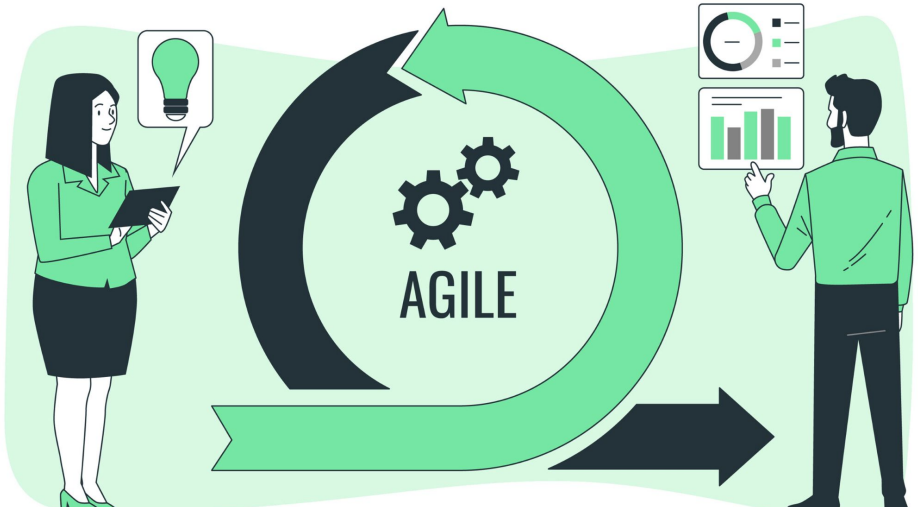
Metodologias Ágeis e Scrum

O que são Metodologias Ágeis?



O que são Metodologias Ágeis?

Metodologias ágeis são um conjunto de práticas e princípios que visam otimizar o desenvolvimento de projetos, tornando-os mais rápidos, flexíveis e adaptáveis.

Diferente dos métodos tradicionais, que seguem um plano rígido do início ao fim, as metodologias ágeis incentivam a entrega incremental e iterativa, com feedback contínuo e adaptação às mudanças.

Características

Características

Foco no cliente: As metodologias ágeis valorizam a colaboração com o cliente e a entrega de valor para ele em cada etapa do projeto.

Entrega contínua: Os projetos são divididos em ciclos curtos, chamados sprints, com entregas parciais e frequentes de funcionalidades.

Características

Adaptação: Às metodologias ágeis incentivam a adaptação a mudanças e a novos requisitos durante o desenvolvimento.

Colaboração: A comunicação e a colaboração entre os membros da equipe são essenciais para o sucesso do projeto.

Características

Transparência: As informações sobre o projeto são compartilhadas com todos os membros da equipe e com o cliente, promovendo a transparência e a confiança.

Benefícios

Maior velocidade e eficiência: As metodologias ágeis permitem entregas mais rápidas e eficientes, reduzindo o tempo de desenvolvimento e o tempo de lançamento do produto.

Maior flexibilidade e adaptabilidade: A capacidade de adaptação às mudanças e aos novos requisitos permite que os projetos se ajustem às necessidades do mercado e do cliente.

Benefícios

Redução de riscos: A entrega incremental e o feedback contínuo ajudam a identificar e corrigir problemas rapidamente, reduzindo o risco de falhas no projeto.

Maior satisfação do cliente: A colaboração com o cliente e a entrega de valor em cada etapa do projeto aumentam a satisfação e a fidelidade do cliente.

Benefícios

Melhoria contínua: As metodologias ágeis incentivam a melhoria contínua dos processos e do produto, resultando em um produto de maior qualidade.

Manifesto Ágil

Manifesto Ágil

O Manifesto Ágil é um documento publicado em fevereiro de 2001, que define os valores e princípios fundamentais para o desenvolvimento ágil de software.

Ele surgiu como uma alternativa aos métodos tradicionais, considerados pesados e burocráticos, buscando uma abordagem mais flexível e colaborativa.

Manifesto Ágil

Princípios: O manifesto também estabelece 12 princípios que orientam a prática ágil, como a entrega contínua de valor, a adaptação às mudanças e a colaboração próxima com o cliente.

Origem: O manifesto foi criado por um grupo de 17 desenvolvedores de software que se reuniram em Snowbird, Utah, em 2001, para discutir novas formas de trabalhar.

Manifesto Ágil

Impacto: O Manifesto Ágil teve um impacto significativo na indústria de software e foi adotado por diversas empresas, como Uber, iFood, Netflix e Airbnb.

Flexibilidade: Uma das principais características do desenvolvimento ágil é a capacidade de se adaptar às mudanças e aos novos requisitos, ao contrário dos métodos tradicionais que costumam ser mais rígidos.

