GPS - Gestão de Projetos de Software

Aula 4: Escopo e Cronograma

Prof. Aléssio Miranda Júnior
alessio@cefetmg.br

CEFET-MG - Campus Timoteo
Dep. Engenharia de Computação

Setembro de 2026

Sumário

Objetivos da Aula

Gestão de Escopo (O "Quê")

- **Dualidade:** Escopo do Produto vs. Projeto.
- **EAP (WBS):** Decomposição hierárquica visual.
- **Regra dos 100%:** A fronteira absoluta.
- **Inimigos do Prazo:** Mitigar *Scope Creep* e *Gold Plating*.

Gestão de Cronograma (O "Quando")

- **Granularidade:** Pacotes em Atividades (4h–40h).
- **Estimativas:** PERT, Story Points e Fluxo.
- **Caminho Crítico (CPM):** A magia da folga zero.
- **No GitLab:** Milestones, Issues e dependências.

🎓 **Meta Prática:** *Sair da teoria do PMBOK e estruturar o repositório do Projeto Integrador no GitLab.*

A Mensagem do Mentor

O Erro do Amador

"Escuta aqui, garoto(a). O maior erro de um amador é construir o prédio antes de desenhar a planta."

Neste capítulo, entraremos no núcleo duro do planejamento de engenharia. Vamos definir com precisão matemática **o que** será feito (**Escopo**) e **quando** será entregue (**Cronograma**).

Escopo: *A planta arquitetural*

Cronograma: *O calendário da obra*

O que é o Escopo? A Dualidade Fundamental

No PMBOK, o escopo não é uma declaração vaga: é o contrato basilar do projeto.

Escopo do Produto (O Quê)

O que o software É e FAZ:

- Recursos, funções e regras de negócio.
- Critérios de aceitação validados pelo cliente.

Exemplo: *"Bot Telegram, integração OpenAI, login JWT."*

Escopo do Projeto (O Como)

Todo o trabalho para CONSTRUIR:

- Esforço gerencial, técnico e operacional.
- Reuniões, testes, infraestrutura e CI/CD.

Exemplo: *"Daily meetings, config Docker, testes e EAP."*

Regra dos 100% da EAP: *A EAP engloba AMBOS! Se consome tempo ou recurso da equipe, deve estar na EAP.*

Projeção do Escopo no Tempo: O Elo Bidimensional

Projeção integrada: EAP → Gantt

Eixo Vertical (Y): O "Quê" (EAP / WBS)

Decomposição Hierárquica:

- Fatiamento do projeto até os **Pacotes de Trabalho**.
- Garante que nada fique de fora (*Regra dos 100%*).

Eixo Horizontal (X): O "Quando" (Gantt)

Sequenciamento e Prazos:

- Distribuição das atividades no calendário.
- Mapeamento de **dependências** e marcos (**Milestones**).

Princípio de Engenharia: *Não se estima tempo no escuro. O cronograma nasce da granularidade da EAP.*

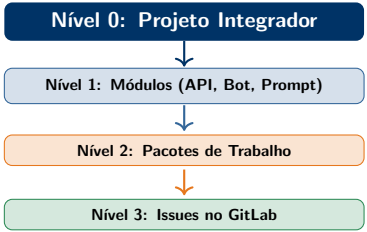
A Ferramenta de Ouro: WBS / EAP

EAP (Estrutura Analítica do Projeto)

Em inglês **WBS** (*Work Breakdown Structure*). É a decomposição hierárquica e visual do escopo total em partes menores e gerenciáveis.

A Metáfora da Árvore:

- **Tronco (Nível 0):** O produto final (ex: Coach ICPC).
- **Galhos (Nível 1):** Módulos e grandes entregas.
- **Folhas (Nível 2+):** Pacotes de Trabalho executáveis.



A Regra dos 100%: Nem Mais, Nem Menos

Está na EAP → É Projeto!

- Engloba **100%** do trabalho acordado.
- Possui orçamento, horas alocadas e responsável.
- Inclui código, testes, documentação e reuniões.

Não está na EAP → Não Existe!

- Se não foi fatiado na EAP, **não deve ser feito**.
- Não recebe recursos nem tempo da equipe.
- Evita o desperdício com recursos não solicitados.

⚖ **Regra Suprema do PMI:** *A soma dos filhos deve ser exatamente igual a 100% do pai. Sem lacunas, sem sobras.*

O Pacote de Trabalho (Work Package)

Descendo a hierarquia da EAP, chegamos à folha da árvore: o **Pacote de Trabalho** (*Work Package*).

- É a **menor unidade do Escopo** gerenciável.
- Agrupamento lógico de esforço para produzir um entregável concreto (ex: *"Módulo de Integração com API"*).
- Permite estimar custo e tempo com confiabilidade.

