



Chain of Responsibility

Eric, Marko

Inhalt

- Was ist die „Chain of Responsibility“?
- Problem
- Lösung
- Beispiel
- Struktur
- Anwendbarkeit
- Pros & Cons
- Relationen mit anderen Design Patterns
- Code

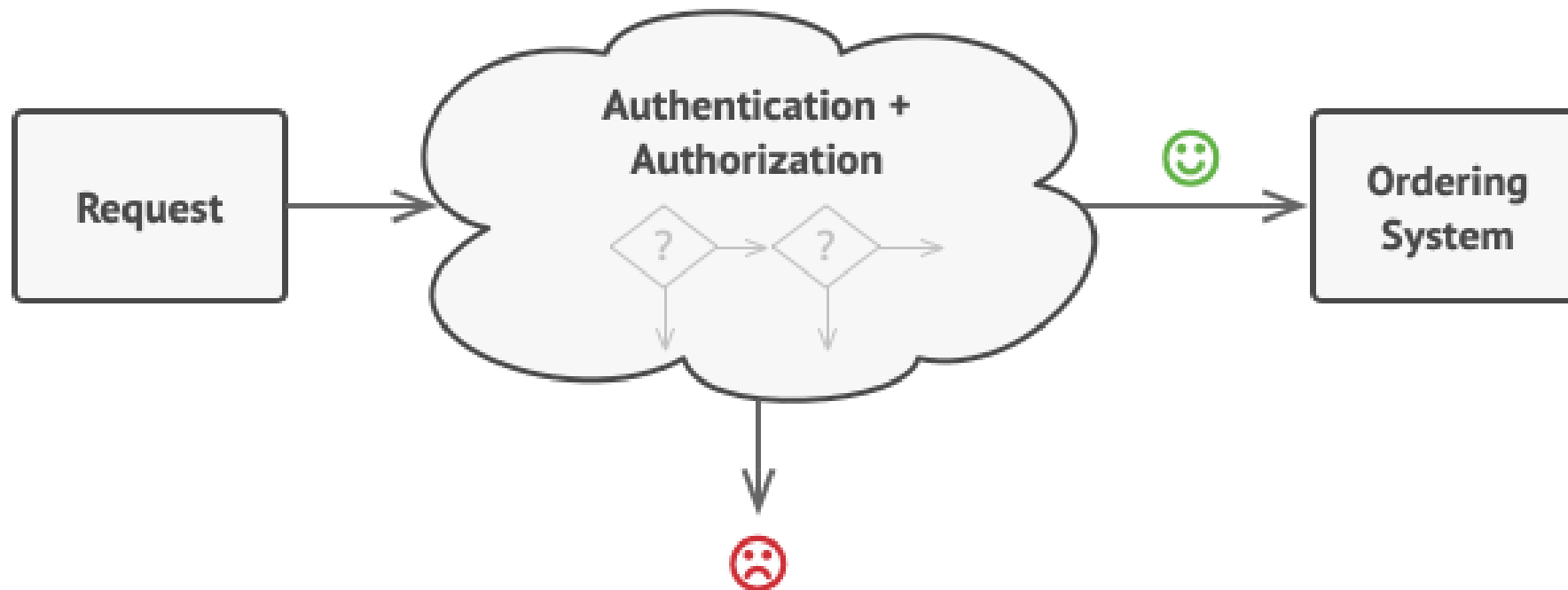
Was ist die „Chain of Responsibility“?

- Behavioral Design Pattern
- Algorithmen & Verantwortlichkeit zwischen Objekten
- Requests → Kette von Handlern
- Handler können:
 - Requests empfangen
 - Request bearbeiten
 - Request zum nächsten Handler senden
 - Weitergabe abbrechen



