

A photograph of a large, rugged mountain peak. The upper portion of the mountain is composed of light-colored, layered rock, likely sandstone or limestone, with distinct horizontal bands and vertical weathering streaks. Sparse green vegetation, including small trees and shrubs, is scattered across the rocky surface. The sky above is a clear, vibrant blue.

SOLID - Single Responsibility Principle
(SRP)

Foi definido por Tom DeMarco e Meilir Page-Jones, eles o chamavam de **coesão**, ou seja, o nível de relacionamento entre elementos de um módulo, entendendo como módulo uma classe, um componente, um arquivo.

Na visão de Robert Martin, devemos ir além do relacionamento, **separando as coisas que mudam por motivos diferentes** e nesse contexto a palavra responsabilidade significa motivo para mudar.

If a class has more than one responsibility, then the responsibilities become coupled and changes to one responsibility may affect others.

A class should have only one reason to change.

Robert C. Martin

Arquitetura Hexagonal (Ports and Adapters)



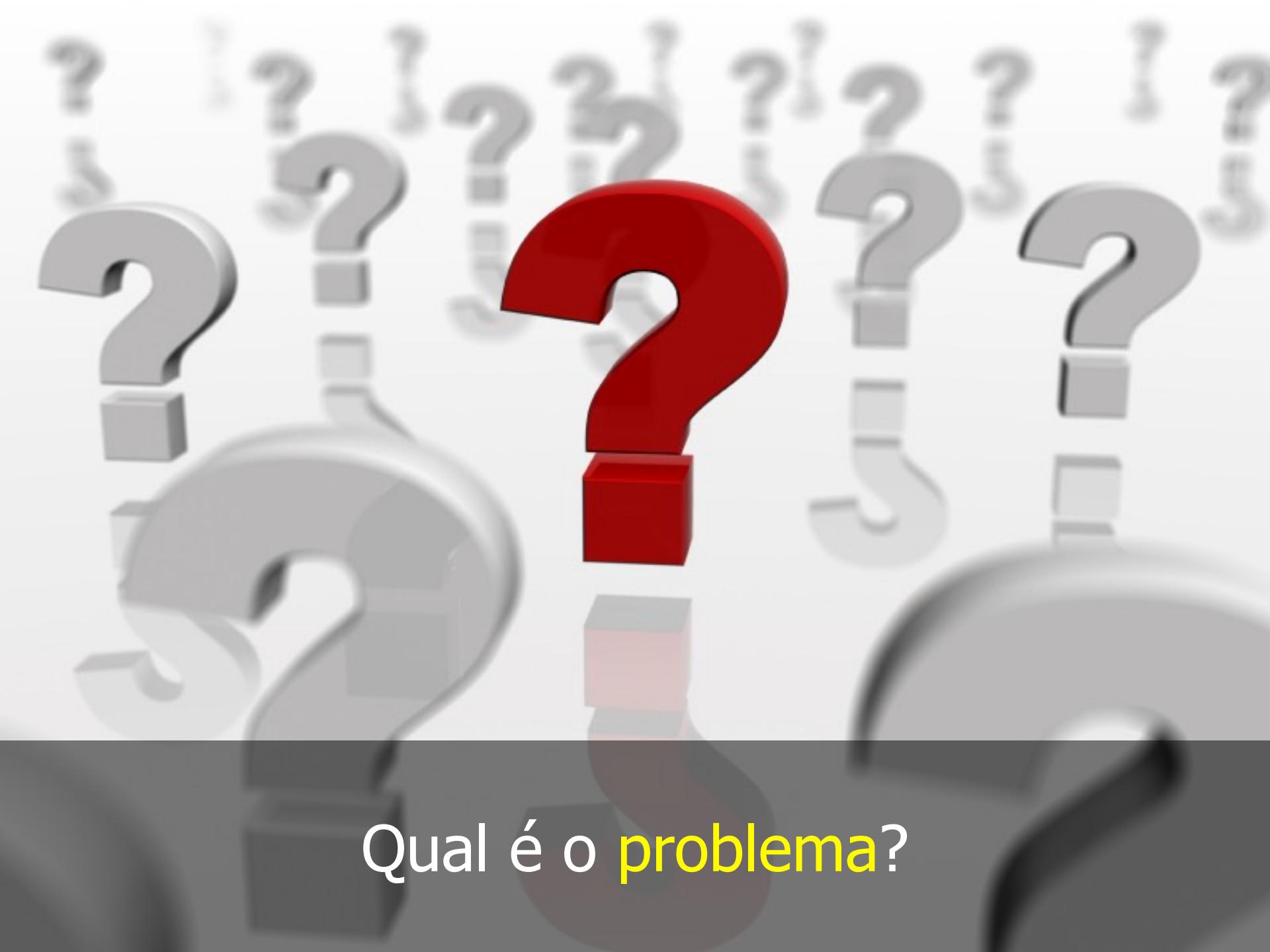
●

●

user.php U X

✖ ⓘ ...

```
1  <?php
2      import_request_variables("pg", "form_");
3      $db = mysql_connect("localhost:/export/mysql/mysql.sock");
4      mysql_select_db("app", $db);
5      $rows = mysql_query("select * from transaction where hash = '$form_hash'", $db);
6      if(mysql_num_rows($rows) == 0) {
7          $error = "Transaction not found";
8      } else {
9          $transaction_hash = mysql_result($rows, 0, 0);
10         $transaction_amount = mysql_result($rows, 0, 1);
11         $transaction_due_date = mysql_result($rows, 0, 2);
12         $today = time();
13         $diff_in_days = ($today - $transaction_due_date) / (60 * 60 * 24);
14         $transaction_penalty = ($transaction_amount * 2) / 100;
15         $transaction_interest = ((($transaction * 0.033) / 100) * $diff_in_days);
16         $transaction_total_amount = $transaction_amount + $transaction_penalty + $transaction_interest;
17     }
18 ?>
19 <html>
20     <head>
21         <title>Transaction</title>
22     </head>
23     <body>
24         <?php
25             if($error) {
26                 echo "<h1>Error accessing transaction</h1>\n";
27                 echo "<p>$error</p>\n";
28             } else {
29                 echo "<h1>Information about $form_hash</h1>\n";
30                 echo "<p>amount: $transaction_amount</p>\n";
31                 echo "<p>due date: $transaction_due_date</p>\n";
32                 echo "<p>penalty: $transaction_penalty</p>\n";
33                 echo "<p>interest: $transaction_interest</p>\n";
34                 echo "<p>total: $transaction_total</p>\n";
35             }
36         </?php>
37     </body>
38 </html>
```



Qual é o problema?



