PARADIGMAS DA ENGENHARIA DE SOFTWARE COMO FATORES CRÍTICOS DE SUCESSO PARA A GESTÃO DE PROJETOS DE SOFTWARE NO CONTEXTO BRASILEIRO

MARCELO NOGUEIRA (UNIP)
marcelo@noginfo.com.br
JAIR MINORO ABE (UNIP)
jairabe@uol.com.br

Resumo: *OS PROCESSOS DE QUALIDADE DE SOFTWARE PASSARAM A FAZER PARTE DO PROCESSO DE PRODUÇÃO DE SOFTWARE NAS EMPRESAS BRASILEIRAS NOS ÚLTIMOS ANOS. ESSA ADOÇÃO, NÃO VEIO PELA ADERÊNCIA E SIM PELA NECESSIDADE DE DESENVOLVER SOLUÇÕES EMPRESARIAIS QUEE ATENDESSEM AOS REAIS E COMPLEXOS MODELOS DE NEGÓCIOS PRESENTES NAS ORGANIZAÇÕES NO PAÍS. AS ADEQUAÇÕES VINDAS DE NORMAS E MODELOS DE QUALIDADE DIRECIONAM O RESULTADO E PROPICIA MELHORIA CONTÍNUA NOS PROCESSOS DE DESENVOLVIMENTO DE SOFTWARE, TORNANDO-O MENOS DISPENDIOSO. DENTRE ESTES OS PROCESSOS, DEZ SÃO CONSIDERADOS FATORES CRÍTICOS DE SUCESSO: ENGENHARIA DE REQUISITOS, GESTÃO DE CONFIGURAÇÃO, GESTÃO DE RISCOS, MODELAGEM VISUAL, METODOLOGIAS DE DESENVOLVIMENTO, NORMAS E MODELOS DE QUALIDADE, MÉTRICAS, CRONOGRAMAÇÃO, IMPLEMENTAÇÃO E TESTES. ELES PERMITEM AINDA QUE O PROFISSIONAL BRASILEIRO TENHA O SEU VALOR AGREGADO TANTO NA CRIATIVIDADE QUANTO NA DISCIPLINA PROFISSIONAL, MITIGANDO OS RISCOS DE FRACASSO E AUMENTANDO A PARTICIPAÇÃO EM PROJETOS ANTERIORMENTE ENCAMINHADOS PARA O EXTERIOR.*

Palavras-chaves: *QUALIDADE DE SOFTWARE, ENGENHARIA DE SOFTWARE, GESTÃO DE PROJETOS*

PARADIGMS OF SOFTWARE ENGINEERING AS CRITICAL SUCCESS FACTORS FOR THE MANAGEMENT OF SOFTWARE PROJECTS IN THE BRAZILIAN CONTEXT

Abstract: *PROCESSES OF SOFTWARE QUALITY BECAME PART OF THE PROCESS OF SOFTWARE PRODUCTION IN BRAZILIAN COMPANIES IN RECENT YEARS. THIS ADOPTION DID NOT COME BY COMPLIANCE BUT BY THE NEED TO DEVELOP BUSINESS SOLUTIONS THAT MET THE REAL AND COMPLEX BUSSINESS MODELS IN ORGANIZATIONS IN THE COUNTRY. THE ADEQUACY OF STANDARDS AND MODELS WELCOME QUALITY RESULTS AND PROVIDES DIRECT THE CONTINUOUS IMPROVEMENT IN THE PROCESSES OF SOFTWARE DEVELOPMENT, MAKING IT LESS EXPENSIVE. AMONG THESE CASES, TEN ARE CONSIDERED CRITICAL SUCCESS FACTORS: REQUIREMENTS ENGINEERING, CONFIGURATION MANAGEMENT, RISK MANAGEMENT, VISUAL MODELING, DEVELOPMENT METHODOLOGIES, STANDARDS AND QUALITY MODELS, METRICS, CRONOGRAMAÇÃO, IMPLEMENTATION AND TESTING. THEY ALSO ALLOW THE BRAZILIAN PROFESSIONAL HAS ITS VALUE BOTH IN CREATIVITY AND IN PROFESSIONAL DISCIPLINE, MITIGATING THE RISK OF FAILURE AND INCREASING PARTICIPATION IN PROJECTS PREVIOUSLY TRANSMITTED EXTERNALLY.*

Keyword: *SOFTWARE QUALITY, SOFTWARE ENGINEERING, PROJECT MANAGEMENT*

1. Introdução

Num ambiente competitivo e de mudança cada vez mais complexo, a gestão adequada da Informação assume uma importância decisiva no processo de tomada de decisão nas organizações.

Tratando-se de uma temática simultaneamente abrangente e especializada, a adoção dos processos de Engenharia de Software como linha base da Gestão da Informação, possibilita a consolidação dos conhecimentos no desenvolvimento de software, bem como a preparação dos indivíduos para encarar com confiança os novos desafios inerentes à profissão. O mundo dos negócios de software requer ao profissional um reforço constante de competências, mantendo-se atualizado em relação ao potencial dos sistemas de informação e das novas tecnologias numa perspectiva empresarial e competitiva globalmente.

