

Aula 5: Ciclo de Vida do Processo de Testes de Software

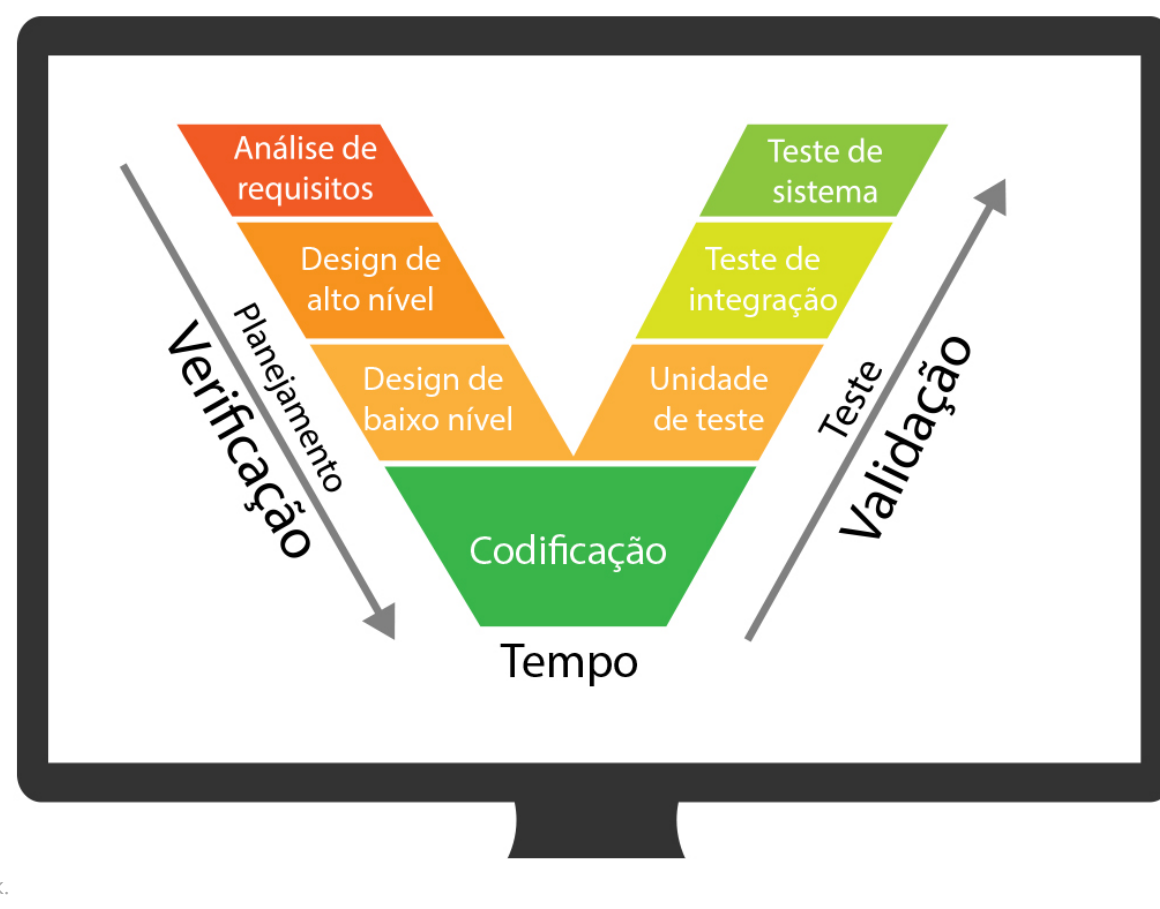
Apresentação

Nesta aula, vamos conhecer o modelo V de desenvolvimento de software para validação e verificação, identificando o paralelismo entre as atividades de desenvolvimento e teste de software.

Vamos identificar e reconhecer testes de verificação e de validação, diferenciar quando usar testes estáticos e dinâmicos, aprender as técnicas de teste e diferenciar os testes estruturais dos funcionais.

Objetivos

- Descrever o modelo V (verificação e validação), com paralelismo entre as atividades de desenvolvimento e teste de software;
- Identificar os testes estáticos dos dinâmicos e suas técnicas;
- Distinguir os testes estruturais dos funcionais.



Modelo V

Você se lembra do modelo cascata no processo de desenvolvimento de software? O **modelo V** é uma melhoria do cascata do desenvolvimento de produto, pois esse modelo tinha um problema de reatividade.

