



Processo de Desenvolvimento de Software

Modelo Tradicional, Manifesto Ágil, Scrum

Professora: M.Sc. Gabriela Martins de Jesus

E-mail: gabriela.jesus@uvv.br

Agenda

1. Modelo Tradicional

2. Manifesto Ágil

a. História

b. Valores

c. Princípios

3. Framework Scrum



UNIVERSIDADE
VILA VELHA
ESPIRITO SANTO

Nivelamento

Responda

1. Sobre o modelo Cascata, explique:
 - a. Suas principais características;
 - b. Seus principais benefícios;
 - c. Seus principais problemas;
 - d. Fases e o que acontece em cada uma.

2. Explique o que é o Manifesto Ágil



Modelo Tradicional Cascata / *Waterfall*

O Modelo Cascata (Waterfall)

O Modelo **Cascata**, também conhecido como **Waterfall** ou **Top-Down**, é um modelo de gestão de projeto de desenvolvimento de software criado em **1970** por **Winston W. Royce**. Ainda é um dos modelos mais utilizados em desenvolvimento de software e foi o responsável pelo surgimento de muitos outros modelos.

Esse modelo é definido como uma **abordagem linear** quando o assunto é a Gestão de Projetos, onde as partes interessadas e os clientes possuem os **requisitos** coletados no **início** do projeto e, em seguida, é criado um **plano** com uma sequência para abranger os requisitos e se chegar ao **resultado final**, atendendo às expectativas que foram coletadas desde o início de todo o projeto a ser desenvolvido.

As etapas do gerenciamento de projetos no Modelo Cascata seguem a mesma lógica que consta no **PMBOK** (iniciação, planejamento, execução, monitoramento e controle e encerramento). Esse modelo de gerenciamento de projetos, assim como o Guia PMBOK, possui 5 fases que foram definidas por Royce e devem ser seguidas, sendo elas: requerimento, projeto, implementação, verificação e manutenção.

Para que você consiga utilizar esse modelo de gestão é importante entender o que é e o que fazer em cada uma dessas fases. Veja a seguir um pouco sobre as etapas desse modelo e o que cada uma delas contêm:



O Modelo Cascata (Waterfall)

Como se trata de um plano que possui uma sequência, ao migrar para a próxima etapa não é possível regredir para a etapa anterior e por isso é necessário muito cuidado quanto ao avanço das etapas. Essa é uma desvantagem presente no modelo tradicional de gestão de projetos. Com a intenção de resolver esse problema de poder voltar para uma etapa ou tarefa anterior, permitindo assim a alteração de tarefas, o Modelo Cascata Revisto foi desenvolvido, onde a única diferença dele para o tradicional é a possibilidade de poder fazer essas alterações com base em ocorrências que tenham surgido no meio do andamento do projeto.

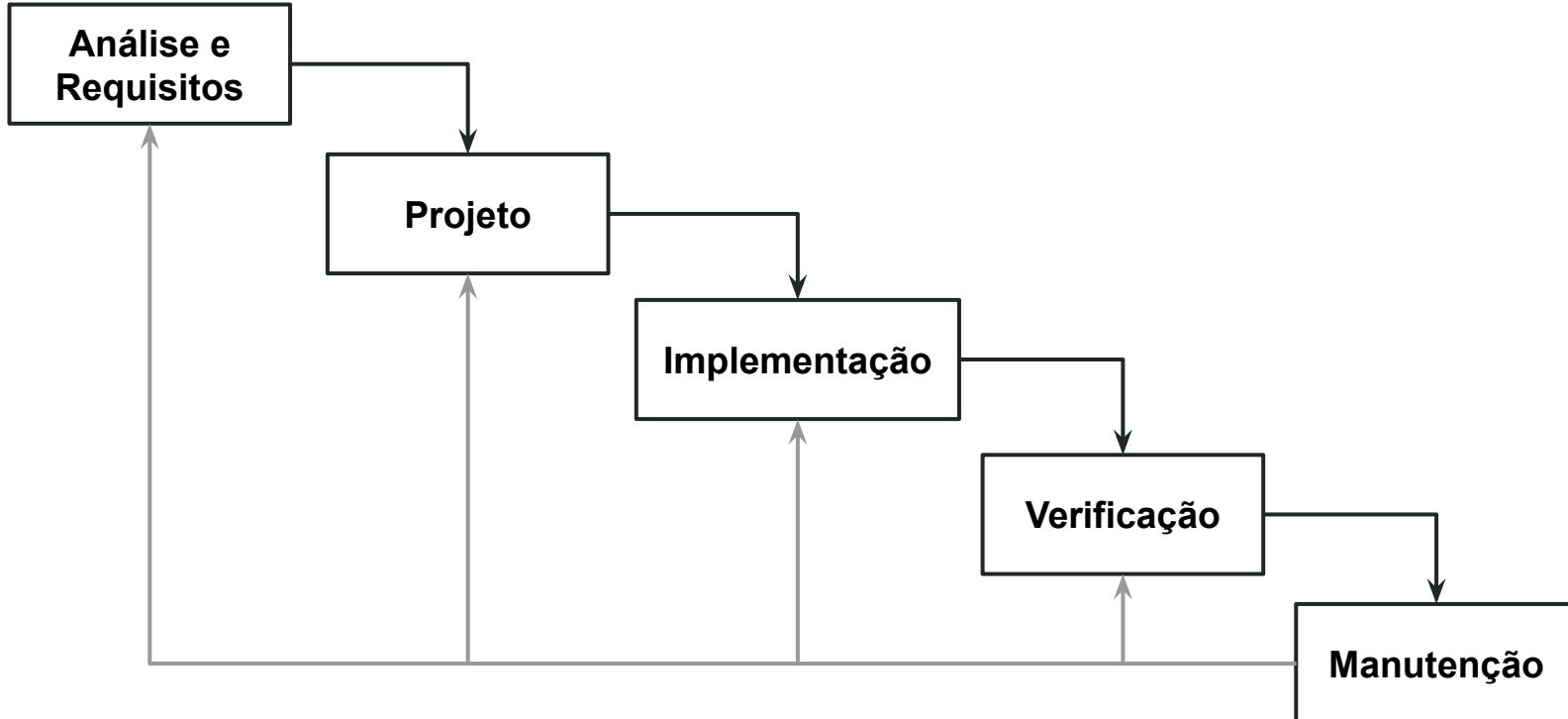
Para regredir ao estágio passado é necessário voltar o projeto todo do início, independente do ponto ao qual gostaríamos de retornar ou não, pois uma etapa depende e afeta a outra, fazendo-se necessário assim reiniciar o projeto e gerando consequentemente muitas vezes um atraso na entrega.

Essa metodologia de gestão até hoje é bem aceita por conta da facilidade administrativa, estabilidade no desenvolvimento do projeto, facilidade na inspeção e controle e por proporcionar maior organização para que a equipe e os gerentes de projetos possam desenvolver um bom projeto.

Esse modelo é um dos mais importantes e utilizados por servirem como uma base para a criação de outros modelos de gestão do desenvolvimento de software, sendo uma base também para os modelos de gestão de projetos mais modernos.

A abordagem dessa metodologia funciona muito bem por ser um modelo voltado para a orientação de documentação, podendo ser essa documentação textos, gráficos, simulações e outros.

O Modelo Cascata (Waterfall)



O Modelo Cascata (Waterfall)

Análise e Requisitos

Na fase do requerimento são definidos todos os requisitos necessários para o desenvolvimento das tarefas para que o objetivo final seja alcançado.

Como requisitos devemos identificar os serviços inclusos, limitações e objetivos. Aqui iremos reunir todas as especificações do projeto, podendo fazer com que todas as outras etapas sejam planejadas de forma independente umas das outras, mas de acordo com o desejo dos interessados.

Com os requisitos definidos é necessário estabelecer uma conexão deles com as etapas seguintes, para que eles possam ser úteis e tenham uma sequência lógica.

Além disso, a primeira fase abrange também toda a parte de documentação e estudo de viabilidade.

O Modelo Cascata (Waterfall)

Projeto

Na fase de projeto, conhecida também como design, é feita a tradução dos requisitos do software para um conjunto de representações que podem ser avaliadas quanto à qualidade, antes que a implementação (codificação se inicie).

Dessa forma, na fase de projeto já conseguimos modelar boa parte do nosso projeto.

Implementação

Uma vez compreendido os requisitos especificados nas fases anteriores, os desenvolvedores os implementam (codificação) para atender as necessidades definidas nas especificações.

Na codificação, é feita a tradução das representações do projeto para uma linguagem “artificial”, resultando em instruções executáveis pelo computador.

O Modelo Cascata (Waterfall)

Verificação

Concentra-se:

- nos aspectos lógicos internos do software, garantindo que todas as instruções tenham sido testadas (Teste Estrutural)
- nos aspectos funcionais externos, para descobrir erros e garantir que a entrada definida produza resultados que concordam com os esperados (Teste Funcional)

Depois de implementado e testado, o cliente do projeto faz uma análise do que foi entregue com o objetivo de garantir que os requisitos definidos na fase inicial do projeto sejam cumpridos. Essa análise é feita com a liberação do projeto para o cliente.

O Modelo Cascata (Waterfall)

Manutenção

Provavelmente o software deverá sofrer mudanças depois que for entregue ao cliente, que podem ter como causas:

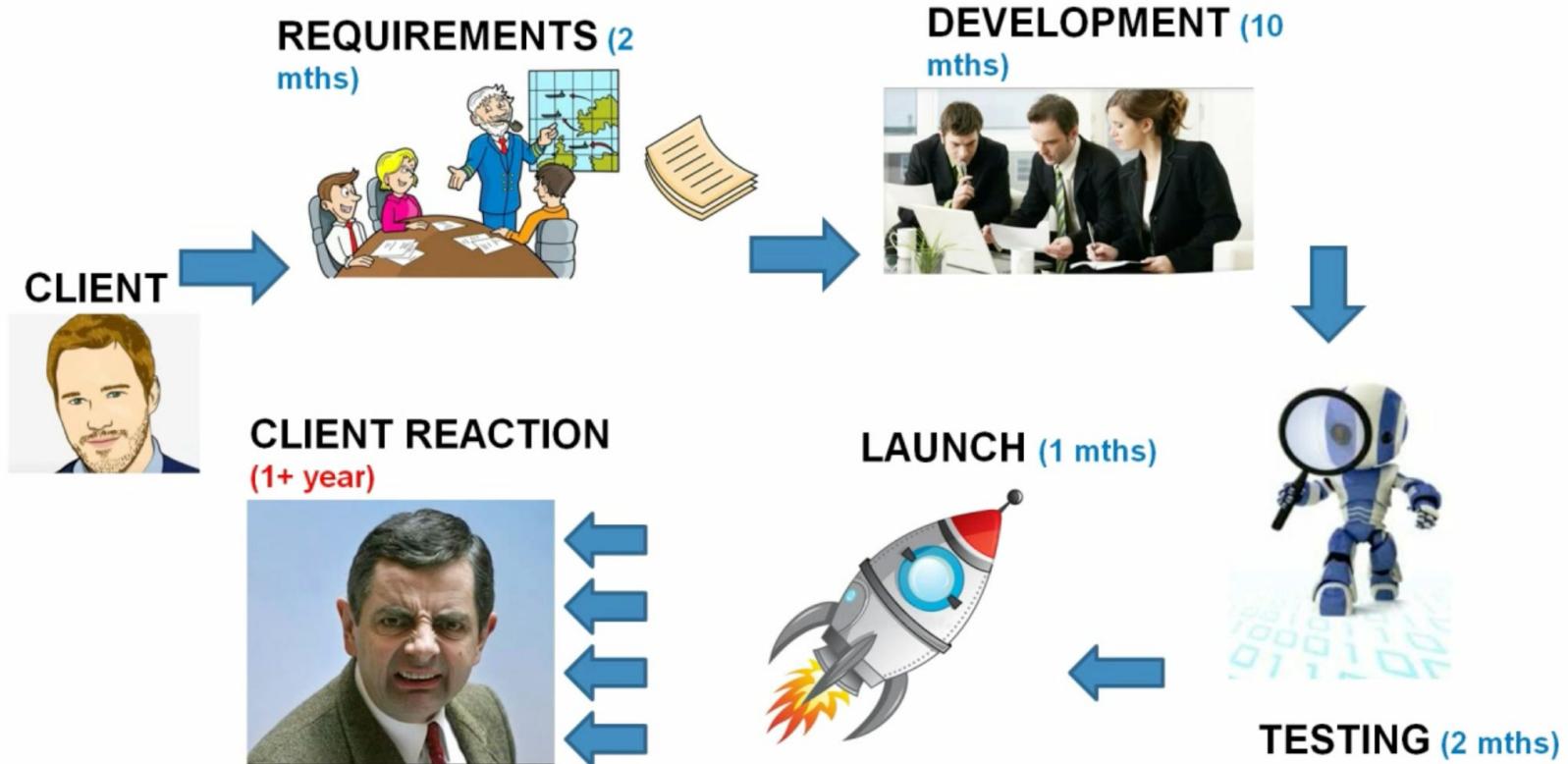
- erros, adaptação do software para acomodar mudanças em seu ambiente externo, exigência do cliente para acréscimos funcionais de desempenho, mudanças em leis, etc

Durante a etapa de manutenção o cliente está fazendo uso do produto de forma regular e isso permite que bugs possam ser descobertos, assim como também a inadequação de recursos e vários outros erros que serão corrigidos nessa fase de manutenção.

A aplicação das correções é realizada pela equipe desenvolvedora do projeto, onde as correções vão sendo aplicadas aos poucos até que “não haja mais erros” e que o cliente esteja satisfeito com o projeto.

Há também situações onde a manutenção é realizada por outra empresa ou equipe distinta da que inicialmente desenvolveu e entregou a solução ao cliente. Por isso a importância de se ter uma documentação atualizada e correta.

O Modelo Cascata (Waterfall)



Problemas do Modelo Cascata

- 💣 Projetos reais raramente seguem o fluxo sequencial que o Modelo propõe;
- 💣 É difícil (senão impossível) estabelecer explicitamente e em detalhes todos os requisitos logo no início do projeto, pois sempre existe uma incerteza natural que vai sendo descoberta e preenchida ao longo do processo;
- 💣 O cliente deve ter paciência. Uma versão executável do software só fica disponível em uma etapa avançada do desenvolvimento, quase no fim do projeto.

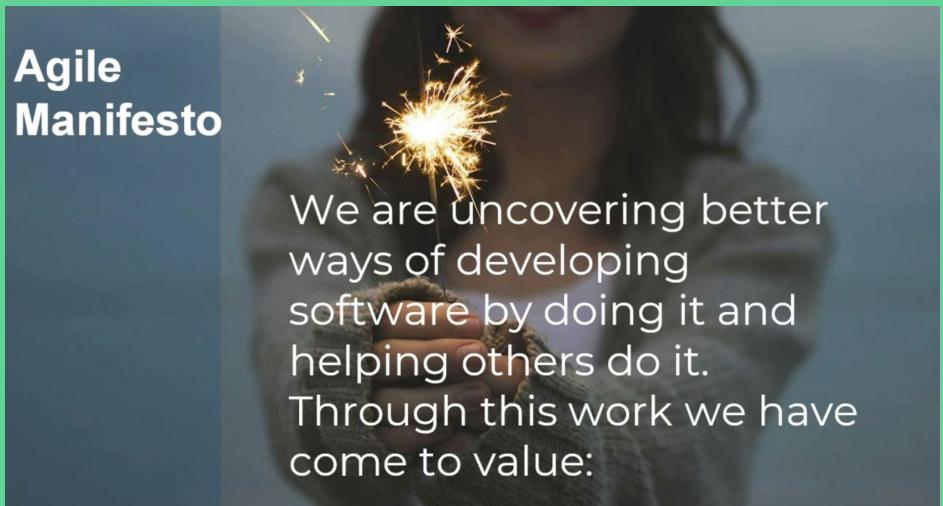
Contribuições do Modelo Cascata

✓ *Embora o Modelo Cascata tenha fragilidades, ele é significativamente melhor do que uma abordagem casual ao desenvolvimento de software*

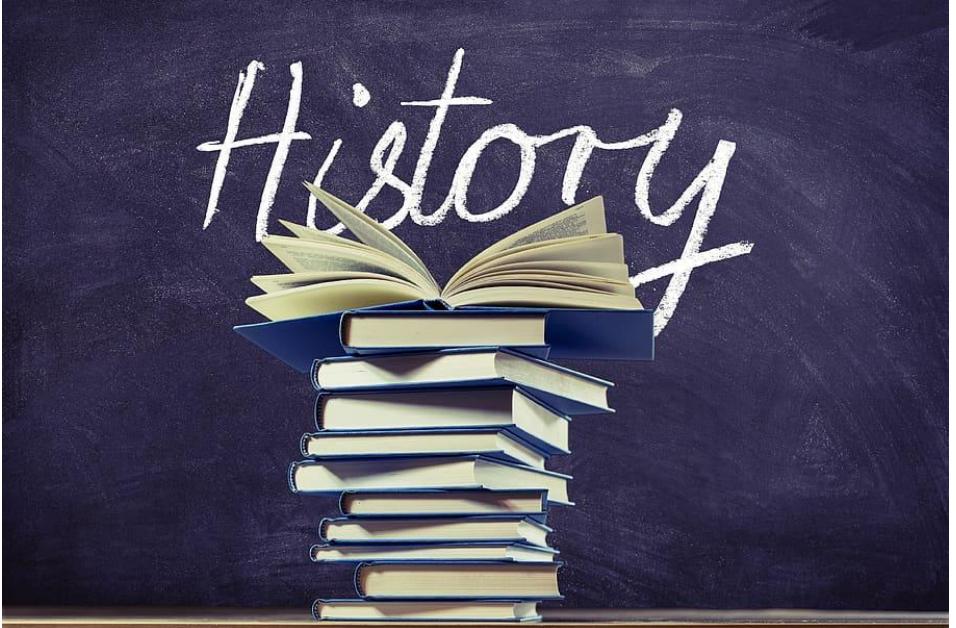
O Modelo Cascata trouxe contribuições importantes para o processo de desenvolvimento de software:

- ★ Imposição de disciplina, planejamento e gerenciamento;
- ★ Implementação do produto deve ser postergada até que os objetivos tenham sido completamente entendidos. - retrabalho

Manifesto Ágil



História do Manifesto Ágil



History: The Agile Manifesto

On February 11-13, 2001, at The Lodge at Snowbird ski resort in the Wasatch mountains of Utah, seventeen people met to talk, ski, relax, and try to find common ground—and of course, to eat. What emerged was the Agile ‘Software Development’ Manifesto. Representatives from Extreme Programming, SCRUM, DSDM, Adaptive Software Development, Crystal, Feature-Driven Development, Pragmatic Programming, and others sympathetic to the need for an alternative to documentation driven, heavyweight software development processes convened.

