

Scrum

Relatório de Aprendizado e Boas Práticas

David Neves
Curso: Scrum
Propriedade: Privada

26 de Março de 2026

Sumário

1	Introdução: o que é o Scrum?	4
1.1	O que realmente é o Scrum?	4
1.2	Scrum como framework — o "metrô" do desenvolvimento	5
2	Abordagem Tradicional (Waterfall) vs. Abordagem Ágil	7
2.1	A abordagem ágil	7
2.2	Agilidade x Rapidez	8
3	Manifesto Ágil	8
3.1	Mentalidade ágil	8
3.2	Os 4 valores ágeis	9
3.3	Os 12 princípios ágeis	9
3.3.1	Princípios de Projeto	9
3.3.2	Princípios de Produto	10
3.3.3	Princípios de Pessoas	10
3.3.4	Princípio de Processo	11
4	A relação entre Ágil e Scrum	12
4.1	O Cone da Incerteza	12
4.2	O modelo VUCA	12
5	Visão Geral do Scrum: Empirismo e Pilares	13
5.1	Os 5 valores do Scrum	14
6	O Scrum Team (Papéis e Responsabilidades)	14
6.1	Product Owner (PO)	14
6.2	Developers	15

6.3	Scrum Master	15
6.4	Boa prática	16
6.5	Responsibility vs. Accountability	16
7	Artefatos do Scrum	16
7.1	Product Backlog	17
7.1.1	User Stories e Épicas	17
7.1.2	Estimativas: Tamanho e Story Points	18
7.1.3	Definition of Ready (Definição de Preparado)	18
7.2	Sprint Backlog	19
7.3	Incremento	19
7.3.1	Definition of Done (Definição de Pronto)	20
7.3.2	Dívida Técnica	21
8	Eventos do Scrum	21
8.1	Sprint	22
8.1.1	Cancelamento de Sprint	22
8.1.2	”Sprints inválidas”(termos fora do Scrum Guide)	23
8.2	Sprint Planning	23
8.3	Daily Scrum	25
8.3.1	Exemplo prático de Daily Scrum e Scrum Board	26
8.4	Sprint Review	26
8.5	Sprint Retrospective	27
9	Monitoramento de Progresso	28
9.1	Gráfico de Burn-down	28
9.2	Gráfico de Burn-up	29
9.3	Diagrama de Fluxo Cumulativo (Cumulative Flow Diagram)	29
10	Scrum em Larga Escala: o Nexus Guide	29
10.1	Nexus Integration Team	30
10.2	Eventos do Nexus	31
10.3	Component Teams vs. Feature Teams	31
10.4	Frequência de Sprints em múltiplos Scrum Teams	32
11	Kanban	32
12	Certificação Profissional Scrum Master I (PSM I)	33
12.1	Custo	33
12.2	Conteúdo da prova	33
12.3	Formato	33

12.4	Dicas para a prova	33
12.5	Exemplos de questões (estilo "sniper")	34
13	Recapitulação Geral do Fluxo do Scrum	35
13.1	Objetivo principal do Scrum: entregar valor	35
13.2	Considerações finais	36

1 Introdução: o que é o Scrum?

Tudo começa com um problema a ser resolvido: o desenvolvimento de um produto ou a realização de um novo serviço. Pode ser a criação de um novo software, de novos recursos, o projeto de um carro, uma estratégia de marketing, um curso online etc.

Seja qual for o problema, a equipe o divide em partes menores. Por exemplo, para um site de e-commerce, resolver o problema significaria listar requisitos como: página de inscrição e login, possibilidade de gerenciar o conteúdo do site (CMS), métodos de pagamento, registro de produtos, carrinho de compras, e assim por diante. A equipe lista todas essas necessidades em uma lista de requisitos chamada **Product Backlog**.

No Scrum, os termos costumam ser mantidos em inglês mesmo em contextos de língua portuguesa. Trabalha-se com entregas intermediárias para receber feedback constante do cliente — esse é o espírito incremental e iterativo do framework.

1.1 O que realmente é o Scrum?

O Scrum é um **mecanismo para otimizar o sistema de entrega de valor ao cliente**. Ele é usado para impulsionar o trabalho, otimizar o esforço que fazemos e entregar valor em ambientes complexos.

A complexidade de um problema é definida por duas dimensões: **o que** precisa ser feito e **como** fazer.

- Se chega uma demanda que exige coisas que a empresa já sabe fazer, o problema tende a ser mais fácil (simples ou complicado).
- Se chega uma demanda envolvendo algo ainda não feito (nova tecnologia, RA, Machine Learning etc.), a interseção entre "o que" e "como" fazer se torna mais difícil de prever — isso é o que define um problema **complexo**.

Existem, de forma geral, quatro tipos de problema (referência ao modelo Cynefin, mencionado de forma implícita no curso):

- **Simples**: fácil prever o que precisa ser feito e fácil fazer acontecer.
- **Complicado**: fácil de resolver tecnicamente, mas difícil de chegar a um acordo sobre o que ou como resolver.
- **Complexo**: você não sabe tudo de antemão; a trajetória muda ao longo do caminho. O Scrum ajuda a lidar com esse tipo de problema.
- **Caótico**: nem sequer é possível identificar causa e efeito com clareza; exige ação imediata para depois entender o que aconteceu.

O Scrum é um mecanismo para orientar mudanças por meio de um ciclo contínuo:

1. Levantamento de requisitos
2. Trabalho
3. Inspeção do resultado
4. Priorização de mudanças
5. Adaptação
6. Nova melhoria disponibilizada

Esse ciclo se repete continuamente até chegar ao sucesso do produto ou até o orçamento se esgotar. A essência do Scrum é ser **orientado a mudanças**: um framework simples para resolver problemas complexos.

1.2 Scrum como framework — o "metrô" do desenvolvimento

Uma boa analogia para entender o Scrum como framework é compará-lo a um sistema de metrô: ele define as estações e as regras de circulação (os eventos, papéis e artefatos), mas não dita como cada passageiro deve se comportar em cada estação — isso fica a cargo de práticas complementares.