Manifesto Ágil

O manifesto define quatro valores principais:

1. **Indivíduos e interações.**
2. **Software em funcionamento.**
3. **Colaboração com o cliente.**
4. **Responder a mudanças.**

Manifesto Ágil

Os 12 princípios são:

1. **Satisfação do cliente através da entrega contínua e antecipada de software valioso.**
2. **Mudanças nos requisitos são bem-vindas, mesmo tardiamente no desenvolvimento. Processos ágeis aproveitam mudanças para oferecer vantagem competitiva ao cliente.**

Manifesto Ágil

3. **Entregar software funcional frequentemente, com uma preferência pela entrega em intervalos de semanas, em vez de meses. Pessoas do negócio e desenvolvedores devem trabalhar juntos de forma constante, durante todo o projeto.**
4. **Pessoas do negócio e desenvolvedores devem trabalhar juntos de forma constante, durante todo o projeto.**

Manifesto Ágil

5. **Construir projetos em torno de indivíduos motivados. Prover o ambiente e o suporte de que eles precisam e confiar neles para realizar o trabalho.**
6. **A comunicação face a face é o método mais eficiente: Priorizar a comunicação direta para facilitar a troca de informações e a resolução de problemas.**
7. **Software funcional é a principal medida de progresso.**

Manifesto Ágil

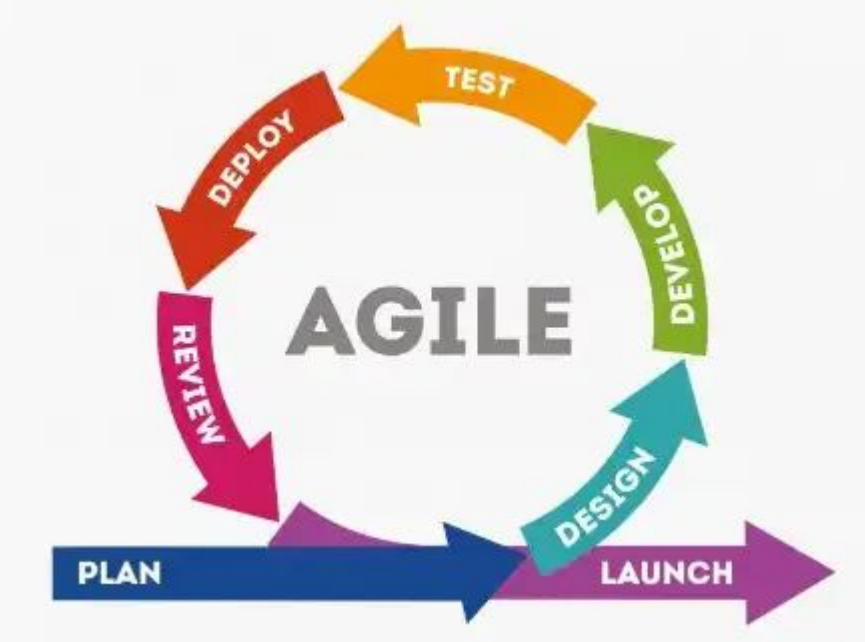
8. **Desenvolvimento sustentável, que permita manter um ritmo constante de entrega.**
9. **Atenção contínua à excelência técnica e ao bom design aumenta a agilidade.**
10. **Simplicidade: a arte de maximizar a quantidade de trabalho não realizado é essencial.**

Manifesto Ágil

11. **As melhores arquiteturas, requisitos e designs surgem de equipes auto-organizadas.**
12. **Reflexão regular sobre como se tornar mais eficaz e ajuste do comportamento da equipe em conformidade.**

Tipos de Metodologias Ágeis

SCRUM



SCRUM

O Scrum é um **framework ágil** focado em entregas incrementais e no trabalho colaborativo das equipes. Ele é baseado em ciclos curtos de trabalho chamados Sprints (geralmente de 1 a 4 semanas).

