# MÉTODOS ÁGEIS

Marcela Souto Castro | André B. Barcaui


FGV

# INTRODUÇÃO

Caro aluno,

Esta apostila foi elaborada para que, de forma complementar ao material do ECLASS, pudesse oferecer uma visão teórica e prática dos métodos ágeis aplicados à gestão de projetos e escalados à organização. Nesse sentido, a apostila reúne os principais e atuais conceitos de agilidade, *frameworks*, ferramentas e reflexões que poderão ser de grande valia para o seu desenvolvimento profissional.

Os métodos ágeis na gestão de projetos não são somente uma tendência ou moda, mas uma realidade que nasceu nos projetos de *software* cuja utilização tem crescido em projetos de outras naturezas.

Para o melhor aproveitamento da disciplina, conjugue a leitura dos módulos com o material *on-line*, as videoaulas, os *podcasts*, as leituras complementares e os exercícios, tudo isso, permitindo-se um tempo para reflexão crítica e interiorização dos conceitos. Afinal, mais importante do que ler o conteúdo é relacioná-lo com o cotidiano e buscar formas de se desenvolver, profissional e pessoalmente.

Ao final do estudo, você será capaz de compreender os principais conceitos de metodologias ágeis utilizados no ambiente de gerenciamento de projetos. Para tal, vamos:

- conceituar o modelo de gestão em empresas ágeis;
- definir práticas de agilidade e colaboração;
- analisar princípios e valores ágeis;
- explicar a importância da gestão de mudanças para a agilidade;
- discutir a preparação do *product backlog*;
- conhecer as melhores práticas *sprint planning* e técnicas de estimativa;
- entender as principais cerimônias – retrospectiva, revisão, *daily stand-up*;
- reconhecer os principais papéis envolvidos – *Scrum master*, *product owner*, time de desenvolvimento;
- aprender a utilizar o *canvas* Kanban;
- discutir sobre a limitação de *work in progress*;
- analisar a mudança de paradigma na restrição tripla;

- conhecer práticas de gerenciamento de projetos originárias no desenvolvimento de *software*;
- discutir *complexity thinking* com base para o pensamento ágil;
- entender como produzir resultado e gerar valor com times autogerenciados;
- conceituar as características e as funções do *agile coach;*
- discutir crenças limitantes para a gestão ágil e
- avaliar como aplicar a proposta do *Management* 3.0.

Bons estudos!

# SUMÁRIO

# MÓDULO I – INTRODUÇÃO À AGILIDADE E AO FRAMEWORK SCRUM

Neste módulo, conheceremos os conceitos básicos de agilidade em gerenciamento de projetos, tendo como principais conceitos: histórico e Manifesto Ágil. Também veremos os principais conceitos de *Scrum* – estrutura, processos, pessoas e ferramentas em organizações ágeis – bem como as suas aplicações na prática de gerenciamentos de projetos. Além disso, estudaremos outros conceitos, como papéis, cerimônias e artefatos.

## Histórico da agilidade

> Agilidade é a habilidade de criar e responder às mudanças. É uma maneira de lidar e, finalmente, ter sucesso em um ambiente incerto e turbulento (AGILE ALLIANCE, 2020, p. 1).

Não é de hoje que as organizações têm enfrentado mudanças constantes e a necessidade de se adaptar e se tornar mais competitivas. As primeiras necessidades para essa mudança aconteceram nas linhas de produção.

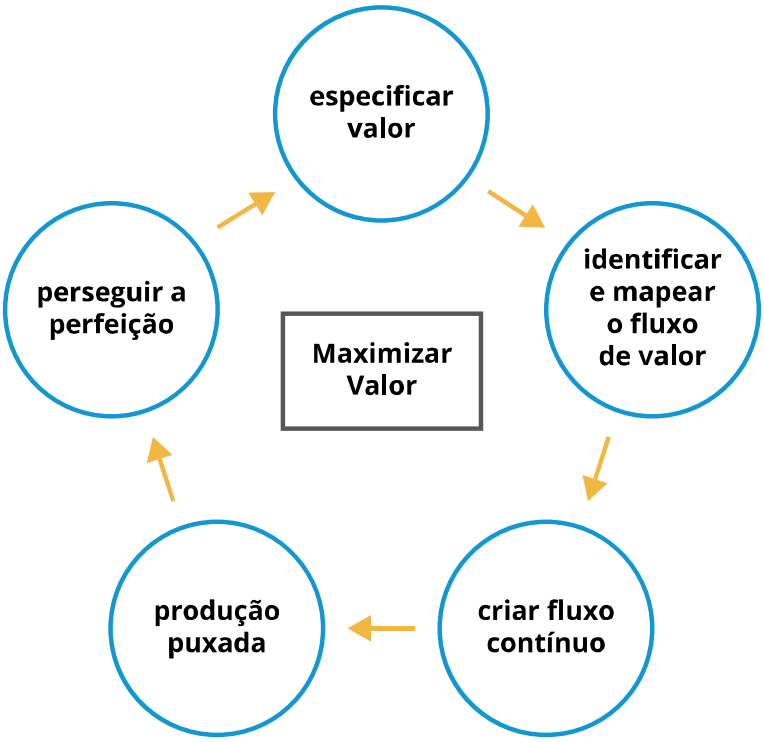
Na década de 1950, a Toyota, com o ambicioso propósito de competir com as gigantes Ford e General Motors (GM), criou o Sistema Toyota de Produção (STP). Em um cenário do Japão no pós-guerra, com uma economia devastada e um mercado interno muito limitado, a Toyota necessitava de alternativas, por isso passou a produzir pequenos volumes de diferentes modelos, utilizando a mesma linha de montagem. Esse sistema de produção foi, posteriormente, batizado de *Lean Production* ou produção enxuta. Os conceitos de *Lean* e agilidade estão fortemente relacionados, como veremos a seguir. Os princípios do *Lean Production* são:

- definir valor na perspectiva dos clientes, e não na do produtor;

- identificar a cadeia de valor e eliminar os passos que não geram valor para o cliente ou geram desperdício;
- garantir que os passos que geram valor ocorram continuamente, sem interrupções nem fronteiras entre departamentos ou entre a organização e os fornecedores;
- estabelecer a produção *puxada*, e não *empurrada*, e
- buscar a perfeição por meio da melhoria contínua.

Vejamos como tais princípios se relacionam:

Figura 1 – Princípios do *Lean Production*




**Fonte:** elaborado pelos autores

A partir desses conceitos de *Lean Production,* nasceram os conceitos de Kanban, Kaizen e *Just in Time*.

Na década de 1980, empresas dos Estados Unidos e do Japão já buscavam formas de reduzir desperdícios, bem como ter flexibilidade e velocidade no desenvolvimento de novos produtos e na incorporação de mudanças, mantendo a qualidade. Takeuchi e Nonaka (1986) afirmam que empresas, como Fuji, Canon, Honda, NEC, Epson, Brother, 3M, Xerox e Hewlett-Packard estavam utilizando uma abordagem holística no desenvolvimento de novos produtos e na melhoria contínua na linha de produção. Essa nova abordagem poderia ser comparada, analogamente, com a formação de um time de rúgbi, que se move como uma única unidade.

**Curiosidade**

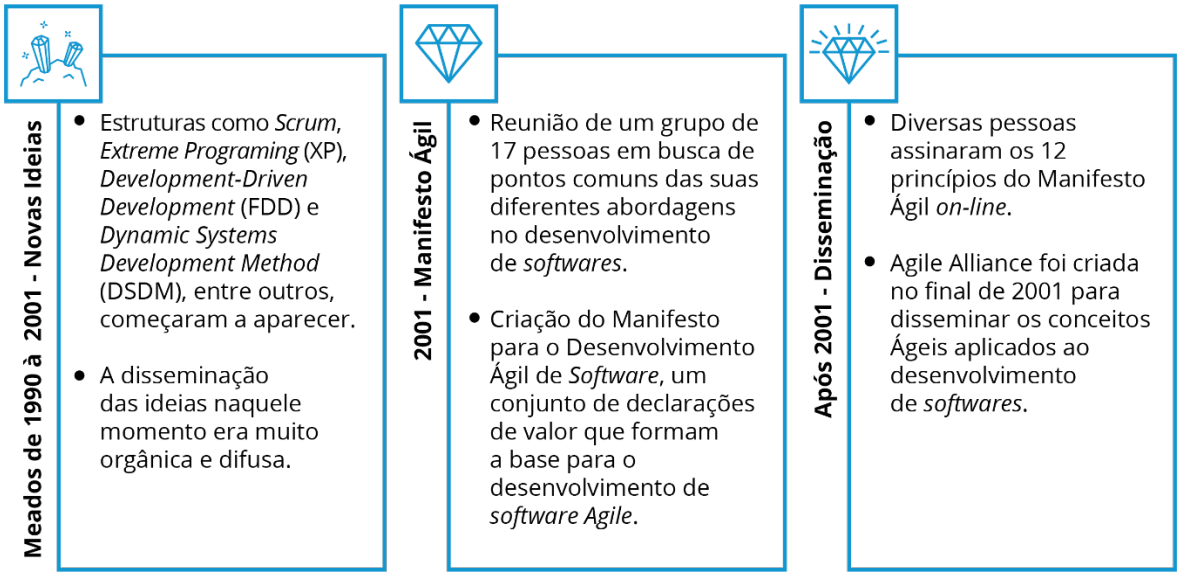
O nome *Scrum*, que veremos mais à frente, vem da formação do time de rúgbi. Em 1986, essa abordagem já era identificada em linhas de produção!

Takeuchi e Nonaka (1986) identificaram seis características em comum nessa nova abordagem organizacional: 1) instabilidades de mercado construídas com base em instabilidade; 2) equipes de projeto auto-organizadas; 3) fases de desenvolvimento sobrepostas; 4) aprendizado múltiplo; 5) controle sutil (controle rígido que prejudica a criatividade e a espontaneidade); e 6) transferência de aprendizado organizacional. Tudo isso baseado em criatividade e foco no cliente.

O princípio ágil defende a geração do resultado somente com o que é, realmente, necessário e suficiente, e da forma mais simples possível. Como o conceito de linha de produção veio parar no desenvolvimento de *softwares*?

A seguir, o quadro representa uma breve história da agilidade no desenvolvimento de *softwares*:

Quadro 1 – Agilidade no desenvolvimento de *softwares*




**Fonte:** adaptado de Agile Alliance (2020)

## Manifesto Ágil

O Manifesto Ágil foi criado em fevereiro de 2001, em uma reunião que aconteceu na estação de esqui de Snowbird, no estado de Utah, nos Estados Unidos. Assinaram-no 17 líderes representantes de ideias, metodologias e processos que, em contraste com as práticas predominantes na época, estavam trazendo valor para os seus clientes por meio de abordagens leves e empíricas para projetos de desenvolvimento de *software*.

Sabia-se que os métodos tradicionais de desenvolvimento de gestão de projetos eram pouco efetivos para o desenvolvimento de *softwares*, porque:

- os requisitos não são completamente conhecidos antes do início do projeto;
- os usuários só começam a entender o que querem após ver uma versão parcial do produto e
- os requisitos, as ferramentas e as tecnologias mudam frequentemente durante o processo de desenvolvimento.

Desse modo, comparar projetos de desenvolvimento de *software* com projetos de construção civil ou projetos militares, nos quais as metodologias tradicionais de gestão de projetos surgiram, não faz sentido.

**Manifesto para desenvolvimento ágil de *software***

Estamos descobrindo maneiras melhores de desenvolver *software*, fazendo-o nós mesmos e ajudando outros a fazerem o mesmo. Por meio desse trabalho, passamos a valorizar:

- **indivíduos e interações** mais do que processos e ferramentas;
- ***software* em funcionamento** mais do que documentação abrangente;
- **colaboração com o cliente** mais do que negociação de contratos e
- **resposta a mudanças** mais do que acompanhamento de um plano.

Em outras palavras, mesmo havendo valor nos itens à direita, valorizamos mais os itens à esquerda.

**Fonte:** Beck et al. (2001)

Vejamos cada um dos itens mencionados a seguir:

### a) Indivíduos e interações

Para Larman (2003), a programação de computadores é uma atividade humana e, sendo assim, depende de relações humanas para ser desenvolvida com sucesso. As pessoas envolvidas nos processos não podem ser trocadas como peças (COCKBURN, 2007), mas devem ser tratadas como um time. Afinal, quem gera produtos e serviços são os indivíduos, não os processos (HIGHSMITH, 2004), já que são os indivíduos que possuem características únicas, individualmente e em equipe, como talento e habilidade (HIGHSMITH, 2004). A qualidade da interação entre os membros do time é importante para a solução de problemas no desenvolvimento do projeto (COCKBURN, 2007), por isso o uso da comunicação e do *feedback* é essencial para a prática ágil (LARMAN, 2003).

Como as pessoas são consideradas mais importantes do que os processos, as ferramentas utilizadas para auxiliar o desenvolvimento de *software* devem ser as mais simples possíveis, a ponto de trazer resultados com o menor esforço possível (LARMAN, 2003).

b) *Software* em funcionamento

Um *software* em funcionamento é mais importante do que uma documentação abrangente. Highsmith (2004) ressalta que clientes se interessam por resultados e geração de valor de negócio, ou seja, *software* em funcionamento. Somente a documentação necessária deve ser produzida (COCKBURN, 2007). Segundo Highsmith (2004), a entrega iterativa de versões do *software* possibilita um *feedback* confiável no processo de desenvolvimento, substituindo a documentação desnecessária.

c) Colaboração com o cliente

É interessante a visão de Cockburn (2007) com respeito a desenvolvedores e clientes: em vez de *nós* e *eles*, há apenas *nós*. O que significa que clientes e times ágeis devem colaborar para produzir soluções que tragam valor para esses clientes e usuários finais. Como vamos observar nas dinâmicas, a exemplo do *Scrum*, ambos são necessários para que se produza *software* de boa qualidade, quando os desenvolvedores recebem *feedback* constante sobre o produto desenvolvido. Essa colaboração se mostra muito mais importante do que contratos bem desenvolvidos; além disso, torna o ambiente propício para adaptações quando há alta volatilidade, ambiguidade e incertezas (HIGHSMITH, 2004).

d) Resposta a mudanças

Muitos dizem que na gestão ágil não há planejamento; no entanto, Highsmith (2004) afirma que todo projeto deve balancear o planejar com o mudar, dependendo do nível DE volatilidade, ambiguidade e incertezas inerente a ele. Construir um plano é útil, mas seguir o plano só é útil até o momento em que ele ainda está próximo o suficiente da situação atual (COCKBURN, 2007). Afinal, manter-se preso a um plano ultrapassado não funciona em favor do sucesso.

## Princípios ágeis

Os 12 princípios ágeis foram elaborados a partir do Manifesto Ágil:

1. A nossa maior prioridade é satisfazer o cliente por meio da entrega cedo e frequente de *software* com valor.
2. Mudanças de requisitos são bem-vindas, mesmo em fases tardias do desenvolvimento. Os processos ágeis utilizam a mudança em favor da vantagem competitiva para o cliente.
3. Entregar *software* em funcionamento com frequência, a cada duas semanas ou até a cada dois meses, sendo comum a preferência por prazos mais curtos.
4. As pessoas do negócio e os desenvolvedores devem trabalhar em conjunto, diariamente, ao longo do projeto.
5. Construa projetos em torno de indivíduos motivados. Dê-lhes o ambiente e o suporte de que precisam, e confie neles para realizar o trabalho.

6. O método mais eficiente e efetivo de se transmitir informação para e entre uma equipe de desenvolvimento é a conversa face a face.
7. *Software* em funcionamento é a principal medida de progresso.
8. Os processos ágeis promovem o desenvolvimento sustentável. Os patrocinadores, desenvolvedores e usuários devem ser capazes de manter, indefinidamente, um ritmo constante.
9. A atenção contínua à excelência técnica e a um bom projeto aumenta a agilidade.
10. Simplicidade – a arte de maximizar a quantidade de trabalho não feito é essencial.
11. As melhores arquiteturas, bem como os melhores requisitos e projetos, emergem de equipes que se auto-organizam.
12. Em intervalos de tempo regulares, a equipe reflete sobre como se tornar mais efetiva e, desse modo, refina e ajusta o seu comportamento de acordo.

## O que é *Scrum*?

> Um *framework* dentro do qual pessoas podem tratar e resolver problemas complexos e adaptativos, enquanto produtiva e criativamente entregam produtos com o mais alto valor possível (SCHWABER; SUTHERAND, 2017).

O *Scrum* foi desenvolvido no início dos anos 1990, fundamentado nas teorias empíricas de controle de processo, inicialmente, para gerenciar e desenvolver produtos. Segundo Schwaber e Sutherand (2017), pode ser usado para desenvolver *softwares*, *hardwares*, redes de funções interativas, veículos autônomos e escolas, bem como para gerenciar a operação da organização. No entanto, o seu uso tem sido amplamente difundido, com aplicações bem além dessas nas organizações.

*Scrum* não é um processo, uma técnica ou um método. É um *framework*, uma cultura, dentro da qual você pode empregar vários processos ou técnicas, já existentes ou criados para propósitos específicos. O *framework Scrum* é composto de papéis, eventos, artefatos, que veremos ao longo deste módulo. Vejamos:

> Por ser um *framework*, *Scrum* pode funcionar bem quando combinado com ou complementado por diferentes métodos e práticas, que podem ser experimentados e adaptados pelo time para seu contexto específico (SABBAGH, 2013, p. 29).

**Curiosidade**

O nome *Scrum* não é um acrônimo. Vem de uma das formações em que as equipes se colocam para a reposição de bola em um jogo de rúgbi. Dessa forma, o time se move coeso em direção ao objetivo como um só bloco.

