REVISTA CIENTÍFICA
Intr@ciência

SCRUM – Método Ágil: uma mudança cultural na Gestão de Projetos de Desenvolvimento de Software

MACHADO, Marcos
Pós-graduado em Tecnologia e Sistemas de Informação - Unisanta
MEDINA, Sérgio Gustavo
Mestre em Ciência da Computação – Instituto Tecnológico de Tokyo

Resumo

Vivemos um momento bastante interessante na área de desenvolvimento de Software – Metodologias Ágeis. O objetivo deste artigo é desmistificar esse paradigma e contribuir com o campo de estudos, através da apresentação e reflexão dos principais conceitos e propostas do tema.

Palavras-Chave: Scrum, Flexibilidade, Metodologia Ágil.

Abstract

This is a very interesting time in the area of software development – Agile Methodologies. This article aims to demystify this paradigm and contribute to the field of studies through the presentation and reflection of the key concepts and theme proposals.

Key words: *Scrum, Flexibility, Agile Methodologies.*

Introdução

Uma nova abordagem para desenvolvimento de software tem despertado grande interesse entre as organizações de todo o mundo. Estamos vivendo uma tendência para o desenvolvimento ágil de aplicações devido ao ritmo acelerado de mudança na tecnologia da informação, pressões por constantes inovações, concorrência acirrada e grande dinamismo no ambiente de negócios (BOEHM, 2006).

Em Tecnologia da Informação os gestores estão cada vez mais sob pressão para obterem resultados que impulsionem uma melhoria do produto final. Ambientes empresariais, independente da queda da economia e cortes no orçamento, continuam a mudar a um ritmo muito rápido e os profissionais de TI lutam para acompanhar o ritmo dessas mudanças.

Estas mudanças levaram a um crescimento no desenvolvimento de metodologias ágeis na gestão de projetos de desenvolvimento de software, cujo objetivo é maximizar a produtividade de um grupo de trabalho com a promessa de entrega rápida, flexibilidade e qualidade.

Fazendo uma analogia ao modelo de gestão industrial idealizado por Henry Ford, fundador da Ford Motor Company, era um modelo de Produção em Massa que revolucionou a indústria automobilística nas décadas de 1950 e 1960. Os veículos eram montados sobre esteiras rolantes que se movimentavam enquanto o operário ficava praticamente parado, realizando uma etapa da produção, ou seja, não era necessária quase nenhuma qualificação por parte desses operários.

Como se tratava de um modelo de produção em massa, Ford percebeu que com a cor preta, a tinta secava mais rápida e consequentemente os carros poderiam ser montados mais rapidamente. Ficou famosa a frase de Ford, que dizia que poderiam ser produzidos automóveis de qualquer cor, *"desde que fossem pretos"*.

Entretanto, a *rigidez* deste modelo de gestão industrial foi à causa de seu declínio. Na década de 70, após a crise do petróleo e a entrada de competidores japoneses no mercado automobilístico, o "Fordismo" e a "Produção em Massa" entraram em crise e começaram a ser substituídos pelo modelo de produção baseado no Sistema Toyota de Produção.

O sistema de produção da Toyota, conhecido como *lean manufacturing system* ou *Just in Time System*, é uma das maiores contribuições ao mundo empresarial. Os fundadores Toyoda Sakichi, seu filho Toyoda Kiichiro e o principal executivo e engenheiro Taiichi Ohno, criaram esse sistema com o objetivo de aumentar a eficiência da produção pela eliminação contínua de desperdícios. Esses conceitos foram apresentados ao mundo pela Toyota quando introduziu as idéias *"jidoka e just in time"*. A primeira refere-se à

automação com toque humano; toda vez que ocorre um problema, o equipamento para imediatamente, com o objetivo de prevenir a produção de outros produtos com defeito; e a segunda o processo produz somente aquilo que é requerido pelo próximo processo, gerando assim um fluxo contínuo.

Pode-se perceber que a indústria automobilística foi um segmento que se estagnou durante muito tempo, assim como a indústria de desenvolvimento de software. Com idéias inovadoras a Toyota estabeleceu um novo *benchmark* de produção efetiva de alta qualidade, visando satisfazer seus clientes e apostando no conhecimento dos seus funcionários.

O Scrum força uma mudança cultural na forma de pensar, na forma de agir e tira todas as pessoas da zona de conforto e dependendo da cultura da empresa, normalmente acaba sendo rejeitado pelas pessoas envolvidas.

