mocks e testabilidade

rodrigo couto
rcouto@touchtec.com.br

Touch TECNOLOGIA

Touch
TECNOLOGIA

touch tecnologia

- 12 anos no mercado
- desenvolve software para área da saúde
- foco em java (ee) e web

- ... e estamos contratando!

unit test

<def term="unit test">

verifica o comportamento de uma pequena parte do sistema

unit test

saídas diretas ou indiretas

asserts

verifica o comportamento de uma pequena parte do sistema

system-under-test (SUT)

método, classe

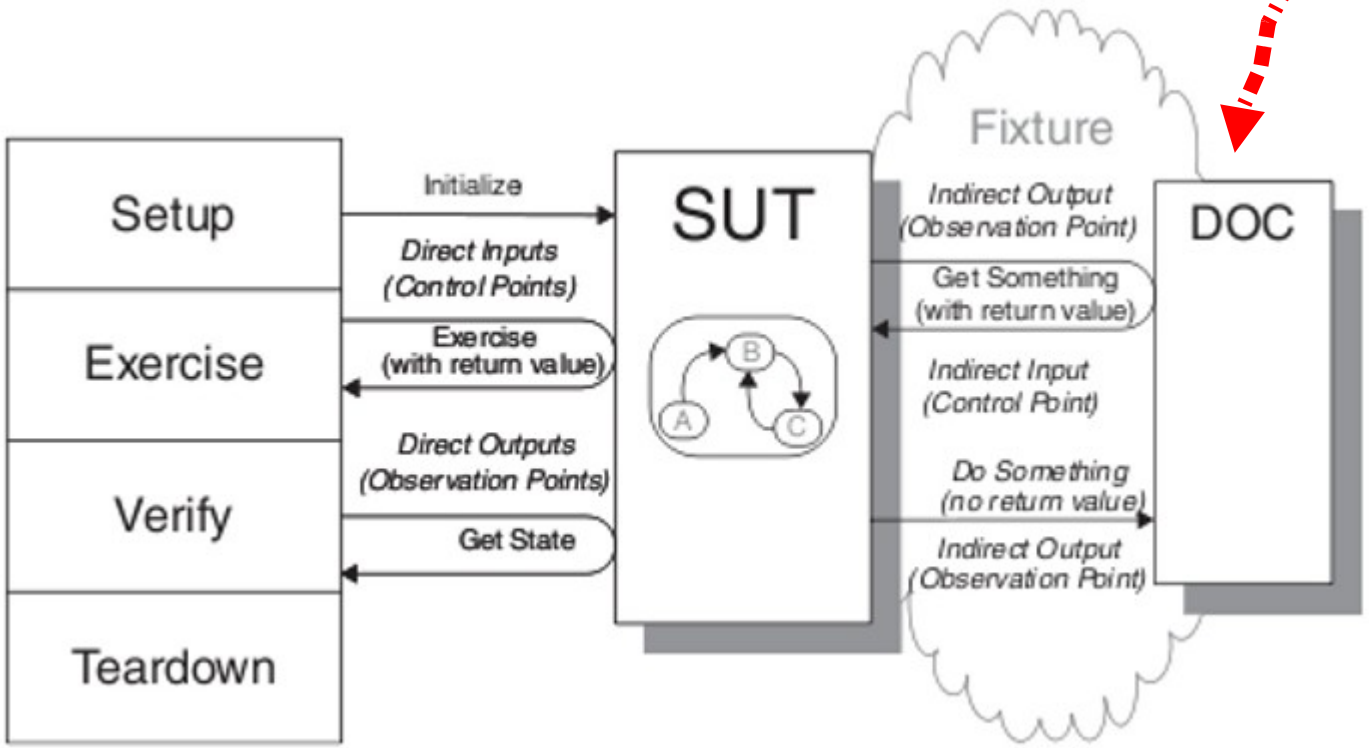
xUnit (p.817)

