



Compliance & Quality Assurance

2025

QAOps e Estratégias de testes para garantia da Qualidade de Software: Níveis, Tipos, Técnicas, Planos, Roteiros e Casos de testes

Conteúdo

- Testes de Software
 - Conceitos Básicos
 - Terminologia
 - Psicologia do Teste
 - Os 7 Princípios do Teste de Software
 - Regra 10 de Myers
 - Modelo V
 - Níveis de teste (Unitário, Integração, Sistema, Aceitação)
 - Pirâmide de Testes

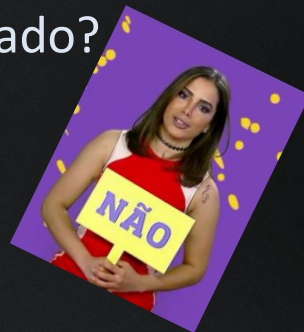
Conceitos Básicos

O que é teste?

1. Testar é o processo de demonstrar que não há mais erros presentes no software?



2. Testar é mostrar que o software executa corretamente as funções para as quais foi programado?



3. Testar é o processo de estabelecer confiança de que um software faz o que ele deve fazer?



Conceitos Básicos

O que é teste?

Testar é o processo de executar um programa de software com
a **intenção de encontrar erros.**

MYERS, Glenford J. - 1979

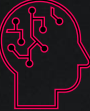




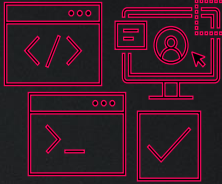
No entanto, não se trata apenas de executar casos de teste...

Conceitos Básicos

O que é teste?

Teste de software é um processo que inclui muitas atividades diferentes, sendo a **execução** do teste (incluindo a verificação dos resultados) apenas uma delas.

O processo de teste também inclui atividades como:

- **Planejamento** de testes 
- **Análise** 
- **Modelagem** de testes 
- **Implementação** dos testes 
- **Relatórios** de progresso e resultados de testes 
- Avaliação da qualidade de um objeto de teste 

Conceitos Básicos

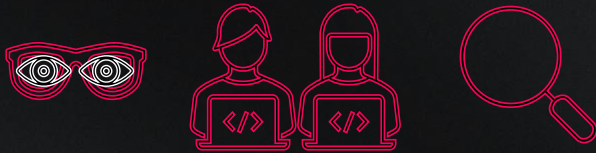
Terminologia

Teste Estático

Atividade de teste que contempla apenas a revisão (manual ou automática) de artefatos de teste ou do próprio código, porém sem executar o software.

Tipos de testes estáticos:

- Revisão
- Acompanhamento (*Walkthrough*)
- Inspeção



Exemplos de artefatos testados estaticamente:

- Requisitos e Casos de Uso
- Arquitetura & Design
- Código fonte
- Manual do usuário



Teste Dinâmico

Atividade de teste que envolve a execução do software, fornecendo entradas e avaliando as saídas e o comportamento apresentado.

Testes dinâmicos podem ser:

- Funcionais



- Não-funcionais



Conceitos Básicos

Terminologia



Erro: Também chamado de **engano** (*mistake*), é uma ação humana que produz um resultado incorreto.



Defeito: Também chamado de **falta** ou **bug**, é uma imperfeição ou deficiência em um produto de trabalho (código ou outro) causada por um erro.



Falha: Evento causado por um defeito no qual um sistema, ou parte dele, não executa uma função conforme os requisitos estabelecidos.

ERRO



DEFEITO



FALHA

Conceitos Básicos

Terminologia

Erros podem acontecer por razões como:

- Ignorância / Falta de conhecimento
- Pressão de tempo
- Negligência
- Inexperiência
- Complexidade
- Mal-entendido
- Falha de comunicação

Conceitos Básicos

Terminologia

Defeitos, causa-raiz e efeitos

As causas-raiz de um defeitos são as primeiras ações ou condições que contribuíram para o surgimento desse defeito.

A análise de causa-raiz (*Root-cause Analysis* – RCA) é a atividade de investigar profundamente o defeito para identificar suas causas-raízes, de modo que possam ser implementadas ações de melhoria que evitem que tais erros e defeitos voltem a se repetir no futuro.

