

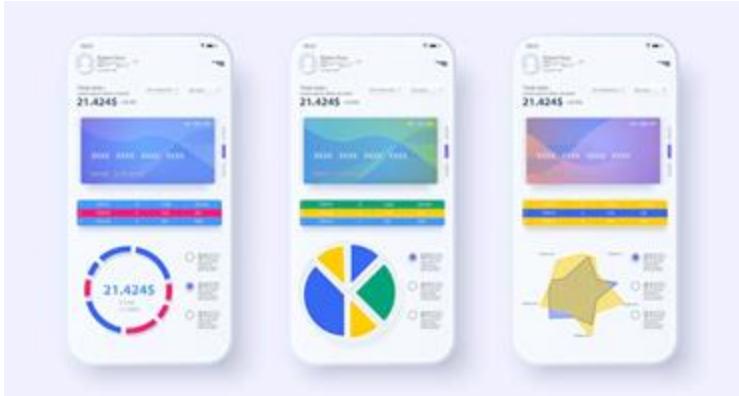
Introdução ao Desenvolvimento Moderno de Software

Denilson Bonatti
Tech Lead - DIO

DIO Bank

Bancos digitais são instituições financeiras que executam suas atividades de forma 100% online. Ou seja, praticamente tudo que o cliente precisa é feito por um celular via aplicativo ou no computador via Browser.

The screenshot shows the DIO Bank website's main menu. On the left, there is a sidebar with icons for Home, Conta Corrente, Transferências, Pagamentos e Recargas, Cartões, Investimento e Previdência, and Emprestimos e Financiamentos. The main content area is titled "Emprestimos e Financiamentos" and lists various services: Empréstimo (Simular e contratar, Meus Empréstimos, Consultar, Crédito Consignado, Crédito Pessoal, UVíque seu crédito, Anteja seu CIP, Salve, Anteja seu Imposto de Renda), Financiamento de Imóveis (Simular e Contratar, Consultar), Financiamento de veículos (Simular e Contratar, Acompanhar Financiamentos), and Consórcio (Simular e Contratar, Portal do Consórcio, Antecipação de parcelas, Ademissão via chat). There is also a section for "Outros serviços" (Agendamentos, Comprovantes, Transações realizadas).



Sistema Desktop X Sistema Web

Ao iniciar o desenvolvimento de um software, o primeiro passo é definir a(s) plataforma(s) onde este software será executado.

Sistema Desktop

Sistemas desktop são sistemas autônomos que podem ser instalados no computador. Esta instalação normalmente é realizada por um arquivo executável. Como exemplo, temos o Microsoft Word, Microsoft Excel, Anti-vírus e jogos.

Date	Services	Products	Total	7-Day Avg	Goal
2010-08-18	1245		2799	2,799	2500
2010-08-19	8750		2,098	2,098	2500
2010-08-20		1259	301*	1,659	2500
2010-08-21			-21*	1,538	2500
2010-08-22	4678		1,266	1,266	2500
2010-08-23		3326	0*	1,014	2500
2010-08-24		3799	0*	946	2500
2010-08-25	867		416*	754	2500
2010-08-26	3712	1205	1337	187	2500
2010-08-27	4609	3424	0*	200	2500
2010-08-28	8312		2320*	575	2500
2010-08-29			2814*	985	2500
2010-08-30			81*	396	2500
2010-08-31			447*	1,060	2500
2010-09-01			1,027*	1,258	2500
2010-09-02			3298*	1,542	2500
2010-09-03					
2010-09-04					
2010-09-05					
2010-09-06					
2010-09-07					
2010-09-08					
2010-09-09					
2010-09-10					
2010-09-11					
2010-09-12					
2010-09-13					
2010-09-14					
2010-09-15					
2010-09-16					
2010-09-17					
2010-09-18					
2010-09-19					
2010-09-20					
2010-09-21					
2010-09-22					
2010-09-23					
2010-09-24					
2010-09-25					
2010-09-26					
2010-09-27					
2010-09-28					
2010-09-29					
2010-09-30					
2010-09-31					
2010-10-01					
2010-10-02					
2010-10-03					
2010-10-04					
2010-10-05					
2010-10-06					
2010-10-07					
2010-10-08					
2010-10-09					
2010-10-10					
2010-10-11					
2010-10-12					
2010-10-13					
2010-10-14					
2010-10-15					
2010-10-16					
2010-10-17					
2010-10-18					
2010-10-19					
2010-10-20					
2010-10-21					
2010-10-22					
2010-10-23					
2010-10-24					
2010-10-25					
2010-10-26					
2010-10-27					
2010-10-28					
2010-10-29					
2010-10-30					
2010-10-31					
2010-11-01					
2010-11-02					
2010-11-03					
2010-11-04					
2010-11-05					
2010-11-06					
2010-11-07					
2010-11-08					
2010-11-09					
2010-11-10					
2010-11-11					
2010-11-12					
2010-11-13					
2010-11-14					
2010-11-15					
2010-11-16					
2010-11-17					
2010-11-18					
2010-11-19					
2010-11-20					
2010-11-21					
2010-11-22					
2010-11-23					
2010-11-24					
2010-11-25					
2010-11-26					
2010-11-27					
2010-11-28					
2010-11-29					
2010-11-30					
2010-12-01					
2010-12-02					
2010-12-03					
2010-12-04					
2010-12-05					
2010-12-06					
2010-12-07					
2010-12-08					
2010-12-09					
2010-12-10					
2010-12-11					
2010-12-12					
2010-12-13					
2010-12-14					
2010-12-15					
2010-12-16					
2010-12-17					
2010-12-18					
2010-12-19					
2010-12-20					
2010-12-21					
2010-12-22					
2010-12-23					
2010-12-24					
2010-12-25					
2010-12-26					
2010-12-27					
2010-12-28					
2010-12-29					
2010-12-30					
2010-12-31					

Sistema Web

Sistemas baseados em tecnologia web, podendo ser utilizados remotamente através de qualquer navegador de internet, sem a necessidade de instalação e atualização local.

The screenshot displays a web-based application interface with a green header bar containing the logo 'pontomais' and several icons. The main content area is divided into several sections:

- Indicadores:** A sidebar menu item.
- Meu perfil:** A sidebar menu item.
- Mais configurações:** A sidebar menu item.
- Meus pontos:** A sidebar menu item.
- Introdução:** A sidebar menu item.
- Controle de ponto:** A sidebar menu item.
- Recursos:** A sidebar menu item.
- Configurações:** A sidebar menu item, currently selected.
- Análise com amigo:** A sidebar menu item.
- Notificações:** A sidebar menu item.

The main content area includes:

- A top navigation bar with buttons for 'Novo ponto', 'Adicionar ponto', and 'Prestar'.
- A section titled 'Introdução automática' with five columns: 'Ponto', 'Ponto', 'Ponto', 'Ponto', 'Missa de Fim de'.
- A section titled 'Semana I' with a table showing data for each day of the week (Segunda-Feira to Sábado) across four columns: 'Dia da semana', 'Ponto', 'Ponto', 'Ponto', 'Início da Missa', 'Missa de Fim de', and 'Lembrete de missas entre'.
- Total values for the week: 'Dias de horário regular total: 40000', 'Prestação de horas regular reduzida: 0000', and 'Total geral de horas: 40000'.
- A large green button at the bottom right labeled 'Aplicar'.

E as aplicações móveis?

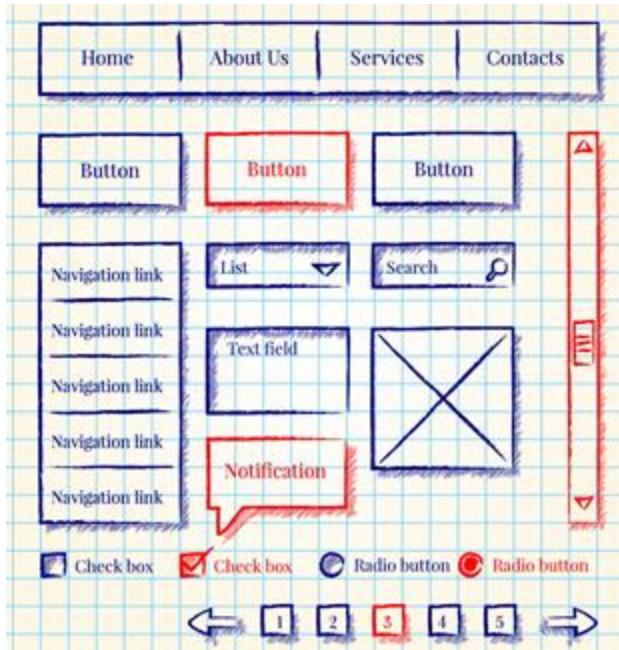
Uma aplicação móvel ou aplicativo mobile é um software desenvolvido para ser instalado em smartphones e/ou tablets. É baixado de uma loja on-line, como Google Play ou App Store, direto para o seu dispositivo portátil.



UX Design (User Experience)

O design da experiência do usuário (ou UX design) é o processo que visa melhorar a satisfação do usuário com um produto ou serviço, melhorando a usabilidade, a acessibilidade e até mesmo a satisfação proporcionada na interação.





COMPORTAMENTO

UX

PESQUISA

PROTOTIPOS

PERSONAS

OBJETIVOS

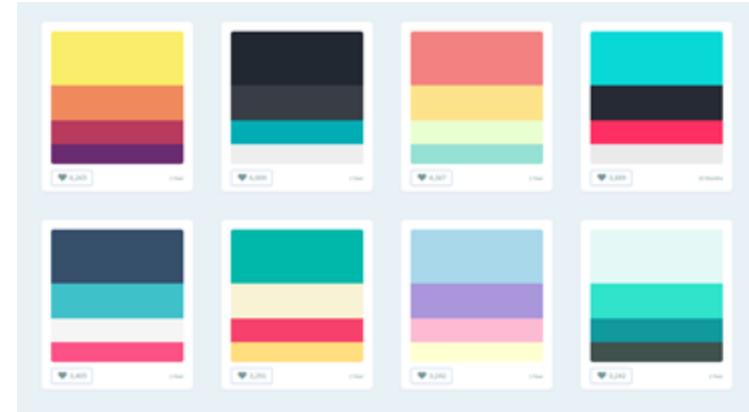
UI Design (User Interface)

O designer de interface do usuário é o profissional responsável em criar o que o usuário irá ver e utilizar o produto. Este profissional entende padrões visuais que podem ajudar o usuário na experiência de utilização do software.

Profissional focado em cores, tipografia, microinterações e estilos.

UI Design (User Interface)

UI designer, ou designer de interface do usuário, promove a criação e o desenvolvimento da interface explorada pelo usuário em um produto ou serviço.



COMPORTAMENTO

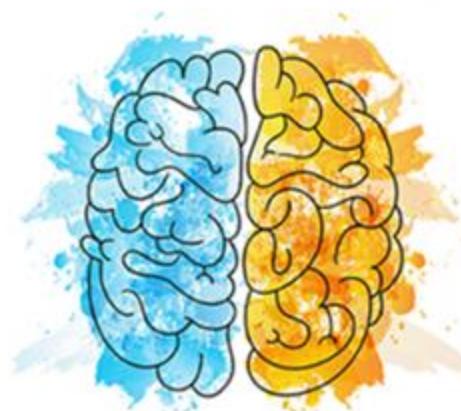
UX

PESQUISA

PROTÓTIPOS

PERSONAS

OBJETIVOS



APARÊNCIA

UI

DESIGN

TIPOGRAFIA

CORES

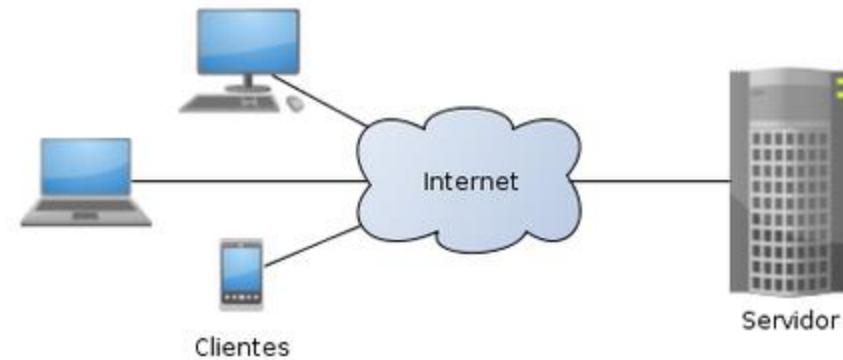
LAYOUTS

Ferramentas utilizadas

1. Invision
2. Visio
3. Adobe XD
4. Figma

Modelo Cliente-Servidor

O modelo cliente-servidor é uma estrutura de aplicação que distribui as tarefas e cargas de trabalho entre os fornecedores de um recurso ou serviço, designados como servidores, e os requerentes dos serviços, designados como clientes.



Desenvolvedor Front End

O desenvolvedor Front End é que programa a parte visual de um site ou aplicativo, ou seja, aquilo que conseguimos interagir. Quem trabalha com Front End é responsável por desenvolver por meio de código uma interface gráfica, normalmente com as tecnologias base da Web (HTML, CSS e JavaScript).

Onde que eu crio os códigos?

Em um IDE!

Um ambiente de desenvolvimento integrado (IDE) é um software para criar aplicações que combina ferramentas comuns de desenvolvimento em uma única interface gráfica do usuário (GUI).

Mas o que são os frameworks?

Framework é, de forma básica, um facilitador.

Ele traz diversas soluções já pré-definidas, que descomplicam o trabalho dos profissionais no desenvolvimento de aplicativos e outros projetos digitais.

Afinal, a atuação de um programador pode ter muito de criatividade, mas também traz aspectos mecânicos, de repetição de tarefas, que seriam maçantes sem a possibilidade de automatização.

Exemplos: Angular, Laravel e Vue

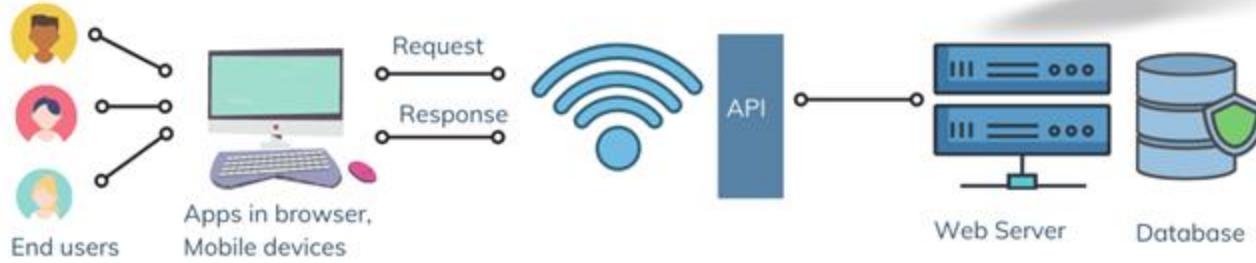
Desenvolvedor Back End

Back End, como o próprio nome sugere, vem da ideia do que tem por trás de uma aplicação. Pode ficar meio abstrato em um primeiro momento, mas pense que para conseguir usar o Facebook no dia a dia, os dados do seu perfil, amigos e publicações precisam estar salvos em algum lugar, sendo esse lugar um banco de dados e processados a partir de lá.

O Back End trabalha em boa parte dos casos fazendo a ponte entre os dados que vem do navegador rumo ao banco de dados e vice-versa, sempre aplicando as devidas regras de negócio, validações e garantias em um ambiente onde o usuário final não tenha acesso e possa manipular algo.
Exemplo: JAVA, PHP e C#

O que é uma API?

Acrônimo de Application Programming Interface (interface de programação de aplicativos), um intermediário de software que permite que dois aplicativos conversem entre si. Cada vez que você usa um app como o Facebook, envia uma mensagem instantânea ou verifica a previsão do tempo em seu telefone, você está usando uma API.



Desenvolvedor Full Stack

Um Desenvolvedor Full Stack é alguém que trabalha com o Back End do aplicativo, bem como o Front End. Desenvolvedores Full Stack precisam ter algumas habilidades em uma ampla variedade de linguagens de programação.

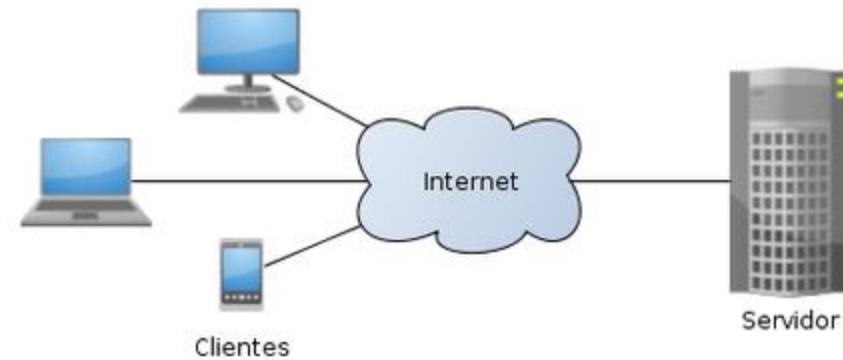
Quality Assurance (QA)

QA pode ser definida como um conjunto de ações que as empresas realizam com o objetivo de entregar aos consumidores um produto ou serviço com alto nível de qualidade. No desenvolvimento de software, aplicar os métodos de QA geram confiança e segurança aos clientes, indicando que os seus produtos terão a qualidade esperada na etapa de implantação.

O profissional de Quality Assurance deve ter conhecimento sobre as atividades do projeto, além de ter um perfil analítico. Ele verifica se os padrões de qualidade estão sendo atendidos e se todos os requisitos mínimos esperados no produto serão entregues.

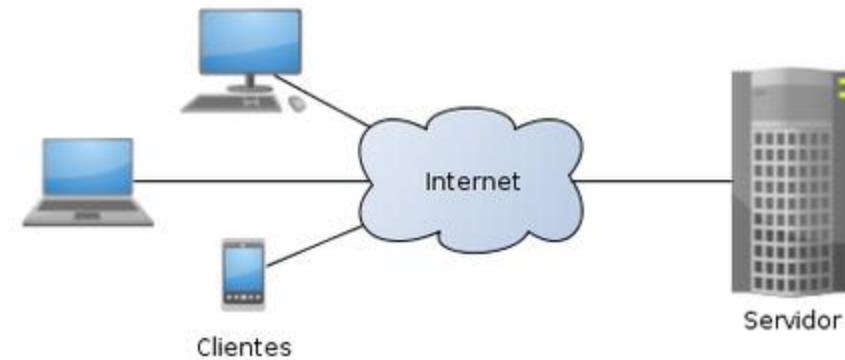
Modelo Cliente-Servidor

O modelo cliente-servidor é uma estrutura de aplicação que distribui as tarefas e cargas de trabalho entre os fornecedores de um recurso ou serviço, designados como servidores, e os requerentes dos serviços, designados como clientes.



Modelo Cliente-Servidor

O modelo cliente-servidor é uma estrutura de aplicação que distribui as tarefas e cargas de trabalho entre os fornecedores de um recurso ou serviço, designados como servidores, e os requerentes dos serviços, designados como clientes.





Atividades e profissionais em nuvem privada

- 1 – Segurança da Tecnologia da Informação (lógica e física)
- 2 – Mão de obra especializada (software e hardware)
- 3 - Infraestrutura local

Nuvem Pública

Os sistemas em nuvem são sistemas de armazenamento de dados disponibilizados via internet, em vez de servidores físicos tradicionais. Hoje, muitas organizações estão migrando o armazenamento de dados de servidores físicos para sistemas baseados em nuvem.



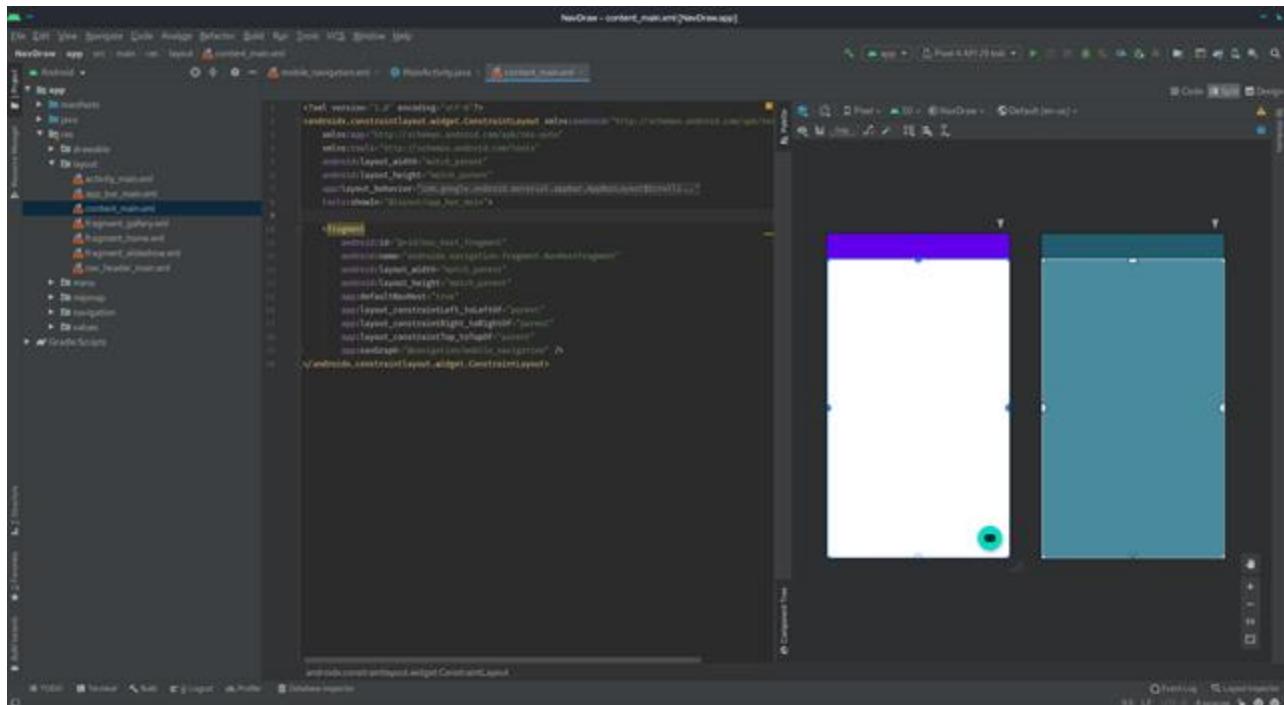
Vantagens de migrar para a nuvem pública

- 1 – Preço (pague somente o que usar)
- 2 - Facilidade de contratação, configuração e infraestrutura
- 3 – Escalabilidade
- 4 - Performance

Professional de Cloud Computing

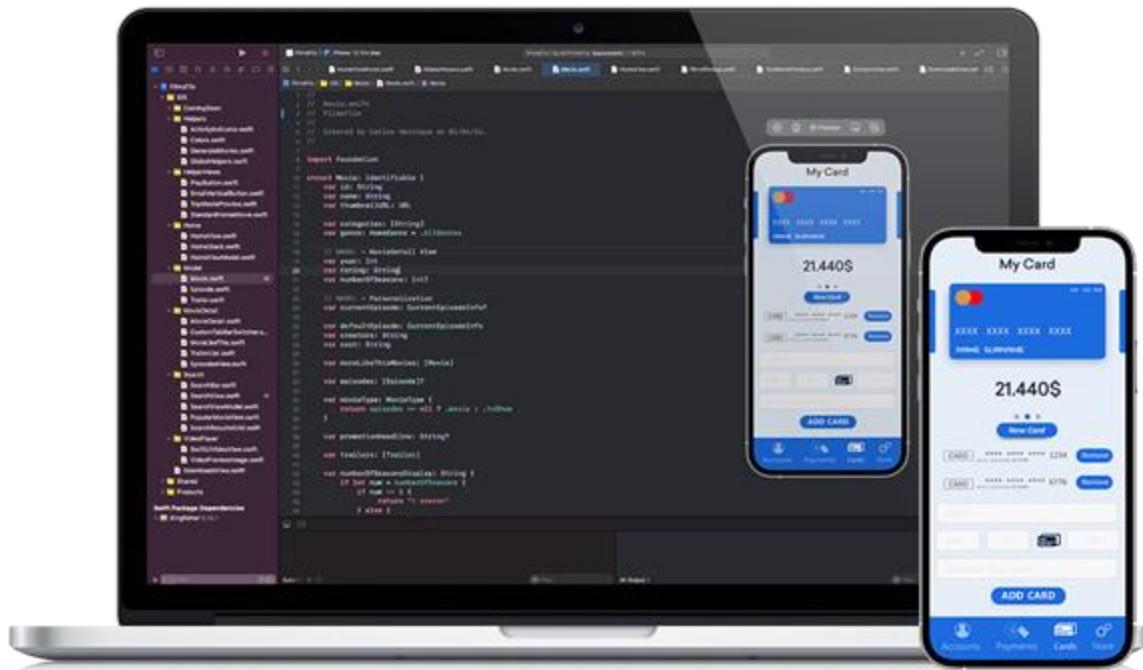
O profissional cloud computing é responsável pela infraestrutura de nuvem oferecida aos clientes. Mais do que desenhar sistemas ou ambientes de TI, ele escolhe as tecnologias que serão usadas, quais operadores são mais interessantes, como as peças vão ser integradas e, no fim, cuida do que foi construído.

Desenvolvimento mobile: Android



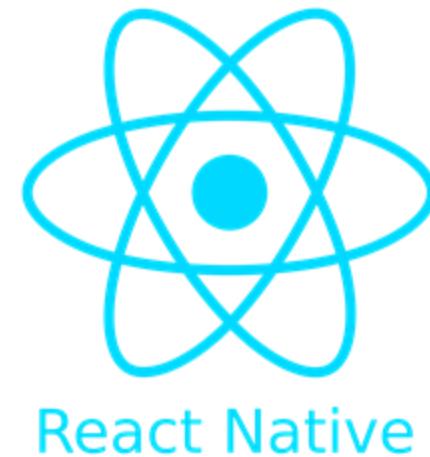
JAVA e/ou
KOTLIN

Desenvolvimento mobile: iOS



SWIFT

Desenvolvimento Híbrido (web)



Conclusão



Introdução ao mercado de qualidade de software

Carolina Santana Louzada
Engenheira de Qualidade de Software na UOLEdtech



Mais sobre mim

- Graduada em Engenharia de Computação- UFS
- Fazendo especialização em qualidade e desenvolvimento de software
- Qualidade de software -> automação
- Educação + tecnologia
- Jogos + música + aprender novas atividades
- LinkedIn -> [Carolina Santana Louzada | LinkedIn](#)

Objetivo do curso

Entender como a área de qualidade de software está inserida no mercado de TI, bem como compreender os perfis, responsabilidades e skills necessárias para se tornar um excelente profissional de qualidade de software.



Percorso

Aula 1

Mercado e tendências

Aula 2

Afinal, o que faz um QA?

Aula 3

Roadmap de aprendizagem para QAs

Dúvidas durante o curso?

- > Fórum do curso
- > Comunidade online (discord)

Aula 1: Qualidade de software: mercado e tendências

Introdução ao Mercado
de Qualidade de Software

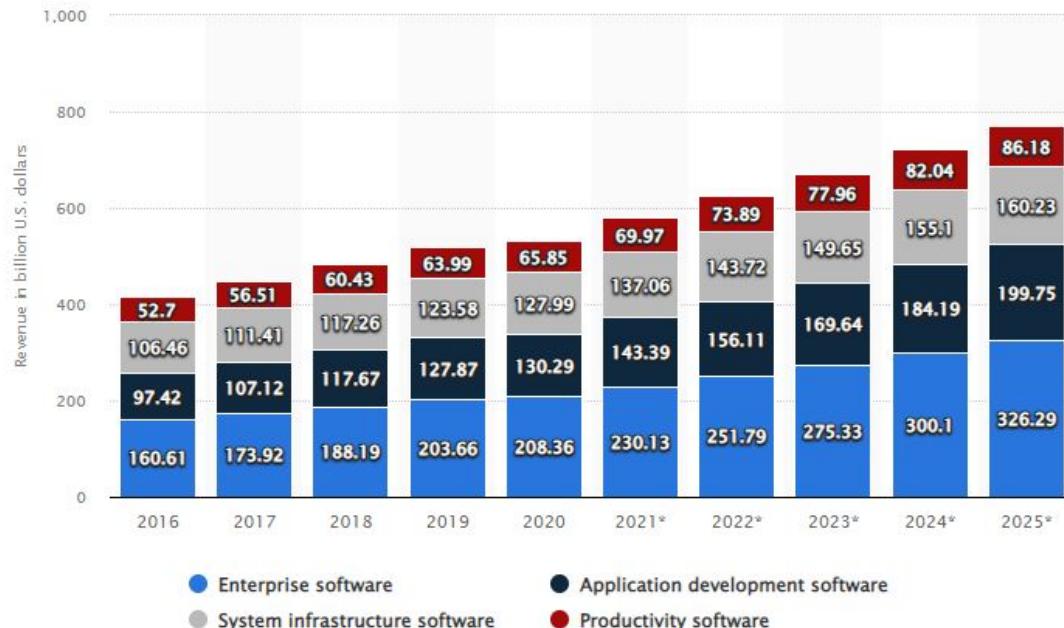


Objetivos

1. Contextualizar sobre o mercado de TI
2. Importância da qualidade de software
3. Compreender os desafios e futuro da área de qualidade de software



Receita do mercado de software a nível mundial



Fonte: Statista(2022)



Software,
Hardware
e Serviços

Investimentos em TI

Investimentos em TI por País	Valor(bilhões)
1. Estados Unidos da América	U\$914
2. China	U\$266
3. Japão	U\$145
4. Reino Unido	U\$117
5. Alemanha	U\$108
6. França	U\$76
7. Índia	U\$58
8. Canadá	U\$53
9. Brasil	U\$49.5
10. Austrália	U\$42

Fonte: Associação Brasileira de Empresas de Software



Investimentos em TI

US\$ 2.39
trilhões

Investimentos em TI a nível mundial no ano de 2020(mercado interno)

Distribuição de Investimentos no Brasil

Hardware

U\$
26.5bi



Software

U\$ 13
bi



Serviços

U\$ 10
bi



U\$
49.5
bi

53.7%

26.3%

20%



Crescimento de TI

Crescimento de TI no ano de 2020

Mundial	2.5%
Brasil	22.9%

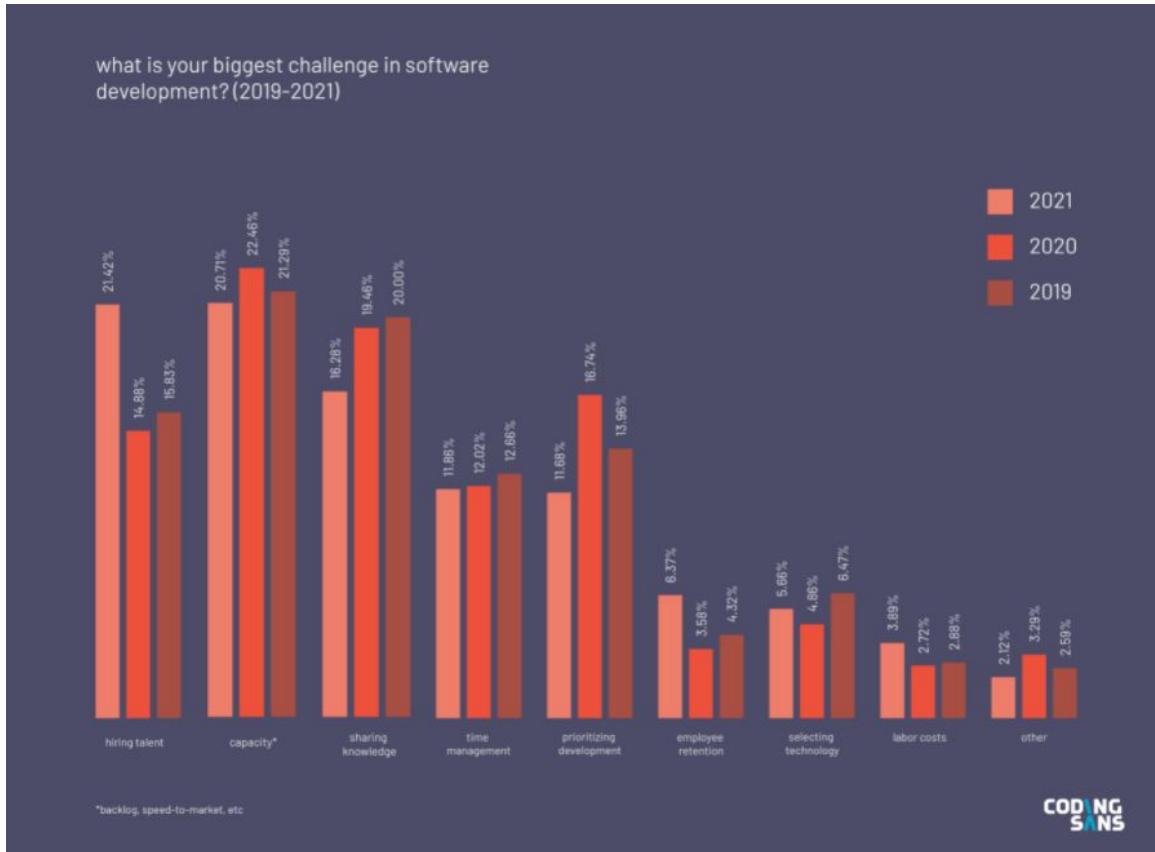
Crescimento esperado de TI no ano de 2021

Mundial	4.3%
Brasil	11.1%

[Dados do Setor | ABES \(abessoftware.com.br\)](http://abessoftware.com.br)



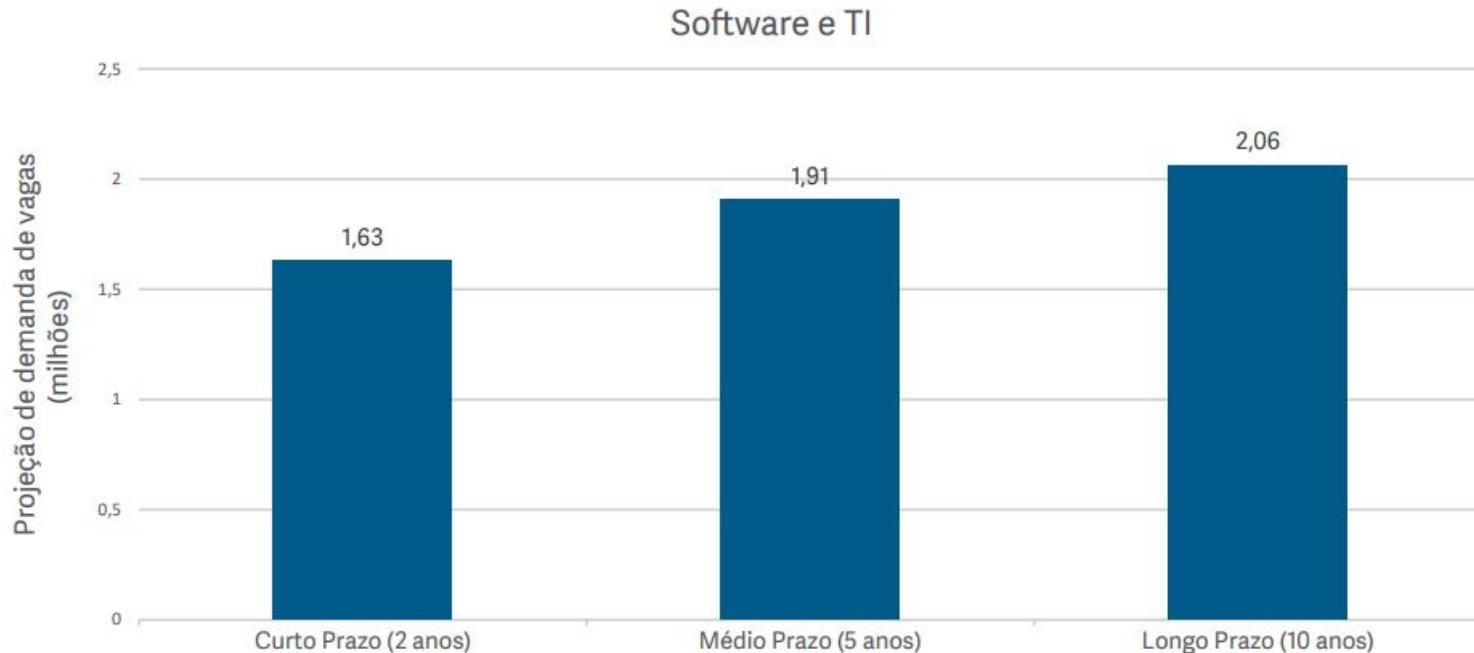
Maiores desafios em desenvolvimento de software





