Curso DevOps

Aula 08 - Testes Unitários de Software

Prof. Esp. Guilherme Jorge Aragão da Cruz


guilherme.cruz@alumni.usp.br


linkedin.com/in/guijac

Roteiro

- Testes de Software;
- Verificação e Validação;
- A Pirâmide de Testes;
- Regra 10 de Myers;
- Testes Unitários de Software:
  - Definição;
  - Importância.
- A Família xUnit:
  - Definição;
  - Principais Características;
  - Métodos Assert.
- Testes Unitários com PyUnit:
  - Definição;
  - Exemplo.
- Referências Bibliográficas.

Testar?

Testar?

> *Processo sistemático que tem por objetivo **encontrar defeitos**.*

MYERS (1979)

> *Verificar se o software está **fazendo** o que **deveria fazer**, de acordo com seus **requisitos**, e se **não está fazendo** o que **não deveria fazer**.*

RIOS E MOREIRA (2006)

Fonte: Testes de Software: quais os tipos e por que implementar? - Firework

Verificar e Validar?

Verificar e Validar?

- Não são a mesma coisa!

> ***Verificação*** *refere-se ao conjunto de tarefas que garantem que o software* ***implemente corretamente*** *uma função específica;*
>
> ***Validação*** *refere-se ao conjunto de tarefas que asseguram que o software foi* ***criado*** *e pode ser rastreado segundo os* ***requisitos do cliente****.*

PRESSMAN (2021)

> ***Verificação****: Estamos criando o produto corretamente?*
>
> ***Validação****: Estamos criando o produto certo?*

BOEHM (1981)

Regra de 10 de Myers

- Glenford Myers, em seu livro “The Art of Software Testing” (1979), lança a Teoria da Regra de 10: quanto **mais cedo** descobrimos e corrigimos o **erro**, **menor é o custo** para o projeto.

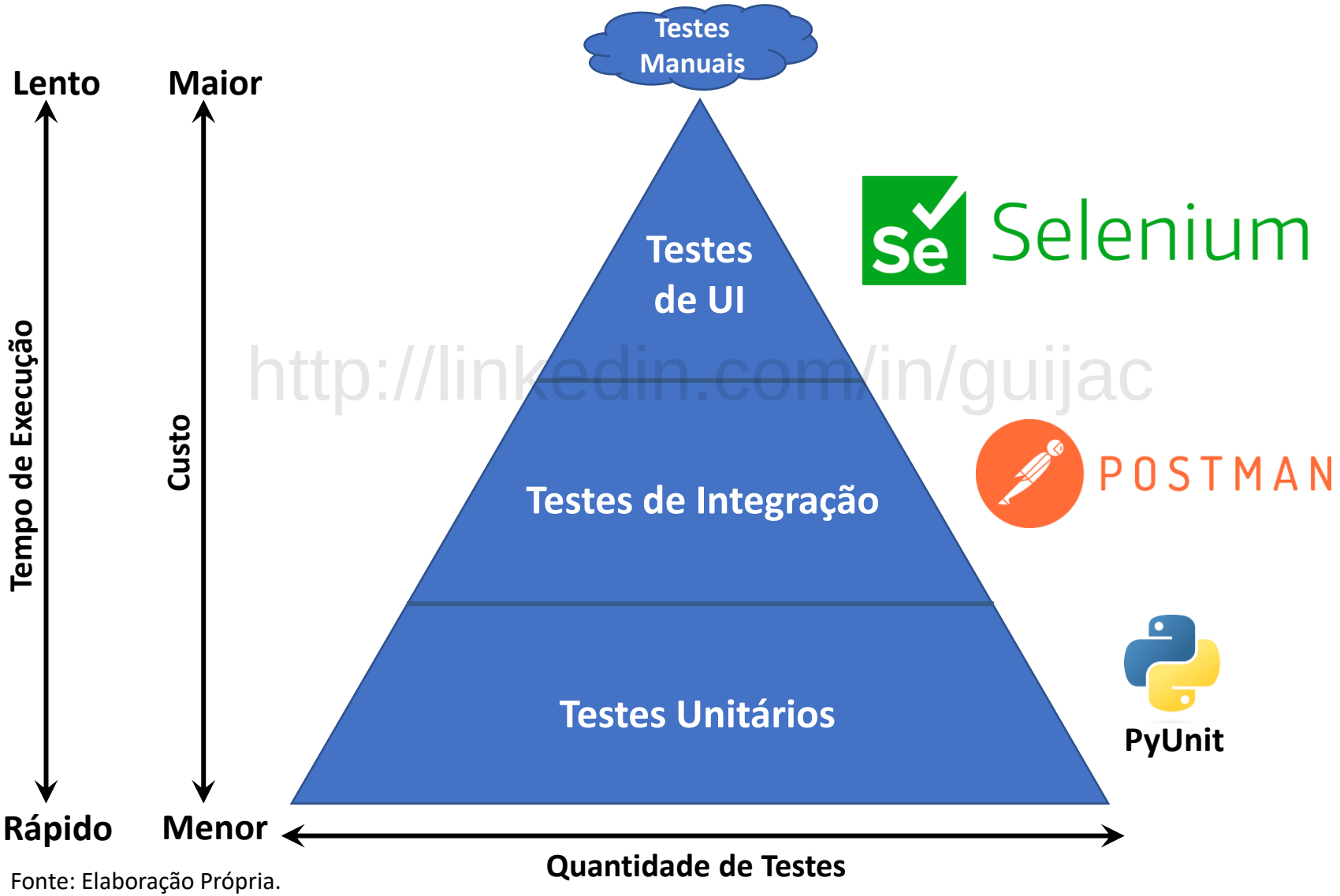
Fonte: Elaboração Própria.

