Marisangila Alves, MSc
marisangila.alves@edu.sc.senai.br
marisangila.com.br

2026/1

Fundamentos de Desenvolvimento de Sistemas

Metodologias de Desenvolvimento de Software

Sumário

Metodologias de Desenvolvimento

O que são metodologias?

Metodologias são **conjuntos organizados de práticas, técnicas e processos** utilizados para planejar, desenvolver e manter sistemas de software.

Elas definem:

- Como o software será desenvolvido;
- Quais etapas devem ser seguidas;
- Como a equipe deve trabalhar;
- Como garantir qualidade no produto final.

Metodologia Classica

> *O modelo cascata, algumas vezes chamado de modelo sequencial linear, sugere uma abordagem sequencial e sistemática 3 para o desenvolvimento de software, começando com a especicação dos requisitos do cliente, avançando pelas fases de planejamento, modelagem, construção e entrega, e culminando no suporte contínuo do software concluído.*
>
> (Pressman; Maxim, 2014)

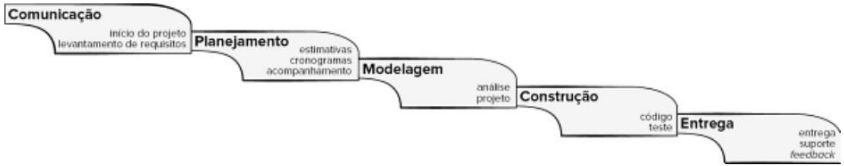


Figura 1: O modelo cascata (Pressman; Maxim, 2014).

- Processo estruturado;
- Forte documentação;
- Planejamento detalhado;
- Mudanças são difíceis após início do projeto.

Metodologia Ágil

Conceito

Metodologias ágeis priorizam **entregas rápidas, adaptação a mudanças e colaboração da equipe**.

12 Princípios do Manifesto Ágil:

1. o cliente por meio da entrega contínua e antecipada de software com valor.
2. Aceitar mudanças de requisitos mesmo em fases avançadas do desenvolvimento.
3. Entregar software funcionando com frequência, em ciclos curtos.
4. Promover colaboração constante entre desenvolvedores e pessoas de negócio.
5. Construir projetos em torno de equipes motivadas e apoiadas.
6. Priorizar comunicação direta e face a face entre os membros da equipe.

7. Utilizar software funcionando como principal indicador de progresso.
8. Manter um ritmo de desenvolvimento sustentável ao longo do projeto.
9. Buscar continuamente excelência técnica e bom design.
10. Valorizar a simplicidade e evitar trabalho desnecessário.
11. Permitir que arquiteturas e soluções surjam de equipes auto-organizáveis.
12. Refletir regularmente sobre o processo e buscar melhoria contínua.

Scrum

> *Scrum é um framework ágil para gestão e planejamento de projetos que permite às equipes entregar produtos de alta qualidade de forma rápida e eficiente através de ciclos iterativos e incrementais.*
>
> (Portal Scrum, 2025)

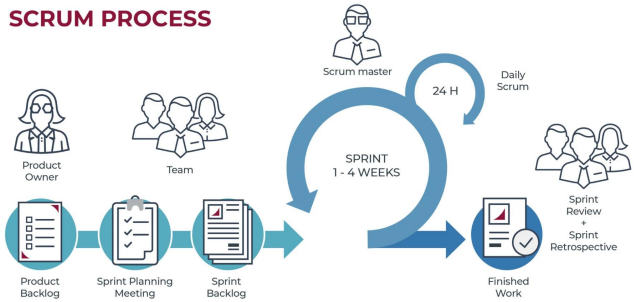


Figura 2: Processo do Scrum.

Pilares:

- **Transparência**: Aspectos significativos do processo devem ser visíveis para todos os responsáveis pelo resultado. Transparência requer que esses aspectos sejam definidos por um padrão comum.;
- **Scrum Inspeção**: Os artefatos e progresso devem ser inspecionados frequentemente para detectar variações indesejáveis. A inspeção é mais benéfica quando realizada de forma diligente por inspetores qualificados.;
- **Adaptação**: Se algum aspecto do processo estiver fora dos limites aceitáveis, o processo ou material deve ser ajustado. O ajuste deve ser feito o mais rápido possível..