Previsão de empregos em TI

Profissões Emergentes na Era Digital



Fonte: Portal da Indústria(2021)



Profissões emergentes



Gestor de mídias
sociais



Engenheiro de
software



Especialista em
Blockchain



Programador
/Coder



Especialista em
inteligencia
artificial



Programador de
jogos digitais



Especialista em
Cloud



Cientista de
dados



Programador
multimídia



Analista de
cibersegurança



Engenheiro de
banco de dados

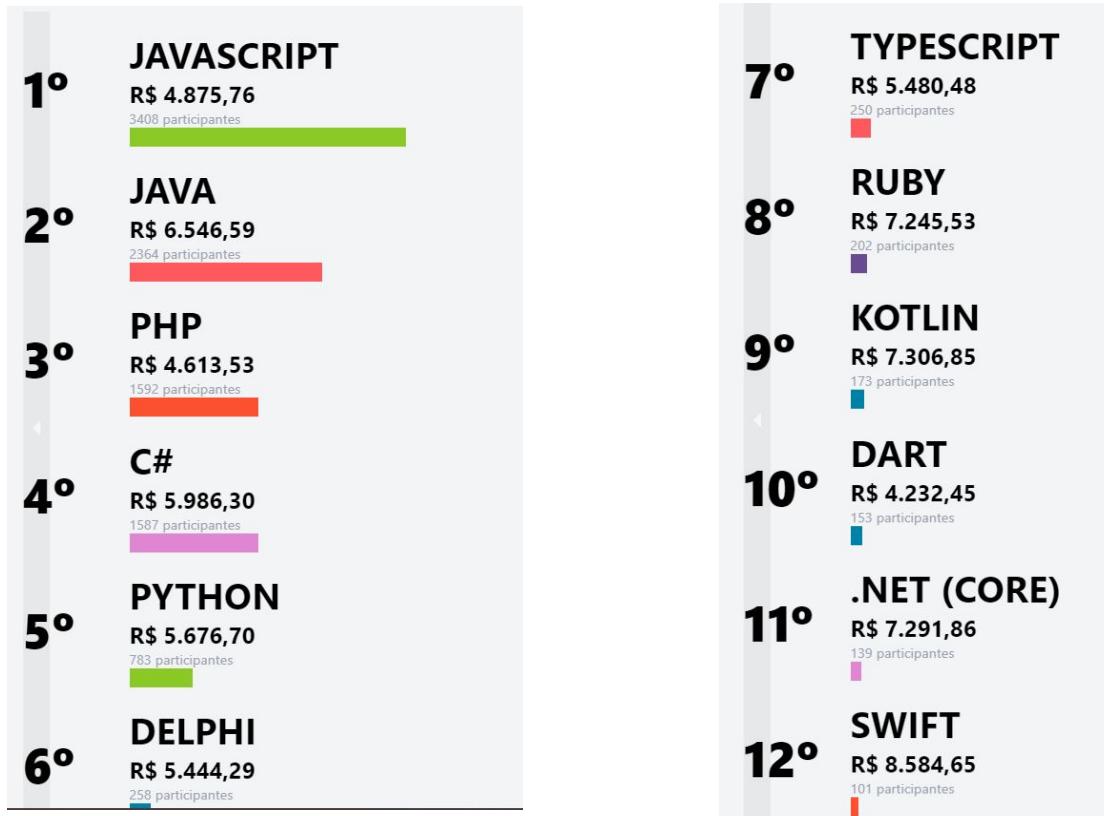


Desenvolvedor de
sistemas

Fonte: Portal da Industria(2021)



Sobre as linguagens mais utilizadas



Fonte: Pesquisa Código Fonte (2021)

Aula 1 | Etapa 2: Importância da qualidade de software

Introdução ao Mercado
de Qualidade de Software

A qualidade na história

- ★ Década de 60 -> desenvolvimento de softwares robustos, mas não confiáveis e de difícil manutenção
- ★ Adoção de métodos formais no gerenciamento de qualidade baseados em métodos usados na indústria de manufatura

Preocupações da qualidade

- ★ Gerenciamento de qualidade:
 - Nível organizacional : processos organizacionais e padrões
 - Nível de projeto:
 - ◆ plano de qualidade
 - ◆ aplicação de processos específicos de qualidade
 - Gerenciamento qualidade != burocratização



Atributos de qualidade de software

Segurança	Compreensibilidade	Portabilidade
Proteção	Testabilidade	Usabilidade
Confiabilidade	Adaptabilidade	Reusabilidade
Resiliência	Modularidade	Eficiência
Robustez	Complexidade	Capacidade de aprendizado

Fonte: Sommerville, Ian. Engenharia de Software. 9. ed. São Paulo: Pearson Prentice Hall, 2011.



Quanto vale testes com qualidade?

Estágio	Equipe sem testes	Equipe com testes
Implementação	7 dias	14 dias
Integração	7 dias	2 dias
Testes e correções	12 dias	9 dias
Tempo de lançamento da feature	26 dias	24 dias
Bugs encontrados em produção	71	11

Fonte: A Arte dos Testes Unitários - 2ª ed

Aula 1 | Etapa 3: O presente e futuro da área de qualidade

Introdução ao Mercado
de Qualidade de Software

O futuro (ou presente) para qualidade

- ★ Experiência de usuário
 - usuários mais exigentes
 - performance
 - acessibilidade
 - segurança
 - usabilidade
 - maior alcance populacional

- ★ Pandemia -> Aceleração do processo de transformação digital

O futuro (ou presente) para qualidade

★ Capacitação para novas tecnologias

- IA
- IoT
- Cloud
- Blockchain

★ Foco em segurança

★ Uso de metodologias ágeis e DevOps

Aula 2: Afinal, o que faz um QA?

Introdução ao Mercado
de Qualidade de Software



Objetivos

1. Engenharia de software e suas vertentes
2. Perfis e Responsabilidade de um QA
3. O papel das certificações na carreira de qualidade de software

A qualidade de software no mundo da engenharia

- ★ Engenharia de software
 - soluções viáveis
 - processos técnicos
 - processos gerenciais
- ★ Processo de software = especificação + desenvolvimento + validação + evolução

A qualidade de software
faz parte da engenharia
de software

Engenharia de software X QA

- ★ Engenharia de software != codificação
- ★ Tipos básicos de engenheiros de software:
 - **Front-End** : parte visual da aplicação e interação com usuário
 - **Back-End**: processamento de dados, regras de negócio
 - **Quality Assurance**: validações e verificações de funcionalidade, gestão de defeitos e processos de qualidade
 - **Devops/SRE**(*Site reliability engineering*): cultura e processos de operações para garantir confiabilidade, monitoramento, desempenho e pipelines de desenvolvimento

Aula 2 | Etapa 2: Perfis e responsabilidades de um QA

Introdução ao Mercado
de Qualidade de Software

Objetivos de QA

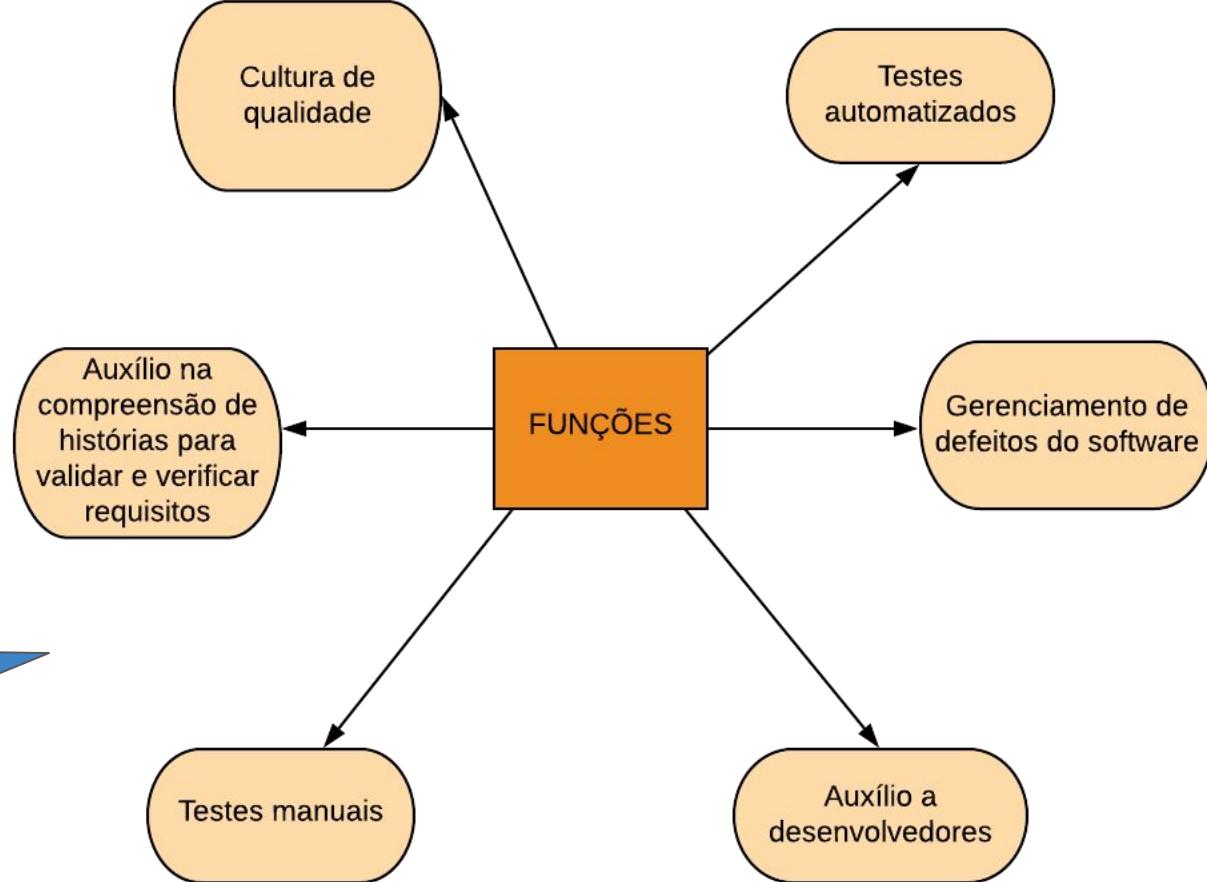
- ◆ Redução de custos e retrabalho
- ◆ Identificação de problemas
- ◆ Entrega de produtos com qualidade
- ◆ Melhora na satisfação do cliente
- ◆ Melhora na estimativa dos projetos
- ◆ Otimização da rotina de trabalho

Papel X Função x Cargo

- **Função:** Time ou grupo de pessoas e ferramentas para realizar um ou mais processos/atividades
- **Papel:** conjunto de responsabilidades, atividades e autoridades definidas em um processo de forma mais específica
- **Cargo:** responsabilidade que a pessoa assume em relação ao processo da empresa



Práticas para
gerenciamentos
de serviços de TI



**QA é mais
que testes!**



Cargos X responsabilidades

- Para um mesmo cargo podemos ter perfis e responsabilidades diferentes:
 - ◆ Gerenciamento
 - ◆ Análise
 - ◆ Testes manuais
 - ◆ Testes automatizados
 - UI/Interface
 - APIs
 - Performance/Desempenho

Aula 2 | Etapa 3: O papel das certificações na carreira como QA

Introdução ao Mercado
de Qualidade de Software



DIGITAL
INNOVATION
ONE

Certificações e sua importância na construção da carreira



I • B • Q • T • S



Certificações para área de qualidade de software - ISTQB



- Esquema de certificações internacionais para desenvolvimento da carreira de quem trabalha com testes de software
- Começou no ano de 1998 com o lançamento do Certified Tester Syllabus pela ISEB(*Information Systems Examinations Board*)

ISTQB

Certificações para área de qualidade de software - IBQTS



IBQTS

- Instituto Brasileiro de Qualidade em Testes de Software
- Certificações reconhecidas internacionalmente para área de engenharia de requisitos e engenharia de testes
- Fundado em 2006
- Reconhecido oficialmente pelo IREB (International Requirements Engineering Board,

Por que tirar certificações?



- Validação internacional de skills em testes de software
- Criação e melhoria nas skills para progressão de carreira
- Credibilidade profissional



I • B • Q • T • S
IREB Licensed Certification Body
www.ibqts.com.br

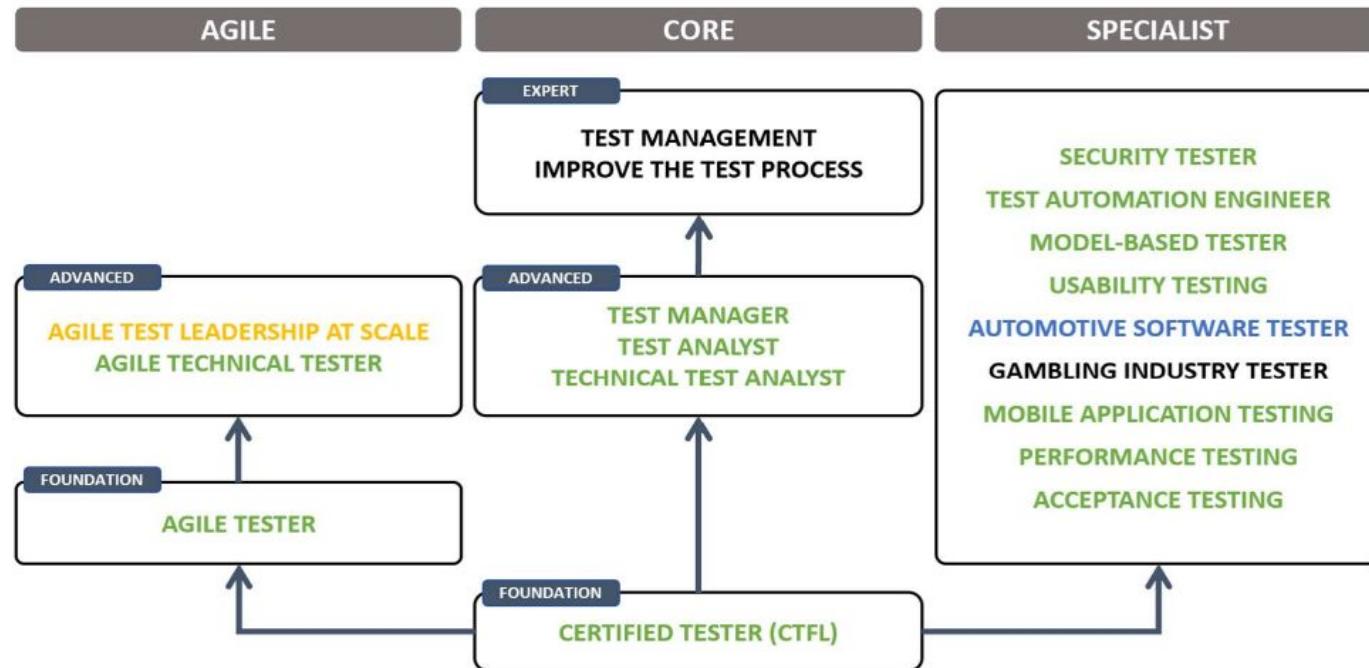


Portfólio de certificações da ISTQB

- Separação por *levels (níveis)*:
 - ◆ *Foundation*
 - ◆ *Advanced*
 - ◆ *Expert*
- Agrupamento de certificações = ***Streams(fluxos)***
 - ◆ *Core*
 - ◆ *Agile*
 - ◆ *Specialist*



Arquitetura do Portfólio ISTQB



Arquitetura do Portfólio ISTQB

★ Core

- Cobertura ampla nos conceitos de testes de software
- Válidos para qualquer domínio de tecnologia, metodologia ou aplicativo
- Entendimento comum

★ Agile

- Foco em práticas de testes dentro de contextos ágeis

Arquitetura do Portfólio ISTQB

★ Specialist

- Abordagem vertical de conhecimento
- Podem abordar características específicas de qualidade (usabilidade, desempenho, segurança...)
- Podem abordar práticas para tecnologias específicas
- Atividades de testes específicas
- Agrupamento de conhecimentos para domínios de aplicativos

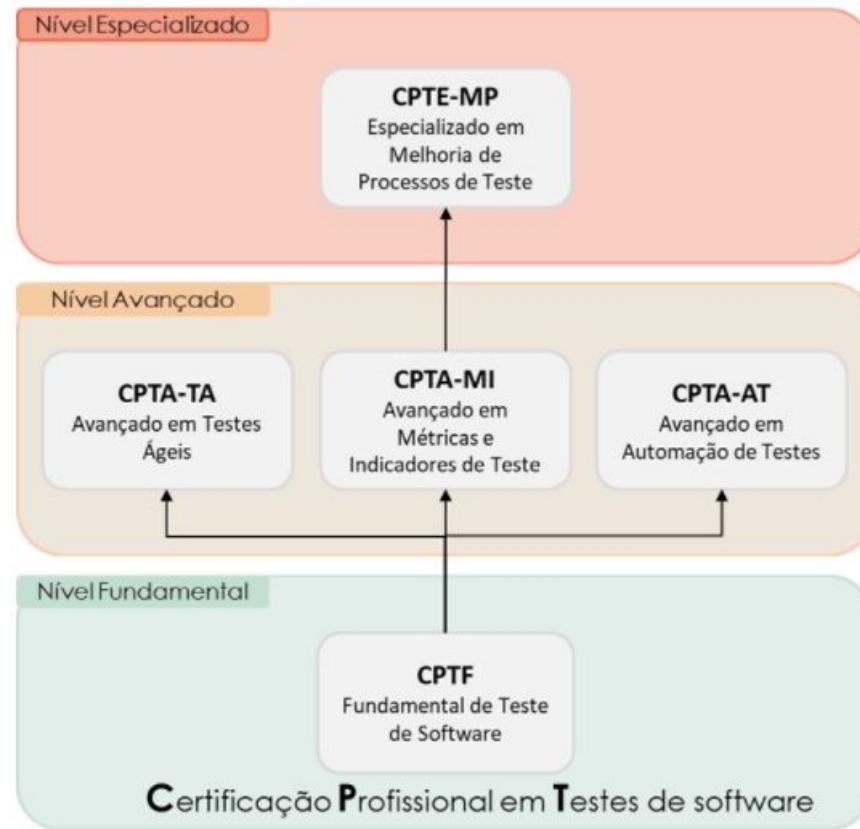
Conhecendo a base

1. CTFL (Certified Tester Foundation Level)

- Base das certificações
- Conhecimento prático de conceitos fundamentais de teste de software
- [Syllabus 3.1](#)



Arquitetura do Portfólio IBQTS - Engenharia de testes



Construindo caminho com outras certificações



- Conceitos sobre nuvem
- Descrição de serviços
- Ferramentas de gerenciamento e soluções
- Descrição de custos, SLA, segurança, privacidade...



[AWS Certification - Valide suas habilidades na nuvem - Seja certificado pela AWS \(amazon.com\)](#)

[Certificações da Microsoft | Microsoft Docs](#)

[Comparação entre as certificações em qualidade de software | by Carla Crude | Training Center | Medium](#)

Aula 3: Roadmap de aprendizagem para qualidade de software

Introdução ao Mercado
de Qualidade de Software

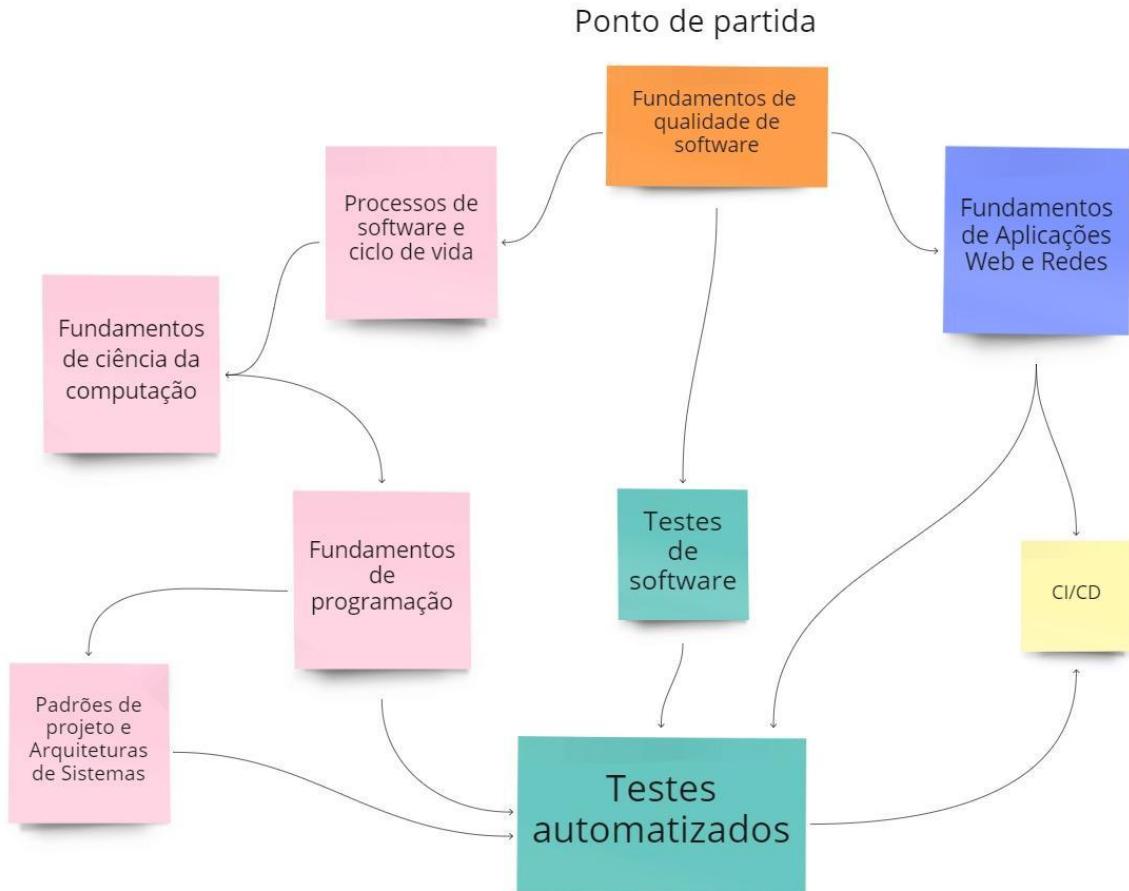


Objetivos

1. Compreender os conceitos e conhecimentos necessários para ser um QA completo
2. Refletir sobre o mindset de um QA e soft skills



Roadmap básico para QAs





Fundamentos de qualidade de software

- Definições de qualidade
- Priorização e técnicas de testes
- Plano de testes e documentação
- Gerenciamento de casos de testes
- Taxonomia de testes
- Gerenciamento de defeitos
- Métricas/Relatórios



Processos de software e ciclo de vida

- Modelos de processo de software
- Desenvolvimento ágil de software
- Testes dentro do modelo ágil



Fundamentos de aplicações Web e Redes

- Fundamentos de redes
- Arquitetura da internet e protocolos importantes
- Funcionamento de webpages
- Linguagens e tecnologias



Fundamentos de ciência da computação

- Representações e estrutura de dados
- Compilação x Interpretação
- Concorrência e threading
- Conceitos de sistemas operacionais
- Algoritmos e complexidade



Fundamentos de programação

- Uso de linha de comando
- Editores e IDEs
- Sintaxe e fluxo de controle
- Paradigmas da programação



Padrões de projeto e arquitetura de sistemas

- Conceitos e tipos de padrões
- Tipos de Arquitetura e
- Modelagem de sistemas



Testes automatizados

- Pirâmide de testes
- Automação como investimento
- Tipos de testes automatizados
- Frameworks para automação
- Objetos falsos e seus tipos
- BDD e linguagem Gherkin

CI/CD

- Estratégias de versionamento e tecnologias
- Deploys, release e orquestração
- Configuração e builds
- Uso de containers
- Testes integrados à pipeline
- Device farms e execução remota

Aula 3 | Etapa 2: **Soft skills e mindset de um QA**

Introdução ao Mercado
de Qualidade de Software



Erros que QAs podem cometer

- Falhas na análise de uma ocorrência
- Medo de fazer perguntas
- Automações falhas e sem padrões
- Esquecer do usuário
- Culpar outros por defeitos/bugs
- Não ter a visão do que ocorre em produção
- Não se importar com processos técnicos do desenvolvimento



Pensando nas características do software

1. Funcionalidade

As funcionalidades
são apropriadas?
Foram
implementadas
corretamente?

Como estão sendo
guardados os
dados? O sistema é
responsivo?



Pensando nas características do software

2. Confiabilidade

Como o software se comporta mediante condições específicas de falha?

Quão frequente falha? Qual tempo de recuperação?



Pensando nas características do software

3. Usabilidade

Os usuários
entendem o
software?

Qual esforço para
essa compreensão?



Pensando nas características do software

4. Eficiência

O time de desenvolvimento segue boas práticas?

A Arquitetura do projeto foi pensada para ser eficiente?



Pensando nas características do software

5. Manutenibilidade

Quão difícil é encontrar um problema e corrigi-lo?

Qual o esforço para modificar o código?



Pensando nas características do software

6. Portabilidade

O sistema se adapta
a mudanças no
ambiente?

Quão difícil é migrar
um componente do
sistema?



Para saber mais

[ISO_9126_NBR_13596_ANALISE_.pdf \(lcvdata.com\)](#)

[nbr-iso-9000-2005.pdf \(wordpress.com\)](#)

[Qualidade, Qualidade de Software e Garantia da Qualidade de Software são as mesmas coisas? \(linhadecodigo.com.br\)](#)

[Software Development Trends 2021: The Latest Research Data \(codingsans.com\)](#)

[Dados do Setor | ABES \(abessoftware.com.br\)](#)

[Software Developer Shortage in the World | Ncube](#)

[Pesquisa Salarial de Programadores 2020-2021 - Código Fonte TV \(codigofonte.com.br\)](#)

[4 grandes tendências de TI e os desafios para a área de QA \(onedaytesting.com.br\)](#)

[Everything you should know about QA in software development: The beginner's guide | by Concise Software | Medium](#)

[estudo_profissoes_emergentes - giz_ufrgs_e_senai.pdf \(portaldaindustria.com.br\)](#)



Para saber mais

[Quality Assurance \(QA\) e sua importância no desenvolvimento de software | Blog TreinaWeb](#)

[A importância da qualidade de software na vida das pessoas - WarmUP \(warmupweb.com.br\)](#)

Dúvidas?

- > Fórum do curso
- > Comunidade online (discord)

Orientações técnicas Vídeo

- **Orientações técnicas Vídeo:** OBS Studio.
- **Resolução:** 1280x720 • Ideal: 1920x1080.
- **Formato de gravação:** AVI ou MP4.
- **Qualidade de gravação:** O padrão do OBS Studio é “A mesma da transmissão”, mude para “Qualidade alta, arquivo normal”.
- Aumentar o tamanho da fonte de terminal/IDE.
- **Microfone:** Teste isolamento acústico (algum ruído é tolerável) e volume da voz entre 5-15 dB, se possível.

Fundamentos de Qualidade de Software

Carolina Santana Louzada

Engenheira de Qualidade de Software - UOLEdtech

Mais sobre mim

- Graduada em Engenharia de Computação- UFS
- Fazendo especialização em qualidade e desenvolvimento de software
- Qualidade de software -> automação
- Educação + tecnologia
- Jogos + música + aprender novas atividades
- LinkedIn -> [Carolina Santana Louzada | LinkedIn](#)

Objetivo do curso

Compreender fundamentos e normas fundamentais da área de qualidade, assim como aprofundar atividades de um analista ou engenheiro de qualidade software no mercado de trabalho . Introduzir níveis e tipos de teste e como estes se inserem no contexto da garantia qualidade.

Pré-requisitos

- Dedicação e vontade de aprender
- Um pouquinho de paciência
- Mente aberta

Percurso

Aula 1

O que é qualidade de software?

Aula 2

Gerenciamento de defeitos

Aula 3

Introdução aos testes de software

Dúvidas durante o curso?

- > Fórum do curso
- > Comunidade online (Discord)



Aula 1

O que é qualidade de software?

// Fundamentos de Qualidade de Software

Objetivos

- Definindo qualidade
- Normas e padrões de qualidade
- Medindo a qualidade
- Processos de gerenciamento de qualidade
de software

Aula 1 . Etapa 1

Definindo qualidade

// Fundamentos de qualidade de software

Definições na literatura

- ★ NBR/ISO 9000:2005: grau no qual um conjunto de características inerentes satisfaz a requisitos

Definições na literatura

- ★ Peters(2002) : “A qualidade de software é avaliada em termos de atributos de alto nível chamado fatores, que são medidos em relação a atributos de baixo nível, chamados critérios.”
- ★ Sanders(1994): “ Um produto de software apresenta qualidade dependendo do grau de satisfação das necessidades dos clientes sob todos os aspectos do produto.”

Definições na literatura

- ★ Pressman: “Qualidade de software é a conformidade a requisitos funcionais e de desempenho que foram explicitamente declarados, a padrões de desenvolvimento claramente documentados, e a características implícitas que são esperadas de todo software desenvolvido por profissionais.”

Definições na literatura

- ★ [ISO/IEC 25010:2011](#) : “capacidade do produto de software de satisfazer necessidades declaradas e implícitas sob condições especificadas”
- ★ [IEEE Standard\(2014\)](#): “ o grau em que um produto de software atende aos requisitos estabelecidos; no entanto a qualidade depende do grau em que esses requisitos representam com precisão as necessidades, desejos e expectativas das partes interessadas ”

Definições na literatura

★ Aspectos importantes:

- requisitos de software são a base para medir qualidade
- padrões especificados definem conjunto de critérios de desenvolvimento
- existem requisitos implícitos que não são mencionados que afetam diretamente a qualidade

Percepções de qualidade

Visão transcendental	Qualidade é reconhecida através de experiência, mas sem uma definição ou metrificação.
Visão do usuário	É personalizado de acordo com a necessidade do usuário.
Visão de manufatura	Qualidade é relacionada com conformidade aos requerimentos
Visão de produto	Produto com boas propriedades internas metrificáveis terá boas qualidade externas.
Visão baseada em valor	Representa o ‘custo-benefício’ na visão do cliente.

Aula 1 . Etapa 2

As normas e padrões de qualidade

// Fundamentos de qualidade de software

O que são normas técnicas?

Documentos publicados por organizações profissionais que objetivam padronizar determinadas atividades, processos, produtos, etc...



