Métodos Ágeis

MDS

Professora: Carla Rocha [UnB]

Email: caguiar@unb.br

Website: carlarocha.org

Agenda

1

Engenharia Tradicional

Modelo Cascata e suas características

2

Manifesto Ágil

Valores e princípios fundamentais

3

Extreme Programming

Valores, princípios e práticas

4

Scrum

Eventos, papéis e artefatos

5

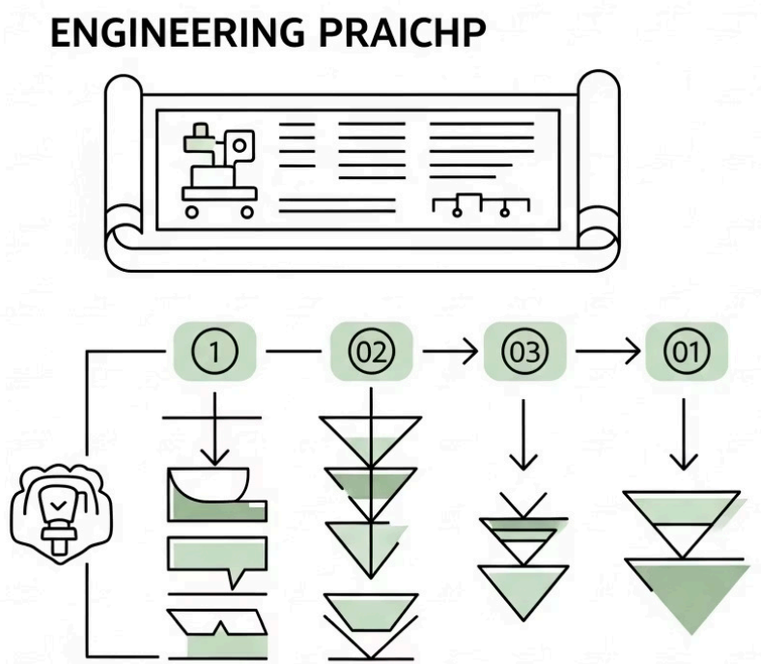
A disciplina

Avaliação, projeto, onboarding e Git

Engenharia Tradicional

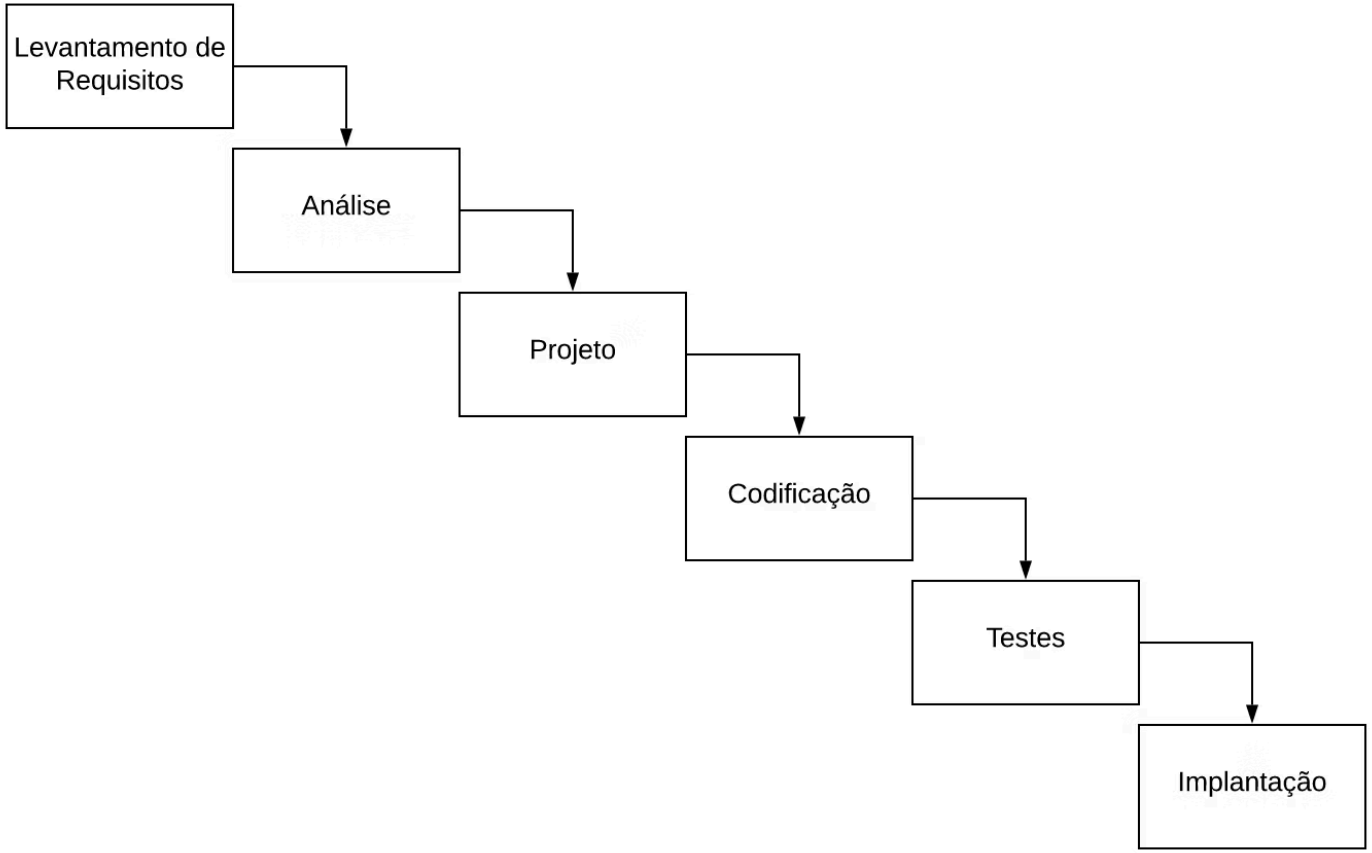
Características

- Aplicada em áreas como civil, mecânica, elétrica, aviação, etc.
- Projetos com planejamento detalhado (big upfront design)
- Processo sequencial e bem definido
- Modelo Waterfall (Cascata)
- Metodologia utilizada há milhares de anos



O modelo cascata segue fases rigidamente sequenciais, com pouca flexibilidade para mudanças.

Natural que ES começasse usando Waterfall



O modelo cascata foi a primeira abordagem formal para o processo de desenvolvimento de software, inspirado diretamente nas engenharias tradicionais.

Waterfall não funcionou com Software!

Software é Diferente

Engenharia de Software ≠ Engenharia Tradicional

O desenvolvimento de software possui particularidades que o tornam único em relação às engenharias convencionais.

Software ≠ Produtos Físicos

Diferente de carros, pontes ou celulares, o software é intangível e não segue as mesmas regras de produção.

Software é Abstrato e Adaptável

A natureza do software permite mudanças constantes e adaptações durante todo o ciclo de desenvolvimento.

Dificuldade 1: Requisitos

Clientes não sabem exatamente o que querem

- As possibilidades de funcionalidades são praticamente infinitas
- É difícil prever todas as necessidades antecipadamente
- O mundo muda rapidamente, e com ele as necessidades

Não é viável passar 1 ano levantando requisitos, 1 ano projetando, 1 ano implementando...

Quando o software ficar pronto, ele estará obsoleto!

Dificuldade 2: Documentação Detalhada

Documentos Verbosos e Pouco Úteis

Documentações extensas raramente são consultadas durante o desenvolvimento e rapidamente ficam desatualizadas.

Desconsiderada na Implementação

Na prática, desenvolvedores frequentemente ignoram a documentação detalhada durante a codificação, seguindo o que faz mais sentido no momento.

