GPS - Gestão de Projetos de Software

Aula 14: Análise, Métricas e Revisão

Prof. Aléssio Miranda Júnior
alessio@cefetmg.br

CEFET-MG - Campus Timoteo
Dep. Engenharia de Computação

Novembro de 2026

Sumário

Objetivos da Aula

- Compreender o pilar ágil do **Empirismo**: Inspecionar e Adaptar.
- Conduzir a **Sprint Review** focada no produto real (sem *PowerPoint*).
- Facilitar a **Sprint Retrospective** focada no processo e nas pessoas.
- Ler o **Burndown Chart** e identificar anti-padrões da equipe.
- Diferenciar **Lead Time** de **Cycle Time**.
- Adotar a cultura do **Blameless Post-Mortem** (Autópsia sem Culpa).
- Usar os painéis *Analytics* do GitLab na prática.

O Mentor Adverte

O Conserto Estrutural

"Garoto, agilidade não é acertar tudo de primeira feito um robô de fábrica; é sobre consertar os danos estruturais mais rápido do que a concorrência consegue piscar."

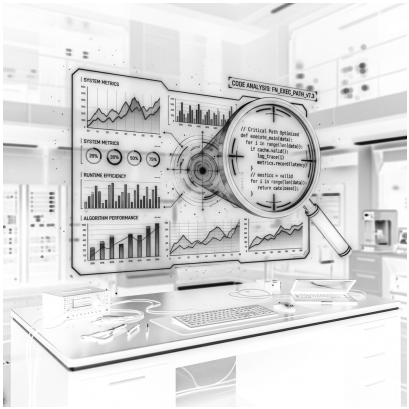
Neste penúltimo capítulo, vamos instalar os protocolos de diagnóstico pesado: validar o armamento com o cliente, consertar o motor da equipe e entender o que explodiu no servidor sem mandar ninguém embora.

O Abismo do Falso Ágil

Se você opera em *Sprints* de duas semanas, mas **nunca para** a produção no último dia para analisar friamente o que foi construído... você não está sendo Ágil.

Você está apenas executando um modelo Cascata fatiado, trabalhando no escuro em um estado de exaustão contínua.

Chegou a hora de ativar os **Radares de Métricas**.



Inspecionar e Adaptar

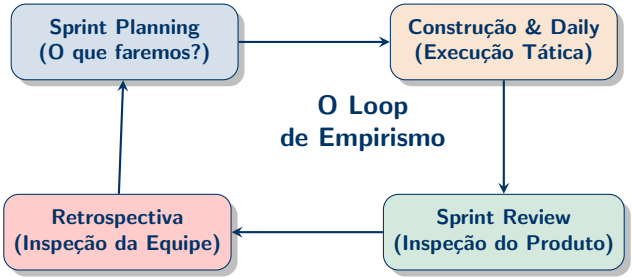
A agilidade de elite é fundamentada no **Controle de Processo Empírico**.

Existem 3 pilares absolutos que sustentam esse ciclo:

1. **Transparência:** As métricas devem ser reais, doa a quem doer, não apenas "o que o chefe quer ouvir".
2. **Inspeção:** Olhar criticamente para o código e para o fluxo de trabalho.
3. **Adaptação:** Mudar a rota imediatamente, sem burocracia, baseado na inspeção anterior.

O Loop Contínuo

As cerimônias do Scrum existem unicamente para forçar a equipe a rodar essa engrenagem de forma oficial e rítmica.



Sprint Review: Focando no "O Quê"

A *Sprint Review* ocorre nas últimas horas da Sprint.

- **Alvo Principal:** O Produto (O Código).
- **Público-Alvo:** Toda a Equipe de Engenharia + *Product Owner* + Principais *Stakeholders* (Cliente, Usuários).
- **Objetivo:** Validar o incremento de software que foi construído nos últimos 15 dias.

PowerPoint está terminantemente BANIDO.

O desenvolvedor jamais deve apresentar um texto explicando o que fez. Ele deve **compartilhar a tela, abrir o sistema em ambiente de Staging e utilizar a funcionalidade crua**.
Se o requisito era o Assistente de IA analisar um PDF, arraste o PDF vivo na reunião. É só nesse choque com a realidade que o cliente grita: *"Funciona, mas o botão ficou muito escondido"*.

O Valor do Feedback Cru

O *feedback* recebido na Review nunca é para "consertar o passado".

Ele gera **novas Histórias** e novos Cards que entram no Backlog do Produto, priorizados para as próximas Sprints. O risco terrível de passar 6 meses construindo a ferramenta errada **morre ali**.

