Módulo 1: Valores e princípios do Manifesto Ágil

Este módulo foca em compreender a essência do conceito de agilidade, que tem como cerne a cultura de **entrega frequente de valor, com qualidade e foco em pessoas**, sejam elas clientes ou quem produz o produto. Os principais objetivos são reconhecer os valores e princípios do Manifesto Ágil.

• **Manifesto Ágil**: Um marco no desenvolvimento de software e gestão de projetos, assinado em 12 e 13 de fevereiro de 2001, em Snowbird, Utah, EUA, por 17 expoentes como Kent Beck, Jeff Sutherland e Ken Schwaber. Muitos dos signatários já utilizavam métodos como XP e Scrum, cujos conceitos foram incorporados ao manifesto.

• **"Cebola Ágil"**: Uma metáfora para a relevância na transformação ágil, onde **valores e princípios são fundamentais para a criação de um *mindset* ágil**, ao contrário de processos e ferramentas, que podem ser impostos sem gerar uma cultura.

• **Propagação**: Embora nascido para o desenvolvimento de software, os benefícios da agilidade se espalharam para diversas áreas como RH, jurídico, comercial e inovação.

Os **quatro valores** do Manifesto Ágil são:

1. **Indivíduos e interações mais que processos e ferramentas**: Prioriza as pessoas e as relações entre elas, focando em dar condições para as equipes executarem suas tarefas, em vez de engessar o trabalho com processos ou supervalorizar ferramentas.

2. **Software em funcionamento mais que documentação abrangente**: Prioriza o produto funcional nas mãos do cliente. Não significa que projetos ágeis não tenham documentação, mas sim que o produto entregue é mais importante do que uma documentação extensa.

3. **Colaboração com o cliente mais que negociação de contratos**: Enfatiza a necessidade de colaboração, compreensão das dores e expectativas do cliente, e ciclos curtos de feedback para validação, o que é mais vital que apenas o contrato.

4. **Responder a mudanças mais que seguir um plano**: Em contextos complexos e incertos, é mais importante experimentar, validar e adaptar-se às mudanças do que seguir rigidamente um plano pré-definido. Agilidade tem planejamento, mas a adaptabilidade é chave.

Os **doze princípios** do Manifesto Ágil são preceitos básicos para a condução de práticas e métodos ágeis:

1. **Nossa maior prioridade é satisfazer o cliente através da entrega contínua e adiantada de software com valor agregado**: Ciclos curtos de desenvolvimento permitem inspeção e adaptação, validando o valor entregue ao cliente.

2. **Mudanças nos requisitos são bem-vindas, mesmo tardiamente no desenvolvimento**: Mudanças são vistas como uma vantagem competitiva para o cliente em um mundo complexo.

3. **Entregar frequentemente software funcionando, de poucas semanas a poucos meses, com preferência à menor escala de tempo**: Reforça o produto funcionando como medida de progresso e busca feedback rápido.

4. **Pessoas de negócio e desenvolvedores devem trabalhar diariamente em conjunto por todo o projeto**: Reduz ruídos de comunicação e promove entendimento através da comunicação face a face.

5. **Construa projetos em torno de indivíduos motivados**: Criar um ambiente favorável, com suporte e confiança, para engajar os colaboradores.

6. **O método mais eficiente e eficaz de transmitir informações para e entre uma equipe de desenvolvimento é através de conversa face a face**: Presume equipes pequenas para facilitar a comunicação. Em grandes projetos, técnicas de escalabilidade são usadas.

7. **Software funcionando é a medida primária de progresso**: A melhor medida de progresso é a entrega de uma parte do produto potencialmente utilizável ao cliente, não apenas relatórios de percentual.

8. **Os processos ágeis promovem desenvolvimento sustentável**: Preza pela qualidade de vida dos trabalhadores e pelo ritmo de trabalho constante, evitando sobrecarga (como horas extras), para que o time mantenha um ritmo sustentável indefinidamente.

9. **Contínua atenção à excelência técnica e bom design aumentam a agilidade**: Evita "dívidas técnicas" (código mal feito, sem testes) que comprometem a capacidade da equipe de entregar novas funcionalidades no futuro.

10. **Simplicidade é essencial**: Buscar soluções simples para problemas, entregando valor com menos recursos e tempo (abordagem Lean), e evitando custos agregados de testes elaborados e riscos de dívida técnica.

11. **As melhores arquiteturas, requisitos e designs emergem de equipes auto-organizáveis**: Times auto-organizados, com perfis diferentes, definem a melhor forma de construir o produto, levando a melhores soluções.

