TECHNISCHE UNIVERSITÄT
KAISERSLAUTERN

o1o1seda o1o1oo

software engineering dependability

Software Entwicklung 2

Strukturierte Analyse 1

Inhalt

- Einführung und Überblick
- Datenfluss-Diagramm
- ER-Diagramm und Data Dictionary
- Das Hierarchiekonzept
- Das Kontextdiagramm
- Verfeinerte Datenflussdiagramme
- DD-Einträge und Datenintegrität
- Mini-Spezifikationen
- Methodik
- Wertung
- Einsatz von CASE-Werkzeugen
- SA im Überblick

© Prof. Dr. Liggesmeyer

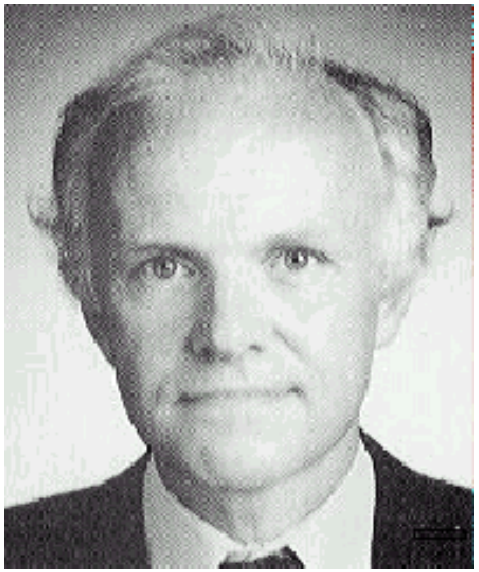
Lernziele

- Den Aufbau von SA beschreiben können
- Die einzelnen Diagramme und Notationen von SA erläutern und anwenden können
- Hierarchische Datenflussdiagramme erzeugen können
- Data-Dictionary-Einträge erzeugen können
- Mini-Spezifikationen erstellen können
- SA-Modelle auf ihre Korrektheit überprüfen können
- Eine verbale Beschreibung in ein SA-Modell umsetzen können

© Prof. Dr. Liggesmeyer

Zur Historie

- Tom DeMarco
  *20.8.1940 in Hazleton,
  Pennsylvania, USA
  Principal, The Atlantic Systems Guild
- Erfinder der Strukturierten Analyse (SA)
- Buch: Structured Analysis and System Specification, 1978
- Bedeutende Beiträge zum Software-Management
- Buch: Controlling Software Projects, 1982
- Buch: Peopleware, zusammen mit T. Lister, 1987
- Buch: The Deadline – A Novel about Project Management, 1997



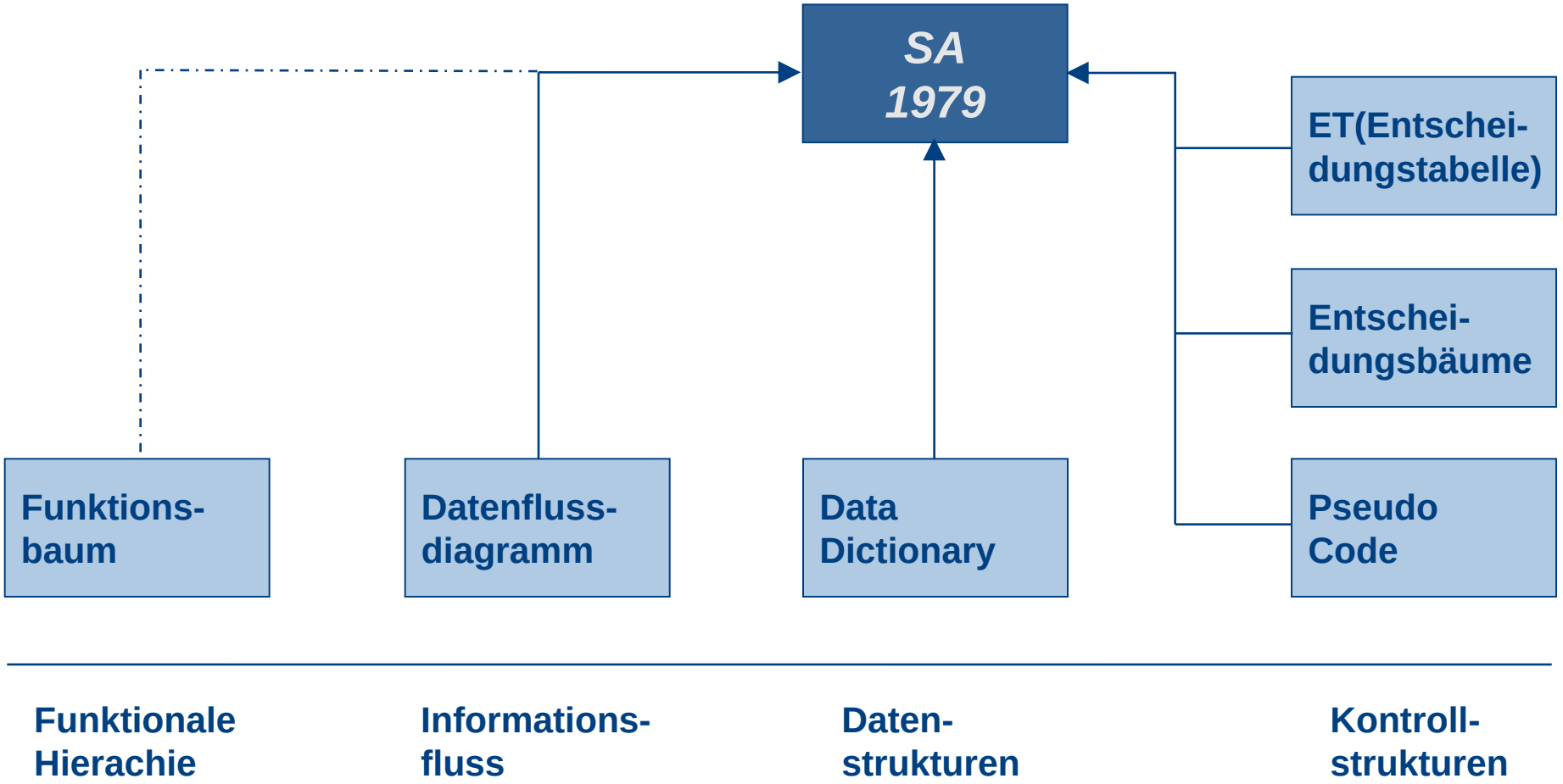
© Prof. Dr. Liggesmeyer

Einleitung und Überblick

- Beschreibt eine Methodenklasse
- Wichtige SA-Varianten
  - Weinberg 1978: »Structured Analysis«
  - Gane/Sarson 1979: »Structured Systems Analysis«
  - McMenamin/Palmer 1984: »Essential Systems Analysis«
  - Yourdon 1989: »Modern Structured Analysis«
- SA nach DeMarco wird hier vorgestellt

© Prof. Dr. Liggesmeyer

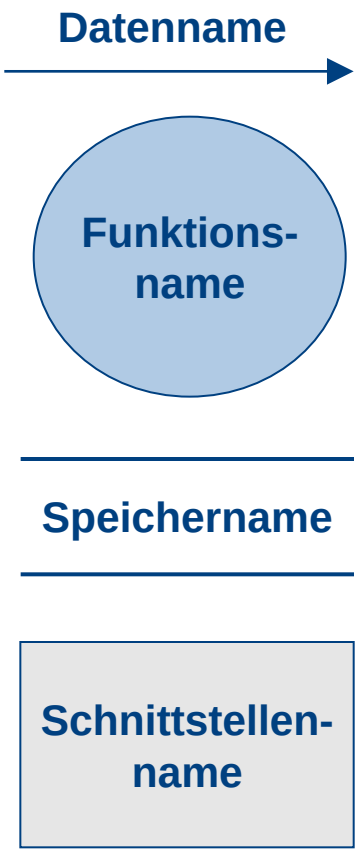
Einleitung und Überblick
Kombination der Basiskonzepte in SA



