



Instituto Federal de Educação, Ciência e Tecnologia da Bahia - IFBA
Departamento de Ciência da Computação
Tecnólogo em Análise e Desenvolvimento de Sistemas

Testes de software - PARTE 2

André L. R. Madureira <andre.madureira@ifba.edu.br>
Doutorando em Ciência da Computação (UFBA)
Mestre em Ciência da Computação (UFBA)
Engenheiro da Computação (UFBA)

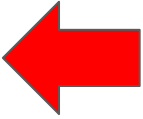
Revisão da aula anterior

- Testes **caixa-preta** vs **caixa-branca**
- **Casos de teste**
- **Desenvolvimento dirigido a testes** (*Test-first*)
 - Cobertura de código
 - Testes de regressão
 - Depuração simplificada
 - Documentação facilitada

Revisão da aula anterior

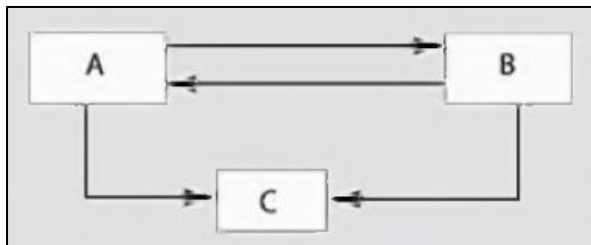
- Testes de **validação x defeitos**
- **Etapas de testes de software**
 - Testes de desenvolvimento
 - **Testes unitários <= paramos aqui**

Testes de desenvolvimento

- Os testes podem ocorrer em três níveis de granularidade:
 - **Teste unitário**
 - **Teste de componentes** 
 - **Teste de sistema**

Testes de componentes

- Componentes de software são compostos por objetos que interagem
 - Erros de interface no componente aparecem em decorrência de interações entre os objetos do componente
- Testes de componentes **tentam demonstrar que a interface do componente** se comporta de acordo com sua especificação



Exercício

Considerando o escopo de testes de software, marque a alternativa que contém **somente** as assertivas VERDADEIRAS.

I - Testes unitários devem verificar cada atributo ou método, e também avaliar todos os estados possíveis de um objeto.

II - O teste componentes permite testar componentes isolados uns dos outros, de forma automática.

III - Os testes de desenvolvimento em sistemas comerciais geralmente tem três estágios: testes de desenvolvimento, testes de release e testes de usuário.

IV - Dentre os testes de desenvolvimento, há o teste subintegração, teste de componentes e de sistema.

- ☐ Somente I e II.
- ☐ Somente II e III.
- ☐ Somente IV.
- ☐ Somente I e III.
- ☐ Somente I, II e III.

Exercício

Considerando o escopo de testes de software, marque a alternativa que contém **somente** as assertivas VERDADEIRAS.

I - Testes unitários devem verificar cada atributo ou método, e também avaliar todos os estados possíveis de um objeto. **V** [Slide 47: Teste unitário](#) e [Slide 49](#)

II - O teste componentes permite testar componentes isolados uns dos outros, de forma automática.

III - Os testes de desenvolvimento em sistemas comerciais geralmente tem três estágios: testes de desenvolvimento, testes de release e testes de usuário.

IV - Dentre os testes de desenvolvimento, há o teste subintegração, teste de componentes e de sistema.

- ☐ Somente I e II.
- ☐ Somente II e III.
- ☐ Somente IV.
- ☐ Somente I e III.
- ☐ Somente I, II e III.

Exercício

Considerando o escopo de testes de software, marque a alternativa que contém **somente** as assertivas VERDADEIRAS.

I - Testes unitários devem verificar cada atributo ou método, e também avaliar todos os estados possíveis de um objeto. **V**

II - O teste componentes permite testar componentes isolados uns dos outros, de forma automática. **F** [Slide 44: Teste unitário](#) e [Slide 65](#)

III - Os testes de desenvolvimento em sistemas comerciais geralmente tem três estágios: testes de desenvolvimento, testes de release e testes de usuário.

IV - Dentre os testes de desenvolvimento, há o teste subintegração, teste de componentes e de sistema.

- ☐ Somente I e II.
- ☐ Somente II e III.
- ☐ Somente IV.
- ☐ Somente I e III.
- ☐ Somente I, II e III.

Exercício

Considerando o escopo de testes de software, marque a alternativa que contém **somente** as assertivas VERDADEIRAS.

