# Curso de ****Scrum: Agilidade em seu projeto****

# Um pouco de história

O termo "Scrum" foi cunhado por Takeuchi e Nonaka em alusão à formação do Rugby, em que todo o time está junto buscando a posse da bola. Se alguém quebra a formação, todo o time perde. Da mesma forma, quando alguém consegue a posse da bola, o time inteiro é responsável. Ou seja, os indivíduos são menos importantes do que o time em si.

Para **Scrum**, o conceito de time é extremamente importante! O product owner, o scrum master e a equipe de desenvolvimento estão todos englobados no time. Todos são igualmente responsáveis pelo sucesso ou fracasso de um projeto ou melhoria na vida do usuário.

A metáfora que utiliza o **Rugby** ainda se relaciona ao conceito de **sprint**, que é quando o time rouba a bola e o jogador deve correr rapidamente com ela. A partir disso mais uma Sprint é iniciada e novos pontos são acumulados. É um processo iterativo e incrementativo.

O Scrum de 1995 não é mesmo de hoje: com o passar dos anos e a grande adoção do processo, ele deixou de ser um método e passou a ser um framework, um conjunto de ideias.

## Time-boxes e Sprints

A **Sprint** é um dos time-boxes, e este conceito é extremamente importante para **Scrum**.

A principal diferenciação entre o Scrum e outros processos ágeis é seu forte embasamento com time-boxes: caixas de tempo com capacidade definida, rígida e não negociável. Uma vez que a Sprint ou qualquer reunião define uma certa duração de tempo, ela passa a ser fixa.

A **Sprint** é o tempo que temos para agregar valor e marcar pontos. Adicionamos valor ao software conforme o que o usuário acredita que é necessário. O tempo máximo para isso é de até um mês, período definido no "**Scrum Guide**", publicada por Ken Schwaber e Jeff Sutherland.

## Duração da Sprint

Conforme comentado, a Sprint limita-se ao máximo de um mês de duração. Esse período, entretanto, pode compreender 1 mês, 1 semana, 3 dias, e assim por diante. A decisão de demarcar um intervalo é de extrema importância, pois ela será a base de todos os demais *time-boxes* do Scrum.

Para delimitar o tamanho da Sprint vários fatores devem ser levados em consideração. Como qualquer começo de projeto, nesse momento não temos conhecimento sobre o destino final dele. Nesse momento tendemos às Sprints com durações de tempo mais curtas.

Ao fim de cada Sprint é realizada uma *Review Meeting*, que é uma reunião na qual são mostrados os produtos elaborados durante esse período. A partir dela recebemos feedbacks, e quanto mais retornos tivermos, melhor será para o cliente e para nós, pois evita a execução de trabalhos desnecessários.

Em determinados períodos do desenvolvimento de um projeto pode ser difícil devolver algum resultado devido à grande porção de trabalho envolvida. Por exemplo, na primeira vez que uma equipe criou uma nota fiscal eletrônica, demorou mais que o tempo previsto.

Há momentos em que a granularidade não pode ser quebrada, pois não adianta entregar meia nota fiscal, é preciso entregá-la inteira. Nesses casos, seremos obrigados a aumentar o tamanho da Sprint.

Outra variável que influencia na sua definição é o conhecimento de tecnologia. Se o time conhece pouco da tecnologia em questão, talvez valha a pena ter uma *Sprint* maior, considerando que será necessário que o time estude para produzir valor. O mesmo pode ocorrer com uma equipe composta de muitos indivíduos inexperientes.

Outra variável são as pessoas mais jovens, que tendem a estabelecer Sprints mais curtas. São pessoas normalmente acostumadas a uma vida mais dinâmica e objetivos mais próximos. Nesse caso, Sprints muito longas podem, inclusive, acarretar em problemas de desmotivação e distração. Outra questão é a "Síndrome do estudante", que implica em pessoas que agem somente em cima da hora, geralmente envolvendo indivíduos de determinada faixa etária.

Por outro lado, quando mais curta for a Sprint, maior é o stress dos envolvidos. Existem aqueles que trabalham bem sobre pressão, mas outros não lidam de maneira positiva com a situação.

A duração ideal da Sprint depende muito da equipe e das pessoas envolvidas. Como dito anteriormente, a duração da Sprint atual não pode ser alterada. No entanto, ao final de um ciclo, ela pode ser revista.

# A review meeting

A Sprint contém diversas etapas: no início há uma reunião para decidir o que fazer, trata-se do *planning* (planejamento). A maior parte é ocupada pelo desenvolvimento em si, que é o momento em que se coloca a mão na massa, parando todos os dias para fazer o *Daily Scrum*. E, por fim, ocorre a *Review Meeting* (reunião de revisão), revisão do que foi feito e a *Retrospective* (retrospectiva), parte dedicada à melhoria contínua.

A Review Meeting é o momento no qual o cliente e o time de desenvolvimento se reúnem para mostrar os incrementos feitos na Sprint. A Review Meeting também é uma time-box, ou seja, uma caixa de tempo rígida que não pode ultrapassar 2,5% do tempo total da Sprint.

Portanto, para uma semana de Sprint, teremos 1 hora dedicada ao consumo do tempo do cliente e da equipe para realização da *review*. Essa reunião é muito importante, porque paramos de "esconder" o time do cliente e vice versa.

A palavra "cliente" é recomendada pelo "*Scrum Guide*", porém vale trazer para junto das reuniões o "usuário". A ideia é envolver um ou ambos: cliente e/ou usuário. Junto ao time de desenvolvimento, eles auxiliam a entender melhor o lado um do outro. As relações entre o time de desenvolvimento com o cliente e vice versa podem não ser tão boas, já que existem alguns estereótipos, como o cliente ser alguém que apenas reclama ou o time de desenvolvimento não querer trabalhar.

