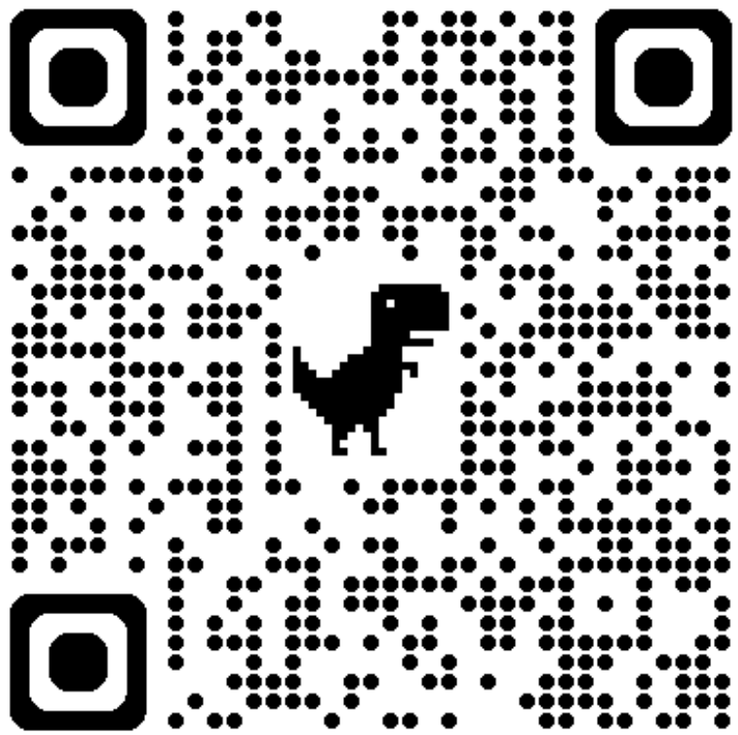
สยามชำนาญกิจ
SIAM CHAMNAN KIT

Full Stack Testing

Full Stack Testing

From manual to automation



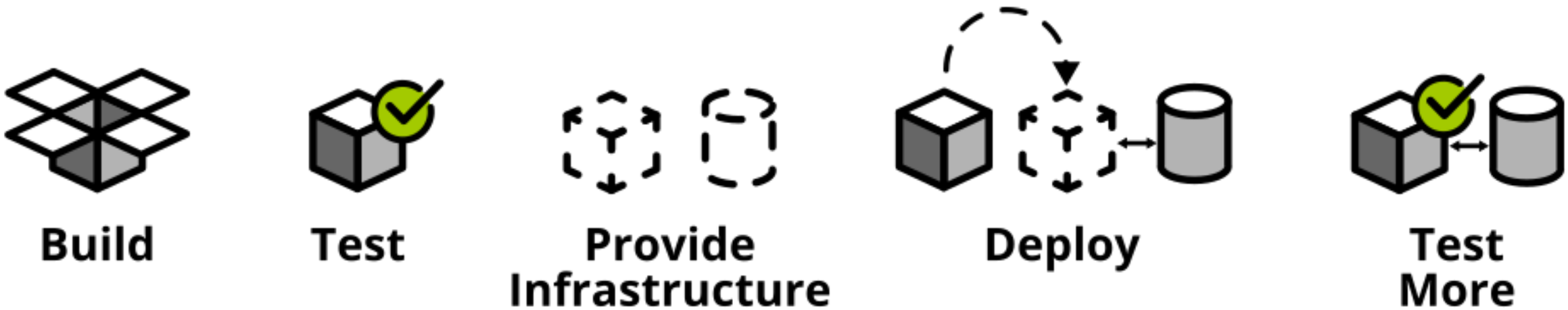
https://github.com/up1/course-full-stack-testing

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

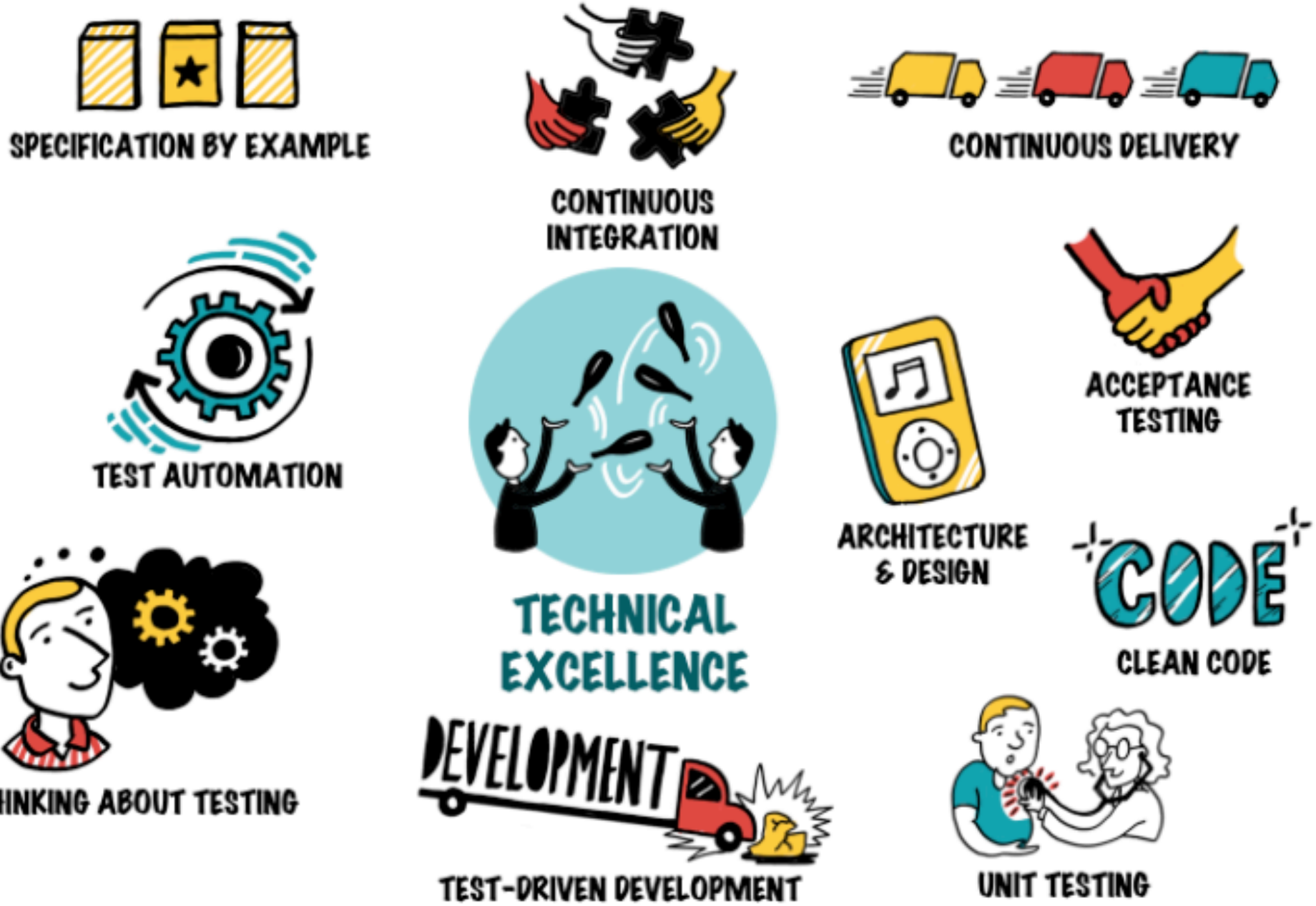
Quality vs. Quantity

Software Delivery



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

Technical excellence

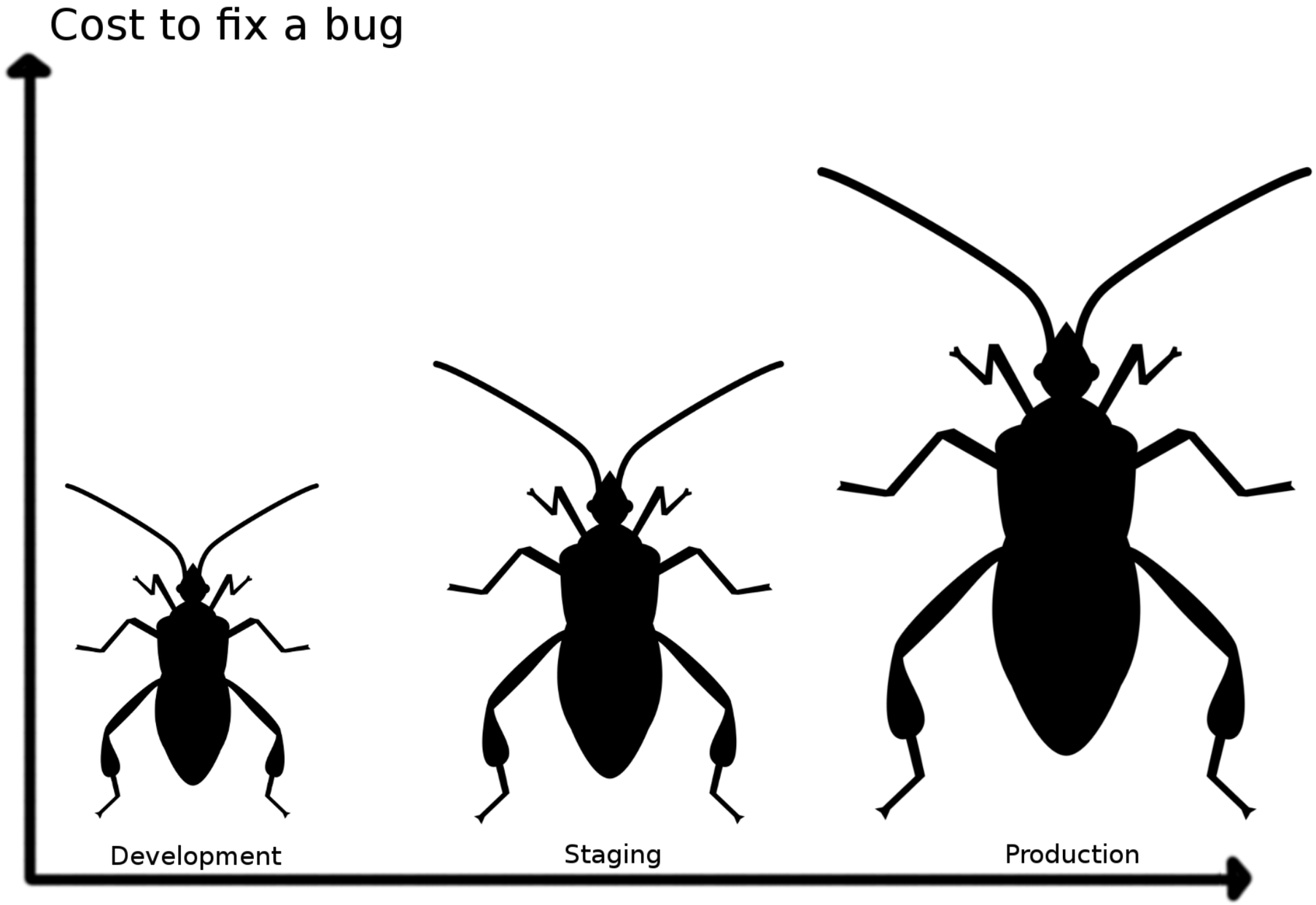


https://less.works/less/technical-excellence/index

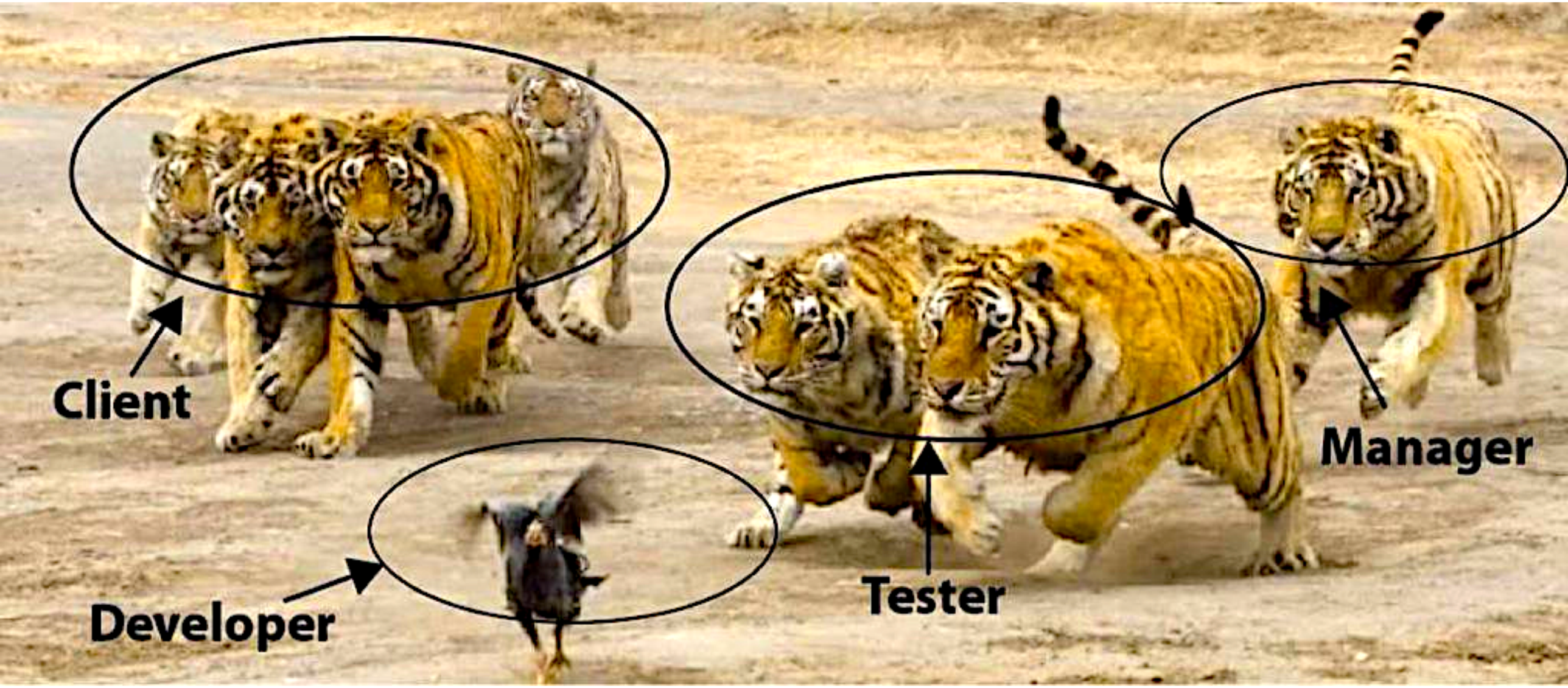
Start with Why …

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

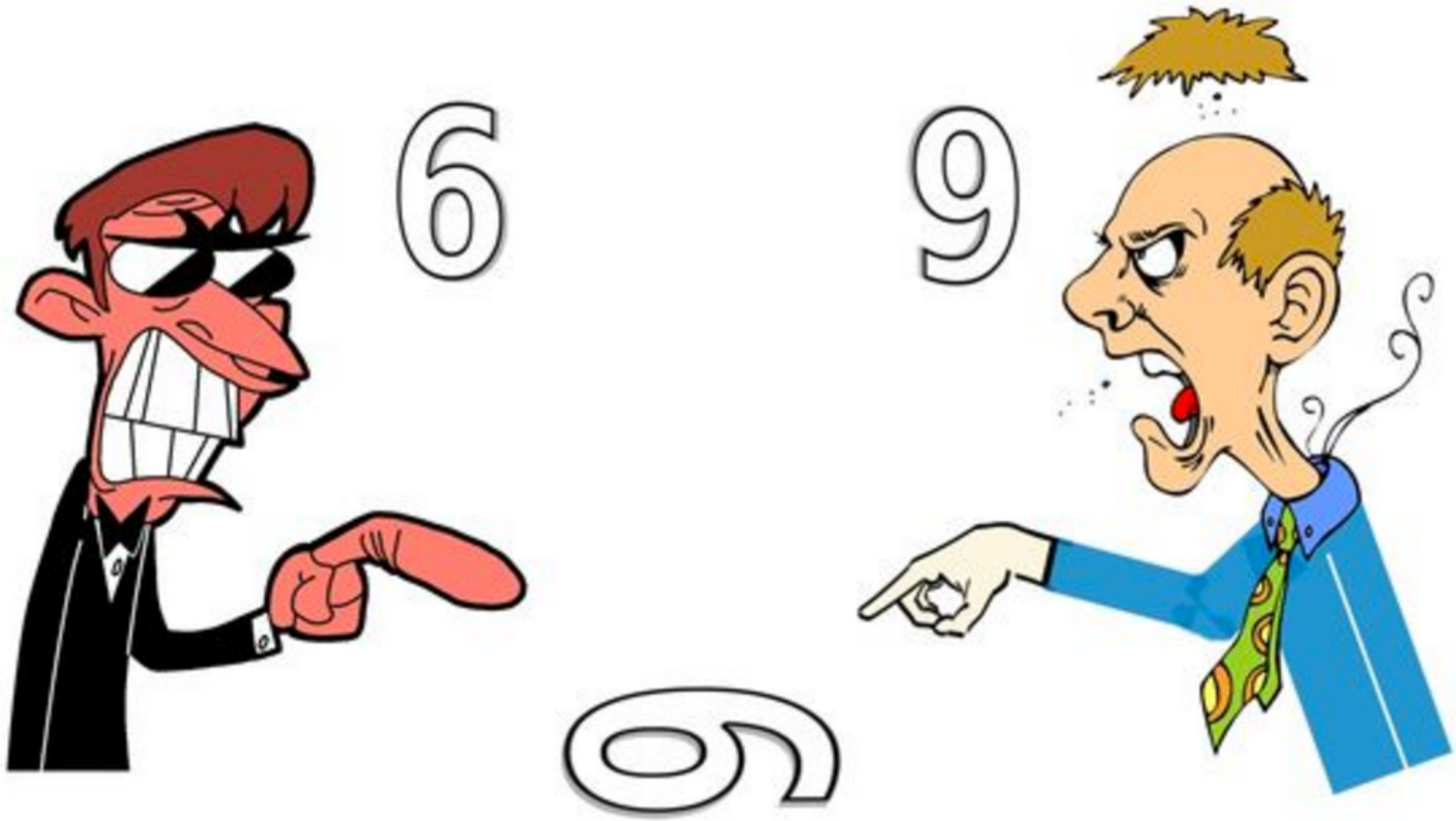
Edsger W. Dijkstra, 1970, Notes on Structured Programming



Deadline **D**riven **D**evelopment

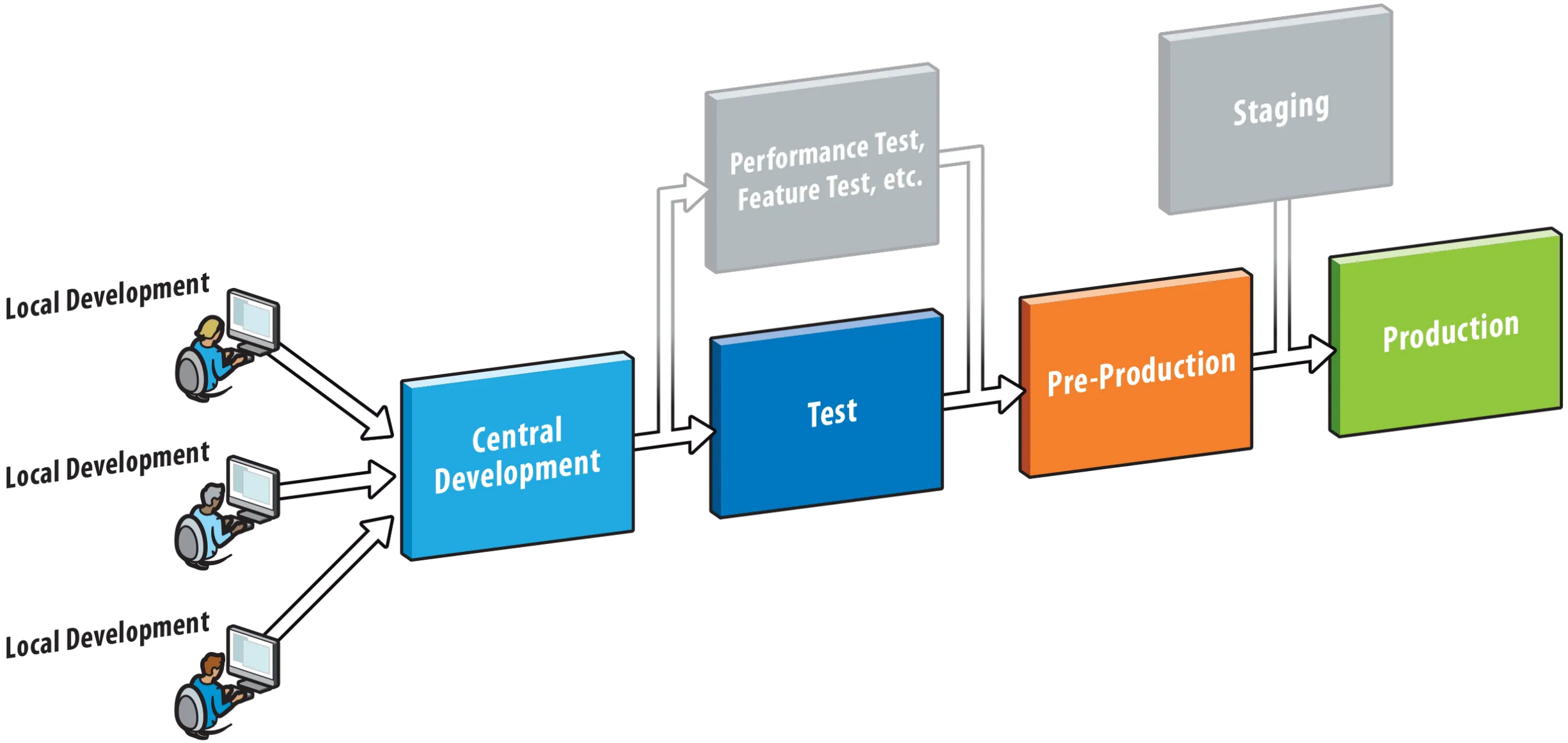


Developer vs Tester



Every time a developer changing the code !!

Environments to deliver software

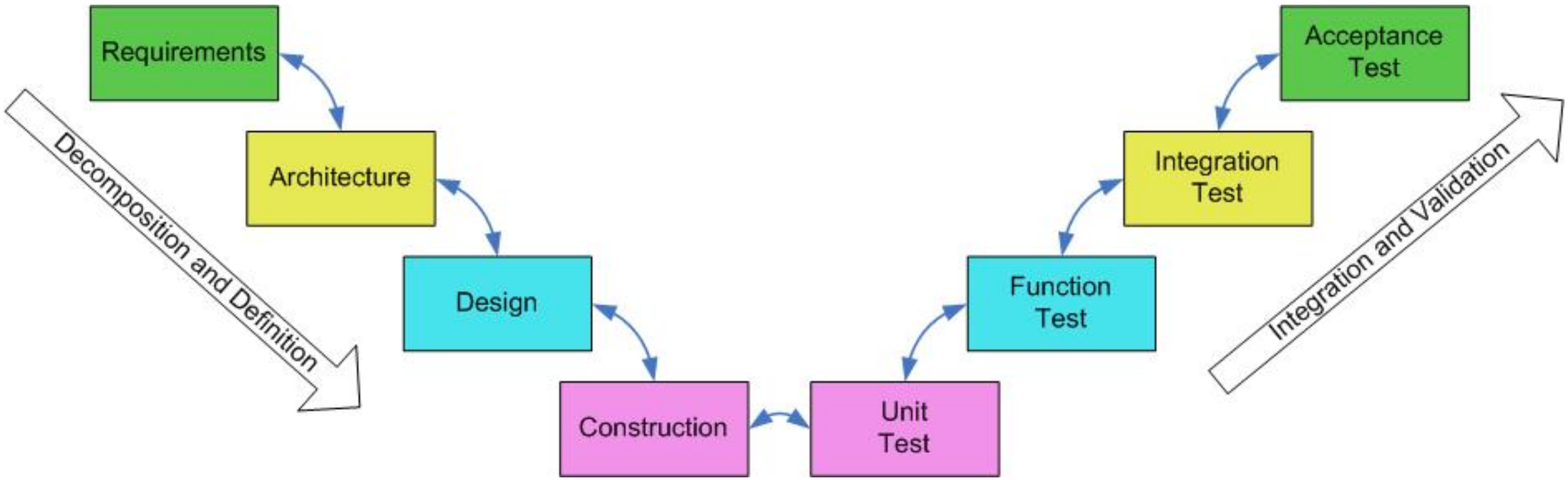


Why we need to test ?

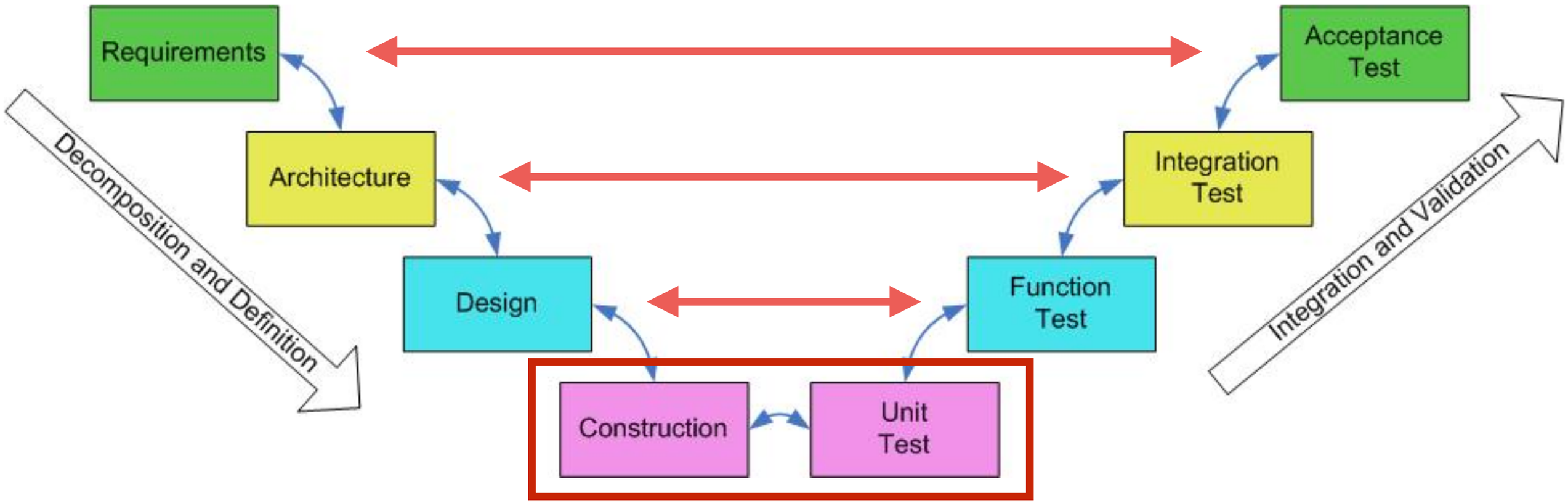
Help you to **catch bugs**
Boosted confidence
Quality code
Enforce **modularity** of your project
Develop features **faster**
Better documentation

What kind of test should we write ?

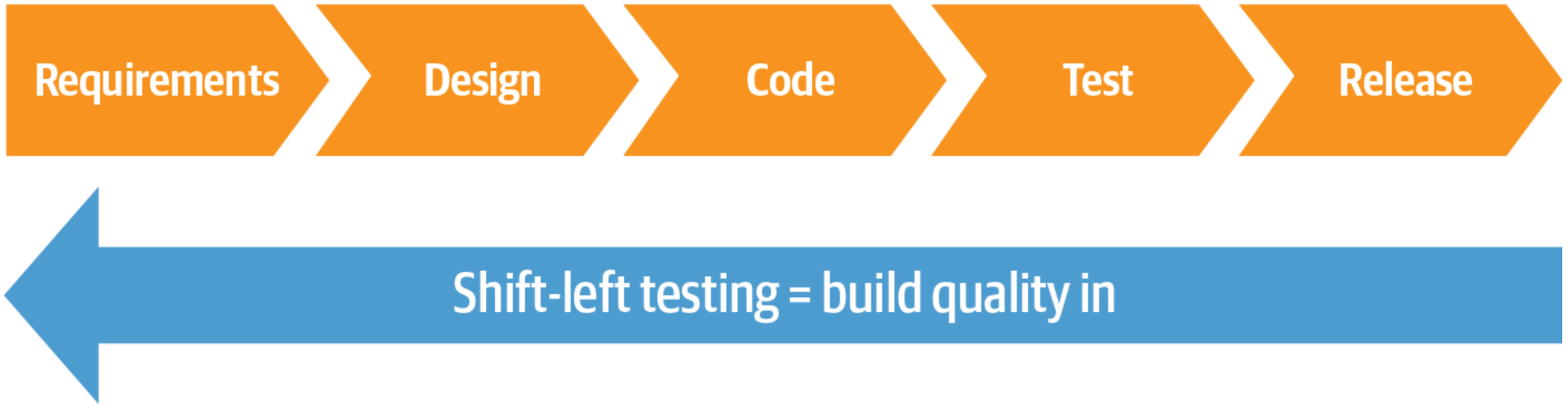
V Model



V Model

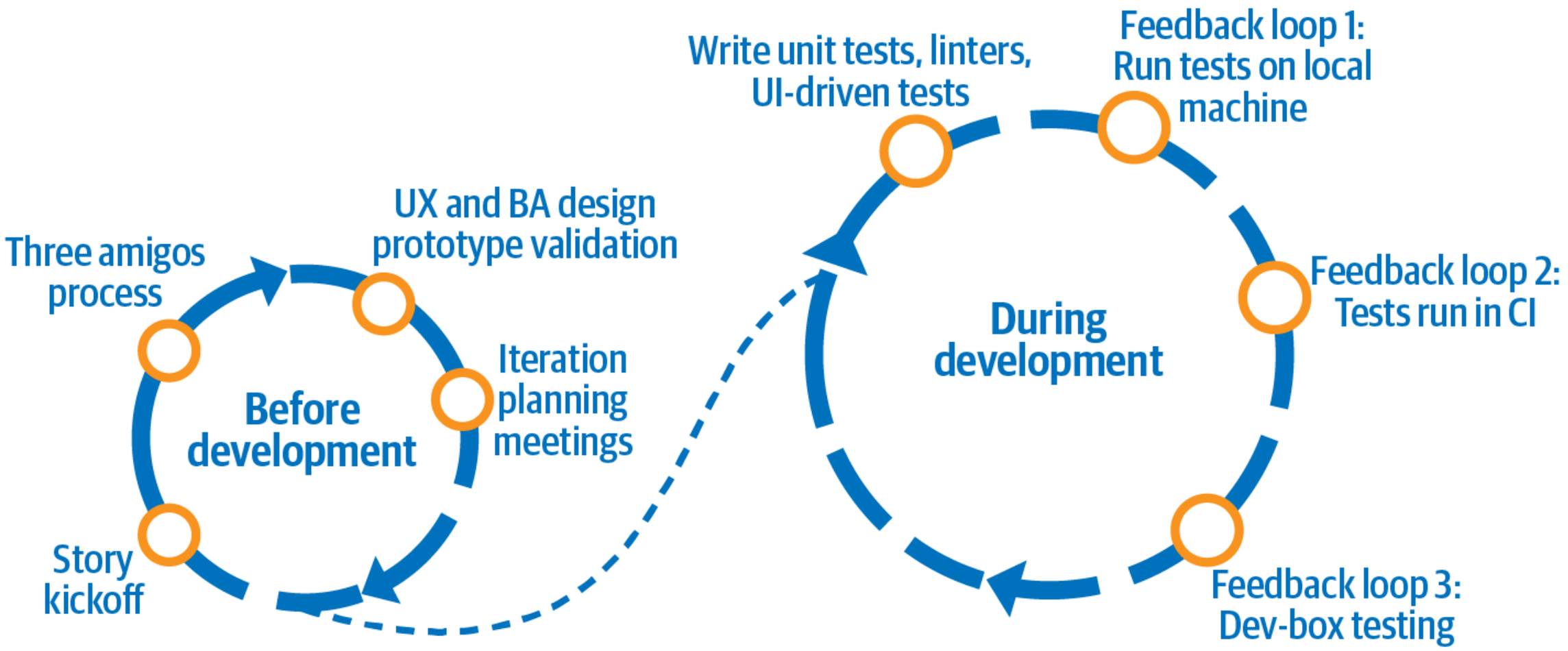


Shift Left Testing



Functional testing	Non-functional testing

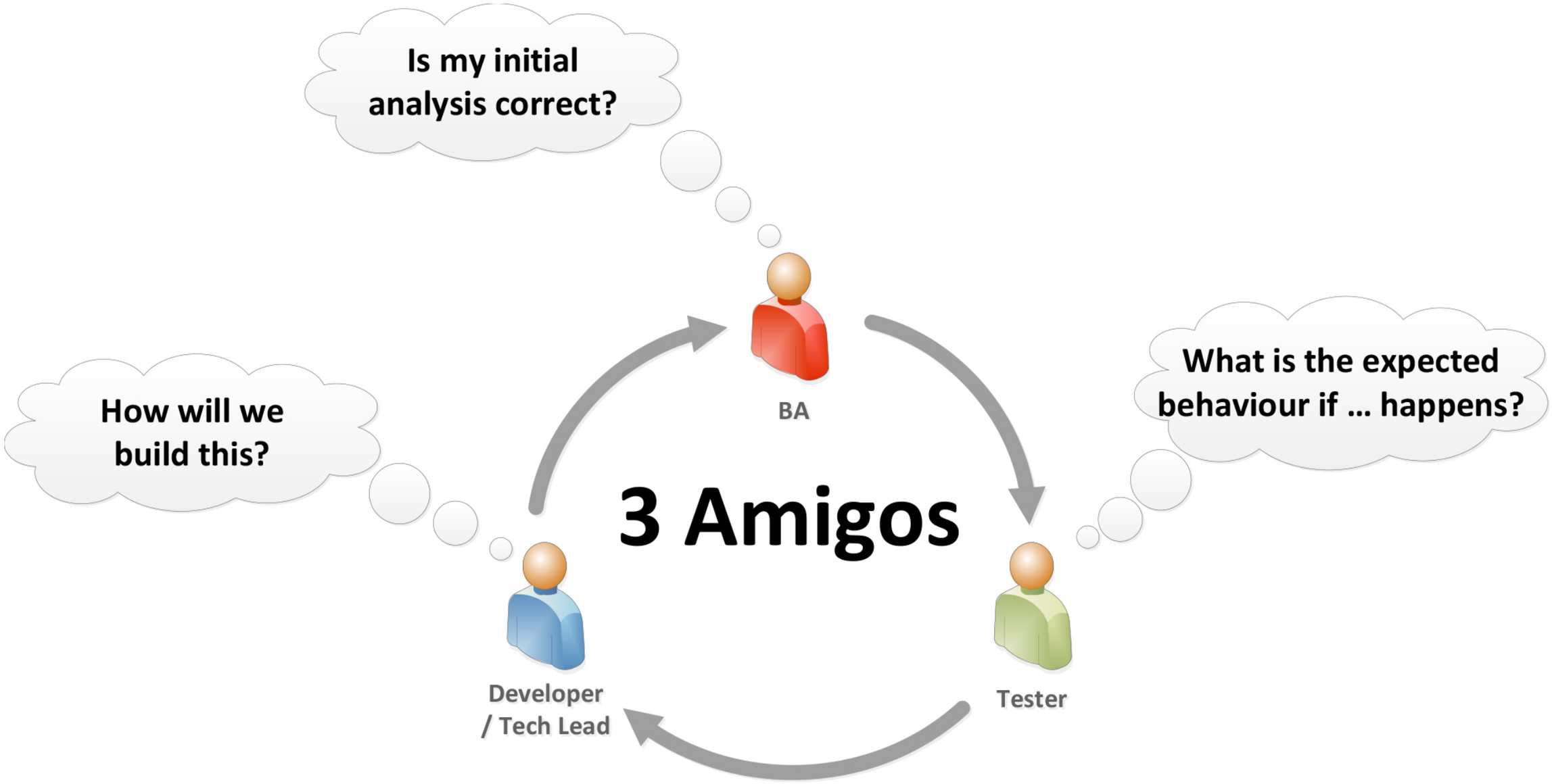
Quality Check ?



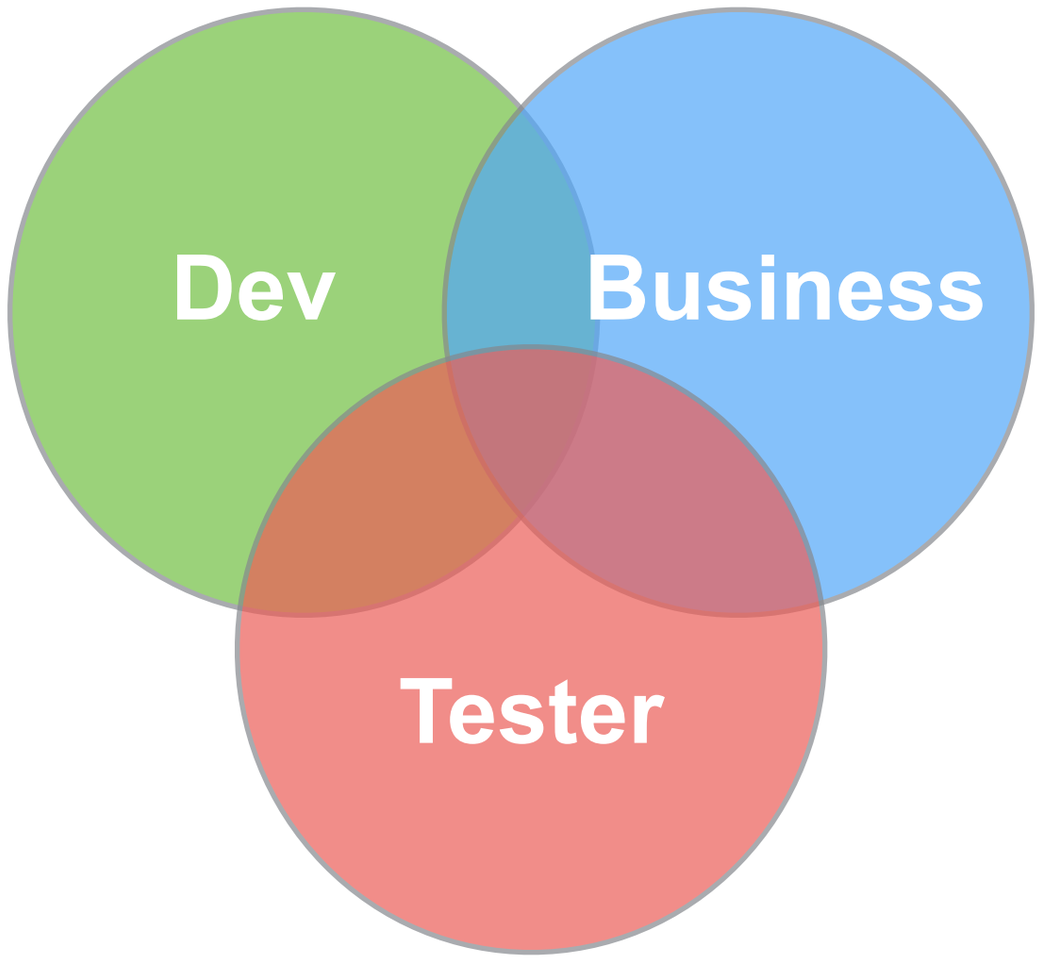
Software quality attributes

Correctness	Flexibility	Reusability
Efficiency	Usability	Interoperability
Reliability	Testability	Cost-effective
Integrity	Portability	Maintainability

Three Amigos Process

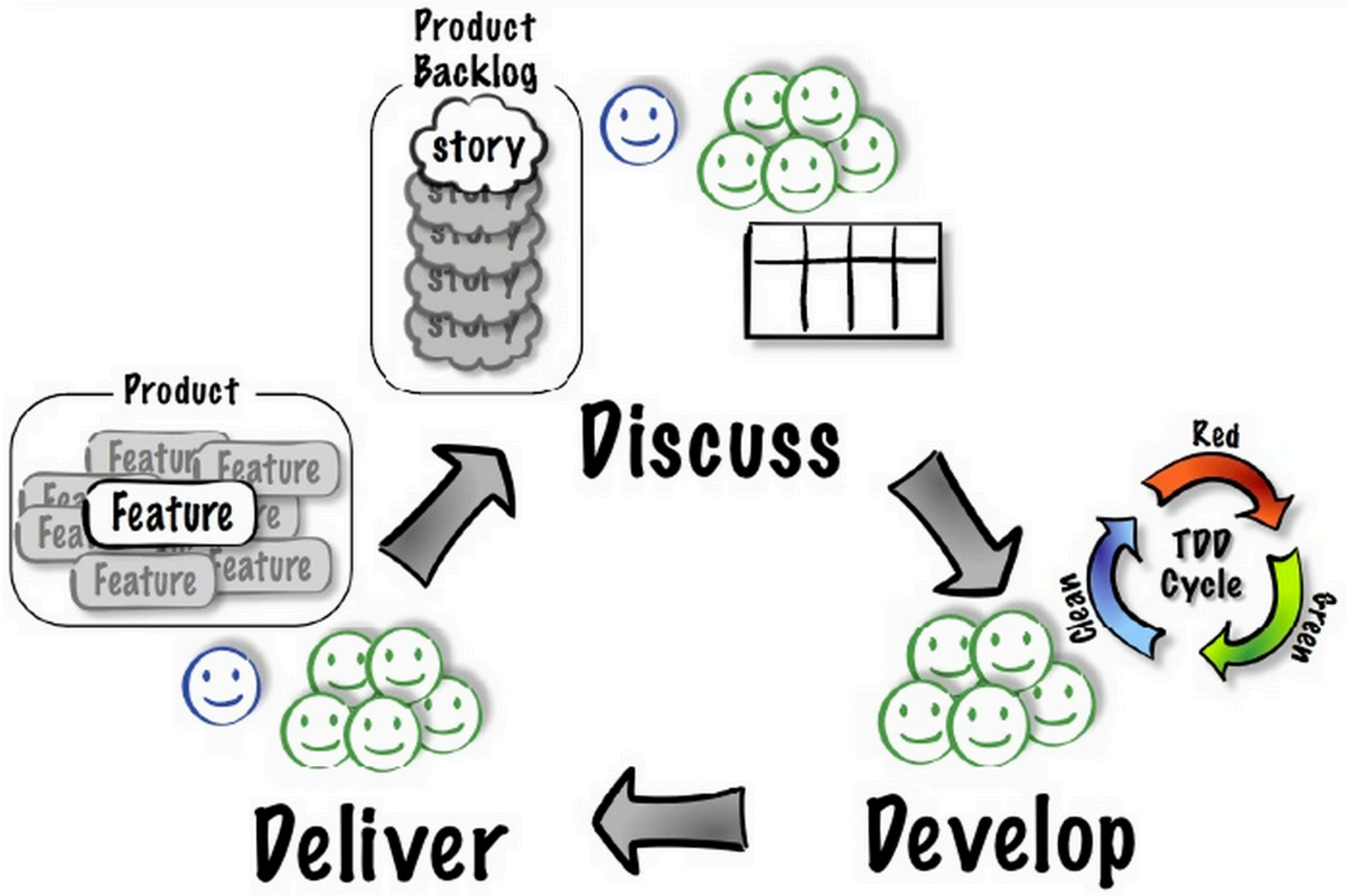


Power of Three



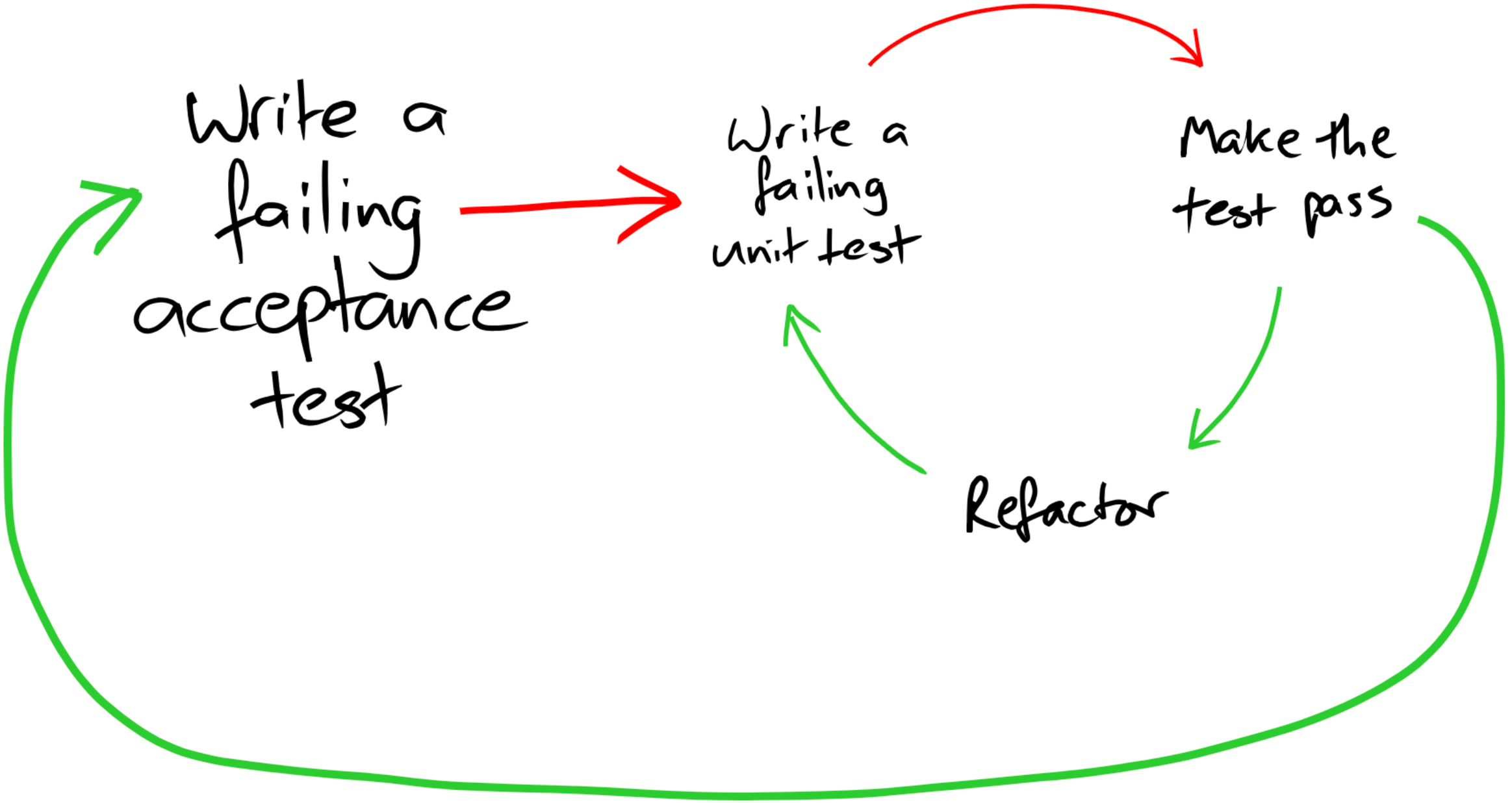
THINK before coding

Acceptance Test-Driven Development



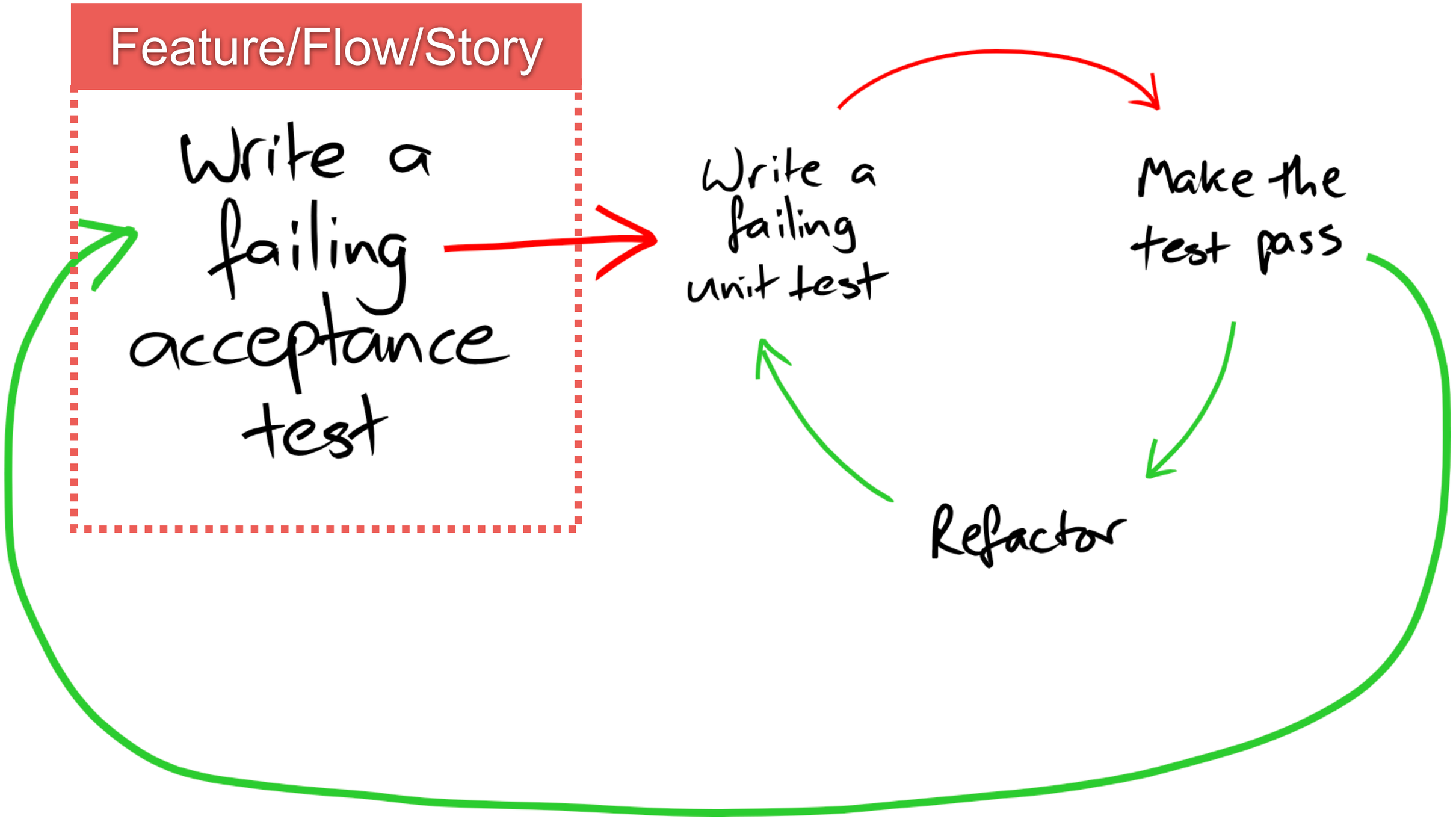
(Model developed with Pekka Klärck, Bas Vodde, and Craig Larman.)

