Compliance & Quality Assurance

2024

SLIDER

Introdução aos Testes de Software
Aula 3

Prof. Silvio Macedo

Conteúdo

1. Testes Funcionais e Não-funcionais
2. Caixa Branca, Caixa Preta, Caixa Cinza
3. Técnicas de Caixa Branca
4. Técnicas de Caixa Preta
    a. Partição de Equivalência
    b. Análise de Valor Limite
    c. Tabela de Decisão
    d. Transição de estados
5. Outros tipos de teste
    a. Confirmação e Regressão
    b. Smoke e Sanity Tests
    c. Testes Positivos e Negativos
    d. Baseados em experiência

Conceitos Básicos

Testes Funcionais e Não-Funcionais

Testes Funcionais

Envolve testes que avaliam as funções que o sistema deve executar. Os requisitos funcionais podem ser descritos em especificações de requisitos, de negócios, épicos, histórias de usuários, casos de uso ou especificações funcionais. As funções são **“o que”** o sistema deve fazer.

Os testes funcionais devem ser realizados em todos os níveis de teste e considera o comportamento do software.

A cobertura funcional é a medida em que algum tipo de elemento funcional foi exercido por testes, ou seja, a porcentagem de requisitos funcionais que é validada por testes.

O projeto e a execução de testes funcionais podem envolver habilidades ou conhecimentos especiais, como o conhecimento específico de um problema de negócios que o software resolve ou o papel específico que o software desempenha.

Conceitos Básicos

Testes Funcionais e Não-Funcionais

Alguns tipos de testes Funcionais:

- Teste de regressão
- Teste de confirmação
- Teste exploratório
- *Smoke Test*
- *Sanity Test*

Conceitos Básicos

Testes Funcionais e Não-Funcionais

Testes Não-Funcionais

Avaliam as características do software, como usabilidade, performance ou segurança. O teste não funcional valida o **"quão bem"** o sistema se comporta.

O teste não-funcional deve, na medida do possível, também ser realizado em todos os níveis de teste. A descoberta tardia de defeitos não-funcionais pode ser extremamente perigosa e complexa de se resolver.

Conceitos Básicos

Testes Funcionais e Não-Funcionais

A norma ISO 25010 define um modelo com oito características de qualidade de software, com diversas classes e subclasses que podem ser objeto de testes não-funcionais:

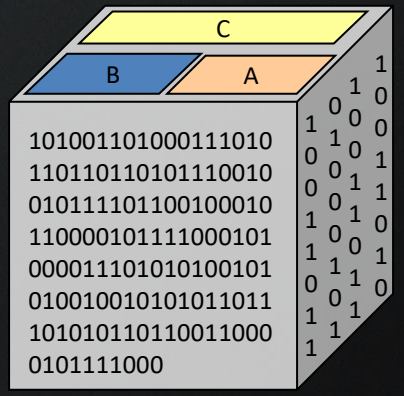
- Teste de Performance
- Teste de Compatibilidade
- Teste de Usabilidade
- Teste de Acessibilidade
- Teste de Confiabilidade
- Teste de Segurança
- Teste de Portabilidade
- Etc.

Conceitos Básicos

Caixa Branca, Caixa Preta, Caixa Cinza

Testes de Caixa-Branca

São testes baseados na estrutura interna ou na implementação do sistema, que pode incluir código-fonte, arquitetura, fluxos de trabalho e fluxos de dados dentro do sistema. Portanto, o testador deve ter conhecimentos de programação e deve entender como o sistema foi implementado.



Testes de Caixa-Preta

São testes baseados na especificação ou comportamento, fundamentados na documentação (requisitos, casos de uso, regras de negócio, etc.), e se concentram nas entradas e saídas do objeto de teste sem referência a sua estrutura interna.

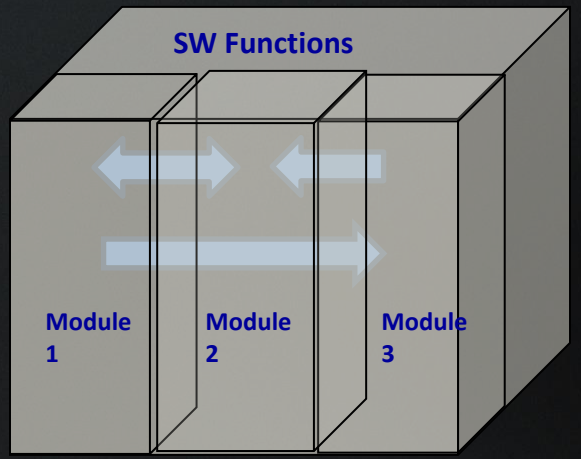
Conceitos Básicos

Caixa Branca, Caixa Preta, Caixa Cinza

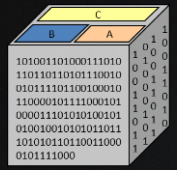
Testes de Caixa-Cinza

É uma combinação dos métodos de teste de caixa-preta e de caixa-branca (portanto, cinza ou semitransparente), onde a estrutura interna do software é parcialmente conhecida pelo testador.