Papéis:

- **Product Owner**: Define o que será construído e prioriza as funcionalidades com base no valor de negócio;
- **Scrum Master**: Garante que o framework Scrum seja seguido e remove impedimentos da equipe;
- **Time de Desenvolvimento**: Profissionais que fazem o trabalho de entregar um incremento potencialmente funcional do produto.

Cerimônias:

- **Sprint Planning**: Cerimônia que inicia o Sprint. A equipe planeja o trabalho a ser realizado durante o Sprint;
- **Daily Scrum**: Reunião diária de 15 minutos para sincronização da equipe e planejamento das próximas 24 horas;
- **Sprint Review**: Demonstração do trabalho concluído durante o Sprint para stakeholders e coleta de feedback;
- **Sprint Retrospective**: Reflexão sobre o processo e identificação de melhorias para o próximo Sprint.

Artefatos:

- **Product Backlog**: Lista ordenada e priorizada de funcionalidades, requisitos, melhorias e correções que constituem as mudanças a serem feitas no produto.
- **Sprint Backlog**: Conjunto de itens do Product Backlog selecionados para o Sprint, mais um plano para entregar o incremento do produto.

Extreme Programming (XP)

> *A Extreme Programming (Programação Extrema) envolve um conjunto de regras e práticas constantes no contexto de quatro atividades metodológicas: planejamento, projeto, codicação e testes.*
>
> (Pressman; Maxim, 2014)

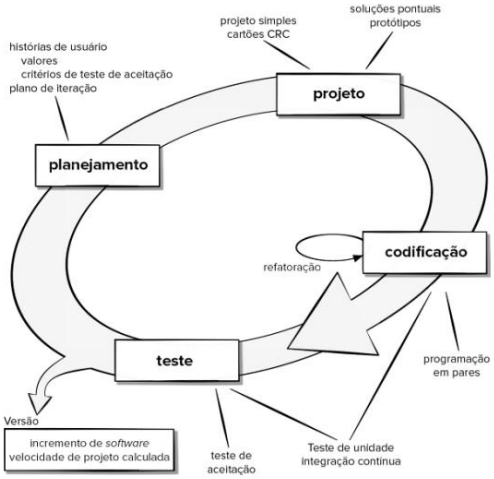


Figura 3: Extreme Programming (Pressman; Maxim, 2014).

- **Planejamento**: clientes e desenvolvedores definem e priorizam histórias de usuário para organizar as entregas incrementais do software.
- **Projeto**: o desenvolvimento segue o princípio da simplicidade (KISS), produzindo apenas o projeto essencial para implementar as funcionalidades necessárias.
- **Codificação**: o código é desenvolvido a partir de testes de unidade e frequentemente utilizando programação em pares e integração contínua.
- **Testes**: testes automatizados e de aceitação garantem que cada funcionalidade implementada atenda aos requisitos definidos pelo cliente.

Práticas importantes:

- Programação em par;
- Integração contínua;
- Testes automatizados;

Kanban

O Kanban nasceu na Toyota, onde era uma série de práticas de engenharia industrial.

> *O método Kanban [And16] é uma metodologia enxuta que descreve métodos para melhorar qualquer processo ou fluxo de trabalho. O Kanban enfoca a gestão de alterações e a entrega de serviços. A Os membros de equipe gerenciam o trabalho e recebem liberdade para se organizarem de modo a completá-lo.*
>
> (Pressman; Maxim, 2014)

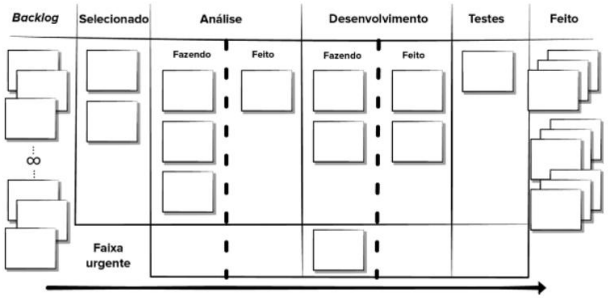


Figura 4: Kanban (Pressman; Maxim, 2014).

Figura 5: Kanban (Pressman; Maxim, 2014).

1. **Visualizar o fluxo de trabalho**: utilizar um quadro Kanban para acompanhar o progresso das tarefas desde "por fazer" até "feito".
2. **Limitar o trabalho em andamento (WIP)**: restringir a quantidade de tarefas simultâneas para melhorar o foco e a qualidade.
3. **Gerenciar o fluxo**: analisar o processo para identificar gargalos e reduzir desperdícios.