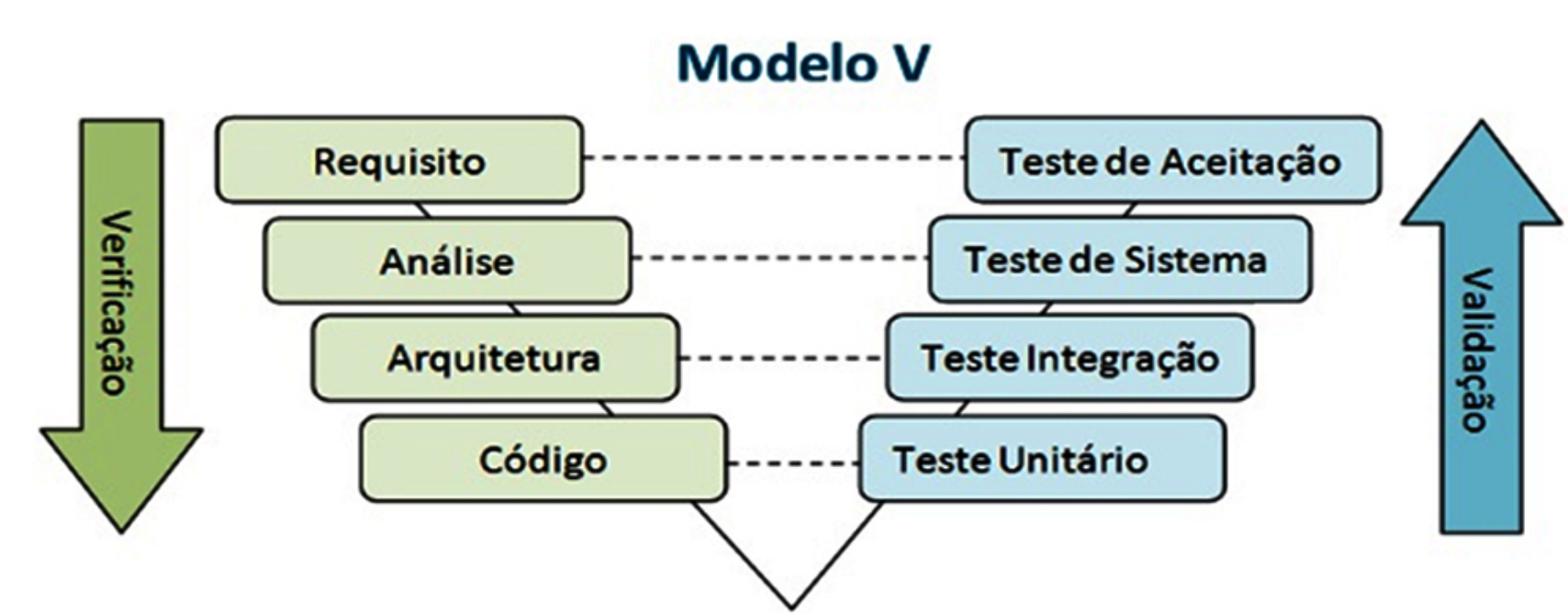
Ele permite que, durante a integração de um sistema, os testes sejam feitos contra os próprios requisitos do componente ou interface que está sendo testado, em contraste com modelos anteriores, nos quais o componente era testado contra a especificação do componente/ interface.

Nesse modelo, cada etapa deve ser concluída antes que a próxima inicie, e o teste é planejado em paralelo com a atividade correspondente no desenvolvimento, ou seja, o modelo considera o teste como uma atividade paralela, e não como uma atividade isolada que ocorre no final do desenvolvimento.

📄 Como o modelo V se configura?

👉 Clique no botão acima.

Vejamos uma representação simples na figura a seguir.



📷 Representação do modelo V.

Em muitos casos, as organizações criam seus próprios modelos usando isso como base. Contudo, o modelo pode se tornar tão complexo quanto você quiser.

É importante ressaltar que devemos utilizar esse modelo para pequenos e médios projetos, com requisitos bem definidos e profissionais experientes.

Quais são os objetivos do modelo V?

- Minimizar os riscos do projeto;
- Melhorar e garantir a qualidade do projeto;
- Reduzir os custos totais ao longo do ciclo de vida do projeto;
- Melhorar a comunicação entre as partes interessadas.

É um modelo mais robusto e completo do que o cascata, podendo produzir softwares de maior qualidade do que com ele.

Esse modelo acrescenta duas partes importantes, que são:

Atenção! Aqui existe uma videoaula, acesso pelo conteúdo online

———— 1 ————

Verificação

Que está relacionado com a questão:
O produto está sendo feito corretamente?


