**Origem do Scrum**

\* 1986 takeuchi e nonaka

- equipes pequenas e multidisciplinares

- the new product development game

\* 1993 jeff sutherland

- Documentação e implementação na Easel

\* 1995 ken sutherland

- formalização e implementação no desenvolvimento SW

\* 2001 Manifesto Ágil

**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:

\* os indivíduos e suas interações acima de procedimentos e ferramentas;

Dar prioridade nas conversas com reuniões pessoais para tomada de decisões para correção de algum problema específico.

\* O funcionamento do software acima de documentação abrangente

Deve existir documentação, porém a funcionalidade do software e correção do bug deve ser prioridade em relação a documentação.

A documentação deve ser suficiente para o cliente e desenvolver entender, não podendo ser exagerada.

\* A colaboração dos clientes acima da negociação de contratos.

O cliente deve estar do seu lado pois nos projetos tanto o cliente ou o desenvolvedor pode precisar um do outro.

Ex.: 4 entregas, entrega 1 correta e no prazo.

Porém na entrega 2 existe problema com time de desenvolvimento e tenta negociar com o cliente a redução de itens na entrega.

A colaboração entre as partes e o jogo de cintura ajuda em eventuais necessidades de ambas as partes.

\* A capacidade de resposta a mudanças acima de um plano preestabelecido.

contornar as situações e dificuldades do dia a dia para entrega e desenvolvimentos para tomar decisões com base no atual momento e dificuldades.

\* Ou seja, mesmo havendo valor nos itens a direita, valorizamos mais os itens a esquerda.

- Os indivíduos e suas interações acima de procedimentos e ferramentas;

- O funcionamento do software acima de documentação abrangente;

- A colaboração dos clientes acima da negociação de contratos;

- A capacidade de respostas a mudanças acima de um plano preestabelecido.

Entre 17 integrantes que escreveram o manifesto estão ken schwaber e jeff sutherland (2001).

**O que é ser ágil**

Agilidade e velocidade com controle.

Nos projetos a entrega deve ser ágil, porém o desenvolvendo ágil deve prever os testes das documentações e sem bugs.

**Quando utilizar o Scrum**

Para que um projeto seja simples a equipe deve saber sobre os requerimentos e ter conhecimento da tecnologia que será aplicada no projeto.

**\* Projeto Simples**

Quanto mais existir conhecimento destes dois pontos mas o projeto será "simples" (sem dúvidas nas regras e na construção).

**\* Projeto Complicado**

Quase todas as regras e suas construções são conhecidas.

**\* Projeto Complexo**

Conhecimento médio das regras de negócios e de sua construção.

**\* Projeto Caos**

Conhecimento baixo das regras de negócios e de sua construção.

Scrum é um framework para desenvolvimento ágil de projetos, uma metodologia ágil de gestão de projetos que se baseia em ciclos curtos de trabalho. Chamados de sprints, com duração de duas a quatro semanas.

Durante cada sprint, a equipe se concentra em entregar um conjunto de funcionalidades ou melhorias ao produto.

O Scrum é uma metodologia eficaz para gerenciar projetos complexos e dinâmicos, pois permite que as equipes se adaptem às mudanças e entreguem valor aos clientes com rapidez.

O Scrum pode ser usado em qualquer setor, mas é mais comumente usado no desenvolvimento de software.

O desenvolvimento cascata é uma metodologia linear, o que significa que as fases são executadas em sequência. Uma vez que uma fase é concluída, não é possível voltar atrás e fazer alterações.

O desenvolvimento cascata é uma metodologia eficaz para projetos com requisitos estáveis e que não estão sujeitos a mudanças ao longo do tempo.

No entanto, o desenvolvimento cascata também tem algumas desvantagens, como:

**- Dificuldade de adaptação:** como as fases são lineares, é difícil adaptar o projeto às mudanças que ocorrem ao longo do tempo.