SCRUM

SCRUM

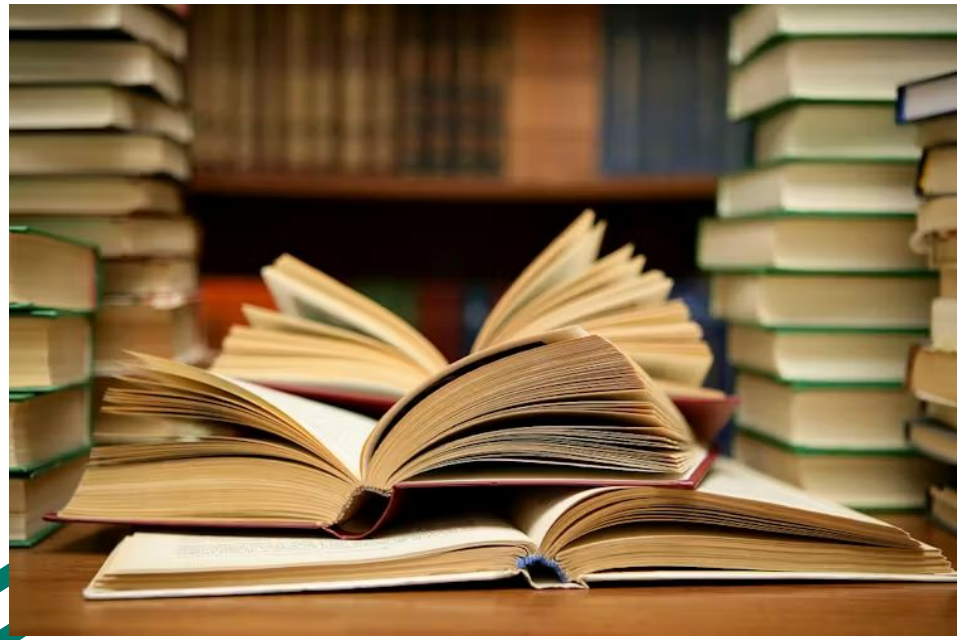
Scrum Master é responsável por facilitar o processo Scrum, garantindo que a equipe siga os princípios e práticas do framework Scrum de forma eficaz.

Product Owner é responsável por maximizar o valor do produto desenvolvido pela equipe. Ele faz isso definindo claramente o que precisa ser feito, garantindo que o time esteja trabalhando nas prioridades certas.

SCRUM

Time de Desenvolvimento, composto pelos profissionais responsáveis por construir o produto.

SCRUM



SCRUM

Product Backlog: é uma lista priorizada de tudo o que é necessário para o produto. Ele inclui funcionalidades, melhorias, correções e outros requisitos.

O Product Owner é responsável por essa parte.

SCRUM

Sprint Backlog: é um conjunto de itens do Product Backlog selecionados para o Sprint, com um plano de como serão entregues.

Etapa exclusiva para a equipe de desenvolvedores, onde irão definir o prazo de entrega. Essas informações são atualizadas diariamente, através de reuniões **(daily)**.

SCRUM

Incremento: é o resultado do trabalho realizado durante a Sprint, ou seja, a funcionalidade entregue e que deve ser potencialmente utilizável pelo cliente.

SCRUM

SCRUM

Sprint Planning (Planejamento da Sprint), ocorre no início de cada Sprint.

Todos os integrantes da equipe se reúnem para conhecer os itens do **Product Backlog** (funcionalidades/requisitos/melhorias do projeto).

SCRUM

Daily Scrum ou simplesmente Daily (Reuniões Diárias), o Scrum Master e a equipe de desenvolvimento se reúne para saber como está o andamento do projeto.

Nessas reuniões, desenvolvedores explicam sobre o que fizeram no dia seguinte e o que irão fazer após a daily, também devem relatar eventuais dificuldades ou dúvidas do projeto.

SCRUM

Sprint Review (Revisão da Sprint), ocorre ao término de cada Sprint.

Todos os integrantes da equipe e os stakeholders (partes interessadas) ficam cientes da elaboração executada em uma determinada Sprint.

SCRUM

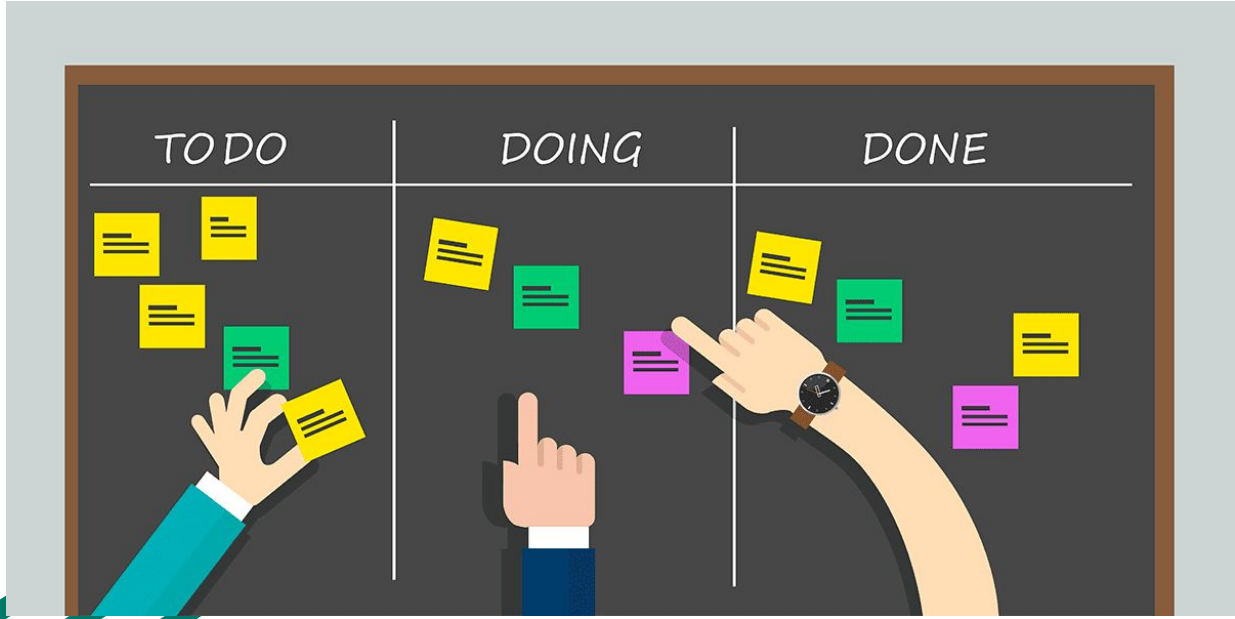
Sprint Retrospective (Retrospectiva da Sprint), ocorre após a Spring Review.

