ENGENHARIA DE SOFTWARE II

Análise e Desenvolvimento de Sistemas

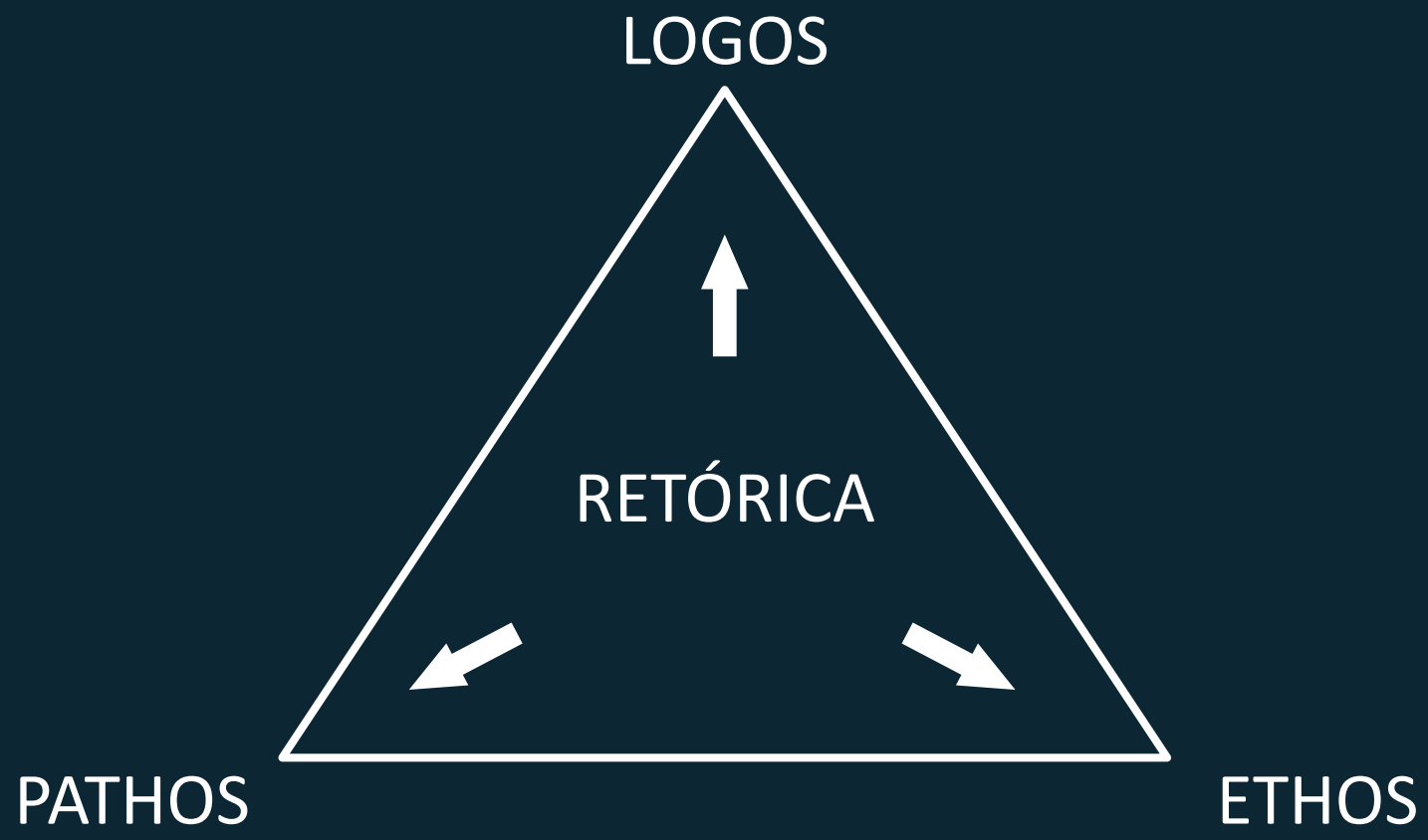
DISCIPLINA

- Qualidade de software;
- Gerência de Configuração e Manutenção;
- Teste de Software;
- Métricas.

RETÓRICA

- Em sua obra **Retórica** (*Ars Rhetorica*), Aristóteles definiu esse conceito como uma técnica de persuasão, capaz de funcionar independente do público, adequando o discurso a cada tipo de audiência;
- Três categorias por meio das quais pode-se apelar à audiência compõem o triângulo retórico ou a Tríade Aristotélica.

