A MANUTENIBILIDADE COMO FATOR PARA A QUALIDADE DO SOFTWARE

Freire, Elias Marques; Ferreira, Welinton Resende; Venâncio, Otair dos Reis; Santos, Davidson Gabriel dos (1); Reis, José Claudio de Sousa (2)

(1) Acadêmicos do Curso de Bacharelado em Ciência da Computação da UNIFENAS;
(2) Orientador

Resumo

Este artigo foi desenvolvido baseado na dificuldade das empresas em prestar manutenção ao software desenvolvido, para amenizar essa dificuldade é importante dar atenção á qualidade do software durante seu desenvolvimento. Para atingir tal meta uma atenção especial foi dada aos fatores que influenciam diretamente a capacidade de se prestar manutenção ao software.

Para tal foi desenvolvido um software de Banco de Questões para o armazenamento de questões de prova, montagem de avaliações e nele foram aplicadas as métricas de Concisão, Consistência, Instrumentação, Modularidade, Autodocumentação e Simplicidade. E com os resultados obtidos podemos observar como a manutenibilidade foi melhorada.

Abstract

That article as developed based in difficulty on companies to provide maintenance to developed software, for ease that difficulty its important give attention to Quality of Software during you development. To achieve this goal a special attention as given to the factors as influence directly the ability to provide maintained the software.

Was developed for such a software question bank for storage of questions, mounting reviews and on him as apply the metrics of Concision, Consistence, Instrumentation, Modularity, Self-Documentation and Simplicity. And with the obtained results we can observe how the maintainability was improved.

1. INTRODUÇÃO

No princípio da informática os softwares eram gratuitos e livremente distribuídos em formato fonte (código-fonte), pois havia poucos computadores, e, o valor real estava na própria máquina e não nos programas, vender software era algo inimaginável. Com o crescimento da indústria tecnológica, o software começou a ser vendido separadamente da máquina. Isso fez com que o mercado de softwares se tornasse cada vez mais competitivo, e, para se defender da acirrada concorrência, as empresas começaram a se preocupar em fazer softwares com melhor qualidade.

Esse termo, qualidade, é um conceito que vem muito antes do software em si, você provavelmente já ouviu falar em qualidade por ai, qualidade de vida, a qualidade da agua, a qualidade de um alimento, ou a qualidade deste texto, é a excelência, de um produto ou de um serviço prestado. Com o software não é diferente, se uma empresa tem uma preocupação com a qualidade ela se destaca de empresas que não tem essa preocupação, e, garante que o produto saia exatamente ou o mais próximo possível do que o cliente deseja. Organizações que possuem programas de baixa qualidade, seja para uso interno ou para venda, está essencialmente sempre "correndo atrás" da concorrência já que, fica gastando tempo e dinheiro na manutenção de softwares prontos.

> "A manutenção de software é um fator de extrema importância, pois, é considerada como a fase onde a organização desprende maior esforço, consumindo em torno de 70% da atenção do pessoal de TI, e, este percentual continua aumentando gradativamente" (PRESSMAN, 1995 p. 876).

Isto faz com que as empresas sigam regras dentro de um padrão de desenvolvimento, permitindo assim que os programas sejam entendidos, corrigidos, adaptados ou melhorados com maior facilidade, não dependendo apenas de quem o desenvolveu.

1.1. Objetivo

Desenvolver um software SISTEMA GERENCIADOR DE BANCO DE QUESTÕES (SGBQ), aplicando em todas as etapas as métricas de qualidade referente à manutenibilidade.

2. REFERENCIAL TEÓRICO

2.1. Histórico sobre a qualidade

A preocupação com a Qualidade do Software pode ser uma coisa nova, afinal surgiu em meados da década de 70 sob um contrato militar (PRESSMAN, 1995), porém qualidade sempre foi uma preocupação da humanidade. Desde a época das pirâmides do Egito até grandes construções: Templos da Grécia antiga, Grandes Feitos de Navegação, catedrais europeias (KOSCIANSKI, [2008]). Todos esses feitos só foram possíveis ao se pensar primeiro em qualidade, para isso eram usados os métodos que eram considerados os melhores em sua época. Apos tantos anos de avanço e pesquisa por melhorias, as construções de hoje em dia tem padrões avançados e bem definidos.