Regra de 10 de Myers

- Glenford Myers, em seu livro “The Art of Software Testing” (1979), lança a Teoria da Regra de 10: quanto **mais cedo** descobrimos e corrigimos o **erro**, **menor é o custo** para o projeto.

Fonte: Elaboração Própria.

A Pirâmide de Testes

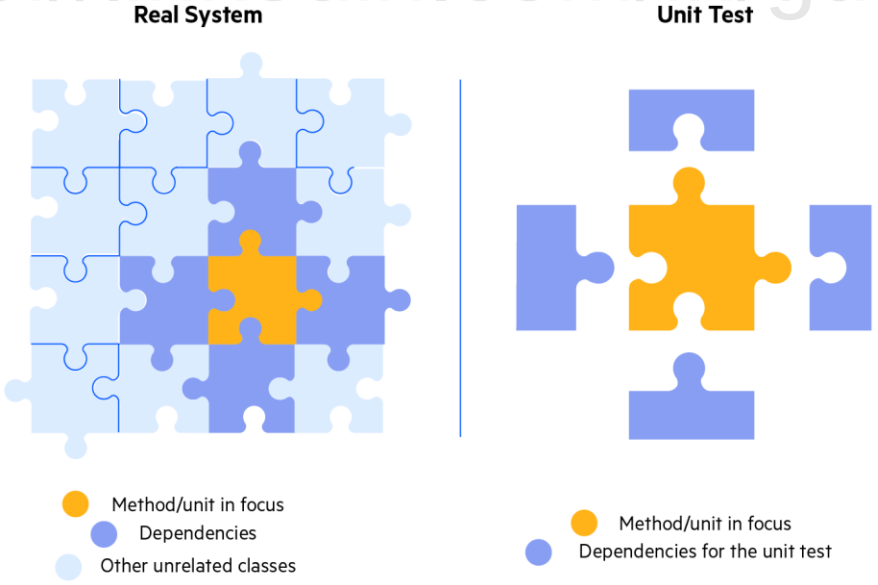


Fonte: Elaboração Própria.

