SÃOJUDAS UNIVERSIDADE

Curso - Especialização em Engenharia de Software

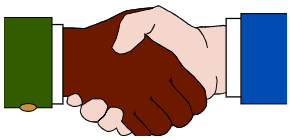
Disciplina: Aspectos Avançados do Teste de Software: Revisões & Inspeções

Prof. Edson Saraiva

Tipos de Revisão de Software

Apresentação

- Nome
- Atividade profissional
- Experiência prévia
- Motivação para especialização
- Expectativa com a disciplina (profissional)
- Experiência com revisões de software?
- Quais livros, jornais, revistas ou artigos você leu recentemente relacionado a práticas de desenvolvimento de software?
- http://www.noop.nl/2008/06/top-100-best-software-engineering-books-ever.html



2

Agenda

- Engenharia de Software – crise do software
- Conceitos de Qualidade de Software
- Custo da Qualidade
- Tipos de Revisão
- Inspeção

3

Objetivos Específicos

- Analisar como a disseminação dos conceitos de qualidade esta impulsionando a melhoria de processos e consequentemente mudanças organizacionais
- Refletir sobre as dificuldades de implantar mudanças organizacionais que não estejam baseadas em evidências - Engenharia de Software Baseada em Evidências
- Compreender a importância do planejamento da qualidade - o contexto da revisão de software como atividade de garantia da qualidade.

4

Bibliografia

- PFLEEGER, S.L. Engenharia de Software, 2ed, Pearson Prentice Hall, 2004
- SOMMERVILLE, I., Engenharia de Software, 8ed, Addison-Wesley, 2007
- PRESSMAN, R.S., Engenharia de Software, 6ed. McGrawHill, 2006
- Wiegers, K.E., Peer Reviews in Software – A Practical Guide, Addison-Wesley Information Technology Series, 2002

5

Bibliografia

- Freedman et.al., Manual de Walkthroughs: Inspeções e Revisões Técnicas de Especificações de Sistemas e Programas, Makron Books, 1993.
- IEEE Std 1028-IEEE Standard for Software Reviews 1997
- Wong, Y.K., Modern Software Review: Techniques and Technologies, IRM Press, 2006
- Material da disciplina
  - https://sites.google.com/site/revisoeseinspecoes/
  - Permite acesso por dispositivos móveis (smartphones)

6

Avaliação – análise crítica de artigos

- Discussão e análise crítica de artigos 30 %
  - Desenvolver uma análise crítica do artigo
  - Elaborar duas questões
  - Entregar impresso de acordo com o calendário proposto.
- Objetivo – atividade prática em grupo: com base em um conjunto de perguntas relacionadas ao tema de revisões de software, os participantes irão discutir propostas de solução e novos questionamentos através de um ambiente de debate e aprendizado colaborativo.
- Não serão aceitas entregas fora do calendário, ou manuscritos, devem ser entregues no inicio da aula impressos (0,25 a 0,5) individual.

7

Avaliação – análise crítica de artigos

Artigo: "Retorno de Investimento de Melhoria de Processo na BL Informática"

Nome do aluno:

Análise crítica

1) O artigo descreveu um problema relevante no contexto da engenharia de software e foi dirigido por uma pergunta claramente formulada? Qual?
2) É possível reproduzir a análise realizada no texto?
3) Que conclusões foram obtidas, elas são justificadas pelos resultados?

Questões para debate e aprendizado colaborativo (uma pessoa que leu o artigo deve estar apta a responder estas perguntas)

1)

2)

8

Avaliação - RT

- Relatório técnico abordando aplicações profissionais de revisões de software:
  - Entregas parciais no calendário solicitado
  - Entrega do relatório final impresso (na aula)
  - Apresentação no último dia de aula (faltas devem ser justificadas)
- Avaliação do relatório – número de citações (3 por integrante do grupo), quantidade de páginas (no mínimo 10), argumentos para defesa da proposta, estudo de caso consistente, aplicação profissional.

9

Avaliação - RT

- Relatório Técnico:
  - identificar uma dificuldade técnica profissional, ou área de interesse de pesquisa, relacionada a revisões de software
  - iniciar pesquisas para selecionar o referencial teórico (individual)
  - formação de grupos (4 no máximo)
  - cada participante deve colaborar com pelo menos 3 citações (12 citações)

10

Avaliação - RT

1. Introdução – por que o trabalho foi feito? motivação
2. Materiais e métodos – como foi feito?
   - Referencial teórico
3. Resultados – o que foi observado?
4. Considerações finais - o que se pode concluir dos resultados?