Now, a bigger gathering of organizational anarchists would be hard to find, so what emerged from this meeting was symbolic—a *Manifesto for Agile Software Development*—signed by all participants. The only concern with the term *agile* came from Martin Fowler (a Brit for those who don’t know him) who allowed that most Americans didn’t know how to pronounce the word ‘agile’.

Alistair Cockburn’s initial concerns reflected the early thoughts of many participants. “I personally didn’t expect that this particular group of agilites to ever agree on anything substantive.” But his post-meeting feelings were also shared. “Speaking for myself, I am delighted by the final phrasing [of the Manifesto]. I was surprised that the others appeared equally delighted by the final phrasing. So we did agree on something substantive.”

Naming ourselves “The Agile Alliance,” this group of independent thinkers about software development, and sometimes competitors to each other, agreed on the *Manifesto for Agile Software Development* displayed on the title page of this web site.

But while the Manifesto provides some specific ideas, there is a deeper theme that drives many, but not all, to be sure, members of the alliance. At the close of the two-day meeting, Bob Martin joked that he was about to make a “mushy” statement. But while tinged with humor, few disagreed with Bob’s sentiments—that we all felt privileged to work with a group of people who held a set of compatible values, a set of values based on trust and respect for each other and promoting organizational models based on people, collaboration, and building the types of organizational communities in which we would want to work. At the core, I believe Agile Methodologists are really about “mushy” stuff—about delivering good products to customers by operating in an environment that does more than talk about “people as our most important asset” but actually “acts” as if people were the most important, and love the word ‘asset’. So in the final analysis, the meteoric rise of interest in—and sometimes tremendous criticism of—Agile Methodologies is about the mushy stuff of values and culture.

For example, I think that ultimately, Extreme Programming has mushroomed in use and interest, not because of pair-programming or refactoring, but because, taken as a whole, the practices define a developer community freed from the baggage of Dilbertesque corporations. Kent Beck tells the story of an early job in which he estimated a programming effort of six weeks for two people. After his manager reassigned the other programmer at the beginning of the project, he completed the project in twelve weeks—and felt terrible about himself! The boss—of course—harangued Kent about how slow he was throughout the second six weeks. Kent, somewhat despondent because he was such a “failure” as a programmer, finally realized that his original estimate of 6 weeks was extremely accurate—for 2 people—and that his “failure” was really the manager’s failure, indeed, the failure of the standard “fixed” process mindset that so frequently plagues our industry.

This type of situation goes on every day—marketing, or management, or external customers, internal customers, and, yes, even developers—don’t want to make hard trade-off decisions, so they impose irrational demands through the imposition of corporate power structures. This isn’t merely a software development problem, it runs throughout Dilbertesque organizations.

In order to succeed in the new economy, to move aggressively into the era of e-business, e-commerce, and the web, companies have to rid themselves of their Dilbert manifestations of make-work and arcane policies. This freedom from the innanities of corporate life attracts proponents of Agile Methodologies, and scares the begebers (you can’t use the word ‘shit’ in a professional paper) out of traditionalists. Quite frankly, the Agile approaches scare corporate bureaucrats—at least those that are happy pushing process for process’ sake versus trying to do the best for the ‘customer’ and deliver something timely and tangible and “as promised”—because they run out of places to hide.

The Agile movement is not anti-methodology, in fact, many of us want to restore credibility to the word methodology. We want to restore a balance. We embrace modeling, but not in order to file some diagram in a dusty corporate repository. We embrace documentation, but not hundreds of pages of never-maintained and rarely-used tomes. We plan, but recognize the limits of planning in a turbulent environment. Those who would brand proponents of XP or SCRUM or any of the other Agile Methodologies as “hackers” are ignorant of both the methodologies and the original definition of the term hacker.

The meeting at Snowbird was incubated at an earlier get together of Extreme Programming proponents, and a few “outsiders,” organized by Kent Beck at the Rogue River Lodge in Oregon in the spring of 2000. At the Rogue River meeting attendees voiced support for a variety of “Light” methodologies, but nothing formal occurred. During 2000 a number of articles were written that referenced the category of “Light” or “Lightweight” processes. A number these articles referred to “Light methodologies, such as Extreme Programming, Adaptive Software Development, Crystal, and SCRUM.” In conversations, no one really liked the moniker “Light”, but it seemed to stick for the time being.

In September 2000, Bob Martin from Object Mentor in Chicago, started the next meeting ball rolling with an email: “I’d like to convene a small (two day) conference in the January to February 2001 timeframe here in Chicago. The purpose of this conference is to get all the lightweight method leaders in one room. All of you are invited, and I’d be interested to know who else I should approach.” Bob set up a Wiki site and the discussions raged.

Early on, Alistair Cockburn weighed in with an epistle that identified the general disinglement with the word ‘Light’: “I don’t mind the methodology being called light in weight, but I’m not sure I want to be referred to as a lightweight attending a lightweight methodologists meeting. It somehow sounds like a bunch of skinny, feble-minded lightweight people trying to remember what it is.”

The fiercest debate was over location! There was serious concern about Chicago in wintertime—cold and nothing fun to do; Snowbird, Utah—cold, but fun things to do, at least for those who ski on their heads like Martin Fowler tried on day one; and Anguilla in the Caribbean—warm and fun, but time consuming to get to. In the end, Snowbird and skiing won out; however, a few people—like Ron Jeffries—want a warmer place next time.

We hope that our work together as the Agile Alliance helps others in our profession to think about software development, methodologies, and organizations, in new—more agile—ways. If so, we’ve accomplished our goals.

Jim Highsmith, for the Agile Alliance

©2001 Jim Highsmith

[Return to Manifesto](#)

O que é "Agilidade"

Agilidade é um conjunto de crenças que guiam a tomada de decisão.

Agilidade é uma filosofia, ou seja, um conjunto de valores e princípios.

História

Um grupo de 17 pessoas em 2001 lideraram o manifesto em busca de uma maneira colaborativa de desenvolver software



Valores e Princípios



Individuals and interactions	over processes and tools
Working software	over comprehensive documentation
Customer collaboration	over contract negotiation
Responding to change	over following a plan

Manifesto para Desenvolvimento Ágil de Software

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

Indivíduos e interações mais que processos e ferramentas
Software em funcionamento mais que documentação abrangente
Colaboração com o cliente mais que negociação de contratos
Responder a mudanças mais que seguir um plano

Ou seja, mesmo havendo valor nos itens à direita, valorizamos mais os itens à esquerda.

Kent Beck
Mike Beedle
Arie van Bennekum
Alistair Cockburn
Ward Cunningham
Martin Fowler

James Grenning
Jim Highsmith
Andrew Hunt
Ron Jeffries
Jon Kern
Brian Marick

Robert C. Martin
Steve Mellor
Ken Schwaber
Jeff Sutherland
Dave Thomas

Os 4 Valores Ágeis

1. Indivíduos e interações mais que processos e ferramentas

O manifesto destaca a importância das pessoas logo início do manifesto, em seu primeiro valor. Dessa forma, reforça-se que as pessoas que estão atuando na execução das tarefas desenvolvendo o produto/serviço são mais importantes do que os processos e as ferramentas. Os princípios da Gestão 3.0 se apoia sobre esse valor.

Portanto, entende-se que quem gera valor e executa a estratégia da Organização são as pessoas. As ações são realizadas conforme a interação, ou seja, comunicação e alinhamento entre stakeholders, time, e outros papéis envolvidos com foco na geração de valor para o cliente e sucesso do Negócio.

Considerando a importância de se ter interações com objetivos claros, diversos modelos de desenvolvimento ágil implementaram ritos/cerimônias/cadências

2. Software em funcionamento mais que documentação abrangente

Ou seja, focar maior esforço para planejamento, desenvolvimento e entrega contínua do que em detalhamentos e diversos tipos de documentações.

Entregar o produto ou serviço para atender às necessidades do cliente no tempo exato é prioridade. “Não adianta fazer o gol após o apito final”. De que adianta entregar o projeto e-commerce, ou uma funcionalidade de promoção para o Natal, dia 26 de dezembro? Já passou a janela de oportunidade de geração de valor para o cliente que, por sua vez, perdeu para a concorrência.

Para que se tenha entregas mais rápidas e no tempo exato, a gestão do projeto deve fazer uso de métricas e análise de dados.

3. Colaboração com o cliente mais que negociação de contratos

Como no item 2, este valor descreve a importância de focar em colaborar com as necessidades do cliente. Quando o time consegue alcançar seus objetivos com o produto/serviço desenvolvido e entregue, certamente o cliente ficará satisfeito e novos contratos (projetos) terão mais chances de serem realizados. Do contrário, o cliente buscará um novo fornecedor.

Ao invés de desperdiçar tempo detalhando com base em suposições, a colaboração entre time e cliente se destaca por envolver e centrar no cliente durante todo o projeto.

Atenção: não se trata em eliminar documentações e contratos, mas em se ter ciclos contínuos de feedback em períodos curtos que viabilizem adaptação e entregas mais aderentes às necessidades do cliente.

4. Responder a mudanças mais que seguir um plano

O backlog do produto é uma lista priorizada das necessidades do cliente, ou seja, uma lista dos requisitos. É um documento vivo e deve ser constantemente revisado, atualizado e (re)priorizado a cada final de período de entregas, conforme o feedback do cliente.

Pode ser um grave erro estabelecer um plano e segui-lo de forma rígida, pois deixa de considerar que mudanças ocorrem em todo projeto e a qualquer momento.

Em contrapartida, ao estabelecer uma estratégia dinâmica e adaptativa por meio da melhoria contínua, aumenta-se a chance de sucesso do projeto e faz com que o produto/serviço continue sendo priorizado pelo cliente.

12 Princípios do Manifesto Ágil



Princípios por trás do Manifesto Ágil

Nós seguimos estes princípios:

Nossa maior prioridade é satisfazer o cliente através da entrega contínua e antecipada de software com valor agregado.

Mudanças nos requisitos são bem-vindas, mesmo tarde, no desenvolvimento.

Processos ágeis tiram vantagem das mudanças visando vantagem competitiva para o cliente.

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

Pessoas de negócios e desenvolvedores devem trabalhar diariamente em conjunto por todo o projeto.

Construa projetos em torno de indivíduos motivados. Dê a eles o ambiente e o suporte necessário e confie neles para fazer o trabalho.

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

Software funcionando é a medida primária de progresso.

Os processos ágeis promovem desenvolvimento sustentável. Os patrocinadores, desenvolvedores e usuários devem ser capazes de manter um ritmo constante indefinidamente.

Continua atenção à excelência técnica e bom design aumenta a agilidade.

Simplicidade—é a arte de maximizar a quantidade de trabalho não realizado—é essencial.

As melhores arquiteturas, requisitos e designs emergem de equipes auto-organizáveis.

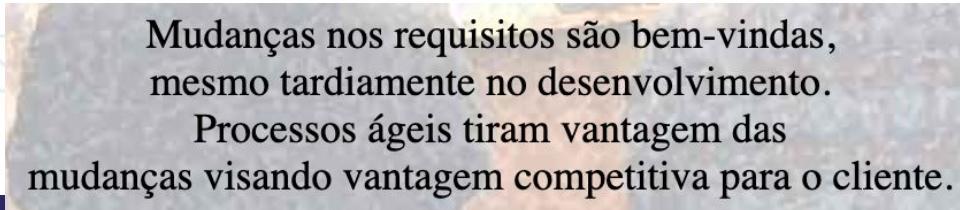
Em intervalos regulares, a equipe reflete sobre como se tornar mais eficaz e então refina e ajusta seu comportamento de acordo.

1 “Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.”

Nossa maior prioridade é satisfazer o cliente através da entrega contínua e adiantada de software com valor agregado.

2

Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.



Mudanças nos requisitos são bem-vindas, mesmo tardivamente no desenvolvimento.

Processos ágeis tiram vantagem das mudanças visando vantagem competitiva para o cliente.

3

“Deliver working software frequently, from a

couple of weeks to a couple of months, with a

preference to the shorter timescale.”

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

**4 “Business people and
developers must work
together daily throughout
the project.”**

Pessoas de negócio e desenvolvedores devem trabalhar
diariamente em conjunto por todo o projeto.

5

**"Build projects around
motivated individuals."**

**Give them the environment
and support they need,
and trust them to get the job
done."**

Construa projetos em torno de indivíduos motivados.
Dê a eles o ambiente e o suporte necessário
e confie neles para fazer o trabalho.

6

"The most efficient and effective method of conveying information to and within a development team is face-to-face conversation"

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

7

“Working software is the primary measure of progress.”

Software funcionando é a medida primária de progresso.

8

“Agile processes promote sustainable development.

The sponsors, developers, and users should be able to maintain a constant pace indefinitely.”

Os processos ágeis promovem desenvolvimento sustentável. Os patrocinadores, desenvolvedores e usuários devem ser capazes de manter um ritmo constante indefinidamente.

9

“Continuous attention to technical excellence and good design enhances agility.”

Contínua atenção à excelência técnica e bom design aumenta a agilidade.

**10 “Simplicity--the art of
maximizing the amount
of work not done--is
essential.”**

Simplicidade--a arte de maximizar a quantidade de trabalho não realizado--é essencial.

11

***The best architectures,
requirements, and designs
emerge from self-
organizing teams.***

As melhores arquiteturas, requisitos e designs
emergem de equipes auto-organizáveis.

12 “*At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.*”

Em intervalos regulares, a equipe reflete sobre como se tornar mais eficaz e então refina e ajusta seu comportamento de acordo.



Framework Scrum

História do Scrum

O nome surgiu do **Rugby**, esporte coletivo de intenso contato físico originário da Inglaterra como uma variação do futebol.

Scrum, ou “formação ordenada”, é uma situação frequente no rugby, que possui diversos reinícios da partida.



