# Agile na Prática: Técnicas aplicadas para gestão ágil

# Iniciando um Projeto Agile

## Vision Box

A Vision Box é uma maneira simples de comunicar o principal objeto e as principais funcionalidades do nosso projeto. Ela é como uma representação física do que seria o Project Charter da maneira clássica, Waterfall.

Teremos a função de um Agile expert, por isso, precisaremos entender a história de nosso cliente. O cliente se chamará Star Web e será um time desenvolvedor de aplicativos. Ele terá sido contratado por uma empresa chamada Fast Sale, a fim de desenvolver um aplicativo mobile, o Sales Handy.

Esse aplicativo deverá ter um algoritmo para identificar oportunidades de venda e gerar insights sobre o mercado. No primeiro release, a empresa Fast Sale (para a qual colaboraremos como Agile experts) terá solicitado que o Sales Handy seja desenvolvido em Android e Ios.

Depois da entrega do primeiro incremento, nos próximos releases ele deverá permitir o acesso simultâneo de 1000 usuários, além de alimentar o banco de dados da companhia.

Nosso primeiro passo para a construção da Vision Box será pegar uma caixa e encapá-la com papel, que poderá ser branco ou pardo. Aprenderemos passo a passo o que a Vision Box deverá conter.

A Vision Box terá o nome do produto escrito na frente. No nosso caso, será o Sales Handy, mas se uma equipe ainda não possuir essa informação, poderá criar um nome fictício e colocá-lo.

Em segundo lugar, deveremos recortar e colar, ou até desenhar nessa mesma face da caixa uma figura de como nosso produto se pareceria. No nosso caso, teremos uma imagem impressa de um celular com o aplicativo aberto. Mas essa representação pode ser feita de outras formas, a depender do produto e da criatividade da equipe.

Em terceiro lugar, na frente da Vision Box também haverá um slogan. Nosso slogan será "As melhores oportunidades para os melhores vendedores".

Então, na frente da caixa, teremos o nome do produto, uma imagem e o slogan.

Na lateral da caixa colocaremos o site da Sales Handy, [www.saleshandy.com](http://www.saleshandy.com/) . No verso, haverá as principais funcionalidades do produto, para que lembremos delas quando estivermos trabalhando no desenvolvimento do aplicativo. Escreveremos, então:

"Disponível para Android e Ios

Alta escalabilidade

Interação com banco de dados personalizado para sua empresa".

Esse texto poderá estar tanto escrito à mão quanto impresso.

A Vision Box é de grande valor primeiramente por ser tangível. É uma caixa e podemos pegá-la. Ela deverá ficar na sala de nosso time de trabalho. Conforme as pessoas passem por lá, vão querer saber do que se trata, e diremos que é o produto no qual estamos trabalhando. A caixa se torna, então, uma maneira clara de chamar a atenção para nosso projeto.

Há Vision Boxes de diversos tamanhos. Criaremos uma de tamanho médio. Porém, elas podem ser imensas, e para não ocupar tanto espaço e para não perdê-la, há empresas que chegam a pendurá-las no teto.

Ela também ajuda a deixar claro para as pessoas que trabalham próximas de nós em outros projetos a que exatamente estamos nos dedicando, pois a caixa sempre terá essas informações registradas.

Além de tudo, às vezes estamos trabalhando questões bastante específicas relacionadas ao projeto, então a Vision Box nos permite uma orientação quanto ao projeto como um todo.

Então, a sugestão de criar uma Box é válida mesmo quando o projeto já foi começado, se for possível unir o time num horário determinado para contruí-la. Ajudará na motivação, será divertido e teremos uma maior noção de para onde estamos indo, inclusive se houver um membro novo na equipe.

Essa pode ser uma ferramenta muito mais poderosa do que um documento extenso como o Project Sharter, geralmente usado nos projetos mais conservadores.

## Tweet

Aprenderemos agora a como fazer um Project Tweet.

O Tweet é uma prática que nos ensina a expressar uma ideia de maneira simples, exercitando, então, nosso poder de síntese.

A instrução geral do Tweet é que descrevamos o objetivo do projeto e as principais prioridades dele através de uma frase curta.

O conceito do Tweet veio do próprio site Twitter, que costumava permitir postagens de no máximo 140 caracteres.

Então, veremos um exemplo de como seria um Tweet falando a respeito de um projeto, ou seja, o Tweet como prática ágil:

"Aqui nasce o **app** que mudará o futuro da análise de vendas no mundo e a **vida do vendedor**, Nosso time é parte dessa história **#SalesHandy**" Restam: 5

Restaram ainda 5 caracteres e já conseguimos expressar as ideias principais.

Destacamos os pontos importantes na frase, são eles: o produto que desenvolveremos (um aplicativo), a funcionalidade dele (análise de vendas), o usuário mais provável (o vendedor), e o nome do aplicativo (Sales Handy).

O Tweet costuma ser utilizado pelos times estampando uma faixa, por exemplo, ficando exposto para quem passar pelo local qual é o objetivo maior do projeto em que estarão trabalhando naquele momento.

Só conseguiremos escrever esse tipo de frase impactante e sucinta através da prática. Para treinar,mapearemos as palavras chaves e contaremos os caracteres delas. É possível destacar os verbos Automatizar (11 caracteres) e Gerar (5 caracteres), que podem ser encontrados nos objetivos do nosso cliente.

"Análise de Vendas 17 App 3 Automatizar 11 Informações 11 Oportunidades 13 Venda 5 Vendedor 8 Gerar 5 Insights 8 Mercado 7

#SalesHandy 11"

Temos, então, uma lista de palavras para utilizar, e é preferível escolher entre 3 ou 4 delas, deixando os demais caracteres livres para escrever os conectores da frase.

Poderemos, então, tentar escrever um Tweet relacionado ao aplicativo, lembrando sempre que o mais importante será conseguir demonstrar os objetivos para a criação desse aplicativo e quais serão as principais funcionalidade dele.

Será importante escrevermos o Tweet como uma forma de exercitar o que aprendemos sobre essa prática antes de darmos sequência ao nosso curso.

# Trabalho em Equipe

## Contrato do Time

Essa é uma ferramenta a fim de estabelecer regras de conduta para equipes ágeis. Esse Contrato contará com diversos itens que as pessoas da equipe terão sugerido acrescentar. Ele poderá ser no formato de um mural, por exemplo, e cada regra será acrescentada conforme a equipe as escreve.

Alguns exemplos de condutas seriam "Dar e aceitar feedback constante", "Vivenciar os valores do Agile", "Ser honesto". Sendo essas sugestões da própria equipe, o Contrato terá muito valor, ajudando a estabelecer relações saudáveis entre os indivíduos.

O Contrato será do time para o time e é muito importante que ele seja unânime. Ele não será democrático, todas as pessoas envolvidas nessa construção deverão concordar com cada um dos pontos, pois isso gerará comprometimento. Então, ele não será elaborado por meio de votação.

Ele será um documento vivo, ou seja, não é inflexível. Deverá ser revisado periodicamente, o que geralmente acontecerá durante retrospectivas.

Isso também funcionará para apontar regras de conduta, facilitando o apontamento de comportamentos que não se adequem ao que foi acordado. Em vez de dizer diretamente que algum membro da equipe está cometendo um erro, será possível relembrarmos que a atitude correta a ser tomada numa determinada situação já estará pré-estabelecida pelo Contrato.

Então, ele facilita conversas difíceis. Pode acontecer, por exemplo, de uma equipe estabelecer que nas reuniões cada membro terá um tempo para falar determinado. Se uma pessoa Sênior na função que exerce, que costuma dominar situações e exerce mais influência descumprir essa regra, o time pode simplesmente apontar o que o Contrato dirá a respeito do tempo de fala. Será muito valioso todos estarem submetidos as mesmas determinações, incluindo pessoas mais poderosas.

Outra situação para exemplificar seria o caso de um gerente, que também faz parte do grupo, não estar muito integrado, possuindo uma sala diferente, maior do que os demais.

Uma pessoa mais corajosa poderia sugerir que ele pudesse trabalhar conjuntamente, havendo uma rotatividade de posições para que as outras pessoas também se sentassem na mesa do gerente. Será muito disruptivo o gerente concordar e esse processo de um funcionário comum sentar na mesa do gerente e um gerente na mesa do funcionário acontecer.

O Contrato, então, ajuda a promover conversas encaradas como um tabu. Ele também auxilia a evitar os comportamentos inadequados, se for combinado nas regras que não pode haver bullying, fofoca, etc., havendo a possibilidade de apontar o acordo caso ocorra alguma situação do tipo.

