MÉTODOS ÁGEIS DE DESENVOLVIMENTO DE SOFTWARE E O ATENDIMENTO AO MODELO CMMI

METHODS OF AGILE SOFTWARE DEVELOPMENT AND SERVICE MODELCMMI

Camila Tais Conti[1]
Fabiano André Trein[2]

RESUMO: A melhoria dos processos cada vez mais é reconhecida como uma forma de capacitar organizações a melhorar o seu potencial, buscando, de forma planejada, novas metas. O mesmo acontece com as organizações desenvolvedoras de *software* e seus processos, onde diversos métodos e modelos, inclusive reconhecidos mundialmente, surgem continuamente com o objetivo de auxiliar no desenvolvimento de produtos com maior qualidade. Sabe-se ainda que a escolha do modelo, ou modelos, de referência adequada à realidade e necessidade da organização é um dos fatores para o sucesso. Assim, este trabalho teve o propósito de promover o conhecimento sobre alguns dos métodos ágeis existentes, uma vez que esses estão cada vez mais em alta, sendo foco comum entre os profissionais de Tecnologia da Informação (TI) em seminários, fóruns, redes sociais, entre outros, além de avaliar quanto satisfazem o modelo *Capability Maturity Model Integration* (CMMI) nível 3 de maturidade, permitindo concluir que é possível, sim, atender aos critérios desse modelo, o qual é referência mundial para o desenvolvimento de *software* de qualidade, através da adoção de práticas ágeis.
Palavras-chave: Desenvolvimento de *Software*. Melhoria de Processos. Métodos Ágeis.

ABSTRACT: Process improvement is increasingly recognized as way organizations to improve their potential and find new goals. The same applies to software development organizations and processes, where various methods and models, includingrecognized world continually to help in the development of quality products. It is known that choice of model, or models, appropriate reference to reality and necessity of the organization is one factor in the success. Thus, this work had the purpose to promote knowledge on some of the existingagile methods, since these are increasingly high, and common focus among the professionals of Information Technology (IT) seminars, forums, social networks , among others, and assess how these meet the model Capability Maturity model Integration (CMMI) maturity level 3, allowing to conclude that it is possible to meet the criteria of this model, which is a world reference for the development ofsoftware quality through adopting agile practices.
Keywords: Software Development. Process Improvement. Agile.

[1] Especialista em Qualidade de Produtos e Processos e Graduada em Sistemas de Informação pela Feevale. Analista de Processos de Desenvolvimento de *Software* na PUC-RS. *E-mail*: camilaconti@gmail.com

[2] Doutorando PPGQA - Universidade Feevale. Mestre em Engenharia de Produção PPGEP UFRGS. Professor e Coordenador Universidade Feevale. E-mail: fabiano@duvinil.com.br

INTRODUÇÃO

A busca por melhores processos tende a ser uma meta estável e cada vez mais real, uma vez que: segundo a Associação Brasileira das Empresas de *Software* (ABES), a necessidade em desenvolver *software* de qualidade tem se tornado uma realidade cada vez mais presente no mercado brasileiro de *software*. Isso faz com que empresas desenvolvedoras estejam em contínuo aprimoramento de seus processos e métodos, para que possam atender a demanda do mercado e satisfazer os clientes de forma precisa, personalizada e eficaz. "O mercado brasileiro de software e serviços manteve a 12ª posição no cenário mundial, tendo movimentado 15 bilhões de dólares, equivalente a 0,96% do PIB brasileiro de 2008. Deste total, foram movimentados 5 bilhões em software, perto de 1,68% do mercado mundial". (ABES, 2009)

Essa busca contínua por melhores processos e produtos de qualidade motiva o aparecimento de uma série de métodos com o objetivo de auxiliar na conquista de projetos de desenvolvimento de sucesso. Nesse contexto, em 2001, foi realizado o Manifesto Ágil, no qual um conjunto de valores e princípios foram firmados por: desenvolvedores experientes, consultores e líderes da comunidade de desenvolvimento de *software* (BECK, 2001).

Os valores e princípios firmados no manifesto abriram caminho para um novo pensamento relacionado ao desenvolvimento de *software*, talvez como uma reação aos métodos tradicionais caracterizados por um forte planejamento e com uma série de resultados negativos, em que, conforme Bartié (2002),

-mais de 30% dos projetos são cancelados antes de serem finalizados;

-mais de 70% dos projetos falham nas entregas das funcionalidades esperadas;

-os custos extrapolam em mais de 180% os valores originalmente previstos;

-os prazos excedem mais de 200% os cronogramas originais.

Contudo, o mercado de TI ainda mostra algum receio sobre os métodos ágeis, talvez por se imaginar que desenvolver com uma abordagem ágil seja o mesmo que abandonar todo e qualquer planejamento. Considerando esse cenário, o problema a que se quer responder é: a melhoria dos processos de desenvolvimento, através de métodos ágeis, é possível? Pergunta com que talvez diversas organizações se depararem ao decidirem trabalhar a melhoria de seus processos de desenvolvimento.

Para responder a essa pergunta, alguns dos métodos ágeis de desenvolvimento de *software* serão estudados e analisados em relação ao atendimento das práticas propostas pelo modelo CMMI nível 3 de maturidade, modelo que é referência mundial para o desenvolvimento de *software*.

Para tanto, o trabalho está distribuído em três seções, as quais são apresentadas resumidamente a seguir. A Seção 1 aborda o processo de desenvolvimento de *software*, o modelo CMMI referência mundial em desenvolvimento de *software* e base para avaliação dos modelos ágeis também apresentados nessa seção. A seção 2 detalha o método utilizado no trabalho. Por fim, com base nos dados dispostos nas seções anteriores, a seção 3 compreende a análise e interpretação dos métodos ágeis e o CMMI.

1 O PROCESSO DE *SOFTWARE*

De acordo com Watts Humphrey, citado por Chrissis, Konrad e Shrum (2008), criador do programa de melhoria de processos de *software* do *Carnegie Mellon University*, o CMMI, o processo de *software* é o conjunto de atividades de engenharia, esforços técnicos e gerenciais necessários para transformar um conjunto de necessidades, expectativas e restrições de cliente em uma solução, conforme representado na figura 1, assim como o suporte a essa solução ao longo da sua vida e a referência necessária aos profissionais para o seu desenvolvimento.

Figura 1 – Processo de Software
Fonte: Criação nossa a partir de 2011.

A engenharia de *software* compreende três elementos fundamentais: os métodos, que definem "como fazer", as ferramentas, que apoiam na realização dos métodos, e os processos, ou ainda procedimentos, que unem métodos e ferramentas e definem a sequência de aplicação desses, além de servir como referência para avaliar o progresso do desenvolvimento. A relação entre esses elementos

compõe as etapas do desenvolvimento, muitas vezes, definidas como paradigmas de engenharia.

