Qualidade de software

Leonardo de Souza

O que é qualidade?

“Qualidade é um termo subjetivo, para o qual cada pessoa, ou setor, tem a sua própria definição.”

No desenvolvimento de software, a qualidade de um projeto engloba o grau de atendimento às funções e características especificadas no modelo de requisitos.

A qualidade de conformidade focaliza o grau em que a implementação segue o projeto e que o sistema resultante atende às suas necessidades e às metas de desempenho.

satisfação do usuário

=

produto compatível +

boa qualidade +

entrega dentro do

orçamento +

prazo previsto

Mas como definir a qualidade de software?

No sentido mais geral a qualidade de software pode ser definida como:

“Garantir que um produto seja bem feito de maneira a ser útil tanto para quem o faz quanto para quem o utiliza, usando métodos eficientes de controle de qualidade.”

Adequação ao Uso:
A qualidade de algo é medida pela sua capacidade de atender às necessidades e expectativas de quem o utiliza. Em outras palavras, se um produto atende bem ao propósito para o qual foi feito, é considerado de boa qualidade.

Aplicação Generalizada:
A qualidade não se limita a contextos específicos. Pode ser aplicada tanto a coisas do dia a dia, como carros, bolos, canetas e celulares, quanto a situações mais especializadas, como processos de fabricação ou desenvolvimento de software.

Um carro de qualidade deve ser seguro e eficiente; um bolo de qualidade deve ser saboroso e bem assado; uma caneta de qualidade deve escrever suavemente; um celular de qualidade deve ter um desempenho confiável; e um software de qualidade deve ser eficaz, atender aos requisitos do projeto e ser livre de erros.

Contudo, considerando-se, o Sistema de Gestão da Qualidade, amplamente aplicável, sob os requisitos da ISO 9000 (é uma série de normas internacionais que define os requisitos para sistemas de gestão da qualidade), a definição normativa é:
“Qualidade é o grau no qual um conjunto de características inerentes, satisfaz requisitos.”

Grau de Conformidade:

"Grau" refere-se ao nível ou extensão em que algo atende a determinados padrões ou requisitos. Quanto mais alto o grau, melhor atende aos requisitos.

Características Inerentes:

São as propriedades intrínsecas ou naturais de um produto ou serviço. Em um contexto de software, por exemplo, poderiam ser a eficiência, a confiabilidade, a usabilidade, entre outros.

Requisitos:

Refere-se às especificações ou condições que devem ser cumpridas. Estes podem ser requisitos do cliente, regulamentares, internos da organização, etc.

Portanto, a definição normativa indica que a qualidade de um produto ou serviço é medida pelo quão bem suas características inerentes atendem aos requisitos estabelecidos.

Uma gestão de qualidade eficaz cria as bases necessárias para garantir que qualquer esforço para desenvolver um software de alta qualidade seja bem-sucedido. Isso significa que, ao estabelecer boas práticas e estruturas organizacionais, a equipe responsável pela criação do software terá as condições adequadas para produzir um produto final de alta qualidade. A gestão de qualidade aqui atua como um suporte essencial, fornecendo a estrutura necessária para alcançar os objetivos de desenvolvimento de software com excelência.

Um produto útil é aquele que oferece exatamente o que o usuário. Além disso, é crucial que o produto seja confiável, ou seja, funcione corretamente sem problemas, e que seja livre de erros, proporcionando uma experiência sem falhas para o usuário. Em resumo, um produto útil não apenas atende às necessidades do usuário, mas também o faz de forma confiável e sem erros.

Quando um software é de alta qualidade, isso traz benefícios tanto para a empresa que o desenvolve quanto para os usuários que o utilizam. Para a empresa, isso significa melhor reputação, satisfação do cliente e possíveis oportunidades de negócios. Já para os usuários, um software de qualidade oferece uma experiência mais satisfatória, eficiente e confiável. Em resumo, a qualidade do software cria um cenário em que todos saem ganhando: a empresa, ao obter sucesso e reconhecimento, e os usuários, ao desfrutar de um produto confiável e útil.

