

Entendendo Scrum para Gerenciar Projetos de Forma Ágil

Paulo Pereira¹; Paula Torreão², Ana Sofia Marçal³

¹ Paulo Pereira

² Paula Torreão

³ Ana Sofia Marçal

C.E.S.A.R

Gerente de Projetos


¹ paulo.pereira@C.E.S.A.R..org.br

² paula.coelho@C.E.S.A.R..org.br

³ ana.sofia@C.E.S.A.R..org.br

Resumo - A melhor forma de ser ágil é construir somente o que o cliente valoriza e não mais que isto. O excesso de formalidade pode limitar o progresso do projeto, mas por outro lado, o caos total, sem a utilização de processos pode impedir que se alcancem os objetivos do projeto. Scrum permite criar produtos melhor adaptados à realidade do cliente de forma ágil. Além do mais, praticar Scrum nos projetos traz grandes benefícios para a equipe como comprometimento, motivação, colaboração, integração e compartilhamento de conhecimento, o que facilita em muito o gerenciamento e sucesso dos projetos. Este artigo explica porque e como usar o Scrum nos projetos.

(Palavras-chave: Scrum, abordagens ágeis, gerenciamento ágil de projetos)

Artigo Aprovado	Versão: <1.8>	
-----------------	---------------	---

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 mudanças na tecnologia da informação, pressões por constantes inovações, concorrência acirrada e grande dinamismo no ambiente de negócios (BOEHM, 2006).

Apesar de existir há um bom tempo, apenas recentemente a expressão “Métodos Ágeis” vem se tornando mais popular no Brasil por usar uma abordagem simplificada. No entanto, “ser simples” geralmente é confundido com falta de controle e completa anarquia. Na verdade, ser simples, ter agilidade, é fazer a diferença e, ao contrário do que parece, exige muita disciplina e organização.

Agilidade quer dizer “a habilidade de criar e responder a mudanças, buscando a obtenção de lucro em um ambiente de negócio turbulento” (HIGHSMITH, 2004); ou ainda, a capacidade de balancear a flexibilidade e a estabilidade. HIGHSMITH (2004) enfatiza que a ausência de estrutura ou estabilidade pode levar ao caos, mas que a estrutura em demasia gera rigidez.

A abordagem ágil aplicada ao desenvolvimento de projetos ficou mais clara e melhor definida a partir de 2001, quando um grupo de 17 autores e representantes das mais variadas técnicas e metodologias ágeis como, eXtreme Programming (XP) , Scrum, *Dynamic System Development Method* (DSDM), *Adaptive Software Development* (ASD), *Crystal*, *Feature Driven Development* (FDD) e *Lean Development*, se reuniu para discutir e identificar o padrão de desenvolvimento de projetos dentre as técnicas e metodologias existentes. O resultado desse encontro foi a criação do Manifesto para Desenvolvimento Ágil de Software (*Agile Manifesto*, 2001), que estabeleceu um *framework* comum para processos ágeis valorizando os seguintes itens:

Indivíduos e interações sobre **Processos e ferramentas**

Software funcionando sobre **Extensa documentação**

Colaboração dos clientes sobre **Negociação em contratos**

Respostas à mudanças sobre **Seguir um plano**

Ou seja, embora haja valor nos itens acima à direita, são mais valorizados os **à esquerda sublinhados**.

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).

Dentro deste contexto, nosso artigo tem o propósito de mostrar como estamos entendendo e usando o Scrum (SCHWABER, 2002), método ágil que vem ganhando grande visibilidade nos últimos 5 anos, em projetos de desenvolvimento de software, ressaltando benefícios como comprometimento da equipe, motivação, colaboração, integração e compartilhamento de conhecimento, o que facilita em muito o gerenciamento e sucesso dos projetos.

Vantagens de ser ágil

Compreender os valores do *Agile Manifesto* traz novas sugestões para a melhoria de métodos, processos e técnicas de desenvolvimento e gestão de projetos. Dentro deste contexto o uso da agilidade traz vantagens como:

- cria um ambiente propício para definição de mudanças de requisitos e inovação durante o ciclo de desenvolvimento do produto, assim como mais colaborativo e produtivo entre desenvolvedores e cliente, resultando em entregas mais rápidas de produto, melhor adaptados à realidade do cliente e com a qualidade esperada.

- facilita o gerenciamento do projeto, uma vez que existe maior integração e comprometimento da equipe do projeto, que consequentemente se sente mais motivada: a moral da equipe é elevada.

- reforça o planejamento constante do projeto, o que minimiza os riscos,

considerando que o planejamento é mais importante do que o plano. Não se deve parar de planejar até que se tenha encontrado a satisfação do cliente com a entrega do produto final.

