FACULDADE IMPACTA

CURSOS

SCRUM

ALEX SOUSA

SÃO PAULO - 01/2024

SUMÁRIO

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 é 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 (requerimentos e conhecimento da tecnologia) mais 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.

Existe também o método de desenvolvimento cascata que 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 possui 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.

Scrum Guide

O Scrum Guide é o documento oficial que descreve o framework Scrum. É uma referência essencial para qualquer pessoa que queira entender e implementar o Scrum de forma eficaz.

O Scrum Guide possuí:

- Papéis (3)
- Eventos (5)
- Artefatos (N)
- Regras

Principais pontos do Scrum Guide

- **Papéis:** Define os papéis principais do Scrum: Time de Desenvolvimento, Product Owner e Scrum Master.
- **Eventos:** Descreve os cinco eventos do Scrum: Sprint Planning, Daily Scrum, Sprint Backlog Refinement, Sprint Review e Sprint Retrospective.

- **Artefatos:** Explica os três artefatos do Scrum: Backlog do Produto, Sprint Backlog e Incremento de Produto.
- **Regras:** Descreve as poucas regras do Scrum, como o limite de duração da Sprint e a frequência da Revisão da Sprint.

Importância do Scrum Guide

- **Padronização:** Garante que o Scrum seja implementado de forma consistente por diferentes equipes e organizações.
- **Clareza:** Fornece uma referência clara e concisa para os princípios e práticas do Scrum.
- **Transparência:** Promove a transparência dentro das equipes Scrum.
- **Foco:** Ajuda as equipes a manter o foco no objetivo do Sprint.

O scrum é baseado em processo incremental e iterativo. Sendo um processo *empírico*[1] 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.

---

[1] *Conjunto de conhecimentos adquiridos pela experiência e pela prática. "empirismo", in Dicionário Priberam da Língua Portuguesa [em linha], 2008-2024, https://dicionario.priberam.org/empirismo.*

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 devem 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.

Adicione títulos (Formatar > Estilos de parágrafo) e eles vão aparecer no seu sumário.

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

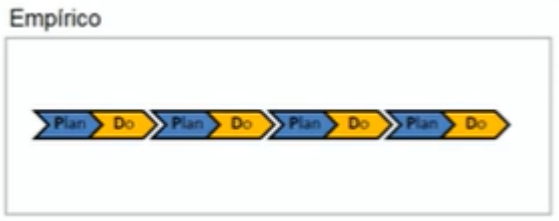
Baseiam-se na estimativa e no planejamento de todo o trabalho a ser feito antes do início do desenvolvimento. Assumem que os requisitos são bem conhecidos e que não mudarão significativamente ao longo do projeto.

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

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

Diferenças entre os processos:

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.

Planning Onion - Planejamento de Vários Níveis

O planejamento de vários níveis, também conhecido como "planning onion", é uma metáfora utilizada para explicar a importância de realizar o planejamento em diferentes níveis de granularidade em projetos complexos. A analogia com uma cebola é utilizada para representar os diferentes níveis de planejamento, como as camadas de uma cebola.

Níveis de planejamento

- Nível estratégico: Define a visão geral do projeto, seus objetivos e metas de alto nível.
- Nível tático: Define os planos de ação para alcançar os objetivos estratégicos, detalhando as atividades, recursos e prazos necessários.
- Nível operacional: Define os detalhes específicos de como as atividades serão executadas, incluindo tarefas, responsabilidades e cronogramas.

Benefícios do planejamento em vários níveis

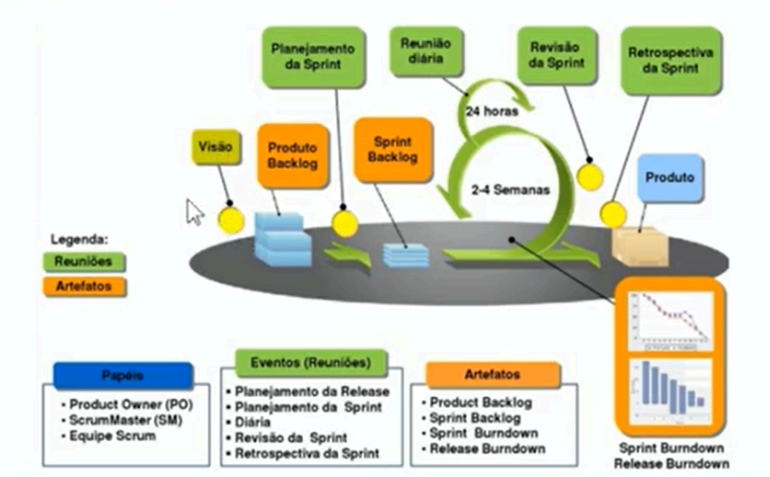
- Maior clareza e foco: Cada nível de planejamento fornece detalhes específicos para um nível de abstração específico, facilitando a compreensão e o foco nas tarefas e responsabilidades.
- Flexibilidade e adaptabilidade: Permite ajustes e adaptações à medida que o projeto avança e novos insights são obtidos.
- Melhor comunicação e alinhamento: Facilita a comunicação e o alinhamento entre os stakeholders em diferentes níveis da organização.
- Redução de riscos: Permite identificar e mitigar riscos com antecedência, através de uma análise mais detalhada das atividades e recursos necessários.

Exemplo de aplicação

Em um projeto de desenvolvimento de software, o planejamento estratégico pode definir a visão geral do produto, seus objetivos e público-alvo. O planejamento tático pode detalhar os recursos e funcionalidades do software, as etapas de desenvolvimento e os prazos de entrega. O planejamento operacional pode definir as tarefas específicas de cada etapa de desenvolvimento, as responsabilidades de cada membro da equipe e o cronograma detalhado.

O planejamento em vários níveis é uma abordagem eficaz para gerenciar projetos complexos, proporcionando maior clareza, flexibilidade, comunicação e alinhamento entre os stakeholders. É importante adaptar a abordagem à natureza específica do projeto e às necessidades da organização.

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
- Gerenciar o backlog do produto

Development Team

- Pessoas que desenvolve o produto
- Entrega de software pronto/Done
- Auto gerência

Scrum Master

- O Scrum Master deve se preocupar em manter o bom relacionamento dentro do time e um bom ambiente de trabalho.
- Mantem todos motivados
- Gerência e estimula o processo do scrum
- Remove impedimentos

Na metodologia Scrum, não existe um papel formalmente chamado de Gerente de Projetos. As responsabilidades tradicionalmente associadas a um Gerente de Projetos são distribuídas entre os outros papéis do Scrum.

1. Scrum Master: O Scrum Master é responsável por garantir que o processo Scrum seja seguido e que a equipe esteja trabalhando de forma eficaz.

2. Product Owner: O Product Owner é responsável por definir a visão e as prioridades do produto e por garantir que o trabalho da equipe esteja alinhado com essas prioridades.

3. Equipe de Desenvolvimento: A equipe de desenvolvimento é responsável por planejar e executar o trabalho da sprint.

Algumas das responsabilidades que um Gerente de Projetos tradicionalmente possui, e que no Scrum são distribuídas entre os outros papéis, incluem:

- Planejamento do projeto: O Product Owner e a equipe de desenvolvimento trabalham juntos para planejar a sprint, definindo as metas e o trabalho a ser realizado.
- Execução do projeto: A equipe de desenvolvimento é responsável por executar o trabalho da sprint.
- Monitoramento e controle do projeto: O Scrum Master e o Product Owner monitoram o progresso da sprint e identificam quaisquer problemas ou impedimentos.
- Fechamento do projeto: O Product Owner e a equipe de desenvolvimento avaliam o resultado da sprint e identificam oportunidades de melhoria.

