# METODOLOGIA ÁGIL

## ORIGEM

A engenharia de Software surgiu depois das outras **engenharias** já existentes e, por ser uma engenharia, ela herdou o *mindset* das que vieram antes. O método de desenvolvimento de projetos usado, chamado de *Waterfall*, foi herdado de outras engenharias, nas quais cada projeto era dividido em fases que dependiam da fase anterior ser aprovada, tendo um conjunto de requisitos. As fases só avançam, e isso tornava seus clientes satisfeitos. Um exemplo que aplica o método *Waterfall* na engenharia civil é o de construção de imóveis: o processo de construção de um imóvel é dividido em fases, como análise do terreno, fundação e pilares de sustentação. Uma vez que uma fase é aprovada e finalizada ela não pode ser alterada.

Porém, para o desenvolvimento de software, esse método herdado de engenharias não era o ideal, porque a cada tarefa concluída eram validadas novas mudanças a partir de feedbacks, o que fazia com que o fluxo fosse alterado. **Para a engenharia de software o método *Waterfall* já não dava conta das mudanças** que ocorriam no meio do caminho. Então ele teve que passar por uma reestruturação para que, depois dos feedbacks de cada tarefa desenvolvida, o fluxo pudesse ser alterado durante o processo de desenvolvimento do projeto.

## WATERFALL x AGILE

Para utilizar o **Método *Waterfall*** em um desenvolvimento de Software é necessário que os requisitos/documentação sejam aprovados e assinados pelo cliente e responsável do projeto. Esse requisito tem todo o detalhamento do fluxo que deve ser seguido para atingir o seu objetivo final e, uma vez assinado, esse fluxo não muda mais (**imutabilidade**). Quando dividimos o modelo em etapas, a primeira etapa se direciona para a segunda e esta se movimenta para a terceira e assim por diante. Com isso, o impacto de um melhor feedback do projeto diminui, porque além do resultado ser visto só no final do processo, as fases não podem ser retrocedidas, excluindo a possibilidade de alguma alteração. Quando o feedback é tardio perdemos espaço para a concorrência e descobrimos os nossos erros depois de muito tempo. Isso faz com que o tempo de reação para corrigir um problema seja muito lento.

Já no ***Agile***, quebraríamos o projeto em partes menores. Poderíamos começar, por exemplo, pensando no tamanho do lote. Eu defini o meu requisito, mas e se eu mudar de ideia? Em vez de definir tudo de antemão, vamos decidindo as coisas aos poucos. Fazendo isso, não priorizamos as etapas, ou seja, não fazemos a análise completa para, depois que um requisito estiver validado, começarmos outra etapa de desenvolvimento. Em vez disso, faremos um pouco de cada etapa para podermos obter um feedback mais rápido, obtendo retorno do ou da cliente a todo momento. Nesse modelo, entregamos um pedaço pequeno do produto e pedimos a validação. Isso porque, ao planejarmos uma ordem de execução do projeto no seu início, pode ser que ela já não faça mais sentido depois de algum tempo, considerando a volatilidade do mundo e da tecnologia.

São três as principais diferenças entre os métodos ***Waterfall*** e ***Agile***:

* **PRIORIZAÇÃO**: um software tem várias funcionalidades, todas muito interessantes para a pessoa usuária final, e fazemos a análise e desenvolvemos todas elas sem quebrá-las em pedaços menores, não estamos priorizando o que é mais importante. Nesse caso, em vez de terminarmos uma funcionalidade, terminamos uma etapa.

Como podemos corrigir isso? Priorizando as atividades. Suponha que temos quatro clientes pedindo propostas de desenvolvimento de software. Se decidirmos iniciar a escrita de todas as propostas simultaneamente, ao fim do primeiro dia, teremos só 25% do trabalho concluído. Não podemos apresentar só 25% de uma proposta para o ou a cliente, essa pessoa ficará insatisfeita porque precisa do valor inteiro da proposta.

Por outro lado, **se priorizarmos, entregaremos uma proposta mais cedo**, em vez de entregar as quatro de uma só vez. Para priorizar, teremos que analisar qual cliente é mais interessante e qual projeto dará mais retorno, seja ele financeiro, de mídia ou de parceria a longo prazo. Cabe a nós escolher o critério para fazer a priorização.

As **Matrizes de Priorização** têm como objetivo **ajudar na visualização das tarefas e dar clareza para os processos decisórios**, além de ajudar a montar uma cadeia de priorização mais eficiente. Uma delas é a ***Matriz de Eisenhower***, um jeito fácil de definir como priorizar suas tarefas de modo que as mais importantes não sejam deixadas de lado pelas que aparecem de repente ou delegar aquelas que são urgentes. A ideia é que todas as suas tarefas possam ser distribuídas em quadrantes, divididas em dois eixos: ***Importância*** e ***Urgência***. Esses quadrantes recebem um valor de 1 a 4 baseado em sua prioridade.



* **FLUXO DE TAREFAS**:  uma vez feita a lista de prioridades, a maneira como lidamos com ela mudará. Para delinear um fluxo, é necessário que as priorizações estejam estabelecidas. Ou seja, **definir o critério** de uma tarefa, para que mudem de ordem no fluxo. Exemplo: A tarefa "dormir" é mais importante do que a tarefa "cozinhar". Primeiramente, o **projeto e o objetivo final devem ser estabelecidos** para que o problema seja resolvido de maneira rápida e eficaz. Então, **segmente em pequenas metas**. Determine a tarefa que deve ser resolvida com prioridade, tendo um fluxo focado no que mais importa.

1. Defina o projeto e seu objetivo final;
2. Defina o que você quer solucionar com esse projeto;
3. Divida o projeto em metas/tarefas para atingir o objetivo final;
4. Defina o critério de priorização;
5. Estabeleça um fluxo que priorize a resolução desse problema, ou seja, foque no que é mais importante.

* **RÁPIDO FEEDBACK**: o feedback nada mais é do que avaliações e críticas sobre os resultados obtidos, sejam elas positivas ou negativas, colaborando para o crescimento da empresa. Primeiro há a **priorização** das tarefas estabelecidas que vão gerar o objetivo final. Depois de priorizá-las, o **fluxo de trabalho** é definido, seja ele dividido por etapas ou finalidades. Quando é terminada uma finalidade, faz sentido ir atrás do ***Feedback*** de seu cliente. Com essa reação do cliente (feedback) é possível fazer alterações no fluxo ou até eliminar tarefas que não sejam mais necessárias.

**Modelo Cascata**: ele favorece os problemas de comunicação (o famoso telefone sem fio), com necessidade de muitos documentos e protocolos, muitos *hand-offs* entre os times e, com isso, a validação de tudo aquilo que foi construído durante o processo de desenvolvimento de um projeto só acontece no final. Por outro lado, há um **foco muito forte na especialização de pessoas**.

**Modelo ágil**: em uma equipe ágil, todas as pessoas que participarão do desenvolvimento do projeto se juntam em todas as etapas desse processo, sendo apenas uma só equipe multidisciplinar. Ela possui **foco em inspeção e adaptação do produto, transparência e eficiência**.

## O QUE É SER ÁGIL

É "**entregar valor mais rapidamente**", reduzindo os custos de mudança e de incertezas. É analisar um problema e abordá-lo da maneira que dá retorno mais rápido. Os ciclos rápidos nos permitem obter feedback constante. Esses ciclos são chamados de **iterações**, as famosas **sprints do SCRUM**. A cada iteração aprendemos algo novo, pois a pessoa nos dá uma informação para construir um produto melhor.

Isso é ser ágil: adotar uma abordagem para os problemas na qual atacamos e temos foco no que dá mais valor. Depois que resolvemos o problema mais importante, partimos para o próximo. O importante é escutar o cliente o tempo todo, pois informações novas podem chegar e mudar a nossa priorização.

Ser ágil é tratar os problemas com naturalidade e abraçar as mudanças, em vez de resistir a elas. Se um ou uma cliente não está satisfeito, temos que tratar o problema, não importa de quem seja a culpa. Pode ser que o cliente não tivesse todas as informações naquele momento para nos orientar. Por isso, temos que tratar o assunto com naturalidade. Falar é mais fácil que fazer, pois isso exige o envolvimento do cliente também.

