5. Semiformale Notationen für Anforderungsspezifikationen

Ziele:
Notationen für die Teilmodelle einer funktionalen Anforderungsspezifikation
Verschiedene Sichten einer AS
Probleme: Konsistenzen, Methodikregeln, Werkzeugunterstützung
Betrachtung eines größeren Beispiels
Weitere Ansätze

Stand: 25.10.2001

Übersicht

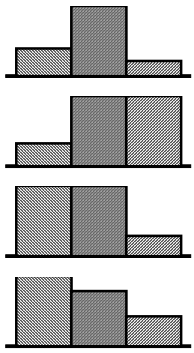
Aufgaben von RE-Sprachen (-Notationen):

- Anwendungsfunktionen und ihre Strukturierung → Funktionsmodell
- Strukturierung der zugrundeliegenden Daten → Informationsmodell
- Abläufe, Koordination, Änderung der Ausführung → Kontrollmodell
- Kontext → alle Teilmodelle

damit wesentlicher Teil der AS (”funktionale” AS) bis auf UI-Festlegung; Rest ist umgangssprachlicher Text

unterschiedl. Systeme haben unterschiedlich reiche Modelle

- Transformationssystem
- reaktives System
- interaktives System
- klass. Informationssystem



Datenmodell

Funktionsmodell

Kontrollmodell

für best. Systeme (Entwicklungssystem im Ingenieurbereich) z. B. SEUen, Simulationssysteme etc. fehlen Ausdrucksmöglichkeiten

- Graphenkomplexe (SEU)
- komplexe Operationen (SEU)
- Gitterstrukturen (SS)
- komplexe Berechnungen (SS)
- Transformation: Logik-Repräsentation

Notationen für das Sollkonzept (und für die Istanalyse)

Forderungen:

- zur Darstellung unterschiedlicher Detaillierung geeignet
- Unterscheidung verschiedenartiger Objekte, Relationen
- Verträglichkeit mit ”Methoden”/Unterstützung solcher Methoden?
- Verständlichkeit und Eindeutigkeit der Konstrukte?
- zur Ermittlung aller Faktoren zwingen?
- leichter oder fließender Übergang zur Entwurfsspezifikation?

Gründe für Diagrammdarstellungen in der AS:

- sonst Übersetzung der AS in ”formale” Notation als erster Schritt des Entwurfs
- Hilfe bei der Kommunikation des Analysators mit sich selbst, des Analysators mit dem Auftraggeber, des Analysators mit späteren Benutzern des Systems, mit Qualitätssicherer, Manager, Entwickler
- Übersichtlichkeit: ein Bild sagt mehr als 1000 Worte
- weniger Redundanz als bei geschriebenem Text
- besser zur Modellierung: Abstraktion, Hierarchisierung, Modularisierung
- leichter wartbar
- leichtere Übersetzung in Entwurfsspezifikation?

(Diagramme best. Klasse von Graphen, die graphisch repr. werden; jede Diagrammklasse entspr. Graphenklasse)

Funktionsmodell mit Structured Analysis

(and Structured/Composite Design) /Yo 89;DeM 79/ ”Methode”

