

Aula 4: Principais Conceitos do Processo de Teste de Software

Apresentação

Nesta aula, você irá responder a perguntas como: o que é testar? O que é teste de software? Como melhorar seus testes? Quando terminar os testes?

Você conhecerá os papéis e responsabilidades do teste de software, o processo do teste e seus princípios. Saberá ainda como planejar o teste e qual a sua importância.

Objetivos

- Definir o que deve ser testado no software e quando se deve parar de testar;
- Identificar os processos de teste de software e os princípios do teste;
- Avaliar como planejar os testes e a importância desse planejamento.



O que é testar?

O teste de software visa garantir a qualidade, minimizando as incertezas e sistematizando os critérios de aceitação. Por meio dele, pode-se avaliar se o software está fazendo o que deveria fazer, de acordo com os seus requisitos, e se não está fazendo o que não deveria fazer.

Ele ajuda a validar se:

- As expectativas de todas as pessoas envolvidas estão sendo atendidas (e se estão alinhadas);
- O software apresenta um bom funcionamento (parte disso está relacionada às expectativas implícitas – aquilo que é inerente ao produto).

Algumas definições:

"Teste é uma parte inevitável de qualquer esforço necessário para desenvolver um sistema de software. "

- Howden, 1987.

"O teste de software é um conjunto de atividades que podem ser planejadas com antecedência e executadas sistematicamente. "

- PRESSMAN, 1985.

"Qualquer atividade que, a partir da avaliação de um atributo ou capacidade de um programa ou sistema, seja possível determinar se alcança os resultados desejados. "

- Hetzel, 1988.

"Processo de executar um programa ou sistema com a intenção de encontrar defeitos. "

- Myers, 1979.

O teste deve ser utilizado para confirmar a qualidade oriunda de aplicação de um processo de Engenharia de Software.

Estratégia de teste

Por que precisamos de uma estratégia de teste de software? O que ela deve incorporar? Essas perguntas são facilmente respondidas assim:

A Engenharia de Software nos auxilia em muitas situações. Uma delas é a atividade de teste, que é um passo do processo de que visa encontrar ou corrigir erros durante toda a construção do software.

Devemos incorporar dois tipos de testes:

1

Teste de baixo nível

Utilizado para verificar um pequeno fragmento de código-fonte. Nesse caso, saberemos se ele foi implementado corretamente.

2

Teste de alto nível

Tem a característica de validar as principais funções do sistema com base nos requisitos definidos pelo cliente.

Os testes podem ser usados para descobrir a presença de erros nos softwares, mas infelizmente não mostram a sua ausência.

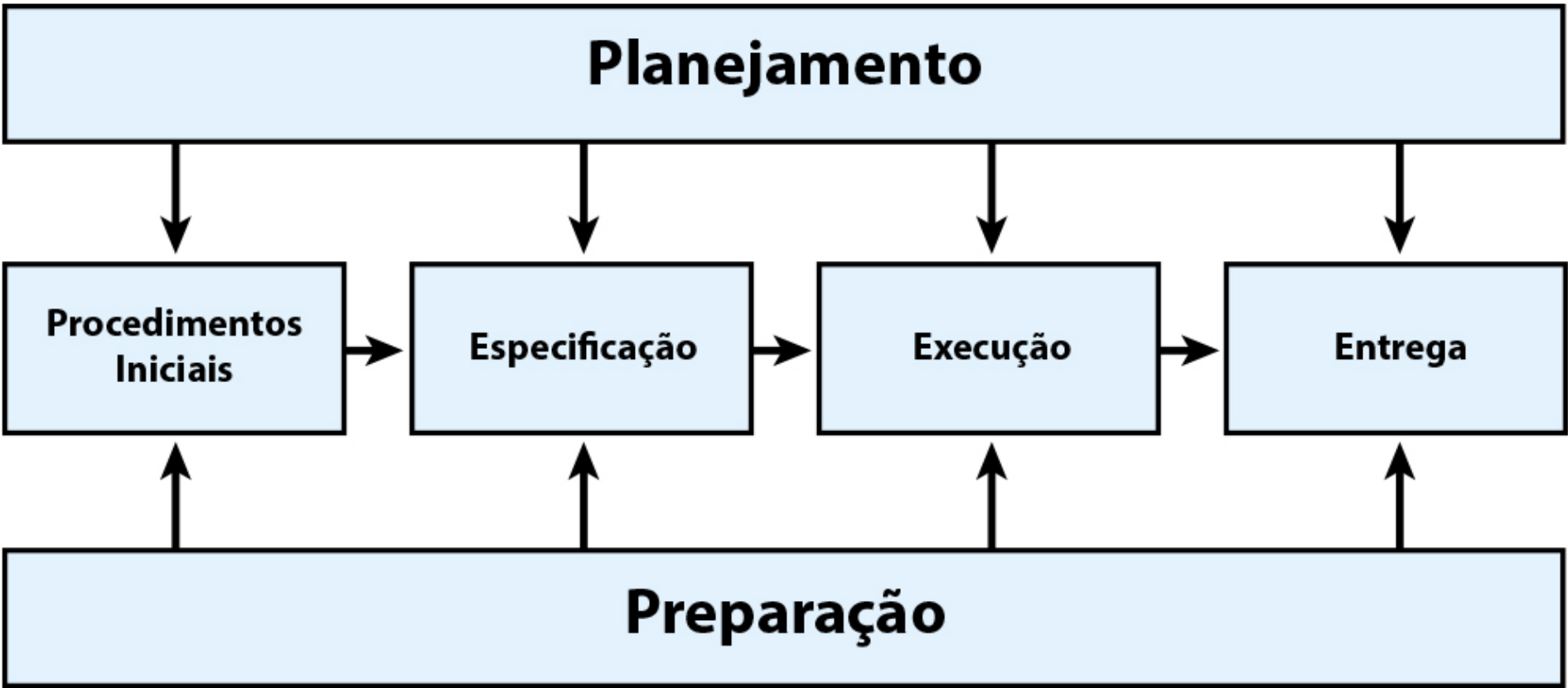
Assim, conseguimos chegar à conclusão que “o teste de software é o processo de executar o software de uma maneira controlada, com o objetivo de descobrir diferenças entre o comportamento previsto e o comportamento observado”.