11

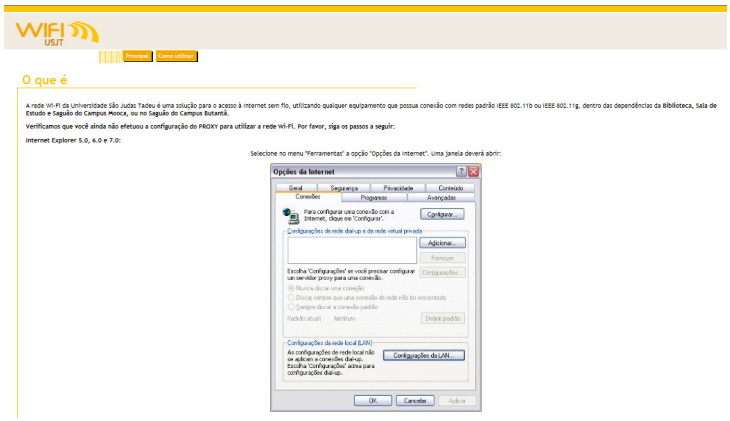
Referencial Teórico

- Principais contribuições ao conhecimento comprovado: pesquisar técnicas de revisões de software mais comumente utilizadas e avaliar o seu uso em modelos de construção para desenvolvimento de software.
- Dinâmica – pesquisas bibliográficas

12

Referencial Teórico - Acesso WIFI

- Configuração



13

Referencial Teórico - Acesso WIFI

- Configuração

14

Referencial Teórico - Base de Dados

- Acesso pelos computadores ligados à rede da Universidade e a rede Wireless USJT

15

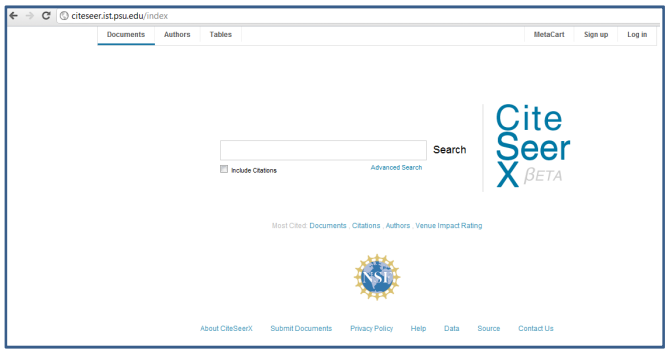
Referencial Teórico

- Bases de pesquisa - http://scholar.google.com.br/

16

Referencial Teórico

- Bases de pesquisa - http://citeseer.ist.psu.edu/index



17

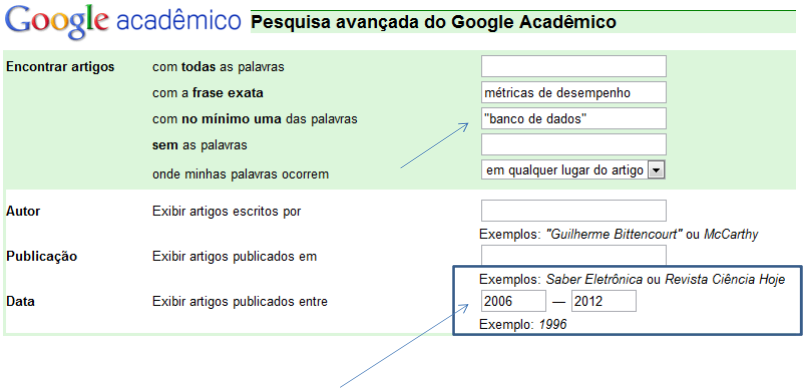
Referencial Teórico

- Resultado da pesquisa

18

Referencial Teórico

- Pesquisa avançada – os critérios de pesquisa podem excluir informações relevantes.



19

Referencial Teórico

Objetivo: Selecionar artigos que contenham informações de aplicações práticas de teste de unidade.

