UFG
Universidade Federal de Goiás
Campus Catalão

Verificação e Validação (V&V)

Docente: Thiago Jabur Bittar

Disciplina: Engenharia de Software

Agenda

- Conceitos Iniciais
- Técnicas de inspeção
- Testes de software
- Análise estática automatizada
- Desenvolvimento de software Cleanroom
- Referências

Conceitos Iniciais

- Verificação e Validação (V&V)
  - Engloba processos de verificação e análise
  - Tenta assegurar que o software cumpra com suas especificações
    - atendendo às necessidades dos clientes

Definições básicas

- Verificação
  - Checar se o software cumpre com suas especificações, ou seja, se o sistema cumpre com seus requisitos funcionais e não funcionais
  - Estamos construindo certo o produto?

Usuário: isso não está funcionando!

- Validação
  - Assegurar que o software atenda às expectativas do cliente, garantindo que ele faça o que o cliente espera que faça
  - Estamos construindo o produto certo?

Usuário: não era isso o que eu queria!

Meta da V&V

- Verificação e Validação devem estabelecer a **confiança** de que o software é adequado ao seu propósito
- Isso não significa que o software tenha de ser inteiramente livre de defeitos
- Ele deve ser suficientemente bom para o grau de uso pretendido
  - O tipo de uso determinará o grau de confiança necessário

Confiança no software

- **Nível de confiança** depende de:
  - **Função do software**
    - O quão crítico o software é para a organização
    - Objetivo para o qual ele foi desenvolvido
  - **Expectativas do usuário**
    - Usuário vem se tornando mais exigente
    - Atualmente é menos aceitável entregar sistemas não confiáveis
    - O que implica em maior dedicação aos processos de V&V.

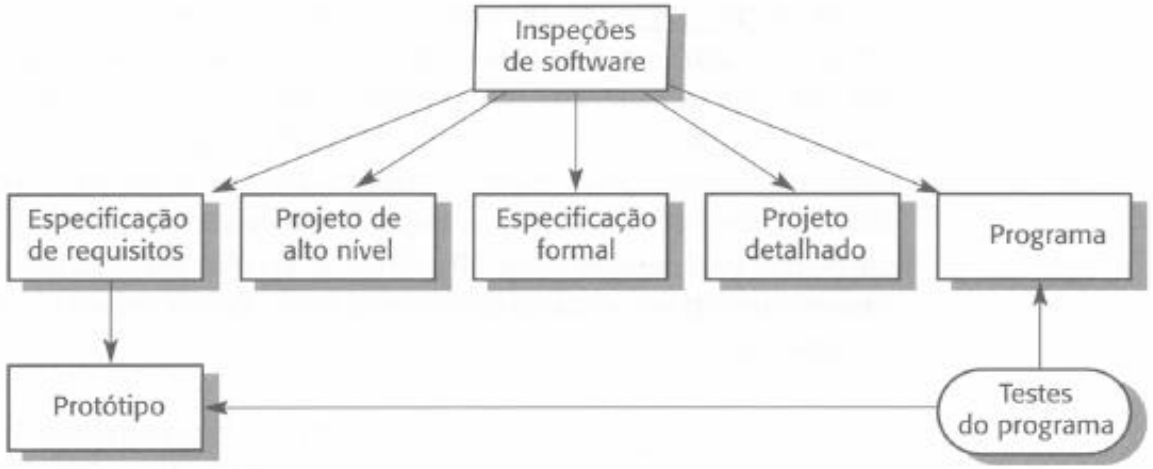
Confiança no software

- **Nível de confiança** também depende de:
  - **Ambiente de mercado**
    - O esforço a ser empregado no processo de V&V depende de vários fatores, tais como:
      - » Concorrência
      - » Preço que o cliente está disposto a pagar
      - » Cronograma, etc...

Técnicas de checagem e análise

- Inspeções de software:
  - Analisam e verificam as representações do sistema
    - documento de requisitos, diagramas de projeto, código-fonte do programa, ...
  - Podem ser aplicadas em todos os estágios do processo
  - São técnicas **estáticas**, pois não requerem que o sistema seja executado
- Testes de software:
  - Executam uma implementação do software com os dados de teste
  - Examinam as saídas e o comportamento operacional
  - Verificam se o sistema se comporta conforme o esperado
  - São técnicas **dinâmicas**

Técnicas de checagem e análise



Sommerville, 2006

- Inspeções de software podem ser empregadas em todos os estágios do processo.
- Os testes somente podem ser utilizados quando um protótipo ou um programa executável estiver disponível.

