

Design

1. Introdução ao Design OO

Etapa de transformação dos requisitos do sistema numa **solução lógica** baseada no paradigma OO.

Objetivos do Design OO:

- ✓ Criar um **modelo lógico** da solução.
- ✓ Definir **responsabilidades** dos objetos.
- ✓ Modelar as **interações** entre objetos.
- ✓ Garantir **modularidade, reutilização e manutenção**.

Artefatos do Design

Produzidos **depois da análise** e servem de base para a **implementação do software**.

- 📌 **Diagrama de Classes** – Representa a estrutura estática das classes.
- 📌 **Diagramas de Sequência** – Representa a troca de mensagens entre objetos ao longo do tempo.

2. Transição dos Requisitos para o Design

Passos no Design OO:


- 1 **Promover conceitos do modelo de domínio a classes de software.**
- 2 **Definir métodos e responsabilidades das classes.**
- 3 **Criar mensagens entre classes/objetos.**

3. Diagramas no Design OO

- **Visão Estática:**

- 📌 **Diagrama de Classes** – Mostra a estrutura do sistema.

- **Visão Dinâmica:**

-  **Diagrama de Sequência** – Mostra a interação entre objetos numa ordem cronológica.

4. Responsibility-Driven Design (RDD)

Princípio fundamental do Design OO, que se foca:

- ✓ Na atribuição de **responsabilidades aos objetos**.
- ✓ Na **colaboração entre objetos**.
- ✓ Na aplicação de **padrões de design** para distribuir responsabilidades.




Tipos de Responsabilidades

- 1 **"Fazer"** – Criar objetos, executar cálculos, coordenar ações.
- 2 **"Saber"** – Manter dados próprios, conhecer outros objetos, calcular informações.

5. GRASP – Princípios de Atribuição de Responsabilidades

O **GRASP (General Responsibility Assignment Software Patterns)** define padrões para atribuir responsabilidades às classes no Design OO.

6. Separação entre UI e Domínio

-  **UI** – Contém classes que gerem a **interface gráfica e interação com o utilizador**.
-  **Domain** – Contém classes que **implementam a lógica de negócio**.
-  **Controllers** – Atuam como **intermediários entre UI e Domínio**.
- ◆ **Boa prática: Evitar comunicação direta** entre a UI e o domínio. A UI deve interagir **apenas com controllers**.

Conclusão

O **Design Orientado a Objetos** define **como os objetos colaboram** para implementar os requisitos do sistema.

✅ **RDD e GRASP** são fundamentais para uma boa atribuição de responsabilidades.

✅ **Diagrama de Classes e Diagramas de Sequência** são essenciais para um design bem estruturado.

✅ **Separação entre UI, Controladores e Domínio** reduz acoplamento e melhora a manutenção.