Nos primórdios da computação, décadas de 1950 e 1960, a qualidade era uma responsabilidade exclusiva do programador (PRESSMAN, 1995), programadores tem por instinto verem a sua obra como uma “arte”, então cada programa tinha seu próprio “padrão de qualidade”. Após 1970 começou a se pensar no termo Métricas de Qualidade de Software, e a partir de então até os dias de hoje vem se pensando e métodos para avaliar a qualidade do software a ser produzido (BUENO, 2012).

2.2 Qualidade de software

Ao pensar no termo “Qualidade de Software”, encontra-se uma dificuldade: O que é Qualidade afinal? Bueno cita a definição no dicionário Aurélio sobre a qualidade: “propriedade atributo ou condição das coisas ou das pessoas capaz de distingui-las umas das outras e lhes determinar a natureza” Bueno (2012, p. 1). Nesse conceito Qualidade se refere a coisas que podem ser medidas. Quando vamos analisar os objetos a nossa volta: roupas, música, comida, filmes prestamos atenção em suas características e, mesmo assim, temos dificuldades em escolher qual possuir a melhor qualidade, a maior prova disso é que existem clientes que compram produtos com marcas diferentes e mesmo assim estão felizes com isso.

Segundo Pressman (PRESSMAN, 1995), Software, no entanto é uma atividade Intelectual algumas de suas características podem ser mensuradas diretamente, outras não, as que dividem a avaliação em dois grandes grupos.

- Fatores que podem ser medidos diretamente: Erros, Unidades de tempo, Capacidade da Rede, etc.
- Fatores que podem ser medidos apenas Indiretamente: Usabilidade, Interoperabilidade, Manutenibilidade, etc.

Obviamente esse agrupamento pode ser modificado e ampliado, na verdade esse assunto é amplo e gera intermináveis discussões sobre o tema, devemos primeiramente identificar os fatores de qualidade do software.

2.2.1 Fatores de qualidade de software

Os fatores irão definir a qualidade, mas eles vão sofrer variações que irão depender de(KOSCIANSKI, [2008]):

- Da aplicação em si, sua complexidade, função;
- Do cliente que a solicita;
- Do custo; afinal cada empresa tem seu nível de investimento.

Uma vez definidos os fatores temos que medi-los, então como medir um fator?

- Primeiro: precisamos definir quais características medir, e como medi-las;
- Segundo: comparar dados para se chegar a uma indicação de qualidade;

Através dessa medição pode-se avaliar a qualidade do software, é muito comum o uso de medição no mundo das engenharias, infelizmente quando se trata da engenharia de Software não é tão simples, ele não é baseado em leis quantitativas básicas e medidas absolutas não são comuns (BUENO, 2012). Através de uma derivação dessas medidas indiretas que podemos chegar a um indicativo de qualidade.

Basicamente McCall classificou 11 fatores que afetariam a qualidade do software, e estes, distribuídos dentro dos três categorias formando uma pirâmide:

Figura 1 - Fatores de Qualidade de Software de McCall

Fonte: Pressman (1995, p. 726)

Para se ter uma ideia da importância do modelo criado por ele, modelos subsequentes foram criados(GALIN, 2004):

- Em 1988, baseado no trabalho de McCall, Deustch e Willis criaram um modelo com 12 a 15 fatores que afetariam a qualidade do software;
- Em 1987 Evans e Marciniak criaram modelos alternativos, não muito diferentes do modelo de McCall.

Este modelo, estudado e revisado, permanece um dos mais utilizados para a classificação dos requerimentos dos métodos (GALIN, 2004). Como Observado no gráfico, após um quarto de século e de “maturação”, o trabalho de McCall continua sendo, cada categoria agrupa requisitos (fatores) que influenciam na qualidade do software (GALIN, 2004).

2.2.2 Métricas de qualidade de software

Segundo (PRESSMAN, 1995), muitas das métricas sugeridas por McCall só podem ser medidas subjetivamente, mas elas podem estar no formato de lista de conferência (ou se preferir o termo *checklist*). Neste formato podemos graduar atributos específicos do software numa escala que varia de 0 a 10 utilizando as

seguintes
métricas:

Fator de Qualidade / Métricas de Qualidade de Software	Corretitude	Confiabilidade	Eficiência	Integridade	Manutenibilidade	Flexibilidade	Testabilidade	Portabilidade	Reusabilidade	Interoperabilidade	Usabilidade
Auditabilidade				X			X				
Acurácia	X										
Comunidade de comunicação										X	
Interoperabilidade	X										
Complexidade		X				X	X				
Concisão			X		X	X					
Consistência	X	X			X	X					
Comunidade de Dados										X	
Tolerancia a Erros		X									
Eficiência de Execução			X								
Expansibilidade						X					
Generalidade						X		X	X	X	
Independência de hardware								X	X		
Instrumentação				X	X		X				
Modularidade		X			X	X	X	X	X	X	
Operabilidade			X								X
Segurança				X							
Autodocumentação					X	X	X	X	X		
Simplicidade		X			X	X	X				
Independência do Software Básico								X	X		
Rastreabilidade	X										
Treinamento											X

