ANÁLISE COMPARATIVA DE METODOLOGIAS E PADRÕES DE QUALIDADE COM FOCO NO GERENCIAMENTO DE PROJETOS DE SOFTWARE

Autores:

Prof. MSc. Anderson Luiz Barbosa – e-mail: anderson.barbosa@am.unisal.br

Professor, Analista de Sistemas, especialista em Metodologia do Ensino Superior, mestre em Informática – Gerenciamento de Sistemas de Informação (PUC-Campinas), mestre em Engenharia Elétrica – Telemática e Telecomunicações (UNICAMP), doutorando em Engenharia Elétrica – Telemática e Telecomunicações - na Faculdade de Engenharia Elétrica e da Computação da UNICAMP; Coordenador dos cursos de Análise de Sistemas, Sistemas de Informação e Tecnologia em Software Livre do Centro UNISAL – Americana; Membro do Núcleo de Educação a Distância (NEaD).

Prof. Dr. Leonardo de Souza Mendes – e-mail: lmendes@decom.fee.unicamp.br

Engenheiro Eletricista (Universidade Gama Filho), mestre em Engenharia Elétrica pela Pontifícia Universidade Católica do Rio de Janeiro, doutor em Electrical Engineering pela Syracuse University, livre-docente - Faculdade de Engenharia Elétrica e da Computação da UNICAMP. Professor titular do Departamento de Comunicações da Faculdade de Engenharia Elétrica e da Computação da UNICAMP.

MSc. Maurício Luis Bottoli – e-mail: bottoli@decom.fee.unicamp.br

Engenheiro Eletricista (Universidade Federal de Santa Maria), mestre em Engenharia Elétrica – Telemática e Telecomunicações (UNICAMP), doutorando em Engenharia Elétrica – Telemática e Telecomunicações - na Faculdade de Engenharia Elétrica e da Computação da UNICAMP; Sócio-proprietário da empresa IgnisCom.

ANÁLISE COMPARATIVA DE METODOLOGIAS E PADRÕES DE QUALIDADE COM FOCO NO GERENCIAMENTO DE PROJETOS DE SOFTWARE

RESUMO

É mundial a preocupação com o gerenciamento adequado de projetos a fim de garantir o atendimento das expectativas e a otimização dos recursos alocados; na área de software, as preocupações com custos, prazos e qualidade são cada vez maiores, forçando um repensar nas práticas até então utilizadas para o gerenciamento. O presente artigo aborda as técnicas de engenharia de software tradicionalmente aplicadas ao desenvolvimento de projetos e apresenta uma descrição dos padrões de qualidade e de gerenciamento, tanto de uso genérico quanto os específicos para software. A partir da comparação desses modelos, é apresentada uma proposta para otimização do gerenciamento de projetos de software.

ABSTRACT

A global concern exists for the appropriate administration of projects in order to achieve the expectations of those involved. On the software area, the concern is bigger when related to costs, time and quality in software development. This forces to rethink the practices used on the administration of these processes. This work addresses the techniques of software engineering that are being applied to the development of projects; it also presents a description of the patterns of quality and management. A proposal is presented for optimizing the management of software development processes, describing the contribution provided by each one of these models, together with the necessary adaptations in order to contemplate the specific aspects of software.

PALAVRAS-CHAVE

Gerenciamento de projetos, Engenharia de Software, Qualidade de Software.

KEY WORDS

Project Management, Software Engineering, Software Quality.

1 INTRODUÇÃO

O problema do gerenciamento de projetos de *software*

A demanda pela produção de *software* cresce desde os primórdios do processamento de dados e surgimento dos computadores, há 60 anos. Desde então o nível de exigência e complexidade dos sistemas evolui à medida que novos ambientes e plataformas, novas metodologias e um número cada vez maior de usuários é envolvidos com estes sistemas.

No início os sistemas eram operados/manipulados por poucas pessoas – a complexidade operacional e a tecnologia disponível eram caras e de difícil acesso. Informações oriundas de *software* eram normalmente acessadas por intermédio de mídia impressa (listagens), com uma defasagem de tempo (atualização) grande. Hoje a demanda exige informações atualizadas quase instantaneamente, favorecida pelas diversas mídias de acesso (CD-Rom, DVD, Internet) disponíveis a um custo acessível.

Sistemas que no passado consumiam anos de desenvolvimento, com grandes equipes, hoje sofrem pressões cada vez maiores de redução de prazo e custo, sem a redução da qualidade e do escopo (ao contrário, cada vez aumentam estas exigências). Infelizmente a evolução dos processos de desenvolvimento – em especial os de gerenciamento – não acompanhou a evolução das exigências dos usuários.

Alguns trabalhos, dentre os quais Moraes (2004), mapearam a relação entre os fatores de sucesso e fracasso no desenvolvimento de projetos de *software* – boa parte dos projetos não é concluída e, dentre os concluídos, apenas uma pequena parte é concluída dentro da meta original de escopo, custo, prazo e qualidade.

Alguns autores procuraram identificar possíveis razões para estes problemas. Pressman (2006) apresenta algumas possibilidades:

> Gerentes de nível médio e superior sem nenhum background em *software* recebem a responsabilidade pelo seu desenvolvimento. Há um antigo axioma da administração que afirma: "um bom gerente pode gerir qualquer projeto". Deveríamos acrescentar: "...se ele estiver disposto a aprender que marcos podem ser usados para medir o processo, aplicar métodos efetivos de controle, não levar em conta a mitologia e tornar-se fluente numa

> tecnologia que se modifica rapidamente". O gerente deve comunicar-se com todas as pessoas envolvidas no desenvolvimento de *software* - clientes, desenvolvedores, pessoal de apoio e outros. A comunicação pode interromper-se porque as características especiais do *software* e dos problemas associados ao seu desenvolvimento são mal compreendidas. Caso isso ocorra, os problemas associados à crise do *software* são exacerbados".

Ainda hoje perguntas presentes há décadas continuam sem resposta satisfatória: por que demora tanto para que os programas sejam concluídos? Por que os custos são tão elevados? Por que os defeitos não são descobertos/sanados antes da entrega? Por que tanta dificuldade em medir o progresso do *software* na fase de desenvolvimento?

O presente artigo propõe algumas medidas gerenciais, baseadas nos conceitos de gerenciamento de projetos tão bem utilizados em outras áreas da ciência e tecnologia, que poderiam tornar o desenvolvimento de *software* um processo controlável.

Estrutura do artigo

O presente artigo está divido em seis capítulos, com as seguintes características:

O presente capítulo apresenta alguns dos problemas existentes no gerenciamento dos processos de desenvolvimento de *software*, motivação para este artigo.

