# Unit tests i PHP

Kom i gang med unit tests i PHP
https://github.com/steinmb/phpbergen_202402

@steinmb (Stein Magne Bjørklund) - Unix time: 1709139600

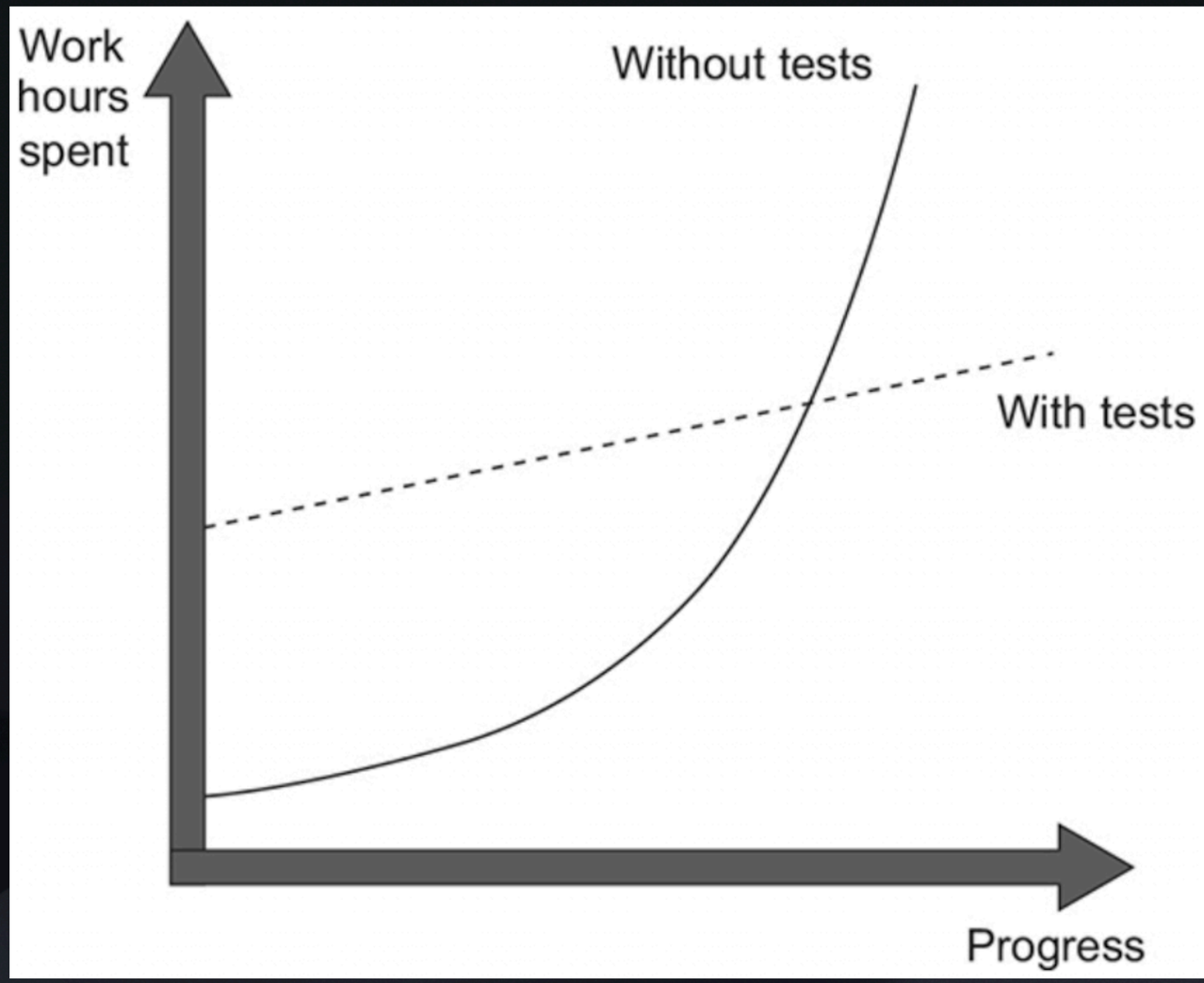
# The goal of unit testing

*"Enable sustainable growth of the software project"*

*— K. Vladimir*

Changes Breaks Everything

*regression bugs*

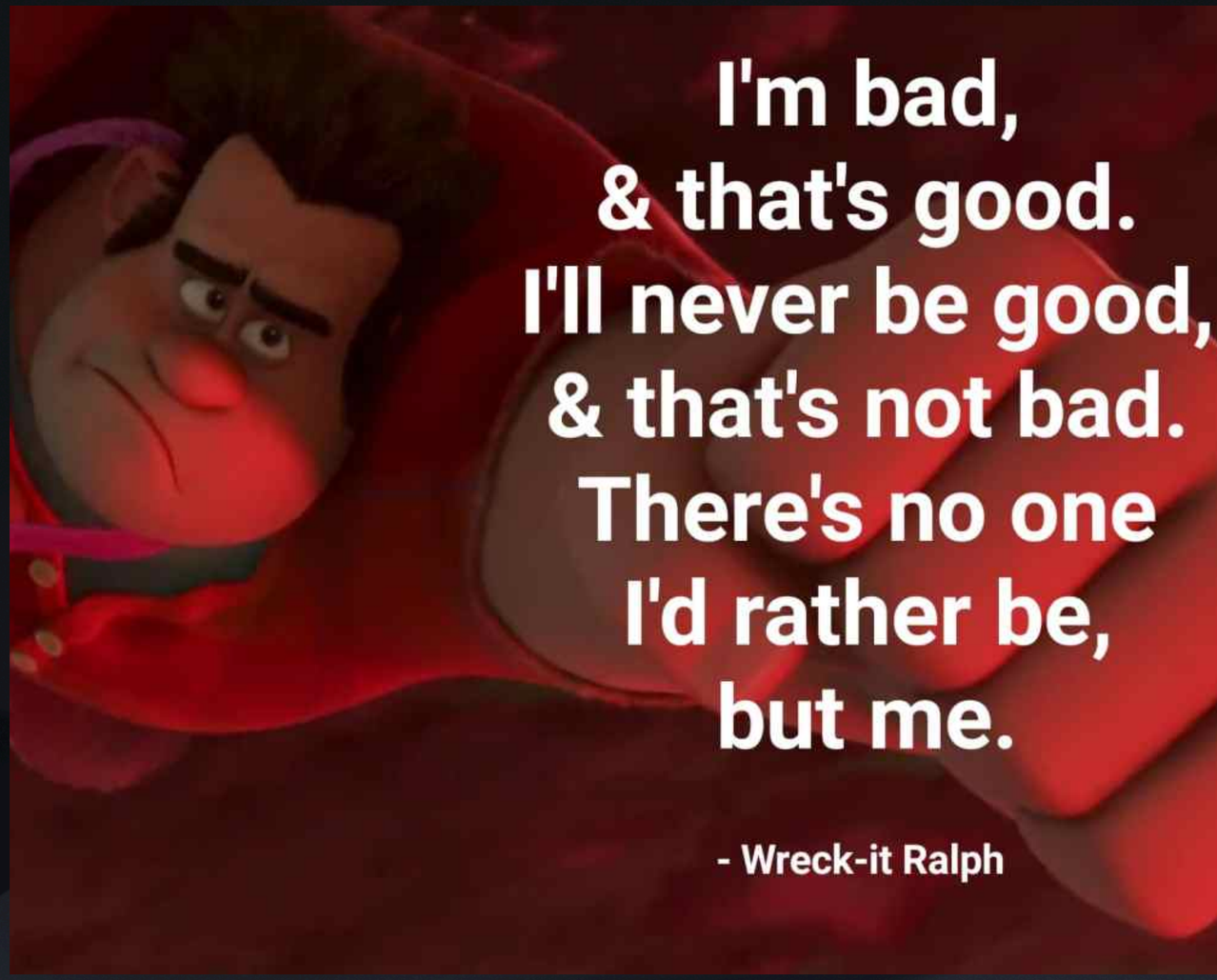


# Code quality indication

Hard to test === poor quality

Tight copeling

Easy to test !== good quality



# Good and bad tests

Code is liability, not an asset

Tests are code, too.

- Needs to be maintained.
  - Refactoring the test when your refactor the underlying code.

# PHP testing frameworks

https://pestphp.com/

composer require pestphp/pest --dev --with-all-dependencies

https://phpunit.de/

composer require --dev phpunit/phpunit

# Lets code: 1

# Your first test

## \PHPUnit\Framework\TestCase

- a type of assertion
- an expected value
- the value generated by the test
- an optional message to be displayed when the test fails

```php
public function testAddition(): void
{
    self::assertSame(
      5,
      \Calculator::add(2, 3),
      'Failed addition',
    );
}
```

# Test coverage

## Code coverage metric

- Low coverage indicate you are not testing enough.
- High is no guarantee that you have good quality tests.
- Example: Calculator::add()

```
public static function add(
    int $a,
    int $b,
): int {
    return $a + $b;
}
```

# Test coverage

## Code coverage metric

- Low coverage indicate you are not testing enough.
- High is no guarantee that you have good quality tests.
- Example: Calculator::add()
- Example: Calculator::isStringLong()

```php
public static function add(
  int $a,
  int $b,
): int {
    return $a + $b;
}

public static function isStringLong(
  string $string,
): bool {
  return strlen($string) > 6;
}
```

# Test coverage

## Code coverage metric

- Low coverage indicate you are not testing enough.
- High is no guarantee that you have good quality tests.
- Example: Calculator::add()
- Example: Calculator::isStringLong()

```php
public static function isStringLong(
  string $string,
): bool {
  return strlen($string) > 6;
}

public function testLogic(): void
{
    self::assertTrue(
      Calculator::isStringLong('phpBergen'),
    );
    self::assertFalse(
      Calculator::isStringLong('php'),
    );
}
```

