2 Métodos Ágeis

2.1. Manifesto Ágil

Em fevereiro de 2001, dezessete representantes de diversas práticas e metodologias de desenvolvimento se reuniram em uma estação de esqui, em Utah nos EUA para discutir métodos mais leves de desenvolvimento do que o tradicional desenvolvimento orientado a documentos.

Autodenominados de "*The Agile Alliance*" criaram o *Manifesto for Agile Software Development* ou simplesmente Manifesto Ágil para definir a abordagem hoje conhecida como 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 às mudanças mais que seguir um plano.

Ou seja, mesmo havendo valor nos itens à direita, valorizamos mais os itens à esquerda." [Highsmith, 2001]

2.2. Princípios

Além do manifesto ágil, foram documentados doze princípios com a intenção de dar suporte a esse manifesto.

- Nossa maior prioridade é satisfazer o cliente através da entrega contínua e adiantada de software com valor agregado.

- Mudanças nos requisitos são bem-vindas, mesmo tardiamente no desenvolvimento. Processos ágeis tiram vantagem das mudanças visando vantagem competitiva para o cliente.

- Entregar frequentemente software funcionando, em poucas semanas a poucos meses com preferência à menor escala de tempo.

- Pessoas de negócio e desenvolvedores devem trabalhar diariamente em conjunto por todo o projeto.

- Construa projetos em torno de indivíduos motivados. Dê a eles o ambiente e o suporte necessário e confie neles para fazer o trabalho.

- O método mais eficiente e eficaz de transmitir informações para e entre uma equipe de desenvolvimento é através de conversa face a face.

- Software funcionando é a medida primária de progresso.

- Os processos ágeis promovem desenvolvimento sustentável. Os patrocinadores, desenvolvedores e usuários devem ser capazes de manter um ritmo constante indefinidamente.

- Contínua atenção à excelência técnica e bom design aumenta a agilidade.

- Simplicidade -- a arte de maximizar a quantidade de trabalho não realizado -- é essencial.

- As melhores arquiteturas, requisitos e designs emergem de equipes auto-organizáveis.

- Em intervalos regulares, a equipe reflete sobre como se tornar mais eficaz e então refina e ajusta seu comportamento de acordo.

[Highsmith. 2001]

2.3. Processos definidos versus processos empíricos

"É típico adotar a abordagem de modelagem definida (teórica) quando os mecanismos subjacentes pelos quais um processo opera são razoavelmente bem entendidos. Quando o processo é muito complexo para ser definido, a abordagem empírica é a escolha apropriada." [Ogunnaike / Ray, 1992]

O modelo de controle de processos definidos requer que cada parte do trabalho seja completamente entendida. Dado um conjunto de inputs bem definidos, os mesmos outputs são gerados todas às vezes. Um processo definido pode ser iniciado e executado até sua conclusão, com os mesmos resultados a cada execução. As principais características de processos definidos que podemos destacar são:

- Processo definido com mecanismos subjacentes claramente entendidos
- Sucessão de atividades claramente definidas e lineares
- Capacidade de estimar tempos de execução de cada atividade

O modelo de controle de processos empíricos provê e exercita o controle através de inspeção frequente, com visibilidade, e adaptação para processos que não são perfeitamente definidos e geram saídas imprevisíveis e não-repetíveis. Por muitos anos, metodologias de desenvolvimento de software foram baseadas no modelo de controle de processo definido. Mas desenvolvimento de software não é um processo que gera a mesma saída dada uma certa entrada. O método ágil de desenvolvimento de software Scrum é baseado no modelo de controle de processos empírico [Schwaber / Beedle, 2001].

2.4. Scrum

2.4.1. Histórico

O Scrum foi criado oficialmente na década de 90, por Ken Schwaber e Jeff Shuterland, com a ajuda de Mike Beedle, John Scumniotales e Jeff McKenna. A sua criação foi na realidade feita de maneira completamente independente, em duas frentes: de um lado, Ken Schwaber, e do outro Jeff Shuterland. Em 1995-1996, Ken e Jeff se reuniram e documentaram o processo do Scrum como se conhece hoje em dia [Sutherland, 2004]. Em 2001, Ken Schwaber e Jeff Shuterland fizeram parte do grupo de profissionais que assinou o *Agile Manifesto*.

Desde então, Scrum vêm sendo usado nos mais diferentes projetos, pelas mais diferentes organizações. Grandes empresas multinacionais, pequenas empresas startup's, desenvolvedores para agências do governo, e até a indústria de games e animação vêm se beneficiando do Scrum [Sutherland, 2004].

2.4.2. O que é?

Scrum é um processo iterativo incremental para desenvolver qualquer produto ou gerenciar qualquer trabalho. No fim de cada *Sprint* (iteração), é produzido um incremento de funcionalidades potencialmente entregáveis. As principais características do Scrum são [Schwaber, 2004]:

- É um processo ágil (*agile*) para gerenciar e controlar trabalho.
- É um "embrulho" para as práticas existentes de engenharia.
- É uma aproximação coletiva (equipes) para desenvolver produtos e sistemas iterativamente e incrementalmente, onde requisitos mudam rapidamente.
- É um processo que controla o caos de interesses e necessidades conflitantes.
- É uma maneira de melhorar a comunicação e maximizar cooperação.
- É uma forma de detectar e remover barreiras que entrem no meio do desenvolvimento e entregas de produtos.
- É uma forma de maximizar produtividade.

• Scrum é escalável de um único projeto a toda uma organização, com vários projetos inter-relacionados.

• Scrum é uma maneira de fazer com que todos se sintam bem em relação ao seu trabalho, suas contribuições, e que eles fizeram o melhor possível, o melhor que eles poderiam fazer.

• Scrum requer trabalho duro.

