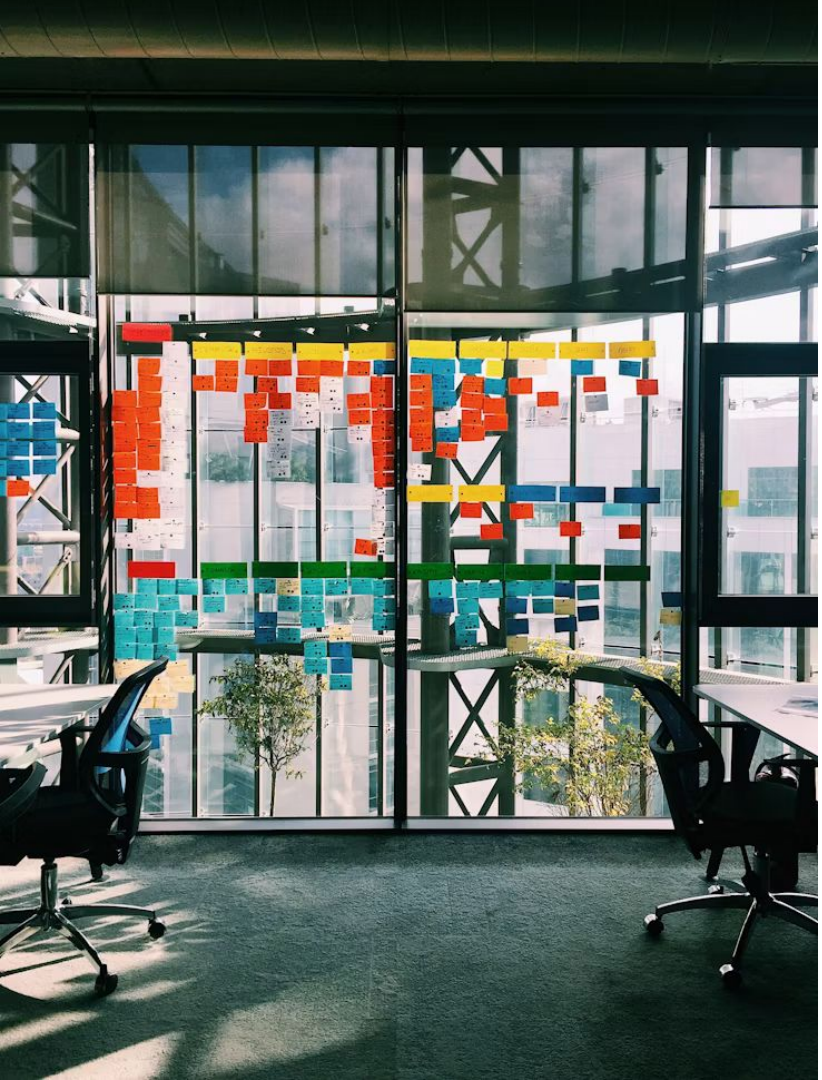


5954014 - Engenharia de Software

Aula 04 - Metodologias Ágeis

Prof. Dr. Denis Martins

DCM | FFCLRP | USP

Objetivos de Aprendizagem

01

Explicar

O contexto que levou ao surgimento dos métodos ágeis e suas diferenças em relação às abordagens dirigidas por planejamento.

02

Interpretar

Os quatro valores e os principais princípios do Manifesto Ágil em situações de desenvolvimento de software.

03

Avaliar

Quando abordagens predominantemente ágeis, preditivas ou híbridas são mais adequadas.

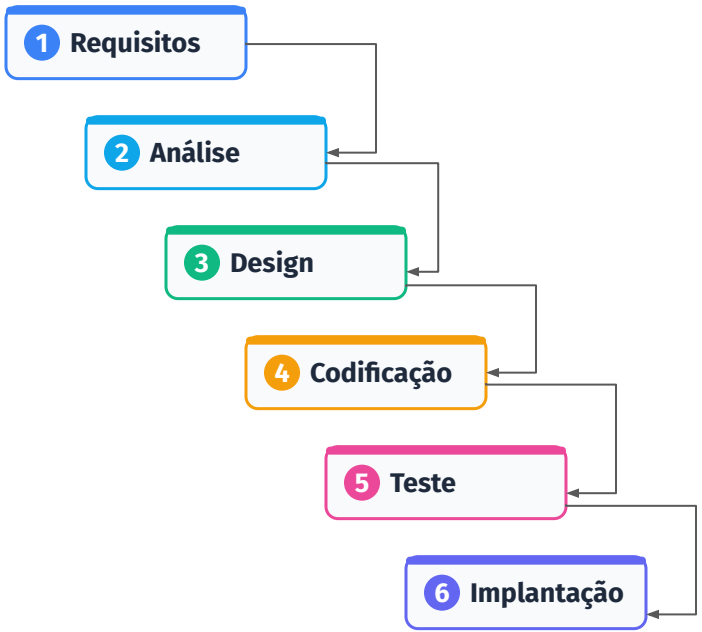
04

Diferenciar

A essência da agilidade em relação à improvisação ou à ausência de planejamento estruturado de projeto.

Revisando: Processo no Modelo Cascata (Waterfall)

Modelo tradicional nas engenharias: O resultado de uma fase é a entrada da próxima



01. Requisitos: Definição do escopo, levantamento de necessidades com os stakeholders e documentação do que o sistema deve fazer.

02. Análise: Modelagem do sistema, refinamento de regras de negócio, casos de uso e estudo detalhado de viabilidade técnica do projeto.

03. Design: Desenho da arquitetura geral do software, definição de esquemas de banco de dados e especificação de interfaces de usuário (UI/UX).

04. Codificação: Fase de desenvolvimento ativo onde programadores implementam as funcionalidades em código-fonte com base nos designs.

05. Teste: Validação da qualidade geral, execução de rotinas de verificação de bugs, testes de integração e homologação do software para uso.

06. Implantação: Lançamento oficial da aplicação estável no ambiente produtivo e entrega final de valor para o cliente e usuários finais.

Motivação

Cenário de Negócio

Uma empresa está desenvolvendo um **sistema de vendas online**. Escopo, prioridades e cronograma foram definidos no início (modelo cascata).

Após três meses de desenvolvimento:

- **Dispositivos Móveis:** Clientes realizam a maioria das compras por celulares.
- **Perda de Foco:** Funcionalidade antes prioritária perdeu a importância.
- **Concorrência:** Concorrente lança funcionalidade altamente estratégica.
- **Mudança de Rumo:** Testes revelam que decisões iniciais devem ser revistas.

Questões para Discussão

1. Qual é o principal problema em seguir **exatamente** o planejamento inicial?
2. O que acontece quando uma funcionalidade secundária se torna **urgente**?
3. Qual é o custo de descobrir **somente no final** que os requisitos mudaram?
4. Seria possível reorganizar as prioridades e entregar valor **mais cedo**?
5. Que características de processo ajudariam a equipe a lidar com essa **dinâmica**?

Motivação (cont.)