Os efeitos são as consequências das falhas, como reclamações de clientes, perda de receita ou reputação, etc.

Conceitos Básicos

Terminologia

- **Caso de Teste:** conjunto de pré-condições, procedimentos e resultados esperados usado pelo testador para determinar se o sistema satisfaz o requisito ou funciona corretamente.



Conceitos Básicos

Terminologia

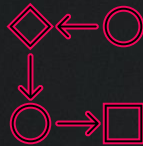
- **Suíte de Teste:** conjunto de casos de teste, organizados em uma ordem lógica, que devem ser executados em uma dada atividade de teste.



Conceitos Básicos

Terminologia

- **Plano de Teste:** documento que descreve o escopo, a abordagem, recursos necessários e cronograma previstos para as atividades de teste do software.
- Também indica as funções que serão testadas, quem executará as tarefas, o ambiente a ser usado, motivações das escolhas, riscos identificados e planos de contingência.



Conceitos Básicos

Terminologia

Verificação

Processo de avaliar criteriosamente **documentos, artefatos e o software** para se certificar que eles atendem aos requisitos especificados.

Estamos construindo o produto
da **maneira certa**?

Validação

Processo de avaliar **o software** com o intuito de avaliar se ele satisfaz as necessidades e expectativas do usuário **em seu ambiente de execução**.

Estamos construindo
o **produto certo**?

Conceitos Básicos

Psicologia do Teste

Teste é um processo "destrutivo", que vai **apontar** defeitos nos requisitos ou falhas no software.



Portanto, pode ser percebido com uma crítica pessoal pelas pessoas autoras/Devs.

A pessoa testadora, sendo portadora de más notícias com um resultado de teste que não passou, pode ser mal interpretada, criando uma tensão entre desenvolvimento e qualidade.



Comunicação construtiva:

- **Atividade colaborativa:** é objetivo comum construir sistemas de qualidade
- **Enfatizar benefícios do teste:** para o desenvolvedor, informação sobre defeitos ajuda a melhorar o produto e suas habilidades.
- **Informar resultados de maneira neutra:** relatórios focados no fato, na falha observada, sem críticas ou julgamentos.
- **Confirmar entendimento.**

Abordagem Desenvolvimento vs. Teste

Desenvolvedores e testadores geralmente pensam de forma diferente...

O objetivo do desenvolvedor é construir o produto, produzir código funcional, “que compile”.

O objetivo do testador é: encontrar falhas!

Saber disso é importante porque nas atividades de teste incumbidas ao desenvolvedor (testes unitários e de integração), ele deve estar ciente de que precisa mudar a mentalidade para criar testes eficazes.

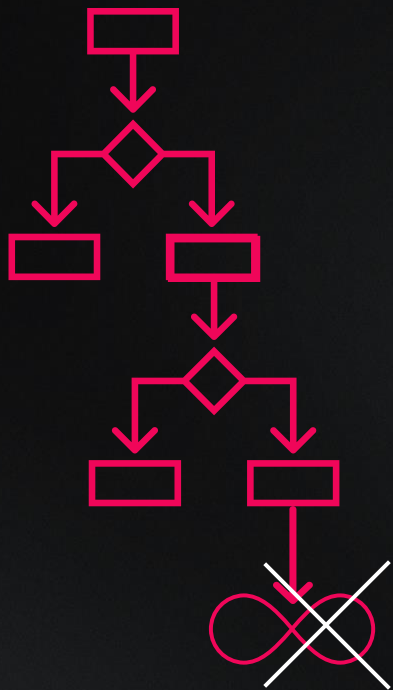
Os 7 Princípios do Teste de Software



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

O teste reduz a probabilidade de defeitos não descobertos permanecerem no software, mas, mesmo se nenhum defeito for encontrado, o teste não é uma prova de correção.

Os 7 Princípios do Teste de Software



2. Testes exaustivos são impossíveis

Testar tudo (todas as combinações de entradas e pré-condições) não é viável, exceto em casos triviais.

Em vez de tentar testar exaustivamente, a análise de risco, as técnicas de teste e as prioridades devem ser usadas para concentrar os esforços de teste.