- valoriza a satisfação do cliente em primeiro lugar.

Como começar a ser ágil em seu projeto?

Várias opções para começar a praticar a agilidade em projetos de software são encontradas na literatura destacando-se, Scrum, *Extreme Programming* (XP), Dynamic System Development Method (DSDM), *Crystal* e *Lean Development*. **Uma pesquisa recente sobre o estado do desenvolvimento ágil (VERSIONONE, 2007) indica que 70% dos entrevistados usam Scrum combinado com outras metodologias e 40% usam Scrum puro.**

Por que Scrum?

Scrum é bastante objetivo, com papéis bem definidos, de fácil adaptação e ainda, sua curva de aprendizado é relativamente baixa.

Segundo o seu autor SCHWABER (2004), o Scrum não é um processo previsível, ele não define o que fazer em toda circunstância. Ele é usado em trabalhos complexos nos quais não é possível prever tudo o que irá ocorrer e oferece um *framework* e um conjunto de práticas que torna tudo visível. Isso permite aos praticantes do Scrum saber exatamente o que está acontecendo ao longo do projeto e fazer os devidos ajustes para manter o projeto se movendo ao longo do tempo visando alcançar os seus objetivos.

Logo, o Scrum não vai dizer exatamente o que fazer, não irá resolver todos os seus problemas, mas com certeza os problemas serão mais facilmente identificados. Por ser um *framework*, irá servir como um guia de boas práticas para atingir o sucesso. Entretanto, as decisões de quando e como usá-lo, quais táticas e estratégias seguir para obter produtividade e realizar as entregas ficam por conta de quem estiver aplicando. O conhecimento das suas práticas permite a aplicação das mesmas de forma variada e este é um dos aspectos positivos do Scrum, a adaptabilidade.

Vale ressaltar que as práticas do Scrum podem ser aplicadas em qualquer contexto onde pessoas precisem trabalhar juntas para atingir um objetivo comum. **Scrum é recomendado para projetos de outras áreas além de software e principalmente para projetos de pesquisa e inovação.**

Como funciona o Scrum?

A figura 01 ilustra o ciclo de desenvolvimento do Scrum de forma simplificada.

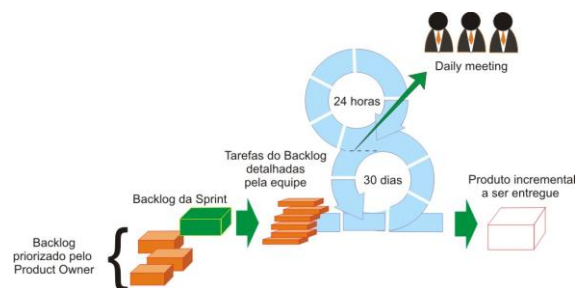



Figura 01: Ciclo do Scrum (adaptado de *The Scrum Development Process*)

O ciclo do Scrum tem o seu progresso baseado em um série de iterações bem definidas, cada uma com duração de 2 a 4 semanas, chamadas **Sprints**. Antes de cada *Sprint*, realiza-se uma **Reunião de planejamento (Sprint Planning Meeting)** onde o time (equipe) de desenvolvedores tem contato com o cliente (**Product Owner**) para priorizar o trabalho que precisa ser feito, selecionar e estimar as tarefas que o time pode realizar dentro da *Sprint*. A próxima fase é a **Execução da Sprint**. Durante a execução da *Sprint*, o time controla o andamento do desenvolvimento realizando **Reuniões Diárias Rápidas (Daily Meeting)**, não mais que 15 minutos de duração, e observando o seu progresso usando um gráfico chamado **Sprint Burndown**. Ao final de cada *Sprint*, é feita uma revisão no produto entregue para verificar se tudo realmente foi implementado.

Ao final da *Sprint*, deve-se realizar uma **Reunião de Revisão (Sprint Review)**, onde o time demonstra o produto gerado na *Sprint* e valida se o objetivo foi atingido. Logo em seguida, realiza-se a **Reunião de Retrospectiva (Sprint Retrospective)**, uma reunião de lições aprendidas, com o objetivo

Artigo Aprovado	Versão: <1.8>	
-----------------	---------------	---

de melhorar o processo/time e/ou produto para a próxima *Sprint*.

Scrum torna-se ideal para projetos dinâmicos e suscetíveis a mudanças de requisitos, sejam eles novos ou apenas requisitos modificados. No entanto, para aplicá-lo, é preciso entender antes os seus papéis, responsabilidades, conceitos e artefatos das fases de seu ciclo.

