Analysemuster

Marc Monecke
monecke@informatik.uni-siegen.de

Praktische Informatik
Fachbereich Elektrotechnik und Informatik
Universität Siegen, D-57068 Siegen

12. Mai 2003

Inhaltsverzeichnis

1 Grundlagen

- **Muster** (*pattern*) gehen zurück auf den Musterbegriff des Architekten Christopher Alexander (70er Jahre) für Bauwerke, Städteplanung
- in der **Softwaretechnik**: Suche nach einem **Architekturhandbuch** für den Software-Entwurf (90er Jahre)
- Einsatz in Analyse, Entwurf, Codierung
- 1995: Erich Gamma et al. schreiben Informatik-Bestseller
  Design Patterns – Elements of Reusable Object-Oriented Software
- darin werden 23 **Entwurfs**muster beschrieben und klassifiziert

1.1 Was sind Muster?

Umgangssprachlich:

Muster == Vorlage, Vorbild, sich wiederholende Struktur

in der Softwaretechnik:

- **Vorlage** für die Konstruktion von Problemlösungen
- **Vorbild** für die Beschreibung und Dokumentation von Entwürfen
- **Struktur**, die bei der Orientierung in komplexen Systemen hilft

Musterbücher enthalten Katalog von Mustern

- allgemeine Muster
- anwendungsspezifische Muster

ermöglichen das Auffinden des passenden Musters, um ein gegebenes Problem zu lösen

Beschreibung von Mustern

- Name, möglichst sprechend
- Kategorie
- Beschreibung
- Beispiele (abstrakt und konkret)

Einsatzbereiche

- Systemanalyse
- Entwurf
- Codierung

entsprechend wechselnder **Abstraktionsgrad**

1.2 Charakteristika eines Musters

- beschreibt **erprobte**, bewährte Lösungsstruktur für **wiederkehrendes** Problem
- ist **uncodiert** (im Ggs. z.B. zu Klassenbibliothek)
- dokumentiert eine **Problemlösung**
- erleichtert die **Kommunikation** zwischen Entwicklern
- hält **Wissen und Erfahrung** von Experten fest, macht beides **zugänglich** für andere
- enthält nicht nur Einzelklassen, sondern **Konfigurationen** von Klassen
  → Wiederverwendung kommunizierender Gruppen von Klassen
  (**Mikro-Architekturen**)

1.3 Beziehungen zwischen Klassen

→ halten Klassen im Muster zusammen

- **Generalisierung**/Spezialisierung → Klasse ist spezieller als andere
- **Aggregation**/Komposition → Klasse umfaßt andere
- **Assoziation** → Kommunikation, Dienstaufruf

beachte: Objekt- vs. Klassenbeziehungen!

Muster geben vor

- **Interaktionen** zwischen Klassen
- **Verantwortlichkeiten** der Klassen

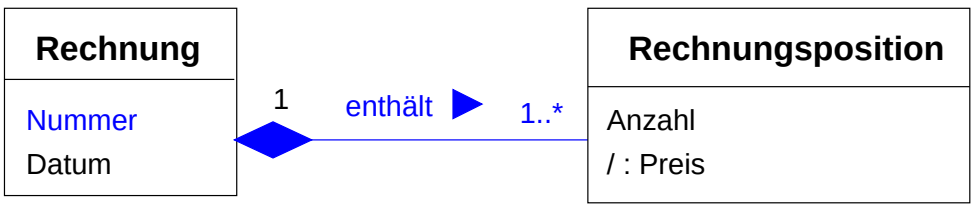
2 Einige Analysemuster mit Beispielen

Eingesetzt in der **Systemanalyse**

nach Heide Balzert: Lehrbuch der Objektmodellierung; LE 5

diese und weitere Beispiele auch in Helmut Balzert: Lehrbuch der Software-Technik; LE 12; Seite 349ff

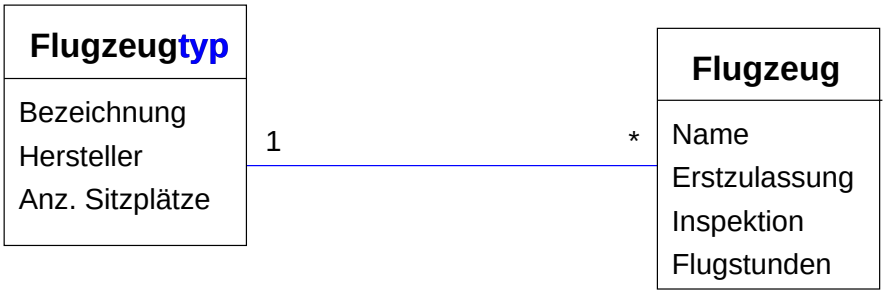
2.1 Muster 1: Liste



- **Komposition** (`Rechnungsposition` kann nicht allein existieren)

