Artigo Engenharia de Software 23 - Scrum na Melhoria do Gerenciamento de Projetos de Software

Artigo da Revista Engenharia de Software edição 23.

Esse artigo faz parte da revista Engenharia de Software 23 edição especial. Clique aqui para ler todos os artigos desta edição

Metodologias Ágeis

Scrum na Melhoria do Gerenciamento de Projetos de Software

Um estudo sobre a implantação de Scrum para otimizar o processo de gerenciamento de projetos de software

De que trata o artigo:
Neste artigo veremos os principais problemas associados ao planejamento e à gestão de projetos de software. Em seguida, mostraremos como os planos são tratados por Metodologias Ágeis de desenvolvimento de software.

Para que serve:
O planejamento e o desenvolvimento de um software sob o paradigma ágil oferecem mais flexibilidade do que os modelos tradicionais para a condução de projetos de software que possuem incerteza ou instabilidade.

Em que situação o tema é útil:
Além de ser um modelo que permite a criação de planos flexíveis, as técnicas ágeis evitam desperdícios de tempo e esforço, pois focam em alcançar rapidamente meios de validar as características do produto em desenvolvimento e de avaliar a evolução do projeto.

Atualmente, devido à alta complexidade inerente ao desenvolvimento de software, os processos definidos fornecem um nível de flexibilidade menos adequado para que se obtenha alta produtividade e qualidade no produto final com as atuais práticas de engenharia. Muitos dos processos definidos existentes e voltados para esta finalidade, como o *Rational Unified Process* (RUP) da IBM (Rational), nem sempre garantem o sucesso do projeto. Tais processos de desenvolvimento de software estabelecem

quais as atividades necessárias para que o produto seja construído de forma repetível (CRUZ, 2008).
Os processos empíricos devem ser utilizados sempre que os processos definidos não forem adequados devido à complexidade do projeto. Ou seja, sempre que não se conheçam todas as variáveis de entrada para que possa estabelecer um processo repetível com a mesma saída (CRUZ, 2008).
O mercado de Tecnologia da Informação (TI) está cada vez mais competitivo, onde a rapidez na entrega de sistemas com qualidade, planejamento adaptativo, flexibilidade e rápida resposta às mudanças torna-se cada vez mais desejada. Diante desse contexto, surgiu um novo conceito de metodologia ágil na gestão de projetos de sistemas, o Scrum.
Uma alternativa possível nesta situação é a adoção de um processo que seja flexível o bastante para acomodar as alterações necessárias exigidas durante o desenvolvimento do produto, e que seja baseado em inspeção e adaptação. Quanto mais próximo do limite do caos a equipe conseguir trabalhar, porém mantendo a ordem, mais competitivo e útil será o produto resultante [CRUZ (2008), MARTINS (2007), p. 252].
O Scrum é uma metodologia ágil para gerência de projetos, baseado em inspeção e adaptação, iterativo e incremental, e pode ser aplicado a qualquer produto ou no gerenciamento de qualquer atividade complexa. Cada iteração de trinta dias conhecida como *Sprint* produz um conjunto de funcionalidades (BISSI, 2007).
Além de estudar uma metodologia que se encaixa em uma categoria já conhecida pelo mercado de TI, chamada *agile development*, iremos também inferir sobre as vantagens de se utilizar tal metodologia em relação a outros processos tradicionais e avaliar a aceitação do Scrum no mercado atual, identificando algumas empresas que já obtiveram sucesso com o seu uso.
O artigo está organizado da seguinte maneiro: primeiramente trataremos da pesquisa bibliográfica realizada sobre o tema, a discussão do histórico, ciclo de vida do processo do Scrum, seus principais artefatos, os papéis desempenhados pela equipe e a sua aceitação no mercado atual. Em seguida, apresentaremos uma avaliação da utilização do Scrum para a melhoria no gerenciamento de projetos de software, os resultados alcançados e as conclusões deste artigo.

Retrospecto do Scrum
Diversas metodologias foram criadas para sistematizar o desenvolvimento de software. Tais metodologias podem ser divididas em tradicionais, que enfatizam a documentação de cada processo do desenvolvimento de software, ou ágeis, que consideram um novo paradigma de desenvolvimento de software.
As metodologias tradicionais surgiram quando os mainframes ainda reinavam e não existiam ferramentas de apoio ao desenvolvimento, o custo de realizar algum tipo de alteração era altíssimo e a documentação tinha de ser muito mais forte. Elas estabelecem uma sequência de etapas, onde cada etapa tem um início e um fim definidos com documentação rígida a ser seguida entre uma etapa e outra, de forma que sem a documentação o processo não avança (FERSTE, 2009).
Percebeu-se que os modelos tradicionais ocultavam uma série de problemas sérios, como:

- Problema para entrega dentro do prazo considerando o escopo de todas as funcionalidades solicitadas [Standish Group, 1995];
- Grande número de erros dado que a metodologia tradicional dificulta a execução de alterações durante o processo de desenvolvimento;
- Um artigo famoso da época: "No Silver Bullet", de Brooks [1987], demonstrou que a ideia de especificar totalmente o software antes de seu início é totalmente impossível.

São considerados processos tradicionais, ou pesados, de desenvolvimento de software, aqueles que tentam prever todos os requisitos do sistema para assim ser feito um planejamento prévio de como o sistema irá atuar. Este planejamento rigoroso é representado como documentos que guiarão todo o processo de desenvolvimento (ROCHA; OLIVEIRA; VASCONSELOS, 2004).
Os processos tradicionais se caracterizam por focar a análise de sistemas, investindo esforços para a obtenção de artefatos de projeto tão completos e precisos quanto possível: relatórios, documentos e diagramas. Grande parte dos processos tradicionais em uso atualmente tem como base os modelos de diagramas da *Unified Modeling Language* (UML), como é o caso do RUP (GALDINO, 2006).
Os processos ágeis surgiram como uma alternativa aos processos tradicionais. Tais projetos partem da premissa de que se devem permitir mudanças nos requisitos a qualquer tempo. Desta forma, o planejamento é feito durante todo o processo, ao mesmo tempo em que é feita a codificação: não há um passo dedicado exclusivamente ao planejamento (ROCHA; OLIVEIRA; VASCONSELOS, 2004).
Em fevereiro de 2001 Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian

Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland e Dave Thomas, reuniram-se em *Wasatch Range* nos Estados Unidos para discutir melhores maneiras de desenvolver software. Esse encontro deu origem ao manifesto ágil, uma declaração com princípios que regem o desenvolvimento ágil:
"Estamos descobrindo maneiras melhores de desenvolver software fazendo-o nós mesmos e ajudando outros a fazê-lo. Através desse trabalho, passamos a valorizar:

- Indivíduos e interação entre eles mais que processos e ferramentas;
- Software em funcionamento mais que documentação abrangente;
- Colaboração com o cliente mais que negociação de contratos;
- Responder a mudanças mais que seguir um plano;

Ou seja, mesmo havendo valor nos itens à direita, valoriza-se mais os itens à esquerda." [TELES, 2008; SANTOS, 2009; GOMES, 2009; CORDEIRO, 2006].