MUDANÇA

Integração com PIX

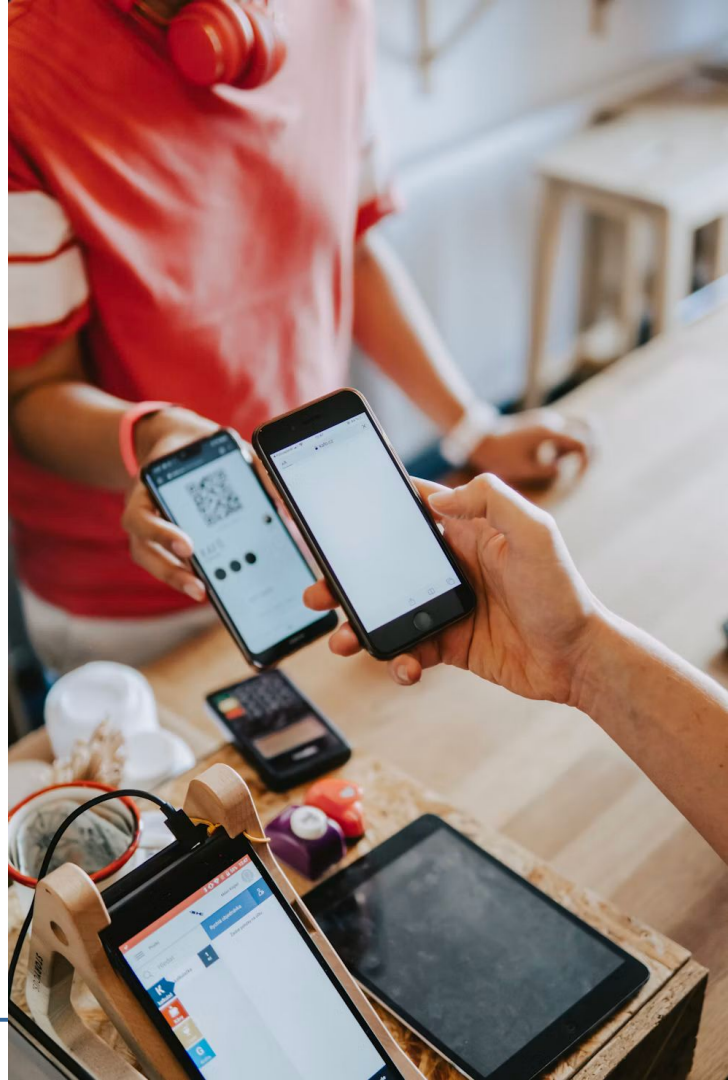
A empresa acaba de informar que a integração com PIX precisa estar disponível em **quatro semanas** para uma importante campanha comercial.

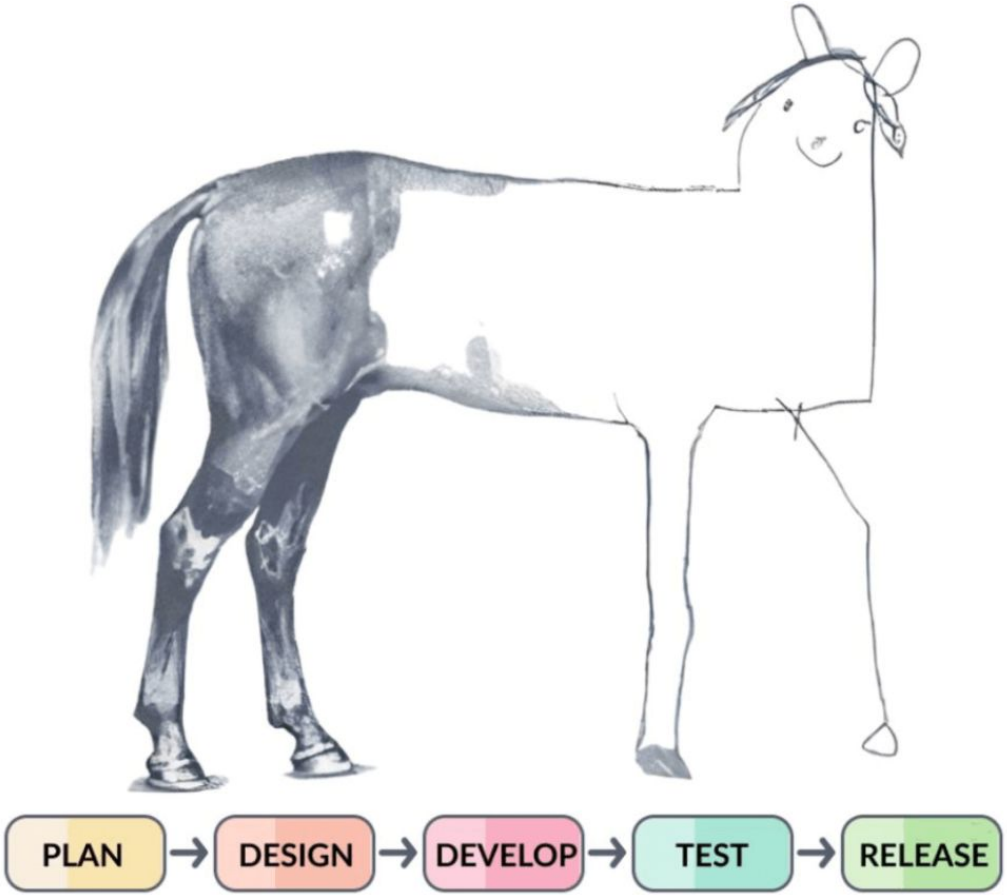
DISCUSSÃO

Impacto no Processo

Com base nessa mudança inesperada, discuta:

- O processo atual **facilita** ou **dificulta** essa mudança de prioridade?
- O que a equipe precisaria fazer para se adaptar?





Fonte da Imagem: https://captainpinguin.com/agile-vs-waterfall/

Processos rígidos e puramente **"preditivos"** tentam prever e predeterminar **todos os detalhes** do desenvolvimento logo no início.

Essa rigidez gera barreiras que comprometem diretamente o sucesso e a relevância do produto final.

Em um mundo extremamente dinâmico, somente planejar não funciona.

Motivação (cont.)

O problema é utilizar um processo rígido no qual:

PROBLEMA 01

Decisões Antecipadas

Grande parte das decisões é tomada antecipadamente, reduzindo a flexibilidade futura.

PROBLEMA 02

Retorno a Etapas

Mudanças posteriores podem exigir retorno custoso a etapas que já haviam sido concluídas.

PROBLEMA 03

Rigidez de Prioridades

As prioridades tornam-se muito mais difíceis de reorganizar durante o ciclo de desenvolvimento.

PROBLEMA 04

Feedback Tardio

O feedback sobre o produto pode ocorrer tardiamente, elevando o risco de desalinhamento com o mercado.

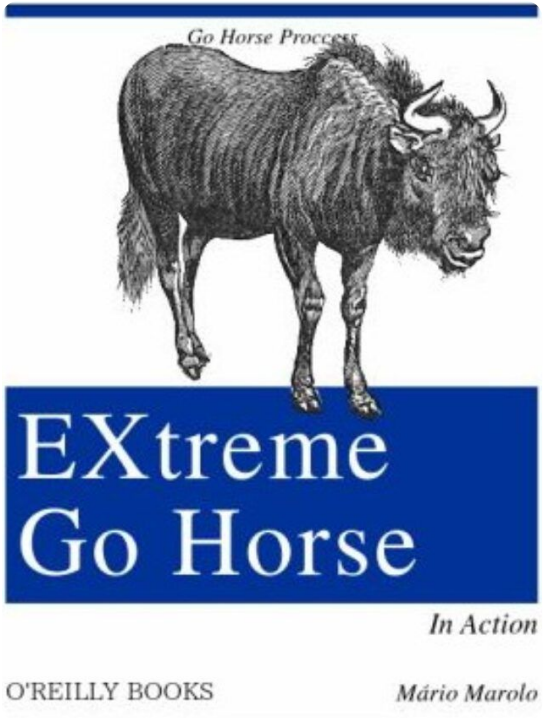
Qual a solução? Deixar de planejar?


GO HORSE
PLOCESS

“Modelo” eXtreme Go Horse (XGH) | Uma **piada**

1- Pensou, não é XGH.

XGH não pensa, faz a primeira coisa que vem à mente. Não existe segunda opção, a única opção é a mais rápida.