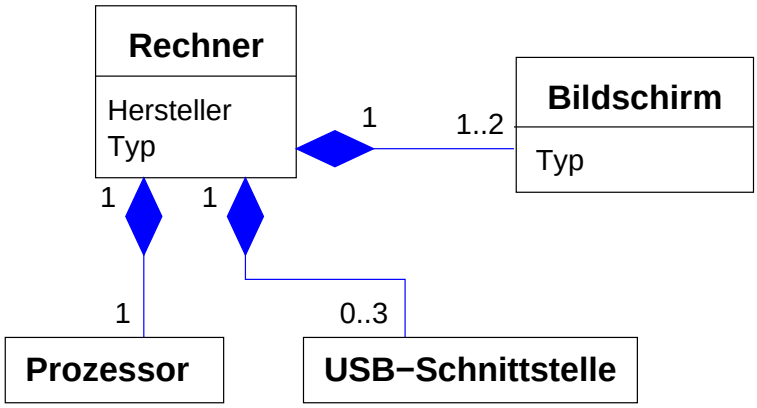
- **gleichartige** Teile → nur eine Teil-Klasse
- Teile sind **einem** Aggregat fest zugeordnet
- Attributwerte des Aggregats gelten auch für Teile (`Nummer` der Rechnung)
- Aggregat enthält i.a. mindestens ein Teil (`1..*`)

2.2 Muster 2: Exemplartyp



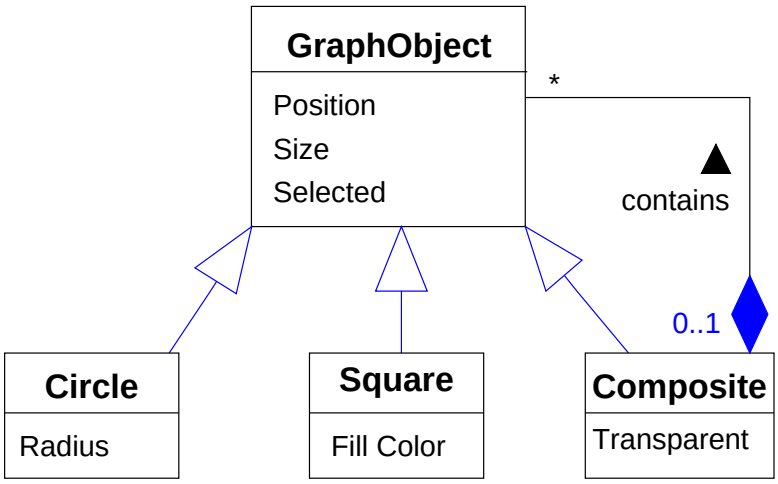
- Ziel: Redundanz vermeiden (Flugzeugeigenschaften)
- Instanzen, Exemplare vs. **-typ**, -gruppe, -beschreibung
- einfache Assoziation
- Verbindungen nicht verändert, nur gelöscht (Flugzeug verschrotten)
- Beschreibung auch ohne Exemplare möglich (*)

2.3 Muster 3: Baugruppe



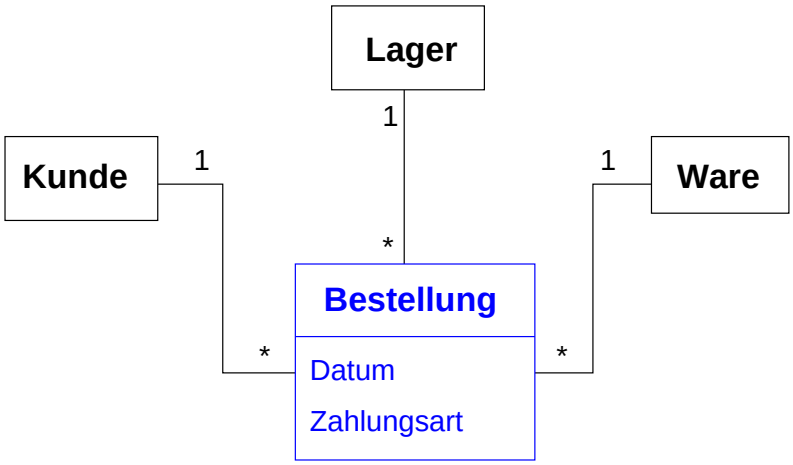
- physische Objekte
- Komposition
- Objektverbindungen bestehen (meist) über längeren Zeitraum
- Trennen der Objekte möglich (Bildschirm an anderen Rechner anschließen)

