# Teste de Integração:

Testes de integração têm a finalidade de verificar a interação entre diferentes unidades e componentes de software, que foram previamente testados de forma isolada. Sua implementação e condução geralmente é responsabilidade da equipe de desenvolvimento, com alguns desses testes também sendo executados pela equipe de QA (Quality Assurance). O foco do teste de integração é identificar falhas nos pontos de comunicação entre os módulos, garantindo que eles funcionem corretamente quando combinados.

O teste de integração é uma técnica sistemática utilizada para validar a interação entre componentes de software ao mesmo tempo que se constrói a arquitetura do sistema. Diferente dos testes unitários, que avaliam componentes isolados, os testes de integração focam na verificação das interfaces e na troca de dados entre diferentes partes do sistema. O objetivo é garantir que módulos individuais, previamente validados, funcionem adequadamente quando interconectados, construindo progressivamente a estrutura do software como um todo.

## As duas abordagens principais quem tem em testes de integração:

### Integração Incremental: Componentes são integrados e testados gradualmente, de forma incremental. Isso permite identificar erros em estágios mais precoces, facilitando a localização e correção de falhas, além de evitar que pequenos problemas se acumulem. Existem variantes, como integração ascendente (bottom-up) e descendente (top-down), dependendo de como os componentes são integrados.

### Integração "Big Bang": Todos os componentes são combinados de uma só vez, e o sistema completo é testado. Embora essa abordagem economize tempo de integração, tende a gerar uma grande quantidade de erros simultâneos, tornando a resolução de problemas muito mais complexa e demorada, já que é difícil isolar a causa das falhas. Segundo Pressman (2010), essa abordagem frequentemente resulta em um ciclo contínuo de detecção e correção de erros.

## Exemplo

Em sistemas de e-commerce módulos como "autenticação de usuários", "gerenciamento de produtos", "carrinho de compras" e "processamento de pagamentos" são feitos de maneira independente. O teste de integração garante que quando fizermos login, o usuário possa acessar a lista de produtos, adicionar itens ao carrinho e, em seguida, conectar-se corretamente ao módulo de pagamento para concluir a compra. Sem o teste de integração, podem surgir problemas como falhas na transferência de dados entre módulos, informações incorretas no carrinho ou erros no processamento de pagamento.

# Testes de Validação:

Os testes de validação têm como objetivo garantir que o software atenda às necessidades e expectativas dos usuários finais, verificando se o sistema entregue está em conformidade com os requisitos e funcionando conforme especificado. Esses testes começam após os testes de integração e se concentram nas ações visíveis e nas saídas reconhecíveis pelo usuário, validando se o comportamento do sistema corresponde ao que foi prometido durante a fase de requisitos.

Um dos princípios fundamentais da engenharia de requisitos é que os requisitos devem ser testáveis. Isso significa que, para cada requisito estabelecido, deve ser possível conceber um teste que verifique se ele foi satisfeito. A metodologia de testes de validação segue uma abordagem sistemática de planejamento, onde um conjunto de testes é elaborado para cada requisito do sistema, garantindo que todas as funcionalidades especificadas sejam rigorosamente verificadas.

A validação de software é alcançada por meio de uma série de testes que demonstram a conformidade do sistema com os requisitos levantados durante a fase de projeto. Um plano de teste define as categorias de testes a serem realizados, enquanto um procedimento de teste estabelece casos de teste específicos que precisam ser executados. Esses casos de teste são projetados para verificar, por exemplo, se todos os requisitos funcionais foram implementados corretamente, se o desempenho do sistema atende aos padrões estabelecidos e se a usabilidade está alinhada com as expectativas dos usuários. Além disso, a documentação e a precisão das informações fornecidas também são validadas (Pressman, 2010; Sommerville, 2011).

## Após a execução de cada caso de teste, duas condições são possíveis:

* A funcionalidade testada está em conformidade com a especificação e é aceita.
* Um erro de especificação é identificado, resultando em uma lista de deficiências que precisam ser corrigidas.

No entanto, Pressman (2010) aponta que os erros encontrados durante a fase de validação geralmente são difíceis de corrigir antes da entrega do software, o que pode atrasar o cronograma do projeto.

## Cenários de Teste

Uma abordagem comum na validação de software é o uso de cenários de teste, que são histórias realistas que descrevem como o sistema será utilizado em situações do dia a dia. Esses cenários permitem que casos de teste sejam desenvolvidos de forma a refletir situações práticas que os usuários reais enfrentarão ao utilizar o software. O objetivo é que os cenários sejam detalhados o suficiente para motivar as partes interessadas, que precisam reconhecer a importância de validar essas interações e acreditar na relevância dos testes. Ademais, os cenários devem ser fáceis de avaliar, de modo que os resultados esperados possam ser claramente comparados com os resultados obtidos (Sommerville, 2011).