Processo de teste de software

Não podemos ter um teste de software sem uma metodologia que seja favorável a esse processo de desenvolvimento, utilizando pessoal técnico qualificado, ambiente e, quando possível, todas as ferramentas adequadas.

O documento básico para organizar a atividade de testar, aplicada no contexto da empresa, tem que ser uma metodologia de teste.

Tanto o processo de desenvolvimento de software quanto o teste de software tem um ciclo de vida. Vejamos a seguir um exemplo de ciclo de vida para teste de software e a especificação de cada um deles.



 Ciclo de vida para teste de software.

Fases iniciais



Planejamento: é a parte inicial. Sem planejamento, não se deve realizar o teste. Nesta fase, é feita a elaboração e revisão da estratégia de teste e do plano de teste;

Preparação: paralelamente ao planejamento, precisamos fazer a preparação do ambiente de teste. Para tanto, precisamos incluir equipamentos, rede, pessoal, software e ferramentas.

Fases internas do ciclo



Procedimentos iniciais: é nesta fase que efetivamente se faz a elaboração do documento. Nela, estabelece-se um acordo entre as partes envolvidas no projeto de teste (as partes envolvidas são usuários e técnicos). Esse acordo deve conter:

- Objetivo do projeto de teste;
- Pessoal a ser envolvido;
- As responsabilidades de cada um;
- O plano preliminar de trabalho;
- A avaliação dos riscos;
- Os níveis de serviços acordados.

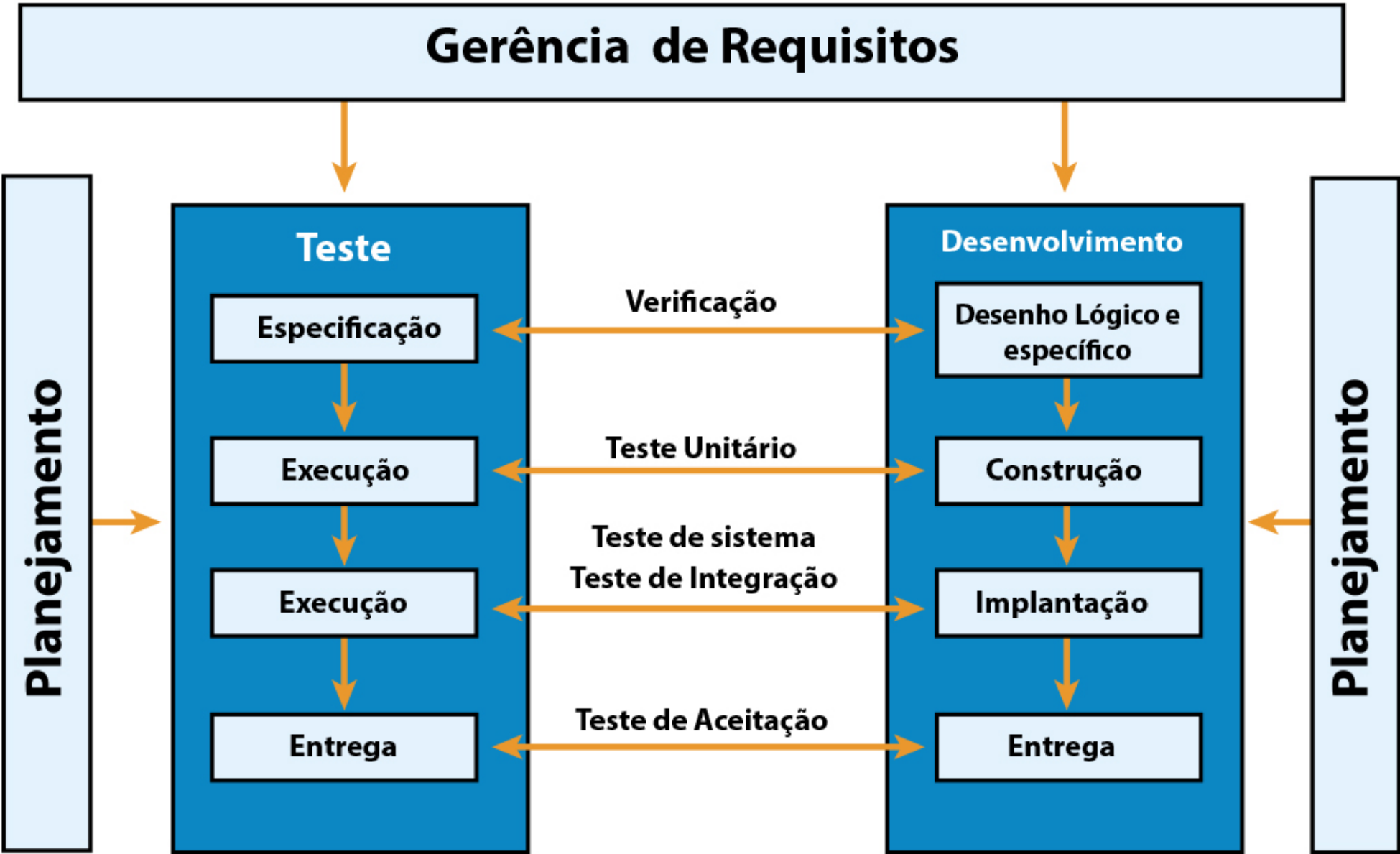
Especificação: nessa fase, realizamos a elaboração e revisão dos seguintes itens: casos de teste, scripts (para o caso de ferramentas de automação de testes), roteiros de teste e execução dos testes de verificação da documentação do sistema (testes estáticos ¹ , que veremos na próxima aula).