• Scrum requer comprometimento

2.4.3. Fluxo

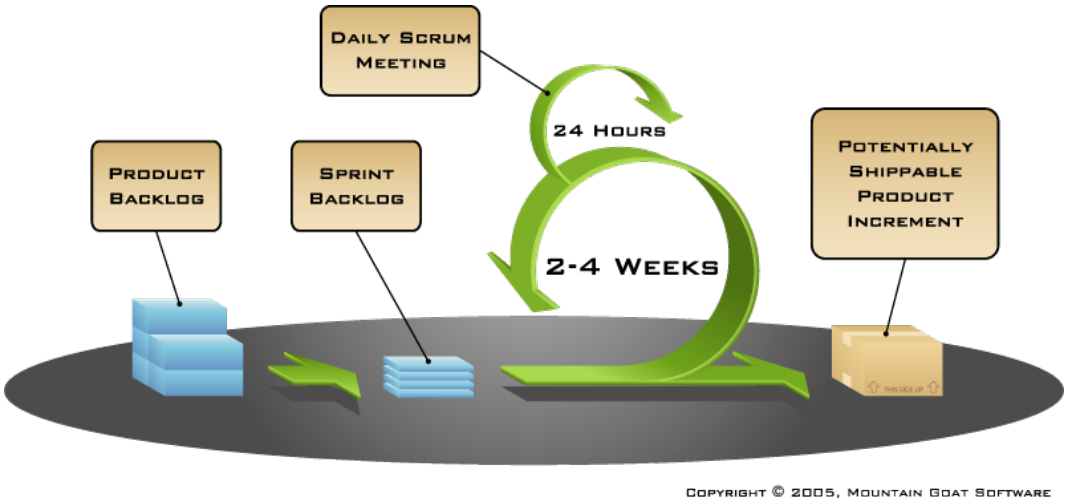

Figura 1 – Fluxo do Scrum

[Mountain Goat, 2005]

O Scrum define papéis, artefatos e cerimônias, que provêm o *Scrum Framework.* O Scrum trabalha em um processo iterativo e incremental, onde as iterações são chamadas de *Sprint*. Os *Sprints* têm duração pré-definida e fixa. Normalmente um *Sprint* dura entre 2 a 4 semanas. Durante o *Sprint*, o produto é planejado, codificado e testado. No final de cada *Sprint* há uma entrega.

2.4.4. Duração dos Sprints

É importante que todos os *Sprints* possuam a mesma duração, pois a utilização de um período constante leva a um melhor ritmo da equipe. Procure

manter a duração o mais constante possível, qualquer que seja o tamanho do *Sprint* que você escolher. Essa constância dá estabilidade ao time.

Ao planejar a duração do *Sprint* do seu projeto, procure responder à seguinte pergunta:

Por quanto tempo é possível manter uma mudança "fora" do *Sprint*?

Dependendo da estabilidade dos seus requisitos e características do seu negócio, você encontrará a duração ideal. Manter a estabilidade dentro de um *Sprint* é a regra.

2.4.5. Papéis e responsabilidades

2.4.5.1. Product Owner

O *Product Owner*, ou simplesmente PO, representa o cliente, patrocinadores e/ou financiadores do projeto. Muitas vezes, pode ser um representante do cliente dentro da empresa, ou um analista de negócio. É quem vai definir, gerenciar e priorizar os requisitos, gerenciar o ROI (*Return over investment*). O *Product Owner* trabalha como parte do time que realiza a entrega.

É responsabilidade do PO:

- Definir e manter uma visão compartilhada do projeto
- Gerenciar o ROI
- Apresentar requisitos iniciais e incrementais ao time
- Priorizar cada requisito em relação ao valor de negócio (*Business Value*)
- Gerenciar e priorizar o *Product Backlog*
- Gerenciar novos requisitos e sua priorização
- Fazer o planejamento de entregas (*Release Planning*)
- Agir como mediador quando houver mais de um cliente
- Garantir que especialistas no domínio do negócio estarão disponíveis para o time quando necessário
- Aceitar ou rejeitar o resultado dos trabalhos

Uma das responsabilidades do *Product Owner*, "Definir e manter uma visão compartilhada do projeto", merece ser mais bem explicada. A visão do produto é,

uma visão geral, ou "*big picture*", do produto, que todos têm conhecimento e entendimento desta visão. Esta visão vai guiar as atividades de desenvolvimento de produto, e provê direção ao time em relação ao que estamos tentando alcançar. Esta visão permite nos abrir para o fato que podemos atingir o possível.

Exemplo:

"Implementar um repositório eletrônico central de documentos para solicitações de hipoteca, provendo ao banco (cliente) a habilidade para lidar com as quantidades crescentes de solicitações de hipoteca sem contratar novos funcionários, através de ganho de produtividade. A solução vai nos permitir reduzir o tempo de espera dos clientes por respostas, mantendo nossa vantagem competitiva como um fornecedor/provedor orientado à clientes de serviços financeiros, enquanto ao mesmo tempo reduziremos atritos entre os funcionários, provendo economia significante em treinamento e reposição de funcionários existentes."

2.4.5.2. Scrum Master

O *Scrum Master*, ou simplesmente SM, representa a gestão do projeto, mas não no sentido de gestão tradicional. Na realidade, o SM funciona mais como um mediador e líder *coach* (em sentindo de *coaching*, como um técnico de time de futebol). O *Scrum Master* trabalha com e para o time.

É responsabilidade do SM:

- Permitir que o time se auto-organize para realizar o trabalho
- Garantir que as vias de comunicação estejam livres e acessíveis
- Garantir e ajudar que o time siga o processo do Scrum
- Cuidar e proteger a equipe de interferências externas, de modo a garantir que a produtividade do time não seja a afetada
- Remover impedimentos (obstáculos) que o time encontrar
- Agir como facilitador nas cerimônias como o *Daily Meeting*
- Garantir a colaboração entre os papéis

2.4.5.3. Equipe

A equipe é composta por indivíduos que estão comprometidos em realizar o trabalho proposto. Os times são *cross-functional*. Mas o que isso quer dizer? Nos métodos tradicionais, nós temos papéis distintos, por exemplo, arquitetos, testadores, etc. Em um time Scrum, temos pessoas com habilidades de arquitetura que podem ter uma habilidade secundária para ajudar nos testes. Ou seja, colaboração é a palavra de ordem.

