



Introdução à Qualidade de Software

Fundamentos de QA e Qualidade de Software

Aula 1

Fred Melo

Trabalho no CESAR há 3 anos como Especialista em QA (Staff QA)

- Pós graduação em Curso Sequencial em Análise de Testes - CIn / UFPE
- ISTQB - CTFL
- Cin/Motorola (14 anos)
- Cin/Samsung (4 anos)





Olá! Eu sou Clauber Ljma

15+ anos em QA e no CESAR desde 2020

- Certificações (*CTFL, ITIL, CPRE, IC Agile Professional, PS4-RPA-FL*)
- Experiência em testes ágeis, automação e qualidade ponta a ponta
- Professor e Mentor
- Motociclista e amante de carros antigos

O que veremos hoje?

- Conteúdo 01 - Introdução a Qualidade de Software
- Conteúdo 02 - O Papel do QA no Time Ágil
- Conteúdo 03 - Atividade de Fixação

Introdução a Qualidade de Software



Qualidade

O que significa?

- Em latim '*qualitas*', que significa '*características*'.
- Segundo a ISO 9000, é o "**Grau no qual um conjunto de características atende a necessidades explícitas ou implícitas.**"
- Em engenharia significa: '*O grau em que um produto atende as expectativas*'.
- **No Software é o quanto o mesmo está em conformidade com os requisitos e com a satisfação do usuário.**

A qualidade é frequentemente percebida de forma subjetiva, variando de pessoa para pessoa e dependendo do contexto.

Conceitos básicos de Qualidade em Software

Grau com que um software atende aos requisitos especificados

- Satisfação das necessidades do usuário
- Tem impacto direto na experiência do usuário e nos custos de manutenção
- Envolve aspectos técnicos (performance, segurança) e humanos (usabilidade)
- Envolve requisitos funcionais e não funcionais



Conceitos básicos de Qualidade em Software

1. Qualidade como Responsabilidade Coletiva

- Testadores não são "garantidores da qualidade", toda a equipe (incluindo PO e devs) deve prioriza-la.

2. Foco no Valor do Usuário

- Qualidade é medida pelo impacto positivo no usuário final, não apenas por métricas técnicas (ex.: bugs resolvidos).

3. Feedback Contínuo

- Testes automatizados e ciclos curtos de feedback garantem qualidade iterativa.

4. Adaptação a Contextos

- Não há "processo perfeito", a qualidade é alcançada através de práticas adaptáveis ao projeto.

Requisitos Funcionais e não Funcionais

REQUISITO
FUNCIONAL



REQUISITO
NÃO
FUNCIONAL

	Requisitos Funcionais	Requisitos Não Funcionais
Funcionalidade	Definem as funções que um sistema terá.	Definem como o sistema executará as funções.
Usuário	Participa diretamente da definição via análise de requisitos de software.	Conceitos essenciais para a equipe de desenvolvimento garantir a eficiência e usabilidade do sistema.
Exemplos	Botões de editar, excluir, salvar um documento.	Tempo de resposta para editar, excluir e salvar um documento.

Atributos de Qualidade de Software

São características essenciais que definem a qualidade do produto e devem ser considerados durante o desenvolvimento. Alguns dos principais atributos incluem:

- **Funcionalidade:** Capacidade do software de realizar suas funções corretamente e de forma eficiente.
- **Confiabilidade:** Capacidade do software de funcionar corretamente e de forma consistente, sem falhas ou erros.
- **Usabilidade:** Facilidade de uso do software pelos usuários, sem confusão ou dificuldades.



Atributos de Qualidade de Software

→ **Eficiência:** Capacidade do software de utilizar recursos de forma otimizada, como memória e tempo de processamento.

→ **Manutenibilidade:** Facilidade de manutenção e atualização do software.

→ **Portabilidade:** Capacidade do software de ser adaptado a diferentes ambientes e sistemas.

→ **Testabilidade:** Facilidade de testar o software para garantir sua qualidade.



Esses atributos são fundamentais para garantir que o software seja de alta qualidade, atenda às necessidades dos usuários e seja fácil de manter e atualizar.