Instituições importantes

- IEEE: “*Institute of Electrical and Electronics Engineers*”
- ISO: “*International Organization for Standardization*”
- IEC: “*International Electrotechnical Commission*”



As normas para qualidade de software

Família ISO 9000	
ISO 9000	Descreve fundamento de sistemas de gestão de qualidade e suas terminologias
ISO 9001	Especifica requisitos para sistema de gestão de qualidade
ISO 9004	Diretrizes que consideram eficácia e eficiência do sistema de gestão da qualidade.
ISO 9126	Modelo de qualidade de produto de software

As normas para qualidade de software

ISO/IEC 14598	Processo de avaliação de produtos de software na visão do desenvolvedor, adquirente e avaliador
ISO/IEC/IEEE 12207:2017 ISO/IEC/IEEE 15288:2015	Processos de ciclo de vida do software
ISO 19011	Diretrizes sobre auditoria de sistemas de gestão da qualidade de software
IEEE 1012:2016	Verificação e validação para sistemas, software e hardware

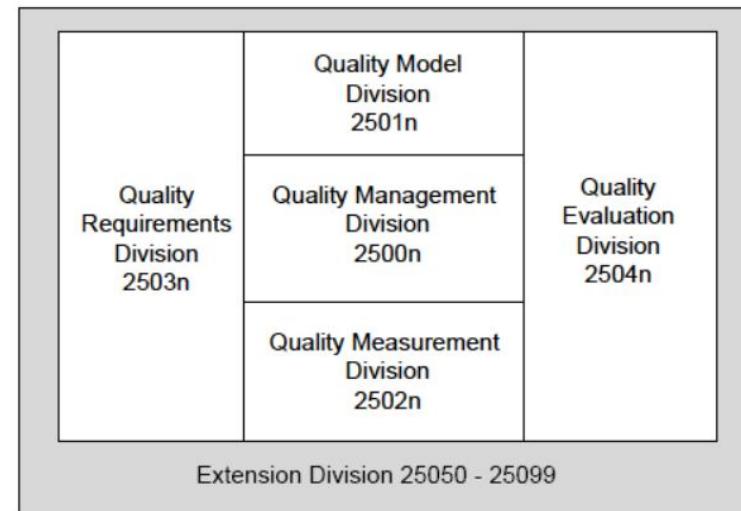
As normas para qualidade de software

IEEE 730:2014	Requerimentos para planejamento, controle e execução de processos de garantia de qualidade de software
ISO/IEC/IEEE 15289:2019	Foco no processo de gerenciamento de informação
ISO/IEC/IEEE 29119:2013	Vocabulário, processo, documentação, modelos e técnicas para teste

As normas para qualidade de software

Família SQuaRE: ISO/IEC 25000-25099 <i>(System and Software Quality Requirements and Evaluation)</i>	Substitui ISO/IEC 9126
---	------------------------

- Requerimentos de qualidade
- Modelo de Qualidade
- Gerenciamento de qualidade
- Metrificação de qualidade
- Avaliação de qualidade



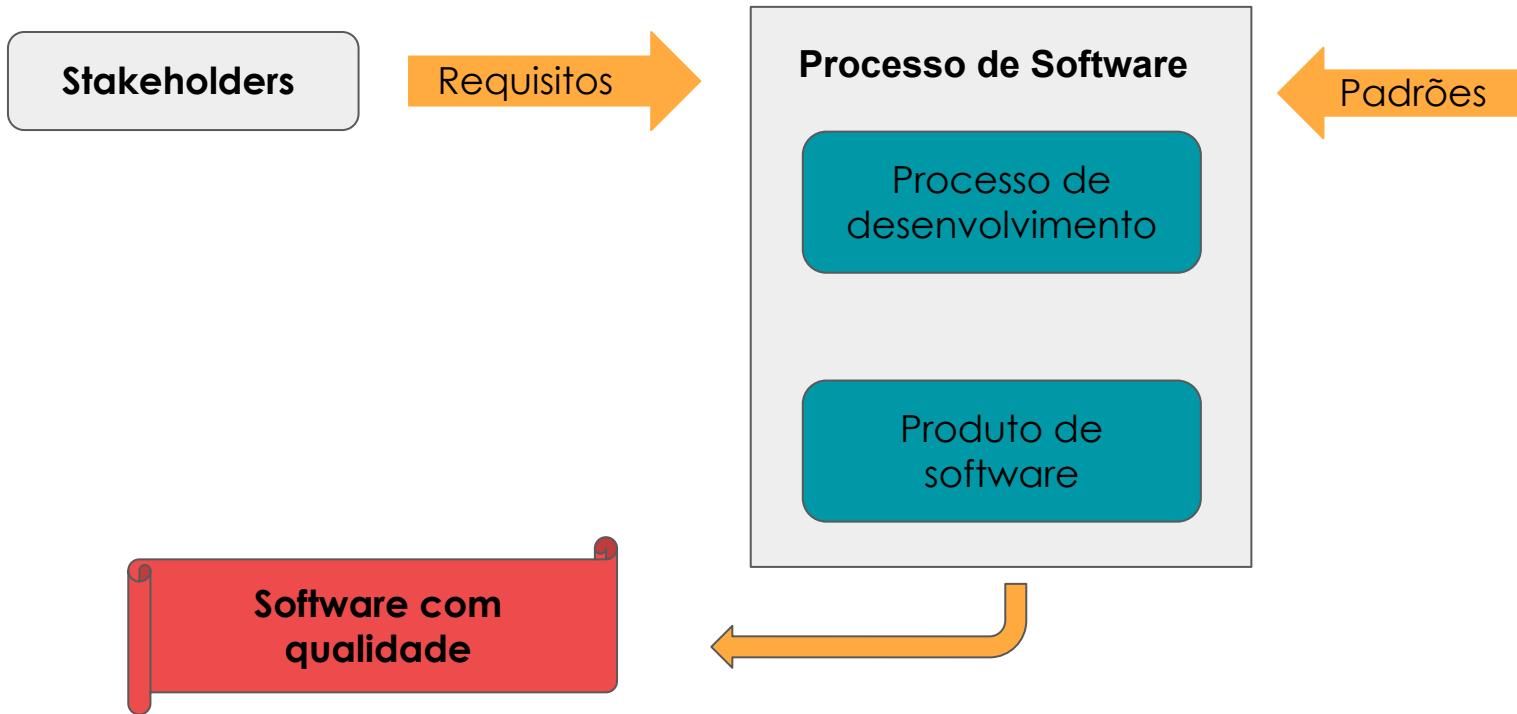
Fonte: ISO/IEC 25010

Aula 1 . Etapa 3

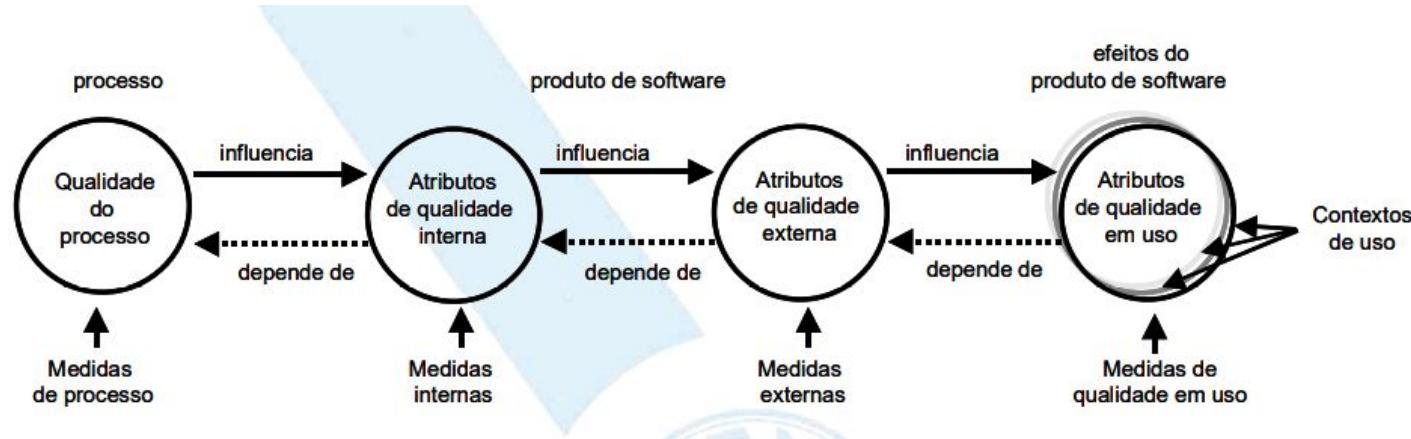
Medindo a qualidade

// Fundamentos de qualidade de software

Processo da qualidade

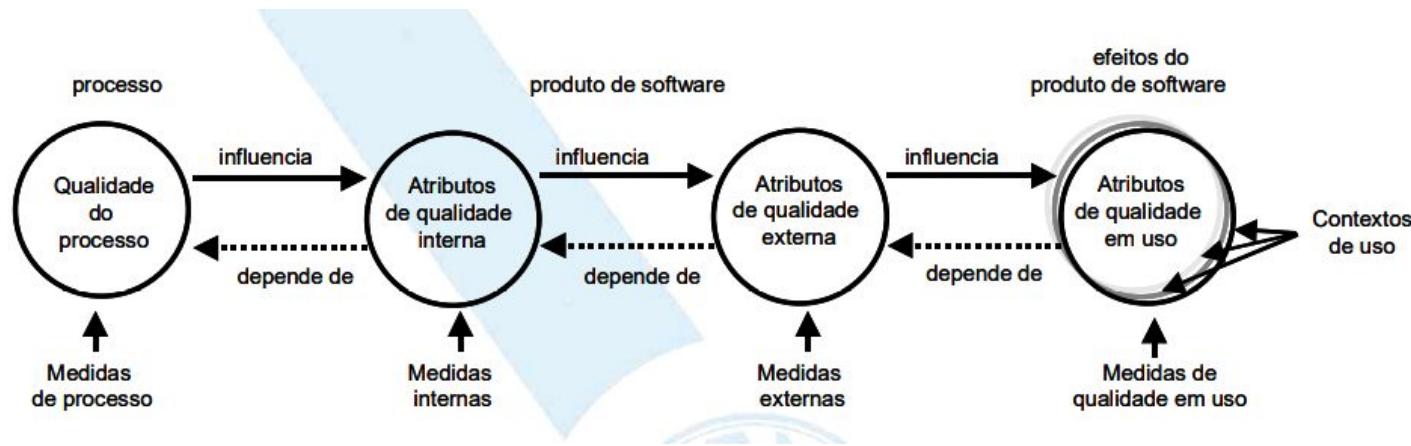


Medindo qualidade



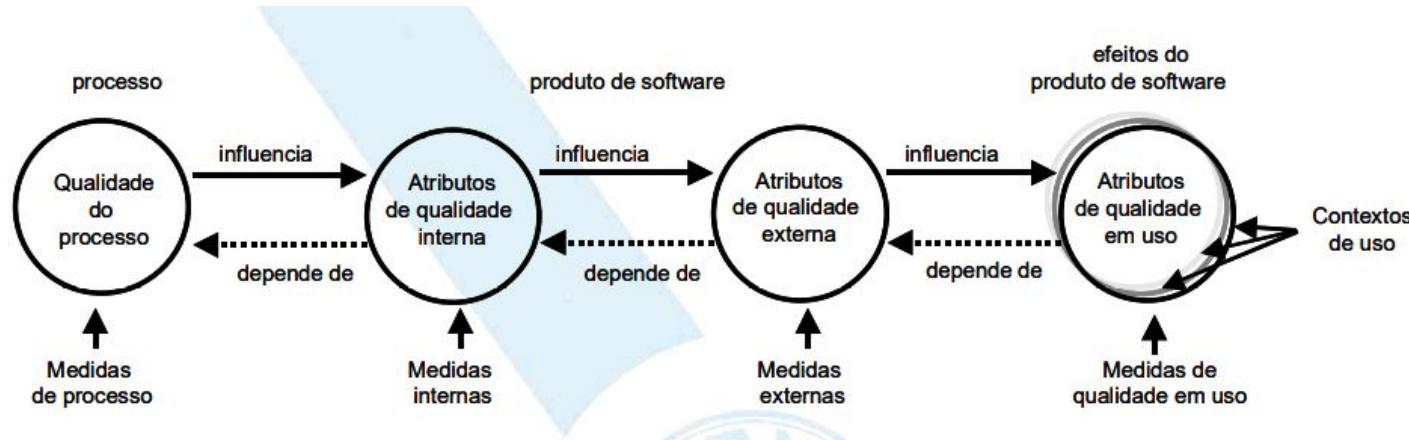
- Qualidade interna: totalidade das características do software do ponto de vista interno
- Métricas internas: podem ser aplicadas a um produto de software não executável, como especificações e código-fonte. Servem para avaliar a qualidade do produto antes do produto se tornar executável. São também indicadores de atributos externos.

Medindo qualidade



- Qualidade externa: totalidade das características do produto do ponto de vista externo, incluindo requisitos derivados das necessidades do usuário e dos requisitos de qualidade em uso
- Métricas externas: utilizam medidas de um produto de software derivadas de medidas do comportamento do sistema através de testes, operação e observação do produto

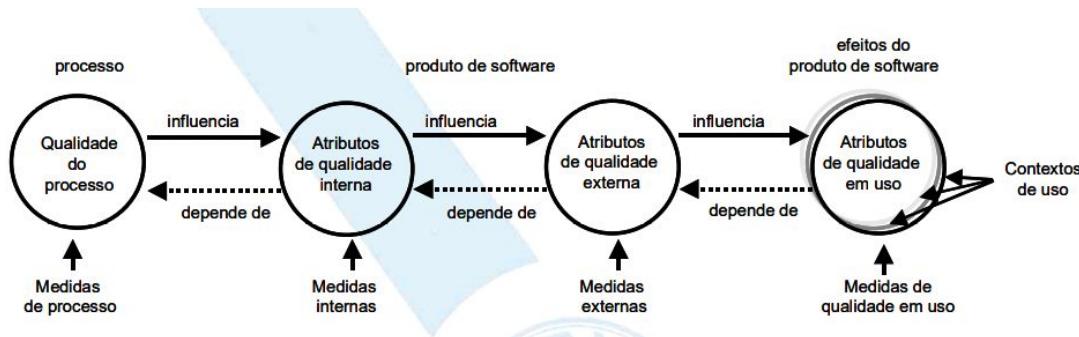
Medindo qualidade



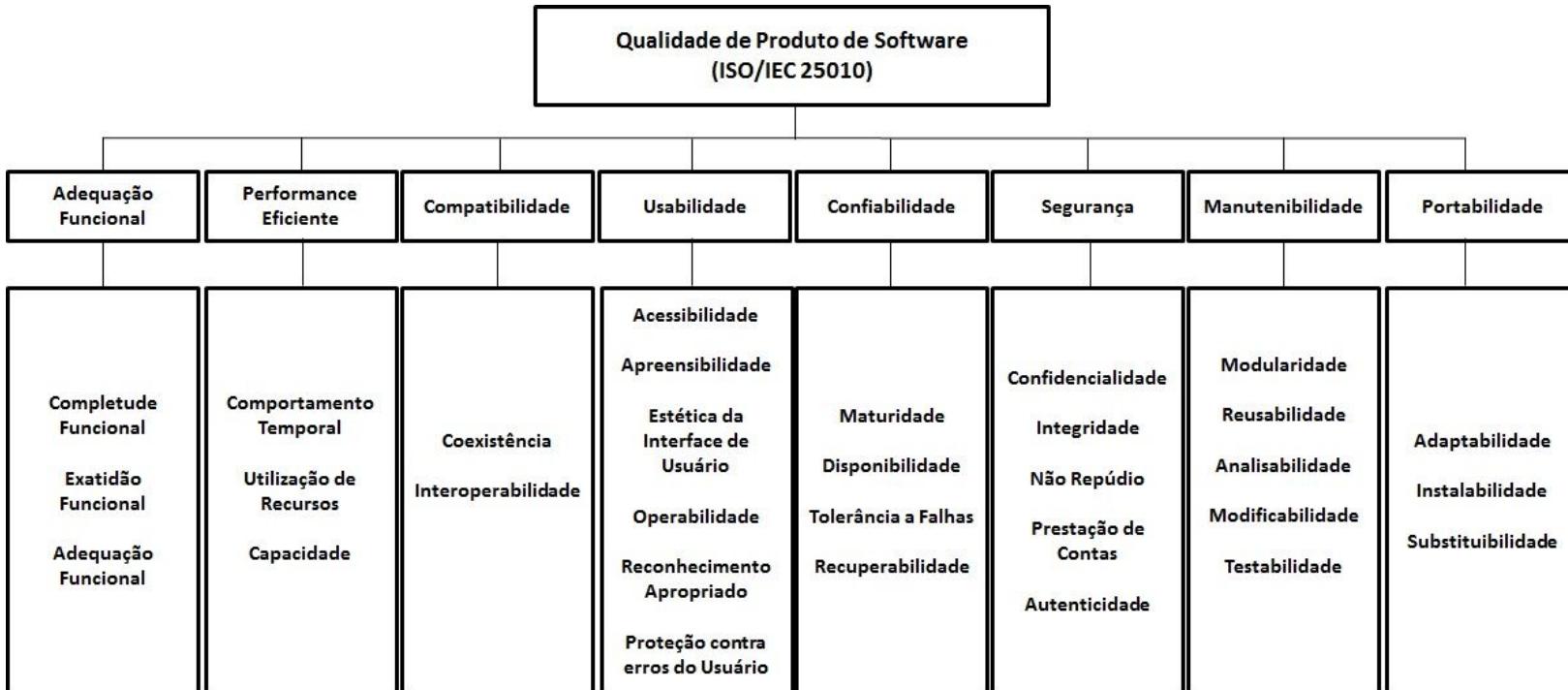
- Qualidade em uso: visão da qualidade do ponto de vista do usuário, em um ambiente e contexto de uso específicos
- Métricas de qualidade em uso: medem o quanto um produto atende às necessidades de usuário especificados

Olhando para ISO/IEC 25010

- Modelo de qualidade de produto de software: composto por 8 características subdivididas em sub-características
- Modelo de qualidade em uso: composto de 5 características e suas subcaracterísticas



ISO/IEC 25010 - Qualidade do produto



ISO/IEC 25010 - Qualidade em uso



Fonte: Kirner, G. Tereza(2021)

Aula 1 . Etapa 4

Processos de gerenciamento de qualidade de software

// Fundamentos de qualidade de software

Gerenciamento de qualidade de software

- ★ Conjunto de todos os processos que garantem que os produtos, serviços e o ciclo de vida vão de encontro com os objetivos da qualidade e forma a alcançar a **satisfação do usuário**.
- ★ Atividades do gerenciamento:
 - Planejamento de qualidade
 - Garantia de qualidade
 - Controle de qualidade
 - Melhoria de processos

Planejamento

Determinar:

- padrões e processos de qualidades a serem utilizados
- metas específicas de qualidade
- esforço e organização de atividades

Garantia de qualidade

- Atividades de que definem e avaliam a adequação dos processos de software de forma a prover evidências que estabelecem confiança no produto produzido.

Controle de qualidade

- Examinação de artefatos do projeto para determinar se os padrões acordados estão sendo seguidos
- Avaliação de produtos intermediários e do produto final

Melhorias de processos

- Se preocupa com melhorias na eficiência, efetividade e quaisquer características que tenham como meta principal a melhoria da qualidade de software

Outra perspectiva

1. Aplicação de métodos técnicos
2. Realização de revisões técnicas formais
3. Atividades de testes de software
4. Aplicação de padrões
5. Controle de mudanças
6. Medição
7. Manutenção de registros e relatórios

Aula 2

Gerenciamento de defeitos

// Fundamentos de Qualidade de Software

Objetivos

- Falando em controle de qualidade
- Caracterizando defeitos
- Ciclo de vida do bug
- Ferramentas de suporte

Aula 2 . Etapa 1

Falando em controle de qualidade

// Fundamentos de qualidade de software

Controle de qualidade

- Análise estática: avaliação de documentação do software e código-fonte -> métodos formais
- Análise dinâmica: relacionado a técnicas com o código em execução

Validação X Verificação

Verificação: Garantir que o produto está sendo construído corretamente

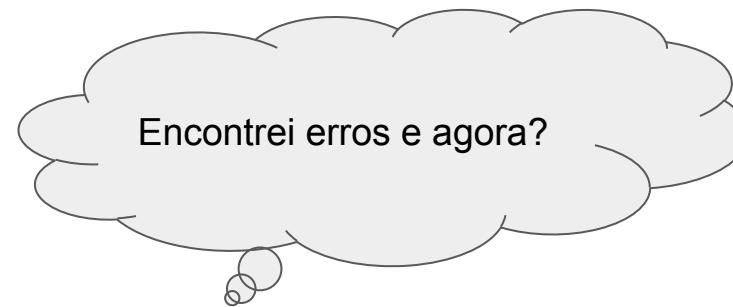
Validação: O produto correto está sendo construído.

Controle de qualidade

- Análise estática: avaliação de documentação do software e código-fonte -> métodos formais
 - ◆ Code review
 - ◆ Ferramentas de automação de processos de verificação de código
 - ◆ Análise de histórias e modelagens

Controle de qualidade

- Análise dinâmica: relacionado a técnicas com o código em execução
 - ◆ Finalmente nossos queridos testes!



Aula 2 . Etapa 2

Caracterizando defeitos

// Fundamentos de qualidade de software

Rastreamento de defeitos

- Entendimento do produto e dos tipos de defeitos encontrados
- Facilitar correção do processo ou do produto
- Reportar status do produto
- Alinhamento de revisões pelo time de desenvolvimento

O que é defeito?

- Genericamente significa qualquer tipo de anomalia encontrada no produto
- Outras definições:
 - ◆ Erro: Ação humana que produz um resultado incorreto
 - ◆ Defeito: Imperfeição ou deficiência relacionada aos requerimentos e especificações do produto que se
 - ◆ Falha de sistema: Evento no qual o sistema não executa uma função sob limites específicos

Importância de padronizar
definições na equipe

Motivos para erros

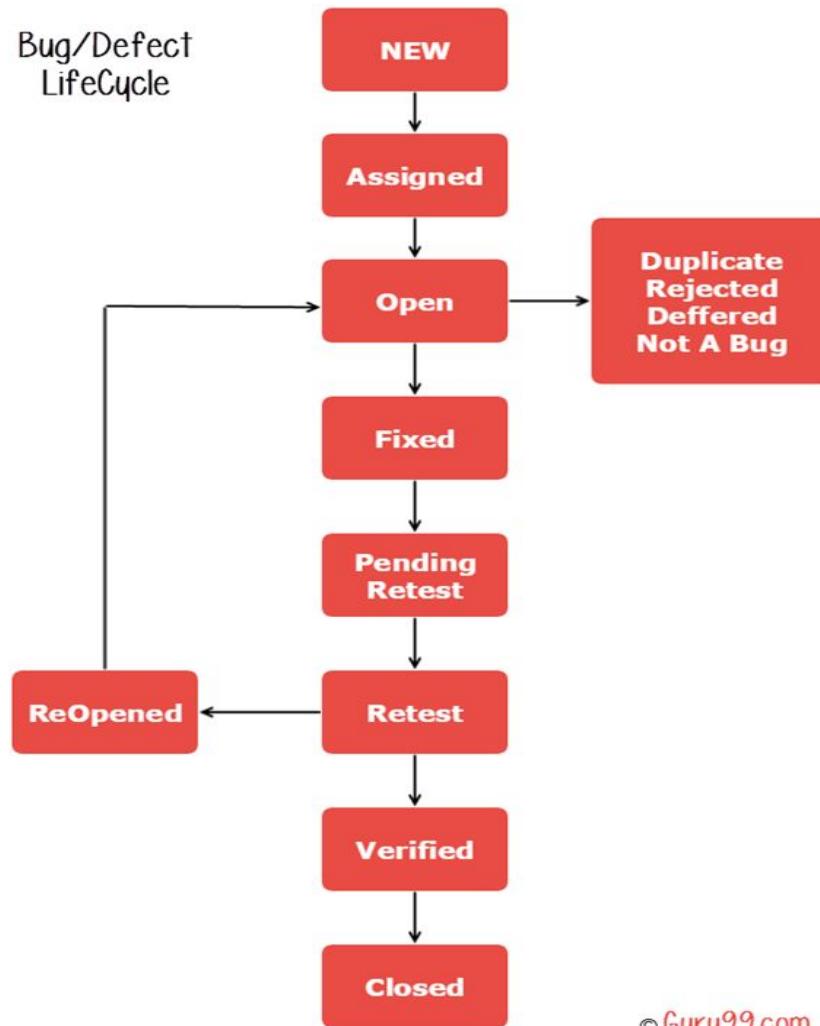
- Pressão do tempo
- Falha humana
- Inexperiência e/ou falta de qualificação
- Falta de comunicação
- Complexidade de código, modelagem, arquitetura...
- Complexidade de tecnologia
- Condições ambientes inesperadas

Aula 2 . Etapa 3

Ciclo de vida do bug: do rastreio ao reporte

// Fundamentos de qualidade de software

Bug/Defect
LifeCycle



Ciclo de vida

1. New: Defeito é identificado e cadastrado pela primeira vez
2. Assigned: defeito é atribuído para desenvolvedor avaliar
3. Open: desenvolvedor inicia análise e correção
4. Fixed: Desenvolvedor finaliza correção
5. Pending Retest: Estado de espera para o time de teste
6. Retest: Estado de execução do reteste
7. Verified: Defeito corrigido
8. Reopen: Defeito não-corrigido.
9. Closed: Corrigido + testado + aprovado
10. Duplicate: efeito já encontrado anteriormente
11. Rejected: Defeito não é novo.
12. Deferred: Será corrigido em versões futuras.
13. Not a bug: Quando a anomalia não é de fato um erro depois de analisado

Considerações importantes

- ★ Os processos se adequam ao que o seu time e seu produto precisam!
- ★ O time precisa estar de acordo e entender todo o fluxo de rastreamento de defeitos
- ★ Os defeitos podem e devem ser rastreados em qualquer momento do ciclo de vida do processo de software.
- ★ Principais objetivos dos reports de defeitos:
 - Provê às partes interessadas informações a respeito do evento anômalo de forma a tentar isolar, reproduzir e corrigir o problema ou o potencial problema.
 - Provê meios para rastrear a qualidade do produto e o impacto destes na atividades de testes e retestes
 - Provê ideias para melhoria no processo de desenvolvimento e testes

Considerações importantes

- ★ Boa comunicação é essencial!
- ★ Uso eficiente de ferramenta de rastreio e report de bugs
- ★ Comprometimento proativo da equipe no gerenciamento dos defeitos



Informações de um reporte de defeito

- ★ Um identificador único
- ★ Título resumindo o problema
- ★ Data/autor
- ★ Identificação do item sob teste e do ambiente
- ★ Fase do ciclo de vida no qual o defeito foi observado
- ★ Descrição completa do defeito para reprodução
- ★ Evidências de auxílio na resolução:
 - logs
 - dumps de banco de dados
 - screenshots
 - gravações
- ★ Resultado esperado
- ★ Severidade
- ★ Urgência/Prioridade
- ★ Estado do defeito
- ★ Conclusões/Sugestões
- ★ Impactos
- ★ Histórico
- ★ Referência do teste

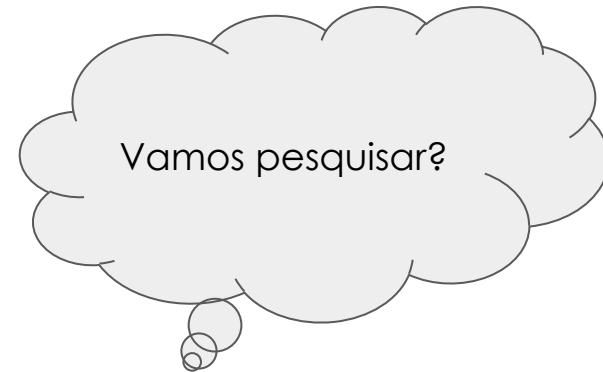
Aula 2 . Etapa 4

Ferramentas de suporte

// Fundamentos de qualidade de software

Algumas ferramentas úteis

- [Bugzilla](#) - gratuito
- [Jira](#) - gratuito e pago
- [Trac](#) - gratuito
- [Redmine](#) - gratuito
- [Asana](#) - gratuito e pago
- [Trello](#) - gratuito e pago
- [Backlog](#) - gratuito e pago
- [ReQtest](#) - pago
- [Mantis](#) - gratuito
- [Axosoft](#) - pago
- [Etraxis](#) - gratuito
- [Lighthouse](#) - pago
- [Azure Devops](#) - pago



Aula 3

Introdução aos testes de software

// Fundamentos de Qualidade de Software

Objetivos

- Conceitos e objetivos
- Processo de teste
- Níveis de teste
- Tipos de teste
- Técnicas de teste

Aula 3 . Etapa 1

Teste: conceitos e objetivos

// Fundamentos de qualidade de software

O que é teste?

- Processo de avaliar e reduzir risco de falhas de software em operação
- Faz parte do controle de qualidade
- O processo de teste não diz respeito somente ao ato de executar um teste



Objetivos gerais

- Evitar defeitos e avaliar produtos de trabalho
- Verificar cumprimento de requisitos
- Validar se produto funciona como cliente espera
- Criar confiança no nível de qualidade do objeto testado
- Redução de riscos
- Atuar junto ao cliente para tomada de decisões

Teste X depuração

- A execução de testes pode mostrar falhas causadas por defeitos de software
- Depuração já é um processo de investigação e correção do erro no processo do desenvolvimento do código
- Essas atividades variam de acordo com a metodologia utilizada na equipe

Princípios de teste

1. Teste mostra presença de defeitos e não a ausência
2. Testes exaustivos são impossíveis
3. Testes iniciais economizam tempo e dinheiro
4. Defeitos se agrupam
5. O mesmo teste não encontra novos defeitos -> atenção com testes de regressão
6. O teste depende do contexto
7. Ausência de erros é ilusão

Aula 3 . Etapa 2

O processo de teste

// Fundamentos de qualidade de software

Fatores de influência

- ★ Modelo de ciclo de vida
- ★ Níveis e tipos de teste
- ★ Risco de produto e projeto
- ★ Domínio do negócio
- ★ Restrições operacionais
- ★ Políticas e práticas organizacionais
- ★ Normas internas e externas

Atividades de teste

1. Planejamento
2. Monitoramento e controle do teste
3. Análise
4. Modelagem
5. Implementação
6. Execução
7. Conclusão

Planejamento do teste

- Definir propósitos do teste
- Definir a abordagem do teste de acordo com restrições do contexto
- Especificar tarefas e estimativas de prazos
- Algumas estratégias:
 - ◆ Analítica: baseada na análise de algum fator
 - ◆ Baseada em modelo: projetados com base em modelo de algum aspecto necessário do produto (modelo de processo de negócio, de estado, de requisitos não funcionais)
 - ◆ Metódica: Conjunto pré-definido de testes, comparando as características de qualidade importantes

Planejamento do teste

- Algumas estratégias:
 - ◆ Compatível com processo: baseado em padrões definidos pela organização
 - ◆ Dirigida: orientado pelos stakeholders
 - ◆ Regressão: evitar regressão de recursos existentes
 - ◆ Reativo: é reativo ao componente ou sistema e aos eventos que ocorrem durante a execução

Monitoramento e controle do teste

- Comparação contínua do progresso real com o plano de teste a partir de critérios de avaliação de saídas...ou seja, o '**done**'!
- Utilização de relatórios de progresso

Análise do teste

- Base de teste é analisada de forma a analisar “o que testar” de acordo com critérios pré-estabelecidos
 - ◆ especificações de requisitos
 - ◆ documentos de arquitetura, fluxograma, casos de uso, etc...
 - ◆ código-fonte
- Avaliar os tipos de defeitos que podem ser encontrados
- Definir e priorizar condições de teste

Modelagem do teste

- Responde a pergunta ‘como testar’?
- As condições de teste são elaboradas em casos de teste de alto nível
- Priorização de casos de teste e conjuntos de casos de teste
- Verificar infraestrutura necessária e projetar ambiente de teste

Implementação do teste

- Desenvolver e priorizar procedimentos de teste e possivelmente script automatizados
- Criar suítes de testes
- Organização lógica e eficiente da execução dos testes
- Preparar dados de teste

Execução do teste

- Conjuntos de testes são executados conforme planejado, seja de forma manual ou automatizada
- Comparar resultados reais com resultados esperados
- Analisar anomalias para estabelecer prováveis causas
- Reportar e registrar essas anomalias
- Reteste

Conclusão do teste

- Coletar dados das atividades de testes já concluídas de forma a revisar e consolidar a experiência
- Criar relatório de resumo de teste
- Finalizar e arquivar dados e registros dos testes
- Melhorar maturidade do processo de teste

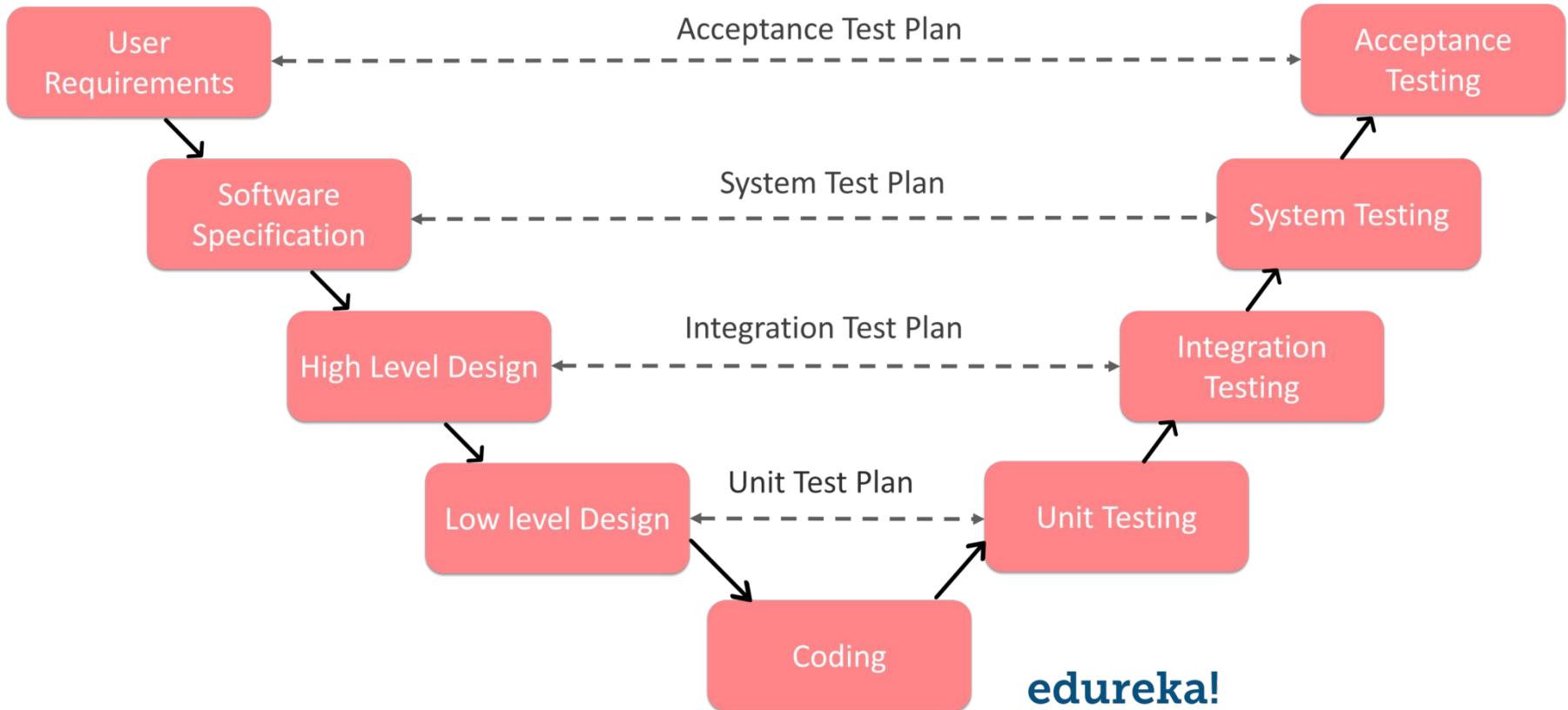
Aula 3 . Etapa 3

Níveis de teste

// Fundamentos de qualidade de software

O que seriam esses níveis?