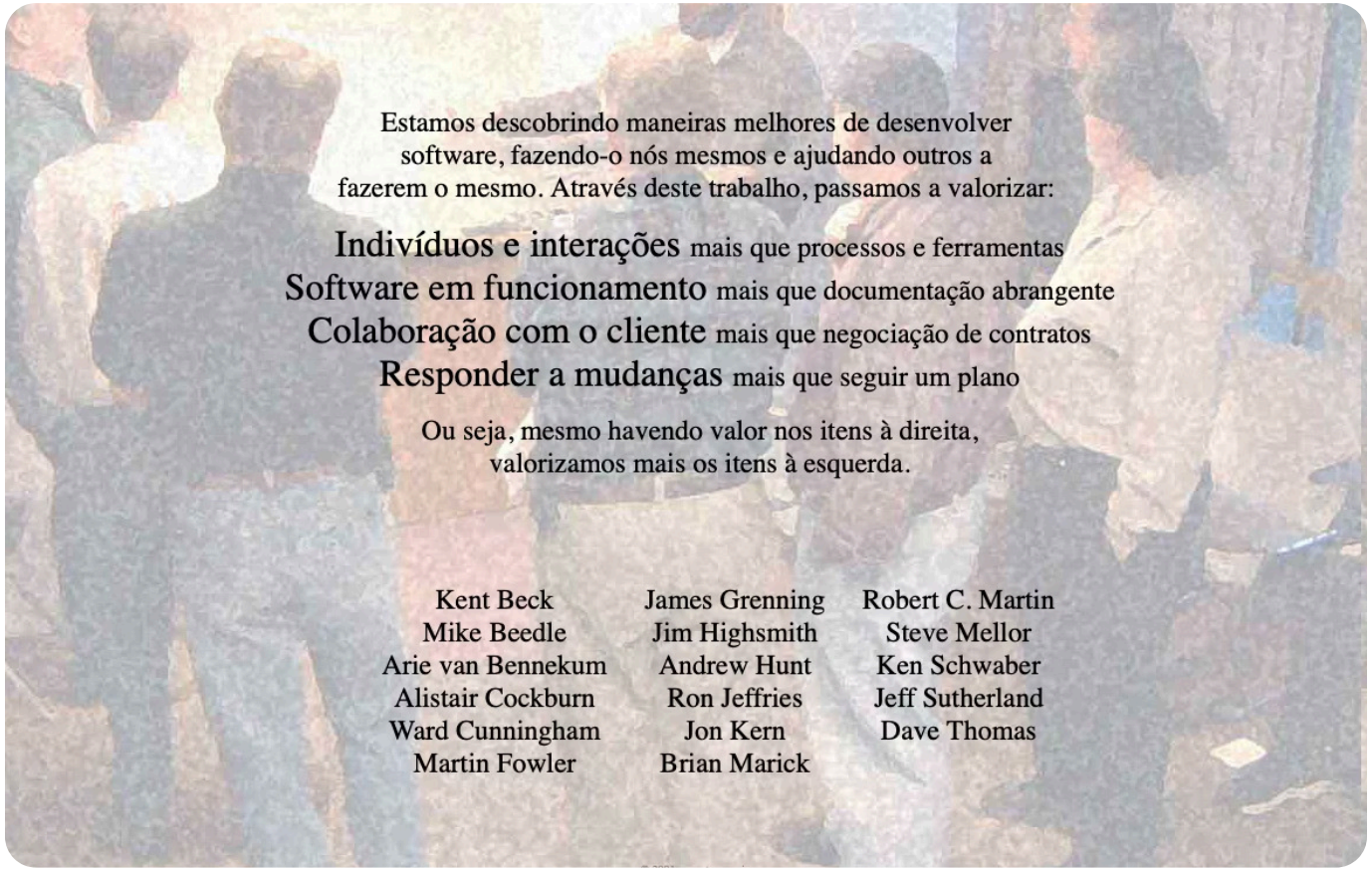
Plan-and-Document Falhou

A abordagem de planejar tudo detalhadamente e documentar exaustivamente antes de codificar não funcionou no desenvolvimento de software.

Manifesto Ágil

Em 2001, 17 desenvolvedores de software se reuniram em Snowbird, Utah, para discutir alternativas aos processos burocráticos de desenvolvimento, criando assim o Manifesto Ágil.

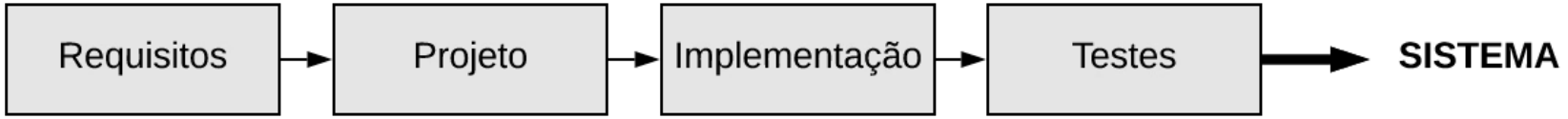
Manifesto Ágil



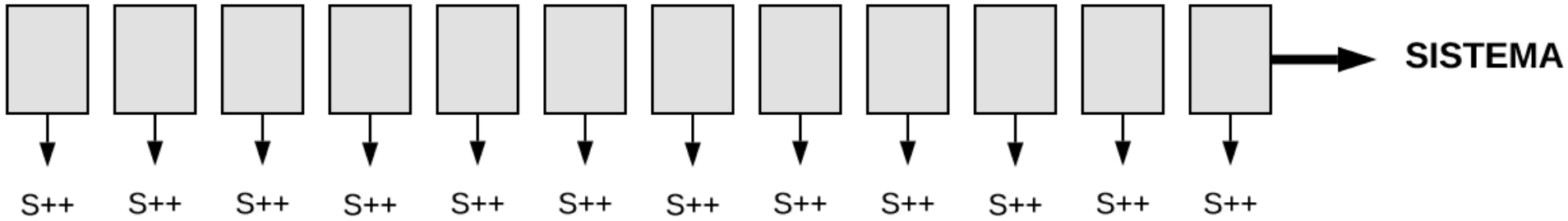
O Manifesto estabelece quatro valores fundamentais que priorizam aspectos humanos e práticos sobre processos rígidos e documentação excessiva.

Ideia Central: Desenvolvimento Iterativo

Waterfall



Ágil



A diferença fundamental: enquanto o Waterfall segue uma sequência linear de fases completas, o Ágil repete pequenos ciclos completos de desenvolvimento.

Desenvolvimento Iterativo

Incrementos Pequenos

Mesmo para sistemas imensos e complexos, devemos identificar o menor "incremento de sistema" que pode ser implementado em um curto período (como 15 dias).

Validação Constante

Cada incremento é validado com o usuário, permitindo feedback imediato e ajustes no direcionamento do projeto.

Descoberta Gradual

Como o cliente raramente sabe exatamente o que quer no início, o processo iterativo permite descobrir e refinar os requisitos ao longo do desenvolvimento.

Reforçando:

Ágil = Iterativo

Outros Pontos Importantes

Menor Ênfase em Documentação

Documentação enxuta e funcional, focada apenas no que realmente agrega valor ao desenvolvimento.

Menos Design Antecipado

Redução do *big upfront design*, permitindo que a arquitetura emerja e evolua com o desenvolvimento.

Envolvimento do Cliente

Participação constante do cliente durante todo o processo de desenvolvimento, não apenas no início e fim.

Outros Pontos Importantes

Novas Práticas de Programação

Testes Automatizados

Criação de testes antes ou durante o desenvolvimento para garantir a qualidade do código.

Refactoring

Melhoria contínua do código sem alterar seu comportamento externo.

Integração Contínua

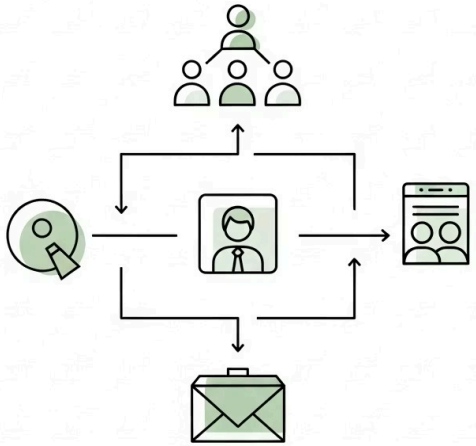
Integração frequente do código ao repositório principal, reduzindo conflitos.

Estas práticas técnicas são essenciais para sustentar o desenvolvimento iterativo e entregar software de qualidade.

Métodos Ágeis

Vamos explorar as implementações práticas dos princípios ágeis através de metodologias específicas.

Métodos Ágeis



Dão mais consistência às ideias ágeis

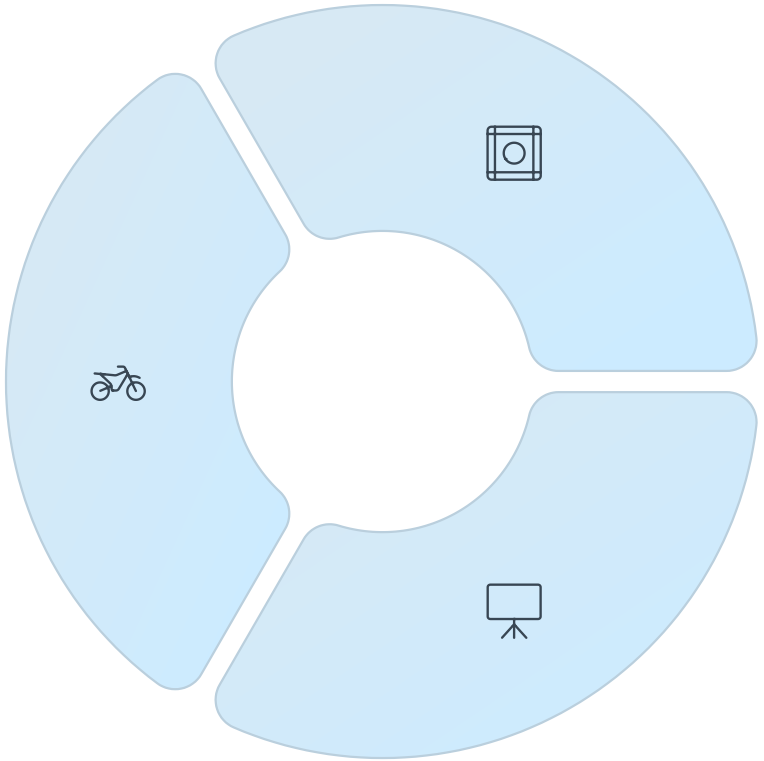
- Definem um processo estruturado, mesmo que leve
- Estabelecem workflow, eventos, papéis e práticas específicas
- Fornecem princípios e diretrizes claras para as equipes
- Criam uma linguagem comum para os participantes
- Facilitam a adoção dos valores ágeis na prática

Métodos Ágeis que Vamos Estudar

Extreme Programming (XP)

