



**Develop High Quality Software
with Automation Testing**



Develop High Quality Software

With Automation Testing



Topics

Software Development Life Cycle
Software quality problem
Test-first vs Test-last vs Test later
Testing strategies
Good testing
Manual to Automated testing
Continuous testing



Learning path



UI

API

Testing tools

Testing strategy

Problems in Software
Development



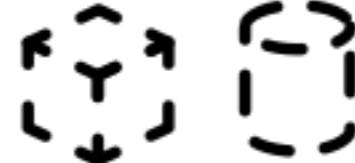
Software Delivery



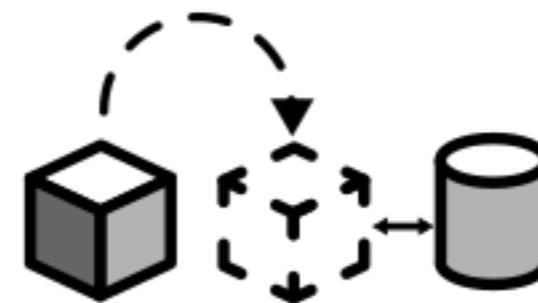
Build



Test



**Provide
Infrastructure**



Deploy



**Test
More**

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



Technical excellence



SPECIFICATION BY EXAMPLE



CONTINUOUS
INTEGRATION



CONTINUOUS DELIVERY



TEST AUTOMATION



TECHNICAL
EXCELLENCE



THINKING ABOUT TESTING



TEST-DRIVEN DEVELOPMENT



ARCHITECTURE
& DESIGN



ACCEPTANCE
TESTING



CLEAN CODE



UNIT TESTING

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



Start with Why ...



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

Edsger W. Dijkstra, 1970, Notes on Structured Programming

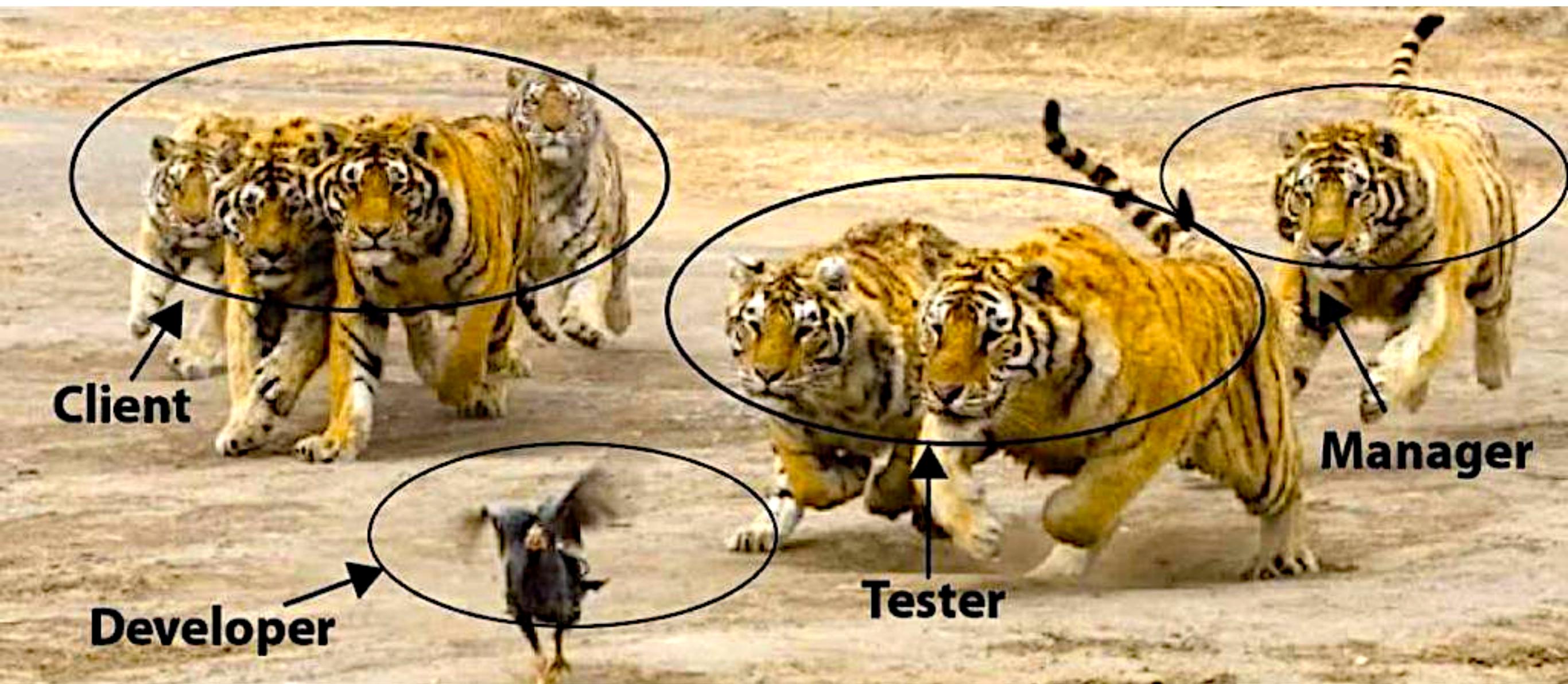


Deadline Driven Development

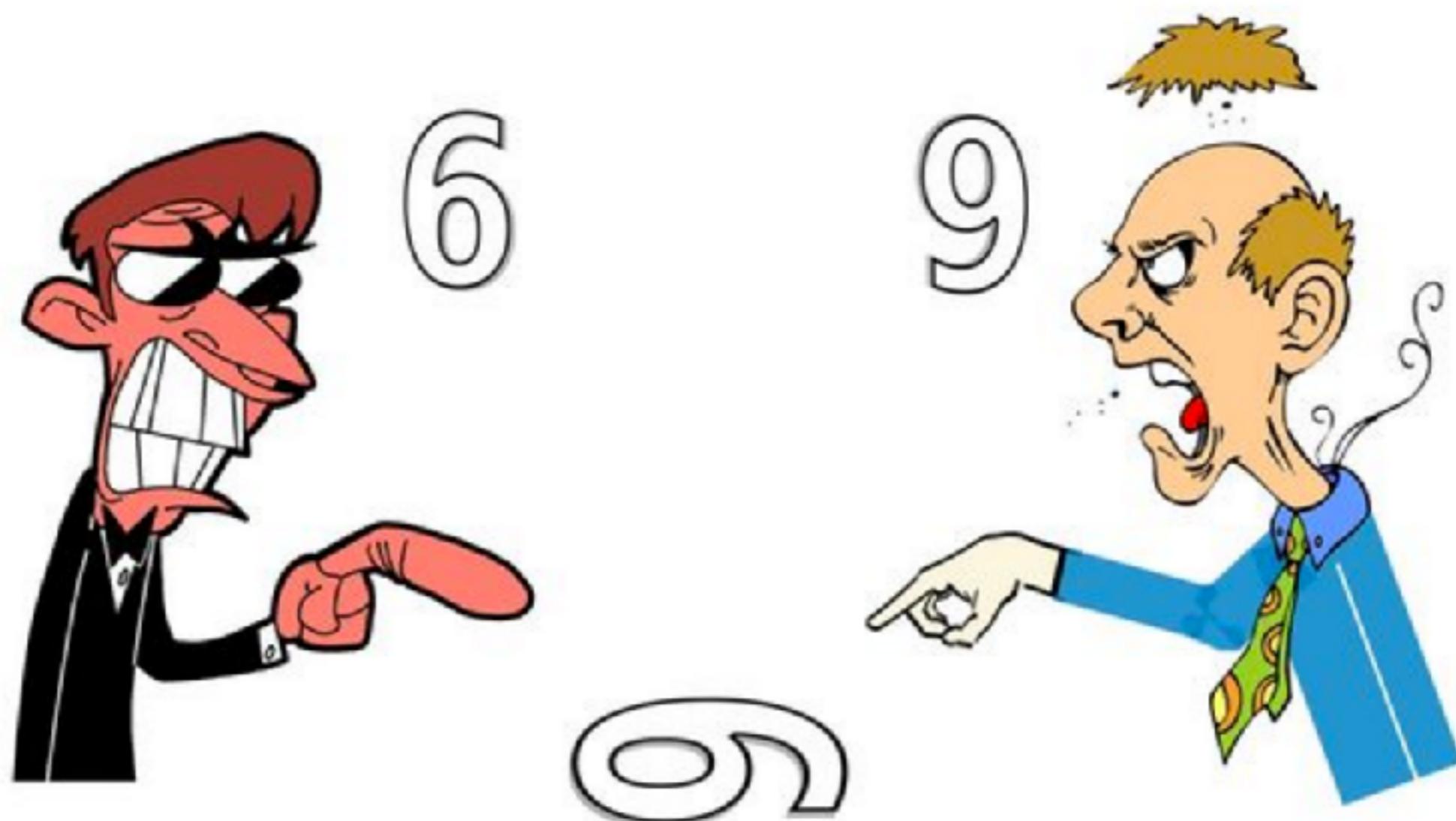


Quality vs. Quantity





Developer vs Tester



Every time a developer changing the code !!





TONIGHT WE TEST

IN PRODUCTION!!!

memegenerator.net

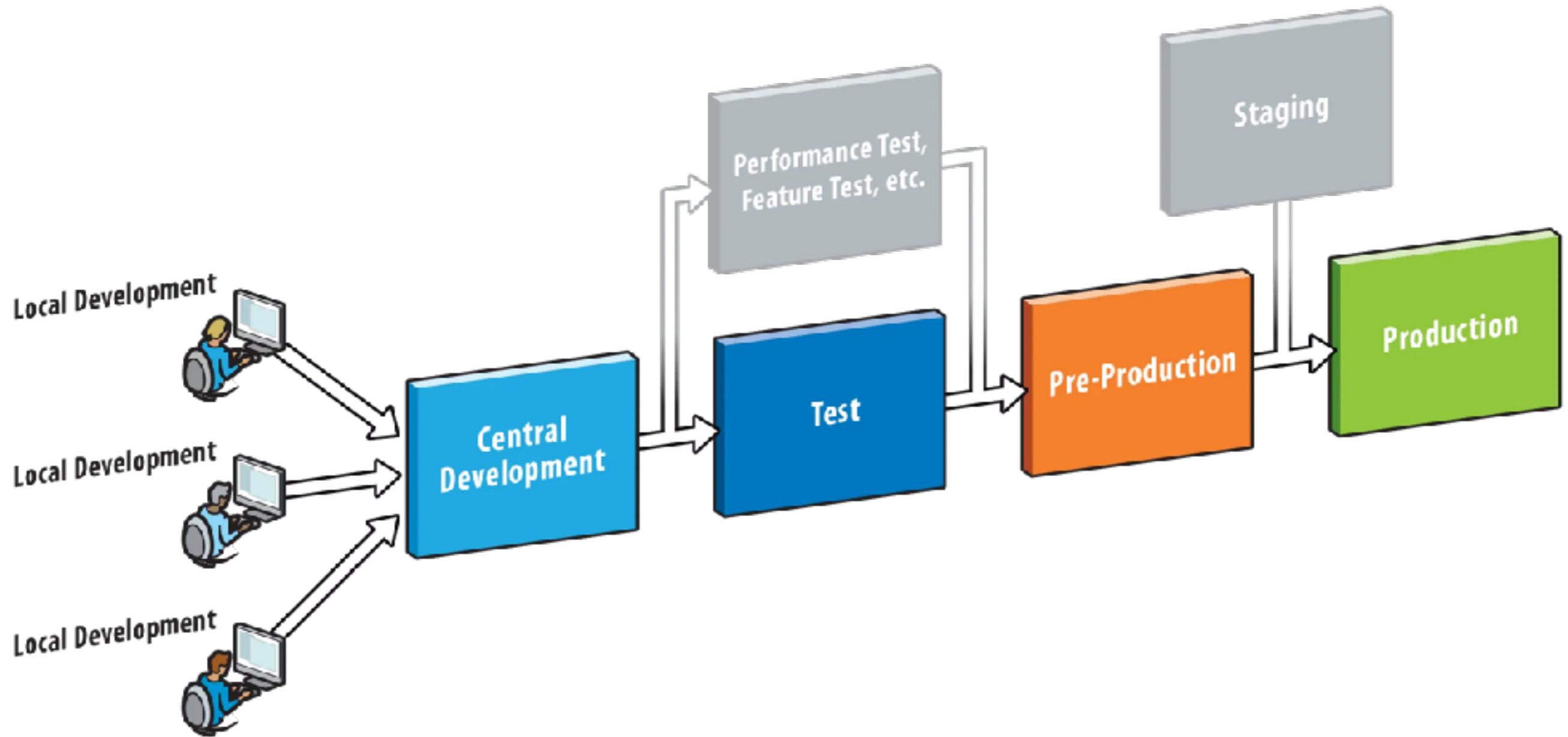




<https://davidwalsh.name/production-testing>

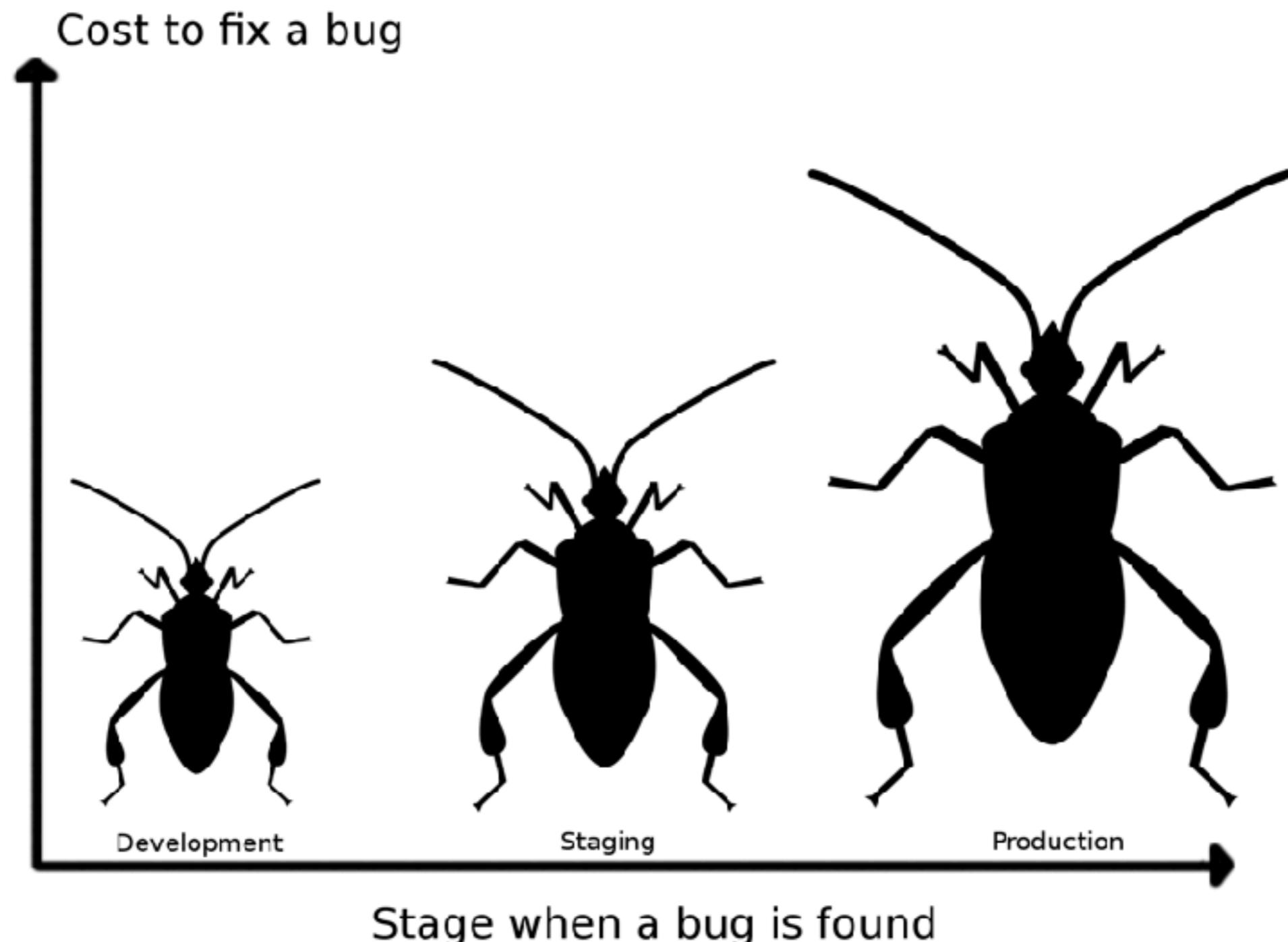


Environments in software delivery



Production != Other





Why we need to test ?



Why we need to test ?

Help you to **catch bugs**

Boosted confidence

Quality code

Enforce **modularity** of your project

Develop features **faster**

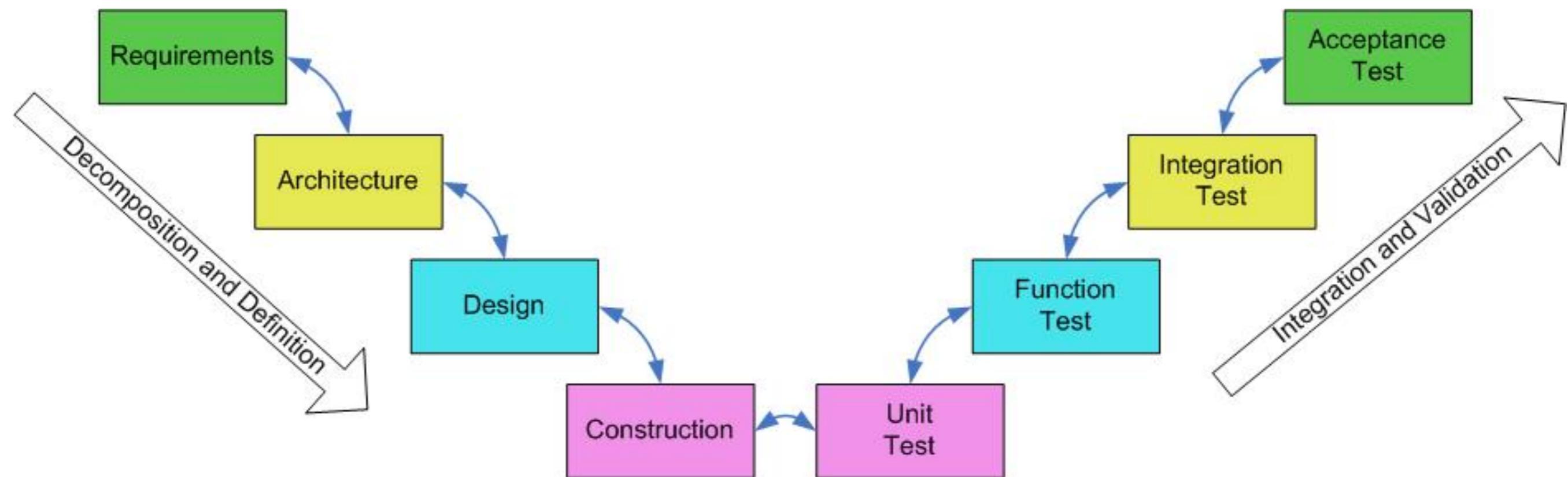
Better documentation



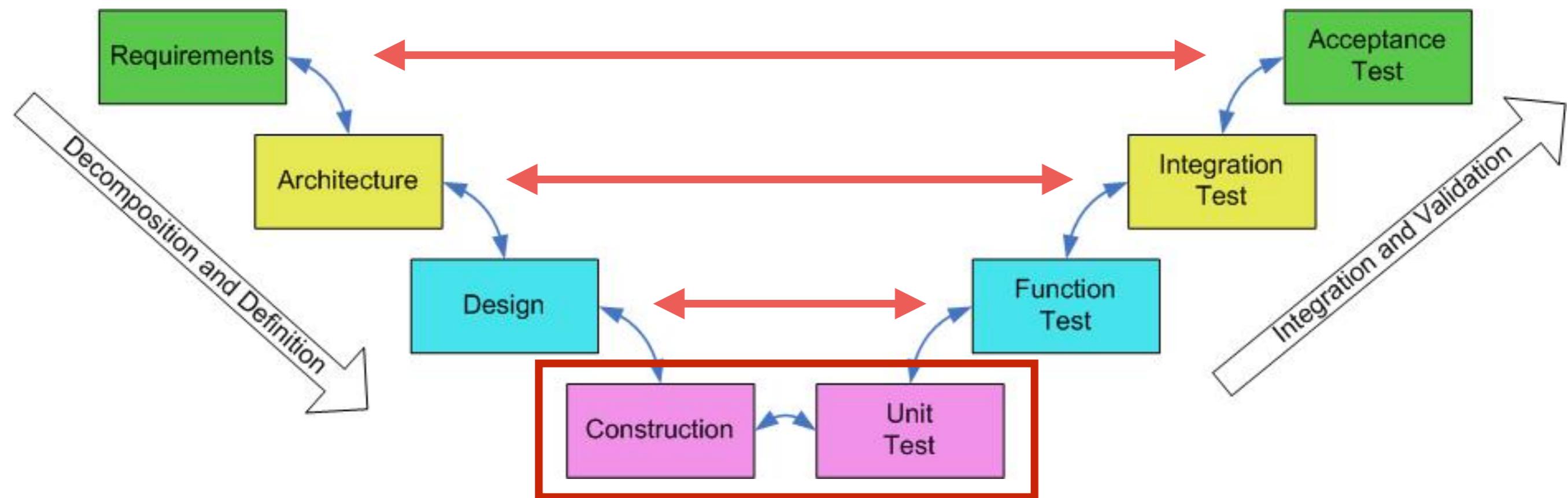
What kind of test should we write ?



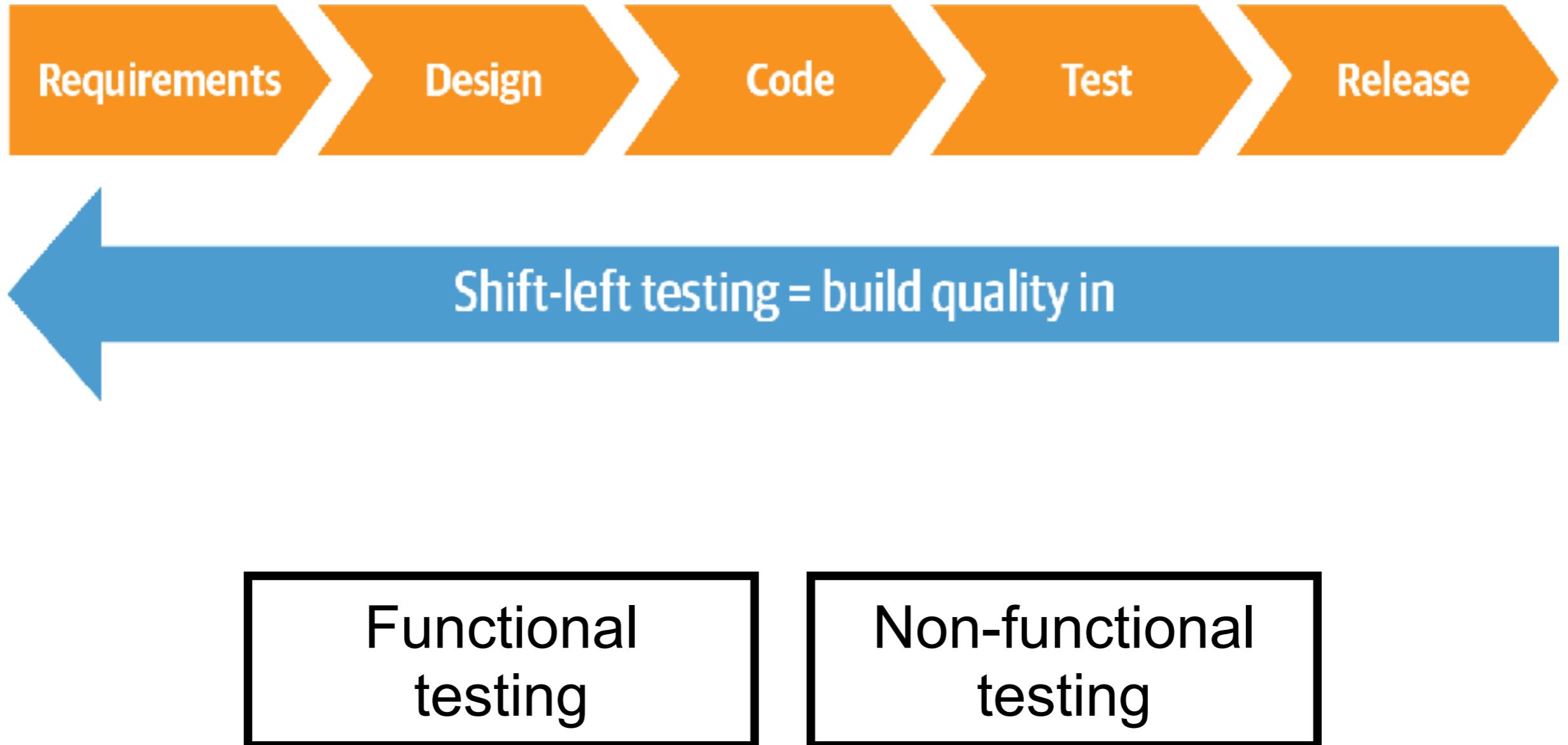
V Model



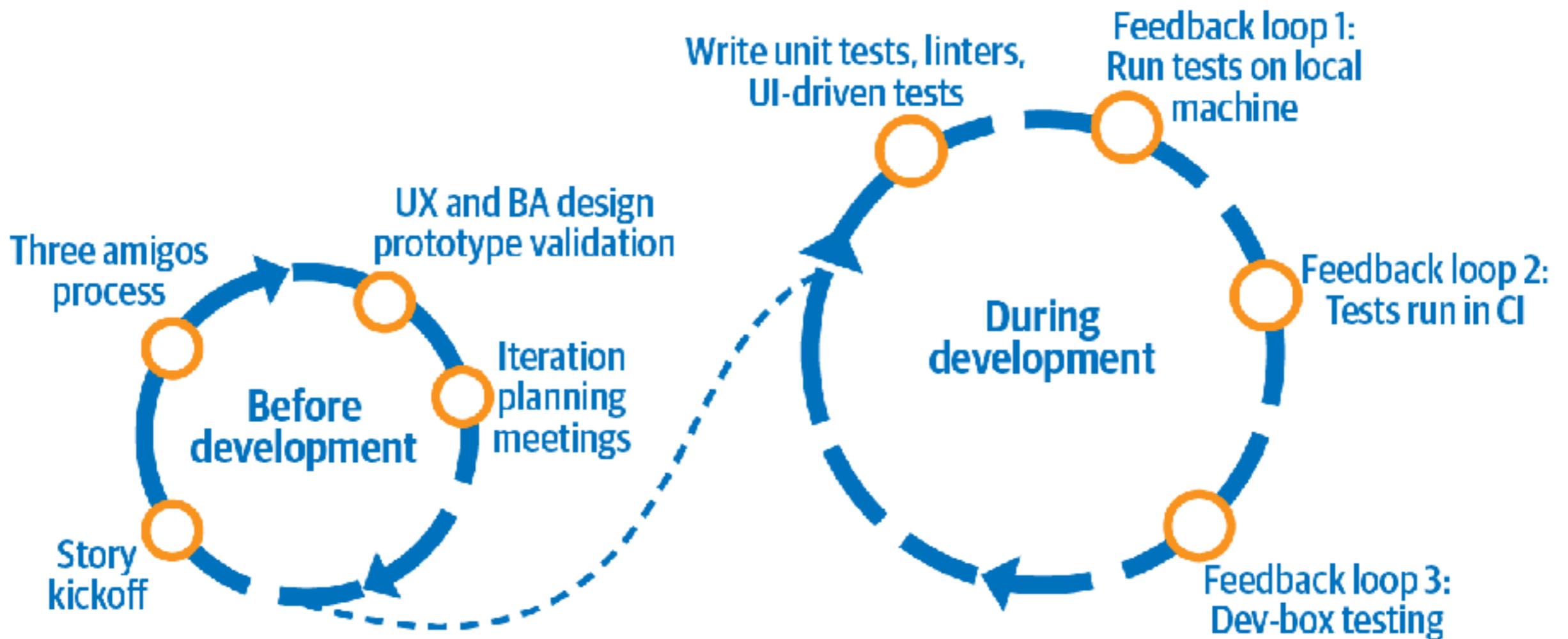
V Model



Shift Left Testing



Quality Check ?



Software quality attributes

Correctness

Flexibility

Reusability

Efficiency

Usability

Interoperability

Reliability

Testability

Cost-effective

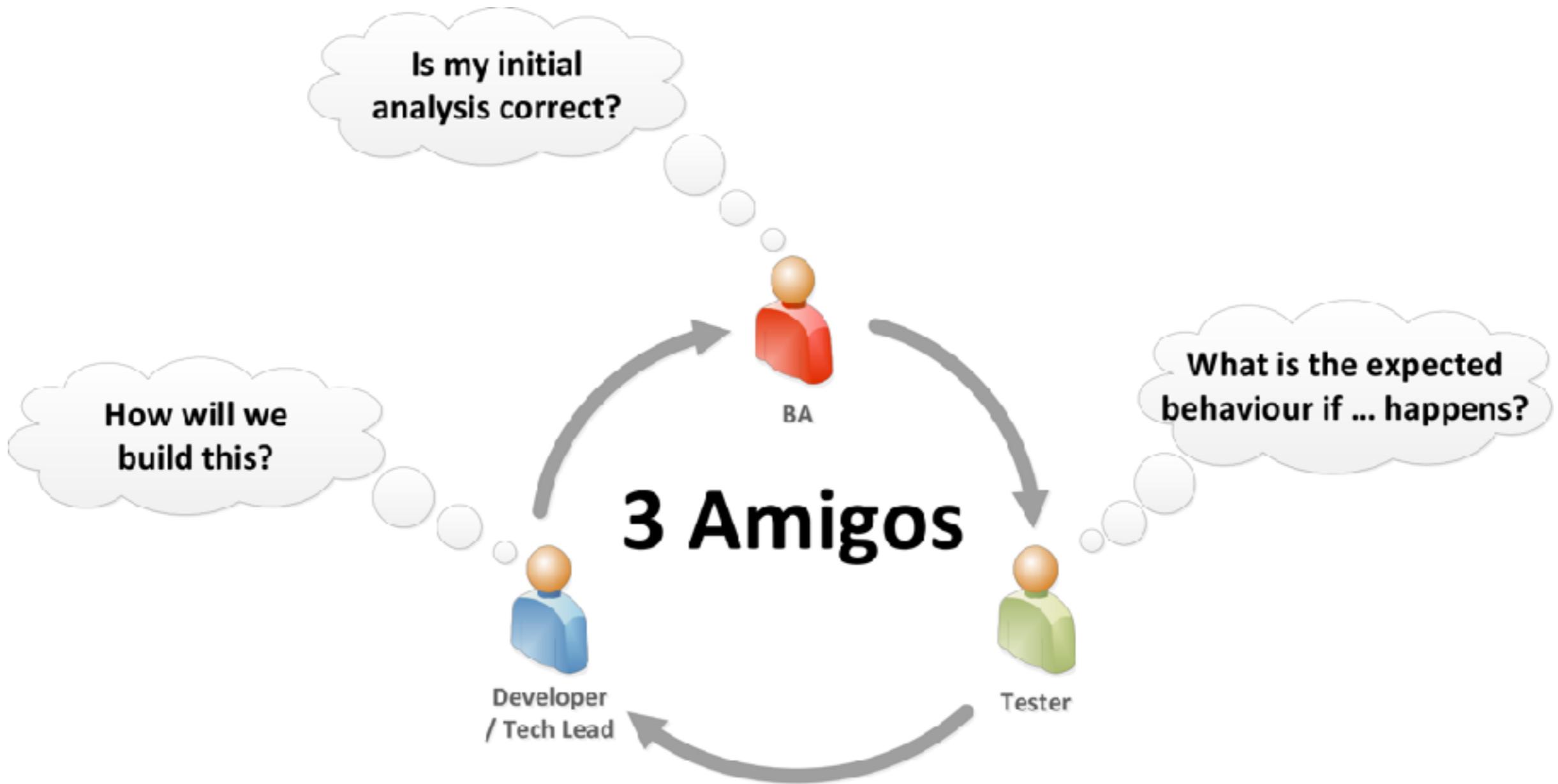
Integrity

Portability

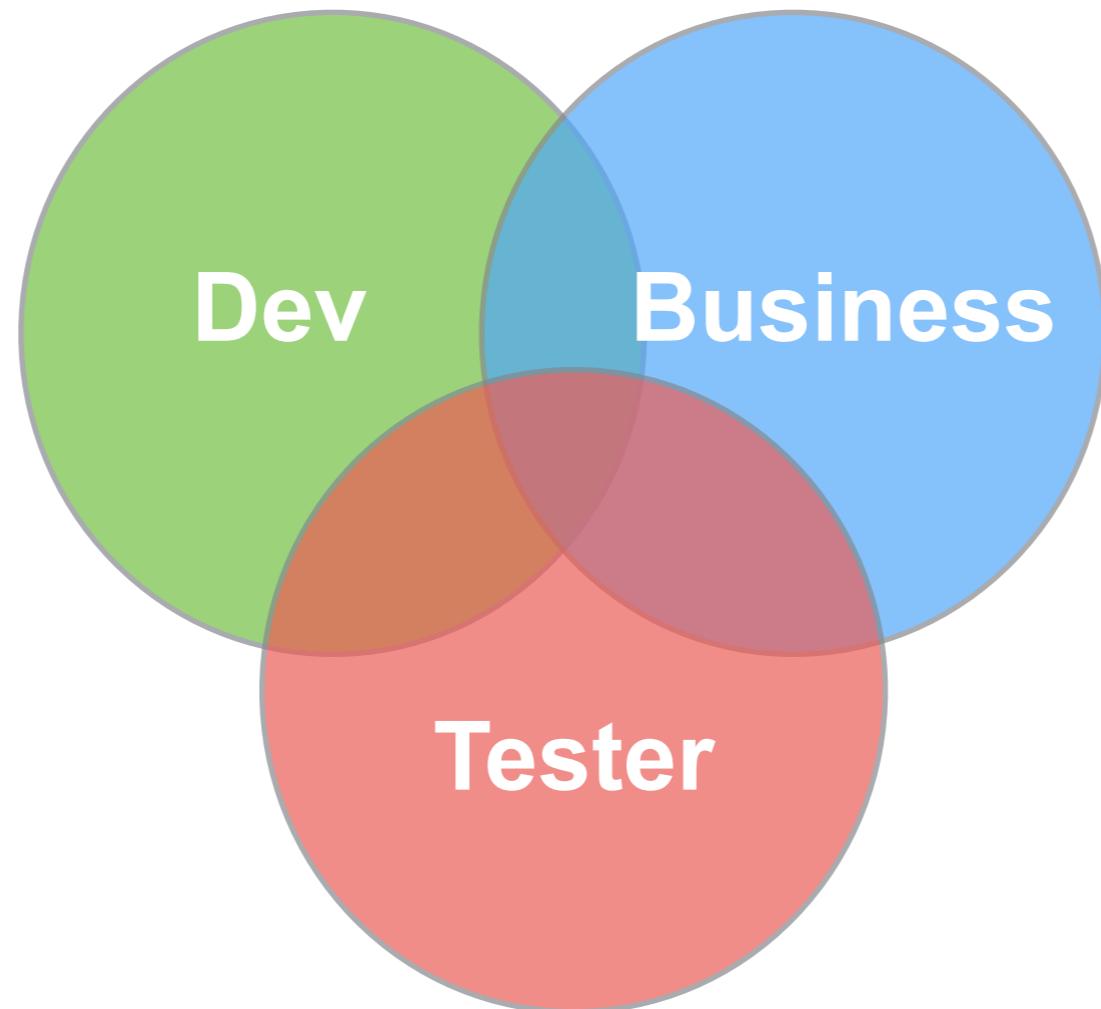
Maintainability



Three Amigos Process



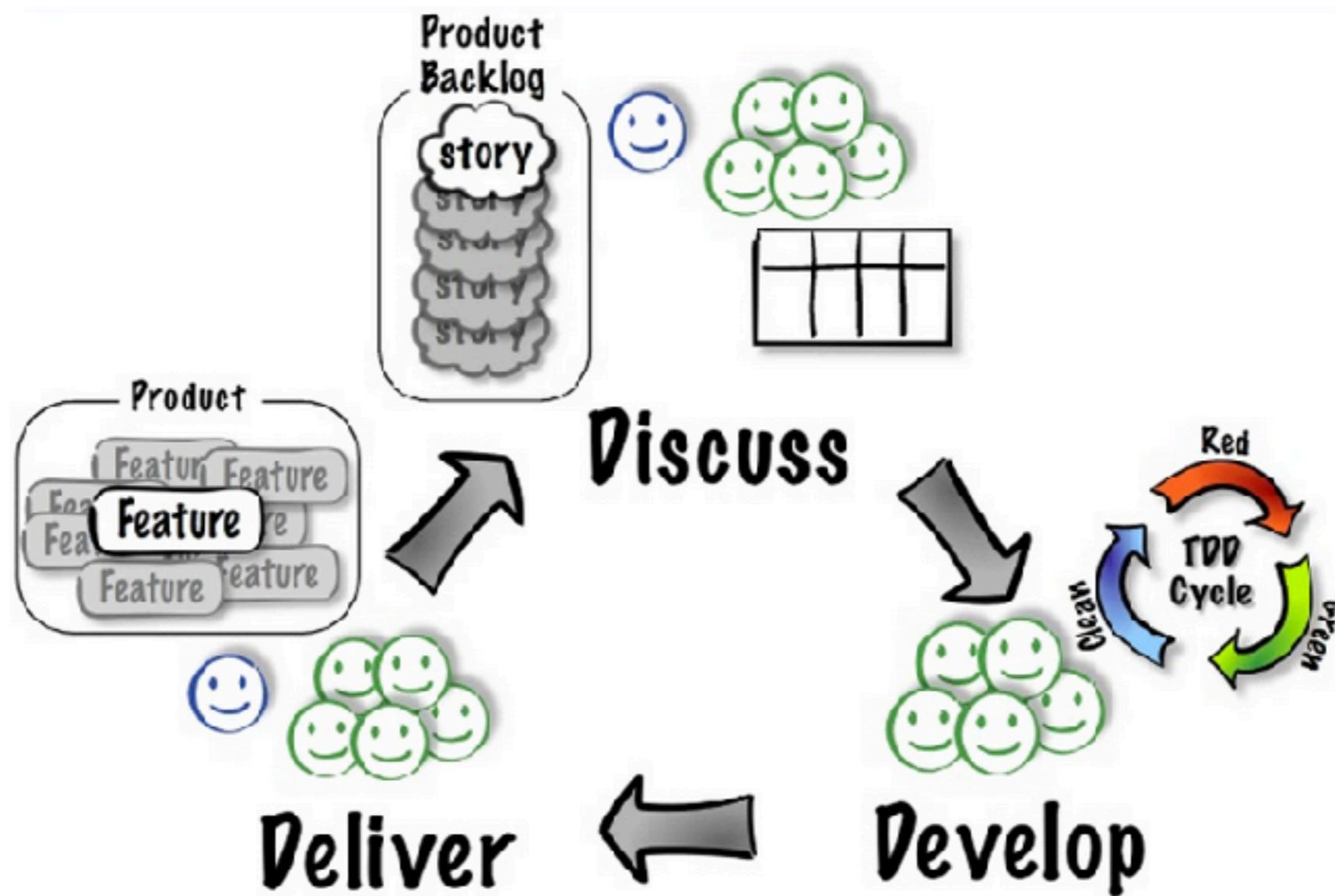
Power of Three



THINK before coding



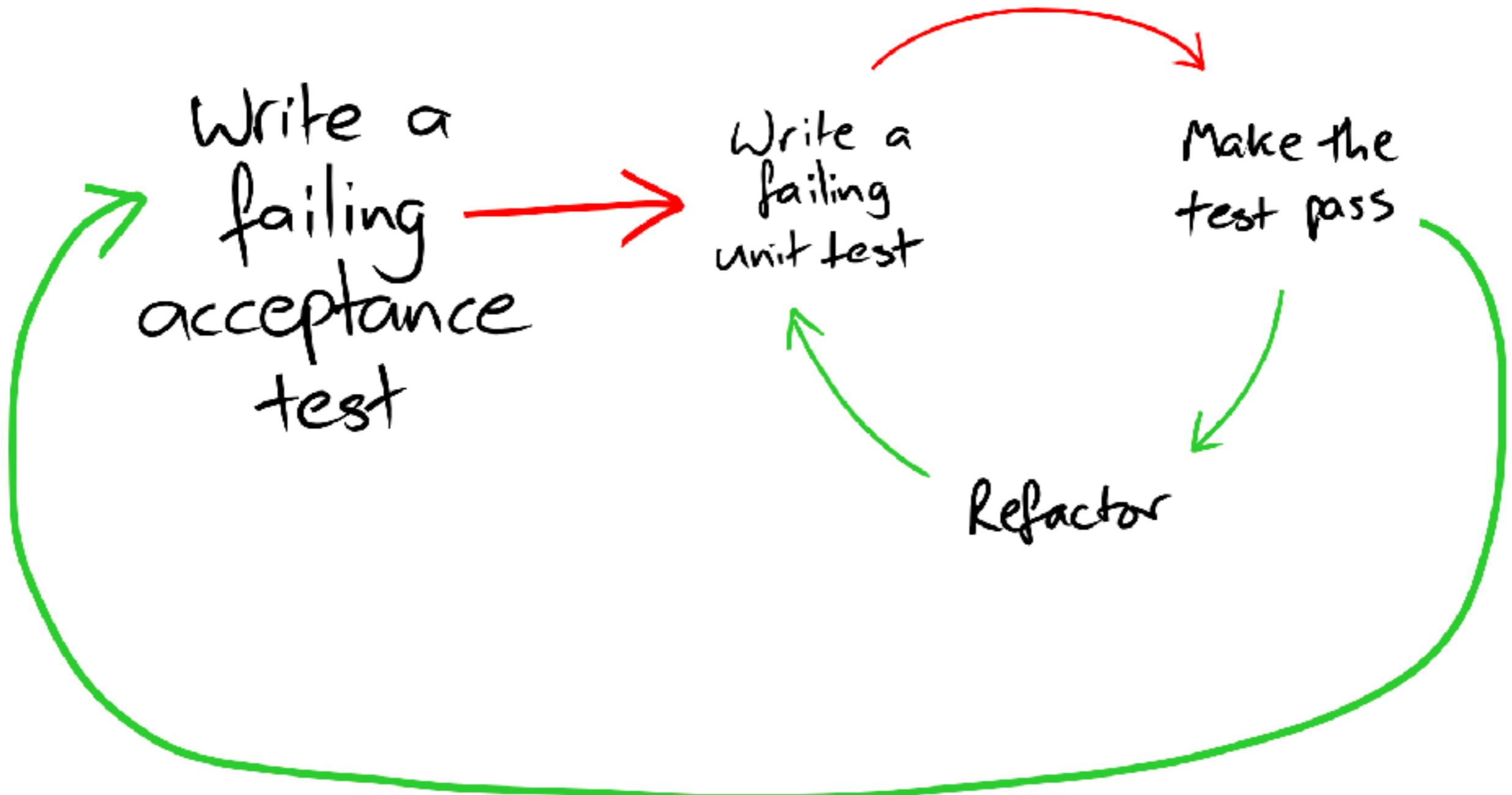
Acceptance Test-Driven Development



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



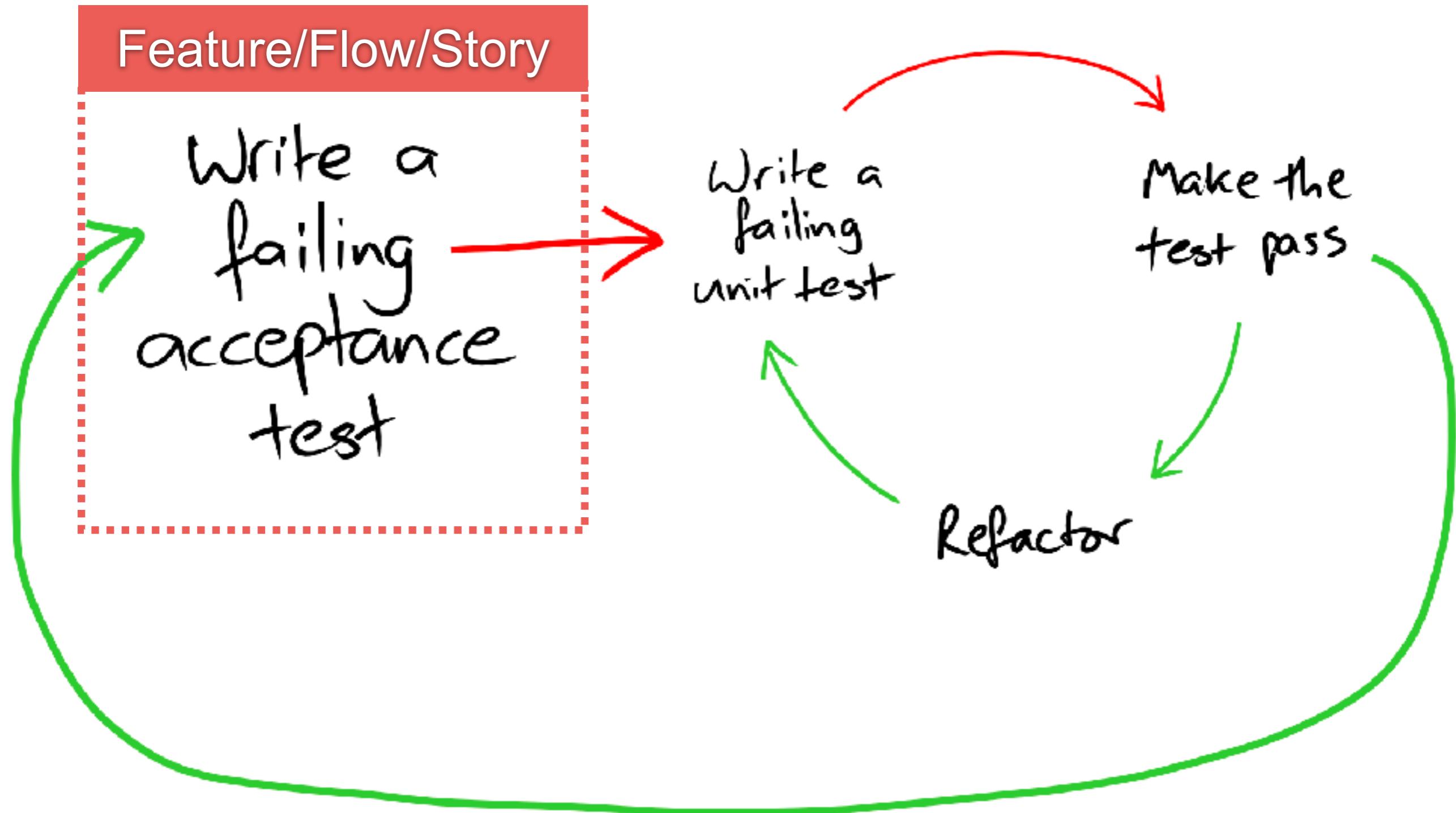
Outside-in develop/test



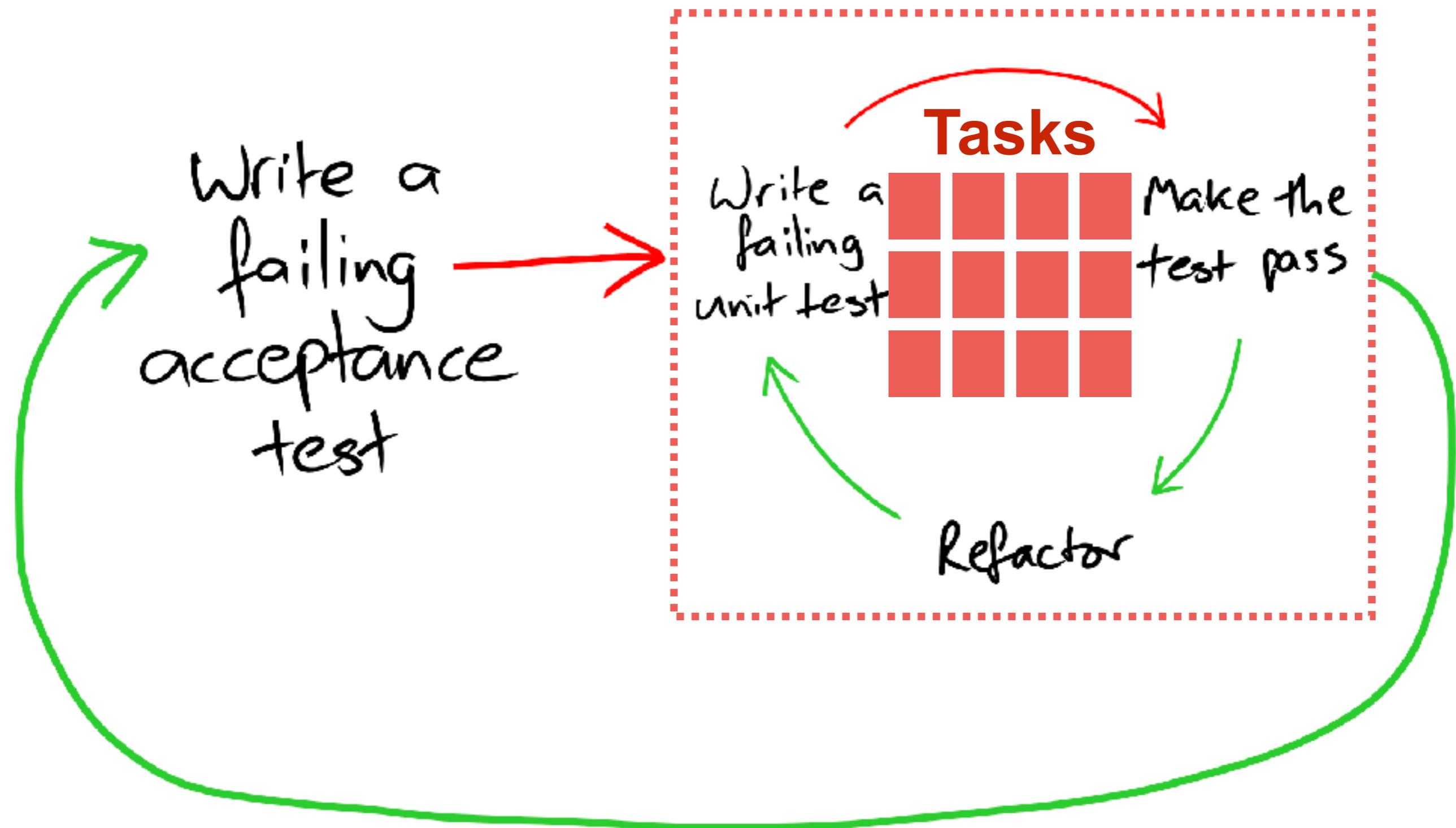
<http://www.growing-object-oriented-software.com/>