Portanto, a Review Meeting é uma ótima oportunidade para unir o cliente, interessado na questão financeira, e o usuário, que relaciona-se diretamente com o produto, junto com a equipe de desenvolvimento.

## Como fazer a review meeting

Ela se inicia com o time mostrando os itens prontos, desenvolvidos durante a Sprint. Existem diferentes formas de se mostrar resultados, uma delas é abrir um arquivo feito no Power Point, e a equipe aponta o que foi realizado. Ao fazê-lo, no entanto, estamos perdendo uma oportunidade incrível de descobrir como o usuário utilizaria o produto. Portanto, uma maneira de apresentar os resultados é fazer um **teste de usuário**, que é pedir ao cliente ou alguém o mais próximo possível de usuário, para testar o produto.

O que pode acontecer ao pedirmos para o usuário testar uma funcionalidade específica é ela não funcionar conforme esperado. Não é raro o desenvolvedor acompanhar o usuário e ficar se perguntando como ele pode proceder de maneira equivocada, e o mesmo ocorre na visão do usuário, que fica pensando na maneira estranha como o produto foi elaborado.

Uma maneira de realizar testes é pedir para o usuário testar uma possibilidade, sem falar como ela funciona. A visão do cliente traz uma nova maneira compreender a aplicação. O **teste de usuário** é útil para descobrir de maneira eficaz o que e como o usuário pensa.

Quando o time vivencia a oportunidade de verificar o produto sendo manuseado pelo usuário, a elaboração do produto pode se tornar mais simples no futuro. Isto é, o time se preocupará em refletir se o produto funcionará conforme imaginado ou até começará a entender o raciocínio do usuário quando ele utiliza o sistema. Desta forma, fica ainda mais fácil acertar nas próximas elaborações.

Isto permite o surgimento de um leque de possibilidades: por exemplo, o usuário pode achar tudo maravilhoso, ou sua opinião pode ser o extremo oposto. Diante de respostas negativas, deve-se manter a calma. O *Product Owner* fará anotações, criará uma história de usuário ou qualquer outra maneira que esteja sendo utilizada para documentação. Essa história volta para o *Product Backlog* (lista priorizada do que devemos fazer no projeto).

Se o bug impedir o restante do projeto de funcionar, sua prioridade será direcionada como alta, mas se for um com menor importância, ele pode entrar no *backlog* de acordo com a prioridade que melhor lhe couber.

No processo de testes, o usuário também pode sugerir ideias, as quais devem ser anotadas e somadas ao *backlog*. Muitas vezes os desenvolvedores podem ficar temerosos em relação a novas ideias, mas no Scrum não há o que temer. Se for uma ideia deveras importante, será preciso o P.O. (*Product Owner*) negociar com o cliente. Isto porque, para que uma grande mudança ocorra, será preciso retirar alguma outra atividade.

O Scrum troca de escopo com muita facilidade e frequência, e afirma que as variáveis de software não se fixam: custo, escopo, qualidade e tempo. Para Métodos Ágeis, a qualidade não pode ser sacrificada, e essa é uma regra que deveria ser levada bastante a sério.

Mudar tempo e custo é bastante difícil, mas **mudar escopo** é mais simples.

Para finalizar a reunião nos despedimos do cliente e avaliamos se a Sprint foi bem ou mal sucedida, se bateu a meta no *planning* ou não.

**Resumindo**: na Review Meeting avaliamos os itens prontos e, se surgem novas ideias ou sugestões, criamos um novo cartão e colocamos no ***Product Backlog***, na prioridade que lhe convém.

## Definição de pronto

É importante que o conceito de **pronto** seja claro e que haja consenso entre os integrantes do time inteiro. Portanto, vamos pensar no seu critério: ele envolve uma sequência de passos que vamos cumprindo até o item ser considerado "feito".

Por exemplo, pensando em um exercício da faculdade, o "pronto" equivale àquilo que foi desenvolvido e testado manualmente. No entanto, trabalharemos com a ideia do pronto um pouco mais aprofundada, e colocaremos nela critérios. Um deles é que o produto está pronto quando foi minimamente desenvolvido e testado automaticamente.

Vamos um pouco além? Adicionaremos como critério de pronto o produto homologado pelo cliente. Mas talvez seu time também sinta a necessidade de garantir certa performance ou que cada item desenvolvido seja documentado. Isso também pode ser parte do critério de "pronto".

Para se obter um critério completo acrescentamos as seguintes ideias:

1. Testes de aceitação automatizados;
2. Desenvolvimento utilizando testes de unidade e integração automatizados;
3. *Code Review* para garantir que o código está dentro do padrão que o time aceita;
4. Homologação, que é pedir para a pessoa que comentou sobre o incremento de tarefas fazer a verificação. No nosso caso, isso é muito fácil e acessível. Também poderíamos pedir para o P.O. aprovar em um ambiente de homologação.

Portanto, o conceito de "pronto" varia dependendo das necessidades da equipe, que pode considerar que pronto é inclusive o produto posto em circulação. Ou seja, na Review Meeting são levados itens com que os usuários já estão em contato. Lembrando que mesmo o produto já estando em uso, a Review Meeting é essencial, para colocar usuário e/ou cliente em contato com a equipe.

É preciso atenção, pois o critério "pronto" também deve atender a um requisito mínimo. Por exemplo, se o critério de pronto for muito grande e complexo, a Sprint nunca será pequena. Porém, com disponibilidade e flexibilidade, o critério pode ser incrementado.

