TÉCNICAS DE TESTE DE SOFTWARE E ESTRATÉGIAS DE TESTE

A atividade de teste de software é um elemento crítico da garantia de qualidade de software e representa a última revisão de especificação, projeto e codificação.

Objetivos da Atividade de Teste

1. A atividade de teste é o processo de executar um programa com a intenção de descobrir um erro.
2. Um bom caso de teste é aquele que tem uma elevada probabilidade de revelar um erro ainda não descoberto.
3. Um teste bem sucedido é aquele que revela um erro ainda não descoberto.

Teste de Caminho Básico

O método de caminho básico possibilita que o projetista do caso de teste derive uma medida da complexidade lógica de um projeto procedimental e use essa medida como guia para definir um conjunto básico de caminhos de execução.

Teste de Estrutura de Controle

Estes testes ampliam a cobertura dos testes e melhoram a qualidade dos testes.

- **Teste de condição:** Põe em prova as condições lógicas contidas num módulo de programa. Uma condição simples é uma variável booleana ou uma expressão relacional, possivelmente precedida por um operador NOT (') . Uma expressão relacional assume a forma *E1<operador relacional>E2.* Uma condição composta é constituída de duas ou mais condições simples, operadores booleanos e parênteses, OR (|). AND (&) e NOT ('). Uma condição sem expressões relacionais é chamada expressão booleana.

  O método de teste de condição concentra-se em testar cada condição do programa.

- **Teste de Fluxo de Dados**: Seleciona caminhos de teste de um programa de acordo com as localizações das definições e usos de variáveis no programa.
- **Teste de Laços**: Se concentra exclusivamente na validade das construções de laços, laços simples, aninhados, concatenados, não-estruturados.

Particionamento de Equivalência

Este método divide o domínio de entrada de um programa em classes de dados a partir das quais os casos de teste podem ser derivados. O particionamento de equivalência procura definir um caso de teste que descubra classes de erros, assim reduzindo o número total de casos de teste que devem ser desenvolvidos. Uma classe de equivalência representa um conjunto de estados válidos para condições de entrada.

Análise de Valor Limite BVA

A análise de valor limite leva à escolha de casos de teste que põem à prova os valores fronteiriços. A análise de valor limita é uma técnica de projeto de casos de teste que complementa o particionamento de equivalência. Em vez de selecionar qualquer elemento de uma classe de equivalência, a BVA leva à seleção de casos de testes nas extremidades da classe. Em vez de se concentrar somente nas condições de entrada, a BVA deriva os casos de teste também do domínio de saída.

Teste de Sistemas de Tempo Real

Neste tipo de deste, encontramos o elemento de combinação alinhado ao teste que é o tempo. Os testes de software devem levar em consideração o impacto das falhas de hardware sobre o processamento do software. Tais falhas podem ser extremamente difíceis de ser simuladas realísticamente.

Estratégia global para sistemas de tempo real:

1. *teste de tarefas:* O primeiro passo da atividade de testes de tempo real consiste em testar cada tarefa independentemente. Este teste revela erros de lógica e de função, mas não revelará erros comportamentais ou de timing.
2. *Teste comportamental:* É possível simular o comportamento de um sistema de tempo real e examinar seu comportamento como uma conseqüência de eventos externos. O comportamento do software é examinado a fim de detectar erros de comportamento.
3. *Teste intertarefas:* As tarefas assíncronas, que sabidamente comunicam-se entre si, são testados com diferentes taxas de dados e cartas de processamento para determinar se ocorrerão erros de sincronização intertarefas.
4. *Teste do sistema:* O software e o software são integrados e uma variedade completa de testes de sistema é levada a efeito, numa

tentativa de descobrir erros na interface software/hardware.

ESTRATÉGIAS DE TESTE DE SOFTWARE

Um esqueleto (template) de teste de software deve ser definido para o processo de engenharia do software. Esqueleto é um conjunto de passos no qual podemos alocar técnicas de projeto de casos de teste e métodos de teste específico.

Uma estratégia de teste de software deve acomodar teste de baixo nível que seja necessário para verificar se um segmento de código-fonte foi corretamente implementado, bem como teste de alto nível que valide funções importantes do sistema contra requisitos do cliente.

Verificação e Validação

A verificação refere-se ao conjunto de atividades que garante que o software implemente corretamente uma função específica. A validação refere-se a um conjunto diferente de atividades que garante que o software que foi construído é rastreável às exigências do cliente.

Os métodos de análise, projeto e implementação (codificação) atuam no sentido de aumentar a qualidade ao oferecer técnicas uniformes e resultados previsíveis.

A análise e o projeto de software (juntamente com a codificação) são tarefas construtivas , ou seja, o engenheiro de software cria um programa de computador, sua documentação e estruturas de dados correlatas. Do ponto de vista do construtor, a atividade de teste pode ser destrutiva, pois sempre vão ser descobertos erros no programa.

Uma Estratégia de Teste de Software

Teste unitário, este teste concentra-se em cada unidade de software de acordo com a implementação do código-fonte, ele exercita caminhos específicos da estrutura de controle de um módulo, a fim de garantir uma completa cobertura e máxima detecção de erros.

Teste de integração, a atenção encontra-se no projeto e na construção da arquitetira de software, este teste cuida das questões associadas aos duplos problemas da verificação e construção em programas.