## Pilares do *Scrum*

Segundo Schwaber e Sutherand (2017), três pilares apoiam a implementação do *Scrum*:

1. **transparência** – aspectos significativos do processo devem estar visíveis aos responsáveis pelos resultados, para isso a linguagem e o entendimento do produto final devem ser comuns a todos do time;
2. **inspeção** – os times do *Scrum* devem, frequentemente, inspecionar os artefatos do *Scrum* e o progresso das entregas com o objetivo de detectar variações. Essa inspeção não deve, no entanto, ser tão frequente que atrapalhe a própria execução das tarefas, e
3. **adaptação** – caso um ou mais aspectos de um processo tenha sido considerado fora dos limites aceitáveis ou o produto final seja inaceitável, este deve ser ajustado. O ajuste deve ser realizado o mais breve possível para minimizar mais desvios.

O *framework* do *Scrum* prescreve quatro eventos formais para inspeção e adaptação, que serão vistos mais à frente nesta apostila:

- planejamento da *sprint*;
- reunião diária;
- revisão da *sprint* e
- retrospectiva da *sprint*.

## Valores do *Scrum*

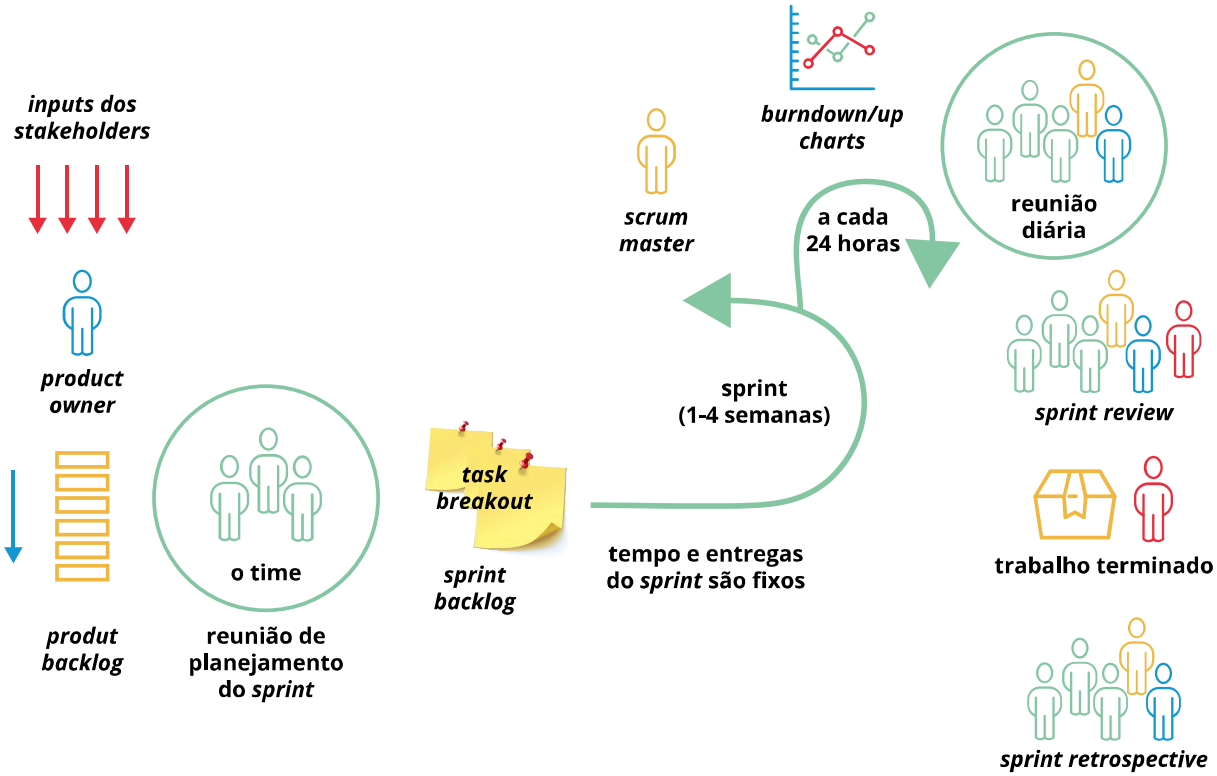
O sucesso no uso do *Scrum* depende do comprometimento do time com estes cinco valores a seguir, pois é isso o que permite uma geração de conhecimento iterativa e incremental:

1. **comprometimento** – as pessoas se comprometem pessoalmente a alcançar os objetivos;
2. **coragem** – o time *Scrum* precisa ter coragem para fazer a coisa certa e trabalhar em problemas difíceis;
3. **foco** – todos focam o trabalho da *sprint* e os objetivos do time *Scrum*;
4. **transparência** – o time *Scrum* e as partes interessadas no projeto concordam estar abertos a todo o trabalho e aos desafios com a execução dos trabalhos e
5. **respeito** – os membros do time *Scrum* respeitam uns aos outros para serem pessoas capazes e independentes.

## *Framework* do *Scrum*

A seguir, veremos o esquema do *Scrum*. Cada elemento será detalhado em seguida:

Figura 2 – Esquema *Scrum*




**Fonte:** elaborado pelos autores

# Papéis do *Scrum*

A essência do *Scrum* é um pequeno time de pessoas, altamente flexíveis, adaptativas, auto-organizáveis e multifuncionais (SCHWABER; SUTHERAND, 2017). O time *Scrum* é composto de: *product owner*, time de desenvolvimento e *Scrum master*.

## *Product owner*

Segundo Schwaber e Sutherand (2017), o *product owner* (PO) ou dono do produto é a pessoa responsável por gerenciar o *backlog* do produto. Mesmo que delegue esse papel à equipe, continua sendo responsável. O gerenciamento do *backlog* do produto inclui:

- expressar, claramente, os itens do *backlog* do produto;
- ordenar os itens do *backlog* do produto para alcançar melhor as metas e as missões;
- otimizar o valor do trabalho que o time de desenvolvimento realiza;

- garantir que o *backlog* do produto seja visível, transparente, claro para todos, e mostrar o que o time *Scrum* vai trabalhar a seguir, e
- garantir que o time de desenvolvimento entenda os itens do *backlog* do produto no nível necessário.

Para garantir um bom trabalho, o PO deve: 1) ter autoridade e respeito na equipe e na organização para que as suas decisões sejam acatadas; 2) ter conhecimento suficiente do negócio e do produto a ser desenvolvido; 3) ter conhecimento da capacidade da equipe do *Scrum*; 4) desenvolver bom relacionamento e escuta ativa com os clientes e time; 5) ter capacidade de, ao mesmo tempo, planejar e ser flexível para mudanças; e 6) estar acessível para dúvidas sobre o *backlog* do produto.

Ninguém, além do PO, tem permissão para falar com o time de desenvolvimento sobre diferentes configurações de prioridade, e o time de desenvolvimento não tem permissão para agir a partir do que outras pessoas disserem.

Mesmo representando os desejos do cliente, da organização ou do time, o PO é uma pessoa, não um comitê.

**Curiosidade**

O PO é o responsável por definir, comunicar e manter a visão do produto ao longo do projeto. Visão do produto é uma forma de traduzir esse objetivo a ser alcançado, sendo elaborada a partir das interações com clientes e usuários finais do produto do projeto.

A partir da visão do produto, o PO deve criar um plano de como se espera que o produto evolua ao longo do tempo, chamado de *roadmap* do produto. É como um plano de projeto da metodologia tradicional, mas seguindo as diretrizes ágeis de levar em conta a melhor relação entre esforço para planejar e benefícios gerados. Esse plano descreve as metas esperadas para cada momento no futuro, chamadas de metas de *roadmap*.

Embora o seu uso seja comum em projetos ágeis, essa fase de planejamento, ou pré-jogo, os artefatos, a visão do produto e o *roadmap* do produto não estão descritos no *framework Scrum*.

**Resumo**

O PO é responsável por *fatiar*, *decompor* e *priorizar* o *backlog* do produto e tem como principal característica o discernimento de transformar as ideias dos clientes em um *backlog* do produto priorizado.

## Time de desenvolvimento

Segundo Schwaber e Sutherand (2017), o time de desenvolvimento é responsável por entregar o produto pronto ou um incremento ao final de cada *sprint*. Além do disso, a partir do *backlog* priorizado disponibilizado pelo PO, o time é responsável por estimar o nível de esforço de cada item do *backlog* e definir a sua capacidade de produção em cada *sprint*.

Os times de desenvolvimento têm as seguintes características:

- são auto-organizados – ninguém, nem mesmo o *Scrum master* ou o PO, diz ao time de desenvolvimento como realizar o seu trabalho;
- são multifuncionais, possuindo todas as habilidades necessárias, como equipe, para criar o incremento do produto;
- são responsáveis tanto pelo desenvolvimento do produto quanto pela qualidade deste;
- não são reconhecidos subtimes no time de desenvolvimento, independentemente dos domínios de conhecimento que precisam ser abordados – tais como teste, arquitetura, operação ou análise de negócios –, todos trabalham e são responsáveis em conjunto. A princípio, não existem times externos responsáveis por avaliar ou garantir a qualidade do produto gerado, já que esse trabalho é de inteira responsabilidade do próprio time que desenvolve o produto (SABBAGH, 2013);
- é permitido aos integrantes possuir habilidades especializadas e áreas de especialização, individualmente, mas a responsabilidade é do time de desenvolvimento como um todo;
- são times pequenos – Schwaber e Sutheran (2017) recomenda de três a nove integrantes;
- são coesos e, muitas vezes, são mantidos temporariamente, mesmo não estando alocados em projetos, porque o esforço de montar um time integrado é maior do que mantê-lo por um tempo, mesmo sem trabalho, e
- são inseridos, de acordo com o modelo de time no *Scrum*, nos valores do *framework* – comprometimento, coragem, foco, transparência e respeito – para desenvolver o produto final da melhor forma.

O time é composto por 10 ou menos pessoas, pois equipes menores podem ser mais integradas, produtivas e comunicativas. Se o time for maior poderá ser dividido em múltiplos times coesos, focados no mesmo produto. Para isso, devem compartilhar do mesmo objetivo, *product backlog* e *product owner*. (SCHWABER *et al*, 2020).

**Curiosidade**

O time *Scrum* deve ter autoridade para inserir ou retirar algum membro do time. Inserir alguém, no caso de necessidade de uma habilidade para a entrega dos produtos; e retirar alguém do time, no caso de falta de integração ou produtividade. Essas mudanças não costumam ser temporárias para uma ou duas *sprints*. Elas devem ser feitas de longo prazo, de forma a garantir a coesão do time. O motivo disso é que o aprendizado gerado e absorvido leva o time a um aumento progressivo da sua eficiência e da sua produtividade (SABBAGH, 2013).

**Resumo**

Times *Scrum* são auto-organizáveis, multifuncionais, coesos, pequenos e com alta maturidade profissional. São responsáveis por definir o nível de esforço das atividades do *backlog* e a sua capacidade de produção, além de desenvolver o produto final.

## *Scrum master*

O *Scrum master* (SM) é um servo-líder do time de desenvolvimento, responsável por promover e suportar o *Scrum* como definido no Guia *Scrum*. O SM faz isso ajudando o time a entender a cultura, as práticas, as regras e os valores do *Scrum* (SCHWABER; SUTHERAND, 2017). O SM também é responsável por eliminar ou diminuir os ruídos externos que possam atrapalhar o desempenho da equipe, e por auxiliar a comunicação e a integração do time.

Segundo Schwaber e Sutherand (2017), o SM serve o PO de várias maneiras, incluindo:

- garantir que objetivos, escopo e domínio do produto sejam entendidos o melhor possível por todos do time *Scrum*;
- encontrar técnicas para o gerenciamento efetivo do *backlog* do produto;
- ajudar o time *Scrum* a entender as necessidades para ter itens de *backlog* do produto claros e concisos.
- compreender o planejamento do produto em um ambiente empírico;
- garantir que o PO saiba como organizar o *backlog* do produto para maximizar valor;
- compreender e praticar a agilidade e
- facilitar os eventos *Scrum* conforme exigido ou necessário.

Também segundo Schwaber e Sutherand (2017), o SM serve o time de desenvolvimento de várias maneiras, incluindo:

- treinar o time de desenvolvimento em autogerenciamento e interdisciplinaridade;
- ajudar o time de desenvolvimento na criação de produtos de alto valor;
- remover impedimentos para o progresso do time de desenvolvimento;
- facilitar os eventos *Scrum* conforme exigidos ou necessários e
- treinar o time de desenvolvimento em ambientes organizacionais nos quais o *Scrum* não é totalmente adotado e compreendido.

Ainda segundo Schwaber e Sutherand (2017), o SM serve à organização de várias maneiras, incluindo:

- liderar e treinar a organização na adoção do *Scrum*;
- planejar implementações *Scrum* dentro da organização;
- ajudar funcionários e partes interessadas a compreender e tornar aplicável o *Scrum* e o desenvolvimento de produto empírico;
- causar mudanças que aumentam a produtividade do time *Scrum* e
- trabalhar com outros SMs para aumentar a eficácia da aplicação do *Scrum* na organização.

**Curiosidade**

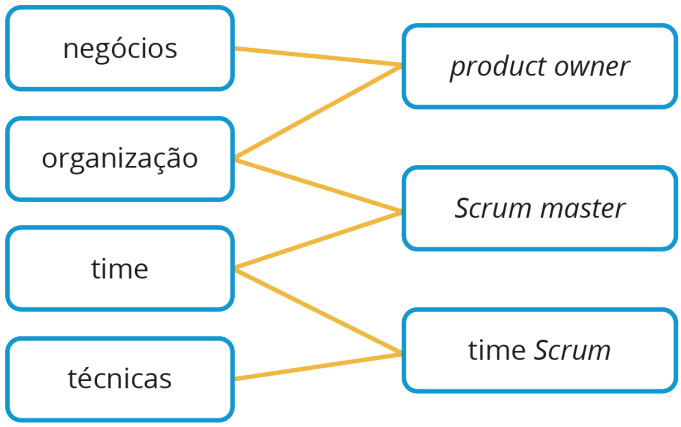
Como um bom facilitador, o SM deve manter-se neutro durante os eventos e as cerimônias, bem como durante as decisões do time. O seu papel é facilitar, estimulando e, ao mesmo tempo, controlando os ânimos das discussões.

**Resumo**

O SM é um líder-servidor, responsável por garantir que o *Scrum* seja entendido e aplicado. Também deve excluir interferências externas; auxiliar o PO, aumentando a eficiência do time; desenvolver o time; e aumentar a eficácia da aplicação do *Scrum* na organização.

A seguir, a figura descreve as relações entre as dimensões de conhecimento e os papéis do *Scrum*, de forma que seja possível identificar as habilidades e os conhecimentos necessários para que cada papel possa evoluir na carreira.

Figura 3 – Relações entre dimensões de conhecimento e papéis do *Scrum*




**Fonte:** adaptado de Sabbagh (2013)

# Cerimônias ou eventos do *Scrum*

## *Sprint*

O coração do *Scrum* é a *sprint*, um período delimitado (*timebox*) de um mês ou menos, durante o qual um produto pronto – versão incremental potencialmente utilizável do produto – é criado (SCHWABER; SUTHERAND, 2017).

As *sprints* têm sempre a mesma duração durante todo o projeto. Normalmente, duram de uma a quatro semanas e devem ser coerentes com o trabalho a ser desenvolvido. Se o tempo da *sprint* ultrapassar um mês, a definição do que será construído pode mudar, a complexidade pode aumentar, e o risco pode crescer, por isso não é indicado que dure mais de quatro semanas (SCHWABER; SUTHERAND, 2017). Uma nova *sprint* inicia imediatamente após a conclusão da *sprint* anterior. Além disso, ao final de toda *sprint*, deve ser entregue um produto pronto ou parte dele.

As *sprints* são compostas de: reunião de planejamento da *sprint*, reuniões diárias, trabalho de desenvolvimento, revisão da *sprint* e retrospectiva da *sprint*.

Durante a *sprint* (SCHWABER; SUTHERAND, 2017):

- não são feitas mudanças que possam pôr em perigo o seu objetivo;
- caso a mudança não afete o seu objetivo, é possível fazer a troca de itens que não foram iniciados;
- as metas de qualidade não diminuem e
- o escopo pode ser esclarecido e renegociado entre o PO e o time de desenvolvimento.

**Curiosidades**

Somente o PO pode cancelar uma *sprint* em andamento, e isso acontecerá se o *backlog* da *sprint* se tornar obsoleto. Esse cancelamento pode ocorrer se a organização mudar a sua direção ou se as condições do mercado ou das tecnologias mudarem.

Mesmo quando o time, perto do final do período estabelecido para a *sprint*, tiver certeza de que não cumprirá o objetivo do ciclo, o seu *timebox* não será estendido, pois este é fixo (SABBAGH, 2013). Da mesma forma, disfunções como horas extras e aceleração forçada do ritmo de trabalho devem ser evitadas (SABBAGH, 2013). Essas situações gerarão aprendizado para a equipe.

A primeira *sprint* servirá como base para criar referências de nível de esforço para cada *user story* e para a velocidade do time de desenvolvimento.

## Reunião de planejamento da *sprint*

O trabalho a ser realizado na *sprint* é planejado na reunião de planejamento da *sprint,* em que o plano é criado com o trabalho colaborativo de todo o time *Scrum* (SCHWABER; SUTHERAND, 2017). A reunião de planejamento da *sprint* possui um *timebox* com no máximo oito horas para uma *sprint* de um mês de duração. Para *sprints* menores, este evento é usualmente menor.

O SM é o facilitador dessa reunião e, dessa forma, deve garantir que o evento ocorra e que os participantes entendam o seu propósito.

O PO deve ajudar a clarificar os itens de *backlog* do produto selecionado e nas decisões conflituosas de troca, assim como decompor itens do *backlog*, caso seja necessário.