Segundo Pressman (2010), os paradigmas mais amplamente conhecidos são: Ciclo de Vida Clássico ou Cascata; Prototipação; Modelo Espiral e Modelo Incremental. Contudo, o autor defende ainda que o processo de desenvolvimento de *software* possui três fases genéricas independentemente do paradigma de engenharia. São elas:

-definição: responder "o que" deve ser feito;

-desenvolvimento: trabalhar em "como" o *software* será realizado;

-manutenção: trabalhar as mudanças necessárias ao *software*, por meio da definição de "o que" e "como" citados anteriormente.

Segundo o *Software Engineering Institute* (SEI), citado por Chrissis; Konrad e Shrum (2008), ao focar em processos, obtêm-se os fundamentos necessários para enfrentar as constantes mudanças impostas pelo mercado, maximizar a produtividade das pessoas e o uso da tecnologia, assim como promover um ambiente adequado para a introdução de novos procedimentos e tecnologias alinhados aos objetivos estratégicos da organização.

A definição de qualidade segundo alguns autores:

-Sommerville (2003): qualidade é definir uma cultura, na qual todos os responsáveis pelo desenvolvimento do produto possuem um nível de qualidade individual a atingir.

-Pressman (1995): "Conformidade a requisitos funcionais e de desempenho explicitamente declarados, a padrões de desenvolvimento claramente documentados e a características implícitas [...]" (PRESSMAN, 1995, p. 724).

-Bartié (2002): "[...] um processo sistemático que focaliza todas as etapas e artefatos produzidos com o objetivo de garantir a conformidade de processos de produtos, prevendo e eliminando defeitos" (BARTIÉ, 2002, p. 16).

Sendo assim, para que se tenha *software* de qualidade, é necessário investir no seu processo de desenvolvimento. A definição desses processos requer que os aspectos culturais e organizacionais sejam cuidadosamente considerados, uma vez que esses são tão ou mais importantes que os aspectos técnicos, sendo essenciais ao sucesso do projeto e à efetiva melhoria no trabalho realizado (ROCHA; MALDONADO; WEBER, 2001).

1.1 MELHORIA DO PROCESSO

Em 1990, a preocupação com o processo de desenvolvimento de *software* e sua melhoria ganhou força, o objetivo era produzir *software* de qualidade atendendo aos prazos e orçamentos definidos. Foi nesse momento que começaram a surgir algumas normas e modelos com o objetivo de auxiliar a melhoria (GUERRA; COLOMBO, 2009).

A melhoria do processo visa, principalmente, a reduzir a probabilidade de erro, aumentar a produtividade e facilitar a manutenção, devendo ser concentrada nas atividades que produzem mais problemas e exigem um esforço maior. Para isso, algumas etapas são sugeridas, conforme Côrtes e Chossi (2001): analisar da situação atual; experimentar e avaliar de novas propostas; implantar as propostas aprovadas para toda a equipe.

A melhoria deve ainda considerar as tecnologias envolvidas, o tipo do *software*, seu domínio de aplicação, a maturidade da equipe e as características da organização, do projeto e da equipe (ROCHA; MALDONADO; WEBER, 2001), além de trabalhar cuidadosamente os pilares que mantêm todo e qualquer processo, ou seja: pessoas com competências, procedimentos e métodos, ferramentas e equipamentos, conforme representado na figura 2.

Figura 2 – As Três Dimensões Críticas da Organização
Fonte: (CHRISSIS; KONRAD; SHRIM, 2008, p. 4).

Os itens pessoas com competências, procedimentos e métodos e ferramentas e equipamentos são necessários a qualquer processo com o mesmo grau de importância.

Para auxiliar as empresas desenvolvedoras a

obter a qualidade almejada em seus processos e produtos, foi criada uma série de modelos, conforme já mencionado, esses em duas linhas distintas, os tradicionais, ou prescritivos, e os ágeis. Os tradicionais são caracterizados por prezarem por uma quantidade considerável de documentação, enquanto as ágeis prezam por ter o *software* funcionando e com foco menor em documentação, despertando um grande interesse entre as comunidades de desenvolvimento de *software* (FOWLER, 2005), além de terem surgido como uma forma de oposição aos métodos tradicionais e ganharam força a partir do Manifesto Ágil, realizado em 2001, no qual um conjunto valores e princípios para o desenvolvimento foi firmado (BECK, 2001).

1.2 CMMI

Desenvolvido pelo SEI, centro de pesquisa da *Carnegie Mellon University*, o CMMI é um modelo de qualidade e maturidade reconhecido mundialmente, que propõe melhoria contínua nos processos de desenvolvimento de *software*. Esse modelo confere às empresas que atendem suas práticas um nível de maturidade ou capacidade gradativo, como um selo de garantia à qualidade do processo e produto desenvolvido, facilitando e padronizando a identificação de empresas capacitadas conforme os critérios do modelo CMMI ao desenvolvimento de *software*. Esse é um critério considerado relevante, uma vez que o número de empresas desenvolvedoras é cada vez maior, mas que, de forma alguma, deve ser definitivo. Outro fato interessante no atendimento do modelo CMMI é que alguns editais de desenvolvimento de *software*, inclusive do governo brasileiro, consideram como pré-requisito o atendimento de algum nível de maturidade ou capacidade desse.

O modelo de maturidade CMMI é um *framework*[3] que acomoda múltiplas disciplinas de modo flexível, o que o permite suportar duas representações: por estágios e contínua.

-Representação por Estágios: busca verificar o nível de maturidade da organização em geral, onde as áreas de processo (PA) são distribuídas em cinco níveis de maturidade e trabalhadas em conjunto para a constatação da maturidade daquele nível (CHRISSIS; KONRAD; SHRUM, 2008).

-Representação Contínua: visa à melhoria da capacidade dos processos isoladamente, evoluindo cada PA individualmente (COUTO, 2007).

-O modelo é composto por 22 PAs, com objetivos, metas específicas, práticas específicas, produtos de trabalhos típicos e subpráticas (CHRISSIS; KONRAD; SHRUM, 2008), não são definidos atividades, indicadores ou artefatos do processo desenvolvimento para o atendimento de uma determinada PA, o que faz necessário que o modelo seja estudado, compreendido e adaptado às características de cada empresa.

A utilização do modelo CMMI como referência permite que os processos sejam planejados e implantados em conformidade com os componentes esperados e requeridos das áreas de processo, garantindo uma evolução contínua e planejada do processo de *software*.

1.3 SCRUM[4]

Conforme Pressman (2010), é um método ágil para gestão e planejamento de projetos de *software* definido por Jeff Sitherland e Ken Schwaber no início da década de 1990.

