**1. Método Kanban**

O Kanban é um método ágil precursor focado na **otimização de fluxos de trabalho** e na **melhoria contínua**, com a finalidade de apoiar times e organizações na escolha e adaptação de abordagens que alavanquem resultados.

• **Origem e Evolução**:

    ◦ Surgiu no final dos anos 1940 na indústria, como parte do **sistema Toyota de produção**.

    ◦ Criado por Taiichi Ohno em 1953 como um sistema de gestão visual com cartões para registrar atividades.

    ◦ Em 2010, David J. Anderson e Don Reinertsen o adaptaram para o desenvolvimento de software, combinando o **lean mindset** (crenças, suposições, atitudes e valores alinhados com o manifesto ágil) com a teoria das restrições.

    ◦ Devido à sua capacidade adaptativa, estendeu-se para além da indústria de software, sendo adotado em diversas áreas como RH e jurídica.

• **Conceitos Fundamentais**:

    ◦ **Kanban (Kan = Visual e Ban = Cartão)**: Um cartão que registra uma atividade.

    ◦ **Quadro Kanban**: Um quadro visual (físico ou virtual) organizado por colunas que representam o processo de trabalho, onde as atividades ficam explícitas para todos os membros do time, gerenciando o fluxo e priorizando o valor.

    ◦ **Sistema Puxado**: A demanda é gerada pelo cliente, que "puxa" a oferta, adaptando o ritmo produtivo e os estoques, em contraste com o sistema "empurrado" que gera estoque.

    ◦ **Método Kanban**: Mais do que o quadro, é composto por um **conjunto de princípios, práticas e cadências**.

• **Valores do Kanban**: Os valores são pilares para que as organizações sejam ágeis, adaptando as abordagens metodológicas. Incluem:

    ◦ **Transparência**: Troca aberta de informações e vocabulário claro.

    ◦ **Equilíbrio**: Eficiência através do balanceamento de exigências, visões e habilidades.

    ◦ **Colaboração**: Melhoria na forma como as pessoas trabalham juntas.

    ◦ **Foco no cliente**: O cliente e o valor percebido por ele são o centro de interesse.

    ◦ **Fluxo de trabalho**: O trabalho como um fluxo contínuo ou pontual de geração de valor, que deve ser mapeado e mantido visível.

    ◦ **Liderança**: Competência necessária em todos os níveis para gerar valor.

    ◦ **Compreensão**: Entender o trabalho e os processos antes de melhorá-los.

    ◦ **Acordo**: Políticas explícitas sobre como as decisões serão tomadas.

    ◦ **Respeito**: A base sobre a qual os outros valores se fundamentam, na forma de reconhecimento, compreensão e consideração.

• **Princípios do Kanban**:

    ◦ **Comece a terminar e pare de começar**: Priorizar o término de tarefas em andamento antes de iniciar novas, puxando cartões das etapas finais do fluxo (leitura da direita para a esquerda).

    ◦ **Faça o trabalho usando um quadro visual**: Gestão à vista para decisões rápidas, comunicação aprimorada e transparência.

    ◦ **Puxe apenas quando houver capacidade para fazer o trabalho**: Evitar gargalos e desperdícios.

    ◦ **Priorize valor**: Fazer primeiro o que entrega mais valor mais cedo.

    ◦ **Gerencie riscos**: Análise primordial para respostas rápidas a ocorrências.

    ◦ **Construa uma cultura de alta confiança**: Promover um ambiente psicologicamente seguro.

    ◦ **Responda rápida e graciosamente às mudanças**: Manter um fluxo otimizado, trabalhando dentro da capacidade para maior flexibilidade.

    ◦ **Desenvolva o processo que já está em vigor**: Mapear o fluxo atual, rodar por um tempo, coletar métricas e só então realizar melhorias nos gargalos.

• **Fluxo de Trabalho e Métricas de Eficiência**:

    ◦ **Customer Kanban (Fluxo end-to-end)**: O fluxo completo da entrega, desde a idealização até o momento em que a ideia gera valor para o cliente. É composto por:

        ▪ **Upstream Kanban**: Gestão do fluxo para avaliar, preparar e refinar ideias/opções antes do desenvolvimento, garantindo tarefas sem ruídos e na granularidade correta.

