Observer & Mediator Pattern

Seminar: Entwurfsmuster

Qais Latif

Hochschule Niederrhein

18 - 19 September 2025

Themenübersicht

1. Einleitung & Problemstellung
   – Grundlagen der Verhaltensmuster und das Problem der Kopplung
2. Die Entwurfsmuster: Observer & Mediator
   – Intention & Motivation
   – Struktur & Teilnehmer
   – Kollaboration & Konsequenzen
3. Direktvergleich & Anwendungsfälle
4. Kernaussagen & Fazit
5. Fragen

Grundlagen & Problemstellung

Was sind Verhaltensmuster?

- Fokus auf dynamische **Kommunikationsmuster.**
- Befassen sich mit Algorithmen und Verantwortlichkeiten.

Das Problem: Enge Kopplung

- Objekte werden voneinander **abhängig** & schwer **wartbar.**
- Änderungen breiten sich unkontrolliert aus.

Zwei Lösungsstrategien

- **Observer** und **Mediator** reduzieren diese Kopplung.
- Beide entkoppeln Sender von Empfängern.

Das Observer-Muster: Eine Einführung

Intention & Motivation

1-zu-N-Abhängigkeit (eins-zu-viele)

- Eine Quelle **(Subjekt)** hat viele Empfänger **(Observer)**

z. B. Ein Datensatz → viele Ansichten (Tabelle, Diagramme)

Automatische Benachrichtigung

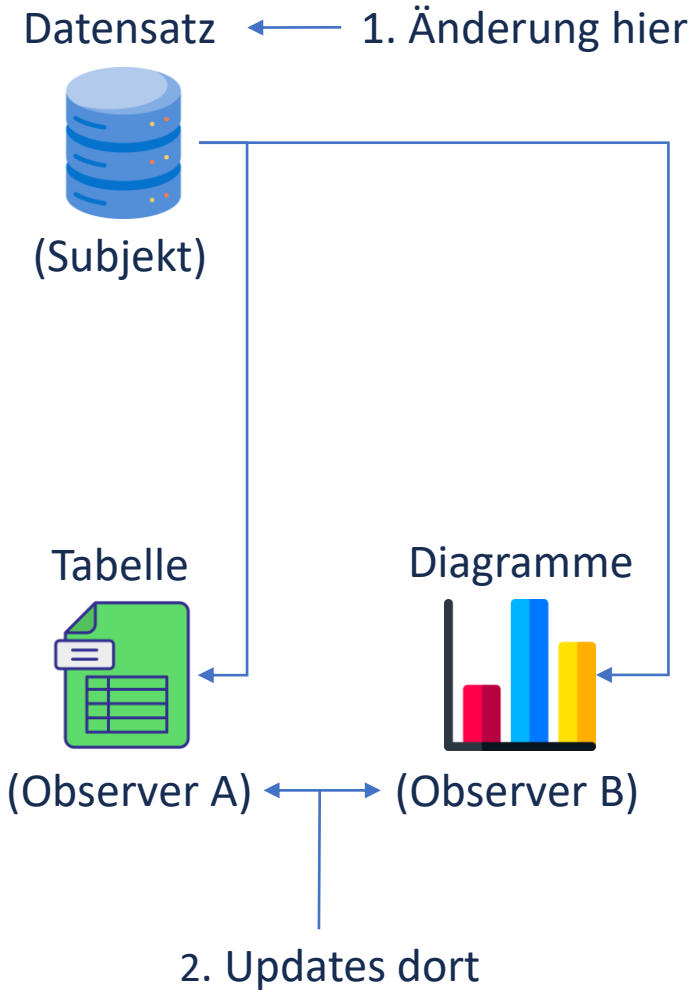
- Wenn sich die Quelle **ändert**, erfahren es alle
- Subjekt ändert Zustand → Notify() an alle Observer