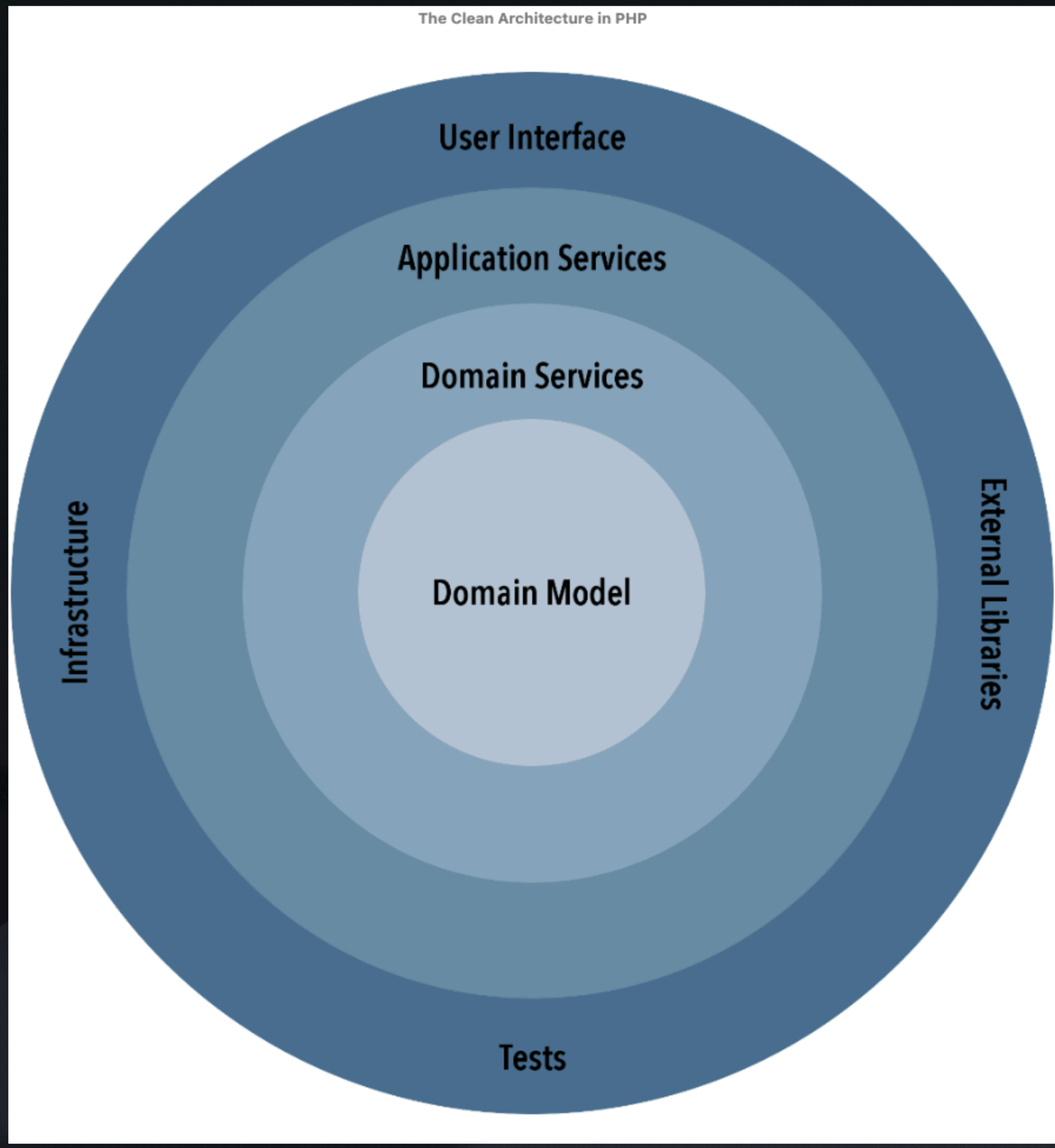
# Start here

## A successful test suite

- Maximum value, minimum maintenance cost
- Target important parts of your code
- Integrated into the dev. Cycle