Critérios de inclusão:

- Disponibilidade na web (bibliotecas digitais ou banco de dados técnicos) e na Biblioteca Profa. Alzira Altenfelder Silva Mesquita
  - ✓ http://www.usjt.br/biblioteca/base_dados/index.php : ScienceDirect
  - ✓ http://scholar.google.com.br/ : Google Acadêmico
  - ✓ http://citeseer.ist.psu.edu/
  - ✓ http://biblioteca.usjt.br/pergamum/biblioteca/pesquisa_avancada.php : biblioteca Profa. Alzira Altenfelder Silva Mesquita

Critérios de exclusão:

- Uma classificação estabelece o grau no qual as referencias oferecem informações substanciais relacionadas a aplicação pratica de testes de unidade:
  0 superficial, 1 pouco detalhado, 2 detalhado
- Os artigos classificados com 0 ou 1 serão descartados
- Publicados entre 2005 e 2011
- Classificados pelo número de citações

Estratégia de pesquisa:

Google acadêmico:
"teste de unidade" e ano de publicação entre 2006 e 2011

Biblioteca Profa. Alzira Altenfelder Silva Mesquita
"engenharia de software" entre 2006 e 2011

ScienceDirect
"unit test" e ano de publicação entre 2006 e 2011

Resultados:
Para o Google acadêmico foram identificados 21 itens. Os seguintes artigos foram classificados como 2.

Para biblioteca Profa. Alzira Altenfelder Silva Mesquita:

1) Engenharia de software - 6. ed. / 2006 - (Livros) - Acervo 61202 PRESSMAN, Roger S. Engenharia de software. 6. ed. Rio de Janeiro, RJ: McGraw-Hill, 2006.
2) Engenharia de software - 8. ed. / 2007 - (Livros) - Acervo 71573 SOMMERVILLE, Ian. Engenharia de software. 8. ed. São Paulo, SP: Addison Wesley, 2007. ,

Elaborar o resumo:
1-introdução (motivação) porque testes de unidade são importantes?
2-materiais e métodos (como o trabalho foi feito) referencial teórico
3 – estudo de caso (foram derivados x casos de teste, executados e x minutos)
4 – Conclusões (sua avaliação do trabalho realizado)

20

Atividades

- Leitura do artigo “Inspeções de Requisitos de Software” (entrega dia 27/10/2012 individual impresso)
- Identificar uma dificuldade técnica profissional, ou área de interesse de pesquisa, relacionada a revisões de software.
- Estabelecer o plano de pesquisa (critérios) do referencial teórico para o RT. (entrega para o dia 27/10 individual impresso)

21

Atividades

- Selecionar um documento do processo de desenvolvimento de software da sua organização para realizar uma revisão técnica discuta a proposta com o professor (entrega no dia 27/10 individual impresso).

22

Engenharia de Software

Engenharia de Software

- Conjunto de princípios, métodos e técnicas que tratam o software como produto de engenharia que requer planejamento, projeto, implementação e manutenção

Objetivo principal

- Produzir software com alta qualidade e baixo custo.

23

Engenharia de Software

Engenharia de Software

- 50 anos de progresso
- a abordagem para o seu desenvolvimento ainda tem se fundamentado em métodos e práticas "artesanais".

Desenvolvimento de Software

- Deve ser encarado como um esforço de engenharia, mas isto é feito com raríssimas exceções e principalmente por organizações cuja finalidade ou negócio é o software.

Crise do Software

- baixa qualidade
- prazos e orçamentos não são atendidos
- métodos gerenciais igualmente empíricos.

24

Engenharia de Software

Importância econômica	• As condições para tratar seu desenvolvimento com práticas mais adequadas começam a ser estabelecidos.
O que está mudando	• Disseminação dos conceitos de qualidade, modelos de melhoria de processos CMMi, MPS.BR

25

Engenharia de Software - Falhas

Falhas em projetos de software	• Mesmo com imperfeições introduzidas e entregues aos usuários temos tido sucesso
Dinâmica do processo de desenvolvimento	• "Apagar incêndios" a um altíssimo custo (infelizmente não mensurado) considerando todo o ciclo de vida do software • Com um "tempo de entrega" inaceitável dada a dinâmica dos negócios e do mercado.
Controle de qualidade	• Para começarmos a tratar o software sob uma abordagem de engenharia é necessário controlar a qualidade do software e compreender a importância da medição para sua gestão.

26

Importância da Medição

Medição

- "Não se pode controlar o que não se pode medir" (Tom de Marco)
- Um elemento chave de qualquer processo de engenharia é a medição.

Atributos de qualidade

- Medidas são utilizadas para entender melhor os atributos de qualidade dos produtos ou sistemas submetidos ao processo de engenharia.

27

Mudanças Organizacionais

Uma boa gestão supõe

- A possibilidade de predizer o comportamento futuro dos produtos e processos de software,
- É necessário contar com dados apropriados e confiáveis (isto significa implantar um programa de métricas).