Execução: aqui executamos os testes planejados conforme os casos de teste; scripts; roteiros de teste com os correspondentes registros dos resultados obtidos.

Entrega: chegamos à última fase do processo. É aqui que fazemos a entrega do sistema para o ambiente de produção.

Como se dá a interação entre os ciclos de vida?

No nosso exemplo, ficaria assim:



📷 Interação entre os ciclos de vida.

_____ 1 _____

Teste das unidades individuais de programa.

_____ 2 _____

Testes destinados a facilitar a integração de unidades.

_____ 3 _____

Testes que usam o sistema concluído.

📄 Vejam algumas estratégias de teste

👉 Clique no botão acima.

Teste de verificação

Deve garantir a qualidade de todas as etapas do desenvolvimento de sistemas. Nesta etapa, são realizadas inspeções (auditorias com foco nas atividades) e revisões (com foco nas documentações) sobre os produtos gerados.

Testes unitários

Cada programa ou componente é testado isoladamente para testar sua resposta aos requisitos definidos. Esses testes são realizados no estágio mais baixo da escala de testes e são aplicados nas menores componentes de códigos criados ².

Assim, é possível garantir que estes atendam às especificações, tanto de garantia quanto de funcionalidade.

Por terem que verificar o funcionamento de um pedaço do sistema ou software isoladamente, normalmente são feitos pelos desenvolvedores.

Testes de integração

Os programas e componentes são integrados pouco a pouco para testar suas interfaces. São executados em uma combinação de componentes para verificar se eles funcionam corretamente juntos, conforme foram especificados, e também normalmente podem ser feitos pelos desenvolvedores.

Testes de sistema

Nesse momento, todos os programas e componentes estão totalmente integrados, formando um sistema. Esses testes visam à execução do sistema como um todo ou um subsistema (parte de um sistema), dentro de um ambiente operacional controlado, o mais próximo possível do ambiente de produção, para validar a exatidão e perfeição na execução de suas funções.

Nesta fase, o teste deve simular a operação normal do sistema, pois serão testadas todas as suas funções da forma mais próxima possível do que irá ocorrer no ambiente de produção. Esses testes são de responsabilidade e feitos pela equipe de teste de software.

Teste de aceitação

O ambiente é o mesmo ou muito semelhante ao utilizado nos testes de sistema. São os testes finais de execução do sistema, realizados pelos usuários.

Nesses testes é verificado se a solução atende aos objetivos do negócio e aos seus requisitos definidos inicialmente, no que diz respeito à funcionalidade e dentro da característica de usabilidade, preocupando-se também com a interação humano/ computador, antes da sua utilização no ambiente de produção.

Atenção! Aqui existe uma videoaula, acesso pelo conteúdo online

Quando a organização trata os testes como um processo bem estruturado e muitas vezes paralelo e integrado ao processo de desenvolvimento, os custos de manutenção com certeza são reduzidos.

Atenção

É importante lembrar que, conforme vimos na aula anterior, segundo Myers, o custo de correção de defeitos tende a aumentar quanto mais tarde o defeito é detectado.

