
FIAP
FIAP
SLIDER

COMPLIANCE & QUALITY ASSURANCE

Prof. M.Sc. Felipe Desiglo Ferrare
proffelipe.ferrare@fiap.com.br

Introdução aos Testes de Software
Aula 2

Cronograma

- Recap da ultima aula
- Conceitos
- Terminologia
- Psicologia do Teste
- Regra 10 de Myers
- Modelo V
- Níveis de Teste
- Pirâmide de Testes

Nos Últimos capítulos (Ultima aula)

- O Que é Qualidade
  - Depende do contexto (expectativa)
  - É de responsabilidade de todos
  - Testes servem para mitigar riscos
  - Qualidade de processo VS qualidade de produto
  - Funcional VS Não Funcional
  - Certificações (ISTQB, ASQ)

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 produtos 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

- **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.

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

- **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

- **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.

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

Psicologia do Teste

- Desenvolvedor – Objetivo de construir
- QA – Objetivo de manter a qualidade

Objetivos podem ser conflitantes
Críticas podem ser vistas como pessoais
Não perceber que a qualidade faz parte do processo de entrega

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.

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. O teste no início do ciclo de vida de desenvolvimento de software ajuda a reduzir ou eliminar alterações dispendiosas.

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).

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.) Em alguns casos, como o teste de regressão automatizado, o paradoxo do pesticida tem um resultado benéfico, que é o número relativamente baixo de defeitos de regressão.

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. Como 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.

Os 7 Princípios do Teste de Software

7. A ausência de erros é uma ilusão

Algumas organizações esperam que os testadores possam executar todos os testes possíveis e encontrar todos os defeitos possíveis, mas os princípios 2 e 1, respectivamente, nos dizem que isso é impossível. Além disso, é uma ilusão (isto é, uma crença equivocada) esperar que apenas encontrar e corrigir muitos defeitos assegure o sucesso de um sistema. Por exemplo, testar exaustivamente todos os requisitos especificados e corrigir todos os defeitos encontrados ainda pode produzir um sistema difícil de usar, que não atenda às necessidades e expectativas dos usuários ou que seja inferior em comparação com outros sistemas concorrentes.

A Regra 10 de Myers

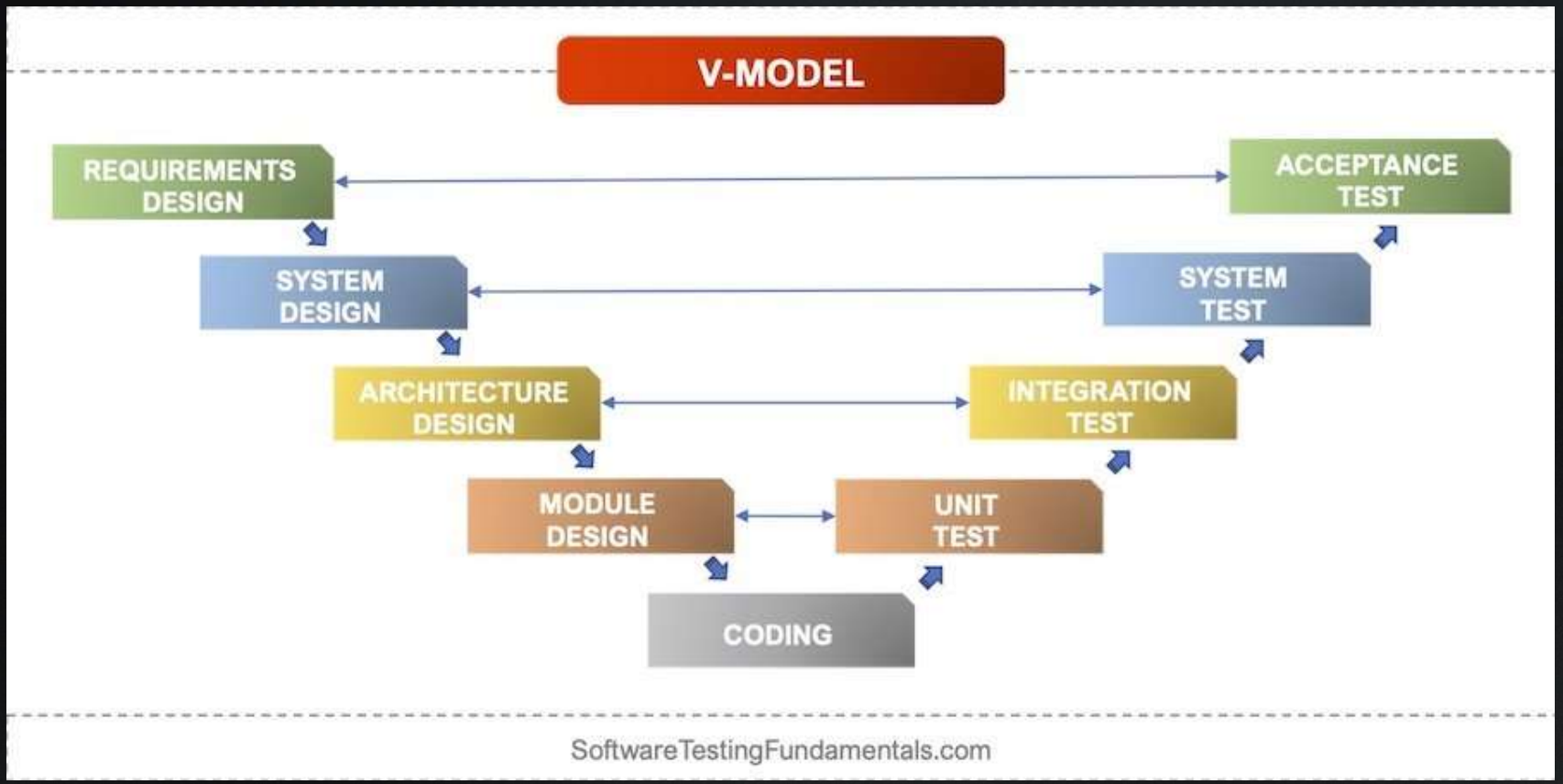
- Prevenir defeitos
- Corrigir eles o quanto antes