© Prof. Dr. Liggesmeyer

Datenfluss-Diagramm

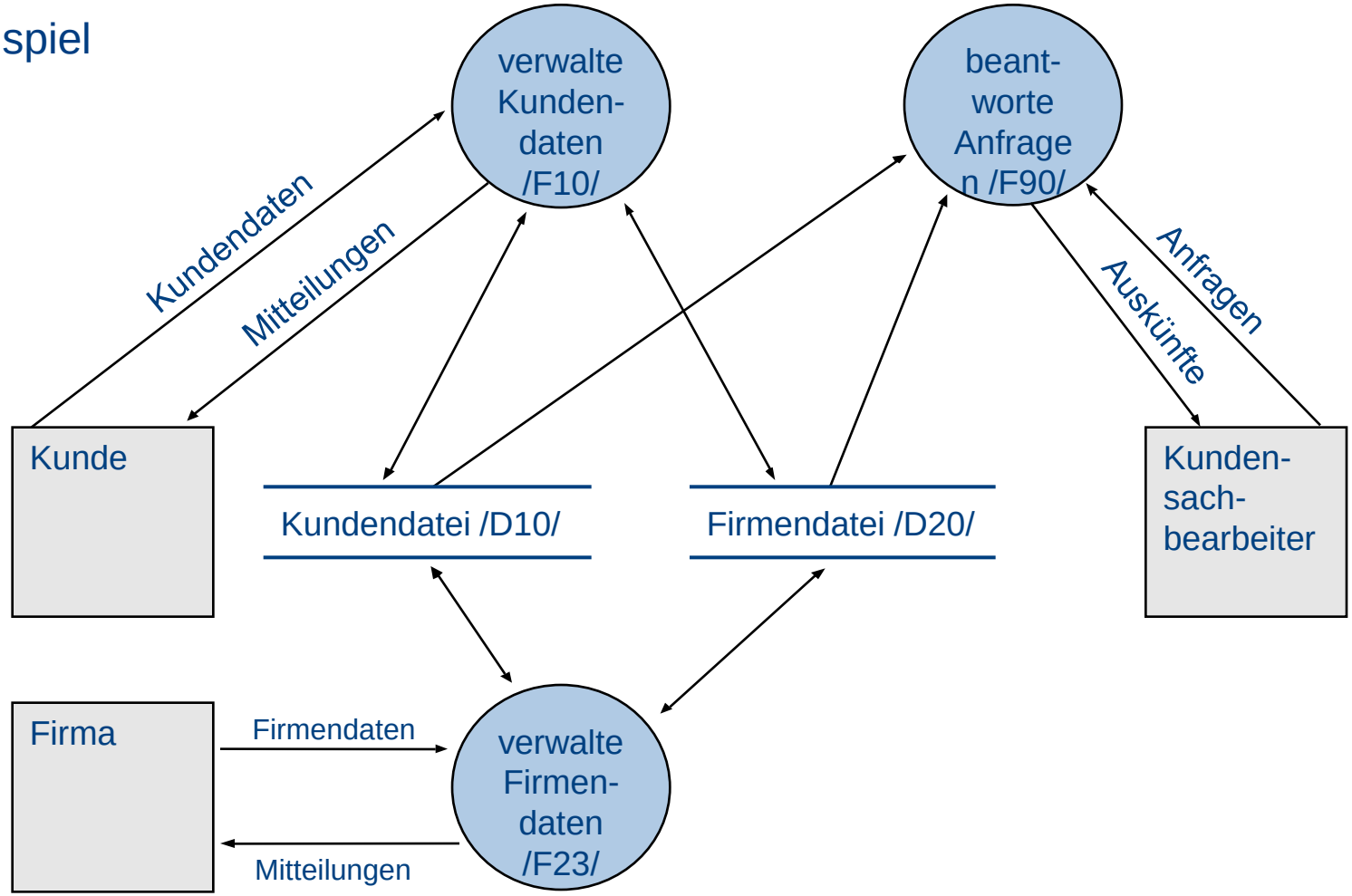
- Notation von DeMarco (1979)



© Prof. Dr. Liggesmeyer

Datenfluss-Diagramm

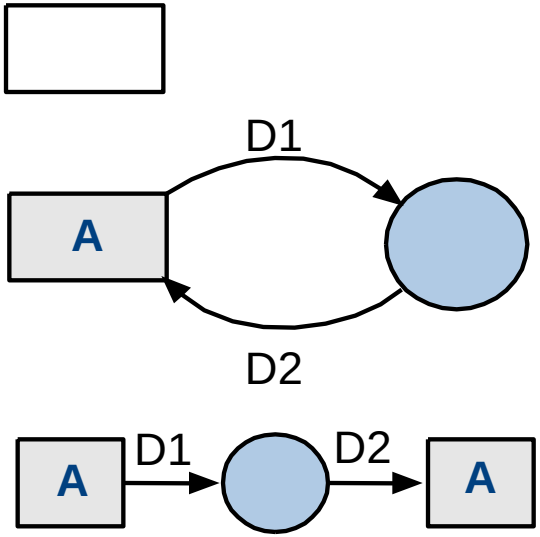
- Beispiel



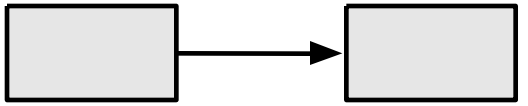
© Prof. Dr. Liggesmeyer

Datenfluss-Diagramm

- Syntaktische Regeln für DFDs
  - Ein DFD enthält mindestens eine Schnittstelle
  - Jede Schnittstelle ist im Allgemeinen nur einmal vorhanden
    - Ausnahme: Wird das DFD unübersichtlich, dann kann eine Schnittstelle mehrfach gezeichnet werden
  - Zwischen Schnittstellen gibt es keine Datenflüsse



Falsch:

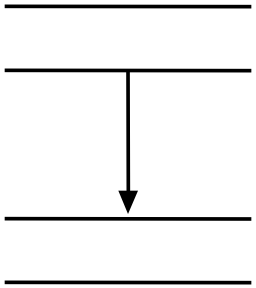


© Prof. Dr. Liggesmeyer

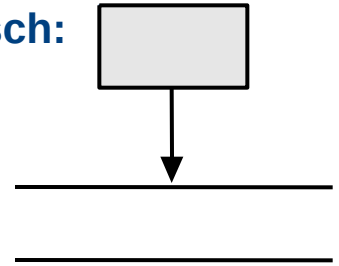
Datenfluss-Diagramm

- Syntaktische Regeln für DFDs
  - Jeder Datenfluss hat einen Namen
    - Ausnahme: Datenflüsse, die zu Speichern führen und keinen Namen haben, beinhalten die gesamten gespeicherten Daten
  - Zwischen Speichern dürfen keine direkten Datenflüsse gezeichnet werden
  - Zwischen Schnittstellen und Speichern dürfen keine direkten Datenflüsse gezeichnet sein

Falsch:



Falsch:



© Prof. Dr. Liggesmeyer

Datenfluss-Diagramm

- Semantische Regeln für DFDs
  - Das DFD beschreibt den Datenfluss, nicht den Kontrollfluss
    - Daher enthält es weder Entscheidungen noch Schleifen
    - Es wird keine Aussage über die Initiierung und Terminierung von Funktionen bzw. Prozessen gemacht
  - Steht eine Schnittstelle für eine Vielzahl von beliebig vielen Instanzen, dann wird sie als eine Schnittstelle dargestellt

© Prof. Dr. Liggesmeyer

Datenfluss-Diagramm

- Semantische Regeln für DFDs
  - Wird das System durch eine eng begrenzte Anzahl von gleichartigen Schnittstellen begrenzt, die sich aber durch unterschiedliche Datenflüsse auszeichnen, dann ist eine getrennte Darstellung sinnvoll
  - Eine Schnittstelle ist so zu wählen, dass sie die ursprüngliche Quelle oder Senke einer Information angibt

Normalfall:

Lieferant

Es gibt 2 unterschiedliche Klassen:

Lieferant 1

Lieferant 2

© Prof. Dr. Liggesmeyer

Datenfluss-Diagramm

- Semantische Regeln für DFDs
  - Bei der Wahl einer Schnittstelle wird von der konkreten Eingabe einer Information in das System über eine Tastatur oder einer Ausgabe auf einem Drucker vollständig abstrahiert
  - Ein Datenflussname besteht aus einem Substantiv oder einem Adjektiv und einem Substantiv
    - Datenflussnamen enthalten niemals Verben
  - Die Datenflussnamen sind so zu wählen, dass sie nicht nur die Daten, die fließen, beschreiben, sondern etwas darüber aussagen, was über die Daten bekannt ist
    - Beispiel: Rechnungs-Nr, gültige Rechnungs-Nr.

© Prof. Dr. Liggesmeyer

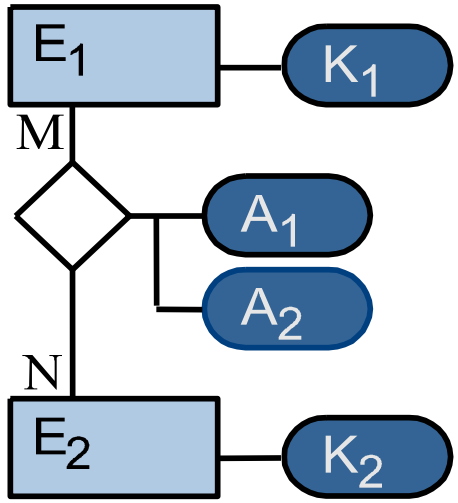
Datenfluss-Diagramm

- Semantische Regeln für DFDs
  - Seichte Namen wie „Daten“, „Informationen“ vermeiden
  - Ein Funktions- bzw. Prozessname besteht aus einem einzigen starken Aktions-Verb gefolgt von einem einzigen konkreten Objekt oder einem konkreten Substantiv gefolgt von einem starken Aktions-Verb
  - Funktionsnamen repräsentieren Aktionen
  - Seichte Namen wie „verarbeite“, „bediene“ vermeiden

© Prof. Dr. Liggesmeyer

ER-Diagramm und Data Dictionary

- ER-Diagramme sind aus dem Entwurf relationaler Datenbanken bekannt



- Das Data Dictionary basiert auf einer geringfügig modifizierten EBNF (Extended Backus-Naur-Form), die aus dem Compilerbau bekannt ist, z.B.

