Analysemuster (Stichworte)

Udo Kelter

27.10.2009

Zusammenfassung dieses Lehrmoduls

Muster sind Anleitungen, bestimmte Entwurfsprobleme zu lösen. Analysemuster zeigen Lösungen für diverse Aufgaben, die bei der Gestaltung von Analyse-Dokumenten, insb. Analyse-Klassendiagrammen, anfallen.

Vorausgesetzte Lehrmodule:

obligatorisch: – Objektorientierte Modellierung

Stoffumfang in Vorlesungsdoppelstunden: 1.0

Inhaltsverzeichnis

1 Einordnung

Muster: Lösungsdetail ("Trick"), das schon ein paarmal funktioniert hat

- kann man als "Erfahrung" oder "Pragmatik" ansehen
  (Begriff Pragmatik in der Linguistik: die Lehre vom Gebrauch von Sprache)
- wird von erfahrenen Entwicklern oft unbewußt angewandt
- müssen oft noch kreativ ausgestaltet werden
- Begriff stammt aus der Architektur; in der Informatik zuerst für Entwurfsdokumente ("Entwurfsmuster") benutzt

Analysemuster: in der Analysephase einsetzbare Muster

- betreffen hier in erster Linie *Datenmodelle*
  → oft unabhängig davon, ob ER- oder OOA-Klassendiagramme als Notationsform gewählt werden
- teilweise auch OOA-Klassendiagramme mit Operationen
- nicht gemeint: Muster in Anwendungsfalldiagrammen oder anderen Diagrammtypen, die in der Analysephase häufig eingesetzt werden

Beschreibungsschema von Mustern

1. Name des Musters
2. Motivation / Problembeschreibung / Anwendungsbereich
3. Lösungsbeschreibung: oft vage, d.h. läßt noch Raum für (kreative) Ausgestaltung
4. Diskussion: Vor- und Nachteile des Musters

Grobe Klassifizierung von Analysemustern:

- allgemein einsetzbare
- applikationsdomänenspezifische Muster, insb. Muster, die typische Probleme bei der Analyse betrieblicher Informationssysteme adressieren

Zusammenhang zwischen Analyse- und Entwurfsmustern:

- einige Analysemuster (Bsp: Kompositum) kann man als *rücktransformierte* allgemein einsetzbare *Entwurfsmuster* verstehen, insb. allgemein einsetzbare Analysemuster
- viele Entwurfsmuster (Bsp: Fabrikmethode) behandeln Mengenverwaltung, ablauftechnische Probleme u.ä., von denen in der Analyse abstrahiert wird

  → es gibt deutlich weniger Analysemuster als Entwurfsmuster
- viele von einzelnen Autoren vorgeschlagene Analysemuster unterscheiden sich nur hinsichtlich der Zugriffsoperationen und deren Performanz (??!?)

  – Optimierung hat in der Analyse eigentlich nichts verloren
- Anwendungsbereiche mancher Muster überlappen, d.h. auf das gleiche Problem sind mehrere Muster anwendbar

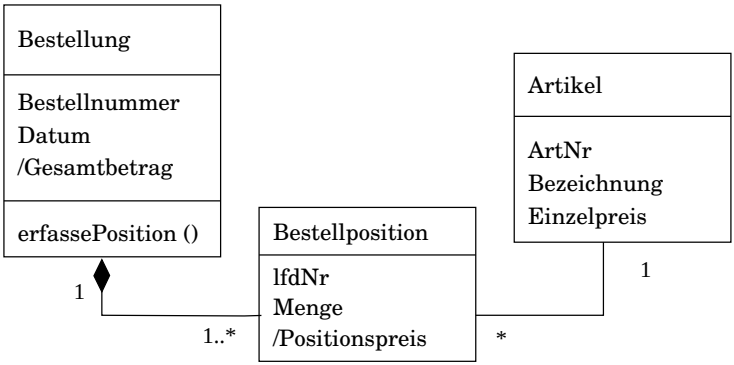
  Entscheidung zwischen konkurrierenden Mustern oft nur mit Vorgriff auf Implementierungsprobleme lösbar
  – bei der Analyse???