Uma característica desse método é a organização do trabalho por *sprints*, que são as iterações, ou ciclos de desenvolvimento, normalmente realizados em um período de duas a quatro semanas. Um projeto de desenvolvimento é composto por um ou mais *sprints*. Os objetivos e as tarefas definidas para um *sprint* não são alterados e novos requisitos não são aceitos, caso algumas dessas necessidades aconteçam, elas serão avaliadas para o próximo *sprint* (SCRUMALLIANCE, 2011). O ciclo *Scrum* é detalhado de forma resumida na figura 3.

[3] Consiste em um banco de dados e processos definidos para inserção de dados, gerenciamento do banco de dados, para geração de modelos de maturidade e capacitação, avaliação e treinamento (COUTO, 2007).

[4] Nome que deriva de uma jogada que ocorre durante o jogo de *rugby* (PRESSMAN, 2010).

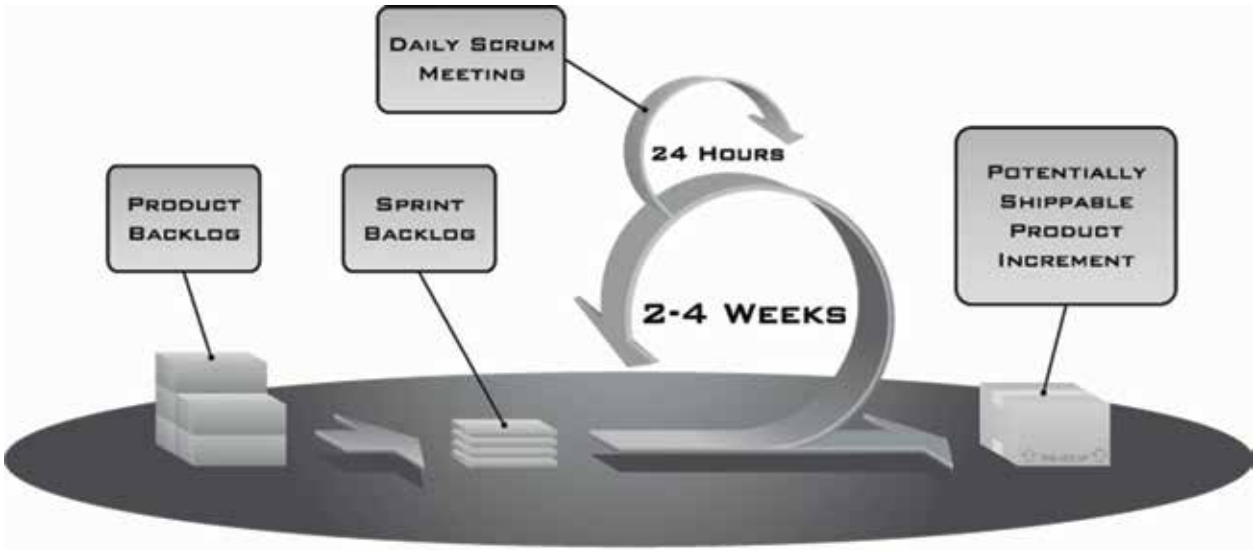

Figura 3 – Ciclo *Scrum*
Fonte: Mountain Goat (2011).

A cerimônia:

***Sprint Planing Meeting*:** reunião de planejamento realizada a cada novo *Sprint*, normalmente, em duas etapas, em que o *Product Owner* prioriza as funcionalidades contidas no *Product Backlog* e o *Scrum Team* define o *Sprint Backlog*.

***Daily Scrum Meeting*:** reunião diária, na qual o objetivo é alinhar a situação do *Sprint*. Para isso, cada membro da equipe responde às perguntas: o que você fez ontem?; O que você fará hoje?; Há algum impedimento para a realização? Caso exista, esse deve ser tratado pelo *Scrum Master* o mais rápido possível, mas fora da reunião em questão.

Sprint Review Meeting: reunião realizada ao final de cada *Sprint*, em que o *Scrum Team* apresenta o que foi realizado durante o *Sprint* ao *Product Owner* e *Scrum Master*, e demais interessados da gerência, clientes e engenheiros de outros projetos.

***Sprint Retrospective*:** ocorre ao final de cada *Sprint* e serve para identificar o que funcionou bem, o que pode ser melhorado e que ações serão tomadas.

Os artefatos:

***Product Backlog*:** é a relação de funcionalidades desejadas para o *software*. Essa pode e deve ser incrementada e ajustada ao longo do projeto.

Os itens detalhados abaixo serão repetidos tantas vezes quanto for necessário até que todas as funcionalidades descritas no *Product Backlog* sejam atendidas.

***Sprint Backlog*:** lista de funcionalidades selecionadas pela equipe para execução.

Burndown Chart: representação gráfica do tempo total, em horas, previsto para o *Sprint* e o decréscimo diário das horas previstas a serem executadas. Nesse gráfico, serão também representadas as informações de horas realizadas e horas disponíveis para a conclusão do *Sprint* (SCRUMALLIANCE, 2011).

Scrum board: quadro para acompanhamento e planejamento das tarefas do *Sprint* (SCRUMALLIANCE, 2011). Nesse quadro, as informações de - a realizar, em realização e concluídas - normalmente estão disponíveis, com mais ou menos detalhes.

Os papéis *Scrum*, esses já citados acima, e suas respectivas responsabilidades (IMPROVEIT, 2011):

***Product Owner*:** responsável por definir os itens do *Product Backlog* e priorizá-los.

***Scrum Team*:** equipe de desenvolvimento, não existindo, necessariamente, uma divisão funcional através de papéis tradicionais. Essa é composta de cinco a nove profissionais e todos trabalham juntos para conquistar os objetivos definidos.

Para trabalhar em equipes maiores, deve-se utilizar o conceito de *Scrum of Scrums*. Nele, cada *Scrum Team* trabalha normalmente, contudo um profissional dessa equipe participará do *Scrum of Scrums Meeting*, em que será coordenado o trabalho das múltiplas equipes *Scrum* (MOUNTAIN GOAT, 2011).

***Scrum Master*:** responsável por garantir que a equipe respeite e siga os valores e as práticas do *Scrum*, além de proteger a equipe, isso quer dizer auxiliá-la e, inclusive, a não se comprometer excessivamente. Pode ser qualquer pessoa da equipe.

1.4 EXTREME PROGRAMMING (XP)

O XP é um método para desenvolvimento de *software* indicado para equipes pequenas e médias, de dois a 10 profissionais, a qual está focada no desenvolvimento rápido, entre suas principais características, está a programação em pares (BECK, 2004). Esse método é composto por quatro valores, de acordo com Wells (2011):

Simplicidade: visa à criação de código simples, que for necessário e solicitado, ou seja, a adição de funcionalidades que podem ser importantes no futuro, mas que, no momento, não se fazem necessárias deve ser evitada.

Comunicação: seu objetivo é manter o melhor relacionamento possível entre desenvolvedores e clientes, desenvolvedores e gerentes, preferindo conversas pessoais a outros meios de comunicação.