Mudanças organizacionais

- Não convence
- Como disparar mudanças organizacionais?

28

Mudanças Organizacionais

- Aguardar uma modificação estrutural da empresa

29

Mudanças Organizacionais

Você não convence

As mudanças iniciam com você

A única pessoa que você pode controlar é você mesmo, se você não tem esperança que você mude como pode esperar que alguém mude?

30

Mudanças Organizacionais

Localização atual?

Próximo passo na direção do destino?

Qual é o destino?
Porque ?

Métrica de progresso – como saber que está se movendo na direção certa?

31

Causas das resistências

Localização atual?
Não sei a situação atual

Próximo passo na direção do destino?
Não vê qualquer caminho

Qual é o destino?
Não quer ir .
Porque ?
Não entende o destino

Métrica de progresso – como saber que está se movendo na direção certa?
Não percebe qualquer progresso

32

Melhoria de Processo

Mudanças na abordagem	• Frequentemente profissionais tomam decisões sobre qual tecnologia adotar em um projeto.
Origem	• Problemas enfrentados com as praticas atuais, gargalos de produção, defeitos
Adoção de uma nova tecnologia	• Quais são os benefícios? • Com poucas evidencias objetivas para confirmar a adequação, limitações, custos e riscos

33

Melhoria de Processo

Consequências	• Outras tecnologias são ignoradas apesar de existirem evidencias de que elas provavelmente seriam mais uteis.
Programação OO	• Promovia o valor da hierarquia nos modelos • Mais tarde evidencias experimentais relevaram que hierarquias profundas eram mais sujeitas a erros

34

Melhoria de Processo

Consultoria	• Mentoria para que os colaboradores desenvolvam mais conhecimentos sobre novos processos ou novas e melhores técnicas de trabalho.
Frustração	• Muitas das iniciativas são frustradas pois os colaboradores não tem motivação para manter tais melhorias.
Resistência	• A "não ação" está relacionada ao modelo mental que lhe faz perceber determinado assunto. • Não significa má fé os colaboradores não acreditam ou não enxergam importância naquela iniciativa.

35

Engenharia de Software Experimental

Engenharia de software	• Necessita do ciclo de construção de modelos, experimentação e aprendizado (cognitivo)
Ciência de laboratório	• O papel do pesquisador é compreender a natureza dos processos produtos e o relacionamento entre os dois • O papel do profissional (engenheiro de software) é construir sistemas cada vez melhores utilizando o conhecimento disponível
Simbiose	• Associação recíproca entre os dois papeis

36

Engenharia de Software Baseada em Evidencias

ESBE

- Tem como objetivo melhorar as decisões relacionadas ao processo de desenvolvimento e manutenção de software por integração das evidencias de pesquisas com a experiência prática e valores humanos.

Resultados

- Não é esperado que uma tecnologia seja universalmente boa, ou ruim somente mais apropriada em algumas circunstancias e para algumas organizações.

37

Engenharia de Software Baseada em Evidencias

Objetivo

- Diminuir o "buraco" entre a pesquisa e a prática encorajando uma forte ênfase no rigor metodológico enquanto mantém o foco na relevância prática.

Rigor metodológico

- Apesar do rigor ser necessário para pesquisas relevantes na ES não é suficiente.

Empírico

- Devem ser integrados com evidencias de estudos de observação de uso, estudo de caso, pesquisas e experimentos de campo.

38

Engenharia de Software Baseada em Evidencias

1. Converter um problema ou informação relevante em uma pergunta
2. Pesquisar na literatura as melhores evidencias disponíveis para responder a questão.
3. Avaliar criticamente as evidências de sua validade, impacto e aplicabilidade.
4. Integrar as evidencias avaliadas com experiência prática os valores do cliente e as circunstâncias para tomar decisões
5. Avaliar o desempenho e buscar formas de melhoria

39

ESBE

- Progresso – como avaliar a efetividade e eficiência do que já estamos fazendo
- Pode-se classificar as técnicas de avaliação em quatro categorias (PFLEEGER, 2004):
  - Análise de características
  - Pesquisa de opinião
  - Estudo de caso
  - Experimento formal

40

Análise de Características

- Utilizada para atribuir um valor e classificar os atributos de vários produtos, de modo que se possa decidir qual ferramenta comprar ou que método utilizar.
- Exemplo: aquisição de uma ferramenta de projeto
  - ✓ Atributos : boa interface com o usuário, tratamento de projeto OO, verificação de consistência, suporte para UC, suporte para ambiente Unix

