

Arbeitsgruppe Software Engineering Prof. Elke Pulvermüller

Universität Osnabrück
Institut für Informatik, Fachbereich Mathematik / Informatik
Raum 31/318, Albrechtstr. 28, D-49069 Osnabrück

elke.pulvermueller@informatik.uni-osnabrueck.de

<http://www.inf.uos.de/se>

Sprechstunde: mittwochs 14 – 15 und n.V.



- 1 Software-Krise und Software Engineering**
- 2 Grundlagen des Software Engineering**
- 3 Projektmanagement**
- 4 Konfigurationsmanagement**
- 5 Software-Modelle**
- 6 Software-Entwicklungsphasen, -prozesse, -vorgehensmodelle**
- 7 Qualität**
- 8 ... Fortgeschrittene Techniken**

5.1 Grundlagen und Modelltypen

5.2 Programmablaufplan

5.3 Struktogramm

5.4 Funktionsbaum

5.5 Strukturierte Analyse (SA)

5.6 EBNF und Syntaxdiagramm

5.7 Entity-Relationship-Modellierung (ERM)

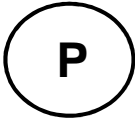


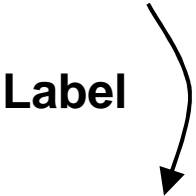
5.5 Strukturierte Analyse: Überblick

- **Structured Analysis (SA): Notation und Methode zur Systembeschreibung**
- **entwickelt von DeMarco und Yourdon (78/79)**
- **Drei zentrale Beschreibungsmittel:**
 - 1) **Datenflußdiagramme (Data Flow Diagrams)**
Daten des Systems und wie diese im System fließen
 - 2) **Datenlexikon (Data Dictionary)**
Struktur der nicht primitiven Daten
 - 3) **Mini-Spezifikation („minispecs“)**
Leistungsbeschreibung einzelner Systemteile
- **Grundgedanke: schrittweise Top-Down-Zerlegung des Systems bis man bei selbsterklärenden Komponenten angekommen ist.**

[DEM78]: DeMarco, T., *Structured Analysis and System Specification*. Yourdon Press, 1978

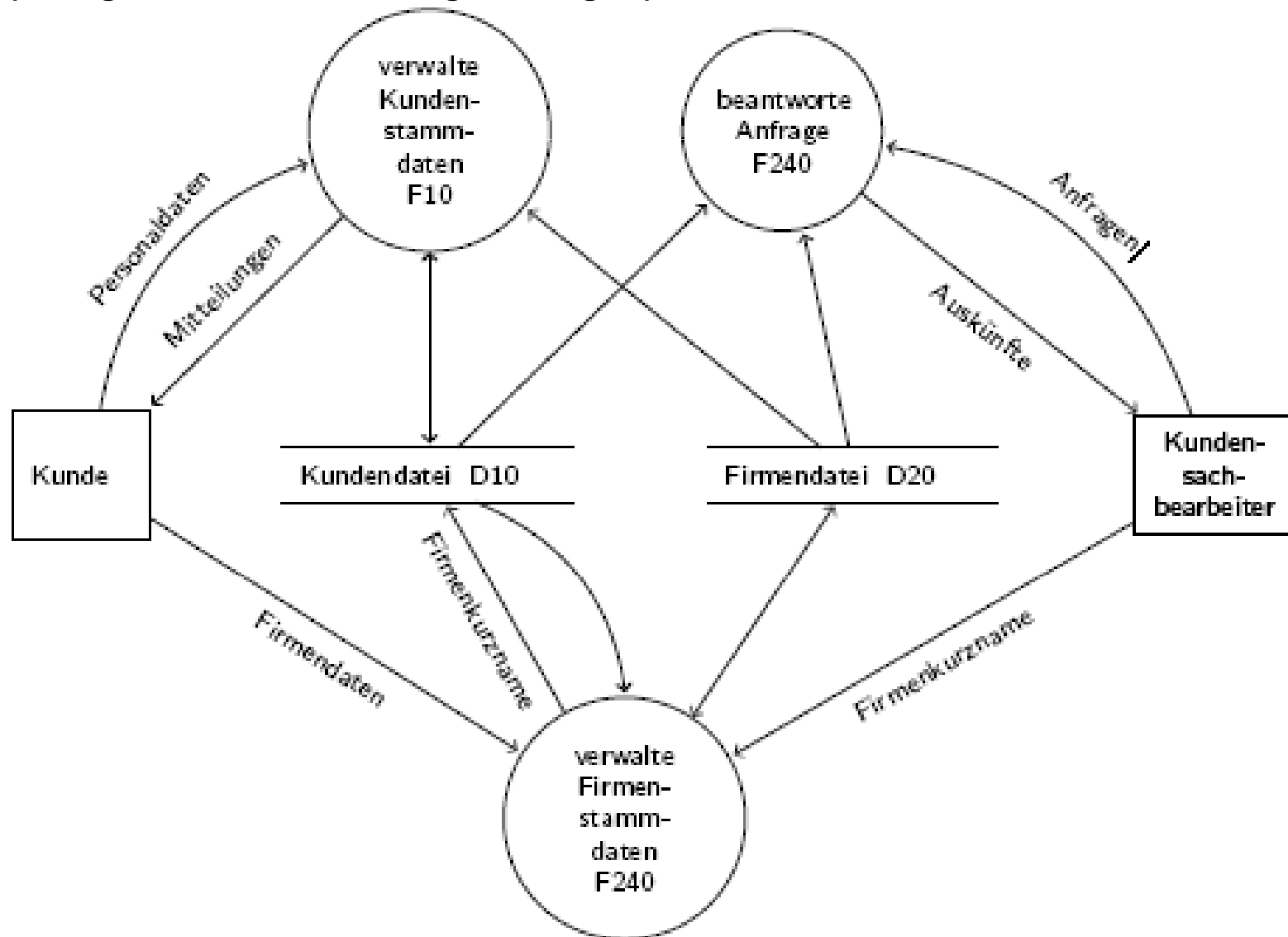
5.5 Strukturierte Analyse: Datenfluss

Datenflussdiagramme: Notation mit 4 Sorten von Objekten

-  **Prozess (Process), Funktion**
-  **Terminator, Schnittstelle**
-  **Datenspeicher (Store)**
-  **Datenfluß (Data Flow)**
- **Vergabe eindeutiger Namen für jedes Element**
- Primitive (nicht unbedingt einfache!) vs.
Nicht-primitive Objekte (hierarchisches Herunterbrechen)

