



Unidade 10

Fundamentos de Teste de Software

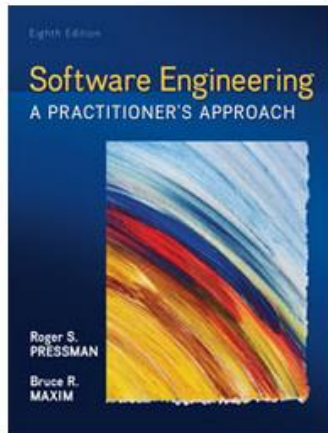


Prof. Aparecido V. de Freitas
Doutor em Engenharia
da Computação pela EPUVSP

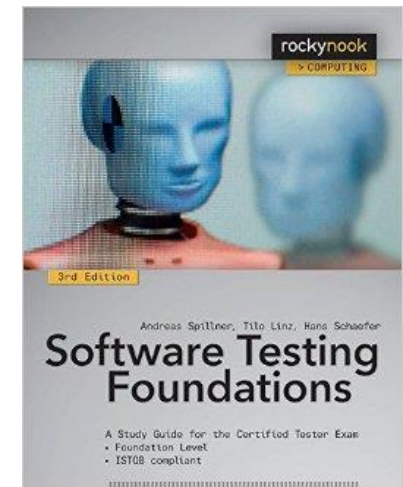
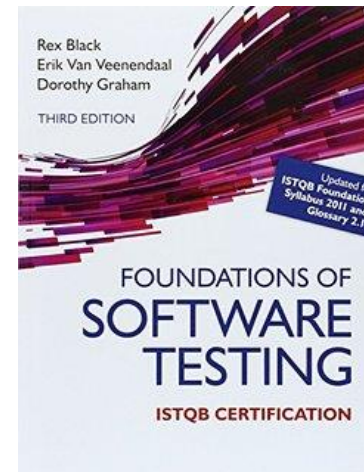
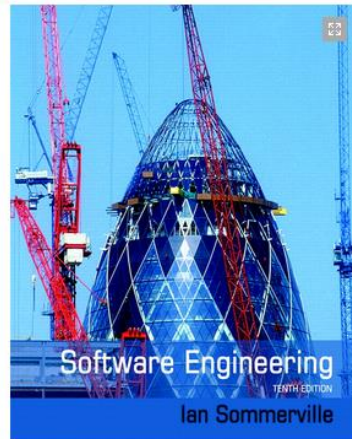


Bibliografia

- **Software Engineering – A Practitioner's Approach – Roger S. Pressman – Eight Edition – 2014**
- **Software Engineering – Ian Sommerville – 10th edition - 2015**
- Engenharia de Software – Uma abordagem profissional – Roger Pressman - McGraw Hill, Sétima Edição - 2011
- Engenharia de Software – Ian Sommerville – Nona Edição – Addison Wesley, 2007
- **Software Testing Foundations – 4th edition – Andreas Spillner, Tito Linz, Hans Schaefer – 2014**
- **Foundations of Software Testing – Third Edition – Rex Black – ISTQB Certification**



[Software Engineering: A Practitioner's Approach, 8/e](#)





Introdução



- ⊕ Procedimentos de teste e avaliação de software são custosos na prática;
- ⊕ Estima-se **25%** a **50%** de custo e tempo de desenvolvimento [Koomen, 99].
- ⊕ **ISTQB** – International Software Testing Qualifications Board
- ⊕ **CTFL** – Certified Tester Foundation Level
- ⊕ **CTAL-TM** - Advanced Level Test Manager
- ⊕ **CTAL-TA** - Advanced Level Test Analyst
- ⊕ **CTAL-TTA** - Advanced Level Technical Test Analyst
- ⊕ www.abramti.org





CTFL – Certified Tester Foundation Level

- ⊕ Composto de 40 questões de múltipla escolha;
- ⊕ **Requerido 70% de acerto (≥ 28 questões);**
- ⊕ 1 hora de duração (1m30s por questão);
- ⊕ Realizado em várias cidades brasileiras;
- ⊕ 1ª sexta-feira de Fevereiro, Maio, Agosto e Novembro;
- ⊕ Baseado no Syllabus – www.istqb.org





CTFL – Certified Tester Foundation Level

- ⊕ Capítulo 1 – **6** questões
- ⊕ Capítulo 2 – **6** questões
- ⊕ Capítulo 3 – **4** questões
- ⊕ Capítulo 4 – **12** questões
- ⊕ Capítulo 5 – **8** questões
- ⊕ Capítulo 6 – **4** questões





Termos e Motivação



- ⊕ Software não é fisicamente tangível;
- ⊕ Examinação direta não é possível;
- ⊕ A única forma de se examinar software é pela leitura da documentação e código;
- ⊕ O comportamento dinâmico do software só pode ser testado pela execução do software em um computador. Seu comportamento deve ser comparado a seus requisitos;
- ⊕ Teste de software contribui para redução dos riscos, pois defeitos podem ser encontrados durante o teste;





Porque é necessário testar ?