Papéis e Responsabilidades do Scrum

O Scrum implementa um esqueleto interativo e incremental através de três papéis principais: o Product Owner, o Scrum Team, e o ScrumMaster como descrito abaixo:

• Product Owner

- Define os requisitos do produto, decide a data de release e o que deve conter nela.
- É responsável pelo retorno financeiro (ROI) do produto.
- Prioriza os requisitos de acordo com o seu valor de mercado.
- Pode mudar os requisitos e prioridades a cada *Sprint*.
- Aceita ou rejeita o resultado de cada *Sprint*.

• ScrumMaster

- Garante que o time esteja totalmente funcional e produtivo.
- Facilita a colaboração entre as funções e áreas e elimina os impedimentos do time.
- Protege o time de interferências externas.
- Garante que o processo está sendo seguido. Participando das reuniões diárias, revisão da *Sprint*, e planejamento.

• Scrum Team

- Multi-funcional, entre 5-9 membros.
- Seleciona, entre os itens priorizados, os que irão ser executados durante a *Sprint*.
- Tem todo o direito de realizar o que quiser dentro da *Sprint*

para cumprir o objetivo da iteração.

- Auto-organizado: Organiza o time e o trabalho entre os membros de forma participativa.
- Ao final da *Sprint*, realiza o demo do produto finalizado.

Conceitos, Artefatos e Fases do Scrum

Sprint

No Scrum o progresso do projeto é baseado em uma série de iterações bem definidas chamadas *Sprints*. A recomendação é de que a duração da *Sprint* fique entre 2 a 4 semanas. Esse período é chamado de **Time-box**.

O Scrum é baseado em entrega de produto a cada *Sprint*, logo é recomendável que durante o desenvolvimento do produto, exista um padrão, um mesmo *time-box*, para todas as *Sprints*. Isto é importante para se conseguir medir o progresso e a produtividade da equipe no projeto.

Product Backlog

Um dos conceitos mais importantes é o *Backlog* do produto (ou *Product Backlog*), e o *Backlog* de Impedimentos (*Impediment Backlog*), ambos considerados o coração do Scrum.

O *Product Backlog* contém uma lista de itens priorizados que incluem tudo o que precisa ser realizado, que possa ser associado com valor de negócio, para a finalização do projeto, sejam requisitos funcionais ou não. É importante ressaltar que cada item no *Backlog* do produto deve ter um valor de negócio associado (*Business Value*), onde podemos medir o retorno do projeto e priorizar a realização dos itens.

O *Impediment Backlog* contém todos os itens que impedem o progresso do projeto e geralmente estão associados a riscos. Estes itens não possuem uma priorização, mas estão geralmente associados a algum item de *Backlog* do produto ou a tarefas do item, Exemplo “instalar ambiente para desenvolvedores”, “Instalação de banco de dados do projeto”, etc. O controle desses itens é muito importante e o *ScrumMaster* é o

grande responsável pela liberação desses impedimentos, abrindo caminho para o time de desenvolvimento executar a realização dos itens do *Backlog* do produto.

Existem muitas formas para gerenciar o *Product Backlog* e o *Impediment Backlog* todas elas precisam que cada item seja identificado e estimado, em tempo ou tamanho, e que a sua ordem de importância seja estabelecida pelo cliente. Com esses atributos é possível ter os itens em uma ordem de priorização.

Preparando o *Product Backlog*

Antes de iniciar a reunião de planejamento, é preciso ter o *Backlog* priorizado e estimado. Uma técnica muito interessante e objetiva conhecida como *Planning Poker* pode ser usada onde a estimativa pode ser feita em horas/tamanho. A seguir, será apresentada a estimativa por tamanho.

Planning Poker

O *Planning Poker* (COHN,2006) é uma forma de estimativa em conjunto, podendo ser feita como um jogo (Figura 02). Todos os membros do time, inclusive o *Product Owner*, participam de forma democrática para chegar a um consenso de estimativa, para cada item do *Backlog*, de forma objetiva e divertida.

O primeiro passo do “jogo” é fornecer para cada membro da equipe um conjunto de cartas com uma seqüência numérica. A seqüência de *Fibonacci* (1,2,3,5,8,13,21, etc.) é usada. Existe uma lógica associada a escolha dessa seqüência, pois os *gaps* são maiores a medida que os números aumentam, deixando claro a diferença entre os itens a medida que eles se distanciam. Uma vez que os *gaps* não são lineares eles expressarão melhor as incertezas associadas as estimativas de grande dificuldade.

O jogo é iniciado selecionando o item de *Backlog* que o *Product Owner* e o time acreditam que seja o mais fácil de implementar, e a ele é associado o menor número da seqüência. Depois o próximo item é selecionado, e é comparado o seu grau de dificuldade com os dos itens já estimados. Neste momento, cada participante da reunião seleciona uma carta, onde se acredita ter o grau de dificuldade associado ao item e o

participante espera que todos os outros participantes terminem as suas seleções. Quando todos selecionaram as suas cartas, as mesmas são exibidas. E assim, havendo um consenso geral nas seleções feitas, o número da carta que corresponde ao tamanho (Size) é associado ao item.