- São grupos de atividades de teste que são organizados e gerenciados juntos com relação ao nível de desenvolvimento
 - ◆ Teste de componentes
 - ◆ Teste de integração
 - ◆ Teste de sistema
 - ◆ Teste de aceite



Testes de componente ou unidade

- Foco em testar componentes do código de forma independente
- Importante para:
 - ◆ Reduzir risco
 - ◆ Verificar requisitos funcionais e não-funcionais
 - ◆ Construir confiança do componente
 - ◆ Encontrar defeitos
 - ◆ Evitar que os defeitos sejam refletidos em níveis mais altos de teste

Testes de integração

- Foco na integração entre componentes ou comunicação de sistemas
- Importante para:
 - ◆ Reduzir risco
 - ◆ Verificar interfaces
 - ◆ Encontrar defeitos nas partes envolvidas e sejam refletidos em níveis mais altos de teste

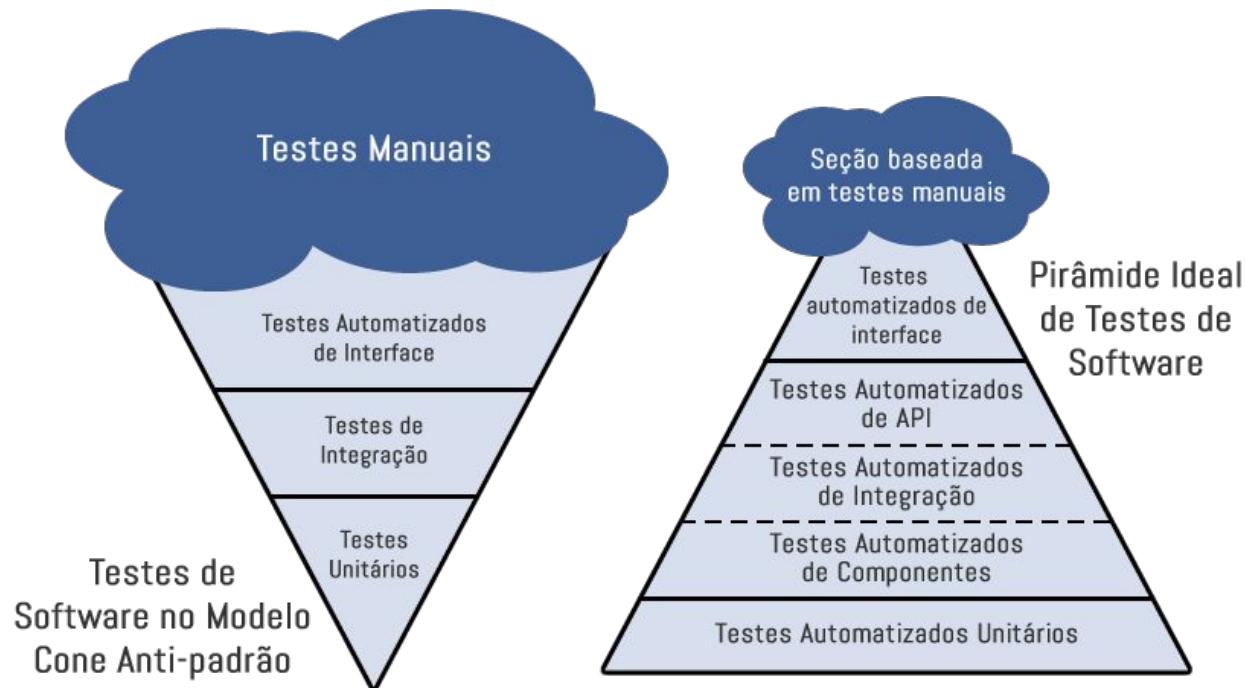
Testes de sistema

- Foco nos requisitos de ponta a ponta do sistema
- Importante para:
 - ◆ Reduzir risco
 - ◆ Validar sistema como um todo
 - ◆ Encontrar defeitos não vistos em níveis mais baixos
 - ◆ Evitar que defeitos se reflitam em produção após aceite do cliente

Testes de aceite

- Foco nos requisitos de ponta a ponta do sistema do ponto de vista validação e conformidade com regras de negócio e necessidades do cliente
- Importante para:
 - ◆ Reduzir risco
 - ◆ Validar sistema como um todo
 - ◆ Encontrar defeitos não vistos em níveis mais baixos
 - ◆ Evitar que defeitos se refletem em produção após aceite do cliente

Pirâmide de testes



Fonte: PrimeControl(2017)

Aula 3 . Etapa 4

Tipos de teste

// Fundamentos de qualidade de software

Tipos de teste e objetivos

- Grupo de atividades de teste destinado a verificar características específicas de um sistema, com base em objetivos específicos.
 - ◆ Avaliar características funcionais
 - ◆ Avaliar características não funcionais
 - ◆ Avaliar estrutura ou arquitetura de componente/sistema
 - ◆ Avaliar efeitos de alterações em outras partes do código

Teste funcional

- Avaliação de funções que o sistema deve executar
- Desenvolvidos a partir de especificações de requisitos, histórias de usuário, casos de uso
- Os testes funcionais podem ser realizados em todos os níveis de teste
- Técnicas caixa-preta são bem úteis para avaliação de comportamentos funcionais do sistema

Teste não funcional

- Avaliação de características não funcionais como usabilidade, eficiência de performance, segurança, etc...
- Também pode ser feito em todos os níveis de teste

Teste caixa-branca

- Foco em testes com base na estrutura interna do sistema
 - ◆ Código-fonte
 - ◆ Arquitetura
 - ◆ Fluxo de dados
- Cobertura de código com testes de unidade ou integração

Testes de mudanças

- Teste de confirmação: Verificação após defeito ser corrigido
- Teste de regressão: Verificação de efeitos colaterais nas alterações de um componente do sistema

Aula 3 . Etapa 5

Técnicas de teste

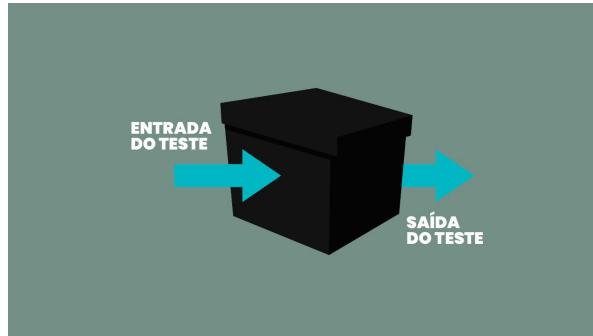
// Fundamentos de qualidade de software

Objetivos das técnicas

- Auxílio na identificação das condições de teste, casos e seus dados
- Técnicas
 - ◆ Caixa-preta
 - ◆ Caixa-branca
 - ◆ Por experiência

Técnicas de caixa-preta

- Fundamentadas em documentos de requisitos, casos de uso, histórias do usuário,etc...
- São aplicáveis para testes funcionais ou não-funcionais
- Foco nas entradas e saídas do teste, abstraindo a estrutura interna



1. **Particionamento de equivalência**
2. **Análise de valor limite**
3. **Tabela de decisão**
4. **Transição de estado**
5. **Caso de uso**

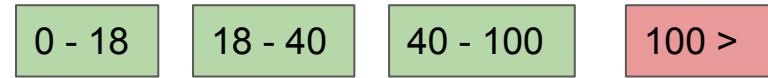
Particionamento de equivalência

- Divide os dados em partições ou classes de equivalência que são processados da mesma forma, em formatos válidos e inválidos.

Exemplo: Um sistema de gestão e simulação de investimentos faz recomendações específicas dependendo da idade, tendo uma pontuação de risco de 0-100:

- até 18 anos : investimentos com risco 60-80
- 18 até 40 anos: investimentos com risco entre 40-60
- idade > 40 : investimentos com risco menor
- a idade máxima que o sistema faz previsões e simulações: 100 anos

Cenário para verificar simulação de investimentos



Válidos

Inválidos

Análise de valor limite

- Estende o particionamento de equivalência quando a partição é ordenada e podemos analisar o valor mínimo e máximo

Cenário para verificar valor de frete

Exemplo: Um sistema de gerenciamento de envio de mercadorias possui as seguintes regras:

- O cliente não paga frete acima 100 reais
- Entre 50 e 100 reais, paga 20 reais
- Menor que 50 reais, o frete sobe para 35 reais
- Caso o valor total da compra chegue a R\$100.000 o cliente deve entrar em contato diretamente ou fazer uma nova compra.

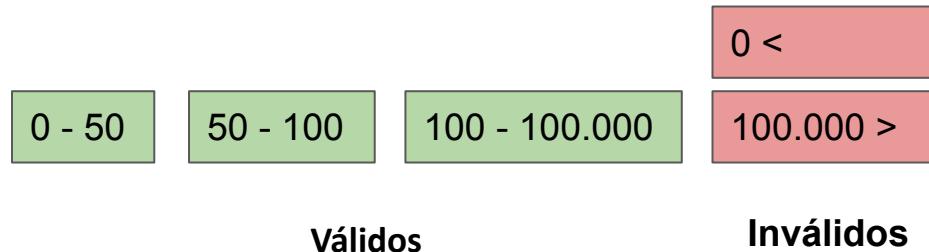


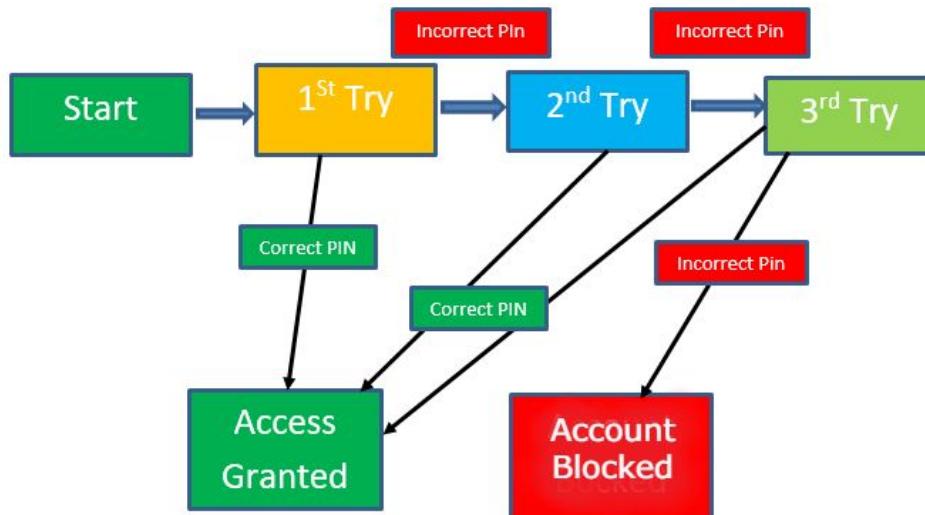
Tabela de decisão

- Úteis para testar requisitos que especificam condições que combinações com diferentes resultados

Variáveis	1	2	3	4
Cartão válido?	Não	Sim	Sim	Sim
Senha Válida?	X	Não	Sim	Sim
Valor solicitado é <= Saldo	X	X	Não	Sim
Saída Esperada	Cartão Inválido	Senha inválida	Saldo Insuficiente	Saque efetuado com sucesso

Transição de estado

- Situações em que o sistema reage diferente a um evento dependendo das condições atuais ou de um histórico, que pode ser resumido como estados



Fonte: Guru99(2022)

Transição de estado

- É gerada uma tabela de transição que vai direcionar os casos de teste

	Correct PIN	Incorrect PIN
S1) Start	S5	S2
S2) 1 st attempt	S5	S3
S3) 2 nd attempt	S5	S4
S4) 3 rd attempt	S5	S6
S5) Access Granted	-	-
S6) Account blocked	-	-

Fonte: Guru99(2022)

Teste de caso de uso

- Derivados naturalmente dos casos de uso
- Associa-se ações com os atores do caso
- Projeta-se testes para casos básicos, alternativos e de erros

Técnicas de caixa-branca

- Baseadas na estrutura interna do objeto de teste
- Podem ser usadas em todos os níveis de teste
- Normalmente usada para testes a nível de componente no código-fonte

1. Cobertura de instruções
2. Cobertura de decisões



Teste de cobertura de instruções

- Testa instruções executáveis do código
- Cobertura medida como (número de instruções executadas)/ (total de instruções)



[Cobertura de código explicada. Relatórios e métricas com Istanbul e o... | by Eduardo Rabelo | Medium](#)

Teste de cobertura de decisões

- Testa as condicionais existentes no código e o que é executada em cada decisão
- Cobertura = número de resultados de decisão executados/ total de resultados de decisão no objeto
- 100% de cobertura de decisão → 100% de cobertura de instrução

Teste de cobertura de decisões

- Testa as condicionais existentes no código e o que é executada em cada decisão
- Cobertura = número de resultados de decisão executados/ total de resultados de decisão no objeto
- 100% de cobertura de decisão → 100% de cobertura de instrução

Técnicas baseadas na experiência

- Baseada em experiência e intuição de quem testa
 - Pode-se identificar situações não encontradas nos métodos mais sistemáticos
 - Cobertura de difícil avaliação e medição
-
1. **Suposição de erro**
 2. **Teste exploratório**
 3. **Baseado em checklist**

Fundamentos de Qualidade de Software

Carolina Santana Louzada

Engenheira de Qualidade de Software - UOLEdtech

Para saber mais

- ★ [CTFL \(bstqb.org.br\)](http://CTFL.bstqb.org.br)
- ★ Família SQuaRE: ISO/IEC 25000-25099
- ★ martinfowler.com
- ★ [Cobertura de código explicada. Relatórios e métricas com Istanbul e o... | by Eduardo Rabelo | Medium](https://medium.com/@eduardorabelo/cobertura-de-c%C3%B3digo-explicada-relat%C3%B3rios-e-m%C3%A9tricas-com-istanbul-e-o-...-by-eduardo-rabelo-1253f3a2a2)

Percurso

Aula 1

O que é qualidade de software?

Aula 2

Gerenciamento de defeitos

Aula 3

Introdução aos testes de software

Dúvidas durante o curso?

- > Fórum do curso
- > Comunidade online (Discord)



Ciclo de desenvolvimento de software e metodologias ágeis

Carolina Santana Louzada

Analista QA - Venturus

Mais sobre mim

- Graduada em Engenharia de Computação- UFS
- Fazendo especialização em qualidade e desenvolvimento de software
- Qualidade de software -> automação
- Educação + tecnologia
- Jogos + música + aprender novas atividades
- LinkedIn -> [Carolina Santana Louzada | LinkedIn](#)

Objetivo do curso

Compreender o ciclo de vida do software e seus processos, bem como entender como a qualidade e os testes atuam nos processos, tendo como foco o pensamento ágil.

Pré-requisitos

- ★ Ter completado curso de Fundamento de Qualidade de Software

Percurso

Aula 1

Processos de software

Aula 2

Desenvolvimento ágil

Aula 3

Testes no mundo ágil

Dúvidas durante o curso?

- > Fórum do curso
- > Comunidade online (Discord)



Aula 1

Processos de software

// Ciclo de desenvolvimento de software e
metodologias ágeis

Objetivos

- Definição de processo, fluxo e padrões de software
- Compreender modelos prescritivos
- Compreender modelos incremental, evolucionário e concorrente
- Compreender modelos mais especializados
- Compreender o processo unificado (RUP)

Aula 1 . Etapa 1

Definindo processo, fluxo e padrões de software

// Ciclo de desenvolvimento de software e
metodologias ágeis

“...o desenvolvimento de software é um processo de aprendizado social.



Trata-se de um processo iterativo no qual a própria ferramenta em evolução serve como meio de comunicação...

Citação de Howard Baetjer Jr. (apud Pressman, Roger S., Maxim, Bruce R. Engenharia de Software: uma abordagem profissional, 8^aed)

O que é processo de software?

Metodologia para as atividades, ações e tarefas necessárias para desenvolver um software.



Algumas reflexões...

- Todos os envolvidos são diretamente ou indiretamente responsáveis.
- Gera estabilidade, controle e organização dentro do contexto.
- Processos são adaptáveis de acordo com o produto a ser construído.

Atividades principais do processo

Segundo Pressman:

- Comunicação
- Planejamento
- Modelagem
- Construção
- Entrega

Segundo Sommerville:

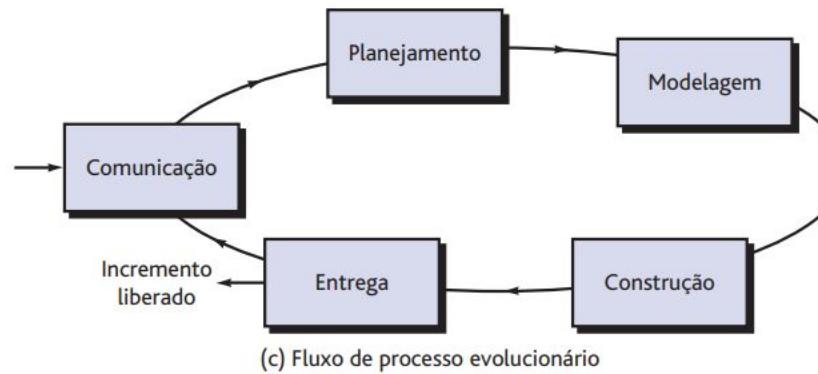
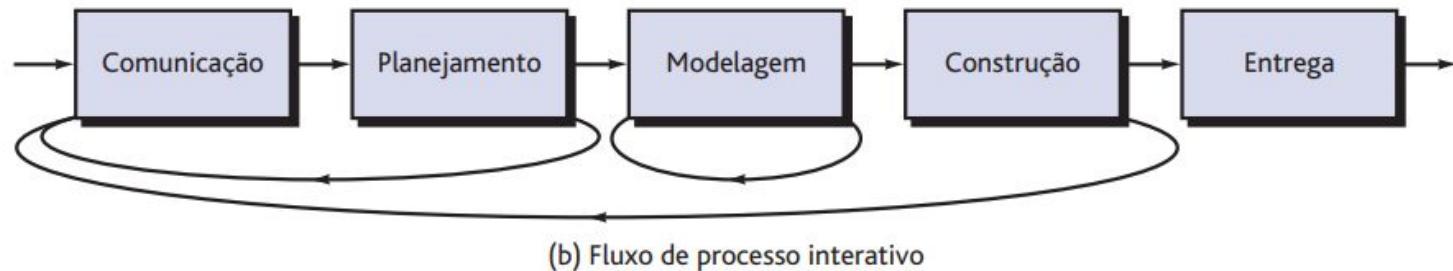
- Especificação
- Projeto e implementação
- Validação
- Evolução

Atividades principais do processo

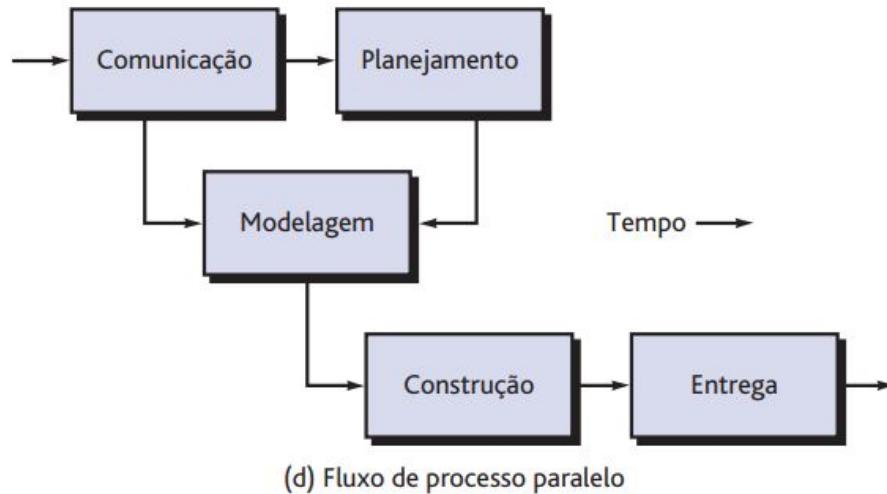
- Organização dos processos e suas relações = **fluxo do processo**



Fluxos de processo



Fluxos de processo



Cada processo é
constituído de tarefas!

Padrões de processo

- Descreve um problema de processo e sugere soluções de acordo com o contexto
- Pode ser definido para qualquer nível de abstração
- Abordagens de avaliação e aperfeiçoamento:
 - ◆ SCAMPI(Standard CMMI Assessment Method for Process Improvement)
 - ◆ CBA IPI(CMM-Based Appraisal for internal Process Improvement)
 - ◆ SPICE(ISO/IEC 15504) - requisitos para avaliação de processos de software
 - ◆ ISO 9001:2000

Aula 1 . Etapa 2

Modelo prescritivo

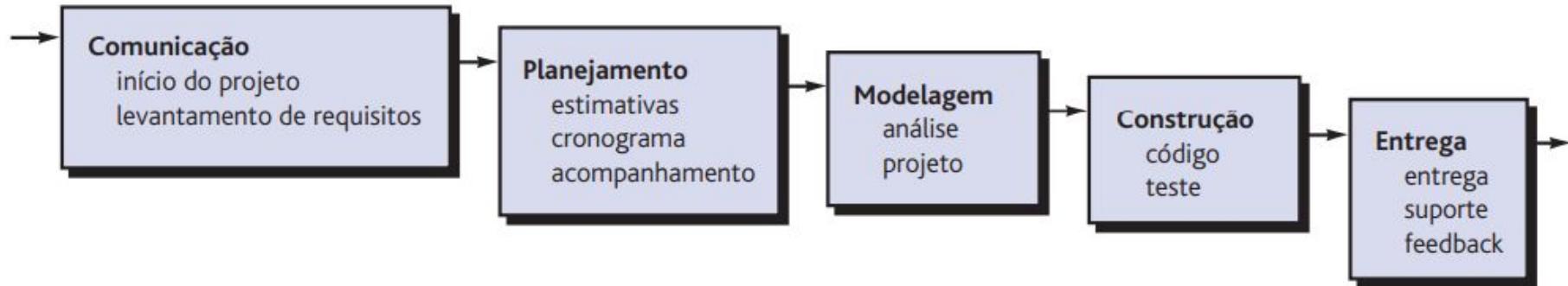
// Ciclo de desenvolvimento de software e
metodologias ágeis

O que é?

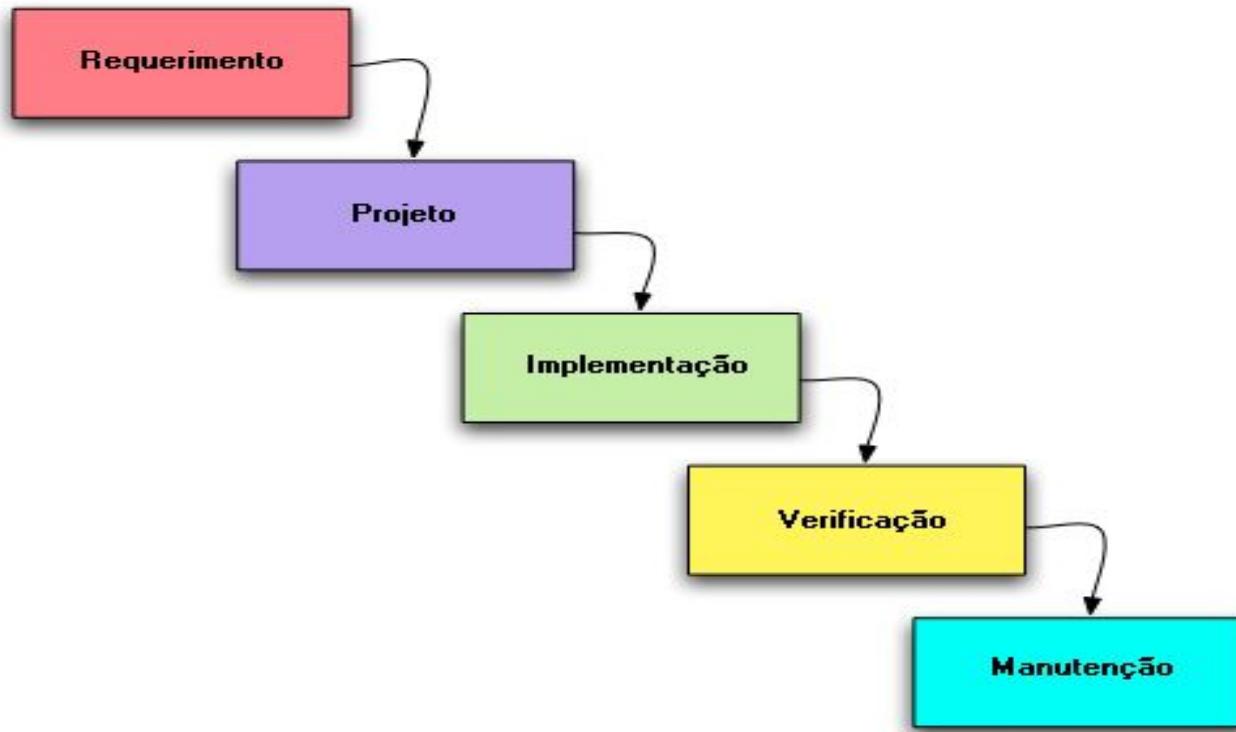


- ★ Também chamado de modelos “tradicionais”
- ★ Foco na ordem e consistência do processo
- ★ Prescrevem conjunto de elementos de processo e fluxos

Modelo Cascata - Clássico



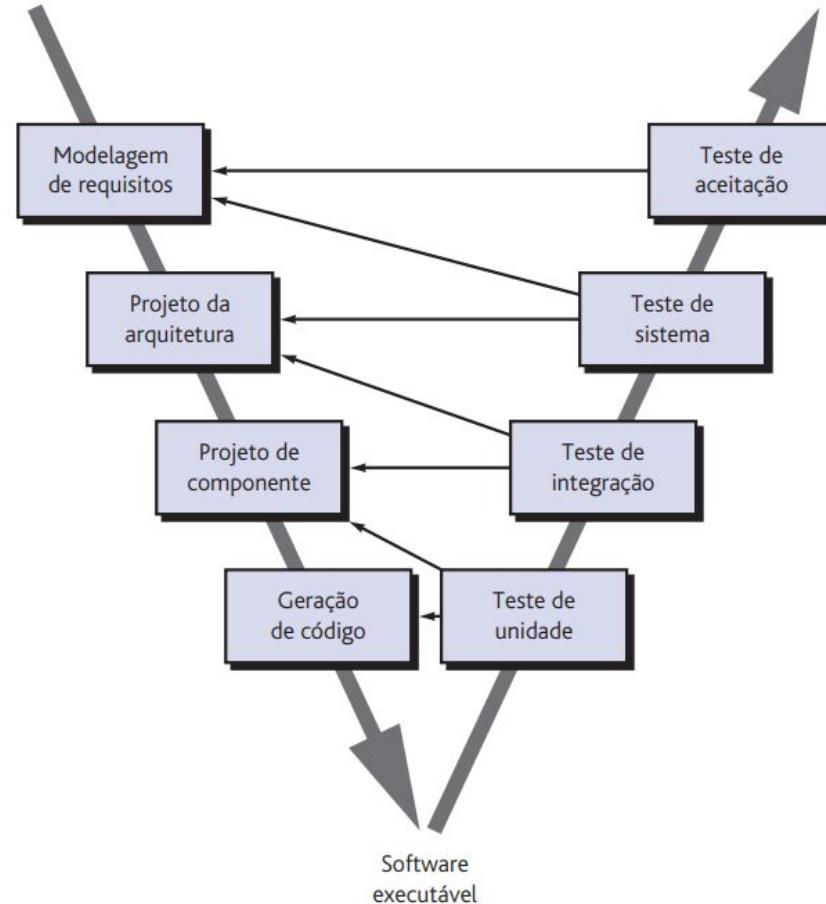
Modelo Cascata - Clássico



Modelo Cascata - Clássico

- Útil para requisitos bem compreendidos, definidos e estáveis
- Processo linear e sistemático

Modelo V



Modelo V

- Relação entre atividades de garantia de qualidade e atividades restantes do processo
- Não há diferença fundamental entre o Cascata e V

Modelo Cascata - Problemas

- Projetos reais não seguem fluxos sequenciais
- Não lida bem com adaptação constante de mudanças
- Requisitos não são bem estabelecidos na primeira fase
- Longo tempo para visualizar primeira versão do software
- Gera estados de bloqueio para a equipe

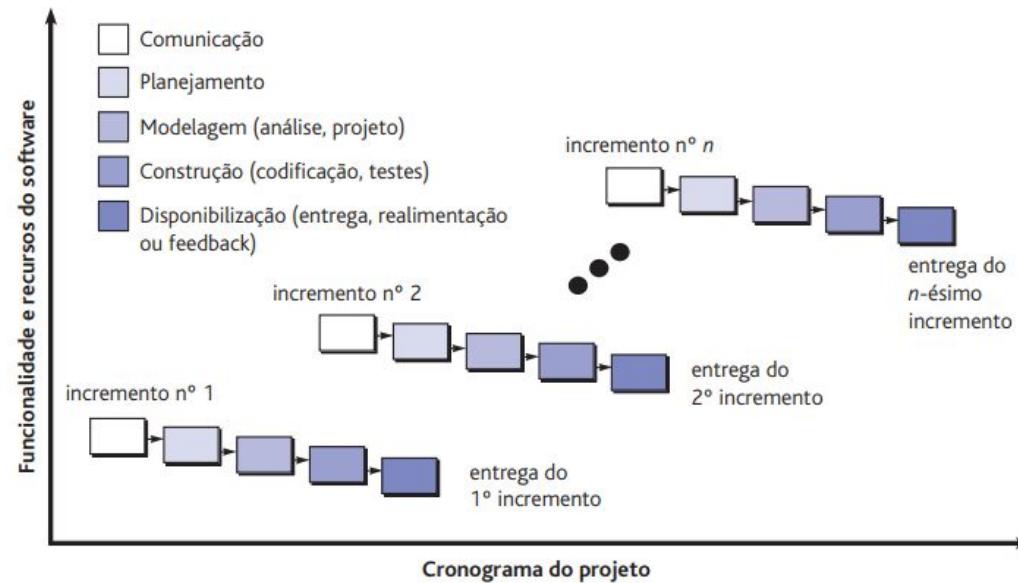
Aula 1 . Etapa 3

Modelos incremental, evolucionário e concorrente

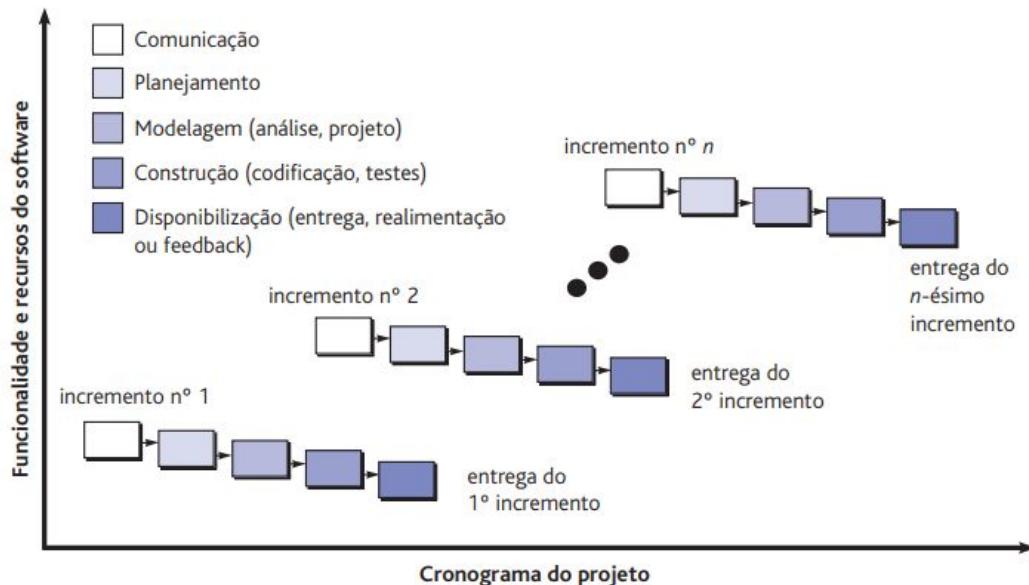
// Ciclo de desenvolvimento de software e
metodologias ágeis

Modelos de processo incremental

- Situações com requisitos iniciais bem definidos, mas não refinados



Modelos de processo incremental

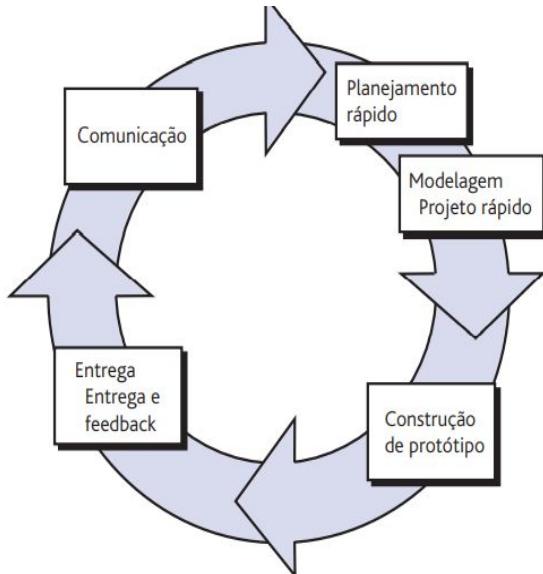


★ Pode-se utilizar prototipagem

Modelo evolucionário

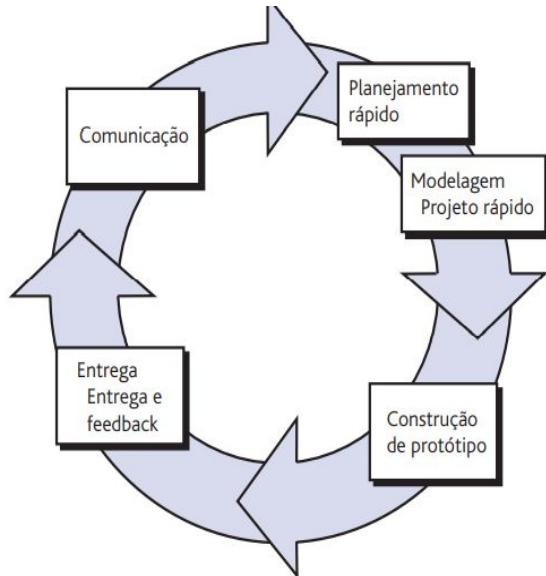
- ★ Modelo que possibilita o desenvolvimento de um software que cresce e se adapta constantemente
- ★ São iterativos
- ★ Modelos:
 - Prototipagem
 - Espiral

Modelo evolucionário - Prototipação



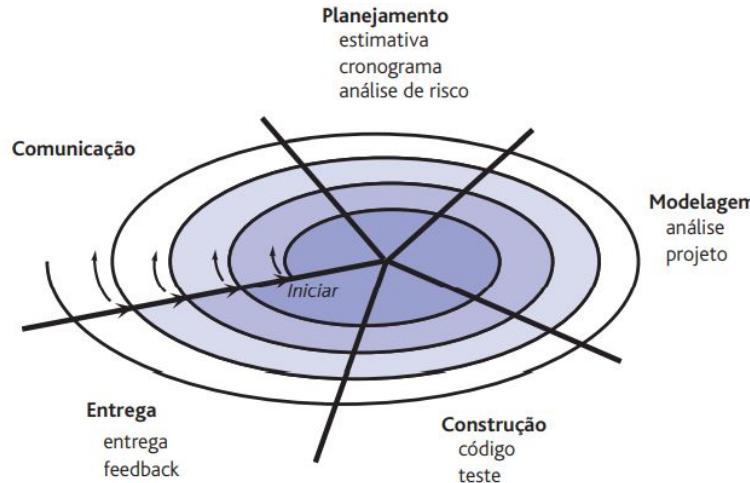
- ★ Útil para refinar requisitos
- ★ Validar eficiência e interação com usuário
- ★ Pode ser aplicado isoladamente ou em conjunto com outros processos
- ★ O protótipo atua como forma de obtenção de requisitos
- ★ Podem ser descartáveis ou podem evoluir

Problemas da Prototipação



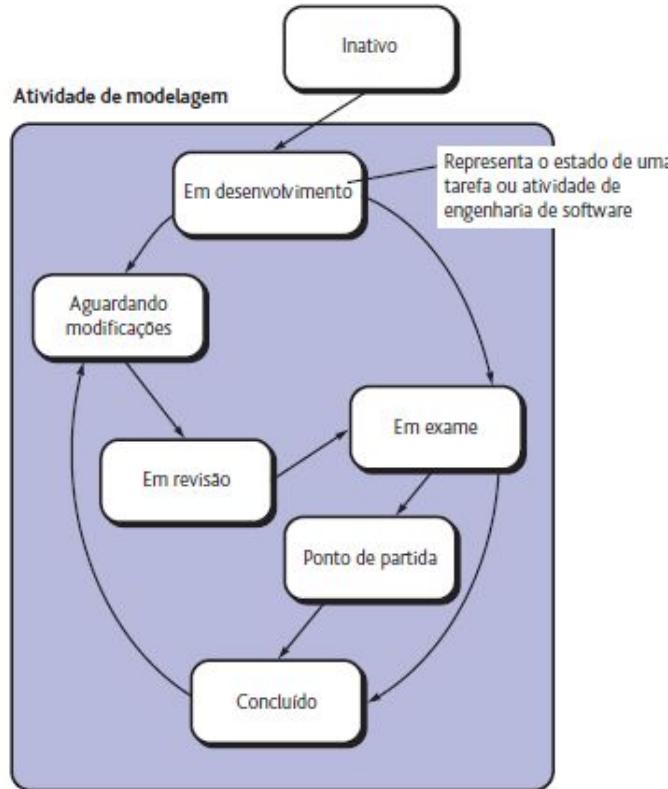
- ★ Falta de consideração da qualidade global do software após prototipação ser considerada “funcional”
- ★ Acomodar com escolhas iniciais da prototipação

Modelo evolucionário - Espiral



- ★ Natureza iterativa da prototipação + aspectos sistemáticos do cascata
- ★ Estratégia cílica incremental com foco em diminuir riscos

Modelo concorrente



- ★ Representação concorrente de atividades de qualquer processo

Aula 1 . Etapa 4

Modelos especializados

// Ciclo de desenvolvimento de software e
metodologias ágeis

Baseado em componentes

- ★ Desenvolvimento com base em componentes com interfaces definidas para serem integradas ao software -> COTS(commercial off-the-shelf)
 - módulos ou pacotes de classes
- ★ Evolucionário por natureza
- ★ Foco em reutilização -> Redução no tempo de desenvolvimento e custos

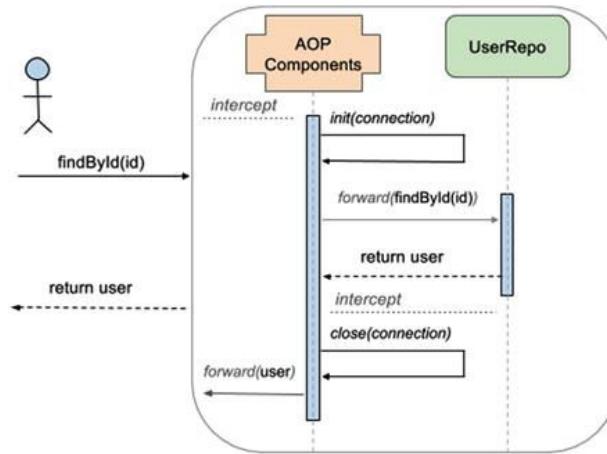
Modelo de métodos formais

- ★ Atividades baseadas em condução à especificação matemática formal do software -> utilização de notação matemática
- ★ Análise matemática auxilia na descobertas de ambiguidades ou inconsistências.
- ★ Desenvolvimento consome tempo e dinheiro
- ★ Complexidade exige formação e treinamento
- ★ Bem visto para softwares com fatores críticos



Modelo orientado a aspectos

- ★ Paradigma que oferece uma abordagem metodológica e de processos para definir, especificar, projetar e construir aspectos, que são pontos de interesse que se propagam e entrecortam outras partes da aplicação.



