CRAFT YOUR TESTS

A STEP TO BE A CRAFTSMAN

Yann Danot @ydanot

Olivier Albiez @OlivierAlbiez

WHAT DO YOU MEAN BY TEST ?

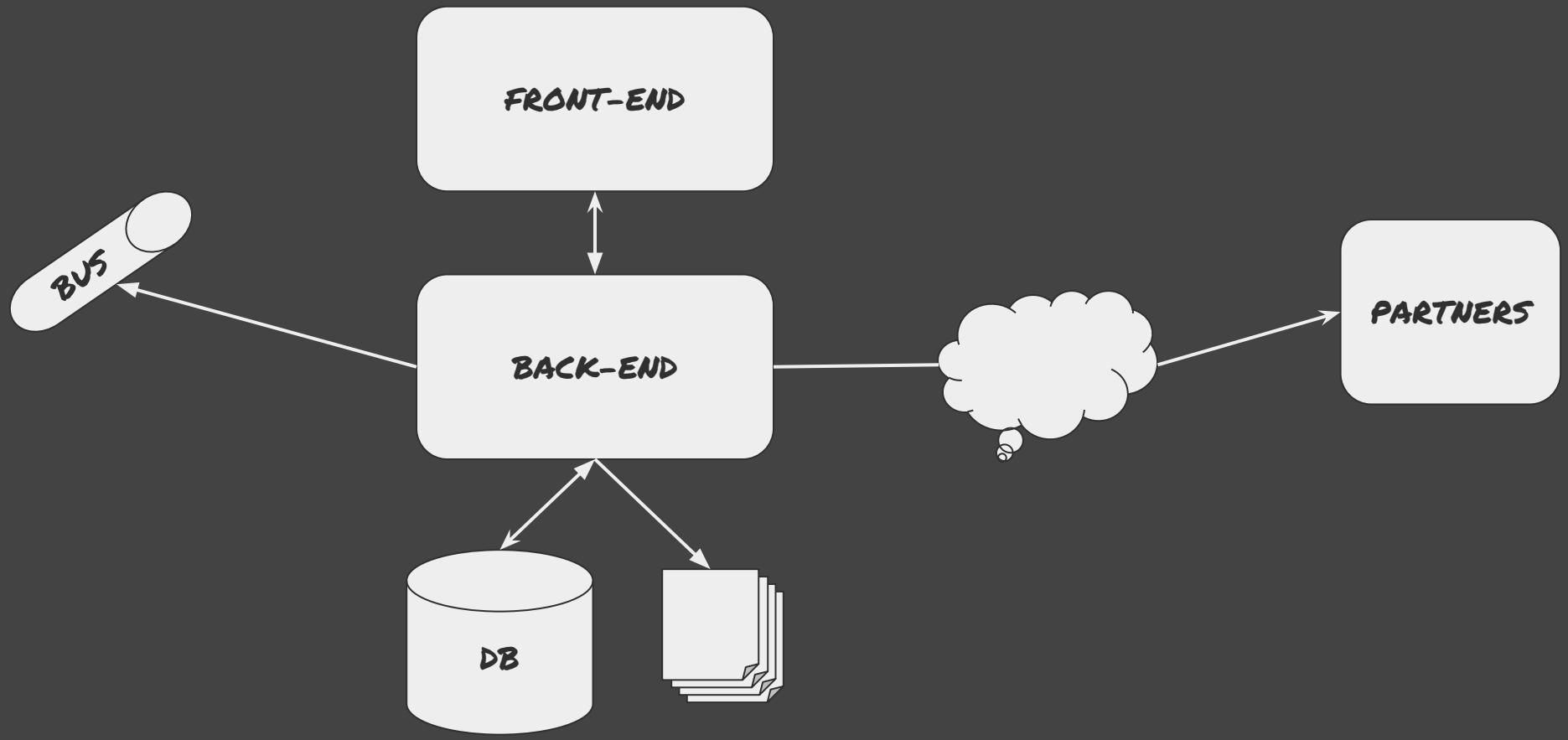
WHAT ABOUT UNIT TESTS ?

WHAT ABOUT UNIT TESTS ?

"

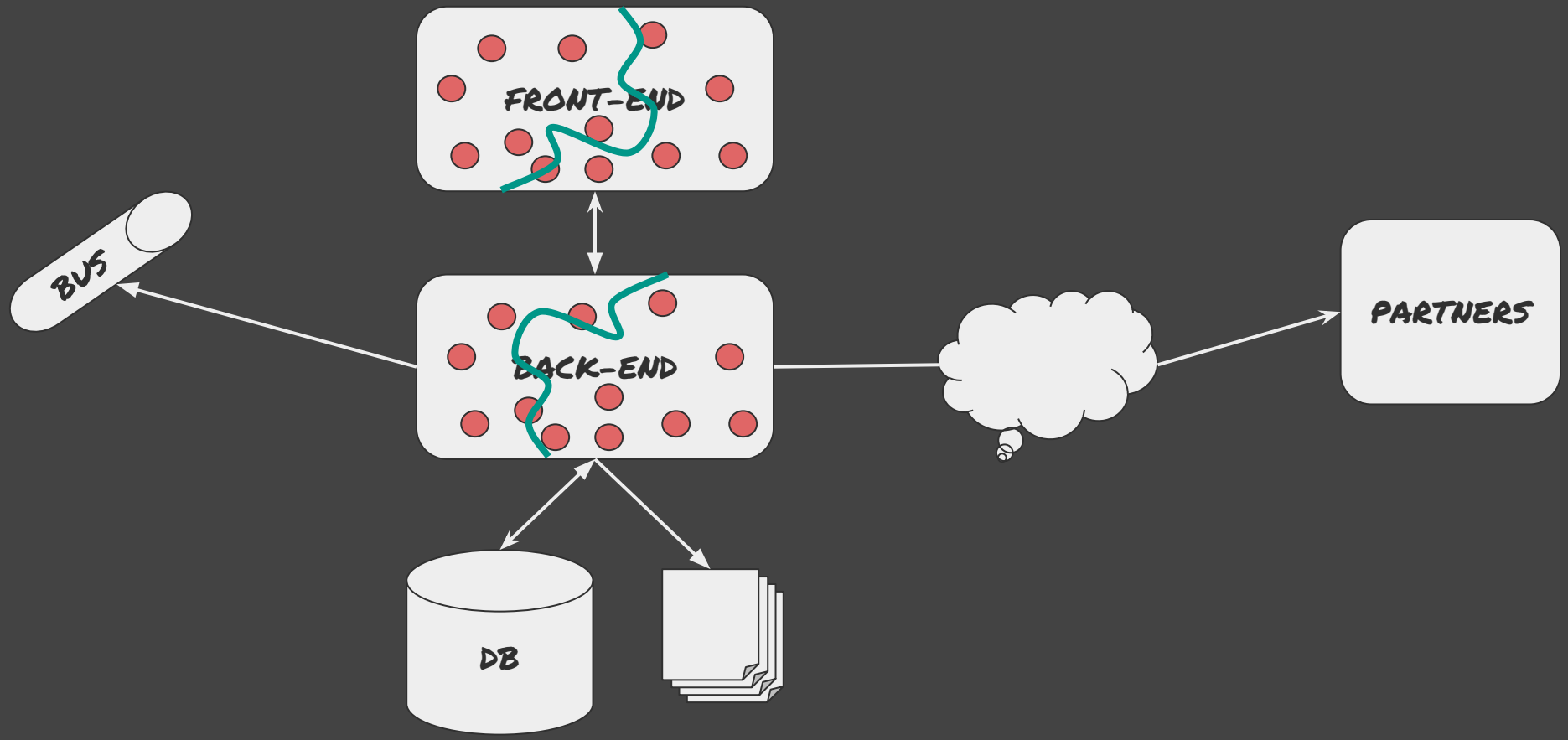
*In computer programming, unit testing is a software testing method by which individual **units** of source **code**, sets of one or more computer program **modules** together with **associated control data**, **usage procedures**, and **operating procedures**, are tested to determine whether they are fit for use.*