———— 2 ————

Validação

Está relacionado com a questão:
O produto está sendo feito, ou seja, o software atende ao objetivo pretendido com precisão?

No **modelo V**, podemos ver as mesmas fases do cascata, mas com uma melhor relação entre elas.

 Vejamos as vantagens e desvantagens desse modelo

 Clique no botão acima.

Vantagens:

- A relação entre os estágios de desenvolvimento e os diferentes tipos de testes facilita a localização de falhas;
- É um modelo simples e fácil de aprender;
- Especifica os papéis dos diferentes tipos de testes para ser executada;
- Envolve o usuário no teste.

Desvantagens:

- É difícil para o cliente expor explicitamente todos os requisitos;
- O cliente deve ter paciência, pois receberá o produto no fim do ciclo de vida;
- O teste pode ser caro e às vezes não ser suficientemente eficaz;
- O produto final pode não refletir todas as necessidades dos utilizadores.

Paralelismo entre as atividades de desenvolvimento e teste de software

Além do paralelismo entre as atividades, o teste paralelo também serve para comparar os resultados do teste do sistema atual com a versão anterior. Essa comparação é importante para determinar se os resultados do novo sistema são consistentes com o processamento do antigo sistema ou da antiga versão.

Temos que executar no teste paralelo os mesmos dados de entrada utilizados nas duas versões da mesma aplicação.



Fonte: Por Gorodenkoff / Shutterstock.

Exemplo

Se os requisitos não forem alterados na nova versão, os resultados com os dados de saída das duas versões (atual e nova) devem ser iguais.

Processo de teste

Com relação aos custos, ao tratar os testes como um processo organizado e muitas vezes **paralelo** e integrado ao processo de desenvolvimento, os custos de manutenção ficarão bem reduzidos.

Segundo Myers, o custo de correção tende a aumentar quanto mais tarde o defeito for detectado.

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

Atenção

Verificação da migração de dados

Com relação aos dados, após o carregamento para o novo sistema, os resultados são submetidos a uma etapa de verificação dos dados para determinar se estes foram corretamente migrados, se ocorreu de forma completa.

Durante essa verificação, pode ser necessário um processo de execução em paralelo de ambos os sistemas para identificar áreas de disparidade e evitar erros de perda de dados.

Veja alguns processos de teste:

Teste de migração de dados



O paralelismo pode ser usado também pelas **equipes de teste de aceitação e operacional**, que irão realizar o teste de concepção e execução em paralelo em duas plataformas completamente diferentes e com objetivos e propósitos diferentes.

Já a **equipe de teste de legado** irá trabalhar com cada um dos outros grupos separadamente.

Equipe de teste operacional

É composta de usuários que irão utilizar o sistema. O grupo operacional realiza **testes em paralelo** com a equipe de aceitação, mas com um objetivo completamente diferente.

Esse grupo trabalha na futura plataforma e no ambiente sob todos os aspectos que não sejam sob o ponto de vista do negócio, ou seja, de conectividade e interoperabilidade.

Esse grupo também testa interfaces, desempenho, sistema de back-ups e demais elementos da nova plataforma.

Simulação do novo sistema em substituição ao antigo

Nesta etapa do processo, deverá ser realizado um simulado do novo sistema em condições reais de funcionamento em paralelo ao antigo sistema, para que o usuário gestor possa avaliar possíveis necessidades de ajustes do fluxo operacional e computacional de forma a trazer benefícios com a instalação do novo sistema.

Scripts estruturados ou scripts compartilhados

Esta técnica aciona mais de um comando, simulando a execução em paralelo de diversas ações.

Teste de unidade



O teste de unidade enfoca o esforço de verificação da menor unidade do projeto de software, que é o módulo.

Usa-se como guia para a execução deste tipo de teste uma descrição detalhada do projeto. Importantes caminhos de controle são testados para descobrir erros nas interfaces dos módulos.

O teste de unidade é do tipo **caixa branca** (veremos esse tipo de teste mais tarde). Pode ser conduzido em paralelo para vários módulos.

Teste de validação



Os testes são baseados no comportamento do software por meio das mais diversas condições baseadas e comparadas com as especificações levantadas pela área de negócio.

Durante o desenvolvimento do software, deverão ser aplicadas diferentes categorias de testes empregando ferramentas, técnicas e abordagens diferentes.

As atividades de teste (conforme o modelo V, planejamento, modelagem, execução e conferência) deverão ocorrer em paralelo às atividades de construção de componentes executáveis e respeitando os estágios de desenvolvimento.