A metodologia ágil é muito adequada para situações em que a mudança de requisitos é frequente, ou seja, para ser ágil, a metodologia deve aceitar a mudança em vez de tentar prever o futuro. O Scrum é uma das metodologias que se encaixa nestes conceitos e será explicado nos próximos parágrafos.
O Scrum, assim como o *Extreme Programming* (XP), são classificados como métodos ágeis de desenvolvimento de software. Como colocado por TELES (2008), este novo conceito de metodologia surgiu nos Estados Unidos na década de 90, na tentativa de criar sistemas em tempo menor, de melhor qualidade e produzidos de forma mais econômica que o habitual. Esses objetivos são alcançados através de um conjunto de valores, princípios e práticas que diferem essencialmente da forma tradicional de se desenvolver software.

Para CORDEIRO (2006), os métodos ágeis focam em simplicidade, software funcional no início das iterações, flexibilidade e intensa comunicação tanto internamente quanto com clientes. Além disso, aplicam desenvolvimento iterativo e incremental, planejamento adaptativo, flexibilidade e rápida resposta a mudanças.

O termo Scrum foi utilizado pela primeira vez aplicado no contexto de processo de desenvolvimento por Ikujiro Nonaka e Hirotaka Takeuchi em um artigo chamado *The New Product Development Game*, em português, "O jogo do desenvolvimento de novos produtos" publicado em *Harvard Business Review* em 1986. Posteriormente foi elaborado novamente em *The Knowledge Creating Company* (Oxford University Press, 1995) pelos mesmos autores (SCHWABER, 2006).

O artigo que sumariza as 10 melhores práticas em empresas japonesas escrito por Takeuchi e Nonata introduziu o termo Scrum para referir-se às reuniões de equipes que praticam a autodireção e a adaptabilidade. Jeff Sutherland é um dos criadores do Scrum e era vice-presidente da *Easel Corporation* em 1994, quando introduziu algumas das práticas do Scrum com base neste artigo. Ele também foi influenciado por um relatório de projeto com produtividade acima da média, na *Borland Corporation*, e esta juntamente com outras empresas introduziu a reunião diária. Em 1995 Ken Schwaber trabalhou com Sutherland na *Easel* e juntos formalizaram o Scrum. Este termo é o nome usado para a reunião de jogadores, no jogo do *Rugby*, quando eles se organizam em círculo para planejar a próxima jogada. É uma forma de mostrar que o projeto deve ser conduzido em pequenos ciclos, mas com uma visão de longo prazo, que é ganhar o jogo (MARTINS, 2007, p.252).

Scrum é um processo bastante leve para gerenciar e controlar projetos de desenvolvimento de software e para criação de projetos de produtos. O Scrum segue as filosofias iterativa e incremental, ele se concentra no que é realmente importante: gerenciar o projeto e criar um produto que acrescente valor para o negócio. O valor decorre da funcionalidade propriamente dita, do prazo em que ela é necessária, do custo e da qualidade (MARTINS, 2007, p. 253).

Sua abordagem é oposta a do modelo em cascata. Inicia-se a análise assim que alguns requisitos estiverem disponíveis, depois que alguma análise tiver sido feita, começam os trabalhos de projeto técnico (*design*), e assim por diante. Em outras palavras, o projeto trabalha em pequenos ciclos de cada vez (*Sprints*) até a conclusão do projeto final. Esta abordagem pode ser chamada de iterativa, e cada iteração consiste na captura de requisitos, um pouco de análise, um pouco de design e mais alguma programação e testes, culminando num ciclo iterativo com várias entregas (MARTINS, 2007, p. 253).

Ciclo do Processo do Scrum

Como o Scrum é uma metodologia ágil e todo o trabalho é dividido em iterações, cada uma delas é chamada de *Sprint*. O *Sprint* representa um ciclo, ele pode durar de duas a quatro semanas, mas é tipicamente mensal, pois 30 dias fornecem uma melhor visibilidade dos objetivos do projeto e comprometimento da equipe. No final de cada *Sprint* deve-se ter uma parte do produto concluída que possa ser apresentada ao cliente. Estas entregas parciais vão sendo implementadas até o produto final

estar concluído, e à medida que forem surgindo novas funcionalidades desejadas, novos *Sprints* vão sendo realizados.
A **Figura 1** demonstra o processo completo do ciclo de uma *Sprint* e pode ser acompanhada para auxiliar no entendimento. Um projeto *Scrum* começa mesmo que ainda se tenha uma visão superficial no princípio, talvez expressa em termos de marketing ou uma visão técnica, mas que se esclarece melhor à medida que o projeto evolui. A partir dessa visão inicial se elabora uma lista enxuta dos principais itens, de tudo o que se deseja para o software, contendo funcionalidades e requisitos que precisarão ser desenvolvidos até a finalização do projeto. Esta lista de itens é conhecida como *Product Backlog* ou "*Backlog* do Produto". Cada item tem uma prioridade de entrega que indica o quanto de valor ele gera para o negócio. Esta lista não precisa estar completa logo no começo, ela pode ganhar outros itens no decorrer do projeto.

Figura 1. Ciclo de vida de uma *Sprint* (Fonte: Adaptado de VICENTE, 2008).

No início de cada *Sprint* (iteração) a equipe seleciona os itens do *Product Backlog*, de acordo com as suas prioridades e o tempo que será gasto para completar as suas funcionalidades. Esta nova lista é conhecida como *Sprint Backlog*. É importante que a equipe identifique os itens e tamanho do *Sprint Backlog*, porque ela estará comprometida a finalizar tais tarefas. Os seus membros são quem deverão escolher com o que irão se comprometer. Nesse momento as tarefas maiores são subdivididas em partes menores.
Como descrito em CISNEIROS (2009), logo após o *Sprint Backlog* estar concluído, o total de horas trabalhadas é comparado com o total previsto anteriormente no *Product Backlog*. Caso haja uma diferença significativa, a equipe Scrum deve negociar novamente com o cliente o número correto de horas, para que o *Sprint* seja realizado com maior probabilidade de sucesso.
Em MITTELBACH; SILVA; ARAUJO (2006), os autores subdividem cada *Sprint* nas seguintes atividades:

- **Scrum Planning Meeting**: É o início de uma *Sprint*, onde o *Product Owner* (representante do cliente ou o próprio cliente) tem a oportunidade de atualizar a priorização dos itens do *Product Backlog* e definir juntamente com a equipe um *Product Increment* (uma parte do produto) a ser entregue ao cliente ao final do *Sprint*.
  Esta reunião é realizada com a presença do *Product Owner*, *Scrum Master* (líder facilitador do projeto), de todo *Scrum Team* (equipe), e quaisquer interessados, diretores ou representantes do cliente. Durante a reunião o *Product Owner* explica as funcionalidades de maior prioridade para o *Scrum Team*, e este faz as perguntas que sejam suficientes para que eles possam, depois da reunião, definir quais atividades eles irão mover do *Product Backlog* para o *Sprint Backlog*.
  Depois da reunião *Sprint Planning*, o *Scrum Team* reúne-se separadamente para discutir o que foi dito e decidir o quanto eles se comprometem a fazer durante o próximo *Sprint*. Em alguns casos, haverá negociações com o *Product Owner*, mas será sempre prerrogativa do *Scrum Team* determinar o quanto eles podem se comprometer;
- **Daily Scrum Meeting**: É um encontro diário realizado pela equipe e o *Scrum Master* onde os