Dentro deste contexto, o propósito deste artigo é apresentar esse framework de processo ágil utilizado para gerenciar e controlar o desenvolvimento de um produto de software através de práticas iterativas e incrementais.

O Paradigma da Mudança

A expressão “Métodos Ágeis” aqui no Brasil vem se tornando popular nos últimos anos por utilizar uma abordagem simplificada e, ao contrário do que muitos imaginam, geralmente acaba sendo confundida com falta de controle e completa anarquia. Porém, ser ágil nos dias de hoje é fazer a diferença em relação aos concorrentes e ao contrário do que se imagina exige muita disciplina e organização.

Agilidade é a habilidade de criar e responder a mudanças com respeito ao resultado financeiro do projeto em um turbulento ambiente de negócios. Agilidade é a habilidade de balancear flexibilidade com estabilidade (HIGHSMITH, 2004). Muita estrutura e organização reduzem a criatividade e a flexibilidade de suportar as mudanças. Ao contrário, permeia a ineficiência e resulta em esforços maiores que o necessário. Highsmith (2004) enfatiza que a ausência de estrutura ou estabilidade pode levar ao caos, mas que a estrutura em demasia gera rigidez.

Eis o grande paradigma da mudança. Defensores da Metodologia Ágil consideram ser difícil obter apoio da gestão para a execução do que parecem ser dramáticas mudanças no desenvolvimento de aplicativos, isto porque, esses métodos causam uma "falsa sensação" de descarte do papel do gestor no sentido de garantir o sucesso.

Métodos, práticas e técnicas para o desenvolvimento ágil de projetos prometem aumentar a satisfação do cliente (BOEHM, 2003) para produzir alta qualidade de software e para acelerar os prazos de desenvolvimento de projetos (ANDERSON, 2003).

Em 2001, um grupo de 17 autores e representantes das Metodologias Ágeis tais como: Scrum, eXtreme Programming (XP), Feature Driven Development (FDD), entre outros; discutiram um padrão de desenvolvimento de projetos dentre as técnicas e metodologias existentes e o resultado desse encontro foi a criação do Manifesto para Desenvolvimento Ágil de Software (AGILE MANIFESTO, 2001). Estabeleceram um framework comum para processos ágeis valorizando os seguintes itens:

- Indivíduos e interações mais que processos e ferramentas;
- Software funcionando mais que documentação compreensiva;
- Colaboração do cliente mais que negociação do contrato; e
- Responder à mudança mais que seguir um plano.

Isto é, embora haja valor nos itens da direita, os mais valorizados são os itens da esquerda.

Por que ser ágil?

Ken Schwaber (2007) em seu livro The Enterprise and Scrum, afirma que em projetos típicos cerca de 50% do tempo são gastos com requisitos, arquitetura e especificação e que tudo isso é feito antes mesmo de se construir qualquer funcionalidade. Entretanto, 35% dos requisitos mudam e 65% das funcionalidades descritas pelos requisitos nunca ou raramente serão utilizadas.

No Waterfall Model – Modelo Clássico Cascata, por exemplo: o cliente só recebe o produto pronto para ser usado ao ser aprovado nos testes de aceitação, depois de inteiramente especificado, implementado e testado, recebendo o valor investido somente no final do projeto.

Por esse motivo o Scrum defende a utilização somente de documentação suficiente e necessária para ajudar no desenvolvimento do projeto. Vale salientar que o grande objetivo do projeto é o produto e não a documentação.

No Scrum, o *Product Backlog* é uma lista que contém as prioridades das funcionalidades a serem implementadas no projeto. Trata-se de uma lista que acompanha todas as necessidades do cliente, ou seja, isso garante que sempre serão executadas as funcionalidades de maior importância no momento anterior ao início de cada ciclo de desenvolvimento com duração de 15 a 30 dias, chamado de *Sprint.*

No ambiente de negócios globalizado o cliente necessita de retorno rápido do capital investido. Ao contrário de outras metodologias, Scrum prioriza explicitamente o retorno de investimento (ROI). O *Product Owner* tem a função de maximizar e priorizar as atualizações do Product Backlog, de forma que os itens de maior valor para o cliente em cada momento sejam implementados primeiro. Dessa forma, o incremento ao produto realizado ao final de cada sprint gera retorno ao cliente por diversas vezes ao longo do projeto.