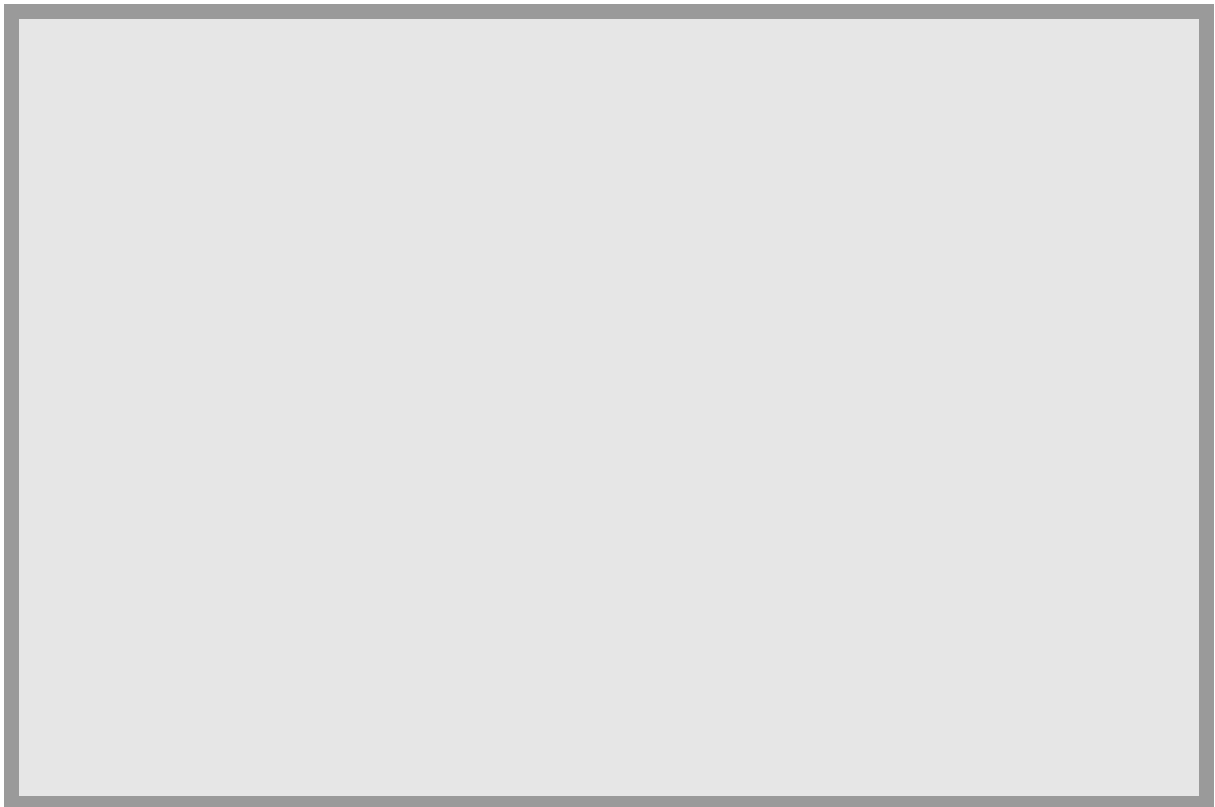


Figura 1: Práticas do Scrum

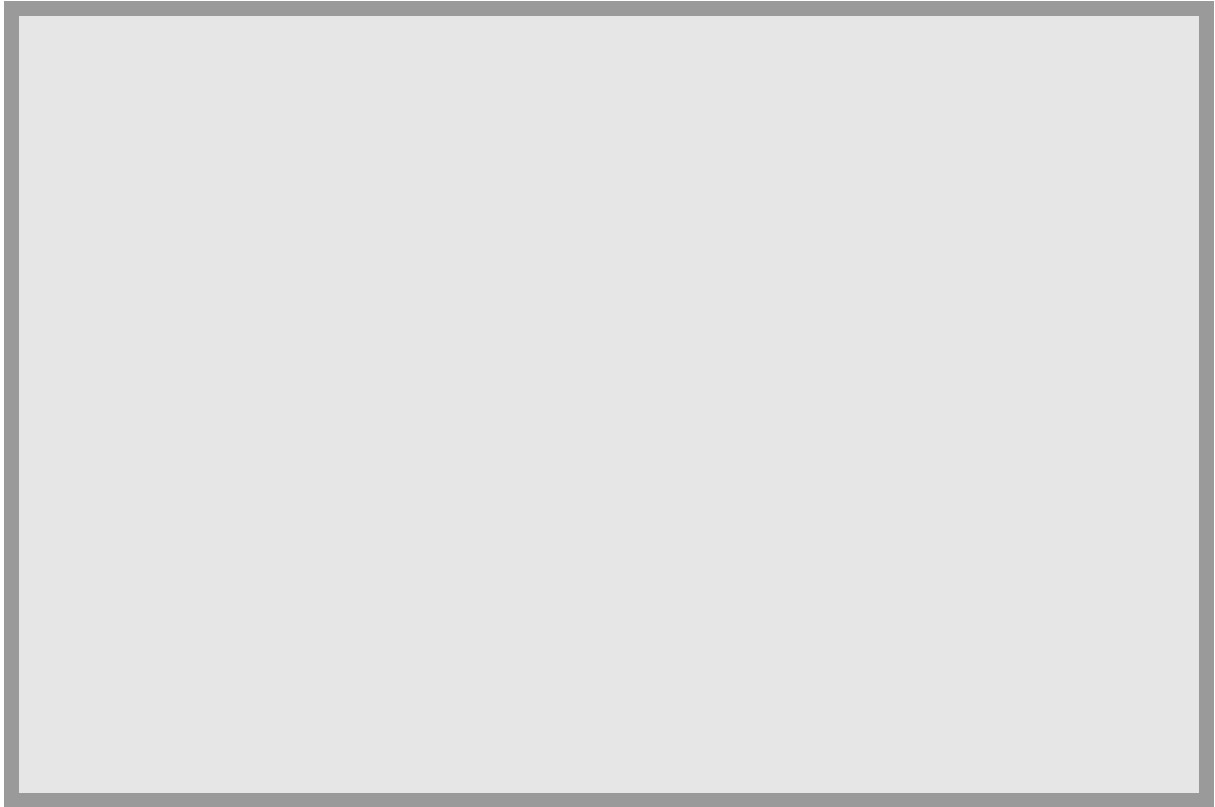


Figura 2: Práticas do Scrum + outras práticas comuns

Como se nota nas figuras, várias áreas e conceitos complementam o Scrum em diferentes contextos e práticas:

- A linha **rosa** representa práticas oficiais do Scrum, de acordo com o Scrum Guide.
- A linha **verde** representa práticas de gerenciamento de produto.
- A linha **laranja** representa práticas de Lean.

Por exemplo, o Product Backlog e a Meta do Produto estão ligados: práticas que pertencem oficialmente ao Scrum estão intimamente conectadas à gestão de produto. As **User Stories** não fazem parte formal das técnicas do Scrum, mas são muito populares para descrever os itens do Product Backlog.

É importante destacar: desenvolver software sem qualquer atividade de DevOps, testes ou design é um empreendimento arriscado — não é recomendado. Porém, quantas atividades de DevOps, testes ou design fazem parte *formalmente* do Scrum? Zero. O Scrum não prescreve como versionar, implantar, projetar ou testar um software. Por isso ele é um **framework**: dá a estratégia, diz para trabalhar com iterações curtas, orienta sobre como lidar com requisitos e sobre o que é importante no dia a dia do produto — mas cabe à equipe e à organização determinar os detalhes operacionais de como o trabalho será executado. Cada produto e cada cultura são diferentes; por isso definir esses

detalhes é sensível e não tem uma resposta de uso geral. O Scrum permanece em nível estratégico — isso não significa que ele seja vago demais, e sim que ele é propositalmente não especificado nesses pontos.

2 Abordagem Tradicional (Waterfall) vs. Abordagem Ágil

A abordagem tradicional foca em seguir rigorosamente o que foi planejado, ou seja, seguir tudo o que foi detalhado no início e depois cumprir o plano à risca.

Uma boa analogia é o uso de um **canhão**: mira-se no alvo, calcula-se distância, vento, peso — e atira-se. Fazendo o paralelo:

- Preparar o canhão = definir a visão do produto, prazo, custo etc.
- Atirar a bala do canhão = executar o que foi planejado.

No modelo **Waterfall** (tradicional), executa-se até o fim — ou seja, depois de um certo tempo, chega-se à visão inicial definida. Mas e se não conseguirmos acertar a visão desde o começo? Assim como no canhão, se o vento mudar ou houver alguma variável inesperada, o tiro erra o alvo.

No mundo dos negócios, o que pode acontecer é que o cliente às vezes nem sabe exatamente o que quer até ver uma versão pronta — a cada incremento ele percebe que o produto poderia ser melhor de outra forma. Por isso, a abordagem tradicional pode levar a um produto inútil sempre que houver incerteza sobre o que construir e como construir.

A alternativa à abordagem tradicional é a **ágil** (ou adaptativa).

2.1 A abordagem ágil

No modelo ágil, reconhecemos que hoje não conseguiríamos elaborar um plano perfeito para chegar ao produto ideal. Por isso, trabalha-se com ciclos curtos (de 1 a 4 semanas), entregando a cada ciclo algo próximo da necessidade do cliente.

O primeiro incremento contém apenas uma versão inicial do produto, que é apresentada aos stakeholders. Depois de experimentar esse primeiro resultado, a equipe, o cliente e os stakeholders podem tomar decisões melhores — sempre com algo utilizável para que sintam "na pele" e deem feedback real. A partir disso, aprendemos mais sobre o que realmente precisa ser entregue, e uma nova versão do produto é construída a cada novo incremento, até chegar à versão correta e final.

Agilidade significa responder às mudanças emergentes ao longo de um determinado desenvolvimento.

- **Waterfall/Tradicional:** mais adequado a empreendimentos com clareza desde o começo, sem previsão de mudanças ao longo do desenvolvimento.
- **Ágil:** mais adequado a ambientes complexos, permitindo a adaptação do produto ao longo do desenvolvimento. Ser ágil é saber lidar com mudanças.

2.2 Agilidade x Rapidez

- **Rapidez:** seguir um fluxo fixo e chegar ao destino o mais rápido possível, sem levar em conta variáveis — apenas mirando na visão inicial.
- **Agilidade:** saber se comportar em intervalos curtos de avaliação. É mais lento em cada passo, porém mais certo: se algo mudar, você gasta menos recursos e tem menos desperdício. A agilidade também é sobre chegar ao produto *correto* de forma eficiente e sem desperdício.

Reduzir desperdício é o princípio do **Lean** (enxuto).

3 Manifesto Ágil

O Manifesto Ágil é um documento — uma declaração de valores revolucionária. Apesar de suas origens ligadas ao desenvolvimento de software, o Manifesto Ágil inspira projetos de várias áreas e remodelou a forma como equipes e projetos trabalham.

É um documento que prevê valores e princípios ágeis. Antes de entrarmos neles, é importante entender a **mentalidade ágil** que os fundamenta.

3.1 Mentalidade ágil

A mentalidade de descoberta significa: não podemos concordar totalmente sobre qual é o problema, nem sabemos qual é o plano perfeito — mas **sabemos que não sabemos** o que fazer, e por isso nos adaptaremos e trabalharemos da melhor forma possível.

Um contratempo não é um fracasso. Por exemplo: entregar um produto de software que não é exatamente o que o cliente precisava não deve gerar culpa — às vezes o cliente só sabe o que quer quando vê a versão pronta. Reconhecemos que estamos lidando com um sistema complexo; por isso aprendemos e nos adaptamos, vendo contratempos como oportunidades de aprendizagem.

Colaboramos com o cliente, e ele colabora conosco — isso é viabilizado adotando ciclos curtos de entrega, colaboração e mudança.

O terceiro pilar da mentalidade é: **concentre-se em entregar valor**. Entregar o que o cliente precisa e ser movido por essa motivação é a essência da mentalidade ágil.

3.2 Os 4 valores ágeis

1. Indivíduos e interações mais que processos e ferramentas.

O ágil se concentra em ter equipes coesas, com boa conexão, dispostas a trabalhar juntas. Mesmo que o problema seja complexo, se cada pessoa compartilhar seu conhecimento, muito provavelmente o resultado será positivo. Cada pessoa tem uma perspectiva diferente e limitada sobre o problema real; ao trabalharem bem juntas, o resultado do grupo tende a ser maior que a soma dos esforços individuais ($1+1=3$). No ágil, processos e ferramentas não são *impostos* às pessoas que fazem o trabalho — eles são *disponibilizados*.

2. Software em funcionamento mais que documentação abrangente.

Esse valor se baseia originalmente em software, mas pode se aplicar a outros campos — um arquiteto poderia pensar em uma planta baixa 2D/3D; no hardware, em protótipos com Arduino, por exemplo. Um produto em funcionamento vale mais que documentação abrangente porque, em problemas complexos, precisamos entregar algo tangível para que o cliente possa ver e tocar.

3. Colaboração com o cliente mais que negociação de contratos.

Precisamos do cliente para colaborar durante as sprints. Fornecedor e cliente devem estar comprometidos com o produto; ter pessoas comprometidas a trabalhar em conjunto facilita as respostas às mudanças.

4. Responder a mudanças mais que seguir um plano.

Importante: valorizar os itens à esquerda não significa que os itens à direita (processos, documentação, contratos, planos) devam ser ignorados — apenas que os primeiros são mais valorizados quando há conflito entre eles.

3.3 Os 12 princípios ágeis

Os princípios podem ser agrupados em quatro categorias: **Projeto**, **Produto**, **Pessoas** e **Processo**.

3.3.1 Princípios de Projeto

Princípio 1: Nossa maior prioridade é satisfazer o cliente através da entrega contínua e adiantada de software (ou produto) com valor agregado.

Dois conceitos relacionados a esse princípio:

- **IKIWISI** (*I Know It When I See It* — "eu sei quando vejo"): a única maneira de obter feedback valioso e aprender o melhor caminho para o sucesso é permitir que clientes e stakeholders tenham experiência real e autêntica com o produto.

- **YAGNI** (*You Ain't Gonna Need It* — "você não vai precisar disso"): trata-se de eliminar desperdícios, entregando apenas as funcionalidades essenciais. Muitas vezes o cliente percebe que o "suficientemente bom" já atende às necessidades dele. Isso pode poupar tempo e dinheiro, sendo a diferença entre sucesso e fracasso em alguns domínios.