Em alguns casos, pode haver uma pessoa na equipe que assume um papel de liderança informal e que atua como um "Gerente de Projetos". Essa pessoa pode ser o Scrum Master, o Product Owner ou outro membro da equipe. No entanto, é importante lembrar que essa pessoa não tem um papel formalmente definido no Scrum e que suas responsabilidades podem variar de acordo com a organização.

Scrum Master

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

O Scrum Master deve se preocupar em manter o bom relacionamento dentro do time e um bom ambiente de trabalho.

O Scrum Master é responsável por analisar o desempenho do time ao longo das sprints, a fim de que estejam trabalhando corretamente segundo o Scrum, ele mede o desempenho da equipe de desenvolvimento.

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.

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.

2. 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.

3. 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.

4. 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.

5. 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.

2. 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.

3. 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.

4. Motivação e Engajamento:

Manter a equipe motivada e engajada no projeto.
Criar um ambiente de trabalho positivo e divertido.
Celebrar as conquistas da equipe.

5. 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

2. 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.

3. 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.

4. 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.

5. 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.

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.

A lista de ScrumBut é uma ferramenta utilizada no contexto do desenvolvimento ágil de software, especialmente no framework Scrum. Essa lista destaca práticas ou aspectos que uma equipe deveria seguir de acordo com o Scrum, mas que, por algum motivo, não está seguindo totalmente. O termo "ScrumBut" é uma combinação de "Scrum" e "but" (mas em inglês), indicando que a equipe está seguindo o Scrum, mas com algumas exceções ou compromissos.

O objetivo da lista de ScrumBut é conscientizar a equipe sobre as áreas em que ela está desviando das práticas recomendadas do Scrum. Isso pode acontecer por diversas razões, como restrições de tempo, falta de compreensão total do Scrum, resistência a mudanças, entre outros motivos.

Exemplos comuns de itens em uma lista de ScrumBut podem incluir a falta de reuniões regulares de retrospectiva, a não realização de revisões de sprint ou a falta de colaboração intensiva entre os membros da equipe.

Ao identificar e discutir esses pontos, a equipe pode decidir se deve ajustar suas práticas para se alinhar mais de perto com o Scrum ou se deve adotar formalmente essas variações como parte de seu processo adaptado. O importante é que a equipe esteja ciente das exceções que está fazendo e que isso seja uma escolha consciente.

Uma lista de todos os pontos do Scrum que não são aplicados na empresa e o motivo dessas ações.

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.

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

Funcionalidade	Aspectos para o Negócio	Peso do Aspecto (1 a 5)	Nota da Funcionalidade (1 a 10)	Valor Parcial Peso * Nota	Valor de Negócio Soma V. Parcial
Home - Exibição dos Produtos	Valor percebido para o Consumidor	5	10	50	174
	Comprometimento com o mercado	5	5	25	
	Melhoria de processos internos	5	1	5	
	Risco para a compania	4	8	32	
	Exigência de orgão regulatório	4	1	4	
	Aumento da Receita (ROI)	4	9	36	
	Redução de custos	3	4	12	
	Diferencial para uma prospecção	1	10	10	

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. Abaixo descrição de 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.

O Guia do Scrum não define um tempo máximo para o refinamento do Backlog do Produto. No entanto, ele recomenda que o tempo dedicado ao refinamento do Backlog do Produto não ultrapasse 10% da capacidade da equipe de desenvolvimento.

Isso significa que, se a equipe de desenvolvimento tem uma capacidade de 40 horas por semana, ela deve dedicar no máximo 4 horas por semana ao refinamento do Backlog do Produto.

É importante ressaltar que essa é apenas uma recomendação, e o tempo real necessário para o refinamento do Backlog do Produto pode variar de acordo com a complexidade do projeto, o tamanho da equipe e a experiência da equipe com o Scrum.

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.

	User Story Estimation	
	Home	Product Details
Planning Poker (PP)	5	3
Working Days (WD)	3	2
WD / PP	0,6	0,7
Average	0,63	

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%.

Story Points - Pontos de História (Pontos Relativos de Esforço)

A unidade de medida de esforço que compara itens a serem estimados com referências de esforço existentes dentro do mesmo projeto é frequentemente chamada de "pontos de história" ou "story points". Essa prática é

comumente usada em metodologias ágeis, como o Scrum, para realizar estimativas de esforço para histórias de usuário, tarefas ou outras unidades de trabalho no contexto do desenvolvimento de software.

Os pontos de história não são uma medida de tempo direta, mas sim uma representação relativa do esforço necessário para completar uma tarefa em comparação com outras tarefas no mesmo projeto. Essa abordagem permite que as equipes expressem a complexidade, incerteza e esforço envolvidos em uma determinada unidade de trabalho sem se comprometerem diretamente com prazos específicos.

Ao usar pontos de história, a equipe compara a complexidade da tarefa a ser estimada com tarefas conhecidas, que já foram estimadas anteriormente e cujos pontos de história são conhecidos. Essa técnica é uma maneira de criar uma escala relativa de estimativas que reflete a compreensão da equipe sobre o esforço envolvido.

É importante notar que pontos de história são uma abordagem subjetiva e relativa. Diferentes equipes podem ter escalas de pontos de história diferentes, mas o objetivo é manter a consistência dentro de uma equipe específica para que ela possa usar as estimativas para planejamento e tomada de decisões.

- São unidades abstratas que representam o esforço relativo necessário para completar um item de trabalho.
- Não representam tempo, horas ou dias.
- São utilizados para estimar o tamanho e a complexidade de itens de trabalho, como histórias de usuário, tarefas ou bugs.
- A equipe define uma referência de esforço para um item de trabalho de tamanho conhecido, geralmente chamado de "história base".
- Outros itens de trabalho são comparados à história base e estimados em pontos de história.

Vantagens

- São mais flexíveis do que unidades de tempo, como horas ou dias, pois podem ser ajustados para diferentes contextos e equipes.
- Facilitam a comunicação e o alinhamento entre os membros da equipe, pois todos estão usando a mesma unidade de medida.
- Promovem o foco no valor do trabalho, pois os itens de trabalho são estimados em termos de esforço e não de tempo.

Desvantagens

- Podem ser subjetivas, pois dependem da experiência e do julgamento da equipe.
- Podem ser difíceis de estimar para itens de trabalho complexos ou novos.

Exemplo de aplicação

Em um projeto de desenvolvimento de software, a equipe pode utilizar pontos de história para estimar as histórias de usuário. A equipe define uma história base como um item de trabalho de tamanho médio e estima o esforço necessário para completá-la em 3 pontos de história. Outras histórias de usuário são comparadas à história base e estimadas em pontos de história. Uma história de usuário complexa pode ser estimada em 8 pontos de história, enquanto uma história de usuário simples pode ser estimada em 1 ponto de história.

Os pontos de história são uma unidade de medida de esforço útil para estimar o tamanho e a complexidade de itens de trabalho em projetos ágeis. É importante considerar as vantagens e desvantagens da técnica antes de utilizá-la em um projeto.

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.