O Manifesto Ágil defende que responder as mudanças é mais importante do que seguir um plano. Por se tratar de um framework ágil, Scrum encara as mudanças como parte natural do processo de desenvolvimento. Com a atualização do Product Backlog, as novas solicitações do cliente podem ser introduzidas no próximo sprint, gerando vantagem competitiva para as empresas.

Conhecendo o Scrum - Framework

No jogo de Rugby, o “Scrum” é a forma de reiniciar o jogo após uma falta. Todos os jogadores se posicionam em um bolo humano para competir pela bola. É uma jogada onde um time de oito integrantes trabalha em conjunto para levar a bola adiante no campo do adversário e concretizar seu único objetivo. O Goal.

O termo SCRUM foi associado ao desenvolvimento pela primeira vez por Hirotaka Takeuchi e Ikujiro Nonaka no livro “The New Product Development Game”. Os autores enaltecem a importância de se adotar uma forma de desenvolvimento onde toda a equipe trabalhe como uma unidade para atingir um objetivo comum.

Em 1995, Ken Schwaber formalizou o Scrum para projetos de desenvolvimento de software baseado no sistema *lean* da Toyota.

Segundo Fonseca (2009), Scrum são times trabalhando como uma unidade altamente integrada com cada membro desempenhando um papel bem definido e o time inteiro focando num único objetivo – entrega do produto. Elimina práticas de controle desnecessárias, inadequadas e burocráticas, se concentrando na essência do processo de confecção de sistemas de informação.

O Scrum é bastante objetivo, possuindo metas claras, equipe bem definida, flexibilidade, comprometimento, cooperação; e sua curva de aprendizado é relativamente baixa. Apesar de o Scrum ser uma metodologia da área de exatas, ela tem muito da área de humanas e isso precisa ser levado em consideração o tempo todo.

Segundo seu autor Schwaber (2004), o Scrum não é um processo previsível, ele não define o que fazer em todas as circunstâncias. Ele é utilizado em trabalhos complexos onde não é possível prever os acontecimentos e oferece um framework e um conjunto de práticas que torna tudo visível. Isso permite a equipe ter uma visão exata dos fatos ao longo do projeto e se necessário, realizar os devidos ajustes visando alcançar seus objetivos. Este é um dos pontos fortes do Scrum: “Adaptabilidade e Flexibilidade”.

O Scrum não é “milagreiro” e muito menos irá oferecer uma “receita pronta” para resolver todos os seus problemas. A única certeza é que esses problemas serão identificados com mais facilidade. Por se tratar de um framework servirá como um guia de boas práticas para alcançar seu objetivo.

SCRUM – Papéis e Responsabilidades

O Scrum implementa um esqueleto interativo e incremental através de papéis e responsabilidades descritas a seguir:

Product Owner

É um especialista de negócios que representa todos os clientes. Conhece a fundo todas as regras de negócios e é o responsável pelo retorno financeiro (ROI) do produto.

O *Product Owner* é o responsável por criar o *Release Plan* e o *Product Backlog*. Como é ele quem sabe o que é mais importante

para o negócio, cabe a ele atualizar e priorizar continuamente o *Product Backlog (Planning)* para assegurar que as funcionalidades mais valiosas sejam produzidas primeiro. Pode mudar os requisitos e prioridades a cada *Sprint*, bem como aceitar ou rejeitar o resultado de cada Sprint.

Em algumas empresas esse papel é conhecido como "Gerente do Produto", assumindo parcelas das atividades habituais do Gerente de Projetos tradicional. Esse papel é composto por um representante do cliente dentro da equipe ou por uma pessoa capacitada a sanear dúvidas sobre requisitos que surjam no dia-a-dia, pois ele faz parte da equipe e deverá ter todo o comprometimento com o projeto. É ele quem decide a data do release e o que deve conter nela.

Scrum Master

É o responsável por fazer o ambiente Scrum funcionar. É ele quem garante que o time esteja totalmente funcional e produtivo.

É o guardião do processo, assegurando que as práticas do Scrum sejam utilizadas com a disciplina necessária e assegura que o projeto e a cultura organizacional estejam ajustados para que as metas (Goal) e o ROI desejados sejam alcançados a cada Sprint (FONSECA, 2009).

Protege a equipe de interferências externas, assegura que os Sprints não contenham itens além do que realmente pode ser entregue e executa um papel de gerenciamento do tipo "técnico de futebol" ao lado dos membros do Scrum Team. Acompanha o progresso do trabalho diariamente participando de reuniões diárias, revisão do Sprint e planejamento.