Sprint Retrospective: Focando no "Como"

Após a Review e a saída dos clientes da sala, a equipe técnica (Scrum Master, PO, Devs) fecha a porta. Inicia-se a *Retrospectiva*.

- **Alvo Principal:** A Engrenagem Humana e os Processos da Equipe.
- **Público-Alvo:** Estritamente interno (apenas a equipe técnica e gestores diretos ligados ao fluxo).

A Estrutura Padrão

O *framework* exige dissecar as duas semanas que passaram respondendo friamente a 3 variáveis:

1. **O que funcionou incrivelmente bem?** (Ex: "A integração do GitLab Runner salvou as noites de *deploy*").
2. **O que falhou de forma crítica?** (Ex: "A API da OpenAI caiu por dois dias no meio da Sprint e destruiu nossa Velocity, pois ninguém pôde testar").
3. **O que vamos melhorar na próxima Sprint?** (Ex: "Fazer um *mock* local da IA para testes").

Técnicas de Facilitação: Os 4 Ls e o Veleiro

Reuniões sem dinâmica viram silêncio absoluto. Como facilitar?

1. Os 4 Ls (Liked, Learned, Lacked, Longed For): Cada membro escreve *Post-its*: **Gostei** (positivo), **Aprendi** (descoberta), **Faltou** (deficiência no projeto) e **Desejei** (o que queria que tivesse acontecido).

2. O Veleiro (Sailboat): A equipe desenha no quadro um veleiro: **Vento** (o que nos acelera), **Âncora** (nossas dívidas técnicas), **Rochas** (os perigos e bugs iminentes), e a **Ilha** (nosso objetivo final).

O Cemitério das Retrospectivas Inúteis

A maior falha estrutural de equipes júniores é transformar a Retrospectiva num **"Muro das Lamentações"**.

Action Items

Reclamar que "o prazo é curto e o cliente muda tudo" por 2 horas é terapia coletiva cara. Uma Retrospectiva só tem valor se gerar **Itens de Ação (Action Items)**.
"O Tech Lead enviará um e-mail formal blindando o escopo ao cliente amanhã às 9h." Se não houver uma ação clara e um CPF responsável para resolvê-la, a reunião foi inútil.

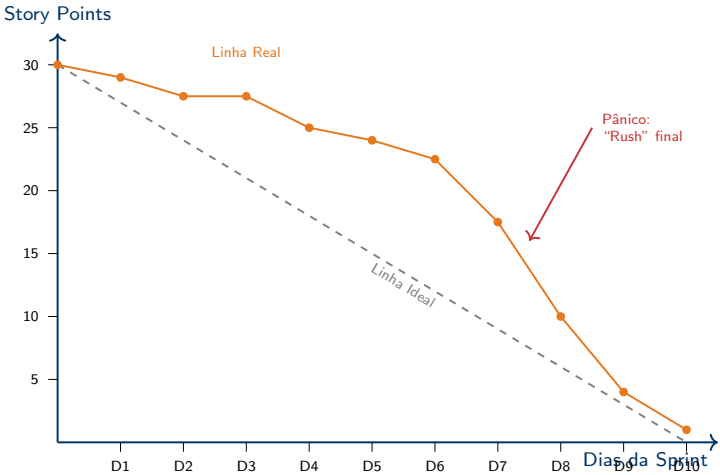
O Termômetro da Sprint

O **Burndown Chart** (Gráfico de Queima) é a métrica visual mais poderosa da agilidade tática.

Ele plota o Trabalho Restante (Story Points ou Cards) vs. o Tempo da Sprint (Dias).

- **Eixo Y:** O peso das tarefas que ainda faltam entregar.
- **Eixo X:** Os dias corridos daquela Sprint.
- **Linha Ideal:** Uma diagonal reta que zera no último dia.

A Anatomia do Burndown



Anti-padrão: O Precipício Acadêmico

Analisando o gráfico anterior, notamos a "Síndrome do Estudante".

A linha de progresso ficou praticamente horizontal (plana) pela primeira semana inteira. Os cartões no quadro permaneceram em *To Do* ou *Doing*. Apenas nas últimas 48 horas da Sprint (D8, D9), a linha despencou brutalmente até zero.
O que isso diagnostica? A equipe não faz *Integração Contínua*. Eles codam isolados a semana toda e "juntam o monstro" de última hora, correndo risco máximo de falha de arquitetura antes da Review.

