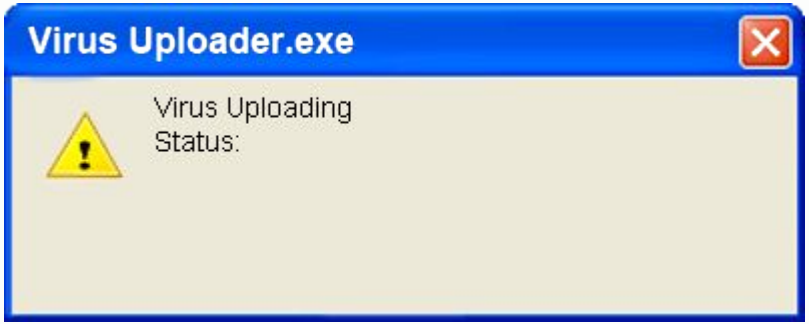


UniCesumar

Projeto, implementação e Teste de Software

Artefatos de Teste de Software

Prof. Esp. Dacio F. Machado


LOADING...

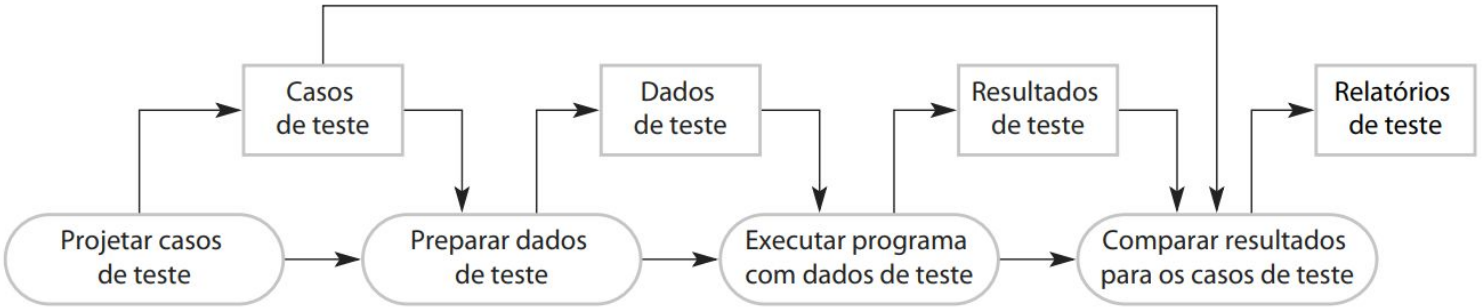
Anteriormente em Teste de Software

- Artefatos de Teste de Software
  - Plano de Teste de Software / Caso de Teste / Roteiro de Teste
  - Relatórios de Teste de Software

Conceitos Básicos em Teste de Software

Teste é um conjunto de atividades que podem ser planejadas com antecedência e executadas sistematicamente. Para isso precisamos definir alguns conceitos, papéis e artefatos necessários para estruturar um teste de software da maneira correta.