  nach Umsetzung in Tabellen manchmal nur marginale Unterschiede

2 Allgemeine Analysemuster

2.1 Analysemuster: Liste

Beispiel: Bestellpositionen in einer Bestellung



weitere Beispiele:
- CD – Stücke darauf
- Kochrezept – Zutaten

Adressiertes Problem: Kollektion einzelner Listenelemente, wobei sowohl die Kollektion Attribute hat als auch die einzelnen Listenelemente. Merkmale:

- die einzelnen Listenelemente kommen nur *innerhalb genau einer* Kollektion vor (zumindest im betrachteten Realweltausschnitt)
  → Komposition
  (Gegenbeispiel: Abteilung – Mitarbeiter, wenn Mitarbeiter auch unabhängig von Abteilungen vorkommen können)
- die Kollektion besteht aus Listenelementen *gleichen Typs*
- einzelne Listenelemente können *erzeugt und gelöscht* werden
- Attribute der Kollektion beschreiben indirekt auch die Listenelemente
- i.d.R. wenigstens 1 Listenelement,
  d.h. die Rolle des Listenelements in der Kompositionsbeziehung hat Kardinalität (= Multiplizität) 1..*
- Kollektion ist oft *geordnet*,
  manchmal nach einem ohnehin vorhandenen Attribut der Listenelemente (Bsp.: CD-Sammlung geordnet nach Titel der CDs),
  manchmal "willkürlich", d.h. es muß ein künstliches Ordnungsmerkmal definiert werden

Lösung: analog zu obigem Beispiel

- je eine Klasse für Kollektion und Listenelemente mit passenden Attributen
- Komposition dazwischen, Kardinalitäten:
  `1..*` für die Rolle des Listenelements,
  `1` für die Rolle der Kollektion
- ggf. künstliches Ordnungsmerkmal bei den Listenelementen

Diskussion: alles klar.

2.2 Analysemuster: Gruppe

Beispiele:

- Abteilungen einer Firma und zug. Angestellte
- Fraktionen im Bundestag und die zu einer Fraktion gehörigen Abgeordneten
- die Lokomotiven, die einer bestimmten Zugverbindung zugeordnet sind

Adressiertes Problem: Es gibt mehrere Gruppen / Mengen, die eine variierende Menge von Mitgliedern haben können

- zeitweise kann die Zahl der Mitglieder auf Null sinken
- jedes Mitglied kann *höchstens in einer Gruppe* Mitglied sein
- ein Mitglied ist ggf. keiner Gruppe zugeordnet

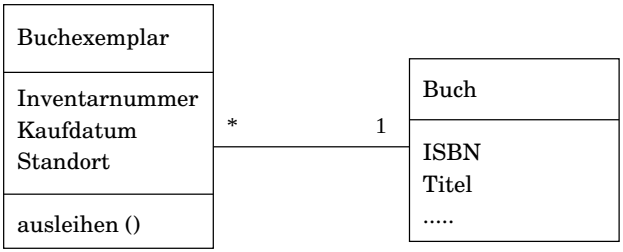
Lösung: einfache Assoziation zwischen Gruppe und Mitglied

Diskussion:
keine Teil-von-Beziehung zwischen Gruppe und Mitglied, weil Mitglieder unabhängig von den Gruppen existieren können

Aufgabe: Worin liegen die Unterschiede zwischen den Mustern Liste und Gruppe?

2.3 Analysemuster: Exemplartyp

Beispiel: Buchexemplar in einem Buchbestand – Buch (-Typ)



weitere Beispiele:
- Exemplare einer Festplatte – Festplattentyp
- Flotte eines Autoverleihers – Wagentyp

Adressiertes Problem: Es sind Daten über eine Kollektion von Objekten zu verwalten;

es gibt Gruppen von Objekten, deren *Eigenschaften i.w. identisch* sind, so daß man sie als *Kopien* voneinander oder *Exemplare* des gleichen Typs ansehen kann. Merkmale:

1. *Eigenschaften des Typs* (Werte seiner Attribute) sind auch Eigenschaften der Exemplare dieses Typs.
2. Die Exemplare haben eine Seriennummer oder Inventarnummer, durch die sie unterscheidbar sind,
   ansonsten nur sehr wenige weitere Eigenschaften, die nicht schon durch den Typ gegeben sind

3. Die *Zahl der Typen* ist deutlich kleiner als die Zahl der Exemplare. Man spart also Erfassungsaufwand, wenn man die Angaben zu den Typen nur einmal erfaßt (vermeidet Redundanz).
4. Man möchte *konsistente Angaben* zu allen Exemplaren eines Typs haben.
5. Oft ist es gewünscht, die Angaben zu den Typen kennen, wenn *noch keine Exemplare* des Typs vorhanden sind.
6. Die Exemplare haben ein Attribut wie ISBN, Produktcode, Bestellnr. o.ä., das den Typ des Exemplars angibt und nach der Entstehung des Exemplars i.d.R. nie wieder verändert wird.

