

Departamento de Engenharia de Sistemas e Computação

Especificação do Projeto

Do problema à estrutura do programa

João Araujo

Algoritmos Computacionais I

Aula 30

Objetivo da aula

Cada grupo entrega uma **especificação** do projeto: descrição detalhada do problema, das funcionalidades principais e da estrutura prevista do programa.

- Feedback **individualizado** para cada grupo
- Avaliação da **viabilidade**: o escopo cabe nas aulas restantes?
- Ajustes para que o projeto fique adequado ao nível da turma

Lembrete do Cap. 9.1

"Pense antes de programar": uma boa especificação economiza horas de retrabalho.

Roteiro do documento de especificação (1 a 2 páginas)

1. **Título e integrantes**
2. **Problema**: o que será resolvido, para quem, por quê
3. **Dados**: entradas (usuário, arquivos), saídas (tela, arquivos, gráficos)
4. **Funcionalidades**: lista numerada, em ordem de prioridade
5. **Estrutura do programa**: módulos e funções previstos
6. **Estruturas de dados**: listas, dicionários, tuplas, arrays, DataFrames
7. **Casos de teste e casos-limite**: o que pode dar errado?
8. **Divisão de tarefas e cronograma**

Exemplo: Gestão de insumos e custos (Eng. Civil/Produção)

Problema. Pequena obra precisa controlar o estoque e o custo de insumos.

Entradas. `insumos.csv` (nome, unidade, preço, estoque); comandos do usuário.

Saídas. Relatório de custos, alerta de estoque baixo, gráfico dos 5 insumos mais caros.

Funcionalidades

1. Cadastrar/editar insumo
2. Registrar entrada e saída de estoque
3. Calcular custo total e por categoria
4. Alertar quando estoque $\leq$ mínimo
5. Salvar e carregar do arquivo CSV
6. Gerar gráfico (Matplotlib)

Exemplo: estrutura prevista

```
projeto_insumos/
    principal.py        # menu e fluxo geral (main)
    estoque.py          # regras: cadastra(), baixa(), custo_total(), alertas()
    arquivos.py         # le_csv(), salva_csv()
    graficos.py         # grafico_mais_caros(), grafico_por_categoria()
    insumos.csv         # dados de exemplo
    testes.py           # asserts das funções de estoque
```

Estrutura de dados:

```
estoque = {
    "cimento": {"unidade": "saco", "preco": 38.5, "qtd": 120, "minimo": 30},
    "areia":   {"unidade": "m3",   "preco": 95.0, "qtd": 15,  "minimo": 10},
}
```

Dicionário de dicionários (Aula 20): chave = nome do insumo; busca direta e legível.

Exemplo para Eng. Elétrica/Mecânica

Análise de ensaio de um resistor

Entradas: arquivo CSV com tensão e corrente.
Funcionalidades: (1) ler e validar os dados; (2) ajustar a reta $I = aV + b$ com NumPy; (3) estimar a resistência $R = 1/a$ e a incerteza; (4) detectar medidas discrepantes; (5) gráfico dos pontos com a reta; (6) relatório em arquivo texto.
Casos-limite: menos de 2 pontos, tensão repetida, arquivo vazio, valores não numéricos.

Séries históricas (Eng. Ambiental/Cartográfica)

Entradas: CSV de temperatura diária.
Funcionalidades: médias mensais e anuais com Pandas; anos mais quentes; gráfico de tendência; conversor de coordenadas.

Como especificar boas funcionalidades

Vago (evite)

- "O programa gerencia tudo"
- "Faz uns gráficos"
- "Trata os erros"

Verificável (prefira)

- "Mostra o custo total e o de cada categoria, com 2 casas decimais"
- "Gera um gráfico de barras com os 5 itens mais caros e salva em PNG"
- "Se o arquivo não existe, avisa e cria um novo"

Teste: para cada funcionalidade, escreva um caso de uso com entrada e resultado esperado.

Planejando o tamanho do projeto

- Classifique as funcionalidades: **essenciais** (sem elas não há programa), **desejáveis** e **extras**
- Restam **5 aulas** (31 a 35): implemente primeiro o essencial
- Estime: "quantas funções? quantas linhas?" (um projeto típico: 150 a 400 linhas)
- Evite *interfaces gráficas* e *bancos de dados*: fora do escopo da disciplina

O ótimo é inimigo do bom

Prefira um programa pequeno, correto e bem organizado a um grande que não funciona.

Atividade desta aula

1. Cada grupo escreve (ou finaliza) o documento de especificação seguindo o roteiro
2. O professor atende os grupos um a um e verifica:
   - O problema está claro? Dá para testar?
   - O escopo é viável? O que é essencial?
   - A divisão em funções e dados faz sentido?
3. Ajustem a especificação com base no feedback
4. **Entrega ao fim da aula** (ou até a próxima, conforme combinado)

Checklist da especificação

- ☐ O problema cabe em duas frases?
- ☐ Entradas e saídas estão definidas, com formato dos arquivos?
- ☐ Há pelo menos 5 funcionalidades verificáveis, priorizadas?
- ☐ Há uma lista de funções/módulos e a estrutura de dados escolhida?
- ☐ Os casos-limite e as entradas inválidas estão listados?
- ☐ Cada integrante tem tarefas definidas?
- ☐ Os dados de exemplo já estão disponíveis?