Importância da Qualidade em Software



- Reduzir falhas e retrabalho
- Aumenta a satisfação do cliente
- Diminui os custos de manutenção e suporte
- Torna o produto mais competitivo
- Aumento da produtividade
- Confiabilidade e Segurança
- Aumentar taxa de retenção de clientes

Relação entre Valor, Qualidade e Produto

Em qualidade de software, a relação entre valor, qualidade e produto é *intrínseca e crucial para o sucesso de um projeto.*



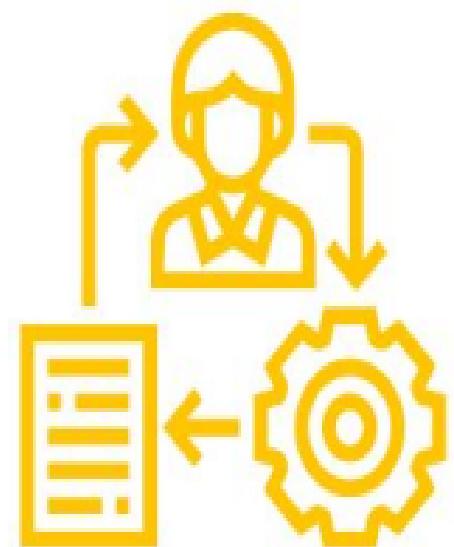
→ **Valor:** É o benefício que o usuário obtém ao usar o produto, *medido pela satisfação, redução de custos, eficiência ou melhoria da experiência.*

→ **Qualidade:** É um conjunto de características essenciais que um produto deve ter para atender às necessidades dos usuários, abrangendo funcionalidade, confiabilidade, usabilidade, eficiência e mais, resultando em um produto mais valioso.

→ **Produto:** É o software entregue ao usuário, e sua **qualidade** resulta diretamente do processo de desenvolvimento, que deve priorizar as necessidades do cliente e buscar a **excelência em todas as etapas**.

Qualidade no Desenvolvimento do Software

- No desenvolvimento de software **Qualidade** refere-se ao conjunto de práticas e processos utilizados para garantir que um produto de software atenda aos padrões de qualidade estabelecidos, desde o planejamento até a entrega e manutenção.
- A **Qualidade** pode (deve) ser incorporada em todas as fases do ciclo de desenvolvimento.
- **Objetivos:**
 - ◆ Identificar e corrigir erros, defeitos ou falhas **antes** que o software chegue aos usuários finais, e
 - ◆ Assegurar que o produto final seja confiável, eficaz e atenda às necessidades e expectativas do cliente.



Qualidade no Desenvolvimento do Software

- **Definição de requisitos:** Garantir que os requisitos sejam claros, precisos e atendam às necessidades do cliente.
- **Desenvolvimento:** Implementar práticas de desenvolvimento que garantam a qualidade do código, como testes unitários e revisão de código.
- **Testes:** Realizar testes rigorosos para identificar e corrigir defeitos antes da entrega do produto.
- **Revisão e validação:** Realizar revisões e validações para garantir que o produto atenda aos requisitos e padrões estabelecidos.
- **Manutenção:** Garantir que a qualidade seja mantida ao longo do ciclo de vida do produto, com atualizações e melhorias contínuas.



Confiável: Funcione corretamente e atenda às expectativas do cliente.

Estável: Seja estável e não apresenta problemas graves.

Como saber se meu produto é de qualidade?

Eficiente: Atenda às necessidades do cliente de forma eficiente e eficaz.

Seguro: Não apresente riscos para os usuários ou para a organização.



Valor x Qualidade x Produto

Grupo A → “Qualidade é mais importante que prazo.”

Grupo B → “Prazo é mais importante que qualidade.”



INTERVALO!



15 min.

O Papel do QA no Time Ágil

O que é um Time Ágil?

Um time ágil é um grupo de profissionais que trabalham juntos de forma colaborativa e flexível para desenvolver produtos ou serviços, utilizando metodologias ágeis.

- **Colaboração:** Trabalho em equipe, comunicação aberta e compartilhamento de responsabilidades.
- **Flexibilidade:** Adaptação às mudanças e priorização de tarefas com base nas necessidades do cliente.
- **Auto-organização:** Times ágeis são auto-organizados, tomando decisões e gerenciando seu próprio trabalho.
- **Foco no cliente:** O objetivo é entregar valor ao cliente de forma rápida e eficiente.
- **Melhoria contínua:** Times ágeis buscam constantemente melhorar seus processos e práticas.