I - Testes unitários devem verificar cada atributo ou método, e também avaliar todos os estados possíveis de um objeto. **V**

II - O teste componentes permite testar componentes isolados uns dos outros, de forma automática. **F**

III - Os testes de desenvolvimento em sistemas comerciais geralmente tem três estágios: testes de desenvolvimento, testes de release e testes de usuário. **V**

IV - Dentre os testes de desenvolvimento, há o teste subintegração, teste de componentes e de sistema.

- ☐ Somente I e II.
- ☐ Somente II e III.
- ☐ Somente IV.
- ☐ Somente I e III.
- ☐ Somente I, II e III.

[Slide 39: Etapas de testes em softwares comerciais](#)

Exercício

Considerando o escopo de testes de software, marque a alternativa que contém **somente** as assertivas VERDADEIRAS.

I - Testes unitários devem verificar cada atributo ou método, e também avaliar todos os estados possíveis de um objeto. **V**

II - O teste componentes permite testar componentes isolados uns dos outros, de forma automática. **F**

III - Os testes de desenvolvimento em sistemas comerciais geralmente tem três estágios: testes de desenvolvimento, testes de release e testes de usuário. **V**

IV - Dentre os testes de desenvolvimento, há o teste subintegração teste de componentes e de sistema. **F**

[Slide 41: Testes de desenvolvimento](#)

☐ Somente I e II.

☐ Somente II e III.

☐ Somente IV.

☒ Somente I e III.

☐ Somente I, II e III.

Classificação de Erros de Interface

- Erros de interface são classificados nas seguintes classes:
 - **Mau uso de interface**
 - **Mau-entendimento de interface**

Mau uso de Interface

- Componente chamador **conhece o comportamento esperado** do outro componente porém **comete um erro no uso** de sua interface
- **Ex:** função chamada com parâmetros com
 - Tipo errado
 - Ordem errada
 - Número errado de parâmetros

```
#include <stdio.h>

int sum(int a, int b)
{
    return a + b;
}

int main()
{
    int add = sum("a", "b");

    printf("Sum is: %d", add);

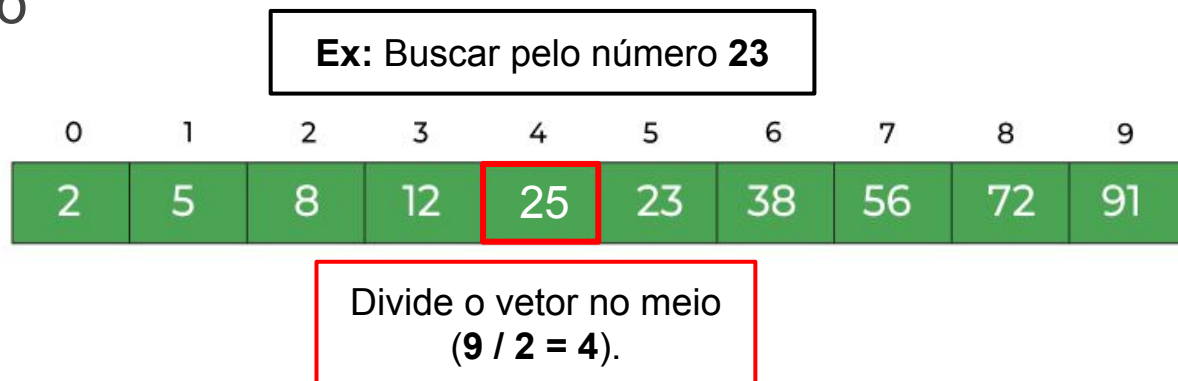
    return 0;
}
```