Teste de validação, os requisitos estabelecidos como parte da análise de requisitos de software são validados em relação ao software que foi construído, o teste de validação oferece a garantia final de que o software atende a todas as exigências funcionais, comportamentais e de desempenho.

Teste de sistema, neste teste o software e outros elementos do sistema são testados como um todo, ele verifica se todos os elementos combinam-se adequadamente e se a função/desempenho global do sistema é conseguida.

Critérios para a Conclusão de Teste

A atividade de teste jamais terá completada, a carga simplesmente transfere-se de você (o projetista) para o cliente.

Teste de Unidade

O teste de unidade concentra-se no esforço de verificação da menor unidade de projeto de software - o módulo. Caminhos de controle importantes são testados para descobrirem erros dentro das fronteiras do módulo.

A interface com o módulo é testada para ter a garantia de que as informações fluem para dentro e para fora da unidade de programa que se encontra sob teste. A estrutura de dados local é examinada para ter a garantia de que os dados armazenados temporariarmente mantêm sua integridade durante todos os passos de execução de um algoritmo. As condições de limite são testadas para ter a garantia de que o módulo opera adequadamente nos limites estabelecidos para demarcarem ou restringirem o processamento.

Entre os erros mais comuns de computação estão:

1. precedência aritmética incorreta ou mal compreendida;
2. operações em modo misto;
3. inicialização incorreta;

Os casos de teste devem descobrir erros tais como:

1. comparação de diferentes tipos de dados;
2. operadores lógicos ou precedência incorretos;
3. término do laço impróprio ou inexistente;

Teste de Integração

Este teste é uma técnica sistemática para a construção da estrutura de programa realizando-se, ao mesmo tempo, testes para descobrir erros associados a interfaces. O objetivo é, a partir dos módulos testados no nível de unidade, construir a estrutura de programa que foi determinada pelo projeto.

Freqüentemente, existe uma tendência para tentar a integração não-incremental; ou seja, construir o programa usando uma abordagem "big bang" onde todos os módulos são combinados antecipadamente. O programa completo é testado como um todo.

A integração incremental é a antítese do big bang. O programa é construído e testado em pequenos segmentos, onde os erros são mais fáceis de ser isolados e corrigidos; as interfaces têm maior probabilidade de ser testadas completamente; e uma abordagem sistemática ao teste pode ser aplicada.

Integração Top-Down: é uma abrdagem incremental à construção das estruturas de programa. Os módulos são integrados movimentando-se de cima para baixo através da hierarquia de controle, iniciando-se do módulo de controle principal(programa principal).Os módulos subordinados ao módulo de controle principal são incorporados à estrutura de uma maneira depht-first (primeiramente pela profundidade) ou breadth-first (primeiramente pela largura).

Integração Bottom-Up: inicia a construção e os testes com módulos atômicos (isto é, módulos localizados nos níveis mais baixos da estrutura de programa). Uma vez que os módulos são integrados de baixo para cima, o processamento exigido para os módulos subordinados em determinado nível está sempre disponível, e a necessidade de stubs é eliminada.

A maior desvantagem da abordagem top-down é a necessidade de ter stubs e as dificuldades de teste resultantes que podem ser compensados pela vantagem de testar logo as principais funções de controle. A maior desvantagem da integração bootom-up é que o programa não existe como entidade até que o último módulo seja adicionado.

Os critérios que estão citados a seguir são aplicados a todas as fases de testes:

Integridade de interface: As interfaces internas e externas são testadas à medida que cada módulo é incorporado à estrutura.

Validade funcional: Testes projetados para a descoberta de erros funcionais são aplicados.

Conteúdo de informação: Testes projetados para a descoberta de erros associados às estruturas de dados globais e locais são aplicados.

Desempenho: Testes projetados para a verificação dos limites de desempenho estabelecidos durante o projeto de software são aplicados.

Teste de Sistema

O teste de sistema é na verdade , uma série de diferentes testes, cujo propósito primordial é por completamente à prova o sistema baseado em computador. Não obstante cada teste tenha uma finalidade diferente, todo o trabalho deve verificar se todos os elementos do sistema foram adequadamente integrados e realizam as funções atribuídas.

Teste de Recuperação

O teste de recuperação é um teste de sistema que força o software a falhar de diversas maneiras e verifica se a recuperação é adequadamente executada.

Teste de Segurança

Este teste tenta verificar se todos os mecanismos de proteção embutidos num sistema o protegerão, de acessos indevidos.

Durante o teste de segurança, o analista desempenha papéis de pessoas que desejam penetrar no sistema, qualquer coisa vale, tentando desarmar o sistema, e derrubar as defesas que tenham sido construídas.

Teste de Desempenho

É idealizado para testar o desempenho de run-time do software dentro do contexto de um sistema integrado. O teste de desempenho ocorre ao longo de todos os passos do processo de teste.

Estudo Teórico e Aplicado de Critérios de Teste e Validação na Produção de Software

Consiste em estudo teórico e empírico, definição e comparação de critérios de teste e implementação de ferramentas de software de apoio às atividades de teste na produção de software. Atividades de comparação entre critérios de teste funcional, estrutural , com ênfase em critérios baseados em análise de fluxo de dados, e em critérios baseados em erros, com ênfase na Análise de Mutantes, também são conduzidos dentro do escopo desta linha. Todos esses aspectos são explorados tanto no nível de teste de unidade como no teste de integração, assim como à luz do paradigma de desenvolvimento de software orientado a objeto.