

Engenharia de Software II

Verificação e Validação

Profa. Thaiana Pereira dos Anjos Reis, Dra. Eng.
thiana.anjos@ifsc.edu.br

Agenda

- Verificação e Validação
- Definição dos Conceitos
- Testes de Software

Qual a diferença entre Verificação e Validação?

Verificação

Visa assegurar que o software seja desenvolvido de um modo apropriado e consistente.

“Estamos construindo corretamente o produto?”

Validação

Visa assegurar que o software corresponda aos requisitos estabelecidos.

“Estamos construindo o produto certo?”

Verificação

Objetivo: mostrar que o software atende a sua especificação.



Validação

Objetivo: mostrar que o software atende as necessidades do cliente.



É um processo aplicado para **melhorar a qualidade** dos produtos e a **produtividade** dos processos de software.



Permite aos desenvolvedores **identificar problemas** de software o **mais rápido possível** e corrigi-los antes de entregarem aos usuários.



Permite identificar e corrigir problemas nas atividades de desenvolvimento para **aumentar a produtividade de novos projetos** de software.



Ocorre durante TODO o ciclo de vida do software



DEFINIÇÃO DOS CONCEITOS

Definição dos Conceitos

Requisitos incompletos e incorretos indicam problemas no produto que podem se manifestar por meio de falhas, caso não sejam corrigidos, durante a operação e uso do software.

A IEEE descreve várias definições de problemas que podem ocorrer:



Erro (*error, mistake*)

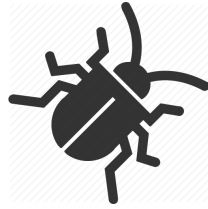


Defeito (*bug, fault, defect*)



Falha (*failure*)

(IEEE 610.12, 1990)

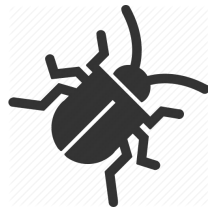


DEFEITO

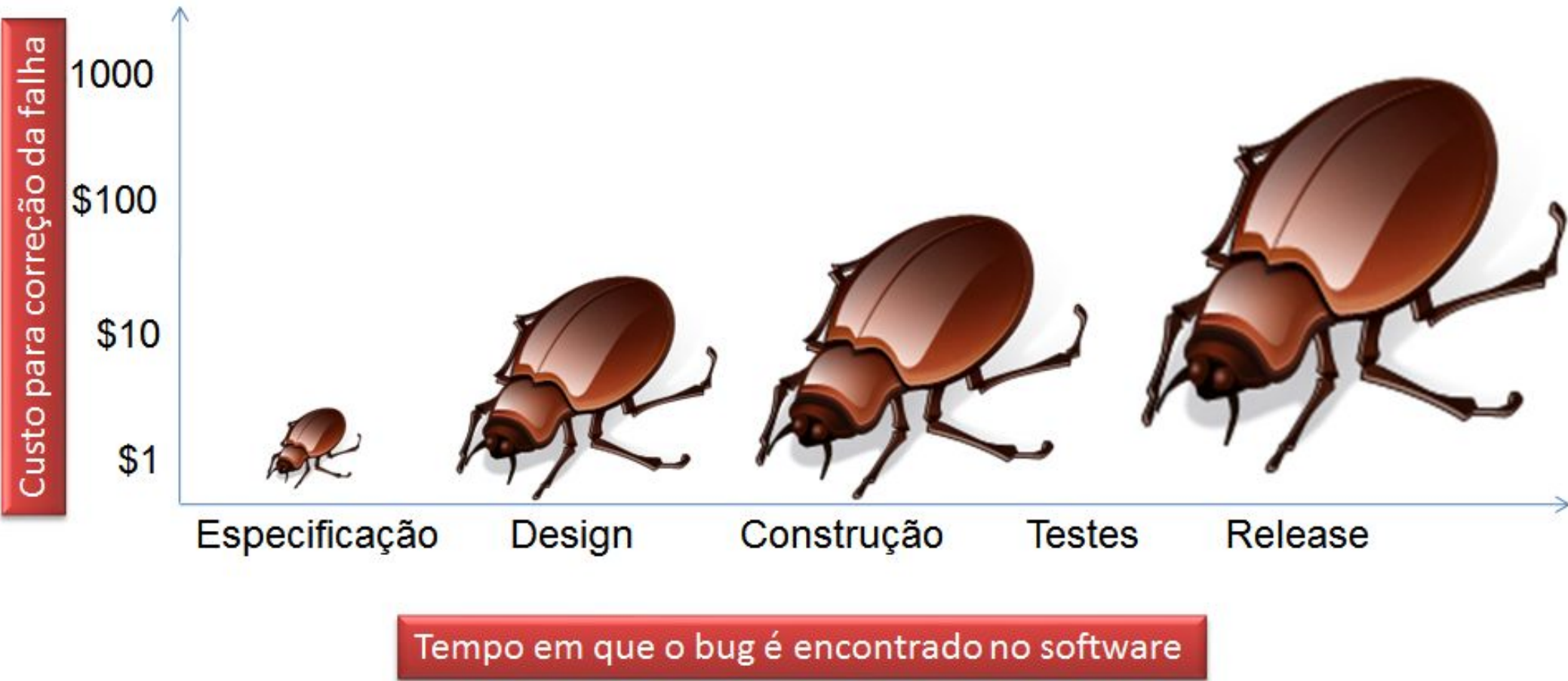
Uma ato inconsistente cometido por um indivíduo ao tentar entender uma determinada informação, resolver um problema ou utilizar um método ou uma ferramenta.

Exemplos:

- Ele ocorre na documentação, como a definição errada de requisitos ou relacionamentos incorretos nos diagramas de classe.
- Ele ocorre em uma linha de código, como uma instrução errada ou um comando incorreto.



DEFEITO





ERRO

Uma manifestação concreta de um defeito num artefato de software

Exemplos:

- Qualquer estado intermediário incorreto ou resultado inesperado na execução de um programa constitui um erro.
- Diferença entre o valor obtido e o valor esperado, ou seja,



FALHA

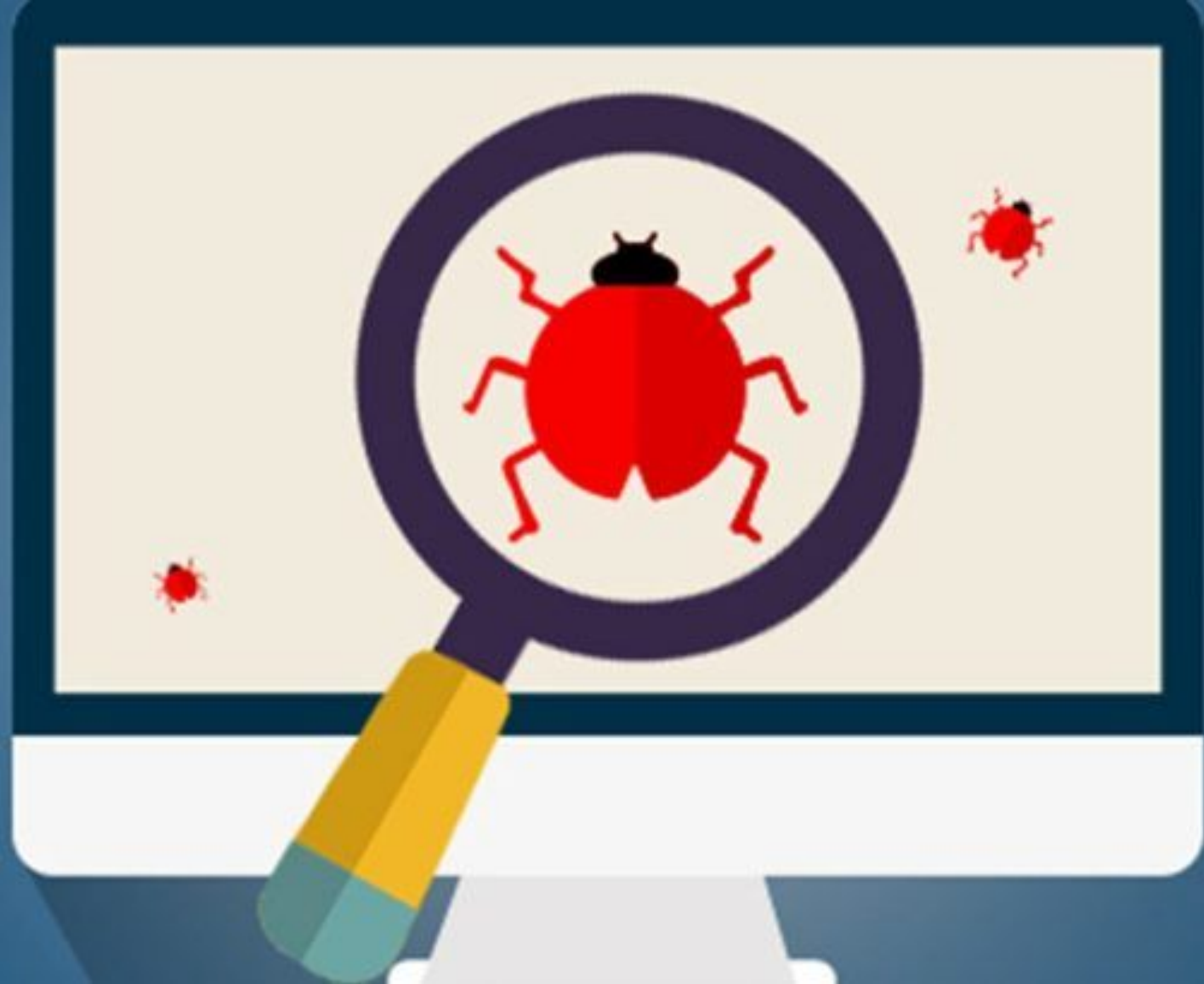
A incapacidade de um sistema ou componente em executar as funções requeridas dentro de um nível de desempenho requerido. É o comportamento operacional do software diferente do esperado pelo usuário.