●

●

user.php U X

```
1  <?php
2      import_request_variables("pg", "form_");
3      $db = mysql_connect("localhost:/export/mysql/mysql.sock");
4      mysql_select_db("app", $db);
5      $rows = mysql_query("select * from transaction where hash = '$form_hash'", $db);
6      if(mysql_num_rows($rows) == 0) {
7          $error = "Transaction not found";
8      } else {
9          $transaction_hash = mysql_result($rows, 0, 0);
10         $transaction_amount = mysql_result($rows, 0, 1);
11         $transaction_due_date = mysql_result($rows, 0, 2);
12         $today = time();
13         $diff_in_days = ($today - $transaction_due_date) / (60 * 60 * 24);
14         $transaction_penalty = ($transaction_amount * 2) / 100;
15         $transaction_interest = (($transaction * 0.033) / 100) * $diff_in_days;
16         $transaction_total_amount = $transaction_amount + $transaction_penalty + $transaction_interest;
17     }
18 ?>
19 <html>
20     <head>
21         <title>Transaction</title>
22     </head>
23     <body>
24         <?php
25             if($error) {
26                 echo "<h1>Error accessing transaction</h1>\n";
27                 echo "<p>$error</p>\n";
28             } else {
29                 echo "<h1>Information about $form_hash</h1>\n";
30                 echo "<p>amount: $transaction_amount</p>\n";
31                 echo "<p>due date: $transaction_due_date</p>\n";
32                 echo "<p>penalty: $transaction_penalty</p>\n";
33                 echo "<p>interest: $transaction_interest</p>\n";
34                 echo "<p>total: $transaction_total</p>\n";
35             }
36         </?php>
37     </body>
38 </html>
```



●

●

user.php U X

```
1  <?php
2      import_request_variables("pg", "form_");
3      $db = mysql_connect("localhost:/export/mysql/mysql.sock");
4      mysql_select_db("app", $db);
5      $rows = mysql_query("select * from transaction where hash = '$form_hash'", $db);
6      if(mysql_num_rows($rows) == 0) {
7          $error = "Transaction not found";
8      } else {
9          $transaction_hash = mysql_result($rows, 0, 0);
10         $transaction_amount = mysql_result($rows, 0, 1);
11         $transaction_due_date = mysql_result($rows, 0, 2);
12         $today = time();
13         $diff_in_days = ($today - $transaction_due_date) / (60 * 60 * 24);
14         $transaction_penalty = ($transaction_amount * 2) / 100;
15         $transaction_interest = ((($transaction * 0.033) / 100) * $diff_in_days);
16         $transaction_total_amount = $transaction_amount + $transaction_penalty + $transaction_interest;
17     }
18 ?>
19 <html>
20     <head>
21         <title>Transaction</title>
22     </head>
23     <body>
24         <?php
25             if($error) {
26                 echo "<h1>Error accessing transaction</h1>\n";
27                 echo "<p>$error</p>\n";
28             } else {
29                 echo "<h1>Information about $form_hash</h1>\n";
30                 echo "<p>amount: $transaction_amount</p>\n";
31                 echo "<p>due date: $transaction_due_date</p>\n";
32                 echo "<p>penalty: $transaction_penalty</p>\n";
33                 echo "<p>interest: $transaction_interest</p>\n";
34                 echo "<p>total: $transaction_total</p>\n";
35             }
36         </?php>
37     </body>
38 </html>
```



●

●

user.php U X

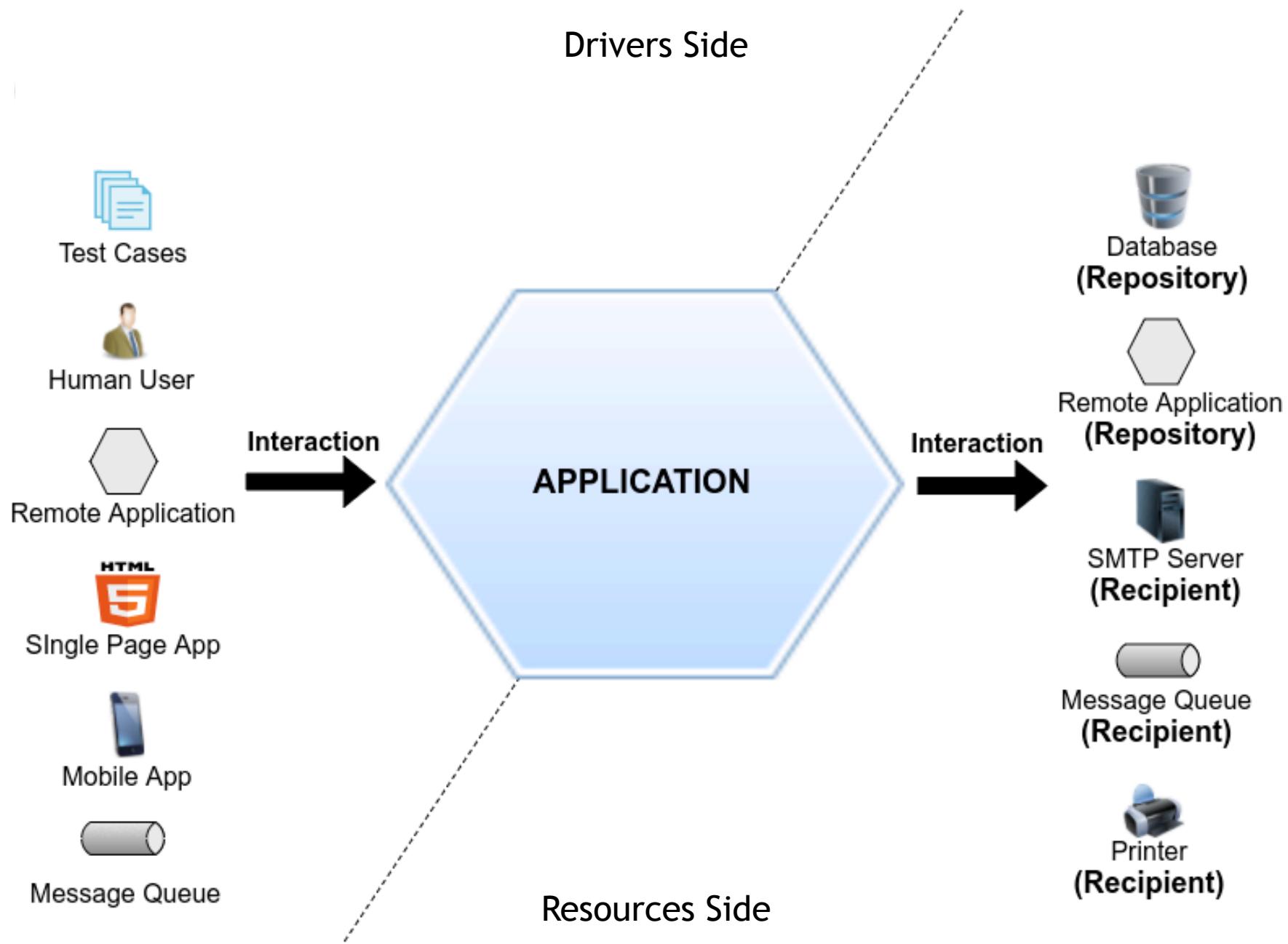
✖ ⓘ ...

```
1  <?php
2      import_request_variables("pg", "form_");
3      $db = mysql_connect("localhost:/export/mysql/mysql.sock");
4      mysql_select_db("app", $db);
5      $rows = mysql_query("select * from transaction where hash = '$form_hash'", $db);
6      if(mysql_num_rows($rows) == 0) {
7          $error = "Transaction not found";
8      } else {
9          $transaction_hash = mysql_result($rows, 0, 0);
10         $transaction_amount = mysql_result($rows, 0, 1);
11         $transaction_due_date = mysql_result($rows, 0, 2);
12         $today = time();
13         $diff_in_days = ($today - $transaction_due_date) / (60 * 60 * 24);
14         $transaction_penalty = ($transaction_amount * 2) / 100;
15         $transaction_interest = ((($transaction * 0.033) / 100) * $diff_in_days);
16         $transaction_total_amount = $transaction_amount + $transaction_penalty + $transaction_interest;
17     }
18 ?>
19 <html>
20     <head>
21         <title>Transaction</title>
22     </head>
23     <body>
24         <?php
25             if($error) {
26                 echo "<h1>Error accessing transaction</h1>\n";
27                 echo "<p>$error</p>\n";
28             } else {
29                 echo "<h1>Information about $form_hash</h1>\n";
30                 echo "<p>amount: $transaction_amount</p>\n";
31                 echo "<p>due date: $transaction_due_date</p>\n";
32                 echo "<p>penalty: $transaction_penalty</p>\n";
33                 echo "<p>interest: $transaction_interest</p>\n";
34                 echo "<p>total: $transaction_total</p>\n";
35             }
36         </?php>
37     </body>
38 </html>
```