Outside-in develop/test

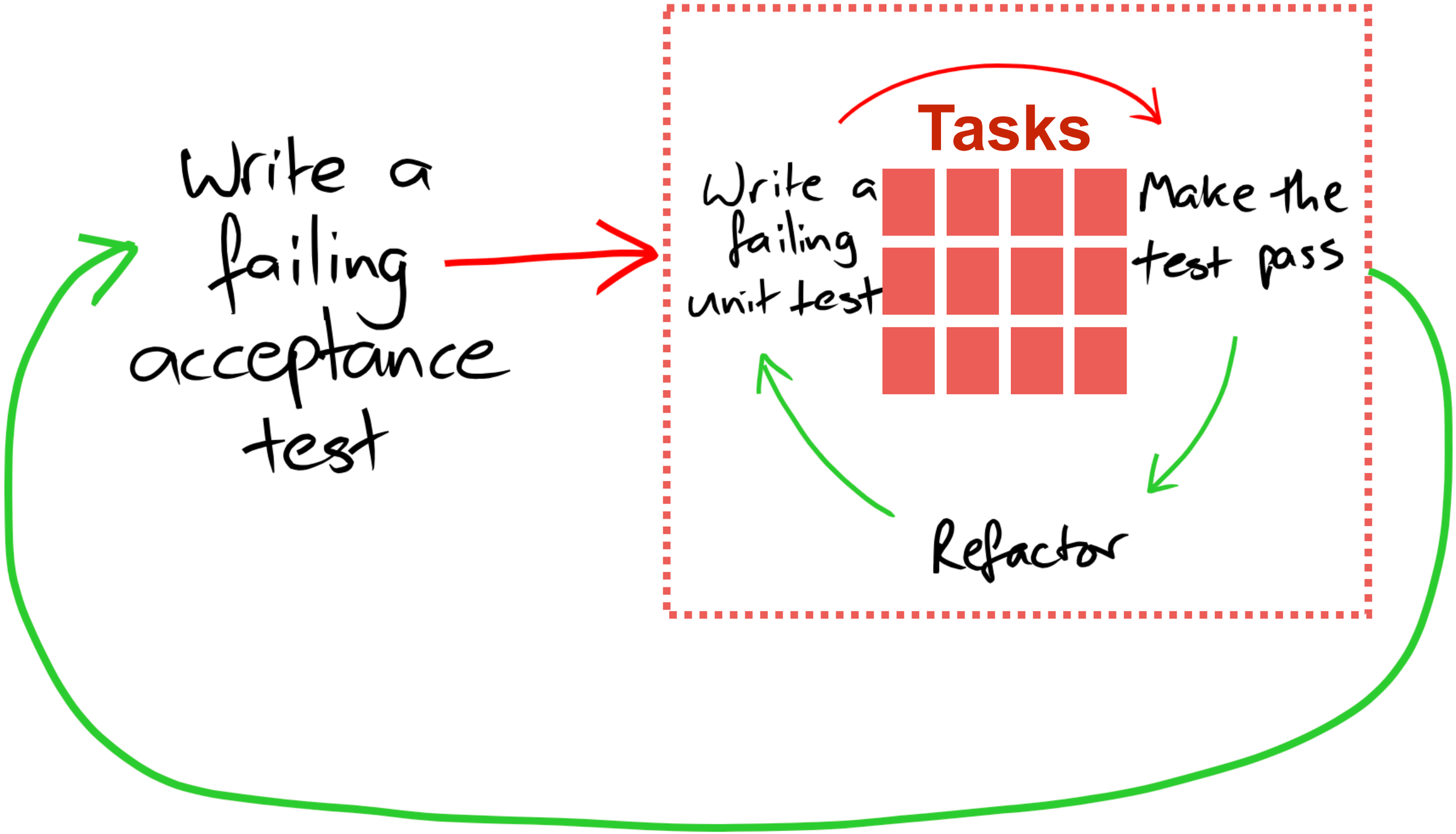


http://www.growing-object-oriented-software.com/

Outside-in develop/test



Outside-in develop/test



Tests Cases

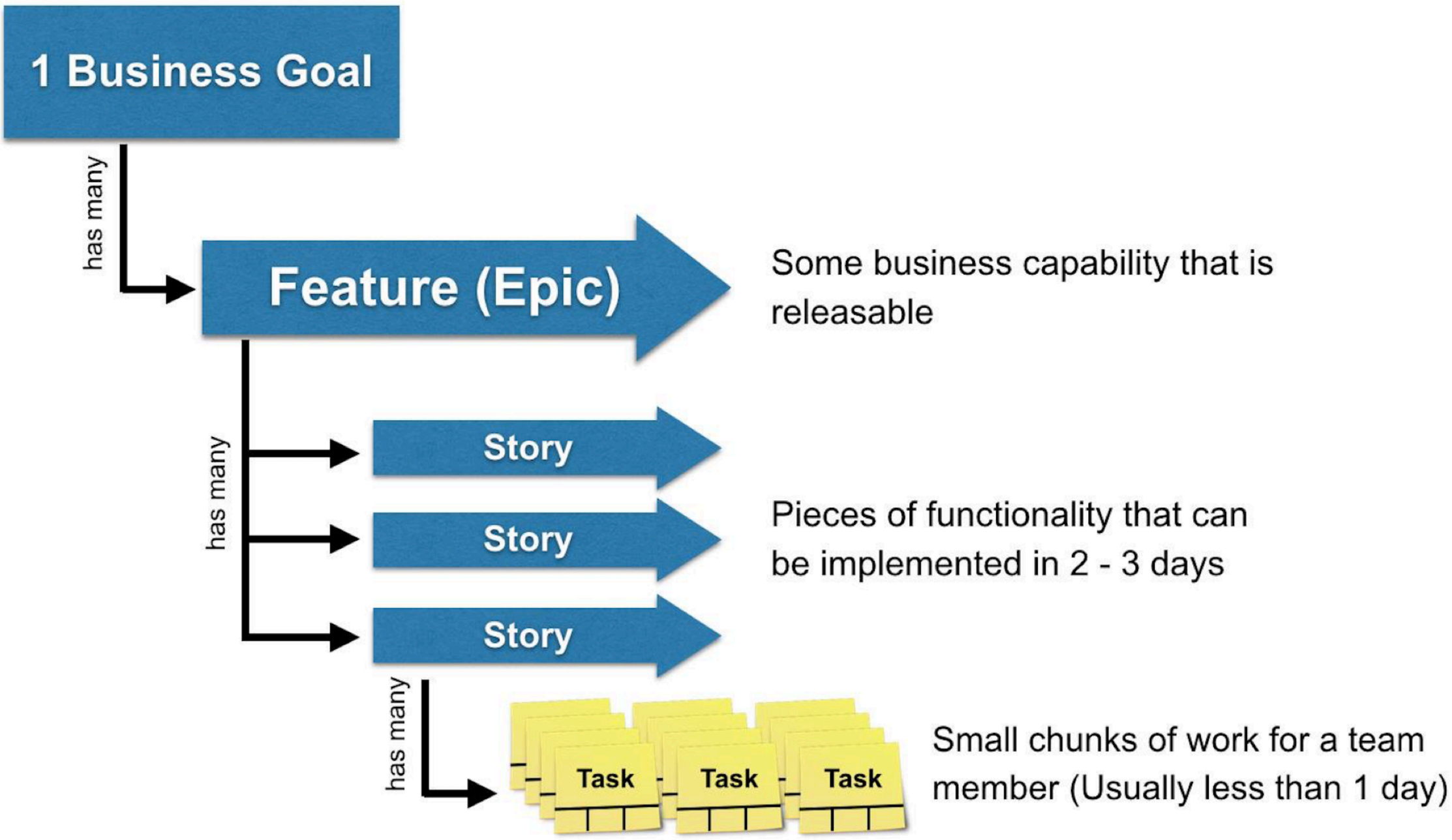
=

Business Criteria

+

Examples (real data)

Work break down



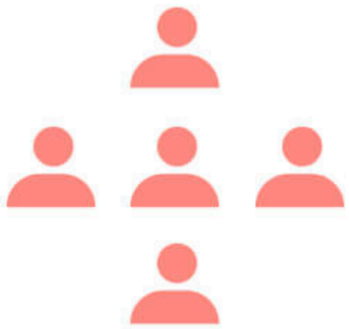
Key success factors

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



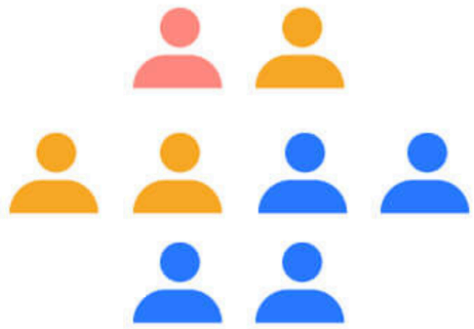
System analysts

Developers

Testers

Cross - Functional

Representatives from the various functions

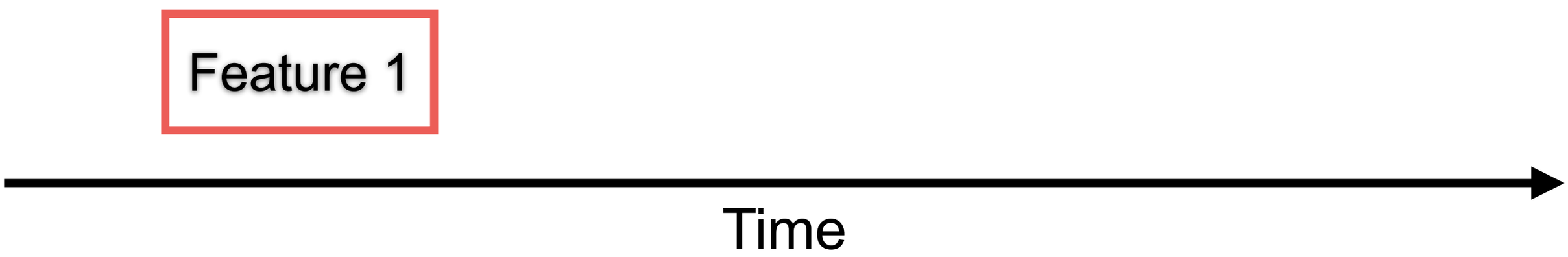


Development Team

Iterative and Incremental process

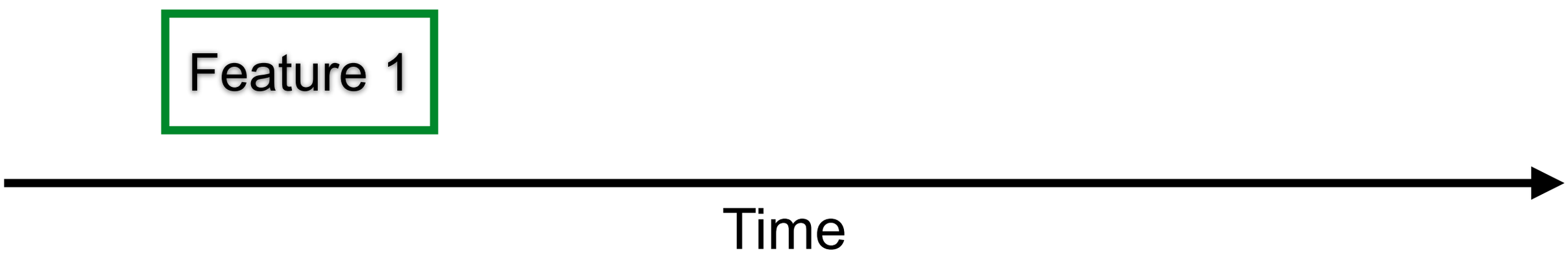
Iterative and incremental process

Feature 1 is Done ?