Os 7 Princípios do Teste de Software

3. O teste inicial economiza tempo e dinheiro

Para encontrar antecipadamente os defeitos, as atividades de teste estático e dinâmico devem iniciar o mais cedo possível no ciclo de vida de desenvolvimento de software.

O teste inicial é por vezes referido como **shift-left testing**.

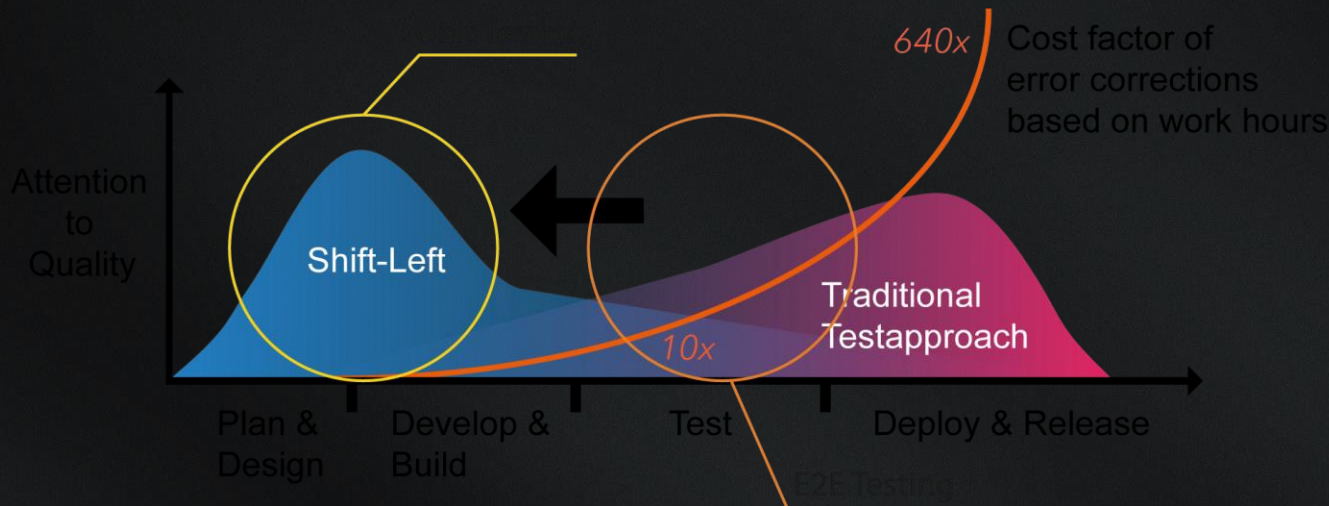
Continua ->

Os 7 Princípios do Teste de Software

3. O teste inicial economiza tempo e dinheiro

O teste no início do ciclo de vida de desenvolvimento de software ajuda a reduzir ou eliminar alterações dispendiosas*. Ou seja: economiza dinheiro ao evitar mudanças caras mais tarde.

*Dispendiosas significa que algo requer muito esforço, recursos ou dinheiro.



Os 7 Princípios do Teste de Software

4. Defeitos se agrupam

Um pequeno número de módulos geralmente contém a maioria dos defeitos descobertos durante o teste de pré-lançamento ou é responsável pela maioria das falhas operacionais.

Agrupamento de defeitos previstos e os agrupamentos de defeitos observados reais em teste ou produção, são uma entrada importante em uma análise de risco usada para focar o esforço de teste (como mencionado no princípio 2).

Continua ->

Os 7 Princípios do Teste de Software



4. Defeitos se agrupam

Isso ocorre devido a diversos fatores, como:

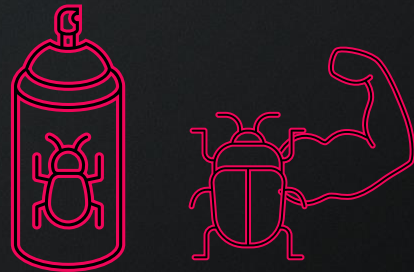
- Similaridade na origem dos defeitos;
- Erros de projeto ou implementação que afetam múltiplas partes do código;
- E a tendência humana de cometer erros em padrões consistentes.