Lösung:

- Modellierung der Exemplare und der Typen durch eigene Klassen (mit jeweils separater Mengenverwaltung!)
- Attribute des Typs kommen bei den Exemplaren nicht noch einmal vor.
- einfache Assoziation zwischen Exemplar und Typbeschreibung, Kardinalitäten:
  1 für die Rolle der Typbeschreibung,
  0..* für die Rolle des Exemplars.
- Name lautet oft “xxx-Typ”, “isOfType”, “xxx-Spezifikation”

Testfrage: wieso kann man dieses isOfType nicht durch Typhierarchien modellieren?

Antwort: Die Zahl der Typen ist nicht statisch, sondern kann jederzeit durch Anlegen neuer Typbeschreibungen erweitert werden

Diskussion: isOfType-Beziehung wird bei Umsetzung in Tabellen i.d.R. zu einem Fremdschlüsselattribut in der Exemplar-Tabelle, das nicht undefiniert sein darf

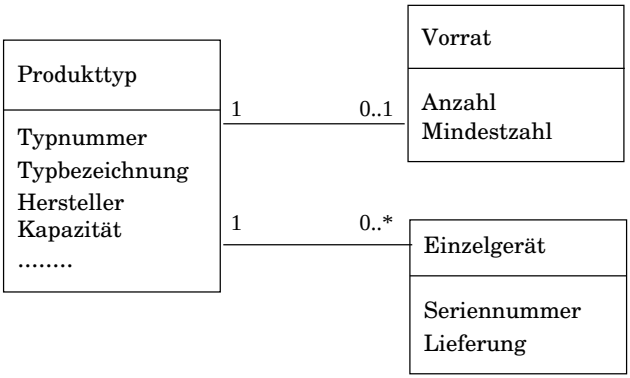
2.3.1 Problem: Koexistenz von anonymen und identifizierbaren Exemplaren

Beispiel: Computerladen, der u.a. Festplatten, CD-Laufwerke und CD-Rohlinge verkauft.

- Festplatten und CD-Laufwerke haben jeweils eine Seriennummer
  → sind einzeln identifizierbare Exemplare
- CD-Rohlinge als anonyme graue Masse auf einem Stapel im Regal; repräsentiert wird der Stapel, nicht jedes einzelne Exemplar
  → von Interesse ist nur die *Anzahl* der vorhandenen Exemplare

Aufgabe: Modellieren Sie die Geräte und Rohlinge.

Lösung:



Merkmale:

- Analysemuster Exemplartyp bei Produkttyp / Einzelgerät
- Ein Produkttyp kann nicht gleichzeitig eine Beziehung zu einem Vorratsobjekt und einem Einzelgeräteobjekt haben
  [kann mit OCL-Ausdruck[1] dargestellt werden]

[1] Die Object Constraint Language (OCL) ist Teil der UML-Standards. Sie ist eine textuelle Sprache, in der diverse Restriktionen formuliert werden können.

Umsetzung in Tabellen:

- Tabellen für Produkttypen und Einzelgeräte wie üblich
- Fremdschlüsselattribut von Einzelgeräten auf Produkttypen für die Beziehung zwischen beiden Klassen
- Alternativen für die Umsetzung der Klasse Vorrat:

  (a) eigene Tabelle mit Fremdschlüsselattribut auf Produkttyp und Spalte für Anzahl
  ... sieht unschön aus: hat sehr wenig Attribute und "fast 1:1"'-Beziehung zu Produkte, daher:

  (b) Attribute von Vorrat zu Tabelle Produkttyp hinzunehmen – Nullwerte bei Produkten mit identifizierbaren Exemplaren

Diskussion:

- man kann nicht gut erkennen, ob ein Produkt anonyme oder identifizierbare Exemplare hat
- Alternative Modellierung:
  2 Subtypen von Produkt für Produkte mit anonymen bzw. identifizierbaren Exemplaren → Hausaufgabe