Com a adoção de ciclos rápidos, nossos clientes precisam estar cientes de que receberá uma versão nova do produto a cada momento e ela sofrerá mudanças. Ou seja, essas pessoas têm que estar prontas para dar feedback constante, tendo, assim, um envolvimento maior no projeto. É o contrário do que aconteceria no *Waterfall*: o cliente aparece para uma reunião no início do projeto, recebe um documento gigantesco daqui a seis meses e se encontra com a equipe de desenvolvimento só depois de um ano para ver os resultados.

Enfim, ser ágil significa ***entregar valor rapidamente***, ter uma abordagem com foco naquilo que gera mais valor para o cliente e empresa, ou seja, quando algum cliente apresenta um problema, devemos oferecer a ele ***entregas parciais*** de acordo com a necessidade desse cliente. Esses ciclos rápidos de entrega fazem com que os ***fluxos e feedbacks sejam constantes*** e o aprimoramento do serviço/produto seja cada vez melhor, alcançando ***melhores resultados*** no final do processo.

A agilidade não trará mais velocidade logo de imediato, mas sim ao longo do tempo quando o desperdício for eliminado, e a qualidade for aumentada e percebida.

## O MANIFESTO ÁGIL

O Manifesto Ágil escrito em 2001 por pessoas que já tinham um vasto conhecimento na área de desenvolvimento de Software. O objetivo era descobrir quais eram as semelhanças entre os diversos desenvolvimentos de software pelos quais eles haviam passado e que faziam com que os clientes ficassem mais ou menos satisfeitos. A partir desse pensamento, eles chegaram a **4 valores** que foram escritos claramente para qualquer um entender:

O **primeiro valor** diz que as ***interações entre os indivíduos*** são mais importantes que os processos e as ferramentas, ou seja, a maneira como a equipe trabalha impacta diretamente no desenvolvimento do que será produzido.

O **segundo valor** diz que ter um ***software funcionando*** é mais importante do que uma documentação detalhada e compreensiva, ou seja, devemos trabalhar para desenvolver um projeto para ele funcionar.

O **terceiro valor** diz que a ***colaboração do cliente*** é mais importante que a negociação de contrato, ou seja, a proximidade com o cliente é essencial para que o feedback tenha valor e fazer com que isso tenha impacto no fluxo de trabalho.

O **quarto e último valor** diz que ***responder às mudanças*** é mais importante do que seguir o plano, ou seja, a mudança é algo natural e, se batermos de frente com essa situação, teremos um cliente insatisfeito.

## CASES DO MÉTODO ÁGIL

A [Agile Alliance](https://www.agilealliance.org/agilealliancebrazil/" \t "_blank) e a [Scrum Alliance](https://www.scrumalliance.org/) são instituições que promovem eventos com o objetivo de compartilhar experiências e desafios de como foi introduzir os métodos de agilidade dentro de suas empresas, além de promover *networking* com outros empresários e empreendedores. Também conseguimos observar diversos resultados com a implantação de métodos de agilidade dentro de pequenas e grandes empresas a partir de ***cases***.

Ao ler tais *cases*, não “copie e cole” as práticas de agilidade que já foram feitas dentro de outras organizações. Não é porque elas geraram um resultado enorme que você terá o mesmo resultado. **Deve ser entendido o porquê** de determinada empresa ter aplicado certa característica ágil, **o que** ela queria resolver com isso, e analisar vários fatores dentro da empresa antes de implantar a mesma metodologia na sua empresa/equipe.

## GESTÃO ÁGIL

A gestão tradicional pode ter funcionado durante muito tempo, mas hoje começa a dar sinais de que está perdendo espaço, pois está intimamente ligada com um ambiente de baixa autonomia, baixo engajamento e baixa motivação, que possuem impacto direto em uma baixa taxa de sucesso de projetos. Estamos cada vez mais vivendo em um mundo **VUCA**:

* **Volátil**: algo é dado como certo hoje pode não ser mais certo amanhã;
* **Incerto**: temos muitas dúvidas sobre como resolver os problemas;
* **Complexo**: as soluções para os problemas não são lineares;
* **Ambíguo**: soluções que funcionam em um contexto não funcionam em outro similar.

Além disso, vimos que com isso, precisamos reduzir o tempo em que se pega o feedback do cliente. Precisamos desse feedback mais rápido para podermos agir também o mais rápido possível. Em um mundo VUCA, onde as coisas mudam rapidamente, o principal bem é o conhecimento e pra isso, a principal habilidade passa a ser aprender.

**Obs.**: ***Personal Maps*** é uma atividade onde você procura entender melhor como outra pessoa funciona, desenhando em um diagrama as categorias de interesses que essa pessoa possui. É um exercício muito usado para estimular a colaboração entre pessoas de um mesmo time.

Como já dizia Ricardo Semler: *"Medos e inseguranças criam as estruturas das nossas empresas"*. Para contra atacar esse problema, podemos investir nos **3 pilares da confiança**:

* **Dar autonomia**: posso fazer as coisas da maneira que eu acredito ser a melhor;
* **Propósito**: tudo o que eu faço tem um propósito final claro pra mim e que eu concorde;
* **Maestria**: empresa dá espaço pra que eu me torne melhor naquilo que gosto de fazer.

A liderança em um ambiente ágil emerge naturalmente entre as pessoas, de acordo com as habilidades demonstradas para aquela posição e também o momento em questão. Além disso, essa liderança é situacional, ou seja, ela não é definitiva e sim para aquele contexto. E isso pode ir rodando conforme for necessário. **Liderança Situacional** é quando lideranças emergem de acordo com o contexto. Em uma situação uma pessoa pode ser a líder e em outra situação a liderança, dentro de um mesmo time de pessoas, pode ser de outra pessoa. Isso pode variar de acordo com as habilidades da pessoa, o momento do time e vários outros fatores.

Em um nível hierárquico alto as pessoas geralmente possuem um maior conhecimento de todo o contexto e muitas informações, mas elas são mais abstratas. Já em um nível hierárquico mais baixo, há um maior conhecimento dos detalhes, mas a visão é um pouco mais limitada por não conhecer o todo. O segredo para um bom processo de tomada de decisão é justamente **balancear esse conhecimento** e para isso, uma forma de mapear como as decisões e o nível de empoderamento nas organizações, é através do *Delegation* *Poker*. O ***Delegation Poker*** é uma ferramenta do Management 3.0 que nos auxilia a mapear como uma decisão é tomada em determinada circunstância na empresa. São 7 níveis possíveis que pela ordem são:

**1)** Ordem **2)** Venda **3)** Consulta **4)** Consenso **5)** Conselho **6)** Informação **7)** **Delegação completa**

A partir do momento em que conseguimos trabalhar um pouco a delegação, é hora de dar o feedback, porque não adianta delegar uma atividade, uma tarefa para alguém e não comentar se aquilo funcionou bem ou se não funcionou, e o porquê. Um **feedback** precisa ser composto por:

**1)** Contextualização **2)** Exposição de Fatos **3)** Valor da Melhoria **4)** Perguntas Poderosas

A contextualização mostra o cenário para a pessoa, enquanto a exposição deixa claro para a pessoa a situação onde aquilo aconteceu para que ela possa se lembrar do momento. O valor da melhoria deixa claro pra pessoa o que se tem a ganhar corrigindo aquele comportamento. E por fim, as perguntas poderosas geram reflexão na pessoa sobre aquela situação.

**Obs.**: uma boa iniciativa de feedback positivo e reconhecimento é o ***Kudo Box/Cards***. Dê um prazo de uma semana para as pessoas depositarem os reconhecimentos em uma caixa. Depois, abra a caixa e deixe exposto esses reconhecimentos. Repita o processo indefinidamente. Lembre-se sempre de identificar quem está dando e quem está recebendo o feedback.

Feedback gera conhecimento e evolui as pessoas, fazendo com que elas saiam de um estágio de incompetência inconsciente e levando ela a um estágio de consciência, que vai fazê-la chegar na competência. Para isso, ele precisa ser dado com cuidado. Contextualização, fatos, valores e perguntas poderosas precisam sempre ser usadas na hora de dar um feedback para alguém. Esse feedback gera conhecimento para a organização, melhorias.