Testes Unitários de Software: Definição

> *Testes de baixo nível, concentrando-se em **pequenas partes** de uma aplicação;*
> ***Escritos** no **dia a dia** pela própria **equipe de desenvolvimento**, utilizando ferramentas da família "**xUnit**";*
> *Significativamente **mais rápidos** que outras abordagens de testes.*

FOWLER (2014)



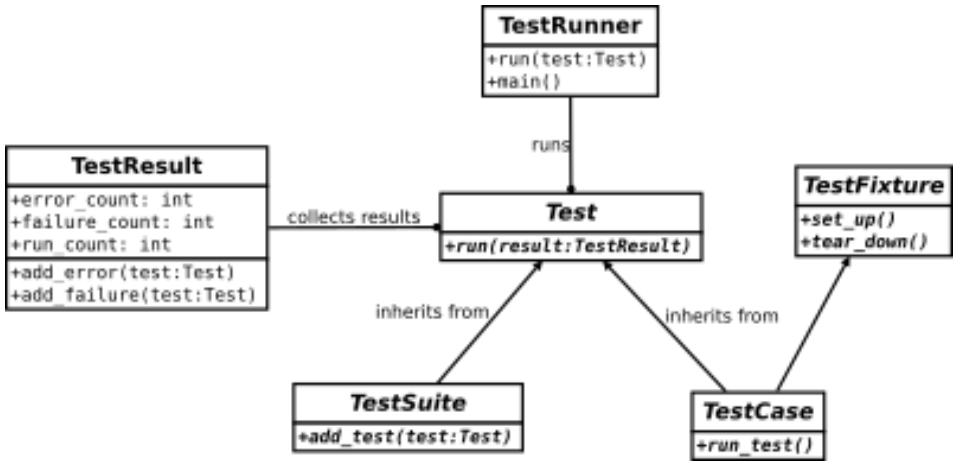
Fonte: Top 9 Reasons To Unit Test Your C# Code (telerik.com)

Testes Unitários de Software: Importância

- Testes unitários bem escritos **antecipam a identificação de erros e bugs**;
- A identificação de erros e bugs antes de colocar uma aplicação em uso (produção) possibilitam uma **redução de custos**;
- Ao desenvolver de forma orientada a testes criamos métodos com responsabilidades bem definidas e que são fáceis de testar, **aumentando a qualidade do código e facilidade de manutenção**;
- Um teste unitário não garante apenas que uma unidade de software esteja funcionando, ele **garante que uma funcionalidade permanece funcionando após alguma alteração de código**.

xUnit: Definição

- Nome coletivo para as diversas estruturas de testes de unidade derivadas do “SUnit”, da linguagem Smalltalk, projetada por Kent Beck[1] em 1998;
- Por sua característica orientada a objetos hoje é utilizada em diversas linguagens, substituindo o “S” pela primeira letra (ou letras) da linguagem em uso (“JUnit“ para Java e “PyUnit” para Python, por exemplo).

