Scientia
Interdisciplinary Studies in Computer Science
19(2): 102-116, July/December 2008
 – doi: 10.4013/sct.20082.03

Elementos para uma abordagem sociotécnica do desenvolvimento de software com *Extreme Programming*

Rodrigo Pereira dos Santos

COPPE/UFRJ – Universidade Federal do Rio de Janeiro
Centro de Tecnologia, Bloco H, Cidade Universitária, Ilha do Fundão
Caixa Postal 68511, Rio de Janeiro, 21945-970, RJ, Brasil

rps@cos.ufrj.br

Resumo
Considerando que empresas de desenvolvimento de software convivem em ambientes de negócios sujeitos a mudanças freqüentes, *Extreme Programming* (XP) surge como uma alternativa que almeja a criação de software de qualidade, de maneira ágil, econômica e flexível. Por outro lado, modelos de maturidade e capacidade tradicionais, como o CMMI, são caracterizados por uma grande quantidade de atividades e de artefatos que buscam organizar e proteger o software contra mudanças. Inicialmente, XP e CMMI foram consideradas vertentes de desenvolvimento antagônicas e pesquisas realizadas apresentam diferentes perspectivas em relação ao tratamento dessa questão, seja pela valorização de uma vertente em detrimento da outra, seja pela verificação de possíveis misturas entre elas. Nesse sentido, o objetivo deste trabalho é realizar uma investigação das relações entre duas vertentes de desenvolvimento de software, a tradicional (representada pelo CMMI) e a ágil (representada por XP). Busca-se identificar elementos para uma abordagem sociotécnica do desenvolvimento de software com XP, na forma de um debate que envolve percepções diferenciadas na Engenharia de Software. A partir de uma combinação entre raciocínio lógico e informações extraídas de alguns especialistas e de publicações relacionadas, apresentam-se possíveis combinações e similaridades entre essas duas vertentes atuais de desenvolvimento de software.

PALAVRAS-CHAVE: Engenharia de Software, abordagem sociotécnica de *Extreme Programming*, métodos dirigidos a planos *versus* métodos ágeis, processos de desenvolvimento de software.

Abstract
Elements for a sociotechnical approach of software development with Extreme Programming. Considering that software development companies are in business atmospheres subject to frequent changes, Extreme Programming (XP) appears as an alternative that longs for the creation of quality software, in an agile, economic and flexible way. On the other hand, traditional maturity and capacity models like CMMI are characterized by a large amount of activities and artifacts that aim to organize and to protect the software against changes. Firstly, XP and CMMI were considered antagonistic styles of software development and studies carried out so far show different perspectives in relation to how such issue is handled, or they give more value to one of them than to the other, or we verify possible mergings of both. In this sense, the objective of our study is to investigate the relationships between traditional (represented by CMMI) and agile (represented by XP) software development styles. The identification of elements for a sociotechnical approach of software development with XP is aimed, showing a discussion that involves different perceptions in Software Engineering. Starting from a combination between logical reasoning and information extracted from some specialists and related works, possible combinations and similarities among these current styles of software development are presented.

KEY WORDS: Software Engineering, sociotechnical approach of Extreme Programming, plan-driven methods versus agile methods, software development processes.

1 Introdução

Um dos problemas na fase de concepção de produtos de software é a dificuldade da definição do escopo do projeto (Ferreira e Lima, 2006). Uma vez que essa fase envolve lidar com as funcionalidades do produto de software (requisitos funcionais e não-funcionais), deve-se cuidar para que o processo adotado pela organização seja flexível com relação ao atendimento das necessidades dos clientes. Almejar uma etapa de negociação e de elaboração de documentação inicial não "desgastante" deve ser uma meta, dado que a inserção do cliente no projeto traz melhorias para a definição do escopo (Boehm, 2006). Outro ponto importante consiste em reduzir o distanciamento que ocorre entre as etapas de concepção e de entrega do produto de software, sobretudo considerando o cenário dinâmico do mundo atual (Boehm, 2003). Nesse caso, deve-se encarar a mudança como uma característica presente em qualquer projeto que demanda espírito de equipe e que visa a aprendizagem, e não como um problema ou falha de projeto (Fonseca Filho, 1999).

A partir dessas informações, percebe-se que as questões que envolvem o processo de produção de software extrapolam os fatores técnicos de um projeto. Assim, um processo de desenvolvimento de software precisa contemplar os fatores não-técnicos envolvidos, ou seja, ter o discernimento suficiente para entender que o sucesso de um projeto não se deve apenas ao correto cumprimento de atividades estabelecidas e de boas práticas, e que as falhas existentes não decorrem apenas dos fatores não-técnicos (Teixeira e Cukierman, 2007). Somam-se, ainda, as características peculiares de cada organização, tais como o porte e a capacidade de investir e prover a qualidade idealizada por processos e resultados definidos em modelos de maturidade e capacidade de escala global, como o CMMI (*Capability Maturity Model Integration*) (CMMI, 2006). Cada organização, com seu estilo próprio, implementa esses processos e eventualmente "se culpa" por quaisquer falhas em projetos, o que pode gerar frustração e desestímulo em seus recursos humanos.

A idéia incutida nos *stakeholders* do processo de que modelos de maturidade e capacidade focam em fatores técnicos e objetivos ("infalíveis") e não se referem a fatores não-técnicos (Teixeira, 2006) pode formar uma corrente que acredita que *implementá-los é a solução de todos os problemas de falhas*. Deve-se ressaltar que a implantação de processos definidos por alguns desses modelos, como o CMMI, exige gastos elevados, o que dificulta a sua adoção, dependendo do porte da organização. Nesse sentido, algumas alternativas emergiram, tais como modelos mais focados na realidade nacional, como o programa MPS.BR (Melhoria de Processo do Software Brasileiro) (SOFTEX, 2007), e abordagens ágeis, como as Metodologias Ágeis, com destaque para *Extreme Programming* (XP) (Beck, 2000).

Por um lado, o MPS.BR pode trazer resultados interessantes a um custo menor para a organização, favorecendo a sua adoção em pequenas e médias empresas, ao considerar características específicas e locais e ao suavizar a relação custo-benefício acerca da implementação e avaliação de processos que buscam agregar qualidade ao software (SOFTEX, 2007). Por outro lado, as metodologias ágeis, como XP, apresentam valores, princípios e práticas, aproveitados de processos industriais e adaptados ao software, os quais sugerem a obtenção de qualidade, ainda sem tanta clareza sobre em que nível e em que escala (Boehm, 2006). Mais especificamente, a estrutura de XP visa melhorar as relações com os clientes e entre os membros da equipe, realçando o ambiente e os recursos disponíveis na organização e buscando "quebrar" problemas grandes em um conjunto de vários sub-problemas, com o intuito de melhorar a qualidade das soluções (Teles, 2005). Entretanto, deve-se cuidar para que XP não seja encarada como mais uma "bala de prata" (Travassos, 2007).

Em todos os casos (CMMI, MPS.BR e metodologias ágeis, como XP), vale ressaltar a importância de se explorar uma discussão baseada em observação e em evidência no campo da Engenharia de Software, que permita delinear *o que* deve ser aplicado *em um* cenário específico, que contenha *certas* características e que necessite de *certos* tipos de soluções. Isso reflete um dos desafios da Engenharia de Software: tornar-se experimental e basear-se nos resultados provenientes deste processo (Pfleeger, 1999; Kitchenham *et al.*, 2002). Nessa linha, destaca-se a importância de se considerar elementos que extrapolam uma mera análise de fatores técnicos do desenvolvimento de software – seja usando XP, seja seguindo o CMMI ou o MPS.BR –, buscando relacionar fatores técnicos e não-técnicos e entender a influência dessa combinação no cenário industrial e a sua "co-modificação" a partir da experiência social, somente percebida por uma aproximação concomitantemente social e técnica. Conforme Cukierman *et al.* (2007), isso representa o olhar sociotécnico, o qual almeja apreender a Engenharia de Software sem fragmentá-la em fatores (ou aspectos) técnicos de um lado e fatores (ou aspectos) não-técnicos de outro; sem fatorá-la em quaisquer outras dualidades ("fatores técnicos" *versus* "fatores humanos, organizacionais, éticos, políticos, sociais etc.") que terminem por desfigurar o "pano sem costura" que imbrica, na Engenharia de Software, o técnico e o social em um mesmo e indivisível "tecido".

Assim, no cenário atual da indústria de software, torna-se relevante entender o surgimento e o processo de construção e reconstrução das novas vertentes de desenvolvimento de software (XP e MPS.BR) diante de "modelos universais" como o CMMI, uma vez que elas analisam o

"local" e o "situado" (que resiste ao "global" e "universal") e apontam o "caso a caso" e a contingência. Mais especificamente no escopo deste trabalho, o objetivo consiste em realizar uma investigação inicial das relações entre duas vertentes que guiam o processo de desenvolvimento de software, a tradicional (representada pelo CMMI) e a ágil (representada por XP), visando reunir elementos para uma abordagem sociotécnica do desenvolvimento de software com XP, na forma de um debate que envolve percepções diferenciadas na Engenharia de Software. A partir de uma combinação entre raciocínio lógico e informações extraídas de alguns especialistas e de publicações relacionadas, apresentam-se possíveis combinações e similaridades entre essas vertentes atuais de desenvolvimento.

Com essa finalidade, será considerado o cenário de surgimento da Engenharia de Software (seção 2). A partir disso, resgatam-se algumas pesquisas que relacionam vertentes de desenvolvimento de software tradicionais e ágeis, de tal forma que esse resgate embase uma discussão acerca de perspectivas distintas sobre as relações entre as características de XP e do CMMI, incluindo uma visão de XP, a partir do "mundo" do CMMI, e uma visão do CMMI, a partir do "mundo" de XP (seções 3 e 4). Assim, procurar-se-á entender a existência de iniciativas (XP) que buscam construir uma identidade a partir da perspectiva de um "modelo universal" (CMMI) (seção 3). Por outro lado, agregando uma visão da realidade, compõe-se um paralelo entre uma experiência relativa à "antítese" software proprietário/software livre e duas entrevistas que envolvem experiências e pontos de vista distintos (um professor/pesquisador e um analista de sistemas) acerca da "antítese" tradicional/ágil (seção 4). Por fim, traçam-se as considerações finais e apontam-se trabalhos futuros (seção 5).