Formal Parameter

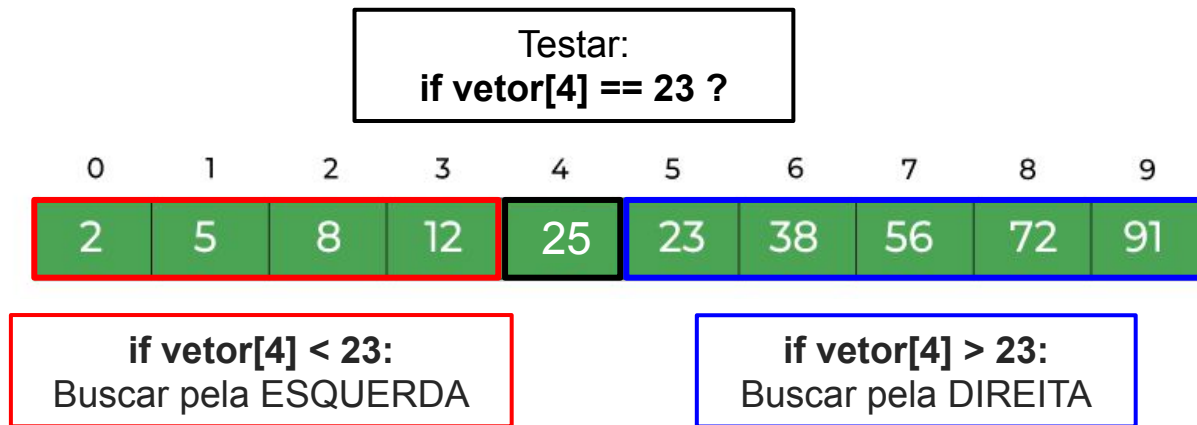
Actual Parameter

Mau-entendimento de interface

- Componente chamador **desconhece o comportamento** da interface do componente chamado e **faz suposições** sobre seu comportamento
- **Ex:** função de busca binária chamada com um vetor não ordenado



Mau-entendimento de interface



**Qual o problema de
usar esse algoritmo
com esse vetor
especificamente?**

O algoritmo só funciona se o vetor estiver ordenado.
Ou seja, quem usou esse algoritmo não sabe como ela
funciona (**mau entendimento da interface**).

Diretrizes para testes de componentes (interface)

- Examinar código explicitamente, projetando casos de teste com valores de parâmetros extremos
 - **Ex:** função para calcular números primos ($p < 0$, $p = 0$, $p > 0$)
- Sempre testar componentes cujos parâmetros são ponteiros com o valor NULO
 - **Ex:** passar um vetor nulo para uma função *soma(*vetor)*

Diretrizes para testes de componentes (interface)

- Projete testes de estresse para descobrir erros de *timing* em sistemas complexos
 - **Ex:** em uma caixa de email, projete um cenário onde muitos emails são enviados ao mesmo tempo para a caixa de saída (e veja se algum erro ocorre)
 - A caixa fica cheia? Há estouro da fila? O programa sofre *crash* com *segmentation fault*?)



Diretrizes para testes de componentes (interface)

- No caso de componentes que compartilham memória, teste se a ordem da execução deles causa algum problema
 - **Ex:** se eu executar primeiro o programa para enviar emails da caixa de saída, será que isso gera algum problema?



Diretrizes para testes de componentes (interface)

- No caso de componentes que compartilham memória, teste se a ordem da execução deles causa algum problema
 - **Ex:** se eu executar primeiro o programa para enviar emails da caixa de saída, será que isso gera algum problema?
 - Em teoria o programa deve verificar se há emails para serem enviados, antes de qualquer ação
 - E se eu executar primeiro o programa para colocar um email na caixa de saída?



Gmail

Testes de sistema

- Testes realizados com os componentes do sistema integrados, compondo o sistema completo
 - **Objetivos:**
 - Encontrar erros nas interações entre componentes e problemas de interface entre componentes
 - Atestar que o sistema cumpre com seus requisitos (funcionais e não-funcionais)

Testes de sistema

- Sistemas grandes podem exigir **processos multi-estágio**
 - **Processos multi-estágio:** componentes são integrados para formar subsistemas
 - Subsistemas são testados individualmente antes de serem combinados para compor o sistema final

Testes de sistema x Testes de Componentes

- **Teste de sistema verificam todos os componentes** (incluindo os reusáveis e de prateleira)
 - Testes de componente testam componentes recém desenvolvidos
- Testes de sistema **verificam a integração** entre componentes desenvolvidos por diferentes membros da equipe ou grupos
 - Testes de componentes nem sempre verificam a integração

Testes de sistema

- Verificam o comportamento do sistema como um todo (**comportamento emergente**)
- **Ex:** Integração de componentes de autenticação e atualização de dados.
 - Permite testar um sistema no qual há restrição na atualização de informações, permitida apenas para usuários autorizados

Testes de sistema

- Verificam o comportamento do sistema como um todo (**comportamento emergente**)