- ⊕ Porque todos nós cometemos erros;
- ⊕ Alguns desses erros não são importantes, mas outros são perigosos e caros;
- ⊕ Nossos erros podem ser sistemáticos, assim outra pessoa talvez possa ter mais facilidade em encontrar nossos erros;
- ⊕ É assim importante, considerar quando os erros podem causar problemas...





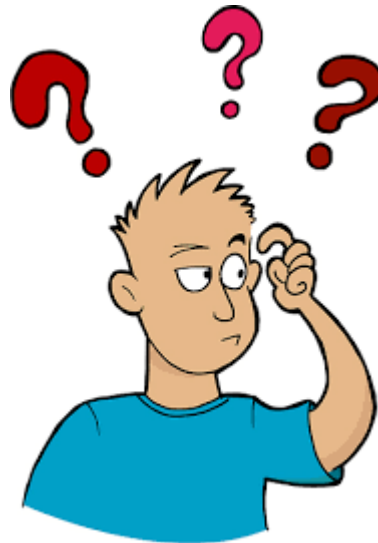
Danos dos Bugs

- ⊕ **Prejuízos** financeiros e da imagem;
- ⊕ **Empresas / Organizações** – Atrasos , Perda de Confiança, Vendas;
- ⊕ **Pessoas** – Constrangimentos, Risco de Vida, Acidentes;
- ⊕ **Governos** – Vulnerabilidade de Informações, Decisões Estratégicas Incorretas;
- ⊕ **Meio Ambiente** – Alertas atrasados, Poluição, Desperdício de Recursos.





Um erro de digitação em um texto na WEB é irrelevante ?





Erros podem causar diferentes impactos

- ⊕ Num website pessoal, talvez seja irrelevante. É fácil de ser corrigido e poucas pessoas irão notar (provavelmente...)
- ⊕ Para uma grande corporação, esses pequenos erros de digitação podem afetar a imagem da instituição, passando a impressão de não profissionalismo.



ERRO → DEFEITO → FALHA



- ⊕ Seres humanos cometem erros (enganos). Pessoas são falíveis;
- ⊕ As pessoas que construíram o software podem ter cometidos erros;
- ⊕ Esses erros podem produzir **DEFEITOS** ou **bugs** no software;
- ⊕ Quando o software é executado, esses **DEFEITOS** podem causar **FALHAS (danos ou problemas)**;
- ⊕ Nem todos os **DEFEITOS** resultam em **FALHAS**. Alguns podem permanecer dormentes no código e podem nunca ser notificados;
- ⊕ **DEFEITOS** são estáticos e **FALHAS** são dinâmicas.
- ⊕ **FALHAS** geram insatisfação com a **QUALIDADE**.