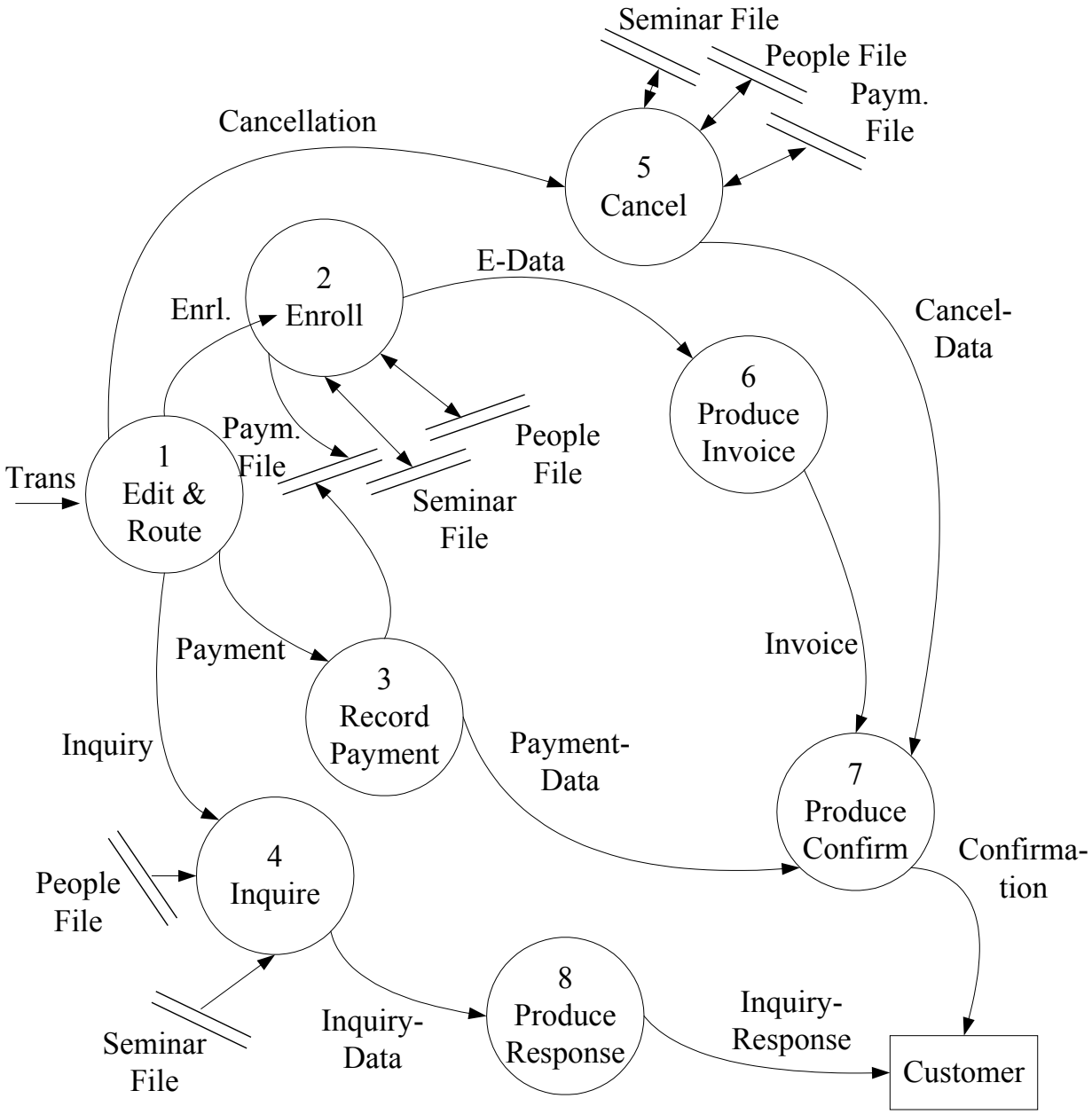
- spezielle Art von Datenflußdiagrammen
- Data Dictionary
- Minispecs

verwandt mit ähnlichen Ansätzen: SADT, SREM, (JACKSON) s. u.

Vielzahl ähnlicher Methoden in der Literatur (89–93): s. Literatur dieses Kapitels

- Unterschiede kaum ermittelbar
- langatmige Erläuterungen
- Präzision der Darstellungen mangelhaft
- ausgearbeitete Beispiele nicht vorhanden

SA-Datenflußdiagramme (Bsp. Seminarverwaltung)



kein Kontrollfluß:
mögl. Datenwege (Kanäle)
nicht welche aktiv, in welcher Situation, in welcher Reihenfolge

