**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:**

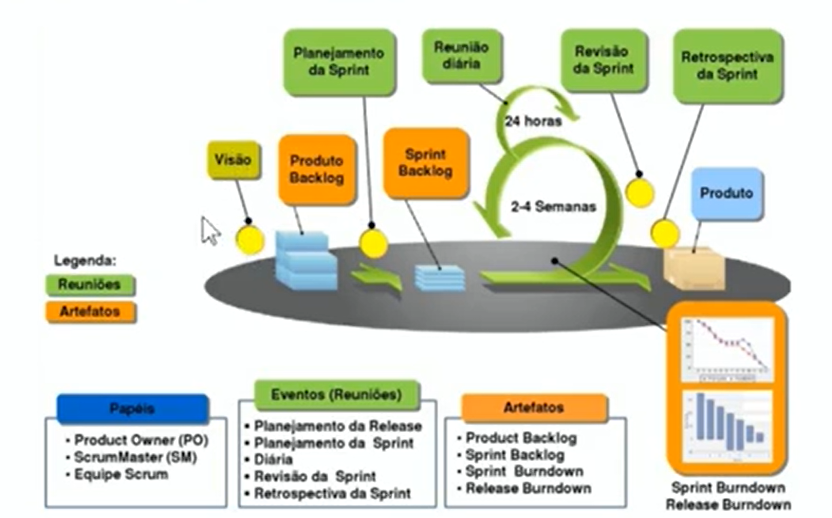
* Projetos com requisitos incompletos ou mutáveis.
* Projetos com alto risco de mudança.
* Projetos que exigem flexibilidade e adaptabilidade.

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.

**A Equipe de Desenvolvimento**

O principal objetivo da equipe de desenvolvimento é transformar **Product Backlog Items** em incrementos de software pronto “Done”.

De acordo com scrum gid todos devem programar e não deve conter alguém que tenha que gerenciar as pessoas por isso o tamanho deve ser no máximo 9 pessoas.

***Product Backlog*** *é uma lista priorizada de todos os requisitos do produto que ainda não foram implementados.*

***Product Backlog Items (PBIs)*** *são itens de trabalho no Product Backlog que representam funcionalidades ou requisitos do produto a ser desenvolvido*

A equipe deve ser:

* Auto-Organizados
  + Capazes de dividir as tarefas entre si, de forma que, no final da sprint, todo o trabalho necessário esteja pronto.
* Multidisciplinar
  + A equipe é composta por profissionais com diferentes habilidades e conhecimentos, o que permite que ela trabalhe de forma mais eficiente e eficaz.
* Comprometido com a meta
  + A equipe está focada em entregar valor ao cliente e trabalha para atender às suas necessidades.
* Tamanho: 6 +/- 3
  + A equipe deve ser pequena o suficiente para ser ágil e eficaz.
* Comunicativos
  + A equipe se comunica de forma clara e frequente para garantir que todos estejam alinhados e trabalhando em conjunto.
* Resolvem seus conflitos
  + A equipe tem autonomia para tomar decisões e resolver problemas.
* Transparência
  + O trabalho da equipe é transparente e acessível a todos os stakeholders.

O scrum master deve estimular a comunicação do time.

Toda a equipe de desenvolvimento deve estar comprometida com a meta e ela é responsável pelo desenvolvimento e entrega e a divisão da tarefa entre membros da equipe.

Quando não houver comprometimento de todos os membros da equipe, o time deve buscar as razões e colocar os pingos nos “i’s” internamente para sanar os problemas e se auto-regular.

Caso os ajustes não forem suficientes para que o membro não comprometido volte a ter o foco adequado deve ser levado ao scrum master para remoção do membro.

**Quais são os desafios da equipe de desenvolvimento**

A equipe de desenvolvimento enfrenta diversos desafios, mas é possível superá-los com o investimento em comunicação, treinamento, organização, liderança e cultura de mudança.