user.php U X

```
1  <?php
2      import_request_variables("pg", "form_");
3      $db = mysql_connect("localhost:/export/mysql/mysql.sock");
4      mysql_select_db("app", $db);
5      $rows = mysql_query("select * from transaction where hash = '$form_hash'", $db);
6      if(mysql_num_rows($rows) == 0) {
7          $error = "Transaction not found";
8      } else {
9          $transaction_hash = mysql_result($rows, 0, 0);
10         $transaction_amount = mysql_result($rows, 0, 1);
11         $transaction_due_date = mysql_result($rows, 0, 2);
12         $today = time();
13         $diff_in_days = ($today - $transaction_due_date) / (60 * 60 * 24);
14         $transaction_penalty = ($transaction_amount * 2) / 100;
15         $transaction_interest = ((($transaction * 0.033) / 100) * $diff_in_days);
16         $transaction_total_amount = $transaction_amount + $transaction_penalty + $transaction_interest;
17     }
18 ?>
19 <html>
20     <head>
21         <title>Transaction</title>
22     </head>
23     <body>
24         <?php
25             if($error) {
26                 echo "<h1>Error accessing transaction</h1>\n";
27                 echo "<p>$error</p>\n";
28             } else {
29                 echo "<h1>Information about $form_hash</h1>\n";
30                 echo "<p>amount: $transaction_amount</p>\n";
31                 echo "<p>due date: $transaction_due_date</p>\n";
32                 echo "<p>penalty: $transaction_penalty</p>\n";
33                 echo "<p>interest: $transaction_interest</p>\n";
34                 echo "<p>total: $transaction_total</p>\n";
35             }
36         </?php>
37     </body>
38 </html>
```



"Allow an application to equally be driven by users, programs, automated test or batch scripts, and to be developed and tested in isolation from its eventual run-time devices and databases"

Alistair Cockburn



Porque um **hexagono**?

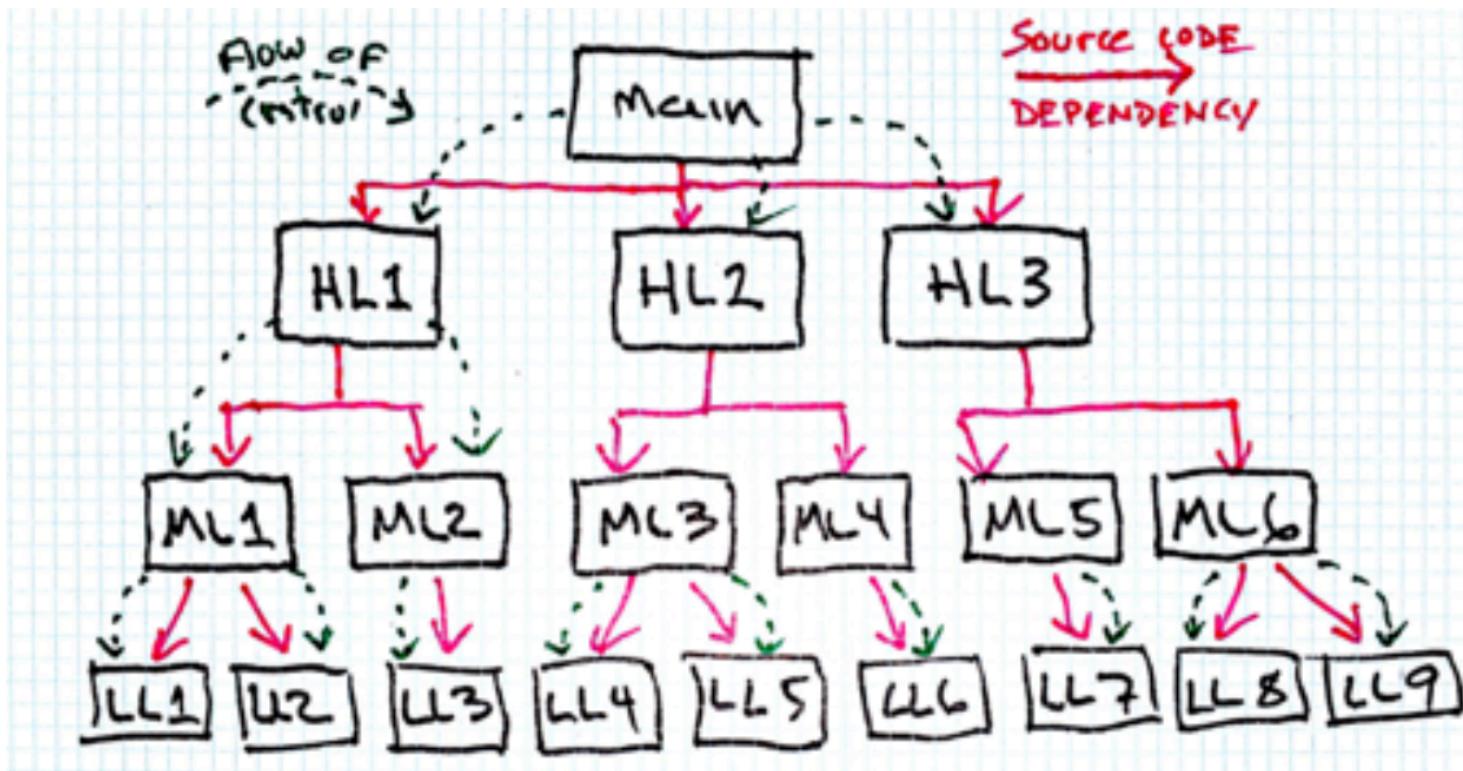
"The hexagon is not a hexagon because the number six is important, but rather to allow the people doing the drawing to have room to insert ports and adapters as they need, not being constrained by a one-dimensional layered drawing. The term hexagonal architecture comes from this visual effect"