Exemplos:

- Uma falha pode ter sido causada por diversos erros e alguns erros podem nunca causar uma falha.
- Tela azul no monitor do computador
- Retorno de um valor não esperado, como null, isso é um erro, e por causa desse null ocasionou uma falha no sistema.

Definição dos Conceitos





TÉCNICAS DE V&V

Técnicas de V&V

Podem ser divididas em **estáticas** e **dinâmicas**.

Técnicas de V&V

1 ESTÁTICAS

Nas técnicas estáticas, a avaliação de um produto de software é realizada por um **grupo de revisores**, com intuito de identificar defeitos. Elas podem ser:

a) Revisões Técnicas e Walkthroughs: Indicadas para todas as fases do ciclo de desenvolvimento de software.

b) Inspeções: Indicadas para a fase de codificação.

Técnicas de V&V

1 ESTÁTICAS

a) Revisões Técnicas e Walkthroughs: Indicadas para todas as fases do ciclo de desenvolvimento de software.

- **Revisões Técnicas** permitem avaliar um produto de software para determinar a sua adequação ao uso pretendido. Identificar discrepância entre especificações e padrões aprovados.

- **Walkthroughs** permitem também avaliar um produto de software. O objetivo é identificar anomalias, melhorar o produto, considerar alternativas de implementação e avaliar conformidade a padrões e especificações.

Técnicas de V&V

1 ESTÁTICAS

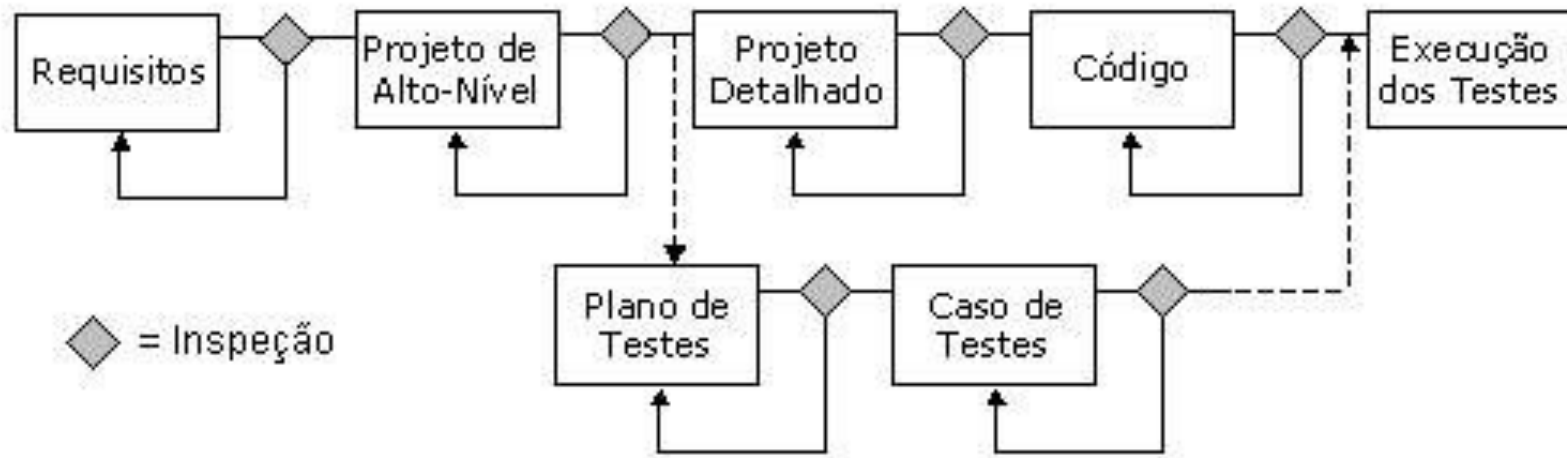
b) Inspeções

- Inspeções têm por objetivo detectar e identificar anomalias no produto de software.

As técnicas de inspeções de software são/devem ser aplicadas em todas as fases do ciclo de desenvolvimento de software.

Isso justifica-se porque os defeitos podem ocorrer em códigos, arquitetura, requisitos, especificações e documentação em geral.

Ocorre durante TODO o ciclo de vida do software



Técnicas de V&V

1 ESTÁTICAS

b) Inspeções

- Inspeções têm por objetivo **melhorar** a **qualidade de artefatos** de software através de sua análise, detectando e removendo defeitos **antes** que o artefato seja passado para a próxima fase do processo de desenvolvimento de software.

A inspeção, quando realizada no **início do desenvolvimento** do software, leva à detecção de **60% a 90% dos defeitos** potenciais em um projeto de software.

**A inspeção aumenta a produtividade em 30% a 50%
(GILB e GRAHAM, 1993)**

**Inspeções reduzem o tempo de desenvolvimento
de 10% a 30%
(GILB e GRAHAM, 1993)**

Inspeções de código podem **reduzir** os **custos**
de implementação de projetos em 39%

Técnicas de V&V

1 ESTÁTICAS

b) Inspeções

Um grupo de inspeção (3 a 8 participantes) envolve desenvolvedores de software, entre outros participantes, em um processo formal de investigação.

- **Autor** que é o desenvolvedor do produto a ser inspecionado;
- **Moderador** que é o membro da equipe que lidera a inspeção, programa e controla as reuniões;
- **Redator** que é aquele que tem como função relatar os defeitos.

Técnicas de V&V

1 ESTÁTICAS

b) Inspeções - ETAPAS

O processo de inspeção foi descrito primeiramente por Michael Fagan e é composto por seis fases:

- Planejamento
- Apresentação
- Preparação
- Reunião de Inspeção
- Retrabalho
- Acompanhamento.

Inspeções

Planejamento - Os inspetores são selecionados e os materiais a serem revisados são preparados.

Apresentação - O grupo recebe instruções essenciais sobre o material a ser inspecionado, especialmente sobre o que deve ser inspecionado.

Preparação - Os Integrantes do time de inspeção se preparam para desempenhar o papel designado a cada um.

Reunião - Os defeitos são encontrados, discutidos e categorizados.

Retrabalho - O autor do documento corrige os defeitos encontrados pelo time de inspeção.

Acompanhamento - O time de inspeção é responsável por assegurar que todos os defeitos encontrados foram corrigidos e nenhum outro tipo de defeito foi introduzido na fase de Retrabalho. Pode ser realizado somente pelo moderador.

Técnicas de V&V

TIPOS de DEFEITOS

Técnicas de V&V

TIPOS de DEFEITOS

Durante o processo de inspeção alguns defeitos podem ser identificados. Os tipos de defeitos podem ser caracterizados como:

- **Incorreção**: Implementação incorreta da especificação do cliente/usuário.
- **Ausência**: Checagem de requisito especificado que não foi incorporado no produto.
- **Extra**: Checagem de requisito não especificado e incorporado no produto.

Técnicas de V&V

CLASSIFICAÇÃO de DEFEITOS

Outra forma de ver os defeitos é pelo tipo de danos que eles podem causar no software: Os feitos podem ser:

- **Pequenos:** Podem ser corrigidos rapidamente e não causam mau funcionamento do software. Exemplos: defeitos de digitação, omissões em textos ou precisam ser esclarecidos para seu entendimento e etc.
- **Grandes:** São defeitos relacionados às especificações que podem causar mau funcionamento do software. Exemplos: ausência de funções, problemas de interfaces etc.
- **Muito sério:** São defeitos que podem comprometer o projeto ocasionando o reprojeto total (ou quase) e a recodificação do software.

Técnicas de V&V

2 DINÂMICAS

As técnicas dinâmicas implicam na execução do produto que está sendo validado.

Exemplo é o [teste de software](#), na qual o programa é executado seguindo alguns casos de teste.



Observação:

Teste de software é o próximo tópico a ser visto na disciplina.

*Qual a definição de
TESTE?*



Erro (*error, mistake*)



Defeito (*bug, fault, defect*)



Falha (*failure*)



Fechamento não esperado de um programa

Durante um teste,
ocorreu uma falha



Fechamento não esperado de um programa



FALHA

A incapacidade de um sistema ou componente em executar as funções requeridas dentro de um nível de desempenho requerido.