Em caso de divergência, é necessário que os participantes expliquem os motivos de sua escolha, para que todos possam refletir e validar a sua escolha. Finalizada a discussão realiza-se uma nova rodada para que todos tenham a oportunidade de reavaliar seus julgamentos.

Esse processo segue até que seja encontrado um consenso. É importante que exista um moderador para que as discussões sejam produtivas e não polemizadas. Esse ciclo é seguido até que todos os itens do *Backlog* sejam estimados.

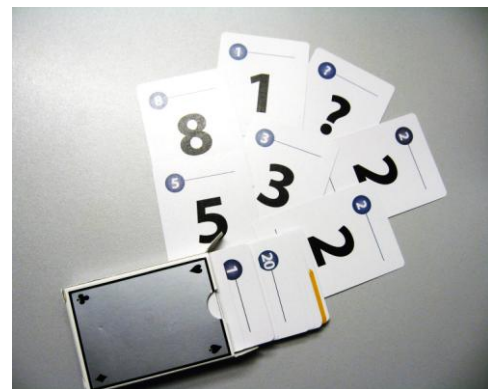


Figura 02: Exemplo de cartas que podem ser usadas no *Planning Poker*

Planejamento da *Sprint*

A atividade que precede o início da *Sprint* chama-se *Sprint Planning Meeting* (reunião de planejamento da *Sprint*). Essa atividade é muito importante e precisa de alguma preparação. Deve-se ter cuidado para que essa reunião não extrapole a duração planejada e seu objetivo. É lógico que a equipe precisa de tempo para poder estimar com segurança, mas ela deve ser sempre guiada pelo *ScrumMaster* para que seja produtiva e não disperse e perca o foco.

A primeira lição aprendida no planejamento da *Sprint* é a de que o *Product Backlog* deve estar pronto antes de cada reunião. Mas o que significa isso? Que todos os requisitos têm que estar perfeitamente

definidos? Que as estimativas iniciais devem estar corretas? Que todas as prioridades estão finalizadas? Não, precisamos apenas:

- Que o *Product Backlog* exista e cada item esteja estimado.
- Deve existir apenas um *Product Backlog* e um *Product Owner*.
- Todos os itens devem ter uma nota em função de sua importância. Essa nota serve apenas para organizar os itens por importância.
- O *Product Owner* deve *entender* todos os itens do *Product Backlog*, normalmente ele é o autor do mesmo, mas em alguns casos outras pessoas podem ter colocado itens no Backlog.

Ele não precisa saber como cada item deverá ser feito, mas precisa saber o motivo do item estar lá. Apenas o *Product Owner* tem o poder de associar a nota de importância de cada item do *Product Backlog*.

Estando preparados os itens acima, pode-se iniciar a reunião de planejamento. O objetivo da reunião é priorizar os itens que serão executados na *Sprint*, além de dar ao time informação suficiente para que o mesmo possa validar e estimar o esforço em horas para cada item. Vale lembrar que esta reunião é muito crítica para o sucesso do projeto e que uma reunião mal planejada pode afetar e muito o andamento da *Sprint* e causar danos grandes ao prazo do projeto.

Com o *Product Backlog* priorizado o time seleciona os itens que acham possível de executar durante a *Sprint*. As dúvidas do time são esclarecidas e ao final temos então o *Sprint Backlog*, que são os itens do *Backlog* priorizados para a *Sprint*.

Para cada item, o time inicia o detalhamento de suas atividades, estimando em horas, a duração de cada uma delas. Não é mandatório, mas é sugerido que cada tarefa não ultrapasse a duração de 16h, se for necessário, deve-se quebrar a tarefa até que individualmente todas as tarefas estejam dentro do limite de 16h. Uma vez que todas as tarefas foram estimativas, o time questiona se realmente consegue assumir o compromisso de realizar as tarefas dentro da *Sprint* e finalizar o item selecionado. Uma vez decidido o item, o time passa para o próximo e esse processo continua até que todos os itens do *Sprint Backlog* sejam validados.

Nesse momento não são alocados os recursos para as tarefas; apenas se estabelece as estimativas em horas para cada uma. Após a estimativa refinada, pode-se calcular o total de horas necessário para realizar todas as tarefas. É importante deixar sempre uma folga, já que mesmo detalhando a estimativa sempre podem aparecer surpresas.

