SWEBOK
Construção & Teste de Software

Profº. Msc. Rodrigo Santos

Agenda

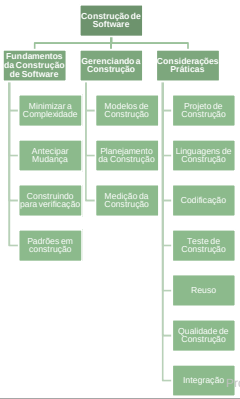
- Construção de Software
  - Objetivo da disciplina
  - Organização no guia
  - Destaques
  - Considerações práticas
  - Outras referências
- Teste de Software
  - Objetivo da disciplina
  - Organização no guia
  - Destaques
  - Considerações práticas
  - Outras referências

Construção de Software - Objetivo

- Têm por objetivo tratar a criação detalhada de um produto significativo de software através da combinação da codificação, verificação, teste unitário, teste de integração e debugging.
- É uma das disciplinas de maior interação com outras:
  - Projeto (Design)
  - Teste
  - Ger. de Configuração
  - Qualidade, etc.

Construção de Software - Organização

Tópicos da área de conhecimento Construção de Software (SWEBOK 2004)

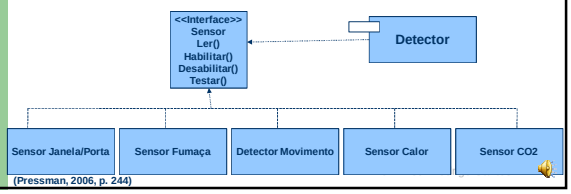


Construção de Software - Destaques

- Limiar entre Projeto (Design) e Construção e Construção e teste:
  - Depende da metodologia e processos adotados na empresa. Ex: Prototipação, Teste Unitário
  - Outro exemplo é que parte dos fundamentos poderiam ser encarados como definições de Projeto (Design)

Construção de Software - Destaques

- Componentização
  - Fomenta o reuso;
  - Padronização;
  - Simplificação;



(Pressman, 2006, p. 244)

Construção de Software - Destaques

- Componentização
  - Preocupações e princípios;
    - Compatibilidade
      - Alterações no componente master devem atender a todos os seus "clientes"
    - Coesão no empacotamento
      - Tratamento da mesma área funcional
      - Ajudar a minimizar os testes

Construção de Software – Considerações Práticas

- Teste e Qualidade de Construção
  - Inspeção de códigos
    - Verificação do código em busca de problemas ou não conformidades com padrões.
    - Exemplo:
      - Funções e variáveis não utilizadas
      - Códigos duplicados
      - Trecho de códigos vazios (if sem ações, tratamento de erros sem ações, etc...)
      - Documentação e formatação do código;

Construção de Software – Considerações Práticas

- Teste e Qualidade de Construção
  - Inspeção de códigos
    - Podem ser facilmente automatizados
      - Redução no tempo de inspeção
      - Identificação de erros mais comuns e vícios de programação
      - Reduz (ou elimina) a necessidade de um revisor especializado (Eficiência do teste)
      - Identificação de problemas nas fases iniciais do desenvolvimento
      - Menor custo de correção
      - Possibilidade de validação automática dos padrões do projeto

Construção de Software – Considerações Práticas

- Teste e Qualidade de Construção
  - Testes Unitários
    - Manter o princípio: “Quem desenvolveu um código não o testa!”.
    - Formalmente deve ser executado após a inspeção/revisão do código
    - Também podem ser automatizados
      - Já apresenta um custo considerável;
      - Priorizar os testes com maior chance de repetição (Principais módulos e componentes)

Construção de Software – Considerações Práticas

- Teste e Qualidade de Construção
  - Builds automáticos
    - Automatiza uma atividade que irá se repetir durante os vários ciclos ou execuções dos testes e releases do sistema
    - Builds periódicos com o objetivo de encontrar erros de compilação ou integração dos módulos
    - Faz a primeira parte do teste de integração: Verificam se os componentes compilam juntos

Construção de Software – Outras Referências

- http://www.componentsource.com/index.html
- http://sourceforge.net/
- Ferramentas:
  - http://ant.apache.org/
  - Maven (http://maven.apache.org/)
  - PMD - http://pmd.sourceforge.net/
  - Code Inspector - http://www.safedevelop.com/Products/CodeInspector/
  - JUnit - http://junit.sourceforge.net/
  - NUnit - http://www.nunit.org/

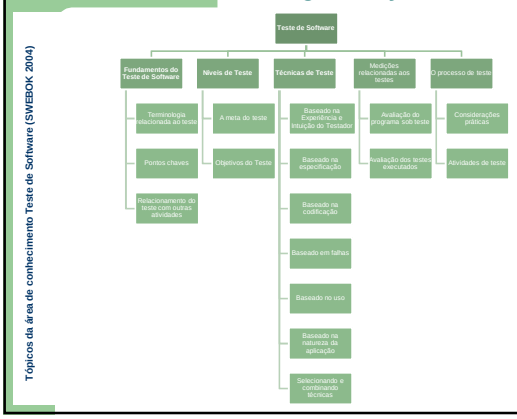
Construção de Software – Outras referências

- Software Components: Guidelines and Applications, Muthu Ramachandran, 2009.
- Component Software: Beyond Object-Oriented Programming, 2nd Edition, Clemens Szyperski, 2002.