Esse princípio se concentra em satisfazer o cliente com um produto valioso que resolva seus problemas — foco no cliente, não na tecnologia. Problemas que antes levavam meses para serem resolvidos passam a ser resolvidos em dias.

Princípio 2: Mudanças nos requisitos são bem-vindas, mesmo tardiamente no desenvolvimento. Processos ágeis tiram vantagem das mudanças, visando vantagem competitiva para o cliente. Mudanças são o que leva ao sucesso em ambientes complexos.

Princípio 3: Entregar frequentemente software funcionando, de poucas semanas a poucos meses, com preferência à menor escala de tempo.

Princípio 8: Processos ágeis promovem desenvolvimento sustentável. Patrocinadores, desenvolvedores e usuários devem ser capazes de manter um ritmo constante indefinidamente. Trabalhar muitas horas extras não é o caminho a seguir — o ágil é projetado para desenvolver problemas complexos, e profissionais de desenvolvimento precisam tomar decisões difíceis para manter esse ritmo sustentável.

Princípio 7 (último de projeto): Software funcionando é a medida primária de progresso. A entrega real só é alcançada por meio de produtos funcionais — é preciso ter experiência autêntica com o produto.

3.3.2 Princípios de Produto

Simplicidade: a arte de maximizar a quantidade de trabalho *não realizado* é essencial. Há duas perspectivas: de negócio (pense no YAGNI e no Lean) e técnica. Podemos simplificar o problema — a pessoa precisa de algo que resolva seu problema, não necessariamente algo espetacular.

Princípio 9: Contínua atenção à excelência técnica e ao bom design aumenta a agilidade. Queremos uma sequência de entregas rápidas; construir coisas de má qualidade nos deixa lentos, pois a má qualidade aumenta o custo da mudança. Pense na dificuldade de alterar um código de baixa qualidade — qualquer pequena mudança leva uma eternidade. Diante disso, a boa qualidade permite mais agilidade, pois um código com qualidade torna mais fácil continuar evoluindo. É importante deixar a equipe decidir "até certo ponto" o que fazer, e depois confiar nela para decidir os detalhes.

3.3.3 Princípios de Pessoas

Princípio 6: O método mais eficiente e eficaz de transmitir informações para e entre uma equipe de desenvolvimento é através de conversa face a face. A comunicação é a

principal causa de fracasso de projetos. É preciso tomar cuidado com ruídos — aquilo que a outra pessoa pode não entender por falta de experiência compartilhada. Estudos demonstram que a comunicação face a face com quadro branco ou papel é mais eficaz, mas o modo presencial não é a única forma válida: o uso de ferramentas digitais combinado com recursos visuais também ajuda muito, especialmente quando é importante que as informações fiquem registradas (por exemplo, quando a equipe trabalha de forma assíncrona). Ferramentas como o Jira são usadas para discutir questões diretamente nos cartões de tarefas. Documentação detalhada também é muito eficaz e, em alguns casos, necessária dependendo do tipo de contrato — por exemplo, em áreas médicas regulamentadas, onde só é possível lançar produtos se a organização for certificada por agências regulamentadoras.

Princípio 4: Pessoas de negócio e desenvolvedores devem trabalhar diariamente em conjunto por todo o projeto. No modelo tradicional (Waterfall), os clientes normalmente estão fortemente envolvidos apenas no início (para validar o contrato) e no final (para validar se o produto está de acordo com o pedido) — isso funciona bem quando tudo é detalhado com precisão no início. O ágil pressupõe que isso nem sempre é possível, e por isso exige que o cliente esteja disponível a todo momento do projeto. "Disponível" significa que dúvidas técnicas relacionadas a negócio devem ser respondidas rapidamente (idealmente entre 75% e 85% das dúvidas respondidas em poucos minutos), evitando desperdício.

Princípio 11: As melhores arquiteturas, requisitos e designs emergem de equipes auto-organizáveis. Em modelos tradicionais, gestores pensam e decidem "o que" e "como" fazer (inteligência de cima para baixo). Ao trabalharmos com problemas complexos, é mais provável que as pessoas que realizam o trabalho e estão mais imersas no problema estejam mais bem informadas do que os gestores. Em outras palavras, para problemas complexos precisamos de inteligência de baixo para cima — isso é a **autogestão**.

Princípio 5: Construa projetos em torno de indivíduos motivados. Dê a eles o ambiente e o suporte necessários, e confie neles para fazer o trabalho. Esse princípio está ligado à ideia de **líderes servidores** — líderes que servem a equipe, e não o contrário.

3.3.4 Princípio de Processo

Princípio 12 (último): Em intervalos regulares, a equipe reflete sobre como se tornar mais eficaz, e então refina e ajusta seu comportamento de acordo. Embora os membros da equipe ágil aprendam mais sobre o produto enquanto desenvolvem, eles também aprendem a ser mais ágeis em seu próprio processo de desenvolvimento. A cultura ágil analisa o que está funcionando e o que não está, melhorando continuamente.

4 A relação entre Ágil e Scrum

O Scrum é uma forma de implementar a agilidade — ele se baseia no Manifesto Ágil. O Scrum é um framework para resolver problemas complexos, concentrando-se em fazer isso de forma produtiva e criativa. É altamente flexível e tem sucesso em muitos domínios (não apenas em software).

4.1 O Cone da Incerteza

Recebe esse nome pela forma como pode ser visualizado: um cone que representa a incerteza, em um plano cartesiano de duas dimensões.

- O eixo Y representa a **variabilidade** (grau de erro). Quanto maior a incerteza, maior a variabilidade.
- O pior cenário é dado multiplicando a estimativa pela variabilidade; o melhor cenário é dado dividindo a estimativa pela variabilidade.

Por exemplo, se temos uma estimativa de 4 dias para uma tarefa e um fator de variabilidade de 2:

- Pior caso: $4 \times 2 = 8$ dias.
- Melhor caso: $4 \div 2 = 2$ dias.

Em um exemplo de software com estimativa inicial de 2 meses (8 semanas): o cenário mínimo poderia ser 0,5 mês (2 semanas) e o máximo 8 meses (32 semanas) — refletindo fatores de variabilidade diferentes conforme a fase do projeto (0,25x a 4x, por exemplo, nas fases iniciais).

Quanto mais complexo o domínio, maior tende a ser o fator de variabilidade. Isso explica por que, no setor de software, a variabilidade das estimativas costuma ser maior do que em setores mais previsíveis, como a indústria química.

4.2 O modelo VUCA

Quatro fatores formam o acrônimo **VUCA** (Volatilidade, Incerteza, Complexidade e Ambiguidade — do inglês *Volatility, Uncertainty, Complexity, Ambiguity*):

- **Volatilidade**: quanto maiores as mudanças, maior a volatilidade.
- **Incerteza (Uncertainty)**: quanto menos previsível o cenário, maior a incerteza.
- **Complexidade**: nível de interconectividade e interdependência. Quanto mais regras de negócio e dependências tecnológicas, maior a complexidade.

- **Ambiguidade:** nível em que entendemos o significado de algo. Um exemplo é tentar descrever com precisão o que o usuário deseja — muitas vezes precisamos assumir hipóteses para poder avançar.

A variabilidade da indústria de software tende a ser maior que a de indústrias mais tradicionais (como a química) por três motivos principais:

1. **Volatilidade:** novas tecnologias surgem muito rapidamente.
2. **Complexidade:** projetos de software costumam ser mais complexos devido à quantidade de regras de negócio e dependências tecnológicas.
3. **Ambiguidade:** é mais difícil saber com precisão o que os usuários realmente desejam, o que aumenta a incerteza.

A adoção de ciclos de entrega curtos ajuda a reduzir o impacto do VUCA, e a agilidade é considerada a melhor abordagem para lidar com ambientes VUCA.

5 Visão Geral do Scrum: Empirismo e Pilares

A estrutura do Scrum é baseada no **empirismo**: a ideia de que o conhecimento vem da experiência e das decisões são tomadas com base no que é observado, e não apenas planejado. Isso implica trabalhar em ciclos de: fazer, medir/observar, se adaptar conforme se avança.

O empirismo se apoia em três pilares:

- **Transparência:** é preciso ter um padrão comum para que todas as pessoas envolvidas enxerguem as mesmas informações — tudo deve ser visível. Isso é reforçado constantemente nas reuniões e nos artefatos, garantindo compreensão alinhada sobre o que está sendo feito. Aqui também entram os conceitos de item "preparado" (*Definition of Ready*) e item "pronto" (*Definition of Done*) — o conceito de item preparado, inclusive, está presente no próprio Scrum Guide.
- **Inspeção:** testar, verificar e observar constantemente os artefatos e o progresso. Se houver um desvio, é preciso ajustar.
- **Adaptação:** fazer os ajustes necessários com base no que foi inspecionado. É por isso que o Scrum trabalha com sprints curtas — elas favorecem ciclos rápidos de inspeção e adaptação.

5.1 Os 5 valores do Scrum

Os valores do Scrum existem para gerar confiança entre o Scrum Team e os stakeholders:

1. **Comprometimento (Commitment)**: as pessoas se comprometem pessoalmente com os objetivos do Scrum Team, e não apenas "se envolvem" superficialmente. Pessoas de negócio e técnicas trabalham próximas, e todo o Scrum Team é responsabilizado pelos resultados — "se um falha, todos falham" (em certo sentido).
2. **Coragem (Courage)**: os membros do Scrum Team devem ter coragem para sair da zona de conforto, dizer a verdade e enfrentar problemas difíceis. O Scrum Team não existe apenas para deixar a gestão feliz.
3. **Foco (Focus)**: o Product Backlog é ordenado com os itens mais importantes no topo — a equipe se concentra em detalhar apenas o que está no topo.
4. **Abertura (Openness)**: bom trabalho em equipe envolve pedir ajuda quando necessário e admitir erros quando eles acontecem.
5. **Respeito (Respect)**: respeito mútuo entre todos os membros da equipe e stakeholders.