RETÓRICA



RETÓRICA

- ***Logos:*** emprego da razão e do raciocínio (dedutivo ou indutivo) para construção do argumento;
- ***Pathos:*** emprego de apelos emocionais com a intenção de alterar a reação da plateia;
- ***Ethos:*** credibilidade e autoridade sobre determinado tema, identidade, valor.

TESTES E QUALIDADE

ENGENHARIA DE SOFTWARE II

Análise e Desenvolvimento de Sistemas

TESTE DE SOFTWARE

- A atividade de **teste de software** é um elemento crítico da garantia da **qualidade de software** e representa a última revisão de especificação de projeto e codificação.
- O destaque crescente do software como elemento de sistema e os custos envolvidos associados às falhas de software são forças propulsoras para uma atividade de teste cuidadosa e bem planejada.
- Não é incomum que uma organização de software gaste **40% do esforço** de projeto total em teste.

FUNDAMENTOS

- A atividade de teste constitui uma anomalia interessante para o engenheiro de software (desenvolvedor): durante a fase de definição e desenvolvimento anteriores, o desenvolvedor tenta construir o software; agora, na fase de testes, este desenvolvedor cria uma série de casos de testes que têm a intenção de “demolir” o software que ele construiu.

OBJETIVOS

- Segundo MYERS (2011), os objetivos de se executar teste são:

✔ A atividade de teste é o processo de executar um programa com a intenção de descobrir erro;

✔ Um bom **caso de teste** é aquele que tem uma elevada probabilidade de revelar um erro ainda não descoberto;

✔ Um teste bem sucedido é aquele que revela um erro ainda não descoberto.

A atividade de teste não pode mostrar a ausência de bugs!

CASOS DE TESTE

- MYERS (2011) propõe que sejam produzidos casos de teste para um programa com as seguintes características:

✔ O programa lê três valores inteiros informados pelo usuário. Os três valores representam os comprimentos dos lados de um triângulo. O programa mostra uma mensagem que informa se o triângulo é escaleno, isósceles ou equilátero.

✔ Lembre-se que um triângulo escaleno é aquele no qual não há dois lados iguais, um triângulo isósceles é aquele que possui dois lados iguais, e o triângulo equilátero possui três lados de comprimentos iguais.

CASOS DE TESTE

- Agora responda às perguntas, somando 1 ponto para cada resposta “SIM”:

1. Você tem um caso de teste que representa um triângulo escaleno válido? Note que casos de teste como 1, 2, 3 e 2, 5, 10 não garantem uma resposta positiva, pois um triângulo que tenha essas três dimensões não é válido.
2. Você tem um caso de teste que representa um triângulo equilátero válido?
3. Você tem um caso de teste que representa um triângulo isósceles válido? Note que um caso de teste que contenha 2, 2, 4 não conta, pois não é um triângulo válido.

CASOS DE TESTE

4. Você tem ao menos três casos de teste que representam triângulos isósceles válidos de tal forma que você testou as três permutações de dois lados iguais (tais como 3, 3,4; 3, 4, 3; 4, 3, 3)?
5. Você tem um caso de teste no qual um dos lados tem valor zero?
6. Você tem um caso de teste no qual um dos lados tem valor negativo?
7. Você tem um caso de teste com três inteiros maiores do que zero, tais como a soma de dois números é igual ao terceiro?
8. Você tem ao menos três casos de teste na categoria 7, no qual você tentou todas as três permutações onde o comprimento de um lado é igual à soma dos outros dois (exemplo: 1, 2, 3; 1, 3, 2; e 3, 1, 2)?

CASOS DE TESTE

9. Você tem um caso de teste com três inteiros maiores do que zero de forma que a soma de dois dos valores seja menor do que o terceiro (tais como 1, 2, 4 ou 12, 15, 30)?
10. Você tem ao menos três casos de teste na categoria 9 de tal forma que você tentou as três permutações (exemplo: 1, 2, 4; 1, 4, 2; e 4, 1, 2)?
11. Você tem um caso de teste no qual todos os lados são zero?
12. Você tem ao menos um caso de teste especificando valores não inteiros (tais como: 2.5, 3.5, 5.5)?