Como superar esse desafios:

* Responder primeiramente para um time e não para uma pessoa;
* Comunicar-se e entender-se;
* Estar aberto a constante evolução técnica;
* Estar sempre focado na meta;
* Se o mais transparente e sincero possível;
* Colaborar entre si;
* Conhecer a sua velocidade;
* Adotar uma cultura de mudança e adaptabilidade;
* Investir em comunicação e treinamento;
* Melhorar a organização e o planejamento do trabalho;
* Criar um ambiente de trabalho motivador e positivo.

**Product Owner**

O Product Owner (PO) é o responsável por definir a visão do produto, priorizar as funcionalidades e garantir que o produto atenda às necessidades do cliente. É um papel crucial no desenvolvimento ágil, pois é o elo entre o cliente, a equipe de desenvolvimento e os stakeholders.

Product Owner é como um gerente de produtos comum, o PO fica entre o cliente (stakeholders) e a equipe de desenvolvimento.

***Stakeholders*** *são indivíduos ou grupos que têm interesse no produto ou projeto. Eles podem ser clientes, usuários, equipe de desenvolvimento, gerentes, investidores, etc.*



**Quais são características do Product Owner**

* **Colaborativo** com o time e com seu cliente. **Mudança de escopo**? Sem problemas!
  + Compreender as necessidades do cliente: O Product Owner precisa ter um bom relacionamento com o cliente para entender suas necessidades e expectativas.
* **Estimula** e incentiva o time.
  + Colaborar com a equipe de desenvolvimento: O Product Owner precisa colaborar com a equipe de desenvolvimento para garantir que o produto seja desenvolvido de acordo com a visão do produto.
* Faz entregas com **frequência**.
* **Otimiza** a produtividade do time.
  + estimular o time a entender o porquê é importante aquilo, sempre ouvindo as sugestões do time para ajuste de alguma tarefa verificando se é possível o ajuste sem comprometer a demanda e necessidade do cliente.
* Somente entrega de itens de **alto valor** agregado.
  + Priorizar as funcionalidades: O Product Owner precisa priorizar as funcionalidades do produto de acordo com o valor que agregam ao cliente.
* **Dispensa** itens de valor negativo
  + Entregar valor ao cliente: O objetivo principal do Product Owner é entregar valor ao cliente.
* Lidar com mudanças: O Product Owner precisa ser capaz de lidar com mudanças nas prioridades do cliente ou do mercado.
* O Product Owner tem uma visão clara do produto e de como ele atenderá às necessidades do cliente.
* O Product Owner é um excelente comunicador e é capaz de se comunicar com diferentes públicos.

O Product Owner possui a última palavra sobre o que será feito em relação ao produto.

**Quais são os principais objetivos do Product Owner**

* Colher o backlog do produto (funcionalidades) cliente/mercado
* Ordenar o product backlog
  + Gerenciar o Product Backlog: O PO precisa manter o Product Backlog atualizado e priorizado.
* Planejar as releases
  + Priorização: O PO é capaz de priorizar as funcionalidades do produto de forma eficaz.
* Detalhar e explicar o product backlog de maneira clara para o time
  + Para fazer isso pode ser realizado protótipo e documentação de requisitos.
* Satisfazer as necessidades do cliente: O PO precisa garantir que o produto atenda às necessidades do cliente.
* Maximizar o retorno do investimento: O PO precisa garantir que o produto maximize o retorno do investimento para a empresa.
* Criar um produto de sucesso: O PO precisa trabalhar para criar um produto de sucesso que atenda às expectativas do cliente e da empresa.

**Desafio do Product Owner na empresa**

* Falta de poder sobre o produto.
  + o que será feito
  + quando será disponibilizado
  + questão hierárquicas
* Ninguém tem permissão para mudar a prioridade acordada entre PO e a Equipe de Desenvolvimento
* O Product Owner deve ser capaz de negociar com stakeholders para garantir que o produto atenda às necessidades de todos.
* Tomada de decisão: O Product Owner deve ser capaz de tomar decisões difíceis quando necessário.

Quando existe mais de uma pessoa responsável pelo futuro do produto, o que acontece?

A tomada de decisão é fácil

Não.

Quais interesses são levados em consideração?

Pode existir uma equipe que não esteja alinhada com a necessidade e sobre qual interesse do cliente deve ser atendido, gerando conflito entre os PO’s.