Os times ágeis são compostos por profissionais com diversas habilidades e especializações, trabalhando juntos para alcançar objetivos comuns e entregar produtos ou serviços de alta qualidade.

O QA não é só um testador



O que é Teste de Software, afinal?

- Processo sistemático e multidisciplinar de avaliação de um produto de software
- Identificar discrepâncias entre seu comportamento atual e os requisitos esperados
- Engloba atividades técnicas e analíticas para detectar defeitos, validar funcionalidades e assegurar que o sistema atenda às necessidades do usuário final.

Teste é um conjunto de atividades que visam avaliar a qualidade de um produto, reduzindo riscos e fornecendo informações para a tomada de decisão. (*ISTQB (2023)*)

O QA não é só um testador

- O QA (Quality Assurance) não se limita apenas a testar o produto.
- Embora o teste seja uma parte importante do trabalho do QA, suas responsabilidades são mais amplas.
- O QA desempenha um papel fundamental em garantir a qualidade do produto e a satisfação do cliente, indo além da simples execução de testes.

"Enquanto o tester é o detetive que procura o erro, o QA moderno é o arquiteto que ajuda a construir um produto sem falhos."

QUALITY ASSURANCE



O QA não é só um testador!

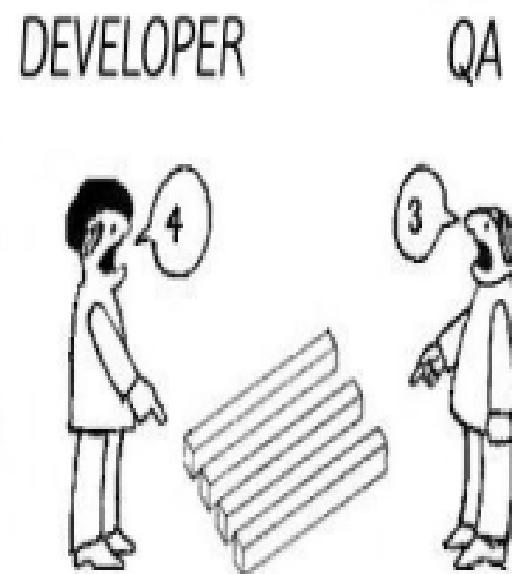
Fonte: CEBRA 2019. Todos os direitos reservados.

Aspecto	Tester Tradicional	QA Moderno
Momento de atuação	Final do ciclo	Inicio ao fim (continuo)
Tipo de teste	Manual	Automatizado + Manual
Ferramentas	Básicas ou nenhuma	Ferramentas modernas e integradas
Foco	Encontrar bugs	Prevenir bugs, melhorar qualidade
Papel na equipe	Isolado	Colaborativo
Método de trabalho	Cascata	Ágil / DevOps
Engajamento com time	Baixa	Alto

Responsabilidades do QA em Times Ágeis

Em times ágeis, o QA (Quality Assurance) desempenha um papel crucial em garantir a qualidade do produto. Algumas das principais responsabilidades incluem:

- **Testes:** Planejar, executar e automatizar testes para garantir a qualidade do produto.
- **Colaboração:** Trabalhar em estreita colaboração com a equipe de desenvolvimento para identificar e resolver problemas.
- **Garantia de qualidade:** Garantir que os padrões de qualidade sejam atendidos em todas as fases do desenvolvimento.
- **Feedback:** Fornecer feedback contínuo sobre a qualidade do produto e sugerir melhorias.
- **Automatização de testes:** Automatizar testes para garantir a eficiência e a eficácia dos processos de teste.



Responsabilidades do QA em Times Ágeis

- **Integração contínua:** Trabalhar com a equipe para implementar práticas de integração contínua e entrega contínua.
- **Análise de riscos:** Identificar e analisar riscos relacionados à qualidade do produto e sugerir mitigação.