z. B. Tabellenwert ändern → Diagramm aktualisiert sofort

Lose Kopplung

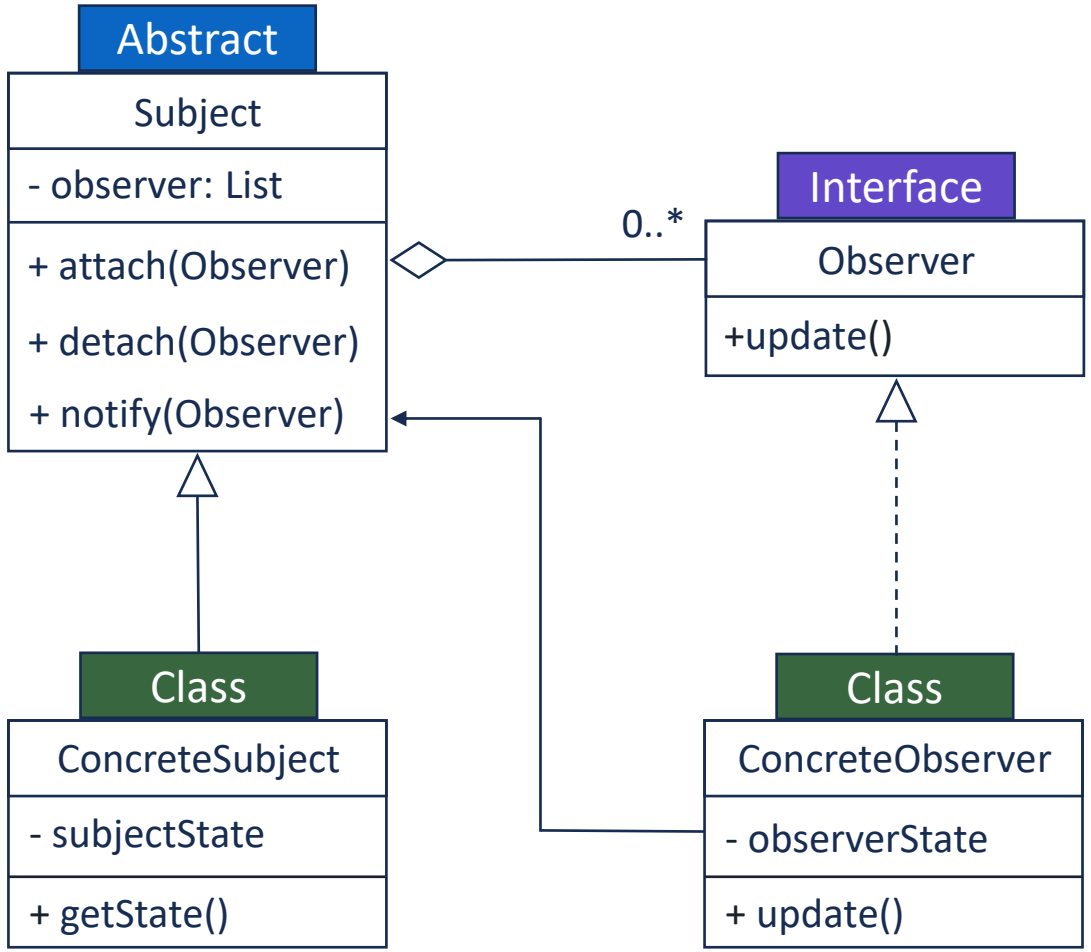
- Objekte bleiben **synchron**, aber **unabhängig** voneinander
- Ziel: Konsistenz wahren ohne direkte Objektreferenzen