unit test

```
</def>
```

unit test – pontos de interação

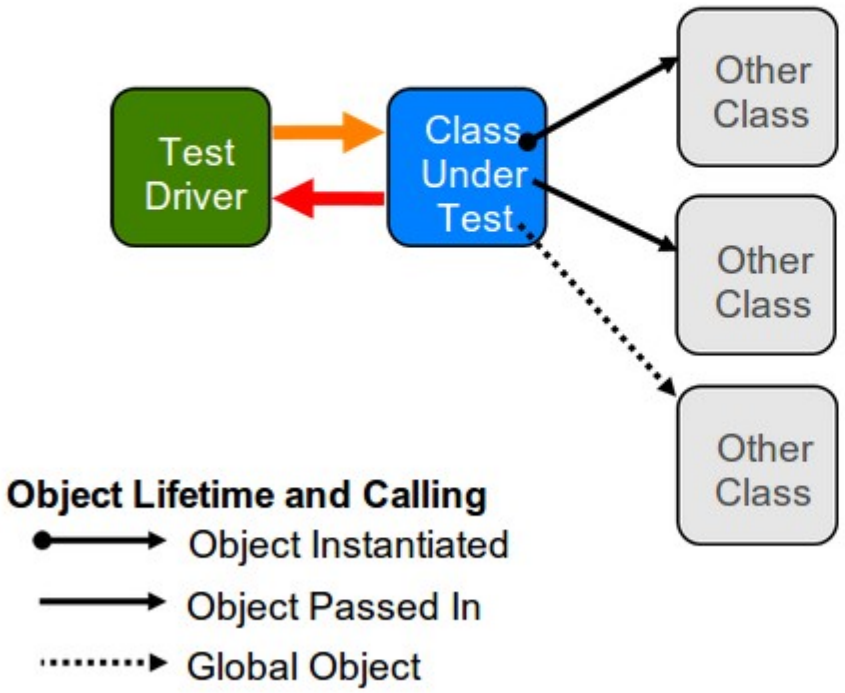
depended-on component



xUnit (p.66)

Touch TECNOLOGIA

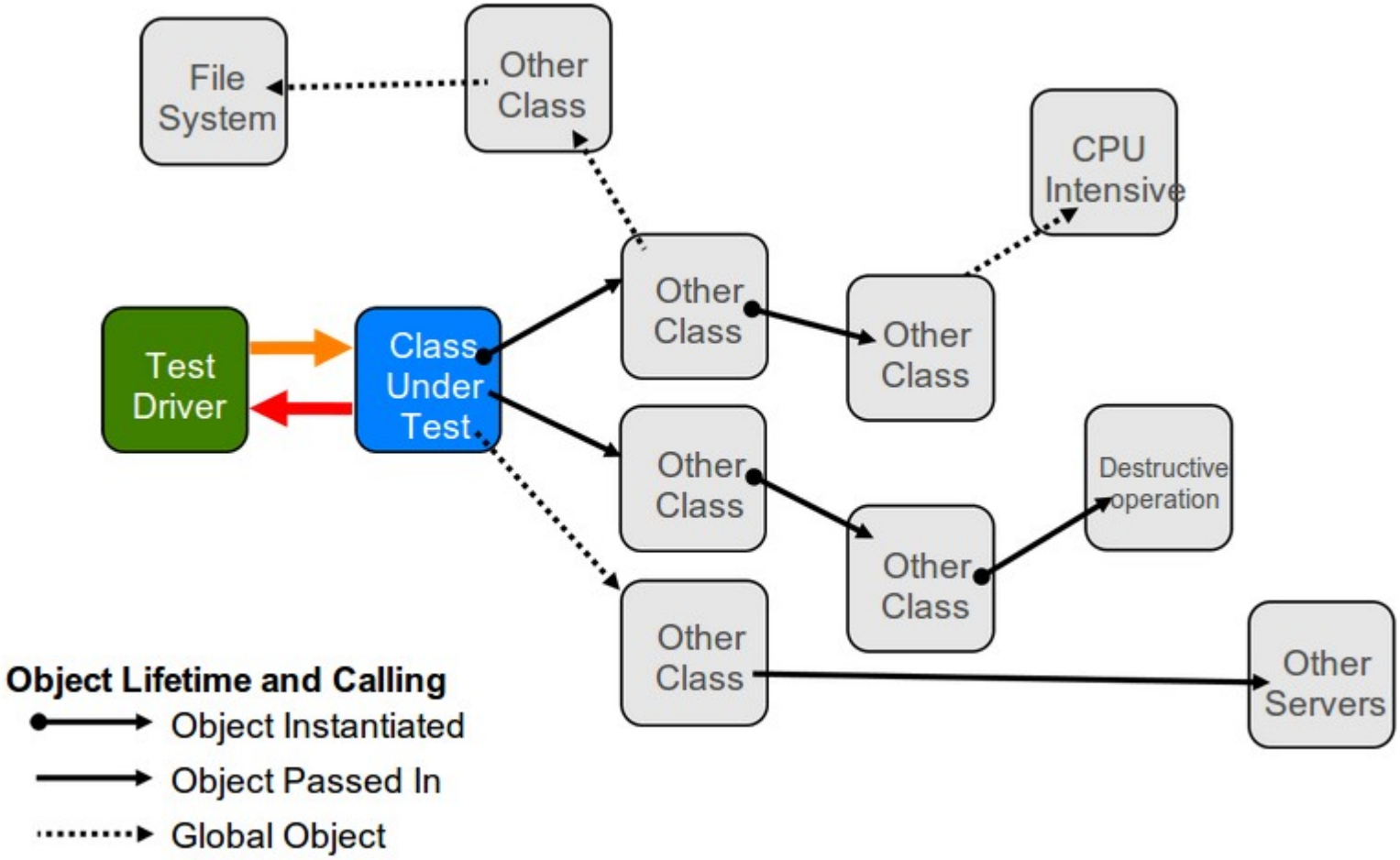
unit test – grafo de objetos



o famoso "new"

- pelo construtor
- parâm. de método
- injetado
- static
- InitialContext
- ServiceLocator
- entre outros...

