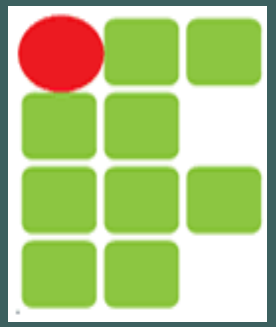


Teste de software

Aula 05

Métodos e Técnicas em Engenharia de Software

Profa. Alexandra

Prof. Thiago

Pós-Graduação em Gestão de Sistemas de Informação

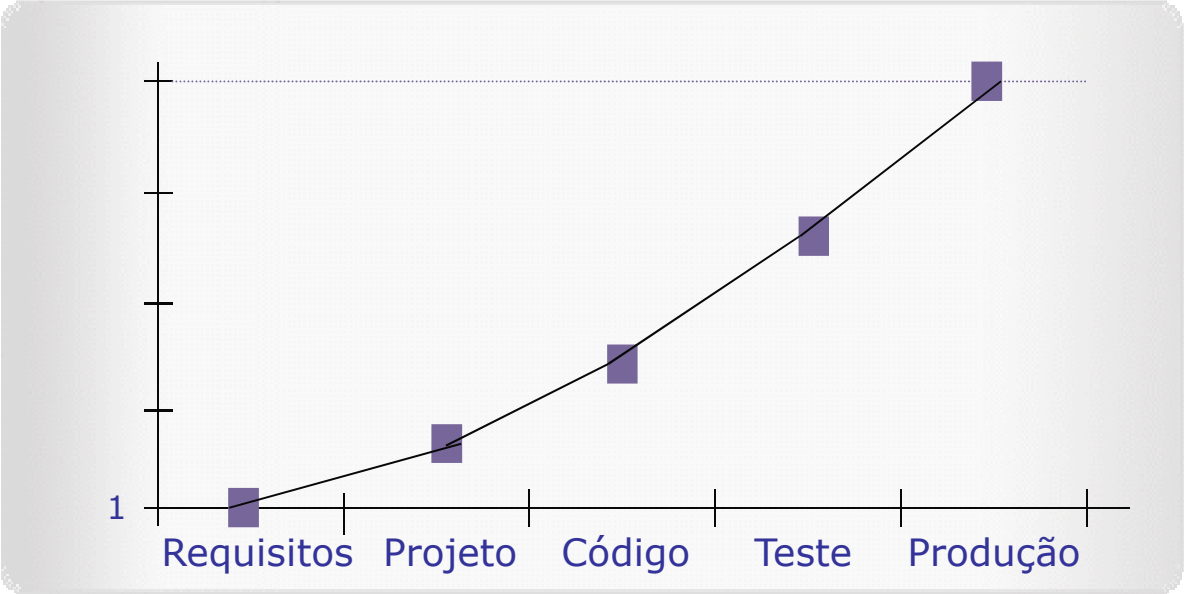
Objetivos

- Após esta aula, você deverá ser capaz de:
  - Conceituar a atividade de teste de software
  - Definir características da atividade de teste de software
  - Diferenciar a classificação de testes quanto ao acesso ao código, extensão e detalhe técnico da especificação
  - Descrever a aplicabilidade de testes automatizados
  - Conceituar o Desenvolvimento Orientado a Testes (TDD)

Motivação

Todo e qualquer tipo de erro gera custo

- Segundo Myers, quanto mais tarde descobrimos o erro, maior o custo
- Para cada erro não identificado em cada fase, o custo para correção é multiplicado por 10
- Erros em produção, além de custarem muito em termos financeiros, causam impactos significativos no negócio da empresa e na sua imagem



Verificação e validação

- Testes fazem parte do grupo de atividades de verificação e validação (V&V) de software

Verificação

Estamos construindo o produto *da forma certa*?

Busca checar se o produto desenvolvido (software) se adequa às especificações feitas

Validação

Estamos construindo *o produto certo*?