***Feedback*:** o objetivo é que o desenvolvedor tenha informações constantes do código proveniente dos testes contínuos que indicam erros tanto individuais quanto do *software* integrado e do cliente, o qual tem acesso contínuo a uma parte totalmente funcional do *software*, podendo avaliar e sugerir novas informações e necessidades aos desenvolvedores.

Coragem: necessária para implantar os três valores anteriores. Para reportar a real situação sobre os progressos e estimativas e para se adaptar sempre que elas acontecerem.

As doze práticas propostas pelo XP:

O Jogo do Planejamento: etapa na qual o escopo da próxima versão é definido de forma breve combinando prioridades, estimativas e técnicas, sendo atualizado sempre que necessário (BECK, 2004).

Entregas Frequentes: coloque rapidamente um sistema simples em produção e trabalhe na liberação de versões frequentes (BECK, 2004).

Metáfora: guie o desenvolvimento com uma história simples de como o sistema funciona, essa deve ser compartilhada e entendida por todos (BECK, 2004).

Projeto Simples: projete o sistema da maneira mais simples possível, a fim de satisfazer os requisitos atuais, qualquer complexidade desnecessária ao atual momento do sistema deve ser removida assim que for descoberta (BECK, 2004).

Testes: os testes unitários devem ser escritos pelos programadores continuamente, os quais devem ser executados sem falhas, para que o desenvolvimento continue. Ao cliente fica a responsabilidade de escrever os testes que garantam que as funcionalidades foram concluídas (BECK, 2004).

Refatoração: o sistema é reestruturado, sem alterar funcionalidades, com o objetivo de remover duplicidades, melhorar a comunicação e flexibilizar o código (BECK, 2004).

Programação em Pares: todo código é produzido por uma dupla de desenvolvedores em uma máquina (BECK, 2004); um desenvolvedor implementa o código, enquanto o outro observa continuamente, a fim de identificar e pensar melhorias possíveis ao código em questão (WELLS, 2011).

Propriedade Coletiva: todos desenvolvedores podem modificar qualquer código, em qualquer lugar do sistema, a qualquer momento, desde que realizem o conjunto de testes necessários (BECK, 2004).

Integração Contínua: integre e atualize as versões do sistema várias vezes ao dia, ou ainda a cada vez que uma tarefa for terminada (BECK, 2004).

Semana de 40 horas: trabalhe no máximo 40 horas por semana e não execute horas extras por duas semanas seguidas. Caso necessário, os planos devem ser alterados, em vez de sobrecarregar as pessoas (WELLS, 2011).

Cliente Presente: inclua um cliente real no time, disponível em tempo integral para responder às dúvidas (BECK, 2004).

Padrões de Codificação: a codificação deve ser realizada respeitando padrões que facilitem a comunicação através do código (BECK, 2004).

2 MÉTODO E METODOLOGIA

Pesquisar é buscar conhecimento, ou ainda, como descreve o dicionário Michaelis *Online*, é buscar com diligência/zelo (MICHAELIS, 2011) Segundo Gil (2007), a pesquisa possui caráter pragmático. Essa é composta por um conjunto de procedimentos intelectuais e técnicos empregados na investigação, com o objetivo de solucionar o problema, finalidade principal da pesquisa. Logo, pode-se dizer que a metodologia confere os caminhos necessários para o autoaprendizado com conclusões potenciais. Detalhamento sobre o tipo de pesquisa realizada

Quanto aos fins / objetivos: foi fundamentada por uma pesquisa exploratória, cujo objetivo é proporcionar maior familiaridade com o tema pesquisado, uma das principais características desse tipo de pesquisa, que tem ainda a finalidade de desenvolver, esclarecer e modificar conceitos e ideias, possibilitando a formulação de problemas mais precisos ou hipóteses para estudos posteriores, além da identificação de variáveis relevantes (GIL, 2007).

Quanto aos meios / procedimentos: foi do tipo bibliográfico, essa visa a conhecer e analisar as teorias já produzidas sobre o tema central, avaliando

as possíveis contribuições para a sua compreensão. A pesquisa bibliográfica tem ainda o objetivo de ampliar os conhecimentos, auxiliando na construção e fundamentação de hipóteses que resolverão o problema (GIL, 2007). De maneira sucinta, pode-se dizer que é a ação de juntar as informações necessárias em torno de um fato para a construção do saber.

Quanto à abordagem: a análise da pesquisa utilizou a abordagem qualitativa, a qual se caracteriza pela interpretação dos fenômenos e a atribuição de significados, em que ambiente natural é a fonte direta para coleta de dados e o pesquisador é o instrumento-chave, o qual tende a analisar seus dados indutivamente, uma vez que não existe a intenção de medir ou categorizar dados, não sendo necessário o uso de métodos e técnicas estatísticas. O foco das pesquisas qualitativas é a compreensão das informações de modo mais global, relacionando fatores variados, privilegiando contextos (GIL, 2007). O critério da análise adotado foi a correspondência direta e indireta das práticas ágeis Scrum e XP e as metas específicas do CMMI.

2.1 UNIVERSO E AMOSTRA

O universo, ou ainda a população, compreende um conjunto de elementos, como empresas e pessoas, que possuem características necessárias à pesquisa. E a amostra é conjunto de elementos desse universo, selecionados (de acordo com uma regra ou plano) para representá-lo (PRODANOV; FREITAS, 2009).

Para responder ao problema de pesquisa do projeto em questão, consideraram-se duas linhas de desenvolvimento de *software*: a tradicional, através do CMMI, e os métodos ágeis através do *Scrum* e XP. A amostragem em questão é caracterizada como não probabilística, por acessibilidade ou conveniência, o que não permite representar o universo, apesar de ser válido para a amostra e os objetivos da pesquisa (GIL, 2007).

2.2 TÉCNICAS DE COLETA DE DADOS

A coleta de dados da pesquisa foi realizada principalmente através de documentação indireta, analisando registros estatísticos, diários, biografias, jornais, revistas, entre outros, os quais são capazes de proporcionar dados suficientemente ricos de modo mais objetivo (GIL, 2007).

3 RESULTADOS E ANÁLISE

Para a obtenção dos dados necessários ao estudo, a análise comparativa foi utilizada, essa tem o objetivo de identificar relações entre variáveis permitindo a comprovação e validação dos modelos, assim, facilitando a avaliação dos métodos ágeis selecionados, *Scrum* e XP, em relação ao modelo CMMI.

O CMMI confere às empresas que atendem suas práticas um nível de maturidade ou capacidade gradativo, como um selo de garantia à qualidade do processo e produto desenvolvido, facilitando e padronizando a identificação de empresas capacitadas conforme os critérios do modelo CMMI ao desenvolvimento de *software*. Esse é um critério considerado relevante, uma vez que o número de empresas desenvolvedoras é cada vez maior, mas que de forma alguma deve ser definitivo.