WIKIPEDIA


FRONT-END
BUS
BACK-END
PARTNERS
DB

UNIT

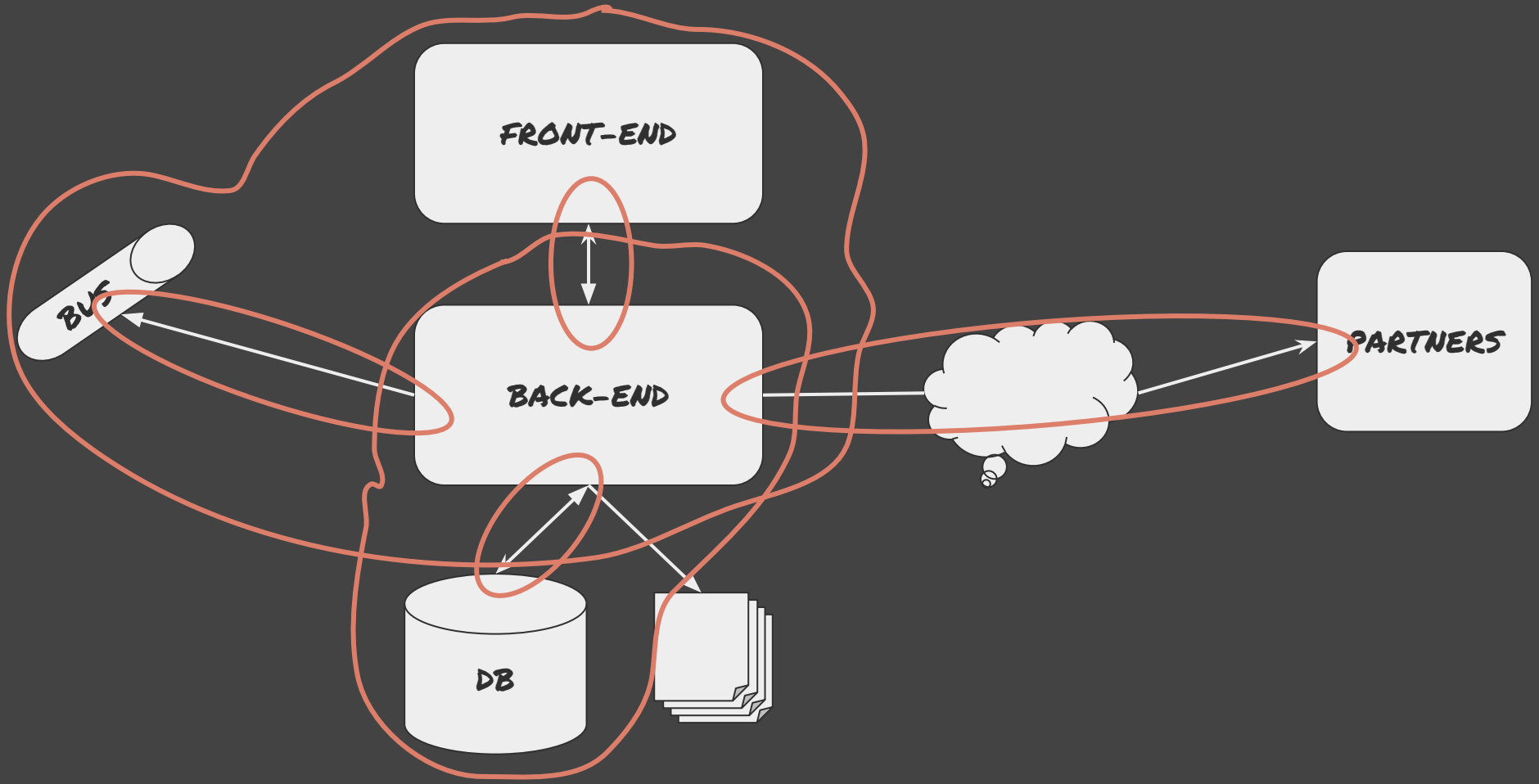
UNIT TESTS



Integration

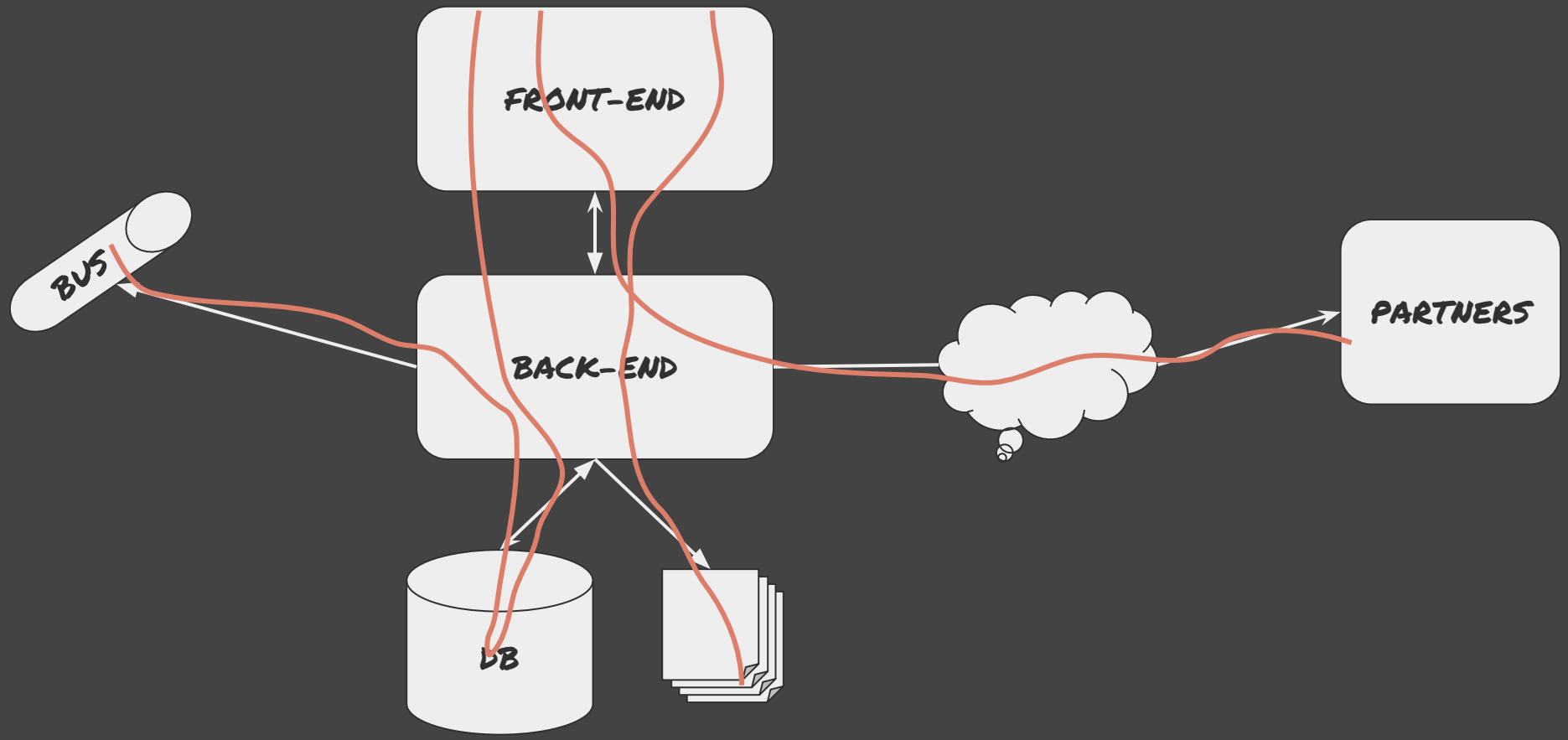
Sub-system

INTEGRATION / SUB-SYSTEM TESTS

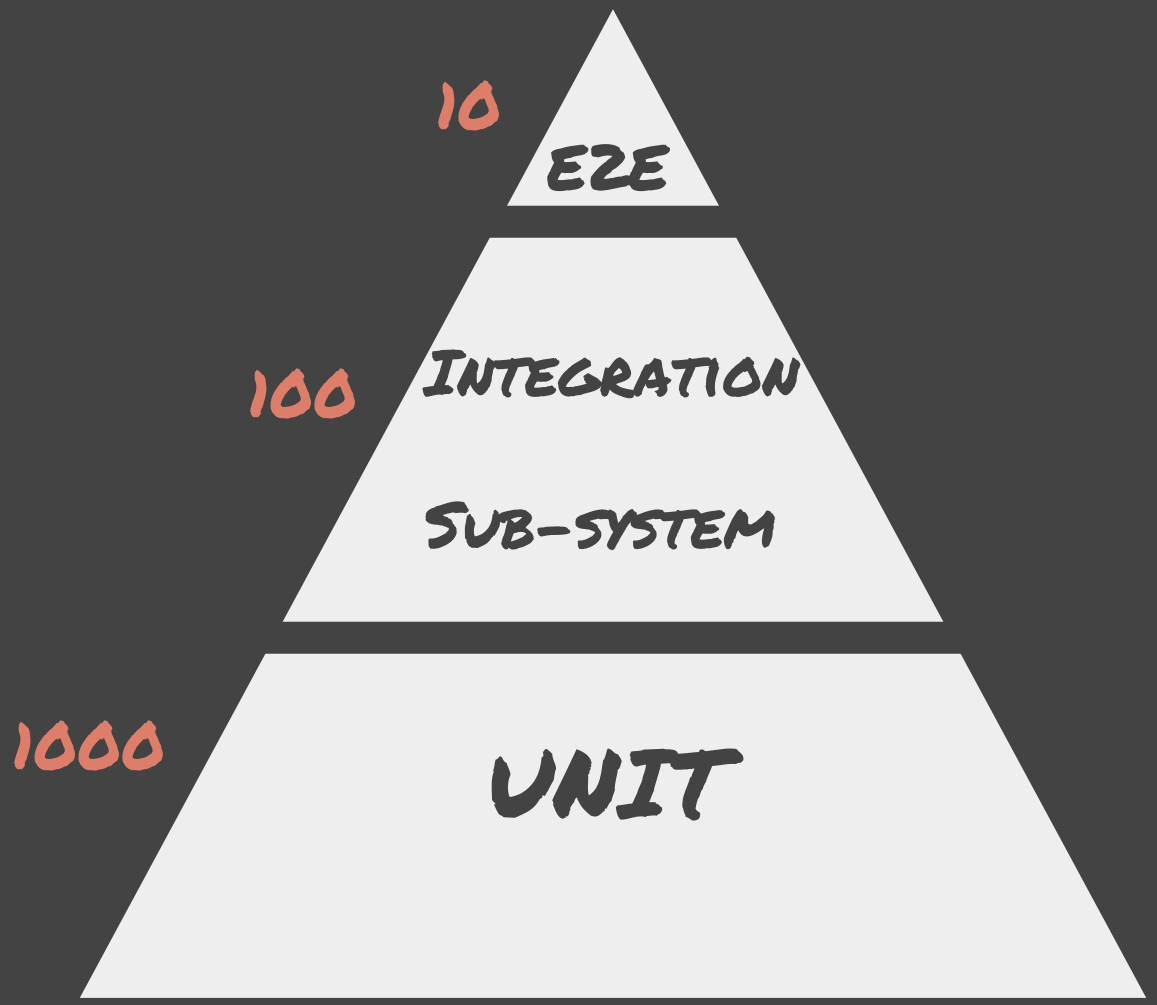


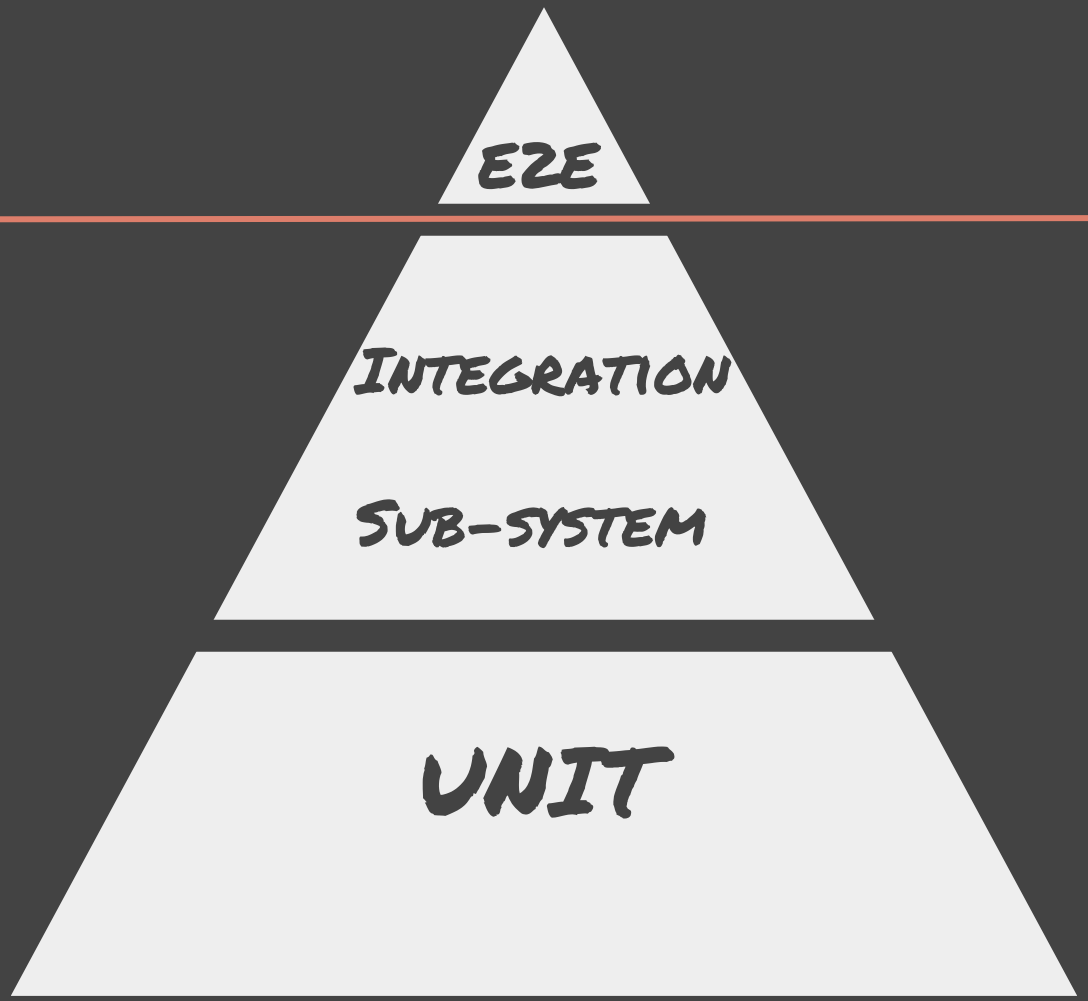
EZE

E2E TESTS



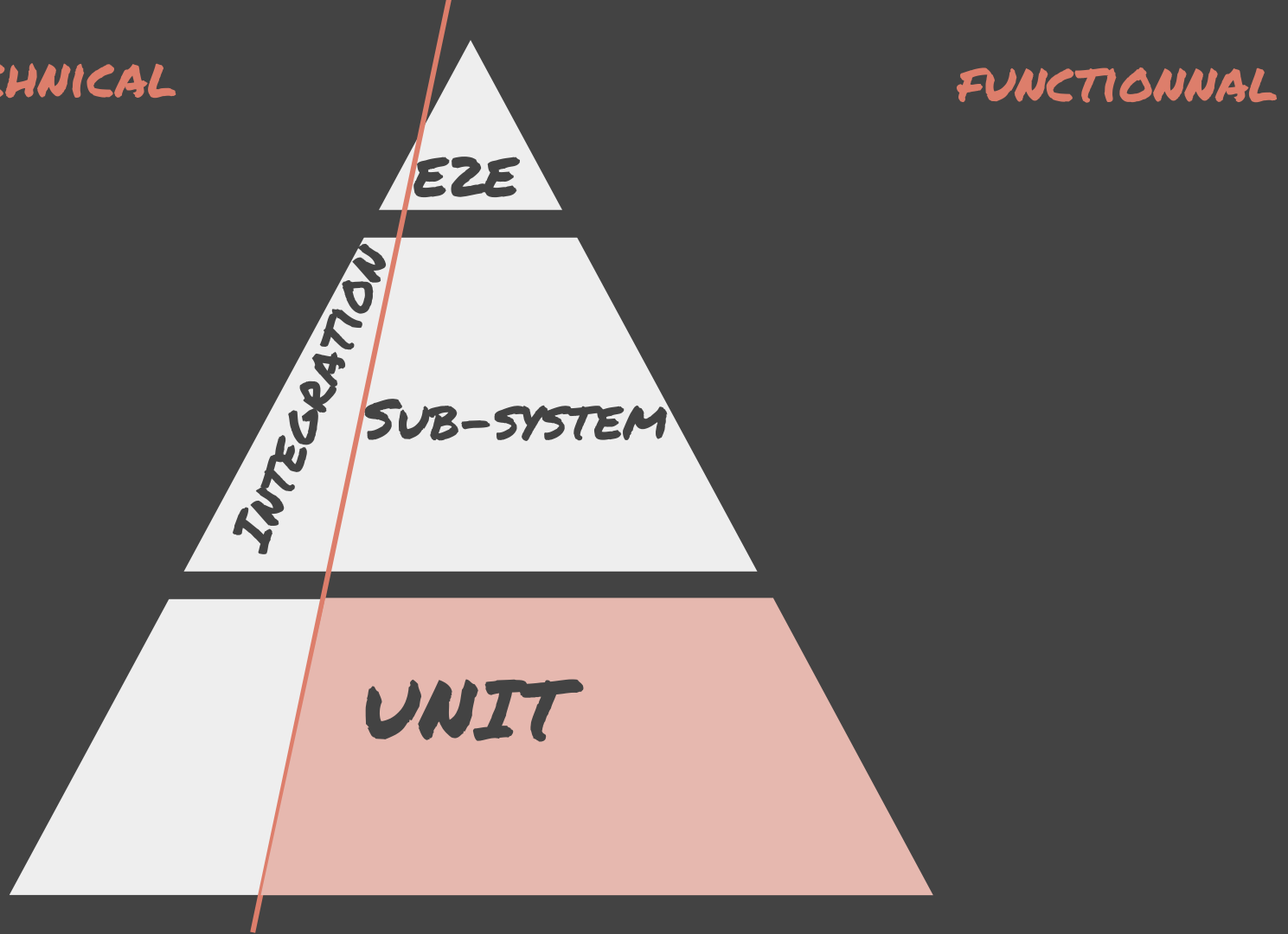
TEST STRATEGY


10
E2E
100
INTEGRATION
SUB-SYSTEM
1000
UNIT



FUNCTIONNAL

TECHNICAL


TECHNICAL
FUNCTIONNAL
E2E
INTEGRATION
SUB-SYSTEM
UNIT

WHY ARE WE UNIT TESTING ?

Our daily work is about learning the business

Captures our understanding of rules

- may be ambiguous / not clear
- may be contradictory
- ~~may~~ will change

Our daily work is about keeping control

Modular and decoupled design

- Keep thing small
- Inverse dependencies

Our daily work is about keeping control

Allows evolutions

- Impact of changes
- Non regression

Our daily work is about keeping control

Quick feedback

- UT is the quickest test suite

- To be able to
- To

ABLE / READABLE

TO BE AGILE !!!

GOOD PRACTICES


THERE ARE GOOD
AND BAD TESTS

WHAT IS A "GOOD" UNIT TEST ?

F.I.R.S.T

PRINCIPLES

F.I.R.S.T

FAST

F.I.R.S.T

INDEPENDENT / ISOLATED

F.I.R.S.T

REPEATABLE

F.I.R.S.T

SELF-VALIDATING

F.I.R.S.T

TIMELY / THOROUGH

F.I.R.S.T ?

```
@Test
public void should_return_2018_template() {
    if (LocalDateTime.now().getYear() == 2018){
        Assertions.assertThat(getVersion()).isEqualTo("2018.1.7");
    }
    else{
        Assertions.assertThat(getVersion()).isEqualTo("1.7");
    }
}

.....................................................................................................
..

public String getVersion() {
    if (LocalDateTime.now().getYear() >= 2018){
        return LocalDateTime.now().getYear() + ".1.7";
    }
    else{
        return "1.7";
    }
}
```

F.I.R.S.T ?

