



EDUCAÇÃO E TREINAMENTO
 PARA ALAVANCAR A SUA CARREIRA

Curso e-learning
TESTER FOUNDATION

Atualizado para o
 Syllabus 2018



Preparatório para o exame **CTFL**
 (Certified Tester Foundation Level)
 do ISQB/BSTQB

Formação essencial de
 analistas em teste de software



Todos os direitos de cópia reservados. Não é permitida a distribuição física ou eletrônica deste material sem a permissão expressa do autor.

Versão: 1.0 Liberação: 26/11/18


Módulo 1: Fundamentos de teste de software					
Módulo 1 – Os sete princípios de testes					
CONTEUDO PROGRAMÁTICO					
Fundamentos de teste	O teste durante todo o ciclo de vida do software	Teste estático	Técnicas de teste	Gerenciamento do teste	Ferramentas de suporte ao teste
O que é teste?	Modelos de ciclo de vida	Noções básicas	Categorias de técnicas	Organização de teste	Considerações sobre ferramentas
Por que o teste é necessário?	Níveis de teste	Processo de revisão	Técnicas caixa-preta	Planejamento e estimativa de teste	Uso eficaz de ferramentas
Os 7 princípios do teste	Tipos de teste		Técnicas caixa-branca	Monitoramento e controle dos testes	
Processos de teste	Teste de manutenção		Técnicas baseadas na experiência	Gerenciamento configurações	
A psicologia do teste				Riscos e testes	
				Gerenciamento de defeitos	

Módulo 1: Fundamentos de teste de software

Objetivos de aprendizagem do módulo

1.3 Os sete princípios de testes

FL-1.3.1 (K2) Explicar os sete princípios de teste



TI.exames

TI.exames © Todos os direitos reservados. Proibida a redistribuição deste material. Slide 3

Módulo 1: Fundamentos do teste de software

Os sete princípios de testes

Uma série de princípios tem sido sugeridos ao longo de décadas para servir como diretrizes comuns para os testes.
Veja ao lado quais são eles:



```
graph TD; P((Princípios de testes)) --- 1((1. O teste mostra a presença de defeitos e não a sua ausência)); P --- 2((2. Testes exaustivos são impossíveis)); P --- 3((3. O teste inicial economiza tempo e dinheiro)); P --- 4((4. Defeitos se agrupam)); P --- 5((5. Cuidado com o paradoxo do pesticida)); P --- 6((6. O teste depende do contexto)); P --- 7((7. Ausência de erros é uma ilusão));
```

1. O teste mostra a presença de defeitos e não a sua ausência

2. Testes exaustivos são impossíveis

3. O teste inicial economiza tempo e dinheiro

4. Defeitos se agrupam

5. Cuidado com o paradoxo do pesticida

6. O teste depende do contexto

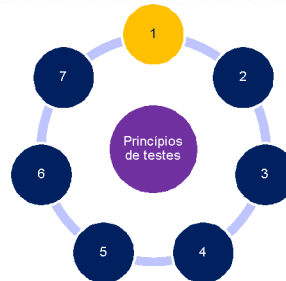
7. Ausência de erros é uma ilusão

TI.exames

Slide 4

1. O teste mostra a presença de defeitos e não a sua ausência

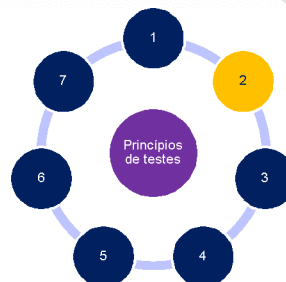
- O teste reduz a probabilidade de defeitos não descobertos permanecerem no software, mas, mesmo que não haja defeitos, isso não é uma prova de que eles não existem.
- Isso porque:
 - Dados de testes podem ter sido selecionados inadequadamente.
 - Pode haver alguma circunstância especial em que software falhe.
 - Exemplo: bug do milênio
- Mesmo uma prova matemática de correção não comprova que não há defeitos, porque a especificação do software pode estar errada ou incompleta.



Testar não é provar que o software está correto!

2. Testes exaustivos são impossíveis

- Em teoria, testes exaustivos poderiam detectar qualquer defeito.
- Porém eles são impraticáveis porque:
 - Requerem muitos recursos (pessoas/equipamentos/ferramentas)
 - Custam muito caro
 - Tomam muito tempo



Teste o quanto você puder, mas testes exaustivos são impossíveis.

2. Testes exaustivos são impossíveis

Exemplo porque não devemos testar tudo

Considere um sistema com 20 telas:

- Em média 4 menus por tela
- Em média 3 opções por menu
- Em média 10 campos por tela
- 2 tipos de entrada por campo
- Aproximadamente 100 valores

Cálculo aproximado para realizar o teste exaustivo:

$$20 \times 4 \times 3 \times 10 \times 2 \times 100 =$$

480.000 testes



- Duração = 1s a cada teste = 17.7 dias
- Duração = 10s a cada teste = 34 semanas
- Duração = 1min a cada teste = 4 anos
- Duração = 10min a cada teste = 40 anos

Conclusão:

Se gastássemos apenas 1 minuto a cada teste, levaríamos 4 anos para realizar todos os testes.

