Quando queremos trabalhar em uma área precisamos saber que área é essa.
Quem nasceu primeiro o software ou o defeito ?
O primeiro software já nasceu com defeito, o defeito é uma manifestação de uma natureza imperfeita.
Quem produz o software é um ser humano e todos nós somos imperfeitos, e para isso que o teste é o melhor caminho para criarmos mais qualidade nos nossos produtos.

Século XIX(1840-1850) **Charles Babbage** criou a primeira máquina de calcular, veio para substituir os Ábacos, esta máquina de babbage era possível ser programada, criou um sistema com tiras de furos para criar dados.

Ada Lovelace ajudou Charles a fazer correção na máquina de calcular, Ada é considerada a primeira programadora,a origem da palavra defeito veio da Ada em uma carta para o Charles.

No final do final do século XIX **Herman Holerite** transformou ideia da máquina do Charles em uma máquina de senso demócrito (dados estatísticos, geográficos), Herman criou a máquina de ponto, uma das novas invenções que Herman realizou foi o cartão perfurado para representar dados das pesquisas, o censo demográfico (tipo nosso IBGE hoje), o cartão perfurado era guardado em sequência dos furos , era guardado em uma caixa branca , por isso surgiu o nome do teste caixa branca,(teste das lógicas, da estrutura do computador).

Alan Turing foi quem ganhou a segunda guerra mundial, ele criou um computador eletromecânico que desvendava a máquina enigma que os amigos usavam durante as batalhas na guerra para comunicar com o governo do seu país, porém essa máquina mudava o enigma a cada 24 horas, então Alan criou uma máquina onde desvendava este enigma em menos de 24 horas. (Jogo da Imitação filme sobre o Alan Turing) .

Grace Hopper, tenente da marinha mecânica (09/09/1947) realizou o registro do primeiro defeito, conforme seus registro de serviço, ela encontrou um inseto dentro da máquina, onde surgiu o nome bug, Grace encontrou o bug no programa mark2, o qual começou a ser utilizado depois da segunda guerra mundial.

Glenn Ford Marx escreveu o livro a Arte de Testar Software, por volta de 1979 trabalhou na empresa IBM, criou a regra 10 de Myers, o aumento do custo vai aumentando quanto mais tarde você descobre um defeito, quanto mais cedo descobrir um defeito mais barato fica o software.

Barry Boehm e Peter Coff realizaram outros estudos, o custo de encontrar um defeito amanhã mês vem ano que vem, não vai ser o custo de encontrar agora, *os defeitos cobrem juros sob juros (rsrsrs).*

No início dos anos de 2000 surgiram vários profissionais no Brasil na área de testes, onde surgiram vários livros, como a certificação CTBS.

- ➔ **Emerson Rios presidente da ALATS,**
- ➔ **Trayahu Moreira Ricardo Cristália,**
- ➔ **Anderson Bastos,**
- ➔ **Leonardo Molinari,**
- ➔ **Alexandre Bartie,**
- ➔ **Marcio Delamaro.**
- ➔ **Osmar Higashi**

Voltado para certificação: **(CTFL ,ISTQB)**

- ➔ **Rex Black,**
- ➔ **Dorothy Graham,**
- ➔ **Martin Paul,**

São autoras que falam sobre a especialização da área de teste**,Lisa Crispin ,Janet Gregory.**

Danos dos Bugs:

- ➔ **Prejuízos Financeiros e de Imagem**

Quando entregamos um produto sem testar causa um dano enorme para clientes, usuários, organizações empresa.
Quando temos um software que tem muitos bugs atrasa o projeto todo, causando uma insatisfação ao cliente, o cliente começa a pensar se a empresa realmente vai entregar algo com qualidade no prazo que foi informado, se vai ter mais bugs, isso reflete em uma perda de contratos, perda de novas vendas, perda de indicações de novos clientes, causando uma perda de imagem muito forte, prejuízo de dinheiro é ruim perco agora, mas prejuízo de imagem é péssimo, pois a minha imagem a maneira como os clientes me enxergam se torna desgastada, diminui minha “popularidade” entre os clientes.

Sistemas funcionando errado nos leva a tomar decisões erradas.

Os 7 fundamentos de Teste (ISTQB)

- ➔ Teste demonstra a presença de defeitos mas nunca a sua ausência;

Os testes reduzem a probabilidade que erros permaneçam no software.
Não existe software perfeito.

- ➔ Teste exaustivo é impossível

Priorizar o que é mais importante

➔ Teste Antecipado

Quanto antes testar melhor é

Se não encontrar o defeito ele vai se espalhando e aumenta a dificuldade em encontrar o defeito.

"Quanto mais cedo encontrar o defeito mais barato será sua correção".

➔ Agrupamento de defeitos

locais onde existe grande risco

Onde o cliente mais reclama

➔ Paradoxo Pesticida

Paradoxo quando penso que estou fazendo bem mas na verdade estou fazendo mal.