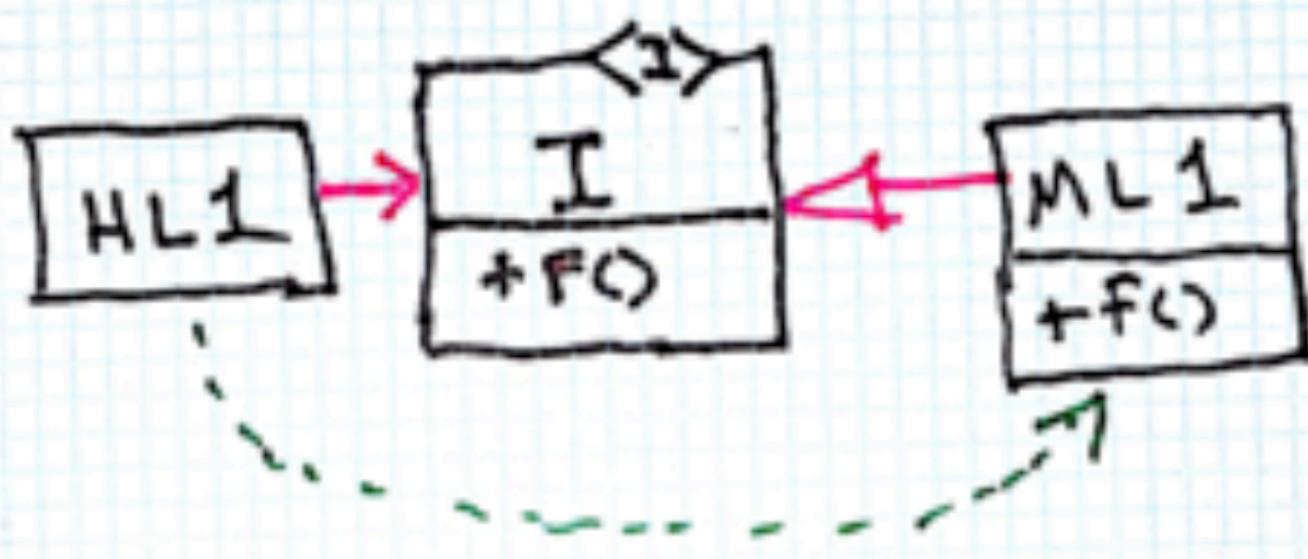
Alistair Cockburn

A photograph of a rugged mountain peak with distinct horizontal sedimentary rock layers. The upper portion of the mountain is covered in sparse green coniferous trees. The lower slopes are exposed, showing various shades of tan, brown, and yellow, with some darker, weathered areas. The sky above is a clear, pale blue.

SOLID - Dependency Inversion Principle (DIP)

High-level modules should not depend on
low-level modules, both should depend on
abstractions







Finished after 0.063 seconds

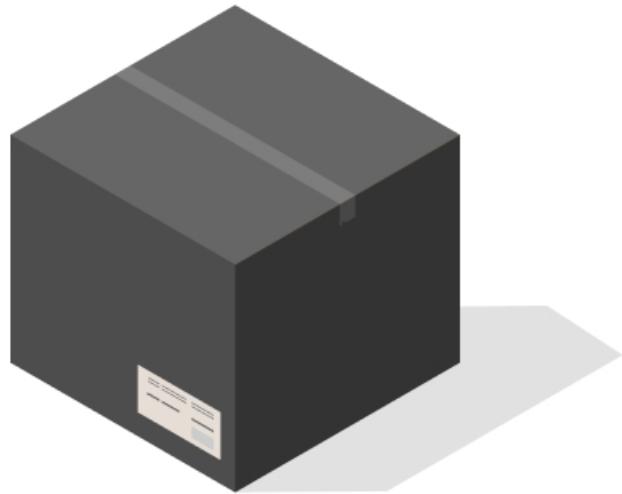
Runs: 7/7

Errors: 0

Failures: 0

- A stack [Runner: JUnit 5] (0.000 s)
 - is instantiated with new Stack() (0.000 s)
 - when new (0.000 s)
 - throws EmptyStackException when peeked (0.000 s)
 - throws EmptyStackException when popped (0.000 s)
 - is empty (0.000 s)
 - after pushing an element (0.000 s)
 - return (0.000 s) Go to File
 - return (0.000 s) ed but remains not empty (0.000 s)
 - it is now empty (0.000 s) Run
 - Debug
- Tipos de testes automatizados