Figura 8.3 Um modelo do processo de teste de software



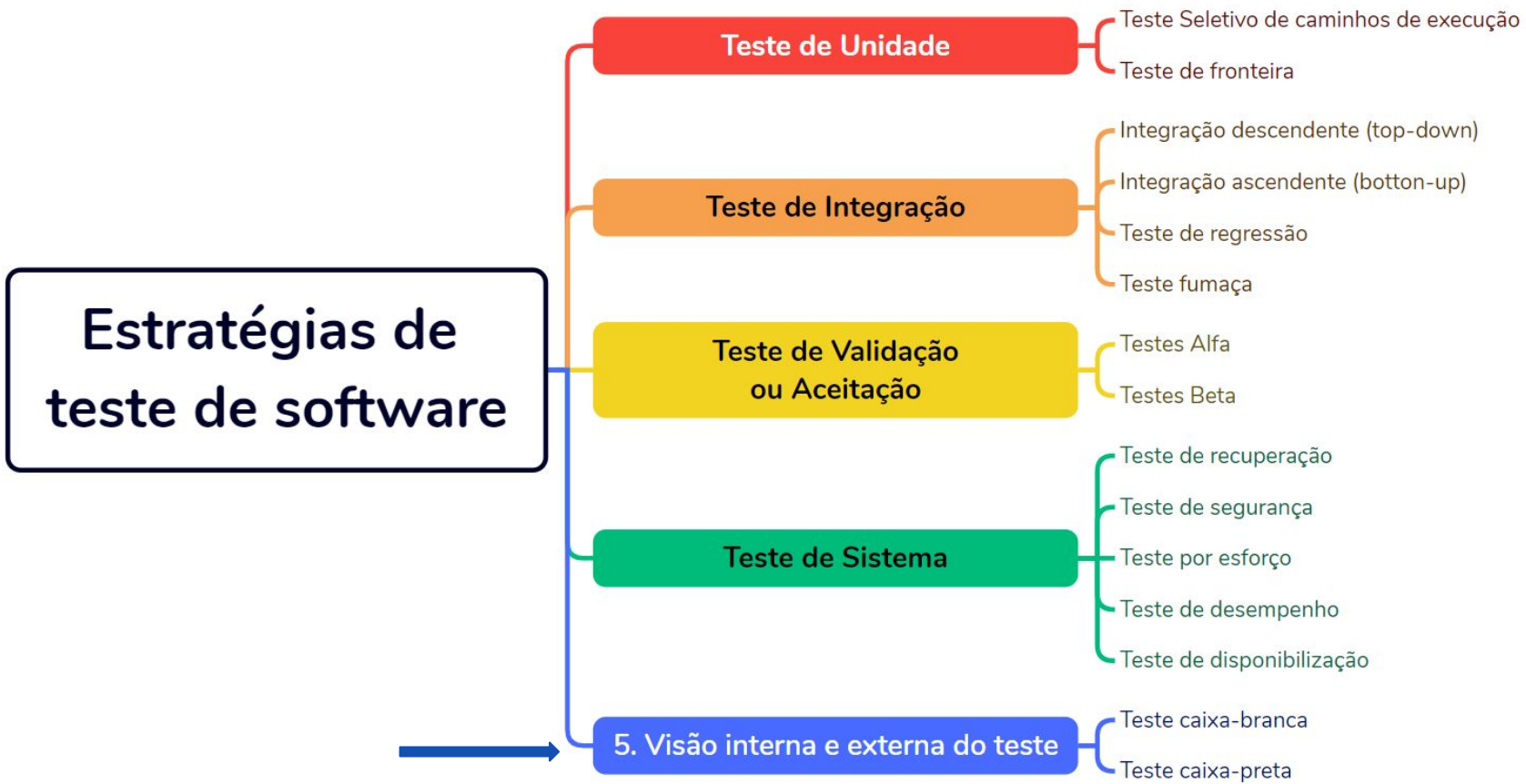
Conceitos Básicos em Teste de Software

PLANO DE TESTE: É um documento que descreve a abordagem geral.

CASOS DE TESTE: São documentos que descrevem os cenários específicos que serão testados

ROTEIRO DE TESTE: São documentos que detalham a sequência de etapas que os testadores devem seguir ao executar os casos de teste

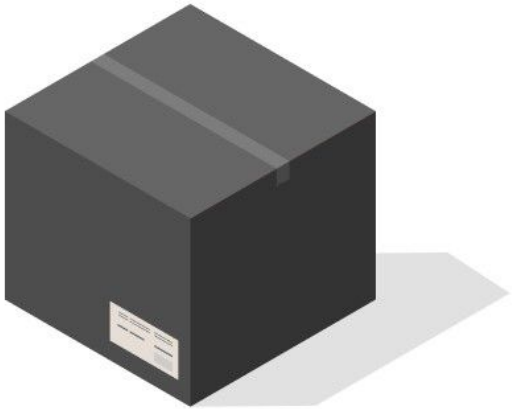
RELATÓRIO DE EXECUÇÃO DE TESTES: São documentos que resumem os resultados dos testes executados


Estratégias de teste de software
Teste de Unidade
Teste Seletivo de caminhos de execução
Teste de fronteira
Teste de Integração
Integração descendente (top-down)
Integração ascendente (botton-up)
Teste de regressão
Teste fumaça
Teste de Validação ou Aceitação
Testes Alfa
Testes Beta
Teste de Sistema
Teste de recuperação
Teste de segurança
Teste por esforço
Teste de desempenho
Teste de disponibilização
5. Visão interna e externa do teste
Teste caixa-branca
Teste caixa-preta

Testes Funcionais

Visão de Teste

Segundo Pressman (2011), qualquer produto de engenharia pode ser testado a partir de duas perspectivas diferentes:



Black box - we do not know anything

White box - we know everything

Visão de Teste

(1) a lógica interna do programa é exercitada usando técnicas de projeto de caso de teste “caixa branca”;

(2) os requisitos de software são exercitados usando técnicas de projeto de casos de teste “caixa preta”.

