

# Testes de Software

## Aula 04

**Conceitos Introdutórios**

# Senac

# Apresentação



## Objetivos

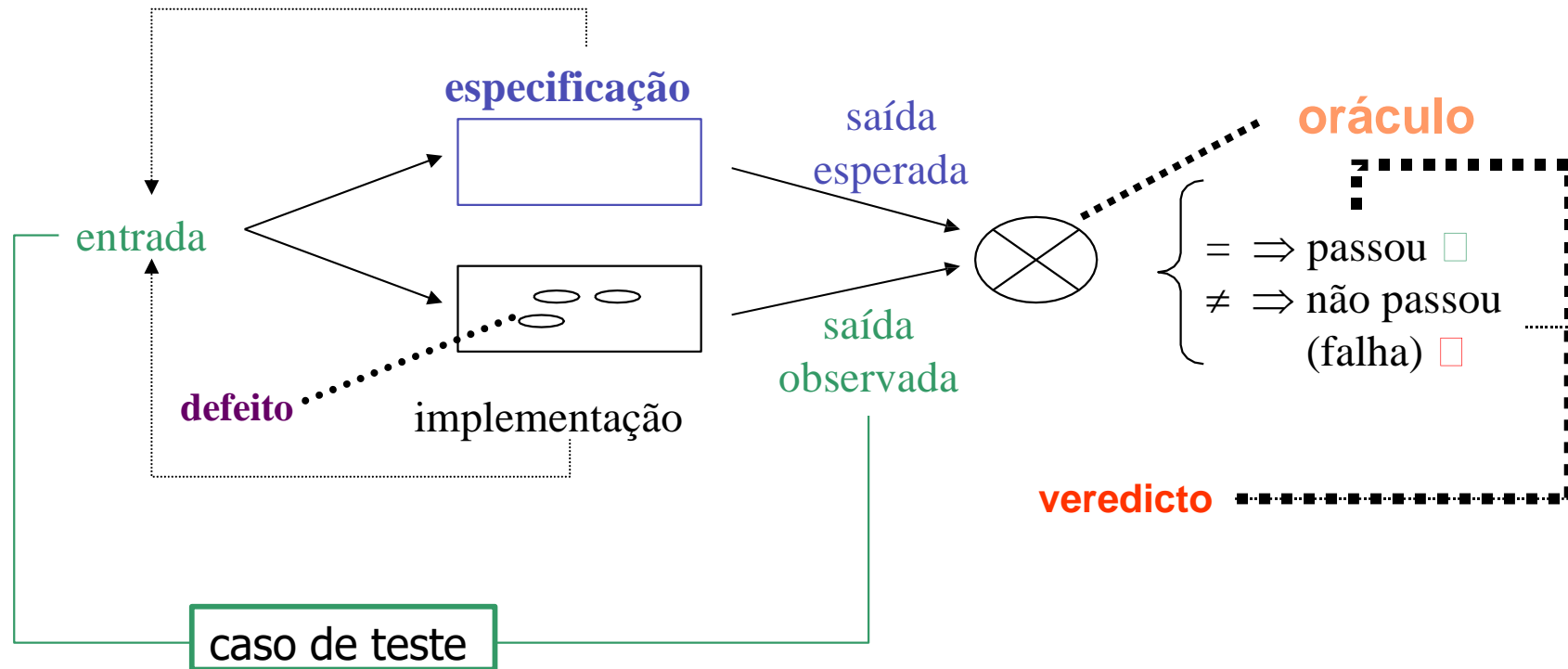
- Compreender o objetivo do projeto de testes e os desafios que o projetista de testes deve vencer.
- Saber quais as condições que devem ser satisfeitas para que os testes sejam reveladores de defeitos
- Tipos de técnicas
- Conhecer a importância de modelos e critérios para os testes

## Tópicos

- Visão esquemática
- Desafios
- Condições para revelação de defeitos
- Técnicas Não Estruturadas
- Técnicas Estruturadas
  - Importância de modelos e critérios



# Projeto de Testes: visão esquemática



# Desafios para o projeto de testes: quais entradas?



- Testes exaustivos são inviáveis!



- O problema:
  - Todo número ímpar é primo?
- Os testes:
  - Testador 1: entrada: 9  
saída: não é primo
  - Testador 2: entradas: 3, 5, 7, 13  
saída: são primos

Geração e seleção de casos de teste:  
qual subconjunto de dados de entrada  
escolher? qual dado de entrada produz  
falha?