QA testers

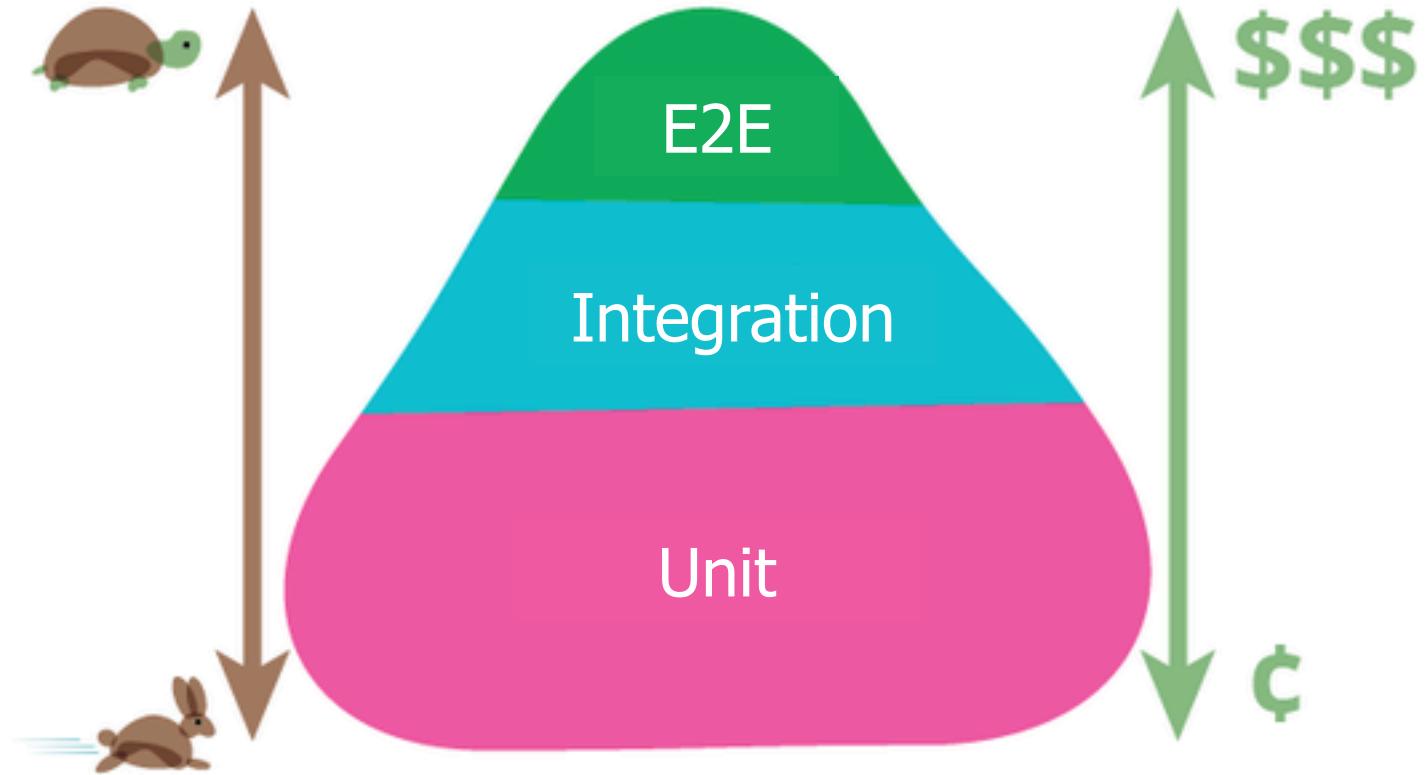


**Black box - we do not
know anything**

Developers

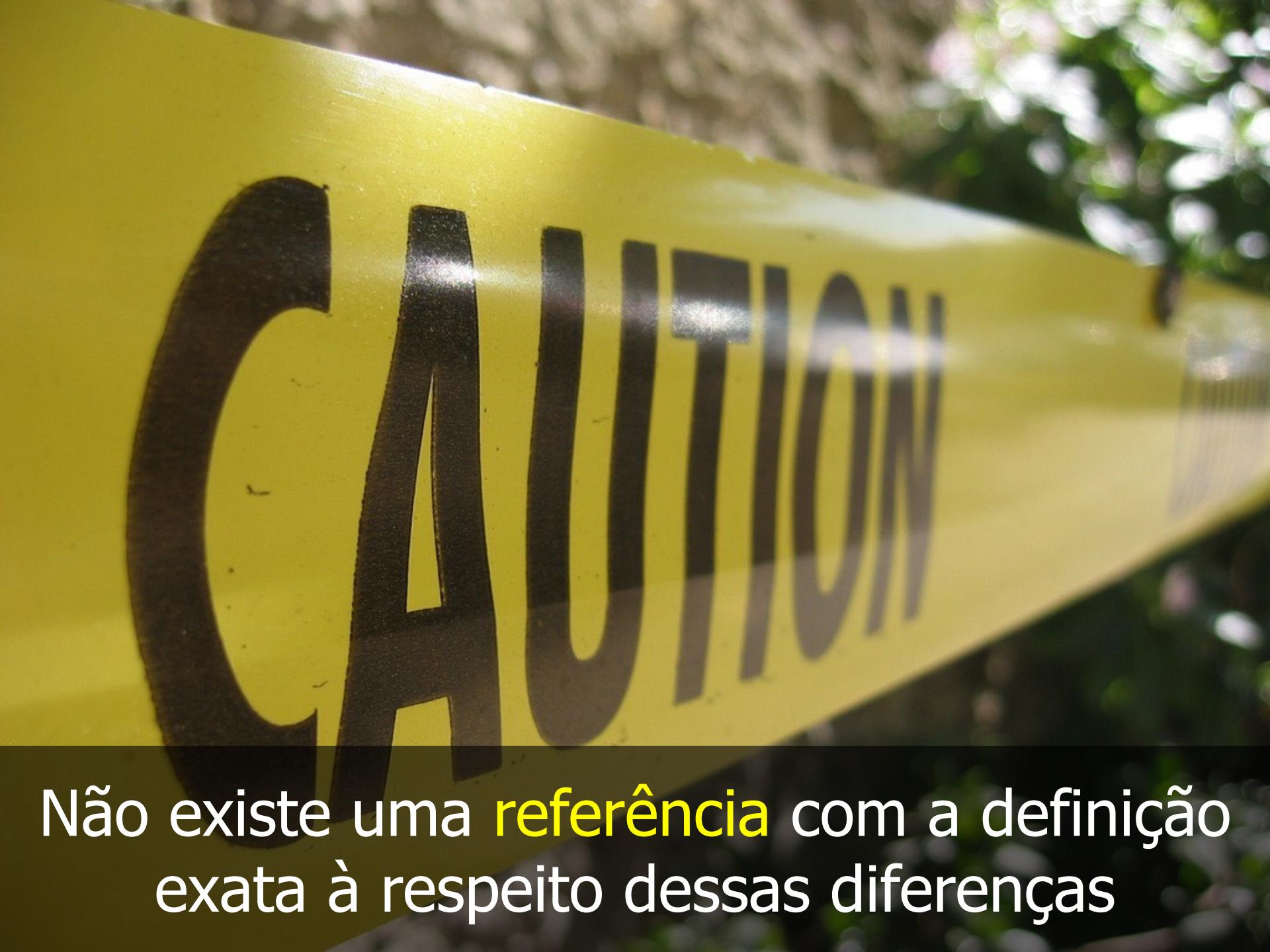


**White box - we know
everything**





Qual é a diferença entre eles?

A close-up photograph of a yellow caution tape. The word "CAUTION" is printed in large, bold, black capital letters. The tape is positioned diagonally across the frame, with a blurred background of green foliage visible behind it.

