Projeto, implementação e Teste de Software

Revisão Pré-Prova
1 Bimestre

Prof. Esp. Dacio F. Machado

Ementa

Disciplina	Distribuição de Carga Horária					
	Teórica	Prática			EAD	Total
PROJETO, IMPLEMENTAÇÃO E TESTE DE SOFTWARE	40	40				80

Ementa

Princípios e técnicas de teste de software. Verificação e Validação. Planos de teste. Documentação de estratégias de testes e outros artefatos. Revisão de software. Testes de unidade. Técnicas de teste funcional (caixa preta). Técnicas de teste estrutural (caixa branca). Testes de integração. Desenvolvimento de casos de teste. Testes de sistema. Testes de aceitação. Testes de atributos de qualidade. Testes de regressão. Testes automatizados e ferramentas de apoio. Teste em ferramentas de integração contínua. Desenvolvimento dirigido por testes. Gerenciamento do processo de teste. Registro e acompanhamento. Conceitos de qualidade de software. Padrões de qualidade de software. Visão geral de CMMI e MPS-BR.

Cronograma da Disciplina

1º	Princípios e técnicas de teste de software	Teoria
2º	Princípios e técnicas de teste de software	Teoria
3º	~~Verificação e validação~~	Teoria
4º/5º	~~Verificação e validação~~	Prática
6º	Planos de Teste	Teoria
7º/ 8º	Planos de Teste	Prática
9º	Documentação de estratégias de testes e outros artefatos.	Teoria
10º/ 11º	Documentação de estratégias de testes e outros artefatos.	Prática
12º	~~Revisão de Software~~	Teoria
14º/ 13º	~~Revisão de Software~~	Prática

Cronograma da Disciplina

15º	Técnicas de teste funcional (caixa preta)	Teoria
16º/17º	Técnicas de teste funcional (caixa preta)	Prática
18º	Técnicas de teste estrutural (caixa branca)	Teoria
19º/20º	Técnicas de teste estrutural (caixa branca)	Prática
21º	~~Teste de unidade~~	Teoria
22º/ 23º	~~Teste de unidade~~	Prática
24º	~~Teste de unidade - Casos de teste~~	Teoria
25º/ 26º	~~Teste de unidade - Casos de teste~~	Prática
27º/28	~~Testes de aceitação~~	Teoria / Prática
29º	PROVA DO PRIMEIRO BIMESTRE	Prática

Teste ?

Testar um software consiste em:

- Verificar se ele atende às expectativas
- Se seu funcionamento é limpo, amigável e correto
- Se ele se enquadra no ambiente para o qual foi projetado

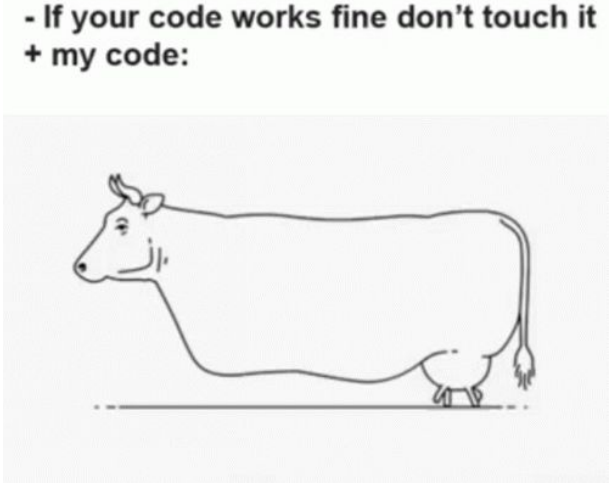
O teste tenta garantir que o programa funcione conforme o esperado, seja seguro, confiável e atenda às necessidades dos usuários

Porque Testar software ?

O teste é uma atividade de verificação e validação do software e consiste na análise dinâmica, isto é, na execução do produto de software com o objetivo de verificar a presença de defeitos no produto e aumentar a confiança de que está correto.