Outside-in develop/test



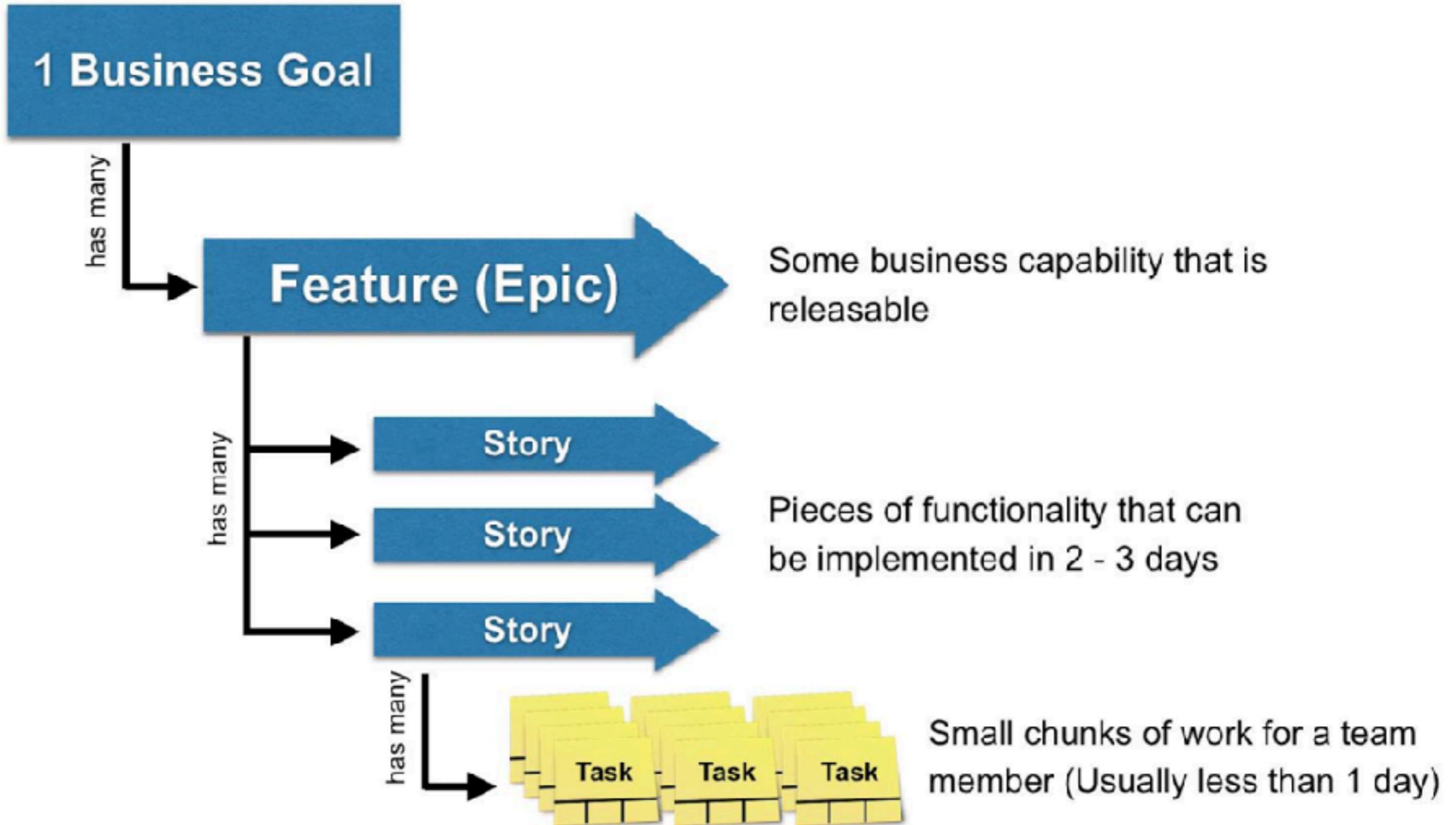
Outside-in develop/test



Tests Cases
= **Business Criteria**
+ **Examples (real data)**



Work break down



Key success factors

Whole team solve problems

Whole team thinks about testing

Whole team committed to quality

Everyone collaborates

Improve technical skills

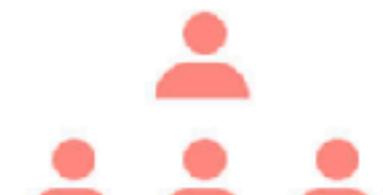
Better testing process (improvement)



Whole team approach

Functional

Common functional expertise



System analysts



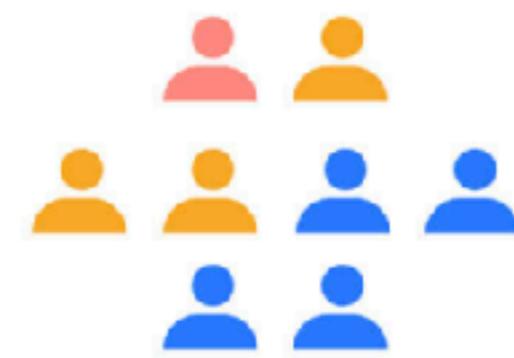
Developers



Testers

Cross - Functional

Representatives from the various functions



Development Team



Iterative and Incremental process



Iterative and incremental process

Feature 1 is Done ?

Feature 1

Time



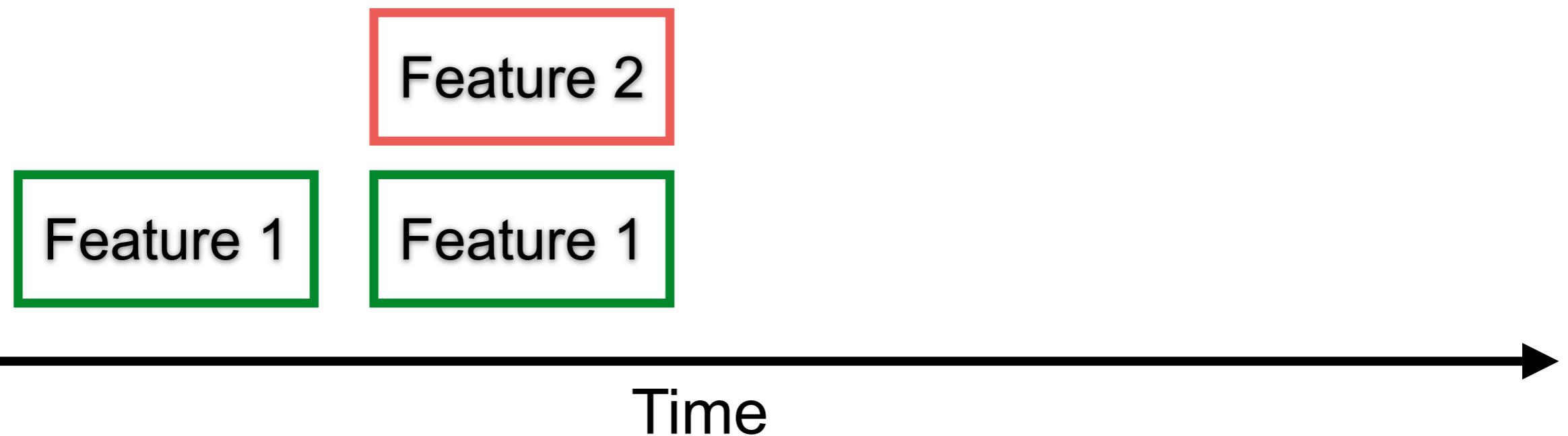
Iterative and incremental process

Done = coded and tested



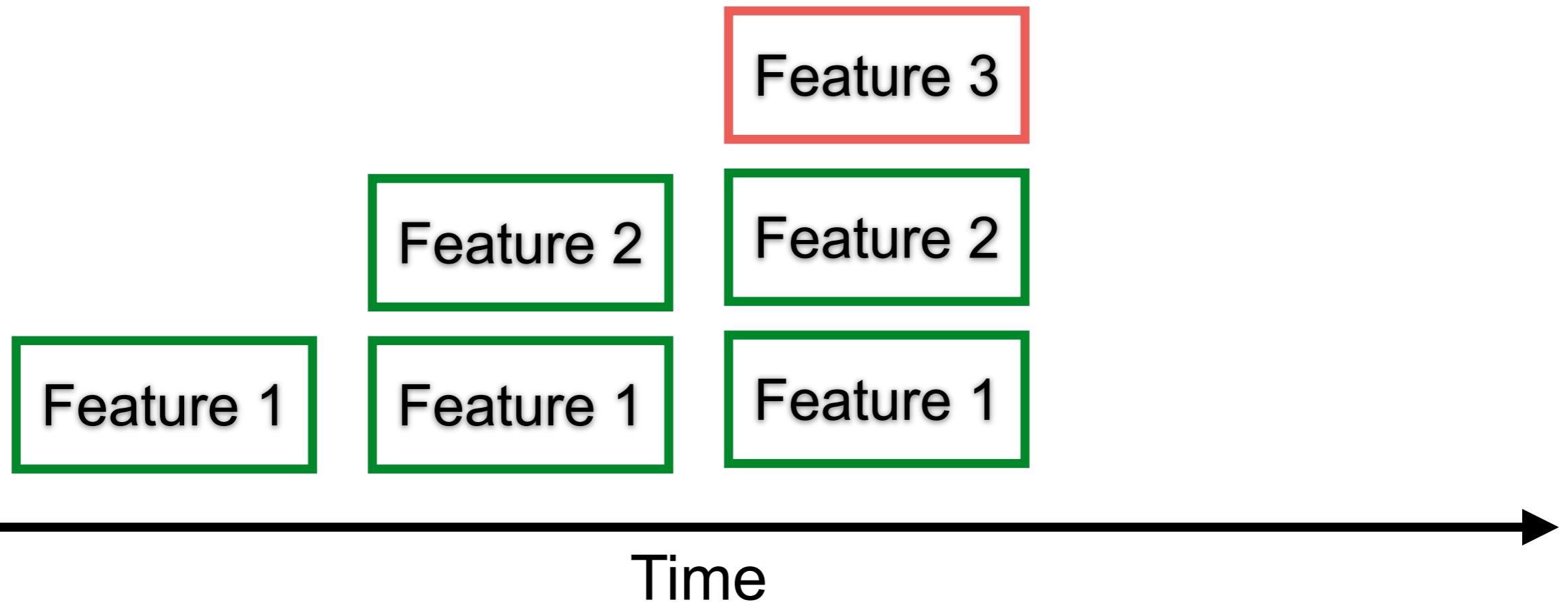
Iterative and incremental process

Done = coded and tested



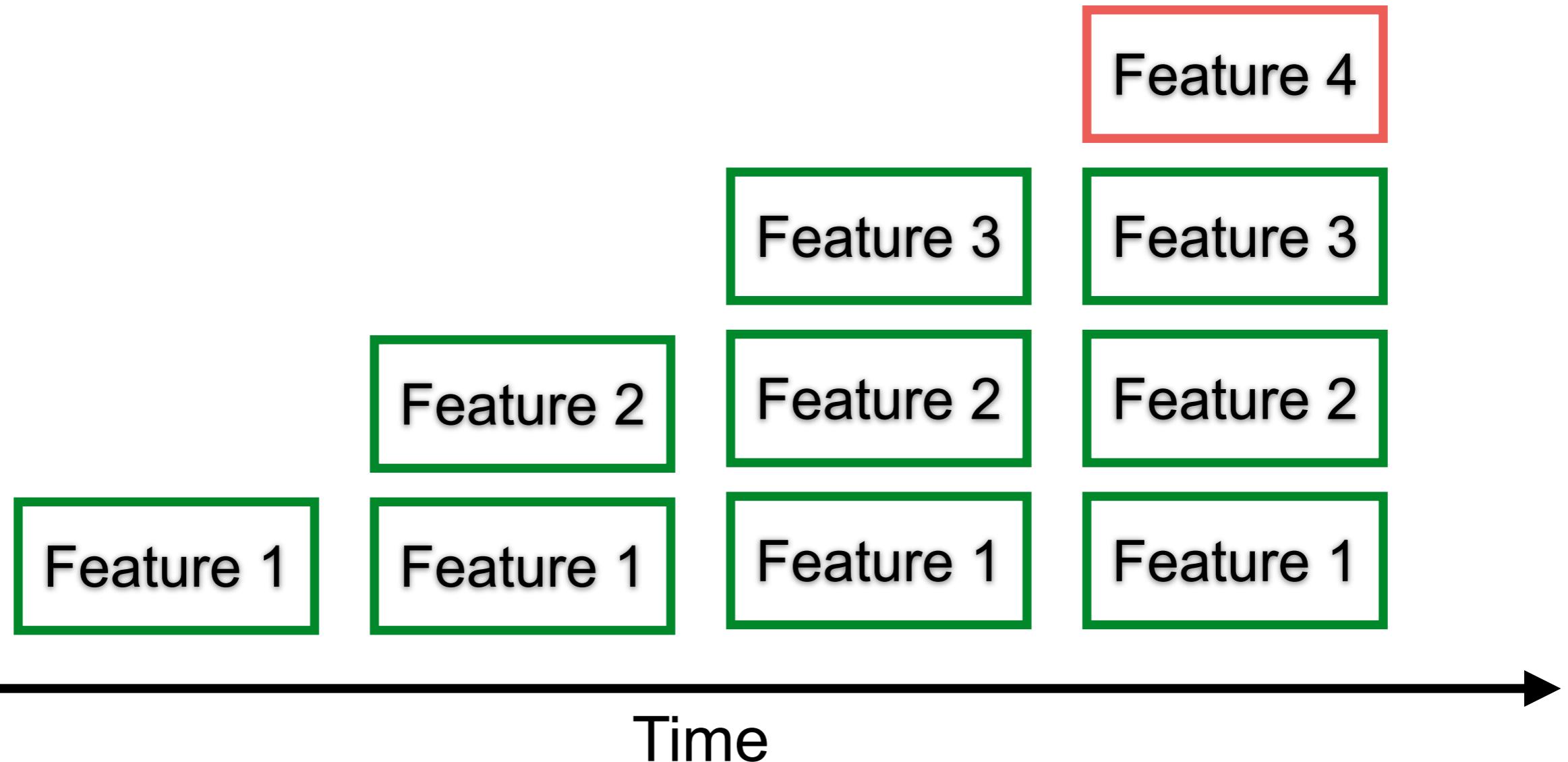
Iterative and incremental process

Done = coded and tested



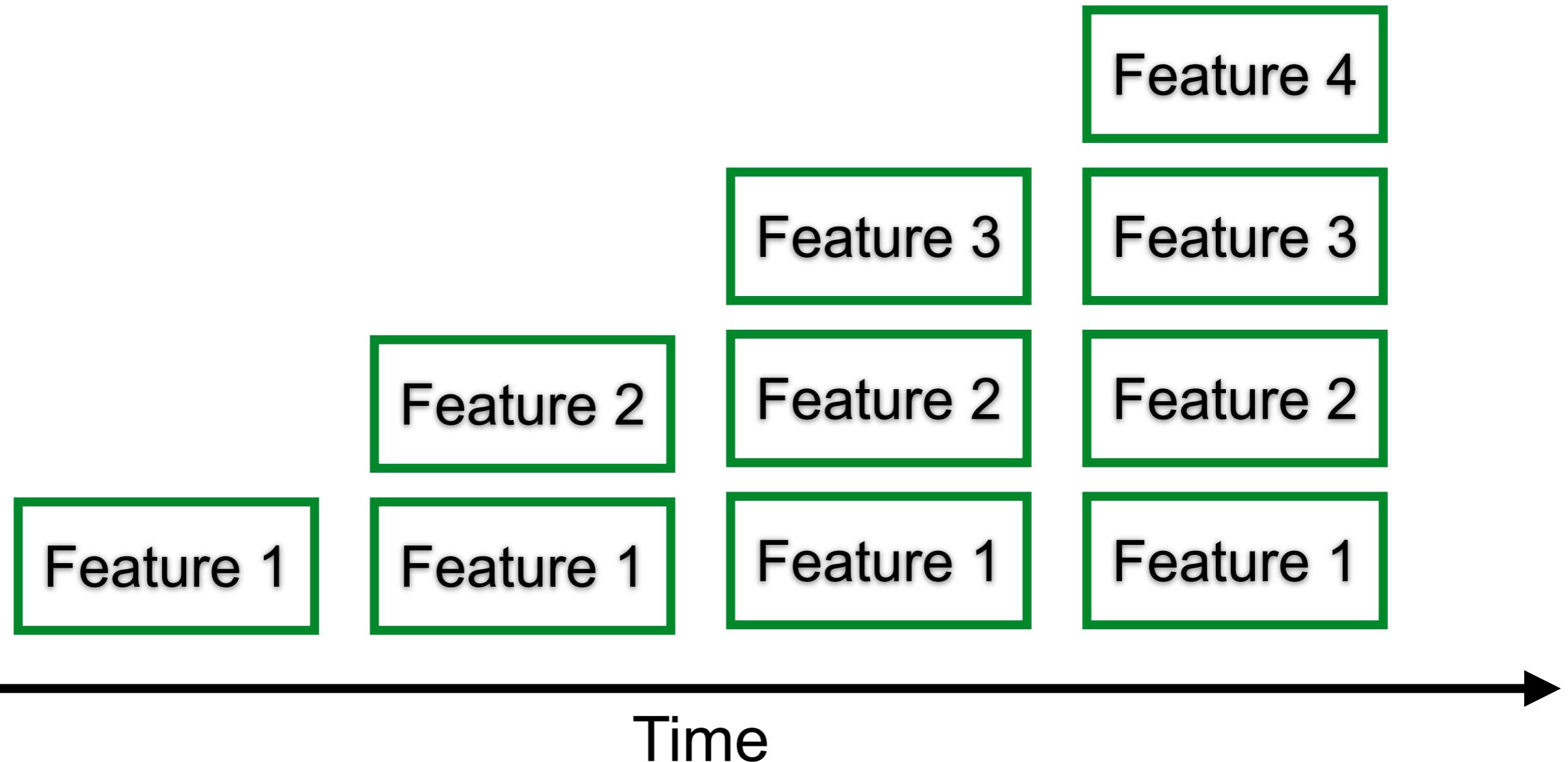
Iterative and incremental process

Done = coded and tested

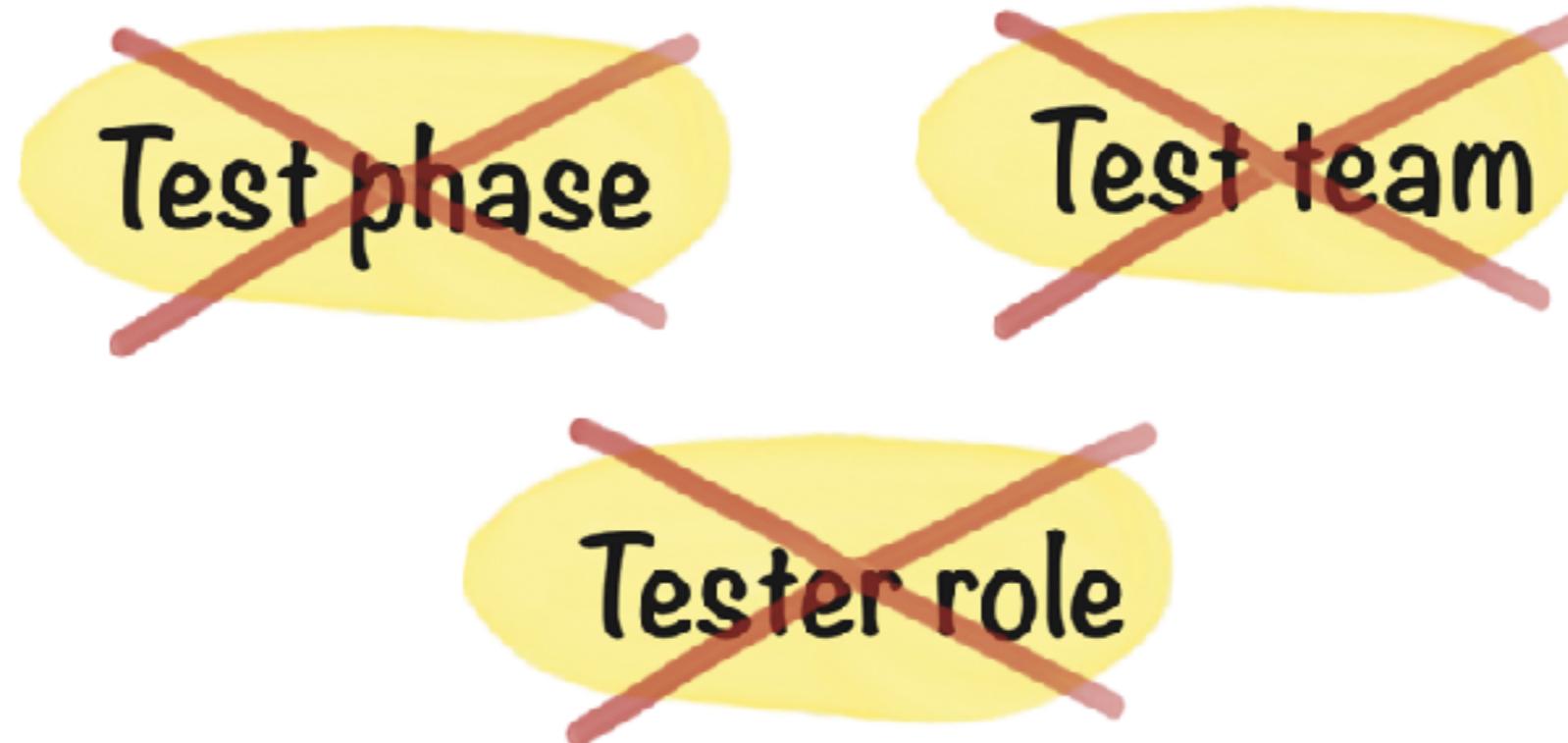


Iterative and incremental process

Done = coded and tested

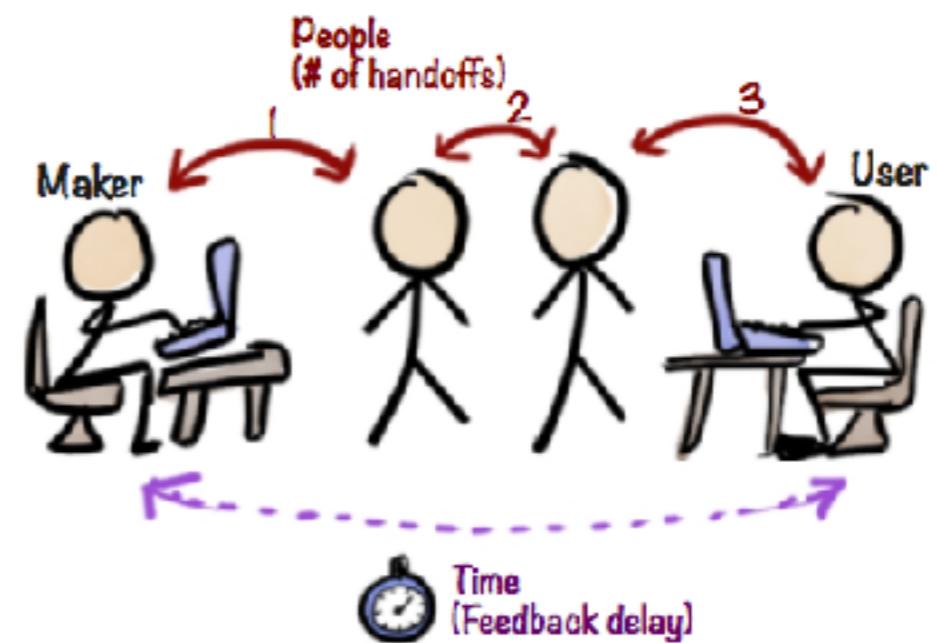
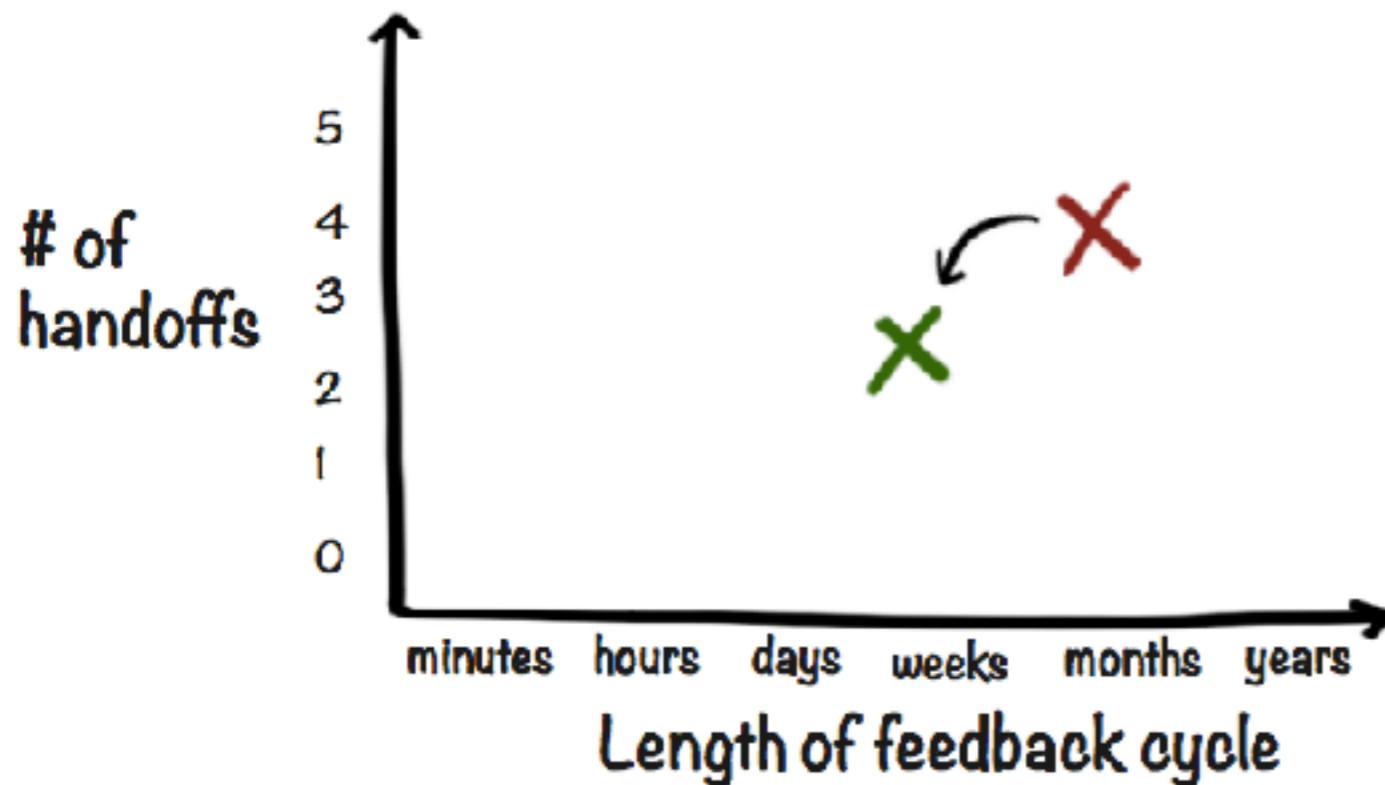


Testing is activity

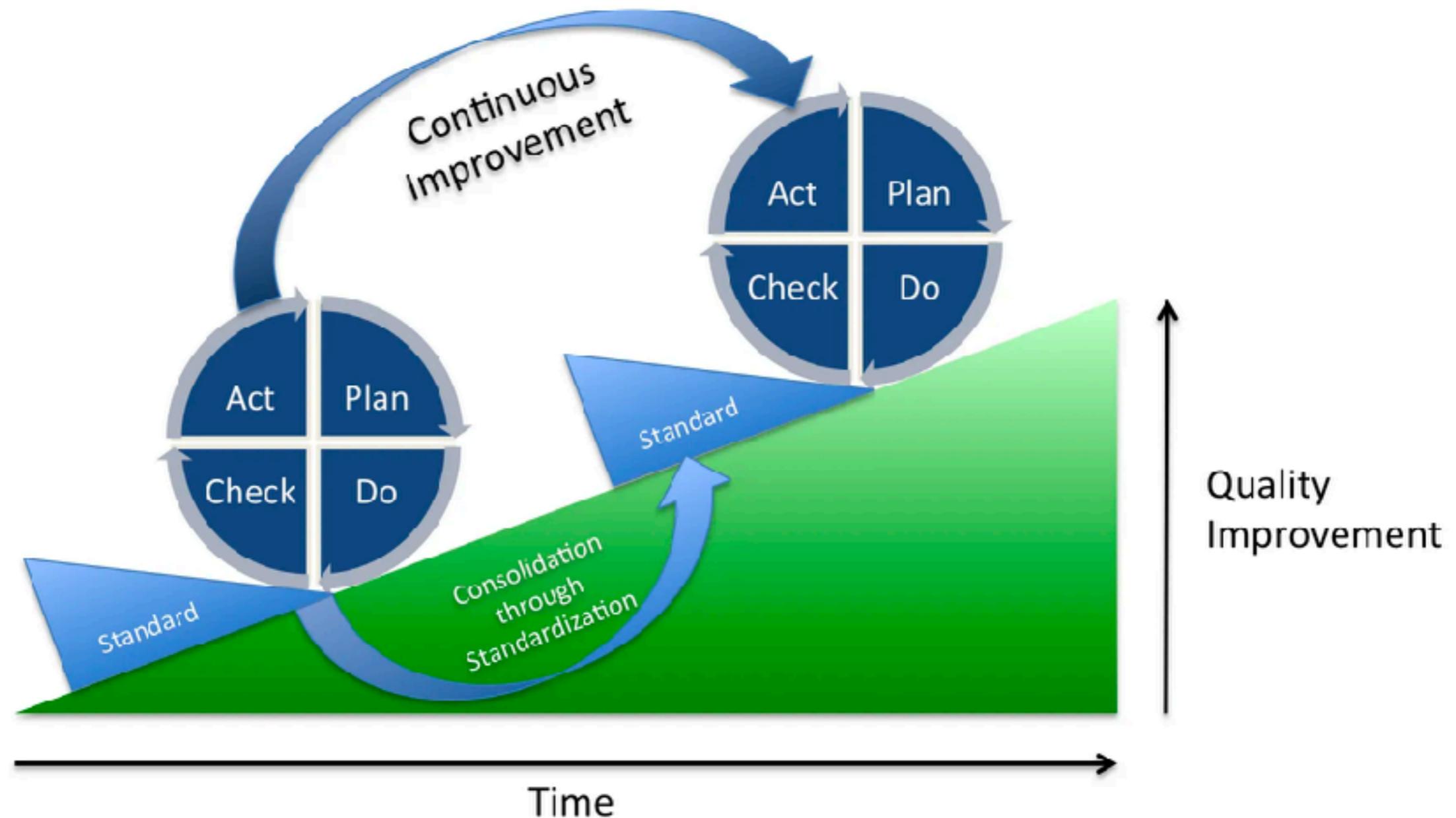


Fast feedback loop

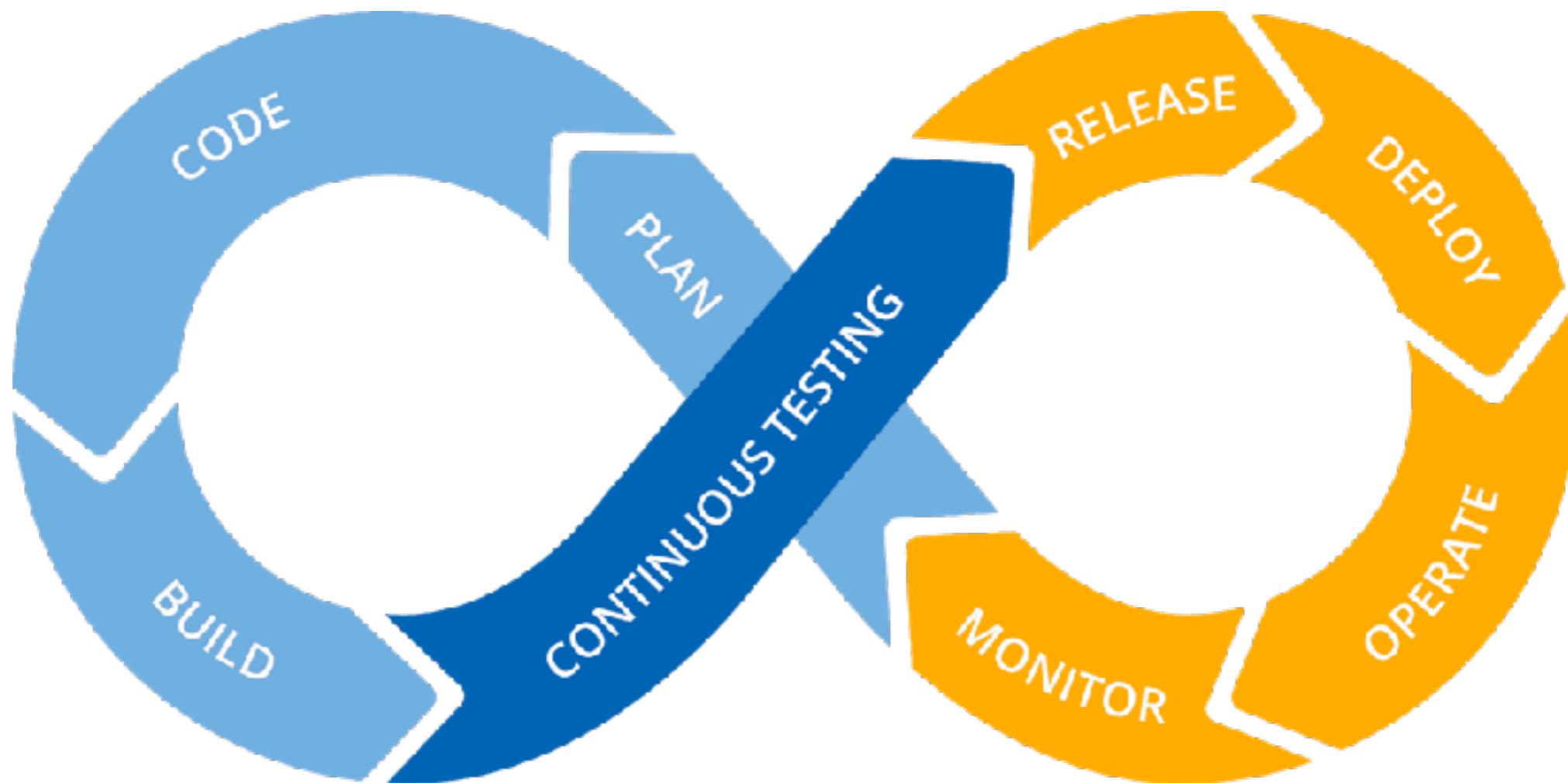
Shorten the feedback loop



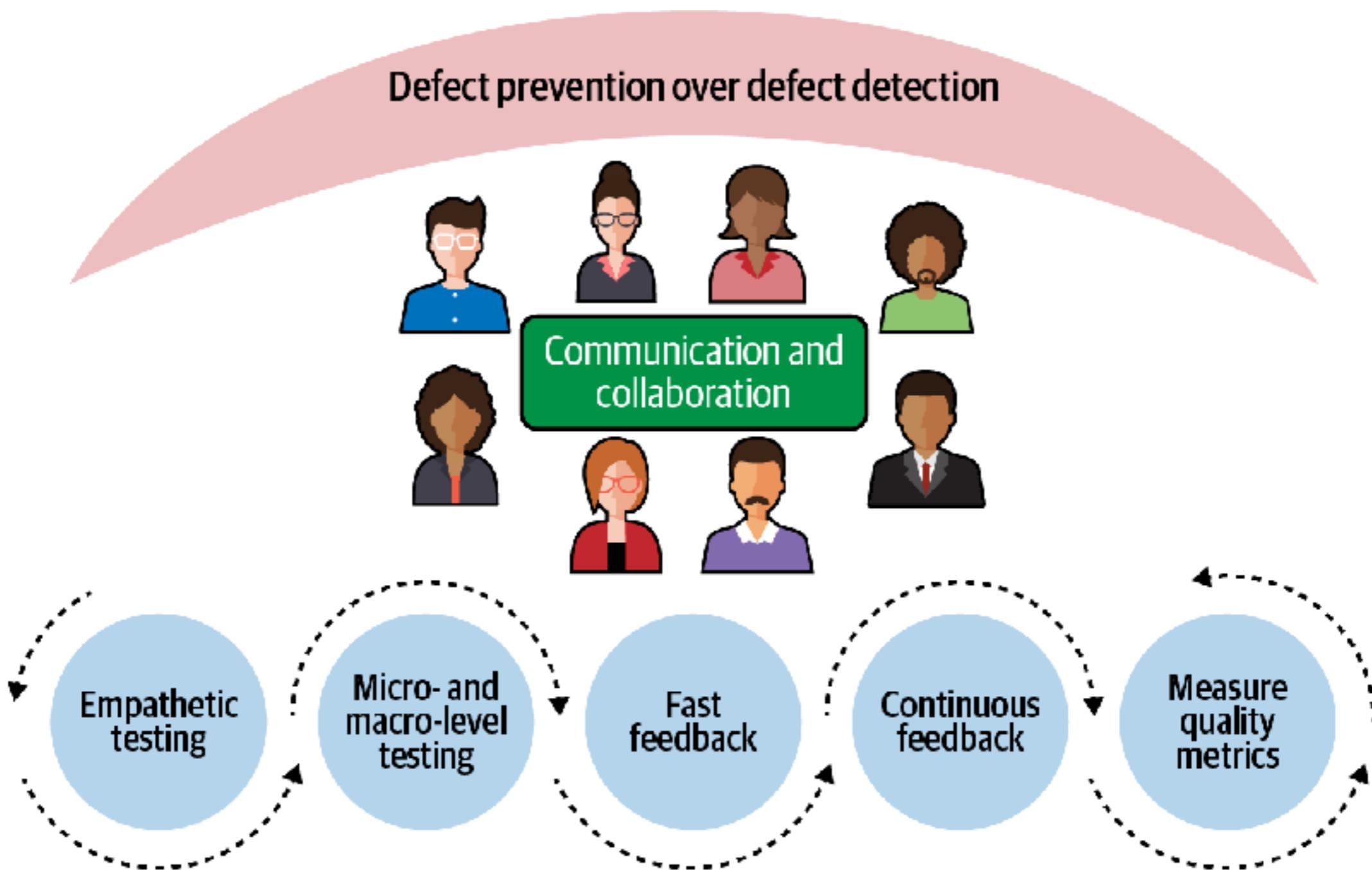
Continuous improvement



Continuous testing and feedback



Principles of Testing



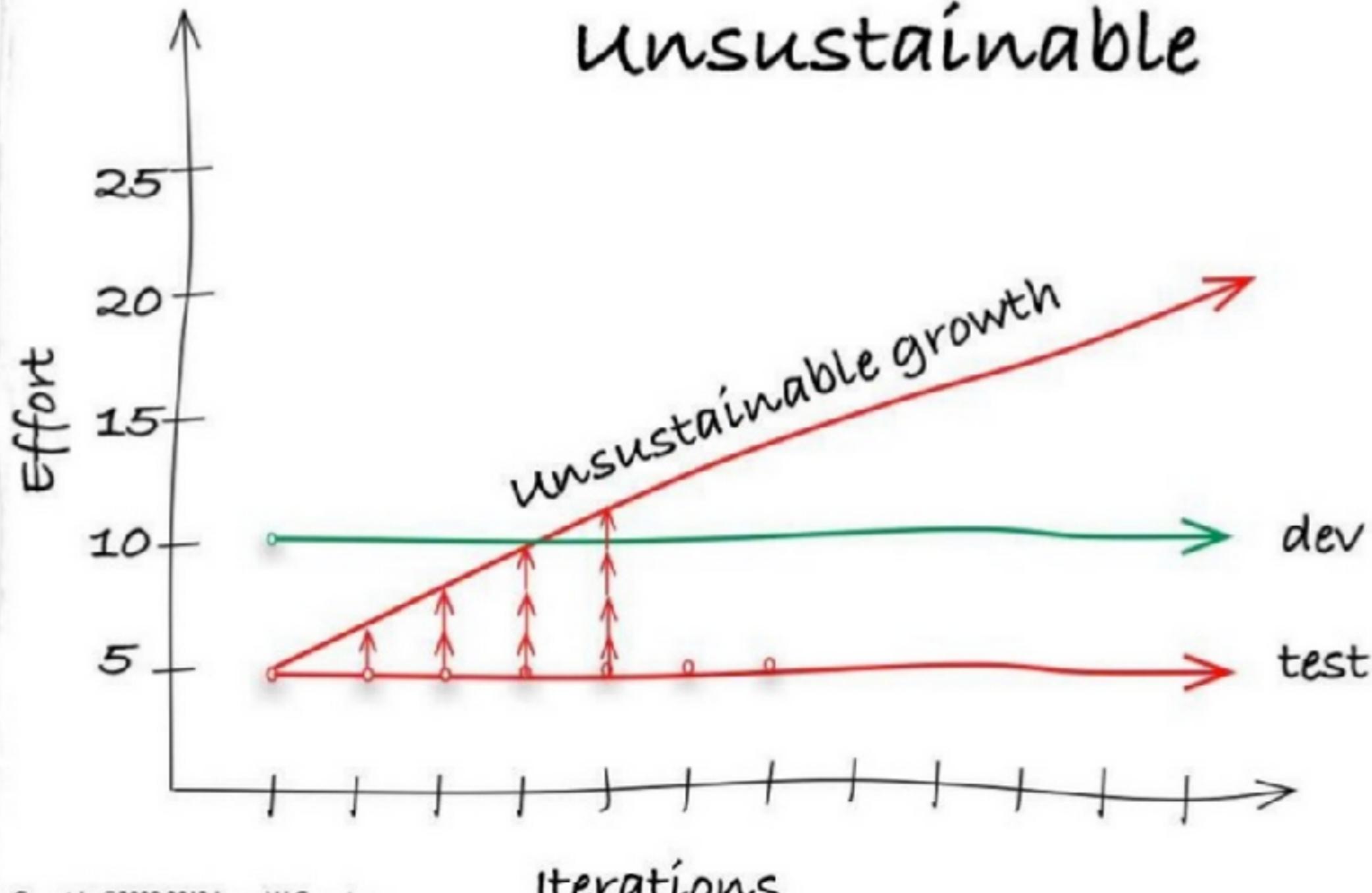
How to test in software delivery ?



Manual testing ?