Caixa Branca

A Primeira abordagem requer uma visão interna e é chamada de teste caixa-branca.

• Fundamenta-se em um exame rigoroso do detalhe procedimental.

• Os caminhos lógicos do software e as colaborações entre componentes são testados.

Uma abordagem para testes de programas onde os testes são baseados no conhecimento da estrutura do programa e de seus componentes.

Acesso ao código-fonte é essencial para testes de caixa branca.

Caixa Preta

• A Segunda abordagem de teste usa uma visão externa e é chamada de teste caixa-preta

• Faz referência a testes realizados na interface do software

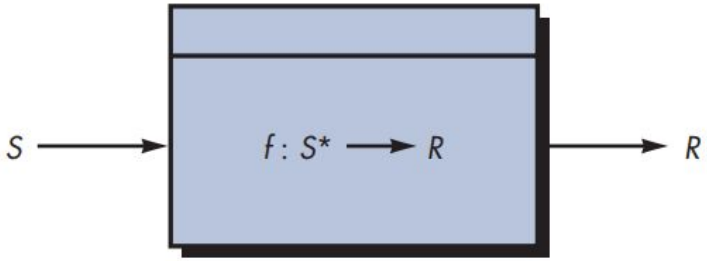
• Examina alguns aspectos fundamentais de um sistema, com pouca preocupação em relação à estrutura lógica interna do software

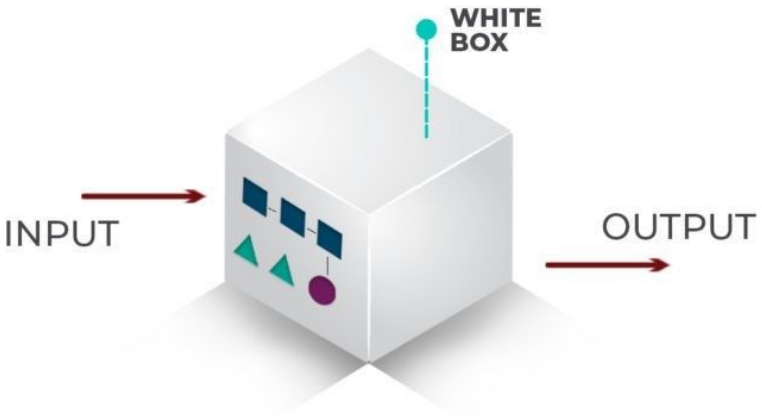
Uma abordagem de testes onde os testadores não têm acesso ao código-fonte do sistema ou seus componentes. Os testes são derivados da especificação do sistema.

O programa executável final é tratado como uma caixa preta, e os testes são planejados para mostrar se o sistema atende ou não a seus requisitos

FIGURA 21.3

Uma especificação caixa-preta





CAIXA PRETA	CAIXA BRANCA
Verifica as operações e ações de uma aplicação.	Verifica o comportamento de uma aplicação.
É baseado nos requisitos do cliente.	É baseado nas expectativas do cliente.
Ajuda a melhorar o comportamento da aplicação.	Ajuda a melhorar o desempenho da aplicação.
O teste de caixa preta é fácil de executar manualmente.	É difícil executar o teste de caixa branca manualmente.
Ele testa o que o produto faz.	Ele descreve como o produto se sai.
O teste é baseado nos requisitos do negócio.	O teste é baseado nos requisitos de desempenho.
Exemplos	
1 - Teste de Unidade **2 - Teste de Integração** **3 - Teste de Regressão**	**1 - Teste de Desempenho** **2 - Teste de Carga** **3 - Teste de Estresse**

Aspecto	Caixa Branca	Caixa Preta
Foco	Estrutura interna, lógica e código	Funcionalidade e comportamento externo
Objetivo	Verificar a execução correta do código	Validar a conformidade com os requisitos
Exemplos de Técnicas	Cobertura de caminho, condição, loop	Testes de equivalência, valor limite
Ferramentas	JUnit, NUnit, PyTest	Selenium, QTP/UFT, Postman

Níveis de Teste

- **Teste de Unidade / Unitário**
- **Teste de Componentes**
- **Teste de Integração**
- **Teste de Sistema**
- **Teste de Regressão**
- **Teste de Aceitação**