first_tests 〉 src 〉 test 〉 java 〉 io 〉 thalesdigital 〉 MySuperTest

Project

first_tests 3 hidden ~/Workspace/craft_your_tests/t

- src
  - main
    - java
      - io.thalesdigital
        - FizzBuzz
        - SpringConfiguration
    - resources
      - application-context.xml
  - test
    - java
      - io.thalesdigital
        - MySuperTest
    - resources
- .gitignore
- pom.xml
- External Libraries

first_tests | MySuperTest.java | SpringConfiguration.java | FizzBuzz.java | application-context.xml

```
1  package io.thalesdigital;
2
3  import org.assertj.core.api.Assertions;
4  import org.junit.Test;
5  import org.junit.runner.RunWith;
6  import org.springframework.test.context.ContextConfiguration;
7  import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;
8
9  import javax.inject.Inject;
10
11 @ContextConfiguration(classes = {SpringConfiguration.class})
12 @RunWith(SpringJUnit4ClassRunner.class)
13 public class MySuperTest {
14
15     @Inject
16     private FizzBuzz sut;
17
18     @Test
19     public void should_return_FIZZ_when_multiple_of_3(){
20         Assertions.assertThat(sut.fizzBuzz( fizzBuzzable: 3)).isEqualTo("FIZZ");
21     }
22 }
```

Run MySuperTest

1 test passed – 95ms

- MySuperTest (io.thalesdigital) 95ms
  - should_return_FIZZ_when_multiple_of_3 95ms

```
/Library/Java/JavaVirtualMachines/jdk1.8.0_181.jdk/Contents/Home/bin/java ...
août 10, 2018 4:12:31 PM org.springframework.test.context.support.AbstractTestContextBootstrapper getDefaultTestExecutionListenerClassName
INFOS: Loaded default TestExecutionListener class names from location [META-INF/spring.factories]: [org.springframework.test.context.web.S
août 10, 2018 4:12:31 PM org.springframework.test.context.support.AbstractTestContextBootstrapper getTestExecutionListeners
