

Como testar unitariamente em PHP

Tales Mota Machado¹²

¹Universidade Federal de Ouro Preto

²Analista de Sistemas -GerenciaNet-

Uma visão geral sobre teste unitários.

Outline

- 1 O que é qualidade?**
 - O que é qualidade?
 - O que é um código de qualidade?
 - Ausência de bugs
 - Por que testar?
- 2 Testes Unitários**
 - xUnit
 - PHPUnit
- 3 Conclusão**

O que é qualidade?

QUALIDADE

O dicionário Michaelis define qualidade como: “2 Excelência, virtude, talento.”, “4 Grau de perfeição, de precisão, de conformidade a um certo padrão.”

O que é um código de qualidade?

O que é um código de qualidade?

“Qualidade depende do ponto de vista!”[2]

- Do usuário: satisfação de requisitos, ausência de bugs, frequência de releases, bom desempenho, etc.
- Do desenvolvedor: facilidade de entender o código, flexibilidade para modificar o código, abrangência da suíte de testes, etc.

Ausência de bugs

Ausência de bugs

É possível escrever um código livre de bug?

Ausência de bugs

o que é um bug?

O que é um bug?

Impossível

É impossível[1]. Por mais simples que seja o código, não há como garantir que não haverá bugs devido às várias dependências:

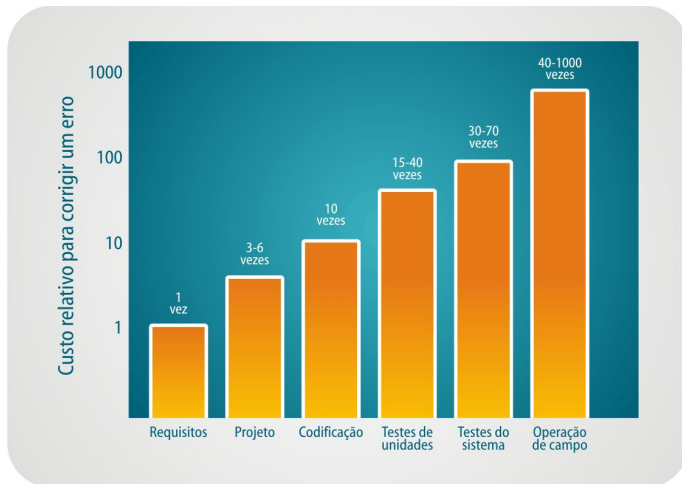
- Compilador/interpretador
- SO
- Usuário final

Toda via o correto seria dizer minimizar a ocorrência de bugs.

“Os teste podem somente mostrar a presença de erros, não sua ausência.” **Dijkstra, et al., 1972**

Se o teste não garante a ausência de bugs, por que então perder tempo escrevendo testes?

Por que testar?

**Figure:** Regra 10 Myers

Os teste unitários podem remover entre 30% e 50% dos defeitos dos programas.[4]

Existem vários framework's para automatizar os testes de unidade. Os mais usados no mercado sem dúvida são os xUnit, abrangindo a quase todas linguagens. Para PHP temos:

- SimpleTest

Existem vários framework's para automatizar os testes de unidade. Os mais usados no mercado sem dúvida são os xUnit, abrangendo a quase todas linguagens. Para PHP temos:

- SimpleTest
- **PHPUnit**

xUnit

SimpleTest I

simpletest “<http://www.simpletest.org/>” usa a mesma API do xUnit:

| | |
|-----------------------------|---|
| assertTrue (\$ x) | Falha se \$ x é falsa |
| assertFalse (\$ x) | Falha se \$ x é verdadeiro |
| assertNull (\$ x) | Falha se \$ x é definido |
| assertNotNull (\$ x) | Falhar se não definir \$ x |
| assertIsA (\$ x, \$ t) | Falha se \$ x não é o tipo ou classe \$ t |
| assertNotA (\$ x, \$ t) | Falha se \$ x é da classe ou tipo de \$ t |
| assertEquals (\$ x, \$ y) | Falha se \$ x == \$ y é falso |
| assertNotEqual (\$ x, \$ y) | Falha se \$ x == \$ y é verdadeiro |