Técnicas de Teste

- **Teste Caixa-Branca / *White-Box* / Estrutural**
- **Teste Caixa-Preta / *Black-Box* / Funcional**

Tipos de Teste

- **Teste de Usabilidade**
- **Teste de Desempenho / *Performance***
- **Teste de Carga**
- **Teste de Estresse / Esforço**
- **Teste de Segurança**

Teste Funcional

Teste **FUNCIONAL** ou **CAIXA-PRETA** focaliza os requisitos funcionais do sistema

- As técnicas de teste caixa-preta permitem derivar séries de condições de entrada que utilizarão completamente todos os requisitos funcionais para um programa

- O teste caixa-preta não é uma alternativa às técnicas caixa-branca.

- É uma abordagem complementar, com possibilidade de descobrir uma classe de erros diferente

Teste Funcional

Segundo Sommerville (2011) Devido ao teste caixa-preta propositadamente desconsiderar a estrutura de controle, a atenção é focalizada no domínio das informações. Os testes são feitos para responder às seguintes questões:

- Como a validade funcional é testada?
- Como o comportamento e o desempenho do sistema é testado?
- Que classes de entrada farão bons casos de teste?
- O sistema é particularmente sensível a certos valores de entrada?
- Que taxas e volumes de dados o sistema pode tolerar?
- Que efeito combinações específicas de dados terão sobre a operação do sistema?

Teste Funcional

O teste caixa-preta tenta encontrar erros nas seguintes categorias:

(1) **funções** incorretas ou faltando;
(2) erros de **interface**;
(3) erros em **estruturas de dados** ou **acesso** a **bases** de **dados externas**;
(4) erros de **comportamento** ou de desempenho;
(5) erros de **inicialização** e **término**

Características do Teste Funcional

Os testes funcionais são projetados para simular situações reais de uso do software, a fim de validar se:

- Os critérios definidos são atendidos
- As funcionalidades estão implementadas de acordo com as **expectativas dos usuários e dos stakeholders**

- Os testes buscam examinar como o software responde às entradas, como o software interage com o usuário e como o software produz saídas esperadas

Características do Teste Funcional

- O teste funcional tem foco no comportamento do sistema
- Os testes funcionais são projetados para avaliar o software a partir da perspectiva do usuário
- NÃO estão preocupados com a implementação interna, apenas com o comportamento externo do sistema
- Os testes funcionais normalmente envolvem a criação de cenários de uso realista
- Simular as ações dos usuários, para verificar se o software executa corretamente nessas situações

Critérios do Teste Funcional

Os critérios mais conhecidos da técnica de teste funcional são Particionamento de Equivalência, Análise do Valor Limite, Grafo Causa-Efeito e Error-Guessing. Além desses, também existem outros, como, por exemplo, Teste Funcional Sistemático.

Como todos os critérios da técnica funcional baseiam-se apenas na especificação do produto testado, a qualidade de tais critérios depende fortemente da existência de uma boa especificação de requisitos.

Critérios do Teste Funcional

Os critérios mais conhecidos da técnica de teste funcional são Particionamento de Equivalência, Análise do Valor Limite, Grafo Causa-Efeito e Error-Guessing. Além desses, também existem outros, como, por exemplo, Teste Funcional Sistemático.

Como todos os critérios da técnica funcional baseiam-se apenas na especificação do produto testado, a qualidade de tais critérios depende fortemente da existência de uma boa especificação de requisitos.

Classes de Equivalência

Classes de equivalência ajudam a organizar e simplificar a criação de casos de teste, permitindo que você escolha representantes de um grupo de dados que compartilham características semelhantes

Isso é particularmente útil quando se trata de testar diferentes valores de entrada que devem ser tratados de maneira semelhante pelo sistema, e que podem pertencer a um conjunto infinito de possibilidades

Ao testar um valor em uma classe, você pode fazer suposições sobre o comportamento do sistema em relação a outros valores na mesma classe.

Classes de Equivalência

Uma classe de equivalência representa um conjunto de estados válidos ou inválidos para as condições de entrada.

Uma vez identificadas as classes de equivalência, devem-se determinar os casos de teste, escolhendo-se um elemento de cada classe, de forma que cada novo caso de teste cubra o maior número de classes válidas possível.

