Revista de Ciências Exatas e Tecnologia

Vol. IV, Nº. 4, Ano 2009

METODOLOGIAS ÁGEIS E A MOTIVAÇÃO DE PESSOAS EM PROJETOS DE DESENVOLVIMENTO DE SOFTWARE

Aplicando práticas de SCRUM e XP para promover a motivação de equipes de projetos de desenvolvimento de software

Kenji Taniguchi
Faculdade Anhanguera de Taubaté
kenji@kenji.com.br

Fernando Eugenio Correa
Faculdade Anhanguera de Taubaté
eugenio.fec@gmail.com

RESUMO

Este artigo tem o objetivo de apresentar as características existentes nas metodologias ágeis que podem ser trabalhadas para aumentar a motivação da equipe. Conhecendo as exigências atuais do mercado, as estruturas organizacionais e os dois tipos distintos de desenvolvimento atuais, um totalmente anárquico e outro totalmente focado em controle (representado pelas metodologias tradicionais), serão apresentadas as fragilidades e problemas atualmente conhecidos. Estes fatores geram desconforto e insatisfação nos desenvolvedores que não se sentem comprometidos ou motivados com suas atribuições. Com estes fatores em mente, os métodos ágeis são apresentados e dois deles, SCRUM e eXtreme Programming, têm suas práticas estudadas de forma a apresentar como estas podem ser trabalhadas para focar ainda mais as pessoas de forma a aumentar seu comprometimento e motivação, fazendo que os indivíduos envolvidos sintam-se parte das tomadas de decisão do projeto.

Palavras-Chave: SCRUM; eXtreme Programming; metodologias ágeis.

ABSTRACT

This article aims to present the features found in agile methodo-logies that can be applied to increase the motivation of the team. Knowing the current requirements of the market, the organizational structures and the two distinct types of current development, one totally anarchic and the other totally focused on control (represented by the traditional methods), will be presented the weaknesses and problems currently known. These factors generate discomfort and dissatisfaction in developers who do not feel motivated or committed to its mission. With this factors in mind, the agile methods are presented and two of them, SCRUM and eXtreme Programming, have their practices studied to present how these can be applied to focus even more on people in order to increase their commitment and motivation, making the individuals involved feel part of decision making phase of the project.

Keywords: SCRUM; eXtreme Programming; agile methodologies.

Anhanguera Educacional

Correspondência/Contato
Alameda Maria Tereza, 2000
Valinhos, São Paulo
CEP 13.278-181
rc.ipade@unianhanguera.edu.br

Coordenação
Instituto de Pesquisas Aplicadas e Desenvolvimento Educacional - IPADE

Informe Técnico
Recebido em: 20/04/2010
Avaliado em: 15/07/2010

Publicação: 21 de dezembro de 2010

1. INTRODUÇÃO

Vive-se um novo paradigma econômico que estabelece a necessidade de atualização contínua, da preocupação com a satisfação do cliente e da geração de produtos e serviços de maior valor agregado (BECKER, HUSELID, ULRICH, 2001).

Hoje, mais do que as tecnologias que uma empresa possui, o conhecimento é a chave para agregar valor aos seus produtos e serviços. Neste cenário, as empresas necessitam cada vez mais do conhecimento, comprometimento e motivação de seus funcionários (SPÍNOLA, 2008).

No entanto, um dos grandes desafios é o de manter equipes motivadas e comprometidas, baseando-se nas metodologias tradicionais de gerenciamento de projetos de desenvolvimento de *software*. A aplicação das metodologias tradicionais têm resultado em não cumprimento de prazo de entrega, escopo não atendido, grande quantidade de erros, funcionalidades desnecessárias, descontentamento por parte do cliente entre outros problemas (CAMPOS, FONSECA, 2008).

Com a proposta de desenvolver projetos de uma maneira capaz de responder rápido às mudanças, com foco nas pessoas e na colaboração com o cliente, surgiram as metodologias ágeis que, devido às suas características, possibilitam gerar produto com maior valor agregado e ao mesmo tempo manter pessoas motivadas dentro das corporações (AKITA, 2009).

Para entender como a aplicação de metodologias ágeis pode aperfeiçoar a qualidade do produto e também promover a motivação das equipes, é necessário efetuar, ao menos, uma rápida avaliação das estruturas organizacionais e como estas afetam direta ou indiretamente a gestão dos projetos e os indivíduos envolvidos, além disso, deve-se efetuar a análise baseada na visão que se tem atualmente de projeto e das metodologias utilizadas (tradicionais e ágeis) e como proceder com o uso conjunto de algumas práticas ágeis, visando a meta do projeto e a motivação dos indivíduos.

2. CENÁRIO ATUAL DAS EMPRESAS DE DESENVOLVIMENTO DE SOFTWARE

No ambiente atual, intensamente competitivo, a busca por melhorar a qualidade do produto entregue, respondendo a mudanças e satisfazendo o cliente, tem se tornado uma meta e uma necessidade para as organizações.

Para alcançar esse objetivo, é necessário um conjunto de atividades e tarefas que após sua aplicação sistemática, resultem em *software* funcional. A esse conjunto de tarefas,

atividades e processos dá-se o nome de *metodologia de desenvolvimento de software* (BASSI, 2009).