membros discutem aquilo em que trabalharam, no que irão trabalhar e possíveis impedimentos que estejam atrapalhando o progresso do trabalho. Esta reunião é uma maneira eficiente de manter os membros cientes dos objetivos e impedir que o projeto "saia do rumo".
São tipicamente rápidas e objetivas, realizadas em pé, preferencialmente pela manhã, duram de 15 a 30 minutos, para evitar perder o foco do que está sendo desenvolvido e possíveis divergências de assuntos. Elas não representam uma forma de cobrança vinda de um gerente de projetos, mas é uma maneira de sincronizar a equipe às tarefas e relatar os impedimentos que podem estar interferindo no bom andamento do *Sprint*.
Nas reuniões diárias, os únicos autorizados a falar são o *Scrum Master* e a equipe envolvida. Os demais participantes devem apenas ouvir, se falarem algo devem ser convidados a se retirar da reunião, pois podem querer mudar o rumo do projeto, tentando priorizar itens que são importantes para eles e não para o projeto ou seu objetivo imediato. O *Scrum Master* é o responsável por manter a ordem nessa reunião, dar a palavra a um membro por vez, evitar desvio de foco do assunto e conversas paralelas (MARTINS, 2007, p. 274).

- **Criação do Product Increment**: A finalização das funcionalidades definidas para um determinado *Sprint* marca a realização de um *Product Increment*. Os membros da equipe trabalham de maneira colaborativa de forma a realizar todas as metas definidas para aquele *Sprint*;
- **Sprint Review**: Ocorre no final de cada *Sprint*, neste momento a equipe exibe o *Product Increment*, potencialmente utilizável que foi construído, ao *Product Owner*, que é responsável por validar e/ou solicitar ajustes para que o projeto se torne adequado aos anseios do cliente. Os participantes desta reunião incluem tipicamente: o *Product Owner*, o *Scrum Team*, o *Scrum Master*, a diretoria, clientes e engenheiros de outros projetos. O ideal é que a equipe tenha concluído todos os itens do *Product Backlog* alocados para o *Sprint*.
Segundo MARTINS (2007, p. 276) é uma reunião informal de quatro horas de duração, que ajuda a equipe de forma colaborativa, a obter um *feedback* e determinar quais os objetivos da próxima iteração, assim como a qualidade e as capacidades das funcionalidades produzidas. Ao final da apresentação das funcionalidades desenvolvidas, os *stakeholders* são questionados, um a um, quanto às suas impressões, mudanças necessárias e alterações de prioridades. Possíveis rearranjos nos itens do *Product Backlog* ou funcionalidades que não foram entregues como esperado podem ser incluídas no próximo *Sprint*;
- **Sprint Retrospective**: Após a reunião de revisão do *Sprint*, o *Scrum Master* faz uma reunião de retrospectiva com a equipe. Esta objetiva identificar os pontos positivos e negativos do *Sprint* que entregou o último *Product Increment* e busca corrigir os problemas encontrados com o propósito de aperfeiçoa-los.
Nessa reunião é debatido o que funcionou na *Sprint*, o que precisa ser melhorado e quais ações devem ser tomadas para colocar as melhorias em prática. Geralmente o *Sprint Retrospective* tem de três a quatro horas de duração (AGUIAR, 2008).
- **Atualização do Product Backlog**: O *Product Owner* é responsável por re-priorizar toda lista de itens do *Product Backlog* para que um próximo *Sprint* possa ser iniciado de acordo com os itens mais prioritários.

Os principais artefatos produzidos no processo do Scrum são o *Product Backlog*, *Sprint Backlog, Burndown Chart* e o *TaskBoard*. Eles serão exemplificados nas **Tabelas 1**, **2** e **3** e nas **Figuras 2** e **3**.
Para melhor exemplificar o ciclo do processo do Scrum, consideremos o exemplo citado em CISNEIROS (2009), sobre o desenvolvimento de um sistema web de *streaming* de vídeo para internet. O escopo do projeto em questão é: "Um sistema Web de *streaming* de vídeo, que permitirá aos usuários na Internet enviar seus vídeos, que serão armazenados no sistema e poderão ser gerenciados por seus donos e vistos pelo resto do mundo através das visitas ao site. O sistema terá como funcionalidades principais a conversão dos vídeos mandados para um *codec* leve que permite qualidade e rapidez na visualização pela Internet; cadastro de usuários; interface de gerenciamento de vídeos; comentários para os vídeos e usuários; notas para vídeos; sistema de busca; reconhecimento de sons dos vídeos; proteção contra SPAM; sistema de legenda de vídeos feitos por usuários; canais (grupos) de vídeos e usuários".
Com as funcionalidades levantadas, o *Product Owner* monta o *Product Backlog* com as prioridades definidas de acordo com o valor de importância para o cliente. Em nosso exemplo, usaremos a notação de quanto maior o número de prioridade, menor prioridade ele tem (1 - prioridade máxima, 2 - prioridade menor, e assim por diante). A **Tabela 1** demonstra a primeira versão feita do *Product Backlog*, enquanto a **Tabela 2** apresenta o mesmo *Product Backlog* mais detalhado, com o custo-horas que cada tarefa irá ocupar.

Funcionalidade	Prioridade
Modelagem de dados	1
Cadastro e gerenciamento de usuários	2
Conversão de vídeo para visualização na Internet (Codec)	3
Cadastro e gerenciamento de vídeos pelos usuários	4
Layout (Aparência) da página	5
Comentários para os vídeos e usuários	6
Notas para vídeos (*ranking*)	7
Proteção contra SPAM	8
Canais (grupos) de vídeos e usuários	9
Sistema de legendagem de vídeos	10
Reconhecimento de sons dos vídeos	11

Tabela 1. *Product Backlog* (Fonte: Adaptado de CISNEIROS, 2009).

Funcionalidade	Prioridade	Custo-horas
Modelagem de dados	1	32
Cadastro e gerenciamento de usuários	2	26
Conversão de vídeo para visualização na Internet (Codec)	3	80
Cadastro e gerenciamento de vídeos pelos usuários	4	48
Layout (Aparência) da página	5	28
Comentários para os vídeos e usuários	6	20
Notas para vídeos (ranking)	7	16
Proteção contra SPAM	8	20
Canais (grupos) de vídeos e usuários	9	26
Sistema de legendagem de vídeos	10	64
Reconhecimento de sons dos vídeos	11	92

Tabela 2. *Product Backlog* mais detalhado (Fonte: Adaptado de CISNEIROS, 2009)

Com o novo *Product Backlog* definido na **Tabela 2**, define-se quais serão as metas do primeiro *Sprint*:

- Modelagem de dados;
- Cadastro e gerenciamento de usuários;
- Conversão de vídeo para visualização na Internet (*Codec*);
- Cadastro e gerenciamento de vídeos pelos usuários.

Sendo assim, o *Scrum Master* junto à equipe de desenvolvimento define o *Sprint Backlog* da **Tabela 3**, subdividindo as tarefas maiores em menores.