**- Alto risco:** como as fases são lineares, um erro em uma fase pode afetar todas as fases subsequentes.

O Scrum é uma metodologia que pode ser utilizada em qualquer projeto, mas é mais adequado para projetos que apresentam as seguintes características:

**- Complexidade:** o Scrum é uma metodologia eficaz para gerenciar projetos complexos, pois permite que as equipes se adaptem às mudanças e entregam valor aos clientes com rapidez.

**- Dinâmica:** o Scrum é uma metodologia flexível, que pode ser adaptada às mudanças que ocorrem durante o projeto.

**- Requisitos mutáveis:** o Scrum é uma metodologia que permite que os requisitos sejam alterados ao longo do projeto, sem comprometer o cronograma ou o orçamento.

**- Trabalhar em equipe:** o Scrum é uma metodologia que incentiva a colaboração entre os membros da equipe.

**- Necessidade de entregas frequentes:** o Scrum permite que as equipes entreguem valor aos clientes de forma rápida e frequente.

**- Necessidade de feedback constante:** o Scrum incentiva o feedback constante do cliente, o que ajuda a garantir a qualidade do produto.

A colaboração do time e a vontade de entregar as sprints é muito importante.

O scrum é:

- Leve

- Simples de entender

- Extremamente difícil de dominar, sempre será utilizado por outras equipes que observam o ambiente em que se aplica o scrum.

O Scrum Guide possuí:

- Papéis (3)

- Eventos (5)

- Artefatos (N)

- Regras

O scrum é baseado em processo incremental e iterativo. Sendo um processo empírico de gerenciamento e controle.

O processo empírico se baseia no que deu certo e na correção de rotas com base no conhecimento adquirido nas entregas anteriores.

O scrum é muito mais atitude do que processos em si, sendo escalável em projetos grandes e largos.

**Pilares do Scrum**

Os pilares do Scrum são três princípios fundamentais que sustentam a metodologia. Esses pilares são essenciais para o sucesso do Scrum. Eles ajudam a garantir que as equipes Scrum sejam eficazes, produtivas e inovadoras.

**Transparência**

Garantir que todos da equipe estejam falando a mesma língua (cliente x PO) (PO x desenvolvedores). E a definição de **Done (pronto)**.

As equipes Scrum são transparentes sobre seu trabalho e compartilham informações de forma aberta e honesta. Isso ajuda a garantir que todos estejam na mesma página e que as decisões sejam tomadas com base em informações precisas.

**Inspeção**

A cada sprint inspecionar o que está acontecendo para verificar se deve ter ajuste da rota.

As equipes Scrum inspecionam seu trabalho regularmente para identificar problemas e oportunidades de melhoria. Isso ajuda a garantir que o trabalho seja de alta qualidade e que as equipes estejam no caminho certo para alcançar seus objetivos.

**Adaptação**

Ao perceber o que está incorreto no projeto, realizar ajustes necessários o mais rápido possível.

As equipes Scrum se adaptam às mudanças à medida que elas acontecem. Isso ajuda a garantir que as equipes sejam capazes de responder às mudanças nas necessidades dos clientes ou no ambiente.

Aqui estão alguns exemplos de como os pilares do Scrum podem ser aplicados no mundo real:

* **Transparência:** uma equipe Scrum pode usar um quadro Kanban para visualizar seu trabalho. Isso ajuda a garantir que todos na equipe saibam o que está acontecendo e que as tarefas sejam priorizadas de forma eficaz.
* **Inspeção:** uma equipe Scrum pode realizar uma reunião diária para discutir o progresso do sprint. Isso ajuda a identificar problemas potenciais e a tomar medidas corretivas antes que eles se tornem grandes problemas.
* **Adaptação:** uma equipe Scrum pode usar uma revisão de sprint para receber feedback do cliente. Isso ajuda a garantir que o produto atenda às necessidades dos clientes e que possa ser adaptado às mudanças nas necessidades dos clientes.

