



Metodologia de Desenvolvimento

▼ 🏗️ Metodologias de Desenvolvimento Clássicas

▼ 📖 Definição

As metodologias de desenvolvimento de software, também chamadas de tradicionais, são abordagens estruturadas e sequenciais. Elas se caracterizam por:

- ✅ Processo linear
- ✅ Fases bem definidas, onde cada etapa depende da anterior
- ✅ Forte planejamento e documentação
- ✅ Execução controlada e previsível

🎯 O principal objetivo é reduzir riscos e garantir que o produto final atenda aos requisitos estabelecidos.

▼ ? Por que utilizar uma metodologia de desenvolvimento?

Adotar uma metodologia clara traz organização e eficiência para os projetos. Entre os benefícios, destacam-se:

- 🔄 Gerenciamento de mudanças
- 📈 Aumento da produtividade com etapas bem definidas
- 🕒 Cumprimento de prazos
- 💬 Comunicação clara entre os membros da equipe
- 🎯 Alinhamento com as expectativas dos clientes e do negócio
- 🧹 Redução de desperdícios e otimização dos recursos
- 🔧 Melhor resposta a obstáculos e garantia de qualidade

▼ 📌 Exemplos de Metodologias Clássicas

▼ 💧 Modelo Cascata (Waterfall)

A metodologia Waterfall, ou cascata, é uma metodologia clássica de desenvolvimento de software que ajuda as equipes a seguir os processos em um fluxo linear. É uma das mais conhecidas, onde cada fase é concluída antes da próxima, sem retorno às etapas anteriores.

✅ Vantagens:

- ✓ Clareza no progresso
- ✓ Ideal para requisitos estáveis
- ✓ Gerenciamento facilitado em grandes equipes

❌ Desvantagens:

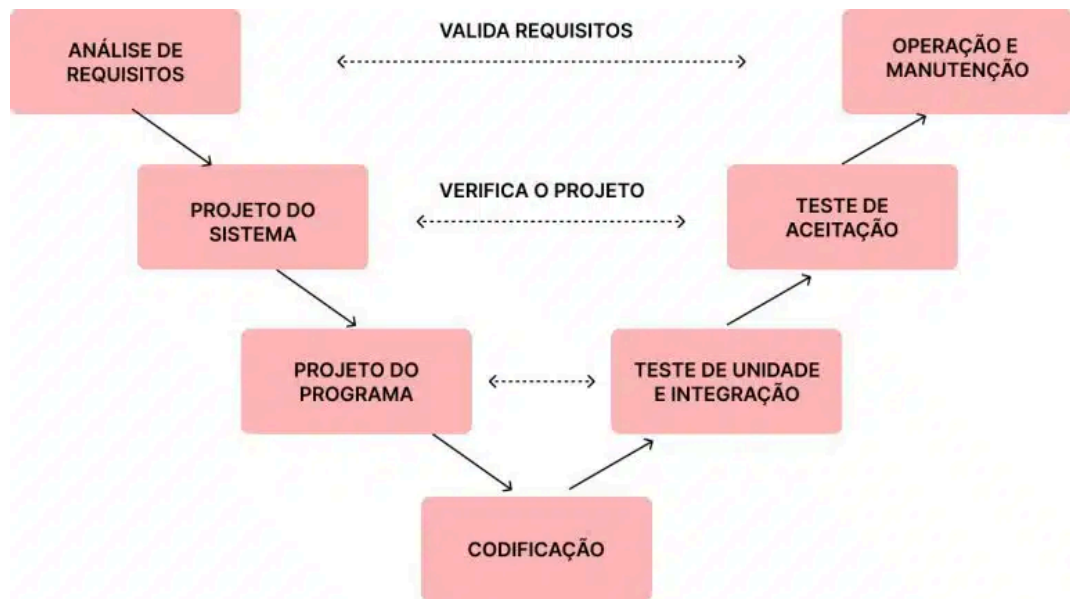
- ⚠️ Pouca flexibilidade
- ⚠️ Risco de desalinhar com as necessidades reais
- ⚠️ Testes tardios

▼ ✅ Modelo em V

Variação do modelo cascata com ênfase em testes e validação. Cada fase de desenvolvimento tem um teste correspondente, formando um "V".

▾ **Na descida do V**, são realizadas as atividades de **especificação, análise e design**, que definem o funcionamento esperado do sistema.

▴ **Na subida do V**, ocorrem os **testes correspondentes** a cada fase anterior, com o objetivo de **validar se o sistema foi implementado corretamente e atende aos requisitos definidos**.