Habilidades Essenciais do QA Ágil

- **Conhecimento de metodologias ágeis:** Entender os princípios e práticas ágeis.
- **Testes automatizados:** Saber criar e manter testes automatizados.
- **Ferramentas de teste:** Conhecer ferramentas de teste e automação.
- **Análise de dados:** Analisar dados de teste para identificar tendências e problemas.
- **Comunicação eficaz:** Comunicar problemas e resultados de forma clara e eficaz.
- **Colaboração:** Trabalhar em estreita colaboração com a equipe de desenvolvimento.
- **Flexibilidade:** Adaptar-se às mudanças e priorizar tarefas com base nas necessidades do projeto.



Habilidades Essenciais do QA Ágil

- **Conhecimento técnico:** Ter conhecimento técnico em áreas como programação e bancos de dados.
- **Pensamento crítico:** Identificar problemas e sugerir soluções inovadoras.
- **Melhoria contínua:** Buscar constantemente melhorar processos e práticas de teste.

Essas habilidades permitem que o QA ágil contribua para a entrega de produtos de alta qualidade e satisfaça as necessidades do cliente.

Colaboração QA + Devs

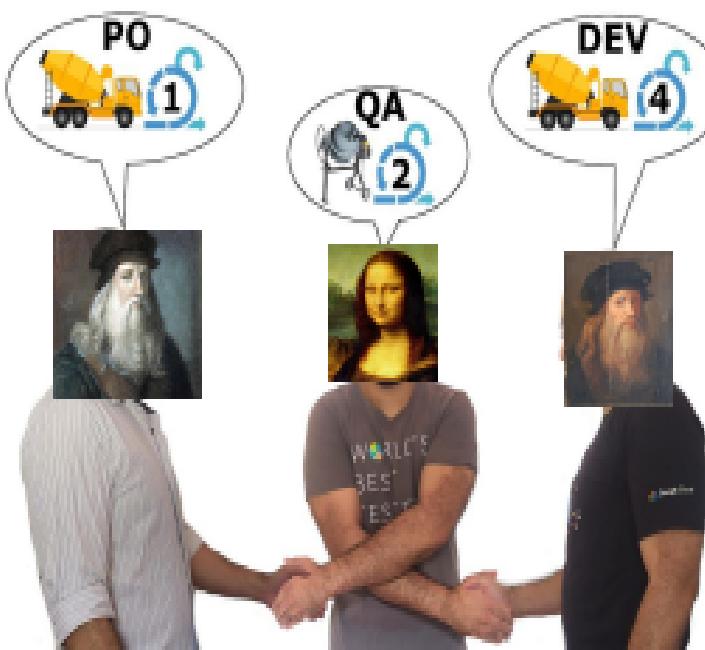
- **Planejamento em conjunto:** QA e Devs participam das plannings e refinamentos juntos.
- **Definição de critérios de aceitação:** QA ajuda a garantir que as user stories sejam claras, testáveis e alinhadas com os objetivos do negócio.
- **Code reviews focados na testabilidade:** QA sugere melhorias no código pensando na cobertura de testes.
- **Automação de testes integrada:** Trabalham juntos para criar pipelines de CI/CD com testes automatizados (unitários, integração, UI).
- **Testes de APIs e componentes:** QA pode escrever ou validar testes de serviços junto aos Devs.



Colaboração QA + Devs

Benefícios

- Prevenção de bugs (em vez de só detecção).
- Redução do retrabalho.
- Aumento da cobertura de testes automatizados.
- Produtos entregues com mais confiança e rapidez.