Verifica se os requisitos foram atendidos e se o software se comporta da forma esperada

Por que testar?

- A disciplina de teste atua como uma “prestadora de serviços” para as outras disciplinas do processo de desenvolvimento de sistemas

- Ela foca na análise e avaliação da qualidade do produto, que envolve:
  - Encontrar e documentar defeitos no software;
  - Verificar, validar e provar as premissas definidas nas especificações de requisitos;
  - Validar se um software está trabalhando como foi projetado;
  - Validar se todos os requisitos foram implementados corretamente;

Por que testar?

- Dentre os objetivos em que estão baseados os serviços que compõem o Teste de Software, podemos destacar:
  - A medição da qualidade do software;
  - O aumento na confiança na qualidade do software;
  - O auxilio na estimativa de defeitos presentes e ainda não descobertos;
  - A prevenção de defeitos, determinando sua causa raiz.

Por que testar?

- Defeitos e suas causas tornam-se lições aprendidas que:
  - Incrementam a experiência do time de desenvolvimento e de testes
  - Previnem que os mesmos erros sejam repetidos em projetos futuros

Fundamentos de Teste

Conceito

- Teste é a execução de um sistema ou componente, por meios automáticos ou manuais, para verificar se ele atende a sua especificação ou para identificar as diferenças entre os resultados obtidos e os esperados (IEEE90).

Objetivo

- Encontrar defeitos ou Não-Conformidades, segundo um conjunto de requisitos.

Fundamentos de Teste

Testar não é o mesmo que "*Debugar*"

- Teste: atividade que visa encontrar defeitos e mostrar falhas causadas por defeitos.

- Debug: é uma atividade que visa:
  - identificar a causa de um defeito;
  - reparar o defeito e;
  - verificar que o defeito foi sanado.

Princípios fundamentais do teste

- Um teste, na sua definição, deve levar em consideração cinco princípios, a saber:
  - **Princípio da cobertura**
  - **Princípio do teste exaustivo**
  - **Princípio da dependência do contexto**
  - **Princípio do teste antecipado**
  - **Princípio do cluster de defeitos**

Princípios fundamentais do teste

Princípio da Cobertura

- O teste visa descobrir e corrigir defeitos em cenários e condições de uso previamente determinados;

- O processo de teste visa provar que, nos cenários e condições cobertos pelos casos de teste:
  - Nenhum defeito é mais encontrado, ou
  - Foram encontrados e corrigidos.

- **Assim, em última instância, cada teste deve ser rastreável a um requisito do software**

Princípios fundamentais do teste

Princípio da Cobertura

- Teste de Regressão
  - Evoluções ou alterações em software pode apresentar defeito quando em conjunto com componentes inalterados, causando defeitos neles;
  - Quando isso ocorre, diz-se que o sistema em teste **regrediu*;***
  - O objetivo do Teste de Regressão é garantir que o sistema mantém a sua operacionalidade após uma intervenção (inclusão de novas funcionalidades ou manutenção corretiva);

Princípios fundamentais do teste

Princípio do Teste Exaustivo

- Testar todas as condições não é viável;
- Para funções complexas:
  - Requer uma quantidade enorme de recursos;
  - É muito caro e toma muito tempo.
- Para planejar o teste deve-se levar em conta:
  - O nível de risco inerente a cada funcionalidade;
  - Restrições do Projeto, tais como tempo, recursos e orçamento.
- A **Análise de Risco** deve ser usada para determinar o quanto testar em cada componente e funcionalidade.

Princípios fundamentais do teste

Princípio da Dependência do Contexto

- Teste de Software é Dependente de Contexto, ou seja, depende da natureza ou do tipo do mesmo;
  - Exemplo: o teste de um software crítico para a segurança de pessoas é diferente do teste de um software de *e-commerce*

- O que muda?
  - O tempo e recursos dedicados ao teste durante o projeto de desenvolvimento do software;
  - O nível de defeitos aceitável, visto sob várias métricas.