As melhorias podem ser aplicadas de duas formas diferentes. A primeira é a chamada “***kaikaku***”. São conceitos “kaikaku” e “***kaizen***”, provindos do TPS, Toyota Production System. O ***Kaizen*** envolve mudanças pequenas realizadas de forma incremental ao longo do tempo, enquanto o ***Kaikaku*** significa uma grande mudança feita de uma vez.

A ideia do ***kaizen*** é a melhoria contínua, e para ser contínua, as mudanças tem que ocorrer em doses homeopáticas. O que acontece quando trabalhamos com mudanças e aprimoramentos no nosso trabalho é que precisamos ter um certo ritmo. A intensidade dessa mudança, do trabalho executado precisa ser saudável. Quando não temos melhorias acontecendo no trabalho, o que acontece é que ficamos desmotivados. No ambiente onde tudo continua sempre igual as pessoas ficam desmotivadas. Um ambiente onde tudo muda o tempo todo, e muda bruscamente, é um ambiente de muito estresse. Se oscilamos entre as duas coisas, entre o estresse e o desanimo, também não é bom. Temos que ficar no ponto intermediário, onde existem desafios, melhorias, mas que são alcançáveis. Se conseguimos ficar nesse nível intermediário, conseguimos manter um nível de motivação ideal para que as melhorias aconteçam de forma natural.

## A EMPRESA ÁGIL

Quando falamos de agilidade, queremos uma melhor adaptabilidade às mudanças e não – como interpretações equivocadas que acontecem – tornar um ambiente mais rápido. **Removendo burocracias desnecessárias** nos processos é possível **tornar uma organização mais adaptável a mudanças**, pois quando uma mudança for necessária, ela será menos traumática, pois haverá menor barreira para se mudar. Para isso, faz-se necessário mapear burocracias entre as áreas e removê-las para trazer ganhos ao negócio.

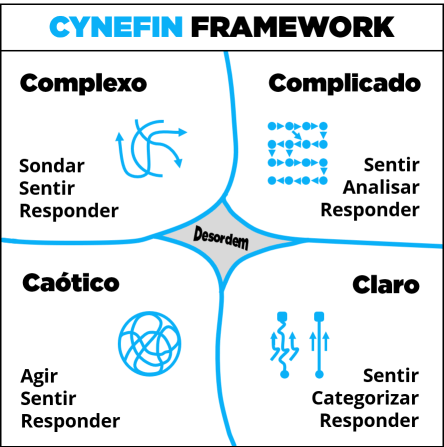
A **entrega contínua de valor** é essencial para a agilidade e está diretamente conectada a três grupos importantes para o sucesso da empresa: **(I)** **valor para cliente**; **(II)** **valor para colaboradores** e colaboradoras; e **(III)** **valor para parceiros** e parceiras da empresa. O **primeiro caso** abrange o aumento do valor dos bens ou dos serviços destinados a clientes, além de reduzir os efeitos adversos ao adquirirem este bem ou serviço. O **segundo** envolve melhorar a satisfação de colaboradores e colaboradoras e reduzir os erros. O **terceiro**, semelhante ao primeiro, está relacionado ao aumento do valor de bens e serviços dessas parcerias, reduzindo proporcionalmente os efeitos adversos.

Adotar *Business Agility* também significa uma mudança de cultura, começando pela **priorização das entregas que realmente têm valor ao(à) cliente final** e entregar este valor continuamente, em pequenos pedaços, evitando feedbacks tardios. Do ponto de vista do gerenciamento, é vital ter **times auto-organizados** e **lideranças que sejam servidoras**. Uma estrutura de comando e controle não se encaixa em *Business Agility*.

## IDENTIFICAR COMPLEXIDADES

Temos algumas formas de identificar a complexidade da sua empresa, entre elas: **(I)** **através do quão perecível é sua informação**, **(II)** **se as informações do negócio são subjetivas ou intelectivas**. Os problemas basicamente se dividem em dois grupos: **Intelectivos** ou **Subjetivos**. Os **problemas subjetivos** são os que possuem mais de uma forma de ser resolvida e só sabemos se a forma empregada foi a mais correta após tentar resolvê-lo. Quanto mais problemas subjetivos temos, mais complexo podemos dizer que é o ambiente que estamos mapeando. Por outro lado, um **problema intelectivo** é aquele em que mapeamos uma única forma possível de resolver e essa forma já está testada e provada. Isso faz com que um ambiente que possua mais processos e problemas intelectivos, seja menos complexo.

Mas, considerando diferentes contextos no ambiente de trabalho, como identificar e lidar com eles? É nesse momento que entra o *framework* ***Cynefin*** — palavra de origem galesa, que significa habitat, ou ambiente, em português —, desenvolvido pelo pesquisador e consultor Dave Snowden, para **auxiliar na identificação de contextos e na tomada de decisões**, de acordo com as diferentes relações entre causa e efeito. A estrutura é composta por quatro quadrantes e um fator central, chamados de domínio. Cada um desses domínios traz formas de lidar com as situações dentro da empresa, conforme o enquadramento do contexto dentro desses domínios. É essencial falar que esse **framework** é **fluído**, ou seja, acompanha as percepções da realidade.



Considerando a identificação de contextos e as respectivas tomadas de decisões, a partir de um problema, no domínio:

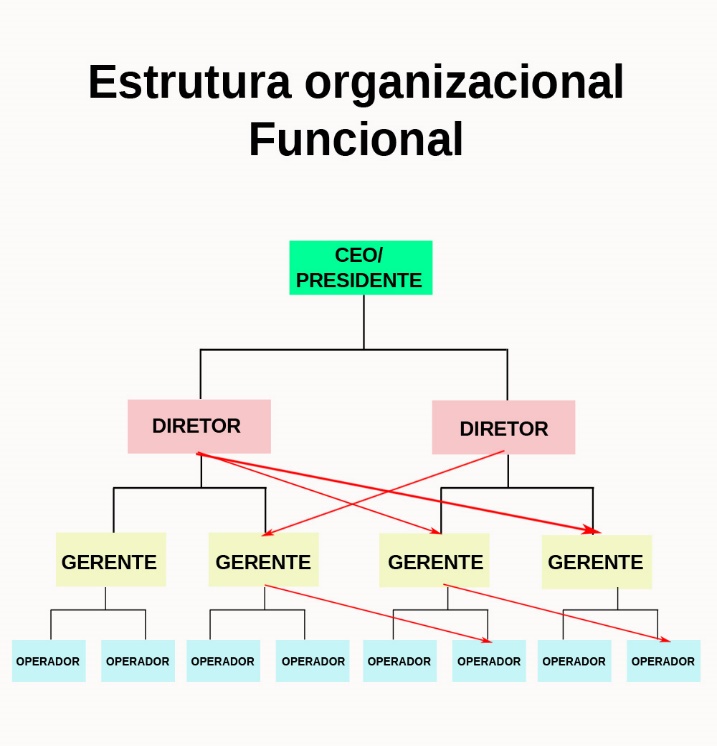
* **Simples**: a **decisão** é **identificada de forma rápida e fácil**, pois a **relação** entre **causa e efeito** é **evidente**. Nesse contexto é possível: sentir; categorizar; responder. As pessoas já sabem quais ações são necessárias no dia a dia, o que e como devem fazer. Sendo assim, não há necessidade de *Business Agility* nesse cenário;
* **Complicado**: as **decisões** são **100% entendidas**, podendo ser **previstas**, com tempo e conhecimento especializado. Sendo assim, neste ambiente recomenda-se: sentir; analisar; responder. A agilidade nesse contexto é útil, pois **ajuda a reduzir riscos**, feitas as análises para obter respostas necessárias. Neste domínio existem diversas possibilidades, diante de um impedimento há múltiplas formas para lidar com ele, então é necessário analisar a situação e a solução que entregue maior valor ao cliente, em um menor tempo e com menos riscos;
* **Complexo**: as **decisões** são **descobertas parcialmente, após conhecer o problema**. **Não é possível prever** uma decisão, como no domínio "Complicado''. Aqui, você não sabe o que não sabe. **É preciso sondar** (cultura de experimentação), **sentir** (aprender) **e responder** (agir em sua melhoria). Uma situação que entra nesse domínio é o estabelecimento de uma nova concorrência, como aconteceu com os taxistas com o surgimento da Uber. É preciso entender se é uma ameaça ou não, aprender com as diferenças positivas e negativas, para responder, se haverá uma adaptação ao novo **modelo posto pela concorrência** e etc. Este é o contexto ideal para *Business Agility*.
* **Caótico**: não há decisões previsíveis, mesmo conhecendo o problema. Caracteriza-se pela imprevisibilidade de causa–efeito ou problema–decisão. **São necessárias decisões rápidas** em situações completamente imprevisíveis, como no Mercado Financeiro, que dificilmente um plano cobrirá os riscos e variáveis que podem afetar esse ambiente. **É preciso agir**, ver o que acontecerá **e responder com base na observação** do que aconteceu, portanto neste contexto a agilidade não ajudará;
* **Desordem**: não se tem compreensão assertiva do contexto, problema e da(s) possível(is) solução(ões). Por essas características, a desordem está localizada no centro da estrutura, entre os quadrantes dos demais domínios.

