Guia - Abordagens de Testes de Software

SUMÁRIO

1. INTRODUÇÃO

Este guia tem por finalidade descrever as abordagens para testes de software, sendo que uma abordagem envolve a definição de:

- Nível de teste (unitário, integração, sistema);
- Tipo de teste (de função, stress, volume, desempenho, usabilidade, distribuição, etc.);
- Método de teste (caixa preta e caixa branca)
- Técnicas de teste usadas (manuais e automatizadas);
- Critérios de avaliação usados (cobertura de teste baseada em código, cobertura de teste baseada em requisitos, número de defeitos, intervalo entre falhas, etc.).

2. NÍVEIS DE TESTES

Normalmente, o teste é aplicado a diferentes objetivos em diferentes estágios ou níveis de esforço de trabalho. Esses níveis distinguem-se normalmente pelos papéis mais habilitados para projetar e conduzir os testes, e pelas técnicas mais apropriadas para o teste em cada nível.

2.1. Testes Unitários

O esforço de teste unitário concentra-se em avaliar se todas as unidades estão funcionando de forma independente. Executado pelo Implementador durante o desenvolvimento da unidade, ele se concentra na verificação dos menores elementos testáveis do software. O teste unitário normalmente é aplicado para verificar se os fluxos de controle e de dados estão cobertos e funcionam conforme o esperado. Essas expectativas baseiam-se em como o componente participa da execução de um requisito.

2.2. Testes de Integração

O teste de integração é executado para garantir que os componentes de software funcionem corretamente quando combinados. O objetivo do teste de integração é detectar imperfeições ou erros nas especificações das interfaces dos componentes integrados.

- A técnica de teste de integração **Big Bang** consiste em combinar todas as unidades que compõem o sistema e depois efetuar os testes. Não deve ser usado para sistemas complexos, pois nesse caso, dificulta o diagnóstico e correção de falhas encontradas.
- A técnica de teste de integração **Incremental** consiste em testar partes do sistema e depois integrá-las umas às outras até que o sistema seja todo construído. Pode ser top-down, ou bottom-up, ou mista e deve ser baseada no grau de acoplamento e coesão dos componentes sendo integrados.
    - Na estratégia top-down inicia-se a integração a partir dos níveis mais altos da hierarquia de controle, sendo necessária à utilização de stubs para substituir os componentes ainda não integrados.
    - Na estratégia bottom-up inicia-se a integração pelos componentes de níveis mais baixos, seguindo em direção aos de níveis mais altos da hierarquia de controle, sendo necessária à utilização de drivers para substituir os componentes ainda não integrados.
    - Na estratégia mista, deve ser utilizada top-down para os níveis mais altos e bottom-up para os níveis mais baixos.

A escolha de determinada técnica de testes de integração é baseada no planejamento de integração do projeto.

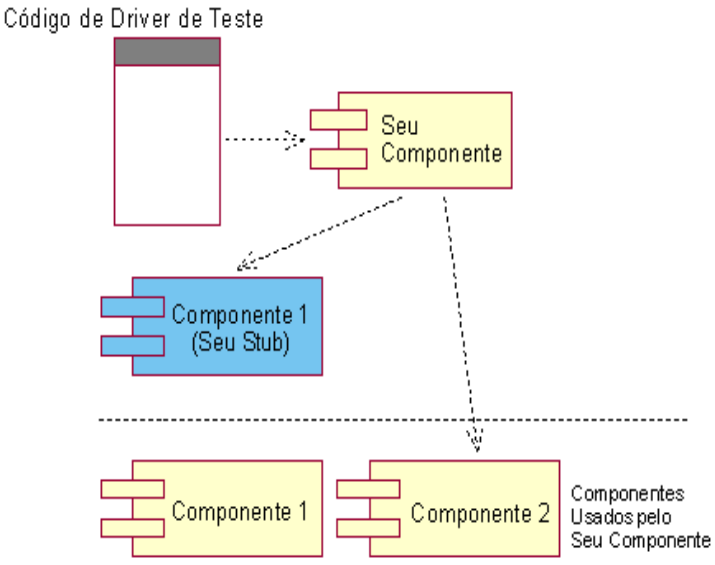
2.2.1. Componentes de Teste de Integração

De acordo com a estratégia de integração, testes são necessários sem que se tenham disponíveis todos os componentes de software que compõem o produto integrado. Nestes casos são criados Componentes de Teste de Integração (Drivers e Stubs) que simulam o funcionamento e a interface do produto de software sendo testado.