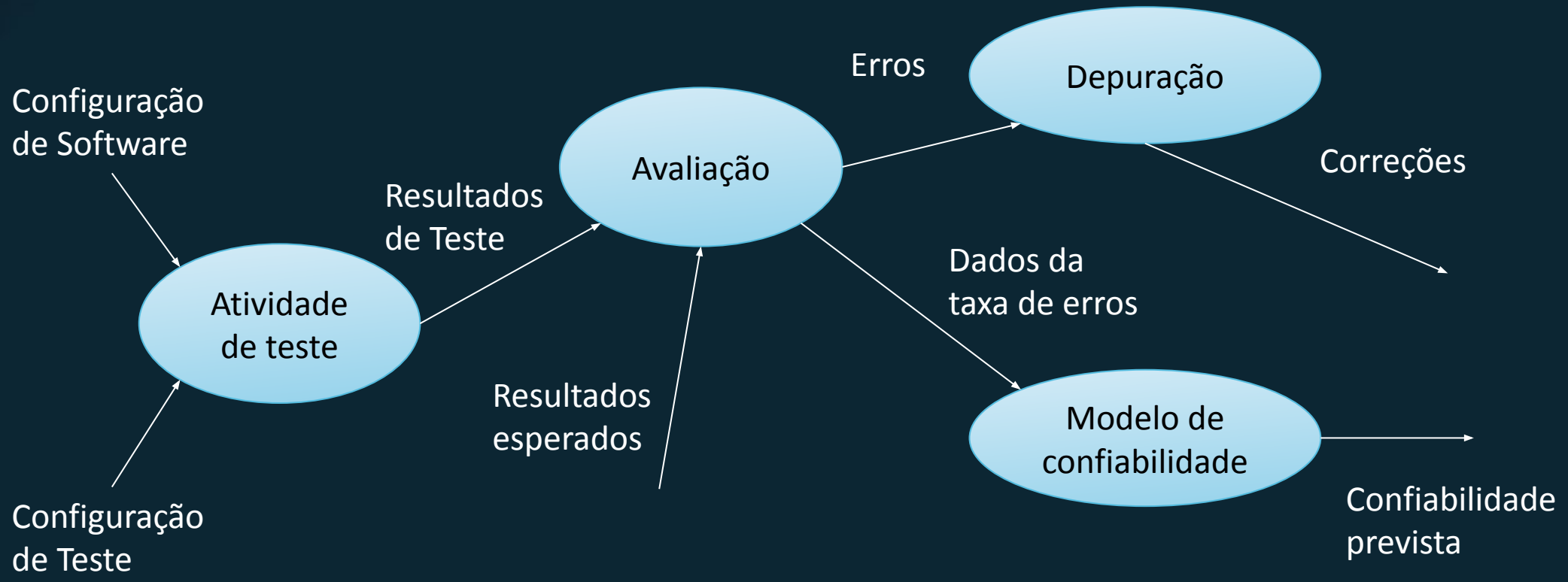
CASOS DE TESTE

13. Você tem ao menos um caso de teste especificando número errado de valores (dois, ao invés de três inteiros, por exemplo?
14. Para cada caso de teste, você especificou a saída esperada do programa de acordo com os valores de entrada?

- Antes que você se desespere com seu desempenho, considere:

✔ Desenvolvedores experientes costumam marcar, em média, apenas 7.8 (o máximo seria 14) !

FLUXO DE INFORMAÇÃO DE TESTE



FLUXO DE INFORMAÇÃO DE TESTE

- Testes são realizados e todos os resultados são **avaliados**, ou seja, os resultados dos testes são comparados com os resultados esperados.
- Quando dados errôneos são descobertos, infere-se a presença de erro e inicia o processo de **depuração**.

FERRAMENTAS DE TESTES AUTOMATIZADOS

- Com base no fato de que aproximadamente 40% do tempo de projeto é destinado a testes, pesquisas são difundidas afim de obter ferramentas que possam encurtar esse tempo sem cair a qualidade de testes;
- MILLER (1979) aponta algumas categorias de ferramentas de testes automatizados:

FERRAMENTAS DE TESTES AUTOMATIZADOS

- **Analisadores estáticos:** capazes de testar a comprovação de afirmações estáticas: sobre a estrutura e o formato de um programa;
- **Auditores de código:** São filtros de propósito especial usados para verificar a qualidade do software a fim de garantir que ele atenda a padrões mínimos de codificação;
- **Geradores de dados de teste:** Sistemas de análise automatizados que auxiliam o usuário a selecionar dados de teste que fazem um programa comportar-se de uma forma particular.