Os testes não conseguem demonstrar que o software está livre de defeitos ou que vai se comportar sempre de acordo com a sua especificação, em qualquer circunstância.

Sempre é possível que um teste negligenciado descubra mais problemas com o sistema



Erro, Defeito e Falha

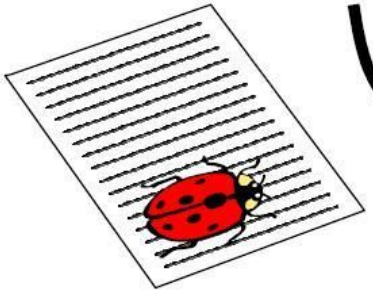
- **Defeito fault:** é um passo, processo ou definição de dados que está incorreto em um programa de computador
- **Engano mistake:** é entendida como uma ação humana que produz um defeito
- **Erro error:** é um estado interno incorreto de um programa que é a manifestação de algum defeito
- **Falha failure:** é quando o erro se propaga para a saída do programa, levando a uma saída diferente da esperada, ou seja, o programa produz um comportamento incorreto em relação a sua especificação

Erro

Uma pessoa comete um erro ...

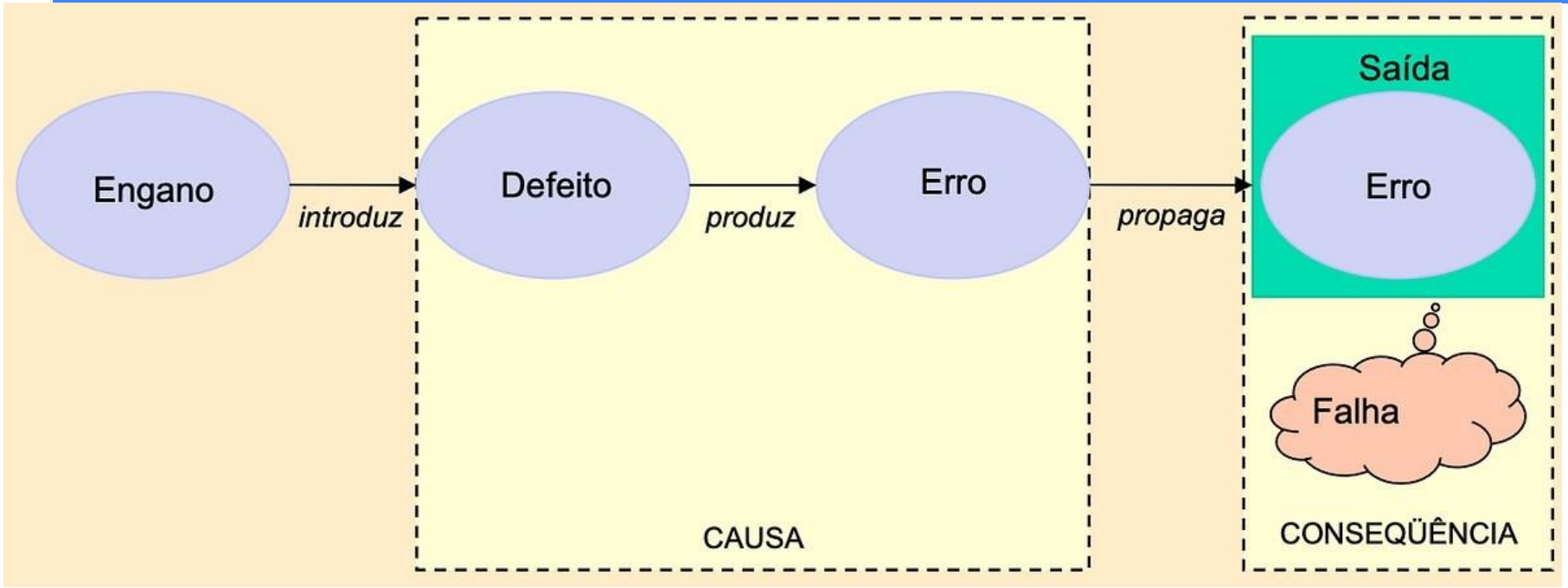
Defeito

... que cria um defeito no software...