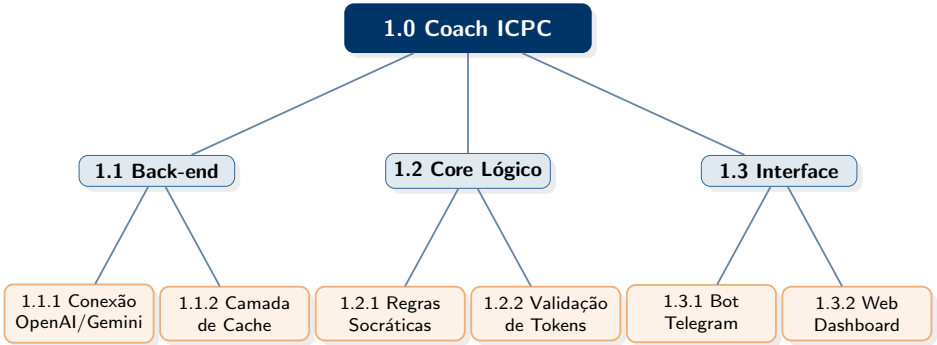
No nosso Projeto

Cada **Pacote de Trabalho** desenhado na EAP vira diretamente uma **Issue** aberta no GitLab.

Granularidade Ideal

Esforço recomendado por pacote: entre **4 e 40 horas**.

A EAP do Assistente IA (Exemplo Real)



Os Destruidores de Prazo

O Plano Parecia Perfeito...

- EAP desenhada e aprovada.
- Horas orçadas e equipe alocada.
- Repositório e CI/CD prontos.

A Cilada da Engenharia

- O maior risco **não é a tecnologia**.
- É a **mudança descontrolada de escopo**.
- Sem gestão, o escopo infla como um balão de ar até estourar o prazo.

A Regra de Sobrevivência: *Existem apenas duas formas do escopo inchar: por fora (***Scope Creep***) ou por dentro (***Gold Plating***).*

Inimigo 1: Scope Creep (Pressão Externa)

Corrupção de Escopo

Inchaço gradual e silencioso do projeto provocado por pedidos informais de clientes/stakeholders **sem revisão de prazo ou orçamento**.

Origem Externa: O cliente pede de boca; o time aceita por simpatia sem registrar issue.

Anatomia do Desastre:

- *"Dá para colocar só mais um botãozinho?"*
- *"Seria ótimo se o bot também enviasse áudio."*
- Não se cria issue nem se ajusta a data final.
- Dez pedidos de 3 horas → **Quase 1 semana de atraso!**

Antídoto

Controle Formal de Mudanças: Registrar no Charter e renegociar prazo!

Inimigo 2: Gold Plating (Iniciativa Interna)

Folheamento a Ouro

Inchaço do escopo gerado pela **própria equipe de desenvolvimento**, adicionando complexidade não solicitada e sem aprovação.

Origem Interna: O programador cria recursos extras *"porque é legal"* ou para testar tecnologia nova.

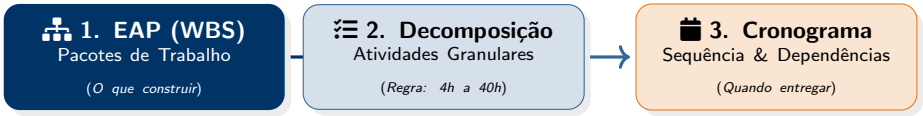
Sintomas Típicos:

- Reescrever o módulo com biblioteca nova experimental.
- Criar animações 3D ou IA complexa onde bastava um menu simples.
- O cliente não pediu, o escopo não cobre, e a entrega atrasa.

Antídoto

Foco no Valor/MVP: Entregar com excelência o combinado antes de inventar moda.

Transformando Pacotes em Tempo



A Regra de Ouro da Granularidade:

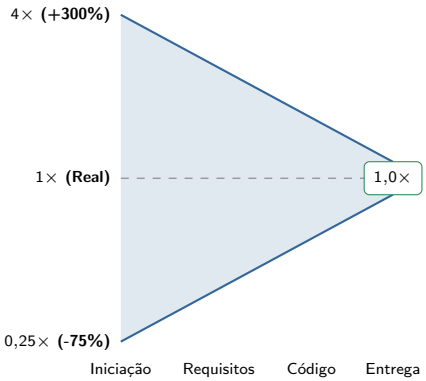
- **Abaixo de 4 horas:** Gera microgerenciamento e sobrecarga de burocracia.
- **Acima de 40 horas:** Gera incerteza descontrolada; precisa ser fatiada em tarefas menores.

Como Estimar Tempo? O Cone da Incerteza

O Cone da Incerteza (McConnell)

No início de um projeto de software, a incerteza absoluta varia de 0,25× **a** 4× o tempo real.