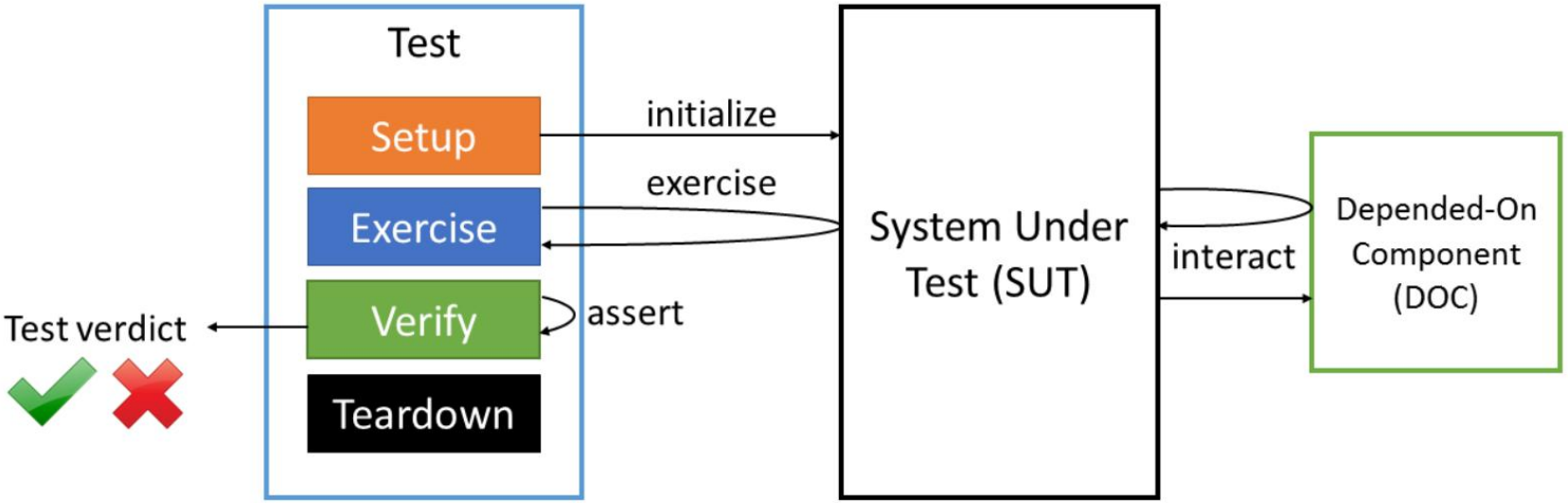


Fonte: Enter the Panopticon – Towards a Test Driven Development Framework

[1] engenheiro de software estadunidense criador do Extreme Programming (xP) e Test Driven Development (TDD). Foi um dos 17 signatários originais do Agile Manifesto em 2001.

xUnit: Principais Características

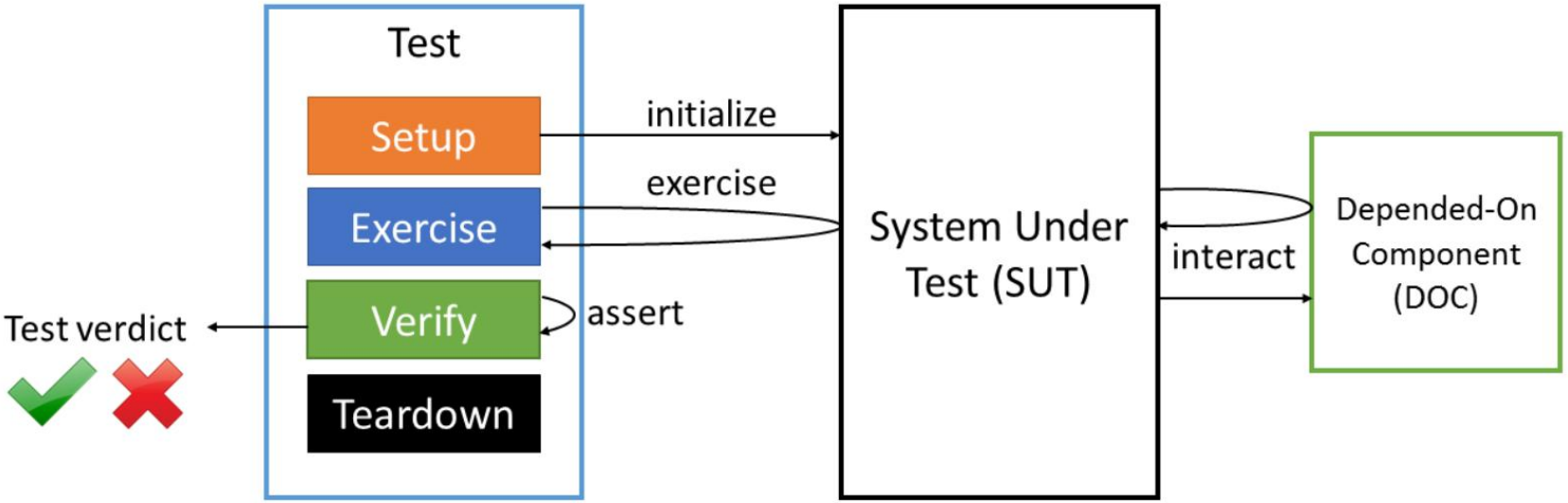
- **System Under Test (SUT):** definido a partir da perspectiva do que está sendo testado, quando estamos escrevendo testes de unidade, o sistema em teste (SUT) é qualquer classe, objeto ou método(s);
- **Dependend-On Component (DOC):** componente no qual o SUT possui dependência, normalmente possuindo a mesma granularidade: por exemplo se SUT é uma classe, então o mesmo depende outras classes.