O "*Scrum Guide*" afirma que "o critério de pronto cresce conforme o tempo passa". Mas não necessariamente! Às vezes, quando o time amadurece, é possível retirar alguns critérios, pois determinados comportamentos já estão interiorizados no time e não é mais necessário defini-los.

Lembrando que manter as diretrizes que compõem o conceito de **pronto** em formato visual é essencial. Uma ideia é utilizar o quadro do time:

Independente de ser físico ou visual, este recurso auxilia a deixar tudo mais visível. As colunas são as partes que compõem o critério, e usar essa estratégia facilita a compreensão de qual parte estamos, pois podemos mover o conteúdo indicando que ele está realmente pronto.

**Retomando:** "pronto" é uma sequência de passos pré-estabelecidos pelo time, que devem ser feitos para certificar de que tudo estará funcionando corretamente. O "pronto" serve para garantir que o produto elaborado está finalizado e, ao mesmo tempo, é um conceito resumido o suficiente para caber na Sprint, do time-box definido.

# A Retrospectiva

Terminada a Review Meeting nos despedimos do cliente e partimos para o próximo processo. Claro que o início de uma nova etapa não é tão imediato, o seguimento pode ocorrer no dia seguinte ou após a reunião de revisão. Mas é importante destacar que entre a reunião e a retrospectiva, não existe trabalho de desenvolvimento.

**O que é retrospectiva?**

É o último time-box da Sprint. O primeiro é o *planning*, o segundo o desenvolvimento, o terceiro a Review e o último é Retrospective. Após finalizarmos a última etapa emendamos uma nova Sprint e as próximas iterações do processo.

A reunião de retrospectiva é a mais importante, pois é por meio dela que é possível exprimir o que afinal é agilidade. Esta reunião fornece a possibilidade de melhoria contínua em que pode-se "lavar roupa suja" para nos reinventarmos para uma próxima Sprint. O problema não é errar, é ficar enroscado sempre no mesmo problema, no mesmo erro! Quando essa situação ocorre é um desperdício de tempo e esforço.

Versões mais novas do "*Scrum Guide*" afirmam que a cada duas semanas de Sprint deve ser feita 1h30 de retrospectiva. A versão anterior do Scrum determina que a retrospectiva deve equivaler a 5% da Sprint, ou seja, igual a 2h a cada semana de Sprint. Particularmente, o investimento nas melhoras é algo que deve ser feito com certa constância, portanto, a versão anterior que traz a diretriz de 5% do tempo é mais adequada.

É preciso lembrar que retrospectiva não deixa de ser um time-box, portanto seu tempo de duração é rígido. Então, é bom fazer uso consciente desse tempo.

Retrospectiva é uma reunião na qual o time conversa, e dela participam:

* desenvolvedores
* Scrum Master
* Product Owner

Para começar a retrospectiva, iniciamos citando o *Prime Directive*, inclusive vários facilitadores da comunidade ágil mundial iniciam assim. *Prime Directive* é a diretiva primária, que resume-se a um parágrafo com o seguinte conteúdo: não importa o que descobriremos nessa reunião, consideraremos que as pessoas agiram dessa forma devido aos conhecimentos que possuíam na época, tempo e recursos disponíveis. Considerando esses aspectos, as pessoas fizeram seu melhor, e agora devemos seguir adiante.

Algumas empresas criam adesivos de parede com os dizeres da **Prime Directive**, justamente para que o time visualize com facilidade a mensagem. Retrospectiva é um momento de verificarmos as possibilidades para melhorar na próxima iteração, em um mesmo projeto e com o mesmo time e contexto. Todos devem saber que é um momento de melhoria contínua, não um momento de apontar culpados.

## Ações e Sugestões

A retrospectiva é o momento de melhoria contínua; é quando levantamos pontos positivos e negativos.

Uma forma muito comum de fazê-lo é deixar espaço para que todos falem ou façam perguntas. Porém, se há alguém tímido, ele provavelmente não se sentirá à vontade para participar. Uma estratégia é entregar canetas e post-its para todos e pedir para que anotem pontos positivos e negativos: um ponto por post-it. O facilitador da reunião pode passar e recolher esses papéis, ele mesmo colocando-os na lousa. Dependendo da maturidade do time, e se os integrantes não tiverem medo de falar, ele mesmo pode pegar o papel e colocar na lousa.



Os post-its podem ser agrupados caso os assuntos sejam parecidos. Esses grupos de post-its criam o que chamamos de *clusters*.

Ao observarmos os post-its, podemos verificar que os pontos que juntam aglomerações deles indicam problemas que necessitam ser analisados, ou de repente um aspecto positivo possível de ser discutido mais a fundo. Assim, as aglomerações indicam tópicos.

Supondo que iniciamos pelos pontos negativos, com vários post-its indicando que o cliente sumiu, enquanto a equipe precisava tirar diversas dúvidas com ele. Como agir diante dessa situação? O desdobramento poderia ser um muro de lamentações por parte da equipe, ou alguns integrantes poderiam partir para o famoso *wishful thinking*, pensando de maneira positiva para que a situação não se repita.

A partir da retrospectiva devem sair **ações**! Tanto dos pontos negativos quanto dos positivos.

No caso dos aspectos negativos levantados, qual deve ser a ação para que o problema diminua ou até desapareça no futuro?

No caso do cliente ter sumido, o que poderia ser feito?

Uma ação possível é o scrum master conversar com o cliente, explicando que suas ações acarretam em grandes impactos no projeto como um todo, portanto, o fato dele sumir no meio de uma iteração tem consequências. Ou então decidimos que o P.O. se informará melhor, mas isso pode causar mais trabalho para ele, que já está atolado de afazeres.