A gestão de um projeto de desenvolvimento de software cabe, muitas vezes, a um gerente de projetos ou a uma equipe de gerenciamento de projetos. Esta pessoa ou equipe tem a responsabilidade de determinar qual o melhor conjunto de boas práticas para um projeto específico (HELDMAN, 2006).

3. ESTRUTURAS ORGANIZACIONAIS

Cada organização é única, possuindo cultura e peculiaridades próprias, porém, elas podem ser estruturadas segundo três modalidades: funcional, por projetos ou matricial.

O modelo de estrutura acaba interferindo no nível de autoridade que um gerente de projeto terá dentro da organização e no comprometimento que os indivíduos envolvidos nos projetos terão com as metas e objetivos do mesmo (HELDMAN, 2006).

3.1. Organizações Funcionais

Esta é a estrutura organizacional mais comum e antiga dos três modelos, também sendo conhecida como estrutura tradicional (HELDMAN, 2006). Neste modelo organizacional, as equipes são formadas em torno de suas especialidades e funções, sendo administradas pelos respectivos gerentes de cara área, estes chamados de gerentes funcionais.

Considerando o escopo de projetos, este tipo de organização apresenta algumas desvantagens muito claras à gerência. Pode-se destacar a clara falta ou ausência de autoridade do gerente de projeto, assim como o fator de que em projetos de desenvolvimento de software tem-se a necessidade de formar equipes multidisciplinares, o que não é algo muito fácil de administrar dentro de uma estrutura funcional, pois os indivíduos tendem a manter comprometimento com sua área e não propriamente com o projeto (MARTINS, 2002).

3.2. Organizações por Projetos

Nesta estrutura, o foco da empresa é o projeto e suas metas. As diversas áreas da empresa tendem a se comprometer com o projeto.

Para definir projeto deve-se primeiro atentar a duas características básicas (MARTINS, 2002):

- Um projeto é uma iniciativa única de alguma forma. Diferenciando assim, projetos de operações. O desenvolvimento de um novo jogo para computador pode ser considerado um projeto, enquanto a gravação deste em mídias para a venda consiste numa operação.
- Projetos possuem um fim definido, ou seja, um objetivo que ao ser atingido define o final do projeto.

As contratações e a própria atividade da área de recursos humanos é vinculada às necessidades dos projetos, fazendo com que as características dos profissionais contratados e mantidos sejam totalmente vinculadas ao que é necessário ao projeto. O maior problema encontra-se no fato de haver pouco comprometimento com a empresa e sim com o projeto, no qual os indivíduos são, normalmente, contratados para desenvolver o projeto ou mesmo uma determinada fase do mesmo.

Este ambiente focado no projeto tende a ser desestimulante para as pessoas que visam crescimento profissional dentro da organização (HELDMAN, 2006).

Como as equipes são formadas "em torno" de um projeto, a relação entre elas é de curta duração, podendo ainda haver realocação devido à necessidade das características de determinado perfil em outro projeto mais importante para a empresa (MARTINS, 2002).

3.3. Organizações Matriciais

Este tipo de organização baseia-se na fusão de características da estrutura funcional com a estrutura por projetos. As organizações matriciais surgiram como meio de minimizar os pontos negativos das organizações por projetos e das funcionais e trabalhar os seus pontos positivos. Os objetivos do projeto são atendidos e focados, mas mantendo-se a estrutura hierárquica da empresa (HELDMAN, 2006).

Essa estrutura possibilita ainda o balanceamento de suas características conforme a escolha da empresa, tendo-se assim, uma organização matricial funcional, por projetos ou balanceada (MARTINS, 2002).

4. EVOLUÇÃO DO PROCESSO DE DESENVOLVIMENTO DE SOFTWARE

O *software* era escrito sem um plano específico e possuía diversas decisões de curto prazo. Este "estilo", conhecido também como codificação *cowboy*, não seguia metodologias e era executado a partir da própria experiência dos programadores (FOWLER, 2005).

Os ajustes necessários devidos a novos requisitos ou defeitos encontrados (que são cada vez mais constantes e de difícil solução) surgiam com prioridade urgente, gerando pressão e *stress*, implicando em horas extras, desgaste físico e mental. Este desgaste acaba colaborando para a baixa qualidade e confiabilidade do *software* desenvolvido e gerando, principalmente, a desmotivação da equipe.

Como alternativas para essa ausência de práticas, surgiram as metodologias tradicionais, que primam pelo controle e têm como objetivo transformar o desenvolvimento de *software* em algo previsível e mais eficiente. Utilizando processos detalhados e que enfatizam o planejamento e documentações.

Apesar de ter como objetivo a eficiência, estas metodologias não têm alcançado suas metas ao longo dos anos de sua aplicação. A maior crítica às metodologias tradicionais é o fato de serem extremamente burocráticas e com elevado número de etapas que não agregam valor ao produto (FOWLER, 2005).

Apesar da previsibilidade ao longo de um projeto ser um fator muitíssimo desejado, em desenvolvimento de software isso não é, normalmente, uma realidade. Segundo Fowler (2005), todo desenvolvedor tem o cliente como seu grande inimigo, pois este nunca sabe o que realmente quer. Devido a este pensamento as metodologias tradicionais buscam descrever da forma mais completa possível os requisitos do software e documentar todas as funcionalidades solicitadas antes do início do desenvolvimento para que, a partir disto, tenha-se uma ferramenta de proteção para as futuras mudanças que (certamente) ocorrerão (FOWLER, 2005).

