GPS - Gestão de Projetos de Software

Aula 8: Fundamentos Ágeis e Scrum

Prof. Aléssio Miranda Júnior
alessio@cefetmg.br

CEFET-MG - Campus Timoteo
Dep. Engenharia de Computação

Novembro de 2026

Sumário

Objetivos da Aula

- Compreender a quebra de paradigma do **Manifesto Ágil** (2001).
- Diferenciar *Filosofia Ágil* de *Framework Tático* (**Scrum**).
- Entender a genialidade da **Inversão do Triângulo de Ferro**.
- Dominar os **Três Pilares do Empirismo** (Transparência, Inspeção, Adaptação).
- Conhecer a **Autoridade Distribuída** (PO, Scrum Master e Developers).
- Destrinchar as **4 Cerimônias Oficiais** do Scrum.
- Identificar as armadilhas do **Teatro Ágil** corporativo.
- Mapear os artefatos do Scrum para as ferramentas nativas do **GitLab**.

O Mentor Adverte

O Fim do Mundo Perfeito

"Bem-vindo à Gestão Ágil, garoto. O modelo Cascata nos ensinou a planejar e prever, mas a realidade é que o mundo real muda mais rápido do que você consegue atualizar um cronograma no MS Project."

Neste oitavo capítulo, vamos ativar o protocolo Ágil. É o nosso reator principal para lidar com o caos, falhar rápido, e entregar um *Assistente de IA* funcional iterativamente, sem perder a sanidade do grupo de desenvolvedores.

A Era das Trevas (Anos 90)

Antes de 2001, o modelo **Cascata (Waterfall)** dominava o mundo do software.

- Engenheiros passavam *meses* documentando requisitos em Word, tentando prever o futuro.
- Arquitetos desenhavam diagramas estáticos complexos, assumindo que as APIs não mudariam.
- Desenvolvedores passavam 1 ano programando isolados do mundo externo.
- **O Desastre Clássico:** No dia do lançamento, descobriam que o cliente havia mudado de ideia há 6 meses, ou que o mercado já não usava mais aquela tecnologia de front-end.

O custo da mudança no final do ciclo era colossal, levando a projetos abortados e milhões perdidos.

A Rebelião de Snowbird (2001)

Em 2001, 17 líderes de engenharia de software isolaram-se em um resort de esqui em *Snowbird, Utah*. Cansados da burocracia que destruía o software, eles redigiram o **Manifesto Ágil**.

A grande revelação

Agilidade não é usar Post-its coloridos na parede. Não é trabalhar sem camisa virando a noite à base de energético. Ágil é uma **Filosofia de trabalho**.

Essa filosofia revolucionou a forma como o mundo constrói software, abolindo a crença no controle absoluto e baseando-se em apenas **quatro valores fundamentais**.

Valor 1: Indivíduos acima de Processos

1. **Indivíduos e interações** mais que processos e ferramentas.

   - O Jira ou o GitLab mais caros do mundo não salvam uma equipe que se odeia ou que não se comunica.
   - O talento e a troca franca de ideias resolvem problemas matemáticos e de arquitetura melhor que regras engessadas de um manual de fábrica.

Valor 2 e 3: Software Vivo e o Cliente

2. **Software em funcionamento** mais que documentação abrangente.
   - Ninguém paga por um manual em PDF de 300 páginas se o botão de "Login" do sistema está quebrado. O único métrico real de progresso é código que executa.

3. **Colaboração com o cliente** mais que negociação de contratos.
   - O cliente (ou Professor) não é seu inimigo em um tribunal. Construa com ele (mostrando protótipos quinzenais) em vez de jogar o contrato na cara dele meses depois.

4. Responder a mudanças mais que seguir um plano rígido.

O Perigo do Extremismo e da Preguiça

Muitos programadores usam o Ágil como desculpa para dizer: "Sou ágil, logo me recuso a planejar e documentar!". O Ágil **não** prega que planos e documentação são inúteis!