# Target important parts of your code

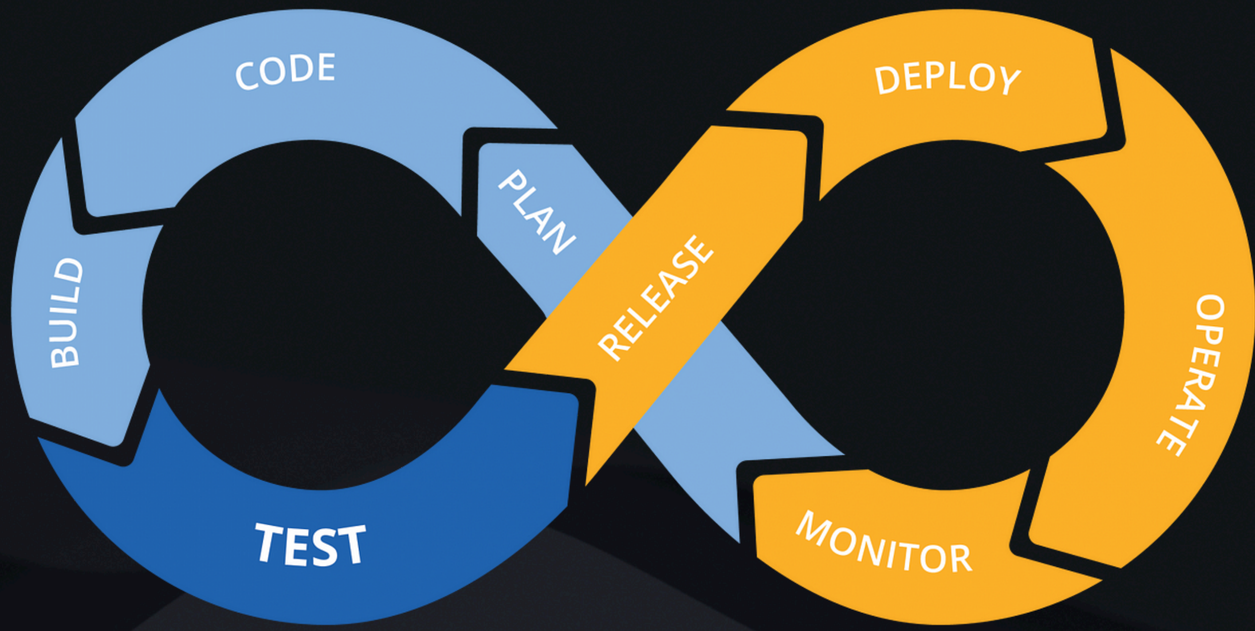
## Layered architecture



# Integrated into the dev. Cycle

## Let others run your tests

- Local runners: phpunit, IDE.