Técnicas de inspeção

- Incluem:
  - Inspeções de programa, análises automatizadas de código-fonte e verificação formal
  - Podem apenas verificar a correspondência entre um programa e sua especificação
- Não podem demonstrar que o software é operacionalmente útil
  - Não checa suas características não funcionais (desempenho, confiabilidade)
- Portanto: alguns testes são sempre necessários para validar um sistema de software.

Testes de software

- Ainda são mais utilizados
  - Utilizam dados processados pelo programa
  - Descoberta de defeitos ou inadequações
    - pelo exame das saídas geradas
- Podem ser realizados durante e após a fase de implementação
- Normalmente são utilizados mesmo após a aplicação das inspeções

Tipos de testes

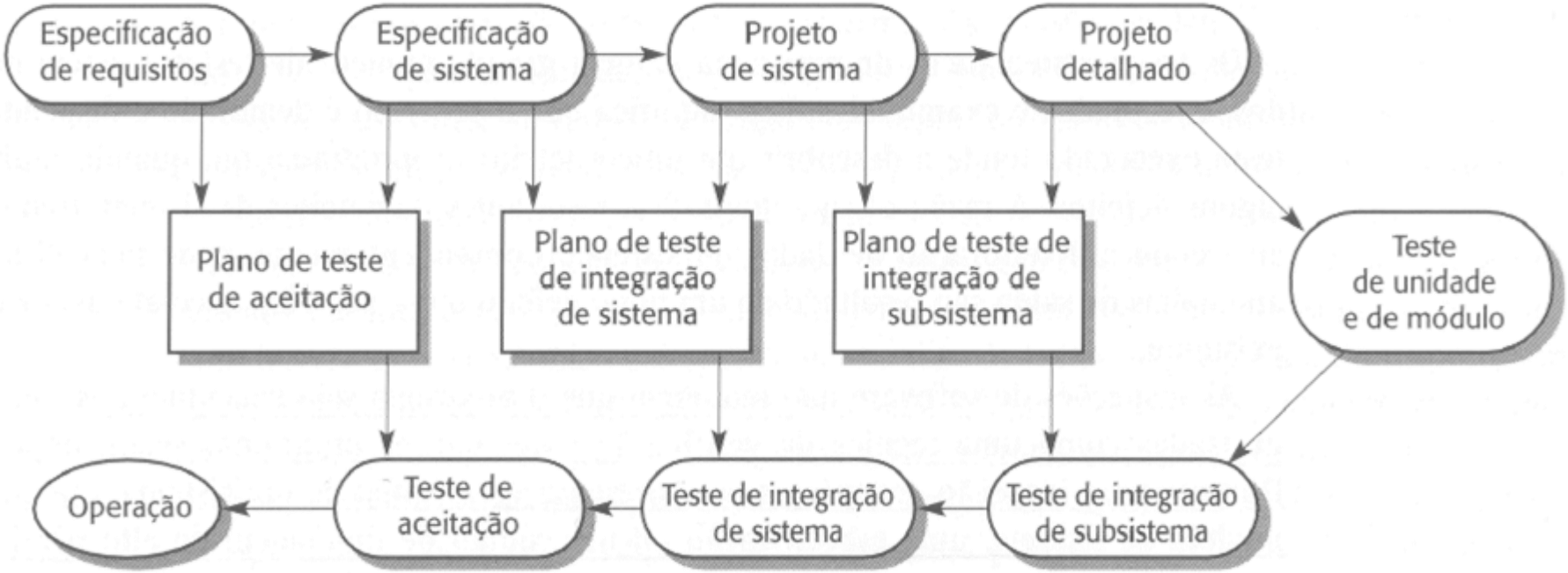
- Testes de defeitos:
  - O objetivo é revelar defeitos no sistema
  - Encontrando inconsistências entre o programa e sua especificação
- Testes estatísticos
  - Visam testar o desempenho e a confiabilidade do programa
  - Checa como o programa trabalha sob condições operacionais
  - Observa-se o número de falhas
  - Estimativa de confiabilidade
  - O desempenho do programa é medido pelo tempo de execução e seu tempo de resposta