O final do manifesto é claro: *"Ou seja, mesmo havendo valor nos itens à direita, valorizamos* **mais** *os itens à esquerda."*
Como o Mentor costuma dizer: *O único código sem bugs é aquele que nunca foi escrito. Coloque o sistema quebrado na frente do usuário o quanto antes e aprenda.*

O Fracasso do Triângulo Tradicional

Na gestão preditiva (Cascata), o **Escopo** (A lista exata de Requisitos e Telas) é fixado e assinado em pedra antes da primeira linha de código ser escrita.

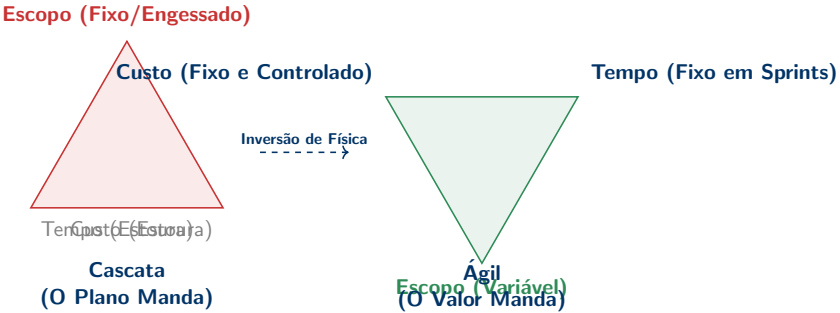
- Se o projeto se provar mais complexo (ex: o LLM demora mais para indexar os dados do que o esperado)...
- Como o Escopo não pode diminuir de tamanho, a Gerência é forçada a estourar a geometria:
  1. Estourar o **Tempo** (Atrasar a data de entrega e irritar o cliente).
  2. Estourar o **Custo** (Pagar horas extras insanas ou contratar *freelancers* urgentes).

A Matemática do Ágil

O Ágil não aceita estourar o orçamento e a paciência do cliente. Ele **inverte a física** do triângulo de projetos.

- O **Tempo** agora é fixo e sagrado: Trabalhamos em ciclos inalteráveis de tempo (ex: 2 semanas).
- O **Custo** agora é fixo e contínuo: A equipe tem sempre as mesmas 5 pessoas recebendo o mesmo salário, sem horas extras heroicas.
- **E o que varia?** Apenas o **Escopo**. Se não der tempo, cortamos as *features* de baixa prioridade (ex: modo noturno) que estavam no fundo da fila. O cliente recebe no prazo combinado.

Visualizando a Inversão



Filosofia vs Framework

- **O Ágil** é a religião corporativa (a filosofia intangível).
- **O Scrum** é o culto prático em que nos reunimos (o *framework* tático e prescritivo).

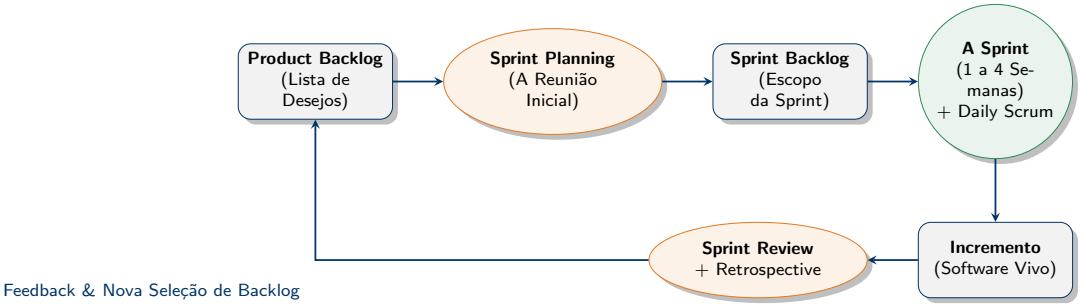
O Scrum é fundamentado no modelo científico do **Empirismo**: o conhecimento técnico nunca advém da imaginação ou da clarividência; ele só surge da experiência prática. Não tente adivinhar se a biblioteca funcionará; instale-a, crie um *Hello World* e descubra a verdade!