  Rechnung = [erste Rechnung | korrigierte Rechnung | stornierte Rechnung]

© Prof. Dr. Liggesmeyer

Das Hierarchiekonzept

- Hierarchisierung zur Steigerung der Übersichtlichkeit
  - Datenflussdiagramme (DFD) werden hierarchisch verfeinert
  - Kontextdiagramm steht als abstraktes DFD in der obersten Hierarchieebene
  - Jeder Prozess des Kontextdiagramms wird in untergeordneten Diagrammen verfeinert
  - Ist keine Verfeinerung mehr möglich, wird eine textuelle Mini-Spezifikation erstellt

© Prof. Dr. Liggesmeyer

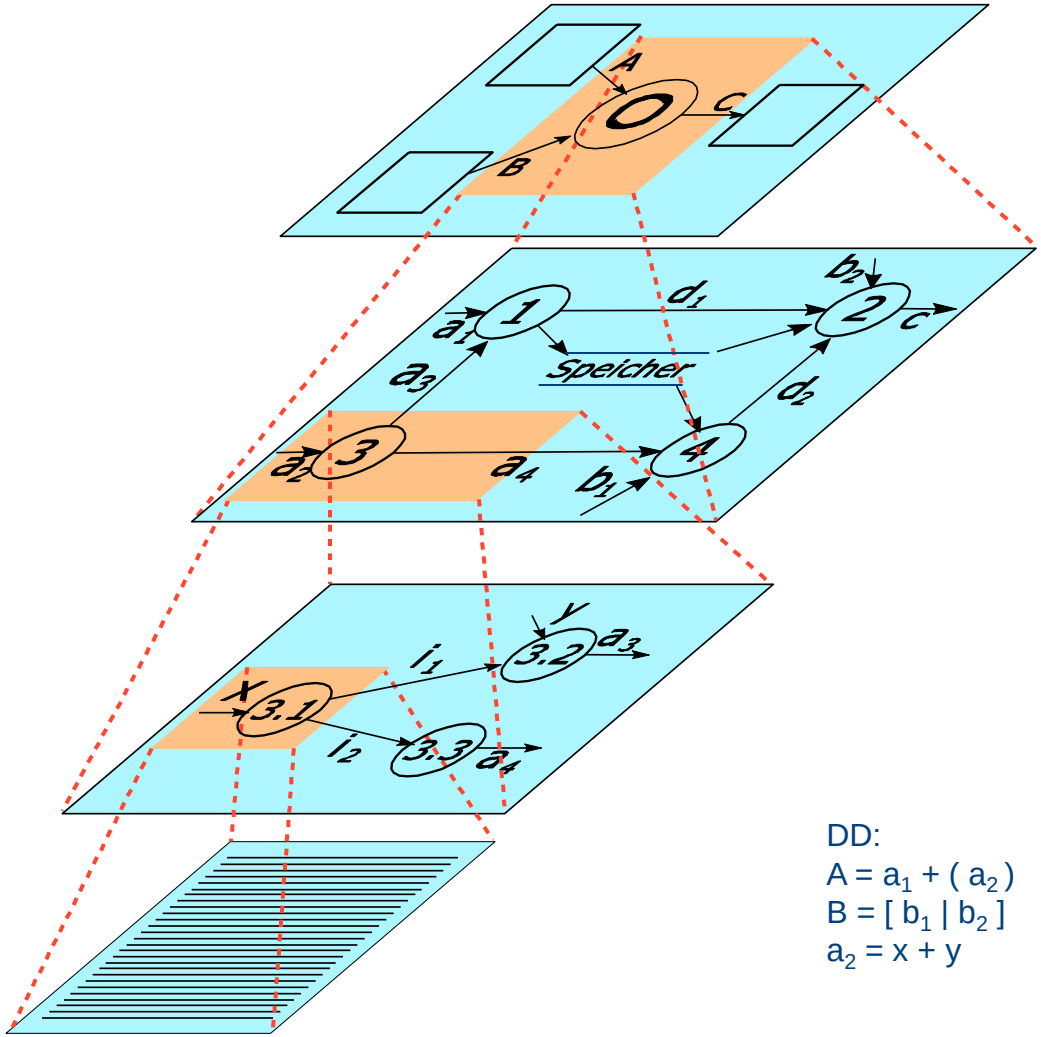
Das Hierarchiekonzept

Kontextdiagramm

Diagramm 0

Diagramm 3

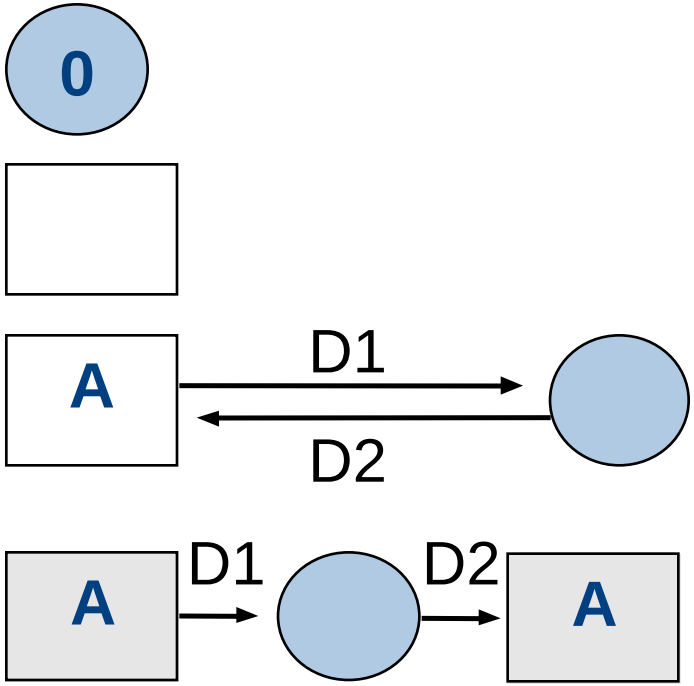
MiniSpec 3.1



DD:
$A = a_1 + (a_2)$
$B = [b_1 | b_2]$
$a_2 = x + y$

Das Kontextdiagramm

- Beschreibt die Schnittstellen des Systems zur Umwelt
- Syntaktische Regeln
  - Enthält nur einen Prozess
    - Repräsentiert das Gesamtsystem
    - Erhält Nummer 0
  - Enthält mindestens eine Schnittstelle
  - Zwischen Schnittstellen keine Datenflüsse
  - Enthält keinen Speicher
  - Jede Schnittstelle ist im Allgemeinen nur einmal vorhanden
    Ausnahme: Übersichtlichkeit



© Prof. Dr. Liggesmeyer

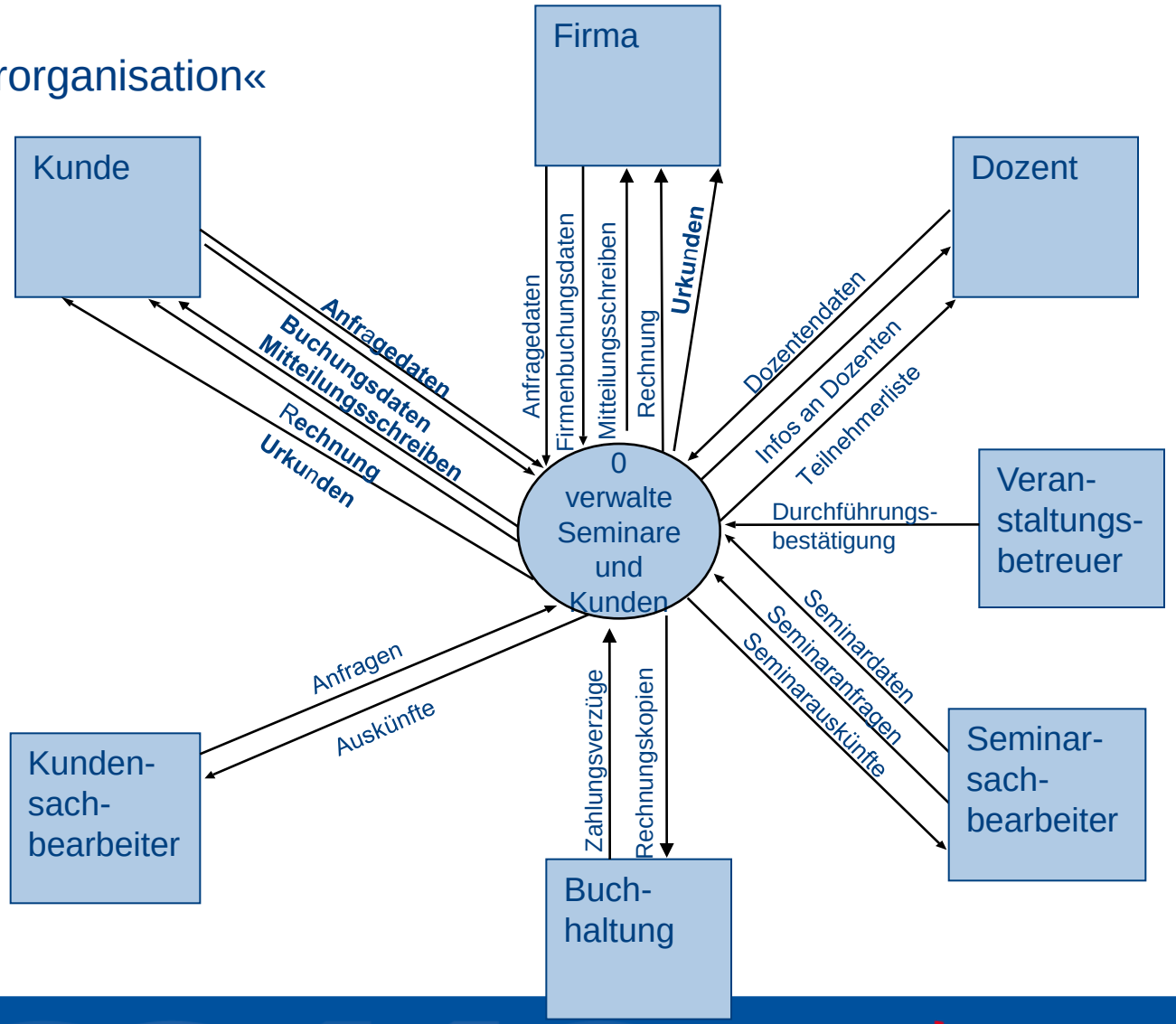
Das Kontextdiagramm

- Semantische Regeln
  - Beschreibt Anwendungsbereich des Systems
  - Zeigt Datenflüsse über die Systemgrenzen
  - Kann es von derselben Schnittstelle mehrere Instanzen geben, wird sie einmal dargestellt
  - Gibt es wenige gleichartige Schnittstellen mit unterschiedlichen Datenflüssen, dann getrennte Darstellung
  - Schnittstelle muss ursprüngliche Quelle oder Senke einer Information angeben
  - Wahl einer Schnittstelle abstrahiert von der konkreten Eingabe oder Ausgabe
    - Tastatur und Drucker im Allgemeinen keine Schnittstelle