INFOS: Using TestExecutionListeners: [org.springframework.test.context.support.DirtiesContextBeforeModesTestExecutionListener@13a57a3b, or
août 10, 2018 4:12:32 PM org.springframework.context.support.AbstractApplicationContext prepareRefresh
INFOS: Refreshing org.springframework.context.support.GenericApplicationContext@77caeb3e: startup date [Fri Aug 10 16:12:32 CEST 2018]; ro
août 10, 2018 4:12:32 PM org.springframework.beans.factory.annotation.AutowiredAnnotationBeanPostProcessor <init>
INFOS: JSR-330 'javax.inject.Inject' annotation found and supported for autowiring

Process finished with exit code 0
```

0: Messages | 4: Run | 6: TODO | 9: Version Control | Terminal | Spring | Event Log

Tests Passed: 1 passed (2 minutes ago)

12:1 LF UTF-8 Git: master

F.I.R.S.T ?

first_tests 〉 src 〉 test 〉 java 〉 io 〉 thalesdigital 〉 MySuperTest

Project

first_tests 3 hidden ~/Workspace/craft_your_tests/t

- src
  - main
    - java
      - io.thalesdigital
        - FizzBuzz
        - SpringConfiguration
    - resources
      - application-context.xml
  - test
    - java
      - io.thalesdigital
        - MySuperTest
    - resources
- .gitignore
- pom.xml
- External Libraries

first_tests | MySuperTest.java | SpringConfiguration.java | FizzBuzz.java | application-context.xml

```java
package io.thalesdigital;

import org.assertj.core.api.Assertions;
import org.junit.Test;
import org.junit.runner.RunWith;
import org.springframework.test.context.ContextConfiguration;
import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;

import javax.inject.Inject;

@ContextConfiguration(classes = {SpringConfiguration.class})
@RunWith(SpringJUnit4ClassRunner.class)
public class MySuperTest {

    @Inject
    private FizzBuzz sut;

    @Test
    public void should_return_FIZZ_when_multiple_of_3(){
        Assertions.assertThat(sut.fizzBuzz( fizzBuzzable: 3)).isEqualTo("FIZZ");
    }
}
```