A solução seria ele coletar especificações mais detalhadas para que não precise mais falar com o cliente. Ou utilizamos o "chato da vez", a pessoa que ficará em contato direto com os clientes insistindo para que ele fale conosco. Ou podemos pedir para o cliente acessar diretamente o usuário, que quer muito que a funcionalidade seja desenvolvida.

Algo que já optamos fazer e até certo ponto arriscamos, e que acabou dando certo é: se o cliente não tem tempo para nos atender é porque a funcionalidade talvez não seja tão importante. Portanto, estabelecemos como regra de que, diante dessa situação, não iríamos desenvolvê-la.

As pessoas normalmente acostumam-se com qualquer restrição. Algumas são aceitas e outras não, claro. Mas dependendo da restrição, o público, cliente e usuários aprendem a lidar.

Voltando a falar de ações, independente daquilo que a equipe pensou depois da retrospectiva, é preciso deixar isso visível na área de trabalho do time (seja em ambiente virtual ou físico).

Na próxima retrospectiva, antes mesmo de levantar pontos positivos ou negativos, vamos verificar o que foi estabelecido como ação. Elas foram feitas? Deixaram de ser feitas? Melhorou? Eram necessárias? Ou seja, buscaremos aprender com as ações passadas.

# Daily Scrum

A *Daily Scrum* é rápida, com duração de no máximo 15 minutos, realizada no próprio ambiente de trabalho do time. Dessa forma, cada integrante compartilha o que fez, o que fará, e quais problemas enfrentou. Esse último aspecto serve para levantar os desafios enfrentados e superados, e os que ainda necessitam de resoluções.

A *Daily Scrum* é um momento no qual cada indivíduo do time precisa pensar e responder essas três perguntas: o que fez? O que fará? Quais problemas enfrentou? É parte do processo responder todo dia essas mesmas questões.

Segundo o "*Scrum Guide*", quem participa da reunião são os desenvolvedores. Também está implícito que o Scrum Master deve participar, e vários agilistas acreditam que, além deles, deveria ser considerada a presença do *Project Owner*, que é o representante do cliente dentro do time.

Diante disso, se a equipe está com algum problema que faça com que gastemos mais tempo que o esperado, é bom que o project owner saiba disso. Ele é responsável pelos negócios, portanto, é a pessoa certa para notar um problema, inverter prioridades e entregar o máximo de valor possível ao cliente.

Além disso, o P.O. faz parte do time, e essa é uma forma de criar confiança entre os desenvolvedores e a pessoa de negócios. Todos estão juntos pelo mesmo motivo! Portanto, o P.O. deve participar dessas conversas para ficar a par do processo de desenvolvimento do projeto, tanto para aspectos positivos quanto negativos.

Assim, o P.O. pode conversar com o time de maneira a rearranjar as atividades que estão sendo desenvolvidas para que tudo fique em ordem em casos em que ocorra adiantamentos ou atrasos.

Portanto, a *Daily Scrum* é:

* Uma reunião diária;
* com duração de até 15 minutos;
* feita em pé para cansar caso ultrapasse os 15 minutos;
* três perguntas principais devem ser respondidas: O que fez? O que fará? Quais problemas enfrentou?
* toda a equipe participa: desenvolvedores, Scrum Master e Product Owner.

## Quando fazer a Daily Scrum

É importante fixar que ela deve ser realizada **sempre no mesmo horário e lugar**. Claro que fixar isto em empresas que possuem regras de horários mais flexíveis pode ser complicado, portanto, daremos algumas dicas, saindo do prisma do *framework*.

É essencial que todos os integrantes do time estejam disponíveis para a reunião, dessa forma, marcá-la no início do expediente implica em possíveis ausências, pois algumas pessoas ainda não chegaram no trabalho.

Diante disso, é possível pensar que é mais plausível escolher o primeiro horário em que todos da equipe estejam na empresa. O problema de fazer isso é que a pessoa que acabou de chegar ainda está se situando, entrando no ritmo do trabalho. O mesmo acontece quando se opta pelo último horário. Para quem está encerrando o expediente, fica um pouco complicado, pois a pessoa está na iminência de ir embora.

Uma dica é evitar momentos extremos do dia - início e final do horário de trabalho - e utilizar intervalos existentes na rotina. Por exemplo: o time para e almoça ao meio dia? Então o *Daily Scrum* pode ser realizado às 11:45, ou, se metade do time volta do almoço e metade sai depois, o intervalo entre os almoços pode ser um momento bacana.

Outra opção é escolher outra pausa, algum momento utilizado para descontração do time. Por exemplo, durante muito tempo na *Alura* o Daily era aplicado antes da pausa da tarde, assim, quando as pessoas começavam a ficar com fome, já sabiam que esse era o momento do Daily. O interessante de se fazer isso é que mesmo a reunião acabando, o time ainda conversa sobre isso no momento do lanche.

A dica é fazer a reunião em momentos de pausa ou próximos a eles. Evitando que caia no início ou final do dia, e mantendo sempre o mesmo horário e lugar pra que não seja preciso pensar nessa logística todos os dias.

# Planning Meeting

A Planning Meeting limita-se a 5% da Sprint, e é um **processo de planejamento**. O time se reúne com o Scrum Master, que trabalha mais a parte da facilitação, os desenvolvedores, para pensarem na parte mais técnica de como realizar certas tarefas. Nisso, o *Product Owner* é responsável por priorizar um grupo de itens que agregue o máximo de valor possível.