Considerado o melhor emprego na América do Norte, o de Engenheiro de Software (CNN, 2006), percebe a relevância da sua função ser aumentada proporcionalmente às necessidades das organizações na implementação de sistemas em ambientes cada vez mais hostis e complexos.

A partir do conhecimento adquirido de normas de Qualidade de Software, o profissional será elemento multiplicador de soluções, contribuindo e agregando valor aos sistemas novos e aos já existentes, com aplicação de metodologias e técnicas adequadas para a sua implementação, pois tais sistemas deverão ser capazes de propiciar com sucesso as informações relevantes aos negócios aplicáveis, trazendo ás organizações, vantagens competitivas.

No entanto o profissional brasileiro, considerado o mais empreendedor do mundo (BOTELHO, 2006), encontra várias dificuldades para criar e gerir seu próprio negócio devido à inexistência de uma política econômica que propicie vantagens para exportação de software. As pequenas empresas nacionais representam 65,1% do total de empresas que produzem software no Brasil (MCT, 2002).

Atingir um alto nível de qualidade de produto ou serviço é o objetivo da maioria das organizações. Atualmente não é mais aceitável entregar produtos com baixa qualidade e reparar os problemas e as deficiências depois que os produtos foram entregues ao cliente (SOMMERVILLE, 2003).

Os números que representam o contexto brasileiro sobre a adoção dos processos que visam obter qualidade de software, fazem parte do escopo deste artigo.

1. Relevância

No estudo da Engenharia de Software, o autor Roger S. Pressman (PRESSMAN, 2006), demonstra preocupação com a “Crise do Software” que atualmente ele intitula como “Aflição Crônica”, chegando a determinar números expressivos sobre a não finalização de projetos de sistemas começados e não terminados.

Para generalizar o termo, ocorre quando o software não satisfaz seus envolvidos, sejam clientes e/ou usuários, desenvolvedores ou empresa (REZENDE, 2005).

A expressão "Crise do Software", que começou a ser utilizada na década de 60, tem historicamente aludido a um conjunto de problemas recorrentemente enfrentados no processo de desenvolvimento (Construção, implantação e manutenção) de software (MAFFEO, 1992).

Esses problemas não se referem apenas a programas que não funcionam. Na verdade, a chamada "Crise do Software" abrange todos os problemas relacionados a (REZENDE, 2005):

- Como sistemas computacionais são construídos;
- Como sistemas computacionais são implantados, referindo-se aqui ao processo de substituir sistemas antigos, desativando sistemas correntemente em operação, ou ao processo de instalar um sistema inteiramente novo;
- Como é provida a manutenção da quantidade crescente de software construído, associado a sistemas computacionais cada vez mais complexos;
- Como fazer face à crescente demanda para construção de software, visando satisfazer ao conjunto enormemente variado de anseios por informatização, atualmente detectado na sociedade moderna;
- Como administrar as questões comportamentais, envolvendo os clientes e/ou usuários e a política, cultura e filosofia empresarial.

Apesar da enorme variedade de problemas que caracterizam a crise do software, engenheiros de software e gerentes de projetos para desenvolvimento de sistemas computacionais tendem a concentrar suas preocupações no seguinte aspecto: "A enorme imprecisão das estimativas de cronogramas e de custos de desenvolvimento" (Tabelas 1, 2, 3, 4) (LEE, 2002).

TABELA 1 - Custos em projeto de software por fase de desenvolvimento. Fonte: LEE (2002).

Custos em projeto de software por Fase de Desenvolvimento	
Etapa de Trabalho	**%**
Análise de Requisitos	3
Desenho	8
Programação	7
Testes	15
Manutenção	67

TABELA 2 - Custos para correção de erros de software. Fonte: LEE (2002).

Custos para Correção de Erros de Software				
Fase de desenvolvimento do software	***% de Desvios*** ($)	***Erros Introduzidos*** (%)	***Erros Encontrados*** (%)	***Custo Relativo para*** **Correção**
Análise de Requisitos	5	55	18	1,0
Desenho	25	30	10	1,0 - 1,5
Teste do Código e da Unidade	10			
Teste de Integração	50	10	50	1,0 - 5,0
Validação e Documentação	10			
Manutenção Operacional		5	22	10 – 100

TABELA 3 - Excedente de Custo. Fonte: LEE (2002).

Excedentes de Custo

% Excedente de Custo	% de Respostas
<20%	15,5%
21% - 50%	31,5%
51% - 100%	29,6%
101% - 200%	10,2%
201% - 400%	8,8%
>400%	4,4%

TABELA 4 - Excedente de Prazo. Fonte: LEE (2002).