CAUTION

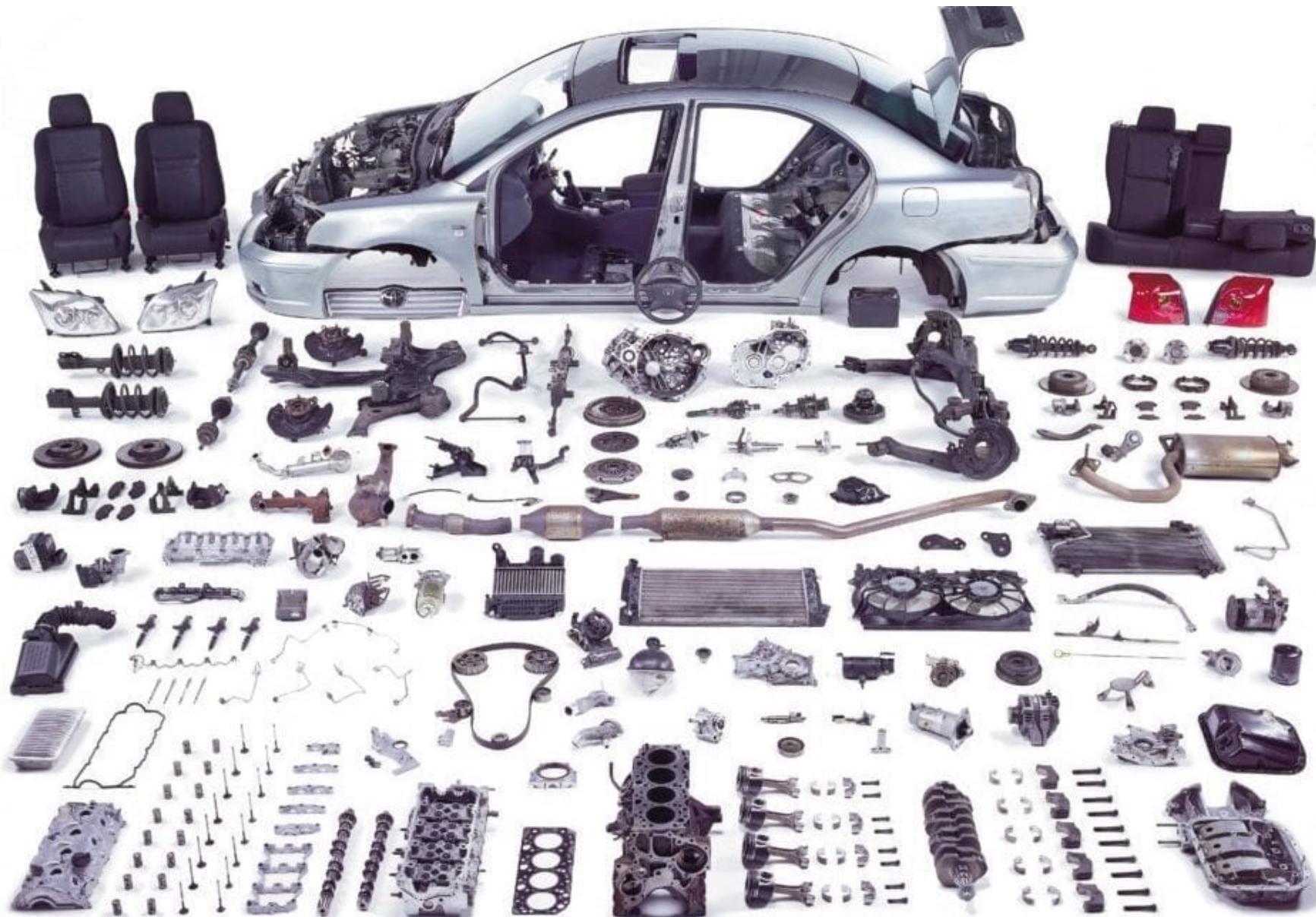
Não existe uma referência com a definição exata à respeito dessas diferenças

Unit Tests

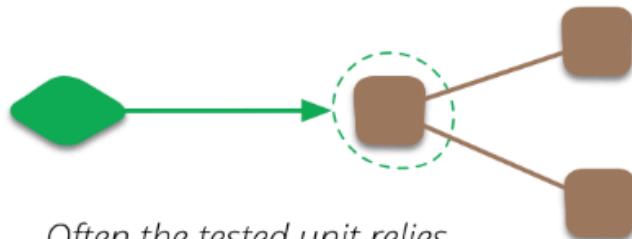
São testes de unidade, não necessariamente unitários, que podem ou não envolver mais de uma classe, que não representam um requisito funcional e sem qualquer interação com recursos externos como um banco de dados, uma API ou o sistema de arquivos



São muito rápidos, estáveis e resistentes

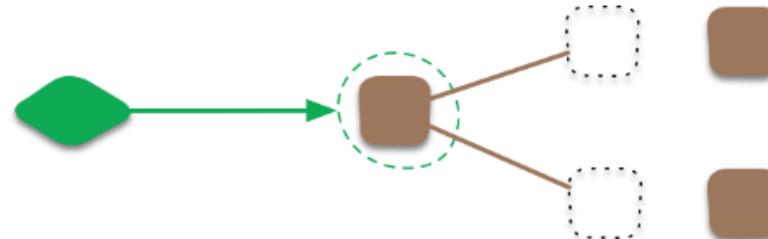


Sociable Tests



Often the tested unit relies on other units to fulfill its behavior

Solitary Tests



Some unit testers prefer to isolate the tested unit

But not all unit testers use solitary unit tests. Indeed when xunit testing began in the 90's we made no attempt to go solitary unless communicating with the collaborators was awkward (such as a remote credit card verification system). We didn't find it difficult to track down the actual fault, even if it caused neighboring tests to fail. So we felt allowing our tests to be sociable didn't lead to problems in practice.

Indeed using sociable unit tests was one of the reasons we were criticized for our use of the term "unit testing". I think that the term "unit testing" is appropriate because these tests are tests of the behavior of a single unit. We write the tests assuming everything other than that unit is working correctly.

Integration Tests

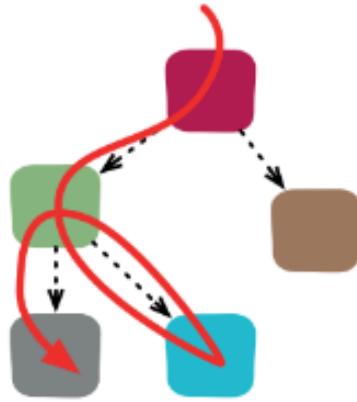
Testam componentes diferentes, pertencentes à múltiplas camadas, normalmente envolvendo recursos externos, sejam eles reais ou não, ou seja, o fato de utilizar um Test Pattern como um stub ou mock não torna o teste de unidade.

Em geral são mais lentos por fazerem I/O

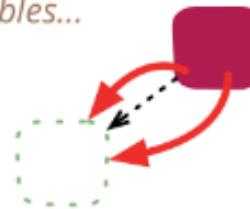


São mais lentos porque acabam **interagindo**
com recursos externos e fazendo I/O

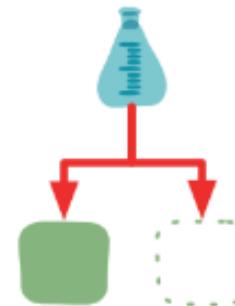
*Integration Testing
commonly refers to
broad tests done with
many modules active...*



*...but it can be done with
narrow tests of interactions
with individual Test
Doubles...*



*...supported by Contract
Tests to ensure the
faithfulness of the double*



narrow integration tests

- exercise only that portion of the code in my service that talks to a separate service
- uses test doubles of those services, either in process or remote
- thus consist of many narrowly scoped tests, often no larger in scope than a unit test (and usually run with the same test framework that's used for unit tests)

broad integration tests

- require live versions of all services, requiring substantial test environment and network access
- exercise code paths through all services, not just code responsible for interactions



Um teste com as dependências **mockadas**
se torna um teste de unidade?