41

Análise de Características

- Três ferramentas são identificadas, valores são atribuídos aos critérios (1-não satisfaz, 5-satisfaz completamente)

Atributo	Ferramenta 1: t-OO-1	Ferramenta 2: ObjectTool	Ferramenta 3: EasyDesign	Importância
Boa interface com o usuário	4	5	4	3
Projeto orientado a objetos	5	5	5	5
Verificação da consistência	5	3	1	3
Casos de uso	4	4	4	2
Execução em UNIX	5	4	5	5
Pontuação	85	77	73	

42

Análise de Características

- Pode ser subjetiva na medida que as avaliações refletem a tendência do avaliador.
- É útil para estreitar o leque de opções de ferramentas que podem ser adquiridas.

43

Pesquisa de Opinião

- Pesquisas de opinião geralmente são feitas para representar as opiniões de uma população fazendo-se uma série de perguntas a um pequeno número de pessoas e então extrapolando as respostas para um grupo maior dentro do intervalo de confiança.

44

Estudo de Caso

- Concepção, definição das hipóteses, projeto, preparação, execução , análise, tomada de decisão
- Geralmente compara os resultados de utilização de uma ferramenta ou um método como os resultados da utilização de outra ferramenta ou método.

45

Estudo de Caso - hipóteses

- Atividades de revisão e validação dos casos de uso reduz o retrabalho na fase de codificação.
- A utilização de um padrão de codificação reduz o número de defeitos por linha de código e reduz os custos de manutenção

46

Experimentos Formais

- Tipo de estudo mais controlado
- Em um experimento formal, diversos métodos são utilizados para reduzir tendências e eliminar fatores que se confundem, de modo que causa e efeito possam ser avaliados com alguma confiança.
- Variáveis independentes são manipuladas e observamos as mudanças nas variáveis dependentes.

47

Qualidade de Software

Planejamento da Qualidade	• Qualidade é importante - não é suficiente é necessário definir explicitamente o que é qualidade de software
Selecionar as atividades	• Conjunto de atividades que permitem garantir que todos os produtos de trabalho exibem alta qualidade
Realizar as atividades	• Realizar as atividades de controle e garantia da qualidade de software em todo o projeto
Medição	• Com objetivos de desenvolver estratégias para aperfeiçoar o processo de desenvolvimento de software

48

Qualidade de Software

Qualidade	• Esta relacionada a uma característica ou atributo de um produto ou serviço
Conjunto de atributos	• Referem-se a características mensuráveis que podemos comparar com padrões conhecidos tais como comprimento, cor, propriedades elétricas etc.
Software	• Estas características incluem a complexidade ciclomatica, a coesão, o número de pontos de função, de linhas de código, etc.

49

Qualidade de Software

Exame baseado em características	• Duas espécies de qualidade podem ser encontradas: qualidade de projeto e qualidade de conformidade
Qualidade de projeto	• Características que os projetistas especificam para um certo item.
Qualidade de conformidade	• Grau em que as especificações de projetos são seguidas durante a fabricação

50

Qualidade de Software

Controle da Qualidade

- Controle da variação envolve uma série de inspeções, revisões e testes usados ao longo do processo de desenvolvimento (PRESSMAN,2006).

Garantia da Qualidade

- Consiste em um conjunto de funções que avalia a efetividade e completeza das atividades de controle da qualidade (PRESSMAN, 2006).

51

Qualidade de Software

Garantia da Qualidade

- É o processo de definição de como a qualidade de software pode ser atingida e como a organização sabe que o software possui o nível de qualidade necessário (SOMMERVILLE, 2007).

Processo de QA

- Esta principalmente relacionado à definição e seleção de padrões que devem ser aplicados ao processo de desenvolvimento de software ou ao produto (SOMMERVILLE, 2007)

2 tipos de padrões

- Pode-se selecionar e adquirir ferramentas e métodos para apoiar os padrões aplicados : padrões de produto e de processo (SOMMERVILLE, 2007).

52

Garantia da Qualidade e Padrões

Padrões de Produto	Padrões de Processo
Formulário de revisão do projeto	Conduta de revisão do projeto
Estrutura de documento de requisitos	Envio do documento para o controle de mudanças
Formato de cabeçalho do método	Processo de liberação da versão
Estilo de programação em Java	Processo de aprovação do plano de projeto
Formato do plano de projeto	Processo de controle de mudanças
Formulário de solicitação de mudança	Processo de registro de teste

(SOMMERVILLE, 2007)

53

Garantia da Qualidade e Padrões

Desenvolvimento de padrões

- Processo difícil e demorado, devem ser basear em padrões organizacionais e padrões nacionais e internacionais.