O Scrum Master e a equipe de desenvolvimento conversam sobre melhorias.

SCRUM

SCRUM

KANBAN



KANBAN

Kanban é um método visual de gestão de projetos e fluxo de trabalho, originado da Toyota, que utiliza um quadro com colunas para representar as etapas de um processo. O objetivo é visualizar o trabalho em progresso, identificar gargalos e otimizar a eficiência.

KANBAN

1. **Melhora a visibilidade do fluxo de trabalho.**
2. **Aumenta a eficiência.**
3. **Reduz o trabalho inacabado.**
4. **Promove a colaboração.**
5. **Adaptação incremental.**

KANBAN

EXTREME PROGRAMMING

EXTREME PROGRAMMING

Programação Extrema (XP) é uma metodologia ágil de desenvolvimento de software que enfatiza a colaboração, a comunicação, o feedback rápido e a entrega contínua de valor ao cliente.

Ela se concentra em práticas de engenharia como programação em pares, testes unitários e de aceitação, e refatoração contínua para garantir alta qualidade e adaptabilidade às mudanças.

EXTREME PROGRAMMING

Kent Back - Criador da Metodologia Ágil Extreme Programming (XP).

EXTREME PROGRAMMING

Características da metodologia Extreme Programming (XP):

1. **Ágil:** Adaptável a mudanças e focado em ciclos de desenvolvimento curtos.
2. **Colaborativo:** Incentiva a comunicação e o trabalho em equipe.
3. **Entregando valor:** Busca fornecer valor ao cliente de forma contínua e rápida.
4. **Qualidade:** Prioriza a qualidade do código e do produto final.

EXTREME PROGRAMMING

Práticas comuns:

1. **Programação em pares:** Dois desenvolvedores trabalham juntos no mesmo código, com um programando e o outro revisando.

2. **Testes unitários:** Criação de testes automatizados para cada unidade de código, garantindo que funcione corretamente.

EXTREME PROGRAMMING

3. **Testes de aceitação:** Testes que verificam se o sistema atende às necessidades do cliente, escritos em conjunto com o cliente.

4. **Refatoração:** Melhoria contínua do código para torná-lo mais limpo, eficiente e fácil de manter.

5. **Integração contínua:** Integração do código com frequência, geralmente várias vezes ao dia, para detectar erros rapidamente.

EXTREME PROGRAMMING

6. **Metáfora:** Uso de analogias para facilitar a comunicação entre desenvolvedores e clientes.

7. **Padrões de código:** Adoção de padrões de código para garantir consistência e facilitar a manutenção.

EXTREME PROGRAMMING

Valores:

1. **Comunicação:** Promover a comunicação aberta entre a equipe e o cliente.
2. **Simplicidade:** Buscar soluções simples e eficazes.
3. **Feedback:** Buscar e fornecer feedback constante.
4. **Coragem:** Ter coragem para enfrentar desafios, tomar decisões difíceis e aprender com os erros.
5. **Respeito:** Valorizar as contribuições e opiniões de todos.

Feature Driven Development

Feature Driven Development

O Feature Driven Development (FDD), ou Desenvolvimento Orientado a Funcionalidades, é uma metodologia ágil que se concentra na entrega de software através do desenvolvimento de funcionalidades específicas, ou "features", de forma iterativa e incremental.

Feature Driven Development

Os 5 processos principais do FDD:

1. **Desenvolvimento do modelo geral:** Criação de uma visão geral do projeto e do domínio do problema.

2. **Criação da lista de funcionalidades:** Identificação e definição das funcionalidades que serão desenvolvidas.

Feature Driven Development

3. **Planejamento por funcionalidade:** Priorização e planejamento da entrega das funcionalidades.

4. **Modelagem por funcionalidade:** Modelagem detalhada de cada funcionalidade, incluindo diagramas e classes.