Causas dos defeitos de software



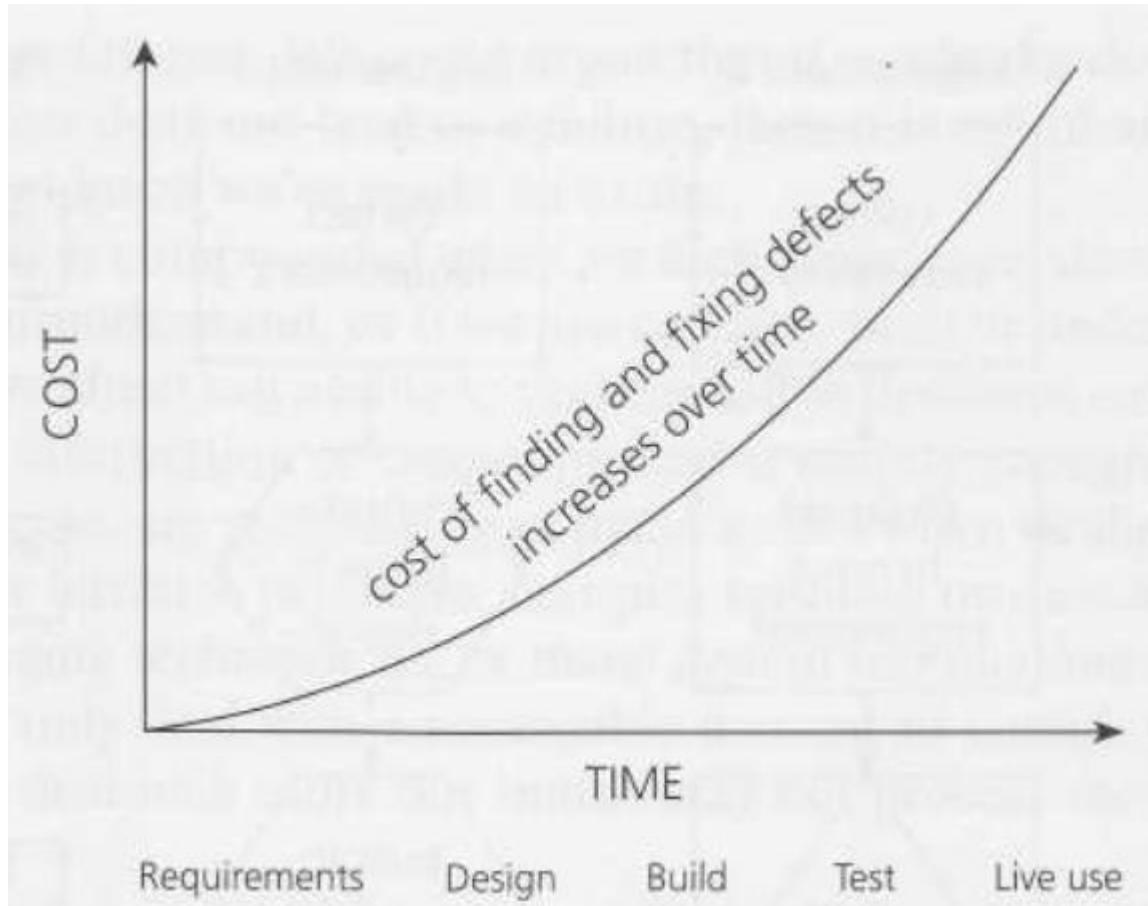
⊕ DEFEITOS / BUGS

- Seres humanos são passíveis de erros (cliente, Programador, Analista, Testador, etc);
- Pressão de Prazo;
- Complexidade de código ou infraestrutura;
- Mudanças de Tecnologia;
- Muitas interações do sistema (Mudanças).





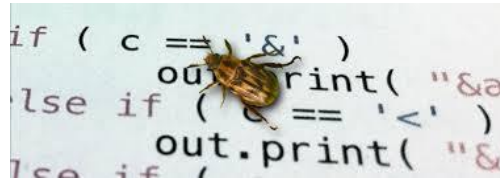
Custo dos defeitos



Fonte: Foundations of Software Testing – Third Edition – Rex Black – ISTQB Certification



Papel do Teste no Desenvolvimento de Software



- ⊕ Rigorosos **testes em software** e **documentação** podem reduzir os riscos de ocorrência de problemas no ambiente operacional;
- ⊕ Contribui para a **qualidade** dos sistemas de software se os defeitos forem corrigidos antes da implantação na produção;
- ⊕ Testes de software também podem ser necessários para atender requisitos **contratuais** ou **legais** ou determinados **padrões de mercado**.





Teste e Qualidade

- ⊕ Teste auxilia na medição da qualidade do software em termos de **defeitos** encontrados;
- ⊕ Testes bem projetados e executados, descobrirão defeitos se estiverem presentes;
- ⊕ Testes mal executados podem descobrir poucos defeitos e nos dar uma falsa impressão de qualidade.
- ⊕ Quando os testes encontram defeitos, a qualidade do software aumenta quando estes são corrigidos.
- ⊕ Aderência às características da Qualidade (ISO 9126) – Confiabilidade, Usabilidade, Eficiência, Manutenibilidade, Portabilidade.
- ⊕ Atendimento dos Requisitos Funcionais (Negócio) ou Não Funcionais (Técnico).