Quanto aos métodos ágeis, vários modelos têm surgido desde o manifesto ágil em 2001, o *Scrum* e o XP surgiram em conjunto com esse, logo, estão há mais tempo sendo utilizados e permitindo a análise de resultados atingidos. Ambos os modelos são populares entre profissionais de desenvolvimento de *software*, assim como os bons resultados obtidos por esses.

As práticas propostas pelo *Scrum* instruem a realização de processos puxados, em que o trabalho a ser realizado pela equipe considera de fato que precisa ser feito de acordo com as priorizações do cliente. O XP, também já mencionado, não faz referência direta nem a métodos puxados, nem empurrados, suas práticas estão mais focadas nas atividades de engenharia a serem realizadas para que o produto final seja entregue, ou seja, como esse produto será construído até sua entrega final, definindo fortemente a realização de algumas práticas, como, por exemplo: programação em pares, considerada essencial para a entrega de um produto de qualidade.

3.1 ANÁLISE COMPARATIVA DOS MODELOS

A análise foi realizada com base no quadro a seguir onde as áreas de processo (PA) dos níveis de maturidade 2 Gerenciado e 3 Definido, do modelo de maturidade em desenvolvimento de *software* CMMI, onde cada nível combina um conjunto de PAs, as quais devem ser atendidas para a constatação da maturidade daquele nível (CHRISSIS; KONRAD; SHRUM, 2008).

Para a análise comparativa a ser realizada entre as PAs dos níveis 2 e 3 de maturidade do CMMI e as práticas propostas pelos métodos ágeis *Scrum* e XP,

CMMI: Nível 2 - Gerenciado			
PA	Objetivo e Metas Específicas	Scrum	XP
REQM	Fornecer subsídios para gerenciar os requisitos dos produtos e componentes de produto, identificando inconsistências entre esses e os planos e produtos de trabalho. 1 Gerenciar Requisitos	Sprint Backlog: artefato onde os requisitos prometidos para o Sprint estão documentados. Daily Scrum Meeting: diariamente o trabalho previsto é realizado e gerenciado, assim como seus desvios.	Não há nenhuma prática direta proposta pelo método.
PP	Fornecer subsídios para estabelecer e manter planos visando a definir as atividades de projeto. 1 Estabelecer Estimativas 2 Elaborar um Plano de Projeto 3 Obter Comprometimento com o Plano	Sprint Planning Meeting: prática em que todo o trabalho é priorizado pelo Product Owner e Scrum Team define o que será entregue e como; nessa etapa, a equipe identifica ainda o tempo necessário para a conclusão, através do Planning Poker. Sprint Backlog: é definida a formalização do escopo acordado.	Jogo do Planejamento: prática que atende parcialmente a PA. Nessa, o trabalho a ser realizado e as estimativas são definidas pelos envolvidos no desenvolvimento, na área de negócio e na equipe de desenvolvimento.
PMC	Fornecer subsídios à visualização do progresso do projeto, possibilitando a tomada de ações corretivas apropriadas. 1 Monitorar o Projeto em Relação ao Plano 2 Gerenciar Ações Corretivas até sua Conclusão	Burndown Chart: gráfico que apresenta a evolução prevista e realizada do trabalho. Scrum board: quadro atualizado diariamente pela equipe através das informações da Daily Scrum Meeting, nessa, as informações de tarefas pendentes, em andamento e concluídas.	Entregas Frequentes: prática que atende parcialmente a PA. Ao trabalhar com o propósito de entregas frequentes, o planejamento do projeto será continuamente revisitado e ajustado conforme necessário.
SAM	Fornecer subsídios para gerenciar a aquisição de produtos de fornecedores. 1 Estabelecer Contratos com Fornecedores 2 Cumprir Contratos com Fornecedor	Não há nenhuma prática direta proposta pelo método.	Não há nenhuma prática direta proposta pelo método.
MA	Desenvolver e manter medições que auxiliem a necessidade de informação para gestão. 1 Alinhar Atividades de Medição e Análise 2 Fornecer Resultados de Medição	Burndown Chart e Scrum board: esses artefatos possibilitam a visualização das informações de andamento do desenvolvimento, úteis à gestão.	Rastreador: profissional responsável por dispor à equipe informações sobre o trabalho: estimativas, andamentos, testes e defeitos.
PPQA	Divulgar à equipe e gerência o atendimento aos processos e produtos de trabalho definidos para a realização do produto. 1 Avaliar Objetivamente Processos e Produtos de Trabalho 2 Fornecer Visibilidade	Scrum Master: uma das responsabilidades desse profissional é garantir que o processo seja seguido.	Programação em Pares: prática que atende parcialmente a PA, uma vez que um dos objetivos é que um dos profissionais esteja avaliando o produto gerado em relação a padrões, definições etc. Treinador: profissional responsável por garantir que a execução do processo XP.
CM	Fornecer subsídios para estabelecer e manter a integridade dos produtos de trabalho. 1 Estabelecer Baselines 2 Acompanhar e Controlar Mudanças 3 Estabelecer Integridade	Não há nenhuma prática direta relacionada.	Integração Contínua: prática em que a equipe estabelece padrões para a contínua integração dos fontes.

serão considerados os objetivos e as metas específicas de cada PA, de forma abrangente.

Os quadros com as análises são dispostos por nível de maturidade.

A análise considerou que as PAs do CMMI poderiam ser atendidas pelas práticas propostas pelos métodos ágeis integral, parcialmente ou ainda não serem atendidas. Com base nisso, foram considerados os percentuais, 0%, 50% e 100% para a elaboração dos gráficos, onde:

-0%: não há nenhuma prática direta ou indireta dos métodos ágeis que instruam o atendimento da PA;

-50%: há práticas indiretas que instruem o atendimento da PA;

-100%: existem práticas diretas que instruem o atendimento da PA.

a integridade e gestão dos produtos de trabalho, além de permitir rastrear o seu uso e as liberações e PPQA, 50%, parcialmente atendida, que se refere às auditorias de produto e processo, cujo objetivo é garantir que os planos e padrões definidos estão sendo seguidos.

No XP, uma única prática foi atendida na íntegra, PPQA, isso porque instrui de forma direta a realização das revisões por pares, essencial ao atendimento da PA, além de instruir a adesão ao processo através das responsabilidades do papel Treinador. As PAs REQM e SAM obtiveram 0% de aderência, as PAs PP, PMC, MA e CM, 50% de aderência.