Prozeß

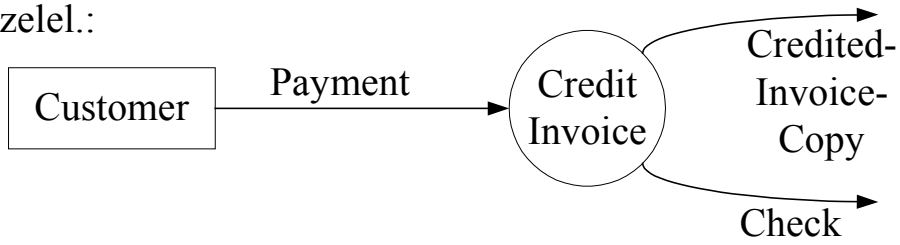
Datenfluß

Datenspeicher:
Datei, Datenbasis

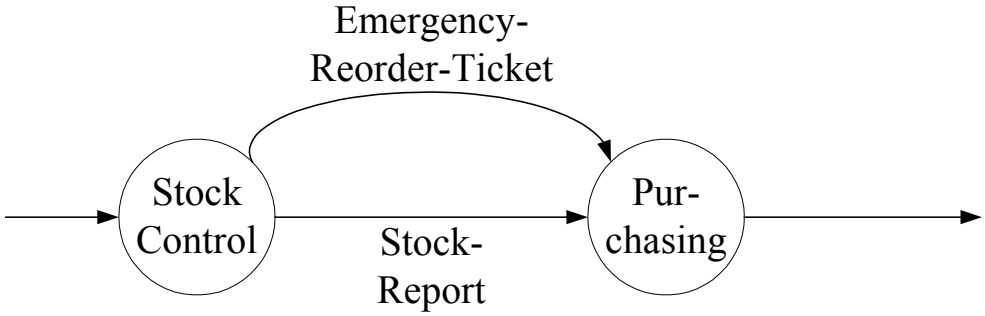
Terminator:
Quelle oder Senke (Außenwelt)
Prozeß oder Datenspeicher

Elemente von SA-DFDs und Konventionen:

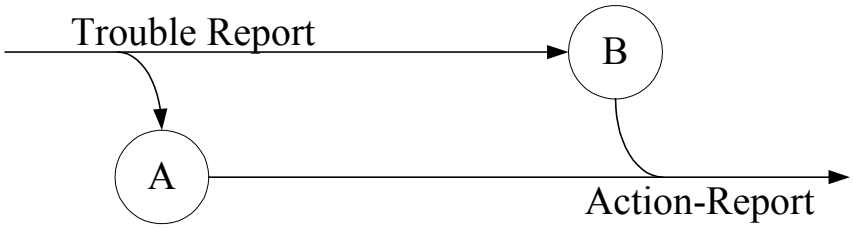
keine Betr. von Einzelel.:
Datenfluß ist Kanal
(Typ, Benennung)



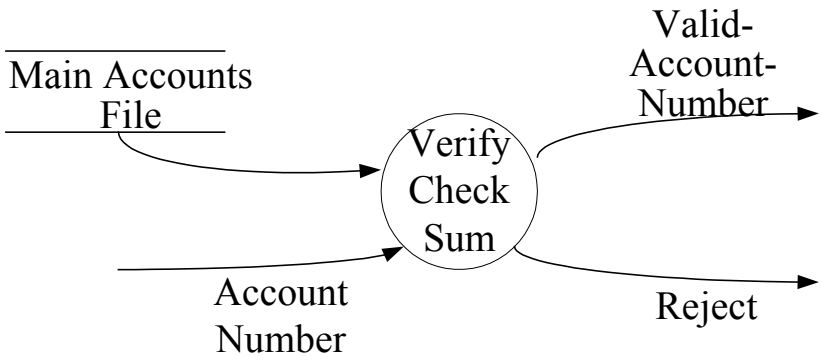
mehrf. Datenfl.



