ?




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

# Why not write automated tests ?

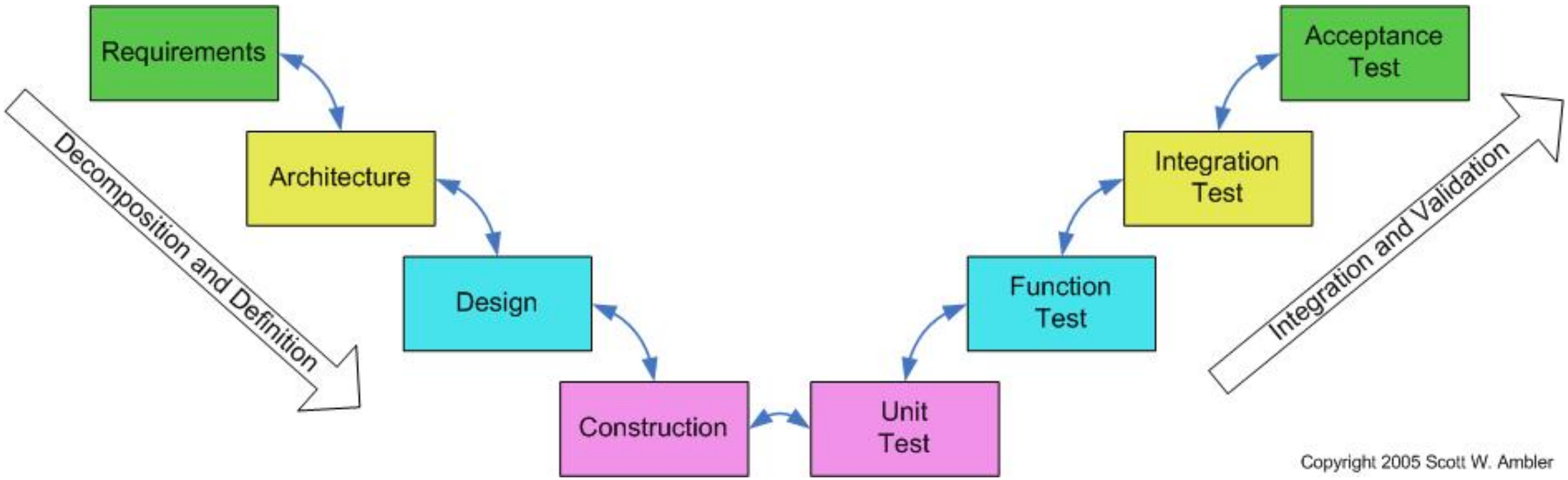
Not enough time !!!



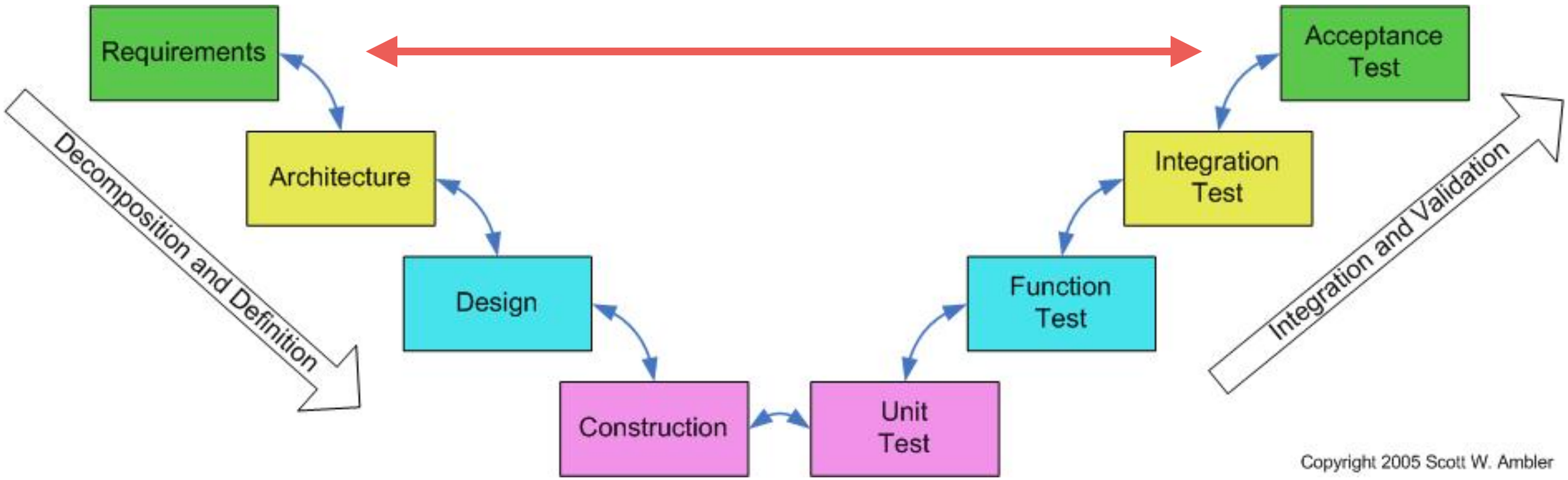

https://slack.engineering/android-ui-automation-part-1-building-trust-de3deb1c5995

# What kind of test should we write ?

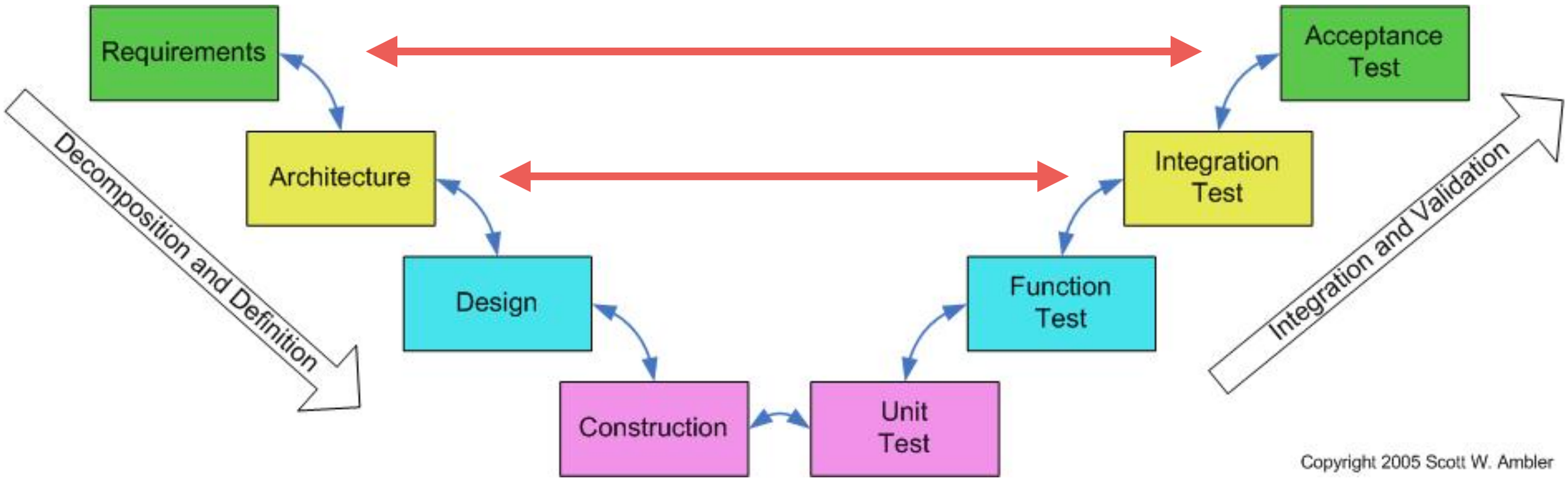
# V Model



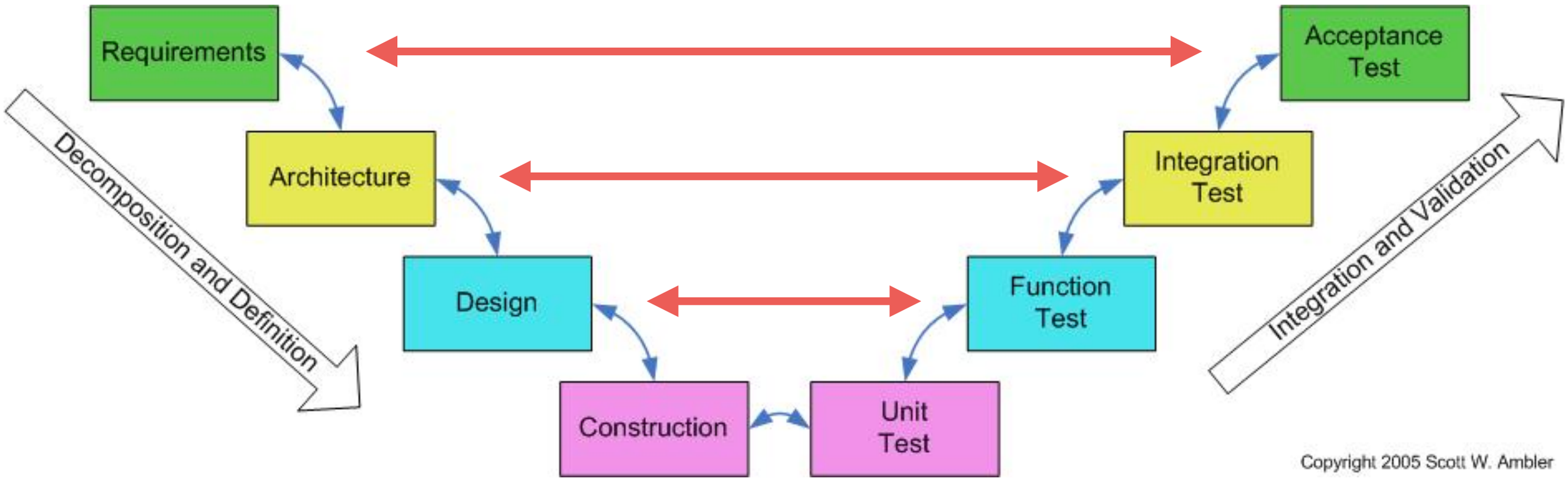
# V Model



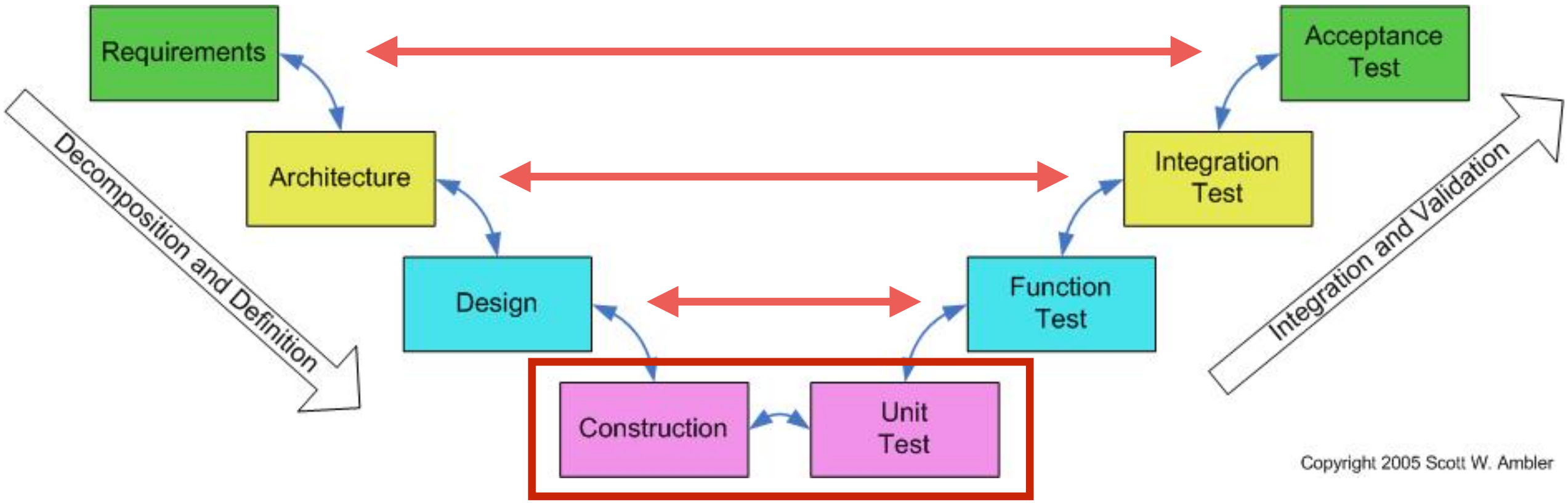
# V Model



# V Model

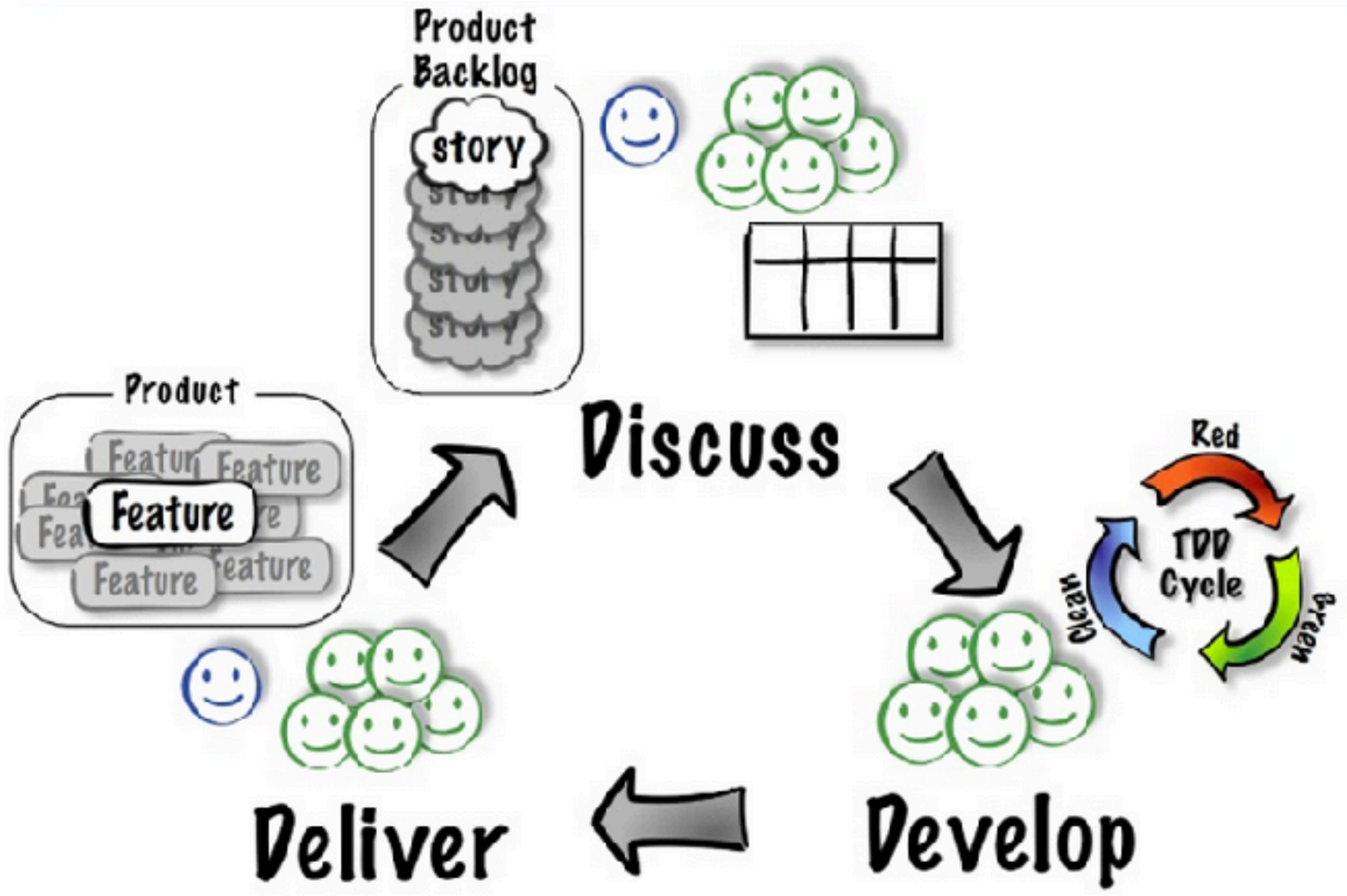


# V Model



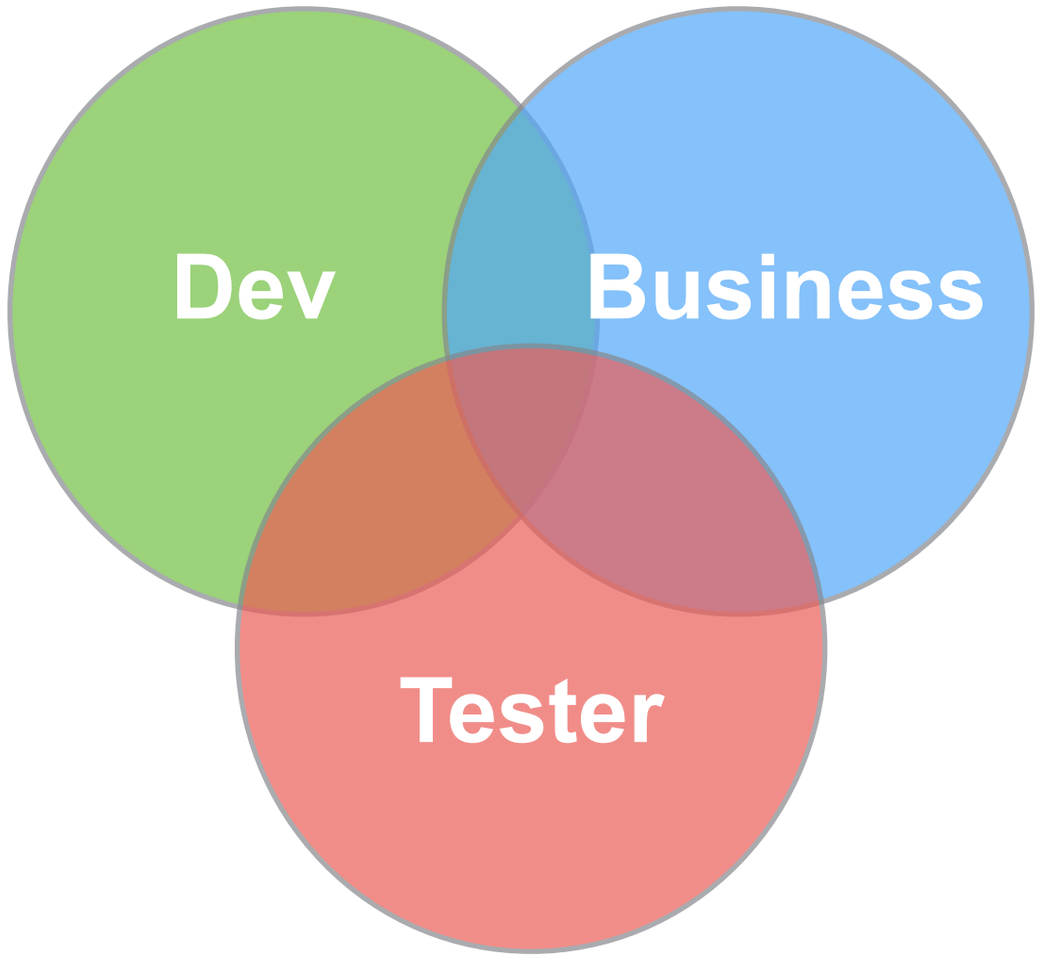
# THINK before coding

# Acceptance Test-Driven Development




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

# Acceptance Test-Driven Development



# Acceptance Tests

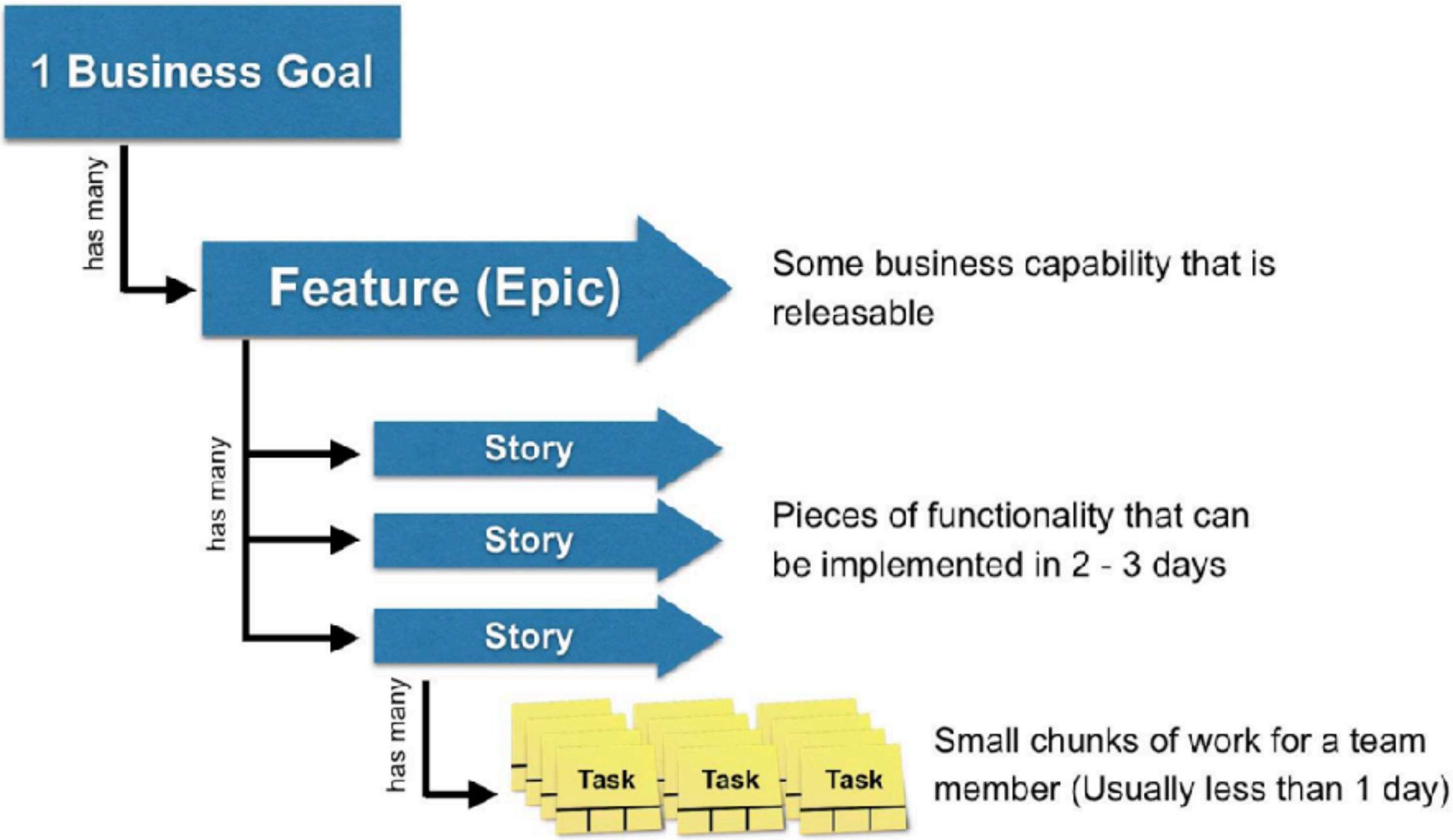
# =

# Business Criteria

# +

# Examples (data)

# Work break down



# Whole team approach

## Functional

Common functional expertise

System analysts

Developers

Testers

## Cross - Functional

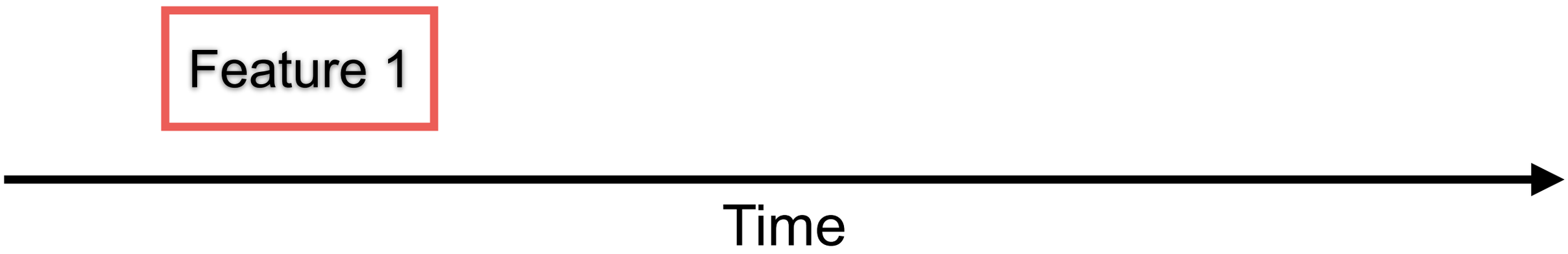
Representatives from the various functions

Development Team

# Key success factors

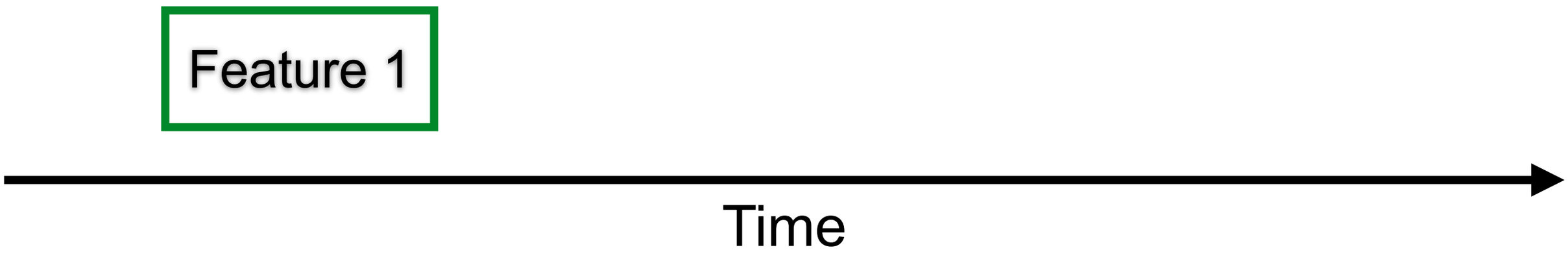
Whole team solve problems
Whole team thinks about **testing**
Whole team **committed to quality**
Everyone **collaborates**