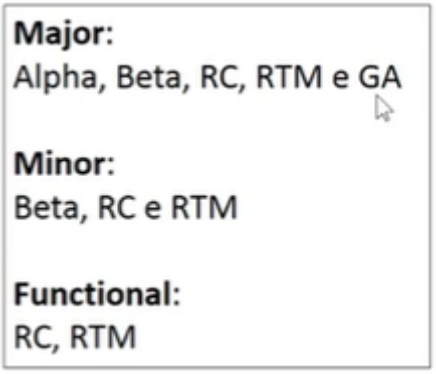
Release to Manufacturing

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



TimeBox

Timebox refere-se a um limite de tempo pré-estabelecido para a realização de uma atividade específica, como um evento do Scrum (Sprint Planning, Daily Scrum, etc.) ou uma tarefa dentro do sprint. A ideia é que ao limitar o tempo, equipes focam na otimização do trabalho e evitam procrastinação ou dispersão de atenção.

Timebox é um período máximo que um evento pode durar, podendo a tarefa ser realizada em um tempo inferior.

A reunião de planejamento pode ser uma exceção no estouro do tempo do timebox.

Características do Timebox

- Fixo e inviolável: O timebox não deve ser alterado uma vez definido. Ele funciona como um compromisso e uma disciplina para a equipe.
- Transparente e acessível: O timebox deve ser conhecido por todos os membros da equipe e facilmente acessível durante o trabalho.
- Adaptado às necessidades: O tempo alocado para cada atividade deve ser adequado à sua complexidade e estimativa de esforço.
- Promove foco e eficiência: Ao limitar o tempo, a equipe é forçada a focar nas tarefas prioritárias e buscar formas de otimizar o trabalho.
- Evita o multitarefas: Com timeboxes curtos, a equipe tende a se concentrar em uma tarefa por vez, evitando a dispersão de atenção e o multitarefas, que geralmente prejudicam a produtividade.

Aplicações do timebox

- Eventos do Scrum: Cada evento do Scrum possui um timebox específico, por exemplo, a Daily Scrum dura 15 minutos, a Sprint Review geralmente dura de 1 a 2 horas.
- Tarefas dentro do sprint: As histórias de usuário dentro do sprint podem ter timeboxes definidos para o desenvolvimento e teste.
- Reuniões e discussões: Mesmo reuniões que não sejam eventos formais do Scrum podem ter timeboxes para evitar prolongamentos desnecessários.

Impacto do timebox

O uso de timeboxes no Scrum traz diversos benefícios, como:

- Melhora o planejamento e a estimativa: As equipes precisam estimar melhor o tempo necessário para as atividades e planejar o sprint de forma realista.
- Aumenta a transparência e a previsibilidade: Com timeboxes claros, o progresso do trabalho fica mais transparente e previsível para todos os envolvidos.
- Reduz o desperdício de tempo: Limitar o tempo força a equipe a evitar atividades e discussões improdutivas, focando no que realmente importa.
- Melhora a comunicação e a colaboração: O foco e a disciplina impostos pelo timebox promovem a comunicação assertiva e a colaboração dentro da equipe.

Lembre-se:

O timebox é uma ferramenta, não uma regra. Ajuste-o às necessidades específicas de sua equipe e do projeto.

A flexibilidade dentro do timebox é possível, mas evite mudanças frequentes ou significativas.

A comunicação aberta e a conscientização da equipe são fundamentais para o sucesso do timebox.

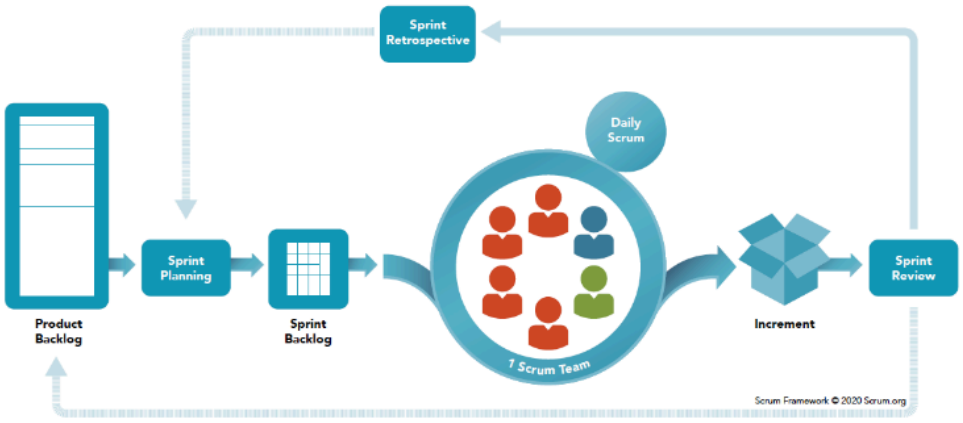
Sprint

Um sprint é um período de tempo fixo, geralmente de 2 a 4 semanas, durante o qual a equipe trabalha para desenvolver um conjunto de funcionalidades.

Também conhecidas como Iteração, as Sprints são o coração do Scrum, onde ideias são transformadas em valor real para os usuários. As sprints sempre devem entregar valor ao produto.

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

As atividades obrigatórias (documentos de arquitetura, configuração de servidor, etc.), mas que não agregam valor, devem ser balanceadas com as de maior ROI (Retorno sobre o Investimento).



Características da sprint

- Planejamento: A equipe planeja o sprint no início, estimando o tempo e esforço necessário para completar as histórias de usuário.
- Foco: A equipe se concentra em um conjunto de tarefas por vez, evitando o multitasking.
- Inspeção e adaptação: A equipe inspeciona o trabalho regularmente e se adapta às mudanças e descobertas.
- Revisão e retrospectiva: Ao final da sprint, a equipe demonstra o trabalho concluído aos stakeholders e reflete sobre o sprint para identificar oportunidades de melhoria.
- 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.
- Uma Sprint pode ser cancelada se a Meta da Sprint se tornar obsoleta. Apenas o Product Owner tem autoridade para cancelar a Sprint.

Importante durante a Sprint

- A equipe permanece com a mesma quantidade de integrantes

- As metas de qualidade não diminuem
- O escopo pode ser detalhado e renegociado entre o PO e a Equipe de Desenvolvimento, porém:
    - **O Objetivo da sprint não pode ser alterado!**
- Podemos adicionar / remover PBI's em uma sprint em execução?
    - Sim podemos, quando o time perceber que orçou errado ou o item cresceu no desenvolvimento pode ser removido um item. A adição de item não possui no scrum guide, porém no dia-a-dia pode ocorrer com a negociação da equipe de desenvolvimento.

Cancelamento de Sprint

A sprint pode ser cancelada. O cancelamento é feito pelo PO se a sprint não possui mais valor ao produto.

O cancelamento de sprint não deve ocorrer com frequência pois não é recomendado. Quando cancelado deve ser realizado reunião com o time para explicação do motivo da recusa.

Pontos Positivos:

- Evita que a equipe trabalho em algo que não terá valor

Pontos Negativos:

- Desmotiva a equipe.
- Sensação de fracasso

A reunião de planejamento

A Reunião de Planejamento (Sprint Planning) do Scrum é um evento do Scrum, onde a equipe se reúne para definir o trabalho que será realizado no próximo sprint.

A Sprint Planning é uma reunião que ajuda a garantir que todos na equipe estejam na mesma página e que o Sprint esteja bem planejado é onde se define o escopo e o plano para o próximo sprint.