2. Testes exaustivos são impossíveis

Necessitamos de uma alternativa pragmática, acessível, rápida e que forneça resultados.



Priorização dos testes

Ao priorizar os testes, leve em consideração primeiramente o nível de risco do projeto.

- Se acontecer uma falha no software, qual a consequência gerada?
- As considerações de risco podem incluir:
 - Implicações financeiras do software que está sendo liberado sem ser adequadamente testado (custos de suporte/possibilidade de ação legal).
 - Risco potencial de perda da vida (para sistemas de missão crítica).
 - Risco potencial de perda de reputação e de confiabilidade.
 - Software sendo liberado tarde para o mercado (*time to market*).

Em segundo lugar, leve em consideração as restrições do projeto

- Restrições do projeto, como tempo e orçamento.



Sempre que você tiver oportunidade, compartilhe os riscos do projeto com envolvidos com o projeto e tome as decisões em conjunto.

2. Testes exaustivos são impossíveis

Priorização dos testes: o princípio mais importante

- Além da avaliação de riscos nos auxiliar a definir o que testar e o que não testar, ela nos auxilia a decidir onde iniciar os testes e onde a maior carga deles é necessária.
- Desta forma, sempre tenha em sua mente a seguinte regra:

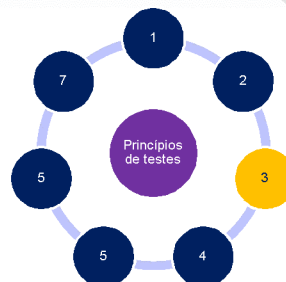
Priorize o teste de modo que, se for necessário pará-lo, você tenha certeza de que realizou o melhor teste no tempo disponível.



Não comece pelos testes mais fáceis ou de forma aleatória. Comece por onde houver um risco maior.

3. O teste inicial economiza tempo e dinheiro

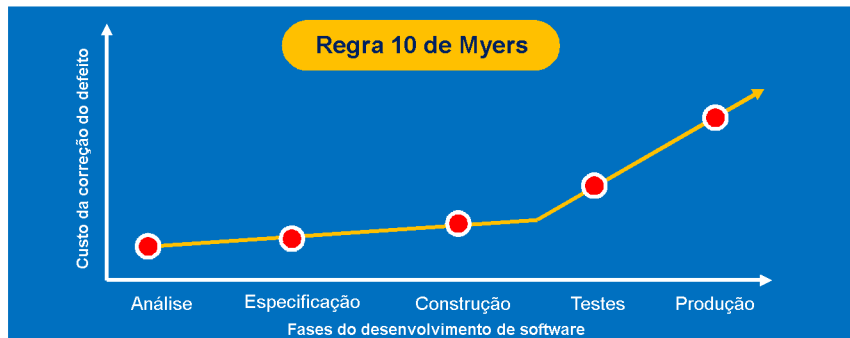
- As atividades de teste durante cada estágio do ciclo de vida devem ser focadas em objetivos definidos.
- Se o processo de revisão de documentos for considerado uma atividade de teste, o teste pode começar assim que os requisitos forem escritos.
- Isso reduzirá a probabilidade de propagação de falhas durante todo o ciclo de vida.



Para descobrir defeitos mais cedo, as atividades de teste precisam iniciar o mais cedo possível no ciclo de vida do desenvolvimento.

3. O teste inicial economiza tempo e dinheiro

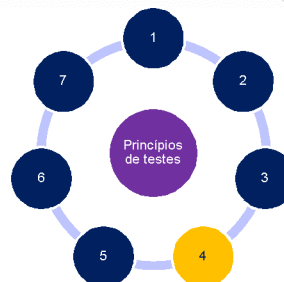
- Em 1979, Myers, escreveu o livro "A arte do teste de software".
- Nesse livro, ele introduziu a Regra 10 de Myers, um dos conceitos mais importantes já definidos:
 - Quanto mais cedo descobrimos e corrigimos um defeito, menor é o seu custo.



- Defeitos encontrados nas fases iniciais da etapa de desenvolvimento do software são mais baratos de serem corrigidos do que aqueles encontrados na etapa de produção.

4. Defeitos se agrupam

- A experiência mostra que, em geral, um pequeno número de módulos de software contém a maioria dos defeitos descobertos durante os testes de pré-lançamento ou mostra a maioria das falhas operacionais.
- Isso porque esses módulos:
 - Possuem mais complexidade.
 - Foram menos compreendidos.
 - Possuem o código mais degradado por falta de especificação apropriada.
 - A equipe de desenvolvimento era inexperiente.