Burocráticos

- Equipes de desenvolvimento de software consideram padrões burocráticos e irrelevantes.

Necessidade

- Embora concordem geralmente sobre a necessidade frequentemente encontram boas razões pelas quais os produtos não são necessariamente apropriados para o seu projeto especificamente.

54

Garantia da Qualidade e Padrões

Envolvimento	• A equipe quando envolvida provavelmente fica mais comprometida com esses padrões.
Revisar e modificar regularmente	• Para refletir as mudanças de circunstancias e de tecnologia.
Automação	• Prover ferramentas de software para apoiar os padrões, sempre que possível.

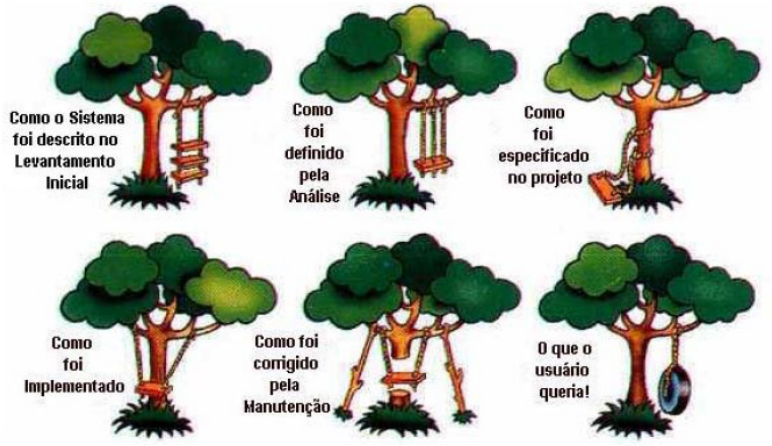
55

Qualidade de Software

Custos de falhas internas	• Ocorrem quando detectamos um defeito antes da entrega.
Custos de falhas externas	• São associados aos defeitos que são encontrados depois que o produto foi enviado ao cliente.
Custo de falhas	• Aumentam significativamente à medida que migramos dos custos de prevenção para detecção de falhas internas/externas.

56

Defeitos no PDS



57

Defeitos no PDS

Origem humana
- A maior parte dos defeitos é de origem humana, são gerados na comunicação e na transformação de informações

Latentes
- Permanecem presentes nos diversos produtos de software produzidos e liberados

Partes não utilizadas
- A maioria encontra-se em partes do produto de software raramente utilizadas e/ou executadas

58

Categorização dos Defeitos

Omissão	•Informação necessária sobre o sistema foi omitida do artefato de software
Fato incorreto	•Alguma informação no artefato de software contradiz informação do documento de requisitos ou o conhecimento geral do domínio
Inconsistência	•Informação contida em uma parte do artefato de software está inconsistente com outra informação no artefato de software
Ambiguidade	•Informação contida no artefato de software é ambígua, isto é, várias interpretações podem ser derivadas da definição levando o desenvolvedor a implementação incorreta
Informação estranha	•Informação que é fornecida não é necessária ou nunca utilizada
Severidade	•Baixa, média, alta e não definida

(SHULL, 1998)

59

Defeitos no PDS - consequências

Estimativas	• Estimativas de esforço e prazo deixam de fazer sentido
Desperdício de recursos	• Em projetos recentes de software, teríamos um esforço de retrabalho entre 40% e 50% do esforço total.
Inconsistência	• Produto final não atende as necessidades do usuário
Ampliação do custo	• corrigir defeitos após a entrega do produto pode ser até cem vezes mais caro que corrigi-los nas primeiras fases do desenvolvimento (em projetos menores, 5:1);

60

Revisões de Software

Revisão

- Em uma revisão um grupo de pessoas além do autor examina um produto de trabalho para encontrar defeitos e identificar oportunidades de melhoria.

Benefícios

- Os benefícios das revisões não estão limitados a encontrar defeitos mas obter sugestões de melhoria inclusive para trabalhos futuros.

Métodos

- O termo revisão, inspeção e walkthrough algumas vezes são usados de maneira intercambiável mas representam diferentes métodos para executar revisões.

61

Revisões de Software

Investimento

- O investimento para garantir a qualidade de um produto é retornado pela redução de custos na correção de defeitos e pela diminuição de problemas reportados pelo cliente.