A reunião de planejamento da *sprint* responde às seguintes questões (SCHWABER; SUTHERAND, 2017):

- O que pode ser entregue como resultado do incremento da próxima *sprint*?
  Somente o time de desenvolvimento pode avaliar o que pode ser executado ao longo da próxima *sprint*.
  Como o trabalho necessário para entregar o incremento será realizado?
- O trabalho planejado pelo time de desenvolvimento para os primeiros dias da *sprint* é decomposto até o final desta reunião, frequentemente, em unidades de um dia de duração ou menos. O time de desenvolvimento se auto-organiza para realizar todo o trabalho do *backlog* da *sprint*, tanto durante o planejamento da *sprint* quanto no que for necessário durante a *sprint*.

A meta da *sprint* é o objetivo definido para a *sprint,* fornece uma direção para o time de desenvolvimento sobre o porquê de estar construindo o incremento. Nessa reunião, os critérios de aceitação também devem ser definidos.

## Reunião diária

A reunião diária do *Scrum* é um evento com prazo delimitado (*timeboxed*) de 15 minutos, para que o time de desenvolvimento possa sincronizar as atividades e criar um plano para as próximas 24 horas.

Essa reunião é feita para inspecionar o trabalho desde a última reunião diária, e prever o trabalho que deverá ser feito antes da próxima reunião diária. Para agilizar o tempo e reforçar o cumprimento do *timebox*, essa reunião costuma ser feita em pé (*stand up meeting*) e no mesmo horário e local.

Alguns times de desenvolvimento utilizarão perguntas, outros se basearão em discussões. Vejamos um exemplo de perguntas que podem direcionar a reunião (SCHWABER; SUTHERAND, 2017):

- O que eu fiz ontem que ajudou o time a atingir a meta da *sprint*?
- O que eu farei hoje para ajudar o time a atingir a meta da *sprint*?
- Eu vejo algum obstáculo que me impeça, ou impeça o time, no atingimento da meta da *sprint*?

## Reunião de revisão da *sprint*

A revisão da *sprint* é executada no final da *sprint* para inspecionar o incremento e adaptar o *backlog* do produto se necessário (SCHWABER; SUTHERAND, 2017). Durante a reunião de revisão da *sprint,* o time *Scrum* e as partes interessadas – isto é, time, PO, SM e clientes – verificam o que foi feito na *sprint*. Essa é uma reunião informal, não uma reunião de *status*, e a apresentação do incremento se destina a motivar, obter comentários e promover a colaboração. Essa é uma reunião com prazo delimitado de quatro horas de duração para uma *sprint* de um mês. Para *sprints* menores, esse evento é, usualmente, menor.

A revisão da *sprint* inclui os seguintes elementos (SCHWABER; SUTHERAND, 2017):

- Os participantes incluem o time *Scrum* e os *stakeholders*-chave convidados pelo PO.
- O PO esclarece quais itens do *backlog* do produto foram *prontos* e quais não foram *prontos*.
- O time de desenvolvimento demonstra o trabalho que está *pronto* e responde às questões sobre o incremento.
- O PO discute o *backlog* do produto tal como está. Além disso, projeta os prováveis alvos e datas de entrega baseado no progresso até a data (se necessário).
- O grupo todo colabora sobre o que fazer a seguir, e é assim que a revisão da *sprint* fornece valiosas entradas para o planejamento da *sprint* subsequente:
  - revisão de como o mercado ou o uso potencial do produto pode ter mudado, e o que é a coisa mais importante a se fazer a seguir, e
  - revisão da linha do tempo, orçamento, potenciais capacidades e mercado para a próxima versão esperada de funcionalidade ou de capacidade do produto.

Cada uma dessas entregas proporciona percepção do valor na visão dos clientes do projeto, além de possibilitar o seu *feedback* rápido sobre o produto para que se realizem as mudanças ou adições necessárias já para a próxima *sprint* (SABBAGH, 2013).

Nessa reunião, os detalhes do produto emergem ao longo do desenvolvimento do produto (SABBAGH, 2013).

## Retrospectiva da *sprint*

A retrospectiva da *sprint* é uma oportunidade para o time *Scrum* inspecionar a si próprio e criar um plano para melhorias a serem aplicadas na próxima *sprint* (SCHWABER; SUTHERAND, 2017). Participam da reunião de retrospectiva o time, o PO e o SM.

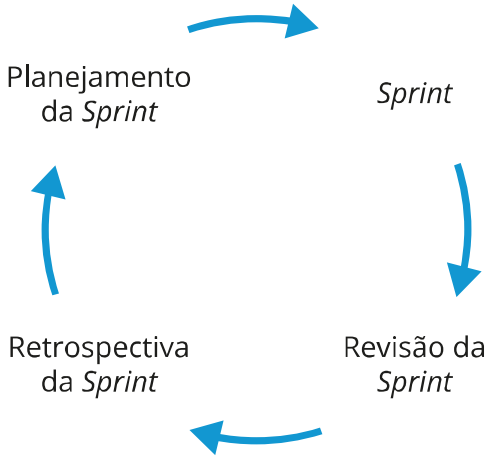
O propósito da retrospectiva da *sprint* é (SCHWABER; SUTHERAND, 2017):

- inspecionar como a última *sprint* foi em relação às pessoas, aos relacionamentos, aos processos e às ferramentas;
- identificar e ordenar os principais itens que foram bem e as potenciais melhorias e
- criar um plano para implementar melhorias no modo que o time *Scrum* faz o seu trabalho.

A retrospectiva da *sprint* deve ocorrer depois da revisão da *sprint* e antes da reunião de planejamento da próxima *sprint*. Esta é uma reunião com prazo delimitado de três horas de duração para uma *sprint* de um mês. Para *sprints* menores, esse evento costuma ser menor.

As cerimônias e os eventos se relacionam da seguinte maneira:

Figura 4 – Cerimônias e eventos




**Fonte:** elaborado pelos autores

O ciclo se inicia com uma reunião de planejamento de *sprint* e se encerra somente ao final do projeto, com uma reunião de retrospectiva, que servirá para aprendizado e integração da equipe.

Um projeto pode ser encerrado porque:

- o tempo de contrato terminou;
- os objetivos do projeto foram entregues para o cliente;
- o orçamento do projeto acabou ou
- o contrato foi cancelado, por exemplo.

Em geral, um projeto possui uma duração determinada, e o máximo de valor é entregue nesse tempo.

No ato do encerramento do projeto, espera-se que as necessidades ou os objetivos de negócios de maior importância já tenham sido entregues, isto é, a *visão do produto* tenha sido alcançada.

Tanto a reunião de revisão da *sprint* como a retrospectiva da *Sprint* devem ocorrer no último dia da *sprint*. A reunião de planejamento da *sprint* seguinte pode ocorrer em seguida ou no início da próxima *sprint*.

## Artefatos do *Scrum*

Os artefatos do *Scrum* representam o trabalho ou o valor a serem desenvolvidos pelo projeto. Eles possibilitam transparência e oportunidades para inspeção e adaptação.

### *Backlog* do produto

O *backlog* do produto é uma lista ordenada, em termos de prioridades, de tudo o que deve ser necessário no produto. É produzida e constantemente atualizada pelo PO. O *backlog* do produto evolui durante o desenvolvimento do produto de acordo com as necessidades dos clientes, do negócio e do ambiente no qual ele será inserido. Isto é, o *backlog* do produto é dinâmico, mudando para ser mais apropriado, competitivo e útil.

**Curiosidade**

Os itens do alto do *product backlog* são os mais importantes, pois serão utilizados naquele momento, por isso possuem mais detalhes para serem desenvolvidos primeiro. Os itens mais abaixo têm, gradativamente, menos detalhes (SABBAGH, 2013). Até porque é muito provável que os itens de baixo sejam modificados ao longo do tempo. Como o nível de esforço do planejamento deve trazer benefícios, detalhá-los antecipadamente pode significar um retrabalho futuro.

### *Backlog* da *sprint*

O *backlog* da *sprint* é definido a partir do *backlog* do produto. Ele representa a previsão do time de desenvolvimento sobre quais itens do *backlog* estarão no próximo incremento e sobre o trabalho necessário para entregar esses itens. O *backlog* da *sprint* torna visível todo o trabalho que o time de desenvolvimento identifica que será necessário para atingir o objetivo da *sprint*.

Junto com o *backlog*, deve ser definido o compromisso do *sprint* (ou *sprint goal*). Este compromisso proporciona flexibilidade em termos do trabalho exato necessário para alcançar a necessidade do usuário. Se o trabalho se revelar diferente do que era esperado, o escopo do *sprint backlog* pode ser negociado dentro do *sprint* sem afetar o *sprint goal*.

O *sprint goal* deve estar bem alinhado com os "porquês" identificados nas *user stories*. (SCHWABER *et al*, 2020).

## Incremento

O incremento é a soma de todos os itens do *backlog* do produto completados durante uma *sprint* somado ao valor dos incrementos de todas as *sprints* anteriores (SCHWABER; SUTHERAND, 2017). O incremento deve estar na condição de ser utilizado, isto é, pronto, independentemente de o PO decidir liberá-lo para o cliente ou não.

Junto com o incremento deve ser definido o compromisso com a qualidade do que será entregue: *a definition of done*. A expressão *a definition of done* é uma descrição formal das medidas de qualidade exigidas para o produto. (SCHWABER *et al*, 2020).

# MÓDULO II – KANBAN E FERRAMENTAS ÁGEIS

Neste módulo, veremos os principais conceitos de Kanban e as suas aplicações na prática de gerenciamentos de projetos. Teremos como principais conceitos: visibilidade e *work in progress* (WIP), bem como ferramentas e aplicativos para gestão ágil de projetos.

## O que é Kanban

Em japonês, Kanban significa, *cartão visual*. O termo tem-se tornado quase sinônimo da implementação dos princípios *Lean* e de agilidade.

Embora sistemas Kanban sejam um conceito relativamente novo em TI, vêm sendo utilizados por mais de 50 anos na Toyota, para visualmente controlar e equilibrar a linha de produção. A metodologia se baseia em transparência total e comunicação em tempo real sobre capacidade de produção.

O Kanban é baseado em uma ideia muito simples: as atividades em andamento devem ser limitadas, de modo que uma nova atividade só deve ser iniciada quando a anterior é terminada. Não é um processo ou um ciclo de vida de gerenciamento de projetos ou de desenvolvimento de *software* (KNIBERG; SKARIN, 2009). É uma abordagem para mudança gerencial, de acordo com o conceito de melhoria contínua. Pode ser utilizado para introduzir mudanças em um ciclo de desenvolvimento de *software* ou metodologia de gerenciamento de projetos.

Para iniciar a abordagem de Kanban é preciso entender o processo atual ao mapear o fluxo de valor, em seguida, limitar as atividades em andamento – do inglês *work in process* (WIP) – para cada estágio desse processo. A partir daí, o Kanban usa um mecanismo de controle visual para acompanhar o trabalho à medida que ele flui por meio das várias etapas do fluxo de valor, dando transparência ao processo e ao seu fluxo, expondo gargalos, filas, variabilidade e desperdício. A visibilidade dos gargalos,

o desperdício, a variabilidade e os seus impactos incentivam a discussão sobre melhorias, e as equipes iniciam, rapidamente, as melhorias nos seus processos. Como resultado, o Kanban encoraja a evolução incremental de processos existentes, geralmente alinhada a valores ágeis e *Lean* (KNIBERG; SKARIN, 2009). Além disso, não demanda uma revolução drástica no modo como as pessoas trabalham, mas encoraja uma mudança gradual (KNIBERG; SKARIN, 2009).

Kanban é um método de gestão de mudanças, que dá ênfase aos seguintes princípios:

- visualizar o trabalho em andamento;
- visualizar cada passo na sua cadeia de valor, do conceito geral até o *software* que se possa lançar;
- limitar o trabalho em progresso – WIP –, restringindo o total de trabalho permitido para cada estágio;
- tornar explícitas as políticas sendo seguidas;
- medir e gerenciar o fluxo, para poder tomar decisões bem embasadas, além de visualizar as consequências dessas decisões, e
- identificar oportunidades de melhorias, criando uma cultura Kaizen, na qual a melhoria contínua é responsabilidade de todos.

**Fonte:** Boeg (2011)

**Curiosidade**

O Kanban não é parte do *Scrum*! No entanto, vem sendo muito utilizado por times de desenvolvimento, de forma integrada ao *Scrum* ou isoladamente.

Quando trabalhado em conjunto com o *Scrum*, normalmente, o método é conhecido como *Scrumban*. O objetivo continua a ser de visualizar o fluxo de trabalho com um quadro Kanban, mas fazendo uso das cerimônias *Scrum* de planejamento, *daily Scrum* e retrospectiva (Kaizen).

No caso do *Scrumban*, novas tarefas podem ser inseridas durante uma iteração, mas continua a preocupação com a limitação do WIP. Normalmente, não são designados papéis para o time e existe a opção por trabalhar com *timebox* ou não.

## Visibilidade no Kanban

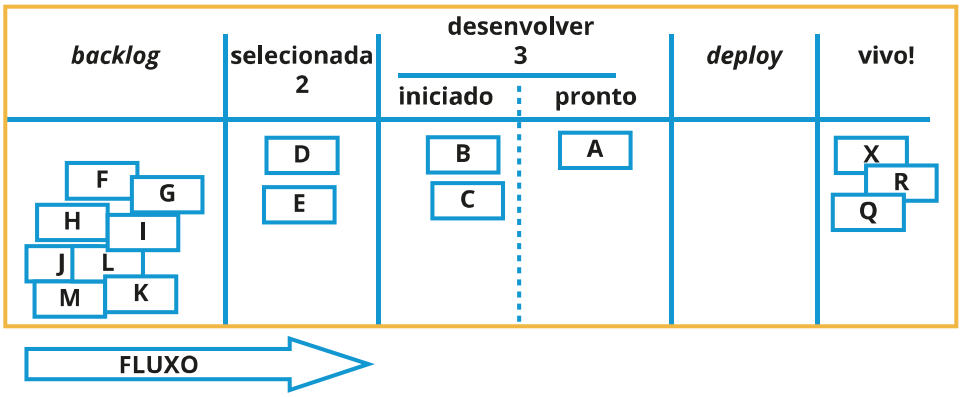
Para ser capaz de tomar decisões bem embasadas, o primeiro passo do Kanban para visualizar o fluxo de trabalho é entender como o seu sistema atual funciona (BOEG, 2011). Para isso, deve-se mapear todo o fluxo de trabalho atual, resistindo à tentação de pensar logo em melhorias. A visibilidade é sempre o primeiro passo!

Quadro Kanban e os seus cartões:

- O quadro Kanban é formado por cartões de especificações.
- As colunas no Kanban representam o estágio do desenvolvimento de cada cartão.

O quadro Kanban é parecido com o que vemos a seguir. Digo parecido, pois ele deve refletir o fluxo atual dos processos da equipe.

Figura 5 – Quadro Kanban




**Fonte:** Kniberg e Skarin (2009)

No exemplo apresentado, a coluna *backlog* é uma lista de coisas desejáveis, sem qualquer ordem especial. A coluna *selecionados* contém os itens com maiores prioridades, com o limite Kanban igual a 2. Dessa forma, deve conter apenas dois itens com alta prioridade ao mesmo tempo. Sempre que estiver pronta para começar a trabalhar em um novo item, a equipe vai pegar o primeiro item da coluna *selecionados*.

A coluna *desenvolver* – dividida em duas subcolunas: *iniciado* e *pronto* – mostra o que está sendo desenvolvido no momento, com um limite Kanban de 3.

Atenção! A qualquer momento, o responsável pelo quadro poderá fazer mudanças nas colunas *backlog* e *selecionados*, mas não poderá mudar as demais colunas.

**Curiosidade 1**

No exemplo, caso não houvesse nenhum item na coluna *iniciado* e três itens na coluna *pronto*, pela limitação de itens no fluxo, os desenvolvedores não poderiam iniciar um novo item selecionado. Isso lhes dá forte incentivo para concentrar esforços e ajudar a colocar as coisas em produção para limpar a coluna *pronto* e maximizar o fluxo.

O limite das atividades em andamento existe para impedir que problemas fujam do controle, uma vez que, se as coisas estão fluindo bem, os limites das atividades em andamento não são usados realmente.

**Fonte:** Boeg (2011)

**Curiosidade 2**

Algumas equipes também utilizam as cores para identificar se o cartão representa uma história do usuário, um defeito, uma tarefa, uma característica ou qualquer outra classificação do produto final que queira comunicar. Como exemplo, vamos tomar o desenvolvimento de um aplicativo:

- História do usuário – é uma descrição de alguma funcionalidade que um aplicativo precisa ter para atender a uma necessidade de uma *persona*. É formada por: *Quem*? *O quê*? *Por quê*? Por exemplo, o criador do aplicativo precisa que os clientes se cadastrem para ter um banco de dados para futuras vendas e comunicados.
- Defeito – algo que já foi feito, mas precisa ser corrigido. Por exemplo, o campo de *e-mail* do cadastro está limitado a 10 caracteres e precisa ser aumentado para 25.
- Tarefa – são atividades simples. Por exemplo, fazer *backup* do banco de dados.
- Característica – são peculiaridades que o produto deve ter. Por exemplo, todas as páginas do aplicativo precisam conter a logomarca da empresa.

**Fonte:** Boeg (2011)

Uma boa analogia é uma corrente de clipes de papel. Ela deve ser puxada, com uma velocidade ótima para não arrebentar. Por outro lado, caso empurrada, ficará embolada em algum ponto, mostrando desperdícios e acúmulo de trabalho. Com essa dinâmica visual, o Kanban permite identificar desperdícios, gargalos e retrabalhos, estimulando a própria equipe e responsáveis a implantar a melhoria contínua do processo.

## *Work in progress* no Kanban

As única duas coisas que o Kanban prescreve é que o fluxo de trabalho deve ser visual e que o WIP deve ser limitado.