Aula 1 . Etapa 5

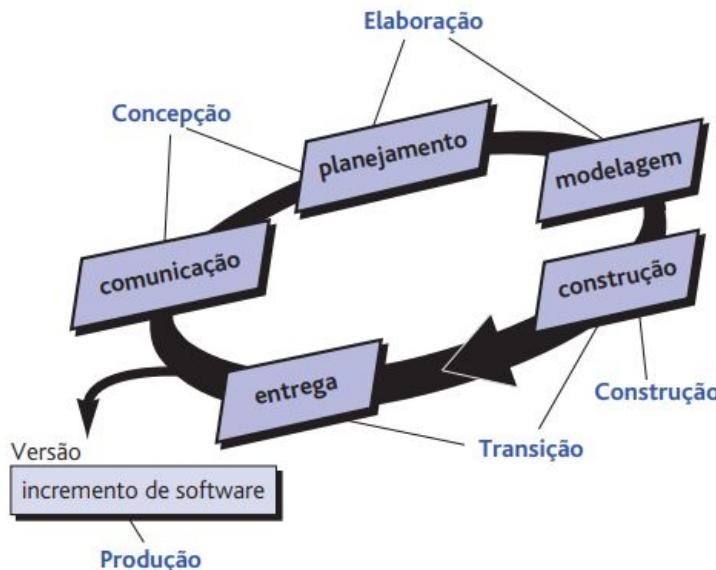
Processo Unificado

// Ciclo de desenvolvimento de software e
metodologias ágeis

Um pouco de história

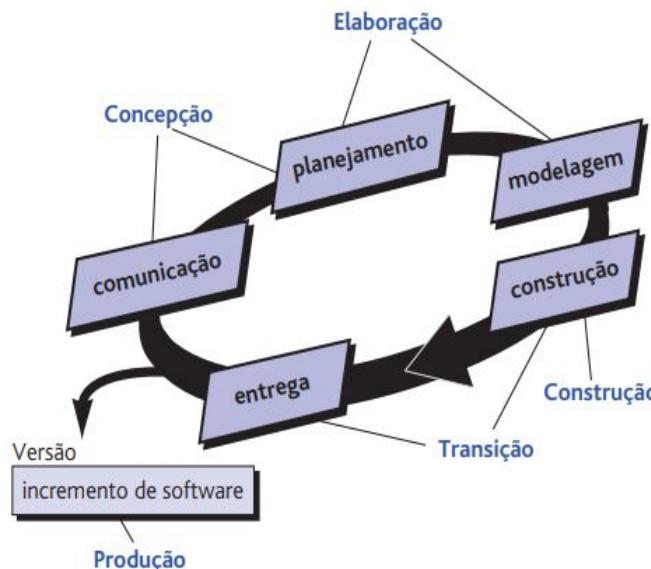
- ★ No início dos anos 90 , James Rumbaugh, Grady Booch e Ivar Jacobson começaram a trabalhar em um “método unificado” que combinasse as melhores características de outros processos -> UML
- ★ Necessidade de um processo de software dirigido a casos de uso, centrado na arquitetura, iterativo e incremental.

Fases do processo unificado



1. Fase de Concepção
 - Comunicação
 - Planejamento
- Requisitos são descritos em conjunto de casos de uso preliminares.
- Identificação de recursos, riscos, cronograma...

Fases do processo unificado

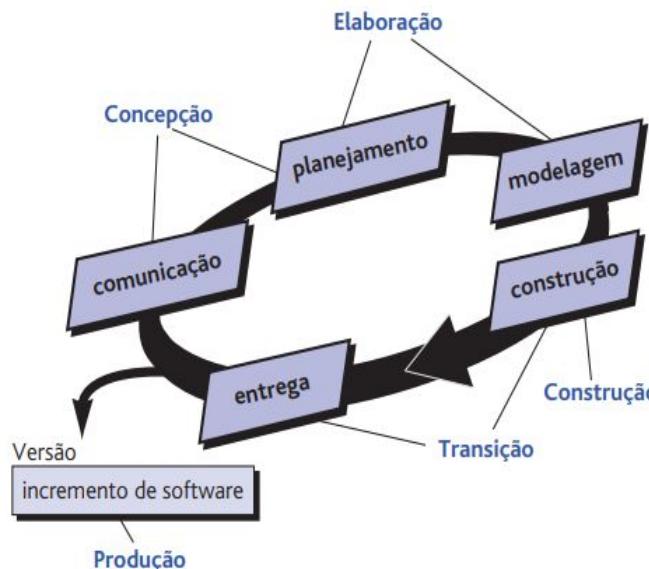


2. Fase de Elaboração
 - Planejamento
 - Modelagem

→ Refinamento e expansão de casos de uso

→ Ampliação de representação arquitetural:
 - ◆ casos de uso
 - ◆ modelo de análise
 - ◆ modelo de projeto
 - ◆ modelo de implementação
 - ◆ modelo de disponibilização

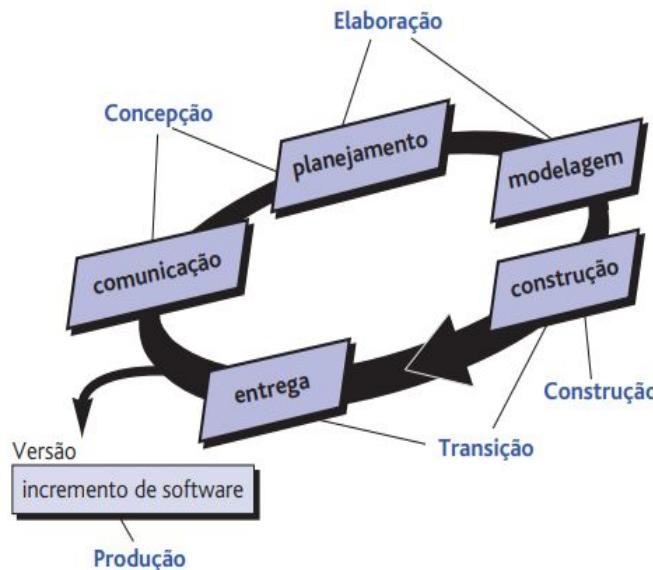
Fases do processo unificado



3. Fase de Construção

- Desenvolvimento de software com base nos modelos
- Uso dos modelos para gerar suíte de testes de aceite
- Utilização de testes conforme desenvolvimento

Fases do processo unificado

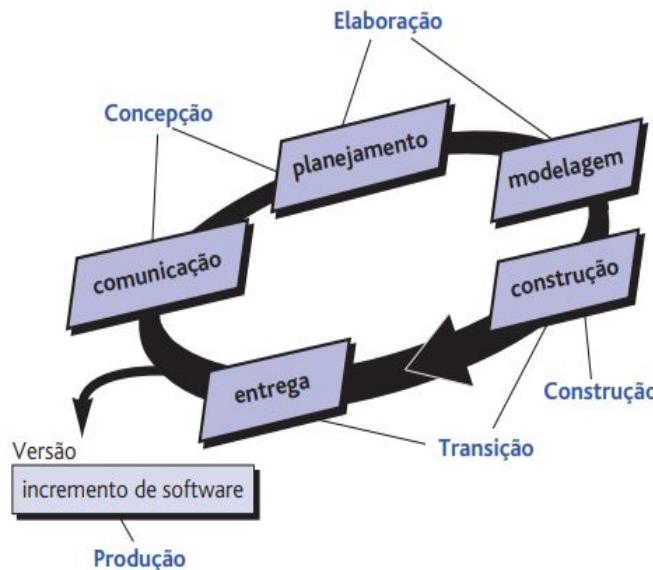


4. Fase de Transição

- Construção
- Entrega

- Comum entrega com testes beta para recebimento de feedbacks;
- O incremento torna-se uma versão utilizável do software

Fases do processo unificado

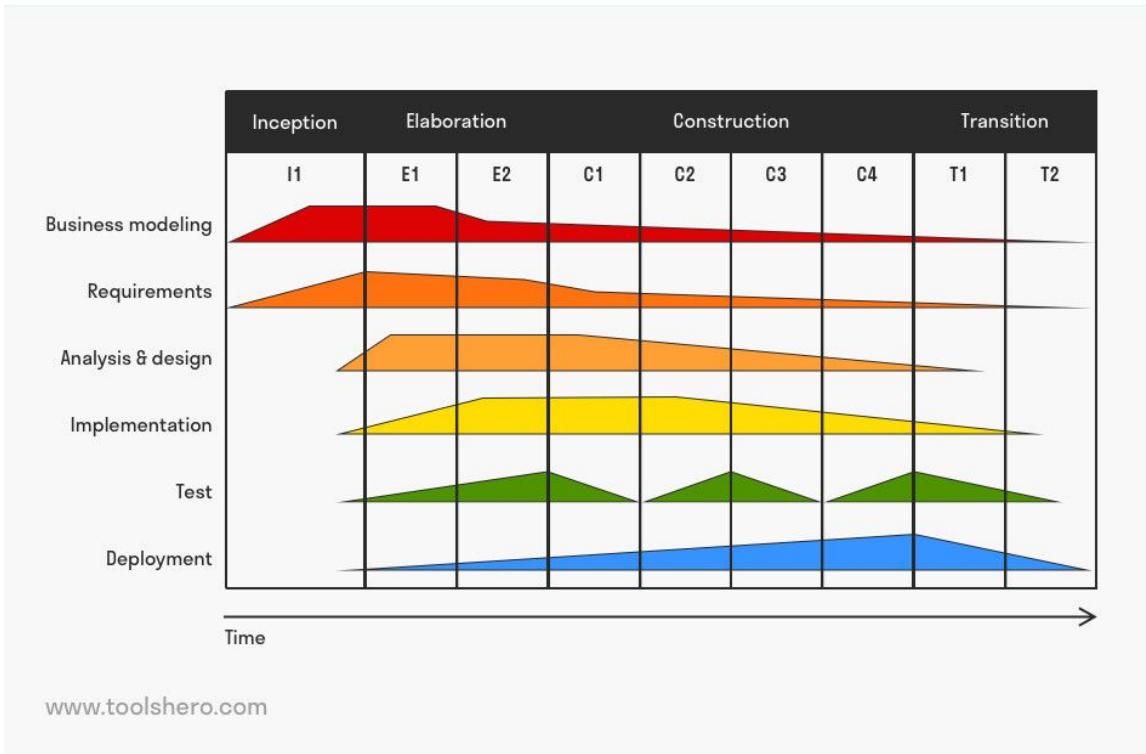


5. Fase de Produção

- Entrega

- Monitoramento de uso contínuo
- Suporte
- Relatórios para defeitos e mudanças

Fases do processo unificado



Para saber mais

- Modelo de processo pessoal e de equipe
- Pesquise sobre os autores do Manifesto Ágil!
- Programação orientada a Aspectos

Aula 2

Desenvolvimento ágil

// Ciclo de desenvolvimento de software e
metodologias ágeis

Objetivos

- Contexto da criação do manifesto ágil e seus conceitos
- Compreender o Extreme Programming - XP
- Compreender o Scrum
- Compreender a existência e conceitos de outros modelos ágeis

Aula 2 . Etapa 1

O manifesto ágil

// Ciclo de desenvolvimento de software e
metodologias ágeis

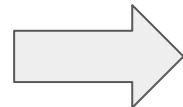
Contexto e surgimento

- ★ Em contramão aos ditos métodos tradicionais ou “pesados”, 17 profissionais que já praticavam os “métodos leves” se reuniram em Utah no ano de 2001 e chegaram ao consenso de métodos e práticas para o desenvolvimento de software -> MANIFESTO ÁGIL

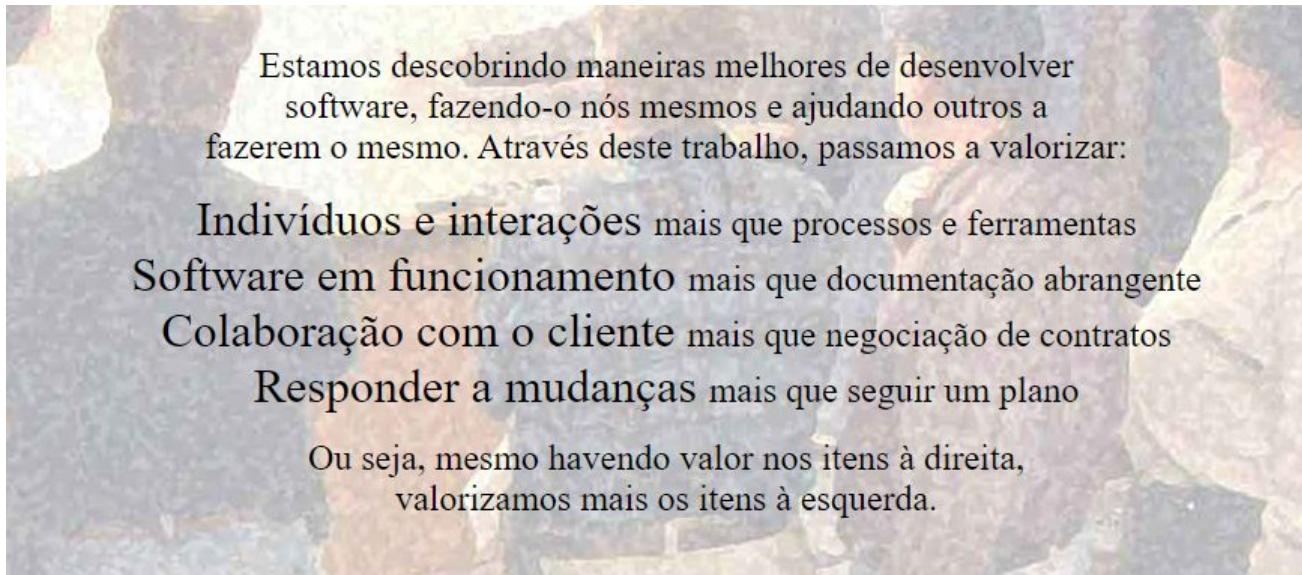
Contexto e surgimento

Manifesto ágil

Seus autores



O que diz o manifesto?



Estamos descobrindo maneiras melhores de desenvolver software, fazendo-o nós mesmos e ajudando outros a fazerem o mesmo. Através deste trabalho, passamos a valorizar:

Indivíduos e interações mais que processos e ferramentas
Software em funcionamento mais que documentação abrangente
Colaboração com o cliente mais que negociação de contratos
Responder a mudanças mais que seguir um plano

Ou seja, mesmo havendo valor nos itens à direita, valorizamos mais os itens à esquerda.

Fonte: agilemanifesto.org

Os 12 princípios

1. Nossa maior prioridade é satisfazer o cliente através da entrega contínua e adiantada de software com valor agregado.
2. Mudanças nos requisitos são bem-vindas, mesmo tardiamente no desenvolvimento. Processos ágeis tiram vantagem das mudanças visando vantagem competitiva para o cliente.
3. Entregar frequentemente software funcionando, de poucas semanas a poucos meses, com preferência à menor escala de tempo.
4. Pessoas de negócio e desenvolvedores devem trabalhar diariamente em conjunto por todo o projeto.
5. Construa projetos em torno de indivíduos motivados. Dê a eles o ambiente e o suporte necessário e confie neles para fazer o trabalho.
6. O método mais eficiente e eficaz de transmitir informações para e entre uma equipe de desenvolvimento é através de conversa face a face.

Os 12 princípios

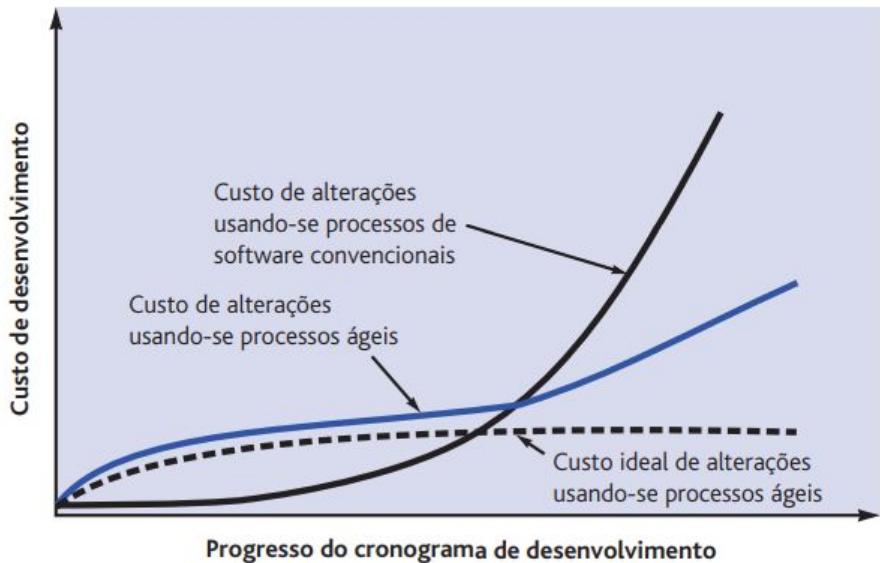
7. Software funcionando é a medida primária de progresso.
8. Os processos ágeis promovem desenvolvimento sustentável. Os patrocinadores, desenvolvedores e usuários devem ser capazes de manter um ritmo constante indefinidamente.
9. Contínua atenção à excelência técnica e bom design aumenta a agilidade.
10. Simplicidade – a arte de maximizar a quantidade de trabalho não realizado – é essencial.
11. As melhores arquiteturas, requisitos e designs emergem de equipes auto-organizáveis.
12. Em intervalos regulares, a equipe reflete sobre como se tornar mais eficaz e então refina e ajusta seu comportamento de acordo.

O que significa ser ágil ?



- ★ Métodos prescritivos não deixam de ser úteis, porém tem um ponto que pode torná-lo falho : **as fraquezas e falhas de quem desenvolve o software!**
- Condutores da agilidade : adaptação + comunicação + auto-organização

O que significa ser ágil ?



- Condutores da agilidade : adaptação + comunicação + auto-organização
- A adaptação auxilia na diminuição de custos por alterações

Motivações

- Difícil prever requisitos de softwares e suas possíveis alterações
- Difícil prever priorizações do cliente
- Análise, projeto e testes não são previsíveis
- As atividades de construção do software não são facilmente estimadas



Aula 2 . Etapa 2

Extreme programming - XP

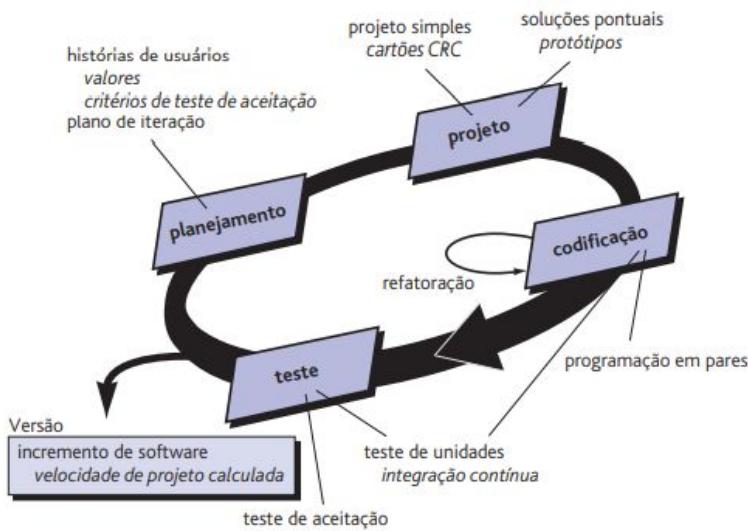
// Ciclo de desenvolvimento de software e
metodologias ágeis

Surgimento ‘Programação Extrema’

- Primeiros trabalhos e métodos associados : 1980
- Trabalho originário por Kent Beck
- Existe variante com refinamentos para grandes organizações:
IXP - Industrial Extreme Programming

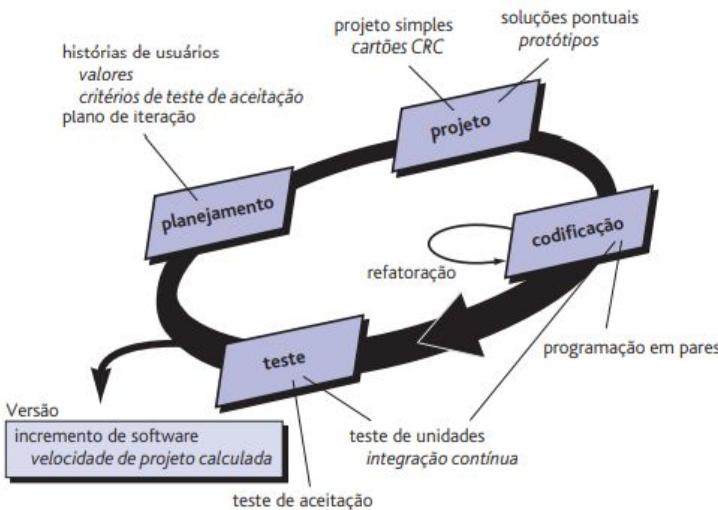
[Extreme Programming Explained: Embrace Change](#)

O processo



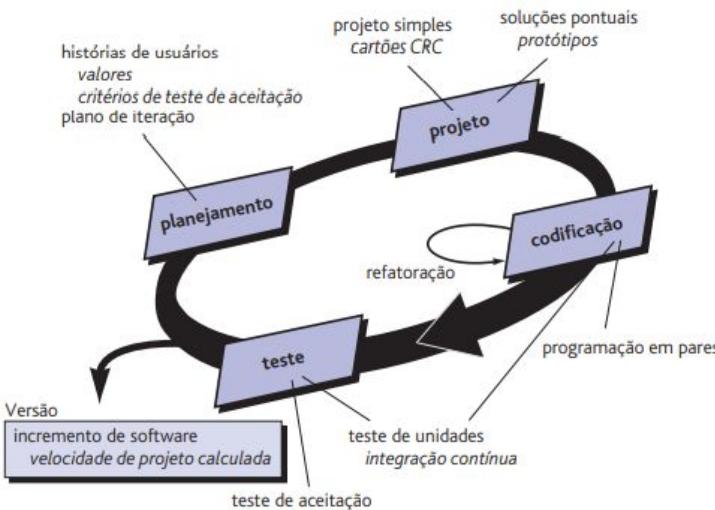
- Paradigma foco: orientação a objetos
- Envolve regras e práticas constantes durante processo de software

O processo: Planejamento



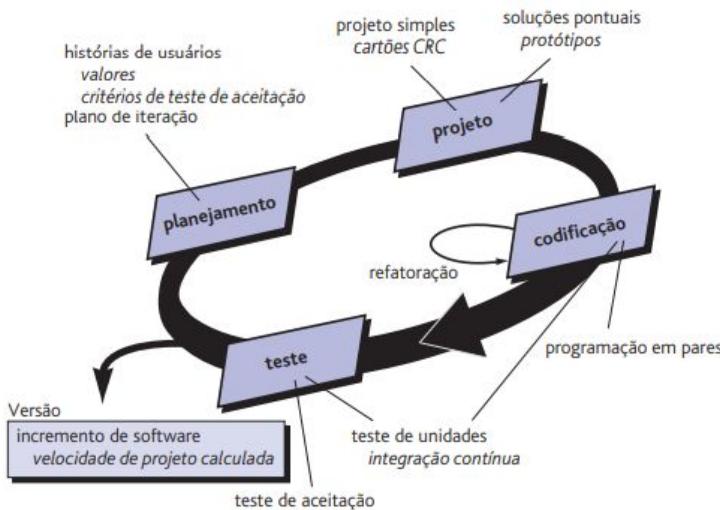
- Foco na comunicação, no 'ouvir' a partir do **planning poker**.
- A atividade leva a criação de **histórias do usuário** pelo cliente, que também as prioriza.
- Membros estimam com base em **semanas** de desenvolvimento -> máximo ideal de 3 semanas

O processo: Planejamento



- Flexibilização para escrita de novas histórias
- Clientes e desenvolvedores trabalham lado a lado = compromisso básico
- **Velocidade:** nº histórias entregues

O processo: Projeto

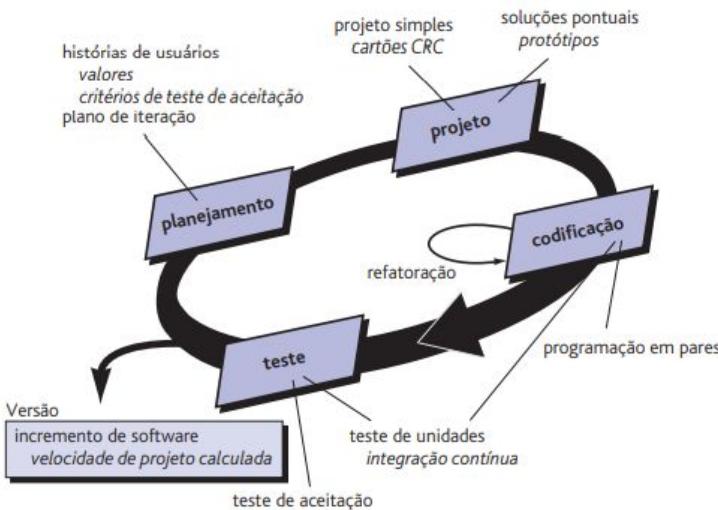


- Princípio KISS(*keep it simple, stupid!*)
- Estímulo no uso de cartões CRC(classe-responsabilidade-colaborador)

Ex: CRC

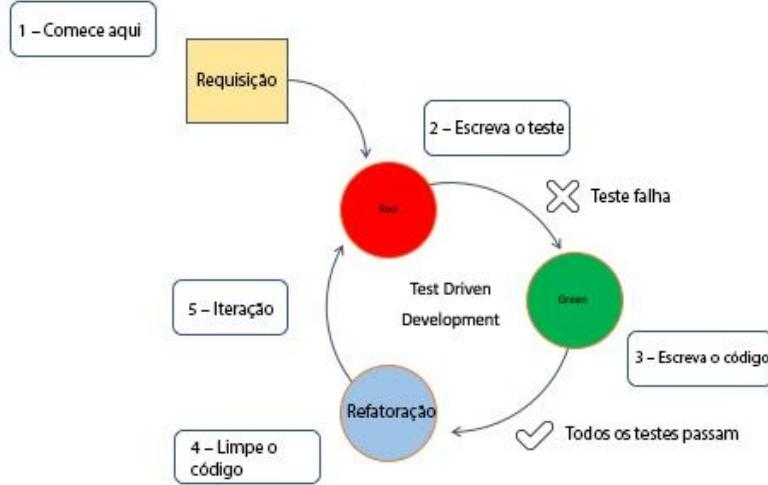
	Classes associadas	
	atributos	métodos
Classe: Conta Corrente		
Responsabilidade	Colaboração	
Saber o seu saldo	Cliente	
Saber seu cliente	Histórico de Transações	
Saber seu número		
Manter histórico de transações		
Realizar saques e depósitos		

O processo: Projeto



- Princípio **KISS**(*keep it simple, stupid!*)
- Estímulo no uso de **cartões CRC**(classe-responsabilidade-colaborador)
- Solução pontual a partir de **protótipos**

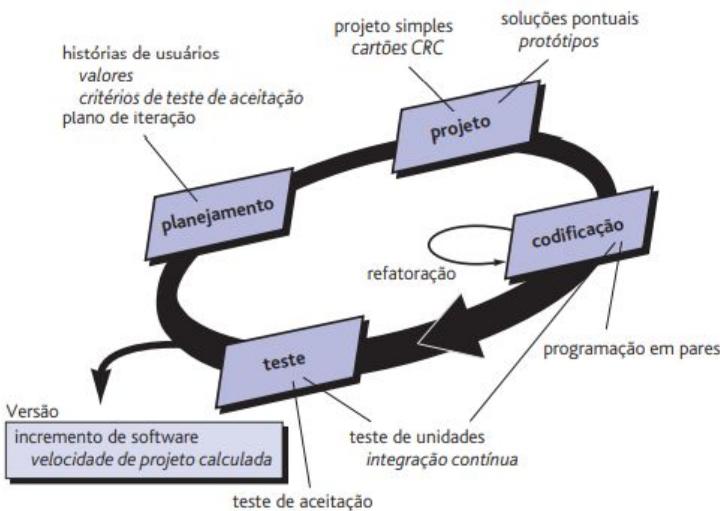
O processo: Codificação



- Uso de **TDD**(*Test Driven Development*)
- **Refatoração**: aperfeiçoamento de código
- Programação em pares

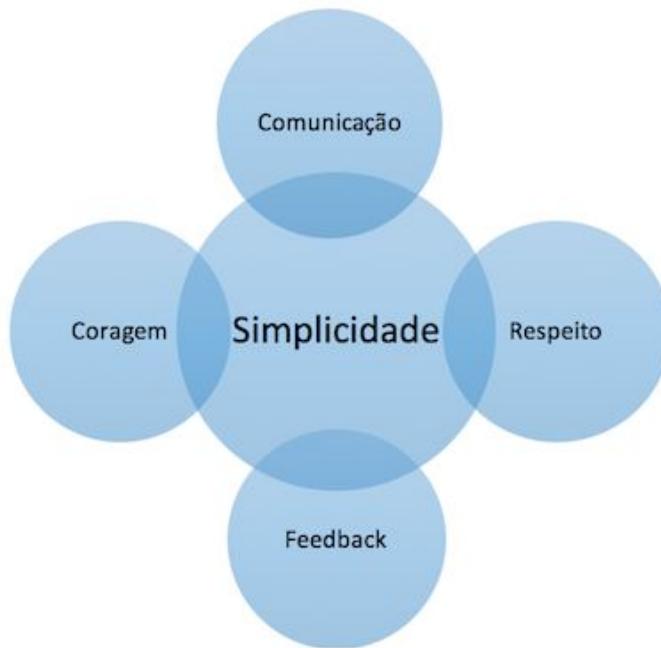
[Fonte: Tecmundo\(2020\)](#)

O processo: Testes



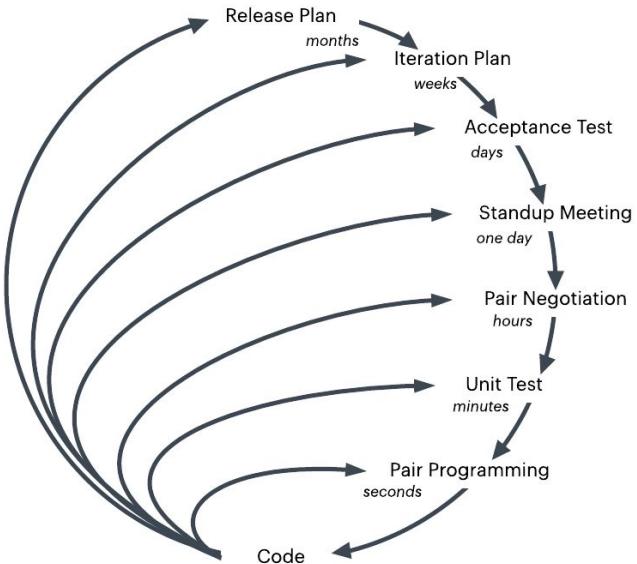
- ➔ Uso de **TDD** (*Test Driven Development*)
- ➔ Integração contínua
- ➔ Inclusão de testes de aceite -> histórias do usuário

Valores



Fonte: RedSpark(2016)

Planning and Feedback Loops



Made in
Lucidchart

Aula 2 . Etapa 3

Scrum

// Ciclo de desenvolvimento de software e
metodologias ágeis

Scrum

- Nome provém de uma ação em partida de rugby: jogadores dos dois times se juntam com a cabeça abaixada e se empurram para obter posse de bola

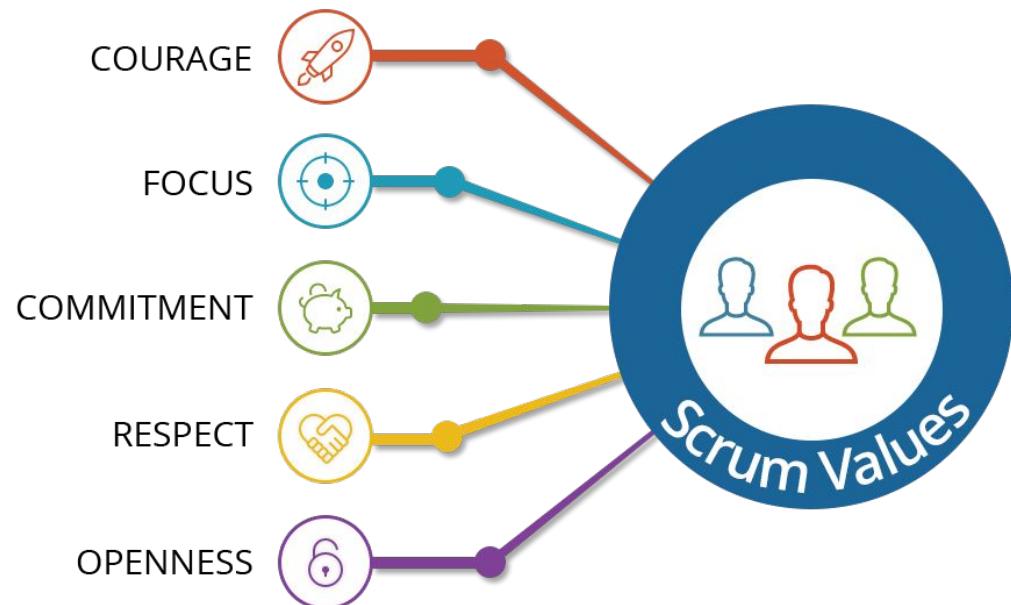
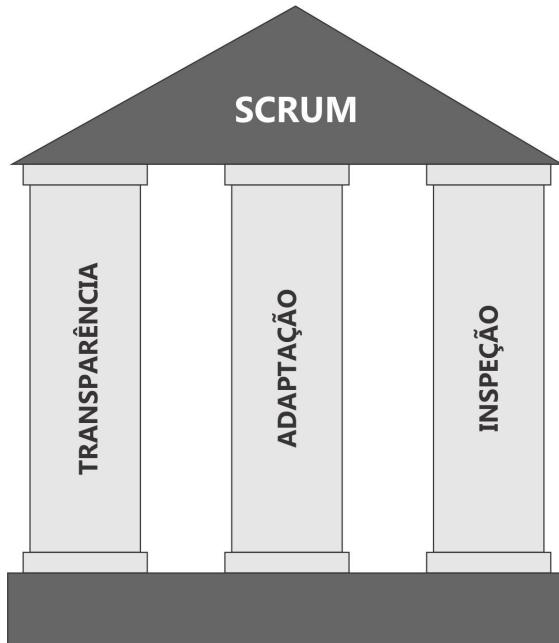