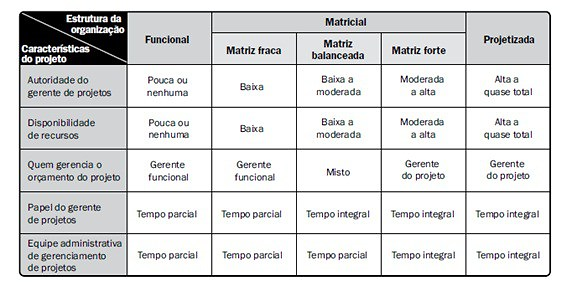
No desenvolvimento de um projeto de software que as decisões e soluções são descobertas à medida que se entrega e obtém feedback, este framework pode ajudar a direcionar os ajustes necessários rumo ao fluxo de trabalho ideal, de acordo com o domínio.

## ESTRUTURA ORGANIZACIONAL

O que vai determinar se uma empresa é ágil ou não é a sua habilidade para reagir a mudanças e para isso, ela **pode ter qualquer estrutura**. Cada uma delas traz vantagens e desvantagens, ou seja, nenhuma é perfeita e possui seus efeitos colaterais. Escolher qual metodologia usar na sua empresa requer um estudo e análise cuidadosa dos impactos nas pessoas e a cultura organizacional.

Exemplos de modelos de estruturas organizacionais: **hierárquica/linear**, **funcional**, **matricial**, **em rede**, **em projetos** etc.



Quebrar as barreiras existentes entre as áreas é fundamental para construir uma empresa mais ágil e que consegue responder melhor às mudanças. No entanto, a forma de fazer essa quebra de silos é uma decisão complexa e que envolve muitos *players* dentro do ambiente corporativo. A cocriação de um processo mais ágil é um passo fundamental para sensibilizar todas as áreas envolvidas e conseguir um "*buy-in*" mais fácil e o comprometimento pode ser criado através de metas compartilhadas e o objetivo final de uma redução de burocracia, com todos os ganhos que isso pode trazer. Estruturas organizacionais mais avançadas, como *squads* ou então uma organização inteira multidisciplinar voltada para a geração de valor são passos que podem ser dados também, mas exigem um bom nível de maturidade e também possuem seus riscos.

O **primeiro passo** possível para reduzir barreiras organizacionais é através da **cocriação de processos entre as áreas que dependem umas das outras**. Para guiar um bom resultado para o negócio e facilitar essa caminhada, é importante simplificar processos e ter uma meta compartilhada entre as áreas, que vai fazer com que o compromisso entre essas áreas fique mais evidente. O foco do *Business Agility* é reduzir burocracias, mas para quebrar silos entre áreas, é preciso que as áreas envolvidas estejam realmente comprometidas. Isso pode ser feito justamente através da simplificação de processos que vão reduzir burocracias e fazendo com que ambas as áreas compartilhem metas de resultados e sejam cobradas por essas metas.

## GERAR VALOR

Uma maneira de mapear dentro da organização quais são as unidades geradoras de valor é **identificar as menores unidades de valor**, ou seja, um grupo mínimo de funções e iniciativas que trabalhando em conjunto conseguem fazer com que um produto ou iniciativa gere um resultado para o negócio. Esse mapeamento é importante para encontrar as possíveis barreiras existentes e mapear quais pontos de integração entre as áreas podem ocasionar gargalos que impedem uma empresa de ser realmente Ágil.

E o que seria essa menor unidade de valor? Um **conjunto de funções e atividades que agrupados** são minimamente **suficientes para que uma determinada operação possa ser realizada** gerando valor para o negócio. Quando encontramos a menor unidade de valor, temos mapeado exatamente tudo o que é necessário para que aquela operação, seja a criação de um produto novo ou de uma nova unidade de negócio, minimamente possa existir e gerar valor para o negócio.

## APRENDENDO A SER ÁGIL

O caminho para uma organização poder ser ágil vai depender muito do seu contexto. Justamente por isso, a primeira coisa que precisa ser feita é pensar o que significa ser ágil para a sua empresa. A partir daí, você vai **problematizar**, ou seja, identificar os problemas que podem ser resolvidos na sua organização, para depois **pesquisar** soluções para esses problemas, **conectando** essas pesquisas com o seu contexto e mundo real. Uma vez conectado com o mundo real, coloque as soluções em **prática**. Valide hipóteses, teste, veja os resultados. Por fim, **compartilhe** esses testes e resultados com outras pessoas da organização. E repita esse processo inteiro de forma indefinida.

## A EQUIPE ÁGIL

Em uma equipe ágil, todos os conhecimentos necessários para que o projeto evolua está contido dentro do time. Isso significa ter um **time multidisciplinar**, que pode envolver, em uma mesma equipe: vendedores, marketing, desenvolvedores, infraestrutura, RH e assim por diante. Claro, cada uma dessas pessoas e áreas vai ter um papel diferente do projeto e no futuro vamos aprender quais papéis podem existir. Mas o ponto principal é: tudo o que o projeto precisa de conhecimento para evoluir, precisa estar contido nessa equipe.

## AUTO ORGANIZAÇÃO

As premissas básicas e essenciais para uma equipe ser auto organizável é ter:

* **Objetivo compartilhado**: defina o objetivo no qual todos da equipe participarão do desenvolvimento do projeto de forma compartilhada;
* ***Timebox***: espaço de tempo necessário para que a auto organização funcione;
* **Restrições**: o que não deve ser feito no desenvolvimento do projeto.

Sabendo para onde o time precisa ir (os objetivos), até quando ele precisa chegar (timebox) e quais os limites que ele possui (restrições) o time pode se auto organizar para conseguir alcançar esse objetivo.

## TIPOS DE EQUIPE ÁGIL

Existem três tipos de equipes que estão presentes nos ambientes ágeis:

**Funcionalidades**: essa equipe se organiza em torno de uma funcionalidade e se preocupa em maximizar o valor na entrega ao usuário para uma melhor experiência possível. A equipe precisa ser multidisciplinar, já que ela precisa trabalhar em diversos pontos no código.

**Componente**: essa equipe funciona muito bem para fazer otimizações locais, porém pode ser crítico mexer em um componente já que ele é usado em diversas funcionalidades, além de garantir a sincronização dos componentes que foram alterados.

**Jornada do cliente**: essa equipe foca em grupos de clientes que possui comportamentos ou necessidades semelhantes.

## O SCRUM TEAM

O *Scrum team* é um conjunto de pessoas que tem o objetivo de fazer produtos excepcionais, e para isso acontecer, são necessárias 3 responsabilidades:

* **Negócio**: responsabilidade em garantir que o produto a ser desenvolvido faça sentido para o mercado. Não adianta você produzir o melhor produto, se não tem cliente para o mesmo.
* **Tecnológica**: responsabilidade em garantir que o produto produzido esteja correto para aquele mercado em específico e público-alvo.
* **Eficiência**: responsabilidade em garantir que o time opere com uma melhor qualidade possível.