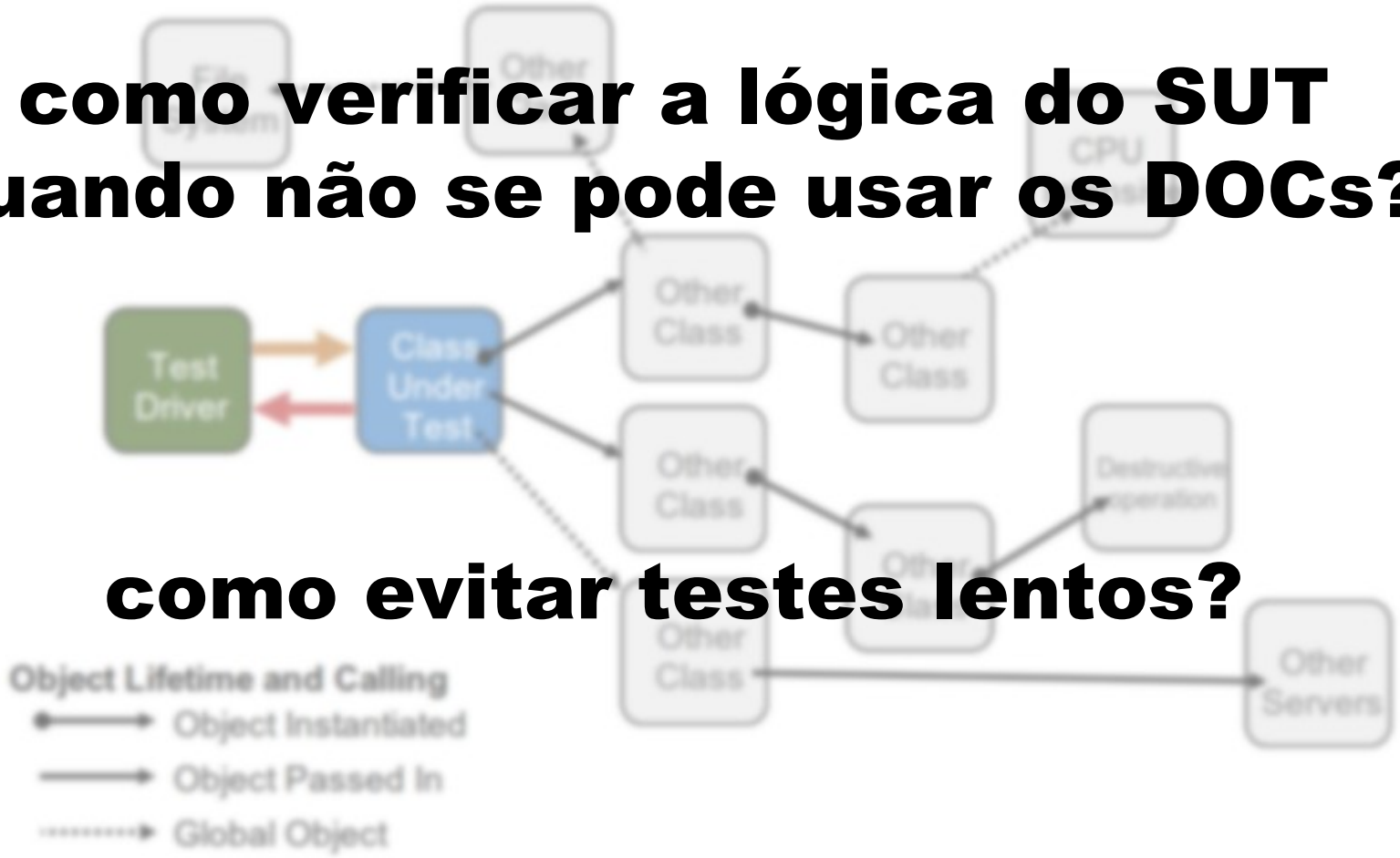
unit test – grafo de objetos



hevery – clean code talk (sl. 20-21)