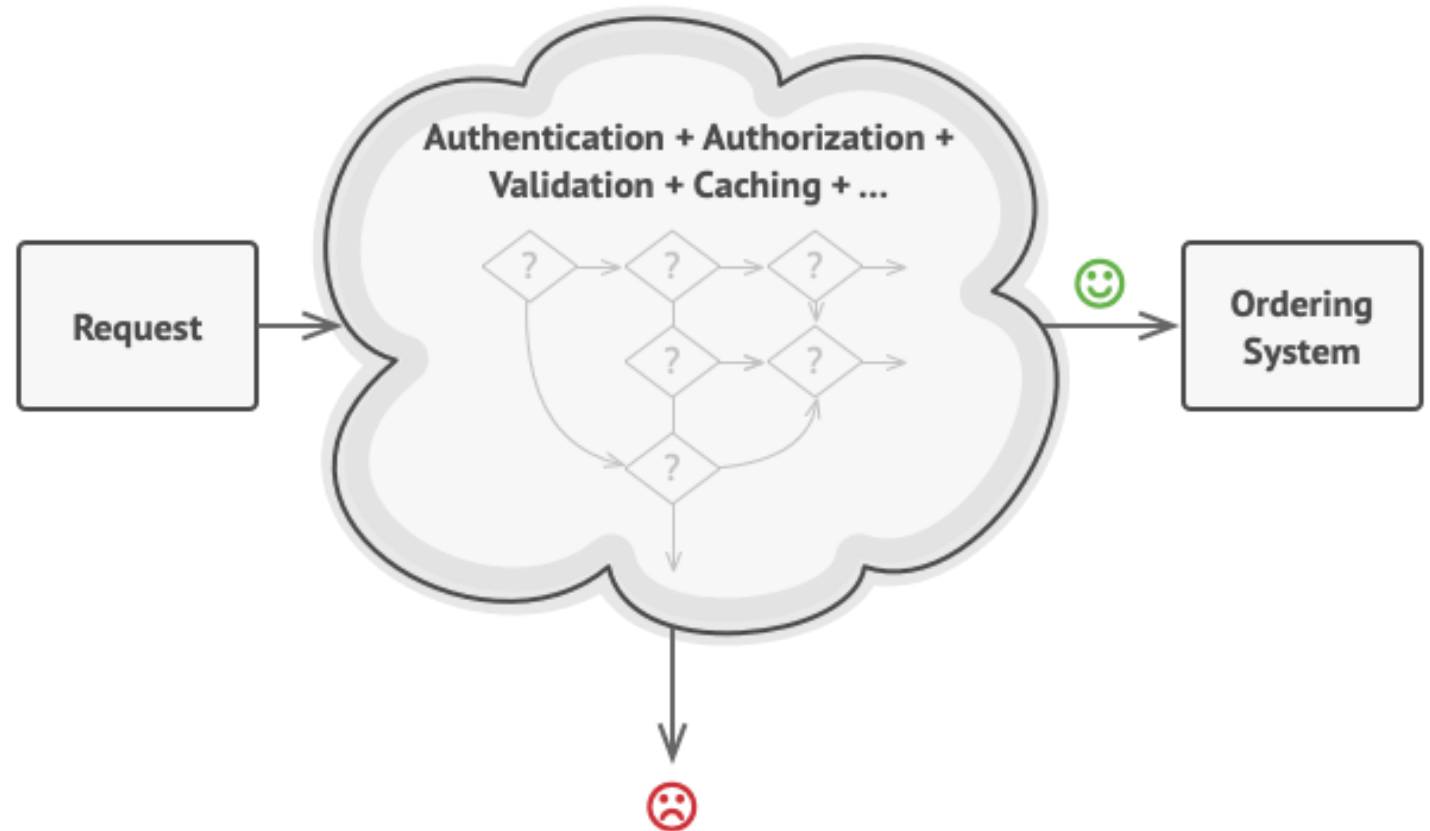
Probleme

- Request muss eine Reihe von Checks durchlaufen
- Checks müssen sequenziell erfolgen