Run MySuperTest

1 test passed – 95ms

MySuperTest (io.thalesdigital) 95m

should_return_FIZZ_when_multiple_of_3 95ms

FAST ?

```
/Library/Java/JavaVirtualMachines/jdk1.8.0_181.jdk/Contents/Home/bin/java ...
août 10, 2018 4:12:31 PM org.springframework.test.context.support.AbstractTestContextBootstrapper getDefaultTestExecutionListenerClassName
INFOS: Loaded default TestExecutionListener class names from location [META-INF/spring.factories]: [org.springframework.test.context.web.S
août 10, 2018 4:12:31 PM org.springframework.test.context.support.AbstractTestContextBootstrapper getTestExecutionListeners
INFOS: Using TestExecutionListeners: [org.springframework.test.context.support.DirtiesContextBeforeModesTestExecutionListener@13a57a3b, or
août 10, 2018 4:12:32 PM org.springframework.context.support.AbstractApplicationContext prepareRefresh
INFOS: Refreshing org.springframework.context.support.GenericApplicationContext@77caeb3e: startup date [Fri Aug 10 16:12:32 CEST 2018]; ro
août 10, 2018 4:12:32 PM org.springframework.beans.factory.annotation.AutowiredAnnotationBeanPostProcessor <init>
INFOS: JSR-330 'javax.inject.Inject' annotation found and supported for autowiring

Process finished with exit code 0
```

0: Messages | 4: Run | 6: TODO | 9: Version Control | Terminal | Spring | Event Log

Tests Passed: 1 passed (2 minutes ago)

12:1 LF UTF-8 Git: master


WE CAN READ A
GOOD TEST

F.I.R.S.T ?

first_tests

Project

first_tests 3 hidden ~/Workspace/craft_your_tests/t

- src
  - main
    - java
      - io.thalesdigital
        - Database
        - FizzBuzzConverter
        - FizzBuzzService
        - SpringConfiguration
    - resources
  - test
    - java
      - io.thalesdigital
        - MySuperTestWithMock
    - resources
- .gitignore
- pom.xml
- External Libraries

first_tests | SpringConfiguration.java | MySuperTestWithMock.java | FizzBuzzConverter.java | Database.java | FizzBuzzService.java

```java
package io.thalesdigital;

import ...

@RunWith(MockitoJUnitRunner.class)
public class MySuperTestWithMock {

    @InjectMocks
    public FizzBuzzService sut;

    @Mock
    public Database database;

    @Mock
    public FizzBuzzConverter fizzBuzzConverter;

    @Test
    public void should_return_FIZZ_when_multiple_of_3(){

        Mockito.when(database.findWords()).thenReturn(Maps.newHashMap(3,"FIZZ"));
        Mockito.when(fizzBuzzConverter.convert(Matchers.anyInt(),Matchers.anyMap())).thenCallRealMethod();

        Assertions.assertThat(sut.fizzBuzz( fizzBuzzable: 3)).isEqualTo("FIZZ");

        Mockito.verify(fizzBuzzConverter, VerificationModeFactory.atLeastOnce()).convert(Matchers.anyInt(),Matchers.anyMap());
        Mockito.verify(database, VerificationModeFactory.atLeastOnce()).findWords();
        Mockito.verify(database, VerificationModeFactory.noMoreInteractions()).findWords();
    }

}
```

Run MySuperTestWithMock

1 test passed – 70ms

- MySuperTestWithMock (io.thalesdigital) 70ms
  - should_return_FIZZ_when_multiple_of_3 70ms

```
/Library/Java/JavaVirtualMachines/jdk1.8.0_181.jdk/Contents/Home/bin/java ...

Process finished with exit code 0
```

0: Messages | 4: Run | 6: TODO | 9: Version Control | Terminal | Spring | Event Log

Tests Passed: 1 passed (moments ago)

39:2 LF UTF-8 Git: master

Readability - Naming

There are 2 differents way of naming :

- "Storytelling"
  - Simply enounce the rule
  - Put severals test cases (examples)
- Test cases
  - test name Answer to this 3 questions

    What is being tested
    Under what circumstances
    What is the expected result

An example of storytelling

```
@Test
public void should_return_FIZZ_when_multiple_of_3 (){...}


@Test
public void should_return_BUZZ_when_multiple_of_5 (){...}


@Test
public void should_return_the_input_otherwise (){...}
```

An example of test cases

```
@Test
public void should_return_FIZZ_when_3 (){...}

@Test
public void should_return_FIZZ_when_6 (){...}

@Test
public void should_return_BUZZ_when_5 (){...}

@Test
public void should_return_BUZZ_when_10 (){...}

@Test
public void should_return_4_when_4 (){...}
```

Readability - Behavior

Behavior = Context + Action + Outcome

Readability - behavior

3 phases

Given a context

setup/arrange

When an event happens

exercise/act

Then an outcome should be observed

verify/assert

Readability - behavior

behavior

Given a context

When an event happens

Then an outcome should be observed

Readability - behavior

Given a context

When an event happens

Then an outcome should be observed

Business Language Only !!!

Readability - encapsulation

```
@Test
public void when_I_create_a_game_there_are_mines_on_grid () {
   given_a_minesweeper()
           .withNbMines( 10);
   when_I_start_the_game();
   then_mines_number_should_be( 10);
}


@Test
public void
when_I_ask_for_adjacent_position_of_top_left_corner_it_returns_only_right_down_positions  (){
   given_a_board()
   when_I_ask_for_adjacent_position_of(position( 0, 0));
   then_adjacent_position_should_be(
                           position( 0, 1),
           position(1, 0), position(1, 1));
}
```