Falha

... que pode causar uma falha na operação

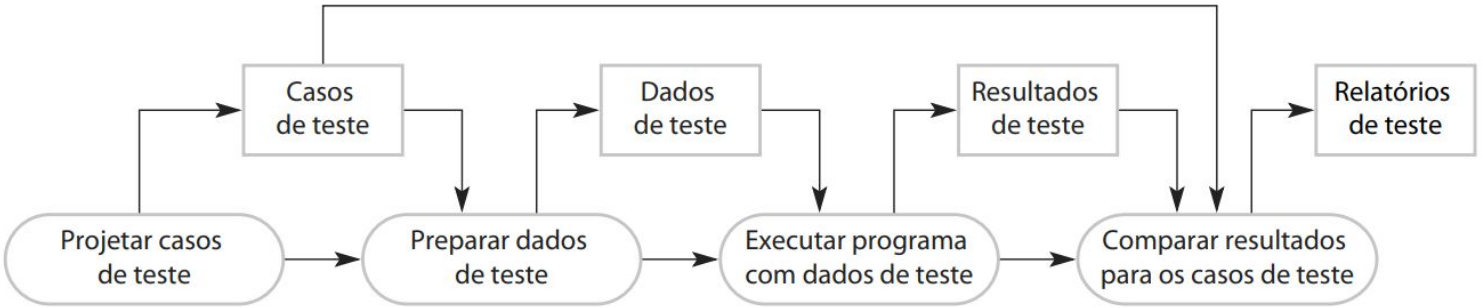


- Erro: é uma ação humana que produz um resultado incorreto.
- Defeito: A manifestação de um erro no software. Também conhecido como Bug
- Falha: quando o sistema se comporta de forma inesperada devido ao defeito

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



Artefatos de Teste	todo o conjunto de documentação gerado pelo processo de teste de software.
Caso de Teste	é composto por um conjunto de entradas, por passos de execução e um resultado esperado
Roteiro de Teste	é composto por um conjunto de casos de teste definidos para uma determinada especificação

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

Classificação dos Documentos

A Descrição dos Testes do Software reúne os documentos redigidos antes da execução dos testes, contendo os planos e especificações destes, podendo ser alterada ao longo do processo.

Os Relatórios dos Testes do Software reúnem todos os relatórios produzidos depois da execução dos testes de um projeto. Esse documento deve ser guardado sob controle, ente acrescentado à respectiva seção.

A principal referência deste padrão é:
IEEE. IEEE Std. 829 - 1983. IEEE Standard for Software Test Documentation, in (IEEE94).

Os próximos slides foram baseados no livro:

PAULA FILHO, Wilson de Pádua. *Engenharia de software: fundamentos, métodos e padrões*. 3. ed. Rio de Janeiro: LTC, 2015.

Plano de Teste

Introdução

Resumem-se aqui aspectos importantes para caracterizar os itens que serão testados na bateria. Pode-se repetir informação presente em outros documentos, focalizando aqui os interesses dos testadores.

São típicos os seguintes tópicos:

- objetivos dos testes (por exemplo, se são testes funcionais, de integração ou de unidade);
- histórico (aspectos do desenrolar do projeto que os testadores devam conhecer);
- escopo dos testes (abrangência e limites dos testes desse plano);
- referências a documentos relevantes.