Tratando o desenvolvimento de *software* como um trabalho empírico, dificilmente seria possível planejar desde o início todos os requisitos e todas as funcionalidades que realmente serão necessárias e úteis.

O processo *Waterfall* (ou Cascata) simboliza muito bem o padrão das metodologias tradicionais. Este modelo é caracterizado por fases muito bem definidas e que a saída da fase anterior gera o produto de entrada da fase posterior. Ao término de cada etapa é associada uma documentação que deve ser aprovada para que tenha início a próxima fase (ROYCE, 1970). As etapas que compõem o processo podem diferir um pouco, porém o padrão documentado em 1970 por Royce cita os seguintes passos: Requisitos do Sistema, Requisitos de *Software*, Análise, *Design*, Codificação, Testes e Implantação, conforme apresentado na Figura 1.

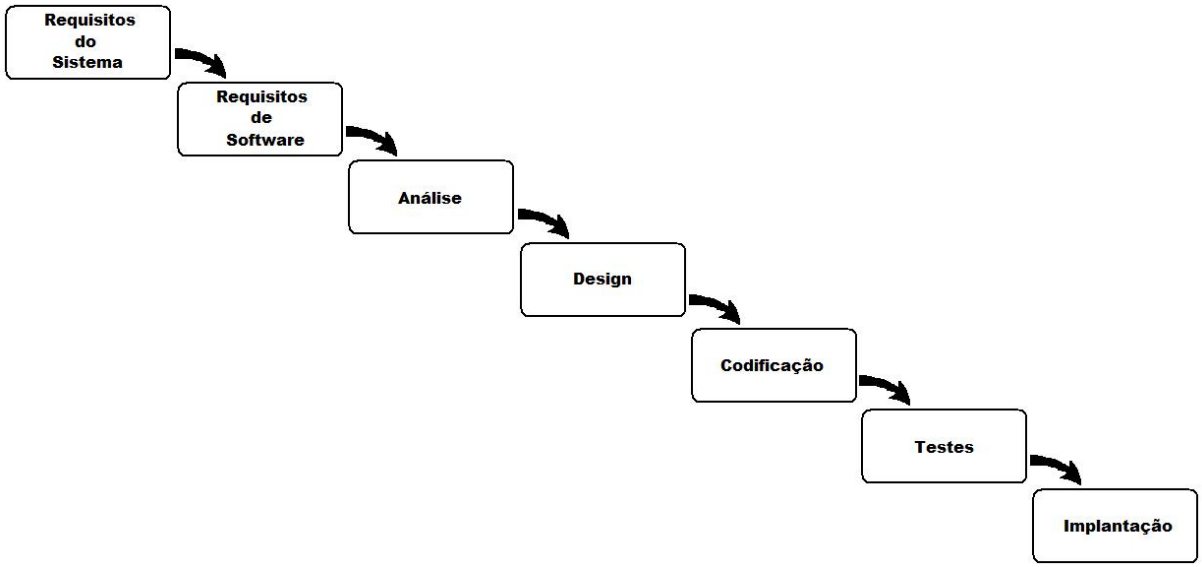

Figura 1 – Modelo Waterfall adaptada de Royce (1970).

Ao documentar o processo, mesmo confiando na ideia do mesmo, já foram verificadas suas deficiências "Eu acredito neste conceito, mas a aplicação descrita é arriscada e é um convite a falhas" (ROYCE, 1970, p. 2).

Conhecidas inicialmente como metodologias leves, as metodologias ágeis surgiram como uma resposta à necessidade de mudanças na forma de gerir projetos de desenvolvimento de *software* e os processos envolvidos nestes projetos.

A principal diferença a citar entre os métodos ágeis e os métodos tradicionais é o conceito de simplicidade ou do "mínimo necessário", ou seja, ao invés de usar um enorme conjunto de processos, busca-se iniciar um projeto com o uso do menor volume de processos possível e, conforme a necessidade, novos procedimentos devem ser incluídos (TELES, 2005, p. 53).

Em 2001 reuniram-se 17 metodologistas para analisar o que eles estavam fazendo de diferente em seus projetos para atingir os resultados que estavam obtendo. Desta reunião, que tinha como meta a definição de uma metodologia que unificasse as práticas que estavam sendo discutidas, resultou num conjunto de valores e princípios, sendo conhecido como Manifesto Ágil (BECK, et al., 2001).

São quatro os valores dos métodos ágeis:

- Indivíduos e interações entre eles mais que processos e ferramentas.
- *Software* funcionando mais do que documentação abrangente.
- Colaboração com o cliente mais do que negociação de contratos.
- Responder a mudanças mais que seguir um plano.

Isto não quer dizer que os itens à direita não possuem valor, e sim que os da esquerda possuem mais valor.