Ele pode e deve variar de equipe para equipe e de projeto para projeto. Não existe um modelo válido para todas as equipes, ele deve ser gerado de acordo com as necessidades e desejos.

A prática funciona tanto para times colocados quanto remotos. As pessoas podem não estar no mesmo lugar e o Contrato ser compartilhado ainda, assim. Ele não estará numa parede, nesse caso, mas deverá estar visível a todos, numa Wiki, por exemplo.

Esse é o conteúdo básico sobre o Contrato do time, mas ainda aprenderemos as armadilhas nas quais não devemos cair na hora de construí-lo.

## Armadilhas ao Fazer um Contrato

Essa é a aula sobre as principais armadilhas na hora de fazer o Contrato da equipe do nosso curso de Agile na prática.

Muitas vezes temos boas intenções, mas quando fazemos o Contrato, cometemos erros que comprometem o valor dele para o time. Então, vamos entender algumas das armadilhas relacionadas à essa prática.

A armadilha número 1 é deixar alguém dominar as discussões. Nos times, sempre haverá alguma pessoa com uma postura de líder nato, mesmo que não tenha uma posição oficial de liderança. Alguns são mais falantes e impõe mais a opinião, mas a equipe não deve permitir que o contrato não tenha as mesmas implicações para todos de forma igualitária.

A segunda armadilha é que quem está fora do time não deve de forma alguma opinar para a criação das regras de conduta, mesmo que sejam pessoas próximas ao grupo ou que ocupem cargos importantes dentro da empresa em que o projeto estará sendo desenvolvido. As pessoas de fora não estarão comprometidas com a iniciativa e atrapalharão o intuito da criação do Contrato.

A armadilha de número 3 é que existe a possibilidade de alguma pessoa tentar forçar uma regra, sendo que o contrato deve ser unânime. Ás vezes existem regras que fazem parte das determinações da organização, mas elas não estarão no Contrato. Nenhuma regra poderá ser estabelecida como uma obrigatoriedade.

A quarta armadilha é não revisar o Contrato periodicamente. Se ele não for revisado, perderá valor com o passar do tempo, assim como acontece com as leis, por exemplo. Será necessário atualizá-lo para rever se os acordos estabelecidos por ele ainda serão válidos.

Um exemplo é que uma equipe pode em algum momento ter criado o acordo de haver um Happy Hour uma vez por mês. Mas ele é opcional, então ninguém descumpre a regra se precisar faltar. E haver Happy Hour pode não representar um problema para a equipe, e ele ocorre normalmente quando as pessoas puderem comparecer. Assim, essa deixa de ser uma regra válida e a manutenção dela no contrato pode ser repensada.

A quinta regra é a mais perigosa. Fazemos o Contrato unânime, com todos de acordo, mas em vez de ficar visível, ele vai para a gaveta. É fundamental que ele fique pregado à parede, onde todos possam vê-lo, ou que todos tenham acesso para sempre visualizá-lo virtualmente se o time for remoto.

Já sabemos as principais armadilhas dessa ferramenta. Posteriormente aprenderemos mais sobre como nos tornar grandes facilitadores numa sessão de Contrato do time. '

## Como Facilitar um Contrato

En primeiro lugar, facilitar qualquer sessão será muito importante para nós na vida de agilistas, e esse é o momento de unir o time todo e promover um encontro para estabelecer as regras de conduta.

Uma técnica para a facilitação é a dos 4P, ou PPPP, sigla para alguns pontos importantes nesse processo, nomeados em Inglês. O primeiro P significará People, ou pessoas. É necessária a reunião do time envolvido no desenvolvimento do projeto para poder formular o Contrato.

É indicado que a reunião aconteça num lugar confortável e afastado do ambiente de trabalho. É interessante haver uma mesa redonda para as poderem ficar de frente umas para as outras e se olharem. A mesa também ajudará para fazer anotações.

Deve-se evitar de levar os notebooks, bem como desligar os celulares para se concentrar na sessão. A não ser, é claro, se tratando de urgências, mas é melhor evitar ao máximo as interrupções.

O segundo P será de Post-its. A sugestão, logo, é a de que compremos e usemos muitos Post-its, preferencialmente os coloridos, para fazer um contrato bonito. Eles deverão ser de tamanho médio ou grande, com bastante espaço para escrever. Todos deverão ter uma boa quantidade de posts, podendo escrever sem a preocupação de economizá-los.

Nosso terceiro P será de Pen ou Pencil, ou seja, canetas ou lápis em Português (o único P cuja tradução não será iniciada com a mesma letra). As canetas deverão ter a ponta mais grossa, para permitir que as regras escritas possam ser lidas com uma distância maior. Podemos variar nas cores, também, para compôr visualmente nosso contrato.

O último P fará grande diferença na nossa vida de facilitadores. Será o P de Pizza. Ele representará que sempre que formos facilitar uma sessão, é indicado levar comida. Pode ser pizza, mas também balas, bombons, enfim. O ponto é que a comida ajudará as pessoas a se sentirem relaxadas. Não a toa, chineses fazem negócios em torno da mesa de refeição. Assim, conceber as regras da equipe será mais confortável.

Revisando, então, os 4P serão: People, Post-its, Pen e Pizza.

Como facilitadores, não precisamos ser Scrum Masters ou termos posições diferenciadas no time. É uma função que qualquer um pode exercer.

Será muito importante, ao começar a sessão, agradecer todas as pessoas envolvidas por estarem ali e estabelecer uma estimativa do tempo que a reunião irá levar, para evitar que a equipe fique ansiosa.

O facilitador deve, então, estabelecer uma agenda com objetivos. Ele esclarecerá o porquê da reunião, quais serão os resultados esperados ao final e quanto tempo ela deverá tomar.

Quem atuar como facilitador precisará levar perguntas para a equipe refletir sobre suas demandas. Apesar de cada facilitador saber que tipo de pergunta melhor se adequará à dinâmica de seu time,há algumas sugestões de perguntas chave.

Um dos exemplos de pergunta guia na sessão será "Como podemos contar um com o outro?". Essa questão contemplará uma equipe ágil, em que há uma troca constante entre os membros, e levantará ainda outras questões que também tenham importância.

Inclusive, "O que é importante" poderá servir como uma pergunta guia para o time pensar no que valoriza, o que deseja e o que priorizará.

"Quais são as experiências negativas que queremos evitar?" é uma pergunta que também permite trazer a tona falas que em outro momento poderiam não surgir. É possível que na criação do Contrato, especificamente, as pessoas se permitam falar sobre essas experiências negativas a ser evitadas

Na hora do Brainstorm, quando as perguntas serão colocadas e todos estarão com canetas e post-its para escrever, o mais indicado é que se faça silêncio. Isso porque existem alguns fenômenos quanto à fala.

O primeiro desses fenômenos é o Band Wagon Effect, ou efeito vagão. Quando alguém que costuma ser influente começa a falar sobre seus pontos de vista, as pessoas que ainda estão em dúvida de como se posicionar tendem a seguir a primeira. É como um comportamento de manada. Assim, quando as pessoas fazem o Brainstorm em voz alta, o grupo pode ser influenciado em vez da autorreflexão por parte de cada um ser estimulada.

O segundo efeito se chamará Highest Paid Person Opinion,ou a opinião da pessoa mais bem paga, em tradução livre. Se trata de como a pessoa mais bem posicionada na empresa da equipe poderá ter uma influência maior também.

Esses dois efeitos podem dificultar que todos se expressem, e o Brainstorm silencioso, escrevendo uma ideia por post-it para ser lido com facilidade posteriormente, auxiliará quem é mais tímido a não ser influenciado. Devemos lembrar de escrever nos post-its sempre com uma letra bastante legível, já que isso será pregado na parede ou num flip chart para ser sempre enxergado.

10 minutos, em média, costuma ser tempo o suficiente para as pessoas escreverem. Mas é importante avaliar como estará a energia das pessoas durante essa produção. No começo haverá muita energia, que depois diminui ao longo do tempo.

Quando esse tempo acabar, dependendo de como a agenda estará sendo controlada, poderemos sugerir mais alguns minutos para as pessoas concluírem suas ideias. Depois, começaremos a "clusterizar" os post-its, ou seja, agrupar por semelhanças de ideias.