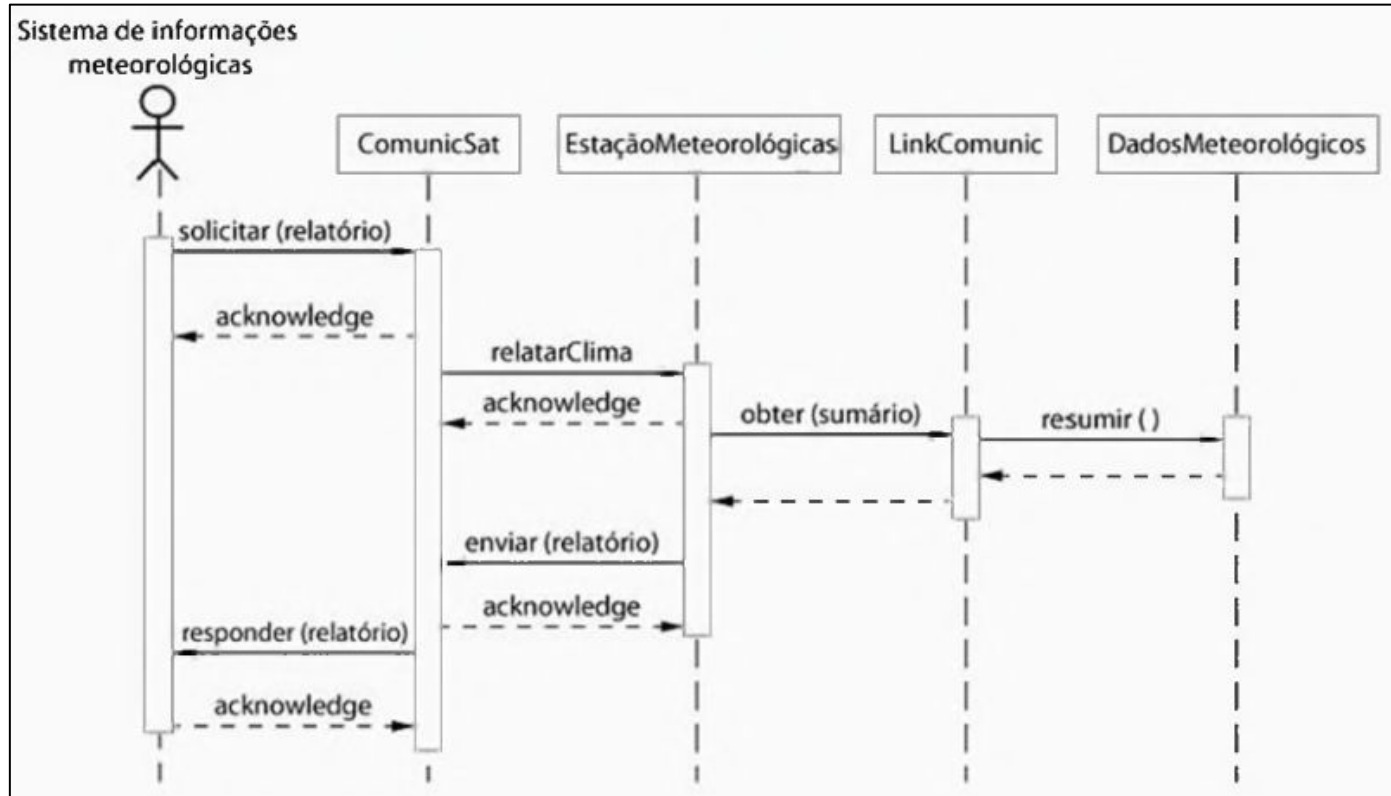
Lembrete:

Devemos testar apenas os recursos que estão previstos na especificação de requisitos do sistema

Testes de sistema

- Também permitem identificar equívocos dos desenvolvedores de componentes sobre outros componentes do sistema.
 - Pressuposições sobre a interface dos componentes
 - Mau uso da interface
 - E demais falhas que também são detectadas pelos testes de componentes
- Testes de sistema são úteis também para testes de desempenho e confiabilidade

Caso de uso e diagramas de sequência são abordagens eficazes para organizar os **testes de sistema**



Testes de sistema

- Testes baseados em caso de uso e diagramas de sequência são abordagens eficazes para testes de sistema
 - Descrevem a interação entre componentes
 - Facilitam a visualização do comportamento esperado do sistema (requisitos, entradas e saídas)

Exercício

Considerando o escopo de testes de componentes, marque a alternativa que contém **somente** as assertivas VERDADEIRAS.

I - Os testes de componentes verificam erros de interface, que podem surgir em decorrência de interações entre componentes.

II - Um dos tipos de erros de interface é o mau uso da interface. Este erro ocorre pois componente chamador de outro componente desconhece a especificação da interface deste (ex: chamar uma função que necessita de um vetor ordenado passando como parâmetro um vetor desordenado).

III - Outro tipo de erro de interface é o mau-entendimento da interface. Este erro ocorre pois componente chamador de outro componente conhece o comportamento esperado do componente, porém comete um erro ao utilizar sua interface (ex: chamada de função com parâmetros errados).

IV - O último tipo de erro de interface são os erros de timing. Eles ocorrem em sistemas em tempo real que usam uma memória compartilhada ou uma interface para passagem de mensagens.