**Definição de Done (Pronto)**

**Definition of Done (DOD)**

Todas as tarefas que precisam estar concluídas para que o PB itens seja considerado como **Pronto/Done**.

A definição de **Done** é um acordo entre o Product Owner, a equipe Scrum e quaisquer outras partes interessadas. Ela deve ser clara e concisa, de modo que todos estejam na mesma página sobre o que significa um item estar completo.

A definição de done é importante porque ajuda a garantir que os itens do Backlog do Produto sejam entregues com qualidade e que atendam às expectativas do cliente. Ela também ajuda a equipe Scrum a se organizar e a priorizar seu trabalho.

Exemplo para um item estar **Done**:

* Item deve estar documentado (PO) e aprovado pelo cliente;
* Nenhum bug impeditivo pode estar associado ao item;
* Teste deve ser realizado no ambiente de homologação;
* Código fonte deve estar no repositório e versionado;
* Manual do produto deve está atualizado com este item;
* Documento de gestão de mudança deve contemplar este item;

Mais itens possíveis para análise:

* Teste de performance
* Teste unitário
* Teste de regressão
* Teste de aceitação do usuário
* Revisão do código fonte

Em última análise, a definição de done é específica para cada projeto e organização. No entanto, as características principais mencionadas acima são geralmente aplicáveis em todos os contextos.

**Desenvolvimento Iterativo e Incremental**

O Scrum é baseado no desenvolvimento **iterativo** e **incremental**.

A entrega é por parte no scrum, a iteração dentro de uma sprint é baseado nos processos, requisitos, planejamento e design, desenvolvimento e testes, feedback e integração.

Ao final desta iteração iremos para próxima sprint que é incremental a sprint anterior.

Lembrando que os pilares transparência, inspeção e adaptação fazem parte do scrum a cada iteração o time deve ficar melhor tendo em vista a execução repetitiva do processo e a aplicação dos pilares para o melhor desenvolvimento.

**Incremental vs. Sequencial**

O desenvolvimento de software pode ser baseado em dos tipos de metodologias para realizações de projetos.

**Sequencial**

O projeto é desenvolvido em uma sequência linear, com cada etapa sendo concluída antes da próxima começar.

**Vantagens:**

* É mais fácil de planejar e gerenciar do que o desenvolvimento incremental.
* Requer menos comunicação entre a equipe de desenvolvimento e o cliente.
* É mais fácil documentar o processo de desenvolvimento.

**Desvantagens:**

* O cliente só pode fornecer feedback após o projeto ter sido concluído.
* É mais difícil de adaptar o projeto às mudanças nas necessidades.
* O cliente só pode começar a usar o produto após o projeto ter sido concluído.



**Incremental**

O projeto é dividido em partes menores, chamadas de incrementos, que são desenvolvidos e entregues de forma iterativa.

**Vantagens:**

* Permite um feedback mais frequente do cliente e a adaptação do projeto às mudanças nas necessidades.
* Reduz o risco de erros e problemas, pois cada incremento é testado antes de passar para o próximo.
* Permite que o cliente comece a usar o produto mais cedo, mesmo que ainda não esteja completo.

**Desvantagens:**

* Pode ser mais difícil de planejar e gerenciar do que o desenvolvimento sequencial.
* Requer uma comunicação mais frequente entre a equipe de desenvolvimento e o cliente.
* Pode ser mais difícil de documentar o processo de desenvolvimento.



A escolha entre o desenvolvimento incremental e o desenvolvimento sequencial depende de uma série de fatores, como a natureza do projeto, as necessidades do cliente e a experiência da equipe de desenvolvimento.

Os dois tipos de desenvolvimentos podem ter o mesmo tempo de execução de entrega, porém o feedback constante no desenvolvimento incremental pode ser mais vantajoso e o desenvolvimento pode sofrer maiores intervenções para melhorias e correções de bugs.