Ser visual permite uma comunicação aberta com o time e o exterior, e todos podem contribuir com ideias para melhorar o fluxo de trabalho de forma contínua.

Já limitar a quantidade de trabalho faz com que o time não entre em zonas de conforto, por exemplo, começar algo menos importante, e não correr atrás de um impedimento de um trabalho em andamento. A afirmação de que "quanto menos trabalho fazemos ao mesmo tempo maior é a nossa eficiência" pode parecer um pouco contraintuitiva, no entanto, como demonstra a Lei de Little, essa restrição é a chave para diminuir tempo de entrega (SABBAGH, 2013).

A Lei de Little foi apresentada por John Little, em um artigo publicado no jornal acadêmico *Operations Research*, em 1961, e sugere que, em um sistema estável, a quantidade média de tempo – *throughput* – que algo leva para atravessar um processo é igual ao número de itens no processo, dividido pela sua taxa de conclusão média.

Em outras palavras:

| *throughtput* = WIP / *lead time* |
|---|

e

| *lead time* = WIP / *throughput* |
|---|

Desse modo, quanto maior for o WIP, maior tende a ser o *throughtput*.

O objetivo do Kanban é criar um bom fluxo por meio do sistema e minimizar o *lead time*. Nesse sentido, segundo Kniberg e Skarin (2009), você precisa, regularmente, levantar questões como:

**Quais colunas devemos ter?**

Cada coluna representa um estado de fluxo de trabalho ou uma pilha – fila – entre dois estados de fluxo de trabalho. Comece simples e adicione colunas conforme o necessário.

**Quais devem ser os limites do Kanban?**

Quando o limite de Kanban para as suas colunas for atingido e você não tiver nada para fazer na coluna seguinte, comece a procurar por um gargalo e por formas de minimizá-lo ou corrigi-lo.

Se você notar que muitos itens continuam por um longo tempo sem ser trabalhados, é uma indicação de que o limite de Kanban pode estar muito alto:

- Limite de Kanban muito baixo → pessoas ociosas → má produtividade.
- Limite de Kanban muito alto → tarefas inativas → *lead time* ruim.

**Quão restritos são os limites de Kanban?**

Algumas equipes tratam os limites como regras estritas que não podem exceder, outras os tratam como diretrizes ou disparadores de discussão. Desta forma, quebrar um limite de Kanban é permitido, mas deve ser uma decisão intencional com uma razão concreta.

Segundo Boeg (2011), 10 passos são necessários para a implantação do Kanban:

1. **Passo 1** – visualização do fluxo de trabalho;
2. **Passo 2** – limite do trabalho em progresso;
3. **Passo 3** – estabelecimento de políticas explícitas para a garantia de qualidade;
4. **Passo 4** – ajuste de cadências;
5. **Passo 5** – medição do fluxo;

6. **Passo 6** – priorização – não é obrigatório para o Kanban, no entanto, se houver necessidade, o *backlog* pode ser priorizado;
7. **Passo 7** – identificação de classes de serviço – veja alguns exemplos típicos de tipos de trabalho:
   - histórias de usuário (pequenas, médias e grandes);
   - defeitos (cosméticos, críticos e impeditivos);
   - relatórios manuais;
   - edições textuais;
   - tarefas de suporte e
   - instalação.
8. **Passo 8** – gerenciamento do fluxo – ao chegar a este ponto, já estaremos operando em um ambiente ágil e maduro. Visualizamos o fluxo de trabalho, o trabalho em progresso está limitado, políticas de qualidade estão estabelecidas, o fluxo está sendo medido;
9. **Passo 9** – estabelecimento de *Service Level Agreement* (SLAs) e
10. **Passo 10** – melhoria contínua.

A seguir, vejamos um quadro comparativo entre Kanban e *Scrum*:

Quadro 2 – Kanban *versus Scrum*

| ***Scrum*** | **Kanban** |
|---|---|
| **semelhanças** | |
| São *Lean* e *Agile*. | |
| Usam controle de cronograma. | |
| Limitam atividades em andamento. | |
| Usam transparência para direcionar a melhoria do processo. | |
| Concentram-se na entrega de *software* que funcione, o mais rápido possível e frequentemente. | |
| São baseados em equipes auto-organizáveis. | |
| Exigem que o trabalho seja dividido em partes. | |
| O planejamento de *release* é continuamente otimizado, baseado em dados (velocidade/tempo de execução) empíricos. | |

| diferenças | |
|---|---|
| Iterações *timeboxed* prescritas. | Iterações *timeboxed* opcionais – pode ter cadência separada para o planejamento, o *release* e a melhoria de processos. Pode ser orientada para eventos, em vez de *timeboxed*. |
| Equipe se compromete a uma quantidade específica de trabalho para essa iteração. | Compromisso opcional. |
| Usa a velocidade como padrão métrico para o planejamento e a melhoria de processos. | Usa o *lead time* como padrão métrico para o planejamento e a melhoria de processos. |
| Equipes multifuncionais prescritas. | Equipes multifuncionais opcionais. Equipes de especialistas autorizadas. |
| Os itens devem ser divididos para que possam ser concluídos dentro de uma *sprint*. | Nenhum tamanho especial de item é prescrito. |
| Gráfico *burndown*. | Nenhum tipo específico de diagrama é prescrito. |
| WIP limitado indiretamente (por *sprint*). | WIP limitado diretamente (por situação do fluxo de trabalho). |
| Estimativa prescrita. | Estimativa opcional. |
| Não poderá adicionar itens à iteração em uso. | Pode adicionar novos itens sempre que houver capacidade disponível. |
| O *sprint backlog* é de uma equipe específica. | Quadros Kanban podem ser compartilhados por várias equipes ou individualmente. |
| Possui três papéis: PO, SM e time. | Não determina nenhum papel. |
| Um quadro *Scrum* é reiniciado entre cada *sprint*. | Um quadro Kanban continua. |
| Prescreve um *product backlog* priorizado. | Priorização é opcional. |

**Fonte:** Kniberg e Skarin (2009)

# Ferramentas ágeis

## Sete dimensões do produto

As sete dimensões de um produto são uma ferramenta como um *canvas*. É uma técnica de análise do produto que pode servir para a criação e o alinhamento do que será necessário para o desenvolvimento e a identificação de critérios de análise e critérios de qualidade.

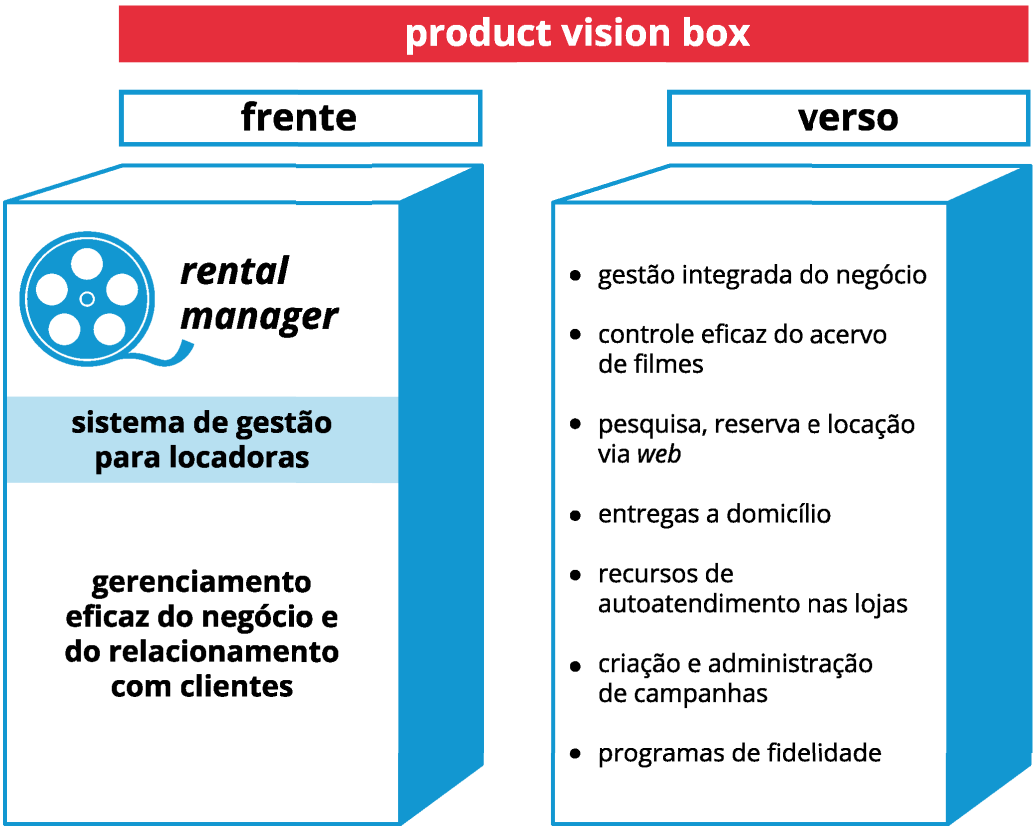
As dimensões estão listadas a seguir:

- **Ator** – Quem é o ator para qual o produto está sendo construído?
- **Interface** – O que é necessário para a interação com o usuário? Por exemplo: tela, serviço ou parceria com serviço externo existente.
- **Ação** – O que é possível fazer com essa interface? Quais as ações previstas durante a utilização da tela? Por exemplo: alterar, incluir, buscar, localizar, etc.
- **Informação** – Quais tipos de informação podem ser manipuladas pelo ator?
- **Regras do negócio** – Existe alguma regra de negócio a ser considerada?
- **Ambiente** – Em qual plataforma o ambiente vai funcionar?
- **Requisitos não funcionais** – Existem requisitos não funcionais que impactam a funcionalidade que está sendo construída?

## *Product vision box*

Na criação de um produto, o PO pode solicitar à equipe que transforme a ideia em uma caixa de um produto, como se fosse ser exposta na prateleira de uma loja. Essa caixa, ou embalagem do produto, deve conter as principais funcionalidades. A *vision box* é usada, geralmente, no início do projeto para alinhar os conhecimentos sobre o negócio e garantir que a visão do que deve ser o produto foi compreendida, como pode ser visto na figura a seguir.

Figura 6 – Exemplo de *product vision box*

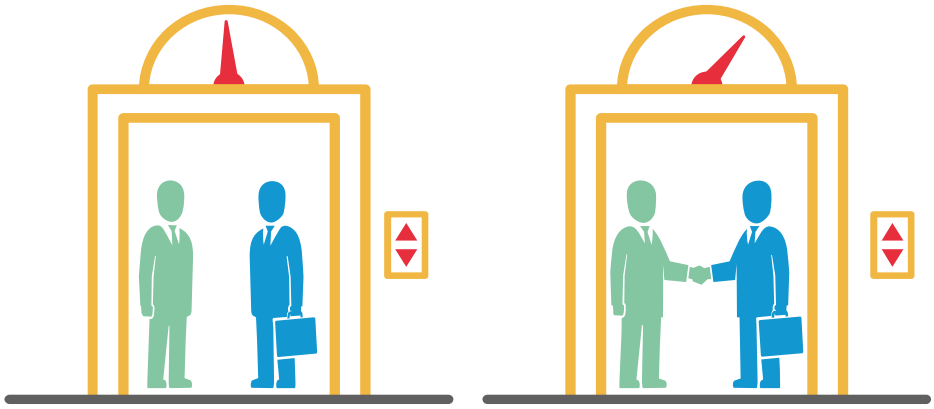



**Fonte:** INFANTE, Ricardo. *Dinâmica vision box*. 23 jan. 2013. Disponível em: https://pt.slideshare.net/ricardoinfante/dinamica-vision-box2013.

## *Elevator speech*

O *elevator speech* é um resumo usado para definir, de forma rápida e simples, um processo, produto, serviço, uma organização ou um evento e a sua proposição de valor. O termo reflete a ideia de que deve ser possível explicar um conceito de forma adequada, apesar de resumida, no tempo de um passeio de elevador, ou seja, no máximo em três minutos.

Figura 7 – Exemplo de *elevator speech*



**Fonte:** Thrive Agency (2017).
Disponível em: https://vemagnet.com/2017/12/11/importance-elevator-pitch-develop-one.

## *Persona*

Também chamada de *perfil dos usuários*, a *persona* é um arquétipo de um usuário do produto.

Dicas para desenvolver personas:

- mantenha as suas *personas* concisas – elas devem ter uma história, características e necessidades próprias;
- separe as *personas* do cliente e do usuário;
- faça com que as suas *personas* sejam críveis – podem ser associadas a imagens, fotos ou desenhos para parecerem mais realistas;
- teste as suas *personas* comparando com a realidade do mercado;
- conecte as *personas* com as histórias de usuário e
- visualize as *personas* e se coloque no lugar delas para entender as necessidades delas.

## História de usuário ou *user story*

*User story* são definições em alto nível dos requisitos do produto final, que incluem apenas informações suficientes para que os desenvolvedores possam produzir estimativas razoáveis do esforço necessário para implementá-las.

Essa é a forma preferida de times ágeis para representar cada um dos itens do *product backlog* (SABBAGH, 2013). Elas tratam das necessidades e dos objetivos de negócios, e são descritas sob o ponto de vista dos usuários do produto – ou das *personas* – e de uma forma concisa, simples e leve. Devem conter:

Como um <QUEM?>, eu quero <O QUÊ?>, para <POR QUÊ?>.

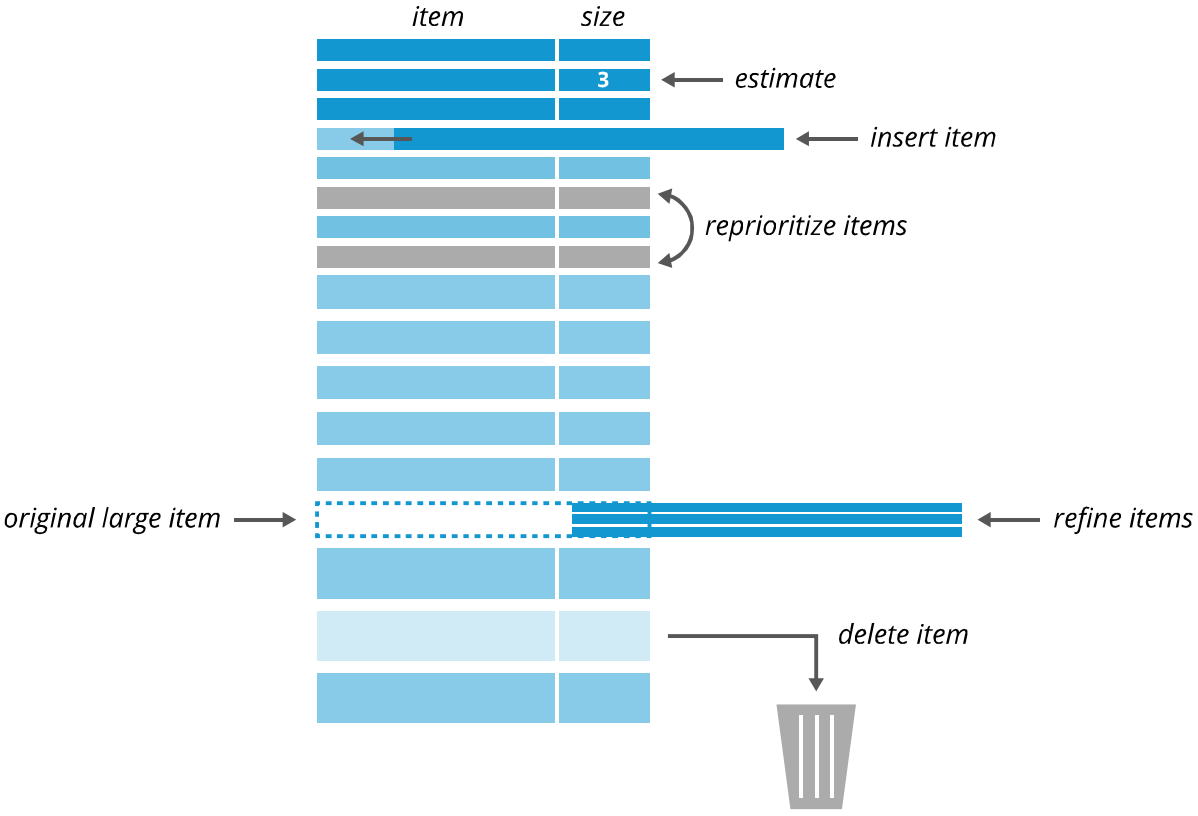
## Refinamento do *backlog*

O refinamento do *backlog* do produto ocorre regularmente e pode ser uma reunião oficialmente agendada ou uma atividade contínua.

Algumas das atividades que ocorrem durante esse aperfeiçoamento do *backlog* incluem:

- remover histórias de usuário que não sejam mais relevantes;
- criar histórias de usuário, em resposta a novas necessidades descobertas;
- reavaliar a prioridade relativa das histórias;
- atribuir estimativas a histórias que ainda não tenham sido estimadas;
- corrigir estimativas à luz das informações recentemente descobertas e
- dividir histórias de usuário que sejam de alta prioridade, mas estejam estimadas de forma superficial – grosseira – para caber em uma próxima iteração.

Figura 8 – Exemplo de refinamento do *backlog*




**Fonte:** Kenneth (2012)

## *Planning poker*

Os membros da equipe – *Scrum team* – podem estimar o esforço de cada item do *backlog* do produto por meio da técnica denominada *planning poker.*

Entre histórias ou itens do *backlog* do produto, a equipe seleciona a que julga ser a mais simples de todas, isto é, a que demandará menos esforço para implementação, e ela será chamada de história de referência, cuja estimativa será o menor ponto do baralho selecionado. As cartas podem ser de baralho comum ou customizadas.