Retorno do investimento

- Se o custo de estabelecer e sustentar um programa de revisão com treinamento, reuniões for menor do que a economia na redução do retrabalho e satisfação do cliente o investimento tem retorno.

62

Revisões de Software - ROI

Inspeções de projeto	• Taxa recomendada de sete páginas por hora para a preparação e inspeção.
Exemplo	• Supondo um documento de 35 páginas • 1 inspetor revisa 7 páginas por hora • 5 horas de inspeção
Inspeções de requisitos	• Geralmente são mais demoradas

63

Revisões de Software - ROI

Inspeções de código	• Taxa recomendada de 150 linhas por hora para a preparação e inspeção.
Exemplo	• Supondo uma aplicação com 1000 linhas • 1 inspetor revisa 150 linhas por hora • 1000 / 150 = 6.66 horas para 1 inspetor revisar esta aplicação

64

Revisões de Software - métricas

Densidade de defeitos	• Numero de defeitos encontrados por unidade de material inspecionado.
Esforço de inspeção	• Total de horas gasta na inspeção
Esforço por defeito	• Média total de horas gasta para encontrar um defeito
Retrabalho por defeito	• Média do número de horas necessários para verificar e corrigir um defeito

65

Revisões de Software

Retrabalho	• Organizações de desenvolvimento de software podem consumir de 40% a 60% do total do esforço de desenvolvimento com retrabalho.
Raytheon Electronic Systems	• A implementação de um programa de inspeção permitiu a Raytheon Electronic Systems reduzir o nível de retrabalho de 41% do total do custo do projeto para 20% em 2 anos.
Space Shuttle Onboard Software	• Mediu o custo relativo da correção de um erro de projeto ou código: • $1 se encontrado durante uma inspeção • $13 se encontrado durante o teste de sistema • $92 se encontrado depois da entrega

66

Revisões de Software

Amplificação dos defeitos	• O fator de amplificação é menos severo para pequenas empresas ou sistemas não críticos, mas nunca é zero.
Dados da industria	• Poucos dados estão disponíveis para quantificar exatamente o ROI para uma organização especifica.
Revisões como atividades de QA	• Revisões podem não encontrar todos os defeitos mas devem ser parte do arsenal para detecção de defeitos.

67

Revisões e Teste de Software

Revisão	• É um método de verificação que pode ser utilizado para melhorar produtos de trabalho antes que tenham sido concluídos.
Técnicas de Teste Dinâmico	• Permitem encontrar defeitos somente depois que o produto tenha sido concluído. Em um estagio de teste é improvável que se remova mais do que 35% de erros no produto de trabalho, em contraste com inspeções de código e projeto que tipicamente removem entre 50% e 60% dos defeitos presentes
VV&T	• A combinação de revisões com analisadores de código estático e dinâmico, teste, ferramentas de analise de cobertura oferecem mecanismos poderosos para detecção de defeitos.

68

Aspecto Cultural

Interação social	• Revisões envolvem mais interação social do que praticas técnicas.
Relutância	• Muitos profissionais são relutantes em gastar tempo examinando o trabalho de colegas.
Evidência de benefícios	• A gerencia deve ser orientada a compreender os benefícios para o projeto do tempo gasto com revisões planejando recursos para realização desta atividade

69

Aspecto Cultural

Proteção contra críticas	• A auto-proteção contra a critica leva programadores a sentirem o ego ferido quando erros são identificados em seu trabalho (Weinberg, 1998)
Programação sem ego	• A programação "sem ego" considera que o produto de trabalho deve ser facilmente compreendido por outros e aceita melhorias para facilitar entendimento.

Weinberg, The Psychology of Computer Programming, 1998

70

Escala de Maturidade

1. Revisões não são realizadas
2. Revisões “ad hoc” são realizadas
3. Revisões informais são realizadas
4. Inspeções são planejadas e mantidas, um processo de revisão e inspeção é adotado
5. Um programa de inspeção é definido e gerenciado, inspetores são treinados, inspeções são reconhecidas como atividade critica para o sucesso do projeto, dados das revisões são analisados para melhoria do processo.

71

Tipos de Revisões

Classificação

- Revisões podem ser classificadas de acordo com o grau de formalidade ou seu nível relativo de disciplina e flexibilidade.