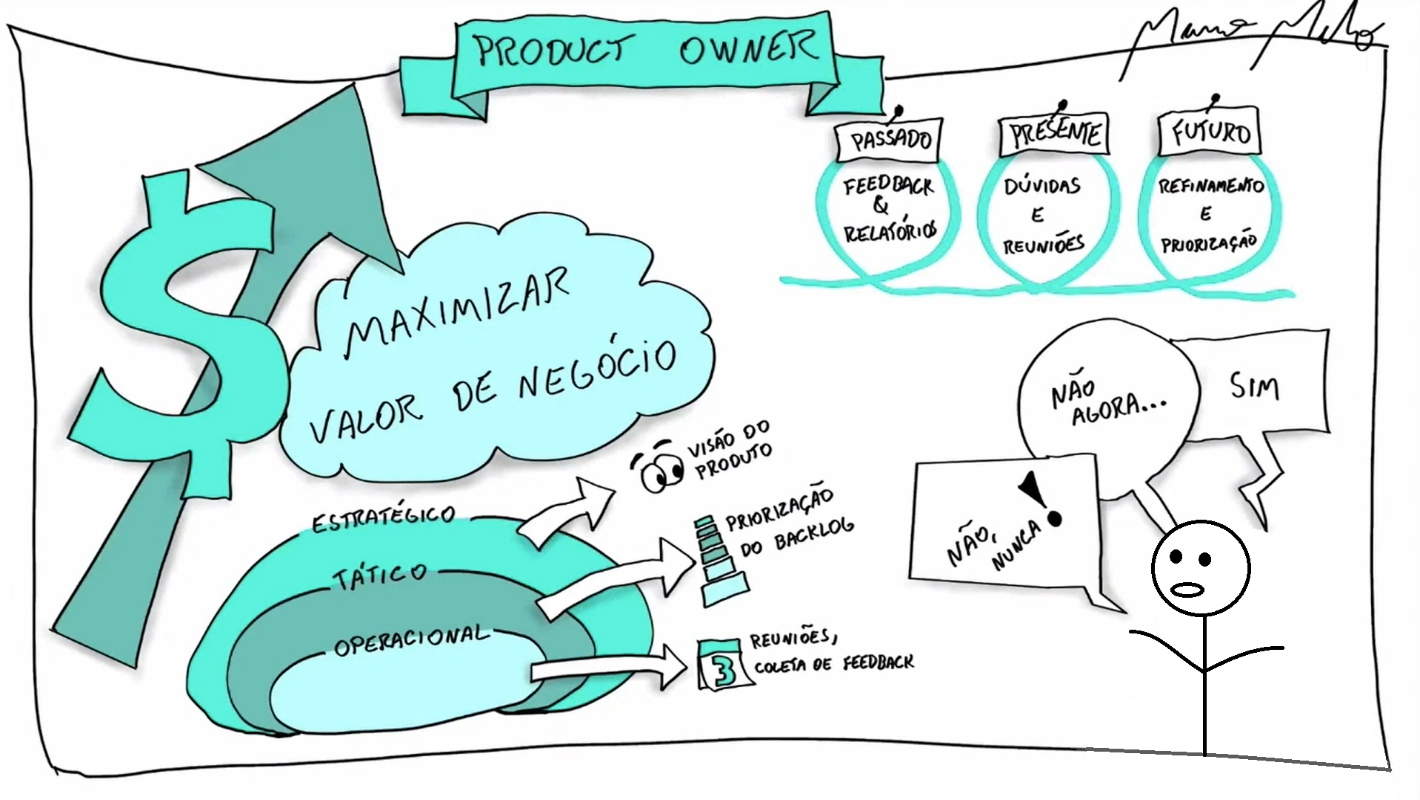
Ou seja, nessa aula aprendemos que um time ágil é organizado de forma que possua pessoas que fiquem responsáveis por garantir que o produto a ser desenvolvido esteja alinhado com o que o mercado espera, que a qualidade e tecnologias empregadas estão sendo as adequadas e também por pessoas responsáveis por garantir que a equipe opere de uma forma eficiente, sem gargalos.

## O PRODUCT OWNER

A pessoa responsável pelo papel de *Product Owner* tem o desafio de conseguir **maximizar o valor** que aquele produto irá gerar para o negócio.

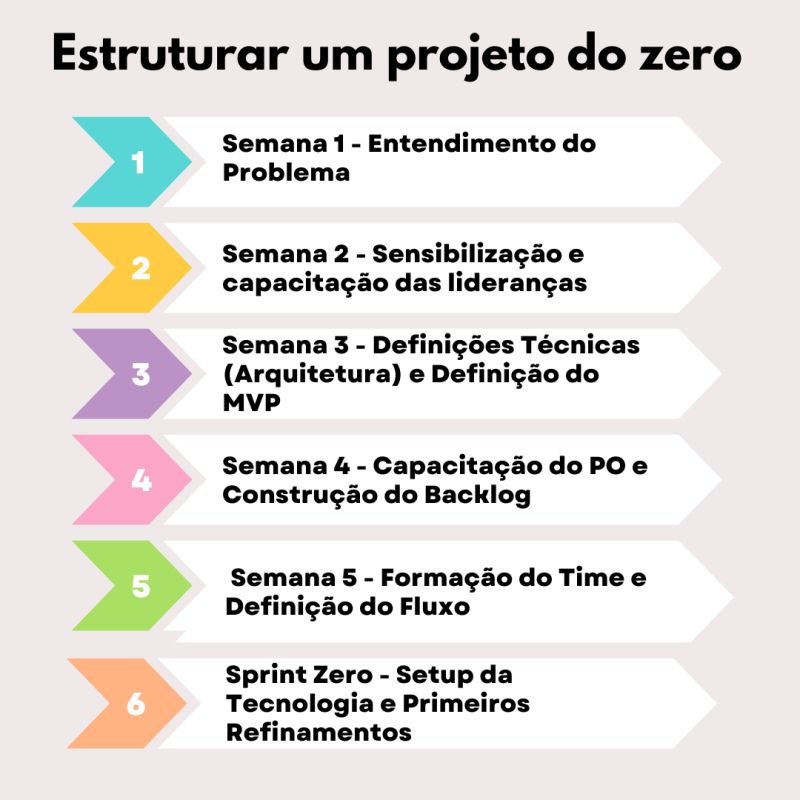
Para isso, o *PO*, além de precisar ter um **foco alto e senso de dono**, precisará **conversar frequentemente com as outras áreas** que fazem parte da empresa, para entender quais são os pontos-chave que remetem a dor dos clientes, e conhecer a fundo o produto e mercado em que a empresa está inserida.

No seu dia a dia, o *PO* terá que **descrever o que precisa ser feito**, e para isso, é necessário conversar com outras áreas para entender as necessidades **e também o que pode ou não ser feito**. Além de **priorizar o que precisa ser construído**, muitas vezes o *PO* terá que **dizer não** para funcionalidades que agregam menos valor para o negócio/projeto naquele momento.



O trabalho do PO é quase que inteiramente composto por **planejamento, comunicação e mais comunicação**. O time precisa ter uma visão clara sobre o que fazer em cada sprint, os stakeholders precisam ter um canal aberto para feedback e entregas e todos devem seguir a mesma visão definida para o produto. Entenda as responsabilidades do PO:

* **Refinamento do Backlog:**com o *input* de Arquitetos, Engenheiros de software e outros stakeholders, o PO tem a responsabilidade de construir, aprimorar e manter o backlog do time. O backlog é geralmente constituído de novas funcionalidades, porém também contém erros (bugs) e melhorias. Ele é priorizado com base em seu valor para os usuários, tempo de desenvolvimento e outras dependências, que podem ser identificadas na reunião de planejamento de cada Sprint;
* **Planejamento de Sprints:** o PO revisa e reprioriza as estórias do backlog como parte do trabalho preparatório da reunião de planejamento de sprint. Isso pode incluir a coordenação e gerenciamento de dependências entre outros times (com outros Product Owners). Durante as reuniões de planejamento de sprints, o PO deve ser a maior fonte de informações sobre o detalhe das estórias e prioridades e tem a responsabilidade de aceitar o planejamento final feito pelo time;



