TESTE E QUALIDADE DE SOFTWARE

João Choma Neto

joao.choma@gmail.com

Unicesumar – Maringá – 2023/2

O QUE É TESTE DE SOFTWARE?

TESTE

Tudo o que consumimos no dia a dia é testado antes de chegar às prateleiras

Seja para garantir nossa segurança, como no caso de produtos físicos, seja para garantir nossa saúde, como no caso de alimentos

Programas de computador também são testados

TESTE DE SOFTWARE

Definição técnica:

"Teste de software é todo e qualquer **procedimento** que ajuda a determinar se o programa **atinge** as **expectativas** para as quais **foi criado** (BRAGA, 2016 apud NETO, 2010)."

Exemplo: Editor de texto

Expectativa: criar ou editar textos.

TESTE DE SOFTWARE

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

TESTE DE SOFTWARE

O principal objetivo do teste de software é assegurar a qualidade do produto final

TESTE DE SOFTWARE

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

TESTE DE SOFTWARE

Teste de software é uma das principais atividades empregadas para garantir a qualidade do software em desenvolvimento

Segundo a literatura, pode consumir mais de 50% do custo total de desenvolvimento

Padrão IEEE 610.12

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

DEFEITO

Defeito (bug ou fault): É um problema ou imperfeição no código ou na lógica do software que causa um comportamento indesejado

DEFEITO

Resultado de um código mal escrito

Implementação equivocada

Causa anomalia no funcionamento do sistema

O usuário final normalmente não vê o defeito propriamente dito

Os defeitos podem permanecer no software até que sejam descobertos e corrigidos por meio de atividades de teste

ERRO

Falha humana e produz resultado incorreto

Exemplo: a falha na escrita de um código específico

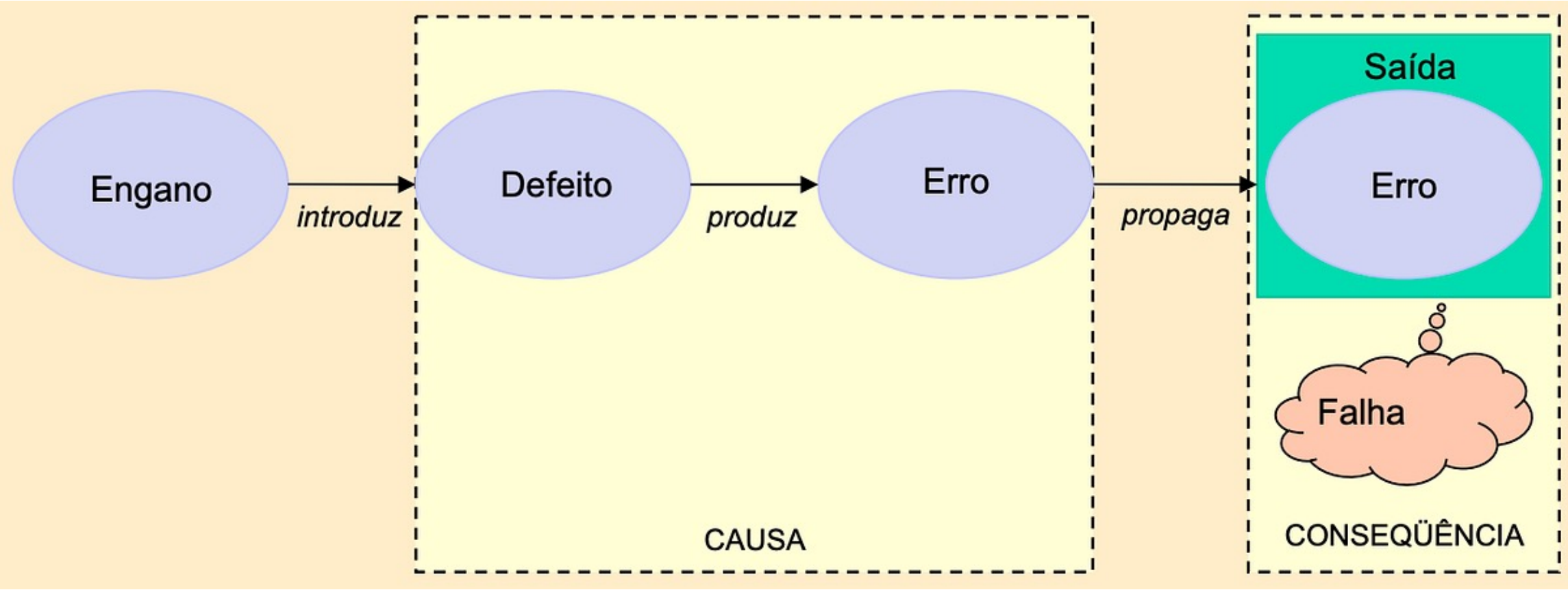
É o ato humano de cometer um engano ou incorreção durante o desenvolvimento do software