Manual Test is unsustainable

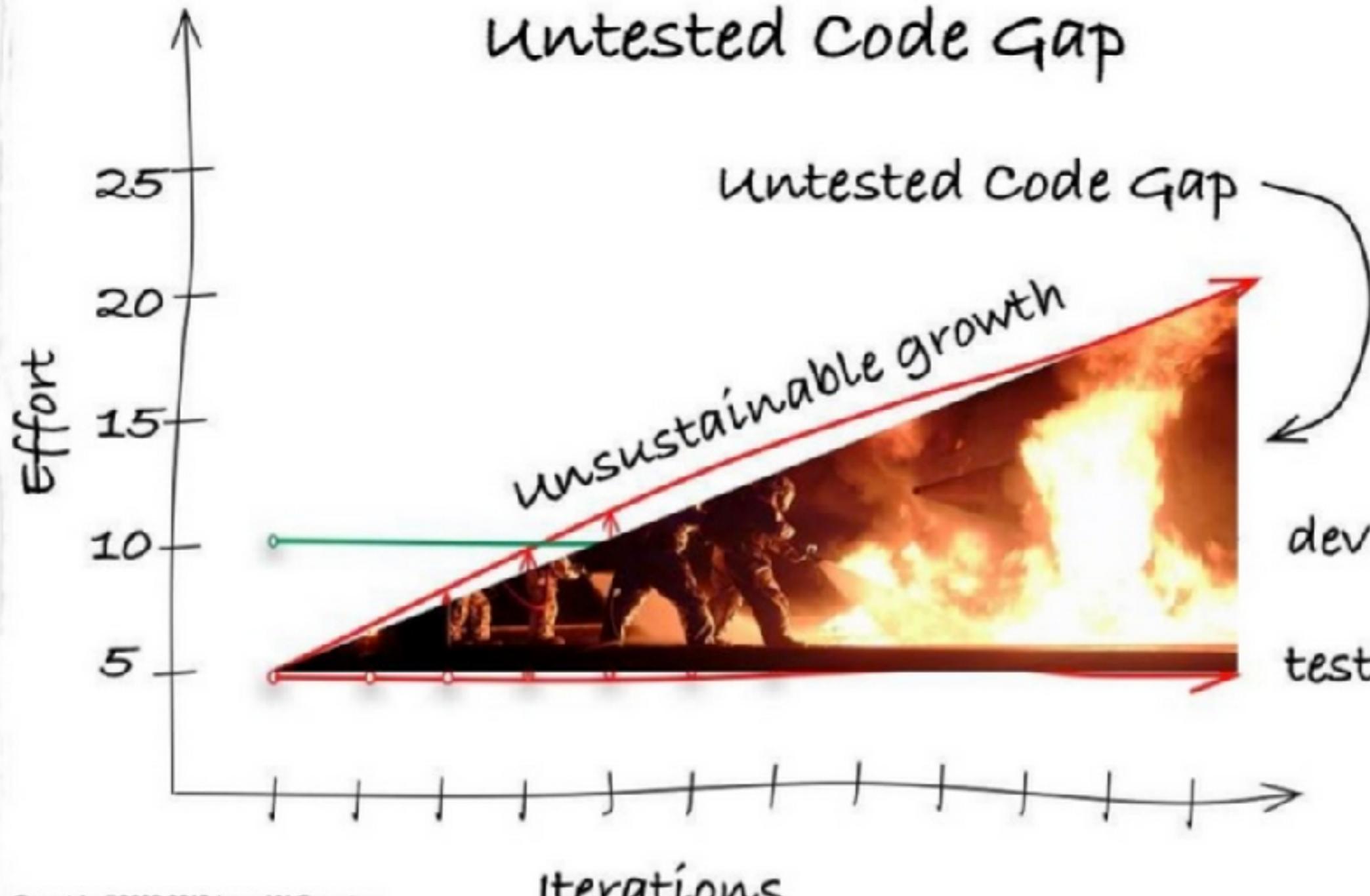


Copyright ©2008-2012 James W. Grenning
All Rights Reserved.

<https://wingman-sw.com/>



Risk Accumulates in the Untested Code Gap



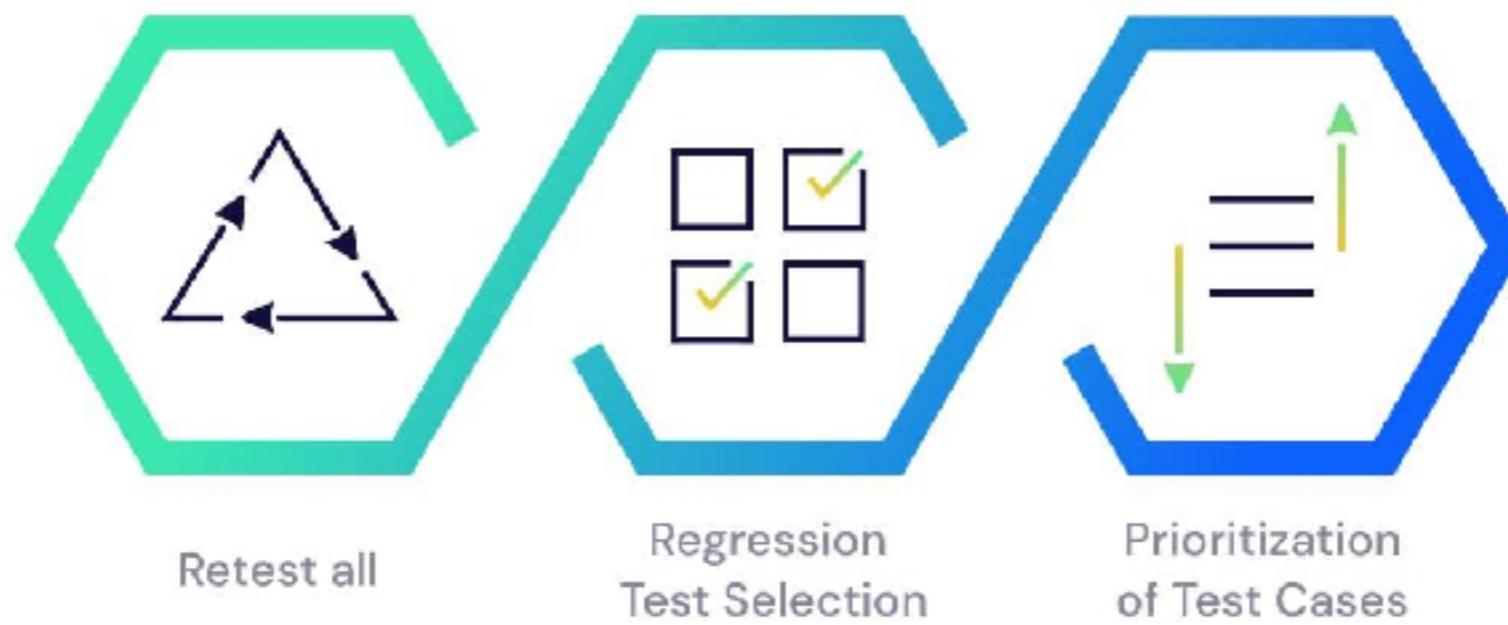
Copyright ©2008-2012 James W. Grenning
All Rights Reserved.

<https://wingman-sw.com/>



Regression Testing ?

Runs after every change to ensure
that the change introduces no unintended
breaks



CAUTION

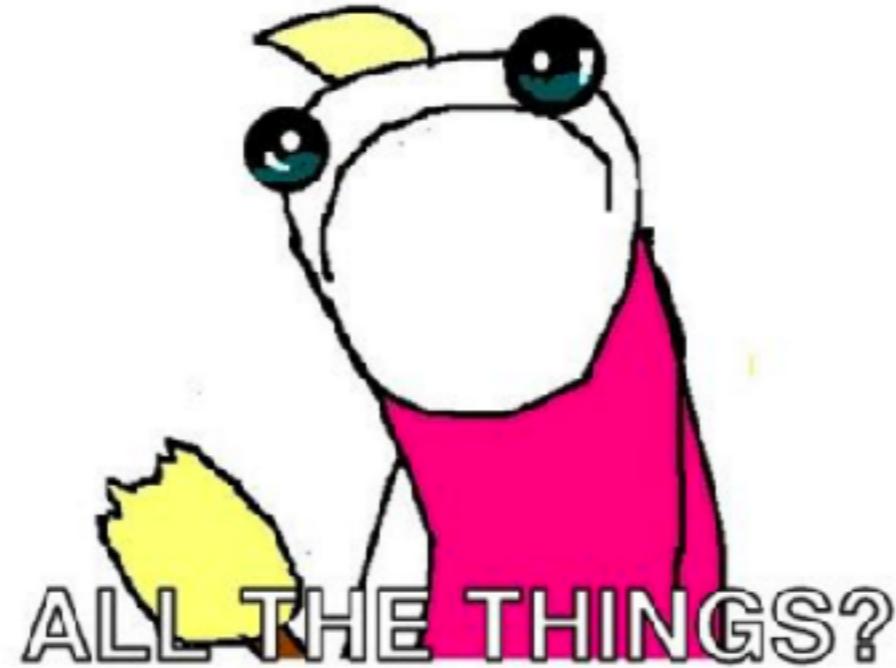


Human error



We need automation !!

AUTOMATE

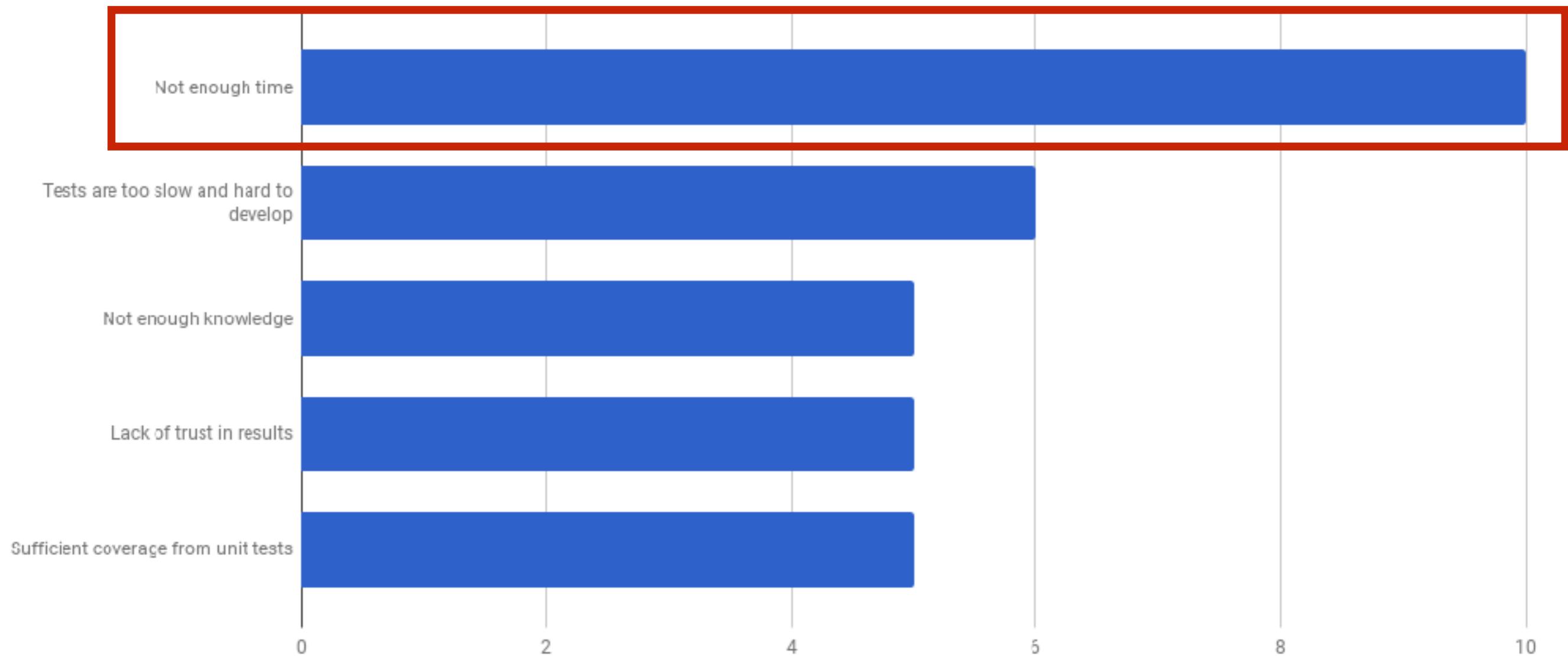


But,
It's take time to learning and
practice !!



Why not write automated tests ?

Not enough time !!!



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



Why should you automate ?

Manual checking **take too long**

Manual checks are **error prone**

Free people to do their best work

Provides living document

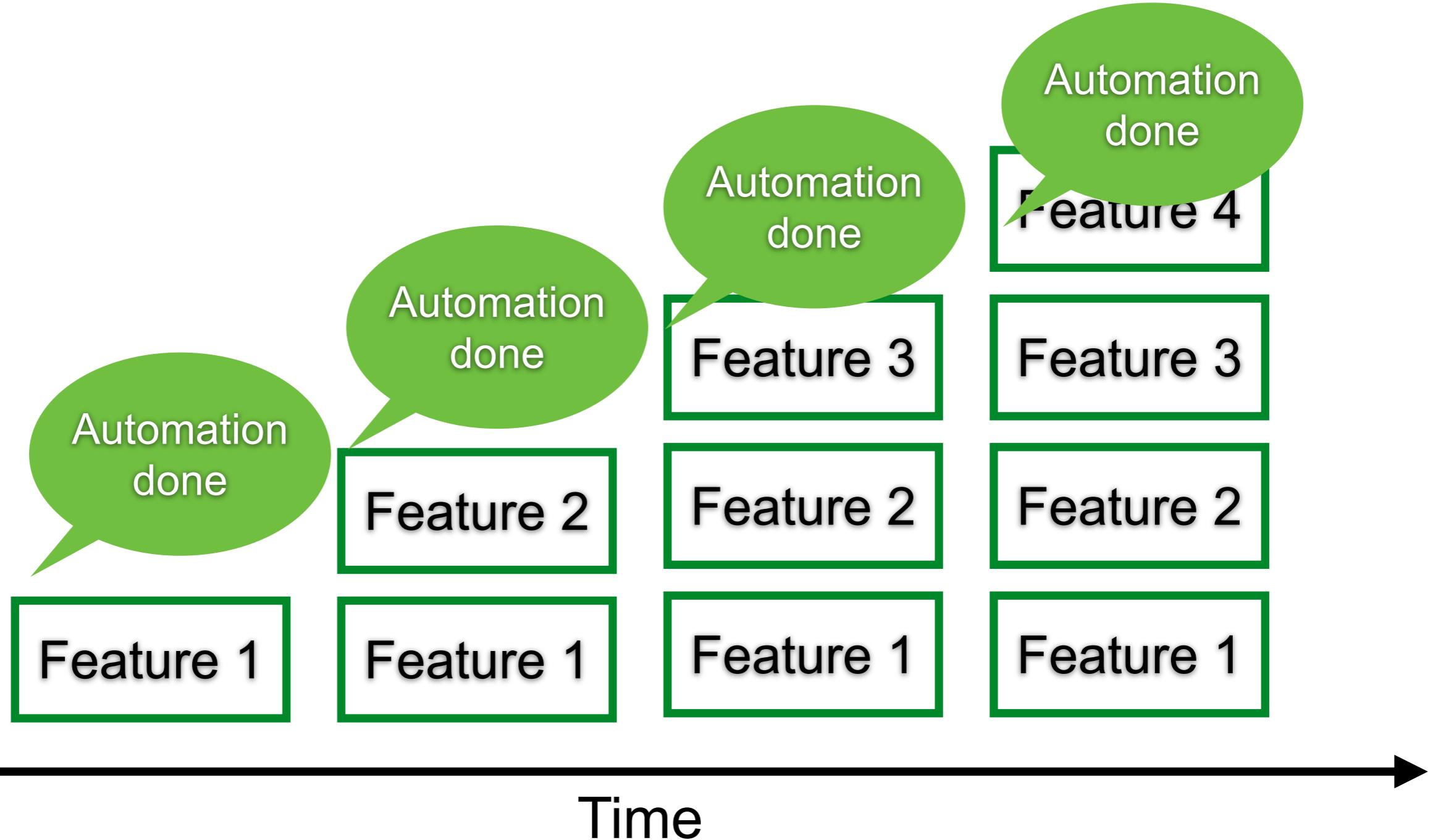
Repeatable

Save time



Iterative and incremental process

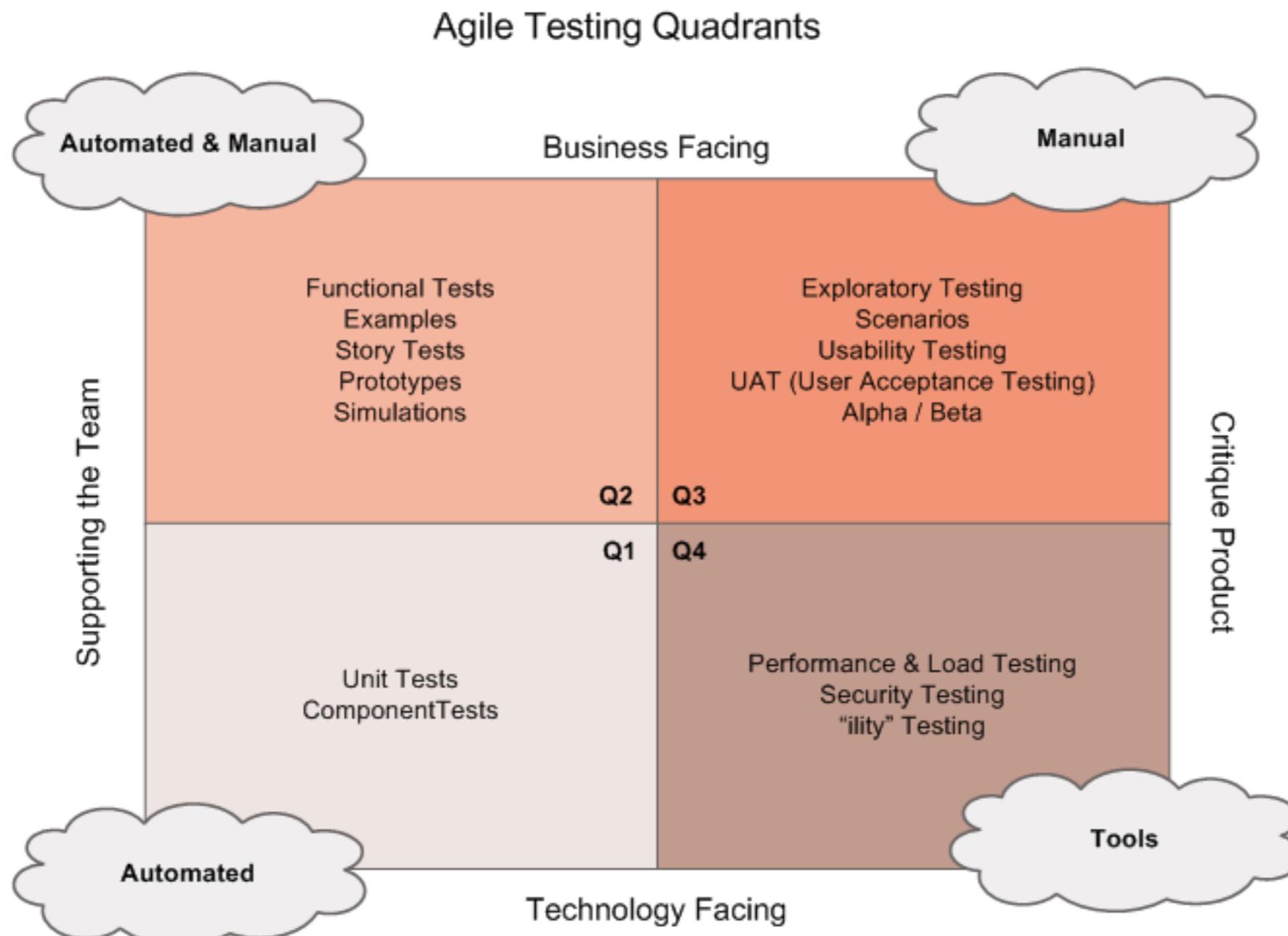
Done = coded and tested



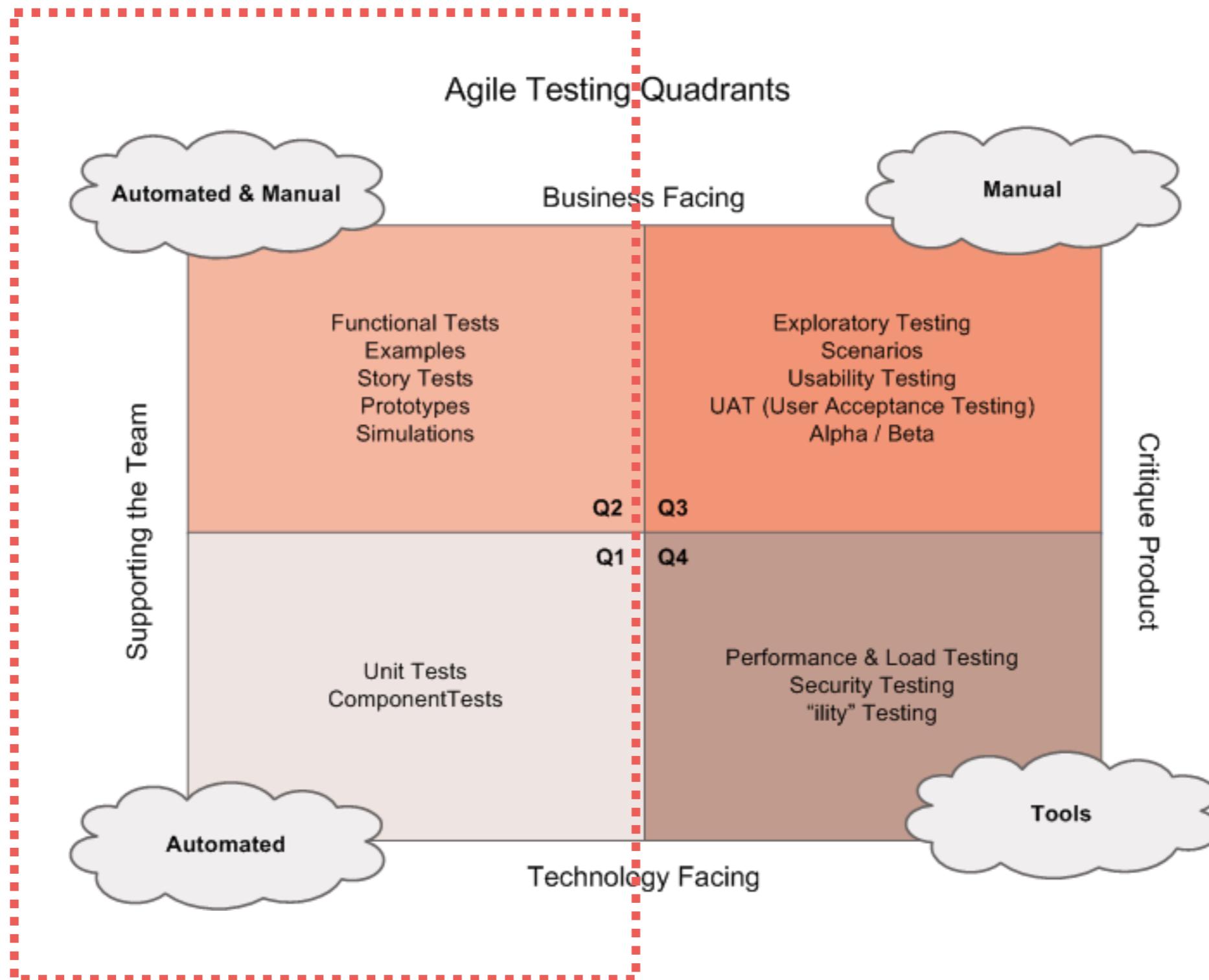
Testing Quadrant



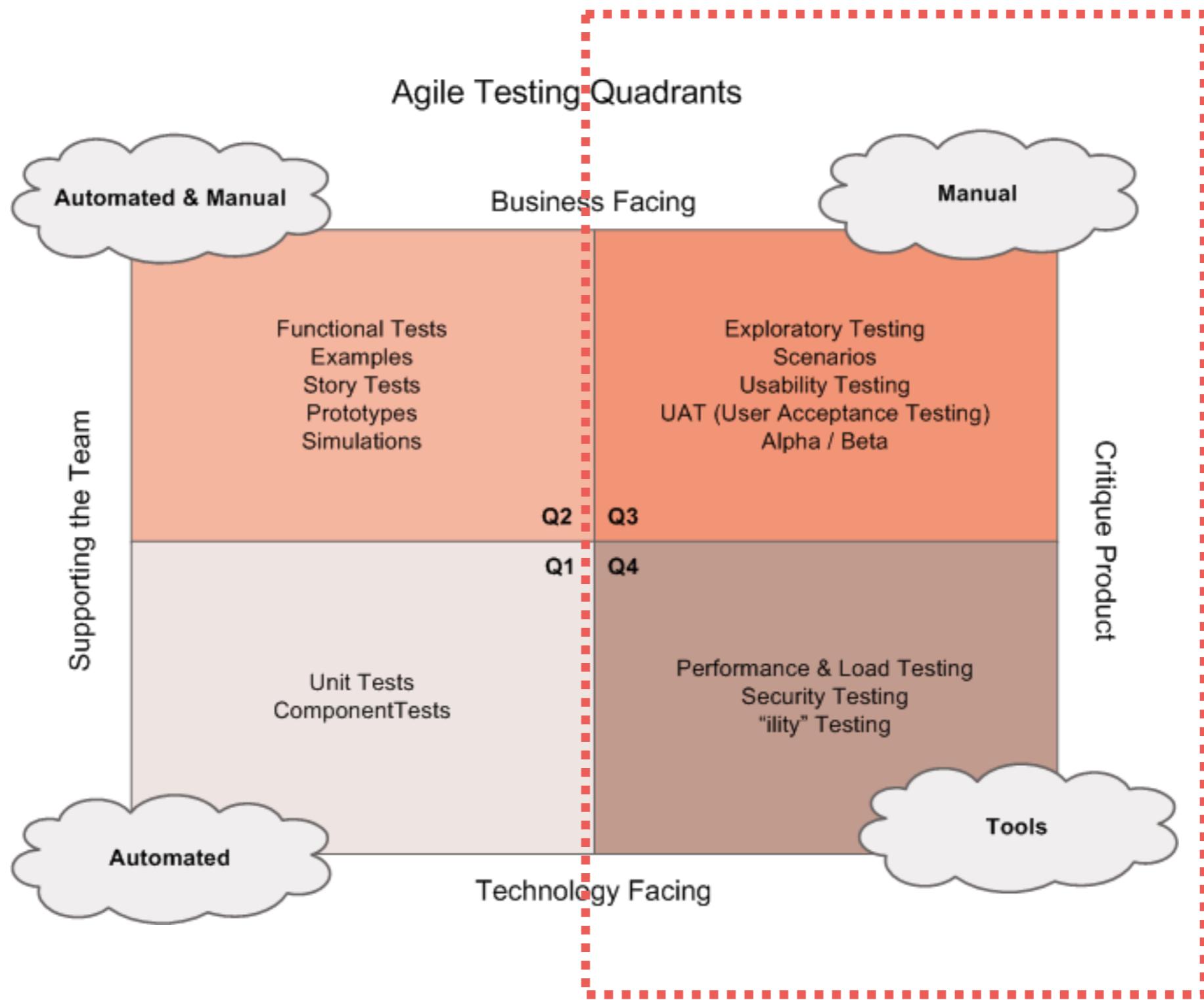
Testing Quadrant



Prevention



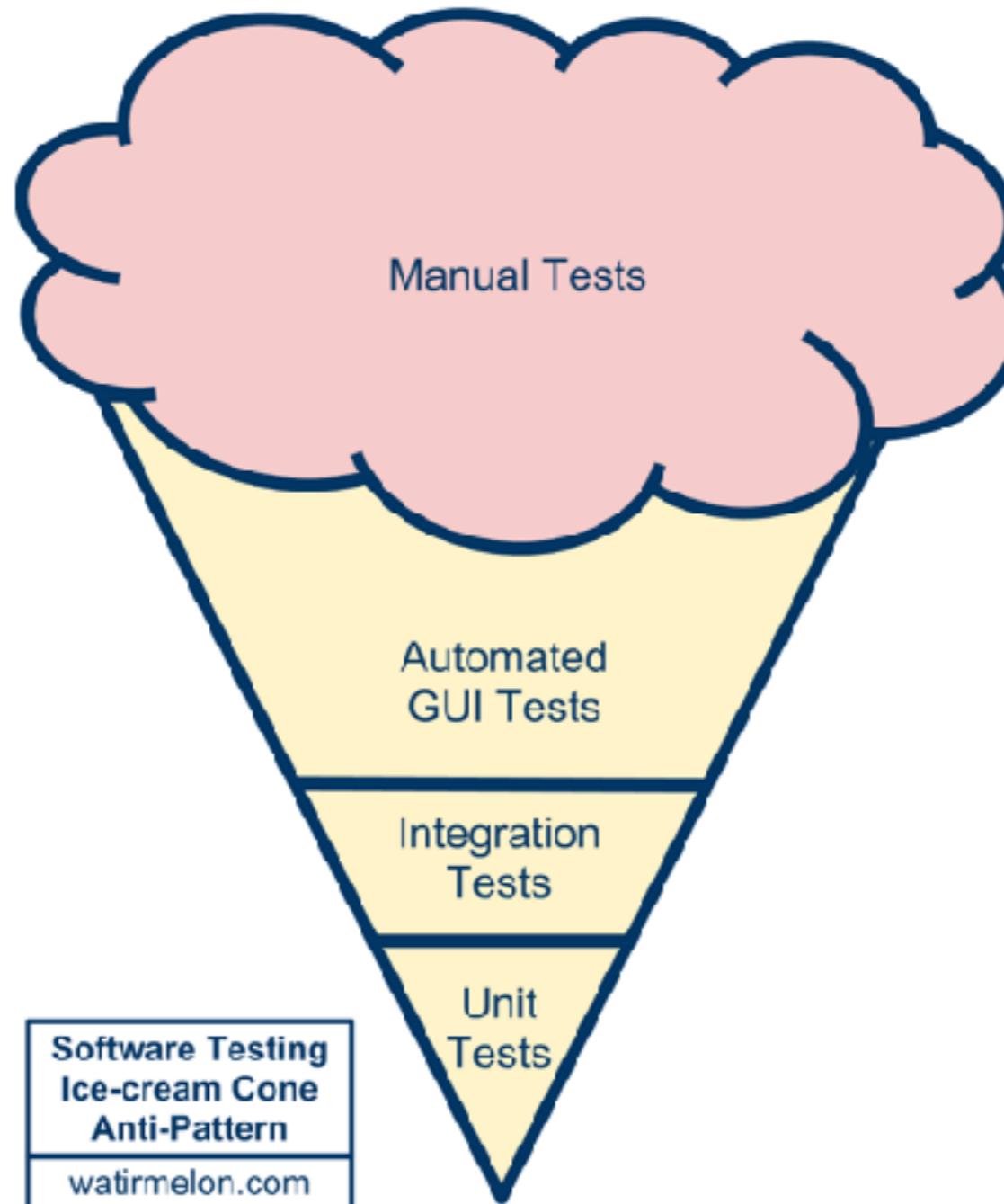
Detection



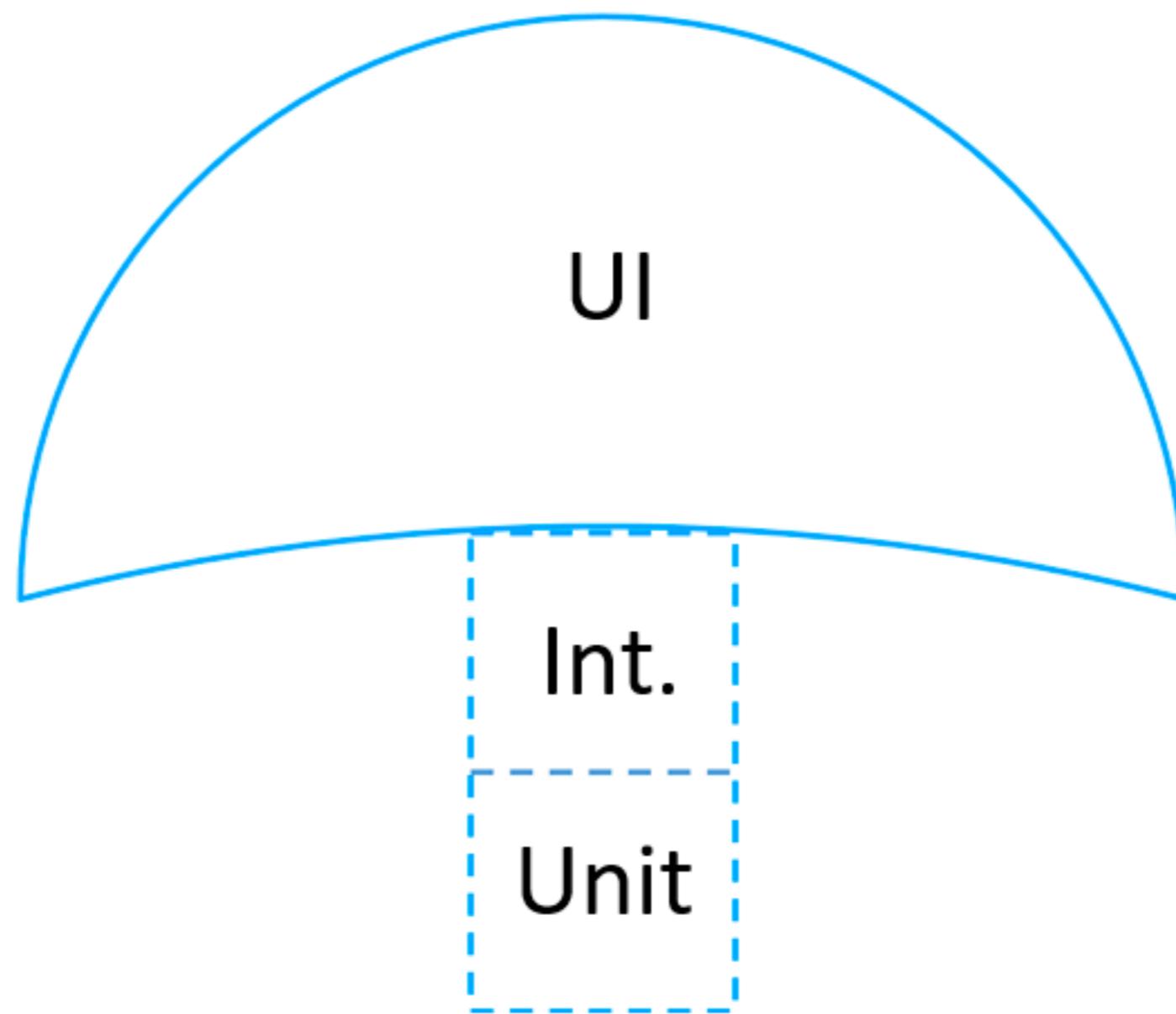
Testing pyramid



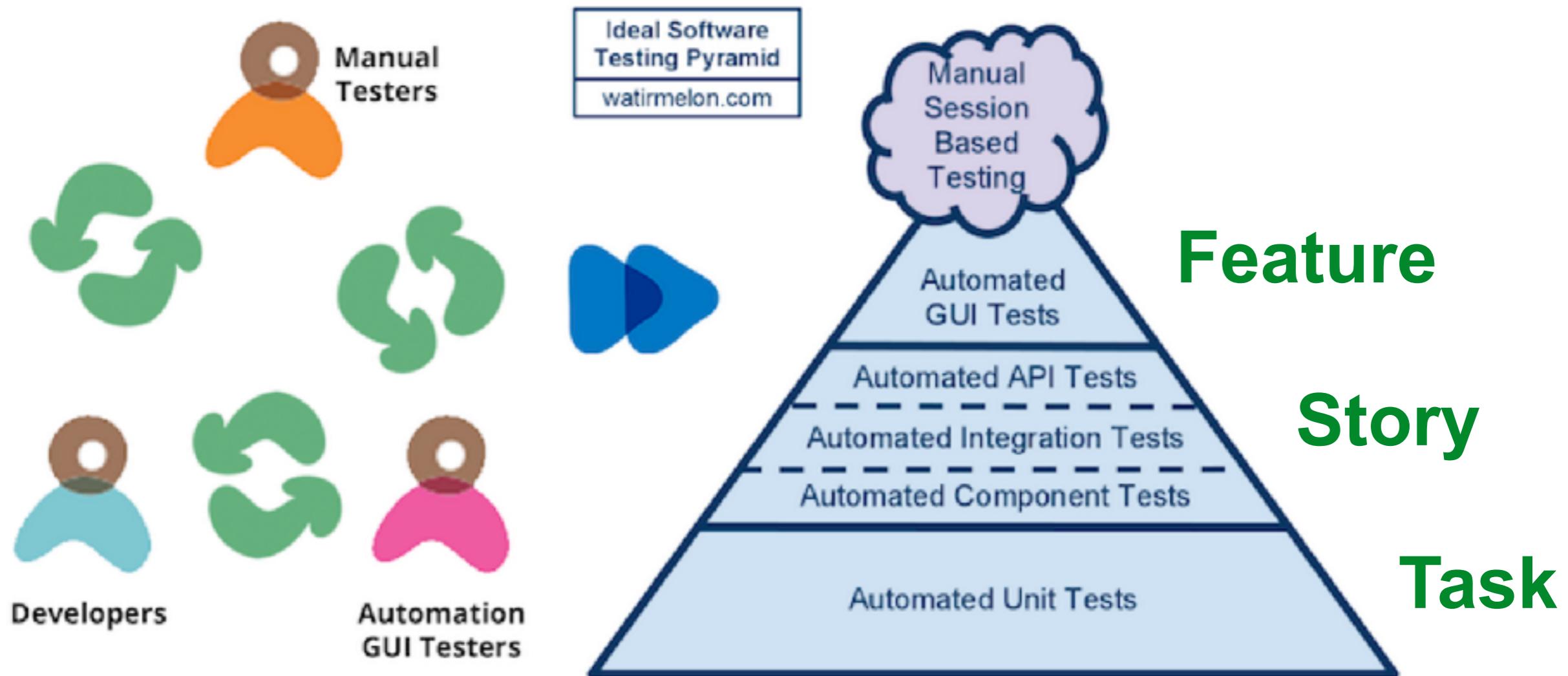
Ice-cream testing



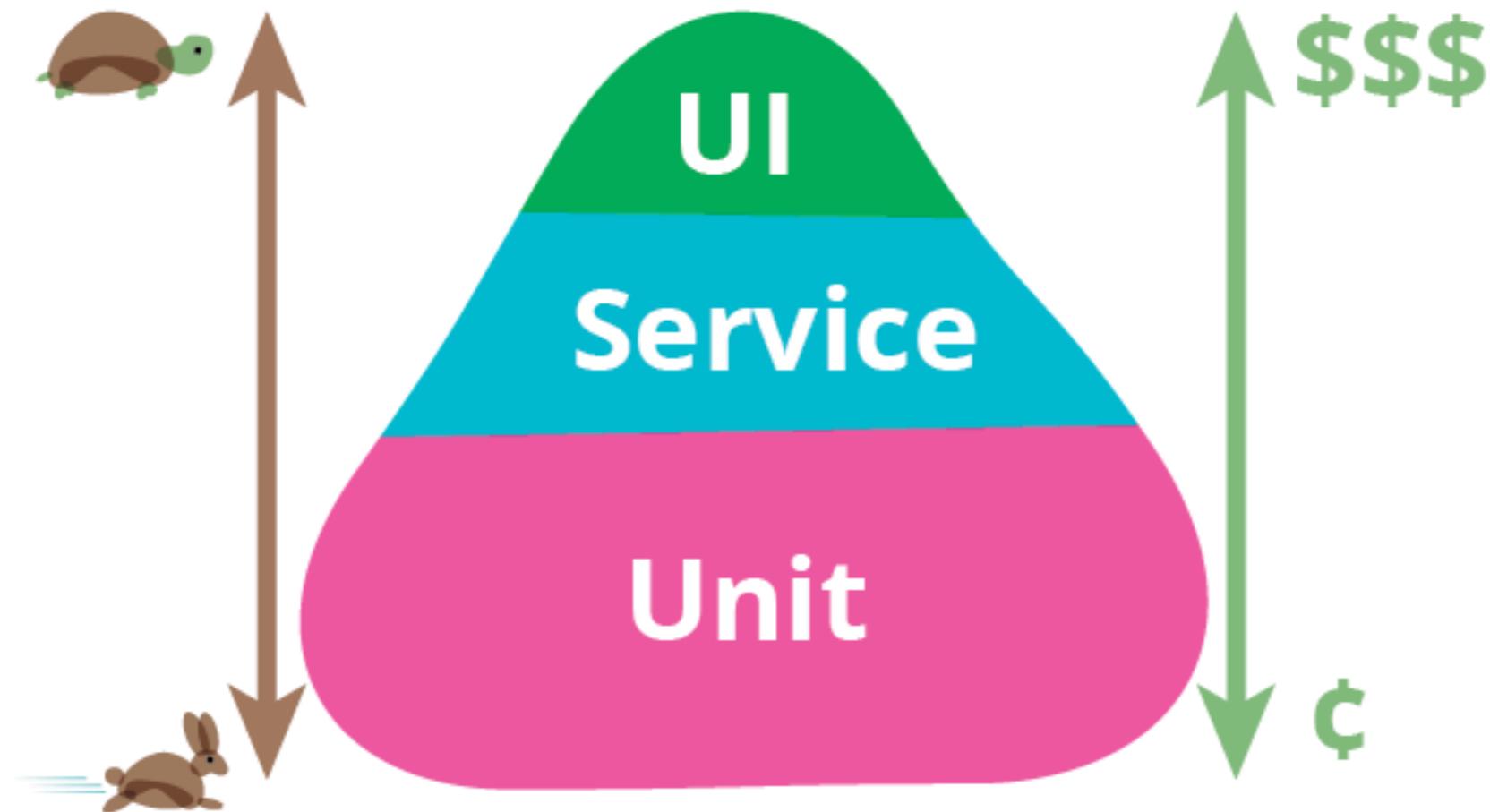
Mushroom testing



Testing Pyramid



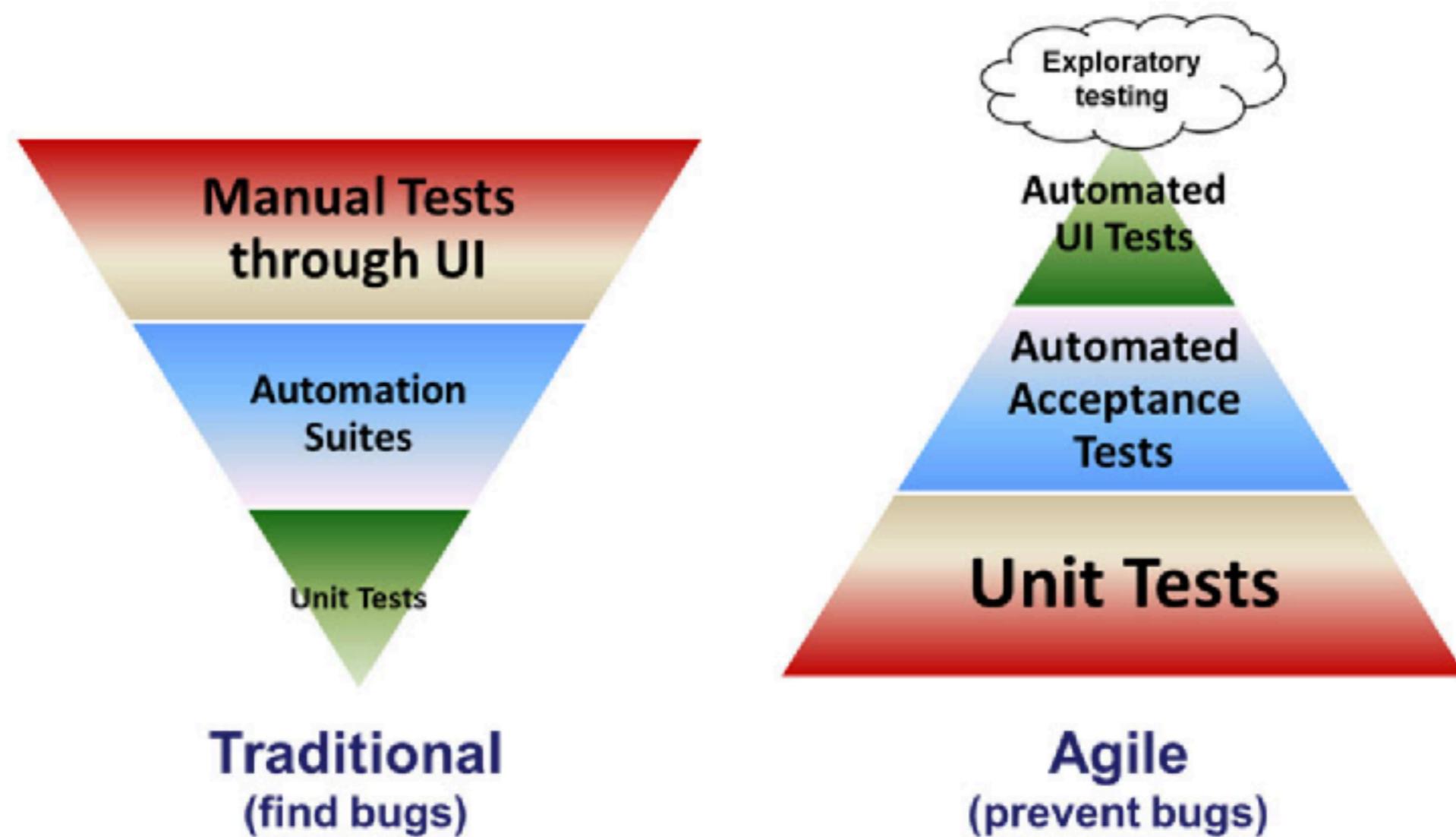
Testing Pyramid



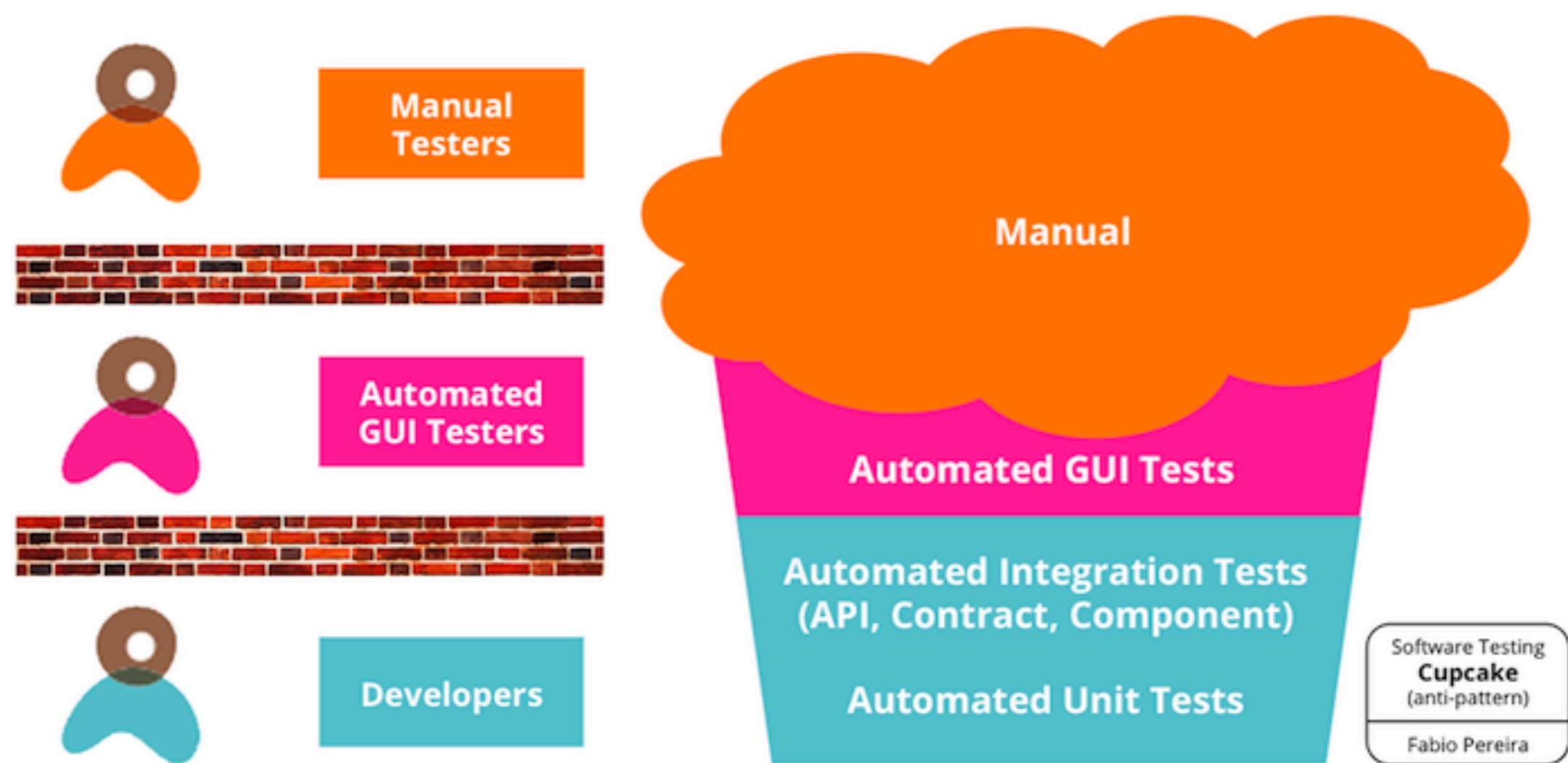
<https://martinfowler.com/bliki/TestPyramid.html>



Find vs Prevent



Cupcake testing !!