6 O Scrum Team (Papéis e Responsabilidades)

O Scrum Team é formado por três papéis (accountabilities, na terminologia oficial): **Product Owner**, **Developers** e **Scrum Master**. Se o Scrum Team se tornar muito grande, ele precisa se reorganizar em vários Scrum Teams.

6.1 Product Owner (PO)

O Product Owner maximiza o valor do produto resultante do trabalho do Scrum Team. Ele faz isso principalmente **gerenciando o Product Backlog**.

Responsabilidades do PO em relação ao Product Backlog:

- Desenvolver e comunicar explicitamente a Meta do Produto (*Product Goal*).
- Criar e ordenar claramente os itens do Product Backlog.
- Garantir que o Product Backlog seja visível, transparente e compreendido por todos.

O PO pode delegar parte dessas responsabilidades a outras pessoas, mas continua sendo o único responsável (accountable) pelo resultado.

Pontos importantes:

- Stakeholders não podem interromper ou dar ordens diretamente aos desenvolvedores — eles passam pelo PO.
- O PO precisa ter **autoridade** para decidir; sem isso, o processo de sprints perde o sentido, pois decisões acabam demorando demais (longos ciclos de feedback geram retrabalho e frustração).
- O feedback deve ser rápido (idealmente em minutos, não em meses).
- O PO precisa ter conhecimento e autoridade no âmbito de negócios, e precisa estar disponível para a equipe.
- É importante que todas as pessoas envolvidas — técnicas e de negócio — entendam o Scrum, para se beneficiarem plenamente dele (por exemplo, por meio de um mini curso ou palestra). Caso não haja esse entendimento ou interesse, talvez seja mais adequado seguir uma gestão tradicional com contrato de escopo fixo.

6.2 Developers

Os Developers transformam os itens do Product Backlog em um Incremento utilizável. Indivíduos podem ter uma área de foco especializada, mas **não existem subequipes com hierarquia** dentro do time de desenvolvimento.

No Scrum, se você contribui para qualquer aspecto da criação de um incremento, você é considerado um "developer"— independentemente do seu cargo formal (design, QA, programação etc.).

O Scrum Team se autogerencia: os próprios desenvolvedores decidem quem faz o quê e quando. A equipe precisa de flexibilidade para se organizar dessa forma.

6.3 Scrum Master

O Scrum Master tem duas funções principais: servir a **organização** e servir o **Scrum Team**.

- Ajuda "de fora" a aumentar a eficácia do Scrum Team.
- Auxilia o Product Owner a maximizar o valor entregue (por exemplo, ensinando técnicas de gerenciamento do Product Backlog).
- Treina o Scrum Team a se autogerenciar e a entregar incrementos valiosos.
- Ajuda a estabelecer e ensinar o Scrum a todos os envolvidos.
- Atua na remoção de impedimentos que atrapalham o progresso da equipe.

- Verifica se a organização está seguindo as regras e princípios do Scrum, e ensina a organização a obter os benefícios do framework.

Exemplo prático — lidando com trabalho não finalizado ao fim de uma sprint:

- O trabalho não finalizado **não** deve ser considerado parte do Incremento, pois o incremento é composto apenas por itens que atendem à *Definition of Done*.
- O trabalho não finalizado **não** deve ser apresentado formalmente na Sprint Review (embora possa ser mencionado ou discutido).
- Esse trabalho retorna ao Product Backlog para consideração futura.

Exemplo prático — lidando com um desenvolvedor hostil ou com conflitos internos: a responsabilidade primária de resolver o problema é dos próprios desenvolvedores; somente se eles não conseguirem resolver sozinhos é que o assunto deve ser escalado ao Scrum Master.

6.4 Boa prática

Bons Scrum Masters tendem a ser "invisíveis" no sentido de que apoiam o Scrum Team sem se colocar no centro das atenções — o sucesso da equipe é o indicador do seu trabalho.

6.5 Responsibility vs. Accountability

Esses dois termos, embora frequentemente traduzidos da mesma forma em português ("responsabilidade"), têm significados distintos em inglês:

- **Responsibility:** a pessoa que efetivamente *executa* uma atividade.
- **Accountability:** a pessoa que *responde* pelo resultado da atividade, mesmo que não a execute diretamente.

Essa distinção é especialmente relevante ao estudar para provas de certificação em inglês, como a PSM I (ver Seção 12).

7 Artefatos do Scrum

Os artefatos do Scrum representam trabalho ou valor. Eles são projetados para maximizar a transparência das principais informações, de modo que todos que os inspecionam tenham a mesma base para adaptação.

Cada artefato contém um **compromisso** que reforça o empirismo e os valores do Scrum:

- Para o **Product Backlog**, o compromisso é a **Meta do Produto** (*Product Goal*).
- Para o **Sprint Backlog**, o compromisso é a **Meta da Sprint** (*Sprint Goal*).
- Para o **Incremento**, o compromisso é a **Definição de Pronto** (*Definition of Done*).

7.1 Product Backlog

O Product Backlog é uma lista **ordenada** e **emergente** do que é necessário para melhorar o produto. É a **única fonte de trabalho** realizado pelo Scrum Team.

- Quem **ordena** o Product Backlog é o Product Owner.
- Ter uma Meta do Produto (*Product Goal*) é **obrigatório**, não opcional.
- Itens que podem ser realizados pelo Scrum Team em uma única Sprint são considerados "preparados" para seleção no Sprint Planning — geralmente atingindo esse nível de transparência após atividades de refinamento.

Product Backlog refinement: é o ato de quebrar os itens do backlog em partes menores e adicionar definição adicional a eles (descrição, ordem, tamanho), tornando-os mais precisos. É uma atividade contínua, e os atributos utilizados variam conforme o domínio do trabalho.

Os **Developers** são responsáveis pelo dimensionamento (estimativa de tamanho) dos itens; o Product Owner pode influenciá-los ajudando a entender e a negociar *trade-offs* (trocas entre itens).

Meta do Produto (Product Goal): descreve um estado futuro do produto que serve como alvo para o planejamento do Scrum Team. A Meta do Produto está contida no Product Backlog, e o restante do backlog emerge para definir "o que" cumprirá essa meta.

Um produto é um veículo para entregar valor — tem um limite claro, stakeholders conhecidos e usuários/clientes bem definidos. Pode ser um serviço, um produto físico, ou algo mais abstrato. A Meta do Produto é o objetivo de longo prazo do Scrum Team: ele deve cumprir (ou abandonar) uma meta antes de assumir a próxima.

7.1.1 User Stories e Épicas

Dentro do Product Backlog, itens grandes costumam ser chamados de **épicas** (por exemplo: "cadastro e autenticação", "notificações", "sistema de lances em tempo real"). Já os itens menores, mais refinados, costumam ser descritos como **User Stories**.

A User Story tem três componentes clássicos (conhecidos como "os 3 Cs"):

1. **Cartão (Card):** uma descrição simples da funcionalidade sob a perspectiva do usuário — geralmente registrada em ferramentas como o Jira.

2. **Conversação (Conversation):** facilita o diálogo entre stakeholders, desenvolvedores e Product Owner sobre os detalhes do item.
3. **Confirmação (Confirmation):** os critérios de aceitação que definem quando a história está completa.

Um formato popular para escrever User Stories segue a estrutura: *"Como [ator/persona], eu quero [ação], para que [benefício/motivo]"*.

7.1.2 Estimativas: Tamanho e Story Points

Quando falamos do "tamanho" dos itens do backlog, estamos falando de estimativas — determinar, de forma relativa, o esforço necessário para concluir cada item.

- **Story Points:** uma medida relativa (e específica de cada equipe) de esforço/complexidade, e não uma medida direta de tempo (horas). Costuma-se usar uma sequência como a de Fibonacci ou "semi-Fibonacci" (ex.: 1, 2, 3, 5, 8, 13, 20, 40...) para dimensionar os itens, evitando falsa precisão.
- O conhecimento da equipe tem influência direta sobre as estimativas — por isso Story Points não são comparáveis entre equipes diferentes.

Planning Poker: técnica popular de estimativa em grupo. Cada participante possui um baralho de cartas com valores (geralmente seguindo Fibonacci). Todos escolhem secretamente uma carta representando sua estimativa para o item em discussão, e revelam-na ao mesmo tempo. Diferenças grandes de estimativa geram uma conversa para alinhar entendimento antes de uma nova rodada.

Uma prática comum é começar atribuindo o valor **2** ao menor item do Product Backlog — deixando espaço (o valor 1) para itens ainda menores que possam surgir no futuro.

Valor de negócio (Business Value): além da estimativa de esforço, também é comum atribuir uma estimativa de **valor** a cada item do backlog (por exemplo, em uma escala qualitativa como P/M/G, ou "Baixo/Médio/Alto"), usando alguma das diversas técnicas disponíveis para calcular valor de negócio. Isso ajuda o Product Owner a priorizar itens que entregam mais valor com relativamente menos esforço.

7.1.3 Definition of Ready (Definição de Preparado)

É um conjunto de critérios que define quando um item do Product Backlog está preparado para ser selecionado e trabalhado em uma sprint. Normalmente inclui critérios de aceitação do que precisa ser feito e a garantia de que o item cabe dentro de uma única sprint, além de ser bem compreendido pela equipe.

O conceito de "preparado" existe para facilitar a colaboração e reduzir surpresas durante o Sprint Planning. Um exemplo prático: se o Product Backlog não tem nenhum

item preparado (porque todos são muito grandes), a equipe pode refinar o primeiro épico (por exemplo, "registro e autenticação") em User Stories menores, atualizando o Product Backlog adequadamente até que os itens estejam prontos para seleção.

7.2 Sprint Backlog

O Sprint Backlog é composto por três elementos:

1. A **Meta da Sprint** (*Sprint Goal*) — o "porquê".
2. Os itens do Product Backlog selecionados para a sprint — o "o quê".
3. O plano para entregar o incremento — o "como".

O Sprint Backlog nasce da seleção de itens de maior prioridade do Product Backlog durante o Sprint Planning. Esses itens serão desenvolvidos em, no máximo, um mês (o período da sprint).