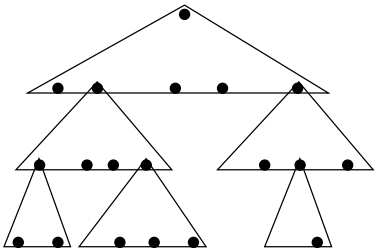
2.4 Analysemuster: Kompositum

engl. Composite, wird auch Stückliste genannt

In der EI II schon als Entwurfsmuster vorgestellt (unter objektbasierte Strukturmuster) - kann fast unverändert auf die Analyse übertragen werden

Beispiele:

1. Stückliste einer großen Maschine (oder Anlage oder PKW oder ...), die aus Bauteilen besteht; jedes Bauteil ist entweder zusammengesetzt aus (kleineren) Bauteilen oder "atomar" (wie Bestandteile interessieren nicht)



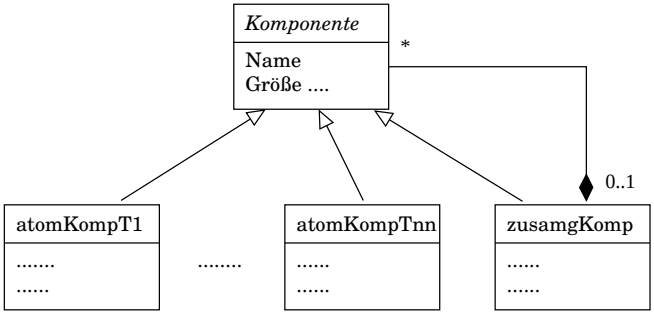
2. eine Graphik, die aus graphischen Objekten besteht, die entweder elementar (Linien, Punkte, ...) oder zusammengesetzt sind
3. Dateibaum in einem MSDOS-Datei-System (nicht UNIX....[ist intelligenter...]): ein Dateibaum besteht aus einem Verzeichnis; ein Verzeichnis enthält elementare Dateien oder Verzeichnisse

Adressiertes Problem: Modellierung einer Teil/Ganzes-Hierarchie; Merkmale:

- unbekannte / ungleichmäßige / zeitlich variierende Schachtelungstiefe
- die Teil-von-Struktur der Objekte ist ein *Baum* (keine Halbordnung), die atomaren Bestandteile bilden die Blätter; es kann Blätter verschiedenen Typs geben.
- die zusammengesetzten Teile müssen oft genauso wie atomare Bestandteile behandelt werden (haben lokale Namen, werden komplett gelöscht oder kopiert, ...)
- alle Knoten des Baums weisen einheitliche Merkmale / Verhaltensweisen auf

  Anmerkung: das einheitliche Verhalten steht beim *Entwurfsmuster* Kompositum im Vordergrund und interessiert hier weniger!

Lösung:



- je eine Klasse für jeden Typ von *atomarem* Bestandteil (also Blätter der Teil-von-Struktur)
- eine Klasse für die *zusammengesetzten* Bestandteilen (also die inneren Knoten der Teil-von-Struktur)

  typische Namen: Verzeichnis, Block, Baugruppe, ...
- eine abstrakte Klasse, die die *gemeinsamen Eigenschaften* und einheitliche Operationen von atomaren und zusammengesetzten Teilen modelliert

  als Generalisierung aller vorstehenden Klassen

  typische Namen: Datei, Komponente, Bauteil, ...
- Kompositionsbeziehung von zusammengesetzten Bestandteilen nach Generalisierungsklasse

  Alle Komponenten außer der Wurzel sind Teil von 1 anderer Komponente.

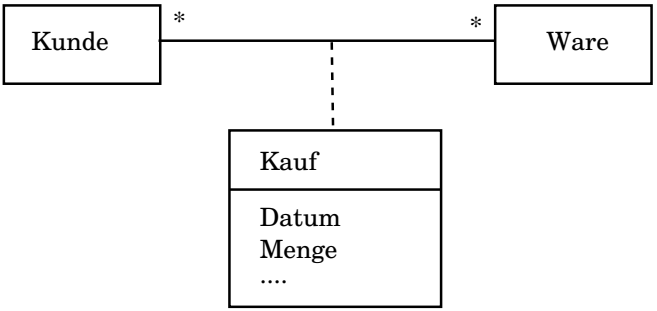
Diskussion:

- kann beliebige Schachtelungen abbilden
- jeder zusammengesetzte Bestandteile kann aus *beliebigen* Teilen zusammengesetzt werden – keine Kontrolle / Einschränkungen auf Typ-Ebene vorhanden

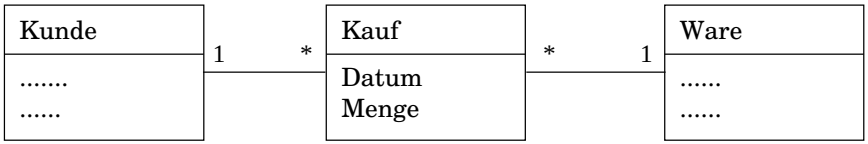
  falls Einschränkungen der Bestandteile gewünscht:
  weitere Arten von zusammengesetzten Knoten mit reduzierter Auswahl an Bestandteilen → unübersichtlich, Muster nicht mehr sinnvoll anwendbar

2.5 Analysemuster: Koordinator

Beispiel: beliebiger mehrstelliger Beziehungstyp ohne Attribute oder mit wenigen Attributen, z.B.



Lösung: anstelle des Beziehungstyps wird eine Klasse verwendet, die pro Rolle eine 2-stellige Assoziation zu der entsprechenden Klasse hat:



Objekte der Koordinatorklasse haben i.w. die Aufgabe, die anderen Objekte zusammenzuführen, zu koordinieren

Diskussion: – Einsatz ist z.T. Geschmacksache

Aufgabe: Setzen Sie die beiden vorstehenden Beispiele nach den Standardmethoden in Tabellen um; nutzen Sie das spezielle Verfahren für 1:n-Beziehungstypen. Vergleichen Sie die Resultate.

Aufgabe: Wie schätzen Sie die Verwendung den Einsatz einer Koordinatorklasse beim Beziehungstyp `schreibt_Diplomarbeit_bei` zwi-

schen den Klassen `Student` und `Dozent` ein?

2.6 Analysemuster: Rolle

Beispiele:

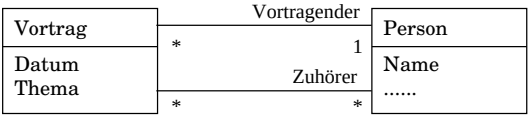
- eine Person ist Zuhörer oder Vortragender in einer Vortragsveranstaltung
- ein Fußballer wird als Tormann, Libero, Stürmer oder Ersatzmann in einem Spiel aufgestellt
- ein Musiker spielt Piano, Baß, Gitarre oder Schlagzeug auf einem Konzert
- ein Künstler ist in einem Film als Schauspieler, Regisseur, Drehbuchautor, Beleuchter usw. beteiligt

Adressiertes Problem: Darstellung der gespielten Rollen;

Unterscheidungsmerkmale der Beispiele:

- die *Menge der Rollen*: ist
  (a) klein und fixiert (Zuhörer vs. Vortragender) oder
  (b) relativ groß (>10) und es können im Laufe der Zeit neue Rollen hinzukommen (Film)
- Zahl der Rollen, die *eine* Person spielen kann:
  (a) maximal eine (entweder Zuhörer oder Vortragender),
  (b) mehrere (Hitchcock war nicht nur Regisseur, sondern trat in vielen Filmen auch als Schauspieler in Nebenrollen auf)
- teilweise treten rollenspezifische Zusatzdaten auf

Lösung für kleine Rollenmengen: pro Rolle ein separater Beziehungstyp



Diskussion: die Beziehungstypen können verschiedene Kardinalitäten haben und werden dementsprechend in Tabellen umgesetzt, also:

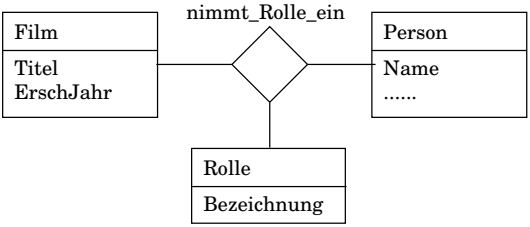
- Vortragender als Fremdschlüsselattribut in der Vortragstabelle
- Zuhörer als separate Verbindungstabelle

Nachteile der Lösung:

- die Bedingung, daß man nicht zugleich Zuhörer und Vortragender in einem Vortrag sein kann, müßte durch einen zusätzlichen OCL-Constraint formuliert werden
- mehrere m:n-Beziehungstypen sind eher lästig (ebenso die daraus bei der Standardumsetzung entstehenden Tabellen)

Lösung für große oder wachsende Rollenmengen:

- neue Klasse *Rolle*, durch die Rollen beschrieben werden können (Bezeichnung einer Rolle als Substantiv)
- dreistelliger Beziehungstyp *nimmt_Rolle_ein* zwischen Person, Rolle und Film / Ereignis / ...