Podemos usar o exemplo de post-its com o assunto pontualidade e atrasos. "Chegar no horário em reuniões", "Atrasar máximo de 3 minutos para reuniões", "Se me atrasar, pago café para o time". Quando houver o Pitch Chart, colocaremos esses post-its juntos e os circularemos. Ao final, tentaremos sumarizar essas ideias num post -it único. No nosso exemplo, ele ficaria de forma semelhante a "Atraso máx. de 3 min. Pagar café para o time se > 3 min."

Outra questão é que nos primeiros Contratos de time geralmente aparecerão as dores da equipe. Por exemplo, caso apareça uma frase como "Não jogamos o jogo da culpa" isso significará que provavelmente a equipe tem cometido esse erro e existe o incômodo. Ou ainda, se surge "Ser honesto" entre as diretrizes de conduta, é possível que o time enfrente problemas com relação à honestidade.Deveremos estar atentos a isso.

Com o tempo, conforme ocorram as retrospectivas desse Contrato, teremos a oportunidade de observar se tem sido possível refinar esses comportamentos a ponto de alguma regra se tornar obsoleta. Por exemplo, chega-se à conclusão de que realmente as pessoas são honestas. Será interessante acompanhar essa evolução conjunta de uma equipe de trabalho.

Tendo consolidado as normas do Contrato, todos deverão lê-las. Se todos estiverem de acordo, celebra-se a finalização dessa tarefa. Porém, se alguém estiver em desacordo com alguma das ideias, teremos opções, como reescrevê-las. Se a regra "Atraso máx. de 3 min. Pagar café para o time se > 3 min." não for unânime, talvez possa-se chegar ao consenso de manter os 3 minutos de tolerância, mas sem a necessidade de pagar o café.

Existem outras opções além de reescrever, como foi feito no caso da regra dos 3 minutos de atraso (tirando a obrigatoriedade de pagar café caso ultrapassasse). Poderemos também discutir, estabelecendo um tempo para a discussão, moderada pelo facilitador. Também pode acontecer dessa regra ser removida, pois é necessário lembrar das armadilhas e de como as regras não podem ser forçadas. Um facilitador deve se atentar que nenhuma regra esteja sendo empurrada no Contrato.

Precisaremos nos lembrar que visibilidade é tudo. Uma vez que o Contrato for terminado, devemos pregá-lo num lugar onde todos passem diariamente e possam vê-lo sempre, inclusive pessoas de outras equipes. Isso trará força e validação para ele, além da facilitação de simplesmente apontar caso alguém aja contrariamente à alguma norma.

# Mapeando a Vibe

## Mood Marbles

Mood Marbles é uma prática ágil para poder capturar o humor de um grupo. Isso é valioso porque muitas vezes as pessoas não falam abertamente sobre como estão se sentindo, e essa técnica nos permite acessar isso de forma rápida, barata e pouco invasiva.

Veremos como implementar essa técnica numa equipe. A tradução para Mood Marbles seria algo como "bolinhas de gude do humor". Mas não precisamos necessariamente usar bolinhas de gude. Podem ser bolinhas de piscina de bolinhas, ou mesmo post-its.

O time decidirá o que cada cor de bolinhas deve significar. Por exemplo, verde que está tudo bem, amarelo, neutro, e vermelho ou rosa, que algo vai mal. Também será estabelecido se os humores dirão respeito apenas ao trabalho ou se tratarão do humor das pessoas de modo geral.

É recomendável que as bolinhas fiquem dentro de uma cesta transparente, ou de uma sacola de plástico também transparente, para podermos enxergar as cores predominantes. É possível se identificar escrevendo nomes em cada bolinha, e entenderemos quais são as vantagens e desvantagens em se identificar quando repprtamos nosso humor.

O poder dos Mood Marbles é que de longe já será possível enxergar qual tem sido o humor predominante de um grupo por causa das cores majoritárias. Especialmente se você é um Scrum Master ou Agile Coach, poderá rapidamente perceber como está o grupo naquele dia.

A equipe combinará que em algum momento do dia cada um colocará sua bolinha representando o humor. É recomendável que isso não ocorra assim que as pessoas chegam ao trabalho, para o humor não estar influenciado por fatores externos, como pegar trânsito ou brigar com alguém antes de sair. O melhor horário será definido pelo time.

O Mood Marbles não deverá ser complexo, pois a ideia é capturar numa imagem como está a equipe naquele dia, sem acrescentar mais carga de trabalho.

Se tratando das vantagens de expressar o humor anonimamente é, em primeiro lugar, que os tímidos podem se sentir mais à vontade para isso. E pensando nos valores ágeis, o grupo como um todo estará em pé de igualdade, pois não estarão sendo captados dados sobre nenhum indivíduo em específico.

As desvantagens são que pode acontecer de uma pessoa precisar de uma atenção especial por estar enfrentando problemas, e a identificação mostrará se constantemente ela apresentará um humor negativo, permitindo que a equipe preste mais atenção à situação.

Outra desvantagem é que dependendo do tamanho do time, por exemplo, 10 pessoas, um número que está quase muito grande para ser ágil, ou mesmo da maturidade do time, pode acontecer das pessoas começarem a fofocar e especular sobre quem acrescentou um humor negativo.

O Agile Expert será a melhor pessoa para propor a técnica, e o time como um todo decidirá se a identificação ocorrerá. Porém, é recomendável começar com o anonimato.

Essa é uma técnica BBB, boa, bonita e barata, pois os custos de comprar bolinhas de piscina não serão altos. E é uma prática amigável e fácil de ser introduzida em qualquer grupo de trabalho. É uma boa porta de entrada para que a equipe comece a adotar práticas ágeis, fornecendo a visão de como o time está a cada dia.

É possível tirar uma foto por dia do Mood Marbles para acompanhar as tendências do time, e fará diferença observar os humores sem precisar perguntar para as pessoas como elas se sentem diretamente.

## Calendário Niko Niko

Estamos na aula do Calendário Niko Niko. Ele é uma prática para ajudar a mapear o nível de satisfação de um grupo em um período de tempo.

No Calendário Niko Niko as pessoas precisam se identificar. E diferentemente dos Mood Marbles, as pessoas expressam o estado de espírito ao final do dia.

Um exemplo de Calendário Niko Niko seria uma tabela com uma linha para cada membro da equipe e colunas para os dias da semana trabalhados. Para representar os humores, as pessoas poderiam acrescentar rostinhos. Os verdes seriam felizes, amarelos, neutros, e rosas ou vermelhos, tristes.

Então, como ele captura os humores conforme um período de tempo, será possível perceber alguns padrões. Por exemplo, na segunda-feira, o time pode estar todo chateado de acordo com as carinhas do Calendário. Na terça, mais da metade do time pode ainda estar chateado, e na quarta-feira haver uma melhora.

Se na próxima segunda-feira a equipe toda apresentar um humor feliz, significa que provavelmente aconteceu alguma transformação. É importante observar isso ao longo das semanas, pois é possível que haja algum dia em que comumente todos fiquem mais chateados. Nesse dia, pode ser interessante promover uma conversa com o grupo para entender os motivos da insatisfação.

O Calendário Niko Niko foi criado no Japão por Akinori Sakata em janeiro de 2006, num grupo de desenvolvimento de software. Ele foi feito pela primeira vez com adesivos redondos colados, com a cor significando o nível de humor e satisfação. Pode ser produtivo que além de colar, as próprias pessoas envolvidas desenhem as carinhas nos adesivos. O adesivo colorido comunica melhor os humores à distância devido as cores correspondentes, apesar de ser possível também apenas desenhar os humores.

Embaixo das carinhas do primeiro calendário Nikp Niko estão anotadas o número de horas extras que as pessoas fizeram no dia que as adicionaram. Olhando para a coluna do dia 29, há uma pessoa triste que fez 4 horas extras e uma pessoa que fez mais horas extras, 4,5h, mas está feliz. Outra está triste e fez 4,5h. Outra está feliz, mas não fez horas extras. Isso indica que não necessariamente são as horas extras que deixam as pessoas infelizes.

Esse é mais um dado que há possibilidade de implementarmos no Calendário, caso a equipe faça muitas horas extras. Mas o melhor é que não faça e o time consiga entregar os resultados com uma velocidade ideal, sem precisar sobrecarregar as pessoas em momentos determinados.

Então, o Calendário Niko Niko é uma maneira simples de monitorar como a equipe está se sentindo. Ele pode ser utilizado por Sprint ou Iteração, principalmente quando já se trabalha em esquema ágil, mas também podemos utilizá-lo por release ou semana. Fazendo-o por Sprint, esse se torna um dado para levar às retrospectivas, podendo suscitar discussões interessantes.