# Iterative and incremental process



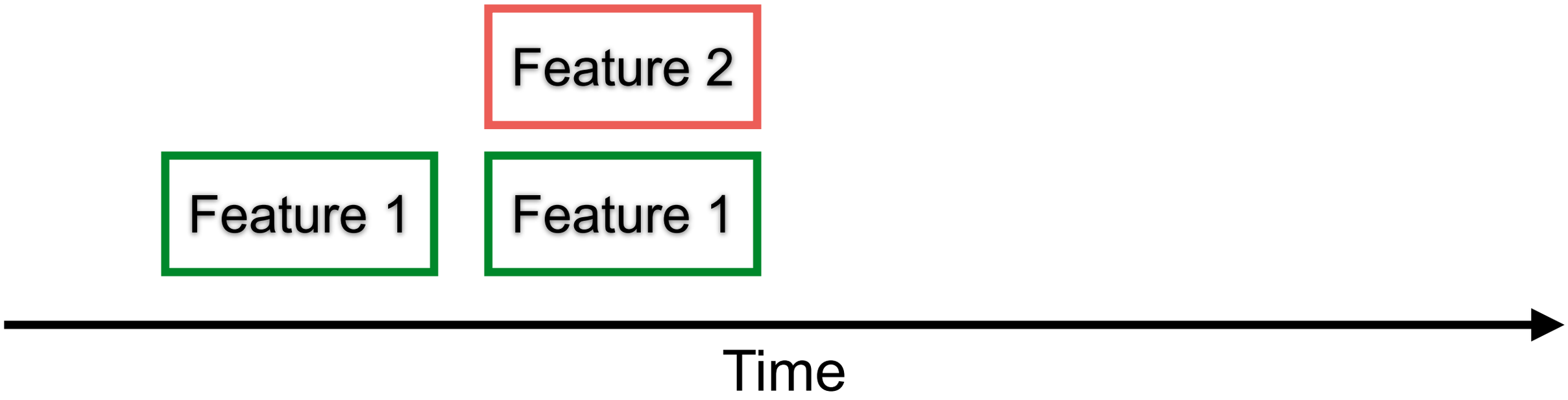
# Iterative and incremental process

Done = coded and tested



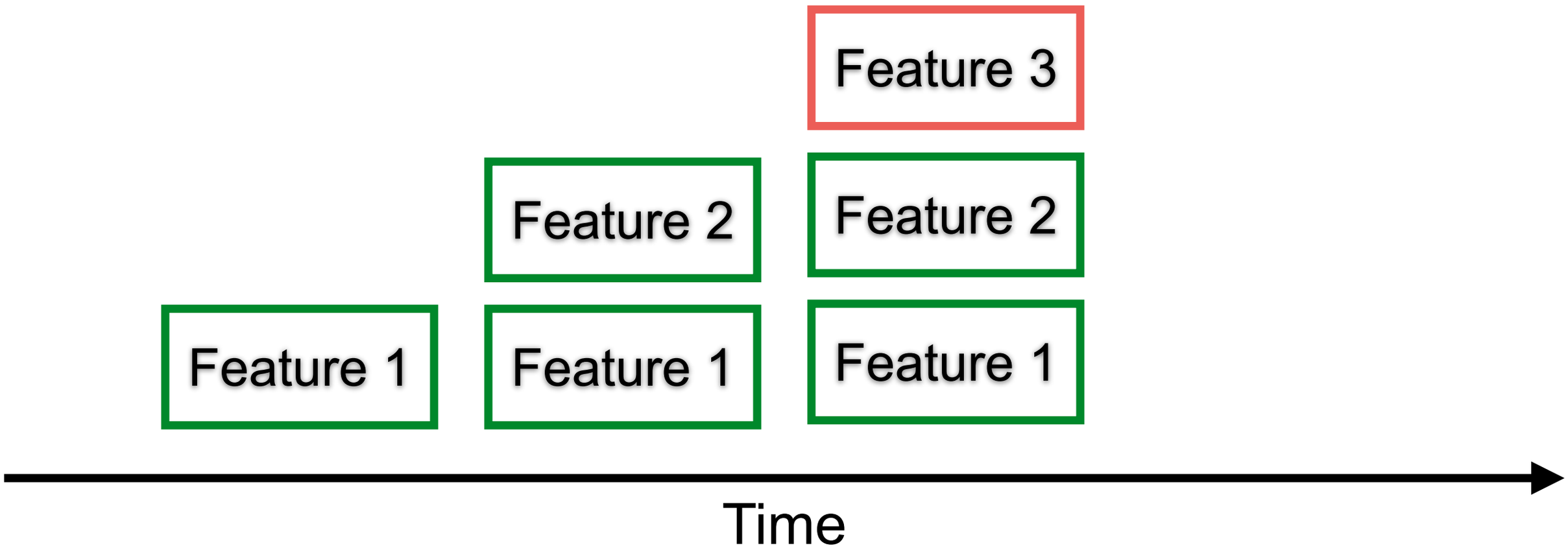
# Iterative and incremental process

Done = coded and tested



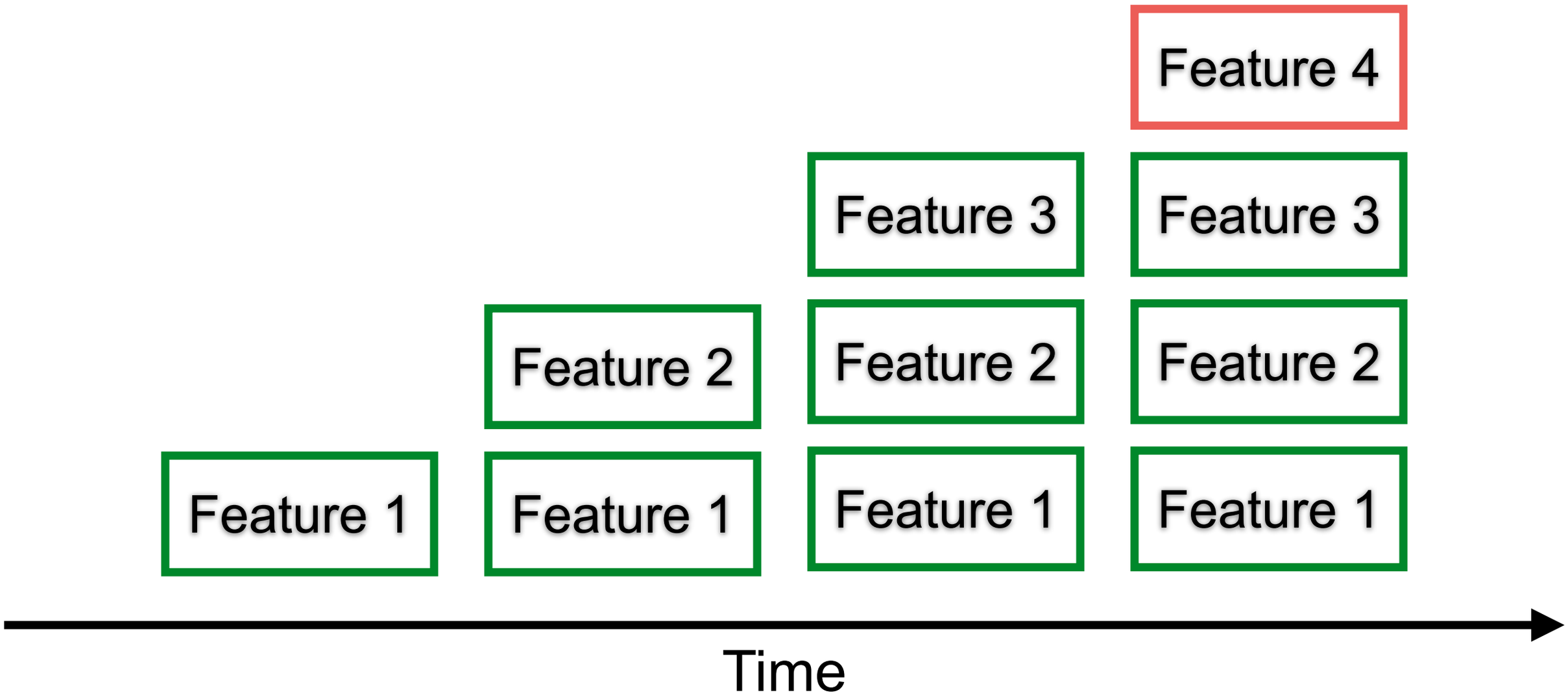
# Iterative and incremental process

Done = coded and tested



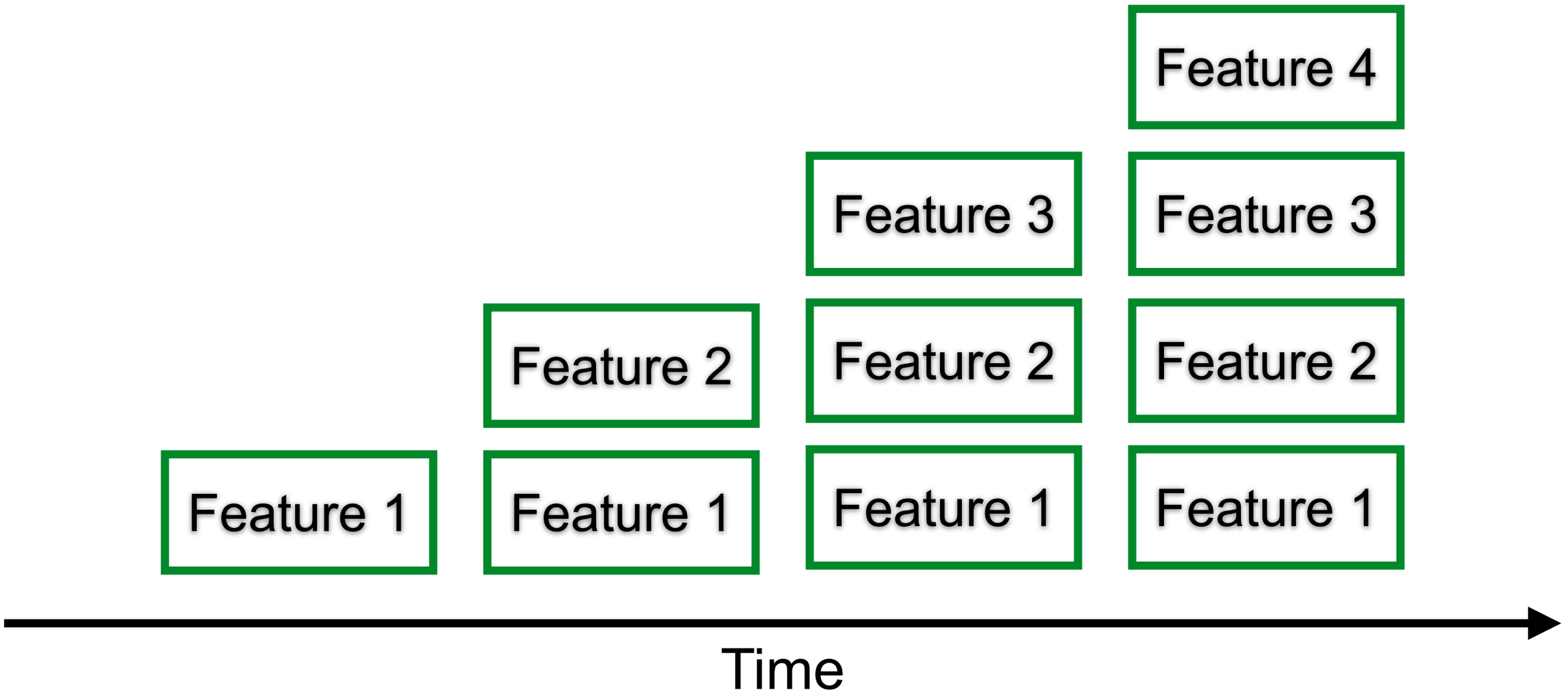
# Iterative and incremental process

Done = coded and tested



# Iterative and incremental process

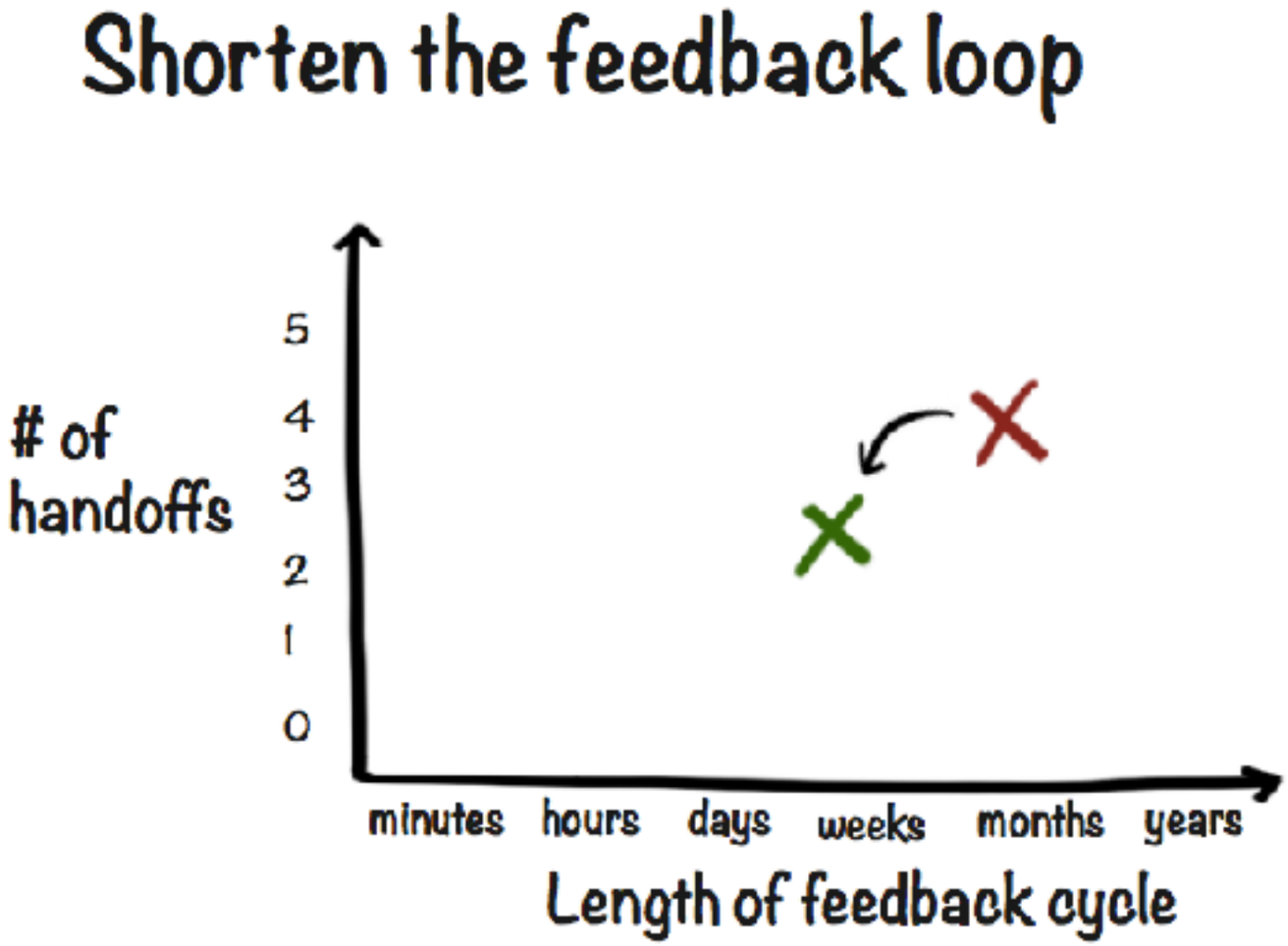
Done = coded and tested

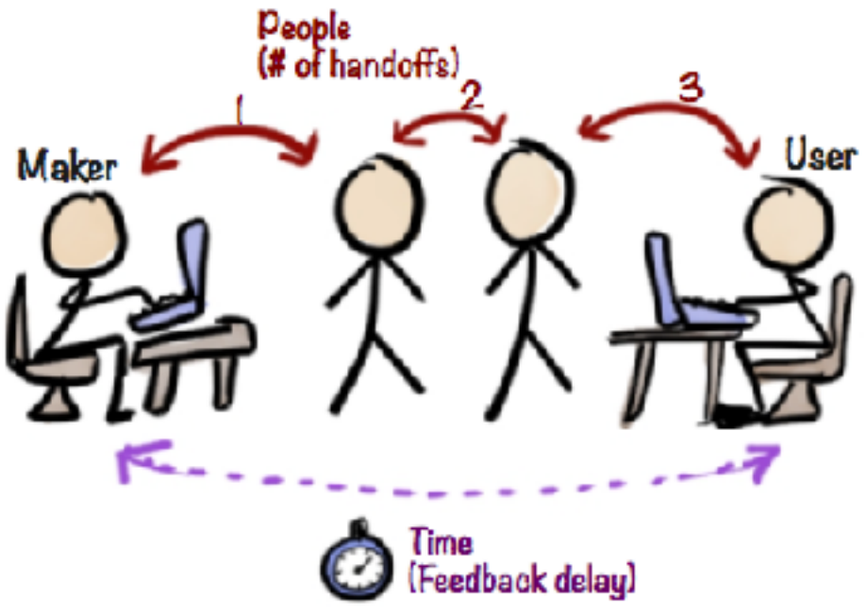


# Testing is activity

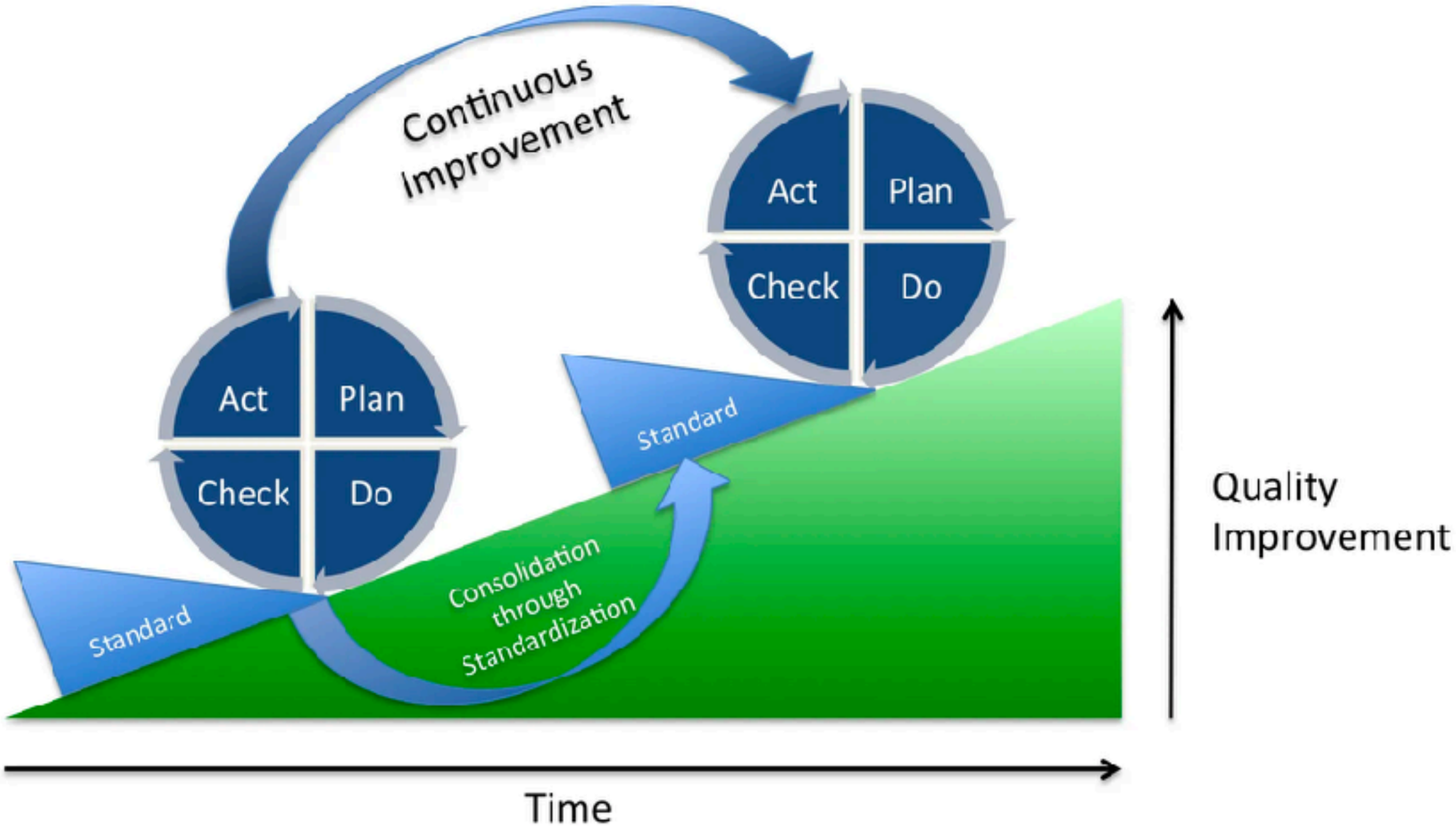


# Fast feedback loop

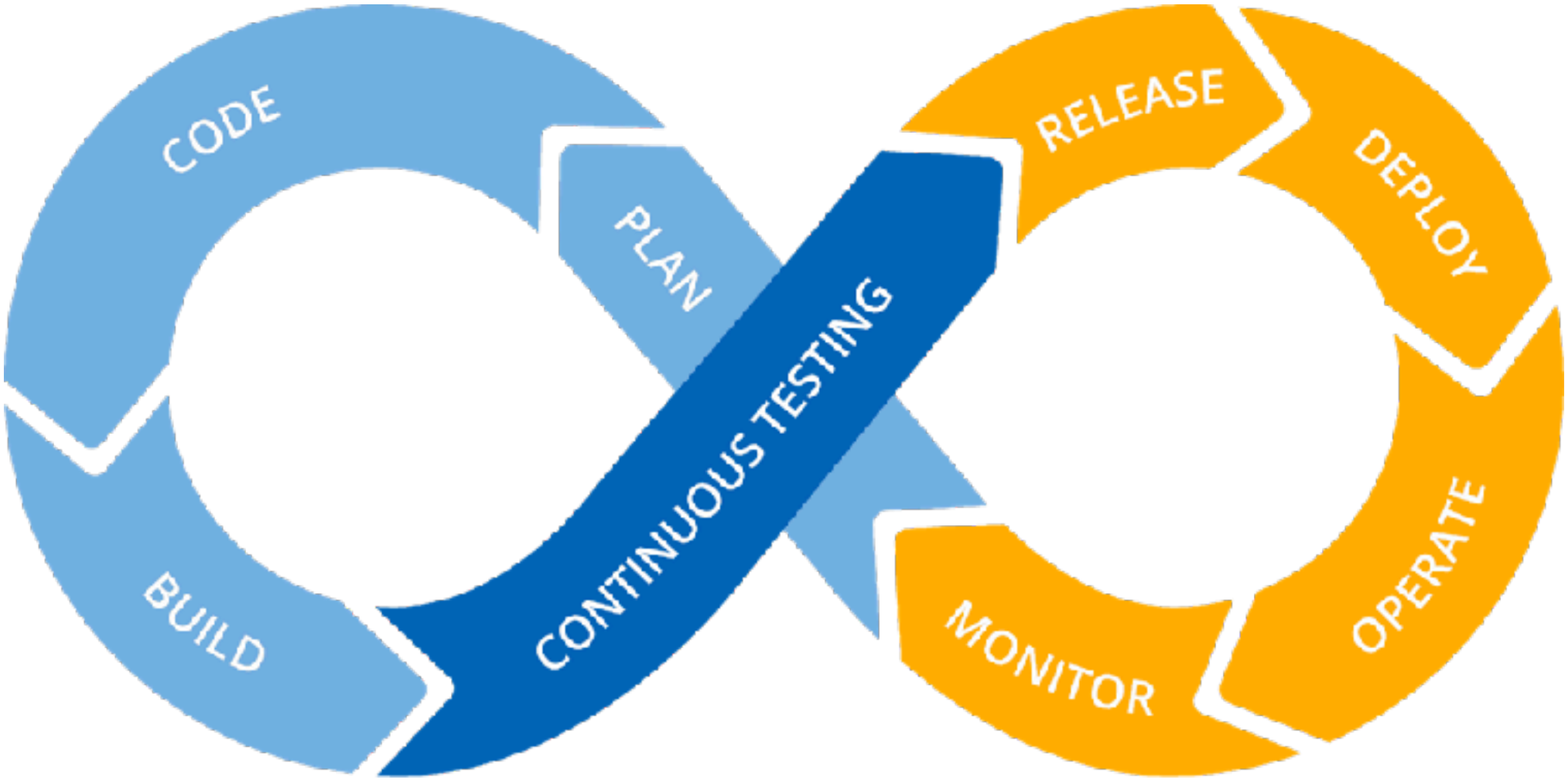


# Continuous improvement

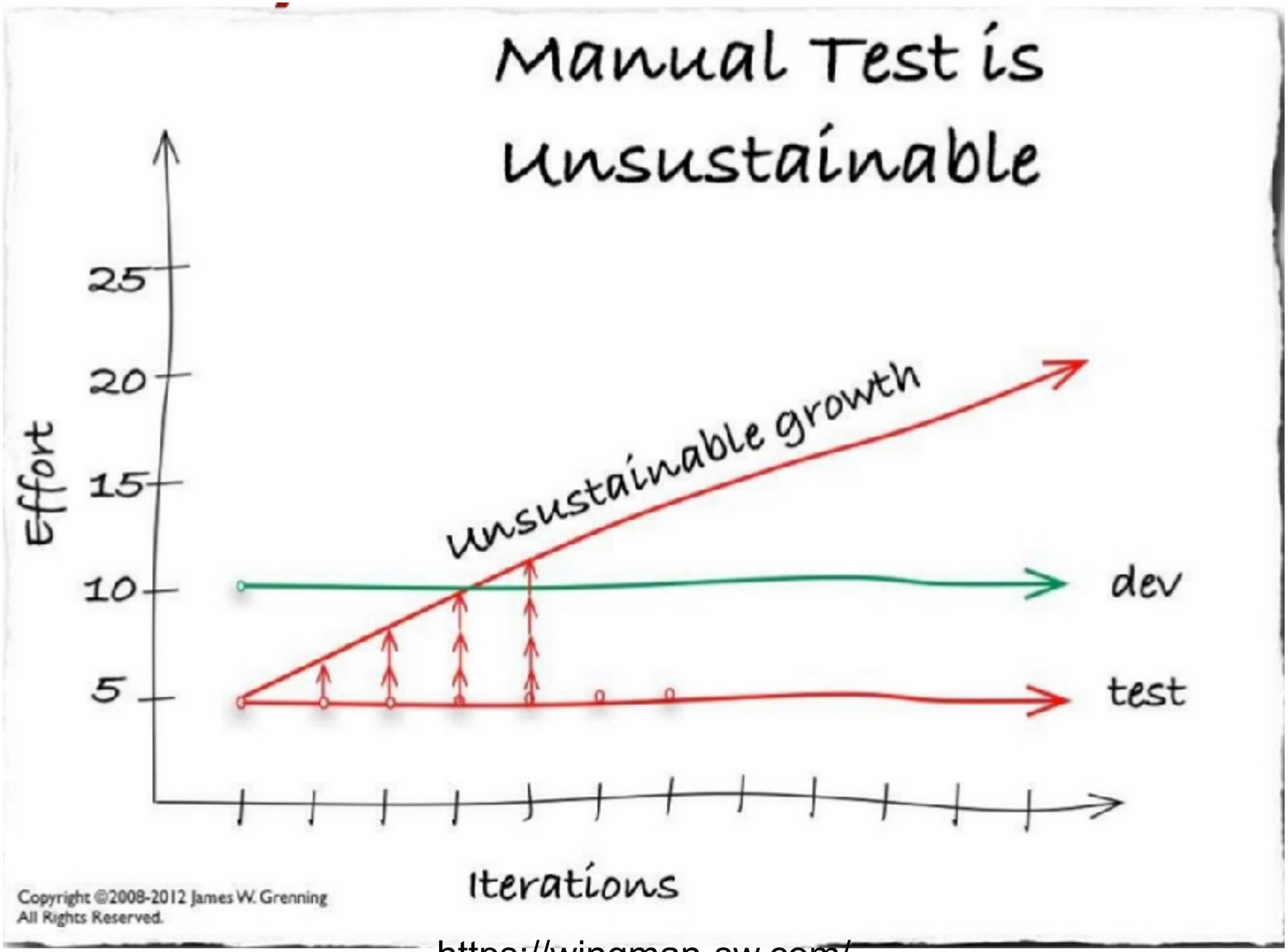


# Continuous testing

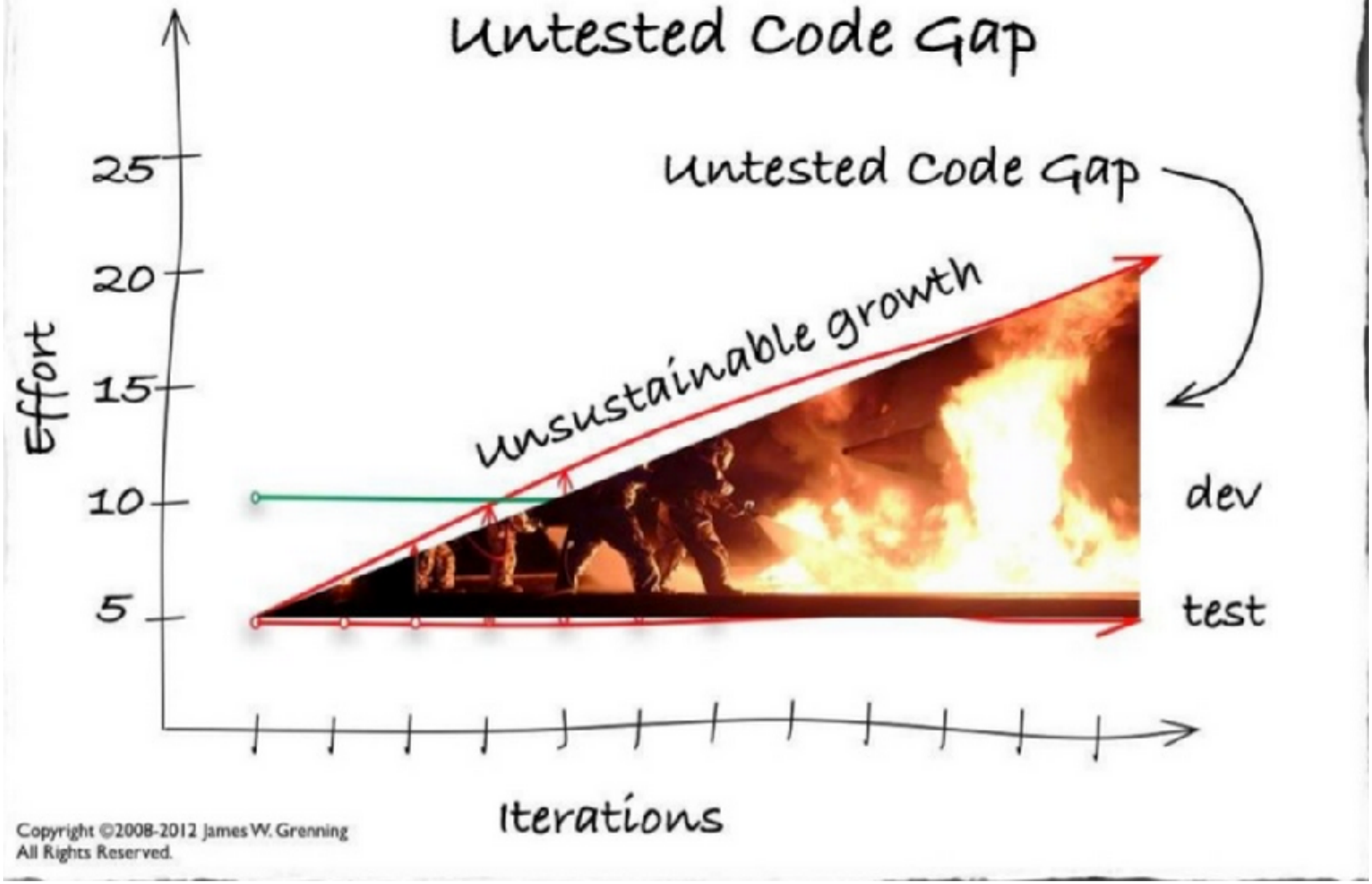


# But …

# Manual testing ?




https://wingman-sw.com/




https://wingman-sw.com/

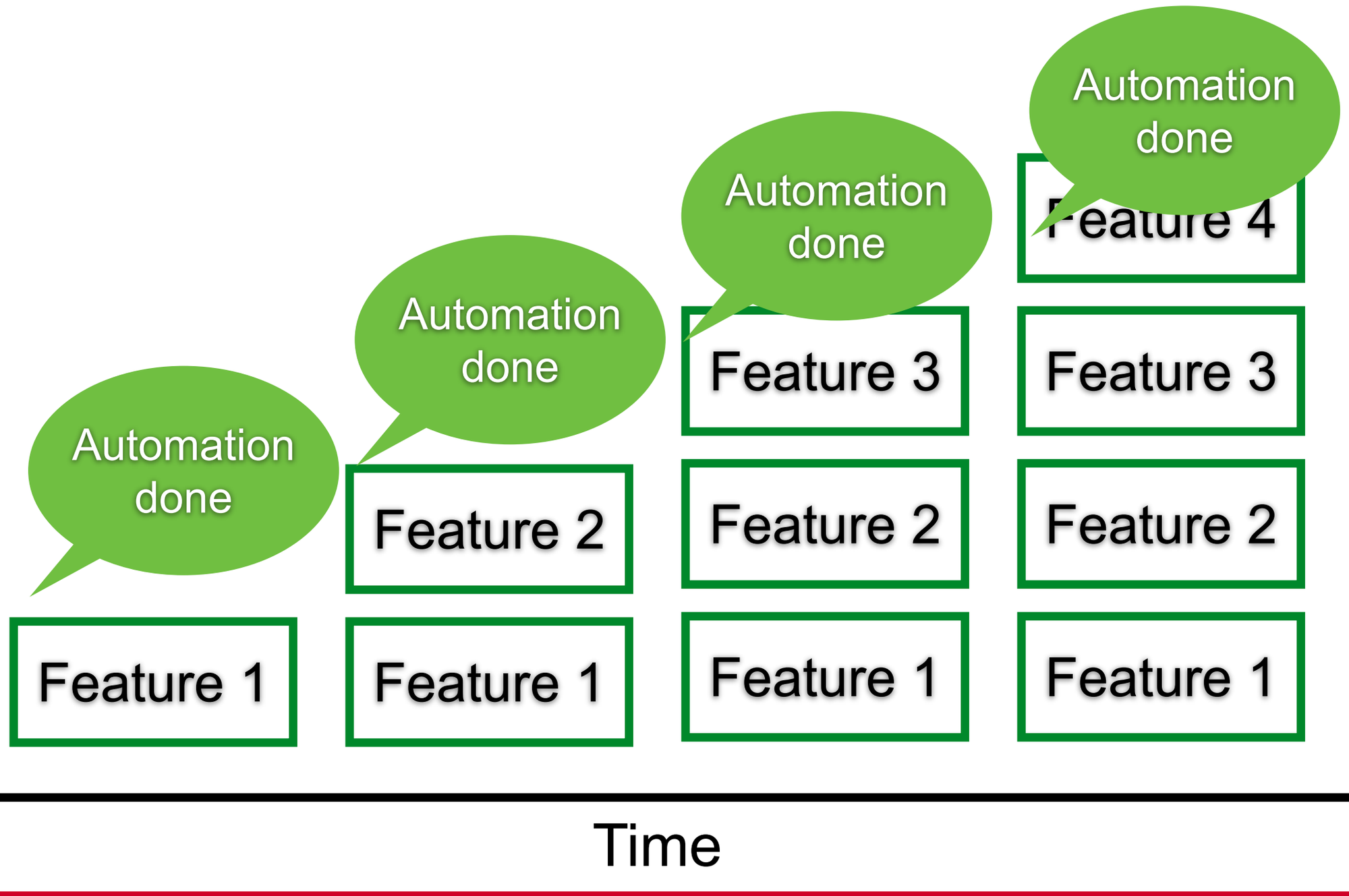
# We need automation !!