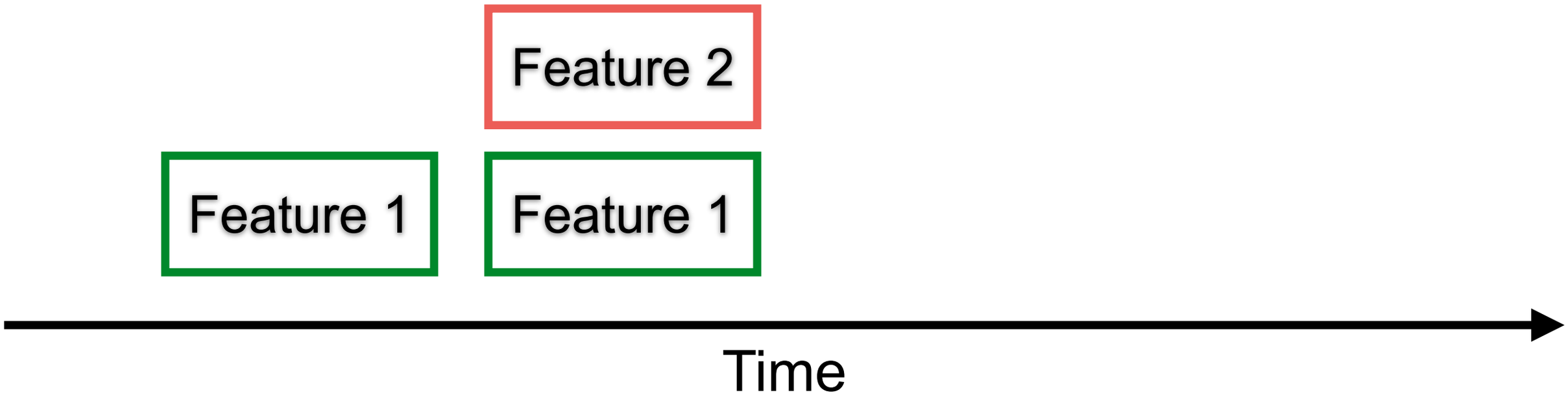
Iterative and incremental process

Done = coded and tested



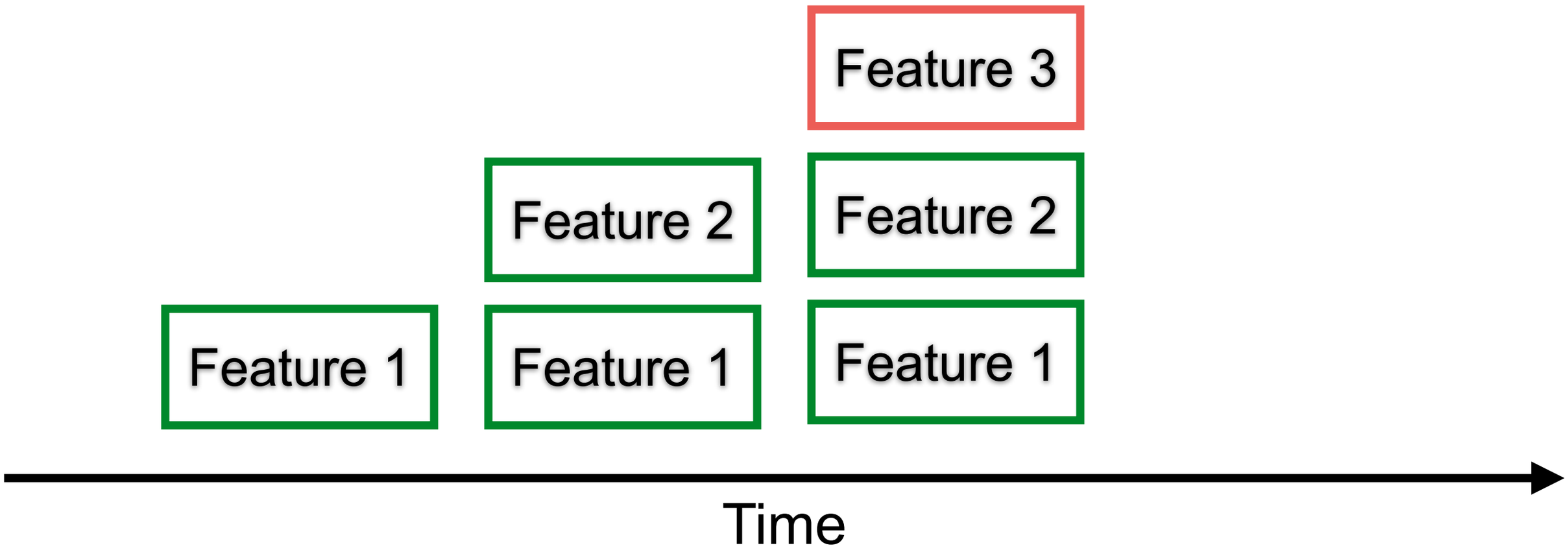
Iterative and incremental process

Done = coded and tested



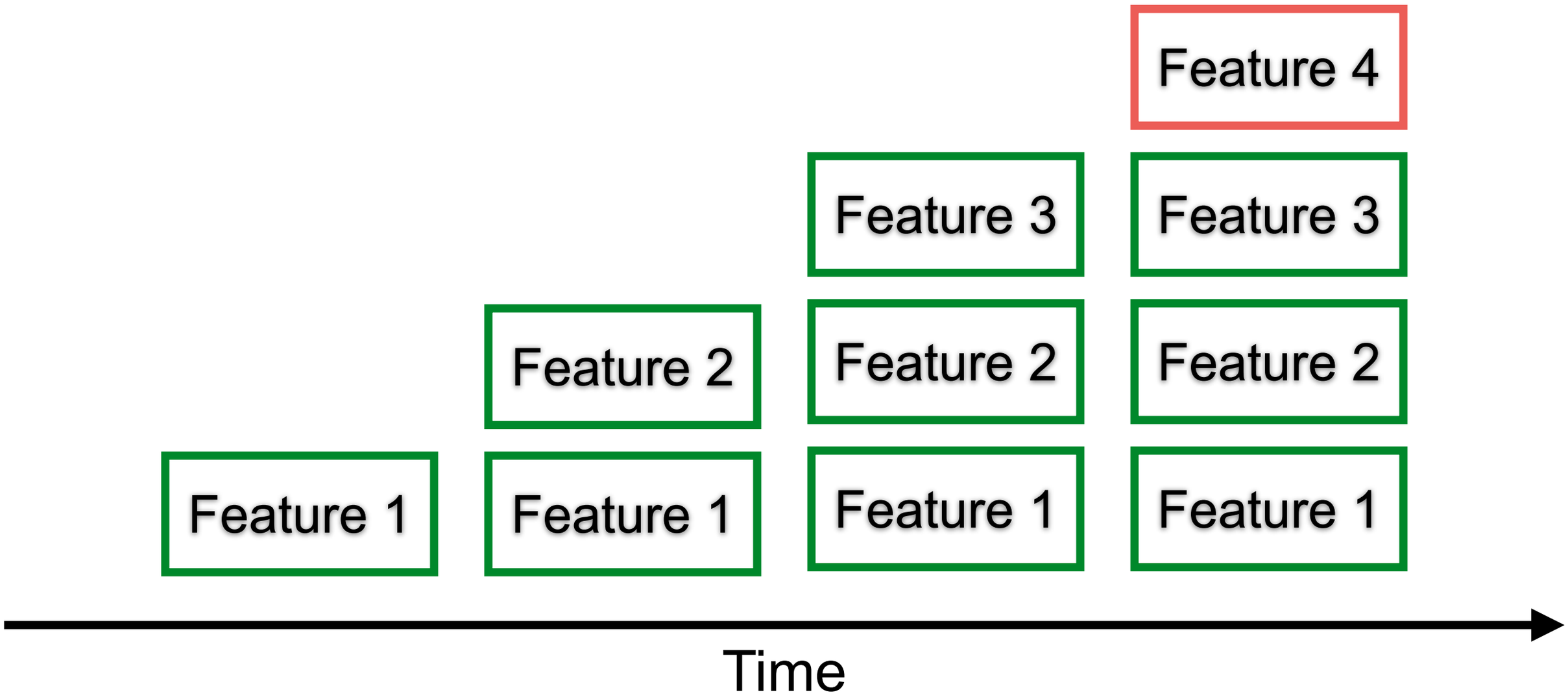
Iterative and incremental process

Done = coded and tested



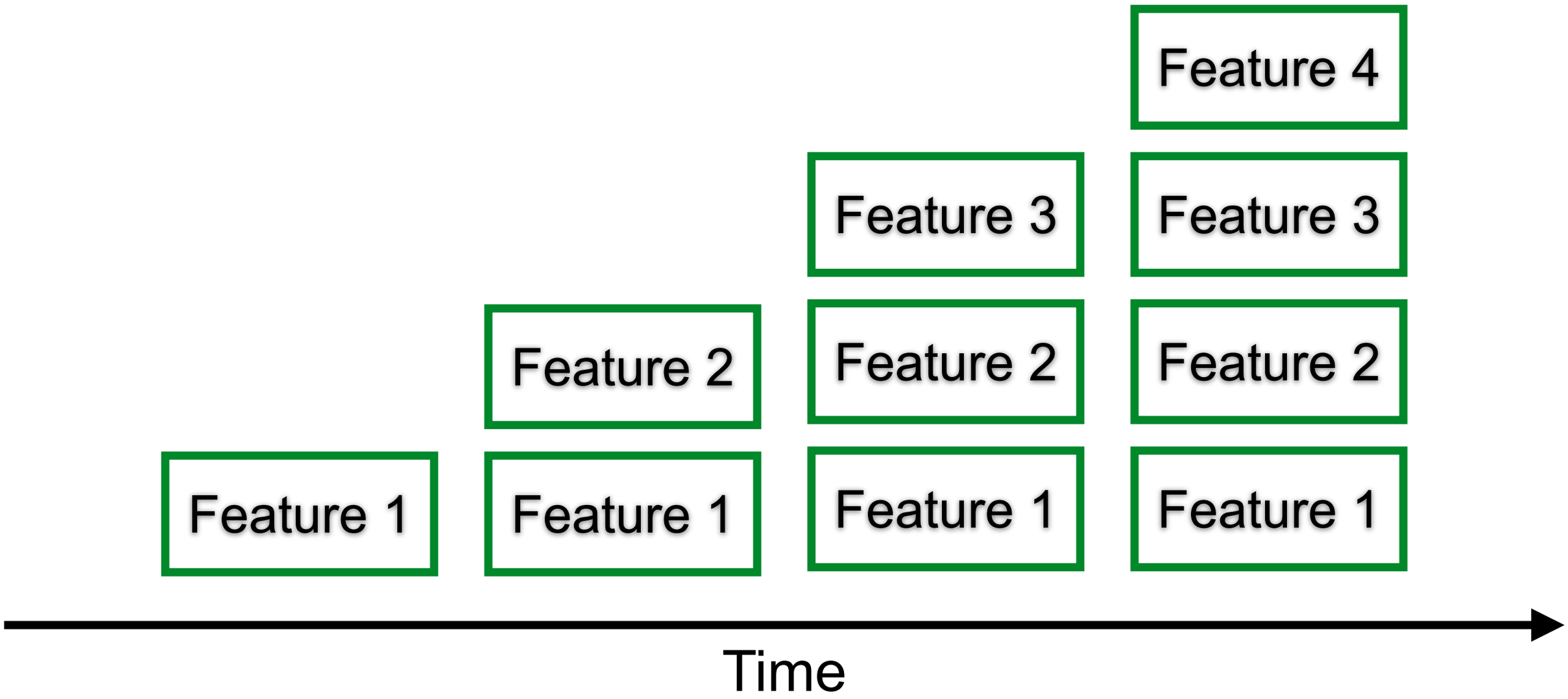
Iterative and incremental process

Done = coded and tested

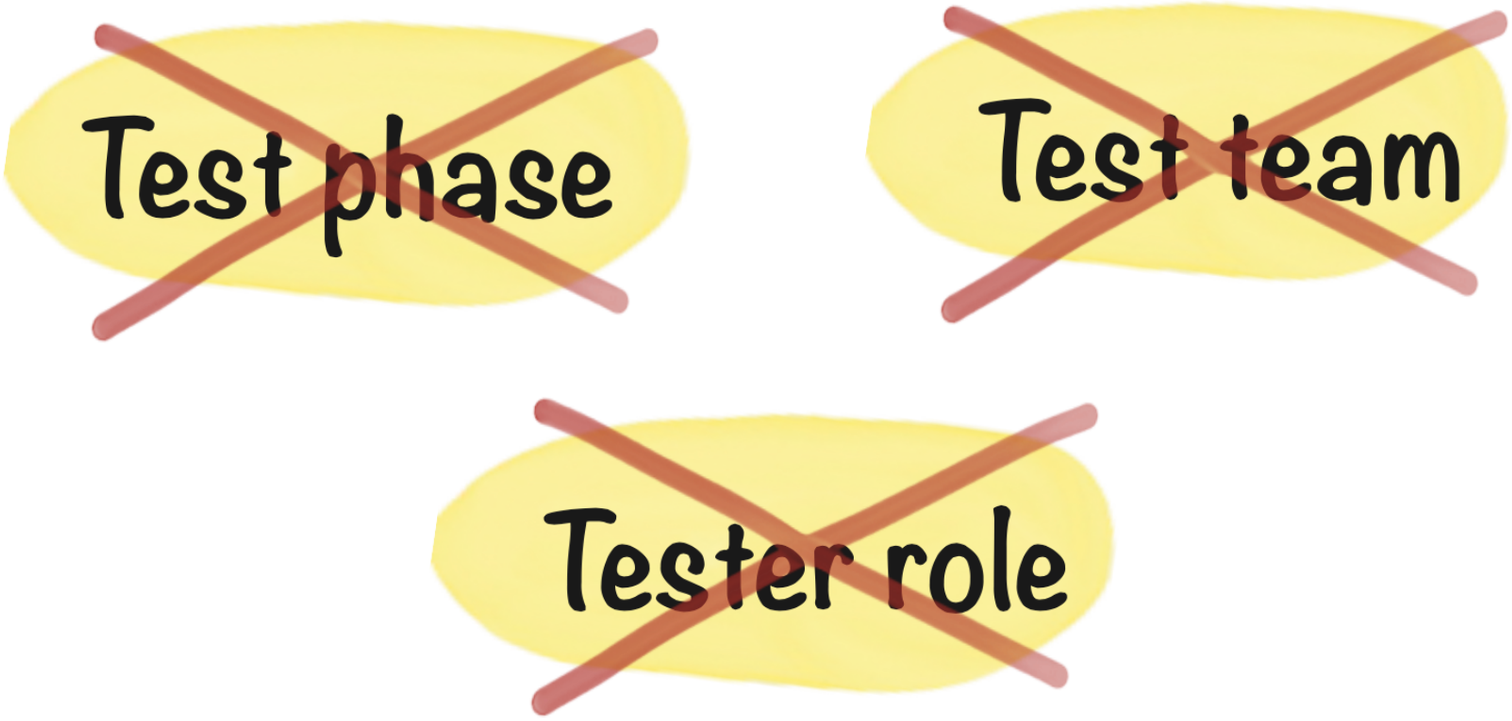


Iterative and incremental process

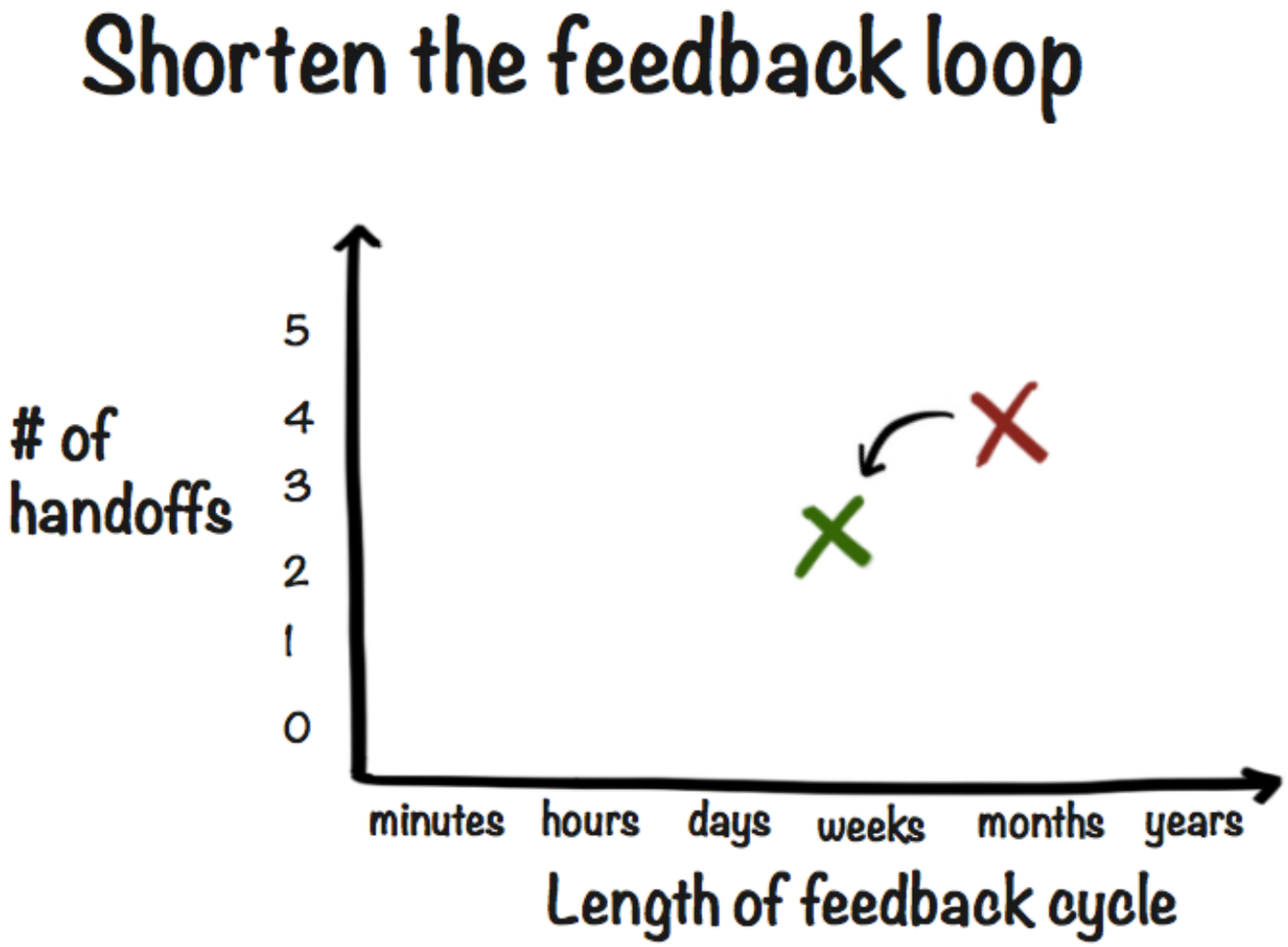
Done = coded and tested

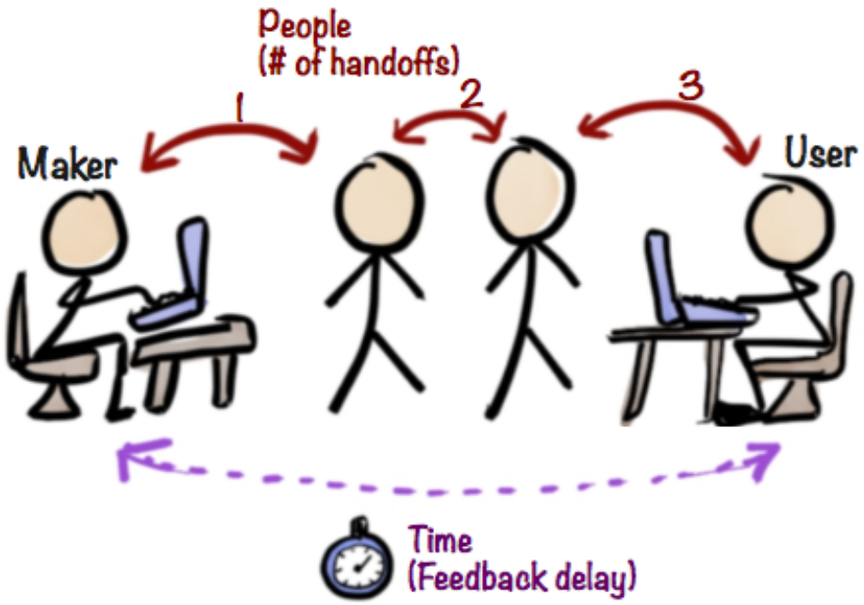


Testing is activity

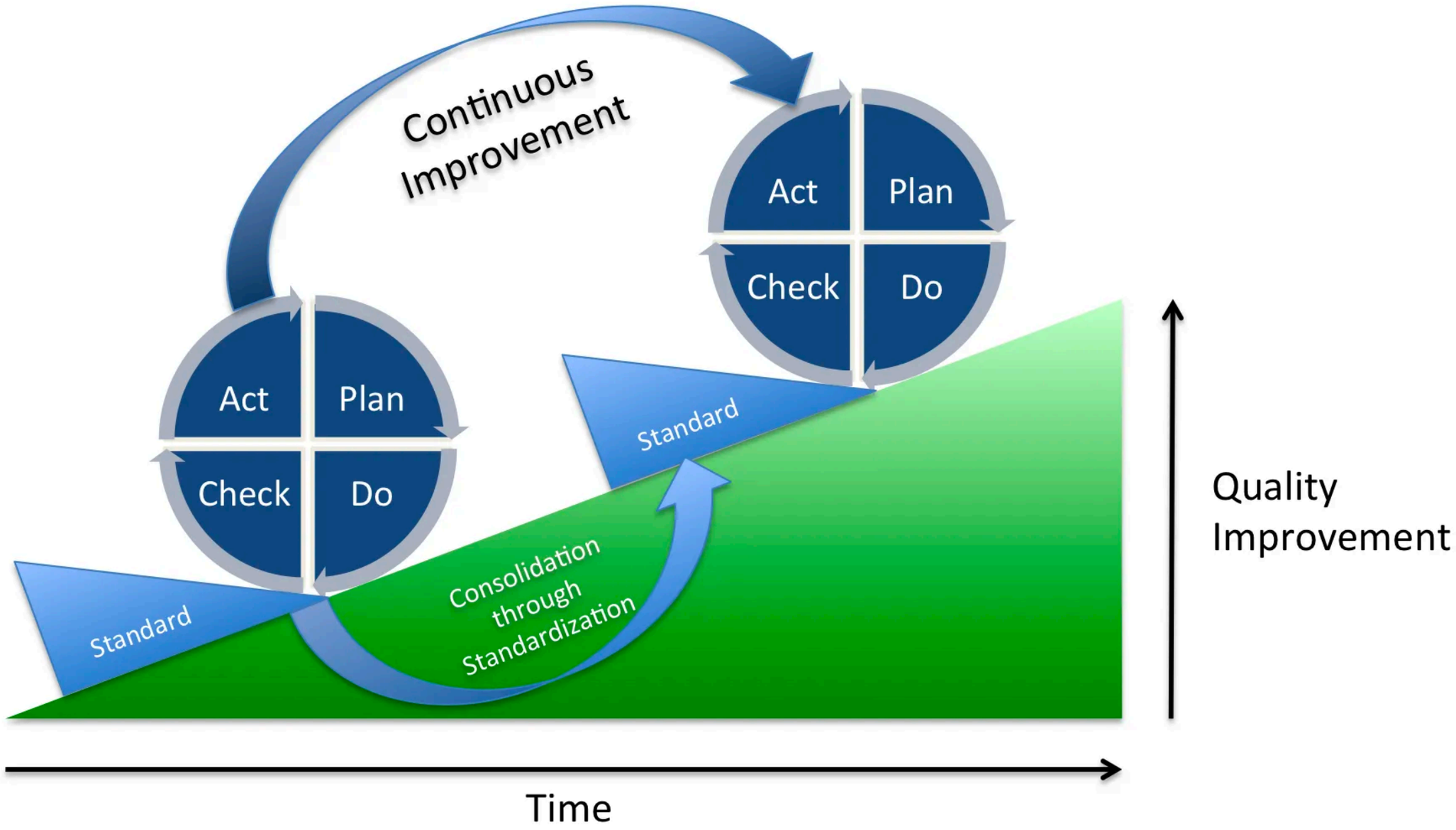


Fast feedback loop

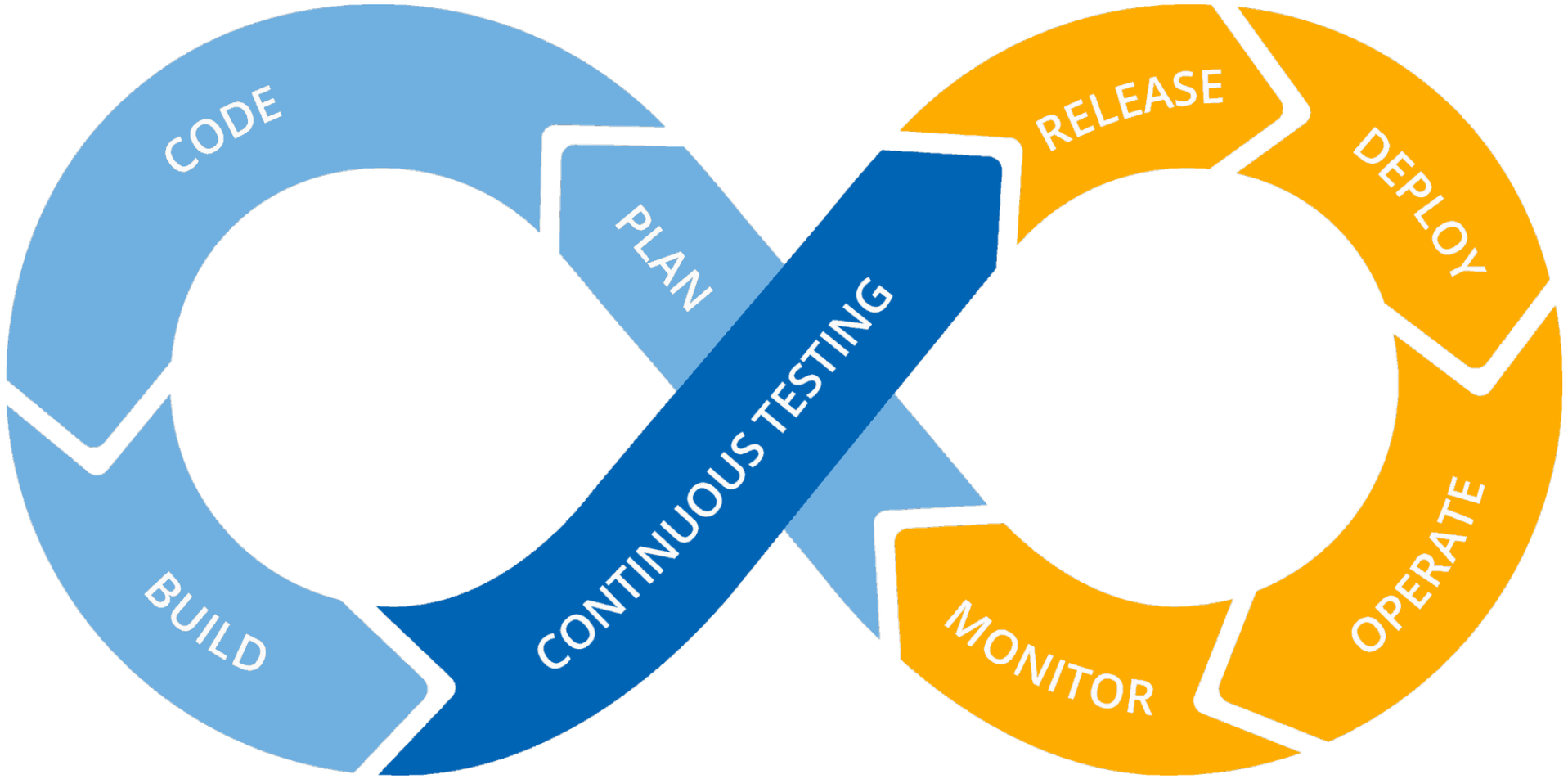




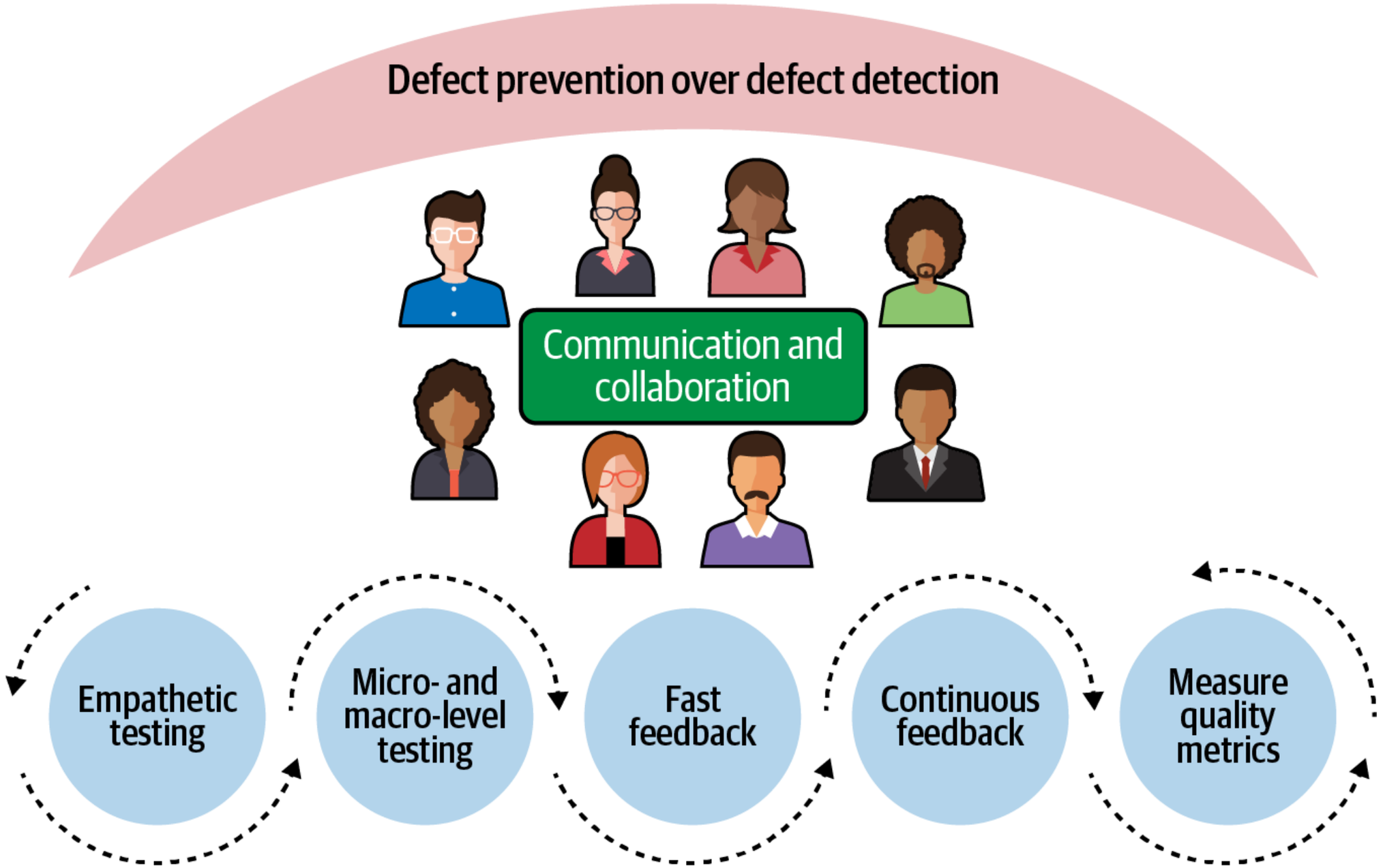
Continuous improvement



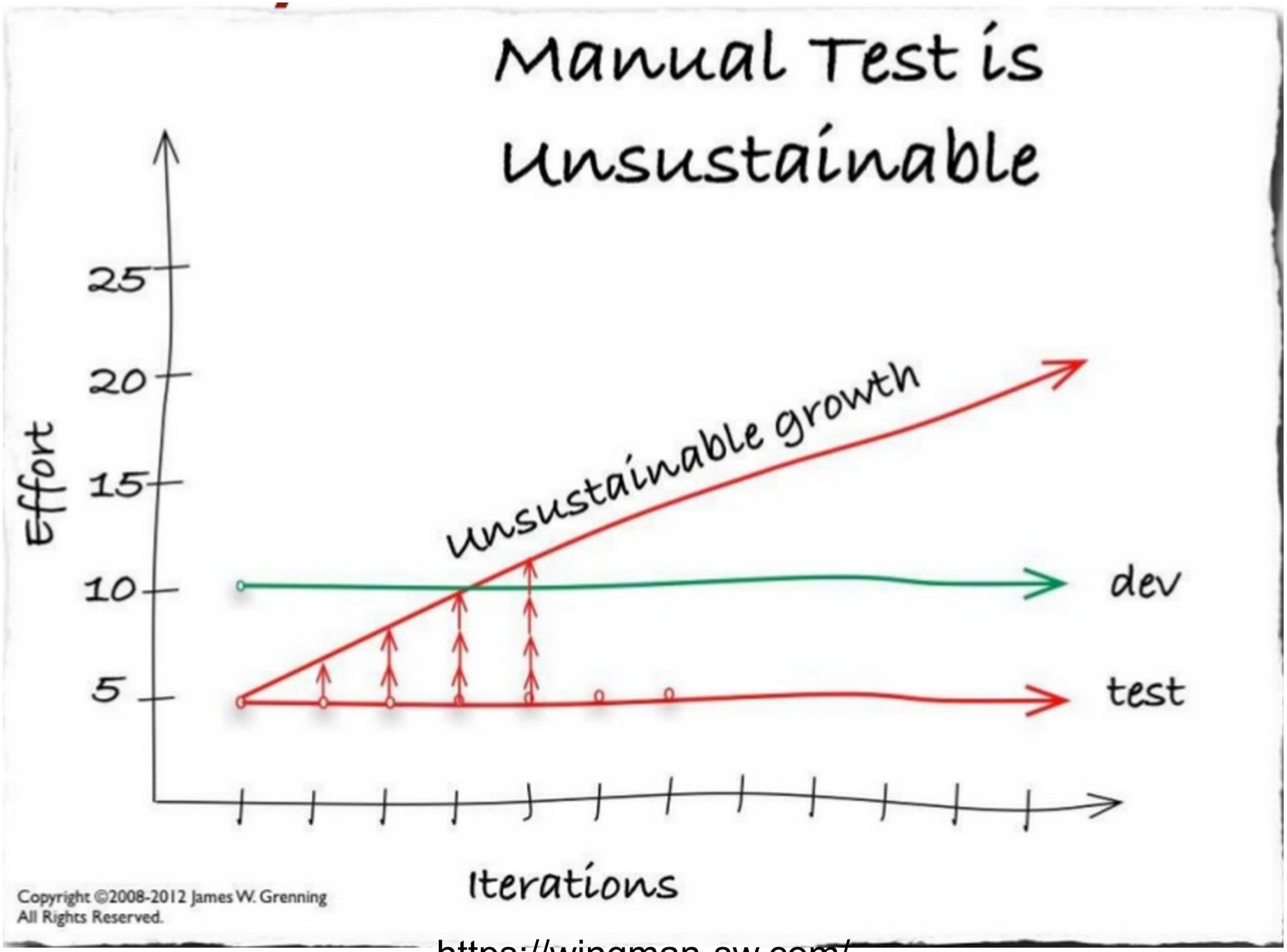
Continuous testing and feedback



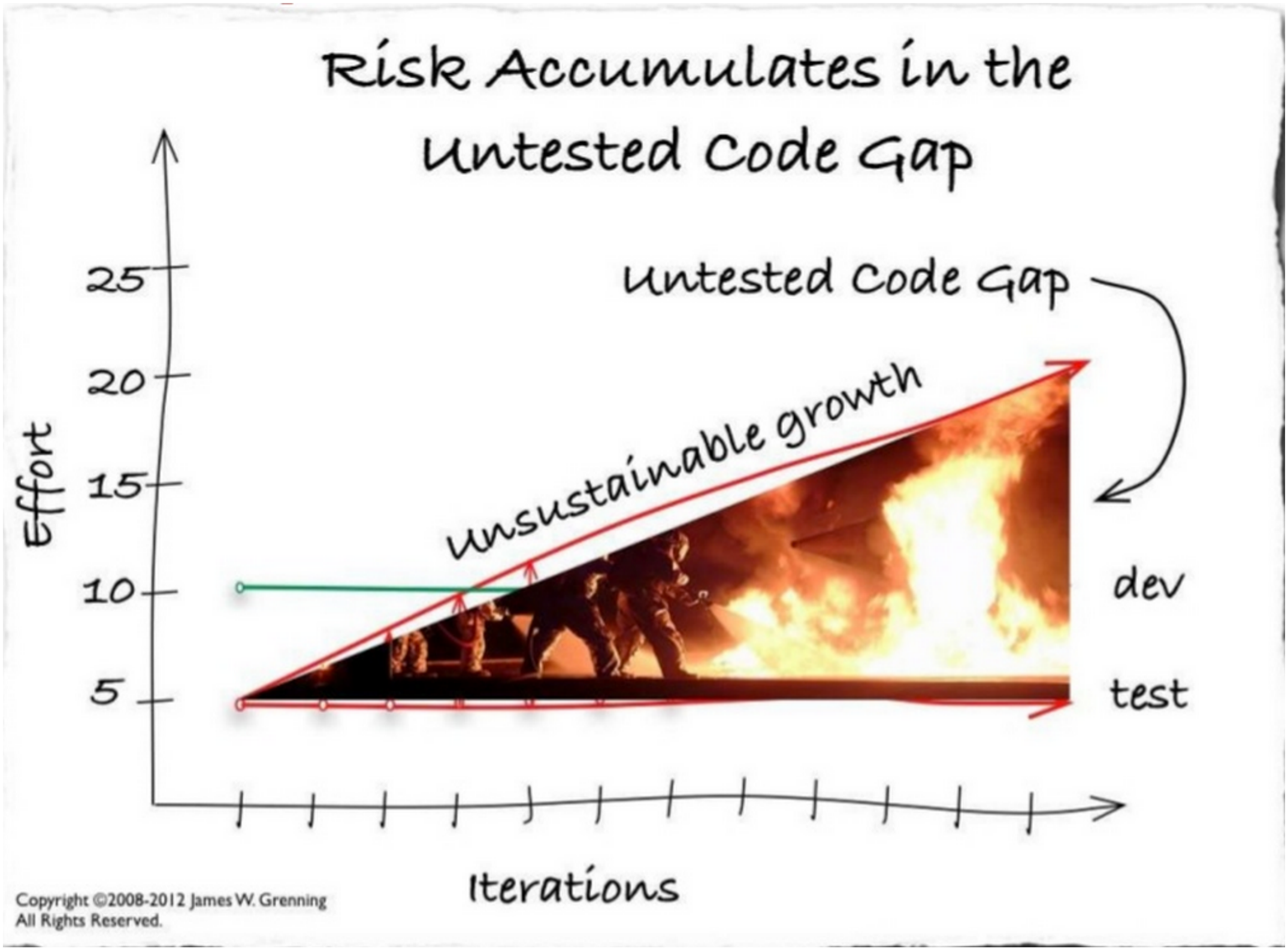
Principles of Testing



Manual testing ?



https://wingman-sw.com/

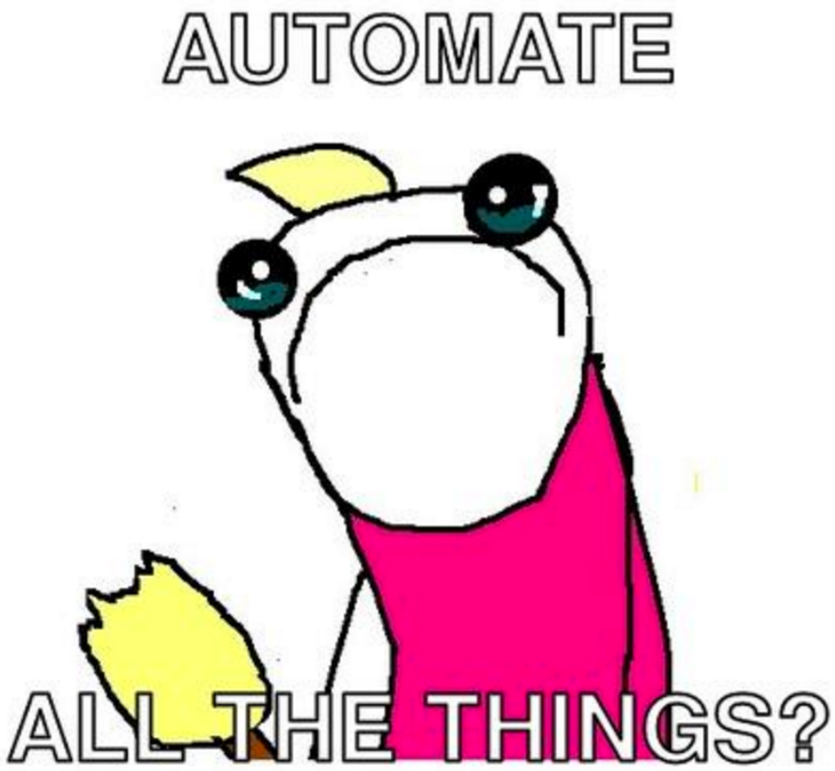


https://wingman-sw.com/

Regression Testing

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

We need automation !!

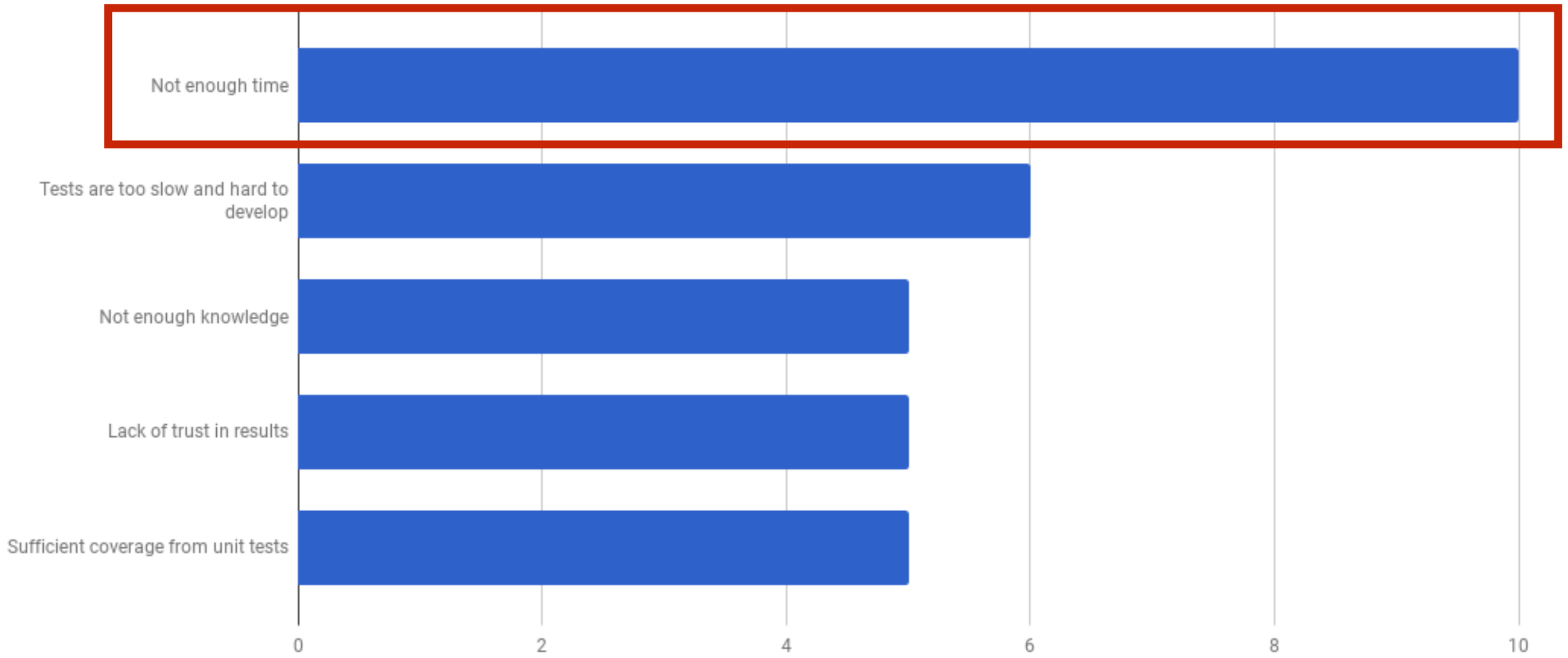


But,

It’s take time to learning and practice !!

Why not write automated tests ?

Not enough time !!!



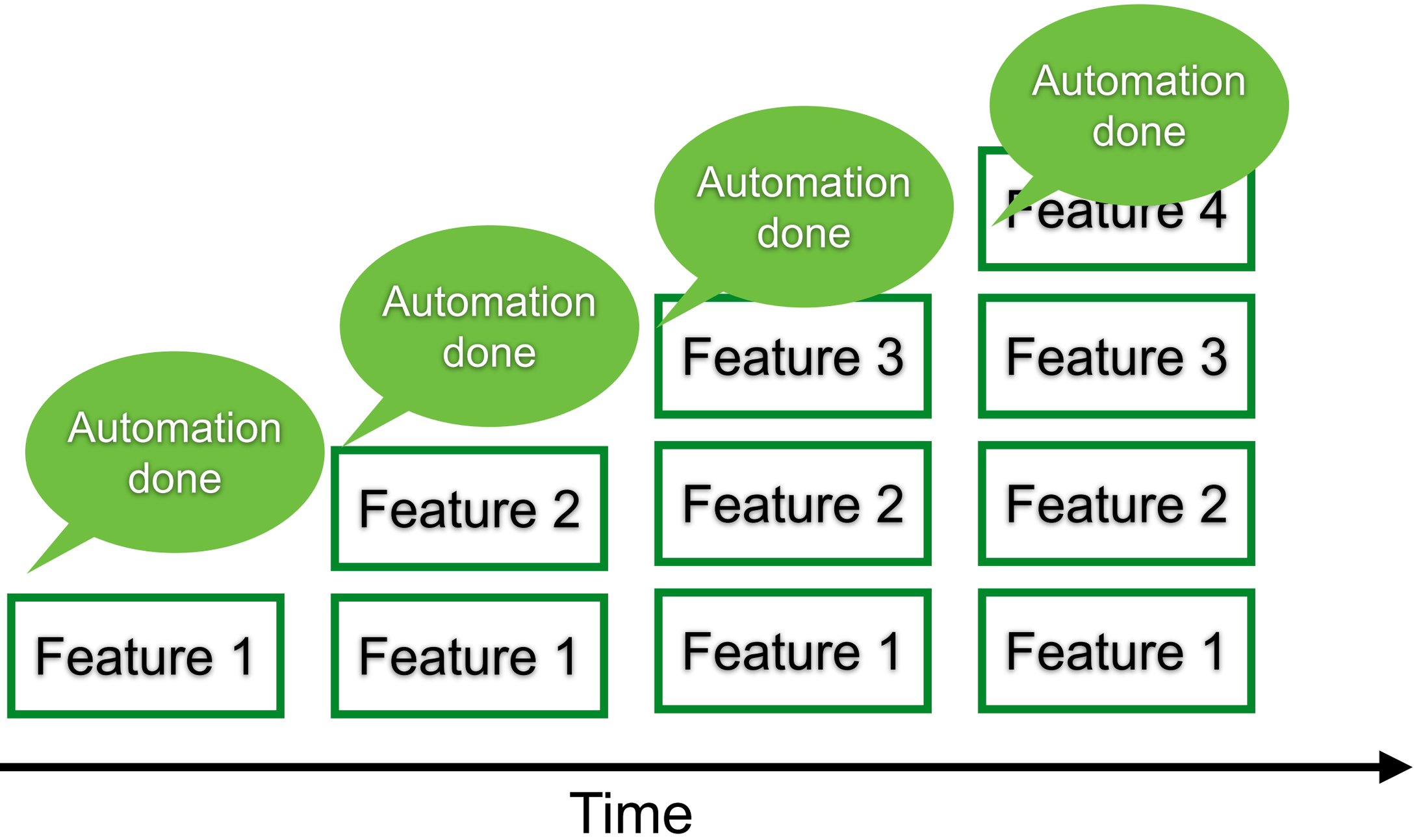
https://slack.engineering/android-ui-automation-part-1-building-trust-de3deb1c5995

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

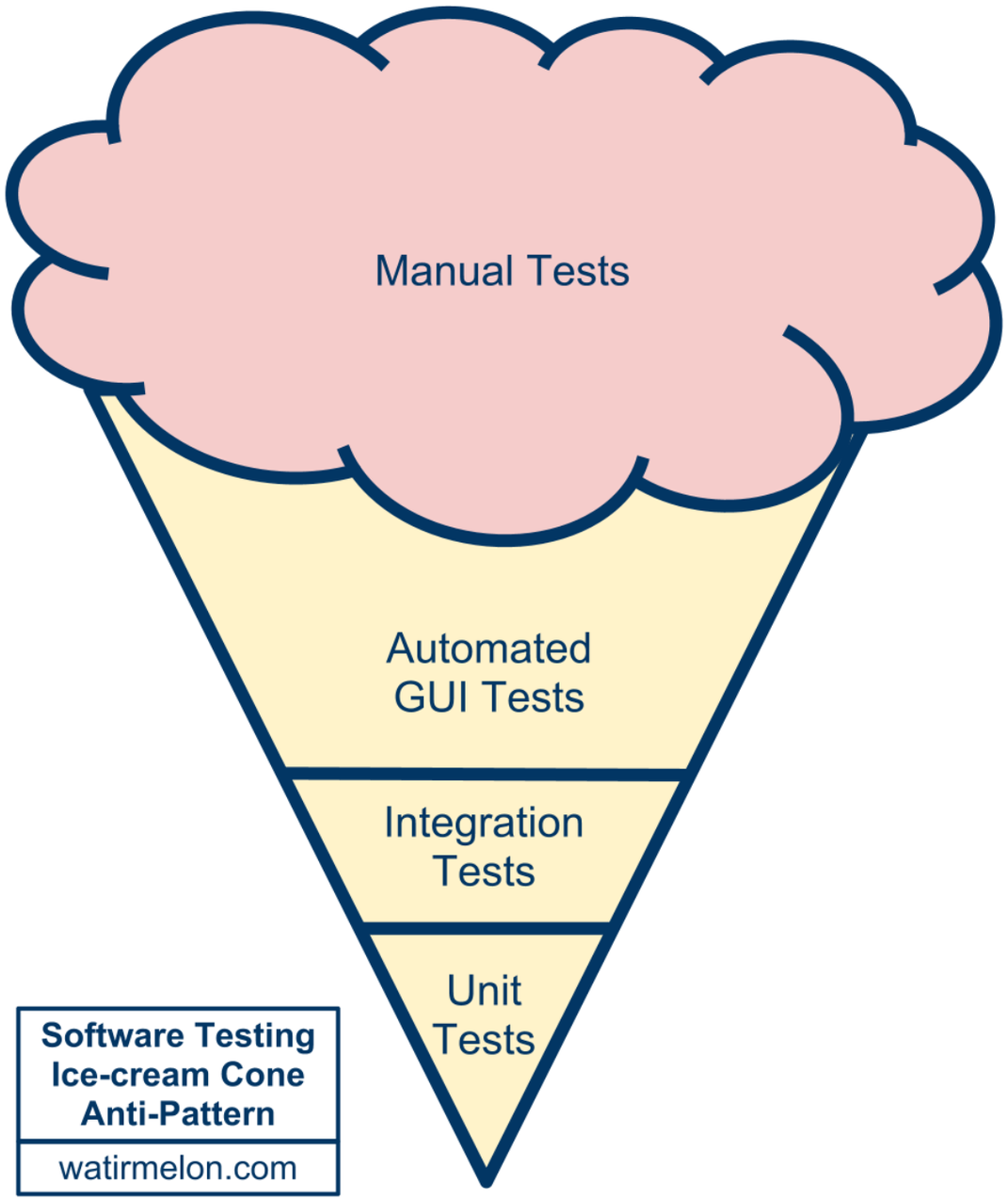
Iterative and incremental process

Done = coded and tested

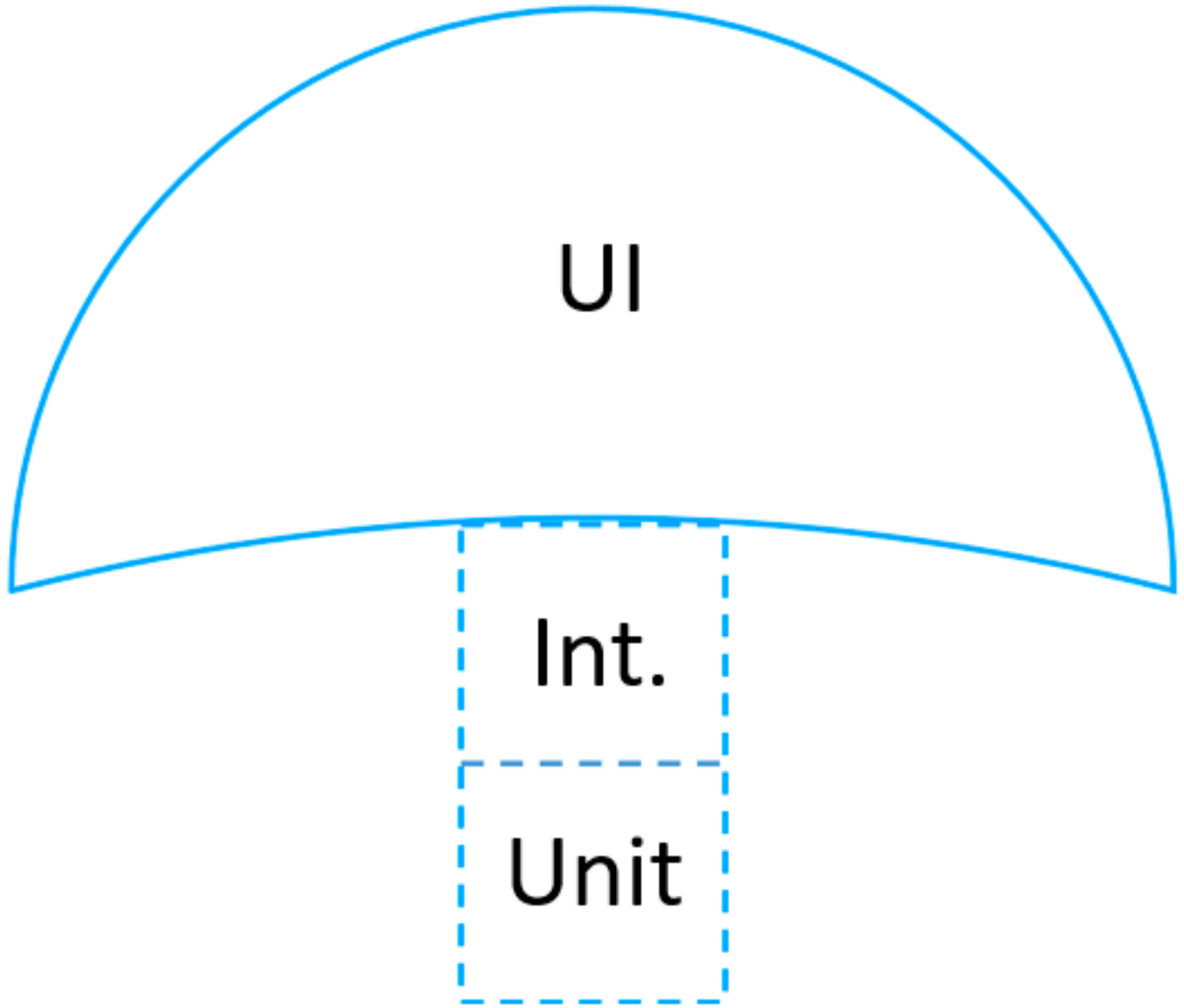


Testing pyramid

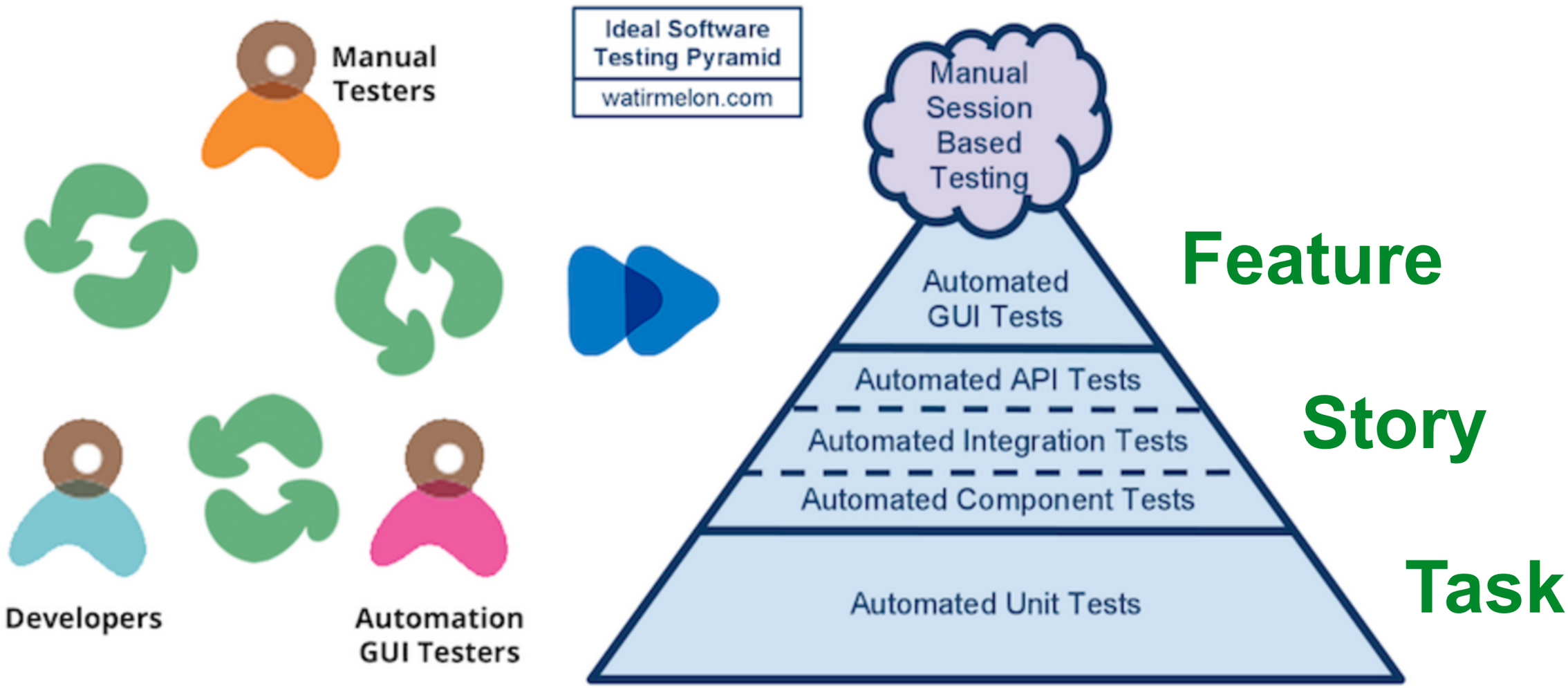
Ice-cream testing



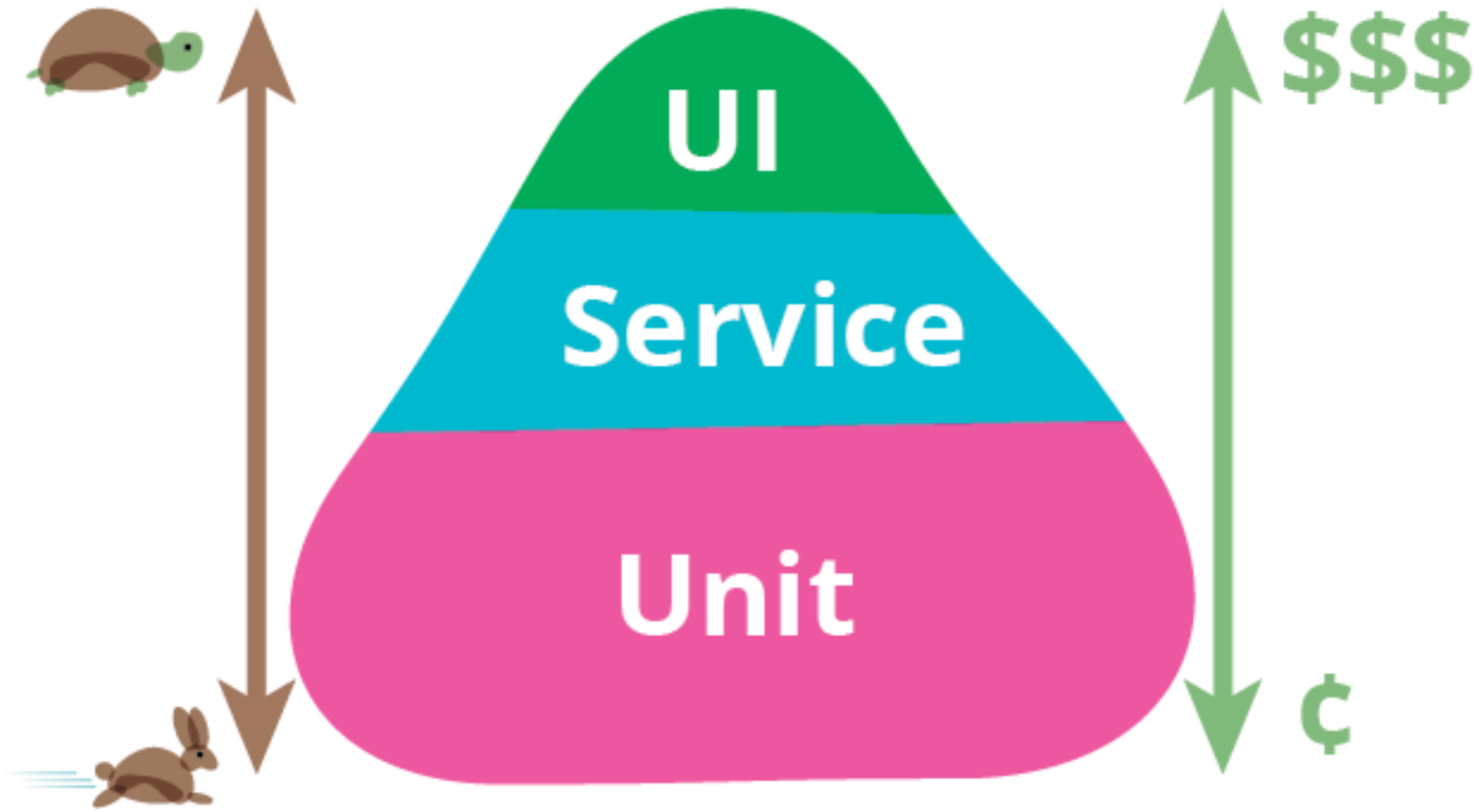
Mashroom testing



Testing Pyramid

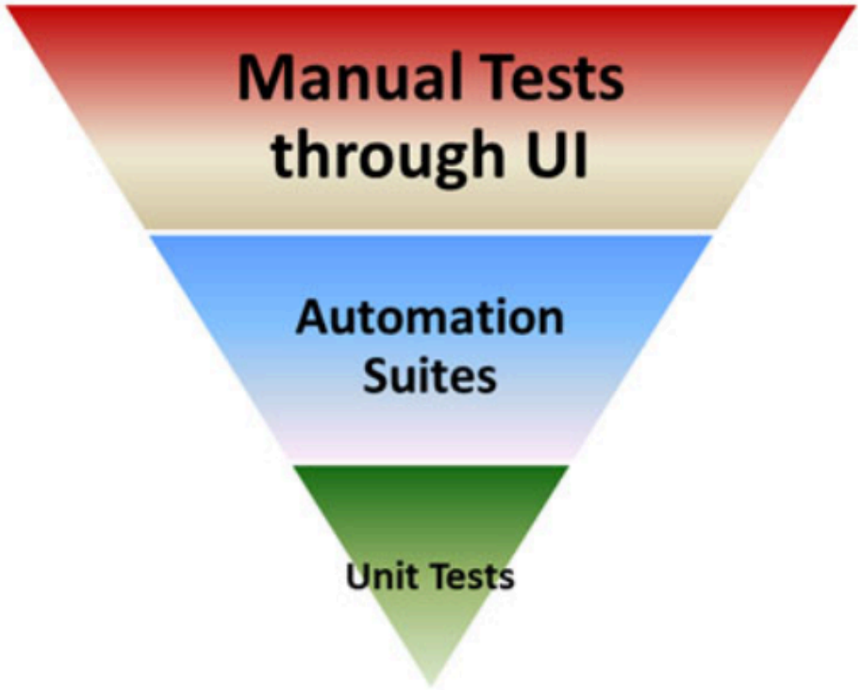


Testing Pyramid

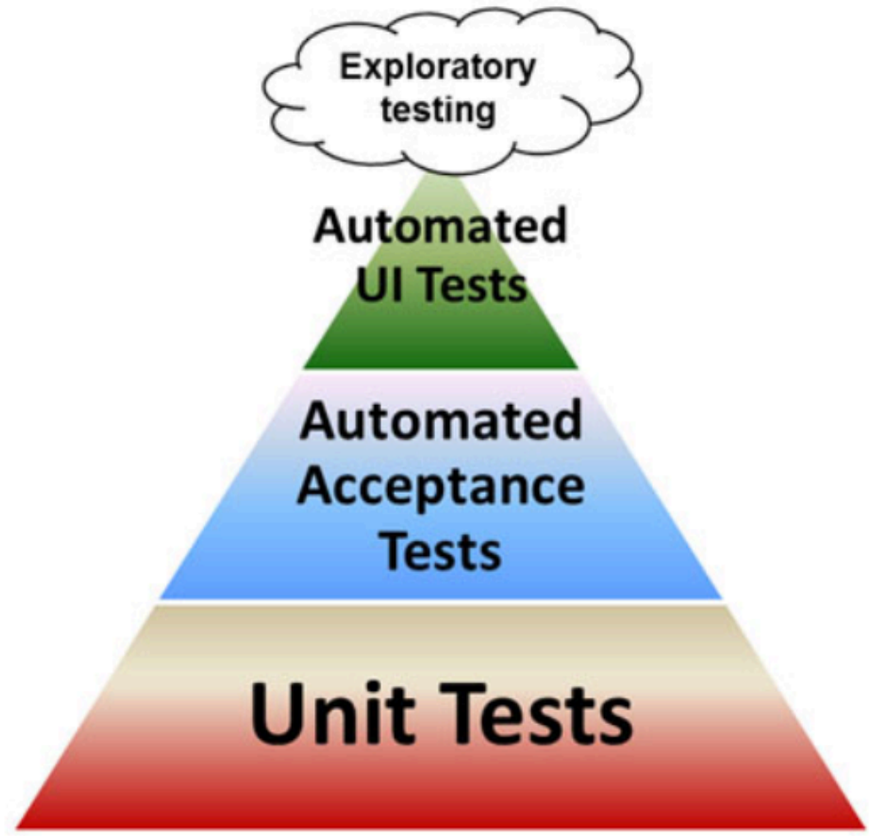


https://martinfowler.com/bliki/TestPyramid.html

Find vs Prevent

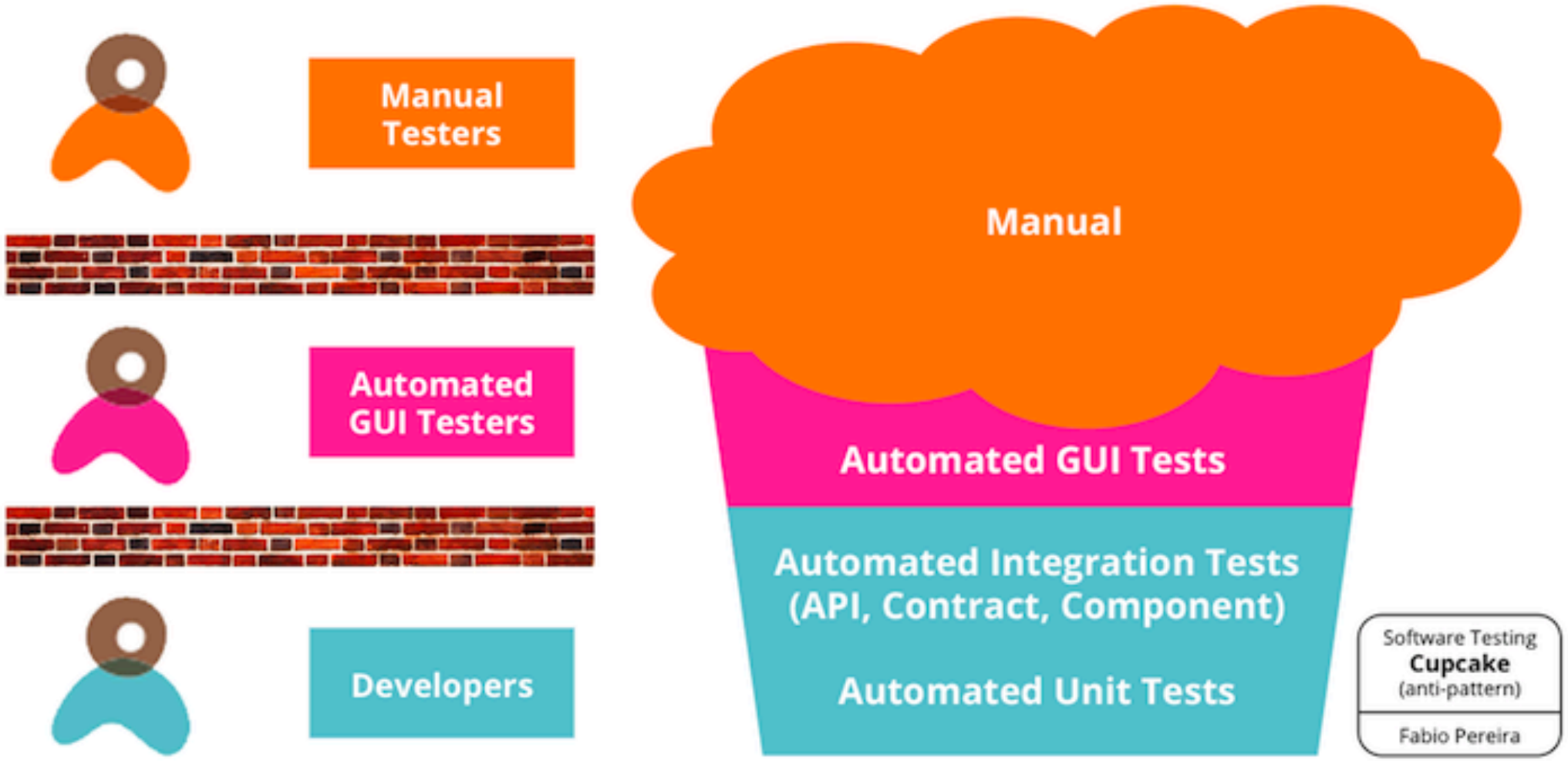


Traditional
(find bugs)



Agile
(prevent bugs)