Aprenderemos, então, a fazer um termômetro do humor, que é relativamente simples mas oferece boas informações para quem for facilitar a retrospectiva. Para cada nível de satisfação, será atribuído um valor. Por exemplo, 5 para as carinhas felizes, 3 para as neutras e 1 para as tristes. Depois, somaremos os valores para calcular os humores relacionados a cada período. Se estivermos mapeando todos os dias, somaremos todos os humores de cada um dos dias da semana.

Por fim, tiramos a média desse valor. Trabalhando 5 dias por semana, dividiremos o valor da soma por 5. Suponhamos que essa média aritmética tenha dado 18,2. Marcaremos esse valor no termômetro, que deverá ser desenhado também, para que esses valores sejam visíveis. Então, poderemos levar só esse termômetro para a retrospectiva e ele colaborará para discutir os motivos dos humores negativos. Podemos colá-lo posteriormente na parede de trabalho, servindo como um indicativo de como o grupo tem se sentido a cada Sprint.

Como essa é uma média simples, não é necessário fazermos planilhas em Excel ou aumentar a quantidade de trabalho. No começo das práticas ágeis é melhor usar papel, pois assim não há a intermediação de uma ferramenta e ninguém precisa aprender a usar uma ferramenta nova, em especial quando falamos de grupos trabalhando num mesmo escritório.

Devemos focar nas Big Visual Charts, ou seja, métricas grandes, visualizáveis, palpáveis, sempre numa parede ou quadro branco. Assim há maior flexibilidade e uma desburocratização desses processos.

Enfim, acompanhando os humores ao longo das Sprints por meio do termômetro, será possível levar essa informação para a release. Esperamos poder adotar essa prática.

# Estimativas e Histórias de Usuário

## Estimando Usando T-shirt

Nessa aula aprenderemos a Estimar usando T-shirt. Essa é uma prática eficaz, rápida e muito utilizada nos projetos ágeis. Então, entenderemos como usá-la apropriadamente.

Começamos a usar as T-shirts no começo do projeto, como uma estimativa geral, para termos noção do tamanho e do tipo do esforço que será empregado no desenvolvimento desse projeto. A estimativa deverá ser utilizada antes mesmo do planejamento da primeira release, se trabalharmos num esquema em que haja release.

Com essa prática, mapearemos as funcionalidades, como as melhorias e incrementos grandes de projetos em desenvolvimento, como os softwares.

Devemos lembrar que as T-shirts são dividas pelos tamanhos P, M ou G. É mais recomendável trabalharmos apenas com esses tamanhos de camisetas para evitar a ilusão de precisão, pois ainda estaremos afastados disso numa primeira estimativa.

A prática se chama T-shirt porque quando compramos uma camiseta, não informamos aos vendedores de uma loja nossas medidas de forma minuciosa. Nós apenas dizemos o tamanho da camiseta que procuramos, M, por exemplo. E o tamanho M pode variar bastante dependendo da loja. Usaremos o mesmo princípio para se referir ao tamanho das funcionalidades do nosso produto em desenvolvimento.

Teremos como exemplo a empresa Fast Sale, que desenvolverá um aplicativo mobile chamado Sales Handy. Consideraremos algumas funcionalidades, apenas para ilustrar, e atribuiremos tamanhos de camisetas para elas.

Salvar dados de oportunidades de venda e perfis de usuário será uma funcionalidade de tamanho P. Consideraremos como de tamanho M a Integração ao banco de dadps da Hard Core Data. Permitir acesso em tempo real ao banco de dados será uma funcionalidade G.

Classificando mais algumas funcionalidades, Gerar relatórios e gráficos no app será de tamanho P e Desenvolvimento em Android e IOS, M. Então, o importante será comparar o tamanho relativo de funcionalidades, não o valor absoluto. Assim, avaliando o tamanho de features diferentes, poderemos considerar que ambas se encaixam no mesmo tamanho. Isso ocorrerá na maioria dos tipos de estimativas ágeis.

Estaremos mapeando de uma forma mais grosseira, mas nossa primeira estimativa deverá fazer sentido.

Aprenderemos nesse curso sobre como escrever Story Cards, nos quais serão documentadas as histórias de usuários. É recomendado alinhá-los às funcionalidades com que os relatos escritos neles estiverem relacionados

Assim, será mais simples visualizar se a quantidade de cards ligados à funcionalidade é compatível com o tamanho dela, por exemplo, se a funcionalidade a qual estimamos uma T-shirt G também terá mais cards relacionados a ela.

Não significará necessariamente que o número de cartões refletirá o quão complexa será a funcionalidade, porque as histórias não estarão estimadas ainda. Mas, dessa forma, teremos a opção de observar depois que fizermos a estimativa de pontos nos cartões de usuários, e saberemos se estimamos bem em T-shirt na primeira rodada, auxiliando que estimemos melhor das próximas vezes.

## Como Escrever uma História de Usuário

Hoje aprenderemos como escrever histórias e por que usá-las.

As histórias servem para facilitar a comunicação entre o time de desenvolvimento e o de negócios. Elas devem ser entendidas pelos dois grupos.

Utilizando as histórias, em vez de focar essa conversa nos detalhes técnicos do desenvolvimento do produto, como detalhes do código, focamos na funcionalidade, ou seja, no que ele será capaz de fazer.

Por isso, quem escreve uma história é o Product Owner, e não o time de desenvolvedores. Uma segunda razão para isso é que o cliente é quem sabe como nosso produto, que no nosso caso será um aplicativo, deverá se comportar. Fugiremos de detalhes como a arquitetura e a linguagem. Desenvolvedores podem auxiliar no processo, mas não serão os escritores.

Retomando o nosso aplicativo em desenvolvimento, o Sales Handy, a Starweb deverá ser responsável por ele. A Fast Sale será a empresa contratante. Teremos três tipos de usuários para o aplicativo, o vendedor, o analista e o CEO.

Geralmente, começa-se mapeando os potenciais usuários que vão interagir com o aplicativo. Depois desse mapeamento, é recomendável eleger um usuário principal para começar o Workshop de escrita de histórias.

No nosso caso, o principal usuário será o vendedor, pois ele sempre atualizará o aplicativo. É importante notar que há uma quebra de paradigmas nas histórias de usuários, pois antigamente o foco não era o usuário, mas sim o sistema. A forma de escrever o requerimento costumava se basear no que o sistema deve fazer. Agora, o que está no centro são as demandas do consumidor, o que modifica a forma de desenvolver um software.

Para ilustrar, poderemos pensar que um bom Product Owner deve saber fatiar o bolo. Fatiar o bolo será saber escrever uma história que permeie todas as camadas da arquitetura do software, porque assim ela será construída de forma que o cliente perceba qual o valor do uso do produto, além de poder mapear os riscos desde o começo.

Um bom Product Owner, portanto, deve saber fatiar, descartar e priorizar. Exemplificando, veremos o exemplo de uma história que representará uma fatia muito grande:

"Um vendedor pode adicionar todos os detalhes de uma oportunidade de venda em um aplicativo."

Outro exemplo, tratando de como o aplicativo funciona internamente:

"Os dados de venda são transmitidos do aplicativo a um banco de dados que os processa, gerando insights para oportunidades futuras.">

Com base nessas duas histórias, temos praticamente a história do nosso produto. Mas não é possível estimá-las por serem excessivamente amplas.

O papel do Product Owner, então, é saber pegar uma história como essas e conseguir fatiá-las em histórias menores, que sejam estimáveis e cabem dentro de uma Sprint.

Praticando como fatiar, pegaremos só um pedaço do bolo, ou seja, da história:

"Como um **vendedor** eu gostaria de adicionar **apenas os detalhes básicos das minhas oportunidades de vendas como valor, nome do cliente, produto e mês.** Então, **eu teria o\* total das oportunidades de vendas no mês corrente\* para a consulta.**"

Pegando agora uma fatia maior, poderemos escrever a seguinte história:

"Como um **vendedor**, eu gostaria de adicionar **todos os detalhes das minhas oportunidades de vendas, como o meu diretor gostaria de ver**. Então, \*\*eu teria todos os detalhes das vendas da minha região para fácil consulta."

Assim, teremos pego uma grande fatia e escrito uma épica, que é como uma grande história com outras histórias menores envolvidas. Essa história não tem um tamanho adequado para ser estimada, porém, ela será fatiada mais adiante. Essa será a lógica. primeiro, pegar uma fatia pequena para ser estimada agora, e mais para frente conseguirmos dividir a maior parte, ainda que já tenhamos escrito, a fim de reservar um lugar para essas necessidades.