Defeitos encontrados durante a produção tendem a custar muito mais que defeitos encontrados em modelos de dados e em outros documentos do projeto do software.

Quais os benefícios que cada um desses testes pode trazer?

Testes unitários	Podem remover entre 30% e 50% dos defeitos dos programas.
Testes de sistemas	Podem remover entre 30% e 50% dos defeitos remanescentes, mas mesmo assim, os sistemas podem ir para a produção ainda com aproximadamente 49% de defeitos.
Revisões de códigos	Podem reduzir entre 20% e 30% desses defeitos.




Princípios de teste de software

Pensando no teste como parte fundamental do **ciclo de vida de um software**, vamos mostrar os **sete princípios fundamentais** que envolvem o **processo de teste** que devem servir como um **guia geral**, tanto para testadores quanto para desenvolvedores.

Veremos mais adiante que ambos participam efetivamente do processo de amadurecimento do sistema.

 Princípios de teste de software

 Clique no botão acima.

1º Princípio: **teste demonstra a presença de defeitos**

Os testes conseguem identificar a **existência de falhas**, mas não podem garantir a ausência delas.

Mesmo se nenhum erro for identificado em uma **bateria de testes**, não é possível afirmar que o software está livre de falhas.

2º Princípio: **teste exaustivo é impossível**

Deve-se calcular o esforço dos testes baseando-se nos riscos e prioridades.

3º Princípio: **teste antecipado**

Ao desenvolver um software, as **atividades de teste** devem começar o mais cedo possível no ciclo de vida do desenvolvimento do software. Assim diminuimos o custo das correções e possibilitamos que erros de design, requisitos e arquitetura sejam encontrados no momento ideal.

Logo que os requisitos ou modelagem do sistema estiverem prontos, é possível começar o trabalho de modelagem do **plano de testes**.

Quanto antes uma falha for identificada no ciclo de vida de um sistema, mais barata e mais simples será a correção.

4º Princípio: **agrupamento de defeitos**

A maioria das falhas encontradas durante a **execução dos testes** está concentrada em um número pequeno de módulos. Sempre existe uma área do software que é responsável pelo maior número de erros.

5º Princípio: **paradoxo do pesticida**

Um conjunto de testes, se executado várias vezes, pode não mais detectar novas falhas. Para contornar esse problema, os **casos de teste** devem ser frequentemente revisados e atualizados.

Eles devem ser reformulados para abordar novas áreas do sistema e assim aumentar a chance de detectar novas falhas.

Os testes precisam ser revisitados com frequência.

6º Princípio: **teste é dependente do contexto**

Diferentes tipos de aplicações exigem a aplicação de técnicas diferentes de teste.

Por exemplo: um sistema bancário deve ser testado de maneira diferente de uma rede social. Há questões de segurança que devem ser mais precisamente abordadas no primeiro caso.

Da mesma forma, **testes web** são elaborados com foco diferente dos **testes de aplicações desktop**.

7º Princípio: **a ilusão da ausência de defeitos**

Identificar e corrigir os problemas de um software não garante que ele está pronto.

Questões a serem respondidas:

- Os testes foram elaborados para identificar todas as possíveis falhas?
- O sistema atende às necessidades e expectativas dos usuários?

Ou seja, existem outros fatores que devem ser considerados para garantir a qualidade do sistema.

Não adianta o sistema ser correto funcionalmente, mas não atender à real necessidade do usuário.

Todos os princípios são importantes, porém entre os princípios listados, podemos citar que os números 3 e 7 representam os principais aspectos da atividade de teste.

A busca por antecipar cada vez mais as possíveis falhas da aplicação e assegurar que o sistema entregue atenda as reais necessidades do cliente, agregando valor ao seu negócio, é constante.

Características das estratégias de teste



- Para executar um teste eficaz, devem-se proceder revisões técnicas eficazes. Isto é necessário porque assim muitos erros são eliminados antes do começo do teste;
- Para executar um teste eficaz, devem-se proceder revisões técnicas eficazes. Isto é necessário porque assim muitos erros são eliminados antes do começo do teste;
- É importante que se usem diferentes técnicas de teste para diferentes abordagens e em diferentes momentos;
- O teste deve ser feito pelo desenvolvedor e por um grupo independente de teste (se for um grande projeto);
- Saiba que tanto o teste quanto a depuração são atividades diferentes, mas a depuração só acontece em consequência da existência de um teste.

Com a atividade de teste, conseguimos executar um programa com a intenção principal de descobrir um erro.

Para que um caso de teste seja considerado bom ou ótimo, ou seja, bem-sucedido, ele deve ter uma grande probabilidade de revelar um erro ainda não descoberto.