- Toda estimativa inicial é uma **hipótese probabilística**.
- Conforme refinamos a arquitetura e os testes, o cone se fecha em direção a 1,0×.
- Não busque precisão mágica: adote métodos estruturados e conheça seus limites!



Panorama de Métricas e Estimativas

Paradigma	Técnicas / Métricas	Como funciona?	Limitações e Uso
Probabilístico (3 Pontos)	**PERT** / Triangular	Média ponderada entre O, M e P.	Heurística útil; ainda baseada em palpites subjetivos.
Relativo (Ágil)	**Story Points**, Planning Poker	Compara complexidade e esforço (Fibonacci).	Não mede horas de calendário; exige calibrar *velocity*.
Empírico / Fluxo (Lean)	**Lead/Cycle Time**, Monte Carlo, Throughput	Análise estatística sobre o histórico real de entrega.	Exige histórico consolidado; altíssima precisão empírica.
Paramétrico / Tamanho	**Pontos de Função (APF)**, COCOMO II	Índices baseados em entradas, saídas e regras.	Alto custo de modelagem; pouca aderência ao ágil.

Um Exemplo Clássico: O Método PERT

A Média Ponderada do PERT

$$Te = \frac{\mathbf{O} + 4\mathbf{M} + \mathbf{P}}{6}$$

- **O (Otimista):** Cenário perfeito, sem imprevistos.
- **M (Mais Provável):** Cenário padrão com peso 4.
- **P (Pessimista):** Falha de API, rate limit e bugs.

Limitação Fundamental

O PERT **não é infalível**. Ele apenas calcula a média de 3 estimativas subjetivas (*Garbage In, Garbage Out*).

Benefício Real: Obriga a equipe técnica a pensar explicitamente no **pior cenário** antes de fechar o prazo.

Exemplo Prático com PERT

Pacote: Integrar API da OpenAI

O (Otimista): 8 horas (*"Docs clara e chave OK"*)
M (Mais Provável): 16 horas (*"Parsing, exceções e testes"*)
P (Pessimista): 32 horas (*"Rate limit e cache"*)

Cálculo do Tempo Esperado (Te):

$$Te = \frac{8 + (4 \times 16) + 32}{6} = \frac{104}{6} = \mathbf{17{,}3}\ \text{h}$$

Decisão de Gestão

Reservar **18 horas** no cronograma protege o time contra a ilusão das "8 horas otimistas".

Monitoramento

Registrar no GitLab o tempo real gasto via `/spend` para calibrar a precisão da equipe.

Alternativa Ágil: Story Points

Em vez de adivinhar **horas exatas**, métodos ágeis costumam usar **Story Points**:

- Mede **tamanho relativo**, **complexidade** e **incerteza**.
- Sequência de Fibonacci: $1, 2, 3, 5, 8, 13, 21 \ldots$
- **Planning Poker:** Consenso da equipe em vez de estimativa individual isolada.

Por que Pontos?

Humanos são péssimos em estimar tempo absoluto, mas ótimos em comparar grandezas relativas (*"esta issue é o dobro daquela"*).

Velocidade (Velocity): Após 2 Sprints, medimos quantos pontos o time entrega por semana.

A Lição Fundamental: Conheça sua Métrica

PERT / Horas

Trate como estimativa probabilística com margem de risco, nunca como data gravada em pedra.

Story Points

Meça a **Velocidade Real** da equipe ao longo dos Sprints para projetar datas futuras.

Métricas de Fluxo

Acompanhe **Lead Time** e **Cycle Time** no GitLab para eliminar gargalos.

Calibração Empírica

O histórico real de entregas do time vale mais que qualquer fórmula mágica teórica.

Duração vs. Esforço: O Choque de Realidade

Esforço (Effort)

Total de horas brutas de trabalho real no teclado.

Exemplo: *A issue exige* **16 horas** *de programação pura.*

Duração (Duration)

Tempo de calendário decorrido para concluir o esforço.

Exemplo: *O dev só pode dedicar* **2h/dia** $\rightarrow$ *Duração:* **8 dias úteis!**

Atenção no Projeto: *Na faculdade, nenhum aluno trabalha 8h por dia. Calcule o calendário pela capacidade real do time!*

A Rede de Dependências (Fim-para-Início)

Atividades de software não ocorrem no vácuo simultaneamente: elas possuem relacionamentos lógicos.

Dependência Fim-para-Início (F-I)

A Atividade 2 só pode *Iniciar* quando a Atividade 1 for *Finalizada*. Se a API não conecta, é impossível testar as respostas socráticas do bot!

O Caminho Crítico (Critical Path)

Definição do CPM

O **Caminho Crítico** é a sequência mais longa de tarefas dependentes do projeto. Ele dita a data final de entrega.