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



Trophy testing

THE FOUR TYPES OF TESTS

End to End

A helper robot that behaves like a user to click around the app and verify that it functions correctly.

Sometimes called "functional testing" or e2e.

Integration

Verify that several units work together in harmony.

Unit

Verify that individual, isolated parts work as expected.

Static

Catch typos and type errors as you write the code.



<https://kentcdodds.com/blog/the-testing-trophy-and-testing-classifications>



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 !!



Understand What and Why to test ?

Discussion is very important



Remember !!

Test pyramid is a tool
To talk about automation tasks
How to prioritize 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
- Automated every paths



Thinking about automation

Incremental testing
Easy to test
Testable
Fast customer feedback

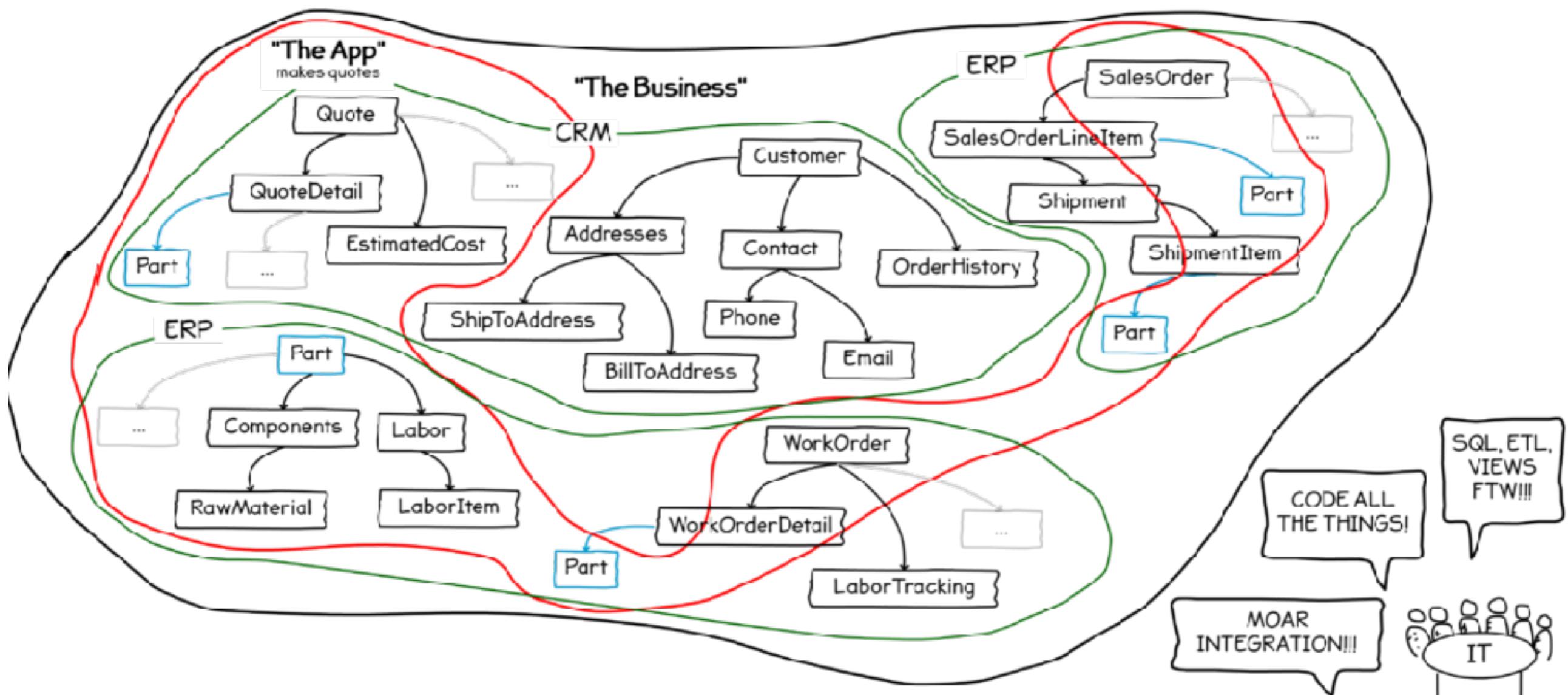


Know your System Architecture

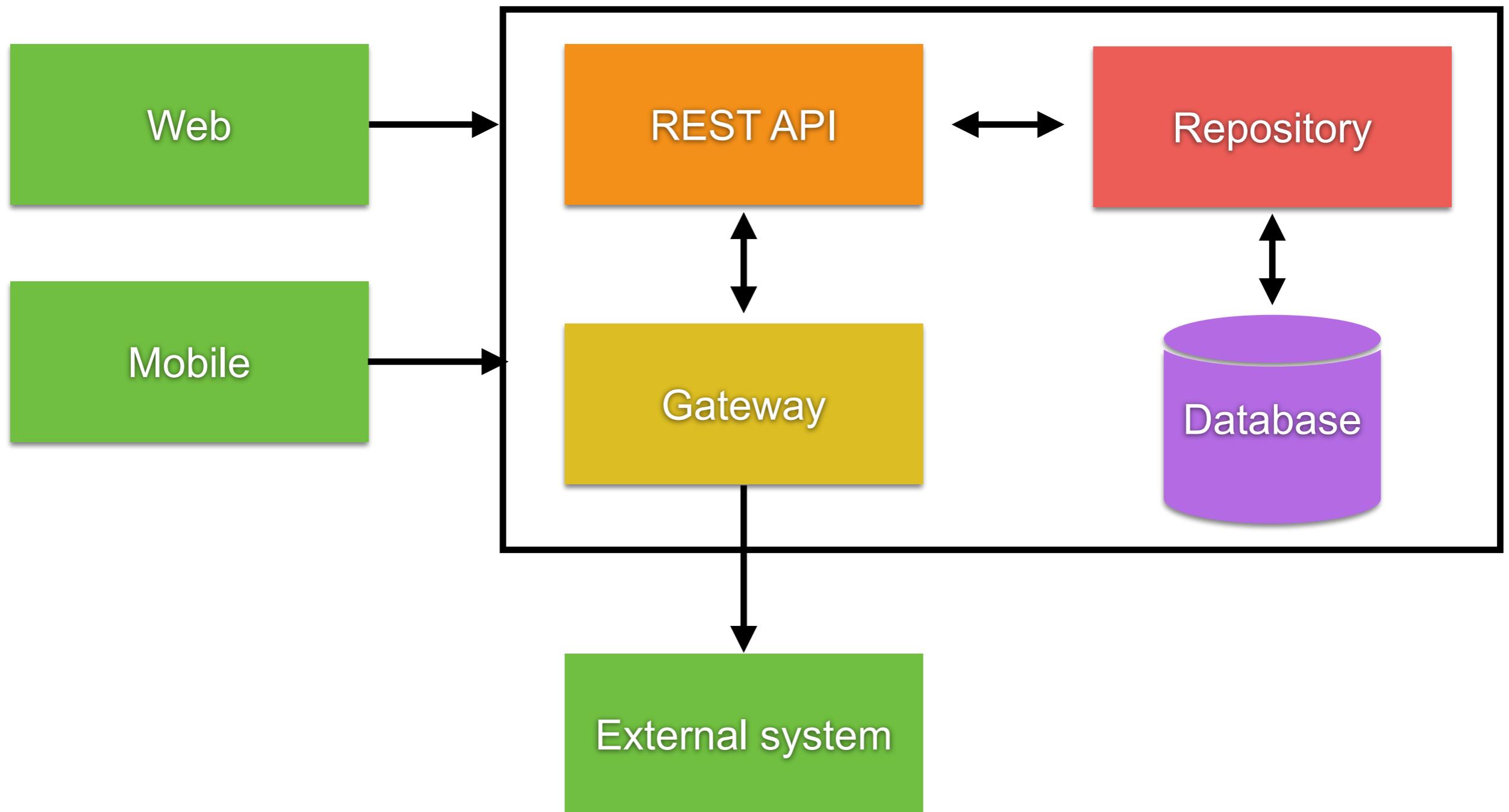


System impacts !!

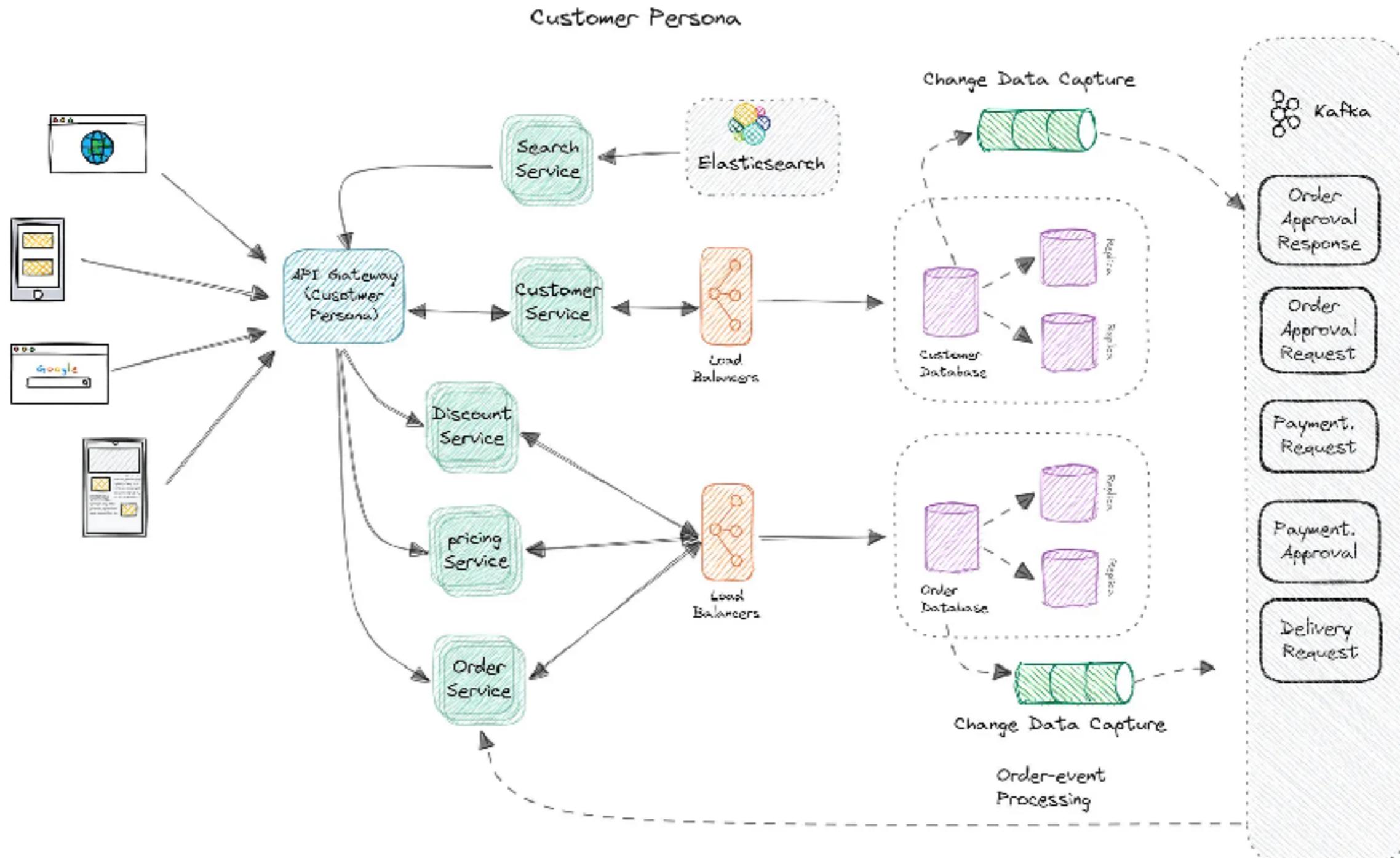
Know your systems



Simple Architecture



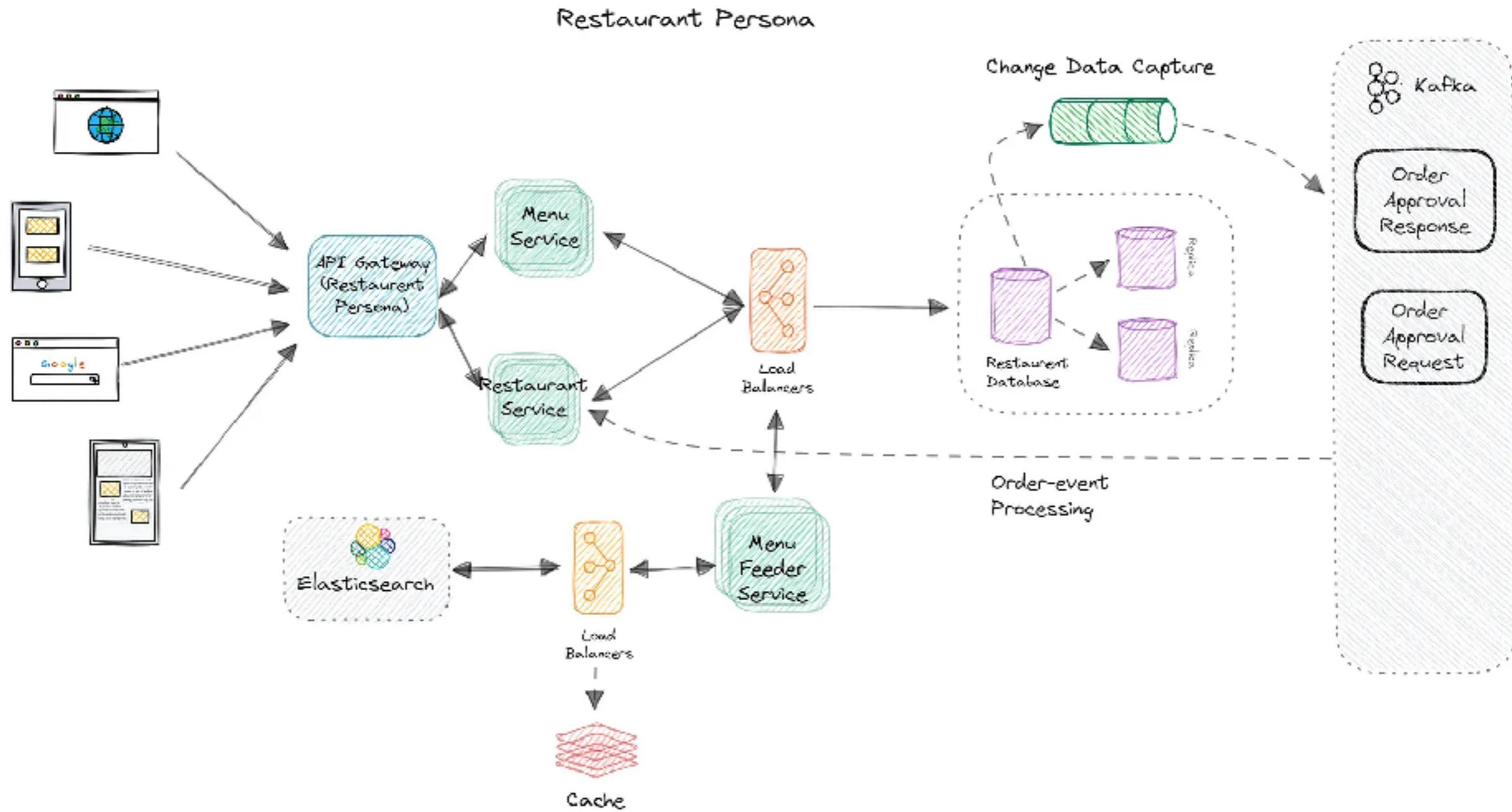
Food Delivery (Customer)



<https://medium.com/@systemdesignbychk/system-design-a-deep-dive-into-the-food-ordering-system-f84ae6375ce3>



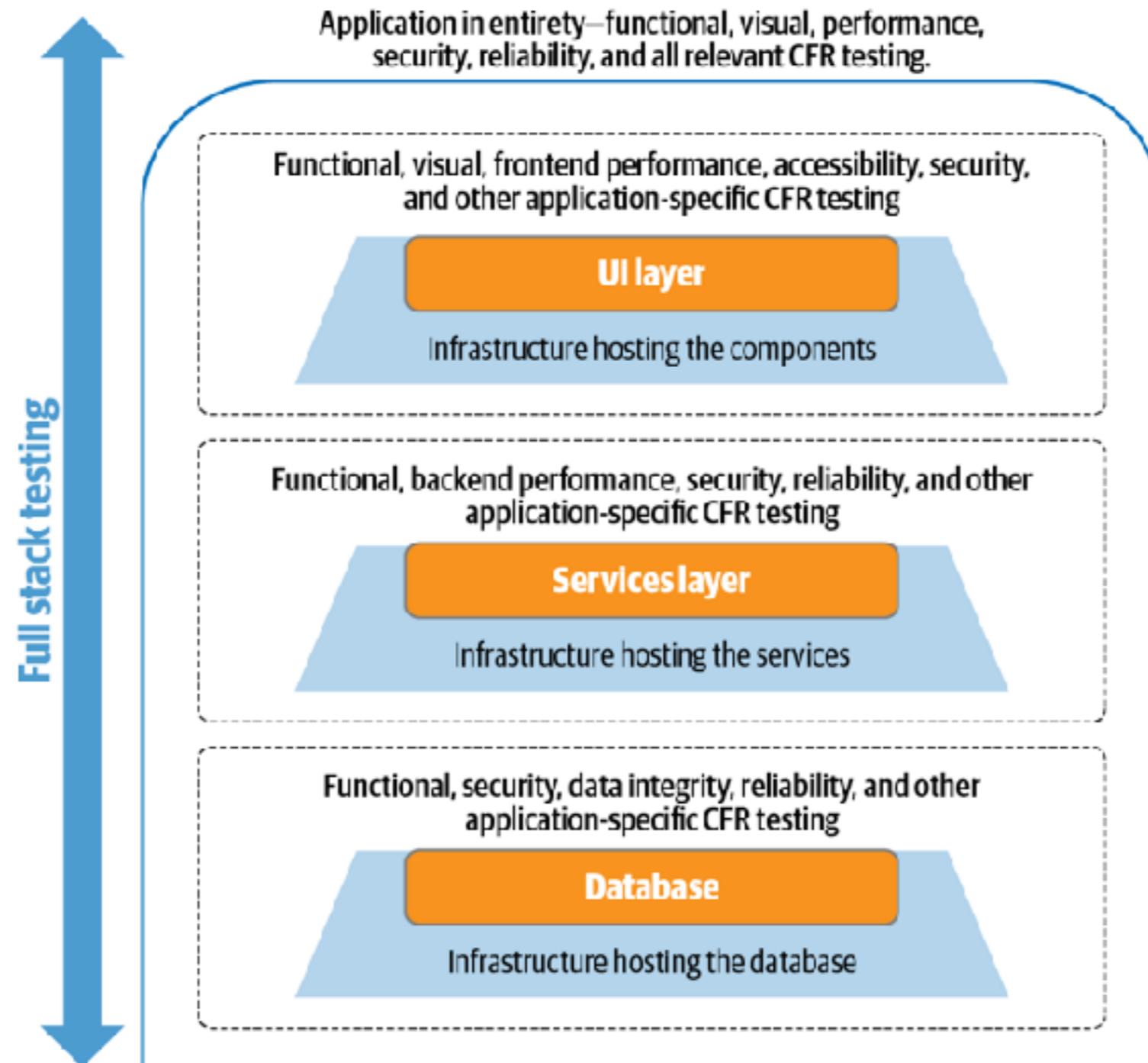
Food Delivery (Restaurant)



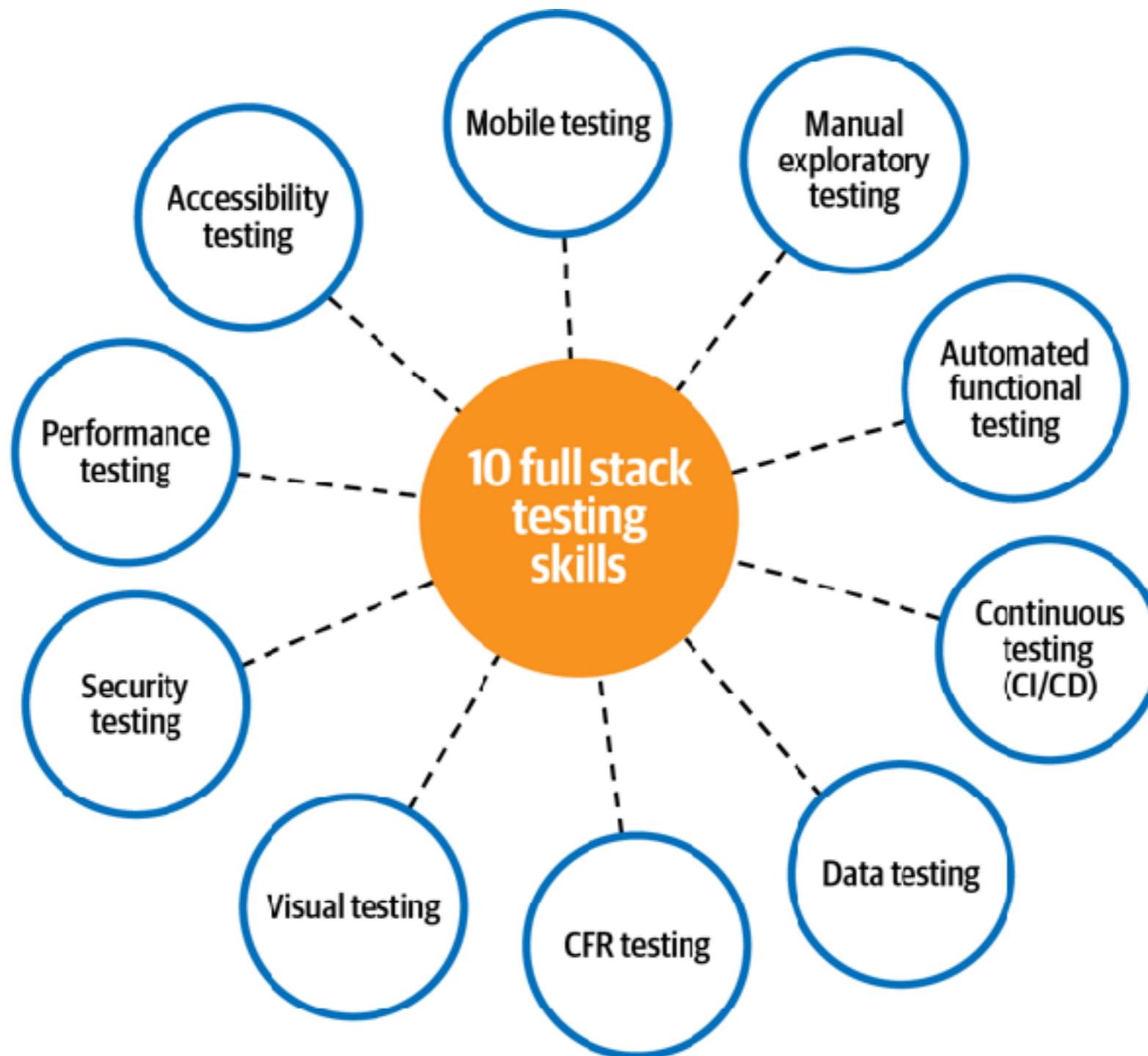
<https://medium.com/@systemdesignbychk/system-design-a-deep-dive-into-the-food-ordering-system-f84ae6375ce3>



Full Stack Testing

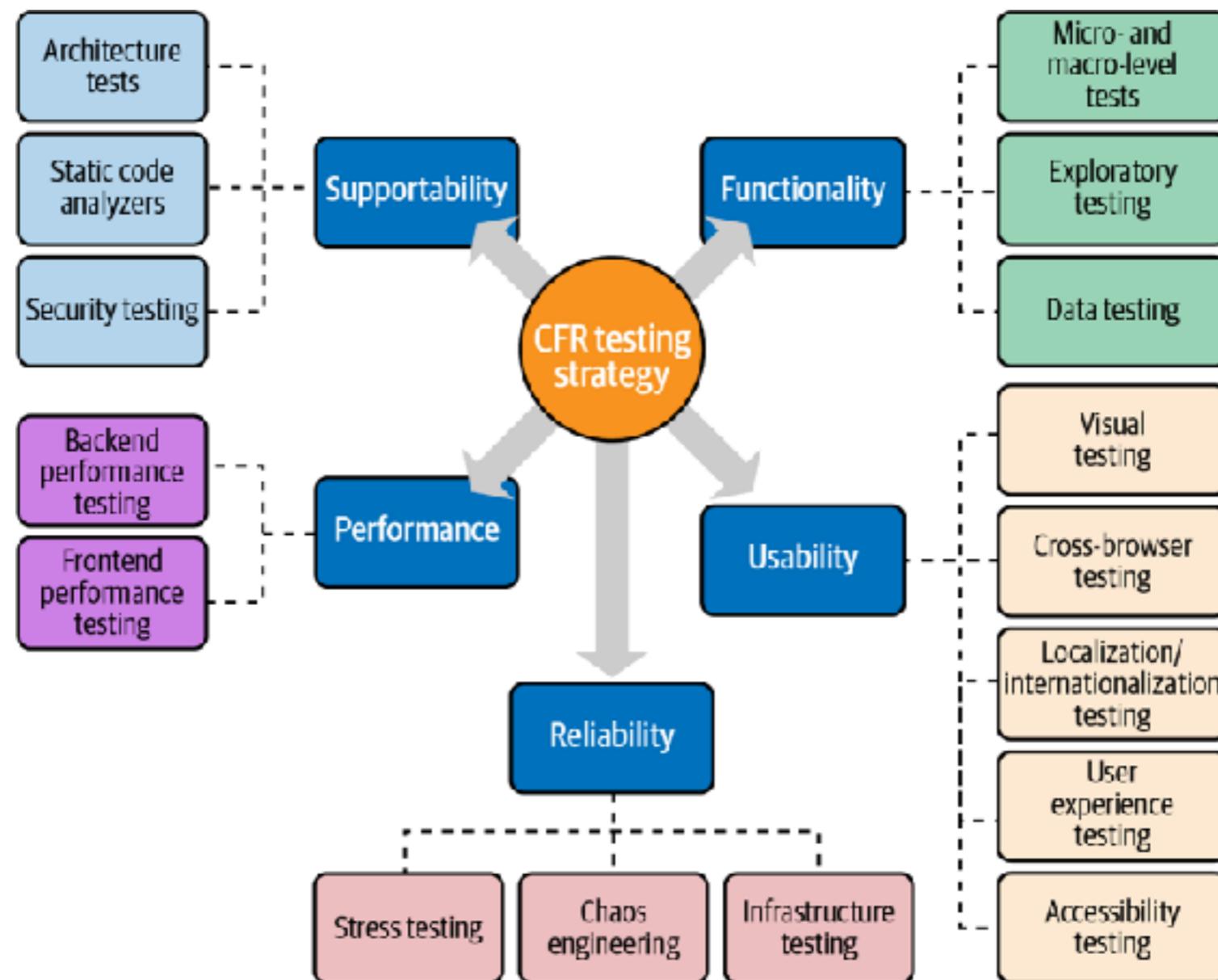


Full Stack Testing Skills



CFR testing

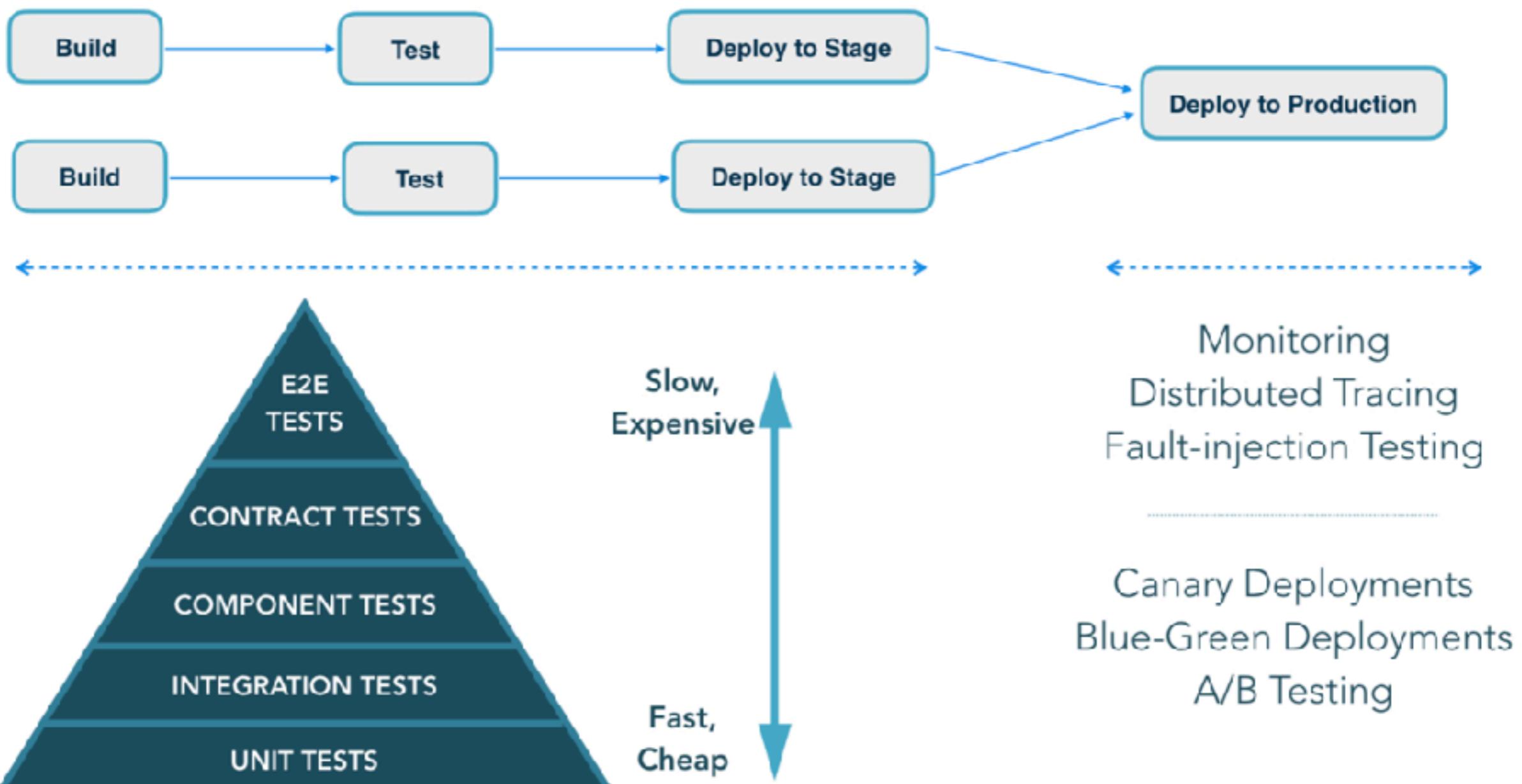
Cross Functional Requirement



Test Strategies



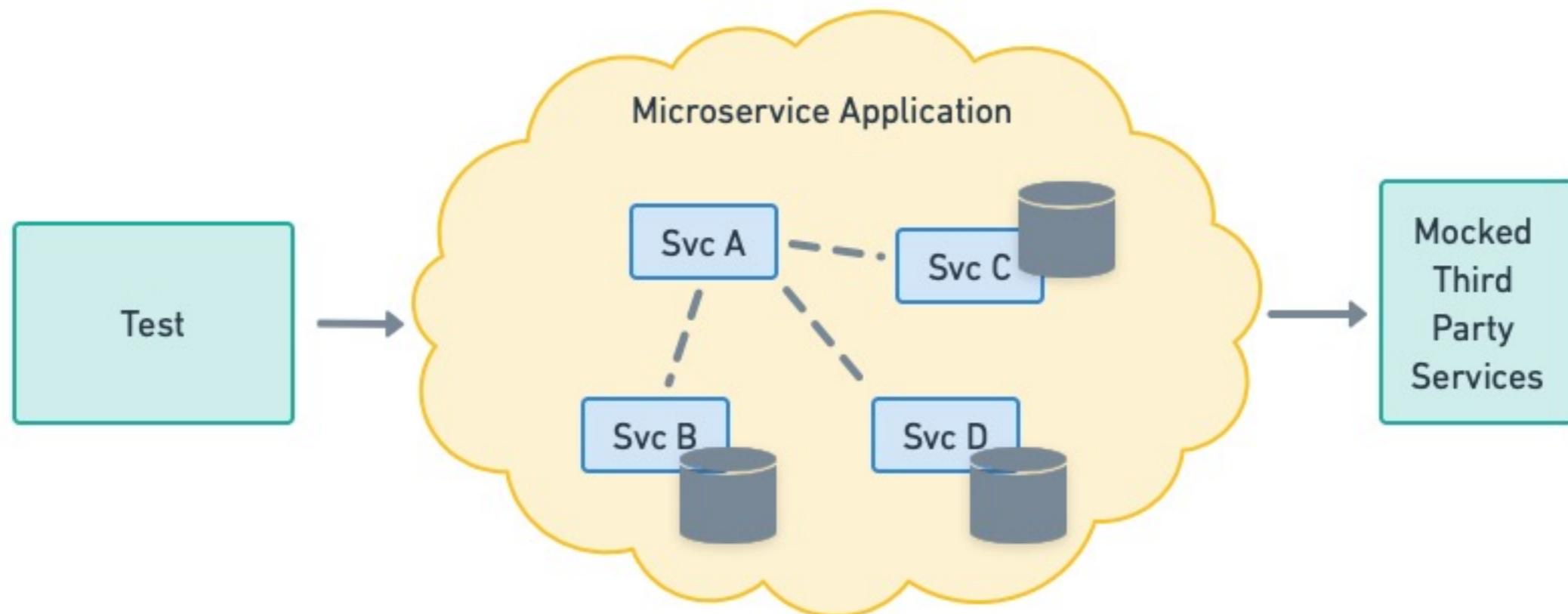
Test strategy



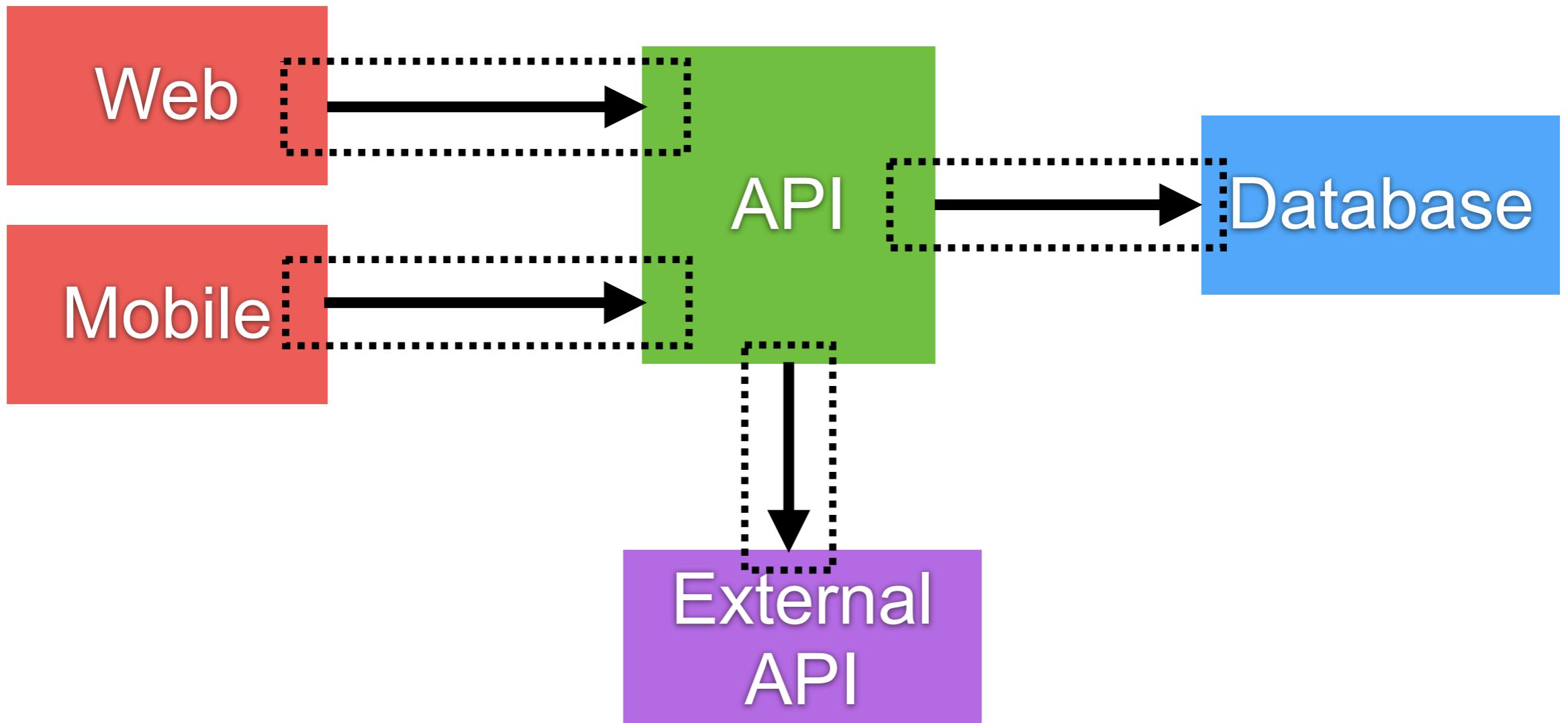
<https://martinfowler.com/articles/microservice-testing/>



