

Unit tests i PHP

Kom i gang med unit tests i PHP

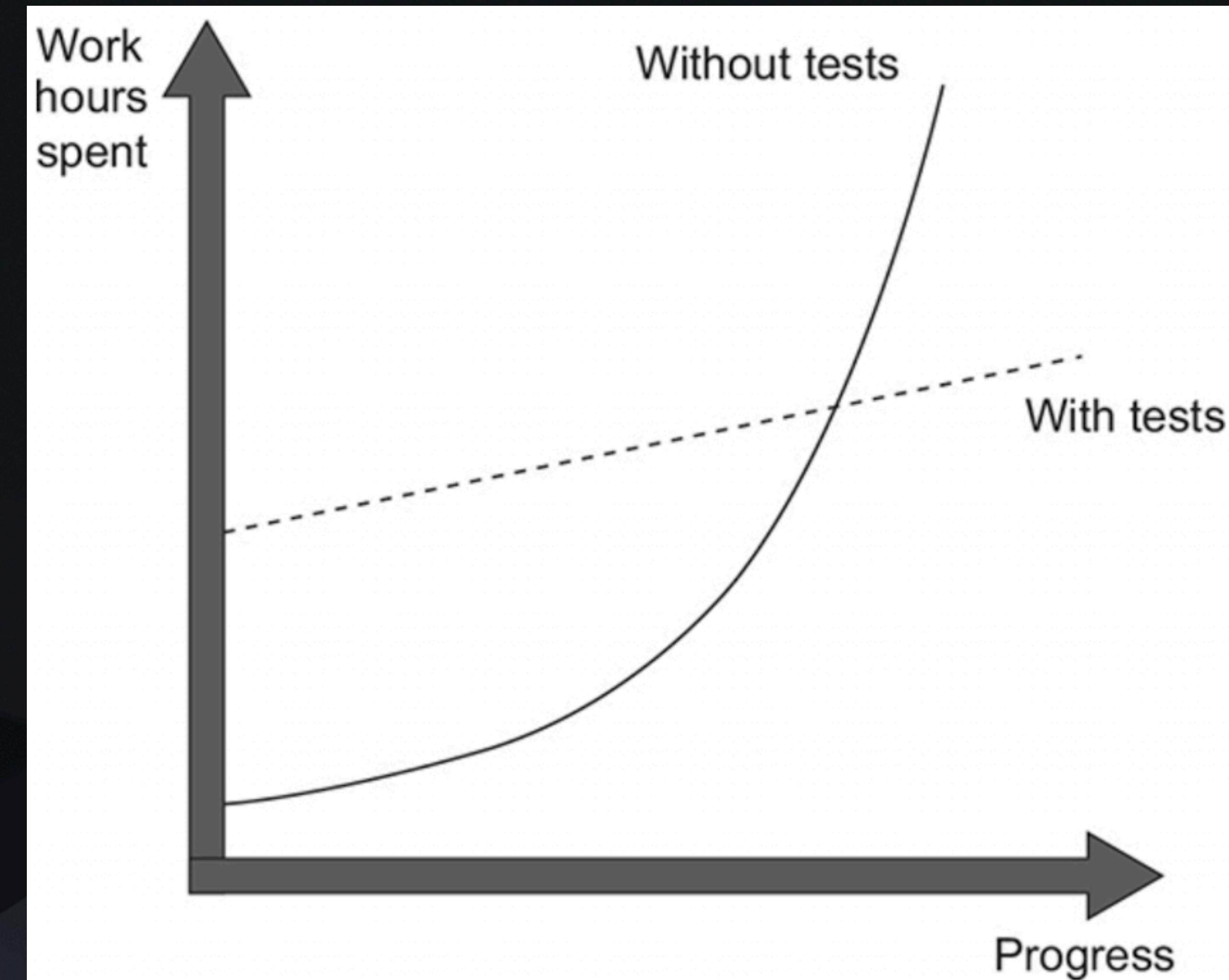
https://github.com/steinmb/phpbergen_202402

The goal of unit testing

“Enable sustainable growth of the software project”