Isso envolve ter acesso a estruturas de dados e algoritmos internos para fins de design de casos de teste, mas ainda testando no nível do usuário e usando técnicas de caixa-preta.

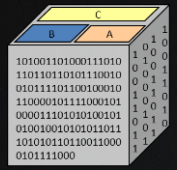


Caixa-Branca
Técnicas de Teste



- **Cobertura de instrução:** técnica que testa as instruções executáveis do código. A cobertura é uma porcentagem do número de instruções executadas pelos testes em relação número total de instruções executáveis existentes, e o objetivo deve ser sempre a cobertura total (embora na prática isso nem sempre seja possível).
- **Cobertura de decisão:** técnica que testa as decisões existentes no código e percorre o código implementado com base nos resultados da decisão. Decisões são pontos no código onde o fluxo de controle escolhe um de dois ou mais resultados possíveis (IF, ou SWITCH/CASE). Os testes de decisão não consideram como uma decisão com múltiplas condições é tomada e podem falhar na detecção de defeitos causados por combinações dessas condições.
- **Cobertura de condição/decisão modificada:** técnica que considera como uma decisão é tomada quando inclui múltiplas condições. Verifica se cada uma das condições únicas afeta de forma independente e correta o resultado da decisão global, proporcionando, portanto, um nível de cobertura mais forte do que a instruções e decisões (quando há múltiplas condições).

Caixa-Branca
Técnicas de Teste



FIAP

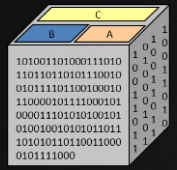
Exemplo: Cobertura de instrução

Para garantir uma cobertura de instrução completa, precisamos criar casos de teste que cubram todas as instruções do código.

```
// Função para calcular o preço final do produto
function calcularPrecoFinal(precoBase, desconto, impostos) {
  let precoComDesconto = precoBase - (precoBase * (desconto / 100));
  let precoFinal = precoComDesconto * (1 + (impostos / 100));
  return precoFinal;
}

// Casos de teste para cobrir todas as instruções
console.log(calcularPrecoFinal(100, 10, 5)); // Teste com desconto e impostos
console.log(calcularPrecoFinal(200, 0, 10)); // Teste sem desconto com impostos
```

Caixa-Branca
Técnicas de Teste



FIAP

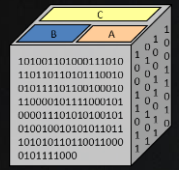
Exemplo: Cobertura de Decisão

Neste caso, queremos garantir que todas as decisões tomadas no código sejam avaliadas tanto para verdadeiro quanto para falso.

```
// Função para verificar se um número é par ou ímpar
function verificarParOuImpar(numero) {
  if (numero % 2 === 0) {
    return "par";
  } else {
  return "ímpar"; }
}

 // Casos de teste para cobrir todas as decisões
console.log(verificarParOuImpar(2));  // Teste com número par console.log(verificarParOuImpar(3)); //
Teste com número ímpar
```

Caixa-Branca
Técnicas de Teste



FIAP

Exemplo: Cobertura de Condição/Decisão Modificada

Esta técnica é uma extensão da cobertura de decisão, onde as condições dentro de cada decisão são consideradas. Aqui, buscamos garantir que todas as combinações possíveis de resultados de condições sejam testadas.

```
// Função para determinar elegibilidade para desconto com base na idade e renda
function verificarElegibilidadeDesconto(idade, renda) {
  if (idade >= 18 && renda >= 20000) {
    return "Elegível para desconto";
  } else {
    return "Não elegível para desconto"; }
}
// Casos de teste para cobrir todas as condições modificadas
console.log(verificarElegibilidadeDesconto(25, 30000)); // Teste elegível para desconto
console.log(verificarElegibilidadeDesconto(30, 20000)); // Teste elegível para desconto
console.log(verificarElegibilidadeDesconto(17, 30000)); // Teste não elegível para desconto
```

Caixa-Preta
Técnicas de Teste

- **Partição de equivalência:** usada para criar partições de equivalência (geralmente chamadas classes de equivalência) a partir de conjuntos de valores que precisam ser processados da mesma maneira. Ao selecionar um valor representativo de uma partição, a cobertura para todos os itens na mesma partição é assumida. Alguns exemplos de classes de equivalência são: entradas válidas e não válidas; letras, números e caracteres especiais; valores segmentados por faixa, etc.).