Esses percentuais representam de alguma forma a proposta de cada um dos métodos, considerando que o *Scrum* se propõe a auxiliar na gestão e nos planejamentos dos projetos, era esperado que seu percentual de atendimento ao nível 2 fosse superior

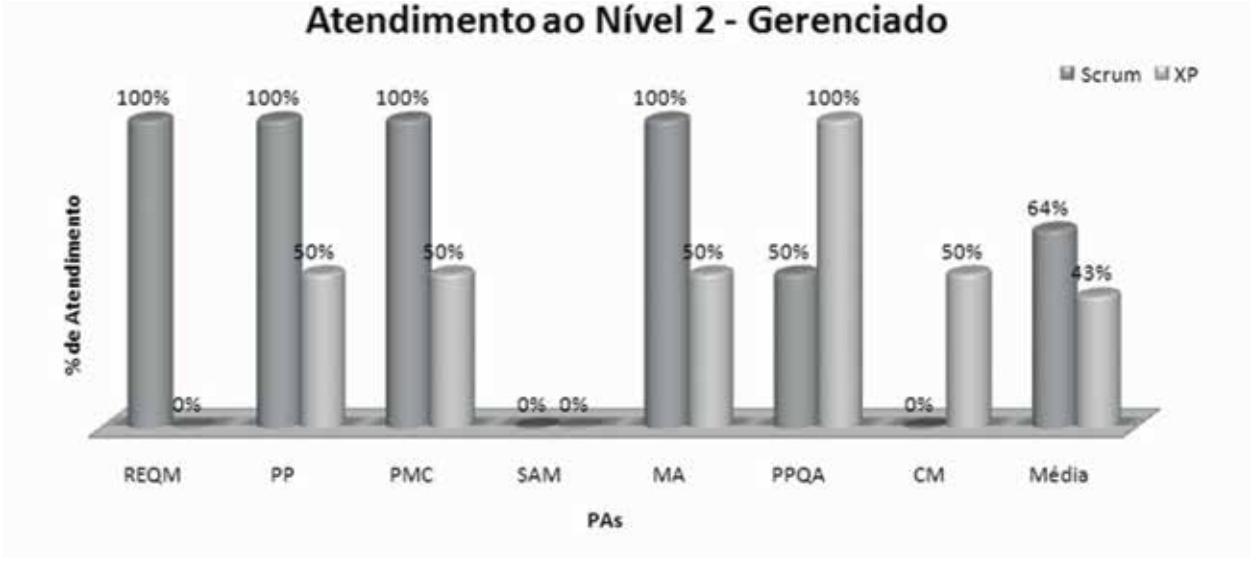

Gráfico 2 – Percentuais de Atendimento ao Nível 2 do CMMI

O gráfico 2 apresenta o resumo da análise e o percentual de atendimento dos métodos ágeis *Scrum* e XP, as PAs do nível 2 Gerenciado.

No nível 2 de maturidade, o *Scrum* obteve uma média de 64% de atendimento ao CMMI *versus* 43% obtidos pelo XP, uma diferença de apenas 21%.

No *Scrum,* apenas três das PAs não foram atendidas na íntegra, SAM, 0%, que se refere à contratação de terceiros, que podem de fato não existir em um desenvolvimento de *software*, não sendo assim considerada em algumas avaliações CMMI, CM, 0%, que se refere à gestão de configuração, ou seja, garantir ao XP, que é um método que se propõe a auxiliar no desenvolvimento de *software* propriamente dito.

O grande valor proposto pelas práticas *Scrum* está em focar esforços nos requisitos de maior valor ao cliente, tratando mudanças, requisitos e sua definição tardia como parte do processo.

CMMI: Nível 3 - Definido			
PA	Objetivo e Metas Específicas	Scrum	XP
RD	Produzir e analisar os requisitos de cliente, de produto e de componente de produto. 1 Desenvolver Requisitos de Cliente 2 Desenvolver Requisitos de Produto 3 Analisar e Validar Requisitos	Execução do Sprint Backlog: essas práticas serão realizadas durante a execução do sprint, sendo comum o uso de user history como ferramenta. Daily Scrum Meeting: nessa reunião, é garantido que todos os requisitos solicitados estão sendo trabalhados até que estejam prontos para a entrega.	Metáfora: nessa prática, a funcionalidade do sistema será detalhada e disseminada. Testes: garantem a correta implementação e liberação, para então ser iniciado um novo desenvolvimento. Cliente Presente: possibilita o fácil e rápido esclarecimento de dúvidas.
TS	Projetar, desenvolver e implementar soluções para os requisitos. Soluções, designs e implementações englobam produtos, componentes de produto e processos de ciclo de vida relacionados ao produto, seja de forma isolada ou em conjunto. 1 Selecionar Soluções de Componentes de Produto 2 Desenvolver Design 3 Implementar Design do Produto	Execução do Sprint Backlog: essas práticas serão realizadas durante a execução do sprint, contudo não há nenhuma regra ou instrução de como devem ser realizadas. A sua realização e forma de realização são de responsabilidade da equipe.	Projeto Simples: onde os requisitos serão projetados conforme as necessidades atuais. Refatoração: prática que instrui a reestruturação do sistema (sem modificar funcionalidades), mas melhorando a codificação. Padrões de Codificação: definições que facilitem o entendimento de todos os envolvidos.
PI	Fornecer subsídios para montar o produto a partir de componentes de produto, garantindo que o produto integrado execute as funções de forma adequada. 1 Preparar-se para Integração de Produto 2 Assegurar Compatibilidade das Interfaces 3 Montar Componentes do Produto e Entregar Produto	Idem ao item anterior.	Integração Contínua: prática em que a integração é planejada, padrões definidos e a prática realizada continuamente, garantindo a integridade do produto.
VER	Garantir que os produtos de trabalho selecionados satisfaçam os seus requisitos especificados. 1 Preparar-se para Verificação 2 Realizar Revisão por Pares 3 Verificar Produtos de Trabalho Selecionados	Idem ao item anterior.	Programação em Pares: que garante a codificação padrão e otimizada. Testes: que garante que todo desenvolvimento atende a sua necessidade requerida.