Resultados de Teste

- ⊕ Afetam os níveis de Qualidade do Software;
- ⊕ Expõe o Nível de Risco do Sistema;
- ⊕ Definem ações e Melhoria.





Verificação e Validação de Software (V & V)

⊕ VALIDAÇÃO:

- Estamos construindo o produto que agrega valor ao usuário ?
- O software está de acordo com as necessidades do usuário ?
- A especificação do software está correta ?

⊕ VERIFICAÇÃO:

- Estamos construindo o produto corretamente ?
- O software está de acordo com a especificação ?
- O produto foi corretamente construído ?





Quanto de teste é suficiente ?

- 3 escolhas: Testar **TUDO**, Testar **NADA** ou Testar uma **parte** do software;
- **O que é melhor ?**





Princípio de Teste: Teste exaustivo é impossível

- Quantos testes são necessários para um teste completo em um campo numérico de 1 dígito?
- Há dez valores numéricos possíveis.
- Mas, há também outros valores que devemos **REJEITAR**. Há 26 caracteres alfabéticos maiúsculos, 26 minúsculos e pelo menos 6 caracteres especiais e de pontuação, bem como espaço em branco, que devem ser rejeitados.
- Assim, deveria haver pelo menos **68 testes** para o exemplo de um campo numérico de 1 dígito.





Princípio de Teste: Teste exaustivo é impossível

- Suponha uma tela com 15 campos de input, cada qual tendo 5 valores possíveis;
- Então, testar todos os valores possíveis, significa testar todas as combinações de entrada válidas: **5^{15}** combinações que representa **30.517.578.125** testes.





Como decidir sobre a quantidade de teste a ser efetuada ?



Quanto de teste é suficiente ?

- Para decidir quanto teste é suficiente, deve-se levar em consideração **o nível do risco**, incluindo **risco técnico**, do **negócio** e do **projeto**, além das restrições do projeto como **tempo** e **orçamento**.

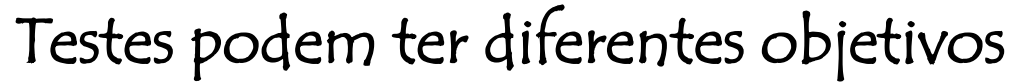




O que é Teste ?

- ✓ Testar o programa durante a sua execução é apenas uma parte do **TESTE**;
- ✓ **TESTE** é um processo e não apenas uma única atividade;
- ✓ Há uma série de atividades envolvidas no processo de **TESTE**;
- ✓ Existem atividades de teste antes e depois da fase de execução;
- ✓ Por exemplo: planejamento e controle, escolha das condições de teste, modelagem dos casos de teste, checagem dos resultados, avaliação do critério de conclusão, geração de relatórios sobre o processo de teste e sobre sistema alvo e encerramento ou conclusão (ex.: após a finalização de uma fase de teste);
- ✓ Teste também inclui revisão de documentos (incluindo o código fonte) e análise estática.





-





Objetivos de Teste – Testes durante Desenvolvimento

✓ Teste de Componentes, Integração e de Sistemas

- Principal objetivo é causar o maior número de falhas possíveis, de modo que os defeitos no software possam ser identificados e resolvidos.



✓ Teste de Aceite (Homologação ou Certificação)

- Principal objetivo pode ser confirmar se o sistema está funcionando conforme o esperado, ou seja, prover a confiabilidade de que esteja de acordo com os requisitos.

- Teste de **Componente (ou Unitário)**: (método, classe ou função) – Feito pelo Desenvolvedor
- Teste de **Integração**: (comunicação entre 2 ou mais componentes) – Feito pelo Líder de projeto
- Teste de **Sistema**: Software pronto, teste como um todo. Feito pelo Analista de Teste
- Teste de **Manutenção (Regressão)**: Verificar se não foram inseridos erros durante as mudanças
- Teste de **Operação**: Avaliar características como confiabilidade e disponibilidade





Hã diferença entre Depuração de Código e Teste ?





Depuração de Código e Teste

- ✓ Testes podem demonstrar falhas que são causadas por defeitos;
- ✓ Depuração de código é a atividade de desenvolvimento que identifica a causa de um defeito, repara o código e checa se os defeitos foram corrigidos corretamente;
- ✓ Testadores **testam** e desenvolvedores **depuram**.