Beispiel: Datenflussdiagramm für eine Kundenverwaltung

(nicht ganz korrekt, s. nachfolgende Regeln)



5.5 Strukturierte Analyse: Datenfluss

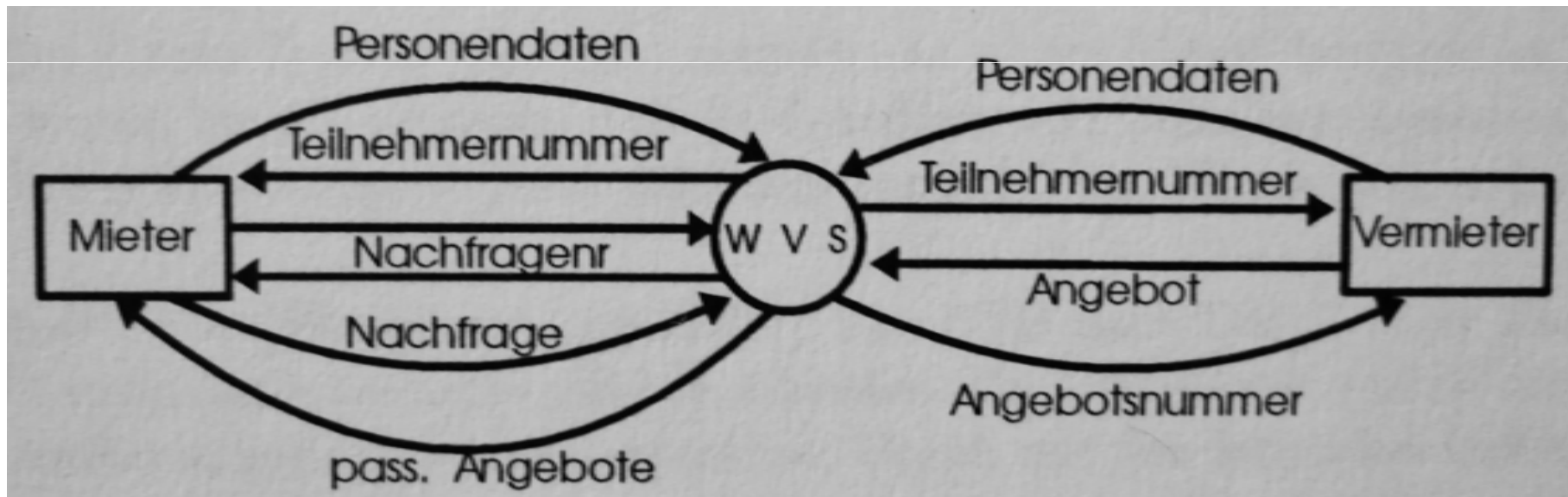
Datenflussdiagramme: Definition

Ein Datenflußdiagramm (DFD) ist ein beschrifteter Graph, dessen Knotenmenge durch Terminatoren, Prozesse und Speicher gegeben ist und dessen Kanten durch Datenflußnamen beschriftet sind, so daß die folgenden Bedingungen gelten:

- Das DFD wird durch einen eindeutigen Namen bezeichnet
- Der Graph ist schlingenfrei
- Ein Datenfluß von oder zu einem Terminator oder Datenspeicher muß unbedingt mit einem Prozeß verbunden sein
- Terminatoren haben mindestens einen ankommenden oder abgehenden Datenfluß; Prozesse und Speicher haben sowohl mindestens einen ankommenden wie auch abgehenden Datenfluß (es gibt keine Datensenken ins oder Datenquellen aus dem Nichts)

Der Graph sollte (muß aber nicht) zusammenhängend sein.

Datenflussdiagramme: Wohnungsvermittlungssystem (Übersichts-DFD)



Terminatoren nur im Übersichts-DFD (höchste Abstraktionsstufe)!

Das Übersichts-DFD enthält keine Speicher!

[Kla2001] H. Klaeren. Skript Software Technik, Universität Tübingen, 2001

5.5 Strukturierte Analyse: Datenstrukturen

Datenlexikon, Datenkatalog (Data Dictionary):

Eine endliche Menge von Datendefinitionen

Verzeichnis der vorkommenden Datenelemente

Zweck: ergänzende Modellierung der Datenstrukturen

- Eine Datendefinition hat die Form:
Einfacher Datenfluß- oder Speichernamen = Datenausdruck
- Datenausdruck wird induktiv konstruiert (semantisch formalisiert):
 - primitiver Datenfluß oder Speichernamen
 - Datenausdruck, der sich durch die Operationen definiert durch ...:

	=
Produkt (Sequenzbildung, „und“):	+
Iteration (Wiederholung):	{ ... }
Wiederholung von M bis N:	M{ ... }N
Selektion (Auswahl, „entweder...oder...“):	[... ]
Option:	(...)
Kommentar:	* ... *

...über gültigen Datenausdrücken bildet

Beispiel:

```
Kundendatei    = {Kundeneintrag}
Kundeneintrag = Kunden-Nr. + Name + Adresse
               + (Geburtsdatum) + (Funktion) + Umsatz
Name           = Anrede + (Titel) + Vorname + Nachname
Adresse        = [Strasse + Haus-Nr. | Postfachnummer]
               + (L\"anderkennzeichen) + PLZ + Ort
               + (Telefon) + (Fax)
```

5.5 Strukturierte Analyse: Funktionsdetails

Beschreibung der Semantik von Prozessen: Minispecs

- Weniger formalisiert als die Semantik der Datenbeschreibungen
- Beliebig gestalteter Text, der die Aufgaben des Prozesses beschreibt
- Einsatz von z.B.
 - Entscheidungstabellen,
 - Pseudocode,
 - structured English mit Namensmarkierungen
(→ automatische Konsistenzprüfung)
 - andere halbformale Mittel
- Beispiel: MSP Eis verkaufen
 Aufgrund des #Kundenwunsch wird eine #Tüte mit #Eis
 zusammengestellt.