A partir do conhecimento dos valores, podem-se citar os doze princípios das metodologias ágeis (BECK et al., 2001):

- A maior prioridade é satisfazer o cliente, por meio da entrega adiantada e contínua de *software* de valor.
- Aceitar mudanças de requisitos, mesmo no fim do desenvolvimento. Processos ágeis se adéquam a mudanças, para que o cliente possa tirar vantagens competitivas.
- Entregar *software* funcionando com freqüência, na escala de semanas até meses, com preferência aos períodos mais curtos.
- Pessoas relacionadas ao negócio e desenvolvedores devem trabalhar em conjunto e diariamente, durante todo o curso do projeto.
- Construir projetos ao redor de indivíduos motivados, dando a eles o ambiente e suporte necessário, e confiar que farão seu trabalho.
- O método mais eficiente e eficaz de transmitir informações para - e por dentro de um time de desenvolvimento - é através de uma conversa cara a cara.
- *Software* funcional é a medida primária de progresso.
- Processos ágeis promovem um ambiente sustentável: os patrocinadores, desenvolvedores e usuários, devem ser capazes de manter, indefinidamente, passos constantes.
- Contínua atenção a excelência técnica e bom design aumentam a agilidade.
- Simplicidade: a arte de maximizar a quantidade de trabalho que não precisou ser feito.
- As melhores arquiteturas, requisitos e designs emergem de times auto-organizáveis.
- Em intervalos regulares, o time reflete sobre como ficar mais efetivo, então, ajustam-se e aperfeiçoam seu comportamento.

Apesar de todos os princípios acima citados e dos quatro valores, o maior destaque para a "real agilidade" consiste em dois pontos principais: metodologias ágeis são adaptativas e orientadas às pessoas (FOWLER, 2005).

A partir destes dois pontos principais enaltecidos por Fowler, têm-se duas das mais conhecidas metodologias ágeis, SCRUM e eXtreme Programming (também conhecida como XP).

O maior destaque é dado atualmente para estas duas metodologias devido ao fato de que elas podem ser consideradas totalmente compatíveis e complementares, pois SCRUM possui características voltadas à gestão de projetos e das pessoas que estão

incluídas no mesmo, enquanto XP está mais direcionada à gestão dos processos (KNIBERG, 2007).

4.1. eXtreme Programming (XP)

EXtreme Programming, XP ou programação extrema são as formas que se encontra mais comumente na literatura uma das metodologias ágeis mais famosas.

A metodologia ágil XP foi criada para equipes pequenas ou médias de desenvolvimento de *software* que tenham que tratar constantemente com requisitos vagos e em constante mudança (BECK, 1999).

A base da metodologia XP é formada por quatro valores: comunicação, *feedback*, simplicidade e coragem. Estes valores são completados por 12 práticas (FOWLER, 2005).

Algumas destas práticas, se aplicadas juntamente com outras metodologias, permitem melhoria de processos e motivação. Além de citar as referidas práticas, serão explicados os motivos de sua seleção a seguir (KNIBERG, 2007):

a) Programação em pares: permite código de melhor qualidade, pois recebe "refatoração" quase que em tempo de execução e também dissemina parte do conhecimento entre os membros da equipe, gerando ainda aumento de produtividade.

b) Desenvolvimento orientado a testes (ou TDD): serve de mecanismo de proteção, pois através do desenvolvimento de testes que poderão ser executados automaticamente, assegura-se que ao desenvolver novas funcionalidades que serão adicionadas ao sistema, qualquer inconsistência seja mais facilmente reconhecida. Esta facilidade na detecção de possíveis inconsistências dá maior segurança aos desenvolvedores.

c) Integração contínua: poupa tempo e resolve possíveis problemas futuros. Caso algum problema de integração ocorra este será facilmente detectado, pois se trata de um pequeno lote de integração.

d) *Design* simples (ou incremental): partindo de um *design* simples inicialmente, pode-se efetuar a melhoria contínua do mesmo, permitindo ainda uma manutenção mais rápida e fácil.

e) Propriedade coletiva do código: facilita manutenção, disseminação de conhecimento e gera qualidade e simplicidade. Aumenta o comprometimento da equipe com o projeto e não apenas com um determinado conjunto de códigos de "sua responsabilidade".

f) Padrão de codificação: é uma premissa básica para a propriedade coletiva do código e para que haja qualidade e facilidade de manutenção. Caso não haja um padrão definido, cada desenvolvedor sempre terá o seu próprio padrão para utilizar.

g) Ritmo sustentável: trabalhar mais horas gera desgaste físico e psicológico, baixa qualidade no serviço efetuado e sintomas destrutíveis como descontentamento e desmotivação na equipe, muitas vezes sem prover real aumento de produção.

4.2. SCRUM

SCRUM é uma metodologia ágil que visa fornecer *software* aos clientes de forma rápida e com maior qualidade. É fundamentada na teoria de controle de processos empíricos e empregando abordagem iterativa e incremental. SCRUM é descrito, pelos seus criadores, como um *framework* de inspeção e adaptação, dentro do qual é possível aplicar vários processos e técnicas visando o desenvolvimento de produtos complexos, de forma a aumentar a qualidade e previsibilidade dentro dos projetos (SCHWABER; SUTHERLAND, 2009).

As metodologias ágeis foram criadas baseando-se, em boa parte, nas práticas que davam certo nas empresas japonesas. Influenciadas principalmente pelo conceito de desenvolvimento enxuto (*lean development*) de indústrias como Toyota e Honda (SUTHERLAND, 2007).

Além dos pais do processo SCRUM, Ken Schwaber e Jeff Sutherland, muitos são os responsáveis pelo conjunto de práticas que existem e são atualmente aplicadas, porém os padrinhos do SCRUM são Takeuchi e Nonaka, os primeiros a utilizar o termo (SUTHERLAND, 2007).

