**Metodologia XP (Extreme Programming)**

**O que é?**

* Uma metodologia ágil de desenvolvimento de software focada em **qualidade do código**, **feedback rápido** e **resposta a mudanças**.

**Origens**

* Criada por **Kent Beck** em meados dos anos 1990, no projeto **C3 (Chrysler Comprehensive Compensation System)**.
* Popularizada com o livro *Extreme Programming Explained* (1999).

**Principais características**

* Iterações curtas.
* Entregas frequentes.
* Práticas de engenharia de software como **Test-Driven Development (TDD)** e **pair programming**.

**Valores**

1. **Comunicação**
2. **Simplicidade**
3. **Feedback**
4. **Coragem**
5. **Respeito**

**Princípios**

* Desenvolvimento incremental.
* Melhoria contínua.
* Entregar o software mais simples que funcione.

**Propósito**

* Reduzir riscos técnicos.
* Garantir **alta qualidade de código**.
* Atender a mudanças frequentes do cliente.

**Práticas**

* **Pair Programming**
* **Refatoração contínua**
* **TDD** (Testes antes do código)
* **Integração Contínua**
* **Pequenas releases**
* **Co-locação da equipe**
* **Propriedade coletiva do código**
* **Cliente sempre disponível**

**Fases ou ciclo de vida**

* **Exploração** (identificação das histórias)
* **Planejamento** (planning game)
* **Iterações curtas** (1 a 2 semanas)
* **Produção** (entrega)
* **Manutenção** (ajustes pós-entrega)

**Papéis**

* **Programadores**
* **Cliente in loco**
* **Coach (facilitador)**
* **Tracker (mediador de métricas)**
* **Testador**

**Artefatos**

* **Histórias de usuário**
* **Cartões de tarefas**
* **Testes automatizados**

**Cerimônias**

* **Planning Game**
* **Stand-ups** (diários, opcionais)
* **Retrospectivas**

**Vantagens**

* Alta qualidade de software.
* Feedback constante.
* Reduz falhas de comunicação.

**Desvantagens**

* Pode ser **difícil em equipes grandes**.
* Exige **muita disciplina técnica**.
* Alto custo se não houver cliente disponível.

**Terminologias importantes**

* **Spike** → Experimento rápido para explorar soluções técnicas.
* **Planning Game** → Planejamento das próximas iterações.

**Métricas**

* **Velocidade da equipe**
* **Cobertura de testes**
* **Defeitos encontrados**

**✅ Método Ágil Kanban**

**O que é?**

* Um **sistema visual** para gerenciar o fluxo de trabalho, baseado em **quadros e cartões**.
* Focado em **melhoria contínua** e **eliminação de gargalos**.

**Origens**

* Criado na **Toyota** nos anos 1940 (Taiichi Ohno), no sistema de produção Lean.
* Adaptado para o desenvolvimento de software por **David Anderson** nos anos 2000.

**Principais características**

* Visualização do fluxo de trabalho.
* Limite de trabalho em progresso (WIP).
* Entrega contínua.

**Valores e princípios**

* Transparência.
* Foco no fluxo.
* Evolução incremental.
* Colaboração.

**Propósito**

* Melhorar a eficiência do processo.
* Identificar gargalos rapidamente.
* Evitar sobrecarga da equipe.

**Práticas**

* **Quadro Kanban** (To Do / Doing / Done ou personalizado).
* **Limite de WIP**.
* **Lead Time** e **Cycle Time** tracking.
* **Feedback loops**.

**Fases ou ciclo de vida**

* Não há fases fixas → é **contínuo**.
* O ciclo é guiado pelo fluxo: **Ideia → Desenvolvimento → Conclusão**.

**Papéis**

* Não define papéis específicos.
* Geralmente mantêm papéis existentes da organização.

**Artefatos**

* **Quadro Kanban**
* **Cartões de trabalho**

**Cerimônias**

* Não são obrigatórias.
* Muitas equipes usam: **reuniões de revisão de fluxo** ou **retrospectivas**.

**Vantagens**

* Simples de implementar.
* Adaptável a qualquer contexto.
* Foco em eficiência e fluxo.

**Desvantagens**

* Falta de diretrizes claras pode causar caos se mal implementado.
* Pouca ênfase em engenharia (comparado ao XP).

**Terminologias importantes**

* **Lead Time** → tempo total do pedido até a entrega.
* **Cycle Time** → tempo de execução efetiva da tarefa.
* **WIP (Work In Progress)** → limite de tarefas em andamento.

**Métricas**

* **Throughput** (tarefas concluídas).
* **Lead Time / Cycle Time**.
* **Eficiência do fluxo**.

**🔄 Comparativo XP x Kanban x Scrum**

| **Aspecto** | **XP** | **Kanban** | **Scrum** |
| --- | --- | --- | --- |
| **Origem** | Kent Beck, anos 90 | Toyota (Lean), adaptado p/ software 2000 | Jeff Sutherland e Ken Schwaber, anos 90 |
| **Foco** | Qualidade do código | Fluxo contínuo | Gestão de entregas em sprints |
| **Iterações** | Curtas (1–2 semanas) | Contínuo | Sprints fixos (2–4 semanas) |
| **Papéis** | Cliente, Devs, Coach, Tracker | Não define | PO, SM, Dev Team |
| **Artefatos** | Histórias, testes, cartões | Quadro Kanban, cartões | Product Backlog, Sprint Backlog, Incremento |
| **Cerimônias** | Planning Game, Retrospectivas | Reuniões de fluxo (opcionais) | Sprint Planning, Daily, Review, Retrospective |
| **Métricas** | Velocidade, qualidade | Lead/Cycle Time, WIP | Velocidade, burndown |
| **Vantagem** | Código de alta qualidade | Simplicidade e fluxo claro | Estrutura disciplinada e popular |
| **Desvantagem** | Exige muita disciplina técnica | Pode virar caos sem limites claros | Rigidez nas regras pode incomodar |

**🎯 Cenários de uso**

* **XP** → Projetos de software que exigem **alta qualidade técnica** e **mudanças constantes** (ex.: startups de tecnologia, sistemas críticos).
* **Kanban** → Equipes que precisam de **flexibilidade** e querem **melhorar processos gradualmente** (ex.: manutenção, suporte, operações).
* **Scrum** → Projetos que pedem **previsibilidade, ritmo e alinhamento forte entre papéis** (ex.: produtos digitais com roadmap claro).