Feature Driven Development

5. **Construção por funcionalidade:** Desenvolvimento e teste da funcionalidade, seguindo os planos e modelos definidos.

Feature Driven Development

Vantagens do FDD:

1. **Entrega rápida e frequente:** Permite a entrega de funcionalidades em curtos períodos de tempo, com resultados visíveis para o cliente.

2. **Redução de riscos:** A divisão em funcionalidades menores e o desenvolvimento iterativo ajudam a identificar e mitigar riscos mais cedo.

Feature Driven Development

Vantagens do FDD:

3. **Maior adaptabilidade:** Permite ajustes e mudanças no escopo do projeto de forma mais fácil, devido à sua natureza iterativa.

4. **Melhora na colaboração:** O FDD incentiva a colaboração entre as equipes e com o cliente, o que pode levar a um melhor alinhamento de expectativas e resultados.

Crystal

Crystal

A Metodologia Crystal é um método ágil de desenvolvimento de software e gerenciamento de projetos que foi criado para atender às necessidades específicas de cada projeto, com foco nas características da equipe e no contexto do trabalho.

Crystal

Criador da Metodologia Agil Crystal: Alister Cockburn

Crystal

Method	Team size	Project size
Crystal Clear	6 people or less	Small projects
Crystal Yellow	7–20 people	Small to medium projects
Crystal Orange	20–40 people	Medium projects
Crystal Red	40–80 people	Medium to large projects
Crystal Maroon	80–200 people	Large projects
Crystal Diamond **or** Crystal Sapphire	Very large teams (200+ people)	Very large projects with high criticality

Fonte: https://www.wrike.com/agile-guide/faq/what-is-agile-crystal-methodology/

Crystal

Cockburn (2004) sugere que o líder do projeto proceda da seguinte forma :

1. Identifique a faixa em que o software a ser desenvolvido se enquadra.

2. Estabeleça os papéis de cada participante do projeto : Líder, Programador, Analista, Arquiteto de Software, Analista de Qualidade, etc.

Crystal

3. Comunicação: etapa em que são levantados os requisitos junto aos clientes e usuários.

4. Planejamento: refere-se ao plano de ação para desenvolver o software, incluindo cronograma e planilha de custos.

5. Modelagem: fase em que analistas modelam os requisitos levantados.

Crystal

6. Construção: codificação do software com a linguagem e uso de banco de dados.

7. Entrega do projeto ou de uma determinada funcionalidade.

Crystal

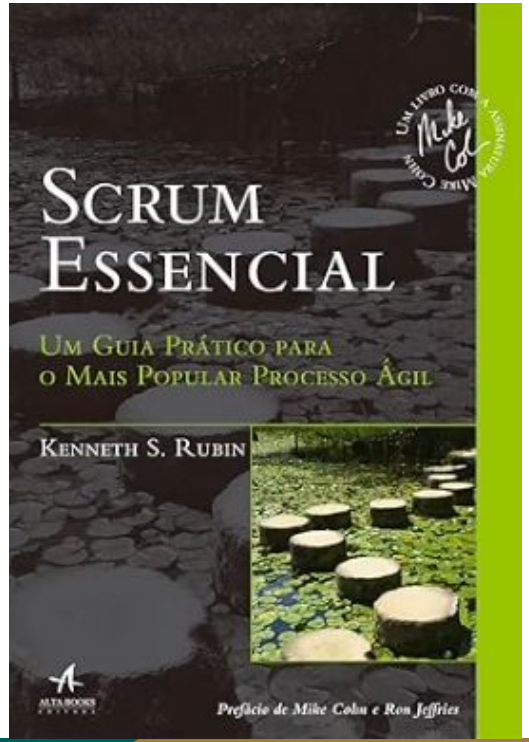
6. Construção: codificação do software com a linguagem e uso de banco de dados.

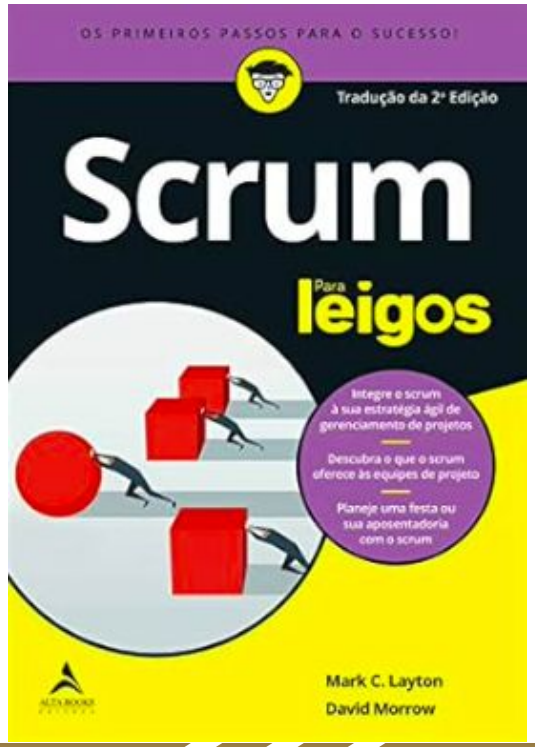
7. Entrega do projeto ou de uma determinada funcionalidade.