É o comportamento operacional do software diferente do esperado pelo usuário.

Falha pode ser entendida como consequência



Mas qual é a causa?



ERRO de acesso à memória

The screenshot displays a Windows debugger interface with the following components:

- Project Explorer:** Shows a project named 'newApp' with folders for 'Header Files', 'Resource Files', and 'Source Files'. The file 'newApp.cpp' is listed under 'Source Files'.
- Code Window:** Displays the source code of 'newApp.cpp'. A breakpoint is set at line 67, which is highlighted in red. The code is as follows:

```
61 case ID_APP_EXIT:  
62     if (PostMessage(hwnd, WM_CLOSE, 0, 0)) {  
63         return TRUE;  
64     }  
65     break;  
66 case ID_APP_ABOUT:  
67     if (!wNotify)  
68         DialogBoxParam(hInstance,  
69             MAKEINTRESOURCE(IDD_ABOUT), hwnd, AboutDialogProc,
```
- Assembly Window:** Shows the assembly instructions corresponding to the code. The instructions are as follows:

```
00401330 mov byte ptr [e  
00401331 add byte ptr [e  
00401332 add byte ptr [e  
00401333 add byte ptr [e  
00401334 cmp dword ptr [  
00401335 jge newApp!OnCo  
00401336 hlt
```
- Variable Window:** Displays the values of variables in the current scope. The variables are:

Name	Address	Value	Type
wId + wNotify	0x00000000	0	int
wId	0x0012FCB8	57664	int
hInstance	0x00415924	{...}	stru
- Watch Window:** Displays the values of variables being watched. The variables are:

Name	Address	Value
hwnd	0x0012FCC4	{...}
unused	0x0012FCC4	32983
wParam	0x0012FCC8	57664
lParam	0x0012FCCC	0

DEPURA-SE para encontrar o **DEFEITO**



FALHA foi o fechamento não esperado do software.



ERRO foi um valor de endereço de memória inválido ou protegido.



DEFEITO estava na codificação. Por exemplo, o programador esqueceu de inicializar uma variável.



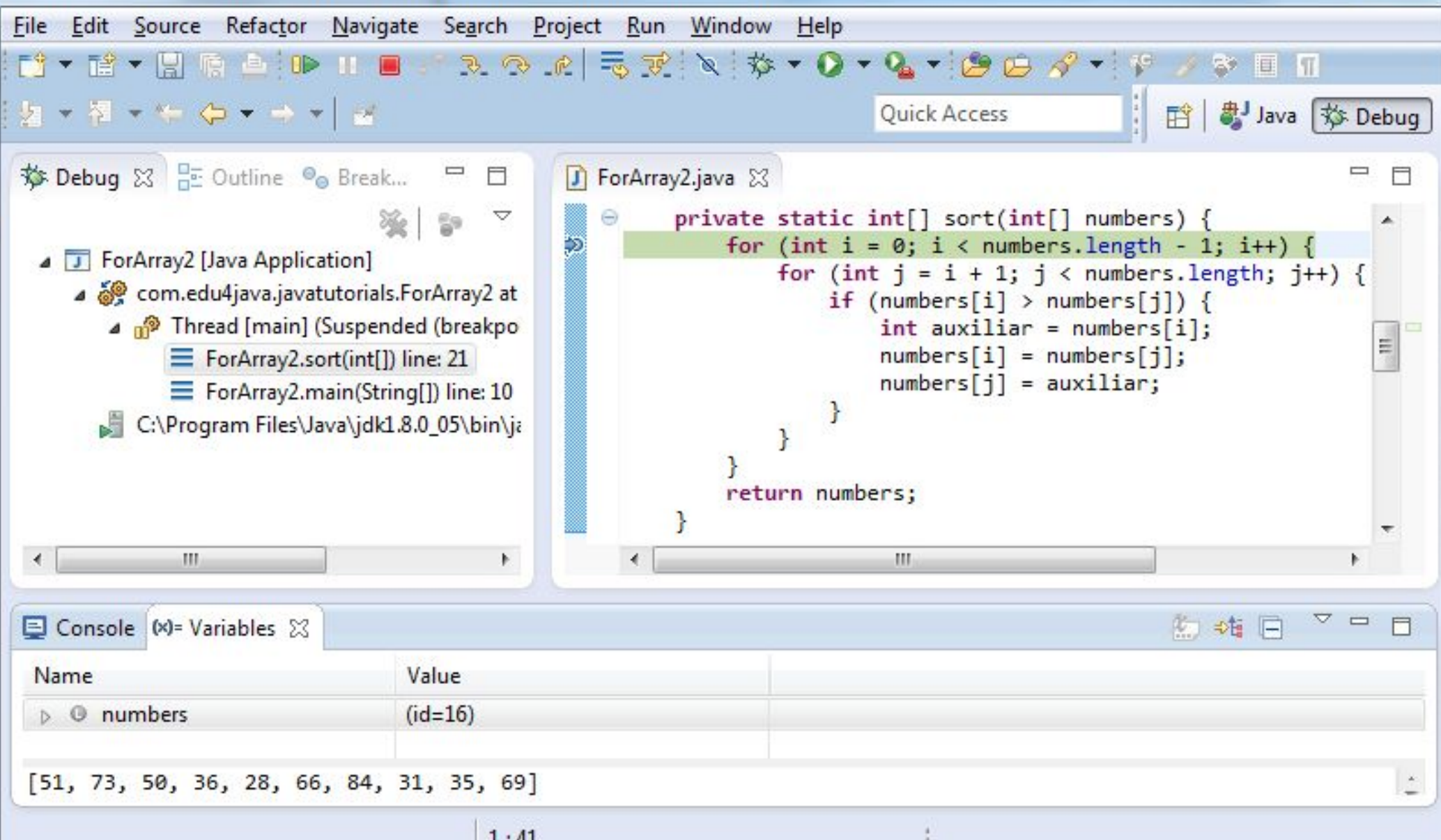
O QUE É O TESTE DE SOFTWARE?



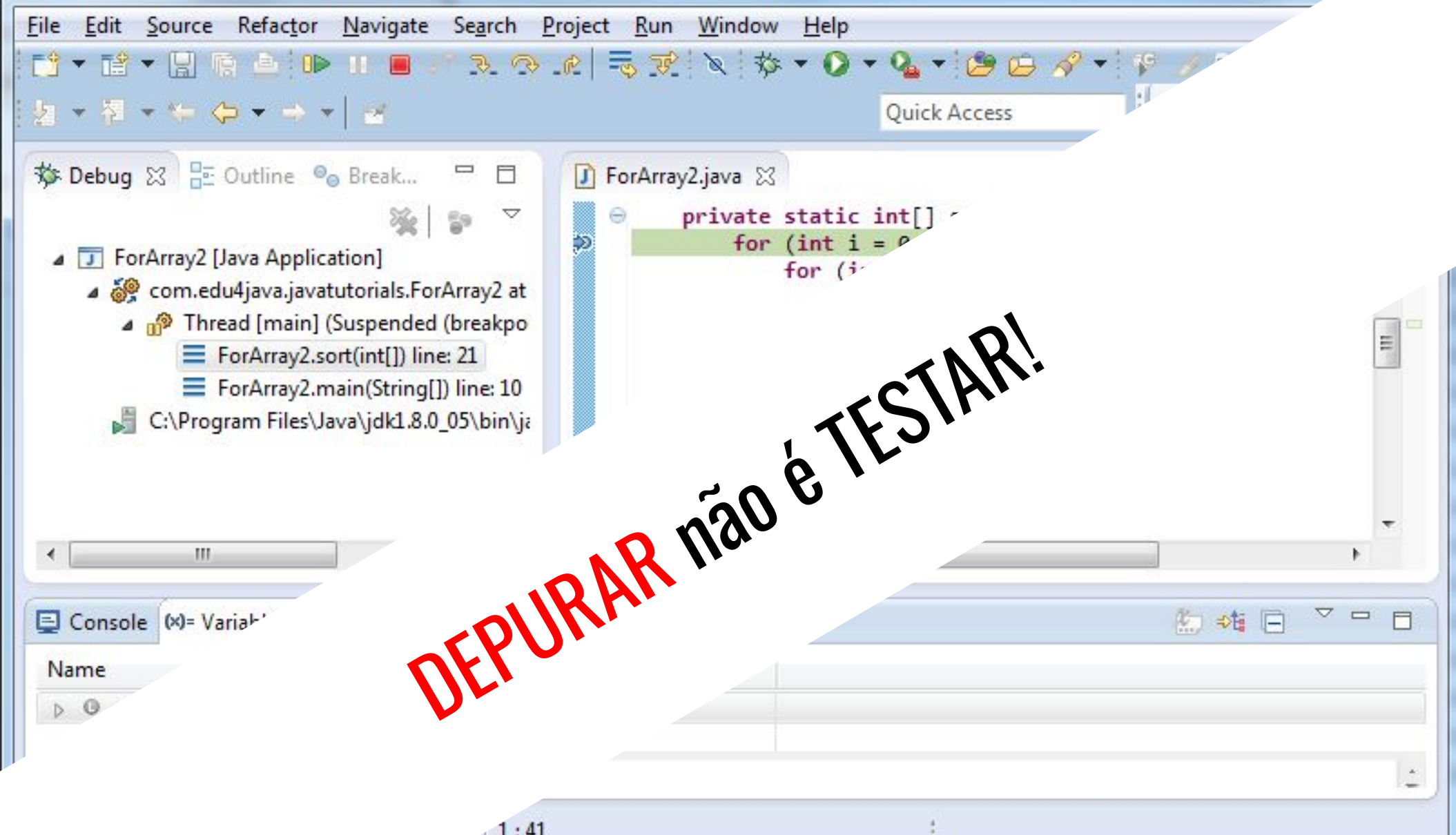
TESTE DE SOFTWARE tem o objetivo de encontrar **FALHAS** em um programa.