- github.com
- gitlab.com
- <include *.com>



# Lets code: 2

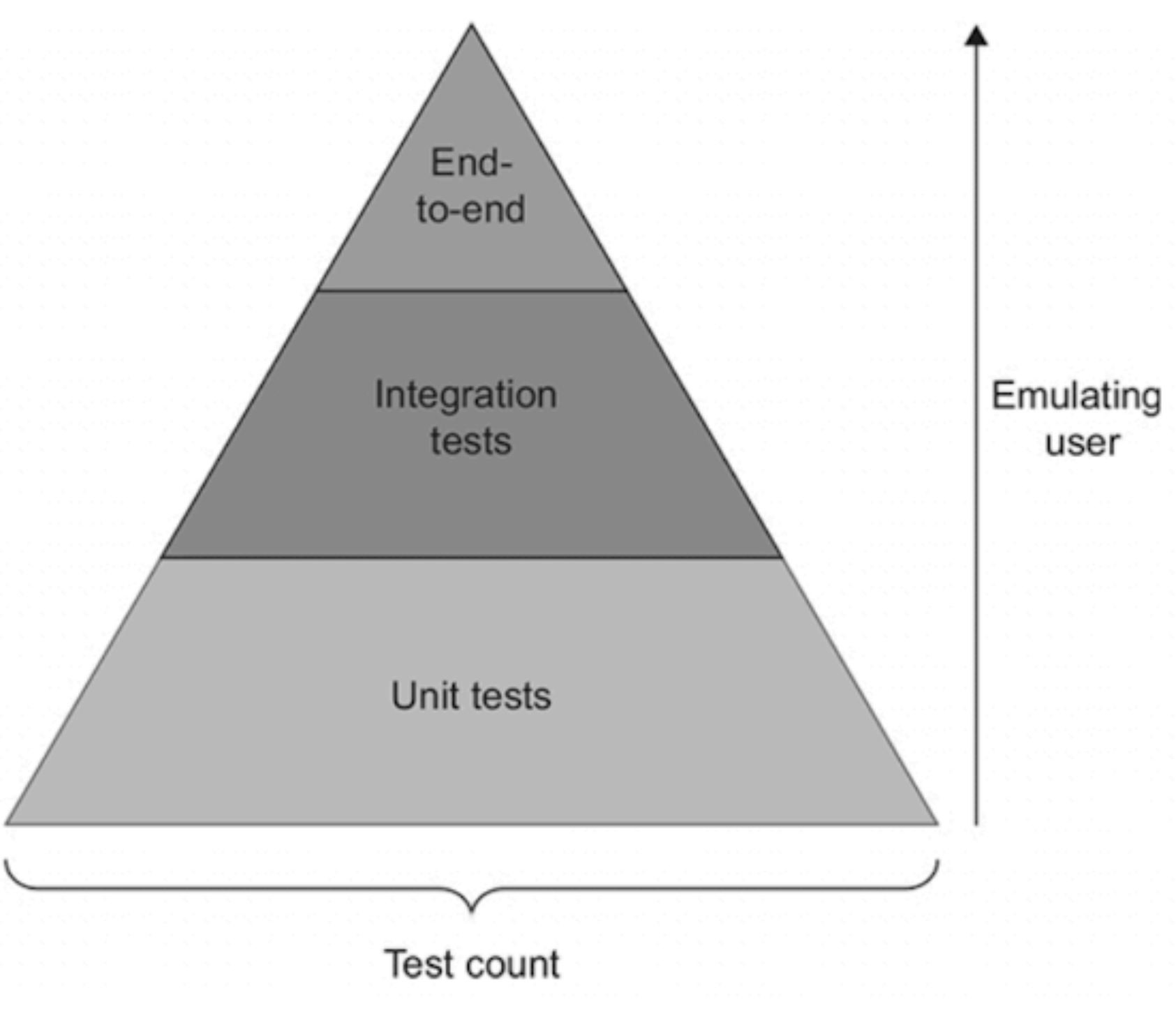
# Value object

# Services

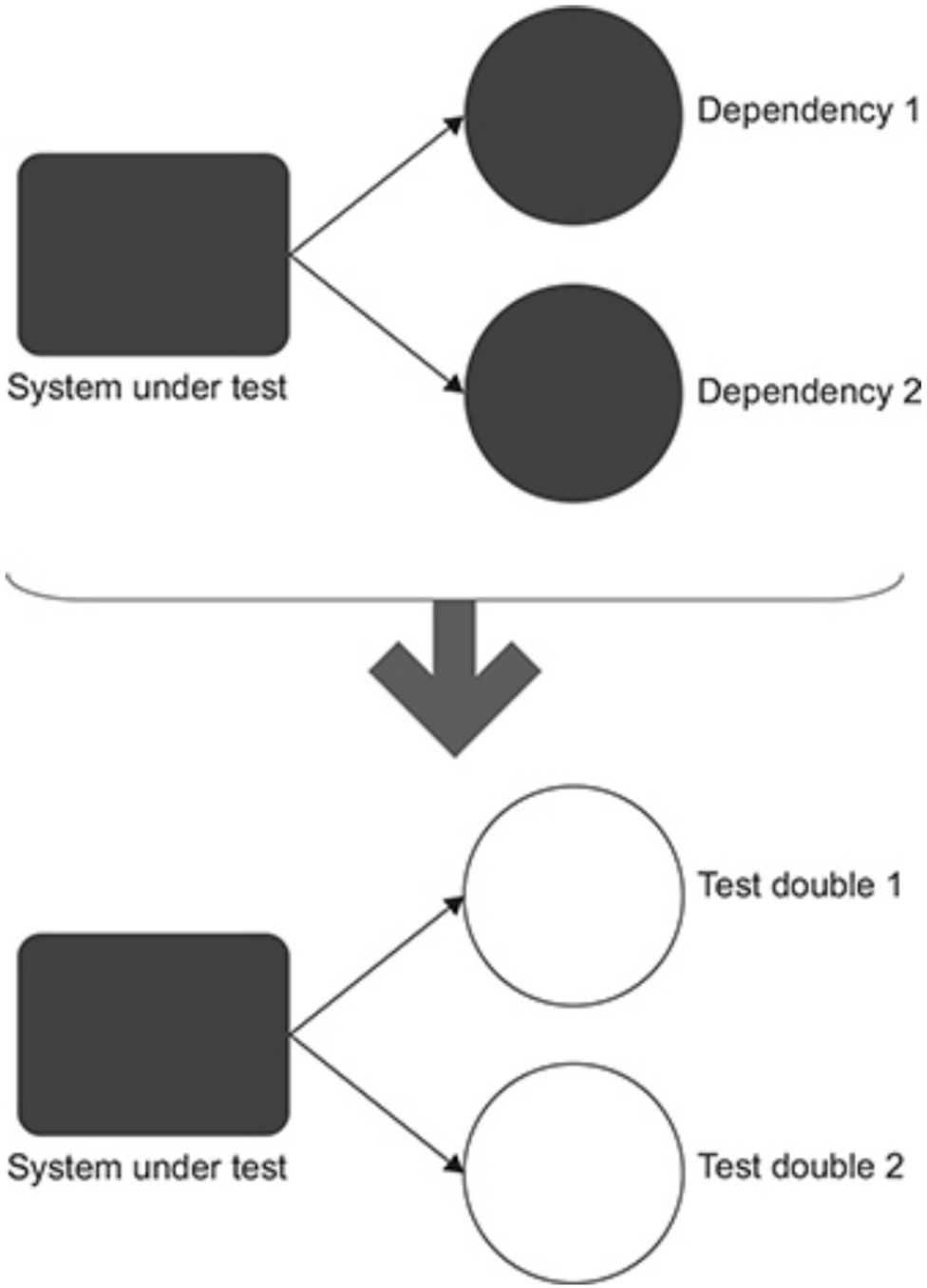
# What types of tests do we have?

