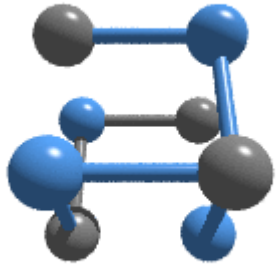


Observer|Chain of Responsibility|Mediator

Seminar Software Design Pattern - Sommersemester 2009
Vertragender: Christian Kube
Betreuer: Frank Schumacher

Paket Kommunikation - Übersicht

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

- Observer (Beobachter)
  aka. publish-subscribe (publiziere und aboniere)
- Chain of Responsibility (Zuständigkeitskette)
- Mediator (Vermittler)

Alle hier vorgestellten Muster gehören zur Kategorie der Verhaltens-muster (Behavioral Patterns).

Observer - Übersicht

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Observer

Beobachter aka. publish-subscribe (publiziere und aboniere)

Zweck: Definiere eine 1-zu-n-Abhängigkeit zwischen Objekten, so dass die Änderung des Zustands eines Objekts dazu führt, dass alle abhängigen Objekte benachrichtigt und automatisch aktualisiert werden.

Motivation: Aufrechterhaltung der Konsistenz interagierender Objekte in einem System ohne enge Kopplung der Klassen.

Observer - Problembeispiel

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

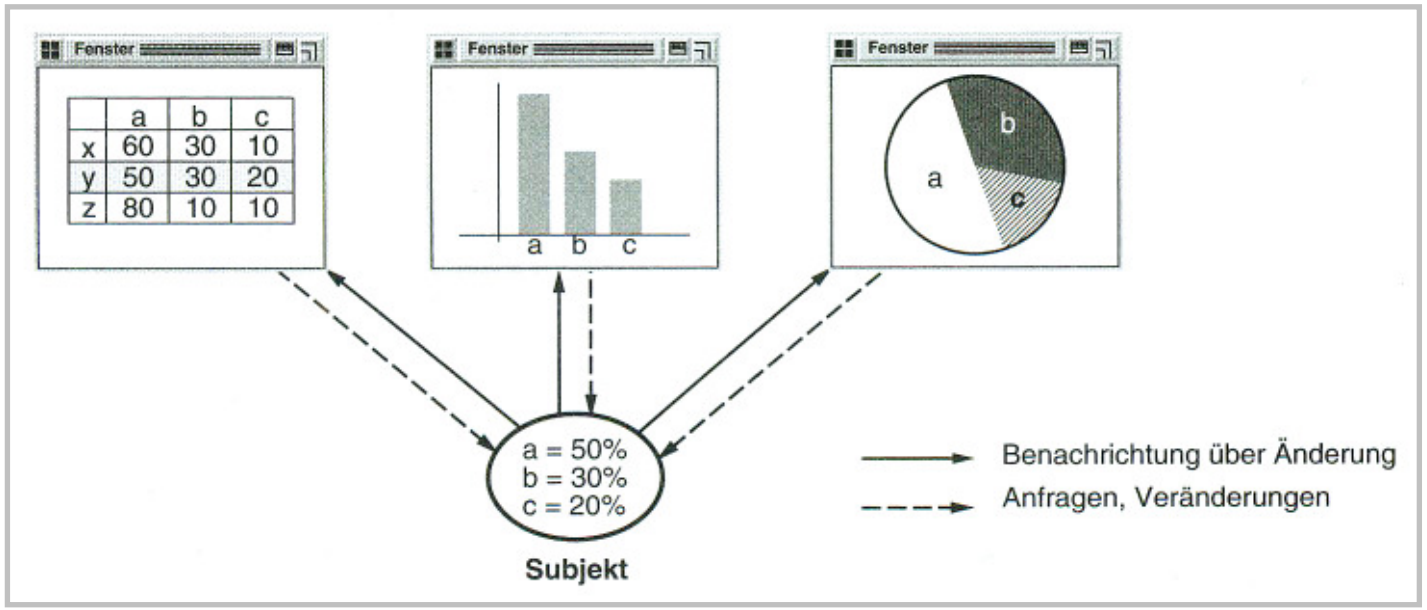
- Problembeispiel: Darstellung von Daten in einer Tabellen-kalkulation in „Echtzeit“.

- Lösung gesucht für:
  - Zugriff auf das gleiche Datenobjekt
  - geänderte Daten werden sofort auch von den anderen Darstellungen übernommen

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Observer - Problembeispiel - Lösungsansatz

Lösungsansatz durch Observer-Pattern:

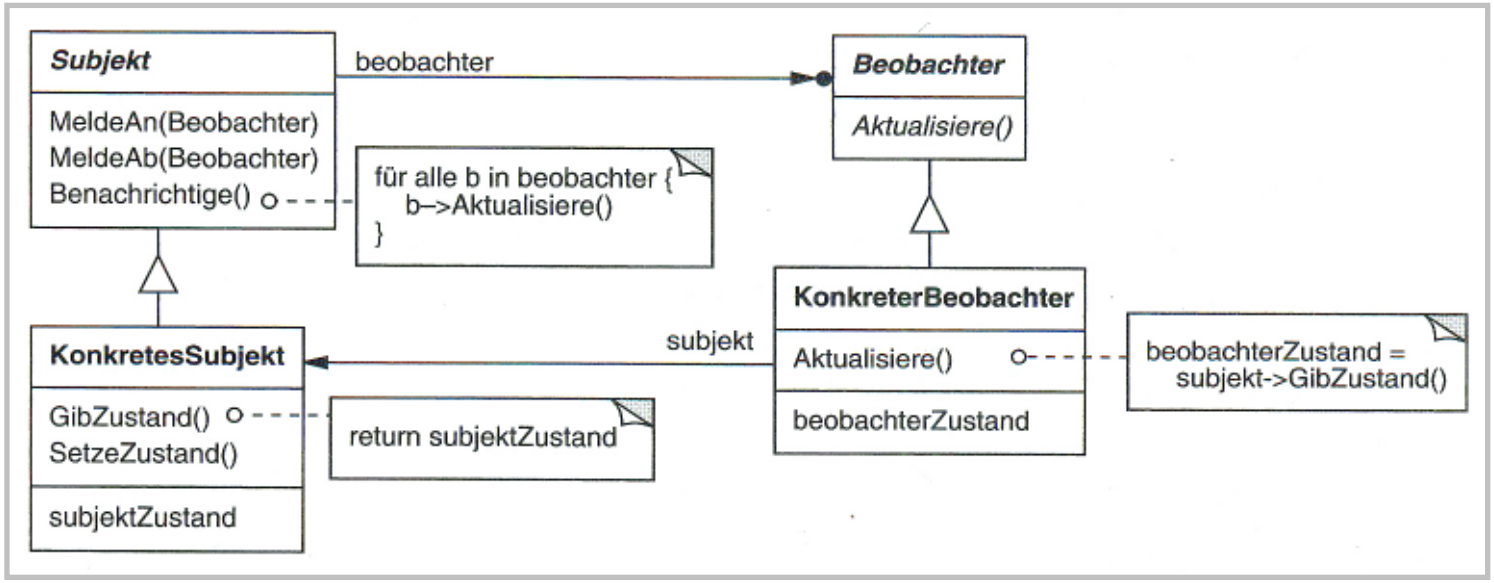


GoF - Abbildung 5.7

- Benutzer ändert Daten z.B. über ein Widget in der Tabellen-darstellung. Dieses führt die Änderung auf den Daten durch.
- Datenobjekt meldet eine Änderung an die Darstellungen
- Darstellungen fragen die Änderung beim Datenobjekt ab und ändern den dargestellten Wert

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Observer - Struktur und Interaktionen

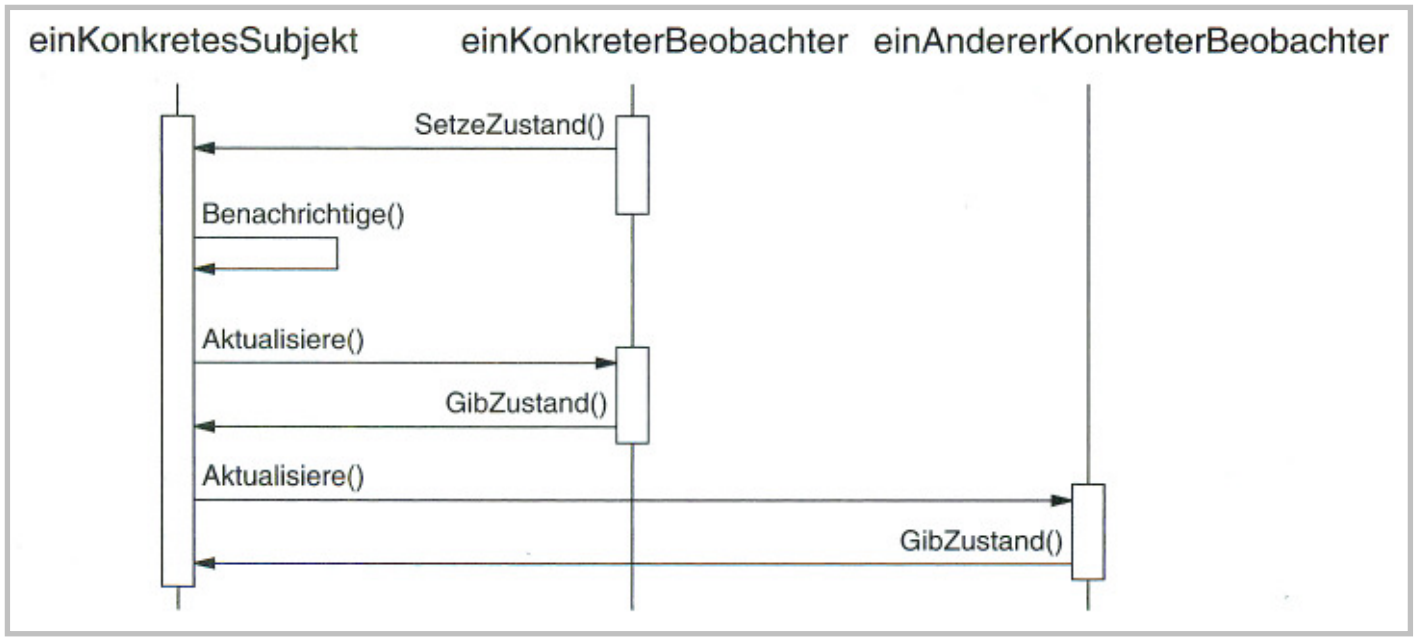


GoF - Abbildung 5.8

- KonkretesSubjekt benachrichtigt bei Zustandsänderung alle seine Beobachter
- nach Benachrichtigung erfragt KonkreterBeobachter neuen Zustand von KronkretesSubjekt
- KronkreterBeobachter synchronisiert seinen alten mit dem neuen Zustand

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Observer - Interaktion im Diagramm



GoF - Abbildung 5.9

Ist Beobachter auch der aktive Client, der eine Änderung am Subjekt vornimmt, so wird jedoch erst der vom Subjekt bei der Aktualisierung übermittelte Zustand übernommen und nicht der bei Initiierung der Änderung dem Beobachter bekannte (anzunehmende) neue Zustand.

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Observer - Konsequenzen

- Abstrakte Kopplung zwischen Subjekt und Beobachter
- Unterstützung von Broadcast-Kommunikation
- unerwartete Aktualisierungen

... für die Implementierung:

- direkte Referenzierung eventuell zu teuer
- Unterscheidung betreffendes Subjekt durch Beobachter
  - Übergabe des Subjekts selbst bei Benachrichtigung
- wann benachrichtigen um die Konsistenz zu wahren
  - direkt nach Änderung Benachrichtigung durch Subjekt

    vs.
  - Client startet Benachrichtigung selbst

    (weniger Aktualisierungen aber fehleranfälliger)