# 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

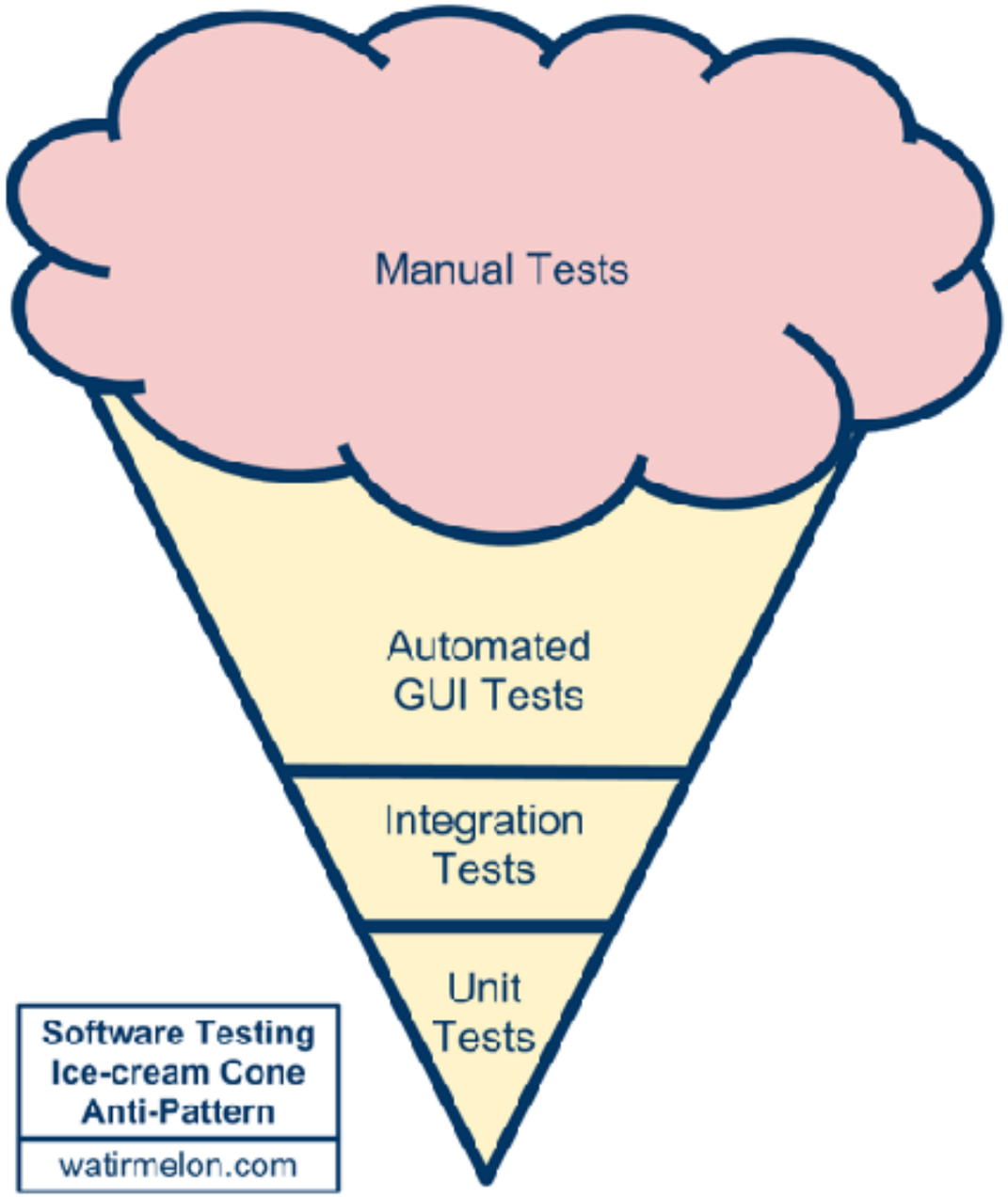
# Iterative and incremental process

Done = coded and tested

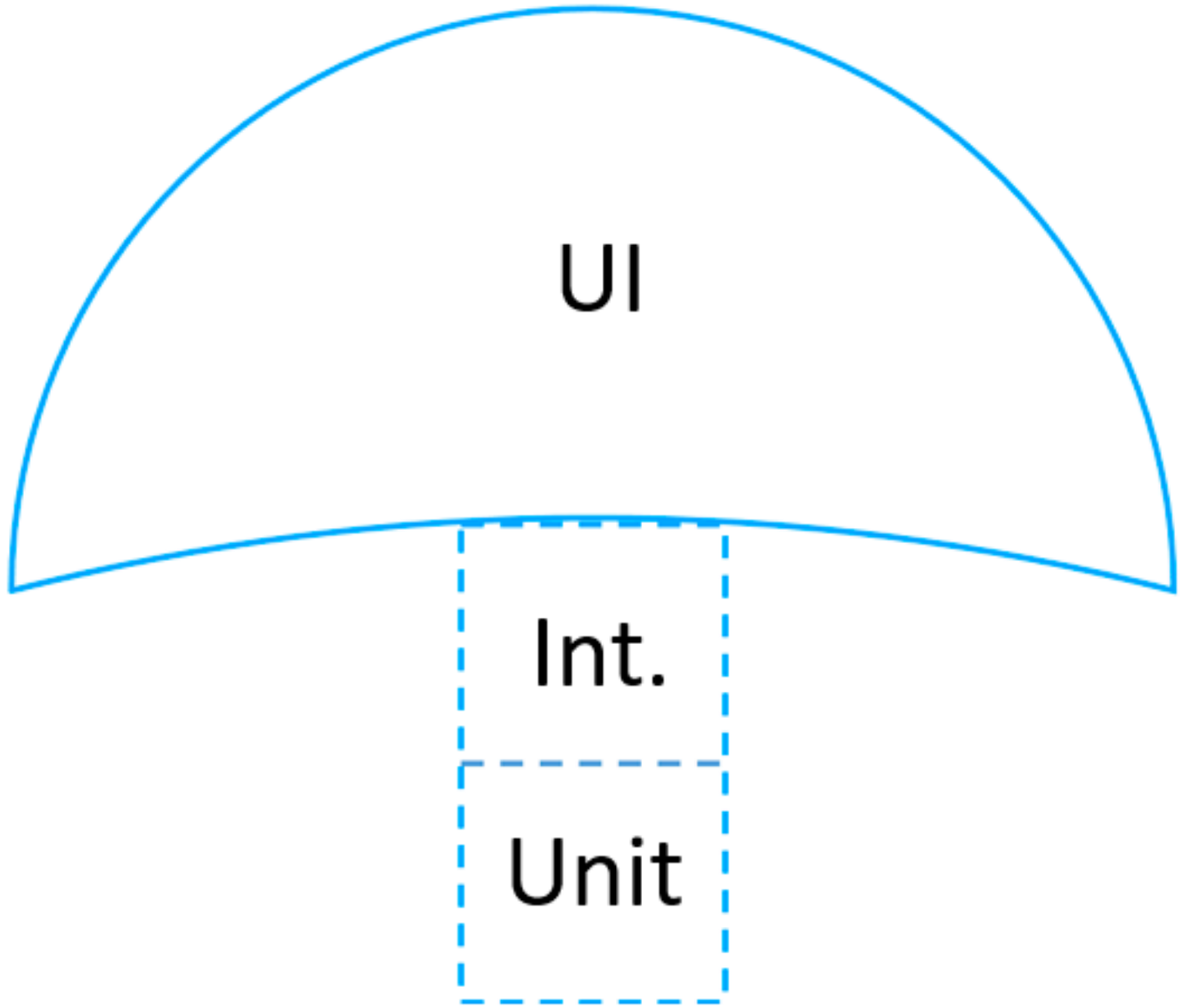


# Testing pyramid

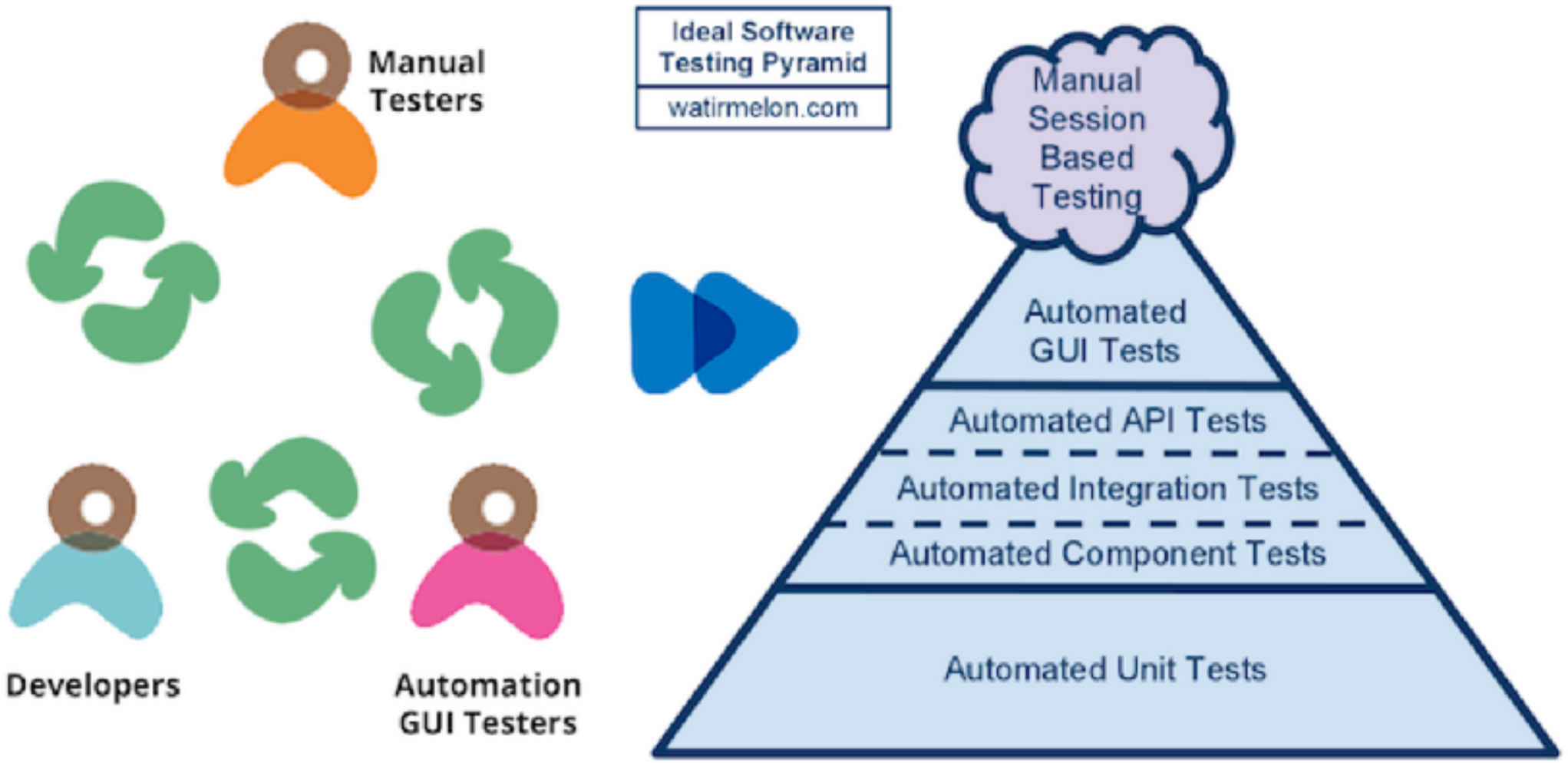
# Ice-cream testing



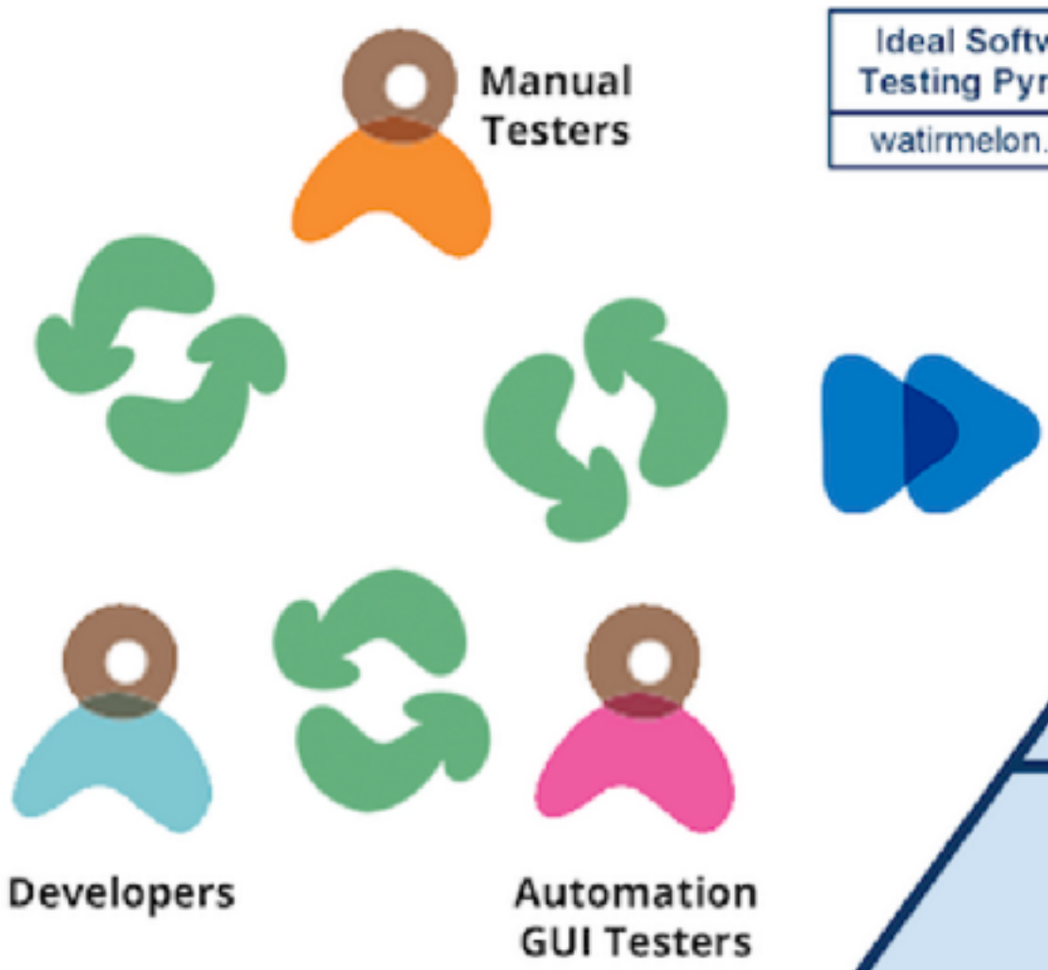
# Mashroom testing

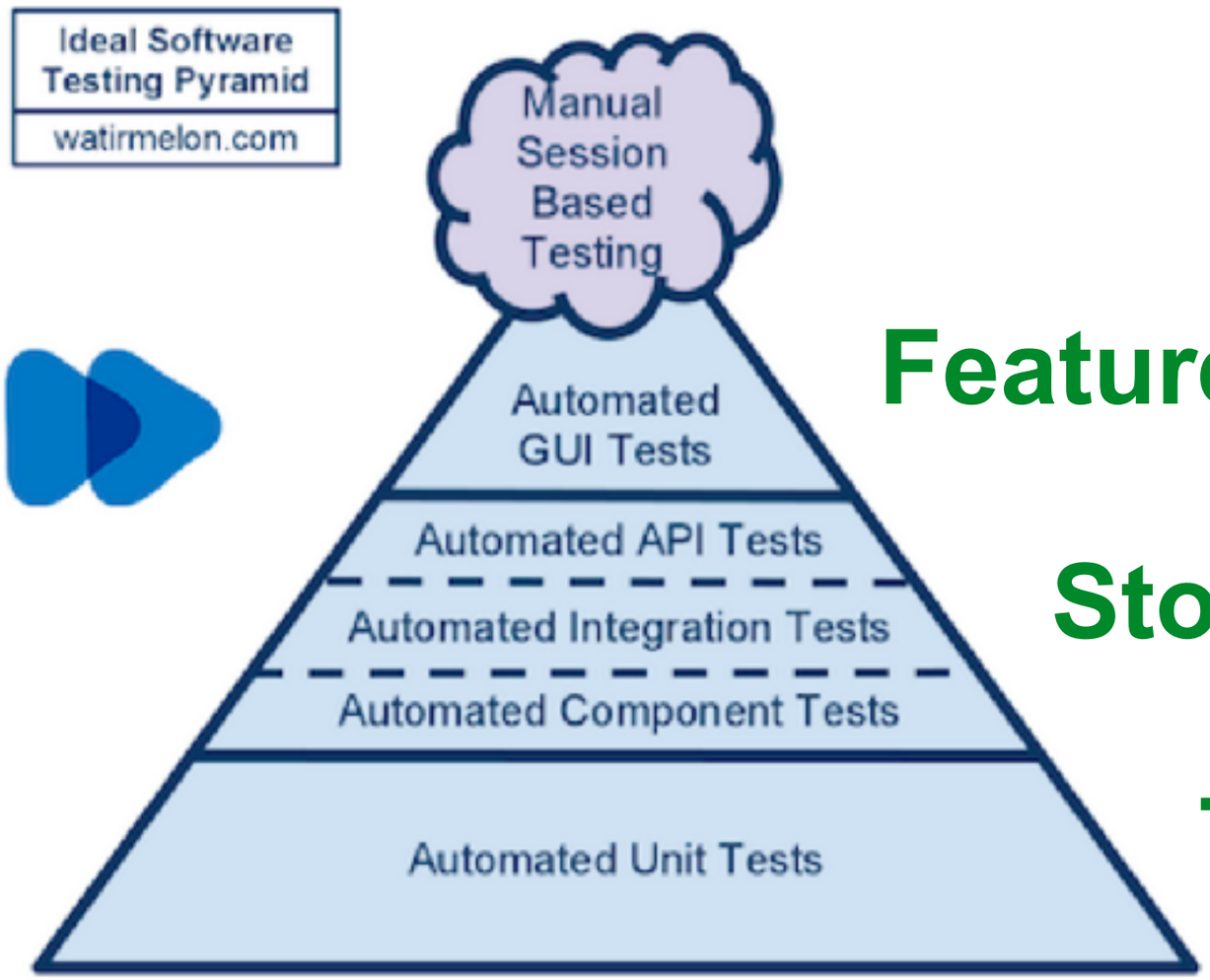


# Testing Pyramid



# Testing Pyramid



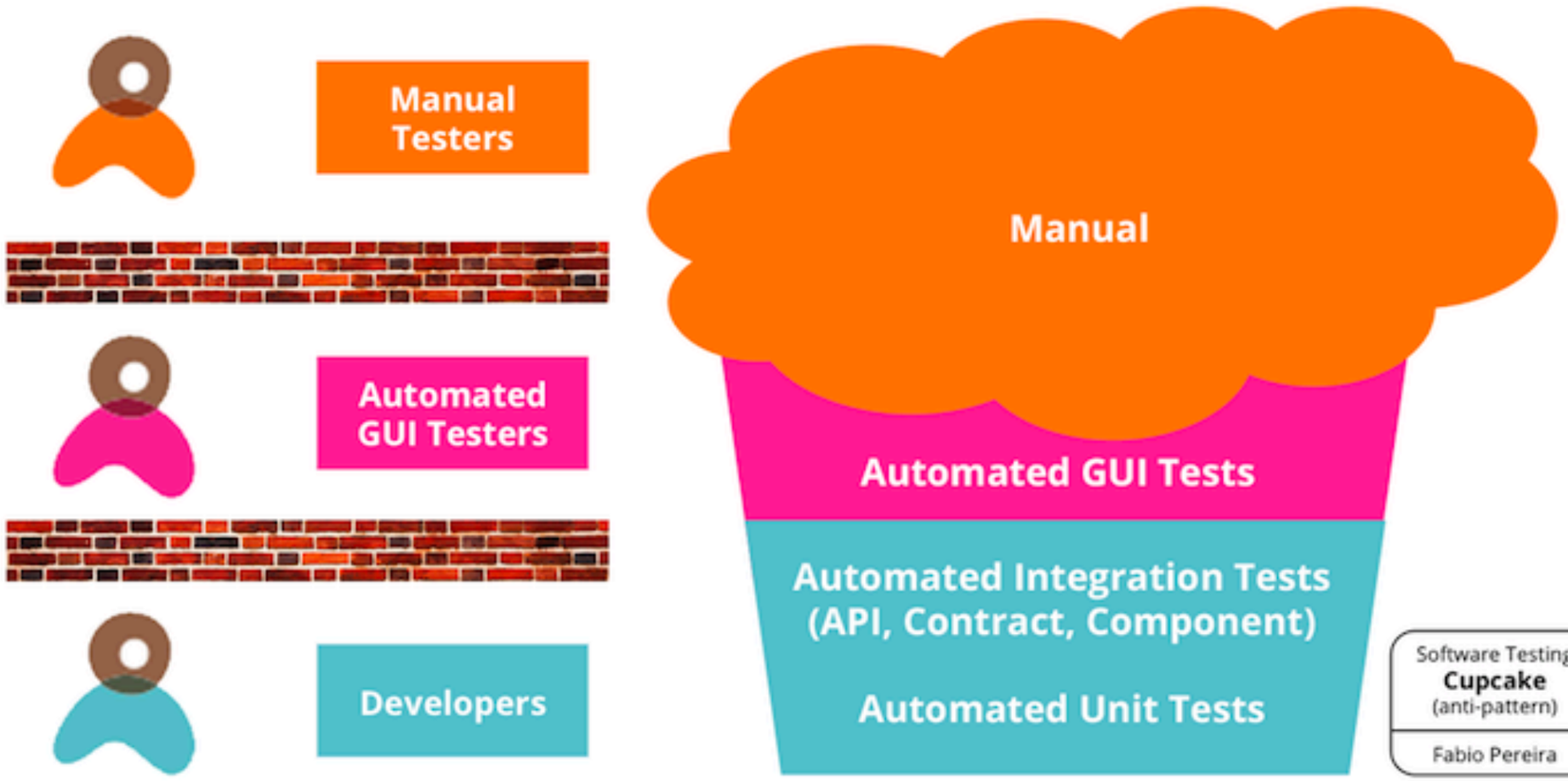




Feature

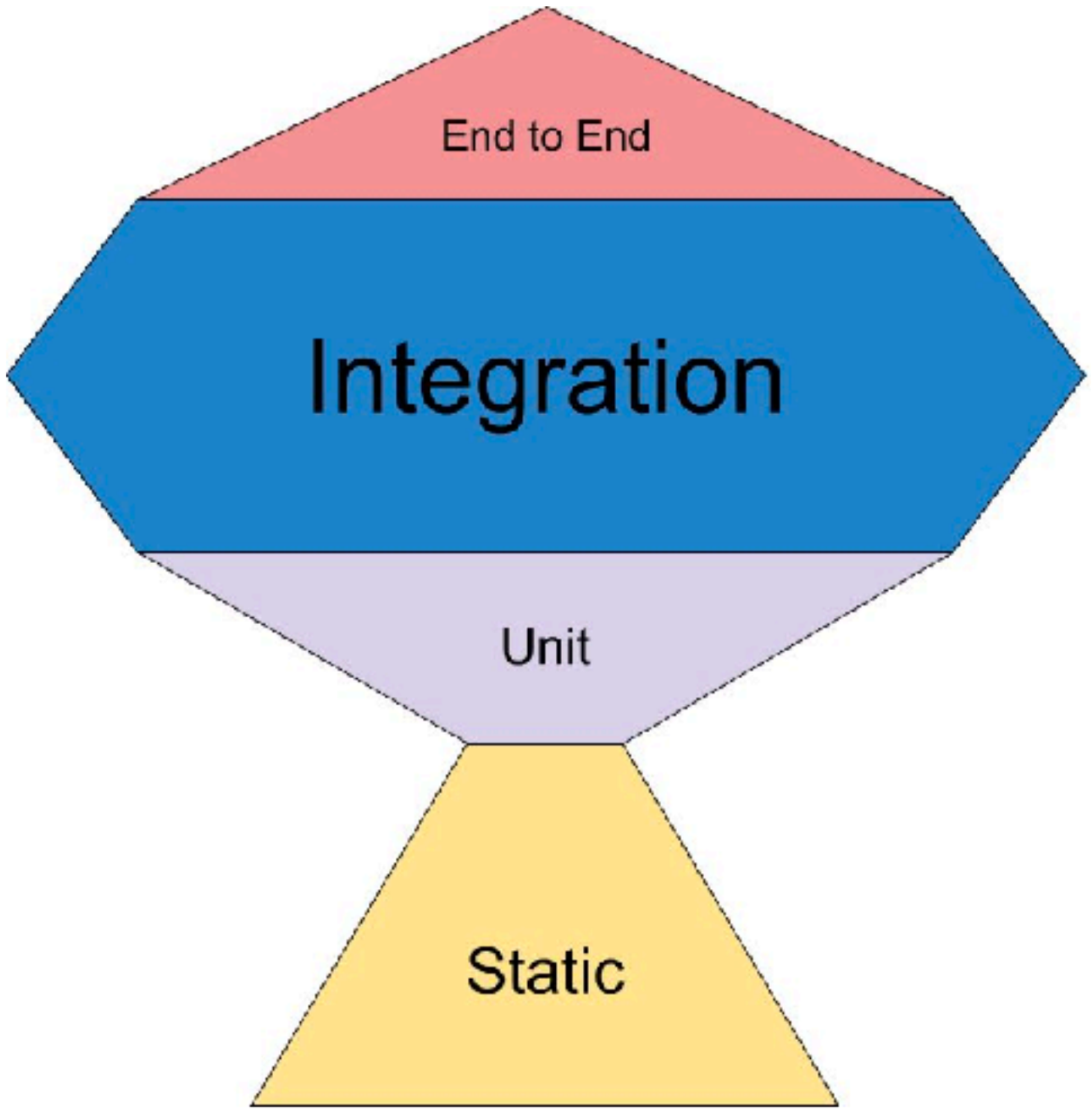
Story

Task