Continua

VAL	Demonstrar que o produto ou componente de produto satisfaz o seu uso pretendido, quando colocado em seu ambiente pretendido. 1 Preparar-se para Validação 2 Validar Produto ou Componentes de Produto	Sprint Review Meeting: na realização dessa prática, a orientação a apresentar resultando do sprint funcional, ou seja, utilizar o próprio sistema para demonstrar o que e como foi feito.	Não há nenhuma prática direta relacionada.
OPF	Planejar, implementar e implantar melhorias nos processos da organização a partir da identificação de seus pontos fortes e fracos. 1 Determinar Oportunidades de Melhoria de Processo 2 Planejar e Implementar Melhorias de Processo 3 Implantar os Ativos de Processo da Organização e Incorporar Lições Aprendidas	Sprint Retrospective: nessa prática, será avaliado o que deu certo e o que não deu certo na realização do sprint, essas informações servirão de insumo para a melhoria do trabalho, o próximo sprint.	Refatoração: prevê a melhoria do código fonte. Propriedade Coletiva: instrui que todos podem melhorar qualquer melhoria nos fontes, desde que realizem a bateria de testes com sucesso.
OPD	Estabelecer e manter um conjunto utilizável de ativos de processo da organização e de padrões de ambiente de trabalho. 1 Estabelecer Ativos de Processo da Organização	Não há nenhuma prática direta relacionada.	Não há nenhuma prática direta relacionada.
OT	Desenvolver habilidades e conhecimentos da equipe para que essa possa desempenhar seu trabalho de forma eficiente e eficaz. 1 Estabelecer uma Capacidade de Treinamento na Organização 2 Proporcionar Treinamento Necessário	Sprint Retrospective: nessa prática, a equipe tem a oportunidade de expor as dificuldades encontradas, resultando inclusive nas orientações necessárias aos profissionais da equipe.	Programação em Pares: os pares devem ser alterados com frequência, o que permite a contínua troca de conhecimento e a aprendizagem entre as pessoas.
IPM	Possibilitar o gerenciamento do projeto e envolvimento das partes interessadas relevantes de acordo com um processo definido e integrado adaptado a partir do conjunto de processos. 1 Utilizar o Processo Definido para o Projeto 2 Coordenar e Colaborar com as Partes Interessadas Relevantes	Sprint Review Meeting: essa reunião possibilita parte do atendimento dessa prática através do envolvimento de todos os interessados e da apresentação dos objetivos atingidos.	Não há nenhuma prática direta relacionada.

Continua

Conclusão

RSKM	Viabilizar a identificação de potenciais problemas (riscos) antes que ocorram, permitindo o tratamento desses para mitigar impactos indesejáveis que comprometam a realização dos objetivos. 1 Preparar-se para Gestão de Riscos 2 Identificar e Analisar Riscos 3 Mitigar Riscos	Daily Scrum Meeting: diariamente, a equipe expõe os problemas e impedimentos para a realização do problema, que são os riscos relacionados ao sucesso do sprint, esses serão solucionados pelo Scrum Master.	Entregas Frequentes: apesar de não estar diretamente relacionada a PA, essa prática possibilita o constante feedback do cliente e ajustes nos planos e estratégias conforme informações obtidas junto esse. É classificada como uma forma de também gerenciar riscos.
DAR	Viabilizar a tomada de decisões baseado em um processo formal de alternativas em relação a critérios estabelecidos. 1 Avaliar Alternativas	Não há nenhuma prática direta relacionada.	Não há nenhuma prática direta relacionada.

No gráfico 3, é apresentado o resumo da análise e do percentual de atendimento dos métodos ágeis *Scrum* e XP, as PAs do nível 3 do CMMI, Definido.

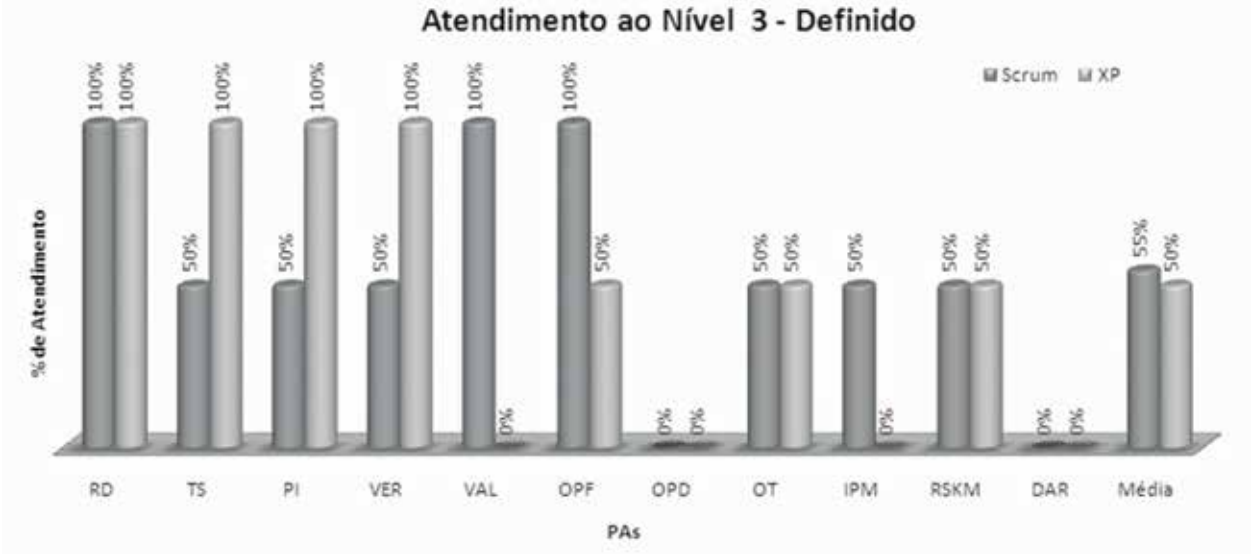

Gráfico 3 - Percentuais de Atendimento ao Nível 3 do CMMI

Nesse nível, os métodos ágeis *Scrum* e XP obtiveram, respectivamente, a média de 55% e 50% de atendimento ao CMMI. Porém, conforme já mencionado, esses métodos são complementares, podendo ser adotados em conjunto, o que nesse caso certamente resultaria em um percentual final de atendimento ao CMMI superior ao percentual em questão.

Nesse nível, o CMMI conta com PAs que orientam a parte de engenharia e construção de *software*, a saber: RD, TS, PI, VER, VAL e IPM, sobre as quais as práticas *Scrum* permitiram atender na íntegra (100%) 2 PAs e parcialmente (50%) 4 PAs *versus* 4 PAs atendidas na íntegra (100%) e 2 não atendidas (0%) pelo XP, o propósito desse modelo é justamente auxiliar nas atividades de desenvolvimento de *software*. Esses dados conferem ao *Scrum* e ao XP uma média específica de 66,67%.

Quanto às PAs relacionadas às estratégias de gestão, têm: OT, RSKM e DAR, tanto o *Scrum* quanto o XP atenderem parcialmente (50%) a 2 PAs e não atenderam (0%) a 1 PA, o que confere a ambos os métodos equilíbrio nesse quesito.

Para a melhoria e continuidade dos processos de desenvolvimento, têm-se as PAs OPF e OPD diretamente relacionadas, e demais PAs do nível 3 e também do nível 2 indiretamente relacionadas. Para as diretamente relacionadas, OPF e OPD, as práticas *Scrum* tiveram aderência maior, atendendo a OPF integralmente (100%) enquanto o XP atendeu parcialmente (50%), OPD não foi atendida por nenhum dos métodos.

O valor do XP está em características como: priorização do código executável e funcional, garantido através da prática de testes em que a continuidade do trabalho está diretamente relacionada ao resultado positivo do código em questão, assim como o contínuo nivelamento de conhecimento entre os profissionais através da programação em pares.