Foco em práticas técnicas de excelência em engenharia de software

- Programação pareada
- TDD (Test-Driven Development)
- Integração contínua



Scrum

Framework de gerenciamento de projetos iterativo e incremental

- Sprints
- Papéis bem definidos
- Eventos e artefatos específicos

Kanban

Sistema visual para gerenciar o trabalho enquanto ele se move pelo processo

- Fluxo contínuo
- Limites de trabalho em progresso
- Melhoria contínua

Antes de Começar

> ## Nenhum processo é uma bala-de-prata

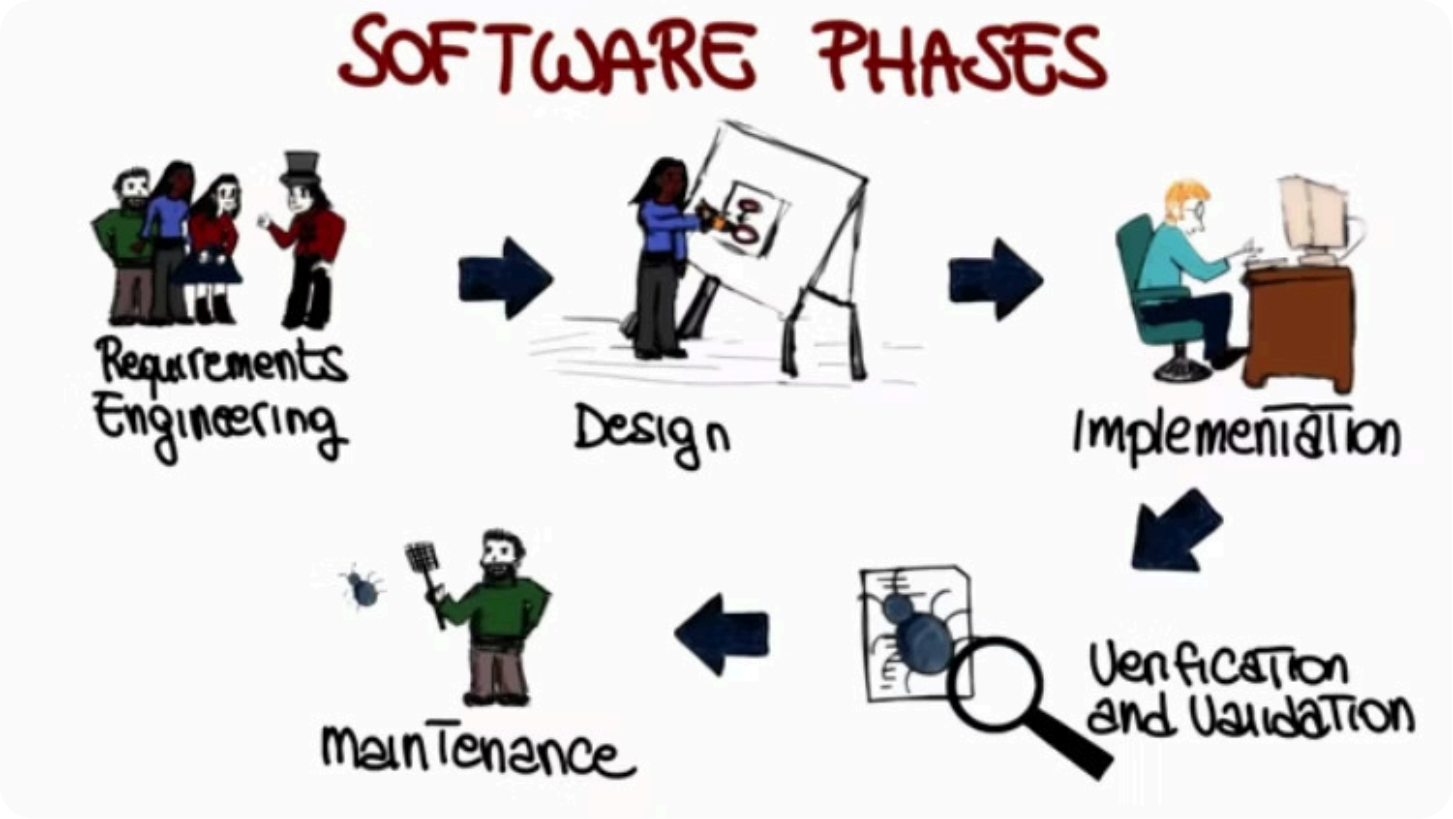
Limitações dos Processos

- Processos ajudam a evitar "grandes erros", mas não garantem sucesso
- Cada projeto e equipe têm necessidades únicas
- Métodos ágeis são frameworks, não receitas rígidas

Adaptação Necessária

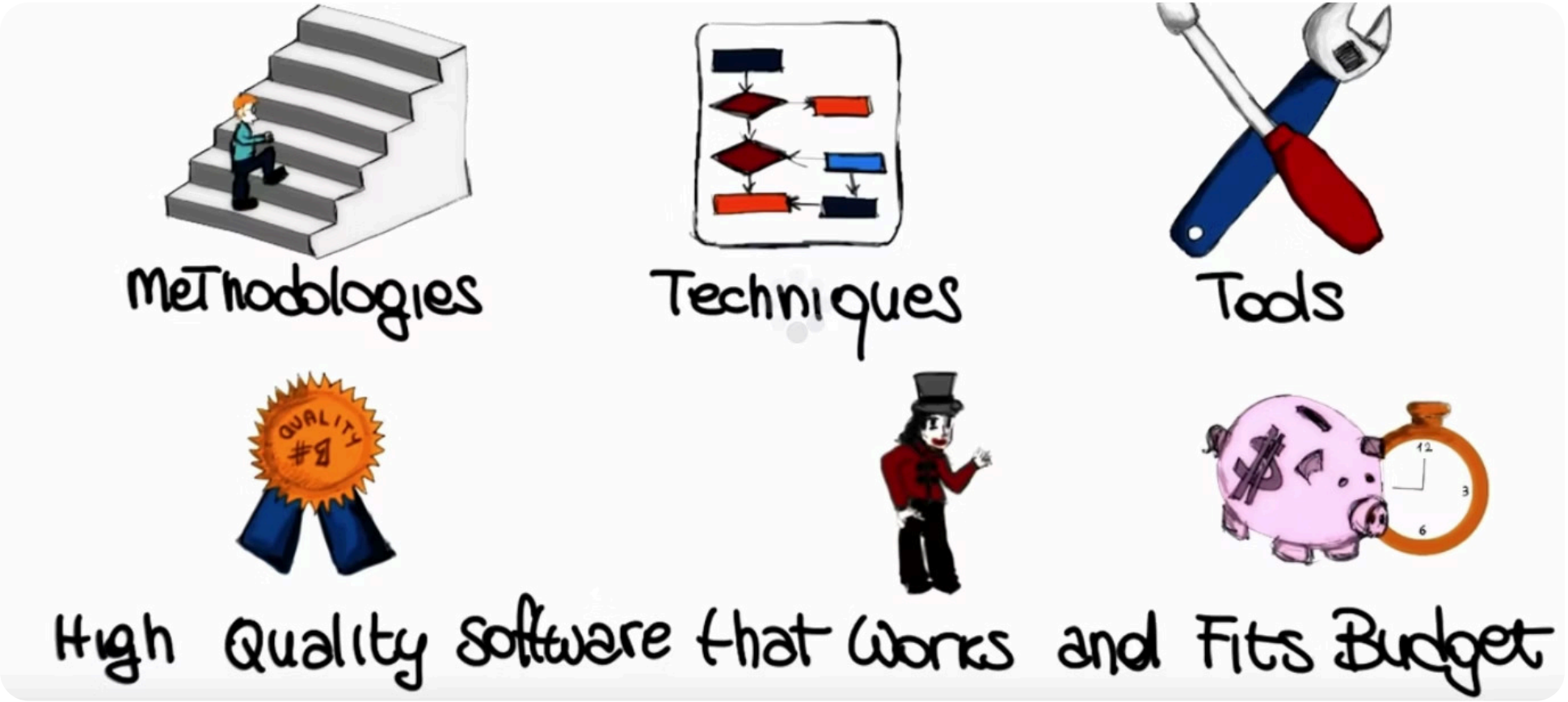
- Processos raramente são adotados 100% como no manual
- Bom senso e discernimento são fundamentais
- Experimentação e ajustes contínuos são esperados

Antes de Começar



É importante entender a diferença entre seguir um processo rigidamente e adotar seus princípios de forma adaptativa. A escolha entre processos deve considerar o contexto e necessidades específicas.

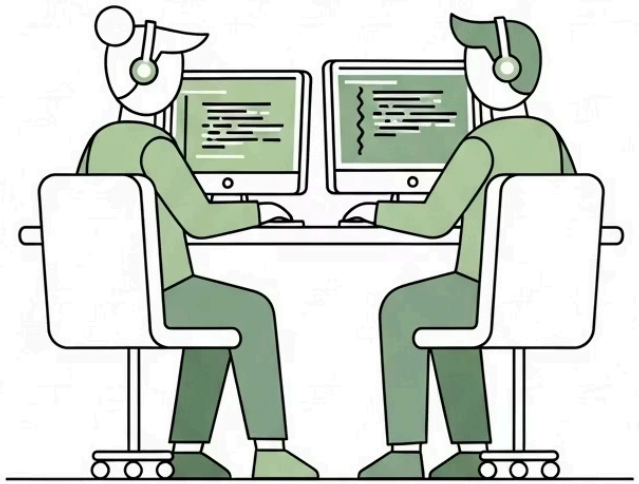
Antes de Começar



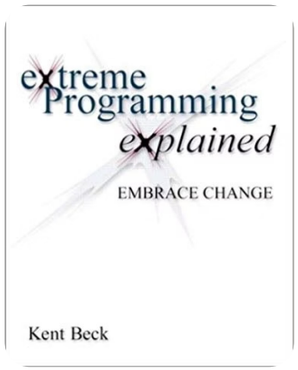
O gráfico demonstra como diferentes contextos exigem diferentes abordagens. Um método único não é adequado para todos os tipos de projetos.

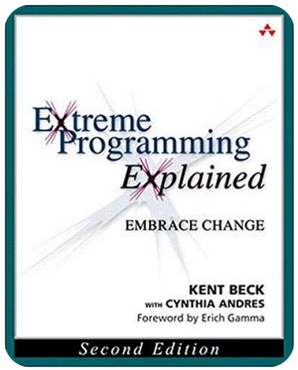
Extreme Programming (XP)

Uma metodologia ágil focada em práticas técnicas de excelência em engenharia de software.