Readability - KISS

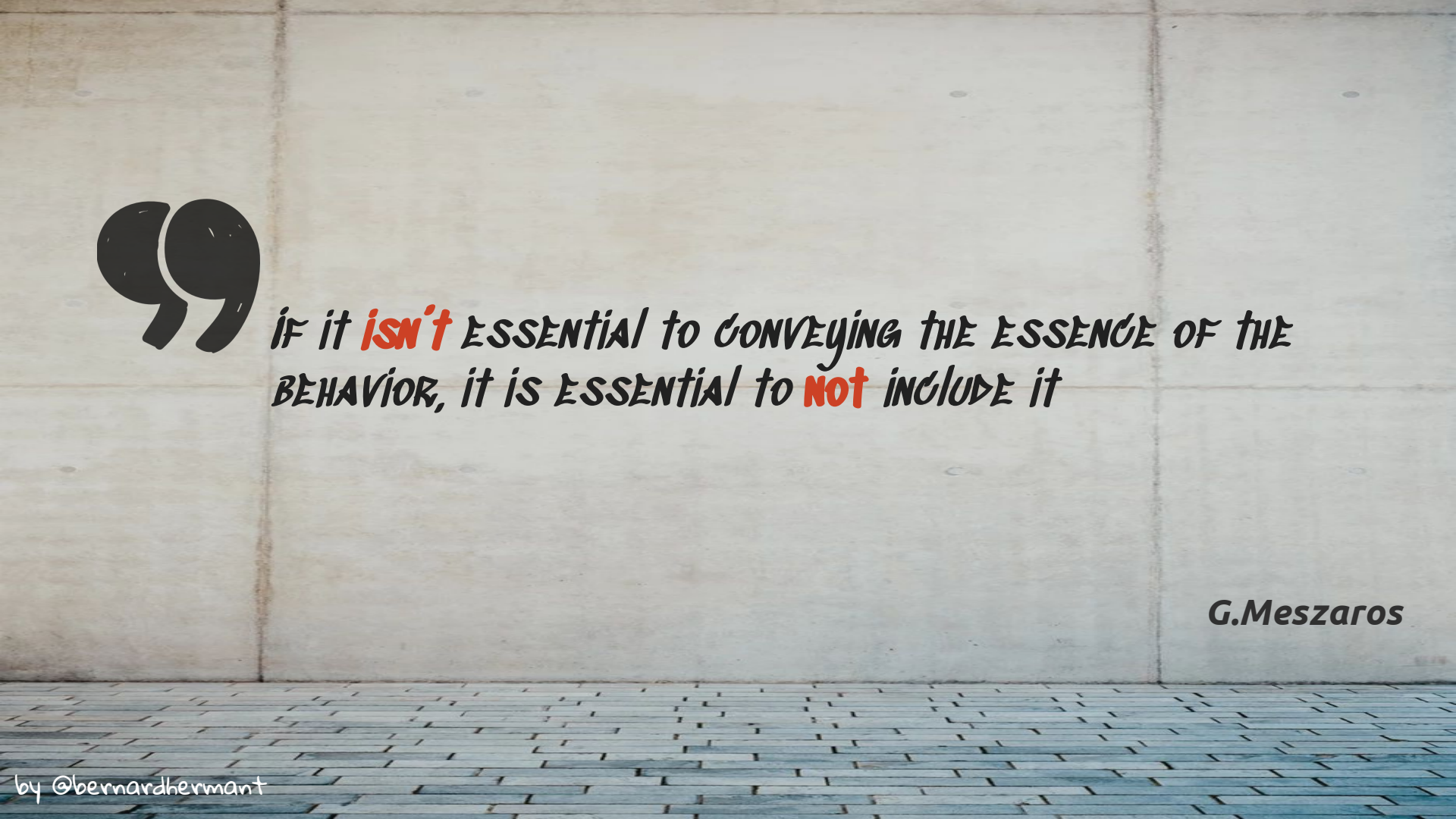
```
@Test
public void a_player_invite_the_opponent_to_pick_a_card_when_he_does_not_have_the_asked_card(){
   given_a_seven_famillies_game()
           .with_familly("Martin").withGrandfather("Robert").andGrandmother("Andrée").withFather("Didier").withMother("Isabelle").withSon("Kevin").withDaughter("Kevina")
           .with_familly("Garcia").withGrandfather("Juan").andGrandmother("Lucia").withFather("Pedro").withMother("Maria").withSon("Pablo").withDaughter("Lisa")
           .with_familly("Murphy").withGrandfather("Eddie").andGrandmother("Elizabeth").withFather("John").withMother("Abbie").withSon("Brandon").withDaughter("Brenda")
           .with_familly("Miller").withGrandfather("Thomas").andGrandmother("Eva").withFather("Andrea").withMother("Alexandra").withSon("Claus").withDaughter("Anna")
           .with_familly("Smith").withGrandfather("Adam").andGrandmother("Ava").withFather("Donald").withMother("Michell").withSon("Steven").withDaughter("Scarlett")
           .with_familly("Rossi").withGrandfather("Antonio").andGrandmother("Alessandra").withFather("Luigi").withMother("Isabella").withSon("Mario").withDaughter("Maria")
           .with_familly("Silva").withGrandfather("Manuel").andGrandmother("Louisa").withFather("Joao").withMother("Andreia").withSon("Luis").withDaughter("Lucia");

   given_player_1_has().card("Martin Robert").card("Martin Andrée").card("Martin Didier").card("Garcia Juan").card("Garcia Lucia").card("Garcia Pedro")
   given_player_2_has().card("Murphy Eddie").card("Murphy Elizabeth").card("Murphy John").card("Miller Thomas").card("Miller Eva").card("Miller Andrea")
   deck_is_composed_by()
           .card("Martin Isabelle").card("Martin Kevin").card("Martin Kevina")
           .card("Garcia Maria").card("Garcia Pablo").card("Garcia Lisa")
           .card("Murphy Abbie").card("Murphy Brandon").card("Murphy Brenda")
           .card("Miller Alexandra").card("Miller Claud").card("Miller Anna")
           .card("Smith Adam").card("Smith Ava").card("Smith Donald").card("Smith Michell").card("Smith Steven").card("Smith Scarlett")
           .card("Rossi Antonio").card("Rossi Alessandra").card("Rossi Luigi").card("Rossi Isabella").card("Rossi Mario").card("Rossi Maria")
           .card("Silva Manuel").card("Silva Louisa").card("Silva Joao").card("Silva Andreia").card("Silva Luis").card("Silva Lucia");

   when_player_1_ask_for("Rossi Antonio");

   then_he_should_be_invited_to_pick_a_card_in_deck()
}
```

Too much details !!!


IF IT ISN'T ESSENTIAL TO CONVEYING THE ESSENCE OF THE BEHAVIOR, IT IS ESSENTIAL TO NOT INCLUDE IT
G.Meszaros
by @bernardhermant

Readability - KISS

```
@Test
public void a_player_invite_the_opponent_to_pick_a_card_when_he_does_not_have_the_asked_card(){
   given_a_seven_famillies_game()
           .with_familly("Martin").withGrandfather("Robert").andGrandmother("Andrée").withFather("Didier").withMother("Isabelle").withSon("Kevin").withDaughter("Kevina")
           .with_familly("Garcia").withGrandfather("Juan").andGrandmother("Lucia").withFather("Pedro").withMother("Maria").withSon("Pablo").withDaughter("Lisa")
           .with_familly("Murphy").withGrandfather("Eddie").andGrandmother("Elizabeth").withFather("John").withMother("Abbie").withSon("Brandon").withDaughter("Brenda")
           .with_familly("Miller").withGrandfather("Thomas").andGrandmother("Eva").withFather("Andrea").withMother("Alexandra").withSon("Claus").withDaughter("Anna")
           .with_familly("Smith").withGrandfather("Adam").andGrandmother("Ava").withFather("Donald").withMother("Michell").withSon("Steven").withDaughter("Scarlett")
           .with_familly("Rossi").withGrandfather("Antonio").andGrandmother("Alessandra").withFather("Luigi").withMother("Isabella").withSon("Mario").withDaughter("Maria")
           .with_familly("Silva").withGrandfather("Manuel").andGrandmother("Louisa").withFather("Joao").withMother("Andreia").withSon("Luis").withDaughter("Lucia");

   given_player_1_has().card("Martin Robert").card("Martin Andrée").card("Martin Didier").card("Garcia Juan").card("Garcia Lucia").card("Garcia Pedro")
   given_player_2_has().card("Murphy Eddie").card("Murphy Elizabeth").card("Murphy John").card("Miller Thomas").card("Miller Eva").card("Miller Andrea")

   deck_is_composed_by()
           .card("Martin Isabelle").card("Martin Kevin").card("Martin Kevina")
           .card("Garcia Maria").card("Garcia Pablo").card("Garcia Lisa")
           .card("Murphy Abbie").card("Murphy Brandon").card("Murphy Brenda")
           .card("Miller Alexandra").card("Miller Claud").card("Miller Anna")
           .card("Smith Adam").card("Smith Ava").card("Smith Donald").card("Smith Michell").card("Smith Steven").card("Smith Scarlett")
           .card("Rossi Antonio").card("Rossi Alessandra").card("Rossi Luigi").card("Rossi Isabella").card("Rossi Mario").card("Rossi Maria")
           .card("Silva Manuel").card("Silva Louisa").card("Silva Joao").card("Silva Andreia").card("Silva Luis").card("Silva Lucia");

   when_player_1_ask_for("Rossi Antonio");

   then_he_should_be_invited_to_pick_a_card_in_deck()
}
```

SIMPLIFY !

Readability - KISS

```
@Test
public void a_player_invite_the_opponent_to_pick_a_card_when_he_does_not_have_the_asked_card(){
  given_a_seven_families_game()
          .with_familly("Martin").withGrandfather("Robert").andGrandmother("Andrée").withFather("Didier").withMother("Isabelle").withSon("Kevin").withDaughter("Kevina")
          .with_familly("Garcia").withGrandfather("Juan").andGrandmother("Lucia").withFather("Pedro").withMother("Maria").withSon("Pablo").withDaughter("Lisa")
          .with_familly("Murphy").withGrandfather("Eddie").andGrandmother("Elizabeth").withFather("John").withMother("Abbie").withSon("Brandon").withDaughter("Brenda")
          .with_familly("Miller").withGrandfather("Thomas").andGrandmother("Eva").withFather("Andrea").withMother("Alexandra").withSon("Claus").withDaughter("Anna")
          .with_familly("Smith").withGrandfather("Adam").andGrandmother("Ava").withFather("Donald").withMother("Michell").withSon("Steven").withDaughter("Scarlett")
          .with_familly("Rossi").withGrandfather("Antonio").andGrandmother("Alessandra").withFather("Luigi").withMother("Isabella").withSon("Mario").withDaughter("Maria")
          .with_familly("Silva").withGrandfather("Manuel").andGrandmother("Louisa").withFather("Joao").withMother("Andreia").withSon("Luis").withDaughter("Lucia");

  given_player_1_has().card("Martin Robert").card("Martin Andrée").card("Martin Didier").card("Garcia Juan").card("Garcia Lucia").card("Garcia Pedro")
  given_player_2_has().card("Murphy Eddie").card("Murphy Elizabeth").card("Murphy John").card("Miller Thomas").card("Miller Eva").card("Miller Andrea");

  when_player_1_ask_for("Rossi Antonio");

  then_he_should_be_invited_to_pick_a_card_in_deck()
}
```