O esforço de teste pode ser definido após observada a densidade de defeitos dos módulos.

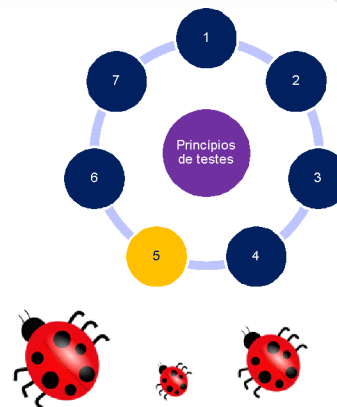
4. Defeitos se agrupam

- Esse fenômeno também está relacionado ao princípio de Pareto, o qual é chamado regra 80/20.
- Esta regra diz que aproximadamente 80% dos efeitos vêm de 20% das causas.
- Então, se você quer descobrir um número maior de defeitos, é útil empregar esse princípio nas áreas do aplicativo em que houver uma maior proporção de defeitos descobertos.
 - Mas é importante considerar que o teste não pode se concentrar exclusivamente nestas áreas.
 - Podem existir ainda defeitos remanescentes em outras áreas.



5. Cuidado com o paradoxo do pesticida

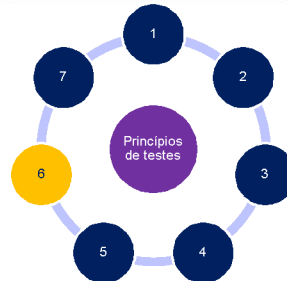
- Este princípio pode ser comparado ao princípio de se livrar dos indesejados insetos de uma horta.
 - O jardineiro pode usar um pesticida para pulverizar os insetos a fim de tentar matá-los.
 - Inicialmente pode funcionar, mas os insetos podem se tornar resistentes ao pesticida.
- Da mesma forma, se um conjunto de testes for repetido várias vezes, o mesmo conjunto de casos de teste será menos eficaz.
- Para superar isso, os casos de teste precisam ser regularmente revisados, e testes novos e diferentes precisam ser escritos para exercitar diferentes partes do software.



Se os mesmos testes forem executados várias vezes, eles não encontrarão mais novos defeitos.

6. O teste depende do contexto

- Podemos usar diferentes ciclos de vida, ferramentas, níveis de documentação e abordagens de teste dependendo do contexto.
- Exemplo: um software hospitalar crítico precisa ser testado de forma diferente de um comércio eletrônico.



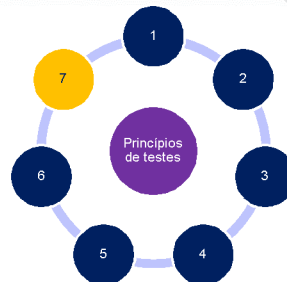
- O risco pode ser um grande fator ao determinar o tipo de teste necessário.

→ Quanto maior o risco de perda, maior o investimento em testes.

O teste precisa ser realizado de forma diferente em diferentes domínios de aplicativos.

7. Ausência de erros é uma ilusão

- Software sem erros não necessariamente significa que ele está pronto para ser lançado.
 - Essa não pode ser uma boa razão para lançar um software.
- Encontrar e corrigir defeitos não ajuda se o produto construído não for utilizável e/ou não atender às **necessidades e expectativas dos usuários**.




Independentemente de quantos defeitos foram encontrados e corrigidos, isso não significa necessariamente que os usuários estejam satisfeitos com o produto ou que ele atenderá ao seu propósito.

Módulo 1: Fundamentos de teste de software

Fim

Pronto, você finalizou esta seção do treinamento. Leia as instruções abaixo:

- Recomendamos neste momento fazer uma revisão dos slides para confirmar o entendimento de tudo o que foi apresentado nesta seção do treinamento.
- Após assistir à aula, recomendamos que você responda às questões do quiz clicando no botão "RESPONDER QUESTÕES DO QUIZ", disponível abaixo do vídeo da aula ou na opção "Realizar quiz" ao lado do link de cada vídeo aula.
- A sua nota obtida no quiz será exibida na lista dos módulos.



The screenshot shows a user interface for a quiz. At the top, a message says: 'Após assistir ao vídeo acima, recomendamos que você responda as questões do quiz clicando no botão ao lado.' Below this are two buttons: 'RESPONDER QUESTÕES DO QUIZ' and 'FECHAR JANELA E SALVAR'. Below these are two links: 'Realizar quiz' and 'Baixar Pdf'. At the bottom, there is a section for 'Última nota quiz' showing '100%' and another set of 'Realizar quiz' and 'Baixar Pdf' links.

TI.exames © Todos os direitos reservados. Proibida a redistribuição deste material. Slide 17