Engenharia de Software II

Aula 8

http://www.ic.uff.br/~bianca/engsoft2/

Ementa

- Processos de desenvolvimento de software
- **Estratégias e técnicas de teste de software** (Caps. 13 e 14 do Pressman)
- Métricas para software
- Gestão de projetos de software: conceitos, métricas, estimativas, cronogramação, gestão de risco, gestão de qualidade e gestão de modificações
- Reengenharia e engenharia reversa

Estratégia Global

1. Teste de unidade
   - Focaliza cada componente individualmente, garantido que funciona.
   - Faz uso intensivo de técnicas que exercitam caminhos específicos na estrutura de controle.
2. Teste de integração
   - Focaliza o pacote de software completo e trata da verificação do programa como um todo.
   - Faz uso de técnicas de projeto de casos de teste que enfocam as entradas e saídas, além de exercitar caminhos específicos.
3. Teste de validação
   - Critérios de avaliação estabelecidos durante a análise de requisitos são avaliados.
4. Teste de sistema
   - Testa a combinação do software com outros elementos do sistema (como hardware, pessoal e bancos de dados).
   - Verifica se a função/desempenho global do sistema é alcançada.

Teste de Validação

- Começa no fim do teste de integração.
  - Componentes individuais já foram exercitados.
  - O software está completamente montado como um pacote.
  - Erros de interface já foram descobertos e corrigidos.
- A validação é bem-sucedida quando o software funciona do modo esperado pelo cliente.
  - As expectativas razoáveis são as definidas no documento de *Especificação de Requisitos do Software*.
  - O documento deve conter uma seção de *Critérios de Validação*.

Critérios do Teste de Validação

- A validação é conseguida por intermédio de uma série de testes que demonstram conformidade com os requisitos.
- Os testes são projetados para garantir que:
  - Os requisitos funcionais sejam satisfeitos.
  - Os requisitos de desempenho sejam alcançados.
  - A documentação esteja completa e correta.
  - Usabilidade e outros requisitos sejam satisfeitos.
    - Exemplo: transportabilidade, compatibilidade, etc.
- Caso haja desvios, uma lista de deficiências é criada.
  - Raramente erros descobertos nesse estágio podem ser corrigidos antes da entrega.
  - É necessário negociar com o cliente para estabelecer um método de resolução de deficiências.

Validação: Revisão da Configuração

- Também chamada de auditoria.
- Garante que todos os elementos da configuração de software tenham sido adequadamente desenvolvidos e catalogados.
  - Código-fonte e executável.
  - Documentação.
  - Ferramentas que foram usadas no desenvolvimento.

Validação: Testes de Aceitação

- É impossível para um desenvolvedor prever como o cliente usará o software.
  - Instruções de uso podem ser mal interpretadas.
  - Uma saída que parecia clara para o desenvolvedor pode ser ininteligível para o usuário.
- Por esse motivo, testes de aceitação devem ser definidos pelo cliente e conduzidos pelo usuário final, a fim de validar todos os requisitos do cliente.
  - Pode ser conduzido ao longo de um período de semanas ou meses.

Validação: Testes Alfa e Beta

- Se o software é desenvolvido como um produto a ser usado por vários clientes, é difícil realizar testes de aceitação com cada um.
  - Nesse caso usa-se os testes alfa e beta.
- Teste alfa: conduzido na instalação do desenvolvedor com os usuários finais.
  - O desenvolvedor "olha sobre o ombro" dos usuários para registrar erros e problemas de uso.
- Teste beta: conduzido nas instalações dos usuários finais.
  - O desenvolvedor não está presente.
  - O cliente registra todos os problemas (reais ou imaginários) e os relata ao desenvolvedor em intervalos regulares.
  - Depois que os problemas são corrigidos, o produto é liberado para toda base de clientes.

Teste de Sistema

- O software é apenas um elemento de um sistema maior.
  - Outros elementos do sistema: hardware, pessoal, dados, sistema operacional.
- Uma série de testes de integração e validação do sistema devem ser conduzidos.
- Tipos de teste de sistema:
  - Teste de recuperação.
  - Teste de segurança.
  - Teste de estresse.
  - Teste de desempenho.

Teste de Recuperação

- É um teste de sistema que força o software a falhar de diversos modos e verifica se a recuperação é adequadamente realizada.
- Se a recuperação é automática:
  - A reinicialização e a recuperação dos dados são avaliados.
- Se a recuperação requer intervenção humana:
  - O tempo médio para o reparo é avaliado para determinar se está dentro de limites aceitáveis.

Teste de Segurança

- Verifica se os mecanismos de proteção incorporados a um sistema vão de fato protegê-lo de:
  - *Hackers* que tentam invadí-lo por esporte.
  - Ex-empregados que tentam invadí-lo por vingança.
  - Indivíduos que tentam invadí-lo em busca de ganhos pessoais ilícitos.
- O testador deve desempenhar o papel do indivíduo que deseja invadir o sistema.
  - Deve tentar obter senhas de funcionários externos.
  - Deve tentar tomar o sistema negando serviço a outros.
  - Deve tentar causar erros no sistema para invadí-lo durante a recuperação.
- Com tempo e recursos suficientes, o bom teste de segurança vai acabar invadindo o sistema.