Regras importantes:

- A partir do momento em que o Sprint Backlog é finalizado, apenas os **Developers** podem alterá-lo — ele é a única fonte de trabalho durante a sprint.
- A Meta da Sprint é **fixa** e não pode ser ameaçada; já os itens do Product Backlog selecionados podem mudar (ser renegociados com o Product Owner) se necessário, desde que a Meta da Sprint não seja comprometida.
- Se alguém de fora da equipe de desenvolvimento quiser alterar o Sprint Backlog, deve buscar primeiro o Product Owner e depois negociar diretamente com os Developers.

7.3 Incremento

Um incremento é um "trampolim" concreto em direção à Meta do Produto. Cada incremento é somado a todos os incrementos anteriores e completamente verificado, garantindo que todos funcionem juntos. Para fornecer valor, o incremento deve ser **utilizável**.

Vários incrementos podem ser criados dentro de uma única Sprint. A soma dos incrementos é apresentada na Sprint Review, apoiando o empirismo — porém um incremento pode ser entregue aos stakeholders **antes** do final da sprint. A Sprint Review nunca deve ser vista como o único marco para liberar valor.

O trabalho só pode ser considerado parte de um incremento se atender à **Definition of Done**.

7.3.1 Definition of Done (Definição de Pronto)

É uma descrição formal do estado do incremento quando ele atende às medidas de qualidade exigidas para o produto.

- No momento em que um item do Product Backlog atende à Definition of Done, um incremento "nasce".
- A DoD cria transparência, dando a todos um entendimento compartilhado do que foi realmente concluído.
- Se um item não atender à DoD, ele **não pode** ser liberado nem apresentado na Sprint Review — ele retorna ao Product Backlog para consideração futura.
- Se a organização já tiver uma Definition of Done padrão, todos os Scrum Teams devem segui-la como **mínimo** — podendo adicionar critérios próprios, mais rigorosos, conforme a necessidade de cada produto ou equipe.
- Se não houver um padrão organizacional, o próprio Scrum Team deve criar uma DoD apropriada para o produto.
- Cada produto possui uma única Definition of Done "oficial" (ou a base mínima compartilhada), mas cada equipe (quando há mais de uma trabalhando no mesmo produto) pode adicionar critérios próprios mais rigorosos.

Exemplos de critérios comuns de DoD: garantia de qualidade, conformidade arquitetural, atendimento a requisitos não funcionais, gerenciamento de processos e de configuração, build executado com sucesso local e no servidor, testes unitários e funcionais aprovados, critérios de aceitação atendidos, manual do usuário atualizado, notas de lançamento criadas, ausência de defeitos críticos ou importantes.

Importante: a Definition of Done é diferente dos critérios de aceitação — estes últimos são específicos de uma User Story em particular, enquanto a DoD se aplica a **todos** os itens do incremento. Por exemplo, "as páginas devem carregar em menos de um segundo em uma conexão de internet mediana" é um critério de DoD (aplicável a todas as páginas), enquanto um critério de aceitação seria específico de uma única história.

Definir a Definition of Done é um esforço contínuo: uma equipe pode começar com poucos critérios e, com o tempo, adicionar mais — sempre que a organização ou a equipe julgar necessário. Esse pode inclusive ser um tópico de discussão em uma Sprint Retrospective.

Contexto real de aplicação: em organizações maduras, a DoD costuma estar fortemente ligada a padrões e convenções de qualidade — que, por sua vez, podem precisar estar em conformidade com padrões externos (por exemplo, em uma empresa de software para a área da saúde, pode ser necessário seguir normas como ISO 9001, CMMI

ou ISO 13485, além de garantir segurança e privacidade dos dados de pacientes). Nesses casos, a DoD da organização funciona como o mínimo obrigatório, e o Scrum Team pode incrementá-la com critérios mais rigorosos e específicos.

7.3.2 Dívida Técnica

É uma consequência de práticas inadequadas no desenvolvimento de software, que pode levar à deterioração progressiva do código e da arquitetura. Um código com dívida técnica alta costuma ser muito difícil de manter, e o custo de qualquer mudança se torna bastante alto.

A Definition of Done pode ajudar a controlar a ocorrência de dívida técnica, mas é importante notar que a dívida técnica não é necessariamente algo ruim — às vezes ela é apenas parte natural do processo de aprendizado da equipe (decisões tomadas com o conhecimento disponível no momento).

Sprints por incremento: a primeira sprint é a soma de todos os itens do Product Backlog concluídos naquela sprint. A segunda sprint é a soma dos itens concluídos na sprint 2 mais o que já havia sido entregue na sprint 1 — e assim por diante. Sempre que um item atende à DoD, ele já nasce como parte do incremento; porém o incremento final oficialmente reconhecido é definido na Sprint Review.

8 Eventos do Scrum

O Scrum é composto por 5 eventos:

1. **Sprint:** funciona como um "container" para os demais eventos.
2. **Sprint Planning:** acontece no início de cada sprint, para planejamento do trabalho.
3. **Daily Scrum:** onde a equipe observa o progresso das tarefas e ajusta o que for necessário, mantendo foco total na execução da sprint.
4. **Sprint Review:** revisão do incremento e, se necessário, atualização do Product Backlog — momento de apresentar o incremento ao cliente e receber feedback.
5. **Sprint Retrospective:** foco na melhoria contínua da equipe — revisão dos próprios processos de trabalho para melhorar a cada sprint.

Esses eventos são essenciais para criar regularidade e minimizar a necessidade de reuniões não previstas no Scrum. Para manter o time focado, **todos os eventos são *time-boxed*** — o Scrum Guide prescreve uma duração máxima para cada um. A duração da sprint em si deve ser definida pela equipe *antes* de a sprint começar, e uma sprint só termina quando seu time-box se esgota.

8.1 Sprint

A Sprint é conhecida como o "coração" do Scrum — é como uma corrida curta em que o time trabalha por um período limitado para desenvolver apenas uma parte do produto. Ao final de cada sprint, um incremento do produto (uma versão parcial e utilizável) deve ser entregue.

O que acontece imediatamente após uma sprint? Outra sprint — assim como um batimento cardíaco seguido de outro. A sprint é um time-box de, no máximo, **um mês**.

Três fatores influenciam a duração ideal de uma sprint:

1. A sprint deve ser **curta o suficiente** para manter os riscos em um nível aceitável — quanto mais incerto o ambiente de negócio, mais curta a sprint deve ser.
2. A intenção de sempre ter um incremento pronto ao final — dependendo do número de desenvolvedores e do rigor da DoD, pode ser necessário mais ou menos tempo.
3. Fatores externos de sincronização — por exemplo, se um evento importante ocorre em 3 semanas, pode fazer sentido ajustar a duração da sprint para se alinhar a essa data, mesmo que a equipe normalmente use sprints de 2 semanas.

Regras relacionadas às sprints:

- As sprints têm duração **fixa**, definida no Sprint Planning — não deve ser alterada com frequência.
- Nenhuma alteração pode colocar em risco a Meta da Sprint.
- A qualidade não deve diminuir.
- O Product Backlog é refinado conforme necessário; o escopo pode ser esclarecido e renegociado com o Product Owner conforme se aprende mais durante a sprint.

8.1.1 Cancelamento de Sprint

Uma sprint pode ser cancelada — por exemplo, se, na primeira semana, uma mudança relevante no mercado, na empresa ou na tecnologia tornar a Meta da Sprint obsoleta ou inútil. **Somente o Product Owner** tem autoridade para cancelar uma sprint (outras pessoas podem influenciar essa decisão, mas não têm a autoridade formal). O cancelamento deve ser raro, e só deve ocorrer quando a Meta da Sprint realmente se tornar obsoleta.

8.1.2 "Sprints inválidas"(termos fora do Scrum Guide)

Alguns termos usam a palavra "sprint", mas não fazem parte da prescrição oficial do Scrum:

- **Hardening Sprint** (também chamada de sprint de integração ou estabilização): usada quando o Scrum Team acumulou muita dívida técnica e reserva uma sprint inteira só para "pagá-la". Costuma acontecer quando a equipe não seguiu bem a DoD ou simplesmente a ignorou durante o desenvolvimento. Isso **não está alinhado** ao Scrum — é uma má prática conhecida. Toda sprint deveria entregar, no mínimo, um pouco de funcionalidade de negócio real.
- **Sprint 0**: geralmente usada para atividades iniciais de configuração do projeto. Como toda sprint deve entregar funcionalidade de negócio, tecnicamente não é uma sprint "válida"segundo o Scrum Guide — mas é um termo amplamente usado na prática.
- **Release Sprint**: também considerada um antipadrão.
- **Design Sprint**: voltada para prototipagem e geração de ideias de produto (popularizada pelo Google). Não existe formalmente no Scrum — a técnica "pegou carona"na fama do termo "sprint"do Scrum, mas tem origem e propósito diferentes.

Cada sprint deve entregar um incremento de valor e alguma funcionalidade de negócio real — mesmo que pequena — desde a primeira até a última sprint do projeto.

8.2 Sprint Planning

O planejamento do que será executado começa, na prática, antes mesmo do evento formal de Sprint Planning: o esforço de planejamento do Product Owner começa analisando o que precisa ser feito a seguir e pensando em qual objetivo deveria ser alcançado na próxima sprint — sempre mapeado com a Meta do Produto, para manter a equipe focada nela. O PO pode entrar no evento já pensando em questões como: quais ideias e suposições precisam ser validadas, quais riscos precisam ser abordados, ou quais qualidades precisam ser esclarecidas.

No Scrum, costuma-se visualizar a sprint como um "mini projeto", e o Sprint Planning é o momento em que o Scrum Team define a **Meta da Sprint** — um plano flexível para alcançá-la (o plano em si é o Sprint Backlog).

Participantes: o Scrum Team e, eventualmente, stakeholders convidados (por exemplo, quando os stakeholders pedem algo que exige conhecimento externo que o time não possui).

Time-box: no máximo 8 horas para uma sprint de um mês (proporcionalmente menor para sprints mais curtas).