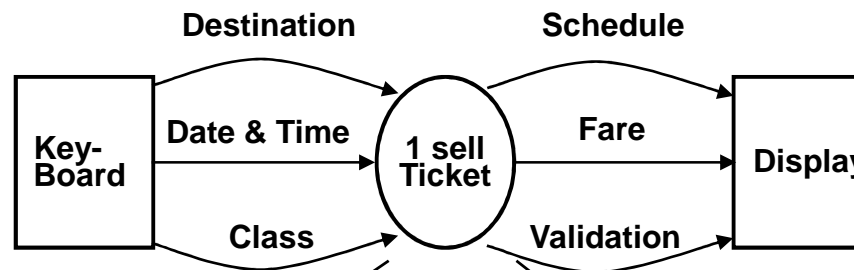
5.5 Strukturierte Analyse: Verfeinerung

Verfeinerungs-DFD ist ein DFD

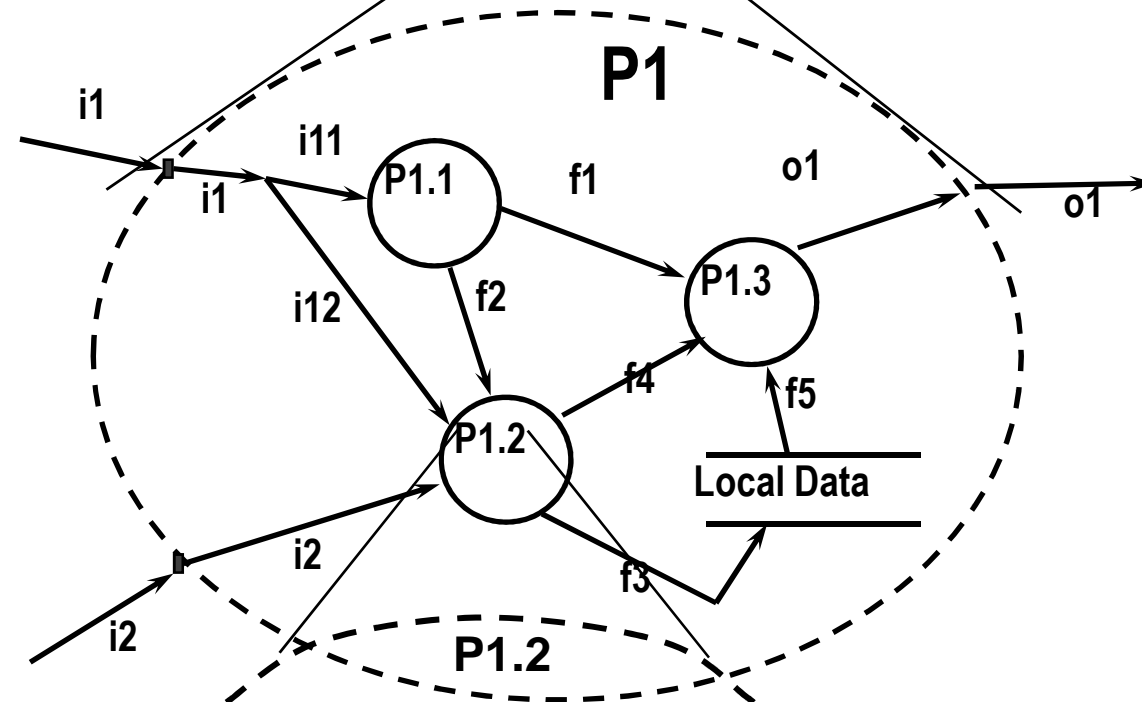
- Mit einer ausgezeichneten Menge sogenannter Anfangs- und Endknoten ohne Beschriftung
- Mit zusätzlichen Datenflüssen von Anfangsknoten zu Prozessknoten und von Prozessknoten zu Endknoten
- Ohne Terminatoren
- Von jedem Anfangsknoten gibt es genau einen abgehenden Datenfluß (=ankommende Datenflüsse des Verfeinerungs-DFD); zu jedem Endknoten gibt es genau einen ankommenden Datenfluß (= abgehende Datenflüsse des Verfeinerungs-DFD)
Bemerkung: in der graphischen Darstellung werden die Anfangs- und Endknoten gar nicht eingezeichnet

Datenflussdiagramme: Schachtelung der Diagramme (Verfeinerung)