No incremental ainda é possível gerir melhor o prazo de entrega do projeto, tendo em vista a divisão das entregas através das sprints e a definição de entregas e a possibilidade de melhor “visualizar” se o prazo das entregas estão corretas, podendo desta forma ainda se necessário ser compensado através das entregas das próximas sprints.

Aqui estão alguns exemplos de quando cada abordagem pode ser mais adequada:

**Desenvolvimento incremental**

* Desenvolvimento de software com requisitos que podem mudar ao longo do tempo.
* Desenvolvimento de produtos com um ciclo de vida curto.
* Desenvolvimento de produtos com um alto grau de incerteza.

**Desenvolvimento sequencial:**

* Desenvolvimento de produtos com requisitos bem definidos.
* Desenvolvimento de produtos com um ciclo de vida longo.
* Desenvolvimento de produtos com um baixo grau de incerteza.

**Processos Preditivos e Empíricos**

Os processos de desenvolvimento de software podem ser divididos em duas categorias principais: preditivos e empíricos.

**Processos preditivos**

Baseiam-se na estimativa e no planejamento de todo o trabalho a ser feito antes do início do desenvolvimento. Assumem que os requisitos são bem conhecidos e que não mudarão significativamente ao longo do projeto.



**Exemplos:**

* Cascata
* V-Model

**Processos empíricos**

Baseiam-se no desenvolvimento iterativo e incremental do software, com feedback frequente do cliente. Assumem que os requisitos são incompletos e que podem mudar ao longo do projeto.



Sequência pequenas e definidas de especificação e desenvolvimento.

**Exemplos:**

* Scrum
* Kanban

Diferenças entre os processos:

| **Característica** | **Processos Preditivos** | **Processos Empíricos** |
| --- | --- | --- |
| **Foco** | Planejamento e controle | Adaptabilidade e mudança |
| **Requisitos** | Completos e estáveis | Incompletos e mutáveis |
| **Abordagem** | Sequencial | Iterativa |
| **Feedback** | No final do projeto | Frequente |
| **Risco** | Maior | Menor |
| **Custo** | Menor | Maior |
| **Complexidade** | Menor | Maior |
|  |  |  |

Aqui estão alguns exemplos de quando cada abordagem pode ser mais adequada:

**Processos preditivos:**

* Projetos com requisitos bem definidos e estáveis.
* Projetos com baixo risco de mudança.
* Projetos com prazos e orçamentos apertados.

**Processos empíricos:**

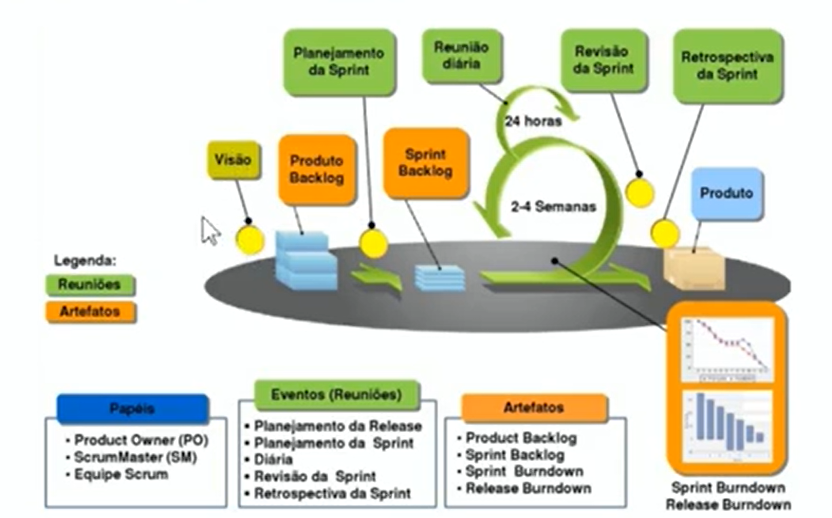
* Projetos com requisitos incompletos ou mutáveis.
* Projetos com alto risco de mudança.
* Projetos que exigem flexibilidade e adaptabilidade.