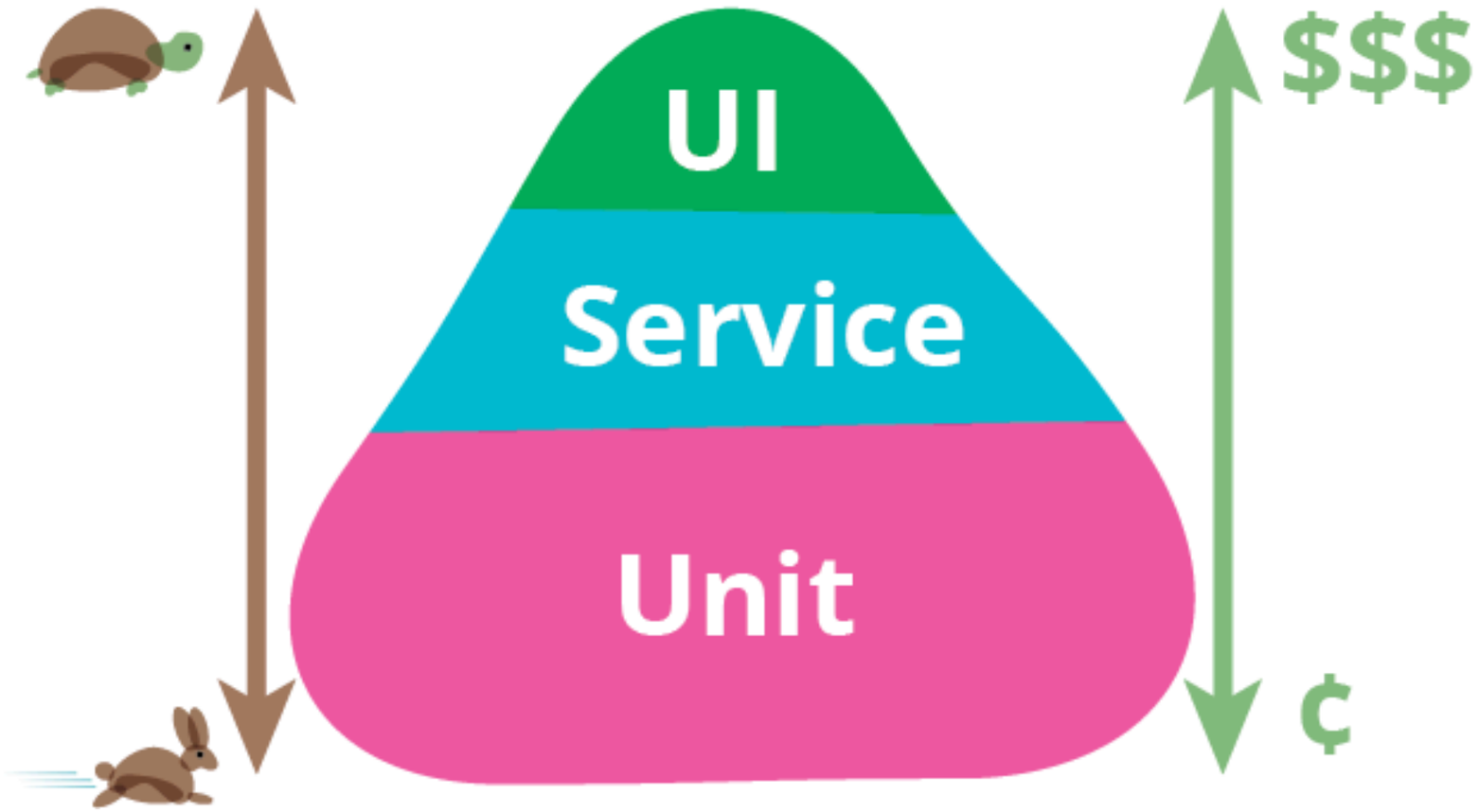
# Cupcake testing



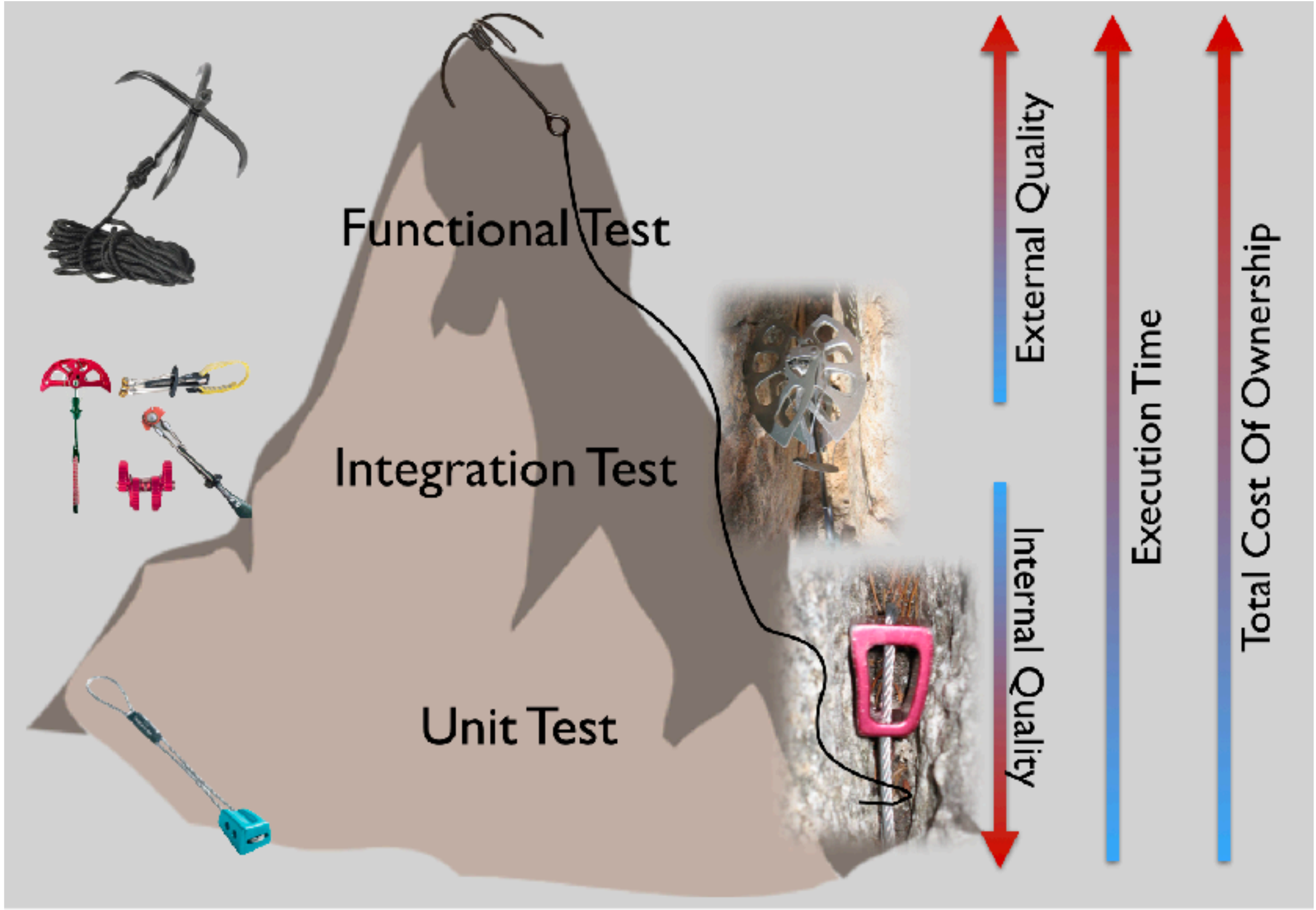
# Trophy testing



# Testing Pyramid

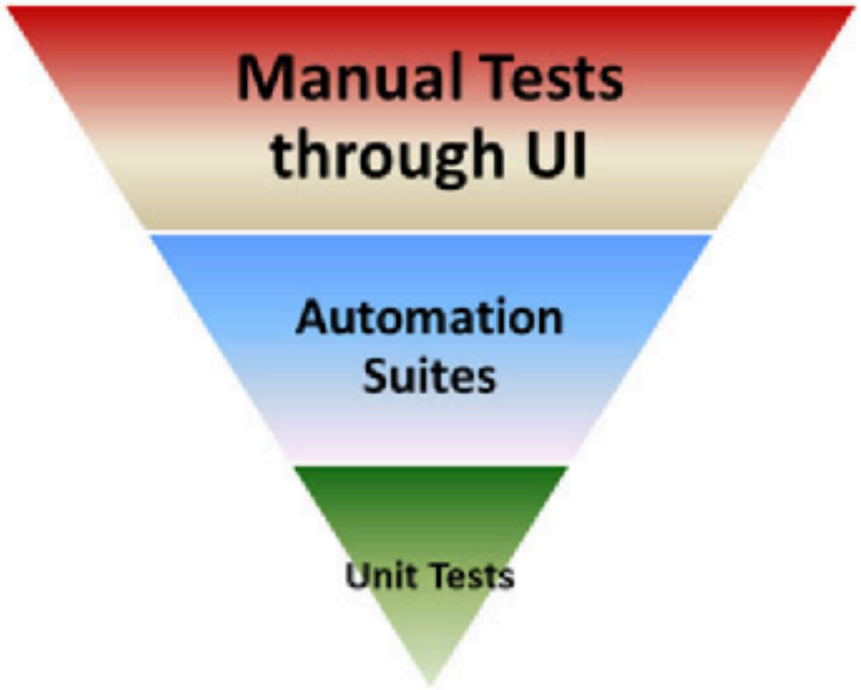



https://martinfowler.com/bliki/TestPyramid.html

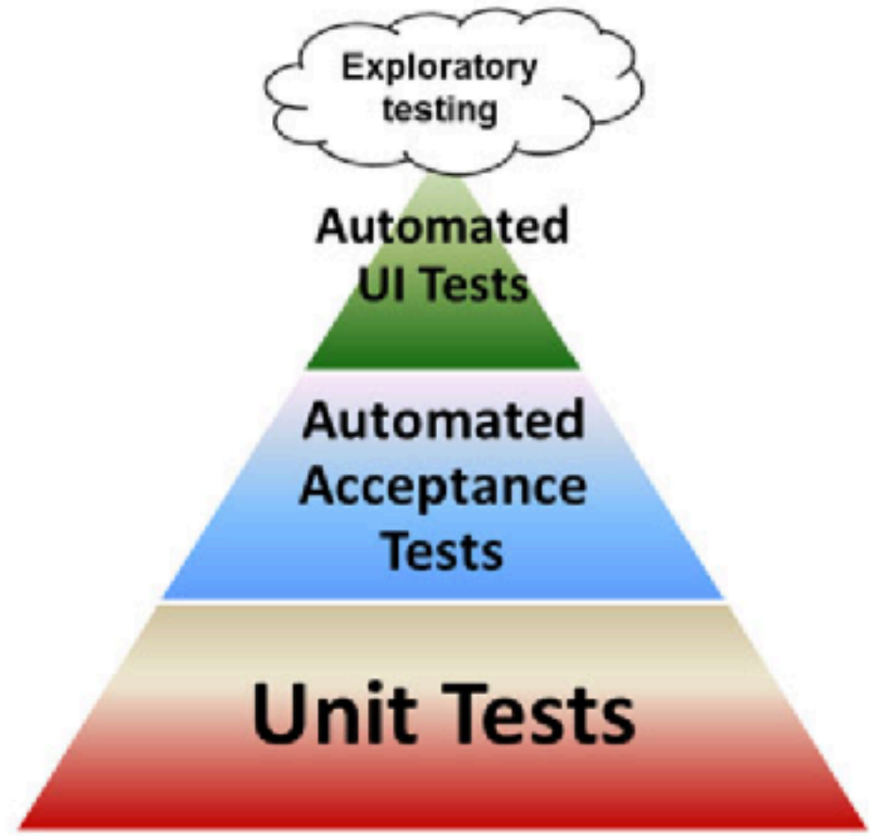



https://less.works/less/technical-excellence/unit-testing.html

# Find vs Prevent




**Traditional**
**(find bugs)**




**Agile**
**(prevent bugs)**

# 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

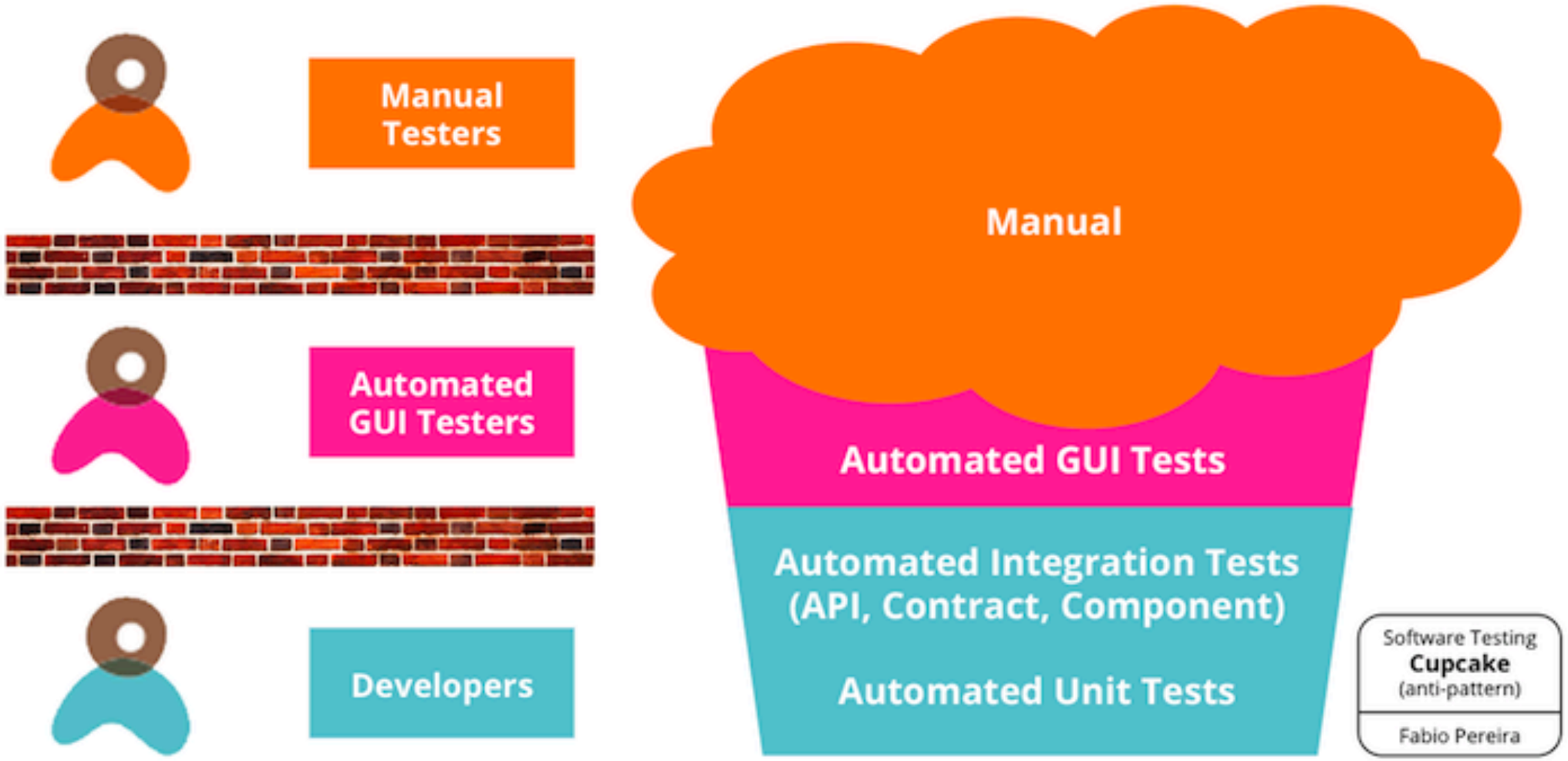
We are here to ensure requirement are met