Para avaliar a qualidade de um software, precisamos considerar diferentes fatores, os quais podem ser divididos em duas grandes categorias.:

Fatores que podem ser medidos diretamente (por exemplo, defeitos revelados durante testes)

Fatores que podem ser medidos apenas indiretamente (como a facilidade de uso ou a manutenção do software) porque dependem da experiência e percepção individual do usuário.

Isso destaca a importância de abordar diversos aspectos ao avaliar a qualidade de um produto de software, tanto aqueles que são facilmente mensuráveis quanto aqueles que requerem uma avaliação mais subjetiva.

Fatores de qualidade de software de McCall

Os fatores de qualidade de software propostos por McCall, desenvolvidos por John McCall em 1977, oferecem uma estrutura para avaliar e medir a qualidade de um sistema de software. Esses fatores são agrupados em três categorias principais: operação, revisão e transição.

1. Fatores de Operação:

a. **Correção (ou exatidão):** - Avalia a capacidade do software de fornecer resultados corretos.

b. **Confiabilidade:** - Mede a capacidade do software de realizar suas funções conforme esperado, mantendo um desempenho consistente ao longo do tempo.

c. **Eficiência:** - Avalia o desempenho do software em relação aos recursos computacionais que utiliza, como tempo de resposta e uso de memória.

d. **Integridade:** - Refere-se à capacidade de proteger contra acesso não autorizado e garantir a consistência dos dados.

e. **Usabilidade:** - Mede a facilidade com que os usuários podem aprender e operar o sistema.

2. Fatores de Revisão:

a. **Manutenibilidade:** - Avalia a facilidade com que o software pode ser modificado ou aprimorado.

b. **Flexibilidade:** - Mede a facilidade com que o software pode ser adaptado para diferentes ambientes ou requisitos.

c. **Testabilidade:** - Refere-se à facilidade de testar o software para garantir que ele funcione conforme esperado.

d. **Facilidade de Entendimento:** - Avalia a clareza e compreensibilidade do código-fonte e da documentação.

3. Fatores de Transição:

a. **Portabilidade:** - Mede a facilidade com que o software pode ser transferido para diferentes ambientes ou plataformas.

b. **Reusabilidade:** - Avalia a capacidade do software de ser reutilizado em diferentes contextos ou em partes diferentes do mesmo sistema.

c. **Interoperabilidade:** - Refere-se à capacidade de interagir efetivamente com outros sistemas.

Avaliar a qualidade de uma aplicação usando esses fatores possibilitará uma sólida indicação da qualidade de um software.

Custo da qualidade

Para analisar os impactos das falhas encontradas no software é necessário avaliar o impacto de quando os erros foram encontrados.

Portanto, o impacto de se encontrar e consertar os defeitos aumenta consideravelmente ao longo do tempo.

Lembrem-se: quanto mais cedo encontramos o problema, menos caro é resolvê-lo!

O custo da qualidade pode ser dividido em custos associados à prevenção, avaliação e falhas.

Os custos de prevenção incluem:

(1) o custo de atividades de gerenciamento necessárias para planejar e coordenar todas as atividades de controle e garantia da qualidade. **Exemplo:** Salários de gerentes de projeto, custos associados a ferramentas de gerenciamento de projeto, recursos para reuniões e comunicação.

(2) o custo de atividades técnicas adicionais para desenvolver modelos completos de requisitos e de projeto. **Exemplo:** esforço adicional de desenvolvedores.

(3) os custos de planejamento de testes. **Exemplo:** Salários de testadores, recursos para ambientes de teste, custos de tempo dedicado à elaboração de documentação detalhada.

(4) o custo de todo o treinamento associado a essas atividades. **Exemplo:** Custos de treinamento formal, tempo dedicado à aprendizagem de novas habilidades, materiais de treinamento.

Os custos de avaliação incluem:

(1) o custo da realização de revisões técnicas. **Exemplo:** Tempo dos revisores, custos associados a ferramentas de revisão de código, esforço para corrigir problemas identificados.