No desenvolvimento as funcionalidades e os projetos:

* 35% dos requerimentos mudam ao longo do projeto;
* 65% das funcionalidades não são ou raramente são utilizadas.

Um protótipo navegável pode garantir que o projeto seja desenvolvido da forma que o cliente idealizou em sua cabeça o que ele deseja.

**Processo Scrum - Visão Macro**



**Visão** - O que é desejado.

**Backlog Produto** - Lista de desejos, priorizadas do mais relevantes e menos relevantes.

**Planejamento Sprints** - Do que e Como. Apresentação das funcionalidades que deseja ser desenvolvidas e definição de metas da sprints e a equipe de desenvolvimento irá verificar como executar os itens e funcionalidade que deve ser desenvolvimento.

**Sprint Backlog** - Os itens que foram planejados para a sprint.

**Período de Desenvolvimento** - De 2 a 4 semanas

**Reunião diária** - Em pé de no máximo de 15 minutos.

**Revisão da Sprint** - Apresentação das funcionalidades da sprint, inspeção do que foi produzido

**Retrospectiva da Sprint** - Inspeção do time para o time, o que foi feito de bom para continuar fazendo e o que foi feito de ruim para parar de fazer e a forma de como corrigir.

**Produto** - Entrega do integrável para incrementar o produto em desenvolvimento.

**O Time Scrum em Ação**

Conheceremos o time scrum, seus componentes, objetivos e desafios. Além disso, abordaremos como a interação entre eles é importante e essencial para todo o processo scrum.

* **Product Owner**
  + Conhece muito sobre o produto/negócio
  + Garante entrega de valor
  + Gerencia o backlog do produto
* **Development Team**
  + Pessoas que desenvolve o produto
  + Entrega de software pronto/Done
  + Auto gerencia
* **Scrum Master**
  + Mantem todos motivados
  + Gerencia e estimula o processo do scrum
  + Remove impedimentos



**O Scrum Master - Principais Objetivos**

No desenvolvimento Scrum, o Scrum Master é um facilitador e líder que garante que a equipe Scrum siga os princípios e práticas do Scrum, garantir que não haja ScrumBut’s

Garantir que o processo scrum seja praticado na empresa de forma correta.

Um SM precisa ter perfil de liderança?

Resposta: Sim

Precisa ter certos poderes e autonomias?

Resposta: Sim, com certeza. Realizar compras, contratações, contratar serviços. Etc.

O SM é um papel gerencial ou operacional?

Resposta: Gerencial.

O que um bom SM não deve fazer?

Resposta: Não deve se preocupar com micro-gerenciamentos. Após o treinamento da equipe de gerenciamento e o PO deve garantir que estejam trabalhando da melhor forma possível e cobrar desempenho (metas, especificações e cliente feliz com entregas).

**Funções do Scrum Master**

**Facilitar eventos Scrum:** O Scrum Master é responsável por facilitar os eventos Scrum, como Sprint Planning, Daily Scrum, Sprint Review e Sprint Retrospective.

**Garantir que os princípios e práticas do Scrum sejam seguidos:** O Scrum Master é responsável por garantir que a equipe Scrum siga os princípios e práticas do Scrum.

**Remover impedimentos:** O Scrum Master é responsável por remover impedimentos que impedem o progresso da equipe Scrum.

**Proteger a equipe:** O Scrum Master protege a equipe Scrum de distrações e interrupções.

**Treinar e orientar a equipe:** O Scrum Master treina e orienta a equipe sobre os princípios e práticas do Scrum.

**Ser um líder servidor:** O Scrum Master é um líder servidor que coloca as necessidades da equipe em primeiro lugar.

**Habilidades importantes para um Scrum Master**

**Excelentes habilidades de comunicação:** O Scrum Master precisa ser capaz de se comunicar de forma eficaz com a equipe, o Product Owner e outras partes interessadas.