Verificação e validação

A atividade de teste de software é elemento de um tema mais amplo chamado **verificação e validação**.

1

Verificação

Refere-se ao conjunto de atividades que garante que o software implemente corretamente uma função específica.

2

Validação

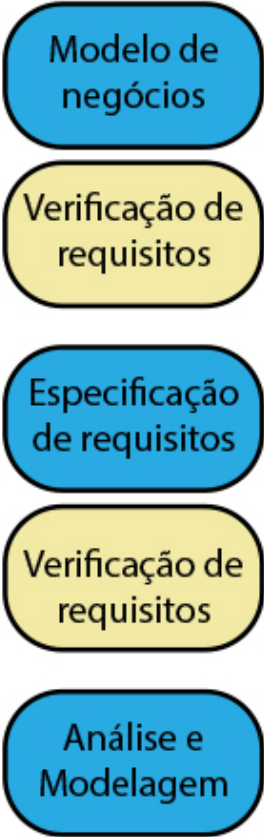
Refere-se ao conjunto de atividades que garante que o software que foi construído segue as exigências do cliente.

A definição de verificação e validação abrange muitas das atividades às quais está diretamente ligada a garantia da qualidade de software (SQA).

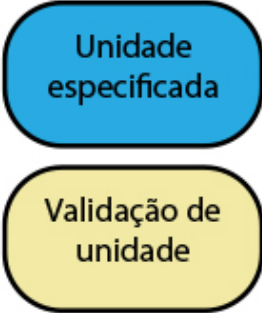
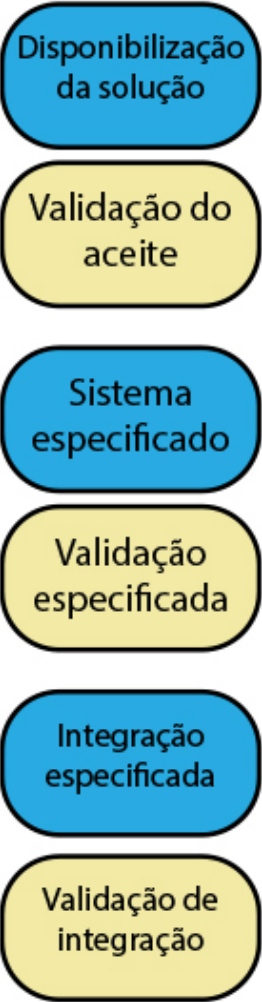
Vamos ver de outra forma o modelo V como um U.

Formato de “U”

Testes de verificação



Testes de verificação



Verificação de
análise e
modelagem

Como são esses momentos do processo de desenvolvimento de software?


Teste



 Representação do modelo U.

Os testes de verificação e validação são complementares, não devendo ser encarados como atividades redundantes. Cada um possui natureza e objetivo distinto, fortalecendo desta forma o processo de detecção de erros e aumentando a qualidade final do produto.

 Testes de verificação e validação

 Clique no botão acima.

Teste de verificação

- É a coleta de informações de negócios e o planejamento da arquitetura do software.
- A principal preocupação é o entendimento e a coerência entre o negócio a ser atendido e o software a ser construído.
- Nesta fase não existem componentes tecnológicos, mas documentos que especificam o comportamento a ser seguido pelo software a ser desenvolvido.
- É um processo de auditoria de atividades e avaliação de documentos gerados em todas as fases do processo de desenvolvimento do software (PDS).

Esses testes não envolvem o processamento de softwares, pois eles ainda não nasceram.

Vejamos cada um dos processos de verificação.

Verificação dos negócios

Seu objetivo é garantir que os diversos documentos produzidos tenham total aderência às necessidades apontadas pelos clientes.

Verificação dos requisitos

Seu objetivo é fazer a verificação das especificações do levantamento dos requisitos funcionais ¹ e não funcionais ² do software a ser desenvolvido.

Verificação da implementação

Seu objetivo é garantir a qualidade do código-fonte gerado pela equipe de desenvolvimento.

Como se garante essa qualidade?

- Pela prática das regras da boa programação;
- Pelo processo formal de verificação do código produzido.

Teste de validação

É um processo formal de avaliação de produtos tecnológicos que podem ser aplicados em componentes isolados, módulos existentes ou mesmo a totalidade do sistema.

Qual é o seu objetivo?

Avaliar a conformidade do software com os requisitos e especificações analisadas e revisadas nas etapas iniciais do projeto.

Como ele se caracteriza?

Pela presença física do software e de seu processamento em um ambiente tecnicamente preparado.