# Desafios para o projeto de testes: quais as saídas esperadas?



- O problema:
  - Todo número ímpar é primo?
- Os testes:
  - Testador 1: entrada: 9  
saída: não é primo
  - Testador 2: entradas: 3, 5, 7, 13  
saída: são primos
  - Testador 3: entradas: 3, 5, 9, 11  
saída: são primos

**Problema do oráculo:**  
qual a saída esperada?



# O problema do oráculo

- Oráculo:
  - Mecanismo que avalia se um caso de teste **passou** (□) ou **falhou** (□).

- **Oráculo perfeito** pode ser construído? 

Implementação **sem defeitos** que, para todas as entradas possíveis, responde **sempre** corretamente !!!



O termo software **correto** não faz sentido



# Importância do oráculo

- Um oráculo falho ...

Saídas esperadas incorretas  
Baixa observabilidade

... pode levar a:

- Falsos negativos:

- Item **falha** mas passa nos testes □ não realiza as correções **necessárias**

- Falsos positivos:

- Item **fornece resposta esperada**, mas não passa nos testes □ realiza depuração **desnecessárias**



IeT e Oráculo				Caso de teste		Resultados			Validade do teste
Item em Teste		Oráculo		x	y	Certo	Esperado	Obtido	
Correto	$z = x + y$	Correto	$x + y$	5	3	8	8	8	<input type="checkbox"/> Válido
		Falho	$x * y$	5	3	8	15	8	<input type="checkbox"/> Inválido (falso positivo)
				2	2	4	4	4	<input type="checkbox"/> Inválido (coincidência) <input type="checkbox"/> falso negativo
Falho	$z = x - y$	Correto	$x + y$	5	3	8	8	2	<input type="checkbox"/> Válido
				0	0	0	0	0	<input type="checkbox"/> Inválido (coincidência) falso negativo
	$z = x / y$	Falho	$x / y$	5	3	8	0,6	0,6	<input type="checkbox"/> Inválido falso negativo
	$z = x * y$		$x - y$	5	3	8	2	15	<input type="checkbox"/> Inválido

Base: [Binder 2000, cap 18]

