Fala dev:

Definição de teste de software:

É a exploração do sistema de forma que ele atenda os requisitos implicitamente e explicitamente.

Falando sobre a garantia da qualidade do software.

Não tem como construir um software sem bugs, ou seja, não tem como garantir qualidade essa não é uma palavra muito adequada. Funcional.

Diferença da literatura para a prática, que na prática não é utilizado nem 5% do que se é visto na literatura.

Empresas usarem o que elas acreditam ser testes de software sem realmente usar teste de software como a gente estudou.

Certificação de testes?

Falando sobre a divisão que desenvolvedor só desenvolve e o teste só testa

Não existe bala de prata que se aplique a tudo o que você irá trabalhar no mercado.

Testes para se iniciar em uma empresa, deve se começar pelos testes de unidade, depois os módulos e por último o sistema, sempre focando mais na unidade. Meio impossível aplicar testes integrados para projetos grandes com micro serviços.

Converter teste de unidade em teste integrado é complicado.

Programar pensando nos testes.

Falando sobre a dificuldade de isolar tudo nos testes de unidade com moks.

Pensar em técnicas de teste para um certo módulo, ou seja, modelar testes.

Falando que no mercado acabam fugindo do teste pelo tempo de execução dos testes.

O Brasil não cria seus desenvolvedores para testar códigos, só usa quem já tem muita experiência.

Testar pelo contrário de perder tempo você ganha pelo fato de não perder muito tempo com bug.

Automatizar os testes para realizar testes de forma rápida, ou seja, teste unitário não deveria existir mais.

Muitas vagas hoje em dia pedem testes automatizados.

Testabilidade é uma característica de qualidade de software.

Compreensão de código:

Leitura de código é um negócio que depende de vários fatores.

Quem observar o código deve ter uma habilidade para entender.

Manutenabilidade por conta da leitura do código, ou seja, padrões no código ajudam a legibilidade diminuindo a introdução de novos bugs.

Legibilidade nem sempre é boa para a evolução do código.

Não se deve repetir trechos de código ou comentários.

Nem sempre código curto é fácil de entender.

Depende muito de várias variáveis para ocorrer um entendimento de software, é complexo saber tudo o que está sendo feito.

Muito achismo na área acadêmica sobre códigos.

Não se deve fazer código o mais genérico do que precisa, porém refatorar código no mercado nem sempre é bem visto.

A linguagem implica muito no entendimento de código, mas nem sempre a linguagem com comandos menores é mais legível.

Dificuldade de leitura de ponteiros.

Exceções é outra coisa difícil para leitura.

Código com indentação ajuda o compreendimento, porém muito espaço também é difícil.

Medir a facilidade de alterações, porém existem problemas em métricas pois são meio inviáveis.

Não tem como saber como é a melhor forma de dizer como o código é a melhor forma possível de se ler o código, ou seja, não existe bala de prata.

Bot de melhoria automática em legibilidade de código.