— K. Vladimir

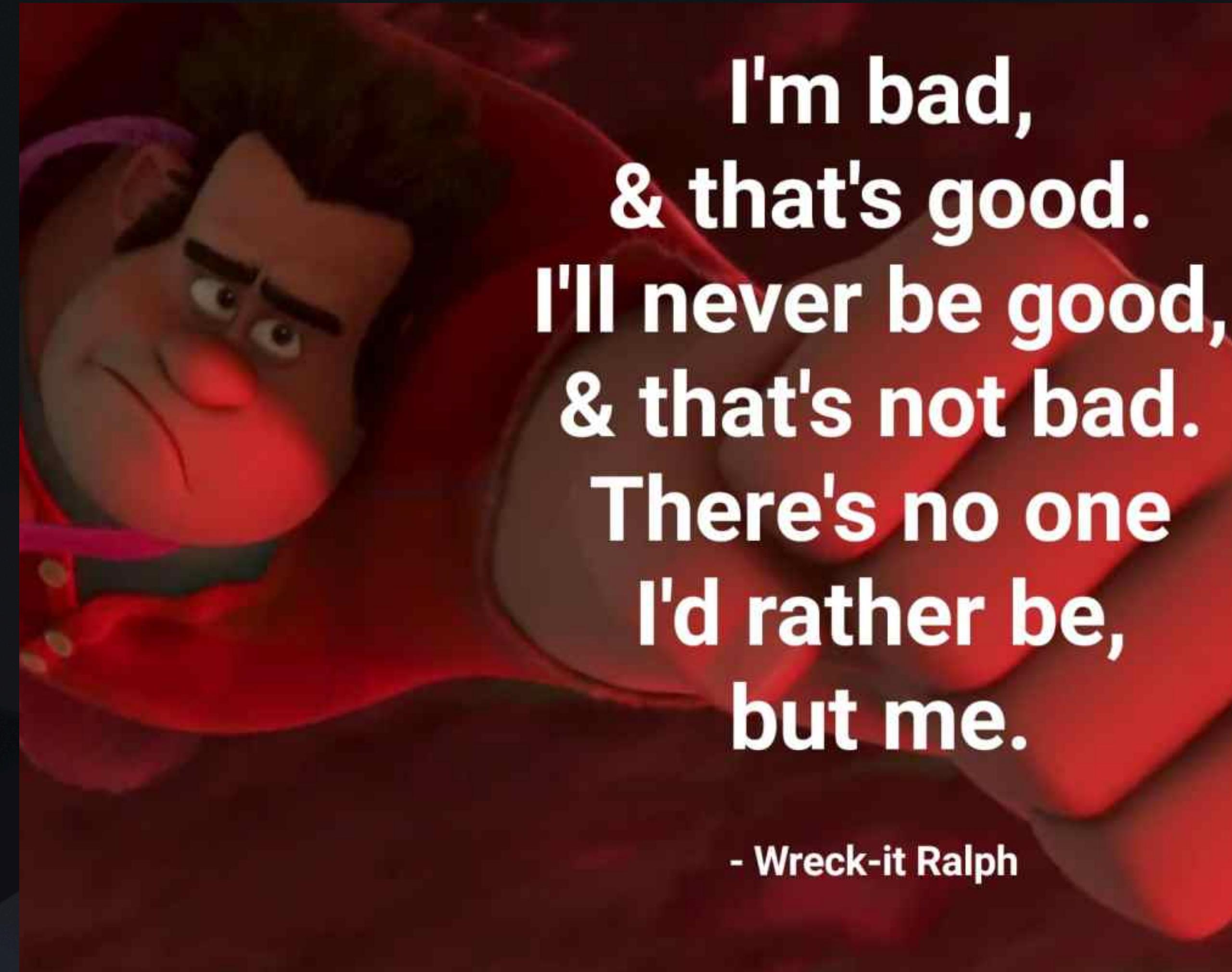
Changes Breaks Everything
regression bugs



Code quality indication

Hard to test === poor quality
Tight coupling

Easy to test != good quality



I'm bad,
& that's good.
I'll never be good,
& that's not bad.
There's no one
I'd rather be,
but me.

- Wreck-it Ralph

Good and bad tests

Code is liability, not an asset

Tests are code, too.

Needs to be maintained.

- Refactoring the test when you 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

```
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()

```
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()

```
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'),  
    );  
}
```