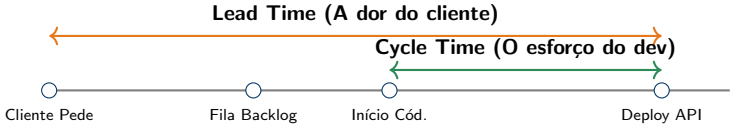
Abandone o "Achismo"

Retrospectivas sem dados geram conflitos emocionais infrutíferos. Para calibrar a máquina da equipe, o Gerente de elite utiliza métricas como as do **Framework DORA** (DevOps Research and Assessment) e de Fluxo Lean.

Dois termômetros dominam o cenário corporativo:

- **Lead Time**
- **Cycle Time**

Lead Time vs Cycle Time



- **Lead Time:** Tempo total percorrido desde o pedido do cliente até o código chegar efetivamente em produção. Mede a burocracia inteira do negócio.
- **Cycle Time:** Tempo desde o primeiro *commit* até a aprovação. Mede *apenas* a velocidade da engenharia.

O Paradoxo do Gargalo

E se o seu *Cycle Time* é de incríveis **2 dias**, mas o seu *Lead Time* sangra por penosos **3 meses**?

Diagnóstico Implacável: O seu time de desenvolvimento (programadores) é ágil e rápido. O código sai veloz. Contudo, o cartão fica meses parado numa esteira de aprovação burocrática, comitês de segurança ou validação gerencial.
Solução: Otimizar os programadores não vai adiantar nada; você precisa destruir os gargalos da gerência.

O Blameless Post-Mortem

Quando a infraestrutura de pagamentos cai no meio da *Black Friday*, catástrofes milionárias ocorrem. O que gigantes como Netflix e Google fazem no rescaldo?

Exigem uma Autópsia de Sistema Sem Culpa (**Blameless Post-Mortem**).
A premissa suprema: *"Assumimos que todos os engenheiros envolvidos agiram com a melhor das intenções, baseados nas informações e ferramentas que possuíam na hora do desastre"*.

Blindando a Arquitetura, Não Culpando as Pessoas

O objetivo dessa cultura não é proteger incompetência, mas garantir a honestidade dos dados. Se o engenheiro tiver medo de ser demitido, ele esconderá *logs* cruciais e mascarará os fatos.

O Post-Mortem se pergunta:

- *"Por que a nossa infraestrutura e permissões de acesso permitiram que o Júnior fizesse um 'DROP TABLE' na produção com apenas um clique sem alertar ninguém?"*

O indivíduo nunca falha isolado; é o sistema que falhou em bloqueá-lo.

Analytics no GitLab (Check 04)

A telemetria do GitLab (Analytics → Issue Analytics / Value Stream) não mente sobre a sua equipe:

- **Monitoramento de Burndown:** Vocês têm o hábito de mover as *Issues* para *Closed* regularmente ao longo da semana, ou só arrastam no desespero da noite de domingo para a nota?
- **Ciclo Real:** O GitLab avisa o tempo entre a *Issue* ser aberta e o *Merge Request* ser acatado. Se a Sprint demora 15 dias, mas o *Merge Request* fica parado 13 dias sem revisão de código, seu gargalo é **Code Review**.

Check Analítico

Análise e Empirismo

O projeto deve operar sob o pilar de melhoria contínua!

- **A Review Fiel:** O *Bot* de vocês deve rodar ao vivo para o cliente testar; nada de promessas verbais ou PDFs bonitos.
- **Action Items:** Se a API limitou as consultas diárias e quebrou a Sprint, qual foi a decisão técnica real tomada na Retrospectiva para colocar *Cache*?
- **Post-Mortem:** Assumam as catástrofes dos repositórios locais (quando alguém apaga a *branch* acidentalmente) consertando a falta de *Protections Rules* do próprio repositório.

Perguntas Reflexivas

1. Qual é a diferença fundamental no protocolo de acesso e foco entre a *Sprint Review* (externo) e a *Sprint Retrospective* (interno)?
2. Se uma equipe faz retrospectivas elogiadas e prazerosas, mas não gera nenhum *Action Item* designado para um responsável na segunda-feira... o que acontecerá inevitavelmente?
3. Sob a engenharia moderna, por que caçar e demitir o desenvolvedor que derrubou o servidor impede que a empresa melhore sua arquitetura a longo prazo?

Obrigado!

Se você não consegue medir, você não consegue consertar.

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

Revisem as Métricas do GitLab!

Próxima aula: O desafio final (Pitch)! Aprenderemos a formatar a apresentação do *MVP* para encantar a banca.