✅ Vantagens:

🛡️ Alta qualidade com testes integrados

📅 Planejamento claro

🔍 Rastreabilidade entre fases

❌ Desvantagens:

🚫 Pouca flexibilidade

💰 Custo inicial elevado

🐌 Testes executados tardiamente

🧩 Pouco eficaz em ambientes dinâmicos

▼ 🕒 Aplicabilidade

As metodologias clássicas são mais adequadas para:

🧱 Projetos com requisitos estáveis

👥 Grandes equipes ou times distribuídos

📐 Escopo fixo, com pouco espaço para alterações

⚠️ Menos eficaz em ambientes ágeis, que exigem mudanças frequentes, feedback rápido ou entregas iterativas.

▼ 🧩 Fases de Desenvolvimento Clássico

📌 Fases típicas no modelo cascata:

1. 🔍 **Levantamento de Requisitos** – entender as necessidades do cliente

2. 🧠 **Análise de Requisitos** – formalização e validação das demandas
3. 🏗️ **Design do Sistema** – definição da arquitetura e componentes
4. 💻 **Implementação** – codificação com base no design
5. 🧪 **Testes** – verificação e validação do sistema
6. 🚀 **Implantação** – entrega para uso em produção
7. 🛠️ **Manutenção** – correções e melhorias pós-entrega

🔄 Em variações como o **modelo incremental** ou **modelo em V**, essas fases podem se sobrepor ou acontecer em ciclos, mas mantêm a essência estruturada.

▼ ⚡ **Metodologias de Desenvolvimento Ágeis**

▼ 📄 **O que são metodologias ágeis?**

As metodologias ágeis são abordagens modernas de gestão de projetos que visam maior **rapidez**, **flexibilidade** e **colaboração** durante o ciclo de vida de um projeto — da concepção à entrega.

🔄 Em vez de seguir um modelo linear e inflexível, as metodologias ágeis adotam um fluxo iterativo e incremental, permitindo adaptações constantes conforme o projeto evolui.

▼ ⭐ **Benefícios das metodologias ágeis**

- ✅ Otimizam os fluxos de trabalho
- ✅ Aumentam a produtividade das equipes
- ✅ Elevam as chances de sucesso do projeto
- ✅ Promovem maior satisfação do cliente

🌐 O objetivo é tornar as empresas mais eficientes e competitivas frente ao mercado em constante mudança.

✨ Complemento: o uso do **Design Thinking** pode potencializar o processo criativo ao lado das metodologias ágeis.

▼ 📖 **O Manifesto Ágil**

🕒 Em 2001, 17 desenvolvedores se reuniram em Utah (EUA) para criar um conjunto de princípios mais leves e flexíveis para o desenvolvimento

de software.

Dessa iniciativa surgiu o **Manifesto Ágil**, que estabeleceu:

4 valores principais:

- Indivíduos e interações mais que processos e ferramentas
- Software funcionando mais que documentação abrangente
- Colaboração com o cliente mais que negociação de contratos
- Responder às mudanças mais que seguir um plano

 12 princípios que guiam a prática ágil




▼ Principais Metodologias Ágeis

▼ Kanban

Origem: Criado pela Toyota nos anos 40 para gerenciar a produção. Adaptado para software por David J. Anderson.

O que é? Um sistema visual baseado em cartões e colunas que sinalizam o status das tarefas.

 17 Estrutura básica:

-  A Fazer
-  Em Progresso
-  Concluído

Cada cartão representa uma tarefa, que se move pelas colunas conforme avança.



Cada cartão no quadro representa uma tarefa. À medida que a equipe trabalha, os cartões são movidos de uma coluna para outra.



Exemplo de tarefas:

- Criar layout do site
- Desenvolver login e senha
- Implementar banco de dados



Flexível: colunas adicionais como Teste ou Revisão podem ser incluídas conforme a necessidade.

▼ 🏀 Scrum

Origem: Criado por Jeff Sutherland e Ken Schwaber na década de 1990. Inspirado no trabalho em equipe do rugby ("Time Scrum").

O que é? Um framework (estrutura) ágil que organiza o trabalho em ciclos curtos, chamados Sprints, com duração de 1 a 4 semanas.



▼ 🏆 Pilares do Scrum:

- ✨ Transparência
- 🌐 Inspeção
- ⚖️ Adaptação



▼ 📖 Papéis no Scrum:

Product Owner (Dono do Produto): Representa os interesses dos envolvidos e é responsável por maximizar o valor do produto. Define a visão do projeto, prioriza o Product Backlog (lista de requisitos ou tarefas) e garante que o time entenda o que precisa ser entregue.

Toma decisões sobre o escopo e aceita ou rejeita o trabalho concluído em cada Sprint. Não interfere na execução técnica, mas foca nas necessidades do projeto.