Top-level DFD

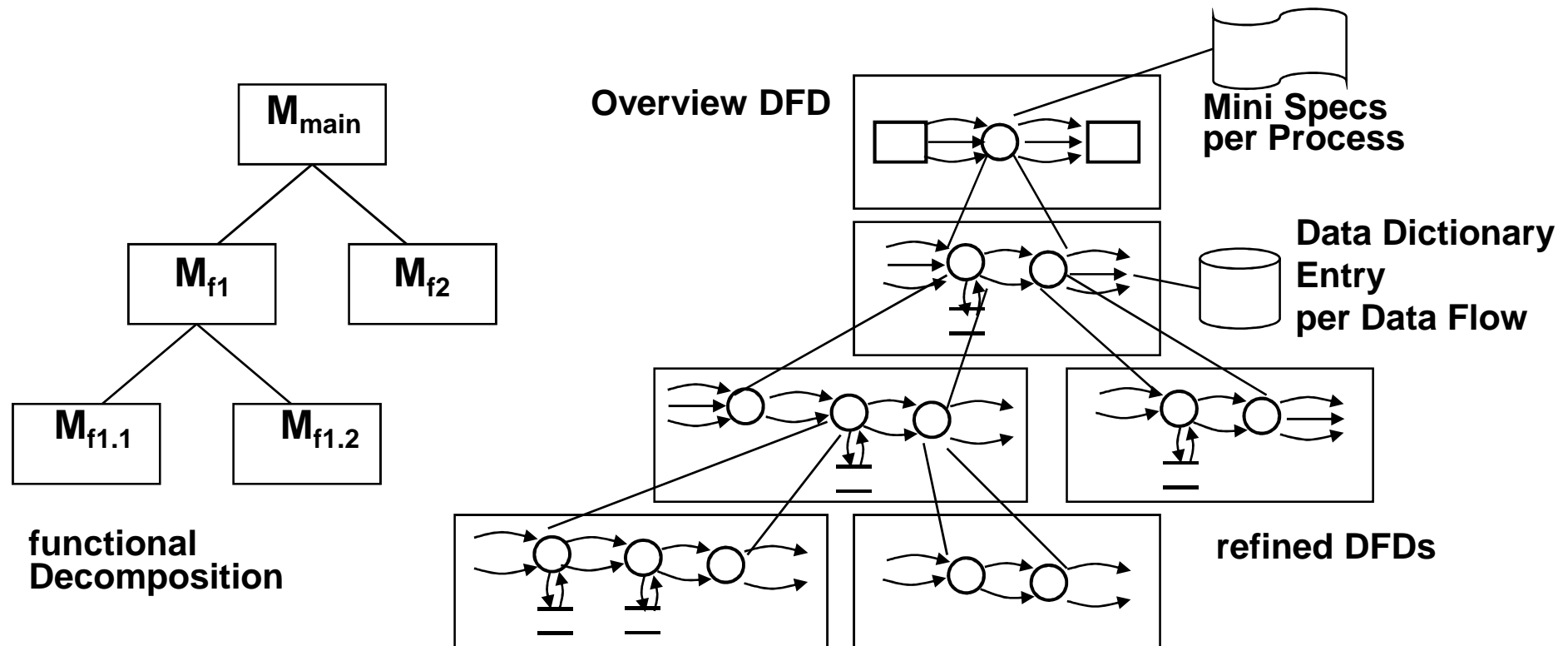


Verfeinertes DFD
(2. Ebene)



Verfeinertes DFD
(3. Ebene)

Datenflussdiagramme: Schachtelung der Diagramme (Verfeinerung)



5.5 Strukturierte Analyse: Verfeinerung

Datenflussdiagramme: Verfeinerung eines DFD

- ist eine Menge von DFDs, wobei zu jedem Prozessknoten des DFD höchstens ein Verfeinerungs-DFD existiert, welches den gleichen Namen trägt wie der entsprechende Prozessknoten
- Numerierungskonvention für eine eindeutige Bezeichnung möglich (Numerierung sagt dann aber nichts über die Reihenfolgen!)
- Einfache Konsistenz: ankommende und abgehende Datenflüsse stimmen im Prozessknoten und seiner Verfeinerung überein (keine Datenverfeinerung!)
- Abgeleitete Konsistenz: ankommende Datenflüsse finden sich im Verfeinerungs-DFD mit seinen Datenkomponenten wieder (für nicht-primitive Daten); dual für abgehende Datenflüsse
Achtung:
beim Produkt (+) müssen in der Verfeinerung nicht alle Produktkomponenten in der Verfeinerung von einem Prozess empfangen werden; bei der Selektion ([...|...]) müssen alle Komponenten von mind. einem Prozess des Verfeinerungs-DFD empfangen werden

5.5 Strukturierte Analyse: Gesamtmodell

SA-Modell (strukturierte Spezifikation):

- Ein SA-Modell besteht aus einer Menge von DFDs, einem Datenlexikon, einer Menge von Minispezifikationen, wobei gilt:
- Es gibt genau ein Übersichts-DFD.
- Für jeden Prozess in den DFDs gibt es eine Minispezifikation.
- Datenflüsse von / zu Datenspeichern sind entweder unbeschriftet (implizit wird ein Datensatz angenommen) oder mit einer Komponente des Speicher-Datentyps beschriftet.
- Für jeden Prozessknoten eines Verfeinerungs-DFD gilt entweder die einfache oder die abgeleitete Konsistenz für ein- und ausgehende Datenflüsse.

5.5 Strukturierte Analyse: Anwendungsvorgehen

Anwendung von SA:

- **Allgemein:**
 - Primär: Datensicht, sekundär: Prozesse**
 - Kurze, aussagekräftige und eindeutige Namensvergabe**
 - Vermeiden überflüssiger Sequentialisierung**
 - Initialisierung, Terminierung und triviale Fehlerfälle besser ignorieren**
 - Falls nötig: von vorne anfangen!**
- **Vorgehen:**
 - 1. Bestimmung der Schnittstellen zur Umwelt (wesentliche Datenflüsse)**
 - 2. Bildung von Informationsgruppen, Datenspeicher und Datenstrukturen, die von mehreren Prozessen benötigt werden, Hinzufügen aller Zugriffe**
 - 3. Datenflüsse und dann Knoten benennen und numerieren**
 - 4. Minispecs für die Knoten schreiben, ein- und abgehende Datenflüsse erwähnen**

5.6 EBNF und Syntaxdiagramm

- **Zwei wichtige Beschreibungsformalismen für Datenstrukturen**
 - **BNF, EBNF (Extended Backus-Naur-Form)**
 - **Syntaxdiagramme**

- **Diese Beschreibungsmittel werden häufig verwendet zur Beschreibung der Syntax von Programmiersprachen**

Bemerkung:

Die Notation für Datenkataloge in der Strukturierten Analyse entspricht der EBNF (erweiterte Backus-Naur-Form).