Uma dica importante para escrever boas histórias é que elas devem estar em primeira pessoa, assim como a última história que vimos foi escrita. O vendedor deverá relatar o que que funcionalidade ele quer que o produto tenha e o que essa funcionalidade trará de valor para ele como usuário.

A receita será semelhante à frase:

"Como um (papel de usuário) eu gostaria de (funcionalidade). Então, eu (valor)."

Analisaremos mais um exemplo:

"Como um **vendedor**, eu gostaria que **o aplicativo salvasse as oportunidades que eu insiro a cada dois minutos**. Então,\*\* eu não perderia os dados caso esquecesse de salvá-las\*\*.

Parece simples escrever uma história seguindo essa fórmula, mas ainda será necessário praticar para que não peguemos uma fatia muito grande, que não caiba na iteração.

É claro que o tamanho da história dependerá da iteração com que o time trabalha. Geralmente as interações são de no máximo até 30 dias. Isso dependerá da realidade do projeto.

Outra dica é que devemos escrever histórias fechadas, que permitem uma sensação de completude. As histórias possuem camadas, e quando uma história é completa, deverá ser capaz de ser completa em todas essas camadas.

"Como um **vendedor** eu gostaria **de exportar meu relatório de oportunidades em CSV.** Então, **eu poderia ter a visão de quantas oportunidades eu tenho no sistema para este mês.**"

A história acima transmite uma sensação de que o objetivo do vendedor é completar uma etapa, o que é útil para todos que trabalham no time. As histórias fechadas, portanto, passam a impressão de que há começo, meio e fim para uma tarefa e a possibilidade dela ser completada.

Então, o tamanho da história é importante, assim como que ela seja fechada.

Devem ser escritas restrições em cartões. Ás vezes, o que será escrito nos cartões, em vez de histórias serão as restrições que o sistema apresentará. Se tratando de nosso projeto, um dos pedidos da Fast Sale para os próximos releases é que o sistema deve suportar um pouco de 1000 acessos simultâneos.

Isso poderá ser escrito num Story card para que seja priorizado, pois serve ao menos como um lembrete da necessidade de fazer testes de pico no aplicativo. Assim, apesar de não ser uma história, esse cartão guardará o registro de uma informação importante para não ser esquecida.

Também é importante já escrever lembretes para testes atrás dos próprios cartões de histórias, porque isso ajudará na elaboração progressiva desses testes. Como eles serão automatizados, o sistema ficará mais robusto.

Entendemos que existem bons caminhos para escrever histórias bem elaboradas, mas só será possível se tornar um bom escritor através da prática.

## Boas e Péssimas Histórias

Boas vindas. Nessa aula do nosso curso de Agile na prática aprenderemos o que tornam histórias de usuários boas ou más.

Começaremos entendendo como escrever as boas histórias. Há uma sigla em Inglês, a INVEST, comum no meio das práticas ágeis, e ela servirá como um teste, assim como os testes de software, que testará a qualidade de nossas histórias.

A primeira letra da sigla será o I de Independent, ou independente em Português. É importante testar se uma história é independente, porque caso ela dependa de uma outra será difícil para o Product Owner priorizar só uma delas.

A próxima negra significará Negoatiable, ou negociável, traduzindo. Isso indicará que a história escrita no Story card deve ser como um lembrete para conversar com o Product Owner e o time de desenvolvimento. Então, a história será negociável, e não será cobrada como se fosse um contrato. Pelo diálogo se esclarecerá o que realmente a história quer dizer.

O V será de Valuable, ou valioso. A história, então, deve ter valor tanto para o usuário (que no caso da Fast Sale é o vendedor) quanto para quem compra o produto (a empresa que emprega o vendedor). Portanto, esse valor da história não dirá respeito ao desenvolvedor, porque colocar na história que o código do produto em desenvolvimento será em Java, por exemplo, não representará uma informação relevante para o usuário ou para o comprador.

O E será de Estimable, estimável. Toda história deve ser estimável de alguma forma.

As razões pelas quais as histórias podem não ser estimáveis normalmente são três. A primeira, é por falta de domínio em relação ao negócio, o que exige que conversemos melhor com o Product Owner. A segunda, por não haver muito conhecimento da tecnologia, ou seja, haver falhas técnicas de programação, no caso de um aplicativo ou software. Por fim, o último motivo é a história ser grande demais.

Isso nos levará para a próxima letra, o S de Small. Nossas histórias, então, deverão ser pequenas o suficiente para caber numa iteração. O que pode prejudicar escrever histórias pequenas é a história ser composta, com narrativas menores relacionadas a ela. É importante sabermos cortar o que não for realmente necessário no momento, de forma parecida com pegar um pedaço da torta deixando o resto para depois.

Outro fator que compromete o tamanho das histórias é a complexidade. Às vezes, quando há uma nova tecnologia envolvida ou a implementação de uma expansão, é necessário fazer um teste que chamamos de Spike para saber se o desenvolvimento funcionará. Separamos a funcionalidade desse teste de risco. Nesse caso,por exemplo, separaríamos as histórias, havendo uma para Spikes e uma para a funcionalidade. O mais importante é conseguir fazer com que a história caiba na iteração.

O último T será de Testable. A história precisará ser passível de ser testada. Falamos, é claro, da grande maioria das histórias em que os testes podem ser automatizados, mesmo que existam exceções. É fundamental poder sempre agilizar o processo de testar, e isso é senso comum para desenvolvedores de software.

Devemos, então, guardar a palavra INVEST para sempre realizar esse teste de qualidade nas histórias escritas.

Agora, veremos primeiramente um exemplo de história ruim:

"O banco de dados será uma nuvem híbrida para armazenar os dados do usuário."

Essa é uma má história a princípio porque a nuvem híbrida é uma informação do interesse de quem trabalhará com o banco de dados.

Portanto, essa história não tem utilidade para o usuário (vendedor) nem para o comprador. Quando analisarmos a história que escrevemos, podemos começar avaliando de que forma e para quem ela é valiosa, e ela não será uma boa história se não trouxer valor para o usuário e para o comprador.

Mais uma questão é que essa é uma história que não está escrita em 1a pessoa, então não funcionará como um relato pessoal.

Veremos mais um exemplo de história:

"O release do aplicativo estará pronto em 3 meses."

É um problema que uma história traga prazos, pois nas práticas ágeis já temos outros modos para estimar prazos.

"Gráficos e imagens virão de um banco de dados externos."

Essa também não será uma boa história, pois não adiciona valor para o vendedor ou comprador.

Uma boa dica para quando analisarmos histórias escritas e escrevemos novas histórias é que tenhamos a palavra INVEST anotada, bem como lembremos das palavras para testar se a história é boa. Assim, existirá a garantia de que estaremos seguindo o caminho certo.

Outra dica é para deixar a interface do usuário para ser tratada no final, pois pode acontecer de escrevermos uma história que já presuma uma solução de interface que poderá não ser definitiva.

Por exemplo, se escrevermos "O vendedor selecionará o estado num filtro que estará disponível no aplicativo". Essa história presume que existirá um filtro na interface do usuário. No caso de um aplicativo novo, isso adianta etapas que podem estar em desenvolvimento. Então, é melhor escrever histórias que envolvam o design apenas posteriormente.

Também devemos tirar vantagens das diversas iterações ao escrever uma história. Caso seja necessário, devemos fatiar uma história muito grande em histórias menores. Por vezes, a fatia ainda estará grande, e as informações dela servirão ainda para escrever outras histórias menores.

Não precisamos nos preocupar em escrever a próxima história já do tamanho apropriado para a própria iteração. Deixaremos a a fatia grande, então, pois o feedback das demonstrações mudará o que o cliente precisa. Logo, bastará pensarmos em histórias de tamanho aproximado apenas para o momento presente, para não gastar tempo tentando prever o futuro.

Não devemos enumerar cartões de histórias, porque isso renderá conversas tratando os cartões por esses números, o que pode causar muita confusão. É melhor conversarmos sobre o que está escrito nos cartões, como os lembretes, em vez de conversar sobre os números.

Por fim, nem tudo é passível de virar uma história. As histórias não resolverão todos os problemas do que será necessário documentar e se lembrar durante o desenvolvimento. Principalmente quando trabalhamos com fornecedores externos, pode acabar sendo mesmo necessário escrever um documento. Quanto à Interface, por exemplo pode ser necessário haver várias screenshots demonstrando como ela será.