Planejamento de verificação e validação

- V&V: processo dispendioso
  - Assim, deve-se haver um planejamento cuidadoso
    - Controlando os custos do processo
    - Obtendo o melhor das inspeções e testes.
- Deve iniciar no início do processo de desenvolvimento
- Estabelecimento de *checklists* para orientar as inspeções de programa
- Definição do plano de teste de software
  - Deve indicar o equilíbrio entre a verificação estática e testes
  - Define padrões e procedimentos para o processo

Modelo em V de desenvolvimento

- Mostra como os planos de testes devem ser derivados da especificação e projeto de sistema



Plano de testes

- Destinado aos engenheiros de software envolvidos em projetar e realizar os testes.
- Estabelecimento do cronograma e dos procedimentos de teste.
- Define os recursos de hardware e software necessários.
- Em processos ágeis (por exemplo, XP), o teste é inseparável do desenvolvimento.
- Planos de teste não são documentos estáveis
  - evoluem durante o processo de desenvolvimento

Estrutura de um plano de testes

1. O processo de teste
   - Descrição das principais fases
2. Facilidade de rastreamento dos requisitos
   - Definir como será projetado o software para ser fácil de rastrear os requisitos
   - Todos os requisitos devem ser individualmente testados
3. Itens a serem testados
   - Especificação dos produtos do processo de software
4. Cronograma de testes
   - Apresentar um cronograma geral de testes e alocação de recursos para ele

Estrutura de um plano de testes

5. Procedimento de registro de testes
   - Registro sistemático dos resultados dos testes.
   - O processo de teste deve ser auditado para checar se foi conduzido corretamente.
6. Requisitos de hardware e software
   - Ferramentas de software necessárias e a utilização estimada de hardware.
7. Restrições
   - Previsão de restrições que afetam os testes

Inspeções de software

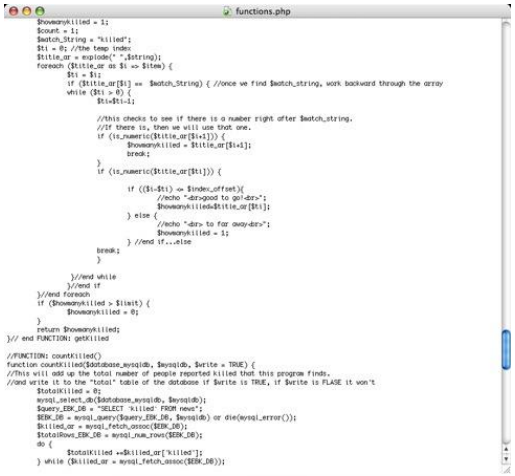
- Processo de V&V estático
  - O sistema é **revisto** para se encontrar erros, omissões e anomalias.
- **Objeto:**
  - Código-fonte, requisitos, modelo de projeto, dentre outros artefatos

Inspeções de software

- Inspeções podem ser usadas no processo de desenvolvimento
  - quando o sistema estiver integrado, aplica-se os testes para verificação de funcionalidade
- Difícil a introdução de inspeções formais dentro de muitas organizações de desenvolvimento de software
- Dificuldade para convencer engenheiros de software e gerentes

Inspeção em código fonte

- Antes de inspeções em código fonte deve-se:
  - Ter uma especificação precisa;
  - Haver a familiarização da equipe de inspeção com os padrões organizacionais;
  - Obter uma versão atualizada e completa do código disponível;



Vantagens da inspeção em relação aos testes

- Muitos defeitos diferentes podem ser detectados em uma única sessão.

- Inspeções podem acontecer em versões incompletas ainda não funcionais de um sistema
  - enquanto que os testes não

Vantagens da inspeção em relação aos testes

- Reutilizam o conhecimento de domínio e de linguagem de programação
  - Os inspetores sabem os erros mais comuns
- Além de procurar por defeitos de programas
  - É possível considerar os atributos de **qualidade**

Inspeção: atributos de qualidade

- Alguns atributos de qualidade que podem ser vistos na inspeção:
  - Conformidade com padrões
  - Portabilidade
  - Existência de comentários
  - Facilidade de manutenção
  - Ineficiências
  - Algoritmos inapropriados
  - Estilo de programação pobre
- Tais características dificultam a manutenção e a atualização de sistemas