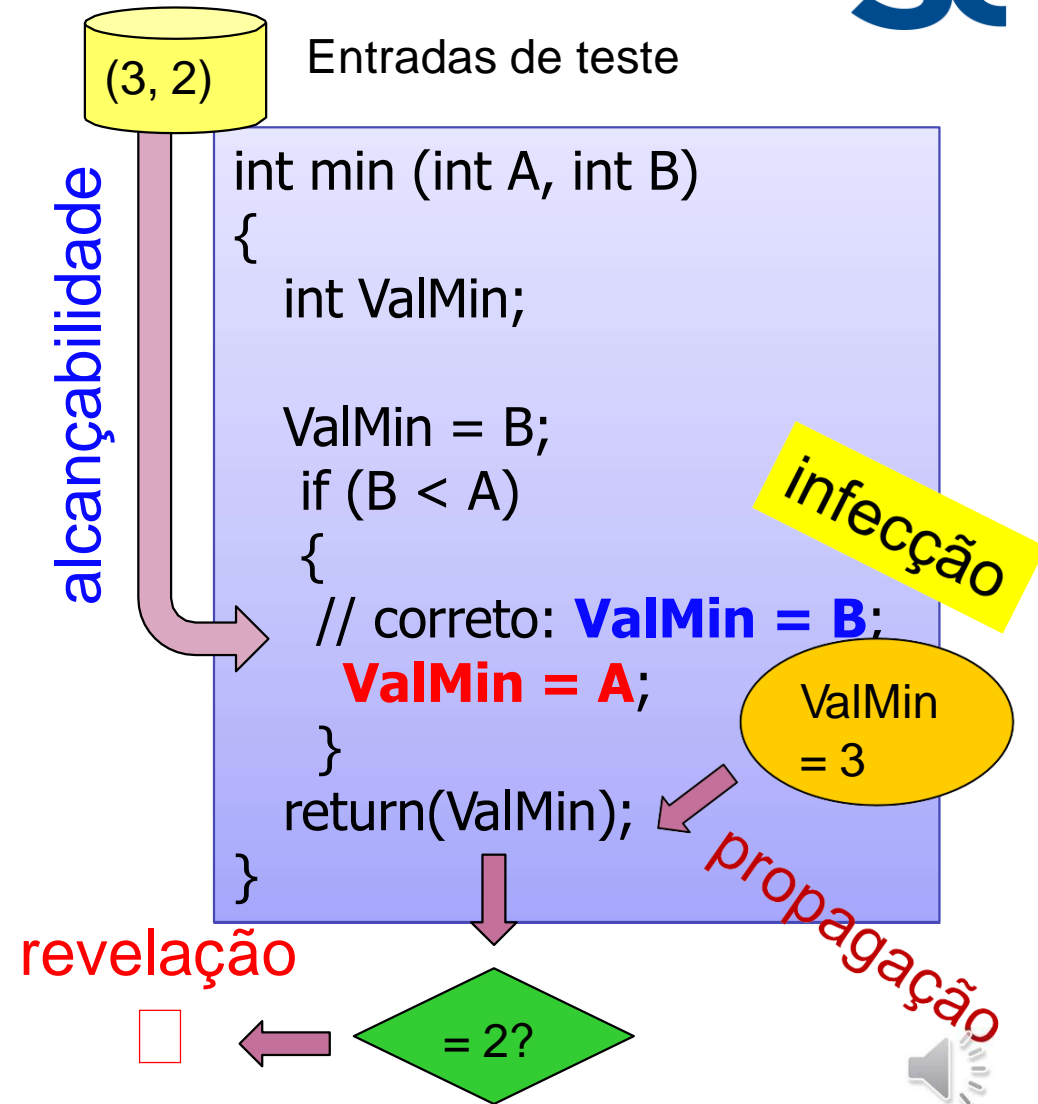




# Condições para revelação de defeitos

- Condições para que um caso de teste cause falhas:
  - Alcançabilidade (**R**eachability)
  - Infecção (**I**nfection)
  - Propagação (**P**ropagation)
  - Revelação (**R**evealability)

[Ammann e Offutt 2016, cap. 2]



# Objetivo do Projeto de Testes

- Objetivo

- buscar as entradas (e estados) que irão **alcançar** e **ativar** os **defeitos** (faults) existentes, gerando **erros** que irão se **propagar** até as saídas, **levando à ocorrência** de **falhas** (*failures*)

- Dificuldade

- como selecionar esse **subconjunto finito**?



- Diversas técnicas



# Tipos de Técnicas

## Não Estruturadas



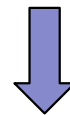
Ad-hoc



Informal

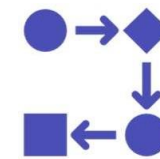


Heurísticas



Baseadas na intuição  
e criatividade  
humanas

## Estruturadas



Caixa Branca

Caixa Preta

...



Baseadas em  
modelos e  
critérios





# TÉCNICAS HEURÍSTICAS

<https://www.satisfice.com/blog/archives/5222>



# Técnicas Não-Estruturadas

**Criação ou seleção de testes se baseia na experiência, no conhecimento ou na intuição do testador**

[Marchetti e Bertolino 2004]

Para James Bach:

**técnicas não-estruturadas** (depende do ser humano)

x

**técnicas estruturadas** (depende de algoritmos)

A decorative graphic consisting of overlapping squares in blue, purple, and green, with a black crosshair-like structure.

# Suposição de erros

---

- Também conhecida como *errorguessing*
- Geração ou seleção casos de testes com base na **antecipação de defeitos** que podem estar presentes no item em teste:
  - Defeitos que ocorreram antes
  - Intuição
  - Experiência
  - ...

# Testes exploratórios

[Cem Kaner 2008]

Criadores: James Bach e Cem Kaner (anos 90)

É um estilo de testar software,

que enfatiza a liberdade e responsabilidade pessoal do testador,

para melhorar continuamente o seu trabalho.