Depois do **TESTE DE SOFTWARE**, precisamos encontrar e corrigir **DEFEITOS**, ou seja, **DEPURAR**.



Depois do **TESTE DE SOFTWARE**, precisamos encontrar e corrigir **DEFEITOS**, ou seja, **DEPURAR**.



TESTE DE SOFTWARE, precisamos encontrar e corrigir
DEFEITOS, ou seja, **DEPURAR**.

TESTE

Objetivo: verificar se um software (ou uma parte dele) faz aquilo que deveria fazer e tem por finalidade causar falhas no software.



DEPURAÇÃO

Objetivo: encontrar e corrigir defeitos de codificação durante o desenvolvimento do software.

**“Testes podem somente mostrar
a presença de erros,
não a sua ausência”**

(Dijkstra et. al., 1972)

Teste de Software

O objetivo do teste é **causar uma falha**, então a inexistência de falha pode ser explicada por:

a) Software é de alta qualidade?

b) Os testes foram de baixa qualidade?

**“Se a equipe de desenvolvimento não encontrar a falha,
com certeza ela será identificada pelo usuário”**





TIPOS DE TESTE

Testes podem ser classificados pela **TÉCNICA** usada e
pelo **NÍVEL DE ABRANGÊNCIA**

TÉCNICAS

**FUNCIONAL
CAIXA PRETA**



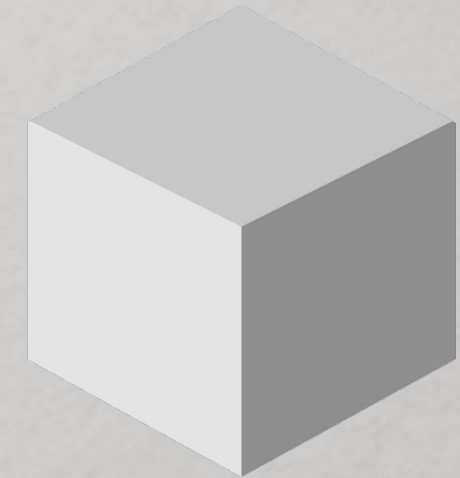
**ESTRUTURAL
CAIXA BRANCA**



Verifica apenas as **entradas e saídas**.

- Fornecemos entradas e verificamos se as saídas estão corretas.

FUNCIONAL CAIXA PRETA



ESTRUTURAL CAIXA BRANCA

- Se baseia na **lógica interna** do componente.
- Necessita acesso ao código do componente.

Teste de unidade

Testa um componente único do software (p.ex. uma única classe)

- Serve para descobrir erros na lógica de um componente

“As menores unidades funcionam corretamente?”

Teste de unidade

Testa um componente único do software (p.ex. uma única classe)

- Serve para descobrir erros na lógica de um componente

“As menores unidades funcionam corretamente?”

**Se as minhas unidades funcionam como o esperado,
não significa que meu software irá funcionar como o esperado?**

Teste de unidade

Testa um componente único do software (p.ex. uma única classe)

- Serve para descobrir erros na lógica de um componente

“As menores unidades funcionam corretamente?”

Infelizmente, não!

Software é complexo e defeitos podem acontecer de várias formas

- perda de dados entre as interfaces (dados incompatíveis)
- efeito imprevisto ou adverso sobre outro (variáveis globais)
- subfunções quando integradas podem não produzir o resultado esperado pela função principal, entre outros

Teste de unidade

Testa um componente único do software (p.ex. uma única classe)

- Serve para descobrir erros na lógica de um componente

“As menores unidades funcionam corretamente?”

Teste de integração

Testa um conjunto de componentes funcionando em conjunto

- Serve para descobrir desacordos entre os componentes
- Exemplo: uma classe espera que método de outra retorne uma mensagem de erro quando item não é encontrado na lista, mas na verdade retorna null

“Quando integradas, elas continuam a produzir o resultado esperado?”

Teste de sistema

Testa o sistema como um todo: hardware, pessoas e informação

- Serve para encontrar falhas em relação aos objetivos do software

“O programa funciona como esperado no seu ambiente como todo?”

Teste de sistema

Testa o sistema como um todo: hardware, pessoas e informação

- Serve para encontrar falhas em relação aos objetivos do software

“O programa funciona como esperado no seu ambiente como todo?”

É uma série de diferentes testes

- Testes de recuperação: “O sistema é tolerante a falhas?”
- Testes de desempenho: “O sistema integrado tem bom desempenho?”
 - Testes de segurança: “O sistema protege contra invasões?”
- Testes de estresse: “O sistema aguenta a alta demanda de recursos?”

Teste de sistema

Testa o sistema como um todo: hardware, pessoas e informação

- Serve para encontrar falhas em relação aos objetivos do software

“O programa funciona como esperado no seu ambiente como todo?”

Teste de aceitação (ou validação)

Testa se os usuários finais estão satisfeitos com o software

“O programa produz o resultado esperado pelo usuário?”

Teste de sistema

Testa o sistema como um todo: hardware, pessoas e informação

- Serve para encontrar falhas em relação aos objetivos do software

“O programa funciona como esperado no seu ambiente como todo?”

Teste de aceitação (ou validação)

Testa se os usuários finais estão satisfeitos com o software

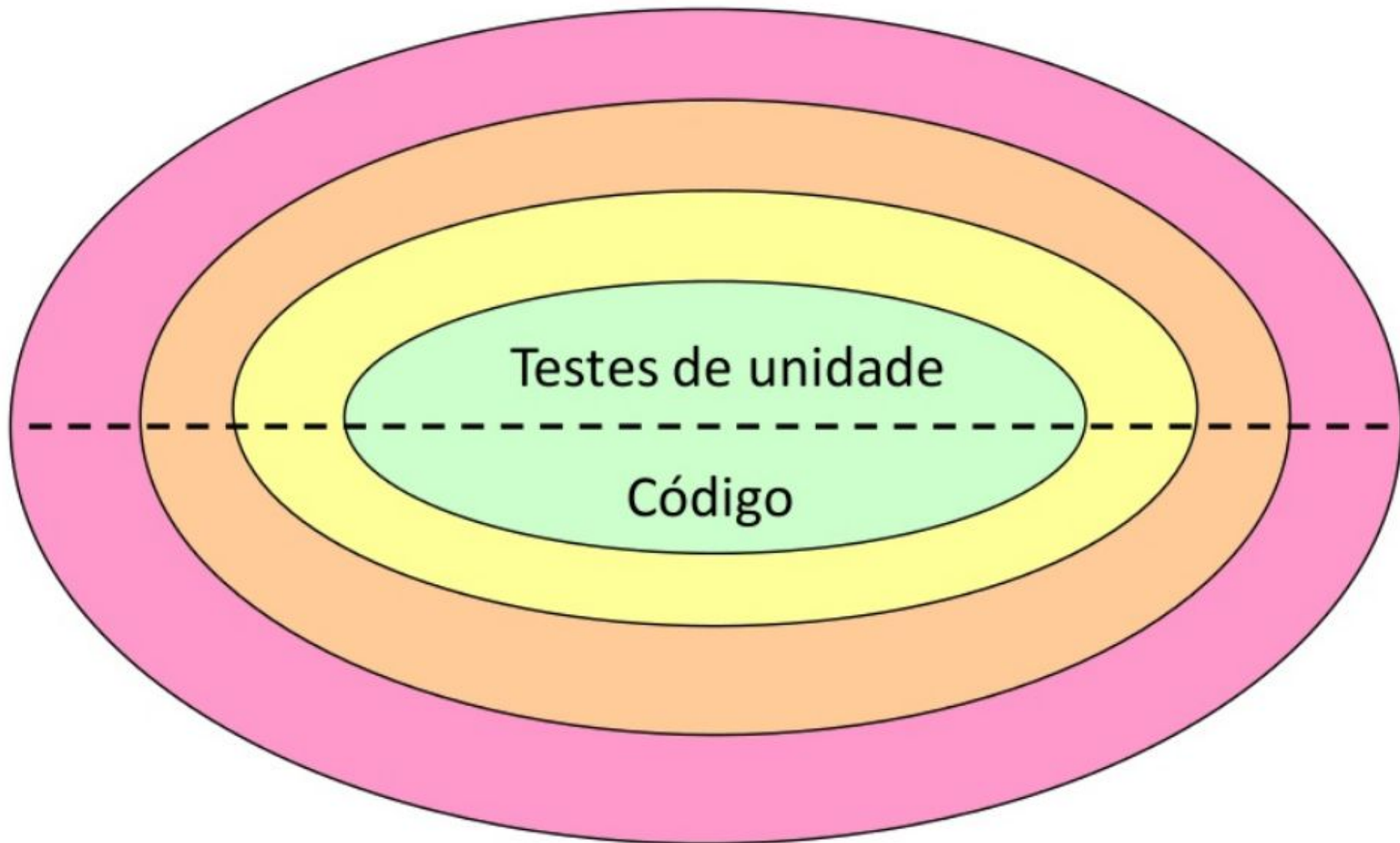
“O programa produz o resultado esperado pelo usuário?”