Existe um alinhamento estratégico entre as direções mencionadas?

**O que é ser Proxy Product Owner**

O Proxy Product Owner (PPO) é um profissional que assume o papel de Product Owner (PO) em situações específicas. O PPO não é o dono do produto em si, mas sim um representante que atua em nome do PO real.

**Quando usar um Proxy Product Owner**

* Quando o PO está indisponível: O PPO pode ser usado quando o PO real está de férias, doente ou ocupado com outras tarefas.
* Quando o PO não tem as habilidades necessárias: O PPO pode ser usado quando o PO real não tem as habilidades ou conhecimentos específicos necessários para gerenciar o produto.
* Quando o PO está em conflito de interesses: O PPO pode ser usado quando o PO real tem um conflito de interesses que o impede de tomar decisões imparciais.

**Responsabilidades do Proxy Product Owner**

* Representar o PO: O PPO deve representar o PO real e tomar decisões em seu nome.
* Gerenciar o Product Backlog: O PPO deve manter o Product Backlog atualizado e priorizado.
* Colaborar com a equipe de desenvolvimento: O PPO deve colaborar com a equipe de desenvolvimento para garantir que o produto seja desenvolvido de acordo com a visão do produto.
* Comunicar-se com os stakeholders: O PPO deve comunicar-se com os stakeholders para entender suas necessidades e expectativas.

**A interação entre o time Scrum**

Abaixo representação gráfica entre a interação dos players do time scrum.



A interação do PO e a Equipe de desenvolvimento deve haver esclarecimento dos items, como os products backlog items devem funcionar.

A interação do PO com o Scrum master gera o planejamento do produto, o PO deve saber a ordem e o motivo para esta ordem, deve explicar o planejamento das releases e fazer com que o scrum master entenda o alinhamento estratégico que compõem a ordem demonstrada.

O scrum master interagindo com a equipe de desenvolvimento vai gerar a questão relativa ao progresso, o SM deve cobrar do time de desenvolvimento a questão com relação a bater metas estabelecidas, o SM deve cobrar desempenho para entregas e metas das sprints.

Um PO pode até ser um SM e um SM pode até fazer parte da equipe de desenvolvimento, porém um PO não deve fazer parte da equipe de desenvolvimento.

**O que é Product Backlog**

Product Backlog é uma lista priorizada de todos os requisitos do produto que ainda não foram implementados.

É um inventário ordenado de “desejos” em relação ao produto.

**Características**

* Priorizado: Os itens do Product Backlog são priorizados de acordo com o valor que agregam ao cliente e ao negócio.
* Dinâmico: O Product Backlog é dinâmico e pode ser alterado à medida que novos requisitos são descobertos ou as prioridades mudam.
* Transparente: O Product Backlog deve ser transparente e acessível a todos os membros da equipe.
  + Devem ser transparentes, todas as partes envolvidas devem ter acesso e entendimento dos itens.
* O trabalho deve ser estimado.
* O desenvolvimento e a estimativa de tempo para execução é realizado pela equipe de desenvolvimento.

**Conteúdo**

O Product Backlog pode conter diversos tipos de itens, como:

* Funcionalidades do produto
* Novas funcionalidades
* Melhorias de desempenho
* Correções de bugs
* Tarefas técnicas
* Requisitos não funcionais
* Casos de usos
* User stories

**Responsabilidades**

O Product Owner é responsável por criar, manter e priorizar o Product Backlog. A equipe de desenvolvimento é responsável por estimar o esforço e implementar os itens do Product Backlog.

**Product Backlog Items (PBIs)**

Product Backlog Items (PBIs) são itens de trabalho no Product Backlog que representam funcionalidades ou requisitos do produto a ser desenvolvido.

**Características**

* Tamanho: Os PBIs devem ser pequenos o suficiente para serem concluídos em uma única sprint.
* Estimativa: Cada PBI deve ter uma estimativa de esforço para que a equipe possa planejar o trabalho.
* Critério de Aceitação: Cada PBI deve ter critérios de aceitação claros para que a equipe possa verificar se o trabalho foi concluído com sucesso.