Um conjunto de testes repetitivos pode não encontrar novos defeitos, mas um conjunto de testes novos e diferentes pode aumentar a possibilidade de encontrar mais erros.

➔ Teste depende do contexto

Testes são realizados de forma diferente conforme o contexto.

É mais importante achar um bug que pode falir a empresa do que um bug sem impacto.

➔ A ilusão da ausência de erros

Nada adianta fazer algo perfeito se não atender a necessidade do cliente.

Um QA precisa sempre garantir que um projeto atenda as necessidades dos clientes.

Não é questão de ser perfeito ou não, mas sim entender se esse software atende as necessidades do cliente naquele momento.

Teste e Qualidade

Diferença entre tester e QA.

Teste trabalha no produto.

O teste é voltado para avaliar o produto.

Testador realiza testes manuais e automatizados.

QA é para melhorar o processo para fazer o produto.

Projeto anteriores devem prover lições aprendidas:

Entendimento da causa raiz dos defeitos encontrados;

Aprimorar os processos;

Melhoria contínua, busca melhorar a qualidade dos sistemas futuros;

Análise de defeitos, boas práticas de desenvolvimento, treinamentos.

Agrupamento de Erros

Erro> Defeito> Falha trata-se do momento em que o projeto está.

Erro ➡️Cometido por pessoas/usuários
Defeito ➡️ Defeito é algo que outra pessoa fez de errado.
Algo que não está funcionando no código ou na documentação.
Falha ➡️ Consequência do defeito executado, quando sistema está sendo executado.
Aquilo que foi pedido para desenvolver mas nunca ninguém usou pode estar com diversos bugs que quando executados se transformam em falhas.

Eu jamais vou encontrar o erro de outra pessoa, pois somente a pessoa que montou sabe onde está o erro.

As falhas geram insatisfação com a qualidade.

Base IEC/ISO 25010
Padronizar é para facilitar a vida das pessoas
SQuaRE = Requerimento e Avaliação da Qualidade de Sistemas e Software

1. **AF Adequação Funcional**
   Completude Funcional - Se tudo que foi solicitado para o software fazer, se ele faz, faz parte do que foi solicitado.
   Correção Funcional - É dar o resultado está certo.
   Apropriado a Funcionalidade - Faz o resultado de maneira apropriada, são coisas que estão corretas mas a forma como mostra não está apropriado.

2. **U Usabilidade**
   Reconhecibilidade- Facilidade que o usuário reconheça os elementos e comportamentos da tela.
   Aprendizibilidade- Facilidade de aprendizado do usuário- auto explicativo
   Operabilidade- Facilidade na operação e navegação, quando é mais prático o uso.
   Proteção Contra Erro do Usuário- Quando limitados o usuário a digitar uma informação que pode dar erro em novos processos.
   Estética (da Interface do Usuário)- Uma API não tem estética, mas uma tela tem, precisa ser ergonômica, a estética sempre nos ganha.
   Acessibilidade- Facilitar acesso a todas as pessoas com deficiência ou sem.

3. **C Compatibilidade**
   Coexistência- Facilidade de Coexistir, funciona sem “atrapalhar outros apps” (um app não pode dar conflito com outro app).
   Interoperabilidade- Facilidade de se comunicar com outros software, consegue enviar e receber dados, consegue se autenticar. (ex, uber, iFood,99, banco)

4. **C Confiança/Confiabilidade**
   Ter a certeza que o software está disponível para o uso.

   Maturidade- Perceber Prevenir a falha antes que aconteça.
   Disponibilidade- O software está disponível para o uso
   Tolerância a falhas- Perceber e compensar as falhas em tempo real.
   Recuperabilidade- Recuperar-se de falhas e travamentos.

5. **E Eficiência de Desempenho**

   Comportamento em Relação ao Tempo - Performance desempenho
   Utilização de Recursos - Como utiliza a memória RAM e os recursos que tem.
   Capacidade - Atender transações e usuários (teste de volume/ Estresse).

6. **M Manutenibilidade**

   Modularidade- Organizado em módulos.
   Reusabilidade- Facilidade de reutilizar, criar de uma maneira fácil de entender.
   Analisabilidade- Facilidade de entender e analisar o código.
   Modificabilidade- Facilidade de modificar, entender a estrutura para trocar módulos
   Testabilidade- Facilidade de Testar.

7. **P Portabilidade**

   Adaptabilidade- Facilidade de funcionar em um novo ambiente (responsividade)
   Instabilidade- Facilidade de instalar e desinstalar.
   Substituibilidade- Facilidade em substituir.

