Estratégias Para Teste

Pasteur Ottoni de Miranda Junior – PUC Minas
Postado em www.pasteurjr.blogspot.br

1-Visão Global de Estratégias Para Teste

A estratégia definida para testes comporta os seguintes passos implementados sequencialmente:
-*Testes de módulos atômicos:* cada módulo(procedimentos ou funções) que não invoca outros módulos é testado individualmente. Faz muito uso das técnicas de Teste de Caixa Branca, exercitando caminhos específicos da estrutura de controle de um módulo, a fim de garantir uma completa cobertura e máxima detecção de erros.
-*Testes de Integração:* consiste na montagem e integração dos módulos básicos a fim de formarem um pacote de software.
-*Testes de Validação:* consiste na realização de testes funcionais, baseados na especificação de requisitos funcionais, comportamentais e de desempenho. Os Critérios de Validação definidos após a fase de Análise de Requisitos devem ser testados. Utilizam-se exclusivamente Testes de Caixa Preta.
-*Testes de Sistema:* o software, uma vez validado, é combinado com outros elementos do sistema(por exemplo, hardware, pessoas, bancos de dados). Verificam se todos os elementos combinam-se adequadamente e se a função/desempenho global do sistema é conseguida.
-*Testes de Aceitação:* consiste no processo de comparar o programa com seus requisitos iniciais e as necessidades dos usuários finais. É usualmente realizado pelo usuário final. Também denominado de Alfa- Teste (realizado nas dependências do desenvolvedor) ou Beta-Teste (realizado nas dependências do usuário final).

2-Teste de Módulos

Para testes de módulos sugere-se que os seguintes aspectos sejam considerados:
-as *condições de limite* são testadas para ter a garantia de que o módulo opera adequadamente nos limites estabelecidos.
-todos os *caminhos independentes* são testados;
-realizar *testes de cobertura* quando desejar-se confiabilidade adicional nos testes.
-testar caminhos no fluxo de execução do programa que envolvem tratamento de erros.
Para executar tais módulos, programas que os invocam devem ser criados. Tais programas são denominados *drives.*

3-Teste de Integração

Para integração dos módulos, sugere-se uma abordagem *botton-up.* Esta abordagem não requer a criação dos chamados *stubs*-estruturas de programa criadas para substituir módulos subordinados ao módulo sob teste.
A integração botton-up começa com os módulos atômicos, de baixo para cima. Assim sendo, o processamento exigido para os módulos subordinados em determinado nível está sempre disponível.
A estratégia botton up pressupõe as seguintes etapas:
1.Módulos de baixo nível são testados utilizando-se teste de módulos

2-Módulos são agrupados em *clusters* que realizam uma subfunção específica do software.
3-Um *drive* é escrito para coordenar a entrada/saída do caso de teste.
4-O cluster é testado.
5-*Drives* são removidos e clusters são combinados movendo-se em direção ao topo da hierarquia da estrutura do programa.

Quando a necessidade de teste de funções de controle de nível mais alto for premente, uma integração *top-down* pode ser utilizada. Esta integração envolve os seguintes passos:
1-Começa por módulos de nível mais alto, caminhando pela árvore de módulos primeiro em profundidade ou primeiro em largura.
2-Esta implementação demanda a utilização dos *stubs* mencionados anteriormente para substituir módulos subordinados não testados ainda.
3-Após os testes, estes *stubs* vão sendo substituídos pelos módulos reais e novos *stubs* vão sendo criados até que se chegue aos módulos de nível mais baixo.
4-Como cada novo módulo introduzido na integração pode gerar erros devido a, por exemplo, novos caminhos estabelecidos, novas entradas saídas ou novos fluxos de controle lógico, testes de *regressão* devem ser realizados. Tais testes realizam a re-execução de alguns testes já realizados, de forma a garantir que as modificações não produziram efeitos colaterais. Podem ser focados em uma amostra dos testes que exercitarão todas as funções do software ou em funções que são prováveis de terem sido afetadas pela mudança ou ainda nos componentes modificados.