Cada membro do time é responsável por:

- Definir a meta do *Sprint* (*Sprint Goal*)
- Comprometer-se com o trabalho, e fazê-lo com o máximo de qualidade
- Trabalhar tendo em mente a visão do produto e o objetivo do *Sprint*
- Colaborar com demais membros do time e ajudar o time a se auto-organizar
- Comparecer às reuniões do *Daily Meeting*
- Levantar impedimentos
- Estimar requisitos
- Manter seu próprio progresso (com ajuda do *Scrum Master*)

As equipes de Scrum têm o tamanho ideal de 7 +/– 2 pessoas. 7 é o número ótimo. Times pequenos são mais produtivos que times grandes, por causa da quantidade de vias de comunicação existente entre os membros do grupo. Se o seu time for maior que o recomendado, então ele deve se dividir em dois e o trabalho de ambos deve ser sincronizado. Outro ponto importante a lembrar é manter a estabilidade do time durante o *Sprint*, ou seja, evitar mudanças na equipe no meio de um *Sprint*.

2.4.6. Cerimônias

As cerimônias do Scrum são reuniões, formais ou não, que acontecem em momentos específicos do *Sprint*. A seguir, falaremos sobre as seguintes cerimônias do Scrum:

- *Sprint Planning*
- *Daily Meeting*
- *Sprint Review*
- *Sprint Retrospective*

2.4.6.1. Sprint Planning

A reunião de *Sprint Planning* é uma reunião de curta duração (geralmente 4 horas), e tem como objetivo planejar o trabalho da equipe durante o *Sprint*. O dia em que ocorre a reunião de *Sprint Planning* é considerado o primeiro dia do *Sprint*. Esta reunião é dividida normalmente em 2 partes: *Sprint Planning* I e II, de igual duração.

Nesta reunião, participam somente as pessoas comprometidas com a entrega. Pessoas interessadas ou especialistas no negócio podem ser convocadas à reunião pontualmente, entretanto devem se retirar após responder os questionamentos ou prestar os esclarecimentos devidos.

2.4.6.1.1. Sprint Planning I

- Participantes:
    - Equipe (*Team*)
    - *Scrum Master*
    - *Product Owner*
- Entradas:
    - *Product Backlog* (Estimado e Priorizado)
    - Dados históricos de capacidade da equipe (se houver)
    - Condições do negócio
- Saídas
    - *Selected Product Backlog*
    - Objetivo do próximo *Sprint*
    - Equipe comprometida e com entendimento dos requisitos

O objetivo do *Sprint* é decidido pelo *Product Owner* em conjunto com o time. Todos discutem os itens de *Backlog* priorizados e estimados e definem o objetivo, que deve ser uma frase simples e objetiva. Por exemplo:

"Permitir que pessoas se cadastrem em nossa newsletter corporativa e se descadastrem caso eles não desejem mais recebê-la."

Esta reunião também tem como saída o comprometimento da equipe em realizar o trabalho proposto durante o *Sprint*. A equipe, e somente a equipe, pode decidir e se comprometer a respeito do trabalho que será executado, seja a partir de dados históricos de capacidade da equipe, ou a partir da capacidade que a equipe espera atingir no *Sprint*. Em outras palavras, o *Product Owner* prioriza e seleciona, mas só a equipe pode dizer o que será atingido durante o *Sprint*. Se o *Product Owner* ficar insatisfeito, tudo o que ele pode fazer é repriorizar as tarefas e ajustar o objetivo do *Sprint*.

2.4.6.1.2. Sprint Planning II

Na reunião de *Planning* II, participam somente as pessoas que efetivamente irão realizar o trabalho. Aqui já não cabe mais a participação do *Product Owner*, somente da equipe e do *Scrum Master*. O *Product Owner* pode ser consultado durante o *Sprint Planning* II caso ocorram dúvidas a respeito dos requisitos selecionados.

- Participantes
    - Equipe (*Team*)
    - *Scrum Master*
- Entradas
    - *Selected Product Backlog*
    - Objetivo do próximo *Sprint*
- Saídas
    - *Sprint Backlog*

O *Sprint Backlog* é uma expansão do *Product Backlog* selecionado durante a reunião de *Planning* I. A equipe vai ler, de um por um, os itens selecionados, e decompor em tarefas necessárias para considerar o item de *Backlog* implementado, pronto. Este trabalho é feito colaborativamente, com a participação de toda a equipe. É recomendável manter as tarefas num nível de granularidade suficiente para admitir tarefas de 4 ou 8 horas (ou seja, no máximo 1 dia de

trabalho). Estas tarefas podem ser estimadas individualmente, ou pode-se assumir o tamanho do requisito previamente estimado (isso fica à cargo da equipe).

É importante firmar o conceito de "pronto" (*done*), para garantir o entendimento de todos. Pronto significa que o requisito foi implementado e testado, e está num estado entregável, caso seja necessário. Ter isto firmado em mente é importante porque na eventualidade de um item de *Backlog* ter ficado parcialmente implementado, digamos, 80% num *Sprint*, isso significa que ele não está pronto e deverá ser selecionado como item de *Backlog* para o próximo *Sprint*.

No término desta cerimônia, temos a equipe comprometida, uma meta de velocidade pré-estabelecida e o trabalho decomposto em tarefas de granularidade suficiente para serem trabalhadas.

2.4.6.2. Executando o Sprint

Após a reunião de *Planning* I e II, a equipe está pronta para iniciar o trabalho, auto-organizando-se para decidir quem executa o quê e quando. É importante que os membros da equipe se comuniquem para efetivar esta auto-organização.

Durante esta fase, que dura entre 2-4 semanas, é que o trabalho será executado e impedimentos serão levantados. Aqui é que o trabalho do *Scrum Master* será mais evidenciado, agindo como um desobstruidor, protetor e mentor. De maneira a manter todos informados, é feita uma cerimônia de acompanhamento diário, o *Daily Meeting*, sobre a qual vamos conhecer mais adiante. É também durante a fase de execução que o *Scrum Master* deve agir como um escudo para a equipe, evitando que trabalho não planejado ou novos itens de *Backlog* sejam incluídos no *Sprint*. Vamos para um exemplo:

Estamos trabalhando em um *Sprint* de 4 semanas. Uma nova solicitação de negócio chegou para o *Product Owner*, com um grau de importância alto. O *Product Owner* então, apressa-se em comunicar ao time que eles devem começar a trabalhar nesta nova funcionalidade/solicitação agora.