É importante que a equipe esteja engajada na seleção das histórias de usuário e na estimativa do esforço. O Sprint Goal deve ser desafiador, mas atingível.

O time irá decidir quais PBI's conseguirão ser transformados em software *done* dentro da sprint.

É neste evento que transforma de Product Backlog em Sprint Backlog.

Duração

O timebox varia de acordo com tamanho da sprint:

Tam. Sprint	Duração Sprint Planning	
	Tópico 1	Tópico 2
4 semanas	8 horas	
3 semanas	6 horas	
2 semanas	4 horas	

Tópico 1 - O que

- PO apresenta cada funcionalidade (PBI) segundo sua priorização, de forma clara e detalhada.
- A equipe de desenvolvimento entende cada item e verifica quantos itens conseguirão ser trabalhados na sprint.
- Time scrum define a meta da sprint (sprint goal). PO explica porquê!

Participa desta etapa a Equipe de Desenvolvimento e PO.

Tópico 2 - Como

- O time verifica como irá transformar os PBI's selecionados em parte de software done.
- Geração de tarefas para cada PBI (quadro kanban ou TFS, etc.)
    - Boa prática: cada tarefa de no máximo 1 dia de trabalho.
- Possível readequação do sprint backlog (para + ou para -)

Participa desta etapa a Equipe de Desenvolvimento, porém PO também é importante!

Atividades

Selecionar as histórias de usuário

O Product Owner apresenta as histórias de usuário priorizadas do backlog do produto. A equipe decide quais histórias serão trabalhadas no próximo sprint, considerando a capacidade da equipe e o tempo disponível.

Estimar o esforço

A equipe estima o tempo e esforço necessário para completar cada história de usuário. Essa estimativa é feita em conjunto, com a participação de todos os membros da equipe.

Definir o Sprint Goal

A equipe define um objetivo claro e conciso para o sprint, que serve como guia para o trabalho. O ***Sprint Goal*** deve ser desafiador, mas atingível dentro do tempo e recursos disponíveis.

> ***Sprint Goal*** é o objetivo principal de um sprint no Scrum, É uma declaração concisa e clara que define o que a equipe se propõe a alcançar durante o sprint.

Criar o plano de sprint

A equipe define as tarefas necessárias para completar cada história de usuário. As tarefas são distribuídas entre os membros da equipe, considerando suas habilidades e experiência.

Revisar o plano

A equipe revisa o plano de sprint para garantir que ele seja claro, completo e compreensível. O Product Owner e stakeholders podem fornecer feedback sobre o plano.

Dicas para uma Reunião de Planejamento

- Inicie com um backlog do produto priorizado: O Product Owner deve priorizar as histórias de usuário no backlog do produto antes da reunião.
- Envolva toda a equipe: Todos os membros da equipe devem participar da reunião e contribuir para o planejamento.
- Faça estimativas realistas: A equipe deve estimar o tempo e esforço com cuidado, evitando subestimar ou superestimar o trabalho.
- Seja flexível: O plano de sprint pode ser ajustado durante o sprint, se necessário.
- Comunique-se com clareza: A equipe deve se comunicar de forma clara e concisa durante a reunião.

Quadro Resumo - Sprint Planning

Questões

Um membro da equipe de desenvolvimento pode se comprometer sozinho com um item durante o planning?
Resposta: a meta é de um time e não pode ser de um único desenvolvedor.

O SM deve participar da reunião de planejamento?
Resposta: O SM não é necessário.

O PO pode tendenciar os dimensionamentos?
Resposta: Não, o dimensionamento do esforço é da equipe de desenvolvimento. O PO pode questionar os motivos daquela estimativa.

Planilha da reunião de planejamento, Documento de requisitos e Documento de gestão de mudanças

Apresentação de algumas planilhas e documentos para gestão e aplicação do método Scrum.

- Ensinando a utilizar planilha de cálculo das horas de trabalhos e planejamento das sprints possíveis da equipe.
- Exemplo de documento de requisito de software.
- Exemplo de documento de gestão de mudança.
- Planilha para planos de testes

Dicas para a reunião de planejamento (Sprint Planning)

Abaixo algumas dicas para que a sprint planning seja realizada da melhor forma possível.

- Implementar um gap de descanso (almoço) entre cada parte da reunião.
- Realizar Planning sempre no começo da semana (segunda / terça).
- Cobrar a participação ativa de todo o time durante a reunião. Afinal, o comprometimento com a meta é de todo o time.
- Faça a equipe detalhar em conjunto cada PBI, discutindo o esforço de cada tarefa.
- As tarefas devem ser escritas com as palavras do time.
- Aproveita e já armazene cada estimativa dada.
- O PO não pode interferir na ordem de execução das tarefas da equipe de desenvolvimento.
- A equipe de desenvolvimento deve ter e manter atualizado um plano de como alcançar a meta da sprint.
- Status das tarefas devem ser atualizados diariamente
- Caso a equipe perceba que estimou errado os PBI's da Sprint, o PO deve ser avisado o quanto antes.

A reunião diária - Daily Scrum

O Daily Scrum é uma reunião diária realizada por equipes Scrum para sincronizar o progresso do trabalho, identificar impedimentos e planejar o próximo dia. É uma reunião curta, timebox no máximo 15 minutos, que deve ser realizada em pé para manter o foco e a energia.

Objetivos

- Durante a reunião diária, a equipe discute o progresso das histórias de usuário em andamento.
- Se a equipe identifica que o tempo ou esforço estimado para completar uma história está subestimado, a estimativa pode ser ajustada.
- O novo valor da estimativa deve ser acordado por toda a equipe.
- Compartilhar progresso: Cada membro da equipe relata o que foi feito no dia anterior, o que será feito hoje e quais são os impedimentos que estão enfrentando.
- Buscar soluções: A equipe trabalha em conjunto para encontrar soluções para os impedimentos.
- Manter o foco: O Daily Scrum ajuda a manter a equipe focada no Sprint Goal.
- Mais uma oportunidade de inspeção de progresso.
- Bom momento para mover as tarefas do kanban.

Como realizar os eventos do Scrum

- Timebox: 15 minutos.
- Sempre no mesmo local e horário.
- Participantes: equipe de desenvolvimento.
    - PO somente como ouvinte.
    - Scrum Master não deve participar, deve orientar e garantir que as regras do scrum estejam sendo cumpridas.
- Atividades: sincronizar o trabalho, identificar impedimentos, buscar soluções.
- Cada membro da equipe responde a 3 perguntas:
    - O que foi feito ontem?
        - Cada membro da equipe responde a essa pergunta, resumindo o que foi feito no dia anterior.
    - O que será feito hoje?
        - Cada membro da equipe responde a essa pergunta, descrevendo o que planeja fazer no dia atual.
    - Quais são os impedimentos?

        - Cada membro da equipe informa se há algum impedimento que esteja atrapalhando o seu trabalho.
- A equipe busca soluções para os impedimentos.