Teste de operação

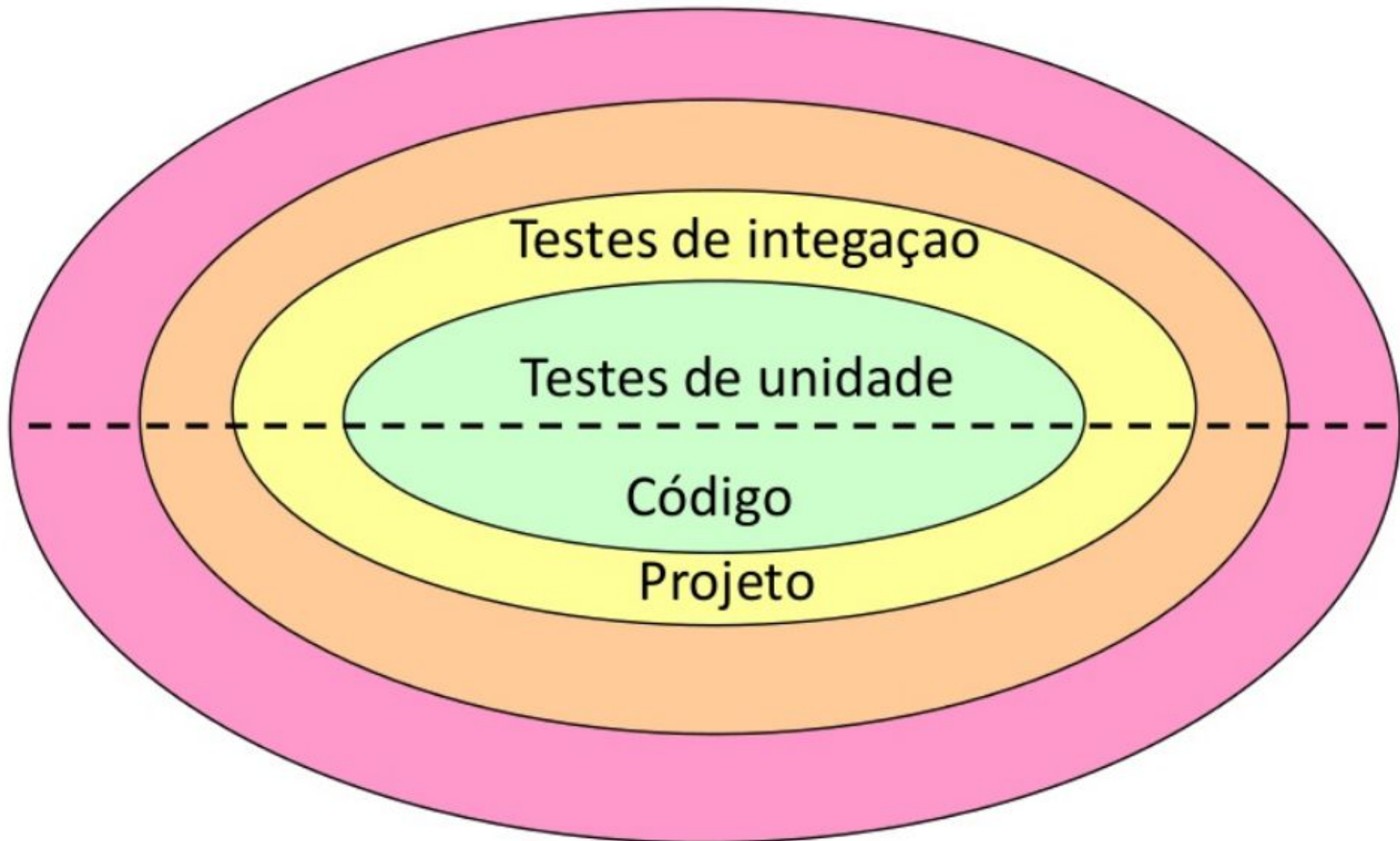
Testa se software pode ser operado adequadamente (p.ex. instalação, logs, backups, gerenciamento, etc.)

“O programa pode ser operado de forma adequada no ambiente?”

