

The background is a dark, textured surface composed of a repeating pattern of hexagons. The hexagons are outlined with glowing lines in two colors: a vibrant magenta/pink and a bright cyan/blue. These glowing lines are not uniform; they appear to pulse or have a gradient, giving the pattern a dynamic, futuristic feel. The overall color palette is dark, with the glowing lines providing the primary visual interest.

Hexagonal

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             }
```



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             }
```

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             }
```


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             }
```

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         }
```

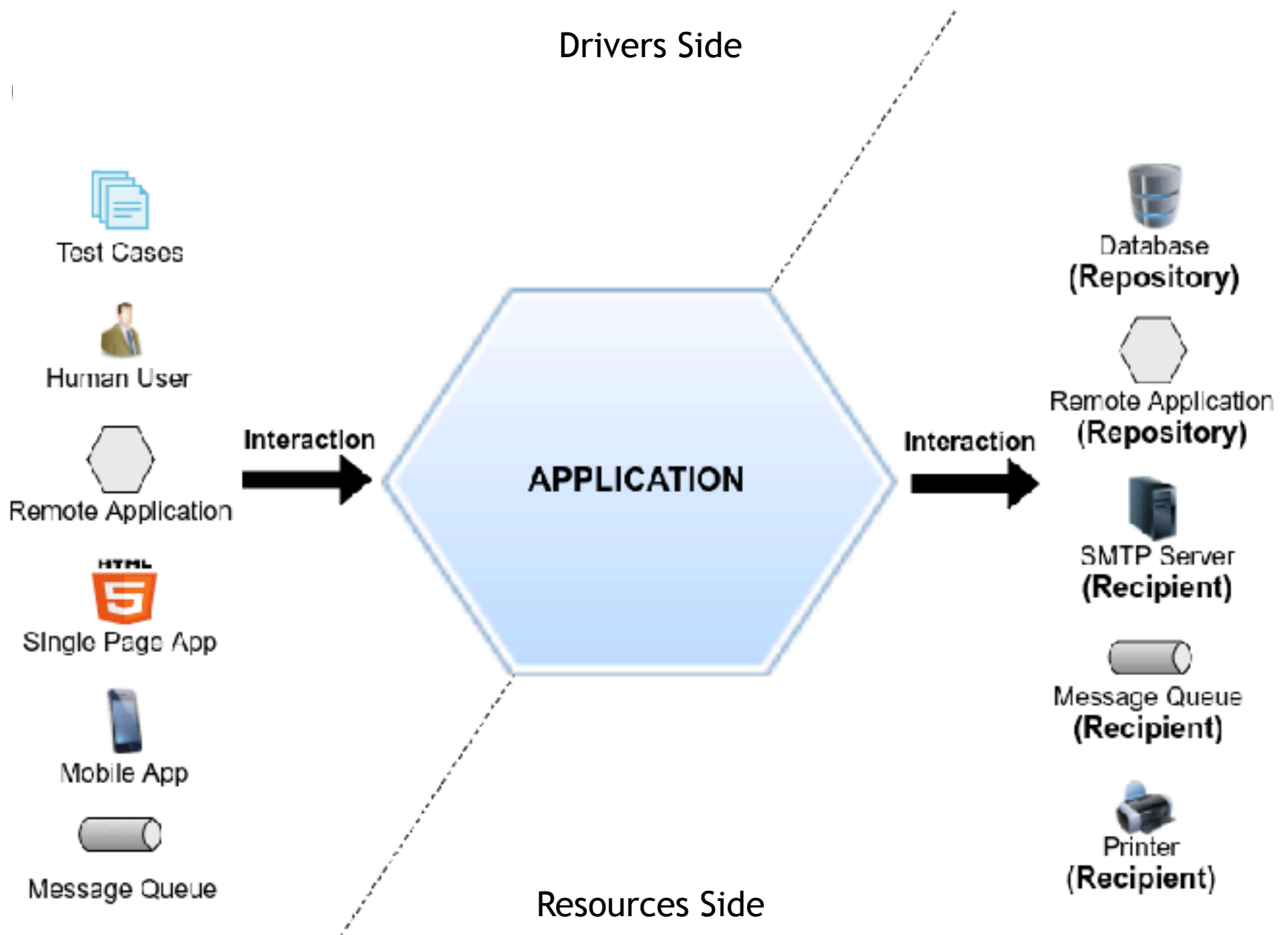
Temos uma **mistura de responsabilidades** como a apresentação das informações na tela, regras de negócio para cálculo da multa e dos juros e acesso a recursos do banco de dados

Single Responsibility Principle

Devemos separar as coisas que mudam por motivos diferentes e nesse contexto a palavra responsabilidade significa motivo para mudar

The SRP is one of the simplest of the principle, and one of the hardest to get right. **If a class has more than one responsibility, then the responsibilities become coupled and changes to one responsibility may affect others**

Robert C. Martin



"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"

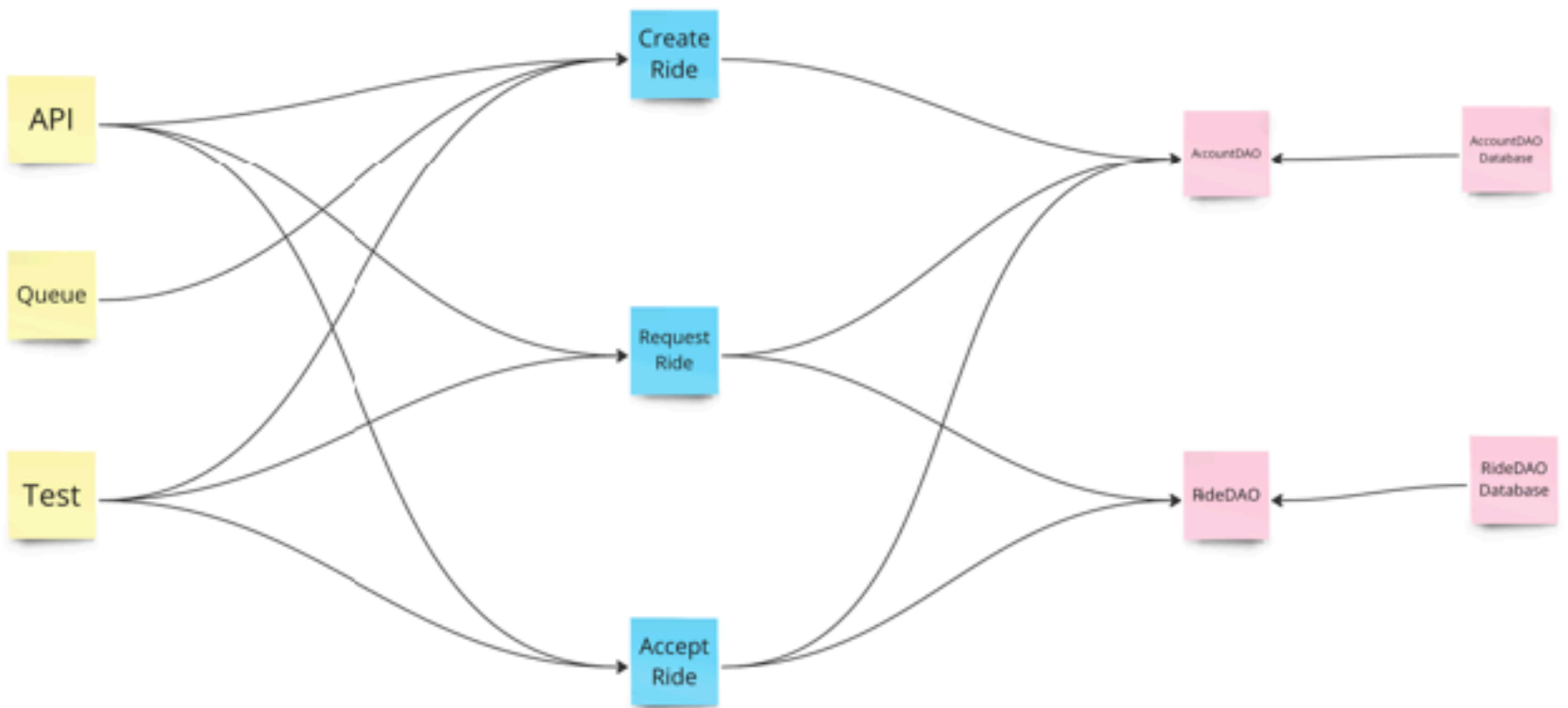
Alistair Cockburn



Porque um **hexagono**?