unit test – grafo de objetos

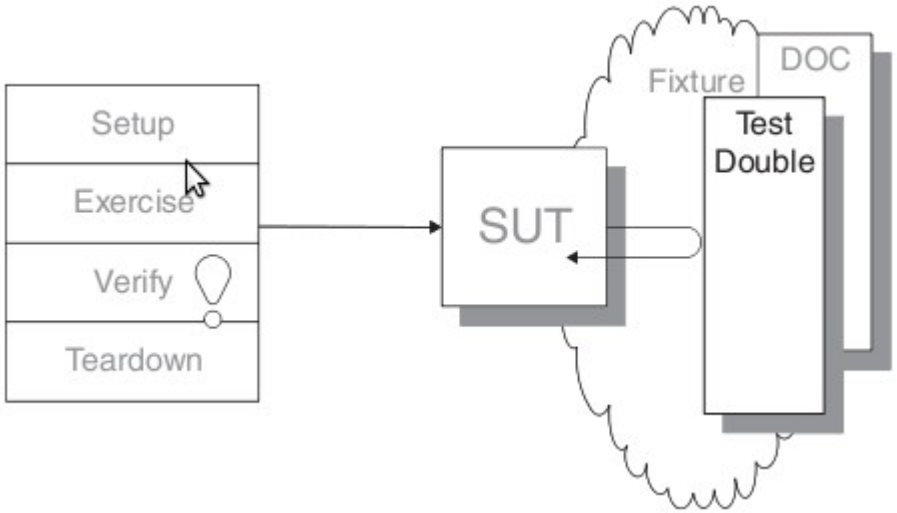
como verificar a lógica do SUT quando não se pode usar os DOCs?



como evitar testes lentos?

hevery – clean code talk (sl. 20-21) / xUnit (p. 522)

test pattern – test double

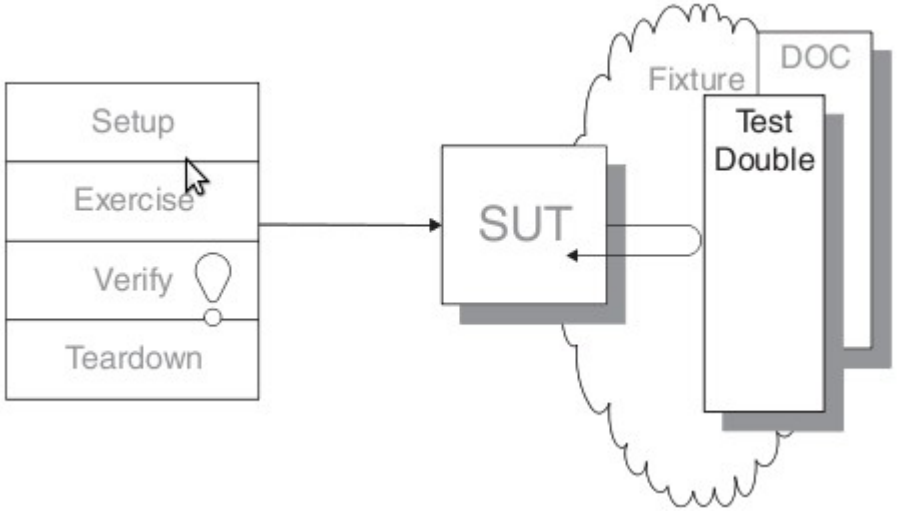


nós <u>trocamos</u> um componente que o SUT depende por um <u>equivalente</u> específico para o teste

xUnit (p. 522)

Touch TECNOLOGIA

test pattern – test double



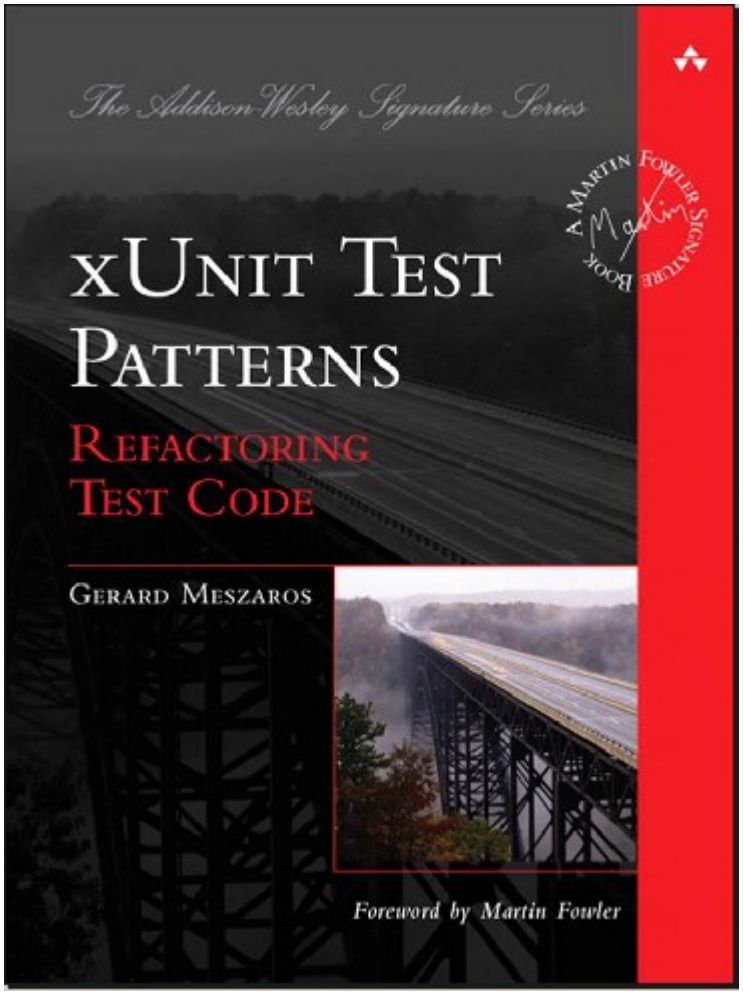
nós (2) trocamos um componente que o SUT depende por um (1) equivalente específico para o teste