* **Elaboração de estórias no dia-a-dia:** a maioria dos **itens do backlog** são **elaborados e quebrados em estórias para desenvolvimento**. Isso pode acontecer antes da sprint, durante a reunião de planejamento ou durante a própria sprint. Qualquer membro do time pode escrever estórias e critérios de aceitação e o PO tem a responsabilidade de manter o processo fluido. Geralmente é bom ter o equivalente a duas sprints em estórias escritas, pois ajuda a manter o ritmo do time em caso de imprevistos;
* **Auxiliar no BDD** (***Behavior Driven Development*** - Desenvolvimento Guiado por Comportamento)**:**o PO também pode participar no desenvolvimento dos critérios de aceitação para estórias. O BDD é uma abordagemque **visa focar no comportamento de uma determinada parte do sistema** onde se espera atingir um objetivo e geralmente excede os limites da equipe de tecnologia, com uma linguagem de fácil interpretação;
* **Aceitando estórias:** o PO é o único membro do time que pode aceitar estórias como concluídas. Isso inclui a validação de que a estória possui os critérios para aceitação e tem os testes de aceitações persistentes e apropriados. Fazendo isso, o PO realiza sua função de **garantia de qualidade**, focando primariamente em produtos prontos para usar;
* **Entendendo o trabalho de facilitador:** facilitadores são as pessoas que ajudam os implementadores a se concentrar na implementação. Eles se certificam que todos os implementadores estão aptos a trabalhar e com o mesmo objetivo, removendo obstáculos do caminho. O PO é o facilitador do time e deve abraçar essa causa para que o time faça um trabalho bem feito;
* **Participar na retrospectiva e no demo:** como um membro integral do time e o responsável pelos requerimentos, o PO tem um importante papel na reunião de demonstração para revisar e aceitar estórias. Nessa reunião o PO também deve prover feedback para o time e coletar informações para os stakeholders.

E, não se esqueça, procedimentos e processos são ótimos, mas como o próprio Manifesto Agile diz: **indivíduos e interação** sobre processos e ferramentas; **software em funcionamento** sobre documentação abrangente; **colaboração com o cliente** sobre negociação de contratos; **responder a mudanças** sobre seguir um plano.

## O DEVELOPMENT TEAM

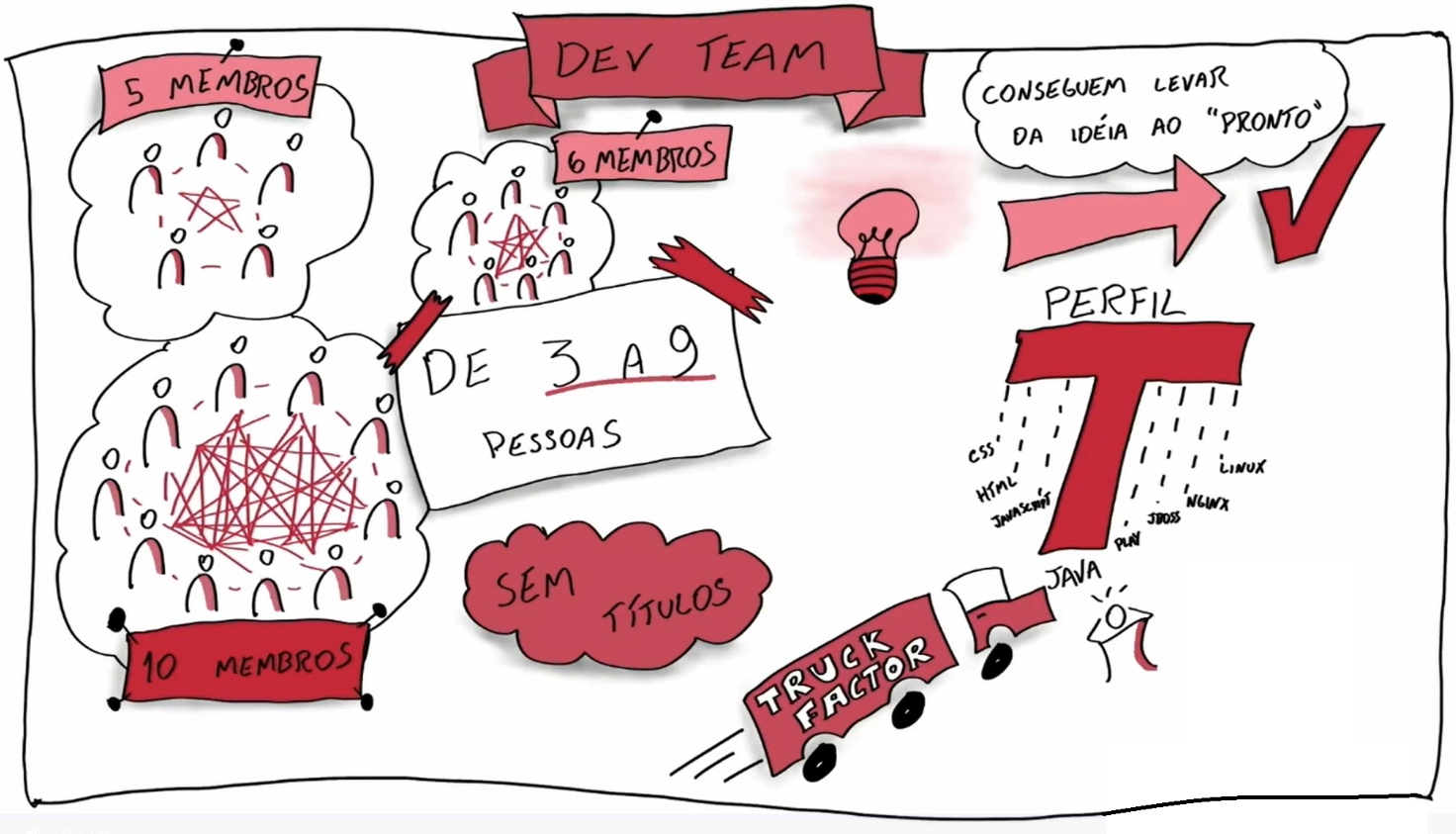
Uma equipe de desenvolvimento é responsável por **construir, validar e integrar o incremento de um produto***.* Além disso, essa equipe é composta por pessoas que não possuem títulos, todos são "membros" daquele time. Uma das características mais interessantes para a pessoa fazer parte de uma *Development Team* ou até mesmo de uma equipe ágil é ela ser ***T-shaped***, ou seja, além dela ter pelo menos uma expertise na qual ela é especialista, ela também consegue ser generalista, podendo atuar com tranquilidade em diferentes papéis e assuntos.

Uma característica muito importante de um *Development Team* é ele **ser capaz de levar a ideia** que teve **ao resultado**, seja um código escrito, um teste etc. Todas essas atividades a equipe precisa conseguir realizar. Ela não pode chegar em determinado ponto e passar para outra equipe. Para conseguir essa equipe, precisamos do perfil T. Mesmo quando falamos de uma equipe multidisciplinar, de pessoas generalistas, uma pessoa vai ter uma habilidade mais desenvolvida. Por exemplo, uma pessoa que conhece mais de Java. Esse código que ela desenvolver vai ser utilizado em vários outros lugares, e cada uma dessas coisas a pessoa conhece menos. A ideia é ter um perfil T para haver sobreposição. Tenho uma pessoa que conhece muito Java, outra que conhece HTML. Tenho ali duas pessoas que podem trabalhar aquilo.

Um *Development Team* é composto por pessoas que podem ser:

* **Especialista**: alguém com profundidade de conhecimento em algum assunto;
* **Generalista**: pessoa que sabe um pouco de cada coisa, mas não tem nenhum conteúdo muito aprofundado;
* **T-shaped**: alguém com especialidade em algo, mas com um conhecimento generalizado em outras coisas;

Um indicador muito usado para determinar se uma equipe tem o perfil T é o ***Truck Factor***. Ele diz quantas pessoas de uma determinada equipe precisam ser atropeladas por um caminhão para que ela trave, ou seja, pare de trabalhar. Quanto menor o número, pior é. A ideia de ter esse perfil T é que as pessoas consigam cobrir umas às outras e fomentar a evolução dessas pessoas em outras especialidades. Ter alguém para ensinar Java Script para alguém que não é tão bom e conseguir aos poucos tornar-se melhor em todas as habilidades existentes.