Colaboração QA + POs

Atividades

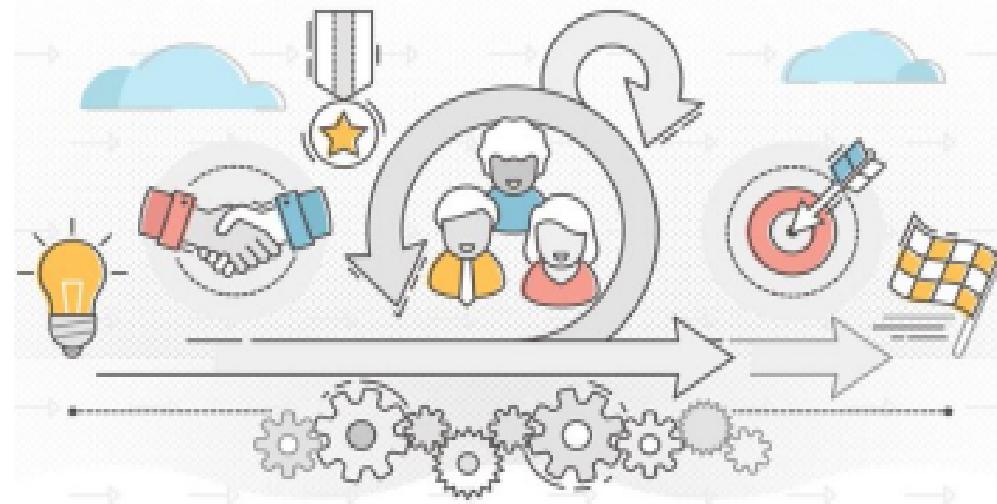
- ➔ **Refinamento de requisitos:** QA atua com o PO para transformar requisitos vagos em critérios de aceitação objetivos.
- ➔ **Priorização de testes:** Juntos, decidem o que realmente precisa ser validado de forma mais intensa (testes de risco).
- ➔ **Validação de hipóteses do produto:** QA ajuda a monitorar o comportamento do usuário e métricas pós-entrega.
- ➔ **Feedback contínuo:** QA traz dados reais (ex: bugs recorrentes, falhas críticas, testes não cobertos) para orientar melhorias no backlog.

Benefícios

- ➔ Produto mais aderente ao que o cliente realmente quer.
- ➔ Melhora na comunicação entre negócio e tecnologia.
- ➔ Entregas com maior valor agregado e menor risco.