Então, a história de usuário serve muito bem para muitas coisas, não para todas. Escreveremos histórias quando for apropriado.E só a prática permite uma melhora da escrita.

## Jogando Pôquer Estimando em Pontos

Usaremos o exemplo do GPS. O GPS nos diz em quantos metros temos que virar à esquerda ou a direita quando estamos dirigindo. Pode ser mais difícil para muitos conseguir deduzir quanto é essa distância em metros, em vez de entender se ele dissesse algo como "Vire a terceira rua à direita".

Os pontos fazem um trabalho parecido no sentido de que são unidades de medida relativas. Uma quadra é sempre uma quadra, independentemente das variações de tamanho.É possível haver uma quadra de 200 e uma de 300 m. Portanto, essa é uma medida relativa.

Se tratando de desenvolvimento de software, no começo não saberemos muitos detalhes quanto às histórias dele, e por isso tentaremos estimar o esforço que despenderemos para desenvolver essas histórias. Então, teremos uma sequência de números a ser usados na estimativa.

Essa sequência de números encontrada nos baralhos de Planning Pôquer ("1, 2, 3, 5, 8, 13, 20, 40, 100") é baseada na sequência de Fibonacci. Funciona da seguinte forma: começa-se com 1. 1+1=2. 2+1= 3. 3+2=5. 5+3=8, 8+5=13, e assim por diante. Porém, do 20 em diante arredonda-se o resultado dessa soma para mais. A sequência é valorosa porque se uma história vale 1, ela é simples. Uma história que vale 2, portanto, deve valer o dobro da primeira.

Esse será um valor relativo, não absoluto. Ele não se baseia em horas, por exemplo.

Os pontos são All-Inclusive, pois quando estimamos uma história em pontos. precisamos lembrar que não há apenas o esforço em desenvolvê-la, como também o risco envolvido tanto na arquitetura quanto na tecnologia dela, a refatoração da história, o teste. Então, todas essas questões deverão contar na estimativa.

Jogamos Planning Pôquer porque é rápido e divertido. Jogar permite que o grupo interaja e estime um grande número de histórias de maneira rápida. Essa velocidade também depende de termos antes escrito boas histórias.

Todos terão um baralho com o valor 1, ou A, 2, 3, 5 e 8. Esse baralho pode ser comprado ou produzido pelas próprias pessoas da equipe. Cada um tendo um baralho, haverá confidencialidade.

O Planning Pôquer funcionará de forma que todos terão suas cartas. Uma pessoa lê uma história. Antes de começarmos, haverá uma base para comparar os pontos. Para isso, há duas maneiras. A primeira é pegar uma história considerada simples para todo o grupo e definir que ela valerá 1. A partir de então, todas as outras serão estimadas em comparação a essa.

Uma outra maneira é colocar uma história mediana como valendo 5. Então, as histórias mais simples valerão menos (3, 2 ou 1) e as mais complexas, mais (20, 40 ou 100).

É mais recomendável começar pegando uma história valendo 1, simplificando depois o trabalho de estimar. Mas a melhor maneira de começar dependerá de como a equipe se sentir confortável.

É claro que não devemos olhar o que o outro pretende colocar, "colar" de que forma pretende estimar. Uma pessoa lerá uma história em voz alta, então:

"Como **vendedor**, eu gostaria que **o aplicativo salvasse as oportunidades que insiro a cada dois minutos**, então **eu não perderia os dados caso esquecesse de salvá-las**."

Depois da leitura, todos jogam, Vamos supôr que três pessoas jogaram a carta indicando que consideram essa uma história de complexidade 3 e uma, considerou de complexidade 5.

A história, então, é estimada como 5 por regra, pois o número 5 vem logo depois do 3. Quando dois números estimados são próximos, deverá prevalecer o maior.

A próxima história, será a seguinte:

"Como um\* vendedor***, eu gostaria de adicionar*** \*apenas os detalhes básicos das minhas oportunidades de vendas como valor, nome do cliente, produto e mês.\* Então,\*\* eu teria o \*total das oportunidades de vendas do mês corrente para fácil consulta.\*\*

Jogando novamente, duas pessoas estimaram como 3, uma como 5 e uma como 8. Há um gap entre 3 e 8, então, será necessária uma conversa, pois como já foi dito, o mais importante sobre o Story card é proporcionar a conversa, mais do que o que está escrito ser adotado como uma regra. Deverá ocorrer um entendimento comum entre a equipe sobre o que o desenvolvimento dessa história envolverá.

E é importante que o tempo dessa conversa seja monitorado, para não demorar muito nessa etapa. É recomendável ter uma ampulheta ou timer, que pode ser do próprio celular de alguém do time para o monitoramento. Geralmente, estipula-se 2 minutos. Ao final desse tempo, os membros apontam os dedões para cima para indicar se precisarão continuar a conversa, e pra baixo se já conversaram o bastante. Se a conversa não se esgotar naquele momento, é preferível deixá-la para um outro momento e continuar estimando.

Supondo que essa conversa tenha sido suficiente, joga-se de novo. Então, três pessoas consideram como 3 e uma, como 5. O valor da história, portanto, passará a ser 5.

A próxima história, será:

"Como um\* vendedor\*\* eu gostaria de adicionar **todos os detalhes\*\*** das minhas \*oportunidades de vendas como o meu diretor gostaria de ver.\* Então, \*\*\*eu teria todos os detalhes das vendas da minha região para fácil consulta\*\*."

Jogando, todos lançarão o 100. Isso porque é uma história muito grande e precisa ser melhor fatiada, conforme o valor alto indica. Então 100 será o número para a história e seguiremos estimando.

A quarta história, será:

"Como um vendedor, eu gostaria de exportar meu relatório de oportunidades em CSV. Então eu poderia ter a visão de quantas oportunidades tenho no sistema para esse mês."

Jogando, três pessoas estimaram como 3. Mas para um desenvolvedor, pode não estar claro o que o cliente deseja de acordo com a história, então lança uma carta com um ponto de interrogação. Talvez, quem esteja desenvolvendo o projeto não tenha entendido a tenologia envolvida, ou qual é a área do negócio. Essa falta de compreensão ocorre em especial quando uma história está mal escrita, apesar de não ser esse o caso.

Então, deve ser discutido durante um tempo para que tudo seja entendido. Alguns baralhos do Planning Pôquer também podem ter uma carta com uma xícara de café, indicando que uma pessoa está cansada e pedindo um break.

Depois da conversa, então, esperamos que chegue-se num entendimento. Digamos que a partir de agora não só a pessoa entendeu, como o time todo passa a acreditar que a história era mais simples do que haviam imaginado, com as quatro pessoas da equipe estimando como 2.

Quando temos um problema de defeito no código, como pode acontecer, poderemos inclui-lo num cartão para ser priorizados, pedindo por ajustes (bug fix). Os defeitos muitas vezes são pequenos. Caso a equipe tenha um Story card para ajustar defeitos, todas as tarefas relacionadas a esse tema devem ser grampeadas juntamente com esse Story card e estimadas para que os defeitos não se acumulem. O melhor é estimar com pontos baixos, e colocar isso no Backlog. Todos os bug fixes, então, vão incluídos num Story card, o que também é um esforço feito pelo time.

Com o passar do tempo, pode haver uma inflação dos pontos, principalmente se o time estiver com problemas para entregar histórias ao fim das iterações. É recomendável ter um baralho e as histórias pregadas embaixo, para ficar claro se as histórias estimadas de modo semelhante realmente se assemelham no conteúdo. É possível comparar, inclusive, de iteração para iteração, se uma história estimada como 5 numa iteração passada, na atual permanece com esse valor. Caso contrário, pode ficar difícil comparar as métricas.

# Métricas

## Métricas para que te quero

Parece simples que precisemos medir o que fazemos para saber se estamos na direção certa e nos aproximando do planejado. Saberemos, agora, um pouco sobre o conceito de Information Radiator.

Information Radiator é um termo cunhado por Alistair Cockburn, cientista da computação e um dos pioneiros no Agile. Se refere aos grandes charts visuais que geralmente são fixados nos corredores e paredes do ambiente em que a equipe trabalha. Chama-se Radiator porque não há como ignorar essas métricas, mesmo que vê-las não seja uma vontade pessoal, porque estarão presentes no local. Uma segunda razão para o nome é que vale a pena olhá-los, pois são atualizados frequentemente.