xUnit (p. 522)

test pattern – test double

<digressão>

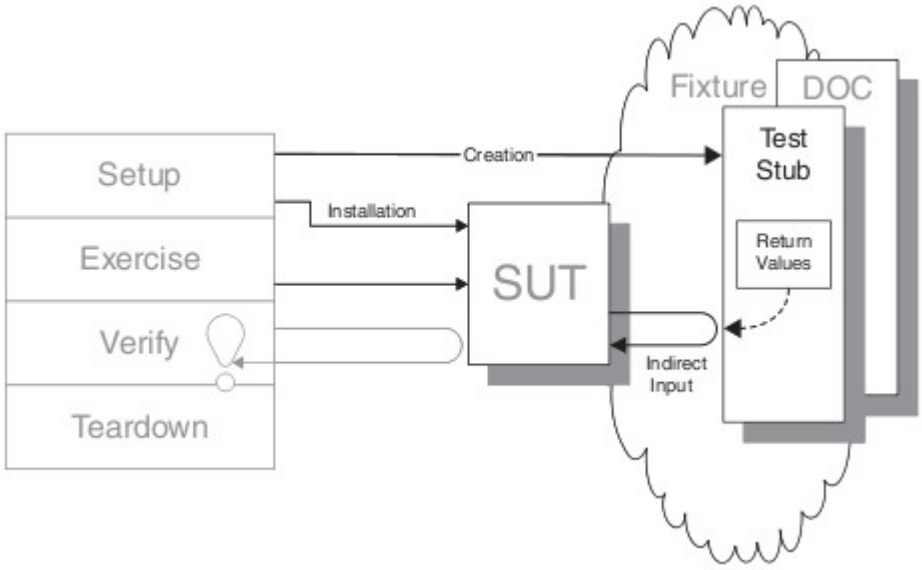
test pattern – xUnit



test pattern – test stub

flickr.com/photos/hedrok/366082663

test pattern – test stub



alimenta o SUT com entradas indiretas controladas

xUnit (p.529)