Dicas para a reunião diária Daily scrum

Dicas para um Daily Scrum eficaz:

- Seja breve e conciso. A reunião deve durar no máximo 15 minutos.
- Mantenha o foco. A reunião deve se concentrar nos três tópicos principais: progresso, impedimentos e planejamento.
- Seja honesto e transparente. Todos os membros da equipe devem ser honestos sobre o seu progresso e os impedimentos que estão enfrentando.
- Resolva impedimentos rapidamente. Se um impedimento for identificado, a equipe deve tentar resolvê-lo o mais rápido possível.
- Faça a reunião em pé. Isso ajudará a manter o foco e a energia.
- Fazer diante do quadro kanban
- o time deve reportar para ele mesmo.
- Todos da equipe devem participar principalmente os SR's
- Todos devem participar
- Sem celular
- Questionar o time sobre o sentimento para alcançar a meta.
- Fazer a reunião no começo do dia.

Revisão da sprint

A Revisão da Sprint é um evento do Scrum que ocorre no final de cada Sprint. É uma oportunidade para a equipe de desenvolvimento demonstrar o trabalho concluído durante a Sprint e obter feedback das partes interessadas, inclusive o cliente podendo ser convidado para participação.

A reunião de Revisão deve ter um Timebox máximo de 4 horas para uma Sprint de 4 semanas ou 1 mês ou 2 horas para uma Sprint de 2 semanas

Tratasse de uma reunião de trabalho onde o PO análise e inspecione o trabalho desenvolvido e apresentado. Com isso o PO deve atualizar, analisar, reordenar e se necessário repriorizar o backlog do produto.

Se um time não está batendo a meta, o SM pode alterar a configuração do time.

Objetivos da Revisão da Sprint

- Obter feedback sobre o trabalho em andamento: As partes interessadas fornecem feedback sobre o trabalho que foi demonstrado, o que pode ajudar a equipe a melhorar o produto.
- A equipe de desenvolvimento responde a questionamentos sobre o incremento do produto..
- Inspecionar e adaptar o backlog do produto: O Product Owner pode inspecionar o backlog do produto com base no feedback recebido e fazer alterações se necessário.
- Celebrar o sucesso: A equipe comemora o trabalho que foi concluído durante a Sprint.
- PO avalia e identifica o que está realmente Done.

Participantes da Revisão da Sprint

- Equipe de Desenvolvimento: Todos os membros da equipe de desenvolvimento devem participar da Revisão da Sprint.
- Scrum Master: O Scrum Master facilita a reunião e garante que ela siga as regras do Scrum.
- Product Owner: O Product Owner é responsável por apresentar o trabalho da equipe às partes interessadas.
- Partes interessadas: Quaisquer pessoas que tenham interesse no produto podem participar da Revisão da Sprint, como clientes, usuários finais, gerentes e outros stakeholders.

Formato da Revisão da Sprint

- Demonstração do trabalho: A equipe de desenvolvimento demonstra o trabalho que foi concluído durante a Sprint.
- Coleta de feedback: As partes interessadas fornecem feedback sobre o trabalho que foi demonstrado.
- Discussão e planejamento: A equipe discute o feedback recebido e planeja as próximas etapas do projeto.

Dicas para a revisão da sprint

- **Convide as partes interessadas certas.** É importante convidar as partes interessadas certas para a Revisão da Sprint.
- **Prepare-se para a demonstração**. A equipe de desenvolvimento deve se preparar para a demonstração para garantir que ela seja clara e concisa.
- **Incentive cada desenvolvedor** a apresentar o que ele mesmo desenvolveu.
- **Incentive o feedback**. As partes interessadas devem ser incentivadas a fornecer feedback honesto e construtivo.
- **Faça a reunião interativa**. A Revisão da Sprint deve ser uma reunião interativa, onde todos os participantes possam se envolver.
- **Acompanhe o feedback.** A equipe deve acompanhar o feedback recebido e tomar medidas sobre ele.
- Estimule o PO a dar a opnião a cada item apresentado.
- Permita o PO trabalhar algumas horas entre a revisão e a próxima planning.
- PO iniciar reunião de revisão questionando se a meta foi alcançada.

Débito técnico

O débito técnico se refere ao acúmulo de soluções não ideais adotadas durante o desenvolvimento de um projeto. É como se fosse uma "dívida" que a equipe contrai com o código, precisando ser paga com refatoração ou retrabalho no futuro.

A cada sprint é interessante que o PO defina um período para “pagamento” de Undone Work, para que no futuro os débitos técnicos no futuro não sejam de um tamanho grande.

O que pode ser considerado como um débito técnico

- Defeitos
- Falta de testes de aceitação
- Código duplicado
- Regra de negócio em lugares errados

- Impossibilidades de voltar para a versão anterior
- Algoritmos de difícil manutenção

Causas do Débito Técnico

- Pressão por prazos apertados: A necessidade de entregar funcionalidades rapidamente pode levar a soluções rápidas e não ideais.
- Falta de conhecimento: A equipe pode não ter o conhecimento técnico necessário para implementar a solução ideal.
- Escopo do projeto em constante mudança: Mudanças frequentes nos requisitos do projeto podem levar a soluções que não são mais adequadas.

Consequências do Débito Técnico

- Dificuldade em manter o código: O código com débito técnico é mais difícil de entender e modificar, o que pode aumentar o tempo e o custo de desenvolvimento.
- Bugs e problemas de qualidade: Soluções não ideais podem levar a bugs e problemas de qualidade no produto final.
- Desmotivação da equipe: Trabalhar com código com débito técnico pode ser frustrante e desmotivador para a equipe.

Como Gerenciar o Débito Técnico

- Conscientização: É importante que a equipe esteja ciente do que é débito técnico e quais são seus impactos.
- Priorização: A equipe deve priorizar a refatoração do código com débito técnico mais crítico.
- Comunicação: O Product Owner e stakeholders devem ser informados sobre o débito técnico e seus impactos no projeto.
- Planejamento: O débito técnico deve ser considerado no planejamento do projeto, reservando tempo e recursos para refatoração.

Dicas para Evitar o Débito Técnico

- Investir em testes: Testes automatizados podem ajudar a identificar bugs e problemas de qualidade antes que eles sejam entregues ao cliente.
- Refatorar regularmente: É importante reservar tempo para refatorar o código regularmente, evitando o acúmulo de débito técnico.
- Adotar boas práticas de desenvolvimento: Seguir boas práticas de desenvolvimento, como SOLID e Clean Code, pode ajudar a reduzir o débito técnico.

Lembre-se, o débito técnico é inevitável em qualquer projeto de software. O importante é gerenciar o débito técnico de forma eficaz para minimizar seus impactos.

Existem diversas técnicas e ferramentas que podem ajudar a gerenciar o débito técnico.

Defeito Escapado

Um defeito escapado é um defeito que não foi detectado durante o processo de desenvolvimento e teste e que só é descoberto após o produto ser lançado para o cliente.

Existem diversas causas para defeitos escapados, entre elas:

- Falta de testes: O produto não foi suficientemente testado, o que permitiu que o defeito passasse despercebido.
- Testes inadequados: Os testes realizados não foram adequados para detectar o defeito.
- Erro humano: Um membro da equipe de desenvolvimento ou teste cometeu um erro que permitiu que o defeito passasse despercebido.

Os defeitos escapados podem ter diversos impactos negativos, como:

- Insatisfação do cliente: O cliente fica insatisfeito com o produto porque ele apresenta defeitos.
- Danos à reputação da empresa: A empresa pode ter sua reputação danificada se o produto apresentar muitos defeitos.
- Custos adicionais: A empresa pode ter que arcar com custos adicionais para corrigir os defeitos e compensar os clientes afetados.