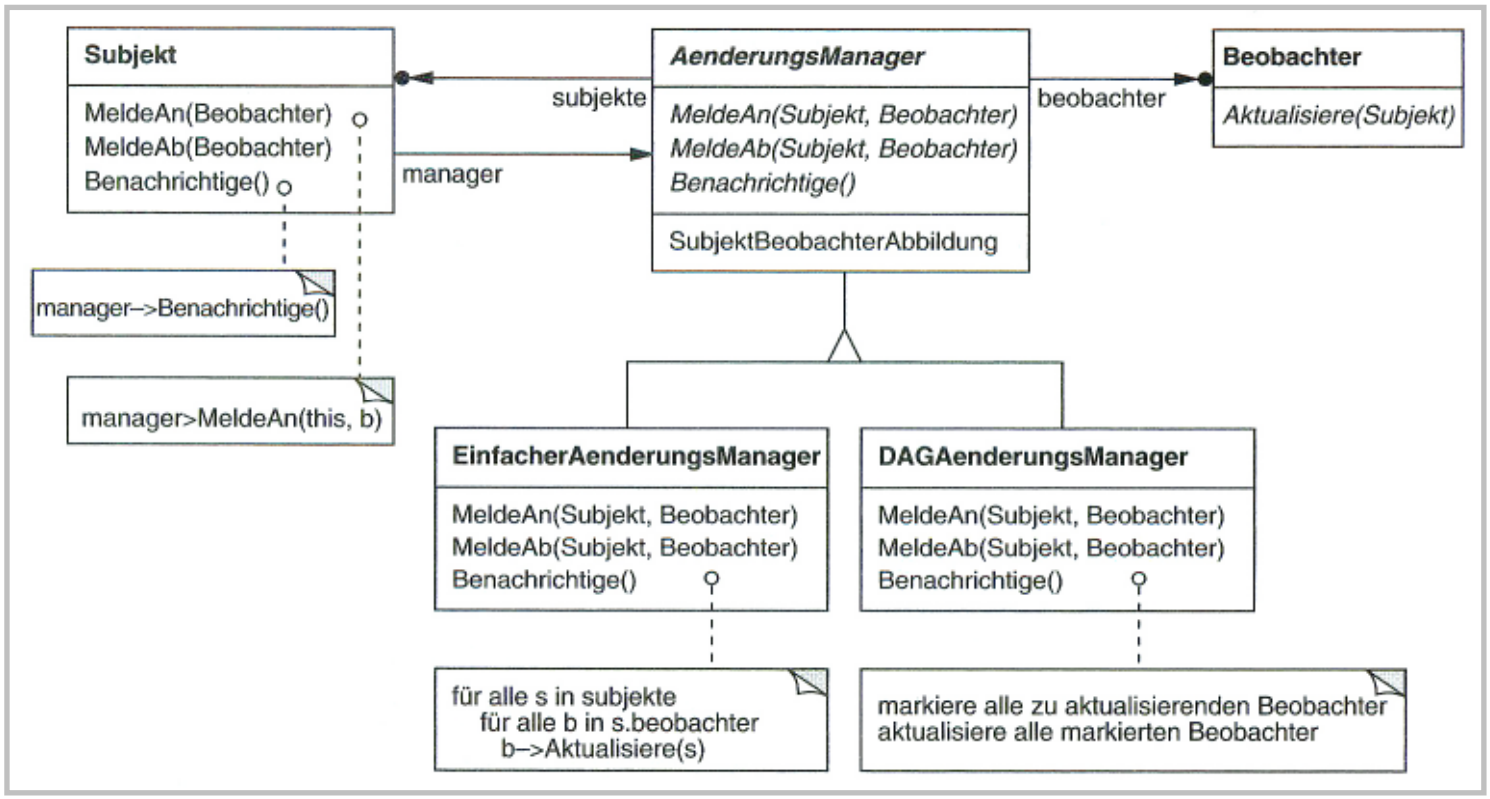
UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Observer - Konsequenzen (2)

- Referenzen auf gelöschte Objekte
- Konsistenz bei Benachrichtigung
- Vermeidung beobachterspezifischer Aktualisierungsschnittstellen
  Push- und Pull-Modell
  Push: Subjekt teilt gleichzeitig Details über Änderungen mit
  Pull: Subjekt gibt minimal Info, Beobachter erfragt Details
- Benachrichtigung nur bei Interesse an der Änderung
- Kapseln komplexer Änderungen/Aktualisierungssemantik
  - Lösung: ÄnderungsManager (siehe Abb. 5.10)
    - Verwaltet Referenzen auf Subjekte und Beobachter
    - Definition bestimmter Aktualisierungsstrategien
    - benachrichtigt stellvertretend alle Beobachter
    - EinfacherÄnderungsManager vs. DAGÄnderungsManager
- Kombination von Subjekt- und Beobachterklassen

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Observer - Konsequenzen (3)



GoF - Abbildung 5.10

- ÄnderungsManager ist Anwendung von Mediator (siehe Teil 3)
- Als Singleton?

Observer - Anwendung

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Nachträgliche Implementierung in bestehende System ist schwierig bis unmöglich.

Observer empfiehlt sich bei:

- Abstraktionen mit zwei voneinander abhängigen Aspekten
- Objektänderung in Abhängigkeit anderer Objekte
- Benachrichtigung anderer Objekte bei loser Kopplung

- Siehe Model/View/Controller-Konzept (MVC)
  - Model stellt Subjekt dar
  - View übernimmt quasi Beobachterfunktion

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Chain of Responsibility

Zuständigkeitskette

Zweck: Vermeide die Kopplung des Auslösers einer Anfrage mit seinem Empfänger, indem mehr als ein Objekt die Möglichkeit erhält, die Aufgabe zu erledigen. Verkette die Empfängerobjekte und leite die Anfrage an der Kette entlang, bis ein Objekt sie erledigt.

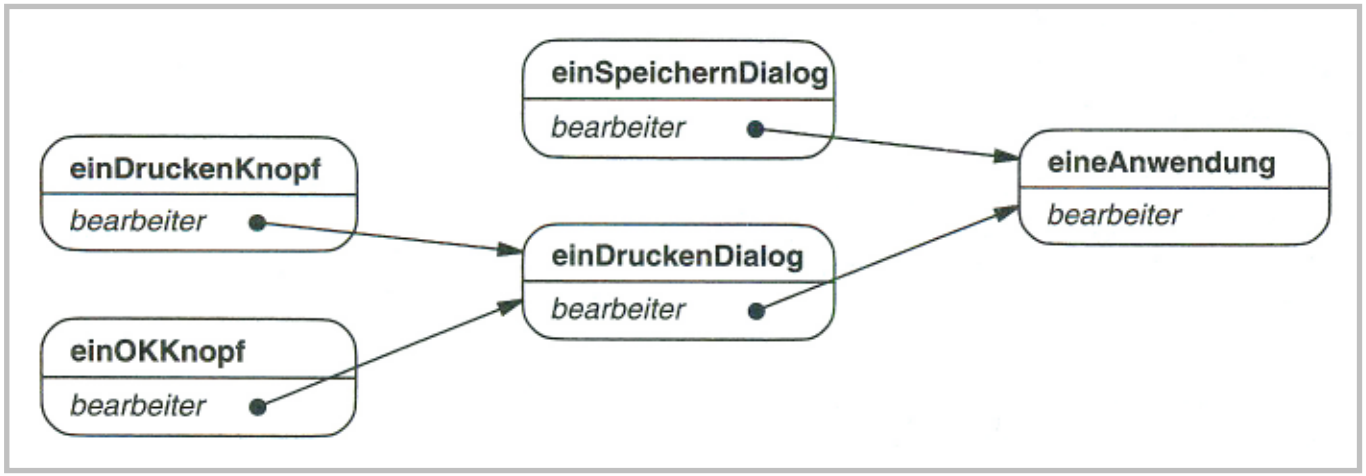
Motivation: Entkopplung interagierender Klassen (Sender und Empfänger), wobei (möglichst) der optimalste Empfänger ermittelt wird.