Fonte: Software testing | Mastering Software Testing with JUnit 5 (packtpub.com)

xUnit: Principais Características

- **Setup**: permite a configuração de pré-condições para a execução de um teste, ideal para instanciar objetos com valores predefinidos;
- **Exercise**: realiza a interação com o SUT, obtendo assim, algum resultado a ser validado;
- **Verify**: realiza a execução dos **métodos assert**, explicado a seguir;
- **Teardown**: executado ao final de cada teste, ideal para fechar conexões ao banco de dados, arquivos, etc.



Fonte: Software testing | Mastering Software Testing with JUnit 5 (packtpub.com)

xUnit: Principais Características

- Os métodos assert (asserção) são responsáveis por verificar se o resultado gerado durante um teste é igual ao esperado, caso contrário o teste resultará em uma falha e exibirá uma mensagem especificada;
- Uma especificação completa está disponível na documentação da biblioteca unittest (em Python).


```
# sintaxe:
assertEqual(“valor esperado", valor_obtido, "Deu Ruim no teste!")

assertEqual(a, b)     # a == b
assertNotEqual(a, b)  # a != b
assertTrue(x)    # bool(x) is True
assertFalse(x)   # bool(x) is False
```

Principais métodos assert utilizados.

PyUnit: Definição

- Framework para construção de Testes Unitários com Python, baseado na família “xUnit”;
- Presente por padrão no Python desde a versão 2.1 como biblioteca chamada “unittest”, não sendo necessária nenhuma instalação adicional.


```python
import unittest

class TestClass(unittest.TestCase):

def test_meu_metodo(self):
    self.assertEqual(valor_esperado , valor_real, "mensagem caso o teste falhe")

if __name__ == "__main__":
    unittest.main()
```

Uso do PyUnit em uma classe de testes.
Fonte: DEVMEDIA (2020)

PyUnit: Exemplo

```python
import unittest
from app.app import app

class AppTest(unittest.TestCase):

    def setUp(self):
        self.app = app.test_client()

    def test_print_hello_success(self):
        response = self.app.get('/hello?name=guijac')
        self.assertEqual("Hello, guijac!", response.get_data(as_text=True)
                         , "Deu Ruim no test_print_hello_success!")

    def test_print_hello_error(self):
        response = self.app.get('/hello')
        self.assertEqual(400, response.status_code, "Deu Ruim no test_print_hello_error!")

if __name__ == "__main__":
    unittest.main()
```

Exemplo de uma classe para execução de testes unitários com PyUnit. Notem que o método setup recebe o sistema em teste (SUT) e permite a instanciação de um Client Flask (app) para chamada dos seus métodos (como o “GET”, por exemplo).

Referências Bibliográficas

Boehm, B., **Software Engineering Economics.** Prentice-Hall, 1981.

DEVMEDIA. Teste unitário com PyUnit. Disponível em https://www.devmedia.com.br/teste-unitario-com-pyunit/41233. Acesso em 10 mar 2023;

KACZANOWSKI, Tomek. **Practical Unit Testing with TestNG and Mockito**. Tomasz Kaczanowski, 2012. Disponível em https://kaczanowscy.pl/tomek/category/books/practical-unit-testing. Acesso em 25 mai 2023;

MYERS, Glenford J. **The art of software testing**. New York: John Wiley & Sons, 1979.