O P.O. chega na reunião com o **Product Backlog**, lista ordenada ou priorizada dos itens a serem feitos durante o projeto. A ideia é sair da reunião com a lista priorizada dos itens e sub-itens técnicos que o time está comprometido a fazer durante a Sprint em questão. E também com uma meta, uma frase que exprima o valor a ser entregue ao cliente.

Resumindo, a Plannig Meeting é uma reunião de planejamento que reúne a equipe inteira. Entramos nela com uma lista de todos os afazeres e saímos com outra, de tarefas específicas a serem concluídas. Lembrando que o **Planning equivale a um tempo de 5% da Sprint**, se o time-box é de 2 semanas, passaremos 4 horas planejando. Para isto, o P.O. deve ter passado um tempo considerável pegando o topo do Backlog, os itens mais importantes, e refinando-os.

Esta parte de refinamento do topo do Backlog é um dos trabalhos do P.O. durante todo o desenvolvimento. Ele chega à reunião pronto, com o topo do *Backlog* refinado, itens menores que também façam sentido para os clientes e pessoal técnico.

**Podemos consumir cerca de 10% do tempo do time, pelo menos no início, para perguntar o que faz sentido e o que não faz**. Isso é feito individualmente, para que na Planning Meeting o P.O. traga tudo pronto.

## Mecânica da reunião

O P.O. chega com o *Product Backlog*, a lista de coisas priorizadas, e inicia a reunião apresentando o item de maior prioridade para a visão de negócios. Ele deve lidar apenas com a questão do cliente, sem "dar pitaco" em relação à parte técnica.

Os desenvolvedores vão discutir sobre o item, pensar na parte técnica, talvez quebrar o item de negócios em tarefas menores e mais técnicas e, a partir disso, estimar quanto de esforço deve ser empregado nesse tipo de tarefa.

Ou seja, não chegamos a discutir a nível de código. A ideia dessa discussão é compreender a complexidade embutida no processo e fornecer uma estimativa do que fazer. Assim, seguimos por cada item da lista de prioridades até passarmos um pouco do quanto o time consegue de fato fazer. Passando disto, vamos para a próxima parte da reunião, de negociação daquilo que realmente cabe na Sprint.

Por exemplo, há um item na lista que é muito grande, então iremos deixá-lo para a próxima Sprint. Isso não acarreta em nenhum problema, pois puxaremos algum outro item para cima.

Claro que isso é decidido em equipe, e se o P.O. achar que não é tranquilo deixar esse item maior de lado, é porque provavelmente não é uma boa ideia.

A ideia é ponderar, não podemos deixar muito tempo de trabalho vago e tampouco ficar com a corda no pescoço, com prazos muito estreitos.

O Scrum também faz parte da agilidade, sendo o objetivo trabalhar em ritmo constante indefinidamente. Isto faz parte dos princípios ágeis, e para isso é preciso ter uma noção do quanto conseguimos entregar sem nos sobrecarregarmos, e sem deixar espaços não preenchidos.



A ideia é que essa análise seja negociável, pois os desenvolvedores podem visualizar algo que pode dar problema no futuro, uma vez que a dívida técnica é muito grande. Assim, em vez de pegar uma tarefa nova a ideia é investir o tempo na solução da dívida técnica. Inevitavelmente esta falha técnica precisará de atenção, e isto pode custar mais caro se continuarmos adiando a situação.

A negociação pode ser feita conforme a demanda dos participantes. O essencial é pensar no *Sprint Backlog*, lista priorizada do que o time deve fazer na Sprint. Lembrando que não é nesse momento que decidimos quem vai fazer o quê: isso cabe ao escopo do Daily.

O que nós obteremos ao final desse processo é o que o time irá entregar ao final da Sprint.

Uma vez que o grupo está fechado é preciso se perguntar: qual é o principal valor de negócio que conseguimos tirar do time? Assim, a equipe decide, por exemplo, que três dos itens devem agregar muito valor de negócio à Sprint, portanto, a meta que vamos traçar pode ser feita a partir deles!

# Product Backlog, Sprint Backlog e suas partes

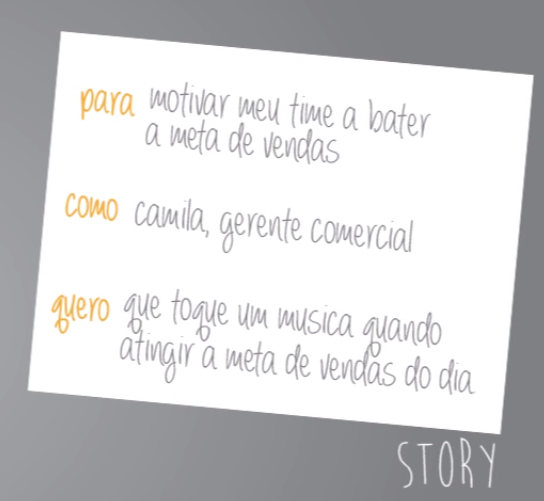
Estes itens podem ter qualquer formato, desde que sejam da esfera dos negócios. Todos os itens que o P.O. leva para o Planning contém a visão de negócios, isto é, a **visão do usuário**.

Cada item do Backlog deve ser algo que agregue valor ao usuário, seja lá em que formato. A sugestão que damos é aplicar a *story* (história em inglês).

Qual é a diferença entre história e caso de uso? A história é um item que agrega valor ao usuário, escrita de forma bem diferente ao caso de uso, deve ter um título, um porquê e para quem esse item é importante.