2 O surgimento da Engenharia de Software e a concepção dos processos de desenvolvimento na década de 1990 e nos tempos atuais

Tradicionalmente, nota-se uma característica da modernidade ocidental: a busca por "modelos universais" a partir da identificação de estruturas e hierarquias que dêem suporte à padronização e à comunicação; e isso não ocorreu de modo diferente no desenvolvimento de software (Dahlbom e Mathiassen, 1993). De acordo com Fonseca Filho (1999), a investigação histórica sob o prisma das idéias e conceitos que embasaram o desenvolvimento da Computação contribuiu para uma questão decisiva e crucial: a **qualidade do software**. Deve-se recordar que, em meados da década de 1960, ocorreu um ajuste compreensível entre os custos relacionados ao hardware e ao software. A redução de custos de hardware viabilizava a utilização de computadores em aplicações que requeriam produtos de software mais complexos e a escalabilidade os tornava difíceis de supervisionar, controlar e coordenar, uma vez que os problemas que afligiam os desenvolvedores eram mais gerenciais do que técnicos. A questão ganhou foco e foi rotulada pela expressão "Crise do Software" (Pressman, 2006). Surgia, ainda, a percepção de que as organizações precisavam reduzir a dependência dos profissionais, necessitando de metodologias efetivas para o gerenciamento dos projetos e para o controle do processo de desenvolvimento (Teixeira e Cukierman, 2007). Assim, visando agregar princípios de engenharia (processo e produto) ao desenvolvimento de software de qualidade, surgiu a Engenharia de Software (Pressman, 2006).

Segundo Boehm (2006), as estratégias de formalização (abordagens quantitativas) e o processo de desenvolvimento de software no modelo cascata (em substituição à abordagem codifica-corrige da década anterior) surgiram na Engenharia de Software na década de 1970. Na década seguinte (1980), a produtividade e a escalabilidade de projetos sofreram com problemas relativos à variedade de métodos quantitativos e à parcela do esforço destinada aos testes, devido ao retrabalho. Diante dessa realidade, Boehm (2006) destaca um importante passo em direção à centralização, à hierarquização e à padronização, entendido como um elemento importante para a construção e o estabelecimento da Engenharia de Software. O cenário envolve um personagem notável na construção e evolução dos computadores, o Departamento de Defesa Americano (DoD) (Edwards, 1996).

> *Os problemas com o descumprimento do processo foram resolvidos inicialmente por padrões contratuais, [...] os quais reforçaram o modelo cascata por amarrar seus marcos a revisões da gerência, bonificações e prêmios para progressos ao longo do processo de desenvolvimento. Ao falharem (freqüentemente) em distinguir entre os bons desenvolvedores [que seguiam os processos] e os desenvolvedores ad hoc [desatentos aos processos], o DoD escalou* CMU *[Carnegie Mellon University]* Software Engineering Institute *[SEI] para desenvolver um* modelo de maturidade e capacidade de software (SW-CMM) *[Software Capability Maturity Model] e métodos associados [...]. (Boehm, 2006, p. 17,* grifo nosso*).*

Dessa forma, o SW-CMM surge como uma proposta de modelo de referência a ser seguido pelas empresas de desenvolvimento de software. Boehm (2006) ainda explica quais eram os fundamentos desse modelo e mostra que, em resposta à iniciativa americana, os europeus desenvolveram um padrão equivalente.

> *Baseados extensivamente nas práticas de software altamente disciplinadas da IBM e nos níveis de maturidade e práticas de qualidade de Deming-Juran-Crosby, o SW-CMM resultante forneceu um framework efetivo para a definição e melhoria de capacidade. […] Um* padrão similar *para práticas de qualidade aplicáveis ao software,* International Standards Organization ISO-9001*, foi desenvolvido concorrentemente, sob* liderança européia *(Boehm, 2006, p. 17,* grifo nosso*).*

A criação do SEI, na década de 1980, e o surgimento do SW-CMM e da ISO-9001 evidenciaram o estabelecimento de processos universais que representavam padrões a serem alcançados pelas organizações. Apesar desses padrões simplificarem a visualização dos resultados esperados, um certo mal-estar foi gerado diante da exigência do cumprimento desses processos, os quais demandavam altos custos de implantação e treinamento (possíveis apenas para as grandes organizações). Boehm (2006) afirma que a ameaça de desqualificação para as ofertas de projetos levou a maioria das empresas a investir em adequações ao SW-CMM e à ISO-9001. A maior parte dos lucros dos investimentos registrados se referia à redução do retrabalho. Estes resultados espalharam o uso dos modelos de maturidade e conduziram a uma rodada de refinamento e desenvolvimento dos padrões e modelos discutidos na década de 1990. A difusão do SW-CMM (baseado nas melhores práticas do DoD e por ele financiado) se intensificou como uma importante contribuição para a manutenção da Engenharia de Software.

A partir de experiências e expectativas, o SW-CMM evoluiu para CMMI (*Capability Maturity Model Integration*), em 2001 (Chrissis *et al.*, 2003). O CMMI procura estabelecer um modelo único para o processo de melhoria corporativo, integrando diferentes modelos e disciplinas que são passíveis de implantação e de avaliação, e enquadrando as organizações em cinco níveis de maturidade (CMMI, 2006). Devido ao grau de dificuldade de compreensão organizacional, este modelo não favorece as pequenas e médias empresas. Este fato abriu oportunidades para iniciativas como o MPS.BR, no Brasil. Entretanto, o cenário da década de 1990 mostrava a necessidade de desenvolvimento rápido de aplicações, devido à aceleração das mudanças na tecnologia da informação, nas organizações, em parceiros competitivos e no ambiente econômico (Teles, 2005). Isso causou um aumento da frustração com planos, especificações e outras documentações pesadas impostas pela inércia contratual e pelo critério de cumprimento do modelo de maturidade (Boehm, 2006).

Diante da estrutura do CMMI e da realidade das pequenas e médias empresas, e considerando o fato de que *mudanças*, *comunicação* e *pessoas* são aspectos inerentes ao desenvolvimento de software de qualidade (Sato, 2007), o final da década de 1990 presenciou o surgimento dos métodos ágeis, dentre eles o mais utilizado, *Extreme Programming* (XP), atribuído a Kent Beck, Ron Jeffries e Ward Cunningham (Beck, 2000). Um grupo – formado por 17 desenvolvedores experientes, consultores e líderes da comunidade de software – se reuniu em Utah, em fevereiro de 2001, para discutir idéias e procurar alternativas aos processos burocráticos e às práticas apregoadas pelas abordagens tradicionais da Engenharia de Software e da Gerência de Projetos. Essa reunião conduziu à redação do *Manifesto do Desenvolvimento Ágil de Software* (Beck *et al.*, 2001), o qual aponta quatro valores principais: (i) *indivíduos e interações vêm antes de processos e ferramentas*; (ii) *software funcionando vem antes de documentação abrangente*; (iii) *colaboração do cliente vem antes de negociação de contrato*; e (iv) *resposta à modificação vem antes do plano em andamento*.

XP tem como premissa técnica a combinação de: (i) inserção do cliente no projeto; (ii) liberações de pequenos incrementos ao longo das iterações que ocorrem no desenvolvimento; (iii) projeto simples; (iv) programação em pares; (v) refatoração; e (vi) integração contínua, visando obter uma curva mais branda do custo de mudanças em função do tempo (Beck, 2000). A partir de análises de dados coletados, Boehm (2006) afirma que esse abrandamento não acontece em projetos maiores, pois eles necessitam de planos mais explícitos, maior controle e representações de arquitetura de alto nível. No entanto, alguns estudos mostram que é possível aplicar XP em projetos de maior porte, através de extensões que enfatizam *feedback* constante, projeto de software e criação de novos papéis no ciclo de desenvolvimento (Lippert *et al.*, 2003; Eckstein, 2006). Essas extensões foram aplicadas em projetos de diferentes perspectivas (e.g., *web*, *desktop*, processamento em *batch*, dispositivos móveis, dentre outros), localizações (software interno às instalações do cliente ou em ambientes externos), números de integrantes e qualificações dos profissionais envolvidos, e buscaram entender e prover meios para lidar com projetos relativos a domínios de negócio complexos, sem limitar as vantagens oferecidas pelo desenvolvimento ágil com XP.

Em suma, análises relativas a métodos ágeis mostram que eles tiveram maior aplicabilidade em pequenos projetos, de risco relativamente baixo, envolvendo pessoal altamente capacitado e requisitos passíveis de mudanças (Boehm, 2006). Nota-se, então, que XP representa uma das possíveis vertentes de desenvolvimento de software, diante de "modelos universais" como o CMMI, que organizações de pequeno e médio porte adotam, almejando a produção de software de qualidade dentro de suas limitações físicas, temporais, financeiras e de pessoal. Mas deve-se recordar que *pessoal*, *disciplina* e *escopo* são fatores importantes na adoção de XP.

3 Identificando similaridades e combinações entre as vertentes de desenvolvimento de software tradicionais e ágeis

Sabe-se que existe um debate na comunidade de Engenharia de Software sobre a utilidade e a aplicabilidade de metodologias clássicas ou "modelos universais", como o CMMI, em detrimento de metodologias ágeis, como XP. Conforme Jiang e Eberlein (2008), as características típicas desse debate, externalizadas pelos representantes das duas metodologias, são: (i) descrição do modelo "oposto" por meio de termos extremos e enviesados; (ii) desvalorização das metodologias e/ou práticas do modelo "oposto"; e (iii) justificativa dos valores do modelo defendido, seja por meio de uma explicação baseada em experiência, seja através de comparações inadequadas entre as metodologias. Esse debate, às vezes, recebe a denominação de "guerra metodológica" (Glass, 2001; Austin e Devin, 2003).