Hirotaka Takeuchi e Ikujiro Nonaka explicam em seu artigo (*apud* SUTHERLAND, 2007) "*The New New Product Development Game*" o conceito de auto-organização e de pequenas equipes multidisciplinares e suas vantagens, além de utilizar pela primeira vez a referência de SCRUM do Rugby. Neste mesmo artigo, ainda explicam, por meio de *cases*, como configurar uma equipe auto-organizada e o papel da gerência no processo.

Com isso pode ser dito que SCRUM nasceu a partir de ótimas influências para atingir com êxito a motivação de pessoas, já que também tem em suas raízes muito das influências de gestão do conhecimento, foco de estudo de Takeuchi e Nonaka.

Os três pilares de SCRUM são:

a) Transparência: caso algo esteja errado no processo, todos devem ter conhecimento, nada deve ser escondido ou evitado. E se alguém acredita que algo está pronto, isso deve ser equivalente ao que conta na definição de "pronto".

b) Inspeção: a inspeção deve ser feita de maneira que fatores inaceitáveis à qualidade sejam encontrados, porém dentro do limite de que a própria inspeção não gere necessidade de nova inspeção.

c) Adaptação: após inspeção, se necessário efetuar alteração, esta deverá ser feita e, preferencialmente, o mais rápido possível para que não gere desvios futuros.

Como o foco deste artigo está na aplicação de SCRUM com a adição de algumas práticas de XP para alcançar a qualidade do produto desenvolvido e principalmente a motivação dos envolvidos, o SCRUM será explicado com mais detalhes abaixo.

SCRUM é formado basicamente por papéis, cerimônias e artefatos, tudo isso mantendo um conceito importante que é o de time boxes (SCHWABER; SUTHERLAND, 2009).

Os papéis são:

a) *Product Owner* (PO): é o responsável pelos requisitos e é quem tem o conhecimento do negócio e das reais necessidades do cliente. Tem a responsabilidade de priorizar e validar os requisitos (SCHWABER; SUTHERLAND, 2009;
b) SCRUM *Master* (SM): responsável por manter o processo SCRUM funcionando, que todos apliquem as práticas e trabalha como um facilitador para o time (removendo impedimentos e protegendo-o de interferências externas), além de auxiliar e guiar o *Product Owner* em suas tarefas, se isto for necessário (SUTHERLAND, 2007);
c) Time SCRUM (ou simplesmente Time): é formado por pessoas com as competências necessárias para transformar os requisitos levantados pelo PO em um pedaço de *software* "potencialmente concluído" (SUTHERLAND, 2007).

A partir destes papéis, uma visão prática que se pode ter em relação ao micro gerenciamento e macro gerenciamento, se apresentam da seguinte forma:

a) O PO é responsável pelo macro gerenciamento, devido ao conhecimento do negócio e mantendo a meta a ser atingida que é a visão do projeto;
b) o Time é responsável pelo micro gerenciamento, isso devido as suas características de auto organização e multidisciplinaridade, mantendo o foco na meta estipulada para cada iteração;
c) o SM tem a função de remover impedimentos que o Time possa ter durante o projeto e cuidar para que as práticas da metodologia sejam bem aplicadas.

SCRUM possui três artefatos principais que permitem o planejamento (*Product Backlog e Sprint Backlog*) e gerenciamento (Gráfico *Burndown*) do projeto pelo próprio Time conforme o mesmo é desenvolvido. Estes artefatos são apresentados abaixo:

a) *Product Backlog*: lista montada e mantida pelo PO e priorizada pelo ROI (*return over investment*) sobre tudo que se acredita ser necessário no produto.
b) *Sprint Backlog*: conjunto de tarefas selecionadas pelo Time a partir do *Product Backlog* para gerar um subconjunto funcional do *software*.
c) Gráfico *Burndown*: gráfico que demonstra o quanto resta do *Backlog* em relação ao tempo disponível.

As cerimônias em SCRUM servem como base para a inspeção e adaptação constante pregada pela metodologia, pois através destas o Time é capaz de constantemente avaliar o andamento do projeto e das iterações. As cerimônias citadas a seguir:

a) Reunião de planejamento do *release*.
b) Reunião de planejamento da *sprint*.
c) *Sprint*.
d) Reunião diária.
e) Revisão da *Sprint*.
f) Retrospectiva da *Sprint*.

A *Sprint* é uma iteração e respeita o conceito de *time boxes* (eventos com duração fixa) do SCRUM. A duração das *Sprints* deve estar entre uma e quatro semanas, tendo-se como prática recomendada, que a duração seja de duas semanas, para equipes iniciantes na prática de SCRUM. Das cerimônias acima citadas, a *Sprint* pode ser destacada como a alma geradora de valor a ser entregue ao cliente (SCHWABER; SUTHERLAND, 2009).

5. SCRUM E XP COM FOCO NAS PESSOAS

Métodos ágeis são adaptativos, portanto, se é intenção da organização implantar este tipo de processos, ela deve ter em mente o conceito de que deverá confiar nos indivíduos que farão parte dos projetos, envolvendo-os nas decisões. (FOWLER, 2005). Um dos pontos mais importantes em processos adaptativos é que é necessário que os envolvidos sejam competentes e motivados, e os métodos ágeis disponibilizam isso através da participação nas decisões e na gestão das tarefas pelos envolvidos, gerando motivação e competência.