- **Análise de valor limite:** também chamada de BVA (*Boundary Value Analysis*), a técnica é usada para testar o manuseio adequado dos valores existentes nos limites das partições de equivalência ordenadas. O comportamento nos limites das partições de equivalência tem maior probabilidade de estar incorreto do que o comportamento dentro das partições. Normalmente usa-se o teste com 3 valores: o próprio valor limite, um valor antes e um valor depois.

Caixa-Preta
Técnicas de Teste

FIAP

Exemplo: Imagine que estamos criando uma função para validar a idade de uma pessoa, que deve estar entre 18 e 65 anos.

```
// Função para validar idade
function validarIdade(idade) {
  if (idade >= 18 && idade <= 65) {
    return "Idade válida";
  } else {
    return "Idade inválida";
  }
}
```

```
// Casos de teste usando Partição de Equivalência
console.log(validarIdade(20)); // Teste válido dentro da faixa
console.log(validarIdade(16)); // Teste inválido abaixo da faixa
console.log(validarIdade(70)); // Teste inválido acima da faixa
```

```
// Casos de teste usando Análise de Valor Limite
console.log(validarIdade(18)); // Teste válido no limite inferior
console.log(validarIdade(17)); // Teste inválido logo abaixo do limite inferior
console.log(validarIdade(65)); // Teste válido no limite superior
console.log(validarIdade(66)); // Teste inválido logo acima do limite superior
console.log(validarIdade(30)); // Teste válido dentro da faixa
console.log(validarIdade(70)); // Teste inválido fora da faixa
```

Caixa-Preta
Técnicas de Teste

- **Tabela de decisão:** é uma representação tabular de um conjunto de condições e ações relacionadas, regras expressas que indicam qual ação deve ocorrer para qual conjunto de valores de condição. Condições e ações formam as linhas da tabela. Cada coluna corresponde a uma regra de decisão (combinação de condições que resultam na execução de determinadas ações) e deve ter um teste associado a ela.

Exemplo

Regra de negócio de um site de comércio eletrônico:

- Ao adicionar no carrinho de compras produtos da categoria Eletrônicos com valor igual ou superior a 500,00, aplicar desconto de 10%. Se o usuário for cliente VIP, o desconto deve ser de 15%.
- Ao adicionar no carrinho de compras produtos da categoria Eletro-domésticos com valor igual ou superior a 1000,00, aplicar desconto de 10%. Se o usuário for cliente VIP, o desconto deve ser de 15%.

Caixa-Preta
Técnicas de Teste

FIAP

Exemplo

Condições	Teste 1	Teste 2	Teste 3	Teste 4	Teste 5	Teste 6	Teste 7
É Eletrônico	S	S	S				
É Eletro-doméstico				S	S	S	
Outra categoria							S
Preço >= 500,00	S	S	N				
Preço >= 1000,00				S	S	N	
Cliente VIP	S	N	-	S	N	-	
Ações							
Aplicar desconto de 10%		X			X		
Aplicar desconto de 15%	X			X			
Não aplicar desconto			X			X	X

Caixa-Preta
Técnicas de Teste

- **Transição de estado:** usado para testar a capacidade do sistema de entrar e sair de estados definidos por meio de transições válidas, assim como para tentar entrar em estados inválidos ou cobrir transições inválidas. Eventos fazem com que o sistema faça transições de estado e execute ações. Dependendo do estado do sistema, o mesmo evento pode resultar em comportamentos diferentes. Recomenda-se a criação de um diagrama de estados, com as transições válidas, e depois traduzi-lo a uma tabela de estados, adicionando também condições potencialmente inválidas.

Caixa-Preta
Técnicas de Teste

Exemplo: Máquina de estados de um relógio simples.