(WIEGERS, 2002)

72

Tipos de Revisão

Revisões informais	• Revisões informais podem atender as necessidades em certas situações.
Baixo custo	• São rápidas, baratas, não exigem um planejamento complexo, não demandam infra-estrutura organizacional e podem ajudar o autor a realizar uma melhoria no encaminhamento do trabalho.
Restrições orçamentárias e de tempo	• As forças e limitações das varias abordagens ajudam a organização a selecionar um tipo de revisão para cada situação que atenda sua cultura, restrições de tempo, de negócio e objetivos técnicos.

73

Atividades típicas

	Activity				
Review Type	**Planning**	**Preparation**	**Meeting**	**Correction**	**Verification**
Inspection	Yes	Yes	Yes	Yes	Yes
Team Review	Yes	Yes	Yes	Yes	No
Walkthrough	Yes	No	Yes	Yes	No
Pair Programming	Yes	No	Continuous	Yes	Yes
Peer Deskcheck, Passaround	No	Yes	Possibly	Yes	No
Ad Hoc Review	No	No	Yes	Yes	No

(WIEGERS, 2002)

74

Tipos de Revisão - inspeção

Inspeção

- Inspeção é o tipo mais sistemático e rigoroso dos tipos de revisão.

Processo

- Um processo genérico de inspeção envolve: planejamento, preparação, retrabalho, acompanhamento e uma analise de causa e efeito.

Maximizar a Eficácia

- Para se maximizar a eficácia, inspetores devem ser treinados no processo de inspeção.

75

Tipos de Revisão - inspeção

Uso de checklists

- Inspeções contam com *checklists* de defeitos comumente encontrados em diferentes tipos de produtos de trabalho de software.

Papéis

- Uma característica importante da inspeção é que o líder da inspeção (moderador) e o apresentador do material para inspeção (leitor) não podem ser o autor.

76

Tipos de Revisão - inspeção

Dinâmica	• Durante a reunião o leitor apresenta uma pequena parte do material por vez permitindo que os inspetores façam perguntas e apontem possíveis defeitos.
Melhorar o entendimento	• Os inspetores podem comparar seu entendimento com o entendimento expresso pelo leitor.

77

Tipos de Revisão - inspeção

Resultado da reunião	• Ao final da reunião os inspetores decidem se existe necessidades de mudanças e se estas mudanças serão alvo de uma nova reunião.
Produtos de alto risco	• Inspeções são especialmente importantes para produtos de alto-risco que devem ser livres de defeitos tanto quanto possível.
Dinâmica	• A interação entre os inspetores durante a reunião pode revelar novos problemas quando a observação de um participante estimula o pensamento dos outros.

78

Tipos de Revisão - inspeção

Perfil da equipe	• Os revisores devem ser selecionados de maneira que o material seja adequadamente coberto (hardware/software)
Tamanho da equipe	• O tamanho do grupo depende do material a ser coberto e das habilidades em revisões dos participantes.
Sugestão	• Três ou quatro participantes no mínimo no máximo 7 e no máximo 2h de duração.

79

Tipos de Revisão - inspeção

Perfil da equipe	• É relevante a participação do usuário em todos os produtos de trabalho no qual ele pode interagir isto pode incluir: especificações e manuais, mensagens de erro, procedimentos de operação e critérios de teste.
Artefatos técnicos	• Dependendo do envolvimento técnico do usuário poderá haver seu envolvimento mesmo em revisões de projeto e codificação.
Participação como ouvintes	• Exemplo equipe de teste

80

Tipos de Revisão - inspeção

Evitar que o autor se sinta pressionado	• Para manter os esforços direcionados para o produto as pessoas que avaliam seu autor não devem estar presentes na revisão do produto
Foco no processo	• O autor pode ficar mais preocupado em parecer bem aos olhos do seu chefe.
Problemas de afinidade pessoal	• Pessoas são incapazes de trabalhar juntas devem ser evitadas.

81

Bibliografia complementar

- WIEGERS, K.E., Peer Reviews in Software: A Practical Guide, Boston:Addison Wesley, 2002.
- Gilb, Tom, and Dorothy Graham. Software Inspection. Addison Wesley: 1993.
- Radice, Ronald A. High Quality Low Cost Software Inspections. Paradoxicon Publishing: 2002. (http://www.stt.com/BookInspections.html)
- Software Inspection and Review Organization website: http://www.ics.Hawaii.Edu/~siro which has a link to Philip Johnson's Formal Technical Review pages.
- Weller, Ed. Defect Depletion and Cost Analysis Spreadsheet.
- SHULL, F. Developing Techniques for Using Software Documents: A Series of Empirical Studies, Ph.D. Thesis, University of Maryland, College Park, 1998

82