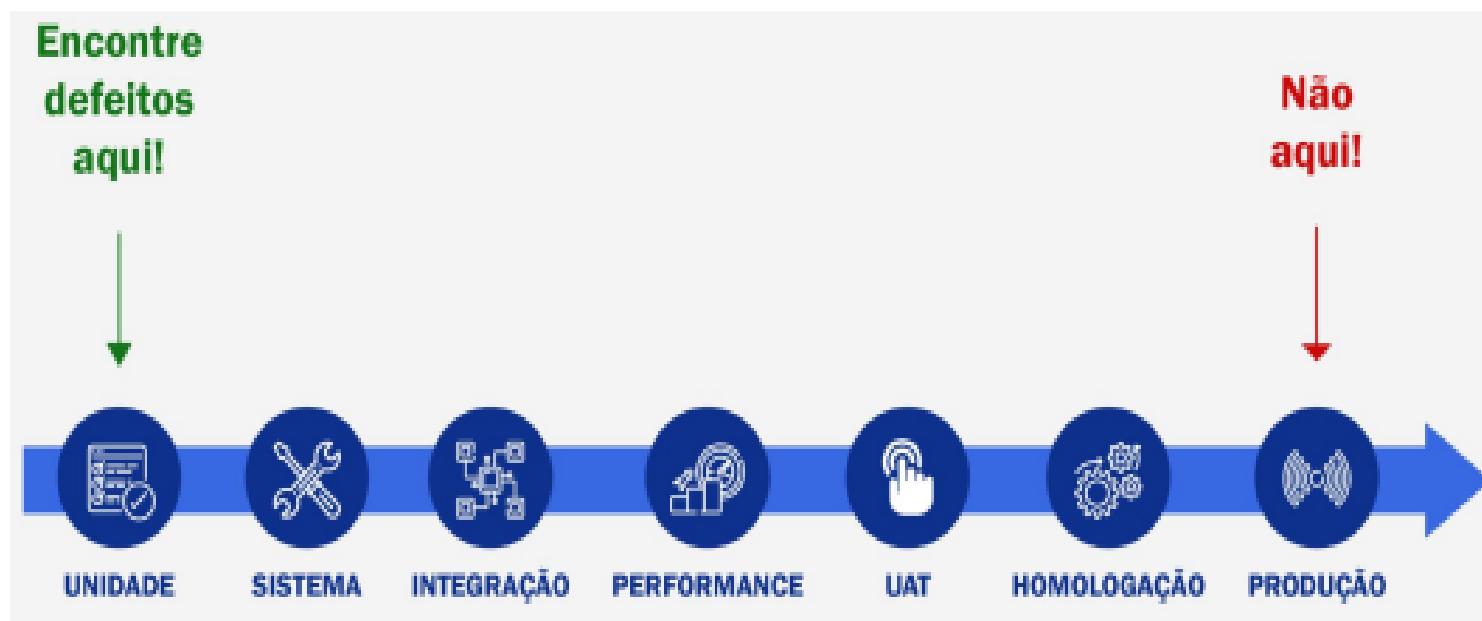
Colaboração no Fluxo de Trabalho Ágil

- **Participação Precoce:** Envolvimento no planejamento do sprint, fornecendo insights sobre a testabilidade e identificando potenciais riscos à qualidade.
- **Feedback Contínuo:** Fornecer feedback constante sobre o desenvolvimento, contribuindo para a melhoria contínua do processo.
- **Colaboração com a Equipe:** Trabalhar em conjunto com a equipe de desenvolvimento, promovendo um ambiente de trabalho colaborativo e uma cultura de responsabilidade compartilhada pela qualidade.
- **Comunicação Aberta:** Promover a comunicação aberta entre os membros da equipe, garantindo que todos estejam alinhados e informados sobre o progresso do projeto.



Colaboração no Fluxo de Trabalho Ágil

- **Testes Automatizados:** A utilização de testes automatizados contribui para a eficiência do QA, permitindo a validação rápida e repetitiva das funcionalidades do software. Além de liberar tempo para execução de testes exploratórios.
- **Foco na Qualidade:** Não se limitar apenas à identificação de defeitos, mas também garantir que o software atenda às necessidades e expectativas do usuário, entregando valor ao cliente.



Ferramentas e Práticas Comuns

Os QAs utilizam várias ferramentas e práticas para garantir a qualidade do produto.
Algumas das mais comuns incluem:

- **Ferramentas de teste automatizado:** Selenium, Appium, Cypress.



- **Ferramentas de gerenciamento de testes:** TestRail, TestLink, JIRA.



- **Ferramentas de rastreamento de defeitos:** JIRA, Bugzilla, Trello.



Ferramentas e Práticas Comuns

- Ferramentas de automação de testes de API: Postman, RestAssured.



- Práticas de teste: Teste unitário, teste de integração, teste de sistema.
- Metodologias ágeis: Scrum, Kanban.
- Ferramentas de colaboração: Slack, Microsoft Teams.



Desafios comuns

1. Ambiguidades ou ausência de requisitos claros

→ Desafio:

- Falta de documentação ou histórias de usuários mal definidas prejudica a criação de testes eficazes.

→ Como enfrentar:

- Participar ativamente das **cerimônias ágeis** (refinamento, planning).
- Fazer **perguntas ao PO e ao time de desenvolvimento**.
- Criar critérios de aceitação claros juntos aos stakeholders.



Desafios comuns

2. Comunicação com Devs e POs

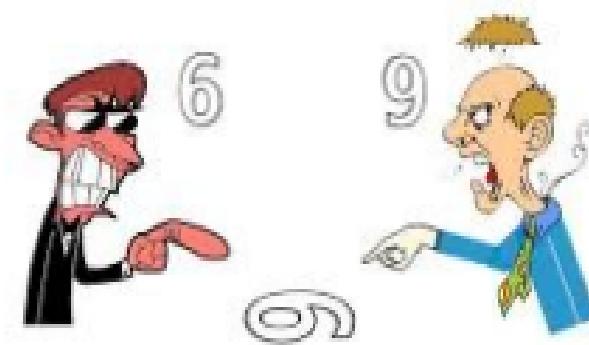
→ Desafio:

- Barreiras na comunicação entre QA, Devs e POs podem causar retrabalho.

→ Como enfrentar:

- Incentivar a cultura de colaboração e feedback constante.
- Utilizar ferramentas visuais como quadros Kanban ou fluxos de testes.
- Praticar pair testing com desenvolvedores.

Developer vs Tester



Desafios comuns

3. Ambiente de testes instável

→ Desafio:

- Ambientes de staging ou QA frequentemente estão indisponíveis ou inconsistentes.