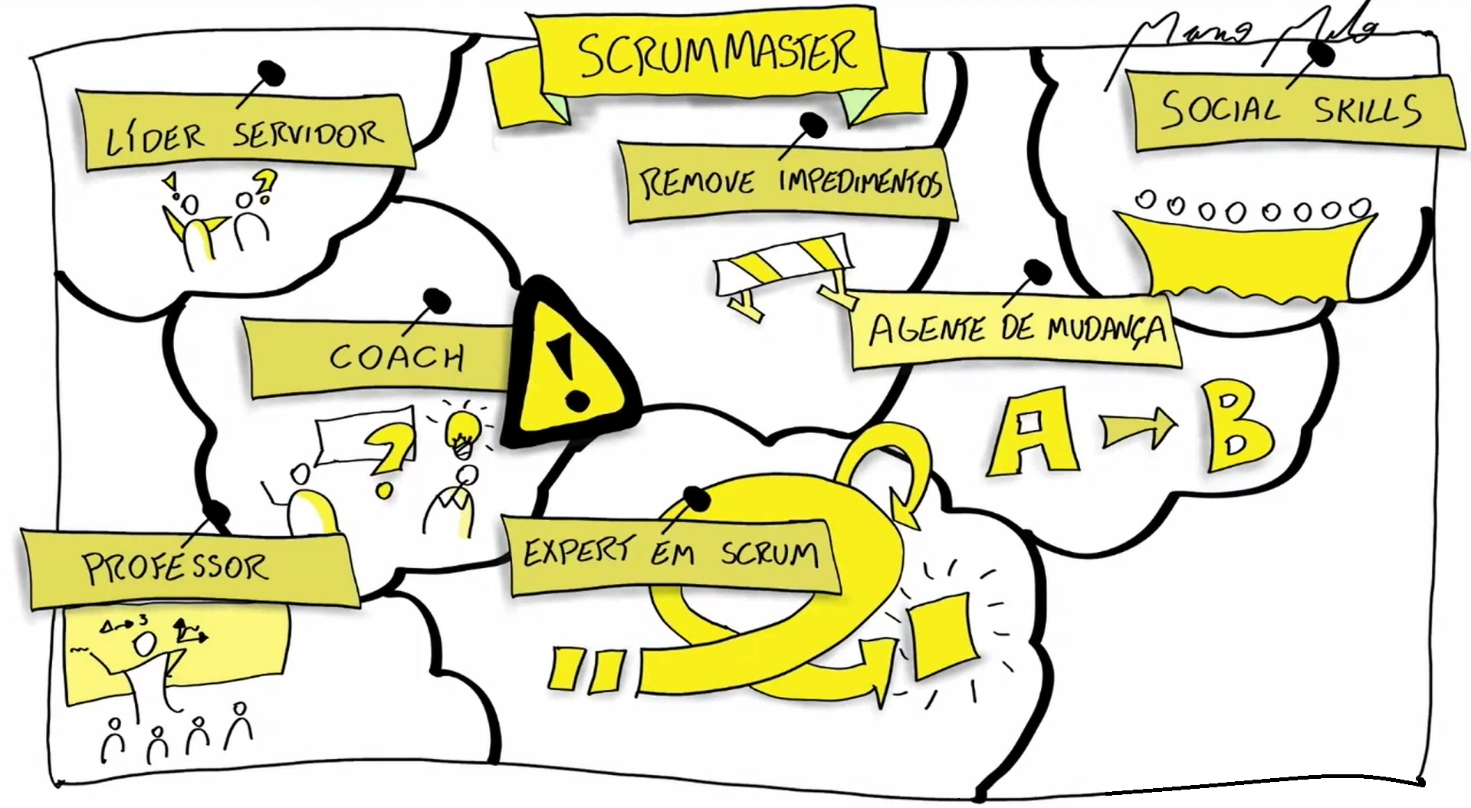


## O SCRUM MASTER

A pessoa no papel de Scrum Master é **responsável por maximizar a eficiência** do *Scrum Team*, promovendo e suportando o *Scrum* e ajudando a melhorar as interações existentes. Essa pessoa possui algumas características principais:

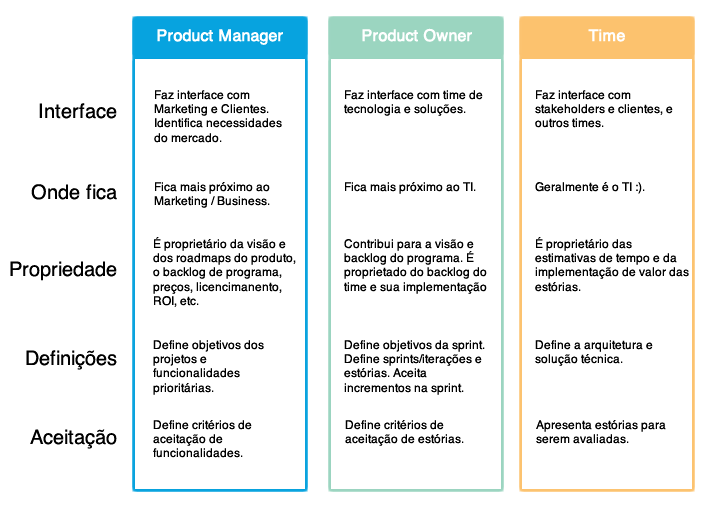
* **Líder servidor:** tem como principal objetivo fazer com que as outras pessoas tenham sucesso;
* **Coach:** uma pessoa que consegue fazer perguntas que gerem reflexão;
* **Facilitador:** consegue organizar o trabalho;
* **Professor:** consegue ensinar para a equipe;
* **Remove impedimentos**;
* **Expert em *Scrum***;
* ***Social Skills*:** capacidade de conseguir conversar e interagir com outras pessoas;
* **Agente de mudanças:** tem o objetivo de melhorar a organização, não apenas o *Scrum Team*.

**Obs.:** **fazer perguntas para o time** que gere reflexão e faça com que as pessoas consigam chegar em suas próprias respostas é papel do *Coach*, sendo justamente uma das tarefas do *Scrum Master*. O **objetivo** é colocá-las para refletir sobre um determinado assunto e sempre explorar novas alternativas pensando com os próprios conhecimentos que possuem, colocando-os em prática, seja para se colocar em alguma situação, imaginar como resolveria um determinado problema etc. Não há certo ou errado.



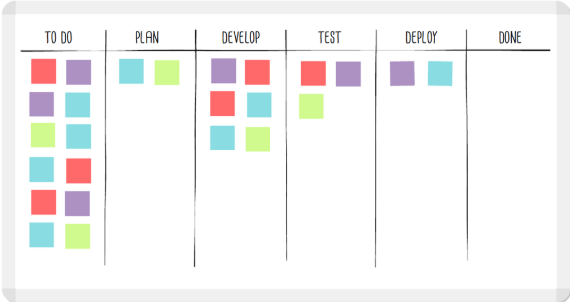
## DIFERENÇA ENTRE PO E PM

Em escala, é impossível que uma pessoa lide com a estratégia de produto e marketing e também se dedique a um time ágil. Por isso, já que o Product Manager e o Product Owner compartilham a autoridade sobre o conteúdo e estórias de cada projeto, é importante haver uma distinção clara dos papéis e responsabilidades de cada um, conforme ilustrado abaixo:



## O TIME KANBAN

O ***Kanban*** é uma forma extremamente poderosa de visualizar e limitar o trabalho que está sendo feito por uma equipe, é uma evolução de como organizar o trabalho de um *Scrum Team*, sem possuir alguns dos papéis obrigatórios de *Scrum* e com um foco grande em garantir a capacidade de trabalho. Ele é uma ferramenta interessante de se conhecer e explorar com mais detalhes para ver se sua aplicação é interessante para o seu ambiente organizacional.



O método Kanban é constituído por tarefas e é dividido em 5 etapas: **documentação, prototipação, codificação, teste** e **entrega**. Em cada uma das etapas são afixados *post-its*, que costumam conter uma breve explicação da tarefa. Um determinado grupo de pessoas atuam em cada etapa, podendo ser uma, duas, ou até mais. Temos os papéis opcionais dentro da ferramenta Kanban que são:

* **Service Delivery Manager**: é a pessoa que garante que os itens não fiquem parados, tem papel de facilitador no fluxo de mudanças e melhorias;
* **Service Request Manager**: tem como papel principal fazer a ordenação e papel dos itens, qual item terá maior impacto e retorno de investimento.