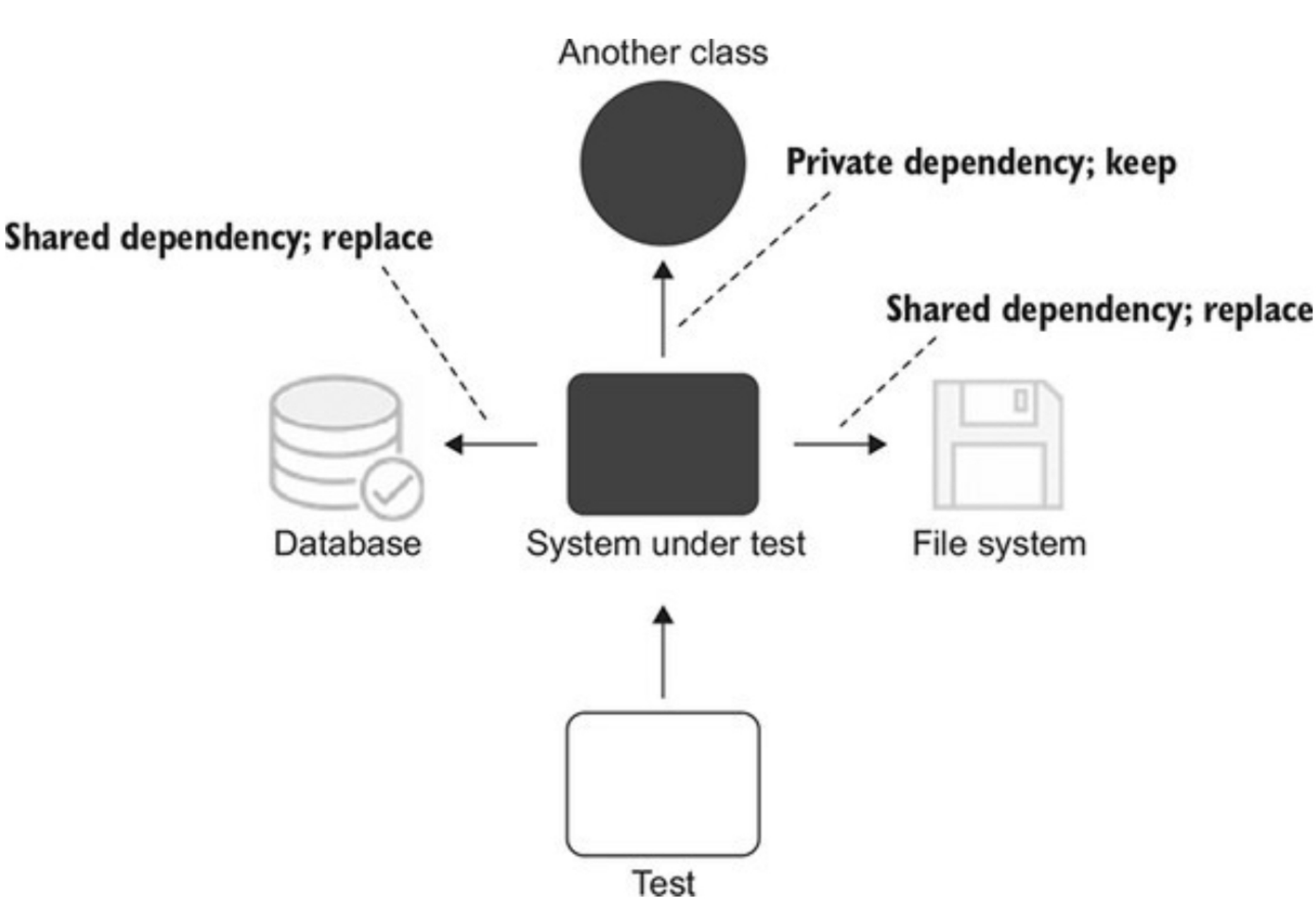
## Unit tests, integration test, end-to-end test

- Verifies a small piece of code (a unit)
- Runs quickly
- Runs in an isolated manner