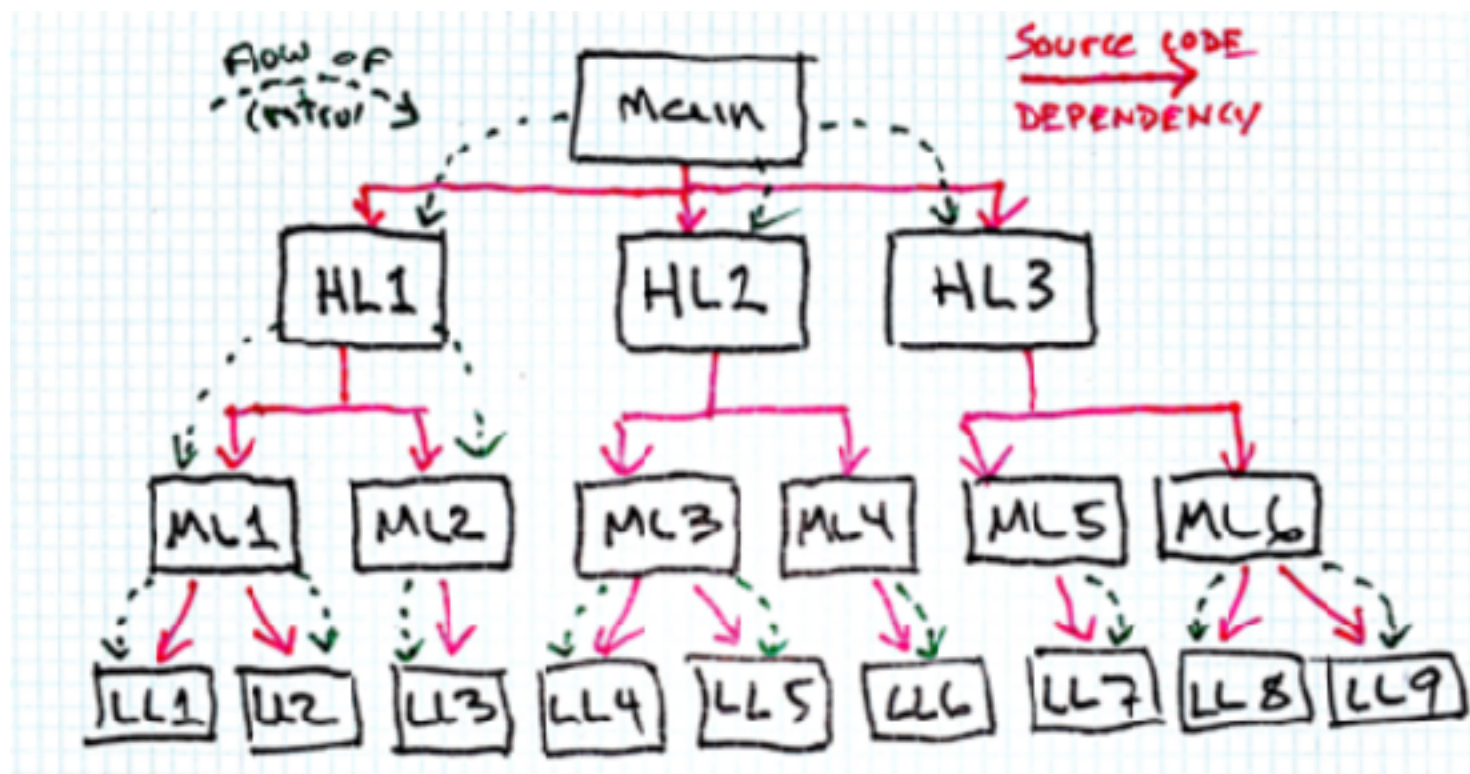
"The hexagon is not a hexagon because the number six is important, but rather to allow the people doing the drawing to have room to insert ports and adapters as they need, not being constrained by a one-dimensional layered drawing. The term hexagonal architecture comes from this visual effect"

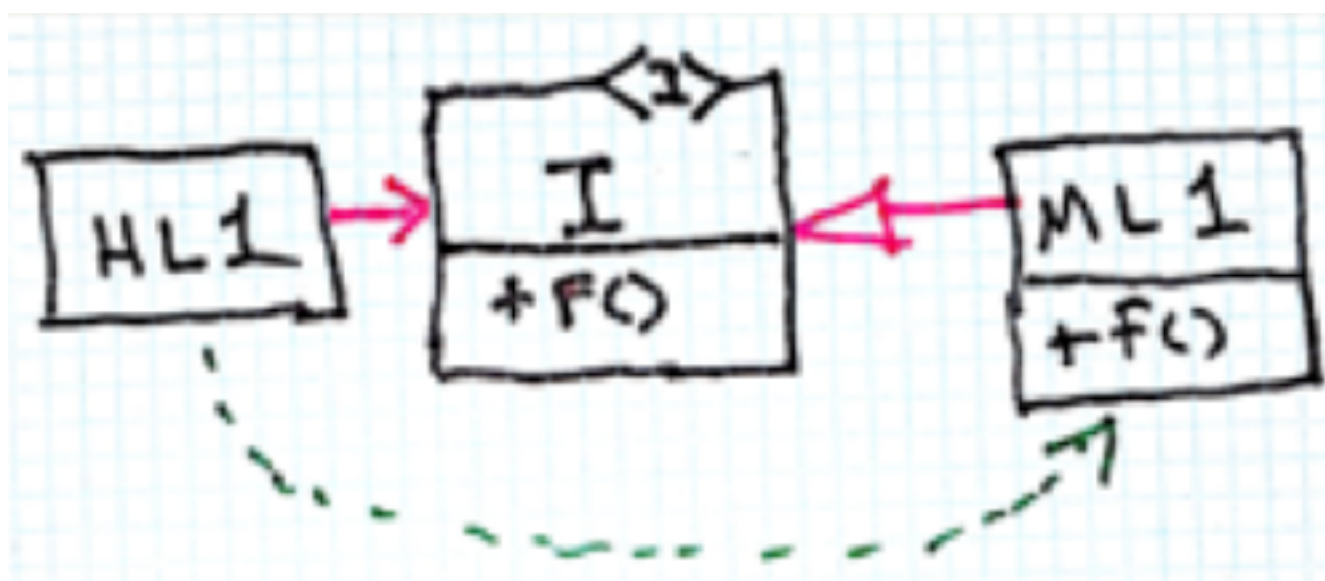
Alistair Cockburn



Dependency Inversion Principle

É a base do desacoplamento permitindo intercambiar dependências de acordo com a necessidade, facilitando os testes e a evolução com o passar do tempo





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

Frankly, it is because more traditional software development methods, such as Structured Analysis and Design, **tend to create software structures in which high level modules depend upon low level modules**, and in which abstractions depend upon details. Thus, the dependency structure of a well designed object oriented program is “inverted” with respect to the dependency structure that normally results from traditional procedural methods.

Robert C. Martin



Faz sentido **aplicar** em qual tipo e tamanho de projeto?