LEIS DE MURPHY

1. Se uma coisa pode sair errado, sairá.
2. Todas as coisas sempre sairão erradas, é só questão de tempo.
3. Entregues a si mesmas, as coisas irão sempre de mal a pior.
4. A natureza está sempre a favor da falha oculta.
5. Se tudo parece estar andando bem, é porque você não olhou direito.

NORMAS RELACIONADAS A TESTES

ENGENHARIA DE SOFTWARE II

Análise e Desenvolvimento de Sistemas

NORMAS

- Normas são documentos publicados por organizações profissionais responsáveis por padronizar determinadas atividades, processos, dispositivos, etc.
- O motivo da existência de normas é para que se tenha uma padronização nas formas com que são realizados processos, produzidos documentos, e para que cada termo tenha o mesmo significado e seja entendido em diferentes situações facilitando a comunicação tanto entre as equipes internas de uma organização, quanto entre organizações diferentes.

NORMAS EM TESTES

- Nessas normas são descritas as etapas de processos, entradas e saídas esperadas de cada etapa, as formas e conteúdos de documentações, bem como outras informações necessárias a atividade de teste.
- As principais instituições utilizadas como referência para normas em testes de software são:

✔ **ISO**: *International Organization for Standardization* (Organização Internacional para Padronização);

✔ **IEC**: *International Electrotechnical Commission* (Comissão Eletrotécnica Internacional);

✔ **IEEE***: *Institute of Electrical and Electronics Engineers* (Instituto de Engenheiros Eletricistas e Eletrônicos).

* *pronuncia-se I-3-E*

✔ **ABNT:** Associação Brasileira de Normas Técnicas.

ISO 9126-1-SOFTWARE QUALITY CHARACTERISTICS (2003)

- ISO/IEC 9126-1 (2001) – Características de qualidade de software (NBR 13596)

✔ Substituída pela ISO/IEC 25010 (2011).

- Esse padrão define um modelo de qualidade o qual pode ser aplicado a qualquer tipo de software sem fazer especificações sobre os requisitos desse produto.

- O objetivo desse padrão é proporcionar um framework para avaliação da qualidade dos produtos de software.

NORMA ISO/IEC 9126-1

- A norma define seis características de qualidade de software:

✔ **Funcionalidade**: finalidade do produto;

✔ **Usabilidade**: esforço para utilizar o produto;

✔ **Confiabilidade**: frequência de falhas e recuperabilidade;

✔ **Eficiência**: desempenho;

✔ **Manutenibilidade**: esforço necessário para modificar o produto;

✔ **Portabilidade**: capacidade de transferir o produto para outros ambientes.

NORMA ISO/IEC 25010

- Substituta da ISO/IEC 9126-1, define mais duas, totalizando oito características:

1. **Funcionalidade**: é a capacidade que o software tem de prover funções que atendam aos requisitos implícitos e explícitos. Pode ser entendido como “o que o software faz”.
2. **Portabilidade**: capacidade da transferência de um produto de software de um ambiente para outro.
3. **Confiabilidade**: capacidade que o produto de software tem de repetir sua funcionalidade dadas as mesmas condições antes aplicadas. Ou ainda, é a capacidade que um produto tem de executar determinada função sem sofrer desgaste ou envelhecimento.

NORMA ISO/IEC 25010

4. **Manutenibilidade:** capacidade que o produto de software possui de ser modificado. As modificações incluem correções, melhorias ou adaptações.
5. **Usabilidade:** capacidade que o produto tem de ser compreendido pelo usuário. O quão fácil é para que o usuário aprenda e possa utilizar esse produto.
6. **Eficiência:** capacidade do produto de apresentar um desempenho satisfatório quando lhe é proporcionado recursos suficientes.
7. **Segurança:** confidencialidade e integridade;
8. **Compatibilidade:** coexistência e interoperabilidade.

ISO/IEC 12207 – SYSTEMS AND SOFTWARE ENGINEERING – SOFTWARE LIFE CYCLE PROCESSES (2008)

- Essa norma descreve a arquitetura dos processos de ciclo de vida de software, sem especificar os detalhes de implementação ou execução das atividades e tarefas incluídas nos processos.
- Estabelece ainda um framework para processos de ciclo de vida de software, com terminologias bem definidas, que podem ser referenciadas por desenvolvedores de software.

ISO/IEC 12207 – SYSTEMS AND SOFTWARE ENGINEERING – SOFTWARE LIFE CYCLE PROCESSES (2008)

- Nele são descritos processos, atividades e tarefas que são executadas durante a aquisição de um produto de software ou serviço, e durante o fornecimento, desenvolvimento, operação, manutenção e o descarte de produtos de software.
- Em determinados contextos software pode incluir porções de hardware.