Excedente de Prazo	
% Excedente de Prazo	**% de Respostas**
<20%	13,9%
21% - 50%	18,3%
51% - 100%	20,0%
101% - 200%	35,5%
201% - 400%	11,2%
>400%	1,1%

Segundo o Standish Group, entidade americana, através de um estudo chamado "Chaos Report", para projetos na área de Tecnologia da informação, obteve as seguintes conclusões (STANDISH, 1994):

- Apenas 16% terminam no prazo e dentro do orçamento previsto;
- 94% têm pelo menos um reinício;
- Há um aumento de 188% no seu custo e 222% no cronograma;
- Apenas 61% são concluídos com os objetivos originais pré-estabelecidos.

Em 2001, foi divulgado novo relatório que apresentava sensível melhoria (TELES, 2005):

- 72% dos projetos:
- Consomem mais recursos que o orçado;
- Consomem mais tempo que o estimado;
- Não entregam o que foi combinado;
- Todas as alternativas acima em conjunto.
- Excedem o Prazo = 63%;
- Excedem os Custos = 45%;
- Cancelados antes da implantação = 23%;
- Universo Pesquisado = 280.000 projetos de software.

O relatório de 2001, também apresentou uma avaliação quanto as funcionalidades do software:

- 45% nunca são usadas;
- 19% raramente são usadas;
- 16% às vezes;
- 13% com freqüência.
- 07% sempre são usadas.

Num mundo cada vez mais de recursos financeiros escassos, como é possível aceitar tal desperdício de tempo e dinheiro. Pressman também aponta para o possível problema causador de tal absurdo: “A falta de adoção de métodos, ferramentas e procedimentos no

desenvolvimento de software e a difícil relação de entendimento entre o usuário com o desenvolvedor".

A parte mais difícil do desenvolvimento de software é decidir precisamente o que será desenvolvido. Nenhuma outra parte do trabalho é tão difícil quanto estabelecer (definir) os detalhes técnicos necessários incluindo todas as interfaces para pessoas, máquinas e para outros sistemas de software. Nenhuma outra parte do trabalho é tão possível de ocasionar erros no sistema como essa. Nenhuma outra parte é tão difícil de ser posteriormente consertada (BROOKS, 1986).

Apesar da "Crise de Software" ter sido detectada desde os anos 60, até hoje ainda enfrentamos seus efeitos. Quando o produto não atende as expectativas dos clientes/usuários e possuem falhas de concepção da real necessidade da empresa, excedem prazos e custos, eles se enquadram na perspectiva da "Crise do Software".

Em 2002 o Ministério da Ciência e Tecnologia publicou um estudo chamado Qualidade e Produtividade no setor de Software Brasileiro, onde apresentou o contexto da qualidade na cultura brasileira de desenvolvimento de software. Além de dados valiosos sobre utilização dos processos de engenharia de software e normas de qualidade, informou que a participação das micro, pequenas e médias empresas de software correspondem 65,1% do total do mercado(MCT, 2002). Essa particularidade exige uma postura diferenciada para desenvolver o mercado de software brasileiro, pois a dimensão deste mercado ainda em franca expansão, demonstra que no contexto nacional o direcionamento da qualidade de software para este segmento deve ser intensificado.

Atualmente com a visão global permitindo a participação nas exportações de software para outros países, cada vez mais a qualidade no processo de desenvolvimento e do produto de software ganham maior observação e adoção das melhores práticas e soluções tecnológicas.

3. Contexto brasileiro da qualidade de software

Para muitos engenheiros de software, a qualidade do processo de software é tão importante quanto à qualidade do produto. Assim na década de 90 houve uma grande preocupação com a modelagem e melhorias no processo de software. Abordagens importantes como as normas ISO 9000 e a ISO / IEC 12207, o modelo CMM (Capability Maturity Model) e o SPICE (Software Process Improvement and Capability dEtermination) sugerem que melhorando o processo de software, podemos melhorar a qualidade dos produtos (MACHADO, 2001).

A qualidade é conseqüência dos processos, das pessoas e da tecnologia. A relação entre e qualidade do produto e cada um desses fatores é complexa. Por isso, é muito mais difícil controlar o grau de qualidade do produto do que controlar os requisitos (PÁDUA, 2003).

Prevê-se que na primeira década dos anos 2000, após ajustarem seus processos para a produção de software de qualidade dentro de prazos e orçamentos confiáveis, as organizações serão pressionadas por seus concorrentes a reduzir substancialmente os prazos para a entrega de produtos.Organizações que sejam capazes de integrar, harmonizar e acelerar seus processos de desenvolvimento e manutenção de software terão a primazia do mercado (MACHADO, 2001).

Segundo o Ministério da Ciência e Tecnologia (MCT, 2002), ainda que divulgadas na década de 90, o conhecimento e utilização das normas e modelos para qualidade de software,

estão distantes de tornar-se realidade nas empresas desenvolvedoras de software, conforme tabelas 5, 6, 7, 8 e 9.

TABELA 5 - Adoção da Norma ISO/IEC 12207. Fonte: MCT (2002).

