# Simulação e Teste de Software (CC8550)

## Aula 03 - Planejamento do Teste-Mestre

Prof. Luciano Rossi

Ciência da Computação
Centro Universitário FEI

1º Semestre de 2025

# O que é o Teste Mestre?

## Definição

- Processo de **validação integrada** no qual diferentes tipos de testes são coordenados e executados **conjuntamente.**

centro universitário FEI

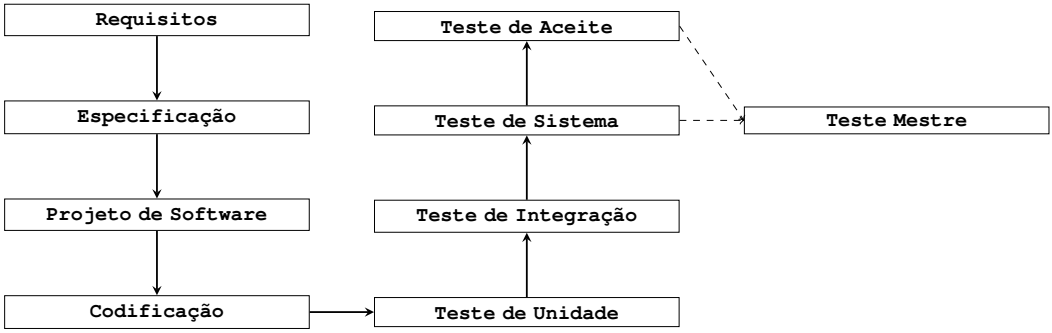
# Onde o Teste Mestre se Encaixa?

**Sequência dos Testes:**

1. Testes Unitários – Verificam módulos isoladamente.
2. Testes de Integração – Avaliam a interação entre módulos.
3. **Teste Mestre – Consolida os testes do sistema e aceite para validar a qualidade final.**
   - 3.1 Testes de Sistema – Avaliam o sistema completo, garantindo que todos os módulos funcionam corretamente juntos e atendem aos requisitos funcionais e não funcionais.
   - 3.2 Testes de Aceitação – Validam o sistema sob a perspectiva do usuário final ou da equipe de QA, garantindo conformidade com os requisitos de negócio.
4. Liberação para Produção – Implantação para uso real.

centro universitário FEI

# Principais Características do Teste Mestre



# Exemplo de Teste de Unidade

```
1 def calcular_saldo(depositos, saques):
2
3         return sum(depositos) - sum(saques)
```

# Exemplo de Teste de Unidade

```python
import unittest
from minha_aplicacao import calcular_saldo

class TestCalcularSaldo(unittest.TestCase):

        def test_saldo_positivo(self):
                depositos = [100, 200, 300]
                saques = [150, 50]
                self.assertEqual(calcular_saldo(depositos, saques), 400)

        def test_saldo_negativo(self):
                depositos = [100, 50]
                saques = [200, 100]
                self.assertEqual(calcular_saldo(depositos, saques), -150)

        def test_saldo_zero(self):
                depositos = [100, 200]
                saques = [300]
                self.assertEqual(calcular_saldo(depositos, saques), 0)

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

# Exemplo de Teste de Integração

```
1  class ContaBancaria:
2          def __init__(self, numero_conta, saldo=0):
3                  self.numero_conta = numero_conta
4                  self.saldo = saldo
5
6          def atualizar_saldo(self, valor):
7                  self.saldo += valor
```

centro universitário FEI

# Exemplo de Teste de Integração

```
1  class Transacao:
2          @staticmethod
3          def deposito(conta, valor):
4                  if valor > 0:
5                          conta.atualizar_saldo(valor)
6                          return True
7                  return False
8
9          @staticmethod
10         def saque(conta, valor):
11                 if 0 < valor <= conta.saldo:
12                         conta.atualizar_saldo(-valor)
13                         return True
14                 return False
```

centro universitário FEI

# Exemplo de Teste de Integração