**Exemplos**

* Implementar um novo botão de login
* Adicionar um filtro de pesquisa
* Melhorar o desempenho da página inicial

**Benefícios**

* Visibilidade
  + Os PBIs fornecem visibilidade do trabalho que precisa ser feito.
* Priorização
  + Os PBIs permitem que a equipe se concentre nas tarefas mais importantes.
* Flexibilidade
  + Os PBIs permitem que a equipe se adapte às mudanças nas prioridades.

**Ferramentas**

Existem diversas ferramentas disponíveis para gerenciar PBIs, como Jira, Trello e Asana.

**Requerimentos não funcionais (RNFs)**

Requerimentos não funcionais (RNFs) são as características ou qualidades que descrevem como um sistema deve se comportar, em vez do que ele deve fazer. Eles complementam os requerimentos funcionais, que definem as funcionalidades do sistema.

**Exemplos de RNFs**

* Desempenho: O sistema deve ser rápido e responsivo.
* Segurança: O sistema deve proteger os dados dos usuários.
* Usabilidade: O sistema deve ser fácil de usar e entender.
* Escalabilidade: O sistema deve ser capaz de lidar com um grande número de usuários ou dados.
* Manutenabilidade: O sistema deve ser fácil de manter e atualizar.
* Confiabilidade: O sistema deve ser confiável e funcionar sem erros.

**Importância dos RNFs**

* Garantir a qualidade do sistema: Os RNFs ajudam a garantir que o sistema atenda às expectativas dos usuários e stakeholders.
* Evitar problemas futuros: Os RNFs podem ajudar a evitar problemas futuros, como falhas de segurança ou desempenho.
* Melhorar a comunicação: Os RNFs fornecem uma linguagem comum para a equipe de desenvolvimento e os stakeholders.

**Categorização dos RNFs**

* Requerimentos de produto: descrevem as características do produto, como desempenho, usabilidade, segurança e confiabilidade.
* Requerimentos de projeto: descrevem as restrições do projeto, como orçamento, tempo e recursos.
* Requerimentos de negócio: descrevem as necessidades do negócio, como retorno do investimento e vantagem competitiva.

**Como calcular valor de negócio ao PBI**

Pontuando os PBIs de acordo com aspectos relevantes para o negócio.

Funcionalidade - Descrição da funcionalidade.

Aspectos para o negócio - Lista com aspectos inerentes a funcionalidade.

Peso do Aspecto - Nota dada de 1 a 5 para cada aspecto listado na funcionalidade.

Nota de Funcionalidade - Nota dada de 1 a 10 para o peso de cada aspecto listado referente a funcionalidade.

Valor parcial = peso do aspecto \* nota de funcionalidade

Valor de Negócio = ∑ Valor parcial



Sempre a funcionalidade com maior valor de negócio deve constar no topo da lista para desenvolvimento. Revise e atualize as estimativas de valor de negócio regularmente.

Envolva o cliente e os stakeholders no processo de definição dos critérios de impacto e na atribuição das pontuações.

Utilize outras técnicas de priorização, como a Matriz de Priorização de MoSCoW, para complementar a estimativa de valor de negócio.

**Benefícios**

* Estratégia alinhada com o negócio para organizar o Product Backlog.
* Agregar o máximo de valor de negócio ao produto o quanto antes.
* Envolver o cliente na priorização.
* Ter embasamento para possíveis trocas e prioridades.
* Manter o Product Backlog ordenado e atualizado.

**Casos de Uso x User Stories**

Para descrição das funcionalidade, de uso e regras do que se espera dos PBI’s podem ser utilizados os métodos Casos de Uso e User Stories.

**Casos de uso (use case)**

Descrevem as funcionalidades do sistema de software a partir da perspectiva do usuário. Eles representam as interações entre o usuário e o sistema.

**Desafios**

* Podem ser complexos e difíceis de entender.
  + São mais formais.
* Podem ser difíceis de manter atualizados.
* Podem não ser representativos das necessidades reais dos usuários.

