Fundamentos Engenharia de Software

UNIDADE 06

Testes funcionais

Esta unidade inicia com uma introdução ao teste funcional trazendo o objetivo, os níveis de aplicação e tipos de requisitos associados a este tipo de teste. Posteriormente são apresentados alguns tipos de critérios que podem ser utilizados para especificar casos de testes que busquem identificar possíveis defeitos no código desenvolvido. Ao longo da unidade são apresentados alguns exercícios resolvidos para exemplificar a aplicação da técnica.

Introdução ao Teste Funcional

Como visto na unidade anterior (Testes de Software), os testes de software devem acontecer em etapas distintas do desenvolvimento do software e portanto, há a necessidade de técnicas diferentes para atender os níveis e os tipos de requisitos envolvidos.

O teste funcional pode ser utilizado em todos os níveis de desenvolvimento do software, desde os testes de unidades até os testes de regressão.

O teste funcional, como o próprio nome diz, está focado na funcionalidade, e portanto, a ênfase são os requisitos funcionais. O foco deste tipo de testes são as entradas e as saídas, sem a preocupação em como a funcionalidade está codificada. O componente de *software* a ser testado é abordado como se fosse uma caixa-preta, ou seja, não se considera o comportamento interno do mesmo (Figura 1).

Neste tipo de teste, os dados de entrada são fornecidos, o teste é executado e o **resultado obtido** é comparado a um **resultado esperado.** Se ambos os resultados forem iguais, significa que o código foi desenvolvido conforme a especificação.

A elaboração dos casos de testes, neste tipo de técnica, baseia-se na especificação do requisito funcional. Neste caso, não se pode garantir que partes essenciais ou críticas do *software* foram executadas, pois a ênfase é a especificação da funcionalidade e não como a funcionalidade foi codificada.

Figura 1: Teste funcional (caixa preta)

A figura representa o conceito de teste funcional onde o foco são as entradas e as saídas e não em como o código internamente está implementado. Fonte: O autor (2021).

Critérios para Teste Funcional

Elaborar e executar todos os casos de testes possíveis para identificar possíveis defeitos no código, pode ser uma missão impossível. Quanto mais complexa a funcionalidade, mais entradas e caminhos de execução distintos são possíveis. Portanto, há que se adotar critérios que possibilite a execução da maior parte dos cenários existentes, porém com o menor esforço possível.

A seguir serão apresentados alguns critérios que podem ser utilizados nos testes funcionais de maneira a executar casos de testes de qualidade que possibilite a maior cobertura possível.

Catálogo de ideias

Este critério é o mais informal, ele é baseado nas experiências de projetos anteriores e conforme a equipe identifica casos de testes recorrentes, ela atualiza uma base de casos de testes que devem ser executados por padrão nos projetos.

Exemplo: as ideias de testes para campos numéricos podem ser catalogadas em conjunto para que sejam aplicadas em qualquer campo dessa categoria.

EXERCÍCIO

Exercício 1: Um campo de formulário pode aceitar valores inteiros entre 20 e 50. Quais testes poderiam ser realizados?

Resolução do exercício

Tabela 1: Possíveis resultados exercício 1

Caso de teste	Por que é interessante	Resultado esperado
20	Menor valor válido	Válido

19	Menor valor válido – 1	Inválido
0	0 é sempre interessante	Inválido
Vazio	Campo vazio. O que fazer?	Inválido
49	Valor válido	Válido
50	Maior valor válido	Válido
51	Maior valor válido + 1	Inválido
-1	Número negativo	Inválido
4294967296	2^32, estouro de inteiro	Inválido

A tabela apresenta possíveis casos de testes, elaborados a partir de uma especificação, que precisam ser executados para identificar defeitos na implementação. Fonte: O autor (2021).

Particionamento por equivalência

Este tipo de critério tem o objetivo de minimizar o número de casos de teste, tomando como ponto de partida a identificação de classes que devem ter comportamentos equivalentes.

Neste tipo de critério, inicia-se identificando conjunto de estados válidos e inválidos para uma condição de entrada. Os conjuntos de entradas são sempre uma faixa de valores, um conjunto de valores relacionados, ou uma condição lógica.