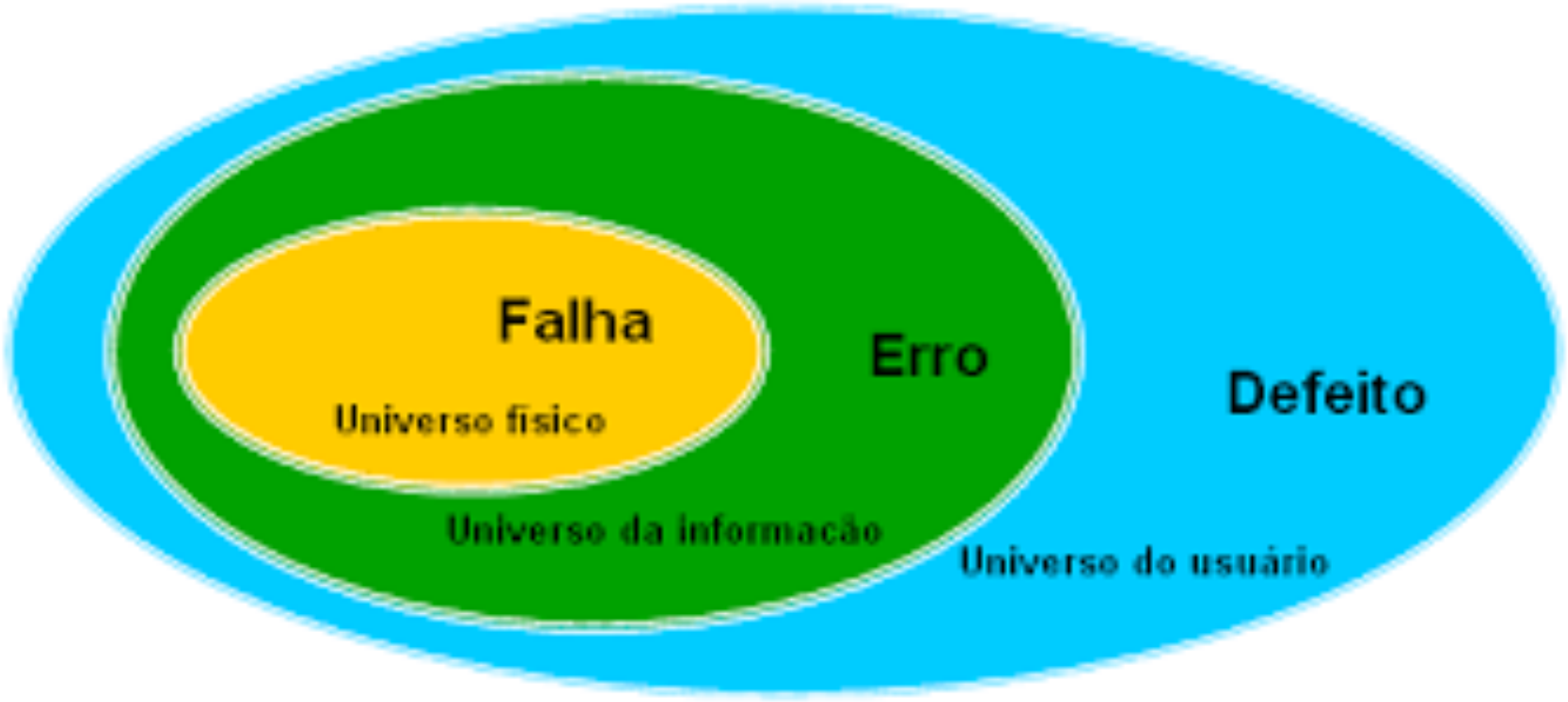
FALHA

É a ocorrência indesejada de um defeito durante a execução do software em um ambiente de produção ou em condições reais de uso

FALHA

Quando um defeito é ativado em tempo de execução, ele pode levar a falhas no software, resultando em comportamentos inesperados, travamentos, resultados incorretos ou outros problemas que afetam a funcionalidade do programa


Engano
introduz
Defeito
produz
Erro
propaga
Saída
Erro
Falha
CAUSA
CONSEQÜÊNCIA


Falha
Universo fisico
Erro
Universo da informacão
Defeito
Universo do usuário

RAZÕES PARA TESTAR

Demonstrar ao desenvolvedor e ao cliente que o software atende aos seus requisitos

Deve haver pelo menos um teste para cada requisito no documento de requisitos

Deve haver testes para todas as características do sistema que serão incluídas no lançamento do produto

REFLEXÃO

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

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

REFLEXÃO

Edsger Dijkstra (1972): ***“O teste só consegue mostrar a presença de erros, não a sua ausência.”***

ARTEFATOS

Os artefatos de teste são documentos e entregas produzidos durante o processo de teste de software

Esses artefatos têm o propósito de fornecer informações sobre o planejamento, execução e resultados dos testes

PLANO DE TESTE

É um documento que descreve a abordagem geral de teste:

1. Objetivos do teste
2. Escopo
3. Cronograma
4. Recursos necessários e
5. Critérios de aceitação

CASOS DE TESTE

São documentos que descrevem os cenários específicos que serão testados

1. Entrada de dados
2. Ações a serem realizadas
3. Resultados esperados

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

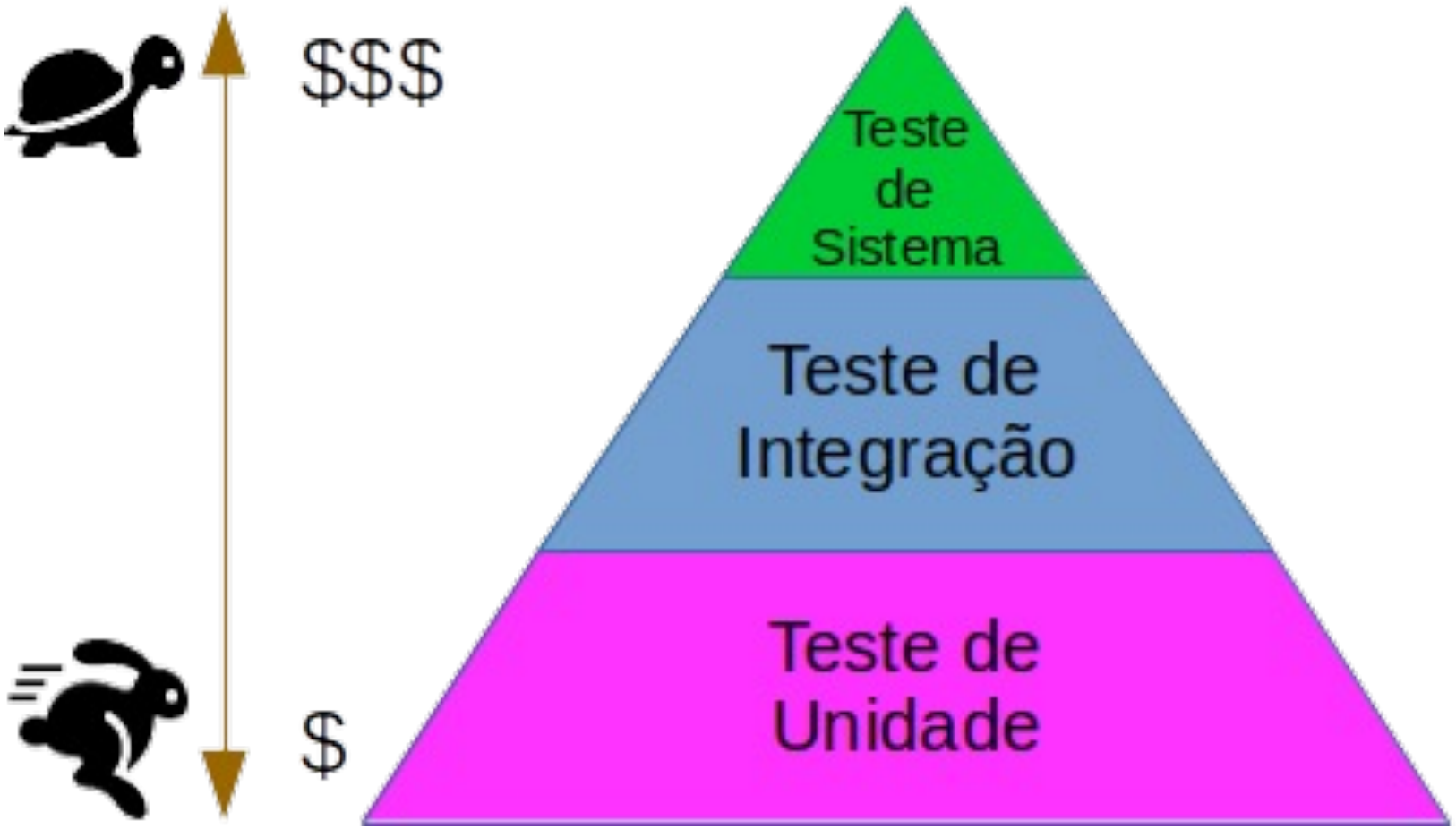
São documentos que resumem os resultados dos testes executados

1. Mostram quais casos de teste foram bem-sucedidos
2. Mostram quais casos de teste falharam
3. Medem o progresso do teste
4. Identificam quais áreas que exigem atenção do testador

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



Maior granularidade
Menor quantidade
Mais lentos
Maior custo

Menor granularidade
Maior quantidade
Mais rápidos
Menor custo

TESTE UNITÁRIO

- Teste unitário é uma prática de teste de software na qual cada componente individual ou "unidade”

TESTE UNITÁRIO

- O objetivo principal dos testes unitários é identificar defeitos nas unidades de código antes que elas sejam integradas ao restante do sistema

TESTE UNITÁRIO

- Isolar as unidades de código e testá-las independentemente
- Os desenvolvedores podem verificar se cada unidade executa as tarefas designadas de acordo com suas especificações

TESTE UNITÁRIO

- Cada teste unitário deve ser automatizado para que possa ser executado repetidamente e integrado a fluxos de trabalho de desenvolvimento contínuo

VISÃO PARA TESTAR (1)

- Segundo Pressman (2011), qualquer produto de engenharia pode ser **testado** por uma de duas maneiras:
  - Conhecendo a **função** para o qual um produto foi projetado para realizar
  - Construir testes que demonstram que cada uma das funções é totalmente operacional

CAIXA PRETA

- A primeira 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

TESTE FUNCIONAL

- Teste FUNCIONAL ou CAIXA-PRETA focaliza os **requisitos funcionais** do software
- As técnicas de teste caixa-preta permitem derivar séries de condições de entrada que utilizarão completamente **todos os requisitos funcionais** para um programa

TESTE FUNCIONAL

- 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

TESTE FUNCIONAL

- 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

VISÃO PARA TESTAR (2)

- Segundo Pressman (2011), qualquer produto de engenharia pode ser **testado** por uma de duas maneiras:
  - Conhecendo o **funcionamento interno** de um produto, podem ser realizados testes para garantir que “tudo se encaixa”
  - Construir testes que demonstram que as operações internas foram realizadas de acordo com as especificações

CAIXA BRANCA

- A segunda 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.

