Teste de Software

[4]

Metodologias de desenvolvimento de software

Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas
Prof. Felippe Scheidt – IFPR – Campus Foz do Iguaçu
2024

Tópicos

- O que é teste de software
- Caso de teste
- Criação de testes no python

Introdução

- **Teste** consiste na **investigação** do código-fonte realizada pelos envolvidos no projeto para avaliar a **qualidade** das funcionalidades implementadas.
- Identificar potenciais **riscos** durante o desenvolvimento.
- A realização de testes depende de uma versão **executável** do software.
- Variando as **possibilidades de entradas** podemos verificar a saída esperada, além do tempo de espera de processamento.
- A possibilidade de testes é praticamente **infinita**, requerendo uma estratégia de seleção de testes (plano de teste).

Teste de software

- Avaliar a robustez do código em diferentes cenários.
- Garantir a qualidade do código antes de ser entregue ao usuário.
- Execução de um plano de testes de acordo com um checklist para verificar se os resultados estão de acordo com o comportamento esperado.
- Teste pode ser manual ou automatizado.

Teste unitário

- Teste unitário (unit testing) consiste na metodologia de escrever um teste para cada função ou unidade do programa.
- No desenvolvimento guiado por testes (*test-driven development*) testes unitários são escritos antes de se escrever o código ou função. O código só é considerado completo ou *livre* de falhas até que todos os testes completem com sucesso.

Caso de teste

- Define as condições usadas no teste de software.
- Elaborado para identificar defeitos
- Garantir que os requisitos do software foram atendidos na implementação do software.
- O caso de teste deve especificar os valores de **entrada** e os **resultados esperados** do processamento.

Caso de teste

Pré-condições	Descreve o estado obrigatório do sistema **antes** do início do teste.
Ação	Ações que o usuário deve fazer para que o sistema possa cumprir com o que será testado.
Resultados esperados	É o estado resultante **após** a execução da ação do teste
Pós-condições	Descreve o estado do sistema após a execução do teste.

Caso de teste (exemplo)

Caso de teste: 1.1 - Alterar Senha	
Pré-condições	
• Usuário user1 deve existir	
• Usuário está autenticado como user1	
• Usuário encontra-se na tela de configurações	
Ação	**Resultados esperados**
Botão alterar senha é clicado	Sistema exibe um formulário de alteração de senha
Nova senha digitada (123456)	O campo com a senha digitada possui 6 digitos mascarados
Botão Confirma é clicado	O sistema exibe uma mensagem de sucesso na alteração
Espera 2 segundos	A mensagem de alteração é ocultada
Pós-condições	
• A senha de user1 é 123456	

Configuração ambiente python

Projeto python com pytest:

```
python3 -m venv env
source env/bin/activate
pip install pytest flask
```

Exemplo de test (python)

Ao executar o comando "**pytest -v**" no terminal, a biblioteca pytest procura por classes ou funções que iniciam com o nome **test_*** e executa automaticamente o teste.

```
# arquivo test_math.py
def test_soma():
    n1 = 10
    n2 = 20
    assert n1 + n2 == 300

# digitar no terminal:
pytest -v
```

Assert

- É uma expressão lógica que sempre deve retornar **true** em tempo de execução do código.
- É um recurso importante para **detectar falhas** no código.
- Quando o assert avaliar **false**, ocorrerá uma exceção interrompendo a execução do programa.
- Pode ser colocada no início do código para validar certas condições necessárias para a correta execução do programa.
- Quando colocada no final do código podemos definir o estado final esperado do processamento.

Assert

Por exemplo, o código abaixo contém **dois** asserts.

Ambos devem avaliar a expressão como **true** durante o processamento, caso contrário uma **exceção** será lançada.

```
x = 1;
assert x > 0;
x++;
assert x > 1;
```

Exemplo - cálculo da área

```
# arquivo geometria.py
def calcular_area_quadrado(lado):
    if lado < 0:
        return None
    return lado * lado
```

Exemplo - cálculo da área

```python
# test_geometria.py
# realiza os testes unitarios das funções de cálculo
# da área definidas no arquivo geometria.py
import geometria as geo
class TestAreas:
    def test_calcular_area_quadrado(self):
        assert geo.calcular_area_quadrado(4) == 16
        assert geo.calcular_area_quadrado(-4) is None
        assert geo.calcular_area_quadrado(0) == 0
```

Exercício 1

Criar **3 casos** de teste para a função que faz o cálculo do IMC (índice de massa corporal). Exemplo:

- Peso 70, altura 1.70 - IMC == 24.2

Exercício 2

Criar **casos** de teste para testar a função que faz a classificação do IMC calculado considerando as classes conforme a imagem abaixo. Exemplo: se **IMC é 18**, a função retorna "**Magreza**":

IMC	Classificação
Menor que 18,5	Magreza
18,5 a 24,9	Normal
25 a 29,9	Sobrepeso
30 a 34,9	Obesidade grau I
35 a 39,9	Obesidade grau II
Maior que 40	Obesidade grau III

https://www.tuasaude.com/calculadora/imc/

Referências

1. Unit testing
2. Pytest
3. Assertion