- ☐ Todas as assertivas são verdadeiras.
- ☐ Somente II e III.
- ☐ Somente I e IV.
- ☐ Somente I, II e III.
- ☐ Nenhuma das alternativas anteriores.

Exercício

Considerando o escopo de testes de componentes, marque a alternativa que contém **somente** as assertivas VERDADEIRAS.

I - Os testes de componentes verificam erros de interface, que podem surgir em decorrência de interações entre componentes. **V**

II - Um dos tipos de erros de interface é o mau uso da interface. Este erro ocorre pois componente chamador de outro componente desconhece a especificação da interface deste (ex: chamar uma função que necessita de um vetor ordenado passando como parâmetro um vetor desordenado).

III - Outro tipo de erro de interface é o mau-entendimento da interface. Este erro ocorre pois componente chamador de outro componente conhece o comportamento esperado do componente, porém comete um erro ao utilizar sua interface (ex: chamada de função com parâmetros errados).

IV - O último tipo de erro de interface são os erros de timing. Eles ocorrem em sistemas em tempo real que usam uma memória compartilhada ou uma interface para passagem de mensagens.

- ☐ Todas as assertivas são verdadeiras.
- ☐ Somente II e III.
- ☐ Somente I e IV.
- ☐ Somente I, II e III.
- ☐ Nenhuma das alternativas anteriores.

Exercício

Considerando o escopo de testes de componentes, marque a alternativa que contém **somente** as assertivas VERDADEIRAS.

I - Os testes de componentes verificam erros de interface, que podem surgir em decorrência de interações entre componentes. **V**

II - Um dos tipos de erros de interface é o mau uso da interface. Este erro ocorre pois componente chamador de outro componente desconhece a especificação da interface deste (ex: chamar uma função que necessita de um vetor ordenado passando como parâmetro um vetor desordenado). **F**

III - Outro tipo de erro de interface é o mau-entendimento da interface. Este erro ocorre pois componente chamador de outro componente conhece o comportamento esperado do componente, porém comete um erro ao utilizar sua interface (ex: chamada de função com parâmetros errados).

IV - O último tipo de erro de interface são os erros de timing. Eles ocorrem em sistemas em tempo real que usam uma memória compartilhada ou uma interface para passagem de mensagens.

- ☐ Todas as assertivas são verdadeiras.
- ☐ Somente II e III.
- ☐ Somente I e IV.
- ☐ Somente I, II e III.
- ☐ Nenhuma das alternativas anteriores.

Exercício

Considerando o escopo de testes de componentes, marque a alternativa que contém **somente** as assertivas VERDADEIRAS.

I - Os testes de componentes verificam erros de interface, que podem surgir em decorrência de interações entre componentes. **V**

II - Um dos tipos de erros de interface é o mau uso da interface. Este erro ocorre pois componente chamador de outro componente desconhece a especificação da interface deste (ex: chamar uma função que necessita de um vetor ordenado passando como parâmetro um vetor desordenado). **F**

III - Outro tipo de erro de interface é o mau-entendimento da interface. Este erro ocorre pois componente chamador de outro componente conhece o comportamento esperado do componente, porém comete um erro ao utilizar sua interface (ex: chamada de função com parâmetros errados). **F**

IV - O último tipo de erro de interface são os erros de timing. Eles ocorrem em sistemas em tempo real que usam uma memória compartilhada ou uma interface para passagem de mensagens.

- ☐ Todas as assertivas são verdadeiras.
- ☐ Somente II e III.
- ☐ Somente I e IV.
- ☐ Somente I, II e III.
- ☐ Nenhuma das alternativas anteriores.

Exercício

Considerando o escopo de testes de componentes, marque a alternativa que contém **somente** as assertivas VERDADEIRAS.

I - Os testes de componentes verificam erros de interface, que podem surgir em decorrência de interações entre componentes. **V**

II - Um dos tipos de erros de interface é o mau uso da interface. Este erro ocorre pois componente chamador de outro componente desconhece a especificação da interface deste (ex: chamar uma função que necessita de um vetor ordenado passando como parâmetro um vetor desordenado). **F**

III - Outro tipo de erro de interface é o mau-entendimento da interface. Este erro ocorre pois componente chamador de outro componente conhece o comportamento esperado do componente, porém comete um erro ao utilizar sua interface (ex: chamada de função com parâmetros errados). **F**

IV - O último tipo de erro de interface são os erros de timing. Eles ocorrem em sistemas em tempo real que usam uma memória compartilhada ou uma interface para passagem de mensagens. **V**

- ☐ Todas as assertivas são verdadeiras.
- ☐ Somente II e III.
- ☒ Somente I e IV.
- ☐ Somente I, II e III.
- ☐ Nenhuma das alternativas anteriores.