As seguintes situações motivam a criação de Drivers e Stubs:

- O componente encontra-se em desenvolvimento ou ainda não foi implementado;
- O componente possui defeitos que impeçam o funcionamento dos testes ou que fazem o testador perder muito tempo descobrindo que uma falha de teste não foi causada pelo componente;
- O componente pode dificultar a execução dos testes. Caso o componente insira restrições ao teste como janela de tempo de execução, autenticação de usuário, etc.
- O componente torna o teste muito lento, de maneira que os testes não sejam executados com freqüência suficiente. Por exemplo, a inicialização do banco de dados pode levar cinco minutos por teste.
- Situações excepcionais devem ser provocadas nos componentes para produzir certos resultados. Por exemplo, o teste de um tratamento de erros de comunicação da rede, ou falta de espaço em disco.

Os drivers e stubs são necessários somente durante os testes de integração, sendo substituídos pelos componentes de software “reais” à medida que estes sejam desenvolvidos.



Testes de integração com drivers e stubs

2.2.1.1. Drivers

Drivers são componentes de software usados para disparar um teste e, muitas vezes, fornecer dados de teste, controlar e monitorar execução e relatar resultados de teste. O driver de teste sequencia e controla a execução de um ou mais testes, passando informações para o produto ou componente de software sendo alvo do teste.

2.2.2.1. Stubs

Geralmente, os componentes de software dependem de outros componentes para concluir suas tarefas. Os problemas surgem quando esses componentes secundários não são operacionais. Às vezes, ainda estão em desenvolvimento, ou então têm muitos erros. De qualquer modo, o teste dos componentes principais não precisa ser interrompido até que os componentes secundários estejam disponíveis. Em vez disso, um stub ou componente temporário pode substituir qualquer componente não operacional para fins de teste. O

stub não implementa a funcionalidade do componente real, ele simplesmente reage a entradas. Os stubs retornam uma resposta programada para um determinado conjunto de valores, sem implementar qualquer lógica. É um simples relacionamento de estímulo/resposta.

2.3. Testes de Sistema

O teste de sistema consiste em testar o sistema como um todo, totalmente integrado.

O objetivo é assegurar que o produto de software e demais elementos que compõe o sistema, tais como, hardware e banco de dados, combinam-se adequadamente em relação à funcionalidade e desempenho desejados.

2.4. Testes de Aceitação

O teste de aceitação envolve a participação dos usuários finais nos testes com foco nas funcionalidades do sistema e em sua usabilidade.

O teste de aceitação do produto é o teste final antes da implantação do software. O objetivo desse teste é verificar se o software está pronto e pode ser usado pelos usuários finais para executar as funções e as tarefas para as quais o software foi criado.

Esse teste geralmente envolve mais do que a verificação da integridade do software. Também envolve todos os artefatos de produto fornecidos ao(s) cliente(s), como treinamento, documentação e pacotes.

3. TIPOS DE TESTES

A qualidade do produto de software é percebida em função de 5 dimensões, conhecidas como modelo **FURPS+:**

- Funcionalidade
- Usabilidade
- Confiabilidade
- Desempenho
- Suportabilidade

A avaliação da qualidade do produto de software é realizada através de diversos tipos de teste, todos eles relacionados a alguma dimensão da qualidade.

Dimensão de Qualidade	Tipo de Teste	Descrição	Ocorrência (Níveis de Teste)
Funcionalidade	Teste de função	Testes destinados a validar as funções do produto ou componente de software conforme as especificações de requisitos funcionais.	➢ Unitário; ➢ Integração; ➢ Sistema; ➢ Aceitação.

