# Fundamentos de 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 JUnit:
  - 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.

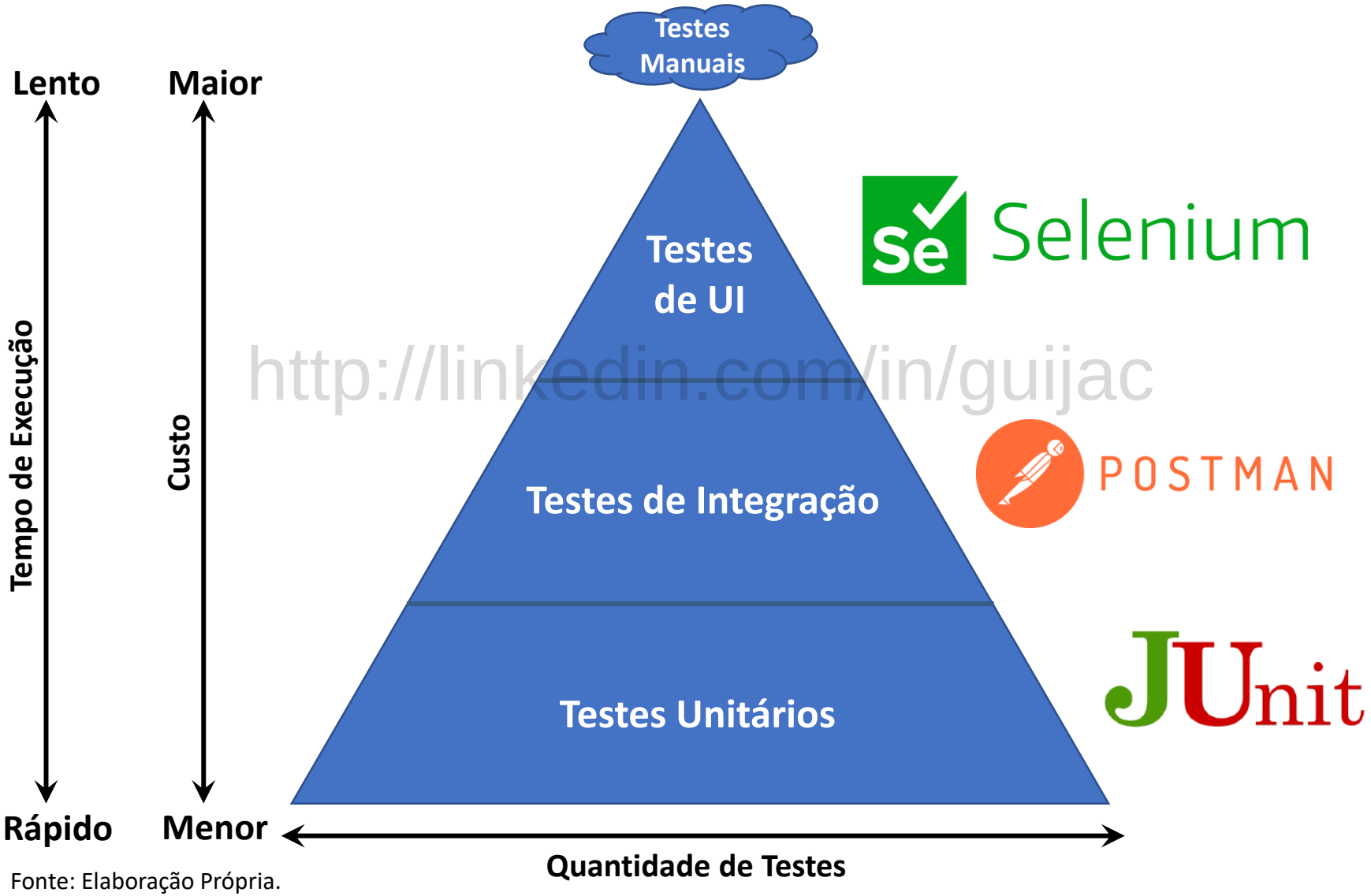
# Regra de 10 de Myers

- 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