Figura 2 - Fatores e Métricas de Qualidade
Fonte: Pressman (1995, p. 730,731)

Esses conjuntos de requisitos, de métricas, quando seguidos rigorosamente e aplicados ao produto garantem a qualidade, por isso também podemos chamar de ferramentas de SQA.

2.2.3 Garantia de qualidade de software

Todas essas ferramentas descritas sevem para um propósito, a garantia da qualidade de software ou SQA (*Software Quality Assurance*) e são muito mais abrangentes do que foi descrito nesta monografia, ela se estende desde a fase inicial do projeto do software, de seu funcionamento, sua manutenção, até as pessoas envolvidas diretamente ou indiretamente no projeto, onde cada equipe ou parte da equipe tem seu papel a cumprir e normas a seguir para que o software tenha de fato a “garantia” para o cliente de que o produto está sendo feito do jeito

que ele e/ou sua empresa precisam (GALIN, 2004).

Garantir a qualidade do software significa que o software a partir do ponto de vista do cliente (PRESSMAN, 1995), cumprir os requisitos baseado no que ele deseja para o produto, afinal ele é a pessoa mais interessada no produto final e é para ele que se está sendo gasto tempo e pessoal qualificado. Dentro de tantos fatores, a Manutenibilidade é um dos fatores de qualidade que devem ser avaliados para garantir a SQA do software, e, o objetivo é demonstrar a importância crucial que esse quesito de qualidade tem além do impacto que uma empresa pode ter negativamente quando esse fator é ignorado.

2.3 Manutenção de software

2.3.1 O conceito de manutenção

“Não é difícil associar a palavra, tradicionalmente quando se diz que um software precisa de manutenção logo se imagina, temos um software que parou de funcionar e precisamos repará-lo” (Souza, 2012). "Parece uma atividade simples a princípio, mas na verdade se trata de uma ponta de iceberg em que muitas empresas se perdem e gastam em torno de 70% do tempo nessa atividade”.

“É normal duvidar desses fatos. Alguém pode dizer que não gasta todo esse tempo consertando erros, e, tem toda razão quanto a esse ponto.” (PRESSAN, 1995).

> Manutenção de software é a totalidade de atividades necessárias para prover, minimizando o custo, suporte a um sistema de software. As atividades são executadas tanto nos estágios pré-entrega quanto nos pós-entrega. As atividades de pré-entrega incluem o planejamento para entrada em operação, suportabilidade e noção de logística. As atividades de pós-entrega incluem modificação do software treinamento e operação de um Help-Desk (PRESSAN, 1995).

2.3.2 Estrutura da manutenção

Toda manutenção deve ser estrutura e seguir algumas regras, um software foi construído com base em técnicas de qualidade e a manutenção não deve fugir a essa regra. Quando se recebe um pedido de manutenção, primeiro

deve-se avaliar o impacto das modificações do software antes e depois da modificação sugerida, assim é possível estipular um prazo para a realização da mesma e documentar tudo o que será feito. (AGUAYO, 2012)

Nesse ponto muitos programadores encontram algumas dificuldades ao se tentar realizar uma boa manutenção (CORDEIRO, 2012):

- Alguns softwares, geralmente mais antigos não foram feitos com desprezo à qualidade de software, consequentemente não possuem uma documentação para a avaliação do mesmo, ou se possui, ela se deteriorou com o tempo;
- O módulo do software que será modificado pode ter forte vinculação com outros módulos já existentes, qualquer modificação pode causar erros nesse outros módulos;
- Alta complexidade algorítmica do software o torna de difícil manutenção, já que exige compreensão do mesmo;
- Um software mal configurado pode gerar inconsistências quando qualquer modificação for feita;
- A inexistência de um ambiente para teste da manutenção realizada, o software fica exposto logo após a manutenção e também seus erros;
- Alterações frequentes no software ocasionadas por motivos diversos, quanto mais se modifica um software, maior o risco de surgir um novo problema;

Diante dessas dificuldades, é importante que antes de cada abordagem de manutenção tomemos alguns cuidados, executando-a em etapas.