Testes de release

- Uma versão completa do sistema é testada
 - **Objetivo:** verificar se o sistema atende aos requisitos dos ***stakeholders do sistema*** (usuários e demais interessados na operação do sistema)
 - **Quem executa os testes?**
 - Equipe de teste independente (que não esteve envolvida com o desenvolvimento do sistema)

Teste de release vs Teste de sistema

- Testes de release são **testes de validação**
 - **Objetivo:** convencer o fornecedor do sistema de que o sistema é bom o suficiente para uso

Teste de release vs Teste de sistema

- Testes de release são **testes de validação**
 - **Objetivo:** convencer o fornecedor do sistema de que o sistema é bom o suficiente para uso
 - Atende aos seus requisitos (de sistema e usuários finais)
 - Mostrar que o sistema não falha durante o uso normal, fornecendo funcionalidade, desempenho, e confiança

Teste de release vs Teste de sistema

- Testes de release são **testes de validação**
 - **Objetivo:** convencer o fornecedor do sistema de que o sistema é bom o suficiente para uso
 - Atende aos seus requisitos (de sistema e usuários finais)
 - Mostrar que o sistema não falha durante o uso normal, fornecendo funcionalidade, desempenho, e confiança
- Testes de sistema são **testes de defeitos**
 - **Objetivo:** encontrar bugs / falhas

Testes de release

- São feitos considerando que o sistema é um caixa-preta
 - Comportamento do sistema só pode ser previsto quando fornecemos uma entrada para ele
 - Para cada entrada, há uma saída
 - O foco está na funcionalidade do sistema, e não na sua implementação
 - Esta forma de testar é chamada de **teste funcional**

Classificação de Testes de release

- **Testes baseados em requisitos**
 - Cada requisito é construído pensando em um conjunto de testes a ser executado
- **Testes de cenário**
 - Imaginar cenários típicos de uso e os usa para desenvolver casos de teste para o sistema
- **Testes de desempenho**
 - Ter certeza que o sistema consegue processar a carga a que se destina

Testes de release baseados em requisitos

- Abordagem sistemática para projeto de casos de teste
- Cada requisito deriva de um conjunto de testes, projetado especificamente para validar o requisito
- São **testes de validação**
- **Ex:** sistema de prescrições médicas com os seguintes requisitos:

Se um paciente é alérgico a algum medicamento específico, uma prescrição para esse medicamento deve resultar em uma mensagem de aviso.

Se um médico opta por ignorar um aviso, ele deve justificar sua decisão.

Testes de cenário

- Consiste em imaginar **cenários típicos de uso** e usa-los para desenvolver casos de teste para o sistema
 - **Cenário de uso**: uma história que descreve uma maneira de usar o sistema

Testes de cenário

- Consiste em imaginar **cenários típicos de uso** e usa-los para desenvolver casos de teste para o sistema
 - **Cenário de uso:** uma história que descreve uma maneira de usar o sistema
 - Cenários devem ser realistas, e usuários reais do sistema devem ser capazes de se relacionar com eles
 - Permite testar vários requisitos dentro de um mesmo cenário de uso

Testes de cenário

- **Ex:** A enfermeira Kate vai utilizar o sistema de prescrição médica que vimos nos exemplos anteriores
 - Conforme Kate utiliza o sistema, ela irá cometer erros
 - Devemos anotar os erros, e o comportamento do sistema em resposta a esses erros
 - “O sistema apresentou falha?”
 - “O sistema teve desempenho aceitável?”
 - “Kate acessou áreas do sistema que não deveria?”

Testes de desempenho

- Assegurar que o sistema consegue processar a carga a que se destina
- Consiste em executar uma série de testes em que você aumenta a carga até que o desempenho do sistema se torne inaceitável
 - **Objetivo:**
 - Demonstrar que o sistema atende seus requisitos
 - Descobrir problemas e defeitos do sistema

Testes de desempenho

- Para testar se os requisitos de desempenho estão sendo alcançados, você pode ter de construir um **perfil operacional**.
 - **Perfil operacional**: análise do funcionamento do sistema que reflete o ambiente de trabalho real ao qual ele será submetido