→ Como enfrentar:

- Automatizar a **configuração de ambientes** com containers (ex: Docker).
- Trabalhar com **mockagens e dados fakes** quando possível.
- Criar um **processo claro de deploy** para ambientes de teste.



Desafios comuns

4. Falta de tempo para testar

→ Desafio:

- Prazos curtos levam a testes superficiais ou não realizados.

→ Como enfrentar:

- Priorizar testes usando **análise de risco**.
- Automatizar **testes de regressão**.
- Negociar **tempo de QA no planejamento da sprint**.



Desafios comuns

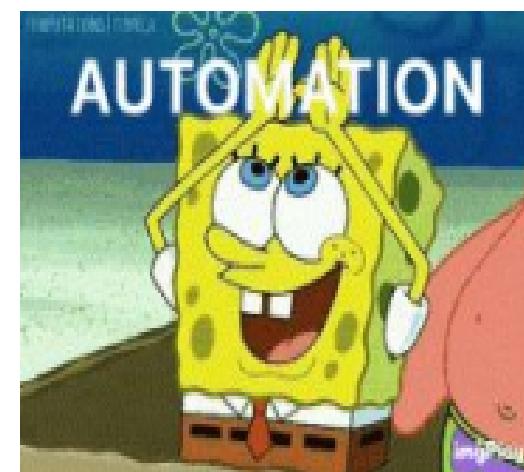
5. Automação mal planejada

→ Desafio:

- Testes automatizados quebrando com frequência ou não agregando valor.

→ Como enfrentar:

- Usar padrões de automação (como Page Object).
- Manter os testes curtos, independentes e confiáveis.
- Revisar periodicamente os testes automatizados para remover os obsoletos.



Desafios comuns

6. Interação valor ao time

→ Desafio:

- QA ser visto apenas como "testador" e não como parte estratégica do time.

→ Como enfrentar:

- Mostrar o valor através de **métricas de qualidade** (bugs evitados, falhas em produção reduzidas).
- Contribuir para a **melhoria contínua do processo de desenvolvimento**.
- Propor iniciativas como **testes exploratórios, melhoria de cobertura, workshops internos**.



Desafios comuns

7. Lidar com mudanças constantes

→ Desafio:

- Alterações frequentes nos requisitos afetam planejamento de testes.

→ Como enfrentar:

- Investir em testes modulares e manutáveis.
- Adotar BDD (Behavior Driven Development) com Gherkin para alinhar expectativas.
- Ter processo leve de versionamento e replanejamento de testes.



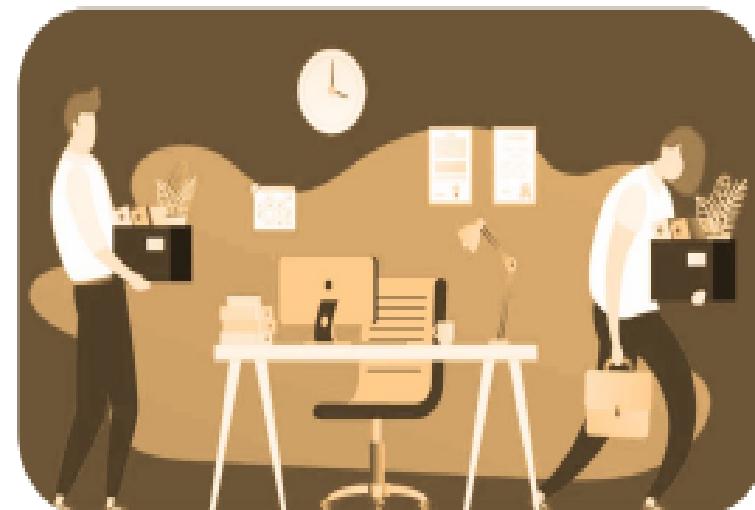
Desafios comuns

No fim, queremos evitar...