2.3.3 Etapas da manutenção

Um pedido de avaliação deve ser avaliado, dependendo da manutenção ela segue por caminhos diferentes para ser executado, existem determinados tipos de manutenção (BRUSAMOLIN, 2012):

- Corretiva: manutenção necessária para corrigir erros;
- Adaptativa: o ambiente se modifica; isso provoca a necessidade do software se adaptar ao ambiente modificado;
- Perfectiva: manutenção com o objetivo de melhorar o software.

Essa definição facilita o processo de manutenção, reduzindo a quantidade de esforço gasto na manutenção (CORDEIRO, 2012).

As etapas para a manutenção do software seguem os seguintes passos (CORDEIRO, 2012):

- Avaliar o pedido de mudança, para ver sua real necessidade;
- Qual o impacto dela no sistema e sua integrações;
- Estimar o custo e o tempo para a sua realização;
- Implementar as modificações;
- Testar os módulos modificados;
- Testar a integração do sistema;
- Atualizar a Documentação;
- Integrar

Devido à alta demanda de manutenção e complexidade ao longo dos últimos anos veio surgindo um novo conceito, manutenibilidade, conceito que tem como objetivo facilitar a manutenção tomando cuidado com determinadas etapas durante o processo de desenvolvimento e que tem como objetivo, reduzir o custo (CORDEIRO, 2012).

3. MATERIAL E MÉTODOS

3.1. Etapas do trabalho

Etapa 1: Levantamento Bibliográfico.

Etapa 2: Definição do tema da aplicação a ser construída utilizando políticas de qualidade e de segurança.

Etapa 3: Desenvolvimento e aplicação de técnicas de Manutenibilidade

Etapa 4: Análise do trabalho quanto à qualidade de seu produto final e sua confiabilidade de acordo com a Etapa 3, dificuldades encontradas e os resultados obtidos.

3.2. Ferramentas e tecnologias utilizadas

Para o desenvolvimento do Software de Gerenciamento de Banco de Questões (SGBQ), utilizou-se o ASP.NET e a linguagem C#, através da ferramenta de desenvolvimento Visual Studio 2010 *Ultimate,* e, através da mesma, utilizou-se a linguagem SQL para o gerenciamento do Banco de Dados a ser construído com a Ferramenta SQL Server 2008.

4. DESENVOLVIMENTO DO APLICATIVO SGBQ

4.1. Descrição Geral do Sistema

Desenvolveu – se o Sistema SGBQ para testar os procedimentos de qualidade de software voltados para a manutenibilidade, o software SGBQ tem o intuito de facilitar o professor a armazenar as questões para uso em avaliações de uma maneira prática e criar questionários para impressões.

4.1.1.Tela Principal – Menu do Sistema

Figura ? - Tela Principal

A tela principal do Sistema possui um menu para acesso a todas as áreas do software, porém, dependendo do seu nível de acesso você será redirecionado para essa tela principal e informado sobre seu acesso restrito. A tela principal também dá boas vindas ao usuário de forma amigável e personalizado, exibindo o nome do mesmo.

