Universidade Federal do Acre
Centro de Ciências Exatas e Tecnológicas
Curso de Bacharelado em Sistemas de Informação

Engenharia de Software II

Verificação e Validação

Prof. Dr. Daricélio Moreira Soares

Verificação e Validação (V&V)

É o nome dado aos processos de verificação e análise, a fim de assegurar que o software cumpra com suas especificações e atenda às necessidades dos clientes que estão pagando por ele (SOMMERVILLE, 2007).

- Incluem as tarefas:
  - ✓ revisões dos requisitos;
  - ✓ revisões de projeto;
  - ✓ inspeções de código;
  - ✓ testes de produto.

- A diferença entre verificação e validação pode ser assim definida:

  - ✓ **Verificação: “estamos construindo o produto corretamente?”.**
  - ✓**Validação: “estamos construindo o produto correto?”.**

- **Verificação**: checar se o software cumpre com suas especificações, ou seja, se o sistema cumpre com seus requisitos funcionais e não funcionais.
- **Validação**: assegurar que o software atenda às expectativas do cliente, garantindo que ele faça o que o cliente espera que faça.

- **V & V:** estabelecer a confiança de que o software está adequado ao seu propósito.

- **Nível de confiabilidade** 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 e, hoje, é menos aceitável entregar sistemas não confiáveis, o que implica em maior dedicação aos processos de V&V.

Abordagens para a verificação e análise de sistemas do processo de V&V

✓ *Inspeções de software* ou *revisões por pares*: 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, a fim de examinar as saídas e o comportamento operacional, verificando-se, assim, se o sistema se comporta conforme o esperado. São técnicas dinâmicas.

- As revisões de requisitos e revisões de projeto são as principais técnicas usadas para detecção de erros na especificação e no projeto.

- Técnicas de inspeção (estáticas) podem apenas verificar a correspondência entre um programa e sua especificação, ou seja, elas não podem demonstrar que o software é útil operacionalmente, nem verificar desempenho e confiabilidade.

Quanto às técnicas estáticas:

✓ Incluem: inspeções de programa, análises automatizadas de código-fonte, verificação formal.

✓ Podem apenas verificar a correspondência entre um programa e sua especificação.

✓ Não podem: demonstrar que o software é operacionalmente útil ou checar suas características não funcionais (desempenho, confiabilidade).

Quanto aos testes de software:

✓ São mais utilizados, pois 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 para validar um software, mesmo após a aplicação das inspeções.

Planejamento de verificação e validação

- V&V: processo dispendioso.
- É necessário planejamento cuidadoso para controlar os custos do processo de V&V e obter o máximo de inspeções e testes.
- Equilíbrio entre as abordagens estáticas e dinâmicas para a verificação e validação.
- Especificação de padrões e procedimentos para as inspeções e os testes de software.
- Estabelecimento de *checklists* para orientar as inspeções de programa.
- Definição do plano de teste de software.

• Tipo de sistema e a habilidade com inspeções de programas determinam o esforço dedicado às inspeções e aos testes.

• Quanto mais crítico o sistema, mais esforço deve ser dedicado às técnicas de verificação estática.

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.
- Planejamento do processo de testes.
- Em processos ágeis (por exemplo, XP), o teste é inseparável do desenvolvimento.
- Planos de testes não são documentos estáveis, mas evoluem durante o processo de desenvolvimento (atrasos nos estágios do processo).

Estrutura do plano de testes

1. **Processo de teste**: descrição das fases principais do processo de teste.
2. **Rastreabilidade de requisitos**: todos os requisitos devem ser individualmente testados.
3. **Itens testados**: especificação dos produtos do processo de software.
4. **Cronograma de testes**: apresentar um cronograma geral de testes e alocação de recursos para ele.
5. **Procedimentos de registro de testes**: trata-se do registro dos resultados dos testes. O processo de teste deve ser auditado para verificar que foi conduzido corretamente.

Estrutura do plano de testes

6. **Requisitos de hardware e de software**: ferramentas de software necessárias e a utilização estimada de hardware.

7. **Restrições**: o que afeta o processo de teste (por exemplo: falta de pessoal, falta de recursos).