Estrutura do evento (3 tópicos principais):

1. **Por que esta sprint é valiosa?** O Product Owner discute as oportunidades para aumentar o valor do produto. Como consequência, todo o Scrum Team define a Meta da Sprint (*Sprint Goal*) — que comunica *por que* a sprint é valiosa. Exemplo: "Os usuários acessam facilmente o sistema usando um e-mail válido."
2. **O que pode ser feito nesta sprint?** Depois de definida a Meta da Sprint, os Developers discutem com o Product Owner os itens do Product Backlog (podendo ser necessário refiná-los durante o próprio evento). Os desenvolvedores podem usar informações como:
 - **Capacidade futura:** o tempo de desenvolvimento disponível para a sprint.
 - **Desempenho passado / Velocidade:** por exemplo, se nas últimas três sprints a equipe entregou 20, 25 e 18 pontos, a média (21 pontos) pode servir como referência de velocidade esperada.
 - **A Definition of Done:** quanto mais rigorosa, mais tempo pode ser necessário para deixar os itens realmente prontos.

A partir disso, os Developers selecionam os itens candidatos do Product Backlog para a sprint.

3. **Como o trabalho selecionado será entregue?** Aqui, os Developers têm a palavra final — ninguém pode dizer a eles como transformar os itens em incremento de valor; o Scrum não prescreve isso. Na prática, os times costumam decompor os itens em **tarefas técnicas** menores (geralmente de um dia ou menos de trabalho), com o objetivo de tornar o trabalho mais claro e melhorar a previsibilidade do que a equipe consegue realizar durante a sprint — sem a pretensão de mapear absolutamente todas as tarefas de antemão.

Exemplo prático de decomposição de tarefas: a equipe pode começar esboçando a solução (interfaces de usuário, integrações com outros sistemas etc.) em cerca de 15–20 minutos, para ter uma visão geral do que precisa ser feito. Em seguida, cada membro da equipe escreve secretamente, em post-its, as tarefas que imagina necessárias para cada item selecionado (cerca de 5 minutos). Depois, compartilham e discutem suas anotações, buscando chegar a um conjunto final de tarefas para cada item do backlog. É uma boa prática, ao final, revisar o esboço inicial e mapear as tarefas planejadas com cada parte da interface ou integração que precisa ser desenvolvida, como uma etapa extra de validação.

Não existem regras rígidas sobre como anotar as tarefas, mas dois pontos são essenciais:

1. Os desenvolvedores devem conseguir se lembrar do que se trata cada tarefa ao longo da sprint — ou seja, cada tarefa precisa ter detalhes suficientes.
2. O objetivo não é prever *todas* as tarefas da sprint antes de começar — é normal (e esperado) que novas tarefas surjam durante a execução.

Ao final do Sprint Planning, é criada a **primeira versão do Sprint Backlog**, contendo a Sprint Goal, os itens do Product Backlog selecionados e o plano (tarefas técnicas iniciais). A partir desse momento, somente os Developers podem alterar o Sprint Backlog.

8.3 Daily Scrum

Encontro que ocorre diariamente, com **time-box de 15 minutos**. Os Developers se reúnem para sincronizar seu trabalho em direção à Meta da Sprint, ajustando o Sprint Backlog conforme necessário.

- O Scrum Master apenas garante que a reunião aconteça e que respeite o prazo de 15 minutos, mantendo-a produtiva.
- O Product Owner **não precisa** participar; se participar, deve fazê-lo apenas para tirar dúvidas ou dar suporte, sem se tornar o centro das atenções.
- A Daily Scrum é um evento para os Developers — a exceção é quando o PO ou o Scrum Master estão trabalhando ativamente em itens do Sprint Backlog; nesse caso, participam *como desenvolvedores*.

Geralmente é feita no mesmo horário e local todos os dias, para reduzir a complexidade de coordenação. Um formato comum (embora o Scrum Guide não o exija) é responder a três perguntas:

1. O que eu fiz desde a última reunião?
2. O que eu farei até a próxima Daily para ajudar a equipe a alcançar a Sprint Goal?
3. Eu vejo algum impedimento que atrapalhe a equipe de atingir a Sprint Goal?

O Scrum Guide não define exatamente como a reunião deve ser conduzida — o importante é que ela se concentre em avaliar o progresso da equipe em direção à meta.

Atenção: a Daily Scrum **não** é uma reunião técnica para *resolver* problemas. Se alguém está com dificuldade em uma tarefa, deve avisar a equipe durante a reunião e talvez pedir ajuda — mas a resolução detalhada do problema deve acontecer depois, entre as partes interessadas.

Os desenvolvedores também podem usar a Daily para apoiar a tomada de decisões, monitorando o progresso da sprint (por exemplo, acompanhando o trabalho restante por meio de um gráfico de burn-down).

8.3.1 Exemplo prático de Daily Scrum e Scrum Board

Suponha um Scrum Team com três desenvolvedores: Maique, João e Alice. Eles decidem conduzir a reunião sempre em ordem alfabética, respondendo às três perguntas populares.

No início da sprint, o Sprint Backlog tem um total de, por exemplo, 11 story points distribuídos entre as tarefas selecionadas por cada um. Supondo uma sprint de duas semanas (10 dias úteis), a equipe pode montar um gráfico de burn-down para acompanhar o progresso.

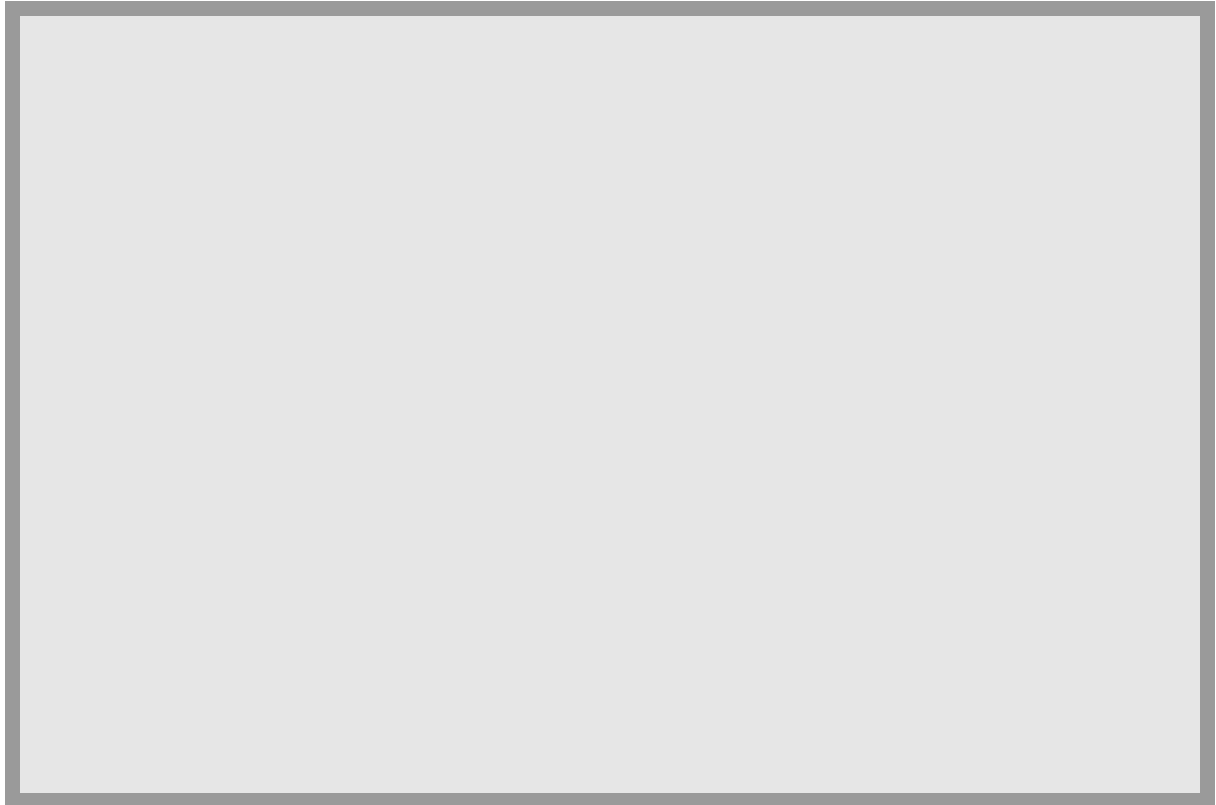


Figura 3: Exemplo de gráfico de Burn-down

No dia seguinte, na próxima Daily, cada um relata seu progresso: por exemplo, Alice concluiu suas tarefas; João concluiu uma, mas encontrou um impedimento técnico na outra (e Maique se voluntaria para ajudá-lo após a reunião); Maique concluiu todas as suas tarefas e já puxou tarefas adicionais. Se, nesse dia, nenhum item do Product Backlog foi totalmente concluído (segundo a DoD), nenhum ponto é "queimado" no gráfico — mesmo que várias tarefas técnicas tenham avançado. Esse ciclo se repete até o último dia da sprint, quando ocorrem a Sprint Review e a Sprint Retrospective.

8.4 Sprint Review

Ocorre ao final de cada sprint. O objetivo é inspecionar o incremento (ou qualquer outro resultado da sprint) e, se necessário, adaptar o Product Backlog.

Participantes: o Scrum Team, além de usuários, clientes e stakeholders internos e externos convidados (marketing, colecionadores, proprietários de loja etc.). Se algum stakeholder relevante não puder comparecer, o Product Owner deve representar seus interesses. É essencial que estejam presentes, pois ao inspecionar o incremento eles podem fornecer feedback valioso, que pode levar a atualizações no Product Backlog.

Time-box: no máximo 4 horas para uma sprint de um mês.

Preparação: antes do evento, os Developers verificam quais itens do Product Backlog estão prontos de acordo com a DoD e os preparam para apresentação.

Pergunta importante: o trabalho não concluído deveria ser apresentado durante a Sprint Review? **Não** — apresentá-lo passaria a falsa impressão de progresso aos stakeholders e clientes.

O que a Sprint Review NÃO é: não é uma reunião de demonstração tradicional, nem uma oportunidade para os desenvolvedores simplesmente "mostrarem progresso" ou surpreenderem a plateia com slides e capturas de tela. Sem uma inspeção real do incremento em funcionamento, o evento perde seu propósito.

O que a Sprint Review realmente é: um momento de **transparência e colaboração** entre o Scrum Team e os principais stakeholders. Ideia central: dar aos stakeholders acesso real ao incremento sempre que possível — por exemplo, disponibilizando a URL de um site para testes reais, ou entregando um smartphone com a versão mais recente de um app (ou até o próprio APK, se o stakeholder tiver um dispositivo Android compatível). Durante o evento, os resultados da sprint são inspecionados, o progresso em direção à Meta do Produto é discutido, e o Product Backlog pode ser atualizado com base no feedback recebido.