TESTE ESTRUTURAL

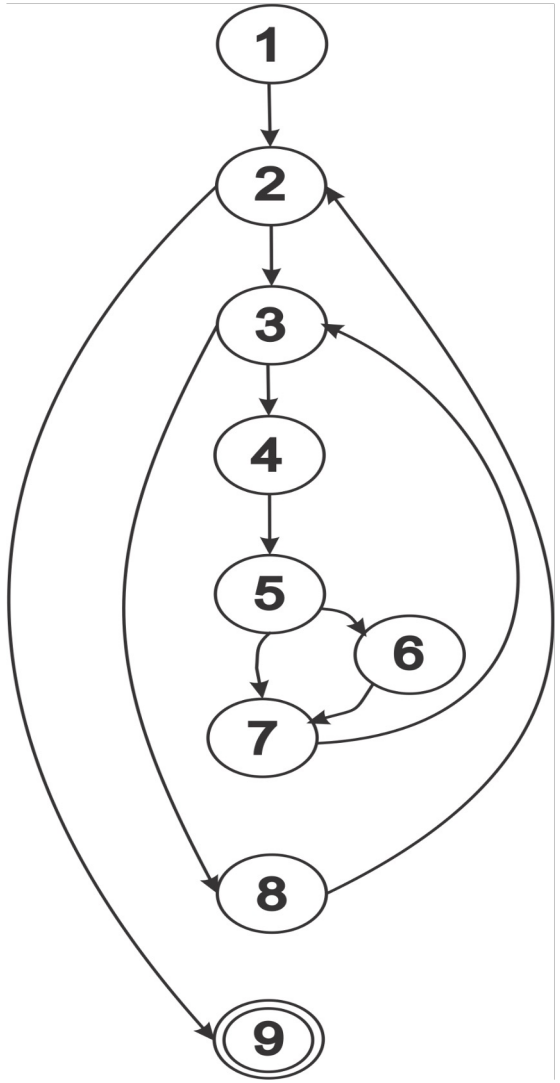
- Propriedades que devem ser avaliadas no teste

1. Baseados em **complexidade**
2. Baseados em **fluxo de controle**
3. Baseados em **fluxo de dados**

- Todo critério de teste é composto por um conjunto requisitos de teste
- Caminhos, laços de repetição, definição e uso de variáveis

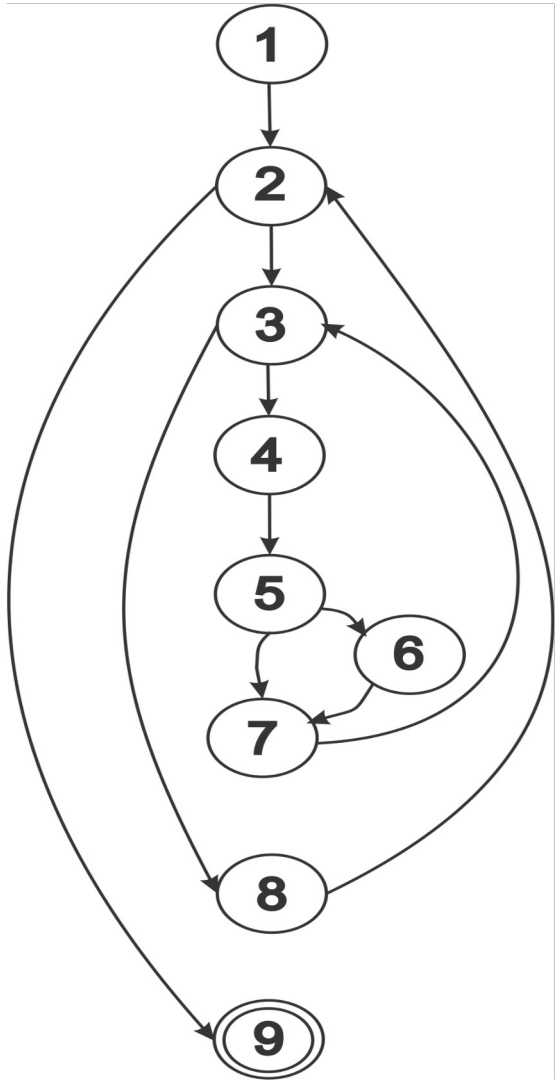
Abstração do código Grafo de Fluxo de Controle