Plano de Teste

Ambiente

Definem-se aqui o hardware e software das configurações usadas para o conjunto dos testes. Incluem-se também as ferramentas que serão usadas, os componentes de testes que sejam necessários e os documentos que deverão estar disponíveis para os testadores

Ambiente	Descrição
Hardware	Os testes deverão ser executados em um Pentium 133 MHz, com no mínimo 512mb de RAM instalados. Será utilizada uma impressora específica para a emissão dos tickets de venda, configurável como impressora serial.
Software	O ambiente operacional a ser utilizado é o Windows 95 (ou compatível). Com o conjunto de Atualizações KB_XXX; Deve ser utilizado o sistema de gerência de bancos de dados < nome do sistema >. E drives XXX e YYYY de comunicação serial
Ferramental	< nome da ferramenta de testes >, a ser utilizada apenas em caso de testes de regressão.

Plano de Teste

Responsabilidades

Descrevem-se as responsabilidades de cada um dos participantes dos testes desse plano.

Nº	Função	Responsabilidades
1	Testador	Planejamento dos testes de aceitação; especificação dos desenhos de teste e dos casos de teste de aceitação; realização dos testes alfa; redação do relatório resumo dos testes alfa; assessoria à elaboração do relatório resumo dos testes beta.
	Implantador	Instalação e configuração do sistema para os testes alfa e beta; assessoria ao testador na especificação dos desenhos de teste e dos casos de teste;
2	Desenvolvedor	Assessoria ao planejamento dos testes de aceitação.
3	Usuário	Realização dos testes beta; redação do relatório resumo dos testes beta.

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	

Procedimento de Teste (Roteiro)

Será descrito aqui cada um dos procedimentos de teste listados anteriormente. A descrição de cada procedimento deve conter os seguintes tópicos:

- identificação do procedimento;
- objetivo específico do procedimento;
- requisitos especiais para a execução do procedimento (por exemplo, montagens de volumes de dados);
- fluxo passo a passo do procedimento, detalhado o suficiente para que possa ser executado manualmente ou convertido em um script de teste.

Identificação	ET-001-PT-01c
Objetivo	Verificar se a consulta de usuário no banco de dados funciona corretamente
Requisitos	Banco de dados vazio
Fluxo	Abrir a interface de Tela de Usuários Acionar Novo Inserir: Nome, Login Senha. Acionar Salvar. Inserir Login. Acionar Entrar

MATRIZ DE RASTREABILIDADE

É uma tabela que mostra a relação entre os requisitos do software e os casos de teste A matriz tem o objetivo de garantir que todos os requisitos foram cobertos por testes.

ID do Requisito	Descrição do Requisito	Casos de Teste Associados
REQ-001	O sistema deve permitir a criação de novas tarefas.	CT-001, CT-002
REQ-002	As tarefas devem conter um título, descrição, data de início e data de conclusão.	CT-001, CT-003, CT-004
REQ-003	O sistema deve permitir a edição de tarefas existentes.	CT-005, CT-006
REQ-004	As tarefas podem ser classificadas por prioridade (alta, média, baixa).	CT-007, CT-008

Registro dos Testes

Dentro das entradas de atividades e eventos é registrada informação dos seguintes tipos:

• descrições de execução - início, interrupção, retomada e término de atividades de teste, indicando os responsáveis e outros participantes;

• resultados de execução - registros do sucesso ou falha dos testes, deixando-se detalhes de falhas para a descrição dos incidentes;

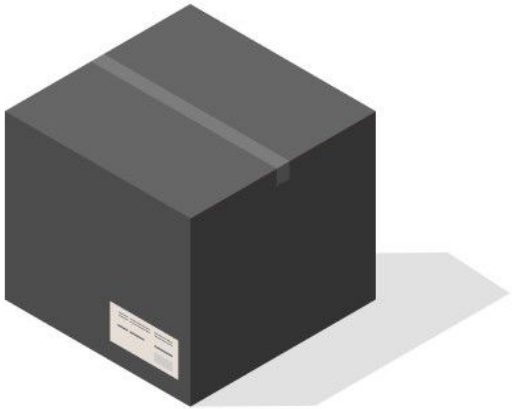
• alterações ambientais - por exemplo, mudanças no hardware ou na configuração do ambiente;

• anormalidades - eventos inesperados em relação ao planejamento e especificação dos testes.