O capítulo 2 apresenta uma visão geral do gerenciamento de projetos, com a apresentação dos conceitos, do modelo proposto pela International Organization for Standardization (ISO) e do modelo proposto pelo Project Management Institute (PMI); o modelo do PMI – o Project Management Body of Knowledge (PMBoK) - é a base para a verificação da hipótese de que a aplicação dos conceitos de gerenciamento de projetos pode ser um grande fator de evolução para a condução de projetos de *software*.

O capítulo 3 procura definir *software* enquanto produto e apresentar o histórico da evolução de seu desenvolvimento. Uma visão geral da disciplina engenharia de software também é apresentada, englobando os ciclos de vida dos sistemas de informação e algumas abordagens tradicionalmente utilizadas.

O capítulo 4 apresenta os modelos de qualidade propostos pela ISO, pelo Software Engineering

Institute (SEI) e pela Associação para Promoção da Excelência do Software Brasileiro (Softex), que visam garantir a qualidade, a padronização e o aperfeiçoamento dos processos produtivos.

O capítulo 5 tem por objetivo verificar os ganhos advindos da utilização do PMBoK como modelo de gerenciamento de projetos de desenvolvimento de software, comparando com os modelos de qualidade e confrontando com a experiência acumulada pelo autor na coordenação de diversos sistemas, tanto para uso interno de uma organização quanto para implementações em outras organizações, e na área acadêmica, onde leciona disciplinas relacionadas a engenharia de software, gerenciamento de projetos e análise e projeto de sistemas de informação.

Concluindo, o capítulo 6 apresenta uma síntese dos temas estudados e das conclusões oriundas das análises comparativas. Espera-se que estas conclusões possam servir para melhorar o processo de desenvolvimento de software e que inspirem novos trabalhos visando a melhoria contínua do desenvolvimento de software.

2 GERENCIAMENTO DE PROJETOS

O PMI (2004) define gerenciamento de projetos como “a aplicação de conhecimentos, habilidades, ferramentas e técnicas no sentido de concluir atividades que atendam ou excedam às necessidades e expectativas dos envolvidos deste projeto”. Para atingir esses objetivos, algumas questões devem ser observadas: escopo - tudo o que deve ser produzido; prazos - ordem cronológica dos resultados/produtos; custo - valor para a produção; qualidade - tanto do processo de condução quanto dos produtos gerados.

Um projeto pode ser definido como "empreendimento temporário - com início e término bem definidos - cujo objetivo é criar um produto ou oferecer um serviço único, distinto de todos os produtos ou serviços produzidos anteriormente". (PMI, 2004). Podem variar em tamanho, envolver uma ou diversas organizações (formando consórcios, parcerias ou outros tipos de associação). O conceito de temporalidade em projetos é fundamental; deve ser claramente determinado o início e o fim no sentido de serem identificados os objetivos. Isto é vital a fim de que seja possível determinar se um projeto está concluído (atingiu seus objetivos) ou ainda se é

impossível de se concluir. Não necessariamente este limite é pequeno - existem projetos com durações de vários anos (por exemplo, a construção de uma usina nuclear). No caso de software, após o término do projeto o mesmo pode (ao menos deveria) entrar na fase de operação.

A norma ISO 10006 (ISO, 2003), inicialmente publicada em 1997 e atualizada em 2003, se refere às atividades de gerenciamento da qualidade no tocante ao gerenciamento de projetos; provê guias para os elementos do sistema de qualidade, conceitos e práticas para aprimorar e garantir a qualidade no processo de gerenciamento de projetos. Foi desenvolvida para ser aplicada ao gerenciamento de projetos de qualquer porte/complexidade, servindo de base para que profissionais de gerenciamento de projetos e auditores de qualidade possam trocar experiências a respeito do projeto.

Faz a distinção entre os dois aspectos básicos da aplicação dos conceitos de qualidade no gerenciamento de projetos: a qualidade do processo e a qualidade do produto, considerando que ambos os aspectos são vitais para o sucesso dos projetos, requerendo uma abordagem sistemática e contínua de forma a garantir a conformidade. Também descreve os processos de gerenciamento de projetos aplicáveis à maioria dos projetos, agrupando-os em dez grupos: estratégicos; gerenciamento de interdependência; escopo; tempo; custo; recursos; pessoal; comunicação; risco; compras. A norma contempla ainda um glossário com os principais termos utilizados e uma relação entre os processos descritos com a série ISO 9000 (ISO, 2000).

Segundo PMI (2004), o PMBoK descreve um conjunto de conhecimentos referentes à atividade de gerenciamento de projetos; assim como em outras profissões, este conjunto de conhecimento é gerado e atualizado pelos acadêmicos e profissionais, através de práticas tradicionalmente utilizadas, estudos, pesquisas e outras atividades. O propósito do documento é identificar e descrever um subconjunto deste conjunto de conhecimentos referentes ao gerenciamento de projetos que tenha uma aceitação global, ou seja, reunir um subconjunto que seja aplicável à maioria dos projetos e que tenha seu valor reconhecidamente comprovado e aceito pela comunidade. A primeira iniciativa de consolidar este conjunto de conhecimento ocorreu em 1987;

em 1996 foi publicado o PMBoK 1996, com diversas atualizações em relação à proposta anterior – é a base para o formato da versão atual; em 2000 foi publicado o PMBoK 2000, com atualizações pontuais, principalmente no capítulo dedicado aos processos de gerenciamento de riscos; finalmente, em 2004, foi publicada a versão 2004 – chamada PMBoK Terceira Edição, com diversas atualizações, em especial nos processos de integração.

A fim de facilitar a compreensão do processo de gerenciamento de projetos, o PMBoK agrupa os diversos processos em: processos de iniciação: reconhecem que um dado projeto ou fase pode ser iniciado e concluído; processos de planejamento: estabelecem e mantém um esquema de trabalho que objetiva atingir às necessidades que o projeto visa atender; processos de execução: coordenam pessoas e outros recursos no sentido de realizar o planejamento estabelecido; processos de monitoramento e controle: asseguram que os objetivos do projeto estão sendo atingidos através de monitoramento e medições de progresso, tomando as medidas corretivas quando necessário, e processos de encerramento: garantem a aceitação formal do projeto ou fase, finalizando-o.

O PMBoK também divide os processos de gerenciamento de projetos em nove áreas de conhecimento: integração, escopo, tempo, custo, qualidade, recursos humanos, comunicações, riscos e aquisições em projetos.

3 DESENVOLVIMENTO DE SOFTWARE

Presman (2002) conceitua software como:

> (1) instruções (programas de computador) que, quando executadas, produzem a função e o desempenho desejados; (2) estruturas de dados que possibilitam que os programas manipulem adequadamente a informação; e (3) documentos que descrevem a operação e o uso dos programas.