O template da história pode ser feito com um cartão de papel. A história serve para convidar o desenvolvedor a tirar dúvidas sobre os detalhes de uma tarefa. A ideia da história é que ela seja legível e fluente, como uma historinha, mesmo.

Um exemplo de história elaborada é a descrita abaixo. No caso, é para incentivar os funcionários a baterem uma meta. Como gerente de vendas, a Camila quer que toque uma música quando a meta for atingida:



Apesar de parecer pouco, isso realmente ajudou os funcionários a baterem a meta. Um possível título para essa história seria "Comemoração musical de meta". O título deve transmiti-la de maneira simples. Além disso, ela deve ser escrita de maneira simples e fluida.

Uma história faz parte do *Product Backlog*, e é apresentada no *planning*, no planejamento. Mas no momento de desenvolvê-la, de colocar a mão no código e fazer ele acontecer, vamos **paralelizar os sub-itens da história**, chamados de tarefa, *task*, para terminá-las o mais rápido possível.

Levando em consideração o exemplo anterior, dividiremos as seguintes tarefas:

* Seleção de músicas (pode ser feito por qualquer pessoa);
* o javascript para mandar rodar as músicas;
* uma notificação por WebSocket.

Essas tarefas formam a história que comentamos antes. Escrevê-la não é tão fácil quanto parece, por isso, mais adiante, há um exercício sobre isso.

Sabendo o que são **história** e **tarefa**, passaremos para as listas priorizadas e ordenadas dos itens. Elas auxiliam na organização e fazem o P.O. manter tudo em ordem, facilitando também a visualização da importância relativa de cada item.

O artefato padrão no Scrum é o **Product Backlog**, uma lista ordenada, que prioriza todos os itens de negócio e histórias até o final do projeto.

Quem cuida dessa lista é quem tem o domínio sobre o que entra e sai, e essa pessoa, o *Product Owner*, também estabelece prioridades nessa lista. O P.O. é o dono - *owner* - e manda nela, mas o cliente, os desenvolvedores e o Scrum Master podem influenciar o P.O. através de sugestões de re-priorização, colocando um item a mais ou tirando outro... Todo mundo pode dar sua opinião! Mas quem decide é o *Product Owner*.

Resumindo, a lista é do P.O., e nela constam os itens que devem ser desenvolvidos no projeto como um todo. Os itens mais prioritários são aqueles que estão no topo da lista e provavelmente já pensamos melhor a respeito deles quebrando-os em funcionalidades menores. Fica até mais fácil estimá-los e levá-los para o *planning*.

O processo de quebrar histórias muito grandes em menores de maneira que cada uma siga agregando valor ao usuário final, é um processo que chamamos de **refinamento - o *grooming* - do Product Backlog**.

Os itens menos prioritários, mais abaixo na lista de prioridade, não precisam ser todos refinados. É até bom que eles não estejam, pois não sabemos se serão úteis para o projeto futuramente. Pode ser que apareça algo mais urgente ou que agregue maior valor ao projeto, e neste caso o P.O. pode desconsiderar um item no qual já investimos certa quantidade de trabalho de refinamento.

Se algo perde importância no Backlog o item acaba ficando para baixo na lista de prioridade, e isso vai sendo deixado de lado até que o projeto acabe. Se ainda for interessante realizar itens e tarefas que ficaram para fora do prazo estimado, pode ser feito um adendo de contrato, prosseguindo-se a execução desses itens caso agregarem valor suficiente.

Ao chegar no *planning*, o P.O. traz o topo do *Product Backlog*, e isso é discutido com o time e quebrado em histórias e tarefas, fazendo com que seja mais simples estimá-las. Ao quebrarmos as tarefas em histórias, fica ainda mais simples de compreendermos suas complexidades.

É bastante natural que no Sprint Backlog não existam só histórias, mas também tarefas. Isso inclusive é esperado. No "*Scrum Guide*", comenta-se que cerca de 70% das tarefas vão surgir do *planning*.

O *Sprint Backlog*, diferentemente, é do time como um todo, e a regra é que a equipe pode renegociar prioridades (lembrando que o time é composto de desenvolvedores, P.O. e Scrum Master). Assim, é possível mudar o escopo de maneira a agregar valor - essa é uma decisão interna do time.

**O Sprint Backlog possui a regra de que o cliente não pode dar sua opinião**. Se algo entrou para ser feito nessa Sprint, então é porque são itens de maior prioridade. Se o cliente pensa em mexer nisso é porque o P.O. não conversou direito com ele, ou as prioridades não estão bem estabelecidas, e ele muda de opinião o tempo inteiro.

A única razão pela qual se faz um cancelamento de Sprint no Scrum é quando **o Sprint Backlog se altera tanto a ponto da meta não fazer mais sentido**. Os cancelamentos também podem ocorrer no caso de **problema de comunicação grave, um cliente muito indeciso ou eventualmente outro tipo de razão externa**, como uma justificativa legal. No entanto tais situações não são comuns ou não devem ser recorrentes.

**Retomando:**

* O *Product Backlog* é a lista priorizada das histórias que agregam valor para o cliente. São histórias que envolvem o projeto inteiro. Somente o P.O. mexe nele, mas todo o time pode palpitar;
* O *Sprint Backlog* engloba histórias e tarefas que estão no topo das prioridades. Conforme os itens entram nele, já quebramos as histórias em tarefas. No Sprint Backlog o time altera essas tarefas sem que o cliente palpite sobre elas.

# Scrum Master

Precisamos realizar reuniões curtas e, pelo tempo disponível, elas devem ser muito focadas. Porém, nem sempre é simples fazer isso. Uma das responsabilidades do Scrum Master é ser um facilitador das reuniões, ajudando o time a se comportar bem, com foco, auxiliando a entender o propósito em pauta. No fundo, ele explica o que é o Scrum e como cada encontro desses colabora para que o processo fique mais redondo e amarrado, tornando-o mais interessante de se trabalhar.