Dimensão de Qualidade	Tipo de Teste	Descrição	Ocorrência (Níveis de Teste)
	Teste de segurança	Testes destinados a garantir que o produto ou componente de software possa ser acessado apenas por determinados perfis de usuários ou atores. Esse teste é implementado e executado principalmente nos componentes de segurança do software, como os que realizam *login* do usuário.	➢ Unitário: no teste específico do componente de segurança. ➢ Sistema: avaliando as regras gerais de acesso. ➢ Aceitação.
	Teste de volume	Teste destinado a verificar a capacidade do produto ou componente de software em lidar com um grande volume de dados, como entrada e saída ou residente no banco de dados. O teste de volume abrange estratégias de teste, como, por exemplo, a entrada de dados do volume máximo de dados em cada campo ou a criação de consultas que retornem todo o conteúdo do banco de dados ou que tenham tantas restrições que nenhum dado seja retornado.	➢ Unitário: no teste específico do componente de acesso aos dados. ➢ Sistema: avaliando os requisitos gerais de volume. ➢ Aceitação.
Usabilidade	Teste de usabilidade	Testes que enfatizam: ➢ fatores humanos, ➢ estética, ➢ consistência na interface do usuário, ➢ ajuda on-line e contextual, ➢ assistentes e agentes, ➢ documentação do usuário e material de treinamento.	➢ Unitário: enquanto prova de conceito para avaliação dos requisitos de interface do software. ➢ Sistema: através da avaliação geral dos padrões de interface especificados. ➢ Aceitação.
Confiabilidade	Teste de integridade	Testes destinados a avaliar a robustez do produto ou componente de software (resistência a falhas) e a compatibilidade técnica em relação à linguagem, sintaxe e utilização de recursos.	➢ Unitário: enquanto prova de conceito arquitetural ➢ Integração.
	Teste de estrutura	Testes destinados a avaliar a adequação do produto ou componente de software em relação a seu design e sua formação. Em geral, esse teste é realizado em aplicativos habilitados para a Web, garantindo que todos os links estejam conectados, que o conteúdo apropriado seja exibido e que não haja conteúdo órfão.	➢ Sistema; ➢ Aceitação.

Dimensão de Qualidade	Tipo de Teste	Descrição	Ocorrência (Níveis de Teste)
	Teste de stress	Tipo de teste de confiabilidade destinado a avaliar como o sistema responde em condições anormais. O stress no sistema pode abranger cargas de trabalho extremas, memória insuficiente, hardware e serviços indisponíveis ou recursos compartilhados limitados. Deve ser planejado para que a carga seja constantemente aumentada até que o desempenho do sistema se torne inaceitável. Normalmente, esses testes são executados 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 e orçados com bastante antecedência.	➢ Sistema.
Desempenho	Teste de avaliação de desempenho	Tipo de teste de desempenho que compara o desempenho do produto ou componente de software (novo ou desconhecido) a um sistema e uma carga de trabalho de referência conhecidos. Requer instrumentação do sistema para monitorar o uso dos recursos, tempos de resposta para determinar as situações que levam à degradação do desempenho.	➢ Unitário: quando prova de conceito para avaliação de desempenho ➢ Sistema abrangendo o desempenho geral do software.
	Teste de carga	Tipo de teste de desempenho usado para validar e avaliar a aceitabilidade dos limites operacionais de um sistema de acordo com cargas de trabalho variáveis, enquanto o sistema em teste permanece constante. Em algumas variáveis, a carga de trabalho permanece constante e a configuração do sistema em teste é que varia. Geralmente, as medições são tomadas com base na taxa de transferência de dados da carga de trabalho e no tempo de resposta da transação alinhado. As variações na carga de trabalho normalmente incluem a emulação das cargas de trabalho médias e máximas que ocorrem dentro de tolerâncias operacionais normais.	➢ Unitário, quando prova de conceito para avaliação de desempenho. ➢ Sistema.
Suportabilidade	Teste de configuração	Teste destinado a garantir que o produto ou componente de software funcione conforme o esperado em diferentes configurações de hardware e/ou software. Normalmente utilizado quando requisitos não funcionais de portabilidade fazem parte do sistema, como a necessidade visualizar a interface Web em diferentes fabricantes e versões de navegadores.	➢ Unitário: quando prova de conceito de portabilidade ➢ Sistema.

Dimensão de Qualidade	Tipo de Teste	Descrição	Ocorrência (Níveis de Teste)
	Teste de instalação	Teste destinado a garantir que o produto ou componente de software do teste seja instalado conforme o esperado em diferentes configurações de hardware e/ou software e sob diferentes condições (como no caso de espaço insuficiente em disco ou interrupção de energia). Esse teste é implementado e executado em aplicativos e sistemas.	➢ Sistema

4. MÉTODOS DE TESTES

Os métodos de testes são definidos de acordo com o tipo de testes a ser aplicado.