Extreme Programming





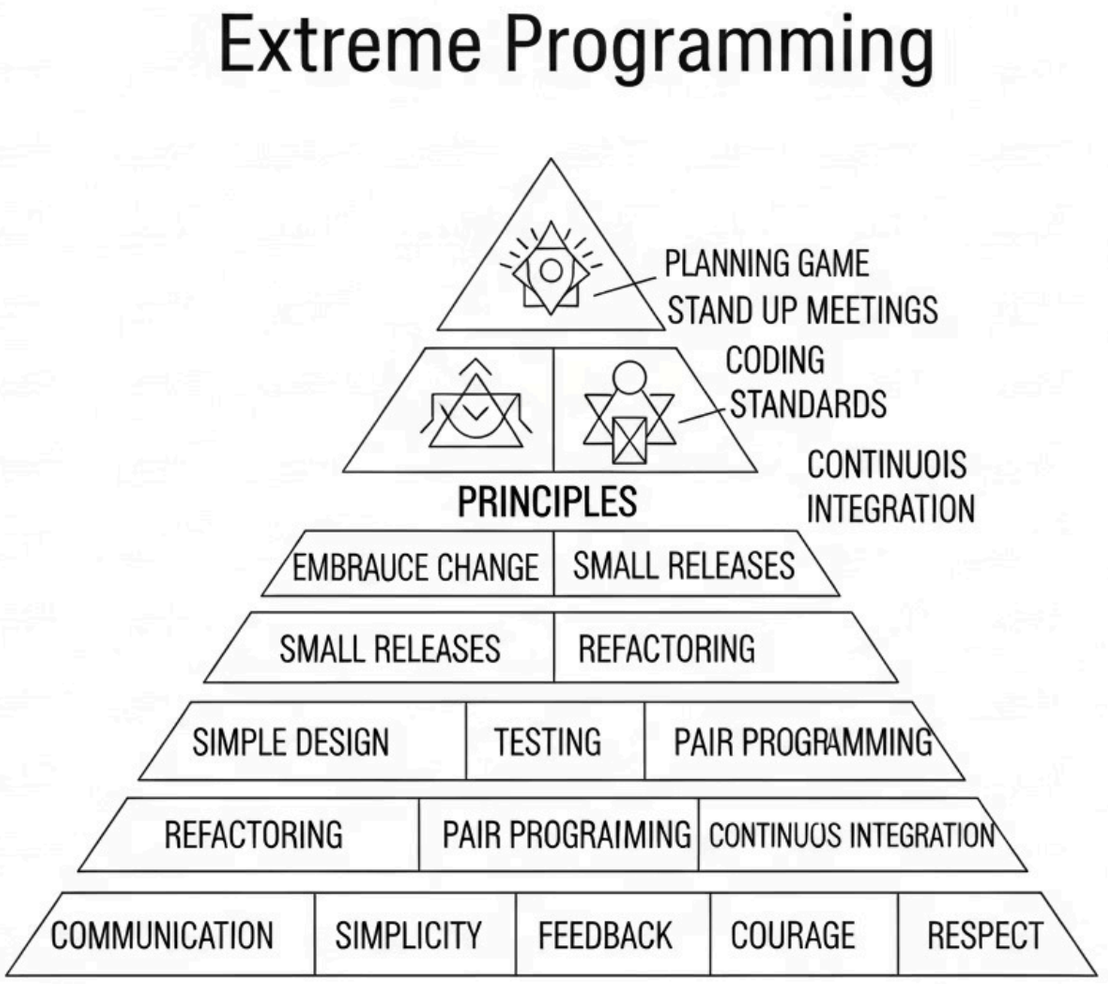
Criado por Kent Beck em 1999

O XP foi uma das primeiras metodologias ágeis formalizadas, com foco em:

- Práticas técnicas de programação
- Feedback rápido e contínuo
- Comunicação clara e direta
- Simplicidade em todos os aspectos
- Coragem para enfrentar mudanças

Em 2004, a segunda edição do livro expandiu os conceitos originais, reforçando valores e princípios.

XP = Valores + Princípios + Práticas



O XP é estruturado em três níveis complementares: valores fundamentais que sustentam princípios, que por sua vez orientam práticas específicas.

Valores

Comunicação

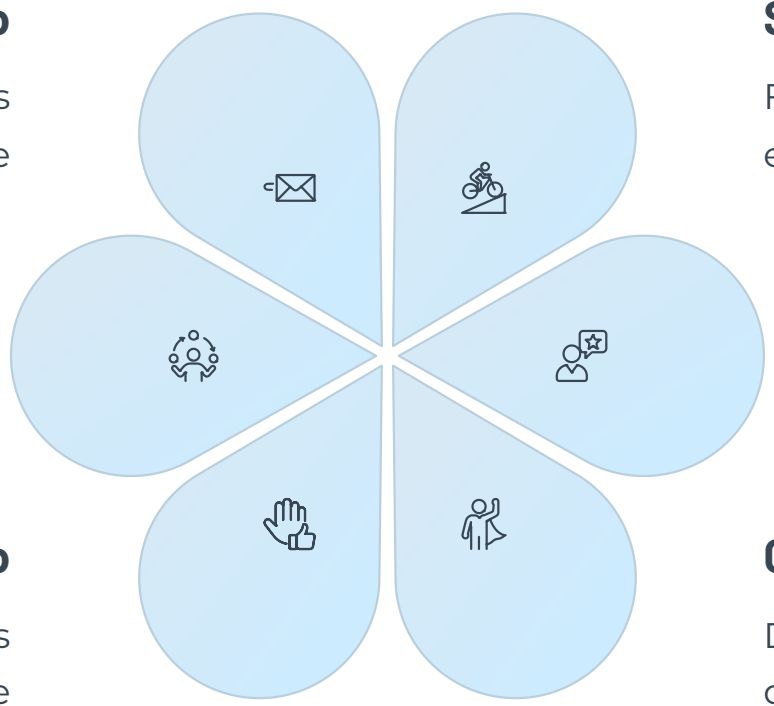
Diálogo constante entre todos os membros da equipe e com o cliente

Qualidade de Vida

Semana de 40 horas, evitando sobrecarga e burnout

Respeito

Valorização e consideração por todos os membros da equipe



Simplicidade

Fazer o mais simples que funcione, evitando complexidade desnecessária

Feedback

Ciclos curtos de retorno para orientar ajustes e melhorias

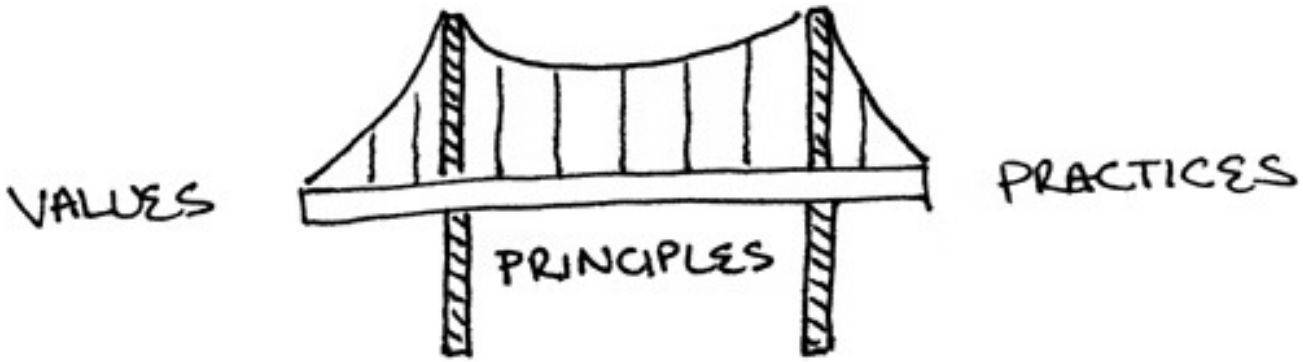
Coragem

Disposição para enfrentar mudanças e dizer verdades difíceis

Valores ou "cultura" são fundamentais em software!

Mais importante que processos ou ferramentas, a cultura organizacional e os valores compartilhados determinam o sucesso de projetos de software.

XP = Valores + Princípios + Práticas



Os princípios do XP estabelecem a ponte entre os valores fundamentais e as práticas concretas, fornecendo diretrizes mais específicas para orientar decisões.

Princípios

Economicidade

Priorizar o que traz maior valor ao cliente com o menor custo possível.

Melhorias Contínuas

Sempre buscar formas de melhorar o processo e o produto.

Falhas Acontecem

Aceitar que erros são parte do processo e aprender com eles.