End-
to-end
Integration
tests
Unit tests
Emulating
user
Test count

# London style || classic style

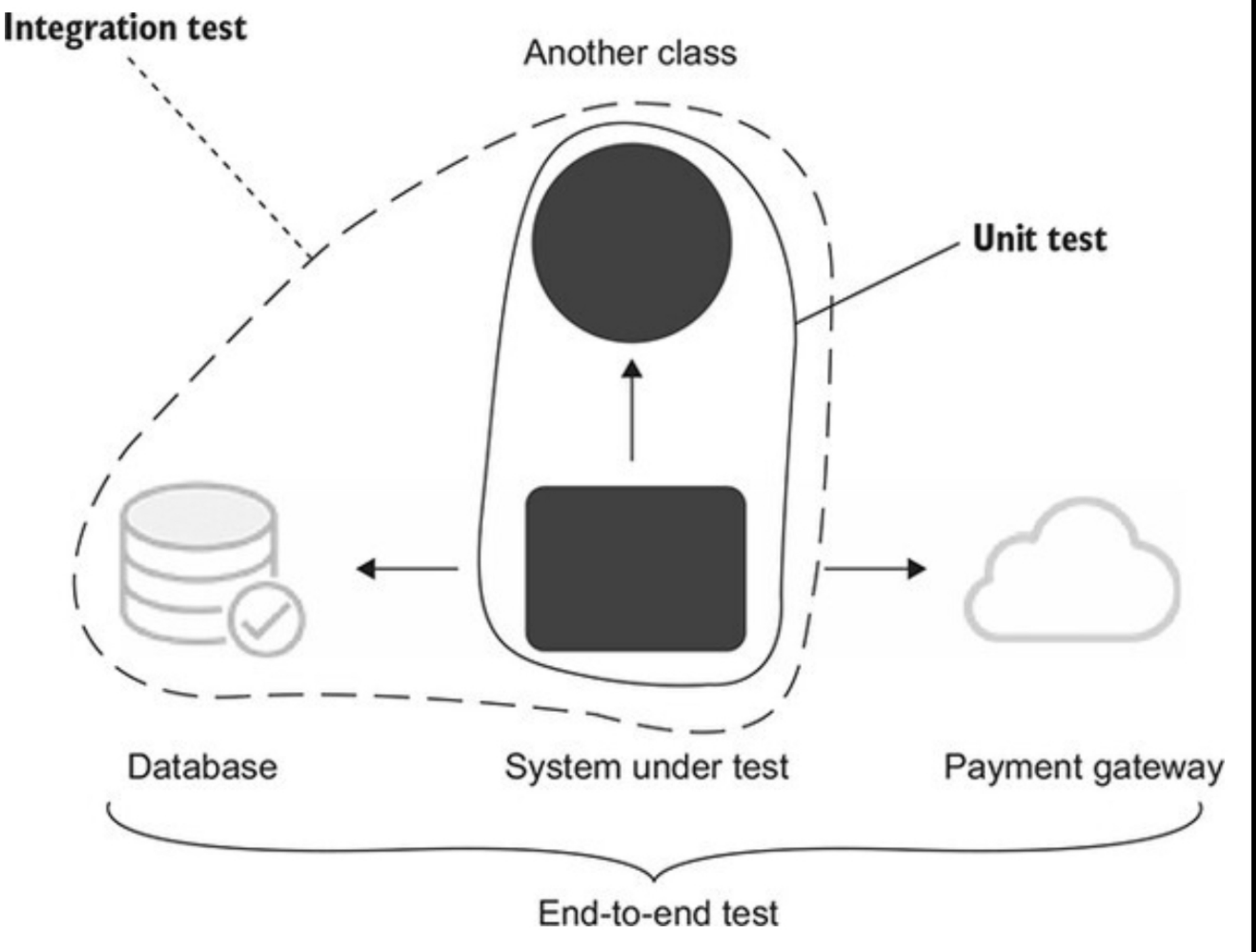

Dependency 1
System under test
Dependency 2
Test double 1
System under test
Test double 2


Another class
Private dependency; keep
Shared dependency; replace
Shared dependency; replace
Database
System under test
File system
Test

# What types of tests do we have?

## Unit tests, integration test, end-to-end test

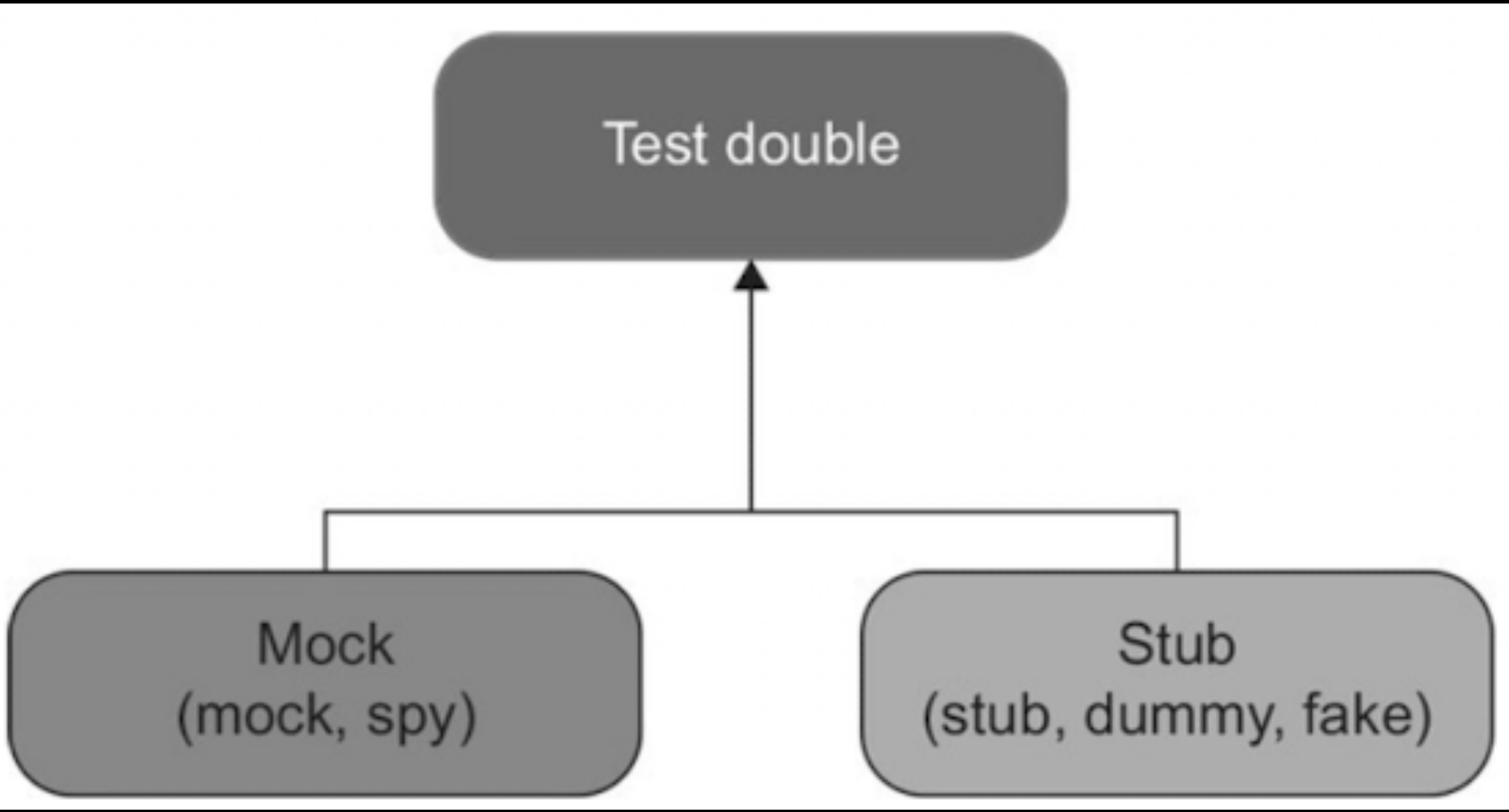
- Verifies a small piece of code (a unit)
- Runs quickly
- ~~Runs in an isolated manner~~