Especificação do programa “Cadeia de Caracteres”:

O programa solicita do usuário um inteiro positivo no intervalo entre 1 e 20 e então solicita uma cadeia de caracteres desse comprimento. Após isso, o programa solicita um caractere e retorna a posição na cadeia em que o caractere é encontrado pela primeira vez ou uma mensagem indicando que o caractere não está presente na cadeia. O usuário tem a opção de procurar vários caracteres.

Classes de Equivalência

Considerando a especificação dada anteriormente, têm-se quatro entradas:

T – tamanho da cadeia de caracteres;

CC – uma cadeia de caracteres;

C – um caractere a ser procurado;

O – a opção por procurar mais caracteres.

De acordo com a especificação, as variáveis CC e C não determinam classes de equivalência, pois os caracteres podem ser quaisquer. Já o tamanho da cadeia deve estar no intervalo entre 1 e 20 (inclusive) e a opção do usuário pode ser “sim” ou “não”.

Variável de entrada	Classes de equivalência válidas	Classes de equivalência inválidas
T	$1 \leq T \leq 20$	$T < 1$ e $T > 20$
O	Sim	Não
C	Caractere que pertence à cadeia	Caractere que não pertence à cadeia

Variáveis de entrada				Saída esperada
T	CC	C	O	
34				entre com um inteiro entre 1 e 20
0				entre com um inteiro entre 1 e 20
3	abc	c		o caractere c aparece na posição 3 da cadeia
			s	
		k		o caractere k não pertence à cadeia
			n	

Entrada				Saída esperada
T	CC	C	O	
21				entre com um inteiro entre 1 e 20
0				entre com um inteiro entre 1 e 20
1	a	a		o caractere a aparece na posição 1 da cadeia
			s	
		x		o caractere x não pertence à cadeia
			n	
20	abcdefghijklmnopqrst	a		o caractere a aparece na posição 1 da cadeia
			s	
		t		o caractere t aparece na posição 20 da cadeia
			n	

Classes de Equivalência

Considerando a especificação dada anteriormente, têm-se quatro entradas:

T – tamanho da cadeia de caracteres;

CC – uma cadeia de caracteres;

C – um caractere a ser procurado;

O – a opção por procurar mais caracteres.

De acordo com a especificação, as variáveis CC e C não determinam classes de equivalência, pois os caracteres podem ser quaisquer. Já o tamanho da cadeia deve estar no intervalo entre 1 e 20 (inclusive) e a opção do usuário pode ser “sim” ou “não”.

Identificação	CT_001	
Itens a Testar	Caso de uso Abertura do Caixa	
Entradas	**Campo**	**Valor**
	Nome	Vendedor_A
	Login	VD_A
	Senha	ADM123
Saídas Esperadas	**Campo**	**Valor**
	Nome	Ademirson Luiz da Silva
	Grupo do Usuário	Vendedor
Ambiente	Banco de Teste	
Procedimento	Inclusão de Usuário - ET-001-PT-01	
Dependência	Banco de dados vazio	

Realização de Teste Funcional

O primeiro passo no teste caixa-preta é entender os objetos que são modelados no software e as relações que unem esses objetos.