Diretrizes que devem ser levadas em conta para o teste



- Quando o teste deve ser interrompido? - Deve ser determinado no planejamento;
- Atribuir a responsabilidade do teste a um testador? - Deve ser atribuído também no planejamento;
- Quais os resultados esperados para cada caso de teste? - Devem ser descritos antecipadamente;
- Quais os resultados esperados para cada caso de teste? - Devem ser descritos antecipadamente;
- Qual o resultado de cada teste por completo? - Cada um deles deve ser inspecionado depois de testado;
- Quais programadores devem ser alocados para o teste? - Sempre os mais criativos.

A importância dos testes

O processo de desenvolvimento de sistemas (PDS) requer uma série de atividades em que as oportunidades de inserção de falhas são muito grandes.

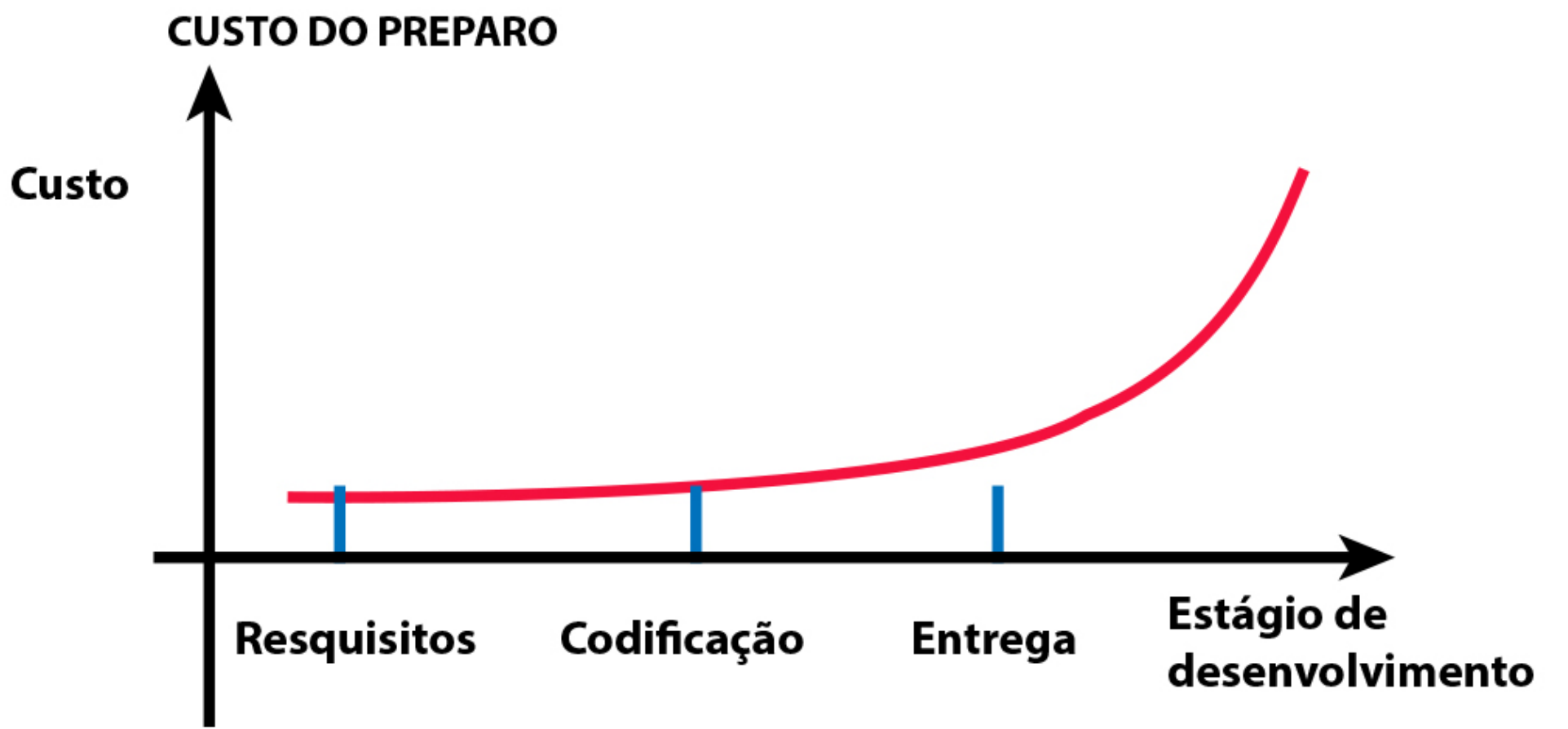
Se os objetivos foram especificados erradamente, os erros podem começar a aparecer logo no início do processo, podendo também aparecer erros em fases de projeto e desenvolvimento posteriores.

Como as informações para o projeto são determinadas pelo usuário/cliente, e o desenvolvimento é feito por técnicos especializados (programadores, analistas, gerentes etc.), muitas vezes pode haver falhas na comunicação.



Nesse momento, é importante que o desenvolvimento seja acompanhado de **garantia de qualidade**, em função de não haver essa habilidade de realizar e de se comunicar com perfeição.

Vemos então que a atividade de teste de software passa a ser um **elemento crítico** da garantia de qualidade. Esta passa a ser a última revisão de especificação, projeto e codificação.



