Engenharia de Software Aula 17

Testes de Software

Eduardo Figueiredo
http://www.dcc.ufmg.br/~figueiredo
dcc603@gmail.com

7 Maio 2012

Desenvolvimento de Software

1. Especificação de requisitos
2. Projeto de Software
3. Implementação
4. Verificação e Validação
5. Evolução de software

Onde estamos...

1. Especificação de requisitos
2. Projeto de Software
3. Implementação
4. **Verificação e Validação**
5. Evolução de software

Tópicos da Aula

- Verificação e Validação
- Inspeção de Software
- Teste de Software
- Projeto de Casos de Teste

Testes de Software

- *"Testes podem somente revelar a presença de defeitos, não a ausência"*
  Dijkstra
- Testes fazem parte do processo de verificação e validação (V&V)
  - Devem ser usados em conjunto com a verificação estática (inspeção)

Verificação x Validação

- Verificação
  - Verifica se o software atende aos requisitos funcionais e não funcionais especificados
  - *Estamos construindo o produto corretamente?*
- Validação
  - Procura assegurar que o sistema atenda as expectativas e necessidades do cliente
  - *Estamos construindo o produto correto?*

Níveis de Confiabilidade

- Objetivo da V&V
  - Estabelecer a confiança de que o software está pronto para ser usado
- O nível de confiabilidade de software depende
  - Do próprio sistema
  - Das expectativas dos usuários
  - Do ambiente de mercado

Finalidade do Sistema

- Sistema Crítico
  - O nível de confiança é muito maior
  - O software deve ser confiável
- Protótipo
  - O nível de confiança é menor
  - O software pode falhar

Expectativa de Usuários

- Usuários podem já estar acostumados com software de baixa qualidade
- Usuários tendem a aceitar falhas quando os benefícios superam as desvantagens
  - A tolerância a falhas dos usuários diminuí a medida que o software amadurece

Ambiente de Mercado

- Sistemas comerciais devem levar em conta os programas concorrentes
- Quando não há concorrência
  - O sistema pode ser liberado mais cedo pelo pioneirismo
  - V&V pode não ter sido bem feita
- Quando clientes não querem pagar caro pelo produto
  - Eles geralmente aceitam alguns defeitos

Teste de Validação e Teste de Defeito

Verificação e Validação

- Tem por objetivo garantir que o sistema satisfaça os requisitos
- Consiste da realização de testes para encontrar erros
- A inexistência de erros não representa a *adequação operacional* do sistema
  - Deve ser feita a **validação** com o cliente

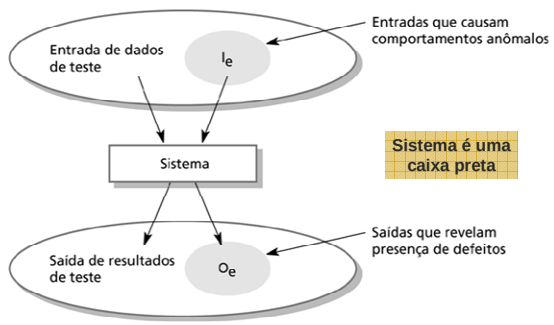
Teste de Validação

- Pretende mostrar que o software atende aos seus requisitos
  - Faz o que o cliente deseja
- Um teste bem sucedido mostra que o requisito foi implementado
- Podem ser usados testes estatísticos para garantir desempenho e confiabilidade

Teste de Defeito

- Destinado a revelar defeitos no sistema
- Um teste de defeitos bem sucedido é aquele que revela defeitos no sistema
- Não há uma distinção rígida entre Teste de Validação e Teste de Defeito

Modelo de Entrada e Saída



Inspeção e Depuração

Técnicas de V&V

- A aplicação de **testes de software** é a maneira mais comum de se verificar se o programa atende a sua especificação
  - Testes são apenas uma técnica de V&V
- Para complementar os testes, é comum o uso de **inspeção de software** ou **revisão por pares**

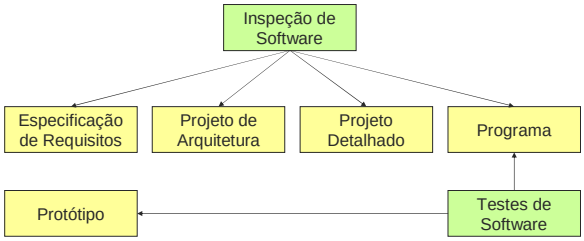
Inspeções e Revisões

- Técnica estática de V&V
- Pode ser usada em qualquer atividade de desenvolvimento
  - Requisitos, projeto, código fonte, etc.
- Pode ser semi-automatizada por análise estática
  - Análise não automatizada é cara

Testes de Software

- Teste é uma técnica dinâmica de V&V
- Consiste em executar uma implementação com dados de teste
- Além de verificar as saídas, pode ser usada para avaliar requisitos não funcionais
  - Desempenho, confiabilidade, segurança, etc.