Wenn die Rolle nur das Attribut Name hat, dann kann man dies direkt als Attribut des Beziehungstyps modellieren

Problemvariante: Bei manchen Rollen können Zusatzinformationen relevant sein, Bsp.:

- bei Rollenbezeichnung “ist Schauspieler in” oder “ist deutscher Synchronsprecher von” die gespielte Filmfigur
- bei Rollenbezeichnung “spielt Musik in” das gespielte Musikstück

Lösung: Für Rollen mit Zusatzinformationen können Subtypen von `Rolle` gebildet werden, z.B. `Schauspieler`.

Diskussion: Die Integritätsbedingungen werden komplex.

Aufgabe: Könnte man diese Aufgabenstellung auch mit dem Exemplartyp-Muster lösen? (s. Hausaufgaben)

3 Analysemuster für zeitlich variante Daten

3.1 Analysemuster: Historie

Andere Namen: Gültigkeitsbereich, engl. *effectivity*

Beispiele:

- sich mit der Zeit ändernde Gehaltsgruppe
- sich mit der Zeit ändernde Preise
- sich mit der Zeit ändernde Anschrift einer Person
- sich mit der Zeit ändernde Gruppenzugehörigkeit
- sich mit der Zeit ändernder Arbeitgeber (Lebenslauf)

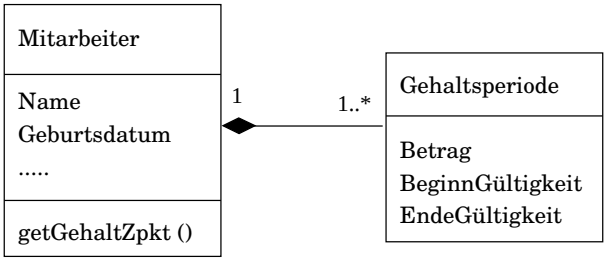
Adressiertes Problem: verwaltet werden soll der *zeitliche Verlauf eines Merkmals* einer Entität;
Merkmal kann durch ein einfaches Attribut oder eine Beziehung (mit Kardinalität 1) zu einer anderen Klasse repräsentiert sein

- die erfaßten Daten müssen es erlauben, folgende Frage zu beantworten: “Welche Ausprägung hatte das Merkmal zum Zeitpunkt

X?"[2]

- in den meisten Fällen darf zu einem Zeitpunkt nur eine Angabe gelten

Lösung für Merkmale, die als Attribut modelliert werden können:
Statt des Attributs werden Komponentobjekte verwendet, die einen Attributwert und einen Gültigkeitszeitraum beinhalten



weitere Merkmale / Benutzung:

- beim letzten Zeitraum ist oft das Ende noch nicht fixiert ("bis auf weiteres") → Nullwert eintragen
- Zeiträume dürfen nicht überlappen – relativ aufwendiger Test
- bei lückenlos aneinandergrenzenden Zeiträumen (häufiger Fall): Attribut "EndeGültigkeit" weglassen, ist identisch mit nächstgrößerem Beginn eines Zeitraums
- für ein zeitlich variantes Attribut gibt es folgende get-/set-Operationen (hier: in der Klasse Mitarbeiter!):

[2]Vorsicht: manchmal muß man Frage beantworten können: "Welche Ausprägung hatte das Merkmal zum Zeitpunkt X gemäß Informationsstand zum Zeitpunkt Y?" Beispiele hierfür sind rückwirkende Änderungen an Gehältern oder Preisen, nachdem bereits Gehaltsabrechnungen oder Rechnungen auf Basis der früher gültigen Preise erstellt worden sind. Dieses Problem ist deutlich komplizierter und wird von diesem Muster nicht behandelt.