Uma vez ajustados os valores e com o time se comprometendo com a execução das tarefas, existe um ambiente completo para a produção do resultado final da *Sprint*. O próximo passo é iniciar a execução da *Sprint*.

É importante lembrar que a *Sprint* possui um limite de horas disponível. Este limite é conhecido por LHS (Limite de Horas da *Sprint*).

Limite de Horas da Sprint (LHS)

O Limite de Horas da *Sprint* (LHS) pode ser medido utilizando uma fórmula simples:

$$LHS = (R \times H) \times D,$$

Onde: R = Total de recursos do time
H = Total de horas disponível para cada recurso
D = Total de dias úteis da *Sprint*

Note que se você tiver o H variando para alguns recursos, é importante considerar isso na fórmula.

A relação de 01 homem/dia efetivo = 06 homens/hora efetivas é mais real dependendo do tipo do projeto. Isso significa que se deve considerar apenas seis horas efetivas de cada recurso das oito normalmente trabalhadas, apenas 75% do tempo real do recurso é considerado produtivo para a *Sprint*. Esta sobra é importante, pois fornece uma idéia mais realista da produtividade de cada recurso e também garante uma margem de segurança para eventuais imprevistos.

Com a simulação a seguir pode-se entender melhor como calcular o LHS:

Para uma *Sprint* com *time-box* padronizado em 5 dias e um time possui 5 recursos onde 3 são de 08h, um de 06h e outro de 08h mas apenas alocado por 50% de seu tempo, temos o LHS diário considerando a relação acima sugerida de:

$8h \times 75\% = 6h$ (3)
 $6h \times 75\% = 4,5h$ (1)
 $8h \times 50\% = 4h \times 75\% = 3h$ (1)
 Dias úteis da *Sprint* = 5

logo,

$((3 \times 6) + (4,5) + (3)) = 31,50$ (dia)

$LHS = 5 \times 31,50 = 157,50$

Estes são cálculos bem simples e você vai precisar usá-los apenas antes de cada planejamento da *Sprint*, pois o LHS é que vai indicar se você está alocando as horas corretas para o seu time.

Tamanho da Sprint

É importante salientar que uma vez estabelecido a duração da *Sprint* não se deve alterar sua data final, pois todo o projeto é guiado por essa duração. Logo, adiantar ou atrasar não é interessante para ninguém.

Outro motivo importante para se manter o padrão de duração da *Sprint* é a relação de produtividade do time dentro da *Sprint*. A métrica de esforço (*velocity*) para o grupo de Itens que foram priorizados na *Sprint* é que vai dizer se conseguimos medir para um mesmo *time-box*, o total de itens que foram finalizados por *Sprint*, e isso comparado a um histórico. Esta métrica é muito importante, e apenas será real se as *Sprints* forem de tamanhos iguais.

Execução e Controle da Sprint

Durante a *Sprint*, o time, de forma organizada, controla como as tarefas devem ser executadas. Durante a *Sprint* não deve existir interferência externa, esse é um dos principais papéis do *ScrumMaster*, blindar o time de qualquer desvio do objetivo traçado. O acompanhamento do progresso é feito realizando reuniões diárias (*daily meeting*). Essas reuniões devem ser rápidas, não mais que 15 minutos e objetivas. Todos participam, o *ScrumMaster* e o time. Visitantes são bem vindos, mas devem ser apenas ouvintes, pois o *daily meeting* resume-se ao time.

No *Daily Meeting* a equipe trabalha apenas três perguntas, que são respondidas por cada membro:

- 1) O que foi feito desde ontem?
- 2) O que você planeja fazer para amanhã?
- 3) Você tem algum impedimento?

Todo e qualquer problema encontrado durante o *Daily Meeting* deve ser tratado em uma outra reunião, apenas com os envolvidos.

Durante a execução da *Sprint*, a alocação de recursos para cada tarefa é dirigida pelo próprio time, cada membro da equipe seleciona as tarefas que podem realizar e o time estabelece a ordem e dependência entre elas. **Um fator importante de sucesso para o time com o uso do Scrum é o controle da disciplina. O time é considerado auto-gerenciável, e deve responder como tal.** Para isto todos os membros do time devem reportar as horas utilizadas em tarefas para que o valor de horas restantes seja calculado corretamente e o time possa verificar o seu progresso. Para esse acompanhamento o time usa um gráfico chamado de *Sprint Burndown*.