Chain of Responsibility - Problembeispiel

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Problembeispiel:

- Bereitstellung von Hilfeinformationen in einem Drucken-Dialog
  - der Klient soll die beste verfügbare Hilfe erhalten

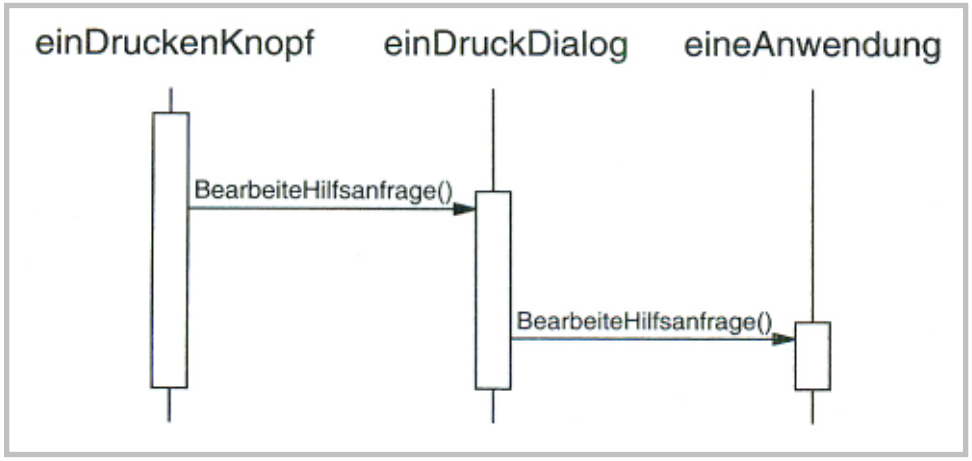


GoF - Abbildung 5.40

- Hilfeanforderung für Knopf „Drucken"
- Benutzer erwartet Hilfe für den Knopf, wenn nicht, dann zumindest irgendeine Antwort

Chain of Responsibility - Problem-Lösungsansatz

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme



GoF - Abbildung 5.41

Weiterleitung der Hilfsanfrage entlang der Objekthierarchie.

- Hilfe für Drucken-Knopf und -Dialog nicht verfügbar
  > Weiterleitung der Anfrage bis zur Anwendung selbst

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

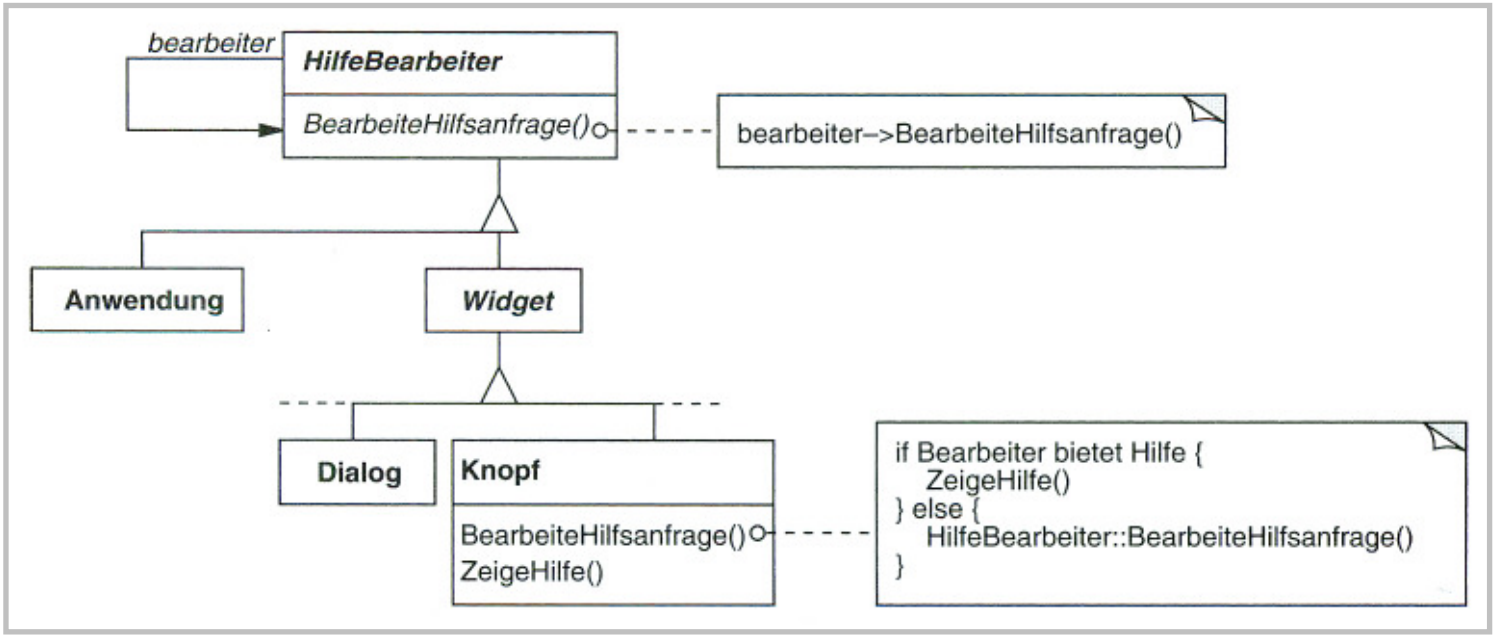
Chain of Responsibility - Problem-Lösungsansatz(2)

Vorgabe:

- Klient besitzt keine Referenz auf das antwortende Objekt
- Empfänger (Bearbeiter) kennen sich untereinander nicht

Lösung:

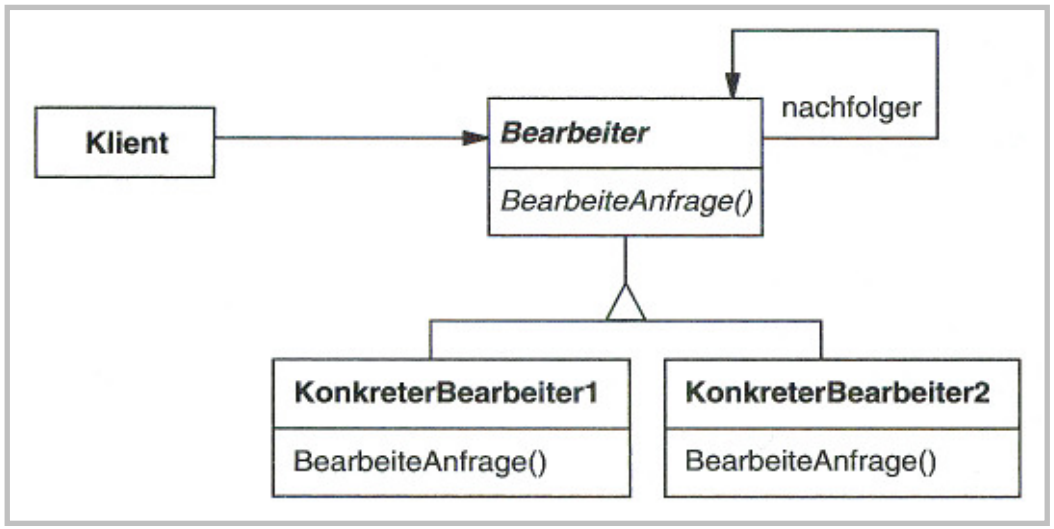
- Definition einer Oberklasse HilfeBearbeiter



GoF - Abbildung 5.42

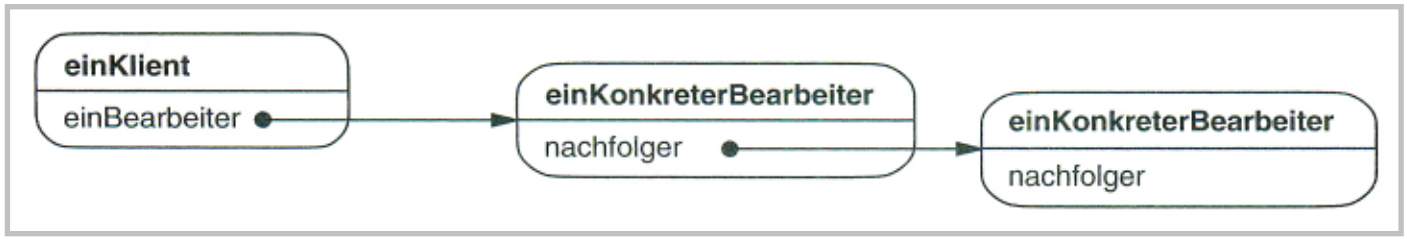
UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Chain of Responsibility - Struktur



GoF - Abbildung 5.43

mit Objektstruktur



GoF - Abb. 5.44

Chain of Responsibility - Interaktionen

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

- Bearbeiter stellt Schnittstelle für die Bearbeitung zur Verfügung und eventuell auch die Verbindung zum Nachfolger
- KonkreterBearbeiter antwortet nur, wenn auch dafür zuständig, sonst Weiterleitung an nächsten Bearbeiter, falls verfügbar.
- Klient löst initiale Anfrage aus.

Chain of Responsibility - Konsequenzen

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

- Reduzierte Kopplung
  - Objekt (Klient) muss nicht wissen, wer antwortet
  - Sender und Empfänger müssen sich nicht kennen
  - Vereinfachung: nur Nachfolger statt alle möglichen Bearbeiter
- Zusätzliche Flexibilität (auch dynamisch)
  - Zuständigkeiten können eingefügt oder
  - die Kette kann reorganisiert werden
- keine Abarbeitungsgarantie
  - ist letzter Bearbeiter in der Kette nicht zuständig ...

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Chain of Responsibility - Konsequenzen (2)

... für die Implementierung:

- Nachfolgerkette
  - Erstellung neuer Verbindungen (wie im Beispiel)
    vs.
  - Verwendung existierender Verbindungen
  - > Lösung mit Hilfe des Kompositionsmuster (Composite)
- Verbinden von Nachfolgeobjekten
  - keine Kette vorhanden(?), Defaultimplementierung notwendig
- Repräsentation von Anfragen
  - bisher nur Bearbeitung/Weiterleitung für definierte Anfragen
  - mittels Anfragecode/ID (einheitliche Codierung erforderlich und unsicher, da keine typsichere Weitergabe)
  - mittels Anfrageklasse (Parameterübergabe möglich, aber spezialisierte Kenntnisse über Anfrageklassen beim Bearbeiter erforderlich)
- Automatische Weiterleitung an KeineAntwort (SmallTalk-like)

Chain of Responsibility - Anwendung

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Chain of Responsibility empfiehlt sich, wenn:

- die Antwort auf eine Anfrage durch ein vorab unbekanntes, zur Laufzeit erst bestimmtes Objekt erfolgen können soll.
- eine Anfrage an mehrere Objekte gerichtet werden soll, ohne den richtigen Empfänger zu kennen.
- beantwortende Objekte erst zur Laufzeit festgelegt werden.

Mediator - Übersicht

Mediator

Vermittler

Zweck: Definiere ein Objekt, welches das Zusammenspiel einer Menge von Objekten in sich kapselt. Vermittler fördern lose Kopplung, indem sie Objekte davon abhalten, aufeinander explizit Bezug zu nehmen. Sie ermöglichen es Ihnen, das Zusammenspiel der Objekte von ihnen unabhängig zu variieren.