Identificação	RLT_001	
Descrição dos Testes	Esse Registro refere-se aos Testes em Alpha de acordo com o Plano de teste PLT_001 Configuração de teste Participante: Jovino(I), Ludessa (U); Dejair (T)	
Data / hora	**Atividade / Evento**	**Incidente**
10.11.99 - 16:30	Jovino começou a instalação e configuração da máquina para os testes de Regilson	
= 19:45	Joino terminou o processo de configuração da máquina de teste	N/D
12.11.99 - 8:30	Ludessa começou a execução do caso de Teste CT_001	
= 9:30	Defeito descoberto no caso de Uso Caso de Teste Abertura do Caixa (CT_001): Dados de senha inseridos no banco como *	IC_00_CT_01
= 11:53	Ludessa finalizou os testes.	
13.12.99 10:00	Dejair finalizou o relatório de Teste	

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”.

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

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

- 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

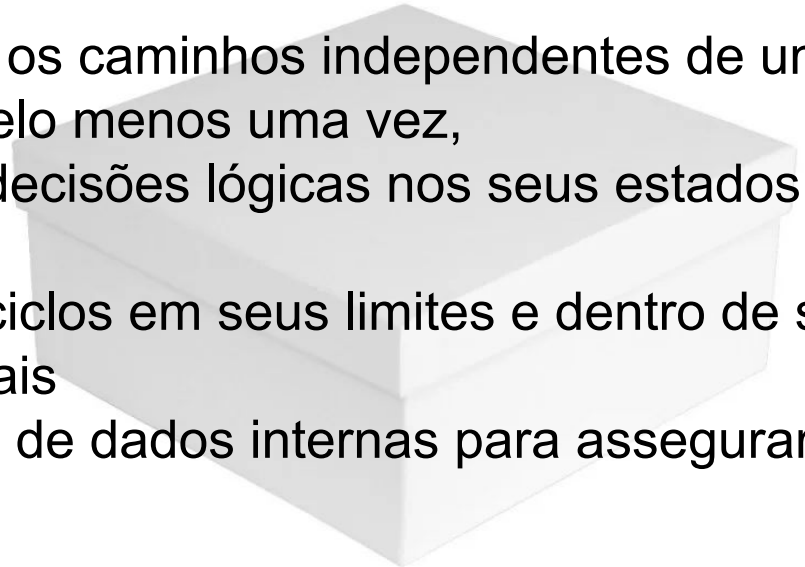
Os critérios mais conhecidos da técnica de teste funcional são Particionamento de Equivalência, Análise do Valor Limite, Grafo Causa-Efeito e Error-Guessing. Além desses, também existem outros, como, por exemplo, Teste Funcional Sistemático.

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.

Teste CAIXA BRANCA

Usando métodos de teste caixa-branca, o engenheiro de software pode criar casos de teste que:

1. garantam que todos os caminhos independentes de um módulo foram exercitados pelo menos uma vez,
2. exercitam todas as decisões lógicas nos seus estados verdadeiro e falso,
3. executam todos os ciclos em seus limites e dentro de suas fronteiras operacionais
4. exercitam estruturas de dados internas para assegurar a sua validade.

