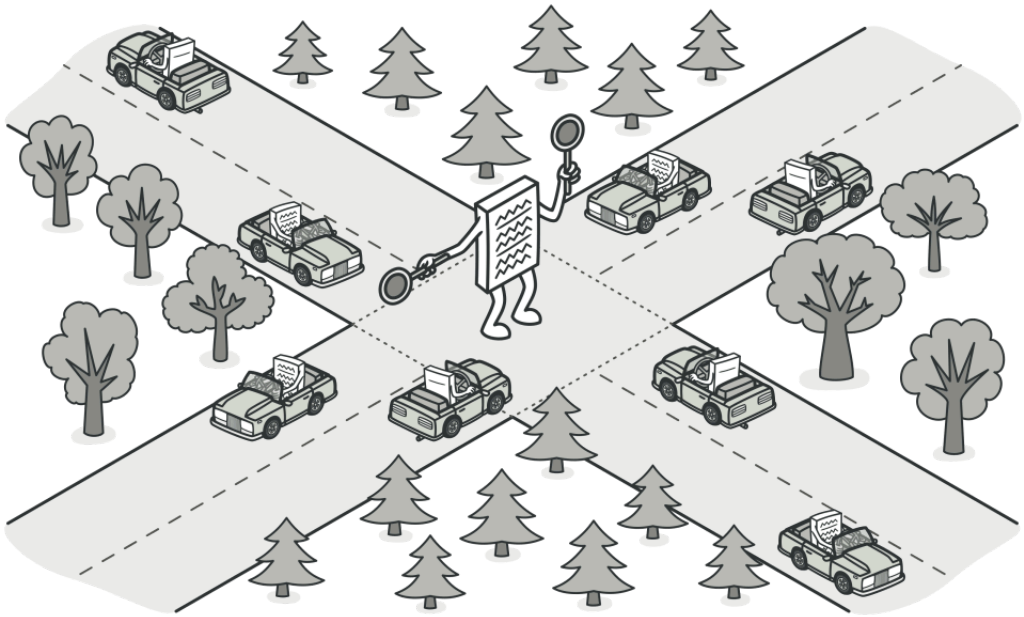
TÓPICO 18 - MEDIATOR

Design Patterns - Professor Ramon Venson - SATC 2026.1

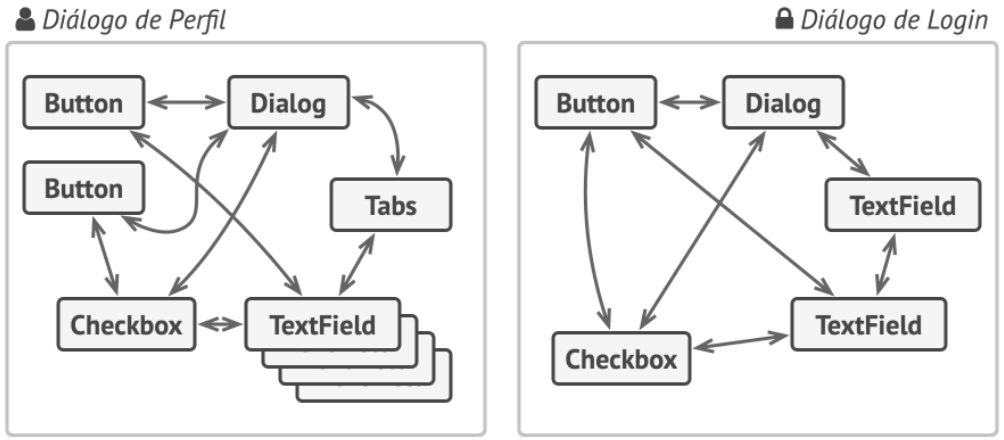
Motivação

Sistemas reais possuem componentes que colaboram o tempo todo.

O problema começa quando cada objeto passa a conhecer detalhes de vários outros ao mesmo tempo.