Nesse contexto, alguns pesquisadores têm apresentado similaridades e compatibilidades entre as duas metodologias (Glass, 2001; Turner e Jain, 2002; Eberlein, 2003; Paetsch *et al.*, 2003; Fritzsche e Keil, 2007). Boehm (2002) e Boehm e Turner (2003) afirmam que cada metodologia oferece seus próprios valores para diferentes projetos de software e discutem sugestões e argumentos para guiar equipes que desejam utilizar metodologias ágeis em empresas que também utilizam metodologias clássicas. Glass (2001) reforça que a discordância entre os defensores das metodologias ágeis e das clássicas se baseia em crenças conflitantes, relativas a como um processo de Engenharia de Software é melhor organizado e realizado, indicando que a "disputa" pode ser, na realidade, um problema de atitudes individuais e/ou organizacionais, hábitos de trabalho e visões precipitadas sobre as diferentes metodologias e seus relacionamentos.

Além disso, um dos principais aspectos do debate entre os defensores de cada metodologia se refere ao valor dos chamados "processos pesados" que compõem modelos de processo baseados em planos (como o CMMI). Reifer (2003) argumenta que práticas de XP e do CMMI são compatíveis filosoficamente e discute o fato de que mais pesquisas se tornam necessárias para prover diretrizes de uso das melhores práticas de cada uma das metodologias. Da mesma forma, Boehm (2002) mostra que a utilização pura de metodologias ágeis ou clássicas pode desfavorecer as soluções requeridas por projetos de software e discute que uma mistura das melhores práticas das metodologias é altamente desejável. Glass (2001) também defende a convergência de valores e princípios de ambas as metodologias, a fim de contribuir para o desenvolvimento e para a evolução da Engenharia de Software. Ainda, Turner e Jain (2002) mostram que as práticas ágeis suportam onze componentes do CMMI, ressaltando a possibilidade de combinação entre as duas metodologias.

Apesar das pesquisas existentes, Jiang e Eberlein (2008) apontam que a literatura existente ainda não interpreta bem o suficiente o relacionamento entre as duas categorias de metodologias, em termos das respectivas filosofias operacionais. A fim de contribuir com esse cenário, as subseções a seguir visam verificar como o "modelo universal", provido pelo CMMI, visualiza iniciativas (como XP) de modificação ou reestruturação de processos de desenvolvimento de software, ao perceber a essência e a missão envolvida na criação de uma metodologia de desenvolvimento baseada em agilidade. Para isso, os pontos de vista de alguns autores serão analisados e discutidos.

3.1 Visão de XP a partir da perspectiva do SW-CMM

Mark C. Paulk, membro do SEI ("berço do SW-CMM e do CMMI"), apresenta, em seu trabalho, uma visão de XP a partir da perspectiva do SW-CMM[1] (Paulk, 2001). Segundo Paulk (2001), XP consiste em uma "metodologia ágil" defendida por algumas pessoas devido a sua *velocidade*, à *volatilidade do mundo do software* e ao *desenvolvimento web*. Embora XP seja um processo disciplinado, Paulk afirma que alguns o utilizam como argumento contra "rigorosos" modelos de melhoria de software, como o CMMI. No entanto, XP tem boas práticas de engenharia, que podem trabalhar bem em conjunto com outros métodos altamente estruturados: a chave está em considerar cuidadosamente as práticas de XP e implementá-las no ambiente certo. Assim, Paulk sugere o que se pode chamar de "hibridação" um trabalho conjunto entre a abordagem ágil, representada por XP, e o modelo de melhoria da qualidade de software, o CMMI, por meio de composição de seus princípios.

[1] A análise cuidadosa do artigo de Paulk decorre do fato de que ele representa um material valioso para a captura de elementos para uma abordagem sociotécnica de XP, uma vez que seu discurso vislumbra uma estratégia de *marketing* interessante, aos moldes do próprio CMMI. Ainda, Paulk utiliza SW-CMM ou CMM para se referir ao modelo de maturidade e capacidade do CMU/SEI. Como o artigo data de 2001, quando SW-CMM evoluía para CMMI, será utilizada a denominação atual CMMI, pois ambos apresentam, em suma, conceitos e princípios equivalentes.

> SW-CMM *foca em* problemas de gerência *envolvidos* na implementação de processos eficientes e efetivos e na melhoria de processos de software. XP*, por outro lado, é um* conjunto específico *de práticas – uma 'metodologia' – que é* efetiva no contexto de equipes pequenas, co-localizadas, onde requisitos mudam rapidamente. *Tomadas* juntas*, as duas metodologias podem* criar sinergia*, particularmente em* conjunção com outras boas práticas de gerência e engenharia *(Paulk, 2001, p. 5,* grifo nosso*).*

Observa-se que o ponto de vista do CMMI (expresso por um membro do SEI) agrega princípios norteadores (centralização, hierarquia e estrutura) e demonstra um posicionamento pacífico diante de abordagens alternativas como XP, com um bom discurso. A racionalidade se faz importante para lidar com abordagens alternativas, uma vez que o objetivo do artigo é apontar a viabilidade de sinergia e conjunção entre as práticas do CMMI e de XP. Apesar disso, inicialmente, percebe-se a existência de duas seções separadas no artigo, uma para desenvolver o CMMI e outra para desenvolver XP, a fim de caracterizar os "mundos" de cada abordagem antes de discorrer sobre uma possível combinação entre elas.

Com base nisso, Paulk indaga "*Why explore XP?*". Ele aponta que as organizações deveriam capturar os valores de XP em seus projetos (embora seus processos tenham implementações diferentes de XP) e mostra que as idéias de comunicação e simplicidade de XP existem em processos tradicionais, sob outra nomenclatura (e.g., coordenação e elegância) – mas enfatiza que o que deve acontecer é a prática de tais valores. Na parte final do artigo (*False Opposition*), Paulk mostra que XP (implicitamente) necessita de práticas do CMMI para atingir seus objetivos: *apenas tocar em questões gerenciais e organizacionais (enfatizadas pelo CMMI) é a principal objeção quanto ao uso XP para melhoria de processo*. Ou seja, implementar o ambiente colaborativo que XP assume requer administração disciplinada e infra-estrutura organizacional apropriada. Paulk desmistifica o argumento de que o CMMI (estatisticamente estável e rigoroso) é antitético a XP, pois este tem processos disciplinados e é claramente bem definido.

Paulk atinge o ápice de seu artigo ao considerar CMMI e XP "complementares". Recorda-se que as duas abordagens foram originalmente concebidas para atuarem em escopos e ambientes distintos. No entanto, Paulk conclui que o CMMI (com práticas gerenciais e organizacionais) "diz" às organizações *o que* fazer em termos gerais, mas não "diz" *como* fazer. XP consiste de um conjunto de práticas que contém informações de "como fazer" bastante específicas (implementação de processo) para um tipo particular de ambiente. Isto é, as práticas de XP podem ser compatíveis com as práticas do CMMI, mesmo não se referindo a estas diretamente. Deve-se encarar XP não como oposta ao CMMI, mas como inversa em seu discurso, encerrando um ideal de multiplicidade que permite a inserção da qualidade em empresas de pequeno e médio porte, ao enfatizar fatores não-técnicos e promover a melhoria da qualidade a partir da calibragem do processo, com base na experiência e na percepção dos *stakeholders* envolvidos (i.e., estabelecer um cenário socioténico). Isso também deve ser assimilado pelo CMMI, visando instigar e melhorar o tratamento desses fatores, em empresas de grande porte.

Por fim, Paulk encerra o artigo mencionando que as organizações que querem melhorar sua capacidade deveriam tirar proveito das vantagens e das boas idéias de XP e do CMMI, e exercitar o bom senso em selecioná-las e implementá-las. Mas ironiza, ao colocar em foco um aspecto crucial: "as organizações deveriam usar XP (da forma como foi definido) para sistemas que requerem alta confiabilidade e que ofereçam riscos à vida? *Provavelmente não*" (Paulk, 2001, p. 8, grifo nosso). Independente das possíveis combinações e similaridades entre os princípios e as práticas de XP e do CMMI, Paulk justifica sua resposta, inserindo dúvidas ao leitor quanto à confiabilidade de XP diante de tanta flexibilidade, e aponta qual seria o objetivo de uma decisão rumo à melhoria de processos de software. Com isso, pode-se depreender que Paulk impõe a XP responsabilidades que não lhe pertencem: suas práticas são oriundas de outras abordagens (muitas delas de modelos convencionais) e, no fundo, representam práticas de agilidade geral – e não simplesmente uma contribuição de XP.

> *A falta de documentação e de ênfase em arquitetura de XP é arriscada. Entretanto,* uma das virtudes de XP é que você pode modificá-la e melhorá-la para diferentes ambientes. *Isso quer dizer que, quando se modifica XP, você* arrisca perder as propriedades emergentes que agregam valor ao contexto apropriado. *Em última instância,* quando você escolhe melhorar processos de software, a ênfase deveria estar em deixar o bom senso prevalecer – *e* usar dados *para fornecer perspicácia* em relação a questões desafiadoras *(Paulk, 2001, p. 8,* grifo nosso*).*

3.2 Equilibrando agilidade e disciplina

Barry Boehm e Richard Turner afirmam que a rapidez das mudanças e o aumento da criticalidade do software levam as organizações de sucesso em desenvolvimento e aquisição de software a equilibrarem agilidade e disciplina

em seus processos-chave, dado que a emergência das metodologias ágeis na comunidade de Engenharia de Software ampliou as expectativas de clientes e de gerentes de projeto (Boehm e Turner, 2004). No entanto, essas metodologias têm pontos sensíveis e sua compatibilidade com métodos tradicionais (baseados em planos), como o CMMI, não é largamente explorada.

