TÉCNICAS DE DECOMPOSIÇÃO

Desenvolvedores:

Victor ignacio – 823125249

Kaue Brito Vieira - 824211851

Giulia Gabriella - 823122979

Kauê Dib de Souza Dias - 823149871

Vinícius Santana Teixeira - 82319112

Murilo Bonuccelli de Oliveira – 823148988

São Paulo – SP

2025

1. Estimativa baseada em LOC (Lines of Code)

Como funciona:

- Decompõe o sistema em módulos/funções e estima quantas linhas de código cada parte terá.
- A produtividade da equipe (ex: LOC/dia) é usada para calcular o esforço necessário.

Aplicação:

Imagine um **software embarcado para um controlador de semáforo inteligente**.
A equipe divide o projeto em:

- Controle de sinais: 1.200 LOC
- Leitura de sensores: 800 LOC
- Comunicação com central: 1.000 LOC
- Interface de configuração: 500 LOC
  Total estimado: 3.500 LOC

Se a produtividade da equipe é de **50 LOC/dia por programador**, com 2 programadores:

3.500 LOC ÷ (2 × 50 LOC/dia) = **35 dias**

Melhor cenário para usar LOC:

Tipo de projeto:

- **Software técnico ou de baixo nível**, como:
  - Sistemas embarcados
  - Firmwares
  - Drivers de dispositivos
  - Compiladores
  - Sistemas operacionais
  - Aplicativos de linha de comando com foco técnico

Condições ideais:

1. **Requisitos bem definidos e estáveis** — pouco espaço para mudanças significativas no escopo.
2. **Experiência anterior com sistemas semelhantes** — há dados históricos sobre quantas LOC foram necessárias e a produtividade da equipe.
3. **Linguagem de programação já definida** — essencial, pois LOC variam muito entre linguagens (ex: 100 LOC em C pode equivaler a 20 LOC em Python).
4. **Equipe experiente** — os desenvolvedores conseguem estimar com boa precisão a quantidade de código por funcionalidade.

Exemplo prático ideal:

Um time está desenvolvendo o **firmware de um microcontrolador para controle de um motor elétrico industrial**, usando C.

- Os módulos são bem conhecidos: leitura de sensores, controle de motor, comunicação serial.
- A equipe já trabalhou em projetos semelhantes e sabe que, em média, cada módulo exige cerca de 1.000 LOC.
- A produtividade média da equipe é 100 LOC/dia por desenvolvedor.

Estimam 4 módulos principais → 4.000 LOC
Com 2 desenvolvedores → 4.000 ÷ (2 × 100) = **20 dias de trabalho**

Quando *não* usar LOC:

- Projetos com **alta interação com usuário final** (ex: apps com muitas telas ou regras de negócio) — melhor usar FP ou Casos de Uso.
- Projetos em **linguagens de alto nível/dinâmicas** (Python, JavaScript) — o LOC perde precisão.
- Ambientes ágeis ou de **desenvolvimento iterativo** — onde o escopo muda constantemente.
- Quando há **pouca ou nenhuma experiência prévia** com o tipo de sistema a ser desenvolvido.

Resumo – Use LOC quando:

Critério	LOC é adequado?
Sistema técnico/baixo nível.	Sim
Linguagem bem definida	Sim
Requisitos estáveis.	Sim

Equipe com histórico similar.	Sim
Projeto com alto nível visual.	Não recomendado
Escopo pode mudar bastante.	Não recomendado

2. Estimativa baseada em FP (Function Points)

Como funciona:

- Mede a **complexidade funcional** do sistema, considerando o que será entregue ao usuário.
- Baseia-se em 5 componentes:
    - Entradas externas (EE)
    - Saídas externas (SE)
    - Consultas externas (CE)
    - Arquivos internos (ILF)
    - Arquivos externos (EIF)
      Cada item recebe um peso (simples, médio, complexo), e somam-se os pontos de função.