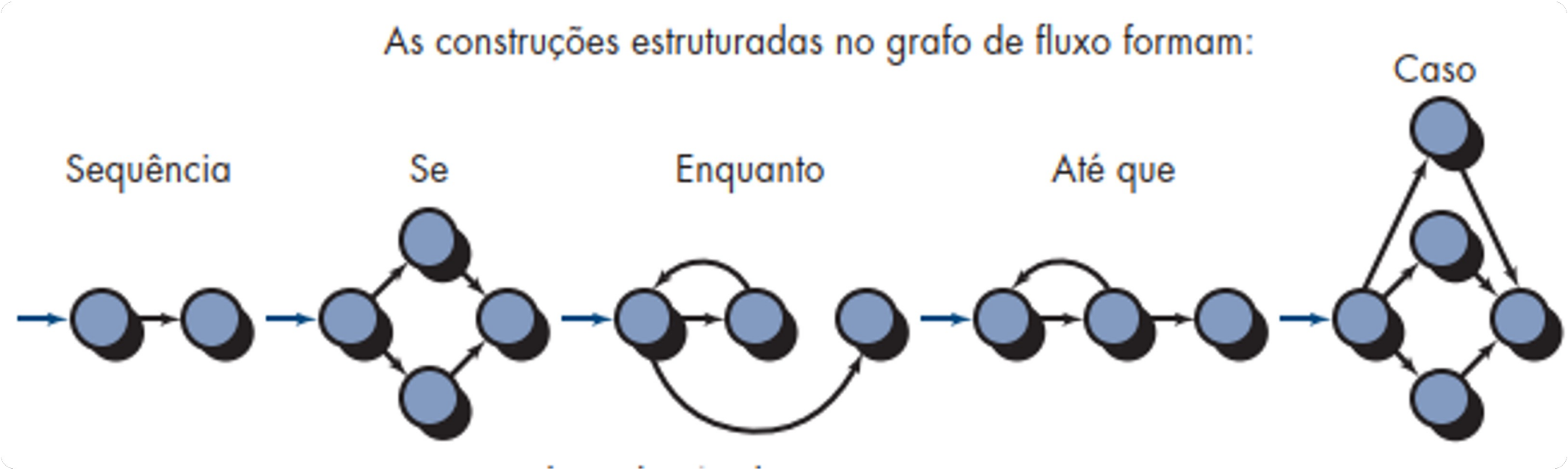
- O comportamento do código fonte de um programa pode ser representado por Grafo de Fluxo de Controle
  - Um nó corresponde a uma instrução
  - As arestas denotam o potencial fluxo de controle entre as instruções



FLUXO DE CONTROLE

- É a sequência de passos que o computador segue para executar as operações do programa
  - Sequência
  - Condicionais
  - Estruturas de repetição





TESTE DE CAMINHO BÁSICO

- O **grafo de fluxo** representa o fluxo de controle lógico.

Quantos caminhos procurar?

- 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.

REFERÊNCIAS

Ian Sommerville – Engenharia de Software. 10ª Edição. São Paulo: Pearson Education do Brasil, 2019.

Roger S. Pressman – Engenharia de software: uma abordagem profissional. 7ª Edição. Porto Alegre: AMGH Editora Ltda, 2011.

Shari Lawrence Pfleeger – Engenharia de Software: teoria e prática. 2ª Edição. São Paulo: Pearson Education do Brasil, 2004.

TESTE E QUALIDADE DE SOFTWARE

João Choma Neto

joao.choma@gmail.com

Unicesumar – Maringá – 2023/2