A importância dada pelas metodologias ágeis às pessoas envolvidas nos projetos já está clara nos valores constantes no manifesto ágil, no qual se destaca o primeiro destes que cita que metodologias ágeis valorizam "indivíduos e iterações entre eles mais que processos e ferramentas" (BECK et al., 2001).

Com este valor como principal guia, a metodologia SCRUM se destaca pelas características de procurar manter a visão na gestão do projeto e não em processos e controles. Em outras palavras, aplicando-se SCRUM, será possível descobrir o que há de errado nos processos e o que pode ser feito para corrigir estes problemas (SCHWABER; SUTHERLAND, 2009). Algumas práticas de XP permitem melhoria de processos mantendo o foco no cliente e nos funcionários.

Para esclarecer a visão de que o uso de SCRUM juntamente com algumas práticas complementares de XP, possibilita a motivação dos indivíduos envolvidos em um dado projeto, será apresentada uma seqüência resumida das atividades básicas de um projeto no qual são aplicadas as técnicas acima citadas.

Um projeto nasce a partir de uma visão, uma necessidade ou uma idéia. Mas pode ser dito que o objetivo do projeto é simbolizado pela visão. A partir desta visão o *Product Owner* fica responsável por "alimentar" e priorizar o *Product Backlog*, que basicamente é formado por um conjunto de estórias (conceito de XP para uma forma de representar os requisitos ou casos de uso em um projeto).

É muito importante também que seja definido o conceito de "pronto" para o projeto, ou seja, se uma estória for considerada terminada, ela deve obrigatoriamente encaixar-se no conceito de "pronto" que foi estipulado. Garantindo assim, que todos os envolvidos no projeto (PO, SM e Time) tenham o mesmo significado de "pronto".

A partir das estórias contidas no *Product Backlog*, fica a cargo do Time, juntamente com o PO, estimar e selecionar as estórias que estejam "bem descritas" que farão parte da *Sprint*. Isto ocorre durante a reunião de planejamento de *Sprint*. As principais informações que se deve ter para esta reunião são o próprio *Product Backlog*, a capacidade (ou velocidade) do Time, o último incremento gerado e o histórico de desempenho do Time (SCHWABER; SUTHERLAND, 2009). Recomenda-se o uso de *Planning Poker* (variação do método de estimativa *Wideband Delphi*) para que o time possa efetuar a tarefa de estimativa das estórias.

A partir de um conjunto de cartas com uma série de valores (baseados na série de Fibonacci) e a partir de informações obtidas através do PO, cada integrante do Time seleciona o valor que acredita ser a estimativa que mais se aplica. Quando todos já tiverem selecionado sua estimativa, as cartas devem ser apresentadas e caso haja uma diferença muito grande entre os valores selecionados, os que apresentarem o maior e o menor valor devem explicar seus motivos e após recolherem as cartas uma nova "rodada" é efetuada tendo em mente as explicações ouvidas (BASSI, 2008). A grande vantagem no uso de *Planning Poker* é o fato de que se evita muito a possibilidade de que haja influências durante a tarefa de estimativas (KNIBERG, 2007).

Com o *Sprint Backlog* montado e a meta da *Sprint* clara, as estórias selecionadas deverão ser "quebradas" em tarefas e o Time procede com o desenvolvimento. Como anteriormente citado, as estórias só poderão ser consideradas prontas se cumprirem o que consta como "pronto".

Uma ferramenta muito utilizada em projetos SCRUM é o SCRUM *Board* (Figura 2). Ferramenta de micro gerenciamento de extrema valia ao Time para que possa se gerenciar e organizar de maneira transparente a todos, pois, como um quadro de *Kanban*, fica visível a todos, e permite que a todo e qualquer instante um dos membros do Time o atualize, selecionando uma tarefa para execução ou mesmo atualizando as informações do andamento de determinada tarefa, podendo inclusive mudá-la para a coluna de "pronto" (KNIBERG, 2007).

Durante o período de duração da iteração, deve ocorrer a reunião diária (ou *Daily Meeting*). Esta reunião é de curta duração (normalmente 15 minutos) e serve para o Time manter-se informado, coeso e integrado sobre o andamento das tarefas, permitindo identificar rapidamente dentro da *Sprint* a necessidade de melhorias ou mudanças, é um ponto de inspeção e adaptação que incrementa a organização e comprometimento da equipe. (SCHWABER; SUTHERLAND, 2009). Nesta reunião os membros do Time devem responder a três perguntas:

a) O que fiz ontem;
b) o que planejo fazer hoje;
c) quais os impedimentos (se existirem).

Caso haja algum impedimento, o mesmo deverá ser passado ao SM para que este trate de resolvê-lo para o Time (SCHWABER; SUTHERLAND, 2009).

Figura 2 – Desenho representativo de um SCRUM Board.
Fonte: Adaptado de Kniberg (2007).

Ao término da *Sprint*, é executada a revisão da *Sprint*, momento este no qual o resultado alcançado é apresentado ao PO para sua análise e aprovação. Esta reunião fornece dados importantes para análise no próximo planejamento de *Sprint* (SCHWABER; SUTHERLAND, 2009).