Fonte da Imagem: gohorse.com.br

Leia mais em: https://hackernoon.com/you-might-be-using-extreme-go-horse-process-and-not-even-know-it

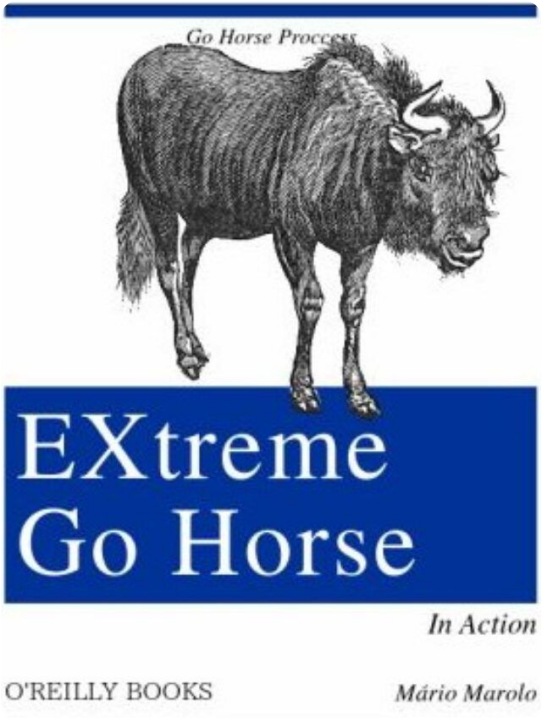
“Modelo” eXtreme Go Horse (XGH) | Uma **piada**

1- Pensou, não é XGH.

XGH não pensa, faz a primeira coisa que vem à mente. Não existe segunda opção, a única opção é a mais rápida.

2- Existem 3 formas de se resolver um problema...

A correta, a errada e a XGH, que é igual à errada, só que mais rápida. XGH é mais rápido que qualquer metodologia que você conhece.



Fonte da Imagem: gohorse.com.br

Leia mais em: https://hackernoon.com/you-might-be-using-extreme-go-horse-process-and-not-even-know-it

“Modelo” eXtreme Go Horse (XGH) | Uma **piada**

1- Pensou, não é XGH.

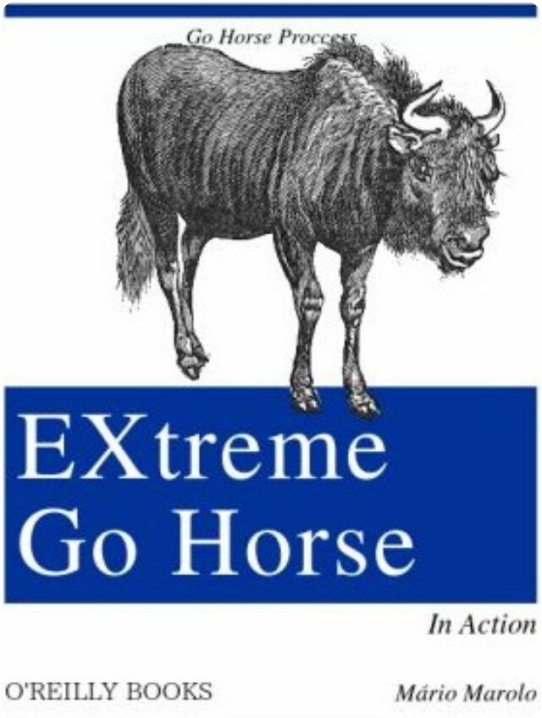
XGH não pensa, faz a primeira coisa que vem à mente. Não existe segunda opção, a única opção é a mais rápida.

2- Existem 3 formas de se resolver um problema...

A correta, a errada e a XGH, que é igual à errada, só que mais rápida. XGH é mais rápido que qualquer metodologia que você conhece.

3- Quanto mais XGH você faz, mais precisará fazer.

Para cada problema resolvido usando XGH, mais uns 7 são criados. Mas todos eles serão resolvidos da forma XGH. XGH tende ao infinito.



Fonte da Imagem: gohorse.com.br

Leia mais em: https://pt.stackoverflow.com/questions/164124/o-que-%C3%A9-xgh-extreme-go-horse

Sem um processo (mesmo que simples), um projeto de software enfrenta sérios riscos estratégicos.

Processos claros ajudam profissionais a tomar **consciência plena das tarefas** e dos resultados exatos que se esperam de sua atuação.

Como organizar o desenvolvimento de software quando sabemos que requisitos e prioridades provavelmente mudarão ao longo do projeto?

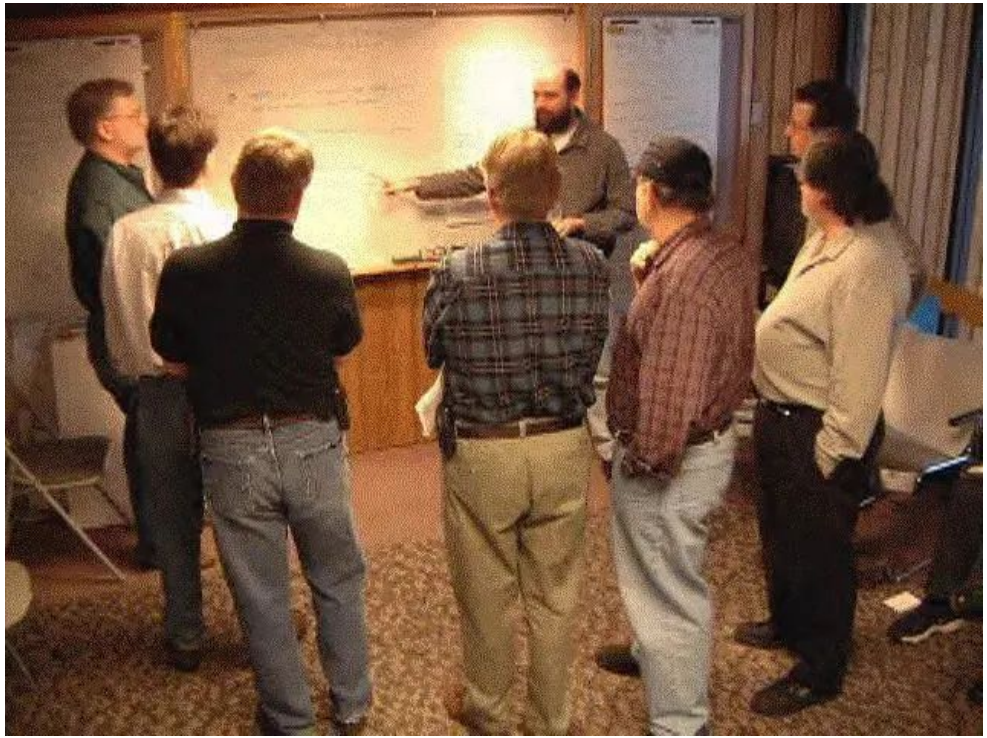
Movimento Ágil

Novo Modelo Processo

- **Indivíduos e interações** mais que processos e ferramentas, garantindo alinhamento e rápida comunicação.
- **Software em funcionamento** mais que documentação abrangente, entregando valor real de forma contínua.
- **Responder a mudanças** mais que seguir um plano estático, aceitando que o escopo evolui no tempo.
-

Manifesto Ágil (2001)

- **17 especialistas** se reuniram em Snowbird, Utah (EUA) para discutir alternativas leves aos processos tradicionais e compartilhar o que já funcionava na prática.
- O documento que reúne os princípios e práticas desta metodologia de desenvolvimento.