Funcionalidade	Prioridade	Custo-horas
Modelagem de dados	**1**	**32**
Definição de dados	1.1	8
Organização de tabelas	1.2	12
Relacionamento	1.3	8
Implementação em SGBD	1.4	4
Cadastro e gerenciamento de usuário	**2**	**26**
Formulários	2.1	6
Interação com cadastro na base de dados	2.2	6
Visualização de perfil	2.3	6
Mudança de dados	2.4	4
Relacionamento entre usuários	2.5	4
Conversão de vídeo para visualização na Internet (Codec)	**3**	**80**
Definição e Implementação de Codec	3.1	56
Integração de Codec com sistema	3.2	24
Cadastro e gerenciamento de vídeos pelos usuários	**4**	**48**
Upload de vídeos	4.1	16
Remoção de vídeos	4.2	14
Perfis de vídeos	4.3	18

Tabela 3. *Sprint Backlog* (Fonte: Adaptado de CISNEIROS, 2009).

O *Burndown Chart* é o gráfico de andamento do *Sprint,* relacionando horas e data em relação ao restante de tempo hábil para o término do ciclo (VICENTE, 2008).
Com o *Burndown Chart* podemos ver claramente o andamento do projeto ao longo do seu ciclo de desenvolvimento (*Sprint*). Também, no meio do projeto, podemos calcular facilmente a velocidade com que o projeto está andando e assim estimar uma data para que o *Sprint* seja concluído. Observe a **Figura 2**.

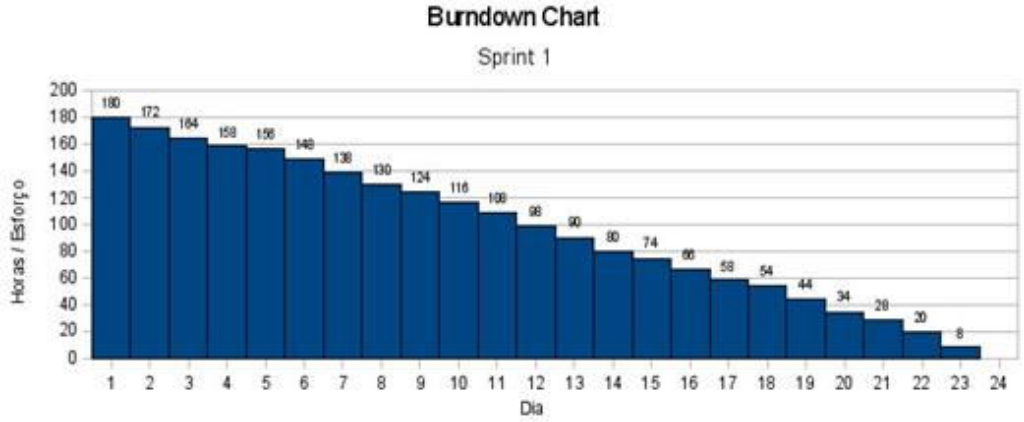

Figura 2. *Burndown Chart* (Fonte: Adaptado de CISNEIROS, 2009).

Este dado estimativo pode ser comparado com o prazo que o *Scrum Master* definiu para que possamos saber se o projeto vai acabar ou não no prazo. Por exemplo, calculamos que a velocidade do projeto está como 8 horas por dia em média. Isso significa que o projeto irá terminar no dia 24, como o *Burndown Chart* mostrou. Se o prazo fosse, por exemplo, para o dia 19, logo nos cinco primeiros dias já poderíamos ter calculado essa velocidade e ver que estava pouco, tomando providências para que essa velocidade se tornasse, por exemplo, 10 horas por dia em média, assim no dia 19 o projeto estaria pronto.
Este é um dos mais importantes trabalhos que o *Scrum Master* terá que fazer e o *Burndown Chart* é o indicador perfeito para ele gerenciar o tempo de projeto e sua equipe de desenvolvimento (CISNEIROS,

2009).
O *taskboard* (**Figura 3**) é um grande painel onde podem ser colocadas várias informações importantes para o acompanhamento do S*print*. O *Sprint Backlog*, ou seja, as atividades não iniciadas, as que estão em andamento e as concluídas ficam sempre visíveis e disponíveis para todos os interessados no projeto.
Algumas características do *taskboard* são:

- Normalmente é desenhado em uma parede e as atividades são descritas em *post-its*;
- Apresenta uma visão geral do S*print*;
- Fica acessível a todos os interessados no projeto (KNIBERG, 2007).

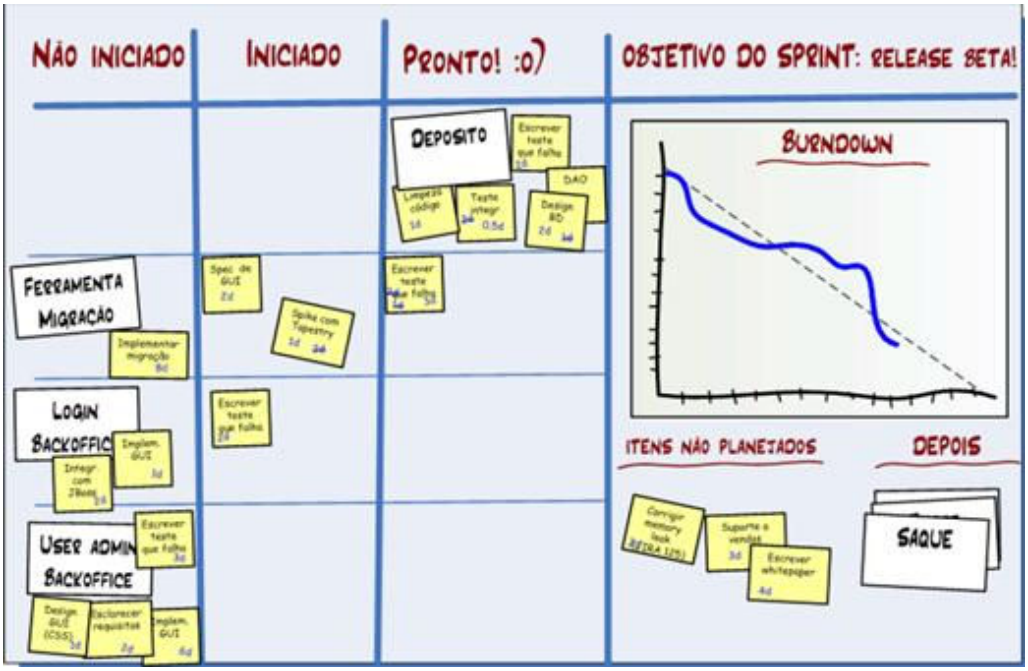

Figura 3. *TaskBoard* (Fonte: Adaptado KNIBERG, 2007)

Papéis da equipe no Scrum
O Scrum trabalha basicamente com três papéis: o *Product Owner*, o *Scrum Master* e a equipe do projeto, também chamada de *Scrum Team*.
O *Product Owner* é o responsável pela visão de negócios e por representar os interesses das pessoas que apostam no projeto, provavelmente será um gerente de projeto, ou um patrocinador do projeto, um membro da equipe de marketing ou um cliente interno. É ele quem define e prioriza o *Product Backlog* e consegue a verba. Geralmente é o papel desempenhado pelo cliente (MARTINS, 2007, p. 255).
Suas principais responsabilidades são:

- Definir as funcionalidades do produto;
- Concentrar as informações vindas de usuários, *stakeholders* ou do mercado de maneira que se obtenha uma visão única dos requisitos do sistema;
- Sua maior responsabilidade é o *Return on Investment* (ROI) do projeto;
- Priorizar o *Product Backlog*;
- Decidir sobre a data de término;
- Ajustar recursos e priorizar tarefas a cada *Sprint*, como necessário;
- Aceitar ou rejeitar os resultados dos trabalhos.

