QUALIDADE DE SOFTWARE - (ISO/IEC 25010:2011)

[tem slides e estão bem completos]

Qualidade - ISO9000:2005 - conjunto de características inerentes que satisfazem requisitos
capacidade do produto de software de satisfazer
necessidades declaradas e implícitas sob condições especificadas
Foco no que o cliente quer

Aspectos que devemos considerar:
Requisitos de software - base para medir qualidade
O cliente tem uma ideia do software mas ele não sabe tudo o que precisa, existem requisitos que são implícitos

Percepções de qualidade - cada parte que fala de qualidade terá um visão

Visão transcendental - olha para o produto e sabe que ele é bom
Visão do usuário - personalizar de acordo com a necessidade do usuário
Visão de manufatura - conformidade aos requerimentos
Visão de produto - visão da galera e negócios, pensar no produto de dentro para fora
Visão em valor - custo deve estar balanceado ao que o cliente deseja

Normas e padrões de qualidade
A padronização é importante para que todos se entendam, ela tenta minimizar ao máximo os erros
Instituições importantes: IEEE, ISO, IEC - eles podem entrar em consenso para definir as normas

Família ISO 9000 - específica para qualidade de software
ISO 9126 é super importante

IEEE 730:2014 - importante

Família SQuaRE: ISO/IEC 25000-25099 - substitui ISO/IEC 9126

Normas, padrões não são obrigatórios. A qualidade do software começa desde a conversa com o cliente.

Medição da qualidade
Qualidade está em tudo, desde a criação dos requisitos.

Métricas internas influenciam nas métricas externas.
Qualidade externa influencia diretamente no produto final do software
Métricas de qualidade de uso: quanto o produto atende às necessidades do usuário (não são métricas técnicas)

ISO/IEC 25010 - 8 características e várias subcaracterísticas

Qualidade do produto de software - são muito internas, o usuário não faz ideia delas
Qualidade do produto em uso - são questões da visão do usuário. Exemplo: Satisfação do usuário, Segurança - o usuário não quer saber como foi implantado, mas utilizar o software com segurança.

Processo de gerenciamento de qualidade de software
Buscar a satisfação do usuário.
PLANEJAMENTO - determinar padrões, utilizar as normas, ter olhar crítico para definir como as coisas serão. Organiza as atividades de qualidade, independente de ser metodologia ágil ou não.

GARANTIA DE QUALIDADE - garantindo que o que está sendo produzido é realmente o que foi pedido pelo cliente.

CONTROLE DE QUALIDADE -

MELHORIA DOS PROCESSOS - melhoria do processo do ciclo de vida, sugerir mudanças para o time. Ter um olhar crítico, ver se os requisitos estão incompletos.

GERENCIAMENTO DE DEFEITOS

Controle de qualidade
Análise estática: documentação e códigos estarem em conformidade, se preocupa com o código-fonte em si
Análise dinâmica: técnicas com o código em execução, testes automatizados ou manuais
Hoje em dia essas atividades não necessariamente são separadas.

Ato de testar é uma verificação da qualidade do produto.

Verificação: garantir que o produto está sendo construído corretamente. Isso não significa que está de acordo com o que o cliente quer
Validação: conformidade com o que o usuário pediu e precisa

Encontrei erros e agora? **Caracterizando defeitos/erros**

- Analisa o erro e entende o produto, suas necessidades
- Facilita a correção

Rastreamento de defeitos é importante para área de controle, dar um feedback aos desenvolvedores.

Importante padronizar os termos utilizados pela equipe!

Existem vários motivos para erros: falta de tempo, pressão, complexidade, inexperiência…

Ciclo de vida do bug
1- Bug é reportado e cadastrado (mas nem tem certeza se é um bug)
2- Desenvolvedor pega o defeito para corrigir
3- Ainda não começa a correção
4- Corrige e coloca como FIXED - do ponto de vista do desenvolvidor
5- Tempo de espera para a equipe de teste testar
6- A equipe de testa envia para o re-teste
7- Verified: defeito corrigido *ou* Reopen: não corrigido
8- Closed: corrigido + testado + aprovado

Essas acima não são regras, os processos se adequam ao time e ao que o produto precisa.

Reports: são importantes para ter métricas e estatísticas do processo.

Ferramentas de suporte
Nem todas são específicas para rastreamento de defeitos.
Bugzilla (específica para defeitos) / Jira (gerenciamento completo) / Azure Devops / Asana

INTRODUÇÃO AOS TESTES DE SOFTWARE

Teste: serve para avaliar e reduzir o risco de falhas
O processo de teste faz parte do controle de qualidade

Objetivos de se fazer teste como um controle:
- Evitar defeitos
- Está cumprindo requisitos?
- Está como o cliente espera?

TESTE x DEPURAÇÃO
Teste: vai mostrar falhas causada por defeitos de software
Depuração: processo de investigação e correção do erro e não é somente função do desenvolvedor

Existem 7 princípios básicos de teste
- Ele prova que existe defeito e não que não existe
- Impossível testar tudo, todas as situações
- Teste inicial, desde quando o código é desenvolvido economiza tempo e dinheiro

- Onde normalmente há um erro, podem acontecer outros.
- Para verificar novos erros, é preciso novas entradas e condições
- Para testar depende do cliente, do time, do contexte
- Se você não quer que seu software tenha bugs, não construa um

Processo de teste

Planejar - qual o propósito do teste e qual abordagem
Monitorar - sabe que o teste foi feito e gerou resultado esperado
Analisar - base de teste para saber o que testar. Com base no requisito do cliente, conseguimos organizar os testes
Modelar o teste - como testar?
Implementar - pensar não só nos cenários, mas na sequências dos cenários
Executar os testes -
Conclusão do teste

Níveis de teste
4 níveis de teste

Sistema de baixo nível = teste de unidade
Produto design de alto nível = teste de integração
Especificação Software = verificação do sistema, teste de sistema
Requerimentos usuário = teste de aceite

Teste de unidade: testa pequenas partes do código
Teste de integração: por exemplo, sistema externo que conversa com o sistema que está sendo desenvolvido. Comunicação de back-end e front-end.
Teste de sistema: verificar todo o percurso de usabilidade do sistema
Teste de aceite: validação, ponto de vista do usuário

Tipos de teste
Teste funcional: exemplo - teste de interface
Teste não funcional:
Teste caixa-branca:
Testes de mudança: Confirma que o erro encontrado foi corrigido

Técnicas de teste
Caixa-preta: baseadas no comportamento do sistema. Não é visível a estrutura interna do projeto
Caixa-branca: estrutura interna do projeto é visível