Pode-se notar a preocupação em ampliar o entendimento comum de que software corresponde apenas ao programa - de fato, a complexidade atualmente existente requer tratamentos específicos ao que se chama de configuração de software, englobando, além dos aspectos já citados, o ambiente necessário ao software (bibliotecas dinâmicas, sistema operacional e tipo de

equipamento, entre outros).

O desenvolvimento de software passou por diversas etapas desde sua origem, na década de 50; a percepção de sua importância também se modificou ao longo do tempo: para Pressman (2006), enquanto a preocupação com o software era secundária até a década de 70 (uma vez que a meta era desenvolver um hardware que reduzisse o custo de processamento e armazenamento de dados), o que se vê é uma preocupação mais acentuada a partir da década de 80 (muito em função da evolução da microeletrônica, que resultou em um maior poder de computação a um custo cada vez mais baixo).

A programação de computadores nasceu como uma arte (e secundária); os desenvolvimentos de software eram feitos até que os prazos começassem a se esgotar e os custos a subir abruptamente. As pessoas envolvidas com o software eram oriundas de diversas áreas (cientistas, administradores e outros), não especializados em software, dificultando ainda mais o entendimento dos problemas relacionados à computação. Os primeiros sistemas eram especificados, desenvolvidos, operados e mantidos pelo mesmo grupo de pessoas, de forma bastante restrita - eram sistemas desenvolvidos com propósitos específicos, apenas para uma empresa; seu processamento era em lote (batch). A documentação do software era, na maior parte das vezes, negligenciada, ora não existindo (e estando apenas "na cabeça" daquele que desenvolveu o software) ora existindo, porém incompleta, ou desatualizada com o passar das manutenções.

Um segundo momento surge com o aumento do número de usuários e solicitações por novos desenvolvimentos; começa a ênfase em sistemas multiusuários, com o início da preocupação com a interface homem-máquina (até então isto não era necessário uma vez que o usuário era o próprio desenvolvedor). Com o aumento da demanda, várias empresas começaram a criar software para distribuição, com vários clientes; no início, estas empresas foram representadas pelos próprios fabricantes de hardware (o software ainda era considerado secundário). Começa então o que vários autores – Pfleeger (2004), Pressman (2006), Rezende (2005) e Sommerville

(2003), entre outros - chamam de "crise de software": os profissionais não conseguiam atender toda a demanda por novos sistemas (*backlog*) em função de um número crescente de sistemas disponibilizados para os usuários estarem em constante manutenção corretiva. Isto ocorria por diversas razões: falta de documentação adequada do software produzido; falta de entendimento das necessidades dos usuários; falta de capacitação e métodos aos profissionais envolvidos no processo de produção; constantes atrasos na entrega do software, o que poderia defasá-lo antes mesmo do início de sua utilização.

Nota-se que vários destes itens constituem ainda hoje problemas no desenvolvimento de software.

O terceiro momento no desenvolvimento de software tem início em meados da década de 70 e é marcado pelo advento da microinformática, com a expansão do número de usuários e com exigências cada vez maiores por sistemas que apresentassem informações instantaneamente e com grandes volumes de dados. O processamento de dados, anteriormente restrito aos computadores, agora também é utilizado em outros equipamentos com microprocessadores: automóveis, microondas, robôs, equipamentos de diagnóstico médico e outros. Tem-se então um problema histórico: o profissionalismo já existente no desenvolvimento da tecnologia de hardware agora se vê a frente de novos desafios com o software para estes novos equipamentos, o que faz estes profissionais buscarem alternativas para esta capacitação.

Várias empresas de software foram criadas a partir do advento do microcomputador (Microsoft, por exemplo); com a expansão deste equipamento, os usuários potenciais passaram rapidamente das centenas para centenas de milhares, gerando um mercado consumidor cada vez mais cobiçado. Neste momento também inverte-se a curva de valor: o hardware passa a ser considerado um produto primário, o software passa a ter um campo enorme de novas aplicações a serem desenvolvidas; assim é comum que o custo do software para um computador pessoal ultrapasse em várias vezes o custo do hardware deste mesmo equipamento.

A quarta geração de software é marcada pelo surgimento de novas tecnologias para o desenvolvimento do software - em especial a tecnologia de orientação a objetos. Novas

aplicações de software levaram à evolução de sistemas especialistas, sistemas baseados em redes neurais e às novas arquiteturas (cliente servidor, por exemplo).

A evolução do hardware e do software em função da disseminação da Internet trouxe novos desafios; vários autores consideram que surgiu uma quinta era de desenvolvimento de software, onde a distribuição das informações é radicalmente modificada: se existiam sistemas com centenas de milhares de potenciais usuários, este número agora pode ultrapassar aos milhões de usuários de diferentes países. O tempo para o desenvolvimento de sistemas é cada vez menor em função das exigências das empresas que pretendem atuar neste cenário; surgem novas áreas de aplicação: *E-commerce* (comércio eletrônico); ASP - *Application Services Provider*, empresas que passam a oferecer serviços de *outsourcing* de aplicações (notadamente de sistemas de gestão empresarial); CRM - *Customer Relationship Management* e SCM - *Supply Chain Management*. O grande desafio é garantir o cumprimento dos prazos, custo e escopo dos sistemas necessários neste novo ambiente com a qualidade adequada.

Ciclos de Vida de Software

Para Pressman (2006), "a engenharia de software compreende um conjunto de etapas que envolvem métodos, ferramentas e procedimentos". Essas etapas são citadas pelos autores como sendo os paradigmas, ou ciclos de vida, da engenharia de software. Um paradigma da engenharia de software é escolhido tendo como base a natureza do projeto e da aplicação, os métodos e as ferramentas a serem usados, os controles e os produtos a serem entregues. A seguir são descritos alguns dos principais ciclos de vida de software.

O ciclo de vida clássico, também conhecido como cascata ou *waterfall*, foi o primeiro estudado pela engenharia de software e ainda hoje é um dos mais difundidos e utilizados. Requer uma abordagem sistemática e seqüencial no desenvolvimento do software. Funciona de maneira seqüencial, contemplando as etapas de análise de requisitos e de sistemas, projeto, codificação, testes, implementação e manutenção. Enfrenta uma série de questionamentos quanto à sua efetividade: os projetos reais raramente seguem o fluxo seqüencial que este modelo propõe, o que

causa grande dificuldade e ansiedade a todos os envolvidos; muitas vezes é difícil para o cliente declarar todas as exigências de forma explícita, o que requer dos profissionais muita habilidade; o cliente deve ter paciência - o produto somente será disponibilizado no final do ciclo, podendo não contemplar às necessidades; alto custo de correção de problemas quando identificados tardiamente (nas fases de manutenção, testes etc.); manutenção: fase na qual o sistema passará sua maior parte - infelizmente em correção.