Fonte da Imagem: https://agile-lounge.com/18-years-of-agile-manifesto-for-software-development/

Participantes

Metodologias e Conceitos

Os 17 autores do Manifesto Ágil trouxeram consigo a bagagem e os fundamentos de diversas práticas e tecnologias:

Extreme Programming (XP)

SCRUM

UML (Unified Modeling Language)

Ruby

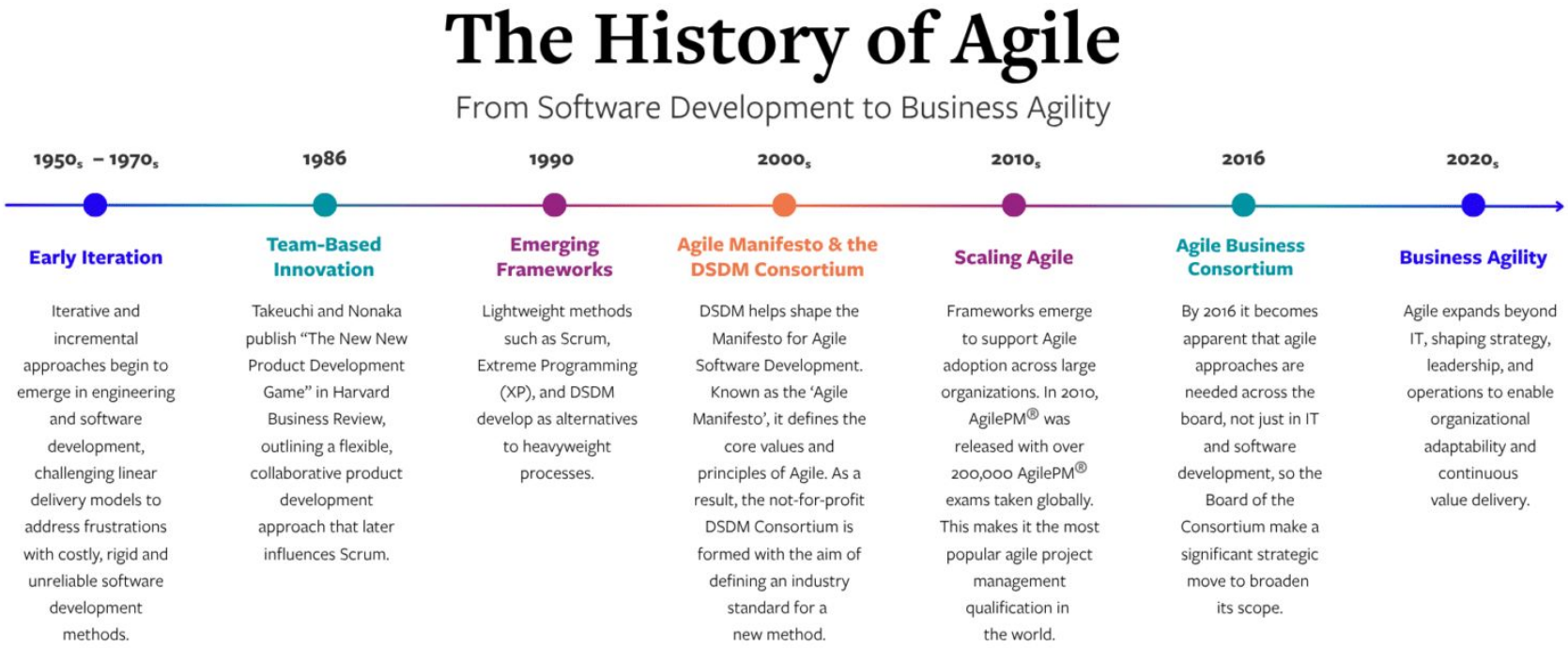
Adaptive Software Development

Princípios SOLID

Análise Orientada a Objetos

Fonte da Imagem: https://agile-lounge.com/18-years-of-agile-manifesto-for-software-development/

Linha do Tempo



Fonte da Imagem: https://www.agilebusiness.org/resource/what-is-agile/

Manifesto Ágil: Princípios

4 Valores Fundamentais

01. Indivíduos e interações mais que processos e ferramentas

02. Software funcionando mais que documentação abrangente

03. Colaboração com o cliente mais que negociação de contratos

04. Responder a mudanças mais que seguir um plano

Entrega de Valor

- Entregar frequentemente
- Buscar feedback
- Priorizar valor

Adaptação

- Aceitar mudanças
- Trabalhar de forma incremental
- Aprender no processo

Pessoas

- Colaboração ativa
- Comunicação diária
- Equipes motivadas
- Auto-organização

Excelência Técnica

- Qualidade e simplicidade
- Melhoria contínua
- Ritmo sustentável

Ser ágil não significa trabalhar sem planejamento ou de forma caótica, mas ter a disciplina de planejar estruturadamente, entregar em ciclos curtos, obter feedback real e adaptar continuamente o rumo do produto.

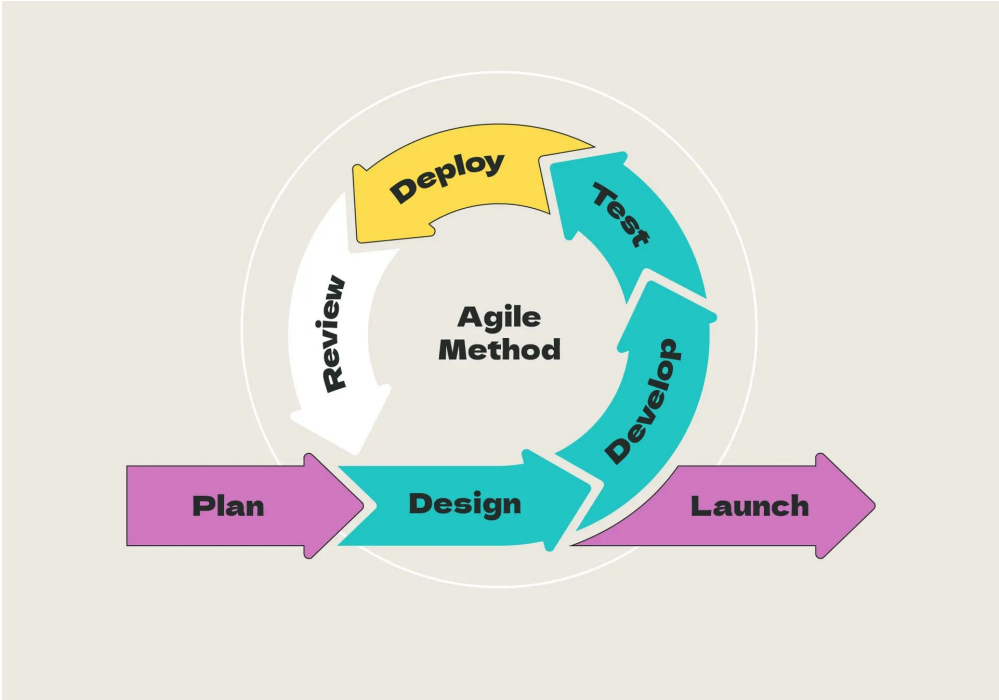
Fundamento do Modelo Ágil

Desenvolvimento Iterativo
O sistema evolui em ciclos curtos

Feedback Frequente
Usuários e stakeholders avaliam entregas parciais de forma constante

Adaptação a Mudanças
Requisitos e prioridades mudam de forma natural ao longo do projeto

Excelência Técnica
Testes automatizados, integração contínua, refatoração constante