Método de Teste	Base para os Testes	Descrição	Tipos de Testes
Caixa Branca	Código Fonte	Teste baseado na implementação do produto ou componente de software tem a base na informação obtida a partir do código. É chamado de teste de caixa branca, caixa de vidro ou estrutural. Pode ser usado para busca de falhas na estrutura do programa, e é focado em **como** o sistema opera. Teoricamente, cada caminho possível ao longo do código deve ser testado, mas isso só pode ser feito em unidades muito simples.	➢ Teste de segurança: quando necessário garantir que instruções do código não firam a segurança do sistema, como por exemplo: chaves e senhas visíveis no código, instruções que realizem transferências financeiras desautorizadas ou que desviem informações sigilosas do usuário. ➢ Teste de integridade: Avaliação do código quanto a ausência de pontos de falha e adequação aos padrões de implementação da linguagem.
Caixa Preta	Especificações	O método de teste baseado na especificação determina se todos os requisitos são atendidos. É chamado de caixa preta ou comportamental. Pode ser baseado nas especificações de requisitos, de projeto, ou em uma descrição informal do que o software deve fazer. É focado em **o que** o sistema deve fazer. É realizado sem o conhecimento de como o produto ou componente de software é implementado.	➢ Teste de função ➢ Teste de segurança ➢ Teste de volume ➢ Teste de usabilidade ➢ Teste de estrutura ➢ Teste de stress ➢ Teste de avaliação de desempenho ➢ Teste de carga ➢ Teste de configuração ➢ Teste de instalação

5. TÉCNICAS DE TESTES

As técnicas de testes definem como testar o produto ou o componente de software em relação ao método de teste. As técnicas podem ser combinadas simultaneamente para testes de um determinado produto ou componente de software.

Método	Técnica de Teste	Descrição
Caixa Preta	Particionamento de Equivalência	Esta técnica consiste em dividir o domínio de entrada do produto ou componente de software em classes de equivalência, minimizando o número de casos de teste. Uma classe de equivalência representa um conjunto de valores válidos e inválidos como condições de entrada para o produto de *software* que está sendo avaliado. Tipicamente, uma condição de entrada é um valor numérico específico, uma faixa de valores, um conjunto de valores relacionados ou um valor lógico. As classes de equivalência podem ser definidas de acordo com as seguintes condições de entrada: ➢ para um valor específico, são determinadas uma classe válida e duas inválidas; ➢ para uma condição expressa por uma faixa de valores, tem-se uma classe válida e duas inválidas; ➢ para um membro de um conjunto de valores, uma classe válida e uma inválida; e, ➢ para um valor lógico, serão definidas uma classe válida e uma inválida.
	Análise de valor limite	Técnica que assume que um número maior de erros tende a ocorrer mais nas fronteiras do domínio de entrada que nos valores centrais esperados. Este método é um complemento do particionamento de equivalência e tem como objetivo verificar se a aplicação trabalha corretamente quando exercitado os limites de seus valores de entrada Identificar os domínios de entrada, derivar casos de teste com valores que estejam nos limites de cada domínio segundo algumas regras: ➢ se uma condição de entrada especifica uma faixa delimitada pelos valores *a* e *b* os casos de teste devem ser projetados para trabalhar com os valores logo abaixo (*a* -- e *b* --) e acima (*a* ++ e *b* ++) de a e b, além dos próprios valores (*a* e *b*); ➢ se uma condição de entrada especifica um número de valores, os casos de teste devem ser projetados para exercitar os números máximos e mínimos bem como os valores logo acima e abaixo destes;
	Árvores de Decisão	Combinação de condições e ações.
	Baseados em estados	Consiste na verificação de estados e o modelo de sistema. Realizado através de máquina de estados finita.
Caixa Branca	Fluxo de controle	Os casos de teste são desenhados para executar os caminhos e pontos de decisão dentro de uma unidade de forma a cobrir as instruções, condições e ciclos (*loopings*).

Método	Técnica de Teste	Descrição
	Fluxo de dados	Cobertura de definições e uso de variáveis a fim de identificar situações onde dados são usados sem que tenham sido definidos.
	Análise de mutantes	Consiste em definir um conjunto de operadores de mutação e aplicar no produto ou componente de software em teste, gerando versões modificadas do produto (mutantes) para a detecção de erros que podem ser cometidos ao longo do desenvolvimento.