Norma ISO/IEC 12.207	Total (%)
Conhece e usa sistematicamente	3,9
Conhece e começa a usar	8,3
Conhece, mas não usa	55,1
Não conhece	32,7

TABELA 6 - Adoção da Norma ISO 9000. Fonte: MCT (2002).

Norma ISO 9000-3	Total (%)
Conhece e usa sistematicamente	19,4
Conhece e começa a usar	14,8
Conhece, mas não usa	52,4
Não conhece	13,4

TABELA 7 - Adoção do Modelo CMMI. Fonte: MCT (2002).

Modelo CMMI	Total (%)
Conhece e usa sistematicamente	3,9
Conhece e começa a usar	17,1
Conhece, mas não usa	53,7
Não conhece	25,3

TABELA 8 - Adoção do Projeto SPICE. Fonte: MCT (2002).

Modelo SPICE / Norma ISO/IEC 15.504	Total (%)
Conhece e usa sistematicamente	1,0
Conhece e começa a usar	3,2
Conhece, mas não usa	56,7
Não conhece	39,1

TABELA 9 – Adoção da norma ISO/IEC 9.126 – Avaliação do Produto de Software.

Norma ISO/IEC 9.126 = NBR 13.596	Total (%)
Conhece e usa sistematicamente	3,9
Conhece e começa a usar	7,5
Conhece, mas não usa	54,4
Não conhece	34,2

Diante destes números, é possível concluir que a baixa adoção das normas ou modelos de qualidade no Brasil, propicia qualidade contestável e inclui no cenário efetivo da "Crise do Software".

3.1 Há solução?

A solução pode não estar somente na adoção de uma única prática da Engenharia de Software. Para Brooks (Brooks, 1986) "não existe bala de prata", ou seja, uma solução única capaz de resolver a "Crise do Software". As soluções devem ser combinadas com a agregação

de vários processos e adaptados para cada contexto. Para tanto é necessário então considerar alguns paradigmas fundamentais como fatores críticos de sucesso na produção de software.

Segundo Rezende (REZENDE, 2005), pode-se resumir que a anticrise é a união e trabalho conjunto e harmonioso de três elementos: Empresa (Alta Administração), Cliente e/ou usuário e a unidade de informática (Desenvolvedores de soluções).

E na prática, cabe principalmente à unidade de informática aceitar este conceito e fazer o possível para a efetivação desta tese (Anticrise), utilizando-se de todos os recursos disponíveis para tal. A Unidade de informática é um dos principais agentes de mudança nas organizações, preocupando-se com o negócio empresarial, auxiliando efetivamente os gestores nos processos de tomada de decisão, tanto operacionais, como gerenciais e estratégicas.

4. Paradigmas da engenharia de software

Segundo Nogueira (NOGUEIRA, 2004), para que se obtenha qualidade no processo de produção de software, bem como no produto de software, é preciso adotar sistematicamente 10 paradigmas da Engenharia de Software como fatores críticos de sucesso.

4.1 Engenharia de Requisitos

"O processo de descobrir, analisar, documentar, e verificar as funções e restrições do sistema, é chamado de engenharia de requisitos" (SOMMERVILLE, 2003).

Engenharia de requisitos, uma subárea da engenharia de software, tem por objetivo tratar o processo de definição dos requisitos de software. Para isso estabelece um processo pelo qual o que deve ser feito é elicitado, modelado e analisado. Esse processo deve lidar com diferentes pontos de vista e usar uma combinação de métodos, ferramentas e pessoal. O produto desse processo é um modelo, do qual um documento chamado 'requisitos' é produzido. Esse processo é perene e acontece em um contexto previamente definido e que chamamos de 'Universo de informações' (LEITE, 2001).

A engenharia de requisitos fornece um mecanismo adequado para entender o que o cliente deseja, analisar as necessidades, avaliar a exeqüibilidade, negociar uma solução razoável, especificar a solução de maneira não-ambígua, validar a especificação e administrar os requisitos à medida que eles são transformados num sistema em operação. O processo da engenharia de requisitos pode ser descrito em seis passos distintos (PRESSMAN, 2006): Elicitação de requisitos, Análise e negociação de requisitos, Especificação de requisitos, Modelagem do sistema, Validação de requisitos e Gestão de requisitos.

É importante que os desenvolvedores de software reconheçam que não é possível desenvolver sistemas com qualidade, cumprir prazos e custos e atender às expectativas dos usuários sem ter um processo de engenharia de requisitos definido, compreendido e utilizado por toda a equipe.

4.2 Gestão de Configuração

"A arte de coordenar o desenvolvimento de software para minimizar a confusão é chamada de gerência de configuração. A gerência de configuração é a arte de identificar, organizar e controlar modificações de software que está sendo construído por uma equipe de programação. O objetivo é maximizar a produtividade pela minimização dos erros" (BABICH, 1986).