Princípios fundamentais do teste

Princípio do Teste Antecipado

- O teste não se inicia quando o código do software começa a ser entregue;
- O teste do software deve ocorrer em sincronia com o desenvolvimento e deve se iniciar o mais cedo possível dentro do ciclo de desenvolvimento
- Nesse princípio se baseia o Gráfico de Myers apresentado no início desta aula.

Princípios fundamentais do teste

Princípio do Cluster de Defeitos

- O teste espera encontrar a maioria dos defeitos em poucos componentes do software;
- Este princípio é também conhecido por Regra de Pareto, ou Princípio 80-20
  - 80% das falhas em 20% dos componentes

Princípios fundamentais do teste

Algumas Verdades sobre o Teste:

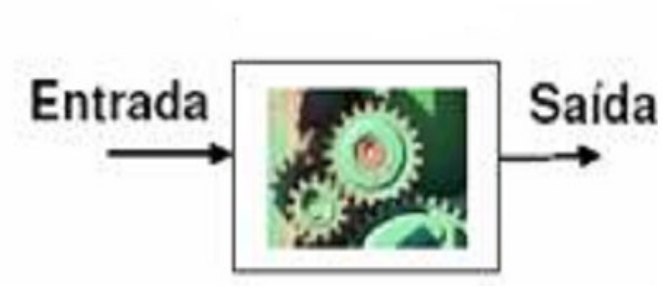
- Testes precisam evoluir, mudar ou incrementar. Perdem seu efeito com o tempo.
- Um processo de teste que corrige todos os defeitos encontrados não prova que a aplicação em si é a aplicação que o usuário espera (“the right system”).

Tipos de teste

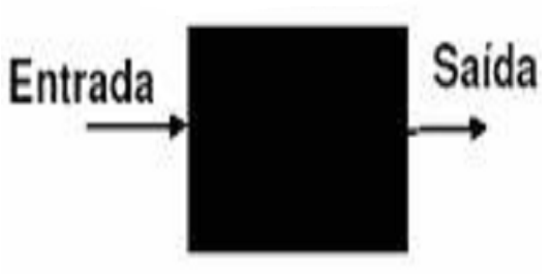
- Os testes podem ser classificados em função de algumas características, dentre elas:
  - **Acesso ao código fonte** → teste caixa-branca e caixa-preta
  - **Extensão da funcionalidade testada** → teste unitário e de integração
  - **Nível técnico de especificação dos casos de teste** → teste funcional e de aceitação

Teste em função do acesso ao código-fonte

- Teste Caixa-Branca (White Box): Acesso ao código



- Teste Caixa-Preta (Black Box): Focado nos requisitos funcionais do software



Teste em função do acesso ao código-fonte

Teste caixa-preta

- Considera o programa realmente como uma “caixa-preta”, ou seja, o teste é desenvolvido sem que o analista de teste tenha acesso ao código fonte

- Dessa forma, os casos de teste são baseados na especificação funcional do sistema (p. ex., casos de uso)
  - Vantagem: planejamento de testes pode começar bem cedo no processo de desenvolvimento

Teste em função do acesso ao código-fonte

Teste Caixa-Branca

- Tem como objetivo exercitar todos os possíveis “caminhos” no código do software
  - Em outras palavras, cada linha do programa deve ser testada em algum momento de algum caso de teste
- Baseado na inspeção do código, do ponto de vista do desenvolvedor

Testes em função da extensão da funcionalidade testada

Teste unitário

- Objetivo: busca por defeitos em um módulo / classe em particular
- Conduzido pelo próprio desenvolvedor ao longo do desenvolvimento do sistema
- É também uma técnica de apoio para depurar alterações no módulo
- Geralmente utiliza a abordagem caixa-branca
- **Responsável: Programador**