© Prof. Dr. Liggesmeyer

Das Kontextdiagramm

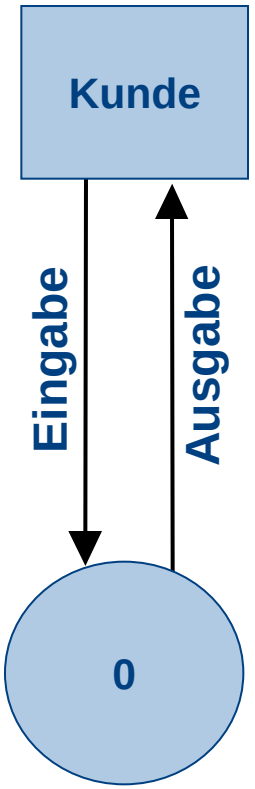
- Beispiel »Seminarorganisation«



© Prof. Dr. Liggesmeyer

Das Kontextdiagramm
Abstraktion

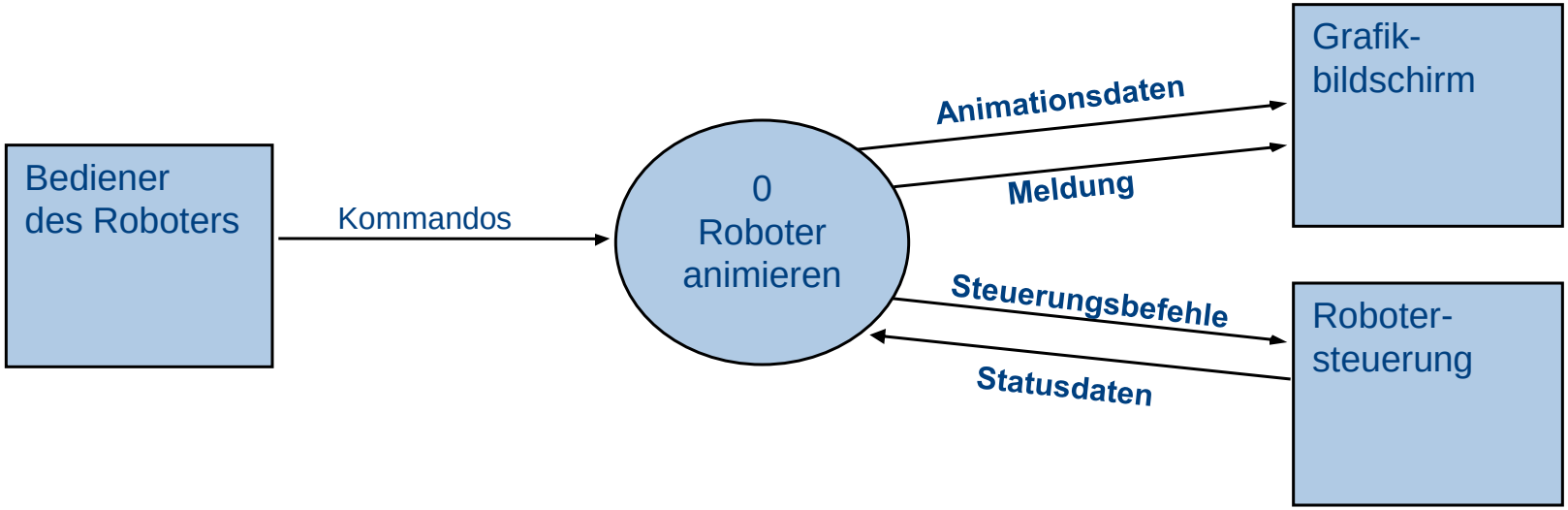
- Datenflüsse auf angemessenem Abstraktionsniveau beschreiben
- Zu detailliert: unübersichtlich, überladen
- Zu abstrakt: nichtssagend
- Richtschnur
  - Anhand des Kontextdiagramms müssen für einen Außenstehenden die wesentlichen Informationen über die Umwelt erkennbar sein
  - Namensgebung muss problembezogen sein
  - Alle Datenflüsse auf demselben Abstraktionsniveau



© Prof. Dr. Liggesmeyer

Das Kontextdiagramm

- Beispiel »Animation eines Roboter«



- Da die Animation im Mittelpunkt steht, ist Grafikbildschirm eine eigene Schnittstelle
- Modelliert man auch die Ansteuerung eines Roboters, kommt die Schnittstelle Robotersteuerung hinzu

© Prof. Dr. Liggesmeyer

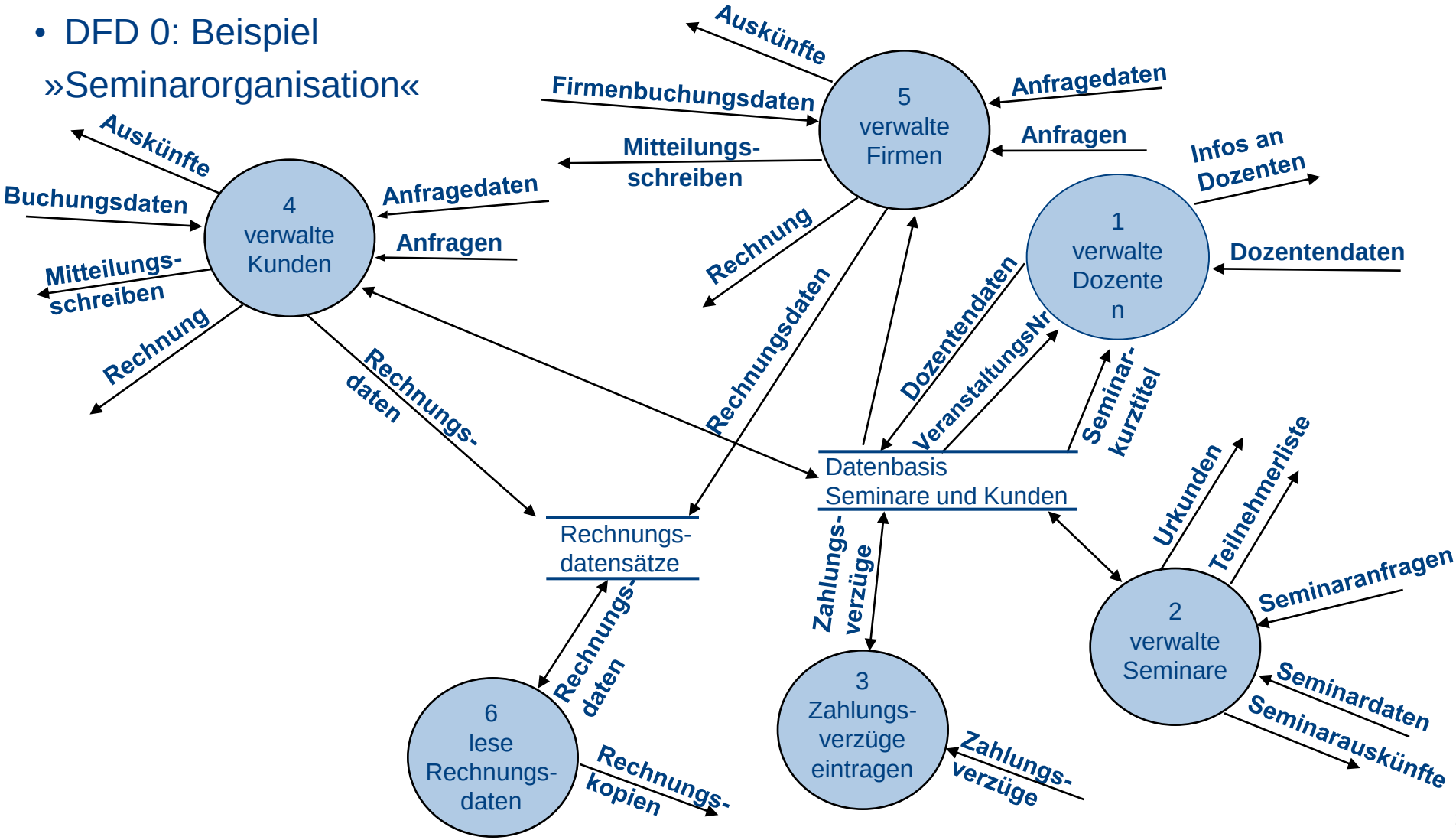
Verfeinerte Datenflussdiagramme

- Verfeinerte Datenflussdiagramme
  - Prozess 0 im Kontextdiagramm wird in Teilprozesse gegliedert
    ➔ funktionale Dekomposition
  - Darstellung erfolgt im Diagramm 0
  - Jeder Prozess wird fortlaufend numeriert
  - Es werden Speicher eingefügt
    - Ist die Ermittlung der Anzahl und Struktur der Speicher schwierig, sollte zunächst ein ER-Modell erstellt werden
    - Evtl. die im ER-Modell gefundenen Speicher für die Darstellung im SA-Modell komprimiert als Datenbasis darstellen

© Prof. Dr. Liggesmeyer

Verfeinerte Datenflussdiagramme

- DFD 0: Beispiel »Seminarorganisation«



© Prof. Dr. Liggesmeyer

Verfeinerte Datenflussdiagramme
Regeln für DFDs

- Schnittstellen können nicht verfeinert werden
- Speicher können nicht verfeinert werden
- Speicher können aber auf allen Verfeinerungen unverändert wiederholt werden
- Anzahl der Prozesse auf einem Diagramm soll nicht größer als sieben sein
- Abstraktionsniveau der Prozesse und Datenflüsse sollte gleich sein