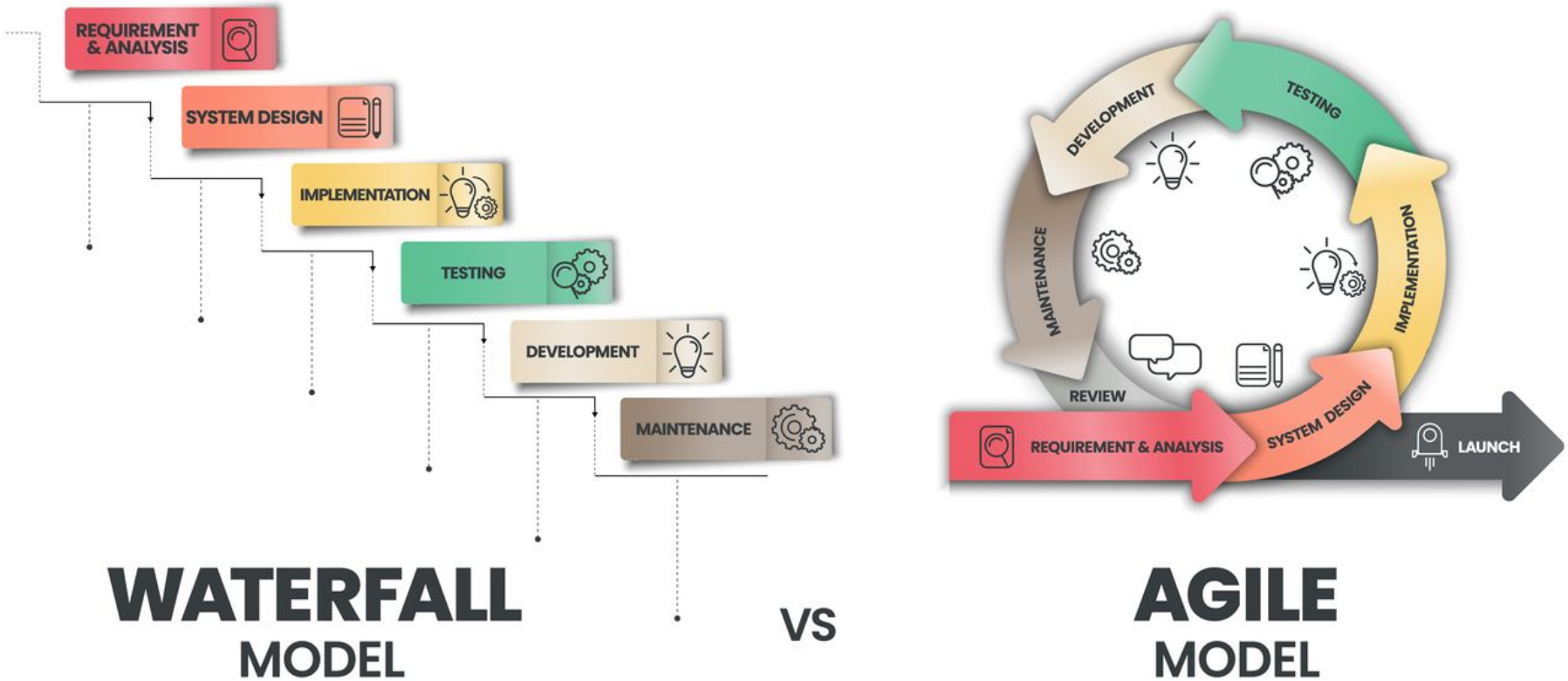


Fonte da Imagem: agilie.com

Ideia Central

Ser ágil não significa trabalhar sem planejamento ou de forma caótica, mas ter a disciplina de planejar estruturadamente, entregar em ciclos curtos, obter feedback real e adaptar continuamente o rumo do produto.

Cascata x Ágil



Fonte da Imagem: https://jdmeier.com/wp-content/uploads/2023/03/Waterfall-vs.-Agile.jpg

Atividade: Análise de Posturas Ágeis

Analisem as situações apresentadas e indiquem se representam ou não uma postura compatível com os valores e princípios ágeis.

Instruções:

Para cada uma das 5 situações ao lado:

1. Classifique como **Compatível** ou **Não Compatível**.
2. Identifique um valor ou princípio relacionado.
3. Escreva uma justificativa em uma única frase.

Situação A: “Nossa equipe é ágil, portanto não produzimos documentação de nenhum tipo.”

Situação B: “O cliente solicitou uma mudança. A equipe avaliou seu impacto e repriorizou o backlog de trabalho.”

Situação C: “A equipe somente apresentará o sistema aos usuários após todas as funcionalidades estarem 100% concluídas.”

Situação D: “A equipe entrega pequenas funcionalidades frequentemente e utiliza o feedback para decidir o que fazer depois.”

Situação E: “Como somos uma equipe ágil, não fazemos estimativas nem planejamento de atividades.”

Quando usar métodos ágeis?

Nenhuma abordagem é sempre melhor. A escolha depende de diversos fatores do projeto:

Dinâmica do Escopo e Processo

Estabilidade dos requisitos: Quanto menor a estabilidade, mais indicado o uso de métodos ágeis.

Frequência de mudanças: Ambientes de alta mudança demandam ciclos de feedback curtos.

Necessidade de documentação: Projetos que exigem menos formalidade documental estrita.

Regulamentação: Setores altamente regulados podem exigir abordagens mais híbridas ou preditivas.

Contexto, Equipe e Risco

Tamanho e experiência da equipe: Equipes menores e seniores se adaptam melhor ao ágil puro.

Participação do cliente: Métodos ágeis exigem colaboração frequente e ativa do cliente.

Criticidade do sistema: Sistemas de missão crítica tendem a demandar mais controles rígidos.

Risco do projeto: O gerenciamento iterativo mitiga riscos de entrega de forma progressiva.

Exercício: Qual abordagem escolher?

Instruções

Trabalhando em grupos, analisem as situações ao lado e escolham a abordagem mais adequada:

- **Tradicional (Cascata)**
- **Predominantemente Ágil**
- **Híbrida**

Não se esqueçam de **justificar a decisão** *tomada para cada caso.*

Cenário A: Startup

Aplicação inédita, mercado incerto, feedback constante de usuários.

Cenário B: Sistema Crítico

Software embarcado, regulamentação rigorosa, documentação obrigatória.

Cenário C: Sistema Administrativo

Regras conhecidas, mas algumas funcionalidades precisam ser testadas gradualmente.

How Extreme Programming (XP) Works

XP combines values, planning discipline, and engineering practices to create a fast feedback loop that improves both delivery speed and software quality.

Core values

Values create the cultural foundation for how the team works together.

- Communication
- Simplicity
- Feedback
- Courage
- Respect

Values in action

- **Communication:** real-time collaboration removes ambiguity
- Simplicity: build only what is needed now (YAGNI)
- **Feedback:** tests, demos, and customer
- **Courage:** [illegible]
- **Respect:** healthy collaboration sustains performance

Key practices

Planning and engineering practices reinforce one another.

A. Planning & flow:

- Planning Game
- User Stories
- Short Iterations
- On-Site Customer / Product Owner
- Spikes

B. Engineering discipline

- Collective Code Ownership
- Coding Standards
- Acceptance Testing (ATDD)
- Sustainable Pace

Why XP works

- Business value and customer priorities drive planning
- Team plans with real delivery capacity, [illegible] speculative forecasts
- [illegible] practices [illegible] slow [illegible]
- [illegible] and acceptance [illegible] create rapid learning

Extreme Programming (XP)

How the pieces fit together

1. **Plan** — priorities + user stories + iteration goals
2. **Build** — pair, code, refactor, integrate
3. **Verify** — TDD, CI, acceptance tests, demos
4. **Learn** — feedback updates next priorities

XP creates a tight plan-build-verify-learn cycle that keeps teams responsive without sacrificing quality.

Project management lens

XP uses lightweight planning, short iterations, visible progress, and customer prioritization.

Engineering backbone

TDD, CI, pairing, refactoring, and simple design make frequent delivery safe.

Operating insight

Quality is the enabler of speed: disciplined engineering reduces rework and increases learning velocity.