12. **Em intervalos regulares, a equipe reflete sobre como se tornar mais eficaz e então refina e ajusta seu comportamento de acordo**: Os times ágeis realizam retrospectivas para identificar o que deu certo/errado e planejar melhorias para o próximo ciclo.

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

Módulo 2: As principais características do método Extreme Programming (XP)

O Extreme Programming (XP) é um **método de desenvolvimento de software** criado por Kent Beck, Ward Cunningham e Ron Jeffries, sendo totalmente **aderente aos valores e princípios do Manifesto Ágil**. Ele é fundamentado em seus próprios valores e práticas.

Os **cinco valores do XP** são (segundo Wildt, 2015):

• **Comunicação**: Considerada a principal causa de fracassos em projetos de software. Dada a incerteza no desenvolvimento, a comunicação deve ser priorizada, especialmente em equipes remotas.

• **Feedback**: Feedback rápido e frequente minimiza riscos. Permite que o cliente perceba suas reais necessidades e que o time faça adaptações precocemente.

• **Simplicidade**: Ajuda o time a focar no valor real a ser entregue, evitando desperdícios.

• **Respeito**: Deve ser praticado por todos (membros do time, clientes e *stakeholders*). Cria um ambiente agradável, forma times de alta performance e contribui para a qualidade do produto.

• **Coragem**: Estimulada pelo respeito, permite que os membros do time façam o que é preciso (ex: refatorar código) em um ambiente "safe to fail" (seguro para falhar), sem medo de penalidades.

As **práticas do Extreme Programming** são:

• **Equipe Inteira**: Colaboração constante da equipe com partes interessadas (como o cliente) para requisitos e prioridades. A equipe deve ser formada por generalistas.

• **Jogo de Planejamento**: Onde cliente e time de desenvolvimento se reúnem para decidir as entregas da próxima *release* e iterações. O cliente define escopo, prioridade e data, e o time estima o trabalho (histórias de usuário) e dependências.

• **Pequenas Entregas**: Entregas pequenas e frequentes são essenciais para obter feedback rapidamente.

• **Testes de Aceitação (Cliente)**: O cliente define critérios de aceitação para requisitos, e a equipe desenvolve e implementa testes automatizados para validar o que foi pedido.

• **Propriedade Coletiva do Código**: Códigos são compartilhados entre os membros da equipe para evitar que indivíduos sejam seus únicos "proprietários", aumentando a produtividade e a qualidade pelo compartilhamento de conhecimento.

• **Padrão do Código**: Times XP seguem um padrão de codificação para manter a produtividade coletiva e facilitar o entendimento entre os membros.

• **Ritmo Sustentável**: Mantém um ritmo de trabalho que pode perdurar por muito tempo, evitando variações de carga de trabalho, como horas extras.

• **Metáforas**: Usadas para manter uma linguagem comum no time XP, facilitando a identificação rápida de rotinas ou sistemas (ex: "carrinho de compras" em loja virtual).

• **Integração Contínua**: Códigos são integrados várias vezes ao dia para facilitar ajustes de erros.

• **Testes Unitários**: Fornecem feedback imediato ao desenvolvedor sobre erros no código.

• **Refatoração**: Técnica de melhoria contínua do código para torná-lo mais elegante e legível, minimizando problemas como duplicidade, sem alterar o comportamento da funcionalidade.

• **Design Simples**: O design do software deve começar e permanecer simples, mesmo com o incremento de funcionalidades.

• **Programação em Par**: Códigos são desenvolvidos por dois programadores em uma mesma máquina, promovendo compartilhamento de código e conhecimento, melhor qualidade e formação de equipes de alto desempenho.

• **História de Usuário**: Técnica comum no XP (e também no Scrum), que utiliza um formato simples para coletar detalhes do cliente através de conversação face a face (ex: "Eu, como médico, quero pesquisar prontuário do paciente pelo número do CPF"). As práticas do XP são tão consolidadas que são usadas com outros métodos ágeis, como o Scrum.

Os **papéis do XP** visam equilibrar as responsabilidades no desenvolvimento de software:

• **Desenvolvedor**: Responsável direto pela criação do software com qualidade, desde a estimativa até a entrega.

• **Cliente**: O maior conhecedor do negócio (similar ao Product Owner no Scrum). Define requisitos (histórias de usuário), prioriza-os e fornece feedback rápido.

• **Coach**: Apoia o time na assimilação dos valores e práticas do XP, atuando como um líder servidor que facilita cerimônias e remove impedimentos. Pode acumular o papel de desenvolvedor.