z. B. Das Diagramm muss die Existenz der Tabelle nicht kennen

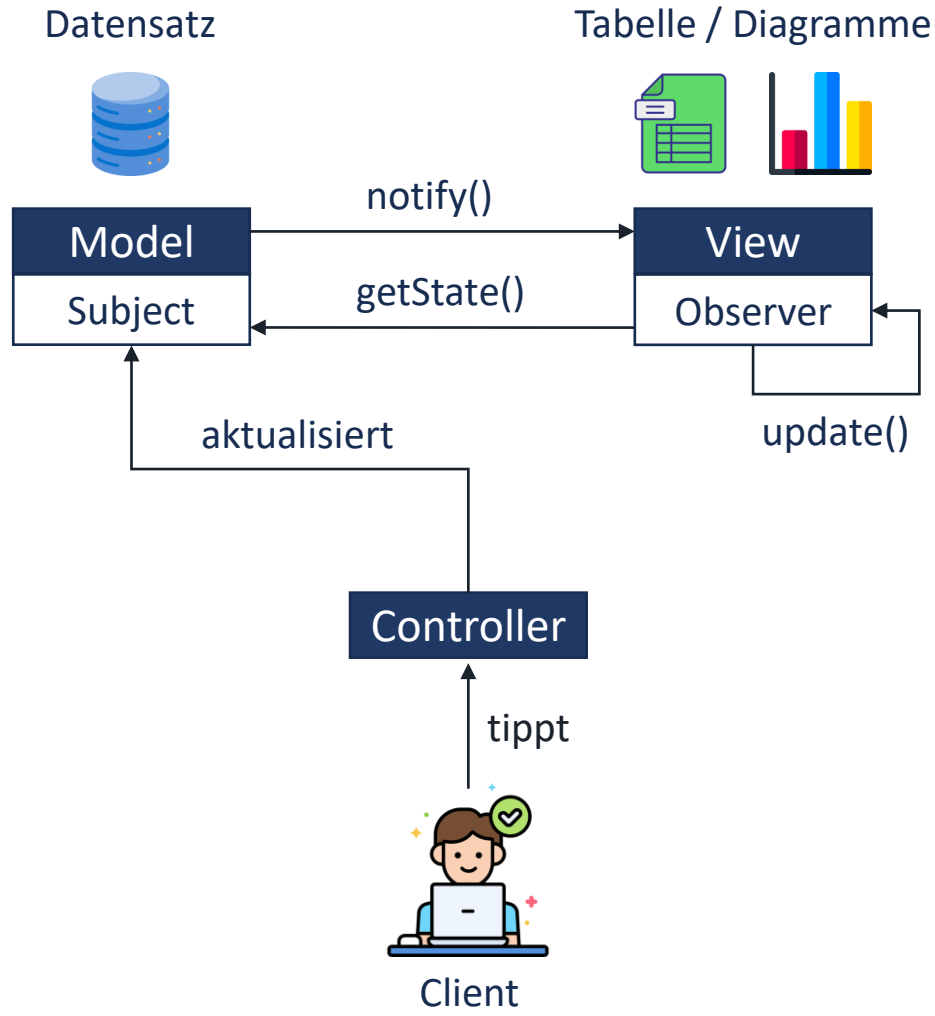


Observer: Struktur & Teilnehmer

UML-Diagramm: Die vier Teilnehmer

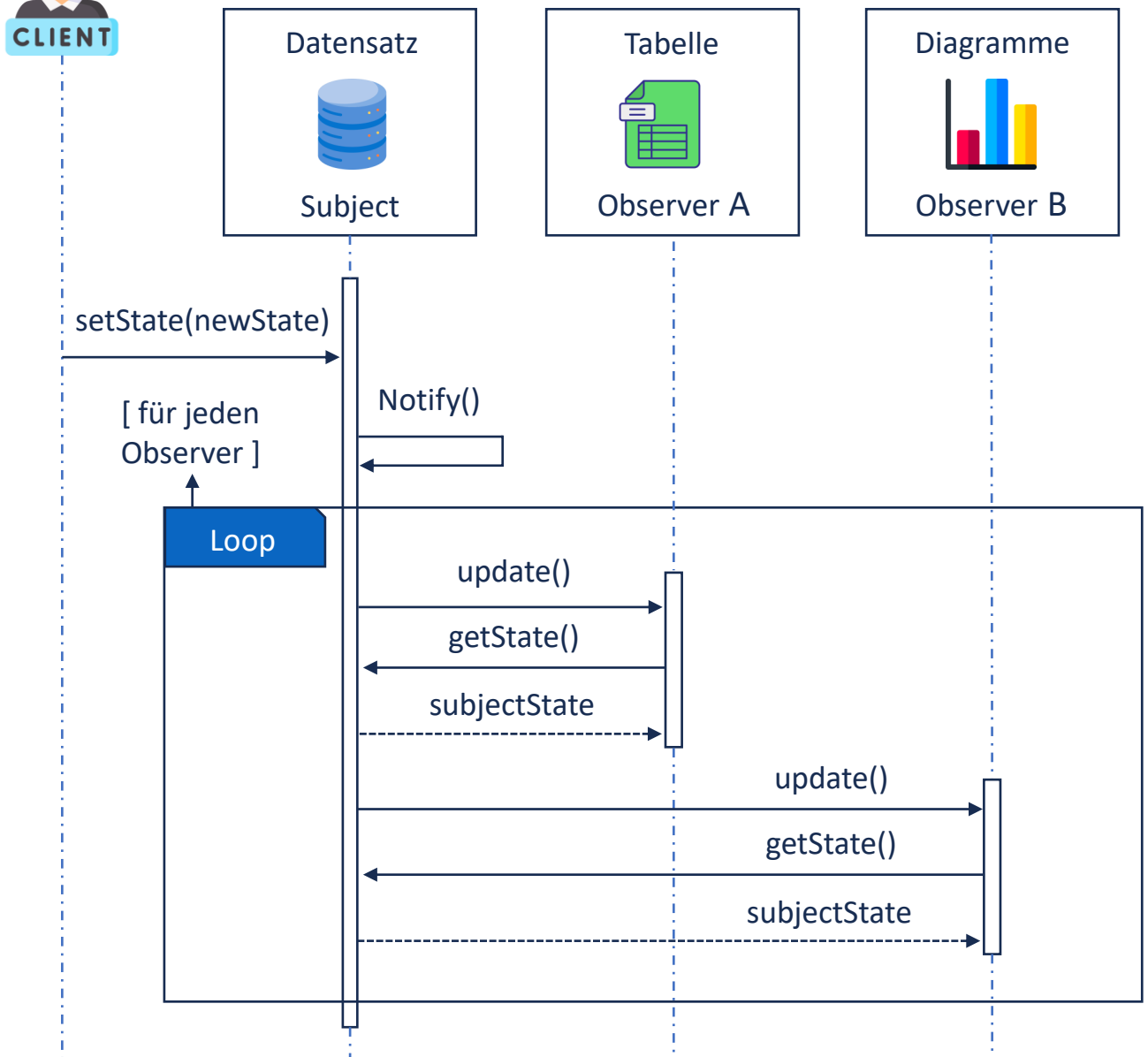


Zuordnung zum MVC-Pattern



Observer: Kollaboration & Implementierung

Ablauf einer Benachrichtigung (Pull Model)



Implementierung: Push- vs. Pull-Modell

Push-Modell: Subjekt sendet detaillierte Infos.

Pro: Effizient (kein Nachfragen nötig)

Con: Höhere Kopplung (Subjekt macht Annahmen).

Pull-Modell: Subjekt sendet minimale infos.

- Observer holt benötigte Daten aktiv ab.

Pro: Maximale **Entkopplung**.

Con: Potenziell **ineffizient** (viele Abfragen).

Observer: Vor- & Nachteile

Vorteil

Abstrakte Kopplung

- Das Subjekt kennt seine Observer nur über deren abstraktes Interface.

Broadcast-Kommunikation

- Ein Subjekt kann eine beliebige Anzahl von Observern mit einem einzigen Notify()-Aufruf benachrichtigen.

Dynamische Beziehungen

- Observer können zur Laufzeit flexibel hinzugefügt und entfernt werden.

Nachteil

Unerwartete Updates

- Eine kleine Operation kann eine Kaskade von Updates auslösen, was zu schwer zu debuggenden Performance-Problemen führen kann.

"Lapsed Listener"-Problem

- Wenn ein Observer zerstört wird, ohne sich vom Subjekt abzumelden, kann dies zu Speicherlecks oder Abstürzen führen.