Isso é disruptivo para o time, e introduz instabilidade. O time já gastou tempo planejando o *Sprint*, e está mentalmente focado em atingir o objetivo do *Sprint*, e mais importante, entregar software funcionando. O *Product Owner* deve tentar conversar com o cliente, e explicar como isso irá interferir no trabalho do

time, e priorizar esta demanda no *Product Backlog* para ser trabalhado no próximo *Sprint*.

Se um bug crítico foi encontrado, e a não correção deste bug trará maiores consequências à organização ou a seus clientes (por exemplo, "pagamentos de cartão de crédito estão sendo duplicados"), então é compreensível que um item do *Sprint Backlog* seja retornado ao *Product Backlog* e substituído pela nova demanda.

Se por alguma razão, a situação for muito caótica, então é melhor dissolver o *Sprint* e começar novamente com uma nova reunião de *Sprint Planning*. Garanta que uma reunião de retrospectiva desse *Sprint* prematuramente terminada irá acontecer, para que possam ser encontradas as causas reais desse acontecimento. A prática de interromper um *Sprint* deve ser adotada como último recurso. Ao ser aplicada alguns fatores que possivelmente causariam a aplicação desse recurso devem ser observados, como por exemplo:

- Planejamento pobre
- Priorização não está correta
- Falta de entendimento e de suporte das práticas ágeis
- Ambiente de negócio passou por mudança massiva
- Políticas corporativas
- Não há um acompanhamento cíclico do cliente quanto à evolução do *Backlog*

2.4.6.3. Daily Meeting

A reunião de *Daily Meeting* (ou *Daily Scrum*) é um encontro público da equipe, de curtíssima duração (no máximo 15 minutos). Nesta reunião são admitidos membros do time e interessados no projeto, entretanto tudo que os interessados podem fazer é observar, sem fazer comentários nem perguntas.

Seguem algumas regras:

- Reunião com frequência diária
- Duração: no máximo 15 minutos
- Mesmo local e horário, todos os dias
- Participantes:

- *Scrum Master, Team, Product Owner*
- Interessados (somente como ouvintes, não podem interromper sob hipótese alguma)

• Entradas são geradas a partir de três perguntas:

- O que eu tenho atingido desde a nossa última reunião?
- O que me proponho a atingir até a próxima reunião?
- Quais problemas estão me impedindo ou atrapalhando na realização do meu trabalho?

• Como saída, temos impedimentos e decisões tomadas.

Os principais benefícios que podemos enumerar desta reunião são:

• Visibilidade para todo o grupo: todos sabem o que está acontecendo

• Ajudar o *Scrum Master* a identificar impedimentos, para que os mesmos possam ser resolvidos e proteger a produtividade do time.

É obrigação do *Scrum Master* trabalhar em cima dos itens de impedimento levantados durante a reunião, e escalá-los à alta direção caso ele não seja capaz de resolver. Impedimentos detém que o time realize seu trabalho em um ambiente ótimo, e podem comprometer o objetivo do *Sprint*.

É função do mediador da reunião (normalmente o *Scrum Master*) manter discussões técnicas ou a respeito de solução de problemas de fora, para que sejam por exemplo feitas imediatamente após a reunião. Muitas vezes pessoas podem se alongar demais na discussão de um problema técnico, e isso pode atrapalhar o bom andamento e o propósito da reunião. O mediador da reunião deve declarar claramente para os envolvidos o término do *Daily Meeting*, deixando assim o time à vontade para discutir problemas ou questões técnicas, ou para agendar uma outra reunião para discussão do tema entre os envolvidos.

2.4.6.4. Sprint Review

Esta cerimônia é realizada no último dia do *Sprint*. É uma reunião informal (em geral 4 horas) onde todos são convidados a participar, principalmente os clientes do projeto. Normalmente é realizada numa sala de reunião ou auditório, de portas abertas, e o objetivo desta reunião é dar a oportunidade à equipe para apresentar o resultado do trabalho realizado durante o *Sprint*. Tipicamente acontece como uma demonstração do incremento de funcionalidade, ou da sua arquitetura informal. A equipe tem no máximo 1 hora para se preparar para esta reunião, na qual deve ser apresentada a demo do sistema.

Funcionalidade que não ficou pronta (aqui, mais uma vez, destaco para a importância de alinhar o significado de "pronto" entre todos os envolvidos, pois este conceito pode variar de organização para organização) não pode ser apresentada. Artefatos que não são funcionalidade também não podem ser apresentados, com exceção de quando eles são usados como suporte para entendimento da funcionalidade demonstrada. Artefatos não podem ser demonstrados como produtos de trabalho, e seu uso deve ser minimizado para evitar confundir os clientes, ou requerer que eles entendam como funciona desenvolvimento de sistemas.

O ambiente de demonstração deve ser o mais próximo do ambiente de produção. Na maioria do tempo da reunião, membros do time irão apresentar funcionalidades e responder perguntas dos clientes a respeito dos produtos apresentados. No final desta reunião, os clientes são questionados, um por um, e devem declarar suas impressões, mudanças desejadas e a prioridade dessas mudanças. O *Product Owner* então discute com os clientes e o time sobre a potencial reorganização do *Product Backlog* baseado no feedback recebido.

Adicionalmente, os clientes podem apontar e identificar funcionalidades que não foram entregues, ou não foram entregues conforme esperado, e solicitar que esta funcionalidade em questão seja retornada ao *Product Backlog* para repriorização. Além disso, os clientes também podem solicitar que novas funcionalidades sejam adicionadas ao *Product Backlog*. O objetivo desta reunião é claro: apresentar o incremento de funcionalidade atingido durante o *Sprint*, e receber feedback dos clientes, de forma a repriorizar o *Product Backlog* e adicionar novos itens ao mesmo.

2.4.6.5. Sprint Retrospective

Esta é uma reunião formal, fechada (geralmente 3 horas). Participam dela somente Time, *Scrum Master* e *Product Owner*, sendo a presença deste último opcional.

Todos os membros do time devem responder à três perguntas:

- O que foi bem?
- O que pode melhorar?
    - Quem tem controle do que pode melhorar? (Time/organização)

• Neste ponto, é preciso tomar cuidado. A função da retrospectiva não é apontar culpado. Deve ser definida somente a responsabilidade da implementação da melhoria: do time, ou da organização.