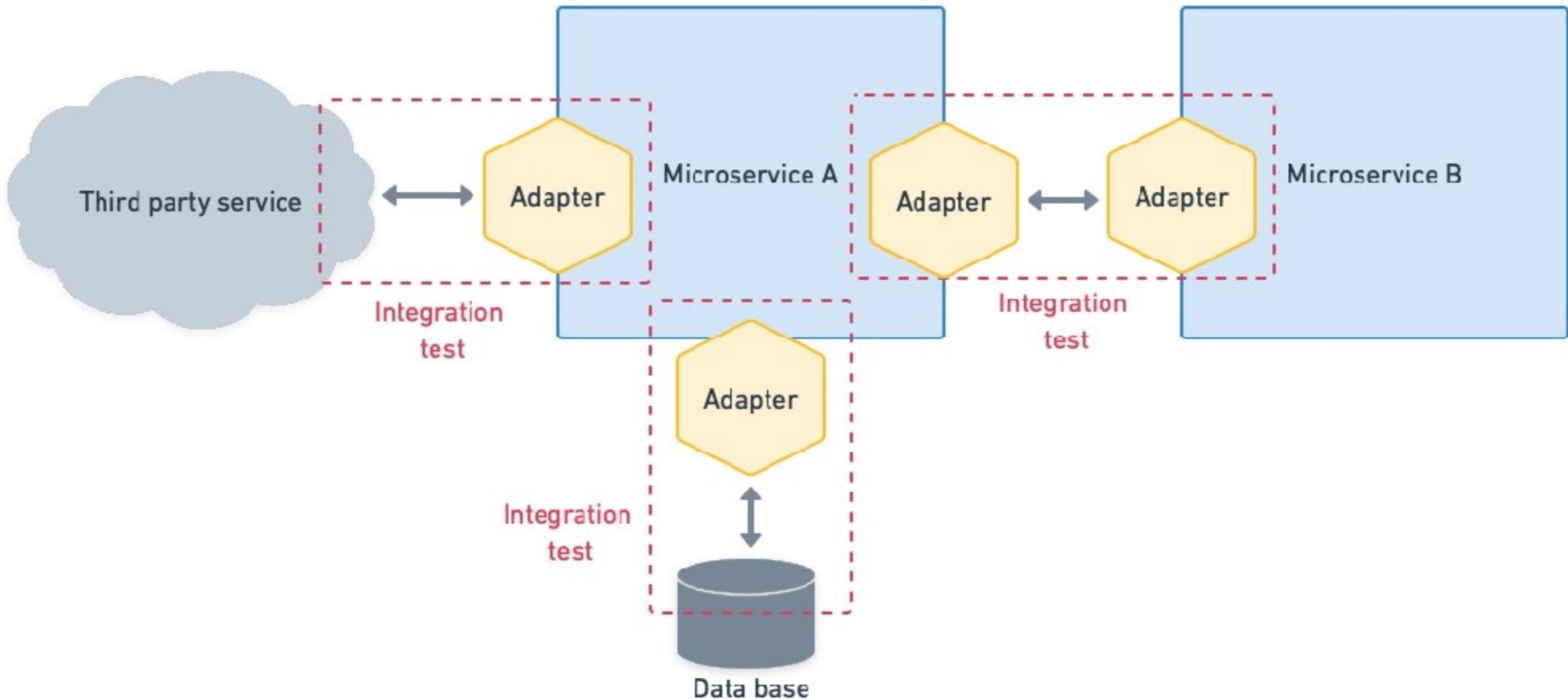
End-to-End testing



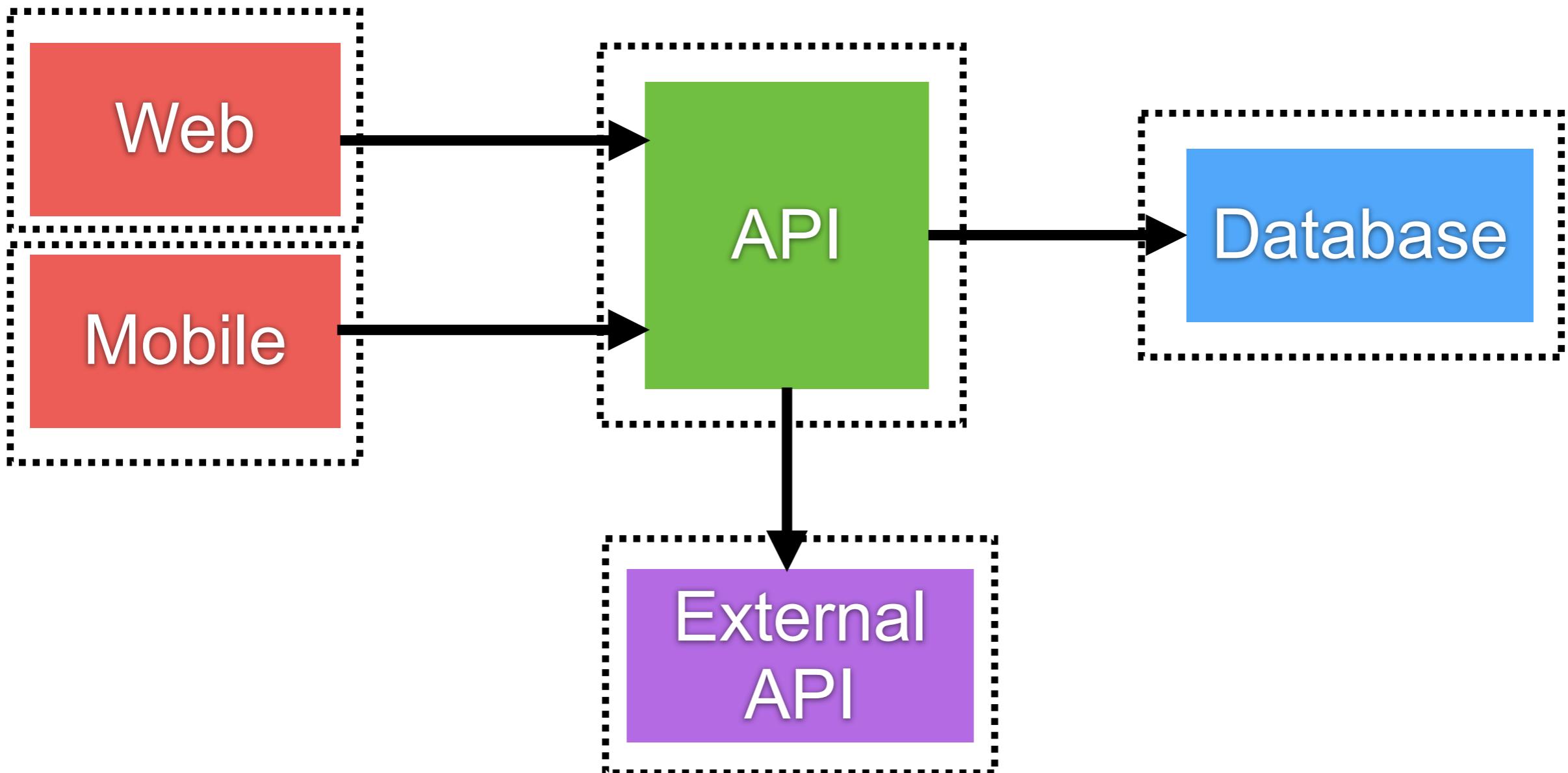
Integration testing



Integration testing

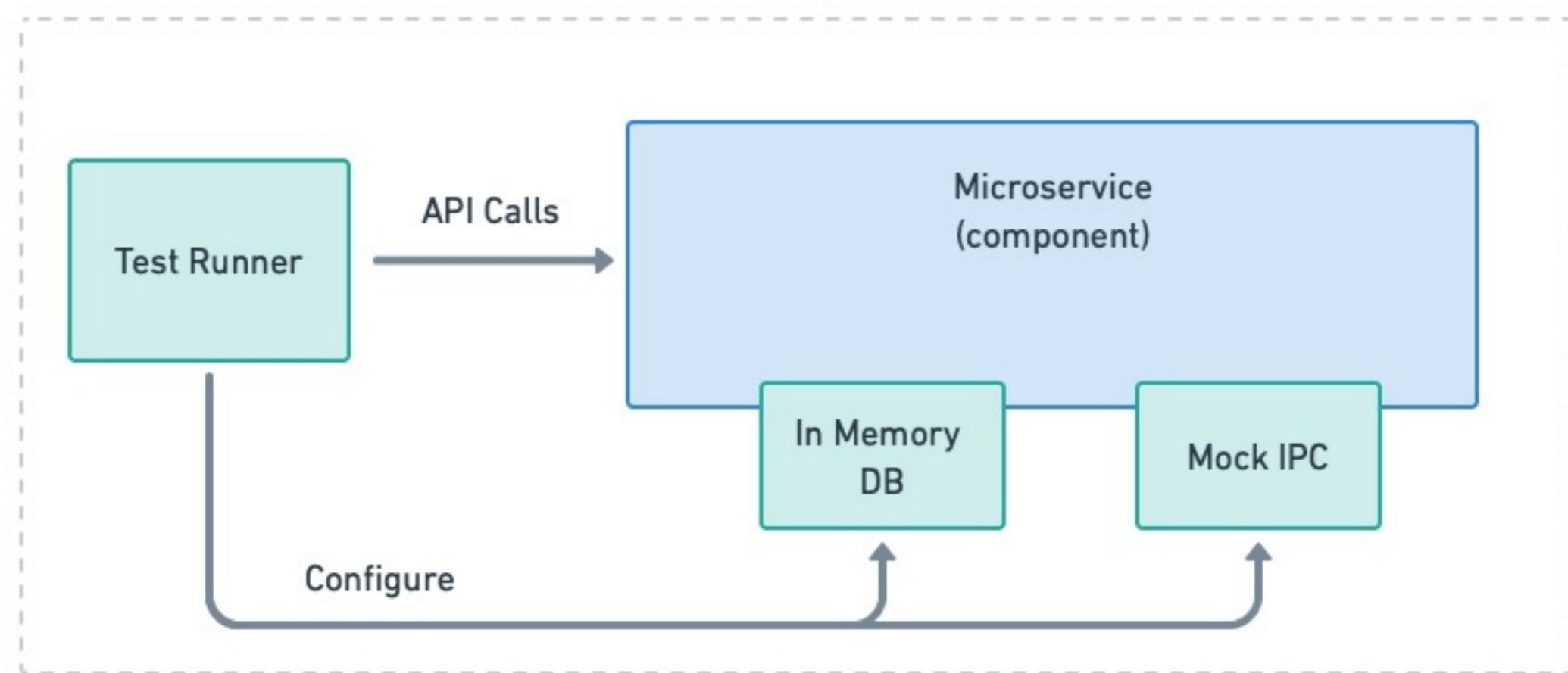


Service component testing

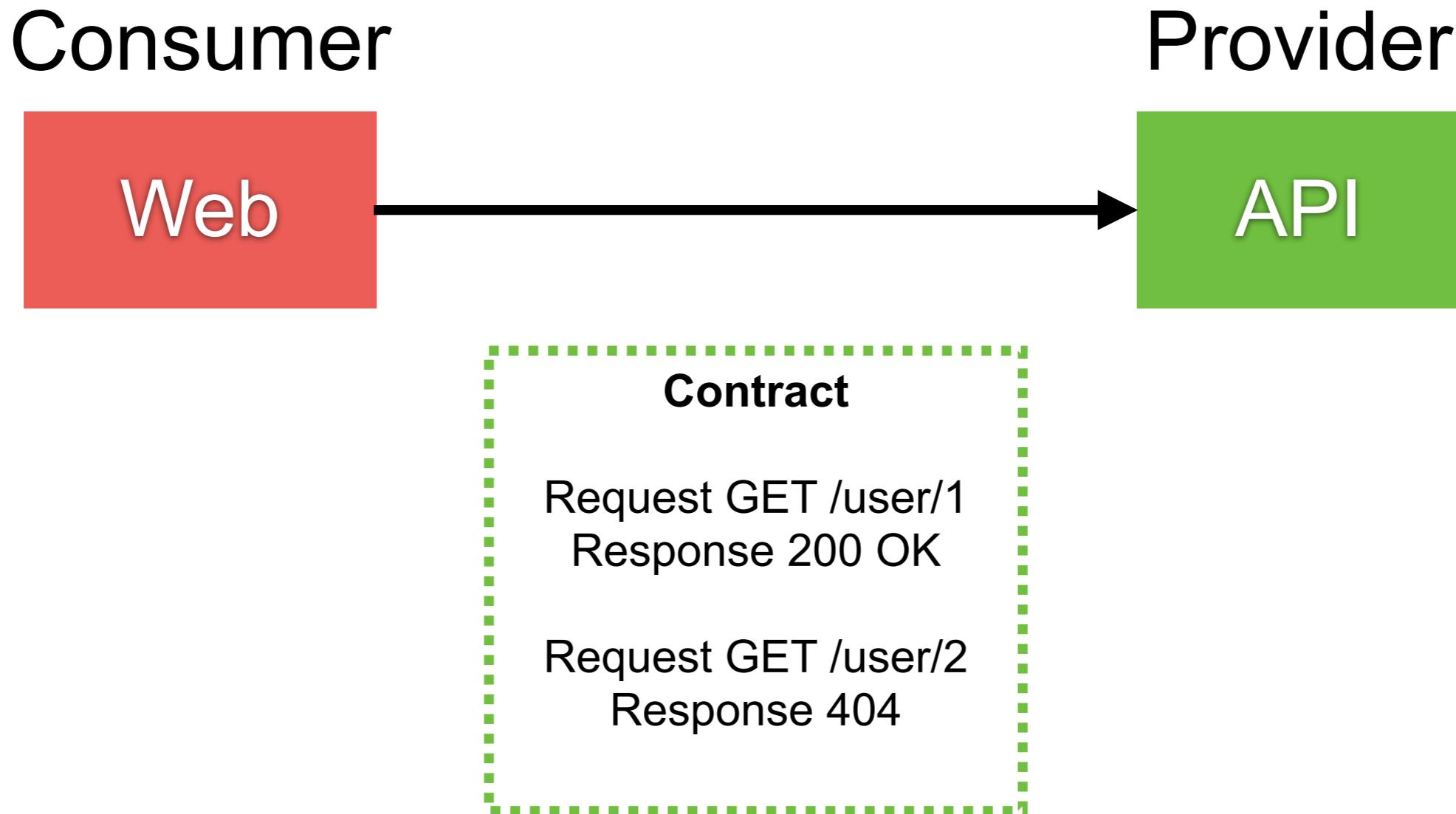


Service component testing

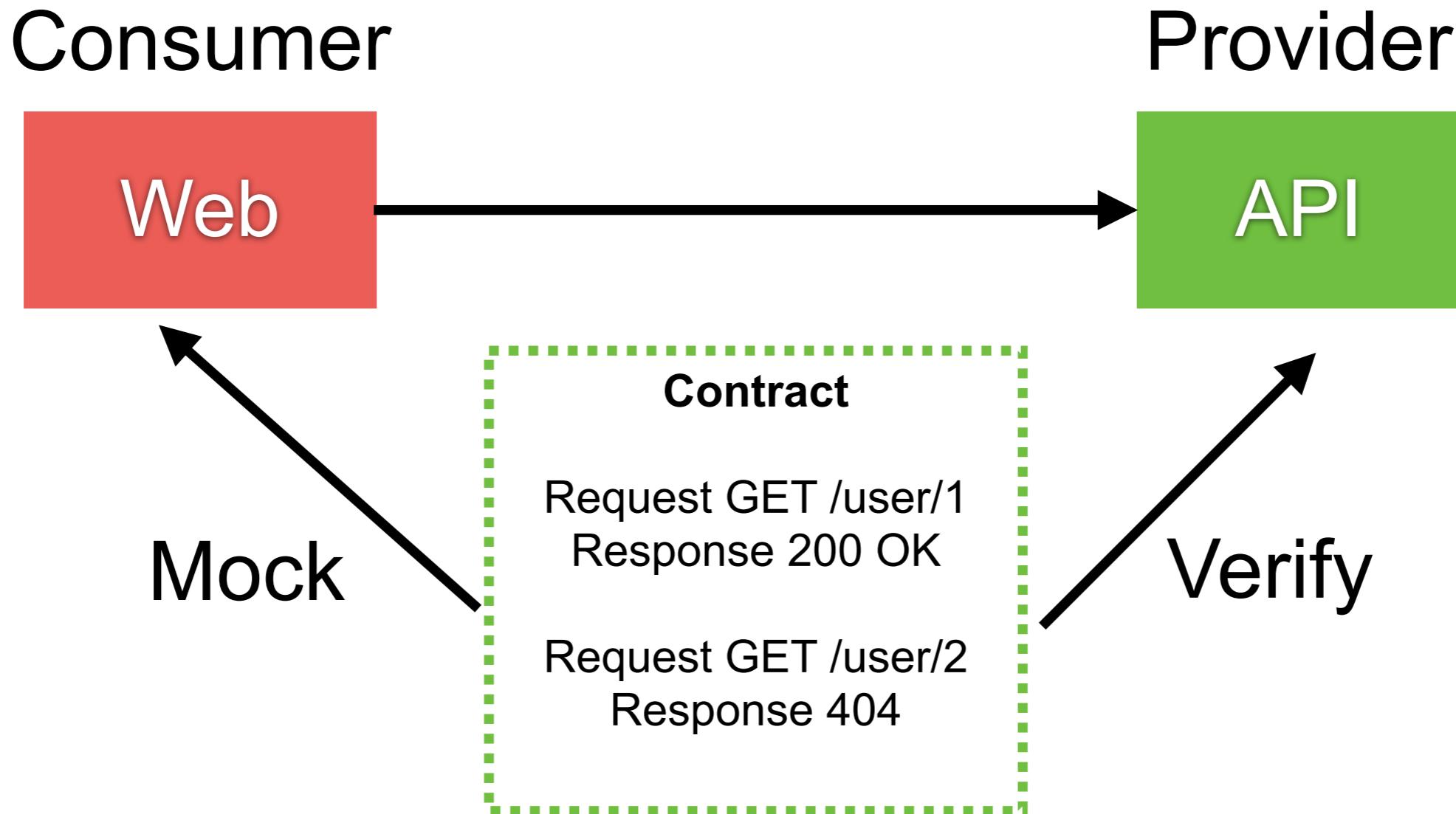
In-process component test



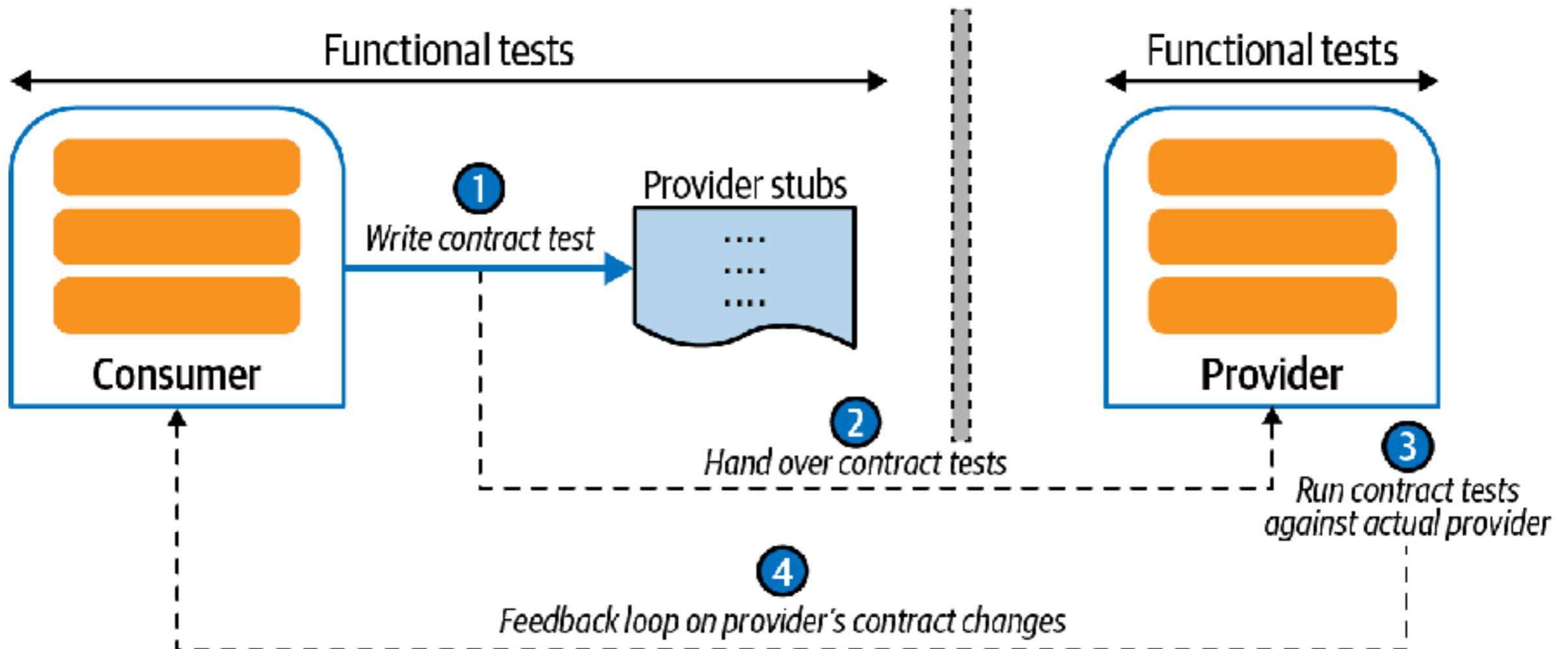
Contract testing



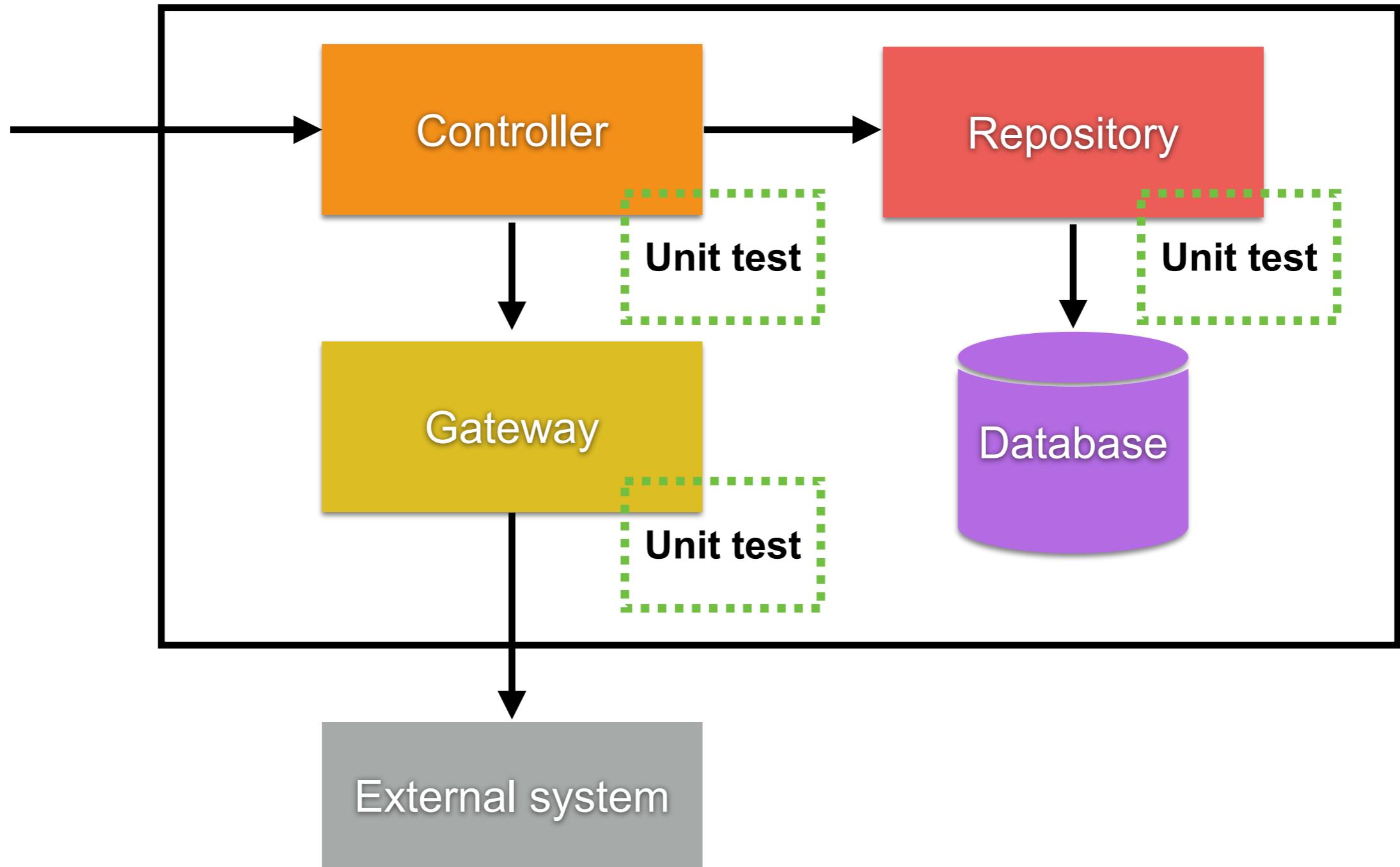
Contract testing



Contract testing



Unit testing



Test Double

Dummy

Stub

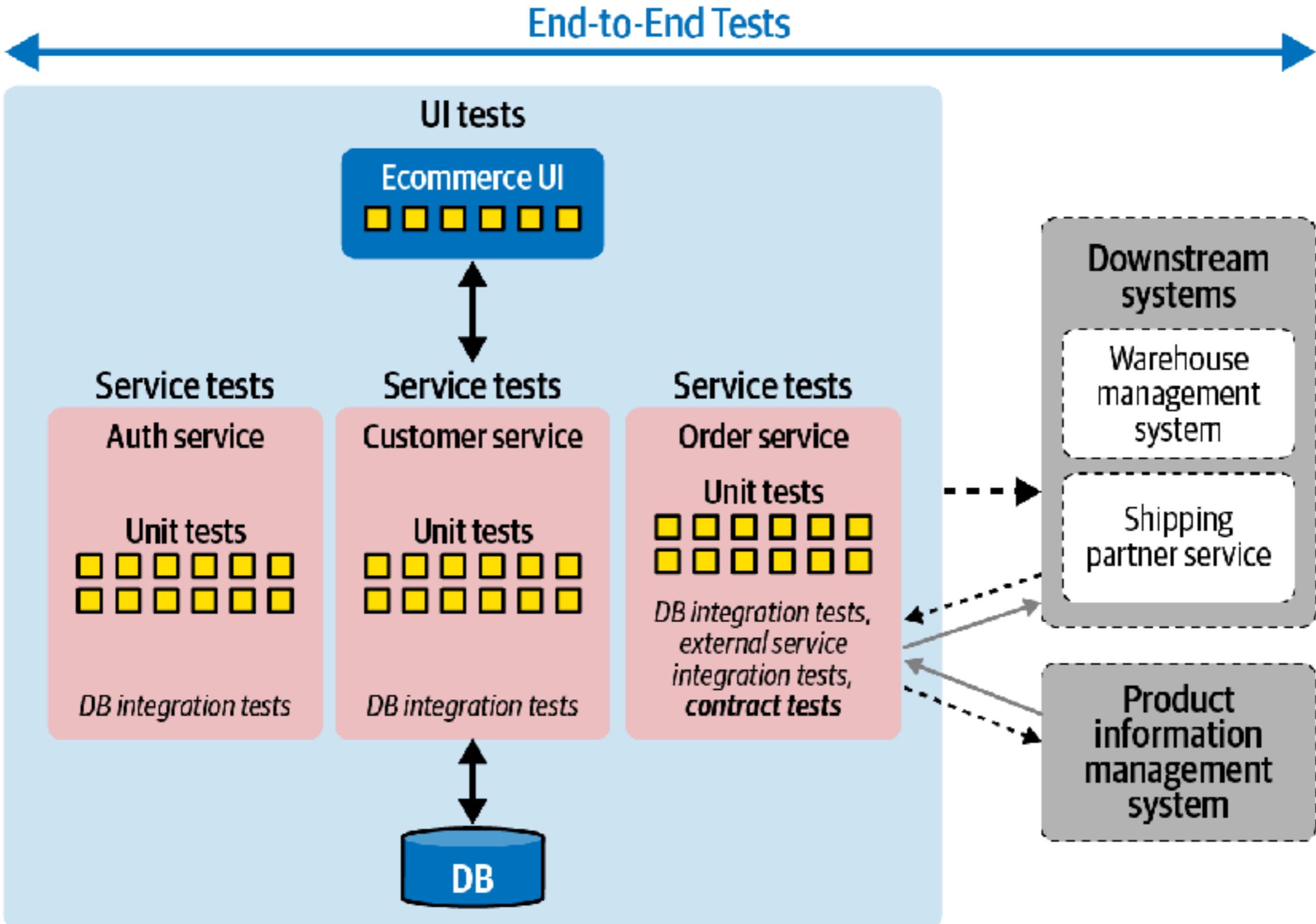
Spy

Mock

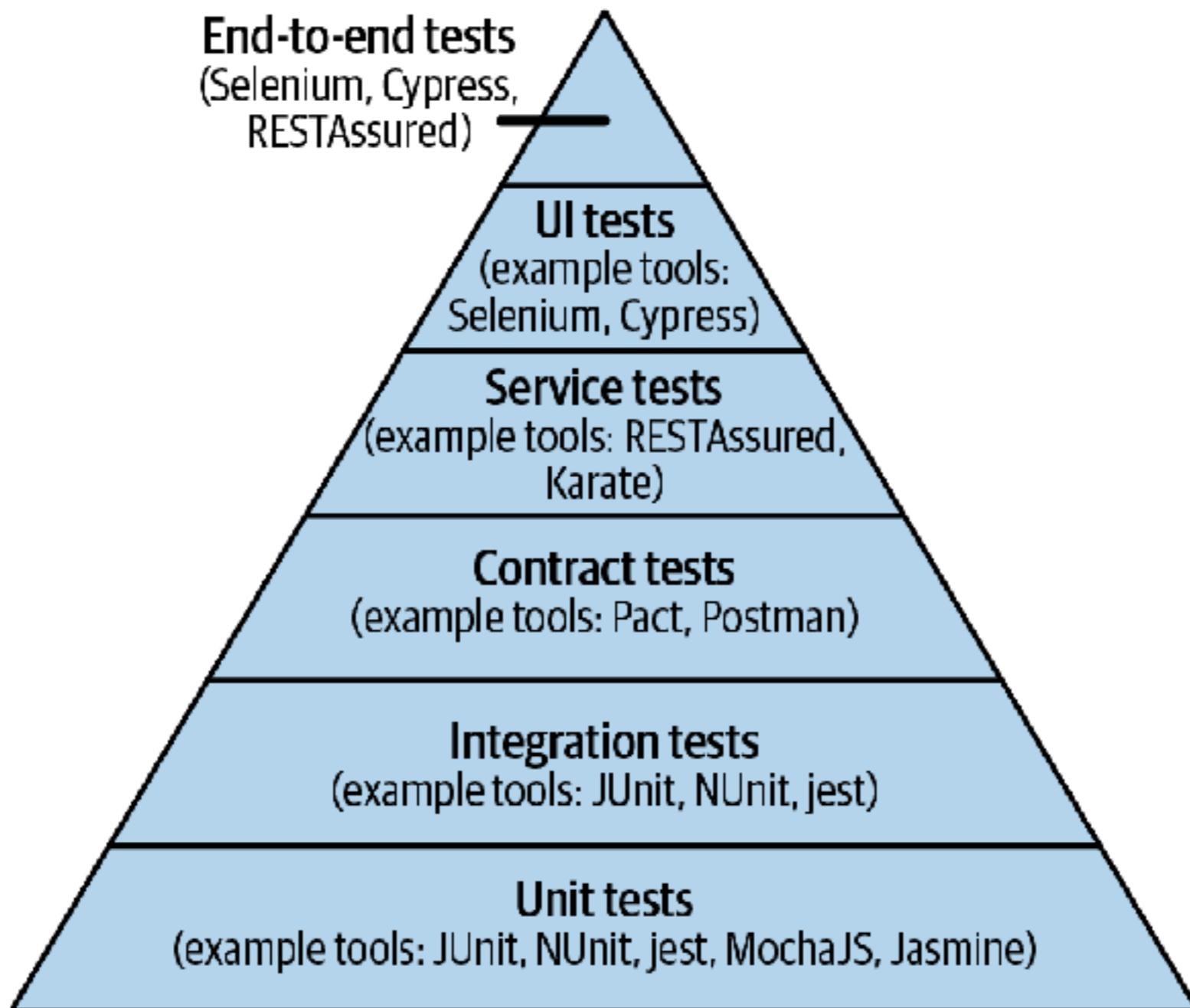
Fake

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





Example Testing Tools



Test design



Economy of Test Design

Easy to understand

Easy to maintain

Readable by the business

One purpose per test

Repeatable

Poor test practices reduce the benefits



Good Test

F.I.R.S.T + U

Fast

Isolate

Repeat

Self-verify

Timely
Thought

Understand



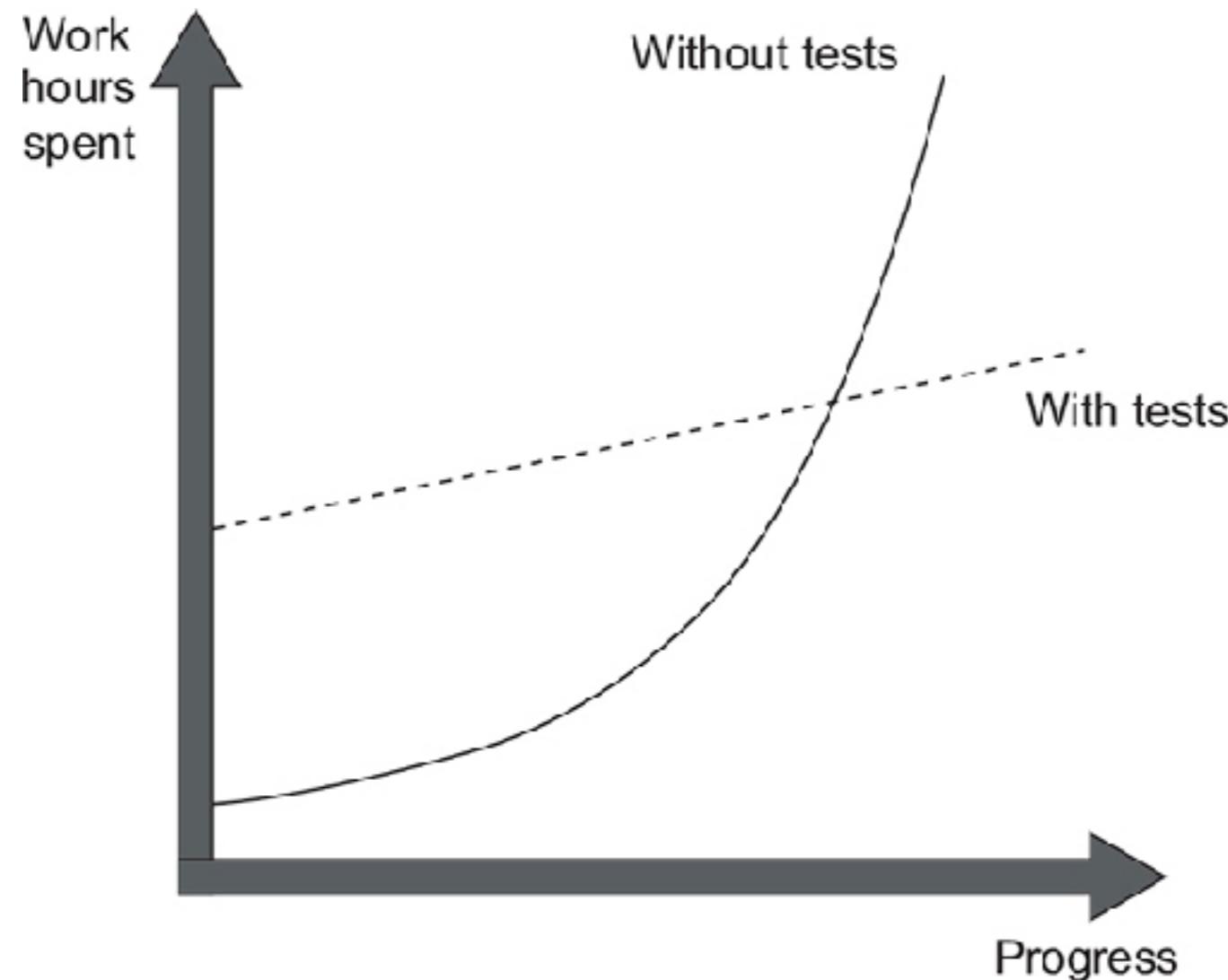
Respect your tests

Don't ignore it if it fails
Fix the code or fix the test
Always refactor or improve

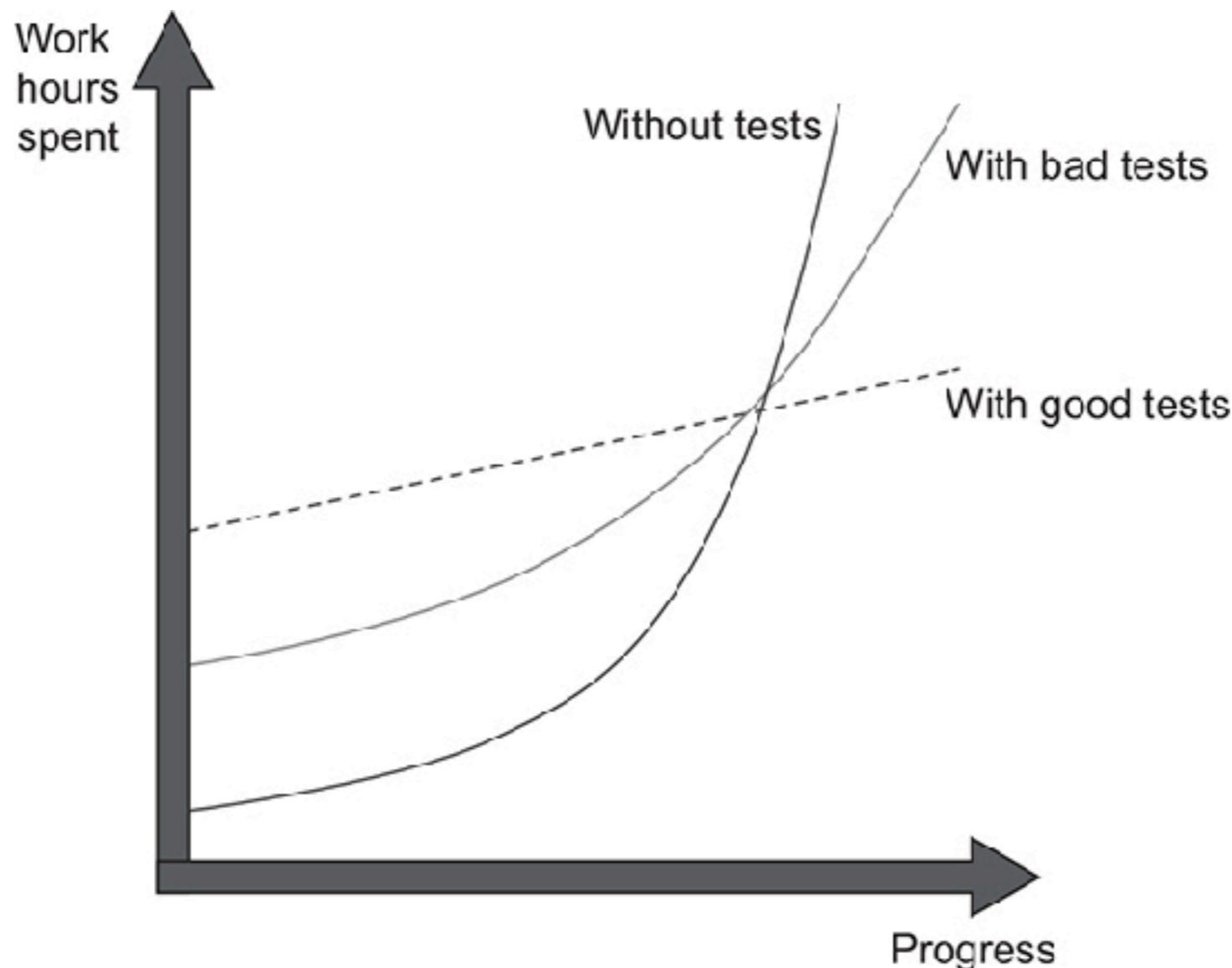
**100% of regression tests
must pass all the time**



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



Start with simple
Use feedback to improve



Workshop



Register workshop !!

As a new user,

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

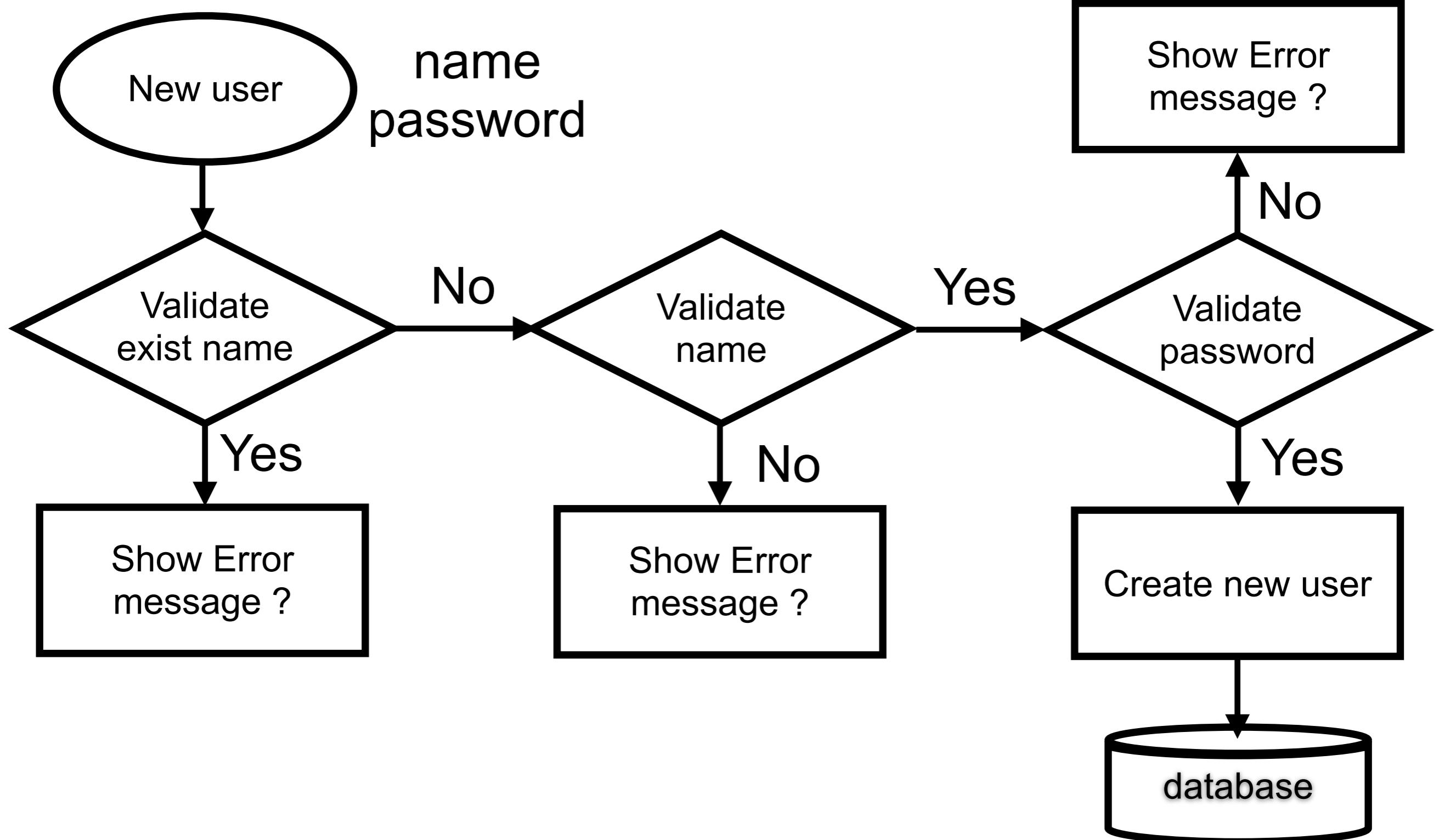
So that only I can access my information



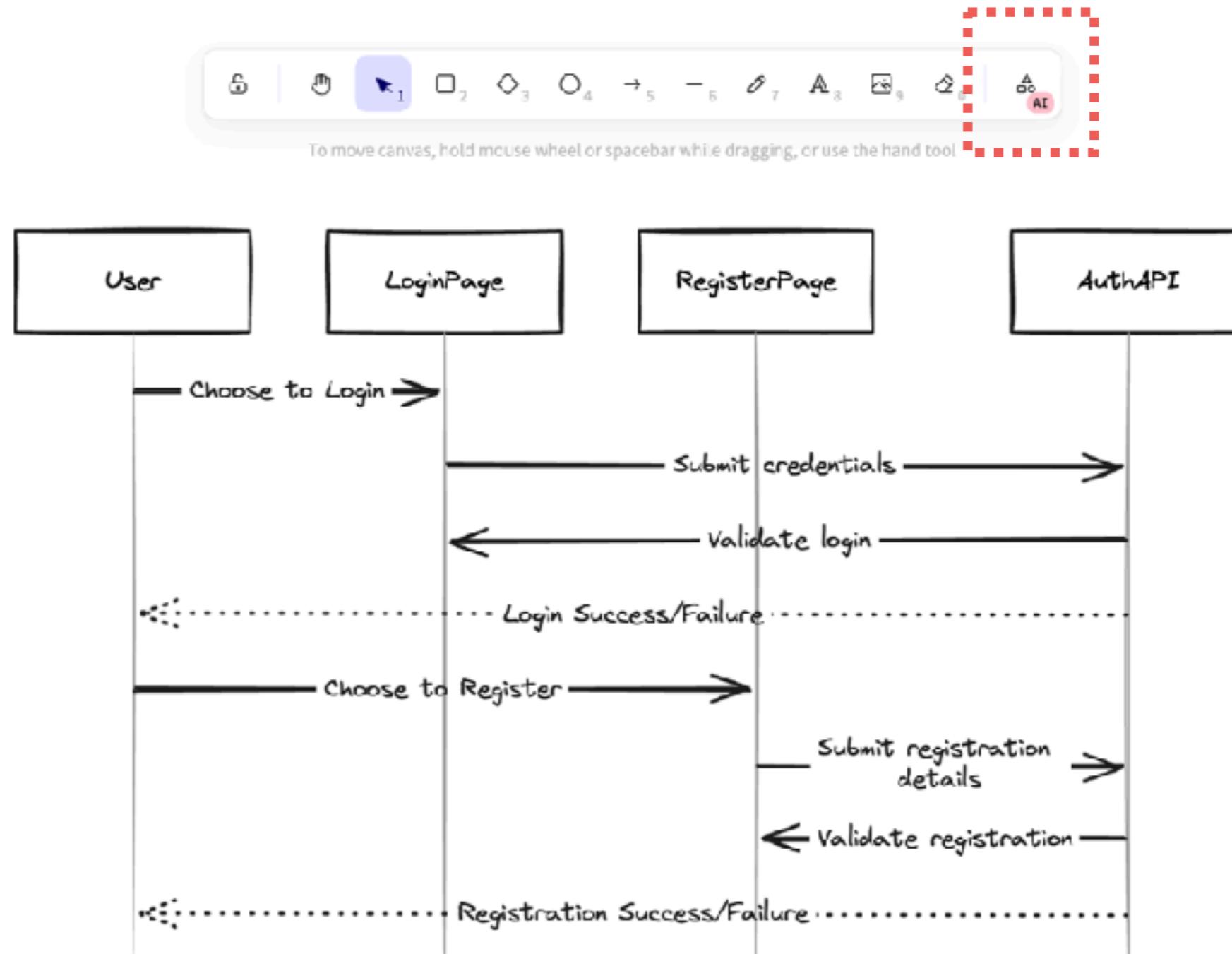
Design your Test cases ?



Flow ?



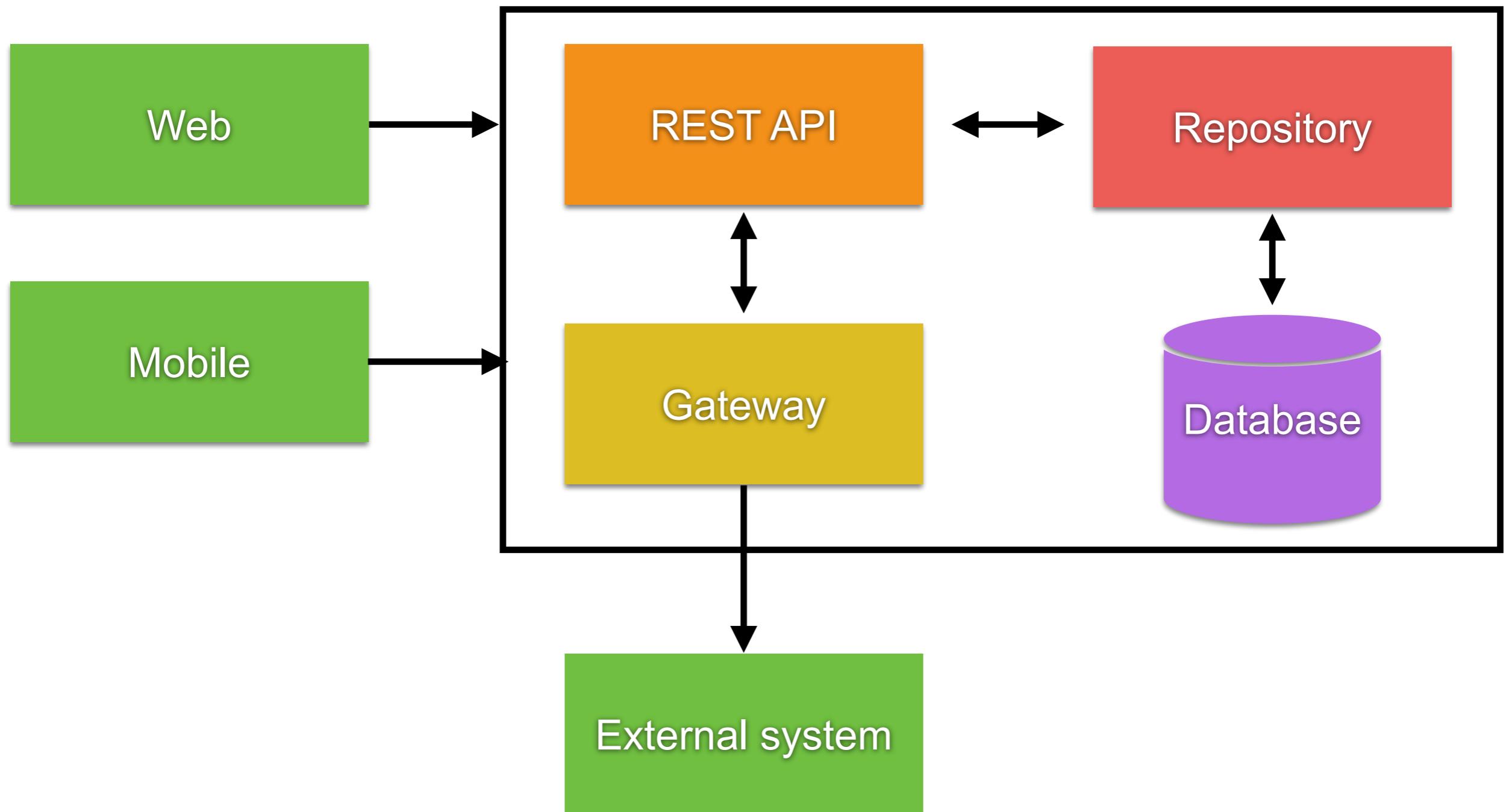
Generate Flow with AI



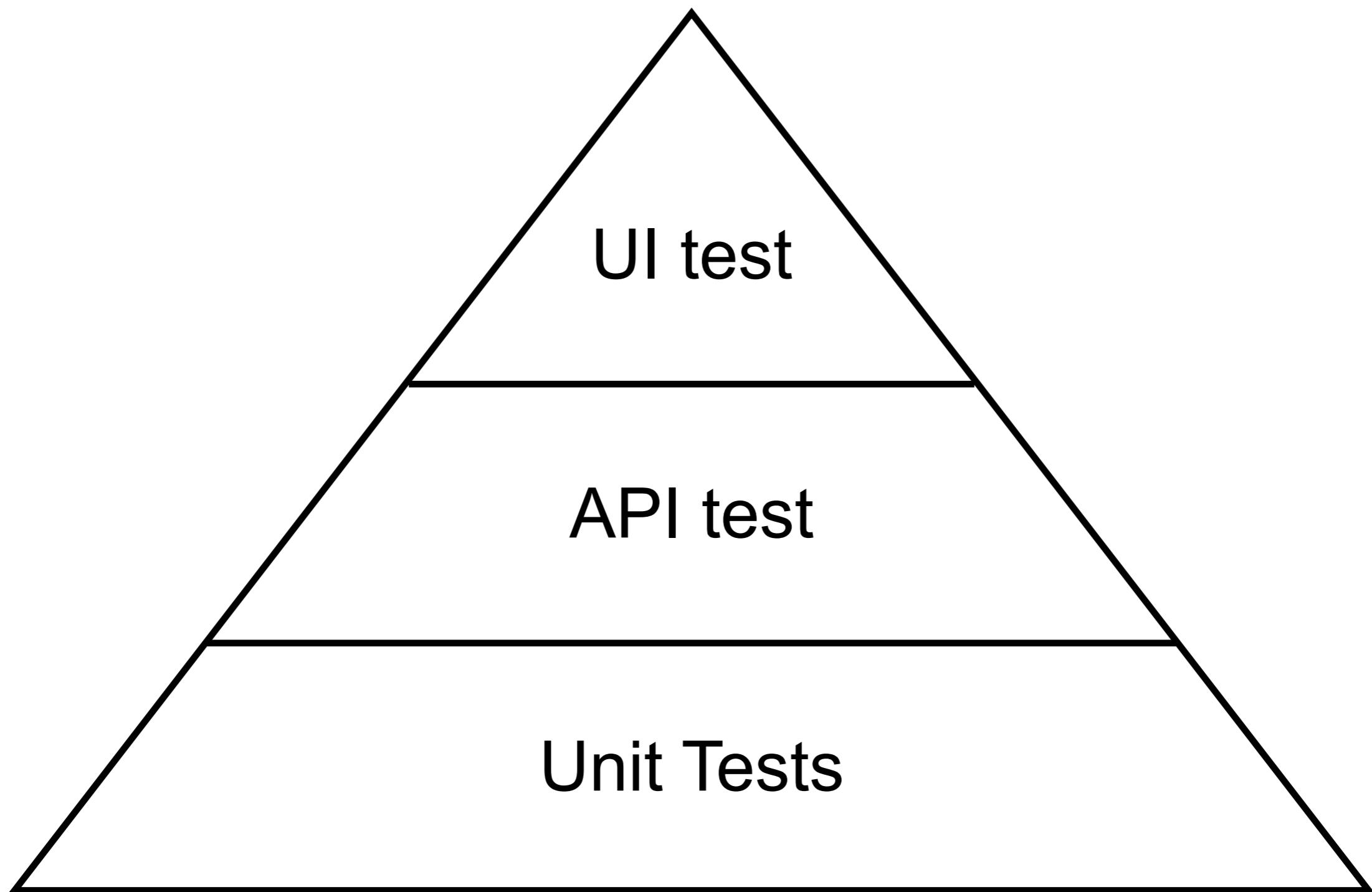
<https://excalidraw.com/>



Architecture



Test cases with Test Level ?