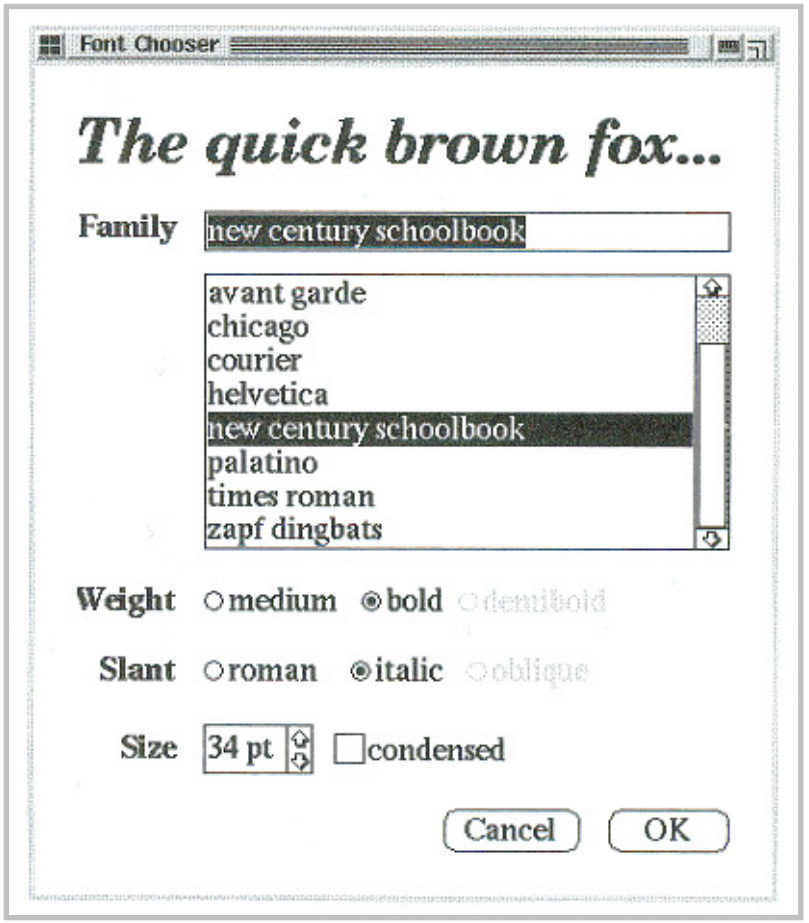
Motivation: Um die Bildung eines monolithischen Klotzes durch die steigende Anzahl von Verbindungen von Objekten in einem wachsenden System zu vermeiden, wird durch die Einführung (globaler) Vermittler versucht, das Verhalten verbundener Objekte zentral zu steuern. Auch spätere Änderungen von Aktionen sollen so effektiver erfolgen können, da die Steuerung nicht mehr weit verteilt in den Objekten erfolgt.

Mediator - Problembeispiel und Lösungsansatz

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Problembeispiel:

- Eingabefelder sind unterschiedlich abhängig untereinander
- z.B. deaktivierte Optionen, solange nichts ausgewählt
- bei Auswahl in Liste automatische Übernahme in Textfeld



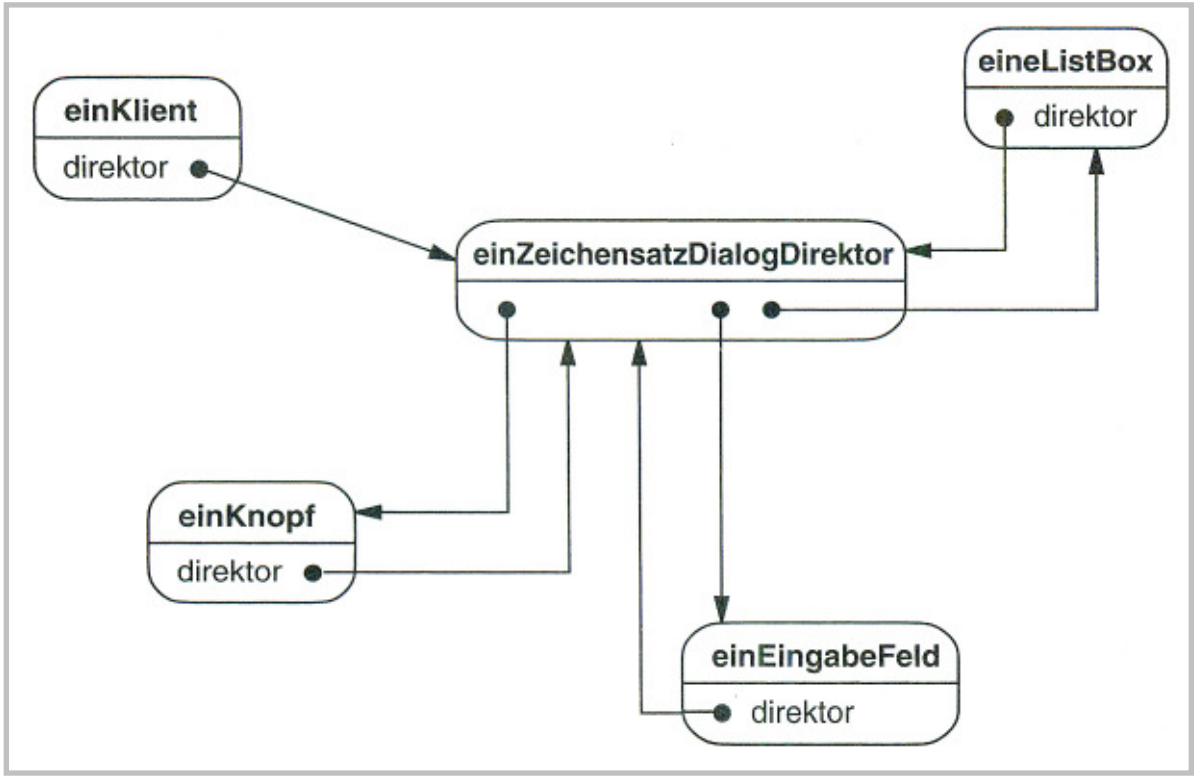
GoF - Abbildung 5.30

Lösungsansatz:

- Kapseln des Gesamtverhaltens in einem separaten Vermittlerobjekt

Mediator - Problembeispiel - Lösungsansatz

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme



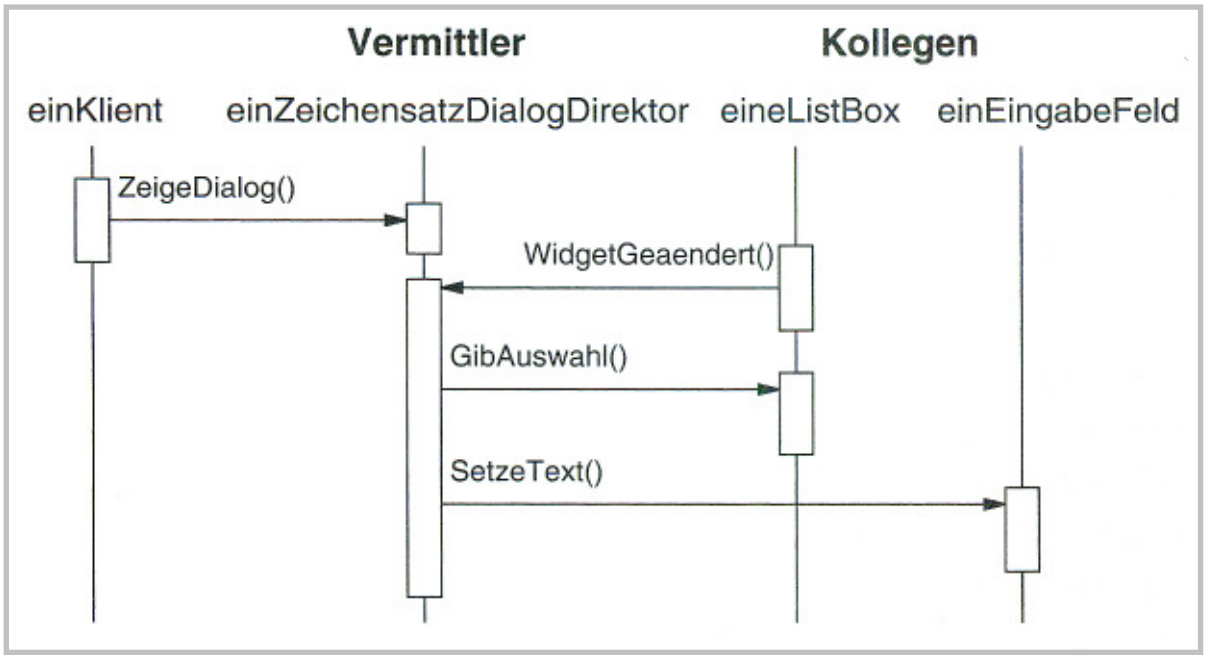
GoF - Abbildung 5.31

- Ein Widget meldet Änderung seinem Vermittler.
- Vermittler holt sich Änderung und führt Änderungen der anderen Widgets gemäß festgelegtem Verhalten durch.

Mediator - Problembeispiel - Interaktion

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Die Interaktion im Diagram

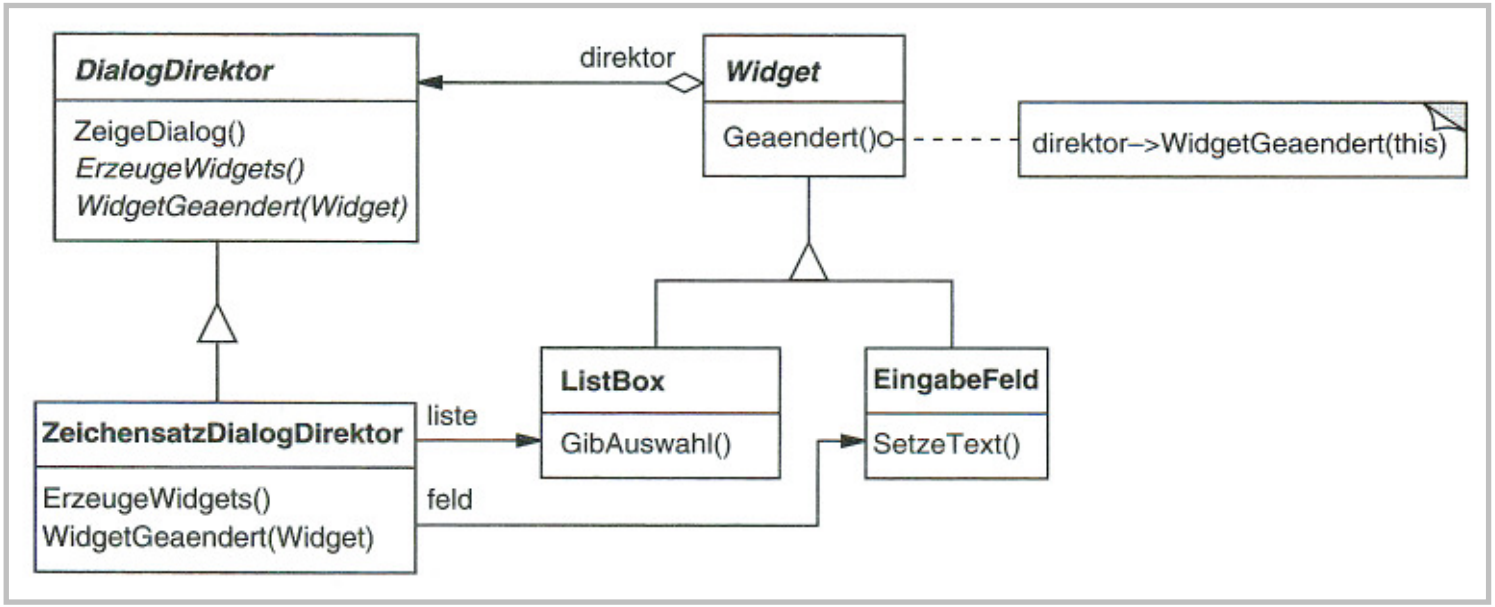


GoF - Abbildung 5.32

- unvollständig, da der Vermittler durch das Vorhandensein einer Eingabe im Textfeld nun auch die Optionen für die Schriftart aktiviert sollte