We are here to break the software

**Think**

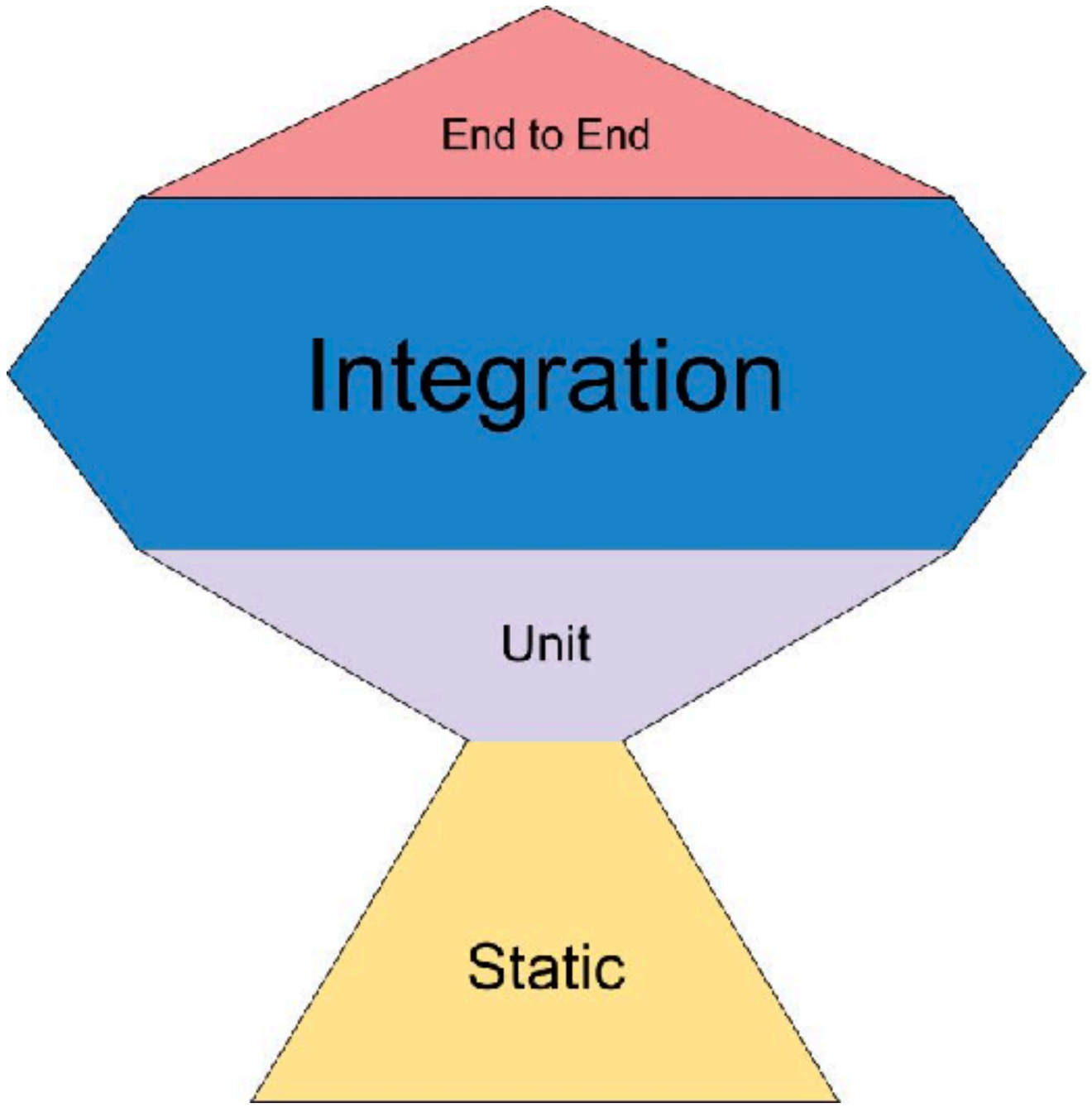
What can I do to help deliver the software successfully !!

# Cupcake testing !!




https://www.thoughtworks.com/insights/blog/introducing-software-testing-cupcake-anti-pattern

# Trophy testing




https://kentcdodds.com/blog/write-tests

# 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

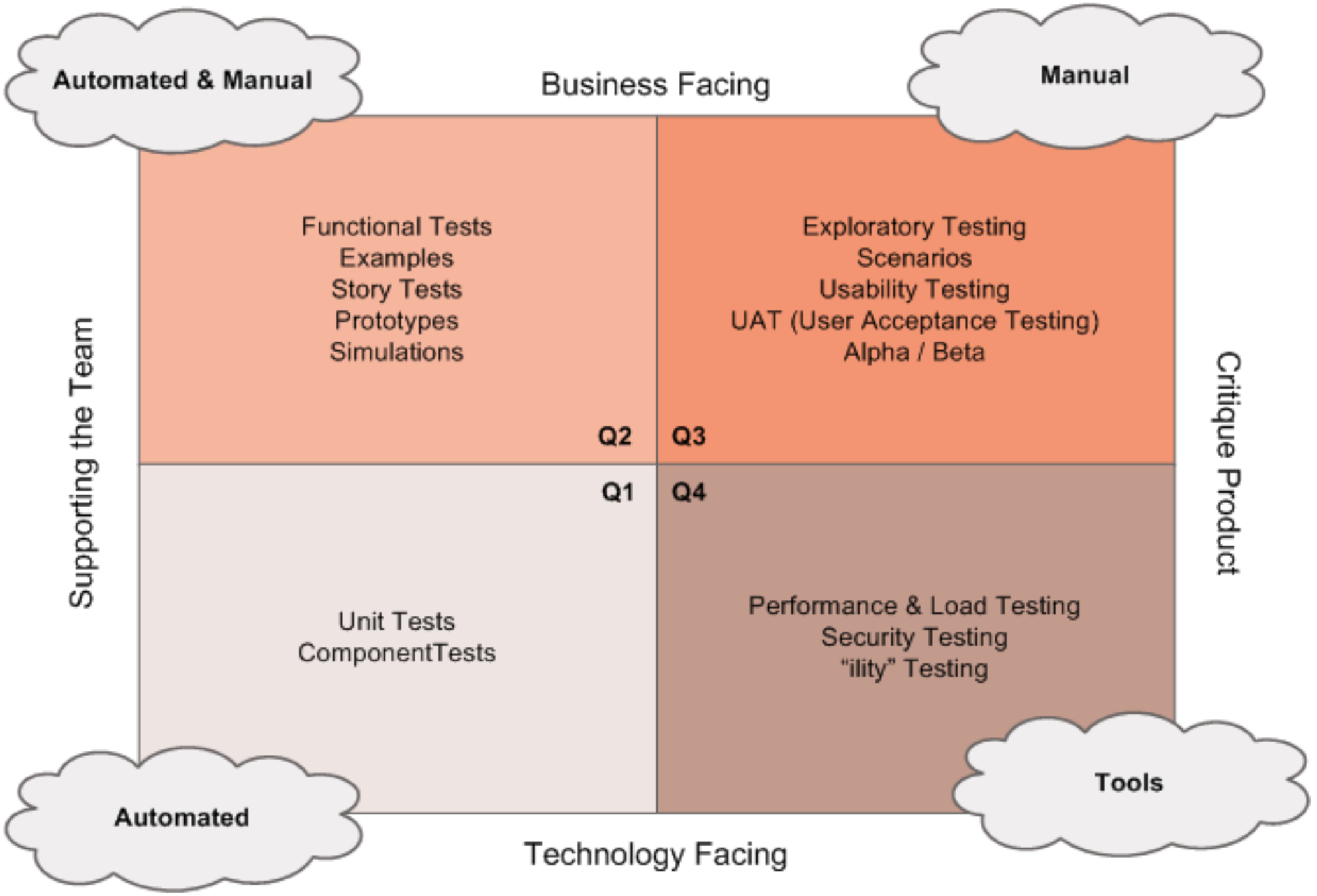
Automating every paths

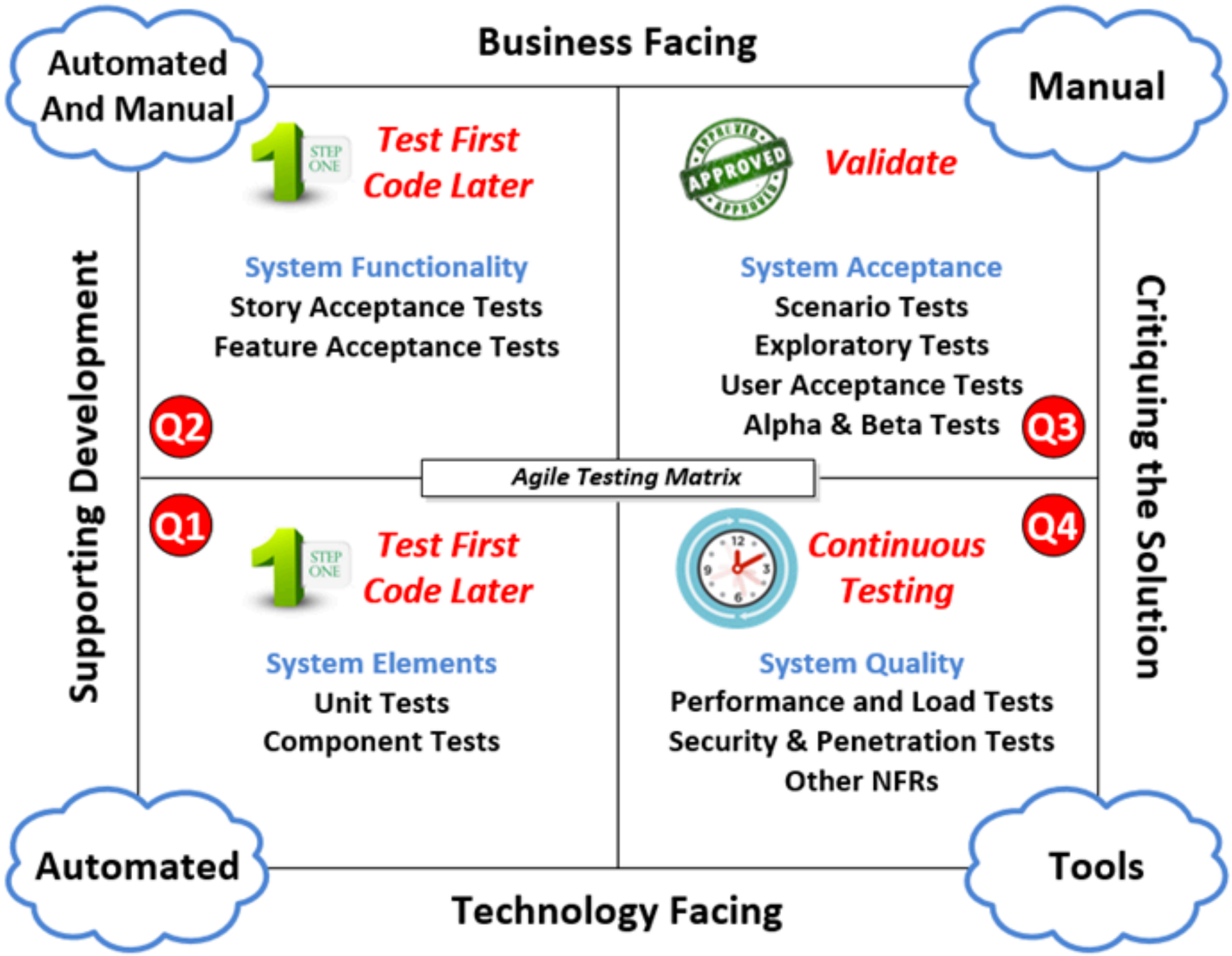
# Testing Quadrant

# Testing Quadrants

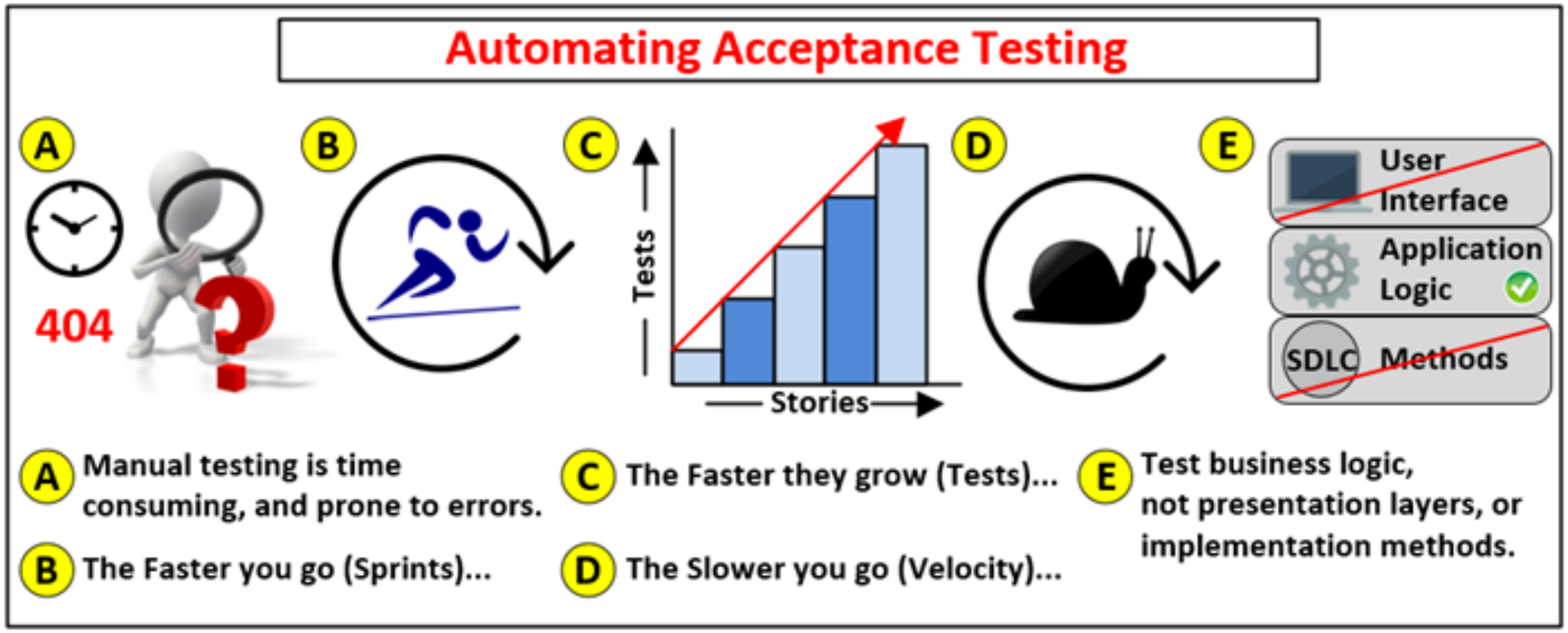
Scope of tests
Classify your tests
Visible your tests

# Agile Testing Quadrants



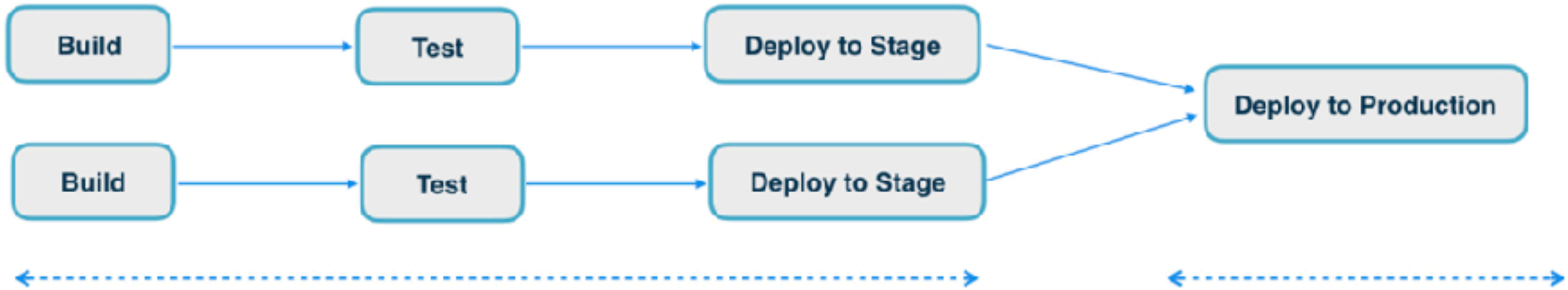


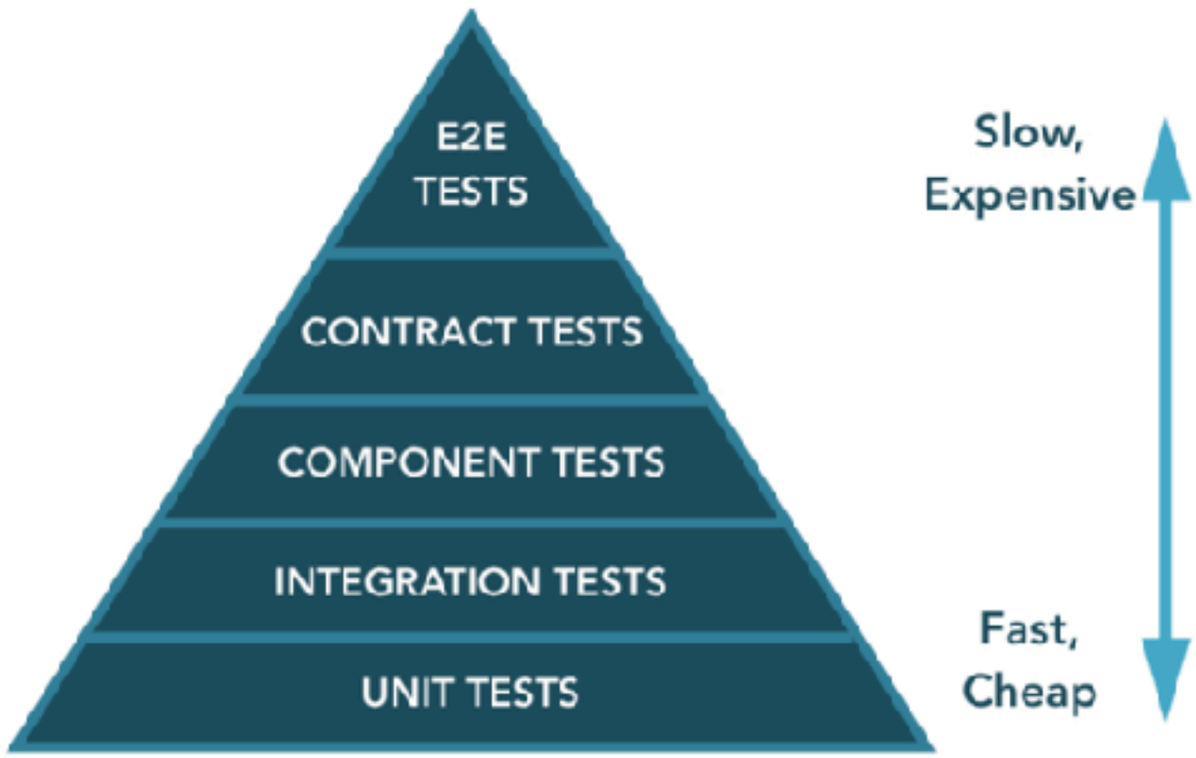
# Test automation is essential



# Test strategy

# Test strategy



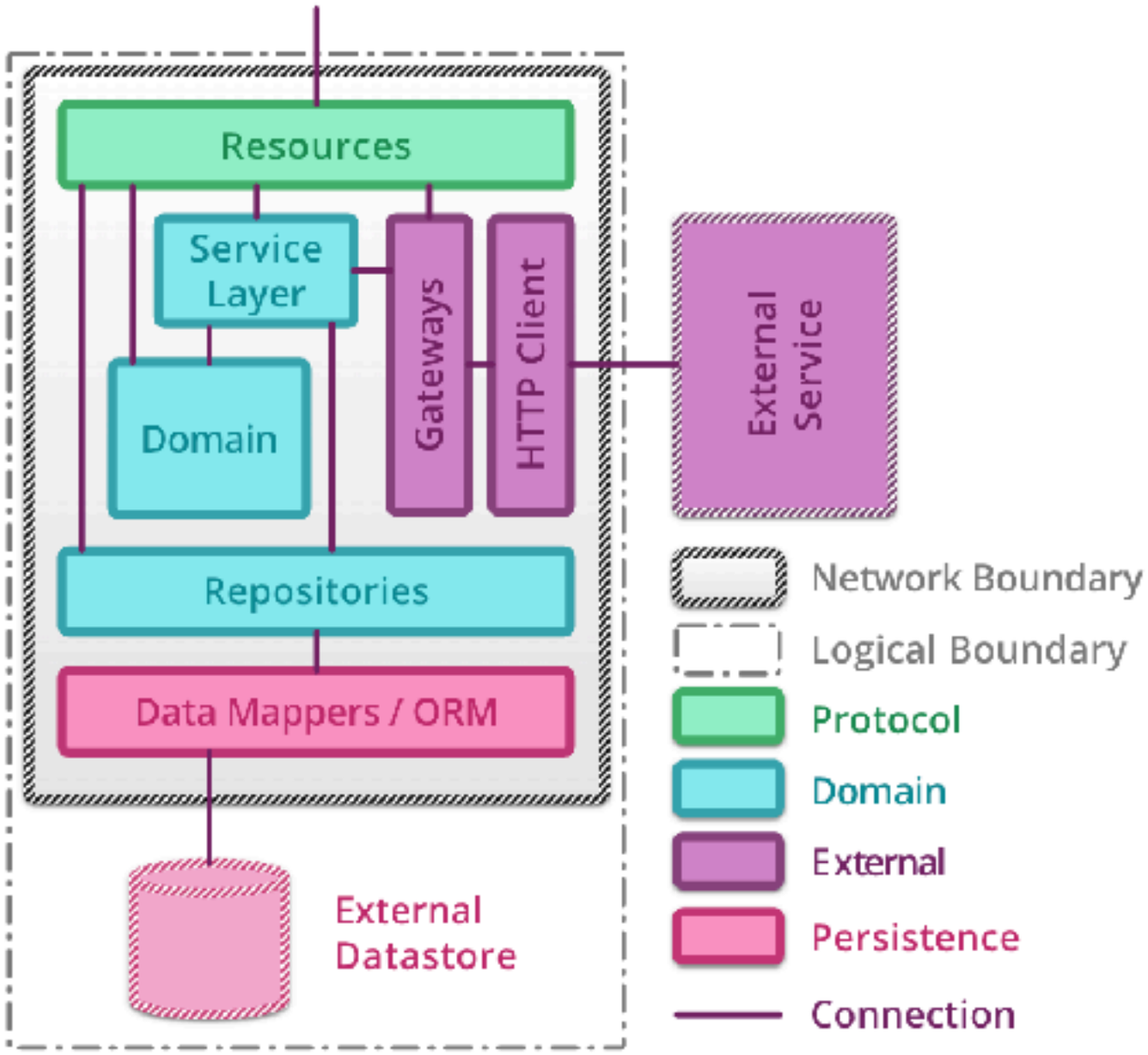




Monitoring
Distributed Tracing
Fault-injection Testing

---

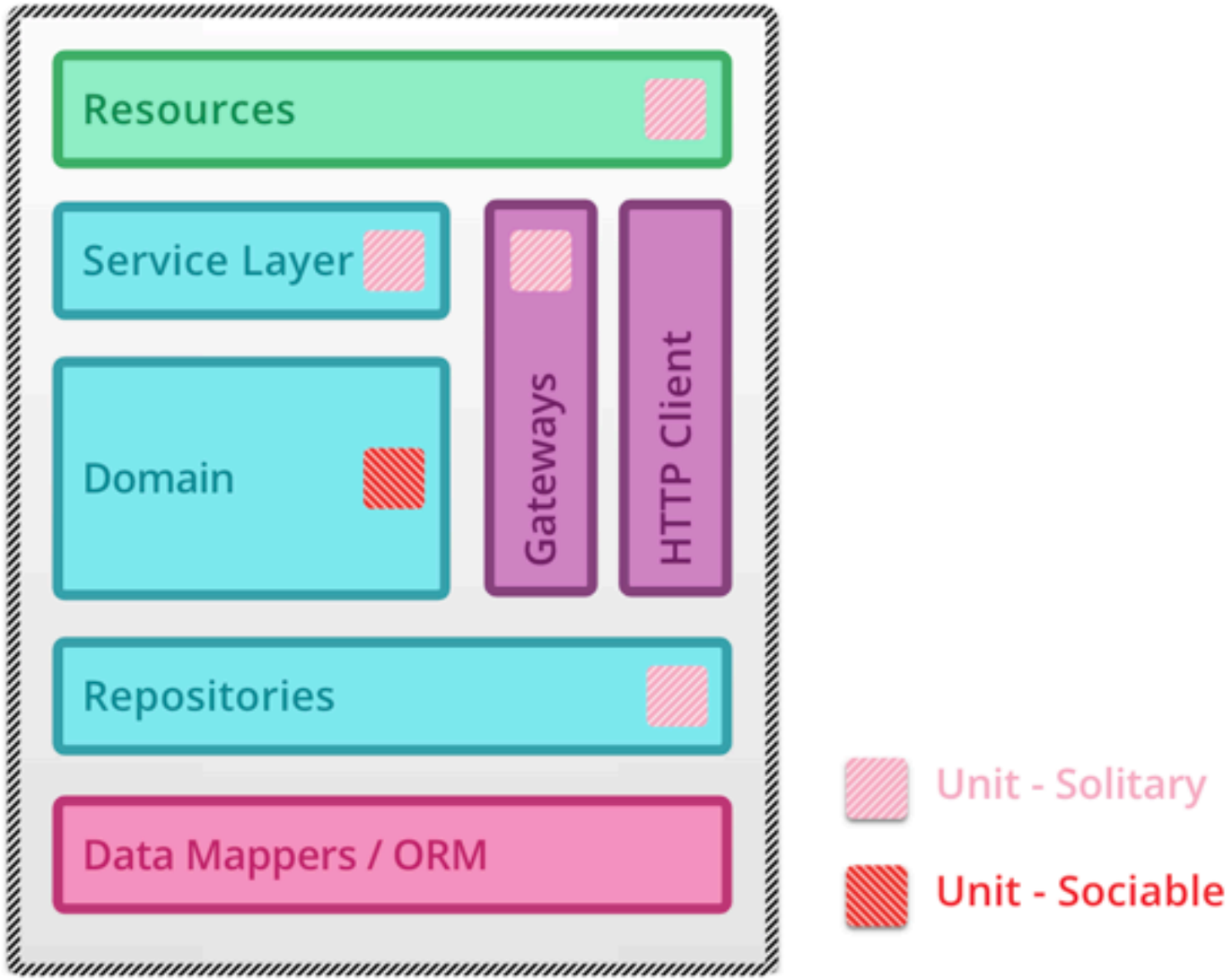
Canary Deployments
Blue-Green Deployments
A/B Testing