O *Scrum Master* é uma mistura de gerente de projeto, facilitador e mediador. Ele é um líder e não somente um gerente, está sempre em contato com o *Product Owner* (SANCHEZ, 2007).
Entre as suas principais responsabilidades, temos:

- Assegurar que a equipe de desenvolvimento funcione plenamente e seja produtiva;
- Ajudar na cooperação entre todas as funções e papéis do time;
- Remover obstáculos da equipe e assegurar que as práticas de Scrum estão sendo executadas com eficiência pelas pessoas envolvidas. Alguns obstáculos podem ser resolvidos pelo time, outros podem ser resolvidos através de vários times, e outros podem precisar de envolvimento da gerência, pois podem ser problemas relacionados à empresa que bloqueiam qualquer membro do time a fazer qualquer coisa. Como exemplo deste tipo de impedimento externo, temos as questões judiciais;

- Proteger a equipe de interferências externas;
- Assegurar-se de que a metodologia está sendo seguida, incluindo chamadas para reuniões diárias (*Daily Scrum Meeting*), revisões de atividade (*Sprint Reviews*) e reuniões de planejamento das atividades (*Sprint Planning*);
- Identificar quais atividades foram concluídas, quais foram iniciadas, quaisquer novas tarefas que foram descobertas e qualquer estimativa que possa ter mudado. Isto permite a ele atualizar sempre o *Burndown Chart*, que permite mostrar quanto trabalho falta para um *Sprint* acabar, dia por dia. Ele também tem que sempre olhar cuidadosamente para o número de tarefas em aberto. Estes trabalhos em aberto devem ser minimizados o máximo possível para garantir um trabalho sempre limpo e eficiente;
- O *Scrum Master* deve perceber e resolver problemas pessoais ou conflitos entre os integrantes do time. Este tipo de problema deve ser resolvido por diálogo com o time, ou então o *Scrum Master* terá que ter ajuda da gerência ou do departamento de Recursos Humanos. Alguns estudos apontam que 50% dos problemas de desenvolvimento acontecem por razões pessoais. O *Scrum Master* precisa estar sempre atento ao time para fazer dele totalmente funcional e produtivo [LIMA, 2008; CISNEIROS, 2009; AGUIAR, 2008; VICENTE, 2008; SANTOS, 2009].

Por último, a equipe do projeto ou *Scrum Team* é o grupo de pessoas responsáveis por desenvolver ou construir as funcionalidades do produto. Algumas características das equipes de desenvolvimento são:

- São autogerenciadas, auto-organizadas e multifuncionais;
- São equipes pequenas, compostas por 5 a 10 membros (o recomendado são 7 pessoas). Podem ser compostas por desenvolvedores e participantes externos (marketing, vendas, clientes, etc.);
- Demonstrar o resultado do *Sprint* para o *Product Owner* e outros *Stakeholders*;
- Deve ter a capacidade e o conhecimento técnico sobre todo o processo de desenvolvimento do produto;
- O time deve ter pessoas capazes de analisar a solução, codificá-la e testá-la sem necessitar de outros times ou outras pessoas;
- Definem metas de cada *Sprint*, junto ao *Scrum Master*, e especificam seus resultados de trabalho;
- Têm o direito de fazer tudo dentro dos limites das diretrizes do projeto para atingir a meta de cada *Sprint*;
- Organizam os trabalhos para atingir os objetivos dos *Sprints* [LIMA, 2008; CISNEIROS, 2009; AGUIAR, 2008; VICENTE, 2008; SANTOS, 2009].

Aceitação do Scrum no mercado atual

O mundo enfrenta uma das piores crises financeiras de sua história. O Fundo Monetário Internacional (FMI) divulgou um relatório no ano de 2009, onde fez uma previsão de que a economia mundial encolheu 1,3%. Mesmo o Brasil, que fortaleceu sua economia e vem resistindo à recessão, também teve previsão de retração destes mesmos índices.

Por consequência, os investimentos em tecnologia sofreram uma queda de 3 ou 4% em 2009, em previsões das consultorias Forrester e Gartner, respectivamente. Os gastos com programas de computador deverão permanecer praticamente estáveis, já que software é visto como algo que pode ajudar as empresas a economizarem, mas os investimentos em hardware e serviços de TI fecharão o ano em queda.

O mercado de projetos vem se reconfigurando diante da crise. Os projetos de tecnologia, em particular, sofrem um forte impacto. Neste ambiente de incertezas e constantes mudanças, os atores do mercado de projetos já encontram racionalização do uso de recursos, acesso limitado a crédito, pressões por margens menores e problemas financeiros com grande parte dos clientes, tornando o processo decisório mais longo e gerando uma notável redução na demanda por projetos. Esta nova configuração exige que as organizações mudem sua forma de trabalhar para conseguirem sobreviver, se fazendo necessária uma verdadeira quebra de paradigma. Essa nova forma de trabalhar deverá:

- Funcionar bem em ambientes que mudam rapidamente, permitindo replanejamento frequente;
- Focar-se em maximizar o retorno do investimento (ROI) do cliente;
- Ajudar a reduzir o tempo de entrada em produção ou o tempo de lançamento do produto no mercado;
- Evitar desperdício de esforço e tempo com subprodutos e funcionalidades que nunca serão utilizados;
- Sempre entregar valor para o cliente, mesmo que o projeto seja interrompido antes do seu final previsto;

- Priorizar a comunicação e feedback entre as pessoas do projeto, para que todos saibam o que deve ser feito e o que está sendo feito.

O Scrum é focado em lidar com todas estas questões, e se apresenta como uma das melhores opções para projetos em momentos de crise (SABBAGH; GARRIDO, 2009).
SCHWABER (apud NETO, 2008, p. 10) afirma que apesar de o Scrum ser uma metodologia relativamente nova, a demanda por sua adoção tem crescido fortemente e o Scrum Master, por ser o responsável pelos objetivos do projeto e um dos principais atores na realização do Scrum, precisa conhecer profundamente suas técnicas e processos para que obtenha êxito em sua utilização.
De acordo com a publicação de MARCH (2009), o Scrum está virando febre entre empresas brasileiras. O conceito que chegou ao Brasil há três anos e agora começa a chamar a atenção, inclusive de áreas além da computação. Algumas das empresas internacionais de grande porte que já utilizam o Scrum são: *Microsoft*, *Yahoo*, *Google*, *Philips*, *Siemens*, *Nokia*, *BBC*, *Salesforce.com*, *Capital One, Oce, Sabre, John Deere, First American Real Estate, BMC Software, Oracle, Motorola, HP, Coca-Cola,* entre outras (COHN, 2002; BROD, 2008; FUSCO, 2008). Algumas empresas brasileiras que também já estão usando o Scrum são a *Globo.com*, *UOL*, *Cemig, Tribunal de Justiça de Rondônia* (TJRO), *MicroPower*, *Crivo, Summus, Synchro, YMF, Infraero*, *Instituto Municipal de Tecnologia da Informação de Campo Grande e Venturus* (YOSHIMA, 2009).