O ciclo de vida de prototipação é um processo que capacita o desenvolvedor a criar um modelo do software que será implementado; requer uma participação ativa do usuário, uma vez que sempre dependerá de sua avaliação para avançar para a próxima fase. As etapas normalmente aplicadas são: coleta e refinamentos dos requisitos, projeto rápido, construção, avaliação pelo cliente, refinamento, repetição de um novo ciclo a partir do projeto rápido para a incorporação das novas necessidades e, finalmente, engenharia do produto, onde uma vez definidos todos os produtos a serem gerados e dirimidas todas as dúvidas com o protótipo, o sistema propriamente dito pode ser desenvolvido utilizando conceitos de engenharia de software.

O ciclo incremental foi desenvolvido a partir do ciclo clássico, incluindo a filosofia interativa da prototipagem. Parte do princípio que um software não precisa ser concluído na sua totalidade para ser entregue aos usuários – procura dividir o desenvolvimento em partes, chamadas incrementos, funcionais, antecipando o contato dos usuários com o produto em elaboração. No primeiro incremento existe o planejamento do número total de incrementos a serem construídos, bem como a especificação das interfaces e interdependências, de modo a garantir que um incremento necessite apenas de itens já desenvolvidos em incrementos anteriores – e não posteriores - o que poderia causar inconvenientes. É uma abordagem utilizada por diversas empresas que produzem software, com grandes equipes. É importante garantir que os incrementos, em especial o primeiro, tenham boa aceitação por parte dos usuários a fim de garantir a continuidade dos demais; caso isso não seja possível, problemas de retrabalho podem acarretar em atraso e custos elevados para a conclusão do software.

O ciclo espiral foi desenvolvido a fim de conter as melhores características dos ciclos clássico e prototipação, acrescentando a análise de riscos; é definido em quatro quadrantes: planejamento (determinação dos objetivos, alternativas e restrições para o desenvolvimento da versão atual do software), análise de riscos (análise das alternativas e identificação, priorização, acompanhamento e resolução dos riscos), engenharia (desenvolvimento do produto, incorporando às versões já existentes as novas funcionalidades definidas nos quadrantes anteriores) e avaliação do cliente (avaliação dos resultados da engenharia). Ao final de cada iteração ao redor da espiral (iniciando-se do centro e avançando para fora), são geradas versões mais completas do software. Vale notar que, na etapa de engenharia, a técnica de prototipação poderá ser utilizada a etapa de análise de riscos tenha identificado um volume de incertezas que justifique a construção de um protótipo; caso contrário, a abordagem clássica poderá ser utilizada para desenvolver as novas funcionalidades. Componentes previamente existentes podem ser utilizados – para alguns autores, como Pressman (2006) e Peters e Pedrycz (2001), essa utilização de componentes gera um novo paradigma – o modelo de componentes.

A incorporação da análise de riscos é a grande virtude do modelo espiral. Incorpora a necessidade de uma avaliação criteriosa dos riscos envolvidos com o projeto e a decisão de prosseguir, ou não, com o desenvolvimento; esta decisão, que no ciclo de vida clássico muitas vezes não é visível até que seja tarde de mais (muitas vezes apenas após a etapa de codificação, quando o sistema começa a ser testado visando a implantação), é tomada baseada nos levantamento e requisitos da atual fase de desenvolvimento. Uma preocupação constante é manter o entendimento de todos os envolvidos de que a abordagem deste ciclo é evolutiva e requer a participação ativa e constante de todos. Um dos principais benefícios é a entrega de produtos intermediários já operacionais, que podem – e devem - ser utilizados a fim de fornecer subsídios para o planejamento do próximo ciclo de desenvolvimento.

O modelo de métodos formais parte da especificação do software com a utilização de métodos matemáticos; tem como principal objetivo a especificação de um software livre de ambigüidades,

causa comum de problemas no desenvolvimento de software. É um processo mais lento e normalmente utilizado em partes de um determinado software, dificilmente no desenvolvimento de um sistema completo. Moura (2001) apresenta a concepção desse modelo, com enfoque na fase de especificação, utilizando a notação Z. Pressman (2006) e Sommerville (2003) também apresentam a utilização de métodos formais no desenvolvimento de software.

Ambler (2004) descreve os princípios da Modelagem Ágil e sua relação com Programação Extrema (XP). Os principais objetivos são acelerar o processo de desenvolvimento de aplicações através da utilização de processos incrementais e interativos; procura dividir o produto a ser entregue em pequenas partes, de forma a serem especificadas, desenvolvidas, testadas e entregues rapidamente. Pelo fato de ser uma metodologia muito recente, os resultados práticos de sua adoção ainda são questionáveis, em especial por parte das empresas que produzem software – ainda é necessário desenvolvimento e acompanhamento para verificar sua viabilidade.

4 MODELOS E PADRÕES DE QUALIDADE

São diversas as iniciativas, tanto de uso geral quanto de uso específico para software.

A norma ISO 90003 (ISO, 2004b), teve origem com a norma ISO 9000-3:1997, onde são apresentados os roteiros para aplicar a ISO 9001:1994 (ISO, 1994), especificamente na área de desenvolvimento, fornecimento e manutenção de software. Todas as orientações giram em torno de uma "situação contratual", onde uma empresa contrata a empresa em questão para desenvolver um produto de software. A norma aborda três grandes blocos: 1) estrutura do sistema de qualidade: responsabilidade do fornecedor e do comprador e análise crítica conjunta; 2) atividades do ciclo de vida: análise crítica do contrato, especificação dos requisitos do comprador, planejamento do desenvolvimento, projeto e implementação, testes e validação, aceitação, cópia, entrega, instalação e manutenção; e 3) atividades de apoio: gerenciamento de configuração, controle de documentos, registros da qualidade, medição, regras e convenções, aquisição, produto de software e treinamento.

A norma ISO/IEC 9126 (ISO, 2001) representa a proposta da ISO para padronização mundial

dos aspectos relativos à qualidade de produtos de software, tendo sido publicada em 1991; lista o conjunto de características que devem ser verificadas em um software para que ele seja considerado um "software de qualidade". São seis grandes grupos de características: funcionalidade, confiabilidade, usabilidade, eficiência, manutenibilidade e portabilidade. Em 2001 foi revisada e subdividida em quatro: 9126-1: modelo de qualidade para o produto software; 9126-2: métricas externas para a avaliação do modelo de qualidade; 9126-3: métricas internas para a avaliação do modelo de qualidade; 9126-4: como aplicar as métricas ao modelo de qualidade.