Teste de Estresse

- São projetados para submeter programas a situações anormais e verificar se eles ainda funcionam.
- Executa o programa de tal forma que demande recursos em quantidade ou freqüência anormais:
  - A velocidade de entrada de dados pode ser muito aumentada ou diminuída.
  - A memória do sistema pode ser diminuída.
  - Casos de teste que podem causar busca excessiva de dados residentes em disco são criados.

Teste de Desempenho

- É projetado para testar o desempenho do software durante a execução, no contexto de um sistema integrado.
- Requerem instrumentação de hardware e software.
  - É necessário medir de modo preciso a utilização de recursos (memória, processamento).
- Importante para sistemas de tempo real e embutidos.

Técnicas de Teste de Software

- Fornecem diretrizes sistemáticas para projetar casos de testes que:
  - Exercitam a lógica interna e as interfaces de cada componente de software.
  - Exercitam os domínios de entrada e saída do programa para descobrir erros na função, no comportamento e no desempenho do programa.

Características de Testabilidade

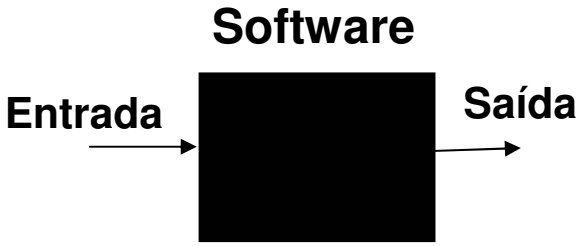
- As seguintes características levam a um software testável e devem ser levadas em conta quando ele é construído:
  - Operabilidade: o software deve ser projetado e implementado com qualidade.
  - Observabilidade: variáveis devem ser visíveis ou consultáveis durante a execução.
  - Controlabilidade: variáveis devem poder ser controladas diretamente pelo engenheiro de teste.
  - Decomponibilidade: módulos independentes podem ser testados independentemente.
  - Simplicidade: o conjunto de características deve ser o mínimo necessário para atender os requisitos.
  - Estabilidade: o software se recupera bem de falhas.
  - Compreensibilidade: a documentação técnica é acessível instantaneamente, bem organizada, específica, detalhada e precisa.

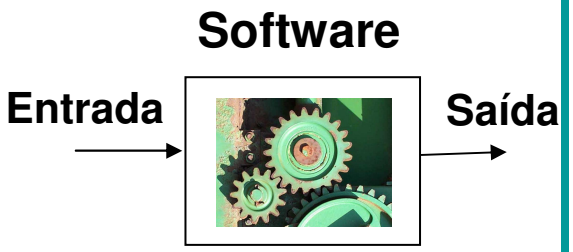
Características do Teste

- Um bom teste deve ter os seguintes atributos:
  1. Alta probabilidade de encontrar um erro.
     - O testador deve entender o software.
     - As diferentes classes de falha devem ser investigadas.
  2. Não deve ser redundante.
     - Tempo e recursos para testes são limitados.
     - Ex.: testar várias senhas inválidas diferentes é redundante.
  3. Não deve ser muito simples nem muito complexo.
     - Cada teste deve buscar encontrar um tipo de erro.

Testes Caixa-Preta e Caixa-Branca

- Testes Caixa-Preta
  - São conduzidos na interface do software, sem preocupação com a estrutura lógica interna do software.
- Testes Caixa-Branca
  - São baseados em um exame rigoroso do detalhe procedimental.
  - Caminhos lógicos e colaborações entre componentes são testadas.





Teste Exaustivo (impraticável!)

- À primeira vista pode parecer que um teste caixa-branca bastante rigoroso levaria a programas 100% corretos.
  - "Apenas" seria necessário gerar casos de teste para exercitar toda a lógica do programa.
  - Exemplo: programa de 100 linhas de código C com dois loops aninhados (de tamanho variável), com quatro if-then-else dentro do loop interno.
    - 10^{14} caminhos de execução possíveis.
    - 3.170 anos para testar todos os caminhos em um computador atual.

Técnicas de Teste Caixa-Branca

- Usando métodos de teste caixa-branca, podemos derivar casos de teste que:
  1. Garantam que todos os caminhos independentes de um módulo sejam executados pelo menos uma vez.
  2. Exercitem todas as decisões lógicas de seu lado verdadeiro e falso.
  3. Executem todos os ciclos (loops) nos seus limites e dentro de seus intervalos operacionais.
  4. Exercitem as estruturas de dados internas.
- Tipos de teste caixa-branca:
  - Teste de Caminho Básico
  - Teste de Estrutura de Controle