**Habilidades de facilitação:** O Scrum Master precisa ser capaz de facilitar os eventos Scrum de forma eficaz.

**Habilidades de resolução de problemas:** O Scrum Master precisa ser capaz de resolver problemas e remover impedimentos.

**Conhecimento de Scrum:** O Scrum Master precisa ter um conhecimento profundo dos princípios e práticas do Scrum.

**Liderança:** O Scrum Master precisa ser um líder servidor que coloca as necessidades da equipe em primeiro lugar.

**ScrumBut**

É um termo usado para descrever a implementação incompleta ou incorreta do Scrum. Ocorre quando uma equipe ou organização tenta usar o Scrum, mas não segue todos os seus princípios e práticas.

**Exemplos de ScrumBut**

**Não realizar reuniões Scrum:** Uma equipe pode decidir não realizar reuniões Scrum, como Daily Scrum ou Sprint Retrospective.

**Não ter um Product Owner:** Uma equipe pode não ter um Product Owner dedicado para definir e priorizar os requisitos do produto.

**Não ter um Scrum Master:** Uma equipe pode não ter um Scrum Master para facilitar os eventos Scrum e garantir que os princípios e práticas do Scrum sejam seguidos.

**Não ter um Backlog do Produto:** Uma equipe pode não ter um Backlog do Produto transparente e priorizado.

**Não ter um sprint de tempo fixo:** Uma equipe pode ter sprints de duração variável.

**Consequências do ScrumBut**

* **Ineficiência:** O ScrumBut pode levar a equipes ineficazes e improdutivas.
* **Frustração:** O ScrumBut pode frustrar a equipe e as partes interessadas.
* **Falha no projeto:** O ScrumBut pode levar ao fracasso do projeto.

**Como evitar o ScrumBut**

**Treinamento:** É importante que a equipe e as partes interessadas sejam treinadas em Scrum.

**Coaching:** Um coach de Scrum pode ajudar a equipe a implementar o Scrum de forma eficaz.

**Compromisso:** É importante que a organização esteja comprometida com a implementação do Scrum.

O ScrumBut é um problema sério que pode ter consequências negativas para o projeto. É importante tomar medidas para evitar o ScrumBut e garantir que o Scrum seja implementado de forma eficaz.

**Quais são os principais objetivos do scrum master em relação aos players do Scrum**

**Em relação do Product Owner:**

O Scrum Master e o Product Owner são parceiros que trabalham juntos para alcançar o sucesso do projeto. Ao colaborar, ser transparente, eficiente, empoderar e promover o crescimento um do outro, podem criar um ambiente de trabalho eficaz e produtivo que maximize as chances de sucesso.

1. Colaboração:

Facilitar uma comunicação clara e aberta entre o Scrum Master e o Product Owner.

Criar um ambiente de trabalho colaborativo onde ambos possam trabalhar juntos para alcançar os objetivos do projeto.

Alinhar-se com o Product Owner sobre as prioridades do produto e as necessidades do cliente.

Ajudar a encontrar técnicas no gerenciamento do *Backlog*\*.

\* ***Backlog*** é uma lista de tarefas ou itens de trabalho que precisam ser concluídos. No contexto do desenvolvimento de software, o backlog do produto é uma lista de todos os requisitos do produto que ainda não foram implementados.

1. Transparência:

Garantir que o Product Owner tenha visibilidade do progresso do projeto e das atividades da equipe.

Fornecer feedback honesto e transparente ao Product Owner sobre o desempenho da equipe e os riscos do projeto.

Manter o Product Owner atualizado sobre as mudanças nos requisitos do produto e no backlog do produto.

1. Eficiência:

Ajudar o Product Owner a definir e priorizar os requisitos do produto de forma eficaz.

Facilitar a criação e o gerenciamento do backlog do produto.

Compreender o planejamento do produto.

Proteger o Product Owner de distrações e interrupções que possam prejudicar sua produtividade.