Segundo Sommerville (SOMMERVILLE, 2003) o gerenciamento de configuração (*configuration management – CM)* é o desenvolvimento e aplicação de padrões e procedimentos para gerenciar um produto de sistema em desenvolvimento. É necessário gerenciar os sistemas em desenvolvimento porque, à medida que eles se desenvolvem, são criadas muitas versões diferentes de software. Essas versões incorporam propostas de mudanças, correções de defeitos e adaptações para diferentes hardwares e sistemas operacionais. É possível que haja várias versões em desenvolvimento e em uso ao mesmo tempo. É necessário manter o controle das mudanças que foram implementadas e de como essas mudanças foram incluídas no software.

Todas as normas e modelos de qualidade para software têm por objetivo buscar organização e melhoria continua no processo de desenvolvimento de software. A implementação da gerência de configuração de software está totalmente ligada a essas normas e modelos. É de suma importância a adoção dessa prática para que o desenvolvedor tenha controle dos itens de software bem como as alterações ocorridas durante o desenvolvimento.

4.3 Gestão de Riscos

Gestão de Riscos é composta por atividades coordenadas para direcionar uma organização em relação ao risco. A gestão de riscos, geralmente inclui avaliação, tratamento, aceitação e comunicação de riscos (MCT, 2002).

A gestão de riscos envolve cinco atividades principais: Planejamento, controle, monitoração, direcionamento e recrutamento (PETERS, 2001).

A gestão de riscos é particularmente importante para projetos de software, devido às incertezas inerentes que a maioria dos projetos enfrenta (SOMMERVILLE, 2003).

Os riscos não permanecem constantes durante a execução de um projeto. Alguns desaparecem, outros novos surgem, e outros sofrem alterações de probabilidade e impacto, mudando, portanto a prioridade. Um relatório de acompanhamento do projeto juntamente com uma tabela atualizada para monitoração dos riscos. A tabela de estimativa deve ser repetida e atualizada para refletir as modificações ocorridas, até que os riscos sejam concretizados ou completamente eliminados (PADUA, 2003).

Cabe ao gerente de projetos, constante identificação, análise, planejamento, monitoramento, avaliação e gestão dos riscos em projetos de software que podem aparecer de acordo com cada empresa, diferindo ao seu porte, cultura organizacional, política e estratégias de negócios. Assim eleva-se a probabilidade de sucesso na implementação do software diante de que os fatores são críticos e envolvem áreas de conhecimentos amplas e complexas com a mitigação dos riscos.

4.4 Modelagem Visual

A modelagem visual é o uso de notações de design gráficas e textuais, semanticamente ricas, para capturar design de software. Uma notação, como a UML, permite que o nível de abstração seja aumentado, enquanto mantém sintaxe e semântica rígida. Dessa maneira, a comunicação na equipe de design melhora, à medida que o design é formado e revisado, permitindo ao leitor raciocinar sobre ele e fornecendo uma base não ambígua para a implementação (TONSIG, 2003).

A UML é uma linguagem de modelagem, totalmente orientada a objetos, que une as melhores práticas e metodologias da Engenharia de Software. É considerada a sintaxe geral para criar um modelo lógico de um sistema. Ela é utilizada para descrever pontos de um sistema e da forma como ele é percebido de várias visões durante a análise e sua arquitetura. É uma linguagem que visa capturar conhecimento e expressar esse conhecimento. Seu propósito é a modelagem de sistemas, documentar de maneira interativa e visual, proporcionar melhor compreensão e sinergia entre o analista e o cliente envolvido no processo de desenvolvimento.

Apesar da importância, não é unânime a utilização de modelagem nos projetos de software. A seguir na tabela 10, os números apresentam essa utilização (MCT, 2002).

TABELA 10 – Utilização de Métodos de Modelagem.

Métodos de Modelagem	Total (%)
Estruturado	**40,1**
Orientado a Objetos	**53,8**
Outros métodos	**6,1**

4.5 Metodologias de Desenvolvimento

Com a implementação de uma metodologia, a empresa produtora de software terá mais controle e gerenciamento através de uma metodologia focada aos processos fundamentais de desenvolvimento de software. Exemplos: RUP, XP e ALM (NOGUEIRA, 2004).

O RUP (Rational Unified Process) é um framework genérico para processos de desenvolvimento de software, criado pela empresa Rational Software Corporation, que está fortemente centrado na arquitetura, funcionalidade (caso de uso) e o desenvolvimento iterativo e incremental (inspirado no ciclo de vida espiral de Boehm), que aplica a UML, para o projeto e a documentação (TONSIG,2003).

Extreme Programming (XP) é uma metodologia de desenvolvimento de software, nascida nos Estados Unidos ao final da década de 90. Vem fazendo sucesso em diversos países, por ajudar a criar sistemas com qualidade, que são produzidos em menos tempo e de forma mais econômica que o habitual. Tais objetivos são alcançados através de um pequeno conjunto de valores e práticas, que diferem substancialmente da forma como se desenvolve software na grande maioria dos projetos (TELES, 2005).