O Infomation Radiator é válido porque em vez do time precisar parar uma reunião para mostrar o status do projeto, qual o objetivo pretendido, qual é o problema, etc., essas informações já ficam disponíveis para quem está fora do projeto. Geralmente, quem está dentro do projeto sabe o que se passa, mas quem está fora apenas especula, e pode até parar o trabalho do time para saber.

Existe o conceito do Information Radiator, que significa quando há uma planilha de Excel super complexa, por vezes está dentro de um diretório pouco acessível, podendo até ser acessível a uma única pessoa, que precisa dedicar todo seu tempo a atualizar esse documento. O Radiator, pelo contrário,é feito para economizar tempo de trabalho e permitir que todos tenham acesso às informações sobre o desenvolvimento do projeto, priorizando a transparência, que é um valor do Agile. Então, é importante desenvolver essa prática e ter vários Information Radiators no time.

Para fazer bons Information Radiators, teremos o conceito de "low tech, high touch", que significa baixa tecnologia, mas grande toque, em tradução livre. Ou seja, comece em papel. Apesar de hoje em dia termos todos os tipos de softwares para fazer todos os tipos de softwares, sempre existe uma barreira tecnológica para aprender a utilizá-los e fazer as devidas atualizações. E sempre alguém da equipe terá mais ou menos dificuldade para esse aprendizado.

Usando papel, todos saberão escrever, e será mais rápido e simples, pois não haverá a questão da tecnologia funcionar como uma barreira.Além de também ser tangível, assim como a Vision Box. Se estamos começando nas práticas ágeis, iniciar usando papel é o mais recomendável, e é uma prática utilizada até hoje pelos grandes desenvolvedores de software.

Independente do projeto que estivermos desenvolvendo, sempre surgirá a pergunta sobre quando afinal de contas ele ficará pronto. Para que possamos respondê-la, existe a Velocidade, que mede a capacidade de trabalho da equipe por iteração, tempo. Há times que medem isso em balas de goma, ou em horas ideais. Nesse curso, estaremos tratando de Story Points.

Quando fazemos uma estimativa em Planning Pôquer, prevemos, por exemplo, que numa release (soma de iterações) teremos que entregar 300 pontos. A partir da Velocidade, que é o quanto entregamos por iteração, e tendo cada iteração duas semanas, podemos calcular em quantas semanas conseguiremos entregar a primeira release. Isso nos ajuda a responder a pergunta sobre quando o projeto ficará pronto.

A partir da Velocidade, conseguiremos estimar um prazo no calendário para informar ao cliente quando será possível que o projeto fique pronto. Desenvolvendo os gráficos, fica mais simples entender como a Velocidade auxiliará nessa resposta.

Responda rápido: Qual é a **velocidade** desta equipe?

| **História** | **Pontos por História** | **Status** |
| --- | --- | --- |
| História 1 | 4 | Pronta |
| História 2 | 3 | Pronta |
| História 3 | 5 | Pronta |
| História 4 | 3 | Incompleta |
| História 5 | 2 | Pronta |
| História 6 | 4 | Não começada |
| História 7 | 2 | Pronta |
|  | 23 |  |

A velocidade é 16. Da história 1 até a 3 todas as histórias estão prontas, e contarão no cálculo de velocidade. A história 4, no entanto, não contará, porque está incompleta. As histórias incompletas não contam no cálculo de velocidade ainda que falte muito pouco para completá-las. Isso porque o cliente não poderá utilizá-la, além da dificuldade de calcular apenas uma porcentagem de uma história.Seria muito mais difícil lidar com a métrica, quanto mais tendo em vista que só usamos valores comparativos e não absolutos.

A história 5 também conta, pois está pronta. A 6 não conta porque nem foi começada. a 7 estará pronta. Portanto a velocidade será 16, fazendo a soma dos pontos por história.

É fundamental lembrarmos da regra de não contar com as histórias incompletas, e impedir que qualquer um force que elas sejam contabilizadas, quando trabalharmos com Velocidade.

Quando jogamos Planning Pôquer, pode acontecer de estimarmos uma história como sendo 3, por exemplo, e depois descobrirmos que aquele 3 era um 8 no momento de desenvolver a história. Porém, não poderemos ajustá-la na hora do cálculo da velocidade. Essa história deverá ser contabilizada como um 3, caso contrário, a métrica será alterada. O que se pode fazer é aguardar o próximo Planning para estimar corretamente.

Não existirá ajuste de velocidade. Uma vez que uma história foi estimada de determinada maneira no Planning, esse continuará sendo o valor atribuído a ela até estimarmos novamente

## Burnup na Iteração

É necessário sabermos fazê-los porque os gráficos Burnup e Burndown são muito utilizados em Agile. Como agilista, é fundamental identificar se um gráfico é um Burnup só de olhar, e saber construí-lo.

Veremos, então, como se constrói um gráfico. O primeiro passo será estabelecer o alvo, ou seja, geralmente um time estabelece o quanto acha que entregará numa iteração ou Sprint. Consideremos que serão 30 pontos. Na linha horizontal, colocaremos o número de dias. A iteração, nesse caso, terá 7 dias.

É recomendável desenhar o gráfico no Flip chart, em modo paisagem em vez de retrato, para conseguir completá-lo no formato esperado. Desenhando os pontos pontilhados com o lápis, conseguiremos fazer o gráfico ficar proporcional.

O alvo deverá ser de uma cor diferente. No caso de nosso exemplo, o alvo será de 30 pontos em 7 dias, como foi dito. O eixo y do gráfico, vertical, conterá os Pontos. O eixo x, horizontal, os 7 dias da semana

Conforme os dias forem passando e as histórias forem ficando completas, qualquer um do time poderá ir preenchendo o gráfico, não necessariamente o Scrum Master. É interessante combinar que essa atualização seja feita sempre no mesmo horário.

Agora, vamos pensar que no 1° dia o time completou um pouco menos de 5 Pontos. Então, será desenhada uma linha no gráfico que parta do 0 e suba, parando um pouco abaixo do 5 no eixo y do gráfico. No 2° dia, serão quase 10 pontos. Ao 3°, menos de 5 pontos. O 3° dia, então, terá sido praticamente metade de uma iteração, o que poderá indicar que o time estará com algum impedimento.

No 4° dia, um pouco mais de 15 pontos. No 5°, o time chegará a quase 20 pontos.

A linha do gráfico Burnup, que em tradução livre significa algo como queimar para cima, é ascendente. Depois da atualização do 5° dia já se tornará possível visualizar a olho nu se caso o time continuar entregando pontos nessa velocidade diária, conseguirá alcançar o alvo. Nesse caso, traçando uma linha pontilhada a partir da linha que teremos até o 5° vamos visualizar que chegaremos ao alvo no 7° dia.

Essa é a grande vantagem desse tipo de gráfico. Como ele é simples e ficará bastante visível, qualquer pessoa conseguirá prever se o time cumprirá suas metas.

Então, é importante lembrar que no Burnup o alvo é reto, e a importância dele é que conseguiremos prever se alcançaremos o alvo traçando uma linha reta a partir da linha que já foi criada anteriormente, com base nos Pontos que o time tem feito com relação ao tempo.

## Burndown na Release

É importante saber fazer esse gráfico porque ele faz a medição que pode trazer a resposta para a grande pergunta: Quando o projeto finalmente ficará pronto?

A release se refere à entrega derradeira, que geralmente demorará de 3 a 6 meses.

O primeiro passo para a construção dele é sempre colocar o objetivo. No caso do nosso exemplo, estimamos que na primeira release entregaríamos 240 pontos em 8 iterações. Cada iteração será de 2 semanas.

O eixo y do gráfico, horizontal, terá os valores referentes aos pontos estimados (50, 100, 150, 200, 250, 300). O eixo x do gráfico, ou horizontal, trará as 8 iterações.

Então, em 16 semanas entregaremos nossa primeira release, então é possível colocar essa estimativa no calendário para nosso cliente. Essa informação, é claro, não é inflexível, pois podem haver imprevistos durante as 16 semanas. Mas há uma estimativa de quando será entregue o primeiro incremento.

Tendo passada a 1ª iteração, ou seja, as duas primeiras semanas, será possível começar a desenhar o gráfico. Após esse período, abaixaremos os Pontos para 210. Porém, na 2ª iteração, os pontos subirão de 210 para 220. Analisando as Velocidades das iterações, esse não terá sido o problema, mas na 1ª iteração a velocidade foi de 30, e na 2a, de 40.