© Prof. Dr. Liggesmeyer

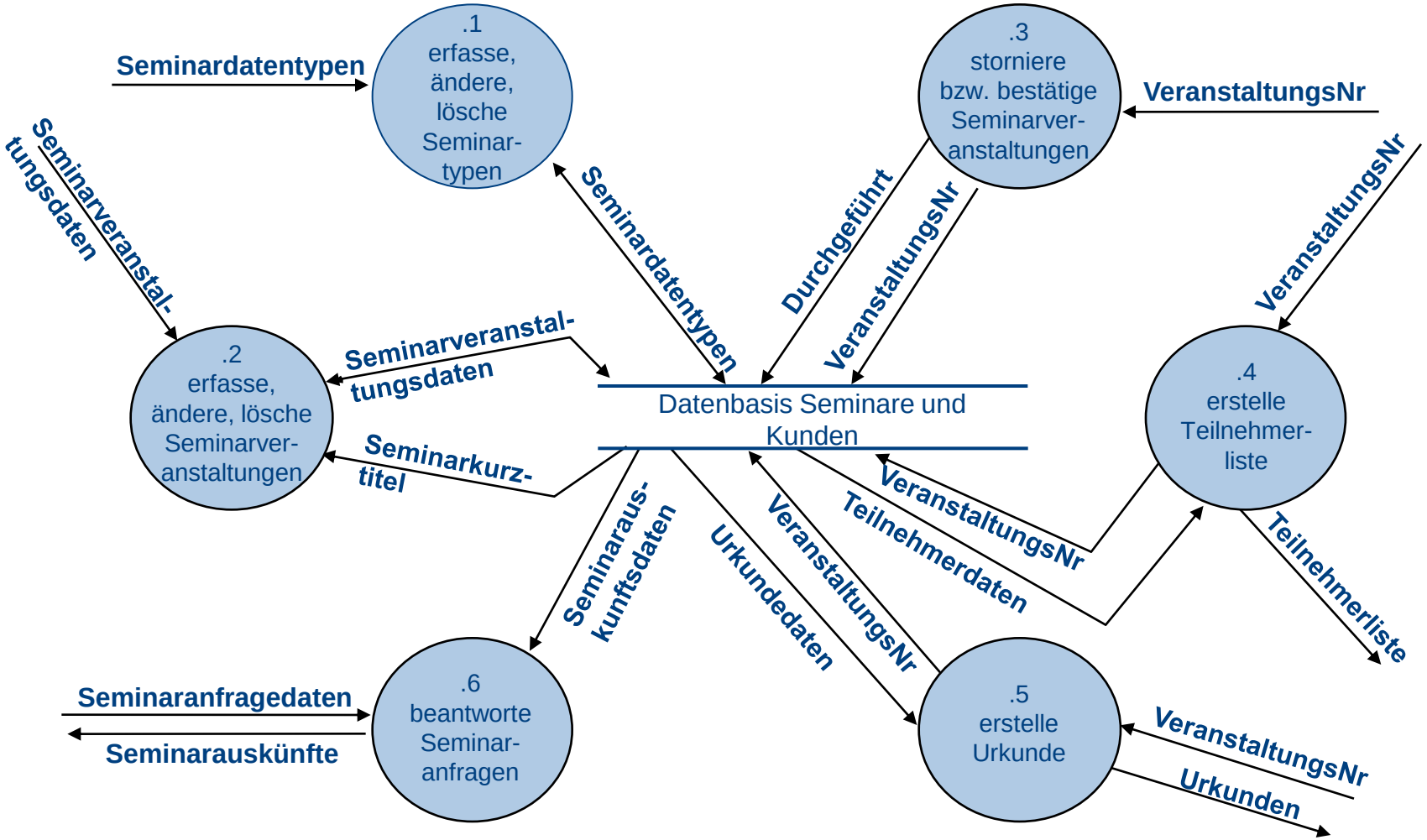
Verfeinerte Datenflussdiagramme
Regeln für DFDs

- Prozesse werden von 1 beginnend fortlaufend nummeriert
- Vor jeder Zahl steht ein Punkt: .1, .2, .3
- Jedes Diagramm trägt die Nummer, die seine Stellung in der Hierarchie angibt
  - DFD 4.3
    Verfeinerung des Prozesses 3 des DFDs 4
- Zur eindeutigen Kennzeichnung wird vor jeden Prozess die DFD-Nummer gesetzt
  - Prozess 4.3.1
    1. Prozess im Diagramm 4.3
- Die Nummernsystematik wird im Allgemeinen vom eingesetzten CASE-Werkzeug verwaltet

