Curso de Teste de Software

https://www.udemy.com/share/101t0KBEAdc11WQ34=/

O que é:

Verificação feita sobre um sistema ou parte dele para garantir que uma determinada entrada produza sempre uma saída esperada.

Verdades sobre o Teste:

1. Testes **não** verificam completamente as saídas de um sistema, pois as entradas são infinitas.
2. Teste não garante Qualidade, Qualidade vai além dos testes. Qualidade é relativa.
3. Testes custam em média 20 a 30% da elaboração total de um software.

Características de um Testador

- Visão Crítica
- Visão Voltada para a Qualidade
- Detalhista
- Criterioso
- Paciente
- Perspicaz
- Focado
- Motivado
- Ser referência de critério e qualidade.

Um Testador deve ser capaz de:

- Elaborar testes relevantes
- Executar testes no tempo planejado

Ser o Sherlock Holmes da TI:

Investigativo, o testador deve ser um profissional curioso, aquele que procura nas combinações mais absurdas erros que podem comprometer o sistema. Aquele que investiga os comportamentos do software além do que a documentação descreve, tentando compreender o erro a fim de criar novos casos de testes para descobrir novos defeitos.

Usar de sensatez:

Fazer uso do bom senso é uma das principais características de um profissional da qualidade, afinal é este profissional que mensurará o nível de qualidade do sistema. Saber diferenciar a severidade de um erro básico na interface da severidade de um problema na integridade dos dados de um determinado software é um ponto positivo e essencial para o testador.

Ser o negociador:

Assim como os vendedores do Grand Bazaar em Istambul, que tem a capacidade de convencer um cliente a comprar um produto que nunca será usado, o testador deve usar a sua capacidade de negociação e persuasão a favor da qualidade, sugerindo melhorias e possíveis soluções. Convencer o desenvolvedor de que a lógica empregada está errada nem sempre é uma tarefa fácil, principalmente em projetos que não tenham documentações técnicas e funcionais bem detalhadas.

Fundamentos do Teste de Software

Quando testar?

Níveis de Teste

1. **Teste de Unidade**

   Valida a menor “parte” testável dentro de um sistema. Este tipo de teste garante o funcionamento de uma pequena parte de um sistema que independe do todo.

   Exemplos: Classe, método(função).

   Geralmente este tipo de teste é feito pelo programador.

2. **Teste de Integração**

   Valida a comunicação entre os componentes de um sistema.

   Exemplos: Funcionalidades que envolvem integração entre os sistemas. Componente A invoca um método de um componente B que espera o valor X mas recebe Y.

   Geralmente este tipo de teste é feito pelo programador.

3. **Teste de Sistema**

   Tem o objetivo de encontrar falhas no sistema verificando funcionalidades existentes do ponto de vista do usuário final.
   São realizados após a codificação do sistema estar concluída.

   Exemplos: Verificar se o campo de email aceita um e-mail inválido.

   Desenvolvido e executado pela equipe de Teste, os casos de testes devem ser coerentes com os requisitos especificados para o sistema.

4. **Teste de Aceitação**

   Tem o objetivo de permitir ao usuário final decidir se aquele sistema vai atender as necessidades dele ou não.
   Similar ao teste de sistema, é realizado após o sistema ser desenvolvido, tem o objetivo de encontrar falhas verificando as funcionalidades existentes no sistema ele também é realizado quando a codificação do sistema for concluída. Porém o teste de aceitação é feito realmente pelo usuário do sistema.

   Executado pelos usuários finais do sistema que simulam operações de rotina e vão verificar se o resultado está de acordo com o esperado.

5. **Teste Alfa**

   Similar ao teste de Sistema, porém realizado por um número maior de usuários que vão simplesmente utilizar o sistema de forma não planejada, ou seja, sem casos de testes ou um fluxo que essas pessoas deveriam seguir.
   Objetiva a identificação de possíveis erros.

   O time de desenvolvimento acompanha o processo de perto para coletar falhas a serem corrigidas ou melhorias a serem implementadas.

6. **Teste Beta**

   Similar ao teste Alfa porém é realizado por um grande número de usuários que normalmente são pessoas desconhecidas (sem contato com a equipe de desenvolvimento ou de testes. Esses usuários geralmente obedecem a certos critérios estabelecidos pelo fornecedor do sistema. (Dentro do público alvo)
   Exemplo: Usuários do Sudeste, faixa etária de idade, idioma.

   Não é acompanhado pela equipe de desenvolvimento, semelhante a um pré-lançamento.