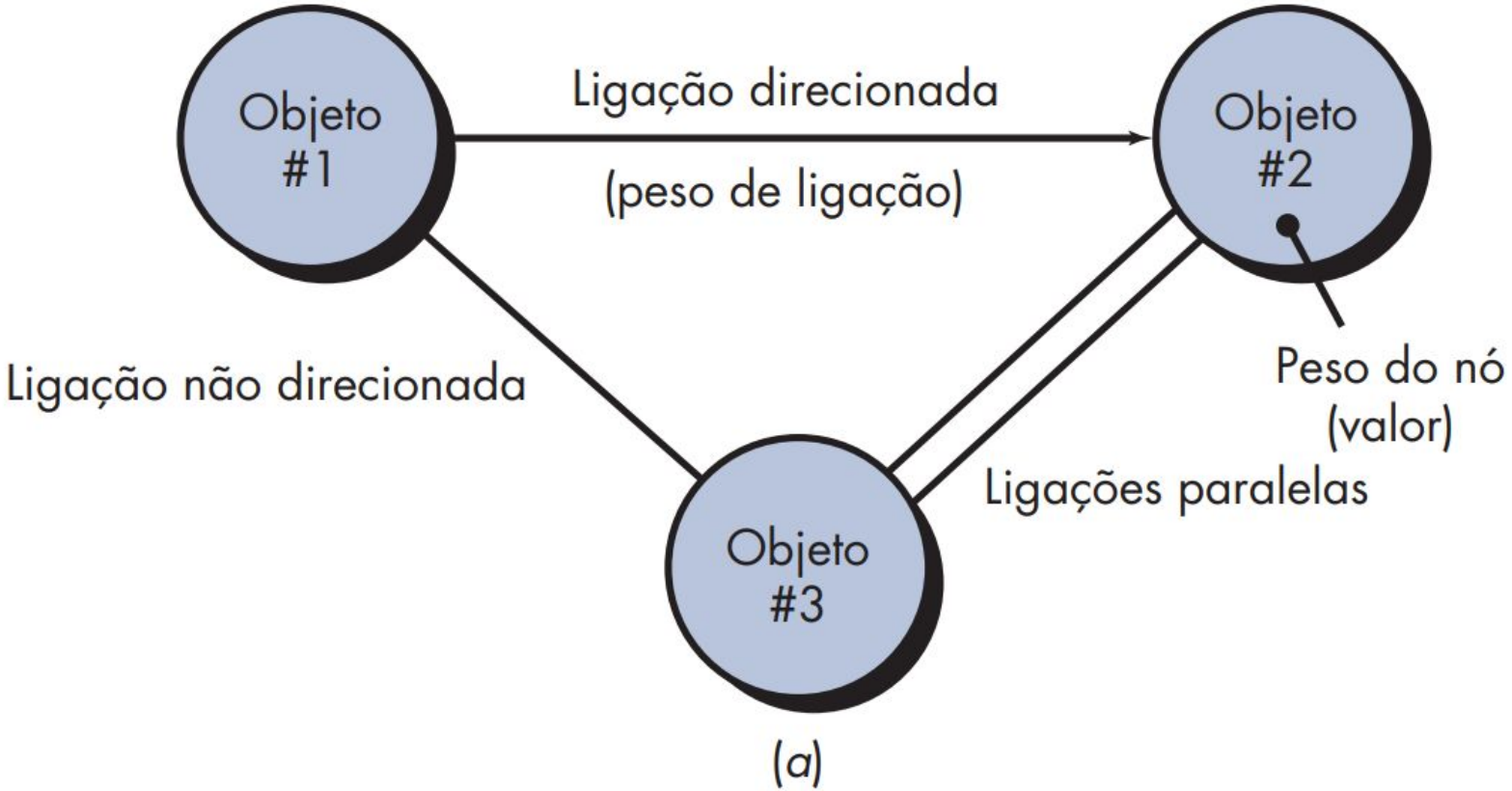
Uma vez conseguido isso, o próximo passo é definir uma série de testes que verificam que “todos os objetos têm a relação esperada uns com os outros”.

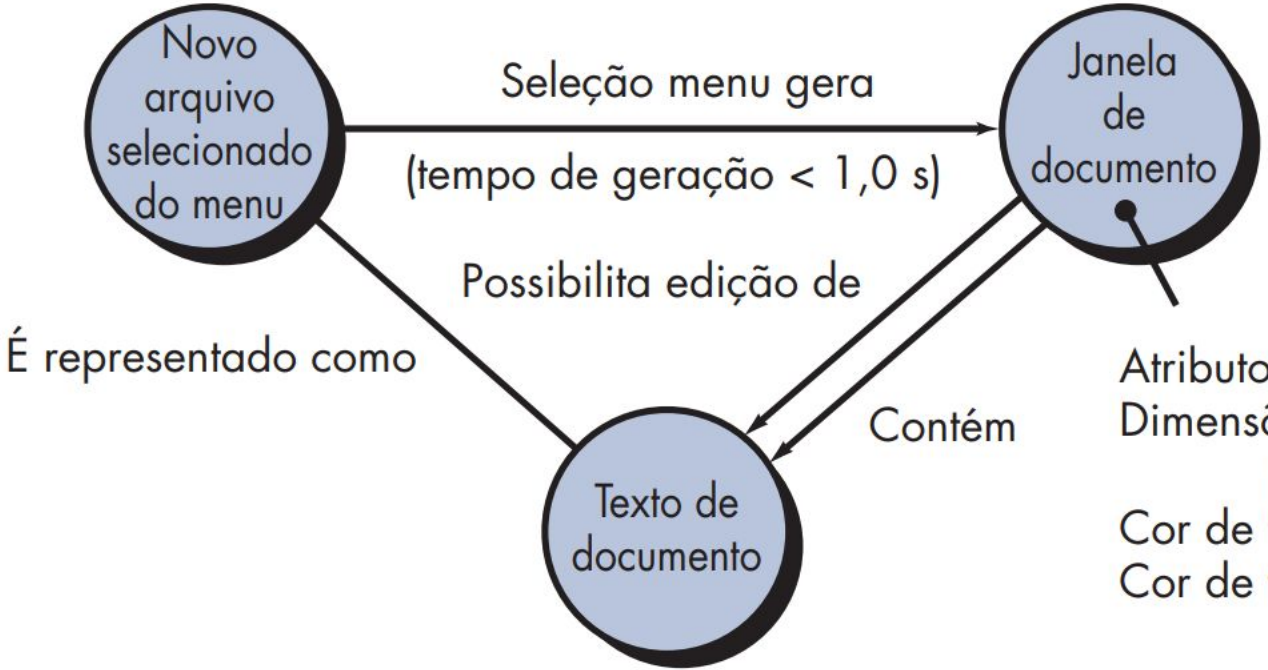
O teste começa criando um grafo de objetos importantes e suas relações e então imaginando uma série de testes que abrangerá o grafo.