CONCLUSÃO

Em busca por melhores processos de desenvolvimento, as organizações enfrentam muitos impedimentos, pois a melhoria deve considerar as necessidades da organização, as práticas existentes, os recursos disponíveis, a cultura da organização, entre outros. De modo que não há um método, seja esse ágil ou tradicional, que resolva todos os problemas da organização, infelizmente, a sonhada "bala de prata" não existe. O que existe, sim, são melhorias através do estudo das práticas disponíveis e identificação do que cabe à realidade e necessidade da organização.

Os modelos selecionados, CMMI, *Scrum* e XP, são os mais difundidos e conhecidos atualmente entre empresas desenvolvedoras de *software* e profissionais de TI. O CMMI é um modelo reconhecido mundialmente no desenvolvimento de *software*, sendo uma das características desse modelo não detalhar como as atividades devem ser realizadas, e sim os objetivos a serem atingidos, por isso, aplicáveis aos mais diversos processos de desenvolvimento, inclusive aos processos orientados por práticas ágeis. Outra característica importante é a cultura de melhoria constante que este agrega ao processo.

Quanto aos métodos ágeis, o *Scrum* e o XP orientam a gestão ágil de desenvolvimento de *software* e algumas práticas a serem implementadas no dia a dia do processo de desenvolvimento respectivamente. Uma característica comum desses métodos é a minimização dos riscos por meio de entregas rápidas e contínuas, ou seja, a disponibilização constante de *software* funcionando ao Cliente, o que permite um *feedback* contínuo, além dos alinhamentos e das adequações necessárias. A prática de entregas contínuas nos métodos ágeis é um pré-requisito, o que garante a minimização dos riscos, nos métodos tradicionais, a prática também pode ser aplicada, contudo não será obrigatória, podendo não ocorrer e, logo, não gerando os mesmos resultados.

Embora os métodos ágeis sejam diferentes em suas práticas, esses compartilham várias características, como: respeito pelas pessoas, comunicação e foco em atividades e documentações que agreguem valor ao produto final. Muitas dessas práticas estão diretamente alinhadas ao pensamento *lean*. Outra característica dos métodos ágeis estudados é que esses suportam, de forma madura, a mudança constante das necessidades do cliente em relação ao produto em desenvolvimento. Característica fortemente sustentada pelos métodos ágeis em questão prezarem mais adaptação e menos planejamento, ao contrário da maioria dos métodos tradicionais.

A análise realizada entre CMMI *versus Scrum* e XP, permitiu identificar que, na média, os métodos ágeis estudados atendem, facilmente, mais de 50% das PAs do modelo CMMI. Logo, conclui-se que é possível obter laudo positivo nos níveis: 2 Gerenciado e 3 Definido do CMMI, por meio da adoção das práticas ágeis propostas pelos métodos *Scrum* e XP, claro que adotando em conjunto algumas práticas específicas para atendimento do modelo em questão.

Os modelos estudados *Scrum* e XP são complementares em muitas etapas e podem e devem ser adotados em conjunto, assim como uma série de outros métodos possivelmente possam ser parcialmente adotados, melhorando pequenas etapas do processo de desenvolvimento. Contudo, essa definição deve ser realizada com cautela e de acordo com as necessidades da organização, pois o melhor processo de desenvolvimento certamente é aquele que está adequado às necessidades da organização e da equipe, ou seja, um processo coeso e que atenda às necessidades estratégicas, tal como a adoção do CMMI. Para que isso seja possível, provavelmente a adoção purista de um único método não seja a melhor alternativa.

REFERÊNCIAS

ABES – Associação Brasileira das Empresas de Software. **Mercado Brasileiro de Software:** Panorama e Tendências 2009**.** Disponível em: <http://www.abes.org.br/arquivos/MercadoBR-2009-ResumoExec.pdf>. Acesso em: 23 jun. 2010.

BARTIÉ, Alexandre. **Garantia de Qualidade de Software.** Rio de Janeiro, RJ: Campus, 2002. 291p.

BECK, Kent et al. **Manifesto Ágil.** 2001. Disponível em: <http://www.manifestoagil.com.br>. Acesso em: 30 jun. 2010.

BECK, Kent. **Programação Extrema Explicada**: Acolha as Mudanças. Porto Alegre, RS: Bookman, 2004. 182p.

CHRISSIS, Mary B.; KONRAD, Mike; SHRUM, Sandy. **CMMI**: guidelines for process integration and product improvement**.** 2. ed. Estados Unidos: Addison Wesley, 2008. 675p.

CÔRTES, Matrio L.; CHIOSSI, Thelma C. dos S. **Modelos de Qualidade de Software.** São Paulo, SP: Unicamp, 2001. 148p.

COUTO, Ana Brasil. **CMMI – Integração dos Modelos de Capacitação e Maturidade de Sistemas.** Rio de Janeiro, RJ: Editora Ciência Moderna, 2007. 275p.

FOWLER, Martin. **The New Methodology**. 2005. Disponível em: <http://www.martinfowler.com/articles/newMethodology.html>. Acesso em: 05 jul. 2010.

GIL, Antonio C. **Métodos e Técnicas de Pesquisa Social.** 5. ed. São Paulo, SP: Atlas, 2007. 206p.

GUERRA, Ana C.; COLOMBO, Regina M. T. **Tecnologia da Informação**: Qualidade do Produto de Software. Brasília, DF: PBQP Software, 2009. 423p.

IMPROVEIT, **Improve IT**. Disponível em: < http://improveit.com.br/scrum>. Acesso em: 09 abr. 2011.

MICHAELIS, **Michaelis Moderno Dicionário Da Língua Portuguesa:** versão online. Disponível em: <http://michaelis.uol.com.br/moderno/portugues/index.php?lingua=portugues-portugues&palavra=software >. Acesso em: 20 mar. 2011.

MOUNTAIN GOAT, **Mountain Goat Software**. Disponível em: <http://www.mountaingoatsoftware.com/scrum>. Acesso em: 22 abr. 2011.

PRESSMAN, Roger S. **Engenharia de software.** 5. ed. São Paulo, SP: Makron Books, 1995. 1056p.

PRESSMAN, Roger S. **Engenharia de software**. 6. ed. Porto Alegre, RS: AMGH, 2010. 720p.

ROCHA, Ana R. C. da; MALDONADO, José C.; WEBER, Kival C. **Qualidade de Software**. São Paulo, SP: Prentice Hall, 2001. 303p.

SCRUMALLIANCE. **Scrum Alliance Tranforming the world of work**. Disponível em: <http://www.scrumalliance.org/>. Acesso em: 15 mai. 2011.

SOMMERVILLE, Ian. **Engenharia de Software**. 6. ed. São Paulo, SP: Addison Wesley, 2003. 592p.

WELLS, Don. Extreme Programmig: A Gentle Introduction. Disponível em: http:<//www.extremeprogramming.org/>. Acesso em: 18 abr. 2011.