O *Sprint Burndown* exibe o progresso diário do time em função do total de horas estabelecido pela soma de horas das tarefas dos itens do *Product Backlog* selecionados. A figura 03 mostra um exemplo:

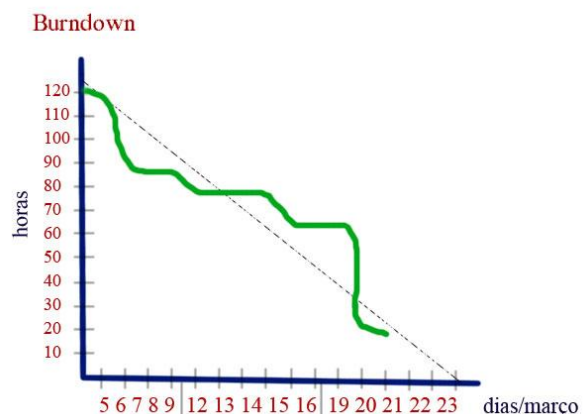


Figura 03 – Burndown da Sprint (adaptado de *Scrum and XP from the Trenches, How we do Scrum*)

O *Burndown* é um gráfico muito simples que indica o consumo de horas diárias. O eixo X indica a escala de horas totalizando o valor de horas estimado para a *Sprint*, e o eixo Y os dias que representam o tamanho da *Sprint* de acordo com seu *time-box*.

Mesmo se existir um comportamento linear para os dias, a soma de horas pode oscilar para cima. Este comportamento indica que algumas estimativas foram erradas ou novas tarefas foram adicionadas, de uma forma ou de outra, o total de horas naquele momento aumentou.

Estas oscilações podem ser perigosas e precisam ser acompanhadas diariamente. Os gráficos das figuras 04 e 05 ilustram alguns dessas oscilações.

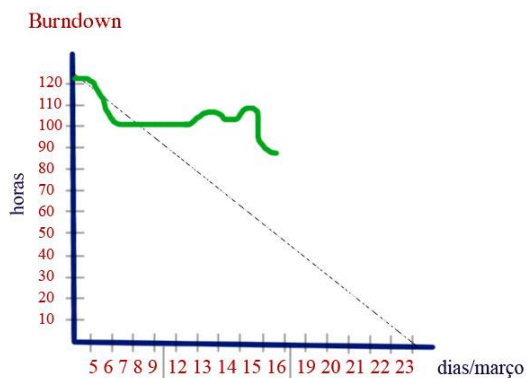


Figura 04 - Burndown da Sprint indicando atraso na execução das tarefas (adaptado de *Scrum and XP from the Trenches, How we do Scrum*)

O gráfico acima indica que a *Sprint* irá acabar depois da data final. Logo, é preciso rever as estimativas, pois muitos imprevistos não apareceram durante a preparação da *Sprint*, e a indicação é que não será possível terminar o que foi previsto. Talvez seja necessário rever os itens do *Product Backlog* remanescentes e eliminar aqueles com grau de importância menor.

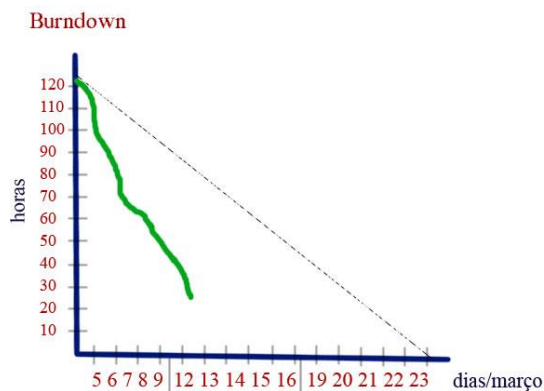


Figura 05 - Burndown da Sprint indicando adiantamento na execução das tarefas

(adaptado de *Scrum and XP from the Trenches, How we do Scrum*)

A figura 05 ilustra uma situação inversa. Neste caso é preciso adicionar novos itens do *Product Backlog* pois o gráfico indica que a *Sprint* será concluída antes do esperado. Em ambos os casos as decisões sempre devem ser tomadas junto com o *Product Owner*, nunca apenas pelo time.

O time também possui um “quadro de trabalho” onde organiza as atividades, dos itens de Backlog da *Sprint*, separando-as em basicamente em quatro estados: Para fazer, Em andamento (inclui o nome do responsável por executar a tarefa), Para Verificar e Concluído. Esse “quadro” é muito produtivo, pois basta olhar para ele para realizar a leitura do progresso da *Sprint*, inclusive nele pode-se indicar se existe algum impedimento para que as atividades sejam executadas conforme planejado. Veja sugestão na Figura 06.