Modelo V

Múltiplos Vs

- Validação = “Estamos planejando o processo para atingir nosso objetivo ?”
- Verificação = “Estamos construindo o produto certo?”

Modelo V


V-MODEL
REQUIREMENTS DESIGN
ACCEPTANCE TEST
SYSTEM DESIGN
SYSTEM TEST
ARCHITECTURE DESIGN
INTEGRATION TEST
MODULE DESIGN
UNIT TEST
CODING
SoftwareTestingFundamentals.com

Níveis de Teste

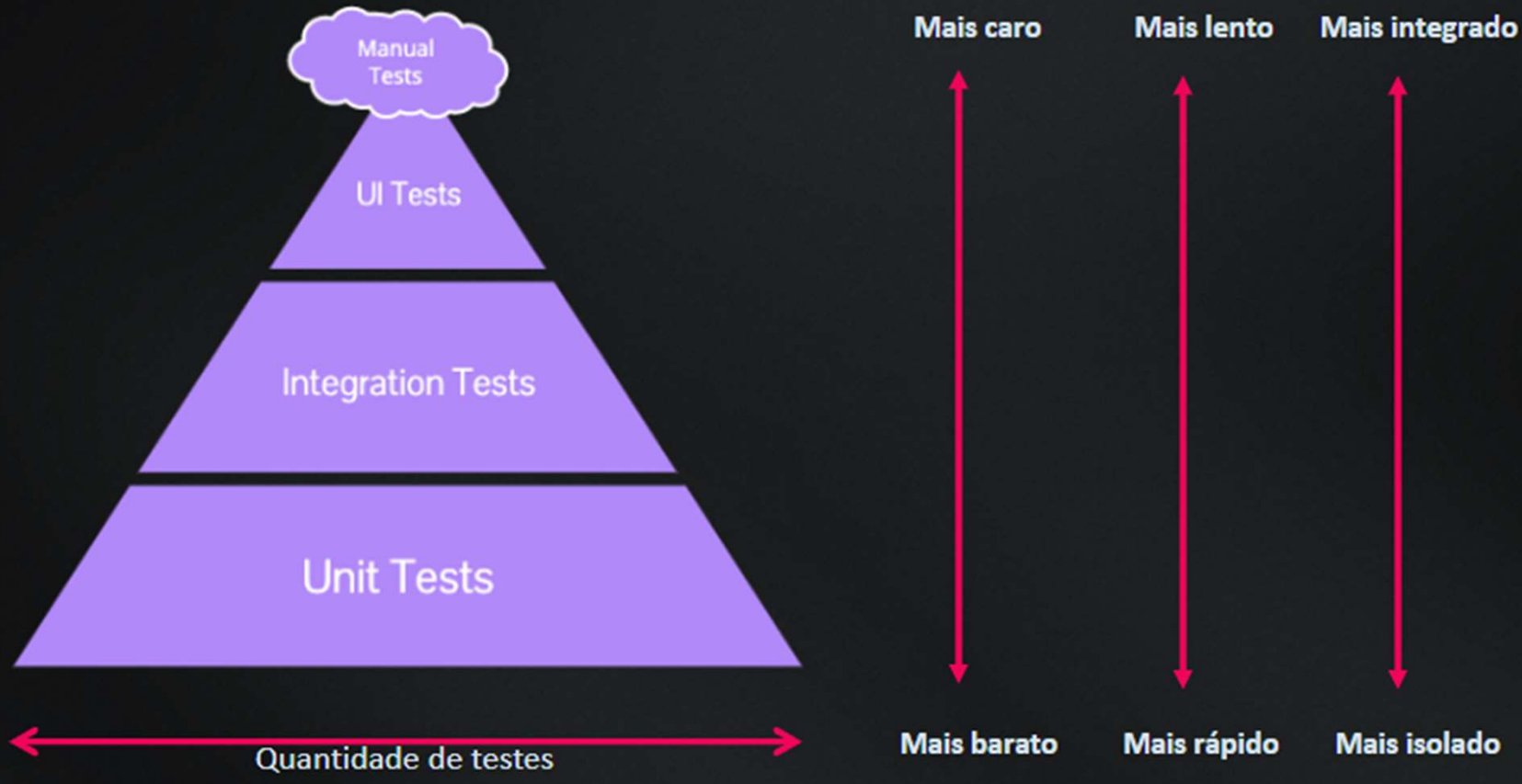
Testes Unitários: consiste em escrever testes automatizados para testar pequenas unidades de código, geralmente funções ou métodos individuais, para garantir que eles funcionem como esperado. Testes unitários são geralmente escritos por desenvolvedores e executados durante o processo de desenvolvimento de software.

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.

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.

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.

Pirâmide de Testes



Referências

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

Obrigado!


FIAP