Os 7 Princípios do Teste de Software

5. Cuidado com o paradoxo do pesticida

Se os mesmos testes forem repetidos várias vezes, esses testes não encontrarão novos defeitos.



Para detectar novos defeitos, os testes existentes e os dados de teste podem precisar ser alterados e novos testes precisam ser gravados.

(Testes não são mais eficazes em encontrar defeitos, assim como pesticidas não são mais eficazes em matar insetos depois de um tempo.)

Continua ->

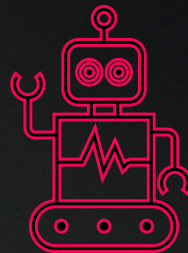
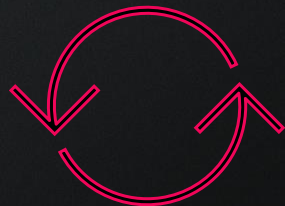
Os 7 Princípios do Teste de Software

5. Cuidado com o paradoxo do pesticida

Em alguns casos o paradoxo do pesticida tem um resultado benéfico, como no teste de regressão automatizado.

Isso porque, apesar do paradoxo do pesticida, o resultado é positivo porque geralmente há um número relativamente baixo de defeitos de regressão.

Em resumo, o teste de regressão automatizado ainda é benéfico, apesar das possíveis complicações.



Os 7 Princípios do Teste de Software

6. O teste depende do contexto

O teste é feito de forma diferente em diferentes contextos.

Por exemplo:

O software de controle industrial de segurança crítica é testado de forma diferente de um aplicativo móvel de comércio eletrônico.



Outro exemplo:

O teste em um projeto ágil é feito de forma diferente do que o teste em um projeto de ciclo de vida sequencial.

Ciclo de vida sequencial = Cascata



Os 7 Princípios do Teste de Software

7. A ausência de erros é uma ilusão

Algumas organizações esperam que os testadores executem todos os testes possíveis e encontrem todos os defeitos, mas os princípios 2 e 1 nos dizem que isso é impossível.

Além disso, a crença de que apenas encontrar e corrigir muitos defeitos assegura o sucesso de um sistema é uma ilusão.



Continua ->

Os 7 Princípios do Teste de Software

7. A ausência de erros é uma ilusão

Por exemplo, testar exaustivamente todos os requisitos e corrigir todos os defeitos ainda pode resultar em um sistema difícil de usar, que não atenda às necessidades dos usuários ou que seja inferior a outros sistemas concorrentes.