8.5 Sprint Retrospective

Oportunidade para o Scrum Team refletir sobre seus métodos de trabalho e sua dinâmica em equipe. Trata-se de tornar a equipe melhor de forma contínua — podendo abordar problemas interpessoais, questões de qualidade do produto, adequação da Definition of Done, questões de produtividade, questões de processo, ou até incômodos práticos (como o horário das reuniões).

Momento: ocorre logo após a Sprint Review, concluindo a sprint.

Time-box: no máximo 3 horas para uma sprint de um mês.

Em geral, o objetivo é responder a três perguntas:

1. **O que deu certo durante a sprint?** Queremos identificar os pontos positivos para repeti-los tanto quanto possível.
2. **O que não deu certo?** Precisamos identificar os pontos negativos — mas não apenas identificá-los, e sim entender como podemos melhorar e combatê-los na próxima sprint.

3. **Como podemos melhorar?** Por exemplo, a equipe pode decidir alterar o horário da Daily Scrum para as 11h30. Vale notar que essa não é uma "tarefa do Scrum" a ser medida por esforço formal — é um compromisso de melhoria contínua da equipe.

9 Monitoramento de Progresso

O Scrum Guide **não exige** o uso de nenhuma prática específica de monitoramento visual da sprint — os gráficos a seguir são exemplos populares, não regras obrigatórias.

9.1 Gráfico de Burn-down

Representa o trabalho restante ao longo do tempo ("queimando" de cima para baixo).

- Eixo X: dias úteis da sprint.
- Eixo Y: trabalho restante (geralmente em story points).

Exemplo: se o total de pontos do Product Backlog selecionado para a sprint é 100, começamos no dia 0 com 100 pontos restantes. Após a primeira sprint (em um contexto multi-sprint até um grande lançamento), poderiam restar 85 pontos, e assim por diante, ao longo de várias sprints (por exemplo, até a sprint 8, quando um grande lançamento está previsto).

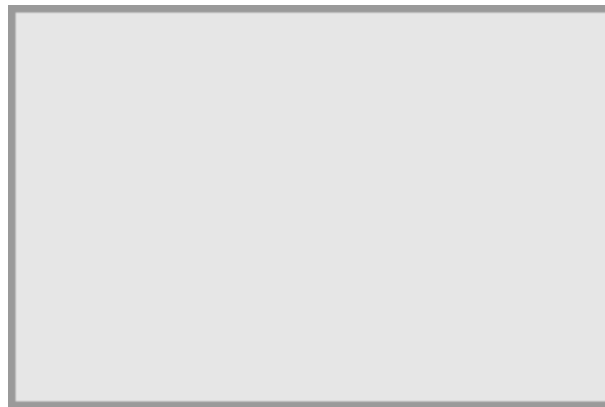


Figura 4: Exemplo de gráfico de Burn-down de release

Nesse tipo de gráfico, uma linha (por exemplo, vermelha) pode representar a trajetória ideal necessária para cumprir o prazo — indo de 100 até 0 na sprint prevista para o lançamento. Se a linha real (por exemplo, azul) estiver acima da linha ideal, a equipe está atrasada; se estiver abaixo ou alinhada, está no caminho certo. Vale lembrar que esse tipo de linha de referência com datas fixas só costuma fazer sentido em casos especiais, já que o Product Backlog é um artefato vivo e emergente — novos itens podem surgir e

outros podem ser removidos, fazendo o total de pontos aumentar ou diminuir ao longo do tempo.

Limitação importante: como o backlog é emergente, o número total de pontos pode aumentar durante o projeto (novos itens sendo adicionados), o que pode dar a falsa impressão de que a equipe "não queimou nada" durante certas sprints, mesmo tendo trabalhado normalmente. Essa é uma limitação conhecida dos gráficos de burn-down.

9.2 Gráfico de Burn-up

Semelhante ao burn-down, mas com duas linhas que sobem (de baixo para cima) em vez de descer.

- Uma linha representa o trabalho **concluído** acumulado.
- Outra linha representa o **escopo total** (que pode crescer ao longo do tempo).

Por exemplo: começando com 100 pontos no Product Backlog, na sprint 1 a equipe pode "queimar"(concluir) 10 pontos, enquanto 5 pontos novos são adicionados ao escopo total. Isso torna mais visível — em comparação ao burn-down — quando o escopo está crescendo, já que a linha de escopo total também é exibida explicitamente no gráfico.

9.3 Diagrama de Fluxo Cumulativo (Cumulative Flow Diagram)

Outra forma de visualizar o progresso, geralmente usada com um quadro de estados dos itens (por exemplo: "não pronto" e "pronto", ou estados intermediários como "a fazer", "em progresso" e "concluído"). O diagrama mostra a quantidade de itens em cada estado ao longo do tempo, ajudando a identificar gargalos no fluxo de trabalho.

Para leitura complementar oficial, recomenda-se a seção "Scrum Artifacts" do Scrum Guide.

10 Scrum em Larga Escala: o Nexus Guide

Quando várias equipes Scrum trabalham juntas no mesmo produto, o Scrum Guide "puro" não entra em detalhes sobre como coordenar esse cenário. Para isso, a Scrum.org publica um guia complementar específico: o **Nexus Guide**.

O Nexus Guide é baseado no Scrum Guide, mas o complementa, adicionando e modificando alguns eventos e artefatos. (Não é necessário dominar profundamente o Nexus para a certificação PSM I — existe uma certificação específica para isso, chamada *Scaled Professional Scrum*, ou SPS.)

Estrutura básica do Scrum em larga escala:

- Há vários Scrum Teams, cada um funcionando da forma padrão internamente.
- Porém, existe um **único Product Owner** para todos os times — o que representa uma carga de trabalho significativa. O PO pode (e normalmente deve) delegar parte do trabalho, mas continua sendo o responsável final (accountable), podendo dedicar o tempo que julgar necessário a cada equipe conforme suas necessidades.
- Ao final de cada sprint, deve haver **pelo menos um Incremento integrado**. Os Scrum Teams não precisam estar profundamente interligados no dia a dia — cada um se autogerencia e decide o que é melhor para si.

Principal desafio: lidar com as **dependências** entre as equipes. Esse desafio é tão relevante que o Nexus Guide criou uma responsabilidade específica para tratá-lo: o **Nexus Integration Team**.

10.1 Nexus Integration Team

Composto pelo Product Owner, por Scrum Masters das equipes envolvidas (quando aplicável) e por membros dos Scrum Teams. Sua função é resolver dependências e problemas de integração entre as equipes.

Como todas as equipes precisam contribuir para um único incremento integrado, elas precisam atender, no mínimo, a uma **Definition of Done comum**, criada pelo Nexus Integration Team. Isso não significa que cada Scrum Team esteja restrito apenas a esses critérios — cada equipe pode aplicar seus próprios critérios adicionais, mais rigorosos, formando sua própria DoD "local", desde que respeite a DoD global como o mínimo obrigatório.

10.2 Eventos do Nexus

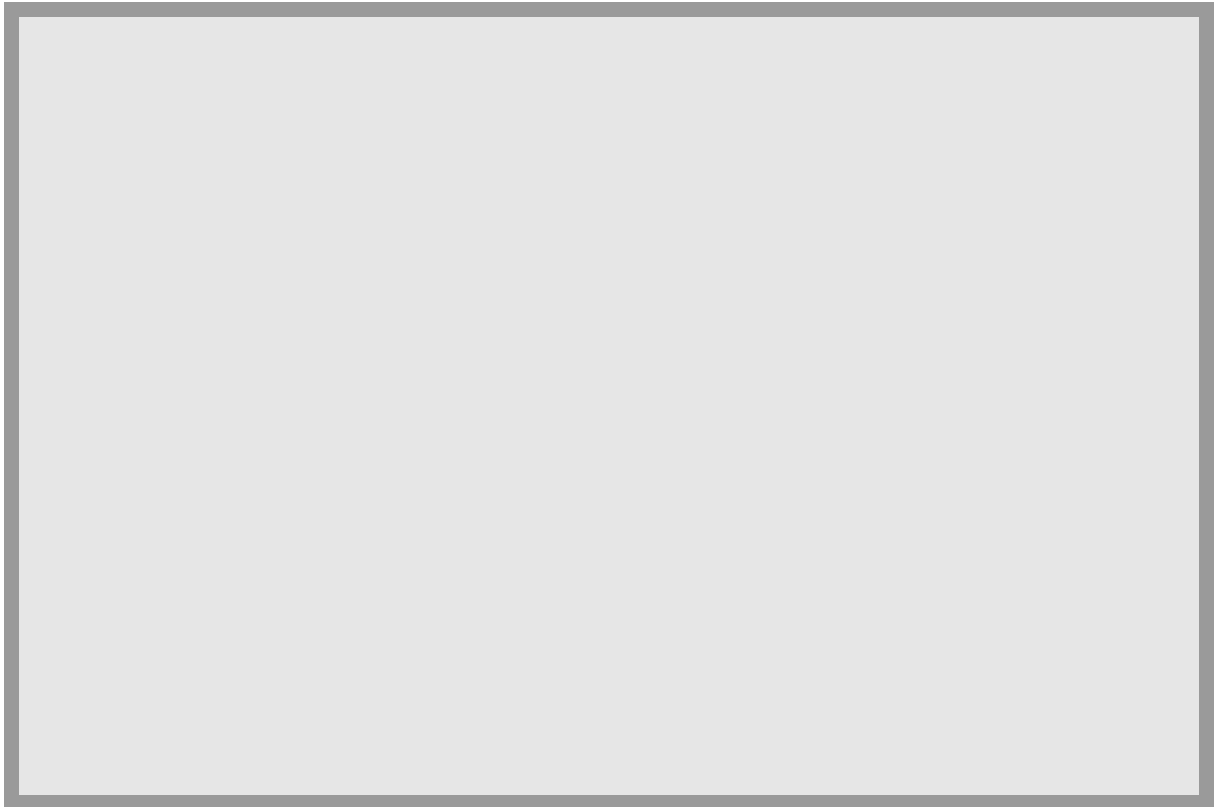


Figura 5: Estrutura do Nexus comparada ao Scrum tradicional

O Nexus adiciona os seguintes eventos, que complementam (mas não substituem, exceto quando indicado) os eventos de cada Scrum Team individual:

- **Nexus Sprint Planning:** o Product Owner se reúne com representantes de todas as equipes para planejar as atividades da sprint em conjunto. Desse evento nascem o **Nexus Sprint Backlog** e a **Nexus Sprint Goal**, que servem de referência para todos os Scrum Teams durante a sprint — embora, depois desse alinhamento inicial, cada equipe siga sua própria jornada de execução.
- **Nexus Daily Scrum:** complementa as Daily Scrums individuais, focando especificamente em dependências entre equipes.
- **Nexus Sprint Review:** este evento **substitui** as Sprint Reviews individuais de cada equipe, sendo realizado uma única vez para apresentar o incremento integrado a todos os stakeholders.