Boehm e Turner iniciam seu artigo "confrontando" as qualidades de cada abordagem (ágil e clássica), não se restringindo apenas à XP e ao CMMI: enfatizam, em abordagens ágeis, as promessas relativas a alto grau de satisfação do cliente, menores taxas de defeitos, *time-to-market* e soluções rápidas para mudanças de requisitos, e ressaltam, em abordagens baseadas em planos, as promessas de estabilidade, previsibilidade e garantia de segurança. No entanto, os autores apontam que ambas as abordagens têm negligências que, se não consideradas, podem conduzir a falhas em projetos (Boehm e Turner, 2004, p. 718). Essa discussão motiva a idéia dos autores de convergir as duas abordagens por meio da utilização equilibrada de suas práticas, mediante a identificação de cinco dimensões críticas que descrevem o *spectrum* ágil/dirigido a plano: *tamanho*, *criticalidade*, *dinamismo*, *pessoal* e *cultura*.

Diferentemente de Paulk (2001), que foca seu trabalho em analisar a peculiaridade e a instabilidade de XP, Boehm e Turner apontam que *a agilidade é a contrapartida da disciplina*, isto é, a disciplina cria uma história com memórias e experiências bem organizadas, e a agilidade aplica essas memórias (história) com foco na adaptabilidade diante de incertezas e situações inesperadas, atualizando a base de experiência para o futuro. Nessa linha, os autores realizaram dois estudos de caso, nos quais definiram cenários para que gerentes de projeto "criativos" misturassem as abordagens (desmistificando a idéia de oposição): um dos estudos de caso partiu de uma "base" ágil e o outro, de uma "base" dirigida a planos. Por outro lado, apesar de entenderem que a flexibilidade das metodologias ágeis requer a utilização de métodos dirigidos a riscos, os autores criticam a forma como se busca balancear as práticas das duas categorias de metodologias.

> *Abordagens de desenvolvimento ágeis e dirigidas a planos têm sido visualizadas geralmente como duas perspectivas opostas, e a* retórica envolvida ainda permanece essencialmente "confrontacional". *Os pontos de vista antagônicos, as discordâncias de representação e a habilidade dos defensores de cada abordagem* criam um senso de perplexidade naqueles profissionais (como nós) que simplesmente desejam completar os projetos com sucesso e agradar os clientes *(Boehm e Turner, 2004, p. 718,* grifo nosso*)*.

3.3 Agilidade como processo de desenvolvimento, modelos tradicionais como controle

Além das linhas de raciocínio de Paulk (2001) e de Boehm e Turner (2004), existem autores que sugerem XP como um processo de ciclo de vida ou como um regulador da produtividade das equipes de desenvolvimento que trabalham em ambientes "controlados". Estas duas "visões" são observadas nos trabalhos de Kalayci (2003) e Bartie (2007), respectivamente.

Kalayci (2003) afirma que o "mundo" do CMMI está interessado em "conhecer" e entender os processos de software, mas que a recíproca não é verdadeira, uma vez que os processos de software se preocupam mais com os resultados do negócio. O autor aponta XP como um "novo caminho" para o desenvolvimento de software e organiza seu texto da mesma forma que Paulk, separando os "mundos" das duas metodologias para uma posterior análise de possíveis similaridades e combinações. Nessa ocasião, Kalayci instiga o leitor com uma questão crucial: "existe algum conflito entre o CMMI e a definição e prática de XP?", e responde, "não necessariamente", justificando sua posição pelo seguinte argumento: "quando a organização quer otimizar a implementação de XP, o CMMI deve ser utilizado, ou seja, XP não consiste em uma alternativa a algum dos processos do CMMI. Esses processos existem para que um ambiente apropriado seja desenvolvido a fim de manter o ciclo de vida do software (XP) sob controle, definindo-o, medindo-o e otimizando-o" (Kalayci, 2003, p. 5). Esse argumento demonstra que o autor encara XP como um "instrumento" para o desenvolvimento de software, como o RUP (*Rational Unified Process*), o modelo cascata e o modelo espiral, o que expressa sua opinião de que uma postura mais adequada esteja em não relacioná-lo diretamente com abordagens tradicionais, como o CMMI.

Nessa mesma linha, Bartie (2007) externaliza sua opinião já no título do artigo: "Agilidade ou Controle Operacional? Os dois!". O autor também segue a estrutura de Paulk e Kalayci (apresentar cada uma das abordagens separadamente para depois pensar em uma análise de similaridades e combinações). Por ter formação em Administração, Bartie conduz o discurso sob uma perspectiva mercadológica, colocando as decisões acerca de qual abordagem seguir condicionadas à estratégia organizacional. Uma diferença observada em sua postura está no fato de que ele discute separadamente os pontos fracos dos modelos ágeis e critica o equivocado modelo "causa-efeito", estabelecido com relação à adoção de uma das abordagens. Ao propor o chamado "modelo combinado", entendido como a mistura entre agilidade e controle operacional "ao extremo", o autor afirma que: "a abordagem do modelo combinado não é tão diferente da abordagem do modelo

controlado. [...] A grande mudança [...] é definir como diretriz geral que qualquer inovação corporativa, que seja planejada na cadeia produtiva de TI [Tecnologia da Informação] e que acarrete uma perda de agilidade, será antecipadamente compensada por uma inovação que recupere ou aprimore o patamar de produtividade das equipes de trabalho" (Bartie, 2007, p. 27).

Percebe-se que Bartie deseja manter a essência tradicional (hierarquia, estrutura e padronização) e utilizar XP como um instrumento para monitorar a produtividade das equipes da organização, como Kalayci. É de se esperar que uma crítica surja nesse momento: é possível, em algum sistema, adotar abordagens que tenham algumas características tão distintas e peculiares a determinados tipos de cenário, *ao máximo* e *simultaneamente*? Isso realça a necessidade de maior exploração sobre os princípios de engenharia aplicados na indústria de software, a fim de evitar que posturas como essa contribuam para manter a distância entre as abordagens tradicionais e ágeis. Por outro lado, pode-se entender também o "modelo combinado", apresentado por Bartie, como um "motor" baseado na visão de que abordagens alternativas devem emergir de forma "suave" para alcançar "amplitude", ou seja, pregar a inserção de práticas de XP em cenários que implementam processos de CMMI, a fim de difundir seus princípios e valores, e visando propiciar aberturas e transformações.

3.4 Entender relacionamentos sem buscar universalismos

Jiang e Eberlein (2008) iniciam seu artigo expondo a presença e a força dos "conflitos" entre as metodologias clássicas da Engenharia de Software e as metodologias ágeis. No entanto, ao entenderem que o debate entre as metodologias é essencialmente uma disputa entre as filosofias dos seus modos ou estilos de operação em um processo de desenvolvimento, os autores apontam a exploração, não apenas das diferenças fundamentais entre as metodologias, mas também das suas similaridades, como uma idéia razoável. Isto é, "*metodologias clássicas e ágeis têm origens filosóficas comuns e são tecnicamente compatíveis e complementares?*".

Tentando responder a essa questão, Jiang e Eberlein propõem um *framework* na forma de um prisma que se divide em cinco dimensões, considerando: os conceitos envolvidos nas metodologias, o seu *background* histórico e tecnológico, as diferenças notórias entre elas, a variedade de habilidades e as condições econômicas, tecnológicas e organizacionais necessárias para executá-las. Para isso, os autores combinam cinco técnicas de análise de pesquisa: *contextual*, *histórica*, *análise por analogia*, *fenomenologia* e *lingüística*. O objetivo consiste em auxiliar gerentes e desenvolvedores a entenderem a natureza das metodologias da Engenharia de Software, de maneira mais concreta e em um nível tal que eles selecionem as melhores práticas e técnicas para um dado projeto de software.

Ao longo do artigo, percebe-se a preocupação dos autores em considerar as particularidades de cada cenário, onde um conjunto de práticas e princípios provenientes das metodologias deve ser aplicado, conforme a sua adequação ao ambiente (e não o contrário). A postura dos autores é clara: "discussões envolvendo essas metodologias seriam melhor utilizadas para sedimentar o entendimento de suas vantagens e limitações e de como utilizá-las, combiná-las e melhorá-las em processos de Engenharia de Software, mais do que para tentar provar se uma metodologia é universalmente superior a outra" (Jiang e Eberlein, 2008, p. 13). Por fim, os autores reclamam da inexistência de esforços em direção a um mecanismo que conduza à melhor decisão quanto à adoção de uma metodologia, em uma circunstância particular. Dessa forma, semelhantemente a Boehm e Turner (2004), Jiang e Eberlein defendem a combinação saudável das metodologias, em conformidade com as características do cenário de aplicação.

3.5 Observações finais

Observa-se, a partir das subseções anteriores, que existem diferentes pontos de vista sobre as relações entre as abordagens (metodologias) clássicas (CMMI) e ágeis (XP). Independentemente disso, chega-se a uma conclusão: cada abordagem tem seu cenário de atuação e a mistura de seus princípios e práticas sobre um cenário particular representa um passo importante para a Engenharia de Software atual, dinâmica e preocupada com questões econômicas. Entretanto, devido ao foco do presente trabalho, alguns pontos críticos de XP devem ser tratados com cuidado, tais como a flexibilidade, criticada por Paulk (2001), e a propensão a riscos, apontada por Boehm (2006).

Boehm (2006) apresenta relatos de que o aumento da escalabilidade de projetos pode comprometer a adoção de XP. Por outro lado, pode-se pensar na abordagem alternativa de XP e em sua abertura como um alerta de que a multiplicidade necessita de espaço, e isso não deve ser entendido como caótico. Não seria uma iniciativa de criação de um novo modelo (ao estilo do CMMI), mas o estabelecimento de um diálogo entre os discursos das duas abordagens, visando a sua convergência ao congregar um tratamento de fatores técnicos (CMMI) e não técnicos (XP) (Teixeira, 2006) e a evolução do processo organizacional baseada na experiência, compondo um cenário sociotécnico. Dessa forma, tendo a cooperação e a colaboração como princípios construtores da sociedade atual, pode-se almejar

uma nova forma de construção da história da Engenharia de Software, integrando ciência, engenharia, tecnologia e, sobretudo, sociedade.