        ▪ **Downstream Kanban**: Gestão do fluxo das etapas relacionadas ao desenvolvimento e entrega da solução (desenvolvimento, teste, homologação, implantação).

    ◦ **Métricas Lean**: Essenciais para medir a eficiência:

        ▪ **Lead time médio**: Tempo entre a data de compromisso e a entrega ao cliente.

        ▪ **Cycle time médio**: Tempo entre o compromisso e o término do desenvolvimento do item de trabalho.

        ▪ **Throughput**: Quantidade de itens entregues em um período.

        ▪ **Work in Progress (WIP)**: Trabalho em progresso, evidenciando a capacidade do fluxo. **Limitar o WIP** garante maior foco, finaliza atividades mais rapidamente e identifica gargalos, aumentando a frequência de entrega de valor.

    ◦ **Cumulative Flow Diagram (CFD)**: Gráfico de acompanhamento que registra cumulativamente a quantidade de itens no fluxo, distinguindo as etapas por cores. É útil para identificar gargalos e oportunidades de melhoria.

• **Práticas e Cadências**:

    ◦ **Práticas**: Visualizar o fluxo, limitar o WIP, gerenciar o fluxo (foco na gestão do trabalho, não microgerenciamento de pessoas), tornar as políticas explícitas, implementar ciclos de feedback e melhorar colaborativamente.

    ◦ **Papéis Sugeridos**:

        ▪ **Service Request Manager (SRM)**: Entende as necessidades e expectativas dos clientes.

        ▪ **Service Delivery Manager (SDM)**: Gerencia o fluxo de trabalho e entrega os itens selecionados.

    ◦ **Cadências (Rituais)**: Sugestões para planejamento e sincronismo:

        ▪ *Foco na Entrega*: Delivery Planning (conforme necessidade), Replenishment (semanal), Kanban Meeting (diária).

        ▪ *Foco na Estratégia*: Strategy Review (trimestral), Operations Review (mensal), Delivery Review (bissemanal), Risk Review (mensal).

• **Classes de Serviços**: Reconhecem que itens de trabalho não têm o mesmo valor ou pressão de prazo. Facilitam a auto-organização da equipe e o atendimento às necessidades do negócio. Os tipos comuns são:

    ◦ **Expedite**: Alta urgência, alto impacto, necessitam de entrega imediata (ex: bug crítico em produção).

    ◦ **Fixed Date (Date-Driven)**: Demandas com prazo fixo, geralmente sazonais ou regulatórias (ex: hotsite de Natal, implementação regulatória).

    ◦ **Standard**: Itens padrões priorizados por critérios explícitos (urgência, valor) (ex: nova funcionalidade de e-commerce).

    ◦ **Intangibles**: Demandas com ganhos difíceis de mensurar no curto prazo, mas importantes a médio/longo prazo (ex: substituição de plataforma).

    ◦ **Aspectos a considerar**: Visualização, impacto no WIP, prioridade e fluxo de trabalho diferente.

--------------------------------------------------------------------------------

**2. eXtreme Programming (XP)**

XP é um **framework ágil de desenvolvimento de software** que surgiu no final dos anos 1990 para atender à necessidade de mudanças no ciclo de desenvolvimento. Seu propósito é produzir software de maior qualidade e, ao mesmo tempo, proporcionar **melhor qualidade de vida para a equipe**. É o mais específico dos frameworks ágeis em termos de práticas de engenharia de software.

• **Origem**: Criado por Kent Beck após aplicar técnicas revolucionárias de engenharia de software em um projeto crítico na Chrysler.

• **Contextos de Uso Adequados**: Indicado quando há:

    ◦ **Mudança dinâmica dos requisitos de software**.

    ◦ Riscos causados por projetos de tempo fixo que utilizam novas tecnologias.

    ◦ **Pequena equipe de desenvolvimento trabalhando no mesmo local**.

    ◦ Tecnologia que permite testes automatizados de unidade e funcionais.

• **Valores do XP**:

    ◦ **Comunicação**: O desenvolvimento é um trabalho de equipe que depende da transferência de conhecimento, destacando a comunicação cara a cara.