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

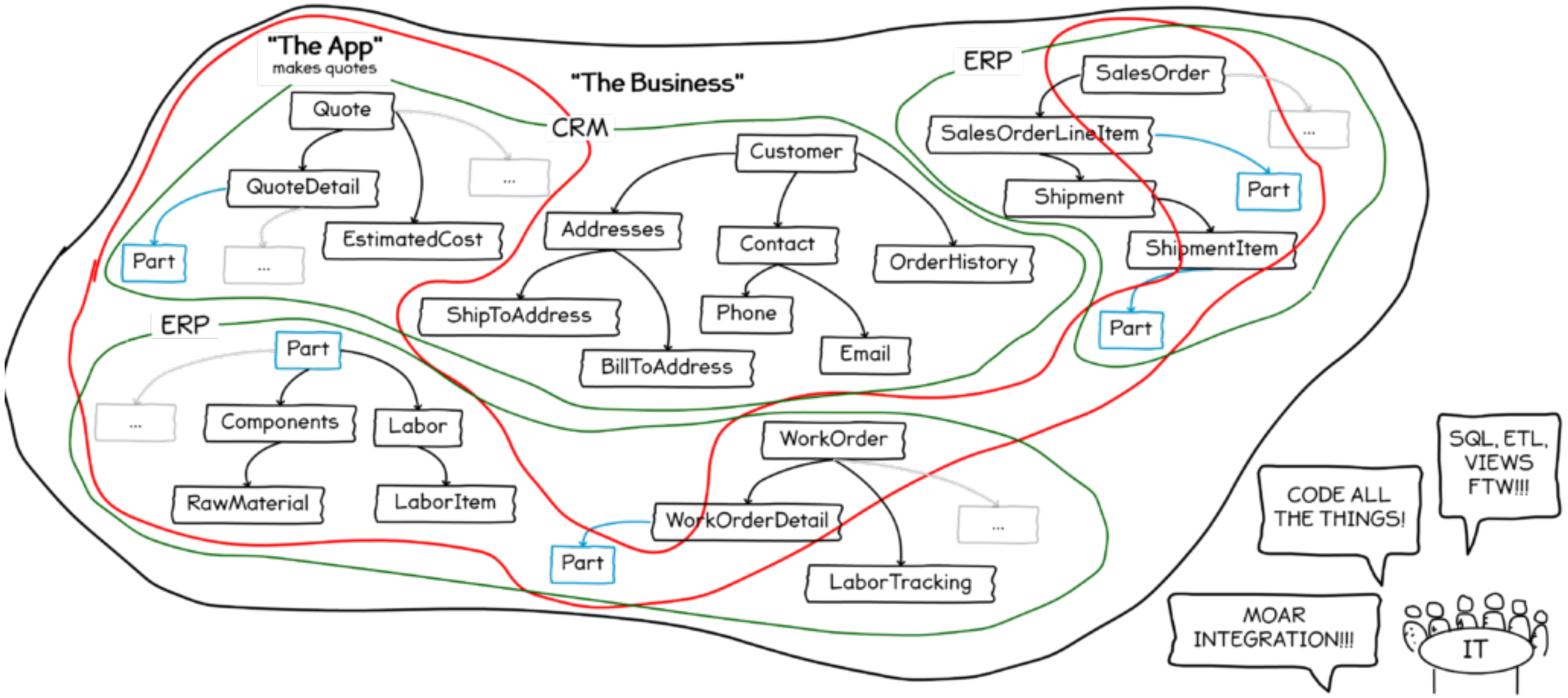
Thinking about automation

Incremental testing
Easy to test
Testable
Fast customer feedback

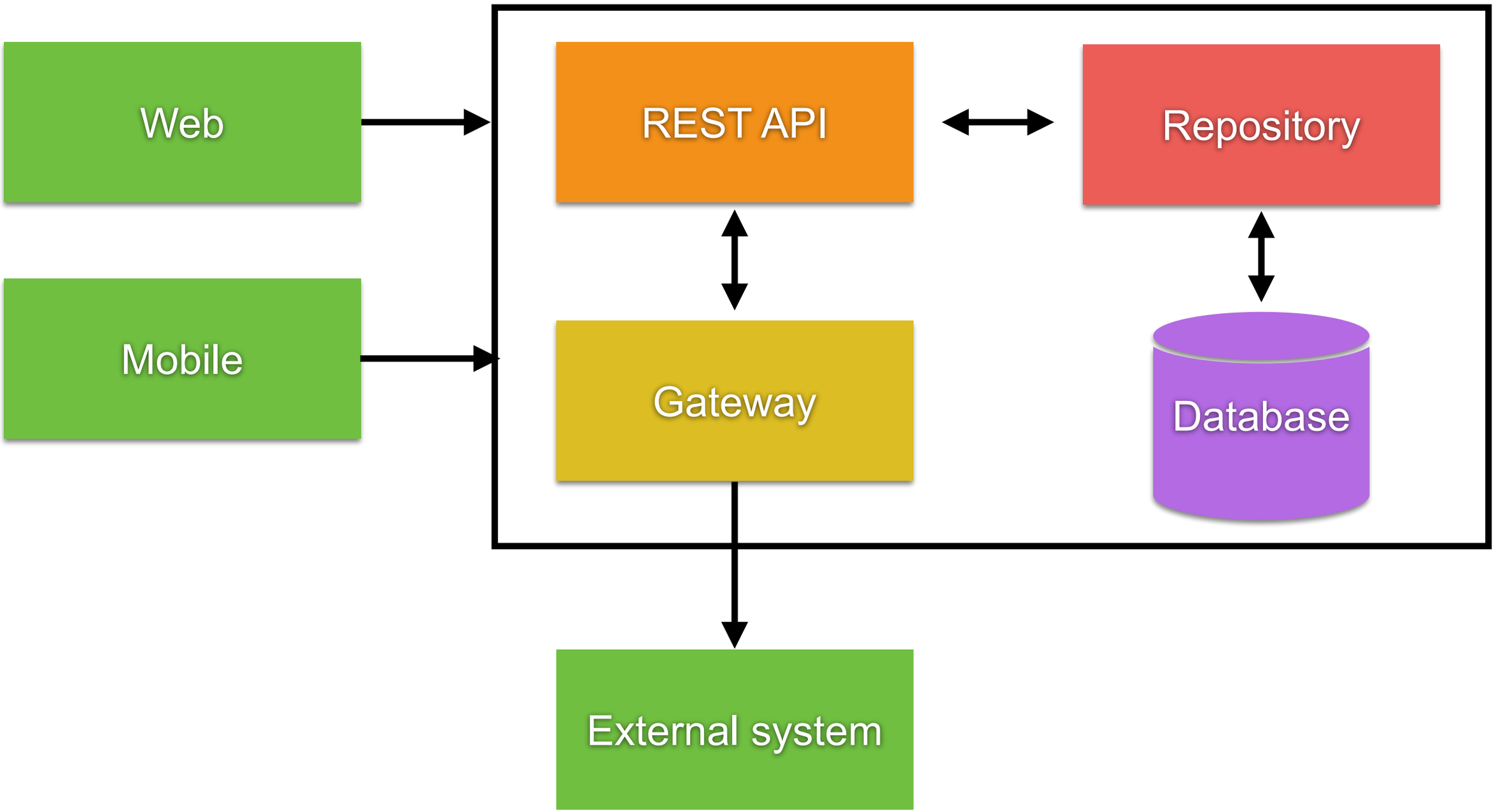
Know your System Architecture

System impacts !!

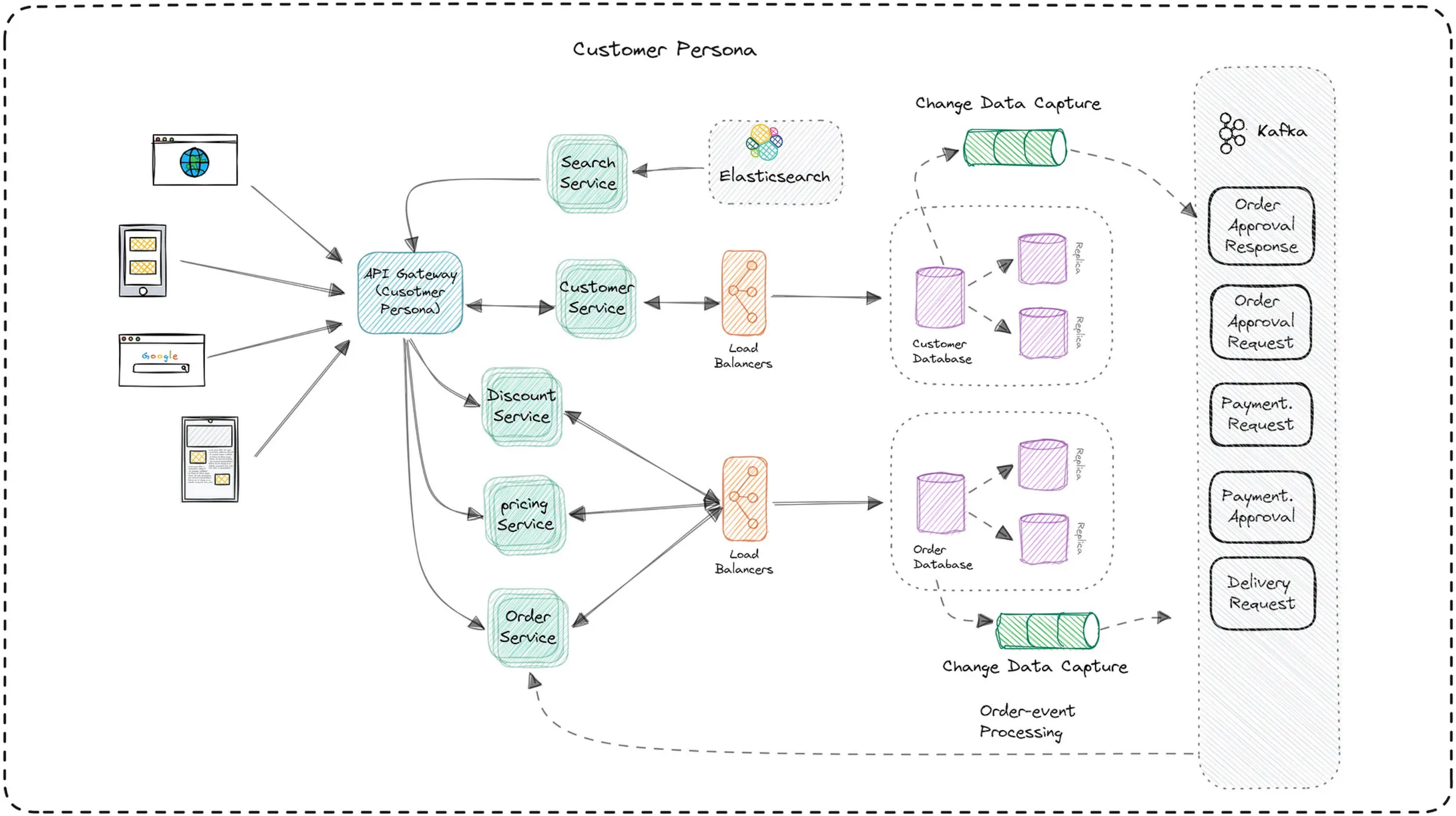
Know your systems



Simple Architecture

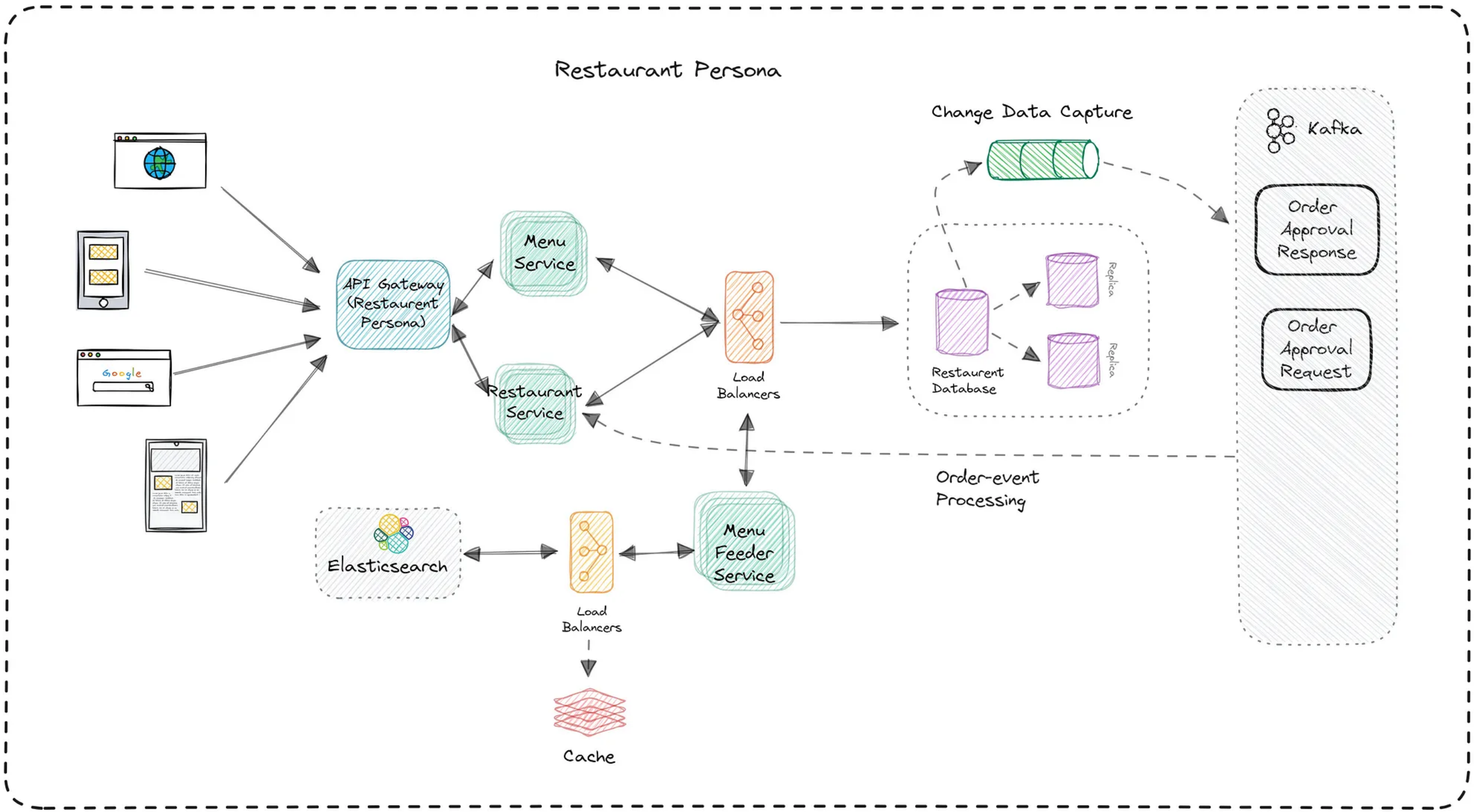


Food Delivery (Customer)



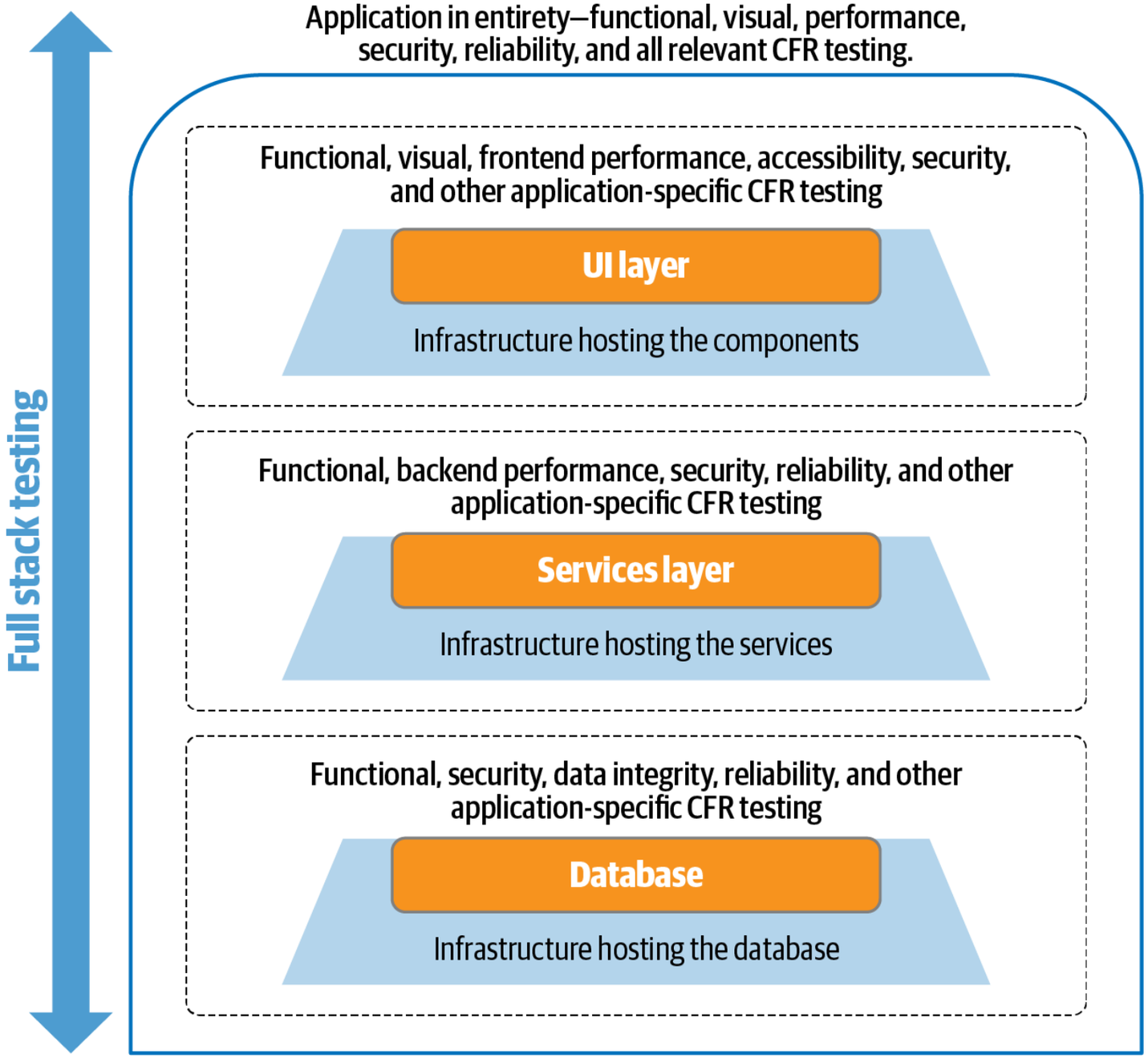
https://medium.com/@systemdesignbychk/system-design-a-deep-dive-into-the-food-ordering-system-f84ae6375ce3

Food Delivery (Restaurant)

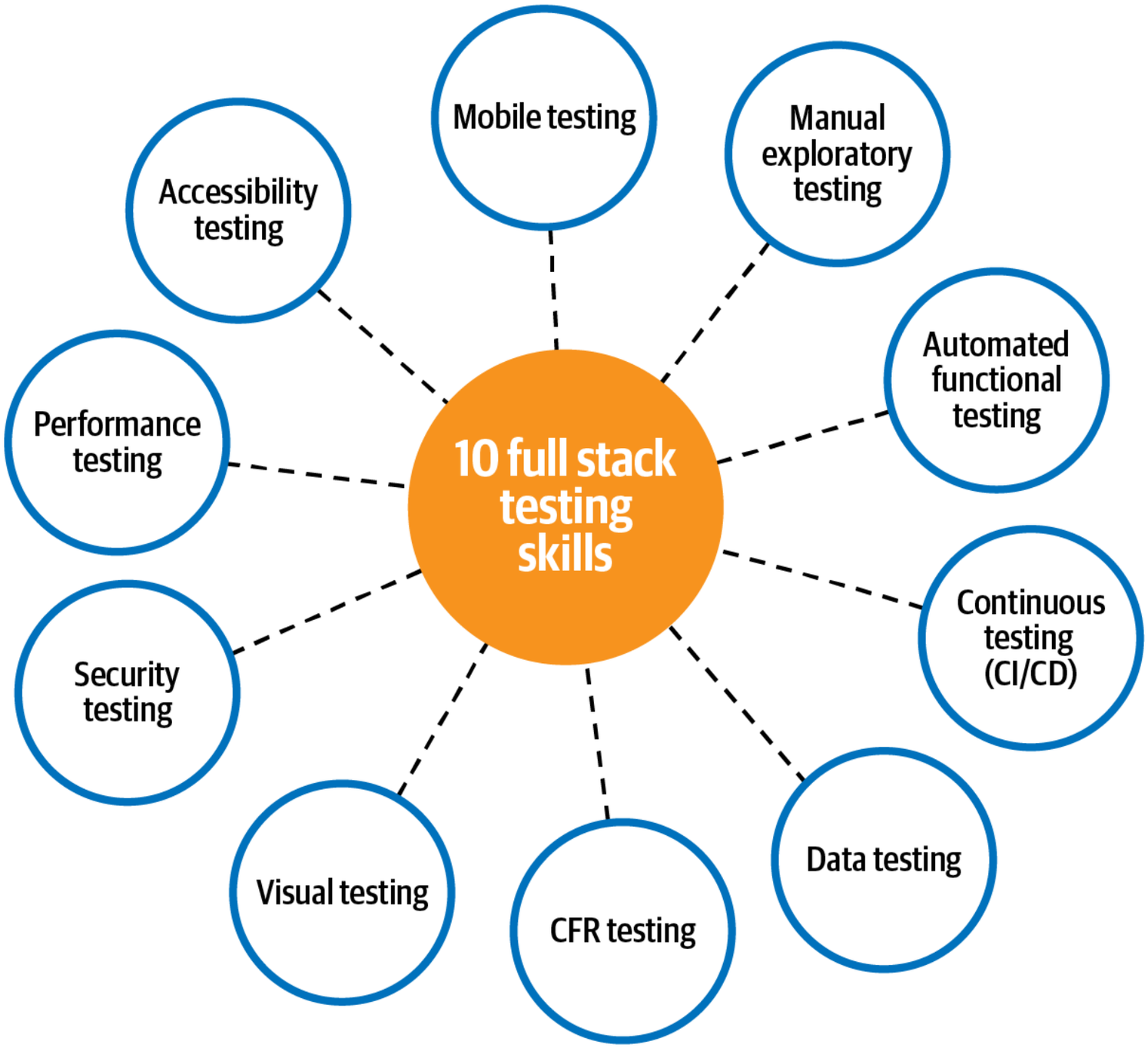


https://medium.com/@systemdesignbychk/system-design-a-deep-dive-into-the-food-ordering-system-f84ae6375ce3

Full Stack Testing

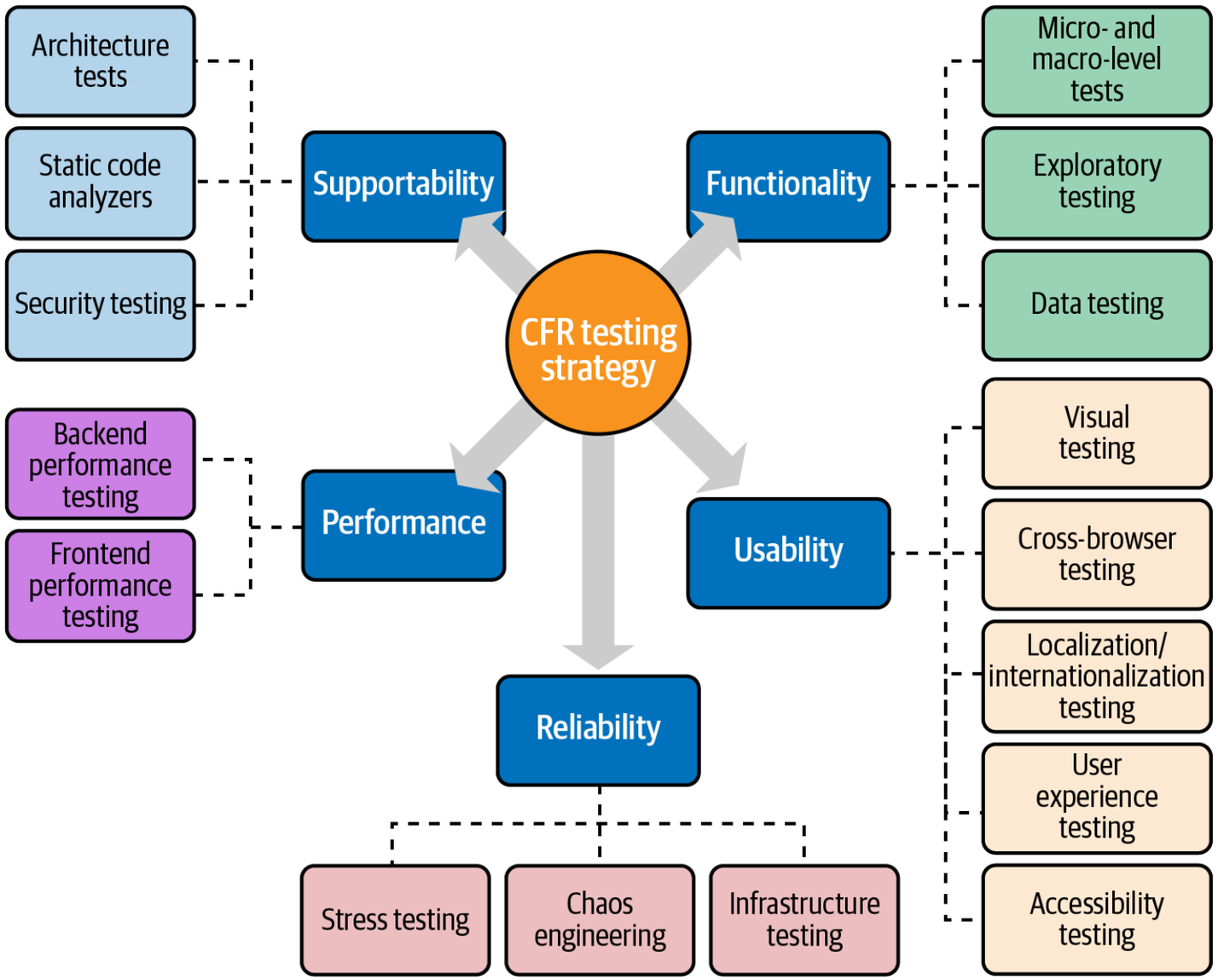


Full Stack Testing Skills



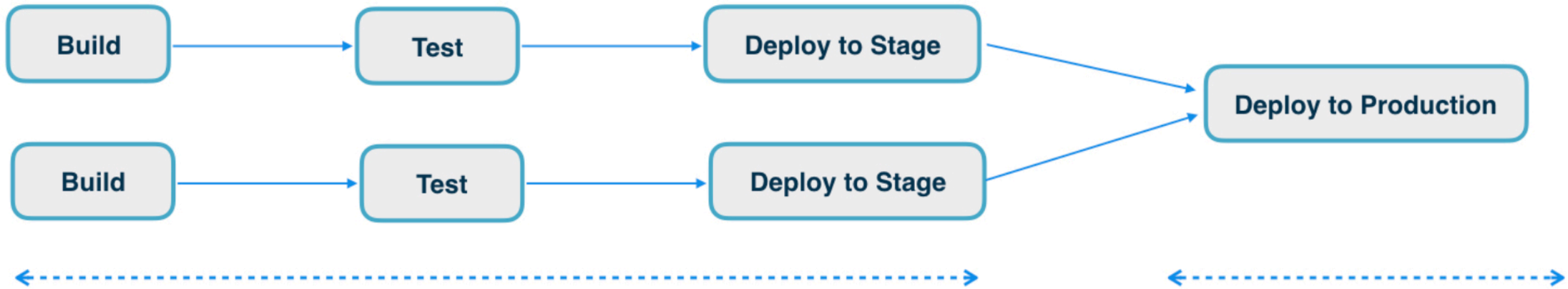
CFR testing

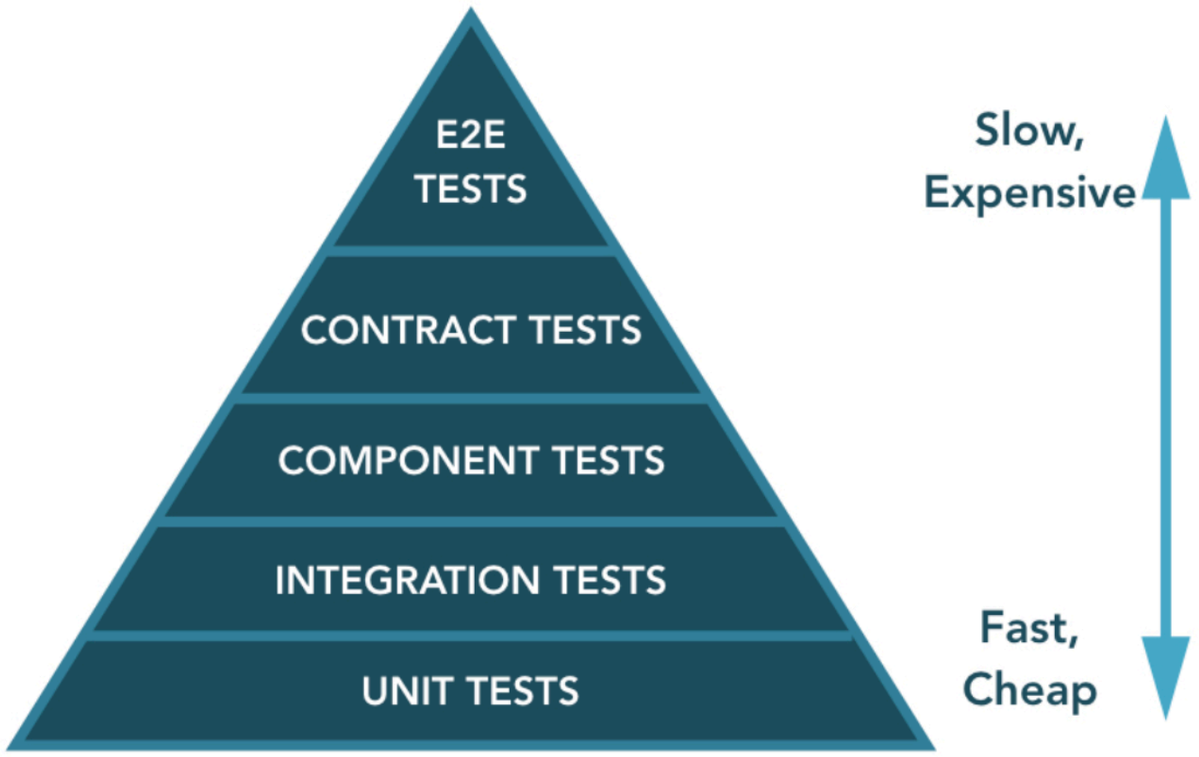
Cross Functional Requirement



Test Strategies

Test strategy



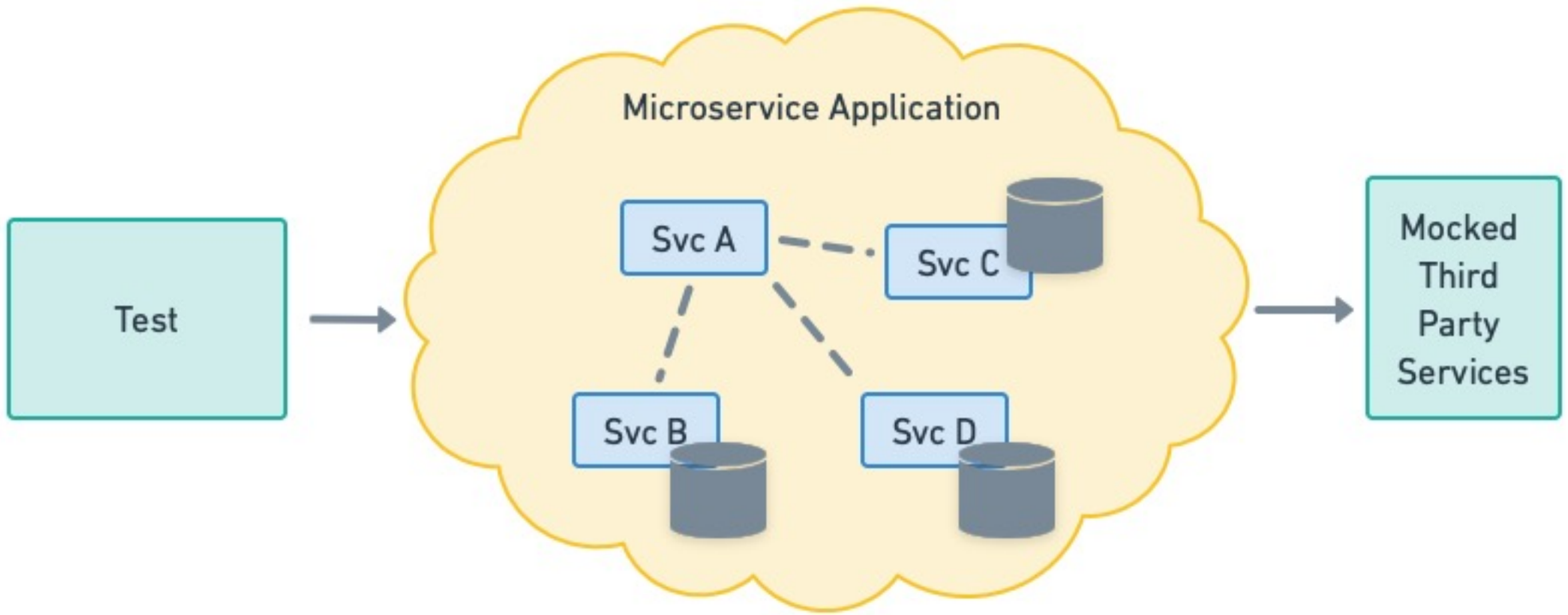


Monitoring
Distributed Tracing
Fault-injection Testing

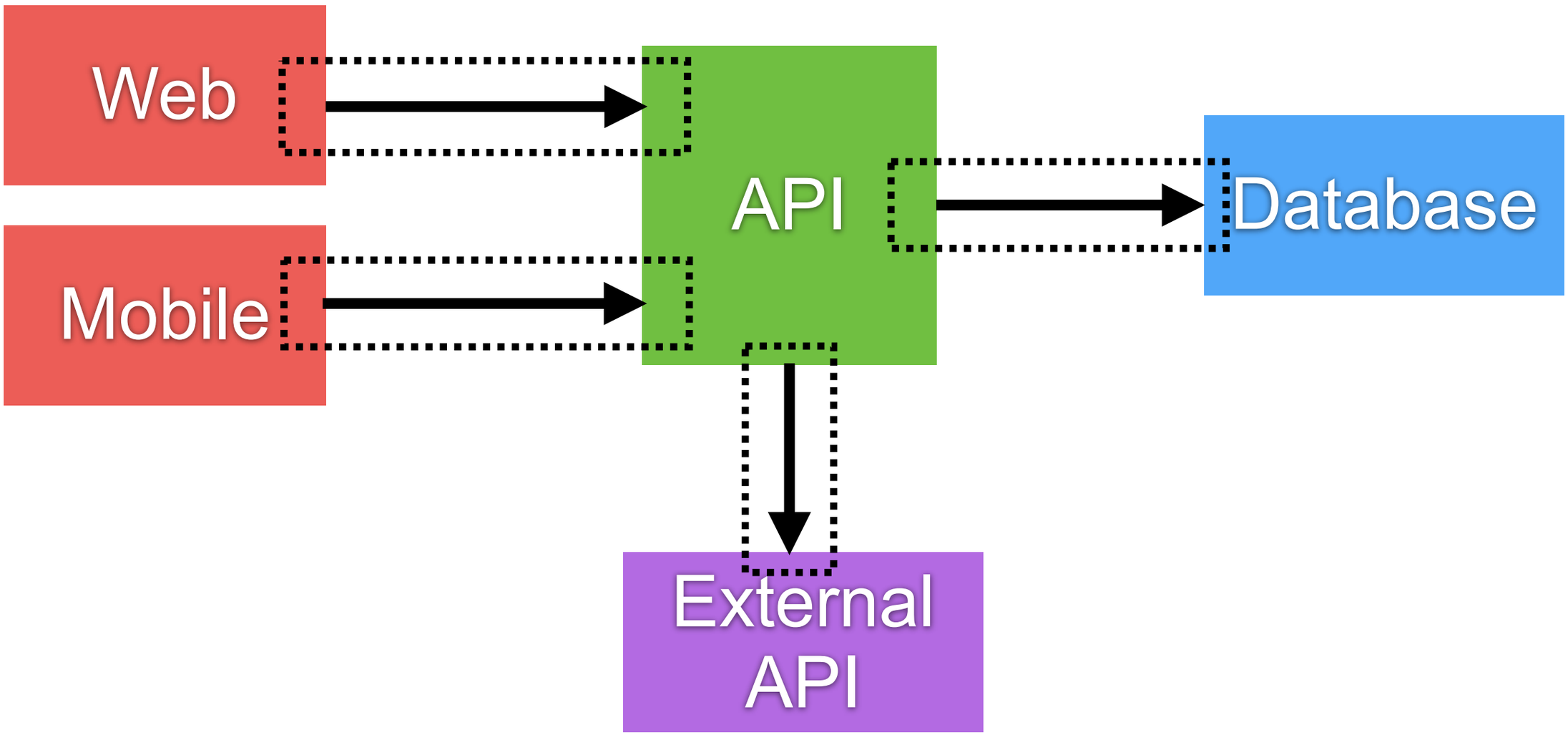
Canary Deployments
Blue-Green Deployments
A/B Testing