Os Três Pilares Empíricos

Para o motor do Empirismo funcionar sem explodir, toda a equipe tem a obrigação de sustentar três vigas de suporte:

1. **Transparência:** Ninguém omite código defeituoso. Todos (incluindo o chefe) veem o *Kanban Board* real e os *commits* defeituosos no GitLab. A verdade liberta a equipe do estresse da mentira.
2. **Inspeção:** A equipe interrompe a codificação e audita severamente o que foi construído em ciclos curtos, analisando métricas (ex: O *Assistente IA* respondeu muito devagar?).
3. **Adaptação:** Se as métricas estiverem ruins, a equipe descarta o código inútil e altera a rota de arquitetura no mesmo dia. Não espere 6 meses para refatorar!

A Anatomia do Motor (Fluxo Scrum)



1. Sprint Planning

A **Sprint** é o próprio coração do Scrum. É o *Timebox* fixo (1, 2, 3 ou 4 semanas) no qual as coisas acontecem.

Ela é iniciada pela **Sprint Planning** (O Planejamento):

- O Product Owner apresenta as ideias e prioridades mais urgentes de negócios.
- *O Segredo do Puxar:* O chefe NÃO empurra as tarefas para os operários. São os *Developers* que, olhando o tamanho do timebox, **puxam** do topo da fila apenas a quantidade de *Issues* que eles acreditam tecnicamente ser possível entregar em 14 dias com qualidade.

2. Daily Scrum (A Reunião Diária)

Todos os dias, no mesmo horário, a equipe técnica para tudo e se reúne por cravados **15 minutos** em pé (para ninguém sentar e contar histórias longas).

NÃO é uma reunião para dar status ao gerente. É uma verificação de impedimentos entre os programadores, focada em 3 respostas rápidas:

1. O que eu fiz ontem para ajudar a atingir a meta da Sprint?
2. O que farei hoje?
3. Existe algum impedimento técnico bloqueando o meu código ou o da equipe? (Aqui o *Scrum Master* anota a missão dele de destravamento do dia).

3. Review e 4. Retrospective

O fim do Timebox exige inspeção do produto e do processo:

- **Sprint Review (O Produto):** A equipe técnica demonstra o *Software em Funcionamento* real (o Incremento) para o Cliente/Sponsor. PowerPoint é proibido. Se você codificou o banco de dados, mostre o script rodando. É a hora do *feedback* externo.
- **Sprint Retrospective (O Processo):** Cliente não participa. Apenas o time interno lava a própria roupa suja tecnológica e pessoal. O que fizemos bem no DevOps? Onde a comunicação falhou entre Front e Back? Como podemos automatizar para não sofrer de novo amanhã?

A Morte do Chefe Centralizador

A maior ruptura do Scrum com empresas conservadoras é a **extinção e fracionamento do Gerente de Projetos Tradicional** (o ditador do MS Project).

O Scrum quebra esse poder absoluto para evitar que um chefe de férias atrase o lançamento do aplicativo. O modelo é de **autoridade distribuída e autônoma** através de três papéis invioláveis:

- O Product Owner (A Visão Financeira).
- O Scrum Master (O Guardião Ágil).
- Os Developers (A Força Motriz de Engenharia).

Product Owner (O Dono do "O QUE")

O **PO** é o elo direto com o cliente que paga a conta.

- Detém autoridade máxima sobre *O QUE* será construído.
- É o **único** responsável autorizado a reordenar a fila de desejos do *Product Backlog*, colocando no topo o que traz maior Lucro (ROI) e valor aos usuários finais.
- Ele *não pode* dizer para a equipe como fazer o banco de dados. Ele diz "eu preciso de uma tela que liste clientes rápido".

Scrum Master (O Guardião do Processo)

- O **SM** atua como *Líder-Servidor*. Ele NÃO é o chefe dos programadores e não julga a velocidade deles.
- A missão dele é garantir que o Ágil não vire bagunça corporativa. Ele obriga a equipe a focar e participar das cerimônias corretas.
- Atua como um trator limpando neve: se o servidor da AWS cair e travar o programador Sênior, o SM para tudo, abre um chamado ou liga para o Suporte. O foco do programador continua intacto no código, e o SM resolve o fogo.

Os Developers (A Autonomia Técnica)

Autonomia Total sobre o "COMO"

Os Developers são o esquadrão multidisciplinar de trincheira. Programadores, Arquitetos, QA, UI/UX. Todos são chamados de "Developers".