(2) o custo dos artefatos de engenharia de software. **Exemplo:** Tempo de desenvolvedores, custos de ferramentas de modelagem e documentação, recursos para manter e atualizar artefatos.

(3) o custo da coleta de dados e avaliação de métricas. **Exemplo:** Tempo dedicado à coleta e análise de métricas, custos de ferramentas de medição, esforço para interpretar e aplicar insights derivados das métricas.

(4) o custo dos testes e depuração. **Exemplo:** Salários de testadores, custos de ferramentas de automação de teste, esforço de desenvolvedores para corrigir bugs.

Os custos de falhas incluem:

Os custos de falhas podem ser subdivididos em custos de falhas internas e custos de falhas externas. Os custos de falhas internas ocorrem quando se detecta um erro em um produto antes de ele ser entregue. As falhas externas são reclamações, devoluções ou substituição depois de entregue o produto. Eles abrangem:

(1) o custo necessário para realizar reformulações (reparos) para corrigir um erro. **Exemplo:** Tempo e recursos dedicados à correção do problema, custos adicionais de materiais ou mão de obra.

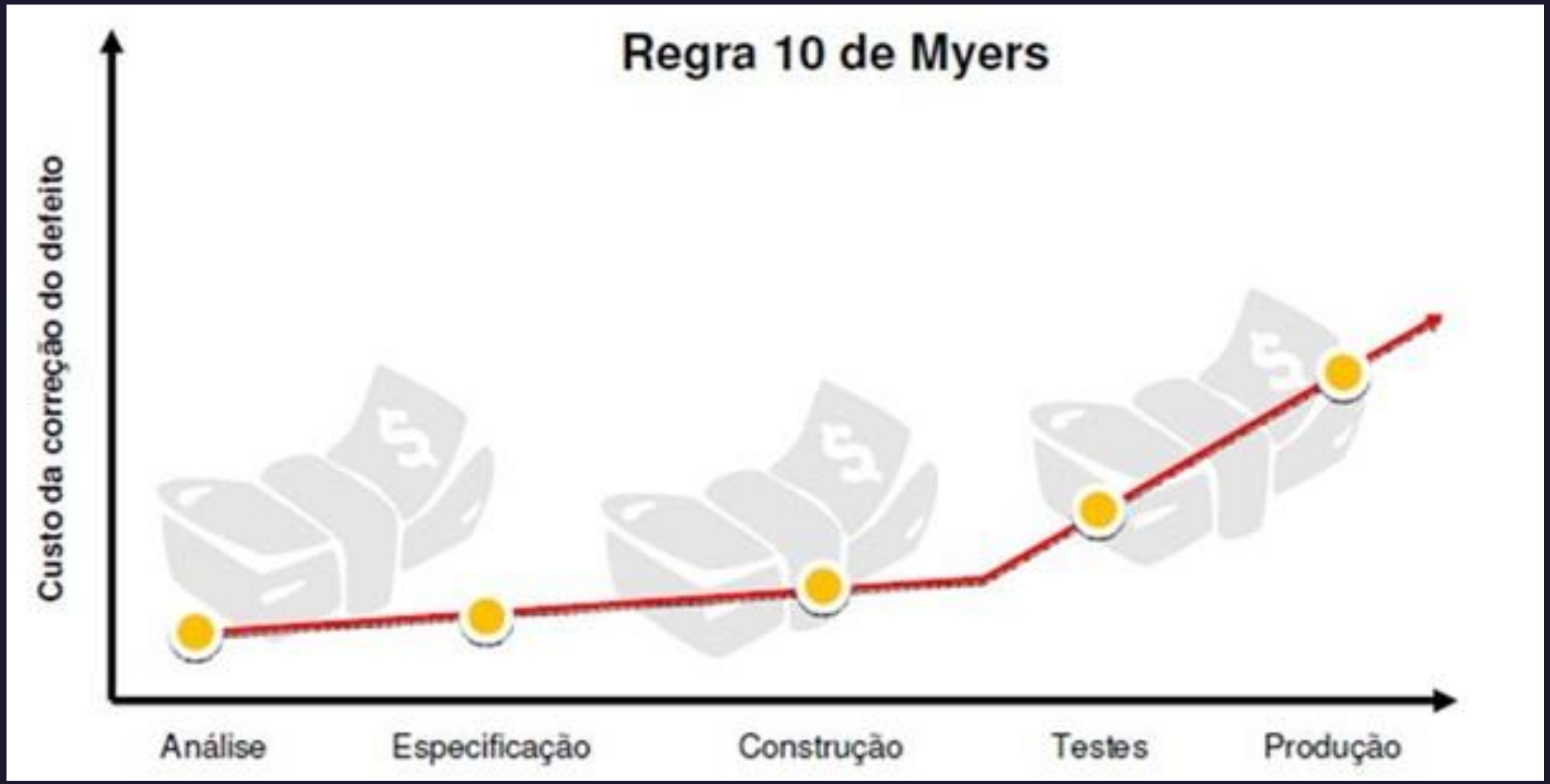
(2) o custo que ocorre quando reformulações geram, inadvertidamente, efeitos colaterais que devem ser reduzidos. **Exemplo:** Esforço adicional de desenvolvedores para resolver problemas causados por modificações anteriores, custos de retrabalho.