Datenkataloge können deshalb auch durch Syntaxdiagramme definiert werden.

5.6 EBNF und Syntaxdiagramm

Beispiel:

Deklaration von Java Attributen (vereinfacht) in EBNF:

AttributeDeclaration ::= [Modifier] Type Identifier
["=" Expression]
{"," Identifier
["=" Expression]} ";"

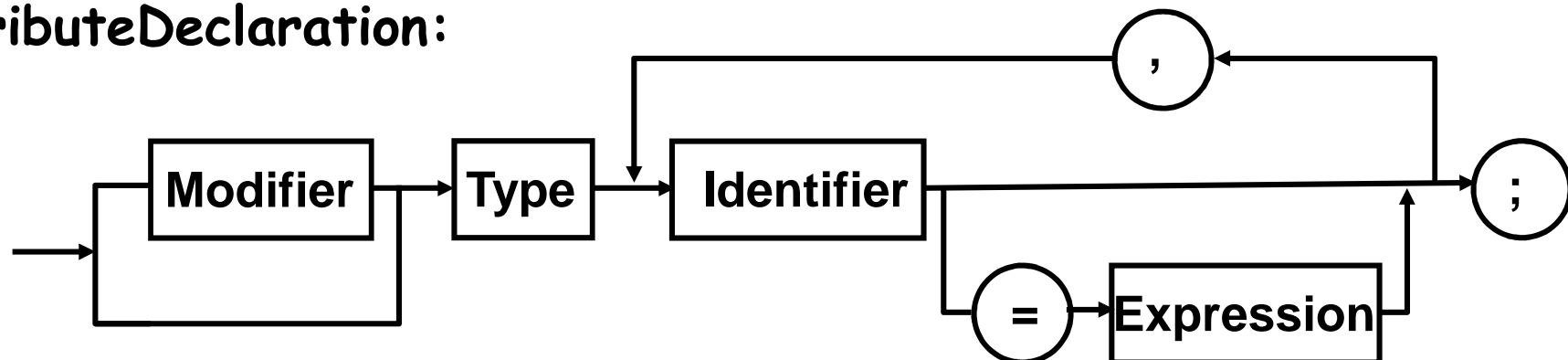
Nicht-Terminal



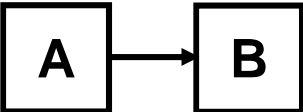
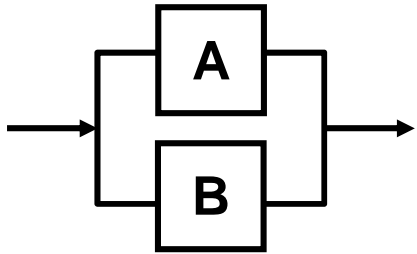
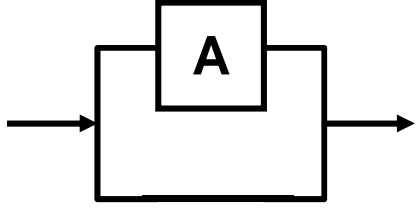
Terminal

Modifier ::= "public" | "private"

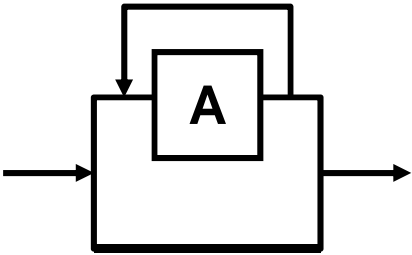
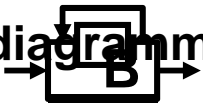
Deklaration von Java Attributen (vereinfacht) als Syntaxdiagramm:

AttributeDeclaration:



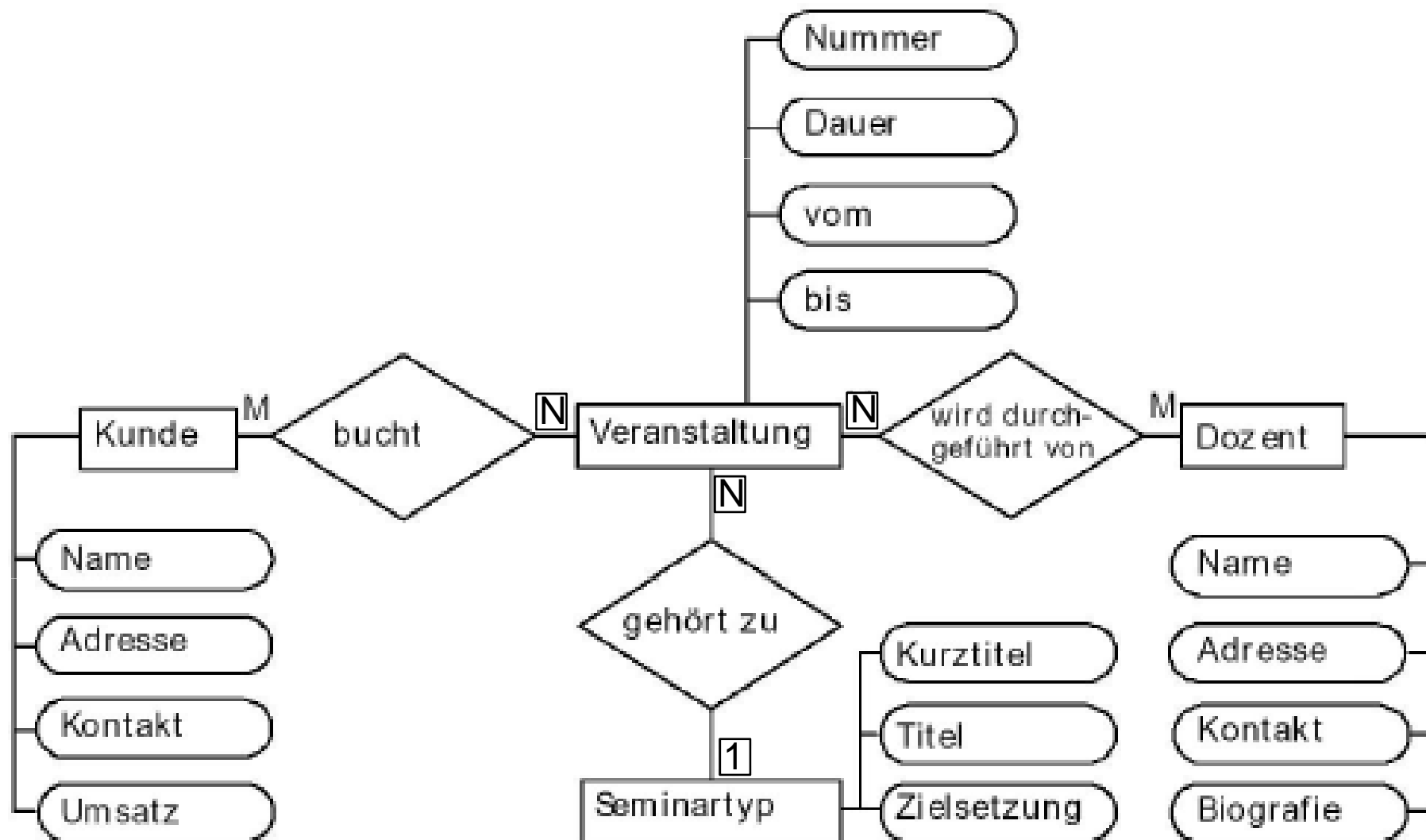
EBNF:		Syntaxdiagramm:
A	Nicht-terminales Symbol	
“X” oder: ‘X’	Terminales Symbol	
A B	Aneinanderreihung	
A B	Alternative	
[A]	Option	

5.6 EBNF und Syntaxdiagramm

EBNF:		Syntaxdiagramm:
$\{A\}$	beliebig häufige Wiederholung (inkl. Null)	
$A ::=$	zu definierendes nicht- terminales Symbol (Platzhalter)	A: Syntaxdiagramm  Name über dem Syntaxdiagramm
$()$	Gruppierung	

5.7 Entity-Relationship-Modellierung (ERM)

- **Einsatz: Datenmodellierung beim Datenbankentwurf (Chen 1976)**
- **Grundidee:**
 - Modellierung der Welt durch Entitäten und Beziehungen
 - Entitäten = Objekte (wohl unterscheidbare Dinge)
 - Zusammenfassung zu Entitätenmengen (Objekttypen)
 - Beziehungen definieren Assoziationen zwischen Objekten
 - Zusammenfassung von Beziehungen zu Beziehungsmengen (Beziehungstypen)
 - Rolle einer Entität in einer Beziehung
 - Attribute von Entitäten und Beziehungen
- **Ein System wird durch die Menge aller Entitäten, Attribute und Beziehungen beschrieben.**



5.7 Entity-Relationship-Modellierung (ERM)

- **Einsatz:** Modellierung von Daten, Datenabhängigkeiten und Datenbeziehungen
Datenstrukturmodellierung (v.a. für Datenbanksysteme)

Notation:

- **Entität (Entity; Objekt)**
eigentlich: Menge von Entitäten



- **Attribut (Attribute; Eigenschaft)**
Schlüsselattribute mit Unterstreichung



- **Beziehung (Relationship)**
(Gleichartige Beziehungen werden zu Beziehungstypen zusammengefasst.)

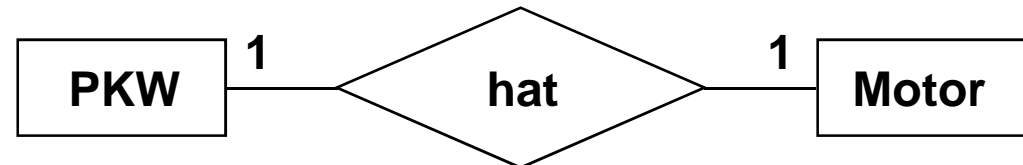


5.7 Entity-Relationship-Modellierung (ERM)

Kardinalitäten der Beziehungen:

1 zu 1

ein PKW hat einen Motor



1 zu n

eine Universität hat mehrere Student(Inn)en immatrikuliert



n zu m

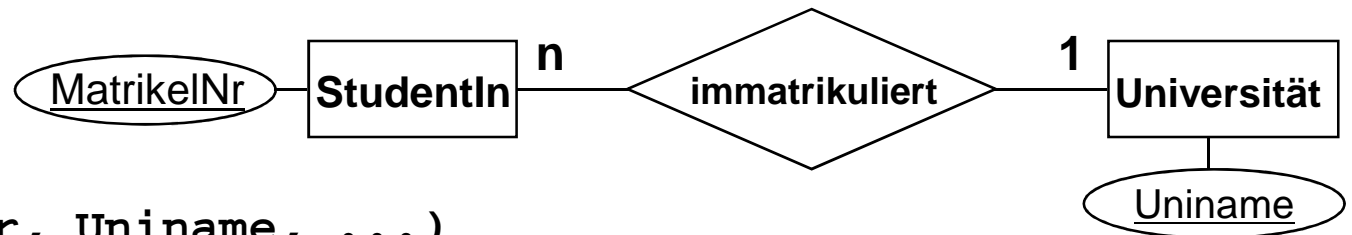
(mehrere) Artikel gehören zu [unterschiedlichen] mehreren Bestellungen

(mehrere) Bestellungen umfassen mehrere Artikel



Übertragung von Beziehungen (ER-Modelle zu Datenbanktabellen):

1 zu n Beziehung



StudentIn (MatrikelNr, Uniname, ...)