Outras normas ISO podem ser aplicadas no desenvolvimento de software: ISO 12207 (ISO, 1995): descreve a arquitetura e os processos do ciclo de vida do software; ISO 15504 (ISO, 2004a), chamada SPICE - Software Process Improvement and Capability dEtermination: apresenta uma padrão para a avaliação do processo de software, visando determinar a capacitação de uma organização; ISO 19759 (ISO, 2005), chamada SWEBoK - Guide to the Software Engineering Body of Knowledge: corresponde ao resultado de um projeto iniciado em 1998 e concluído em 2004, contando com a participação de diversas instituições, entre as quais IEEE Computer, Boing, Rational Software e SAP, especialistas (como Pressman e Sommerville, referências na área de engenharia de software) e com mais de uma centena de revisores, tendo como principal propósito congregar um conjunto de conhecimentos aceitos internacionalmente como pertinentes à atividade de engenharia de software.

Em 1986, o SEI iniciou o desenvolvimento de um modelo (*framework*) de maturidade de processo que pudesse auxiliar as organizações a evoluírem seus processos de software; em 1987 o SEI publicou uma descrição do modelo de maturidade de processo e um questionário de maturidade (HUMPHREY, 1995); em 1991 o SEI gerou o Modelo de Capacidade e Maturidade para Software (CMM) (PAULK et. al., 1993). O CMM é baseado no conhecimento adquirido e acumulado pelos processos de desenvolvimento de software e pelo extensivo feedback obtido tanto das indústrias quanto do governo (CMU; SEI, 1995). Em 2000 o CMM foi atualizado para

o CMMI versão 1.1 (Integração), que passa a focar quatro grandes áreas: engenharia de sistemas, engenharia de software, produção integrada e desenvolvimento do processo (CMU; SEI, 2003); procurou aproximar o CMM dos conceitos da norma ISO 15504, apresentada anteriormente. O CMMI é um modelo para melhoria de processo de software, não definindo como implementar, e sim orientando na aplicação dos conceitos de gerência de projetos e de melhoria de qualidade para desenvolvimento e manutenção de software. É organizado em cinco níveis (inicial, gerenciado, definido, gerenciado quantitativamente e otimizado), de acordo com a maturidade das organizações no tocante ao processo de software (CMU; SEI, 2002). Estes níveis definem uma escala para medir e classificar a maturidade do processo de software de uma dada organização e para avaliar a capacidade do processo de software; estes níveis também auxiliam as organizações a priorizar seus esforços nas áreas carentes. Está previsto para o segundo semestre de 2006 o lançamento do modelo CMMI versão 1.2.

O projeto mps-Br - melhoria de processo do software brasileiro - é uma iniciativa sob a coordenação da SOFTEX e envolve instituições de ensino, pesquisa e centros tecnológicos (Coppe/UFRJ, Universidade Católica de Brasília, Cesar, CenPRA); uma sociedade de economia mista, a Companhia de Informática do Paraná (Celepar) e Agentes SOFTEX (Riosoft e SOFTEX Campinas); iniciado em 2003, prevê um Modelo de Referência de Processos, baseado nos conceitos de maturidade e capacidade de processo, para avaliação e melhoria da qualidade e produtividade de produtos e serviços de software, visando uma maior utilização pela indústria nacional e uma capacitação para definição de ajustes e adaptações dos modelos internacionais para alinhamento com a realidade e decisões da política de software brasileira. Também prevê um Modelo de Negócio para melhoria de processo de software, com grande potencial de replicabilidade no Brasil e em outros países com características semelhantes no que se refere à indústria de software – a idéia é ter esses modelos para cada país, em especial da América Latina, onde os contatos para potenciais parcerias já estão em andamento.

O mps-Br é um projeto que visa promover a qualificação de um amplo número de empresas

compatível com os padrões de qualidade aceitas internacionalmente pela comunidade de software, a custos acessíveis para a maioria das empresas brasileiras, sendo adequado ao perfil e cultura das empresas de software do país. Tem como base os modelos ISO 12207 e 15504 e o modelo SEI-CMMI, apresentados anteriormente. O modelo proposto prevê sete níveis de maturidade: G - parcialmente gerenciado; F – gerenciado; E - parcialmente definido; D - largamente definido; C – definido; B - gerenciado quantitativamente; A - em otimização. A adoção do modelo mps-Br pelas empresas começou em 2005; MCT (2006) apresenta as avaliações realizadas até o momento, com cinco empresas, uma no nível G, três no nível F e uma no nível E.

5 ANÁLISE COMPARATIVA

Boa parte da análise comparativa foi realizada ao longo da descrição de cada modelo (de qualidade genérico, específico para software ou de gerenciamento de projetos), abordando os pontos positivos e negativos. Este capítulo procura sumarizar as constatações efetuadas, ressaltando os principais aspectos envolvidos.

No tocante aos modelos de gerenciamento de projetos, existem várias metodologias e *frameworks*. Duncan (1999), membro do PMI, critica a norma ISO 10006, considerando um grande risco as empresas a adotarem como padrão de gerenciamento de projetos, entendendo que omite diversos aspectos relevantes presente no PMBoK. Martins (2005) aborda de forma interessante o uso do PMBoK em desenvolvimento de software orientado a objetos utilizando o padrão UML.

Em relação aos padrões de qualidade, vários autores consideram que o nível de exigência do CMM/CMMI é bem mais elevado que os preceitos da ISO 9000; Paulk (1994) considera que a ISO 9000 pode corresponder a uma qualificação CMM nível 3, podendo existir inclusive casos de empresas que ainda estejam no CMM nível 1; enquanto a ISO 9000 prevê um sistema de certificação de terceira parte, com órgãos credenciados para realizar as auditorias, em intervalos regulares de tempo, nos sistemas da qualidade e garantir que estes estão funcionando adequadamente, o CMM/CMMI é avaliado por auditores credenciados pelo SEI. O grau de

abstração do modelo ISO é mais alto que o CMM/CMMI e se aplica a qualquer tipo de empresa enquanto o modelo CMM/CMMI é muito mais profundo no tocante aos quesitos de desenvolvimento de software e, portanto, mais exigente. Vale ressaltar que os modelos ISO e CMMI são compatíveis, sendo perfeitamente viável uma empresa que possui ISO 9000 implantar o modelo CMMI, assim como é possível uma empresa com ISO 9000 estar ainda no nível 1 do CMMI uma vez que são modelos com diferentes requisitos.