# Understand project structure



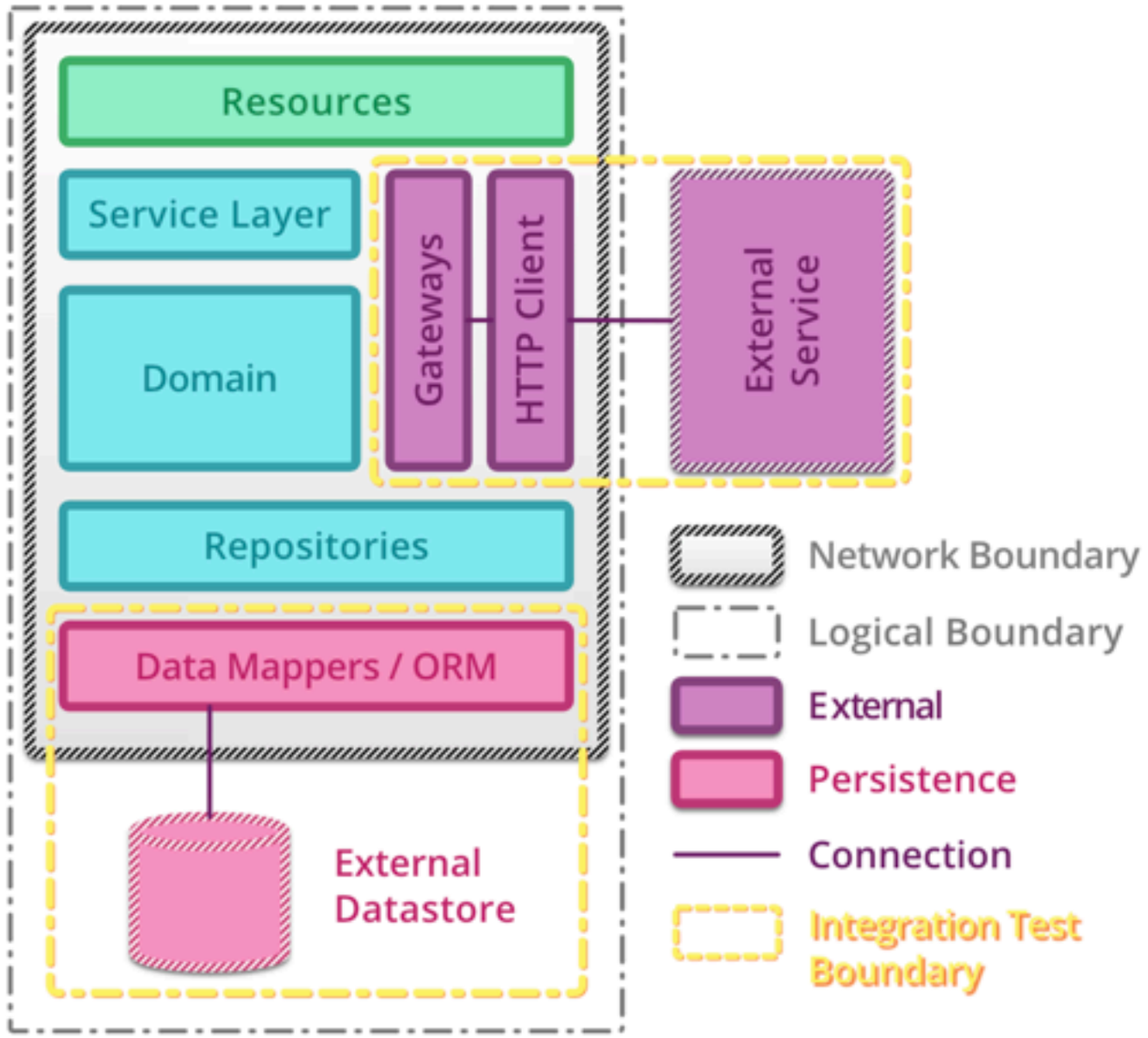

https://martinfowler.com/articles/microservice-testing/

# Unit testing



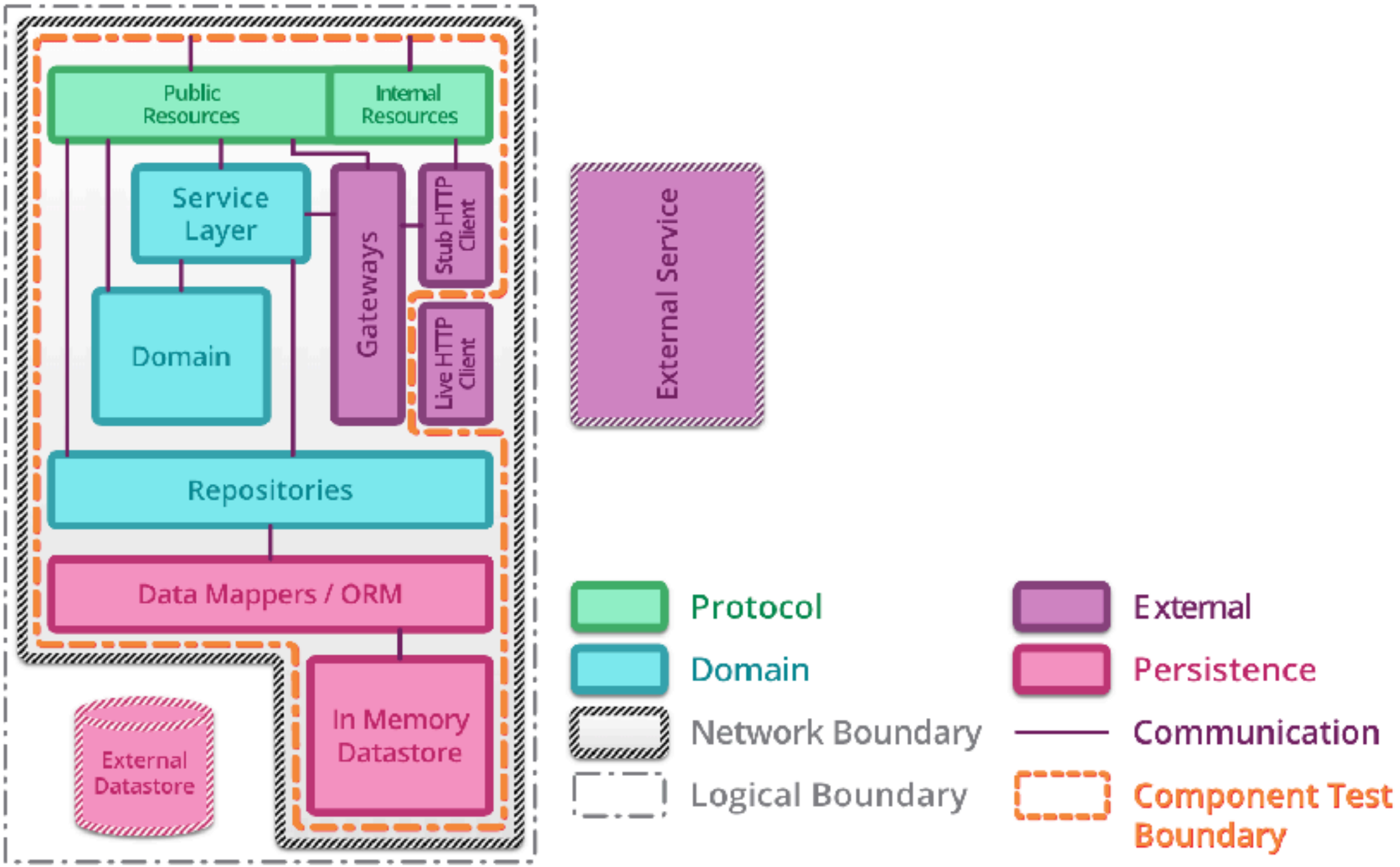

https://martinfowler.com/articles/microservice-testing/

# Integration testing




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

# Component testing




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

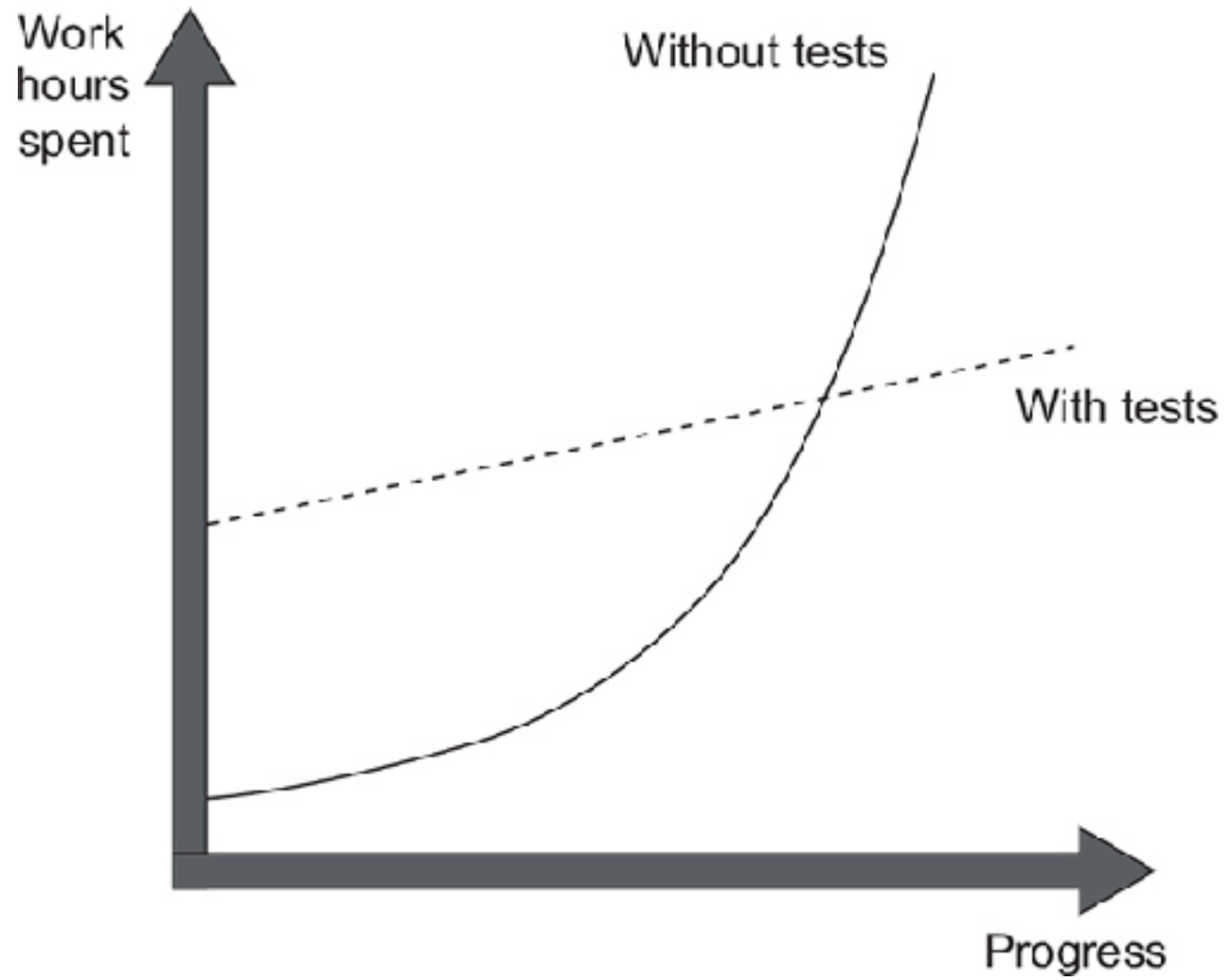
Don’t ignore it if it fails

Fix the code or fix the test

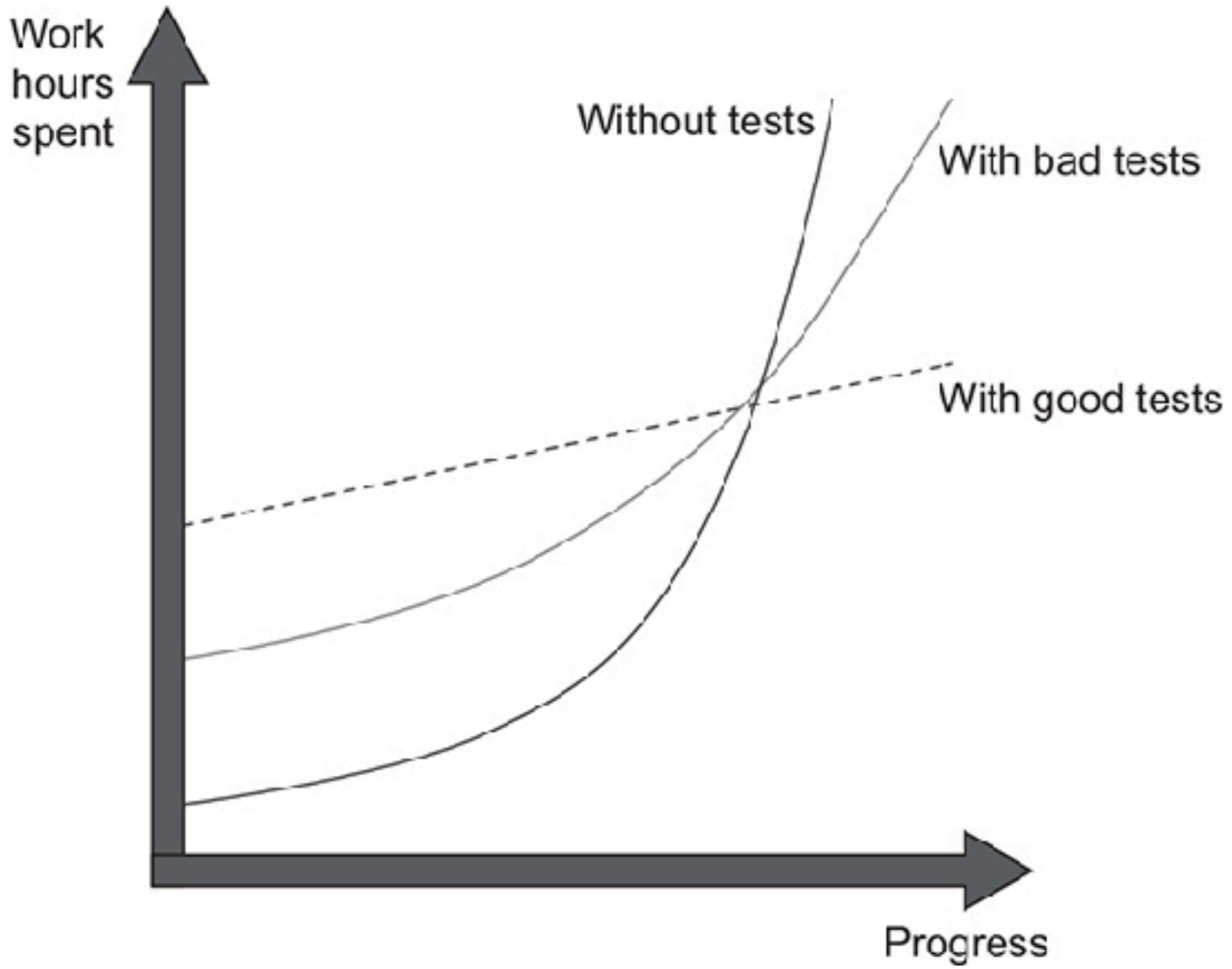
**100% of regression tests must pass all the time**

Always refactor or improve

# Reduce work hours with tests



# Good vs Bad tests



# Test data !!

Avoid database/external system access

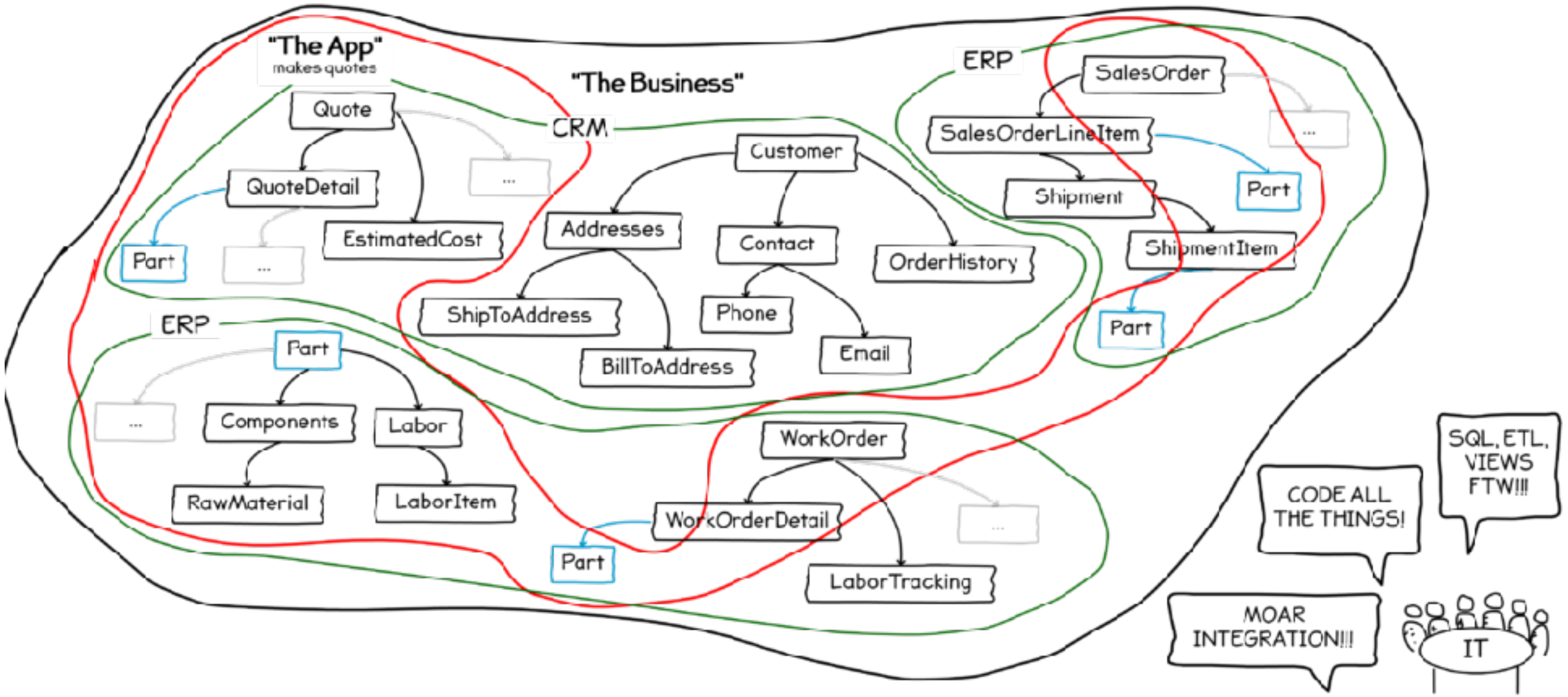
Setup/ tear down test data

**Use production-like data**

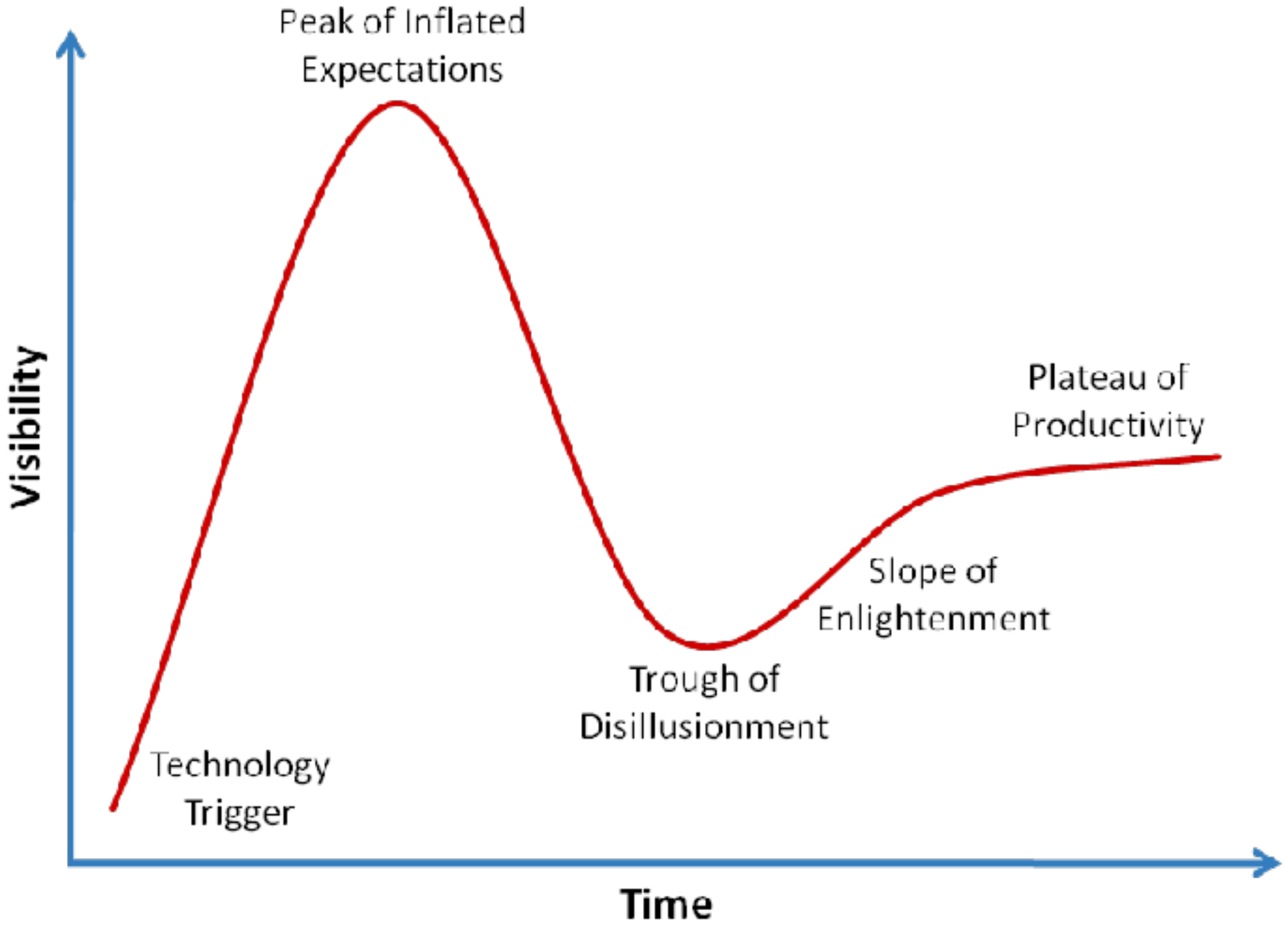
Need to control your data test

# System impacts !!

## Know your systems



# Hype cycle



# Automation feedback

Easier over time ?