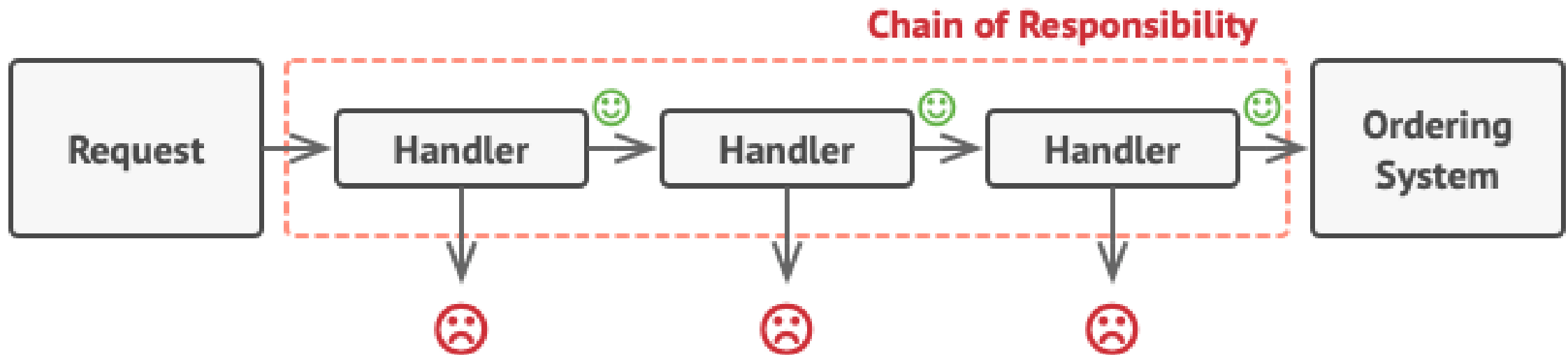
Probleme

- Code wächst und wird unübersichtlicher
- Teure Wartung des Systems
- Änderungen können mehrere Checks beeinflussen
- Mehrmalige Benutzung gleicher Codeteile