(3) os custos associados à reunião de métricas de qualidade que permitam a uma organização avaliar os modos de falha. **Exemplo:** Tempo dedicado à coleta e análise de métricas, custos de ferramentas de medição, recursos para interpretar e aplicar insights derivados das métricas.

Quanto mais cedo descobrimos e corrigimos um defeito, menor é o seu custo para o projeto.

Defeitos encontrados nas fases iniciais da etapa de desenvolvimento do software são mais baratos de serem corrigidos do que aqueles encontrados na etapa de produção.

Em 1979, Glenford Myers em seu livro ‘The Art of Software Testing’ (A arte de teste de software), já apresentava o conceito no qual quanto mais cedo descobrimos e corrigimos o erro, menor é o seu custo para o projeto. Esse custo em correção de erros cresce 10 vezes para cada estágio em que o projeto do software avança.


Regra 10 de Myers
Custo da correção do defeito
Análise
Especificação
Construção
Testes
Produção

Verificação e Validação

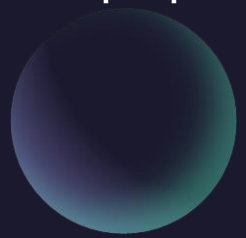
O teste de software é um elemento de um tema mais amplo, muitas vezes conhecido como verificação e validação (V&V).

Verificação: "Estamos criando o produto corretamente?"

A verificação refere-se ao processo de avaliar se o produto está sendo construído de acordo com as especificações e requisitos. Em outras palavras, verifica se o software está sendo desenvolvido corretamente, seguindo os padrões, normas e requisitos técnicos estabelecidos. Isso inclui atividades como revisões de código, inspeções e testes para garantir que o software atenda às suas especificações.

Validação: "Estamos criando o produto certo?"

A validação, por outro lado, diz respeito à confirmação de que o produto final atende às necessidades e expectativas do cliente. Está relacionada à pergunta fundamental de se o software está atendendo aos objetivos de negócio e às necessidades reais dos usuários. A validação envolve a avaliação do software em termos de usabilidade, funcionalidade e adequação ao propósito.



O objetivo da verificação é analisar se o software atende aos requisitos funcionais (funcionalidades específicas que o software deve oferecer) e não funcionais (desempenho, segurança, usabilidade, entre outros).

O objetivo da validação é garantir que o software atenda às expectativas do cliente.

Atividade V&V

- Pesquise de que forma podemos aplicar os principais conceitos relacionados à atividade de VV – Validação, Verificação (dê exemplos)

- Cite e descreve 3 técnicas que podem ser utilizadas para Validação ou Verificação de software.

Os testes não podem demonstrar se o software é livre de defeitos ou se ele se comportará conforme especificado em qualquer situação.

É sempre possível que um teste que você tenha esquecido seja aquele que poderia descobrir mais problemas no sistema.

Os testes têm a capacidade de revelar a presença de erros, mas não podem garantir que o software está completamente livre de defeitos.