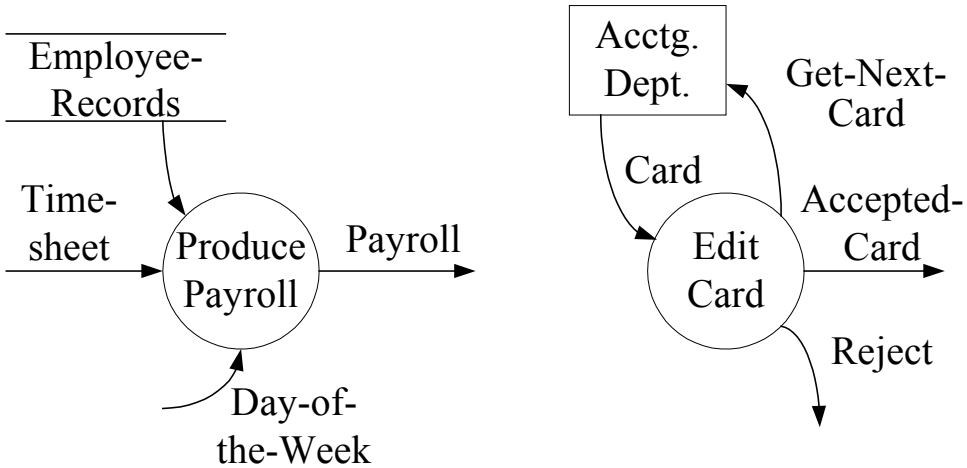
Zusammenführung
und Aufspaltung



unbenannter
Datenfluß



falsche
Situation



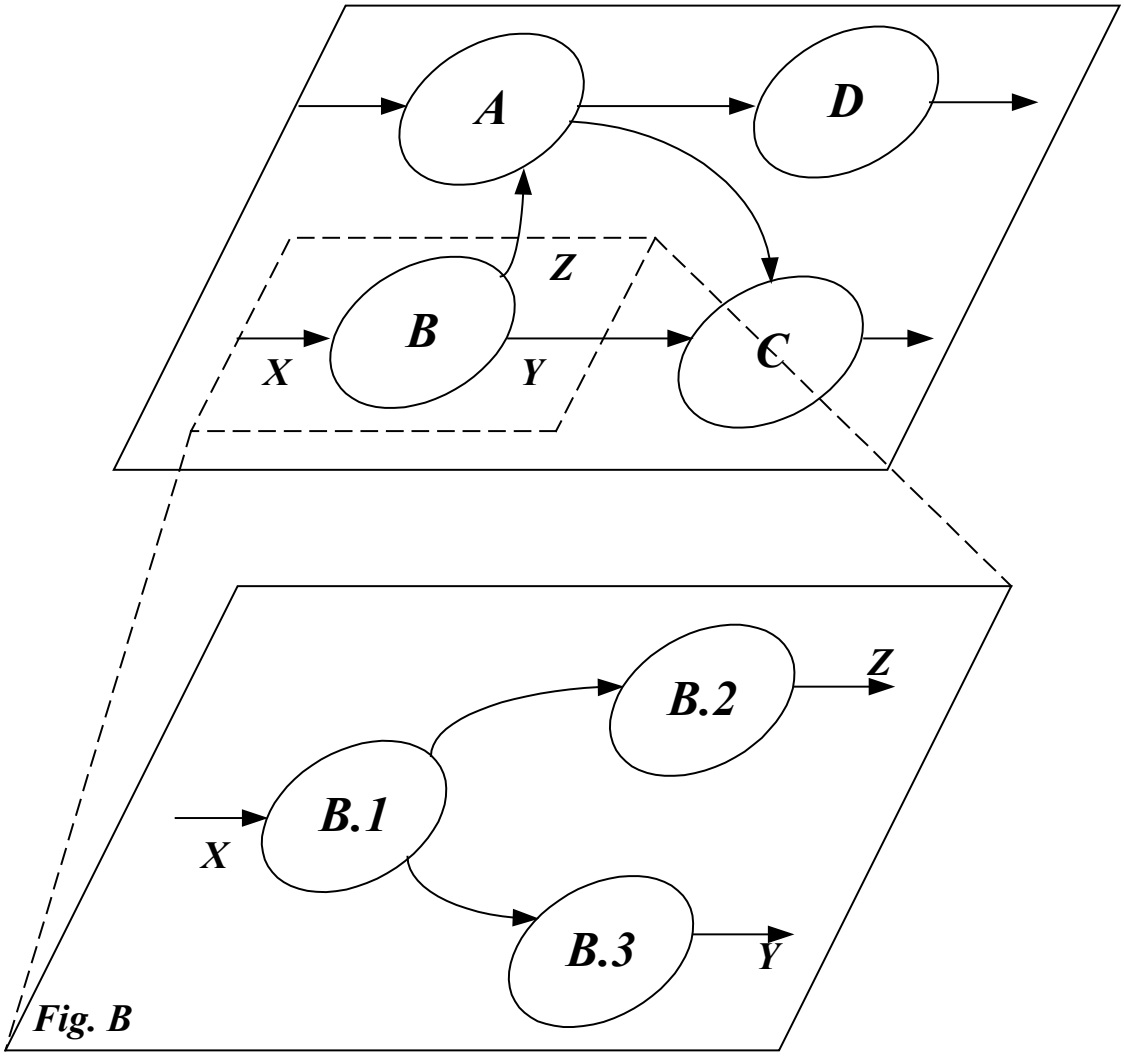
Elemente ... Fortsetzung

Prozeß: Ist eine Transformation von Eingangsdaten in Ausgangsdaten

Datei: Ist ein Datenbehälter
nur Nettodatenfluß ist interessant

Quelle/
Senke: Person oder Organisation, die außerhalb des betrachteten Systems liegt und die Nettodatenerzeuger und Nettodatenempfänger für Systemdaten ist. Alternativ anderweitig erzeugter oder zu bearbeitender Datenbestand.