Uma avaliação da utilização do Scrum para a melhoria no gerenciamento de projetos de software

Os processos de desenvolvimento de software definidos ou também chamados de tradicionais são às vezes fatores limitadores aos desenvolvedores. Muitas empresas não possuem recursos ou inclinação para processos pesados de produção de software, por isso muitas acabam por não usar nenhum processo, o que pode ocasionar efeitos desastrosos em termos de qualidade de software.
Como alternativa a esse cenário, surgem as metodologias ágeis, que apesar de possuírem documentação, são mais flexíveis, orientadas a entregas e possuem uma maior iteratividade no processo de desenvolvimento e codificação do produto. Preocupam-se em gastar menos tempo com documentação e mais com a implementação. São adaptativas ao invés de serem preditivas, elas se adaptam a novos fatores decorrentes do desenvolvimento do projeto, ao invés de procurar analisar previamente tudo o que pode acontecer no decorrer do desenvolvimento.
A maioria das metodologias ágeis nada possui de novo, o que as diferencia das metodologias tradicionais são o enfoque e os valores. A ideia das metodologias ágeis é o enfoque nas pessoas e não em processos ou algoritmos.
Para ser considerada ágil a metodologia deve aceitar a mudança ao invés de tentar prever o futuro. Elas variam em termos de práticas e ênfases, mas compartilham algumas características, como desenvolvimento iterativo e incremental, comunicação e redução de produtos intermediários, como documentação extensiva. Dessa forma, existem maiores possibilidades de atender aos requisitos do cliente, que muitas vezes são mutáveis.
As metodologias pesadas devem ser aplicadas apenas em situações em que os requisitos do software são estáveis e requisitos futuros são previsíveis. Estas situações são difíceis de serem atingidas, uma vez que os requisitos para o desenvolvimento de um software são mutáveis. Dentre os fatores responsáveis por alterações nos requisitos estão a dinâmica das organizações, as alterações nas leis e as mudanças pedidas pelos *stakeholders*, que geralmente têm dificuldades em definir o escopo do futuro software.
Ao contrário do que alguns imaginam, as metodologias ágeis não rejeitam os processos e ferramentas, a documentação, a negociação de contratos ou o planejamento, mas simplesmente mostra que eles têm importância secundária quando comparado com as pessoas e interações, com o software estar executável, com a colaboração do cliente e as respostas rápidas a mudanças e alterações.
É um equívoco entender que o Scrum não enfatiza o planejamento. O planejamento no seu contexto é dividido em vários *Sprints*. Se a tarefa for muito complexa, ela pode ser dividida em subtarefas que são então alocadas em dois ou mais *Sprints*, onde no primeiro faz-se o planejamento e no segundo acontece a execução.
As características do projeto irão determinar entre o uso das abordagens tradicional ou ágil. Os projetos que têm como natureza a inovação tecnológica inviabilizam o uso da abordagem tradicional, pois o risco de ser necessário alterar um produto depois da conclusão de uma fase de seu ciclo de vida é alto. A abordagem ágil consegue uma adaptação mais fácil nesses casos de mudança. A abordagem tradicional é mais adequada para projetos que necessitam de um forte planejamento e muita disciplina no processo, mas ela não promove tão ampla comunicação entre os membros de equipes e seus gerentes.
O Scrum como sendo uma das principais metodologias ágeis e foco deste artigo, tem como ideia

principal que o desenvolvimento de softwares envolve muitas variáveis técnicas e do ambiente, como requisitos, recursos e tecnologia, que podem mudar durante o processo. Isto torna o processo de desenvolvimento imprevisível e complexo, requerendo flexibilidade para acompanhar as mudanças.
O Scrum se encaixa muito bem principalmente para produtos com características mutáveis durante o projeto. Como existem projetos de natureza empírica que não funcionam corretamente com processos bem definidos, o Scrum entra como possível solução, sendo um processo definido para projetos empíricos com características muito dinâmicas.
Um processso Scrum bem implementado poderá alcançar o efeito *Toyota*: quatro vezes mais produtividade e doze vezes mais qualidade. Um dos motivos principais no ganho de produtividade é que sua organização faz com que um Scrum funcional produza resultados de ótima qualidade de negócio na primeira vez, evitando que se façam componentes que nunca serão usados pelo usuário. Esse problema atinge muitas organizações que não implementam Scrum (SCHWABER, 2006).
Outra vantagem de se utilizar o Scrum é a produtividade aliada ao conforto. Ele permite aos colaboradores desenvolverem software na medida em que reduzem a pressão de administração, pois oferece liberdade para que cada membro da equipe selecione seus próprios trabalhos e organizem entre si.
Como toda metodologia ou processo, sempre existem vantagens e desvantagens advindas da sua adoção. Algumas das desvantagens observadas no Scrum são:

- Não é fácil de ser implementada, principalmente devido a resistência de mudanças culturais;
- Ausência de práticas de Engenharia de Software, pois é voltada para o gerenciamento do projeto e não para o desenvolvimento, podendo necessitar da associação de outras metodologias de Engenharia de Software simultaneamente;
- Pode ser mais difícil de se aplicar em grandes equipes ou em equipes geograficamente distribuídas;
- A documentação do software pode acabar sendo desprezada, tornando insuficiente quando forem necessárias manutenções futuras;
- Sensação de informalidade, pois a documentação formal do escopo do software só é criada caso os envolvidos considerem necessário.

Todavia, diante de tantas vantagens já discutidas nos parágrafos anteriores, o Scrum ainda se mostra como uma opção capaz de driblar vários problemas atuais no gerenciamento de projetos de software. Os principais benefícios obtidos com a prática do Scrum na intenção de obter uma melhoria nos processos de gerenciamento de software, podem ser assim resumidos:

- Melhor adaptação no gerenciamento de projetos com constantes mudanças, pois a grande maioria dos projetos se inicia sem total detalhamento e sempre existem alterações nos requisitos ao longo do desenvolvimento;
- Encoraja a comunicação entre os membros da equipe e o espírito colaborativo. As chances de sucesso em projeto são muito maiores em times onde há uma boa comunicação, onde há entendimento e cooperação mútua;
- Por ser iterativo e incremental, permite a entrega antecipada de uma parte funcional do software ao cliente, que já poderá validar se está conforme o esperado. Caso não esteja, a equipe não perderá o mesmo tempo que perderia se esse feedback só fosse feito após a conclusão do software. Além de impedir que se implemente funcionalides que nunca serão utilizadas pelo cliente;
- A entrega do software utilizando Scrum é mais rápida do que se utilizasse alguma outra metodologia tradicional, uma vez que a codificação já se inicia logo nos primeiros *Sprints*, e não se espera ter antes uma extensa documentação do software;
- A qualidade no software utilizando Scrum é promovida pelos seguintes fatores: as iterações, a remoção de impedimentos, inspeção e adaptação, times multifuncionais e autonomia da equipe, o que significa menos pressão sobre cada membro;
- O Scrum traz um ROI mais rápido, pois minimiza a burocracia dos processos e se aproxima do cliente. Este está sempre envolvido, participando da demonstração do software ao final de cada *Sprint*, oferecendo *feedback* e atento às mudanças e suas consequências.