Inspeções X Testes

- A comparação anterior não significa que as inspeções devam substituir inteiramente os testes de sistema
- Quase sempre é impraticável inspecionar o sistema inteiro
- As inspeções podem verificar a conformidade com uma especificação
  - mas não podem validar o comportamento dinâmico
- Revisões e testes não são técnicas concorrentes de V&V
  - Devem ser utilizadas em conjunto

– Corrigir um erro pode ocultar e produzir outros
  - após sua correção, caso novo erro aconteça, não se sabe se é um efeito sobre a correção, ou um erro novo

Análise estática automatizada

- Analisadores estáticos
  - Ferramentas que analisam o código-fonte de um programa e detectam possíveis defeitos e anomalias
- Percorrem o texto do programa
  - Reconhecem os diferentes tipos de declarações
  - Detecção de anomalias no programa
    - variáveis sem iniciação
    - varáveis não utilizadas
    - dados com valores excedidos, dentre outros
- Muito eficaz como um auxílio para inspeções
  - Um suplemento e não um substituto

Estágios da Análise estática automatizada

- Análise do fluxo de controle
  - Destaca loops com múltiplos pontos de saída ou de entrada e código inacessível
- Análise da utilização de dados
  - Detecta variáveis que são utilizadas sem prévia iniciação, variáveis declaradas mas nunca utilizadas, etc
- Análise de interface
  - Verifica a consistência das declarações de rotinas e procedimentos e seu uso
  - Funções e procedimentos que são **declarados** e **nunca chamados**
  - Resultados de funções que nunca são utilizados

Estágios da Análise estática automatizada

- Análise do fluxo de informações
  - Identifica as dependências
    - entre as variáveis de entrada e as de saída.
- Análise de caminho
  - Identifica todos os caminhos possíveis no programa
  - E exibe as declarações executadas nesse caminho

Uso da Análise Estática

- Particularmente valiosa no uso da linguagem C.
  - C é fracamente tipada
  - Muitos erros podem não ser detectados pelo compilador
- Baixo custo benefício para linguagens como Java que tem uma checagem forte de tipos
  - Muitos erros podem ser detectados durante a compilação

Desenvolvimento de software Cleanroom

- O nome é derivado do processo ‘Cleanroom’ na fabricação de semicondutores.
- A filosofia é evitar defeitos em vez de removê-los
  - Uso de um rigoroso processo de inspeção

Desenvolvimento de software Cleanroom

- Baseia-se em:
  - Especificação formal
  - Desenvolvimento incremental
  - Programação estruturada
    - É utilizado um número limitado de construções abstratas de controle e dados
    - Preservação de correção
  - Verificação estática
    - Utilizando rigorosas inspeções de software
  - Teste estatístico do sistema
    - Para determinar a sua confiabilidade

Características do processo Cleanroom

- Especificação formal utilizando um modelo de transição de estados
- Desenvolvimento incremental
- Programação estruturada – é utilizado um número limitado de construções abstratas de controle e dados
- Verificação estática utilizando rigorosas inspeções de software
- Teste estatístico do sistema

Equipes envolvidas no Cleanroom

- Equipe de especificação
  - Responsável por desenvolver e manter a especificação do sistema
- Equipe de desenvolvimento
  - Responsável por desenvolver e verificar o software. O software **não** é executado ou mesmo compilado durante este processo
- Equipe de certificação
  - Responsável por desenvolver um conjunto de testes estatísticos para testar o software, após o desenvolvimento. Os modelos de aumento da confiabilidade podem ser utilizados para decidir quando interromper os testes

Avaliação do processo Cleanroom

- Resultados na IBM tem sido muito animadores
  - com poucos defeitos descobertos nos sistemas entregues
- Uma avaliação independente mostrou que o processo não é muito mais dispendioso financeiramente que outras abordagens
- Menos erros que em um processo de desenvolvimento ‘tradicional’
- Não é claro se esta abordagem pode ser transferida
  - para um ambiente com engenheiros menos habilitados ou motivados

Referências

- SOMMERVILLE, I. **Engenharia de Software.** 8ª ed. São Paulo: Addison Wesley, 2006.
- PRESSMAN, R. S. **Engenharia de Software.** 5ª ed. McGraw Hill, 2002