Relação entre o custo do software e as fases do desenvolvimento.

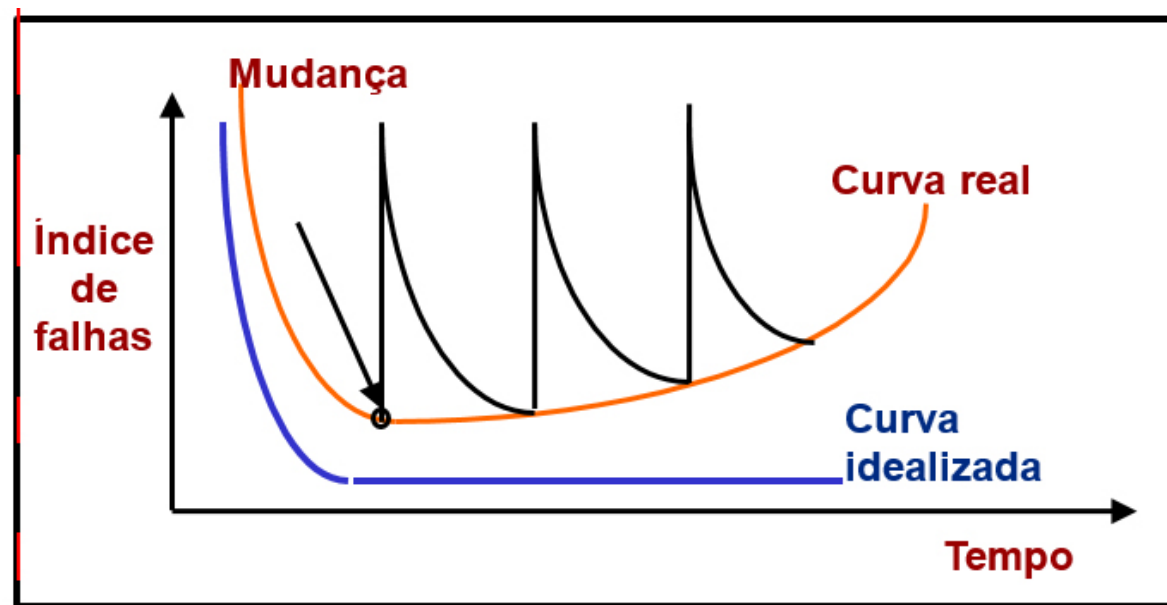
Exemplo - Projetos de controle de voo

Clique no botão acima.

Como vimos, é possível que os gastos fiquem entre 30% e 40% do esforço total do projeto no teste de software.

Considere, por exemplo, os projetos de controle de voo, monitoramento de reatores nucleares etc.

Esses projetos são considerados críticos para testes, pois sua falha pode resultar em significativos prejuízos econômicos, humanos ou ambientais, e por isso podem custar de três a cinco vezes mais do que todos os outros passos de engenharia de software combinados.



 Relação ente o tempo e o índice de falhas.

Devem-se utilizar as principais estratégias de teste de software, de forma a **promover uma abordagem de teste personalizada** e de diferentes níveis, visando atingir todas as fases do ciclo de desenvolvimento do software.

O uso de uma equipe independente no processo de teste de software cria um ambiente de teste e torna a aplicação de teste unitário pelo desenvolvedor.

A aplicação de estratégias que integrem técnicas de projeto de casos de teste, numa série bem definida de passos, produz um mapa que descreve os passos a serem dados como parte da atividade de teste.

Essa estratégia deve incorporar o planejamento de teste, o projeto de casos de testes, a execução e a resultante coleta e avaliação.

Deve-se também responder às seguintes perguntas:

Como conduzir os testes de software?