Aplicação:

No sistema de **controle de ponto eletrônico**:

- 5 entradas (ex: registrar entrada, saída): 5 × 4 (simples) = 20 FP
- 3 saídas (ex: relatórios): 3 × 5 (médio) = 15 FP
- 2 consultas (ex: consultar horários): 2 × 4 = 8 FP
- 2 arquivos internos (funcionários, registros): 2 × 7 = 14 FP

- 1 arquivo externo (dados da folha de pagamento): 1 × 5 = 5 FP
  Total: 62 FP

Com produtividade de **5 FP/dia**, o esforço seria:

62 ÷ 5 = **12,4 dias**

Melhor cenário para usar FP:

Tipo de projeto:

- **Sistemas de informação** com muitas entradas, saídas, relatórios e regras de negócio, como:
  - Sistemas ERP, CRM, RH, financeiro, contábil
  - Portais corporativos e intranets
  - Sistemas de controle de estoque, faturamento, atendimento ao cliente
  - Aplicações com muitos formulários e relatórios

Condições ideais:

1. **Foco funcional** — quando o principal é "o que o sistema faz", não "como ele faz".
2. **Escopo funcional claro** — os requisitos são voltados às funcionalidades de negócio.
3. **Independência de linguagem** — ideal quando a tecnologia ainda não foi escolhida.
4. **Histórico de produtividade baseado em FP** — equipes ou organizações maduras têm produtividade estimada em FP/dia.

Exemplo prático ideal:

Uma empresa está desenvolvendo um **sistema de controle de contratos e faturamento de serviços**.

- Requisitos: cadastrar contratos, registrar consumo de serviços, emitir faturas, gerar relatórios, integrar com contabilidade.

- Analistas identificam:
    - 10 entradas externas (cadastros)
    - 5 saídas externas (faturas, relatórios)
    - 4 consultas
    - 3 arquivos internos
    - 2 arquivos externos

Aplicam os pesos dos FP e chegam a um total de **150 FP**.
Sabendo que a produtividade média da equipe é de **6 FP/dia**, estimam:

150 ÷ 6 = **25 dias de esforço**

Vantagens dos FP:

- **Independe da linguagem e da tecnologia**
- Ideal para contratos (por ser métrica padronizada pelo IFPUG)
- Mais estável frente a mudanças tecnológicas (ex: refatoração não afeta FP)
- Reflete valor funcional ao usuário

Quando *não* usar FP:

- Em **sistemas técnicos ou embarcados**, onde há pouco foco funcional e mais lógica técnica (ex: firmware, drivers)
- Em **protótipos rápidos ou projetos ágeis com pouca documentação inicial**
- Quando **não há analistas capacitados** para aplicar a contagem corretamente
- Se **a modelagem funcional ainda está muito imatura ou indefinida**

Resumo – Use FP quando:

Critério	FP é adequado?
Sistema de informação / negócios.	Sim
Requisitos funcionais bem descritos.	Sim
Linguagem de implementação indefinida.	Sim
Projeto técnico (embarcado, driver, etc.).	Não recomendado
Equipe sem conhecimento de FP.	Não recomendado

3. Estimativa baseada em processo

Como funciona:

- Usa o **modelo de processo de desenvolvimento** como base (ex: RUP, Scrum, CMMI).
- O projeto é dividido em fases ou sprints e estimativas são feitas para cada etapa com base em dados anteriores.

Aplicação:

Desenvolvimento de um **jogo educativo** em Scrum:

- A equipe tem uma **velocidade de 30 pontos de história por sprint**.
- O backlog inicial tem 120 pontos.

120 ÷ 30 = **4 sprints**

Se cada sprint tem 2 semanas:

4 × 2 = **8 semanas**

Melhor cenário para usar decomposição em processos:

Tipo de projeto:

- Projetos **médios a grandes** com um ciclo de vida estruturado.
- Projetos onde é possível **planejar por fases ou sprints**, como:
    - Sistemas corporativos internos com etapas formais (requisitos, design, codificação, testes)
    - Produtos de software com roadmap a médio/longo prazo
    - Projetos com **governança forte** (ex: bancos, governo, telecom)

Condições ideais:

1. **Processo de desenvolvimento bem definido** (ágil ou tradicional)

2. **Histórico de produtividade por fase** (ex: sabemos que testes levam 25% do tempo total)
3. **Organização madura em engenharia de software**, com métricas de produtividade por processo
4. **Equipe com papéis bem definidos** (analistas, devs, QA, PO)

Exemplo prático ideal:

Um time está desenvolvendo um **sistema bancário para análise de crédito**, seguindo o modelo RUP (Rational Unified Process), com as fases:

- Iniciação
- Elaboração
- Construção
- Transição

Com base em projetos anteriores, sabem que o esforço normalmente se distribui assim:

- Iniciação: 10%
- Elaboração: 20%
- Construção: 50%
- Transição: 20%

A estimativa inicial de esforço global é de **1.000 horas**.
Aplicando a decomposição por processo:

- Iniciação: 100 h
- Elaboração: 200 h
- Construção: 500 h
- Transição: 200 h

Cada fase é detalhada em tarefas específicas com seus responsáveis.