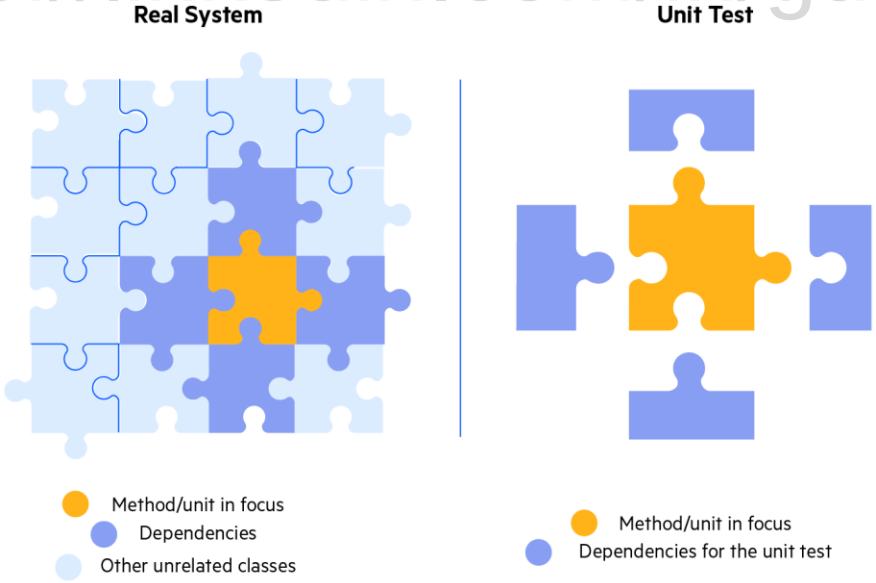

Fonte: Elaboração Própria.

# 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)

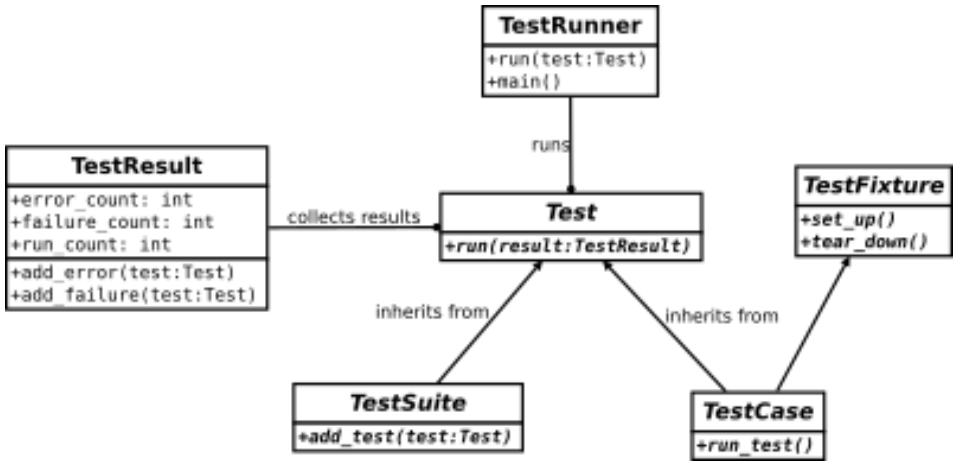
# Testes Unitários de Software: Importância

- Testes unitários, quando 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).