Vejamos cada um dos **processos de validação**.

Validação da unidade

A validação de unidade é a primeira etapa do processo de validação. Seu objetivo é testar os componentes individuais de uma aplicação.

Validação da integração

A validação de integração é uma continuação natural dos testes unitários. Seu objetivo é validar a compatibilidade entre componentes de um software.

Validação do sistema

Seu objetivo é validar a solução como um todo. Quando este estágio é atingido, a maior parte das falhas de funcionalidade deve ter sido detectada pelos testes unitários e pelos de integrações.

Validação do aceite

É o último estágio do processo de validação, sendo o último processo formal de detecção de erros no sistema, antes de sua disponibilização no ambiente de produção.

Nessa etapa, o software é disponibilizado para clientes e usuários com o objetivo que eles mesmos validarem todas as funcionalidades requisitadas no início do projeto.

Exemplo de testes de validação

Validação: testes que demonstram conformidade com os requisitos.

Plano de teste: descreve classes de testes + procedimento de teste + casos de teste.

Após a execução de cada caso de teste de validação ser conduzido:

- 1) Ou a característica testada está de acordo com a especificação – é aceita;
- 2) Ou descobre-se um desvio da especificação – lista de deficiências.

Objetivo: verificar o produto nas referências e padrões de usabilidade e de desempenho estabelecidos no planejamento da qualidade.

O teste de validação verifica também a interação entre os componentes do produto, como por exemplo a relação entre formulários e os bancos que recebem os dados coletados, o suporte e a ajuda oferecidos aos usuários.

Atenção! Aqui existe uma videoaula, acesso pelo conteúdo online

Testes estáticos X testes dinâmicos

Falhas de software são uma constante para quem trabalha com desenvolvimento.

Falhas menores podem representar apenas pequenos problemas na execução de um sistema. Já em casos mais graves, um bug ou vulnerabilidade pode levar à exposição de dados de usuários e informações privadas de empresas.

Algumas divisões podem ser estabelecidas em relação ao teste de software. Uma delas corresponde à forma de utilização do código obtido na etapa de implementação (ou codificação). Segundo esta ótica, podem-se organizar os testes em **estáticos e dinâmicos**.

Os métodos de verificação utilizam técnicas de testes estáticos para avaliar a qualidade dos documentos gerados durante todas as etapas do desenvolvimento do software.

Para avaliarmos a qualidade de um sistema, os testes não podem ser estáticos; precisam ser dinâmicos, pois devemos submeter o software a determinadas condições de uso de forma a avaliar se o comportamento está de acordo com o esperado.

Testes estáticos

São do tipo caixa branca. O principal objetivo dessa técnica é identificar erros de programação, tais como:

- Práticas ruins;
- Erros de sintaxe;
- Falhas de segurança.

Os testes estáticos são aqueles realizados sobre o código-fonte do software, utilizando como técnica básica a inspeção visual. Este tipo de teste é de simples implementação, já que não há a necessidade de execução do programa para se obter resultados.

Uma técnica bastante utilizada é a da leitura cruzada, na qual um leitor é atribuído para avaliar o trabalho de cada programador do software.

Outra técnica de realizar o teste é por **inspeção**, na qual uma equipe designada analisa o código segundo o entendimento de um questionário especialmente concebido, através de um checklist.

Nesses dois casos, os responsáveis pela realização do teste não fazem nenhum tipo de correção no código, apenas limitam-se a assinalar os erros encontrados.

Em alguns poucos casos, o teste estático pode ser automatizado com o auxílio de ferramentas de análise estática, que podem ser simples geradores de referências cruzadas ou, no caso de ferramentas mais sofisticadas, serem dotadas de funções de análise do fluxo de dados do programa.

Testes dinâmicos

São testes do tipo **caixa preta**. Consistem nos procedimentos baseados na “execução do código binário” do programa, sendo esta execução realizada com base em subconjuntos de dados, o que é caracterizado como o jogo de teste.

A escolha do subconjunto de dados a ser utilizado para o teste será feita com base em aspectos estruturais do software (obtido a partir do código-fonte) ou em aspectos funcionais (a partir da especificação do programa).

Exemplo

As abordagens de depuração podem ser complementadas com **ferramentas de depuração** como: compiladores de depuração, auxílio rastreadores de **depuração dinâmicos**, geradores de casos de teste automáticos, geradores de dumps de memória e geradores de mapas de referência cruzada.

O teste dinâmico pode complementar a análise estática. São verificados itens como:

- O tempo de resposta;
- O desempenho da aplicação;
- A capacidade do software se adaptar a diferentes ambientes;
- O comportamento funcional.