4 Identificando elementos da abordagem sociotécnica de XP em casos práticos

Com o intuito de verificar as relações tecidas na seção 3, será apresentado um diálogo entre entrevistas[2] realizadas com um professor/pesquisador e com um analista de sistemas, que trabalham com XP, visando observar e explorar posturas, desejos e ansiedades diante dos valores, princípios e práticas do CMMI e de XP (Santos, 2008), que contribuem, de certa forma, para que o debate existente entre essas vertentes se mantenha vivo e interfira na essência de cada abordagem (ao estilo sociotécnico). Foram abordadas cinco questões nas entrevistas, cujas respostas, apesar de expressarem um "tom" tendenciosamente favorável a XP, têm o mérito de identificar fatores que levaram ambos os entrevistados a adotarem XP (já que eles trabalhavam anteriormente com práticas de SW-CMM e CMMI) e a refletirem sua concepção sobre processos de desenvolvimento de software diante da experiência: (i) *uso de XP*: "Qual a sua motivação para o uso de XP?"; (ii) *diferenças*: "Quais as diferenças notórias entre as práticas do CMMI e de XP?"; (iii) *observação*: "Quais os impactos e resultados da adoção de XP na prática, se comparados àqueles dos processos estabelecidos pelo CMMI?"; (iv) *clientes*: "Em relação aos clientes que utilizavam modelos tradicionais (CMMI), qual a sua percepção acerca das reações sobre a adoção de XP?"; e (v) *validação*: "Considerando diferentes contextos, XP é bom em todas as situações? Ou existem situações e cenários específicos que o favorecem e/ou o desfavorecem?".

Além das entrevistas, será considerada a "antítese" de Eric Raymond: *a catedral e o bazar*. Raymond (2001) discute teorias em termos de dois estilos fundamentais e diferentes de desenvolvimento de software: o modelo "catedral" (estilo de desenvolvimento de software do mundo comercial/proprietário) contra o modelo "bazar" (software livre e o "mundo" do Linux). Sua motivação vem da experiência pessoal, ao confrontar a forma tradicional de trabalhar com desenvolvimento e a flexibilidade e o dinamismo de projetos *open source*.

Ao verificar o cenário de desenvolvimento de software livre por meio do projeto do sistema operacional Linux e passar por uma reflexão, Raymond explica o choque que levou ao observar como uma forma tão "barulhenta" de trabalho poderia chegar tão longe e obter um sucesso relativamente atrativo. Para Raymond, o cenário tradicional do software proprietário – bem estruturado, estável e com documentação mais elaborada – trazia uma sensação de impotência diante do dinamismo e das mudanças do mundo atual, ao almejar ser universal e completamente estável (sem riscos).

No presente trabalho, a idéia será esboçar um paralelo entre o estilo "catedral" do software comercial e o "modelo universal" do CMMI – dado que têm em comum o ideal de "universalidade"– e entre o estilo "bazar" do software livre e as metodologias ágeis (XP, neste caso), por representarem abordagens alternativas. Dessa forma, as posturas, ansiedades e sensações de Raymond e dos entrevistados poderão ser mapeadas, a fim de compreender a necessidade de se valorizar cenários individuais e particulares em detrimento de generalizações e hierarquizações.

A curiosidade de Raymond o levou a questionar seu estilo tradicional de desenvolvimento diante do "mundo" do Linux e sua "explosão" como um bazar, e a demonstrar ansiedade para entender por que uma abordagem tão confusa não se dividia, mas aumentava sua força continuamente, "aos olhos de uma catedral" (Raymond, 2001).

> *Eu acreditava que os softwares mais importantes [...] precisavam ser* construídos como as catedrais, habilmente criados com cuidado por mágicos ou por pequenos grupos de magos trabalhando em esplêndido isolamento*, com nenhum 'beta' para ser liberado antes.* O estilo *[...] de desenvolvimento* – libere cedo e freqüentemente, delegue tudo que você possa, esteja aberto ao ponto da promiscuidade – *veio como* uma surpresa *(Raymond, 2001, p. 1,* grifo nosso*).*

A partir dessas observações, nas subseções a seguir, as posturas dos entrevistados serão exploradas, considerando a expressividade de suas respostas. Em alguns momentos, nota-se que sensações semelhantes às de Raymond (quando ele se defrontou com a percepção de características conflitantes entre o estilo de desenvolvimento de software proprietário e o livre) são percebidas nos entrevistados, diante da percepção de características conflitantes entre o estilo de desenvolvimento de software baseado em planos e o ágil. E isso contribuiu para que o diálogo entre essas diferentes vertentes de desenvolvimento emergisse e se mantivesse, causando impactos na definição dos valores, princípios e práticas de cada uma delas e, conseqüentemente, realçando a importância de

[2] As entrevistas ocorreram no dia 22/08/2007, entre 10h e 12h via Skype com o professor/pesquisador, e entre 15h e 20h na empresa do analista de sistemas.

um olhar sociotécnico acerca de elementos que compõem a Engenharia de Software.

4.1 O uso e as práticas de XP

A seguir, são exibidos alguns trechos importantes relativos à primeira questão da entrevista (a extensão dos trechos se deve à importância da observação dos detalhes para a continuação do diálogo). Assim como Raymond refletiu sobre seu estilo tradicional de desenvolvimento de software proprietário diante daquele do software livre, ambos os entrevistados resgataram de sua experiência com modelos controlados (principalmente fracassos e desgastes) a motivação para estudar e aplicar XP. A partir desse ponto, o leitor deve refletir a respeito dessas experiências e não entender o diálogo apenas como um posicionamento favorável à adoção de XP. Deve-se perceber que fatores não-técnicos representam a mola mestra de abordagens alternativas como XP: a necessidade de considerá-los é importante para a evolução da Engenharia de Software e de modelos tradicionais, como o CMMI. Tanto Raymond como os entrevistados demonstram uma forte ansiedade diante da observação de novas linhas de pensamento, em detrimento das linhas tradicionais, sobretudo ao "enxergarem" as promessas de sucesso.

> Professor: *Em 2000, comecei a estudar XP. Em 2001 comecei a lecionar disciplinas sobre XP e depois comecei a pensar na indústria.* Participei de consultorias e treinamentos *para a implantação de XP em empresas, juntamente com os nossos alunos, além de acompanhar o nascimento de empresas que iriam utilizar XP.* Orientei dissertações na área e atualmente participo de um projeto para um órgão do governo, que utiliza XP. *[...] Minha* motivação *principal se deu pelo fato de que* desenvolvo software há vinte anos – *nos dez primeiros, por meio de processos tradicionais. Mas* eu percebia que esses eram muito teóricos e, na prática, acabávamos os ignorando inconscientemente, pois era 'diferente'. *Foi quando,* em 2000*,* ouvi falar sobre XP – uma forma produtiva de desenvolvimento de software, que visava qualidade, e não documentação em demasia, sem hierarquia. Comecei a estudar e a tentar entender essa nova metodologia *(*grifo nosso*).*
>
> Analista de sistemas: Em meus tempos de graduação (estágios em empresas), percebia que todos os projetos eram catastróficos, com problemas não-técnicos *(peopleware) e desenvolvi* uma ferramenta web que se integrava com um ambiente de modelagem. Fiquei dois anos no mercado *e depois participei da* fundação de uma empresa *com soluções voltadas para design. A* empresa não ia bem nesse ramo *e acabamos nos voltando para a prestação de serviços. Decidi fazer* mestrado. *Em* conversas com meu orientador, sobre os fracassos nos projetos, ele me falou sobre *um 'tal' de* XP, um processo voltado para as pessoas, *e resolvi investigar. Comecei a me aprofundar na área e* adotar XP na empresa. *Tivemos um* projeto em uma empresa nacional de grande porte *que compôs uma parte de meus estudos na pós-graduação* (grifo nosso).

É evidente que aspectos teóricos e tradicionais do desenvolvimento de software contribuíram para que os entrevistados se sentissem ansiosos, e às vezes desestimulados, levando-os a adaptar processos tradicionais e tornando inevitável a iminência de alternativas; e XP aparece na forma de "murmurinhos" que aguçam a curiosidade dos entrevistados. E esse cenário não foi diferente na realidade de Raymond: ao ter uma maneira de testar a sua teoria (concretizar um projeto de código aberto), executou-o conscientemente ao "estilo bazar" e obteve sucesso significativo (Raymond, 2001, p. 1).

Analogamente à surpresa de Raymond diante do sucesso de um projeto de código aberto (o "despertar" para uma abordagem alternativa), os entrevistados experimentaram XP com sucesso em projetos de pesquisa e na indústria. Ao questioná-los sobre as diferenças notórias entre as práticas de XP e do CMMI (segunda questão), ambos focaram no princípio ágil de que *indivíduos e interações vêm antes de processos e ferramentas*. Ou seja, concedia-se maior importância a: reuniões diárias, programação em pares, *feedback* constante do cliente e iterações semanais. O professor acrescentou que os valores, princípios e práticas de XP contribuem para o ideal de agilidade e prosperam de forma semelhante a processos tradicionais, que agradam a gerentes e clientes, ao passo que o analista de sistemas comentou sobre o resultado "executável" (característica de XP), que elimina o desperdício ao se concentrar no necessário (*just in time*) e ao pensar no final da cadeia produtiva (software funcionando). O analista exemplificou, ainda, alguns pontos delicados do processo tradicional, como o problema de contratos com escopo fixo e o caso do *Microsoft Word*, onde a proporção entre as funcionalidades implementadas e aquelas efetivamente utilizadas é muito alta. Com isso, pode-se depreender que o analista toca na questão do planejamento do escopo do projeto, o qual muitas vezes se encontra distante da etapa de entrega do produto de software, em modelos tradicionais de desenvolvimento.

