Disciplina: **Teste**

Antonio Carlos Ferreira de Almeida - RA:2438194

Exercício 1

Condições de conclusão
A fazer: Fazer um envio
Aberto: segunda, 9 jan 2023, 08:00
Vencimento: domingo, 22 jan 2023, 23:59
Introdução ao teste de software:

1. Explique as fases do teste (unidade, integração e sistema).
2. Explique as técnicas de teste (funcional e estrutural).
3. Explique o critério de teste funcional particionamento em classes de equivalência?

Entregar em .pdf.

Respostas:

1.Explique as fases do teste (unidade, integração e sistema).
R: O ***teste de unidade*** concentra esforços na menor unidade do projeto de software, ou seja, procura identificar defeitos de lógica e de implementação em cada módulo do software.
Esse tipo de teste geralmente é realizado isoladamente do resto do sistema e pode utilizar objetos simulados, virtualização de serviços, simuladores e controladores. O teste unitário pode cobrir funcionalidades (por exemplo: correção de cálculos), características não funcionais e propriedades estruturais (por exemplo: teste de decisão). Para realizar o teste de unidade deve-se obrigatoriamente ter acesso ao código que está sendo testado. Esse tipo de teste geralmente é elaborado pelo desenvolvedor que escreveu o código. São simples e rápidos, mas não são suficientes. É importante que eles sejam complementados com outras fases de teste.

O ***teste de integração*** é uma atividade sistemática aplicada durante a integração da estrutura do programa visando a descobrir erros associados às interfaces entre os módulos. O objetivo é, a partir dos módulos testados no nível de unidade, construir a estrutura de programa que foi determinada pelo projeto.
Os testes de integração são os mais comuns de serem esquecidos na hora de escrever testes para um projeto. Mesmo testando duas unidades que interagem entre si separadamente, usando mocks, virtualização etc, e concluindo que ambas estão funcionando como esperado. Ainda assim, é possível que as duas unidades não funcionem bem em conjunto. Realizar testes de integração não é testar a lógica das unidades, mas testar como as diferentes unidades interagem entre si. Por outro lado, testes de integração são bem mais simples de desenvolver, manter e são bem mais rápidos que os testes de ponta a ponta.

Já o ***teste de sistema***, realizado após a integração do sistema, visa a identificar erros de funções e características de desempenho que não estejam de acordo com a especificação. O teste de sistema se concentra no comportamento e nas capacidades de todo um sistema ou produto.

Geralmente considerando as execuções das tarefas de ponta a ponta do sistema e os comportamentos não funcionais exibidos ao executar tais tarefas. O teste do sistema geralmente produz informações que são usadas pelos stakeholders para tomar decisões de liberação. O teste do sistema também pode satisfazer requisitos ou padrões legais e regulatórios. Esse tipo de teste é o mais comum de se implantar, pois simula a experiência do usuário. São, portanto, muito importantes, pois são os testes mais próximos do que o usuário realmente vai encontrar ao utilizar a aplicação.

Geralmente esse tipo de teste é realizado no ambiente que antecede o ambiente de produção. O teste de sistema deve focar no comportamento geral, funcional e não funcional, de ponta a ponta do sistema como um todo.
Os testes de sistema agregam muito valor a qualidade do software, mas possuem alguns problemas como, por exemplo:
são lentos de serem escritos e executados, são passíveis de “Flaky Test” (teste com falso negativo no caso de automação) e apresenta
baixo detalhe do erro (ao encontrar um erro, algumas vezes é difícil saber a origem exata do problema, já que esses testes são definidos em um nível muito alto e cada passo pode envolver muitas unidades).

2.Explique as técnicas de teste (funcional e estrutural).
R: Os ***testes funcionais*** (Black Box) de um sistema envolve testes que avaliam as funções que o sistema deve executar. Suas técnicas se escalam nos requisitos funcionais e podem ser descritos em produtos de trabalho, como especificações de requisitos, de negócios, épicos, histórias de usuários, casos de uso ou especificações funcionais, podendo ainda não estarem documentados. As funções são “o que o sistema deve fazer”.