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

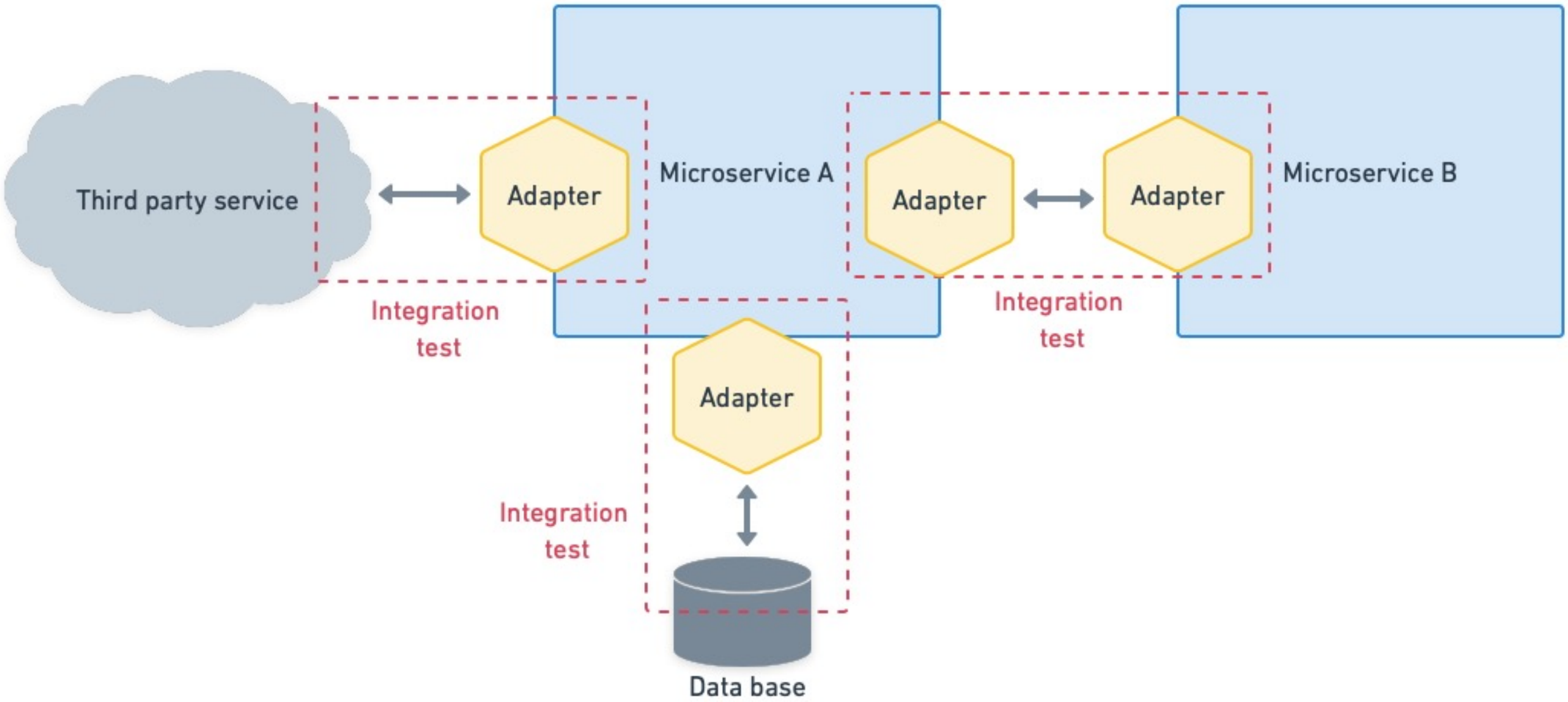
End-to-End testing



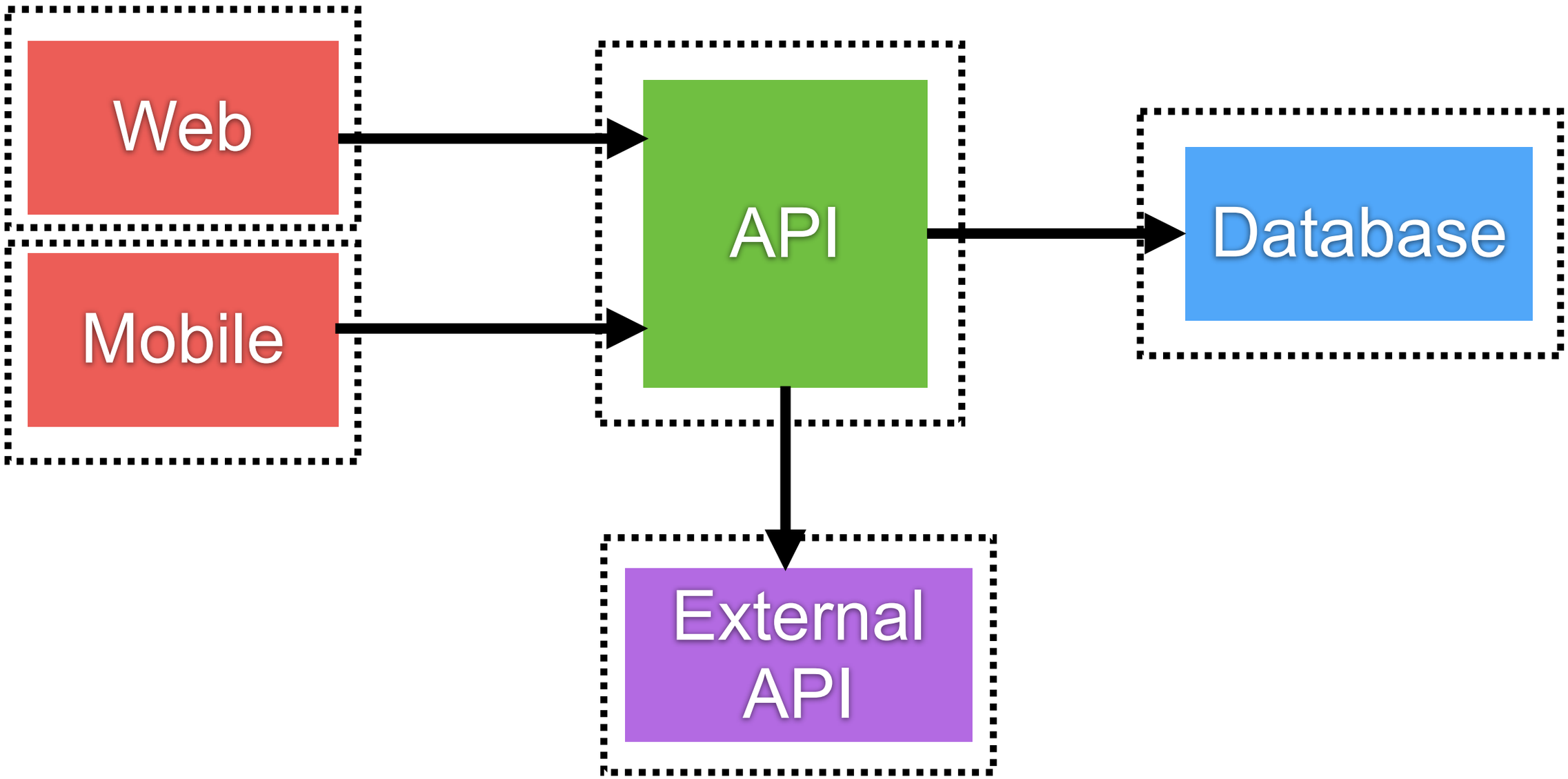
Integration testing



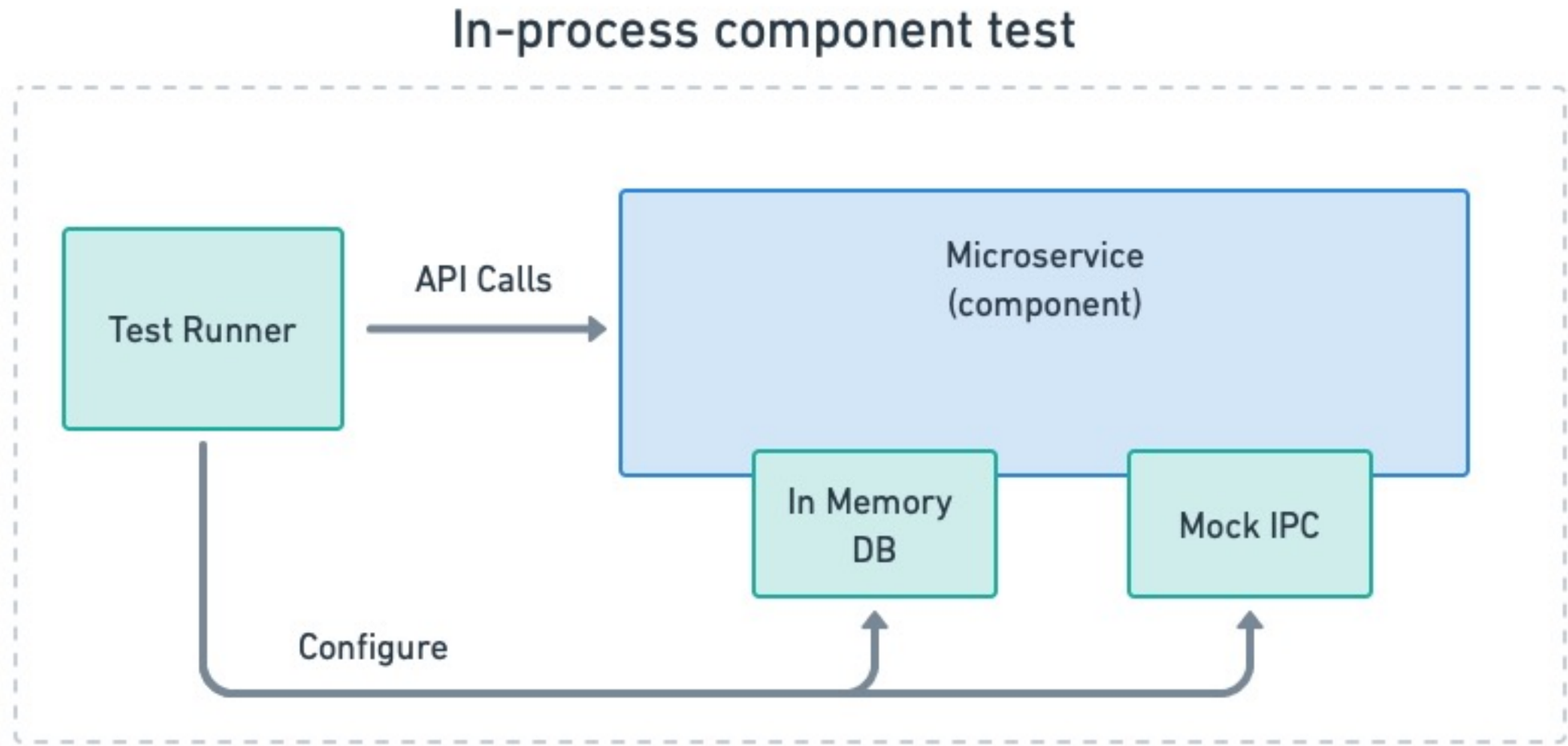
Integration testing



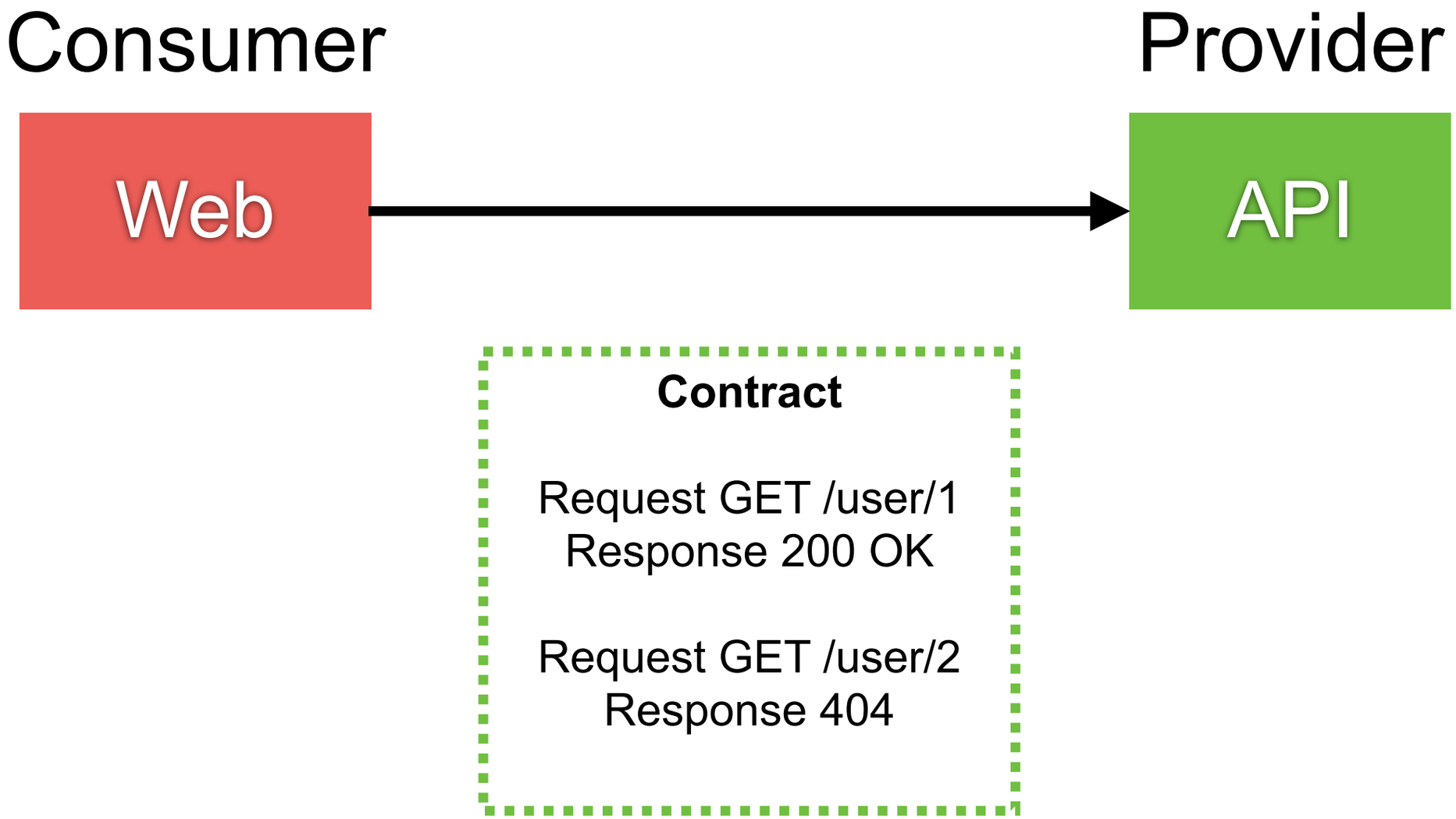
Service component testing



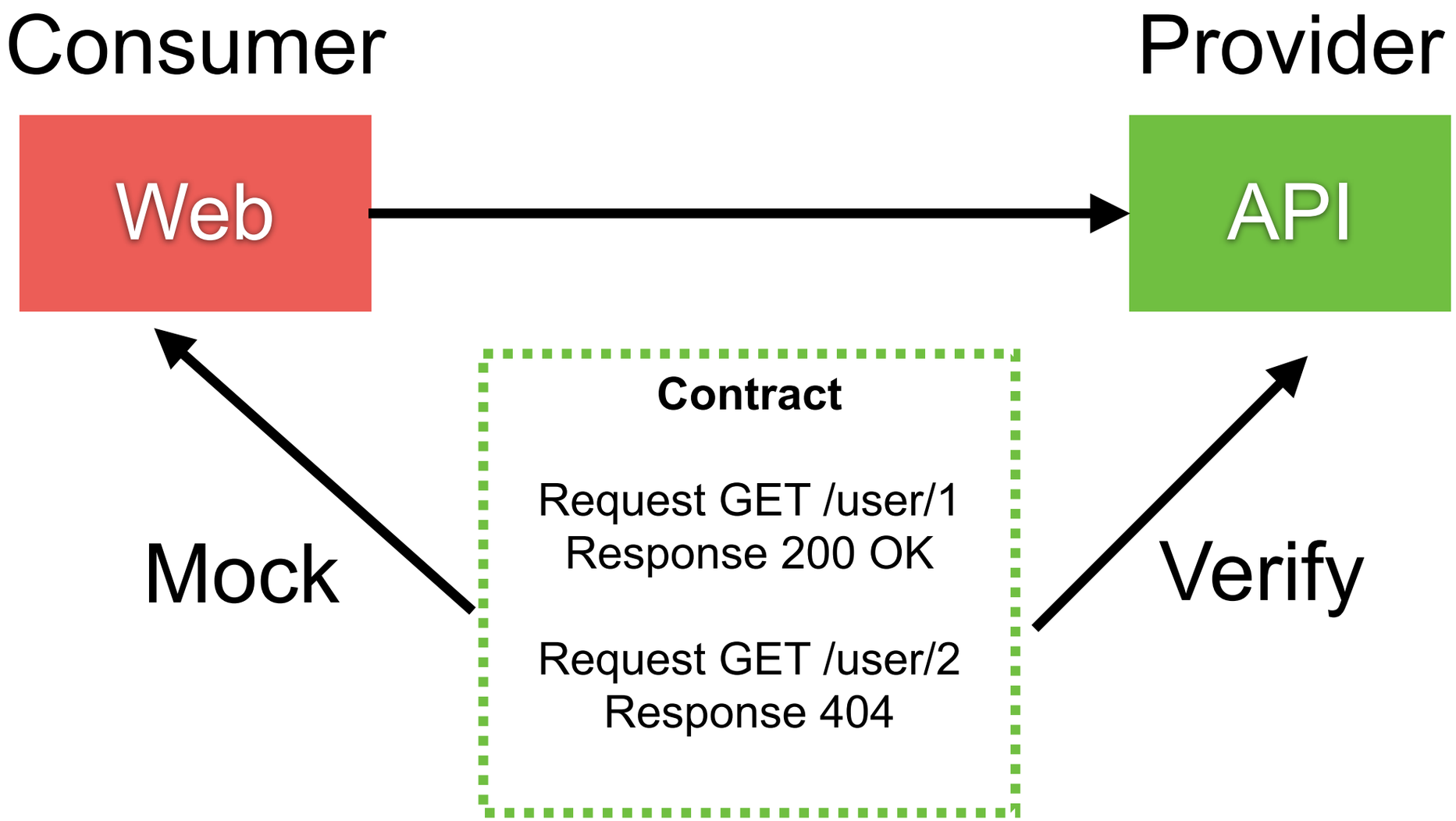
Service component testing



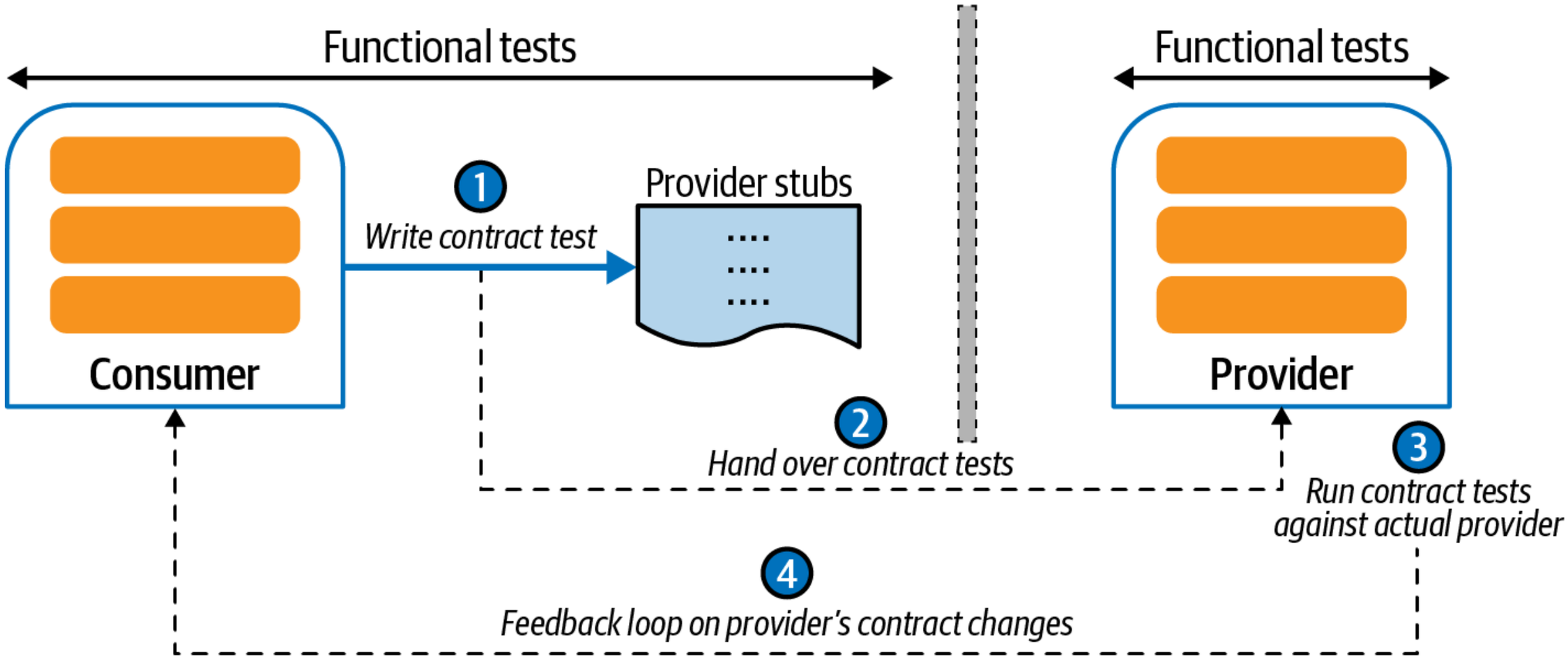
Contract testing



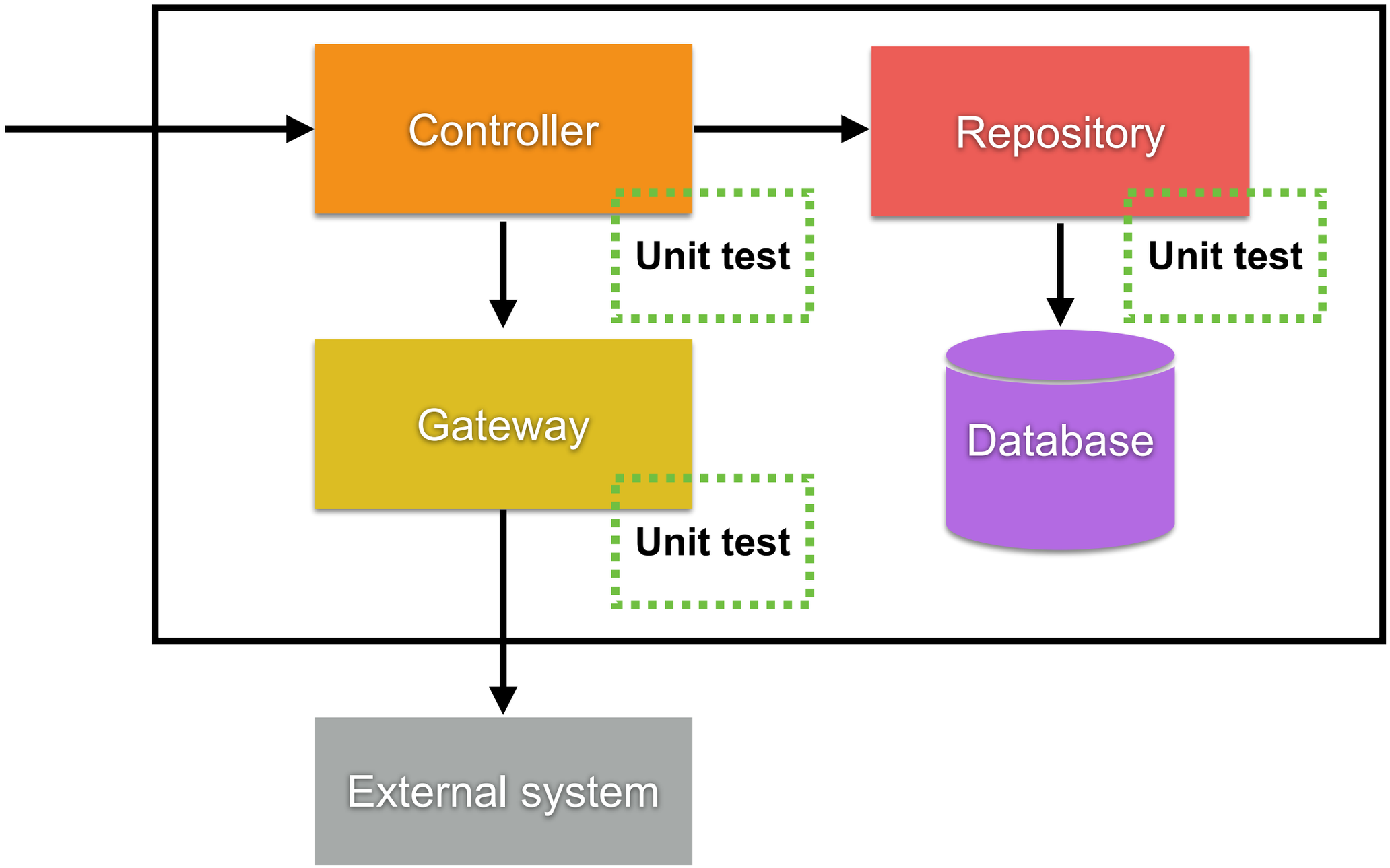
Contract testing



Contract testing



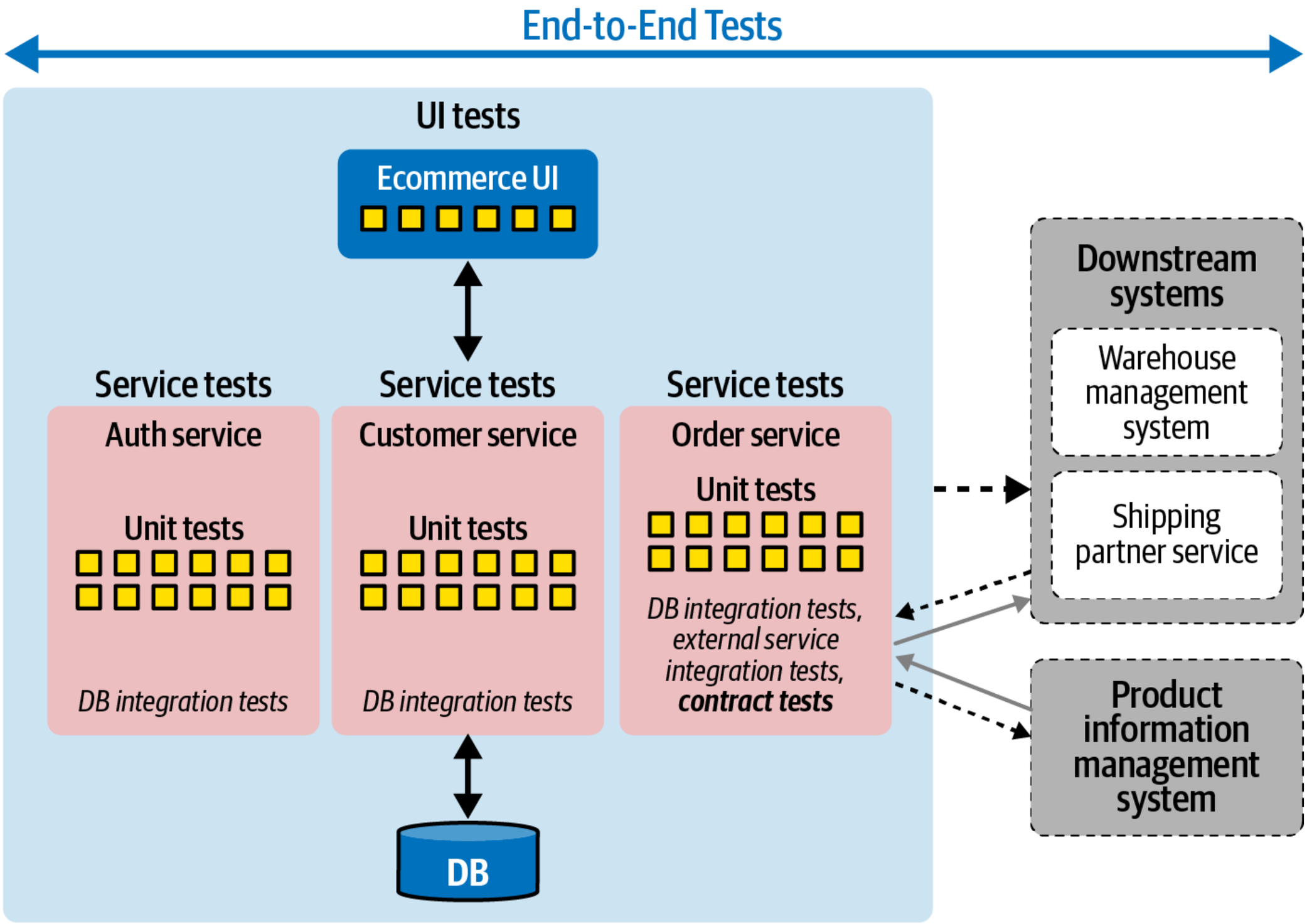
Unit testing



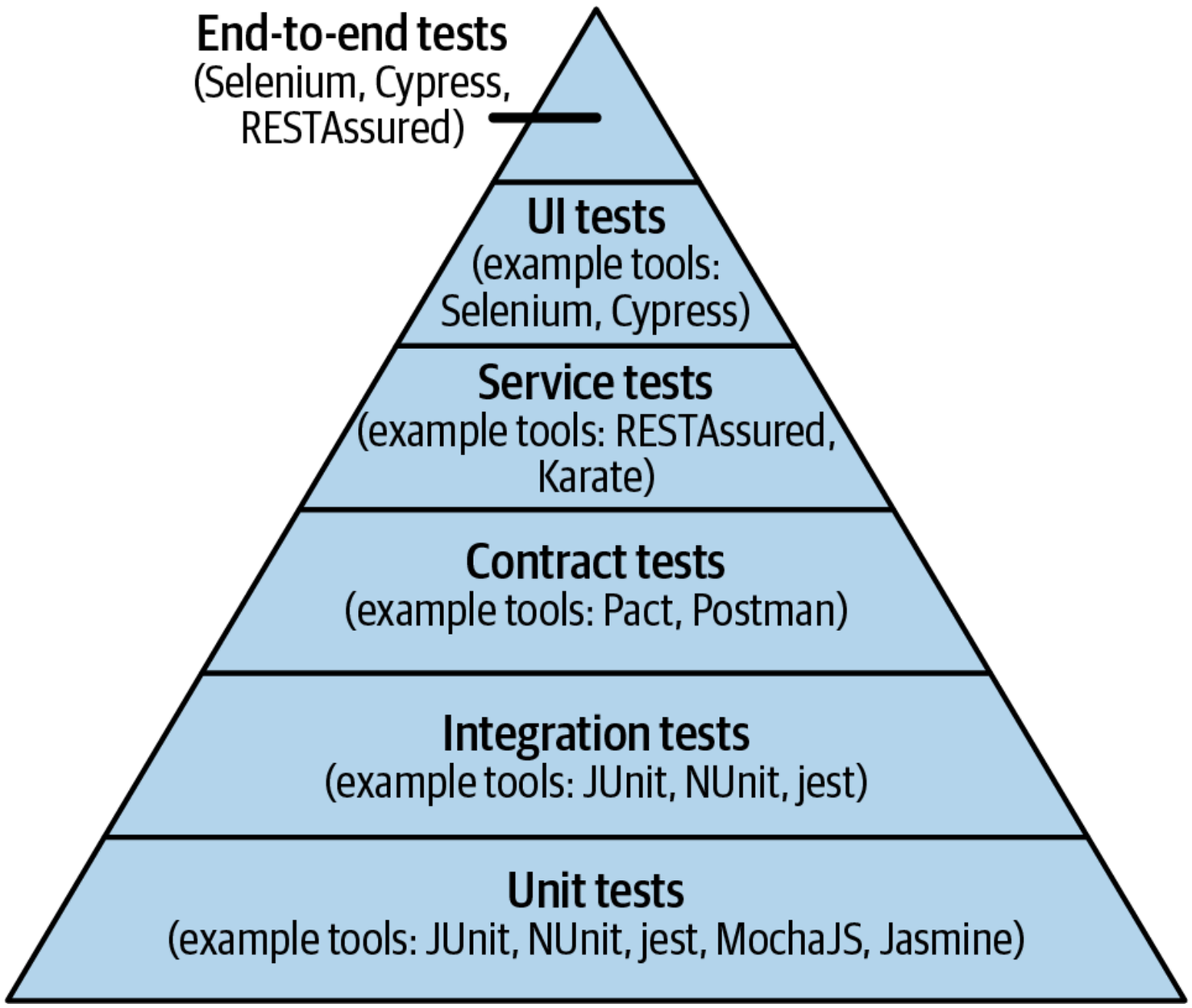
Test Double

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

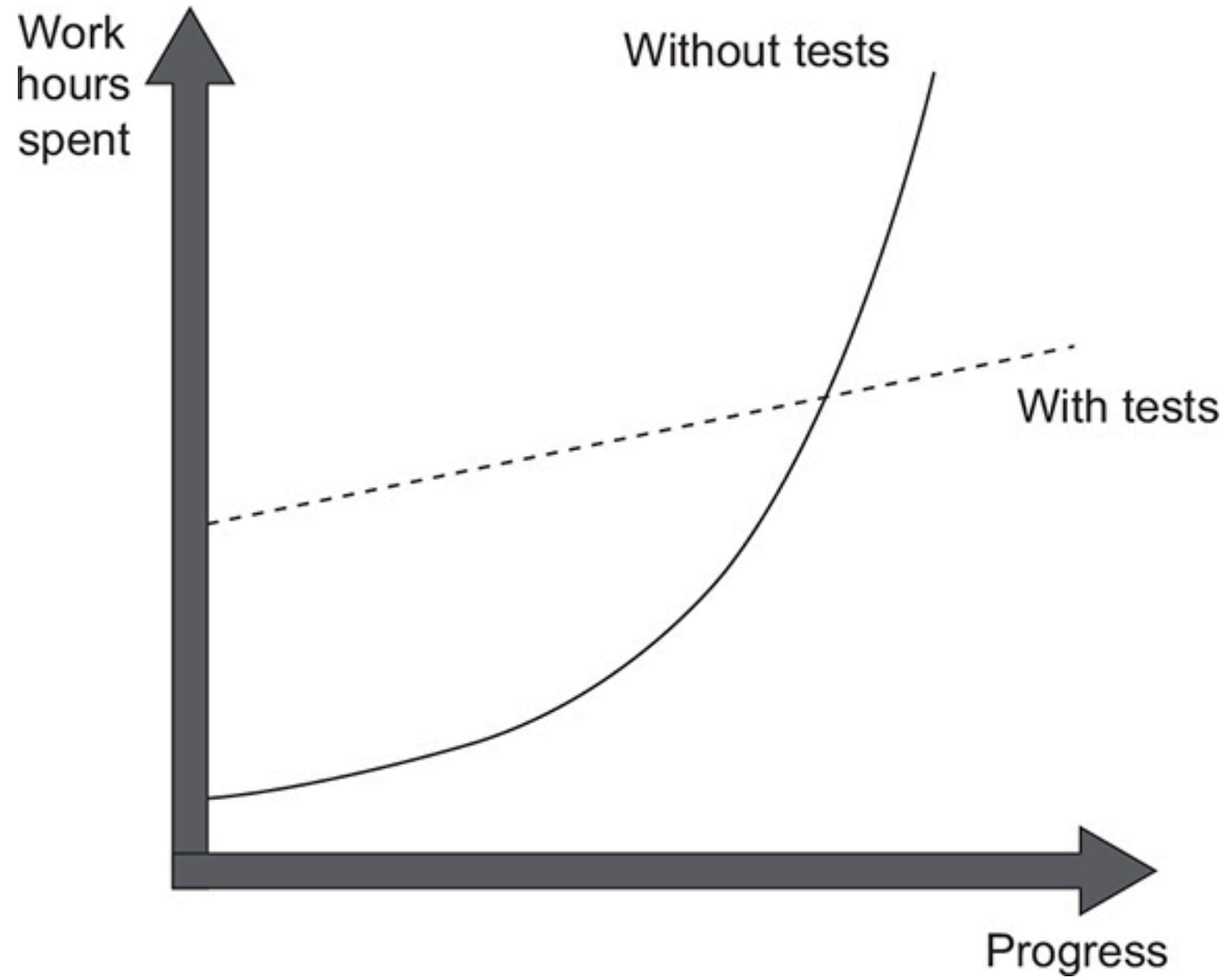
Fast	Isolate	Repeat
Self-verify	Timely	Understand

Respect your tests

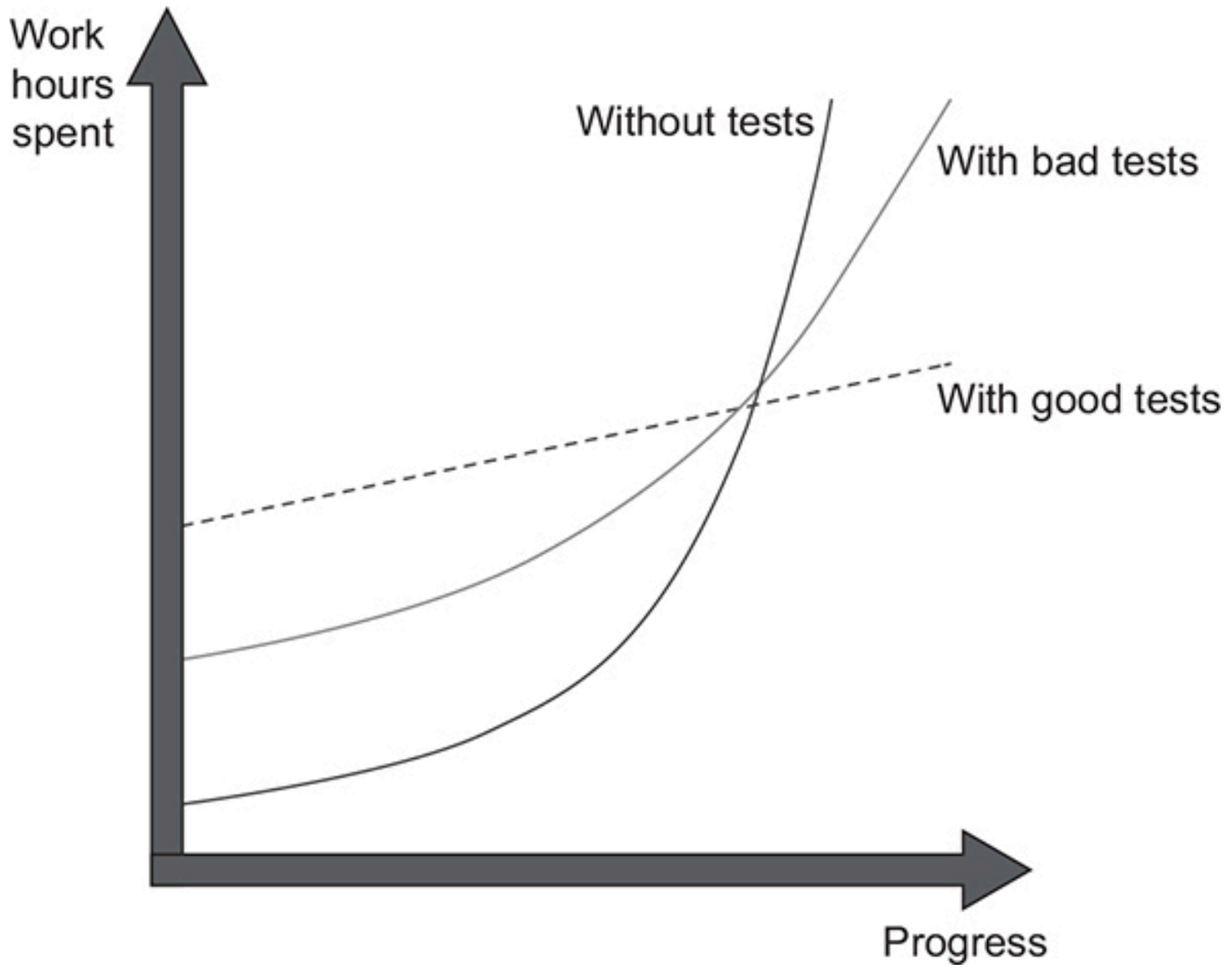
Don’t ignore it if it fails
Fix the code or fix the test
Always refactor or improve

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 !!

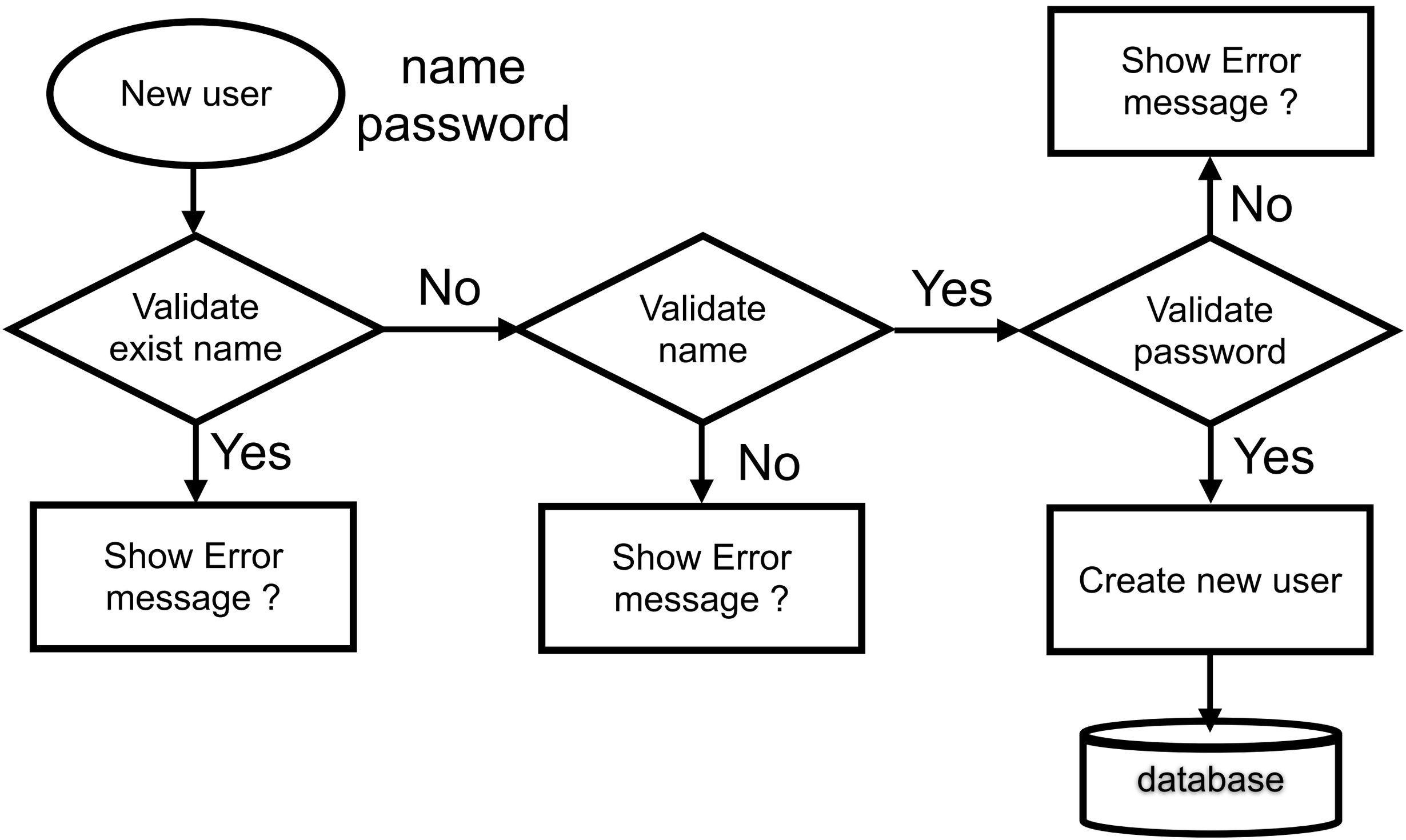
As a new user,

I want to create an account with
User name and password

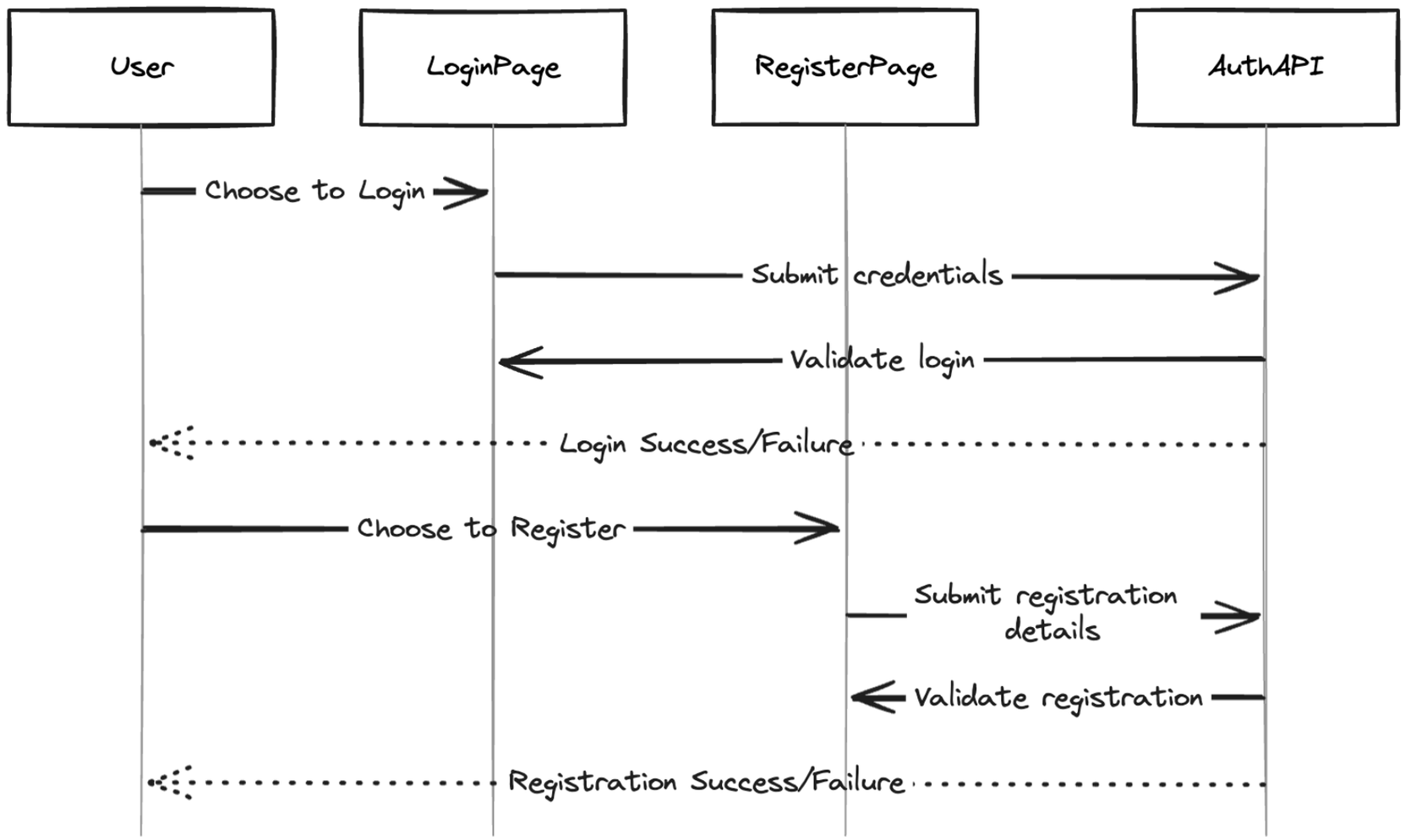
So that only I can access my information

Design your Test cases ?

Flow ?

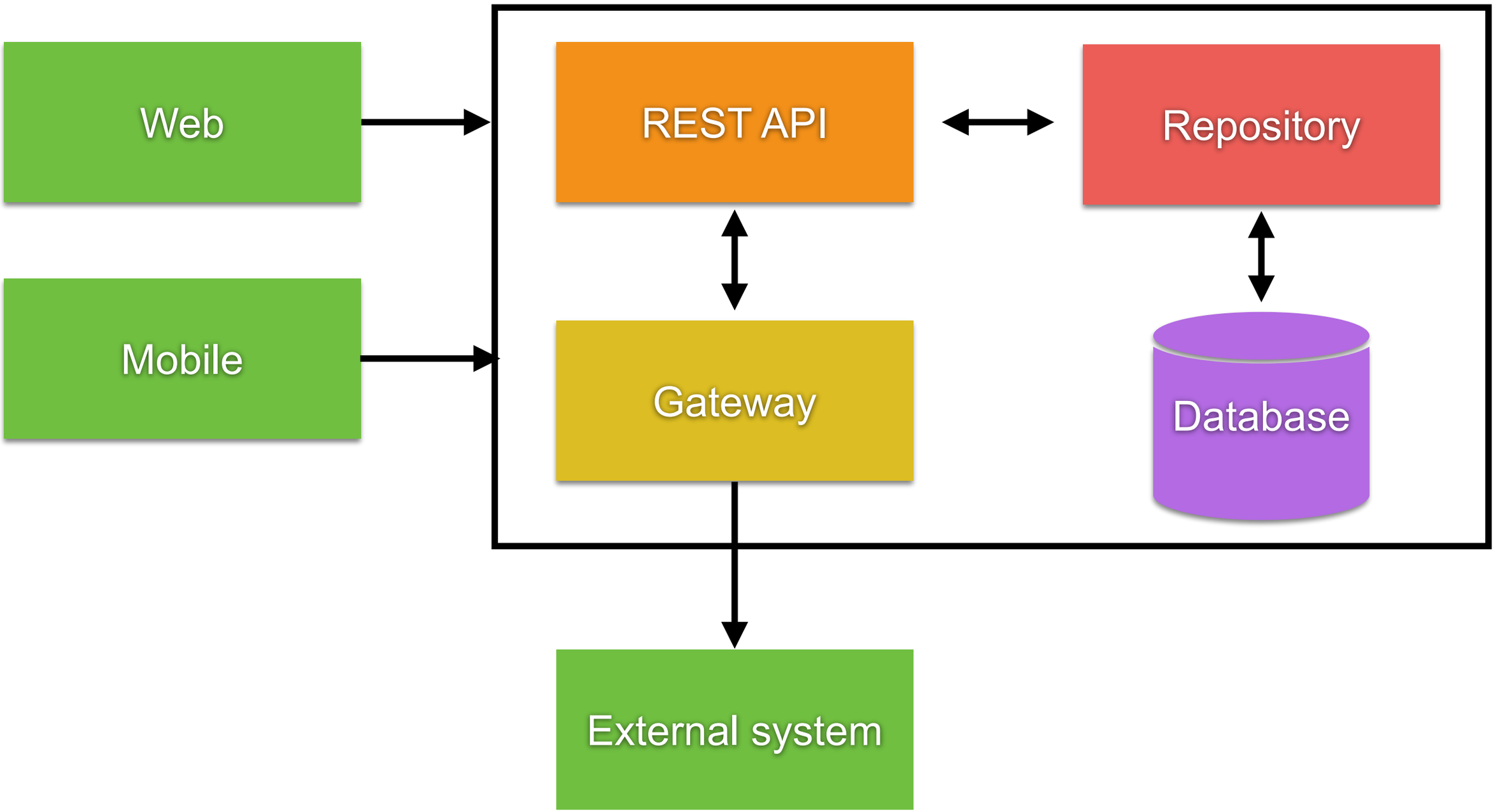


Generate Flow with AI

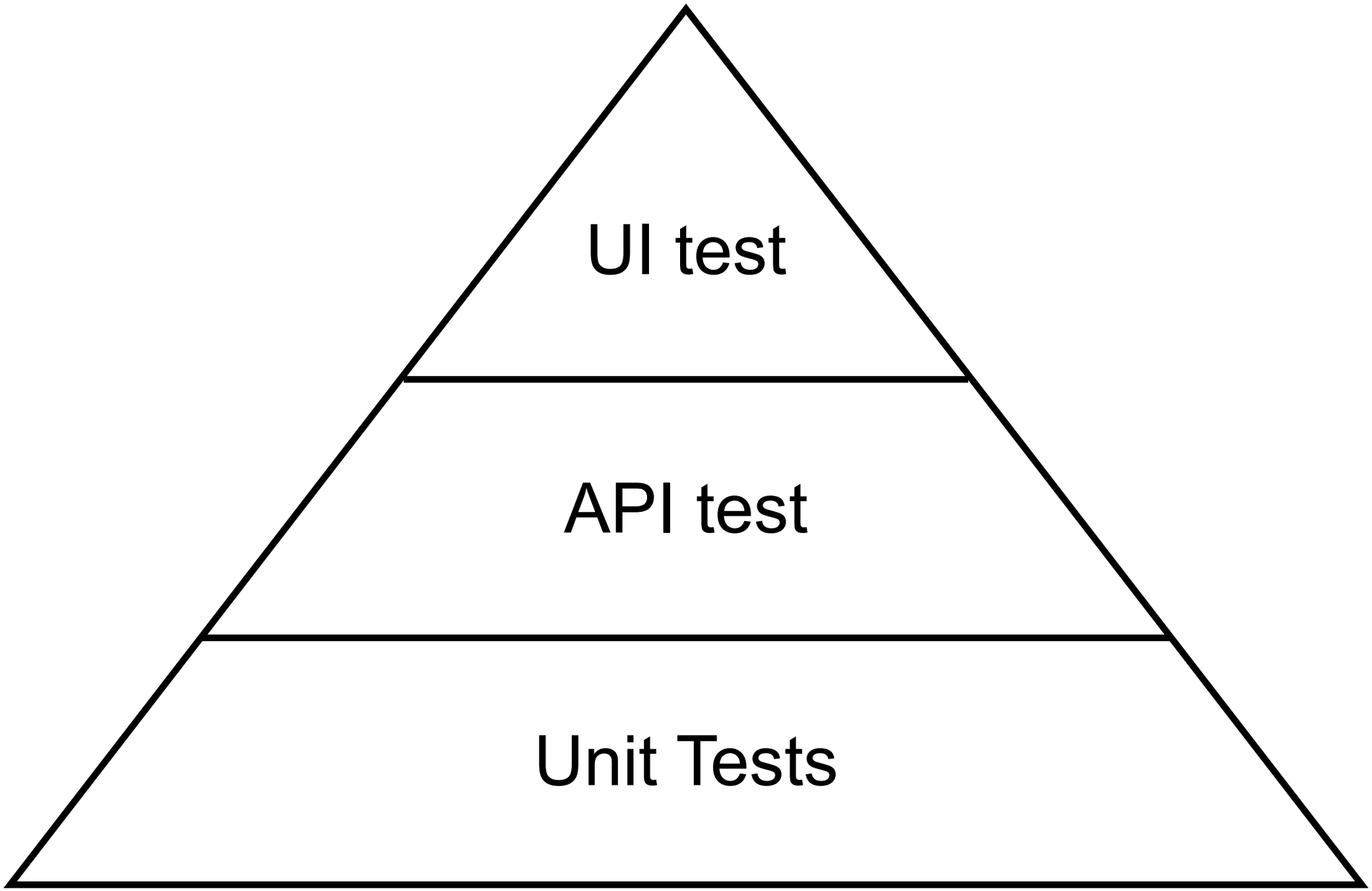


https://excalidraw.com/

Architecture



Test cases with Test Level ?