Para cada história de uso, os membros da equipe devem escolher uma carta que julguem ser um valor estimado de esforço e mostrar as suas cartas ao mesmo tempo, para que as suas opiniões não sejam influenciadas pelos outros membros. Caso os valores estejam muito diferentes, os membros defendem os seus pontos de vista e nova rodada de cartas é mostrada até que se chegue a um consenso.

**Curiosidade**

*Story points* representam complexidade, risco, esforço ou tamanho?

Inicialmente, os *story points* foram criados para que se pudesse continuar realizando estimativas necessárias para o planejamento, mas ofuscando o tempo para evitar o compromisso com uma alta precisão e evitar a pressão da gerência sobre estimativas. Eles representam o tempo, levando em consideração parâmetros como complexidade, incerteza e tamanho.

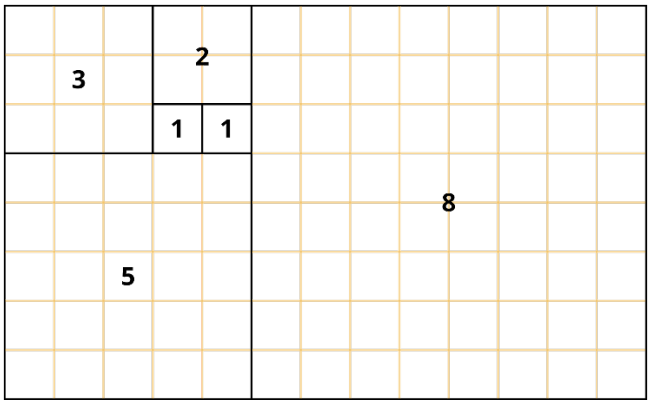
**Fonte:** Sabbagh (2013)

## Escalas para definição de *story points*

Primeiro, deve-se definir a escala de referência; depois, definir o esforço para cada item do *backlog*.

### Sequência de Fibonacci

Figura 9 – Sequência de Fibonacci



**Fonte:** Wikipedia (2020)

Na matemática, a Sucessão de Fibonacci (também Sequência de Fibonacci), é uma sequência de números inteiros, começando normalmente por 0 e 1, na qual, cada termo subsequente corresponde à soma dos dois anteriores. A sequência recebeu o nome do matemático italiano Leonardo de Pisa, mais conhecido por Fibonacci, que descreveu, no ano de 1202, o crescimento de uma população de coelhos, a partir desta. No entanto, essa sequência já era conhecida na antiguidade.

A sequência de Fibonacci tem aplicações na análise de mercados financeiros, na ciência da computação e na teoria dos jogos. Também aparece em configurações biológicas, por exemplo, na disposição dos galhos das árvores ou das folhas em uma haste, no arranjo do cone da alcachofra, do abacaxi, ou no desenrolar da samambaia (WIKIPEDIA, 2020).

## Escalas qualitativas

Também podemos explorar as escalas qualitativas, como nas figuras 10 e 11:

Figura 10 – *T-shirt sizing*: P, M e G, ou PP, P, M, G e GG

**Fonte:** RADIUS ENGINEERING. *Project estimation through T-shirt size*. 23 set. 2017.

Disponível em: https://medium.com/radius-engineering/project-estimation-through-t-shirt-size-ea496c631428.

Figura 11 – Cachorros: poodle toy, cocker spaniel, pastor alemão, etc.

**Fonte**: Dreamstime. ID 28103713.

Disponível em: https://pt.dreamstime.com/fotos-de-stock-grupo-de-assento-marrom-dos-c%C3%A3es-image28103713.

## Velocidade da equipe

Como os esforços para cada história ou item do *backlog* foram mensurados por números – vide *palnning poker*–, é possível, após cada *sprint,* saber qual é a capacidade de realização da equipe, isto é, quantos pontos foram executados. A partir desses números, é possível calcular a velocidade média da equipe e estimar as demais *sprints*.

## Mínimo viável do produto

O mínimo viável do produto (MVP) é o mínimo que deve ser entregue de um produto para que possa ser testado no mercado ou pelo cliente. Se possível, deve ser entregue após a primeira *sprint*.

## Quadro de tarefas

O quadro de tarefas representa o *backlog*. Para o Kanban, todos o itens do *backlog* ficam visíveis em um quadro contínuo. Já para o *Scrum*, apenas o *backlog* da *sprint* fica visível, e todo o quadro deve ser renovado para o próximo ciclo.

Figura 12 – Exemplo de quadro de tarefas


**Fonte:** elaborado pelos autores

**Atenção!**

A equipe sempre deve executar os itens do *backlog* de acordo com a priorização que foi determinada pelo PO, no caso do *Scrum*; e pela equipe ou por um responsável, no caso do Kanban.

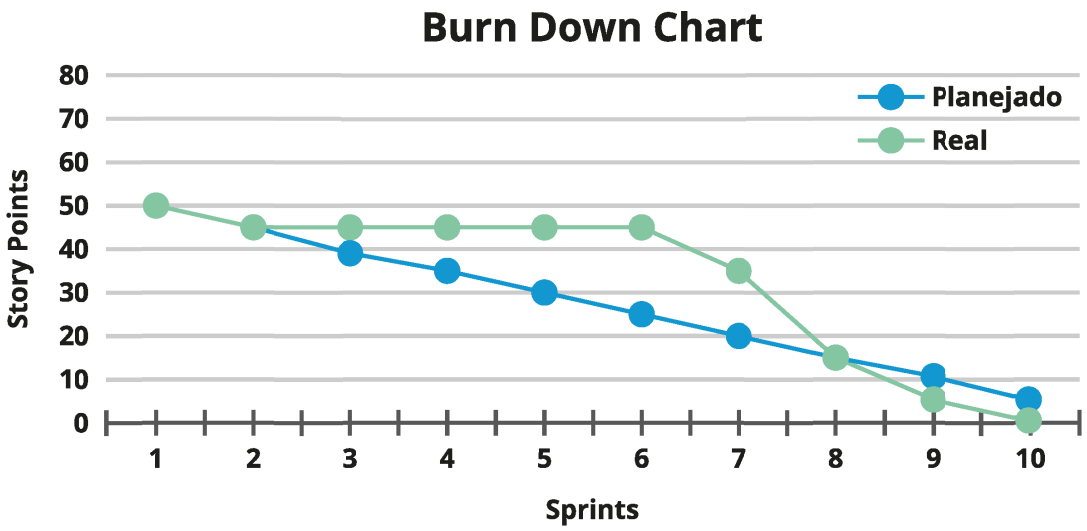
No *Scrum*, mesmo um item estando previsto para uma mesma *sprint,* os itens de baixo só devem estrar em produção quando os de cima estiverem completos.

No Kanban, devem-se respeitar os limites do WIP.

## *Burn down chart*

O gráfico de *sprint burn down* – uma ferramenta que o time de desenvolvimento pode utilizar para monitorar o seu progresso na *sprint* – pode ser atualizado nas reuniões diárias (SABBAGH, 2013). Representa o trabalho restante (eixo y) em função do tempo/*sprints* (eixo x).

Figura 13 – *Burn down chart*: exemplo 1

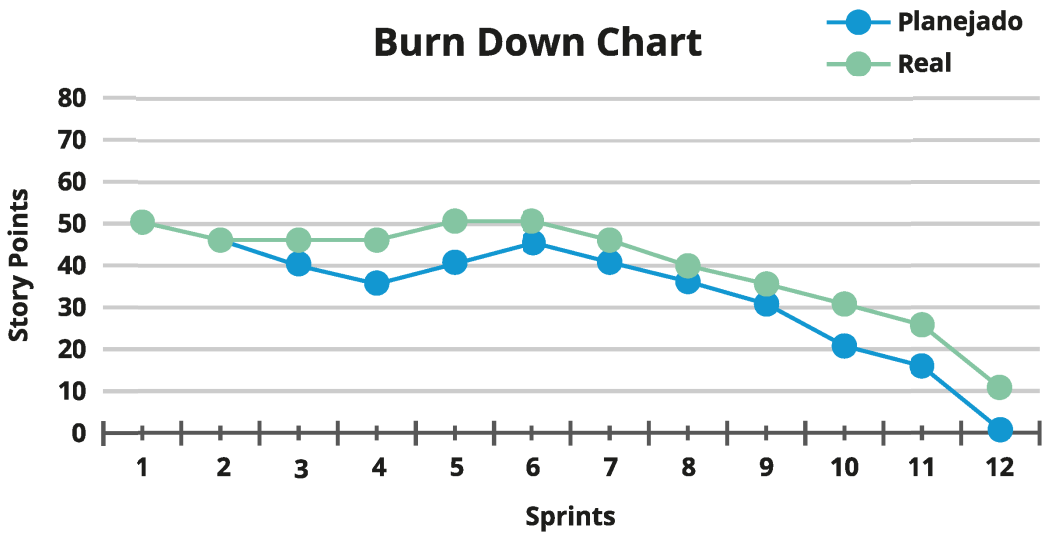



**Fonte:** elaborado pelos autores

**Curiosidade**

Embora o nome seja *burn down*, a curva, em alguns, momentos pode subir. Isso acontece quando o *backlog* do produto sofre um aumento com a inclusão de novas funcionalidades, quando há mudanças nas estimativas dos itens restantes ou quando parte da *sprint* anterior não foi entregue. Veja as *sprints* 5 e 6 do gráfico a seguir.

Figura 14 – *Burn down chart*: exemplo 2




**Fonte:** elaborado pelos autores

## *Burn up chart*

O *burn up chart* possui as mesmas funções que o *burn down chart*. No entanto, ele representa as entregas acumuladas em função do tempo. Sendo assim, a sua curva sobe ao invés de descer.

# MÓDULO III – OUTROS FRAMEWORKS E ESCALADA ÁGIL

Neste módulo, apresentaremos outros *frameworks* ágeis além do *Scrum* e do Kanban. Também discutiremos e exibiremos modelos de escalada ágil na organização.

## Outros *frameworks* ágeis

### Prince2 Agile

Um *mix* da metodologia Prince2®, já reconhecida no mercado, em articulação com as metodologias ágeis. Tem o foco em governança, inovação e colaboração. Pode ser utilizado tanto em projetos de engenharia de *software* como em outras áreas.

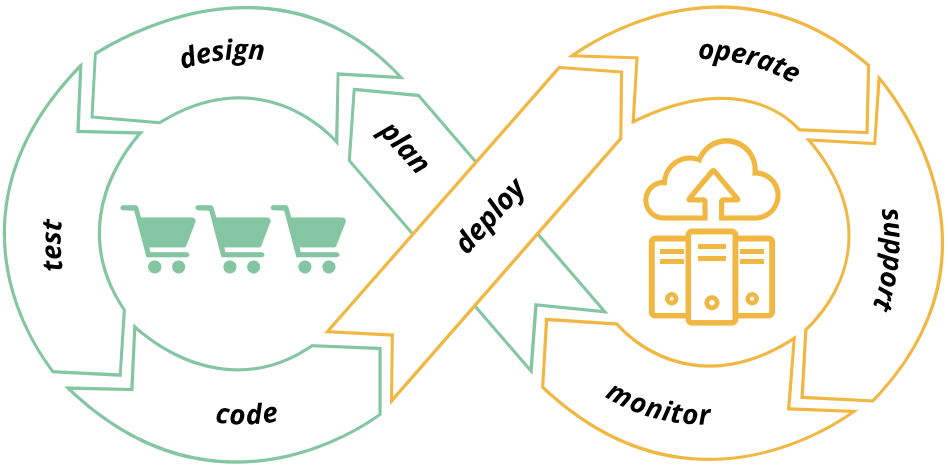
### DevOps

No cenário tradicional de TI, encontramos uma distância muito grande entre as áreas de desenvolvimento e a de operações, o que leva a falhas no processo e muito retrabalho.

DevOps visa promover a integração entre as áreas de desenvolvimento e operações em TI. Não se trata de uma ferramenta, linguagem de programação ou *framework*, mas, sim, de um movimento de comunicação e colaboração com foco no objetivo em comum.

A ideia por trás desse movimento é gerar colaboração e integração entre desenvolvimento e operações, com entrega e *feedbacks* contínuos. Muitas vezes, tais práticas são desenvolvidas por meio da automatização de processos. A figura clássica que representa o DevOps pode ser vista a seguir:

Figura 15 – Representação do DevOps




**Fonte:** Ambler e Lines (2017, p. 64)

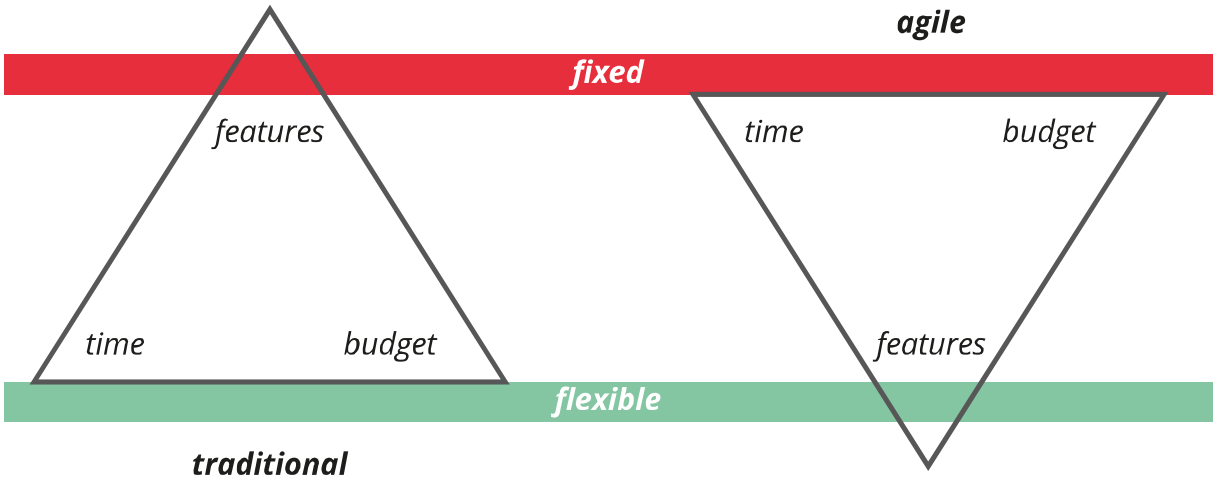
## *Extreme Programming*

A *Extreme Programming* (XP) é bastante prescritiva se comparada ao *Scrum*. Embora seja leve, prescreve um conjunto mais completo de práticas que devem ser seguidas para se obter sucesso no projeto (SABBAGH, 2013). Ela inclui quase tudo do *Scrum* e no Kanban, e mais algumas boas práticas de engenharia bem específicas, como desenvolvimento orientado a testes e programação em par (KNIBERG; SKARIN, 2009). É uma metodologia específica para o desenvolvimento de *softwares*. Foi idealizada para times pequenos e médios, para os quais os requisitos são vagos e o ambiente está em constante mudança. Tem como principais valores: comunicação, simplicidade, *feedback*, coragem e respeito.

## *Dynamic Systems Development Method*

*Dynamic Systems Development Method* (DSDM) é um *framework* que foca a entrega interativa de sistemas de negócios com o uso de *timeboxes* e o envolvimento contínuo de negócios. É composto de oito princípios e uma coleção de técnicas que podem ser usadas em projetos. Uma das suas principais contribuições é a inversão do triângulo tradicional de restrições em gerenciamento de projetos. Enquanto em métodos tradicionais trabalhamos com o escopo fechado, variando custos e cronograma, em agilidade, trabalhamos fixando custos e cronograma, mas com o escopo variável, conforme a figura a seguir.

Figura 16 – DSDM




**Fonte:** adaptado de Morse (2012)

Disponível em: https://www.researchgate.net/figure/The-iron-triangle-in-traditional-and-*Agile-software*-development-Source-Morse-L-2012_fig3_322599612.

Trata-se de um aporte muito importante para a mentalidade ágil como um todo, dada a quebra do paradigma em relação às mudanças de escopo e ao seu fluxo de entrega.

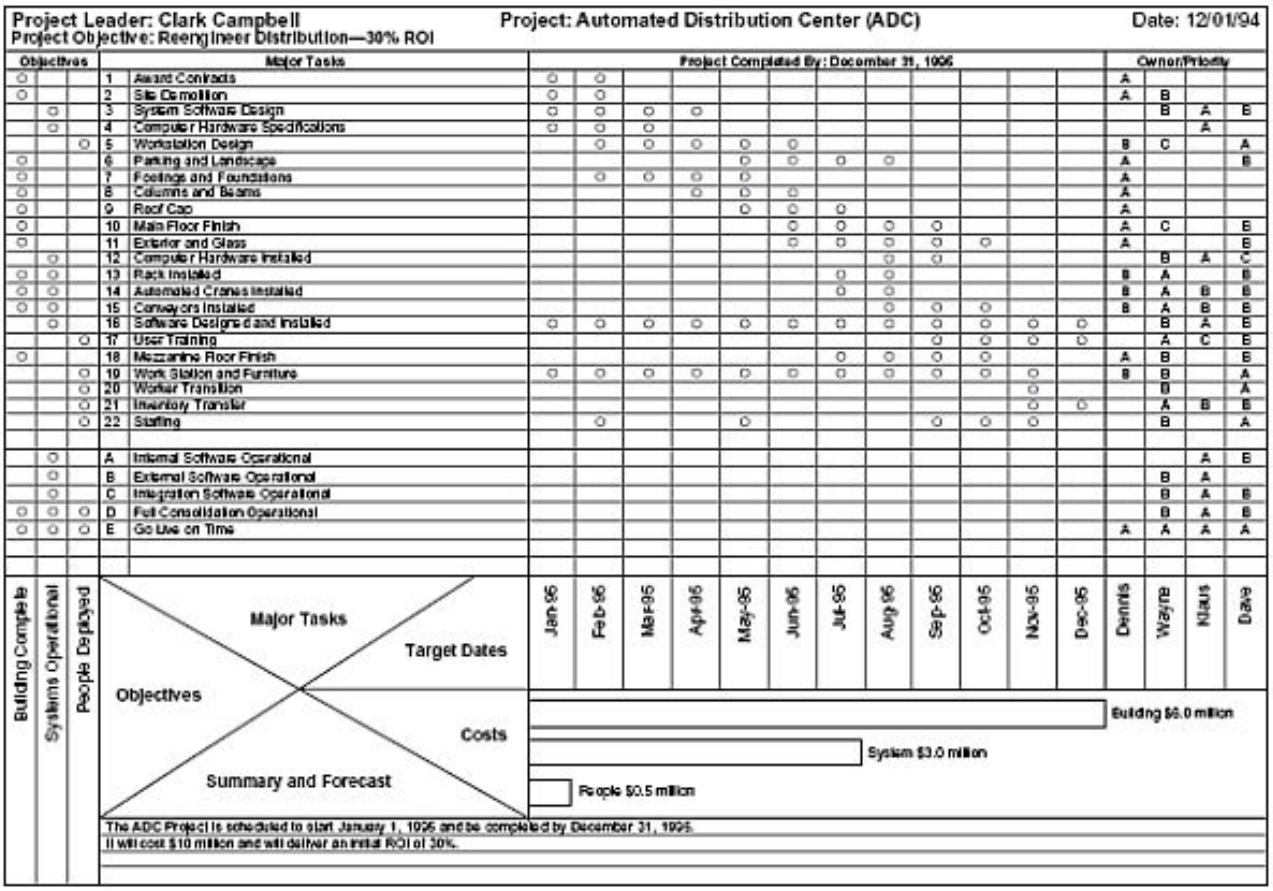
## *Feature Driven Development*

O *Feature Driven Development* (FDD) é um *framework* interativo para o desenvolvimento de *softwares* com foco nas funcionalidades do produto.