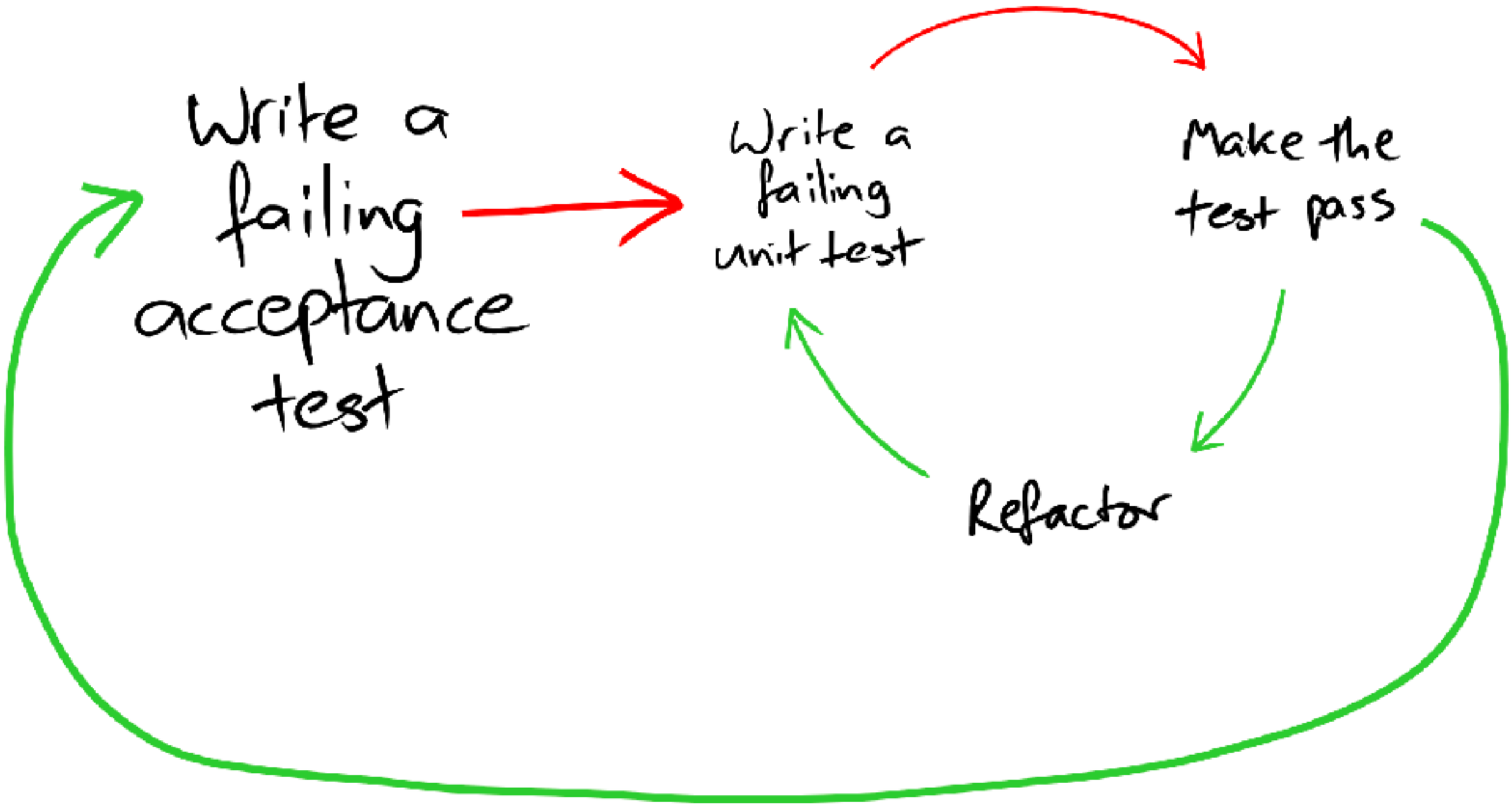
Time spent on maintenance ?

Test find regression bugs ?

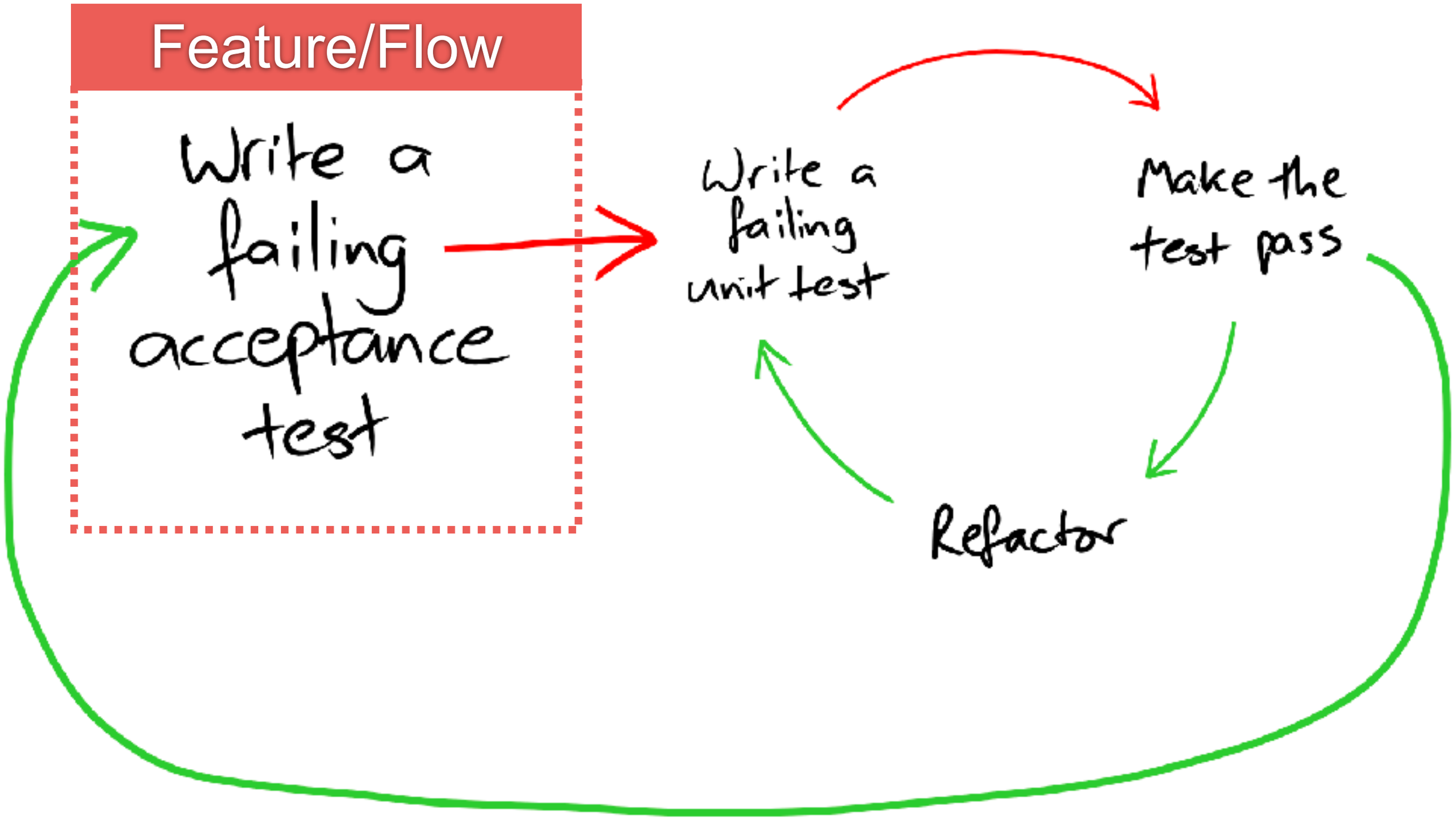
# Start with simple

# Use feedback to improve

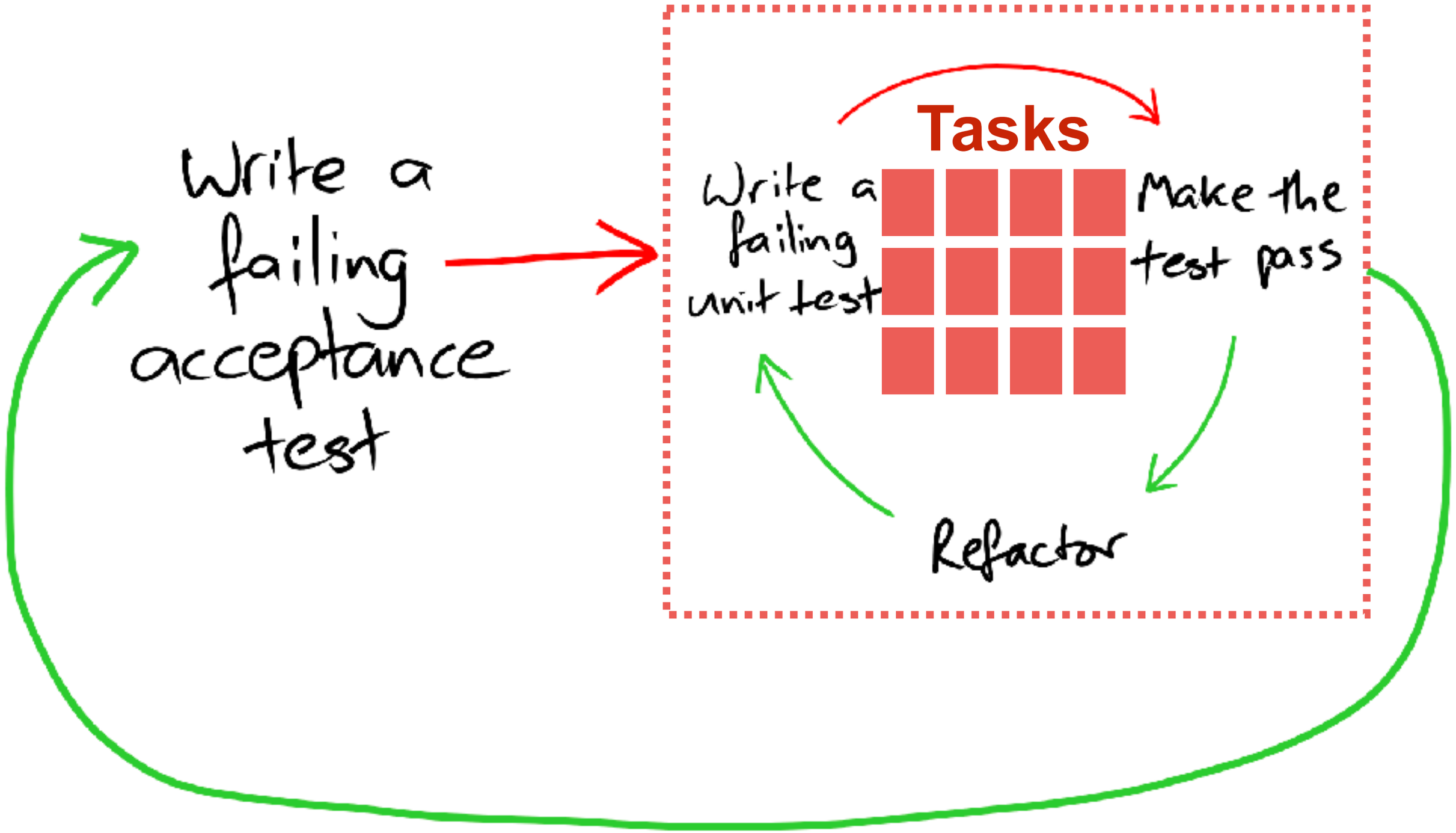
# Outside-in develop/test



# Outside-in develop/test



# Outside-in develop/test



# Workshop

# Register workshop !!

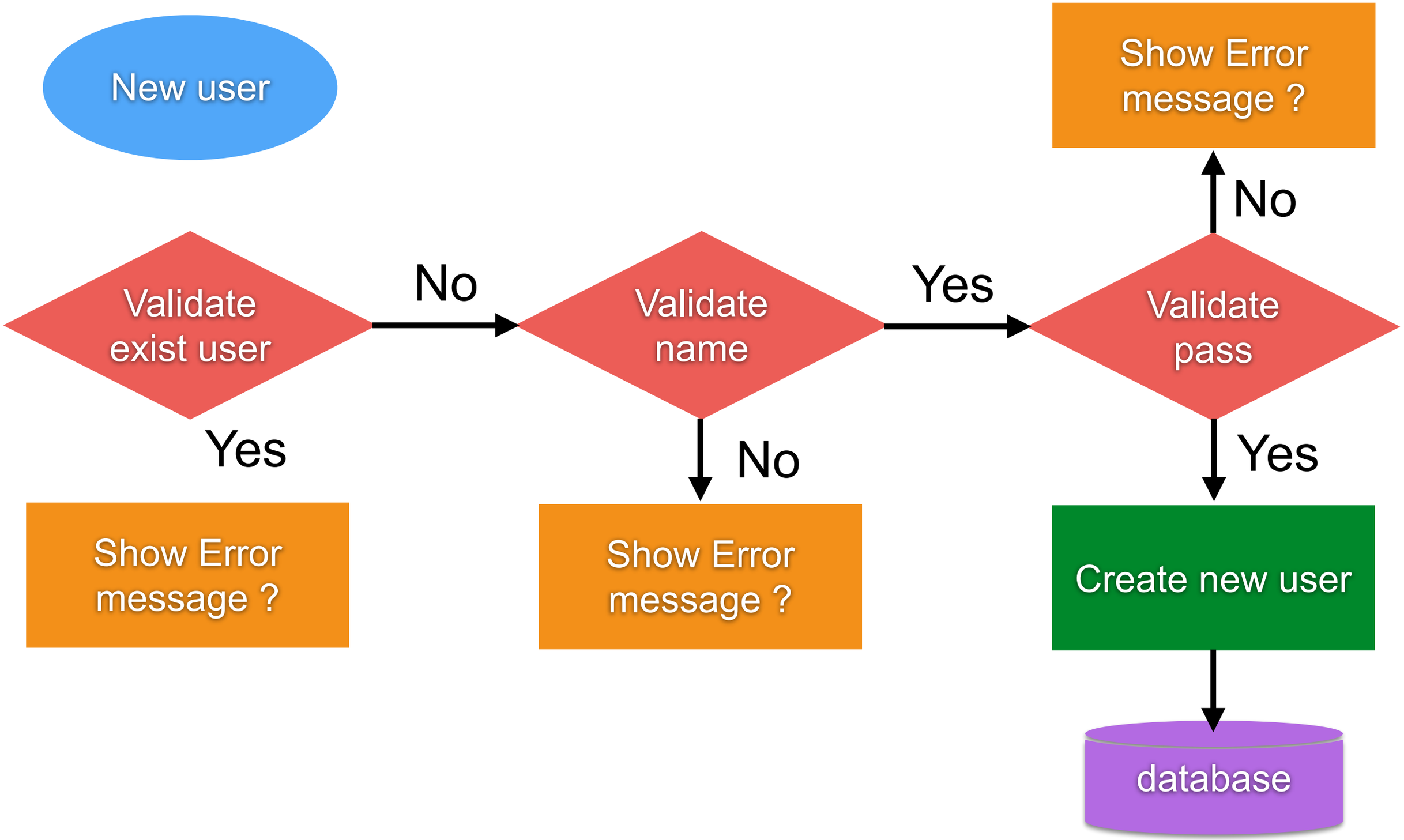
**As a new user,**

**I want to** create an account with
User name and password

**So that** only I can access my information

# Test cases ?

# Flow ?

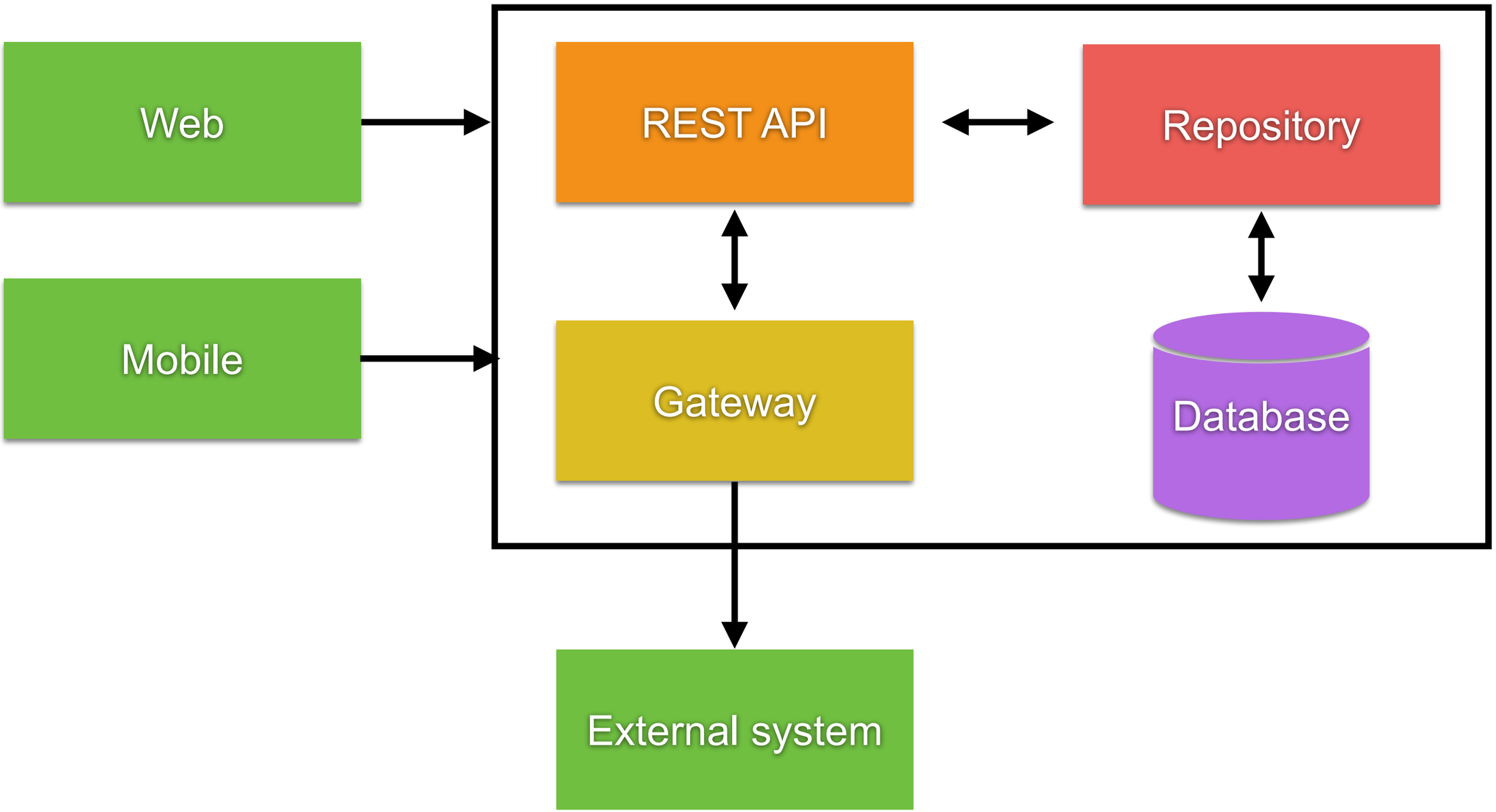


# Thinking about automation

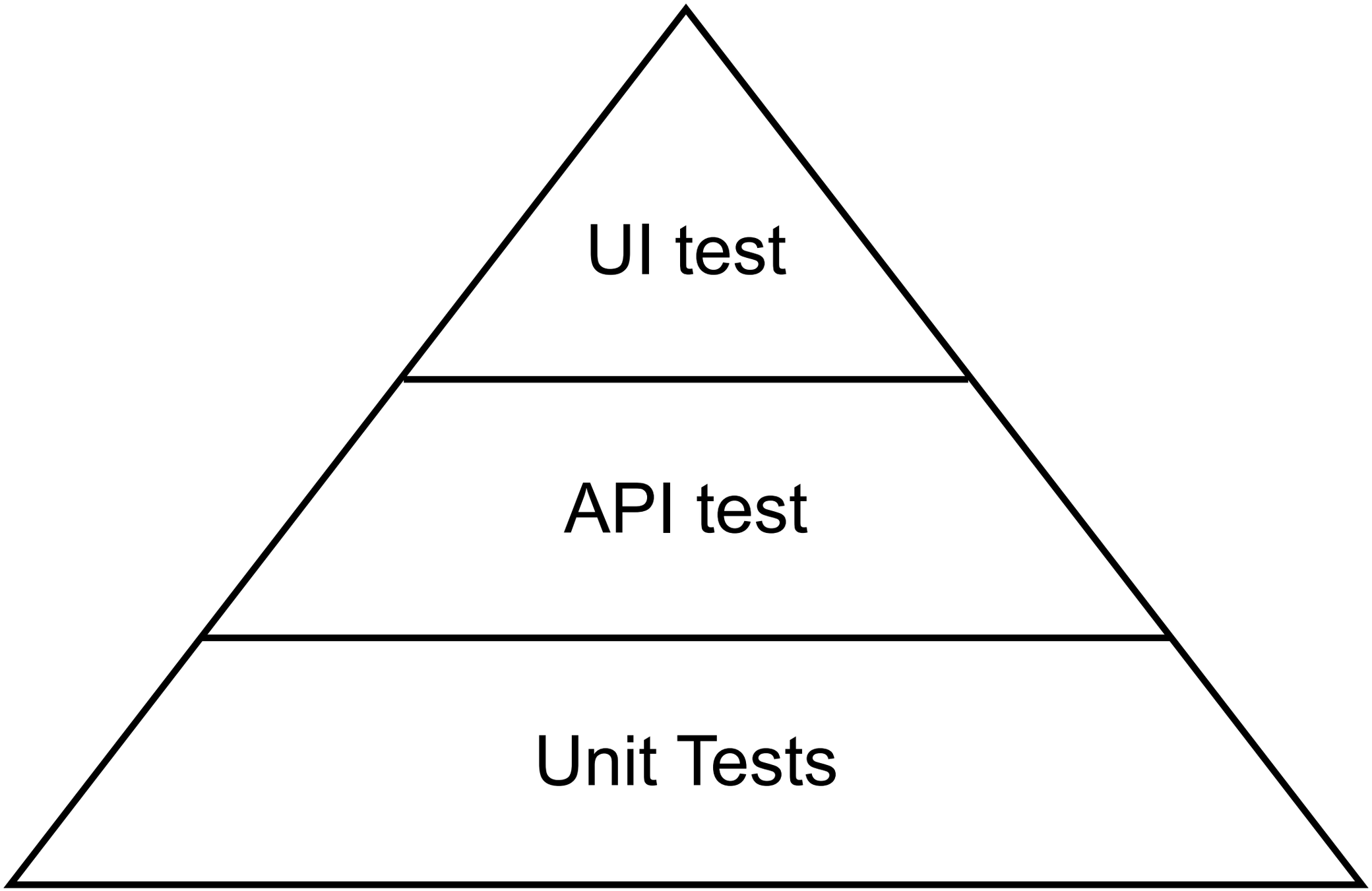
Incremental testing
Easy to test
Testable
Fast customer feedback