Scrum / Sprint

Plan

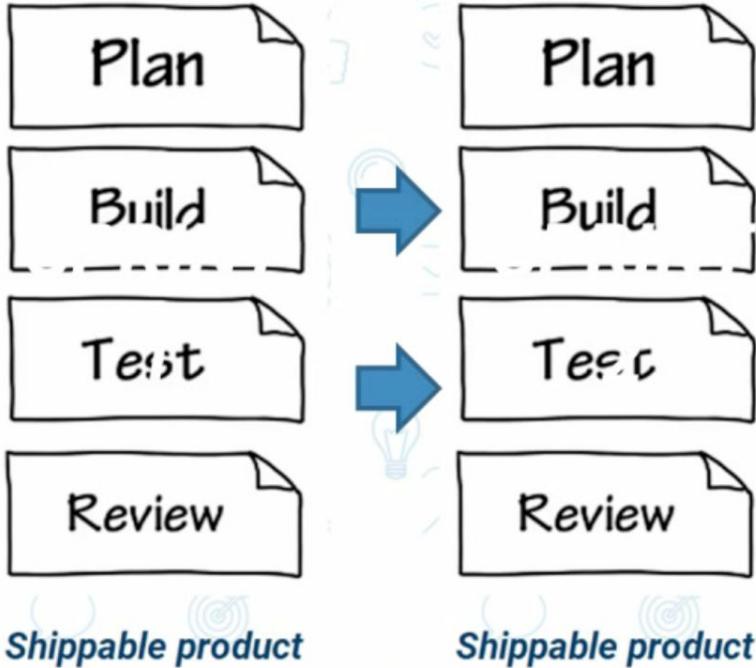
Build

Test

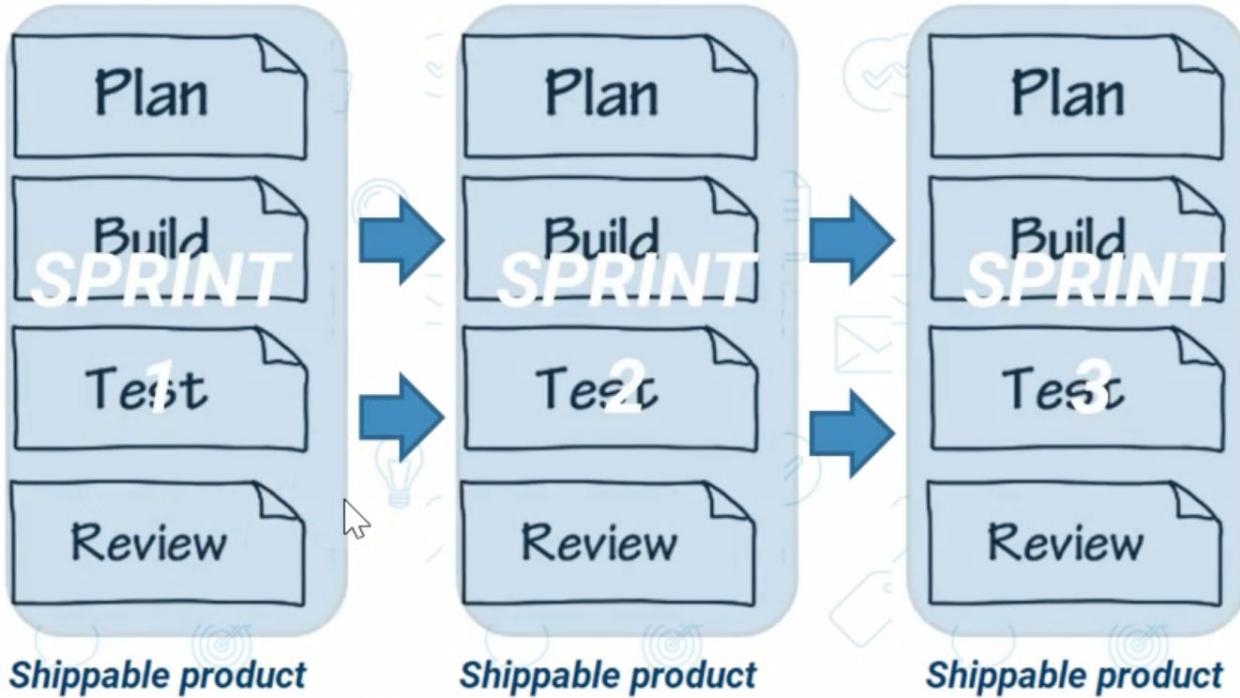
Review

Shippable product

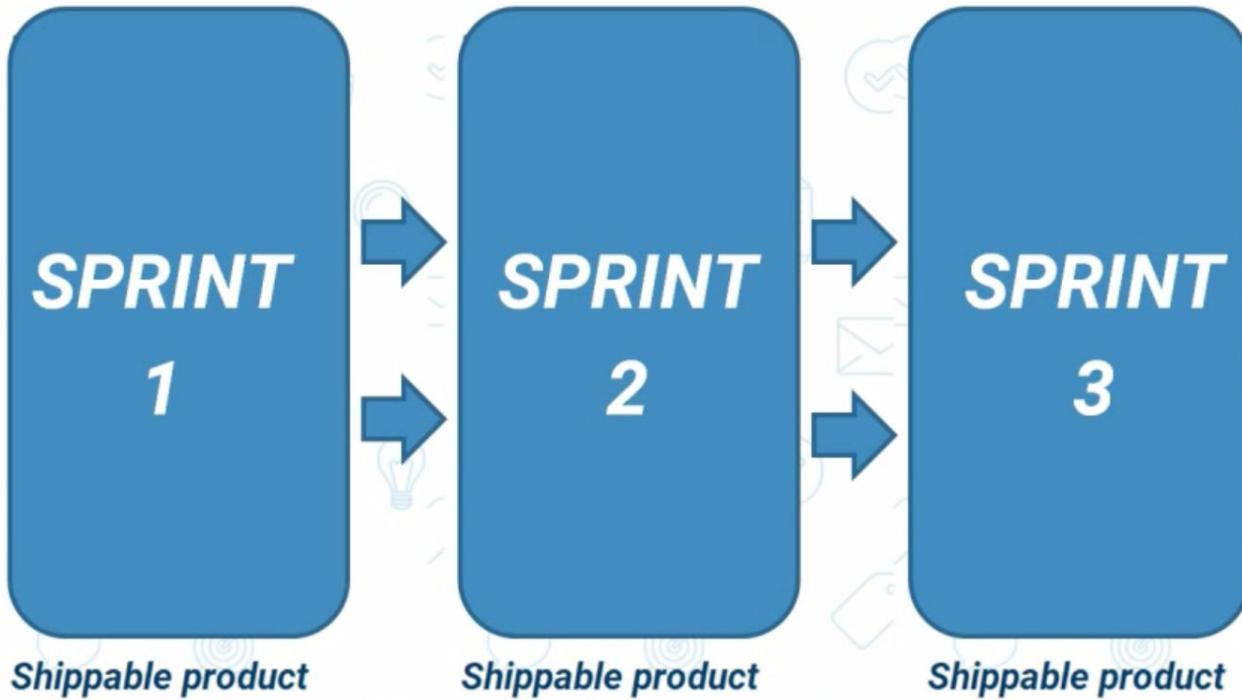
Scrum / Sprint



Scrum / Sprint



Scrum / Sprint



História do Scrum

Foi concebido para projetos em empresas de fabricação de automóveis e produtos de consumo.

Utilizado desde 1986 com sucesso em milhares de projetos e em centenas de organizações.

É gerencial e **NÃO** dita **COMO** o time de desenvolvimento deve realizar suas atividades, tampouco **O QUE FAZER** em todas as situações, pois parte do princípio de que o time tem autonomia e é auto-gerenciável.

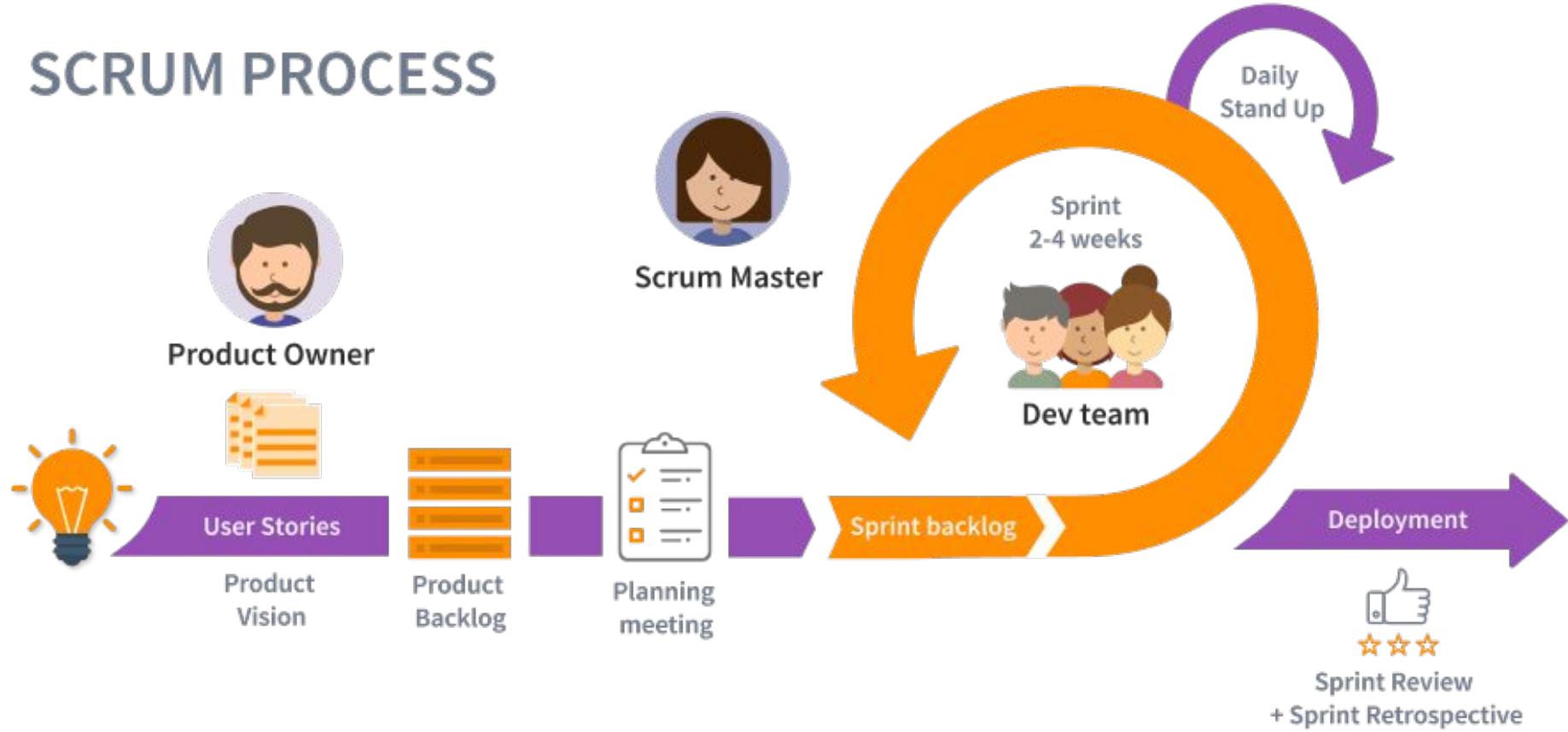
Não garante o sucesso completo do projeto, pois depende de maior administração do projeto em suas etapas, mas garante que o trabalho é focado aos resultados de maior valor agregado.

Requisitos importantes não ficam para o final, pois caso o projeto não seja concluído por motivos de tempo ou custo, o mais importante (prioritário) está pronto e entregue.



SCRUM

SCRUM PROCESS



SCRUM Working

Client, Stakeholders

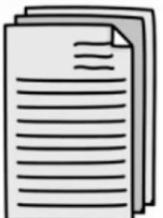
SCRUM Working



Client, Stakeholders



Product Owner



Product Backlog

SCRUM Working



Client, Stakeholders



Product Owner



Sprint Planning



Product Backlog

SCRUM Working



Client, Stakeholders

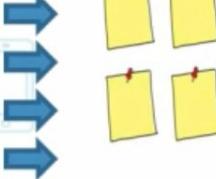


Product Owner

Sprint Planning



Product Backlog



Sprint Backlog

SCRUM Working

Client, Stakeholders

Product Owner

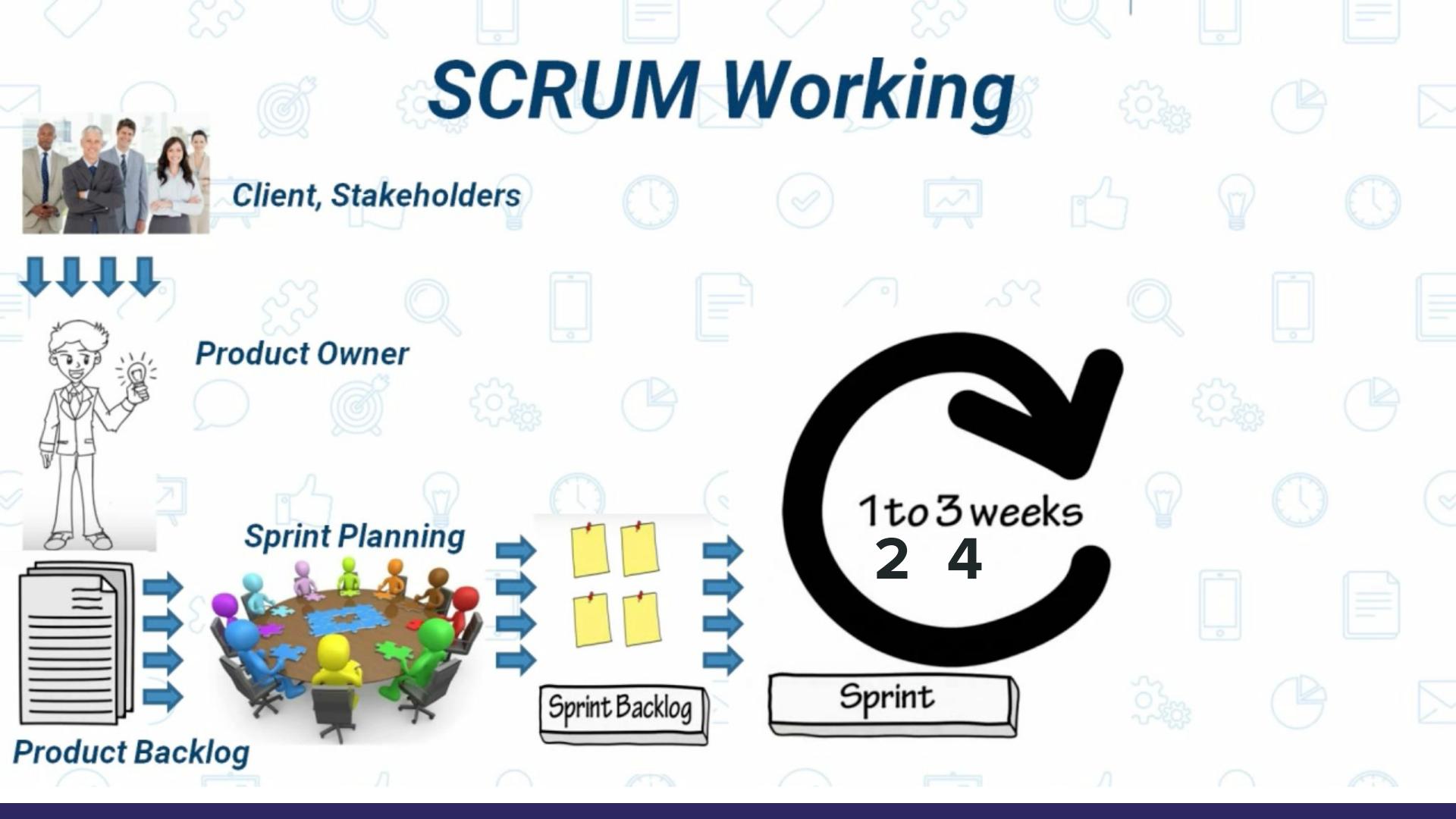
Sprint Planning

Sprint Backlog

1 to 3 weeks
2 4

Sprint

Product Backlog



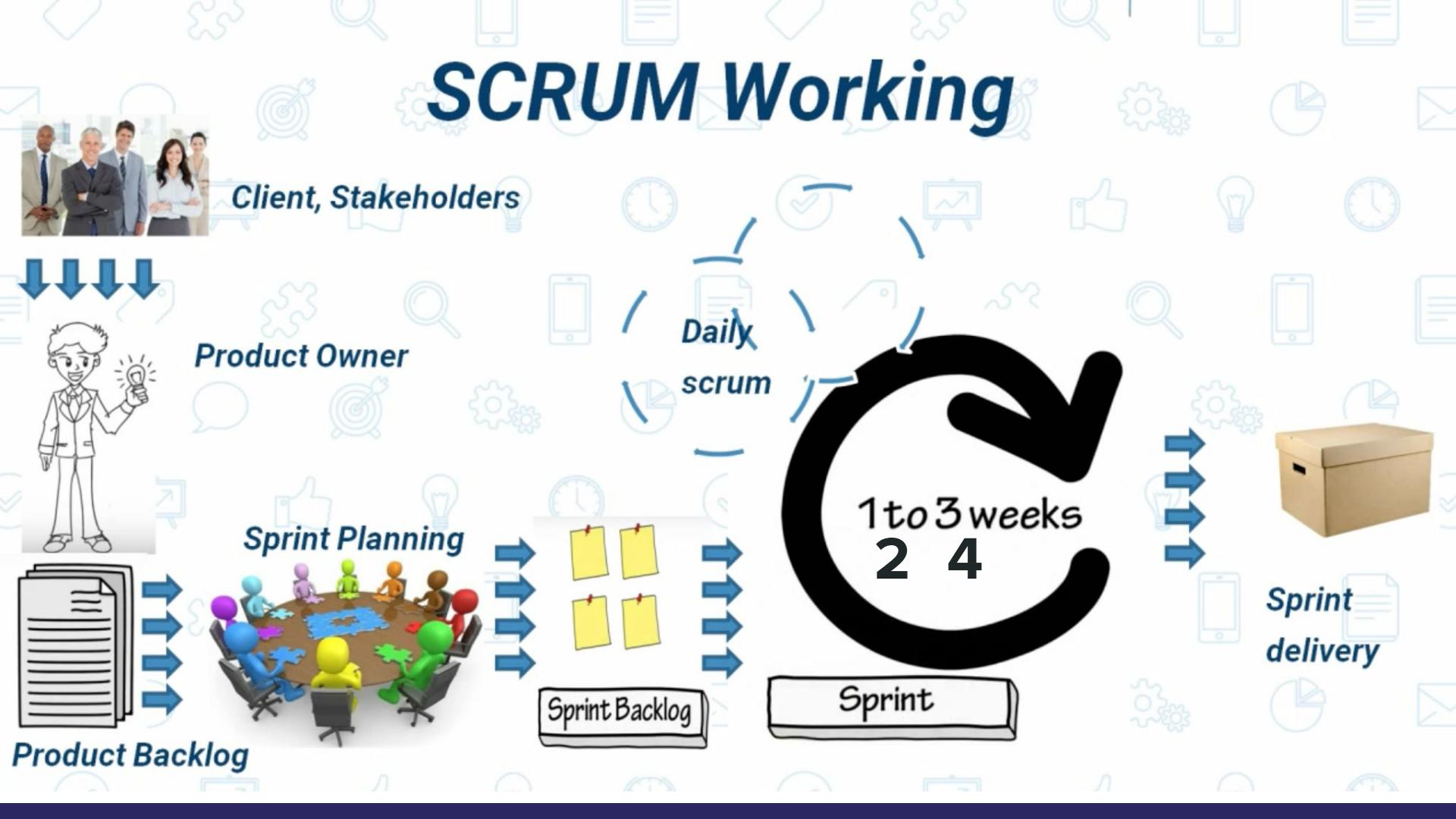
SCRUM Working



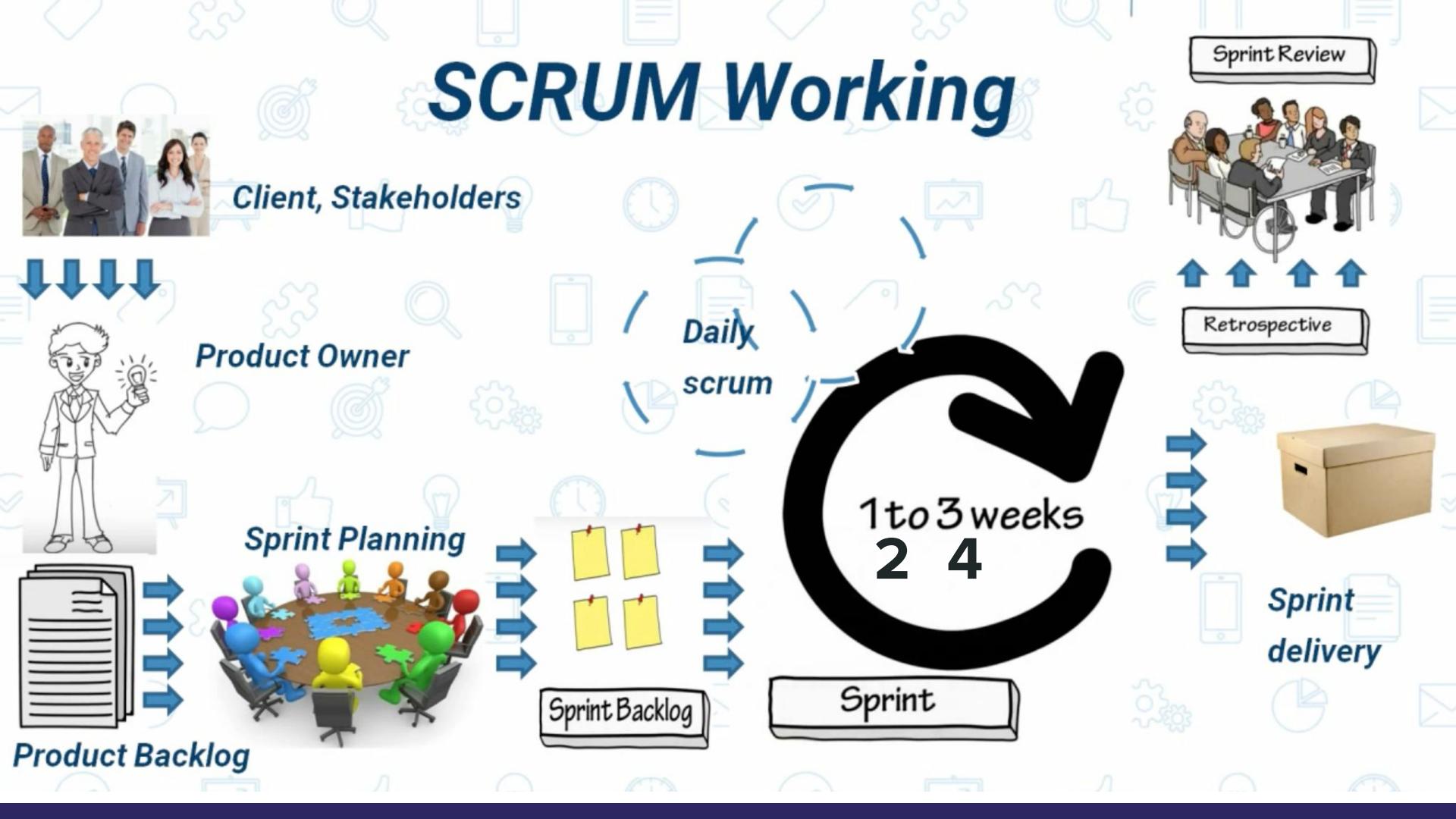
Scrum Meeting

- What did you do yesterday?
- What will you do today?
- Are there any impediments in your way?