Realização de Teste Funcional

Você começa criando um grafo:

- Uma coleção de nós que representam objetos,
- Ligações que representam as relações entre objetos,
- Pesos de nó que descrevem as propriedades de um nó (por exemplo, o valor específico de um dado ou comportamento de estado),
- Pesos de ligação que descrevem alguma característica de uma ligação.


Objeto
#1
Ligação direcionada
(peso de ligação)
Objeto
#2
Ligação não direcionada
Peso do nó
(valor)
Ligações paralelas
Objeto
#3
(a)


Novo arquivo selecionado do menu
Seleção menu gera
(tempo de geração < 1,0 s)
Janela de documento
Possibilita edição de
É representado como
Contém
Texto de documento
Atributos:
Dimensões iniciais: estabelecidas por padrão ou preferências
Cor de fundo: banco
Cor de texto: cor padrão ou preferencial

(*b*)

FRAMEWORKS PARA TESTE FUNCIONAL

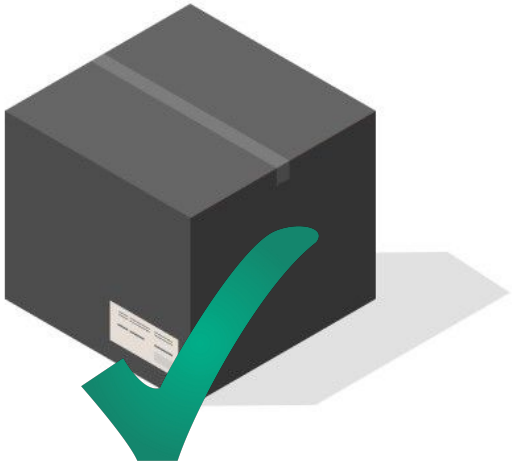
- JUnit (Java): Ele suporta testes de unidade, testes de integração e testes funcionais.
- TestNG (Java): TestNG é uma alternativa ao JUnit para testes de unidade e funcionais em Java.
- pytest (Python): Ele é fácil de usar e oferece recursos avançados de descoberta automática de testes e geração de relatórios.
- NUnit (C#): Ele oferece suporte a parametrização de testes e outras funcionalidades avançadas.

FRAMEWORKS PARA TESTE FUNCIONAL

- Cucumber (Várias Linguagens): O Cucumber é uma ferramenta de teste de aceitação que utiliza a linguagem Gherkin para escrever cenários de teste em linguagem natural. Ele é frequentemente usado para testes funcionais.
- Selenium (Web Applications): O Selenium é uma ferramenta popular para testar aplicativos da web. Ele permite a automação de testes de interface do usuário em navegadores
- Robot Framework (Várias Linguagens): O Robot Framework é uma estrutura genérica de automação de teste que pode ser usada para testes funcionais e de aceitação, suportando várias linguagens de programação.
- PHP Unit (PHP): O PHPUnit é um framework de teste para PHP, projetado para testes de unidade e funcionais. Ele segue uma abordagem semelhante ao JUnit

Próxima Aula

técnicas de projeto de caso de teste “caixa branca”



Black box - we do not know anything

White box - we know everything

OBRIGADO

dacio.francisco@unicesumar.edu.br