Vantagens:

- Ajuda no **planejamento de alocação de recursos por fase**
- Ideal para projetos com **ciclos de vida formais ou ágeis maduros**
- Facilita **gestão de risco**, pois cada fase pode ser monitorada
- Permite **aprimorar a previsão com dados históricos** por fase

Quando *não* usar essa técnica:

- Projetos pequenos, com escopo muito reduzido
- Quando **o processo ainda não está definido**
- Quando não há histórico confiável de produtividade por fase
- Em contextos muito dinâmicos (prototipação rápida, MVPs, startups)

Resumo – Use decomposição em processos quando:

Critério	Decomposição em processo é adequada?
Processo definido e seguido pela equipe.	Sim
Projeto médio ou grande com várias etapas.	Sim
Equipe com histórico por fase do processo.	Sim
Projeto pequeno, rápido ou informal.	Não recomendado
Ambiente com mudanças constantes no escopo.	Não recomendado

4. Estimativa baseada em caso de uso

Como funciona:

- Usa os **casos de uso do sistema** para estimar o esforço.
- Cada caso de uso é classificado (simples, médio, complexo), recebe um peso e é somado para calcular os **Pontos de Caso de Uso (UCP)**.

Aplicação:

Sistema de **reserva de salas**:

- 3 casos simples (reservar, cancelar, ver disponibilidade) × 5 = 15
- 2 médios (notificar por e-mail, adicionar recursos) × 10 = 20
- Total = 35 pontos brutos

Fator técnico e ambiental ajusta esse número (ex: × 1,1):

35 × 1,1 = **38,5 UCP**

Com produtividade de 20 UCP por mês:

38,5 ÷ 20 = **~2 meses**

Melhor cenário para usar decomposição em caso de uso:

Tipo de projeto:

- Projetos com **forte foco em funcionalidades de negócio**, onde:
    - Há diferentes perfis de usuários
    - As interações com o sistema são bem definidas

- Existe ou será construída uma modelagem de casos de uso

Exemplos:

- Sistemas de atendimento ao cliente (call center, helpdesk)
- Plataformas de e-commerce
- Sistemas de gestão acadêmica ou educacional
- Aplicativos web/mobile com muitas operações do usuário
- Portais de autoatendimento

Condições ideais:

1. **Modelagem de casos de uso clara e completa** (compreensão do que cada ator faz no sistema)
2. **Foco funcional claro**, com os requisitos organizados por "ações do usuário"
3. **Ambiente com analistas de sistemas capacitados** a definir e estimar casos de uso
4. Possibilidade de **classificar os casos** como simples, médios ou complexos, com base em:
    a. Número de passos do fluxo principal
    b. Variações/ramificações
    c. Interações com bancos de dados ou sistemas externos

Exemplo prático ideal:

Um time está desenvolvendo um **sistema de reserva de salas de reunião corporativo**.

Casos de uso identificados:

1. Reservar sala → **complexo**
2. Cancelar reserva → **simples**
3. Ver disponibilidade → **simples**

4. Notificar participantes → **médio**
5. Solicitar equipamentos adicionais → **médio**

Pesos típicos:

- Simples = 5 pontos
- Médio = 10 pontos
- Complexo = 15 pontos

Estimativa bruta de pontos:
(2 × 5) + (2 × 10) + (1 × 15) = **45 Pontos de Caso de Uso (UCP)**

Depois se aplica um **fator técnico e ambiental** (ex: × 1,2), resultando em:

45 × 1,2 = **54 UCP**

Com produtividade de 20 UCP/mês por desenvolvedor:

54 ÷ 20 = **~2,7 meses de esforço**

Vantagens:

- Traduz diretamente o que o **usuário espera do sistema**
- Funciona bem com UML e documentação funcional moderna
- Relaciona-se com requisitos funcionais de forma clara
- Ideal para **sistemas interativos e centrados no usuário**