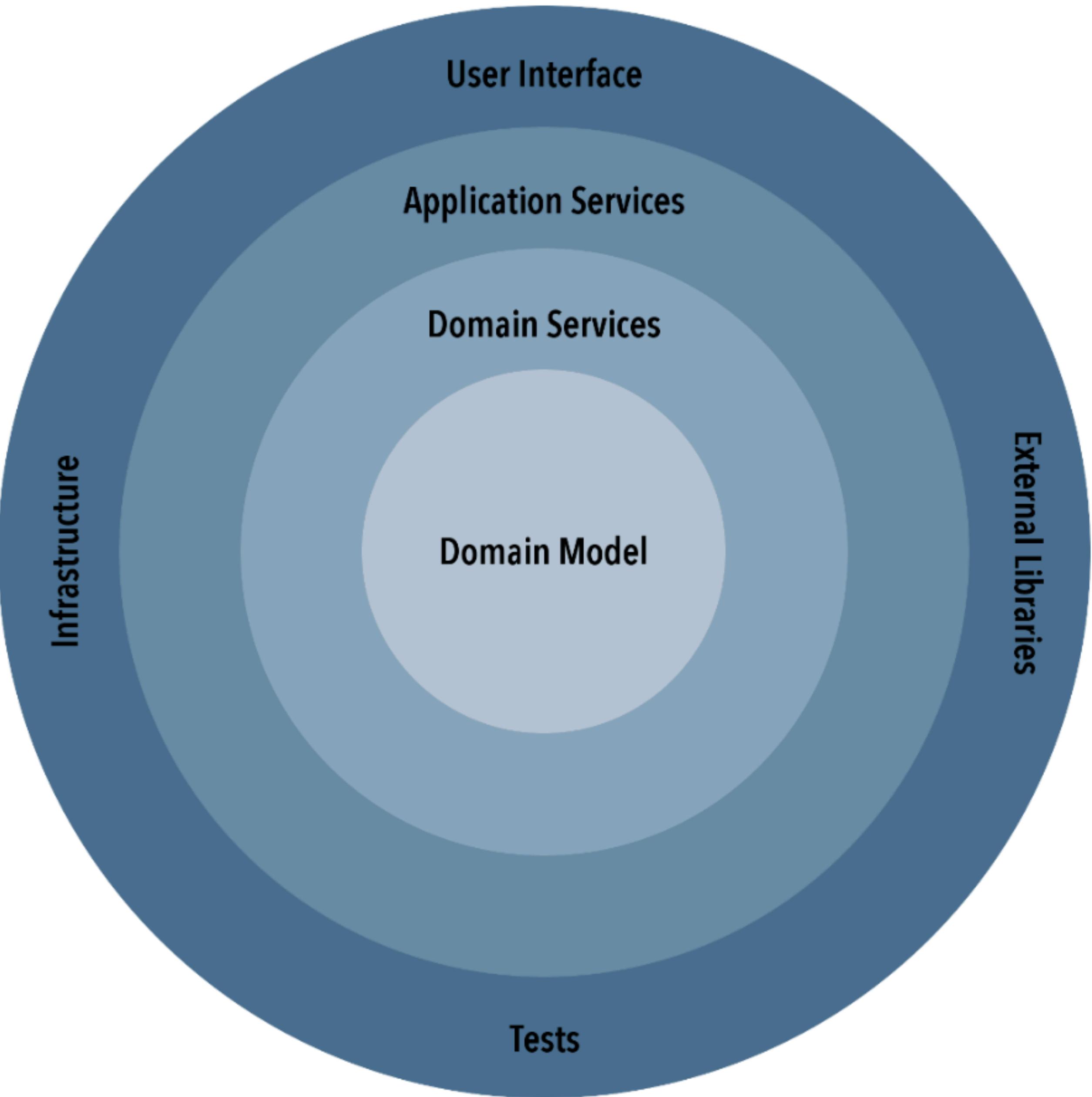
Start here

A successful test suite

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



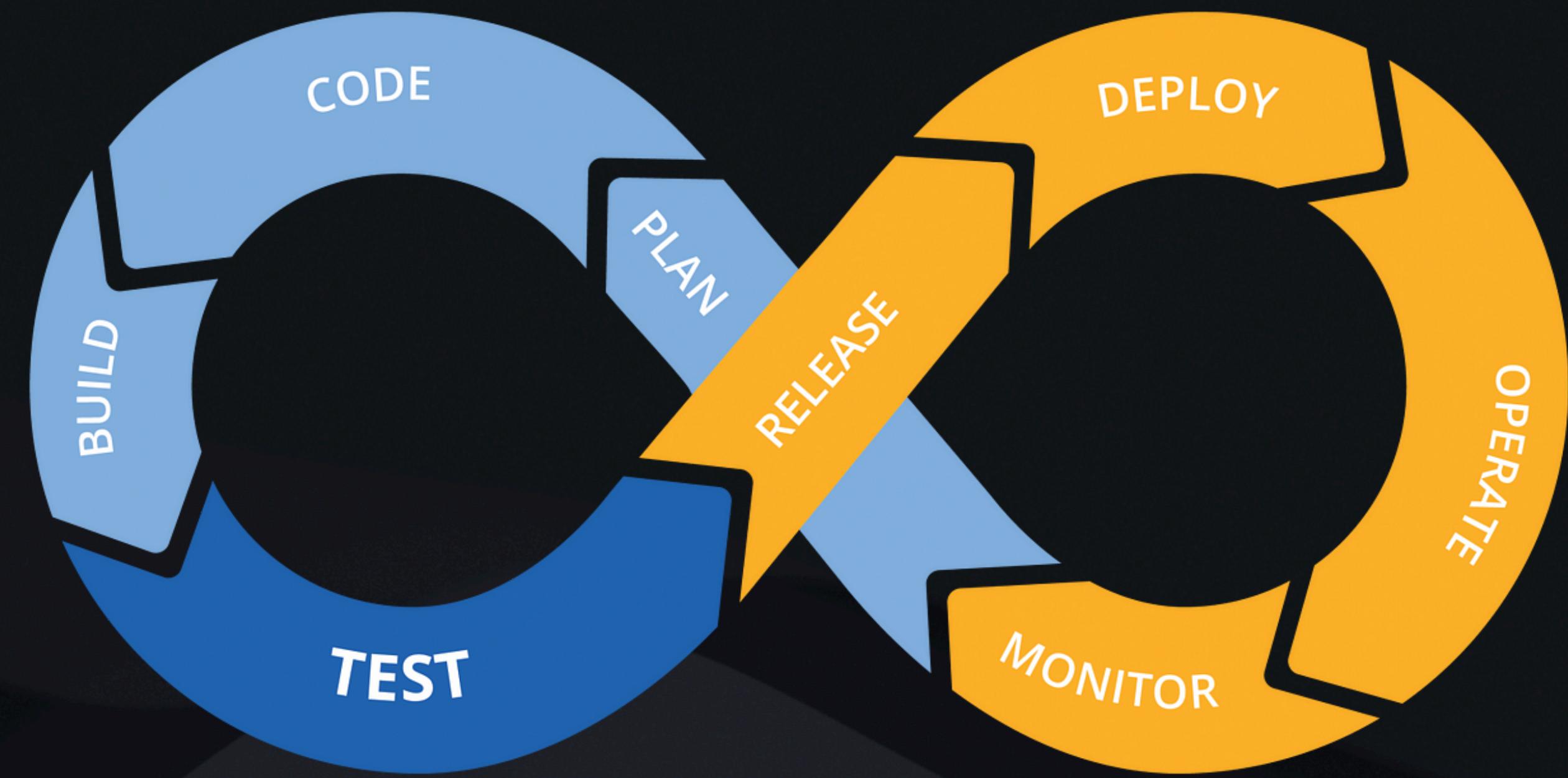
Target
important parts
of your code
Layered architecture