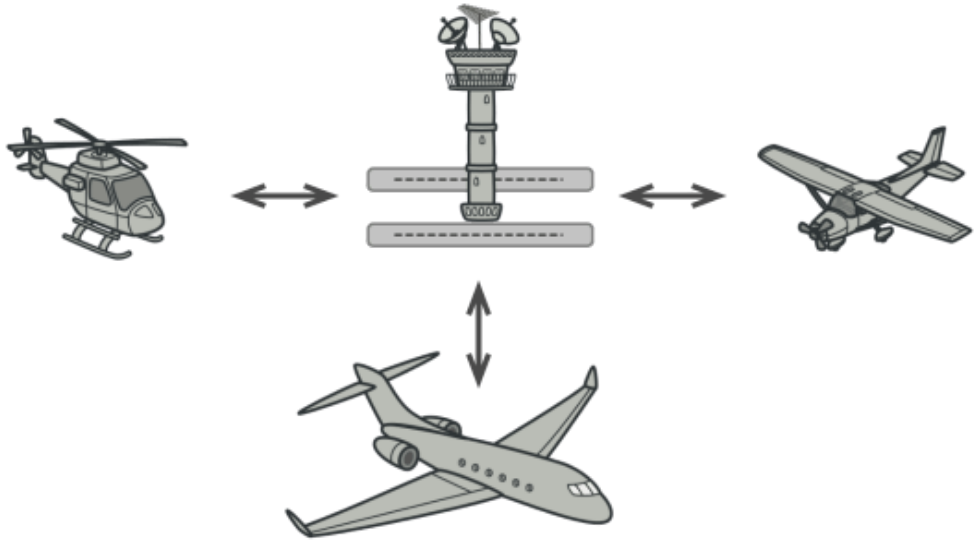
Problema

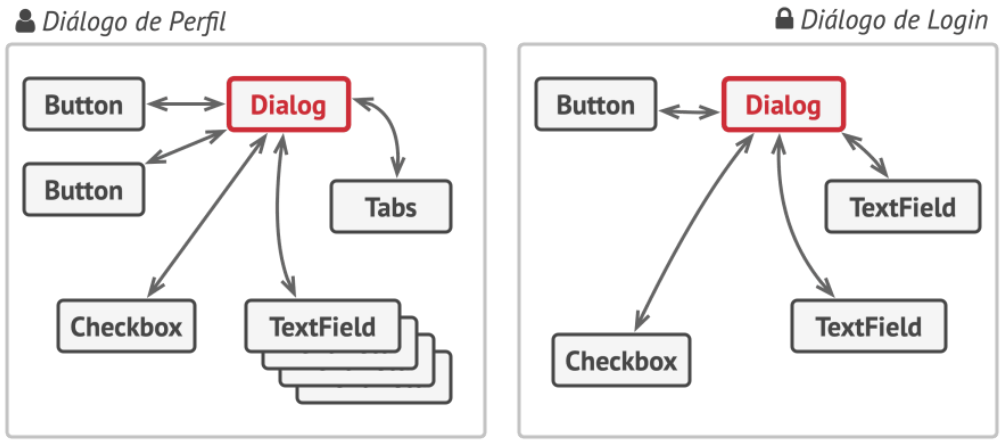


- Campos, botões e mensagens se influenciam
- Cada componente conhece vários colegas
- O acoplamento cresce rapidamente
- Reutilização fica mais difícil
- Mudanças pequenas geram efeito cascata

Conceito principal

- **Mediator** é um padrão comportamental
- Ele centraliza a comunicação entre objetos
- Componentes notificam eventos ao mediador
- O mediador decide as reações necessárias
- Componentes deixam de conversar diretamente





Como funciona

- Cada componente conhece apenas o mediador
- Eventos como clique e seleção são notificados
- O mediador interpreta o contexto
- Ele coordena quem reage e como reage
- A colaboração fica organizada em um ponto central

Estrutura / Componentes

- **Mediator**: define o contrato de coordenação
- **ConcreteMediator**: implementa a lógica de interação
- **Component**: mantém referência ao mediador
- **ConcreteComponent**: executa sua responsabilidade específica
- **Cliente**: monta e conecta os participantes

Código: contrato e componentes

```
interface Mediator {
    void notificar(ComponenteUI remetente, String evento);
}

abstract class ComponenteUI {
    protected final Mediator mediator;

    protected ComponenteUI(Mediator mediator) {
        this.mediator = mediator;
    }
}
```

Código: mediador concreto

```
class DialogoAutenticacao implements Mediator {
    public void notificar(ComponenteUI remetente, String evento) {
        if (evento.equals("alternouModo")) {
            System.out.println("Atualiza a tela.");
        }

        if (evento.equals("clicou")) {
            System.out.println("Processa a acao principal.");
        }
    }
}
```

Quando usar

- Há muitas dependências entre objetos
- A coordenação ficou confusa
- Componentes precisam ser reutilizados
- Mudanças causam efeito cascata
- Existe um fluxo central de colaboração

Quando não usar

- Poucos objetos colaboram diretamente
- A lógica de interação é simples demais
- O mediador só adicionaria burocracia
- A equipe criaria abstração sem ganho real
- Um objeto central ficaria artificial

Vantagens

- Reduz acoplamento entre componentes
- Centraliza a lógica de colaboração
- Facilita manutenção do fluxo
- Melhora reutilização dos participantes
- Ajuda a evoluir telas e workflows

Desvantagens

- Pode virar um objeto grande demais
- Introduz indireção adicional
- Exige disciplina de responsabilidades
- Pode esconder excesso de regras no centro
- É desnecessário em cenários pequenos

Relações com outros padrões

- **Observer** distribui eventos; **Mediator** coordena
- **Facade** simplifica acesso; **Mediator** organiza interação
- **Command** pode ser disparado pelo mediador
- **SRP** ajuda a tirar coordenação dos componentes
- O foco principal aqui é **baixo acoplamento**

Resumo final

- Mediator centraliza comunicação entre objetos
- Componentes notificam em vez de acoplar diretamente
- O sistema fica mais organizado e reutilizável
- O padrão é útil quando a coordenação virou problema
- Use com cuidado para evitar um Objeto Deus