Das Mediator-Muster: Eine Einführung

Intention & Motivation

Kapselung

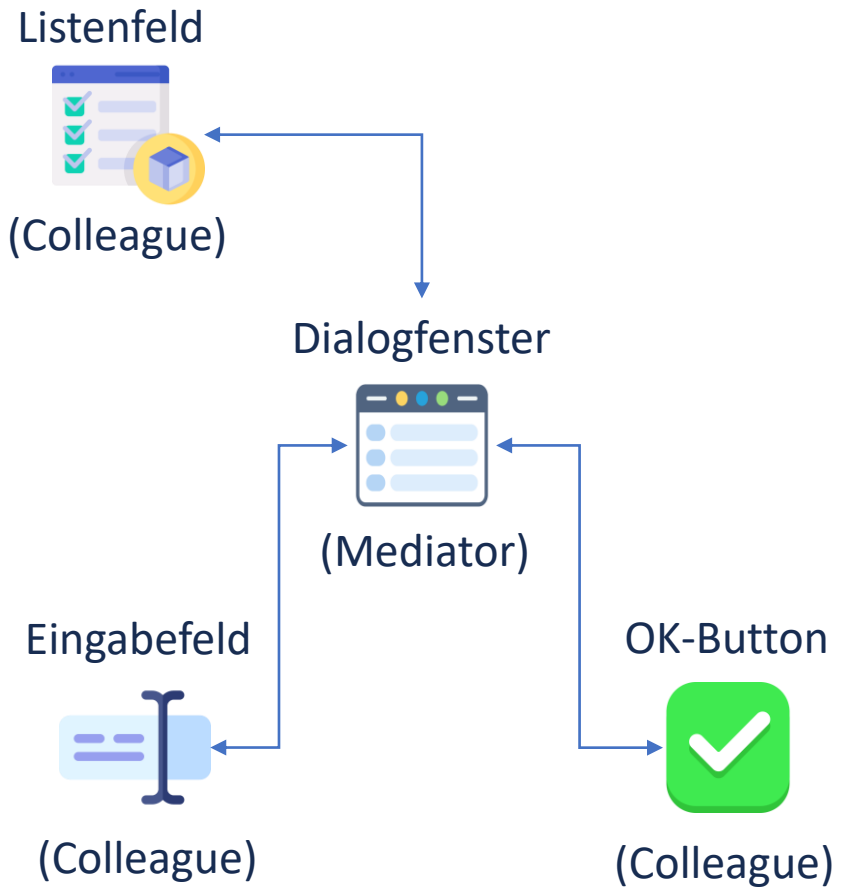
- Ein zentrales Objekt (der **Mediator**) kapselt die gesamte Interaktion einer Gruppe.

Entkopplung

- Objekte kennen sich nicht mehr direkt, sondern kommunizieren nur noch über den Mediator.