© Prof. Dr. Liggesmeyer

Verfeinerte Datenflussdiagramme

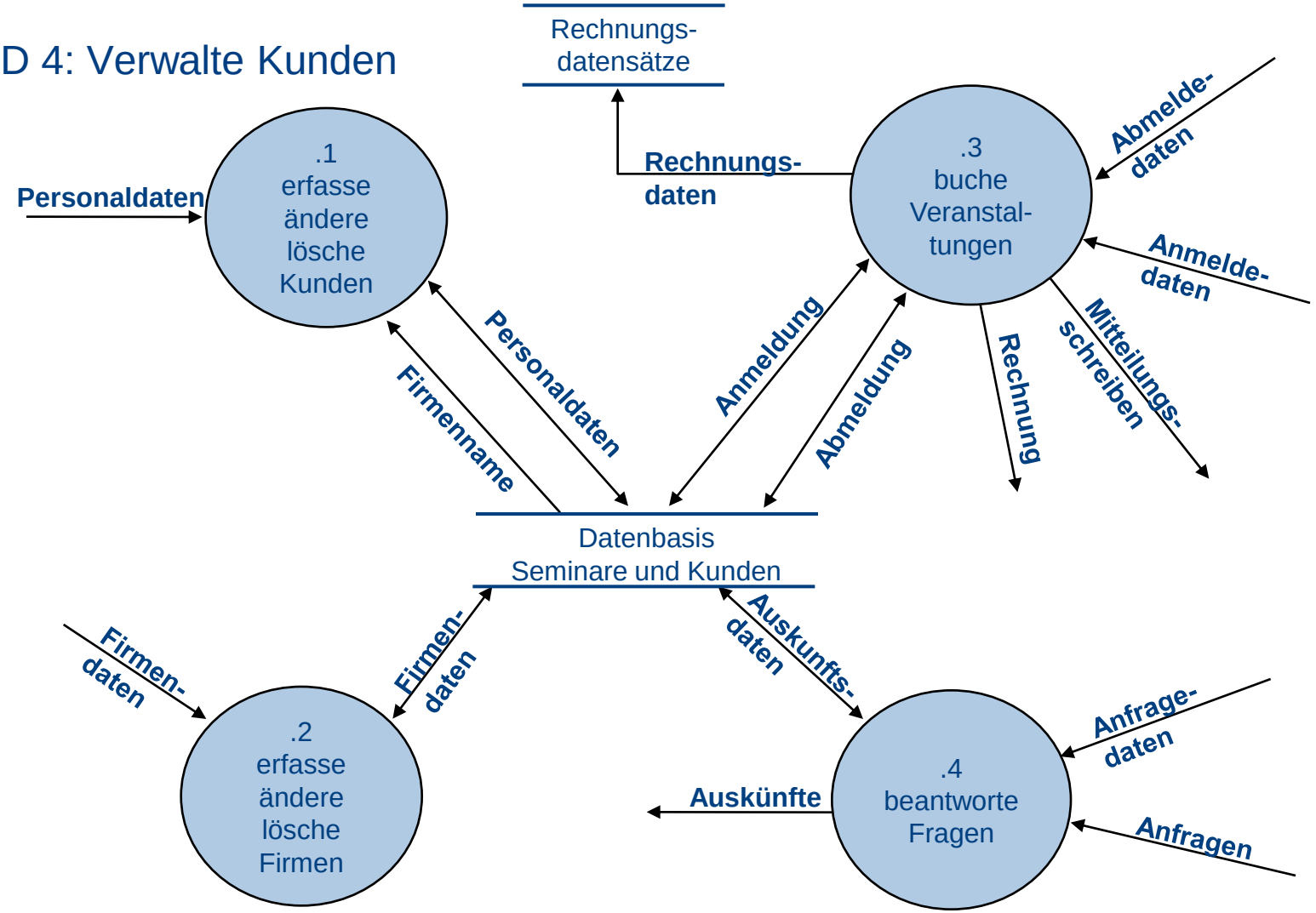
- DFD 2: verwalte Seminare



© Prof. Dr. Liggesmeyer

Verfeinerte Datenflussdiagramme

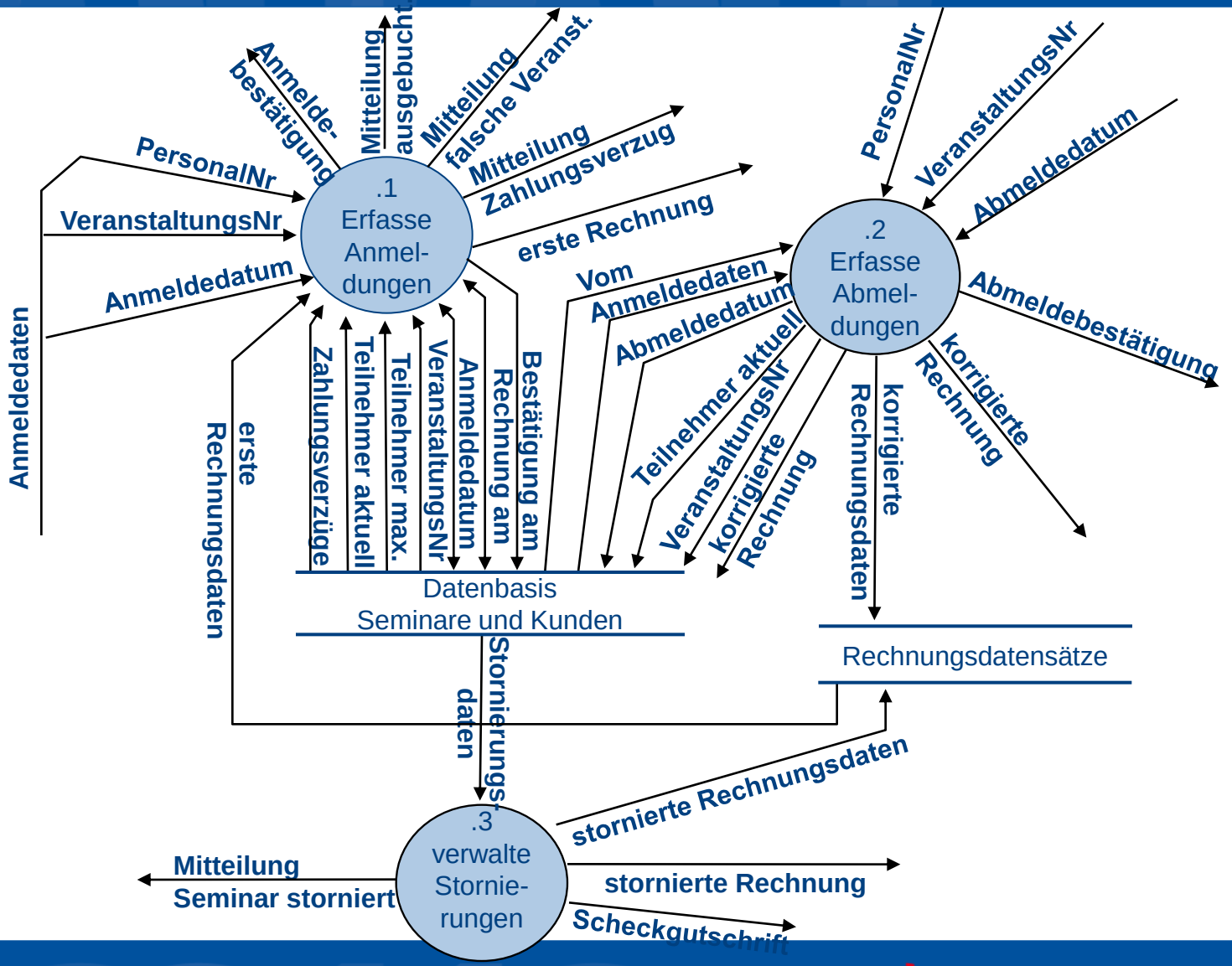
- DFD 4: Verwalte Kunden



© Prof. Dr. Liggesmeyer

Verfeinerte Datenflussdiagramme

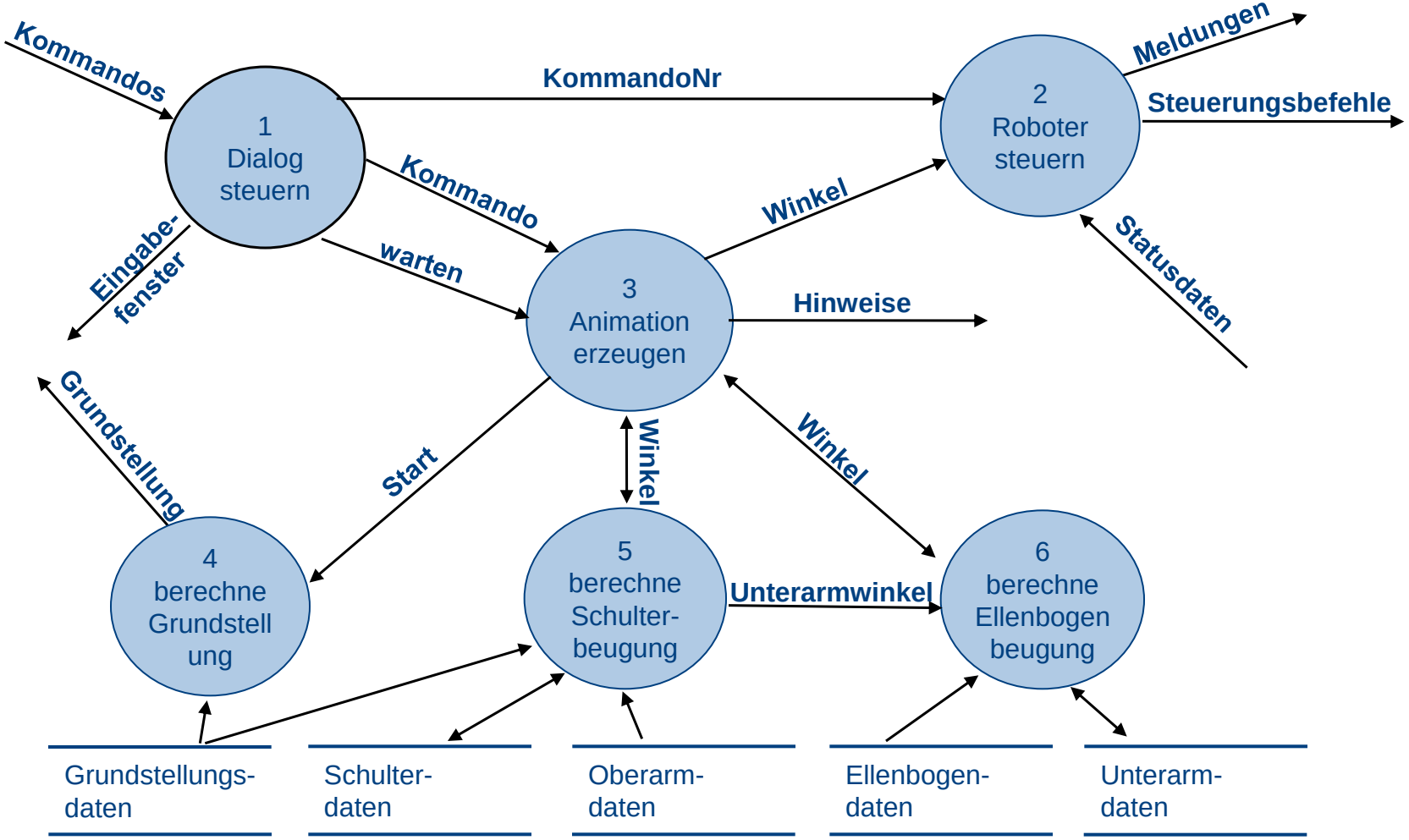
- DFD 4.3: buche Veranstal-tungen



© Prof. Dr. Liggesmeyer

Verfeinerte Datenflussdiagramme

- DFD 0: Roboteranimation



© Prof. Dr. Liggesmeyer

DD-Einträge und Datenintegrität
Verfeinerung der Daten

- Parallel zur Verfeinerung der DFDs und der Prozesse werden auch die Daten verfeinert
- Jeder Datenflusspfeil erhält einen Datenflussnamen
  - Ausnahme: Datenfluss geht zum oder kommt vom Speicher und greift auf den gesamten Inhalt zu
- Jeder Datenflussname ist im Data Dictionary definiert
- Jeder Speicher trägt einen Namen
- Jeder Speichername ist im Data Dictionary definiert

© Prof. Dr. Liggesmeyer

DD-Einträge und Datenintegrität

- Beispiel »Seminarorganisation«
  - Beim Übergang vom Kontextdiagramm auf DFD 0 wurden keine Datenflüsse verfeinert
  - Es wurden neue Datenflüsse zwischen Speichern und Prozessen eingefügt
  - In DFD 4 wurden Datenflüsse verfeinert; dies muss im Data Dictionary festgehalten werden
    - Anfragedaten = Personaldaten + (Firmendaten)
    - Buchungsdaten = Anmeldedaten + (Abmeldedaten)
  - Sind die Datenflussnamen in mehreren DFDs identisch, dann handelt es sich um die selben Datenflüsse

© Prof. Dr. Liggesmeyer

DD-Einträge und Datenintegrität

- Beim Übergang von DFD 4 auf DFD 4.3 ergeben sich folgende DD-Einträge

Rechnung = [erste Rechnung | korrigierte Rechnung |
stornierte Rechnung]

Rechnungsdaten = [erste Rechnungsdaten |
korrigierte Rechnungsdaten |
stornierte Rechnungsdaten]

Mitteilungsschreiben = [Mitteilung Ausgebucht |
Mitteilung falsche Veranstaltung |
Mitteilung Zahlungsverzug |
Anmeldebestätigung |
Abmeldebestätigung |
Mitteilung Seminar storniert]
+ (Scheckgutschrift)

© Prof. Dr. Liggesmeyer

DD-Einträge und Datenintegrität

Anmeldedaten = PersonalNr + VeranstaltungsNr + Anmeldedatum

Abmeldedaten = PersonalNr + VeranstaltungsNr + Abmeldedatum

Anmeldung = Bestätigung am + Rechnung am + Anmeldedatum
+ VeranstaltungsNr + Teilnehmer max
+ Teilnehmer aktuell + Zahlungsverzüge

Abmeldung = Vom + Anmeldedatum + Abmeldedatum
+ Teilnehmer aktuell + VeranstaltungsNr
+ korrigierte Rechnung am

© Prof. Dr. Liggesmeyer

DD-Einträge und Datenintegrität balancing (Konsistenzbedingung)

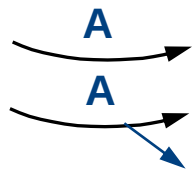
Alle Datenflüsse des untergeordneten DFDs müssen entweder

- Im übergeordneten DFD unter gleichem Namen erscheinen oder
- Teilkomponente eines Datenflusses sein und damit im DD beschrieben werden

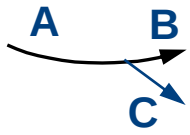
- Ist diese Eigenschaft zwischen allen Diagrammen erfüllt, spricht man von einem ausbalancierten Datenmodell (balancing)
- Alle Datenflussdiagramme müssen innerhalb der Hierarchie ineinander substituierbar sein
- Es gibt keinen Schutzmechanismus, d.h. Namen müssen global eindeutig sein

