Módulo 2 - Estudo Individual

Aluno: Diego Carlito Rodrigues de Souza

Matrícula: 221007690

Turma: 01

Grupo: Grupo 05 - Microondas

The Art of Software Testing

Test-Case Design: Black-Box Testing

Objetivo do teste de caixa preta

- Encontrar áreas do programa que não se comportam de acordo com as especificações.
- Selecionar o subconjunto certo de casos de teste com maior probabilidade de encontrar o máximo de erros.

Propriedades para encontrar casos de teste bem selecionados

1. **Redução de casos de teste**: Cada caso de teste deve reduzir a quantidade de outros casos necessários para atingir objetivos de teste pré-definidos.
2. **Ampla cobertura**: O caso de teste deve cobrir um amplo conjunto de outros possíveis casos de teste.

Metodologia conhecida como Partição de Equivalência

- Dividir cada condição de entrada em grupos para identificar classes de equivalência.
- Escrever casos de teste que cubram todas as classes de equivalência válidas e inválidas.

Análise de Valor de Fronteira (Boundary Value Analysis)

- Seleção de elementos em cada borda das classes de equivalência.
- Consideração do espaço de saída (classes de equivalência de saída).

Diretrizes para análise de valor de fronteira

- Testar valores nas fronteiras dos intervalos de entrada e saída.
- Cobrir valores mínimo e máximo de cada condição de entrada.
- Verificar elementos no início e fim de conjuntos ordenados.
- Considerar casos não óbvios para explorar outras condições de fronteira.

Grafos de Causa-Efeito

- Ajudam na seleção sistemática de um conjunto de casos de teste de alto rendimento.
- Representam condições de entrada (causas) e resultados esperados (efeitos) como um circuito lógico digital.
- Conectam causas e efeitos usando operadores lógicos como *AND*, *OR* e *NOT*.
- Permitem cobrir uma ampla gama de cenários de teste, identificando combinações complexas de entradas que podem causar falhas no sistema.

Os grafos de causa-efeito ajudam a melhorar a cobertura de testes, identificar possíveis problemas e tornam as especificações mais precisas e claras.

Higher-Order Testing

Higher-Order Testing é uma abordagem adicional de testes que complementa os testes de módulo e função para garantir que o software atenda às expectativas dos usuários finais e objetivos gerais do produto.

O processo de teste de ordem superior é dividido em sete etapas:

1. Traduzir as necessidades do usuário em requisitos escritos.
2. Traduzir requisitos em objetivos específicos, considerando viabilidade e prioridades.
3. Traduzir objetivos em especificações externas, considerando interações com o usuário.
4. Projetar o sistema particionando em programas, componentes ou subsistemas.
5. Projetar a estrutura do programa, definindo a função de cada módulo.
6. Desenvolver especificação precisa para a interface e função de cada módulo.
7. Traduzir especificações para algoritmos do código-fonte.

Tipos de teste:

- **Teste de módulo**: Encontra discrepâncias entre módulos e suas especificações.
- **Teste de função**: Mostra se o programa atende às especificações externas.
- **Teste de sistema**: Verifica se o produto atende aos objetivos originais.

Os testes de sistema se concentram em áreas não cobertas por testes de módulo e função, como capacidade, volume, estresse, usabilidade, segurança, performance, armazenamento, configuração, compatibilidade, instalação, confiabilidade, recuperação, manutenção, documentação e procedimentos.

- **Capacidade**: Garantir que todas as funcionalidades descritas nos objetivos do produto sejam implementadas corretamente.
- **Volume**: Avaliar o programa com volumes grandes de dados para testar sua eficiência e estabilidade.
- **Estresse**: Submeter o programa a cargas elevadas ou situações de estresse para testar sua robustez.
- **Usabilidade**: Testar a facilidade de uso do programa para os usuários finais.
- **Segurança**: Avaliar a segurança do programa para evitar vulnerabilidades.
- **Performance**: Verificar se o programa atende aos requisitos de tempo de resposta e transferência de dados.
- **Armazenamento**: Avaliar o gerenciamento de armazenamento do programa.
- **Configuração**: Testar se o programa funciona adequadamente nas configurações recomendadas.
- **Compatibilidade**: Garantir que o programa é compatível com versões anteriores ou outros sistemas.
- **Instalação**: Avaliar a facilidade e eficácia dos métodos de instalação do programa.
- **Confiabilidade**: Verificar se o programa atende às especificações de confiabilidade, como tempo de atividade e MTBF (tempo médio entre falhas).
- **Recuperação**: Testar os recursos de recuperação do sistema em caso de falhas.
- **Manutenção/Serviços**: Verificar se o programa oferece mecanismos adequados para suporte técnico.
- **Documentação**: Validar a precisão de toda a documentação do usuário.
- **Procedimentos**: Avaliar a precisão de procedimentos especiais necessários para usar ou manter o programa.

Teste de Aceitação

- **Definição**: Compara o programa aos requisitos iniciais e às necessidades atuais dos usuários finais.
- **Responsabilidade**: Realizado pelo cliente ou usuário final com o apoio do desenvolvedor.
- **Tipos de aceitação**:
  - Aceitação operacional: Sistema atende às necessidades básicas.
  - Aceitação comercial: Sistema agrega valor suficiente para justificar o investimento.
- **Critérios de aceitação**: Predefinidos e documentados entre cliente e desenvolvedor.
- **Benefícios**: Garante que o sistema atenda às expectativas do cliente, reduz retrabalhos e custos futuros, e promove uma transição suave do desenvolvimento para a operação.

Teste de Instalação

- **Objetivo**: Não é encontrar erros no software, mas verificar falhas durante a instalação.

- **Aspectos verificados**: Seleção de opções pelo usuário, alocação de arquivos, verificação da configuração de hardware, estabelecimento de conexões de rede.
- **Erros identificados**: Arquivos ausentes, permissões incorretas, configurações incompatíveis, falhas de rede.
- **Vantagens**: Evita problemas de pós-implementação, reduz custos de suporte, garante uma experiência positiva para o usuário final.

Planejamento e Controle de Testes

- **Desafios**: Gerenciamento de milhares de casos de teste e correções de erros.
- **Plano de testes**: Inclui objetivos, cronogramas, responsabilidades, biblioteca de casos de teste, ferramentas, tempo de computador, configuração de hardware, integração, procedimentos de monitoramento e depuração, teste de regressão.
- **Critérios de conclusão**: Podem ser baseados em metodologias de criação de casos de teste ou detecção de erros, ou uma combinação de ambos.

Agência Independente de Testes

- **Benefícios**:
    - Maior motivação: Equipe externa focada em encontrar falhas.
    - Competição saudável: Incentiva alta qualidade no software.
    - Independência gerencial: Livre de influências da gerência de desenvolvimento.
    - Especialização: Conhecimento e experiência específicos em testes de software.