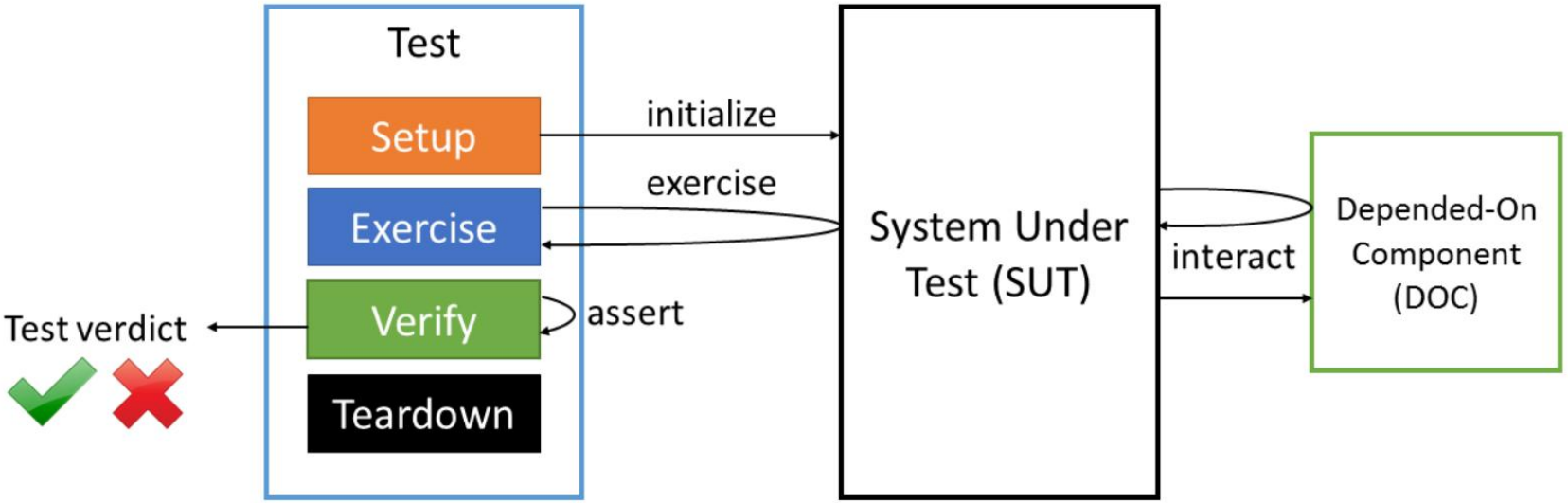

Fonte: Enter the Panopticon – Towards a Test Driven Development Framework

[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):** é o que estamos testando — pode ser uma classe, um objeto ou um método específico;
- **Dependend-On Component (DOC):** é o componente do qual o SUT depende. Se o SUT é uma classe, normalmente o DOC são outras classes relacionadas.



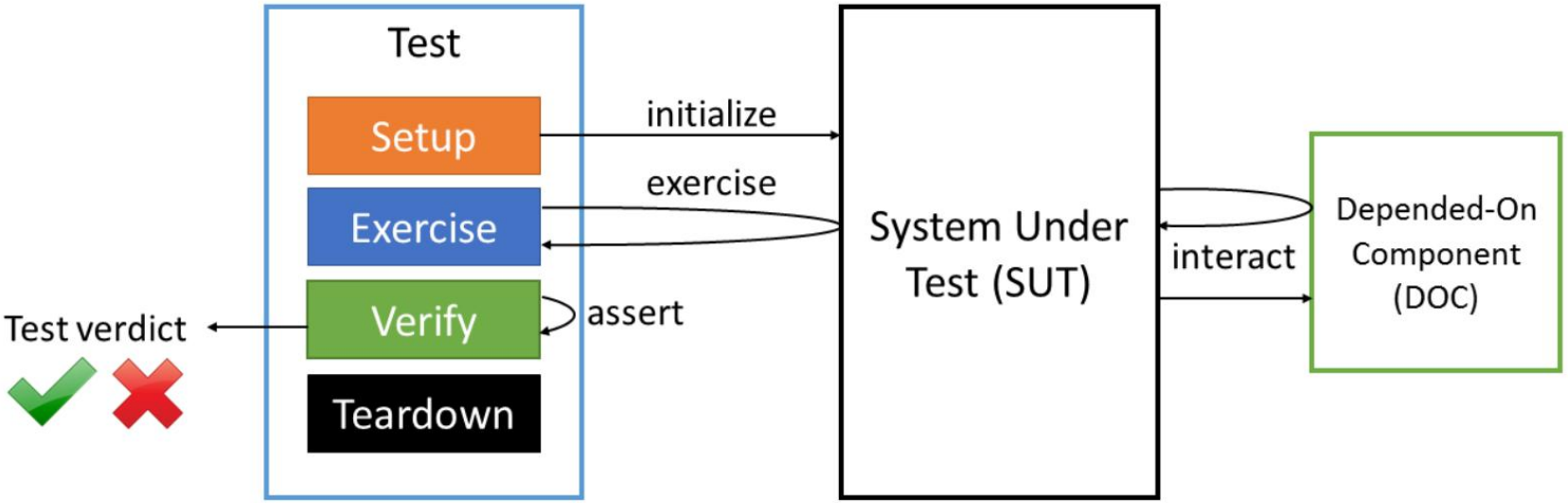



Fonte: Software testing | Mastering Software Testing with JUnit 5 (packtpub.com)

# 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 classe Assertions (JUnit 5.0.1 API).



```
import static org.junit.jupiter.api.Assertions.*;

assertEquals(a, b);       // a == b
assertNotEquals(a, b);    // a != b
assertTrue(x);            // x é verdadeiro
assertFalse(x);           // x é falso
```

Principais métodos assert utilizados.

# JUnit: Definição

- Framework para construção de Testes Unitários com Java, baseado na família “xUnit”;
- Necessita ser adicionado como dependência, mas é facilmente configurado via **Maven Central** ou **Gradle**;
- Uma boa prática é nomear os testes no formato **Given/When/Then (Dado/Quando/Então)**, descrevendo o cenário, a ação e o resultado esperado.



```java
import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;

class CalculadoraTest {
    @Test
    @DisplayName("Deve retornar 2 quando somar 1 + 1")
    void shouldReturnTwo_whenAddingOnePlusOne() {
        // Given (pré-condição)
        int expected = 2;
        // When (ação)
        int actual = 1 + 1;
        // Then (validação)
        assertEquals(expected, actual, "O teste só passa quando 1 + 1 for 2");
    }
}
```

Uso do JUnit em uma classe de testes.