Universität (Uniname, ...)

n zu m Beziehung



Artikel (ArtikelNr, ...)

Bestellung (BestellNr, ...)

GehörenZu (ArtikelNr, BestellNr, ...)

5.7 Entity-Relationship-Modellierung (ERM): Beispiel

Beispiel:

Verbale Beschreibung des zu modellierenden Systems

Eine Unternehmung besteht aus mehreren Zweigstellen, die ihrerseits aus mehreren Abteilungen bestehen.

In jeder Abteilung arbeiten mehrere Mitarbeiter.

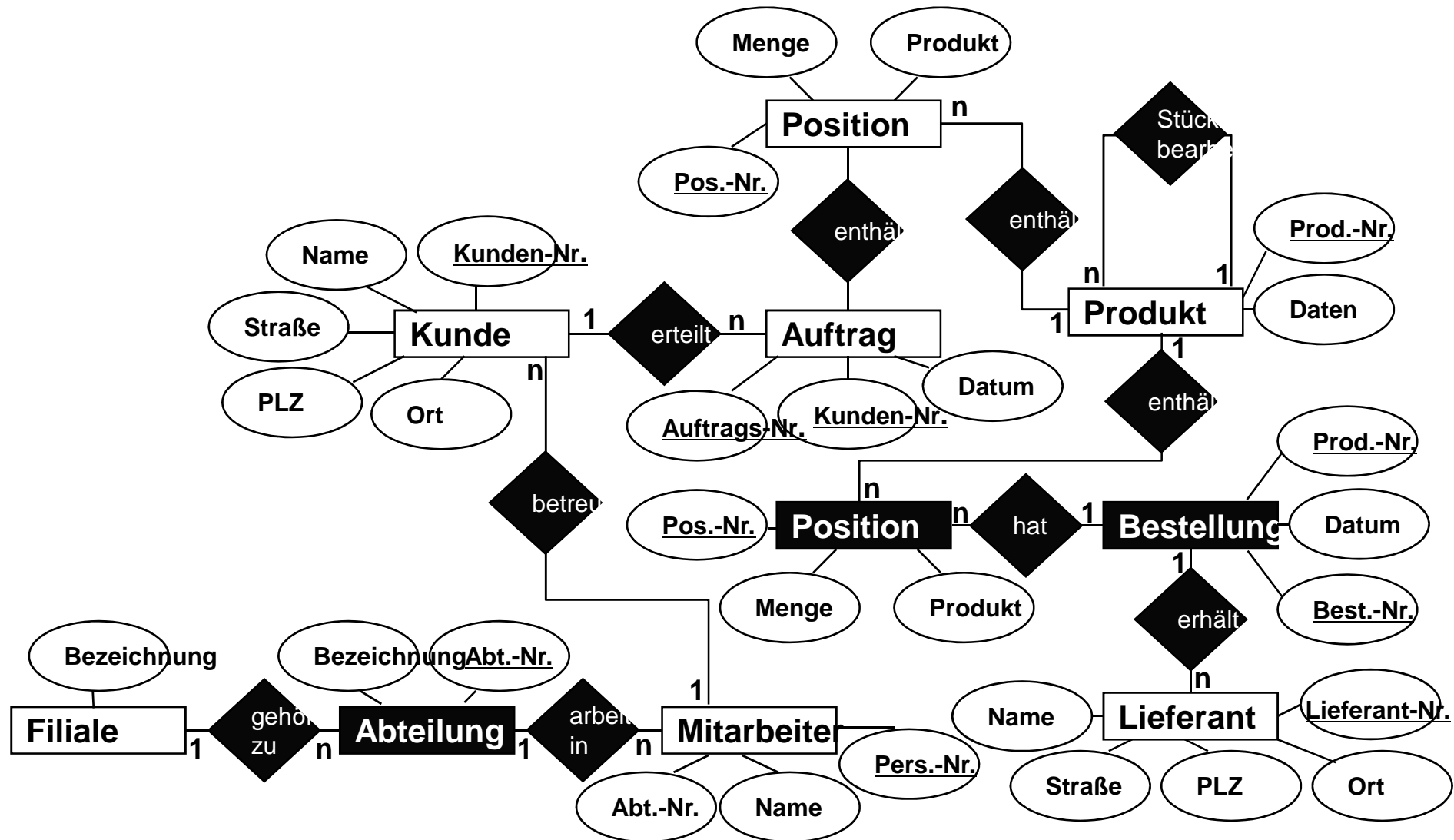
Ein Teil der Mitarbeiter sind Kundenbetreuer, die Kunden zugeordnet sind.

Jeder Kunde hat genau einen Betreuer und ein Kundenbetreuer kann mehrere Kunden betreuen.

Kunden erteilen Aufträge für Produkte.

Die Produkte können aus Subkomponenten bestehen.

Lieferanten versorgen das Unternehmen sowohl mit Produkten als auch mit Produktsubkomponenten.



Entity Relationship Model Extension:

- Ziel: Erhöhung der Semantik, die durch das Modell ausgedrückt werden kann.