Integrated into the dev. Cycle

Let others run your tests

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



Lets code: 2

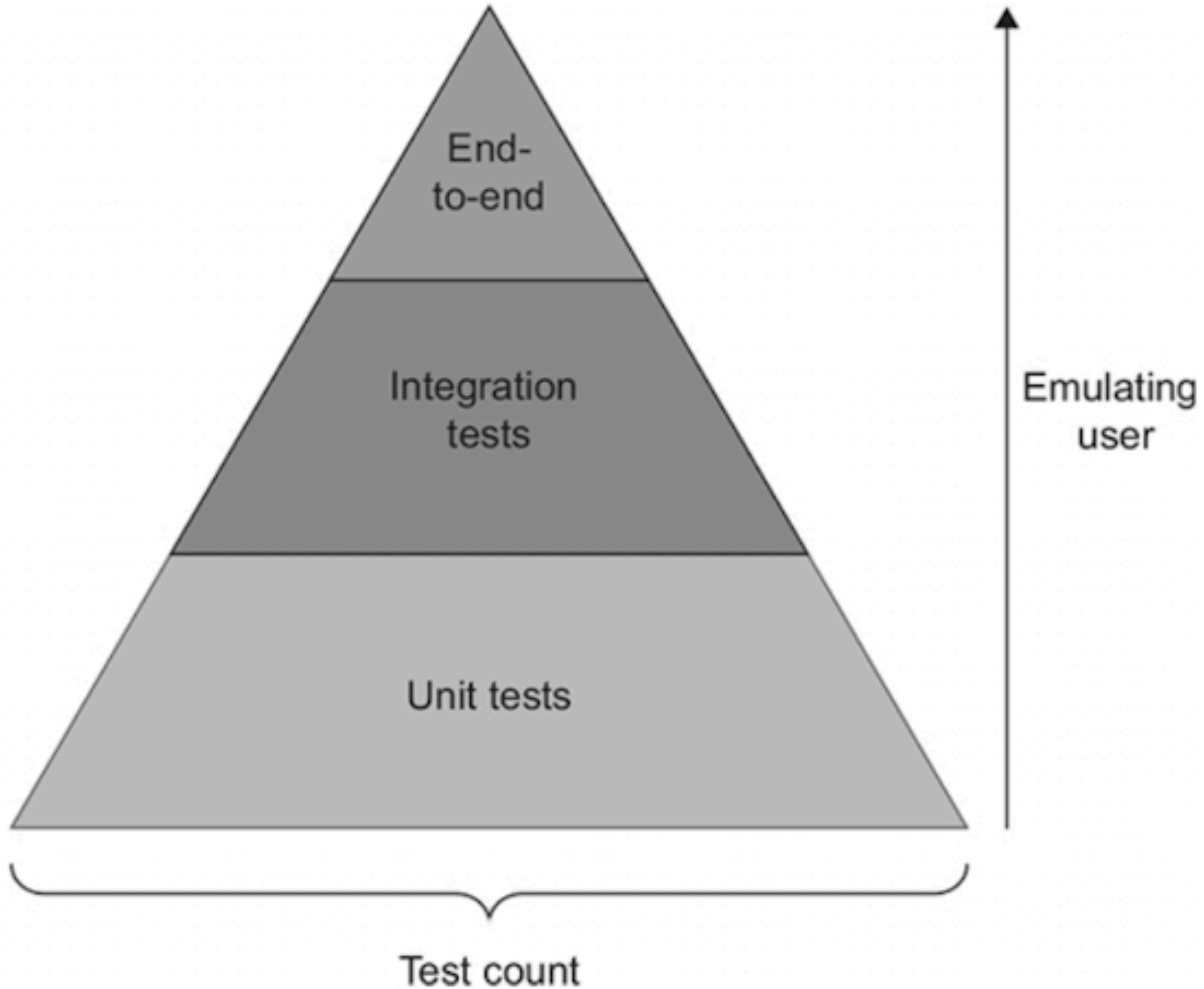