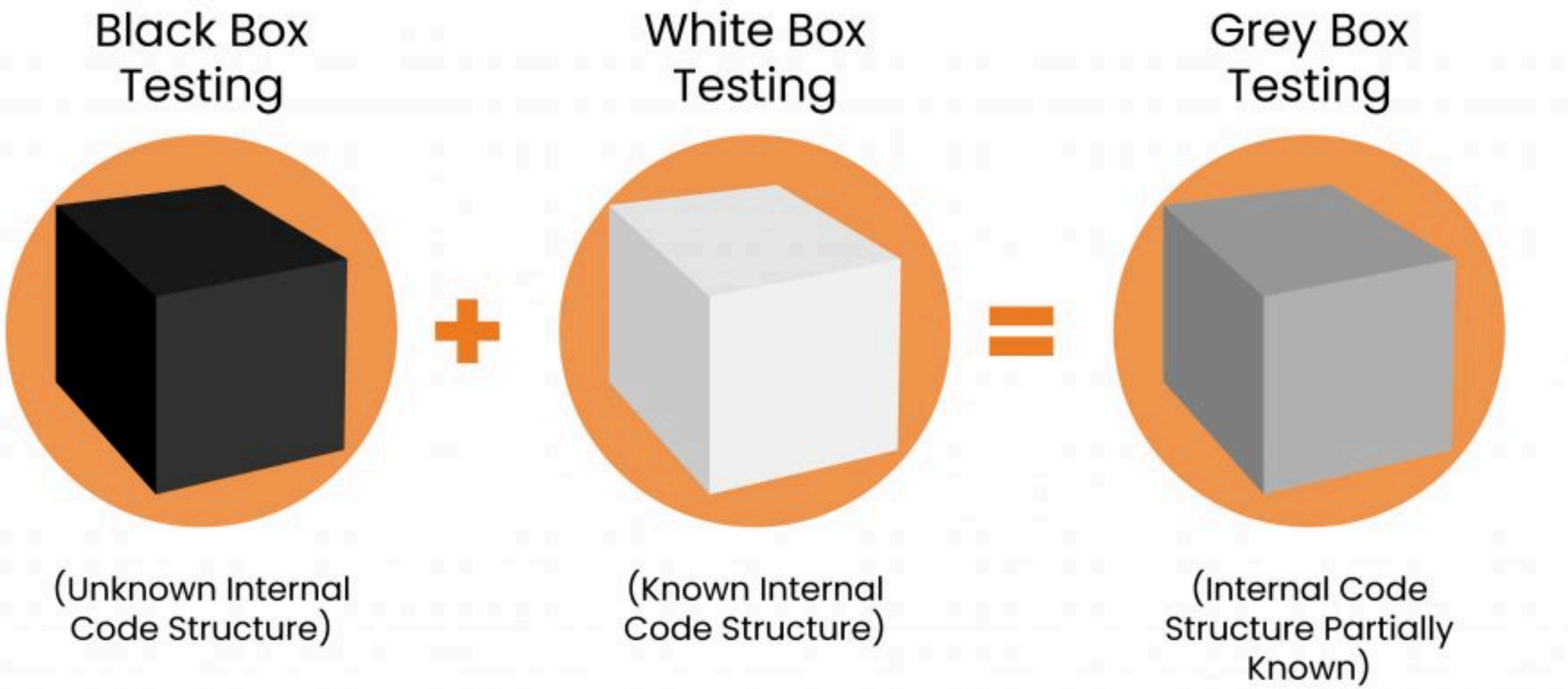


Testing Tools

External testing

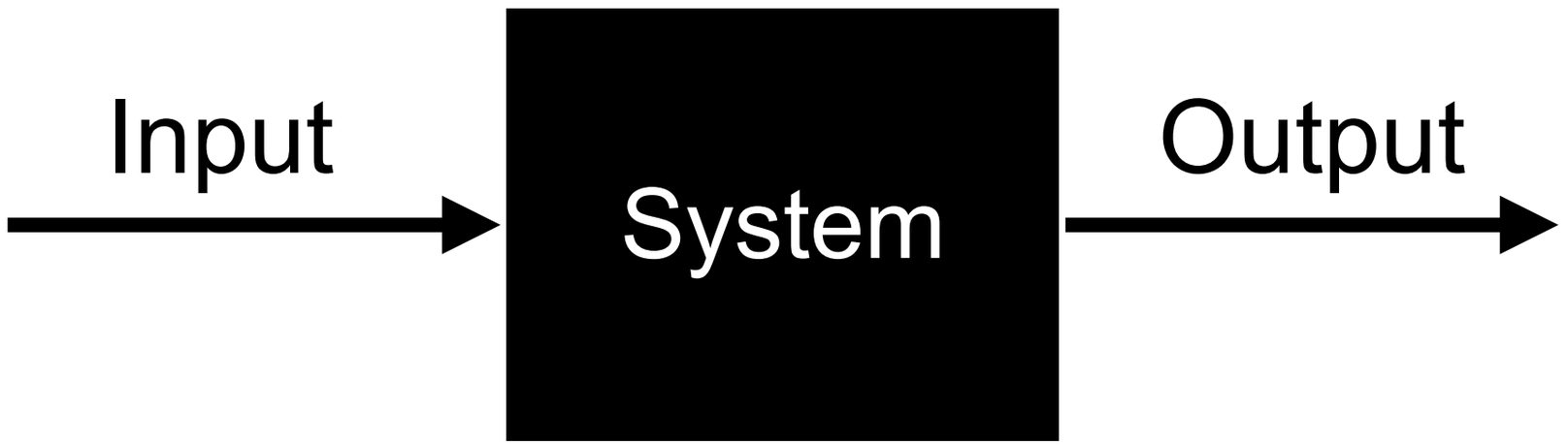
Internal testing

Types of Testing Methods



External Testing

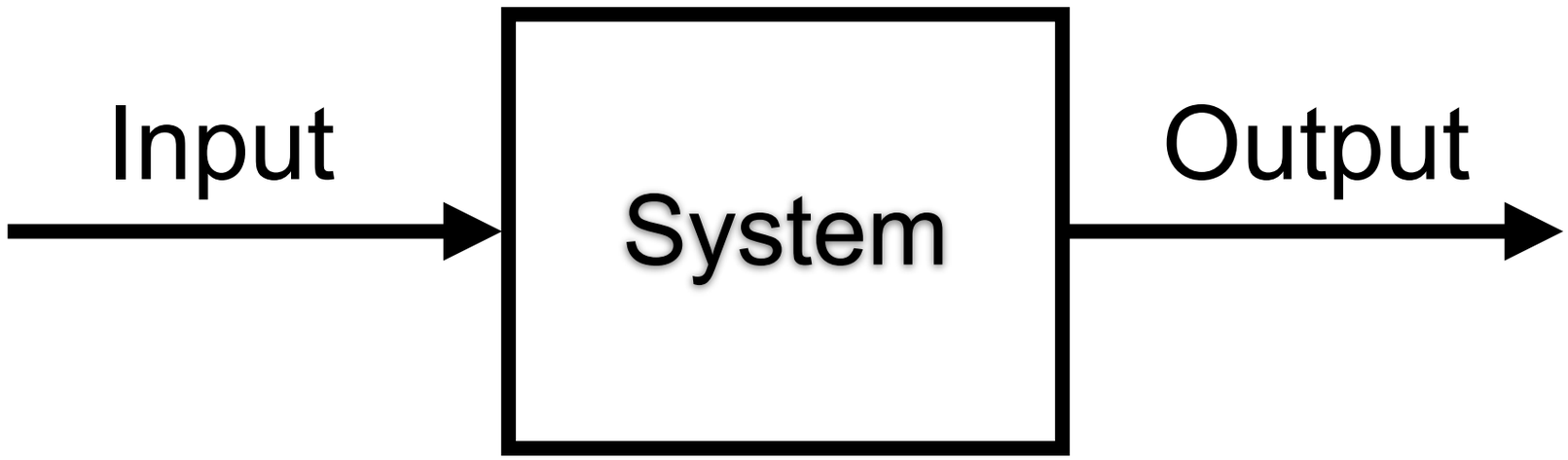
Black box testing



End-to-end testing

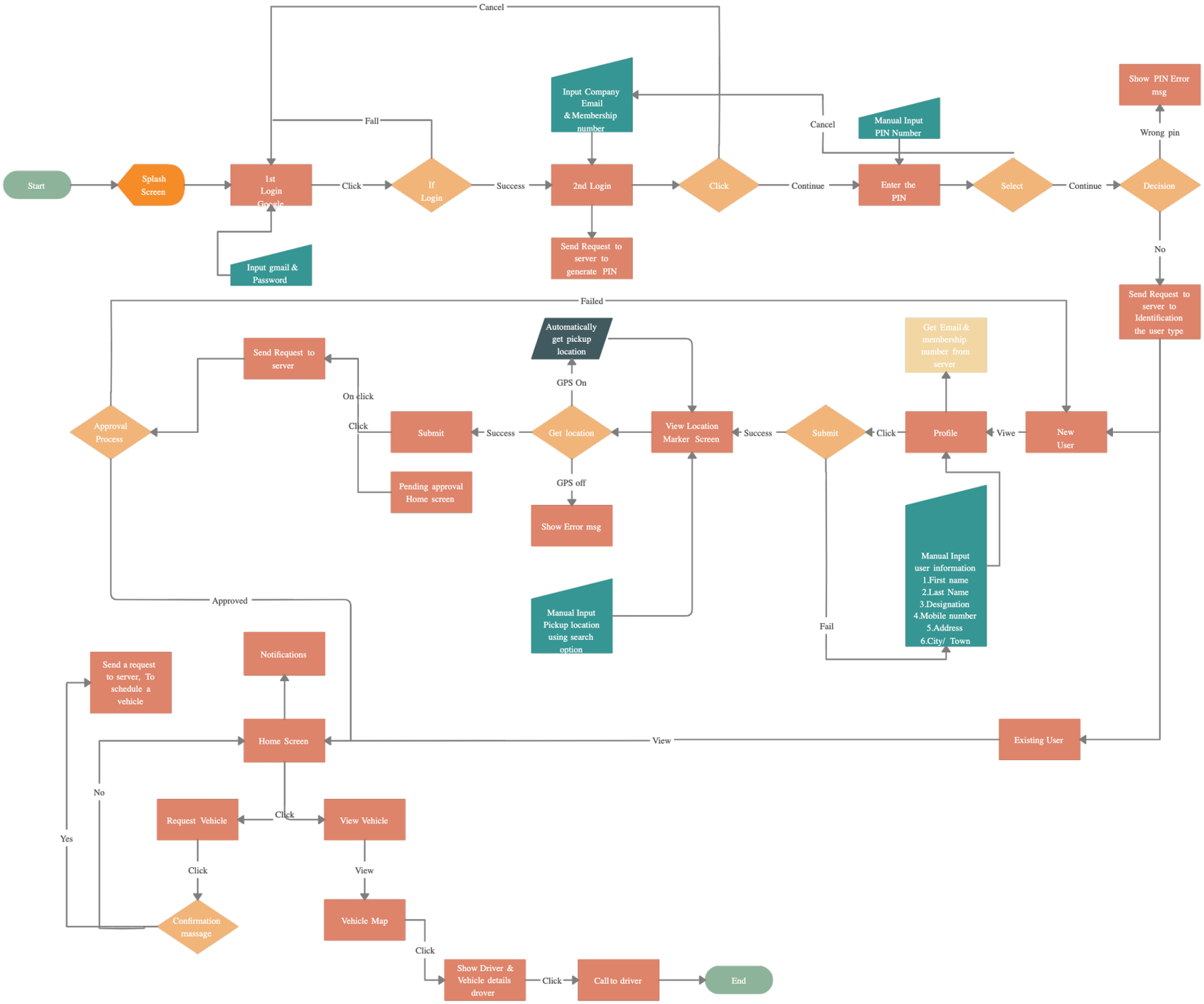
Internal Testing

White box testing

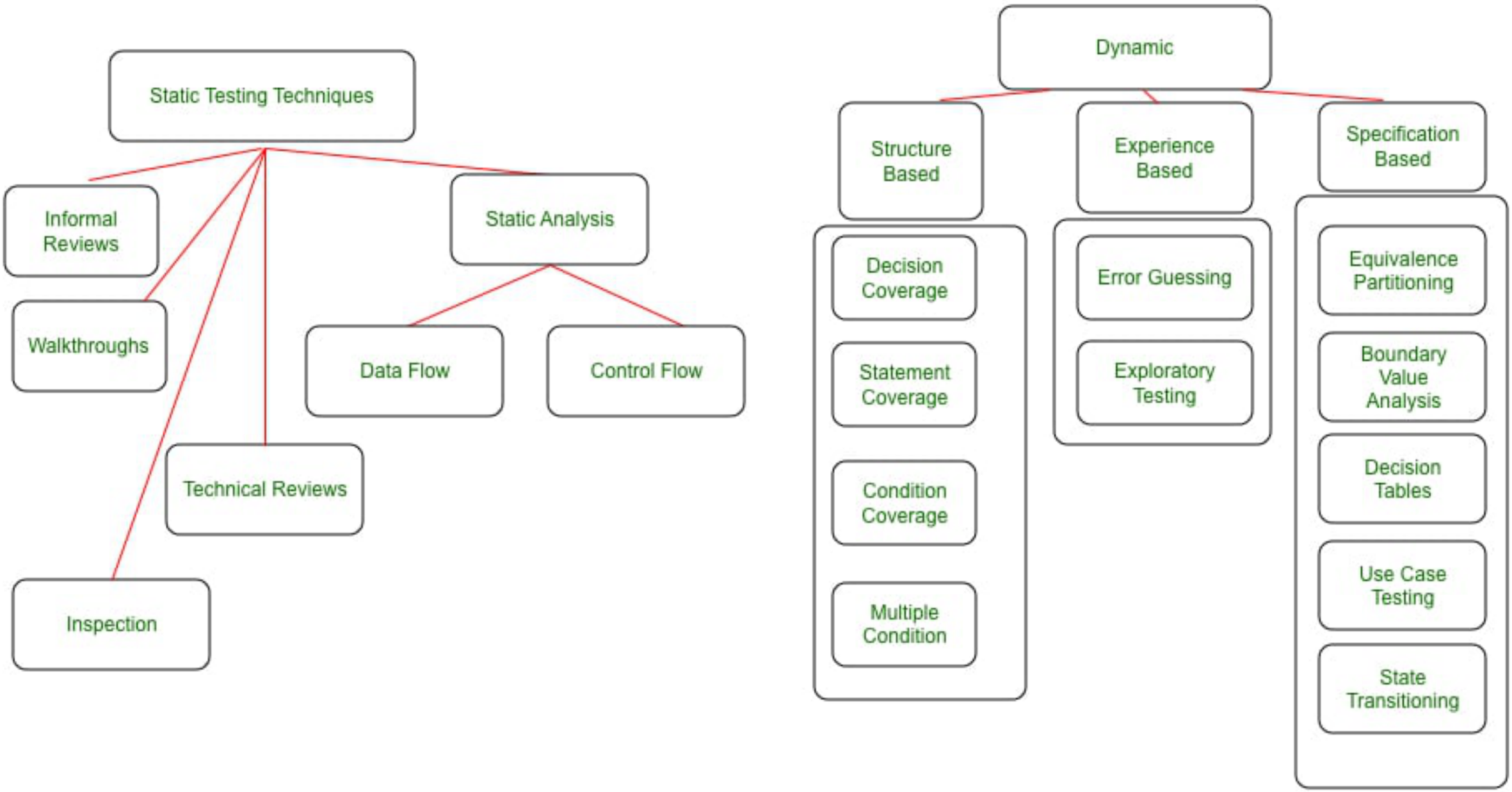


Unit, Integration,
Contract, component testing

Internal Testing

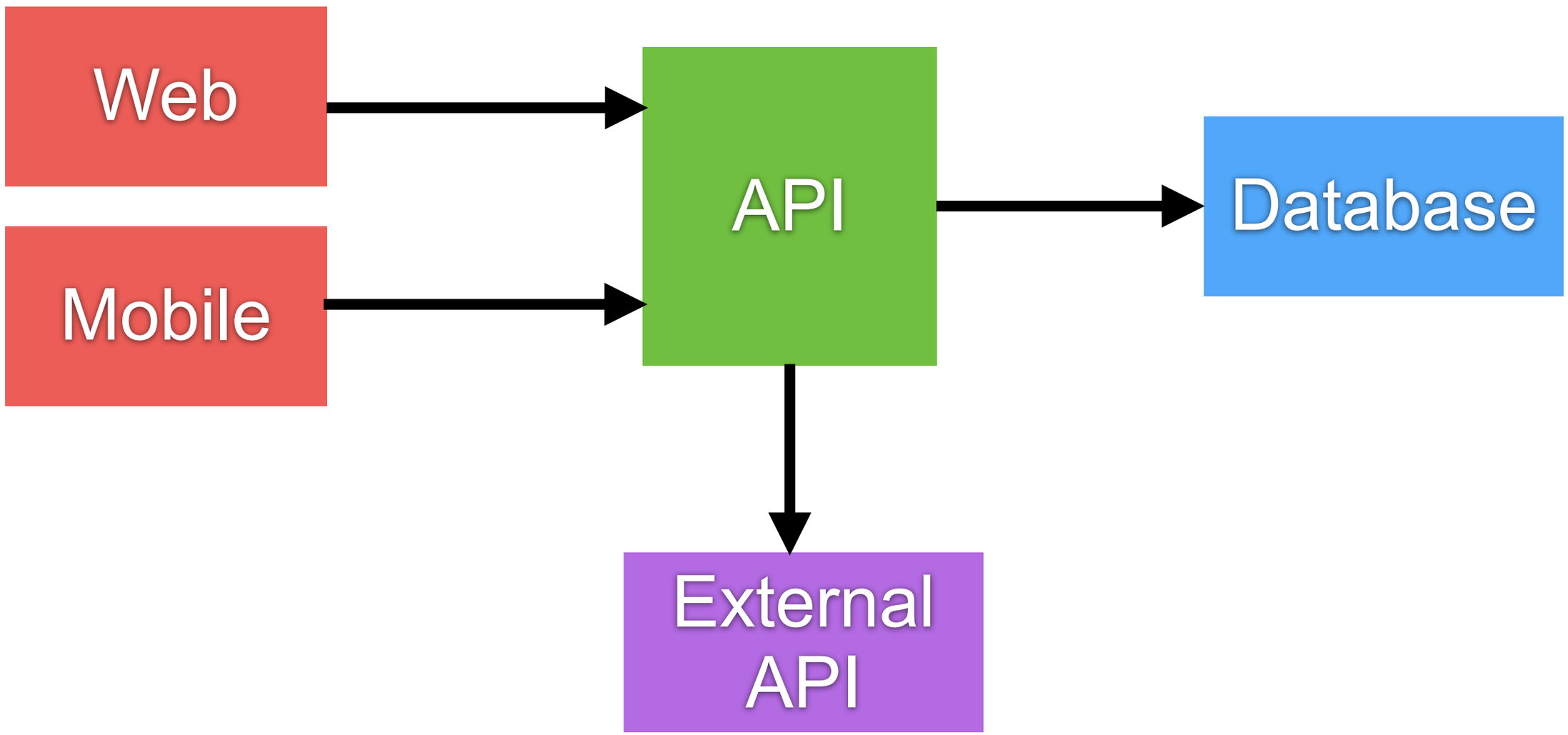


Testing Techniques



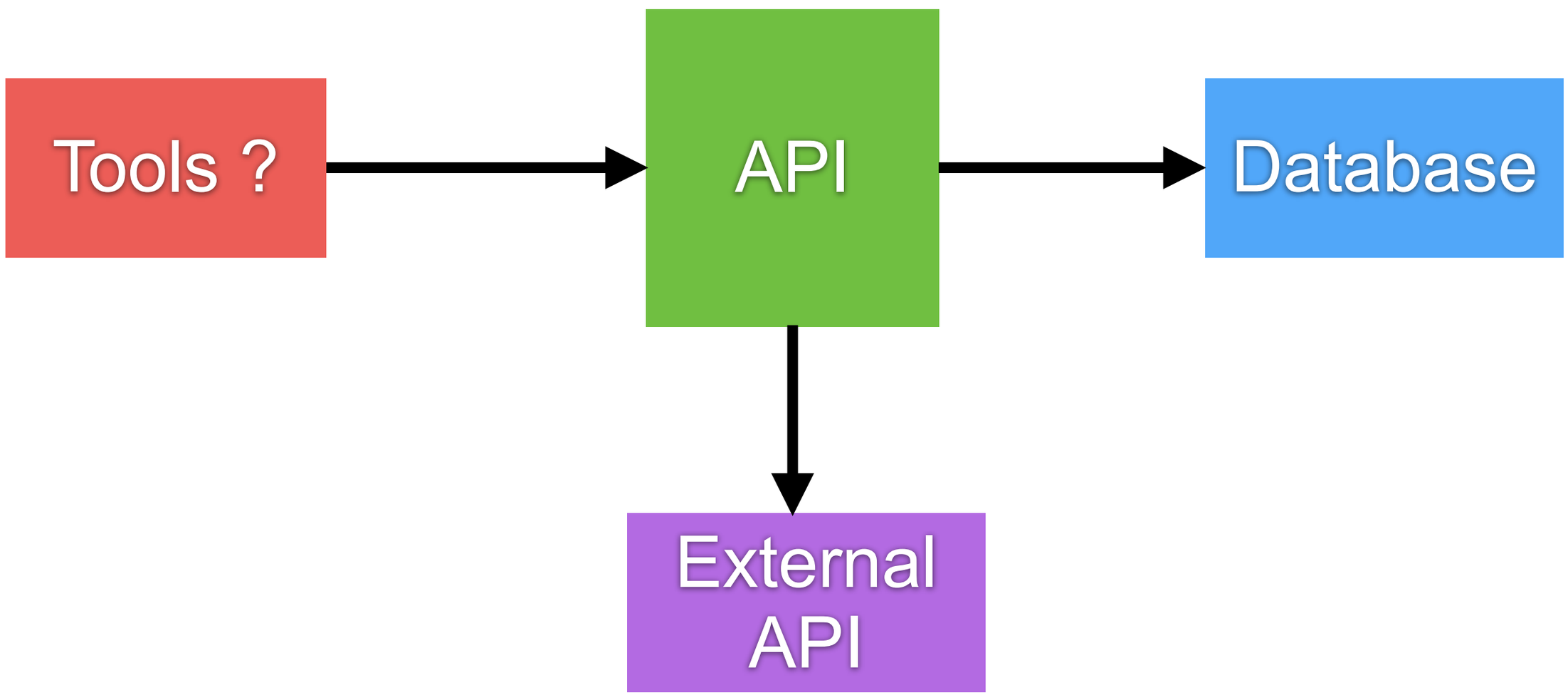
https://www.geeksforgeeks.org/software-testing-techniques/

Architecture



API Testing

API Testing ?



External testing for API

REST API

GraphQL

gRPC

WebSocket

Socket.io

Messaging

External testing for API

List of tools

External testing for API

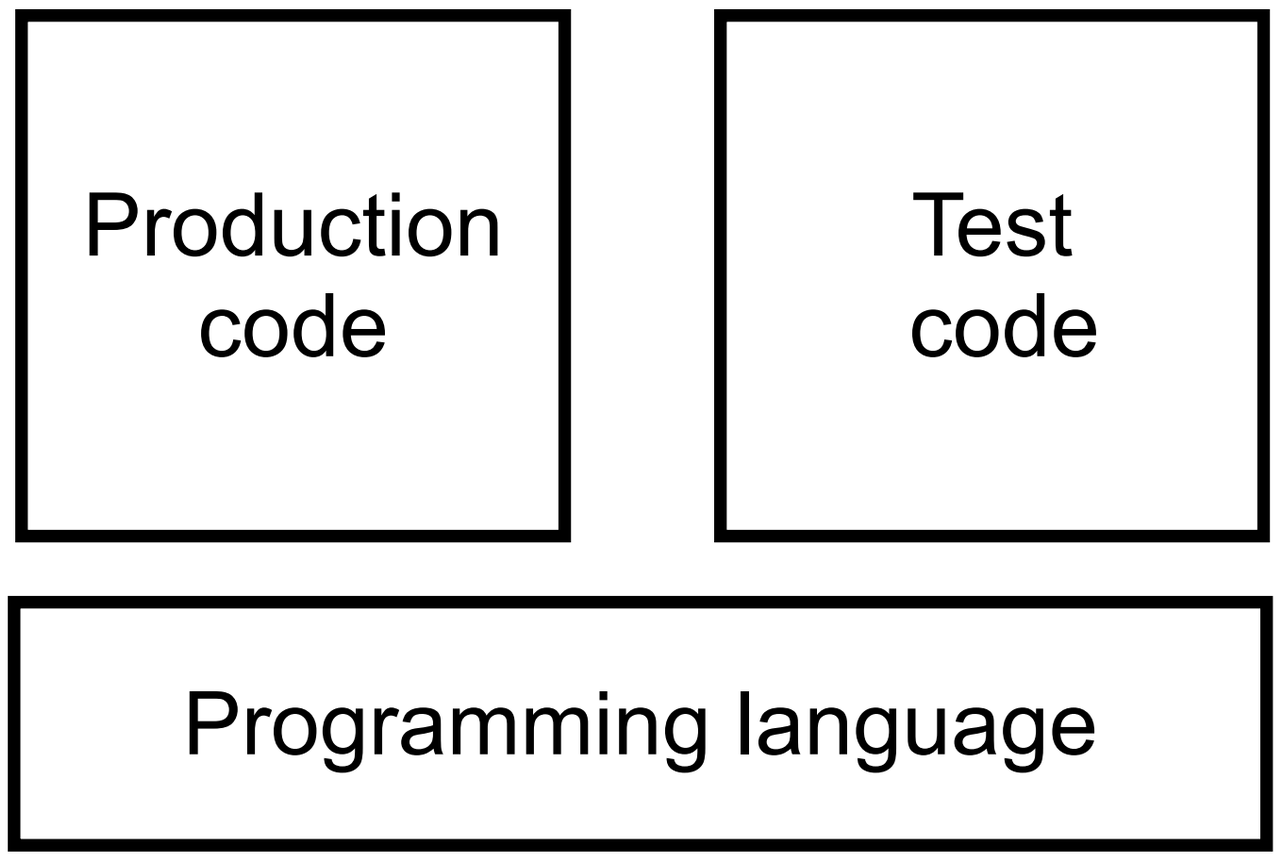
Write test script and coding

Robot Framework

Playwright

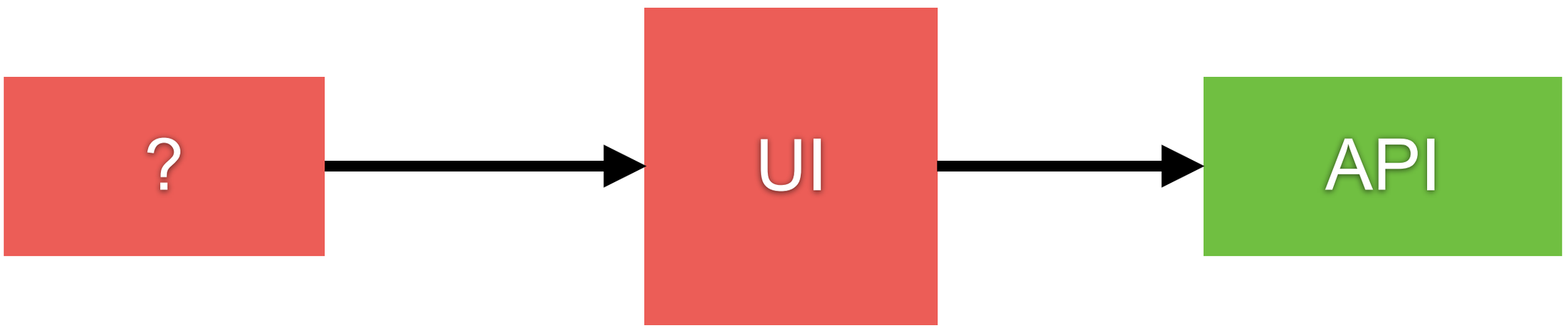
External testing for API

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

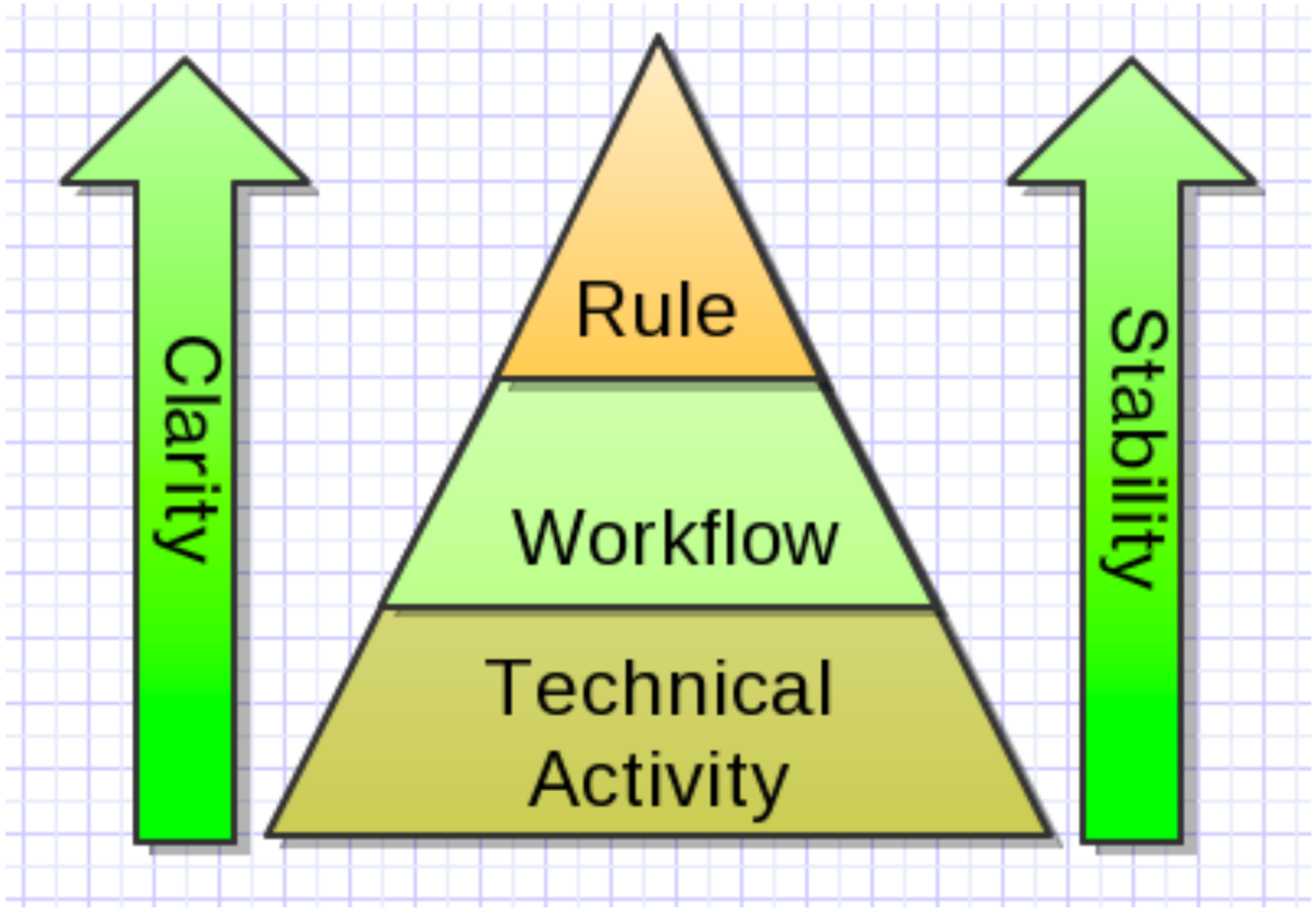
Record and Play back tools

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

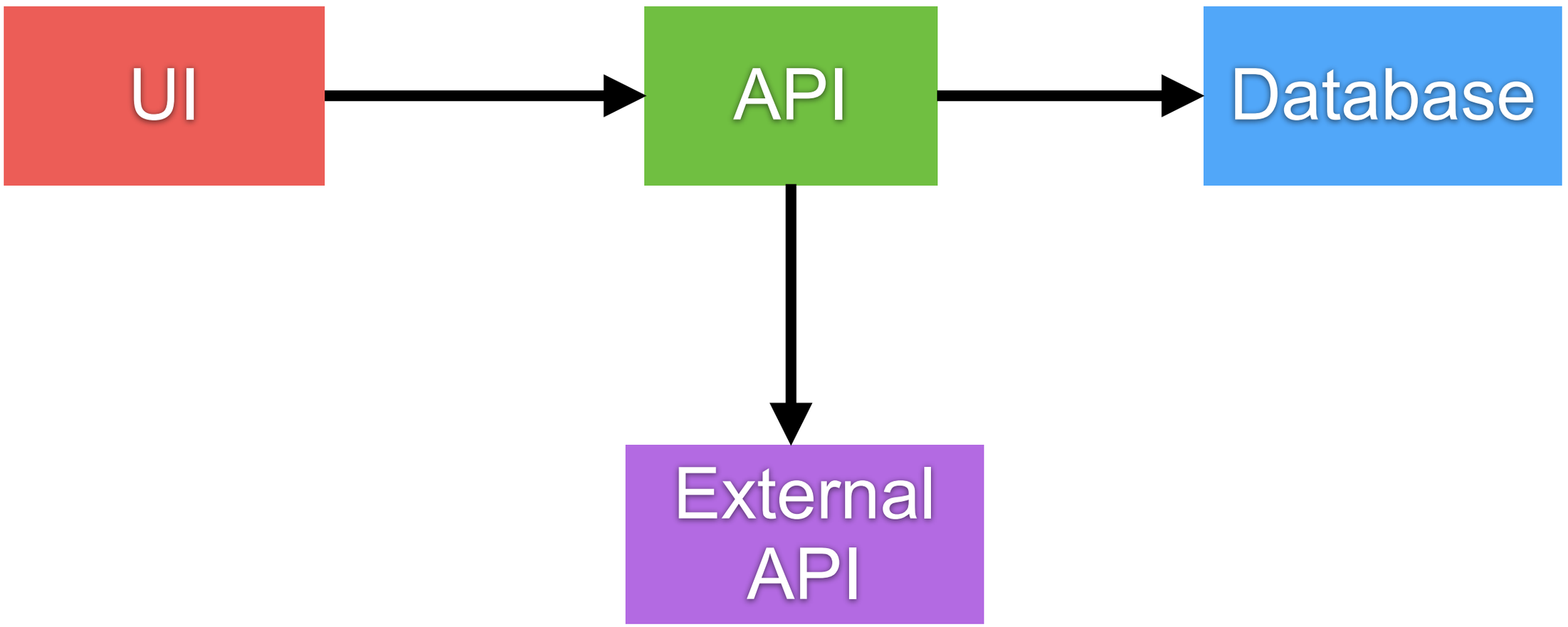
what does a user have to do to exercise the functionality through the UI

Technical activity level

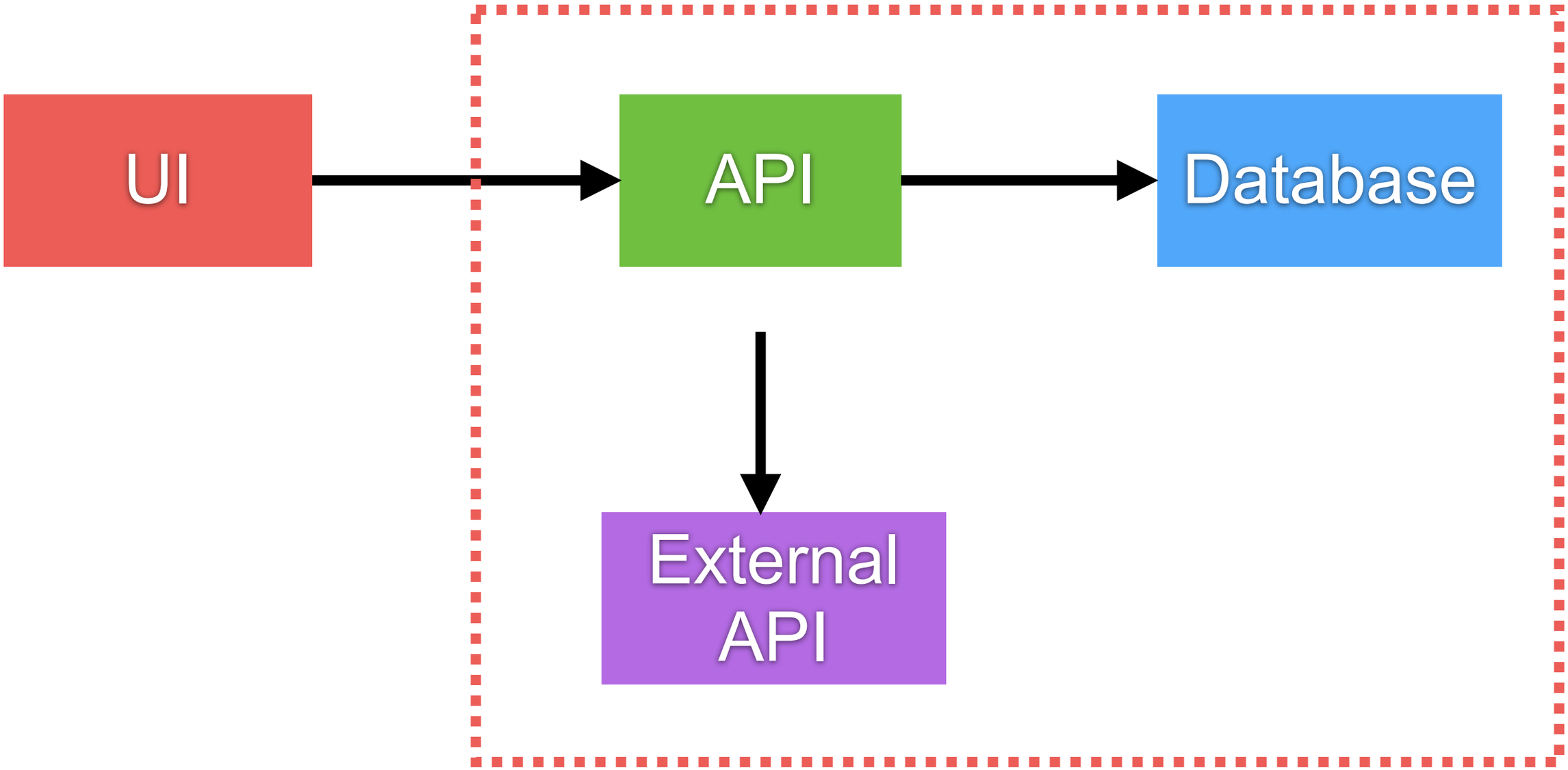
what are the technical steps required to exercise the functionality

Manage dependencies

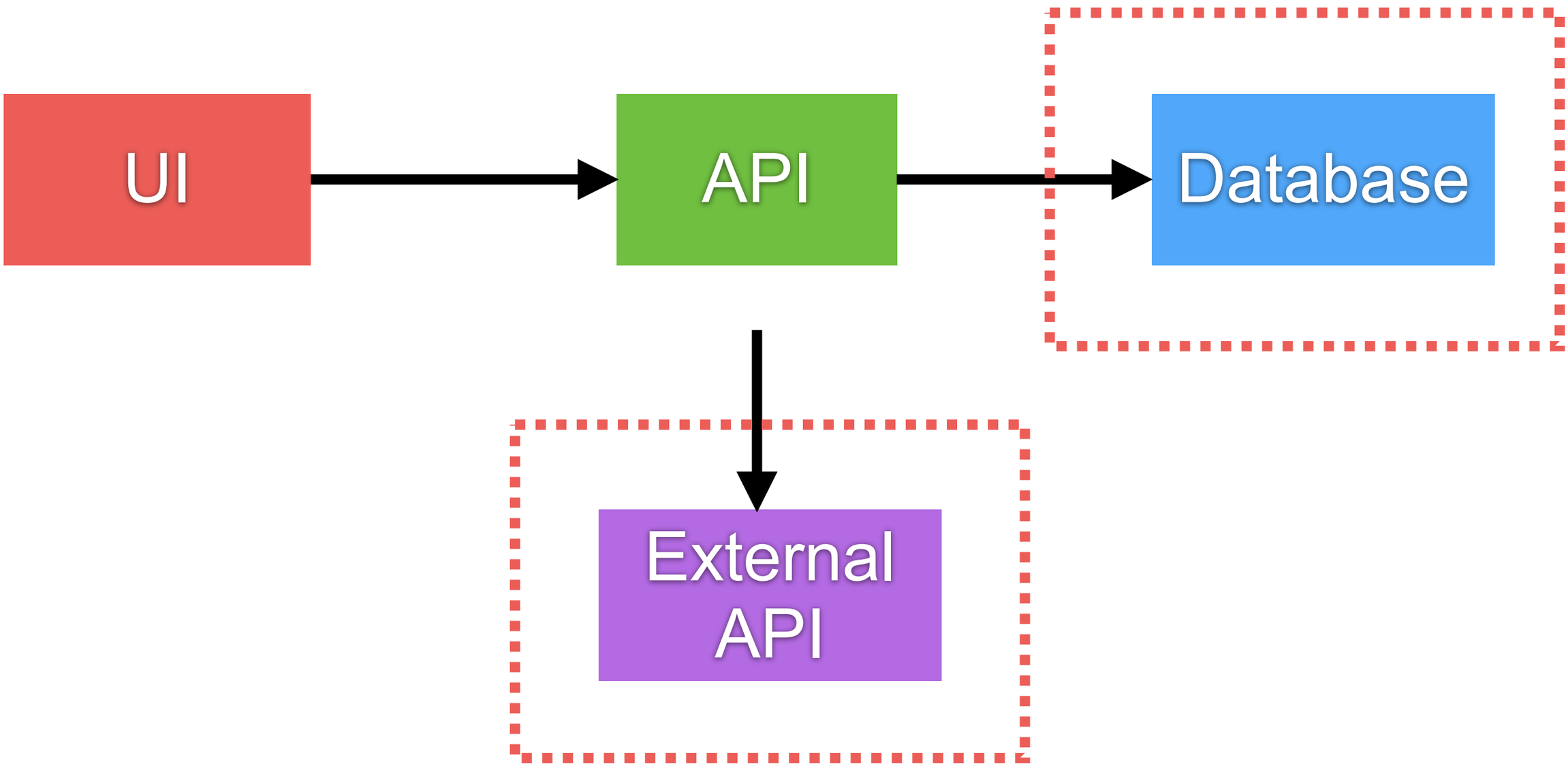
Manage dependencies ?