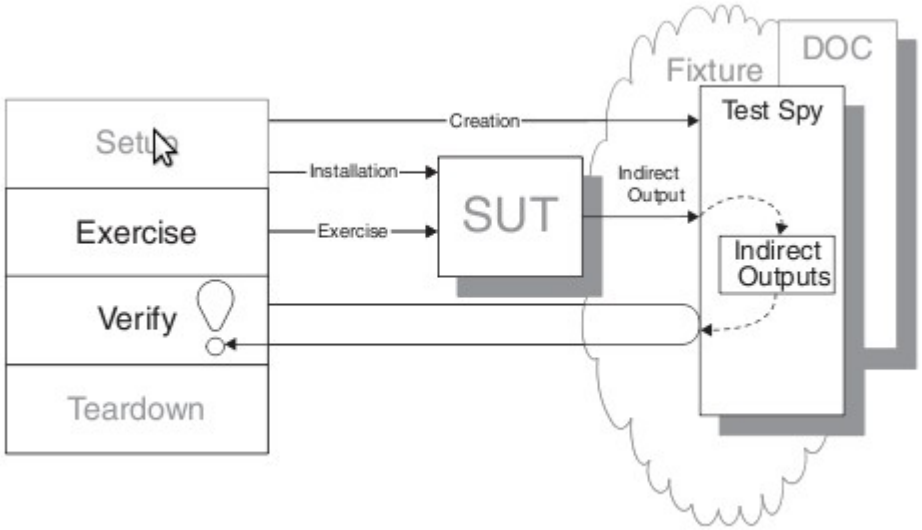
Touch TECNOLOGIA

test pattern – test spy

flickr.com/photos/jamespon/8896319/

Touch TECNOLOGIA

test pattern – test spy



captura saídas indiretas do SUT para verificação posterior

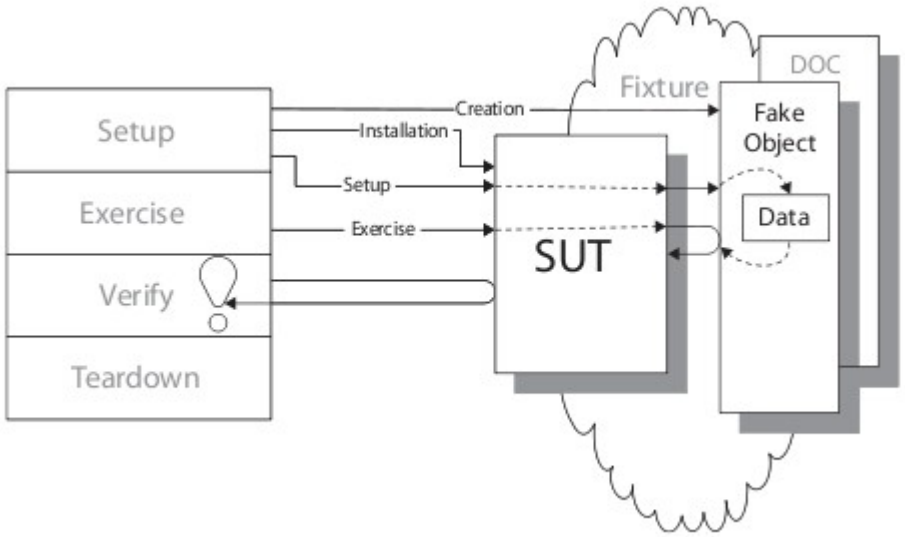
xUnit (p.538)

Touch TECNOLOGIA

test pattern – fake object

flickr.com/photos/sunshine6/339605220/

test pattern – fake object



versão light-weight do DOC

xUnit (p.551)

Touch TECNOLOGIA

test pattern – exemplos

- bd: hsqldb.org, h2database.com, db.apache.org/derby
- web: jetty.codehaus.org
- ejb: openejb.apache.org
- jndi: osjava.org/simple-jndi
- ftp: mockftpserver.sf.net
- smtp: mock-javamail.dev.java.net

test pattern – test double

</digressão>

xUnit (p.817)

mock object

flickr.com/photos/mtsofan/4125854488/

xUnit test pattern
X
bibliotecas mock

mock object - mockito

meet...

mockito - stubbing

```
// You can mock concrete classes, not only interfaces
LinkedList mockedList = mock(LinkedList.class);
// ^^^^^^^: bear with me for this ;)

// stubbing
when(mockedList.get(0)).thenReturn("first");
when(mockedList.get(1)).thenThrow(new RuntimeException());

// following prints "first"
System.out.println(mockedList.get(0));

// following throws runtime exception
System.out.println(mockedList.get(1));

// following prints "null" because get(999) was not stubbed
System.out.println(mockedList.get(999));
```

mockito - spying

```
//mock creation
List mockedList = mock(List.class);

//using mock object
mockedList.add("one");
mockedList.clear();

//verification
verify(mockedList).add("one");
verify(mockedList).clear();
```

mockito - faking

```
// you can create partial mock with spy() method:
List list = spy(new LinkedList());

// you can enable partial mock capabilities selectively on mocks:
MockitoTest mock = mock(MockitoTest.class);
// Be sure the real implementation is 'safe'.
// If real implementation throws exceptions or depends on specific
// state of the object then you're in trouble.
when(mock.someMethod()).thenCallRealMethod();
```

not recommended... use apenas para lidar com legado (mas legal de qualquer jeito ;-)

mockito.org

Touch TECNOLOGIA

mockito – tricks

```
import static org.mockito.Mockito.*;
```

DRY: Mockito.when(...) nunca mais!

```
doThrow(new RuntimeException()).when(list).add(0, "oi");
```

métodos voids tratados com elegância

```
@Mock private List<String> listMock;
```

receba seus mocks prontinhos

```
when(mockIterator.next())
.thenReturn("first")
.thenThrow(new RuntimeException());
```