```python
import unittest
from minha_aplicacao import ContaBancaria, Transacao

class TestIntegracaoContaTransacao(unittest.TestCase):

        def setUp(self):
                self.conta = ContaBancaria(numero_conta="123456", saldo=500)

        def test_deposito(self):
                Transacao.deposito(self.conta, 200)
                self.assertEqual(self.conta.saldo, 700)

        def test_saque_sucesso(self):
                Transacao.saque(self.conta, 300)
                self.assertEqual(self.conta.saldo, 200)

        def test_saque_falha(self):
                resultado = Transacao.saque(self.conta, 600)
                self.assertFalse(resultado)
                self.assertEqual(self.conta.saldo, 500

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

# Exemplo de Teste de Sistema

```python
import time
import pytest
from selenium import webdriver

@pytest.fixture
def browser():
        """Configura o navegador para o teste."""
        driver = webdriver.Chrome()  # Usando Chrome para simulação
        driver.get("https://loja-exemplo.com")  # URL do sistema de pagamento
        yield driver
        driver.quit()

def test_pagamento_sucesso(browser):
        """Teste de sistema que verifica o fluxo completo de pagamento."""

        # Passo 1: Selecionar um produto
        produto = browser.find_element("id", "produto-123")
        produto.click()

        # Passo 2: Ir para o carrinho e finalizar a compra
        browser.find_element("id", "botao-comprar").click()
```

# Exemplo de Teste de Sistema

```
1
2           # Passo 3: Inserir detalhes do pagamento
3           browser.find_element("id", "cartao-numero").send_keys("4111111111111111"
                )
4           browser.find_element("id", "cartao-expiracao").send_keys("12/25")
5           browser.find_element("id", "cartao-cvc").send_keys("123")
6           browser.find_element("id", "botao-pagar").click()
7
8           # Passo 4: Verificar se o pagamento foi processado com sucesso
9           time.sleep(5)  # Aguarda resposta do servidor
10          mensagem_sucesso = browser.find_element("id", "mensagem-sucesso").text
11          assert "Pagamento aprovado" in mensagem_sucesso
```

centro universitário FEI

# Exemplo de Teste de Sistema

```
        # Passo 5: Verificar se o banco de dados foi atualizado corretamente (
            simulado)
        banco_de_dados = {"pedido_123": "Pagamento confirmado"}
        assert banco_de_dados.get("pedido_123") == "Pagamento confirmado"

        # Passo 6: Verificar se o e-mail de confirmação foi enviado (simulado)
        emails_enviados = ["usuario@email.com"]
        assert "usuario@email.com" in emails_enviados
```

centro universitário FEI

# Principais Características do Teste Mestre

## Abrangência Total

- Garante que todos os módulos do software operam corretamente juntos.

## Validação Funcional e Não Funcional

- Confirma conformidade com requisitos especificados e performance esperada.

## Ambiente Realista

- Simula condições reais de uso para evitar falhas em produção.

centro universitário FEI

# Principais Características do Teste Mestre

## Automação e Execução Manual

- Utiliza ferramentas de automação para eficiência e testes manuais para validações críticas.

## Última Verificação Antes da Entrega

- Identifica problemas antes da liberação para clientes ou usuários finais.

# Atividade

| ID | Descrição | Entrada | Saída Esperada | Status |
|---|---|---|---|---|
| UT-01 | Verificar extração do título | XML válido | Retorna o título do artigo | ✓ Passa |
| UT-02 | Verificar extração de autores | XML válido | Retorna lista de autores | ✓ Passa |
| UT-03 | Testar resumo do artigo | XML válido | Retorna o resumo do artigo | ✓ Passa |
| UT-04 | Validar extração de seções | XML válido | Retorna lista de seções | ✓ Passa |
| UT-05 | Validar extração de referências | XML válido | Retorna lista formatada de referências | ✓ Passa |
| UT-06 | Testar comportamento com XML inválido | XML malformado | Retorna erro de parsing | ✓ Passa |