## *One-Page Project Management*

*One-Page Project Management* (OPPM), ou gerenciamento de projeto em uma página, foi criado em 2006 por Clark Campbell, que publicou um guia de como aplicar essa técnica, utilizada por ele há mais de 20 anos, para comunicar o andamento do projeto. O OPPM é uma ferramenta de comunicação, não uma metodologia. Apresenta um resumo executivo do projeto em uma folha A4, contendo o que foi planejado e o progresso alcançado (*status*). Não é simples, mas é relativamente direta. O modelo contém cinco partes essenciais: tarefas, proprietários, objetivos, tempo e custo (planejado *versus* executado).

Figura 17 – OPPM

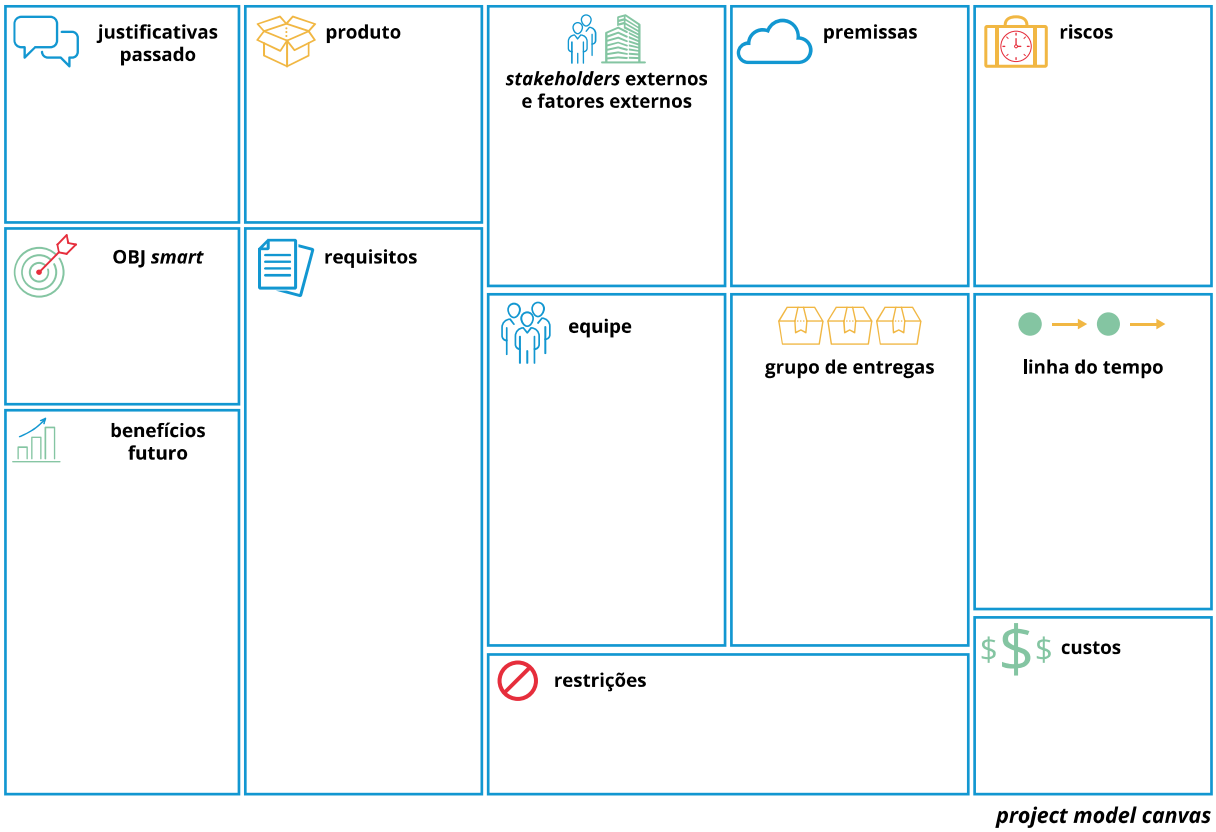



**Fonte:** Campbell (2008)

## *Canvas* para projetos

O modelo de *canvas* também é uma ferramenta de comunicação mais simples do que o OPPM. Pode ser utilizada tanto para facilitar *brainstorming* da equipe antes do planejamento do projeto como para apresentações executivas.

Figura 18 – *Canvas* para projetos




**Fonte:** PMCanvas. Disponível em: http://pmcanvas.com.br/download.

## Escalada ágil

Você já ouviu falar em mundo Vuca? Vuca é um acrónimo em inglês para volatilidade, incerteza complexidade e ambiguidade. Em um ambiente em que muitas empresas fecharam ou faliram por não conseguirem manter-se competitivas, estaremos preparados para evoluir rapidamente para organizações mais ágeis e orgânicas?

Em um ambiente de mudanças constantes, a agilidade é cada vez mais disseminada e aplicada às empresas, que precisam adaptar-se, constantemente, para se manterem competitivas. Por esse motivo, os conceitos de agilidade e *Lean* extrapolam as linhas de produção e o desenvolvimento de produtos e *softwares* para fazer parte da organização. São muitos pontos a serem considerados para manter uma organização competitiva, incluindo a criação de soluções eficientes, respostas rápidas e mudança de mentalidade. Nessas horas, a metodologia ágil é muito útil, principalmente, por incluir as pessoas como centro dessas mudanças.

Escalar o ágil nas empresas requer diferentes ações, que vão depender dos objetivos e da estrutura já existente no ambiente, porque não existem receitas prontas. Existem *frameworks* que servirão de referência e precisam ser adaptados à cultura e aos objetivos da organização. Sem uma

escalada eficiente do ágil, há queda na satisfação do cliente, perda de participação no mercado, falhas em processos e retrabalhos. Quando bem implementada, essa metodologia mantém competitividade e é uma ferramenta primordial para o sucesso.

Não há atalhos em uma jornada ágil! Sustentar o ágil em escala requer uma mudança na estrutura organizacional, nas políticas e no comportamento. Isto é, uma grande mudança de cultura! Como qualquer outra mudança em escala, essa é uma tarefa desafiadora.

O processo de mudança organizacional afeta as pessoas. No caso da escalada ágil, também depende das pessoas, uma vez que as equipes auto-organizadas são a base do conceito ágil. Dessa forma, as pessoas precisam entender a nova maneira de trabalhar. Para que isso aconteça, cada organização deverá criar o seu próprio processo que funcione no seu contexto específico. Dessa forma, as pessoas devem ser envolvidas na mudança, não só no momento da implantação, mas na construção de uma nova cultura. Quando criam os processos, as pessoas identificam o que funciona para elas e, em seguida, uma nova cultura – a forma como fazemos as coisas aqui – vai emergir.

Não é fácil implementar todas essas mudanças culturais pregadas pelo ágil em escala. É preciso que todos estejam comprometidos a melhorar a colaboração e a capacidade de executar projetos de acordo com a estratégia.

Existem diversos *frameworks* para suportar a escala ágil: Scaled Agile Framework®, Scrum@Scale, Disciplined Agile, LeSS, Nexus, Praxis36, que veremos logo a seguir. As diferenças entre diferentes *frameworks* de dimensionamento refletem diferenças filosóficas profundas sobre os principais desafios no escalonamento da agilidade. Identificar o ambiente e as necessidades da organização é determinante para tomar as melhores decisões, criar maior transparência e alinhamento no trabalho.

Em geral, em muitas organizações, o que observamos é a adoção de práticas e ferramentas extraídas das várias estruturas apresentadas a seguir, derivando o seu próprio *framework*.

Assim como as pessoas, as organizações possuem perfis. É muito difícil observar uma organização que mudou, radicalmente, a sua forma de operar. As empresas já nascem com uma cultura. O que observamos no mercado são empresas tradicionais criarem braços – empresas irmãs – já sendo criadas com uma cultura diferente para atender às demandas do mercado, uma vez que, dentro da sua estrutura corporativa, não conseguiriam. Como exemplos, podemos citar a Embraer, que criou o Centro Embraer de Inovação nos Negócios, em Melbourne; e a Fiat, que criou a Future Insights, divisão de pesquisa ao lado da sua fábrica em Betim, Minas Gerais.

## Scaled Agile Framework®

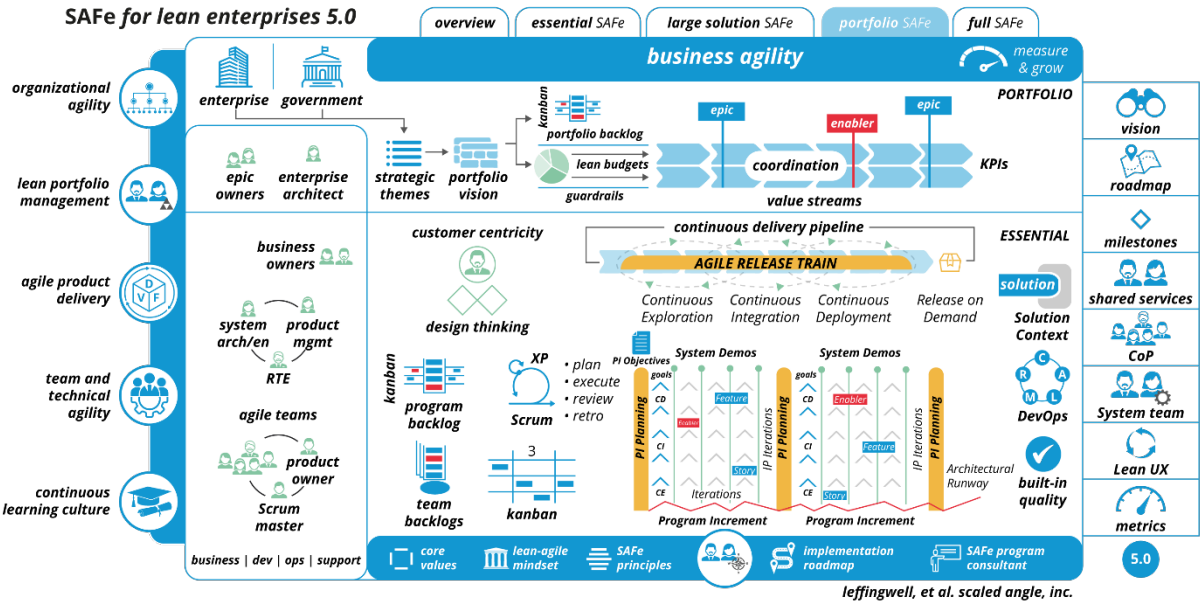
O Scaled Agile Framework® (SAFe®) é um conjunto de padrões – melhores práticas – organizacionais e de fluxo de trabalho. Ele ajuda a implementar práticas ágeis em escala corporativa, provendo um direcionamento nos níveis de portfólio, programas e times de projetos. Está sustentado sobre os pilares de desenvolvimento ágil de *software*, desenvolvimento enxuto – *Lean* – de produtos e pensamento sistêmico.

O SAFe® tenta adaptar-se ao máximo à organização para ser minimamente disruptivo e não provocar conflitos durante a sua implementação na organização.

Os princípios do SAFe® (REAINTHONG, 2020) são:

- tenha uma visão econômica;
- aplique o pensamento sistêmico;
- assuma a variabilidade, preservando as opções;
- crie de forma incremental com ciclos de aprendizado integrados e rápidos;
- avalie o sistema de trabalho de forma objetiva, utilizando marcos;
- visualize e limite o trabalho em andamento, reduza tamanhos de lote e gerencie comprimentos de fila;
- aplique cadência, sincronize com o planejamento integrado;
- motive os profissionais do conhecimento e
- descentralize a tomada de decisão.

Figura 19 – SAFe®: exemplo



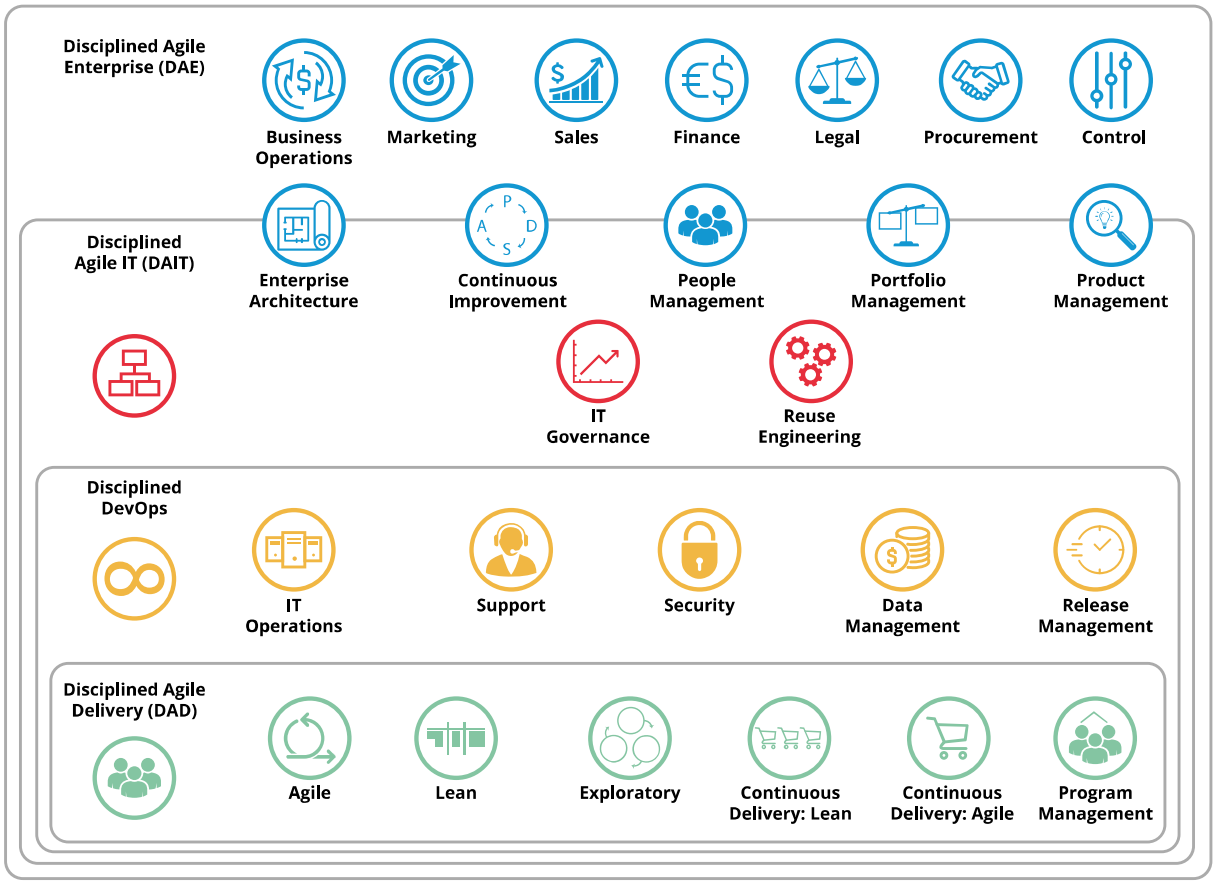

**Fonte:** *Scaled Agile Framework* (2020)

## *Disciplined Agile*

O *Disciplined Agile* (DA), *framework* de decisão, é uma estrutura que fornece orientação para ajudar organizações a simplificar os seus processos, de maneira sensível ao contexto, dando uma fundação sólida para a agilidade nos negócios. O DA foi criado para ajudar pessoas e equipes a fazer escolhas informadas, melhorar a maneira de trabalhar e ajudar as empresas a simplificar estratégias ágeis e flexíveis para a tomada de decisões melhores.