E2E Tests

Replicam o ambiente do usuário final, ou seja, são testes executados de ponta a ponta envolvendo o frontend, backend e banco de dados

Ferramentas

Esse tipo de teste tem uma dependência muito grande de ferramentas como o Selenium, Cypress, Playwright ou variações envolvendo WebDriver ou Puppeteer



Além de lentes, são **frágeis** e **sensíveis** a modificações, principalmente no frontend



CAUTION

Não existe um teste melhor que outro, o ideal é ter uma combinação deles

A close-up photograph of a light-colored wooden board featuring a grid of circular holes. A single, dark, elongated rectangular slot is positioned diagonally across the board, intersecting several of the circular holes. The wood grain is visible on the surface.

Test Patterns

CAUTION

Testar nem sempre é simples

Exemplo

Deve fazer um pedido com 3 itens em dólares

Dado um novo pedido com 3 itens associados, um Livro de \$50,00, um CD de \$20,00 e um DVD de \$30,00

Quando o pedido for realizado

Então deve ser retornado uma confirmação do pedido contendo o código, juntamente com o total do pedido de R\$550,00, se a cotação do dólar for R\$5,50 e o status aguardando pagamento



O que fazer quando existem entradas e saídas indiretas no componente testado?

Um **test double** é um padrão que tem o objetivo de substituir um DOC (depended-on component) em um determinado tipo de teste por motivos de performance ou segurança

CAUTION

Nem tudo é mock



The Addison-Wesley Signature Series

xUNIT TEST PATTERNS

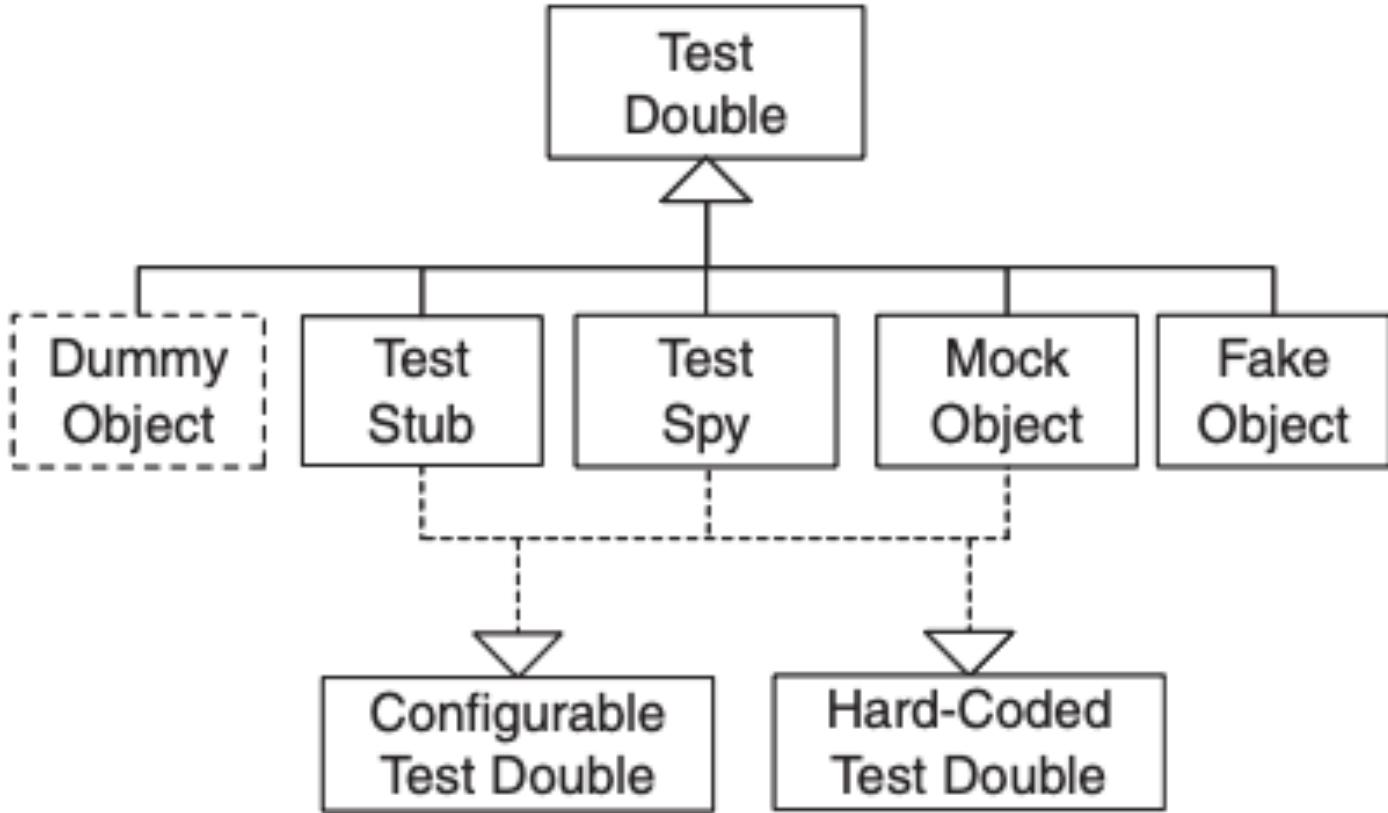
REFACTORING TEST CODE

GERARD MESZAROS

A MARTIN FOWLER SIGNATURE
Book by Martin Fowler



Foreword by Martin Fowler



Test Patterns

Dummy: Objetos que criamos apenas para completar a lista de parâmetros que precisamos passar para invocar um determinado método

Test Patterns

Dummy: Objetos que criamos apenas para completar a lista de parâmetros que precisamos passar para invocar um determinado método

Stubs: Objetos que retornam respostas prontas, definidas para um determinado teste, por questão de performance ou segurança (exemplo: quando eu executar o método fazer pedido preciso que o método pegar cotação do dólar retorne R\$3,00)

Test Patterns

Dummy: Objetos que criamos apenas para completar a lista de parâmetros que precisamos passar para invocar um determinado método

Stubs: Objetos que retornam respostas prontas, definidas para um determinado teste, por questão de performance ou segurança (exemplo: quando eu executar o método fazer pedido preciso que o método pegar cotação do dólar retorne R\$3,00)

Spies: Objetos que "espioram" a execução do método e armazenam os resultados para verificação posterior (exemplo: quando eu executar o método fazer pedido preciso saber se o método enviar email foi invocado internamente e com quais parâmetros)

Test Patterns

Dummy: Objetos que criamos apenas para completar a lista de parâmetros que precisamos passar para invocar um determinado método

Stubs: Objetos que retornam respostas prontas, definidas para um determinado teste, por questão de performance ou segurança (exemplo: quando eu executar o método fazer pedido preciso que o método pegar cotação do dólar retorne R\$3,00)