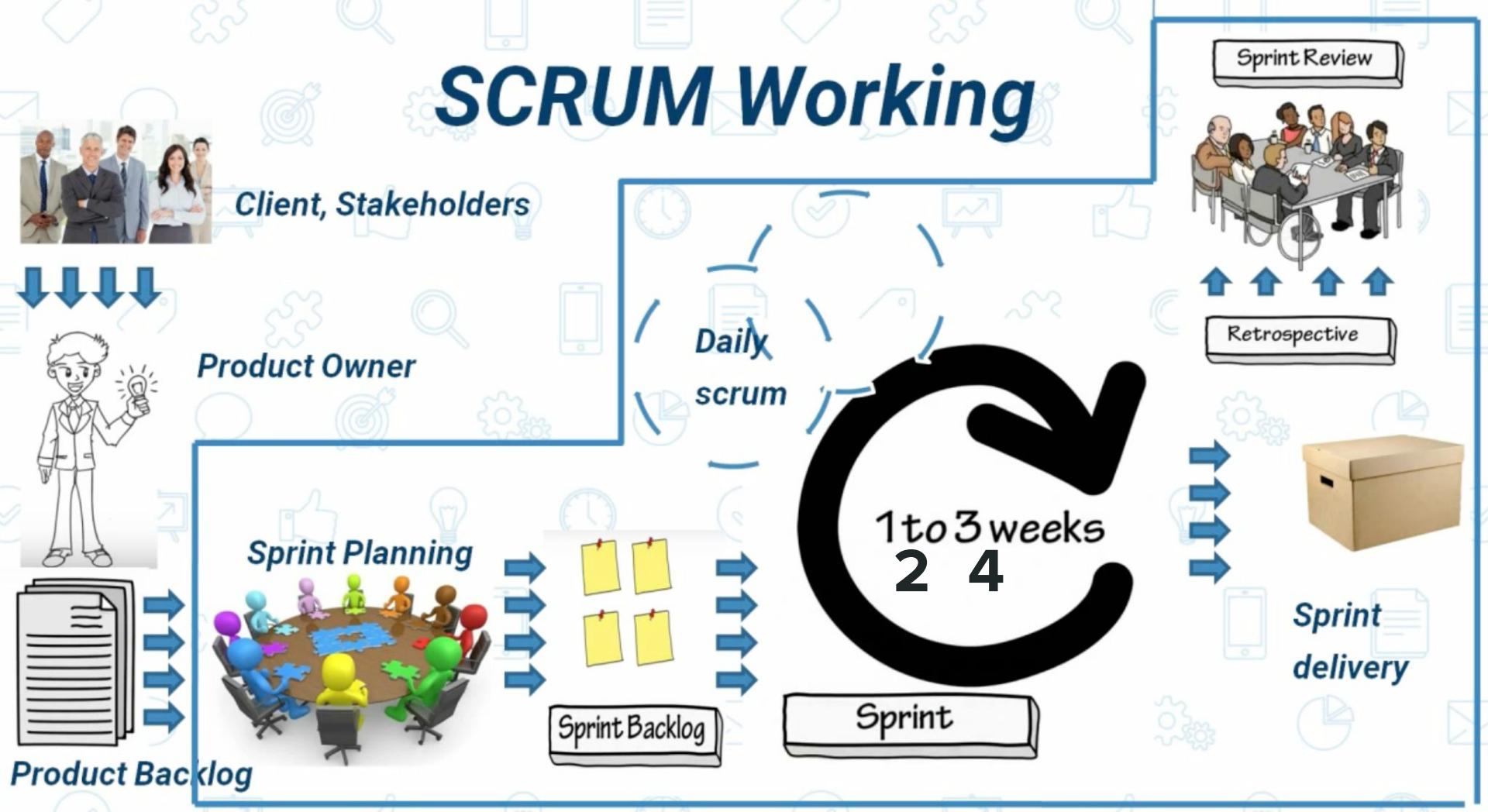
SCRUM Working



SCRUM Working

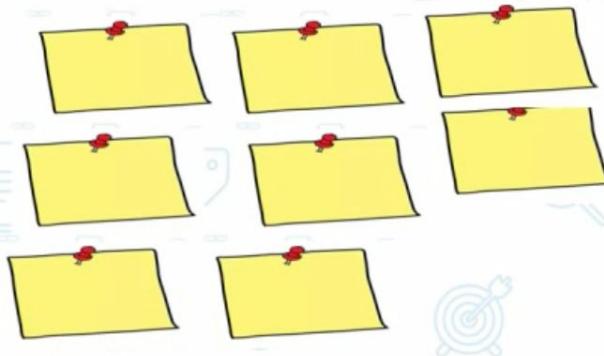


SCRUM Working

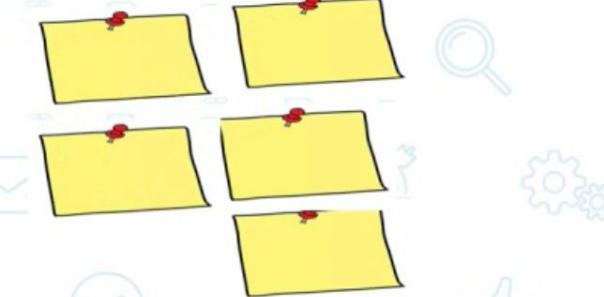


Scrum Board

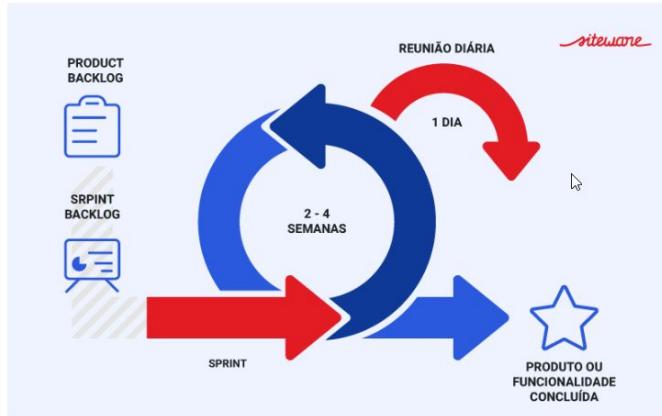
Product Backlog



Sprint Backlog



Open	In Progress	Testing	Closed
1	2	3	4



O Guia do Scrum

O Guia Definitivo para o Scrum: As Regras do Jogo

Os slides a seguir contém textos retirados - na íntegra - do Scrum Guide, documento cujo link é disponibilizado [aqui](#).

Novembro de 2020

Definição do Scrum

Scrum é um framework leve que ajuda pessoas, times e organizações a gerar valor por meio de soluções adaptativas para problemas complexos.

O framework Scrum é propositalmente incompleto, apenas definindo as partes necessárias para implementar a teoria Scrum. O Scrum é construído sobre a inteligência coletiva das pessoas que o utilizam. Em vez de fornecer às pessoas instruções detalhadas, as regras do Guia do Scrum orientam seus relacionamentos e interações. Vários processos, técnicas e métodos podem ser empregados com o framework, pois o Scrum se acopla às práticas existentes ou as torna desnecessárias.

Scrum torna visível a eficácia relativa da gestão atual, meio ambiente e técnicas de trabalho, para que melhorias possam ser feitas.

Teoria do Scrum

Scrum é baseado no empirismo e lean thinking. O empirismo afirma que o conhecimento vem da experiência e da tomada de decisões com base no que é observado. O lean thinking reduz o desperdício e se concentra no essencial.

Scrum emprega uma abordagem iterativa e incremental para otimizar a previsibilidade e controlar o risco. Scrum envolve grupos de pessoas que, coletivamente, possuem todas as habilidades e conhecimentos necessários para fazer o trabalho e compartilhar ou adquirir essas habilidades conforme necessário.

Scrum combina quatro eventos formais para inspeção e adaptação, contidos dentro de um evento, a Sprint. Esses eventos funcionam porque implementam os pilares empíricos do Scrum: transparência, inspeção e adaptação.

Teoria do Scrum

Transparência

O processo emergente e o trabalho devem ser visíveis tanto para quem executa o trabalho quanto para quem recebe o trabalho. Com o Scrum, decisões importantes são baseadas no estado percebido de seus três artefatos formais. Artefatos com baixa transparência podem levar a decisões que diminuem o valor e aumentam o risco. A transparência permite a inspeção. A inspeção sem transparência é enganosa e gera desperdício.

Inspeção

Os artefatos do Scrum e o progresso em direção às metas acordadas devem ser inspecionados com frequência e diligência para detectar variações ou problemas potencialmente indesejáveis. Para ajudar na inspeção, o Scrum fornece cadênciа na forma de seus cinco eventos. A inspeção habilita a adaptação. A inspeção sem adaptação é considerada inútil. Os eventos Scrum são projetados para provocar mudanças.

Adaptação

Se algum aspecto de um processo se desviar fora dos limites aceitáveis ou se o produto resultante for inaceitável, o processo que está sendo aplicado ou os materiais que estão sendo produzidos devem ser ajustados. O ajuste deve ser feito o mais rápido possível para minimizar novos desvios. A adaptação se torna mais difícil quando as pessoas envolvidas não são empoderadas ou auto-gerenciadas. Espera-se que um Scrum Team se adapte no momento em que aprende algo novo por meio da inspeção.

Os Valores do Scrum

O sucesso do uso do Scrum depende das pessoas se tornarem mais proficientes em viver cinco valores:

Compromisso, Foco, Abertura, Respeito e Coragem

O *Scrum Team* se **compromete** a atingir seus objetivos e suportar uns aos outros. Seu **foco** principal é o trabalho da Sprint para fazer o melhor progresso possível em direção a essas metas. O *Scrum Team* e seus stakeholders são **abertos** quanto ao trabalho e os desafios. Os membros do *Scrum Team* se **respeitam** quanto a serem pessoas capazes e independentes, e são respeitados como tal pelas pessoas com quem trabalham. Os membros do *Scrum Team* têm a **coragem** de, fazer a coisa certa e trabalhar em problemas difíceis.

Esses valores orientam o *Scrum Team* em relação ao seu trabalho, ações e comportamento. As decisões que são tomadas, os passos dados e a forma como o Scrum é usado devem reforçar esses valores, não diminuí-los ou miná-los. Os membros do *Scrum Team* aprendem e exploram os valores à medida que trabalham com os eventos e artefatos do Scrum. Quando esses valores são incorporados pelo *Scrum Team* e pelas pessoas com quem trabalham, os **pilares** empíricos do Scrum de **transparência, inspeção e adaptação** ganham vida, construindo confiança.



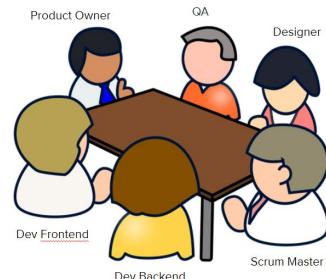
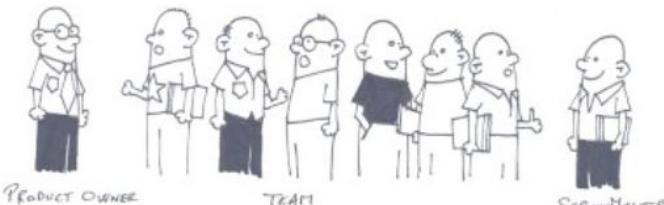
Scrum Team e seus Papéis

A unidade fundamental do Scrum é um pequeno time de pessoas, um Scrum Team. O Scrum Team consiste em um Scrum Master, um Product owner e Developers. Dentro de um Scrum Team, não há sub-times ou hierarquias. É uma unidade coesa de profissionais focados em um objetivo de cada vez, a Meta do Produto.

Os Scrum Teams são multifuncionais, o que significa que os membros possuem todas as habilidades necessárias para criar valor a cada Sprint. Eles também são autogerenciáveis, o que significa que decidem internamente quem faz o quê, quando e como. O Scrum Team é pequeno o suficiente para permanecer ágil e grande o suficiente para concluir um trabalho significativo dentro de uma Sprint, normalmente 10 ou menos pessoas. Em geral, descobrimos que times menores se comunicam melhor e são mais produtivos. Se os Scrum Teams se tornarem muito grandes, eles devem considerar a reorganização em vários Scrum Teams coesos, cada um focado no mesmo produto. Portanto, eles devem compartilhar a mesma meta do produto, Product Backlog e Product Owner.

O Scrum Team é responsável por todas as atividades relacionadas ao produto, desde a colaboração com stakeholder, verificação, manutenção, operação, experimentação, pesquisa e desenvolvimento, e qualquer outra coisa que possa ser necessária. Eles são estruturados e empoderados pela organização para gerenciar seu próprio trabalho. Trabalhar em Sprints em um ritmo sustentável melhora o foco e a consistência do Scrum Team.

Todo o Scrum Team é responsável por criar um Incremento valioso e útil a cada Sprint. Scrum define três responsabilidades específicas dentro do Scrum Team: os **Developers**, o **Product Owner** e o **Scrum Master**.



Papéis do Scrum Team

Product Owner (cliente ou seu representante)

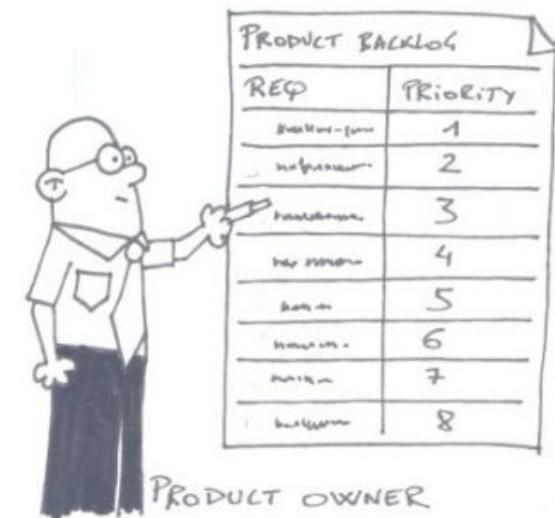
O Product Owner é responsável por maximizar o valor do produto resultante do trabalho do Scrum Team. A forma como isso é feito pode variar amplamente entre organizações, Scrum Teams e indivíduos. O Product Owner também é responsável pelo gerenciamento eficaz do Product Backlog , que inclui:

- Desenvolver e comunicar explicitamente a meta do produto;
- Criar e comunicar claramente os itens do Product Backlog;
- Ordenar os itens do Product Backlog; e,
- Garantir que o Product Backlog seja transparente, visível e compreensível.

O Product Owner pode fazer o trabalho acima ou pode delegar a responsabilidade para outros. Independentemente disso, o Product Owner ainda é o responsável.

Para que os Product Owners tenham sucesso, toda a organização deve respeitar suas decisões. Essas decisões são visíveis no conteúdo e na ordem do Product Backlog e por meio do incremento inspecionável na revisão da sprint.

O Product Owner é uma pessoa, não um comitê. O Product Owner pode representar as necessidades de muitos stakeholders no Product Backlog. Aqueles que desejam alterar o Product Backlog podem fazê-lo tentando convencer o Product Owner.



Papéis do Scrum Team

Scrum Master

O Scrum Master é responsável por estabelecer o Scrum conforme definido no Guia do Scrum. Eles fazem isso ajudando todos a entender a teoria e a prática do Scrum, tanto no Scrum Team quanto na organização. O Scrum Master é responsável pela eficácia do Scrum Team. Eles fazem isso permitindo que o Scrum Team melhore suas práticas, dentro do framework Scrum. Scrum Masters são verdadeiros líderes que servem ao Scrum Team e à organização como um todo.

O Scrum Master serve ao Scrum Team de várias maneiras, incluindo:

- Treinar os membros do time em autogerenciamento e cross-funcionalidade;
- Ajudar o Scrum Team a se concentrar na criação de incrementos de alto valor que atendem à Definição de Pronto;
- Provocando a remoção de impedimentos ao progresso do Scrum Team; e
- Garantir que todos os eventos Scrum ocorram e sejam positivos, produtivos e mantidos dentro do Timebox.

O Scrum Master serve o Product Owner de várias maneiras, incluindo:

- Ajudar a encontrar técnicas para a definição eficaz de meta do Produto e gerenciamento do Product Backlog;
- Ajudar o Scrum Team a entender a necessidade de itens do Product Backlog claros e concisos;
- Ajudar a estabelecer o planejamento empírico do produto para um ambiente complexo; e,
- Facilitar a colaboração dos stakeholder, conforme solicitado ou necessário.