Técnicas de teste

Vamos entender primeiro o que é técnica.

Segundo a norma IEEE 610.12-1990, as técnicas de teste³ são procedimentos técnicos e gerenciais que ajudam na avaliação e melhoria do processo.

Saiba mais

Qual é o objetivo das técnicas de teste?

A criação de **casos de teste** que atendam aos objetivos globais da atividade de teste.

Entendendo melhor essa definição, teste de software é o processo de executar o software de uma **maneira controlada**, com o objetivo de descobrir diferenças entre o **xcomportamento previsto** e o **observado**.

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

Um **caso de teste** bem-sucedido é aquele que tem uma elevada probabilidade de revelar um erro ainda não descoberto.

Teste estrutural X testes funcionais

Estrutural

O teste estrutural também é do tipo caixa branca e tem por objetivo testar o código-fonte, verificando cada linha de código possível para avaliar os fluxos básicos e os alternativos.

O teste estrutural faz o exame de detalhes do procedimento com os seguintes tipos de teste:

————— 1 —————

Teste de caminhos lógicos

Casos de teste para exercitar conjuntos específicos de condições e/ou laços.

————— 2 —————

O estado do programa


Examinado em vários pontos para determinar se o estado esperado ou declarado corresponde ao estado real.

Não é possível testar todos os caminhos lógicos de um programa grande.

Abordagem para esse teste:

- Seleção de um número limitado de caminhos a serem exercitados;
- Estruturas de dados importantes podem ser investigadas para validação;
- Combinação de teste funcional e estrutural para validar a interface do software e assegurar (seletivamente) que as operações internas do software estão corretas.

 Testes estruturais

 Clique no botão acima.

Teste estrutural (caixa branca)

O teste estrutural, do tipo caixa branca, nos permite uma verificação mais precisa do funcionamento do software. Ele será projetado em função da **estrutura interna do sistema**.

O teste é realizado analisando o código-fonte e elaborando casos de teste que cubram as funcionalidades do componente de software.

Utilizamos essa técnica para complementar a técnica funcional. As informações obtidas pela aplicação desses critérios são bastante relevantes para ajudar as próximas atividades de manutenção, depuração e, principalmente, para a confiabilidade do software que está sendo desenvolvido.

Recomendamos a técnica de teste de caixa branca, porque a responsabilidade principal fica a cargo dos desenvolvedores do sistema, pois são eles os conhecedores do código produzido.

Teste de regressão

Os objetivos desse teste são:

- Garantir que nenhum defeito foi acrescentado ao sistema após sua modificação;
- Assegurar que as mudanças realizadas nessa nova versão não gerarão erros em componentes prontos e testados.

Nesse caso, são geralmente executados após a correção de algum defeito ou após a adição de uma nova funcionalidade.

Reteste de um sistema ou componente para verificar se alguma modificação recente causou algum efeito indesejado, além de certificar se o sistema ainda atende os requisitos.

Esse teste consiste em aplicar, antes e depois da alteração, todos os testes que já foram aplicados nas versões anteriores. É uma técnica **aplicável a cada alteração realizada no software**.

Por ter essa natureza de repetição, é imprescindível que seja adotada uma ferramenta de automação de testes. Ela é muito bem aplicada nas fases de testes de unidade, de integração e de sistema.

Teste de carga

Serve para **avaliar os limites operacionais do software**. Normalmente, as medições são tomadas com base na taxa de transferência de dados da carga de trabalho e no tempo de resposta da transação.

Essas variações na carga de trabalho tratam-se da confrontação das cargas de trabalho médias e máximas que ocorrem dentro de tolerâncias operacionais normais.

Essa técnica é aplicada durante as fases de testes de integração e de sistema.

Teste de estresse

É um teste de carga compreendendo cargas de trabalho extremas, memória insuficiente, hardware e serviços indisponíveis ou recursos compartilhados limitados. É destinado a **avaliar como o sistema responde em condições anormais**.

Esse teste é executado com muita antecedência para compreender melhor como e em quais áreas o sistema será dividido, para que os planos de contingência e a manutenção de atualização possam ser planejados.

Essa técnica é indispensável, principalmente para projetos que desenvolvam sistemas críticos, que necessitem de alta eficiência e disponibilidade. É indicada para ser executada durante a fase de teste de sistema.

Teste de usabilidade

Por que devemos fazer esse teste?