Dicas para leitura

SCRUM

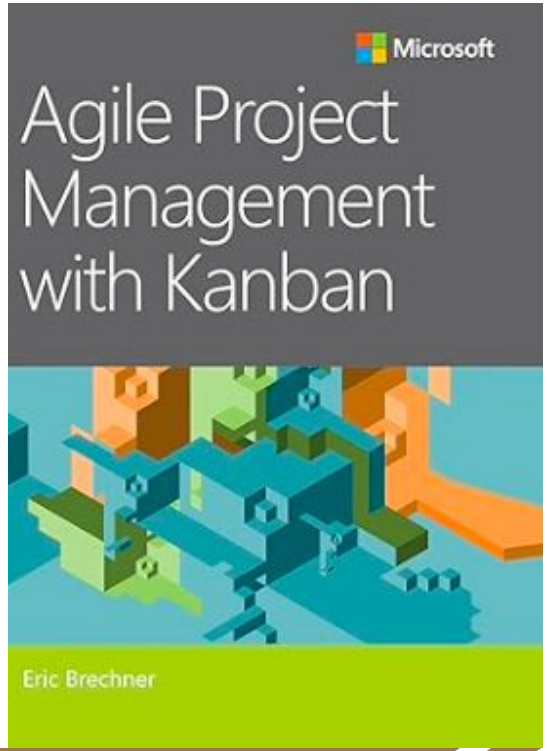

SCRUM
A arte de fazer o dobro do trabalho na metade do tempo
JEFF SUTHERLAND
Cocriador do Scrum
e J.J. SUTHERLAND


SCRUM ESSENCIAL
UM GUIA PRÁTICO PARA O MAIS POPULAR PROCESSO ÁGIL
KENNETH S. RUBIN
Prefácio de Mike Cohn e Ron Jeffries

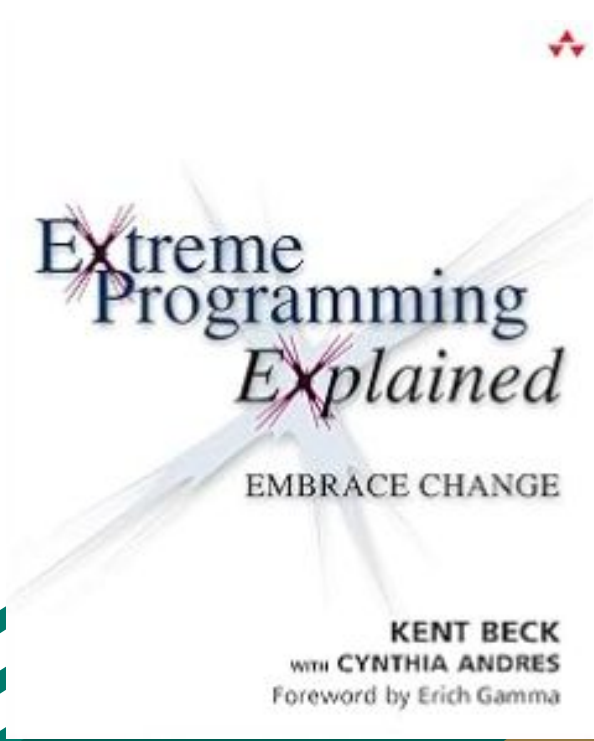

OS PRIMEIROS PASSOS PARA O SUCESSO!
Tradução da 2ª Edição
Scrum
Para leigos
Integre o scrum à sua estratégia ágil de gerenciamento de projetos
Descubra o que o scrum oferece às equipes de projeto
Planeje uma festa ou sua aposentadoria com o scrum
Mark C. Layton
David Morrow