**Características**

* Possuem maior rastreabilidade
* Descrevem o que o sistema faz, não como ele faz.
* São focados nas funcionalidades do sistema.
* São utilizados para documentar os requisitos funcionais do sistema.

**Objetivos**

* Descrever as funcionalidades do sistema de forma clara e concisa.
* Facilitar a comunicação entre os stakeholders.
* Servir como base para o desenvolvimento do sistema.

A UML (Unified Modeling Language) pode ser utilizada na descrição de casos de uso. A UML fornece uma notação gráfica para representar os elementos de um caso de uso, como ator principal, fluxo principal, fluxos alternativos e pós-condições.

**Vantagens de usar UML na descrição de casos de uso**

* Facilita a visualização e compreensão das funcionalidades do sistema.
* Permite a identificação de pontos de falha e oportunidades de melhoria.
* Facilita a comunicação entre os stakeholders.

**Elementos da UML utilizados na descrição de casos de uso**

* **Diagramas de Caso de Uso:** Representam os casos de uso e as interações entre os atores e o sistema.
* **Diagramas de Sequência:** Representam a ordem das mensagens trocadas entre os objetos do sistema durante a execução de um caso de uso.
* **Diagramas de Atividade:** Representam o fluxo de atividades dentro de um caso de uso.

**User Stories**

Descrevem as necessidades do usuário de forma breve e concisa. Elas são escritas na perspectiva do usuário e utilizam linguagem natural.

Descrição contando uma história do que se precisa da funcionalidade execute.

**Desafios**

* Podem ser ambíguas e difíceis de interpretar.
* Podem não conter detalhes suficientes para a implementação.
* Podem ser difíceis de priorizar.

**Características**

* Simples
* Fácil entendimento
* Incentivam conversa
* Descrevem o que o sistema faz, não como ele faz.
* São focados nas funcionalidades do sistema.
* São utilizados para documentar os requisitos funcionais do sistema.

**Objetivos**

* Capturar as necessidades dos usuários de forma clara e concisa.
* Facilitar a comunicação entre a equipe de desenvolvimento e os usuários.
* Servir como base para o planejamento e desenvolvimento do sistema.

**Qual dos dois utilizar?**

Casos de Uso e User Stories são duas técnicas importantes para documentar os requisitos de um sistema de software. A escolha da técnica mais adequada dependerá do contexto do projeto e das necessidades dos stakeholders.

A combinação de Casos de Uso e User Stories pode ser uma estratégia eficaz para documentar os requisitos de um sistema de software. Os Casos de Uso podem ser utilizados para descrever as funcionalidades do sistema de forma geral, enquanto as User Stories podem ser utilizadas para detalhar as necessidades dos usuários.

Porém user stories são melhores para entendimento devido a forma natural e de conto da história.

**Como especificar os itens PBI’s**

* Protótipos navegáveis.
* Criar documento de requisitos com users stories com descrição de como a funcionalidade deve funcionar
* alguns diagramas para descrever por exemplo uma lista de status e as variações

**Como deve ser criado as user stories**

Uma boa User Story deve ser clara, concisa e completa, e deve conter os seguintes pontos primordiais:

**Estrutura**

* Formato: "Como [usuário], eu quero [funcionalidade] para [benefício]."
* Exemplo: "Como cliente, eu quero poder pesquisar produtos por nome para encontrar o que procuro rapidamente."

**Foco no Usuário**

A User Story deve ser escrita na perspectiva do usuário. Use linguagem clara e concisa que o usuário possa entender.

**Valor para o Usuário**

A User Story deve descrever o que o usuário precisa e por que isso é importante para ele. O benefício deve ser claro e tangível para o usuário.

**Independência**

Cada User Story deve ser independente e autocontida. Evite dependências entre as User Stories.

**Estimação**

Forneça uma estimativa de esforço ou tamanho para cada User Story. Isso ajudará a priorizar e planejar o desenvolvimento.

**Pontos Adicionais**

* Critérios de Aceitação: Defina as condições que devem ser atendidas para que a User Story seja considerada concluída.
* Prioridade: Determine a prioridade da User Story em relação às outras User Stories.
* Detalhes: Adicione detalhes adicionais, como requisitos funcionais e não funcionais, se necessário.