4. **Tornar as políticas explícitas**: definir claramente critérios e regras do processo de desenvolvimento.
5. **Melhoria contínua**: utilizar ciclos de feedback para avaliar e aprimorar o processo.
6. **Colaboração no processo**: envolver toda a equipe na adaptação e evolução do método de trabalho.

DevOps

> *O DevOps tenta aplicar os princípios do desenvolvimento ágil e do enxuto a toda a cadeia logística de software.*
>
> (Pressman; Maxim, 2014)

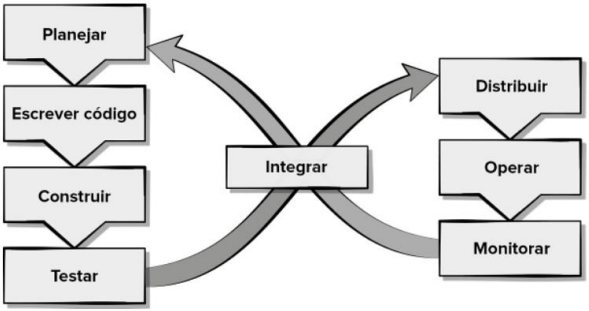


Figura 6: DevOps (Pressman; Maxim, 2014).

1. **Desenvolvimento contínuo**: desenvolvimento incremental e frequente, permitindo entregas frequentes para testes de qualidade.
2. **Teste contínuo**: testes automatizados verificam constantemente o código para detectar defeitos antes da integração.

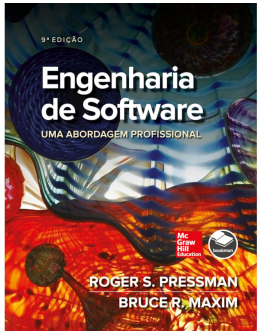
3. **Integração contínua**: alterações no código são integradas frequentemente ao repositório principal e verificadas automaticamente.
4. **Entrega contínua**: o software integrado é disponibilizado rapidamente no ambiente de produção.
5. **Monitoramento contínuo**: o desempenho do software é acompanhado em produção para identificar e corrigir problemas rapidamente.

Glossário

Termo	Descrição
CI/CD	Integração e entrega contínua de software por meio de automação.
Test Driven Development (TDD)	Desenvolvimento baseado na criação de testes antes do código.
Code Review	Revisão de código para garantir qualidade e boas práticas.
Refatoração	Melhoria da estrutura do código sem alterar seu comportamento.
User Story	Descrição simples de uma funcionalidade sob a perspectiva do usuário.
Sprint	Ciclo curto de desenvolvimento usado em processos ágeis.
Refinamento	Atividade de detalhar e esclarecer requisitos antes da implementação.
Stakeholder	Pessoa ou grupo interessado ou impactado pelo sistema desenvolvido.
Backlog	Lista priorizada de funcionalidades, melhorias e correções do projeto.

Tabela 1: Termos comuns no desenvolvimento de software e no mercado de trabalho.

(Stallings, 2017)



BECK, KENT ET AL. **Manifesto para o Desenvolvimento Ágil de Software**. [*S. l.: s. n.*], 2001. https://agilemanifesto.org/iso/ptbr/manifesto.html. Acesso em: 16 mar. 2026.

KANBAN UNIVERSITY. **Kanban University**. [*S. l.: s. n.*], 2021. https://kanban.university/. Acesso em: 16 mar. 2026.

PORTAL SCRUM. **O que é Scrum?** [*S. l.: s. n.*], 2025. https://www.scrum.com.br/blog/o-que-e-scrum.html. Acesso em: 16 mar. 2026.

PRESSMAN, Roger S.; MAXIM, Bruce R. **Software Engineering: A Practitioner's Approach**. 8. ed. New York: McGraw-Hill Education, 2014.

SOMMERVILLE, Ian. **Engenharia de Software**. 10. ed. São Paulo: Pearson, 2019.

STALLINGS, William. **Arquitetura e Organização de Computadores**. Última edição. [*S. l.*]: Pearson, 2017.

Estes slides estão protegidos por uma licença Creative Commons

Este modelo foi adaptado de Maxime Chupin.