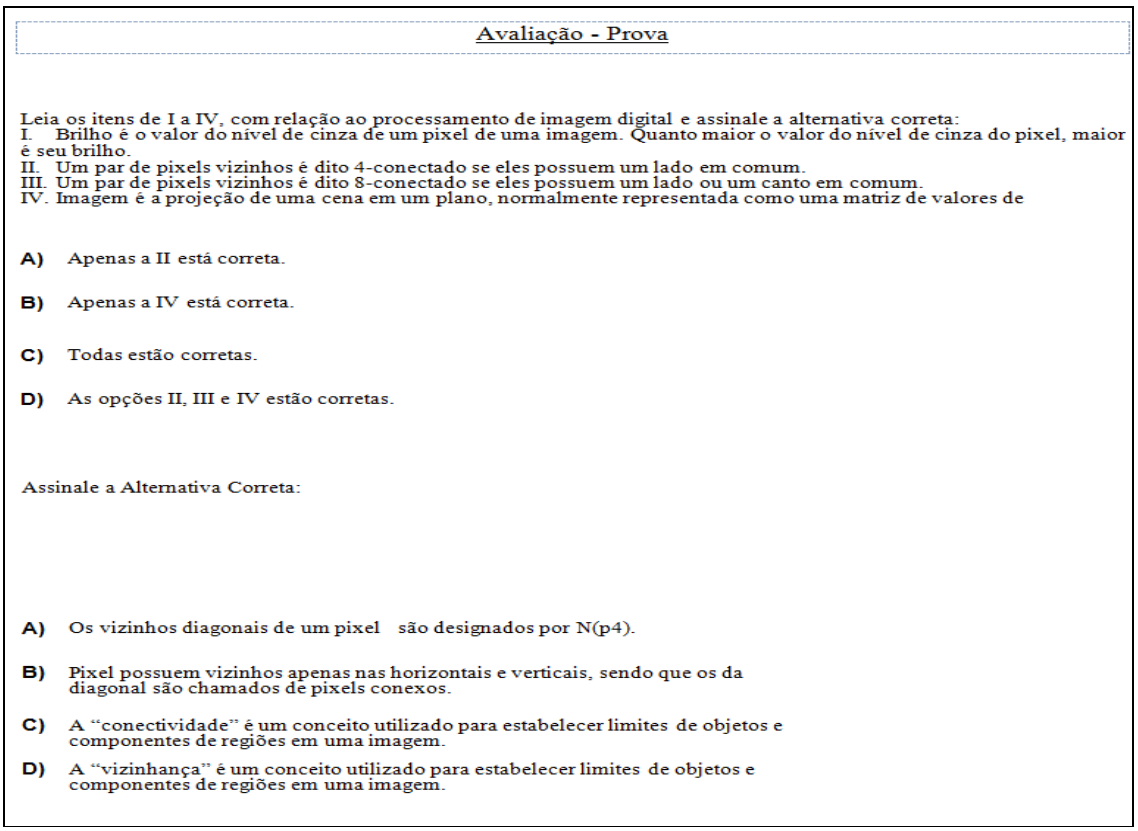
4.1.6. Gerar Prova

Figura 8 – Tela 1 de Gerar Prova

Tela para gerar uma prova de acordo com as questões cadastradas, a questão pode ser inserida diretamente pelo código da questão, ou caso o usuário não saiba o código, pode ser pesquisada de acordo com seu curso e disciplina escolhidos nas primeiras opções, tamanho da prova precisa ser definido para que o sistema saiba quantas questões possuirá a avaliação. Após a consulta o usuário pode conferir o título da questão e adicioná-la a prova, ou cancelar a ação.

Figura 9 – Tela 2 de Gerar Prova

Após a prova ser adicionada um trecho de seu título será exibido nessa lista para que o usuário tenha noção de quantos itens a prova já possui e quais são eles, com um duplo-clique ele pode excluir o item da lista caso ele ache necessário e após definido os itens da prova ela pode ser gerada através do botão Gerar prova.



Avaliação - Prova

Leia os itens de I a IV, com relação ao processamento de imagem digital e assinale a alternativa correta:
I. Brilho é o valor do nível de cinza de um pixel de uma imagem. Quanto maior o valor do nível de cinza do pixel, maior é seu brilho.
II. Um par de pixels vizinhos é dito 4-conectado se eles possuem um lado em comum.
III. Um par de pixels vizinhos é dito 8-conectado se eles possuem um lado ou um canto em comum.
IV. Imagem é a projeção de uma cena em um plano, normalmente representada como uma matriz de valores de

A) Apenas a II está correta.

B) Apenas a IV está correta.

C) Todas estão corretas.

D) As opções II, III e IV estão corretas.

Assinale a Alternativa Correta:

A) Os vizinhos diagonais de um pixel são designados por N(p4).

B) Pixel possuem vizinhos apenas nas horizontais e verticais, sendo que os da diagonal são chamados de pixels conexos.

C) A "conectividade" é um conceito utilizado para estabelecer limites de objetos e componentes de regiões em uma imagem.

D) A "vizinhança" é um conceito utilizado para estabelecer limites de objetos e componentes de regiões em uma imagem.

A área de processamento de imagens vem apresentando crescimento expressivo nos últimos anos e suas aplicações atingem quase todas as áreas das atividades humanas, porém para sua utilização é necessário cumprir algumas etapas de processamento de imagem. Dentre essas etapas, qual das opções abaixo representa o primeiro passo no processamento de imagens.

A) Aquisição de imagem.

B) Segmentação.

C) Reconhecimento e interpretação.

D) Descrição.