**Recomendações**

Inicie com uma frase simples e clara que descreva o que o usuário precisa. Evite jargões técnicos e linguagem complexa. Envolva os usuários na criação das User Stories. Revise e atualize as User Stories regularmente.

**Exemplos de User Stories**

**Exemplo:**

Como cliente, eu quero poder pesquisar produtos por nome para encontrar o que procuro rapidamente.

**Critérios de Aceitação**

O sistema deve permitir que o cliente insira um nome na barra de pesquisa.

O sistema deve exibir uma lista de produtos que atendem ao critério de pesquisa.

A lista de produtos deve ser ordenada por relevância.

**Estimativa**

3 dias

**Prioridade**

Alta

**Detalhes**

O sistema deve ter uma barra de pesquisa na página inicial.

A barra de pesquisa deve-se auto-completar com sugestões de produtos.

A lista de produtos deve mostrar o nome, a imagem e o preço do produto.

**Benefício**

O cliente poderá encontrar o que procura de forma rápida e fácil.

**Qual deve ser o nível de detalhamento do product backlog**

Para os PBI’s que estejam na próxima sprint para execução as user stories devem estar extremamente detalhadas, para melhor entendimento de execução da equipe de desenvolvimento.

As próximas sprints podem ter algumas sprints porém não podem constar muito detalhadas pois pode ser que as demandas das sprints futuras podem sofrem alterações.

**Quando podemos dizer que um PBI está “Ready”?**

Quer dizer que o item da PBI está pronto para ser apresentado para a equipe de desenvolvimento, que a especificação (user stories) está concluída e completa.

**Boa prática do PO no detalhamento**

O PO deve estar 2 sprints à frente do time:

* Enquanto o time trabalha na sprint atual (01), o PO está franularizando os requerimentos da próxima sprint (02)
* E se sobrar tempo, detalhar a outra sprint (03) em requerimentos mais generalista (menos granularidades)

**Backlog Refinement e Planning Poker**

Em um mundo de demandas dinâmicas e prazos desafiadores, o sucesso de projetos ágeis depende de duas práticas fundamentais: Backlog Refinement e Planning Poker. Este white paper irá explorar cada uma dessas práticas, destacando sua importância, principais técnicas e como utilizá-las para otimizar o processo de desenvolvimento de software ágil.

**Backlog Refinement: Destilando a Essência do Produto**

Trata-se de uma ação (não trata de processo formal do scrum) de adicionar detalhes aos PBI’s e obter suas estimativas e ordená-las, são apresentações antecipadas das sprints ao time de desenvolvimento para definição de esforço e melhora e refinamento das histórias dos PBI’s. possibilitando:

* Apresentação antecipada dos PBI’s para o time de desenvolvimento.
* Enriquecimento desses itens com detalhe mencionados pelo time
* Conhecer a estimativa de esforços dos itens
* Re-priorização de Product Backlog para a próxima sprint

O Backlog Refinement é o ato de analisar, priorizar e detalhar os itens do Product Backlog. É um processo contínuo que garante que o backlog esteja sempre organizado, transparente e pronto para o desenvolvimento. Ele abrange atividades como:

* Divisão de Épicos: Decompôr histórias épicas em histórias de usuário menores e mais bem definidas.
* Estimativa de Esforço: Empregar técnicas como Planning Poker para estimar o tempo e esforço necessários para completar cada história.
* Definição de Critérios de Aceitação: Estabelecer critérios claros e mensuráveis para determinar quando uma história está concluída.
* Priorização de Histórias: Classificar as histórias com base em fatores como valor para o negócio, risco e dependências.
* Discussão de Impedimentos: Identificar e solucionar quaisquer obstáculos que possam prejudicar o desenvolvimento.

O PO e a equipe de desenvolvimento participam desta ação.

A atualização do Product Backlog é realizado pelo Product Owner e a estimativa e o esforço para execução da tarefa é dado pela equipe de desenvolvimento.