Diagrama de estados

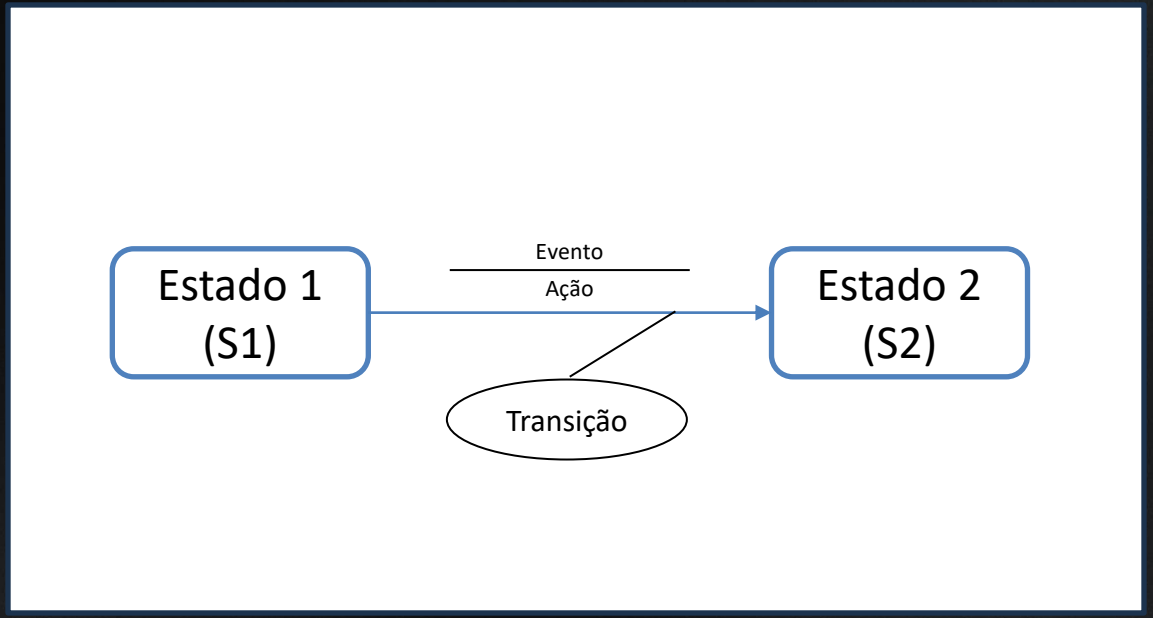


Tabela de estados

- Estado inicial
- Entrada
- Saída
- Estado final

Caixa-Preta
Técnicas de Teste

Exemplo: Máquina de estados de um relógio simples.

Diagrama de estados

Tabela de estados

- Estado inicial
- Entrada
- Saída
- Estado final

Caixa-Preta
Técnicas de Teste

Exemplo: Máquina de estados de um relógio simples.

Diagrama de estados

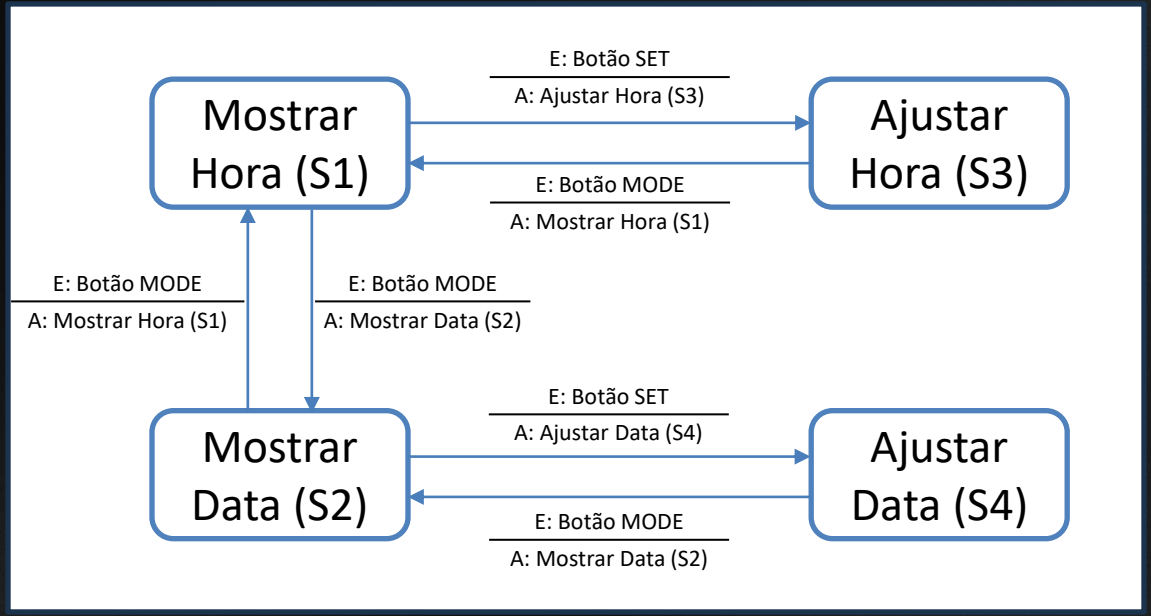


Tabela de estados

Estado inicial	S1	S1	S2
Entrada	MODE	SET	MODE
Saída	Mostrar Data	Ajustar Hora	Mostrar Hora
Estado final	S2	S3	S1

Idealmente, cada transição do diagrama ou cada coluna da tabela deve gerar um caso de teste (TC).