    ◦ **Simplicidade**: Fazer apenas o absolutamente necessário, evitando desperdícios e tratando apenas os requisitos conhecidos.

    ◦ **Feedback**: Times identificam pontos de melhoria e revisam práticas através de feedbacks constantes.

    ◦ **Coragem**: Ação eficaz perante o medo, levantando questões organizacionais, parando o que não funciona e aceitando feedbacks.

    ◦ **Respeito**: Essencial para comunicação fluida, feedbacks construtivos e trabalho colaborativo.

• **Princípios do XP**: Os valores e as práticas são conectados pelos princípios:

    ◦ **Humanidade**: Softwares são desenvolvidos por pessoas com sentimentos e necessidades.

    ◦ **Economia**: Tudo o que o time faz deve ter valor de negócio associado, com design incremental.

    ◦ **Benefícios Mútuos**: Todas as atividades devem trazer benefícios a todos, no presente e futuro.

    ◦ **Semelhanças**: Copiar soluções de problemas semelhantes pode ajudar, com atenção ao contexto.

    ◦ **Melhorias**: Ser pragmático, buscando o melhor hoje e estando preparado para o amanhã, sem buscar perfeição.

    ◦ **Diversidade**: Buscar diferentes habilidades, experiências e perspectivas na formação do time.

    ◦ **Reflexão**: Aprender com triunfos e fracassos, tornando a retrospectiva parte da rotina.

    ◦ **Fluxo**: Buscar um fluxo contínuo de entrega de valores com ritmo sustentável.

    ◦ **Oportunidade**: Problemas são oportunidades para melhorar e aprender.

    ◦ **Redundância**: Aplicar muitas medidas de qualidade (programação em pares, testes, integração contínua) como linhas de defesa.

    ◦ **Falha**: Ter coragem de experimentar alternativas quando o caminho não é claro, prototipando se necessário.

    ◦ **Qualidade**: **Não é negociável no XP**. Pode-se limitar o escopo, mas não cortar a qualidade, mantendo um código limpo em pequenos passos (baby steps).

    ◦ **Responsabilidade**: Não pode ser imposta, mas aceita, criando um ambiente colaborativo.

• **Práticas do XP**: Evoluíram com foco na melhoria contínua e entrega de valor. Incluem:

    ◦ **Time Coeso (Whole Team)**: Equipe composta pelo cliente e time de desenvolvimento.

    ◦ **Jogo de Planejamento (Planning Game)**: Desenvolvedores e cliente priorizam e estimam funcionalidades em iterações semanais.

    ◦ **Pequenas Versões (Small Releases)**: Liberação de pequenas versões funcionais para aceitação do cliente.

    ◦ **Testes de Aceitação (Customer Tests)**: Testes desenvolvidos pelo cliente para aceitar requisitos.

    ◦ **Padrões de Codificação (Coding Standard)**: Acordos para regras e padrões de código.

    ◦ **Ritmo Sustentável (Sustainable Pace)**: Trabalhar com foco na qualidade e ritmo saudável.

    ◦ **Metáfora (Metaphor)**: Traduzir as palavras do cliente para o significado esperado.

    ◦ **Integração Contínua (Continuous Integration)**: Integrar novas funcionalidades no repositório para validação, testes e feedback técnico.

    ◦ **Posse Coletiva (Collective Ownership)**: O código fonte não tem dono; toda a equipe conhece todas as partes do sistema.

    ◦ **Desenvolvimento Orientado a Testes (Test Driven Development - TDD)**: **Primeiro, os testes unitários são criados, depois o código é desenvolvido para que os testes funcionem**. Os testes servem como definição e documentação de requisitos.

    ◦ **Refatoração (Refactoring)**: Processo de **melhoria contínua da programação**, mitigando erros e mantendo compatibilidade com o código existente. Requer boa cobertura de testes automatizados.

    ◦ **Projeto Simples (Simple Design)**: Desenvolver o mínimo necessário, de acordo com a expectativa do cliente.

    ◦ **Programação em Pares (Pair Programming)**: Dois desenvolvedores atuam juntos, um codificando e outro acompanhando, para revisão e redução de defeitos.