2.4 Muster 4: Stückliste



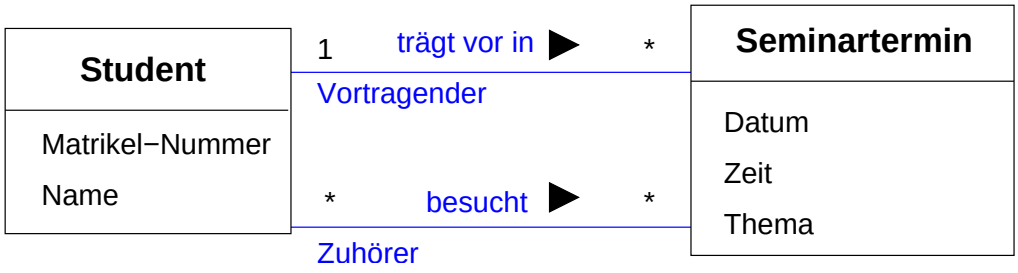
- Aggregat kann aus mehreren Objekten der anderen Klassen zusammengesetzt sein
- Aggregat und Teile einzeln **und** als Einheit handhabbar (kopieren, verschieben, löschen)
- Komposition
- Kardinalität an Aggregat: `0..1` → Teil kann auch allein existieren
- Sonderfall: nur ein Teile-Typ

2.5 Muster 5: Koordinator



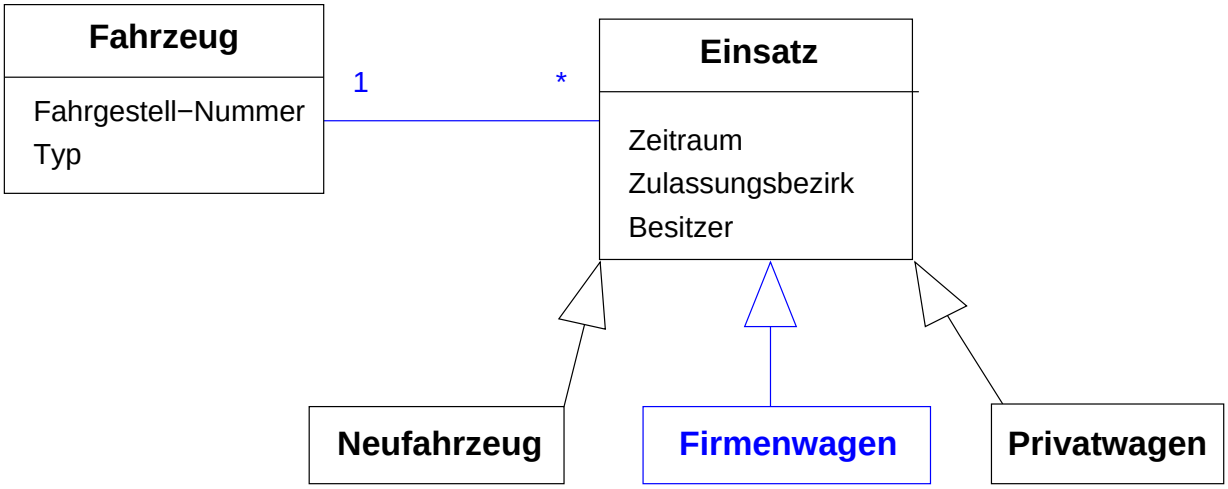
- Koordinator ersetzt n-äre Assoziation ($n \geq 2$) mit assoziativer Klasse (hier: `Bestellung`)
- einfache Assoziationen
- bei Koordinator **Beziehungen** wichtig, nicht Attribute