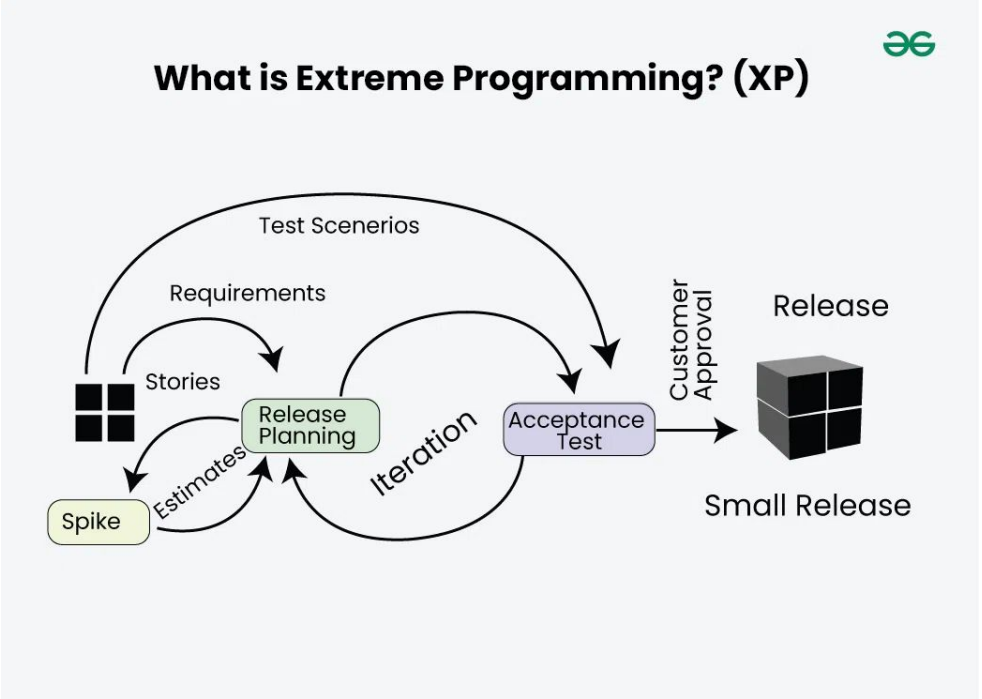
Extreme Programming (XP)

Método ágil de desenvolvimento de software proposto por Kent Beck, focado em ciclos rápidos e adaptabilidade.

Pilares de Execução

Práticas essenciais que guiam o time rumo à excelência técnica:

- **Ciclos curtos e iterativos:** Entregas rápidas de valor.
- **Desenvolvimento incremental:** Evolução constante do produto.
- **Design incremental:** Refatoração e design limpo sob demanda.
- **Planejamento progressivo:** Adaptação contínua aos novos cenários.
- **Times pequenos:** Alta colaboração e comunicação facilitada.
- **Menor ênfase em documentação extensa:** Foco em software funcional.



Fonte da Imagem: geeksforgeeks.org

Extreme Programming (XP)

Como valores, princípios e práticas se articulam para sustentar o desenvolvimento ágil

Valores

Orientam a cultura, o comportamento e as decisões fundamentais da equipe no dia a dia do projeto.

→

Princípios

Conectam os valores abstratos a decisões concretas, traduzindo a filosofia de desenvolvimento em critérios objetivos.

→

Práticas

Transformam os princípios teóricos em ações práticas de engenharia e desenvolvimento de software.

Importante

Adotar apenas as práticas sem compreender os valores correspondentes pode levar a uma aplicação superficial e ineficiente do XP.

Princípios de XP

Diretrizes essenciais que traduzem os valores do Extreme Programming em decisões práticas de engenharia

Humanidade

Pessoas são o principal recurso de projetos de software. Seus limites e necessidades devem ser respeitados.

Economicidade

O software deve sempre produzir valor financeiro e operacional para quem financia e patrocina o projeto.

Benefícios Mútuos

Decisões tomadas no dia a dia devem necessariamente beneficiar as diferentes partes e stakeholders envolvidos.

Melhorias Contínuas

O produto, o design do código e o processo de trabalho evoluem continuamente por meio de ciclos frequentes.

Falhas Acontecem

Erros e falhas devem ser encarados como oportunidades de aprendizado coletivo, e nunca ocultados da equipe.

Baby Steps

Pequenos avanços devidamente validados e testados são muito superiores a grandes mudanças abruptas.

Responsabilidade

Quem assume individualmente ou em dupla uma tarefa técnica também assume o compromisso com sua qualidade.

Práticas de XP

XP combina **feedback rápido, simplicidade, colaboração e qualidade técnica** por meio de práticas concretas de desenvolvimento.

Programação em Pares: Dois desenvolvedores trabalham juntos na mesma tarefa, o que favorece a revisão contínua e o compartilhamento de conhecimento.

TDD (Test-Driven Development): Prática de escrever o teste antes do código, ajudando a manter a qualidade e a reduzir significativamente as regressões.

Refatoração: Melhorar continuamente a estrutura interna do código, garantindo que o comportamento externo do sistema seja totalmente preservado.

Integração Contínua: Integrar alterações de código frequentemente ao repositório principal para permitir a detecção precoce de conflitos e erros.

Design Simples: Implementar rigorosamente a solução mais simples e direta que atenda perfeitamente às necessidades de negócios atuais.

Fonte da Imagem: https://artia.com/blog/extreme-programming/

Quando Utilizar XP?

XP é especialmente adequado para cenários de alta incerteza e necessidade de rápida resposta:

Requisitos Vagos

Requisitos são **vagos ou indefinidos** inicialmente no projeto.

Mudanças Frequentes

Requisitos **mudam com frequência** no decorrer do ciclo de desenvolvimento.

Feedback Rápido

Feedback rápido com o cliente é considerado de extrema importância.

> Não define um passo a passo rígido: é organizado de forma prática em **valores, princípios e práticas**.

Histórias de Usuário em XP

XP recomenda que o time tenha um **representante dos clientes** para guiar o desenvolvimento.

Responsabilidades do Representante

- **Conhecer o domínio** do problema em profundidade.
- **Escrever** as histórias de usuário de forma clara.
- **Definir prioridades** de entrega para o negócio.
- **Estar disponível** para esclarecer dúvidas do time diariamente.
- **Validar** as histórias implementadas e dar feedback.

Conceito e Estrutura

- **Descrições curtas** e objetivas de funcionalidades de valor.
- Escritas na **linguagem do usuário**, sem termos puramente técnicos.
- Representam um **lembrete para uma conversa** posterior e detalhamento entre cliente e desenvolvedores.
- O formato geral é: “Como um <papel>, eu quero <meta/desejo> de modo que <benefício>”

STORY POINTS

Tamanho e Esforço Relativo

- Medem o esforço **relativo** de cada história de usuário de forma comparativa.
- **Não representam** necessariamente horas de trabalho bruto.
- Permitem abstrair a complexidade de forma mais eficiente que métricas absolutas.

Escala de Estimativa Comum:

Planning Pocker: discussão para compreender por que as estimativas diferem.

Atividade Prática:
Story Points e Planning Poker

Em grupos. Instruções em:

https://denmartins.github.io/labs/xp-planning-poker

Organização Temporal de um Projeto XP

Como XP estrutura o tempo e o esforço de desenvolvimento em níveis aninhados.

MACRO

Release

Conjunto de várias iterações: Reúne os ciclos menores de desenvolvimento para uma entrega funcional.

Horizonte de alguns meses: Define o ciclo de planejamento estratégico de médio prazo para o negócio.

MÉDIO

Iteração

Duração de semanas: Ciclos curtos e repetitivos, normalmente variando entre 1 a 3 semanas.