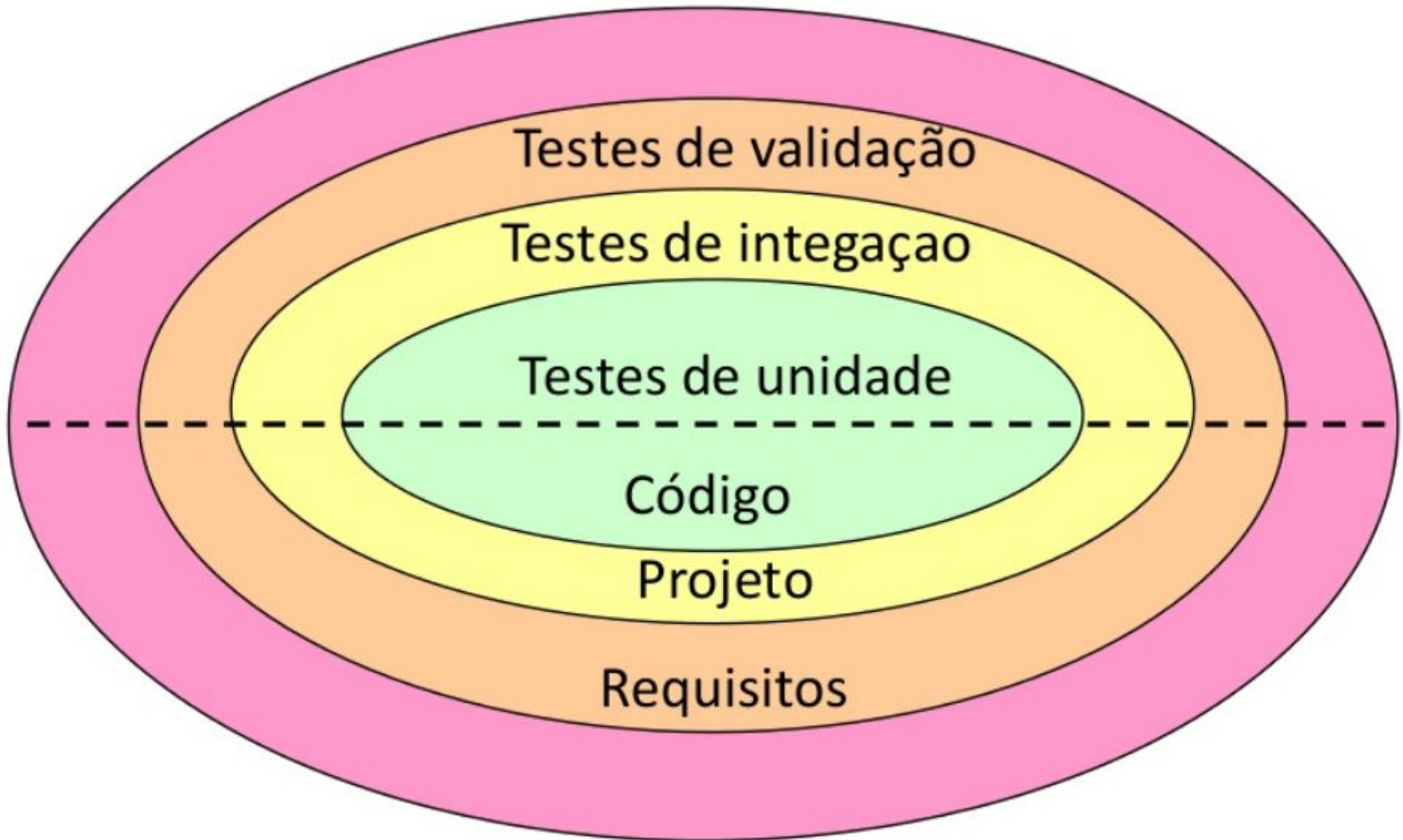
Teste de Software



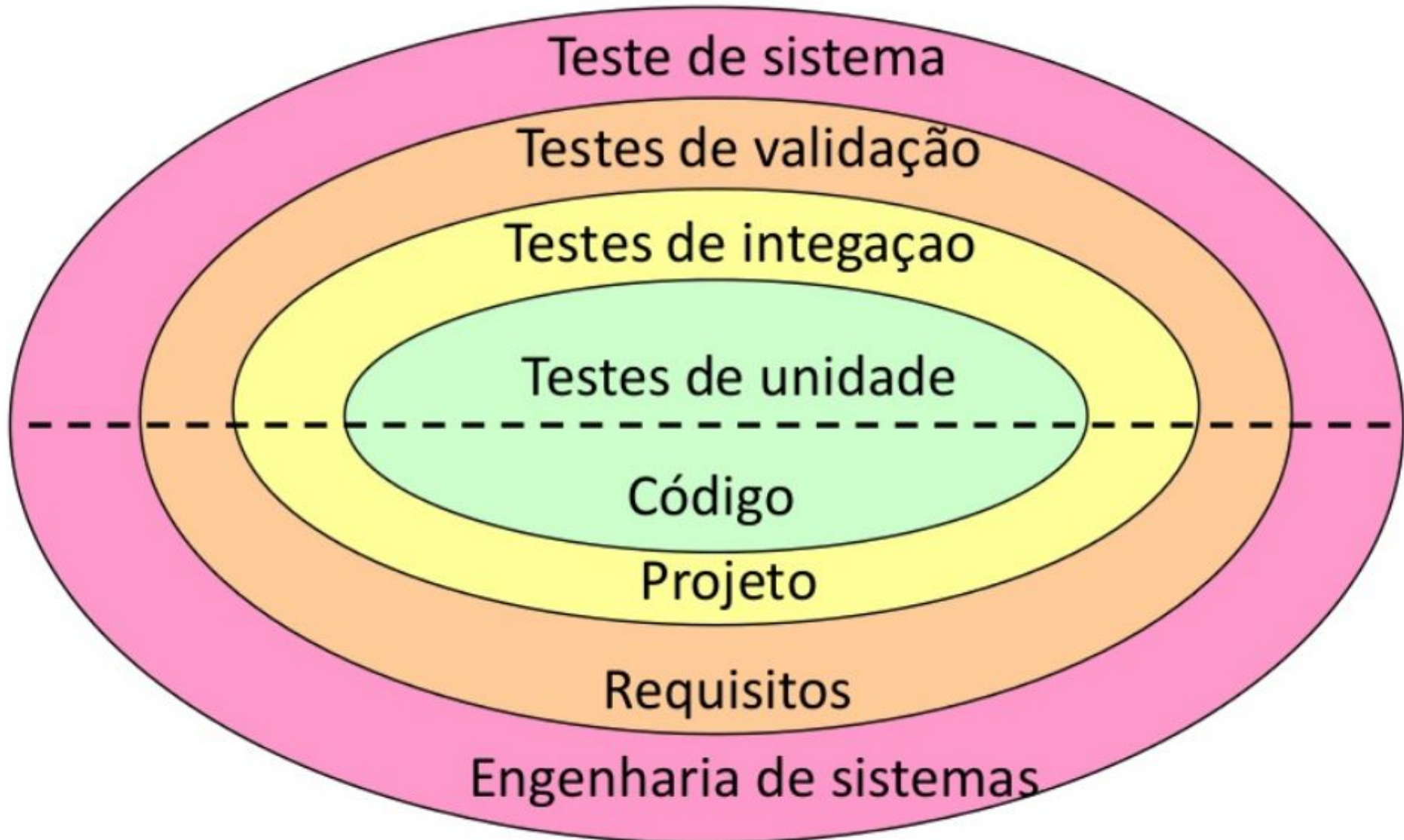
Teste de Software



Teste de Software



Teste de Software





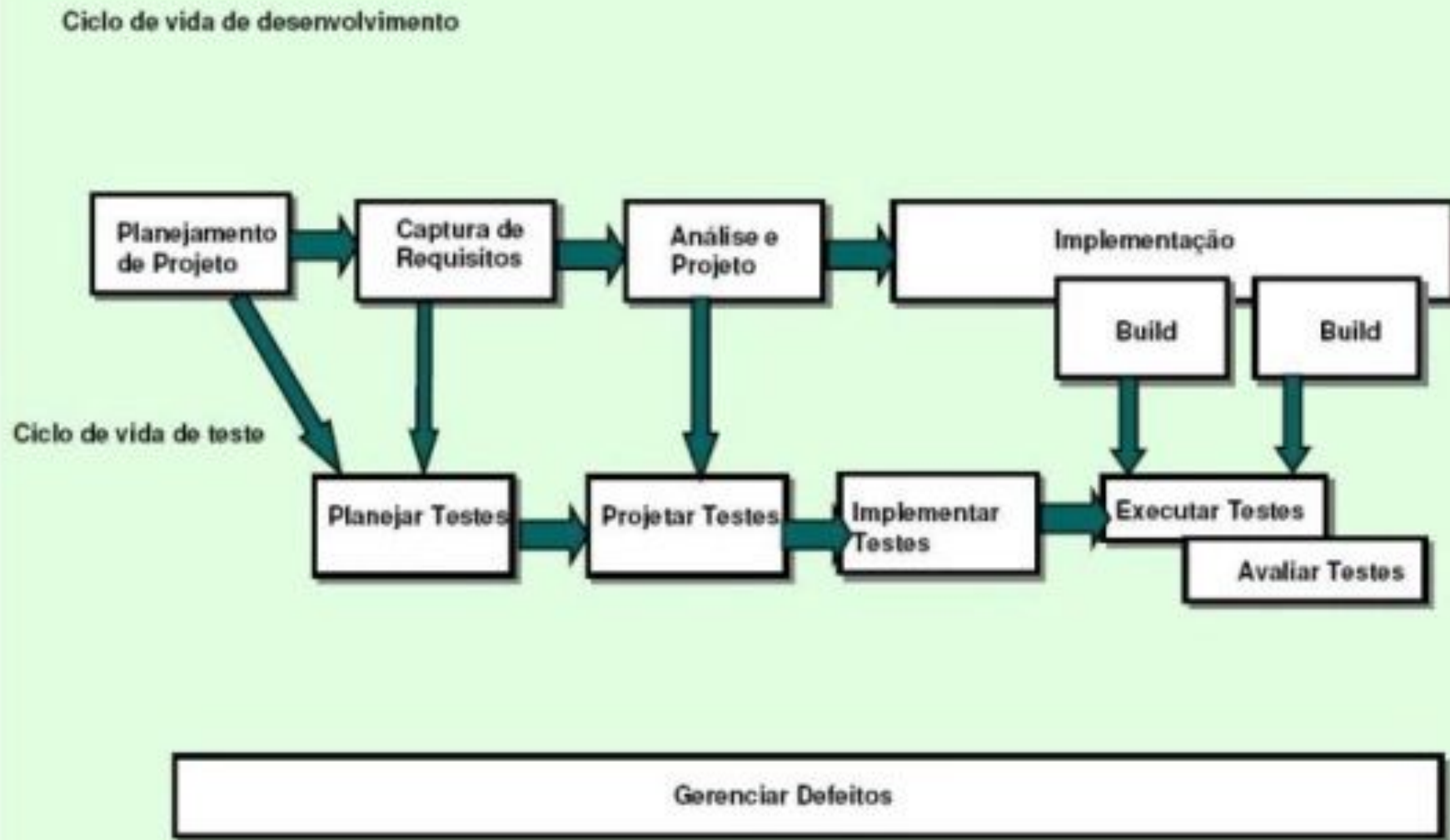
PROCESSO DE TESTE DE SOFTWARE

Processo de Teste de Software

A partir do momento que o planejamento do processo de desenvolvimento do software é **iniciado**, deve-se também iniciar o planejamento dos testes.

Todas as etapas do processo de testes devem ser executadas em paralelo com o processo de desenvolvimento.

Processo de Testes x Processo de Desenvolvimento de Software



Teste de Software

Para garantir a qualidade dos testes, estes devem ser feitos de forma sistemática, que inclui:

1- Planejamento dos Testes

2- Projeto de Casos de Teste

3- Execução e Avaliação dos Casos de Teste

1- Planejamento dos Testes

Planejar é distribuir racionalmente no tempo os recursos disponíveis para realizar alguma atividade.

Testes podem ser formalizados por um documento que descreve como eles devem ser realizados: o **plano de testes**, que deve conter:

- Definir o método
- Recursos necessários
- Cronograma de atividades
- Pessoal necessário
- O que será testado
- O que não será testado
- Responsáveis



1- Planejamento dos Testes

O plano de testes é importante para garantir que os **testes sejam realizados de maneira consistente e completa** durante toda a evolução do software.

O resultado de uma execução do plano de teste deve ser registrado em um **relatório de testes**.

1- Planejamento dos Testes

O plano de testes é dividido em um conjunto de **casos de teste**.

Caso de Teste é um conjunto de ações e os resultados esperados para elas.

1- Planejamento dos Testes

AÇÕES:

- Validar a máscara do campo “CPF”.
- Verificar se o registro foi salvo na base de dados.
- Clique no botão “Voltar”.

1- Planejamento dos Testes

AÇÕES E RESULTADOS:

- Validar a máscara do campo “CPF”.
Resultado: Exibe a máscara 999.999.999-99
- Verificar se o registro foi salvo na base de dados.
Resultado: Registro salvo na tabela <nome_tabela>
- Clique no botão “Voltar”.
Resultado: Sistema retorna à página inicial.