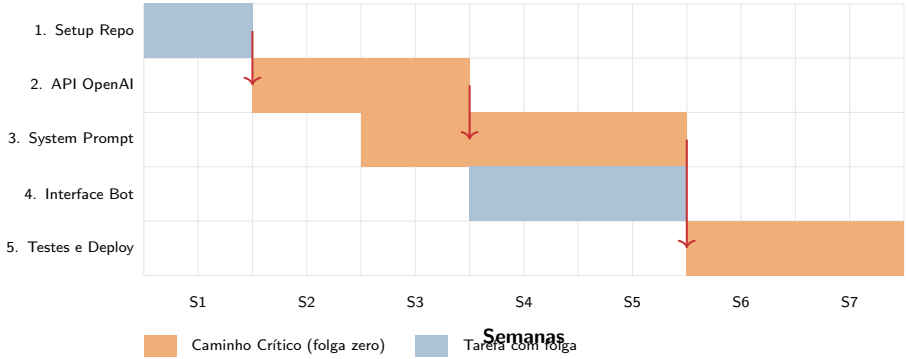
A Magia da Folga Zero

As tarefas no caminho crítico **não possuem folga**. Se atrasarem 1 dia, o projeto inteiro atrasa 1 dia!

Caminho Crítico
Folga $= 0$
(Prioridade máxima do gerente)

Tarefas com Folga
Folga > 0
(Podem atrasar um pouco)

Desenhando o Caminho Crítico (Gantt)



Entendendo o Gráfico

Caminho Crítico (Laranja)

Sequência com Folga Zero:

$$Setup \rightarrow API \rightarrow Prompt \rightarrow Deploy$$

Se a integração com a API atrasar 1 semana, o Deploy atrasará 1 semana.

Tarefa com Folga (Azul)

Interface do Bot: Ocorre em paralelo e termina antes da integração final.

Possui **Folga**: pode atrasar alguns dias sem adiar a data de entrega final.

Foco Gerencial: *O gestor de projetos deve concentrar 80% da sua atenção nas issues que estão no Caminho Crítico.*

Monitoramento e Linha de Base (Baseline)

Linha de Base (Baseline)

- EAP e Cronograma aprovados no Planejamento são **congelados**.
- Servem como o gabarito oficial para medir desvios.

Execução Real (EVM)

- Durante o semestre, cruzamos o planejado com o executado via **GitLab**.
- O EVM mede se o projeto está adiantado ou atrasado.

Princípio: *Sem uma Linha de Base, é impossível afirmar se um projeto está adiantado ou atrasado.*

No GitLab: A EAP e o Cronograma viram Código

1. Pacote de Trabalho → Issue

- Cada caixa da sua EAP vira uma **Issue** aberta.
- *"Não está no GitLab? Não está no escopo."*

2. Decomposição em Atividades

- Use o **Checklist Markdown** dentro da descrição da Issue:
- `- [ ] Criar client HTTP`
- `- [ ] Tratar rate limit`

Rastreabilidade Total: *Commits e Merge Requests devem sempre citar a Issue correspondente (`Closes #12`).*

Tempo e Dependências no GitLab

Estimativas no GitLab

Slash Commands:

- `/estimate 16h` → Define o esforço.
- `/spend 4h` → Registra tempo gasto.
- Ou use o campo **Weight** para Story Points.

Vínculo de Dependências

Mapeando o Caminho Crítico:

- No menu lateral da Issue, use o recurso **Linked Items**.
- Marque: *"Issue #2 is blocked by Issue #1"*.

Regra de Ouro da Equipe: *Todos os integrantes devem ter issues atribuídas e estimadas ao longo de todo o semestre.*

Missão do Grupo

1. Project Charter

Definir bem os limites do produto: o que está **dentro** e expressamente **fora** do escopo.

2. EAP (WBS)

Desenhar no papel ou Miro a árvore completa respeitando a **Regra dos 100%**.

3. Board no GitLab

Lançar todos os pacotes aprovados como **Issues**, com estimativas e responsáveis.

4. Aprovação da Linha de Base

Apresentar o **Caminho Crítico** ao professor para validação e início da execução.

Reflexões do Capítulo

Desafio 1 (Escopo): O cliente pede reconhecimento de voz no seu chatbot, e a equipe aceita sem renegociar prazo. **Qual fenômeno ocorreu e como deveria ser tratado?**

Desafio 2 (PERT): O dev estima 10h (otimista), 40h (pessimista) e 17h (mais provável). **Qual valor vai para o cronograma e qual a margem de segurança?**

Desafio 3 (Caminho Crítico): Atividade de 40h de esforço no Caminho Crítico com estagiário de 20h/semana. **Qual a Duração e por que ela não pode atrasar?**

Obrigado!

Dúvidas?

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

Não esqueça de responder o questionário L04!

Próxima aula: Qualidade e Recursos Humanos