UI Perspective



API Perspective

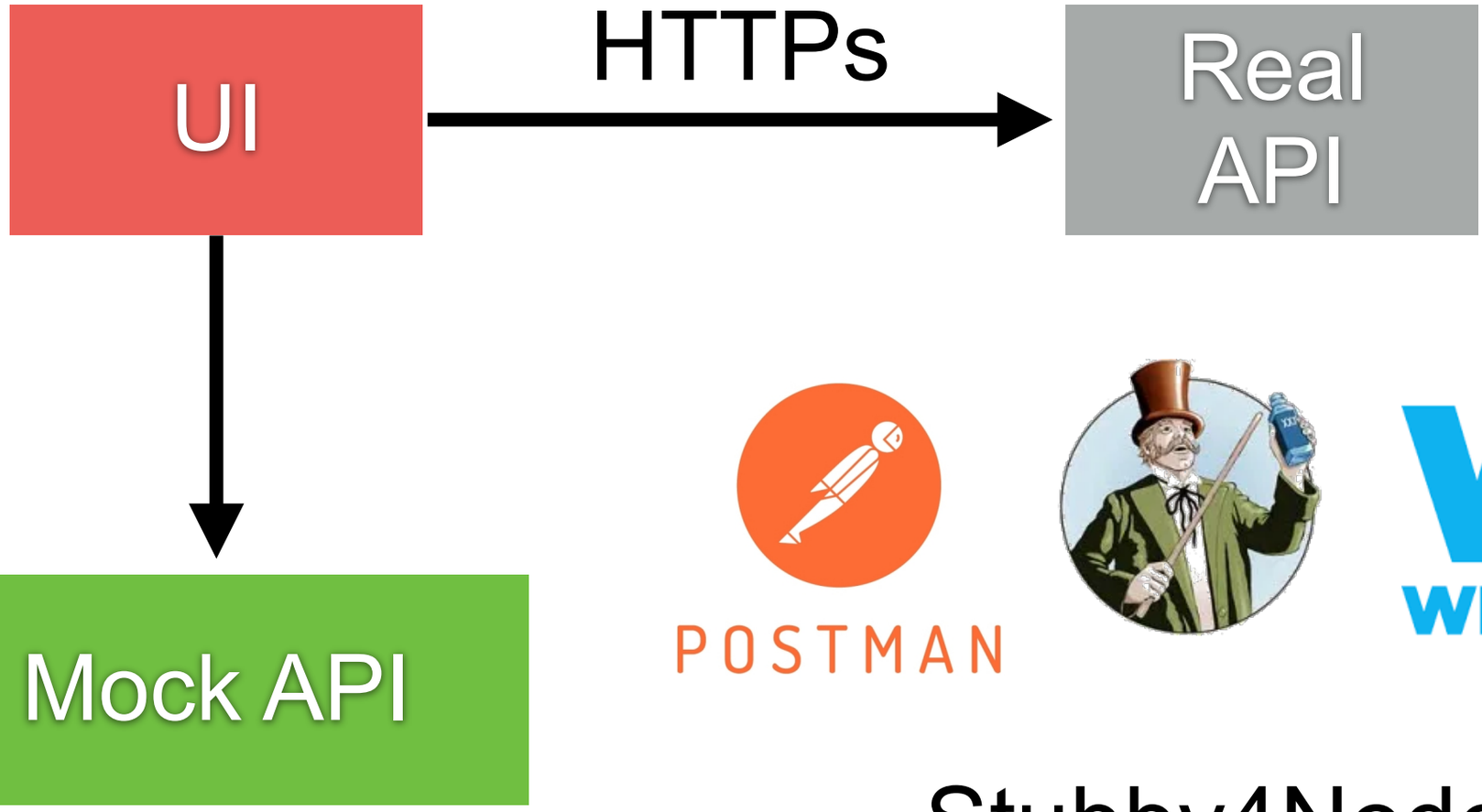


Manage dependencies of UI

Manage dependencies of UI

External mock API server
Interceptor in web browser

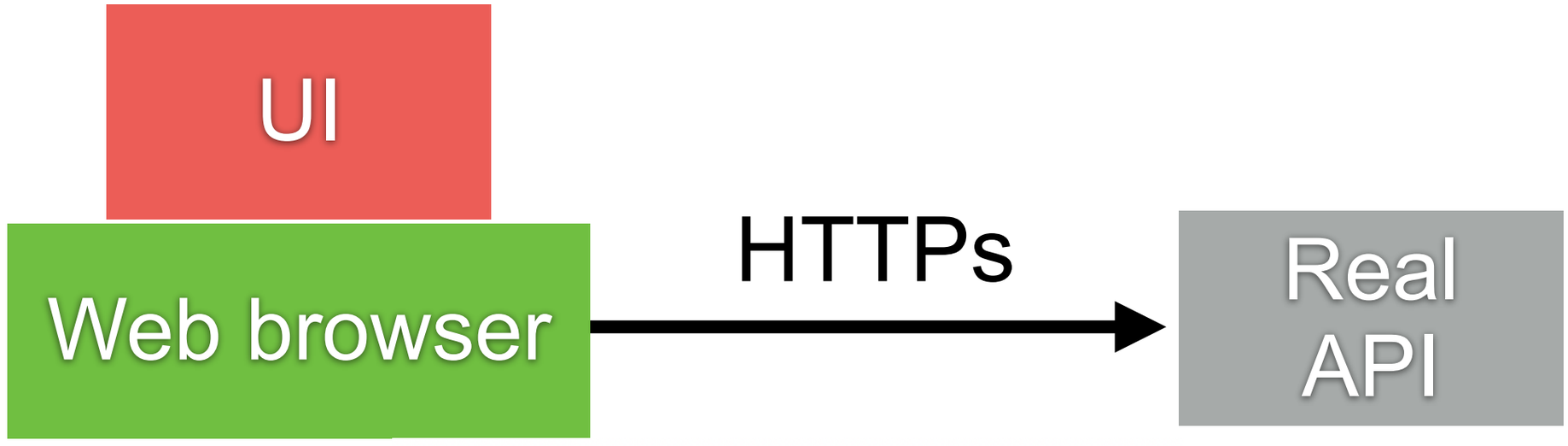
External Mock API Server



Stubby4Node

JsonServer

Intercept in Web browser

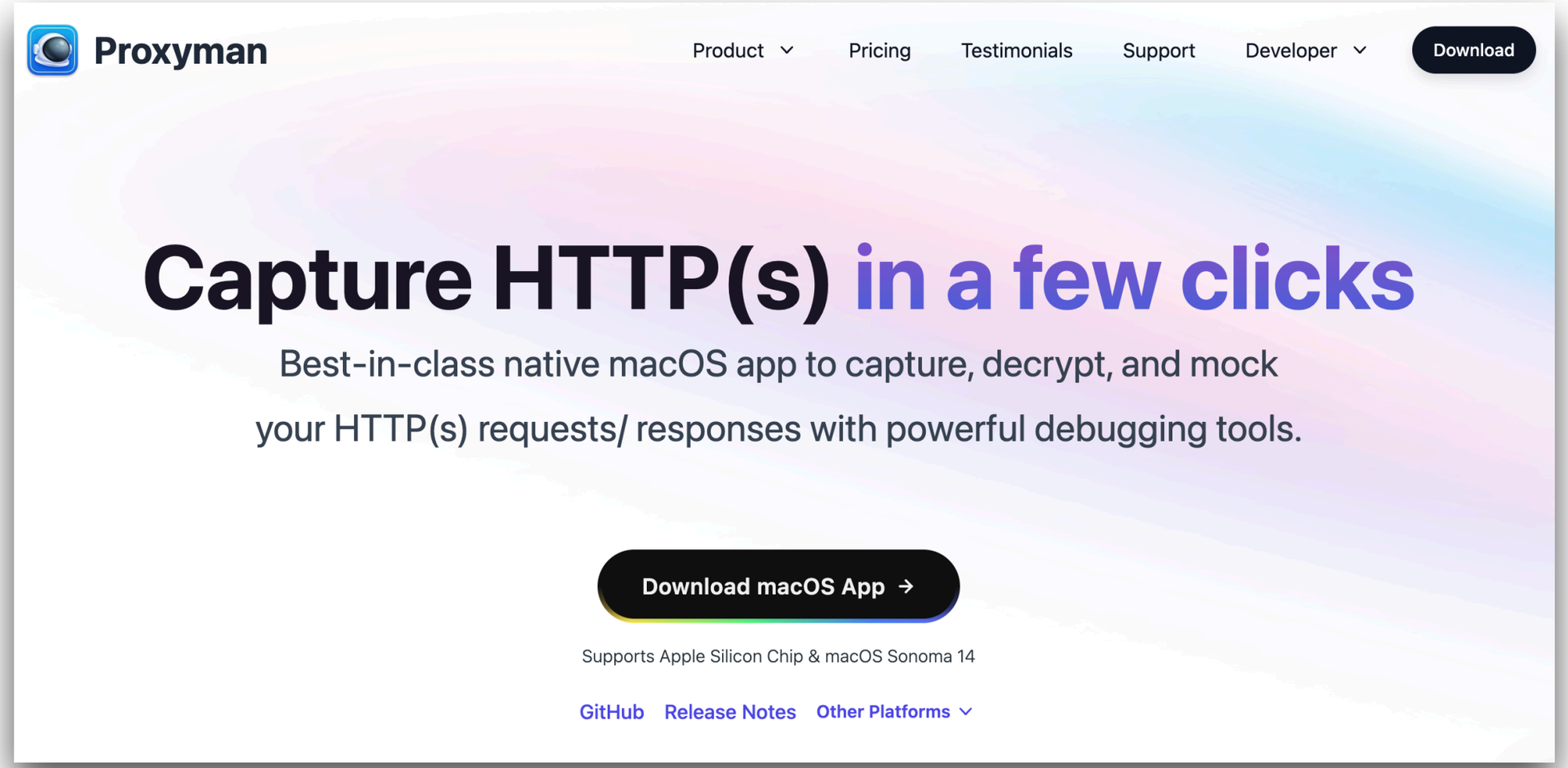


Playwright

cypress

Proxy Man

Capture HTTP request !!

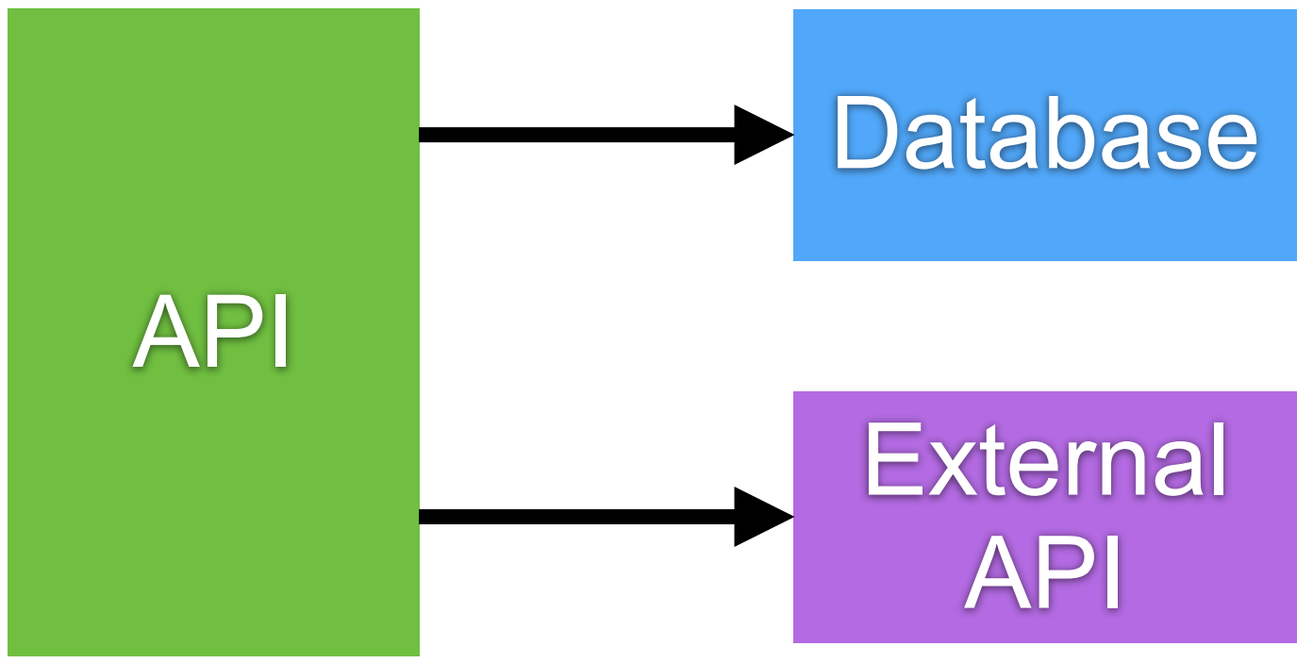


https://proxyman.io/

Manage dependencies of API

Manage dependencies of API

Database
External API



Database

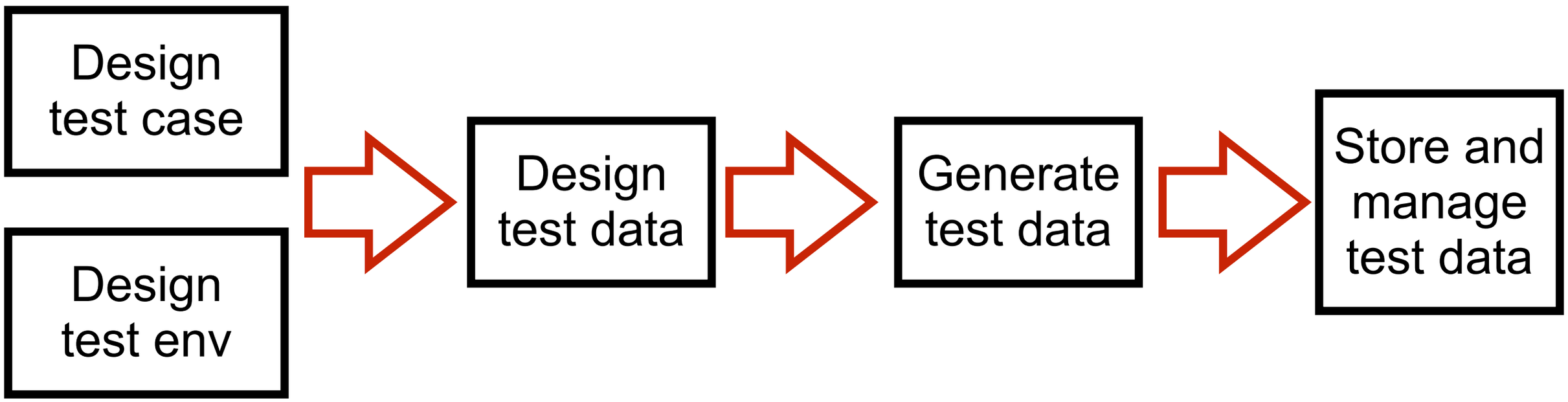
In-memory database
Container-based
Real database

Tables

Data for test

Database migration tools

Design test data ?



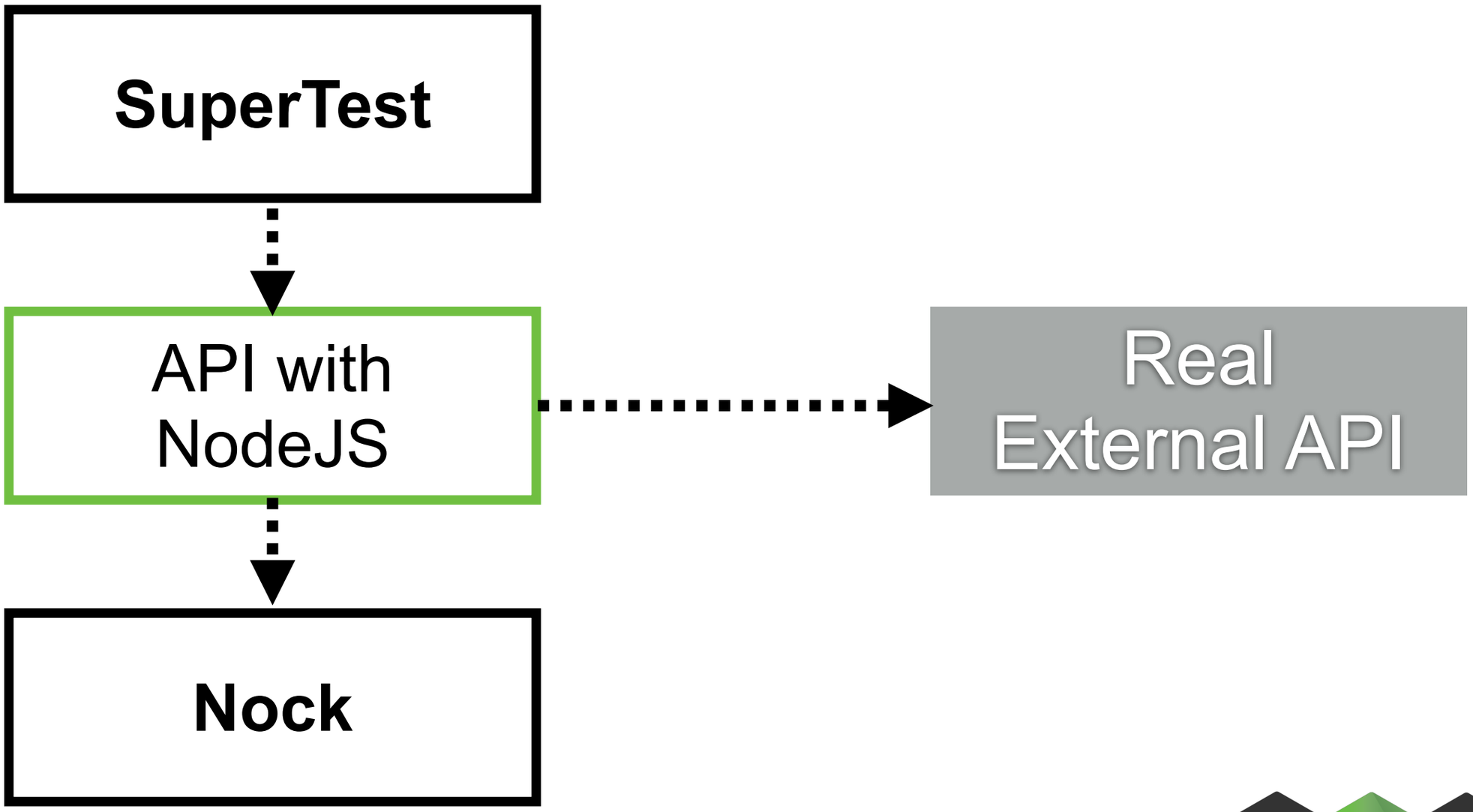
External API

External mock API server

Internal mock API server

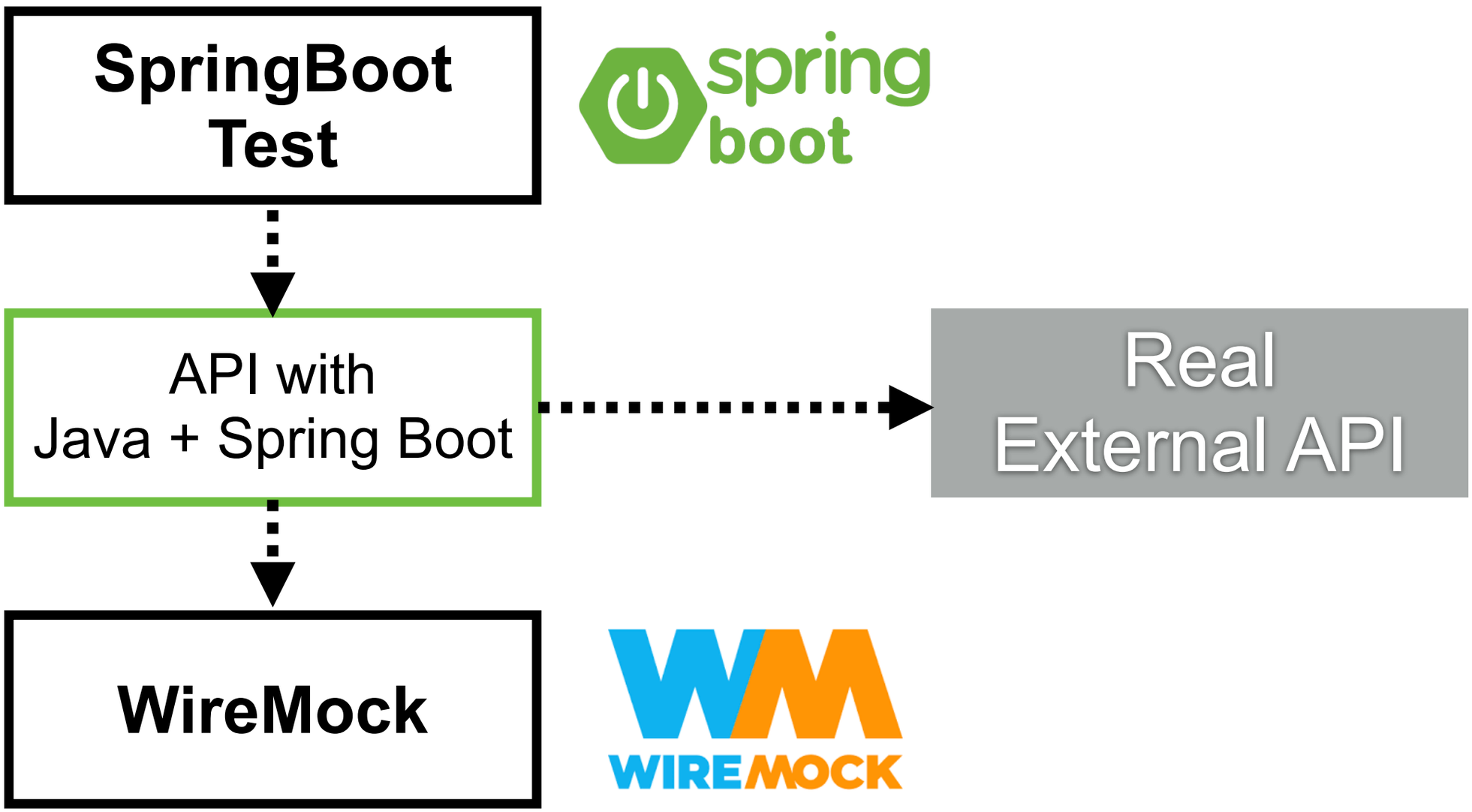
Use test double in code

Internal Mock API with NodeJS



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

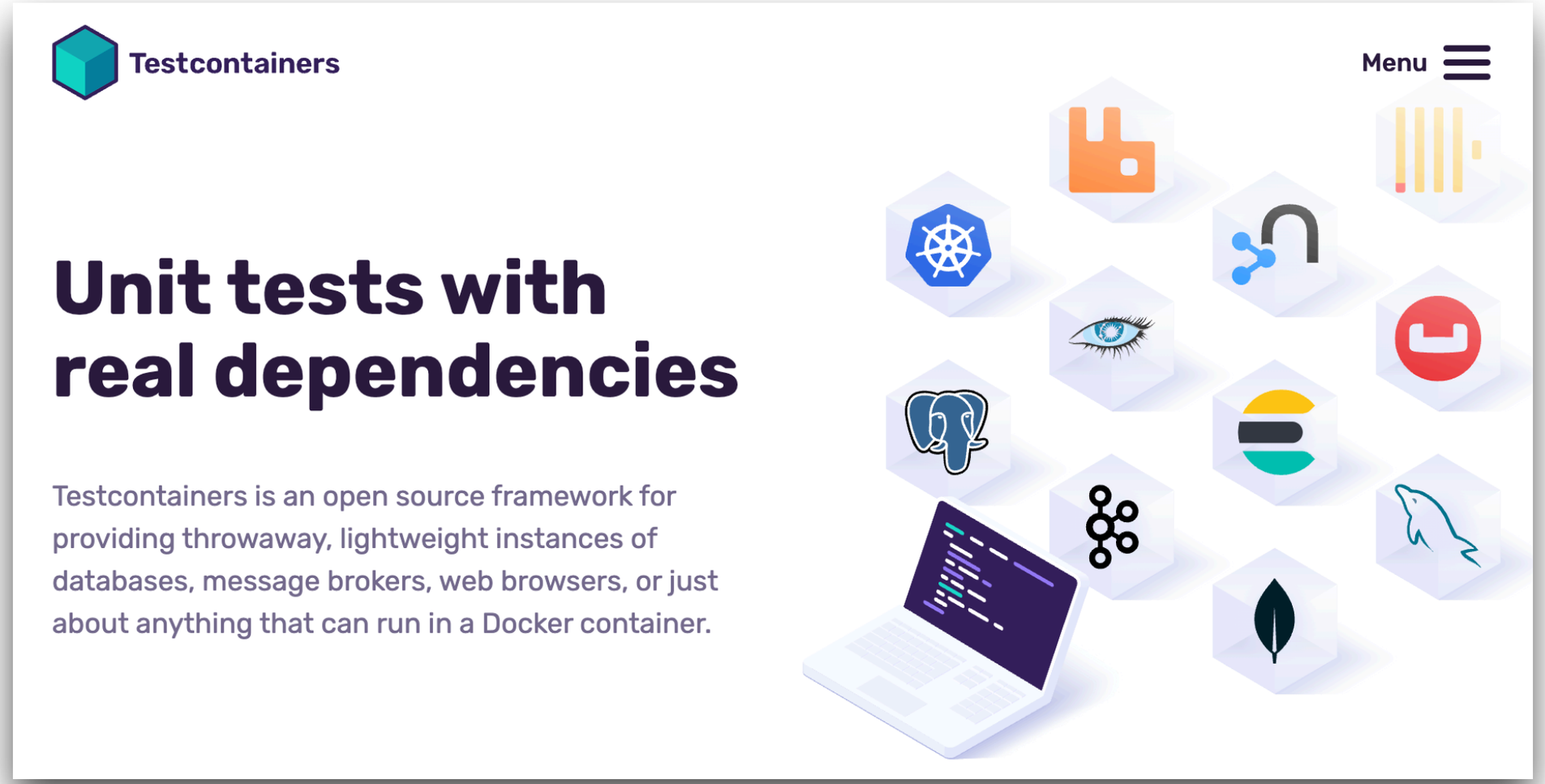
Internal Mock API with Java



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

Test Container

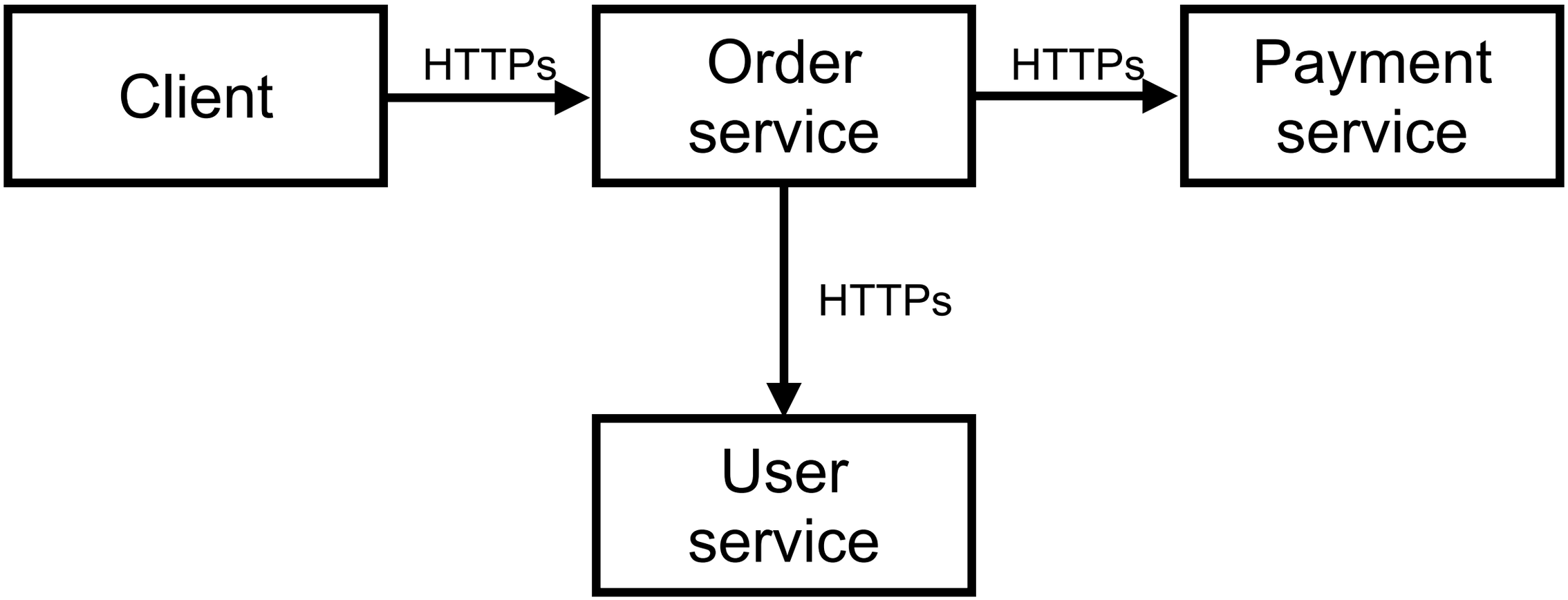
Testing with real dependencies



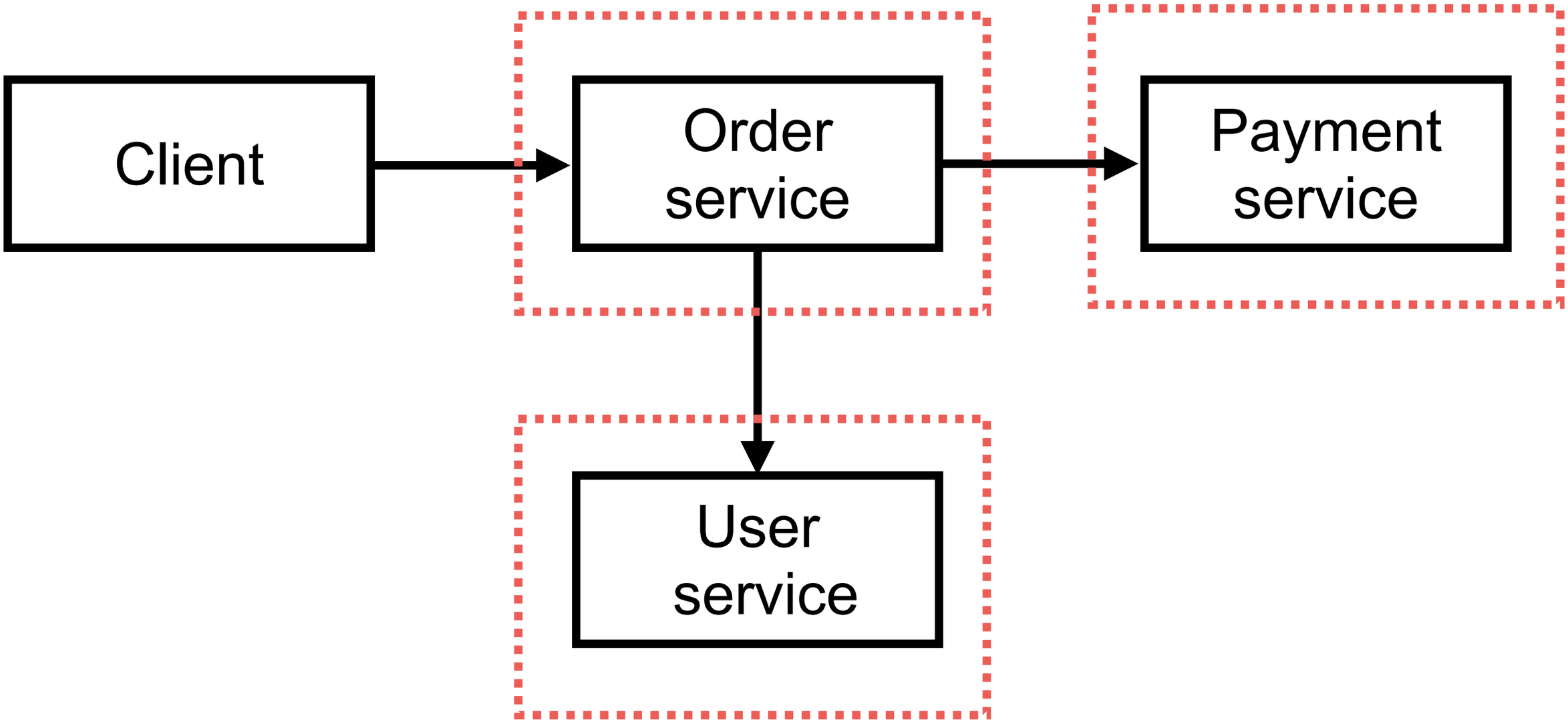
https://testcontainers.com/

More Testing with APIs

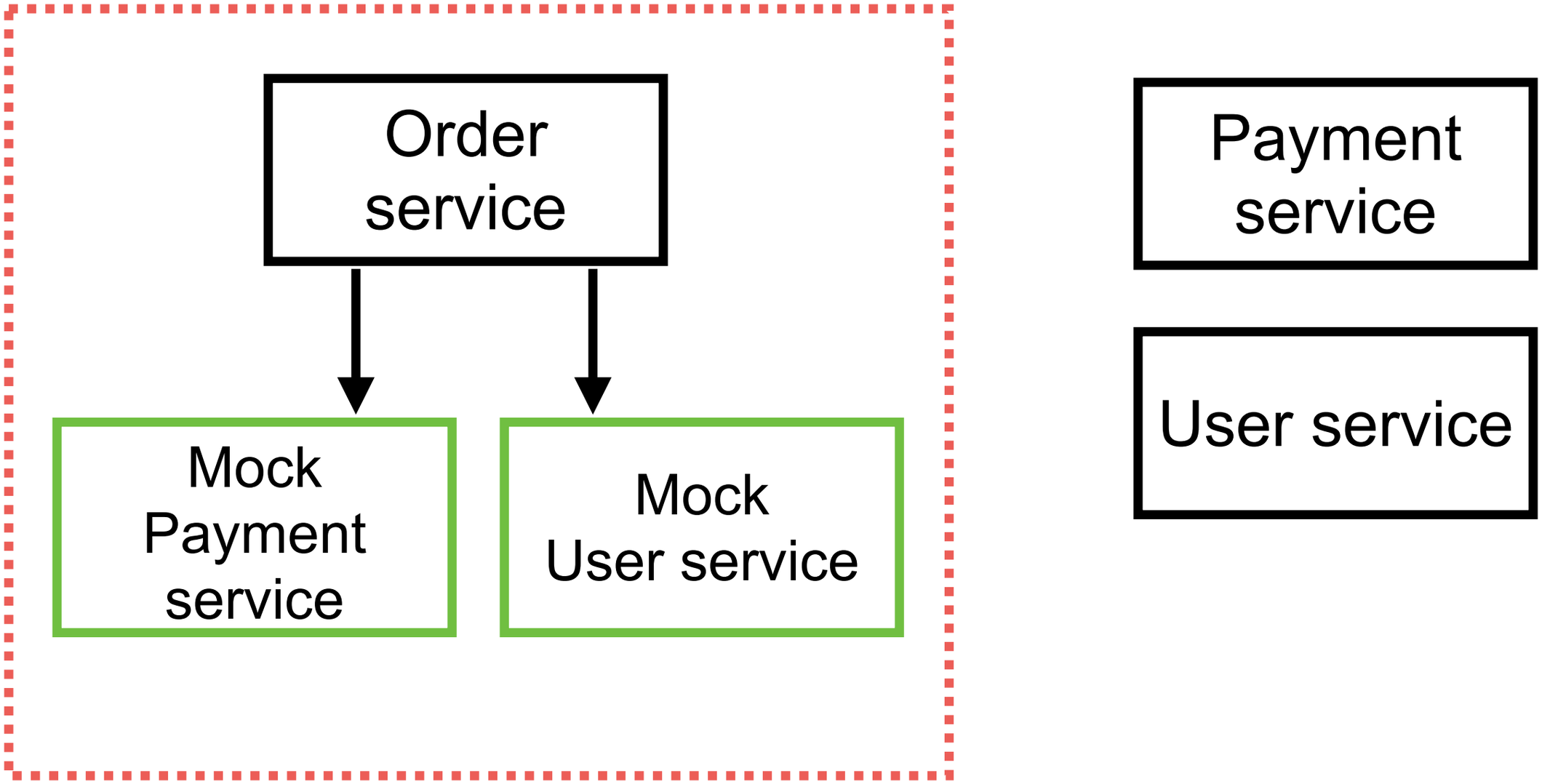
Service-to-Service Testing ?



Component Testing

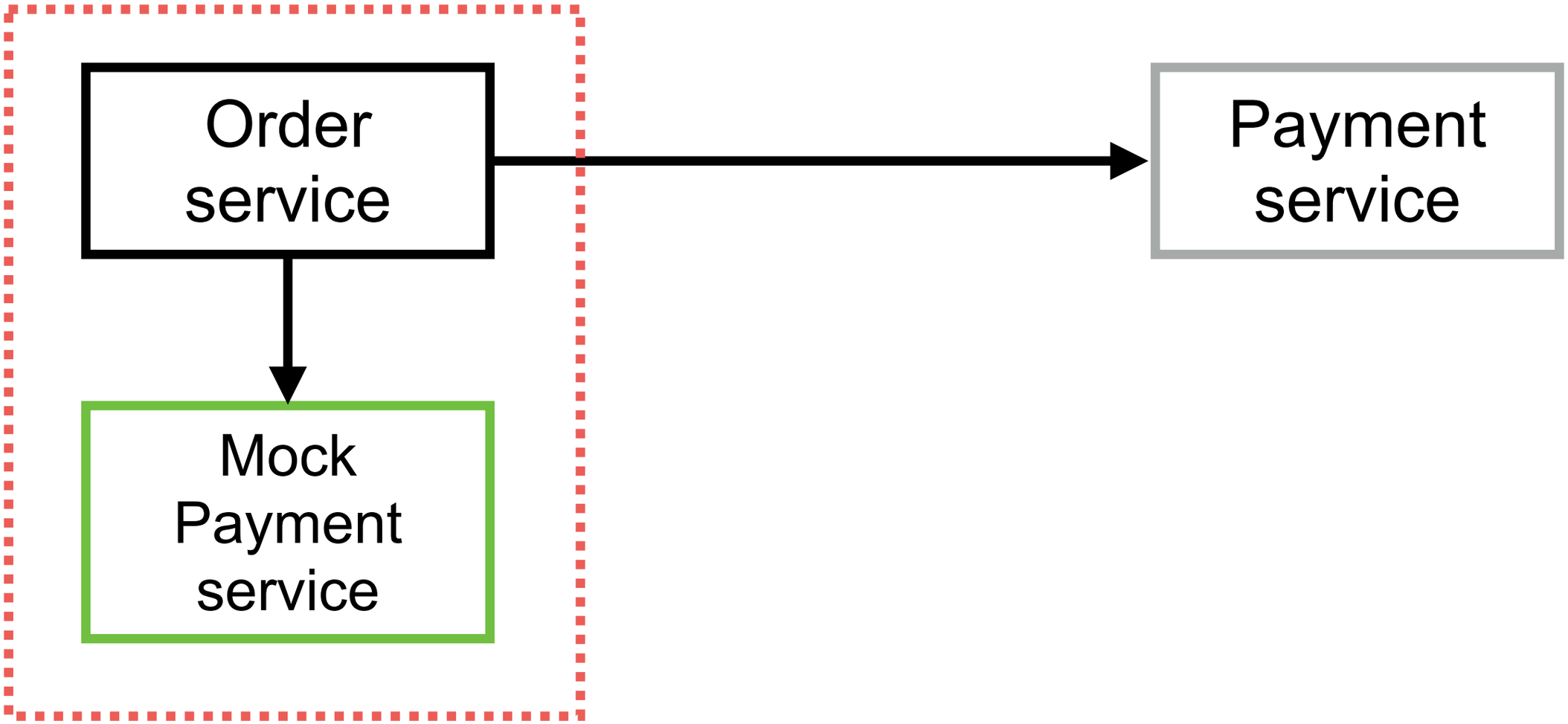


Component Testing



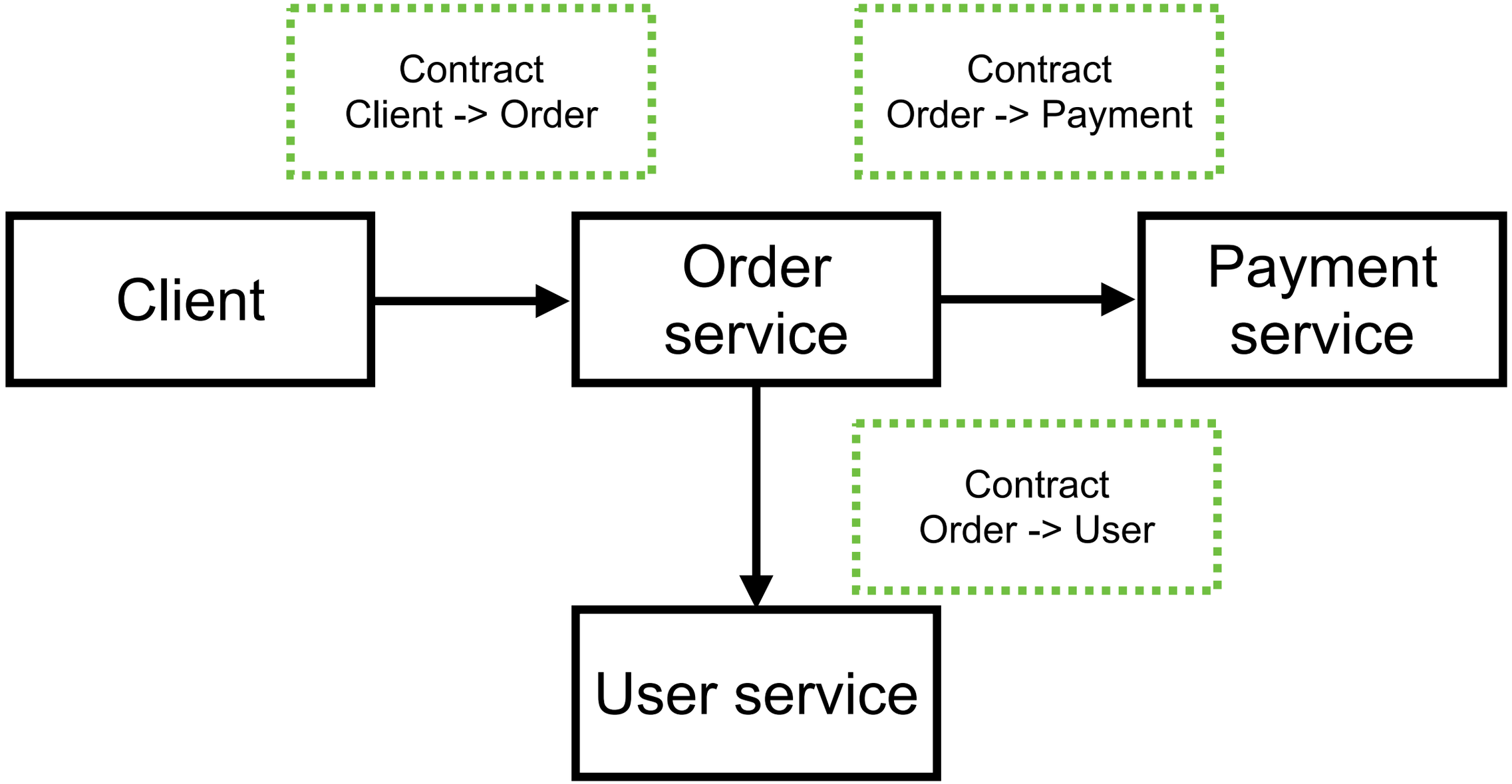
Workshop

Component Testing workshop



Contract Testing

Contract Testing



Contract Format

REST API

Provider name
Consumer name
List of API

```
GET /users/1
Response = 200
{
  “id”:1,
  “name”: “Somkiat”
}
```

```
GET /users/2
Response = 404
{
  “message”: “User not found”
}
```