**Benefícios do Backlog Refinement**

* Melhora a Transparência: Um backlog bem refinado garante que todos os stakeholders estejam cientes das prioridades e do progresso do projeto.
* Aumenta a Previsibilidade: Estimativas precisas facilitam o planejamento de sprints e a entrega de funcionalidades dentro do prazo.
* Reduz o Desperdício: Identificar e remover itens desnecessários ou de baixo valor evita o investimento de tempo e recursos em atividades improdutivas.
* Melhora a Qualidade: Histórias bem definidas e critérios de aceitação claros direcionam o desenvolvimento para a entrega de valor esperado.

Quando a equipe não é envolvida no processo de refinamento ocorre:

* Dimensionamento muito errado.
* Reuniões de planejamento mais trabalhosas.
* PBI’s são arrastados por várias sprints.
* Equipe se sente desmotivada e improdutiva
* Retorno para o cliente fica desgastado

**Planning Poker: Estimação ágil por consenso**

O Planning Poker é uma técnica divertida e colaborativa para estimar o esforço necessário para concluir uma história de usuário. Cada participante recebe cartas com valores de "café" (esforço mínimo) até "Fibonacci" (esforços maiores). O processo funciona da seguinte forma:

* O Product Owner apresenta a história do usuário.
* Cada participante seleciona a carta que representa sua estimativa individual.
* As cartas são reveladas simultaneamente.
* Se todas as cartas forem iguais, a estimativa está aceita. Caso contrário, ocorre uma discussão para que o grupo chegue a um consenso.
* O processo se repete para todas as histórias do sprint.

**Benefícios do Planning Poker**

* Estimativas Precisas: O consenso do grupo tende a gerar estimativas mais realistas do que as realizadas por indivíduos isoladamente.
* Melhora a Comunicação: Estimativas compartilhadas e discutidas em grupo promovem o entendimento mútuo e a transparência.
* Identifica Riscos: O debate durante a estimativa pode revelar possíveis complexidades ou desafios não previstos anteriormente.
* Promove o Team Building: O Planning Poker é uma atividade lúdica que fortalece o trabalho em equipe e a confiança entre os membros do time.

Na estimativa de esforço, quando um dos participantes em uma segunda rodada insistir em uma estimativa mais alta em relação ao restante dos participantes, devemos levar em consideração a estimativa alta para estimar o esforço, este ação irá proteger em um eventual acerto a estimativa alta e caso esteja incorreta fará com que a demanda seja entregue antes da data prevista.

**Combinando Backlog Refinement e Planning Poker: Sinfonia para o Sucesso**

A verdadeira magia acontece quando Backlog Refinement e Planning Poker são utilizadas juntas. O Backlog Refinement detalha as histórias e as prepara para estimativa precisa, enquanto o Planning Poker fornece dados para priorizar o backlog e planejar os sprints de forma eficiente. Essa combinação gera os seguintes resultados:

* Backlog Priorizado: Com estimativas precisas, as histórias de maior valor e com menor esforço são priorizadas para serem desenvolvidas primeiro.
* Sprints Focados: O time de desenvolvimento foca em uma quantidade ideal de histórias por sprint, garantindo a conclusão bem-sucedida dentro do prazo estabelecido.
* Melhora Contínua: O processo contínuo de refinamento e estimativa permite a adaptação aos requisitos dinâmicos e a entrega de valor constante ao cliente.

Backlog Refinement e Planning Poker são práticas indispensáveis para o sucesso de projetos de desenvolvimento ágil. Ao entender sua importância, dominar suas técnicas e combiná-las estrategicamente, equipes ágeis podem transformar o backlog em uma fonte de clareza, prioridade e valor para o cliente, garantindo a entrega de software de alta qualidade de forma consistente.