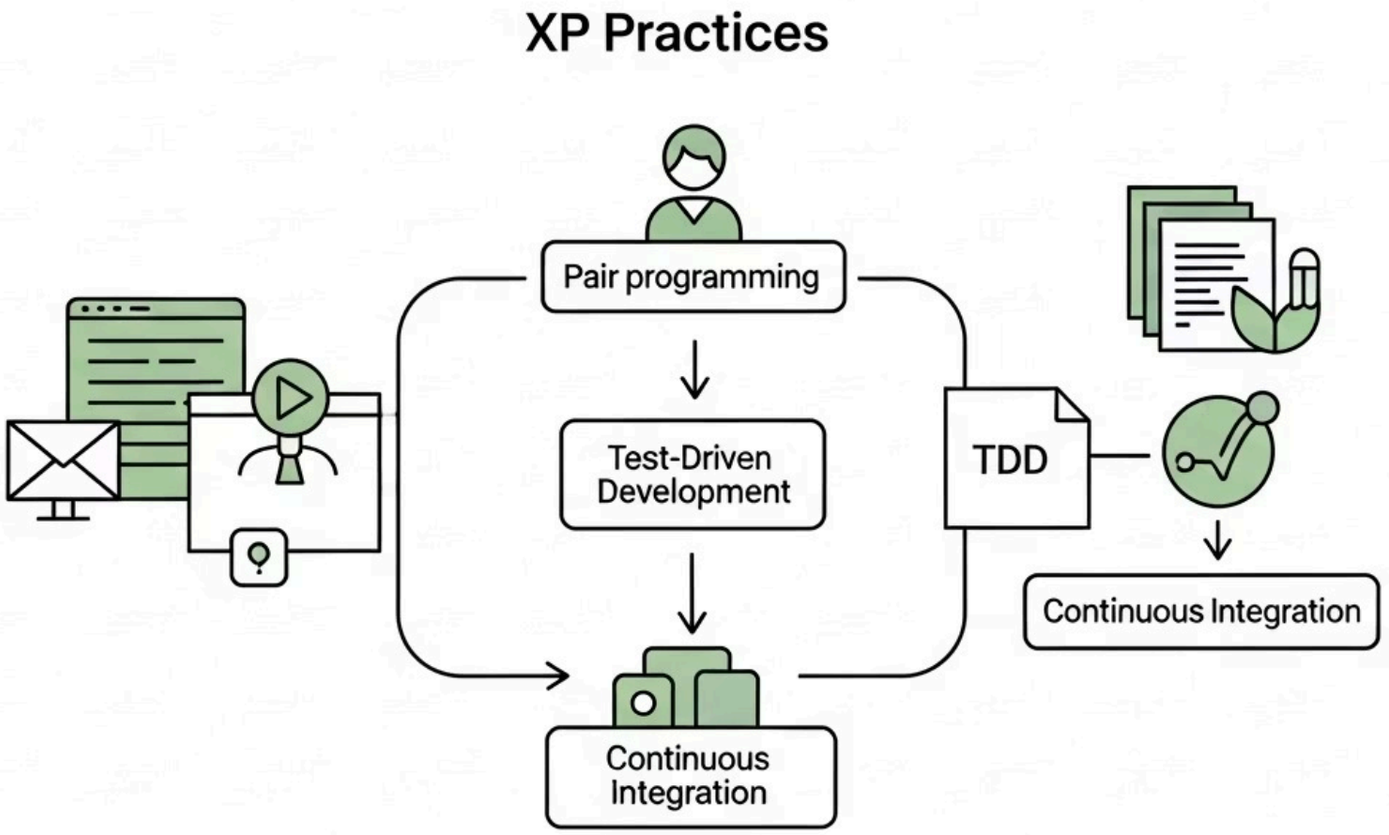
Baby Steps

Avançar em pequenos incrementos para minimizar riscos e maximizar aprendizado.

Responsabilidade Pessoal

Cada membro da equipe assume responsabilidade por seu trabalho e pelo sucesso coletivo.

XP = Valores + Princípios + Práticas



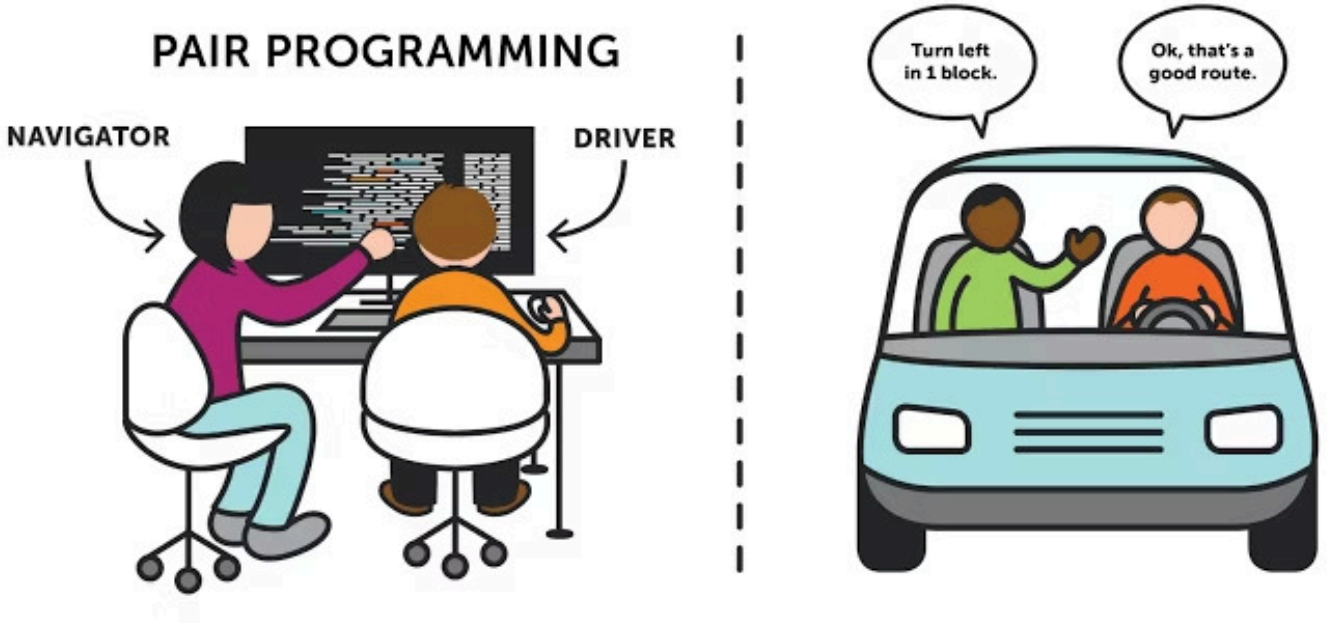
As práticas são a manifestação concreta dos valores e princípios, representando ações específicas que equipes XP realizam no dia a dia.

Práticas

Práticas sobre o Processo de Desenvolvimento	Práticas de Programação	Práticas de Gerenciamento de Projetos
Representante dos Clientes Histórias de Usuário Iterações Releases Planejamento de Releases Planejamento de Iterações Planning Poker Slack	Design Incremental Programação Pareada Testes Automatizados Desenvolvimento Dirigido por Testes (TDD) Build Automatizado Integração Contínua	Ambiente de Trabalho Contratos com Escopo Aberto Métricas

O XP define um conjunto de práticas técnicas e gerenciais que se reforçam mutuamente, criando um sistema coeso de desenvolvimento de software.

Programação Pareada



Uma das práticas mais distintivas do XP, a programação pareada envolve dois programadores trabalhando juntos em um único computador, alternando papéis de piloto e copiloto.

Estudo com Engenheiros da Microsoft (2008)

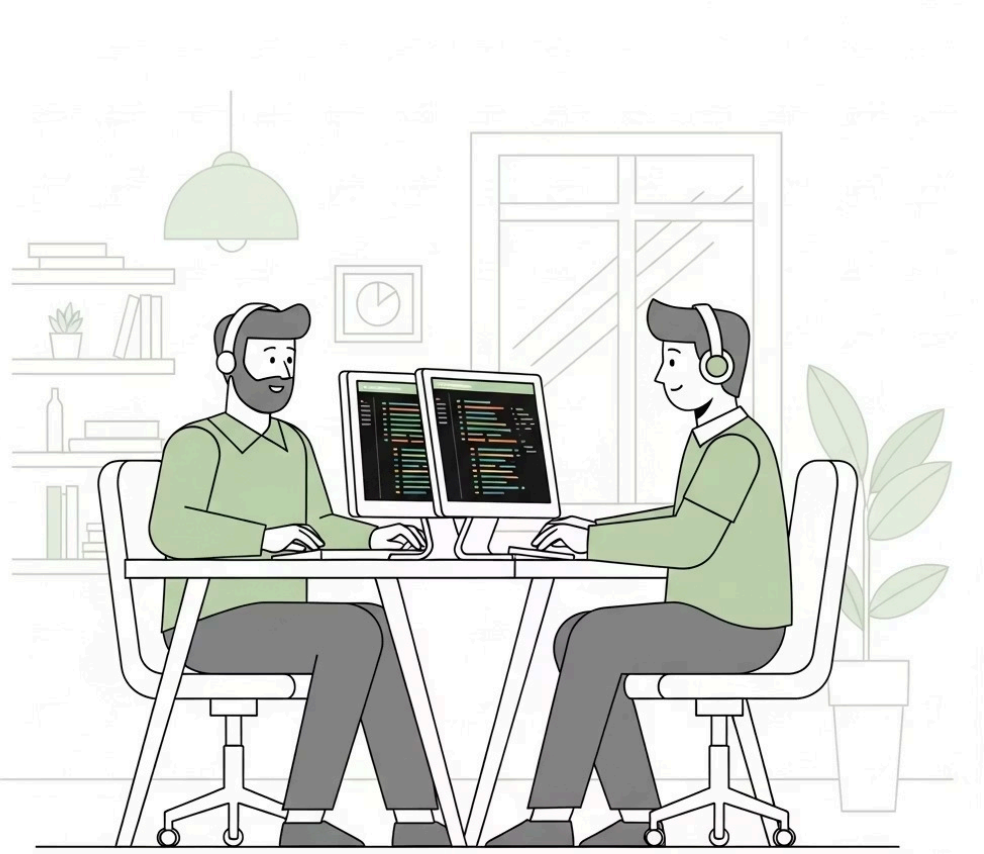
Vantagens

- Redução significativa de bugs no código
- Maior qualidade técnica das soluções implementadas
- Disseminação eficiente de conhecimento pela equipe
- Aprendizado acelerado com a experiência dos pares
- Maior aderência aos padrões de código

Desvantagem

Custo

A alocação de dois desenvolvedores para uma mesma tarefa pode parecer inicialmente mais cara, embora estudos indiquem que a redução de bugs e retrabalho compensa esse investimento.