Value object

Services

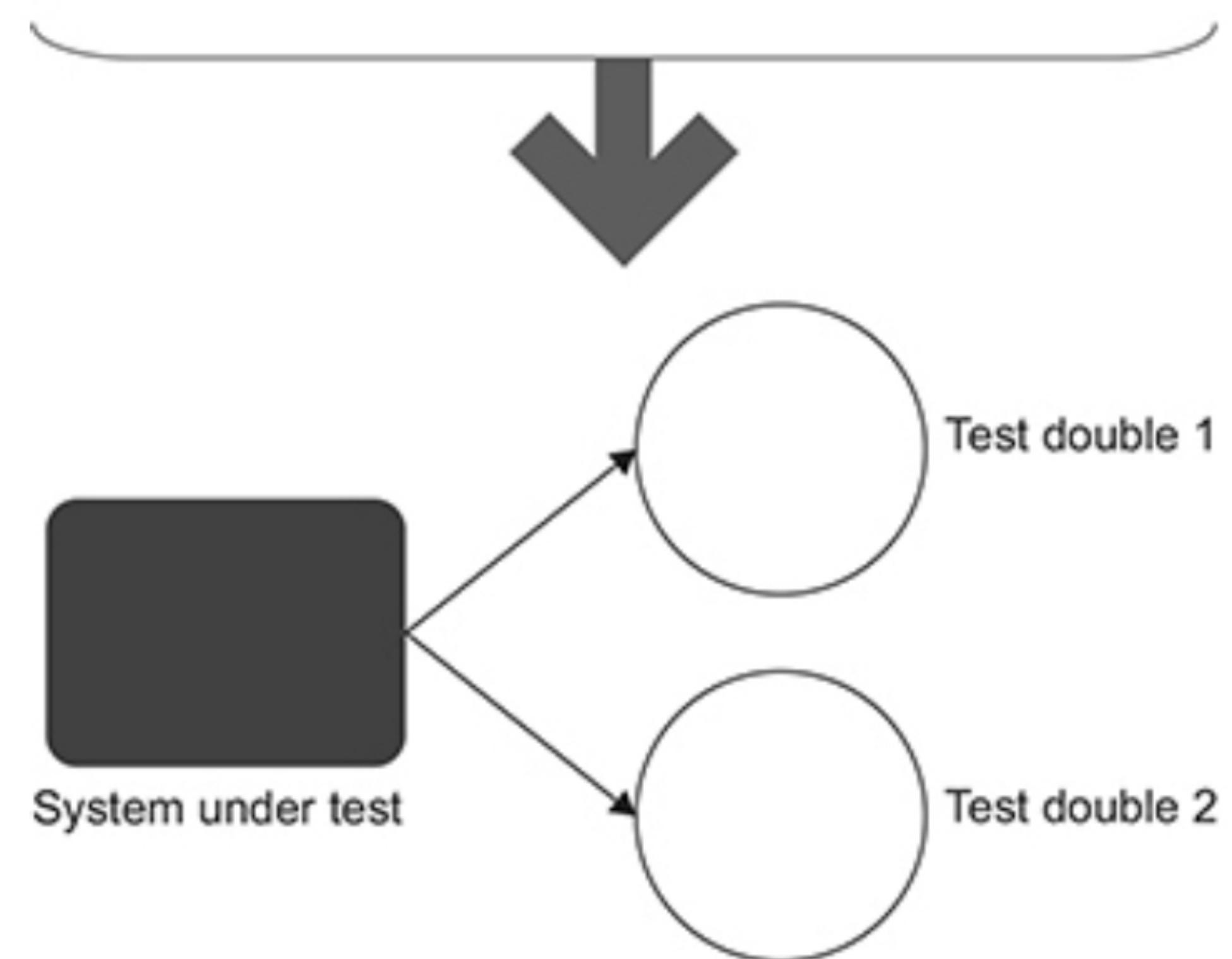
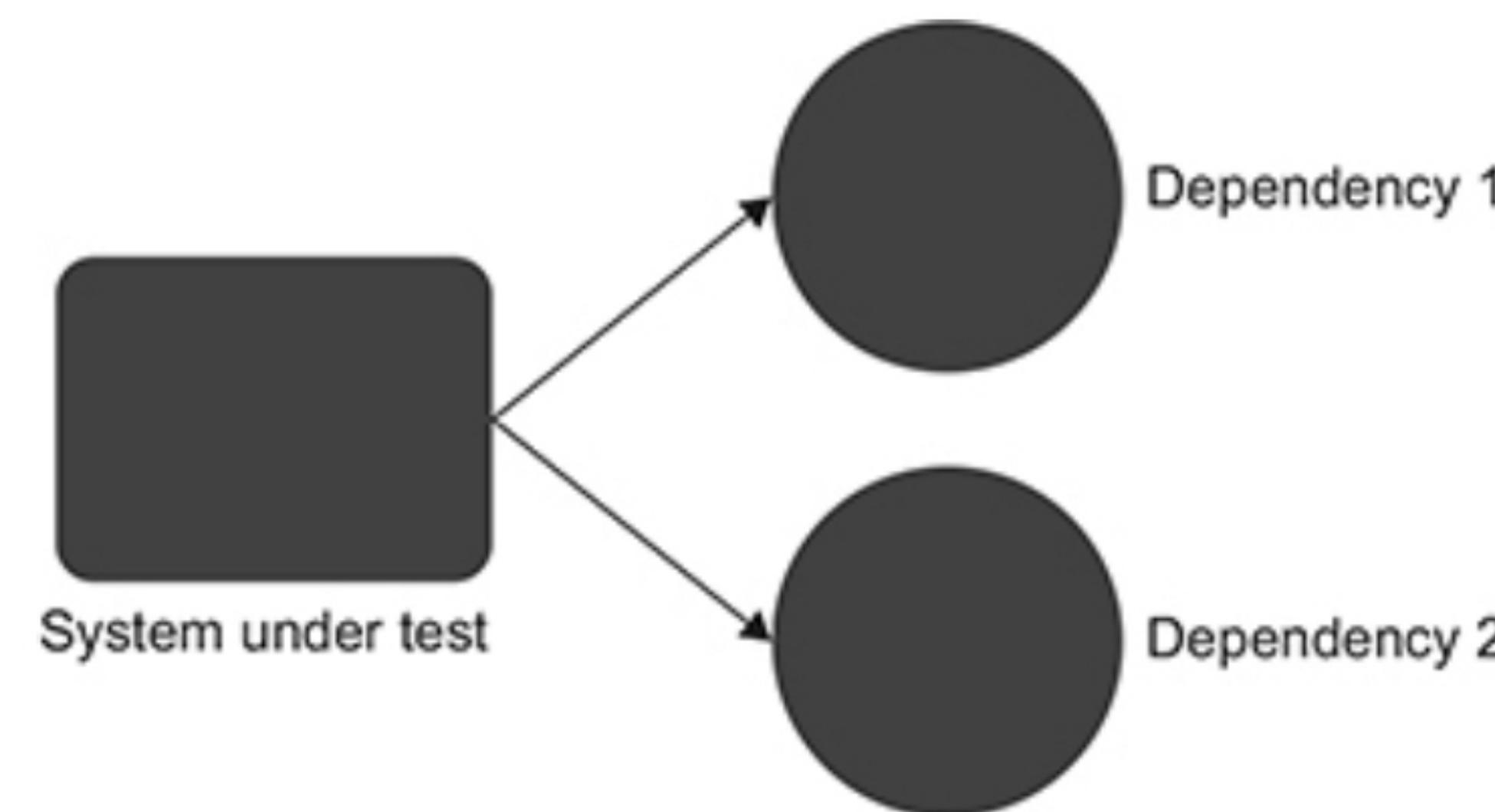
What types of tests do we have?