O DA Consortium foi adquirido em 2019 pelo PMI, sendo que, como o PMBoK, possui um bom *Body of Knowledge* (BoK). A aplicação do *kit* de ferramentas DA possibilita que as empresas personalizem qualquer método ou estrutura – como tradicional, *Scrum* ou SAFe® – para gerar resultados que os diferenciem dos seus concorrentes. A título de facilitar a visualização, no DA, os níveis de escalada estão representados na figura a seguir:

Figura 20 – Níveis de escalada



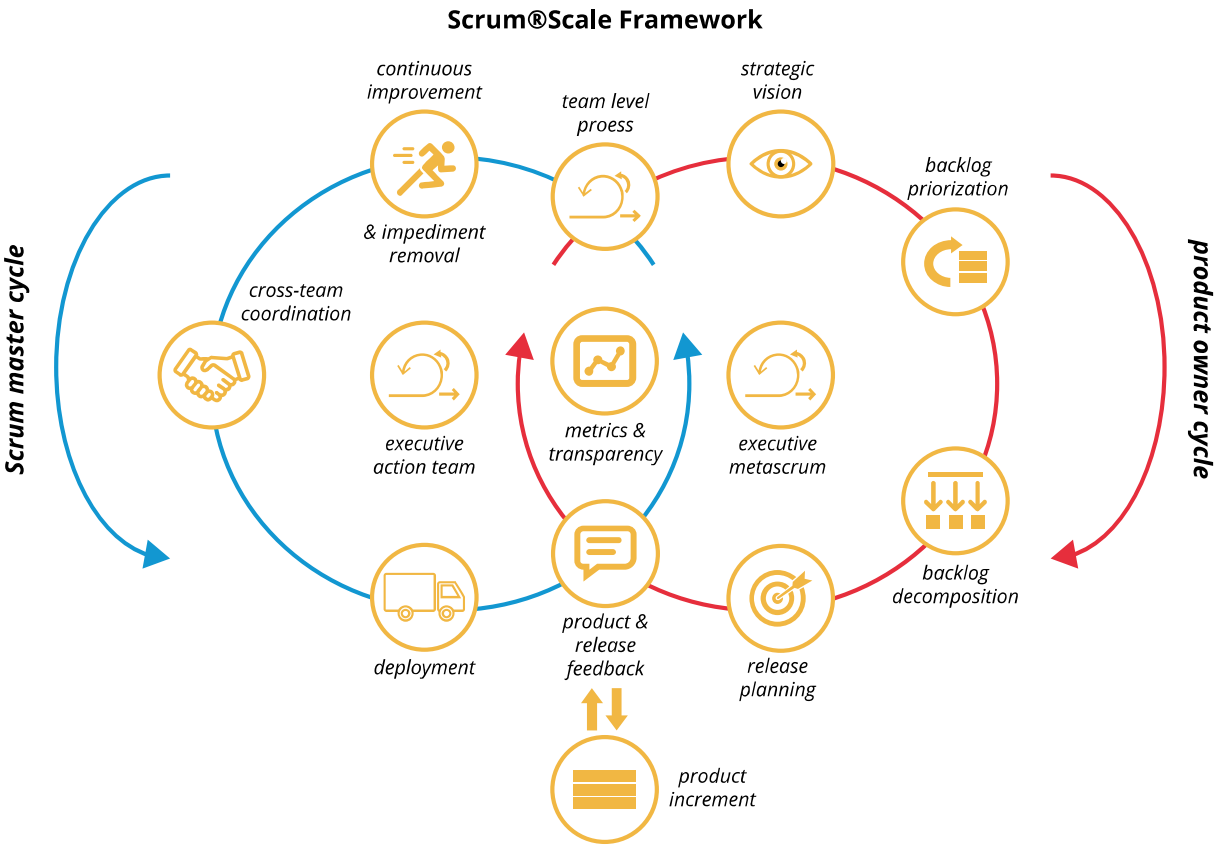

**Fonte:** Ambler e Lines (2017, p. 45)

O nível de *Disciplined Agile Delivery* (DAD) é o que lida com diferentes tipos de ciclo de vida de gerenciamento. O modelo é completamente agnóstico no que diz respeito ao tipo de *framework* a ser utilizado: *Lean*, *Scrum*, etc., contanto que seja bem escolhido em função do contexto e que gere, de fato, valor ao cliente.

## Scrum@Scale

*Scrum*, como originalmente apresentado no *Scrum Guide*, é um *framework* para desenvolvimento, entrega e suporte a produtos complexos por um único time. Scrum@Scale (S@S) foi criado para coordenar, de maneira eficiente, times que precisem otimizar a estratégia geral da organização, criar produtos, serviços ou sistemas de maneira coordenada entre eles (SUTHERLAND, 2019). Geralmente, é adotado por companhias que já utilizaram o *Scrum* com sucesso. O seu objetivo é alinhar diversos times das organizações em torno de objetivos comuns (SUTHERLAND, 2019). O seu *framework* é um conjunto mínimo de recursos que permite inspeção e adaptabilidade por meio de transparência com objetivo de impulsionar a inovação, a satisfação do cliente, o desempenho e a satisfação da equipe (SUTHERLAND, 2019).

Figura 21 – S@S



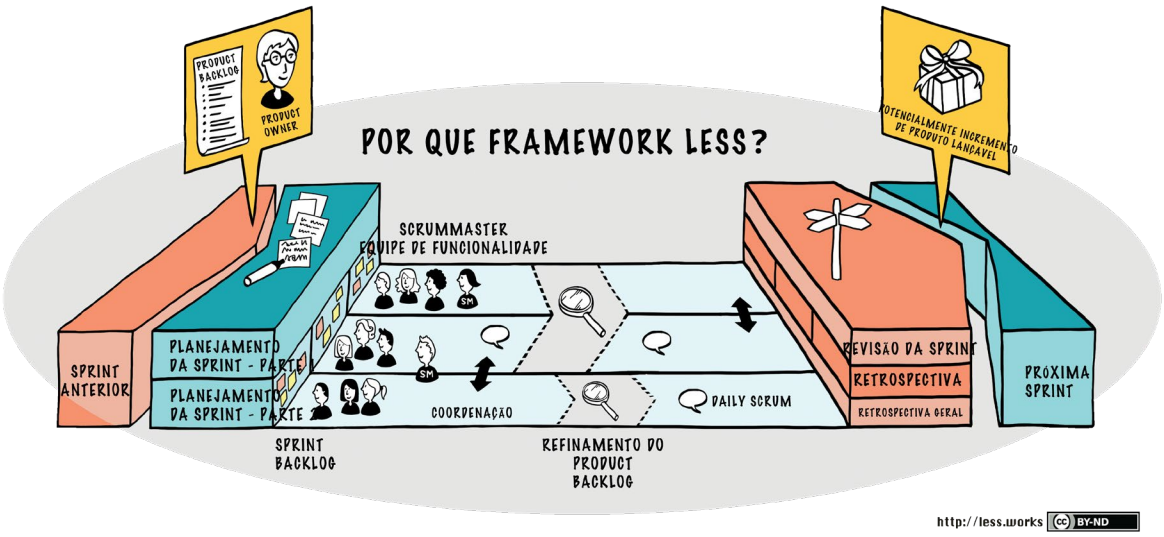

**Fonte:** Sutherland (2019)

## *Large-Scale Scrum*

*Large-Scale Scrum* (LeSS) é uma versão escalada de uma única equipe *Scrum*, mantendo muitas das práticas e ideias de uma única equipe *Scrum*. No *framework* LeSS, todas as equipes estão trabalhando em uma mesma *sprint* para entregar um pacote de produto ao final de cada *sprint*. Como semelhanças do *Scrum*, podemos observar:

- um único *product backlog*, porque se trata de um único produto;
- uma única definição de *pronto* para todas as equipes;
- um pacote de incremento de produto ao final de cada *sprint*;
- um único PO;
- muitas equipes completas e multifuncionais e
- *sprints* de cadência única e em comum para os diversos times.

Figura 22 – LeSS

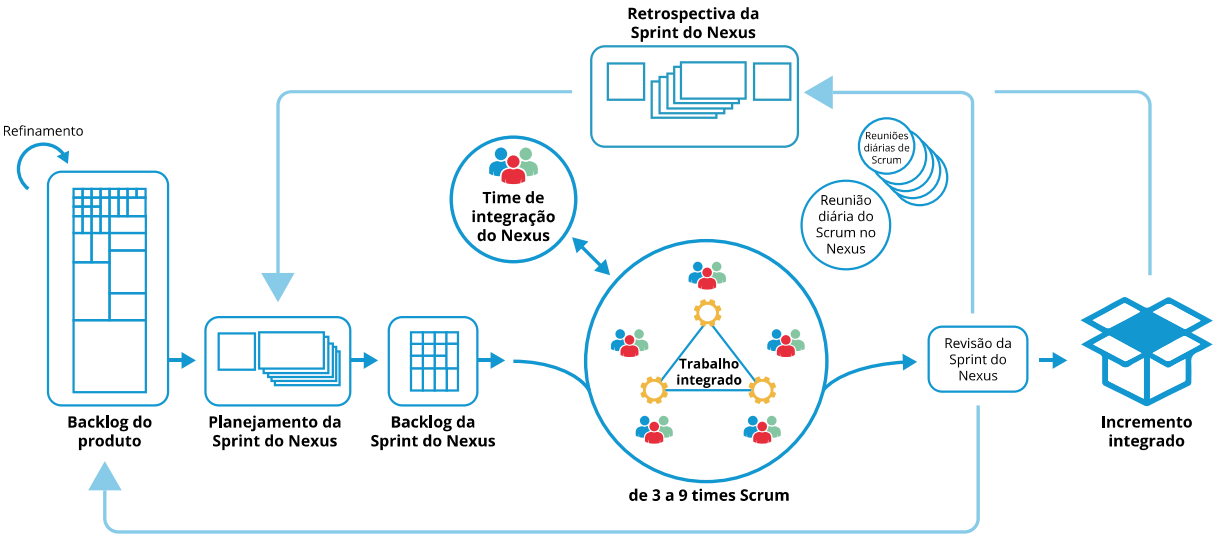



**Fonte:** THE LESS COMPANY B.V. *LeSS framework*. Disponível em: https://less.works/pt/less/framework/index.html.

## Nexus

Nexus é um *framework* constituído de papéis, eventos, artefatos e regras que os unem e entrelaçam junto ao trabalho de aproximadamente três a nove times *Scrum* em um único *backlog* do produto para construir um incremento integrado que alcance uma meta (SCHWABER, 2018). Não tenta resolver o problema da organização, em vez disso, concentra-se no desenvolvimento de produtos grandes com várias equipes trabalhando juntas.

Figura 23 – Nexus




**Fonte:** Schwaber (2018)

# MÓDULO IV – AGILE COACHING

Neste módulo, conheceremos os fundamentos de *agile coaching*. Para isso, teremos como principais conceitos: *agile coaching, agile softskills, agile mindset* e *management* 3.0.

## O que é *agile coaching*

> *Coaching* é um processo de transformação e aperfeiçoamento que possibilita aos indivíduos, times ou empresas, desenvolver as suas mais variadas capacidades para permitir que alcancem o mais rapidamente possível os seus próprios objetivos (BARBOSA, 2018).

A palavra *coach* vem da língua inglesa e significa treinador. É aquele que pode ajudar um indivíduo a identificar o seu estágio atual de desenvolvimento em certa área e auxiliar na definição de objetivos e planos para conseguir alcançá-los. O *coaching* não deve influenciar o indivíduo treinado nas suas decisões. O seu papel é apenas estimular o treinado a agir, a realizar ações em direção aos seus próprios objetivos, metas e desejos.

Nesse contexto, o objetivo do *agile coach* é desenvolver pessoas, times e organizações ágeis para que pensem por si mesmas, sem a necessidade de seguir algum protocolo já existente ou um modelo fixo. O trabalho do *agile coach* é guiar as pessoas, bem como ajudá-las a identificar e superar os desafios da mudança, até que sintam segurança para seguir os seus próprios caminhos (BARBOSA, 2018).

Com o objetivo de atualizar o modelo de gestão das empresas, tornando-as mais competitivas, flexíveis e preparadas para o novo mundo, algumas organizações buscam uma *transformação ágil.*

A transformação ágil é uma abrangente e significativa mudança organizacional guiada, motivada pela adoção e pelo exercício de novas práticas de *comportamentos ágeis*, os quais renovam hábitos e processos, quebram paradigmas, reconfiguram os modelos de gestão, liderança e relacionamento, impactando pessoas, processos, estruturas, cultura, clientes e estratégias do negócio simultaneamente (BARBOSA, 2018).

Para isso, ainda segundo Barbosa (2018), é necessário:

- adotar novas formas de pensar e agir;
- readequar as estratégias dos produtos e negócios;
- aderir aos novos modelos de gestão;
- requintar as práticas de liderança e relacionamento;
- promover uma profunda renovação da cultura organizacional e
- admitir, combinar e praticar os métodos ágeis.

A transformação ágil não é simples e, para que ela aconteça, o *agile coach* pode ser a pessoa que vai motivar as equipes, facilitando a identificação de objetivos e a montagem de planos de ação para alcançá-los. pode ser um agente externo, um especialista contratado pela organização ou um agente interno convocado pelo líder.

Como a transformação ágil deve ser adequada e customizada para cada empresa, a melhor maneira de começar é entendendo a realidade atual da organização. Para esse entendimento, ninguém melhor do que o time para dar informações! O *agile coach* deve começar o seu trabalho por aí, com um *brainstorming* com o grupo, entrevistas individuais e observando, atentamente, as rotinas da organização, servindo de mediador para o alcance das mudanças.

As possíveis barreiras a serem enfrentadas durante a transformação ágil são bloqueadores técnicos, organizacionais ou comportamentais. O *agile coach* deverá estar atento a elas e auxiliar o time a buscar soluções.

O trabalho de um *agile coach* não deve ser eterno, deve ter um prazo final e apesentar as mudanças incorporadas. Afinal, o objetivo do *agile coach* não é virar muleta, mas fortalecer as pessoas e os times para que possam caminhar sozinhos após a conclusão do processo (BARBOSA, 2018).

## *Agile softskills*

Uma função importante do *agile coach* é auxiliar o desenvolvimento das *softskills* do time. Os times ágeis precisam ser estáveis, multifuncionais, pequenos, integrados, dedicados e alinhados. Além disso, devem exercer auto-organização, colaboração, responsabilidade, generosidade, cuidado e comunicação. Os times ágeis devem ter plena consciência das competências coletivas e individuais de que necessitam para melhor desempenhar o seu trabalho (BARBOSA, 2018).

O trabalho do *agile coach* deve ser constituído de vários ciclos, como o PDCA da qualidade: *plan, do, check, act* ou, em português, planejar, fazer, verificar e atuar corretivamente. Isto é, o *coach* deve auxiliar o time a identificar as suas necessidades e metas, planejar como alcançá-las, checá-las por meio de indicadores previamente assumidos e tentar, novamente, agindo corretivamente, caso seja necessário. Isso tudo em um ciclo crescente de aprendizagem e desenvolvimento.

## *Agile mindset*

A mudança de uma cultura tradicional para uma cultura ágil não é nada simples. O desafio da mudança de cultura não é fácil. Sendo assim, é importante verificar a necessidade para a organização e os benefícios que serão alcançados.

*Scrum* é um *framework* para desenvolver, entregar e manter produtos complexos (SCHWABER; SUTHERAND, 2017), mas sob qual conceito de complexidade? O que podemos chamar de projetos complexos? Vejamos o quadro a seguir:

Quadro 3 – Complexidade

| categorias | causas associadas |
|---|---|
| comportamento humano | comportamento individual;<br>comportamento do grupo, da organização e político;<br>comunicação e controle e<br>desenvolvimento e desenho organizacional |
| comportamento do sistema | complexidade do produto do projeto, entre outros, e problemas técnicos e de *design* |
| ambiguidade | incerteza e emergência |

**Fonte:** PMI (2014)

Segundo Sargut e Mcgrath (2011), em termos de propriedades, três delas determinam a complexidade de um ambiente:

- multiplicidade – refere-se à quantidade de elementos potencialmente interagindo;
- interdependência – refere-se à forma como esses elementos estão conectados e
- diversidade – refere-se ao nível de heterogeneidade desses elementos.

Desse modo, quanto maior a multiplicidade, a interdependência e a diversidade, maior é a complexidade. Podemos citar exemplos de projetos complexos:

- construção de uma barragem;
- projetos de pesquisa de materiais ou medicamentos;
- Eurotúnel e
- mudança de dinheiro físico para dinheiro criptografado.

Cada um deles pode ser classificado como complexo de acordo com as abordagens do PMI (2014) ou de Sargut e Mcgrath (2011).

**Exemplo – Projeto de pesquisa de um novo medicamento**

De acordo com a classificação do PMI (2014):

- **comportamento humano** – cada pessoa reage de maneira diferente a um medicamento. Muitos placebos funcionam tão bem como o medicamento em teste, indicando que o fator emocional também influencia os resultados;
- **comportamento do sistema** – interações medicamentosas podem alterar resultados de pesquisas. A qualidade dos equipamentos e os recursos humanos utilizados no projeto também podem alterar o resultado e
- **ambiguidade** – total incerteza dos resultados da pesquisa e, para agravar, se a doença estiver em fase epidêmica, poderá haver uma urgência.