Exemplo de análise de perfil operacional

- Seja um sistema que possui as seguintes características de funcionamento:
 - 90% dos arquivos salvos no HDD possuem no máximo 4 MB (**grupo A**)
 - 7% possuem até 2 MB (**grupo B**)
 - 3% possuem menos que 512 KB (**grupo C**)
- Nesse caso você tem de projetar o perfil operacional para que a maioria dos testes seja do **grupo A**

Testes de desempenho

- Construir perfis operacionais não é necessariamente a melhor escolha para testes de desempenho
 - **Estressar o sistema** se mostrou mais eficiente

Testes de desempenho

- Construir perfis operacionais não é necessariamente a melhor escolha para testes de desempenho
 - **Estressar o sistema** se mostrou mais eficiente
 - Projetar testes para os limites do sistema, fazendo demandas que estejam fora dos limites de projeto do software
 - **Ex:** seja um banco de dados, projetado para atender até 200 usuários ao mesmo tempo
 - Se conseguirmos mostrar que ele atende 300 usuários, esse banco de dados passou no teste de desempenho

Testes de usuário

- Os usuários ou potenciais usuários de um sistema testam o sistema
 - Pode conter testes formais (definido previamente) ou informais (cada usuário é livre para explorar o sistema)
 - **Objetivo:** decidir se o sistema deve ser aceito ou se é necessário um desenvolvimento adicional
 - **Quem executa os testes?**
 - Usuários

Testes de usuário

- **Porque testes de usuário são importantes?**
 - O desenvolvedor, por mais que se esforce, não consegue replicar o cenário de uso real do sistema
 - Testes no ambiente do desenvolvedor são inevitavelmente artificiais
 - **Ex:** Sistema hospitalar é usado em um ambiente clínico, com atividades imprevisíveis ocorrendo ao mesmo tempo (emergências, exames, procedimentos)

Classificação de Testes de usuário

- **Testes alfa**

- Usuários e equipe de desenvolvimento testam o software no local do desenvolvedor

- **Testes beta**

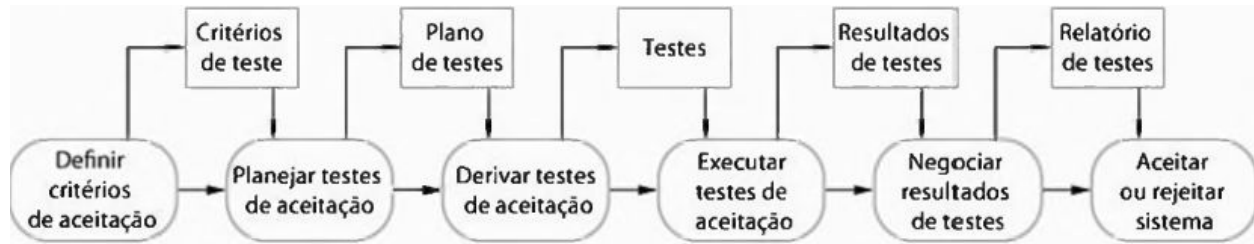
- Release do software é disponibilizado aos usuários para que possam experimentar e relatar problemas aos desenvolvedores

- **Testes de aceitação**

- Clientes testam um sistema para decidir se está ou não pronto para ser aceito e implantado no ambiente do cliente (produção)

Etapas dos Testes de aceitação

- **Definir critérios de aceitação:** requisitos para o aceite do software
- **Planejar testes de aceitação:** recursos e orçamento para os testes
- **Derivar testes de aceitação:** projeto dos testes de aceitação
- **Executar testes de aceitação:** executar testes de aceitação
- **Negociar resultados de teste:** que erros o cliente tolera no sistema?
- **Aceitar/rejeitar sistema:** decisão dos clientes e desenvolvedores



Depuração de software

- Caso um sistema apresente falhas em um dos testes, ele precisará passar por processos de **depuração**
- A **depuração** consiste em técnicas e processos para correção de falhas de software, tais como:
 - **Depuração por Força bruta (*brute force*)**
 - **Rastreamento (*backtracking*)**
 - **Eliminação da causa**

Depuração por Força bruta (*Brute force*)

- “Deixe o computador encontrar a falha (erro)”
 - Análise de logs e despejos de memória e demais informações na esperança de encontrar a falha
- **Problema:**
 - Processo muito lento
 - Grande volume de informações
 - Falha pode não ser encontrada



Depuração por Rastreamento (*Backtracking*)

- O código-fonte é investigado a partir do ponto onde a falha ou sintoma foi detectada até que a causa seja encontrada
 - Busca retroativa pelo código
- **Vantagem:**
 - Maior chance de encontrar a falha
- **Problema:**
 - Processo lento em softwares grandes

Depuração por Eliminação da causa

- É preparada uma lista de todas as causas possíveis (**hipóteses de causa**)
- São realizados testes para eliminar cada uma das hipóteses
 - Para isso, é preciso deduzir as possíveis origens da falha
- **Vantagem:**
 - Maior chance de encontrar a falha
- **Problema:**
 - Requer vários testes de hipóteses de causa

Exercício

Considerando os testes de release, marque a alternativa que contém **somente** as assertivas VERDADEIRAS.