xUnit

SimpleTest II

| | |
|---|---|
| <code>assertWithinMargin (\$ x, \$ y, \$ m)</code> | Falhar se $\text{abs}(\$ x - \$ y) < \$ m$ é falsa |
| <code>assertOutsideMargin (\$ x, \$ y, \$ m)</code> | Falhar se $\text{abs}(\$ x - \$ y) < \$ m$ é verdade |
| <code>assertIdentical (\$ x, \$ y)</code> | Falhar se $y \$ x == \$$ é falsa ou uma incompatibilidade de tipo |
| <code>assertNotIdentical (\$ x, \$ y)</code> | Falhar se $y \$ x == \$$ é verdadeiro e tipos de correspondência |
| <code>assertReference (\$ x, \$ y)</code> | Falhar a menos que $\$$ e $\$ x y$ são a mesma variável |
| <code>assertClone (\$ x, \$ y)</code> | Falhar a menos que $\$$ e $\$ x y$ são cópias idênticas |

xUnit

SimpleTest III

| | |
|------------------------------|---|
| assertPattern (\$ p, \$ x) | Falhar a menos que a regex \$ p \$ x corresponde |
| assertNoPattern (\$ p, \$ x) | Falhar se a regex \$ p \$ x corresponde |
| expectError (\$ x) | Falhar se erro de correspondência não occur |
| expectException (\$ x) | Falha se a exceção de correspondência não é lançada |
| ignoreException (\$ x) | Engole qualquer exceção próximo correspondência |
| assert (\$ e) | Falha na falhou expectativa objeto \$ e |

É o framework oficial seguindo o padrão xUnit. Sua api está disponível em “<http://www.phpunit.de/manual/current/en/>”

PHPUnit

Mock

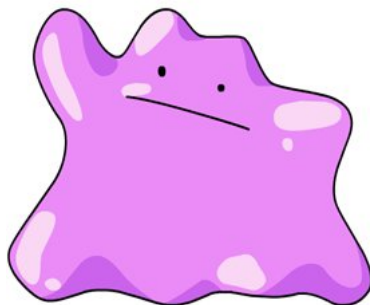


Figure: Mock1

PHPUnit

Mock



Figure: Mock2

E então...?

Vale a pena testar?

E então...?

Vale a pena testar? Sim

E então...?

O tempo gasto escrevendo teste diminui a produtividade do profissional?

E então...?

O tempo gasto escrevendo teste diminui a produtividade do profissional? Não

E então...?

Pense na regra de Myers, vale a pena sacrificar OO?

E então...?

Pense na regra de Myers, vale a pena sacrificar OO? **Nunca será...**



Figure: Nuna

A idéia do framework xUnit, e de todo o movimento de teste relacionado ao desenvolvimento de software foi de **Kent Beck**. A primeira versão do xUnit foi o SUnit para **smalltalk**.

Essa idéia tão simples e fantástica atriu **Erich Gamma** que junto com **Kent Beck** criaram o JUnit, que consolidou a prática.



Engenharia de software Ian Sommerville , 2011, 9ª Edição



Qualidade de Código: mantendo seu projeto de software sob controle 2009, Antonio Terceiro ,
“<http://softwarelivre.org/terceiro/blog/qualidade-de-codigo-mantendo-seu-projeto-de-software-sob-controle>”



Prática: melhore a qualidade do código para evitar uma enchente de bugs 2010, Guilherme Silveira, Caelum ,
“<http://blog.caelum.com.br/pratica-melhore-a-qualidade-do-codigo-para-evitar-uma-enchente-de-bugs/>”



Base de conhecimento em teste de software 2 edição,
Anderson Bastos, Emerson Rios, Ricardo Cristalli, Trayahú Moreita