ALM é uma metodologia da Borland que visa otimizar as organizações de software em cada uma das fases mais importantes do ciclo de vida da aplicação: definição, desenho, desenvolvimento e teste (BORLAND, 2006):

- **Plan:** Otimizar o portfólio de pedidos de projetos para que ele fique alinhado aos objetivos comerciais, defina os recursos corretos para a tarefa e gerencie os ativos;
- **Define:** Identificar os requisitos comerciais de projetos de softwares críticos; alinhar os entregáveis do software e os objetivos comerciais; melhorar a previsibilidade de seu processo de criação de softwares.
- **Design:** Fornecer auxílio de fácil compreensão aos desenvolvedores; manter padrões e requisitos arquitetônicos à medida em que se reforçam os padrões corporativos de desenho de aplicações.
- **Develop:** Alavancar as melhores capacidades de desenvolvimento, para assegurar que as aplicações sejam desenhadas e construídas dentro dos padrões e das especificações.

- **Test:** Fornecer a desenvolvedores e profissionais de garantia de qualidade todas as capacidades de qualidade da aplicação, desde testes funcionais e de regressão até tecnologias de gerenciamento de carga, performance e teste, desenhadas para identificar e solucionar questões de qualidade mais cedo no decorrer do ciclo de criação do software.

Sendo também um processo sistêmico e que contém as melhores práticas da engenharia de software, a adoção de uma metodologia, permitirá aderência de normas e modelos como ISO 12.207 e CMMI.

Além dos benefícios que a adoção da modelagem com a UML já traz, a implementação de um processo completo de desenvolvimento de software com uma metodologia, evitará fracassos nas fases mais críticas de todo o projeto.

4.6 Normas e Modelos

Existem inúmeras norma e modelos de qualidade de software. No entanto 04 são considerados os principais:

O CMMI – (*Capability Maturity Model Integrated*) foi desenvolvido pelo SEI (*Software Engineering Institute)*, ligado à CMU (*Carnegie Mellon University)*, em Pittsburgh, nos Estados Unidos. O desenvolvimento desse modelo foi financiado pelo DoD, Departamento de Defesa Americano, com o objetivo de se estabelecer um padrão de qualidade para software desenvolvido para as forças armadas. O CMMI foi concebido para o desenvolvimento de grandes projetos militares e, para a sua aplicação em projetos menores e em outras áreas, é necessário um trabalho cuidadoso de interpretação e adequação à realidade da organização (FIORINI, 1998).

A norma internacional NBR ISO/IEC 12207 – Tecnologia da Informação – Processos de Ciclo de Vida de Software (ISO12207, 1997) é usada como referência em muitos países, inclusive no Brasil, para alcançar diferencial competitivo. Ela tem por objetivo auxiliar os envolvidos na produção de software a definir seus papéis, por meio de processos bem definidos, e assim proporcionar às organizações que a utilizam um melhor entendimento das atividades a serem executadas nas operações que envolvem, de alguma forma, o software.

A família ISO 9000 é composta de uma série de normas, e reconhece que existem 4 diferentes categorias genéricas de produtos e publicou diretrizes para implementação de sistemas da qualidade para cada uma destas categorias:Hardware: ISO 9004-1; Serviços: ISO 9004-2;Materiais Processados: ISO 9004-3;e Software: ISO 9000-3.

Devido às dificuldades específicas de interpretação de como implantar os requisitos da ISO 9001 ou 9002 em software, é fundamental o uso da ISO 9000-3 para auxiliar a implantação do sistema de gestão da qualidade. Esta dificuldade está relacionada com a terminologia usada na ISO 9001, muito voltada para hardware; usando a ISO 9000-3 esta barreira é eliminada (XAVIER, 2001).

A ISO/IEC 15504, SPICE (*Software Process Improvement and Capability dEtermination)*, presta-se à realização de avaliações de processos de software com dois objetivos: a melhoria dos processos e a determinação da capacidade de processos de uma organização. Se o objetivo for à melhoria dos processos, a organização pode realizar a avaliação gerando um perfil dos processos que serão usados para a elaboração de um plano de melhorias. A organização deve definir os objetivos e o contexto, bem como escolher o modelo e o método para a avaliação e definir os objetivos de melhoria. (SALVIANO, 2001).

A adoção de normas como ISO 12.207, a ISO 9000-3 e modelos de qualidade como o CMMI e SPICE, agrega maturidade ao processo de produção de software. Diante o quadro atual de baixa adoção das normas e modelos aqui no Brasil, segundo o MCT, citado neste trabalho, temos aqui identificado que a "Crise do Software" se justifica pela falta de adoção e uso de forma sistêmica.