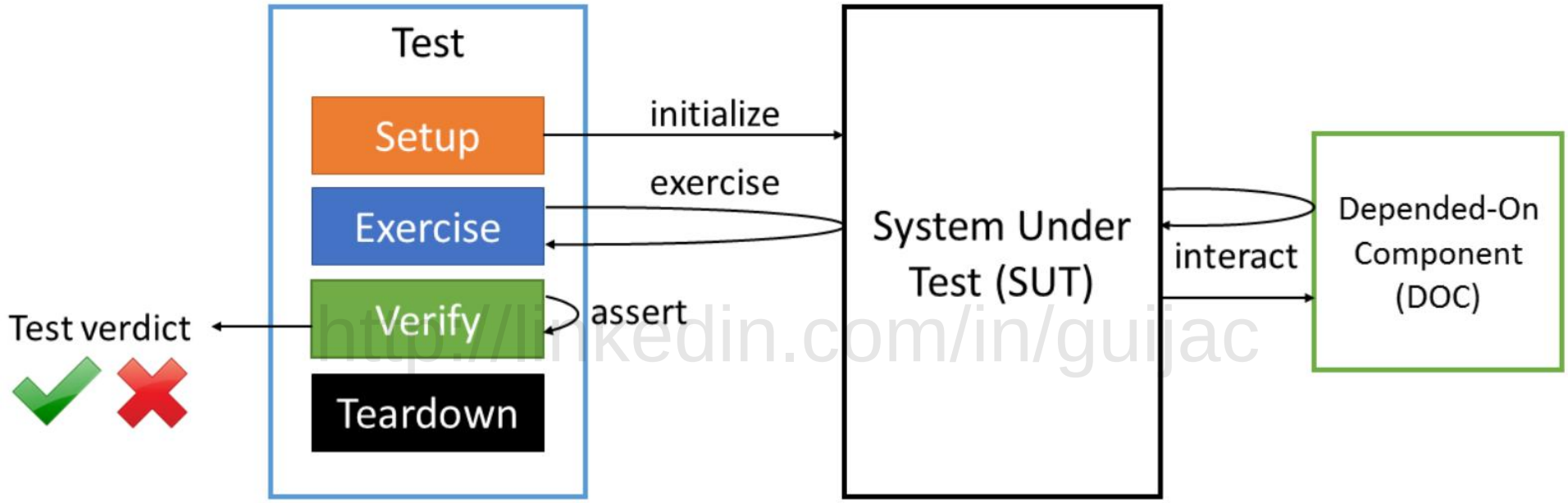
FOWLER, Martin. **UnitTest**. Disponível em https://martinfowler.com/bliki/UnitTest.html. Acesso em 10 mar 2023;

PRESSMAN, Roger S.; MAXIM, Bruce R. **Engenharia de software-9**. McGraw Hill Brasil, 2021.

RIOS, Emerson; MOREIRA, Trayahú. **Teste de Software**. 2.ed. São Paulo: Alta Book, 2006.

XUNITPATTERNS. **xUnit Test Patterns - the book**. Disponível em http://xunitpatterns.com/. Acesso em 25 mai 2023.

Por hoje (de teoria!) é só!



Fonte: Software testing | Mastering Software Testing with JUnit 5 (packtpub.com)

Prof. Esp. Guilherme Jorge Aragão da Cruz

guilherme.cruz@alumni.usp.br

linkedin.com/in/guijac

Licença

- Este conteúdo está licenciado sob a Licença Creative Commons Atribuição-NãoComercial-CompartilhaIgual 4.0 Internacional (CC BY-NC-SA 4.0).
- Todos os direitos autorais sobre este conteúdo pertencem ao autor, e este material não pode ser usado comercialmente sem autorização expressa.
- Para ver o texto completo da licença, acesse o https://creativecommons.org/licenses/by-nc-sa/4.0/legalcode.

Prof. Esp. Guilherme Jorge Aragão da Cruz

guilherme.cruz@alumni.usp.br

linkedin.com/in/guijac