

Testes de Especificação com Behave e Gherkin

Prof. Dr. Valério Gutemberg

 Testes de Software

 Curso de Sistemas para Internet

 Componentes: Emilly Jenniffer, Maria Gabrieli, Riane Brito e Sarah Letícia

O que são Testes de Especificação?

Testes de Especificação são uma abordagem de teste que se concentra na verificação do comportamento do sistema em conformidade com a especificação definida. Utilizando a linguagem Gherkin, esses testes permitem que casos de uso sejam escritos em uma linguagem simples e compreensível, facilitando a comunicação entre as equipes técnicas e não técnicas.

Vantagens dos Testes de Especificação

- **Clareza na comunicação:** Uso de uma linguagem comum (Gherkin) entre desenvolvedores, testers e stakeholders.
- **Facilita a colaboração:** Requisitos são definidos de forma que todos possam entender e contribuir.
- **Documentação viva:** O código dos testes serve como documentação contínua do sistema.
- **Integração com BDD:** Promove a prática de desenvolvimento orientado a comportamento (Behavior-Driven Development).

Desvantagens dos Testes de Especificação

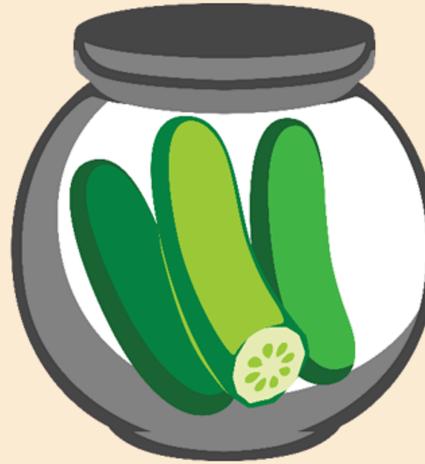
- **Manutenção dos testes:** Pode ser necessário esforço contínuo para manter os testes atualizados conforme o sistema evolui.
- **Curva de aprendizado:** Ferramentas como Behave e a sintaxe Gherkin podem exigir um período de adaptação.
- **Possível complexidade:** Em sistemas muito grandes, a quantidade de cenários pode se tornar difícil de gerenciar.

Quando Usar Testes de Especificação?

- **Requisitos bem definidos:** Ideal quando os requisitos são claros e podem ser descritos de forma precisa.
- **Desenvolvimento Ágil:** Útil em ambientes que seguem práticas de BDD, onde o comportamento do sistema é fundamental.
- **Equipes multidisciplinares:** Quando é essencial que todos, desde desenvolvedores até stakeholders, entendam os testes.

Quando Não Usar Testes de Especificação?

- **Projetos pequenos ou simples:** Pode não valer a pena o esforço para projetos com poucos requisitos.
- **Falta de clareza nos requisitos:** Se os requisitos ainda estão sendo definidos, os testes podem se tornar obsoletos rapidamente.
- **Pouca familiaridade com BDD:** Se a equipe não está familiarizada com BDD, pode ser melhor começar com testes mais tradicionais.



Gherkin

Syntax of BDD

Free text description

Describes a Feature

Including multiple Scenarios

Contains Given, When, Then

used to comments inline

- Feature: Descreve a funcionalidade.
- Scenario: Define uma situação específica.
- Given, When, Then: Estrutura do cenário.
- And, But: Conectores adicionais.

Behave

Behave é uma biblioteca Python que permite a execução de testes escritos em Gherkin. Ele interpreta os arquivos `.feature` e executa os cenários definidos, conectando o texto escrito em Gherkin com o código Python. Os arquivos Python contêm as implementações dos passos (Given, When, Then), e cada função nesses arquivos corresponde a um passo específico definido nos arquivos `.feature`.



```
from behave import given, when, then

@given('que o contato "{nome}" está na lista de contatos')
def step_given_contato_na_lista(context, nome):
    # Prepara a lista de contatos para o teste
    context.contatos = ["João", "Maria", "Ana"]
    assert nome in context.contatos

@when('o usuário busca por "{nome}"')
def step_when_usuario_busca(context, nome):
    # Realiza a busca pelo nome na lista de contatos
    context.resultado = nome if nome in context.contatos else None

@then('o contato "{nome}" deve ser exibido')
def step_then_contato_exibido(context, nome):
    # Verifica se o resultado da busca é o nome esperado
    assert context.resultado == nome
```

Exemplo de Uso do Behave com Gherkin

1. Definição de Cenário:

Feature: Funcionalidade de Login

Scenario: Login bem-sucedido com credenciais válidas

Given que estou na página de login

When eu insiro "gabrieli" como nome de usuário e "senha123" como senha

And clico no botão de login

Then devo ver um alerta de login bem-sucedido

2. Implementação no Behave:

```
@given('que estou na página de login')
def step_given_na_pagina_de_login(context):
    context.browser = webdriver.Chrome() # Inicia o WebDriver para Chrome
    context.browser.get("http://localhost:8000/index.html")
    time.sleep(2) # Pausa de 2 segundos para ver o navegador abrir

@when('eu insiro "{username}" como nome de usuário e "{password}" como senha')
def step_when_inserir_credenciais(context, username, password):
    context.browser.find_element(By.ID, 'username').send_keys(username)
    time.sleep(1) # Pausa de 1 segundo após inserir o nome de usuário
    context.browser.find_element(By.ID, 'password').send_keys(password)
    time.sleep(1) # Pausa de 1 segundo após inserir a senha
```

```
@when('clico no botão de login')
def step_when_clicar_no_botao_de_login(context):
    context.browser.find_element(By.TAG_NAME, 'button').click()
    time.sleep(2) # Pausa de 2 segundos para ver o clique no botão

@then('devo ver um alerta de login bem-sucedido')
def step_then_ver_alerta_de_sucesso(context):
    alert = context.browser.switch_to.alert
    assert alert.text == "Login bem-sucedido!"
    time.sleep(2) # Pausa de 2 segundos para ver o alerta de sucesso
    alert.accept()
    context.browser.quit()
```

3. Continuação da definição do cenário:

Feature: Funcionalidade de Login

Scenario: Falha no login com credenciais inválidas

Given que estou na página de login

When eu insiro "gabrieli" como nome de usuário e "senhaErrada" como senha

And clico no botão de login

Then devo ver uma mensagem de erro "Credenciais inválidas!"

4. Continuação da implementação no Behave:

```
@then('devo ver uma mensagem de erro "{mensagem}"')
def step_then_ver_mensagem_de_erro(context, mensagem):
    error_message = context.browser.find_element(By.ID, 'error-message').text
    assert error_message == mensagem
    time.sleep(2) # Pausa de 2 segundos para ver a mensagem de erro
    context.browser.quit()
```

5. Conclusão

Os Testes de Especificação, especialmente com ferramentas como Behave e Gherkin, são uma abordagem poderosa para garantir que o software se comporte conforme especificado, promovendo uma colaboração mais efetiva entre todos os envolvidos no projeto. Apesar das desvantagens, sua aplicação pode ser crucial em ambientes ágeis e em projetos onde a comunicação clara e a documentação contínua são essenciais.