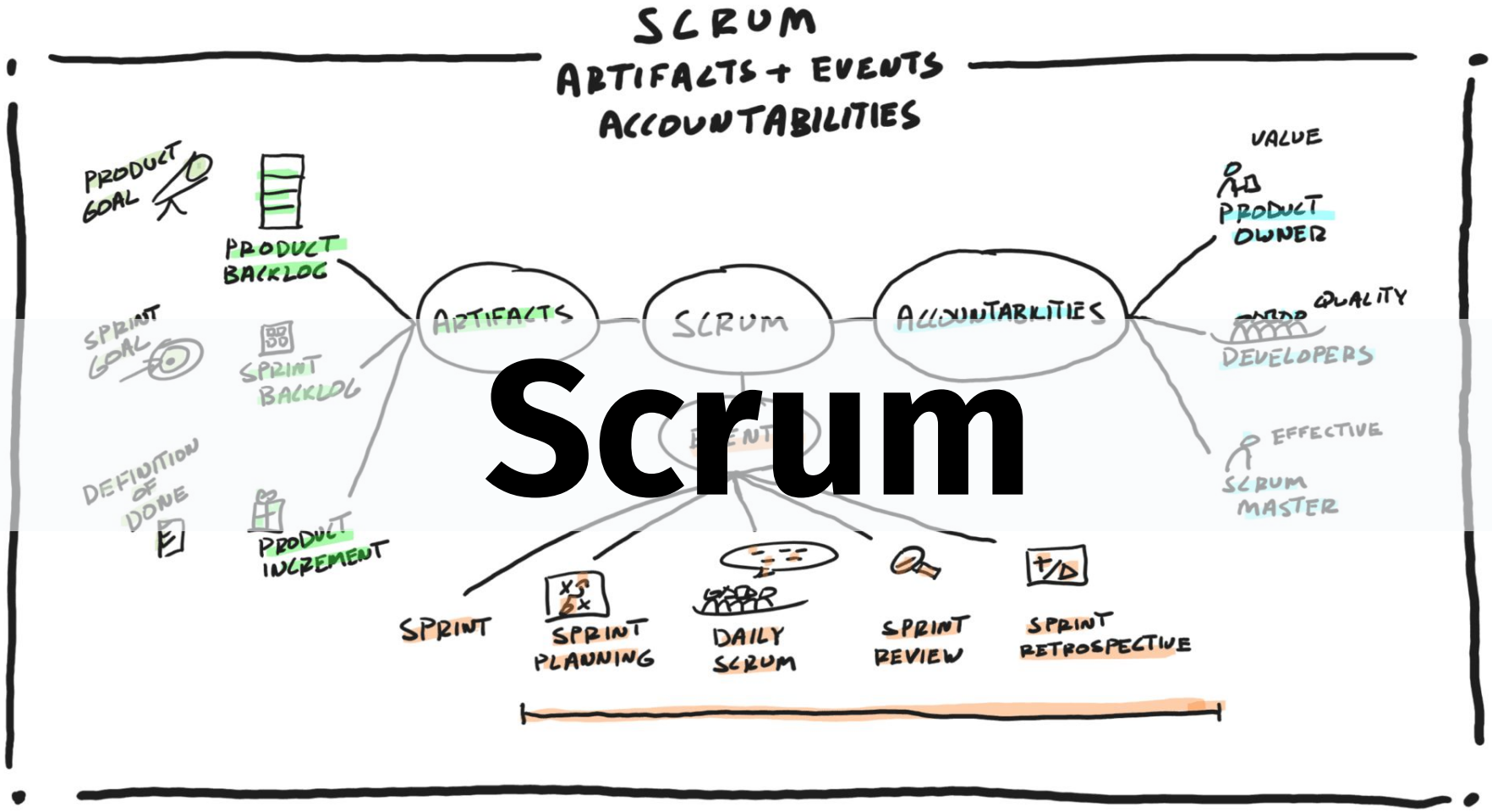
Conjunto de histórias: Implementa e entrega um bloco de funcionalidades de valor direto para o usuário.

MICRO

Tarefas

Atividades técnicas: Divisão prática das histórias de usuário em ações menores de engenharia.

Poucos dias: Devem ser iniciadas e concluídas de forma rápida, otimizando o fluxo diário.


SCRUM
ARTIFACTS + EVENTS
ACCOUNTABILITIES
PRODUCT GOAL
PRODUCT BACKLOG
SPRINT GOAL
SPRINT BACKLOG
DEFINITION OF DONE
PRODUCT INCREMENT
ARTIFACTS
SCRUM
ACCOUNTABILITIES
VALUE
PRODUCT OWNER
QUALITY
DEVELOPERS
EFFECTIVE
SCRUM MASTER
SPRINT
SPRINT PLANNING
DAILY SCRUM
SPRINT REVIEW
SPRINT RETROSPECTIVE

Scrum

Scrum: 3 pilares fundamentais

Scrum baseia-se no **empirismo**: decisões orientadas pela observação e vivência prática da realidade.

1. Transparência

Visibilidade e Compreensão

Informações relevantes do processo devem estar totalmente visíveis para aqueles que executam e recebem o trabalho.

Compartilhamento de:

- Progresso e resultados do time
- Dificuldades reais encontradas
- Riscos e impedimentos mapeados

Problemas nunca devem ser omitidos. A transparência estabelece a base de confiança para tomadas de decisão.

2. Inspeção

Avaliação Regular

Os artefatos e o progresso em direção aos objetivos devem ser inspecionados com frequência planejada.

Focos de Inspeção:

- O produto e os incrementos da Sprint
- Processos e práticas de engenharia
- Colaboração e dinâmica da equipe

O propósito é detectar variações indesejadas e validar hipóteses diretamente com o feedback dos stakeholders.

3. Adaptação

Ajustes e Melhoria

Se um processo desviar dos limites ou se o produto requerer ajustes, as mudanças devem ocorrer imediatamente.

Elementos Adaptáveis:

- Prioridades do Product Backlog
- Forma de trabalho e práticas diárias
- Decisões de rumo do produto

Minimiza desvios adicionais com foco absoluto na melhoria contínua e na entrega acelerada de valor real.

Scrum: Visão Geral

Fluxo básico

1. Stakeholders fornecem necessidades e feedback ao **Product Owner**

2. O **Product Owner** ordena o **Product Backlog**

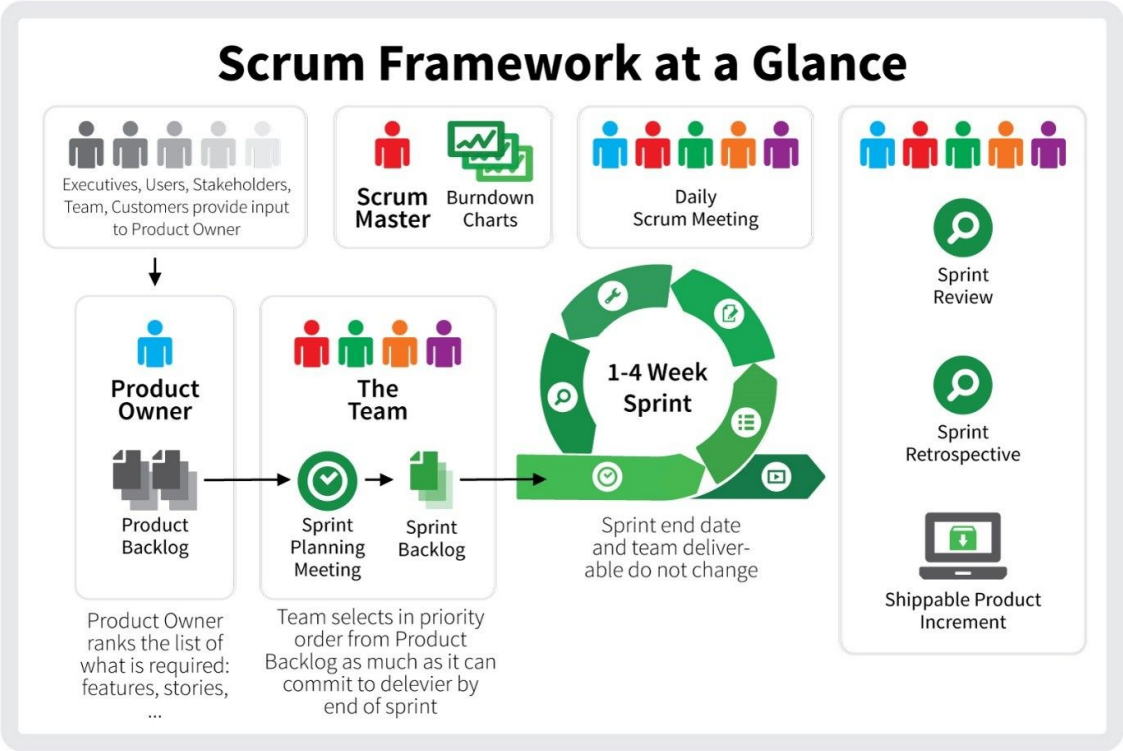
3. Na **Sprint Planning**, a equipe seleciona itens para o **Sprint Backlog**