Após isso, as oportunidades de melhoria detectadas devem ser priorizadas. A partir dessas perguntas, e da priorização, serão inferidas as seguintes saídas:

• *Team Backlog*, o que o Time pode mudar no processo de forma a melhorá-lo.

• Impedimentos abertos que não estão no controle do time, devem ser trabalhadas pelo *Scrum Master*.

Ambos ordenados por prioridade. O *Scrum Master* não está nesta reunião para prover respostas, mas para facilitar a busca do time por melhores maneiras de fazer com que o processo do Scrum trabalhe para o Time. Itens viáveis podem ser adicionados à próximo *Sprint*, como *Product Backlog* não-funcional de alta prioridade.

2.4.7. Artefatos do Scrum

No Scrum, existem somente 3 artefatos requeridos:

• *Product Backlog*

• *Sprint Backog*

• *Burnup/Burndown Charts*

2.4.7.1. Product Backlog

O *Product Backlog* é uma lista de todas as funcionalidades desejadas no produto, estimadas pelo time e priorizadas pelo *Product Owner*. Quando um projeto é iniciado, não é possível escrever todos e quaisquer requisitos que porventura um dia serão incorporados ao sistema. Tipicamente, escrevem-se primeiro os requisitos que são mais óbvios, o que é normalmente mais do que suficiente para o primeiro *Sprint*.

O *Product Backlog* então deve crescer e mudar à medida em que se aprende mais sobre o produto, seus clientes e o negócio, ou seja, é emergente. Os itens de maior prioridade são geralmente mais detalhados, e o mesmo é mantido de uma forma visível para todo o time e o *Scrum Master*. Geralmente, qualquer pessoa pode contribuir com o *Product Backlog*, mas as solicitações devem ser sempre priorizadas pelo *Product Owner*.

Exemplo de um *Product Backlog*, já estimado, priorizado e com valor de negócio definido:

#	Prioridade	Descrição	Estimativa	Valor de negócio
1	Muito Alta	Como gestor, quero controle de acesso por login e senha para acessar o sistema, para controlar o acesso	13	100
2	Muito Alta	Como usuário, quero poder cadastrar fornecedores no sistema, para poder encontrar facilmente seus contatos	5	80
	Alta	Como usuário quero poder cadastrar notas fiscais no sistema, associando-as ao fornecedor, para poder manter registro das notas	8	90
	Baixa	Como operador, desejo tirar backup dos dados do sistema a qualquer momento, para poder garantir a segurança dos dados	5	30
	Média	Como vendedor, desejo poder registrar as vendas que faço sobre meu login, para poder administrar minha comissão.	3	60

Tabela 1 – Exemplo de Product Backlog.

2.4.7.2. Sprint Backlog

O *Sprint Backlog* é a lista de histórias que o time se comprometeu com o *Product Owner* a implementar durante o *Sprint*, após a reunião de *Sprint Planning* I e II. Dependendo do tamanho de uma história em particular, ela pode ser quebrada em duas ou mais histórias menores para facilitar a entrega em etapas.

O *Sprint Backlog* é escolhido pelo time dado o objetivo e prioridades descritos pelo *Product Owner*. Cabe somente ao time decidir quais histórias implementará durante o *Sprint*. Como é o time que se compromete com o objetivo do *Sprint* é ele que decide quais histórias deverão ser feitas para cumprir esse objetivo. O *Sprint Backlog* pode ser mantido da maneira como for mais conveniente para a organização: pode ser numa planilha de Excel, ou num sistema de controle de tarefas, ou até somente no quadro do time.

2.4.7.3. Burndown/Burnup Charts

Os gráficos de *Burndown* ou *Burnup* são as melhores ferramentas do time para manter registro da velocidade atual do trabalho. Esses gráficos geralmente ficam no quadro do time, ou perto dele, e podem ser mantidos manualmente ou via Excel.

Existem várias formas de se representar um gráfico de *burndown* ou *burnup*, a única diferença em ambos é que, no gráfico de *burndown*, a linha representa o quanto de trabalho ainda falta para concluir o *Sprint*, e no gráfico de *burnup* a linha representa quanto de trabalho já foi feito no *Sprint*, considerando que o montante de trabalho é contabilizado pelo fator de complexidade adicionado a cada história durante as estimativas (mais detalhadas a diante). Os gráficos são ferramentas fundamentais para o time se auto-gerenciar, pois permitem de uma certa forma prever o sucesso de um *Sprint*. O gráfico do *Sprint* deve ser mantido pela própria equipe, e atualizado diariamente.

Exemplo de gráfico de *Sprint Burnup*:

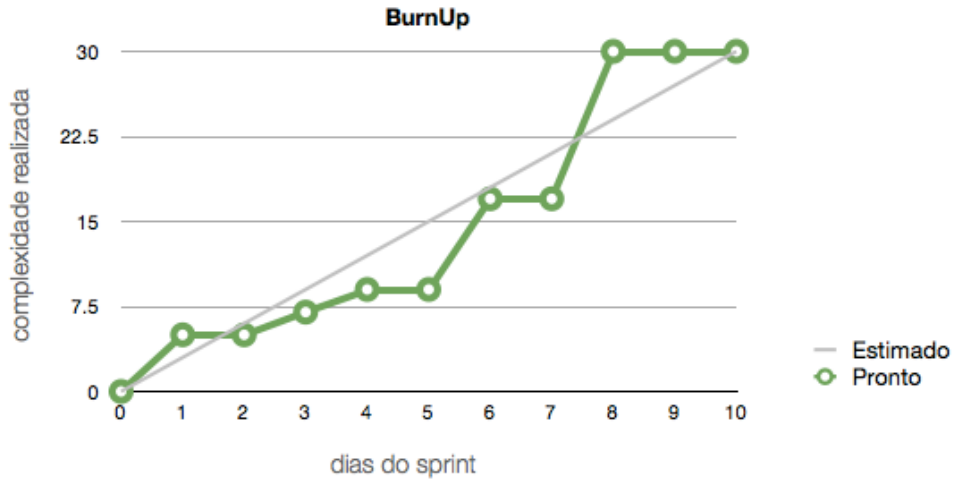

Figura 2 – Exemplo de Sprint BurnUp

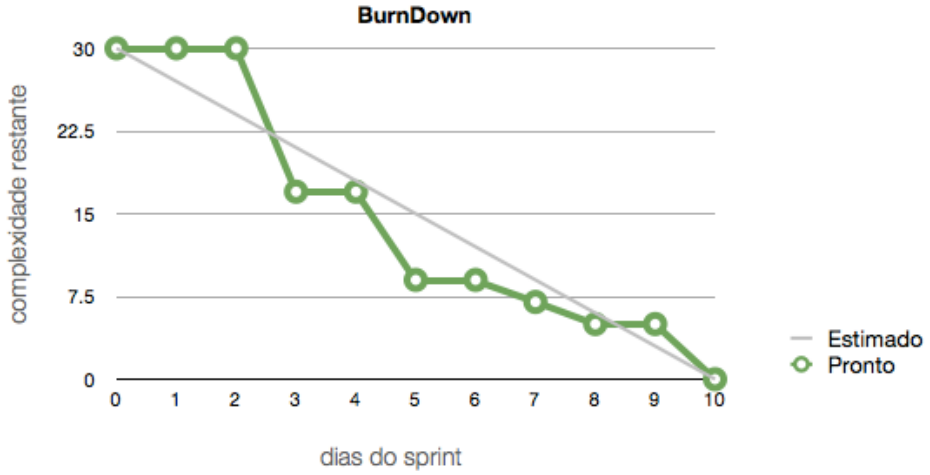

Figura 3 – Exemplo de Sprint BurnDown

Além dos gráficos do *Sprint*, a equipe também pode decidir monitorar sua velocidade entre *Sprints*, utilizando-se de um gráfico de *Product Burndown* ou *Burnup*. Neste caso, a diferença entre *Burndown* e *Burnup* é a mesma, a única diferença é que neste caso, será monitorado o tamanho do trabalho que a equipe consegue entregar a cada *Sprint*. Este gráfico também dá visibilidade ao *Product Owner* e os *Stakeholders* que estão patrocinando o projeto sobre a velocidade do time por *Sprint*.

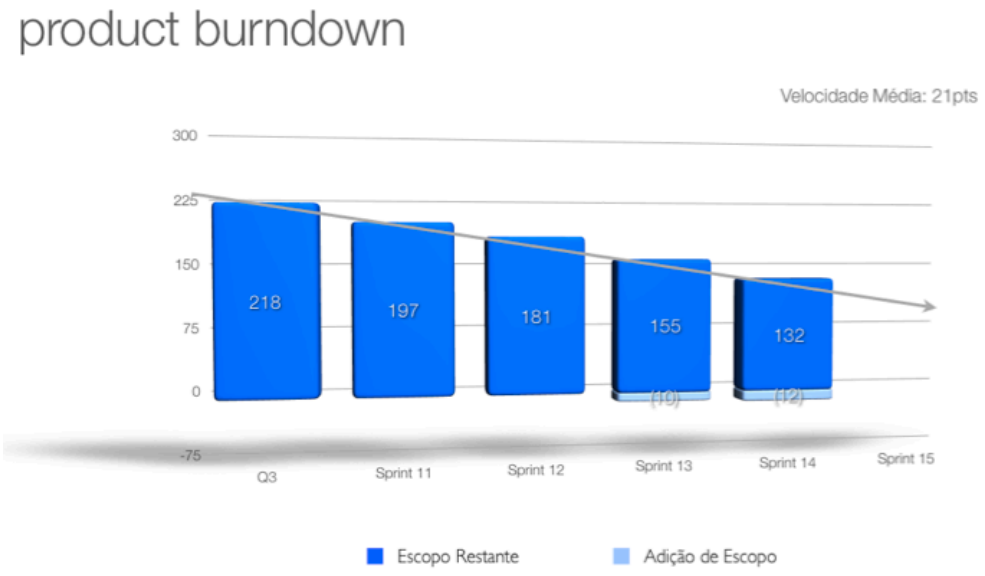

Figura 4 – Exemplo de Product Burndown

2.4.8. Estimativas – Planning Poker

Antes de começar a falar sobre estimativas, gostaria de chamar a atenção para o significado da palavra "Estimativa".

O que é uma estimativa?

1. Um cálculo aproximado
2. Uma declaração por escrito do preço aproximado que será cobrado para um trabalho específico.
3. Um julgamento ou avaliação

Fonte: Oxford Dictionary

Mais importante, estimativas são inerentemente erradas. Estimativas não são exatas, e não refletem a realidade. A única maneira de saber o tempo exato que uma atividade vai durar, ou quanto esforço será necessário despender para executar algum trabalho, é após a sua realização. Dito isso, vamos prosseguir a respeito do *Planning Poker*.

Uma das maneiras mais eficientes e recomendadas de estimar o tamanho de requisitos em times que adotam métodos ágeis (XP/Scrum) é o jogo de cartas

conhecido como *Planning Poker* [Cohn, 2005]. Este método de estimar combina opinião de especialistas, analogias e desagregação em uma aproximação eficiente para estimar de maneira rápida, porém confiável. É uma variação do método de estimativa *Wideband Delphi* (1940).

As estimativas acontecem normalmente em uma reunião (geralmente 4 ou 8 horas, dependendo da quantidade de requisitos a ser estimada). Os participantes do jogo de poker são todos os membros do time do Scrum. O *Product Owner* normalmente comparece a esta reunião para prestar esclarecimentos a respeito dos requisitos, porém não pode opinar e estimar junto com o time. O *Scrum Master* registra os resultados da reunião, e assim como o *Product Owner*, não interefe nas estimativas do time.

No início de uma reunião de *Planning Poker*, cada membro do time recebe um deck de cartas, que contém uma variação da sequencia de Fibonacci, e uma última carta que representa a não possibilidade de estimativa: 0, ½, 1, 2, 3, 5, 8, 13, 20, 40, 50, 75, 90, 100, ?. Ainda no início, todos os itens a serem estimados são lidos pelo *Product Owner* ou *Scrum Master*, e a equipe conversa para decidir qual é o menor item de *Backlog* disponível.