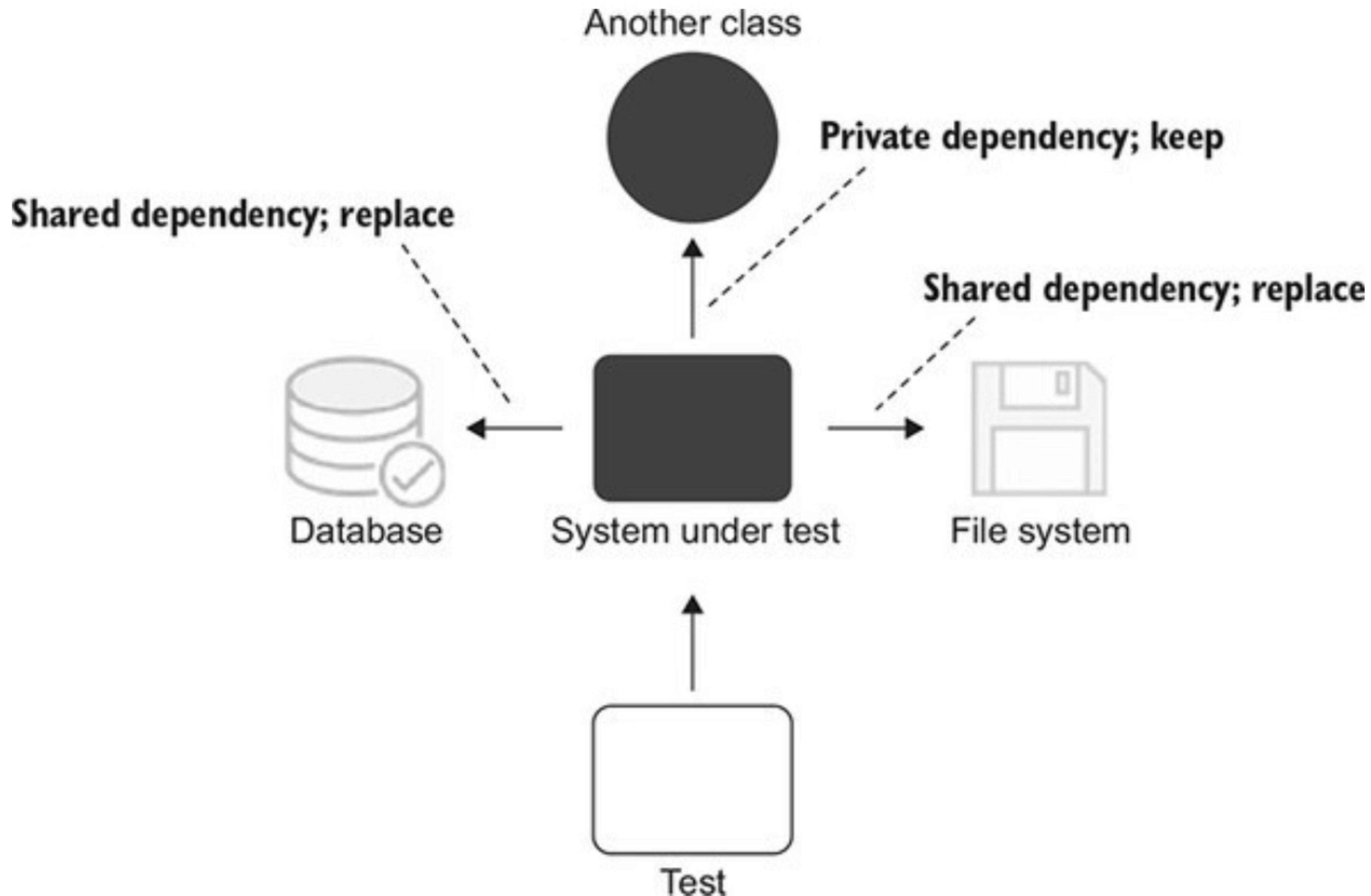
Unit tests, integration test, end-to-end test