Complementarmente às técnicas definidas para cada abordagem de testes, é importante descrever como o teste será realizado aplicado às características específicas do produto ou componente de software a ser testado. Neste sentido deve-se considerar:

- **Testes manuais:** Realizados quando uma pessoa executa, de forma manual, um caso de teste sobre o produto ou componente de software. Embora testes manuais possam ser aplicados à qualquer nível, tipo, método ou técnica de testes, normalmente são utilizados para testes funcionais e não se aplicam à testes de desempenho devido à dificuldade de se obter resultados de teste precisos quando executados manualmente.

- **Testes automatizados:** Testes executados automaticamente, por meio de recurso computacional, através de scripts e condições pré-estabelecidas. A automatização de testes permite a repetição eficiente dos testes tornando-os ideais para testes de desempenho ou quando testes de regressão são planejados para produtos ou componentes de software.

- **Testes de regressão:** Consiste na re-execução dos casos de testes já aplicados para determinar se as alterações não produziram nenhum efeito indesejado. Deve ser utilizado todas as vezes que alguma alteração é feita no sistema. Testes de regressão podem ser realizados para todo o nível, tipo, método e técnica de teste, sendo geralmente, porém não obrigatoriamente, apoiado por ferramentas de automatização de testes.

6. CRITÉRIOS DE CONCLUSÃO DOS TESTES

Os critérios de conclusão dos testes determinam quando os testes realizados em determinada abordagem são considerados concluídos.

As principais medidas de um teste incluem a cobertura e a qualidade.

6.1. Medida da cobertura dos testes

A cobertura é a medida da abrangência do teste e é expressa pela cobertura dos requisitos e casos de teste ou pela cobertura do código executado.

As métricas de cobertura fornecem respostas à pergunta "Qual é a abrangência do teste?". As medidas de cobertura usadas com mais frequência são a cobertura de teste baseada em requisitos e em códigos. Em resumo, a cobertura de teste é qualquer medida de abrangência relacionada a um requisito (baseada em requisitos) ou a um critério de design/implementação do código (baseada em códigos), como a verificação de casos de uso (baseada em requisitos) ou a execução de todas as linhas de código (baseada em códigos).

Qualquer atividade sistemática de teste baseia-se em, pelo menos, uma estratégia de cobertura. Essa estratégia orienta o design de casos de teste declarando a finalidade geral do teste. A declaração da estratégia de cobertura pode ser tão simples quanto verificar todo o desempenho.

Se os requisitos estiverem completamente catalogados, uma estratégia de cobertura baseada em requisitos poderá ser suficiente para produzir uma medida quantificável para testar a abrangência. Por exemplo, se todos os requisitos do teste de desempenho foram identificados, é possível fazer referência aos resultados do teste para obter medidas, como 75% dos requisitos do teste de desempenho foram verificados.

Se a cobertura baseada em códigos for aplicada, as estratégias de teste serão formuladas em termos da quantidade do código-fonte que foi executada pelos testes. Esse tipo de estratégia de cobertura de teste é muito importante para sistemas de segurança crítica.

Ambas as medidas podem ser obtidas manualmente ou podem ser calculadas por ferramentas de automatização de testes.

6.2. Medida de qualidade dos testes

A qualidade é uma medida de confiabilidade, de estabilidade e de desempenho do objetivo do teste (sistema ou aplicativo em teste). Ela se baseia na avaliação dos resultados do teste e na análise das solicitações de mudança (defeitos) identificadas durante o teste.

Enquanto a avaliação da cobertura fornece a medida para testar a conclusão, uma avaliação dos defeitos encontrados durante os testes fornece a melhor indicação da qualidade do software. Qualidade é a indicação do grau em que o software satisfaz aos requisitos. Assim, nesse contexto, os defeitos são identificados como um tipo de solicitação de mudança na qual o objetivo do teste não satisfez aos requisitos.

A análise de defeitos significa examinar a distribuição de defeitos nos valores de um ou mais parâmetros associados a um defeito. Essa análise fornece uma indicação da confiabilidade do software.

Na análise de defeitos, existem quatro parâmetros principais que são geralmente usados:

- Situação - o estado atual do defeito (em aberto, em reparo, concluído, etc.).
- Prioridade - a importância relativa do defeito a ser relatado e solucionado.
- Gravidade - o impacto relativo do defeito. O impacto para o usuário final, uma organização, terceiros etc.
- Origem - onde está e qual é a falha original que está causando o defeito ou o componente que será corrigido para eliminá-lo.

Por exemplo, espera-se que as taxas de detecção de defeitos diminuam à medida que os testes e as correções avançam. É possível estabelecer um limite até onde o software será implantado.