# JUnit: Exemplo

```java
@SpringBootTest(webEnvironment = RANDOM_PORT)
class HelloControllerTest {

    @Value("${local.server.port}")
    private int port;
    private String baseUrl;
    private final TestRestTemplate restTemplate = new TestRestTemplate();

    @BeforeEach
    void setUp() {
        this.baseUrl = "http://localhost:" + port;
    }

    @Test
    @DisplayName("Deve retornar mensagem Alive ao chamar /health-check")
    void shouldReturnAliveMessage_whenCallingHealthCheck() {
        ResponseEntity<String> response =
                restTemplate.getForEntity(baseUrl + "/health-check", String.class);

        assertEquals(HttpStatus.OK, response.getStatusCode(),
                "O endpoint /health-check deveria responder 200 OK");
        assertEquals("<h1>Hello, I'm Alive!</h1>", response.getBody(),
                "O corpo da resposta não corresponde à mensagem esperada");
    }
}
```

Exemplo de classe para execução de testes unitários com JUnit 5. O método **@BeforeEach** prepara o Sistema em Teste (SUT) e permite a configuração de um cliente (**TestRestTemplate**) para simular chamadas HTTP, como um GET, nos *endpoints* da aplicação Spring Boot.

# Referências Bibliográficas

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

DEVMEDIA. **JUnit: Implementando testes unitários em Java**. Disponível em https://www.devmedia.com.br/junit-tutorial/1432. Acesso em 09 set 2025;

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



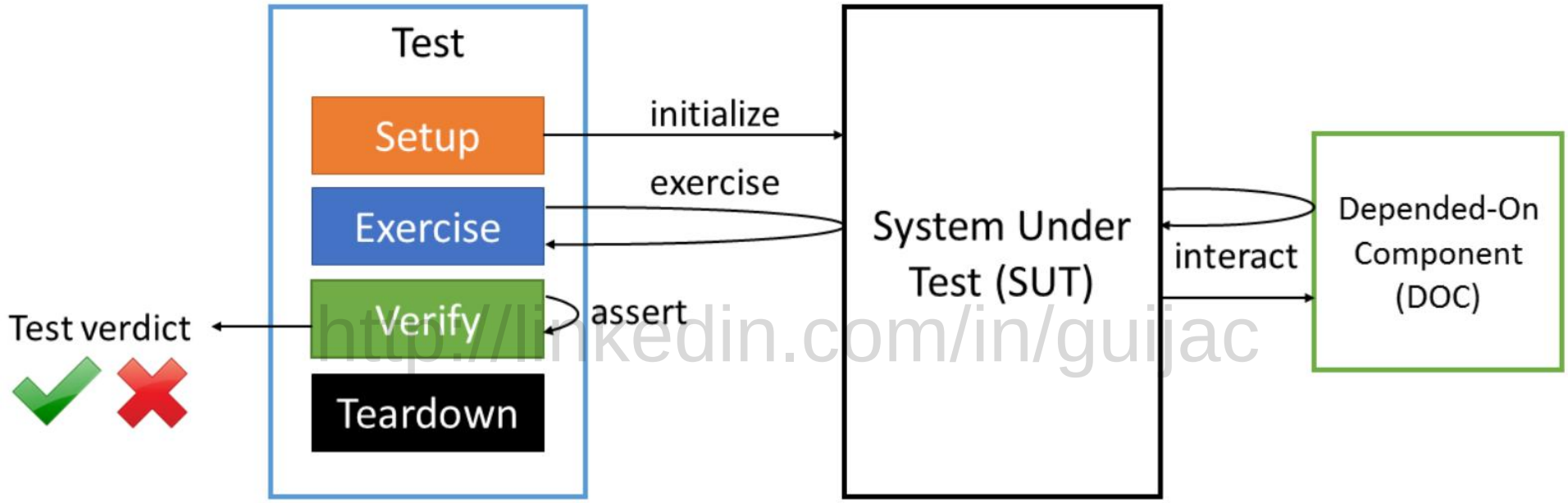
FOWLER, Martin. **UnitTest**. Disponível em https://martinfowler.com/bliki/UnitTest.html. Acesso em 10 mar 2023;

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






**Prof. Esp. Guilherme Jorge Aragão da Cruz**


guilherme.cruz@alumni.usp.br


linkedin.com/in/guijac