4.7 Métricas

Estimativas de esforço, custo, prazo e qualidade são necessárias ao longo de todo o projeto, embora mais críticas no momento de formulação da proposta ou orçamento. A adoção da prática de medição permite ao estimar de forma mais racional todo o processo de desenvolvimento de software. Existem várias métricas determinadas para cada tipo de processo. As principais são: FPA – Pontos por Função, COCOMO – Constructive Cost Model, KLOC – Lines of Code e UCP – Pontos por Caso de Uso.

Apesar da importância da adoção de métricas, a utilização nas empresas brasileira é muito baixa. Apenas 18,6% utilizam (MCT, 2002) (Tabela 11).

TABELA 11 – Métricas utilizadas nos processos de software.

Métrica	Total (%)
Linhas de Código (*LOC*)	**5,6**
Pontos por Função (*Function Point*)	**9,6**
Outras Métricas	**5,8**
Não Utiliza	**81,4**

4.8 Cronogramação

Apesar de haver muitas razões pelas quais o software é entregue atrasado, a maioria pode ser rastreada para uma ou mais das seguintes causas básicas (PRESSMAN, 2006):

- Data de Entrega irrealista estabelecida por alguém fora do grupo desenvolvimento e imposta a gerentes e profissionais do grupo;
- Mudanças nos requisitos;
- Subestimativa honesta da quantidade de esforço;
- Riscos Previsíveis e Imprevisíveis que não foram considerados;
- Dificuldades Técnicas;
- Dificuldades Humanas;
- Falta de Comunicação;
- Falha na gerência do projeto.

A especificação do escopo do projeto permite que seja feita a cronogramação do projeto. Quanto mais detalhado estiver, mais simplificada a tarefa de estimar quanto cada tarefa terá de duração.

Tal prática fundamental faz parte do hall de gerências tratadas pelo manual de conhecimento em gestão de projetos o PMBOK.

4.9 Implementação

A Implementação (*Implementation*) trata-se da criação de programas de computador, segundo as especificações técnicas existentes. Espera-se que os programas sejam criados segundo os conceitos da orientação a objetos e sejam testados (testes de unidade). Também

devem ser construídas as integrações eventualmente necessárias com outros sistemas (TONSIG, 2003).

Atualmente existem inúmeras ferramentas de implementação. No entanto a sua adoção tem relação direta com a produtividade e necessidades do projeto.

Uma ferramenta que integre recursos de especificações de requisitos, gestão de configuração e modelagem são mais adequadas para atender todos os paradigmas aqui especificados, bem como propiciar o efetivo resultado esperado.

4.10 Testes

A adoção do processo de testes é crítico para o desenvolvimento de software. Embora as revisões técnicas sejam mais eficazes na detecção e remoção de defeitos, os testes são importantes para complementar às revisões e aferir o nível de qualidade conseguido. A realização de testes é, quase sempre, limitada por restrições de cronograma e orçamento; eles determinam quantos testes será possível executar. É importante que os testes sejam bem planejados e desenhados, para conseguir-se o melhor proveito possível dos recursos alocados para eles (PRESSMAN, 2006).

Um objetivo central de toda a metodologia dos testes é maximizar a sua cobertura, ou seja, a quantidade potencial de defeitos que podem ser detectados por meio do teste. Deseja-se conseguir detectar a maior quantidade possível de defeitos que não foram apanhados pelas revisões, dentro de dados limites de custo e prazos.

Os testes são indicadores da qualidade do produto, mais do que meios de detecção e correção de erros. Quanto maior o número de defeitos detectados em um software, provavelmente maior também o número de defeitos não-detectados. A ocorrência de um número anormal de defeitos em uma bateria de testes indica uma provável necessidade de redesenho dos itens testados.

Existem basicamente duas maneiras de se construírem testes:

- Método da caixa branca: tem por objetivo determinar defeitos na estrutura interna do produto, através do desenho de testes que exercitem suficientemente os possíveis caminhos de execução;
- Método da caixa preta: tem por objetivo determinar se os requisitos foram total ou parcialmente satisfeitos pelo produto. Os testes de caixa preta não verificam como ocorre o processamento, mas apenas os resultados produzidos.

É importante que os desenvolvedores de software reconheçam que não é possível desenvolver sistemas com qualidade, cumprir prazos e custos e atender às expectativas dos usuários sem ter um processo de testes definido, compreendido e utilizado por toda a equipe.

O nível de complexidade da sua implementação pode ser dimensionado de acordo com o porte do sistema, viabilizando para qualquer tamanho de organizações desenvolvedoras de software.

5. Conclusão

Atualmente já existe um movimento das empresas para adoção de normas e modelos de maturidade do processo de desenvolvimento de software, buscando melhor produtividade. Há ênfase em promover uma reengenharia nos processos de desenvolvimento de software, que até então eram basicamente vindos da experiência dos “desenvolvedores de código” e não de gestores de projetos de grande expressão, e que assumem papel de alta relevância nas

empresas para se obter vantagens competitivas num mercado que busca a informação certa no momento certo.

Este artigo teve o objetivo de propiciar conhecimento para a realização de projetos de software com qualidade através da adoção dos paradigmas de qualidade diante da problemática da "Crise do Software".