Surgimento e teoria

Scrum Guide

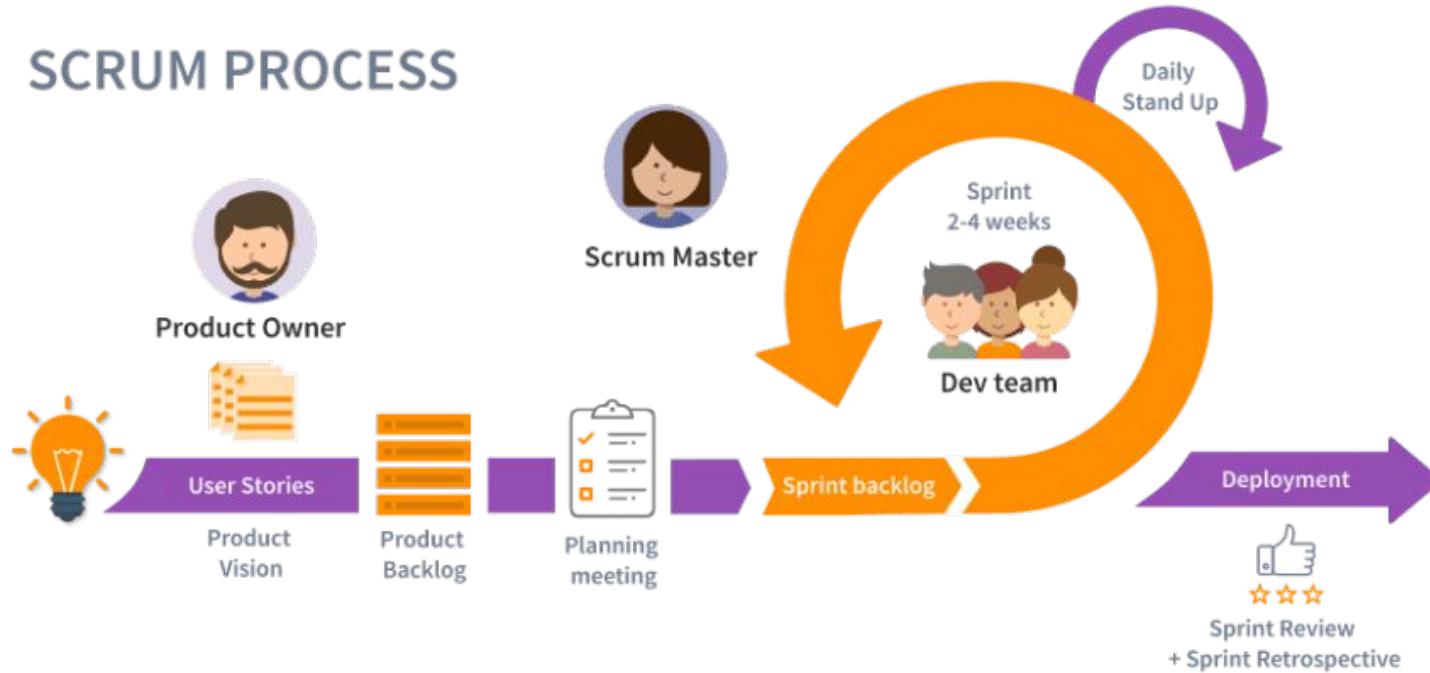
- Criado por Jeff Sutherland no início dos anos 1990 e desde então vem sendo revisado
- “Framework leve que ajuda pessoas, times e organizações a gerar valor por meio de soluções adaptativas para problemas complexos.”
- Baseado no **empirismo** e **lean thinking**
- Iterativo e incremental
- Eventos formais para inspeção e adaptação

Pilares e valores



Processo

SCRUM PROCESS



O Scrum Team

- ★ Pequeno time de pessoas sem hierarquia -> Produto
 - *Scrum master*
 - *Product owner*
 - *Developers*
- ★ São multifuncionais e autogerenciáveis
- ★ Responsáveis por todas as atividades relacionadas ao produtos

Product Owner

- ★ Maximiza valor do produto
- ★ Gerenciamento do Product Backlog:
 - Desenvolver e expressar meta do produto
 - Criar e comunicar itens do Backlog
 - Ordenar itens
 - Garantir que Product Backlog seja transparente, visível e compreensível

Desenvolvedores

- ★ Criação de incremento utilizável a cada sprint
- ★ Habilidades amplas de acordo com domínio do trabalho
- ★ Responsabilidades:
 - Criar Sprint Backlog
 - Alinhar a definição de Pronto
 - Adaptação com direção à meta da Sprint
 - Responsabilizar-se como profissionais

Scrum Master

- ★ Guardião do Scrum -> eficácia da metodologia
- ★ Liderança que serve à organização
- ★ Responsabilidades:
 - Treinar membros para auto-gerenciamento
 - Concentração do time
 - Remoção de impedimentos
 - Manutenção de eventos
 - Auxilia PO com técnicas e melhorias no gerenciamento do Backlog

Eventos : A Sprint

- ★ Eventos de duração fixa com objetivo de gerar incremento
- ★ Atividades = Sprint Planning + Daily + Sprint Review e Sprint Retrospective
- ★ Não se faz mudanças que coloque em risco a meta da Sprint
- ★ Foco na qualidade
- ★ Refinamento conforme necessário
- ★ Somente PO pode cancelar a Sprint

Eventos : Planning

- ★ Inicia sprint : definição do trabalho a ser realizado
- ★ Porque essa sprint é valiosa?
- ★ O que pode ser feito nesta Sprint?
- ★ Como o trabalho será realizado?
- ★ Criação do Sprint Backlog

Eventos : Daily

- ★ Inspeção em direção à meta da sprint
- ★ Adaptação do Sprint Backlog
- ★ Curta duração/diariamente
- ★ Plano de trabalho
- ★ Comunicação
- ★ Remoção de impedimento

Eventos : Sprint review

- ★ Apresentar e inspecionar resultados
- ★ Determinar adaptações
- ★ Ajuste no Product Backlog
- ★ Scrum Team + Stakeholders
- ★ Penúltimo evento da sprint

Eventos : Sprint retrospective

- ★ Planejamento voltado para qualidade e eficácia
- ★ Inspeção de processos, interações, ferramentas...
- ★ O que funcionou?
- ★ O que não funcionou?
- ★ Como foi resolvido?
- ★ Conclusão da sprint

Artefatos

- ★ Product Backlog -> Meta do produto
- ★ Sprint Backlog -> Meta da sprint
- ★ Incremento -> Definição de pronto

Aula 2 . Etapa 4

Outros modelos ágeis

// Ciclo de desenvolvimento de software e
metodologias ágeis

Método de Desenvolvimento de Sistemas Dinâmicos(DSDM)

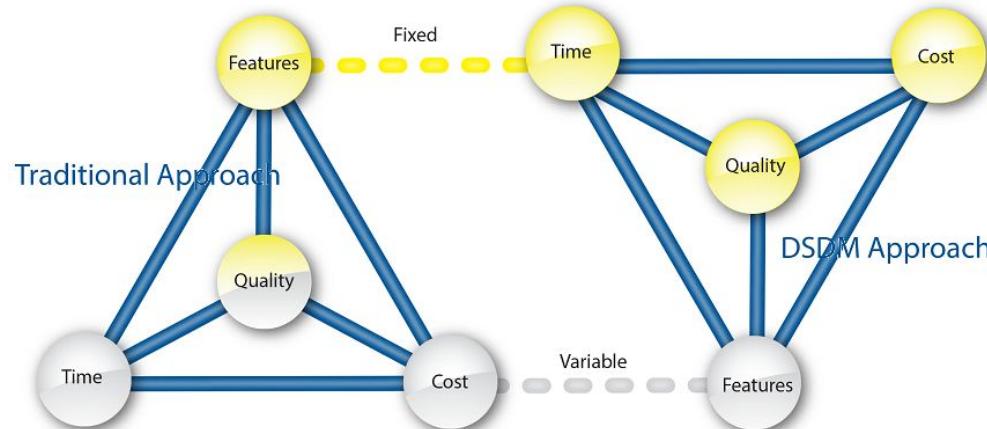
- Foco na construção e manutenção de sistemas que satisfaçam restrições de prazo curto por meio da prototipação em ambiente controlado
- Analogia com Princípio de Pareto: 80% de uma aplicação pode ser entregue em 20% do tempo que levaria para entregar a aplicação completa

PRINCÍPIO DE PARETO
CONHECIDO COMO
REGRA 80/20



Método de Desenvolvimento de Sistemas Dinâmicos(DSDM)

- É iterativo e incremental
- Somente o trabalho suficiente é requisitado para cada incremento
- Mantenedor -> [Agile Business Consortium](#)
- Pode ser combinado com XP



Método de Desenvolvimento de Sistemas Dinâmicos(DSDM)

- Princípios:
 - ◆ Focar na necessidade do negócio
 - ◆ Entregar dentro do prazo
 - ◆ Colaborar
 - ◆ Nunca comprometer a qualidade
 - ◆ Construir incrementalmente a partir de bases sólidas
 - ◆ Desenvolver iterativamente
 - ◆ Comunicar de forma contínua e clara
 - ◆ Demonstrar controle.

Método de Desenvolvimento de Sistemas Dinâmicos(DSDM)

- Fases:
 - ◆ Pré-projeto : Orçamento, contrato e projeto candidatos
 - ◆ Ciclo de vida: Desenvolvimento do produto
 - Análise de viabilidade
 - Iteração de modelo funcional
 - Iteração de design e construção
 - Implantação
 - ◆ Pós-projeto: Manutenção, melhorias e ajustes

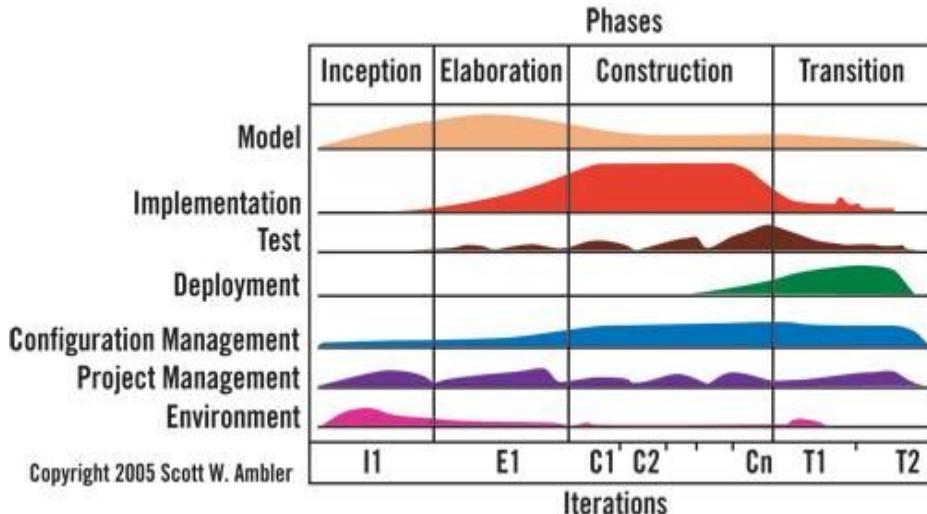
Método de Desenvolvimento de Sistemas Dinâmicos(DSDM)

→ Papéis:

- ◆ Gerente executivo
- ◆ Visionário
- ◆ Intermediador
- ◆ Anunciante
- ◆ Gerente de projeto
- ◆ Coordenador técnico
- ◆ Líder de time
- ◆ Desenvolvedor
- ◆ Testador
- ◆ Escrivão
- ◆ Facilitador

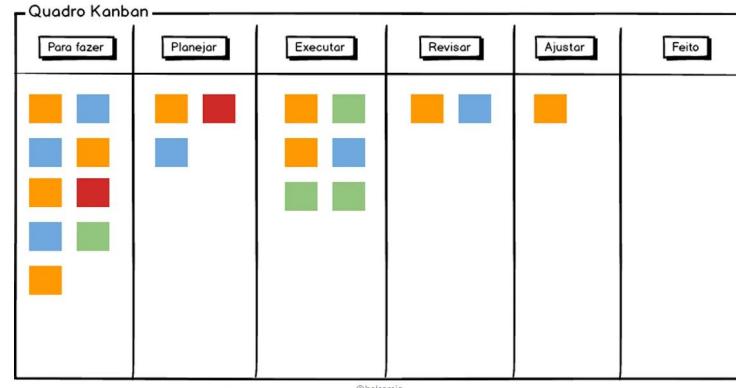
Processo Unificado Ágil

- ★ Filosofia: Sequencial para o que é amplo e iterativa para o que é particular
- ★ Atividades:
 - Modelagem
 - Implementação
 - Testes
 - Entrega
 - Configuração e gerenciamento
 - Gerenciamento de ambiente



Kanban

- ★ Significa cartão/sinalização -> Japão
- ★ Metodologia para organização de tarefas
 - To do
 - Doing
 - Done
- ★ É simples e deve ser aliado com outros frameworks para gerenciamento do projeto



Para saber mais

[Microsoft Word - AMPanfleto.doc \(agilemodeling.com\)](#)

[Agile Business Consortium](#)

[O que é Kanban? Definição e Detalhes Explicados | Kanbanize](#)

[Scrum Guide](#)

[Extreme Programming Explained: Embrace Change](#)

Aula 3

Testes no mundo ágil

// Ciclo de desenvolvimento de software e
metodologias ágeis

Objetivos

- Contextualizando a atividade de teste
- Entender os testes nas abordagens ágeis e suas diferenças dos modelos tradicionais
- Métodos e práticas de testes no Ágil

Aula 3 . Etapa 1

Contextualizando a atividade de teste

// Ciclo de desenvolvimento de software e
metodologias ágeis

A adoção de teste nos ciclos de vida do software

- ★ Para cada atividade de desenvolvimento existe uma atividade de teste
- ★ Cada nível de teste tem objetivos específicos
- ★ A análise e modelagem de testes começam durante a atividade de desenvolvimento
- ★ Participação no processo de requisitos, modelagem, refinamento...

A participação do QA na história do usuário

- ★ Histórias de usuário = requisitos funcionais + não-funcionais
- ★ Conceito 3C:
 - Cartão
 - Conversação
 - Confirmação -> Critérios de aceite
- ★ Perspectiva de quem testa difere do cliente, do PO e do desenvolvedor

Atividades envolvidas no planejamento

- ★ Análise detalhada das histórias
- ★ Determinar testabilidade da história
- ★ Criar testes de aceite
- ★ Criar tarefas para teste
- ★ Estimar esforço
- ★ Identificar aspectos funcionais e não funcionais a serem avaliados
- ★ Participar do processo de automação

Detalhando a abordagem de teste

- ★ Determinar escopo, extensão, objetivos e razões para testes
- ★ Membros que irão atuar
- ★ Ambiente e dados necessários
- ★ Tempo, dependência e pré-requisitos
- ★ Riscos envolvidos

Aula 3 . Etapa 2

Testes nas abordagens ágeis

// Ciclo de desenvolvimento de software e
metodologias ágeis

Diferenças nas abordagens de testes

- ★ As atividades de testes estão sempre relacionadas com o desenvolvimento, portanto, é importante conhecer os diversos processos e ciclos de vida e como a atividade de testes e qualidade se insere!
- ★ Cada empresa adota um processo e o customiza de acordo com necessidade
- ★ Adaptação é palavra-chave

Diferenças nas abordagens de testes

- ★ As atividades de qualidade e teste estão embutidas em cada iteração podendo ocorrer paralelismo e sobreposição com outras atividades
- ★ Cada pessoa do time tem atuação direta na validação e verificação
- ★ Foco nos testes de segurança, performance e exploratórios
- ★ Uso de automação para testes de regressão
- ★ Documentação suficiente para manutenção e garantia de qualidade

Produtos de trabalho comuns

- ★ Testes automatizados -> resultados
- ★ Planos de testes
- ★ Análise de risco
- ★ Evidências de testes manuais
- ★ Relatórios de defeitos

Níveis de teste no modelo ágil

- ★ São sobrepostos
- ★ Foco
 - Testes de unidade
 - Testes de aceite
 - Verificação
 - Validação
- ★ Uso de integração e entrega contínua + automação de testes

Status de testes no modelo ágil

- ★ Adaptação do modelo existe evolução e análise crítica para definir o que está efetivamente concluído
- ★ Atualização frequente de testes manuais e automatizados
- ★ Monitorar status de todas as atividades da equipe -> foco no feedback
- ★ Reuniões diárias para comunicação

Atitudes e habilidades para agilidade

1. Positividade e pensamento na solução com todos da equipe
2. Pensamento crítico com foco em qualidade
3. Comunicação constante com cliente
4. Feedbacks constantes
5. Avaliação de cenários que representem os critérios de aceites
6. Colaboração em tempo integral com programadores
7. Adaptação à mudança
8. Organização e planejamento

Outras atividades na equipe ágil

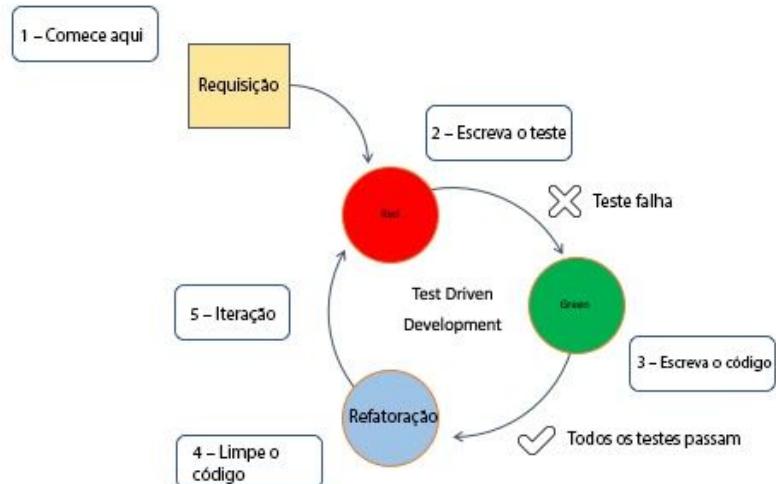
- Compreender e atualizar estratégias de teste
- Medir e informar cobertura de teste
- Garantir uso de ferramentas de forma adequada
- Gerenciar ambientes de teste e seus dados
- Relatar defeitos e gerenciá-los
- Assegurar tarefas de forma adequada e suas estimativas
- Esclarecimento contínuo de requisitos junto à equipe
- Sugerindo melhorias

Aula 3 . Etapa 3

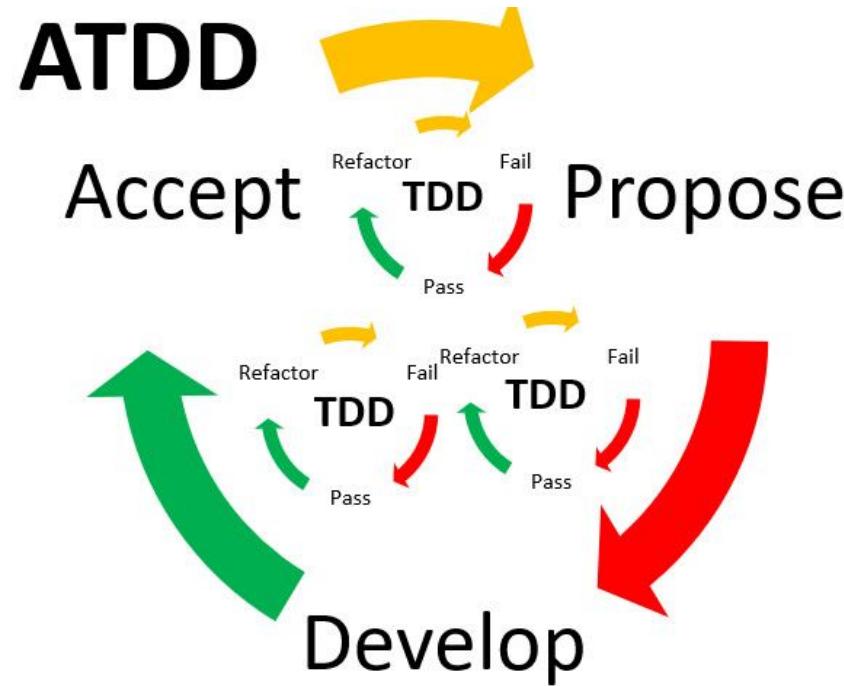
Métodos de testes no modelo ágil

// Ciclo de desenvolvimento de software e metodologias ágeis

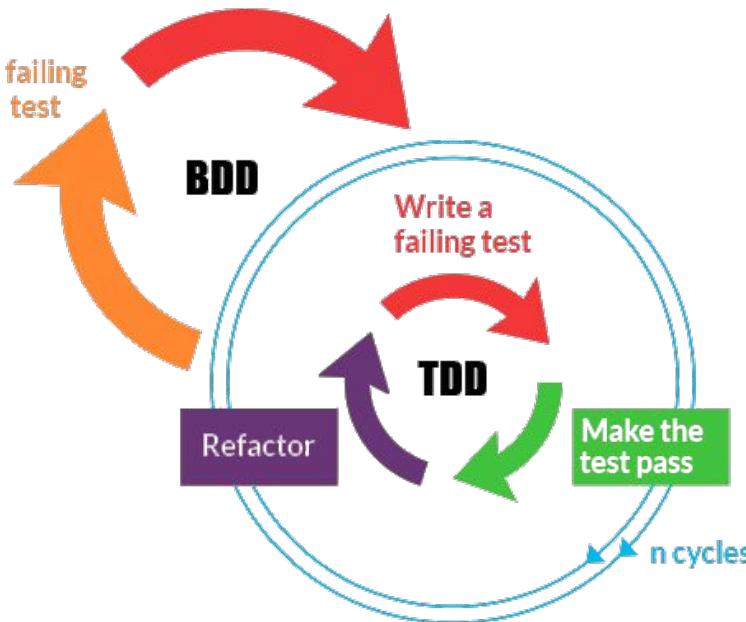
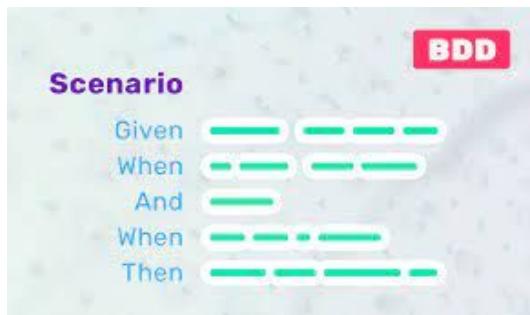
TDD - Desenvolvimento orientado por teste



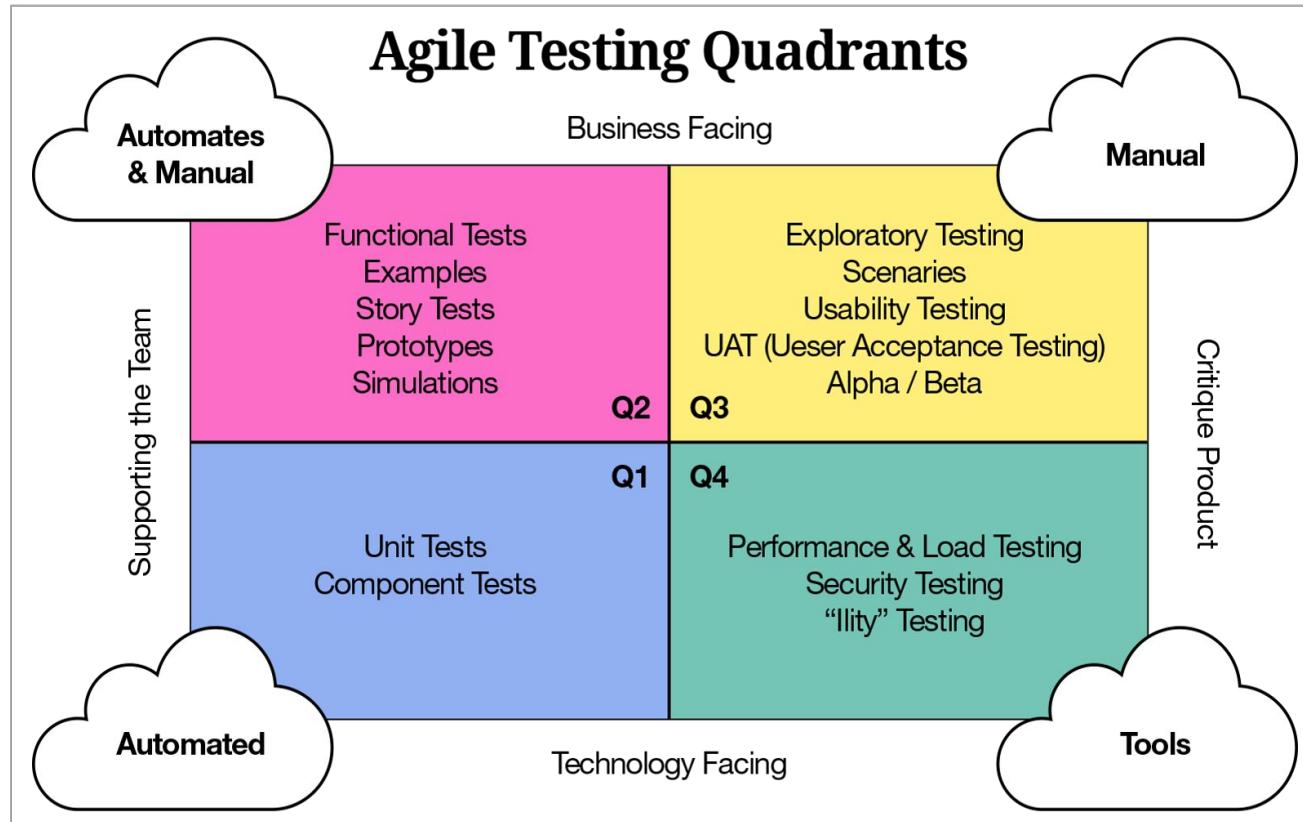
ATDD - Desenvolvimento orientado por teste de aceite

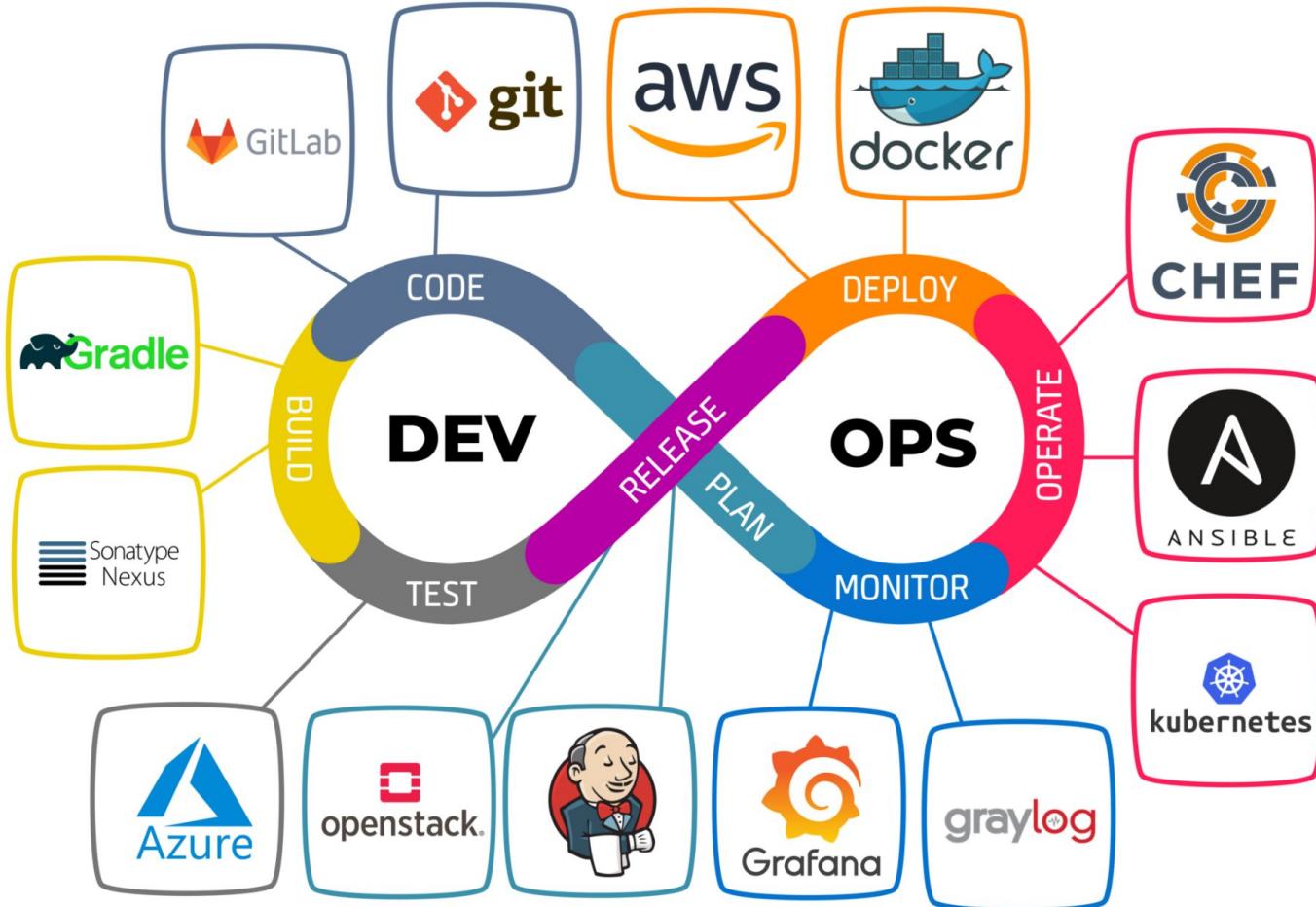


BDD- Desenvolvimento orientado a comportamento



Quadrantes de testes ágeis





Práticas úteis para testes

- ★ Teste assistido
- ★ Testes incrementais
- ★ Mapa mental
 - Estratégias
 - Cenários
 - Dados

Certificações para testes ágeis

- CTFL-AT Agile Tester
- CTFL-ATT Agile Technical Tester



Para saber mais

- ★ [TDD](#)
- ★ [ATDD](#)
- ★ [BDD na prática](#)
- ★ [CTFL-AT \(bstqb.org.br\)](#)
- ★ [CTFL-ATT \(bstqb.org.br\)](#)
- ★ [Cultura DevOps: entenda o que é quais os seus benefícios \(profissionaisti.com.br\)](#)

Ciclo de desenvolvimento de software e metodologias ágeis

Carolina Santana Louzada

Analista QA - Venturus

Percurso

Aula 1

Processos de software

Aula 2

Desenvolvimento ágil

Aula 3

Testes no mundo ágil

Dúvidas durante o curso?

- > Fórum do curso
- > Comunidade online (Discord)



O caminho da certificação CTFL

Carolina Santana Louzada

Analista QA - Venturus

Mais sobre mim

- Graduada em Engenharia de Computação- UFS
- Fazendo especialização em qualidade e desenvolvimento de software
- Qualidade de software -> automação
- Educação + tecnologia
- Jogos + música + aprender novas atividades
- LinkedIn -> [Carolina Santana Louzada | LinkedIn](#)

Objetivo do curso

Ter um panorama do portfólio de certificações da ISTQB, tendo como foco a certificação CTFL. Ver um roadmap do que é importante se aprofundar para ser aprovado na certificação e entender como é o modelo e estrutura da prova.

Pré-requisitos

- Fundamentos de qualidade de software
- Ciclo de Desenvolvimento de Software e Metodologias Ágeis

Percurso

Aula 1

Conhecendo a ISTQB e BSTQB

Aula 2

Estrutura e roadmap para aprovação na CTFL

Aula 3

Revisando conceitos importantes para a CTFL

Dúvidas durante o curso?

- > Fórum do curso
- > Comunidade online (Discord)



Aula 1

Conhecendo a ISQTB e BSQTB

// O caminho da certificação CTFL

Objetivos

- Surgimento e estrutura
- Portfólio e plano de certificações

Aula 1 . Etapa 1

Surgimento e estrutura

// O Caminho da Certificação CTFL

Surgimento da ISTQB

- Fundado em 1998
- Liderança no esquema de certificações em teste de software

Continuamente melhorar e desenvolver as profissões relacionadas a testes de software a partir da definição e manutenção de um corpo de conhecimento...

Cobertura da ISTQB

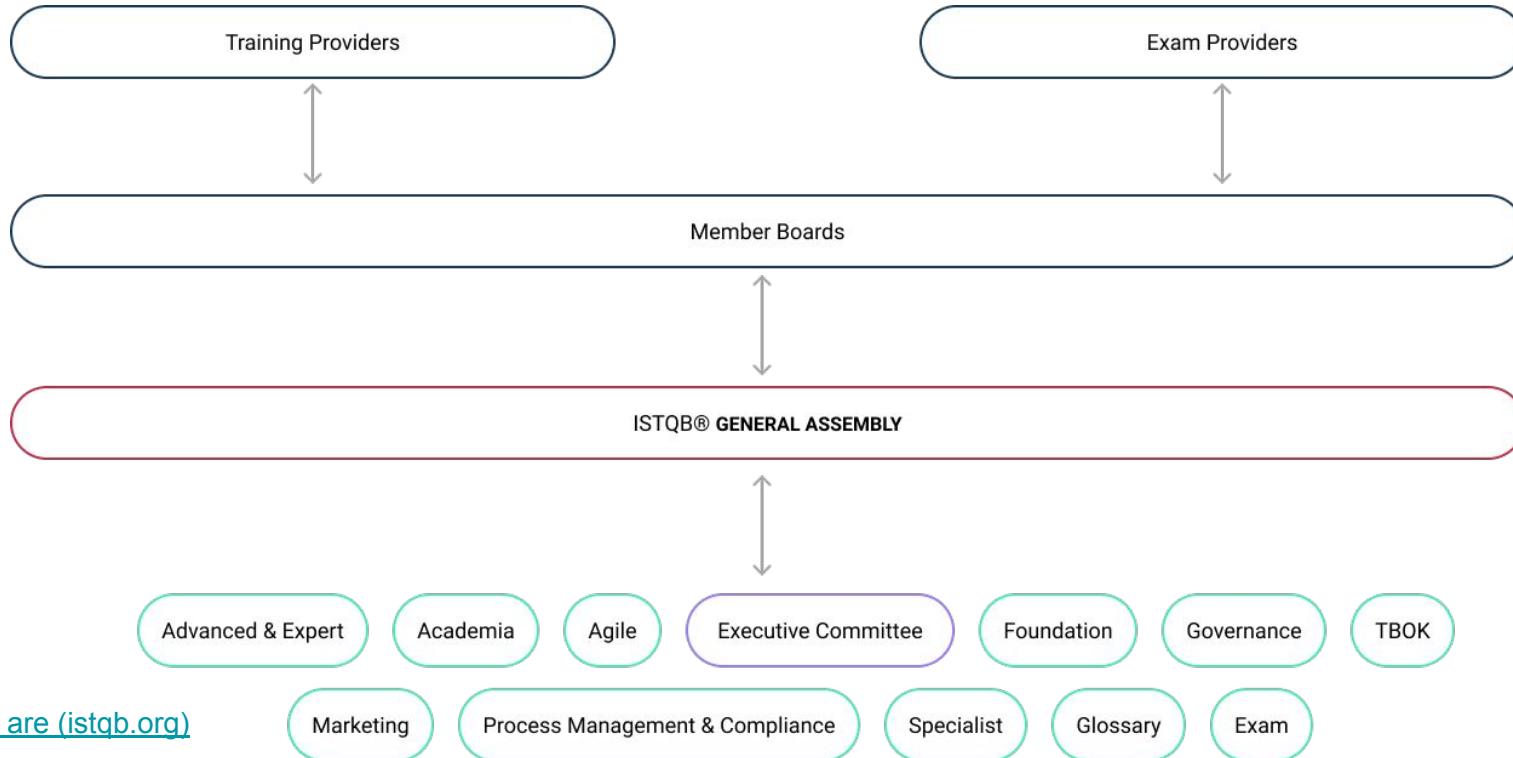


Countries covered by **Member Boards and Global Exam Providers**



Countries covered by **Global Exam Providers**

Estrutura da ISTQB



E a BSTQB?