I - Na abordagem de testes de release baseados em requisitos, cada conjunto de testes é construído para validar um requisito.

II - Na abordagem de testes de cenário, cada caso de teste é utilizado para construir cenários típicos de uso do sistema.

III - Na abordagem de testes de desempenho, o objetivo é garantir o sistema possui alto desempenho.

IV - Os testes de release baseados em requisitos, são testes de defeitos.

- ☐ Somente I.
- ☐ Somente I e II.
- ☐ Somente I, II e IV.
- ☐ Somente III.
- ☐ Nenhuma das assertivas é verdadeira.

Exercício

Considerando os testes de release, marque a alternativa que contém **somente** as assertivas VERDADEIRAS.

I - Na abordagem de testes de release baseados em requisitos, cada conjunto de testes é construído para validar um requisito. **F**

II - Na abordagem de testes de cenário, cada caso de teste é utilizado para construir cenários típicos de uso do sistema.

III - Na abordagem de testes de desempenho, o objetivo é garantir o sistema possui alto desempenho.

IV - Os testes de release baseados em requisitos, são testes de defeitos.

- ☐ Somente I.
- ☐ Somente I e II.
- ☐ Somente I, II e IV.
- ☐ Somente III.
- ☐ Nenhuma das assertivas é verdadeira.

Exercício

Considerando os testes de release, marque a alternativa que contém **somente** as assertivas VERDADEIRAS.

I - Na abordagem de testes de release baseados em requisitos, cada conjunto de testes é construído para validar um requisito. **F**

II - Na abordagem de testes de cenário, cada caso de teste é utilizado para construir cenários típicos de uso do sistema. **F**

III - Na abordagem de testes de desempenho, o objetivo é garantir o sistema possui alto desempenho.

IV - Os testes de release baseados em requisitos, são testes de defeitos.

- ☐ Somente I.
- ☐ Somente I e II.
- ☐ Somente I, II e IV.
- ☐ Somente III.
- ☐ Nenhuma das assertivas é verdadeira.

Exercício

Considerando os testes de release, marque a alternativa que contém **somente** as assertivas VERDADEIRAS.

I - Na abordagem de testes de release baseados em requisitos, cada conjunto de testes é construído para validar um requisito. **F**

II - Na abordagem de testes de cenário, cada caso de teste é utilizado para construir cenários típicos de uso do sistema. **F**

III - Na abordagem de testes de desempenho, o objetivo é garantir o sistema possui alto desempenho. **F**

IV - Os testes de release baseados em requisitos, são testes de defeitos.

- ☐ Somente I.
- ☐ Somente I e II.
- ☐ Somente I, II e IV.
- ☐ Somente III.
- ☐ Nenhuma das assertivas é verdadeira.

Exercício

Considerando os testes de release, marque a alternativa que contém **somente** as assertivas VERDADEIRAS.

I - Na abordagem de testes de release baseados em requisitos, cada conjunto de testes é construído para validar um requisito. **F**

II - Na abordagem de testes de cenário, cada caso de teste é utilizado para construir cenários típicos de uso do sistema. **F**

III - Na abordagem de testes de desempenho, o objetivo é garantir o sistema possui alto desempenho. **F**

IV - Os testes de release baseados em requisitos, são testes de defeitos. **F**

- ☐ Somente I.
- ☐ Somente I e II.
- ☐ Somente I, II e IV.
- ☐ Somente III.
- ☒ Nenhuma das assertivas é verdadeira.



Estudo dirigido

- Em duplas, realizem o estudo dirigido postado no Classroom
- Os resultados do estudo serão discutidos na próxima aula
 - Cada dupla tem 20 - 30 min para apresentar seus resultados
 - Discutir os achados da atividade
 - Complementar a discussão com ferramentas, técnicas e metodologias

Referencial Bibliográfico

- SOMMERVILLE, Ian. **Engenharia de Software**. 6. ed. São Paulo: Addison-Wesley, 2003.
- PRESSMAN, Roger S. **Engenharia de Software**. São Paulo: Makron Books, 1995.
- JUNIOR, H. E. **Engenharia de Software na Prática**. Novatec, 2010.