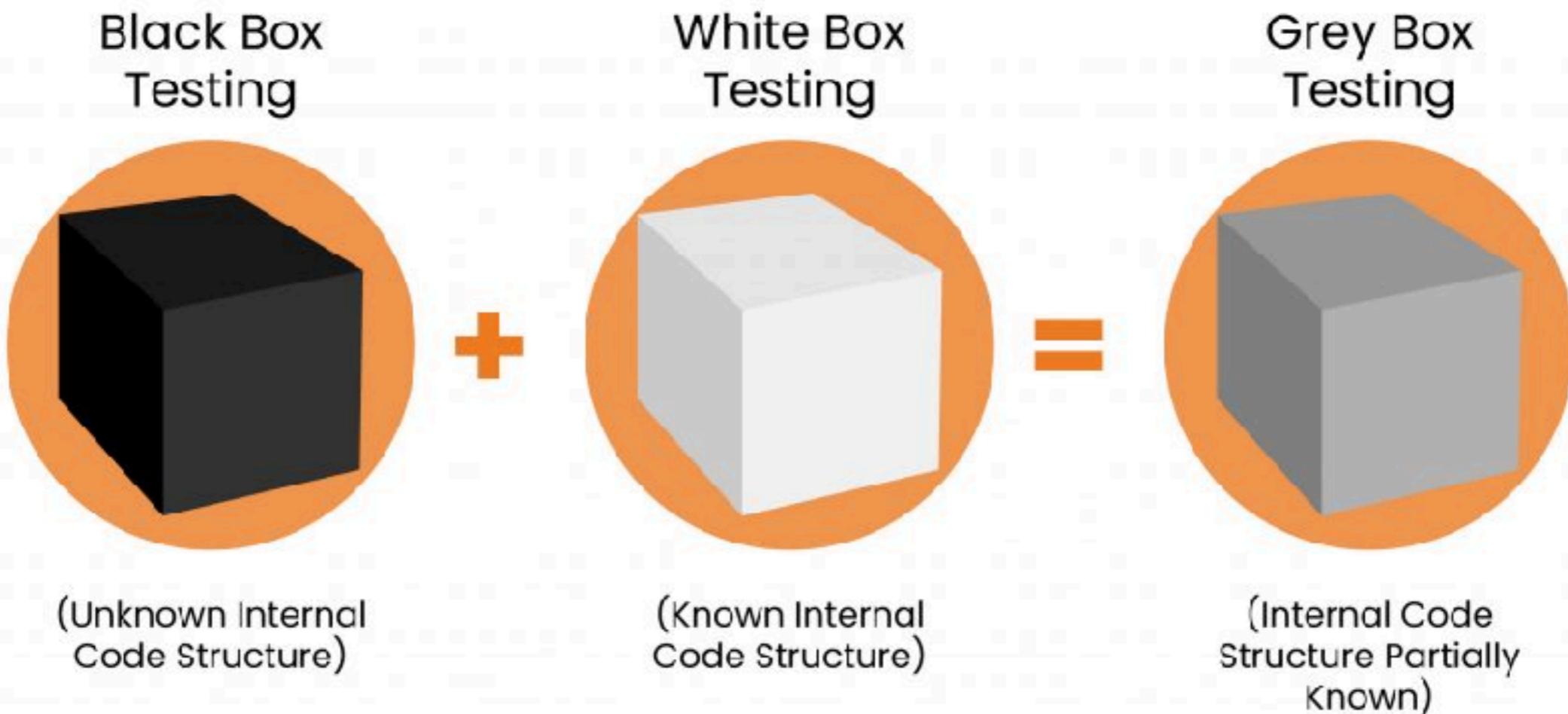
Testing Tools

External
testing

Internal
testing

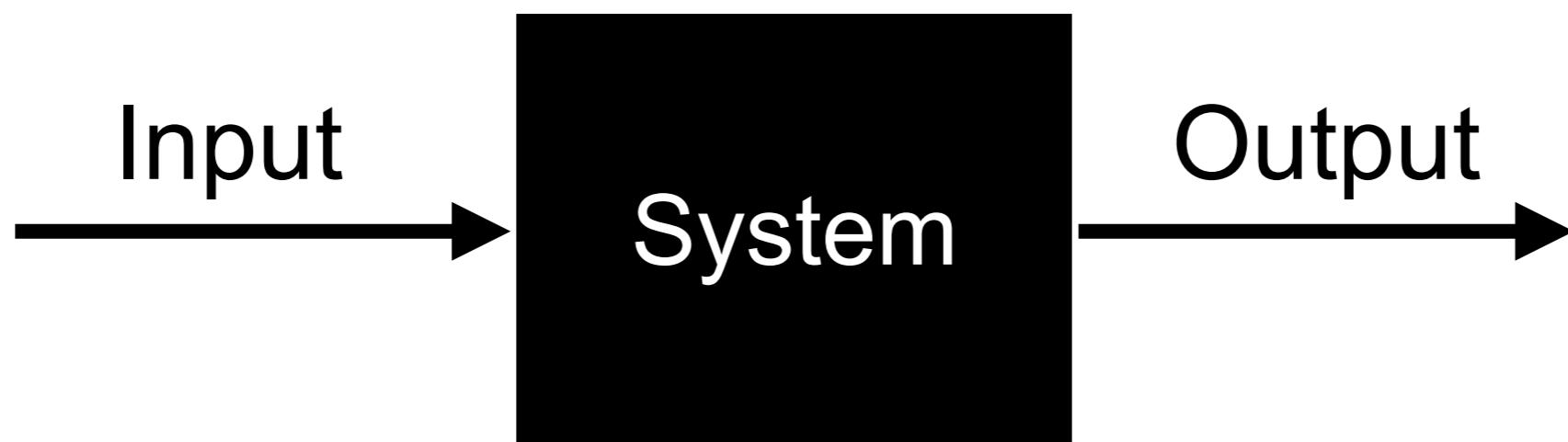


Types of Testing Methods



External Testing

Black box testing

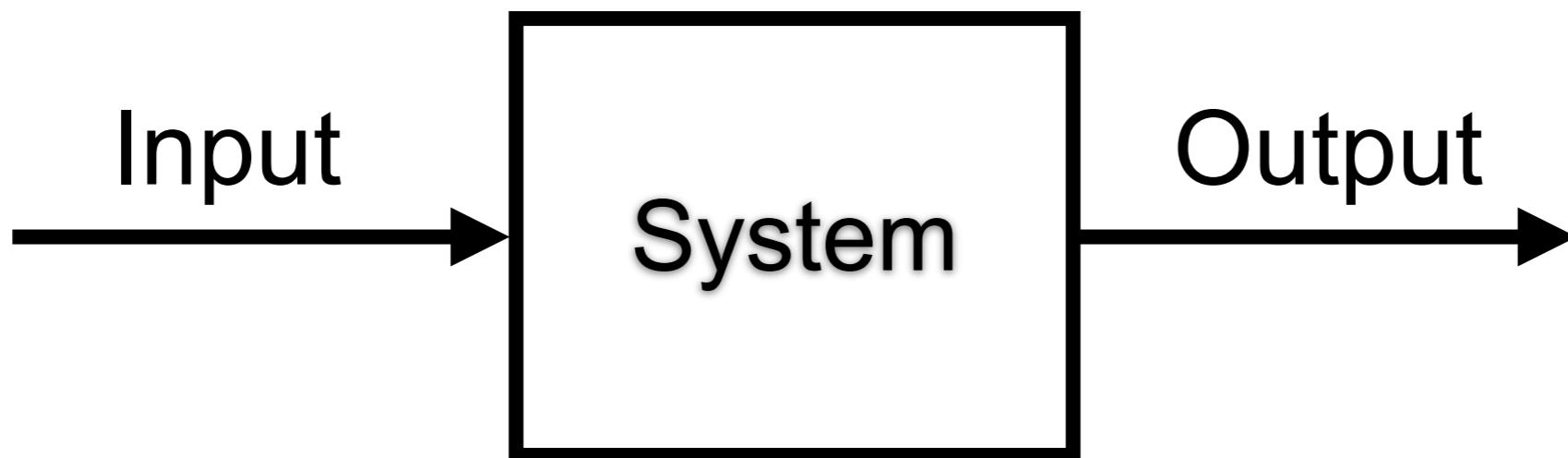


End-to-end testing



Internal Testing

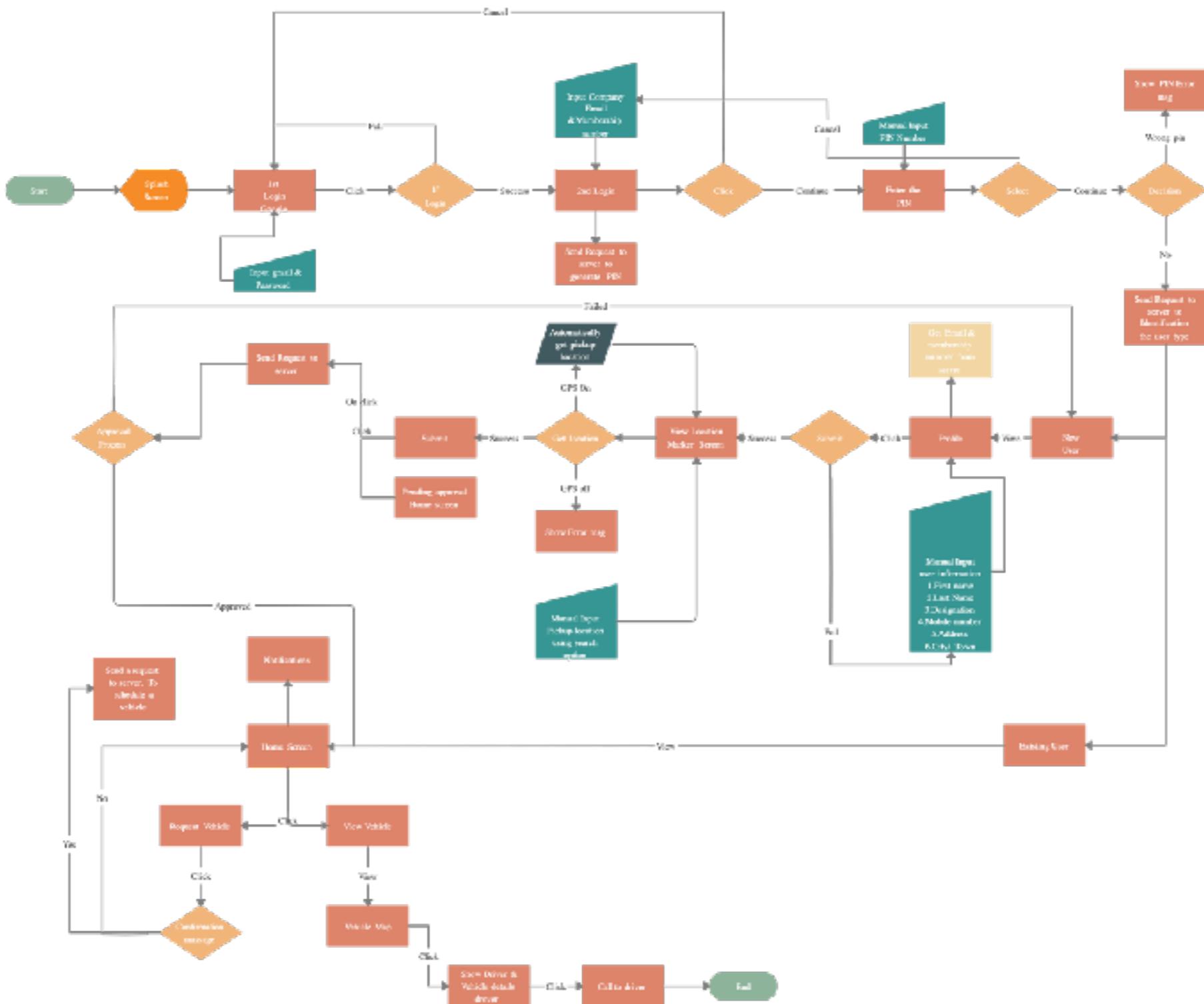
White box testing



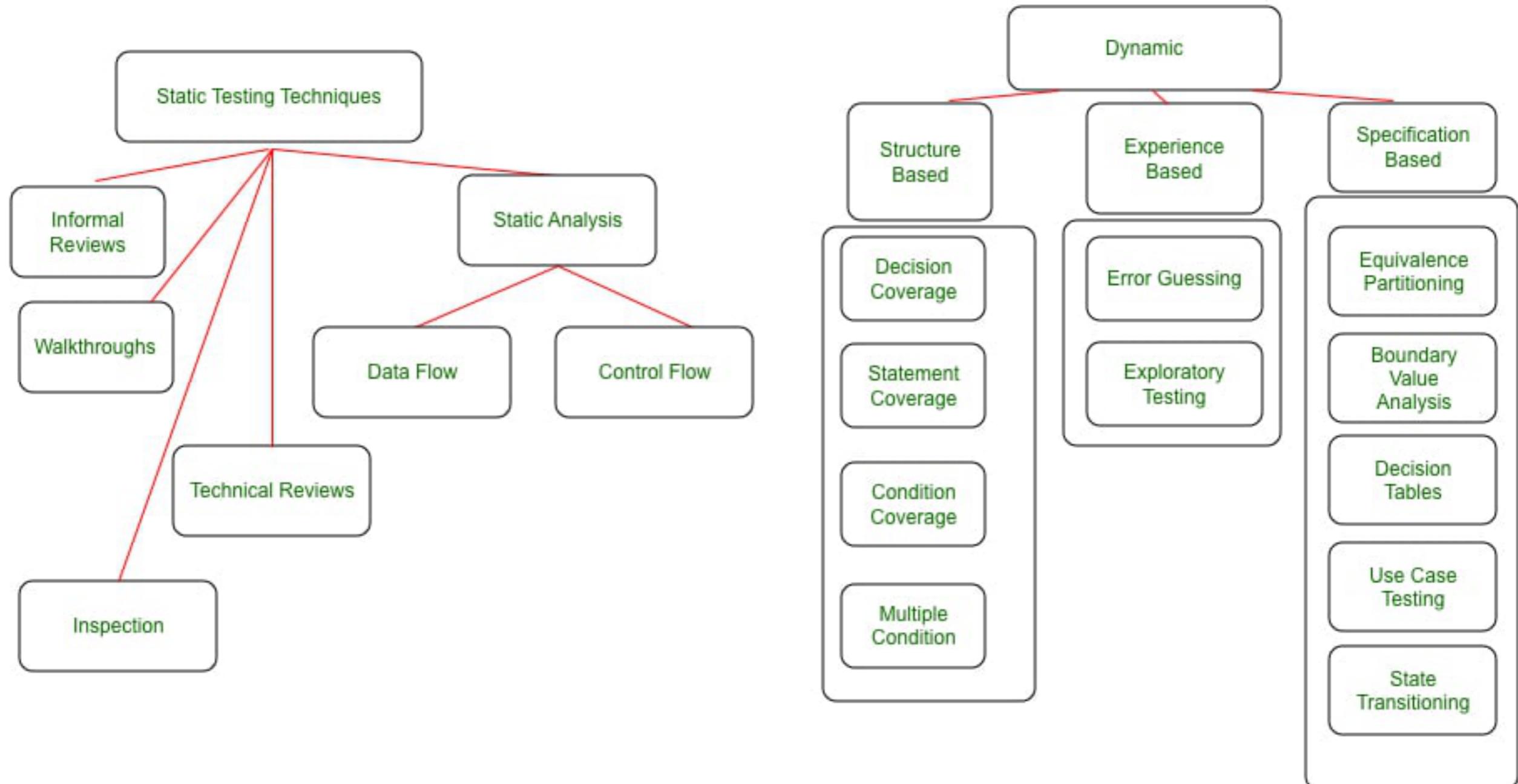
**Unit, Integration,
Contract, component testing**



Internal Testing



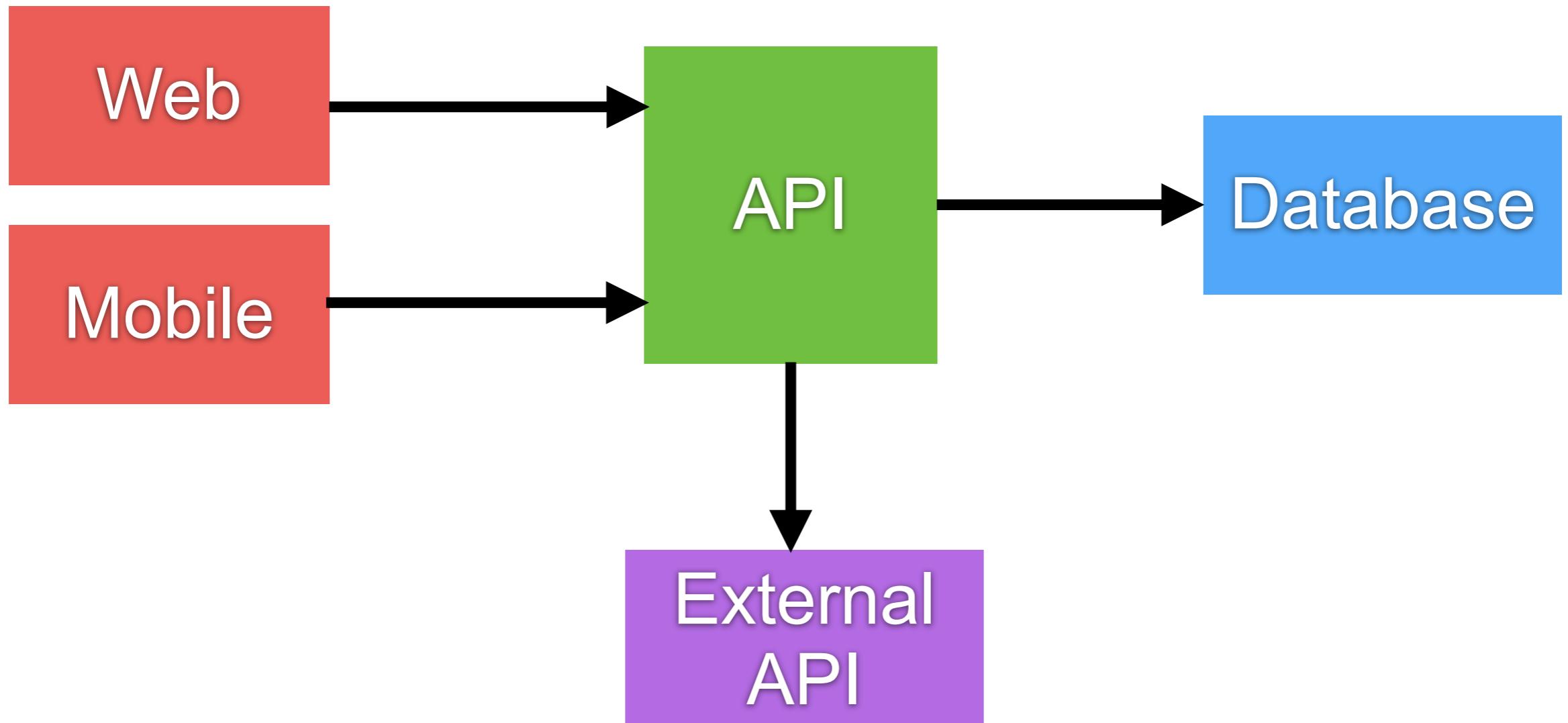
Testing Techniques



<https://www.geeksforgeeks.org/software-testing-techniques/>



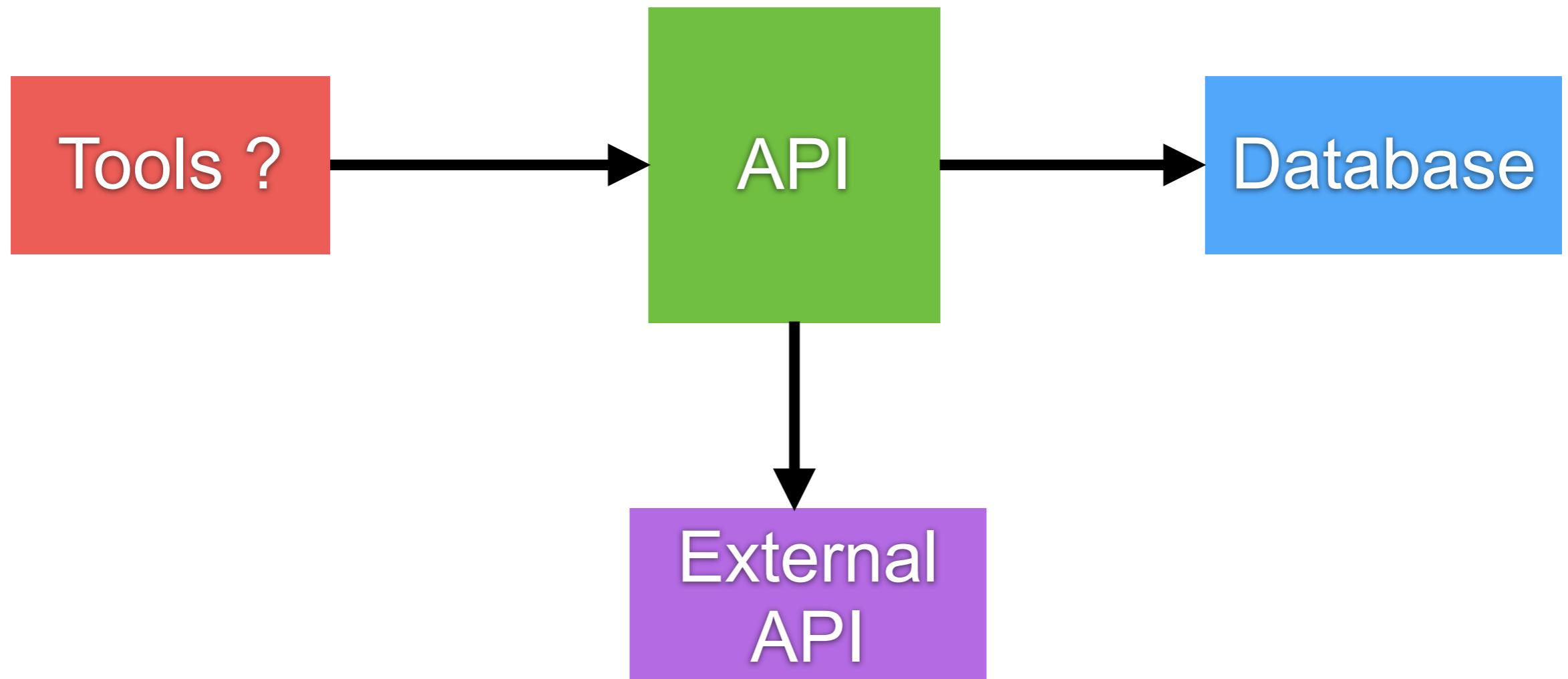
Architecture



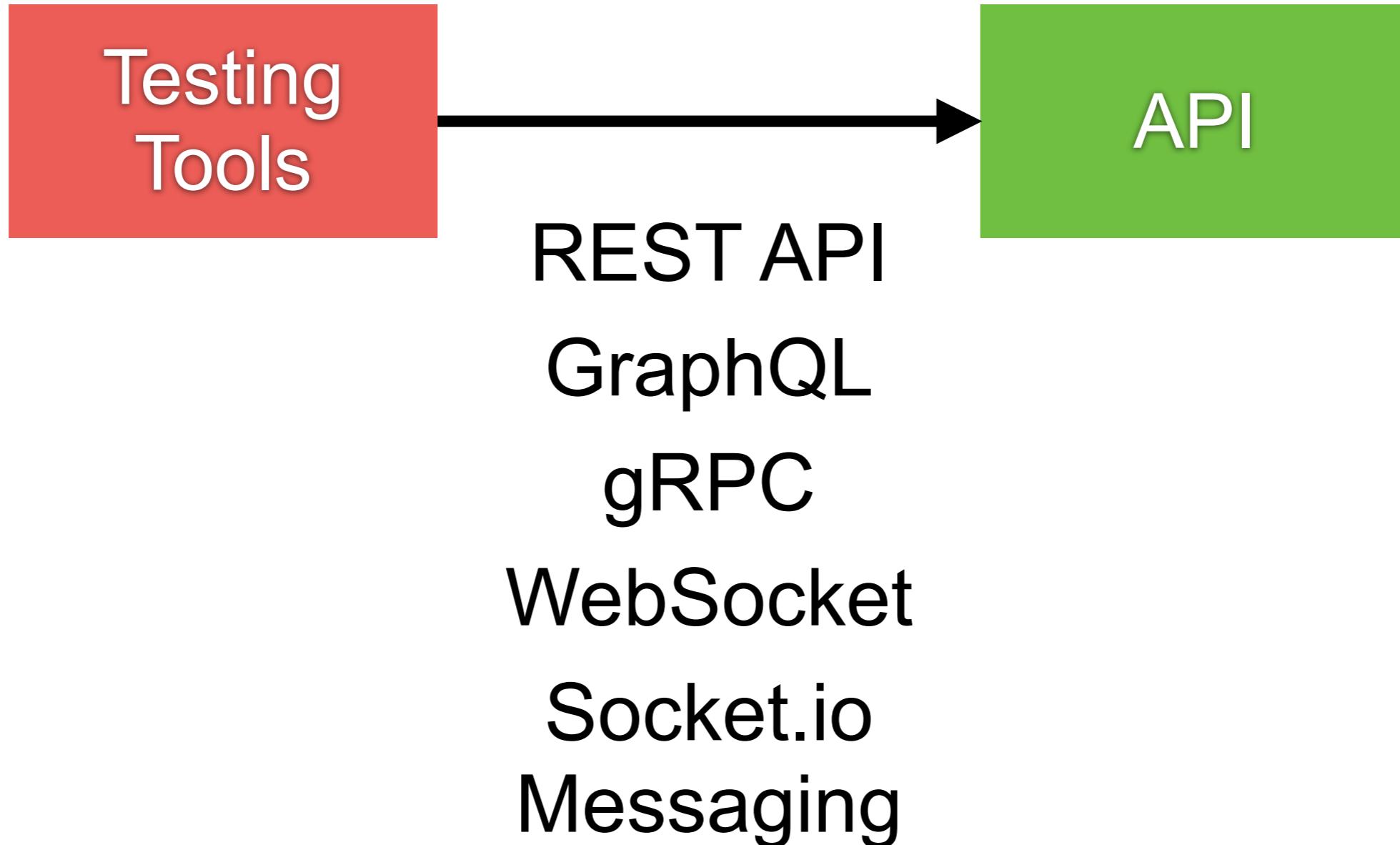
API Testing



API Testing ?



External testing for API



External testing for API

List of tools



POSTMAN



bruno



External testing for API

Write test script and coding



Robot Framework

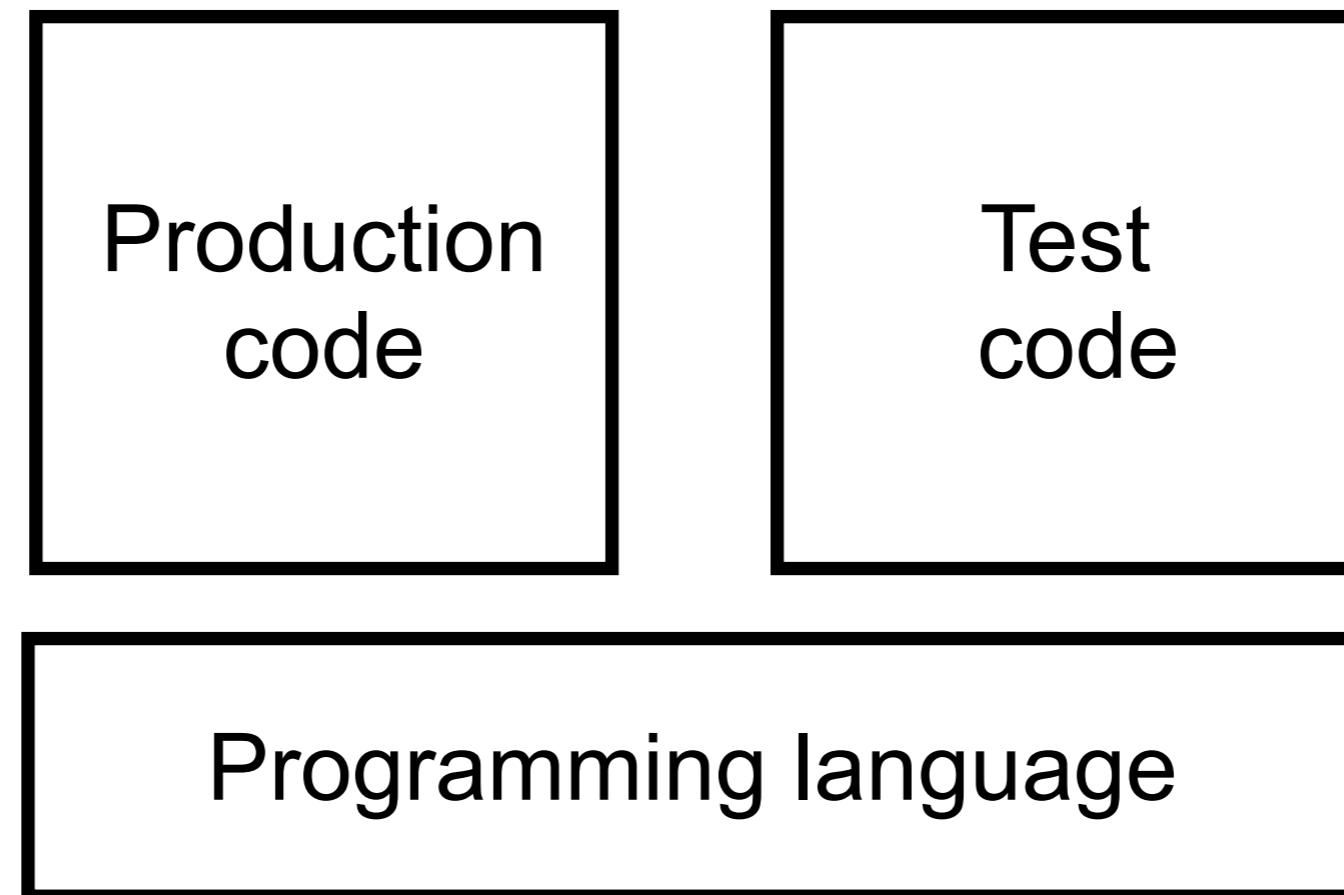


Playwright



External testing for API

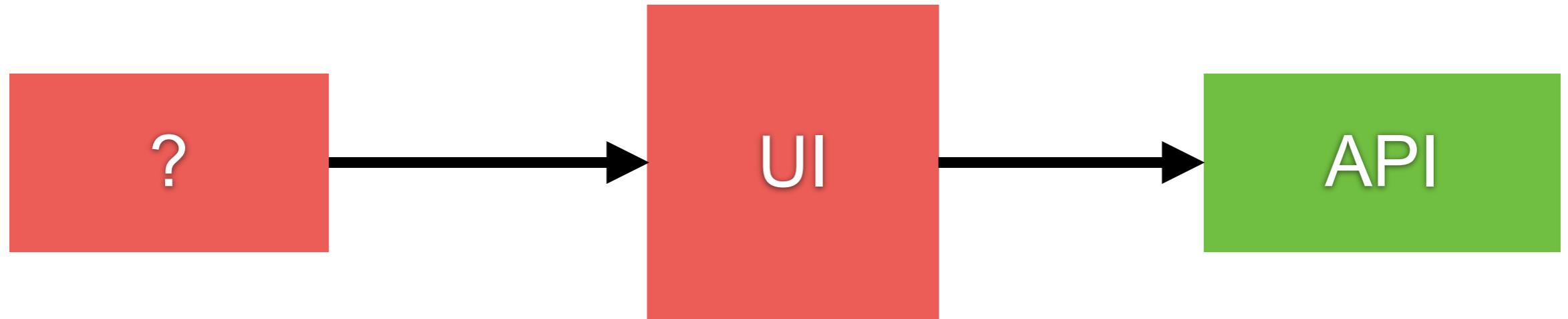
Write test cases with programming language



UI Testing



UI Testing ?



External testing for UI

Write test script and coding



Robot Framework



Playwright



External testing for UI

Record and Play back tools



Playwright

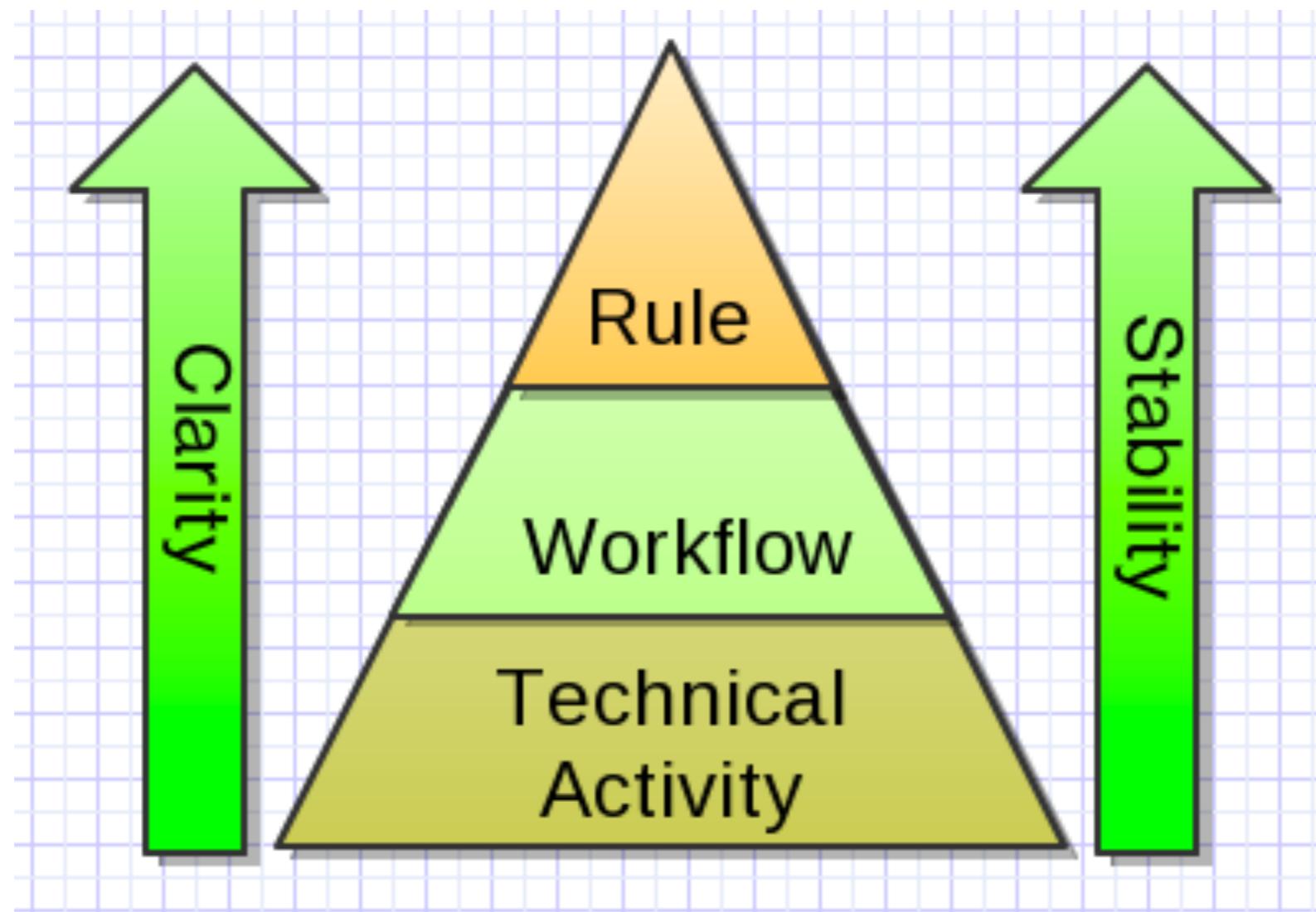


Selenium IDE

Google Chrome Recorder



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

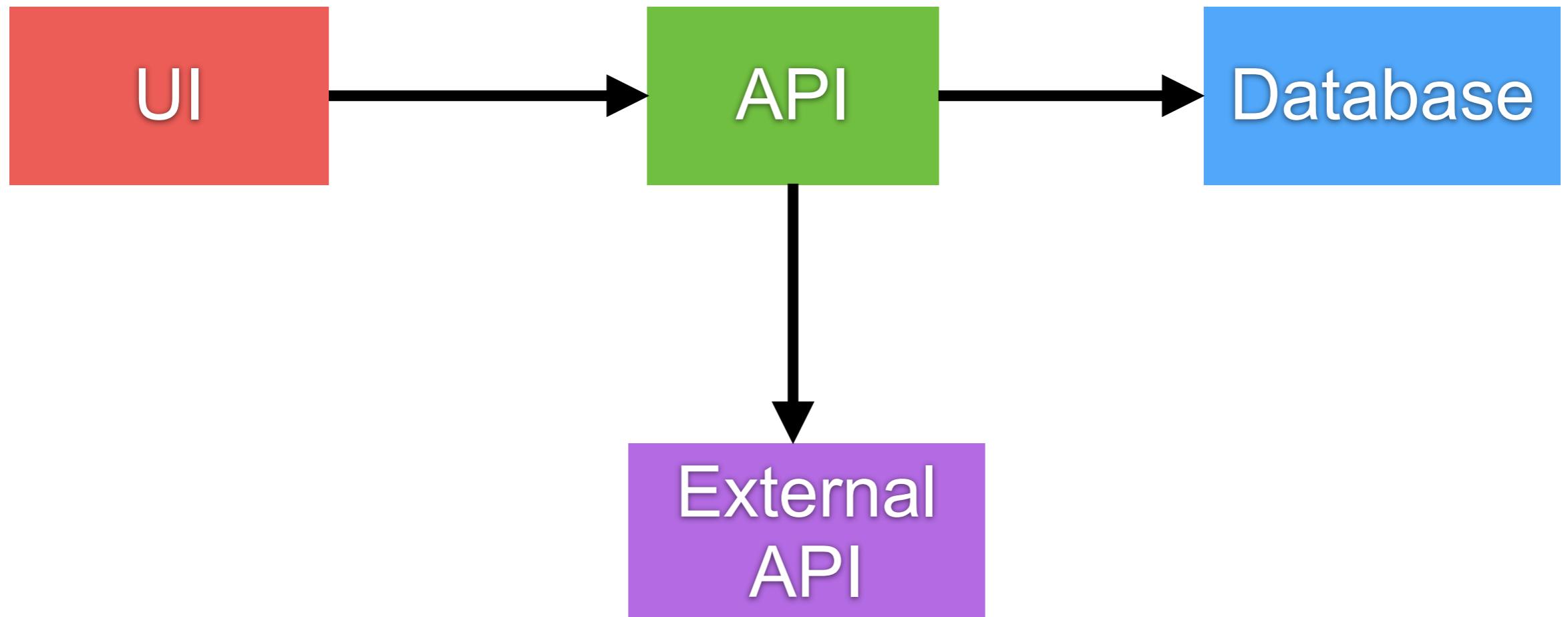
what are the technical steps required to exercise the functionality



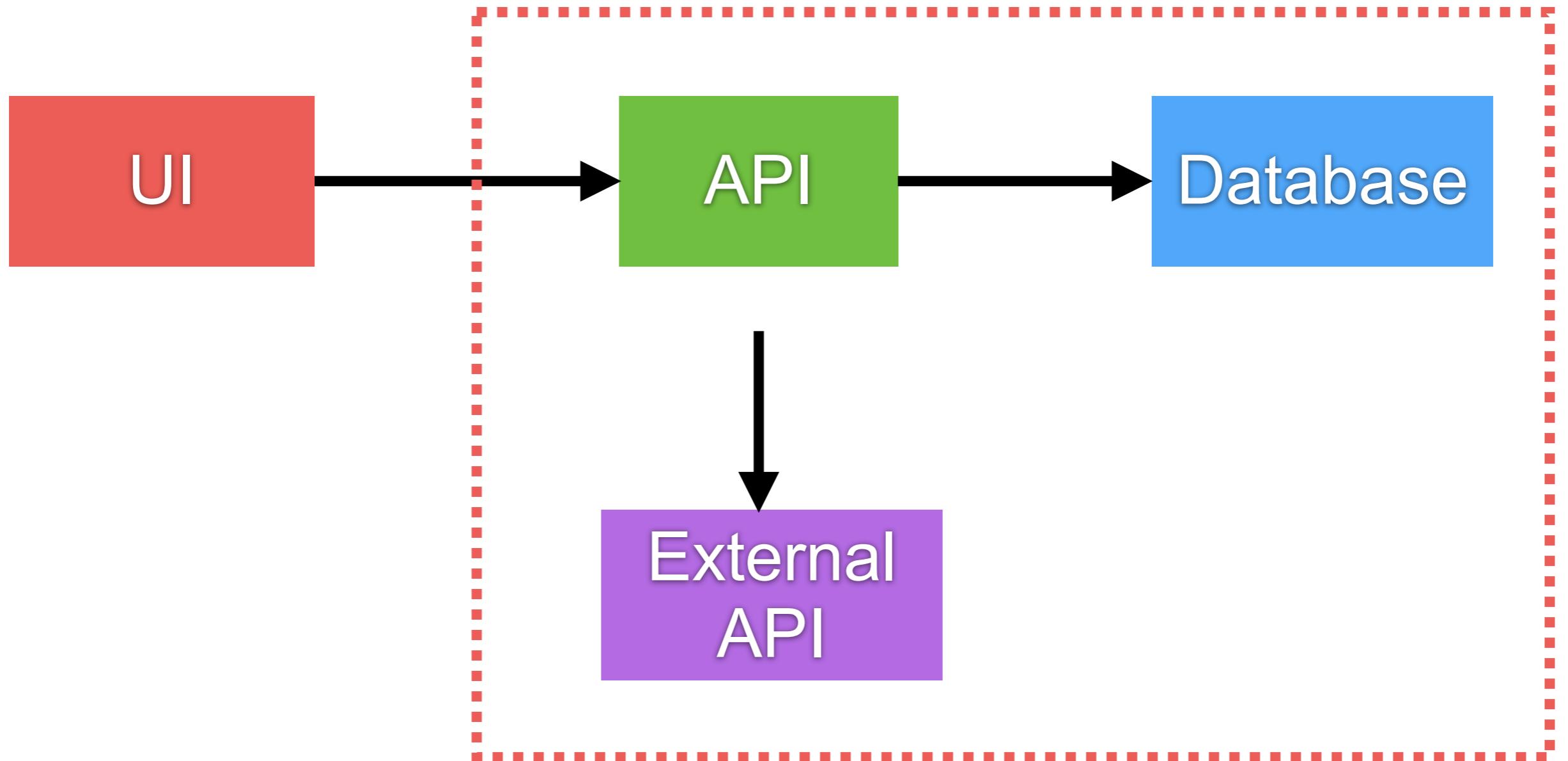
Manage dependencies



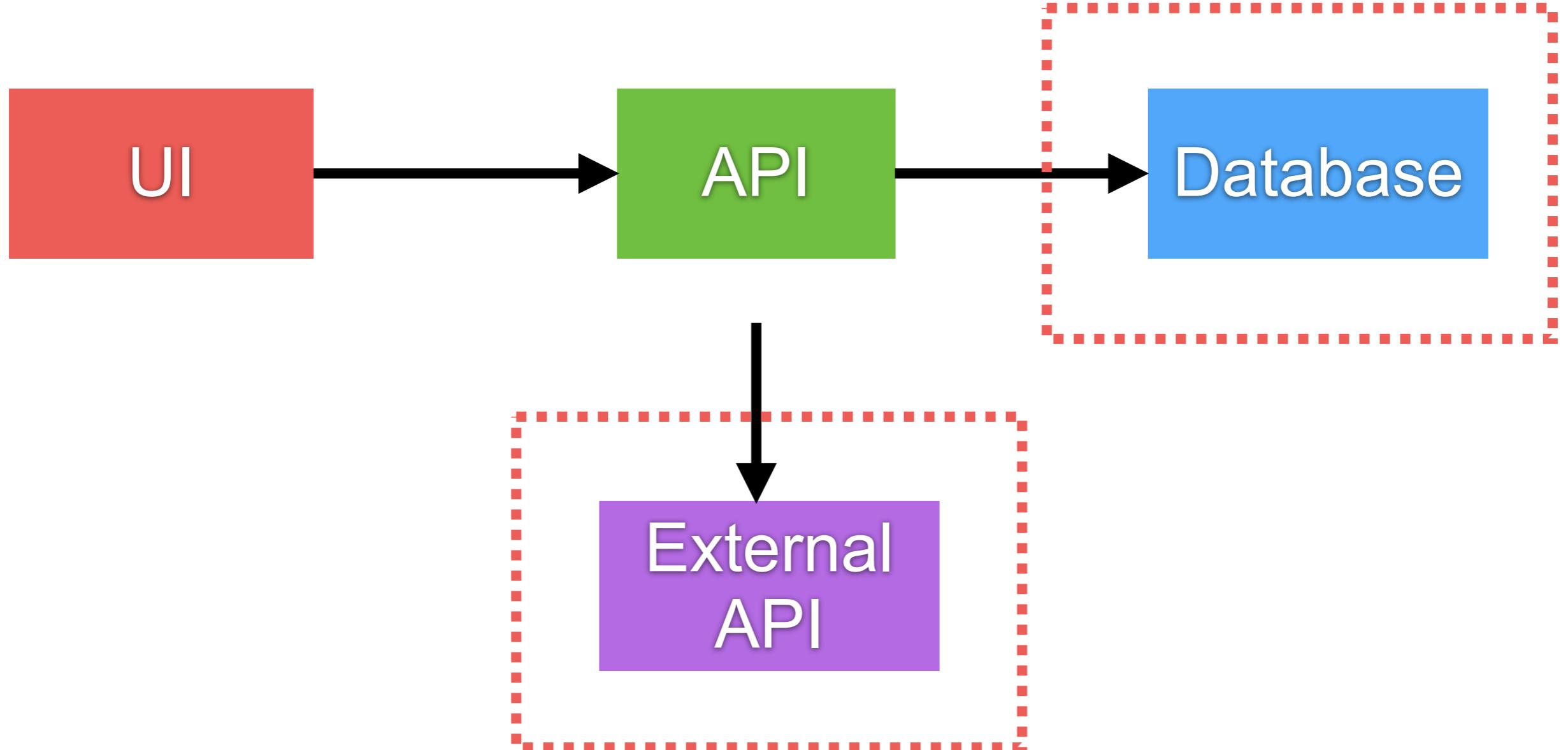
Manage dependencies ?