10.3 Component Teams vs. Feature Teams

Nem o Scrum Guide nem o Nexus Guide tratam formalmente desse tópico, mas é um conceito relevante na prática de escalonamento: equipes organizadas por **funcionalidade**

(*feature teams* — capazes de entregar uma funcionalidade completa, ponta a ponta) costumam ser consideradas mais alinhadas ao espírito ágil do que equipes organizadas por **componente técnico** (*component teams* — cada uma responsável por uma camada ou módulo específico do sistema), pois as *feature teams* reduzem dependências entre equipes e entregam valor de forma mais independente.

10.4 Frequência de Sprints em múltiplos Scrum Teams

Idealmente, todos os Scrum Teams de um Nexus devem estar sincronizados, iniciando e terminando suas sprints no mesmo dia — isso torna o trabalho de coordenação muito mais simples.

No entanto, pode haver equipes que só conseguem entregar algo útil mensalmente (por exemplo, uma equipe que trabalha principalmente com hardware) e outras que conseguem entregar semanalmente. Por que a segunda equipe deveria perder a oportunidade de inspecionar e adaptar semanalmente? Nesse caso, as equipes **podem** ter durações de sprint diferentes.

No exemplo dado, o **Nexus Sprint** seria de um mês — ou seja, igual ao maior comprimento entre as sprints das equipes envolvidas. Os Scrum Teams se autoorganizam dentro dessa restrição, podendo ter sprints mais curtas, desde que os comprimentos mais curtos se distribuam uniformemente dentro do mais longo.

Exemplo importante: um Scrum Team **não poderia** ter uma sprint de 3 semanas nesse cenário. Por quê? Porque dividir 4 semanas por 3 não resulta em um número inteiro — na prática, isso significaria que as equipes só se sincronizariam a cada 6 semanas, o que viola a regra do Scrum de sprints de, no máximo, um mês.

Resumindo:

- Os Scrum Teams de um Nexus podem ter durações de sprint diferentes.
- Porém, as datas de início e término das sprints devem estar sincronizadas, respeitando uma cadência máxima de um mês.

11 Kanban

Kanban: método de melhoria contínua baseado na visualização do fluxo de trabalho, com origem no sistema de controle de estoque da Toyota. Diferente do Scrum, o Kanban não trabalha com sprints time-boxed fixas, mas sim com um fluxo contínuo de itens passando por colunas (estados) em um quadro visual, geralmente limitando o trabalho em progresso (*WIP* — *Work In Progress*) para evitar sobrecarga e identificar gargalos. É frequentemente citado como uma prática complementar ao Scrum, especialmente no contexto de gerenciamento visual de tarefas (scrumboards).

12 Certificação Profissional Scrum Master I (PSM I)

Normalmente, para se candidatar a vagas de Scrum em empresas, costuma-se pedir certificações como a **CSM** (*Certified ScrumMaster*) ou a **PSM** (*Professional Scrum Master*) — o foco aqui é a PSM.

A CSM é considerada menos acessível: exige um treinamento presencial ou online obrigatório antes da prova, além de precisar de renovação a cada 2 anos.

A **PSM** é oferecida pela Scrum.org, fundada pelo próprio criador do Scrum. É uma avaliação online e aberta, que qualquer pessoa pode fazer a qualquer momento — mais barata que a CSM e sem prazo de expiração.

12.1 Custo

Aproximadamente **200 USD** por tentativa.

12.2 Conteúdo da prova

Baseada no **Scrum Guide** (versão 2020).

12.3 Formato

- Duração: 60 minutos.
- 80 questões, nos formatos de múltipla escolha (ME), múltipla resposta (MR) e verdadeiro/falso (V/F).
- Pontuação mínima exigida: 85%.
- Idioma: inglês.

12.4 Dicas para a prova

- Leia a questão inteira com atenção.
- Preste atenção especial a palavras como: *não, poderia, deveria, recomenda, normalmente, obrigatório, sempre, nunca, melhor, pior, ideal* — e marque-as mentalmente ou no rascunho.
- Saiba distinguir **regras** do Scrum de **recomendações**: regras devem ser seguidas — se não forem, não se está realmente adotando o Scrum. Recomendações são sugestões que podem ou não ser aplicadas dependendo do contexto.
- Algumas perguntas pedem para escolher não apenas uma, mas duas das melhores opções — leia com atenção para não errar por desatenção.

- Use o processo de eliminação: descarte primeiro as opções claramente erradas.
- Cuidado com opções parcialmente corretas — os avaliadores costumam pedir a *melhor* resposta entre várias plausíveis.
- Quando não tiver certeza, faça um "chute fundamentado" (elimine o máximo de opções erradas possível antes de decidir).

Preparação complementar recomendada:

- Entender o papel do Scrum Master como facilitador, treinador (*coach*) e líder servidor.
- Cuidar do bem-estar físico antes da prova (alimentação, descanso).
- Fazer simulados oficiais, como o Scrum Open (disponibilizado gratuitamente pela Scrum.org), como forma de "aquecimento".
- Para quem for fazer a prova (ou simulados) e preferir ler em português, é possível usar o plugin de tradução do Google Chrome — lembrando que as questões originais estão em inglês.
- Focar o estudo na compreensão real dos valores do Scrum e dos princípios de auto-gestão, e não apenas na memorização.

Atenção especial — tradução de "accountability" e "responsibility": como já mencionado na Seção 12 (papéis do Scrum Team), essas duas palavras têm significados diferentes em inglês, mas costumam ser traduzidas da mesma forma para o português ("responsabilidade"). Recomenda-se: sempre que aparecer a palavra "responsabilidade" (ou variações, como "responsável") em uma questão traduzida, voltar ao texto original em inglês (por exemplo, usando o plugin de tradução) para verificar qual das duas palavras foi realmente usada — isso pode mudar completamente o sentido da questão.

12.5 Exemplos de questões (estilo "sniper")

P: O que o Cone da Incerteza representa?

R: Ele representa a evolução do conhecimento da equipe em relação ao projeto ao longo do tempo, tornando as estimativas progressivamente mais precisas — atuando em conjunto com a metodologia ágil como forma de reduzir os efeitos do VUCA.

P: Qual valor/princípio ágil afirma que horas extras devem ser evitadas?

R: O princípio do **ritmo sustentável** (desenvolvimento sustentável).

P: Verdadeiro ou falso — o Scrum é um meio ágil de entrega de trabalho que é tanto incremental quanto iterativo?

R: Verdadeiro.

P: Qual afirmação melhor descreve o Scrum?

R: Um framework leve para desenvolver, entregar e sustentar produtos complexos.

Compreender bem o Manifesto Ágil ajuda a entender o Scrum de forma mais profunda — e, consequentemente, a se sair melhor na avaliação.

13 Recapitulação Geral do Fluxo do Scrum

1. O **Product Backlog** é gerenciado pelo **Product Owner**, que garante que os itens do topo sejam os mais importantes.
2. No **Sprint Planning**, o Scrum Team elabora uma **Meta da Sprint** (Sprint Goal), e os Developers selecionam os itens do Product Backlog a serem desenvolvidos, formando o **Sprint Backlog** — que também contém o plano de ação para entregar o incremento.
3. Durante a sprint (no máximo 1 mês), os Developers usam a **Daily Scrum** para maximizar e otimizar seu trabalho em direção à Sprint Goal.
4. Quando algo palpável e usável é alcançado — desde que atenda à **Definition of Done** — nasce um **Incremento**.
5. Um incremento é um passo concreto em direção à **Meta do Produto** (Product Goal) — o objetivo de longo prazo que a equipe busca alcançar.
6. Ao final da sprint, o Scrum Team realiza a **Sprint Review** com os stakeholders convidados, coletando feedback que pode levar à atualização do Product Backlog.
7. Em seguida, o Scrum Team realiza a **Sprint Retrospective**, na qual adapta suas interações, processos e até mesmo a Definition of Done — impactando diretamente o próximo incremento.
8. O **Scrum Master** apoia todo esse processo, que se repete continuamente até que o trabalho termine ou o orçamento se esgote.

13.1 Objetivo principal do Scrum: entregar valor

Como entregamos valor? Começamos com um conjunto de ideias e as transformamos em valor por meio do trabalho. Geralmente, essas ideias estariam em documentos de requisitos tradicionais — mas no Scrum, onde elas ficam? No **Product Backlog**. Ou seja, o Product Backlog não é apenas uma "documentação": é um instrumento de **alinhamento e transparência** entre o pessoal de negócios e o pessoal técnico.

A viabilidade de entregar valor está diretamente ligada à qualidade do backlog — e quem é responsável por essa qualidade é o **Product Owner**. Ele não é um "ditador": colabora com o Scrum Team e com outros stakeholders, mas é o **único** responsável (accountable) pelo gerenciamento do Product Backlog. Um Product Owner que não desempenha bem esse papel afeta diretamente o sucesso do produto em termos de valor entregue.

13.2 Considerações finais

Na prática, algumas empresas dizem "usar Scrum" mas não seguem o framework exatamente como descrito no Scrum Guide — por exemplo, times em que o Product Owner não participa ativamente do processo. É importante lembrar que o Scrum evoluiu e se espalhou para além do desenvolvimento de software, sendo aplicado hoje em diversas áreas (marketing, hardware, educação, entre outras).