Pode ser impossível testar todos os elementos de cada classe de equivalência, porém, pelos menos um elemento de cada classe de equivalência será testado. Se as entradas válidas são especificadas como um intervalo de valores (ex: de 10 a 20), então são definidas uma classe válida, de 10 a 20, e duas classes inválidas: menor do que 10 e maior do que 20.

Para entender na prática a aplicação deste critério, a seguir são apresentados 3 exercícios. Nos dois primeiros exercícios, a solução é apresentada na própria unidade. Já o último exercício, a solução é apresentada em um vídeo com a explicação detalhada do uso do critério.

Para os 3 exercícios o que se pede é: **elaborar casos de testes a partir da especificação utilizando o critério de particionamento por equivalência.**

EXERCÍCIO

Exercício 2: Especificação: "... o cálculo do desconto por dependente é feito de acordo com a idade informada na entrada: Para dependentes com até 12 anos (inclusive) o desconto é de 15%. Entre 13 e 15 (inclusive) o desconto é de 12%. Dos 16 aos 18 (inclusive) o desconto é de 5% e dos 19 aos 21 de 3%"

 Resolução do exercício 

Tabela 2: Possíveis resultados exercício 2

Classes de Equivalência

Classe de entrada	Classe de saída
De 0 a 12 anos	15%
Entre 13 e 15 anos	12%
Entre 16 e 18 anos	5%
Entre 19 e 21 anos	3%
< 0	Idade Inválida
> 21	

Classes válidas

Classes inválidas

Casos de Testes

Caso de Teste	Entrada	Saída
1	8	15%
2	15	12%
3	17	5%
4	21	3%
5	-5	Idade inválida
6	25	Idade inválida

A figura apresenta possíveis casos de testes, elaborados a partir de uma especificação, utilizando o critério particionamento por equivalência. Fonte: O autor (2021).

EXERCÍCIO

Exercício 3: Especificação: Um programa que recebe 3 valores inteiros superiores a zero que correspondem ao tamanho de três vértices representados pelas variáveis A, B e C, obedecem a seguinte classificação:

Equilátero: Três lados iguais

Isósceles: Dois lados iguais

Escaleno: Três lados diferentes

Resolução do exercício

Tabela 3: Possíveis resultados exercício 3

Classes de Equivalência

Classe de entrada	Classe de saída
A = B = C	Equilátero
A = B	Isósceles
A = C	
B = C	
A <> B e B <> C	Escaleno
A < 0	Valor Inválido
B < 0	
C < 0	

Classes válidas

Classes inválidas

Casos de Testes

Casos de Teste	Entrada	Saída
1	5, 5, 5	Equilátero
2	5, 5, 2	Isósceles
3	5, 2, 5	Isósceles
4	2, 5, 5	Isósceles
5	4, 6, 5	Escaleno
6	-5, 1, 2	Valor inválido
7	2, -1, 2	Valor inválido
8	1, 1, -9	Valor inválido

A figura apresenta possíveis casos de testes, elaborados a partir de uma especificação, utilizando o critério particionamento por equivalência. Fonte: O autor (2021).

EXERCÍCIO

Exercício 4: Especificação: Em um programa devem ser lidos os seguintes dados de uma pessoa:

Sexo(1 = masculino e 2 = feminino)

Idade

Estado de saúde (1= boa e = 0 ruim)

Com base nestas informações, o programa deve informar se a pessoa está apta ou inapta para cumprir o serviço militar. Uma pessoa é apta se é do sexo masculino, tem idade igual ou superior a 18 anos e estado de saúde é boa.

Se sexo diferente de 1 e 2 ou idade menor que 0 ou estado de saúde diferente de 0 ou 1 deve ser informada que entrada inválida.

Assista a videoaula desta unidade com a explicação detalhada do resultado.

Análise do valor limítrofe

Os bugs (defeitos) costumam se esconder nas fretas, sendo assim uma boa estratégia é utilizar a técnica de participação por equivalência em conjunto com o critério de análise de valor limítrofe.

A análise de valor limítrofe consiste em considerar valores fronteiriços nas classes de equivalência ou em outros casos de testes.