Quando *não* usar essa técnica:

- Sistemas **sem interação direta com usuários** (ex: serviços em segundo plano, scripts, APIs técnicas)
- Projetos com **foco técnico** ou infraestrutura (ex: backend de servidor, automação de processos)
- Quando **não há modelagem formal de casos de uso**

- Equipe sem experiência com UML ou análise orientada a casos de uso

Resumo – Use Casos de Uso quando:

Critério	Casos de Uso é adequado?
Sistema interativo e centrado no usuário.	Sim
Casos de uso identificados e modelados.	Sim
Projeto com funcionalidades de negócio claras.	Sim
Sistema técnico ou sem interação com usuários.	Não recomendado
Falta de modelagem funcional.	Não recomendado

5. Harmonizando estimativas

Como funciona:

- **Combina múltiplas técnicas** de estimativa (LOC, FP, casos de uso, especialistas, históricos) para melhorar a precisão.
- Pode usar métodos como:
    - **Delphi** (consenso entre especialistas)
    - **Planning Poker** (em ágil)
    - **Média ponderada** entre diferentes métodos

Aplicação:

No desenvolvimento de uma **plataforma EAD**, a equipe:

- Estima 20.000 LOC → 80 dias
- Calcula 250 FP → 50 dias
- Usa casos de uso → 45 dias
- Especialistas estimam → 60 dias

Harmonização via média ponderada:

(80 + 50 + 45 + 60) ÷ 4 = **58,75 dias**

A estimativa final será algo em torno de 59 dias, com margem de ajuste conforme risco e escopo.

Melhor cenário para usar harmonização de estimativas:

Tipo de projeto:

- Projetos **complexos, estratégicos ou com alto risco**, onde:
    - Há **incerteza sobre os requisitos**
    - O impacto de um erro na estimativa é alto
    - Existem múltiplas partes interessadas (TI, negócios, fornecedores)
    - O projeto será usado para **contratação de terceiros ou orçamento executivo**

Exemplos:

- Sistemas governamentais de larga escala (ex: Receita Federal, Justiça Eleitoral)
- Plataformas bancárias, de seguros ou previdência
- Projetos de transformação digital em grandes empresas
- Projetos com várias equipes ou fornecedores envolvidos

Condições ideais:

1. **Você já aplicou múltiplas técnicas de estimativa** com resultados diferentes (por ex., FP = 5 meses, LOC = 6 meses, especialistas dizem 5,5 meses).
2. **Existem dados históricos para validar e ajustar os resultados**.
3. A estimativa precisa ser **justificada ou auditável** (ex: contratos, licitações).
4. A organização está disposta a **investir tempo na estimativa**, para mitigar riscos maiores.

Exemplo prático ideal:

Uma estatal está planejando um sistema nacional de **cadastro de benefícios sociais**.
O time realiza:

- Estimativa por FP → 1.200 FP → 8 meses
- Estimativa por UCP → 150 UCP → 7 meses
- LOC (em Java) → 100.000 LOC → 9 meses
- Estimativa de especialistas → entre 7 e 9 meses

A equipe então harmoniza os resultados:

- Descartam o mais extremo (LOC)
- Fazem uma **média ponderada** entre FP, UCP e especialistas
- Resultado final: **7,5 meses**

Isso é usado para planejar o projeto e **negociar prazos com stakeholders**.

Vantagens:

- Reduz **viés individual de uma técnica só**
- Melhora a **precisão e a confiança na estimativa**
- Permite mostrar **robustez técnica** diante de auditorias

- Ajuda a **alinhar expectativas entre áreas técnicas e de negócio**

Quando *não* usar harmonização:

- Projetos pequenos ou de curta duração
- Situações em que há **pouco tempo ou recursos para estimar**
- Quando **não há múltiplas técnicas viáveis** a serem comparadas
- Equipe inexperiente em mais de uma técnica de estimativa

Resumo – Use harmonização quando:

Critério	Harmonização é adequada?
Projeto grande ou estratégico.	Sim
Múltiplas técnicas de estimativa aplicáveis.	Sim
Requisitos ainda têm incertezas.	Sim
Projeto pequeno e simples.	Não recomendado
Sem tempo para aplicar várias técnicas.	Não recomendado