Apesar do Scrum Master fazer o papel de facilitador, ele ensina as pessoas a tirarem o maior proveito possível das reuniões. A facilitação é bastante presente, principalmente no início. A ideia é que com o tempo o time não necessite tanto da presença do Scrum Master, pois ele aprende a lidar com os time-boxes.

Ser educador, no entanto, não é o único papel que o Scrum Master desempenha. Ele também entra em ação em situações como o cliente sumir no meio do desenvolvimento e reaparecer apenas na *Review*, insatisfeito com o resultado entregado. Diante desta situação o Scrum Master pode conversar com o cliente e explicar que para evitar surpresas desagradáveis na Review é preciso que ele ou uma terceira pessoa acompanhe o desenvolvimento do produto.

O Scrum Master ensina a trabalhar com Scrum e agilidade. Por exemplo, se em dado momento a equipe parar de fazer o Daily Scrum, é responsabilidade do Scrum Master aproximar-se do time e relembrar a importância dessa ferramenta para a comunicação geral.

O papel do Scrum Master é facilitar o ensino sobre o que é o Scrum e as responsabilidades de cada uma das partes envolvidas. Não se trata de ser necessariamente o chefe, mas sim um líder servidor, aquele que está disponível para ajudar e facilitar a comunicação. Sua terceira função é mais prática, fácil e tangível, que é resolver impedimentos.

## Impedimentos

**Uma das responsabilidades do Scrum Master é lidar com impedimentos**, e é importante diferenciar o que é impedimento do que é problema.

Todo impedimento nasce como um problema, e a resolução disso é função do time. Algo está atrapalhando? Algum fator está induzindo a baixa produtividade?

Um exemplo de problema é "morrer de frio por causa do ar condicionado" ou "o cliente sumir". **Qualquer situação que esteja atrapalhando é um problema**.

Passa-se da categoria problema para impedimento **quando o time tenta resolver uma situação mas não é capaz**. Assim, mesmo dividindo com demais colegas na Daily, o time falha em lidar com a situação. Não compensa para o time, como um todo, parar para resolver essa questão ou realocar algumas pessoas produtivas para lidar com isso.

Portanto, o Scrum Master é responsável por resolver impedimentos que o time tentou solucionar, mas não conseguiu.

**Resumindo**: resolver problemas é função do time, mas o Scrum Master é responsável por lidar com impedimentos, pois parte de seu trabalho implica em educar as pessoas.

**Um bom Scrum Master trabalha para que os impedimentos de hoje não se tornem problemas amanhã.**

# Product Owner

O papel do Product Owner é ser **representante do cliente em um time de Scrum**. Ele é quem escutará as diversas opiniões dos clientes e chegará em funcionalidades que façam sentido para os negócios, de maneira a maximizar valor, captando ideias e prioridades do cliente.

**É muito importante que o Product Owner dedique muita atenção para manter o *Product Backlog* o mais atualizado possível**. Apenas relembrando que *Product Backlog* é a lista atualizada e priorizada do que se deve fazer no projeto, isto é, todas as tarefas e itens que estão inclusos em seu desenvolvimento.

Quando falamos que o Product Owner mantém o Backlog atualizado, estamos afirmando que caso haja algo na lista que não precisa mais ser feito, isto pode ser retirado. E se existir algo que seja solução de um *bug* ou uma ideia revolucionária, o P.O. pode adicionar esse item no *Product Backlog*.

Outra estratégia que o P.O. pode utilizar para manter a lista atualizada é **re-priorizar histórias**. A prioridade das tarefas pode diminuir ou aumentar. Se determinada tarefa ganha maior importância, sua prioridade sobe. Se algo que à primeira vista parece importante perde relevância, a prioridade diminui.

O papel do Product Owner, como mencionado, é agregar o máximo de valor possível e, **a cada Sprint, ele é influenciado tanto por clientes quanto pelo time de desenvolvimento**. Todos podem palpitar, mas a palavra final sobre as prioridades do Backlog é do Product Owner. Como o próprio nome diz, *owner* é "dono" em inglês, logo, "dono do *Product Backlog*".

Outra tarefa do Product Owner é **entender os problemas do cliente e ensiná-lo a trabalhar com histórias, se essa for a escolha de documentação do projeto**.

Além disso, o P.O. é responsável por **falar com o time de desenvolvimento quando houver dúvidas**. Nessas, pode acontecer algo interessante, como o desenvolvedor falar com o P.O. para tirar uma dúvida e não saber o que responder ou fazer. Esse é tema para outra aula!

Quando o desenvolvedor possui uma dúvida de negócios, algum detalhe que não está especificado na história, e falta ser implementado. Se o Product Owner souber resolver, ele responde na hora e o problema é solucionado.

Mas se o P.O. não souber responder, no caso de situações que envolvem questões mais técnicas, ou da esfera de finanças, ele buscará alguém a quem perguntar. O problema de pensar dessa maneira é que isso pode acarretar em problemas de comunicação, pois o Product Owner buscará a resposta junto de um terceiro, retornando com a solução para o desenvolvedor. Essa situação em muito se assemelha a um "telefone sem fio", o que cria lacunas na comunicação.

A recomendação para situações em que o Product Owner não souber responder a dúvida é que, junto ao desenvolvedor, deve se perguntar **diretamente a quem souber solucionar a questão**.