- O PO ditou o que é preciso (A tela de listar clientes). A equipe tem poder constitucional e irrevogável de decidir *COMO* (se vão usar Node.js, React ou Assembly).
- Eles são **auto-organizados**: não esperam o SM mandar e não dividem código com base no ego. Dividem para atingir a meta coletiva.

Cuidado com o "Teatro Ágil"

Mais de 60% das empresas hoje cometem **Teatro Ágil**: Fingem usar Scrum para parecerem tecnologicamente modernas no LinkedIn.

A Anatomia da Farsa:

- Obrigam a equipe a ficar em pé 15 minutos de manhã (*Daily* falsa)...
- Mas obrigam a mesma equipe a assinar um contrato com *Escopo Fixo de ferro* que dura 2 anos de desenvolvimento.
- Não há "Adaptação" no processo. É pura e simplesmente o velho modelo Preditivo Cascata vestido com post-its e micro-gerenciamento abusivo do chefe.

O Colapso Bilionário da Nokia (2008-2013)

O Custo Final da Ilusão

A Nokia, gigante inabalável dos celulares e do Symbian, adotou o Scrum de maneira forçada em 2008. Todas as divisões ostentavam *Sprints* e *Scrum Masters* oficiais.

Qual foi o câncer interno? A alta diretoria exigia inovações instantâneas baseadas em escopo imposto goela abaixo, sem tolerar testes errados. Sob pressão, as equipes mentiam na *Review* ("Está tudo perfeito e integrado"). A Nokia não adaptou seu Kernel à revolução do iPhone, o Teatro Ágil cobrou seu preço, e a divisão mobile foi vendida sob humilhação para a Microsoft. Sem honestidade empírica, metodologias são apenas palavras vazias.

Materializando a Teoria na Nuvem (Check 03)

Nenhum gerente usa lousas e canetas para projetos modernos remotos. Como mapear a Teoria Scrum nas tabelas nativas do seu repositório no GitLab?

- **O Product Backlog Vivo:** É toda e qualquer *Issue* cadastrada de forma "solta" na aba geral 'Issues ¿ List', sem estar alocada num prazo específico de entrega.
- **A Sprint (O Timebox):** Nós usamos a aba `Milestones`. Você cria o evento chamado "Sprint 1", travando a data de Início para 01/Out e o Fim exato e fatal para 14/Out. O Milestones força o prazo sobre o sistema.

Sprint Backlog e o Kanban

- **O Sprint Backlog Oficial:** São exatamente as *Issues* solitárias que o seu grupo decidiu puxar, atrelando e alocando-as ao Milestone ativo da vez. Se não estiver no Milestone, a equipe não tem obrigação de construir naquelas semanas.
- **Daily Scrum Assíncrono:** Nós não faremos *Dailies* síncronas de vídeo para o CEFET. A transparência diária e a identificação de gargalos (o que o SM deve destravar) será feita pelo movimento dos *Cards* da esquerda para a direita no **Issue Board Kanban** ('Doing -¿ In Review -¿ Done'). O Quadro é a sua reunião de status definitiva.

Reflexões do Capítulo

Antes de inicializar sua primeira Sprint iterativa, analise criticamente:

1. O Manifesto Ágil manda "Responder a mudanças mais que seguir um plano". Baseado no que você aprendeu com o Professor, isso significa que não devemos fazer a documentação UML nem as Atas ADR nunca mais?
2. Por que a genialidade da inversão física do "Triângulo de Ferro" salva a empresa da falência? (Dica: Se o tempo passa e os desenvolvedores já recebem um salário fixo diário, o que resta para o PO manipular sem quebrar a empresa?)
3. Se, no meio exato da *Sprint* 01 (quarta-feira), o Cliente ligar irritado exigindo a inclusão urgente de um botão "Chat de Voz no Assistente", quem no trio do Scrum tem a obrigação inadiável de barrar e blindar a equipe?

Seu Repositório está Pronto para o Ágil?

Email: alessio@cefetmg.br
Horário de aula: Segunda, 7h às 8:40h
Horário de atendimento: Quarta 16h e whatsapp

Aviso: O Check 03 irá varrer o seu Milestone em busca das estimativas!

Próxima aula: Como escrever requisitos no formato Ágil? Vamos dissecar a taxonomia das **Histórias de Usuário e Épicos**.