– die get-Operation hat als Argument einen Zeitpunkt; kann den Wert undefiniert liefern

– die set-Operation hat neben dem neuen Wert als Argumente Beginn und Ende des Gültigkeitszeitraums und erzeugt ggf. ein neues Komponentenobjekt

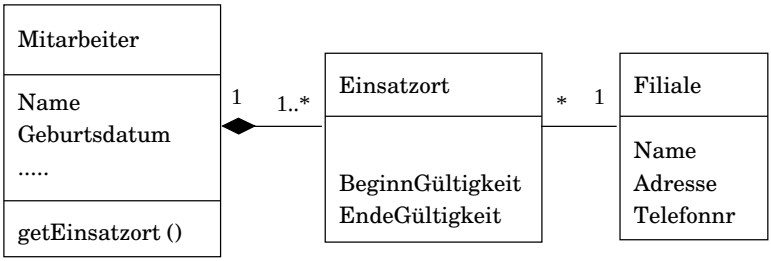
– get-Operation ohne Argument: entweder aktuelles Tagesdatum oder letzter vorhandener Zeitraum

– diverse Varianten der set-Operation je nach Zugriffsverhalten zwecks Optimierung sinnvoll – eigentlich kein Thema für die Analyse

Diskussion:

- die zusätzlichen Parameter bei den get-/set-Operationen sind lästig

Lösung für Merkmale, die als als 1:n-Beziehungstyp modelliert werden können: i.w. wie oben bei Attributen;
in den Komponentobjekte statt des Attributs mit dem aktuellen Wert 1 Beziehungstyp zu der betroffenen anderen Klasse



weitere Merkmale / Benutzung: analog wie bei Attributen

3.2 Analysemuster: wechselnde Rollen

Beispiele:

- sich mit der Zeit ändernde Anschrift einer Person
- sich mit der Zeit ändernde Gruppenzugehörigkeit
- sich mit der Zeit ändernder Arbeitgeber (Lebenslauf)

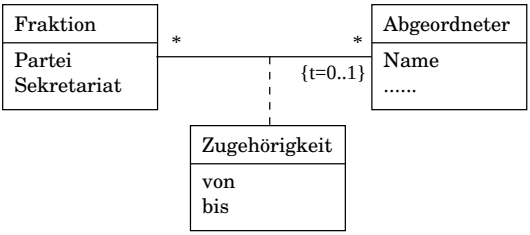
Lösung: kombiniert i.w. das Rollenmuster (Lösung mit Rollenobjekten) mit der Historie: Rollenobjekte werden um Gültigkeitszeitraum erweitert

Diskussion: nicht viel wirklich Neues....

3.3 Analysemuster: Gruppenhistorie

(Gruppe oft kaum unterscheidbar von Rolle ...)

Beispiel:



Adressiertes Problem: zeitlich welchselnde Zugehörigkeiten zu Gruppen

Lösung: Beziehung zwischen Gruppe und Mitgliedern wird attributiert um Attribute, die den Zeitraum der Mitgliedschaft darstellen

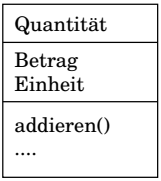
Diskussion / Aufgabe: Vergleichen Sie dieses Muster mit der Lösung des gleichen Problems mit dem Muster Historie für Merkmale, die als 1:n-Beziehungstyp modelliert werden. Vergleichen Sie die ableitbaren Tabellendefinitionen.

4 Weitere Analysemuster

4.1 Analysemuster: Quantität

Adressiertes Problem: es sind Größen- / Mengenangaben, Geldbeträge usw. im verschiedenen Maßeinheiten zu verwalten

Lösung: Statt dimensionsloser Zahlenwerte, deren Einheit sich nur aus dem Kontext ergibt, werden Objekte verwendet, die die Maßeinheit explizit angeben



Diskussion:

- Vorteil ist vor allem, daß passende Rechenoperationen zugeordnet werden können, ggf. incl. passender Konversionen, Rundungen usw.
- die direkte Umsetzung in relationale Tabellen ist meist nicht sinnvoll, die Werte können i.a. beim Einlesen bzw. Schreiben passend konvertiert werden, so daß die Einheit nicht mehr gespeichert werden muß

  → deutliches Indiz, daß es sich meist eher um ein Entwurfsmuster handelt ....

Literatur

[Fo96] Fowler, Martin: Analysis Patterns – Reusable Object Models; Addison Wesley, 1996; ISBN 0-201-89542-0