Resultados
As metodologias ágeis mesmo não tendo muito tempo de existência, já apresentam resultados efetivos. CHARETTE (apud SOARES, p.6) cita um exemplo interessante comparando as metodologias ágeis e tradicionais: um estudo mostrou que os projetos usando os métodos ágeis obtiveram melhores resultados em termos de cumprimento de prazos, de custos e padrões de qualidade. Além disso, o

mesmo estudo mostra que o tamanho dos projetos e das equipes que utilizam as metodologias ágeis têm crescido. Apesar de serem propostas idealmente para serem utilizadas por equipes pequenas e médias (até 12 desenvolvedores), aproximadamente 15% dos projetos que usam metodologias ágeis estão sendo desenvolvidos por equipes de 21 a 50 pessoas, e 10% dos projetos são desenvolvidos por equipes com mais de 50 pessoas, considerando um universo de 200 empresas usado no estudo.
O estudo acima demonstra que é possível obter sucesso com o Scrum mesmo em equipes maiores, mas para isso é necessário empenho e motivação das pessoas envolvidas, treinamentos e disciplina para adotar os processos desta metodologia. Os benefícios são claros e são muitos, porém, não sem muito esforço, recursos e uma completa reformulação na cultura da organização.
Vale ressaltar que as práticas do Scrum podem ser aplicadas em qualquer contexto, no qual pessoas precisem trabalhar juntas para atingir um objetivo comum. Ele é recomendado para projetos nas áreas de software, automotiva, telecom e, principalmente, de pesquisa e inovação. Apresenta-se ideal para projetos dinâmicos e suscetíveis a mudanças de requisitos, sejam eles novos ou apenas modificados. No entanto, para aplicá-lo, é preciso entender antes seus papéis, suas responsabilidades, seus conceitos e os artefatos das fases do ciclo.
A abordagem ágil assim como a abordagem tradicional possui características positivas e negativas, sendo que a principal diferença entre as duas está no conjunto de pressupostos de cada uma. É possível afirmar, ainda, que existe uma sinergia muito grande entre as duas metodologias, ou seja, uma pode complementar a outra.
Observa-se que não existe uma metodologia perfeita, assim como não há uma solução única para todos os casos. O projeto de software tem muito a ganhar com metodologias ágeis e suas práticas, assim como quaisquer projetos de inovação, com alto grau de mudanças, escopo não muito bem definido ou que precisam de resultados no curto prazo.
Por outro lado, as situações em que existem fortes requisitos contratuais, amarrando fortemente prazos e escopo, ou quando a organização já possui maturidade e experiência com projetos semelhantes, podem ser candidatos para uma abordagem mais tradicional.

Conclusão

O mercado de software tem crescido a cada dia e junto com ele crescem também as exigências quanto à complexidade, custos, prazos e qualidade dos sistemas, sendo todas estas questões cruciais e de difícil gerenciamento. No contexto da Tecnologia da Informação, principalmente quando se trata de processos de desenvolvimento de sistemas, a dinamicidade sempre se faz presente, o que explica o surgimento de metodologias de desenvolvimento ágil com a finalidade de atender as necessidades atuais do mercado de software e proporcionar maiores vantagens em relações às metodologias tradicionais.
O Scrum é uma metodologia ágil para gerência de projetos baseado em inspeção e adaptação, iterativo e incremental, e pode ser aplicado a qualquer produto ou no gerenciamento de qualquer atividade complexa. O processo do Scrum como uma metodologia ágil e ciente das questões críticas relativas a software estabelece algumas alternativas na tentativa de driblar os problemas existentes no desenvolvimento de software.
As metodologias ágeis têm ganhado espaço ultimamente, e várias empresas já estão experimentando ou se mostrando interessadas em experimentar esta mais recente abordagem de desenvolvimento de software. Esse interesse nasce da tentativa de aperfeiçoar os processos, aumentar a qualidade dos produtos finais e a satisfação dos clientes.
A aceitação do Scrum no mercado de software tende a crescer, pois além de ser uma metodologia recente, ela está inserida no contexto atual de crise mundial que afeta também o mercado do software. Em tempos de crise, o Scrum desponta como uma boa alternativa para reduzir gastos, enfrentar as exigências e complexidade dos softwares e a competitividade das empresas.
Uma vez superado estes tempos de crise mundial, as organizações que tiverem adotado Scrum estarão mais próximas do cliente, enxutas, focadas em resultado, objetivas e transparentes. Assim, no momento da recuperação do mercado, estas organizações largarão à frente de sua concorrência.
Várias empresas de grande porte, tanto internacionais como nacionais obtiveram sucesso com o uso do Scrum. Um dos casos de sucesso mais conhecido por esta implantação no Brasil é o da Globo.com.
O processo que compreende as fases do ciclo de vida do Scrum mostrou que a metodologia consegue ser iterativa, incremental, adaptativa a mudanças e ao mesmo tempo completa em cada uma das suas fases. Ele abrange todas as atividades que se fazem necessárias para um bom gerenciamento de projetos de software, enfatizando sempre as frequentes reuniões que ajudam a alinhar os objetivos entre todos os envolvidos, inclusive o cliente.
Portanto, o Scrum serve como um guia de boas práticas para o alcance do sucesso. O conhecimento das suas práticas permite uma aplicação de forma variada, e este é um dos seus aspectos positivos – a

adaptabilidade. Vale ressaltar que suas práticas podem ser aplicadas em qualquer contexto, no qual pessoas precisem trabalhar juntas para atingir um objetivo comum. O Scrum torna-se ideal para projetos dinâmicos e suscetíveis a mudanças de requisitos, sejam eles novos ou apenas modificados, e para isso é necessário entender bem seus papéis, suas responsabilidades, seus conceitos e cada fase do seu ciclo.