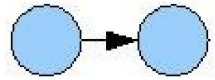


Teste Estrutural

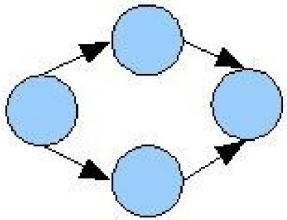
Objetivo principal é garantir que o código-fonte seja testado de maneira abrangente, com ênfase na cobertura de todas as partes do código

- Testar:
  - Instruções
  - Caminhos de execução
  - Ramificações condicionais

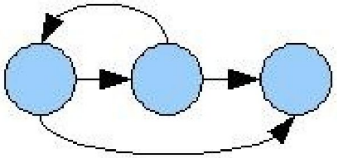
As Construções Estruturadas em forma de grafo de fluxo



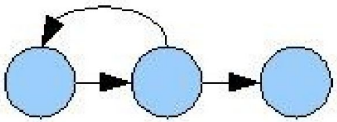
Seqüência



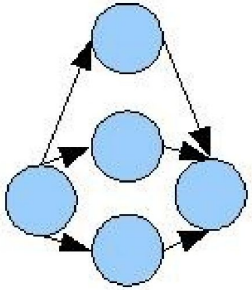
Se-então-senão



Enquanto

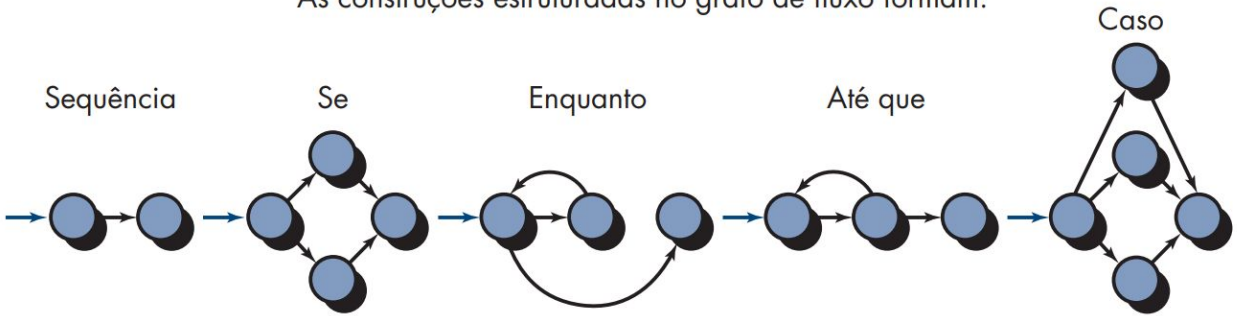


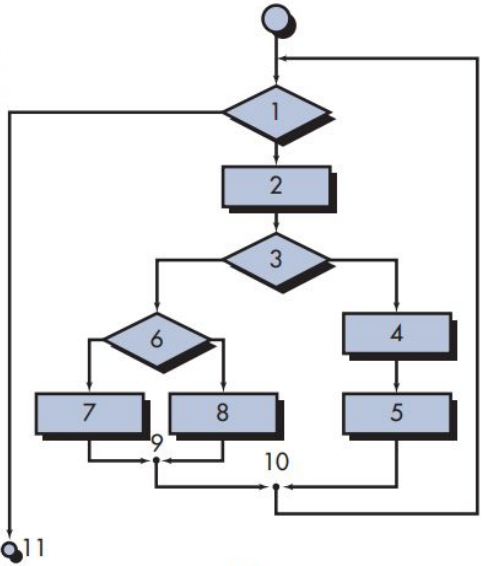
Repita



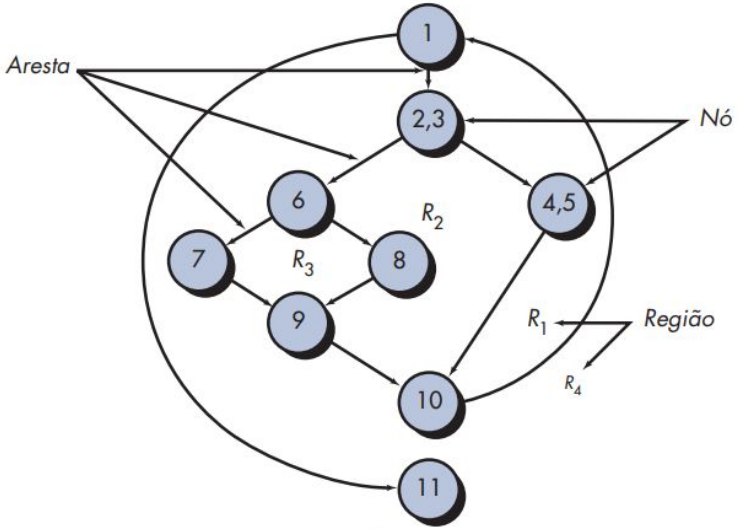
Caso

As construções estruturadas no grafo de fluxo formam:





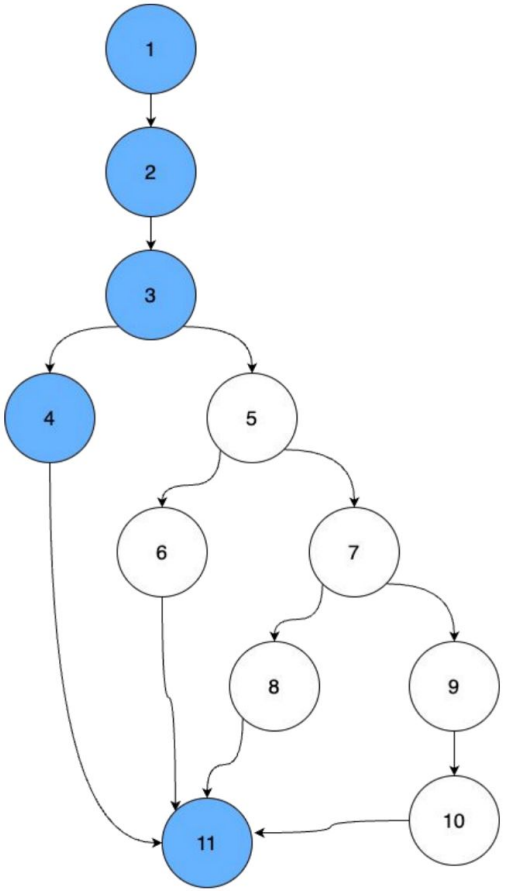
(a)



(b)

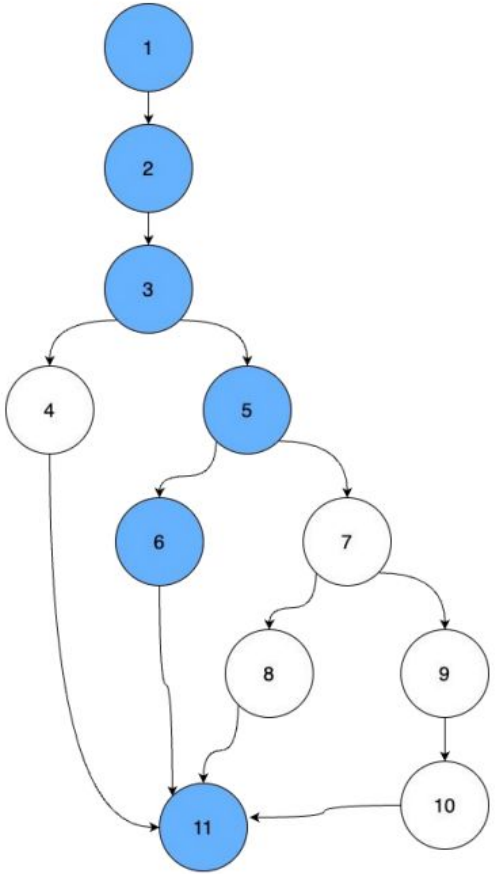
```
1. public class PnE {
2.   public static void calcularValor(int valor) {
3.       if (valor < 0) {
4.               System.out.println("Valor negativo");
5.       } else if (valor > 100) {
6.           System.out.println("Valor maior que 100");
7.       } else if (valor >= 0 && valor <= 100) {
8.           System.out.println("Valor entre 0 e 100");
9.       } else {
10.          System.out.println("Esta linha nunca será executada.");
11.      }
12.  }

13.      public static void main(String[] args) {
14.              calcularValor(-50);
15.      }
16. }
```