O contexto brasileiro apresenta números de baixa adoção de processos fundamentais da Engenharia de Software que são fatores críticos de sucesso nos projetos de software.

Os softwares que atenderem os requisitos dos clientes e aderirem aos processos de negócios da empresa, naturalmente proporcionarão novos negócios e principalmente vantagens competitivas, alcançando efetivamente a excelência empresarial.

Aliada a capacidade e criatividade do profissional brasileiro, espera-se despertar nesses profissionais ligados a área de engenharia de software, a importância da adoção desses fatores, sendo críticos para sucesso no desenvolvimento de software.

Mudando radicalmente o cenário atual com o profissionalismo necessário, será possível atender de fato as necessidades das organizações nacionais e internacionais, obtendo reconhecimento de que o Brasil também é um pólo mundial de produção de software com qualidade.

6. Referências

BABICH, W.A., *Software Configuration Management,* Addison-Wesley, 1986.

BORLAND, ALM , http://www.borland.com/br/products/alm/index.html acessado em 11/08/2006.

BOTELHO, JOAQUIM, Brasil, o país dos empreendedores. Acessado em 10/10/2006. http://www2.uol.com.br/aprendiz/guiadeempregos/palavra/jbotelho/ge171001.htm

BROOKS, F. P. No silver bullet: essence and accidents of software engineering, in H. Kugler, ed., `Information Processing 86', Elsevier Science (North Holland), pp. 1069-1076.

CNN MONEY, *BEST JOBS*, http://money.cnn.com/magazines/moneymag/bestjobs/ acessado em 13/08/2006.

FILHO, WILSON DE PÁDUA PAULA, *Engenharia de Software*, Rio de Janeiro, Ed. LTC, 2003.

FIORINI, SOELI T., et al. *Engenharia de Software com CMM*, Rio de Janeiro, Ed. Brasport, 1998.

LEE, RICHARD C. e TEPFENHART, WILLIAM M., *UML e C++ - Guia de desenvolvimento orientado a objeto*, São Paulo, Ed. Makron Books, 2002.

LEITE, JULIO CESAR SAMPAIO DO PRADO, in WEBER, KIVAL CHAVES, et al. *Qualidade e Produtividade em Software*, São Paulo, Ed. Makron Books, 2001.

MACHADO, CRISTINA ÂNGELA FILIPAK in WEBER, KIVAL CHAVES, et al. *Qualidade e Produtividade em Software*, São Paulo, Ed. Makron Books, 2001.

MAFFEO, BRUNO, *Engenharia de Software e Especificação de Sistemas*, Rio de Janeiro, Ed. Campus, 1992.

MINISTÉRIO DA CIÊNCIA E TECNOLOGIA, Secretaria de Política de Informática, *Qualidade e Produtividade no Setor de Software Brasileiro*, Brasília, N.4, 2002.

NBR ISO/IEC 12207:1997, *Tecnologia de Informação – Processos de Ciclo de Vida de Software*, Rio de Janeiro, ABNT – Associação Brasileira de Normas Técnicas.

NOGUEIRA, MARCELO, Um framework para a gestão de Riscos em Projetos de Software, Dissertação de Mestrado, UNIP, São Paulo, 2004.

PETERS, JAMES F. et al. Engenharia de Software,Rio de Janeiro, Ed. Campus,2001.

PMBOK, Project Management Institute, 2004.

PRESSMAN, ROGER S., *Engenharia de Software*, Rio de Janeiro, Ed. McGraw-Hill, 2006.

REZENDE, DENIS ALCIDES, *Engenharia de Software e Sistemas de Informações*, Rio de Janeiro, Ed. Brasport, 2005.

SALVIANO, CLENIO, in WEBER, KIVAL CHAVES, et al. *Qualidade e Produtividade em Software*, São Paulo, Ed. Makron Books, 2001.

SEI, *Software Engineering Institute*, *Carnegie Melon University*, *http://www.sei.cmu.edu.*

SOMMERVILLE, IAN, *Engenharia de Software*, São Paulo, Ed. Pearson Education, 2003.

STANDISH, GROUP, Chaos Report, 1994. http://www.standishgroup.com/sample_research/chaos_1994_1.php Acessado em 10/08/2006.

STANDISH, GROUP, Chãos Report, 2000. Acessado em 10/08/2006. http://www.standishgroup.com/sample_research/PDFpages/extreme_chaos.pdf

TELES, VINÍCIUS MANHÃES, Um Estudo de Caso da Adoção das Práticas e Valores do Extreme Programming, Dissertação de Mestrado, UFRJ, Rio de Janeiro, 2005.

TONSIG, SERGIO LUIZ, Engenharia de Software, Ed. Futura, São Paulo, 2003.

XAVIER, JORGE HERCULES, in WEBER, KIVAL CHAVES, et al. *Qualidade e Produtividade em Software*, São Paulo, Ed. Makron Books, 2001.