Um sistema de visão artificial deve ser capaz de adquirir, processar e interpretar imagens. Quanto à etapa de pré-processamento de um sistema de visão artificial, podemos afirmar:

A) A etapa de pré-processamento procura aprimorar a qualidade da imagem para as próximas etapas. As operações efetuadas nesta etapa trabalham

B) A etapa de pré-processamento procura dividir uma imagem em suas unidades significativas, ou seja, nos objetos de interesse que a compõe.

C) A etapa de pré-processamento procura extrair características das imagens resultantes da segmentação, através de descritores que devem ser

D) A etapa de pré-processamento procura reconhecer a imagem para as próximas etapas, sem se preocupar com a qualidade resultante.

Figura 10 – Modelo de Prova Gerada.

A figura 10 apresenta um modelo de prova gerada pelo software, através do processamento de imagens. A prova é gerada em arquivo .doc, dando liberdade ao usuário para manipular o arquivo a sua necessidade.

5. RESULTADOS E DISCUSSÃO

Verificou–se através de análise do código, que o sistema sem a aplicação das métricas de manutenibilidade mostrava-se altamente complicado o seu entendimento.

Após a inclusão das métricas todos os problemas relacionados ao desenvolvimento foram tratados de forma separada, o código se tornou modular sendo mais fácil na identificação dos erros.

Outra vantagem do uso das métricas é a independência que se tem em relação aos desenvolvedores, pois o código está mais legível.

É importante observar que a cada modificação, são necessários que sejam aplicados as métricas novamente e toda a documentação também seja modificada, isso manterá a longevidade do sistema, caso contrário, todo o trabalho para que o sistema possuísse uma manutenção facilitada será perdido em pouco tempo durante os pedidos de alterações.

6. CONCLUSÃO

Foi desenvolvido um software de Banco de Questões Acadêmicas (SGBQ) aplicando em todas as etapas as métricas de qualidade referente a manutenibilidade.

Verificou-se através de inúmeras manutenções realizadas no sistema SGBQ, uma melhora significativa na facilidade de manutenção/alteração do código.

Pode-se também verificar, que após a inclusão das métricas de qualidade para a manutenibilidade, houve uma significativa diminuição do tempo/recursos gastos para as alterações no software SGBQ.

REFERÊNCIAS

BRUSAMOLIN, Valério. Manutenibilidade de Software. Instituto Científico de Ensino Superior e Pesquisa – ICESP. Disponível em: http://www.revdigonline.com/artigos_download/art_10.pdf.. Acesso em 10/04/2012.

BUENO, Cassiane de Fátima dos Santos; CAMPELO, Gustavo Bueno. **Qualidade de Software.** Departamento de Informática: Universidade Federal de Pernambuco. Recife, PE. Disponível em: <http://www.cin.ufpe.br/~mrsj/Qualidade/Qualidade%20de%20Software.pdf>. Acesso em: 08 de maio de 2012.

CORDEIRO, Marco Aurélio. Manutenibilidade de Software - Site BateByte. Disponível AGUAYO, Maria Teresa Villalobos, GUERRA, Ana Cervigna, COLOMBO, Regina Maria Thienne. Processo de Avaliação da Manutenibilidade de Produtos de Software. Disponível em http://www.angelicatoffano.pro.br/upload_arquivos/pt/A08_3_artigo14730.pdf.
CROSBY, Philip. **Quality Withouth Tears.** 2. ed. New York: McGraw-Hill, 1984.

GALIN, Daniel. **Software Quality Assurance – From the Theory to Implementation.** 1. ed. [S.l.]: Pearson Education Limited, 2004.

JURAN, Joseph Moses. **Planning for Quality.** 4. ed. New York: The free Pass, 1988.

KOSCIANSKI, André; SOARES, Michel dos Santos. **Qualidade de Software: Aprenda as metodologias e técnicas mais modernas para o desenvolvimento de software.** 2. ed. [S.l.]: Novatec, [2008].

PRESSMAN, Roger S. **Engenharia de Software.** 3. ed. São Paulo: Makron Books, 1995.

RIBEIRO, Adauto Roberto. **Empresas brasileiras desenvolvedoras de software: uma avaliação das condições de qualidade e competitividade.** 1998. 134 f. Dissertação (Mestre em Economia). Universidade Estadual de Campinas. Campinas, SP.

RINCON, André Mesquita. **Qualidade de Software.** Instituto de Informática: Universidade Federal de Goiás. Disponível em: <http://www.clebertoledo.com.br/blogs/tecnologia/administracao/files/files/Qualidade_de_Software.pdf>. Acesso em: 08 de maio de 2012.