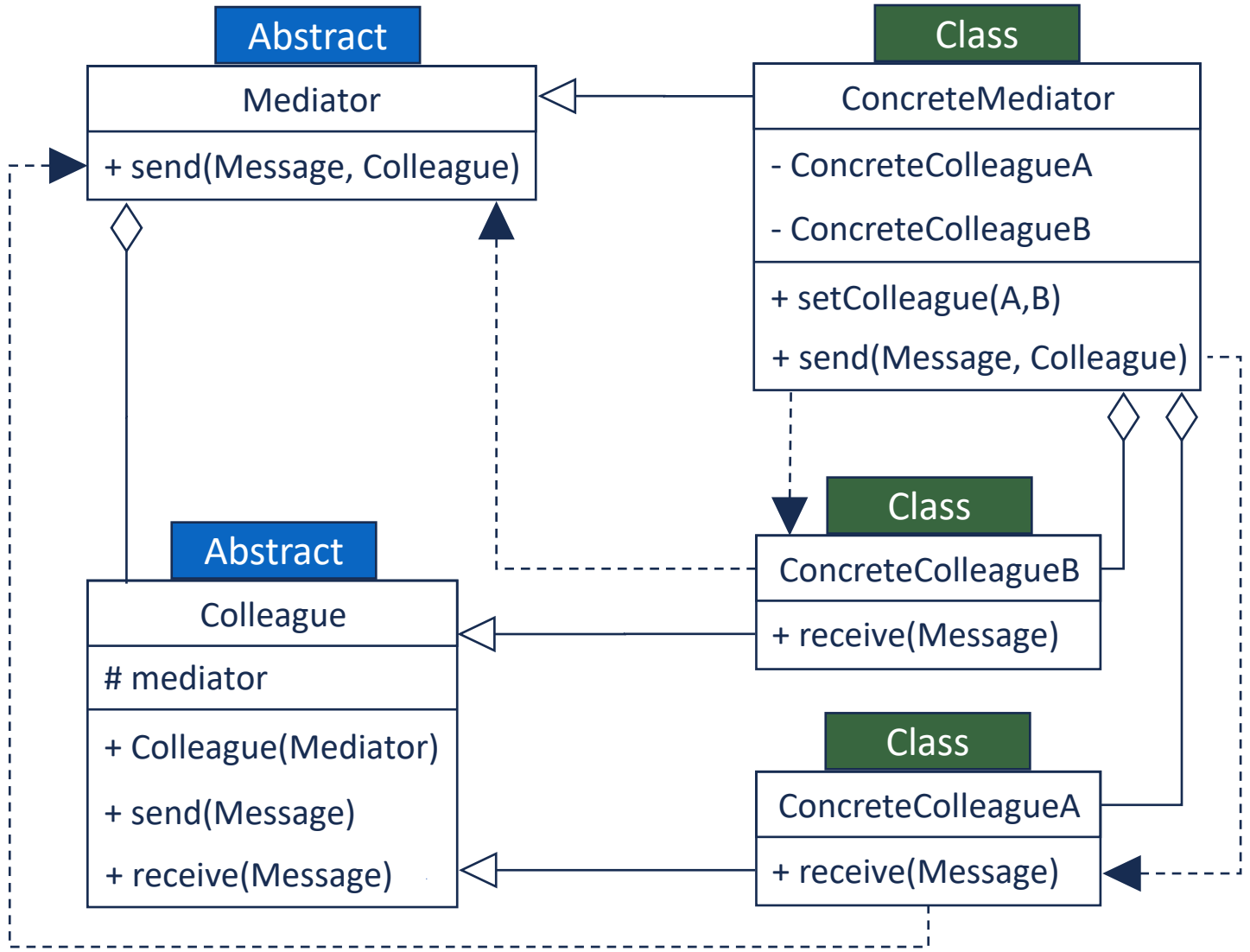
Zentralisierung

- Löst das Problem chaotischer "**Viele-zu-Viele**"-Kommunikation.