Por exemplo, se um programa exige uma entrada que, para ser válida, deve estar no intervalo [n..m], então existe 3 classes de equivalência:

- Válida para qualquer n <= x <= m
- Inválida para qualquer x < n
- Inválida para qualquer x > m

Para cada intervalo é indicado que se incluam casos de testes para:

- Valores limite inferior
- Valores limite inferior -1
- Valores limite superior
- Valores limite superior +1

Utilizando a técnica de Particionamento de Equivalência e posteriormente a Análise do Valor Limítrofe, identifique os casos de testes candidatos.

EXERCÍCIO

Exercício 5: Especificação: Considere, um programa com um parâmetro de entrada idade, do tipo inteiro, definido de acordo com as classes de equivalência a seguir:

Menor que 0: inválido

Entre 0 e 17: válido (menor de idade)

Entre 18 e 64: válido (adulto)

Entre 65 e 120: válido (idoso)

Acima de 121: inválido (improvável)

 Resolução do exercício 

Tabela 5: Possíveis resultados exercício 5

Classes de Equivalência

Classe de entrada	Classe de saída	
Idade entre 0 e 17	Menor de idade	Classes válidas
Idade entre 18 e 64	Adulto	
Idade entre 65 e 120	Idoso	
Idade < 0	Valor Inválido	Classes inválidas
Idade >= 121	Valor Inválido	

Casos de Testes

Casos de teste	Entrada	Saída
1	-1	Idade inválida
2	0	Menor de idade
3	10	Menor de idade
4	17	Menor de idade
5	18	Adulto
6	55	Adulto
7	64	Adulto
8	65	Idoso
9	100	Idoso
10	120	Idoso
11	121	Idade inválida
12	125	Idade inválida

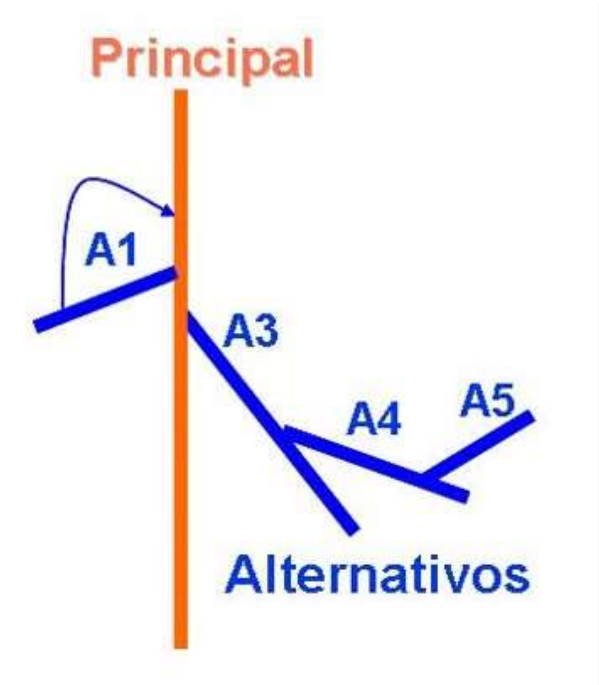
A figura apresenta possíveis casos de testes, elaborados a partir de uma especificação, utilizando o critério particionamento por equivalência. Fonte: O autor (2021).

Caso de uso

Os testes também podem ser especificados a partir de casos de uso ou cenários de negócio. Um caso de uso descreve a interação do sistema com atores e a entrega de um resultado de valor. Cada caso de uso possui pré-condições e pós-condições.

Um caso de uso sempre possui um cenário principal e frequentemente cenários alternativos (inclusive de exceção) (Figura 2).

Figura 2: Representação gráfica de um caso de uso



A figura apresenta os possíveis fluxos que podem ser especificados a partir de um caso de uso. Fonte: O autor (2021).

Como dito, testes funcionais podem acontecer em vários níveis de testes, neste os casos de testes gerados por meio de casos de uso estão alinhados com testes a nível de sistema. O teste de sistema consiste em testar a interface do sistema como se fosse um usuário realizando todos os fluxos possíveis do caso de uso.