Referências

AGUIAR, Luiz. **Introdução ao Scrum.** 2008. Disponível em: <http://www.gonow.com.br /blog/introducao-ao-scrum> Acesso em: 09 ago 2009.
BISSI, Wilson. **SCRUM – Metodologia de Desenvolvimento Ágil**. 2007. 6 p. Centro Universitário de Maringá, Maringá. Disponível em: < http://docs.google.com/gview?a=v&q=cache%3AmoxlrNDQQGsJ%3Arevista.grupointegrado.br%2Frevista%2Findex.php%2Fcampodigital%2Farticle%2Fview%2F312%2F146+iterativo+e+incremental+e+pode+ser+aplicado+a+qualquer+produto+ou+no+gerenciamento+de+qualquer+atividade+complexa&hl=pt-BR&gl=br&pli=1 > Acesso em: 13 ago 2009.
BROD, Cesar. **As raízes do Scrum**. 2008. Disponível em: <http://www.brod.com.br/?q=node/366> Acesso em: 08 jul 2009.
Brooks F. **No Silver Bullet: Essence and Accidents of Software Engineering**. Computer 20(4):10-19, 1987.
CISNEIROS, Hugo. **Modelo de Desenvolvimento Ágil**. 2009. Disponível em: <http://www.devin.com.br/modelo-scrum/> Acesso em: 09 ago 2009.
COHN, Mike. **Uma introdução ao SCRUM**. 2002. Disponível em: < http://www.mountaingoatsoftware.com/system/asset/file/44/PortugueseScrum.ppt> Acesso em: 30 jun 2009.
CORDEIRO, Lucas. **Metodologia de Desenvolvimento Ágil Scrum: O caso XMPM**. Manaus; 2006. Disponível em: <http://users.ecs.soton.ac.uk/lcc08r/artigos/metodologia-agil-scrum-XMPM.ppt > Acesso em: 26 jun 2009.
CRUZ, Leandro Rodrigo Saad. **Desenvolvimento de Software com Scrum.** 2008. Disponível em: **<** http://www.assembla.com/spaces/projeto_ambap_ufal/documents/csaP3ANYqr3A1_ab7jnrAJ/download/DesenvolvimentodeSoftwarecomScrum.pdf **>** Acesso em: 12 ago 2009.
FERSTE, Maurício Antônio. **Metodologias Ágeis**. 2009. Disponível em: < http://waltercunha.com /blog/index.php/2009/06/29/metodologias-ageis/> Acesso em: 07 jul 2009.
FUSCO, Camila. **Scrum: a nova gestão de projetos**. 2008. Disponível em: < http://governanca.wordpress.com/2008/03/18/scrum-a-nova-gestao-de-projetos/> Acesso em: 07 jul 2009.
GALDINO, Cárlisson Borges Tenório. **AWA - Um Processo Ágil para Desenvolvimento de Aplicações Web Atômicas**. 2006. 37 p. Monografia (Pós-Graduação Latu Sensu em Produção de Software). Universidade Federal de Lavras. Lavras. Disponível em: <http://64.233.163.132 /search?q=cache:RXu31pB_sr4J:repositorio.viadigital.org.br/frs/download.php/7/awa.pdf+Grande+parte+dos+processos+tradicionais+em+uso+atualmente+tem+como+base+os+modelos+de+diagramas+da+[[http://www.uml.org/+|+UML]],+como+%C3%A9+o+caso+do+RUP&cd=8&hl=pt-BR&ct=clnk&gl=br&client=firefox-a> Acesso em: 12 ago 2009.
GOMES, André Faria, **Desenvolvendo com Agilidade**. 2009. Revista Java Magazine, Edição 68. Disponível em: < http://www.devmedia.com.br/space.asp?id=217030 > Acesso em: 12 ago 2009.
KNIBERG, Henrik. **Scrum e XP direto das Trincheiras : como nós fazemos Scrum**. 2007. Disponível em: <http://www.infoq.com/resource/minibooks/scrum-xp-from-the-trenches/pt/pdf /ScrumeXPDiretodasTrincheiras.pdf>. Acesso em: 24 jun 2009.
LIMA, Lucimara Benigno de. **Papéis no Scrum**. 2008. Disponível em: <http://lucimarabenigno.wordpress.com/2008/07/31/papeis-no-scrum/> Acesso em: 14 ago 2009.
MARCH, Rodrigo. **Método que prega metas de curto prazo e interação de equipe ganha mercado**. 2009. Disponível em: http://oglobo.globo.com/economia/seubolso/mat/2009/02/14/metodo-que-prega-metas-de-curto-prazo-integracao-de-equipe-ganha-mercado-754418293.asp> Acesso em: 16 ago 2009.
MARTINS, José Carlos Cordeiro. **Técnicas para gerenciamento de projetos de software**. Rio de Janeiro, Editora Brasport, 1ª Edição, 2007.
MITTELBACH, Artur F; SILVA, Juliana M.; ARAUJO, Allan R. S. **Scrum: Novas Regras do Jogo**. 2006. Disponível em: <http://www.cin.ufpe.br/~sbgames/proceedings/files/Scrum.pdf> Acesso em: 10 ago 2009.
NETO, Erasmo Isotton. **Scrumming: Ferramenta Educacional para Ensino de Práticas do SCRUM**.

2008. 80p. Monografia (Bacharelado em Sistemas de Informação). Pontifícia Universidade Católica do Rio Grande do Sul, Porto Alegre. Disponível em: <http://64.233.163.132 /search?q=cache:fJ7X_ZmpYFYJ:www.inf.pucrs.br/~rafael/Scrumming/Scrumming.pdf+uso+scrum+tem +crescido&cd=6&hl=pt-BR&ct=clnk&gl=br&lr=lang_pt&client=firefox-a> Acesso em: 17 ago 2009.
ROCHA, Thayssa Águila da; OLIVEIRA, Sandro Ronaldo Bezerra; VASCONCELOS, Alexandre Marcos Lins de. **Adequação de Processos para Fábricas de Software**. 2004. Disponível em: < http://www.simpros.com.br/Apresentacoes_PDF/Artigos/Art_12_Simpros2004.pdf > Acesso em: 13 ago 2009.
SABBAGH, Rafael; GARRIDO, Marcos. **Scrum e a Crise Mundial – Por que Scrum é melhor opção para projetos em tempos de crise**. 2009. Disponível em: <http://www.infoq.com/br/articles/scrum-crise-mundial> Acesso em: 16 ago 2009.
SANCHEZ, Ivan. **Scrum em 2 minutos**. 2007. Disponível em: < http://dojofloripa.wordpress.com /2007/02/07/scrum-em-2-minutos/> Acesso em: 06 jul 2009.
SANTOS, Rildo F. **SCRUM Experience**. 2009. Disponível em: < http://www.etecnologia.com.br/scrum /Tutorial%20SCRUM%20v14.pdf> Acesso em: 12 ago 2009.
SCHWABER, Ken. **Scrum Et Al**. 2006. Disponível em:<http://engenharia.criarumblog.com/Primeiro-blog-b1/Srum-b1-p5.htm> Acesso em: 26 jun 2009.
SOARES, Michel dos Santos. **Comparação entre Metodologias Ágeis e Tradicionais para o Desenvolvimento de Software.** 2009. Disponível em: <http://wiki.dcc.ufba.br/pub/Aside /ProjetoBitecIsaac/Met._Ageis.pdf > Acesso em: 20 ago 2009.
Standish Group, CHAOS Report. Dennis, MA 02638, USA, 1995.
TELES, Vinícius Manhães. **Extreme Programming**. 2008. Disponível em: < http://improveit.com.br/xp> Acesso em: 23 jun 2009.
TELES, Vinícius Manhães. **SCRUM**. 2008. Disponível em: <http://improveit.com.br/scrum > Acesso em: 30 jun 2009.
VICENTE, Dinei. **O que é SCRUM?**. 2008. Disponível em: <http://www.tuister.com.br/scrum/> Acesso em: 10 ago 2009.
YOSHIMA, Rodrigo. **Valores do Desenvolvimento de Software**. 2009. Disponível em: http://www.aspercom.com.br/ead/mod/resource/view.php?id=272 Acesso em: 14 ago 2009.

por Renata de Souza Alves Paula

Bacharel em Sistemas de Informação pela Universidade Estadual de Goiás, Anápolis, Goiás, e Pós-Graduada em Tecnologia da Informação e Negócios Eletrônicos pela Universidade Salgado de Oliveira, Goiânia, Goiás. Atualmente atua como analista de sistemas no Instituto Federal de Educação, Ciência e Tecnologia de Goiás.