O Scrum Master é um facilitador, alguém que tem a missão de fazer o time funcionar e aplicar corretamente o Scrum. Pode ser entendido como o "Gerente de Desenvolvimento" e responde pelo Product Owner, Stakeholders e Management (Alta Direção).

Scrum Team

O *Scrum Team* é responsável por transformar itens do *Product Backlog* em itens do *Sprint Backlog* e transformá-los em software pronto para ser entregue. Trata-se de uma equipe de

desenvolvimento que trabalha de forma participativa. Não existe necessariamente uma divisão funcional através de papéis tradicionais, por exemplo: o programador programa, o testador testa, o designer faz a programação gráfica, entre outros. Por se tratar de uma equipe multifuncional, caso o designer saia do projeto às telas novas não ficarão sem programação gráfica.

Um Scrum Team é composto geralmente por 5 a 9 membros embora haja relatos de projetos Scrum com equipes maiores. O Scrum Team é responsável por ficar focado nas tarefas atribuídas e quando se deparam com itens que impedem o progresso pleno, identificam qual é o impedimento e reportam ao Scrum Master.

Outra característica importante é que os times são auto-gerenciáveis, sendo responsáveis por controlar as tarefas do desenvolvimento do Sprint. Segundo Fonseca (2009), o time desenvolve de forma iterativa, realizando projeto, codificação, testes de unidade, aceitação e até documentação (JIT – *Just in Time*), para cada *Select Backlog* ("requisito") antes de passar para o próximo Sprint. Desenvolve os itens de Select Backlog rigorosamente em sistema de pilha (LSD – "pull"), do mais importante (maior ROI) para o menos importante, reforçando diariamente a formação para que cada um seja capaz de fazer qualquer item da pilha (multi-aprendizado).

Realizam reuniões diárias – *Daily Scrum* conferindo os Select Backlog realizados e atualizam o *Agile Radiator* e o *Burndown Chart* garantindo assim, sincronia nas tarefas e comunicação plena do time.

Stakeholders

Stakeholder é um termo utilizado na administração que faz referência a qualquer pessoa ou organização que afeta ou é afetada pelo projeto. A palavra vem de:

- Stake – interesse, participação, risco.
- Holder – aquele que possui.

No Scrum, os *Stakeholders* são todos os interessados no software que está em desenvolvimento a começar pelo cliente (contratante), usuários finais, equipe de marketing e vendas, entre outros e são representados pelo *Product Owner*. O Scrum apela a

todos os interessados em fornecer um feedback freqüente em todo o Sprint.

Isto produz um produto que adere mais estreitamente com as necessidades do cliente, ou seja, todos possuem um senso de propriedade. A responsabilidade pela entrega do produto é de toda a equipe, independente de papéis. Caso ocorra alguma falha durante o Sprint, todos são responsáveis pelo seu fracasso.

SCRUM – Fluxo de Processo

Segundo Fonseca (2009), o Scrum divide um projeto em interações (duas a quatro semanas no máximo), chamadas de *Sprints.*

A Figura 1 apresenta uma visão geral do processo. No início de um Sprint, durante o *Sprint Planning Meeting*, são definidas as funcionalidades do Product Backlog que serão tratadas na interação de acordo com a prioridade atribuída pelo Product Owner.

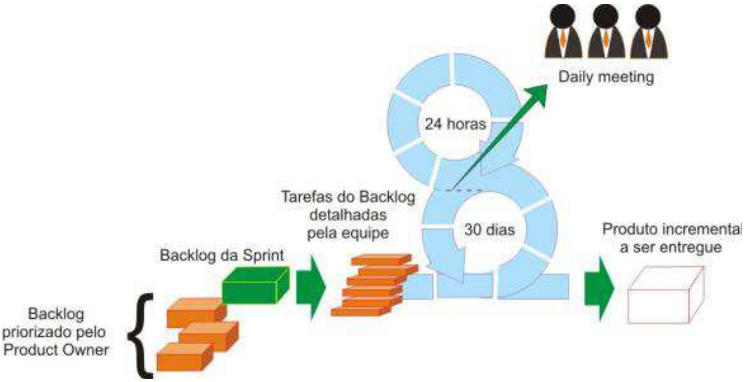

Figura 1: Fluxo de Processo Scrum
Fonte: Adaptado de The Scrum Development Process

Durante o Sprint, a equipe implementa estas funcionalidades disponibilizando-as já prontas para uso no fim da iteração.