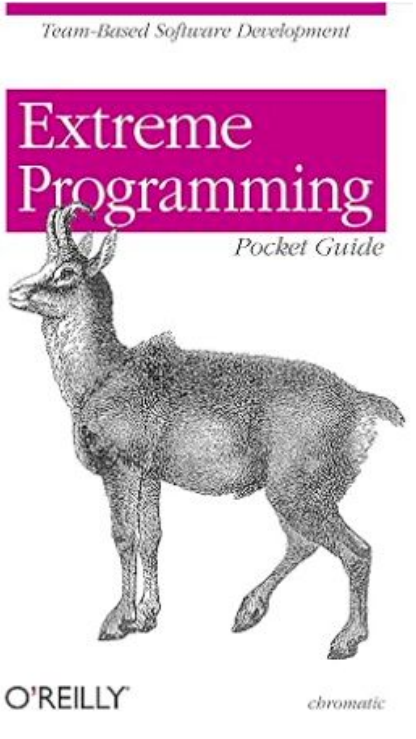
KANBAN


LIVRO UM da série MELHOR COM KANBAN
DESCOBRINDO
KANBAN
O Caminho Evolucionário para
Agilidade Organizacional
DAVID J ANDERSON
Traduzido por Alcides Vieira Junior

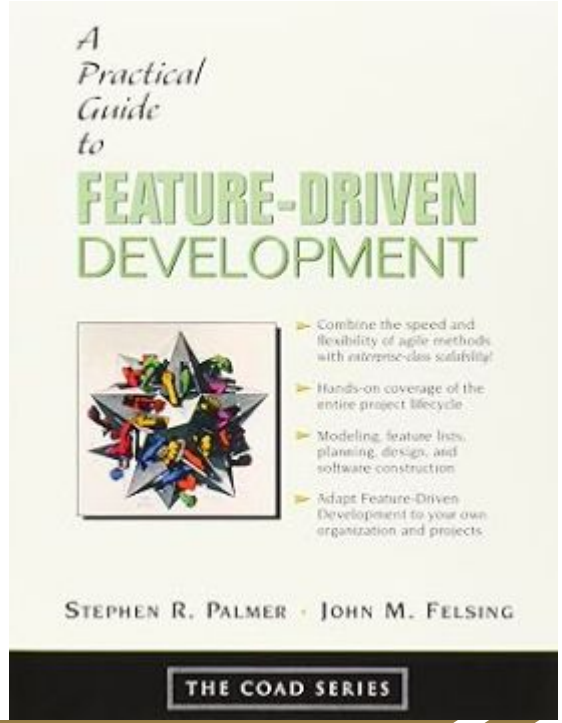

Microsoft
Agile Project
Management
with Kanban
Eric Brechner

EXTREME PROGRAMMING

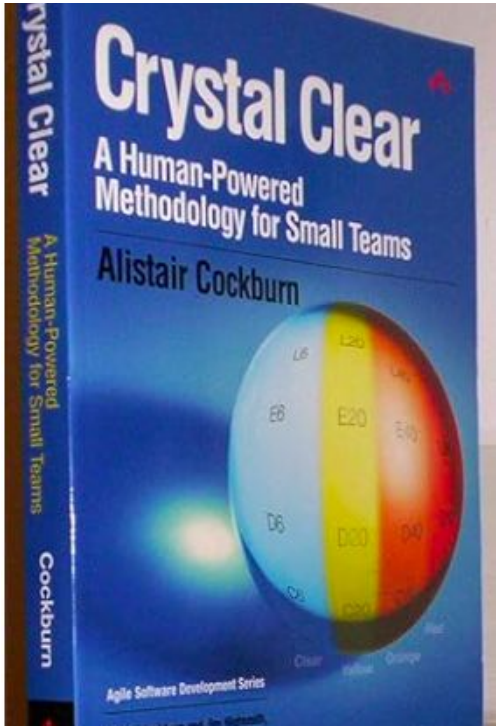




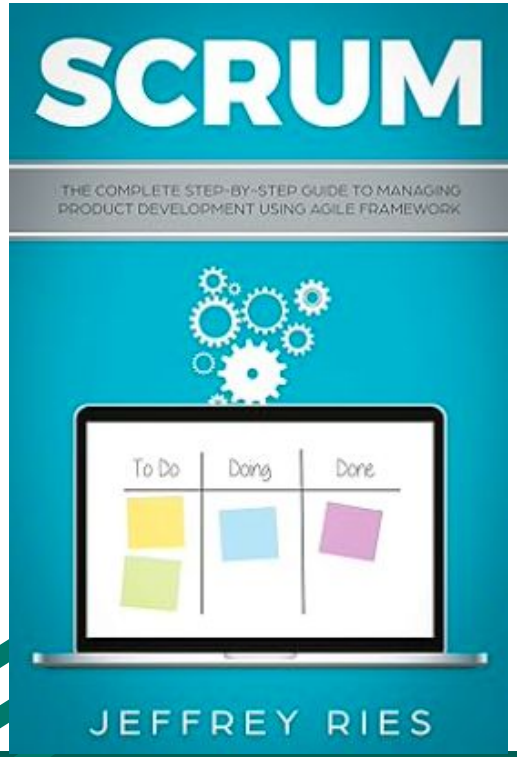
FEATURE DRIVEN DEVELOPMENT



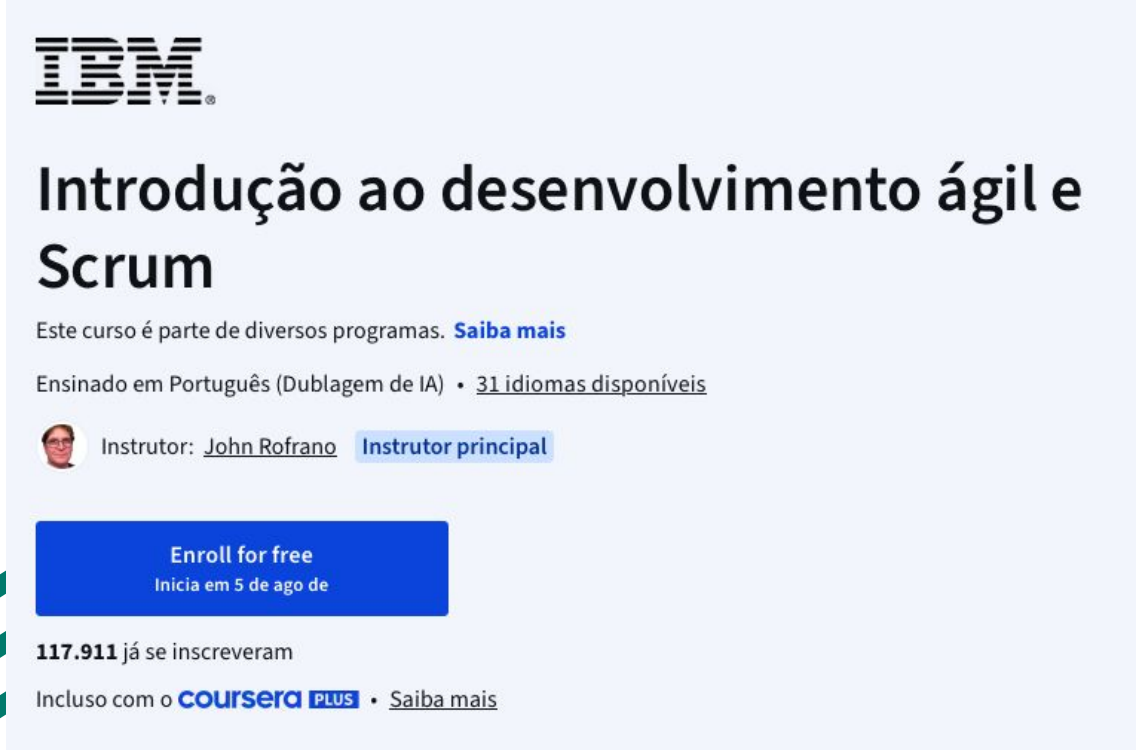
CRYSTAL


Crystal Clear
A Human-Powered Methodology for Small Teams
Alistair Cockburn
Agile Software Development Series
Cockburn

INDICAÇÃO PESSOAL



Dica de curso



Atividade avaliativa

1. O projeto pode ser feito: sozinho, em dupla ou em trio.
2. Data limite para entrega: 07/08/2025 às 22:15.
3. O projeto deverá estar em um README no GitHub (nada de links).
4. O projeto não precisa estar no GitHub de todos os integrantes.

Atividade avaliativa

5. Ao enviar o e-mail, certifique-se de que as seguintes informações estejam disponíveis:
    a. Link do repositório do GitHub contendo o README (deixar público).
    b. Nome de todos os integrantes da equipe.

Atividade avaliativa