8. **S Segurança**

   Confidencialidade- Só você ou alguém acima de você pode ter acesso aos dados.
   Integridade- Somente pessoas autorizadas podem modificar, mas precisa ficar registrado quem fez e quando fez.
   Não repúdio- Garantir que quem está fazendo o acesso é a pessoa real.
   Responsabilidade- Garantir que a pessoa realmente fez ser auditável.
   Autenticidade- Garantir que a transação foi feita, garantir que a pessoa realmente fez a transição.

Testes manuais X automatizados
Quanto mais versátil formos melhor é.

Teste manual para navegar pelo programa manualmente, é um teste mais lento. Só testa é o que é novo, não realiza teste de regressão.

O teste automatizado ocorre em uma base diária e detecta erros de integração de modo antecipado e rápido.

Uma automação que não é executada diariamente vai parar de funcionar, pois precisa de um planejamento e uma rotina para automatizar todos os dias, fazer um pouco todos os dias e rodar tudo vai nos ajudar a acompanhar o projeto, é muito utilizado junto com entrega contínua.

Testes Ágeis hoje o tester é um membro ativo do projeto, melhorando a qualidade da entrega do projeto, seja comunicativo, conselheiro, espalhe agilidade.

As pessoas não conhecem a qualidade.

Teste Tradicionais era muito utilizado nos anos 90/2000, mas tudo mudou, a entrega contínua virou algo normal,

Pressão Organizacional

O tester recebe muita pressão, pressão q vem de todos os lados,
Stakeholders:
Desenvolvedor > Sempre estarão ansiosos com os projetos, precisamos saber lidar com eles;
PO > sempre vem com as dúvidas dos clientes, sempre ser sincero com eles;
Scrum Master > Estão sempre preocupados em resolver impedimentos para que seja uma equipe mais produtiva;
Gestores > responsável por diversas áreas da empresa, temos que entender as mudanças do gestor, nosso produto não é único;
Clientes>
Usuários>

- ➔ Aconselhar, ajudar as pessoas a entenderem o que é qualidade, ajudar a minha equipe a entender o que é qualidade,ajudar o time a melhorar.

- ➔ As empresas hoje querem profissionais comprometidos.

- ➔ As pessoas **envolvidas**, querem dinheiro mas não querem trabalhar, elas querem um benefício mas não querem pagar o preço para chegarem a este benefício.
- ➔ As pessoas **comprometidas**,

Organização é fundamental

Gerenciamento de tempo

- ➔ Saber lidar com a escassez do tempo;
- ➔ Ser organizado;
- ➔ Separe o tempo para cada coisa, tente sempre buscar o equilíbrio.
- ➔ Busque sempre renovar suas energias, descubra qual período você consegue concentrar melhor em determinada atividade, e dedique-se durante esse período;
- ➔ Precisamos encontrar o balanço da nossa energia, qual período você tem maior concentração?
- ➔ Existem tarefas que somente nós podemos fazer;
- ➔ Existem pessoas que só querem ajudar mas não querem ajudar, cuidado com pessoas que querem ficar na sua aba, sempre ajuda quem quer ser ajudado e quem quer ajudar também.

“Quem quer fazer faz, quem não quer fazer reclama”

Comunicação verbal e não verbal

A leitura ajuda e muito nas interpretações do que estamos falando ou escutando.
Falar é diferente do que escrever.
Utilize mapa mental, anotações, planeje o que você vai falar, e perguntar.

Verbal > Escrita > Livro, e-mail, Mensagem, esta mensagem precisa ser exata para que não haja confusão no que a pessoa está lendo.

Oral> Rádio, Televisão, Vídeos, Conversas, Áudios, pode ser feita de forma mais fragmentada, pode haver perguntas durante a fala, sendo assim se tornando uma conversa.

Formal x Informal > Devemos saber com quem falamos, como falamos e onde falamos.
Por ex: em uma festa com amigos não devemos ser formal, assim como em uma reunião de diretoria não devemos falar de forma informal (usando gírias).

Temos que observar com quem conversamos para que as pessoas possam entender o que estamos falando e entender o que estamos querendo passar.

Influência da mensagem > As palavras representam apenas 10 % da mensagem do texto, a fala tem 35 % do impacto da mensagem. A entoação como realizamos nas nossas falas seja ela para uma reunião ou apresentação terá uma consequência para as pessoas na passagem da sua mensagem. O tom de voz ajuda a criar uma ligação com o que você está falando.

Não verbal > Cuidado com as cores que você vai utilizar nas mensagens, não escreva usando a *caps lock* ativada pois isso vai dar a impressão que você está gritando, a comunicação verbal temos que prestar bastante atenção com o que vamos escrever como escrever, cuidado com símbolos, emoticons e figurinhas.

Proxêmica > É a forma como analisamos a pessoa, usa-se a estética do ambiente para ajudar a passar uma determinada mensagem, o ambiente completa a nossa comunicação.

Negociação

Ganha-Ganha
Os negociadores encontram uma solução para ambos saírem satisfeitos.