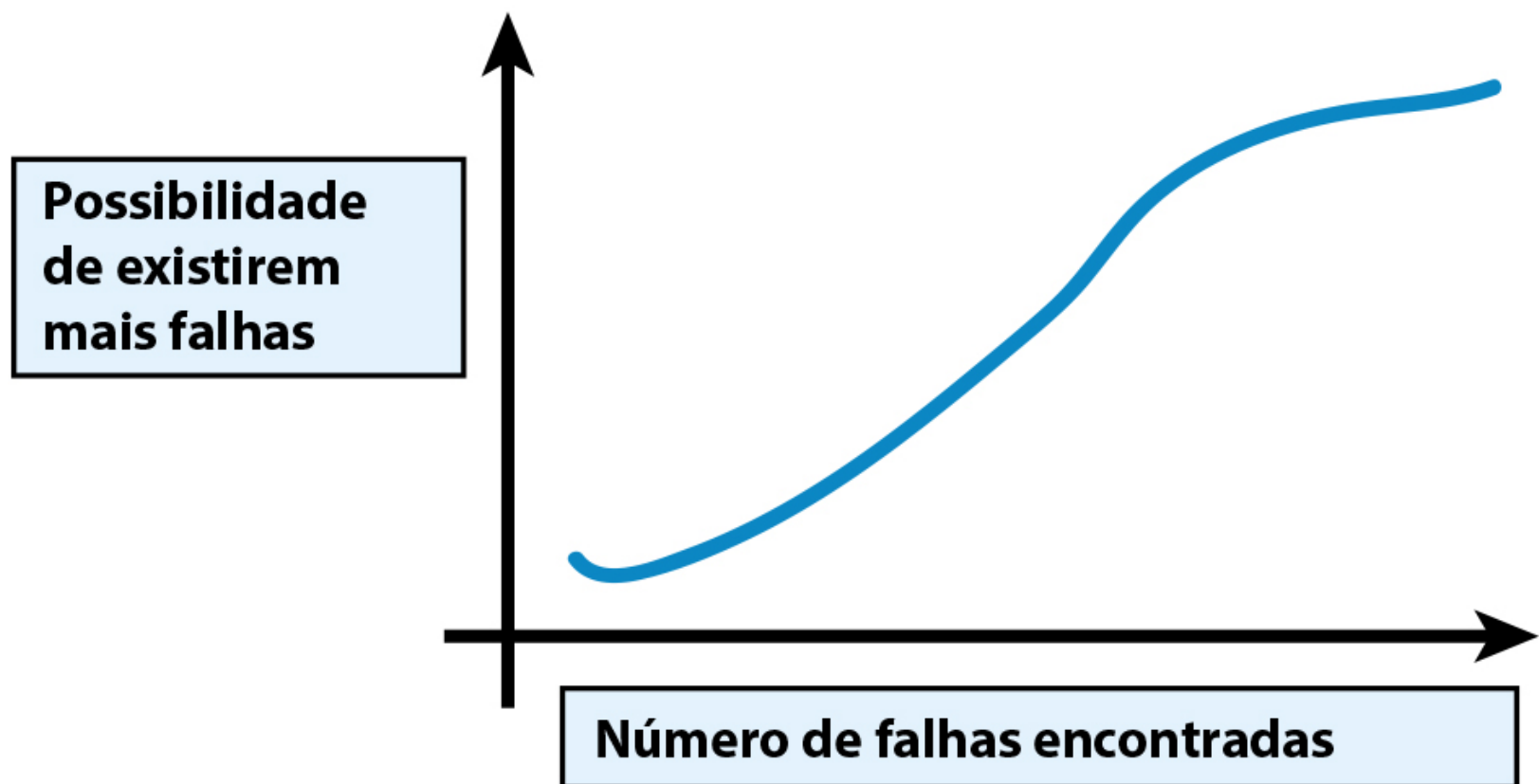
Devemos estabelecer um plano formal para os testes?

Devemos testar o programa como um todo ou executar testes somente em uma parte dele?

Devemos refazer os testes quando acrescentamos novos componentes ao sistema?

Quando devemos envolver o cliente?

Em geral, os testes devem ser terminados quando a maioria das necessidades é atendida, permanecendo poucos erros importantes. Pode-se alguma vez ter certeza de que não existem mais falhas?



📷 Relação entre o número de falhas encontradas e a possibilidade de existirem mais falhas.

Quando o teste é terminado, como saber se ele foi suficiente?

Não há uma resposta única.

Algumas respostas pragmáticas:

- Você jamais terá completado a atividade de teste;
- A carga simplesmente transfere-se do projetista para o cliente;
- O teste para quando não há mais erros “visíveis”;

O teste acaba quando o tempo acaba ou o dinheiro acaba:

- Por restrição de tempo (nesse caso, deve-se negociar esse tempo);
- Por restrição financeira (nesse caso, deve-se evitar).

A atividade de teste jamais termina. Ela passa do projetista para o usuário.

Então, quando terminar o teste?


Basta pensar que:

- O objetivo do teste é encontrar erros, e se eles não forem detectados, o teste não pode afirmar sua ausência;
- Testar tudo é impossível;
- As técnicas de teste são complementares, devendo ser aplicadas em conjunto.

Para testar com eficiência, é preciso conhecer bem o software.

Papéis e responsabilidades de teste de software

Vamos ver dois grupos:

 Clique nos botões para ver as informações.

Desenvolvedor



É sempre responsável por testar unidades individuais (componentes). Em muitos casos, também conduz testes de integração, um passo que leva à construção (e teste) da arquitetura completa do software.

Grupo independente de teste (*independent group test* – ITG)



Normalmente, para que o processo de teste transcorra de forma íntegra, é comum a utilização de um **grupo independente de teste**, já que as pessoas que criaram o software não devem ser as mesmas que irão realizar os testes.

Seria um conflito de interesses, pois foram elas que o “**criaram**”. Eles se envolvem no projeto após a arquitetura do software estar completada.


O engenheiro de software e o ITG trabalham juntos para garantir a execução de testes rigorosos. Existem testes que somente serão conduzidos pelos desenvolvedores, como o teste de unidade, que iremos estudar mais adiante.