O Scrum Master serve a organização de várias maneiras, incluindo:

- Liderar, treinar e orientar a organização na adoção do Scrum;
- Planejar e aconselhar implementações de Scrum dentro da organização;
- Ajudar os funcionários e os stakeholders a compreender e aplicar uma abordagem empírica para trabalhos complexos; e,
- Remover barreiras entre stakeholders e Scrum Teams.



Papéis do Scrum Team

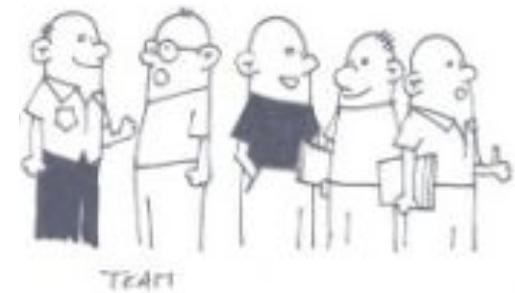
Dev Team

Developers são as pessoas do Scrum Team que estão comprometidas em criar qualquer aspecto de um Incremento utilizável a cada Sprint. As habilidades específicas necessárias pelos Developers geralmente são amplas e variam de acordo com o domínio de trabalho. No entanto, os Developers são sempre responsáveis por:

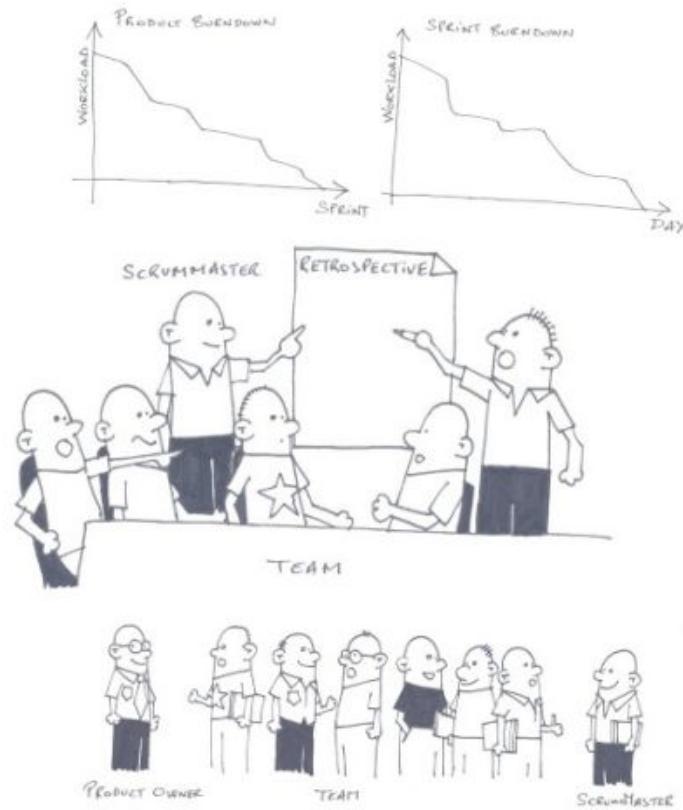
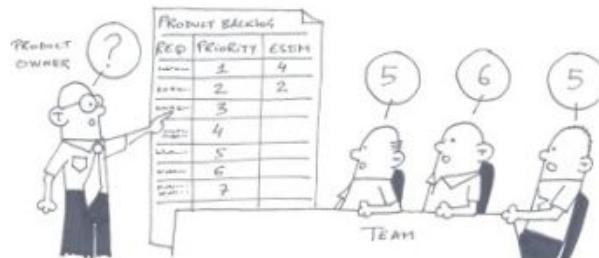
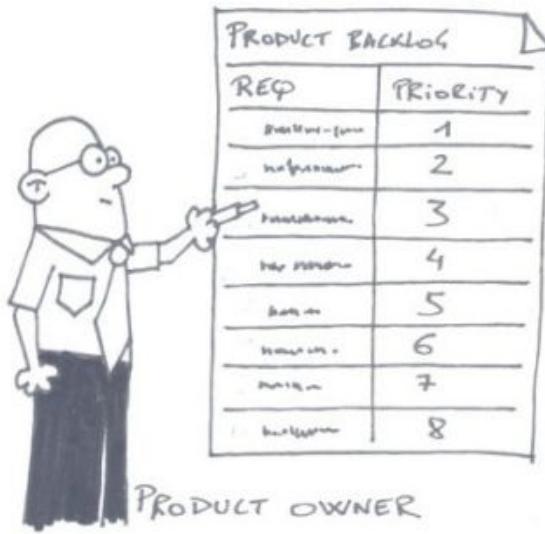
- Criar um plano para a Sprint, o Sprint Backlog;
- Introduzir gradualmente qualidade aderindo a uma Definição de Pronto; Adaptar seu plano a cada dia em direção à meta da Sprint; e,
- Responsabilizar-se mutuamente como profissionais
- Desenvolver as tarefas do Sprint Backlog

O Dev team deve ser auto-gerenciável e auto-organizável (demanda experiência e maturidade) e, normalmente, são times multidisciplinares formados por:

- Desenvolvedor(es) Backend
- Desenvolvedor(es) Frontend
- Analista(s) de Qualidade
- Designer(s)
- Arquiteto(s)
- Technical Writer(s)
- Etc



Scrum Team

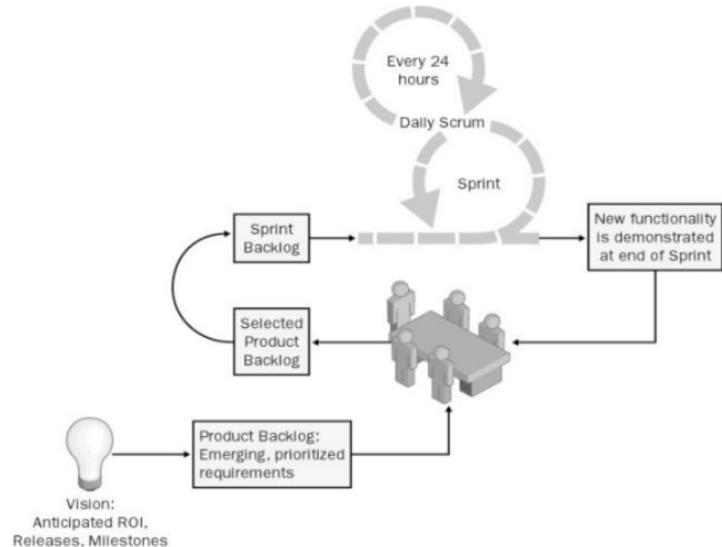


Eventos Scrum

→ Sprint

A Sprint é um contêiner para todos os outros eventos. Cada evento no Scrum é uma oportunidade formal para inspecionar e adaptar os artefatos do Scrum. Esses eventos são projetados especificamente para permitir a transparência necessária. A falha em operar quaisquer eventos conforme prescrito resulta em oportunidades perdidas de inspeção e adaptação. Os eventos são usados no Scrum para criar regularidade e minimizar a necessidade de reuniões não definidas no Scrum. O ideal é que todos os eventos sejam realizados no mesmo horário e local para reduzir a complexidade.

- Sprint Planning Meeting
- Daily Scrum Meeting
- Sprint Review Meeting
- Sprint Retrospective Meeting



Ritos/Cerimônias do Scrum

Sprints são o coração do Scrum, onde ideias são transformadas em valor. São eventos de duração fixa de um mês ou menos para criar consistência. Uma nova Sprint começa imediatamente após a conclusão da Sprint anterior.

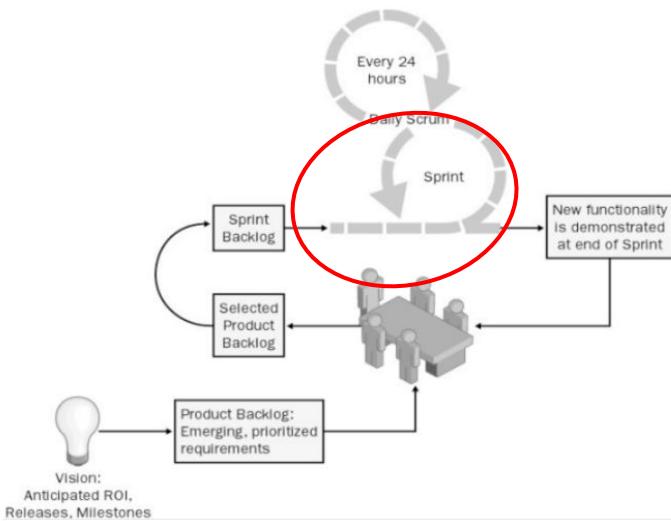
Todo o trabalho necessário para atingir a meta do Produto, incluindo Sprint Planning, Daily Scrums, Sprint Review e Sprint Retrospective, acontece dentro de Sprints. Durante a Sprint:

- Nenhuma mudança é feita que coloque em risco a meta da Sprint;
- A qualidade não diminui;
- O Product Backlog é refinado conforme necessário; e,
- O escopo pode ser esclarecido e renegociado com o Product Owner conforme mais é aprendido.

Sprints permitem previsibilidade, garantindo a inspeção e adaptação do progresso em direção a uma meta do Produto ao menos uma vez por mês. Quando o horizonte de uma Sprint é muito longo, a meta da Sprint pode se tornar inválida, a complexidade pode aumentar e o risco pode aumentar. Sprints mais curtas podem ser empregados para gerar mais ciclos de aprendizagem e limitar os riscos de custo e esforço a um período de tempo menor. Cada Sprint pode ser considerado um projeto curto.

Existem várias práticas para prever o progresso, como burn-downs, burn-ups ou cumulative flows. Embora comprovadamente úteis, eles não substituem a importância do empirismo. Em ambientes complexos, o que acontecerá é desconhecido. Somente o que já aconteceu pode ser usado para a tomada de decisão voltada para o futuro.

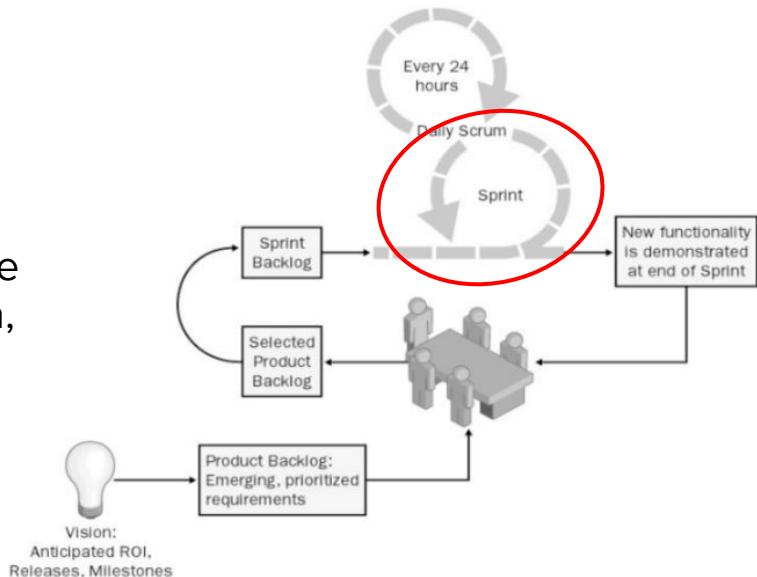
Uma Sprint pode ser cancelada se a Meta da Sprint se tornar obsoleta. Apenas o Product Owner tem autoridade para cancelar a Sprint.



Ritos/Cerimônias do Scrum

Sprint: de 2 a 4 semanas de duração

- Ciclo de desenvolvimento que pode ter 2, 3 ou 4 semanas de duração. Uma vez definido o ciclo, este deve ser mantido em todas as Sprints;
- As tarefas do Sprint Backlog são realizadas durante esse ciclo de desenvolvimento;
- As tarefas do Sprint Backlog são **congeladas** durante a execução da Sprint. Ou seja, quando a Sprint inicia, não há inclusão ou exclusão de itens no Sprint Backlog;
- O Scrum Master pode abortar o Sprint (em casos extremos).



Ritos/Cerimônias do Scrum

Sprint Planning Meeting:

té 4h de duração em Sprints de 2 semanas.

A Sprint Planning inicia a Sprint ao definir o trabalho a ser realizado na Sprint. Este plano resultante é criado pelo trabalho colaborativo de todo o Scrum Team.

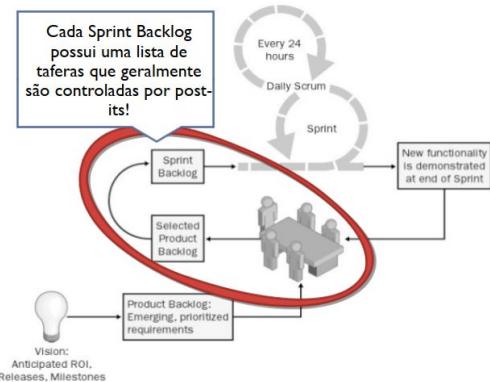
O Product Owner garante que os participantes estejam preparados para discutir os itens mais importantes do Product Backlog e como eles são mapeados para a Meta do Produto. O Scrum Team também pode convidar outras pessoas para participar da Sprint Planning para fornecer conselhos.

A Sprint Planning aborda os seguintes tópicos:

- Tópico um: Por que esta Sprint é valiosa? O Product Owner propõe como o produto pode aumentar seu valor e utilidade na Sprint atual. Todo o Scrum Team então colabora para definir uma Meta da Sprint que comunica porque a Sprint é valiosa para os stakeholders. A meta da Sprint deve ser finalizada antes do final da Sprint Planning.
- Tópico dois: O que pode ser feito nesta Sprint? Por meio de discussão com o Product Owner, os Developers selecionam itens do Product Backlog para incluir na Sprint atual. O Scrum Team pode refinar esses itens durante este processo, o que aumenta a compreensão e a confiança. Selecionar o quanto pode ser concluído em uma Sprint pode ser um desafio. No entanto, quanto mais os Developers sabem sobre seu desempenho anterior, sua capacidade futura e sua Definição de Pronto, mais confiantes eles estarão em suas previsões quanto a Sprint.
- Tópico três: Como o trabalho escolhido será realizado? Para cada item do Product Backlog selecionado, os Developers planejam o trabalho necessário para criar um Incremento que atenda à Definição de Pronto. Isso geralmente é feito decompondo itens do Product Backlog em itens de trabalho menores de um dia ou menos. A forma como isso é feito fica a critério exclusivo dos Developers. Ninguém mais diz a eles como transformar itens do Product Backlog em incrementos de valor.

A Meta da Sprint, os itens do Product Backlog selecionados para a Sprint, mais o plano para entregá-los são chamados juntos de Sprint Backlog.

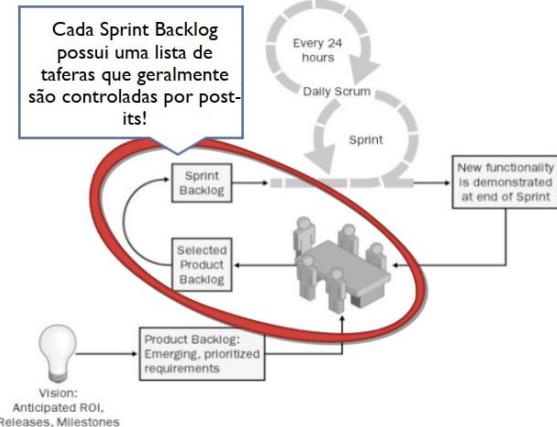
A Sprint Planning tem um Timebox definido com duração máxima de oito horas para uma Sprint de um mês. Para Sprints mais curtas, o evento geralmente é mais curto.



Ritos/Cerimônias do Scrum

Sprint Planning Meeting: até 4h de duração em Sprints de 2 semanas.

- Define itens mais importantes do Product Backlog para serem implementados durante a sprint que se inicia;
- Resultará no incremento da sprint;
- O Product Backlog deve ser previamente preparado e priorizado pelo Product Owner;
- Todos os papéis (stakeholders) participam;
- Product Owner e/ou Scrum Master respondem as perguntas da equipe para detalhamento das tarefas;
- Ao final, tem-se o Sprint Backlog (tarefas dos requisitos que deverão ser desenvolvidas).



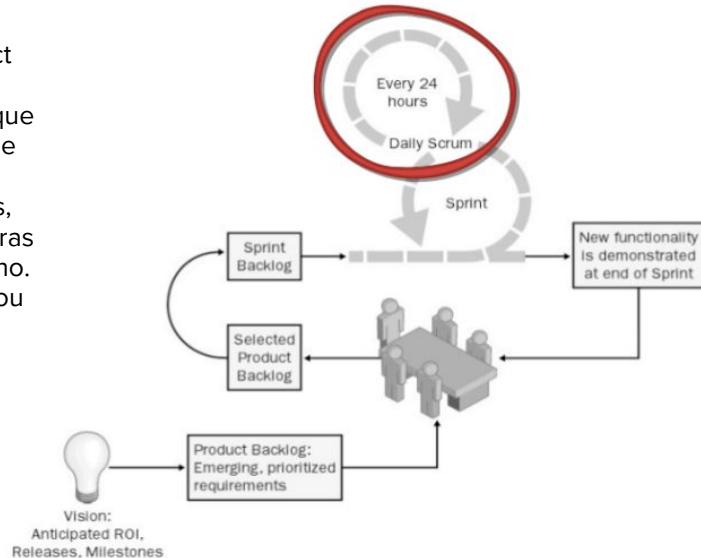
Ritos/Cerimônias do Scrum

Daily Scrum Meeting: até 15 minutos, independente da duração da Sprint.

O propósito da Daily Scrum é inspecionar o progresso em direção a Meta da Sprint e adaptar o Sprint Backlog conforme necessário, ajustando o próximo trabalho planejado.