Testes em função da extensão da funcionalidade testada

Teste de integração

- Objetivo: teste da operação conjunta dos módulos
- Identifica erros nas interfaces entre os módulos, problema que não é identificável no teste unitário
- Emprega a abordagem de caixa-preta
- Recomenda-se que a integração dos módulos seja feita de forma incremental; o sistema não precisa estar necessariamente pronto
- **Responsável: Analista de Testes / Arquiteto**

Testes em função do nível técnico da especificação

Teste Funcional

- Tem como objetivo encontrar defeitos nas funcionalidades do sistema.
- Tradicionalmente é executado no software como um todo (os componentes do software estão totalmente integrados).
- Em geral, é organizado por um ou mais dos seguintes itens de especificação do sistema:
  - Casos de uso
  - Funções de Negócio
  - Requisitos do Usuário
- A abordagem é de caixa preta e sua execução simula a interação do usuário final com a aplicação.
- **Responsável: Analista de Teste**

Testes em função do nível técnico da especificação

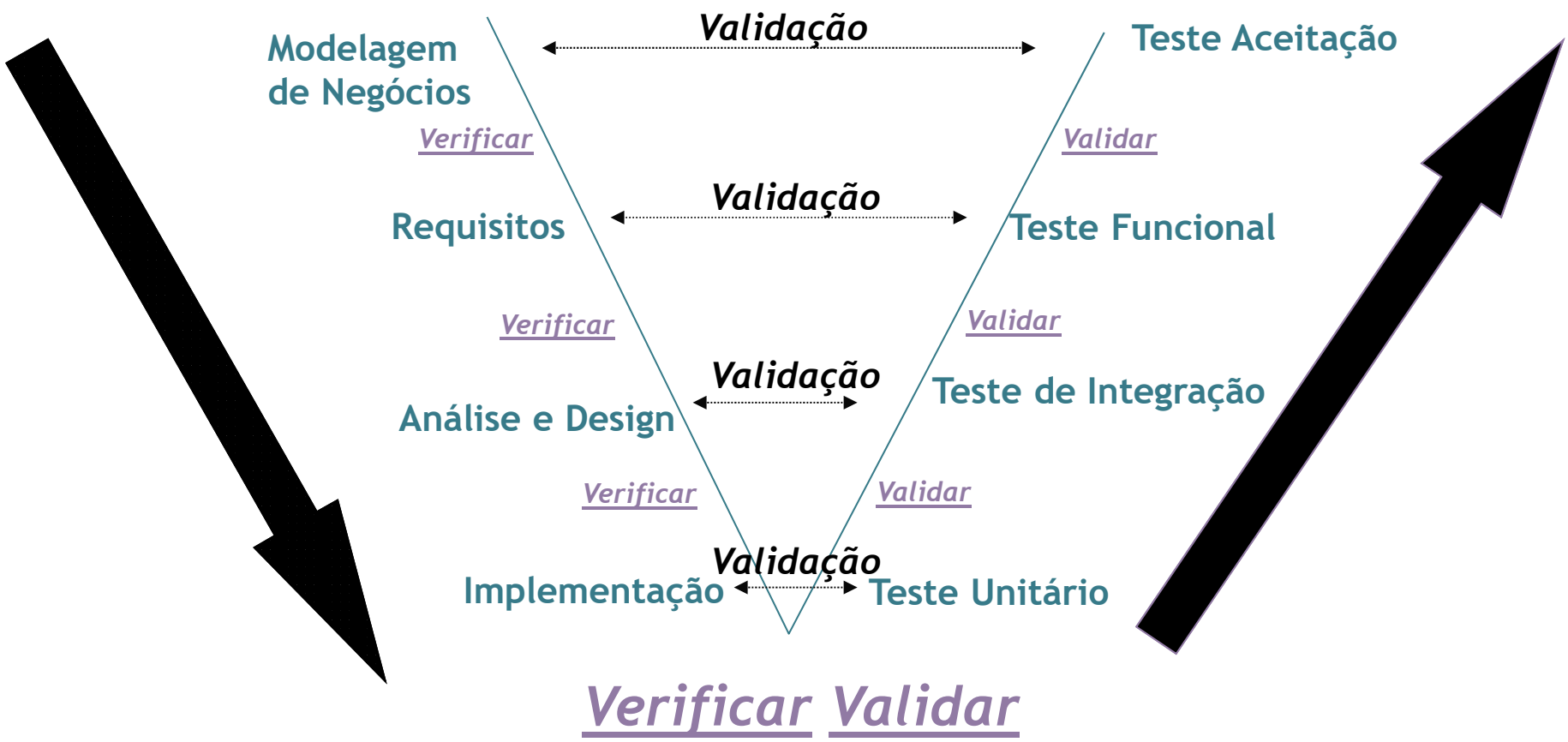
Teste de Aceitação

- Tem como objetivo testar as funcionalidades do sistema a partir do ponto de vista do cliente;
- É uma 'prova' para o cliente de que o software funciona;
- A abordagem é de caixa preta e sua execução é feita a partir da interação do usuário final com a aplicação.
- **Responsável: Usuário Final.**

Planejamento dos testes

- Podemos associar cada tipo de teste a uma atividade do processo de desenvolvimento
- Dessa forma, o planejamento dos testes pode ser antecipado e acontece em paralelo com cada atividade do processo em si
- A execução dos testes, por sua vez, acontece na ordem inversa do planejamento
- Esse modelo de planejamento e execução de testes é chamado Modelo “U” (ou “V”)

Planejamento dos testes



Tipos de Execução de Testes

Teste Manual

- Execução do Sistema feita manualmente pelo Programador ou Executor de Teste, através de situações de teste pré-definidas.

- Características:
  - Concentra-se nos testes baseado na análise de impacto;
  - Pode ser limitado, e assim não confiável;