A usabilidade descreve a qualidade da interação dos usuários com uma determinada interface. Sendo assim, devemos avaliar o sistema do **ponto de vista do usuário final**.

Quais fatores devem ser levados em consideração?

Os fatores humanos, a estética, os manuais, a facilidade de uso, a satisfação subjetiva do usuário etc.

O que são identificados nesse teste?

Identificamos problemas de usabilidade, observação do comportamento dos usuários durante a utilização do sistema.

Em que fase eles devem ser realizados?

Por todos esses motivos, eles devem ser efetuados na fase de testes de aceitação.

Teste de segurança

O que devem ser validados nessa técnica? Precisamos validar os **requisitos de segurança**, visando identificar as vulnerabilidades do sistema.

Seus objetivos são: prevenir ataques, detectar vulnerabilidades e preparar medidas de contingência para casos de falhas.

É indicado durante as fases de testes de integração e de sistema.

Atenção! Aqui existe uma videoaula, acesso pelo conteúdo online

Funcional

Testa os requisitos funcionais do software, as funções e os casos de uso. Responde a seguinte questão:

Comentário

A aplicação faz o que deveria fazer?

Sendo um teste do tipo **caixa preta**, garante que os requisitos funcionem conforme o especificado.

Ele **não se preocupa com a forma** como foi implementado, ou seja, com o comportamento interno do sistema, mas sim **com a saída gerada** após a entrada dos dados especificados.

Desta forma, indicamos essa técnica para detectar erros de interface, de comportamento e/ou de desempenho.

Essa técnica também pode ser aplicada em **todas as fases de testes** (como vimos no modelo U: unidade, integração, sistema e aceitação, que definiremos a seguir).

Uma dificuldade dessa técnica, por questões de tempo e recurso, é testar todas as entradas possíveis. São inseridos alguns dados e espera-se na saída o resultado de como foram projetados os requisitos.

Essa técnica de teste é imprescindível durante o desenvolvimento de um sistema, porém, mostra-se insuficiente para identificar certos riscos num projeto de software.

Os testes funcionais também são classificados conforme segue:



Fases de testes (validação)

É importante que o processo de teste esteja presente durante todo o desenvolvimento do software. Porém, esses testes podem ser divididos em diferentes fases, as quais se diferenciam pela abstração e complexidade dos testes produzidos e executados em cada uma delas.

Geralmente esse processo de teste é dividido em quatro fases, são elas:

Testes de unidade



Testa um componente isolado ou classe do sistema. Nesta fase, os esforços dos testes estão concentrados nas menores unidades do software produzido.

O objetivo é detectar erros de lógica e/ou de implementação em pequenas partes do sistema, independentemente do restante. Consequentemente, cada unidade do sistema é testada isoladamente.

Isso contribui para assegurar a correção dos componentes individuais, “mas não garante que a integração dessas partes funcione como o esperado”.

Exemplo: dividir (x int, y int) = z int.

Caso tenhamos x=1 e y=0, z será um valor com erro e deverá retornar uma mensagem ao usuário, avisando que a operação é inválida.

Caso a expressão seja um dado comum do sistema, para fazer essa validação a autorização deverá ser do usuário, pois faz parte do conjunto de regras de negócio.

Testes de integração



Testa se um ou mais componentes combinados funcionam de maneira satisfatória. Pode-se dizer que é composto por vários testes de unidade.

O ponto central dos testes está voltado para a detecção de falhas provenientes da integração interna dos componentes do sistema. Para isso, os módulos são combinados e testados em conjunto.

Essa fase vem logo após os testes de unidade e antecede os testes de sistema, tendo como resultado o sistema integrado e preparado para os testes.

Não faz parte do escopo dessa fase testar a interação do sistema produzido com outros sistemas, que por ventura venham a se comunicar.

Objetivo dos testes de sistema:

Realizar a execução do sistema como um todo, dentro de um ambiente operacional controlado, para validar a perfeição e requinte na execução de suas funções, acompanhando cenários sistêmicos elaborados pelo profissional de requisitos do projeto. Devem retratar os requisitos funcionais e não funcionais do sistema;

Este tipo de teste é realizado por uma equipe de teste independente, e o analista de teste elabora os casos de testes, normalmente em conjunto com os desenvolvedores e atuando em um ambiente controlado, no caso o ambiente de teste;

O comportamento de todo o sistema/ produto é definido pelo escopo de um projeto ou programa de desenvolvimento;

O ambiente de teste deve corresponder o máximo possível ao objetivo final, ou ao ambiente de produção.