encadeie stubbings em ordem

mockito.org

Touch TECNOLOGIA

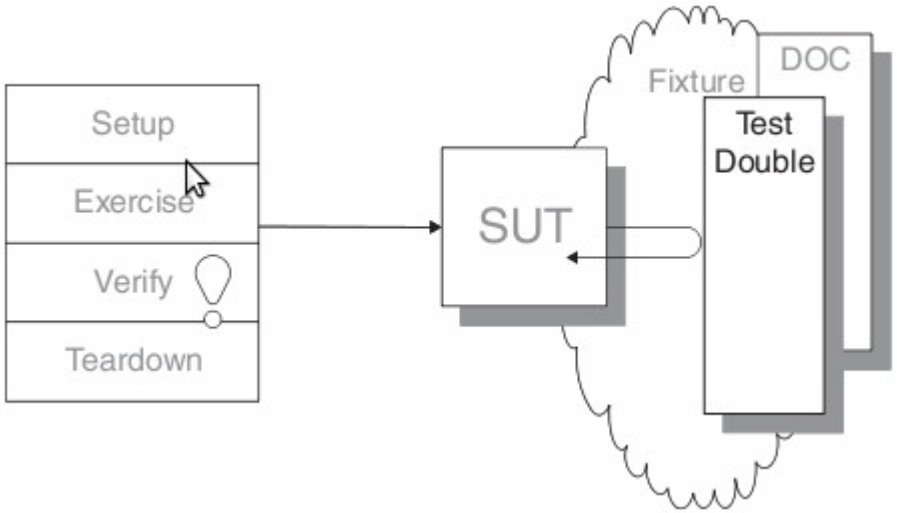
mockito – limitações

- precisa de java +1.5
- não é possível mockar classes final
- não é possível mockar métodos static
- não é possível mockar métodos final
- não é possível mockar equals() e hashCode()

test pattern – test double

<recapitulando>

test pattern – test double

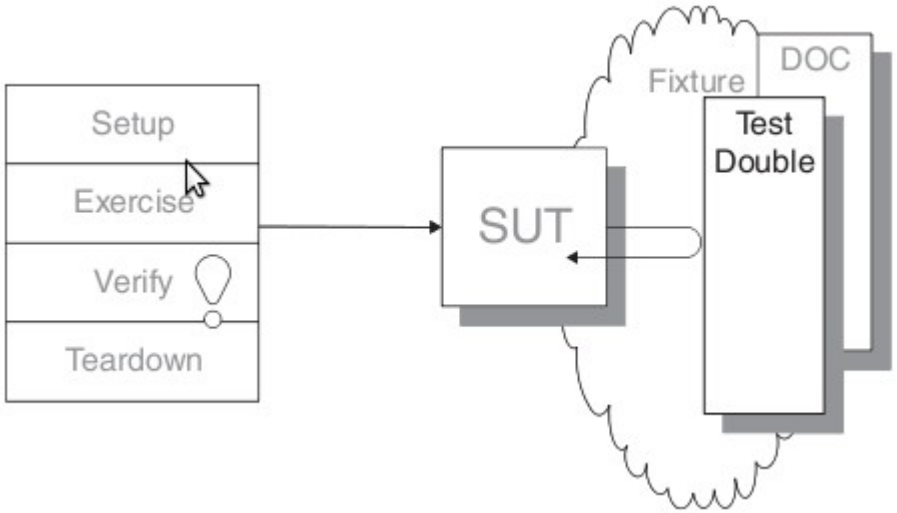


nós <u>trocamos</u> um componente que o SUT depende por um <u>equivalente</u> específico para o teste

xUnit (p. 522)

Touch TECNOLOGIA

test pattern – test double



nós (2) trocamos um componente que o SUT depende por um (1) equivalente ✔ específico para o teste

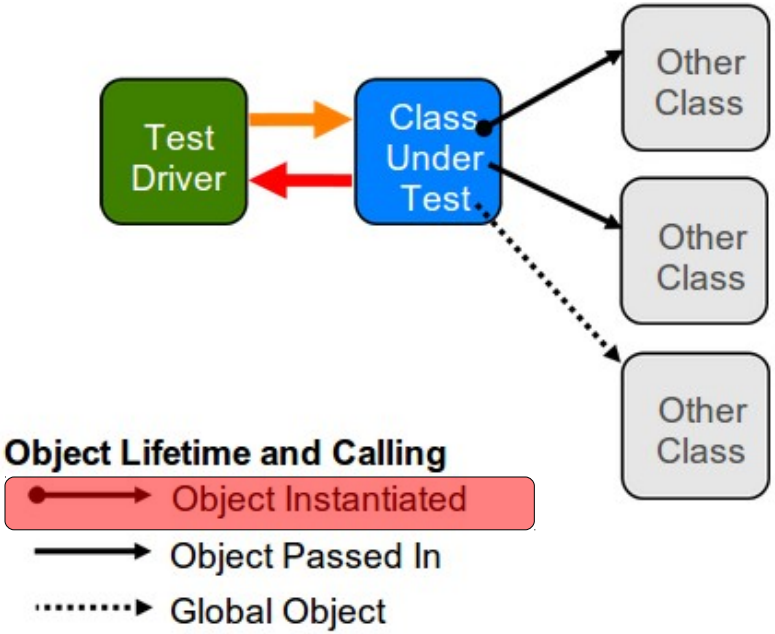
xUnit (p. 522)

test pattern – test double

</recapitulando>

de que forma podemos trocar um DOC?