4.2 Impactos e reações diante da utilização de XP

Como tendência natural, as duas próximas questões da entrevista (terceira e quarta) estavam relacionadas à observação do uso de XP "na prática". Num primeiro momento, questionou-se sobre os impactos e as conseqüências do uso de XP no desenvolvimento de software. Na visão do professor, XP já existia na prática, apenas não rotulado e difundido como hoje, apresentando maior aceitação por empresas em fase de constituição. Por outro lado, o analista de sistemas aponta problemas de processos tradicionais e destaca as qualidades de XP e os pré-requisitos necessários para alcançá-las. Ambos mencionam que conhecem relatos de sucesso de XP, apesar de não entrarem em detalhes quanto às características dos projetos, tais como *tamanho*, *tipo* e *manutenção*. O que se pode perceber é que algumas ressalvas, ainda que tímidas, são apontadas pelos entrevistados, mesmo diante do "entusiasmo" que têm pela prática de XP. Mais especificamente na resposta do analista, verifica-se que os requisitos listados para o bom uso de XP refletem naturalmente os cenários de empresas de pequeno e de médio porte.

> Professor: Existem relatos do sucesso de XP, englobando melhoria de qualidade e de produtividade *em projetos de software. Entretanto, entendo que,* sem alguma prova científica, *é difícil se afirmar com completude os impactos de XP na prática geral. Nesse sentido, a* experimentação *controlada* se faz necessária. *[...] Devemos lembrar que,* antes do ano 2000, tais práticas *[ágeis]* já eram utilizadas*, apenas sem estarem suportadas por um processo ou por um rótulo, e não representam nada de novo.* Hoje, *já podemos considerar XP como uma* área de pesquisa*, na qual* um dos obstáculos é a mudança cultural*, sendo mais fácil surgir uma empresa que usa XP do que ocorrer a alteração dos processos de empresas existentes (*grifo nosso*).*

> Analista de sistemas: Conheço e li relatos de experiência sobre o aumento de qualidade e produtividade em projetos de software com XP. *Considero os* esforços para desenvolvimento de software maiores em processos tradicionais: *geração de vários artefatos que as pessoas se esquecem de ler; pensamento de que 'mudança é ruim'; e falta de compreensão do* cliente, *tido quase sempre como inimigo. XP busca resolver esses problemas. Mas enxergo* alguns pontos que devem ser pensados com cuidado. *A empresa deve cuidar para que exista um suporte para* programação em pares, *sem conflitos, e evitar a* inserção de muitos membros simultaneamente *nos projetos, independente de experiências anteriores, pois XP exige disciplina, ambiente integrado e que todos trabalhem com todos e 'em tudo' (*grifo nosso*).*

A preocupação seguinte consistiu em capturar a opinião dos participantes sobre a reação dos clientes que experimentaram projetos com XP, depois de experiências com processos tradicionais (CMMI). O professor se concentrou em dois casos reais aos quais ele esteve/está relacionado: projetos para uma empresa de telefonia e para um órgão do governo, além de ter apontado a necessidade/vantagem de se construir apenas documentação útil. O analista de sistemas, como ministrante de cursos para empresas e para clientes destas, enfatizou a redução de custos em projetos e a rapidez na entrega.

> Professor: *Conheço* o caso de uma empresa de telefonia *que sempre solicitava* projetos de software *para empresas que trabalhavam* com processos de desenvolvimento tradicionais, *onde havia um tempo de dois meses para a geração do contrato e mais seis ou oito meses para implementação. A empresa de telefonia* ficou espantada com a velocidade do projeto com XP e com a redução de custos. *Em nosso projeto* para um órgão público, *a documentação consiste de manual de usuário, documento arquitetural e testes automatizados (*grifo nosso*).*

> Analista de sistemas: *Percebo* em nossos clientes e em empresas parceiras que querem conhecer XP *uma expressão de* espanto diante da rapidez no desenvolvimento de projetos. *Além disso,* ministro cursos para a implantação de XP *e entendo* este processo como mais rápido e de menor custo. *A questão que* sempre enfatizo *é a* disciplina e o espírito de equipe *(*grifo nosso*).*

Pelas respostas, sente-se que XP precisa amadurecer rumo à validação de suas características, com estudos de caso e experimentos controlados que verifiquem o seu propósito, a sua real segurança e a sua confiabilidade. Na verdade, isso corresponde a uma realidade da Engenharia de Software como um todo (Pfleeger, 1999; Pressman, 2006). Conforme mencionado pelo professor, os clientes se espantam com a aparente redução do custo e do tempo de entrega dos produtos. O estilo incremental e iterativo proporciona a XP um acompanhamento constante (pelo cliente) do produto de software em desenvolvimento. O analista de sistemas não esquece de destacar a disciplina e o espírito de equipe como fundamentais para o sucesso com XP. Nesse momento, pode-se depreender

que existem requisitos e cenários que são propícios para a prática de XP, e deve-se cuidar para que ele não seja "vendido" como uma bala de prata (Travassos, 2007). Isso significa que o ato de enfatizar suas vantagens não encobre a existência de desvantagens e particularidades, e esses aspectos devem estar bem explícitos para que os valores, as práticas e os princípios de XP possam ser verificados, seja com relatos de sucesso ou de fracasso, e possam alertar para o fato de que as abordagens clássicas devem ser evoluídas com base na experiência e por meio de um olhar sociotécnico.

Outro aspecto importante consiste no uso de XP em projetos maiores, como para uma empresa nacional de grande porte (pelo analista de sistemas) e para um órgão do governo (pelo professor). O analista de sistemas comentou sobre um projeto nacional de viabilização de compras através de dispositivos móveis, implementado inteiramente com XP. Isso mostra que os desafios de escalabilidade em XP, apontados por Paulk (2001) e Boehm (2006), serão colocados à prova, de uma forma ou de outra, e faz-se necessária a capacidade de adaptação e flexibilidade de XP para contornar os riscos, porém, de forma estruturada e não *ad hoc*. Além disso, a valorização da caracterização de cenários específicos em detrimento da criação de "universalismos", destacada por Jiang e Eberlein (2008), deverá ser alvo de pesquisas, para que similaridades e combinações de valores, princípios e práticas, tanto de XP quanto do CMMI, contemplem diferentes cenários onde elas se aplicam, realimentando e "co-modificando" a essência dessas abordagens.

Tecendo a analogia entre XP e o "estilo bazar" do "mundo" do Linux, da mesma forma que Raymond (2001) entende que fatores não-técnicos representam o sucesso do Linux (abordagem alternativa), os entrevistados também entendem que esses fatores contribuem para as experiências de sucesso do uso de XP. Raymond ainda afirma que a escalabilidade é possível no "mundo" do software livre, dado que este "mundo" coexiste com o "estilo catedral" do "mundo" do software proprietário, sem distinção ou depreciação (o que pode ser verificado pela vasta utilização e sucesso do Linux). Para esse caso, deve-se recordar: a proposta de Paulk (2001), que contempla a possibilidade de se formar um processo "híbrido" com práticas de XP e do CMMI, devido à complementaridade das abordagens; as propostas de Kalayci (2003) e Bartie (2007), que exploram a inserção de práticas ágeis de XP no desenvolvimento de software em empresas que implementam processos do CMMI, almejando maior produtividade e o despertar "suave" de novos valores em cenários focados em modelos tradicionais; e a proposta de Boehm e Turner (2004), relativa ao cruzamento de valores, princípios e práticas de ambas as abordagens, visando um equilíbrio adequado em cada realidade.

4.3 Impressões acerca da validação de XP

Por fim, a última questão da entrevista busca entender se é possível ou não existir generalidade na aplicação de XP – um requisito para o estabelecimento dessa abordagem diante do "mundo" do CMMI. A perspectiva do professor reflete a necessidade de existência de provas de validade para XP, sobretudo no caso de sistemas críticos e de risco. Sob a ótica do analista de sistemas, depreende-se que não existem tantas ressalvas, mas um alerta de que alguns requisitos devem ser cumpridos para se alcançar a escalabilidade.

> Professor: O processo XP funciona para vários casos, exceto quando um cuidado maior deve ser tomado em relação ao domínio da aplicação *(e.g., sistema nuclear, piloto automático, sistemas que envolvam risco de vida). Diante disso, digo que são necessárias* provas matemáticas *de validade para XP. Mas o estado da arte atual não impede que sistemas sejam desenvolvidos e* camadas de segurança *sejam organizadas* para tratar requisitos de maior risco. *Outro ponto que pode necessitar da flexibilidade de XP* é a etapa de contrato*, que difere da forma tradicional (e.g., contrato por hora em XP, ao invés de contrato por especificação de funcionalidades).* Empresas públicas podem necessitar de documentação mais abrangente*, que contraria o proposto em XP; nesse caso, é de grande valia a* capacidade de adaptação presente em XP *(*grifo nosso*).*
>
> Analista de sistemas: Tecnicamente, não há uma situação em que XP não seria aplicável. *No entanto, XP* demanda mudança cultural, *uma vez que necessita de um* ambiente altamente colaborativo. *Isso pode ser* problemático *quando os desenvolvedores têm* espírito competitivo. *Além disso,* a má-qualificação pode ser um entrave, *pois XP exige disciplina e agilidade, onde todos se comunicam ao longo do desenvolvimento, e* todos conhecem todo o projeto. *Assim,* deve haver dinamismo *na qualificação profissional (*grifo nosso*).*

Segundo os entrevistados, XP poderia ser usado em vários cenários, mas eles não descartam alguns pontos críticos. Na verdade, como qualquer abordagem alternativa, que contempla fatores não-técnicos e que busca com a prática promover modificações em abordagens tradicionais, XP permite o surgimento de riscos dinamicamente, podendo inserir condições instáveis nos projetos, devido ao seu caráter flexível (como o "estilo bazar" do Linux). Talvez essa seja a questão mais explorada pelos adeptos

das práticas do CMMI quando se pensa em utilizar XP (como notado explicitamente no artigo de Paulk). Realmente, o "modelo universal" do CMMI contempla práticas, processos e resultados amadurecidos ao longo do tempo e que visam guiar, de maneira "não arriscada", o desenvolvimento de software.