1- Planejamento dos Testes

O plano de testes é dividido em um conjunto de **casos de teste**.

Um caso de teste é similar a um **caso de uso**, porém especifica formalmente qual o ambiente, os atores e o procedimento do teste, bem como os resultados corretos esperados.



O relatório enumera os casos de testes executados e quais os resultados obtidos, indicando se estão de acordo com os resultados esperados.

2- Projeto de Casos de Teste

O projeto de **casos de teste** pode ser tão difícil quanto o projeto do próprio produto a ser testado.

- Poucos desenvolvedores gostam de teste e, menos ainda, do projeto de casos de teste.
- Será escolhido um grupo específico de características a serem testadas. Descrevendo detalhadamente os métodos e testes que deverão ser executados.
- O resultado será o documento de caso de testes e o procedimento de teste.

2- Projeto de Casos de Teste

Casos de testes

- Especificação de uma entrada para o programa e a correspondente saída esperada
 - **Entrada**: conjunto de dados necessários para uma execução do programa
 - **Saída esperada**: resultado de uma execução do programa
- Um bom caso de teste tem alta probabilidade de revelar uma falha ainda não descoberta.

2- Projeto de Casos de Teste

Exemplo de Caso de Teste (Plano)

Caso de teste: adicionar um evento em uma data ocupada por outro evento.

Nível: sistema

Ator: Usuário

Procedimento:

1. Clicar no botão “Adicionar evento”
2. Preencher título qualquer e uma data onde já existe outro evento
3. Clicar no botão “Ok”

2- Projeto de Casos de Teste

Exemplo de Caso de Teste (**Plano**)

Caso de teste: adicionar um evento em uma data ocupada por outro evento.

Resultados esperados:

- Mensagem de erro indicando que data está ocupada.
- Formulário continua visível
- Lista de eventos inalterada

3- Execução e Avaliação dos Casos de Teste

Os casos de testes serão **executados**.

- Toda atividade de teste deve ser registrada, identificando:
 - Hora do teste
 - Procedimento
 - Pessoal envolvido
 - Resultados obtidos
 - Condições ambientais
 - Eventos não esperados (quando ocorrerem)
 - Situação (avaliação final)

3- Execução e Avaliação dos Casos de Teste

Exemplo de Caso de Teste (Relatório)

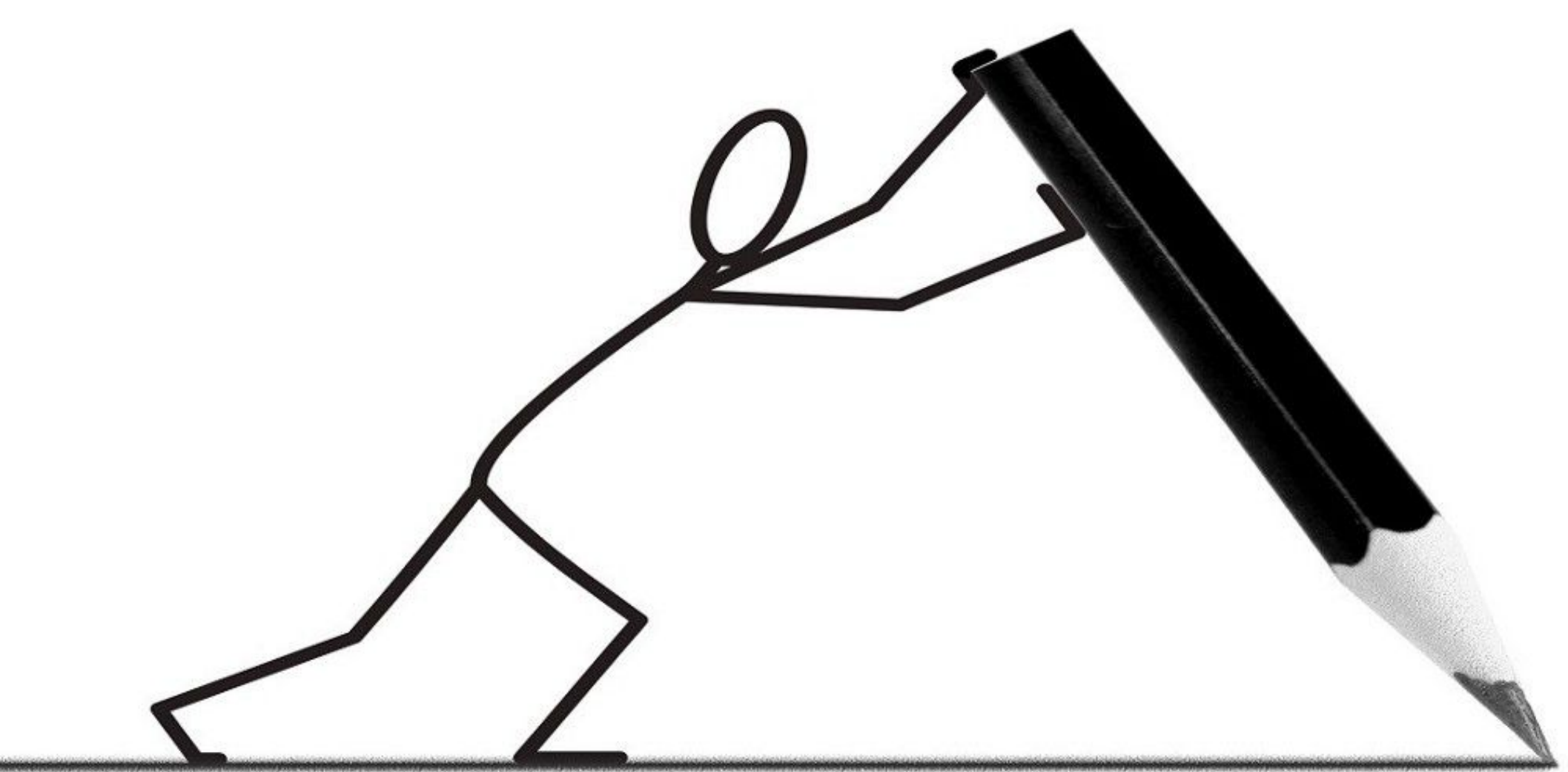
Caso de teste: adicionar um evento em uma data ocupada por outro evento.

Resultados obtidos:


- Programa fecha abruptamente ao clicar botão “Ok”