• **Testador**: Garante a qualidade do software, apoia o cliente na escrita de testes de aceitação e automatiza testes funcionais.

• **Cleaner**: Indivíduo responsável por melhorar o código (refatoração), promovendo excelência técnica e evitando dívida técnica.

• **Tracker**: Coleta métricas do time (ex: histórias de usuário entregues) para avaliar a evolução e prever entregas futuras.

• **Gerente**: Auxilia o time XP na comunicação com clientes e *stakeholders*, elabora relatórios de evolução. Atua como líder servidor, apoiando o time, não como chefe.

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

Módulo 3: As principais características do framework Scrum

O Scrum é um **framework** criado por Ken Schwaber e Jeff Sutherland, sob influência do artigo "The New New Product Development Game". Foi apresentado publicamente em 1995. É **leve e simples de entender, mas difícil de aplicar**, usado para gerenciar trabalho em produtos complexos. O nome "Scrum" vem de uma jogada de *rugby*, simbolizando o trabalho unido em prol de um objetivo comum.

• **Teoria do Scrum**: Baseia-se no **empirismo**, onde o conhecimento vem da experiência. Em ambientes complexos, com incertezas, o Scrum propõe "fazer, aprender e então projetar" entregas futuras com base na velocidade média do time em iterações anteriores.

Os **três pilares** que apoiam a implementação do controle de processo empírico são:

1. **Transparência**: Garante que todos no time Scrum e *stakeholders* tenham o mesmo entendimento sobre a situação da entrega do incremento do produto e o que é uma "entrega pronta".

2. **Inspeção**: O Scrum possui cerimônias que permitem inspecionar o incremento do produto e o atingimento da meta da *sprint* (ex: reunião diária, reunião de revisão).

3. **Adaptação**: Permite a rápida adequação do que não está de acordo. As *sprints* (iterações) de até 30 dias proporcionam ciclos de desenvolvimento curtos e feedbacks rápidos, possibilitando a adaptação do produto à expectativa do cliente.

Os **cinco valores** do Scrum, essenciais para criar um ambiente de confiança e fortalecer os pilares, são:

• **Coragem**: Para fazer o que é certo, reportar problemas e ouvir opiniões divergentes.

• **Foco**: Manter o foco nas entregas prioritárias do negócio com simplicidade e objetividade durante o ciclo de desenvolvimento.

• **Comprometimento**: Empenho, atitude e engajamento do time com os resultados, reconhecendo a complexidade e incertezas.

• **Respeito**: Não só o respeito interpessoal, mas também o respeito ao cliente, entregando software de qualidade.

• **Abertura**: Estar aberto à transparência, colaborar, ser assertivo, e receber críticas construtivas para melhorar o comportamento profissional, processo ou produto.

O **Time Scrum** é um modelo otimizado para flexibilidade, criatividade e produtividade, com papéis bem equilibrados. É:

• **Auto-organizável**: Não precisa de um chefe para dizer como desenvolver as atividades; o time define a melhor forma de atingir o objetivo.

• **Multifuncional**: Possui os perfis necessários para desenvolver o produto de forma iterativa e incremental, maximizando o valor das entregas.

• **Tamanho**: Deve ter entre **3 e 9 componentes** para não perder a capacidade de comunicação e auto-organização.

• **Papéis**: Embora um gerente de projetos possa atuar em um projeto com Scrum, ele não é um papel preconizado no framework. Os papéis são:

    ◦ **Product Owner (PO)**: O "dono do produto". Define e transmite a visão do produto, gerencia o *Product Backlog* (ordenando itens por valor de negócio), interage com *stakeholders* para identificar *features* ou tendências, e é o único com autoridade para cancelar uma *sprint*.

    ◦ **Scrum Master (SM)**: O maior conhecedor do Scrum. Atua como um **líder servidor**, apoiando o time Scrum e a organização no entendimento do framework, ajudando o time a abraçar seus valores e práticas, removendo impedimentos e facilitando os eventos Scrum.

    ◦ **Time de Desenvolvimento (DEV Team)**: Equipe multifuncional e auto-organizável que desenvolve o produto. Eles puxam itens do *Product Backlog* para o *Sprint Backlog* com base em sua capacidade, estimam o trabalho (ex: pontos por história via *planning poker*), e se auto-organizam para criar o plano da *sprint*.

Os **Eventos do Scrum** são oportunidades de transparência e inspeção, cada um com um propósito e *time-box* (tempo limite):

• **Sprint Planning (Reunião de Planejamento)**: O primeiro evento da *sprint*. Time e PO debatem a meta da *sprint* e os itens (PBI) que farão parte do *Sprint Backlog*. Dura no máximo **8 horas** para uma *sprint* de um mês.