Teste de Confirmação

- ✓ Realizado por um testador para certificar se a **falha** foi eliminada.
- ✓ Popularmente chamado de **Reteste**.
- ✓ O Teste de Confirmação se refere ao **Reteste** de um ponto específico do software que apresentou alguma falha prévia.
- ✓ O Teste de Confirmação deve comprovar que o defeito foi sanado.
- ✓ O conserto da falha específica **pode** acarretar problemas em outros pontos do software. Para essa situação, um teste mais geral do sistema é necessário (**Teste de Regressão**).
- ✓ O Teste de Regressão, em geral, é feito com processos de **automação de testes**.





Tarefas antes, durante e depois da execução do Teste

- ✓ Planejamento;
- ✓ Controle (Acompanhamento do Plano);
- ✓ Identificar as condições de teste;
- ✓ Elaborar casos de teste;
- ✓ Avaliar e informar os resultados de teste;
- ✓ Observar os critérios de entrada (quando começar a executar);
- ✓ Observar os critérios de saída (quando parar ou pausar os testes);
- ✓ Garantir o reaproveitamento dos artefatos no encerramento dos testes;
- ✓ Aprender a testar melhor.





Os sete princípios do Teste



Princípio 1 – Teste demonstra a presença de defeitos

- ✓ O teste pode demonstrar a presença de defeitos, mas não pode provar que eles não existem.
- ✓ O teste reduz a probabilidade que os defeitos permaneçam em um software, mas mesmo se nenhum defeito for encontrado, não prova que ele esteja perfeito.
- ✓ Teste **NÃO** garante ZERO defeitos.





Princípio 2 – Teste exaustivo é impossível

- ✓ Testar tudo (todas as combinações de entradas e pré-condições) não é viável, exceto para casos triviais;
- ✓ Em vez do teste exaustivo, riscos e prioridades são levados em consideração para dar foco aos esforços de teste.





Princípio 3 – Teste antecipado

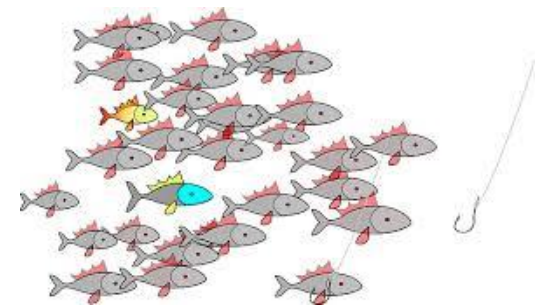
- ✓ Teste deve ser o mais antecipado possível no ciclo de desenvolvimento de software;
- ✓ Teste deve ser focado em objetivos definidos;
- ✓ Quanto mais cedo se encontrar um defeito, menos custo será necessário para sua identificação e correção. (Regra de 10 de Myers).





Princípio 4 – Agrupamento de Defeitos

- ✓ Um número pequeno de módulos contém a maioria dos defeitos descobertos durante o teste;
- ✓ Um número reduzido de módulos pode entregar ou exibir a maioria das falhas operacionais;
- ✓ Bugs gostam de ficar juntos. Os bugs estão distribuídos de forma heterogênea. Alguns módulos têm mais defeitos do que outros.





Princípio 5 – Paradoxo da pesticida

- ✓ Pesticida mata os insetos, mas ao se aplicar o mesmo pesticida muitas vezes, eventualmente os que sobrarem serão imunes;
- ✓ Pode ocorrer que de um mesmo conjunto de testes que são repetidos várias vezes não encontrem novos defeitos após um determinado momento;
- ✓ Assim, casos de teste precisam ser frequentemente revisados e atualizados;
- ✓ Um novo e diferente conjunto de testes precisa ser escrito para exercitar diferentes partes do software, para se aumentar a probabilidade de se encontrar mais erros.
- ✓ Inove regularmente os testes !





Princípio 6 – Teste depende de contexto

- ✓ Testes são realizados de forma diferente dependendo do contexto;
- ✓ Por exemplo, software de segurança crítica é testado de forma diferente de um software de comércio eletrônico.