2.6 Muster 6: Rollen



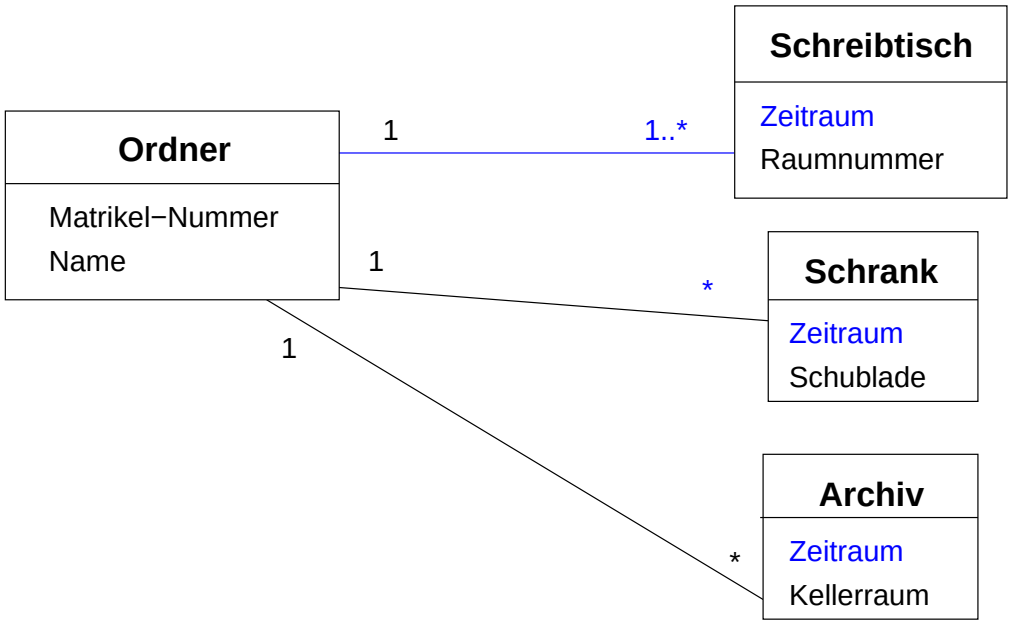
- Objekt kann in Bezug zu Objekten der anderen Klasse **mehrere Rollen** zur gleichen Zeit einnehmen
- $n \geq 2$ Assoziationen zwischen beiden Klassen
- Objekte haben, unabhängig von der Rolle, jeweils **gleiche Eigenschaften**

2.7 Muster 7: Wechselnde Rollen



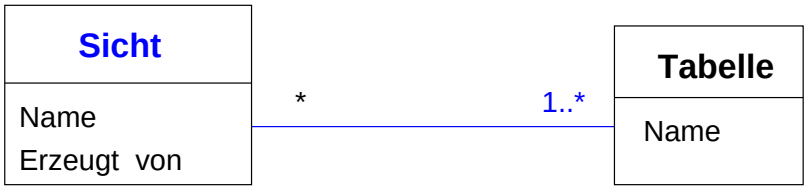
- Objekt kann für bestimmten **Zeitraum** unterschiedliche Rollen annehmen
- dabei **ändern** sich seine **Eigenschaften** → **Subklassen**
- Verbindungen zwischen Objekten nur **erweitern**, nicht löschen oder zu anderen Objekten umbiegen ('Geschichte' aufzeichnen)

2.8 Muster 8: Historie



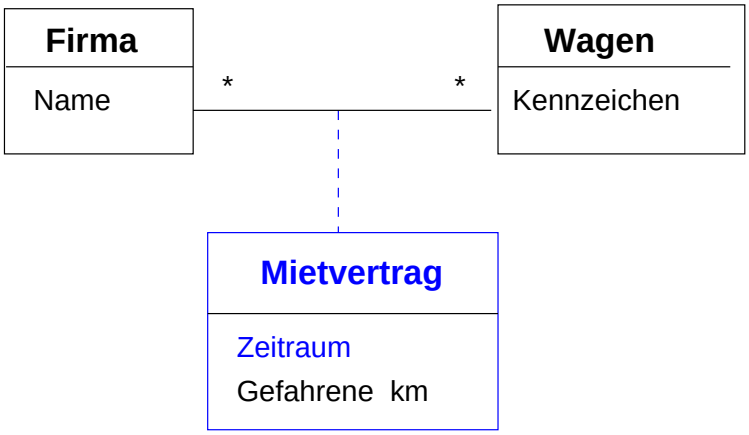
- für ein Objekt mehrere **Fakten, Vorgänge, Zustände** über **Zeitraum** dokumentieren
- einfache Assoziation, nur erweitern
- Zeitliche Einschränkung möglich ({`t=k`})
- hier {`t=1`}, weil sich `Ordner` jeweils nur an einem Ort befinden kann

2.9 Muster 9: Gruppe



- mehrere Objekte (zeitweise) **zusammenfassen** (`1..*` oder `*`)
- einfache Assoziation
- Objektverbindungen können auf- und abgebbaut werden

2.10 Muster 10: Gruppenhistorie



- Zugehörigkeit zur Gruppe über **Zeitraum** dokumentieren
- Zuordnung Einzelobjekt/Gruppe wegen **assoziativer Klasse** deutlich sichtbar
- nur Verbindungen hinzufügen

3 Zusammenfassung

- **Muster** beschreiben häufig auftretende Probleme und bewährte Lösungen
- Analysemuster helfen beim Aufstellen von **Analysemodellen**
- allgemeine vs. **anwendungsspezifische** Muster
- Muster in **Katalogen** zusammengefaßt und dokumentiert