Para evitar defeitos escapados, é importante:

- Realizar testes rigorosos: O produto deve ser testado de forma abrangente para garantir que todos os defeitos sejam detectados.
- Utilizar técnicas de teste adequadas: As técnicas de teste utilizadas devem ser adequadas para o tipo de produto que está sendo desenvolvido.
- Investir em treinamento: Os membros da equipe de desenvolvimento e teste devem ser treinados para evitar erros que possam levar a defeitos escapados.

Aqui estão algumas dicas para lidar com defeitos escapados:

- Corrija o defeito o mais rápido possível.
- Comunique-se com o cliente sobre o defeito e as medidas que estão sendo tomadas para corrigi-lo.
- Compense o cliente pelos inconvenientes causados pelo defeito.

Reunião de Retrospectiva da Sprint - Sprint Retrospective

A metodologia ágil nos ensinou a abraçar a mudança e a iterar constantemente. Em um cenário de transformação dinâmica, é fundamental refletir sobre o que está funcionando e identificar áreas de melhoria. É aqui que a Sprint Retrospective brilha - uma cerimônia crucial no Scrum que permite às equipes aprender com suas experiências e evoluir a cada ciclo de desenvolvimento.

Na reunião de retrospectiva do sprint, a equipe reflete sobre o sprint que foi concluído, buscando identificar oportunidades de melhoria para o próximo sprint.

Inspeção do time pelo próprio time, a equipe pode discutir como as estimativas foram realizadas e como melhorar o processo de estimativa em sprints futuros.

É o último evento realizado dentro de uma sprint, o ideal é montar uma lista com os itens abordados.

É um evento realizado ao final de cada Sprint, onde a equipe (time) se reúne para:

- **Revisar o trabalho realizado:** O que deu certo? O que encontramos de dificuldade?
- **Analisar o Sprint Goal:** Atingimos o objetivo? O quão eficazmente utilizamos nosso tempo e recursos?
- **Identificar impedimentos:** O que atrapalhou o progresso? Como podemos evitar esses obstáculos no futuro?
- **Definir ações para o próximo Sprint:** O que podemos fazer para melhorar o processo, a colaboração e a qualidade do produto?

Como executar sprint retrospective

Duração: 3 horas para uma sprint de 4 semanas.
Participantes: equipe de desenvolvimento (todo o time scrum (PO, SC, Time de Desenvolvimento).
Atividades: refletir sobre o sprint, identificar oportunidades de melhoria.

Por que a Sprint Retrospective é importante

- **Melhoria contínua:** É uma oportunidade para aprender com os erros e acertos, implementando ajustes que elevam a performance da equipe.
- **Transparência e comunicação:** Promove a discussão aberta e honesta sobre os desafios enfrentados, fortalecendo a confiança e o senso de responsabilidade. É importante saber ouvir.
- **Engajamento da equipe:** Permite que todos os membros participem ativamente na busca de soluções, aumentando a motivação e o ownership do trabalho.
- **Adaptação às mudanças:** Em um ambiente ágil, é crucial ajustar o processo conforme as circunstâncias e necessidades evoluem. A Sprint Retrospective oferece o espaço para essa adaptação.
- **Maturidade** para entender os pontos negativos levantados na reunião, não levar para o pessoal.

Como conduzir uma Sprint Retrospective eficaz

- **Criar um ambiente seguro e acolhedor:** É essencial que todos se sintam à vontade para compartilhar suas opiniões e experiências, sem medo de julgamentos.
- **Escolher um formato divertido e dinâmico:** Existem diversos formatos de Retrospective, como o "Start-Stop-Continue", o "Cinco Porquês" ou o "Jogo da Bola Murmurada".
- **Focar em ações acionáveis:** Não basta identificar problemas. A Retrospective deve gerar ações concretas para serem implementadas no próximo Sprint.
- **Monitorar o progresso das ações:** Acompanhe a implementação das ações definidas e avalie sua eficácia na próxima Retrospective.

A Sprint Retrospective não é apenas uma cerimônia. É uma cultura, uma mentalidade de contínuo aprendizado e busca de melhoria. Ao abraçá-la, sua equipe terá todas as armas para trilhar o caminho da excelência ágil.

Após a identificação dos pontos, o plano de melhorias definido na sprint retrospectiva deve ser executado na próxima sprint.

Dicas para a reunião de retrospectiva da sprint

Abaixo lista com alguns pontos para realização de uma reunião de retrospectiva melhor.

- Solicitar que cada membro individualmente coloque os itens de discussão em post-its, todos membros realizarem lista de pontos de melhorias.
- Discutir em conjunto cada um desses itens.

- Estimular o próprio time a fornecer ações para correções dos pontos negativos.
- Estabelecer votação para as alternativas de ação.
- Começar e finalizar um assunto por vez.
- Deixar os próprios membros da equipe tirarem as conclusões.
- Discutir formas de evitar reincidências já mencionadas em retrospectivas anteriores.
- Criar na reunião uma lista dos pontos citados e de suas estratégias para mitigar o erro. Evitando assim reincidência.

Artefatos do Scrum

No Scrum, os artefatos servem como ferramentas essenciais para garantir a transparência, o foco e o progresso do projeto. São elementos visíveis e compartilhados que permitem que todos os envolvidos acompanhem o trabalho realizado, identifiquem problemas e oportunidades, e tomem decisões informadas.

O SC deve estimular a transparência dos artefatos.

Sprint Backlog

- É um subconjunto do Product Backlog que define o trabalho a ser realizado durante uma Sprint.
- É criado durante a Sprint Planning e contém as tarefas necessárias para completar as funcionalidades selecionadas.
- O Sprint Backlog é de propriedade da equipe de desenvolvimento e deve ser atualizado diariamente.

Product Backlog

- É uma lista priorizada de funcionalidades que representam o trabalho a ser realizado no produto.
- O Product Owner é o responsável por manter o Product Backlog atualizado e organizado.
- As funcionalidades são priorizadas de acordo com o valor que agregam ao cliente e à empresa.

Incremento de Produto

- É o resultado do trabalho realizado durante uma Sprint.
- É um produto funcional que pode ser testado e entregue ao cliente.
- O incremento de produto deve ser "Done", ou seja, atender a todos os critérios de definição de "pronto" da equipe.

Radiador de Informação

Um radiador de informação é um dispositivo ou ferramenta que serve para exibir informações de forma clara e concisa, de modo que qualquer pessoa possa visualizá-las e compreendê-las facilmente.

No contexto do Scrum, os radiadores de informação são usados para comunicar o status do projeto e as principais informações sobre o andamento das sprints.

Alguns exemplos de radiadores de informação no Scrum:

- Quadro Kanban: Um quadro que mostra o fluxo de trabalho das histórias de usuário, desde a sua criação até a sua entrega.

- Gráfico de Burndown: Um gráfico que mostra a quantidade de trabalho restante a ser feito em cada sprint.
- Lista de impedimentos: Uma lista que registra os problemas que estão impedindo o progresso da equipe.
- Painel de indicadores: Um painel que apresenta as principais métricas do projeto, como velocidade da equipe, taxa de defeitos e qualidade do código.

Os radiadores de informação são importantes porque:

- Promovem a transparência: Todos os membros da equipe e stakeholders podem visualizar o status do projeto e as principais informações sobre o andamento das sprints.
- Facilitam a comunicação: Os radiadores de informação servem como um ponto de referência para a comunicação entre os membros da equipe e stakeholders.
- Ajudam a identificar problemas: Os radiadores de informação podem ajudar a identificar problemas e impedimentos que estão afetando o andamento do projeto.