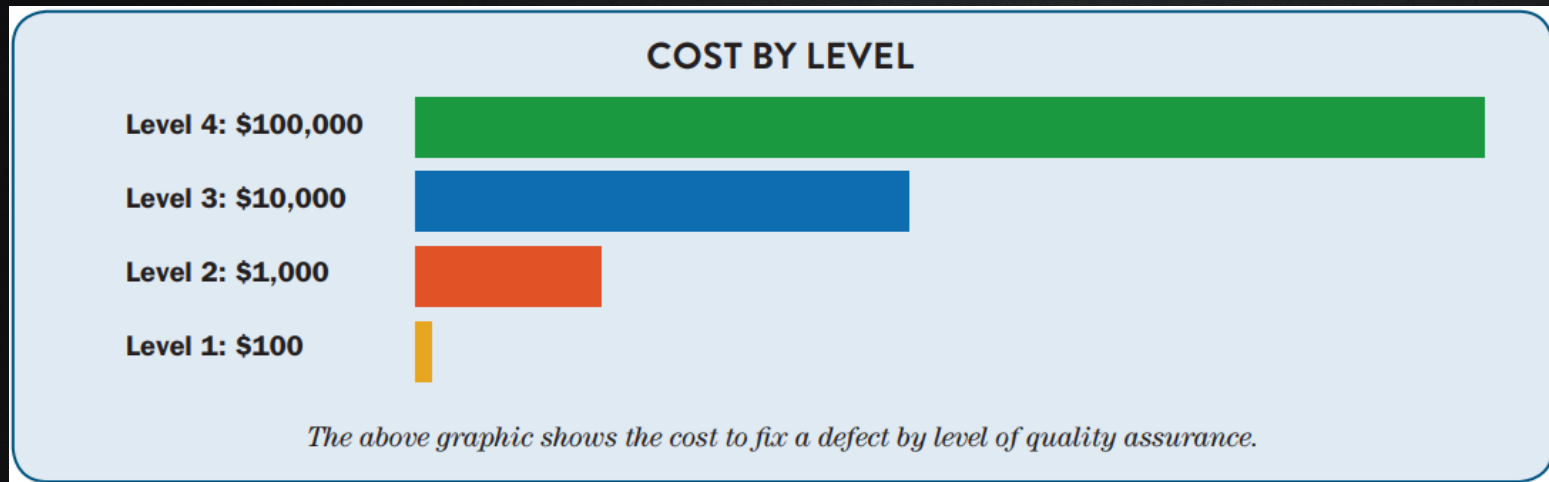
A Regra 10 de Myers

A Regra 10 de Myers sugere que é necessário cerca de dez vezes mais esforço para identificar, corrigir e verificar um defeito de software encontrado em uma determinada fase de testes do que teria sido necessário para prevenir esse defeito durante a fase anterior ou inicial de desenvolvimento.

É a mesma ideia do **Princípio 3** visto anteriormente e mais uma justificativa para shift-left.