Verificação Estática e Dinâmica



Vantagens da Inspeção

- Muitos defeitos diferentes podem ser descobertos em uma única inspeção
  - Um teste revela um defeito e oculta vários
- Versões incompletas do sistema podem ser inspecionadas
- Permite encontrar problemas em outros atributos de qualidade do software
  - Uso de um algoritmos mais eficiente, padrão de projeto, etc.

Limitações de Inspeção

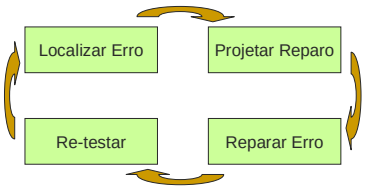
- Técnicas de inspeção podem somente verificar a correspondência entre o software e sua especificação
  - Não permite validar com o cliente
- Inspeção não é adequada
  - Para demonstrar se o software é útil
  - Para verificar requisitos não funcionais, como desempenho, segurança, etc.

V&V x Depuração

- V&V e depuração de defeitos são atividades distintas
- V&V está associada a constatação da existência de defeitos no programa
- Depuração está relacionado à localização desses defeitos

Depuração e Testes

- Depuração e testes costumar ser atividades cíclicas e intercaladas



Atividades e Recursos

- Resultados dos testes apóiam a localização de defeitos
- A especificação do sistema deve ser usada para projetar o reparo
- Testes automatizados facilitam a atividade de re-restar

Processo de Testes

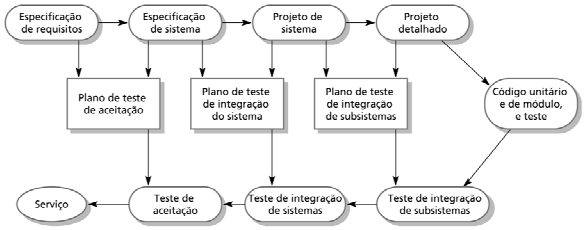
Dos Requisitos ao Teste

- Verificação e validação deve ocorrer durante e depois do desenvolvimento
- V&V ocorre em todas as atividades de desenvolvimento
  - Começa com revisões de requisitos
  - Revisões de arquitetura e projeto
  - Inspeções de código

O Modelo V

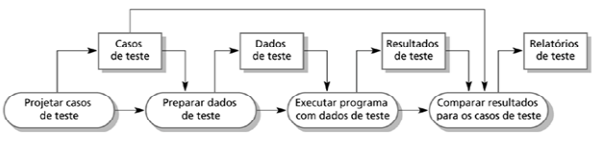
- Modelo que integra o desenvolvimento aos testes
- É fortemente baseado no Modelo Cascata
- Os planos de testes são derivados das atividades de desenvolvimento

O Modelo V



Processo de Teste

- Dados de teste, podem ser gerados automaticamente
- As saídas dos testes somente podem ser previstas por pessoas que conhecem o domínio de negócio



Algumas Políticas de Teste

- Todas as funções acessadas por menus devem ser testadas
- As combinações de funções dos mesmos menus devem ser testadas
- As funções devem ser testadas com entradas corretas e incorretas

Estágios de Teste

- Testes de Desenvolvimento
  - O sistema é testado durante o desenvolvimento (descobrir bugs)
- Testes de Release
  - Equipe independente testa uma versão completa do sistema (verificar os requisitos)
- Testes de Usuário
  - Usuários testam o software no ambiente real (aceitação do produto)

Testes de Desenvolvimento

Classificação

- Classificação dos testes de desenvolvimento
  - Teste Unitário
  - Teste de Componente (integração)
  - Teste de Sistema

Teste Unitário

- Objetivo é garantir que uma classe funciona
  - Testa classes individuais de programa
- Geralmente é de responsabilidade do próprio desenvolvedor da classe
  - Os testes são derivados da experiência do desenvolvedor

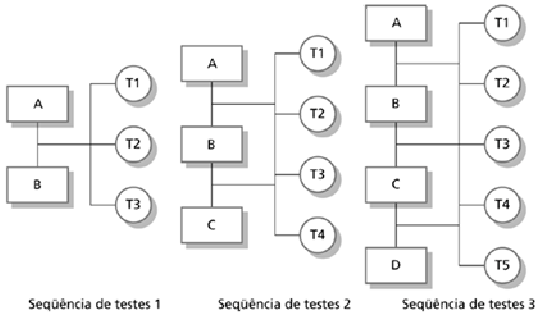
Teste de Classe (OO)

- O teste completo de uma classe de objetos requer
  - Teste de todas as operações associadas com um objeto
  - Atribuir e obter valores a todos os atributos de objeto
  - Exercício do objeto em todos os estados possíveis
- A herança dificulta o teste de classe

Teste de Componente

- O objetivo é garantir que dois ou mais componentes funcionam juntos
  - Testa grupos de componentes integrados para criar um sistema ou um subsistema
- A responsabilidade é de uma equipe independente de teste
  - Os testes são baseados em uma especificação de sistema