Ao escolher um radiador de informação, é importante considerar:

- O tipo de informação que será exibida: O radiador de informação deve ser capaz de exibir as informações de forma clara e concisa.
- O público-alvo: O radiador de informação deve ser adequado para o público-alvo que o utilizará.
- O local onde será instalado: O radiador de informação deve ser instalado em um local de fácil acesso e visibilidade.

Burndown chart

É uma ferramenta visual para monitorar o progresso de uma equipe durante uma Sprint. Ele funciona como um termômetro do trabalho restante e pode ajudar a identificar se a equipe está no caminho certo para atingir o objetivo da Sprint.

Como funciona o burndown chart

- Eixo vertical: Representa o trabalho restante (estimativa em pontos de história, horas-pessoa, etc.).
- Eixo horizontal: Representa o tempo restante na Sprint (dias, horas, etc.).

- Linha de ideal burndown: Uma linha reta que mostra o ritmo ideal de trabalho para completar o backlog da Sprint dentro do prazo.
- Linha de burndown real: Uma linha que mostra o progresso real da equipe, indicando quanto trabalho já foi realizado a cada dia.

Importância dos Artefatos do Scrum

- Transparência: Fornecem visibilidade do trabalho em andamento, facilitando a comunicação e o acompanhamento do progresso.
- Foco: Ajudam a equipe a manter o foco no objetivo da Sprint e nas prioridades do cliente.
- Progresso: Permitem a inspeção e adaptação do processo, garantindo que o produto esteja evoluindo na direção correta.

Dicas para gerenciar os Artefatos do Scrum

- Mantenha os artefatos atualizados: É fundamental que as informações contidas nos artefatos estejam sempre atualizadas para garantir a precisão e a confiabilidade das decisões.
- Faça os artefatos visíveis: Todos os envolvidos no projeto devem ter acesso aos artefatos e poder acompanhá-los facilmente.
- Utilize os artefatos para tomar decisões: Os artefatos fornecem informações valiosas para que a equipe tome decisões informadas sobre o projeto.

Os artefatos do Scrum são elementos fundamentais para o sucesso da metodologia. Ao compreendê-los e utilizá-los de forma eficaz, as equipes podem garantir a transparência, o foco e o progresso do projeto, aumentando as chances de entregar um produto de qualidade que atenda às expectativas do cliente.

Ferramentas do Scrum

Abaixo lista com algumas ferramentas para suporte na aplicação da metodologia Scrum.

- Microsoft Visual Studio (ALM)
- Kanbanize e ScrumMe (Kanban online)
- Sketchflow (prototipação)
- Axure (prototipação)
- Resharper (qualidade de código)

Manifesto para o Desenvolvimento Ágil de Software

O Manifesto para o Desenvolvimento Ágil de Software é um documento criado em 2001 por um grupo de 17 desenvolvedores de software que se reuniram em Snowbird, Utah, nos Estados Unidos. O objetivo do manifesto era definir um conjunto de princípios e valores para o desenvolvimento de software que fosse mais flexível, adaptável e responsivo às mudanças do que as metodologias tradicionais.

O manifesto é composto por quatro valores principais:

1. Indivíduos e interações mais que processos e ferramentas: O manifesto enfatiza a importância da colaboração e da comunicação entre os membros da equipe de desenvolvimento.

2. Software em funcionamento mais que documentação abrangente: O manifesto valoriza a entrega de software funcional em detrimento da criação de documentação extensa.

3. Colaboração com o cliente mais que negociação de contratos: O manifesto defende a importância da colaboração entre o cliente e a equipe de desenvolvimento para garantir que o software atenda às suas necessidades.

4. Responder a mudanças mais que seguir um plano: O manifesto reconhece que as mudanças são inevitáveis e que o processo de desenvolvimento de software deve ser adaptável para lidar com elas.

Além dos quatro valores principais, o manifesto também inclui 12 princípios que fornecem orientação para a aplicação desses valores.

O Manifesto para o Desenvolvimento Ágil de Software teve um impacto significativo na indústria de software. Ele inspirou o desenvolvimento de diversas metodologias ágeis, como Scrum, Kanban e Lean, que são amplamente utilizadas atualmente.

Os Doze Princípios do Manifesto Ágil

Os Doze Princípios do Manifesto Ágil servem como um guia para o desenvolvimento de software ágil. Eles enfatizam a importância da colaboração, da entrega contínua de valor, da adaptação à mudança e da busca pela excelência.

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

2. Aceitar mudanças de requisitos, mesmo no fim do desenvolvimento. Processos ágeis tiram proveito da mudança para a vantagem competitiva do cliente.

3. Entregue software funcional com frequência, de poucas semanas a alguns meses, com preferência à menor escala de tempo.

4. Negócios e desenvolvedores devem trabalhar juntos diariamente ao longo do projeto.

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

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

7. Software funcionando é a medida primária do progresso.

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

9. A atenção contínua à excelência técnica e ao bom design aumenta a agilidade.

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

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

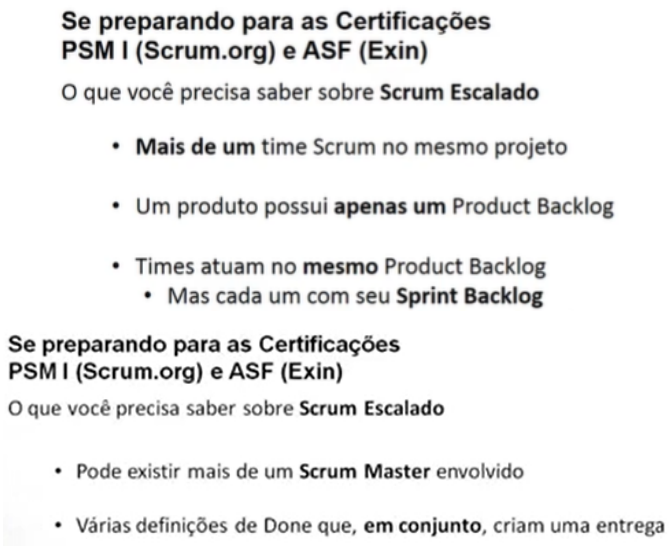
12. Em intervalos regulares, a equipe reflete sobre como se tornar mais eficaz e então ajusta seu comportamento de acordo.

Scrum Escalado

O Scrum escalado é uma abordagem para aplicar os princípios e práticas do Scrum a projetos grandes e complexos que envolvem várias equipes. O objetivo do Scrum escalado é manter a agilidade e a flexibilidade do Scrum, mesmo em projetos de grande escala.

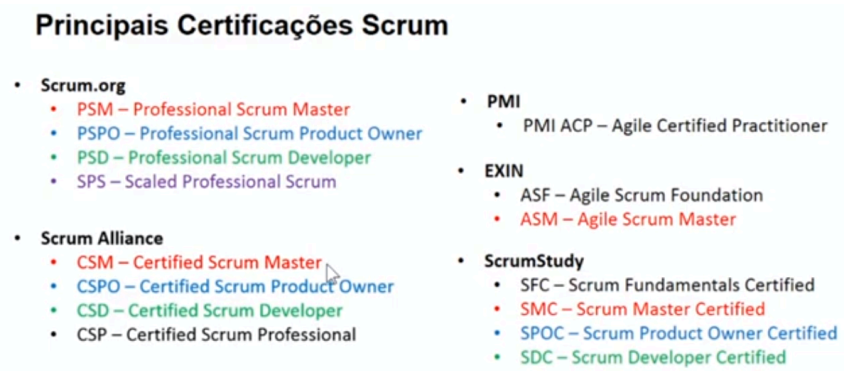
Ao iniciar um projeto com Scrum Escalado a melhor forma de agrupar/montar os times é por funcionalidade a ser desenvolvida.