Isso é bom, já que com a presença do desenvolvedor fica mais fácil passar exatamente a dúvida. Essa é a maneira mais efetiva para resolver problemas. **Uma conversa cara a cara é recomendável** e mais simples para descobrir que detalhe está causando problemas e como solucioná-lo, principalmente se for algo técnico. **É praticamente impossível fazer uma documentação completa e que abranja todos os casos possíveis de falhas**.

Essa é uma das maneiras, com o P.O. junto para estar ciente do que foi discutido. Se o P.O. não achar necessário, ele nem precisa estar presente. Isso geralmente ocorre quando o time está avançado e funcionando bem sozinho. A solução que envolve o P.O., o desenvolvedor e uma terceira pessoa evita o telefone sem fio, uma desgraça para a comunicação no time.

Depois da primeira vez em que o desenvolvedor for falar com o cliente para tirar dúvidas, o cliente pode achar que é possível fazer solicitações diretamente ao desenvolvedor. Isto é, uma comunicação se estabelece entre cliente e desenvolvedor, sem passar pelo P.O. Nessa situação, a resposta do desenvolvedor - ou de qualquer pessoa da equipe que o cliente consultar - deve ser sempre a mesma: "claro que o pedido pode ser desenvolvido, porém você precisará consultar o P.O., responsável pelo controle do Project Backlog e da lista priorizada".

Assim, o P.O. permite que o desenvolvedor fale diretamente com o cliente, mas o inverso não é possível, pois desta forma agregamos menos valor dentro do que poderíamos, para o cliente.

# Desenvolvedores

**Desenvolvedor é aquele que ajuda a executar o projeto e o faz andar para a frente**: programadores, DBA, arquitetos, analistas, front-ends, testers, entre outros. São todas as pessoas e funções que **ajudam o projeto a sair do papel e virar um sistema ou produto final**.

Resumindo, com exceção do Project Owner e do Scrum Master, o restante do time é o grupo de desenvolvimento.

Mesmo o Project Owner e o Scrum Master podem ser *part-time* - parte do tempo - desenvolvedores. É mais usual encontrar Scrum Master como desenvolvedor do que Project Owner, mas é possível para ambos os cargos.

O desenvolvedor fará o trabalho que sempre fez, porém, com algumas outras responsabilidades. Ele destrincha as histórias por um viés técnico, por exemplo, de maneira a estimar o esforço envolvido. Este é o primeiro aspecto que diferencia o Scrum de outros desenvolvimentos de projeto: ninguém deve chegar e falar para o desenvolvedor qual deve ser seu trabalho.

Caso alguém diga "6 horas para executar uma tarefa", são 6 horas independentemente do trabalho levar mais ou menos tempo. E no Scrum esta situação não acontece, pois é **o time de desenvolvimento que estima trabalho e tempo necessários**. Após rodarem várias vezes o ciclo no Planning, o próximo passo é que o P.O. e os desenvolvedores negociem o que realmente cabe na Sprint.

Ou seja, **os desenvolvedores também decidem o quanto de trabalho pode ser feito em um time-box**. **Não é uma pessoa externa que define tempo, são os próprios envolvidos no trabalho**. E o cliente e o P.O. confiam no tempo estimado pelos desenvolvedores, justamente porque eles são os que melhor sabem sobre o que podem fazer.

Em algumas empresas as tarefas são atribuídas a diferentes pessoas por um terceiro. Mas no Daily Scrum é o próprio desenvolvedor e o time como um todo que escolhem, e não alguém externo, tampouco o gerente.

A função de distribuir tarefas que no Scrum costumava estar nas mãos de gerentes e outras figuras de comando, já não ocorre mais. É o próprio time e, especificamente, os desenvolvedores que são responsáveis por isso.

Os desenvolvedores passam a ser responsáveis pelo trabalho feito, e são eles próprios que estimam o quanto podem e são capazes de fazer. Assim como também afirmam quando podem fazer. É uma responsabilidade a mais? Sim! Mas é tratar profissionais como adultos, e é isso que almejamos conquistar após certo treinamento, Sprints e iterações.

No início, o Scrum Master terá mais trabalho, por conta de ser um momento de educar e explicar aos desenvolvedores envolvidos a questão do comprometimento. Porém, a tendência é que isso se ajuste com o tempo!

# Melhoria contínua

O Scrum Master e o Product Owner **não** podem ser a mesma pessoa, pois sua listagem de obrigações é muito grande para ser desempenhada por um único indivíduo. No entanto, o Scrum Master e o P.O. podem ser desenvolvedores, isso não tem problema.

Há algumas tarefas que cabem ao time inteiro, como um todo, como por exemplo a decisão do momento de contratação de uma nova pessoa, uma vez que a equipe sente a necessidade de um novo integrante.

Reforçando que os problemas nunca serão individuais, e sim do time inteiro. É função da equipe fazer o processo caminhar e, quando for necessário educar sobre o funcionamento do Scrum, o Scrum Master entra em ação.

Outro aspecto muito importante é **sempre aplicar as ações retiradas da retrospectivas e resolver os problemas à medida em que eles aparecem**.

Ou seja, a melhoria contínua do processo, da forma de comunicação, do projeto e das pessoas envolvidas é função do time como um todo. A maior vantagem de trabalharmos com uma célula de trabalho é que ela irá refletir continuamente sobre o trabalho, melhorando constantemente.

A melhoria contínua é essencial - vale a pena que isso seja relembrado ao time e também levado a sério. Todos devem pensar em melhorar constantemente, em conjunto.

Talvez essa seja inclusive a função essencial, **melhorar constantemente**, pois isso ajudará na entrega e agregação de valor ao produto final.