Esta é a fase na qual o software já está completamente integrado. Assim sendo, os testes apontam falhas em relação aos requisitos do sistema, no que diz respeito à comunicação com outros sistemas.

Os testes são realizados em condições bastante semelhantes (de ambiente, massa de dados etc.) com as que o usuário utilizará em produção.

Os testes de sistema não se limitam aos requisitos funcionais, mas também objetivam testar os requisitos não funcionais.

Testes de aceitação



Os testes de aceitação são, em geral, uma extensão dos testes de sistema. Nessa fase, o objetivo é verificar se o software está pronto e pode ser usado pelo usuário final.

Para isso, verifica-se se o sistema realiza as funções para as quais foi criado, satisfazendo as necessidades do cliente.

Os testes são planejados e projetados com o mesmo cuidado e nível de detalhe do teste do sistema.

Durante essa fase, os critérios de aceitação são conhecidos, o que permite a automação dos testes de aceitação, além da monitoração e medição.

Atividade

1. Um processo de desenvolvimento de software em geral tem como entrada os requisitos do sistema e como saída um produto fornecido. Analise as afirmativas sobre este tema:

I. O desenvolvimento de software envolve os estágios: análise e definição de requisitos, projeto do sistema, codificação, testes e entrega do sistema. Assim, o ciclo de vida do software descreve a vida do produto de software desde sua concepção até a implementação e entrega.

II. Um dos primeiros modelos propostos foi o cascata, no qual o desenvolvimento de um estágio deve terminar antes do próximo começar. O modelo V é uma variação do cascata, que mostra como as atividades de teste estão relacionadas com a análise e com o projeto.

III. O modelo cascata pode ser incrementado com atividades de prototipação, que é um modelo de processo efetivo em que partes do sistema são construídas rapidamente com o objetivo de validar os requisitos. Caso novas alternativas sejam discutidas, deve-se começar o ciclo em cascata novamente, abandonando-se o protótipo.

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 I está correto.
 - d) Apenas o item II está correto.
 - e) Apenas o item III está correto.
-

2. A utilização do modelo V minimiza os custos da qualidade do software; assim, segundo a regra de 10 de Myers, os testes devem ser iniciados nas inspeções/ revisões de código até os testes de software. Identifique se essa afirmação está certa ou errada.

3. No modelo V, que integra o ciclo de vida de desenvolvimento de software ao ciclo de teste, a validação refere-se ao desenvolvimento, enquanto a verificação se refere ao teste. Identifique se essa afirmação está certa ou errada.

4. _____ geralmente são executados após a correção de algum defeito ou após a adição de uma nova funcionalidade. Seu objetivo é garantir que nenhum defeito foi acrescentado ao sistema após sua modificação.

Assinale a alternativa que preenche a lacuna corretamente:

- a) Testes de regressão
 - b) Testes de estresse
 - c) Testes fumaça
 - d) Testes alfa
 - e) Testes integração
-

Notas

Requisitos funcionais ¹

Descrevem as funcionalidades do sistema. Estão diretamente ligados às especificações da tecnologia envolvida, do perfil do usuário e do tipo do sistema.

Ex.: O sistema deve permitir incluir e excluir fornecedores.

Requisitos não funcionais ²

Restrições sobre os serviços ou funções oferecidos pelo sistema.

Ex.: A impressão do boleto deve ser em no máximo 10 segundos. A consulta ao banco de dados financeiro não deve ultrapassar 3 segundos.

Técnica de teste ³

Processo que assegura o perfeito funcionamento de alguns aspectos de software ou de sua unidade.

Referências

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

BEIZER, Boris. Black-Box Testing: techniques for functional testing of software and systems. New York: Wiley, 1995.

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.

SPILLNER, Andreas. From V-model to W-model: establishing the whole test process conquest 2000. In: Workshop on testing non-functional software requirements. Nuremberg, Germany, 2000.

Próxima aula

- Estratégias de teste de software;
- Definições das estratégias de teste de software;
- Caixa branca X caixa preta.

Explore mais

Leia os textos:

- [Planejar o gerenciamento da qualidade – Escritório de projeto;](#)
- [Entenda as diferenças entre testes de aplicações dinâmicos e estáticos;](#)
- [Estudo da qualidade de software na Metodologia V-model e sua interação com metodologias ágeis \(SCRUM\);](#)
- [Leia o livro Engenharia de Software, de Roger Pressman. Editora McGraw Hill \(6ª edição\).](#)
- Você pode aprofundar seus conhecimentos sobre o tema ciclo de vida do processo de testes de software acessando o website do [Centro de Informática - UFPE](#)