Após a revisão ocorre a reunião de retrospectiva da *Sprint*. Essa reunião é mais uma peça fundamental no processo de inspeção e adaptação em projetos SCRUM. Nela, o Time irá levantar o que funcionou e o que ocorreu de errado na última *Sprint*, analisando o que e como pode ser melhorado. A participação do SM nessa reunião possibilita que este incentive o uso da metodologia e encontre caminhos para tentar resolver ou minimizar os problemas e impedimentos encontrados. O ciclo SCRUM é facilmente visualizado Figura 3.

Incluindo ainda a prática da programação em pares (XP) em alguns momentos, pode-se aumentar o nivelamento do conhecimento dentro da equipe, além de possibilitar revisão de código durante a construção do mesmo (KNIBERG, 2007).

Durante o desenvolvimento do projeto, novas estórias (ou requisitos) poderão ser incluídas no *Product Backlog* sem que isso gere impactos no trabalho do Time, pois o PO deverá incluir estas estórias e priorizá-las para que na próxima reunião de planejamento de *Sprint*, essas novas estórias sejam consideradas (dependendo ainda de sua prioridade).

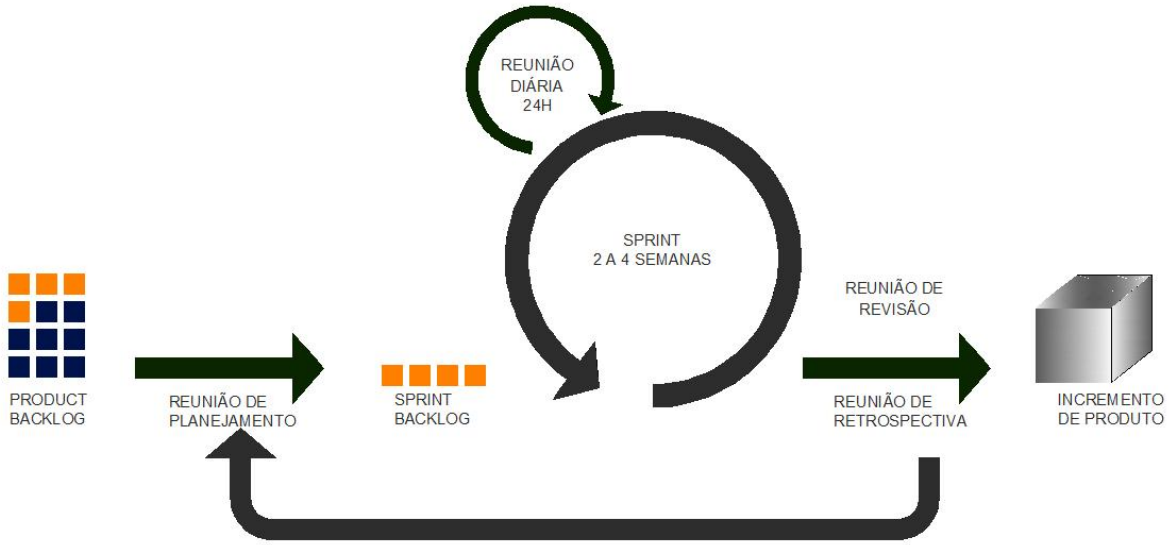

Figura 3 – Fluxo de um projeto SCRUM.
Fonte: Adaptado de Haddad (2009).

Conforme apresentado, através da aplicação de SCRUM, se obtêm vários "momentos" em que a equipe de desenvolvimento pode escolher, adaptar, inspecionar, auto-organizar e mensurar seu próprio trabalho.

6. CONSIDERAÇÕES FINAIS

As metodologias ágeis surgiram como uma alternativa no ambiente atual das organizações. Neste cenário, têm-se as grandes, as pequenas e as médias organizações.

Economicamente, para as grandes organizações é possível implantar processos tradicionais, porém estes são pesados e burocráticos, gerando desperdício e desmotivando seus funcionários devido ao número elevado de procedimentos que devem ser seguidos para efetuar uma dada tarefa. Nestes casos, os membros das equipes são normalmente tratados como meros "recursos", um número, um elemento que permitirá que empresa alcance seu objetivo que é o desenvolvimento de um determinado produto ou serviço com os requisitos descritos no início do projeto e que provavelmente não reflitam mais as reais necessidades do cliente ao término do trabalho.

Na grande maioria das empresas (compostas por médias, pequenas e micro empresas) é muito comum não fazer uso de nenhuma metodologia ou processo. Por falta de recursos financeiros ou pelo motivo que estes processos possam atrasar e encarecer os projetos. Nestes casos, tem-se a típica ocorrência do processo de codificação e correção em ciclos que parecem infindáveis e, devido à falta de planejamento, tem-se prazos não cumpridos, excesso de erros durante todo o período e sobrecarga entre os membros da equipe.

A opção de uma empresa pelo uso de metodologias ágeis já fornece aos desenvolvedores a sensação de que a organização confia em suas competências, somando-se a isso práticas SCRUM, que focam totalmente nas iterações entre os indivíduos da equipe e como estes interagem, tem-se um ambiente totalmente favorável ao crescimento profissional com grande troca de conhecimentos devido à formação de equipes multidisciplinares. Estas equipes se auto-gerenciam (selecionando numa iteração o que será feito, como será e, até mesmo, por quem será feito), mantendo o conhecimento diário do andamento das tarefas e permitindo constantes mudanças caso haja necessidade.