**Como aplicar o Planning Poker e o ajuste de estimativas**

Durante o Jogo, evitar:

* “Cantar” a carta escolhida. Não falar sobre sua estimativa.
* Mostrar a carta antecipadamente
* Tendenciar valores

Porque não fazer isso:

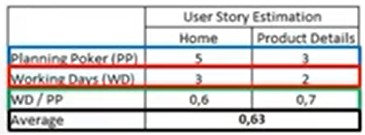
* Desestimula a interação do grupo
* Gera coerção nas estimativas

**Benefícios do Planning Poker**

* Estimativas Precisas: O consenso do grupo tende a gerar estimativas mais realistas do que as realizadas por indivíduos isoladamente.
* Melhora a Comunicação: Estimativas compartilhadas e discutidas em grupo promovem o entendimento mútuo e a transparência.
* Identifica Riscos: O debate durante a estimativa pode revelar possíveis complexidades ou desafios não previstos anteriormente.
* Promove o Team Building: O Planning Poker é uma atividade lúdica que fortalece o trabalho em equipe e a confiança entre os membros do time.
* Amadurecimento dos integrantes.
* Divisão de conhecimento.
* Estimula a troca de informações entre o time.
* Obtenção de detalhes ocultos.

**Alterando a métrica**

* Escolha duas User Stories, as quais você tem certeza de seu esforço relativo.
* Agora estimule-as em dias de trabalho
* Divida a quantidade de dias de trabalho pelo esforço relativo para cada uma. Tire a média dos dois índices.



O ideal é fazer uma nova rodada de estimativa com o time de desenvolvimento para que o time estime com a nova métrica, que no exemplo acima passou de pontos de esforço para dias de trabalho.

**Fazendo ajustes nas estimativas**

Entrosamento x conhecimento técnico x conhecimento do negócio.

Com base nesses pontos existe um planilha que calcula um fator a ser aplicado na estimativa para chegar em valor “seguro”. A média é de 30%.

Quando o projeto está atrasado e todos se mudam para uma sala de reunião para atender a demanda, dá certo porque o time está próximo e está focado na execução do projeto. Quando isso ocorre a redução da estimativa pode ser até de 40%.

**Eventos Scrum**

O Scrum é um framework ágil que se baseia em eventos para promover a colaboração, a transparência e a inspeção e adaptação contínua. Esses eventos fornecem uma estrutura para o desenvolvimento de software, permitindo que as equipes trabalhem de forma eficiente e eficaz.

Os eventos estruturam a organização e o ritmo do trabalho. São momentos distintos, porém interligados, que promovem a colaboração, a transparência e a adaptação contínua.

Os eventos do Scrum:

**Releases**

Uma release representa um conjunto de funcionalidades prontas para serem entregues aos usuários finais. Ela pode ser composta por um ou mais sprints, dependendo da complexidade das funcionalidades e da capacidade da equipe.

**Características da release**

* Foco no valor para o cliente: A release deve entregar funcionalidades que geram valor para o cliente e contribuem para os objetivos do negócio.
* Qualidade: As funcionalidades da release devem ser testadas e validadas antes de serem entregues aos usuários.
* Transparência: Os stakeholders devem estar cientes do conteúdo da release e da data prevista para entrega.

A release sempre deve possuir um objetivo ou propósito, tais como mais valor de mercado, oportunidade de mercado, atender regulamentações, novas funcionalidades, manutenção, correções e etc.

**Estratégias das Release - Tipos de Release**

**Major**

* Grande alterações e funcionalidade acumuladas
* Grande custo de absorção
  + Tendo em vista a grande mudança deve ser levado em consideração o tempo que o usuário levará para aprender a lidar com as alterações.
* Ex.: Windows

**Minor**

* Menos alterações implementadas
* Geralmente mais correções de bugs
* Menor custo de absorção
* Menos valor
* Ex.: Visual Studio

**Functionals**

* Disponibilização ocorrem por funcionalidades
* Das mais relevantes para as menos
* Ocorrem frequentemente
* Menos custo de absorção
* Alta entrega de valor
* Ex.: Google Chrome

**Tipo de disponibilização**

**Alpha**

Alguns usuários da própria empresa.

**Beta**

Usuários mais significativos, pessoas do mercado heavy users.

**Release Candidate**

Potencial versão final, liberado para um grande número de usuários.

**Release to Manufacturing**

Disponibilizado para uma região ou um estado por exemplo, para ter uma distribuição de grande impacto porém ainda com um certo controle de ambiente.

**General Availability**

Distribuição para todo os mercados

**Tipos de Release x Tipos de Disponibilização**