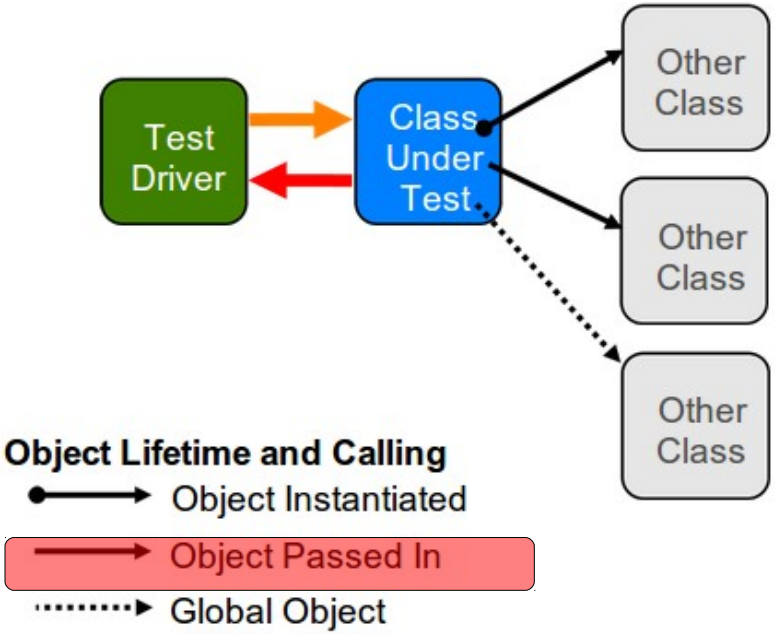
testabilidade – grafo de objetos



- **modificando classloading** ✗
- **refatorando SUT** ✓

hevery – clean code talk (sl. 20-21)

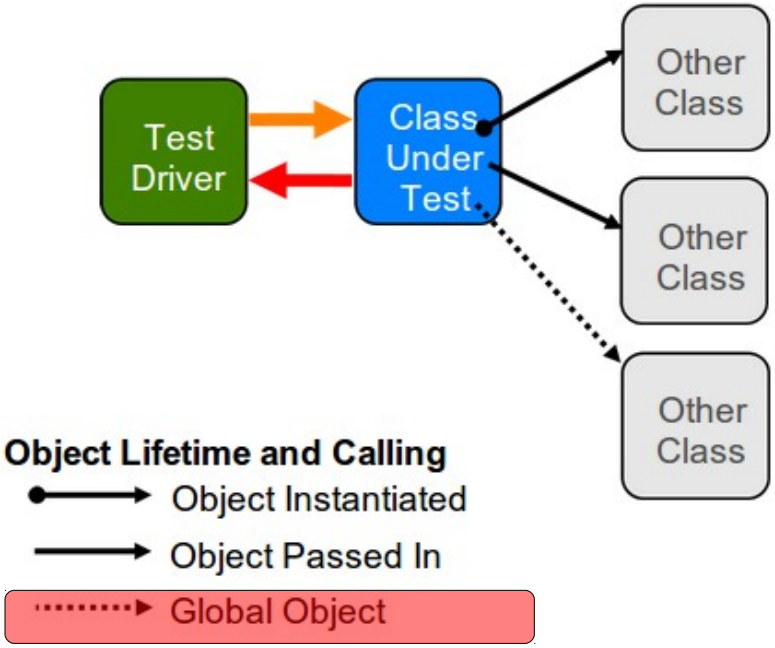
testabilidade – grafo de objetos



- **pelo construtor** ✔
- **como parâmetro de método** ✔
- **atribuindo diretamente ou via reflection** ✘

hevery – clean code talk (sl. 20-21)

testabilidade – grafo de objetos



- **substituindo variáveis estáticas usadas** ✗
- **necessidade de teardown normalmente**

hevery – clean code talk (sl. 20-21)

design para testabilidade

- **requisitos de teste influenciam design de código**
- **em sintonia com boas práticas arquiteturais:**
  - **injeção de dependências**
  - **law of demeter**

- `if(TDD) { dontWorry(); }`

xUnit (p.792)

Touch TECNOLOGIA

testabilidade - testability-explorer

meet...

code.google.com/p/testability-explorer

Touch TECNOLOGIA

testability-explorer – how it works?

- **non-mockable total recursive cyclometic complexity**

lê-se: o que você não consegue trocar, você precisa testar junto...

- **global mutable state**

... e se outros usam, limpe sua parte ;)

testability-explorer

<demo/>

muito obrigado!

contato:
recrutamento@touchtec.com.br

www.touchtec.com.br

fone: 3846-5555