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



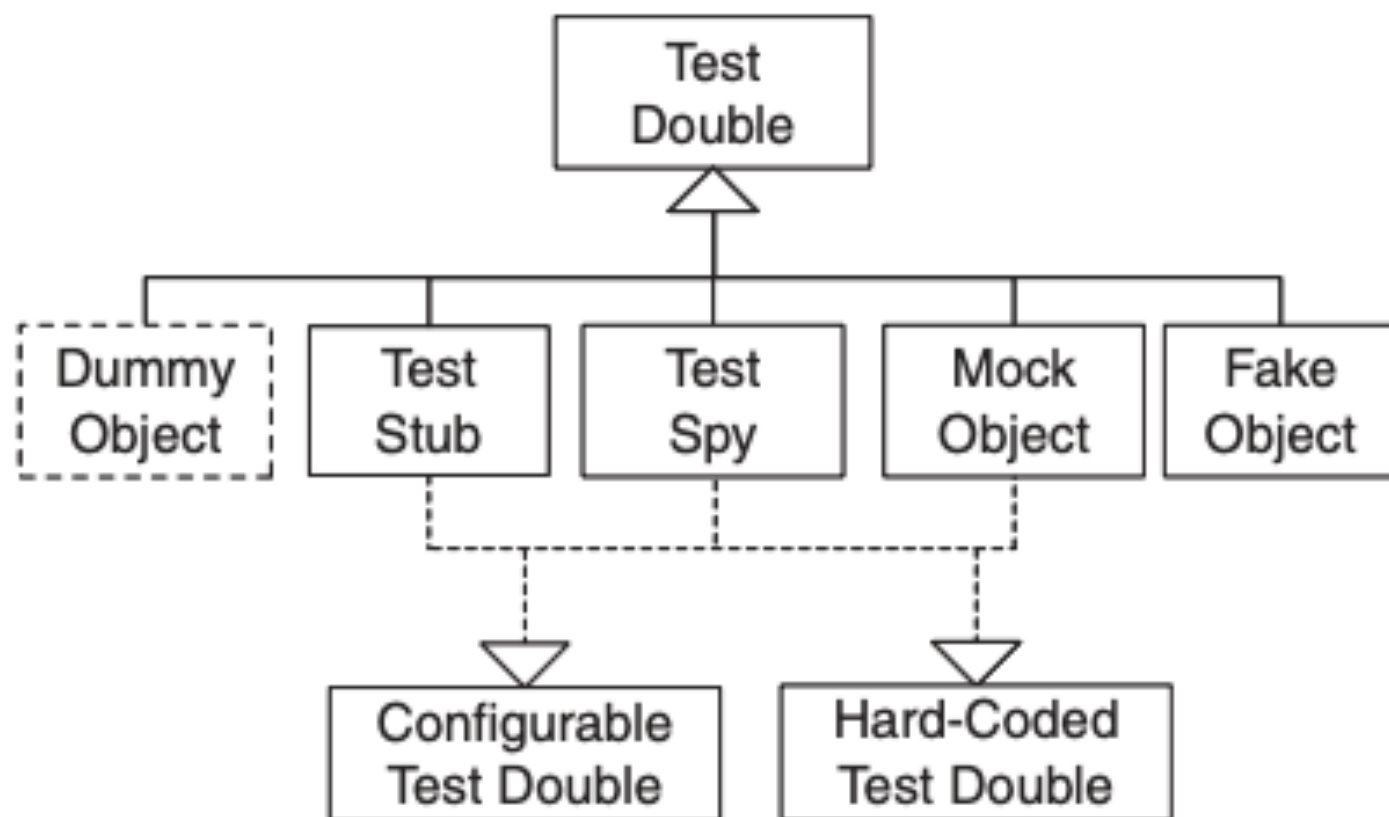
xUNIT TEST PATTERNS

REFACTORING
TEST CODE

GERARD MESZAROS



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 "espionam" 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 "espionam" 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 "espionam" 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

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 Mocklet Testing
- Choosing Between the Differences
 - Driving TDD
 - Fixture Setup
 - Test Isolation
 - Coupling Tests to Implementations
 - Design Style
 - Should I have a class or a mock?

Table of Contents

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



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