Um Sprint termina com um *Sprint Review Meeting*, uma reunião em que as funcionalidades implementadas são demonstradas para os usuários.

Segundo Pressman (2006), *Daily Meeting* são ciclos de reuniões diárias de 15 minutos em média, com características particulares, elas são realizadas "em pé" (*Standup Meeting*) pela manhã, onde cada membro da equipe do projeto deve responder a três perguntas:

1.O que fez para o projeto desde a última reunião?

2.O que fará para o projeto até a próxima reunião?

3.Há algum obstáculo para conseguir seu objetivo? Precisa de ajuda?

SCRUM – Ciclo de Vida

Segundo Koscianski (2006), o ciclo de vida do Scrum é dividido em três fases:

- **Pré-planejamento (Pré-game Phase):** os requisitos são descritos em um documento chamado Backlog. A seguir os requisitos são classificados por prioridade, onde são estimados "o esforço" para o seu desenvolvimento. Nesta fase inclui a definição dos integrantes da equipe, identificação da necessidade de treinamento, as ferramentas a serem utilizadas, como também uma lista com os prováveis riscos de projeto. A fase é concluída com uma proposta de arquitetura de software. As alterações futuras devem ser descritas no Backlog.
- **Desenvolvimento (Game Phase):** os riscos previamente identificados devem ser mapeados e acompanhados ao longo do projeto para avaliar o seu impacto. Nesta fase, o software é desenvolvido em ciclo interativo (Sprints), onde são adicionadas novas funcionalidades. Cada um desses Sprints com duração de 2 a 4 semanas são desenvolvidos de forma tradicional (análise, projeto, implementação e testes).
- **Pós-planejamento (Post-game Phase):** Nesta fase acontece a integração do software, os testes finais e a documentação do usuário. A equipe se reúne para analisar o

estado do projeto e o software atual é apresentado ao cliente.

Para representar o ciclo de vida de um requisito são utilizadas derivações do Product Backlog, como Release Backlog e Selected Backlog (TEAMSYSTEM; ARAUJO et al, 2005, 2006).

A **Figura 2** apresenta uma visão geral do ciclo de vida do Scrum com seus artefatos.

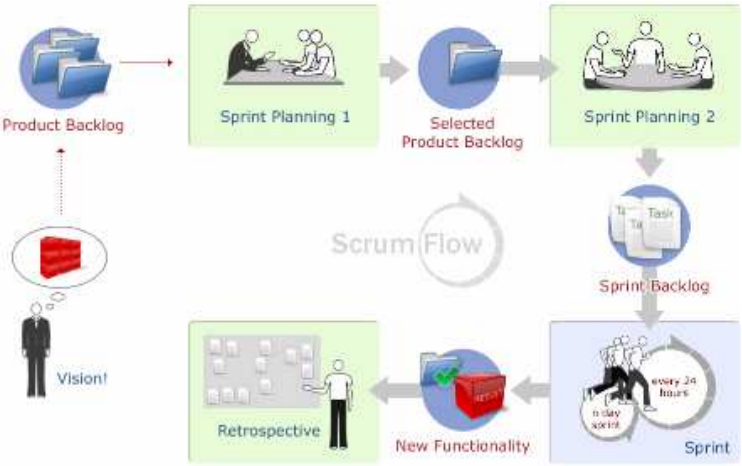

Figura 2: Visão geral do ciclo de vida do Scrum com seus artefatos
Fonte: Adaptado de The Scrum Development Process

O Scrum tem cinco artefatos (subproduto do desenvolvimento de software, por exemplo: manuais, arquivos executáveis, módulos etc.) principais, os mesmos serão descritos na continuação:

1) Product Backlog – é uma lista de todos os requisitos ordenados por prioridades de acordo com o valor do negócio. A prioridade de um ítem no backlog pode mudar, requisitos podem ser adicionados ou removidos.
2) Selected Product Backlog – é o resultado do *Sprint Planning*. Define o que o Time aceitou durante o planejamento. Ele é fixo (não pode ser modificado) durante todo o *Sprint*.