O desenvolvedor deve estar disponível para corrigir eventuais erros descobertos.

"O primeiro erro que o pessoal comete é pensar que a equipe de teste é responsável pela garantia de qualidade. "

- Bryan Marich.

 Existem várias responsabilidades e papéis dentro da equipe

 Clique no botão acima.

Gerente de teste	Gerente de vários projetos de teste ou responsável pela área de teste da empresa.
Líder do projeto de testes	Responsável pela liderança de um projeto de teste, geralmente relacionado a um sistema em desenvolvimento, seja um projeto novo ou manutenção.
Arquiteto	Responsável pela montagem da infraestrutura de teste, monta o ambiente de teste, escolhe as ferramentas e capacita a equipe para executar seu trabalho nesse ambiente.
Analista do teste	Responsável pela modelagem e elaboração dos casos de teste e pelos scripts. Em alguns casos, estes podem ser elaborados pelos testadores.
Testador	Responsável pela execução dos casos de teste e scripts.

Atividade

1. Assinale a única alternativa correta.

O que é necessário para obter resultados positivos em um projeto de testes?

I. Que o projeto se inicie desde a especificação dos requisitos do sistema a ser implementado.

II. Que o projeto se inicie quando a programação estiver sendo desenvolvida.

III. Que o projeto se inicie quando a programação estiver sendo desenvolvida.

- a) Apenas o item I está correto.
- b) Apenas o item II está correto.
- c) Apenas o item III está correto.
- d) Apenas os itens I e II estão corretos.
- e) Apenas os itens II e III estão corretos.

2. Assinale a única alternativa correta.

O processo de testes de software representa uma estrutura das etapas, atividades, artefatos, papéis e responsabilidades. Sendo assim, o que busca esse processo?

- I. Padronizar os trabalhos para um melhor controle dos projetos de testes.
- II. Minimizar os riscos causados por defeitos provenientes do processo de desenvolvimento, assim como a redução de custos de correção de defeitos.
- III. Redução de custos de correção de defeitos.

- a) Apenas os itens I e II estão corretos.
 - b) Apenas os itens II e III estão corretos.
 - c) Apenas o item II está correto.
 - d) Apenas o item III está correto.
 - e) Todos os itens estão corretos.
-

3. Como a equipe de teste deve ser formada?

- a) Apenas com os clientes e seus usuários.
 - b) Apenas com uma equipe de testes independentes.
 - c) Apenas com os desenvolvedores dos programas.
 - d) Apenas com uma equipe de teste independente e os desenvolvedores.
 - e) Apenas com os usuários e os desenvolvedores.
-

4. Quando devemos terminar os testes?

- I. Nunca; o projetista está sempre testando.
- II. Quando o dinheiro ou o tempo acabar.
- III. O teste termina quando não há mais erros.

Assinale a única alternativa correta:

- a) Apenas os itens I e II estão corretos.
 - b) Apenas os itens II e III estão corretos.
 - c) Apenas o item II está correto.
 - d) Apenas o item III está correto.
 - e) Todos os itens estão corretos.
-

Notas

Testes estáticos ¹

São testes realizados pela *análise* do código fonte. O tipo de análise é visual, podendo haver um questionário para acompanhar os testes, inspecionando o código desenvolvido pela equipe de programação.

Componentes de códigos criados ²

Componentes são pedaços de código, módulos, aplicações distintas ou ainda clientes servidores.

Referências

AGUIAR, E. P. Didática da história: uma ciência da aprendizagem histórica? XXIII Simpósio Nacional de História. Florianópolis, 2015. Disponível em: [//www.snh2015.anpuh.org/resources/anais/39/1428351472_ARQUIVO_EdinalvaPadreAguiar-TextoANPUH2015.pdf](http://www.snh2015.anpuh.org/resources/anais/39/1428351472_ARQUIVO_EdinalvaPadreAguiar-TextoANPUH2015.pdf). Acesso em: 10 mai. 2019.

BARTIÉ, Alexandre. Garantia de qualidade de software. 1. ed. Campus, 2002.

HOWDEN, W. E. *Theoretical and Empirical Studies of Program Testing*, 1987.

HETZEL, W. C. *The Complete Guide to Software Testing*, 1988.

MYERS, Glenford J. *The Art of Software Testing*. New York: Wiley, 1979.

PRESSMAN, R. S. Engenharia de Software. 6. ed. Makron Books, 1995.

SOMMERVILLE, Ian. Engenharia de Software. 9. ed. São Paulo: Pearson, 2011.

Próxima aula

- Paralelismo entre as atividades de desenvolvimento e teste de software;
- Verificação e validação;
- Técnicas de teste.

Explore mais

Leia os textos:

- [Análise estática de código](#);
- [Formando equipes eficientes de teste de software](#);
- [Qualidade de software](#);
- [Engenharia de software - Introdução à inspeção de software](#);
- [ALATS - Associação Latino-americana de Teste de Software. Missão e propósito](#);
- [BSTQB](#).