Enunciado: Pense em um sistema e implemente cada Sprint que deve ser desenvolvida.

O projeto deverá ter pelo menos 20 Sprints e no máximo 30.

*** Cada Sprint deverá estar explícita no README.**

Atividade avaliativa

Exemplo:

Sprint 1: Desenvolvimento da autenticação de usuários.
O que foi feito: Implementado o front-end contendo um formulário (e-mail e senha) e também a opção de login via Google ou Microsoft. No back-end foram criados os métodos: autenticarFormulario, autenticarGoogle e autenticarMicrosoft.
Desenvolvedores: João e Maria.

Atividade avaliativa

O que cada desenvolvedor fez: Explique em detalhes o que cada um implementou.

Período de desenvolvimento: 5 dias.

Período de testes: 1 dia.

Período de revisão: 2 dias.

Período de deploy: 1 dia.

Data inicial e final da Sprint: 04/08/2025 - 14/08/2025.

Observações: Opcional.

Atividade avaliativa

Importante: Além das Sprints, no README deverá ter:

- Nome dos integrantes;
- Tecnologias utilizadas (front-end, back-end e banco de dados);
- Descrição do projeto;
- Quantidade total de Sprints;
- Um resumo da equipe sobre as vantagens e dificuldades de implementar a Metodologia Ágil nos projetos.