14 - Sucessive Refinement

2025-10-04 8:55 am

- Estudo de Caso de um parser de argumentos de CLI
  - Mostrar etapas de desenvolvimento-
- Objetivo:
  - receber argumentos pelo CLI para o programa de uma certa forma customizada
  - `<Exemplo de bom código>`
- Código limpo primeiro é sujo e depois é limpado.
  - Primeira versão nunca é a ideal, requer trabalho repetidamente.
  - Só fazer um código que funciona nunca é o trabalho inteiro.
  - Não raro o código começa limpo e vai se bagunçando com o tempo e crescimento.
  - Existe um ponto em que é importante parar de incrementar e começar a refatorar.
  - Sempre ser incremental:
    - TDD ajuda nisso, já que sistema deve estar funcionando com testes.
    - Só progredir quando os testes passam
  - É comum na refatoração inserir código que será removido logo em seguida
    - Vários pequenos passos incrementais que vão e voltam, como resolver um cubo mágico.
  - Boa parte do bom código é compartimentalização:
    - Separar códigos diferentes em lugares diferentes
    - Separação de responsabilidades
- Muito difícil limpar código sujo há muito tempo.
  - Mantê-lo limpo desde o início vale muito mais a pena