- As atividades de:
    - **Aprendizado** relacionado aos testes
    - **Projeto** de testes
    - **Execução** dos testes
    - **Análise** dos resultados dos testes
- se apoiam mutuamente e
- podem ser realizadas em paralelo durante o projeto



# Testes exploratórios

## Quando usar

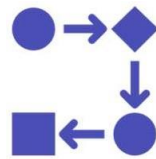
- Quando se sabe pouco sobre o produto
  - Documentação de baixa qualidade ou inexistente
  - Criação e seleção de testes ocorrem dinamicamente, enquanto o testador aprende sobre o sistema
- Quando não há um processo de testes definido
- Para complementar testes já realizados
- Para estudar um produto e usar a informação para criar testes melhores

## Limitações

- Depende da habilidade do testador para criar situações normais e situações de erro
- Não há planejamento dos testes a realizar
- Os testes não são documentados
- Testes dificilmente são reutilizados







# TÉCNICAS ESTRUTURADAS



# Técnicas estruturadas


- Usam modelos explícitos para guiar o projeto de casos de teste

[C.Kaner 2008]

Modelo: "**representação simplificada** de uma **relação**, de um **processo** ou de um **sistema**."

- Por que usar um modelo?
  - Visão mais abstrata □ omite detalhes que não interessam
  - Maior clareza sobre os aspectos que interessam
  - Maior facilidade para entender e manipular



A decorative graphic consisting of overlapping squares in blue, purple, and green, with a black crosshair-like structure.

# Testes guiados por modelos

---

- Termo dado por Amman & Offutt [2016, cap2]
  - Complementa o Desenvolvimento Guiado por Modelos (MDD)
  - Testador pode usar diferentes modelos:
    - Baseado em artefatos produzidos no desenvolvimento
    - Baseado em aspectos interessantes de serem testados
- Modelos podem representar diferentes escopos de testes:
  - Unidade
  - Integração
  - Sistemas
  - ...
- As técnicas para seleção de casos são as mesmas. O que muda?
  - A semântica dos modelos
    - estrutura do código; estrutura do sistema; comportamento de um componente; comportamento de um sistema

# Modelos mais comuns

[Amman & Offutt 2016, cap2]

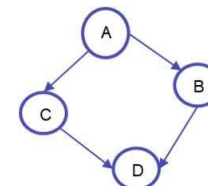
## Domínio de entradas

- interface

A: {0, 1, >1}  
SOp: {Android, Windows, MacOS}

## Grafos

- estrutura do código
- comportamento
- dependências entre módulos
- ...



## Expressões lógicas

- código
- requisitos

A = B || C  
E = A & D

## Sintaxe

- entradas estruturadas
- código

banco ::= transação\*  
transação ::= dep | saq  
dep ::= "depósito" conta quantia

## Defeitos

- código
- modelos
- ...

```

Ler x;
Se x mod 2 == 0
Então y := x + x/2 // original y := x - x/2
...
  
```

# Classificação das Técnicas (antiga)



## Testes Caixa Branca

- Modelos baseados na estrutura interna do código

## Testes Caixa Preta

- Modelos baseados na especificação: especificação usando modelos ou em língua natural, manual do usuário, entre outras

## Testes Baseados em Defeitos

- Modelos de defeitos que podem surgir ao longo do desenvolvimento



# O problema da seleção

## Especificação

**Nome:** Consultar Saldo  
**Descrição:** O cliente, já autenticado, escolhe a opção “Consultar Saldo” e o sistema apresenta o seu saldo.  
**Autores:** eu

**Nome:** Consultar Saldo  
**Descrição:** O cliente, já autenticado, escolhe a opção “Consultar Saldo” e o sistema apresenta o seu saldo.  
**Autores:** eu  
**Atores:** Cliente, Cadastro de contas do banco  
**Localizações:** caixas eletrônicos em diversas cidades  
**Posição:** caminho primário definido  
**Prioridade:** 1  
**Pré-condição:** O cliente já deve ter sido validado pelo caso de uso **Autenticação**.  
**Pós-condição:** O saldo é emitido. O cartão é devolvido. O estado da conta permanece inalterado  
**Fluxo básico:** 1. Se cliente não autenticado então executa o fluxo de exceção E1. 4. Se não há conexão com o banco, execute o fluxo de exceção E2. 5. O saldo é consultado e emitido. 6. Devolve cartão.  
**Fluxos alternativos:** A1. cliente fornece dados do cartão pelo teclado.  
**Fluxos de exceção:** E1: cliente não autenticado: E1.1 Emite mensagem: cliente não autenticado. E1.2. Devolve cartão. E2: sem conexão com banco: E2.1. Emite mensagem: “Tente mais tarde”. E2.2 Se (cliente com cartão) então o cartão é devolvido.