© Prof. Dr. Liggesmeyer

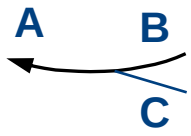
DD-Einträge und Datenintegrität
Darstellungsmöglichkeiten von Datenflüssen



- A fließt in Pfeilrichtung
- Der gesamte Inhalt von A fließt in beide Zweige



- A teilt sich in die Komponenten B und C (weitere Komponenten enthält A nicht, d.h. A = B + C)



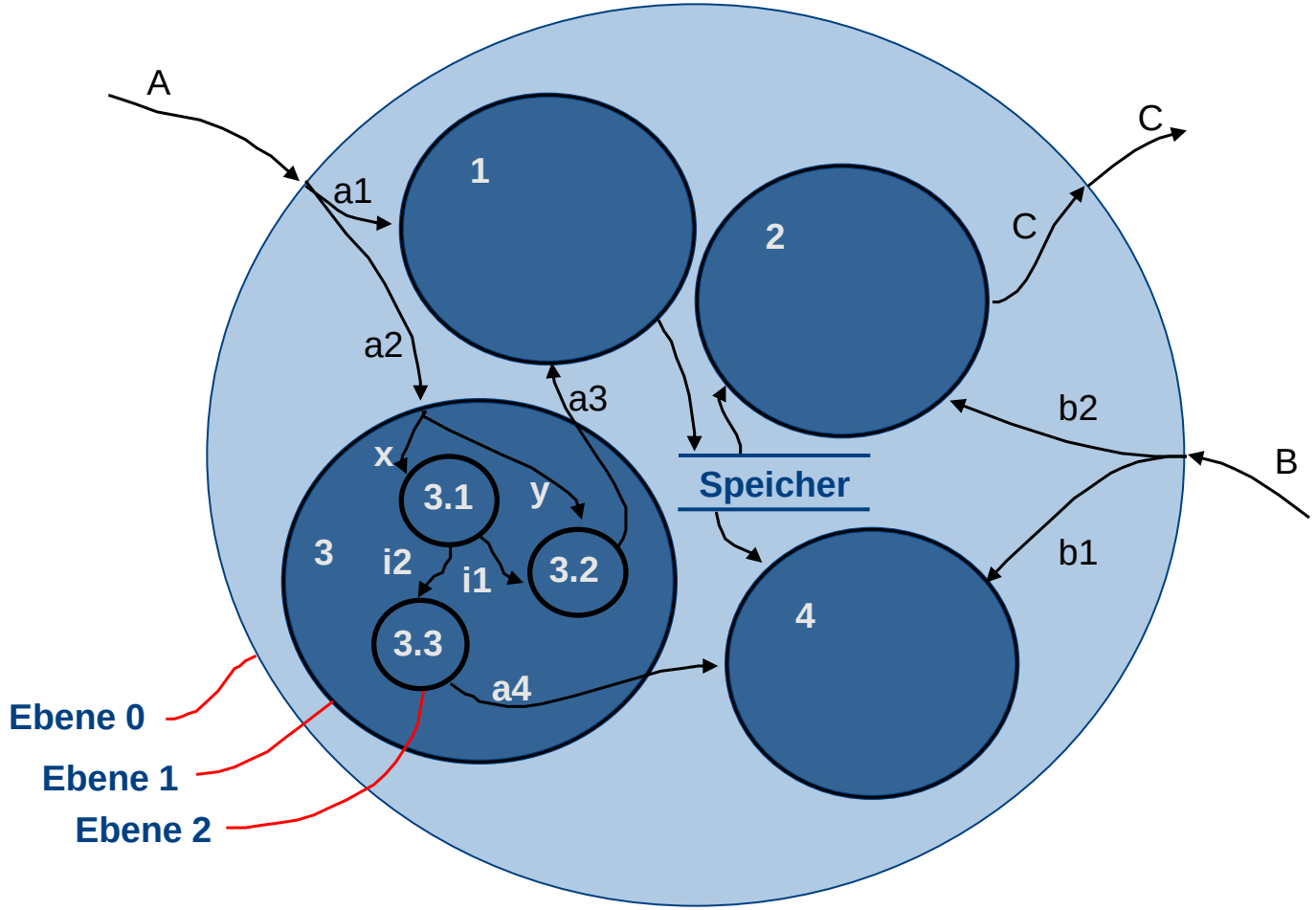
- A entsteht aus den Komponenten B und C (weitere Komponenten enthält A nicht, d.h. A = B + C)

- A fließt in beide Richtungen (z.B. von bzw. zu einem Speicher)

- X, Y, Z fließen getrennt in Pfeilrichtung

© Prof. Dr. Liggesmeyer

- Beispiel: substituierte SA-Hierarchie



© Prof. Dr. Liggesmeyer

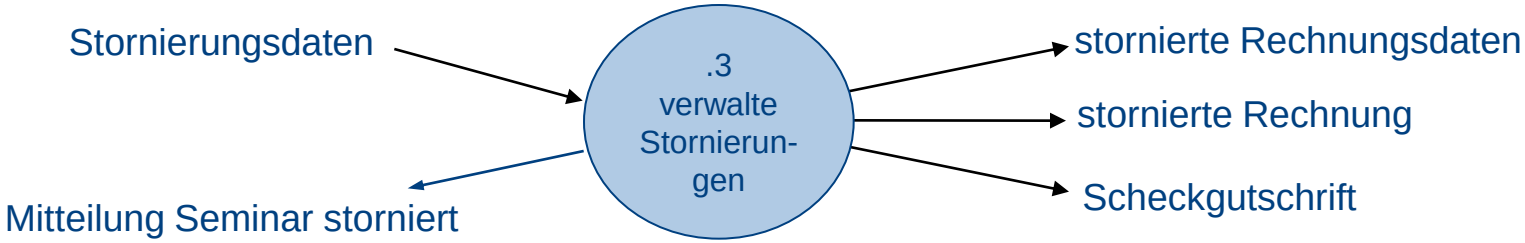
Mini-Spezifikationen

- Jeder Prozess, der nicht weiter verfeinert wird, muss durch eine MiniSpec beschrieben werden
- Jede MiniSpec muss beschreiben, wie die Eingaben in die Ausgaben transformiert werden
- Eine MiniSpec darf keine Implementierungsvorschriften enthalten
- MiniSpecs können als Pseudocode, Entscheidungstabelle oder Entscheidungsbäume abgefaßt sein

© Prof. Dr. Liggesmeyer

Mini-Spezifikationen

- Beispiel



- MiniSpec von Prozess 4.3.3 der »Seminarorganisation«

  `Stornierungsdaten` aus der Datenbank lesen

  An alle Teilnehmer die `Mitteilung Seminar storniert` /F110/ versenden

  if Teilnehmer hat Rechnung bereits gezahlt

  then `stornierte Rechnung` und `Scheckgutschrift` mitversenden

  end if

  `stornierte Rechnungsdaten` in Speicher Rechnungsdatensätze eintragen

© Prof. Dr. Liggesmeyer

Methodik

1. Festlegung der Schnittstellen zur Umwelt
2. Identifizierung aller Eingabe- und Ausgabedatenflüsse von den Schnittstellen zum Prozess 0; vorläufige Datenflussnamen wählen
3. Prozesse und Funktionen ermitteln, die Eingaben in Ausgaben transformieren; vorläufige Prozessnamen wählen
4. Anzahl und Art der Speicher ermitteln, evtl. ER-Modell erstellen
5. Verfeinern der ermittelten Datenflüsse und Zuordnung zu Prozessen und Speichern
6. Definition der Datenflüsse und Speicher im DD

© Prof. Dr. Liggesmeyer

Methodik

7. Überarbeiten des Modells
   - Initialisierung und Terminierung ignorieren, Kontrollflüsse entfernen, triviale Fehlermeldungen weglassen, sorgfältige und eindeutige Namenswahl der Datenflüsse
   - Eindeutige Beschriftung aller Prozesse (Der Prozessname soll alle Aktionen des Prozesses erfassen)
   - Evtl. neue Strukturierung des Systems
8. Ausgehend von der konsolidierten Modellierung des Kontextdiagramms und des Diagramms 0 schrittweise Erarbeitung weiterer Ebenen
9. MiniSpecs für alle nicht weiter verfeinerten Prozesse erstellen
10. Verfeinerung beenden, wenn jeder Prozess überschaubar ist; nicht unnötig tief verfeinern!