- 1) Generalisierung/Spezialisierung
- 2) Überlagerung von Beziehungs- und Entitätstypen

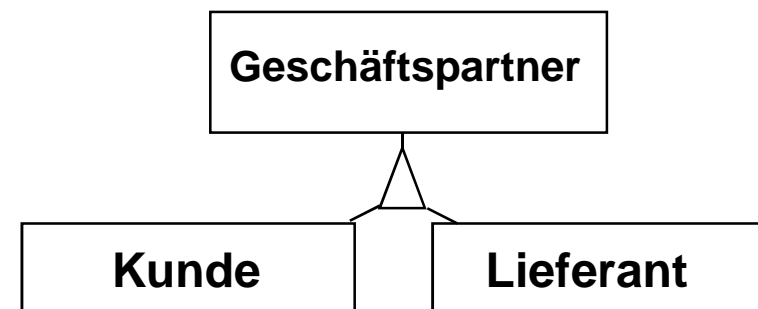
1) Generalisierung/Spezialisierung

Es werden ähnliche Entitätstypen zu einem übergreifenden Entitätstypen zusammengefasst

Beispiele: Kunden und Lieferanten zu Geschäftspartnern,
Autos, Fahrräder und Flugzeuge zu Fahrzeuge

"is a - Beziehung":

Kunde "is a" Geschäftspartner,
graphische Darstellung:



5.7 Entity-Relationship-Modellierung (ERM): Erweiterung

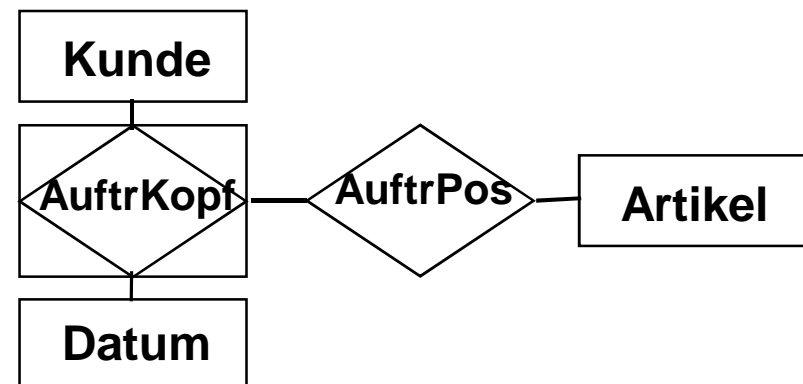
Entity Relationship Model Extension:

2) Überlagerung von Beziehungs- und Entitätstypen

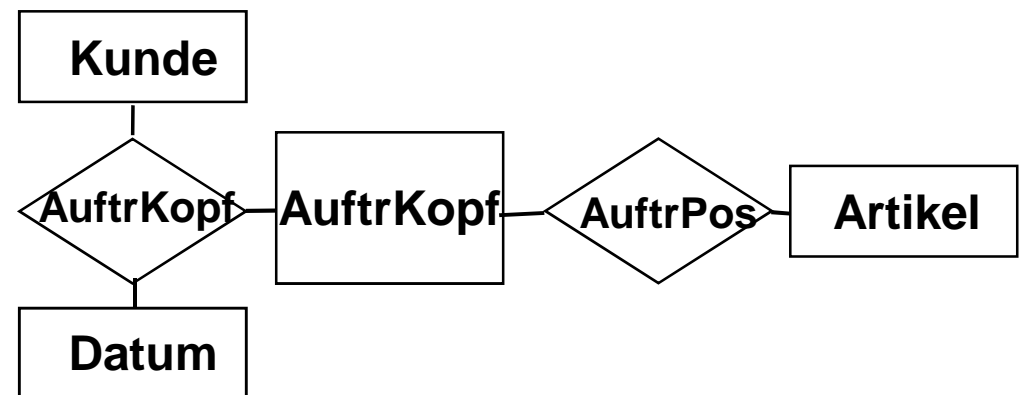
Uminterpretation von Beziehungstypen

Beziehungstyp kann so mit anderem Beziehungstyp in Beziehung gesetzt werden

- Modellierung mit Uminterpretation des Beziehungstyps "*AuftrKopf*" in einen Entitätstypen.



- Auflösung der Überlagerung möglich durch eine Aufspaltung des uminterpretierten Typs in einen Entitäts- und einen Beziehungstyp gleichen Namens, die durch eine (1 :1)-Beziehung miteinander verbunden sind.



5.7 Entity-Relationship-Modellierung (ERM)

Bewertung:

- Übersichtliche, leicht lernbare und formale Beschreibung von Daten in Applikationen
- Graphische und relationale Darstellung
- Direkte Umsetzung in Datenbanksysteme
- Keine Beschreibung der Funktionen, die auf den Daten arbeiten oder der Abläufe
- Keine Beschreibung der Änderungen / Anpassungen in den Daten
- Unterstützt v.a. relationale Datenbanken

Einsatz („Standard“ in der Praxis):

Entwurf und Modellierung von Datenbanken und Informationssystemen

Allg. Bemerkung: Datenstrukturen sind besonders konstant und daher von besonderer Bedeutung (z.B. sehr hilfreich bei Integrationsaufgaben)

Zusammenfassung und Ausblick

- 1 Software-Krise und Software Engineering
- 2 Grundlagen des Software Engineering
- 3 Projektmanagement
- 4 Konfigurationsmanagement
- 5 Software-Modelle
- 6 Software-Entwicklungsphasen, -prozesse, -vorgehensmodelle
- 7 Qualität
- 8 ... Fortgeschrittene Techniken

- 5.1 Grundlagen und Modelltypen
(Modellbegriff, Modellarten/Sichten, Einsatz, Modellvielfalt, Abstraktionsebenen)
- 5.2 Programmablaufplan
- 5.3 Struktogramm
- 5.4 Funktionsbaum
- 5.5 Structured Analysis
- 5.6 EBNF, Syntaxdiagramm
- 5.7 ERM

**bekannte
Modelle bzw.
Modellierungs-
sprachen**

→ Wege im Umgang mit der Software-Krise und Umsetzung der Grundlagen und Prinzipien

Einsatz von Modellen

Weitere bekannte Modelle für verschiedene Sichten