A Daily Scrum é um evento de 15 minutos para os Developers do Scrum Team. Para reduzir a complexidade, é realizado no mesmo horário e local, todos os dias úteis da Sprint. Se o Product Owner ou o Scrum Master estão trabalhando ativamente nos itens do Sprint Backlog, eles participam como Developers. Os Developers podem selecionar qualquer estrutura e técnicas que quiserem, desde que seu Daily Scrum se concentre no progresso em direção a Meta da Sprint e produza um plano de ação para o próximo dia de trabalho. Isso cria foco e melhora o autogerenciamento. As Daily Scrums melhoram as comunicações, identificam os impedimentos, promovem a rápida tomada de decisões e consequentemente, eliminam a necessidade de outras reuniões. A Daily Scrum não é o único momento em que os Developers podem ajustar seu plano. Eles costumam se reunir ao longo do dia para discussões mais detalhadas sobre a adaptação ou replanejamento do resto do trabalho da Sprint.

- Reunião diária realizada em pé sempre no mesmo horário e local;
 - Muita rigidez com presença e pontualidade
 - Cada membro do time responde 3 perguntas:
 - ◆ O que você fez desde a última daily meeting?
 - ◆ O que você vai fazer de agora até a próxima daily?
 - ◆ O que lhe impede de fazer o seu trabalho da forma mais eficiente possível?



Ritos/Cerimônias do Scrum

Sprint Review Meeting: até 4h de duração em Sprints de 2 semanas.

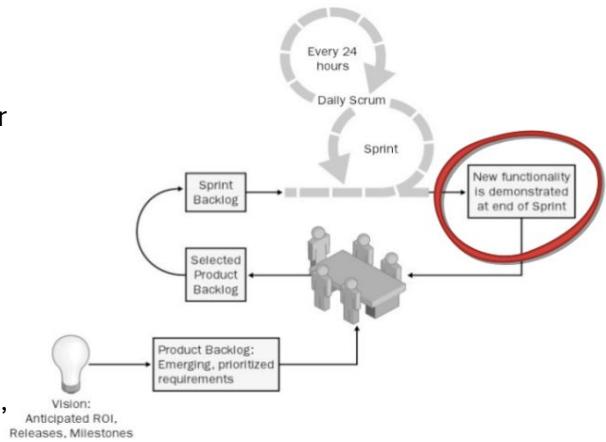
O propósito da Sprint Review é inspecionar o resultado da Sprint e determinar as adaptações futuras. O Scrum Team apresenta os resultados de seu trabalho para os principais stakeholders e o progresso em direção a Meta do Produto é discutido.

Durante o evento, o Scrum Team e os stakeholders revisam o que foi realizado na Sprint e o que mudou em seu ambiente. Com base nessas informações, os participantes colaboram sobre o que fazer a seguir. O Product Backlog também pode ser ajustado para atender a novas oportunidades. A Sprint Review é uma sessão de trabalho e o Scrum Team deve evitar limitá-la a uma apresentação.

A Sprint Review é o penúltimo evento da Sprint e tem um Timebox com prazo máximo de quatro horas para uma Sprint de um mês. Para Sprints mais curtas, o evento geralmente é mais curto.

Highlights:

- Artefatos (PB, SB, Burndown) não pode ser apresentados como produtos de trabalho, exceto o incremento; isso seria uma forma de policiar o contrato. Porém, o que tem valor é o software funcional para o cliente - valor ágil
- Stakeholders são ouvidos, dão feedback e solicitação de mudanças pode acontecer;
- Ao final, a próxima Sprint Review Meeting é agendada



Ritos/Cerimônias do Scrum

Sprint Retrospective Meeting: até 3h de duração em sprints de 1 mês. Até 1h em sprints de 2 semanas.

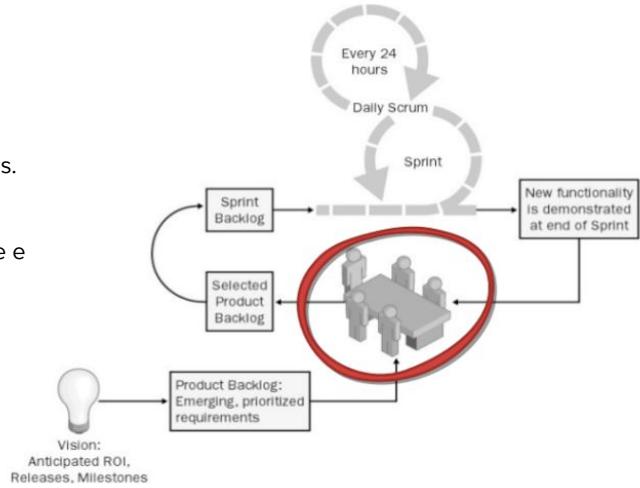
O propósito da Sprint Retrospective é planejar maneiras de aumentar a qualidade e a eficácia. O Scrum Team inspeciona como foi a última Sprint em relação a indivíduos, interações, processos, ferramentas e sua Definição de Pronto. Os elementos inspecionados geralmente variam com o domínio de trabalho. As suposições que os desviaram são identificadas e suas origens exploradas. O Scrum Team discute o que deu certo durante a Sprint, quais problemas encontraram e como esses problemas foram (ou não) resolvidos.

O Scrum Team identifica as mudanças mais úteis para melhorar sua eficácia. As melhorias mais impactantes são endereçadas o mais rápido possível. Essas podem até ser adicionadas ao Sprint Backlog para a próxima Sprint.

A Sprint Retrospective **conclui** a Sprint. É limitada pelo Timebox de no máximo três horas para uma Sprint de um mês. Para Sprints mais curtas, o evento geralmente é mais curto.

Participam o Scrum Master, Dev Team e Product Owner (opcional). Cliente não participa, pois é um momento do time e para o time focar em melhoria contínua. Participações externas podem inibir a transparência do time;

- Perguntas que conduzem a reunião:
 - ◆ O que foi bom no último Sprint?
 - ◆ O que não foi bom ou podemos melhorar?
 - ◆ Plano de ação para as melhorias priorizadas
- Scrum Master cataloga as respostas e facilita a reunião estimulando a participação e transparência de todos. Responsável por criar um ambiente aberto e seguro;
- Dev Team prioriza as melhorias para discussão e ação.



Artefatos do Scrum

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

Cada artefato contém um compromisso para garantir que ele forneça informações que aumentem a transparência e o foco contra o qual o progresso pode ser medido:

- Para o **Product Backlog**, é a Meta do produto;
- Para o **Sprint Backlog**, é a Meta da Sprint;
- Para o **incremento**, é a Definição de Pronto.

Esses compromissos existem para reforçar o empirismo e os valores Scrum para o Scrum Team, e seus stakeholders.

- Product Backlog: lista de requisitos continuamente revisada, atualizada e priorizada.
- Sprint Backlog: tarefas do requisito. É derivado do product backlog e detalha os itens em tarefas para o time desenvolver.
- Burndown Chart: gráfico onde visualiza-se o planejado e executado.
- Incremento de funcionalidade: cada sprint resulta em um incremento, ou seja, um acréscimo de itens de valor. Em um projeto de desenvolvimento de software, por exemplo, considera-se um incremento concluído (pronto): testado, código bem escrito e bem estruturado, disponível em um executável, e com documentação de usuário.

Artefatos do Scrum

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.

Os itens do Product Backlog que podem ser realizados pelo Scrum Team em uma Sprint são considerados preparados para seleção no evento Sprint Planning. Eles geralmente adquirem esse grau de transparência após as atividades de refinamento. O Product Backlog refinement é o ato de quebrar e incluir definição adicional aos itens do Product Backlog para ter itens menores e mais precisos. Esta é uma atividade contínua para adicionar detalhes, como descrição, ordem e tamanho. Os atributos geralmente variam de acordo com o domínio de trabalho.

Os Developers que farão o trabalho são responsáveis pelo dimensionamento. O Product Owner pode influenciar os Developers, ajudando-os a entender e selecionar trade-offs (trocas de itens).

Artefatos do Scrum

Compromisso: Meta do Produto

A Meta do Produto descreve um estado futuro do produto que pode servir como um alvo para o Scrum Team planejar. A Meta do produto está no Product Backlog. O restante do Product Backlog emerge para definir “o que” cumprirá a Meta do Produto.

Um produto é um veículo para entregar valor. Tem um limite claro, stakeholders conhecidos, usuários ou clientes bem definidos. Um produto pode ser um serviço, um produto físico ou algo mais abstrato.

A Meta do Produto é o objetivo de longo prazo para o Scrum Team. Eles devem cumprir (ou abandonar) um objetivo antes de assumir o próximo.

Artefatos do Scrum

Sprint Backlog

O Sprint Backlog é composto pela Meta da Sprint (por que), o conjunto de itens do Product Backlog selecionados para a Sprint (o que), bem como um plano de ação para entregar o Incremento (como).

O Sprint Backlog é um plano feito por e para os Developers. É uma imagem altamente visível, em tempo real do trabalho que os Developers planejam realizar durante a Sprint para atingir a Meta da Sprint. Consequentemente, o Sprint Backlog é atualizado ao longo da Sprint conforme mais é aprendido. Deve ter detalhes suficientes para que eles possam inspecionar seu progresso na Daily Scrum.

Artefatos do Scrum

Incremento

Um incremento é um trampolim concreto em direção a Meta do produto. Cada incremento é adicionado a todos os incrementos anteriores e completamente verificado, garantindo que todos os incrementos funcionem juntos. A fim de fornecer valor, o incremento deve ser utilizável.

Vários incrementos podem ser criados em uma Sprint. A soma dos incrementos é apresentada na Sprint Review, apoiando assim o empirismo. No entanto, um incremento pode ser entregue aos stakeholders antes do final da Sprint. A Sprint Review nunca deve ser considerada um marco para liberar valor.

O trabalho não pode ser considerado parte de um incremento a menos que atenda a Definição de Pronto (**Definition of Done - DoD**).

Artefatos do Scrum

Compromisso: Definição de Pronto (Definition of Done - DoD)

A Definição de Pronto é uma descrição formal do estado do Incremento quando ela atende às medidas de qualidade exigidas para o produto.

No momento em que um item do Product Backlog atende a Definição de Pronto, um incremento nasce. A Definição de Pronto cria transparência ao fornecer a todos um entendimento compartilhado de qual trabalho foi concluído como parte do Incremento. Se um item do Product Backlog não atender à Definição de Pronto, ele não poderá ser liberado ou mesmo apresentado na Sprint Review. Em vez disso, ele retorna ao Product Backlog para consideração futura.

Se a Definição de Pronto para um incremento faz parte dos padrões da organização, todos os Scrum Teams devem segui-la como mínimo. Se não for um padrão organizacional, o Scrum Team deve criar uma Definição de Pronto apropriada para o produto.

Os Developers devem estar em conformidade com a Definição de Pronto. Se houver vários Scrum Teams trabalhando juntos em um produto, eles devem definir e cumprir mutuamente a mesma Definição de Pronto.

Nota Final

Compromisso: Definição de Pronto (Definition of Done - DoD)

Scrum é gratuito e oferecido neste Guia. O framework Scrum, conforme descrito aqui, é imutável. Embora a implementação de apenas partes do Scrum seja possível, o resultado não é Scrum. Scrum existe apenas em sua totalidade e funciona bem como um contêiner para outras técnicas, metodologias e práticas.

Reconhecimentos

Pessoas

Das milhares de pessoas que contribuíram para o *Scrum*, devemos destacar aquelas que foram fundamentais no início: **Jeff Sutherland** trabalhou com **Jeff McKenna** e **John Scumniotales**, e **Ken Schwaber** trabalhou com **Mike Smith** e **Chris Martin**, e todos eles trabalharam juntos. Muitos outros contribuíram nos anos seguintes e sem sua ajuda o Scrum não seria refinado como é hoje.

A História do Guia Scrum

Ken Schwaber e Jeff Sutherland primeiramente co-apresentaram o Scrum na Conferência OOPSLA em 1995. Essencialmente, documentou o aprendizado que Ken e Jeff obtiveram nos poucos anos anteriores e tornou pública a primeira definição formal de Scrum.

O Guia do Scrum documenta o Scrum conforme desenvolvido, evoluído e sustentado por mais de 30 anos por Jeff Sutherland e Ken Schwaber. Outras fontes fornecem padrões, processos e percepções que complementam o framework Scrum. Estes podem aumentar a produtividade, valor, criatividade e satisfação com os resultados.

A história completa do Scrum é descrita em outros lugares. Para homenagear os primeiros lugares onde foi testado e comprovado, reconhecemos Individual Inc., Newspage, Fidelity Investments e IDX (agora GE Medical).

Tradução

Este guia foi traduzido da versão original em inglês, fornecida pelas pessoas reconhecidas acima. Os colaboradores desta tradução incluem **Fábio Cruz**, **Eduardo Rodrigues Sucena** e **Rodrigo Paulo Camargo**.

DEFINITION OF DONE

(Definição de Pronto)

A Definition of Done é um dos 3 Compromissos do Scrum



Como compromisso do Incremento, temos a DoD



Para o Product Backlog, temos o Product Goal



Para o Sprint Backlog, temos o Sprint Goal

A DoD cria transparéncia ao fornecer as etapas de qualidade para produzir um incremento de produto



Um Item do Product Backlog não pode ser considerado parte do incremento até que atinja a DoD

O DoD auxilia:

- A criar o mesmo entendimento entre todos, sobre o que é Done
- A definir um padrão de qualidade
- No alinhamento das expectativas dos stakeholders
- No Planejamento da Sprint

Se um Item do Product Backlog não atender a DoD, ele não pode ser levado para Sprint Review e não pode ser lançado.



Itens que não atinjam a DoD retornam para o Product Backlog

Uma DoD que não inclui todas as etapas para ter um Incremento Potencialmente Lançável, é considerada uma DoD ainda fraca e ineficaz



Exemplo de uma DoD para um time de Software:

- Revisão de outro Desenvolvedor (Code review)
- Versionamento correto do código
- Teste de regressão
- Seguindo padrão de estilo
- Acessibilidade
- Atende as políticas de segurança
- Documentações atualizadas

A DoD deve ser criada pelo Scrum Team, caso a organização não tenha esse padrão definido



Se múltiplos Scrum Teams estão trabalhando no mesmo Produto, eles devem usar a mesma DoD



O Scrum Team pode adaptar a Definition of Done na Sprint Retrospective, para aumentar a qualidade.

Alinhando todos sobre um entendimento comum do que é Done, reduzimos riscos, retrabalhos e atrasos.



Dicas valiosas



Deixe a DoD o mais visível possível, por exemplo: colando na parede do escritório, ou em algum lugar digital que todos do time possam ver e inspecionar)

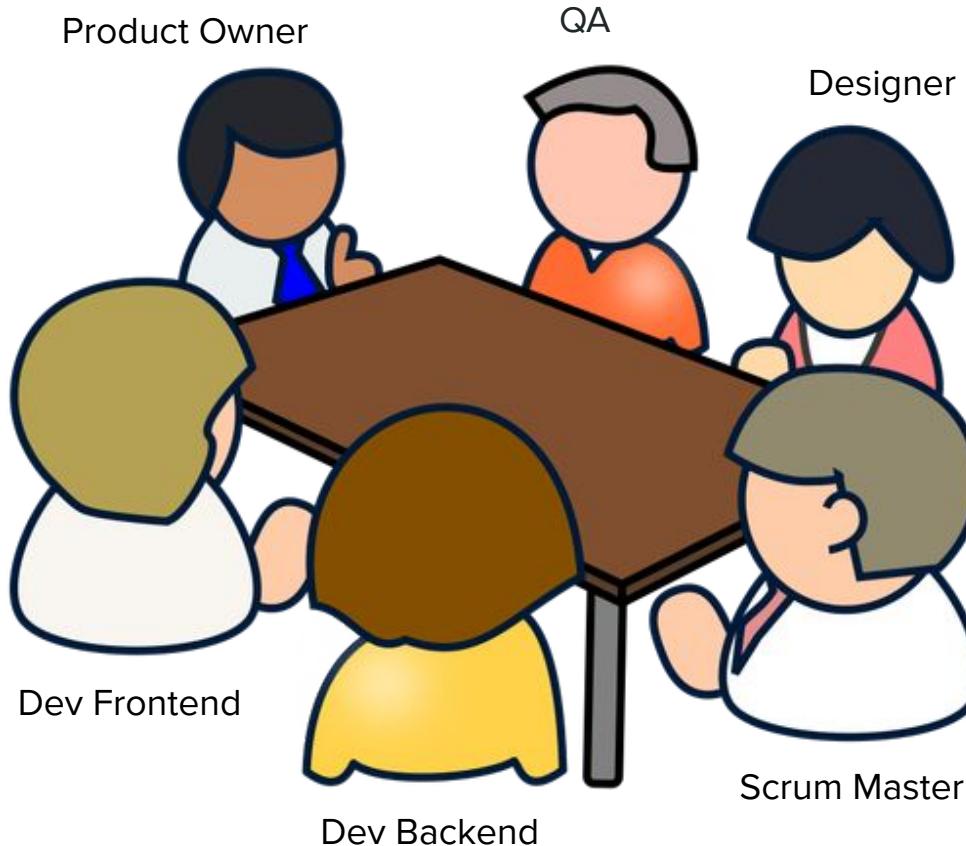


O Incremento tem que estar sempre em ponto de lançamento, atendendo a DoD, mas isso não significa que devemos sempre lançar o Incremento, isso vai depender da estratégia do Product Owner

Como criar a DoD do zero

1. Faça um brainstorm com o Scrum Team
2. Em conjunto, Identifiquem todas as ações necessárias para que o item seja considerado **DONE**
3. Separe as ações que o time realiza (**AS IS**), das que o time ainda não consegue realizar (**TO BE**), independente do motivo
4. Criei um DoD com base nas ações já realizadas
5. Continue aperfeiçoando e incluindo mais ações, até que o Definition of Done, tenha a cobertura necessária para garantir que após concluído o item possa ser lançado de acordo com a visão do Product Owner