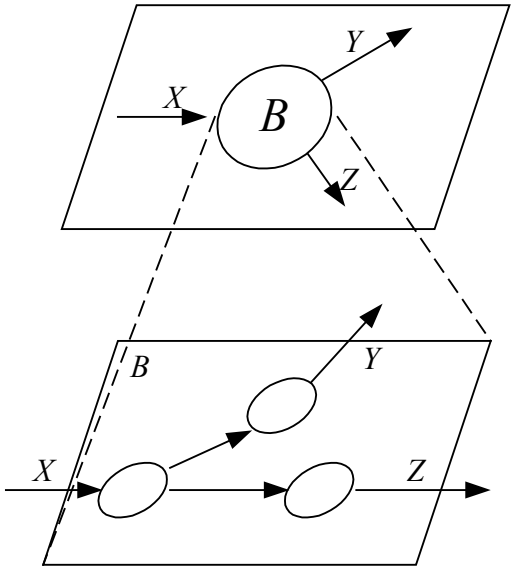
Diagrammhierarchien



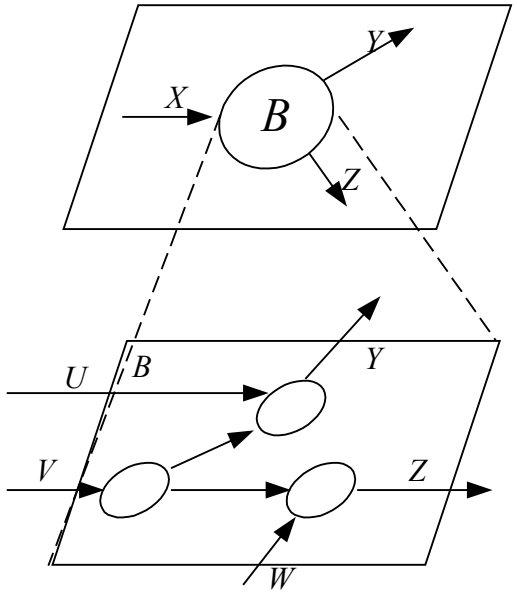
- Kanten bleiben ”erhalten”
- lokale Datenspeicher
- abgeflachte Version durch Ineinandersetzen

Balancierungsregeln:

a) sichtbare Balancierung

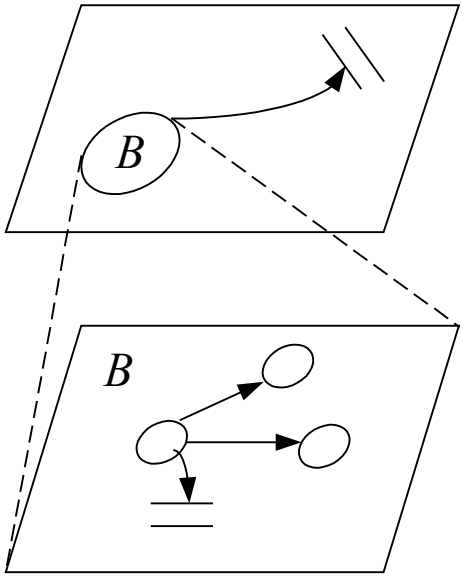


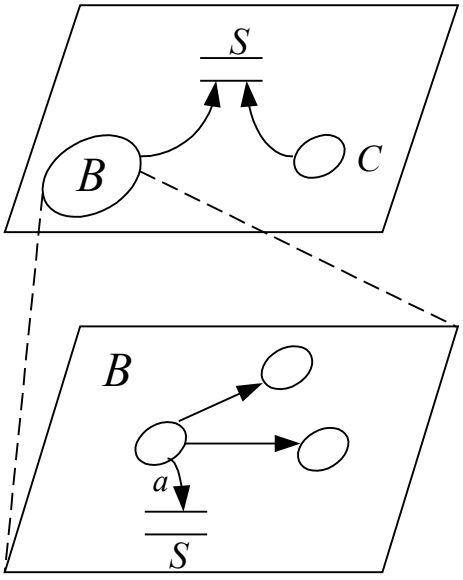
b) Lexikonbalancierung



x=u+v+w
Eintrag im Lexikon

c) Datenspeicherbalancierung





S=a+b+...
Oder S={a}+{b}+...