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



London style || classic style

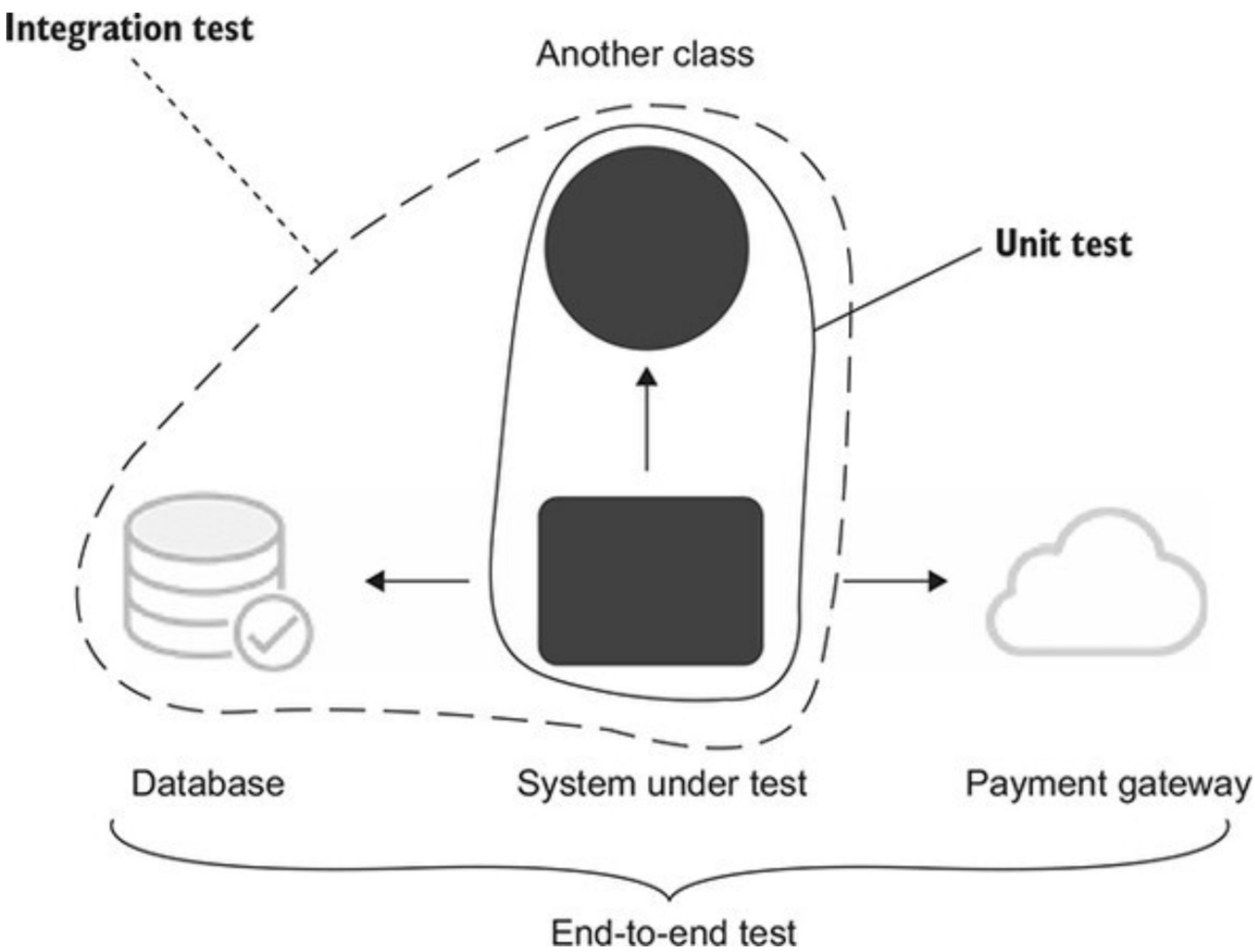




What types of tests do we have?

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

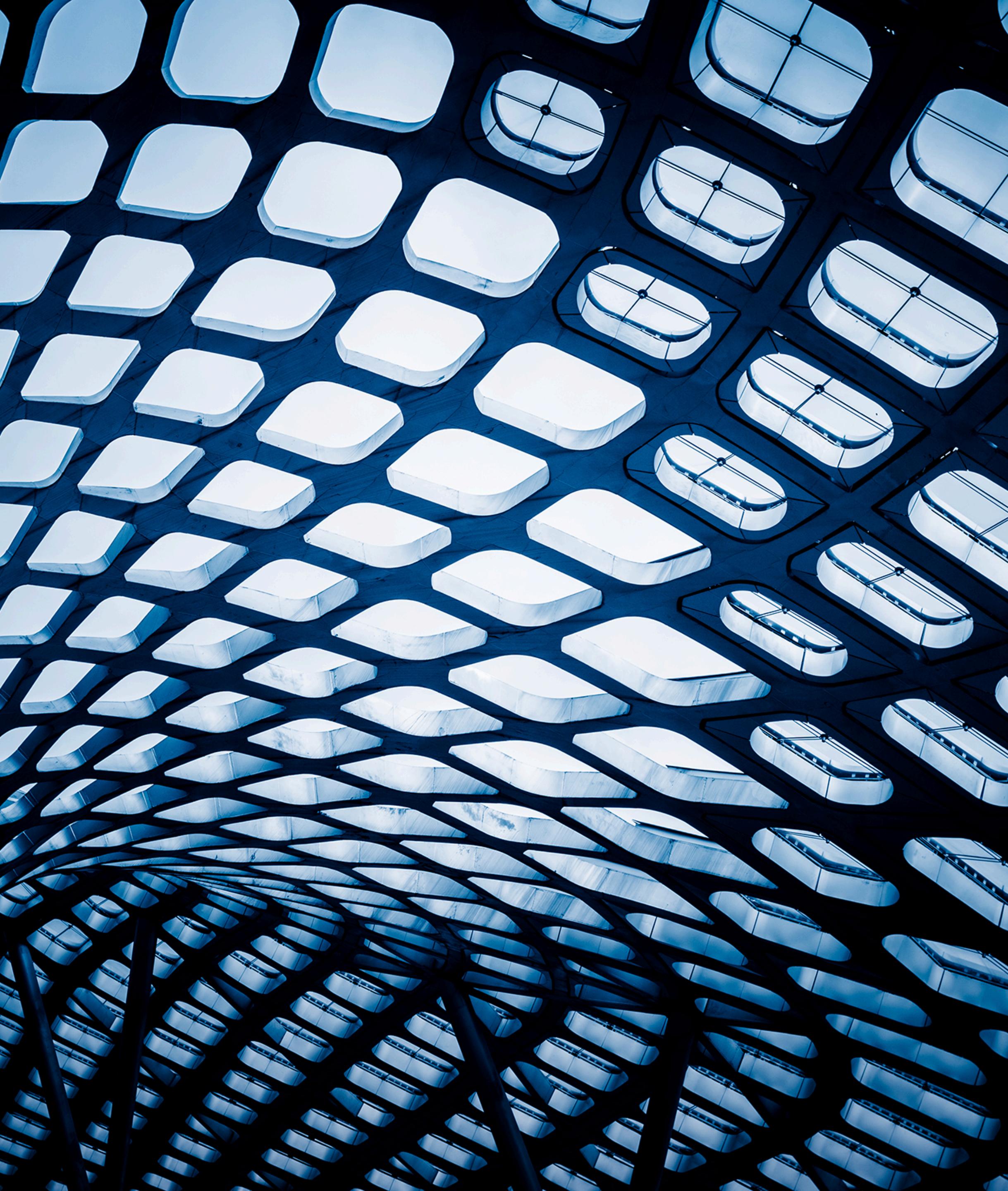
- Verifies a small piece of code (a unit)
- Runs quickly
- Runs in an isolated manner