Teste de Software - Objetivo

- A verificação dinâmica do comportamento de um programa a partir de um conjunto finito de casos de teste, selecionados do domínio infinito de execuções, em relação ao comportamento esperado
  - Dinâmica: Porque prevê a execução do programa ou componente.

Teste de Software - Organização



Testes de software - Destaques

- Terminologia
  - Defeito (fault): passo, processo ou definição de dados incorreto,como por exemplo, uma instrução ou comando incorreto;
  - Erro (error): Diferença entre o valor obtido e o valor esperado, ou seja, qualquer estado intermediário incorreto ou resultado inesperado na execução do programa constitui um erro;
  - Falha (failure): Produção de uma saída incorreta com relação à especificação.

(Maldonado et al, 200?, p. 5)

Teste de software - Destaques

- Um teste só pode afirmar a existência de defeitos em um programa ou componente, nunca a ausência deles (Dijkstra).
- Meta dos testes
  - Teste Unitário
  - Teste de Integração
  - Teste de Sistema
    - Geralmente verifica os requisitos não funcionais como segurança, desempenho, confiabilidade, e questões de compatibilidade (Hardware, por exemplo)

Teste de software - Destaques

- Objetivos dos testes
  - Teste de aceitação e qualificação
  - Teste de instalação
  - Teste de conformidade
  - Teste de desempenho
  - Teste de stress
  - Teste de usabilidade

Testes de Software – Considerações Práticas

- Automação
  - Em um projeto de software de médio/grande porte realizar todos os testes manuais pode ser inviável ou impossível
    - Cobertura insuficiente
    - Custos de pessoal
    - Simular os volumes de produção (Teste de Carga)

Testes de Software- Considerações Práticas

- Automação
  - Benefícios
    - Pode simplificar e agilizar muito os testes de regressão
    - Pode maximizar a precisão dos testes e sua capacidade de repetição
    - Pode trazer redução de custos dos testes

Testes de Software- Considerações Práticas

- Automação
  - O quê não é verdade
    - Todos os testes podem/devem ser automatizados
    - A automatização sempre vai trazer redução de custos
    - 100% dos requisitos podem ser cobertos pelos testes automatizados
    - O trabalho de automatização só pode iniciar após o fim da codificação do produto a ser testado
    - Um único grupo de ferramentas de automação de teste pode atender todos os ambientes e necessidades

Testes de Software- Considerações Práticas

- Gerência de configuração
  - Os testes de software são muito dependentes da gerência de configuração
  - Códigos fontes a serem testados precisam ser confiáveis
  - Os scripts/códigos dos testes devem ser versionados e controlados
  - Em vários projetos os itens de teste fazem parte dos "entregáveis"

Testes de software – Considerações práticas

- Outros pontos importantes:
  - Exibição e rastreamento dos defeitos (bugtracking)
  - Documentação e scripts de testes (podem fazer parte dos entregáveis)
  - Inicio do planejamento/desenvolvimento dos testes o quanto antes ou em paralelo com o desenvolvimento

Teste de Software – Outras Referências

- http://opensourcetesting.org/ - Site com centenas de ferramentas para auxiliar o teste de software
- Dustin, Elfriede. Effective Software Testing: 50 Specific Ways to Improve Your Testing. Addison Wesley, 2002.
- Beck, Kent. Test-Driven Development By Example. Addison Wesley, 2002.
- Hutcheson, Marnie L. Software Testing Fundamentals: Methods and Metrics. John Wiley & Sons, 2003.
- Beizer, Boris. Software Testing Techniques, Second Edition . The Coriolis Group, 1990.
- McGregor, John D; Sykes, David A. A practical guide to testing object-oriented software. Addison-Wesley, 2001.

Referências

- Guide to the Software Engineering Body of Knowledge, 2004. Disponível em: www.swebok.org.
- Engenharia de Software, 2006, Roger S. Pressman.
- Engenharia de Software: Teoria e Prática, 2001, James F. Peters, Witold Pedrycz
- Dustin, Elfriede. Effective Software Testing: 50 Specific Ways to Improve Your Testing. Addison Wesley, 2002.
- José Carlos Maldonado et al, Introdução ao Teste de Software, 200? (http://www.inf.ucv.cl/frames/interface_fr_files/menu_cap/doc_noticias/minicurso.pdf)