ISO/IEC 12207 – SYSTEMS AND SOFTWARE ENGINEERING – SOFTWARE LIFE CYCLE PROCESSES (2008)

- O padrão ISO/IEC 12207 define ao todo 25 processos, 95 atividades e 325 tarefas;

✔ Tarefa: ação com entradas e saídas. Pode ser um requisito (deve, *shall*), recomendação (deveria, *should*) ou permissão (pode, *may*).

✔ Atividade: conjunto de tarefas.

- Nesse padrão, as seções que terão mais importância no âmbito do teste são:

✔ 6.4.6 na qual é discutido o processo de qualificação de teste de sistema;

✔ 7.1.7 referente ao processo de qualificação de teste de software.

IEEE 1012 – STANDARD FOR SOFTWARE VERIFICATION AND VALIDATION (2004)

- O IEEE Std 1012-2004 é um padrão de processo que define os processos de verificação e validação em termos de atividades específicas e tarefas relacionadas a cada um deles.
- De acordo com o padrão IEEE 1012-2012:

  “Processos de verificação e validação (V&V) são utilizados para determinar se o desenvolvimento de produtos de uma certa atividade está em conformidade com os requisitos daquela atividade e se o produto satisfaz sua intenção de uso e as necessidades do usuário”.

IEEE 1012 – STANDARD FOR SOFTWARE VERIFICATION AND VALIDATION (2004)

- Também define os conteúdos do plano de V&V incluindo exemplos e formas os quais servem de diretivas.
- Esse padrão se aplica ao desenvolvimento de software, manutenção e reuso.
- Os processos da V&V são definidos conforme o padrão ISO 12207 e referencia os processos de ciclo de vida do software, sendo compatível com todos os modelos de ciclo de vida.

IEEE 1012 – STANDARD FOR SOFTWARE VERIFICATION AND VALIDATION (2004)

- Os objetivos desse padrão são:

✔ Estabelecer um framework para os processos de V&V, atividades e tarefas que contribuam de maneira direta para todos os processos de ciclo de vida de software. Inclusos nesses estão os processos de aquisição, fornecimento, desenvolvimento, operação e manutenção;

✔ Definir as tarefas de verificação e validação, conjunto de entradas e conjunto de saídas;

✔ Identificar as tarefas mínimas de V&V correspondentes a um esquema de integridade de 4 níveis;

✔ Definir o conteúdo de um plano de V&V.

IEEE 829 – STANDARD FOR SOFTWARE AND SYSTEM TEST DOCUMENTATION (2008)

- A IEEE 829 provê um conjunto de padrões reconhecidos internacionalmente em documentação para planejamento de teste.
- Ela foi desenvolvida especialmente para esse fim e é aplicável a cada fase do ciclo de teste de software, incluindo teste de sistema e aceitação.
- A norma define a padronização da documentação de teste tanto em tempo de desenvolvimento, quanto as versões seguintes após completo esse desenvolvimento do produto de software.

IEEE 829 – STANDARD FOR SOFTWARE AND SYSTEM TEST DOCUMENTATION (2008)

- Não são definidos nesse padrão metodologias específicas para se realizar o teste, bem como não são definidas técnicas, nem ferramentas para fazê-lo.
- A escolha das metodologias, técnicas e ferramentas cabe ao gerente de teste juntamente com a equipe de gerência de projetos.
- Alguns outros documentos de teste não descritos nessa norma podem ser necessários para a composição de uma boa documentação dependendo da metodologia utilizada, ficando a cargo do gerente de teste da instituição optar por quais desses documentos devem ser incluídos.

IEEE 829 – STANDARD FOR SOFTWARE AND SYSTEM TEST DOCUMENTATION (2008)

- O padrão IEEE 829 é composto por oito documentos divididos em três áreas principais, sendo elas:

1. Plano de teste;
2. Especificação de Teste;
3. Relatórios.

ATIVIDADE PRÁTICA

REFERÊNCIAS

- MILLER, E.. ***Automated Tools for Software Engineering***. New York: IEEE, Computer Society Press, 1979.
- MYERS, G. J.; BADGETT, T.; SANDLER, C. ***The Art of Software Testing***. 3rd ed. New Jersey: John Wiley & Sons, 2011.