    ◦ **Meta ou Objetivo da Sprint**: Fundamental para o time, pois define a entrega relevante para o negócio que será realizada, em vez de focar apenas em concluir tarefas. O time se compromete com a meta.

    ◦ **Estimativa**: Geralmente feita em "pontos por história" (estimativa relativa) usando técnicas como *Planning Poker*, que ajuda a equipe a conhecer sua "velocidade" (capacidade de trabalho) para futuras previsões.

• **Sprint**: O "contêiner" para todos os outros eventos. Deve ter no máximo **um mês** de duração e, uma vez definida, não deve ser alterada. O escopo da *sprint* (a meta) não deve ser alterado, assim como o padrão de qualidade. Apenas o PO pode cancelar uma *sprint* se sua meta se tornar obsoleta. O progresso é acompanhado em quadros físicos ou virtuais (ex: colunas *Sprint Backlog*, *Fazendo*, *Feitas*).

• **Daily Meeting (Reunião Diária)**: Evento de no máximo **15 minutos** para o time de desenvolvimento inspecionar o progresso em direção à meta da *sprint*. Cada membro responde: O que fiz ontem? O que farei hoje? Há impedimentos? É um evento do time de desenvolvimento, mas o SM e o PO podem participar.

• **Sprint Review (Revisão da Sprint)**: Evento de no máximo **4 horas** para uma *sprint* de um mês. O time de desenvolvimento apresenta o trabalho feito ao PO e *stakeholders*, e o PO aprova ou não o incremento do produto, que deve atender à "Definição de Pronto" (DoD).

• **Sprint Retrospective (Retrospectiva da Sprint)**: O último evento da *sprint*, com no máximo **3 horas** para uma *sprint* de um mês. Propósito de inspeção do próprio time para identificar o que deu certo/errado e criar um plano de melhoria para as próximas *sprints*. O SM facilita para que o time tenha um ambiente seguro para discutir problemas e melhorar continuamente.

Os **Artefatos do Scrum** representam o trabalho e o valor, maximizando a transparência e oportunidades para inspeção e adaptação:

• **Product Backlog**: Uma lista ordenada de PBI (*Product Backlog Items*) que podem incluir histórias de usuário, itens técnicos e correções, organizados por valor de negócio. É um artefato em constante mudança e refinamento pelo PO.

• **Sprint Backlog**: Os PBI selecionados do *Product Backlog* na *Sprint Planning* para serem realizados na *sprint* atual.

• **Incremento (Increment)**: O produto do trabalho realizado na *sprint*. Deve estar "pronto" conforme o padrão de qualidade estipulado na "Definição de Pronto" (DoD) e é potencialmente utilizável após a aceitação na *Sprint Review*.

• **Definição de "Pronto" (Definition of Done - DoD)**: Uma definição clara sobre o padrão de qualidade que a entrega possuirá. Garante transparência para o time Scrum e *stakeholders* sobre o trabalho da *sprint*.

**Como Rodar o Scrum** em um ciclo contínuo:

1. Identificar e capacitar SM e PO.

2. Compor o time de desenvolvimento com especialistas.

3. SM apoia o PO na criação do *Product Backlog* priorizado.

4. Na *Sprint Planning*, o PO apresenta histórias candidatas e o time puxa os itens para o *Sprint Backlog*.

5. PO e time definem a meta da *sprint*.

6. O time desenvolve o incremento, realizando *Daily Meetings* para inspecionar o andamento.

7. Na *Sprint Review*, o time apresenta a entrega ao PO, que aceita ou rejeita.

8. O time realiza a *Sprint Retrospective* para melhoria contínua.

9. O ciclo se repete até a entrega completa do produto.

Exercícios:  
  
Interface gráfica do usuário, Texto, Aplicativo

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.

Texto

O conteúdo gerado por IA pode estar incorreto.

Interface gráfica do usuário, Texto, Aplicativo, Email

O conteúdo gerado por IA pode estar incorreto.

Tela de celular com texto preto sobre fundo branco

O conteúdo gerado por IA pode estar incorreto.

Interface gráfica do usuário, Texto, Aplicativo, Email

O conteúdo gerado por IA pode estar incorreto.

Texto

O conteúdo gerado por IA pode estar incorreto.

Interface gráfica do usuário, Texto, Aplicativo, Email

O conteúdo gerado por IA pode estar incorreto.

Texto

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, 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.

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.