  - Processo consumidor de tempo e enfadonho;

Tipos de Execução de Testes

Teste Automatizado

- Realiza testes no software de forma automatizada, com base em scripts ou programas de testes previamente especificados, verificando o resultado através de relatórios.

- Características :
  - Um teste pode ser executado facilmente várias vezes;
  - Maior amplitude no escopo do teste, testando as funcionalidades antigas, novas e alteradas;
  - O teste é confiável;
  - Agilidade no processo de execução de Testes.

Testes Automatizados

Tipos de Teste: Baseados na Interface Gráfica (Capture / Playback)

- São realizados por meio da interface gráfica da aplicação.
- Normalmente a ferramenta de automação fornece um recurso para capturar (*Capture*) as ações do usuário enquanto o usuário estiver usando a aplicação.
- Essas ações são traduzidas para comandos na linguagem de *script* suportada pela ferramenta de automação para que então possam ser reproduzidas (*Playback*) posteriormente.
- **Exemplo de Ferramenta: Rational Robot, Atlassian Spira**

Testes Automatizados

Tipos de Teste: Dirigidos a Dados (Data-Driven)

- Representam um refinamento dos testes baseados na interface gráfica. Neste tipo de teste é utilizado um mecanismo para auxiliar a execução de testes que executam as mesmas ações repetidamente, porém com dados diferentes.

- Uma das principais vantagens dessa abordagem é a reutilização dos *scripts*, o que consequentemente diminui a complexidade e o tempo de manutenção.

Testes Automatizados

Tipo de Teste: Dirigidos à Palavra-Chave (Keyword-Driven)

- Este tipo de teste automatizado é realizados por meio da interface gráfica da aplicação. No entanto, os testes são baseados em palavras-chaves (*keywords*). Normalmente a ferramenta de automação oferece um conjunto pré-definido de palavras-chaves para permitir a criação dos testes.

- Cada palavra-chave é um comando em alto nível (praticamente em linguagem nativa) que representa uma ação do usuário. Dessa forma, os testes são facilmente entendidos em virtude do seu alto nível de abstração.