Quais entradas selecionar?

Como saber se o item foi bem testado?



# Critérios de teste

- Um **critério** de teste C determina um conjunto finito de elementos do modelo de teste que devem ser exercitados durante os testes
- **Requisito de teste**: elemento do modelo de teste que precisa ser coberto de acordo com o critério

Ex.:

modelo de base = casos de uso

**critério** ( C ) = todos os casos de uso devem ser exercitados ao menos 1 vez

**requisitos de teste** (  $RT_C$  ) = { todos os casos de uso do diagrama }



# Cobertura de testes

- Medida do quão completo é um conjunto de testes ( $T$ ) com relação ao critério adotado
- É dada na forma de uma proporção:

$$\frac{\text{nº de requisitos de teste (RT}_C\text{) exercitados}}{\text{nº total RT}_C}$$





# Sobre nível de cobertura

[Ammann e Offutt 2016, cap 5]

- É possível obter 100% de cobertura?
  - Custo do critério
    - Grande nº de casos de teste
  - Requisitos de teste infactíveis
    - Requisitos que não podem ser satisfeitos por nenhum caso de teste
- Qual nível de cobertura é adequado?
  - 90%? 85%? 75%? □ revelam defeitos?



# Por que usar critérios de teste?



- Criação de subconjuntos de teste menores e mais eficazes do que os criados de forma ad-hoc
- Critérios são derivados a partir de modelos, que representam artefatos de software □ rastreabilidade + fácil
- Resposta à questão: quando terminam os testes?



# Critérios e Automação

- Uso de modelos e critérios □ automação do projeto de testes:
  - **Geração** automática de casos de teste que satisfaçam aos critérios
  - **Análise** automática **da cobertura** de critérios
- Dificuldade da geração: requisitos de teste infactíveis
  - Problema **indecidível**



# Sumário

- Projeto de Testes:
  - processo de produzir entradas que irão testar o software
- Principais desafios:
  - Como criar conjunto finito de entradas que revelem defeitos?
  - Como determinar resultado esperado?
- Projeto se baseia em modelos que representem aspectos a serem testados. Há várias técnicas:
  - Não estruturadas: usam modelo mental do/a testador/a
  - Estruturadas: usam modelo explícito e critérios que ajudam a determinar o que testar
- Técnicas estruturadas possibilitam a automação do Projeto de Testes



# Alguns termos

Itens de teste	partes do software que serão testadas
Modelo de teste	versão simplificada de aspectos de interesse do item em teste
Critério de teste	condição que conjunto de testes deve satisfazer
Requisitos de teste	Conjunto finito de elementos do modelo de testes que se quer exercitar
Caso de teste	Entradas + saídas esperadas
Conjunto de testes	Composto de vários casos de teste
Dados de teste	Valores de entrada usados para instanciar os casos de teste
Veredicto de testes	Resultado da execução dos testes (passou/falhou)
Incidentes de testes	Evento ou comportamento ocorrido que difere do esperado e que precisa ser analisado

## Referências

- Ammann & Offutt. "Introduction to Software Testing". 2016, cap 2.
- Brian Marick. "The Craft of Software Testing". Prentice Hall Inc., 1995.



# Mais referências

- Sobre oráculos de teste
  - Robert Binder.  
*Testing object-oriented systems: models, patterns, and tools*. Editora Addison-Wesley Professional, 2000. Cap. 18
  - Howden, W.E. (July 1978). "Theoretical and Empirical Studies of Program Testing". *IEEE Transactions on Software Engineering*. **4** (4): 293–298.
  - Weyuker, Elaine J.; "The Oracle Assumption of Program Testing", in *Proceedings of the 13th International Conference on System Sciences (ICSS), Honolulu, HI, January 1980*, pp. 44-49.



## E mais algumas ...

- Sobre testes exploratórios:
  - Bach, James. "Exploratory testing explained." (2003).  
<https://people.eecs.ku.edu/~hossein/Teaching/Fa07/814/Resources/exploratory-testing.pdf> acesso em mar/2022
  - Kaner, Cem. "A tutorial in exploratory testing." Tutorial presented at QUEST2008  
<http://www.kaner.com/pdfs/QAIEExploring> Acesso em mar/2022