UI Perspective



API Perspective



Manage dependencies of UI

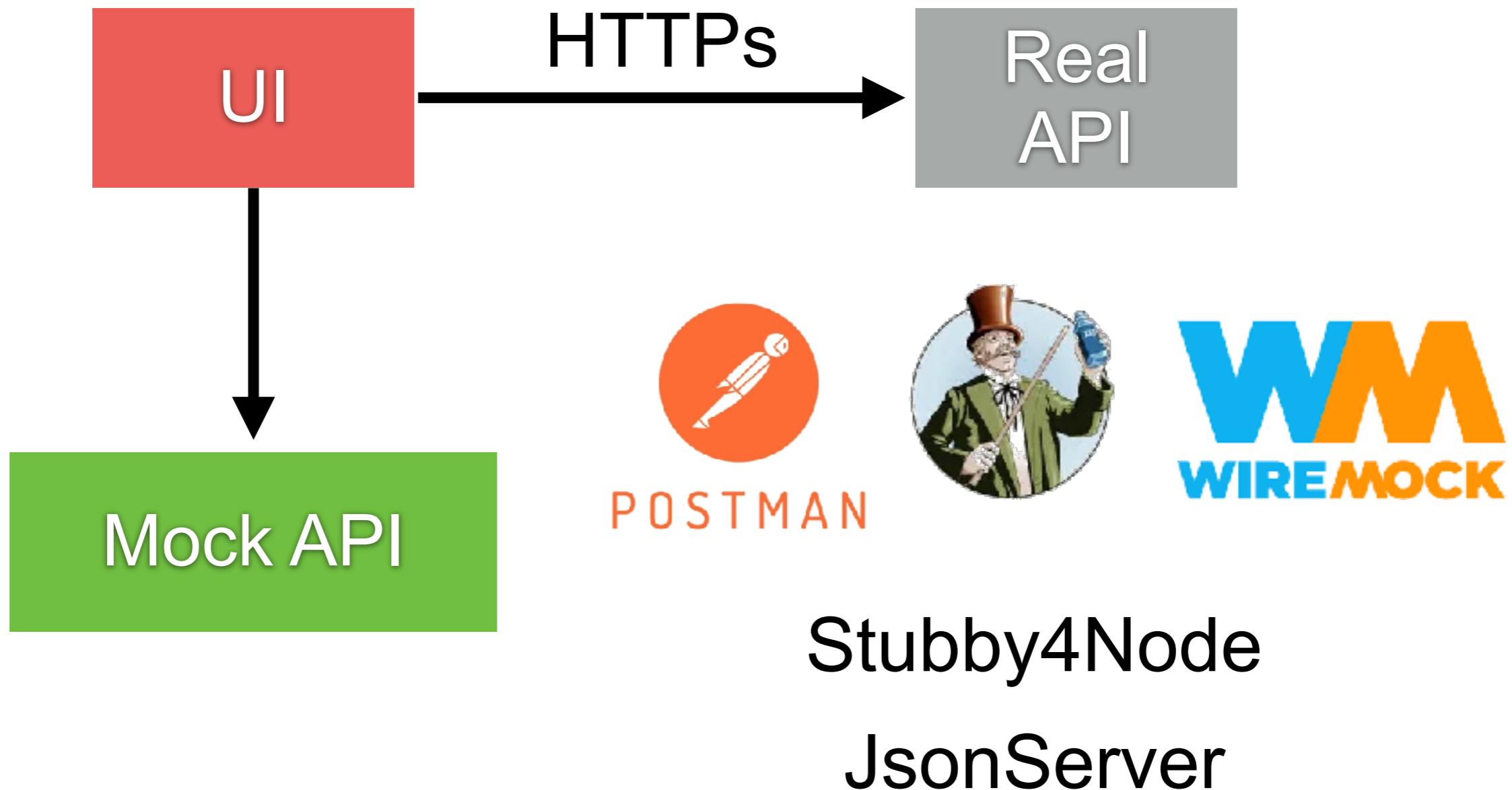


Manage dependencies of UI

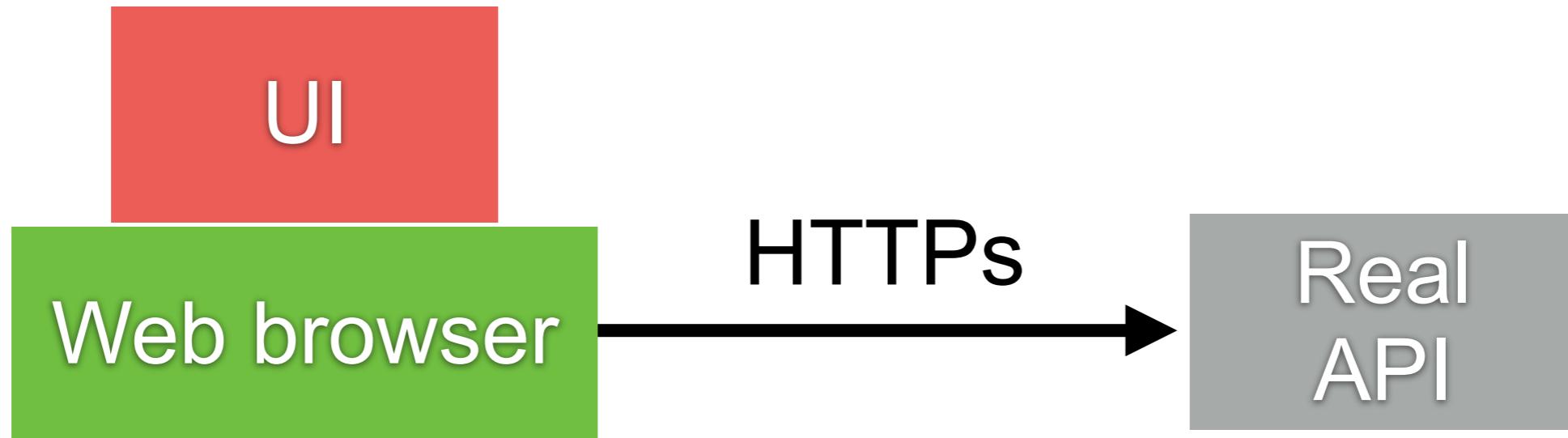
External mock API server
Interceptor in web browser



External Mock API Server



Intercept in Web browser

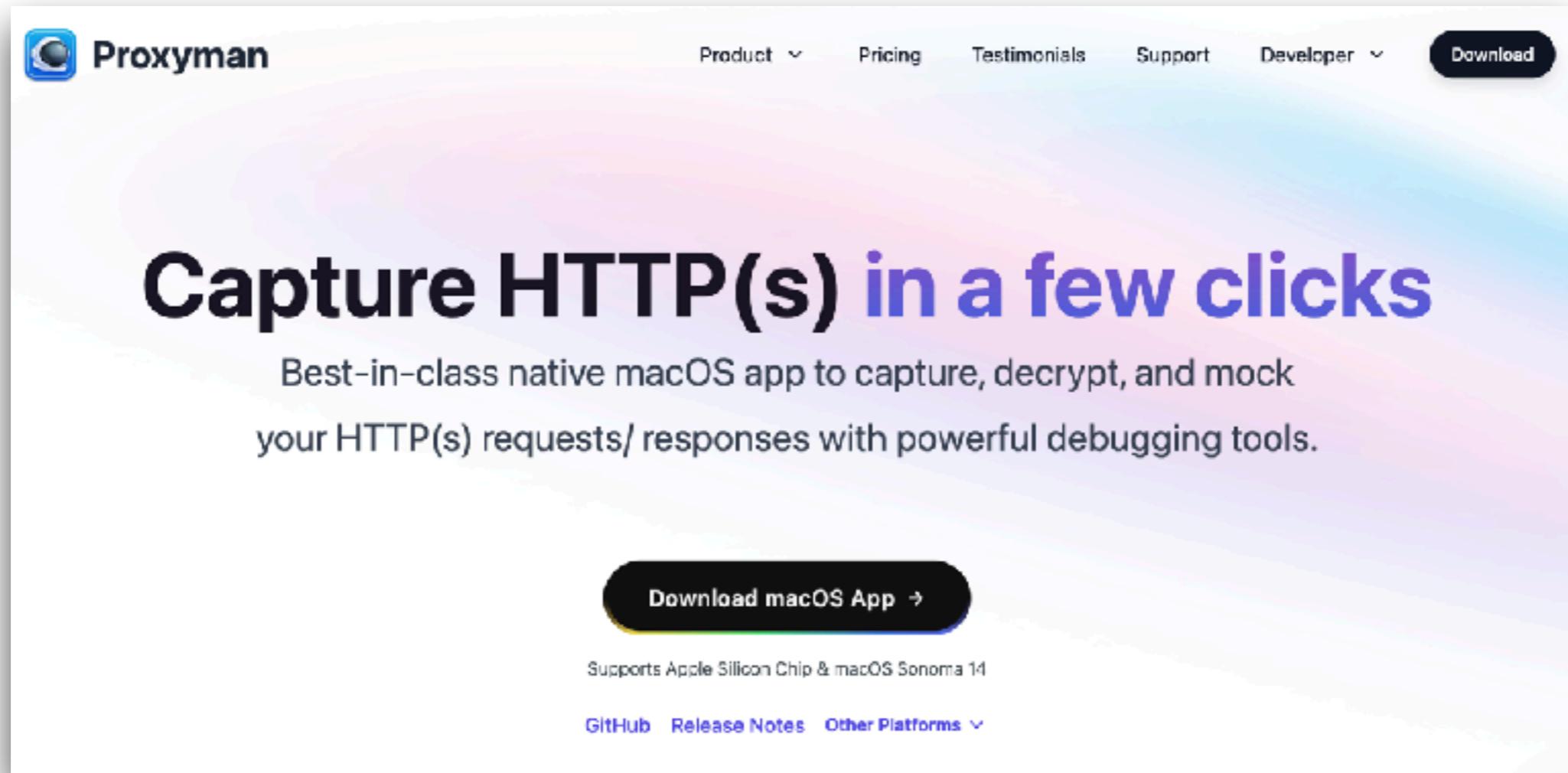


Playwright



Proxy Man

Capture HTTP request !!

A screenshot of the Proxyman website. The header features the "Proxyman" logo with a blue square icon containing a white circle and a checkmark. The main navigation menu includes "Product", "Pricing", "Testimonials", "Support", "Developer", and a "Download" button. The main title "Capture HTTP(s) in a few clicks" is displayed prominently in large, bold, black and blue text. Below it, a subtitle explains the product's purpose: "Best-in-class native macOS app to capture, decrypt, and mock your HTTP(s) requests/ responses with powerful debugging tools." A large "Download macOS App" button is centered below the subtitle. Smaller text at the bottom indicates compatibility with "Apple Silicon Chip & macOS Sonoma 14" and links to "GitHub", "Release Notes", and "Other Platforms".

Proxyman

Product Pricing Testimonials Support Developer Download

Capture HTTP(s) in a few clicks

Best-in-class native macOS app to capture, decrypt, and mock your HTTP(s) requests/ responses with powerful debugging tools.

Download macOS App →

Supports Apple Silicon Chip & macOS Sonoma 14

[GitHub](#) [Release Notes](#) [Other Platforms](#)

<https://proxym.io/>

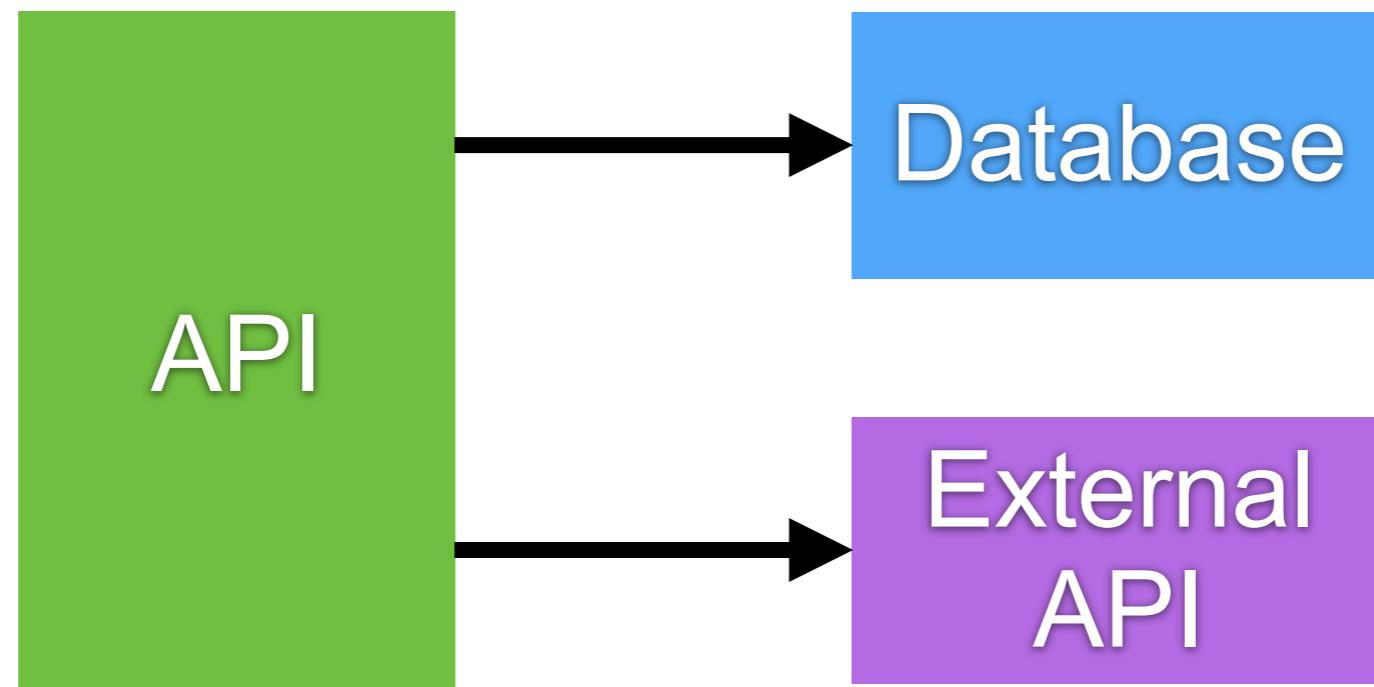


Manage dependencies of API



Manage dependencies of API

Database
External API



Database

In-memory database
Container-based
Real database

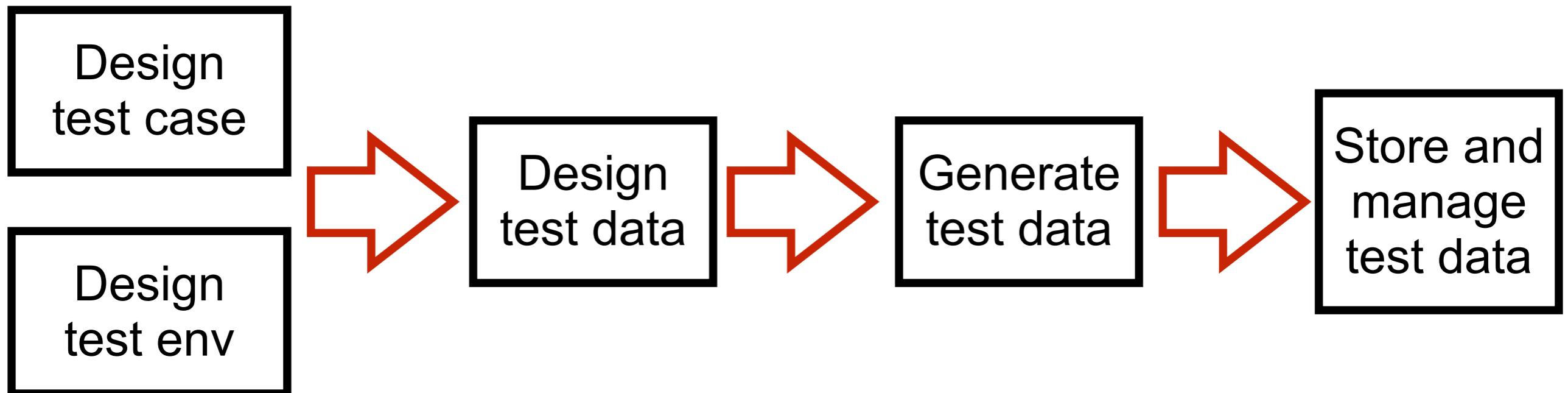
Tables

Data for test

Database migration tools



Design test data ?

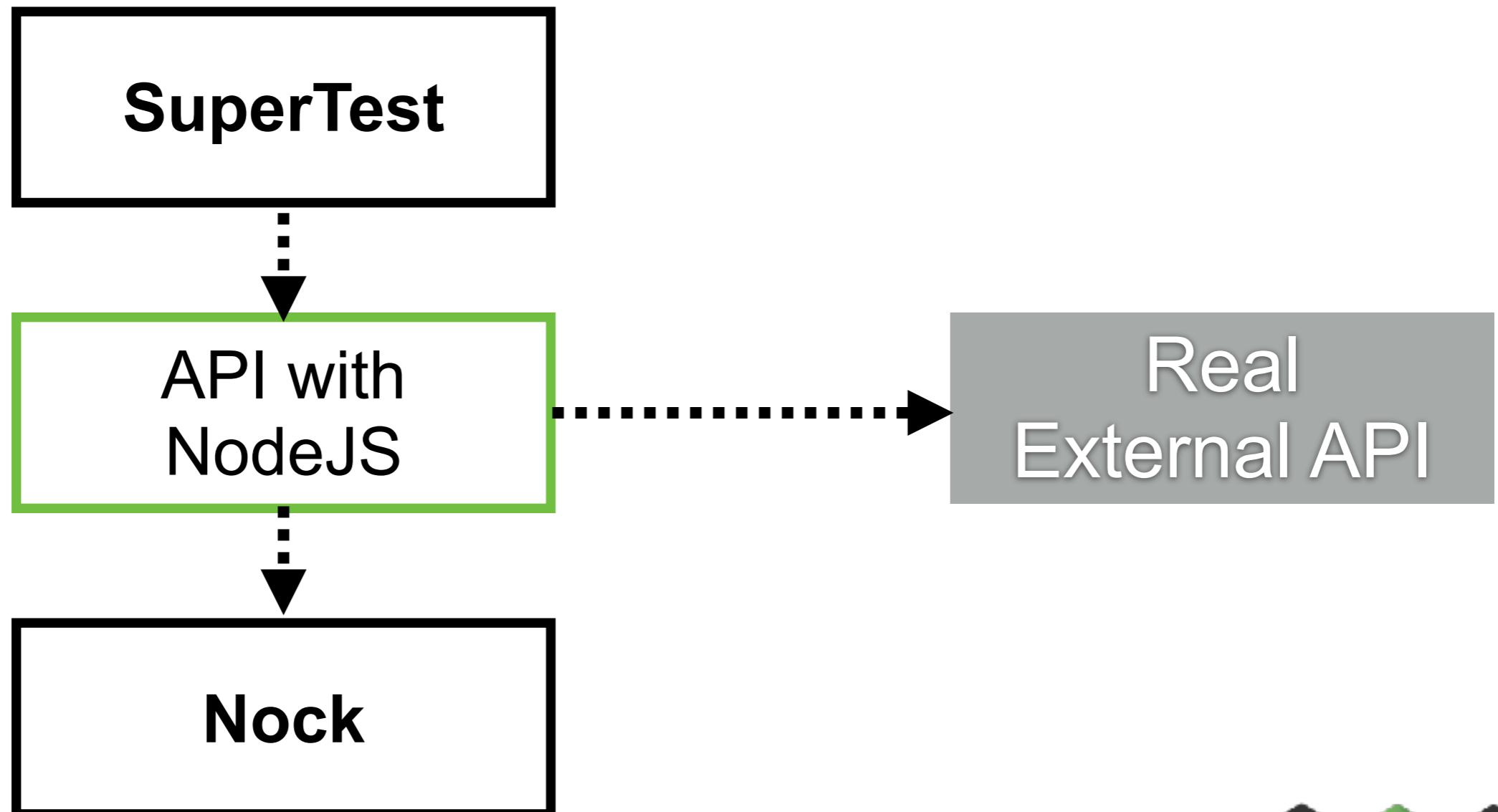


External API

**External mock API server
Internal mock API server
Use test double in code**



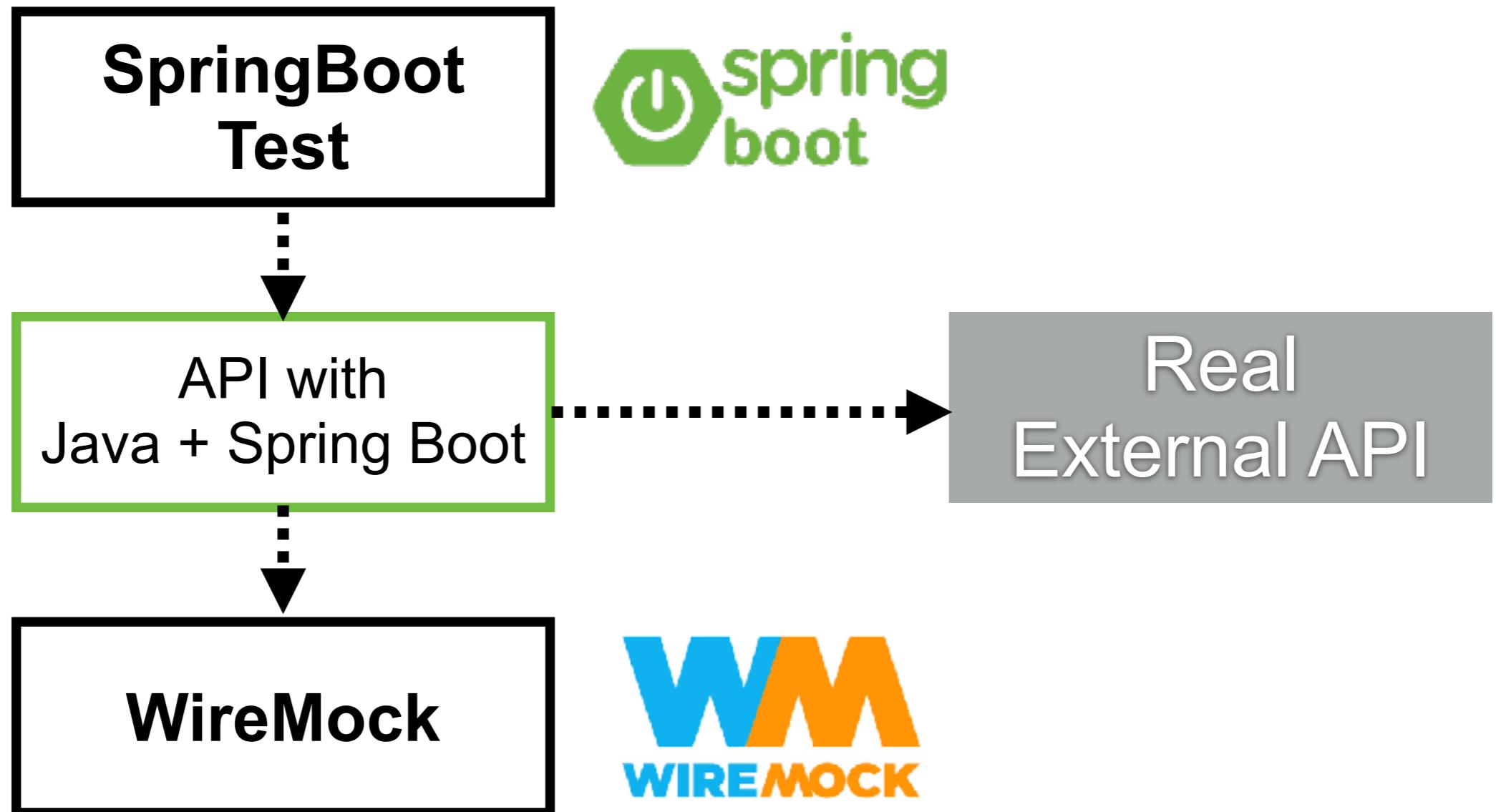
Internal Mock API with NodeJS



<https://www.npmjs.com/package/supertest>
<https://www.npmjs.com/package/nock>



Internal Mock API with Java

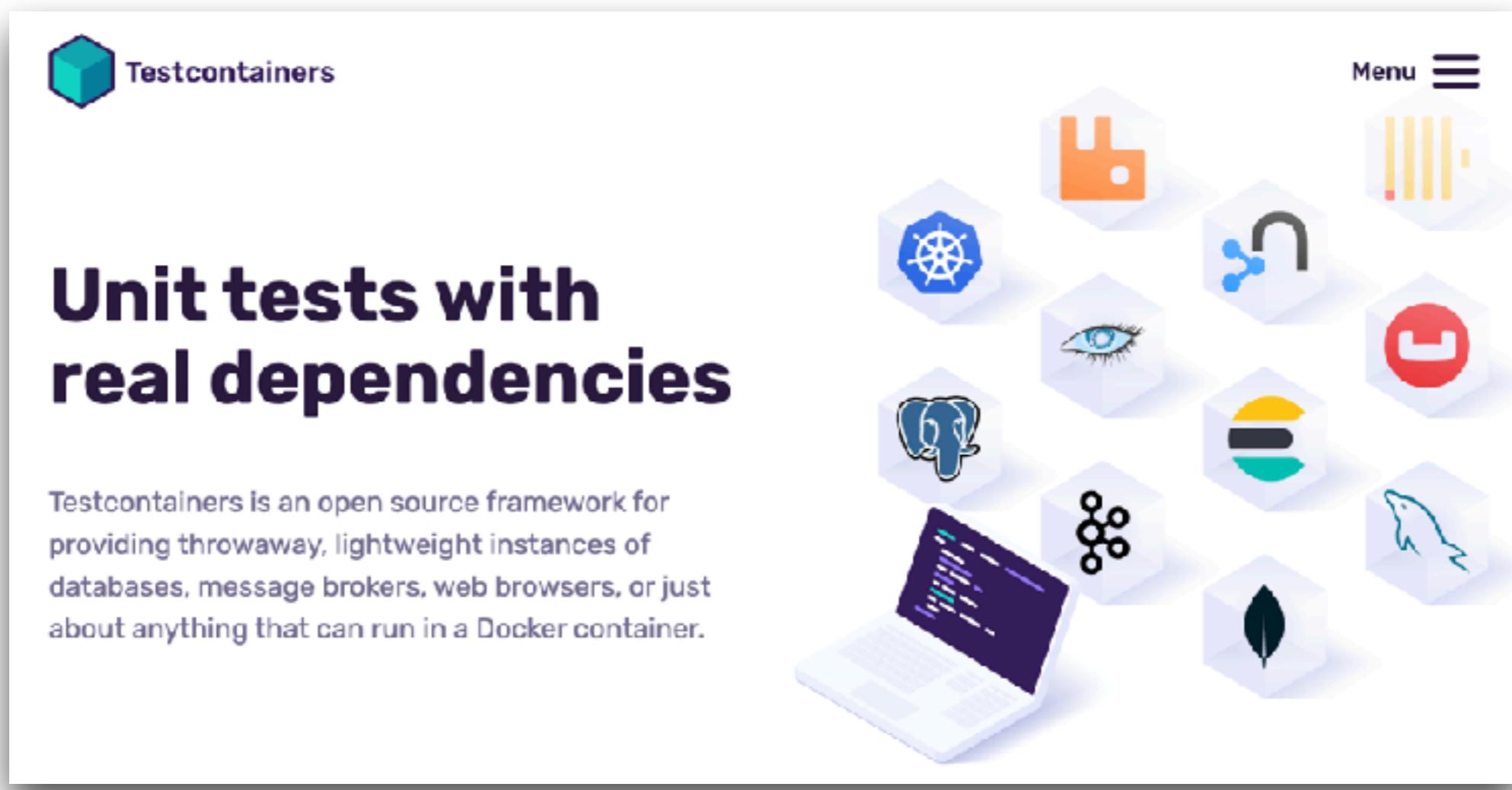


<https://docs.spring.io/spring-cloud-contract/reference/project-features-wiremock.html>



Test Container

Testing with real dependencies



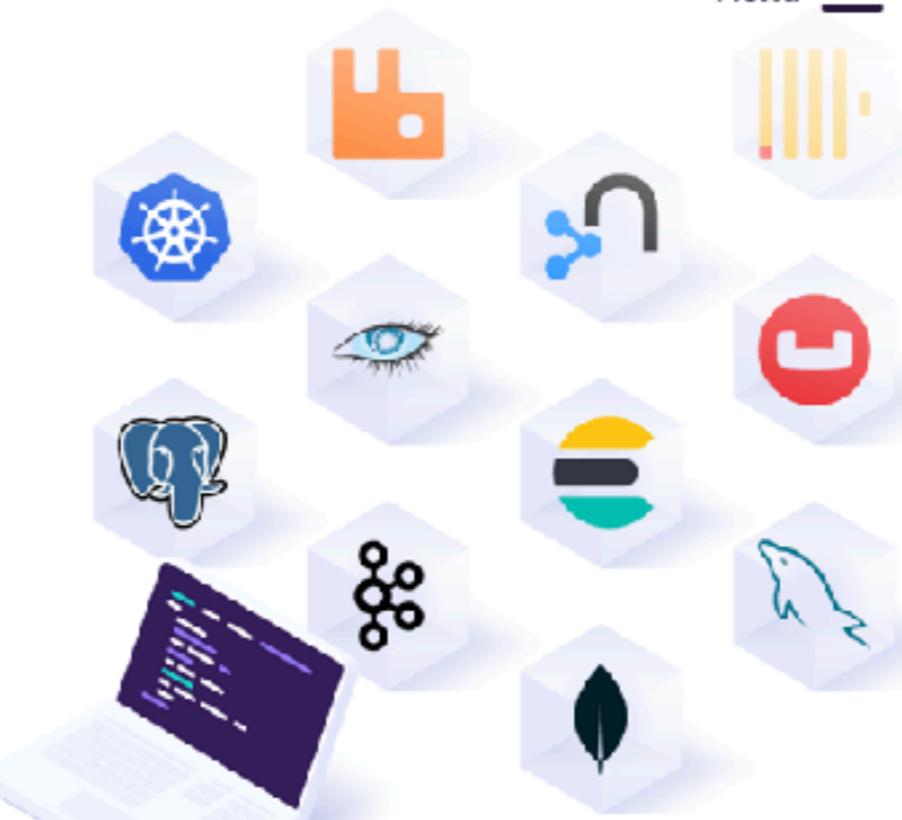
The screenshot shows the homepage of the Testcontainers website. At the top left is the Testcontainers logo (a teal hexagon icon). At the top right is a "Menu" button with three horizontal lines. The main title "Unit tests with real dependencies" is displayed prominently in large, bold, dark blue text. Below the title is a description: "Testcontainers is an open source framework for providing throwaway, lightweight instances of databases, message brokers, web browsers, or just about anything that can run in a Docker container." To the right of the text is a graphic featuring a laptop displaying code, surrounded by various icons representing different services and tools used in testing, such as a database, a message broker, a browser, and monitoring.

Testcontainers

Unit tests with real dependencies

Testcontainers is an open source framework for providing throwaway, lightweight instances of databases, message brokers, web browsers, or just about anything that can run in a Docker container.

Menu



A grid of 15 hexagonal icons representing various services and tools:

- Blue hexagon: Helm
- Orange hexagon: Jenkins
- Yellow hexagon: Grafana
- Red hexagon: Concourse
- Green hexagon: PostgreSQL
- Blue hexagon: MySQL
- Blue hexagon: MongoDB
- Blue hexagon: Redis
- Blue hexagon: Elasticsearch
- Blue hexagon: RabbitMQ
- Blue hexagon: Kafka
- Blue hexagon: Docker
- Blue hexagon: Docker Compose
- Blue hexagon: Dockerfile
- Blue hexagon: Docker Hub

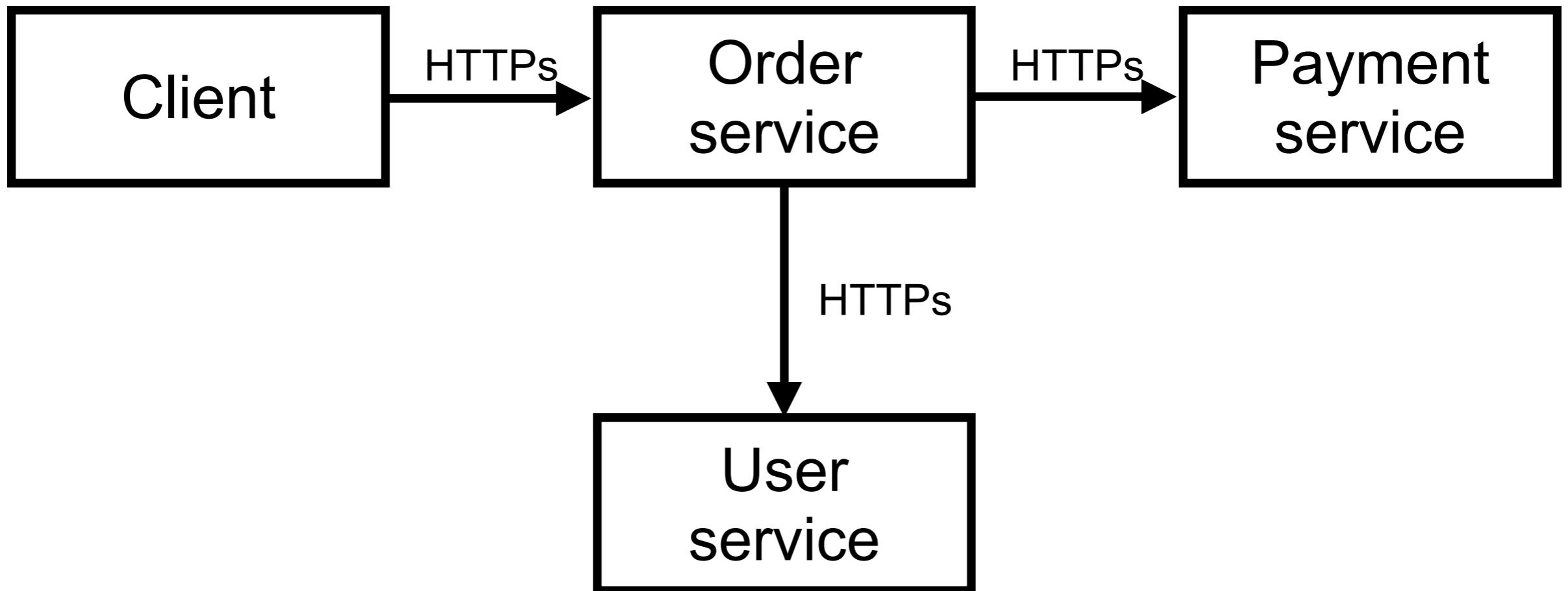
<https://testcontainers.com/>



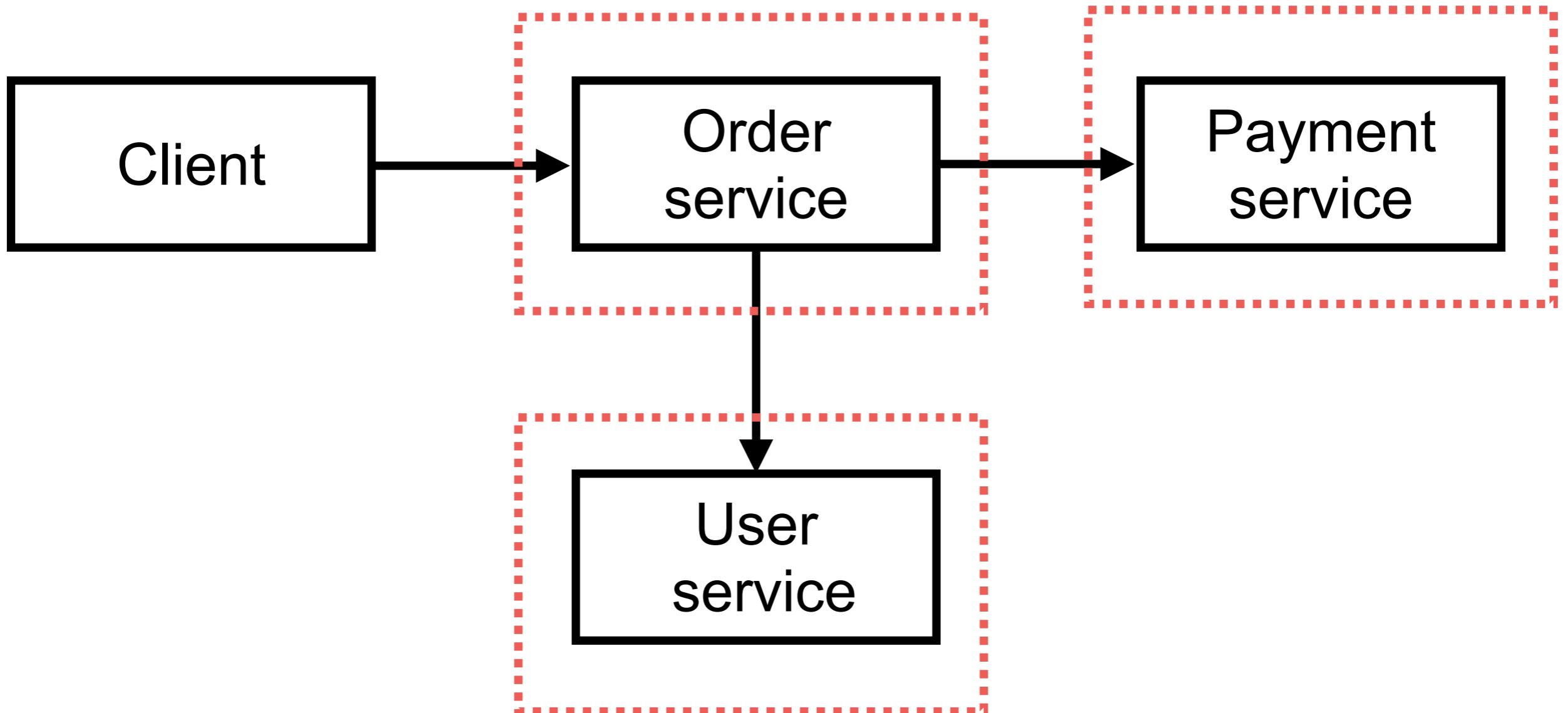
More Testing with APIs



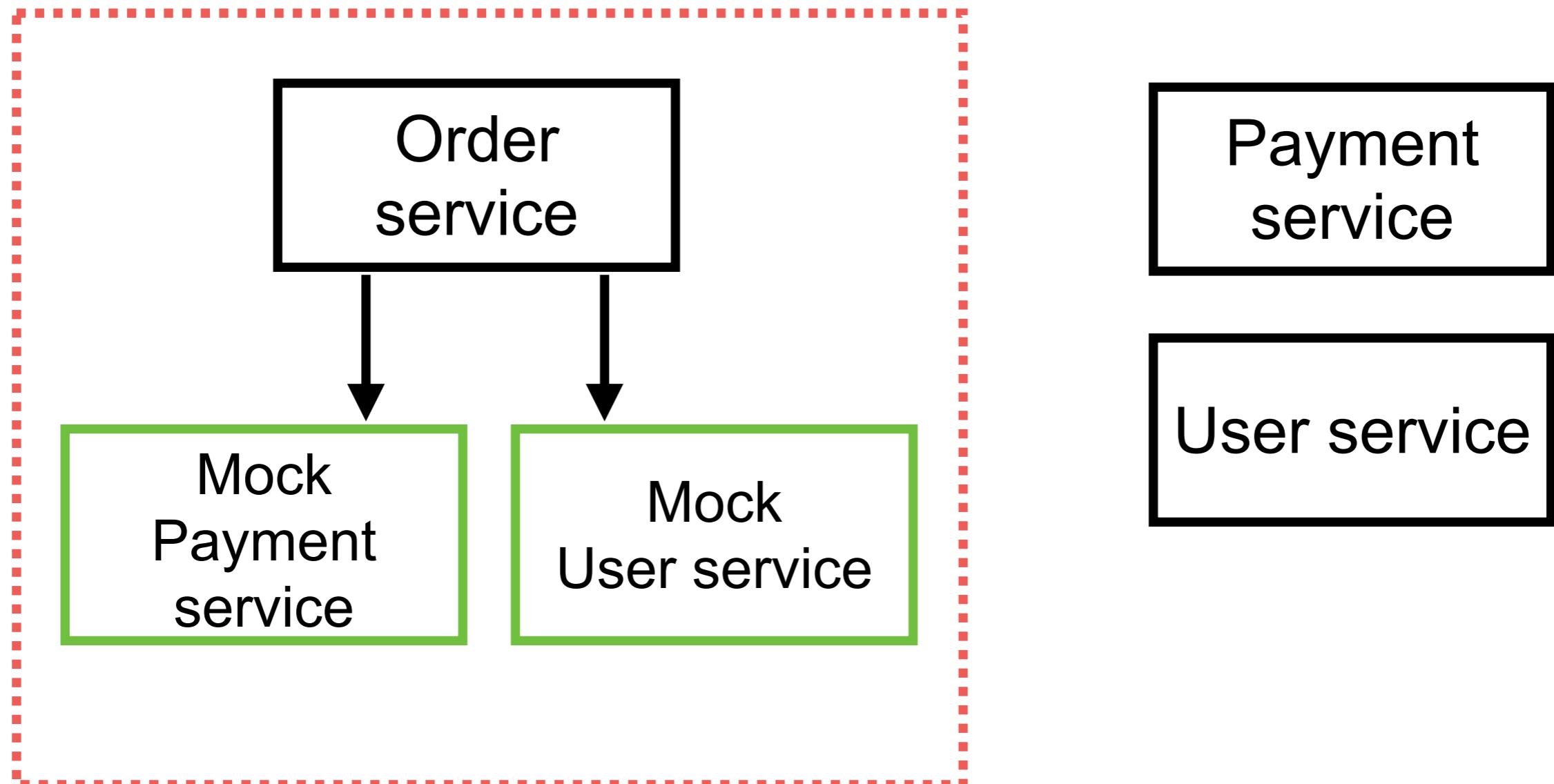
Service-to-Service Testing ?



Component Testing



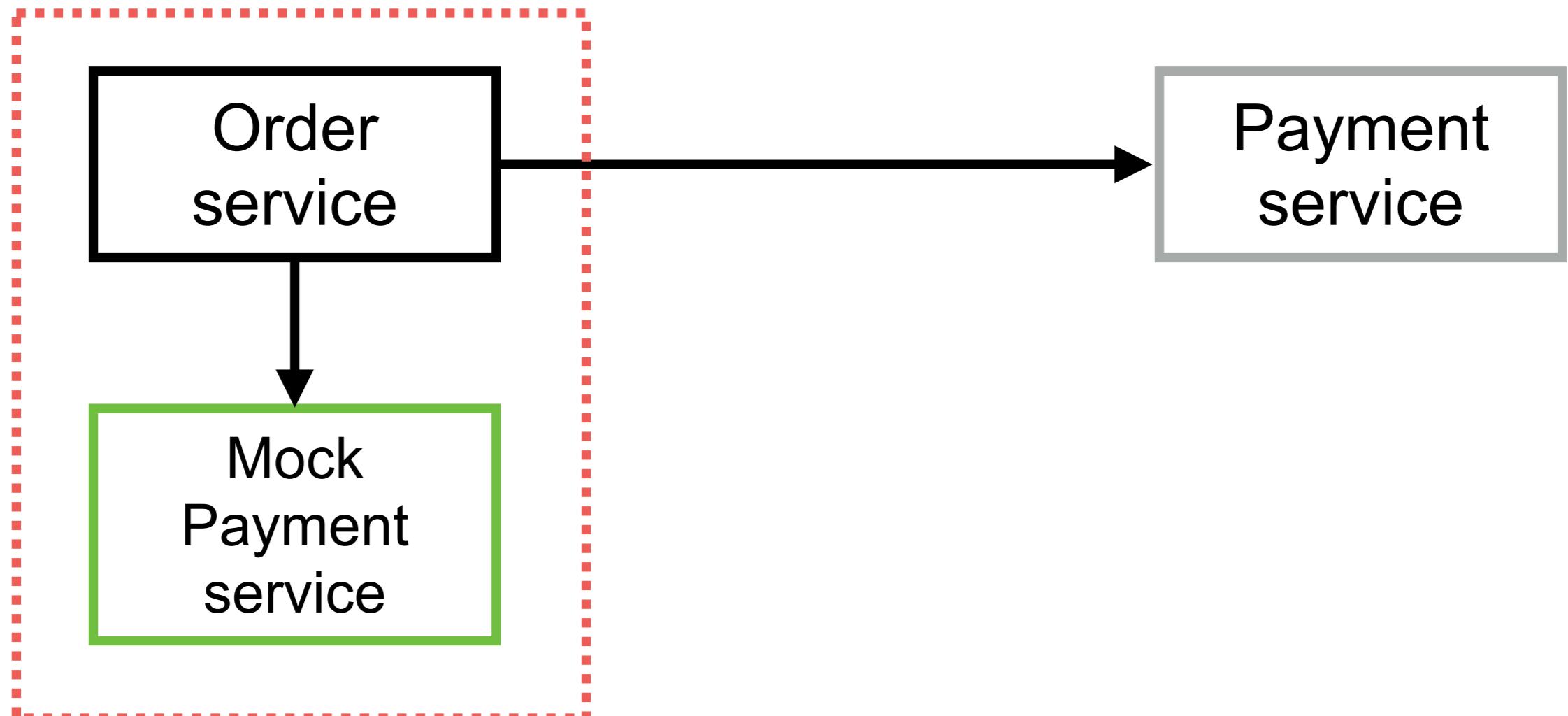
Component Testing



Workshop



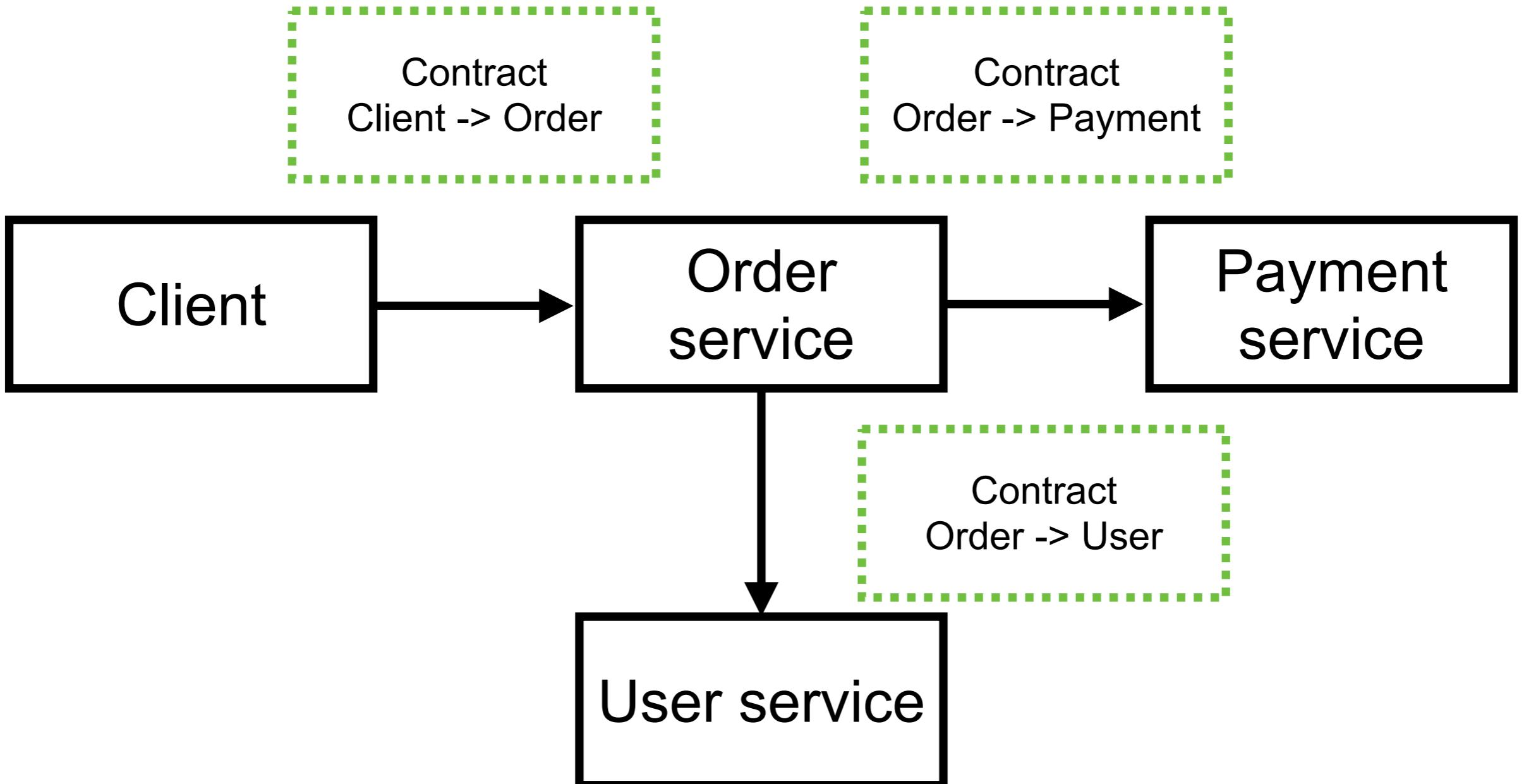
Component Testing workshop



Contract Testing

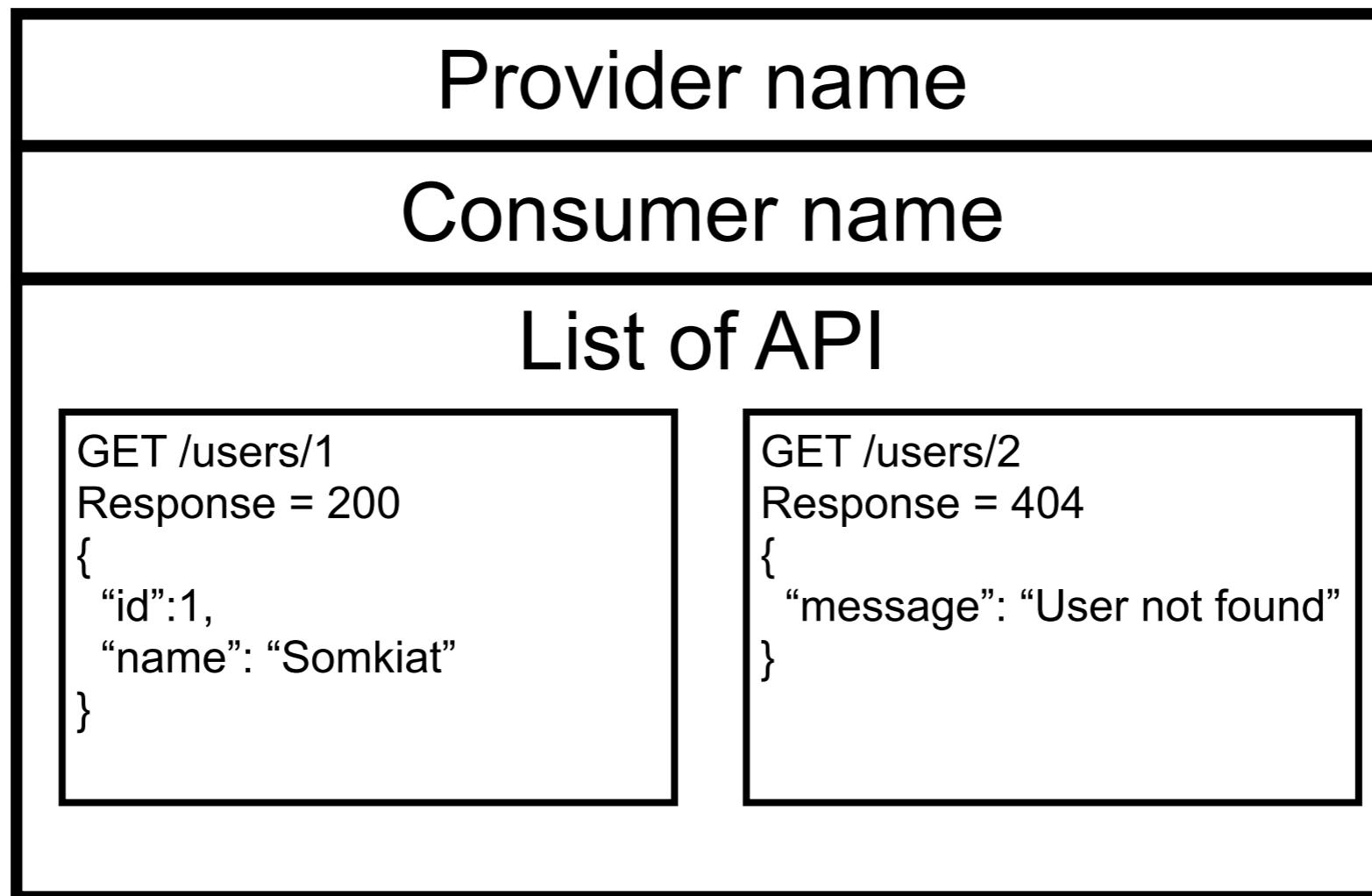


Contract Testing



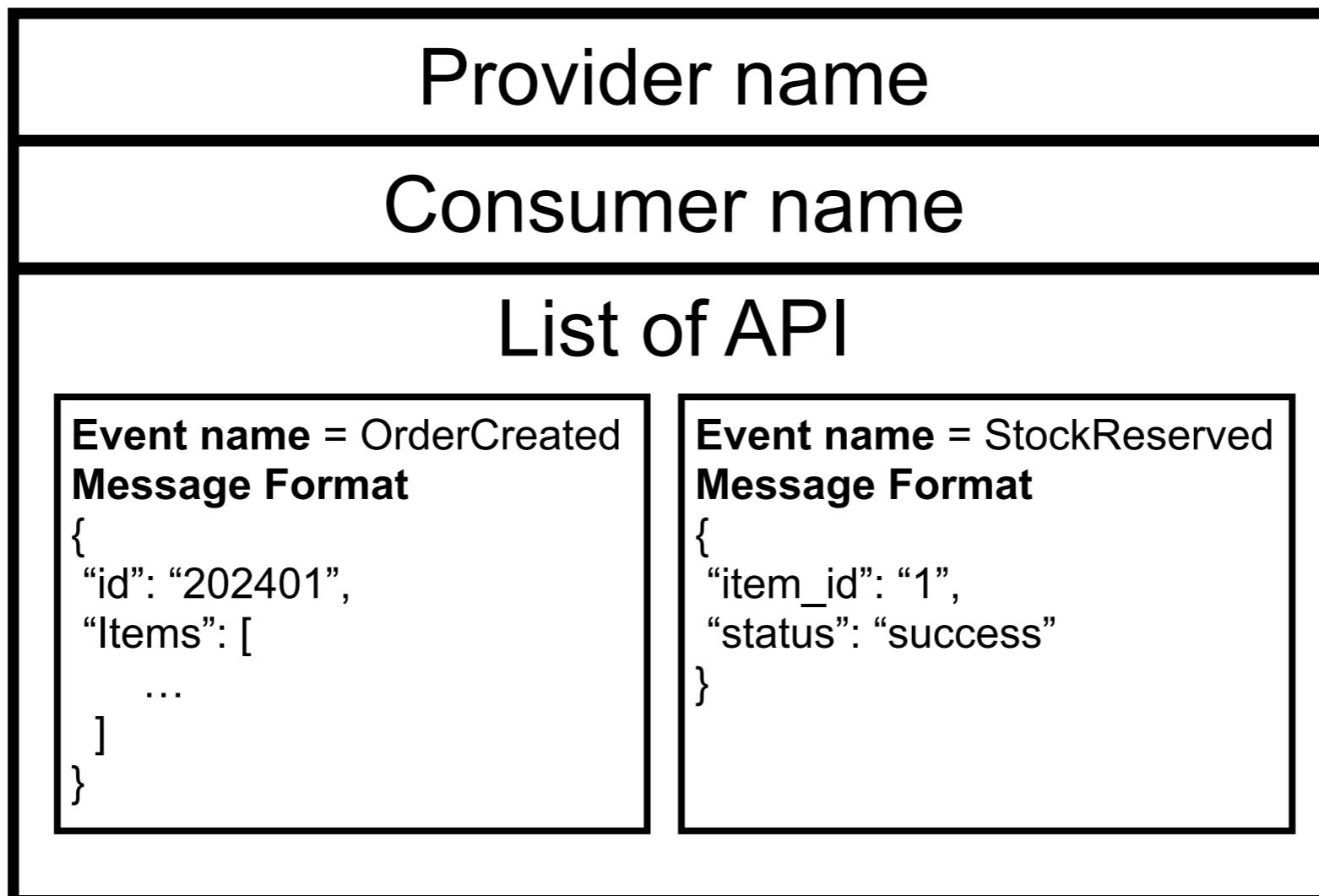
Contract Format

REST API



Contract Format

Messaging (Asynchronous)