© Prof. Dr. Liggesmeyer

Methodik
Hinweise

- Kontextdiagramm und Diagramm 0 können zunächst als ein Diagramm entwickelt werden
  - Erst zum Schluss wird aus diesem Diagramm das Kontextdiagramm abstrahiert
- Kontextdiagramm und Diagramm 0 sind die entscheidenden Diagramme
  - Sie sollten zunächst interaktiv ohne CASE-Werkzeug im Team entwickelt werden
  - Bei umfangreichen Problemen zwei getrennte Teams einsetzen und die Ergebnisse diskutieren

© Prof. Dr. Liggesmeyer

Wertung

- Vorteile
  - Geschickte Kombination bewährter Basistechniken
  - Durch hierarchisch gegliederte DFDs Verbesserung der Übersichtlichkeit
  - Viele analytische QS-Möglichkeiten durch Quervergleiche der in den Basistechniken beschriebenen Aspekte
  - Leicht erlernbar
  - Erlaubt eine top-down-Einarbeitung in ein System
  - Guter Zusammenhang zu ER-Modellen über Speicher herstellbar

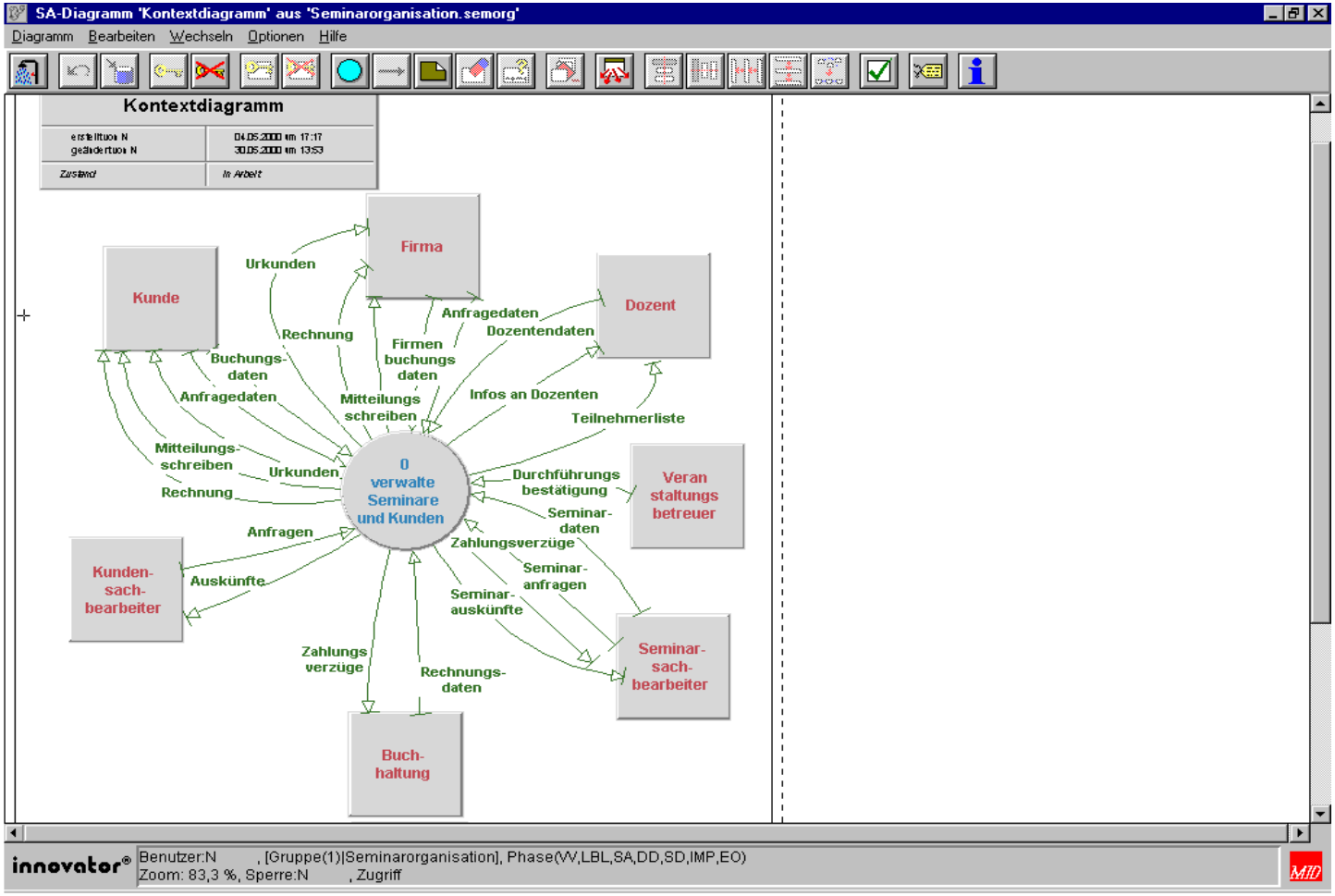
© Prof. Dr. Liggesmeyer

Wertung

- Nachteile
  - Schnittstellen können nicht verfeinert werden
    - Bei umfangreichen Schnittstellen gibt es dann Darstellungsprobleme
  - Speicher können nicht verfeinert werden
    - Ausweg: Globale Datenbasis verwenden, Verfeinerung durch ein ER-Modell beschreiben
  - Es entsteht ein Strukturbruch, wenn ein SA-Modell in einen datenabstraktionsorientierten Entwurf (sogenanntes Modular Design) transformiert werden soll

© Prof. Dr. Liggesmeyer

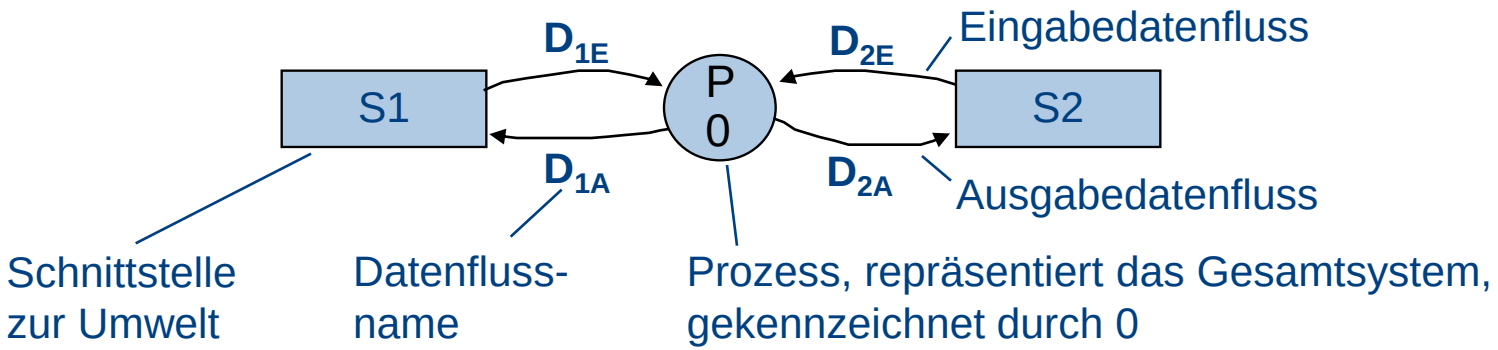
- Beispiel: Innovator



© Prof. Dr. Liggesmeyer

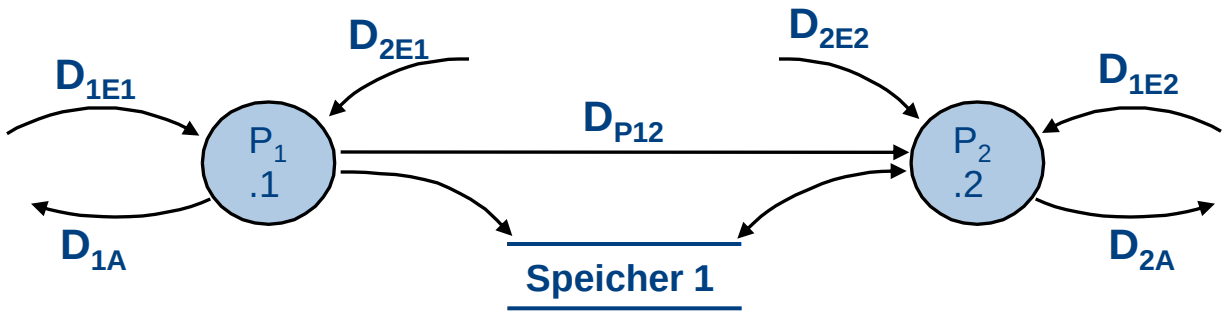
SA im Überblick

- Drei Konzepte
  - Hierarchisch angeordnete DFDs
  - Data Dictionary-Einträge (DDs)
  - Mini-Spezifikationen (MiniSpecs)
- Kontextdiagramm = oberstes DFD
  - Beschreibt die Umwelt des Systems
  - Regeln: Genau ein Prozess, mindestens eine Schnittstelle, kein Speicher



© Prof. Dr. Liggesmeyer

- DFD 0 = Verfeinerung des Kontextdiagramms



DD:

$D_{1E} = D_{1E1} + D_{1E2}$

$D_{2E} = (D_{2E1})\ D_{2E2}$

- Regeln
  - Prozess 0 wird in Teilprozesse (hier: P1 und P2) zerlegt
  - Datenflüsse werden ebenfalls verfeinert (hier: D1E und D2E)
  - Speicher werden eingeführt (hier: Speicher 1)
  - Neue Datenflüsse zwischen Prozessen und zwischen Prozessen und Speichern werden eingetragen
  - Schnittstellen und Speicher können nicht verfeinert werden, dürfen aber auf tieferen Diagrammen wiederholt werden

© Prof. Dr. Liggesmeyer

SA im Überblick

- DFD1, DFD2 …, DFD 1.1, … DFD 2.n …
  - Jeder Prozess kann weiter zu einem neuen Diagramm verfeinert werden
  - Anzahl der Prozesse pro Diagramm kleiner als sieben
  - Ist ein Prozess ausreichend verstanden, dann erfolgt keine weitere Verfeinerung, sondern eine Beschreibung durch eine MiniSpec

© Prof. Dr. Liggesmeyer

SA im Überblick

- DD-Eintrag und Datenintegrität (balancing)
  - Jeden Datenfluss und Speicher im DD definieren
  - Zusammenhang zwischen Datenflüssen der einzelnen DFDs über das DD herstellen
  - Alle Datenflüsse eines untergeordneten DFDs müssen im übergeordneten DFD
    - mit gleichem Namen (hier: D1A, D2A) erscheinen
    - oder Teilkomponenten eines Datenflusses sein
      (hier: D1E = D1E1 + D1E2, D2E = (D2E1) + D2E2)
    - Gilt dies für alle DFDs, dann liegt Datenintegrität vor (balancing)
- MiniSpec = beschreibt einen nicht weiter verfeinerten Prozess
  - MiniSpec beschreibt, wie die Eingabedatenflüsse eines Prozesses in die Ausgabedatenflüsse transformiert werden
  - Beschreibung erfolgt durch Pseudocode, ET oder Entscheidungsbäume

© Prof. Dr. Liggesmeyer