Princípio 7 – A ilusão da ausência de erros

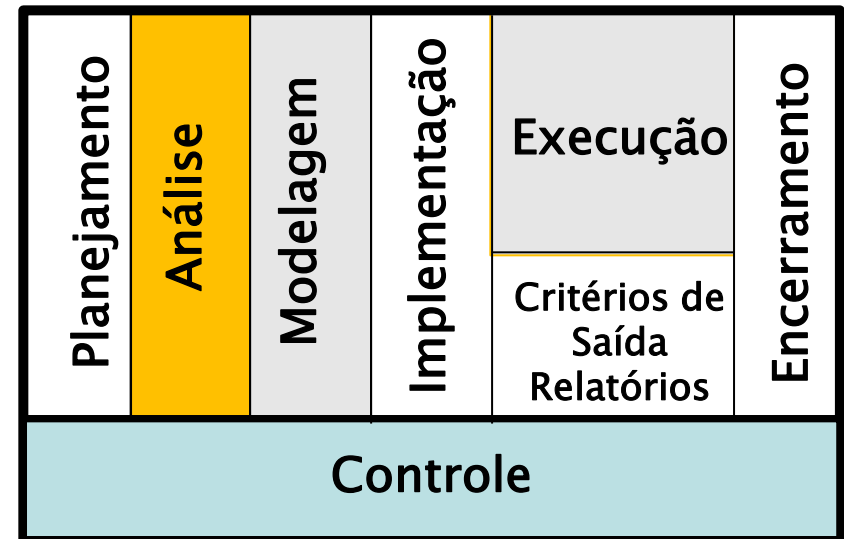
- ✓ Encontrar e consertar defeitos não ajuda se o software construído **não** atende às **expectativas e necessidades dos usuários**.





Fundamentos do Processo de Teste – ISTQB

- ✓ Planejamento e Controle do Teste
- ✓ Análise e Modelagem do Teste
- ✓ Implementação e Execução de Teste
- ✓ Avaliação do critério de saída e Relatório
- ✓ Atividades de Encerramento de Teste



Linha de tempo do projeto →

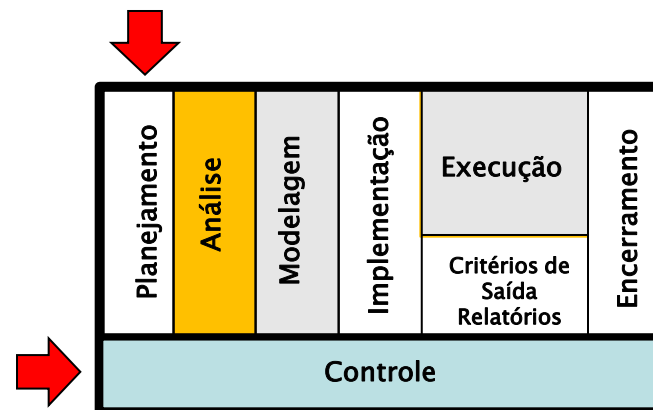


Observação: Essas etapas podem se sobrepor, serem concorrentes e/ou interagir !



Planejamento e Controle do Teste

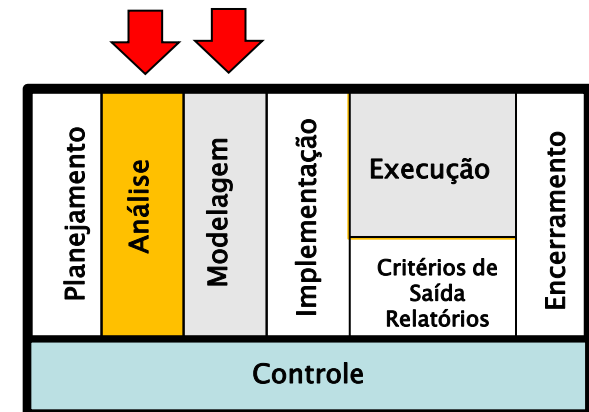
- ✓ O Planejamento é a atividade que consiste em definir os objetivos e especificar as atividades de forma a alcançá-los;
- ✓ O controle do Teste é a atividade constante que compara o progresso atual contra o que foi planejado, comunicando o status e os desvios do plano;
- ✓ O controle do Teste envolve as ações necessárias para alcançar a missão e objetivos do projeto.
- ✓ Para um controle efetivo, o teste deverá ser monitorado durante todo o projeto;
- ✓ O planejamento de Teste leva em consideração o retorno de informações das atividades de Monitoração e Controle.