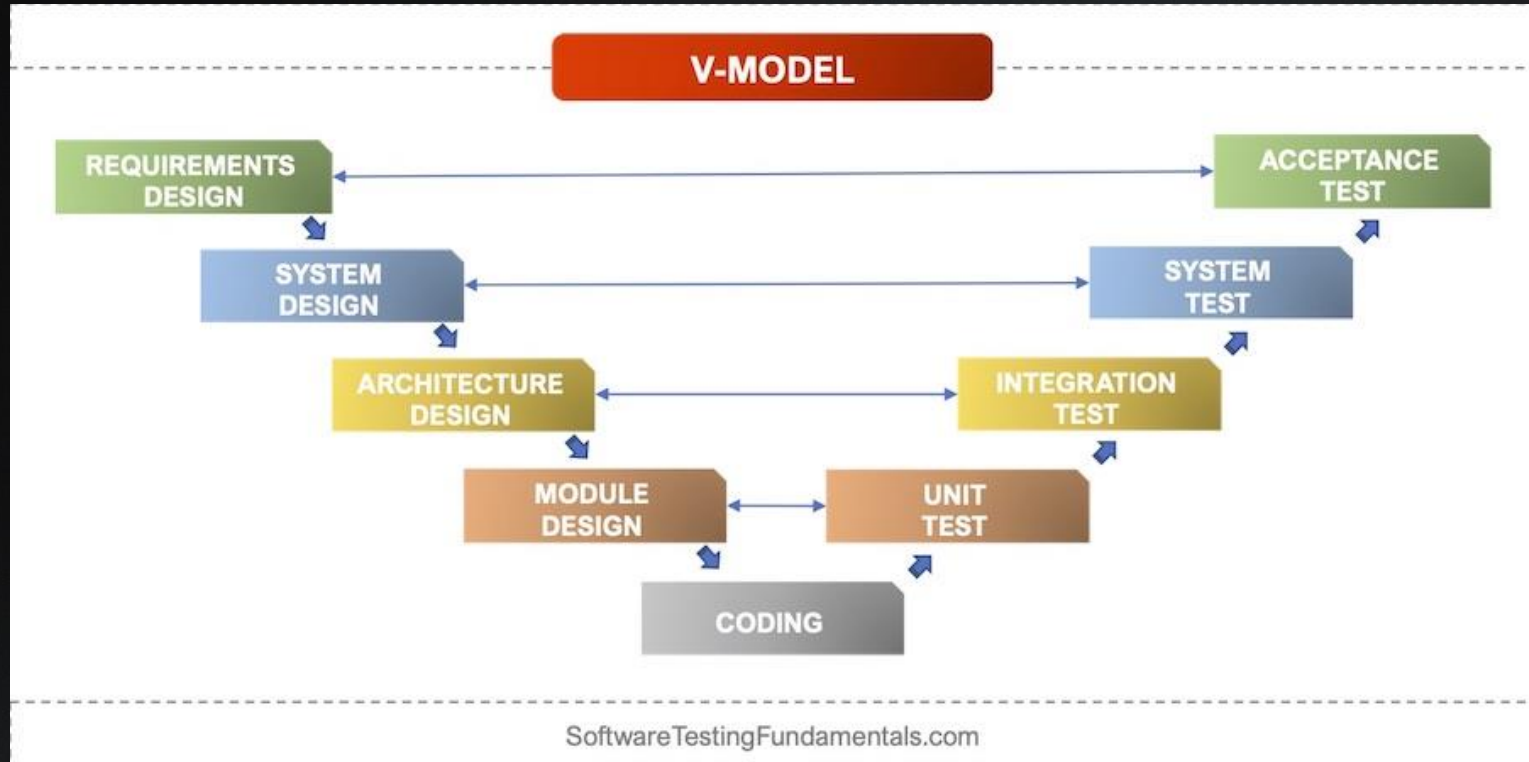
Continua ->

A Regra 10 de Myers



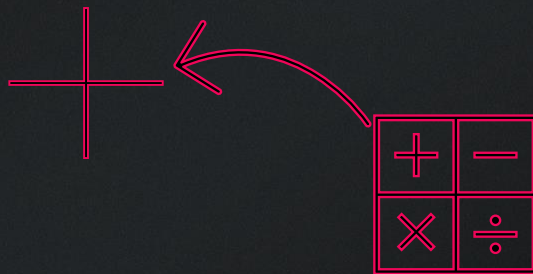
A Regra 10 de Myers enfatiza que o foco dos testes de software e do SQA não deve se limitar a encontrar e corrigir defeitos, mas sim a preveni-los desde o início.

Modelo V



Níveis de Teste

Testes Unitários



Consiste em escrever testes automatizados para testar pequenas unidades de código, geralmente funções ou métodos individuais, para garantir que eles funcionem como esperado.

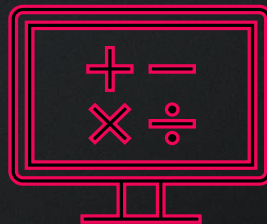
Testes unitários são geralmente escritos por desenvolvedores e executados durante o processo de desenvolvimento de software.

Exemplo:

Testar individualmente cada função do software, como o cálculo de juros em um empréstimo.

Níveis de Teste

Testes de Integração



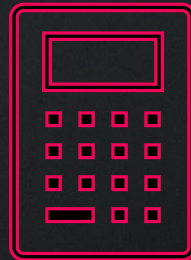
Se concentram em validar as interações entre componentes (unidades de código) ou sistemas, garantindo que eles trabalhem em conjunto corretamente e que suas interações não introduzam nenhum defeito ou comportamento inesperado.

Exemplo:

Testar a integração entre diferentes módulos do software, como a conexão entre o módulo de conta corrente e o módulo de transferência de fundos.

Níveis de Teste

Testes de Sistema



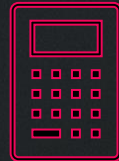
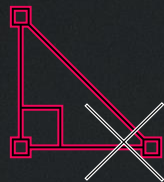
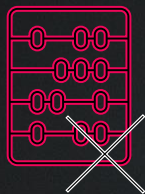
Se concentram no comportamento e nas capacidades de todo um sistema ou produto, geralmente considerando as execuções das tarefas de ponta a ponta do sistema e os comportamentos não-funcionais exibidos ao executar tais tarefas.

Exemplo:

Testar o sistema como um todo, incluindo todos os seus módulos, funcionalidades e interfaces com o usuário.