Contrato com Escopo Aberto

Escopo Fechado (Tradicional)

- Cliente define todos os requisitos no início ("fecha escopo")
- Empresa desenvolvedora se compromete com preço e prazo fixos
- Mudanças geram aditivos contratuais e disputas
- Foco em entregar exatamente o que foi especificado, mesmo que não seja mais o que o cliente precisa

Escopo Aberto (Ágil)

- Escopo definido e refinado a cada iteração
- Pagamento por homem/hora ou por sprint
- Contrato renovado a cada iteração
- Foco em entregar o que realmente agrega valor ao cliente
- Flexibilidade para adaptar às mudanças de necessidades

Contrato com Escopo Aberto

Vantagens

- Privilegia a qualidade do software sobre entregas apressadas
- Cliente não será "enganado" com entregas que tecnicamente cumprem o contrato mas não atendem às necessidades reais
- Possibilidade de mudar de fornecedor se o desempenho não for satisfatório
- Alinhamento de incentivos: a empresa desenvolvedora é recompensada por entregar valor, não por cumprir especificações
- Adaptabilidade às mudanças de mercado e requisitos

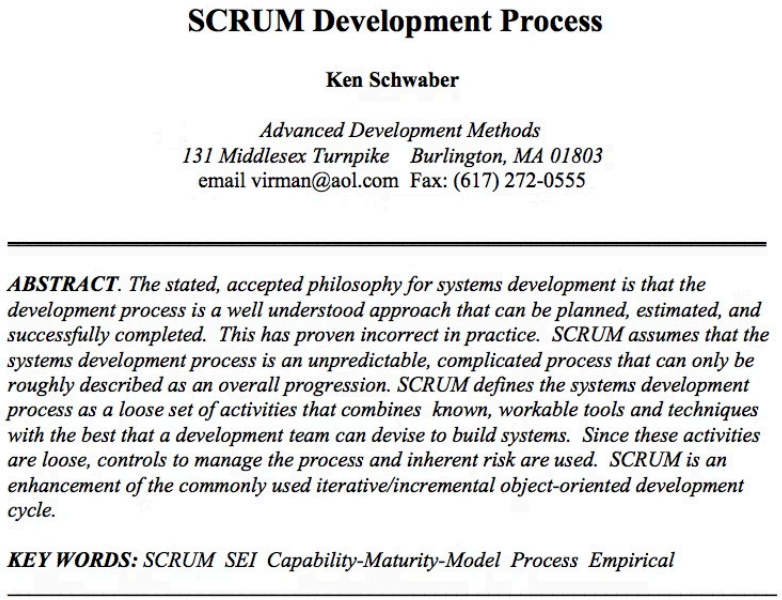
Requisito de Maturidade

Exige maturidade e acompanhamento constante do cliente no processo de desenvolvimento.

Scrum

Uma framework ágil focada em gerenciamento de projetos através de iterações chamadas sprints.

Scrum



SCRUM Development Process

Ken Schwaber

Advanced Development Methods
131 Middlesex Turnpike Burlington, MA 01803
email virman@aol.com Fax: (617) 272-0555

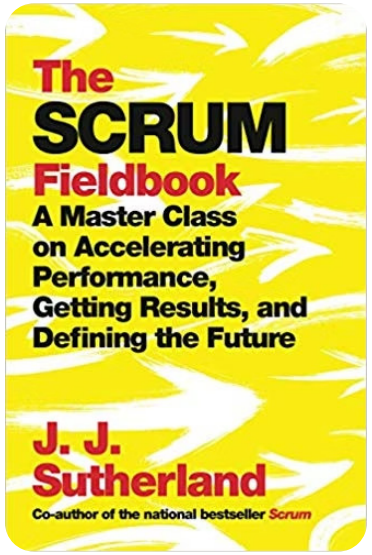
***ABSTRACT**. The stated, accepted philosophy for systems development is that the development process is a well understood approach that can be planned, estimated, and successfully completed. This has proven incorrect in practice. SCRUM assumes that the systems development process is an unpredictable, complicated process that can only be roughly described as an overall progression. SCRUM defines the systems development process as a loose set of activities that combines known, workable tools and techniques with the best that a development team can devise to build systems. Since these activities are loose, controls to manage the process and inherent risk are used. SCRUM is an enhancement of the commonly used iterative/incremental object-oriented development cycle.*

***KEY WORDS:** SCRUM SEI Capability-Maturity-Model Process Empirical*

Origens do Scrum

- Proposto por Jeffrey Sutherland e Ken Schwaber na conferência OOPSLA em 1995
- Inspirado em práticas de desenvolvimento de produtos no Japão
- Nome derivado do rugby, onde um "scrum" é uma formação onde o time trabalha junto para avançar com a bola
- Formalizado no Scrum Guide em 2010
- Atualmente uma das metodologias ágeis mais adotadas no mundo

Scrum



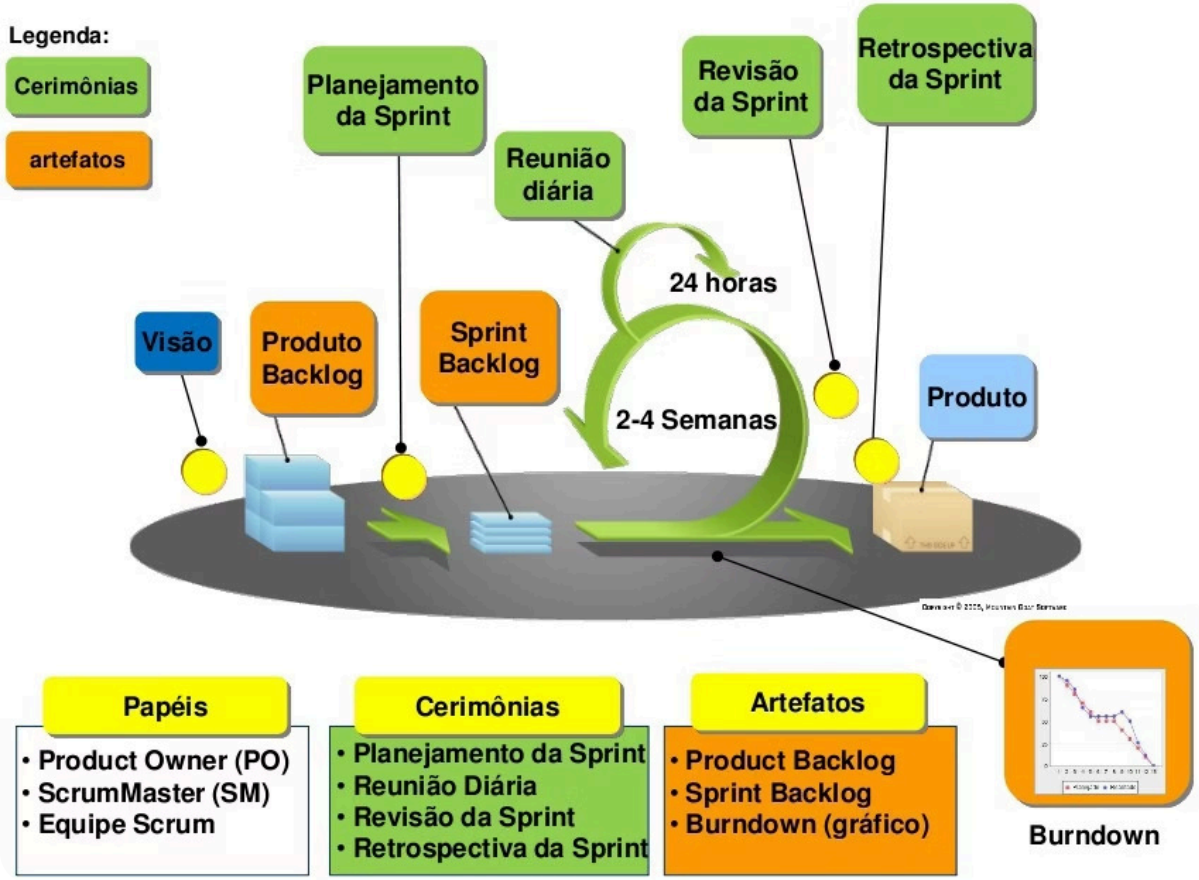
Scrum como Indústria

O Scrum evoluiu para um ecossistema completo de negócios, incluindo:

- Extensa bibliografia com livros, artigos e estudos de caso
- Empresas de consultoria especializadas em implementação
- Sistema de certificações profissionais (Scrum Master, Product Owner)
- Conferências e eventos internacionais
- Ferramentas e softwares específicos para gerenciamento
- Estratégias de marketing e divulgação

Esta comercialização contribuiu tanto para sua popularização quanto para alguns desvios em sua implementação.

Scrum



O framework Scrum define claramente papéis, eventos, artefatos e as regras que os interligam, criando um ciclo contínuo de planejamento, execução e melhoria.

Scrum vs XP

Escopo de aplicação

Scrum pode ser aplicado em diversos tipos de projetos, não apenas software, enquanto XP é específico para desenvolvimento de software.

Estrutura do processo

Scrum define um processo mais estruturado, com eventos, papéis e artefatos bem delimitados, enquanto XP é mais flexível.

Práticas técnicas

XP define práticas específicas de programação (TDD, pair programming), enquanto Scrum não prescreve práticas técnicas.

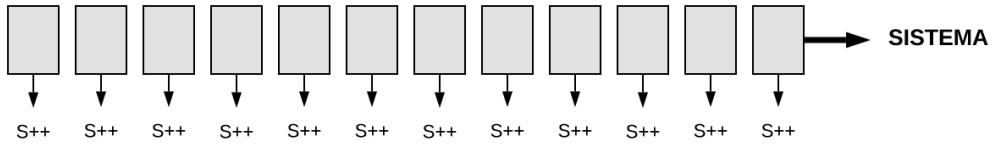
Complementaridade

Na prática, muitas equipes combinam elementos de Scrum e XP, usando o Scrum para gerenciamento e XP para práticas técnicas.