Mediator - Problembeispiel - Lösung im Detail

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

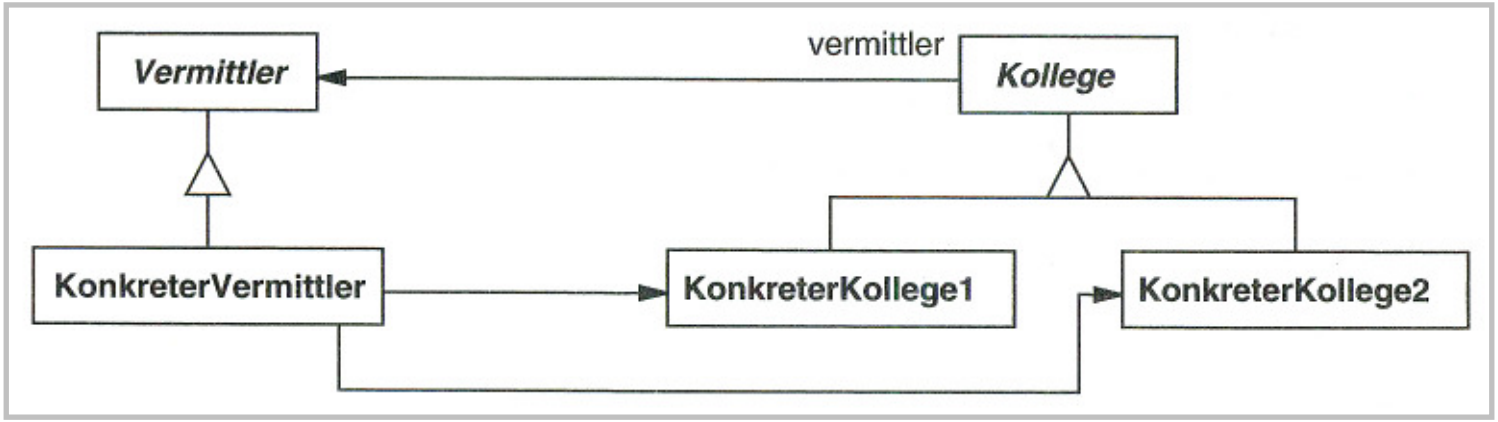


GoF - Abbildung 5.33

- Klienten rufen zeigeDialog des DialogDirektors auf und arbeiten anschließend auf den Widgets
- DialogDirektor legt Gesamtverhalten eines Dialogs fest, definiert die abstrakten Methoden ErzeugeWidgets und WidgetGeaendert

Mediator - Struktur

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

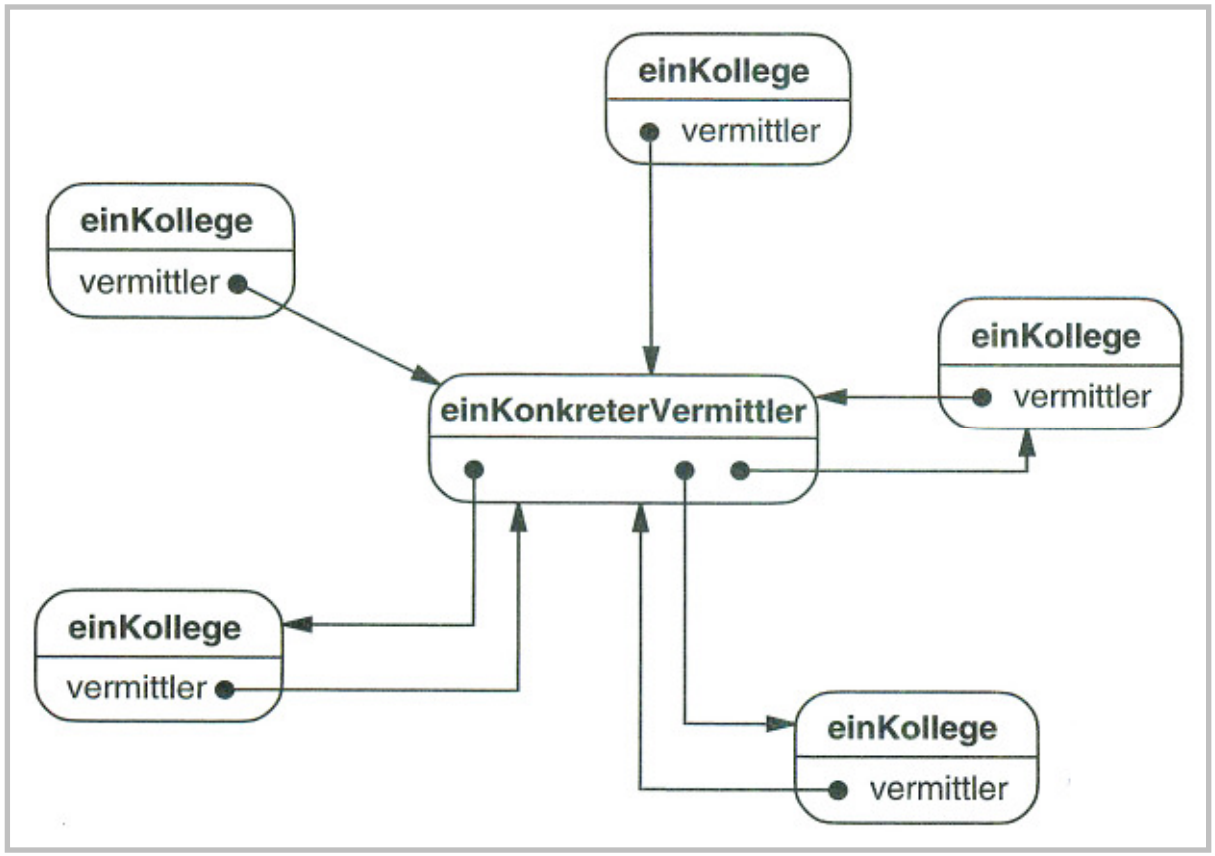


GoF - Abbildung 5.34

- Vermittler definiert Schnittstelle für die „kollegiale“ Interaktion
- KonkreterVermittler kennt und koordiniert die Kollegen
- Kollegen kennen nur ihren Vermittler und keine Abhängigkeiten

Mediator - Interaktion

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme



GoF - Abbildung 5.35

- Senden und Empfangen nur zwischen Kollegenobjekte und Vermittler

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Mediator - Konsequenzen

- Verringerung der Unterklassenbildung
- Entkopplung der Kollegenobjekte
- Vereinfach des Protokolls
  - Ersetzen von n-zu-n- durch zentrale 1-zu-n-Beziehungen
- Abstrahierung der Zusammenarbeit der Objekte
- Zentralisierte Steuerung
  - monolithisches System vs. monolithisches Objekt

... für die Implementierung:

- Weglassen der abstrakten Vermittlerklasse
- Interaktion zwischen Vermittler und Kollegen durch das Beobachtermuster (Observer)
- mehr Informationen durch spezielle Schnittstelle für die Benachrichtigung des Vermittler

Mediator - Anwendung

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Mediator empfiehlt sich bei:

- einer Menge von interagierenden Objekten, wohldefiniert aber komplex und mit unstrukturierten Abhängigkeiten
- Objekten, deren Wiederverwendung durch zu viele Abhängigkeiten und Interaktionen erschwert wird.
- der Vermeidung der Bildung diverser Unterklassen trotz komplexem Verhalten dieser untereinander

Literatur

UNIVERSITÄT LEIPZIG
Institut für Informatik
Betriebliche Informationssysteme

Inhalt und Illustration komplett aus

- Entwurfsmuster
  Elemente wiederverwendbarer objektorientierter Software
  Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides
  ADDISON-WESLEY