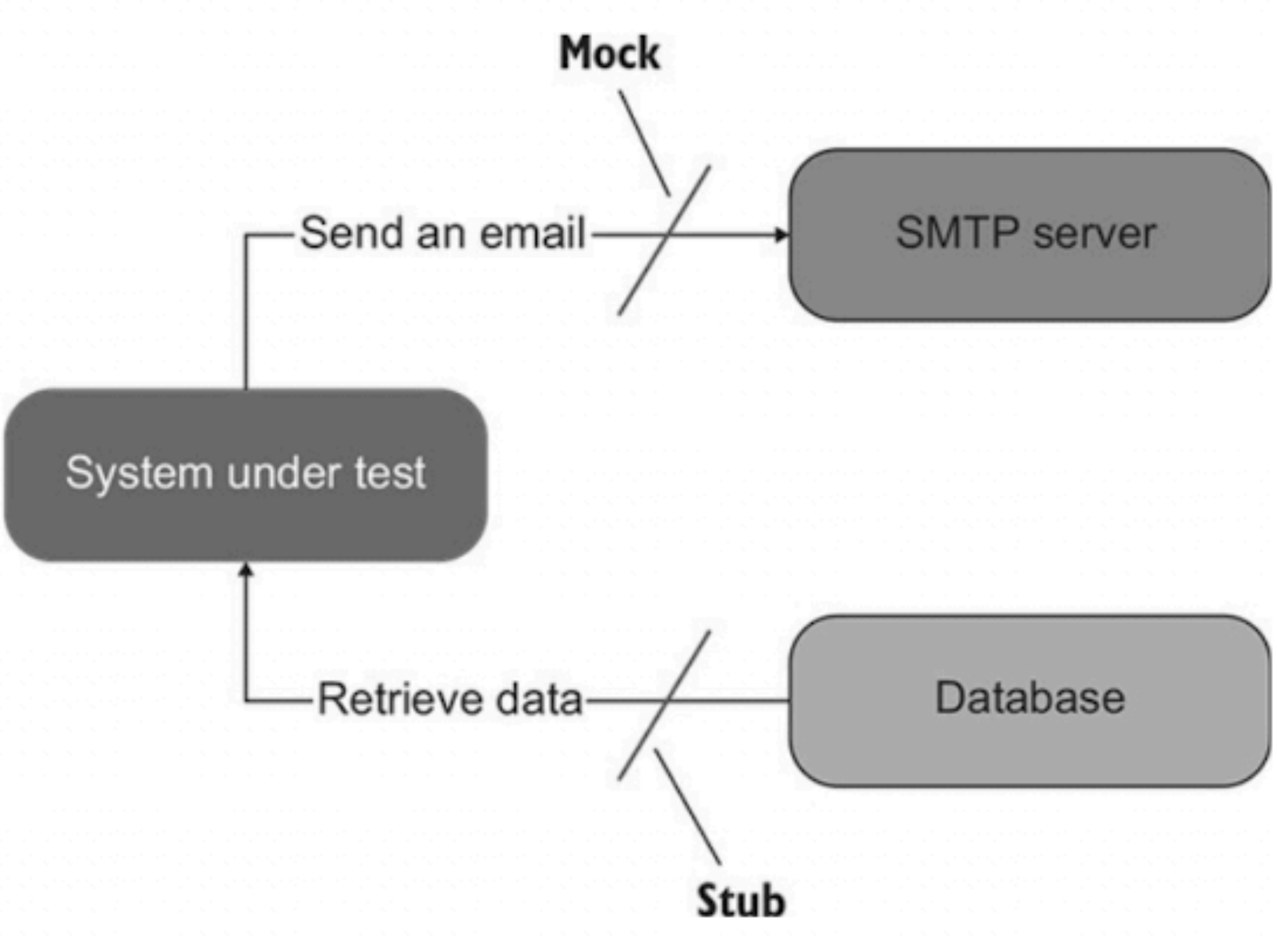

Integration test
Another class
Unit test
Database
System under test
Payment gateway
End-to-end test

# Test doubles

## dummy, stub, spy, mock, and fake

- Mocks help to emulate and examine outcoming interactions. These interactions are calls the SUT makes to its dependencies to change their state
- Stubs help to emulate incoming interactions. These interactions are calls the SUT makes to its dependencies to get input data


Test double
Mock
(mock, spy)
Stub
(stub, dummy, fake)


Mock
Send an email
SMTP server
System under test
Retrieve data
Database
Stub

# The end?

Questions and next meetup topic(s).

https://github.com/steinmb/phpbergen_202402