Se cenários de uso foram criados durante a fase de engenharia de requisitos, eles podem ser reutilizados como parte do processo de validação. Um exemplo seria a criação de um cenário para um sistema de reservas de hotel, onde o usuário realiza todo o processo de reserva, desde a seleção de um quarto até o pagamento. O cenário deve simular todas as interações possíveis e deve ser suficientemente complexo para testar diferentes caminhos alternativos e variações de fluxo, como o pagamento com diferentes métodos ou a modificação da reserva.

## Exemplo

Em um projeto de desenvolvimento de um aplicativo bancário, por exemplo, o teste de validação seria utilizado para garantir que os clientes possam realizar transações financeiras, como transferências, pagamentos e consultas de saldo, de forma precisa e dentro das especificações fornecidas pelo banco.

Além disso, esses testes verificariam se o sistema lida corretamente com situações excepcionais, como erros de conexão ou tentativa de transações fora do saldo disponível, assegurando que todas as funcionalidades críticas estejam implementadas e operando conforme esperado.

# Testes de Recuperação:

Muitos sistemas devem se recuperar de falhas e continuar o processamento com pouco, ou nenhum, tempo indisponível. Em alguns casos, um sistema deve ser tolerante a falhas, isso é, falhas de processamento não devem fazer com que o sistema aborte. Em outros casos, uma falha de sistema deve ser corrigida dentro de um período específico. Caso contrário, um grande prejuízo econômico poderá ocorrer para a organização.

O teste de recuperação de falhas é um teste de sistema que força o software a falhar de várias maneiras, verificando se a recuperação é executada corretamente. Se a recuperação for automática (realizada pelo próprio sistema), os mecanismos de reinicialização, checkpoint e recuperação de dados são avaliados quanto à sua exatidão. Se a recuperação requer intervenção humana, o tempo médio de reparação é avaliado para determinar se está dentro de limites aceitáveis (Pressman, 2010).

## Cuidados

É importante garantir, sempre que possível, que uma falha de software não leve à falha geral do sistema. Portanto, deve-se examinar como o software interage com o seu ambiente imediato para assegurar que:

1. As falhas de software, tanto quanto possível, fiquem contidas nos limites da camada da pilha do sistema e não afetem gravemente a operação das outras camadas do sistema.
2. Você entenda o modo como os defeitos e as falhas nas outras camadas da pilha de sistemas podem afetar o software. É importante considerar também como podem ser criadas checagens no software para ajudar a detectar essas falhas e como o suporte pode ser fornecido para a recuperação a partir da falha. Como o software é inerentemente flexível, os problemas inesperados do sistema frequentemente são deixados para que os engenheiros de software os solucionem.

## Exemplo

Digamos que uma instalação de radar tenha sido feita de modo que ocorram fantasmas na imagem do radar. É inviável mudar o radar para um local com menos interferência, então os engenheiros de sistemas têm de descobrir outra maneira de remover esses fantasmas.

A solução pode ser melhorar a capacidade de processamento de imagens do software para remover esses fantasmas. Isso pode deixar o software mais lento, a ponto de o desempenho dele ficar inaceitável. Então, o problema pode ser caracterizado como uma falha de software, ao passo que, na verdade, trata-se de uma falha no processo de projeto do sistema como um todo.

Esse tipo de situação, na qual os engenheiros de software ficam com a responsabilidade de melhorar a capacidade do software sem aumentar o custo do hardware, é muito comum. Muitas falhas de software conhecidas não são uma consequência de problemas inerentes ao software, mas sim o resultado da tentativa de mudar o software para acomodar requisitos de engenharia de sistemas modificados.

Um bom exemplo foi a falha do sistema de bagagens do aeroporto de Denver (SWARTZ, 1996), onde o software de controle deveria lidar com as limitações do equipamento utilizado.

# Teste de Sistema:

Os testes de sistema são uma série de testes diferentes, cujo objetivo principal é a execução completa do sistema. Embora cada teste tenha uma finalidade específica, todos são executados para verificar se os elementos do sistema foram devidamente integrados, e se desempenharam suas funcionalidades da forma esperada. (Pressman, 2010).

Os componentes do sistema são integrados para criar um sistema completo. Esse processo encontra erros resultantes de interações imprevistas entre os componentes e de problemas de interface. Também busca mostrar que o sistema satisfaz tanto requisitos funcionais quanto não funcionais e testa suas propriedades emergentes. Nos sistemas grandes, esse processo pode ter várias etapas, nas quais os componentes são integrados e formam subsistemas testados individualmente antes de serem integrados ao sistema final.

## Exemplo

Em um software de gestão hospitalar, o teste de sistema seria aplicado para verificar se as funcionalidades de cadastro de pacientes, agendamento de consultas e emissão de relatórios médicos funcionam de forma integrada e sem erros, garantindo que o sistema como um todo opere de maneira adequada.

# Teste de Aceitação:

O teste de aceitação é um tipo de teste de usuário, em que o cliente testa formalmente um sistema para decidir se deveria aceitá-lo do fornecedor ou se mais desenvolvimento é necessário.