O projeto e a execução de *testes funcionais* podem envolver habilidades ou conhecimentos especiais, como o conhecimento específico de um problema de negócios que o software resolve ou o papel específico que o software desempenha. Os tipos de testes funcionais incluem testes de unidade, teste de interface, testes de regressão, controle, entre outros.

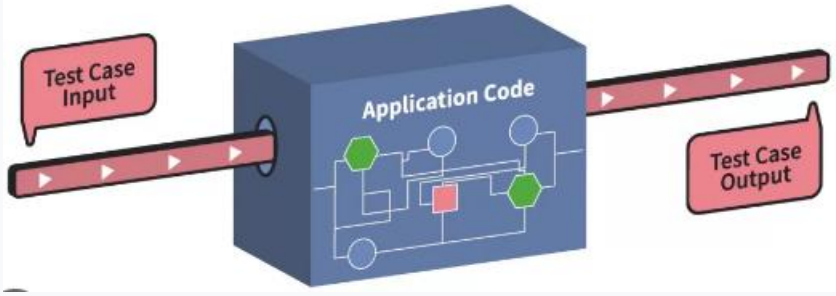
Técnicas de Teste Funcional
- Teste de interconexão;
- Testes paralelos;
- Testes de requisitos;
- Testes de regressão;
- Testes de suporte manual;

- Testes de tratamento de erros.

Os ***testes estruturais*** (White Box) garantem que os softwares e os programas sejam estruturalmente sólidos e que funcionem no contexto técnico onde serão instalados. As técnicas de Testes Estruturais buscam garantir que o produto seja estruturalmente sólido e que funcione corretamente, o foco dos testes é averiguar o comportamento do sistema em determinadas situações. Os tipos de testes estruturais incluem teste de carga, particionamento de equivalência, conformidade, desempenho (performance), entre outros.

Técnicas de Teste Estrutural

- Testes de estresse;
- Testes de execução;
- Testes de operação;
- Testes de recuperação (contingência);
- Testes de segurança;
- Testes de sobrevivência.



3.Explique o critério de teste funcional particionamento em classes de equivalência?

R: O particionamento de equivalência é uma técnica de testes com base em requisitos, também conhecida como *Black Box*. Significa que ao utilizar esta técnica, a pessoa que testa aplicações passa a ter noção clara de cobertura de testes a partir de requisitos e especificações.

Este tipo de técnica não requer conhecimento dos caminhos internos, estrutura ou implementação do software sob teste.

Esta técnica é usada para reduzir o número de casos de teste a um nível gerenciável, mantendo ainda uma cobertura razoável do teste. Esta técnica simples é usada intuitivamente por quase todos os testadores, embora eles possam não estar cientes disso como um método formal de design de teste. Isso faz com que seus testes sejam menos eficazes, gerando duplicidades, ambiguidades ou ausência de cobertura de algumas faixas de valores.

O particionamento de equivalência divide as entradas do usuário na aplicação em partições (também conhecidas como classes de equivalência) e então as divide em faixas de valores possíveis, para que então, um desses valores seja eleito como base para o teste. Existem partições de equivalência para valores válidos e inválidos.

Valores válidos são valores que devem ser aceitos pelo sistema. Valores inválidos são valores que devem ser rejeitados pelo sistema.

Considere que na sua empresa há um sistema de recursos humanos que processa pedidos de emprego com base na idade de uma pessoa e que possui as seguintes regras de negócio:

- Pessoas menores de 16 anos não devem trabalhar;

- Pessoas entre 16 e 65 anos podem trabalhar;

- Pessoas com mais de 65 anos não podem trabalhar;

Dividindo estas regras em entradas possíveis para o sistema, teremos: considerando a Partição de Equivalência {16 a 65} anos, as outras são consideradas invalidas. Para obter uma cobertura de 100% com essa técnica, os casos de teste devem cobrir todas as partições identificadas (incluindo partições inválidas) usando no mínimo um valor de cada partição.

Com esses valores, esperamos que se um caso de teste em uma classe de equivalência detectar um defeito, todos os outros casos de teste na mesma classe de equivalência provavelmente detectarão o mesmo defeito, com seu inverso também em verdade.