1. Empoderamento:

Ajudar o Product Owner a tomar decisões informadas sobre o produto.

Capacitar o Product Owner para liderar o desenvolvimento do produto e representar os interesses do cliente.

Apoiar o Product Owner na resolução de problemas e na superação de obstáculos.

Garantiar que o Product Owner esteja preparada para cada sprint, tenha conhecimento de tudo que será desenvolvido para cada sprint.

1. Crescimento:

Auxiliar o Product Owner no desenvolvimento de suas habilidades e conhecimentos.

Fornecer feedback construtivo ao Product Owner sobre seu desempenho.

Criar um ambiente de aprendizado e desenvolvimento contínuo para o Product Owner.

**Em relação a Equipe de Desenvolvimento:**

O Scrum Master é um papel crucial no desenvolvimento Scrum. Ao facilitar, remover impedimentos, treinar, motivar e liderar a equipe de desenvolvimento, o Scrum Master pode ajudar a equipe a ser mais eficaz, produtiva e inovadora.

1. Facilitação:

Facilitar eventos Scrum, como Sprint Planning, Daily Scrum, Sprint Review e Sprint Retrospective.

Criar um ambiente de trabalho colaborativo e de confiança onde a equipe possa se comunicar abertamente e resolver problemas de forma eficaz.

Promover a comunicação clara e transparente entre os membros da equipe.

1. Remoção de Impedimentos:

Identificar e remover impedimentos que afetam o progresso da equipe.

Proteger a equipe de distrações e interrupções.

Ajudar a equipe a encontrar soluções para os problemas que enfrentam.

1. Coaching e Desenvolvimento:

Treinar a equipe sobre os princípios e práticas do Scrum.

Ajudar a equipe a melhorar suas habilidades e conhecimentos.

Fornecer feedback construtivo à equipe sobre seu desempenho.

Ensina a autogestão para a equipe.

Estimula a equipe a ser autossuficiente (interdisciplinar).

Garantir que a equipe tenha as skills necessárias para o projeto.

1. Motivação e Engajamento:

Manter a equipe motivada e engajada no projeto.

Criar um ambiente de trabalho positivo e divertido.

Celebrar as conquistas da equipe.

1. Liderança:

Ser um líder servidor que coloca as necessidades da equipe em primeiro lugar.

Inspirar e motivar a equipe a alcançar seus objetivos.

Tomar decisões difíceis quando necessário.

**Em relação a Organização:**

O Scrum Master é um agente de mudança na organização. Ao implementar o Scrum, evangelizar o Scrum, melhorar a cultura organizacional, alinhar o Scrum com os objetivos da organização e medir o sucesso, o Scrum Master pode ajudar a organização a se tornar mais eficiente, eficaz e inovadora.

1. Implementar o Scrum:

Facilitar a implementação do Scrum na organização.

Treinar os membros da organização sobre os princípios e práticas do Scrum.

Ajudar a organização a adaptar o Scrum às suas necessidades específicas.

Fazer o time scrum ser entendido

1. Evangelizar o Scrum:

Promover a compreensão e adoção do Scrum na organização.

Criar um ambiente onde o Scrum seja valorizado e apoiado.

Ajudar a organização a superar os desafios da implementação do Scrum.

1. Melhorar a cultura organizacional:

Promover uma cultura de colaboração, comunicação e aprendizado contínuo.

Ajudar a organização a se tornar mais ágil e adaptável.

Fortalecer o trabalho em equipe e a auto-organização.

1. Alinhar o Scrum com os objetivos da organização:

Assegurar que o Scrum esteja alinhado com a estratégia e os objetivos da organização.

Ajudar a organização a alcançar seus objetivos de forma mais eficaz.

Demonstrar o valor do Scrum para a organização.

1. Medir e monitorar o sucesso:

Medir o sucesso do Scrum na organização.

Monitorar o desempenho da equipe e identificar áreas de melhoria.

Fornecer feedback à organização sobre o progresso do Scrum.