Testes Automatizados

Tipo de Teste: Baseados na Linha de Comando (Command Line Interface - CLI)

- Uma Interface de Linha de Comando (*Command Line Interface* - CLI) fornece um mecanismo no qual o usuário pode interagir com a aplicação por meio de um *prompt* ou *shell* do sistema operacional, tornando-o independente da interface gráfica da aplicação

- Isso é feito através da interpretação dos comandos e parâmetros (que exercita a lógica de negócio da aplicação), da execução da função selecionada, juntamente com a apresentação do resultado.

- **Exemplo de Ferramenta: Load Runner.**

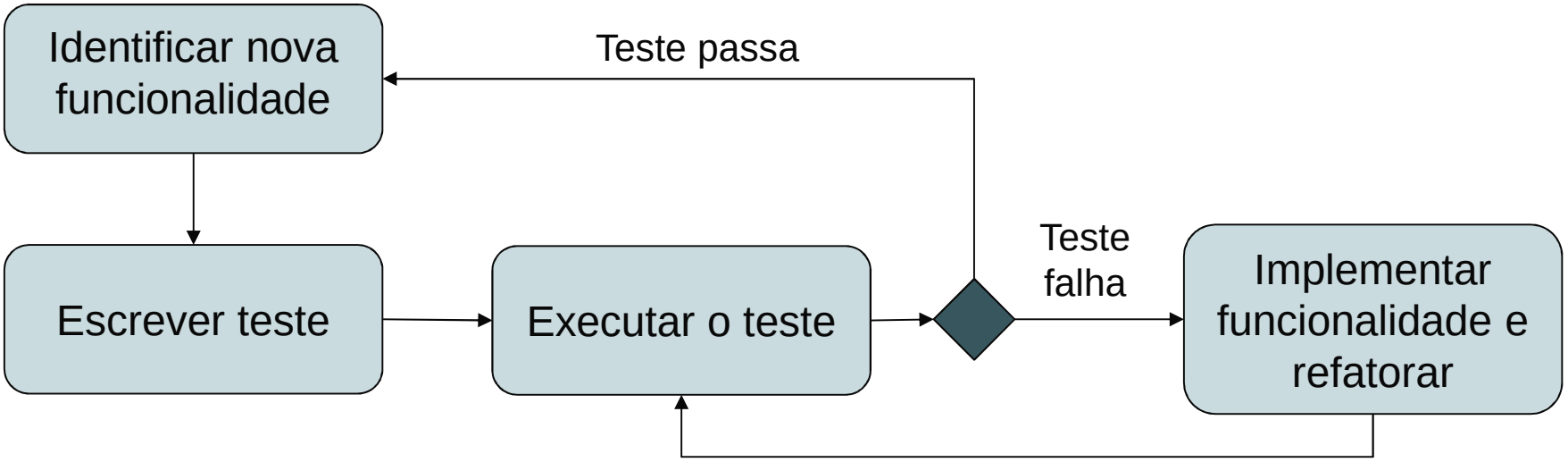
Testes Automatizados

Tipo de Teste: Test Harness

- O Test Harness é um tipo de automação de testes baseado na Lógica de Negócio que prega o uso racional e inteligente da automação.
- Pode ser implementado por meio de um pequeno programa construído para testar uma API ou uma interface de linha de comando.
- Neste tipo de teste automatizado não importa o meio no qual o teste será realizado (contanto que não ocorra interação com a interface gráfica).
- O objetivo é exercitar as funcionalidades críticas da aplicação que exigem dezenas e milhares de cálculos ou variações virtualmente impossíveis (ou demoradas) para serem testadas por meios normais.
- **Exemplo de Ferramenta: EasyMock, JUnit**

Desenvolvimento Orientado a Testes (TDD)

- Abordagem que intercala testes e desenvolvimento de código, muito associada aos métodos ágeis
- A escrita do teste (automatizado) precede a programação da funcionalidade em si



Desenvolvimento Orientado a Testes (TDD)

Vantagens potenciais:

- Cobertura de código
- Teste de regressão
- Depuração simplificada
- Documentação do sistema