7. **Teste de Regressão**

   Consiste em reexecutar testes após serem realizadas alterações no sistema para verificar se tudo ainda continua funcionando. Tem o objetivo de detectar efeitos colaterais.
   A cada nova versão, ou novo ciclo reexecutar os testes que foram aplicados nas versões passadas (Testes de Release).
   Ideal para serem utilizados em sistemas com teste automatizado.

   Pode ser executado tanto pela equipe de teste quanto pelos desenvolvedores.

Como testar?

Técnicas de Teste

1. **Caixa Branca (Teste de caixa aberta)**

   No momento do teste o código do sistema é avaliado.
   - Testes de métodos e classes
   - Testes de comandos de repetição
   - Testes de condições

   Níveis de teste:
   - Teste de Unidade
   - Testes Estáticos - Verificar se o código está documentado

   ### 1.1. Teste Estático

   Tem o objetivo de analisar o código **sem** executá-lo(review) e observar se as boas práticas adotadas na programação estão sendo respeitadas.
   - Documentação de código
   - Variáveis e constantes possuem boa nomenclatura.
   - Código está organizado e com boa legibilidade.
   - Código obedecendo a arquitetura do sistema.
   - Fechamento de conexão com o banco.
   - Tratamento de exceção.

   Exemplo ferramenta: FindBugs

2. **Caixa Preta**

   O código não interessa neste tipo de teste, e sim o funcionamento do sistema através da sua interface.
   - Testes baseados em entradas e saídas de cenários Macro

Níveis de teste:

- Teste de Integração
- Teste de Sistema
- Teste de Aceitação
- Teste Alfa
- Teste Beta

2.1. Teste Dinâmico

Validar o sistema através da sua execução.
Método tradicional inserir entradas, avaliar as saídas.

3. Testes manuais

Testes executados manualmente, usuário utilizando o sistema fazendo casos de testes.

Desvantagens:

- Possui baixa velocidade de execução.
- Repetitivo e cansativo.
- Possui limitações quando o teste envolve grande paralelismo.

Vantagens:

- Executa testes fora do escopo dos testes (além do cenário).
- Não exige conhecimento de tecnologia, porém exige percepção, concentração e observação por parte do testador.
- Verificar questões relacionadas a Qualidade, Usabilidade, Elementos Visuais da Tela.

4. Testes automatizados

Testes executados por um sistema automatizado.

Desvantagens:

- Demora no desenvolvimento.
- Só executa testes dentro do escopo (Se algum caso de teste for esquecido por falha de quem desenvolveu os testes aquilo simplesmente nunca vai ser verificado.)

Vantagens:

- Velocidade de execução.
- Uma vez construídos podem ser executados a qualquer momento.
- Não está limitado ao paralelismo.

O que testar?

Tipos de Teste

1. **Teste de Funcionalidade**

   Validar se as funcionalidades do sistema estão funcionando corretamente. (Regra de negócio).

2. **Teste de Desempenho(Carga)**

   Validar o desempenho do sistema de acordo com o tempo de resposta para determinadas operações (requisições).

3. **Teste de Usabilidade**

   Validam a experiência do usuário para utilizar o sistema.

4. **Teste de Segurança**

   Validam a proteção do sistema contra invasões ou acesso não autorizado.

5. **Teste de Portabilidade**

   Validam o funcionamento do sistema em diferentes plataformas e dispositivos, mas qual o sistema está proposto a funcionar.

6. **Teste de Stress**
   Validam o comportamento do sistema em condições extremas.

Como os testes entram no Desenvolvimento de Software

Imagem Ciclos de Testes Desenvolvimento de Software

Boas Práticas

Não Técnicas:

1. A pessoa que testará o sistema não deve ser a mesma pessoa que programa.
    a. O sistema deve ser bem especificado.
2. Faça uma boa seleção de cenários.
3. Seja pessimista! Teste o sistema acreditando que ele vai falhar.
4. Se coloque no lugar do usuário.(leigo ou não). Pense como ele pensaria.

Técnicas:

1. Não execute cenários roboticamente, explore quando necessário.
2. Se achar uma falha, detalhe ao máximo.
3. A falha foi corrigida. Reteste!
4. Busque padrões ao reportar uma falha.
    a. Verifique se esse erro apresenta um padrão, por exemplo: é um problema daquela tela ou de várias telas.
5. Ferramentas Web, gerenciamento de falhas.
6. Teste sempre valores/intervalos limites.
    a. Formulários:
        i. Campo em branco.
        ii. Campos com valor inválido.
        iii. Quantidade de caracteres.
        iv. E-mail inválido.
        v. Mensagens de erros coerentes.
    b. Cálculos:
        i. Valores positivos.
        ii. Valores negativos.
        iii. ZERO.
        iv. Intervalos fechados e abertos.(Acima do limite máximo e abaixo do limite mínimo.)