O PMBoK faz referência ao preceitos de qualidade previstos pelas séries ISO 9000 e 10000. Uma distinção se faz necessária no tocante à qualidade: ela é diferente de graduação, que pode ser entendida "por uma categorização ou classificação para entidades que tenham a mesma funcionalidade, mas diferentes requisitos de qualidade" (PMI, 2004). Esta distinção é importante uma vez que um projeto com baixa qualidade é sempre um problema, enquanto um projeto com baixa graduação nem sempre o é - por exemplo, um software com pequeno número de funcionalidade pode ser considerado de baixa graduação e isto não significa necessariamente um problema (ao contrário, pode ter sido um requisito), enquanto um software com baixa qualidade (quer seja por problemas de performance, erros, comprometimento quanto a sua entrega) certamente indica problemas. Vale ressaltar que o nível de graduação do software deverá ser determinado em conjunto, com a participação de todos os envolvidos (equipe do projeto, clientes, fornecedores). O PMBoK aponta outro aspecto fundamental em relação ao gerenciamento da qualidade, que é totalmente aplicável ao software: a diferenciação de prevenção e inspeção, que aparentemente são semelhantes (não deixar passar erros), porém diferem drasticamente em suas formas: enquanto a prevenção consiste em um processo contínuo de melhoria, a inspeção é muito mais branda e consiste nos famosos testes apenas ao final do desenvolvimento a fim de evitar que o usuário final detecte o erro antes dos profissionais de informática, perpetuando o erro.

A área de escopo do projeto de desenvolvimento de software é normalmente negligenciada no tocante ao seu gerenciamento e controle. O levantamento e a definição do escopo normalmente são feitos utilizando técnicas de levantamento de dados junto aos usuários e, após isto, utilizando

modelos estruturados (DFD), orientados a objeto (Casos de Uso), texto em linguagem natural ou outra forma de documentação, porém nem sempre são feitas as validações junto aos usuários e normalmente não existe um processo formal de atualização, autorizações de mudanças e controle. As disciplinas de gerenciamento contribuem com a formalização destes processos, instituindo procedimentos e responsabilidades pelas ocorrências. Diversos autores incluem estes processos como dos mais relevantes para o sucesso do desenvolvimento de sistemas.

Tradicionalmente o termo gerenciamento é associado à dimensão de prazo. O estabelecimento dos prazos é realizado após um efetivo entendimento do escopo, seu detalhamento (utilizando técnicas de decomposição e WBS) e identificação dos produtos a serem gerados (*deliverables*); nesse momento as atividades são identificadas, as estimativas de prazo são realizadas e a interdependência entre elas é definida, gerando o cronograma do projeto. Tão importante quanto a elaboração do plano de projeto (cronograma) é o acompanhamento de sua execução a fim de detectar (e evitar) desvios e, caso não seja possível, ativar ações corretivas para que o projeto volte ao planejado. O projeto pode sofrer modificações em função de um maior detalhamento dos aspectos envolvidos (em especial escopo), sendo necessário replanejar, verificar os impactos nas diversas áreas de gerenciamento de projeto e implementar as medidas necessárias.

O aspecto financeiro é um dos mais importantes na contratação de um desenvolvimento de software: quanto irá custar e qual o retorno (financeiro) oriundo de sua implantação. Ainda que todos os cuidados referentes a uma boa orçamentação inicial, baseada nos requisitos (escopo) e recursos necessários (pessoas, materiais), o risco de desvios de custo é grande, sendo necessário um processo de monitoração. As técnicas de acompanhamento de custos, baseadas em indicadores de desempenho atrelados ao trabalho produzido comparados com o custo esperado (o chamado *earned value*), são ferramentas poderosas de detecção de desvios de custos, antecipando problemas. Para exemplificar, se o custo real em uma dada data está menor que o previsto, não necessariamente significa que algo bom (economia) está ocorrendo - ao contrário, pode indicar que os esforços necessários não estão de acordo com o planejado e que um atraso

possa acontecer. Com o uso das técnicas mencionadas estas e outras conclusões podem ser tiradas e medidas preventivas/corretivas podem ser tomadas.

Um processo formal para identificação, quantificação, acompanhamento e controle de riscos é vital para que o projeto tenha sucesso. Alguns autores, como Ambler (1998), argumentam que técnicas de orientação a objetos aplicadas ao desenvolvimento de software podem fazer com que alguns riscos de origem técnica sejam eliminados através da construção e reutilização de componentes de software. Os projetos de desenvolvimento de software estão intimamente ligados a riscos, motivados sobretudo pela natureza do software, a dificuldade em encará-lo como produto, a pressão normalmente feita aos desenvolvedores de software, a falta de foco nos aspectos gerenciais entre outros fatores; isso faz com que o desenvolvimento de um software seja, por si só, um fator de risco a ser considerado. Somente com o estabelecimento do processo formal de gerenciamento de riscos esses problemas podem ser solucionados. É preciso conscientizar sobre a necessidade de se identificar os riscos antes do projeto ser efetivamente iniciado, definir a melhor estratégia para abordá-lo - muitas vezes o risco pode ser compartilhado com os clientes, trazendo benefícios a todos os envolvidos. Vale notar que os riscos podem surgir (e desaparecer) ao longo do projeto; é necessário acompanhamento e atualização constante dos riscos identificados. Outro fator decorrente da análise de riscos consiste em detectar oportunidades. Por vezes, ao iniciar o desenvolvimento de um software, percebe-se uma oportunidade, quer seja de novos negócios com o cliente, quer seja de ampliação do escopo, quer seja de expansão dos clientes atendidos pelo software a ser desenvolvido. É importante a institucionalização de processo formal para tratar estas oportunidades, ainda que a decisão seja por não persegui-las. Várias empresas tiveram sucesso a partir dessa abordagem e outras não conseguiram se manter em função de terem avaliado de forma inadequada.

Outra contribuição dos modelos estudados se refere aos aspectos humanos, a necessidade de capacitação dos profissionais em aspectos técnicos e gerenciais, a necessidade do ser humano em ser mantido informado a respeito da real situação do projeto (tópico gerenciamento de

comunicações do PMBoK, com procedimentos formais para garantir que as comunicações sejam completas, efetivas e claras, tanto dos pontos positivos quanto dos problemas), a habilidade de se trabalhar em times, de resolução de conflitos, inevitável em projetos de software, onde algumas necessidades e soluções para os problemas são antagônicas. Outro aspecto importante está relacionado ao perfil dos profissionais na área de computação: a grande maioria tem vocação técnica, se preocupando apenas com a construção técnica do produto, não reservando tempo para as atividades gerenciais. Existe consenso que a conscientização dos profissionais sobre estas atividades são de vital importância para o sucesso dos projetos, porém a implantação destas recomendações é algo que caminha lentamente, uma vez que aspectos culturais estão amplamente disseminados. Nos cursos de formação de profissionais que lidam com software nota-se que a maioria dos alunos tem necessidade e interesse especial pelos aspectos técnicos, pelas linguagens de programação e ferramentas de desenvolvimento, porém quando o assunto é ligado aos aspectos gerenciais envolvidos com projetos existe grande dificuldade. A cultura da "mão na massa" e "ligar o micro e começar a desenvolver o sistema" é muito forte, sendo uma barreira importante a ser ultrapassada na busca do profissionalismo e da qualidade.