Scrum Master: Atua como facilitador e protetor do time, garantindo que o processo Scrum seja seguido. Remove impedimentos, promove a auto-organização da equipe e ajuda a melhorar continuamente os processos.

Deve ter conhecimento do processo, ter excelentes habilidades de comunicação e ser capaz de treinar e motivar o time.

Time de Desenvolvimento: Composto por profissionais que executam o trabalho técnico como desenvolvedores, designers, testadores, etc., é responsável por entregar incrementos funcionais do produto ao final de cada Sprint. O time é auto-organizado e multifuncional, ou seja, decide como o trabalho será feito e possui todas as habilidades necessárias para completá-lo. Não há hierarquia interna, o foco é na colaboração e na entrega coletiva.

▼ **Artefatos do Scrum:**

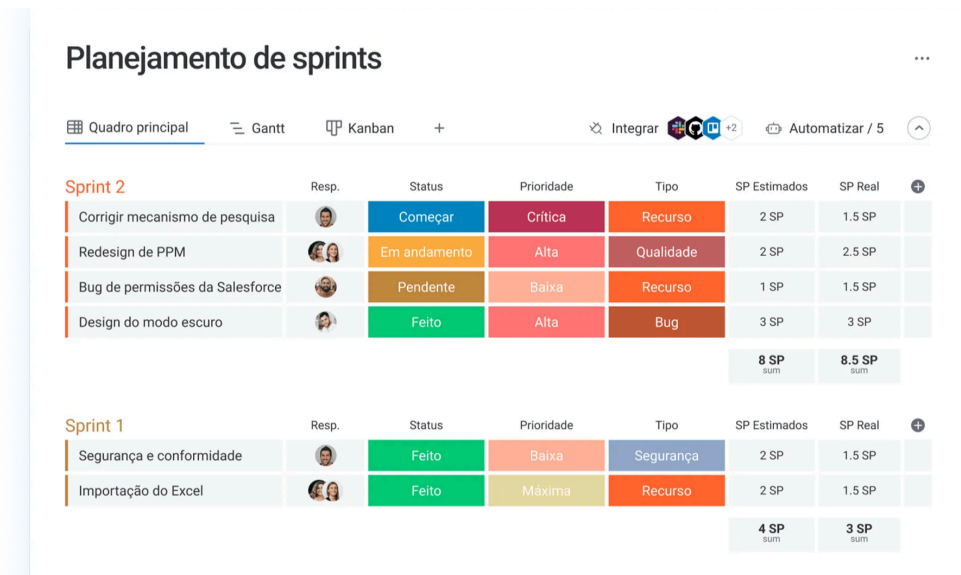
Os artefatos são o que fazem esta metodologia singular, recursos aplicados no dia a dia de implementação e uso do método:

- **Product Backlog:** uma lista de todas as tarefas e características que constituem um projeto. É constantemente atualizada com base no feedback do usuário e na avaliação da equipe sobre seu progresso.
- **Sprint Backlog:** uma lista de tarefas e recursos a serem completados dentro de um sprint em particular.
- **Incremento:** o incremento é uma forma de gerar valor ao produto e eles são adicionados à medida que os itens do backlog são concluídos. Trata-se de itens, recursos e funcionalidades “a mais”, que podem ser integradas ao produto e utilizadas pelos usuários finais.
- **Burndown Chart:** um gráfico que mostra quanto trabalho foi completado em cada sprint e quanto ainda há para ser feito. Ele é usado para rastrear progresso e garantir que os prazos sejam cumpridos.



▼ **Eventos do Scrum:**

- **Sprint:** Ciclo de trabalho para entregar um incremento do produto.
- **Sprint Planning:** Planejamento do que será feito na Sprint baseado no backlog do produto.
- **Daily Scrum:** Reunião diária para alinhamento, onde o time sincroniza as atividades e planeja o dia.

- **Sprint Review:** Ao final do Sprint, o time apresenta o incremento ao Product Owner e demais envolvidos.
- **Sprint Retrospective:** Reflexão sobre o que funcionou e não funcionou como pode ser melhorado.



▼ ? Diferença entre Scrum e Kanban

Critério	 Scrum	 Kanban
Estrutura	Estruturado em ciclos (Sprints)	Fluxo contínuo
Papéis definidos	Sim (PO, Scrum Master, Time)	Não há papéis formais
Regras	Regras bem definidas	Regras flexíveis
Mudanças	Não são permitidas durante a Sprint	Permitidas a qualquer momento
Visualização do trabalho	Acompanhado por Sprints e Burndown	Quadro visual com cartões