Avaliação: Não passou





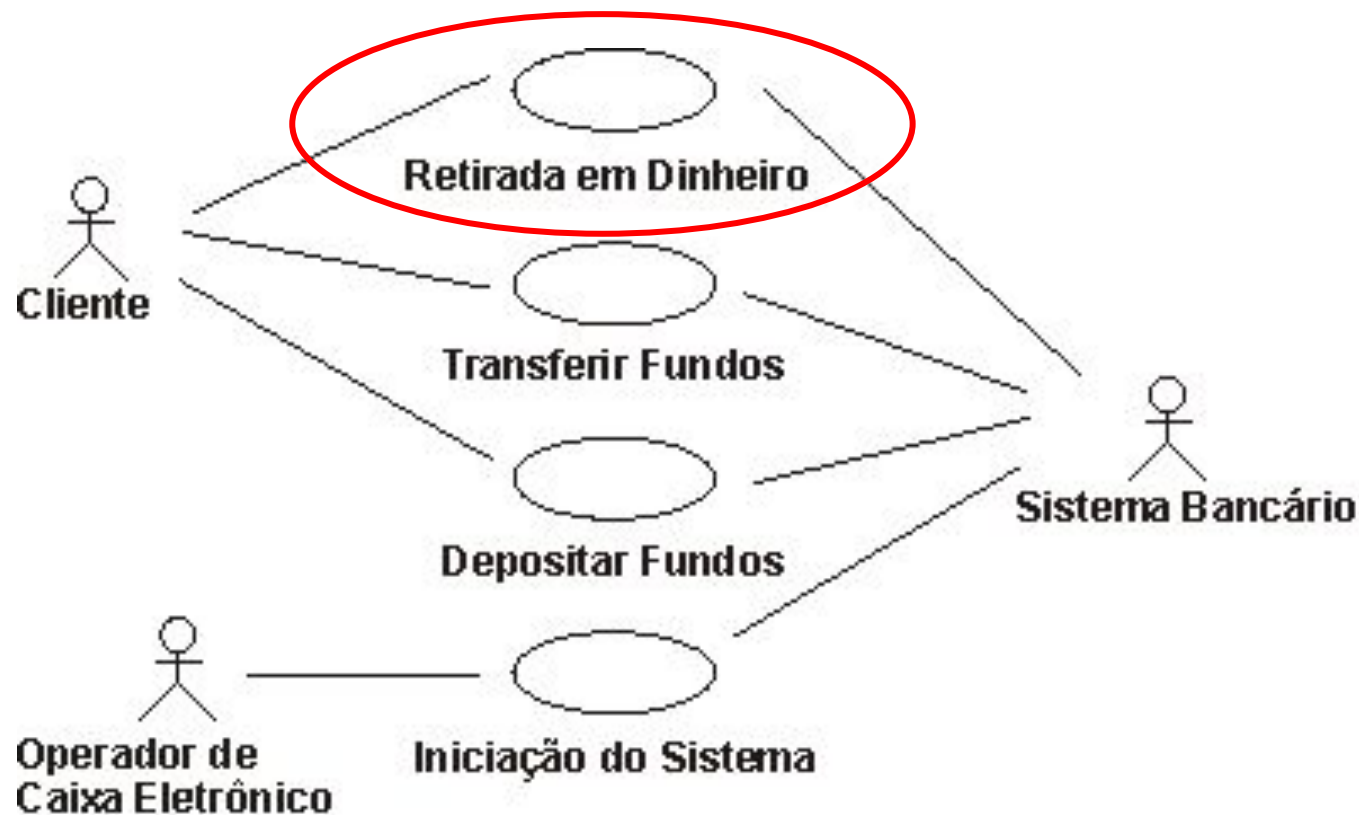
Escrever Casos de Teste



PASSO 1 - IDENTIFICAÇÃO DOS FLUXOS BÁSICOS E FLUXOS ALTERNATIVOS NOS CASOS DE USO



PASSO 1 - IDENTIFICAÇÃO DOS FLUXOS BÁSICOS E FLUXOS ALTERNATIVOS NOS CASOS DE USO



Esse Caso de Uso começa com o caixa eletrônico no Estado Pronto.

1. Iniciar Retirada - O cliente insere o cartão bancário no leitor de cartões do caixa eletrônico
2. Verificar o Cartão Bancário - O caixa eletrônico lê o código da conta a partir da tarja magnética do cartão bancário e verifica se ele é um cartão aceitável.
3. Digitar a senha - O caixa eletrônico pede a senha do cliente (4 dígitos)
4. Verificar o código da conta e a senha - O código da conta e a senha são verificados para determinar se a conta é válida e se a senha digitada está correta. Para esse fluxo, a conta é válida e a senha está corretamente associada a essa conta.
5. Opções do caixa eletrônico - O caixa eletrônico exibe as diversas alternativas disponíveis. Nesse fluxo, o cliente do banco sempre seleciona "Retirada em Dinheiro."
6. Digitar o Valor - O caixa eletrônico solicita o valor a ser retirado. Para esse fluxo o cliente seleciona um valor predefinido (R\$ 10, R\$ 20, R\$ 50 ou R\$ 100).
7. Autorização - O caixa eletrônico inicia o processo de verificação com o Sistema Bancário, enviando o ID do Cartão, a Senha, o Valor e as Informações de conta como uma transação. Para esse fluxo, o Sistema Bancário está on-line e responde com uma autorização para concluir a retirada em dinheiro, atualizando o saldo da conta de forma apropriada.
8. Fornecimento - O Dinheiro é fornecido.
9. Devolução do Cartão - o Cartão do Banco é devolvido.
10. Recibo - O recibo é impresso e fornecido. O caixa eletrônico também atualiza o log interno de forma apropriada.


O Caso de Uso termina com o caixa eletrônico no Estado Pronto

Fluxo Básico	---
Fluxo Alternativo 1 - Cartão Inválido.	No Passo 2 do Fluxo Básico - Verificar o Cartão Bancário, se o cartão não for válido, será ejetado com uma mensagem apropriada.
Fluxo Alternativo 2 - Caixa Eletrônico sem Dinheiro	No Passo 5 do Fluxo Básico - Opções do Caixa Eletrônico, se o caixa eletrônico estiver sem dinheiro, a opção "Retirada em Dinheiro" não estará disponível.
Fluxo Alternativo 3 - Fundos insuficientes no caixa eletrônico	No Passo 6 do Fluxo Básico - Digitar o Valor, se o caixa eletrônico não contiver fundos suficientes para fornecer o valor solicitado, o sistema exibirá uma mensagem apropriada e retornará ao Passo 6 do fluxo básico - Digitar o Valor.
Fluxo Alternativo 4 - Senha Incorreta	<p>No Passo 4 do Fluxo Básico - Verificar a Conta e a Senha, o cliente tem três chances de digitar a senha correta.</p> <p>Se for digitada uma senha incorreta, o caixa eletrônico exibirá a mensagem apropriada, e se houver novas tentativas, esse fluxo retornará ao Passo 3 do Fluxo Básico - Digitar a Senha.</p> <p>Se na última tentativa o número PIN digitado estiver incorreto, o cartão será retido, o caixa eletrônico retornará ao Estado Pronto, e esse caso de uso será encerrado.</p>



PASSO 2 - IDENTIFICAÇÃO DOS CENÁRIOS

Começando pelo fluxo básico e depois combinando esse fluxo com os fluxos alternativos, é possível identificar os seguintes cenários de caso de uso:




Cenário 1 - Retirada em dinheiro bem-sucedida	Fluxo Básico	
Cenário 2 - Caixa eletrônico sem dinheiro	Fluxo Básico	Fluxo Alternativo 2
Cenário 3 - Fundos Insuficientes no Caixa Eletrônico	Fluxo Básico	Fluxo Alternativo 3
Cenário 4 - Senha Incorreta (novas tentativas)	Fluxo Básico	Fluxo Alternativo 4
Cenário 5 - Senha incorreta (sem nova tentativa)	Fluxo Básico	Fluxo Alternativo 4
Cenário 6 - Nenhuma Conta/tipo de conta incorreto	Fluxo Básico	Fluxo Alternativo 5
Cenário 7 - Saldo Insuficiente em Conta	Fluxo Básico	Fluxo Alternativo 6



PASSO 3 - CRIAÇÃO DE MATRIZ OU TABELA DE DECISÃO

É possível identificar e gerenciar os casos de teste usando matrizes ou tabelas de decisão.



ID do TC	Cenário/Condição	Senha	No da Conta	Valor Digitado (ou escolhido)	Valor na Conta	Valor no Caixa Eletrônico	Resultado Esperado
CW1.	Cenário 1 - Retirada em Dinheiro Bem-sucedida	V	V	V	V	V	Retirada em dinheiro bem-sucedida.
CW2.	Cenário 2 - Caixa Eletrônico sem Dinheiro	V	V	V	V	I	Opção Retirada em Dinheiro indisponível, fim do caso de uso
CW3.	Cenário 3 - Fundos insuficientes no caixa eletrônico	V	V	V	V	I	Mensagem de aviso, retorno ao Passo 6 do Fluxo Básico - Digitar o Valor
CW4.	Cenário 4 - Senha Incorreta (> 1 nova tentativa)	I	V	n/a	V	V	Mensagem de aviso, retorno ao Passo 4 do Fluxo Básico, Digitar a Senha
CW5.	Cenário 4 - Senha Incorreta (= 1 nova tentativa)	I	V	n/a	V	V	Mensagem de aviso, retorno ao Passo 4 do Fluxo Básico, Digitar a Senha
CW6.	Cenário 4 - Senha Incorreta (= sem novas tentativas)	I	V	n/a	V	V	Mensagem de aviso, cartão retido, fim do caso de uso

Legenda:

(V) Válido

(I) Inválido



PASSO 4 - IDENTIFICAÇÃO DOS VALORES REAIS DA MATRIZ

Após a aprovação dos casos de teste, será possível identificar os valores reais dos dados (na matriz de implementação do caso de teste) e criar os dados de teste.

ID do TC	Cenário/Condição	Senha	No da Conta	Valor Digitado (ou escolhido)	Valor na Conta	Valor no Caixa Eletrônico	Resultado Esperado
CW1.	Cenário 1 - Retirada em Dinheiro Bem-sucedida	4987	809 - 498	50.00	500.00	2,000	Retirada em dinheiro bem-sucedida. Saldo da conta atualizado para 450,00
CW2.	Cenário 2 - Caixa Eletrônico sem Dinheiro	4987	809 - 498	100,00	500,00	0,00	Opção Retirada em Dinheiro indisponível, fim do caso de uso
CW3.	Cenário 3 - Fundos insuficientes no caixa eletrônico	4987	809 - 498	100,00	500,00	70,00	Mensagem de aviso, retorno ao Passo 6 do Fluxo Básico - Digitar o Valor
CW4.	Cenário 4 - Senha Incorreta (> 1 nova tentativa)	49 <u>7</u> 8	809 - 498	n/a	500,00	2.000	Mensagem de aviso, retorno ao Passo 4 do Fluxo Básico, Digitar a Senha
CW5.	Cenário 4 - Senha Incorreta (= 1 nova tentativa)	49 <u>7</u> 8	809 - 498	n/a	500,00	2.000	Mensagem de aviso, retorno ao Passo 4 do Fluxo Básico, Digitar a Senha
CW6.	Cenário 4 - Senha Incorreta (= sem novas tentativas)	49 <u>7</u> 8	809 - 498	n/a	500,00	2.000	Mensagem de aviso, cartão retido, fim do caso de uso

Dúvidas?

Profa. Thaiana Pereira dos Anjos Reis, Dra.
thaiana.anjos@ifsc.edu.br