Outro aspecto considerado pelos modelos apresentados é a possibilidade de contratação de serviços e produtos a partir de fornecedores externos à organização, prática cada vez mais comum no desenvolvimento de software, que introduz diversas preocupações, desde os aspectos de seleção de fornecedores, confiabilidade do fornecedor, regime de contratação, cláusulas contratuais (jurídicas e técnicas), administração do contrato, critérios de inspeção dos produtos e relacionamento entre o fornecedor e o time de projeto; com isso, a habilidade de negociação e trabalho em times deve ser fortalecida e as preocupações gerenciais redobradas. O porte atual dos projetos de software, associado às especialidades e especificidades existentes, faz com que dificilmente uma organização possua (e que estejam disponíveis) todos os recursos necessários; sendo o processo de contratação de pessoal especializado normalmente demorado, via de regra não atendendo às necessidades do projeto, agravado pelo custo elevado (uma vez que envolve

pessoal especializado valorizado pelo mercado) e pelo fato do projeto ter uma duração finita, findo a qual cessa a necessidade do recurso e passa a não ser interessante para a empresa a manutenção do mesmo - nestes casos é indicada a contratação de serviços junto a fornecedores. A relação pode ser do tipo parceria (cooperação mútua entre os envolvidos), específica para um trabalho, alocação de mão de obra especializada para exercer alguma atividade ou ainda projeto global (chamado de *turn key*), onde um dado fornecedor se responsabiliza por toda a obra/desenvolvimento, subcontratando outros fornecedores, caso necessário. Com a contratação de serviços algumas preocupações adicionais devem ser levadas em consideração, como, por exemplo, porte do fornecedor, regime de contratação dos recursos por parte do fornecedor (isto tem grande impacto na legislação trabalhista brasileira, uma vez que pequenas empresas, via de regra, não contratam como empregados seus funcionários, contratando via empresa individual - a chamada "quarteirização" - com riscos trabalhistas tanto para a empresa que está contratando estes indivíduos quanto para a empresa que originalmente contratou o serviço). Cláusulas contratuais adequadamente revisadas nos seus aspectos jurídicos, bem como a exigência das documentações referentes ao fornecedor, são de vital importância para a contratação de serviços terceirizados; não raro ocorrerem problemas trabalhistas em função de um dado projeto quando este já se encerrou, gerando ônus para os envolvidos.

A comunicação é vital em um projeto, nem sempre feita de forma satisfatória. Manter os envolvidos em um projeto com informações suficientes a respeito do andamento, com freqüência adequada e de forma atualizada, é um desafio, agravado nas etapas mais técnicas do processo (codificação, por exemplo) onde existem problemas para medir efetivamente a situação do projeto. Vale ressaltar que as necessidades de informações são diferentes entre os envolvidos: enquanto o time do projeto necessita de informações detalhadas, o gerente do projeto provavelmente necessitará de outro tipo de informação (normalmente tudo aquilo que está diferente do planejado, a fim de ser colocado em prática o gerenciamento por exceção, ou seja, se concentrar nos pontos onde pode ou irão acontecer os problemas), o cliente necessitará de outro

nível de detalhe (mais voltada ao andamento do projeto). No início do desenvolvimento deve ser feita esta distinção e mecanismos para a geração destes dados devem ser providos e considerados como parte integrante do próprio desenvolvimento.

Após a análise individual das áreas de gerenciamento, com seus impactos pormenorizados, cabe uma avaliação conjunta (integrada) de todas estas áreas, tomando por base os requisitos individuais de cada uma, até o procedimento global de controle de modificações, seja de escopo, prazo ou custo, e seu impacto no projeto. Deve ser constantemente revisada a fim de garantir um efetivo acompanhamento do projeto. O plano integrado do projeto e o controle unificado de modificações são relevantes e primordiais para a conclusão com sucesso do projeto.

6 CONCLUSÕES E PERSPECTIVAS FUTURAS

Pode-se concluir que diversos aspectos podem ser introduzidos às práticas de gerenciamento de processos de desenvolvimento de software. O PMBoK, através da visualização dos processos de gerenciamento de projetos agrupados em áreas de conhecimento e grupos, possibilita que o gerenciamento de projetos seja melhor definido e documentado, propiciando uma avaliação precisa dos impactos em todas as áreas e etapas do projeto, maximizando a possibilidade de sucesso deste projeto.

Vale ressaltar que o reconhecimento das áreas de gerenciamento como grupos de processos, com preocupações distintas, e de que existe um grupo geral que analisa os impactos nas demais áreas de gerenciamento é uma grande contribuição à área de informática. Analisando separadamente estas áreas tem-se, num primeiro momento, o impacto do andamento das atividades em cada segmento para, num segundo momento, verificar o impacto global. O que normalmente ocorre no desenvolvimento de software é a total falta de controle, quanto mais separado por área.

Outra valiosa contribuição é a definição formal das várias fases pelas quais um projeto passa, reforçando a importância da fase de planejamento, onde os diversos aspectos relativos a cada área de gerenciamento serão levantados e planejados. Outra característica importante é a necessidade do início formal do processo, envolvendo todos (clientes, equipe, fornecedores) em busca do

comprometimento.

Também importante é a necessidade de um encerramento formal, com a resolução das pendências; sem isso, o projeto não pode ser considerado como concluído - processo não muito difundido na área de software, com vários produtos gerados e concluídos sem que todos os aspectos tenham sido cuidadosamente verificados, causando desgaste, má qualidade, custos e descrédito para os profissionais.

Os processos de controle são essenciais para garantir que os rumos planejados sejam alcançados e que medidas corretivas eficazes sejam adotadas, quando necessárias. Estes processos introduzem a necessidade de revisões periódicas nas bases do projeto (custo, prazo, escopo etc.) a fim de refletir as modificações, além de verificar o impacto global, uma vez que as modificações ocorrem normalmente em mais de uma área.

Com a introdução destes conceitos nas organizações o gerenciamento de processos de desenvolvimento de software será otimizado. A implantação de uma base histórica contendo dados dos projetos já desenvolvidos, lições aprendidas, modificações ocorridas (com respectivas análises e planos de ação), técnicas de engenharia de software utilizadas (incluindo técnicas de estimativa de software, como pontos de função/funcionalidade e linhas de código), premissas, restrições, características, comparações entre planejado e efetivamente realizado, auxiliará e garantirá a evolução dos processos e dos profissionais envolvidos com o desenvolvimento do software.