Contract Format

Messaging (Asynchronous)

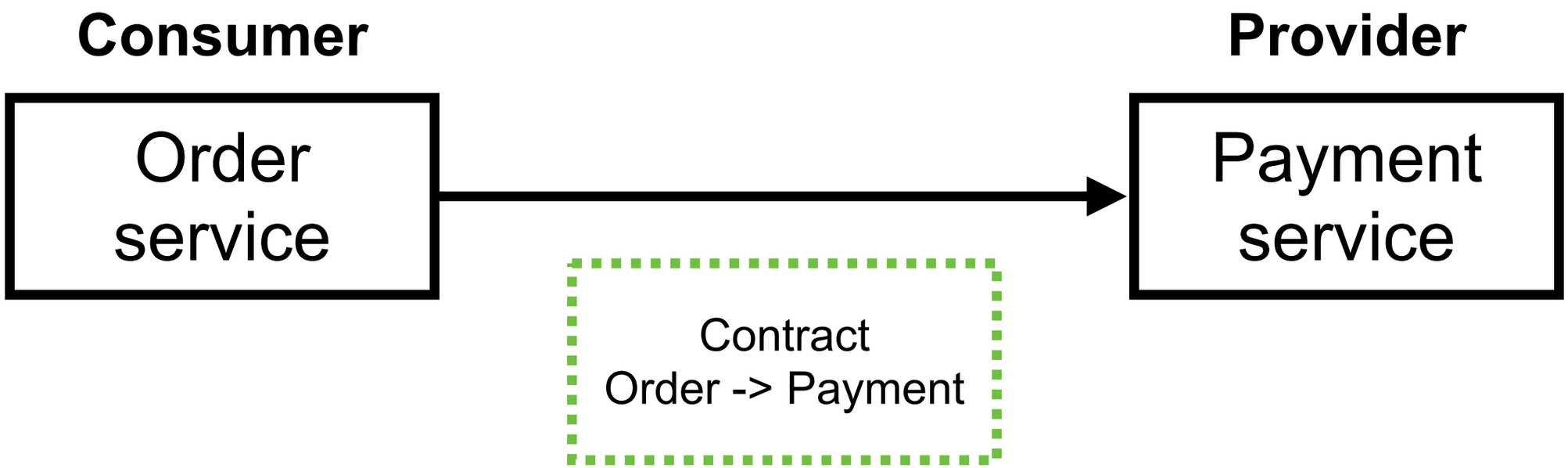
Provider name	
Consumer name	
List of API	

Event name = OrderCreated
Message Format
{
“id”: “202401”,
“Items”: [
…
]
}
	Event name = StockReserved
	Message Format
	{
	“item_id”: “1”,
	“status”: “success”
	}

Process of contract testing

Order to Payment

1. Create contract between services



Order to Payment

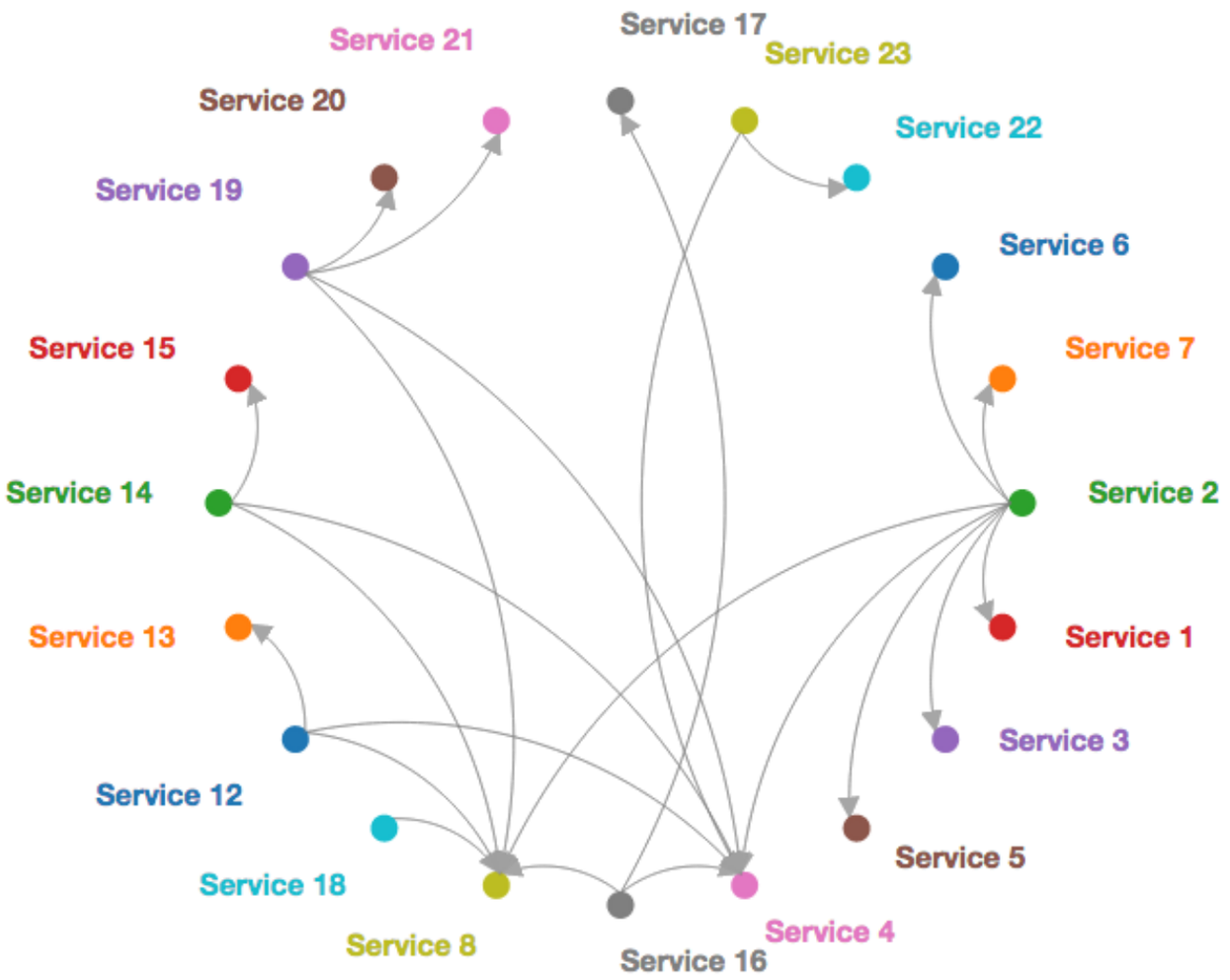
2. Publish contract

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	

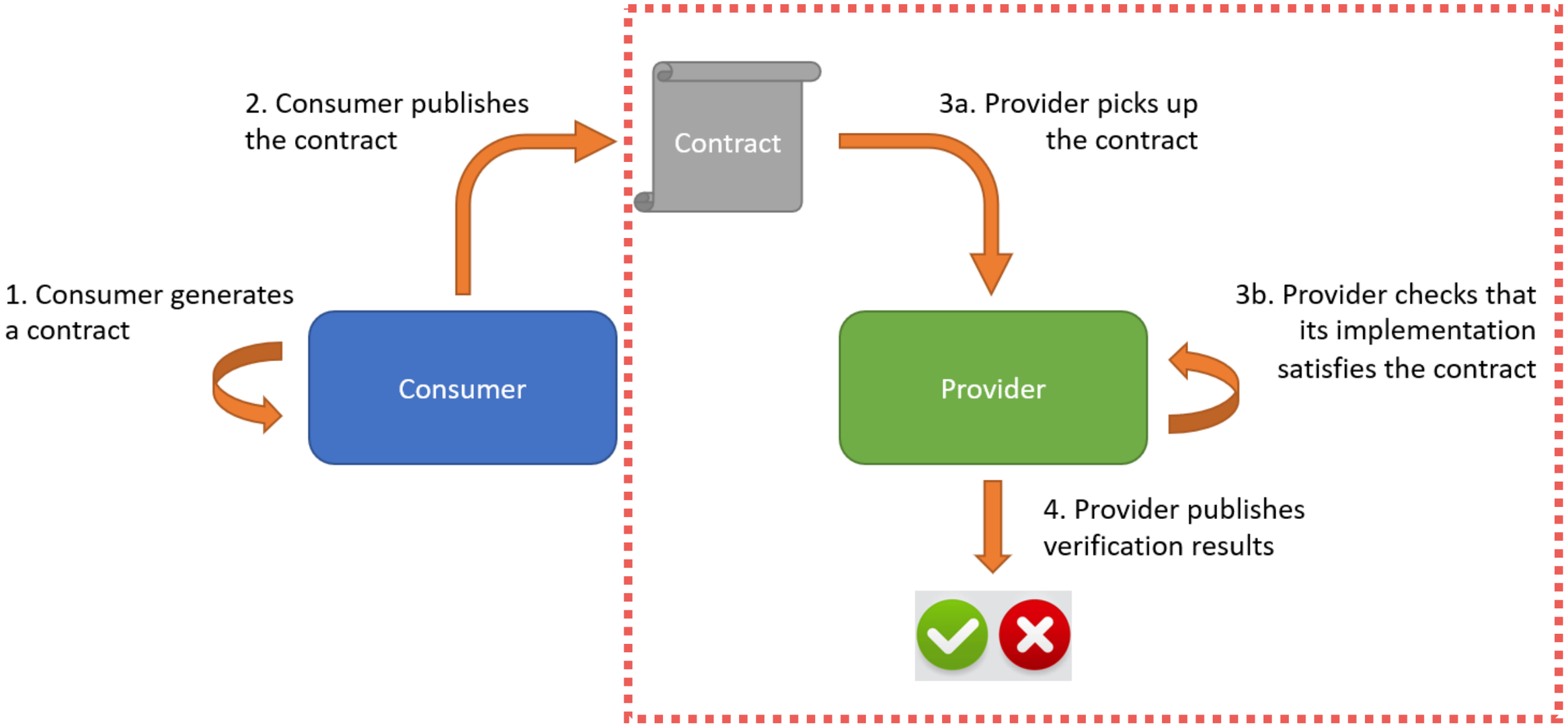
Order to Payment

2. Publish contract



Order to Payment

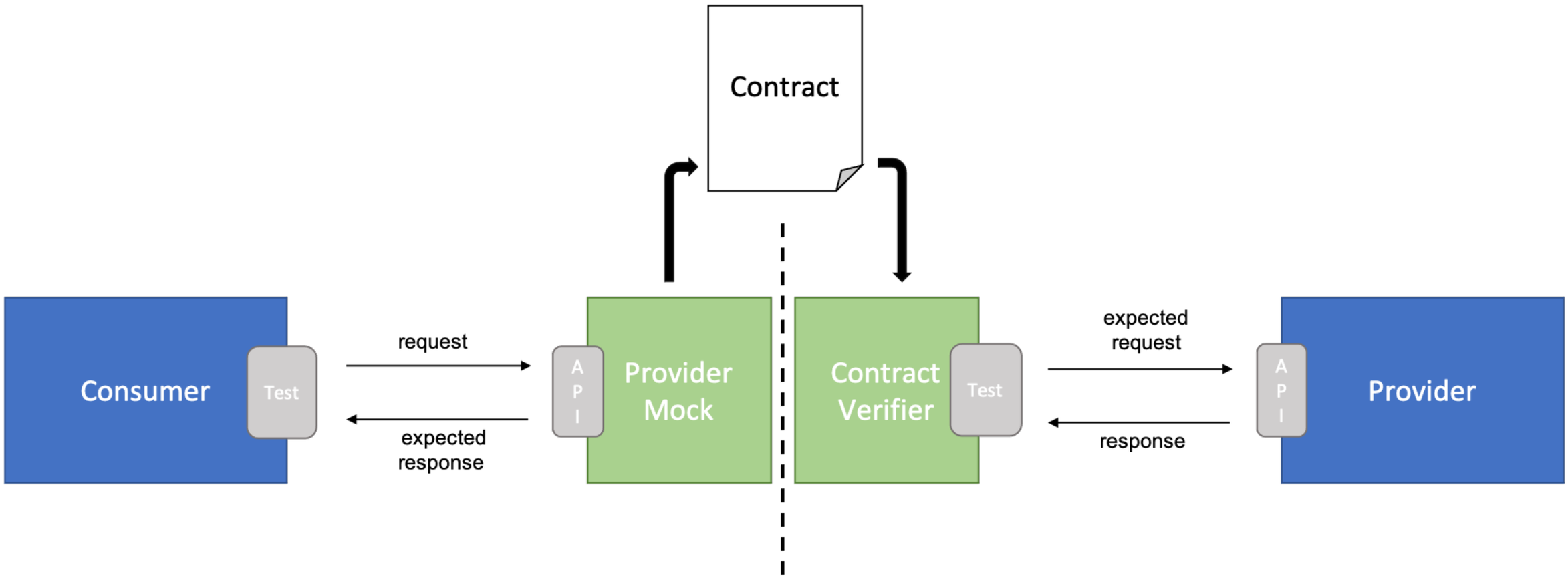
3. Provider pickup contracts to verify



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

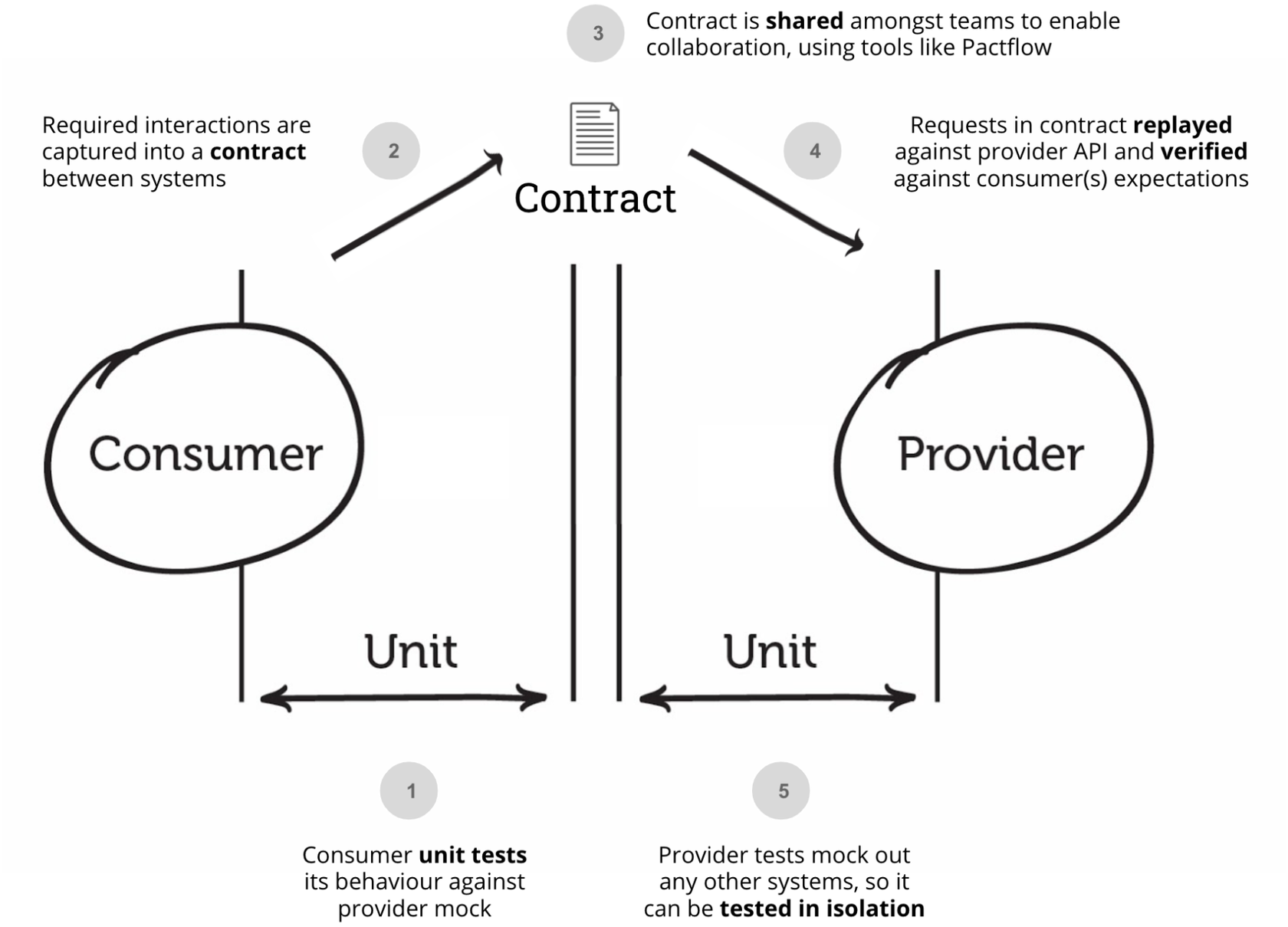
OpenAPI/ Swagger

Karate DSL

Postman

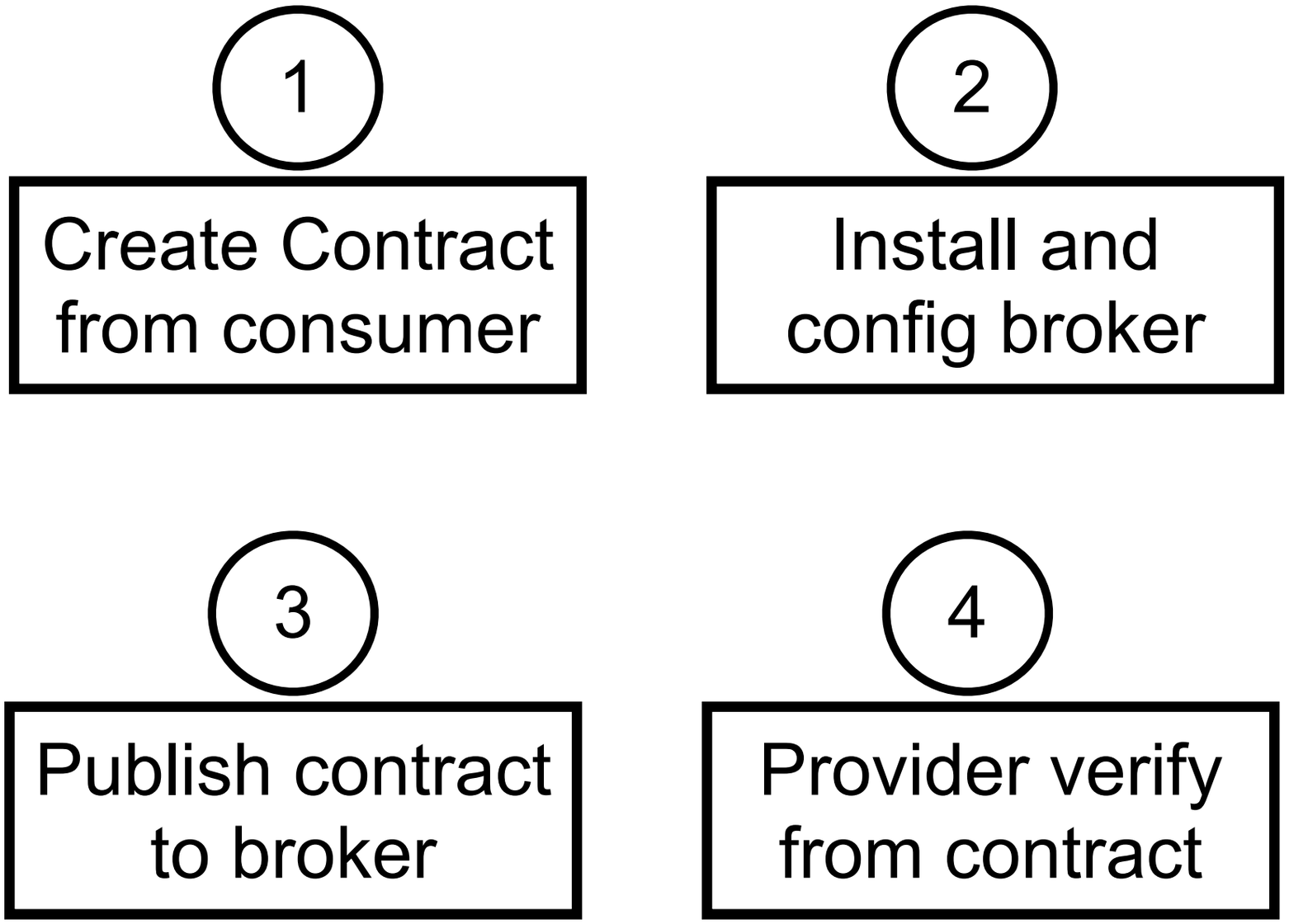
Workshop

Workshop with Pact



https://docs.pact.io/

Workshop with Pact

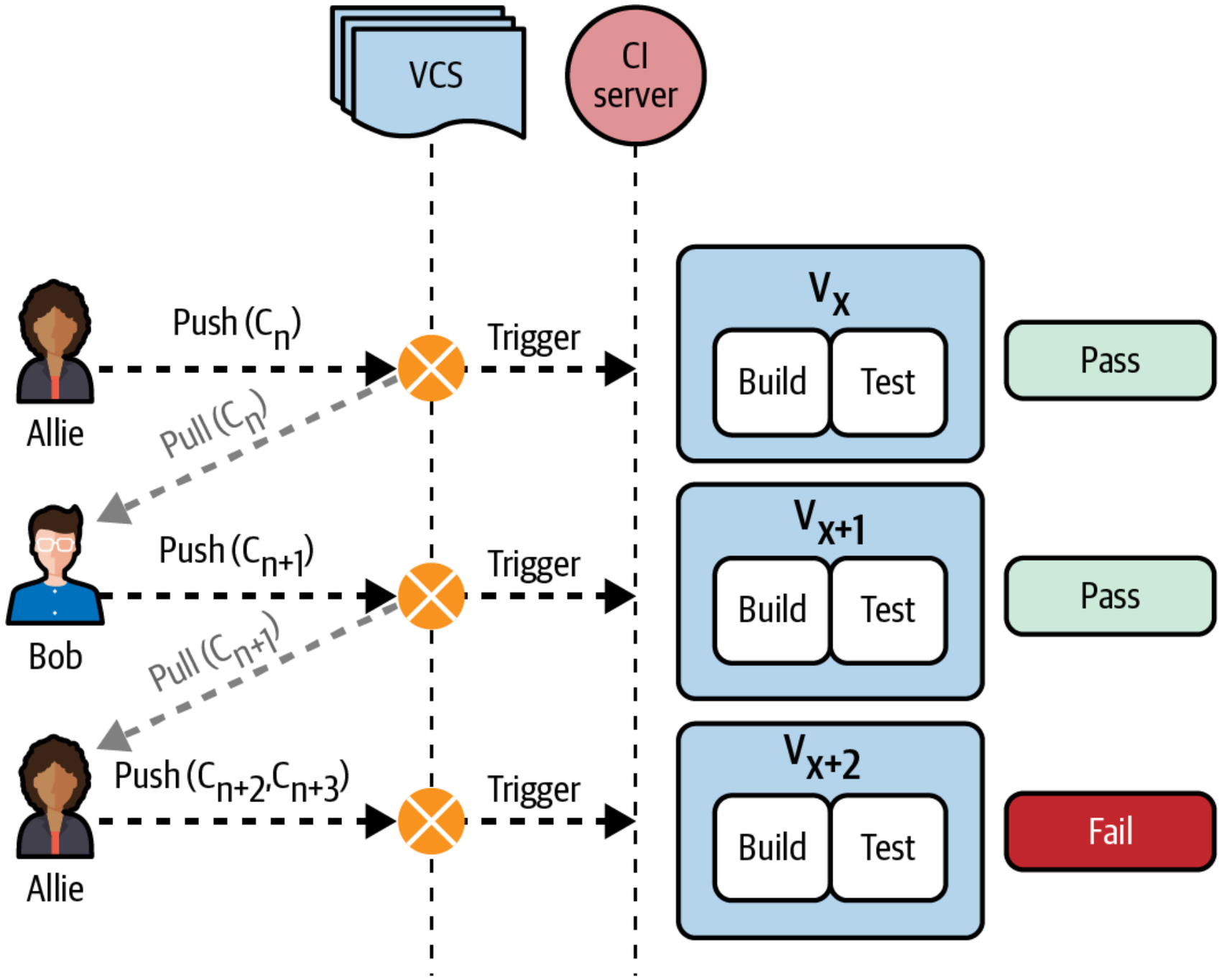


https://docs.pact.io/

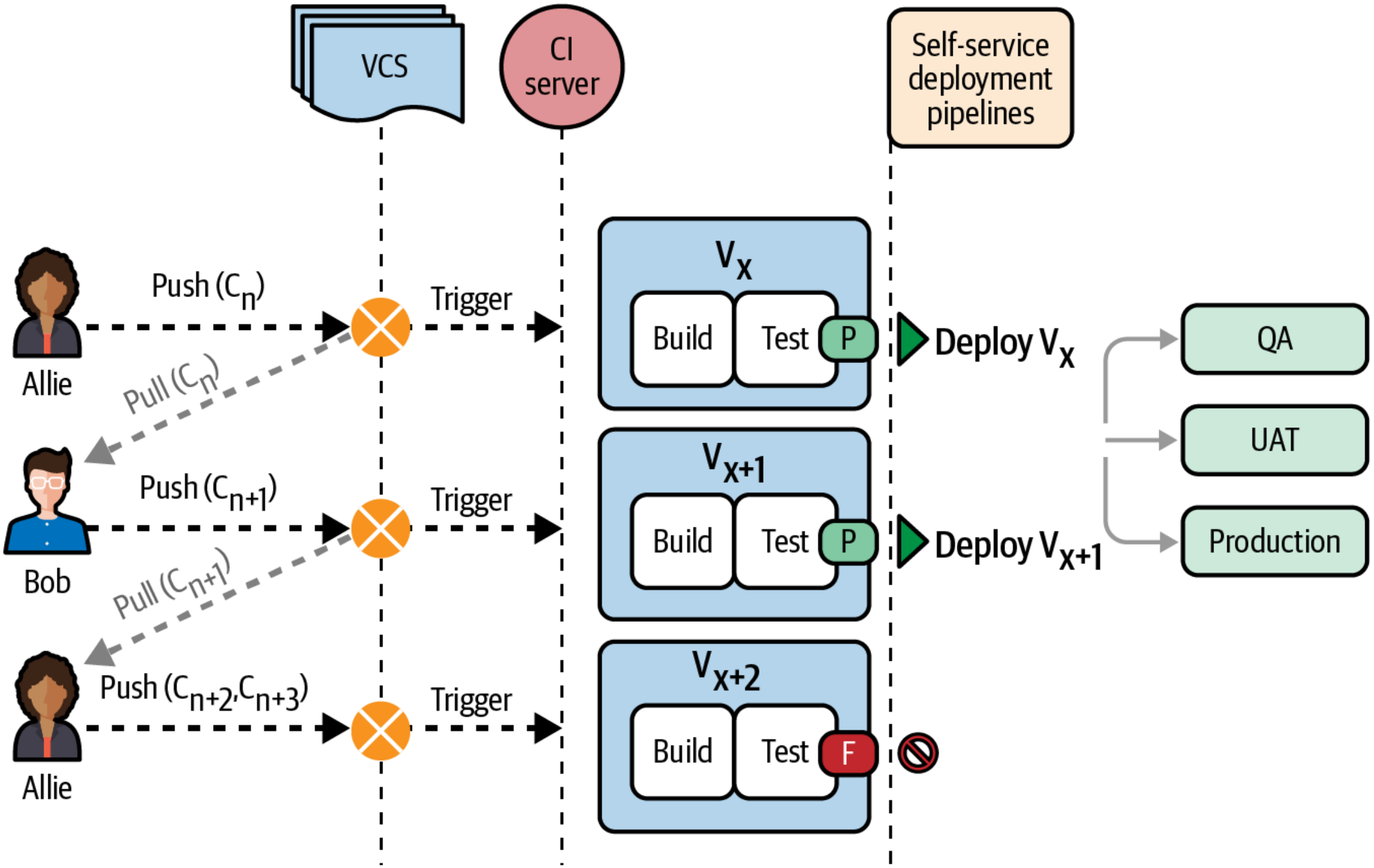
Continuous Integration

Continuous integration (CI)

Continuous Integration (CI)



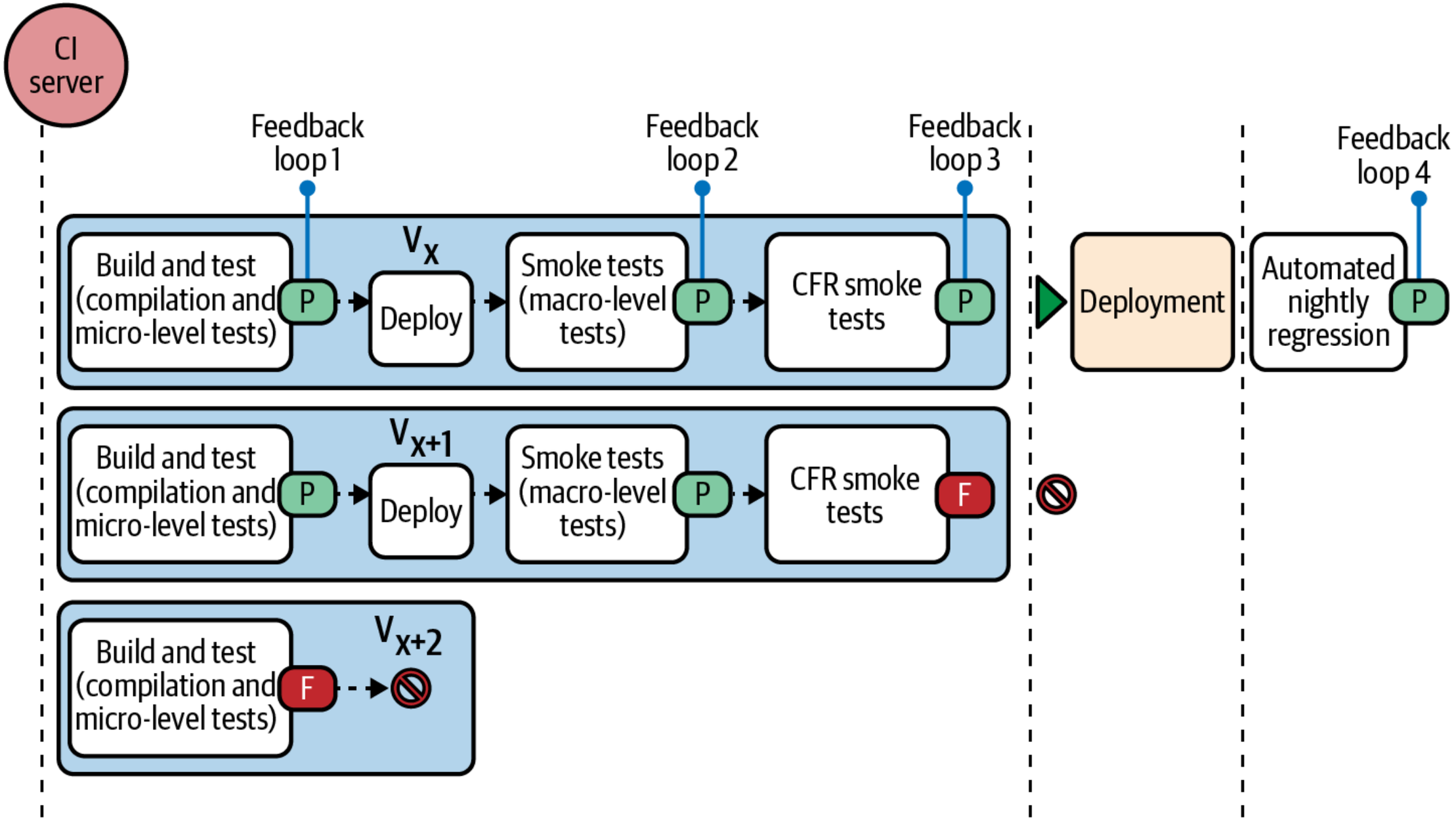
Continuous Delivery (CD)



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

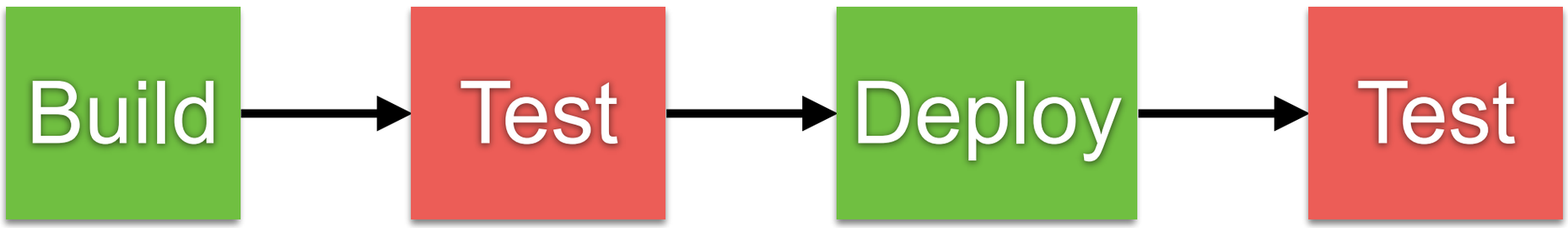
“Everyone is responsible for assuring their commits are ready for deployment”

4 key metrics to measure

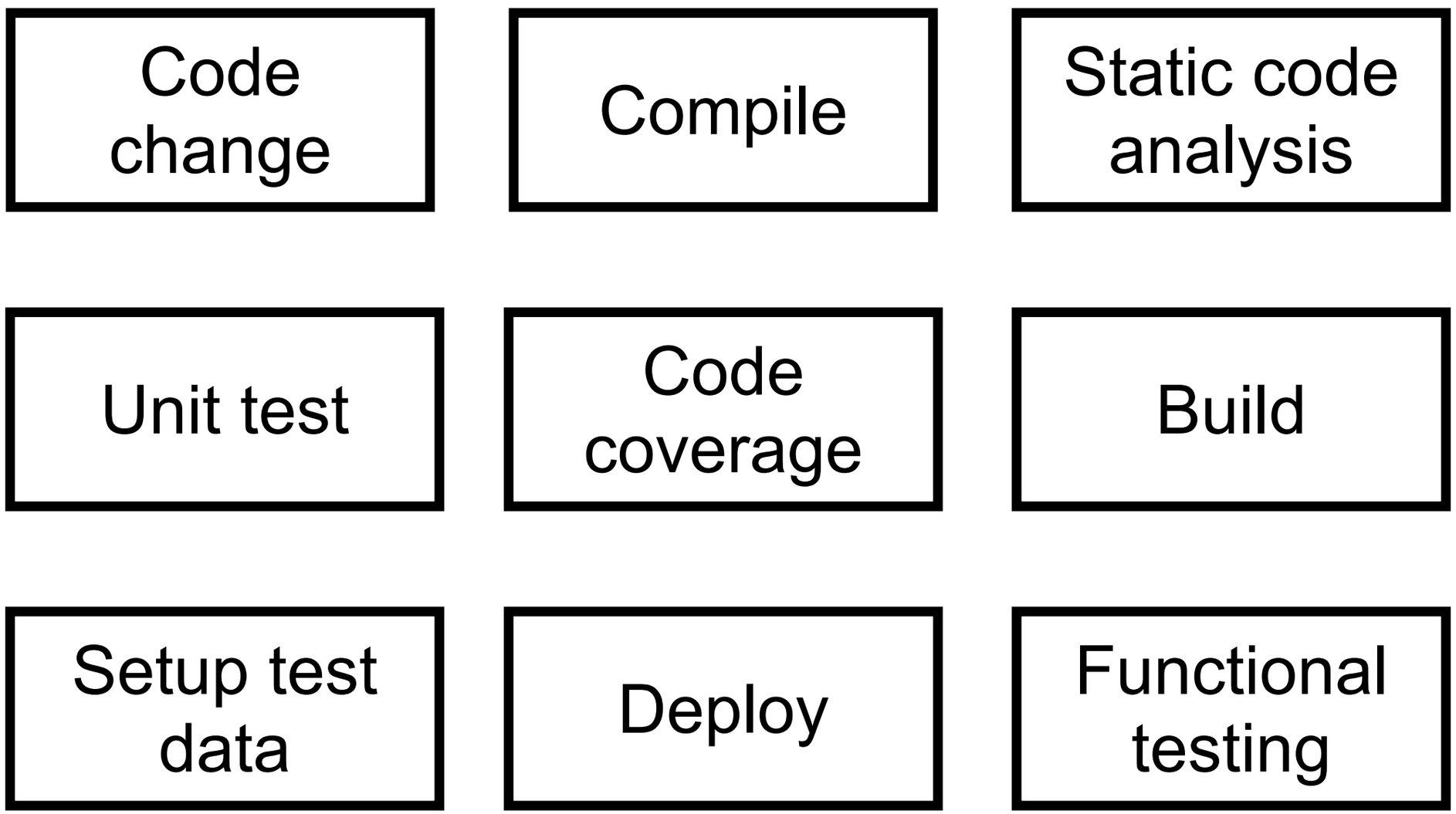
Delivery process

Lead time	Deployment frequency
Mean time to recover	Change fail percentage

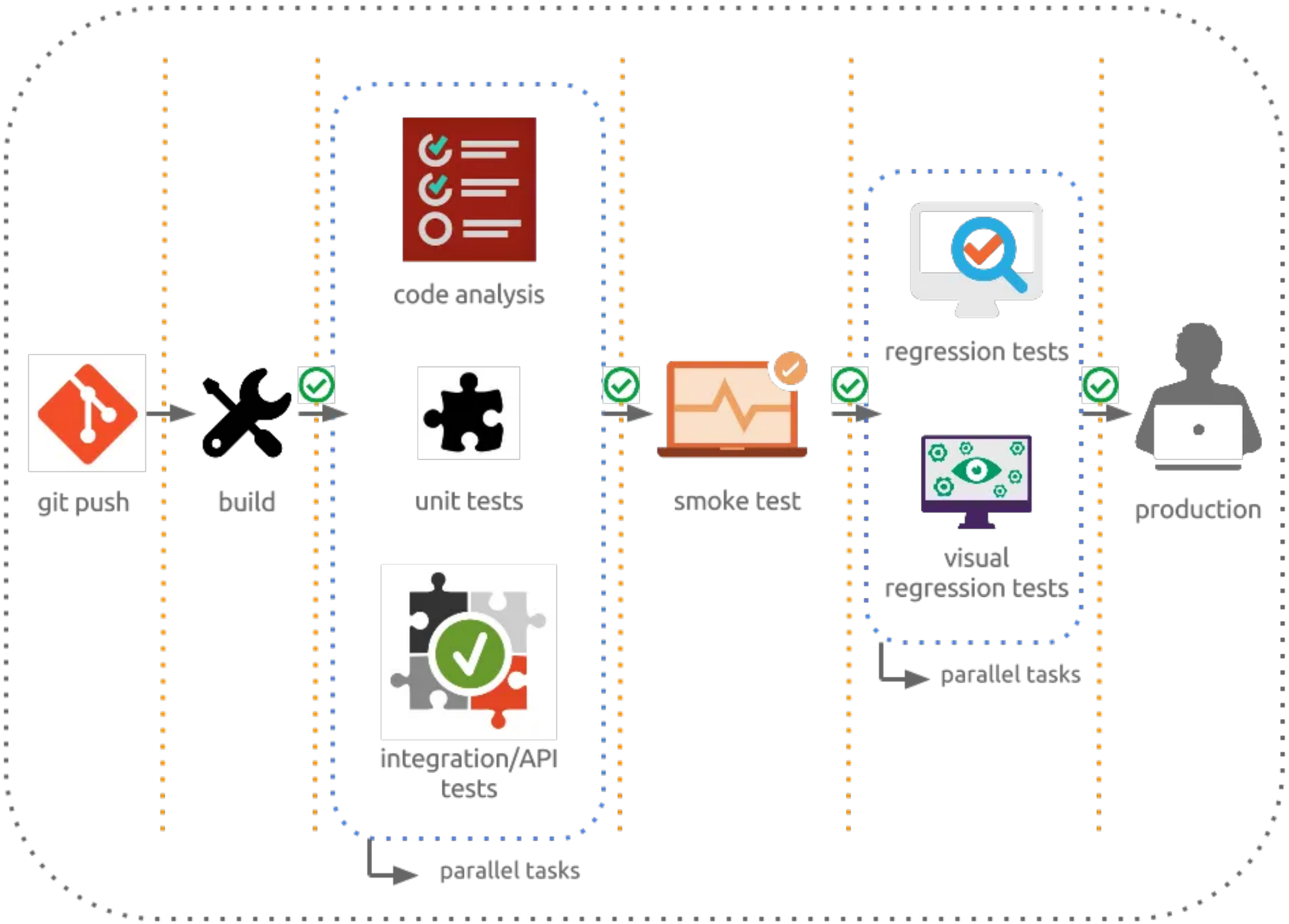
Design your pipeline



Design your pipeline/process



Design your pipeline/process



Continuous integration tools

Jenkins

GITLAB CI/CD

circleci

GitHub Actions

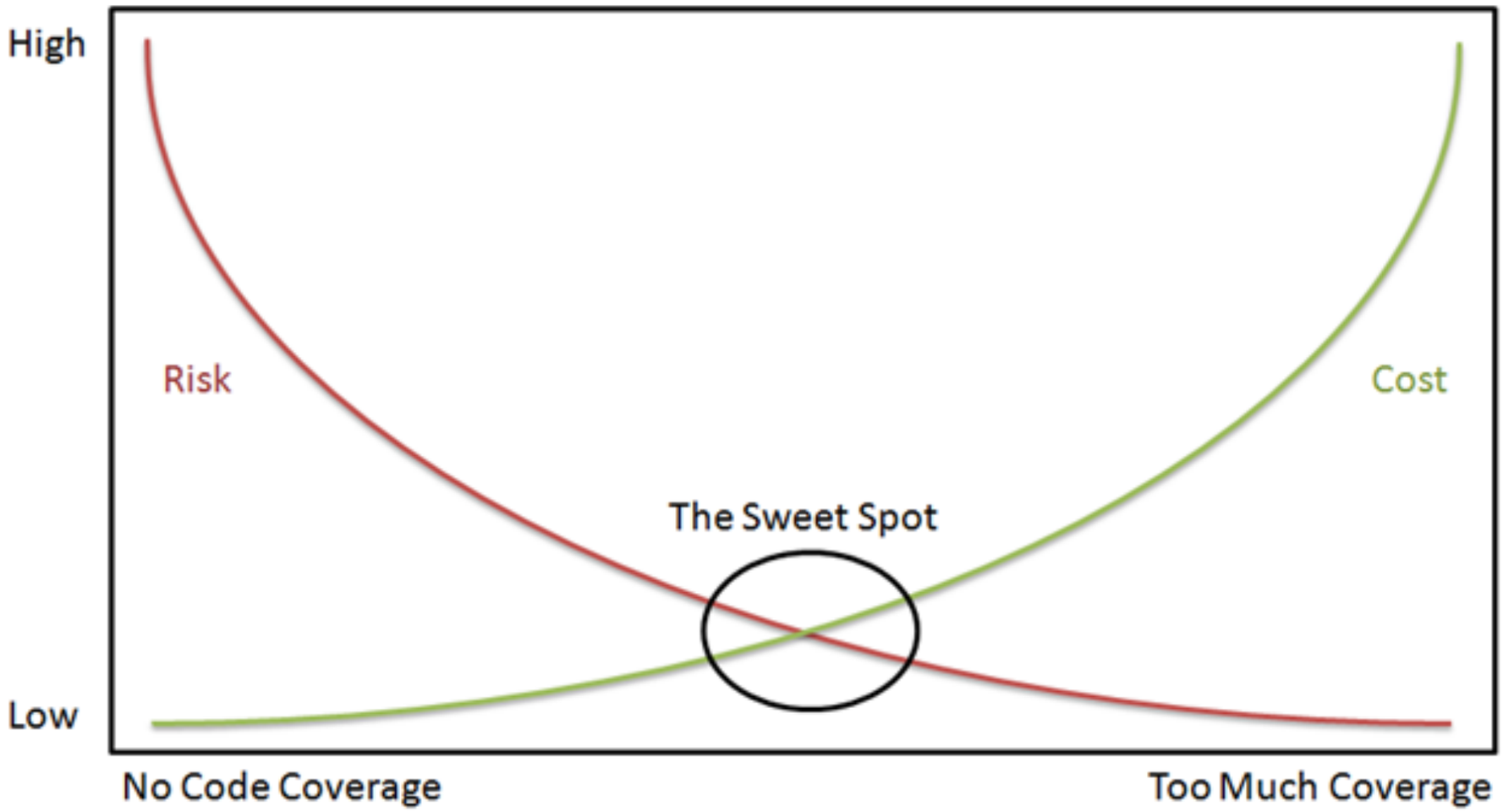
flux

argo

Workshop

Jenkins

Reduce risk with tests



Q/A