Time Scrum



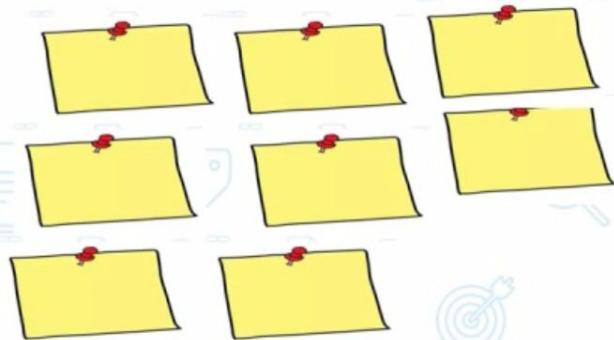
KANBAN



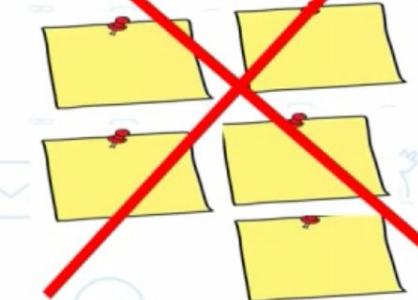
- **No fixed timelines**
- **No fixed planning and meeting**
- **No Task Estimates**
- **No Scrum Master**

Kanban Board

Product Backlog



Sprint Backlog



To Do	Doing	Done
1	2	3
4	5	6
7	8	9
10	11	12

Limit 4



16th State of Agile 2022

16th State of Agile Report - 2022



UNIVERSIDADE
VILA VELHA
ESPIRITO SANTO



Este relatório avalia o Estado da Agilidade sob a ótica de 3 componentes chave:

- People
- Processes
- Tools

Infos da 16^a pesquisa anual:

- Julho 2022 a agosto 2022
- 3.200 respondentes

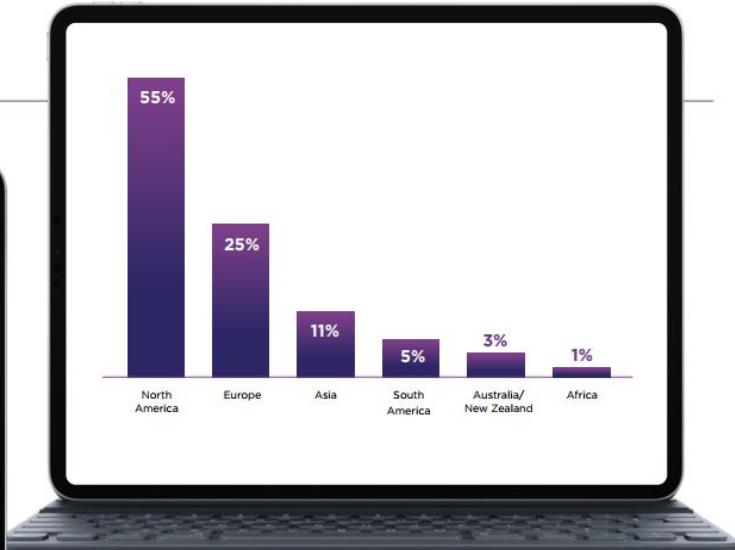
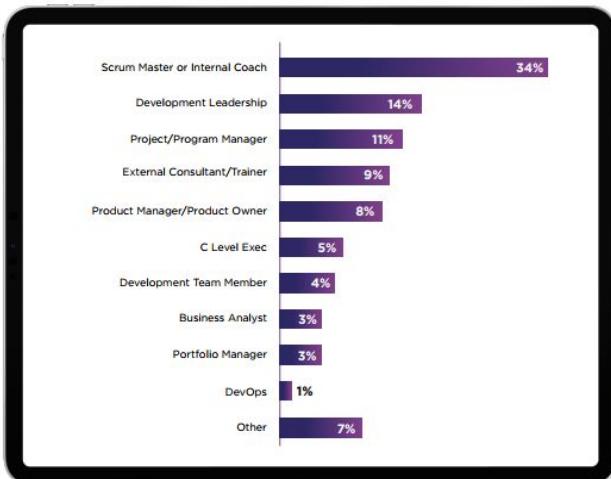
17^a pesquisa está em andamento de julho 2023 a agosto 2023.

Demographics

One-third of respondents come from companies with over 20,000 employees and three in ten from companies with 1,000 or fewer employees



Over half of respondents are located in North America and one-quarter in Europe



One-third of respondents are Scrum Masters or Internal Coaches while over one in ten are in Development Leadership and one in ten a Project/Program Manager or External Consultant

Trending

Agile continues to be top of mind for many enterprises, and with good reason.



According to 89% of respondents, high-performing Agile teams have:

People-centric values, clear culture, tools, and leadership empowerment.

That means, that if Agile is done successfully, benefit accrues not just to the individuals involved, but to the entire organization. That is a powerful reason to invest, and organizations continue to do so. Process was at the forefront of many of the changes in Agile. Traditionally, companies think about achieving one of three things: raising revenue, lowering costs, or reducing risk.

03

Company Experience with Agile

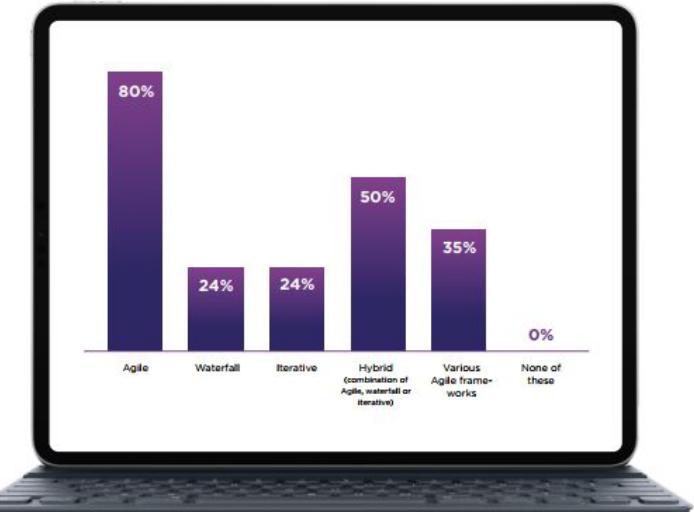
4 out of 5 respondents

say their organization has Agile teams distributed geographically.

This number has dipped slightly from last year, but it remains in the 80th percentile. We'll continue to watch the trend for this.

We pull our respondents from the Agile community, so it's not surprising that 80% of them are using Agile as their predominant approach. What is interesting is that half are using Agile in combination with other approaches, such as waterfall or iterative.

4 in 5 respondents are using Agile and half are using a combination of Agile, waterfall and/or iterative



25%

say this combination of frameworks works well for them



48%

say this combination of frameworks works somewhat well for them



27%

say this combination of frameworks doesn't work for them.

There is room for improvement, which is also supported by the evidence that not all businesses are fully satisfied with their Agile practices.

03

Company Experience with Agile

While over 7 in 10 respondents say they are satisfied with the Agile practices in their company, half are somewhat satisfied and one in five very satisfied

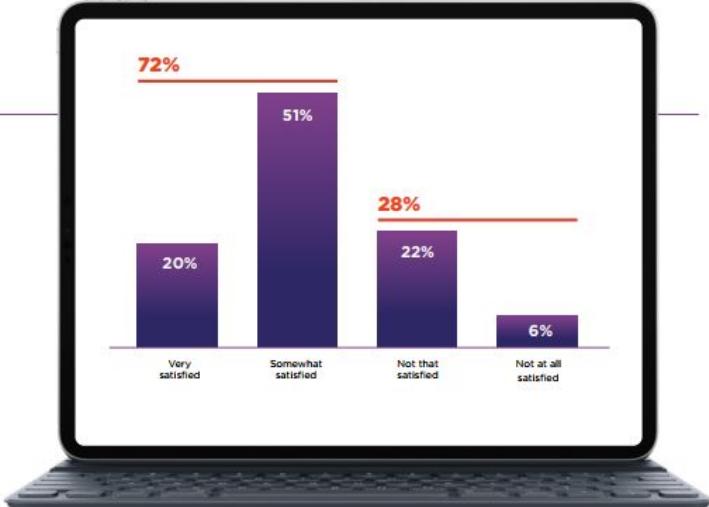
04

What is Working Well Across Agile Practices, People and Technology

What's trending from a people perspective?

Respondents told us that when they are satisfied with Agile practices in their organization, certain things are going well. The most common result satisfied people see is increased collaboration (69%). It's easier to do their jobs with others. The second most common result that slightly more than half experienced is better alignment to business needs (54%). After that, the next two most common results that make them satisfied are a better work environment (39%) and an increased visibility in the application development lifecycle.

This resonates with the increased use of business objectives as metrics mentioned above, as well as the increase in aligning work priorities to business goals. These results validate the statement that more organizations are scaling Agile across the enterprise to achieve digital transformation. It is also worth noting that the increase in visibility to the application development lifecycle is a clear benefit of a continued focus on Agile tools and methodologies for software development itself.



04

What is Working Well Across Agile Practices, People and Technology

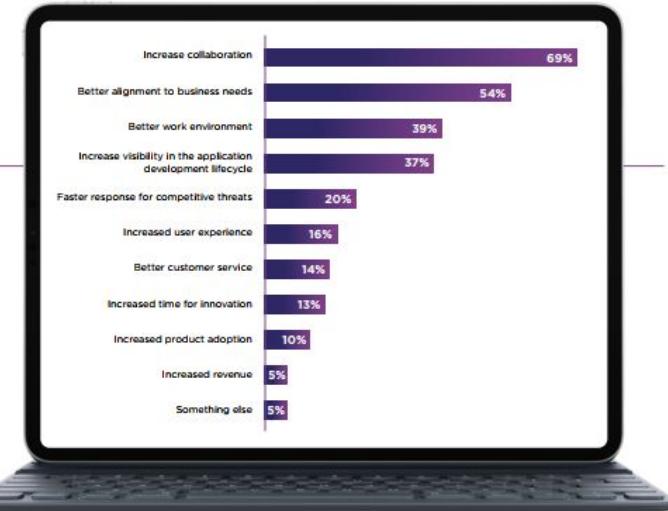
If we look to our opening statement, that high-performing Agile teams have people-centric values, clear culture, tools and leadership empowerment, then these are the results – it's a better place to work, people work together more, they have increased visibility into their work and it leads to better alignment to the business needs.

Among those who are satisfied with Agile practices at their company, seven in ten say they are satisfied because of increase collaboration and over half because of a better alignment to business needs

What are the best practices of high-performance Agile teams?

55% of respondents listed high levels of cross-collaboration and communication as a best practice

51% of respondents listed continuous improvement techniques as a best practice



04

What is Working Well Across Agile Practices, People and Technology



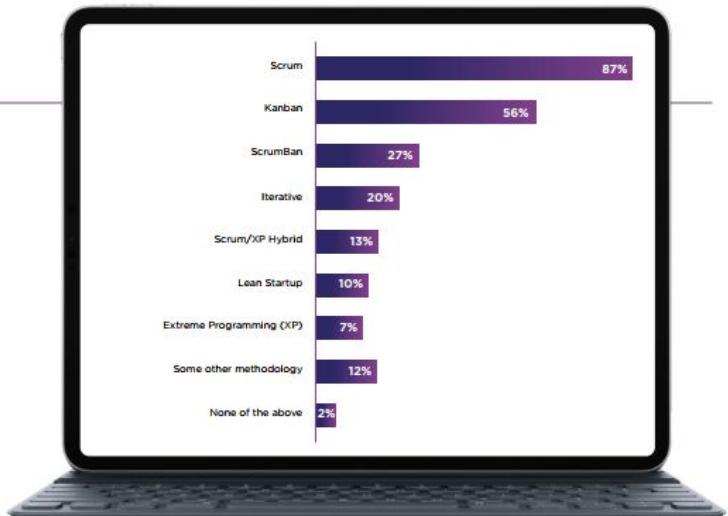
Almost 9 in 10 respondents say they are currently leveraging Scrum while over half are currently leveraging Kanban

If we look at how that has changed over the last 3 years of the survey, Scrum continues to lead, increasing from 58% in the 14th survey to 87% in the current survey. Kanban use has exploded from 7% in the 14th survey to 56% in the current survey. Scrumban has grown modestly from 10% in the 14th survey to 27% in the current survey. Iterative has also grown from 4% in the 14th survey to 20% in this year's survey.

The most popular framework continues to be the Scaled Agile Framework (SAFe). Compared to previous years, SAFe has gone from 37% in year 15 to 53% this year. Scrum@Scale/Scrum of Scrums also saw a rise after years of decline, from 9% in year 15 back to 28% this year. Lean Management likewise grew after falling for several years from 2% in year 15 to 8% this year.

We saw the rise of a new tool in this space, virtual digital whiteboards, as exemplified by Mural or Miro. This may be in part a response to the geographic disbursement of teams and the maintaining popularity of remote and hybrid workforces. We'll continue to track this trend to see what other tools may arise to join this space.

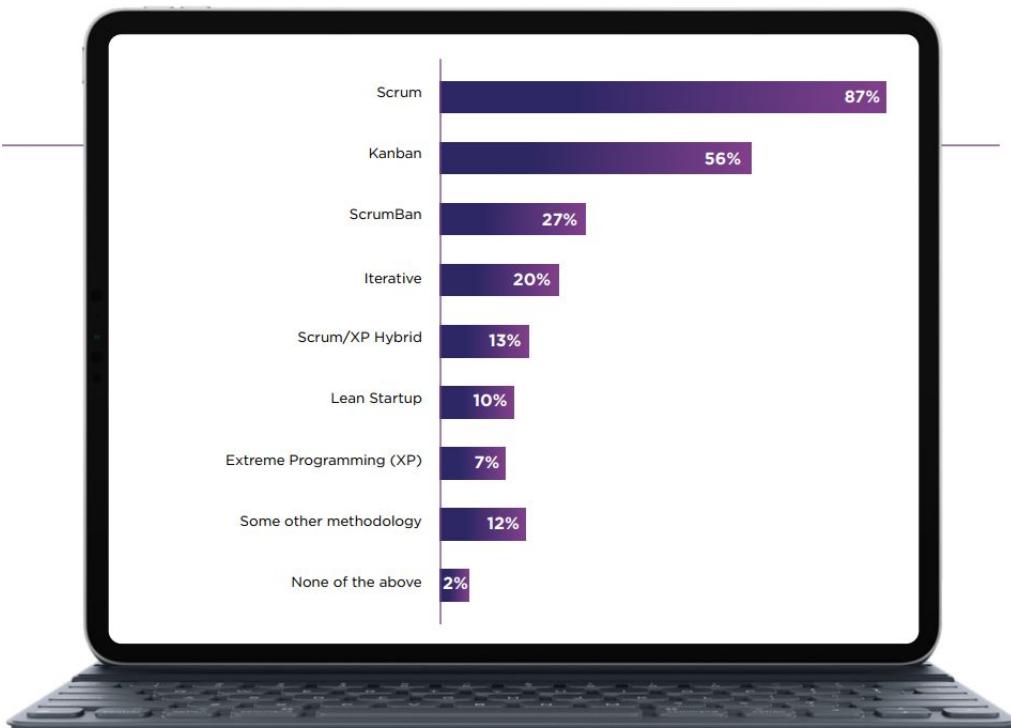
For methodologies, just under nine in ten respondents are leveraging Scrum, and over half are leveraging Kanban.



Metodologias mais usadas

- Scrum;
- Kanban;
- Lean;
- Extreme Programming (XP);
- Test Driven Development (TDD);
- Feature Driven Development (FDD);
- Microsoft Solutions Framework (MSF);
- Desenvolvimento de Sistemas Dinâmicos (Dynamic System Development Model);
- Adaptive Software Development (ASD);
- Scaled Agile Framework (SAFe).

Metodologias mais usadas:



Fonte: 16th State of Agile Report

Relembrando ...

Metodologia ágil é uma soma de habilidades e práticas usadas na gestão de projetos que tem por objetivo garantir mais **rapidez, eficiência e flexibilidade**.

A principal característica de uma metodologia ágil é que ela proporciona uma **melhoria contínua** de todos os processos da organização colaborando para um **aumento na cooperação entre a equipe e o cliente**.

Essa cooperação se dá por meio de **ciclos de feedbacks constantes**. Além disso, ela é caracterizada pelas entregas rápidas e de alta qualidade, bem como pela flexibilidade do escopo do projeto.

A metodologia ágil cria um **valor progressivo** no produto oferecido sempre de olho nas **necessidades dos clientes**. Ela proporciona mais **adaptabilidade** às mudanças assim como um alto nível de inovação.

Alguns dos benefícios dos Modelos ágeis de gestão de projetos:

- Melhoria de performance na empresa;
- Aumento da satisfação do público, por proporcionar uma boa experiência e uma maior entrega de valor;
- Redução de erros;
- Melhoria contínua;
- Foco em resultados.

Referências

Scrum Guide. Disponível em <<https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-PortugueseBR-3.0.pdf>>. Acesso em 03 ago. 2024.

Labone Consultoria. Disponível em <<https://www.laboneconsultoria.com.br/modelo-cascata-o-que-e/>>. Acesso em 03 ago. 2024.

State of Agile Report. Disponível em <<https://digital.ai/resource-center/analyst-reports/state-of-agile-report/>>. Acesso em 03 ago. 2024.

Siteware. Disponível em <<https://www.siteware.com.br/gestao-de-equipe/tipos-de-metodologia-agil/>>. Acesso em 03 ago. 2024.

As falhas numéricas que podem causar desastres. Disponível em
<https://www.bbc.com/portuguese/noticias/2015/05/150513_vert_fut_bug_digital_ml>. Acesso em 03 ago. 2024.