Testes e Integração Incremental



Tipos de Testes de Integração

- Integração *top-down*
  - Desenvolver o esqueleto do sistema e preenchê-lo com componentes
- Integração *bottom-up*
  - Integrar componentes de infra-estrutura e, depois, adicionar componentes funcionais

Teste de Sistema

- Testa uma release de sistema que será entregue ao cliente
- O objetivo é aumentar a confiança do fornecedor de que o sistema atende aos seus requisitos
- Teste funcional (caixa-preta)
  - Baseado somente na especificação
  - Testadores não conhecem a implementação

Diretrizes do Teste Caixa-Preta

- Escolher entradas que forcem o sistema a gerar as mensagens de erro
- Projetar entradas que causem estouro de pilha
- Forçar a geração de saídas inválidas
- Forçar resultados de cálculo a serem muito grandes ou muito pequenos

Teste de Release

Testes e Casos de Uso

- Casos de uso podem ser uma base para derivar os testes de um sistema
  - Ajudam a identificar as operações a serem testadas
  - Ajudam no projeto dos casos de teste
- As entradas e saídas podem ser identificadas em um Diagrama de Sequência do caso de uso

Testes de Desempenho e Estresse

- Testes de desempenho
  1. Planejamento de uma série de testes
  2. A carga é constantemente aumentada
  3. Verifica o limite em que desempenho do sistema se torne inaceitável
- Testes de estresse forçam o sistema além de sua carga máxima de projeto

Métodos Ágeis e Testes

- O teste é inseparável do desenvolvimento em alguns métodos ágeis (exemplo: XP)
  - Desenvolver primeiro os testes
  - Uso de ferramentas para testes automatizados
  - Envolvimento do cliente na escrita e validação dos testes, etc.

Projeto de Casos de Teste

Projeto de Testes

- Objetivo é é criar um conjunto de testes que sejam eficazes em validação e teste de defeitos
  - Envolve a definição de entradas e saídas usadas pelos casos de teste
- Abordagens de projeto de teste
  - Teste baseado em requisitos
  - Teste de partições (das entradas)
  - Teste estrutural

Teste Baseado em Requisitos

- Técnica de teste de validação
  - Considerando cada requisito, deriva-se um conjunto de testes
- Os requisitos devem ser testáveis

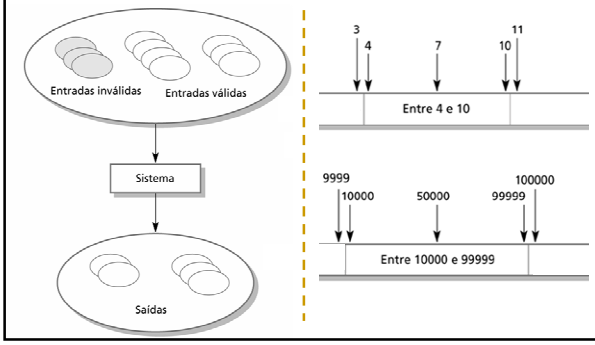
Teste de Partições

- Dados de entrada e resultados de saída podem ser particionado
  - O programa se comporta de maneira equivalente para cada partição
- Casos de teste devem ser escolhidos a partir de cada partição

Diretrizes do Teste de Partições

- Testar o software com sequências de comprimento zero
  - Testar com sequências de tamanhos extremos: um único valor e número máximo
- Usar sequências de tamanhos diferentes em testes diferentes
- Derivar testes de tal modo que o primeiro, o médio e o último elementos da sequência sejam testados

Exemplos de Partições



Teste Estrutural

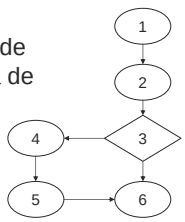
- Frequentemente chamado de teste caixa-branca
- A derivação de casos de teste ocorre de acordo com a estrutura do programa
  - O conhecimento do programa é usado para identificar casos de teste
- O objetivo é exercitar todas as declarações do programa

Teste de Caminho

- O objetivo é assegurar cada caminho pelo programa é executado pelo menos uma vez
  - Ponto de partida do teste de caminho é um fluxograma de programa

    1 – 2 – 3 – 4 – 5 – 6

    1 – 2 – 3 – 6



Resumo da Aula

- Verificação e validação não são a mesma coisa
  - Verificação mostra a conformidade com a especificação
  - Validação mostra que o programa atende às necessidades do cliente
- Em inspeções, o código de programa é sistematicamente verificado

Resumo da Aula

- Os testes podem mostrar a presença de defeitos em um sistema
  - Eles não podem provar que não há defeitos remanescentes
- Diferentes tipos de testes
  - Teste de Validação, Teste de Defeito, Teste de Componente, Teste de Sistema, Teste Estrutural, etc.

Bibliografia da Aula

- Ian Sommerville. **Engenharia de Software**, 9ª Edição. Pearson Education, 2011.
  - Cap. 8 Testes de Software

Próxima aula...

- Laboratório 2011 e 2012
- Tópico
  - Implementação
  - Possivelmente: teste de software (JUnit)