Mediator: Struktur & Teilnehmer

UML-Diagramm: Die Teilnehmer

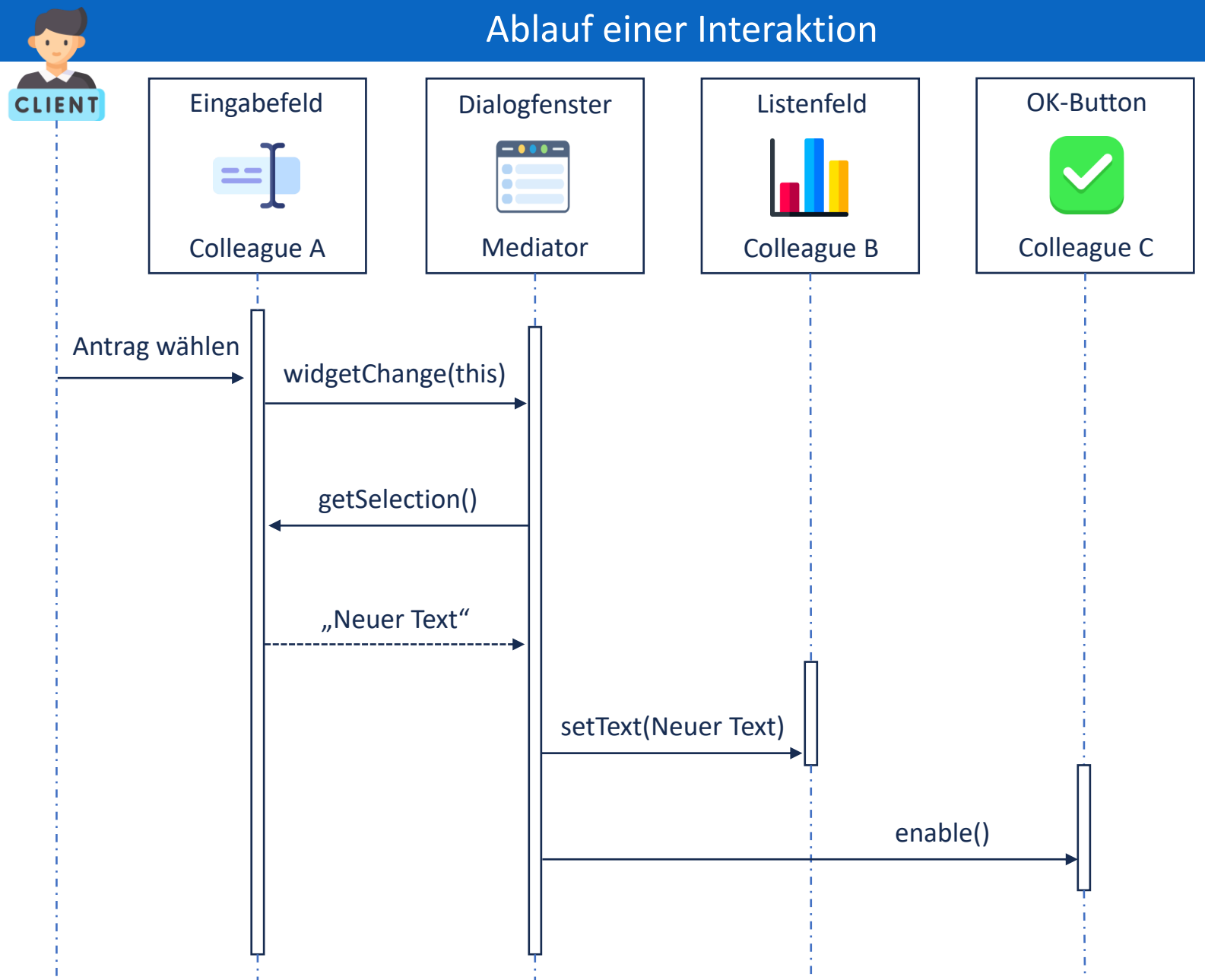


Zuordnung zum Dialogfenster

ConcreteMediator	Dialogfenster
ConcreteColleagueA	Eingabefeld
ConcreteColleagueB	Listenfeld
ConcreteColleagueC	OK-Button

Mediator: Kollaboration & Konsequenzen

Ablauf einer Interaktion



Konsequenzen (Vor- & Nachteile)

Vorteile

1. Entkopplung: Kollegen unabhängig wiederverwendbar.

2. Einfache Protokolle: ersetzt Viele-zu-Viele durch Mediator.

3. Zentralisiert Steuerung: komplexe Logik an einem Ort.

Nachteil

1. God Object: Mediator kann zu groß und schwer pflegbar werden.

Direktvergleich: Observer vs. Mediator

∞

Merkmal	Observer-Muster	Mediator-Muster
Kommunikation	Verteilt (1-zu-N Broadcast)	Zentralisiert (Hub-and-Spoke)
Hauptintention	Konsistenz zwischen Objekten wahren	Komplexe Interaktionen kapseln & steuern
Kopplung	Observer kennt das konkrete Subjekt; Subjekt kennt nur die abstrakte Observer-Schnittstelle.	Kollegen kennen Mediator; Mediator kennt alle Kollegen
Typische Nutzung	MVC, Event-Systeme	Komplexe UI, Workflows
Größte Gefahr	Kaskadierende Updates, Speicherlecks	Mediator wird zum "God Object"

Anwendungsfälle & Kombination

Wähle Observer, wenn...

- Wenn sich ein Objekt ändert und viele andere automatisch informiert werden sollen.
- Wenn das Hauptziel ist, alle verbundenen Objekte zu **synchronisieren**.
- Wenn die Informationen nur in eine Richtung fließen (von einem Objekt zu vielen anderen).

Wähle Mediator, wenn...

- Wenn mehrere Objekte auf komplizierte Weise miteinander reden müssen.
- Wenn du Objekte unabhängiger machen willst, damit sie einfacher wiederverwendbar sind.
- Wenn die Kommunikation zwischen den Objekten an einem **zentralen Ort** organisiert werden soll.

Kombination beider Muster

In komplexen Systemen kann der Observer die Kommunikation übernehmen.

Der Mediator kann dann als einziger Observer für alle anderen Objekte dienen.

Kernaussagen & Fazit

Observer = Verteilte Benachrichtigung:
Dient primär der Synchronisation von Zuständen.

Mediator = Zentralisierte Steuerung:
Dient primär der Vereinfachung komplexer Interaktionen.

Gemeinsames Ziel:
Beide Muster reduzieren die Kopplung und schaffen flexiblere, wartbarere Systeme.

Kontext ist entscheidend:
Die Wahl des richtigen Musters hängt immer vom spezifischen Kommunikationsproblem ab.

Quelle

Literatur

•Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software.* Addison-Wesley Professional.

Werkzeuge & Sonstiges

•Unterstützung: Künstliche Intelligenz (KI)

•Icons: Flaticon

Fragen

Vielen Dank für Ihre Aufmerksamkeit!