Adicionalmente, transições inválidas ou não representadas podem também gerar **testes negativos**.

Outros Tipos de Teste

- **Teste de Confirmação:** Depois que um defeito é corrigido, todos os casos de teste que falharam devido ao defeito devem ser executados novamente na nova versão. No mínimo, as etapas para reproduzir as falhas causadas pelo defeito devem ser executadas novamente, com a finalidade de confirmar se o defeito original foi corrigido com sucesso.

- **Teste de Regressão:** É possível que uma alteração feita em uma parte do código (ou mesmo no ambiente), seja uma correção ou nova implementação, possa afetar acidentalmente o comportamento de outras partes do sistema. Tais efeitos colaterais não intencionais são chamados de regressões. Portanto, o teste de regressão envolve a execução de um conjunto testes básicos que abrangem as principais funcionalidades do sistema, com o intuito detectar eventuais efeitos colaterais indesejados. Os conjuntos de testes de regressão são executados muitas vezes (em toda nova release), portanto é um forte candidato à automação.

Outros Tipos de Teste

- **Smoke Tests:** pequeno grupo de testes que verifica se as funcionalidades básicas do sistema estão operacionais em uma determinada release. Eles devem ser sempre os primeiros testes executados no escopo, sendo que a aprovação neles é a condição para a continuidade das demais atividades de teste programadas.

- **Sanity Tests:** o teste de sanidade é feito quando o tempo disponível para validação é muito curto, sendo então um conjunto pequeno de testes simples, mas que deve cobrir todas as funcionalidades principais do sistema. Pode ser considerado um subconjunto do Teste de Regressão. Ao contrário do *Smoke Test* e do Teste de Regressão, que são os testes iniciais de um escopo maior, o *Sanity Test* normalmente é o único escopo possível para a release, dada a restrição de tempo.

Outros Tipos de Teste

- **Testes Positivos:** também chamado de teste de “caminho feliz”, é normalmente a primeira abordagem que um testador usaria numa atividade de teste. Ele considera o cenário perfeito do usuário final, usando dados válidos e corretos como entrada ou apenas executando as ações como deveria ser feito. O método testa se o sistema faz o que deveria fazer.

- **Testes Negativos:** também chamado de teste de “caminho errado”, tem a intenção de garantir a estabilidade da aplicação. É o processo de validar o sistema usando dados inválidos ou performando ações da maneira incorreta, verificando se o sistema lida bem com tais situações e mostra mensagens de erro ao usuário como deveria. O método testa se o sistema não faz o que não deveria fazer.

Outros Tipos de Teste

Testes Baseados na Experiência

- **Suposição de Erros:** técnica usada para prever a ocorrência defeitos e falhas com base no conhecimento do testador, que usa sua experiência para tentar adivinhar possíveis erros que poderiam ter sido cometidos no desenvolvimento. Informações importantes que compõem esse conhecimento são: comportamento da aplicação no passado, tipos de erros que tendem a ser cometidos, falhas em outros aplicativos similares, etc.

- **Testes Exploratórios:** técnica em que testes informais (não pré-definidos) são modelados, executados, registrados e avaliados dinamicamente durante a execução da tarefa. Um bom teste exploratório é planejado, interativo e criativo. É frequentemente usado para complementar outras técnicas de teste e servir como base para o desenvolvimento de casos de teste adicionais.

Referências

ISTQB CTFL Syllabus v3.1.1: https://bcr.bstqb.org.br/docs/syllabus_ctfl_3.1.1br.pdf
ISTQB CTAL-TA Syllabus v3.1.2: https://bcr.bstqb.org.br/docs/syllabus_ctal_ta_3.1.2br.pdf
ISTQB CTAL-TTA Syllabus v4.0: https://bcr.bstqb.org.br/docs/syllabus_ctal_tta_4.0br1.pdf
https://softwaretestingfundamentals.com/
https://www.softwaretestinghelp.com/