Raymond (2001), ao final de sua narrativa, ressalta que *The Mythical Man-Month*, de Frederick P. Brooks, atribui práticas como o "estilo bazar" a Microsoft, mesmo que a sua comunidade interna de desenvolvimento seja estratificada ("estilo catedral"). Pode-se extrair disso o fato de que abordagens alternativas como XP ou o "mundo" do Linux, em sua iniciativa de semear liberdade e flexibilidade, trazem mudanças e reflexão no cerne das abordagens tradicionais do CMMI e do "mundo" do Windows (software proprietário), direta ou indiretamente, apontando a existência de um cenário sociotécnico. Verificou-se, também, essa questão no artigo de Paulk, em que o discurso do CMMI tenta envolver todo o espaço e conhecer qualquer abordagem alternativa, a fim de manter a sua autonomia e a sua importância no cenário acadêmico-empresarial de desenvolvimento. Por outro lado, a abertura para novas combinações e análises de similaridades, expressa por Boehm e Turner (2004) e por Jiang e Eberlein (2008), mostra que um debate divergente entre essas duas vertentes de desenvolvimento não deve ter espaço diante da evolução da Engenharia de Software rumo à dinamicidade da indústria, à criticalidade dos novos domínios e à produção de software de qualidade.

Percebe-se que o professor e o analista de sistemas seguem com a iniciativa e a reflexão pós-adoção de XP, mantendo o discurso que norteou o seu estabelecimento (o Manifesto Ágil) e contribuindo para a sua verificação. A lição que pode ser extraída aqui, considerando os paralelos feitos entre CMMI/XP e Windows/Linux, é que o *peopleware* representa um fator impactante sobre projetos de software que necessita e "reclama" consideração, e que propicia diretamente a "co-modificação" e a evolução dessas abordagens em sua essência técnica (passível de observação por um olhar sociotécnico). Além disso, os entrevistados entendem a importância do CMMI e pensam na integração de valores, práticas e princípios de XP com o CMMI, mas sem perder de vista a sua motivação em buscar uma alternativa em relação às práticas e aos processos de modelos tradicionais: *um "tecido" de fatores técnicos e não técnicos, sem costuras*.

5 Considerações finais

Como em todo sistema dinâmico, sabe-se da importância de uma *baseline* que oriente a escolha de qual vertente de desenvolvimento de software (ou a combinação de práticas e princípios) é melhor aplicada a cada cenário (considerando as suas particularidades e as características locais), para que valores de XP e/ou do CMMI sejam utilizados na indústria, de modo a maximizar a obtenção de seus benefícios potenciais. Entretanto, voltando à questão filosófica do processo, é interessante entender como o ideal de "modelos universais", baseado em técnica e em engenharia, dominou a realidade e o pensamento ocidental, com o objetivo de "manter a ordem". Dada a existência de possíveis similaridades e combinações entre princípios e práticas do CMMI e de XP, deve-se cuidar para que este não perca sua origem e sua essência, mas que cumpra seu papel de alertar para a necessidade de se considerar a composição de fatores técnicos e não-técnicos no desenvolvimento de software de qualidade, além da evolução das abordagens, instigada e impulsionada pela experiência e prática dos *stakeholders* do processo.

XP aparece como uma abordagem alternativa, sem a intenção de "derrubar" modelos tradicionais, como o CMMI, mas visando questionar o "clássico", abrir caminhos e espaços, chamar a atenção para a combinação de algumas práticas que podem explorar a agilidade, mesmo sem um conhecimento suficiente ainda sobre as suas limitações, estabelecendo e difundindo novos grupos, participantes e realidades (alternativas também se evidenciam em programas como o MPS.BR, que tem organização mais leve que o CMMI). O objetivo não está no caos, mesmo porque a intenção é a simbiose e o tratamento particular de cenários distintos.

Conforme Paulk (2001), XP exige disciplina e exibe um processo bem definido, que o deixa frágil pela dependência do contexto e da maturidade dos princípios e dos *stakeholders* envolvidos. Pode-se dizer ainda que XP surgiu como uma alternativa diante da Engenharia de Software mais tradicional. No entanto, assim como o ideal de abertura da comunidade do Linux em relação ao "mundo" do Windows, XP pode ser entendido como uma abordagem que tenta alertar o "mundo" do CMMI quanto à importância do "particular" perante o "universal". Aponta-se XP e CMMI como passíveis de combinação em seus princípios e práticas, despertando a reflexão e reavaliando processos tradicionais que, devido aos custos, acabam mantendo o monopólio de empresas que podem implantar processos mais robustos. Ao apontar uma alternativa para que pequenas e médias empresas consigam construir seu espaço, surgem iniciativas, como a de Paulk, que propõem o "casamento" entre diferentes abordagens, "quebrando" o tradicional "estilo catedral" de pensar, projetar e agir.

Por outro lado, iniciativas que consideram misturas entre vertentes de desenvolvimento clássicas e ágeis como base para o sucesso de projetos de software, expondo as peculiaridades de cada cenário de aplicação, ganham destaque e importância, como em (Boehm e Turner, 2004) e em

(Jiang e Eberlein, 2008). Um debate "conflitante" não deve ter tanto espaço, de maneira que a pesquisa experimental rumo à melhoria de processos de software ganhe cada vez mais importância, analisando a combinação mútua entre fatores técnicos e não-técnicos. Além disso, a inserção "suave" de práticas de XP em cenários tradicionais, conforme Bartie (2007) e Kalayci (2003), é interessante e importante para a evolução da Engenharia de Software rumo à experimentação, se interpretada como um mecanismo de "quebra" da monopolização e da "perfeição", por vezes embutida em modelos tradicionais; ou seja, deve primar por sua renovação.

Como contribuição, este trabalho mostra o surgimento de XP na Engenharia de Software e no cenário do CMMI, por meio de algumas entrevistas e do esboço de um paralelo com a "antítese" de Raymond. Verifica-se a ansiedade dos participantes do diálogo construído, enxergado a partir de uma abordagem catedrática (CMMI) e do jogo de idéias lançado por Paulk (2001), Boehm e Turner (2004), Jiang e Eberlein (2008), Bartie (2007) e Kalayci (2003), diante do crescimento e evolução de XP, ao "estilo bazar" do "mundo" do Linux. A reflexão esperada não está na simples visão individual do cenário de XP diante do CMMI, mas na percepção do leitor acerca da construção e reconstrução do pensamento, mostrando que fatores não-técnicos são tão importantes quanto fatores técnicos e que ambos agem em conjunto como um "motor" para a reconstrução da tecnologia, "vistos" sob um olhar sociotécnico. Mesmo com as limitações deste trabalho, relativas ao número de entrevistados e ao tratamento de caráter inicial, o texto cumpre o seu papel de possibilitar e instigar reflexões iniciais acerca da construção e evolução de modelos "universais" perante os "alternativos", por meio de uma abordagem sociotécnica.

Surge, então, uma pergunta: *Com a evolução das tecnologias e dos processos, diante de sistemas cada vez mais escaláveis, necessitados de robustez e confiabilidade e passíveis de riscos, existe espaço para discussão e reflexão de qual "mistura ideal" de abordagens agrega mais valor e qualidade ao produto de software?* Resposta: *Não!* Conforme ressaltado por Boehm e Turner (2004) e Jiang e Eberlein (2008), o futuro da Engenharia de Software não será calcado na repetição de universalismos, ou seja, a mistura "ideal" ou "universal" não existe, dado que cada caso é um caso. A teoria subjacente a modelos de processo de software precisa evoluir de uma visão de mundo "universal", "geral", "eterna" e "escrita", para uma comunhão desta com uma visão "particular", "local", "presente" e "oral" (Boehm, 2006). Uma teoria recente de Engenharia de Software Baseada em Valor (*Value-Based Software Engineering*) e processos de software relacionados fornecem um ponto de partida para se referir a esse desafio (Biffl *et al.*, 2006).

Por fim, este trabalho abre espaço para novas oportunidades de pesquisa, uma vez que, neste momento, procurou-se identificar elementos para uma abordagem sociotécnica do desenvolvimento de software com XP. Um passo seguinte consiste em mapear melhor a eventual controvérsia discutida (entre elementos de XP e do CMMI), "ouvindo" outras "vozes", especialmente dos defensores do CMMI ou de outros modelos tradicionais, como a ISO – possivelmente buscando entender como esses modelos tradicionais se relacionam e se combinam (Rout e Tuffley, 2007) –, ou ainda agregando as contribuições dos defensores do programa MPS.BR – que tem organização mais leve que o CMMI e mais voltada à realidade nacional. Deve-se ressaltar que Paulk, em seu artigo, acena com um "diálogo" do "mundo" do CMMI em relação ao "mundo" de XP. Entretanto, os entrevistados não acenam com um "diálogo" do "mundo" de XP em relação ao "mundo" do CMMI, o que leva à necessidade de uma pesquisa mais ampla que busque por "vozes" que acenem com este diálogo, se existirem.

Seria interessante identificar elementos da abordagem sociotécnica do CMMI em casos práticos (neste trabalho, isso foi feito apenas para XP, o que corresponde a uma limitação), ou seja, a visão de especialistas que se deparam com situações nas quais as práticas ágeis não seriam suficientes. Além disso, tentaria se explorar a experiência própria da Engenharia de Software, com o desenvolvimento e observação de fatores técnicos e não-técnicos em estudos de caso, em ambientes "tradicionais" que incorporam características de XP e vice-versa. A "antítese" catedral/bazar, proposta por Raymond e utilizada neste trabalho para analisar as diferenças entre valores, princípios e práticas do CMMI e de XP, é muito interessante. Porém, a discussão que articula essas vertentes à oposição software proprietário/software livre foi apenas esboçada e merece ser mais aprofundada.