Readability - Persona

```
@Test
public void a_player_invite_the_opponent_to_pick_a_card_when_he_does_not_have_the_asked_card(){
   given_a_standard_seven_familles_game();
   given_player_1_has().card("Martin Robert").card("Martin Andrée").card("Martin Didier").card("Garcia Juan").card("Garcia
Lucia").card("Garcia Pedro")
   given_player_2_has().card("Murphy Eddie").card("Murphy Elizabeth").card("Murphy John").card("Miller Thomas").card("Miller
Eva").card("Miller Andrea");

   when_player_1_ask_for("Rossi Antonio");

   then_he_should_be_invited_to_pick_a_card_in_deck()
}

@Test
public void a_player_should_give_his_opponent_a_card_when_this_card_is_asked(){
   given_a_standard_seven_familles_game();
      given_player_1_has().card("Martin Robert").card("Martin Andrée").card("Martin Didier").card("Garcia Juan").card("Garcia
Lucia").card("Garcia Pedro")
   given_player_2_has().card("Murphy Eddie").card("Murphy Elizabeth").card("Murphy John").card("Miller Thomas").card("Miller
Eva").card("Miller Andrea");

   when_player_1_ask_for("Murphy Eddie");

   then_he_should_received("Murphy Eddie")
}
```

Readability - Persona

```
@Test
public void given_a_standard_seven_famillies_game(){
    given_a_seven_famillies_game()
          .with_familly("Martin").withGrandfather("Robert").andGrandmother("Andrée").withFather("Didier").withMother("Isabelle").withSon("Kevin").withDaughter("Kevina")
          .with_familly("Garcia").withGrandfather("Juan").andGrandmother("Lucia").withFather("Pedro").withMother("Maria").withSon("Pablo").withDaughter("Lisa")
          .with_familly("Murphy").withGrandfather("Eddie").andGrandmother("Elizabeth").withFather("John").withMother("Abbie").withSon("Brandon").withDaughter("Brenda")
          .with_familly("Miller").withGrandfather("Thomas").andGrandmother("Eva").withFather("Andrea").withMother("Alexandra").withSon("Claus").withDaughter("Anna")
          .with_familly("Smith").withGrandfather("Adam").andGrandmother("Ava").withFather("Donald").withMother("Michell").withSon("Steven").withDaughter("Scarlett")
          .with_familly("Rossi").withGrandfather("Antonio").andGrandmother("Alessandra").withFather("Luigi").withMother("Isabella").withSon("Mario").withDaughter("Maria")
          .with_familly("Silva").withGrandfather("Manuel").andGrandmother("Louisa").withFather("Joao").withMother("Andreia").withSon("Luis").withDaughter("Lucia");
}
```

"Perfection is achieved not when there is nothing more to add, but when there is nothing left to take away"

A. De Saint Exupery

by @bernardhermant


WE CAN READ A BAD TEST, BUT IT'S A BAD TEST


WE CAN VERIFY A
GOOD TEST

Verifiability - Assertions

- Every test should have (one or more) assertion
  - What do you want to test ?
  - Depends on the way you're writing (story telling or test case)
  - Can be custom to facilitate readability

Verifiability – Custom Assertions

Remove duplicated assertion logic by creating your own Assertion Methods to:

- Improve readability
  - Intent-revealing methods that verify expected outcome
- Simplify troubleshooting
  - Make xUnit failure reports easier to understand
- Define test-specific equality
  - Ignore "don't care" fields when comparing objects
  - "Foreign Method" specific to testing
- Can be defined using Extract Method refactoring.

Verifiability - Challengeable

- Automation can be challenged by modifying test case
  - What if I change this input ?
  - What if I change this output ?
  - What if I change this call ?
  - Test should fail

VERIFIABILITY - CHALLENGEABLE

```
@Test
public void withdrawal_should_be_allowed_to_anyone_who_has_money(){
   given_a_standard_customer()
   .with_account_balance(Money.of(200));
                                  100
   when_he_withdraws(Money.of(20));
            deposits
   then_his_account_balance_should_be(Money.of(180));
}                                                 150
```


WE CAN VERIFY A BAD TEST, BUT IT'S A BAD TEST


WE CAN MAINTAIN A
GOOD TEST

Maintainability – Why

- Tests need to be maintained along with rest of the software.

- Testware must be much easier to maintain than production software, otherwise :
  - It will slow you down
  - It will get left behind
  - Value drops to zero
  - You'll go back to manual testing

Maintainability - Black box

- Black Box vs White Box
  - Black box : know what it should do
  - White box : know how it is built inside
- Even Unit Tests should be black box.

Maintainability – Black box ?

```
    @Test
    public void should_return_the_maximum_transaction() {
        Transaction t1 = transaction("client", "A", 300, LocalDateTime.of(2017, APRIL, 17, 8,
0, 0));
        Transaction t2 = transaction("client", "A", 600, LocalDateTime.of(2017, APRIL, 17, 8,
30, 0));
        stat.updateWith(t1);
        stat.updateWith(t2);
        assertEquals(Optional.of(maximum(600, LocalDateTime.of(2017, APRIL, 17, 8, 30, 0))),
stat.maximum());
    }
```

Maintainability - Black box

- Be independent of implementation
  - + improve readability
  - + Ease storytelling way of writing
  - + Improve test robustness about changing implementation
  - + Abstract rules
  - - Need more test code

Maintainability - Black box

```
@Test
public void should_return_the_maximum_transaction() {
    assertEquals("600 at 08:30:00",
            maximum(transaction(300, "08:00:00"), transaction(600, "08:30:00")));
}
```

Maintainability - Black box

```
@Test
public void should_return_the_maximum_transaction() {
    given().a_transaction().of(300).at("08:00:00");
    and().a_transaction().of(600).at("08:30:00");

   when_maximum_is_computed()
   then().a_transaction().of(600).at("08:30:00").shouldBe_returned()
}
```


WE CAN MAINTAIN A BAD TEST BUT IT'S A BAD TEST

Summary

- A good unit test is :
  - Readable
    - Simple Behavior
    - Storytelling with persona
  - Maintainable
    - Independent of implementation
    - Black box
  - Verifiable
    - Custom assertions
    - challengeable

When a test is failing

- The new rule is changing the tested behavior

  => You have to adapt the test

- The broken behavior is still relevant

  => you need to fix production / test code

- Any doubt ?

  => Share with "domain experts"

When a test is failing

Regression or new rule ?

- Did you change the rule the failing test verify ?

```
@Test
public void testUser(){
   // mock setup ~ 15 lines
   // assertions ~ 10 lines
   // mock verify ~ 10 lines
}
```

- Hard to not change test code !

When a test is failing

Change production or test code ?

- Did you change the rule the failing test verify ?

```
@Test
public void should_not_create_minor_user(){
    given_a_minor_person()
    when_he_creates_his_account()
    then_his_demand_should_be_rejected()
}
```

- You may adapt production code, if it's relevant

かたGAME OF LIFE

Problem Description

- This Kata is about calculating the next generation of Conway's game of life, given any starting position.

- You start with a two dimensional grid of cells, where each cell is either alive or dead.

- In this version of the problem, the grid is finite, and no life can exist off the edges.

- When calculating the next generation of the grid, follow these rules:

    1. Any live cell with fewer than two live neighbours dies, as if caused by underpopulation.
    2. Any live cell with more than three live neighbours dies, as if by overcrowding.
    3. Any live cell with two or three live neighbours lives on to the next generation.
    4. Any dead cell with exactly three live neighbours becomes a live cell.

- You should write a program that can accept an arbitrary grid of cells.

Maintainability - DSL

"A domain-specific language is a language specialized to a particular application domain.

- *Wikipedia*

Test Double

Test double - patterns

- Replace depended-on components with test specific ones to isolate SUT
- Kinds of Test Doubles
  - Dummy is never called by tested code
  - Test Stubs return test-specific values
  - Test Spies record method calls and arguments for verification by Test Method
  - Mock Objects verify the method calls and arguments themselves
  - Fake Objects provide (apparently) same services in a "lighter" way

Test double - dummy object

- **Def**

  an object without implementation not used within the SUT

- **How**

  create an instance of some object
  pass it as the parameter to the method of the SUT
  Should throw an exception if a method is invoked

- **When**

  as attributes of other objects or arguments of methods on the SUT

- **Utility**

  delete the irrelevant code that would be necessary to build real objects
  making it clear which objects and values are not used by the SUT.