4. Durante a **Sprint**, a equipe trabalha coletivamente e realiza o **Daily Scrum**

5. Ao final, entrega um **Incremento utilizável**

6. Na **Sprint Review**, stakeholders avaliam o resultado do trabalho

7. Na **Retrospective**, a equipe melhora continuamente seu processo



Fonte da Imagem: https://projectresources.cdt.ca.gov/agile/agile-tools-and-techniques/

Scrum: Lidando com Complexidade

Um framework ágil e leve projetado para gerar valor por meio de soluções adaptativas para problemas complexos.

FUNDAMENTOS

01 **Entregas incrementais:** Divisão do trabalho em pequenos ciclos focados em gerar valor utilizável.

02 **Inspeção frequente:** Avaliação regular do progresso em direção às metas do projeto.

03 **Adaptação contínua:** Ajuste rápido de processos e planos ao menor sinal de desvio.

04 **Transparência do trabalho:** Visibilidade completa dos processos e do estado real do projeto.

05 **Colaboração entre pessoas:** Sinergia constante entre o time e os stakeholders.

Scrum: Responsabilidades e Papéis

Como os três papéis principais colaboram para garantir o sucesso e a entrega de valor na Sprint.

1. Desenvolvedores

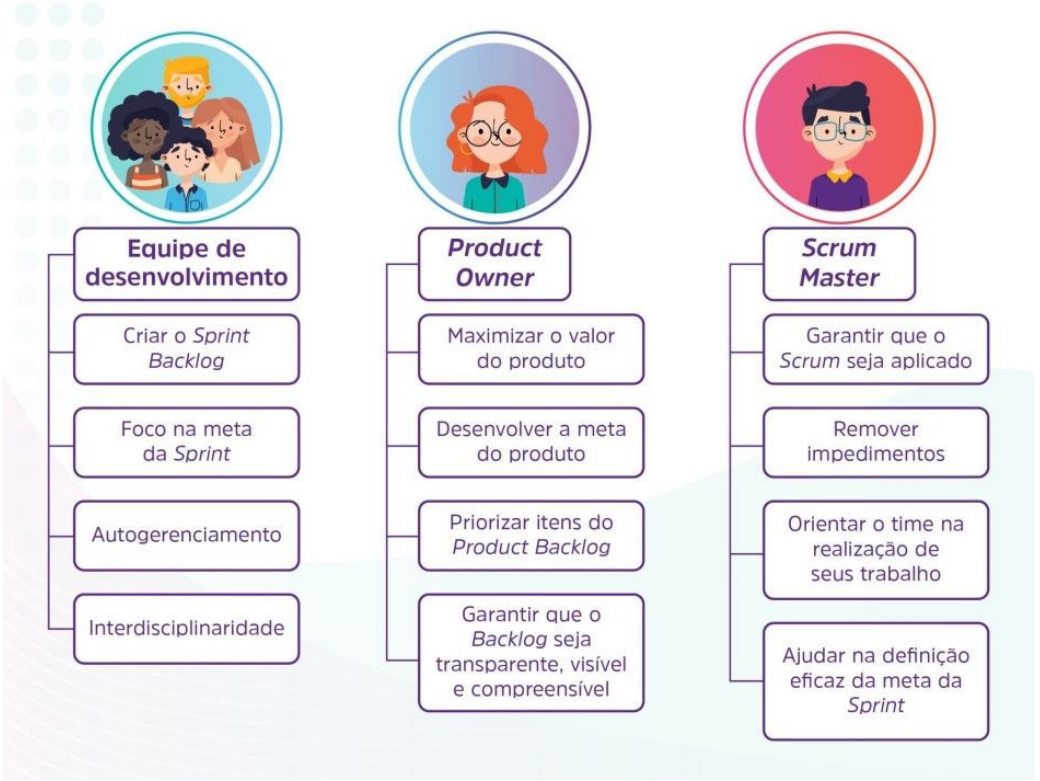
Planejam e produzem o trabalho técnico necessário para atingir a meta da Sprint.

2. *Product Owner*

Gerencia de forma eficaz o backlog e define as prioridades das entregas. Comunica de forma clara os objetivos e os itens de valor para o negócio.

3. *Scrum Master*

Apoia o uso do Scrum ajudando todos a entenderem a teoria e a prática. Ajuda ativamente a remover impedimentos que barram o progresso do time.



Fonte da Imagem: https://br.linkedin.com/in/francisco-nogueira-amorim

Scrum: Artefatos e Compromissos

Product Backlog

O que é necessário para desenvolver o produto.

COMPROMISSO

Product Goal

Indica a meta de longo prazo e onde o produto quer chegar.

Sprint Backlog

Elementos selecionados para a Sprint e o plano para realizá-los.

COMPROMISSO

Sprint Goal

Define o objetivo de curto prazo a ser alcançado na Sprint.

Incremento

Resultado utilizável produzido durante a Sprint.

COMPROMISSO

Definition of Done

Estabelece o critério de qualidade de quando o trabalho está concluído.

Por que os compromissos são importantes?

Enquanto os **Artefatos** tornam o trabalho e o progresso visíveis, os **Compromissos** fornecem foco e critérios objetivos para avaliar o progresso do time.

Atividade Prática:
Scrum Sprint

Em grupos. Instruções em:

https://denmartins.github.io/labs/scrum-first-sprint

Conclusão

Fundamentos Ágeis

Planejamento Contínuo: Agilidade não é ausência de plano; ele é contínuo e adaptado a novas informações.

Ciclos Curtos: O produto evolui de forma iterativa e incremental com entregas frequentes.

Feedback Ativo: Usuários e stakeholders orientam prioridades e mudanças.

Extreme Programming

Técnicas práticas para a gestão ágil:

Programação em Pares;

TDD;

Refatoração;

Integração Contínua;

Design simples.

Scrum

Organiza em Backlog, Sprint, Incremento, Review e Retrospective.

Pilares: Baseia-se em transparência, inspeção e adaptação.

> Equipes ágeis conseguem responder a mudanças com sucesso porque combinam de forma sinérgica: **feedback frequente** + **priorização contínua** + **disciplina técnica** + **melhoria contínua**.

Referências

VALENTE, Marco Tulio. Engenharia de Software Moderna. Capítulo 2.

PRESSMAN, R. S.; MAXIM, B. R. Engenharia de Software: Uma Abordagem Profissional. 9. ed. Capítulos 2 e 3.

SOMMERVILLE, Ian. Engenharia de Software. 10. ed. Capítulo 2.

Manifesto for Agile Software Development: https://agilemanifesto.org/

Dúvidas e Discussão

Veja também:

Agile Evolution & the Future of SW Engineering • Martin Fowler & Kent Beck (YouTube)

What happened to the agile movement? Uncle Bob (YouTube)