Todos esses fatores promovem na equipe o sentimento de que todos estão no controle do projeto, o qual não pertence à empresa, ao cliente ou a um gerente de projeto, e sim a todos os envolvidos, e que desta forma sentem-se mais valorizados e comprometidos com o objetivo que, ao ser alcançado, motivará ainda mais estes indivíduos em busca de conhecimento e crescimento profissional por meio da adaptação constante empregada no processo.

AGRADECIMENTOS

Alguns fatos fazem-nos quase desistir de algumas coisas, felizmente temos pessoas para nos apoiar e estimular. Agradeço à minha esposa, Priscila, por sempre ficar ao meu lado e instigar-me a desenvolver este artigo, aos meus pais, Avandir e Zilda, que juntos formaram tudo que sou hoje e que ainda hei de me tornar, aos professores, Kenji e Robson, pela orientação e pelo lado humano apresentado, e ao meu irmão e mestre, Avandir (Índio), por tudo que passamos e por tudo que ele me mostrou e ensinou durante sua passagem. Obrigado.

REFERÊNCIAS

AKITA, F. **Você não entende nada de SCRUM**. Disponível em: <http://akitaonrails.com/2009/12/10/off-topic-voce-nao-entende-nada-de-scrum>. Acesso em: 21 fev. 2010.

BASSI, D. Planejamento ágil de projetos. **Engenharia de Software Magazine**, Rio de Janeiro, Devmedia Group, 8. ed., jan. 2009. Disponível em: <http://www.devmedia.com.br/articles/viewcomp.asp?comp=11306>.

_____. Estimativas ágeis com planning poker. **Engenharia de Software Magazine**, Rio de Janeiro, Devmedia Group, 9. ed., fev 2009. Disponível em: <http://www.devmedia.com.br/articles/viewcomp.asp?comp=11596>.

BECK, K. **Extreme programming explained**: embrace change. 1. ed. Reading: Addison Wesley, 1999. 224 p.

BECK, K. et al. **Manifesto for agile software development**. Disponível em: <http://www.agilemanifesto.org >. Acesso em: 17 jul. 2009.

BECKER, B.E.; HUSELID, M.A. **The HR scorecard:** linking people, strategy and performance. Boston: Harvard Business School Press, 2001. 235 p.

CAMPOS, A.; FONSECA, I. Por que SCRUM?. **Engenharia de Software Magazine**, Rio de Janeiro, Devmedia Group, 4. ed., set. 2008. Disponível em: <http://www.devmedia.com.br/articles/viewcomp.asp?comp=9868>.

FOWLER, M. **The new methodoly**. Disponível em: <http://martinfowler.com/articles/newMethodology.html>. Acesso em: 10 fev. 2010.

HADDAD, M. **Estrutura do SCRUM**. Disponível em: <http://connections.fiap.com.br/blogs/scrum/entry/estrutura_do_scrum?lang=pt_br>. Acesso em: 25 fev. 2010.

HELDMAN, K. **Gerência de projetos (PMP Project Management Professional):** guia para o exame oficial do PMI. 3. ed. Tradução de Luciana do Amaral Teixeira. Rio de Janeiro: Elsevier, 2006. 529 p.

KNIBERG, H. **SCRUM e XP direto das trincheiras**: como fazemos SCRUM. Infoqueue, 2007. 145 p. Disponível em: < http://www.infoq.com/br/minibooks/scrum-xp-from-the-trenches>. Acesso em: 9 jan. 2010.

MARTINS, J.C.C. **Gestão de projetos de desenvolvimento de software (PMI - UML)**. 1. ed. Rio de Janeiro: Brasport, 2002. 189 p.

NONAKA, I.; TAKEUCHI, H. **The new new product development game**. Harvard Business Review, Boston, 1986. Disponível em: <http://hbr.org/product/new-new-product-development-game/an/86116-PDF-ENG>. Acesso em: 13 jul. 2009.

ROYCE, W.W. **Managing the development of large software systems**. Wescon, Los Angeles, 1970. Disponível em: <http://www.cs.umd.edu/class/spring2003/cmsc838p/Process/waterfall.pdf>. Acesso em: 13 fev. 2010.

SCHWABER, K.; SUTHERLAND, J. **Guia do SCRUM**. Disponível em: <http://www.scrum.org/storage/scrumguides/Scrum%20Guide%20-%20PTBR.pdf#view=fit>. Acesso em: 10 fev. 2010.

SUTHERLAND, J. **The SCRUM papers**: nuts, bolts and origins of an agile process. Disponível em: <http://scrumtraininginstitute.com/home/stream_download/scrumpapers>. Acesso em: 25 jan. 2010.

SPÍNOLA, R.O. Introdução à gestão de conhecimento. **Engenharia de Software Magazine**, Rio de Janeiro, Devmedia Group, 6. ed., nov. 2008. Disponível em: <http://www.devmedia.com.br/articles/viewcomp.asp?comp=10575>.

TELES, V.M. **Um estudo de caso da adoção das práticas e valores do extreme programming**. 2005. 181 f. Dissertação (Mestrado em Informática). UFRJ, IM/DCC, Rio de Janeiro, 2005. Disponível em: <http://www.improveit.com.br/xp/dissertacaoXP.pdf>. Acesso em: 18 fev. 2010.