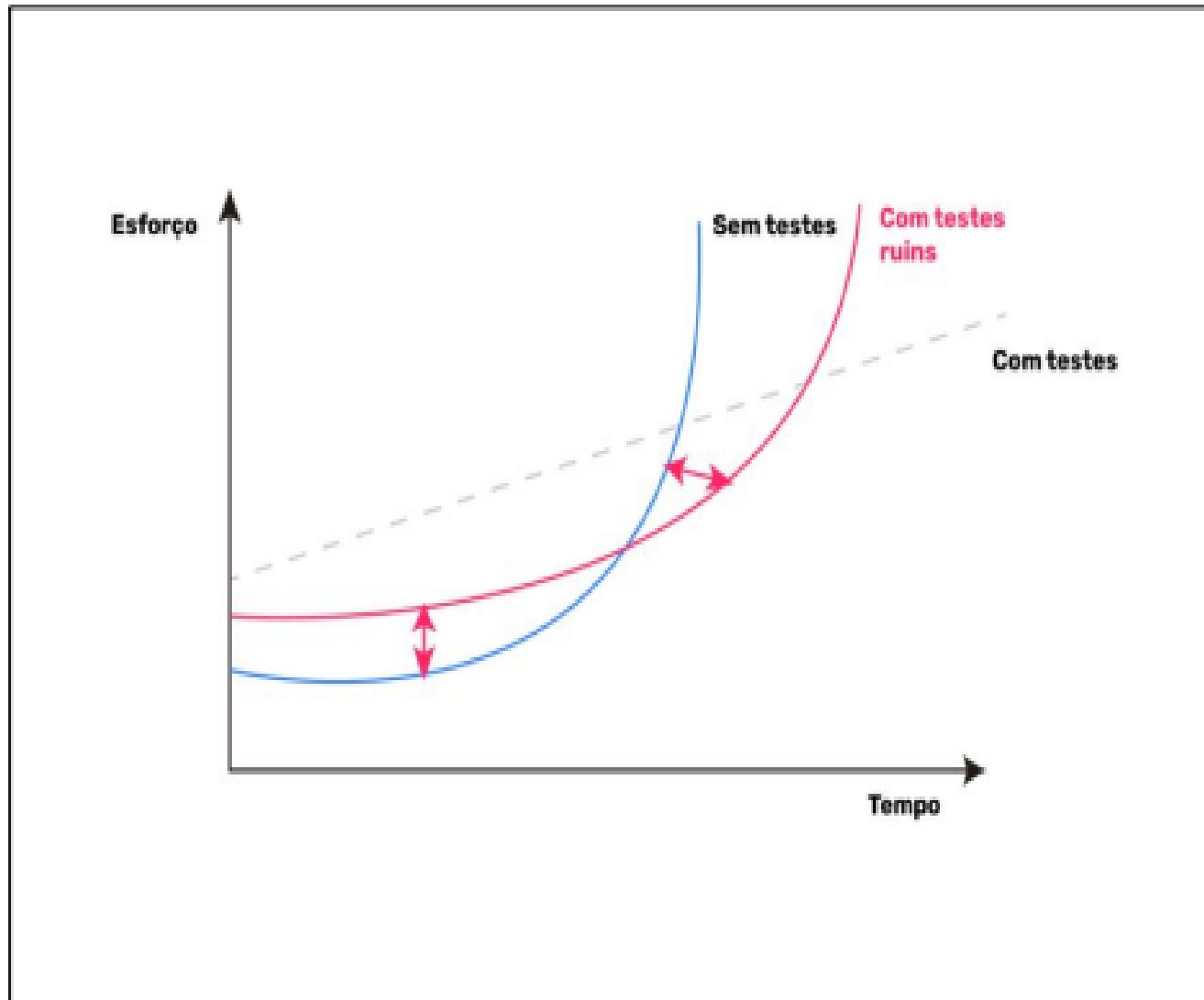


Impactos de NÃO investir em Qualidade

- Aumento de custos (*overhead*)
- Alta rotatividade nos *times* (*Turnover*)
- Reputação prejudicada
- Insatisfação do cliente
- Falta de confiança



Custo x Esforço



Relação de ESFORÇO x CUSTOS x TEMPO

Vamos praticar!

📌 Atividade de Fixação – Aula 01

📝 Agora é hora de testar seus conhecimentos!

👉 Acesse o link abaixo e responda ao formulário com perguntas sobre os fundamentos de QA e Qualidade de Software.

🔗 [LINK FORMS](#)

⌚ Tempo: 10 minutos

💡 Responda com base nos tópicos vistos na aula.

Boa sorte!