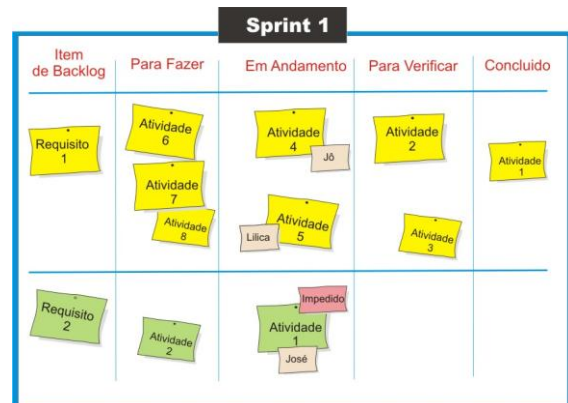



Figura 06 – Sugestão de quadro de trabalho para acompanhamento do projeto usando Scrum

Final da Sprint

Ao final de cada *Sprint*, acontece a reunião de revisão. Nela o time entrega o produto testado e revisado realizando uma demonstração prática. Este é o momento do *Product Owner* inspecionar o produto final e verificar se o mesmo está como foi pensando. Após esta reunião, a reunião de retrospectiva é realizada, podendo ser chamada também de “lições aprendidas”. Nesta, o time levanta tudo de bom e ruim que ocorreu e procura estabelecer pontos de melhoria. Estas ações

Artigo Aprovado	Versão: <1.8>	
-----------------	---------------	---

são levadas para a próxima *Sprint* melhorando o processo e/ou produto.

O ciclo do Scrum é repetido até que todos os itens do *Product Backlog* tenham sido finalizados, atendidos e/ou o produto final tenha sido aceito pelo cliente.

Resultados práticos do uso do Scrum

Aplicar Scrum traz várias mudanças, principalmente culturais na empresa em que trabalhamos. Experimentamos inicialmente Scrum em um projeto de pesquisa, desenvolvimento e inovação que tinha como objetivo construir 7 jogos complexos para celular, com prazo de 5 meses, e equipe composta por 35 pessoas.

O primeiro passo para aplicar Scrum foi criar um grupo de estudos dentro da equipe para aprofundar os conhecimentos sobre o assunto, e depois desenvolver um treinamento para toda equipe com o objetivo de nivelar o conhecimento adquirido.

O segundo passo foi dividir as equipes e formar as células de trabalho, lembrando que no Scrum existe a recomendação de equipes de tamanho de seis a dez membros. Como eram 7 jogos, foram criadas 7 células de 6 a 8 participantes, e cada célula tinha uma composição de perfis que tornava a mesma independente. Cada *Sprint* teve sua duração definida em duas semanas, assim a cada rodada teríamos que entregar uma parte incremental do produto testado e funcionando.

Em seguida, o plano de iteração (do total de *Sprints*) foi montado e apresentado ao cliente. O cliente foi envolvido desde o início, mesmo estando fisicamente afastado a mais de 3.000 km teve uma participação ativa de forma remota. E o próximo passo foi executar o projeto.

Os resultados foram muito satisfatórios, tivemos grande sucesso na aplicabilidade do Scrum em nosso projeto. Entregamos o projeto dentro do prazo e orçamento previsto. Percebemos que realmente Scrum é ideal para projetos dinâmicos e suscetíveis a mudanças de requisitos, sejam eles novos ou apenas requisitos modificados.

O que mudou após usar Scrum?

A equipe evoluiu profissionalmente se tornando muito mais segura com relação à capacidade de estimativa e autogerenciamento. Esse crescimento foi gradativo ao longo das *Sprints* e com a segurança aumentou o comprometimento e o foco com o projeto. Além do mais, a equipe, depois de experimentar Scrum, quer sempre que possível, seguir as suas práticas nos novos projetos.

O cliente reconheceu que participar ativamente do desenvolvimento do projeto traz grandes resultados e redução de riscos e custos. Ele ficou muito mais satisfeito por estar participando ativamente do processo de desenvolvimento, validando as entregas, solicitando modificações dentro do prazo hábil para realizá-las e vendo que a cada produto entregue, suas expectativas estavam sendo atendidas. **As entregas parciais permitem uma melhor avaliação do cliente com relação ao que ele realmente deseja, eliminando-se funcionalidades supérfluas.**


Conclusão

Agilidade é chave para alcançar altos níveis de inovação em projetos. Usar Scrum parece simples, não? Sim, é mais simples do que se possa imaginar. O grande desafio não é iniciar a utilização das boas práticas do Scrum, mas sim deixar o time, o cliente e a sua empresa prontos para as mudanças de paradigmas que a metodologia ágil traz.

Usar Scrum nos projetos ajuda a construir somente o que o cliente valoriza e não mais que isto criando produtos melhor adaptados à realidade do cliente.

As práticas do Scrum trazem vantagens ao gerenciamento dos projetos como:

- participação da equipe mais efetiva quanto à definição das atividades, gerando maior comprometimento, motivação e confiança,
- as pessoas sabem o que estão fazendo e o porquê, diferente quando o processo é imposto e não foi construído com a equipe.
- o estímulo a colaboração entre os membros da equipe deixa o time mais coeso.