3) Sprint Backlog – é uma lista de tarefas (histórias mais relevantes) com suas respectivas estimativas de duração do presente momento, até o fim do *Sprint*. Derivado a partir do *product backlog* onde são detalhados os itens do product backlog em tarefas. É definida pelo time, e negociada com o *Product Owner*. É de extrema importância que o time mantenha esta lista sempre atualizada;

4) Burndown Chart – é um gráfico que mostra a quantidade de trabalho cumulativo restante de um *Sprint*, dia por dia. Neste gráfico, a altura indica a quantidade de tarefas do *Sprint Backlog* não completadas, e o comprimento são os dias. Este gráfico é um dos principais recursos de medição do processo de desenvolvimento e um diferencial para a metodologia SCRUM. Quanto mais horizontal, melhor.

5) Impediment Backlog – é a lista com todos os problemas (impedimentos) que atrapalham o time a progredir. Tais impedimentos devem ser resolvidos pelo *Scrum Master*. Pode ser dividida em duas listas: *Team Impediment*, que são os impedimentos que podem ser resolvidos pelo próprio time; e *Organization Impediment*, onde o time não pode resolver.

Outros autores incluem também o *Agile Radiator* (quadro branco) e os *Reports* como mecanismo de controle para acompanhar a evolução do projeto.

Ao final de um *Sprint* é entregue um *Product Increment*, isto é, uma versão do jogo, contendo todas as estórias implementadas naquele *Sprint*.

A seguir, algumas empresas que utilizam o Scrum: Siemens, Globo.com, Dell Computer, Uol, Terra, Weq, BBC, Google, Microsoft, Nokia, Philips, Nielsen, Yahoo, BmcSoftware, LexisNexis, High Moon Studios, entre outras.

Conclusão

Assim como toda Metodologia Ágil utiliza o princípio “Mudança é a única constante” - duas palavras antagônicas neste

contexto significam que os requisitos irão mudar ao longo do projeto de desenvolvimento, queira ou não.

A única coisa que se tem certeza é que os requisitos irão mudar; só não se sabe quando. Mas, a mudança (de escopo, nas regras de negócio, nas leis governamentais sejam elas Municipais, Estaduais, ou Federais) vai acontecer. A pergunta é: "O profissional está preparado para quando isto vir a acontecer"?

Todo bom profissional deve estar preparado. O SCRUM foca as pessoas que desenvolvem o software e não o processo em si, que geralmente seguem requisitos rígidos definidos no início do projeto. Isto é, difere das metodologias tradicionais (já amadurecidas) que focam o processo (documentação) e fazem a hipótese de que os requisitos não sofrerão alterações.

O desafio futuro das metodologias ágeis será encontrar meios de minimizar as suas desvantagens sem transformá-las em metodologias pesadas, como também aumentar o número de pessoas que integram a equipe sem perder a confiabilidade e eficiência no gerenciamento de mudanças.

Assim como outros processos e metodologias ágeis, o Scrum possui vantagens e desvantagens. Mas vale a pena analisar com cuidado, talvez uma saída seja a adoção de um modelo híbrido.

Bibliografia:

AGILE MANIFESTO, **Manifesto for Agile Software Development**, 2001. Disponível em <http://agilemanifesto.org/:> Acesso em: 23 mai. 2009.

ANDERSON, D. J., **Agile Management for Software Engineering, Applying the Theory of Constraints for Business Results**, Prentice Hall, 2003.

ARAUJO, A. R. S., Silva, J. M., Mittelbach, A. F., **SCRUM: Novas Regras do Jogo,** Jynix Playware, Brasil.

BOEHM, B., **A View of 20th and 21st Century Software Engineering**, ICSE 2006.

BOEHM, B. and Turner, R., **Balancing Agility and Discipline A Guide for the Perplexed**, Addison Wesley, 2003.

FONSECA, Isabella, **Engenharia de Software Conference,** São Paulo, DevMedia, 2009.

HIGHSMITH, J., **Agile Project Management, Creating innovative products**, Addison Wesley, 2004.
KOSCIANSKI, A., Soares, Santos, M. **Qualidade de Software,** São Paulo, Novatec Editora, 2006.
PRESSMAN, R., **Engenharia de Software,** São Paulo, Mc Graw-Hill, 2006.
SCHWABER K., **Agile Project Management With Scrum**, Microsoft Press, 2004.
_____ , **The Enterprise and Scrum,** Microsoft Press, 2007.
TEAMSYSTEM, **Scrum for team system,** 2005. Disponível em <http://www.scrumforteamsystem.com> Acesso em: 06 jul. 2009.