No entanto, será que todos eles podem ser tratados com uma cultura ágil? Para cada tipo de complexidade, o tratamento também será diferenciado. A seguir, o quadro apresenta os fatores de complexidade e as competências a serem tratadas:

Quadro 4 – Fatores de complexidade e competências

| complexidades e os seus fatores | competência a confrontar |
|---|---|
| **estrutural** | |
| falta de compreensão do objetivo do projeto | alinhamento dos objetivos do projeto com os da organização |
| diversidade de disciplinas envolvidas | capacidades (organizacional) |
| coordenação de múltiplas atividades | planejamento (por exemplo: Pert, CPM, etc.); sistema de controle de desempenho |
| desconhecimento; incompatibilidade entre requisitos e subcomponentes | gerenciamento de qualidade e de configuração |
| ocorrência de evento que cause dano ao projeto | gerenciamento de riscos |
| mudanças de objetivo | sistema de controle de alterações |
| **político-social** | |
| agendas conflitantes (aversão ou leniência na adesão ao projeto) | desenvolvimento de relacionamentos; liderança |
| motivação da equipe e parceiros | *motivational fator inventory*; incentivos ligados ao desempenho |
| cultura organizacional avessa à cultura de projetos | *fit* cultural |
| interesse pessoal sobrepujando objetivo do projeto | sistema de governança |
| **emergente** | |
| múltiplas opções para a solução de um problema sob incertezas | pensamento estratégico; gestão do conhecimento |
| diferenciação da concorrência; necessidade de obter soluções originais | colaboração e cocriação; inadequação do quadro conceitual inicial (QCI) |
| fronteiras fluidas (organizações temporárias) | desenvolvimento de relacionamentos; transferência de conhecimento |
| mudança crítica e inesperada por motivações do contexto | atenção ao ambiente do projeto; flexibilidade e adaptabilidade |
| surgimento de problema para o qual não há referência para solução | ambidestria (ampliar a eficiência e inovar) |

**Fonte:** adaptado de Vianna Jr. (2015)

A emergente, que é mais comum tratarmos com a metodologia ágil, diz respeito aos ambientes de projetos incertos, onde a tecnologia, o mercado e as respostas dos consumidores são incertas. Acontece também com projetos inovadores, onde a equipe não tem experiência e o futuro do projeto é cheio de incertezas. Neste caso, devemos planejar a curto prazo e testar os resultados. Vemos muito isso em projetos de *softwares* e de aplicativos.

A complexidade político-social é observada em ambientes onde há muito interferência de fatores humanos, comportamentais e culturais. Como exemplo, projetos governamentais, onde, além do resultado para a população, o gerente do projeto precisa estar atento à fatores políticos e eleitorais. A complexidade se dá no trato com os *stakeholders* e a comunicação do projeto.

Já a complexidade estrutural está ligada à complexidade do escopo do projeto, ao entendimento deste pelas partes interessadas, ou pela diversidade de disciplinas envolvidas. Neste caso, o gerenciamento preditivo do projeto se torna mais efetivo, pois esclarece o escopo, alinha o entendimento e antecipa as necessidades e as dificuldades do projeto.

Podemos perceber que alguns fatores de complexidade, como incerteza, volatilidade, inovação e criatividade, adequam-se mais a uma cultura ágil; outros, como os político-sociais e estruturais, a uma cultura tradicional ou, quem sabe, híbrida. Normalmente, a metodologia ágil é mais bem adequada a fatores de complexidade emergentes, mas isso não é restritivo!

No entanto, alguns pontos são relevantes para a decisão sobre qual metodologia utilizar:

- tipo de projeto e necessidade do cliente;
- tamanho e complexidade do projeto;
- cultura da empresa e cultura do cliente (no meio ágil, o cliente precisa estar disposto a colaborar, constantemente, durante o projeto);
- grau de mudança esperado no escopo, que pode ser determinado pelo nível de informações conhecidas e o grau de inovação do projeto;
- para um gerenciamento ágil do projeto, não é necessário um desenvolvimento linear, e as entregas parciais devem gerar benefícios;
- nível de integração do trabalho da equipe desejado;
- orçamento – fixo ou variável – e
- tempo para planejar ou não, ou possibilidade de realizar testes frequentes.

A mudança do *mindset* pode ser feita na empresa como um todo ou apenas em uma área específica. Após a decisão, deve-se planejar como enfrentar as principais barreiras de mudança de *mindset*:

- **Equipe** – um time maduro, pequeno e auto-organizado é um dos pilares do *mindset* ágil. É importante que a empresa esteja preparada para isso. Caso contrário, mesmo que a equipe seja empoderada, ficará perdida, sem direcionamento, o que comprometerá o resultado. A liberdade virá junto com uma alta carga de responsabilidade. Na mudança de *mindset*, os principais problemas enfrentados relacionados às equipes são:

- falta de confiança e medo de expor as suas fragilidades;
- medo do conflito mantendo um clima de harmonia artificial;
- falta de comprometimento, time sem direção e sem prioridades convergentes;
- fuga da responsabilidade e
- desatenção aos resultados.

- **Contratos** – é preciso pensar nos contratos tanto com os clientes como com os fornecedores. Os clientes deverão entender que as constantes mudanças são bem-vindas. No entanto, alterações no backlog do produto terão como consequência alterações em tempo e custo. Também é importante que estejam abertos e interessados em participar das entregas, dando feedback e contribuindo para a construção do produto final. Quanto aos fornecedores, é preciso avaliar se estarão aptos a fazer entregas com o planejamento de curto prazo.
- **Cultura corporativa** – é normal, dentro de uma organização, que as pessoas tenham um pensamento em comum. Claro, os valores da empresa foram levados em consideração, desde o momento da contratação, e reforçados nos treinamentos, processos e hábitos até a data. Uma mudança requer tempo e muita, muita, muita comunicação.

**Importante**

Não existe uma receita que sirva para todas as organizações e para todas as equipes. É preciso que cada organização ou equipe encontre o seu próprio caminho.

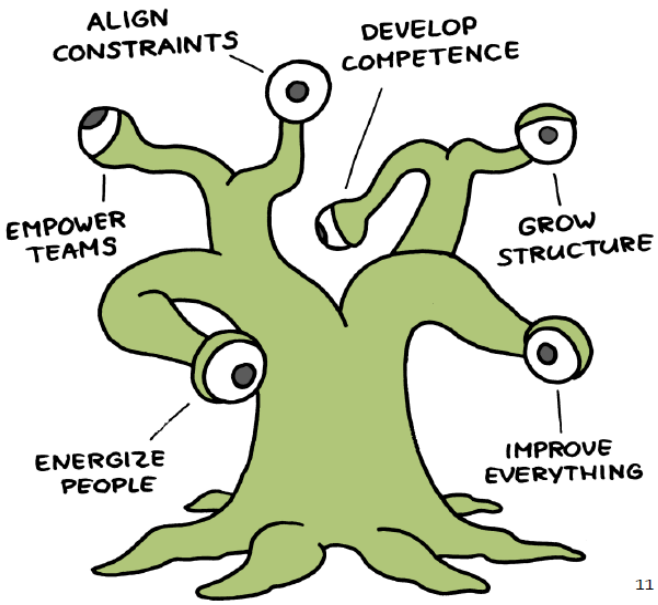
## *Management* 3.0

*Management* 3.0 é um *mindset* que combina uma gama de jogos, ferramentas e práticas para melhorar a liderança nas organizações, com a visão não nos indivíduos, mas no sistema como um todo (MANAGEMENT 3.0, 2020). Nesse sentido, não só um gerente tem responsabilidades, os resultados do trabalho devem ser de responsabilidade de todos. *Management* 3.0 tem como objetivos tornar líderes melhores, aumentar a produtividade, inovar de forma natural, motivar o time, mudar culturas, implantar agilidade e aumentar satisfação no trabalho. É um movimento de inovação, liderança e gestão.

Foi criado pelo holandês Jurgen Appelo, que também criou a imagem a seguir, a qual representa os princípios do *Management* 3.0:

- ser prazeroso para os envolvidos;
- melhorar tudo;
- engajar pessoas e
- gerenciar o sistema, não pessoas.

Figura 24 – Princípios do *Management* 3.0




**Fonte:** Cultura Ágil.
Disponível em: https://www.culturaagil.com.br/management-3-0-o-metodo-agil-de-gestao-empresarial.

A gestão passa a ser uma responsabilidade do grupo, trabalhando junto para encontrar a forma mais eficiente para uma empresa atingir os seus objetivos, mantendo a felicidade dos trabalhadores como uma prioridade.

Espera-se que a eficiência possa ser aumentada por meio da formação de times ágeis empoderados, inovadores, motivados e trabalhando em estruturas que permitam a comunicação eficiente em todas as camadas da organização. As seis visões do *Management* 3.0 são descritas a seguir:

1. **Energizar as pessoas** – como as pessoas são a parte mais importante das organizações, os seus gerentes devem fazer o possível para mantê-las ativas, criativas e motivadas. Para isso, faz-se necessário entender a diferença entre as diversas pessoas, os tipos de motivação intrínseca e extrínseca e identificar o que as motiva à sua volta;
2. **Empoderar times** – gerentes devem entender como times auto-organizados funcionam, quais são os desafios envolvidos e como construir relações de confiança mútua que levem a times auto-organizados de sucesso;
3. **Alinhar restrições** – gestores devem entender a diferença entre restrições e regras, e direcionar os times para que possam trabalhar de maneira auto-organizada, seguindo os objetivos propostos pelos gestores. Quando não há uma meta ou propósito claro, o resultado pode não ser o esperado ou difuso;
4. **Desenvolver competências** – times só conseguem atingir um bom desempenho quando estão habilitados para os seus trabalhos e se sentindo capazes o suficiente. Sendo assim, os gestores devem contribuir para o desenvolvimento das competências necessárias dos membros do time;

5. **Desenvolver a estrutura organizacional** – o modelo de estrutura organizacional impacta, significantemente, o modo de funcionamento da organização. Vários times operam no contexto de uma organização complexa, por isso é importante considerar uma estrutura que privilegie a comunicação. Gestores precisam entender a diferença entre times funcionais e multifuncionais, bem como conhecer técnicas para desenhar áreas que funcionam e se comunicam com eficácia e
6. **Melhorar continuamente** – é um dos maiores desafios nas organizações. Pessoas, times e organizações precisam melhorar continuamente. Na prática, isso significa que gestores e líderes devem ser agentes de mudança.

Uma das diversas questões abordadas por Appelo, na sua obra, diz respeito aos chamados *moving motivators*, relacionados ao nosso autoconhecimento e às nossas motivações intrínsecas, conforme mostra a figura a seguir:

Figura 25 – *Moving motivators*


**Fonte:** Rodrigues (2019)

Normalmente, todos possuímos esses fatores em menor ou maior grau, dependendo do contexto ou do momento vivido. As cartas podem ser utilizadas de várias formas, de acordo com a necessidade: desde um exercício de autoconhecimento que sugere colocar em sequência, do menor para o maior fator motivador, até um estudo de estado futuro, reorganizando as cartas em função de um objetivo, como terminar um curso, arrumar um novo emprego, etc.

Até mesmo em uma reunião de retrospectiva, o uso das cartas pode ser interessante para que o time se conheça melhor. Trata-se de um trabalho reflexivo, mas que pode gerar uma ordenação das cartas pelo time, gerando maior integração.

# BIBLIOGRAFIA


ACADEMY'S, T. K. *What is the praxis project management framework*? Disponível em: https://www.theknowledgeacademy.com/ag/courses/praxis-framework-training/what-is-the-praxis-project-management-framework. Acesso em: fev. 2020.

AGILE ALLIANCE. *Agile 101*. Disponível em: https://www.*Agile*alliance.org/*Agile*101. Acesso em: fev. 2020.

AMBLER, Scott W.; LINES, Mark. *Introdução ao Disciplined Agile Delivery: A Pequena Jornada de um Time Ágil do Scrum ao Continuous Delivery*. CreateSpace Independent Publishing Platform, 2017.

BARBOSA, A. *A essência do Agile coach*. 2. ed. [S.l.]: Agile Institute Brazil, 2018.

BECK, K. et al. *Manifesto Ágil*, 2001. Disponível em: http://*Agile*manifesto.org/iso/ptbr/manifesto.html. Acesso em: fev. 2020.

BOEG, Jesper. Kanban em 10 Passos. *InfoQ Brasil*, 2011. Disponível em: https://doi.org/10.1080/0951192X.2011.579167. Acesso em: fev. 2020.

CAMPBELL, Clark A. The one-page project manager. *MPUG*, 20 mar. 2008. Disponível em: https://www.mpug.com/articles/the-one-page-project-manager.

COCKBURN, A. *Agile software development*: the cooperative game. 2. ed. MA, USA: Addison-Wesley, 2007.

HIGHSMITH, J. *Agile project management*: creating innovative products. MA, USA: Addison-Wesley, 2004.

KENNETH S. Rubin. The importance of the product backlog on a scrum development project. *InformIT*, 25 jul. 2012. Disponível em: https://www.informit.com/articles/article.aspx?p=1928232&seqNum=4.

KNIBERG, H.; SKARIN, M. *Kanban e Scrum*: obtendo o melhor de ambos. 2009. Disponível em: https://fdocumentos.tips/document/kanban-e-scrum-obtendo-o-melhor-de-ambos.html. Acesso em: fev. 2020.

LARMAN, C. *Agile and iterative development: amanager's guide.* MA, USA: LIKER, 2003.

MANAGEMENT 3.0. *What management 3.0 is about*. Disponível em: https://management30.com/learn. Acesso em: fev. 2020.

PMI. *Navigating complexity*, 2014.

REAINTHONG, Tyler. Scaled Agile Framework (SAFe). *ProjectManagement.Com*, 16 fev. 2020. Disponível em: https://www.projectmanagement.com/wikis/312951/Scaled-*Agile*-Framework—SAFe. Acesso em: fev. 2020.

RODRIGUES, Luiz. Moving motivators em terras portuguesas: reflexões em equipa. *Knowledge21*, 2019. Disponível em: https://knowledge21.com.br/blog/moving-motivators-reflexoes-equipa. Acesso em: fev. 2020.

SABBAGH, R. *Scrum*: gestão ágil para projetos de sucesso. São Paulo: Casa do Código, 2013.

SARGUT, G.; MCGRATH, R. G. Learning to live with complexity. *Harvard Business Review*, p. 69-76, 2011.

SCALED Agile Framework. *SAFe 5 for lean enterprises*. Disponível em: https://www.scaled*Agile*framework.com. Acesso em: fev. 2020.

SCHWABER, K. *Guia do Nexus.* O guia definitivo para escalar o *Scrum* com o Nexus: as regras do jogo. 2018. Disponível em: https://scrumorg-website-prod.s3.amazonaws.com/drupal/2018-01/2018-Nexus-Guide-Portuguese-Brazillian.pdf. Acesso em: fev. 2020.

_____; SUTHERAND, J. *Guia do Scrum*. 2017. Disponível em: https://www.Scrumguides.org/docs/Scrumguide/v2017/2017-Scrum-Guide-Portuguese-Brazilian.pdf. Acesso em: fev. 2020.

_____; SUTHERAND, J. *Guia do Scrum*. 2020. Disponível em: https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-Portuguese-European.pdf. Acesso em: fev. 2022.

SUTHERLAND, J. *The Scrum At Scale® Guide*. 2019. Disponível em: https://www.*Scrum*atscale.com/*Scrum*-at-scale-guide-read-online. Acesso em: fev. 2020.

TAKEUCHI, H.; NONAKA, I. The new new product development game: stop running the relay race and take up rugby. *Harvard Business Review*, v. 64, n. 1, p. 137-147, 1986.

VIANNA JR., A. *Explorando o papel da complexidade no gerenciamento de projetos*: um novo espaço de oportunidades. [S.l.]: [s.n.], 2015.

WIKIPEDIA. *Sequência de Fibonacci*. Disponível em: https://pt.wikipedia.org/wiki/Sequência_de_Fibonacci. Acesso em: fev. 2020.

# PROFESSORES-AUTORES

## MARCELA SOUTO CASTRO

### FORMAÇÃO ACADÊMICA

- Cursou doutorado na ESC Rennes.
- Mestre em Sistemas de Gestão pela Universidade Federal Fluminense (UFF).
- MBA em Organizações e Estratégia pela UFF.
- MBA em Gestão de Projetos pela Fundação Getulio Vargas.
- Graduada em Engenharia de Produção pela UFF.
- Certified Scrum Master (CSM).
- *Practitioner* em Programação Neurolinguística.
- Aluna do Programa de Negociação da Harvard Law School em 2011.

### EXPERIÊNCIA PROFISSIONAL

- Presta consultoria e ministra palestras sobre Negociação e Oratória, tendo como clientes diversas empresas, tais como: TIM, Endesa, Claro, Avon, GSK, Vale, IRB e Pinheiro Neto, entre outras.
- Consultora especialista em Gestão de Projetos, elaborando procedimentos de gestão de projetos, implantando escritórios de projetos e realizando controle de qualidade de projetos estratégicos e treinamentos em empresas como Estaleiro Atlântico Sul, Souza Cruz, TecPrima e ValeSul.
- Professora da Universidade Europeia de Lisboa.
- Professora convidada dos cursos de MBA da FGV, presencial e *on-line*, desde 2008, atuando nas disciplinas de Técnicas de Apresentações e Comunicação Interpessoal, Negociação e Gerenciamento de Projetos.
- Experiência na área comercial, tendo atuado em empresas nacionais e multinacionais de porte como Ampla Distribuidora de Energia, General Electric, Repsol YPF Distribuidora e Turbomeca do Brasil.

# ANDRÉ BARCAUI

## FORMAÇÃO ACADÊMICA

- Pós-Doutor em Administração pela FEA/USP.
- Doutor em Administração pela UNR.
- Mestre em Sistemas de Gestão pela UFF-RJ.
- Graduado em Tecnologia da Informação e Psicologia, com formação em terapia cognitivo-comportamental.
- É certificado PMP, PMI-ACP e DASSM pelo *Project Management Institute*, em KMP pela *Kanban University*, *Master Coach* pelo *Behavioral Coaching Institute* (BCI), *Advanced Scrum Master* pela *Scrum Alliance*, *Public-Private Partnerships Foundation* (CP3P) pela APMG, *SAFe Agilist* e *Agile Coaching* (ICP-ACC) pela ICAGile.

## EXPERIÊNCIAS PROFISSIONAIS

- Palestrante e autor de diversos artigos e livros na área gerencial.
- É membro-fundador do *PMI Chapter Rio* e hoje faz parte do seu Conselho Consultivo.
- Foi *project office manager* da *Hewlett-Packard Consulting*, responsável pela região Latino-Americana.