Artigo Aprovado	Versão: <1.8>	
-----------------	---------------	---

- o fortalecimento do trabalho de time: cada um sabe o que o outro faz, escolhe o que vai fazer, as responsabilidades estão visíveis existe transparência e alinhamento para atender o objetivo do projeto.
- estímulo aos relacionamentos e integração entre os membros da equipe, minimizando conflitos.
- incentivo ao compartilhamento e disseminação do conhecimento.
- maior visibilidade do desempenho da equipe e de cada membro.
- maior participação e satisfação do cliente.

Conheça o Scrum, procure algum projeto, aplique as suas práticas e obtenha melhores resultados em seus projetos.

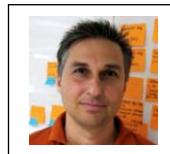
Referências

1. Agile Manifesto, **Manifesto for Agile Software Development**, 2001. Disponível em <http://agilemanifesto.org/> [Novembro, 2005].
2. ANDERSON, D. J., **Agile Management for Software Engineering, Applying the Theory of Constraints for Business Results**, Prentice Hall, 2003.
3. BOEHM, B., **A View of 20th and 21st Century Software Engineering**, ICSE 2006.
4. BOEHM, B. and Turner, R., **Balancing Agility and Discipline A Guide for the Perplexed**, AddisonWesley, 2003.
5. COHN, Mike, **Agile Estimating and Planning**, Prentice Hall, 2006, 330 p.
6. HIGHSMITH, J., **Agile Project Management, Creating innovative products**, AddisonWesley, 2004.
7. KNIBERG, Henrik., **Scrum and XP from the Trenches, How we do Scrum**, Nov., 2006, 90 p.
8. MOUNTAIN Goat Software, **The Scrum Development Process**, Disponível em <http://www.mountaingoatsoftware.com/Scrum> [Junho, 2006].
9. SCHWABER, K., and Beedle, M., **Agile Software Development With Scrum**, Prentice Hall, 2002.
10. SCHWABER K., **Agile Project Management With Scrum**, Microsoft, 2004.

11. **VERSIONONE**, Pesquisa do Estado do Desenvolvimento Ágil, **The State of Agile Development**, Disponível em <http://www.versionone.net/pdf/StateofAgileDevelopmentSurvey.pdf> [Fevereiro, 2007]

Agradecimento: Agradecemos a Anderson Correia, designer do C.E.S.A.R, pela elaboração das figuras.

Sobre os Autores:




Paulo Pereira (Caju), CSM
paulo.pereira@cesar.org.br
 Gerente de projetos do C.E.S.A.R. e Certified ScrumMaster pela *Scrum Alliance*. Possui mais de 18 anos de experiência em desenvolvimento de projetos na área de Tecnologia da Informação atuando nos últimos 5 anos como gerente de projetos.



Paula Torreão (Coelho), MSc, PMP®
paula.coelho@.org.br
 Gerente de projetos do C.E.S.A.R. e membro da diretora do PMI Chapter Recife, PE. Trabalha há 18 anos na área de Tecnologia da Informação e há 7 anos como coordenadora e gerente de projetos. Possui mestrado em inteligência artificial aplicada a educação de gerenciamento de projetos.



Ana Sofia Marçal, PMP®
ana.sofia@cesar.org.br
 Gerente de projetos do C.E.S.A.R., coordenadora do SEPG (Software Engineering Process Group) e mestranda na área de qualidade de software na UNIFOR. Tem sólida experiência na área de desenvolvimento de sistemas de informação atuando no mercado de TI há mais de 17 anos e como gerente de projetos há 7 anos.

Artigo Aprovado	Versão: <1.8>	
-----------------	---------------	---

Histórico de Revisão

Data	Versão	Descrição	Autor	Realizado
10/01/2007	1.0	Primeira versão Draft	Paulo Pereira (Caju)	S
27/01/2007	1.1	Revisão	Paulo Pereira (Caju)	S
02/02/2007	1.2	Revisão e ajustes	Paula Torreão (Coelho)	S
02/03/2007	1.3	Revisão e ajustes	Paula Torreão (Coelho)	S
01/03/2007	1.4	Revisão e ajustes	Paulo Pereira (Caju)	S
03/03/2007	1.5	Revisão e ajustes	Paula Torreão (Coelho)	S
05/03/2007	1.6	Revisão do texto.	Ana Sofia	S
06/03/2007	1.7	Revisão do texto e ajustes das figuras	Paula Torreão (Coelho)	S
06/03/2007	1.8	Revisão das referências bibliográficas.	Ana Sofia	S