Também é recomendável o intercâmbio de experiências com profissionais e empresas que utilizam técnicas para melhorar a efetividade de seus processos de desenvolvimento de software; várias empresas no Brasil estão empenhadas em adotar modelos como CMMI e, mais recentemente, mps-Br e com isso a necessidade de aprofundamento nos requisitos, além da demonstração prática de que processos claros e efetivos de gerenciamento de projetos estão sendo adotados, passa a ser vital. No mundo dos negócios isto passa a ser questão de sobrevivência para todos os envolvidos.

A fim de continuar a abordagem aqui utilizada vários aspectos poderiam vir a serem trabalhados. Algumas sugestões de trabalhos futuros: aprofundar a análise, através de pesquisas, em que medida o mercado brasileiro (e mundial) vem adotando as práticas de gerenciamento de projetos para o desenvolvimento de software; acompanhar a adoção do CMMI e do mps-Br pelas empresas brasileiras e mundiais, em que proporção isto está sendo feito e quais as expectativas futuras; verificar possível convergência entre os modelos do PMI e da ISO; publicação efetiva de um adendo ao PMBoK relativo às peculiaridades com o gerenciamento de processos de desenvolvimento de software.

7 REFERÊNCIAS BIBLIOGRÁFICAS

AMBLER, Scott W. (1998) Maturing the OO Software Process - reducing risks and aumenting quality. Object Magazine, Fevereiro 1998.

AMBLER, Scott W. (2004) Modelagem Ágil: Práticas eficazes para a Programação eXtrema e o Processo Unificado. Porto Alegre: Bookman, 2004.

CMU; SEI. (1995) The Capability maturity model: Guidelines for improving the software process (SEI Series in Software Engineering). Carnegie Mellon University; Software Engineering Institute. Addison Wesley Longman, 1995.

CMU; SEI. (2002) CMMISM for Systems Engineering/Software Engineering/Integrated Product and Process Development/Supplier Sourcing, Version 1.1, Staged Representation (CMMI-SE/SW/IPPD/SS, V1.1, Staged). Carnegie Mellon University; Software Engineering Institute. Março 2002. Disponível em <http://www.sei.cmu.edu/publications/documents/02.reports/02tr012.html>. Acesso em: 02 mar. 2006.

CMU; SEI. (2003) CMMI: Guidelines for Process Integration and Product Improvement (SEI Series in Software Engineering). Carnegie Mellon University; Software Engineering Institute. Addison Wesley Longman, 2003.

DUNCAN, William R. (1999) ISO 10006: Risky Businness. 1999. Disponível em <http://www.pmpartners.com/resources/iso10006.html>. Acesso em: 26 mar. 2006.

HUMPHREY, Watts S. (1995) A Discipline for Software Engineering (SEI Series in Software Engineering). Addison Wesley Publishing Company, 1995.

ISO. (1995) ISO/IEC 12207 - Information technology - Software life cycle processes. Genebra, ISO, 1995.

ISO. (2000) ISO 9001:2000 - Quality management systems - Requirements. Genebra, ISO, 2000.

ISO. (2001) ISO/IEC 9126-1:2001 - Software engineering - Product quality - Part 1: Quality model; ISO/IEC 9126-2:2001 - Software engineering - Product quality -- Part 2: External metrics; ISO/IEC 9126-3:2001 - Software engineering - Product quality - Part 3: Internal metrics; ISO/IEC 9126-4:2001 - Software engineering - Product quality - Part 4: Quality in use metrics. Genebra, ISO, 2001.

ISO. (2003) ISO 10006:2003 - Quality management systems - Guidelines for quality management in projects. Genebra, ISO, 2003.

ISO. (2004a) ISO/IEC 15504-1:2004 - Information technology - Process assessment - Part 1: Concepts and vocabulary; ISO/IEC 15504-2:2003/Cor 1:2004 - Part 2: Performing an assessment; ISO/IEC 15504-3:2004 - Part 3: Guidance on performing an assessment; ISO/IEC 15504-4:2004 - Part 4: Guidance on use for process improvement and process capability determination. Genebra, ISO, 2004.

ISO. (2004b) ISO/IEC 90003:2004 Software engineering - Guidelines for the application of ISO 9001:2000 to computer software. Genebra, ISO, 2004.

ISO. (2005) ISO/IEC TR 19759:2005 - Software Engineering - Guide to the Software Engineering Body of Knowledge (SWEBOK). Genebra, ISO, 2005.

MARTINS, José Carlos Cordeiro. (2005) Gerenciando projetos de desenvolvimento de software com PMI, RUP e UML. 2a. Edição. Rio de Janeiro: Brasport, 2005.

MCT. (2006) Avaliações MA-MPS. Ministério da Ciência e Tecnologia. Disponível em <http://www.softex.br/cgi/cgilua.exe/sys/start.htm?sid=222>. Acesso em: 04 mar. 2006.

MORAES, Renato de Oliveira. (2004) Condicionantes de desempenho dos projetos de software e a influência da maturidade em gestão de projetos. 2004. 138 f. Tese (Doutorado em Administração de Empresas) – Faculdade de Economia, Administração e Contabilidade, Universidade de São Paulo, São Paulo.

MOURA, Arnaldo Vieira. (2001) Especificação em Z: uma introdução. Campinas, São Paulo: Editora da UNICAMP, 2001.

PAULK, Mark C. et. al. (1993) Capability Maturity Model for Software, Version 1.1 Technical Report CMU/SEI-93-TR-024, 1993.

PAULK, Mark C. et. al. (1994) A Comparison of ISO 9001 and the Capability Maturity Model for Software. Technical Report CMU/SEI-94-TR-12, 1994.

PETERS, James F.; PEDRYCZ, Witold. (2001) Engenharia de Software. Rio de Janeiro: Campus, 2001.

PFLEEGER, Shari Lawrence. (2004) Engenharia de Software – Teoria e Prática. São Paulo: Prentice Hall, 2004.

PMI. (2004) PMBOK Guide (Project Management Body of Knowledge). 3rd. Edition. Four Campus Boulevard, Newton Square, PA: Project Management Institute, 2004.

PRESSMAN, Roger S. (2006) Engenharia de Software. 6a. Edição. São Paulo: McGraw-Hill, 2006.

REZENDE, Denis Alcides. (2005) Engenharia de software e sistemas de informação. 3a. Edição. Rio de Janeiro: Brasport, 2005.

SOMMERVILLE, Ian. (2003) Engenharia de Software. 6a. Edição. São Paulo: Addison-Wesley, 2003.