RE-Datenlexikon

jeder Datenfluß
jedes Datenelement (primitiver Datenfluß)
jeder Datenbestand
jeder Prozeß

} ist Eintrag im RE-DD

Datenflußbeschreibung z. B. in EBNF-Notation

Datenstruktur durch Datenstrukturdiagramm übersichtlicher

(Schemadefinition → Datenbankenvorlesung, bzw. s. u.)

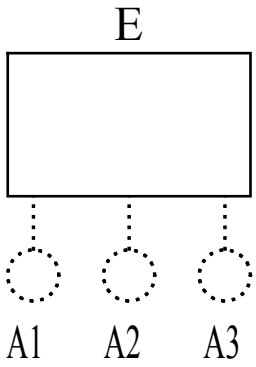
Prozeßkurzbeschreibung (Minispecs) in:

Structured English (wohlstr. Pseudocode)

ET

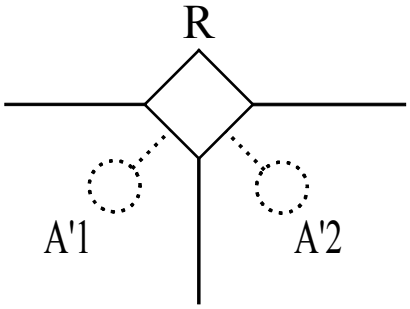
Entscheidungsbäume (ineinanderg. ET)

Informationsmodell: Datenbeschreibungen in ER (Entity-Relationship-Modell)



Entitytypen

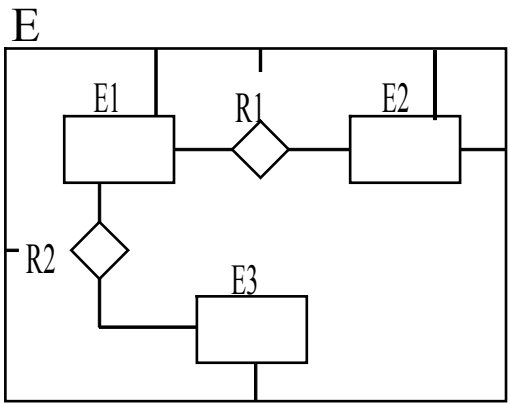
Typ
Name E
Attribute Ai



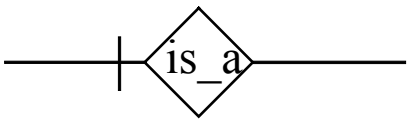
Relationshiptyp

Typ
Name R
Attribute Ai

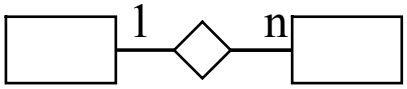
Erweiterungen:



geschachtelte Entitytypen
(geschachtelte Aggregation)
ebenso Relationshiptypen-Aggregation



vordef. Relationen "is_a"
(bzw. "besteht_aus", s. o.)



Kardinalitätsangaben

Kontrollmodell: SA/RT Hatley/Pirbhai, Ward/Mellor

Ereignisse

- Zeitbedingungen
- normaler Kontrollablauf z. B. Schleifen
- Kontrollkoordination: activate, suspend
- Notfallbehandlung: Ausblenden von Prozessen

Erweiterungen gegenüber SA:

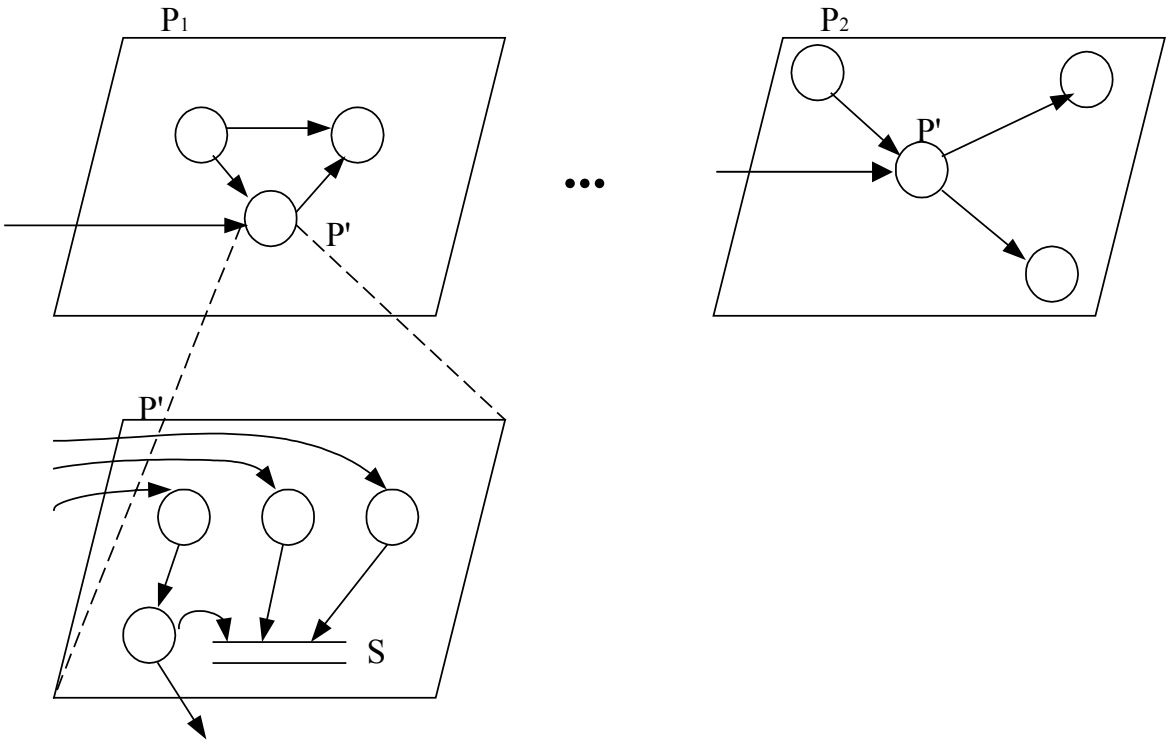
- Signale, Zeitbedingungen
- Kontrollprozesse
- Kontrollprozesse Innenleben: endl. Automaten, erweiterte endl. Automaten (Und/Oder-Dekomposition)

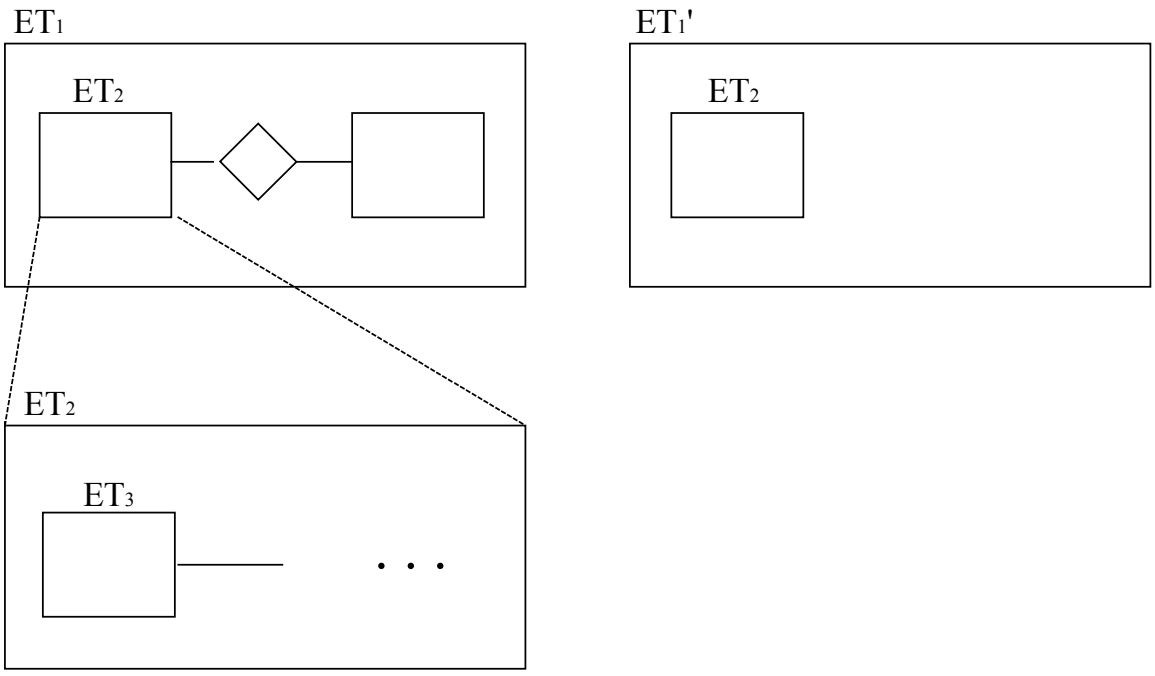
Probleme von RE-Sprachen u. ”Methoden”