A seguir um trecho de uma especificação para um caso de uso “Comprar itens” existente em um contexto de um e-commerce.

Caso de uso: **Comprar itens**

Fluxo principal

1. O usuário insere palavras-chave para localizar item.
2. O sistema apresenta a lista de itens relacionados às palavras-chave.
3. O usuário seleciona um item e informa a quantidade desejada.
4. O sistema informa o itens adicionados até o momento e o valor do pedido.
5. O usuário finaliza a compra.

Fluxo alternativo

A5a. O usuário deseja pesquisar novos itens.

Retorna ao passo 1 do fluxo principal.

Fluxo de exceção

E3a. O usuário indica uma quantidade não disponível no estoque.

E3a1. O sistema informa a quantidade disponível no estoque.

E3a2. O usuário informa uma quantidade disponível no estoque.

Avança ao passo 4 do fluxo principal.

A Tabela 1 apresenta os casos de testes que podem ser elaborados a partir da especificação do caso de uso.

Tabela 1: Casos de testes extraídos da especificação de caso de uso

Caso de teste	Objetivo	Caminho	Como testar	Resultado esperado
1	Fluxo principal.	1, 2, 3, 4, 5	Entrar com palavras-chave, seleciona um item, indica uma quantidade dentro do estoque e finaliza a compra.	Compra com o item e quantidade registrada.
2	Fluxo alternativo: Usuário deseja pesquisar novos itens.	1, 2, 3, 4, 1, 2, 3, 4, 5	Entrar com palavras-chave, seleciona um item, indica uma quantidade dentro do estoque. Em seguida entra com novas palavras-chave, indica uma quantidade dentro do estoque e finaliza a compra.	Compra com os 2 itens e quantidades respectivas registradas.
3	Fluxo de exceção: Usuário indica uma quantidade não disponível no estoque.	1, 2, 3, E3a.1, E3a.2, 4, 5	Entrar com palavras-chave, seleciona um item, indica uma quantidade acima do estoque disponível. Em seguida	Informa quantidade não disponível. Em seguida compra com o item da

			informa uma quantidade disponível no estoque e finaliza a compra.	quantidade disponível registrada.

A tabela apresenta possíveis casos de testes que podem ser elaborados a partir da especificação do caso de uso.
Fonte: O autor (2021).

Considerações finais

- Testes funcionais tem como vantagem requerer apenas a especificação funcional do produto para derivar os requisitos de teste
- Independente da arquitetura do software
- Tanto faz ser orientado a objetos, serviços ou procedural
- Tanto faz se a arquitetura é orientada a componentes ou monolítica
- Não garante que partes críticas e essenciais do código seja cobertos

Conclusão

Nesta unidade foi apresentada a técnica de testes funcionais. Os casos de testes funcionais são gerados a partir da especificação da funcionalidade ou regra de negócio. Estes testes podem ser executados em todos os níveis de teste (unidade, integração, sistema, aceitação e regressão). Ao longo da unidade foram apresentados alguns critérios que podem ser utilizados para a geração dos casos de testes, sendo eles: catálogo de ideias, particionamento por equivalência, análise do valor limítrofe e casos de uso. Para cada tipo de critério foram apresentados exercícios e as respectivas soluções para melhor entendimento da sua aplicação. Embora esta técnica permita a geração de casos de testes com qualidade, é possível que existam defeitos no código que não serão descobertos com os casos de testes gerados por meio da técnica funcional. O tipo de teste estrutural, que tem como foco o código implementado, complementa os testes funcionais e procura garantir a cobertura do código. Na próxima unidade veremos em mais detalhes a aplicação da técnica estrutural, também conhecida como teste de caixa-branca.

Referências bibliográficas

BRAGA, P. H. C. **Testes de Software**. São Paulo, Pearson Education do Brasil, 2016. [Biblioteca Virtual]

PRESSMAN, R. S. ; MAXIM, B. R. **Engenharia de Software: uma abordagem profissional**. 8. ed. – Porto Alegre: AMGH, 2016. [Minha Biblioteca]

WAZLAWICK, R. S. **Engenharia de Software: conceitos e práticas**. 2. Ed. Rio de Janeiro: Elsevier, 2019