# Architecture ?

# Architecture



# Testing ?



# Let’ start with first test case

Single task when you are in a hurry

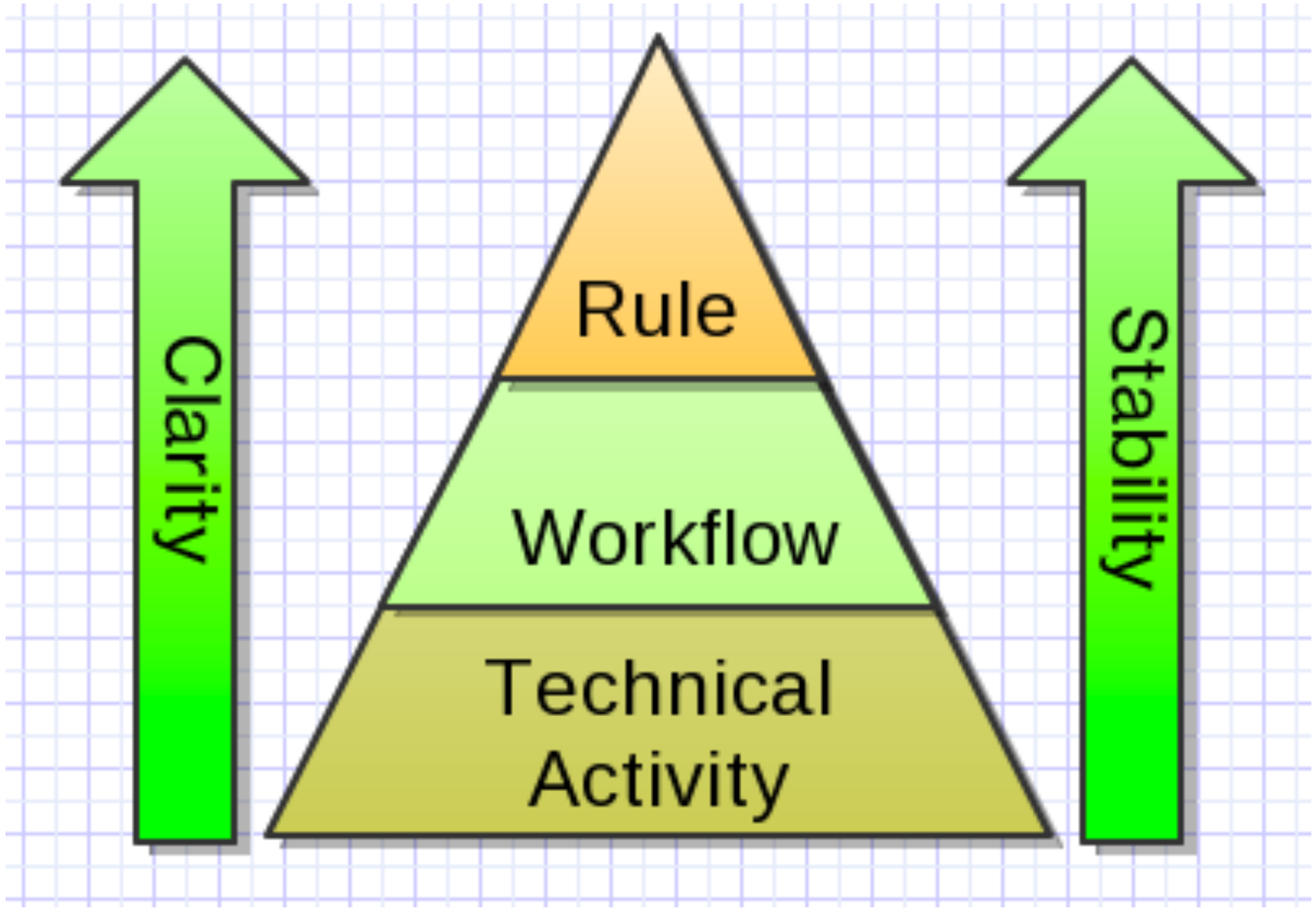
**One test at a time**

# Thinking about automation

Incremental testing
Easy to test
Testable
Fast customer feedback

# UI Test Automation

# 3 levels of UI test automation



# 3 levels of UI test automation

## Business rule/functionality level

what is this test demonstrating or exercising

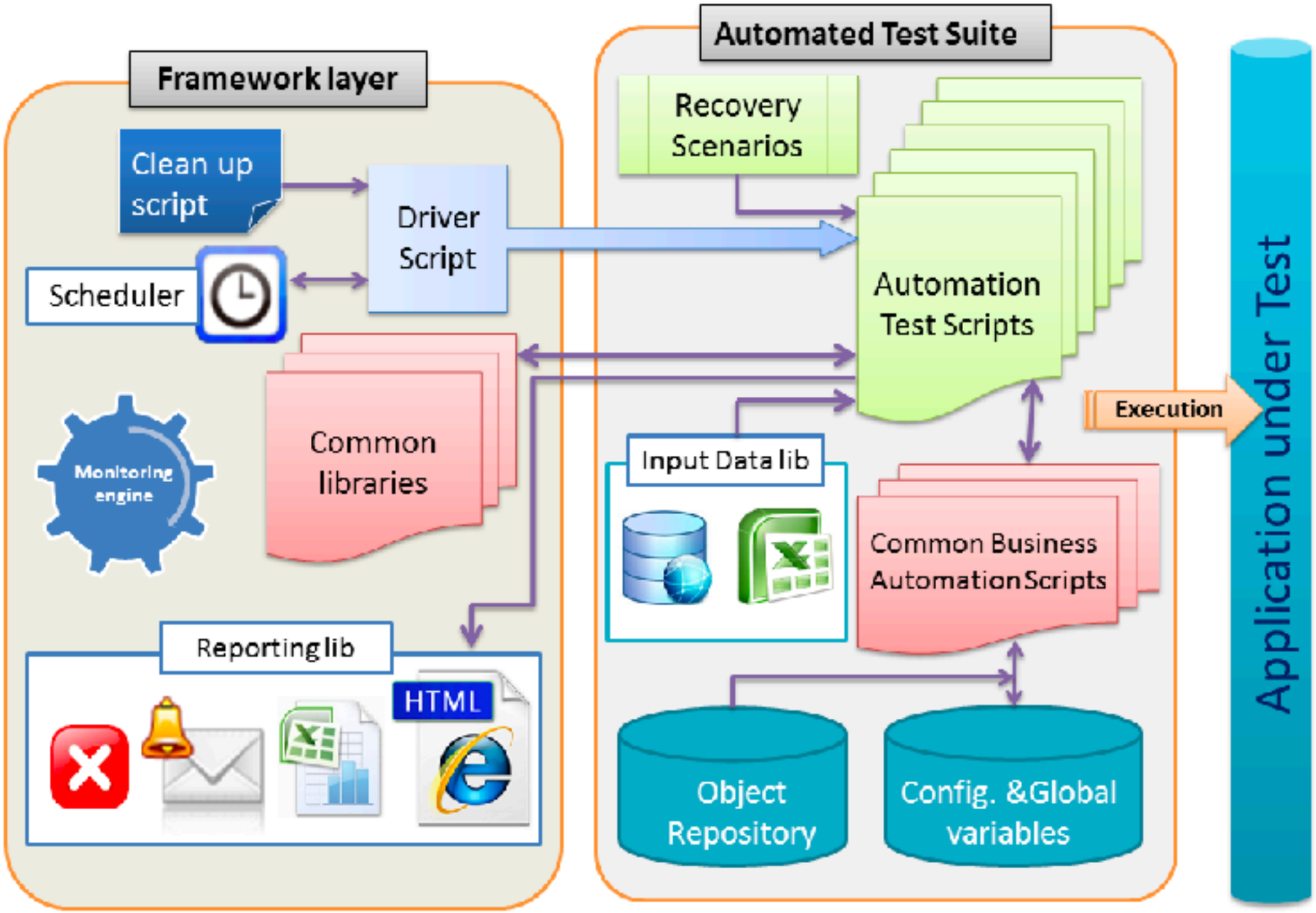
## User interface workflow level

what does a user have to do to exercise the functionality through the UI

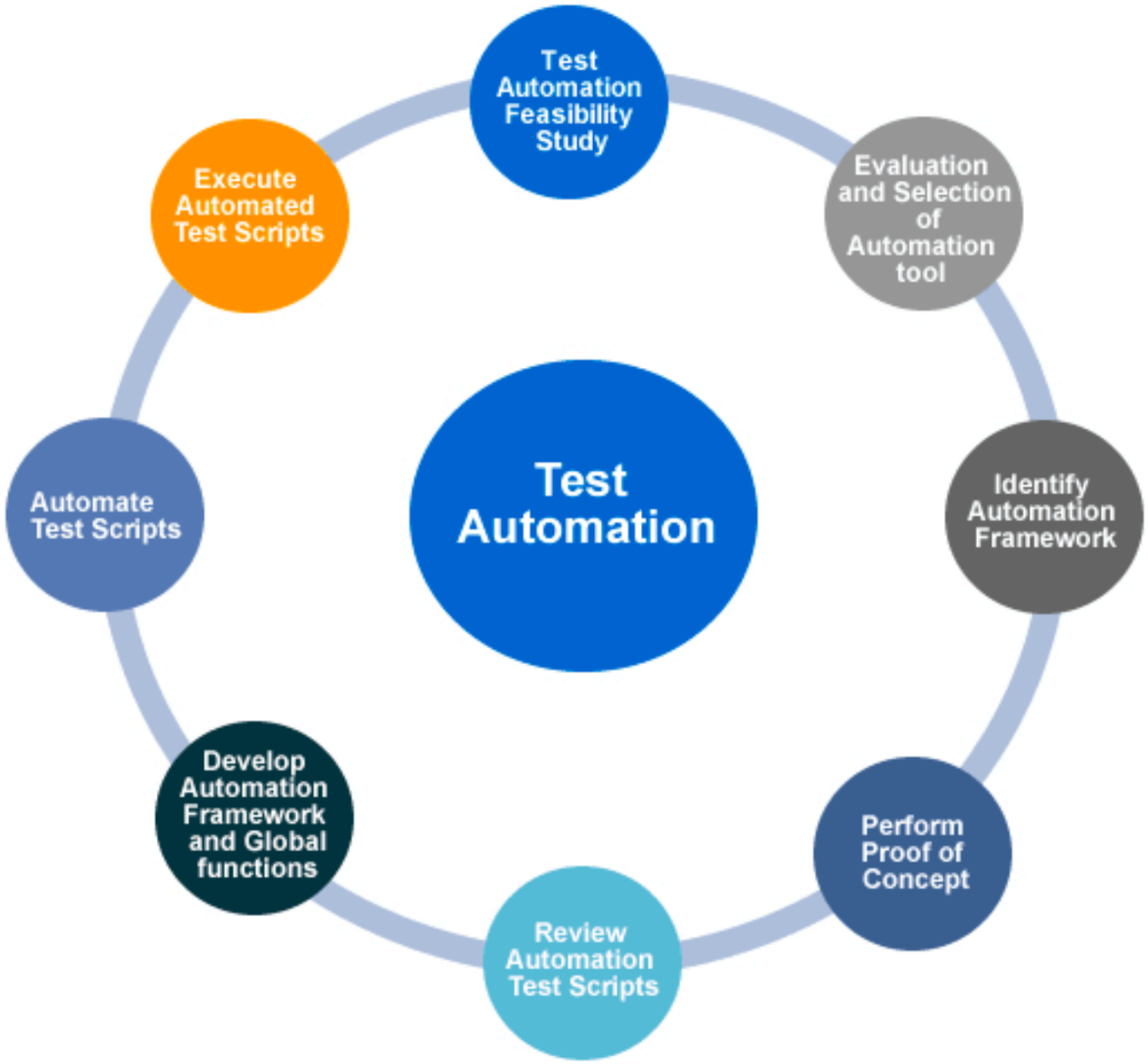
## Technical activity level

what are the technical steps required to exercise the functionality

# Test Framework

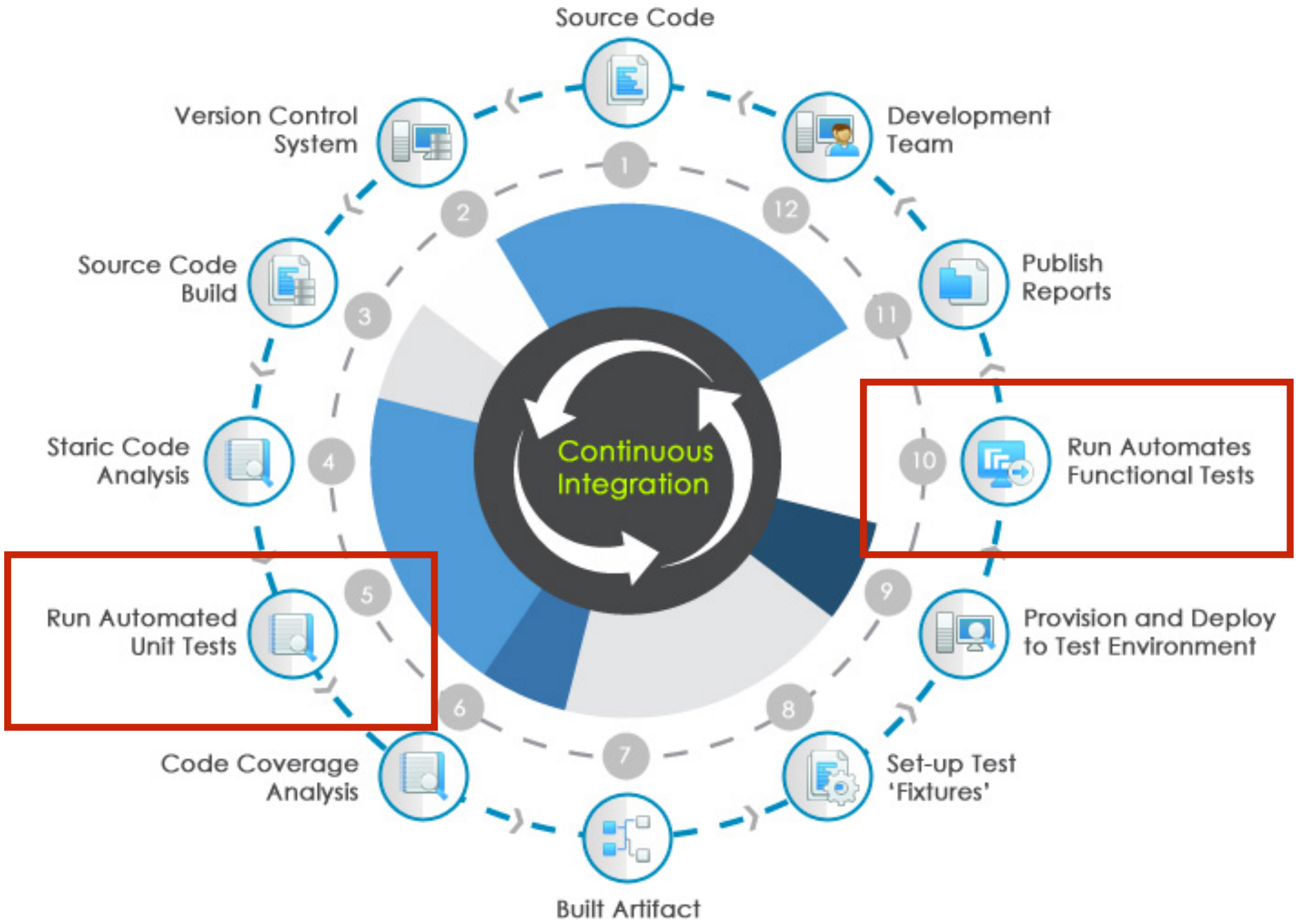


# Test Automation selection

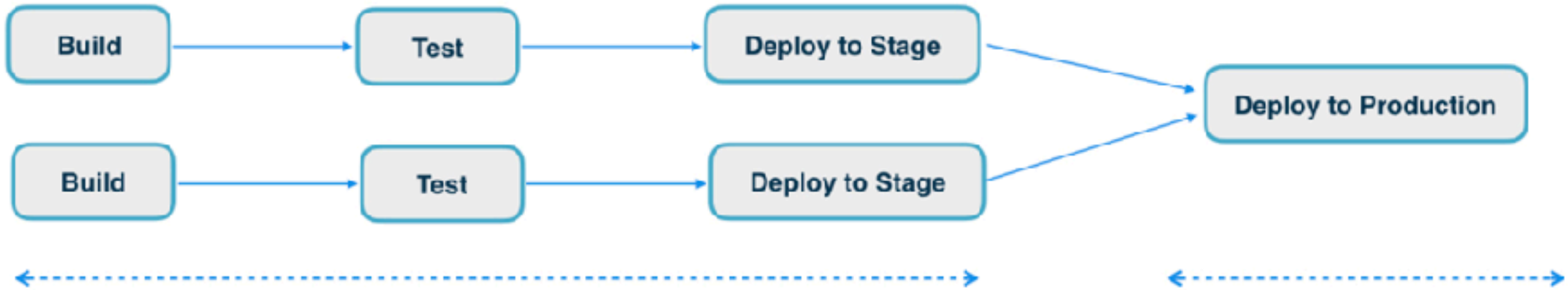


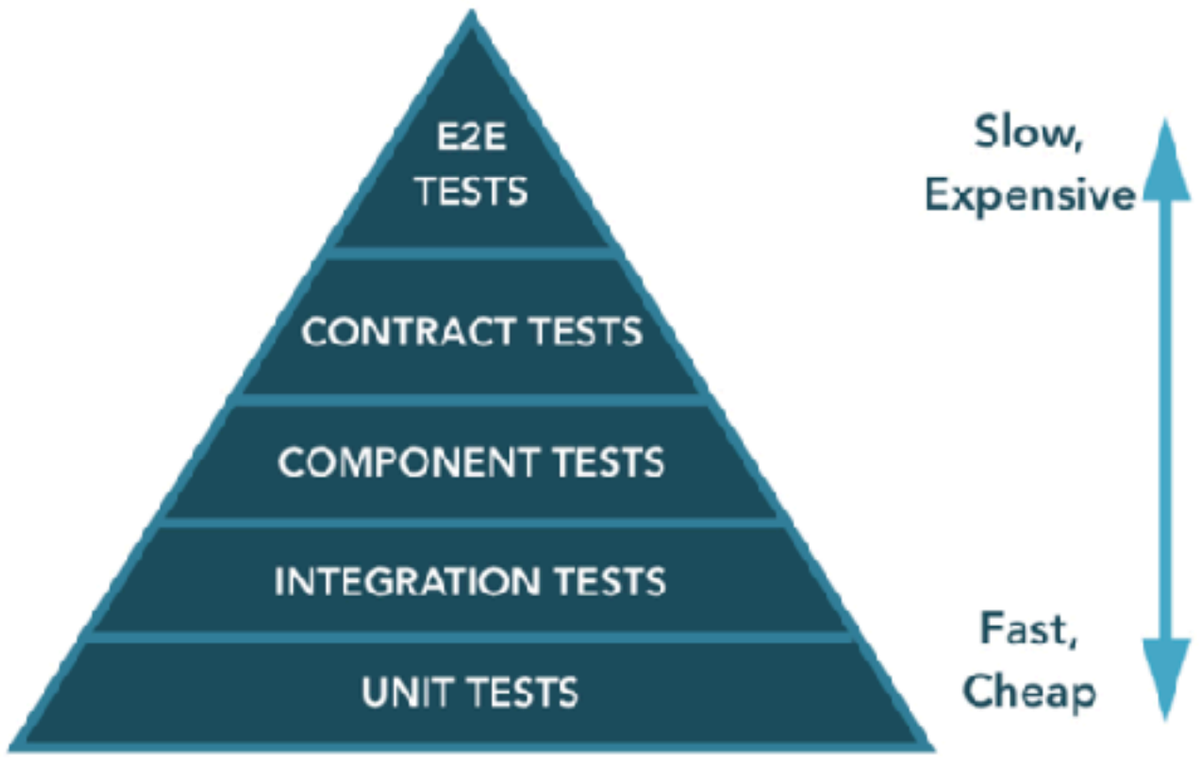
# Continuous Integration

# Continuous integration



# Test strategy



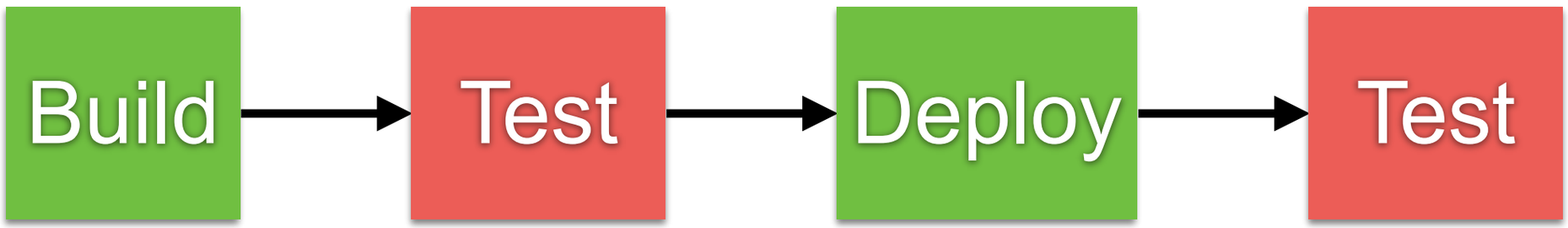




Monitoring
Distributed Tracing
Fault-injection Testing

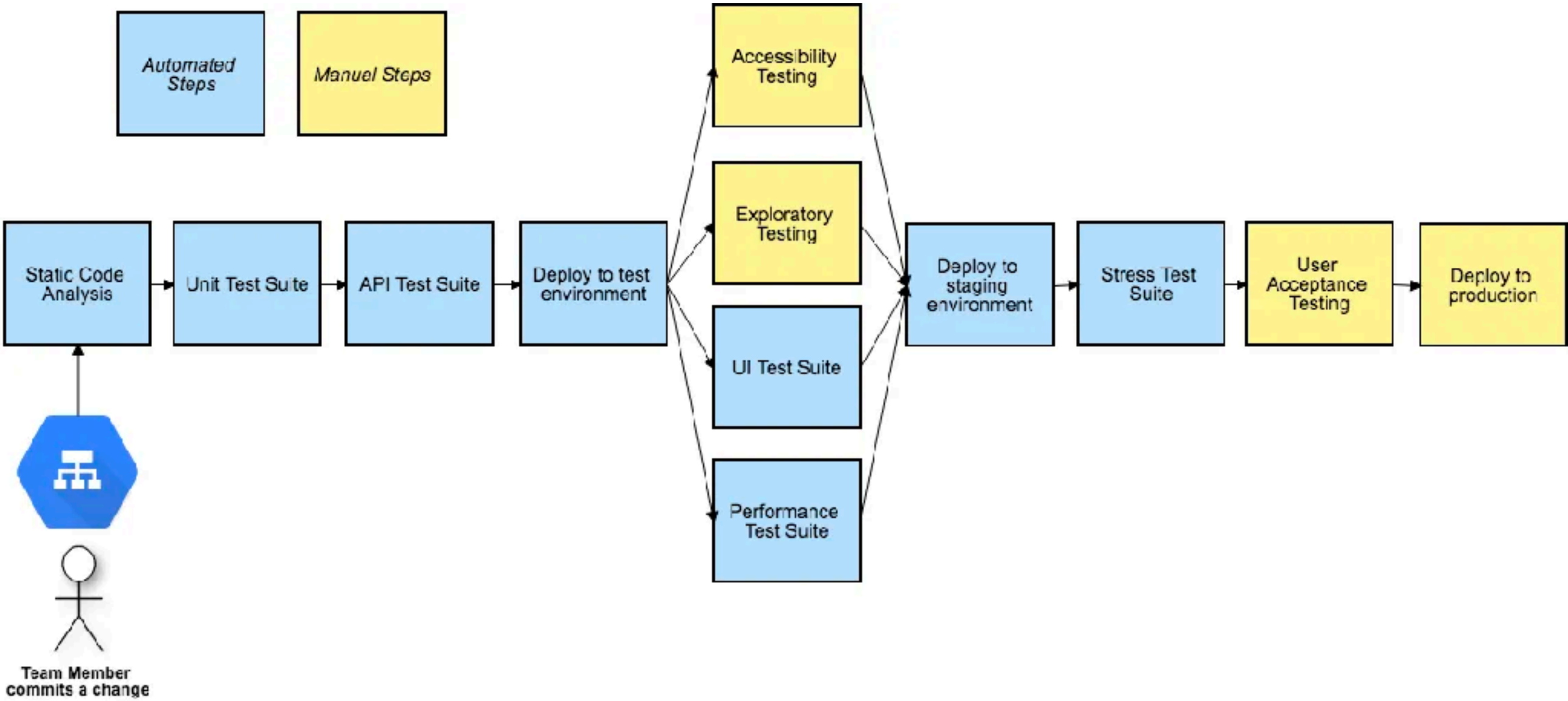
---

Canary Deployments
Blue-Green Deployments
A/B Testing

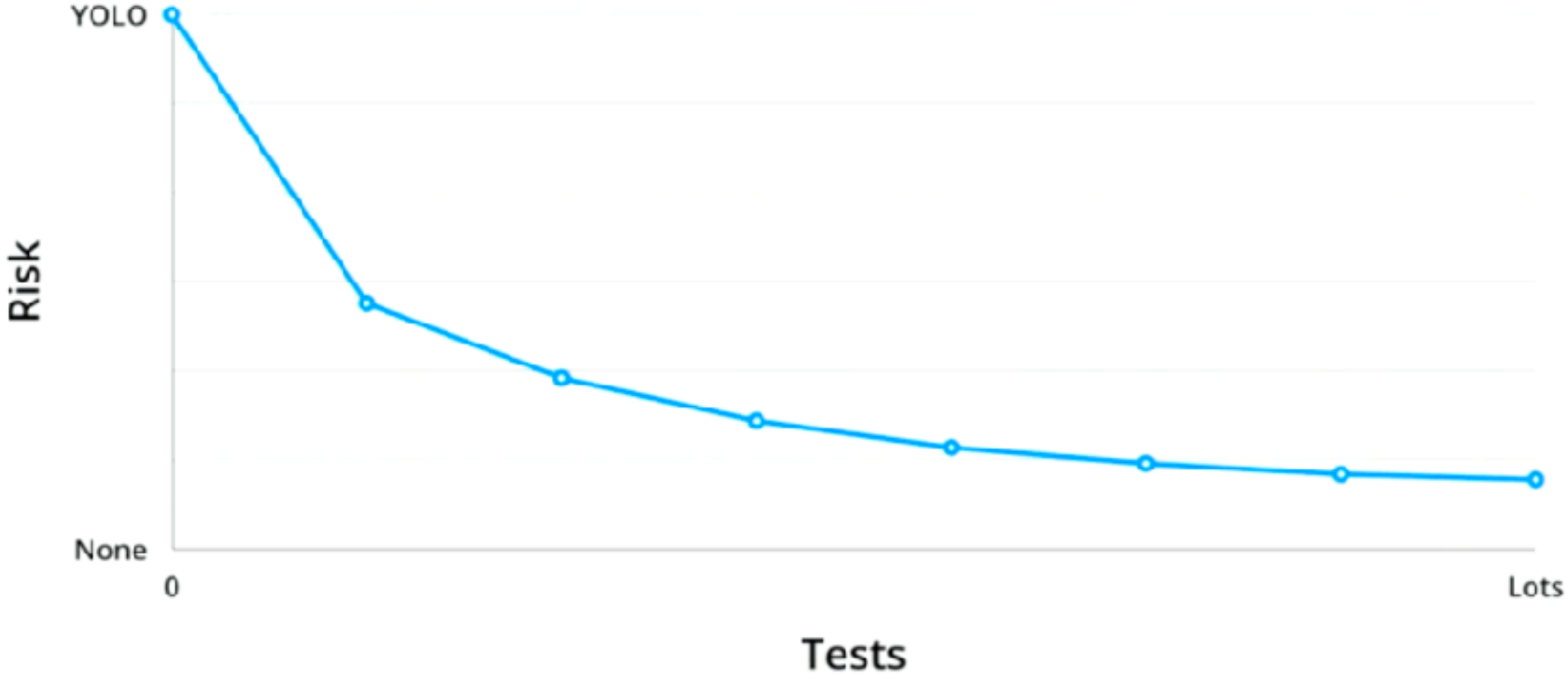
# Design your pipeline



# Design your pipeline/process



# Reduce risk with tests



# Q/A