Agradecimentos

O autor agradece ao CNPq pelo apoio financeiro na realização deste trabalho e ao Professor Henrique Cukierman pelas contribuições e valiosos ensinamentos.

Referências

AUSTIN, R.; DEVIN, L. 2003. Beyond Requirements: Software Making As Art. *IEEE Software*, **20**(1):93-95.

BARTIE, A. 2007. Agilidade ou Controle Operacional? Os dois! *Engenharia de Software Magazine*, **1**(1):22-27.

BECK, K. 2000. *Extreme Programming Explained: Embrace Change*. Boston, Addison-Wesley, 224 p.

BECK, K.; BEEDLE, M.; BENNEKUM, A.; COCKBURN, A.; CUN-

NINGHAM, W.; FOWLER, M.; GRENNING, J.; HIGHSMITH, J.; HUNT, A.; JEFFRIES, R.; KERN, J.; MARICK, B.; MARTIN, R.C.; MELLOR, S.; SCHWABER, K.; SUTHER-LAND, J.; THOMAS, D. 2001. *Manifesto for Agile Software Development*. Disponível em: http://www.agilemanifesto.org. Acessado em: 01/10/2008.

BIFFL, S.; AURUM, A.; BOEHM, B.; ERDOGMUS, H.; GRÜN-BACHER, P. 2006. *Value-Based Software Engineering*. Berlin, Springer-Verlag, 388 p.

BOEHM, B. 2002. Get Ready for Agile Methods, with Care. *IEEE Computer*, **35**(1):64-69.

BOEHM, B. 2003. Value-Based Software Engineering. *Software Engineering Notes*, **28**(2):1-12.

BOEHM, B. 2006. A View of 20th and 21st Century Software Engineering. *In:* INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, 28, Shanghai, China, 2006. *Anais*… Shanghai, ICSE, p. 12-29.

BOEHM, B.; TURNER, R. 2003. *Balancing Agility and Discipline*. 1ª ed., Boston, Addison-Wesley, 304 p.

BOEHM, B.; TURNER, R. 2004. Balancing Agility and Discipline: Evaluating and Integrating Agile and Plan-driven Methods. *In:* INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, 26, Escócia, 2004. *Anais*... Escócia, ICSE, p. 718-719.

CMMI. 2006. *Capability Maturity Model Integration Version 1.2*. Disponível em: http://www.sei.cmu.edu/cmmi/models/index.html. Acessado em: 01/10/2008.

CHRISSIS, M.; KONRAD, M.; SHRUM, S. 2003. *CMMI: Guidelines for Process Integration and Product Improvement*. Boston, Addison-Wesley, 688 p.

CUKIERMAN, H.L.; TEIXEIRA, C.; PRIKLADNICKI, R. 2007. Um Olhar Sociotécnico sobre a Engenharia de Software. *Revista de Informática Teórica e Aplicada*, **14**(2):207-227.

DAHLBOM, B.; MATHIASSEN, L. 1993. *Computers in Context: The Philosophy and Practice of Systems Design*. Oxford, NCC Blackwell, 306 p.

EBERLEIN, A. 2003. Requirements Engineering and Agile Methods: Can they benefit from each other? *In:* CANADIAN INVITED WORKSHOP ON SCALING XP/AGILE METHODS, Banff, 2003. *Position Statement*, Banff.

ECKSTEIN, J. 2006. Agile Software Development in the Large. *In:* INTERNATIONAL CONFERENCE ON OBJECT-ORIENTED PROGRAMMING, SYSTEMS, LANGUAGES, AND APPLICATIONS, 21, Portland, 2006. *Tutorial*, Portland.

EDWARDS, P.N. 1996. *The Closed World: Computers and the Politics of Discourse in Cold War America*. Massachussetts, MIT Press, 462 p.

FERREIRA, R.B.; LIMA, F.P.A. 2006. Metodologias Ágeis: Um Novo Paradigma de Desenvolvimento de Software. *In:* WORKSHOP UM OLHAR SOCIOTÉCNICO SOBRE A ENGENHARIA DE SOFTWARE, II, Vila Velha, 2006. *Anais*... Vila Velha, WOSES/SBQS, p. 107-114.

FONSECA FILHO, C. 1999. *História da Computação – Teoria e Tecnologia*. São Paulo, LTr, 189 p.

FRITZSCHE, M.; KEIL, P. 2007. Agile Methods and CMMI: Compatibility or Conflict? *e-Informatica Software Engineering Journal*, **1**(1):9-26.

GLASS, R.L. 2001. Agile Versus Traditional: Make Love Not War. *Cutter IT Journal*, **14**(12):12-18.

JIANG, L.; EBERLEIN, A. 2008. Towards a Framework for Understanding the Relationships between Classical Software Engineering and Agile Methodologies. *In:* INTERNATIONAL WORKSHOP ON SCRUTINIZING AGILE PRACTICES OR SHOOT-OUT AT THE AGILE CORRAL, 1, Leipzig, 2008. *Anais*… Leipzig, ICSE, p. 9-14.

KALAYCI, O. 2003. CMMI versus XP. *In:* WORKSHOP IMPACT OF SOFTWARE PROCESS ON QUALITY (IMPROQ'2003), 1, Ankara, 2003. *Anais*...Ankara, p. 1-5.

KITCHENHAM, B.A.; PFLEEGER, S.L.; PICKARD, L.M.; JONES, P.W.; HOAGLIN, D.C.; EL EMAM, K.; ROSENBERG, J. 2002. Preliminary Guidelines for Empirical Research in Software Engineering. *Transactions on Software Engineering*, **28**(8):721-734.

LIPPERT, M.; BECKER-PECHAU, P.; BREITLING, H.; KOCH, J.; KORNSTÄDT, A.; ROOCK, S.; SCHMOLITZKY, A.; WOLF, H.; ZÜLLIGHOVEN, H. 2003. Developing Complex Projects Using XP with Extensions. *IEEE Computer*, **36**(6):57-66.

PAETSCH, F.; EBERLEIN, A.; MAURER, F. 2003. Requirements Engineering and Agile Software Development. *In:* IEEE INTERNATIONAL WORKSHOPS ON ENABLING TECHNOLOGIES: INFRASTRUCTURE FOR COLLABORATIVE ENTERPRISES, Linz, 2003. *Anais*… Linz, WETICE, p. 308-313.

PAULK, M.C. 2001. Extreme Programming from a CMM Perspective. *IEEE Software*, **18**(6):19-26.

PFLEEGER, S.L. 1999. Albert Einstein and Empirical Software Engineering. *IEEE Computer*, **32**(10):32-38.

PRESSMAN, R. S. 2006. *Software Engineering: A Practitioner's Approach*. 6ª ed., Nova Iorque, McGraw-Hill, 720 p.

RAYMOND, E.S. 2001. *The Cathedral & The Bazaar: Musings on Linux and Open Source by Accidental Revolutionary*. Cambridge, O'Reiley, 279 p.

REIFER, D.J. 2003. XP and the CMM. *IEEE Software*, **20**(3):14-15.

ROUT, T.P.; TUFFLEY, A. 2007. Harmonizing ISO-IEC 15504 and CMMI. *Software Process: Improvement and Practice*, **12**(4):361-371.

SANTOS, R.P. 2008. Uma Visão Sociotécnica sobre o Desenvolvimento de Software com *Extreme Programming*. *In:* WORKSHOP UM OLHAR SOCIOTÉCNICO SOBRE A ENGENHARIA DE SOFTWARE, IV, Florianópolis, 2008. *Anais*... Florianópolis, WOSES/SBQS, p. 25-36.

SATO, D.T. 2007. *Uso Eficaz de Métricas em Métodos Ágeis de Desenvolvimento de Software*. São Paulo, SP. Dissertação de Mestrado. Universidade de São Paulo – USP (IME), 139 p.

SOFTEX. 2007. *MPS.BR – Melhoria de Processo do Software Brasileiro – Guia Geral – versão 1.2*. Sociedade SOFTEX, 2007. Disponível em: http://www.softex.br/portal/mpsbr/_guias/default.asp. Acessado em: 01/10/2008.

TEIXEIRA, C.A.N. 2006. Algumas Observações sobre os Vínculos entre a Engenharia de Software e o Pensamento Moderno. *In:* WORKSHOP UM OLHAR SOCIOTÉCNICO SOBRE A ENGENHARIA DE SOFTWARE, II, Vila Velha, 2006. *Anais*... Vila Velha, WOSES/SBQS, p. 39-50.

TEIXEIRA, C.A.N.; CUKIERMAN, H.L. 2007. Por que Falham os Projetos de Implantação de Processos de Software?. *In:* WORKSHOP UM OLHAR SOCIOTÉCNICO SOBRE A ENGENHARIA DE SOFTWARE, III, Porto de Galinhas, 2007. *Anais*... Porto de Galinhas, WOSES/SBQS, p. 1-12.

TELES, V. M. 2005. *Um Estudo de Caso da Adoção das Práticas e Valores do Extreme Programming*. Rio de Janeiro, RJ. Dissertação de Mestrado. Universidade Federal do Rio de Janeiro – UFRJ (IMNCE), 179 p.

TRAVASSOS, G.H. 2007. From Silver Bullets to Philosophers' Stones: Who wants to Be Just an Empiricist? *In:* V.R. BASILI; D. ROMBACH; K. SCHNEIDER; B. KITCHENHAM; D. PFHAL; R.W. SELBY (orgs.), *Lecture Notes in Computer Science 4336 – Empirical Software Engineering Issues Critical Assessment and Future Directions*. Berlin, Springer-Verlag, v. 4336, p. 39.

TURNER, R.; JAIN, A. 2002. Agile Meets CMMI: Culture Clash or Common Cause? *In:* D. WELLS; L. WILLIAMS (eds.), *Lecture Notes in Computer Science*. Londres XP/Agile Universe, p. 153-165.

Submitted on July 24, 2008.
Accepted on September 27, 2008.