Desta forma, podemos ter o máximo de velocidade, pois cada time será responsável por algumas funcionalidades em sua sprint. Quanto menos dependência existir entre os times Scrum, melhor será o resultado das sprints dos times no Scrum escalado.



As certificações Scrum: Qual escolher?

Quais principais instituições e tipos de certificações disponíveis para profissionais Scrum?



Qual certificação realizar primeiro?

PSM I (Scrum.org) x ASF (Exin)

	PSM I – Professional Scrum Master	ASF – Agile Scrum Foundation
Instituição	Scrum.org	Exin
Quantidade de Questões	80	40
Tempo para Execução	60 minutos	60 minutos
Nota Mínima	85% (68 de 80)	65% (26 de 40)
Modo	Online (Site)	Online (Exin Anywhere)
Treinamento Presencial?	Não	Não
Revalidação?	Não	Não
Idioma	Inglês	Inglês/Português
Preço	$ 150	$ 168
Experiência Agile/Scrum	Intermediário	Iniciante

Como se preparar para as certificações PSM I e ASF

Dicas para realização das provas para certificação de Scrum.

Se preparando para as Certificações
PSM I (Scrum.org) e ASF (Exin)

- **Este treinamento**
  - Conhecimento sobre todo o processo
  - Questões situacionais do dia-a-dia
  - Cuidados nas provas e dicas

- Ler o Guia Scrum (Scrum Guide) em **português** e **inglês**
- Para a ASF, ler **também** o Guia *"EXIN Agile Scrum Foundation Workbook"*
- Resolver e entender as **questões fornecidas** neste treinamento
- Fazer o *Open Assessment* fornecido pela própria Scrum.org
  - Mesma **dinâmica** da prova real

Scrum Open é um simulado para teste de conhecimento.

Se preparando para as Certificações
PSM I (Scrum.org) e ASF (Exin)

- Questões de **múltipla escolha**
  - Apenas **uma** alternativa correta
  - **Mais de uma** alternativa correta
  - Questões com **Verdadeiro** ou **Falso**

- **Não** existem questões **dissertativas**

- Questões **Teóricas** e **Situacionais. Exemplos:**

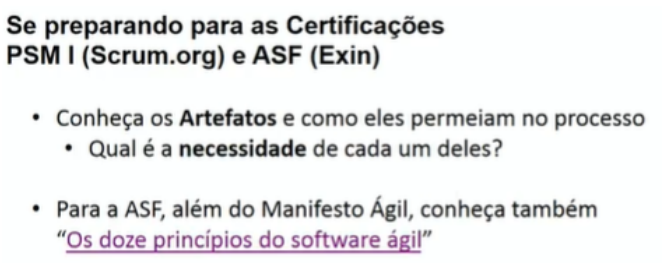
  - *Qual é o **Timebox** da Reunião de Planejamento de uma Sprint de 4 semanas?*
  - *Meu Time de Desenvolvimento se **recusa** a fazer a reunião diária. Qual pode ser o impacto disso na Sprint?*

Se preparando para as Certificações
PSM I (Scrum.org) e ASF (Exin)

- Saiba explicar em **detalhes** cada um dos **eventos**
  - Objetivos, entradas, saídas, participantes, como ocorre e timebox

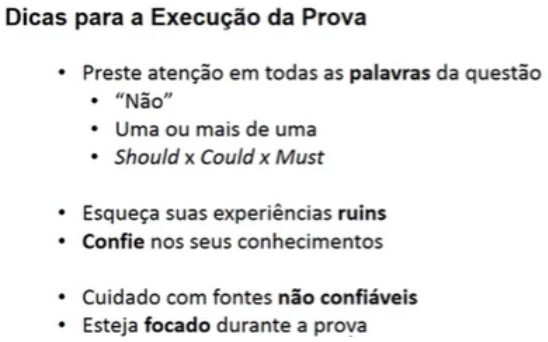
- Saiba o **impacto** de não executar este evento dentro do Scrum

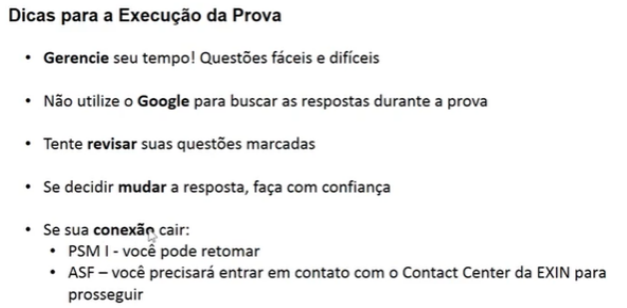
- Saiba os **objetivos** e desafios de cada **membro** do time Scrum e suas interações
  - E também o que cada um não pode fazer!



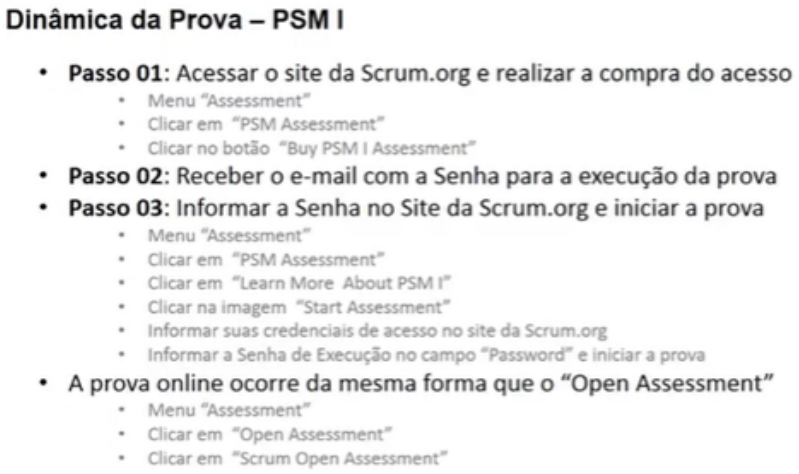
Dicas de como executar as certificações PSM I e ASF

Dicas para execução das provas para certificação de Scrum.





Entendendo a dinâmica das provas PSM I e ASF



Dinâmica da Prova – ASF

- **Passo 01**: Acessar o site da EXIN e realizar a compra do acesso à prova

- **Passo 02**: Receber o e-mail com o Código de Acesso para a execução da prova
  - O tempo de expiração do código é de 3 semanas

- **Passo 03**: Instalar o Programa "Exin Anywhere" em sua Máquina e testar seu ambiente
  - WebCam
  - Microfone

Dinâmica da Prova – ASF

- **Passo 04**: Iniciar a Prova através do Programa "Exin Anywhere"
  - Ter um documento oficial com uma foto sua (RG/Passaporte/CNH)
  - Filmar o ambiente onde você está fazendo a prova
  - Estar sozinho em uma sala
  - Sem outros dispositivos eletrônicos ou livros
  - Não usar fones de ouvido