Níveis de Teste

Testes de Aceitação

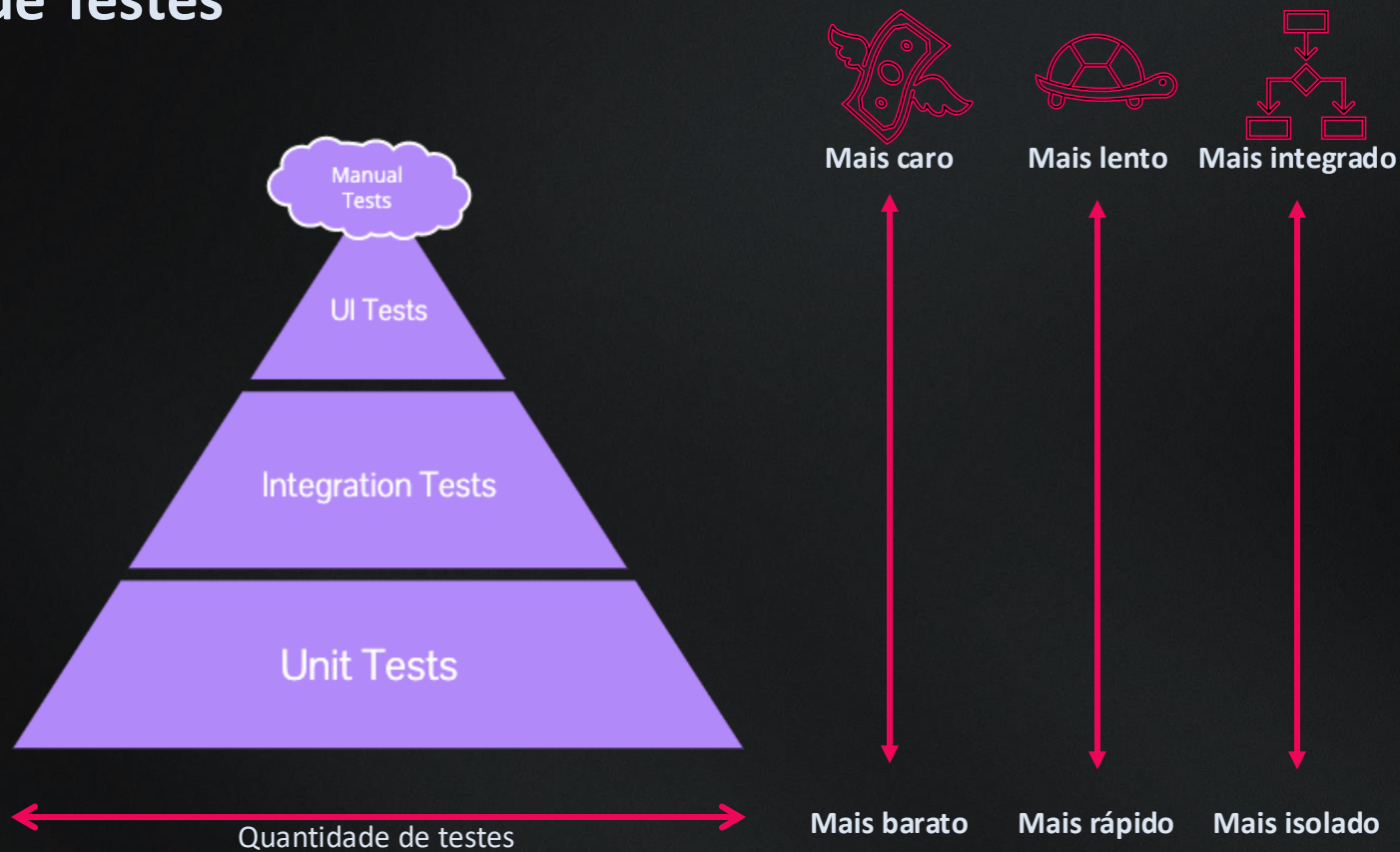


Relacionados às necessidades do usuário, requisitos e processos de negócios, executados para determinar se um sistema satisfaz ou não os critérios de aceitação e para permitir que o usuário determine se aceita ou não a entrega.

Exemplo:

Solicitar que os usuários utilizem o software para realizar operações bancárias e coletar feedback sobre a usabilidade e funcionalidade do sistema.

Pirâmide de Testes



Referências

ISTQB CTFL Syllabus v4: https://bcr.bstqb.org.br/docs/syllabus_ctfl_4.0br.pdf
<https://softwaretestingfundamentals.com/>
<https://malenezi.github.io/malenezi/SE401/Books/114-the-art-of-software-testing-3-edition.pdf>