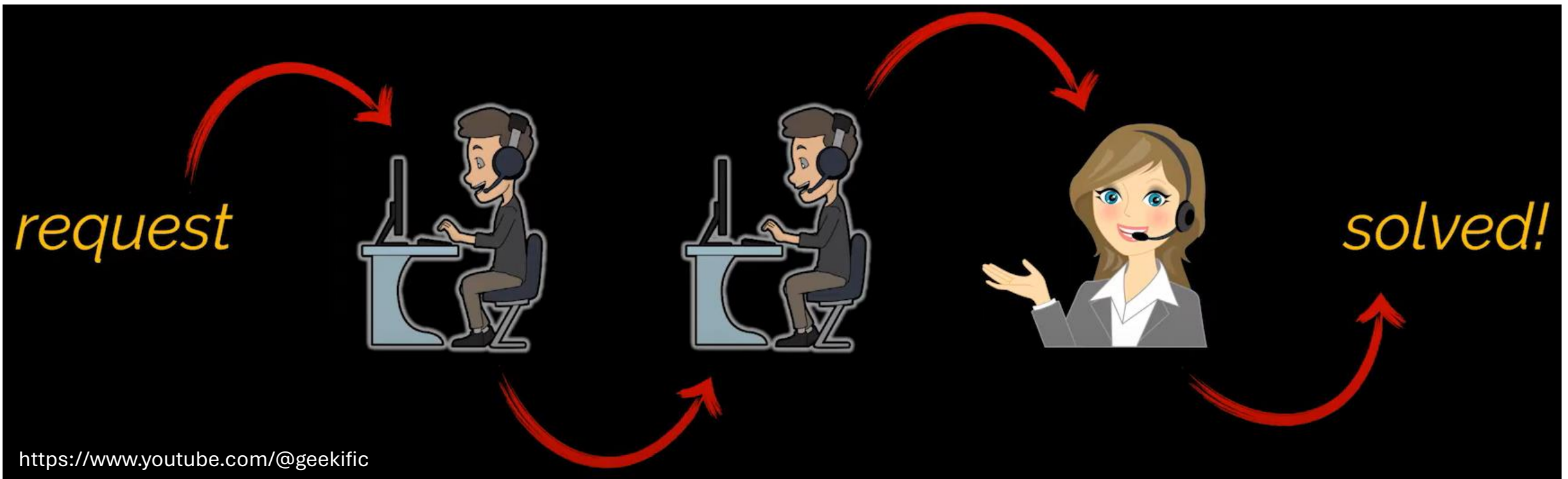


Lösung

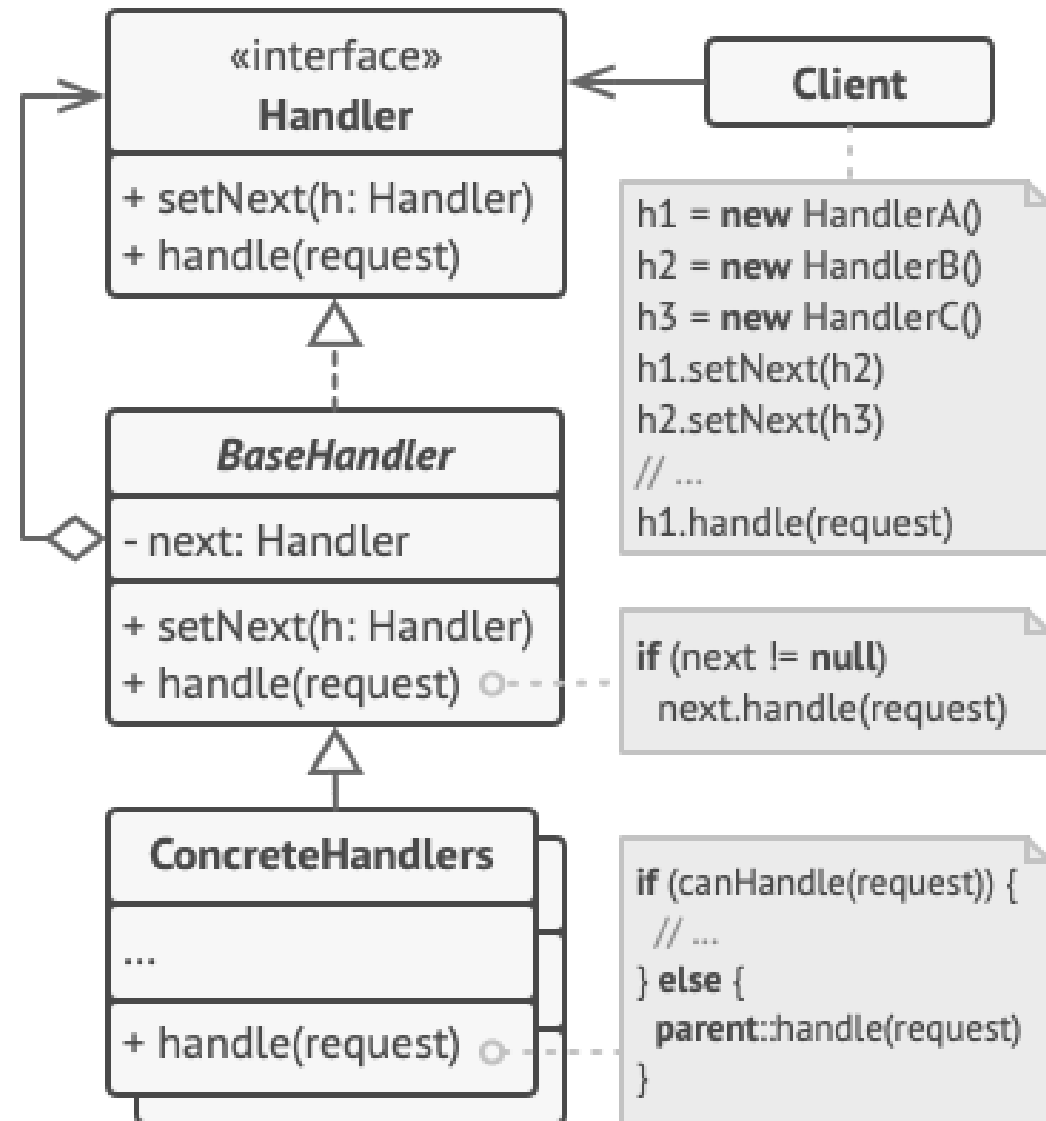
- Request wird durch eine Reihe von Checks durchgegeben
- Handler → Feld zum Speichern eines Verweises
- Handler → Request wird zum nächsten weitergeleitet
- Requests können die Weitergabe abbrechen



Beispiel



Struktur



Anwendbarkeit

Anfragen auf verschiedene Art verarbeitet werden sollen

→ genaue Arten und Reihenfolge vorher nicht bekannt

Mehrere Handler in einer bestimmten Reihenfolge

Menge der Handler soll sich zur Laufzeit ändern

Pros & Cons

Pros

- Request-Handling Reihenfolge kontrollierbar
- Single Responsibility Principle
- Open/Closed Principle
- Flexibilität und Erweiterbarkeit

Cons

- Mögliche unbehandelte Requests
- Komplexeres Debugging
- Laufzeitkonfigurationsaufwand

Relationen mit anderen Design Patterns

Command,
Mediator &
Observer

Sender und Empfänger von Requests miteinander verbinden

Decorator

Sehr ähnliche Klassenstruktur

Leitung von Ausführungen durch eine Reihe von Objekten



Code