Após essa estimativa inicial, esse item é marcado como "2" pontos, e a equipe prossegue para estimar os demais itens de *Backlog*. Isso serve para definir uma referência de tamanho e complexidade para ser utilizada nas demais estimativas. Isso só precisa ser definido na primeira reunião de Estimativa, e deve ficar registrado, para uso nas futuras reuniões. Em casos excepcionais, o time pode decidir mudar a história de referência por uma outra história. Aqui também é importante usar o bom-senso.

Seguindo adiante, para cada história a ser estimada, o *Scrum Master* ou *Product Owner* lê a descrição e os critérios de aceitação da mesma. O *Product Owner* responde a quaisquer questionamentos que o time possua a respeito da história, entretanto procurando manter o nível da discussão em alto nível, e não entrar em detalhes. É comum definir também um *timebox* para esse tempo de perguntas e respostas, de modo a não se estender demais e perder muito tempo em uma única história.

Depois da apresentação e discussão inicial, cada pessoa seleciona sua estimativa (uma carta), baseada em sua opinião pessoal a respeito do tamanho e complexidade da história, e a coloca em cima da mesa, com a face voltada para

baixo. Quando todos os participantes selecionarem uma carta, as mesmas serão viradas ao mesmo tempo.

Se houver uma grande variância entre a maior e menor carta, o time vai discutir o que cada membro estava pensando quando selecionou o tamanho. A ideia é que todos entendam as razões, que mais uma ou outra pergunta sejam respondidas pelo *Product Owner*, e a equipe volta a pensar sobre a história, e faz uma nova estimativa. O ciclo deve ser repetido até que todas as estimativas estejam próximas, ou que a equipe chegue a um consenso. É comum definir um limite de rodadas de poker, para evitar que as coisas fiquem muito arrastadas. Repetir até que todas as histórias estejam 100% estimadas!

2.4.9. Escrevendo Histórias

Uma história de usuário, ou *user story*, é um requisito de sistemas de software formulado com uma ou duas sentenças em linguagem natural. Cada *user story* é limitada e pequena, de forma a caber perfeitamente em um pequeno papel de *post-it*. Isso é feito para de forma a garantir que histórias muito grandes sejam sempre quebradas e granularizadas.

User stories são uma maneira rápida de lidar com requisitos do cliente, sem ter que elaborar documentos de requisito formais e vastos, e sem executar tarefas administrativas super-dimensionadas para mantê-lo. A intenção com a história é ser capaz de responder mais rápido e com menos overhead as mudanças nos requisitos voláteis do mundo real.

Uma história é uma declaração informal do requisito, que em segue normalmente o seguinte formato:

Como um <papel do usuário/ator> quero <funcionalidade> para <valor de negócio>

Na parte de trás da história, normalmente são escritos os critérios de aceitação. Critérios de aceitação funcionam como um balizador de entendimento do que é “pronto” entre o desenvolvedor e o cliente, para aquele requisito específico. É importante frisar que histórias não são requisitos do IEEE nem Casos de Uso (*use cases*).

Exemplo de uma história:

"Como Gestor, quero que as informações pessoais dos clientes fiquem gravadas em formato criptografado no banco de dados, para garantir a privacidade e a segurança dos dados dos meus clientes

Critérios de aceitação:

- Ter os dados armazenados no banco de dados e arquivos de troca do sistema usando algoritmo de criptografia do tipo chave publica/chave privada. "

2.4.10. Scrum of Scrums

"*Scrum of Scrums* é uma importante técnica para escalar Scrum em grande times e projetos. Essas reuniões permitem agrupar os times para discutir seus trabalhos, focando especialmente em área de sobreposição e integração". [Cohn, 2005]

Essa reunião tem por objetivo alinhar informações técnicas entre os times de um mesmo projeto, bem como discutir eventuais problemas ou decisões técnicas.

Imagine um projeto contendo sete times, cada um com sete membros, cada time irá conduzir seu próprio *Daily Meeting*. Cada time elege uma pessoa para fazer parte da reunião do *Scrum of Scrums*, essa deverá ser uma decisão do próprio time, por reconhecerem um membro capaz de dar posição técnica das decisões do time e opinar nas decisões dos demais. Não é aconselhável que o *Scrum Master* ou *Product Owner* sejam os eleitos a participarem dessa reunião, exatamente pelo fato das decisões técnicas deverem ser tomadas pelo Time. Uma vez que o representante foi escolhido, ele deverá ter uma sequência de participações, mas não está preso a essa reunião, eventualmente outro membro do time poderá acompanhá-la. Caso o time já saiba que existe uma determinada necessidade específica de conhecimento dos outros times, como arquitetura ou banco de dados, poderá previamente solicitar a presença de membros dos outros times que tenham o conhecimento específico do problema para facilitar a solução e o alinhamento entre os Times.

O *Scrum of Scrums* pode ser feito também de forma recursiva, principalmente quando existem muitos times envolvidos no projeto. Sendo

possível dividir em diversos grupos por temas, componentes ou funcionalidades. A reunião inicial ocorre somente dentro dos temas e na sequência cada representante do tema se reúne em um novo *Scrum of Scrums*.

Caso o número de participantes na reunião seja pequeno, tanto por poucos times envolvidos no projeto, ou por uma recursão do *Scrum of Scrums*, poderá ser enviado mais de um representante por time, dessa forma é possível enriquecer mais a reunião. Essa é uma medida que deverá ser tomada caso a caso, porque muitas pessoas participando dessa reunião poderá também fazer com que deixe de ser efetiva.

Outra reunião similar ao *Scrum of Scrums* também muito importante é o agrupamento frequente por especialidade, a fim de manter um nivelamento de conhecimento, melhores praticas e padrões. Dessa forma podemos manter a arquitetura, design, componentes *client-side*, banco de dados mais alinhados entre times. Isso não é somente interessante do ponto de vista de unicidade no projeto como também facilita a transição de membros entre times, por manter o mesmo contexto base.

A frequência dessa reunião deverá ser determinada pelo time, mas é importante lembrar que quanto maior a distância entre reuniões fatalmente aumentará seu tamanho, e eventualmente algumas informações podem ser perdidas.