    ◦ **Reuniões em Pé (Stand-up Meeting)**: Reuniões rápidas sobre tarefas realizadas, a realizar e impedimentos.

• **Papéis do XP**: Não há rigidez, mas são úteis para times iniciantes.

    ◦ **Cliente**: Conhece o negócio, suas regras e define prioridades.

    ◦ **Desenvolvedores/Time**: Estimam esforço para funcionalidades e tarefas, escrevem testes automatizados.

    ◦ **Coach**: Conhece bem as práticas e valores do XP, sinalizando ao time quando algo não está alinhado para a melhoria contínua.

    ◦ **Tracker**: Coleta métricas do projeto e identifica bloqueios com a equipe.

• **Desenvolvimento Orientado a Testes (TDD)**:

    ◦ Também chamado de Test First Development, é uma abordagem onde os **casos de teste são desenvolvidos para especificar e validar o que o código fará**.

    ◦ O desenvolvedor escreve novo código somente se um teste automatizado falhar, evitando duplicação.

    ◦ **Níveis de Adoção**:

        ▪ **Aceitação TDD (ATDD)**: Foca em um único teste de aceitação (comportamento geral do sistema). Também conhecido como Behavioral Driven Development (BDD).

        ▪ **Desenvolvedor TDD**: Foca em um único teste de desenvolvedor (teste unitário para pequenas funcionalidades).

    ◦ **Objetivo Principal**: Especificar requisitos detalhados e executáveis com a mentalidade **Just In Time (JIT)**, focando nos requisitos necessários no momento para aumentar a eficiência.

    ◦ **Benefícios**: **Qualidade embutida** (programador responsável, erros encontrados cedo), **design evolutivo** (aprendizado contínuo, refatoração), **adaptabilidade** (feedback instantâneo, correções rápidas), e **fluxo otimizado** (menos tempo com bugs, maior produtividade, testes como documentação).

--------------------------------------------------------------------------------

**3. Conclusão**

Tanto o Kanban quanto o XP compartilham o objetivo de **otimização de fluxo de trabalho, busca pela melhoria contínua e entrega de valor**. A escolha do método mais adequado depende do **contexto** em que o produto é desenvolvido, incluindo a natureza do produto, a maturidade do time e o mindset da organização. É possível que esses dois métodos sejam **adotados simultaneamente** por um time ágil, através de **tailoring** (adaptação e customização) e experiências curtas com feedbacks rápidos, reforçando que um forte embasamento nos princípios ágeis permite ao time encontrar a melhor forma de gerar valor.

Exercícios:

Interface gráfica do usuário, Texto, Aplicativo, Email

O conteúdo gerado por IA pode estar incorreto.

Interface gráfica do usuário, Texto, Aplicativo, Email

O conteúdo gerado por IA pode estar incorreto.

Interface gráfica do usuário, Texto, Aplicativo, Email

O conteúdo gerado por IA pode estar incorreto.

Interface gráfica do usuário, Texto, Aplicativo, Email

O conteúdo gerado por IA pode estar incorreto.

Interface gráfica do usuário, Texto, Aplicativo

O conteúdo gerado por IA pode estar incorreto.

Interface gráfica do usuário, Texto, Aplicativo, Email

O conteúdo gerado por IA pode estar incorreto.

Interface gráfica do usuário, Texto, Aplicativo, Email

O conteúdo gerado por IA pode estar incorreto.

Interface gráfica do usuário, Texto, Aplicativo

O conteúdo gerado por IA pode estar incorreto.

Interface gráfica do usuário, Texto, Aplicativo, Email

O conteúdo gerado por IA pode estar incorreto.

Interface gráfica do usuário, Texto, Aplicativo

O conteúdo gerado por IA pode estar incorreto.

Interface gráfica do usuário, Texto, Aplicativo, Email

O conteúdo gerado por IA pode estar incorreto.

Interface gráfica do usuário, Texto, Aplicativo, Email

O conteúdo gerado por IA pode estar incorreto.

Interface gráfica do usuário, Texto, Aplicativo, Email

O conteúdo gerado por IA pode estar incorreto.

Interface gráfica do usuário, Texto, Aplicativo, Email

O conteúdo gerado por IA pode estar incorreto.