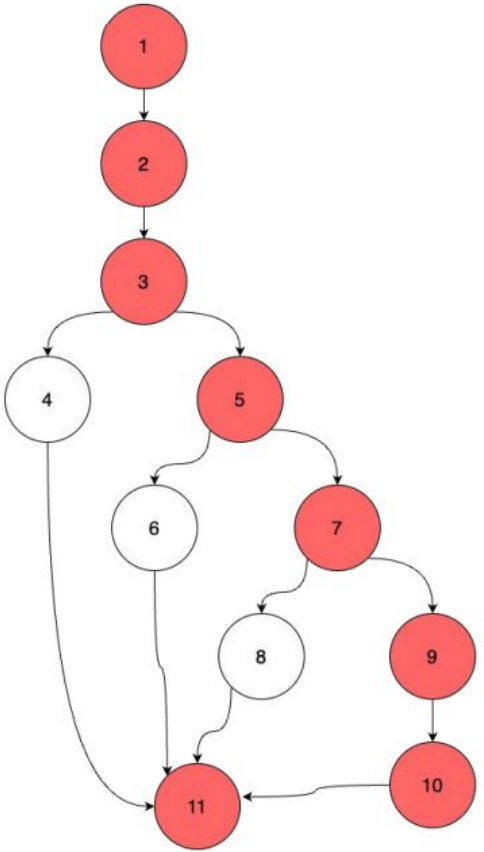
```
1. public class PnE {
2.   public static void calcularValor(int valor) {
3.       if (valor < 0) {
4.               System.out.println("Valor negativo");
5.       } else if (valor > 100) {
6.           System.out.println("Valor maior que 100");
7.       } else if (valor >= 0 && valor <= 100) {
8.           System.out.println("Valor entre 0 e 100");
9.       } else {
10.          System.out.println("Esta linha nunca será executada.");
11.      }
12.  }

13.      public static void main(String[] args) {
14.              calcularValor(101);
15.      }
16. }
```



```
1. public class PnE {
2.   public static void calcularValor(int valor) {
3.       if (valor < 0) {
4.               System.out.println("Valor negativo");
5.       } else if (valor > 100) {
6.           System.out.println("Valor maior que 100");
7.       } else if (valor >= 0 && valor <= 100) {
8.           System.out.println("Valor entre 0 e 100");
9.       } else {
10.          System.out.println("Esta linha nunca será executada.");
11.      }
12.  }

13.      public static void main(String[] args) {
14.              calcularValor(??);
15.      }
16. }
```



Teste de Caminho Básico

- Identificação de caminhos:
  - identificar todos os caminhos possíveis de execução no código-fonte.
  - Isso inclui caminhos que passam por instruções simples, estruturas de controle de fluxo, como condicionais (if/else) e loops (for/while), e qualquer outra estrutura de decisão.

Teste de Caminho Básico

- Simplificação:
  - eliminar caminhos redundantes ou irrelevantes.
- Desenvolvimento de casos de teste:
  - Com os caminhos básicos identificados, são criados casos de teste que sigam esses caminhos.
  - Cada caso de teste visa testar um caminho específico, fornecendo entradas e condições de teste apropriadas.
- Execução de testes:
  - os casos de teste são executados no programa, e os resultados são avaliados

Teste de Caminho Básico

O cálculo da complexidade ciclomática fornece a resposta. Para calcular a complexidade ciclomática de McCabe, você pode usar a fórmula a seguir:

$$V(G) = E - N + 2$$

Onde:
V(G) é a complexidade ciclomática.
E é o número de arestas no grafo de fluxo de controle.
N é o número de nós no grafo de fluxo de controle.