- Brazilian Software Testing Qualifications Board - um dos Conselhos Membros do ISTQB e IREB(International Requirements Engineering Board)
- Exclusivo para o Brasil
- Mantido pelo ABRAMTI - Associação Brasileira de Melhoria em TI
- Foco em Testes de Software e Engenharia de Requisitos
- Tradução e gestão do corpo de conhecimento e realização de exames oficiais

E a BSTQB?

- Brazilian Software Testing Qualifications Board - um dos Conselhos Membros do ISTQB e IREB(International Requirements Engineering Board)
- Exclusivo para o Brazil
- Mantido pelo ABRAMTI - Associação Brasileira de Melhoria em TI
- Foco em Testes de Software e Engenharia de Requisitos
- Tradução e gestão do corpo de conhecimento e realização de exames oficiais

Modalidades de Exames

1. Exame Nacional
 - [6 datas](#) ao ano
 - presencial
2. Exame Empresarial
 - Instalações da empresa
3. Exame de Treinamento
 - Provedores de treinamento
4. Exame Acadêmico
 - Instituição acadêmica parceira
5. Exame Online
 - Parceria com a [iSQI](#)
6. Exame Especial

Aula 1 . Etapa 2

Portfólio e plano de certificações

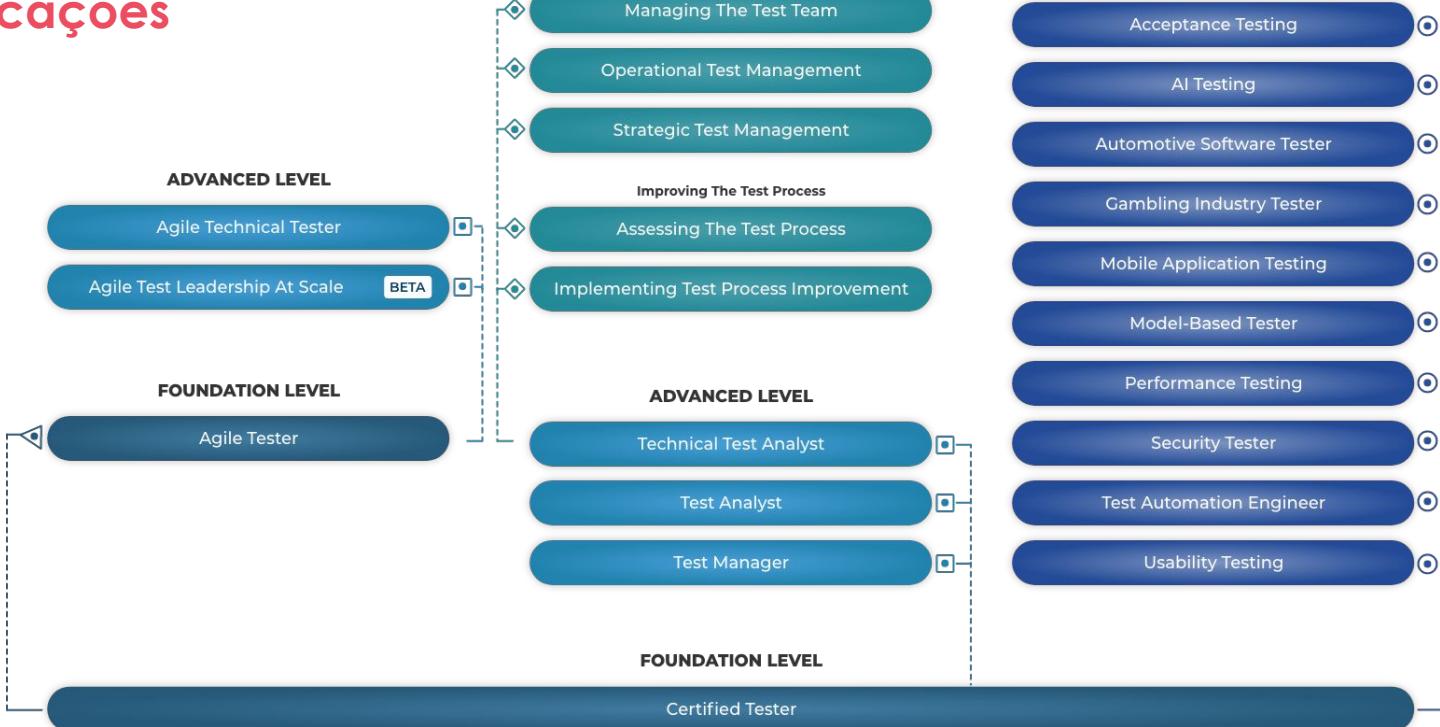
// O Caminho da Certificação CTFL

Portfólio de Certificações

AGILE

CORE

SPECIALIST

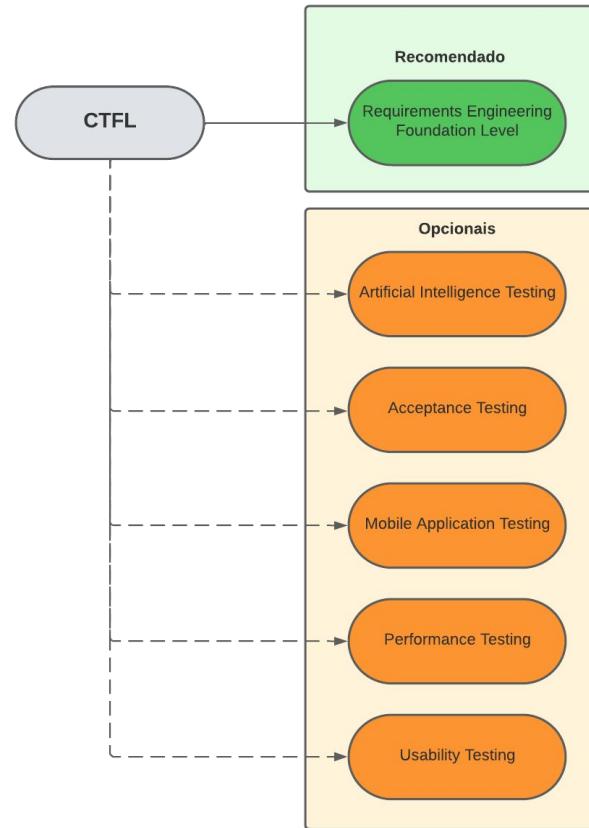


[Certifying Software Testers Worldwide - ISTQB®](#)
[International Software Testing Qualifications Board](#)

START HERE

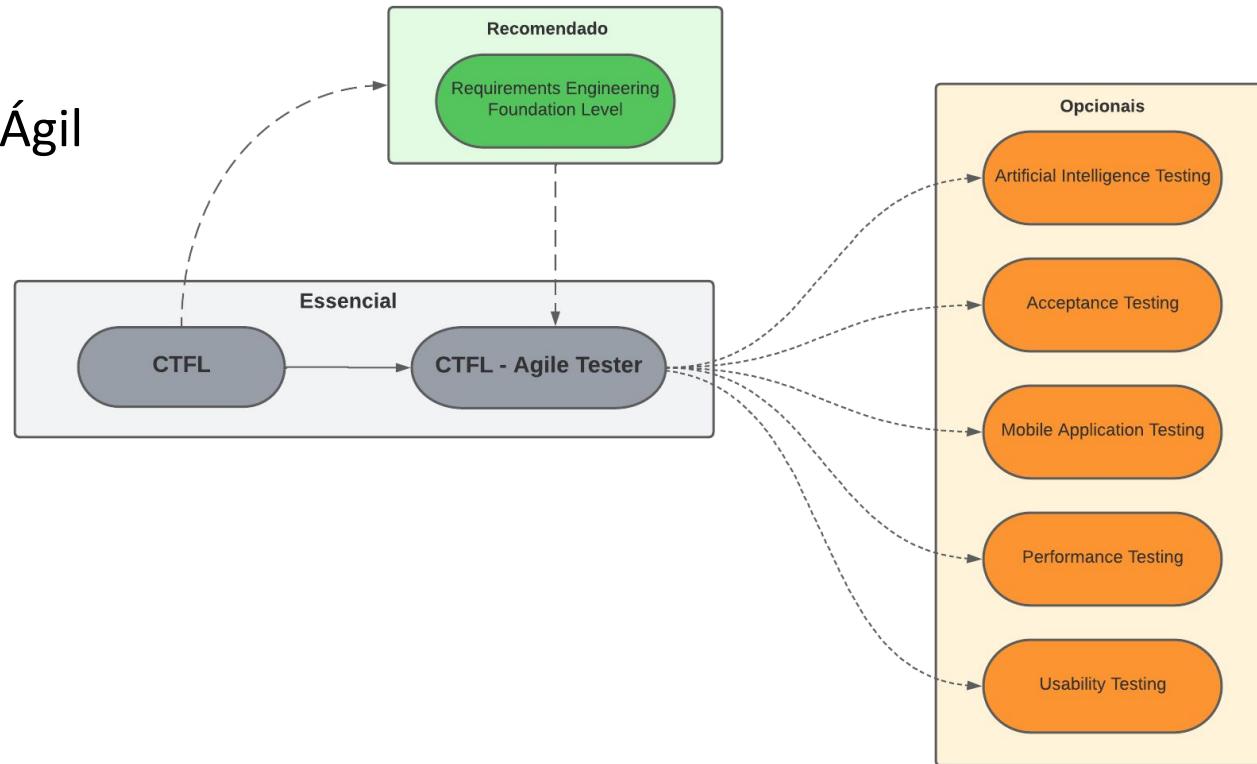
Trilhas de certificações recomendadas - BSTQB

Testador



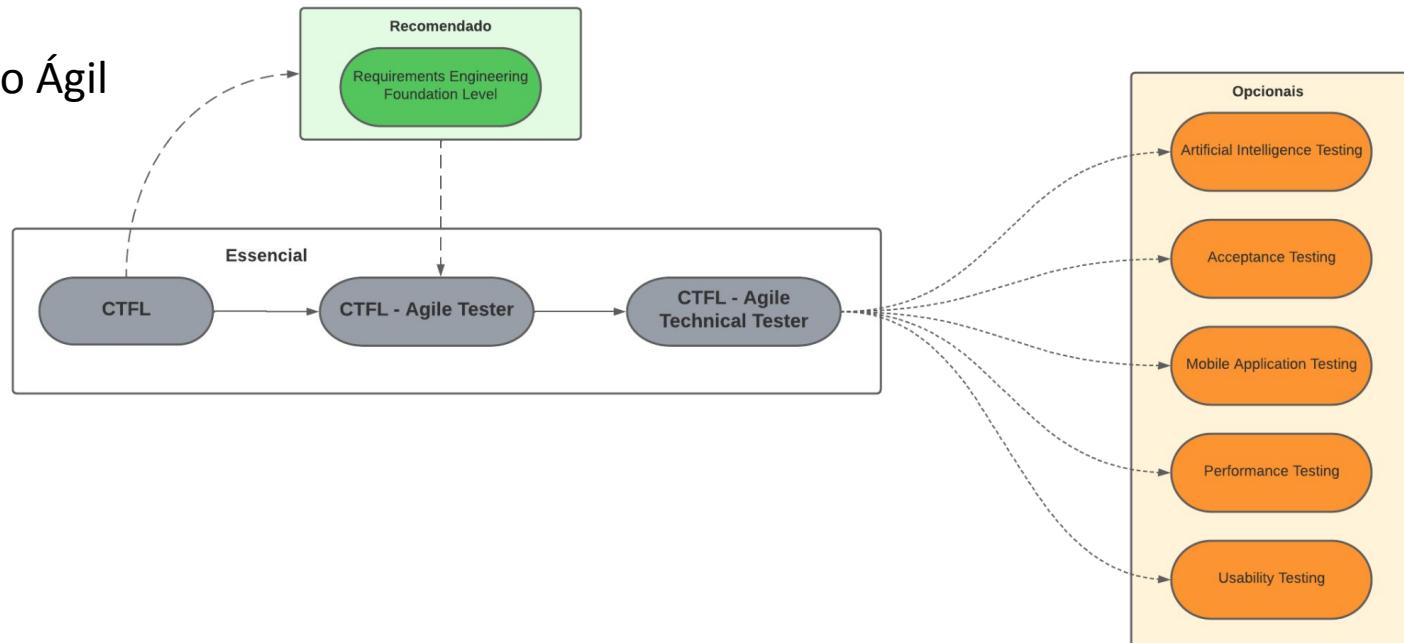
Trilhas de certificações recomendadas - BSTQB

Testador Ágil



Trilhas de certificações recomendadas - BSTQB

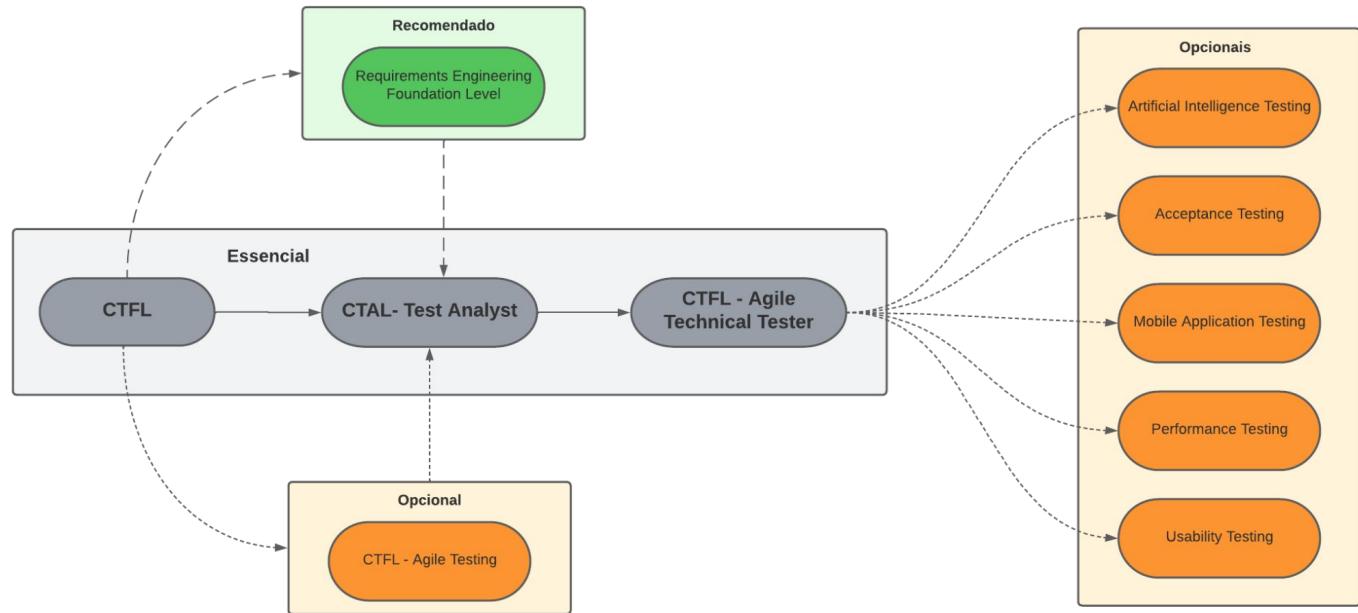
Testador Técnico Ágil



Trilhas de certificações recomendadas - BSTQB

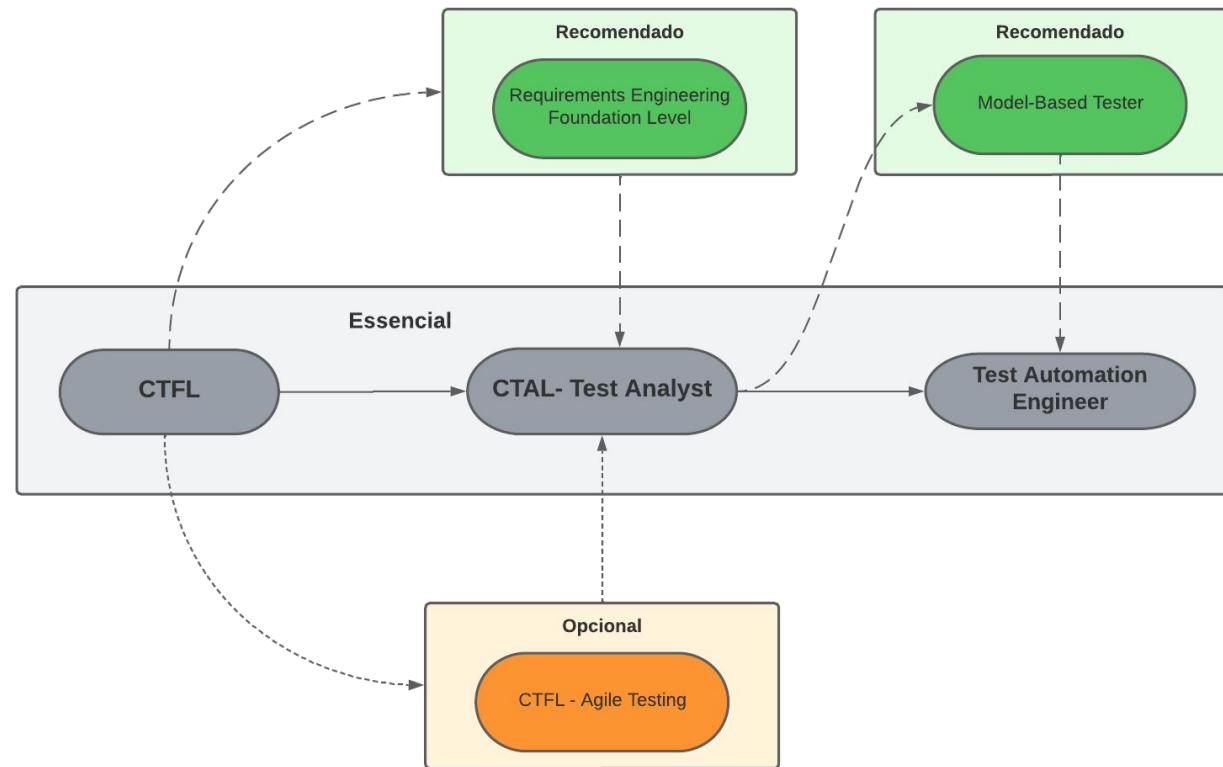
Analista de Teste
Engenheiro de Teste
Líder Técnico

Fundamental



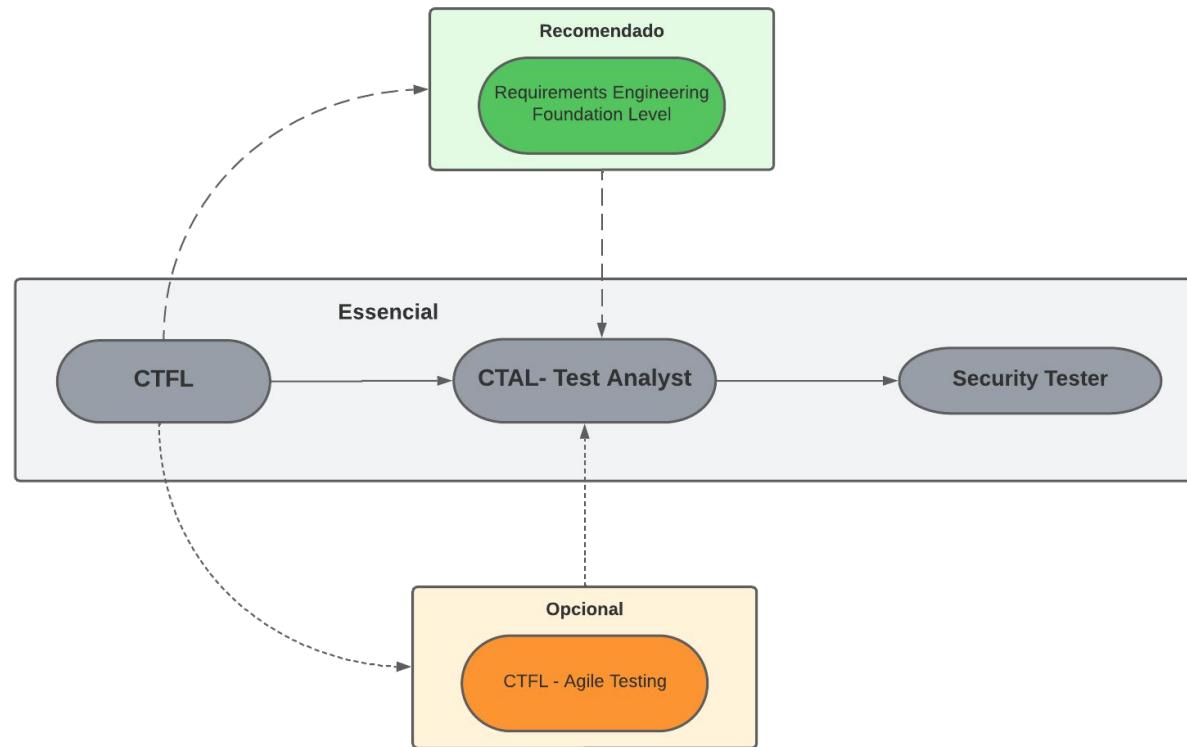
Trilhas de certificações recomendadas - BSTQB

Analista de Teste
Engenheiro de Teste
Líder Técnico



Trilhas de certificações recomendadas - BSTQB

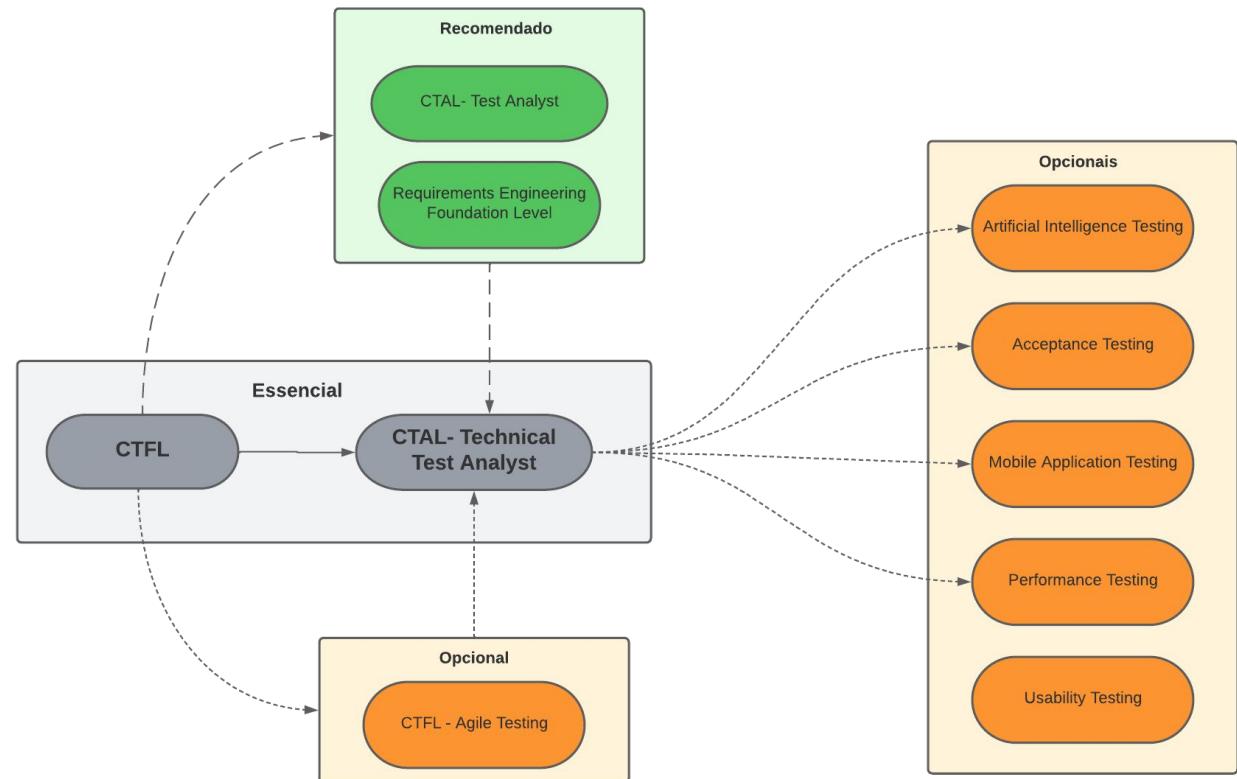
Analista de Teste
Engenheiro de Teste
Líder Técnico



Trilhas de certificações recomendadas - BSTQB

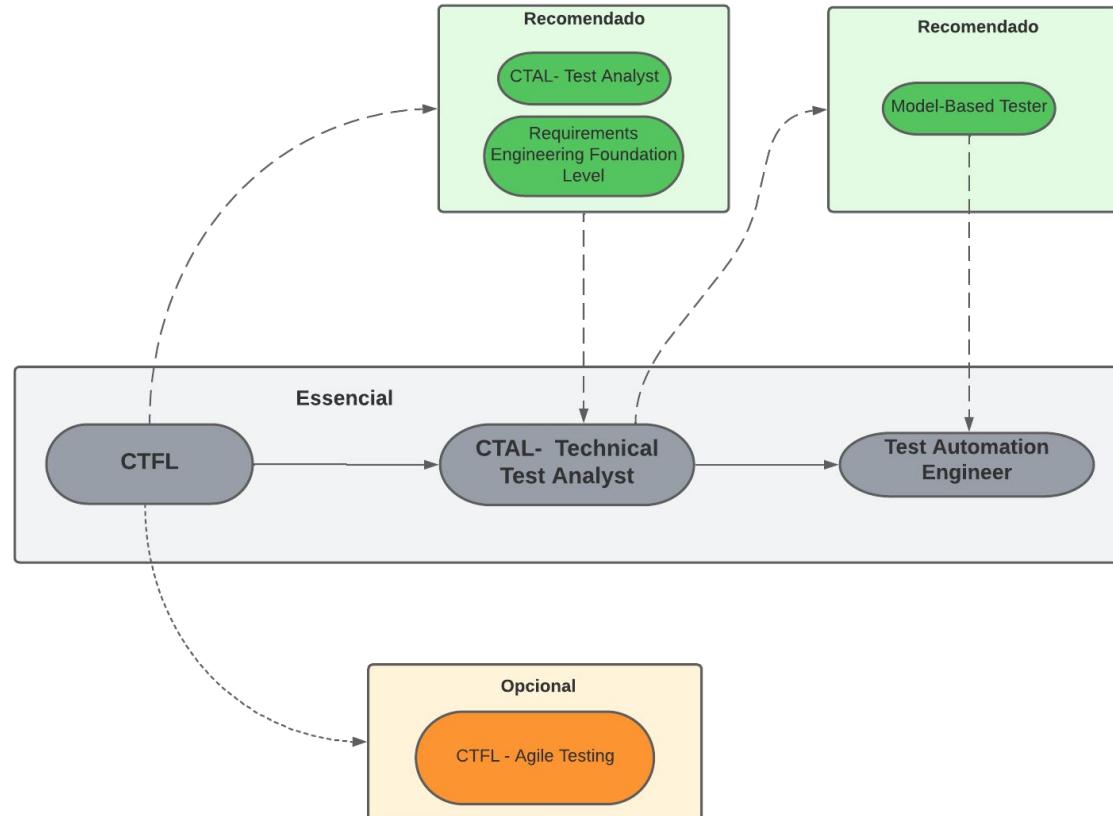
Analista Técnico de Teste
Consultor de Teste
Consultor de Qualidade

Fundamental



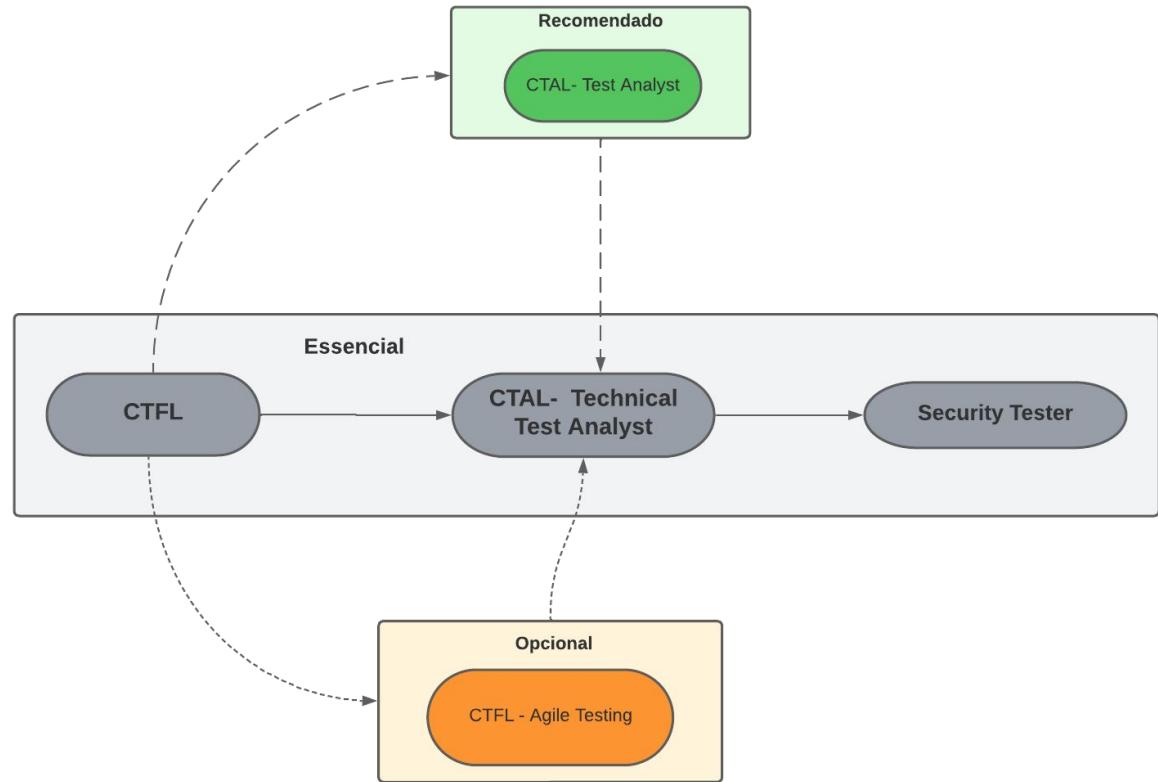
Trilhas de certificações recomendadas - BSTQB

Analista Técnico de Teste
Consultor de Teste
Consultor de Qualidade



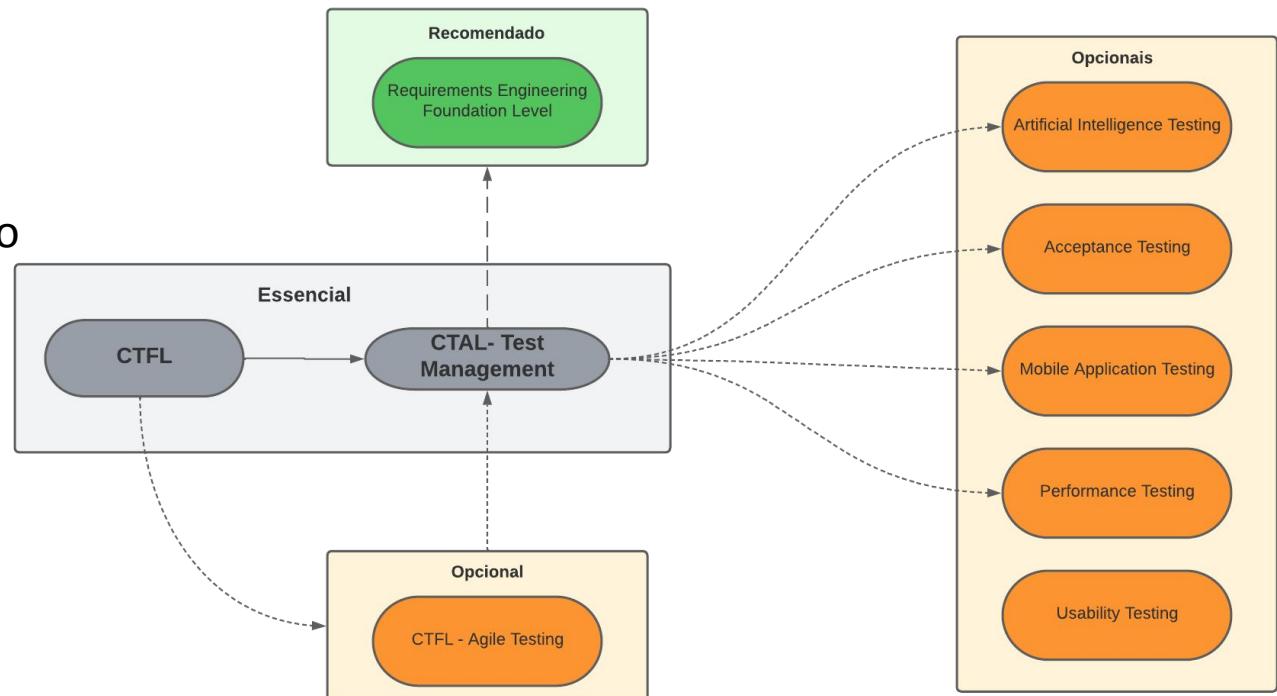
Trilhas de certificações recomendadas - BSTQB

Analista Técnico de Teste
Consultor de Teste
Consultor de Qualidade



Trilhas de certificações recomendadas - BSTQB

Gerente de Teste ou de
Qualidade
Gerente ou Líder de Projeto



Aula 2

Estrutura e roadmap para aprovação na CTFL

// O caminho da certificação CTFL

Objetivos

- Estrutura, objetivos de aprendizagem e níveis cognitivos
- Roadmap de Aprendizagem

Aula 2 . Etapa 1

Estrutura, objetivos de aprendizagem e níveis cognitivos

// O Caminho da Certificação CTFL

Explorando a certificação - CTFL

- Pré-requisito para demais certificações
- Conhecimentos base da área de testes explorando conceitos para quaisquer modelos de entrega
- Para quem?
 - ◆ testadores
 - ◆ analistas de testes
 - ◆ engenheiros de testes
 - ◆ consultores
 - ◆ gerentes
 - ◆ desenvolvedores

Explorando a certificação - CTFL

→ Estrutura do exame:

- ◆ 40 questões -> 40 pontos
- ◆ aprovação : 26 pontos
- ◆ 60 min + 15 min para não-nativos da linguagem

→ Material de estudo:

- ◆ SYLLABUS [Syllabus v3.1.1](#) or [Syllabus-PT](#)
- ◆ Exemplos de simulados fornecidos pelo próprio ISTQB/BSTQB

Objetivos de aprendizagem e níveis cognitivos

- Objetivos de aprendizagem: definições para resultados esperados e criação de níveis de certificações
- Objetivos cognitivos : Classificar os objetivos de aprendizagem
 - ◆ K1: lembrar
 - ◆ K2: entender
 - ◆ K3: aplicar
 - ◆ K4: analisar
 - ◆ K5: avaliar
 - ◆ K6 : criar

Objetivos de aprendizagem e níveis cognitivos - Exemplo

Learning Objectives for Fundamentals of Testing:

1.1 What is Testing?

- FL-1.1.1 (K1) Identify typical objectives of testing
- FL-1.1.2 (K2) Differentiate testing from debugging

1.2 Why is Testing Necessary?

- FL-1.2.1 (K2) Give examples of why testing is necessary
- FL-1.2.2 (K2) Describe the relationship between testing and quality assurance and give examples of how testing contributes to higher quality
- FL-1.2.3 (K2) Distinguish between error, defect, and failure
- FL-1.2.4 (K2) Distinguish between the root cause of a defect and its effects

1.3 Seven Testing Principles

- FL-1.3.1 (K2) Explain the seven testing principles

1.4 Test Process

- FL-1.4.1 (K2) Explain the impact of context on the test process
- FL-1.4.2 (K2) Describe the test activities and respective tasks within the test process
- FL-1.4.3 (K2) Differentiate the work products that support the test process
- FL-1.4.4 (K2) Explain the value of maintaining traceability between the test basis and test work products

1.5 The Psychology of Testing

- FL-1.5.1 (K1) Identify the psychological factors that influence the success of testing
- FL-1.5.2 (K2) Explain the difference between the mindset required for test activities and the mindset required for development activities

Estrutura Syllabus - CTFL

→ Total de no mínimo 16.75h de instrução

1 - Fundamento de teste	175 min
2 - Testes durante o ciclo de vida de desenvolvimento do software	100 min
3 - Teste estático	135 min
4 - Técnicas de teste	330 min
5 - Gerenciamento de testes	225 min
6 - Ferramentas de suporte a testes	40 min

Estrutura Syllabus - CTFL

→ Tempo para resolução por nível cognitivo

K-nível	nº de questões	tempo de resolução	Total de tempo
K1	8	1 min	8 min
K2	24	1 min	24 min
K3	8	3 min	24
Total	40	-	56 min

Estrutura Syllabus - CTFL

- Número de questões por capítulo

Capítulo	nº de questões
1	8
2	5
3	5
4	11
5	9
6	2
total	40

O que devo ser capaz de fazer?