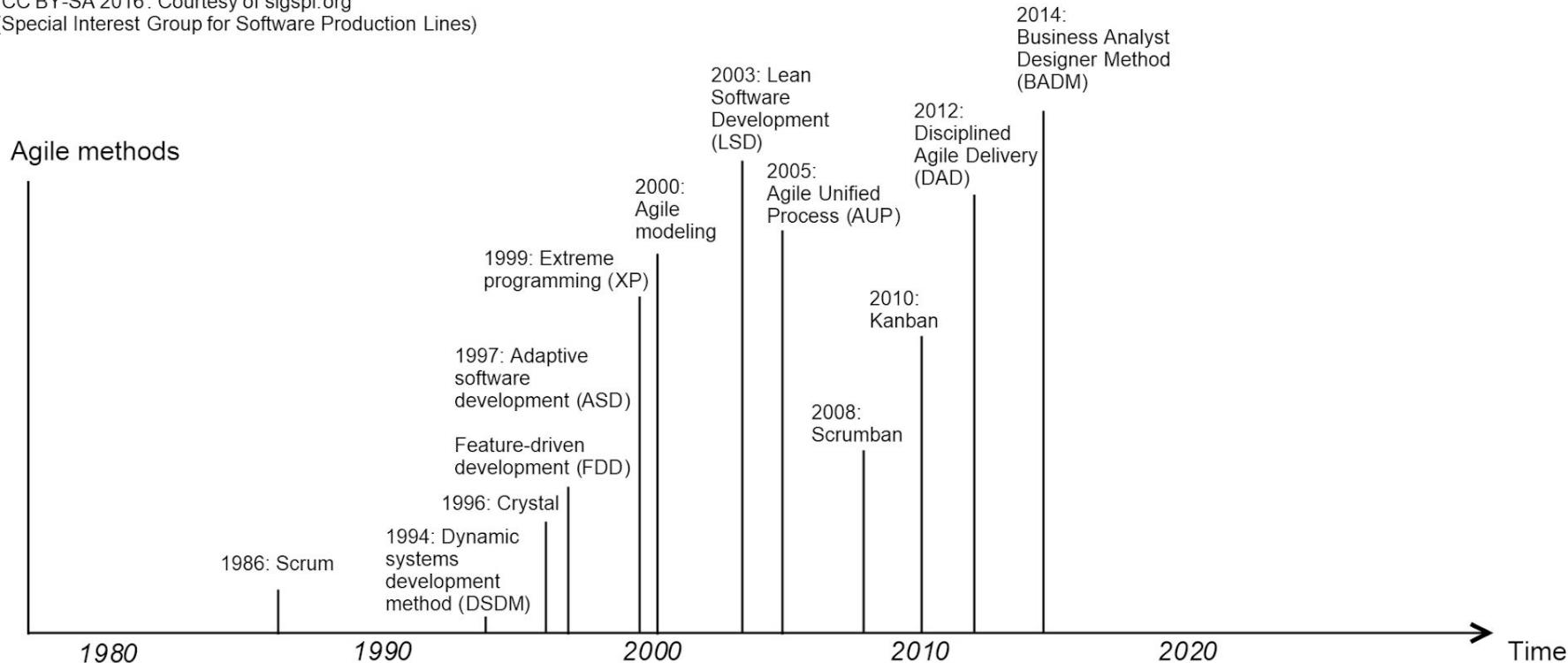


Conhecimento que transforma tudo

Linha do tempo

History of Agile

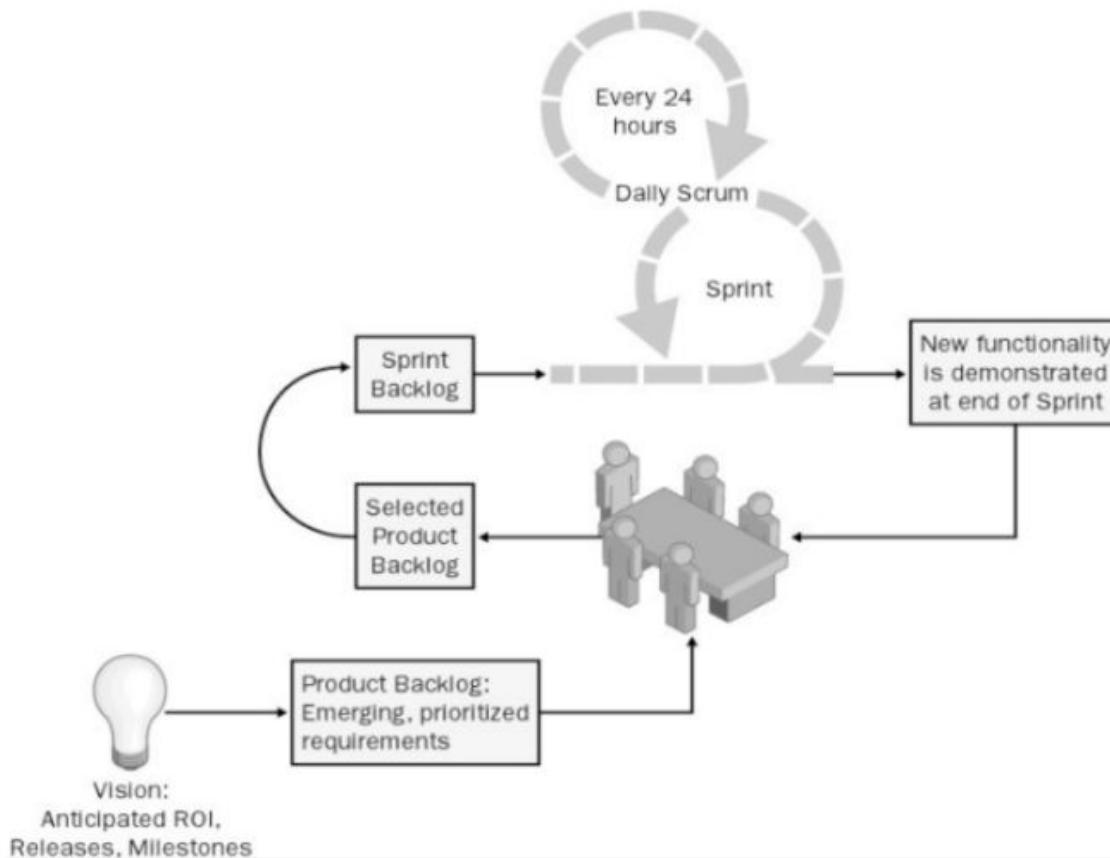
CC BY-SA 2016. Courtesy of sigspl.org
(Special Interest Group for Software Production Lines)

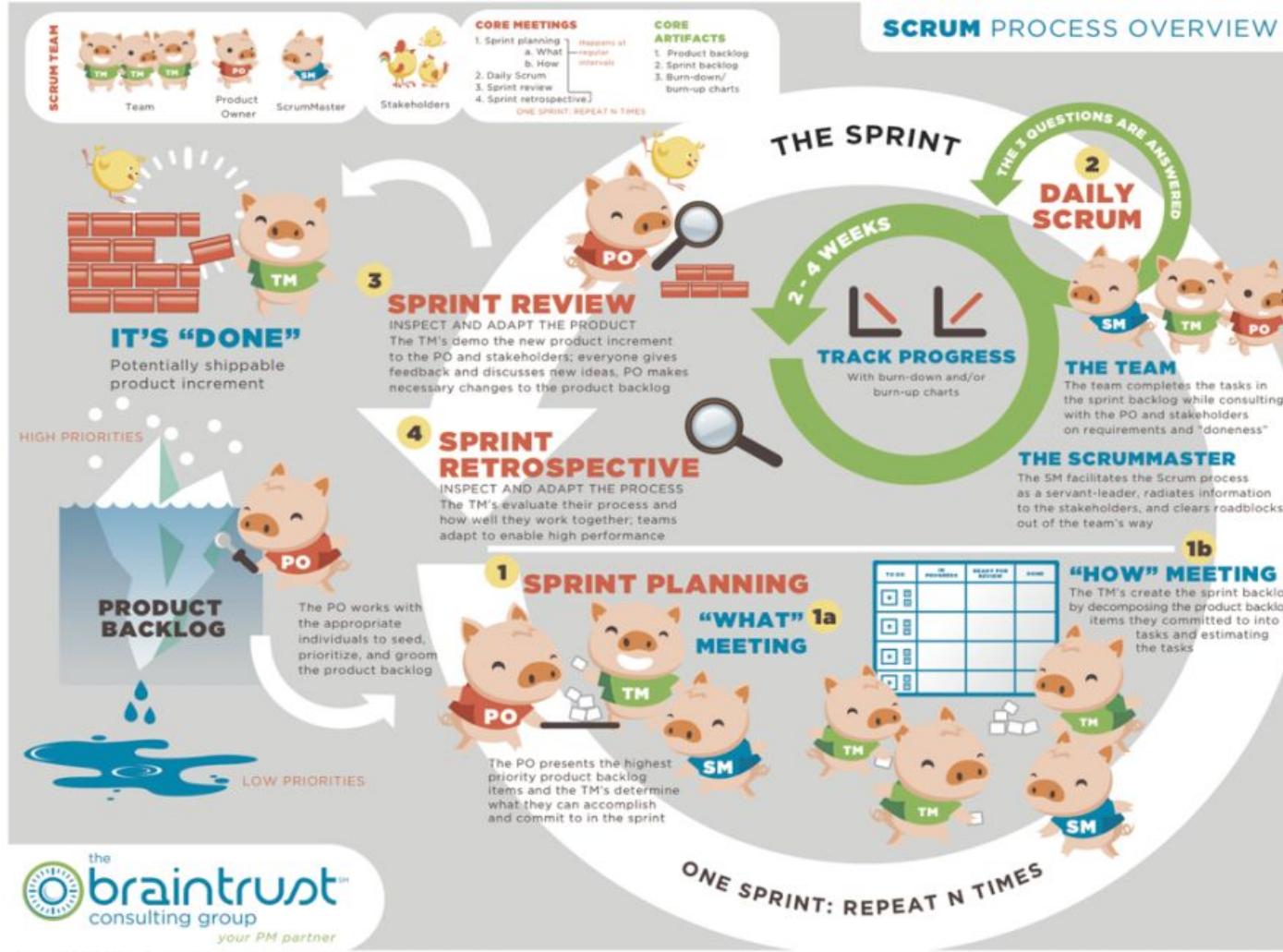




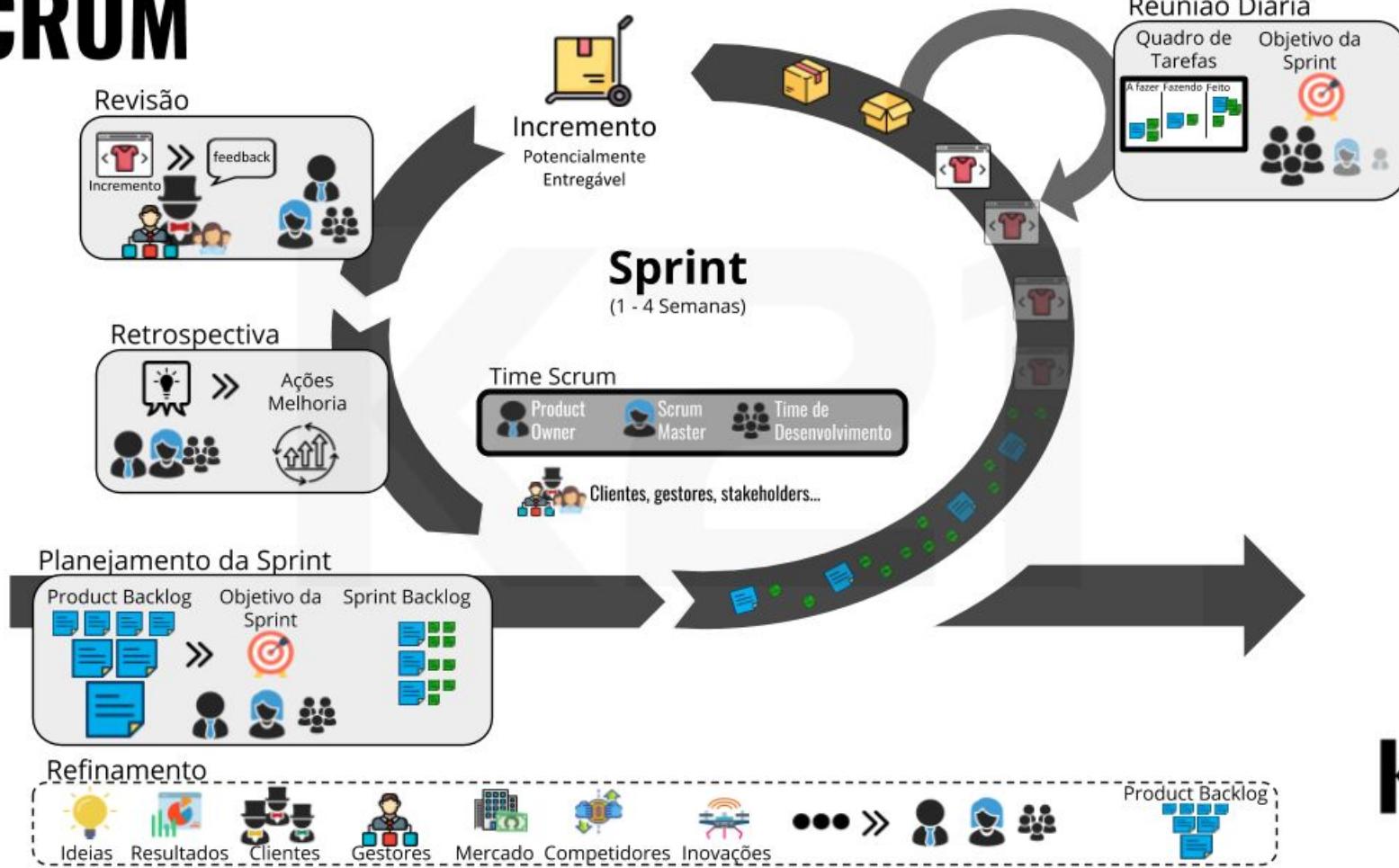
UNIVERSIDADE
VILA VELHA
ESPÍRITO SANTO

Framework Scrum





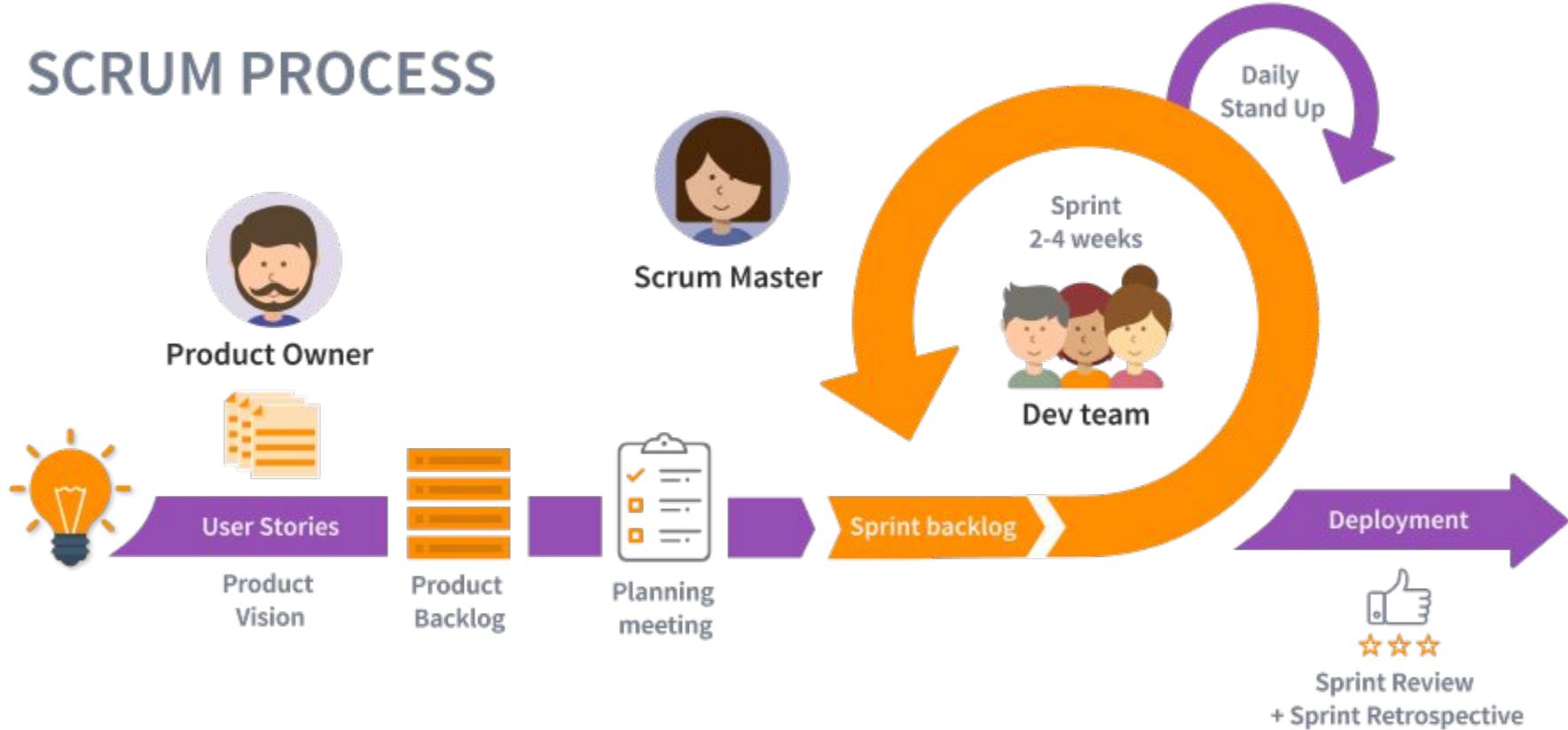
SCRUM



K21

Ilustrações do Framework

SCRUM PROCESS

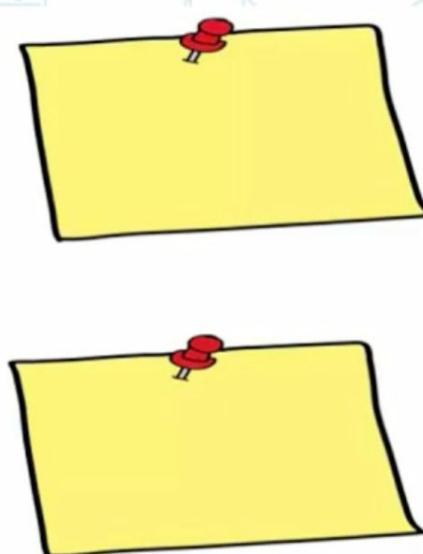


3 ARTIFACTS

Product Backlog



Sprint Backlog



Burndown Chart



3 ROLES

Product Owner



Scrum Master



Team



3 CEREMONIES

Sprint Planning

Daily Scrum

Sprint Review



Passos do Scrum

1. Obtém-se do Product Backlog os itens (requisitos) de maior valor agregado;
2. Planeja-se a iteração (sprint);
3. Faz-se acompanhamento diário;
4. Entrega-se acréscimo de funcionalidades (novo incremento) ao fim da iteração (sprint)