Inspeções de software

- Processo de V&V estático, onde o sistema é revisto para se encontrar erros, omissões e anomalias.
- **Objeto:** código-fonte, requisitos, modelo de projeto.
- Principais vantagens da inspeção em relação aos testes:

1. Basta uma sessão de inspeção para se descobrir erros em um sistema. Os testes podem levar um erro a ocultar outro e, após sua correção, caso novo erro aconteça, não se sabe se é um efeito sobre a correção, ou um erro novo.
2. Inspeções podem acontecer em versões incompletas de um sistema, enquanto que os testes não (mais caros).

Inspeções de software

3. Além de procurar por defeitos de programas, é possível considerar os atributos de qualidade em um programa (conformidade com padrões, portabilidade, facilidade de manutenção), ineficiências, algoritmos inapropriados, estilo de programação pobre (tais características dificultam a manutenção e a atualização de sistemas).

- **Inspeções**: ideia antiga, mas comprovadamente mais eficiente para descobrir defeitos do que os testes (+ 60% dos erros em um programa podem ser detectados por inspeções de programa).
- Revisão estática de código é mais eficiente e menos dispendiosa do que teste de defeitos no descobrimento de defeitos de programa.

Inspeções de software

- **Revisões**: uma boa aplicação é na revisão dos casos de testes para um sistema.
- Revisões e testes devem ser usadas em conjunto no processo de verificação e validação.
- Inspeções podem ser usadas no processo de desenvolvimento e, quando o sistema estiver integrado, aplica-se os testes para verificação da sua funcionalidade.
- Difícil a introdução de inspeções formais dentro de muitas organizações de desenvolvimento de software.
- Dificuldade para convencer engenheiros de software e gerentes.

Inspeções de software

- As inspeções de programas são revisões com o objetivo de detectar defeitos.
- Idéia nascida na IBM (década de 70).
- **Inspeção**: método de verificação de programa amplamente usado, especialmente na engenharia de sistemas críticos.
- Diversas abordagens de inspeção, porém, todas incluem equipes com diferentes experiências para reverem, linha por linha, o código-fonte do programa.

- Processo formal, realizado por uma equipe de, pelo menos, quatro pessoas (podendo variar de uma inspeção para outra).

- Base: uma equipe com membros que apresentam diferentes experiências deve fazer uma revisão cuidadosa do código-fonte do programa.

- Antes de uma inspeções de programa é necessário:
  - ✓ ter uma especificação precisa do código a ser inspecionado;
  - ✓ familiarização da equipe de inspeção com os padrões organizacionais;
  - ✓ versão atualizada do código disponível; o código deve estar completo.
- A inspeção deve ser um processo relativamente curto (não mais de duas horas).
- O responsável pelo planejamento da inspeção (moderador) seleciona a equipe de inspeção e prepara o ambiente (local e o material da inspeção).
- Deve identificar defeitos, anomalias e não-conformidades com padrões.

- A equipe de inspeção não sugere como os defeitos devem ser corrigidos, assim como não recomenda modificações em outros componentes.
- O programa, após a inspeção, deve ser modificado pelo seu autor, para correção dos problemas identificados.
- O processo de inspeção deve ser dirigido por uma *checklist* de erros comuns dos programadores (específicas para cada linguagem de programação).
- Uma análise dos defeitos encontrados pode ser feita, após a inspeção.
- A quantidade de software a ser inspecionado em determinado tempo depende da experiência da equipe de inspeção, da linguagem de programação e do domínio da aplicação.

Análise estática automatizada

- Analisadores estáticos de programa são ferramentas de software que analisam o código-fonte de um programa e detectam possíveis defeitos e anomalias.
- Não requerem que o programa seja executado.
- Percorrem o texto do programa e reconhecem os diferentes tipos de declarações.
- Detecção de anomalias no programa (variáveis sem iniciação, varáveis não utilizadas, dados com valores excedidos, etc.), podendo resultar em erros de programação e de omissões).
- Análise estática automatizada é mais bem utilizada com as inspeções de software.

- Os estágios envolvidos incluem:

  ✓ 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 ou resultados de funções que nunca são utilizados.

  ✓ 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.

Referências

SOMMERVILLE, I. **Engenharia de Software.** 6ª ed. São Paulo: Addison Wesley, 2003. 592p.

SOMMERVILLE, I. **Engenharia de Software. 8ª** ed. São Paulo: Addison Wesley, 2007. 549p.