O teste de aceitação é uma parte inerente do desenvolvimento de sistemas personalizados. Os clientes testam um sistema, usando seus próprios dados, e decidem se ele deve ser aceito do desenvolvedor do sistema. A aceitação implica que o pagamento final pelo software deve ser feito.

## Existem seis estágios no processo de teste de aceitação:

### **Definir critérios de aceitação.**

Em condições ideais, essa etapa deve ocorrer no início do processo, antes de se assinar o contrato do sistema. Os critérios de aceitação devem fazer parte do contrato do sistema e ser aprovados pelo cliente e pelo desenvolvedor. Entretanto, na prática, pode ser difícil definir critérios tão no início do processo. Os requisitos detalhados podem não estar disponíveis, e os requisitos quase certamente mudarão durante o processo de desenvolvimento.

### **Planejar testes de aceitação.**

Essa etapa envolve decisões sobre recursos, tempo e orçamento tanto para o teste de aceitação quanto para o estabelecimento de um cronograma de testes. O plano do teste de aceitação também deve discutir a cobertura necessária dos requisitos e a ordem em que as características do sistema são testadas. Além disso, deve definir os riscos para o processo de teste, como as falhas do sistema e o desempenho inadequado, e discutir como esses riscos podem ser mitigados.

### **Derivar os testes de aceitação.**

Depois que os critérios de aceitação foram estabelecidos, podem ser criados testes para conferir se um sistema é aceitável ou não. Os testes de aceitação devem visar tanto as características funcionais quanto as não funcionais (por exemplo, desempenho) do sistema. Em condições ideais, eles devem proporcionar uma cobertura completa dos requisitos do sistema. Na prática, é difícil estabelecer critérios de aceitação completamente objetivos. Muitas vezes há escopo para argumentar sobre o teste mostrar ou não se um critério foi definitivamente cumprido.

### **Executar testes de aceitação.**

### 

Os testes de aceitação acordados são executados no sistema. Em condições ideais, essa etapa deve ocorrer no ambiente real em que o sistema será utilizado, mas isso pode ser disruptivo e impraticável. Portanto, um ambiente de teste de usuário pode ter de ser configurado para executar esses testes. É difícil automatizar esse processo, já que parte dos testes de aceitação pode envolver o teste das interações entre usuários finais e o sistema. Pode ser necessário algum treinamento dos usuários finais.

### **Negociar os resultados dos testes.**

É muito improvável que todos os testes de aceitação definidos venham a ser aprovados e que não haverá problemas com o sistema. Se for o caso, então o teste de aceitação está completo e o sistema pode ser entregue. Na maioria das vezes, alguns problemas serão descobertos. Nesses casos, o desenvolvedor e o cliente devem negociar para decidir se o sistema é bom o bastante para ser usado. Eles também devem concordar sobre como o desenvolvedor vai consertar os problemas identificados.

### **Aceitar ou rejeitar o sistema.**

Esse estágio envolve uma reunião entre os desenvolvedores e o cliente para decidir se o sistema deve ou não ser aceito. Se ele não for bom o bastante para ser utilizado, então é necessário mais desenvolvimento para consertar os problemas identificados. Feito isso, a fase de teste de aceitação é repetida.

Há quem ache que o teste de aceitação é uma questão contratual clara. Se um sistema não passar nos testes de aceitação, então ele deve ser rejeitado e o pagamento não deve ser realizado. No entanto, a realidade é mais complexa. Os clientes querem usar o software o quanto antes em virtude dos benefícios de sua instalação imediata. Eles podem ter comprado hardware novo, treinado pessoal e mudado seus processos. Podem estar dispostos a aceitar o software, apesar dos problemas, porque o custo de não usá-lo é maior do que o custo de contornar os problemas.

## Exemplo

Um sistema de gestão de estoque poderia incluir a verificação de que o sistema permite o registro e controle de todas as movimentações de entrada e saída de produtos, com a geração de relatórios precisos. Se esses critérios forem atendidos, o software será aceito e considerado pronto para a entrega.

# Referências bibliográficas

* <https://plataforma.bvirtual.com.br/Leitor/Publicacao/188186/pdf/0?code=Egs7p33IjDsxLGRLjSic6YyK6kQr4E4208pVnhfjZnO2oOKxuLgGBoZZJqeFKBEw3EvqHUXpAb/NepPGN47Zxw==> Acessado em 16/09/2024
* <https://plataforma.bvirtual.com.br/Leitor/Publicacao/150962/pdf/0?code=Eh7LdKE6HaxPxg6MGYI5YQP+Sh1RWD/s9q0qSimESIQAcwWeifx8yb5AzSCA0frYwCZ9KUGD4K0TDq+22p1FoA==> Acessado em 16/09/2024
* <https://plataforma.bvirtual.com.br/Leitor/Publicacao/168127/pdf/0?code=Abn7DYNpkTtPR0uItpTkfMKahSOTMKdlXSjeZ+uhutc4Vurm99bT6+oZ62APH3a0h4F56G2lmWvQ1GeVOyOppw==> Acessado em 16/09/2024