Principal Evento: Sprints

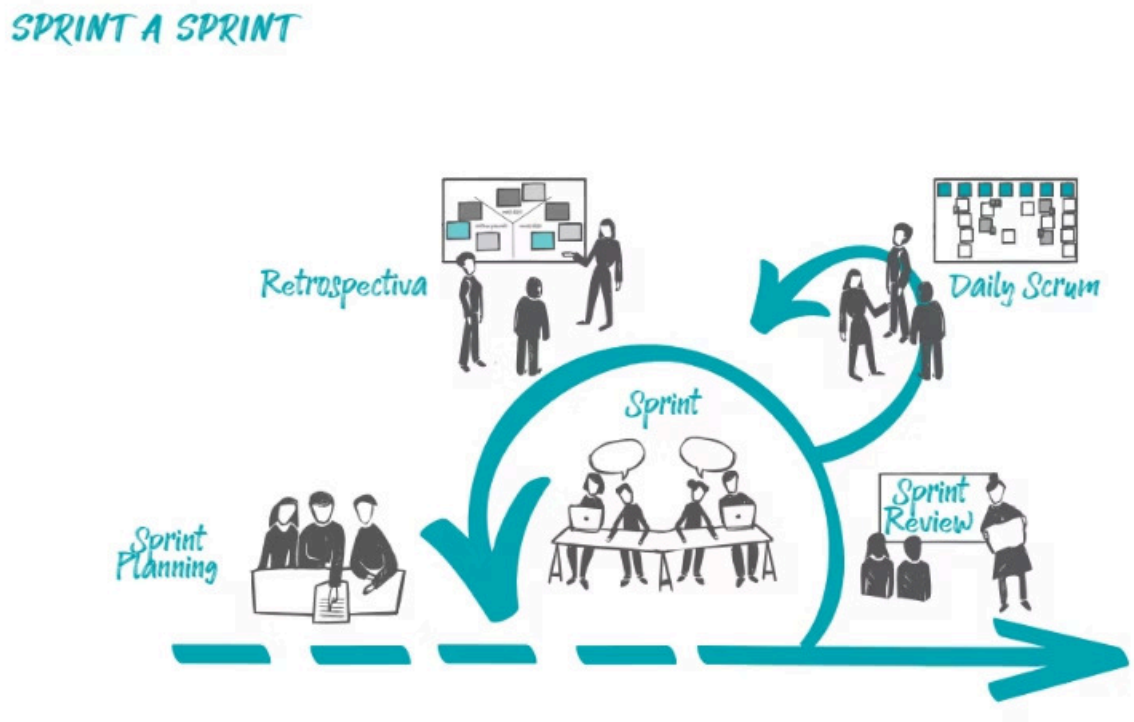
Características das Sprints

- Iterações de desenvolvimento com duração fixa
- Limite máximo de 1 mês, tipicamente 2 semanas (15 dias)
- Tempo consistente para todas as sprints do projeto
- Cada sprint tem início e fim bem definidos
- Uma nova sprint começa imediatamente após a conclusão da anterior



Cada sprint representa um ciclo completo de desenvolvimento, desde o planejamento até a entrega de um incremento de produto funcional.

Principal Evento: Sprints



O ciclo do Scrum inclui eventos específicos dentro de cada sprint: Planejamento da Sprint, Daily Scrums (reuniões diárias), Revisão da Sprint e Retrospectiva da Sprint.

O que se faz em uma sprint?

Implementação de Histórias de Usuário

Durante uma sprint, a equipe se concentra em implementar um conjunto selecionado de histórias do usuário, que são:

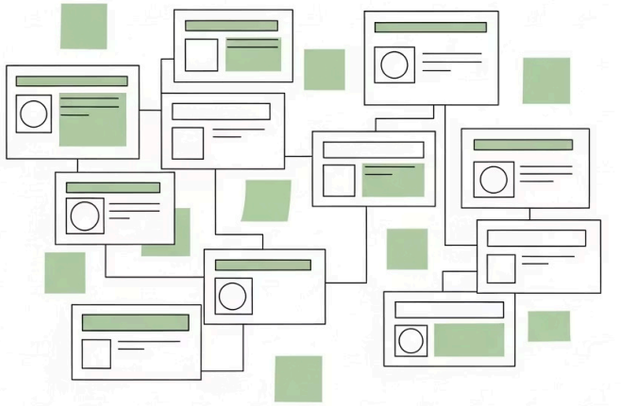
- Descrições curtas e simples de funcionalidades do sistema
- Escritas na perspectiva do usuário (quem, o quê, por quê)
- Pequenas o suficiente para serem implementadas em uma sprint
- Independentes umas das outras, quando possível
- Testáveis e com critérios de aceitação claros

Exemplo: "Como usuário, quero poder fazer perguntas no fórum para que minhas dúvidas sejam esclarecidas pela comunidade."

Postar Pergunta

Um usuário, quando logado no sistema, deve ser capaz de postar perguntas. Como é um site sobre programação, as perguntas podem incluir blocos de código, os quais devem ser apresentados com um layout diferenciado.

Histórias são intencionalmente simples, geralmente escritas em post-its para manter o foco na comunicação e não na documentação extensa.



Quem escreve as Histórias

Product Owner (PO)

O PO é o responsável por escrever e priorizar as histórias de usuário, representando a voz do cliente e stakeholders dentro do time Scrum.

Papel Obrigatório

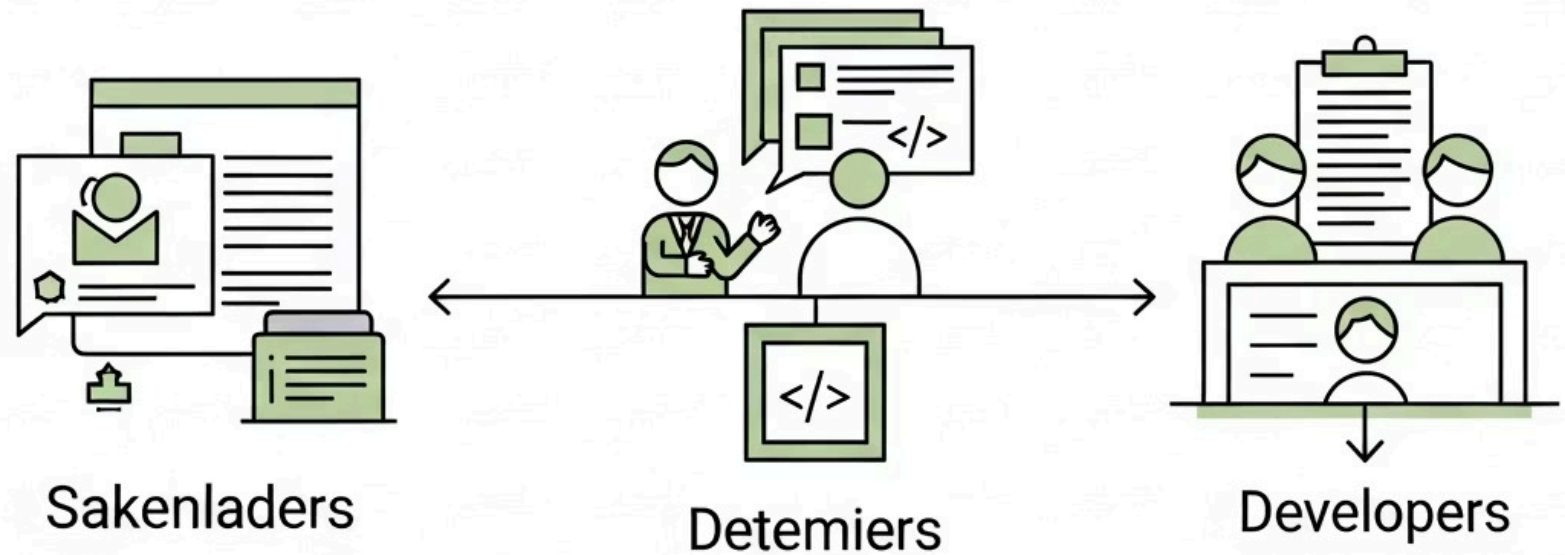
O Product Owner é um dos três papéis essenciais no Scrum, junto com o Scrum Master e o Time de Desenvolvimento.

Especialista no Domínio

O PO deve ter profundo conhecimento do domínio do sistema e das necessidades dos usuários para criar histórias relevantes.

Um bom Product Owner equilibra as necessidades dos stakeholders com as possibilidades técnicas da equipe, priorizando o que gera mais valor para o negócio.

Antes



Stakeholders

Múltiplos interessados com diferentes necessidades e expectativas para o sistema.

Analista de Requisitos

Intermediário que coletava requisitos e produzia documentação extensa.

Linguagem Natural

Documentos formais e detalhados que frequentemente levavam meses ou anos para serem finalizados.

Desenvolvedores

Recebiam documentação pronta e tinham pouco contato com os stakeholders originais.

Hoje (Scrum)

Stakeholders

Comunicam suas necessidades diretamente ao Product Owner.

Product Owner

Traduz necessidades em histórias de usuário e prioriza o backlog.

Desenvolvedores

Interagem diretamente com o PO para esclarecer histórias durante a sprint.

A documentação formal e escrita é substituída por comunicação verbal e direta, privilegiando a clareza e rapidez na transmissão de informações.

Funções do PO

Escrever Histórias dos Usuários

Capturar as necessidades dos stakeholders em formato de histórias claras e concisas.

Explicar Histórias para os Devs

Esclarecer dúvidas e fornecer contexto adicional durante a implementação.

Definir Testes de Aceitação

Estabelecer critérios claros para determinar quando uma história está completa.