- Saber usar o vocabulário comum para testes de software
- Entender conceitos fundamentais para testes de software
- Entender práticas e modelos para desenvolvimento e testes de software
- Como contribuir para revisões
- Interpretar, executar e reportar testes
- Princípios para gerenciamento de testes
- Entendimento e comunicação de reportes de defeitos

O que devo ser capaz de fazer?

- Estabelecer técnicas e estratégias diferentes mediante o contexto
- Entender o valor dos testes de software para os stakeholders
- Entender como as atividades de testes se alinham com os objetivos do projeto
- Dar suporte na seleção e processo de uso de ferramentas de teste.

Aula 2 . Etapa 2

Roadmap de aprendizagem

// O Caminho da Certificação CTFL

Explorando o conteúdo programático - CTFL



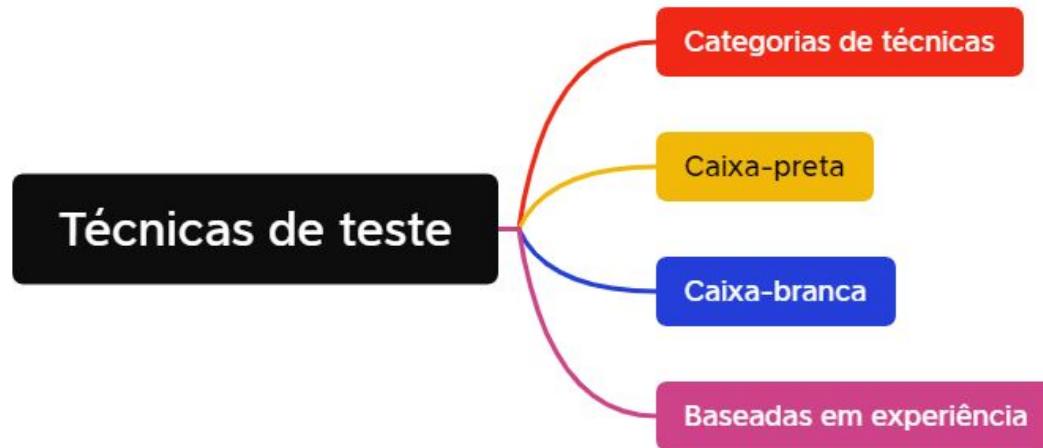
Explorando o conteúdo programático - CTFL



Explorando o conteúdo programático - CTFL



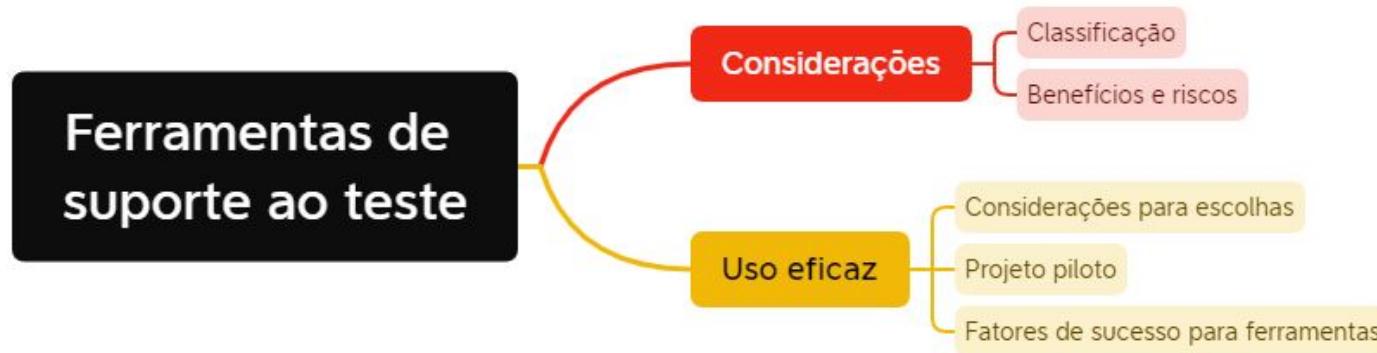
Explorando o conteúdo programático - CTFL



Explorando o conteúdo programático - CTFL



Explorando o conteúdo programático - CTFL



Aula 3

Revisando conceitos importantes para a CTFL

// O caminho da certificação CTFL

Objetivos

- Revisão de questões - Fundamentos de Testes
- Revisão de questões - Testes durante o ciclo de vida
- Revisão de questões - Testes estáticos
- Revisão de questões - Técnicas de teste
- Revisão de questões - Gerenciamento de teste
- Revisão de questões - Ferramenta de apoio ao teste

Aula 3 . Etapa 1

Fundamentos de testes

// O caminho para certificação CTFL

Questões relacionadas

1. Qual das seguintes respostas descreve uma condição de teste?
 - a) Uma característica distinta de um componente ou sistema
 - b) Um aspecto testável de um componente ou sistema identificado como base para os testes
 - c) O grau em que um produto de software fornece funções que atendem às necessidades declaradas e implícitas quando o software é utilizado sob condições específicas
 - d) Casos de teste projetados para executar combinações de condições e ações resultantes delas

Questões relacionadas

1. Qual das seguintes respostas descreve uma condição de teste?
 - a) Uma característica distinta de um componente ou sistema
 - b) Um aspecto testável de um componente ou sistema identificado como base para os testes
 - c) O grau em que um produto de software fornece funções que atendem às necessidades declaradas e implícitas quando o software é utilizado sob condições específicas
 - d) Casos de teste projetados para executar combinações de condições e ações resultantes delas

Funcionalidade

Adequação de
funcionalidade

Tabela de
decisões

Planejamento, monitoramento e controle do teste, análise do teste, modelagem, implementação, execução e conclusão

Questões relacionadas

2 - Qual das seguintes declarações descreve corretamente a diferença entre teste e depuração?

- a) Os testes identificam a fonte dos defeitos; a depuração analista os defeitos e propõe atividades de prevenção
- b) Os testes dinâmicos mostram falhas causadas por defeitos; a depuração elimina os defeitos, que são fontes de falhas
- c) Os testes não removem as falhas; mas a depuração remove os defeitos que causam as falhas
- d) Os testes dinâmicos previnem as causas das falhas; a depuração remove as falhas

Questões relacionadas

2 - Qual das seguintes declarações descreve corretamente a diferença entre teste e depuração?

- a) Os testes **identificam a fonte dos defeitos**; a depuração analisa os defeitos e **propõe atividades de prevenção**
- b) **Os testes dinâmicos mostram falhas causadas por defeitos; a depuração elimina os defeitos, que são fontes de falhas**
- c) Os testes não removem as falhas; mas a depuração remove os **defeitos que causam as falhas**
- d) Os testes dinâmicos **previnem** as causas das falhas; a depuração remove as falhas

Questões relacionadas

3 - Como resultado da análise de risco, mais testes estão sendo direcionados para aquelas áreas do sistema em teste onde os testes iniciais encontraram mais defeitos do que a média. Qual dos seguintes princípios de teste está sendo aplicado?

- a) Cuidado com o paradoxo do pesticida
- b) Os testes são dependentes do contexto
- c) A ausência de erros é uma falácia
- d) Defeitos agrupados

Questões relacionadas

3 - Como resultado da análise de risco, mais testes estão sendo direcionados para aquelas áreas do sistema em teste onde os testes iniciais encontraram mais defeitos do que a média. Qual dos seguintes princípios de teste está sendo aplicado?

- a) Cuidado com o paradoxo do pesticida
- b) Os testes são dependentes do contexto
- c) A ausência de erros é uma falácia
- d) Defeitos agrupados**

Questões relacionadas

3 - Como resultado da análise de risco, mais testes estão sendo direcionados para aquelas áreas do sistema em teste onde os testes iniciais encontraram mais defeitos do que a média. Qual dos seguintes princípios de teste está sendo aplicado?

- a) Cuidado com o paradoxo do pesticida
- b) Os testes são dependentes do contexto
- c) A ausência de erros é uma falácia
- d) Defeitos agrupados**

- ★ O teste mostra a presença de defeitos e não sua ausência
- ★ Testes exaustivos são impossíveis
- ★ Teste inicial economiza tempo e dinheiro

Questões relacionadas

4 - Combine os seguintes produtos de trabalho de teste (1-4) com a descrição correta (A-D)

- (1) Conjunto de teste
- (2) Caso de teste
- (3) Roteiro de teste
- (4) Carta de teste

- (a) Um conjunto de scripts de teste a serem executados em uma execução de teste específica
- (b) Um conjunto de instruções para a execução de um teste
- (c) Contém os resultados esperados
- (d) Documentação das atividades de teste em testes exploratórios baseados em sessões

- a) 1A, 2C, 3B, 4D
- b) 1D, 2B, 3A, 4C
- c) 1A, 2C, 3D, 4B
- d) 1D, 2C, 3B, 4A

Questões relacionadas

4 - Combine os seguintes produtos de trabalho de teste (1-4) com a descrição correta (A-D)

- (1) Conjunto de teste
 - (2) Caso de teste
 - (3) Roteiro de teste
 - (4) Carta de teste
- (a) Um conjunto de scripts de teste a serem executados em uma execução de teste específica
 - (b) Um conjunto de instruções para a execução de um teste
 - (c) Contém os resultados esperados
 - (d) Documentação das atividades de teste em testes exploratórios baseados em sessões

a) 1A, 2C, 3B, 4D

b) 1D, 2B, 3A, 4C

c) 1A, 2C, 3D, 4B

d) 1D, 2C, 3B, 4A

Aula 3 . Etapa 2

Teste durante o ciclo de vida de desenvolvimento de software

// O Caminho da Certificação CTFL

Questões relacionadas

1 - Qual das seguintes declarações sobre tipos e níveis de teste é CORRETA?

- a) Os testes funcionais e não funcionais podem ser realizados nos níveis de teste do sistema e de aceitação, enquanto o teste caixa-branca é restrito aos testes de componentes e de integração.
- b) Os testes funcionais podem ser realizados em qualquer nível de teste, enquanto o teste caixa-branca é restrito ao teste de componentes
- c) É possível realizar testes funcionais, não-funcionais e caixa-branca em qualquer nível de teste
- d) Os testes funcionais e não funcionais podem ser realizados em qualquer nível de teste, enquanto os testes caixa-branca são restritos aos testes de componente e integração.

Questões relacionadas

1 - Qual das seguintes declarações sobre tipos e níveis de teste é CORRETA?

- a) Os testes funcionais e não funcionais podem ser realizados nos níveis de teste do sistema e de aceitação, enquanto o teste caixa-branca é restrito aos testes de componentes e de integração.
- b) Os testes funcionais podem ser realizados em qualquer nível de teste, enquanto o teste caixa-branca é restrito ao teste de componentes
- c) É possível realizar testes funcionais, não-funcionais e caixa-branca em qualquer nível de teste
- d) Os testes funcionais e não funcionais podem ser realizados em qualquer nível de teste, enquanto os testes caixa-branca são restritos aos testes de componente e integração.

Questões relacionadas

2- Qual das seguintes opções é VERDADEIRA?

- a) O objetivo do teste de regressão é verificar se a correção foi implementada com sucesso, enquanto o objetivo do teste de confirmação é confirmar que a correção não tem efeitos colaterais
- b) O objetivo do teste de regressão é detectar efeitos colaterais não intencionais, enquanto o objetivo do teste de confirmação é verificar se o sistema ainda está funcionando em um novo ambiente
- c) O objetivo do teste de regressão é detectar efeitos colaterais não intencionais, enquanto o objetivo do teste de confirmação é verificar se o defeito original foi corrigido
- d) O objetivo do teste de regressão é verificar se a nova funcionalidade está funcionando, enquanto o objetivo do teste de confirmação é verificar se o defeito original foi corrigido

Questões relacionadas

2- Qual das seguintes opções é VERDADEIRA?

- a) O objetivo do teste de regressão é verificar se a correção foi implementada com sucesso, enquanto o objetivo do teste de confirmação é confirmar que a correção não tem efeitos colaterais
- b) O objetivo do teste de regressão é detectar efeitos colaterais não intencionais, enquanto o objetivo do teste de confirmação é verificar se o sistema ainda está funcionando em um novo ambiente
- c) O objetivo do teste de regressão é detectar efeitos colaterais não intencionais, enquanto o objetivo do teste de confirmação é verificar se o defeito original foi corrigido
- d) O objetivo do teste de regressão é verificar se a nova funcionalidade está funcionando, enquanto o objetivo do teste de confirmação é verificar se o defeito original foi corrigido

Questões relacionadas

3 - Dado que os testes que estão sendo realizados têm os seguintes atributos:

- Com base nas especificações da interface
- Focado em encontrar falhas na comunicação
- A abordagem de teste utiliza tanto tipos de teste funcionais quanto estruturais

Qual dos seguintes níveis de teste é o MAIS provável de ser realizado?

- a) Teste de integração
- b) Teste de aceitação
- c) Teste do sistema
- d) Teste de componentes

Questões relacionadas

3 - Dado que os testes que estão sendo realizados têm os seguintes atributos:

- Com base nas especificações da interface
- Focado em encontrar falhas na comunicação
- A abordagem de teste utiliza tanto tipos de teste funcionais quanto estruturais

Qual dos seguintes níveis de teste é o MAIS provável de ser realizado?

- a) Teste de integração
- b) Teste de aceitação
- c) Teste do sistema
- d) Teste de componentes



Questões relacionadas

4 - Qual dos seguintes itens NÃO deve ser um gatilho para testes de manutenção?

- a) Decisão de testar a possibilidade de manutenção do software
- b) Decisão de testar o sistema após a migração para uma nova plataforma operacional
- c) Decisão de testar se os dados arquivados são possíveis de serem recuperados.
- d) Decisão de testar após “hot fixes”

Questões relacionadas

4 - Qual dos seguintes itens NÃO deve ser um gatilho para testes de manutenção?

- a) Decisão de testar a possibilidade de manutenção do software
- b) Decisão de testar o sistema após a migração para uma nova plataforma operacional
- c) Decisão de testar se os dados arquivados são possíveis de serem recuperados.
- d) Decisão de testar após “hot fixes”

Aula 3 . Etapa 3

Teste estático

// O Caminho da Certificação CTFL

Questões relacionadas

1 - Quais das seguintes afirmações sobre testes estáticos são as mais verdadeiras?

- a) Os testes estáticos são uma forma mais barata de detectar e remover defeitos.
- b) Os testes estáticos tornam os testes dinâmicos mais desafiadores.
- c) Os testes estáticos permitem encontrar problemas de tempo de execução no início do ciclo de vida.
- d) Ao testar um sistema crítico de segurança, os testes estáticos têm menos valor porque os dinâmicos encontram melhor os defeitos.

Questões relacionadas

1 - Quais das seguintes afirmações sobre testes estáticos são as mais verdadeiras?

- a) Os testes estáticos são uma forma mais barata de detectar e remover defeitos.
- b) Os testes estáticos tornam os testes dinâmicos mais desafiadores.
- c) Os testes estáticos permitem encontrar problemas de tempo de execução no início do ciclo de vida.
- d) Ao testar um sistema crítico de segurança, os testes estáticos têm menos valor porque os dinâmicos encontram melhor os defeitos.

Questões relacionadas

2 - Qual das seguintes funções e responsabilidades se encaixa corretamente em uma revisão formal?

- a) Gerente - Decide sobre a execução das revisões
- b) Líder de revisão - Assegura o funcionamento eficaz das reuniões de revisão
- c) Redator - Corrige defeitos no produto de trabalho em revisão
- d) Moderador - Monitora a relação custo-benefício contínua

Questões relacionadas

2 - Qual das seguintes funções e responsabilidades se encaixa corretamente em uma revisão formal?

- a) **Gerente - Decide sobre a execução das revisões**
- b) Líder de revisão - Assegura o funcionamento eficaz das reuniões de revisão
- c) Redator - Corrige defeitos no produto de trabalho em revisão
- d) Moderador - Monitora a relação custo-benefício contínua

Funções e responsabilidades em revisão formal

- ★ Autor
 - Cria o produto de trabalho sobre revisão
 - Corrige os defeitos no produto de trabalho sobre revisão
- ★ Gestor
 - Responsável pelo planejamento da revisão
 - Decidir sobre a execução das revisões
 - Atribuir pessoal, orçamento e tempo
 - Monitorar a rentabilidade contínua
- ★ Facilitador
 - Garantir a execução eficaz das reuniões de revisão
 - Mediar, se necessário, entre os pontos de vista
- ★ Líder de revisão
 - Assumir a responsabilidade geral pela revisão
 - Decidir quem será envolvido e organizar quando e onde acontecerá a revisão

★ Revisor

- Especialistas, stakeholders ou outros da equipe com formação técnica
- Identificar possíveis defeitos no produto
- Pode representar diferentes perspectivas

★ Redator

- Coletar possíveis defeitos encontrados durante revisão
- Registrar novos defeitos em potencial, pontos em aberto e decisões da reunião de revisão

Questões relacionadas

3 - Você está lendo uma história de usuário no backlog do produto para se preparar para uma reunião com o PO e desenvolvedor, aparentemente não há defeitos ou erros de acordo com a análise dessa história. Qual das sentenças é verdadeira sobre essa atividade?

- a) Não é um teste estático porque envolve a execução do objeto de teste
- b) Não é um teste estático porque é sempre executado com uma ferramenta
- c) Não é um teste estático porque qualquer defeito encontrado poderia ser encontrado de forma mais barata no teste dinâmico
- d) É um teste estático porque não envolve a execução do objeto de teste

Questões relacionadas

3 - Você está lendo uma história de usuário no backlog do produto para se preparar para uma reunião com o PO e desenvolvedor, aparentemente não há defeitos ou erros de acordo com a análise dessa história. Qual das sentenças é verdadeira sobre essa atividade?

- a) Não é um teste estático porque envolve a execução do objeto de teste
- b) Não é um teste estático porque é sempre executado com uma ferramenta
- c) Não é um teste estático porque qualquer defeito encontrado poderia ser encontrado de forma mais barata no teste dinâmico
- d) **É um teste estático porque não envolve a execução do objeto de teste**

Aula 3 . Etapa 4

Técnicas de teste

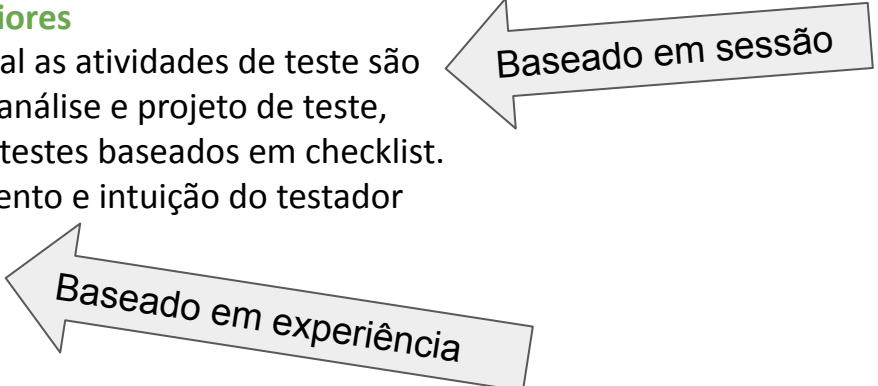
// O Caminho da Certificação CTFL

Questões relacionadas

- 1 - Quais dos itens a seguir fornece a melhor descrição de testes exploratórios?
- a) Uma prática de teste na qual uma investigação aprofundada dos antecedentes do objeto de teste é utilizada para identificar potenciais pontos fracos que são examinados pelos casos de teste
 - b) Uma abordagem aos testes em que os testadores projetam e executam dinamicamente testes baseados em seu conhecimento, exploração do item de teste e nos resultados dos testes anteriores
 - c) Uma abordagem de projeto de teste na qual as atividades de teste são planejadas como sessões ininterruptas de análise e projeto de teste, frequentemente usadas em conjunto com testes baseados em checklist.
 - d) Testes baseados na experiência, conhecimento e intuição do testador

Questões relacionadas

- 1 - Quais dos itens a seguir fornece a melhor descrição de testes exploratórios?
- a) Uma prática de teste na qual uma **investigação aprofundada** dos antecedentes do objeto de teste é utilizada para identificar potenciais pontos fracos que são examinados pelos casos de teste
 - b) **Uma abordagem aos testes em que os testadores projetam e executam dinamicamente testes baseados em seu conhecimento, exploração do item de teste e nos resultados dos testes anteriores**
 - c) Uma abordagem de projeto de teste na qual as atividades de teste são planejadas como sessões ininterruptas de análise e projeto de teste, frequentemente usadas em conjunto com testes baseados em checklist.
 - d) Testes baseados na experiência, conhecimento e intuição do testador



Baseado em sessão

Baseado em experiência

Questões relacionadas

2 - Qual declaração sobre a relação entre a cobertura de instruções e a cobertura de decisões é verdadeira

- a) 100% de cobertura de decisão também garante 100% de cobertura de instrução
- b) 100% de cobertura de declaração também garante 100% de cobertura de decisão
- c) 50% de cobertura de decisão também garante 50 % de cobertura de instrução
- d) A cobertura de decisão nunca pode chegar a 100%

Questões relacionadas

2 - Qual declaração sobre a relação entre a cobertura de instruções e a cobertura de decisões é verdadeira

- a) **100% de cobertura de decisão também garante 100% de cobertura de instrução**
- b) 100% de cobertura de declaração também garante 100% de cobertura de decisão
- c) 50% de cobertura de decisão também garante 50 % de cobertura de instrução
- d) A cobertura de decisão nunca pode chegar a 100%

Questões relacionadas

3 - Um sistema de controle de velocidade e relatórios tem as seguintes características:

- Se você dirigir a 50km/h, nada vai acontecer
- Se você dirigir mais rápido que 50 km/h, mas não mais que 55 km/h, você será avisado
- Se você dirigir mais rápido que 55 km/h, mas não mais que 60 km/h, você será multado
- Se você dirigir a mais de 60 km/h, sua carteira de habilitação será suspensa. - A velocidade em km/h está disponível para o sistema como um valor inteiro.

Qual seria o conjunto mais provável de valores identificado pela aplicação da análise de valores limite?

- a) 0,49,50,54,59,60
- b) 50,55,60
- c) 49,50,54,55,60,62
- d) 50,51,55,56,60,61

Questões relacionadas

3 - Um sistema de controle de velocidade e relatórios tem as seguintes características:

- Se você dirigir a 50km/h, nada vai acontecer
- Se você dirigir mais rápido que 50 km/h, mas não mais que 55 km/h, você será avisado
- Se você dirigir mais rápido que 55 km/h, mas não mais que 60 km/h, você será multado
- Se você dirigir a mais de 60 km/h, sua carteira de habilitação será suspensa. - A velocidade em km/h está disponível para o sistema como um valor inteiro.

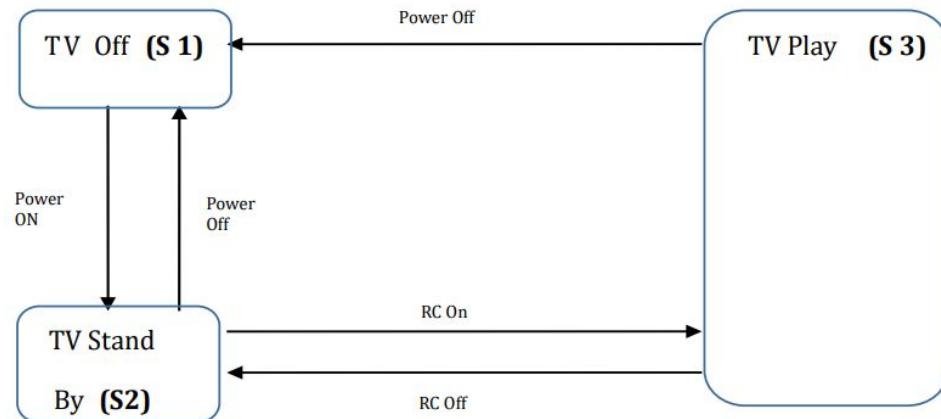
Qual seria o conjunto mais provável de valores identificado pela aplicação da análise de valores limite?

- a) 0,49,50,54,59,60
- b) 50,55,60
- c) 49,50,54,55,60,62
- d) **50,51,55,56,60,61**

Questões relacionadas

4 - Qual das seguintes afirmações sobre o diagrama de transição de estado dado e tabela de casos de teste é VERDADEIRA?

- a) Os casos em questão cobrem transições válidas e inválidas no diagrama de transição
- a) Os casos em questão representam todas as transições válidas possíveis no diagrama
- c) Os casos em questão representam algumas das transições válidas
- d) Os casos em questão representam pares de transições no diagrama de transição.

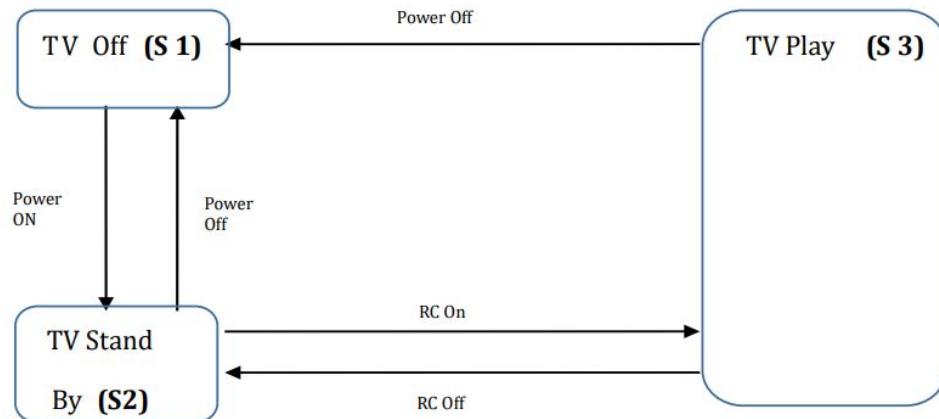


Caso de Teste	1	2	3	4	5
Estado inicial	S1	S2	S2	S3	S3
Entrada	Power On	Power Off	RC On	RC Off	Power Off
Saída esperada	S2	S1	S3	S2	S1

Questões relacionadas

4 - Qual das seguintes afirmações sobre o diagrama de transição de estado dado e tabela de casos de teste é VERDADEIRA?

- a) Os casos em questão cobrem transições válidas e inválidas no diagrama de transição
- a) Os casos em questão representam todas as transições válidas possíveis no diagrama**
- c) Os casos em questão representam algumas das transições válidas
- d) Os casos em questão representam pares de transições no diagrama de transição.



Caso de Teste	1	2	3	4	5
Estado inicial	S1	S2	S2	S3	S3
Entrada	Power On	Power Off	RC On	RC Off	Power Off
Saída esperada	S2	S1	S3	S2	S1

Aula 3 . Etapa 5

Gerenciamento de testes

// O Caminho da Certificação CTFL

Questões relacionadas

1 - Qual das seguintes declarações MELHOR descreve como as tarefas são divididas entre o gerente de testes e o testador?

- a) O gerente de testes planeja as atividades de teste e escolhe os padrões a serem seguidos, enquanto o testador escolhe as ferramentas e estabelece as diretrizes de uso das ferramentas.
- b) O gerente de testes planeja, coordena e controla as atividades de teste, enquanto o testador automatiza os testes.
- c) O gerente de testes planeja, monitora e controla as atividades de teste, enquanto o testador projeta os testes e decide sobre a liberação do objeto de teste
- d) O gerente de testes planeja e organiza os testes e especifica os casos de teste, enquanto o testador executa os testes.

Questões relacionadas

1 - Qual das seguintes declarações MELHOR descreve como as tarefas são divididas entre o gerente de testes e o testador?

- a) O gerente de testes planeja as atividades de teste e escolhe os padrões a serem seguidos, enquanto o testador **escolhe as ferramentas** e estabelece as diretrizes de uso das ferramentas.
- b) **O gerente de testes planeja, coordena e controla as atividades de teste, enquanto o testador automatiza os testes.**
- c) O gerente de testes planeja, monitora e controla as atividades de teste, enquanto o testador projeta os testes e **decide sobre a liberação do objeto de teste**
- d) O gerente de testes planeja e organiza os testes e **especifica os casos de teste**, enquanto o testador executa os testes.

Questões relacionadas

2 - Qual das seguintes métricas seria a mais útil para monitorar durante a execução do teste?

- a) Porcentagem de casos de teste executados
- b) Número médio de testadores envolvidos na execução
- c) Cobertura de requisitos por código fonte
- d) Porcentagem de casos de teste já criados e revisados

Questões relacionadas

2 - Qual das seguintes métricas seria a mais útil para monitorar durante a execução do teste?

- a) **Porcentagem de casos de teste executados**
- b) Número médio de testadores envolvidos na execução
- c) Cobertura de requisitos por código fonte
- d) Porcentagem de casos de teste já criados e revisados

Questões relacionadas

3 - Você está realizando testes de sistema de reserva de trens. Com base nos casos de teste realizados, você notou que o sistema ocasionalmente relata que não há trens disponíveis, embora este deva ser realmente o caso. Você forneceu aos desenvolvedores um resumo do defeito e a versão do sistema testado. Eles reconhecem a urgência do defeito e agora estão esperando que você forneça mais detalhes.

- (1) Grau de impacto (gravidade) do defeito (2) Identificação do item de teste (3) Detalhes do ambiente de teste (4) Urgência/prioridade para consertar (5) Resultados reais (6) Referência à especificação do caso de teste

Qual destas informações é a mais útil para incluir no relatório de defeitos?

- a)1,2,6
- b)1,4,5,6
- c)2,3,4,5
- d)3,5,6

Questões relacionadas

3 - Você está realizando testes de sistema de reserva de trens. Com base nos casos de teste realizados, você notou que o sistema ocasionalmente relata que não há trens disponíveis, embora este deva ser realmente o caso. Você forneceu aos desenvolvedores um resumo do defeito e a versão do sistema testado. Eles reconhecem a urgência do defeito e agora estão esperando que você forneça mais detalhes.

- (1) Grau de impacto (gravidade) do defeito (2) Identificação do item de teste (3) Detalhes do ambiente de teste (4) Urgência/prioridade para consertar (5) Resultados reais (6) Referência à especificação do caso de teste

Qual destas informações é a mais útil para incluir no relatório de defeitos?

- a)1,2,6
- b)1,4,5,6
- c)2,3,4,5
- d)3,5,6**

Questões relacionadas

4 - Qual das seguintes afirmativas é a característica de uma abordagem baseada em métricas para a estimativa de teste?

- a) Orçamento que foi utilizado por um projeto de teste anterior semelhante
- b) Experiência geral coletada em entrevistas com gerentes de testes
- c) Estimativa de esforço para automação de testes acordada na equipe de teste
- d) Média dos cálculos coletados de especialistas empresariais

Questões relacionadas

4 - Qual das seguintes afirmativas é a característica de uma abordagem baseada em métricas para a estimativa de teste?

- a) Orçamento que foi utilizado por um projeto de teste anterior semelhante
- b) Experiência geral coletada em entrevistas com gerentes de testes
- c) Estimativa de esforço para automação de testes acordada na equipe de teste
- d) Média dos cálculos coletados de especialistas empresariais



Aula 3 . Etapa 6

Ferramentas de suporte ao teste

// O Caminho da Certificação CTFL

Questões relacionadas

1 -Dadas as seguintes atividades de teste e ferramentas de teste:

- (1) Medição de desempenho e análise dinâmica
 - (2) Execução de testes e registro
 - (3) Gerenciamento de testes
 - (4) Projeto do teste
-
- (a) Ferramentas de cobertura de requisitos
 - (b) Ferramentas de análise dinâmica
 - (c) Ferramentas de preparação de dados de teste
 - (d) Ferramentas de gerenciamento de defeitos
-
- a) 1B, 2C, 3D, 4A
 - b) 1B, 2A, 3C, 4D
 - c) 1B, 2A, 3D, 4C
 - d) 1A, 2B, 3D, 4C

Qual dos seguintes melhor combina atividades e ferramentas?

Questões relacionadas

1 -Dadas as seguintes atividades de teste e ferramentas de teste:

- (1) Medição de desempenho e análise dinâmica
 - (2) Execução de testes e registro
 - (3) Gerenciamento de testes
 - (4) Projeto do teste
-
- (a) Ferramentas de cobertura de requisitos
 - (b) Ferramentas de análise dinâmica
 - (c) Ferramentas de preparação de dados de teste
 - (d) Ferramentas de gerenciamento de defeitos
-
- a) 1B, 2C, 3D, 4A
 - b) 1B, 2A, 3C, 4D
 - c) **1B, 2A, 3D, 4C**
 - d) 1A, 2B, 3D, 4C

Qual dos seguintes melhor combina atividades e ferramentas?

Questões relacionadas

2 - Qual dos seguintes é o MAIS provável que seja um benefício das ferramentas de execução de testes?

- a) É fácil criar testes de regressão
- b) É fácil manter o controle de versão
- c) É fácil projetar testes para segurança
- d) É fácil executar testes de regressão

Questões relacionadas

2 - Qual dos seguintes é o MAIS provável que seja um benefício das ferramentas de execução de testes?

- a) É fácil criar testes de regressão
- b) É fácil manter o controle de versão
- c) É fácil projetar testes para segurança
- d) É fácil **executar testes de regressão**

O caminho da certificação CTFL

Carolina Santana Louzada

Analista QA - Venturus

Para saber mais

[Syllabus PT - CTFL \(bstqb.org.br\)](#)

[Certified Tester Foundation Level \(istqb.org\)](#)

[Início | BSTQB](#)

[ISTQB Glossary](#)

[Um pouco sobre cobertura de código e cobertura de testes | by Alex Candido | Liferay Engineering Brazil | Medium](#)

[Revisão Técnica Formal \(FTR\) em Engenharia de Software – Acervo Lima](#)

[Ferramentas - Aprendendo a Testar - Um guia para você aprender sobre testes de Software< \(aprendendotestar.com.br\)](#)

Percurso

Aula 1

Conhecendo a ISTQB e BSTQB

Aula 2

Estrutura e roadmap para aprovação na CTFL

Aula 3

Revisando conceitos importantes para a CTFL

Dúvidas durante o curso?

- > Fórum do curso
- > Comunidade online (Discord)