A maior desvantagem da abordagem top-down é a necessidade de stubs. A vantagem é que são funções de controle mais importantes (em nível mais alto) são testadas antes. No caso da botton-up, a desvantagem é que o programa somente existirá funcionando após o último módulo ser testado. Por outro lado, não demanda stubs.
Módulos considerados críticos (de performance crítica, ou que englobem muitos requisitos, ou de alto nível de controle ou complexos) devem ser testados antes.

Normalmente utilizam-se métodos de Caixa Branca para integração, porém em algumas situações podem ser utilizados métodos de caixa preta, por exemplo, quando o código do módulo for longo.

4-Teste de Validação

A validação do software é feita por meio de uma série de Testes de Caixa Preta que demonstram a conformidade com os requisitos. Os casos de teste devem ser projetados no DTS de acordo com com os Critérios de Validação definidos na Especificação de Requisitos de Software.
Depois que cada caso de teste de validação for realizado, existirá uma dentre duas condições: (1) as características de função ou desempenho conformam-se às especificações e são aceitas; ou (2) um desvio das especificações é descoberto e uma lista de deficiências é criada. Os desvios ou erros descobertos nessa fase de um projeto raramente podem ser corrigidos antes da conclusão programada. Muitas vezes é necessário negociar com o cliente para estabelecer um método para superação das deficiências.

5-Teste de Sistema

Os testes de sistema têm como propósito colocar à prova o sistema. A seguir serão descritos os tipos de testes de sistema que deverão ser implementados.

5.1- Testes de Recuperação

Tais testes devem ser feitos de forma a forçar o software a falhar de diversas maneiras, verificando se a recuperação de tais falhas é adequadamente executada.

5.2-Testes de Segurança

Tais testes devem ser realizados de forma a verificar se todos os mecanismos de proteção embutidos num sistema o protegerão , de fato, de acessos indevidos.
Os testes devem ser planejados para testar, por exemplo, segurança de senhas, proteção quanto ao acesso de determinadas funções, proteção dos dados durante queda do sistema, etc.

5.3-Testes de Stress

Consiste em testar-se o programa em situações limites, até que o mesmo falhe.
O teste de stress executa o sistema de forma a exigir recursos em quantidade, volume ou frequência anormais. Por exemplo, limites de memória, volume grande de dados de entrada, pesquisas longas e exaustivas em disco, etc.

5.4-Testes de Desempenho

Estes testes são realizados para se testarem os requisitos de desempenho definidos na Especificação de Requisitos. O teste de desempenho deve ocorrer ao longo de todos os passos do processo de teste. Até mesmo a nível de unidade, o desempenho de um módulo individual pode ser avaliado quando se realizam testes de caixa branca.. Porém, somente quando todos os elementos de um sistema estão plenamente integrados é que o desempenho real do mesmo pode ser avaliado.
Muitas vezes é realizado combinado com testes de stress e frequentemente exige instrumentação de hardware e software. Ou seja, muitas vezes é necessário medir a utilização de recursos (por exemplo, ciclos do processador) rigorosamente. A instrumentação externa pode monitorar intervalos de execução, registrar eventos (por exemplo, interrupções) quando eles ocorrerem e amostrar estados de máquina. Ao instrumentar um programa, o realizador do teste pode descobrir situações que levam à degradação e possível falha do sistema.

6-Testes de Aceitação

Estes testes visam especificamente capacitar clientes a validar seus requisitos. É realizado quase que exclusivamente pelo usuário final. Devem ser realizados ao longo de um período de semanas ou meses, assim descobrindo erros que passaram por testes posteriores ou que poderiam deteriorar o sistema com o decorrer do tempo.

O chamado *alfa teste* é realizado por um cliente normalmente nas instalações do desenvolvedor. O software é usado num ambiente natural , com o desenvolvedor registrando erros e problemas de uso.
O chamado *beta teste* é realizado em instalações de clientes por usuários finais do software. O desenvolvedor, normalmente não está presente. O cliente registra

Referências.

PRESSMANN, R. *Software Engineering – A Practitioners Approach.* 7a edição. 2008. Mc Graw-Hill