Tabela: Casos de Teste Unitários

centro universitário FEI

# Atividade

| ID | Descrição | Entrada | Saída Esperada | Status |
|---|---|---|---|---|
| UT-07 | Testar extração sem título | XML sem <article-title> | Retorna erro ou string vazia | ✓ Passa |
| UT-08 | Testar extração sem autores | XML sem <contrib> | Retorna lista vazia ou erro tratado | ✓ Passa |
| UT-09 | Testar referências incompletas | XML com referências sem ano ou título | Retorna referências com campos ausentes identificados | ✓ Passa |
| UT-10 | Testar título longo | XML com título extenso | Retorna título completo com formatação correta | ✓ Passa |
| UT-11 | Testar resumo e seções vazias | XML sem conteúdo em <abstract> e <sec> | Retorna string vazia ou aviso de ausência de conteúdo | ✓ Passa |
| UT-12 | Testar resumo longo | XML com resumo muito extenso | Retorna texto completo sem cortes no PDF | ✓ Passa |

Tabela: Casos de Teste Unitários

centro universitário FEI

# Atividade – Testes de Integração

| ID | Descrição | Entrada | Saída Esperada | Status |
|---|---|---|---|---|
| IT-01 | Testar extração e formatação completa do artigo | XML válido | Retorna todas as informações corretamente formatadas | ✓ Passa |
| IT-02 | Testar pipeline de extração e geração de PDF | XML válido | Gera um PDF contendo todas as seções corretamente | ✓ Passa |
| IT-03 | Testar integração com XML sem título | XML sem <article-title> | O PDF é gerado sem título ou com mensagem de aviso | ✓ Passa |
| IT-04 | Testar integração com XML sem autores | XML sem <contrib> | O PDF é gerado sem autores ou com aviso de ausência | ✓ Passa |
| IT-05 | Testar integração com XML sem resumo | XML sem <abstract> | O PDF é gerado sem resumo ou com aviso de ausência | ✓ Passa |
| IT-06 | Testar integração com XML contendo referências incompletas | XML com referências sem ano ou título | O PDF exibe referências com campos ausentes identificados | ✓ Passa |

Tabela: Casos de Teste de Integração

# Atividade – Testes de Integração

| ID | Descrição | Entrada | Saída Esperada | Status |
|---|---|---|---|---|
| IT-07 | Testar integração com XML contendo título longo | XML com título extenso | O título é corretamente exibido no PDF sem cortes | ✓ Passa |
| IT-08 | Testar integração com XML contendo resumo longo | XML com resumo extenso | O resumo é exibido corretamente no PDF sem cortes | ✓ Passa |
| IT-09 | Testar integração com múltiplos autores | XML com vários <contrib> | O PDF exibe todos os autores corretamente formatados | ✓ Passa |
| IT-10 | Testar integração sem seções definidas | XML sem <sec> | O PDF é gerado sem seções ou com aviso de ausência | ✓ Passa |
| IT-11 | Testar fluxo de geração de PDF sem referências | XML sem <ref> | O PDF é gerado sem referências ou com aviso de ausência | ✓ Passa |
| IT-12 | Testar fluxo completo com diversas variações do XML | XMLs variados | O sistema gera PDFs consistentes em todos os casos | ✓ Passa |

Tabela: Casos de Teste de Integração

# Atividade

| ID | Descrição | Cenário | Resultado Esperado | Status |
|---|---|---|---|---|
| MT-01 | Validar geração completa do PDF | XML com título, resumo, seções e referências | PDF gerado corretamente com todos os elementos | ✓ Passa |
| MT-02 | Testar fluxo de um XML corrompido | XML malformado | Sistema identifica erro e retorna mensagem | ✓ Passa |
| MT-03 | Testar processamento de um artigo sem referências | XML sem `<ref-list>` | PDF gerado sem referências, sem erro | ✓ Passa |
| MT-04 | Testar artigo com seção vazia | XML com `<sec>` sem `<p>` | PDF gerado com título da seção, mas sem texto | ✓ Passa |
| MT-05 | Testar um artigo sem autores | XML sem `<contrib-group>` | PDF gerado sem erro, sem autores no cabeçalho | ✓ Passa |
| MT-06 | Testar integração com múltiplos arquivos | Múltiplos XMLs | Todos os arquivos são processados sem erro | ✓ Passa |

Tabela: Casos de Teste Mestre

centro universitário FEI

# Simulação e Teste de Software (CC8550)

## Aula 03 - Planejamento do Teste-Mestre

Prof. Luciano Rossi

Ciência da Computação
Centro Universitário FEI

1º Semestre de 2025