Spies: Objetos que "espioram" a execução do método e armazenam os resultados para verificação posterior (exemplo: quando eu executar o método fazer pedido preciso saber se o método enviar email foi invocado internamente e com quais parâmetros)

Mocks: Objetos similares a stubs e spies, permitem que você diga exatamente o que quer que ele faça e o teste vai quebrar se isso não acontecer

Test Patterns

Dummy: Objetos que criamos apenas para completar a lista de parâmetros que precisamos passar para invocar um determinado método

Stubs: Objetos que retornam respostas prontas, definidas para um determinado teste, por questão de performance ou segurança (exemplo: quando eu executar o método fazer pedido preciso que o método pegar cotação do dólar retorne R\$3,00)

Spies: Objetos que "espioram" a execução do método e armazenam os resultados para verificação posterior (exemplo: quando eu executar o método fazer pedido preciso saber se o método enviar email foi invocado internamente e com quais parâmetros)

Mocks: Objetos similares a stubs e spies, permitem que você diga exatamente o que quer que ele faça e o teste vai quebrar se isso não acontecer

Fake: Objetos que tem implementações que simulam o funcionamento da instância real, que seria utilizada em produção (exemplo: uma base de dados em memória)

Mocks Aren't Stubs

martinfowler.com/articles/mocksArentStubs.html

martinFowler.com

Refactoring Agile Architecture About Thoughtworks

Mocks Aren't Stubs

The term 'Mock Objects' has become a popular one to describe special case objects that mimic real objects for testing. Most language environments now have frameworks that make it easy to create mock objects. What's often not realized, however, is that mock objects are but one form of special case test object, one that enables a different style of testing. In this article I'll explain how mock objects work, how they encourage testing based on behavior verification, and how the community around them uses them to develop a different style of testing.

02 January 2007


Martin Fowler

POPULAR
TESTING

CONTENTS

- Regular Tests
- Tests with Mock Objects
 - Using EasyMock
 - The Difference Between Mocks and Stubs
 - Classical and Mockist Testing
 - Choosing Between the Differences
 - Driving TDD
 - Fixture Setup
 - Test Isolation
 - Coupling Tests to Implementations
 - Design Style
- So should I be a classicalist or a mockist?

Table of Contents

<https://martinfowler.com/articles/mocksArentStubs.html>

Clean Coder Blog

https://blog.cleancoder.com/uncle-bob/2014/05/14/TheLittleMocker.html



The Clean Code Blog
by Robert C. Martin (Uncle Bob)

atom/rss feed

- [Space War](#)
11-28-2021
- [Functional Duplications](#)
10-28-2021
- [Roots](#)
09-25-2021
- [More On Types](#)
06-29-2021
- [On Types](#)
06-25-2021
- [if-else-switch](#)
03-09-2021

The Little Mocker
14 May 2014

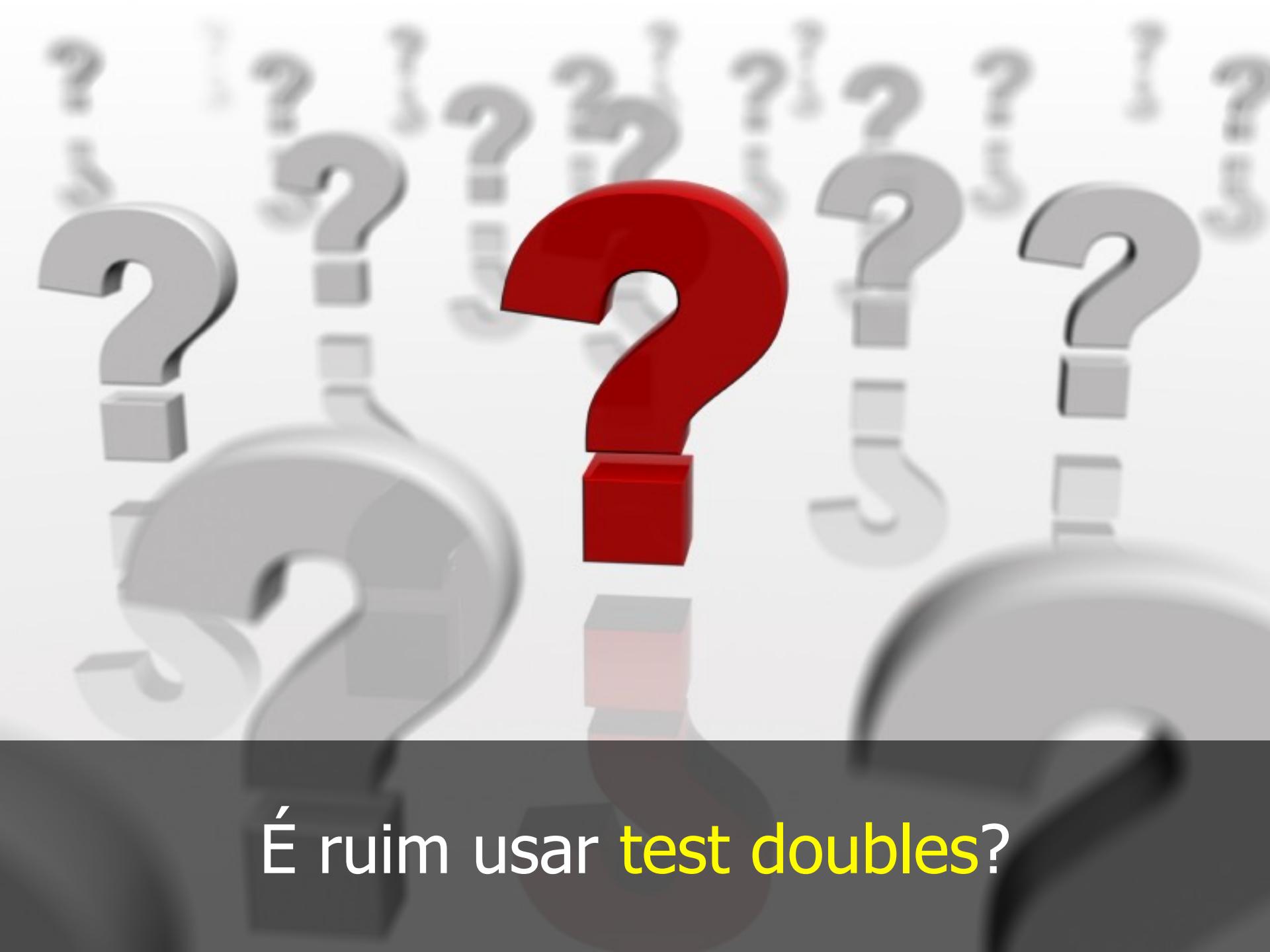
The following is a conversation around mocking:

What is this?:

```
interface Authorizer {  
    public Boolean authorize(String username, String password);  
} >_An interface._
```

So what then, is this?

<https://blog.cleancoder.com/uncle-bob/2014/05/14/TheLittleMocker.html>



É ruim usar test doubles?