Análise e Modelagem do Teste

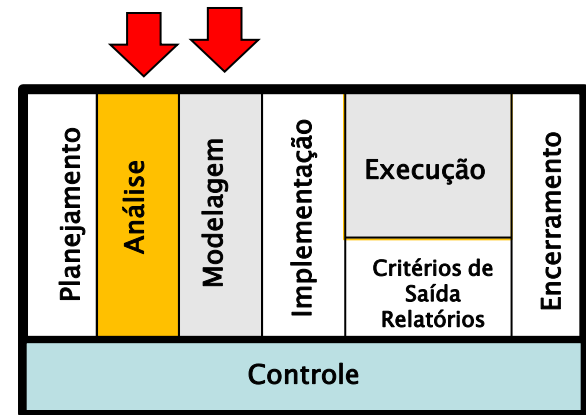
- ✓ Atividades onde os objetivos gerais do teste são transformados em condições e modelos tangíveis;
- ✓ Compreendem as seguintes atividades:
 - Revisar a base de testes (como requisitos, nível de integridade do software, nível de risco, arquitetura, modelagem, interfaces);
 - Avaliar a testabilidade dos requisitos e dos sistema;
 - Identificar e priorizar as condições ou requisitos de testes e dados de testes baseados na análise dos itens de teste, na especificação, no comportamento e na estrutura.





Modelagem e Análise do Teste

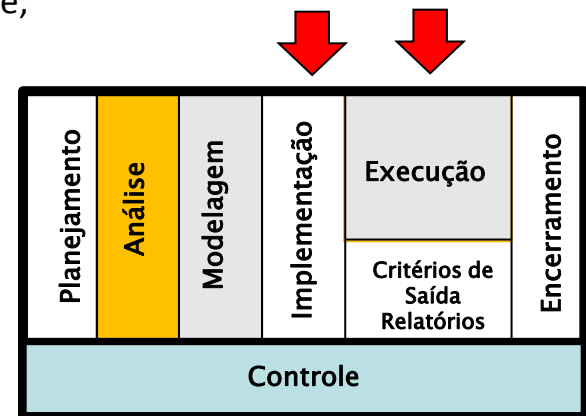
- ✓ Projetar e priorizar os casos de teste de alto nível ;
- ✓ Identificar as necessidades de dados para teste suportando as condições e casos de teste;
- ✓ Planejar a preparação do ambiente de teste e identificar a infraestrutura e ferramentas necessárias;
- ✓ Criar rastreabilidade bidirecional entre os requisitos e casos de teste.





Implementação e Execução de Teste

- ✓ Atividade onde os procedimentos ou os scripts de teste são especificados pela combinação dos casos de teste em uma ordem particular, incluindo todas as outras informações necessárias para a execução do teste, o ambiente é preparado e os testes são executados;
- ✓ Principais atividades:
 - Implementar casos de teste;
 - Criar dados de teste;
 - Verificar se o ambiente de teste está preparado;
 - Verificar rastreabilidade entre casos de teste e base de teste;
 - Executar testes (manualmente ou com ferramentas);
 - Comparar resultados;
 - Reportar discrepâncias;
 - Repetir testes como resultado de ações corretivas.





Confirmação x Regressão

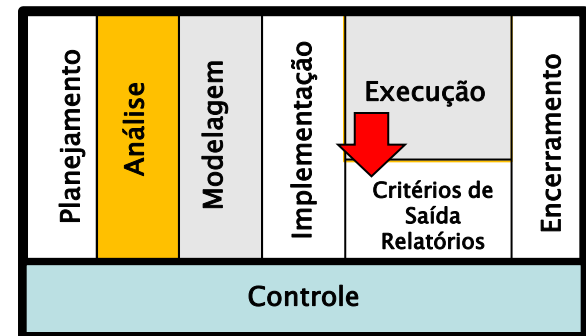
- ✓ **Teste de Confirmação:** Re-execução de um teste que falhou previamente quando da confirmação de uma correção (**Reteste**).
- ✓ **Teste de Regressão:** Execução ou Re-execução de testes a fim de certificar que os defeitos não foram introduzidos em áreas do software que não foram modificadas, ou que a correção do defeito não desvendou outros defeitos (**Side Effects**).