Então, o que terá acontecido, na realidade, é que o Product Owner adicionará mais histórias, e teremos inclusive uma velocidade maior na 2ª iteração. Com uma maior quantidade de histórias, não será possível baixar nosso gráfico, que nesse caso terá uma linha descendente.

Na 3ª iteração, os pontos baixarão para 190. Já a partir dessa, poderemos traçar uma linha imaginária e perceber que não conseguiremos entregar o resultado esperado em 8 iterações.

Então, esse passará a ser um momento de avisar o Product Owner da questão, para que ela ou ele retire algumas histórias, repensando a priorização do backlog, ou permita haver mais uma iteração, já que de acordo com o gráfico, será possível entregar após 9.

Na 4ª iteração, a velocidade terá passado de 35 para 65, e de 190, passaremos a 125 pontos, entrando novamente no eixo em 8 iterações. No entanto, essa situação será muito pouco provável.

Por vezes ocorre de um time, principalmente os que não tem muita experiência, ao ver o número de histórias aumentar muito drasticamente acabar inflacionando pontos. Em vez da velocidade aumentar, o time estará estimando histórias com números maiores, o que causará uma ilusão de velocidade. Esse é um palpite sobre o que pode ter ocorrido para que o gráfico de apresentasse assim na 4a iteração, e é uma situação que não costuma ser positiva para a equipe.

## Burndown na Iteração

Por ser um dos gráficos mais utilizados em Agile, deveremos saber fazê-lo e reconhecê-lo.

Para a construção do gráfico, no eixo y, vertical, colocaremos os pontos que o time estimou de acordo com as iterações (a partir do 0, 10, 20, 30, 40, 50 e 60). No eixo x do gráfico, horizontal, haverão 8 dias.

O primeiro passo para a criação desse Burndown chart será colocar a linha referente ao planejado. O time acredita que poderá fazer 45 pontos em 7 dias, então desenharemos uma linha descendente na diagonal, passando pelo 45 do eixo y e o 7 do eixo x.

Teremos os Story cards estimados. Somando-os, perceberemos que só há 41 pontos, e não 45 pontos (somando 2+3+8+5+8+5+5+5). Portanto, não poderemos fechar em 45 pontos, trazendo mais pontos em histórias do que podemos fechar com essa velocidade. Pode ocorrer de termos menos pontos, apesar da velocidade um pouco maior.

Então, partiremos nossa reta do 41, e começaremos a documentar o número de histórias que fecharemos por dia. No 1° dia, serão 2 pontos. No 2°, nada será fechado, o que poderá demonstrar algum impedimento, ser uma Red Flag de que precisaremos da ajuda do Scrum Master. No 3° dia, 8 pontos serão queimados.

O objetivo deve ser nos aproximarmos da 1a linha traçada, que demonstra como seria nossa atuação ideal. No 4° dia, teremos queimado 10 pontos, e no 5°, 8, apesar do 2° dia ter sido problemático.

Então, se traçarmos uma linha imaginária a partir desse momento, veremos que será possível entregar o pretendido após 7 dias.

O Burndown, portanto, assim como o Burnup, permite conferirmos se estamos no caminho certo antes que chegue o momento futuro em que os prazos terminarão.

Agora já sabemos como construir ambos os gráficos, mas é mais recomendável começar pelo Burndown, que é um dos mais utilizados. Não precisaremos abraçar todas as métricas, bastará começar a partir de uma e utilizá-la atentamente.

Conforme aprendemos a usar uma métrica com qualidade, poderemos partir para um outro método. Velocidade e gráficos são os mais indicados para começar, antes de utilizar outros tipos de métricas mais complexos. E, relembrando, usaremos papel e caneta.

## Análise avançada

Precisamos saber Análise Avançada em Métrica porque às vezes uma métrica só não responde todas as nossas perguntas. Poderá ser interessante comparar uma métrica com a outra para entender o que aconteceu.

Pensemos num gráfico Burnup de release, com uma entrega grande. No eixo y, vertical, teremos os números de Pontos (10, 100, 150, 200, 250, 300, 400). No eixo x, horizontal, teremos 8 iterações.

A release começará com 300 pontos nas duas primeiras iterações. Depois o Product Owner aumentará para 350 pontos na 3ª. tendo permanecido em 350 na 4ª e na 5ª . Na 6ª iteração, chegaremos a 400 pontos, ficando assim até a 8ª . Essa será a linha que diz respeito ao planejamento.

Quando de fato os resultados começarão a ser entregues, na 3ª iteração, quando o Product Owner acrescentará pontos, a inclinação do gráfico do time se alterará.

Na 6ª iteração, se traçarmos uma linha imaginária, veremos que conseguiríamos queimar o combinado inicial, 300 pontos, mas não os 400. Então já na 5ª iteração será possível chamar o Product Owner e avisá-lo de que ou a equipe precisará de mais uma iteração, ou da retirada de 100 pontos.

Então, será muito importante usar a métrica a nosso favor e evitar que não tenhamos tempo para a realização da entrega do produto.

Um novo gráfico que veremos será o de Velocidade versus defeitos por iteração. Ele mostrará se com o aumento da velocidade, haverá impacto no número de defeitos inseridos no código. Os dados desse gráfico serão referentes ao mesmo projeto que o anterior.

Esse gráfico terá colunas cuja altura atingirá a velocidade que atingimos por iteração. Abaixo da coluna, na linha horizontal, estará a informação de quantos defeitos foram adicionados naquela iteração também.

Na 1ª coluna, teremos alcançado a velocidade 45 e inserido 1 defeito no código. Na 2ª iteração, inseriremos 3 defeitos no código à velocidade 55. Na 3ª , inseriremos 6 defeitos com uma velocidade menor. Na 4ª , a velocidade será 75, mas inseriremos 10 defeitos.

Então, a 3ª e a 4ª iteração serão objeto de reflexão. Após o susto de ter inserido 10 defeitos, o que comparativamente é uma quantidade grande, voltaremos a fazer uma velocidade de 40 na 5ª iteração, com 2 defeitos, e na 6ª , 40 de velocidad e 1 defeito.

Isso indicará que a velocidade tende a se estabilizar conforme o passar do tempo, ainda que no começo ela possa ser instável. Ela melhorará no decorrer do projeto, dependendo de aprendermos a estimar melhor no Planning Pôquer, e depois se estabilizará.

Analisando nossos gráficos comparativamente, tanto o Burnup de release quanto o gráfico de colunas, poderemos traçar algumas hipóteses sobre o que terá ocorrido. Uma delas é que conforme os propósitos do time aumentarão, as pessoas poderão se sentir emocionalmente abaladas, achando que não conseguirão entregar tudo para o cliente conforme o planejado, o que tem um impacto moral. Isso poderá causar um aumento no número de defeitos, como ocorrerá na 3ª iteração.

Outra questão a ser levada em conta é que na 3ª iteração também haverá uma redução na velocidade, que poderá se dever ao problema de foco, da mesma forma. Ainda conforme nossa hipótese, na 4ª iteração há possibilidade do time, após essa queda na produtividade, ter querido mostrar mais serviço. Por isso, terá entregues muitos pontos, que não saberemos se foram inflacionados sem conferir no Affinity Mapping, e as histórias terão 10 novos erros dentro do código, o que será mais trabalhoso para solucionar até do que escrever uma nova história.

Um gráfico apontando esse tipo de anomalia torna importante utilizar outras métricas para entender a situação. Uma boa sugestão é que com o tempo, conforme dominemos métricas mais simples como o Burndown, comecemos a monitorar, por exemplo, os defeitos versus a velocidade. Assim, entenderemos se há relação entre a velocidade e os defeitos inseridos no código, o que costuma ser algo bem frequente.

# Conclusão

Para fechar o curso, falaremos sobre o que mais pode nos transformar em experts nas práticas ágeis.

Jeff Sutherland, um dos idealizadores do movimento ágil que fez parte do Agile Manifesto, usa uma importante metáfora que é a seguinte:

Entregar um projeto ágil é como posar um avião-caça.

Segundo ele, quando vamos pousar um avião, temos que ajustá-lo de acordo com o vento, velocidade, etc. Num projeto em Agile, também precisaremos de ajustes de acordo com as iterações para conseguir pousar, ou seja, entregar o projeto.

Então, durante nossa trajetória como agilistas, deveremos sempre estar aprendendo a como melhorar as práticas e ler as métricas. O desenvolvimento será diário, assim como a busca por aprender.

A ideia do curso é que sejamos ótimos pilotos, sabendo não só a teoria, como também praticando frequentemente o aprendizado.