Test double - stub

- Def

  a test-specific object that feeds the desired indirect inputs into the SUT

- How

  define a test-specific implementation of an interface on which the SUT depends
  configured to respond to calls from the SUT with some hardcoded values
  install the Test Stub so that the SUT uses it instead of the real implementation

- When

  inability to control the indirect inputs of the SUT
  the real component is not yet available or is unusable in the development environment

- Utility

  test the behavior of the SUT with various indirect inputs
  inject values that allow us to get past a particular point in the software

Test double - spies

- Def

  an observation point for the indirect outputs of the sut

- How

  install the spy so that the sut uses it instead of the real implementation
  compares values passed to the spy by the SUT with the values expected by the test.

- When

  inability to observe side-effects of invoking methods on the sut
  Interaction with an external system.
  Test asynchronous mechanisms

- Utility

  capture indirect outputs of the SUT.

Test double - mocks

- Def

  special case objects that mimic real objects for testing.
  more capable version of a Test Stub,
  an observation point for the indirect outputs/inputs of the SUT
  all the expected behavior must be specified before the SUT is exercised.

- How

  define a mock object that implements the same interface as an object on which the SUT depends.
  configure the mock object with
    the values with which it should respond to the SUT
    The method calls (complete with expected arguments) to expect from the SUT
  install the mock so that the SUT uses it instead of the real implementation
  Verify the actual arguments received with the expected arguments.

Test double - mocks

- When
  - we need to do behavior verification to avoid having an untested requirement
  - inability to observe side-effects of invoking methods on the S.U.T.
  - we have a part of code not under control
- Utility
  - Fake a single part of the behavior of an object uses by SUT.
  - verify the order of different calls
- caution
  - Facilitate the white box way of testing.
  - make tests harder to write and to understand.
  - Alter the 3-phase test.

Test double - Fake

- Def
  - Simple / light implementation of a component
- How
  - Build a simple implementation of a component
  - Install it so that the SUT uses it instead of the real implementation
- When
  - whenever our S.U.T. depends on other components that
    - are unavailable
    - which make testing difficult or slow
- Utility
  - Make tests fast and isolated
  - Avoid "donothing" mocks
  - Facilitate the black-box style of testing

Test double – patterns

- Test Doubles need to be "installed"
  - Dependency Injection
  - Dependency Lookup
- ... therefore, Design for Testability is key.
- Development + Test Need to Collaborate with Each Other to Do Effective Component Testing

かた

WAhhET

Problem Description

- Given a Wallet containing Stocks, build a function that compute the value of wallet in a currency.

- The Stocks have a quantity and a StockType. The StockType can be for example petroleum, Euros, bitcoins and Dollars.

- To value the portfolio in a Currency you can use external api to provide rate exchanges

Api : https://free.currencyconverterapi.com

Sample : https://gitlab.com/YDanot/finance_api_sample/

INCUBATION

What Does it Take To be Successful?

+ Programing Experience
+ xUnit Experience
+ Testing Experience
+ Design for Testability
- Test Smells
+ Test Automation Patterns
- Test Debt
+ Fanatical Attention to Test Maintainability

= Robust, Maintainable Automated Tests

What's a Code Smell?

A problem visible when looking at test code:

- Tests are hard to understand
- Tests contain coding errors that may result in
  - Missed bugs
  - Erratic Tests
- Tests are difficult or impossible to write
  - No test API on SUT
  - Cannot control initial state of SUT
  - Cannot observe final state of SUT

Common Code Smells

- Conditional Test Logic
- Hard to Test Code
- Obscure Test
- Test Code Duplication
- Test Logic in Production

derrière des problèmes de test se cachent souvent des problèmes de design.

Tests de non regression (TNR)

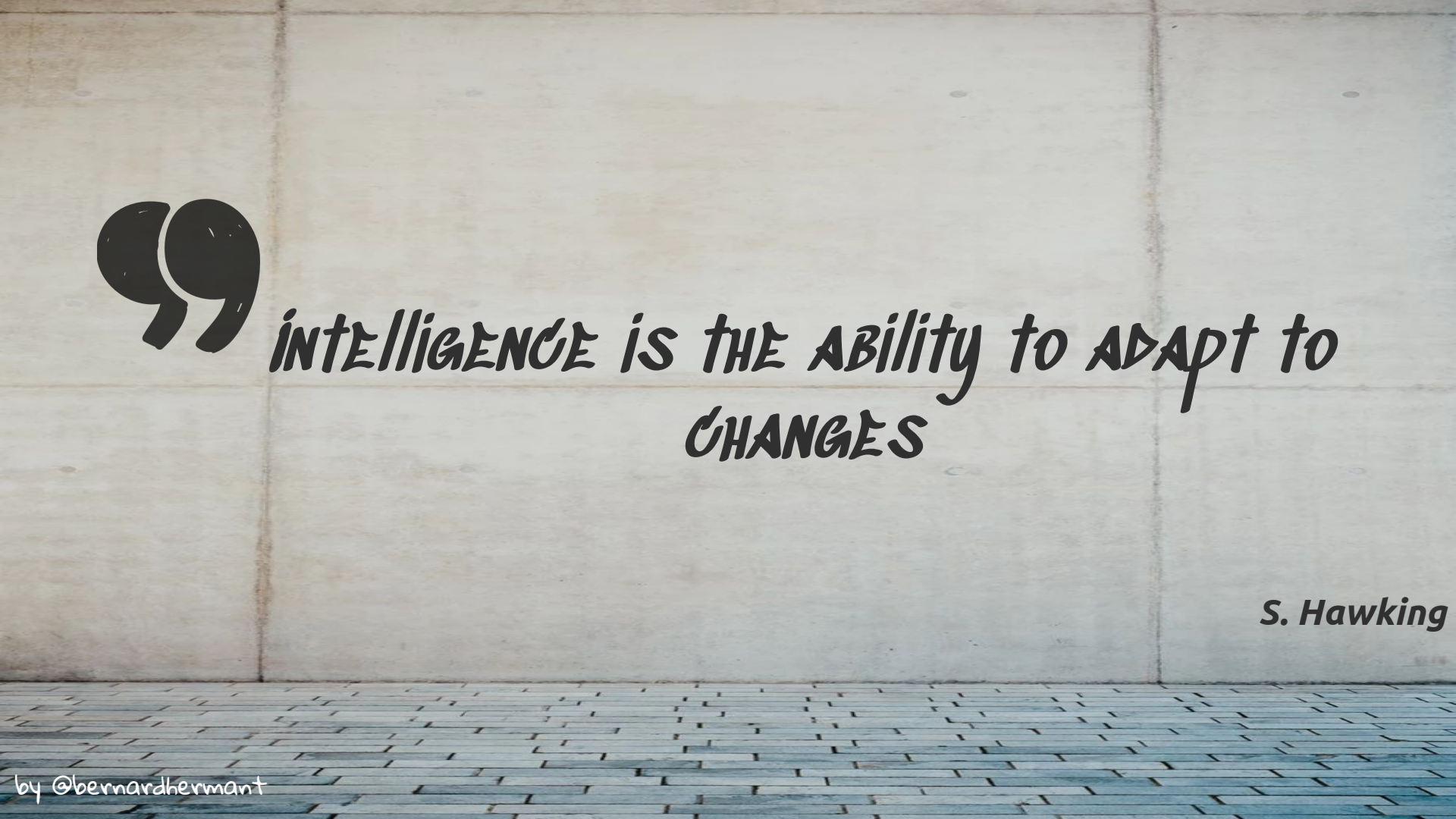
Des assets de tests sont souvent créés avec pour objectif d’avoir des TNR. Souvent à un niveau intégration ou end to end.

Plutôt que de créer des assets spécifique, il est préférable d’utiliser les tests de la pyramid pour faire la non régression.

“NEVER CHANGE A RUNNING SYSTEM !!!

A genius

by @bernardhermant


intelligence is the ability to adapt to changes
S. Hawking
by @bernardhermant


BE WATER MY FRIEND !
by @bernardhermant