Avaliação de critério de saída e Relatórios

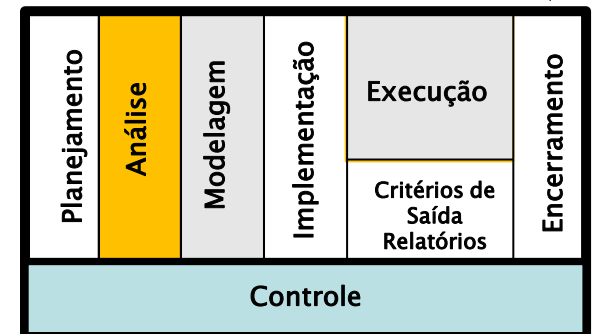
- ✓ Atividade onde a execução do teste é **avaliada** mediante **objetivos definidos**;
- ✓ Fornece a base para se tomar a decisão de parada do teste;
- ✓ Composta pelas seguintes atividades principais:
 - Checar registros de teste (logs);
 - Avaliar se são necessários testes adicionais ou se critério de saída de teste deve ser alterado;
 - Elaborar relatório de teste resumido para os interessados (stakeholders).





Atividades de Encerramento de Teste

- ✓ São coletados os dados de todas as atividades para consolidar a experiência, testware, fatos e números.
- ✓ São compostas pelas seguintes atividades principais:
 - Checar quais entregáveis foram realmente entregues;
 - Fechar relatórios de incidentes;
 - Documentar aceite do sistema;
 - Entregar o **testware** para manutenção da organização;
 - Analisar as lições aprendidas;
 - Utilizar informações coletadas para melhorar a maturidade de teste.





A Psicologia do Teste

- ✓ Formas diferentes de pensar:

Desenvolvedor quando há defeitos...



Testador fica feliz quando encontra defeitos...





Formas diferentes de pensar

- ✓ Desenvolvedores são aptos a testarem seus **próprios códigos e projetos**.
- ✓ Testadores têm uma **visão independente**, profissional e treinada sobre os recursos de teste;
- ✓ A separação da responsabilidade de testar ajuda a focar o esforço e provê benefícios adicionais;
- ✓ É difícil testar o que se constrói.





Independência de Teste

- ✓ Certo grau de independência (sem a influência do autor) muitas vezes representa uma forma eficiente de se encontrar defeitos e falhas;
- ✓ Independência não significa simplesmente uma substituição: desenvolvedores podem encontrar defeitos de maneira eficiente. (Cuidado para não virar alibi...)



Quem faz não checa ...

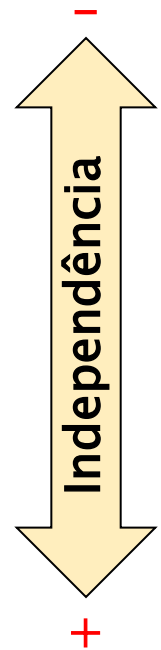


Quem checa não faz ...



Níveis de Independência

- ✓ Teste elaborado por quem escreveu o software (baixo nível de independência);
- ✓ Teste elaborado por outras pessoas da mesma equipe de desenvolvimento;
- ✓ Teste elaborado por pessoas de um grupo organizacional diferente (equipe independente de teste);
- ✓ Teste elaborado por pessoas de diferentes empresas (terceirizada ou certificada por um órgão externo, alto nível de independência);





Testar requer

- ✓ Curiosidade;
- ✓ Pessimismo profissional;
- ✓ Olha crítico;
- ✓ Atenção aos detalhes;
- ✓ Comunicação Eficiente com os Profissionais de Desenvolvimento;
- ✓ Experiência para encontrar Erros;
- ✓ Comunicar bugs de forma construtiva para evitar constrangimentos entre as equipes;





Eficiência de Teste

- ✓ Deve-se preterir testes em áreas de risco;
- ✓ **Engenheiros de Teste efetivos e eficientes**: Têm um talento para consumir tempo onde estão os bugs. Podem fazer uma completa isolamento do bug rapidamente;
- ✓ **Engenheiros de Teste Não efetivos e Ineficientes**: Escrevem testes para procurar bugs improváveis e de baixo impacto. Gastam horas pesquisando bugs triviais;
- ✓ Permanecer focado nos objetivos do projeto de teste.

Efetivo: alcançar uma meta

Eficiente: atingir resultado com pouco esforço





Habilidades do Testador

- ✓ **Leitura.** Especificações, e-mails, casos de teste;
- ✓ **Escrita.** Casos de teste, relatórios de bugs, documentação de teste, etc;
- ✓ **Estatística** e outras operações matemáticas;
- ✓ **Habilidades** pertinentes à Tecnologia, Projeto e Teste:
 - *Linguagens de Programação, Sistemas Operacionais, redes, Web, etc*
 - *Domínio do Negócio das aplicações;*
 - *Produção de Scripts, Automatização, Modelagem de Desempenho;*