Priorizar Histórias

Decidir a ordem de implementação com base no valor de negócio e dependências.

Manter o Backlog do Produto

Gerenciar a lista de histórias, refinando-as e ajustando prioridades continuamente.

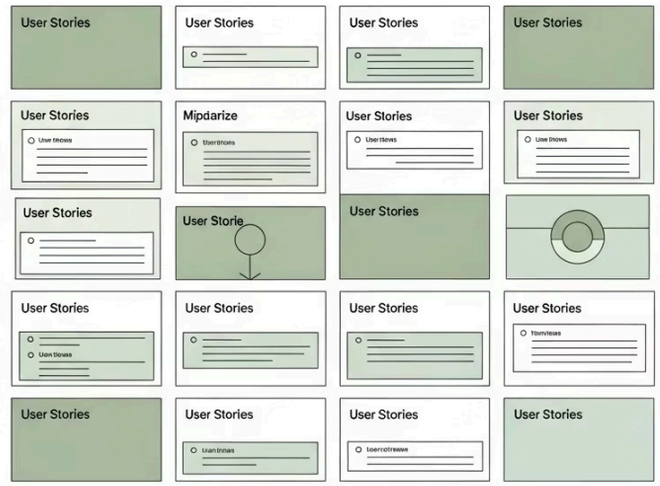
Backlog do Produto

O que é o Backlog do Produto?

Lista ordenada de tudo que é necessário no produto, sendo a única fonte de requisitos para qualquer mudança a ser feita.

Características Principais:

- **Priorizado:** histórias no topo têm maior prioridade e serão implementadas primeiro
- **Dinâmico:** constantemente atualizado conforme o produto evolui e o mercado muda
- **Detalhamento progressivo:** itens no topo são mais detalhados que os do final da lista
- **Transparente:** visível para toda a equipe e stakeholders
- **Nunca completo:** evolui durante toda a vida do produto



Product Backlog

O Product Owner é o responsável final pelo Backlog do Produto, incluindo seu conteúdo, disponibilidade e ordenação.

Resumindo

Sprint (Evento)

Iteração de desenvolvimento com duração fixa, tipicamente de 2 semanas, onde um incremento do produto é criado.

PO e Devs (Papéis)

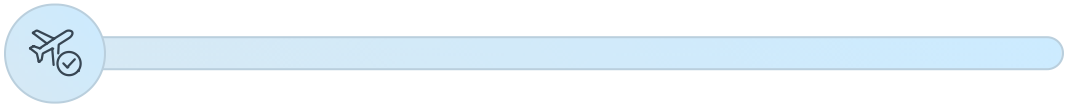
Product Owner representa as necessidades do cliente; Desenvolvedores implementam as funcionalidades.

Backlog do Produto (Artefato)

Lista priorizada de todas as funcionalidades desejadas para o produto.

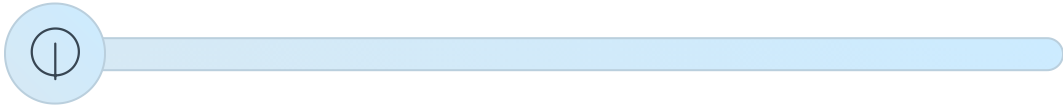
Estes três elementos formam a base do framework Scrum, estabelecendo o quê será feito, quem o fará e quando será realizado.

Quais histórias vão entrar no próximo sprint?



Início da Sprint

A decisão sobre quais histórias serão implementadas é tomada no início de cada sprint.

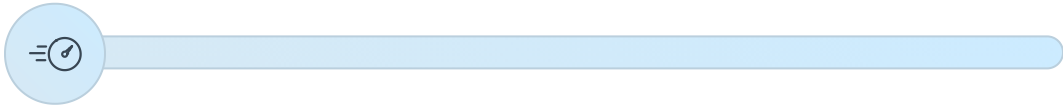


Reunião de Planejamento

Durante a reunião de planejamento do sprint, o PO apresenta as histórias de maior prioridade do backlog.

Proposta do PO

O Product Owner propõe as histórias que gostaria de ver implementadas, baseando-se na prioridade para o negócio.



Decisão dos Desenvolvedores

Os desenvolvedores avaliam sua capacidade (velocidade) e decidem se conseguem implementar todas as histórias propostas.

- As decisões são tomadas colaborativamente, não por imposição

Planejamento da Sprint

1ª Parte: O QUÊ

Definem-se as histórias que serão implementadas na sprint, selecionadas do topo do backlog do produto.

- PO apresenta as histórias prioritárias
- Time discute e esclarece dúvidas
- Desenvolvedores determinam quantidade viável

2ª Parte: COMO

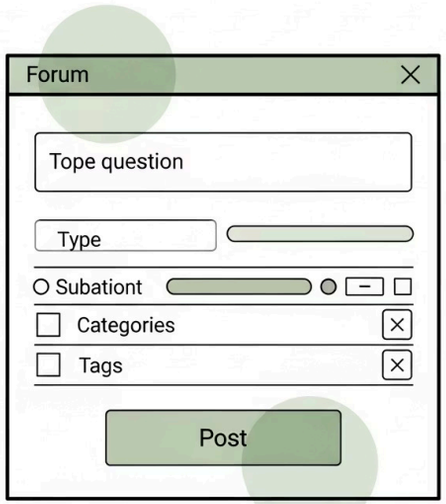
As histórias selecionadas são decompostas em tarefas técnicas e atribuídas aos membros da equipe.

- Histórias são quebradas em tarefas menores
- Estimativas de esforço são realizadas
- Tarefas são alocadas para desenvolvedores

Planejamento da Sprint - Exemplo

História: Postar Perguntas

"Como usuário, quero poder postar perguntas no fórum para que outros usuários possam me ajudar com minhas dúvidas."



Tarefas:

1. Projetar e testar a interface Web, incluindo layout, CSS templates, etc.
2. Instalar banco de dados, projetar e criar tabelas.
3. Implementar a camada de acesso a dados.
4. Implementar camada de controle, com operações para cadastrar, remover e atualizar perguntas.
5. Implementar interface Web.
6. Criar testes automatizados para validar funcionalidade.
7. Documentar API para integração com outros componentes.

Cada tarefa será atribuída a um ou mais desenvolvedores, com estimativas de tempo para conclusão.

Backlog da Sprint

Definição

Lista completa de tarefas técnicas que compõem o trabalho necessário para transformar as histórias selecionadas em um incremento funcional do produto.

Componentes

- Histórias de usuário selecionadas do backlog do produto
- Tarefas técnicas derivadas de cada história
- Desenvolvedores responsáveis por cada tarefa
- Estimativas de duração para cada tarefa
- Status atual de cada tarefa (a fazer, em andamento, concluída)

Evolução

O backlog da sprint é atualizado diariamente para refletir o progresso da equipe, com tarefas sendo concluídas e eventualmente novas tarefas sendo identificadas.

Sprint está pronta para começar!

Com o backlog da sprint definido, as tarefas atribuídas e o objetivo claro, a equipe está preparada para iniciar o trabalho de desenvolvimento.

A disciplina

Vamos entender como aplicaremos os conceitos de métodos ágeis em nossa disciplina, através de projetos práticos e avaliações.

Projeto

Equipe

Grupos de até 5 pessoas trabalhando como uma equipe ágil

Tema

Livre escolha, permitindo que cada equipe explore suas áreas de interesse

Entregas

- Escopo (Épicos, features, User Stories)
- Documento de Arquitetura
- Backlog do produto/sprints
- Protótipo de baixa/alta fidelidade
- Gitpage da Solução

O projeto será desenvolvido ao longo do semestre aplicando os conceitos de metodologias ágeis estudados na disciplina.

Avaliação

Três categorias de avaliação: "iniciante", "saudável", "maduro" e três aspectos: "práticas", "Equipe" e "Projeto"

Artefatos (70%)

- Documento de Escopo do projeto (15%)
- Planejamento/Comunicação Interna e Externa (10%)
- Documento de Arquitetura do Projeto (15%)
- Especificação das histórias de usuários (20%)
- Configuração do repositório (10%)
- Protótipo (10%)

Práticas (30%)

- Pareamento
- Produtividade
- Participação nos rituais
- Desempenho (commits)

Material de Apoio

Livros Recomendados

- RUBIN, K. S. Essential Scrum: A Practical Guide to the Most Popular Agile Process. Addison Wesley, 2012.
- BRECHNER, E. Agile Project Management with Kanban. Microsoft Press, 2015.
- Beck, K., Programação Extrema (XP) Explicada, 1st ed. Bookman, 2004.

Recursos Online

- **Engenharia de Software Moderna - Capítulo 2**
- Scrum Guide (disponível em português)
- Artigos e vídeos recomendados pela professora

O material de apoio servirá como referência para aprofundar os conceitos discutidos em aula e auxiliar no desenvolvimento do projeto.

Vamos colocar em prática os conceitos estudados através de exercícios e atividades práticas.

Onboarding

Contato equipe

Estabelecer canais de comunicação entre membros da equipe e com a professora.

Canais de comunicação

Definir plataformas para comunicação síncrona e assíncrona (Slack, Discord, WhatsApp, etc.).

Dias/horários rituais

Agendar reuniões recorrentes para os rituais ágeis (planejamento, daily, revisão, retrospectiva).

Quadro de conhecimento

Mapear habilidades e experiências de cada membro da equipe para facilitar a distribuição de tarefas.

Configuração git

Preparar repositório, definir convenções de commits e fluxo de trabalho no GitHub.

Definição papéis/atribuições

Estabelecer responsabilidades iniciais (PO, SM, Dev) e rodízio se aplicável.

Roadmap do projeto

Definir o que estudar, tecnologias a explorar e visão geral do desenvolvimento.

Planejamento da Semana

Estabelecer metas e tarefas para a primeira semana de trabalho da equipe.