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

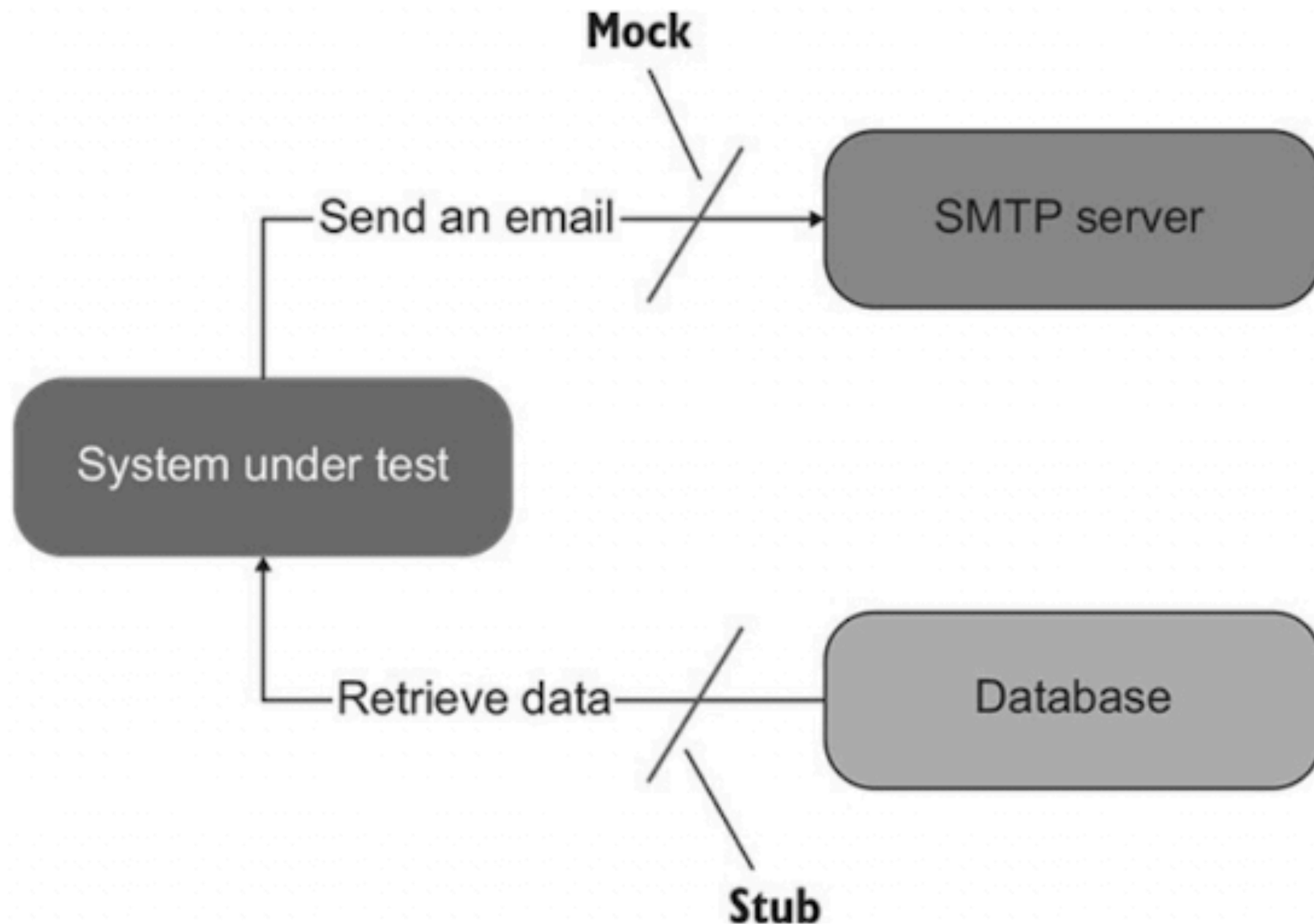


Test double



Mock
(mock, spy)

Stub
(stub, dummy, fake)



The end?

Questions and next meetup
topic(s).

[https://github.com/steinmb/
phpbergen_202402](https://github.com/steinmb/phpbergen_202402)