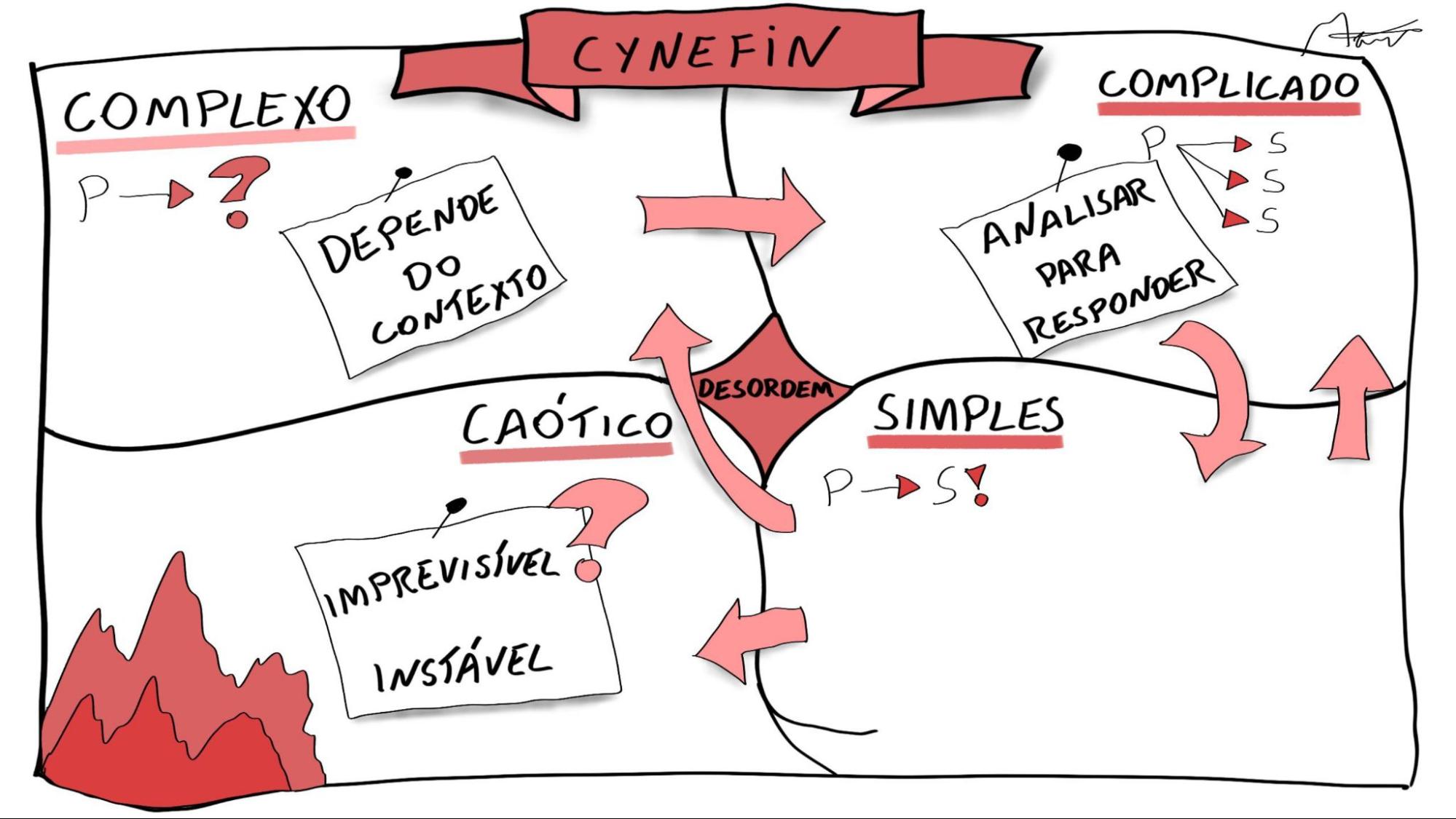
## CYNEFIN

O **framework Cynefin** — palavra de origem galesa, que significa habitat, ou ambiente, em português —, desenvolvido pelo pesquisador e consultor Dave Snowden, para auxiliar na identificação de contextos e na tomada de decisões, de acordo com as diferentes relações entre causa e efeito. A estrutura é composta por quatro quadrantes e um fator central, chamados de domínio, no caso:

* Simples;
* Complicado;
* Complexo;
* Caótico;
* Desordem.

Cada um desses domínios traz **formas de lidar com as situações dentro da empresa,** conforme o enquadramento do contexto dentro desses domínios. É essencial falar que esse framework é **fluído**, ou seja, **acompanha as percepções da realidade**. Por exemplo: A situação X hoje se enquadra no domínio complexo, mas amanhã pode se encaixar no domínio simples. É importante para sua empresa ter a percepção de que sua realidade não é estática, mas sim fluida.

Uma boa ferramenta de análise de contexto pode ajudar por meio de *sensemaking* — que em português pode ser interpretado como "fazer algo com as percepções, sensações” — ao estabelecer uma coesão às coisas que acontecem e assim, entendê-las melhor. Apesar de visualmente parecer uma matriz 2x2 ou ferramenta de categorização, não é. **Trata-se de um framework que ajuda a avaliar a dinâmica das informações inseridas na estrutura**.



Considerando a identificação de contextos e as respectivas tomadas de decisões, a partir de um problema, no domínio:

* **Simples:** a decisão é identificada de forma rápida e fácil, pois a relação entre causa e efeito é evidente. Nesse contexto é possível:
  + Sentir;
  + Categorizar;
  + Responder.
* **Complicado:** as decisões são 100% entendidas, podendo ser previstas, com tempo e conhecimento especializado. Sendo assim, neste ambiente recomenda-se:
  + Sentir;
  + Analisar;
  + Responder.
* **Complexo:** as decisões são descobertas parcialmente, após conhecer o problema. Não é possível prever uma decisão, como no domínio "Complicado''. A partir disso, para este contexto, o framework Cynefin sugere:
  + Sondar;
  + Sentir — aprender com a sondagem;
  + Responder.
* **Caótico:** não há decisões previsíveis, mesmo conhecendo o problema. Caracteriza-se pela imprevisibilidade de causa–efeito ou problema–decisão. Na falta de constantes e padrões, indica-se:
  + Agir;
  + Sentir;
  + Responder.
* **Desordem:** não se tem compreensão assertiva do contexto, problema e da(s) possível(is) solução(ões).Por essas características, a desordem está localizada no centro da estrutura, entre os quadrantes dos demais domínios.

No domínio **simples**, as pessoas já sabem quais ações são necessárias no dia a dia, o que e como devem fazer. Sendo assim, não há necessidade de Business Agility nesse cenário.

No **complicado**, por meio de constatações baseadas em tempo e conhecimento especializado, é possível prever decisões. A agilidade nesse contexto é útil, pois ajuda a reduzir riscos, feitas as análises para obter respostas necessárias. Neste domínio existem diversas possibilidades, diante de um impedimento há múltiplas formas para lidar com ele, então é necessário analisar a situação e a solução que entregue maior valor ao cliente, em um menor tempo e com menos riscos.

No **complexo**, você **não sabe o que não sabe**. É preciso sondar (cultura de experimentação), sentir (aprender) e responder (agir em sua melhoria). Uma situação que entra nesse domínio é o estabelecimento de uma nova concorrência, como aconteceu com os taxistas com o surgimento da Uber. É preciso entender se é uma ameaça ou não, aprender com as diferenças positivas e negativas, para responder, se haverá uma adaptação ao novo modelo posto pela concorrência e etc. **Este é o contexto ideal para *Business Agility***.

No domínio **caótico**, são necessárias decisões rápidas em situações completamente imprevisíveis, como no Mercado Financeiro, que dificilmente um plano cobrirá os riscos e variáveis que podem afetar esse ambiente. É preciso agir, ver o que acontecerá e responder com base na observação do que aconteceu, portanto neste contexto a agilidade não ajudará.

Mesmo que você não venha a utilizar este framework para realizar uma análise de adoção do ágil, é possível utilizá-lo para a tomada de decisões no seu negócio. Inclusive, no curso "Gestão de produtos digitais: Produto vs. Projeto", no vídeo sobre o assunto, são apresentados alguns exemplos, a partir dos quais é possível identificar a tomada de decisões em situações cotidianas, com base no *Cynefin*.