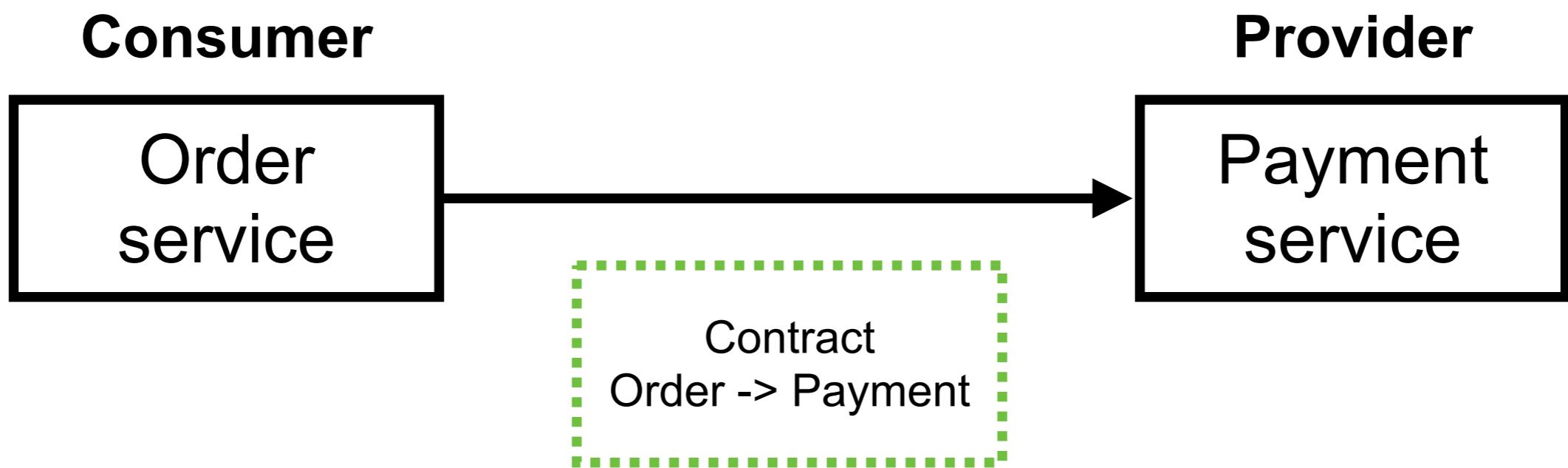


Process of contract testing



Order to Payment

1. Create contract between services



Order to Payment

2. Publish contract

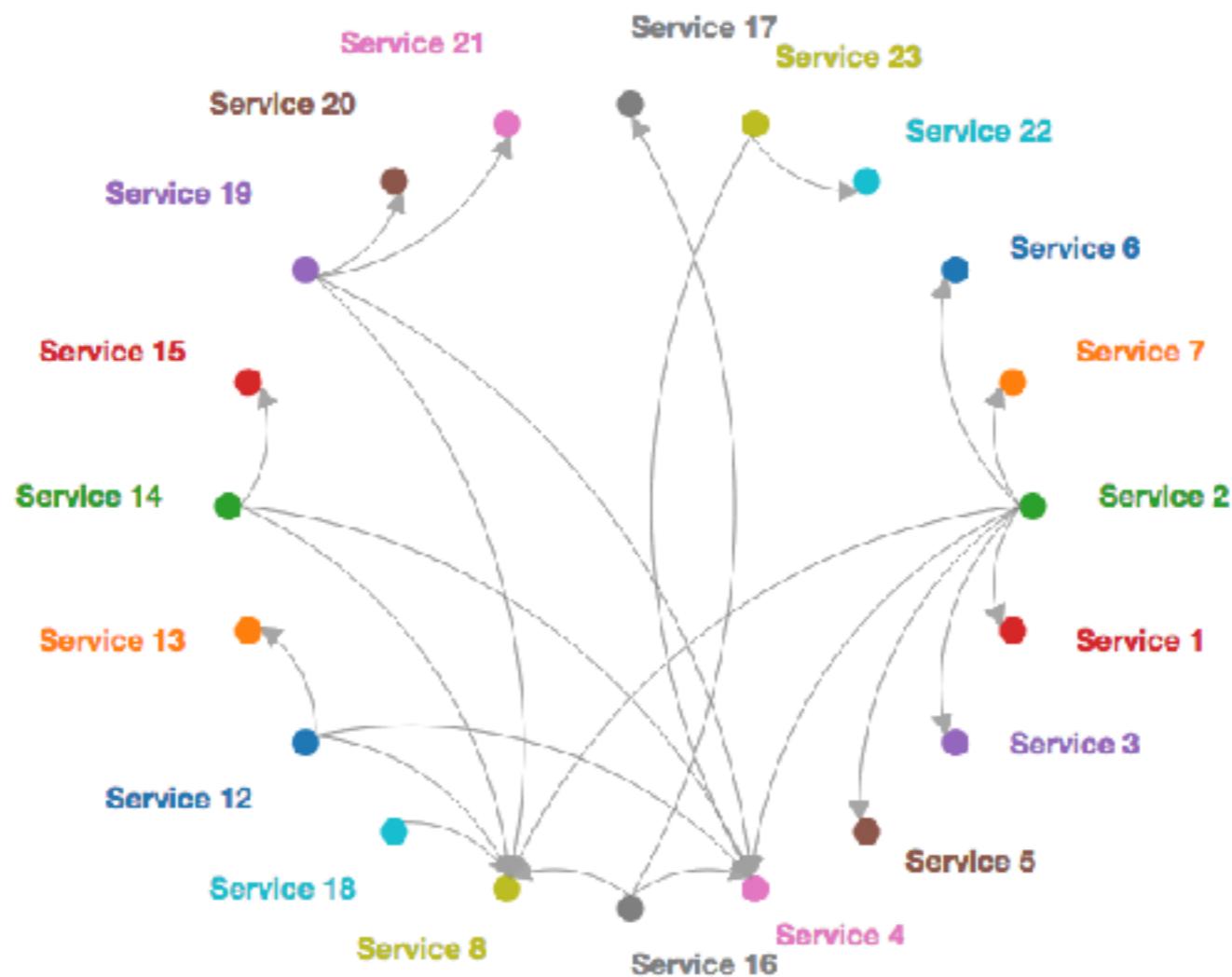
Pacts

Consumer ↓↑	Provider ↓↑	Latest pact published	Last verified
Foo	Animals	2 minutes ago	2 days ago
Foo	Bar	7 days ago	15 days ago ▲
Foo	Hello World App	1 day ago	
Foo	Wiffles	less than a minute ago	7 days ago
Some other app	A service	26 days ago	less than a minute ago
The Android App	The back end	less than a minute ago	



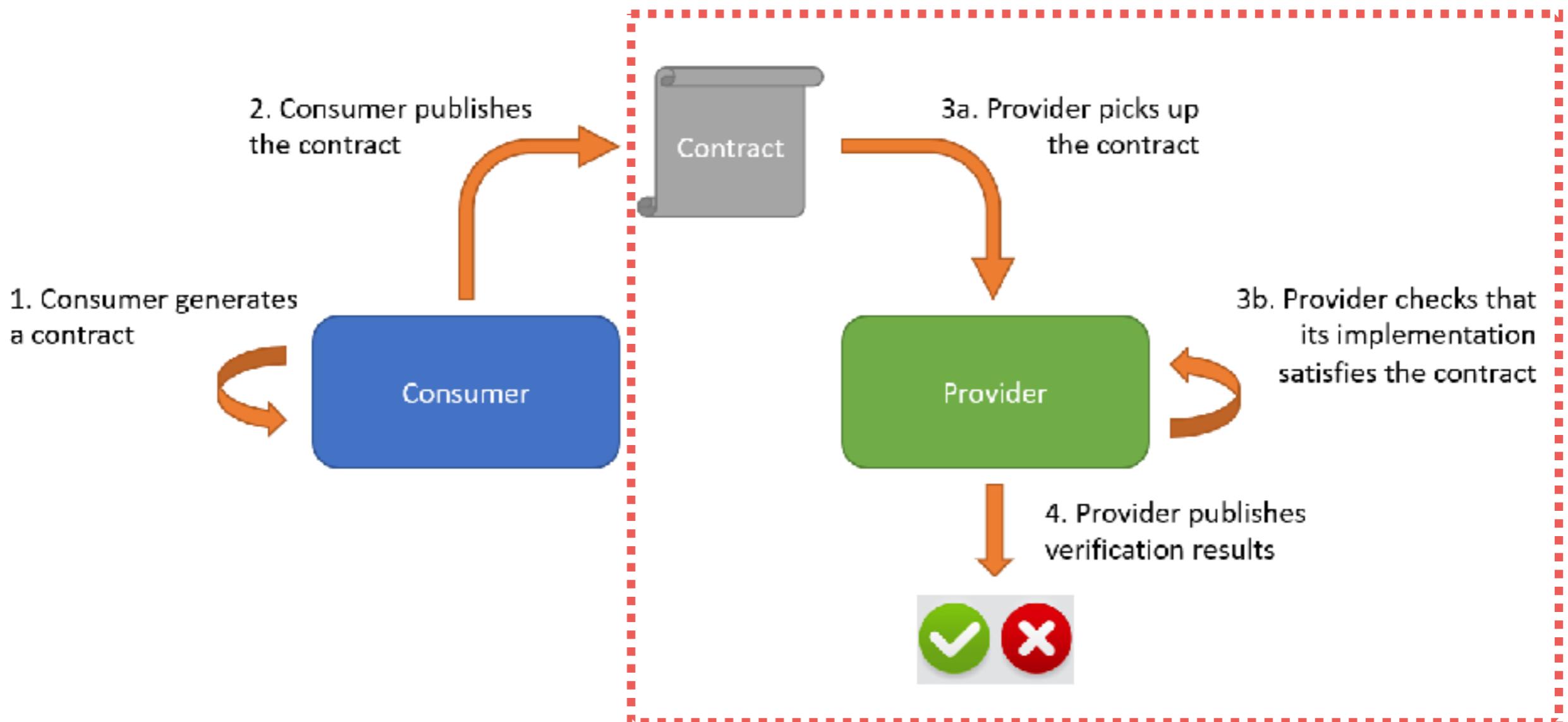
Order to Payment

2. Publish contract



Order to Payment

3. Provider pickup contracts to verify

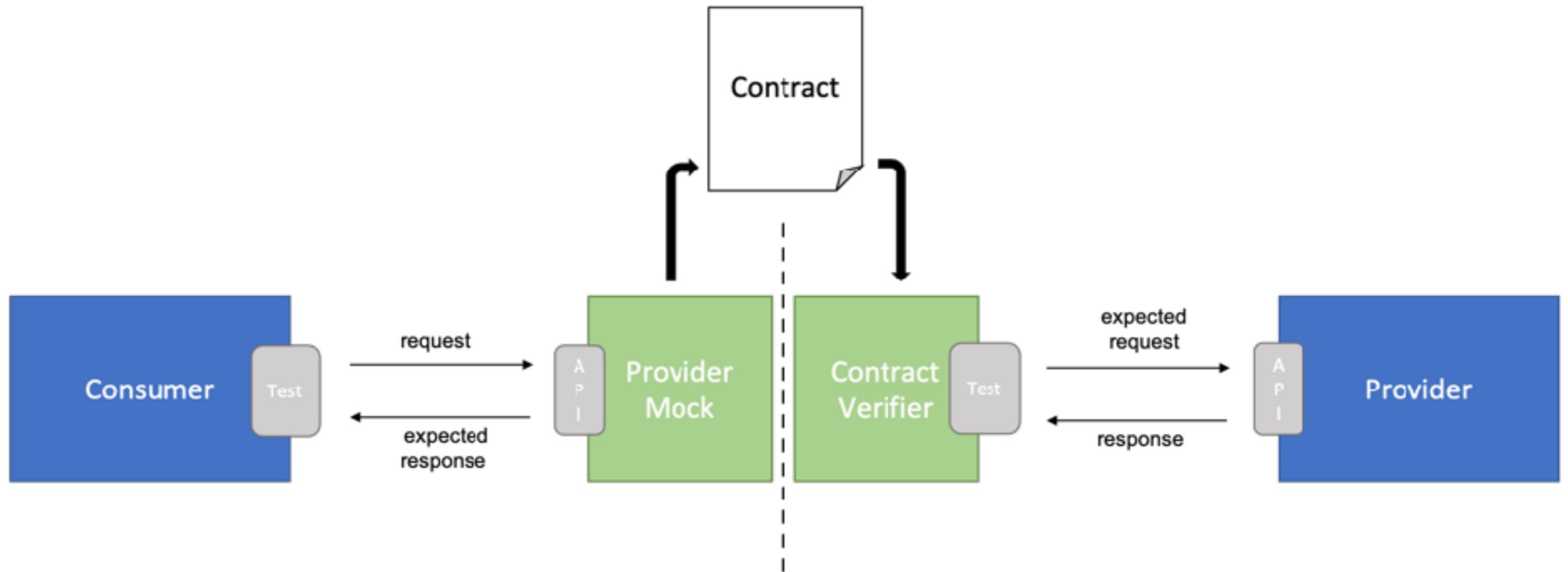


Order to Payment

4. Consumer try to mock api server from contract



Contract Testing



<https://microsoft.github.io/code-with-engineering-playbook/automated-testing/cdc-testing/>



Contract Testing Tools

Pact

Spring Cloud
Contract

OpenAPI/
Swagger

Karate DSL

Postman

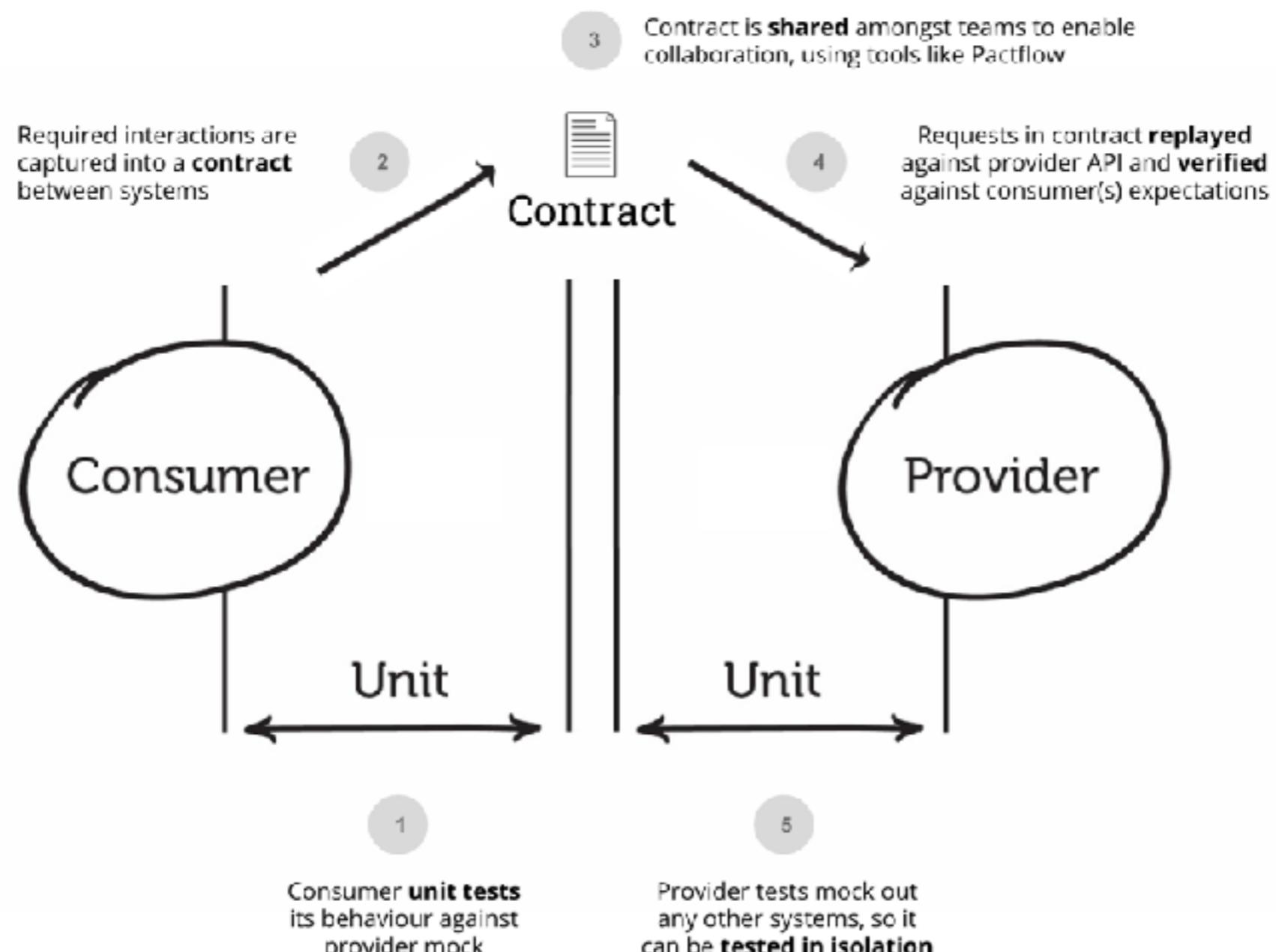


Workshop

PACTS



Workshop with Pact



<https://docs.pact.io/>



Workshop with Pact

1

Create Contract
from consumer

2

Install and
config broker

3

Publish contract
to broker

4

Provider verify
from contract

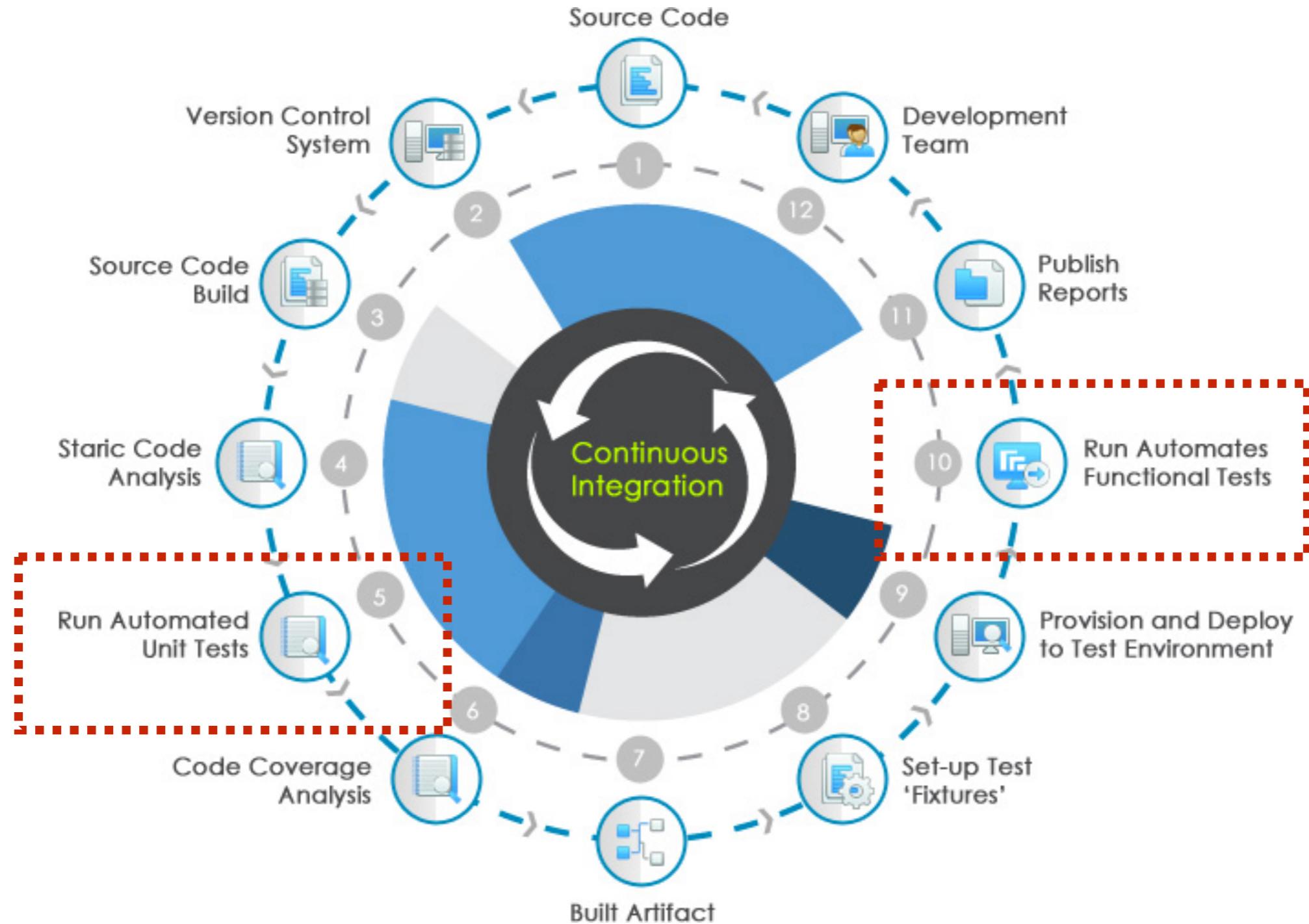
<https://docs.pact.io/>



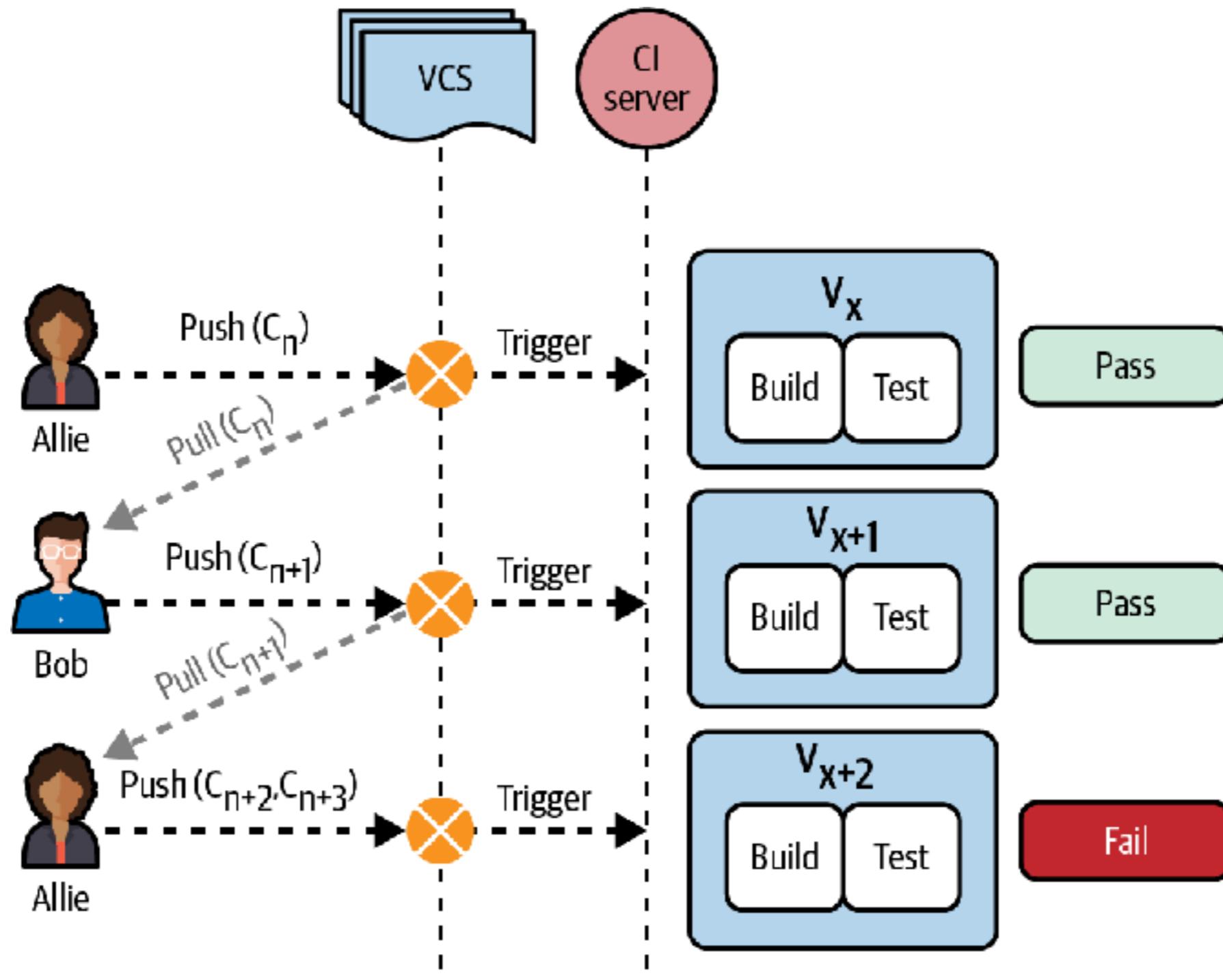
Continuous Integration



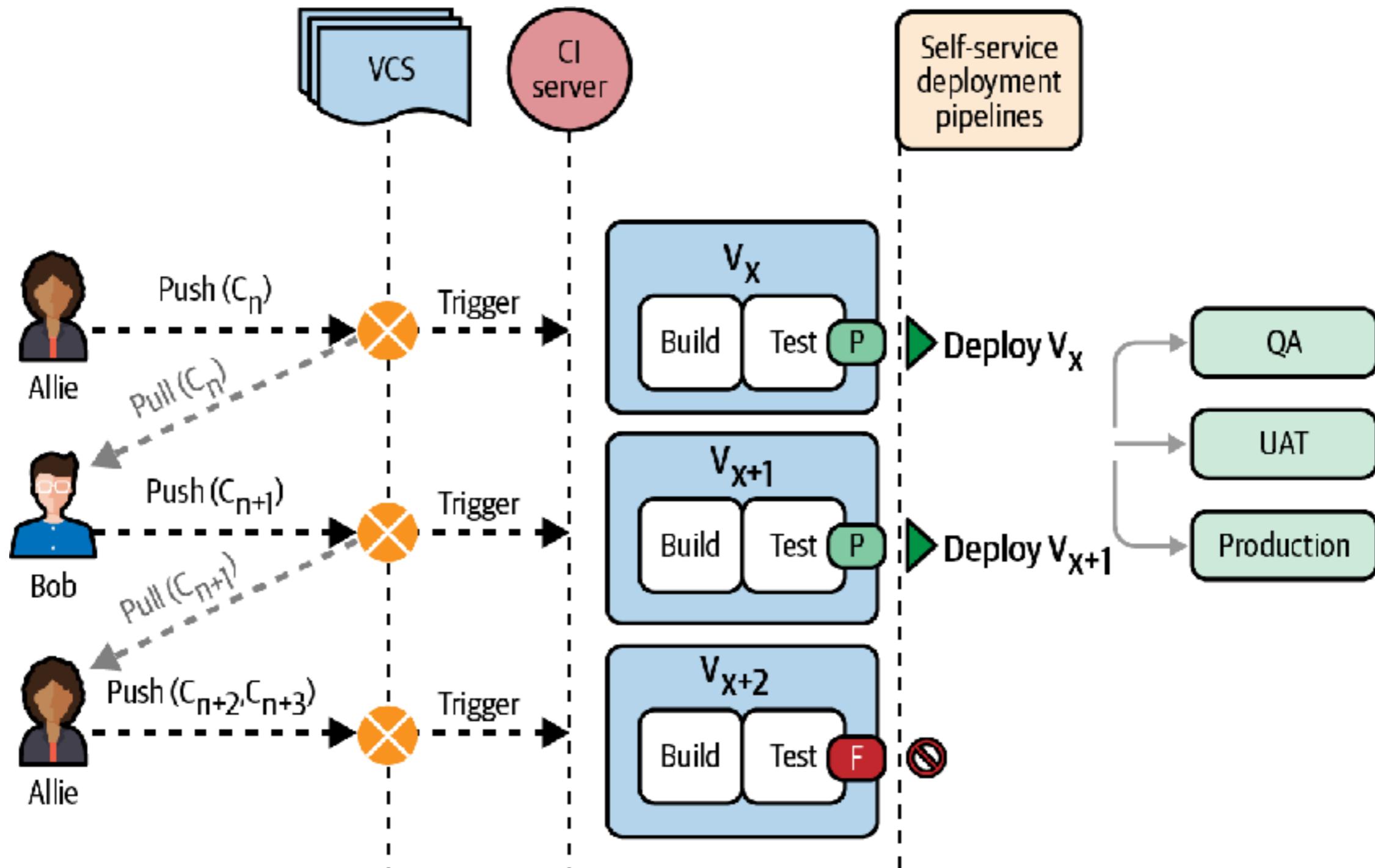
Continuous integration (CI)



Continuous Integration (CI)



Continuous Delivery (CD)



Principle of CI/CD

Do frequent code commits

Always commit self-tested code

Always test passed before move to next task

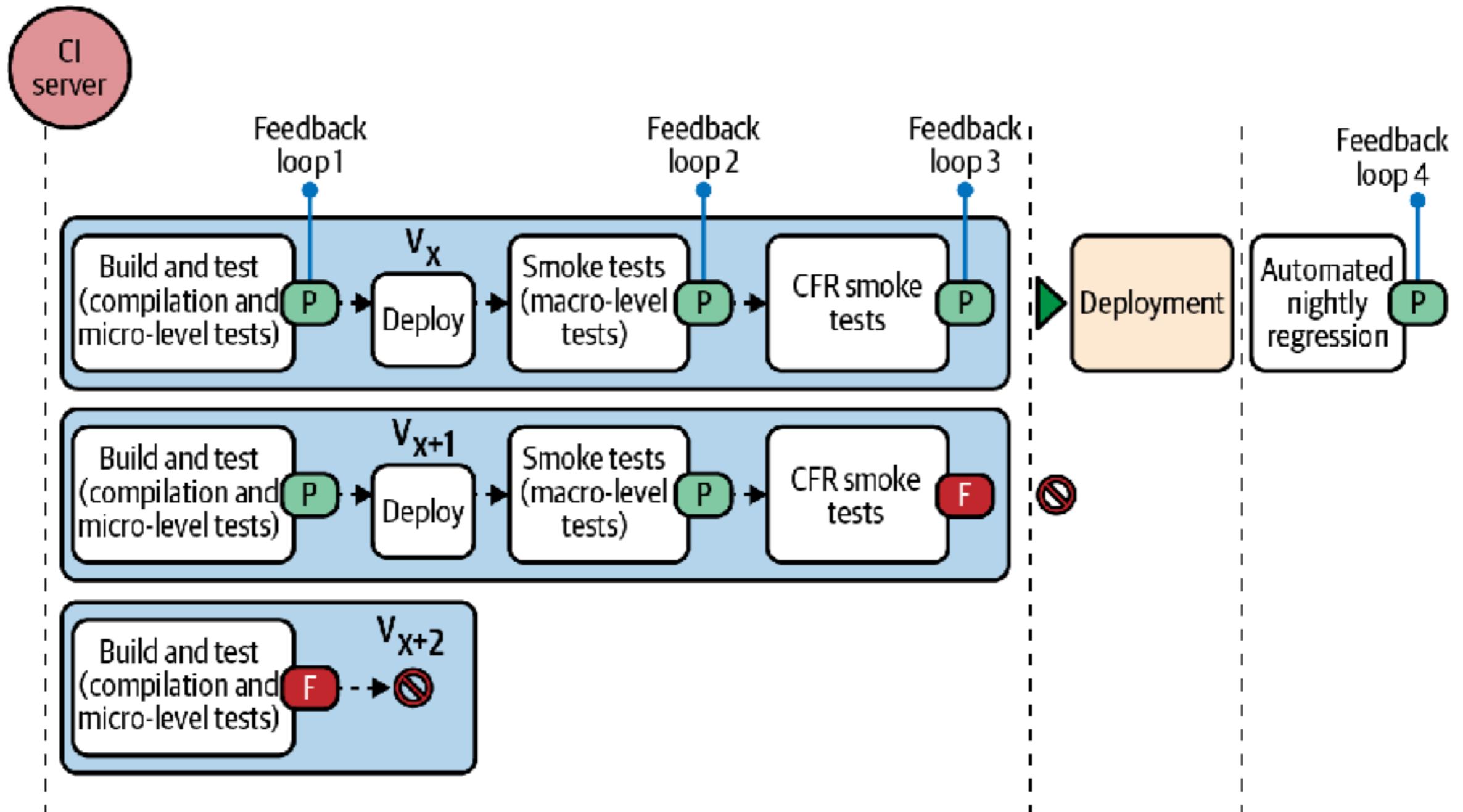
Don't ignore/comment out the failing tests

Don't push to the broken build

Take ownership of all failures



Continuous Testing



Continuous Testing



**“Everyone is responsible for assuring their commits
are ready for deployment”**



4 key metrics to measure

Delivery process

Lead time

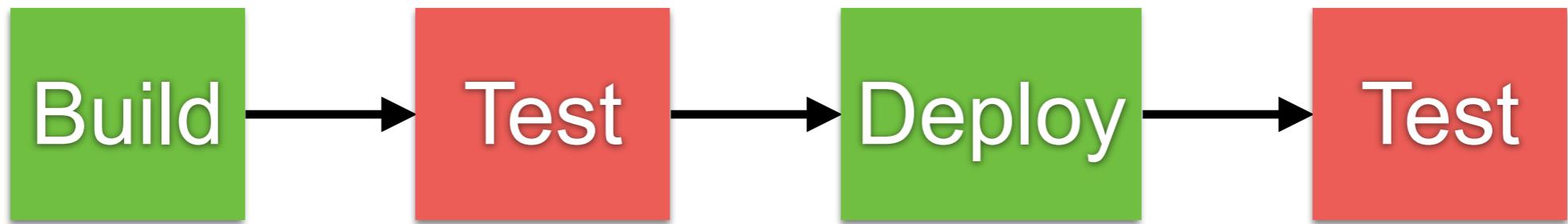
Deployment
frequency

Mean time to
recover

Change fail
percentage



Design your pipeline



Design your pipeline/process

Code
change

Compile

Static code
analysis

Unit test

Code
coverage

Build

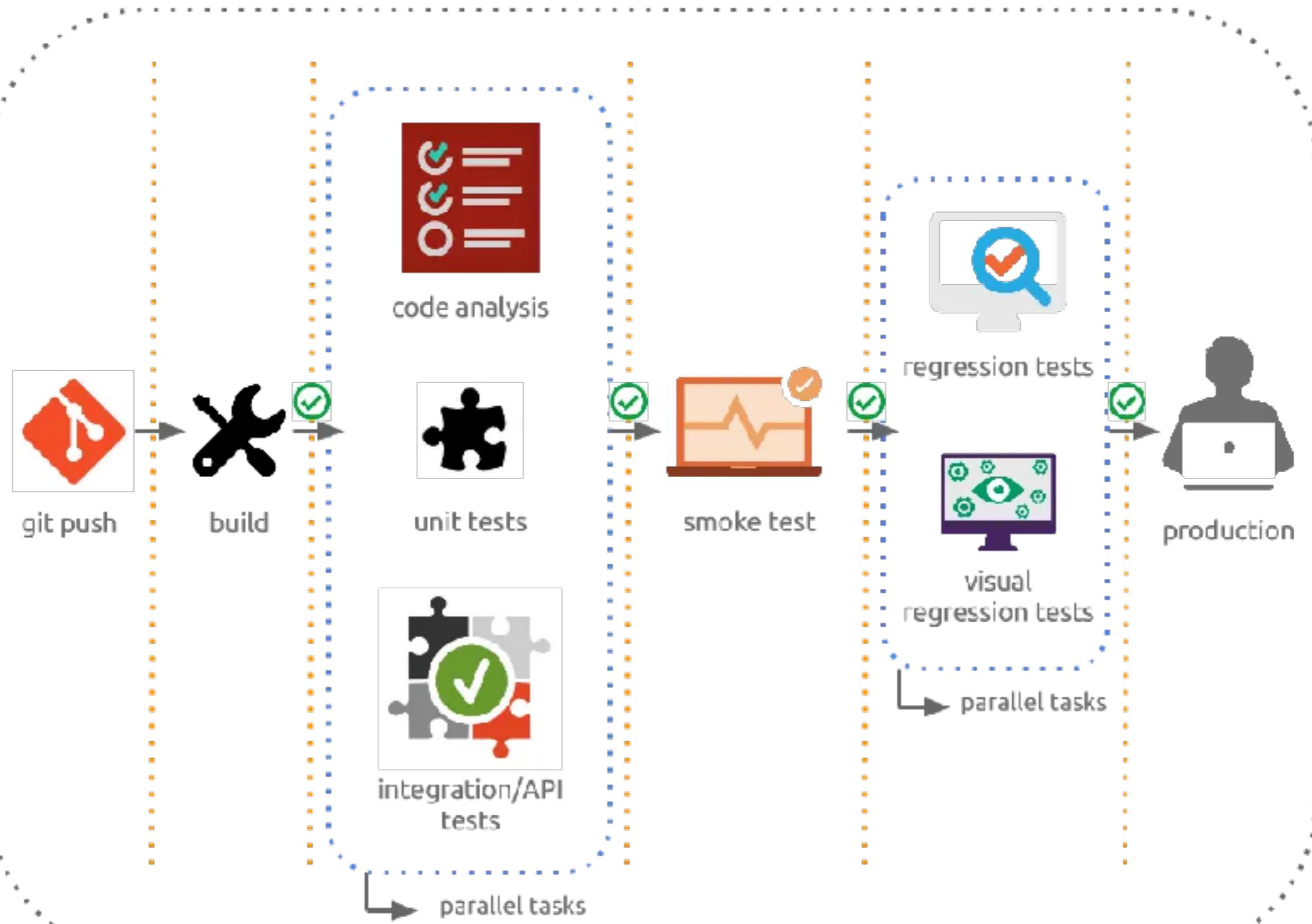
Setup test
data

Deploy

Functional
testing



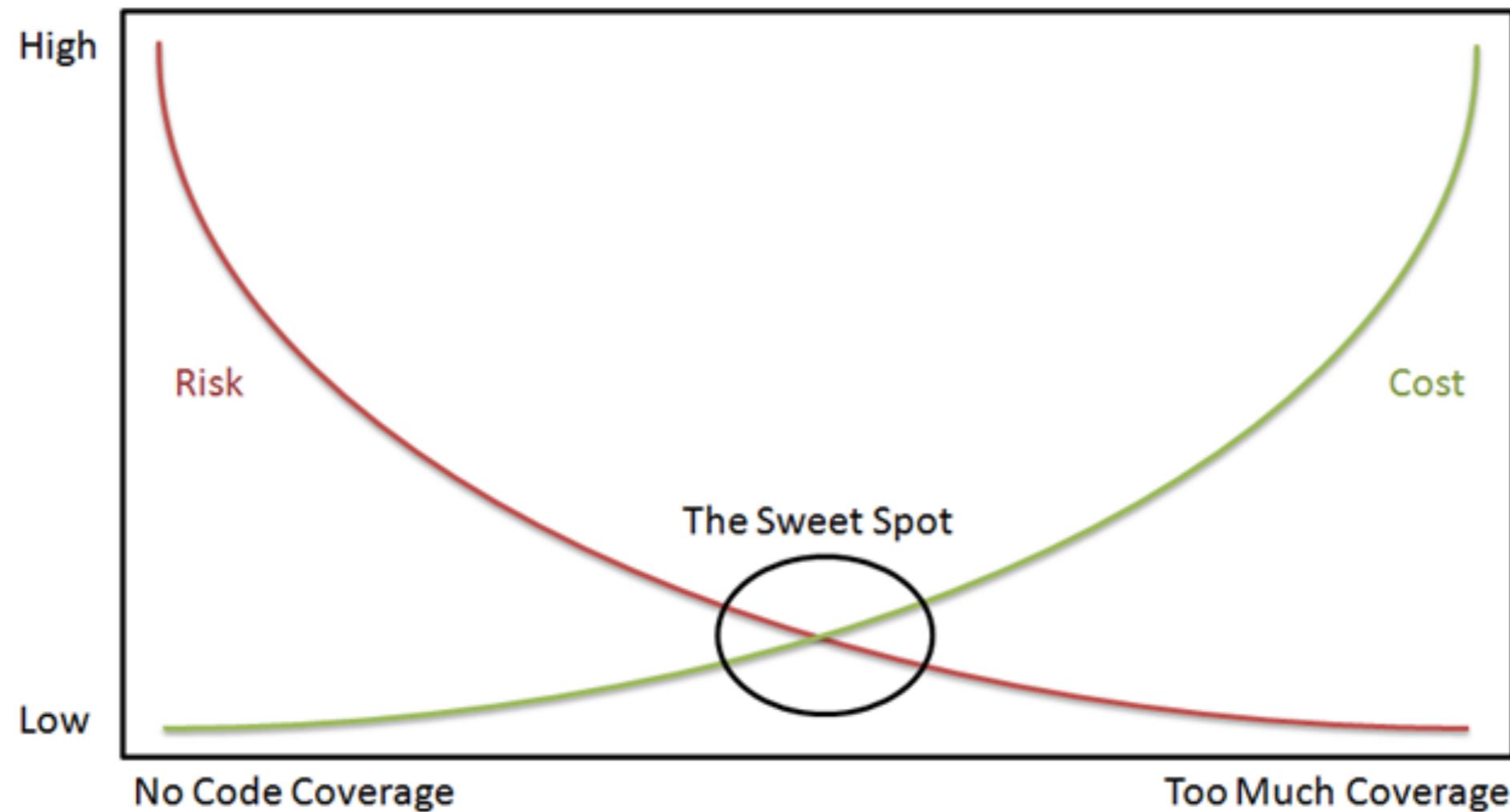
Design your pipeline/process



Continuous integration tools



Reduce risk with tests



Q/A