Principalmente no início do projeto, recomenda-se que o *Scrum of Scrums* seja executado diariamente, ao fim de todos os *Daily Meetings*. [Schwaber, 2007]

É importante que os times foquem na colaboração entre si durante o *Scrum of Scrums*, e no alinhamento das expectativas de integração entre componentes correlacionados.

Se fizermos um paralelo do *Scrum of Scrums* com o *Daily Metting* [Cohn, 2005], as perguntas diárias deverão ter uma leve mudança, além do acréscimo de uma muito importante para essa reunião:

• O que o seu time fez desde a última reunião?

• O que o seu time irá fazer depois desta reunião?

• Algo está diminuindo a velocidade o interrompendo o trabalho do seu time?

• Seu time está a ponto de fazer (ou deixando de fazer) algo que de alguma forma irá impactar outros times?

A última pergunta pode ser extremamente útil no que diz respeito a coordenação de múltiplos times em um projeto, e da expectativa entre times. Eventualmente um time não tem plena visão da profundidade do impacto que poderá causar em outro time por uma decisão técnica, que em algumas situações poderia tomar outra solução.

A idéia das perguntas, como no *Daily Meeting* é apenas para dar um ritmo rápido e efetivo a reunião. Questões mais profundas deverão ser endereçadas após todos passarem por esse pequeno processo. Dessa forma pode ser otimizado o tempo daqueles que eventualmente não serão impactados por um determinado problema. É importante no segundo momento, aonde as discussões tomam profundidades, levantar ações práticas e seus responsáveis para que esses problemas possam ser resolvidos ou que os levantamentos necessários possam ser feitos até o próximo encontro.

2.5. Kanban

Em 2004, David Anderson adaptou o modelo Toyota de controle enxuto de produção utilizando como base a "teoria das restrições", que de maneira bem simplificada diz que se for possível identificar um gargalo em seu processo, dever-se-ia focar toda a atenção para aliviar esse gargalo [Anderson, 2003]. Dessa forma, Kanban se utiliza de filas de estágios (ou etapas) de processo com limites determinados e o acompanhamento da evolução de cada item em cada uma desses estágios, com o intuito de expor gargalos nesse fluxo para que imediatamente sejam trabalhados.

Existem 5 propriedades no Kanban [Anderson, 2010].

2.5.1. Visualização do Fluxo

Visualizar o fluxo de trabalho e fazê-lo visível constantemente é fundamental para o entendimento de como funciona o trabalho e seu fluxo. Torna-se mais difícil fazer mudanças sem o entendimento correto desse fluxo. Uma maneira comum de visualizar o fluxo é utilizando cartões e um quadro com colunas. As colunas no quadro representam os diferentes estágios deste fluxo e os cartões brancos as funcionalidades ou histórias. Ainda na representação abaixo é possível observar figuras que representam o desenvolvedor responsável pela história em questão. Os marcadores em amarelo representam tarefas detalhadas das histórias. Podem ser adicionadas outras representações visuais nas histórias dependendo do seu cenário, como uma estrela para identificar uma história urgente, ou uma marcação vermelha para indicar um impedimento.

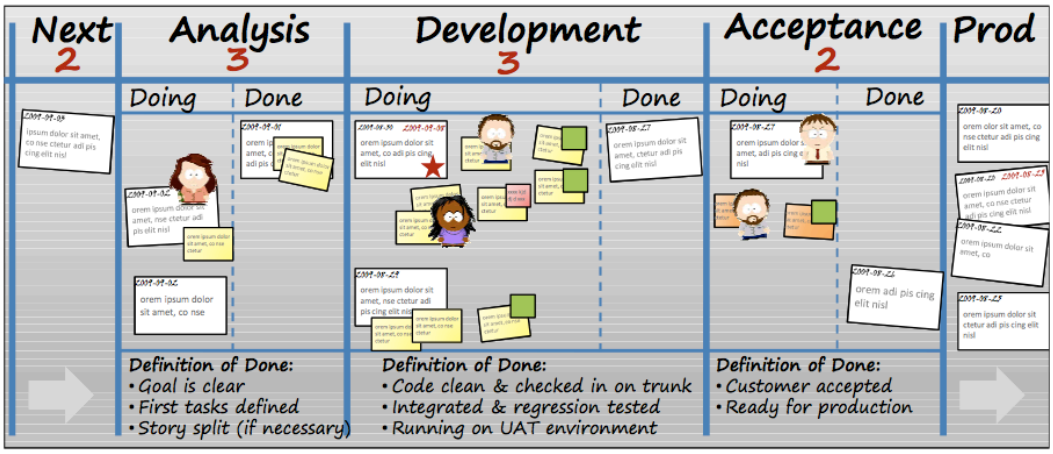

Figura 5 – Exemplo de um quadro Kanban

[Kniberg, 2009]

2.5.2. Limite de trabalho em andamento

O número em vermelho, abaixo da descrição de cada etapa no exemplo de quadro acima, mostra a quantidade de cartões máxima para cada etapa de desenvolvimento. Uma vez observado que uma etapa não pode receber novos

cartões, deverá ser observado o que está evitando o andamento correto do fluxo, para que seja trabalhado pelos membros do time.

2.5.3. Administração do Fluxo

O Fluxo de trabalho por cada estágio deve ser monitorado, medido e exposto. Ao medir o tempo de um cartão a cada estágio, poderá ser notado de forma clara o impacto (negativo ou positivo) de eventuais mudanças no processo.

Uma forma comum de monitorar esses tempos é anotar no verso dos cartões a data que a história foi criada, passou por cada estágio e finalmente foi concluída.

2.5.4. Processo e políticas explicitas

Todo o sistema de trabalho é explicito no quadro. Os estágios de desenvolvimento, a definição de pronto, quem está fazendo qual atividade e principalmente os limites de trabalho em andamento de cada estágio.

2.5.5. Melhoria colaborativa

Kanban encoraja pequenas mudanças de forma incremental e contínua. O time, tendo uma visão compartilhada de seu trabalho, riscos, fluxo e processo, deve sugerir mudanças assim que perceber uma oportunidade de melhoria. Essas mudanças devem ser feitas em consenso, aonde todo o time participa da decisão.