Art der Probleme:

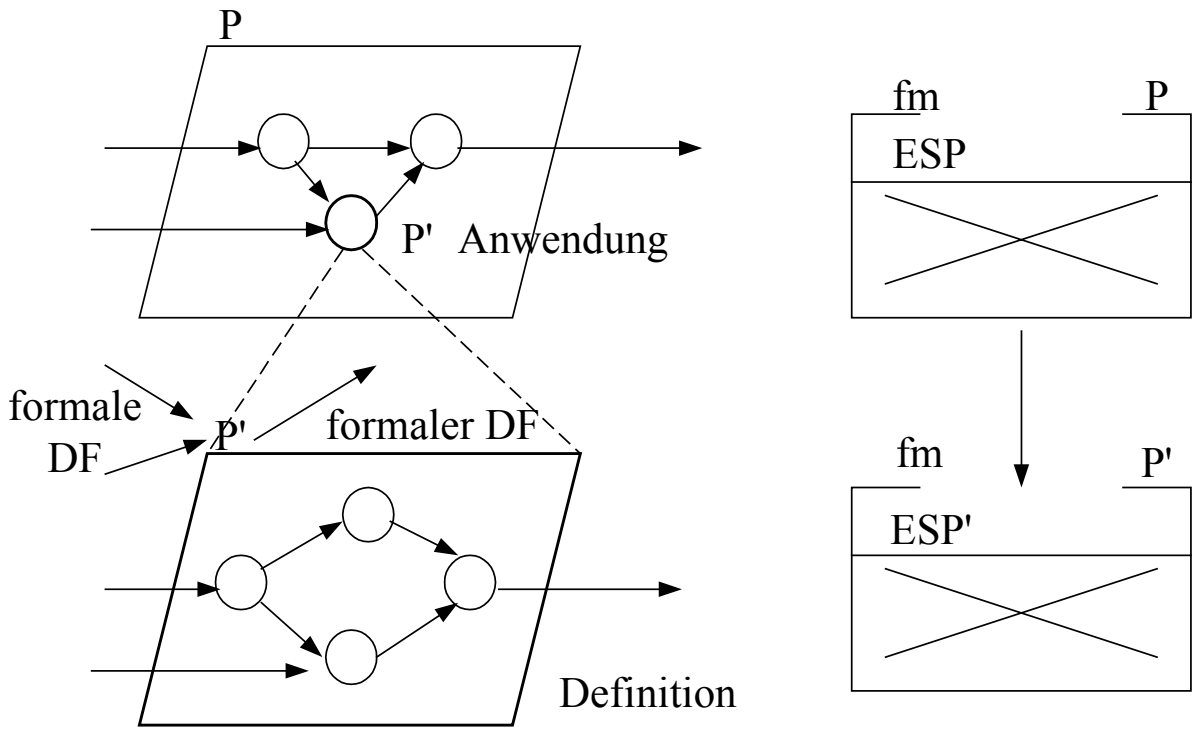
- unsaubere Syntax, Semantik
- mangelnde Ausdruckskraft
- mangelnde Methodik
- Zusammenhang mit anderen ”Methoden”
- (mangelnde Werkzeugunterstützung)

Wie ist Verfeinerung zu verstehen?



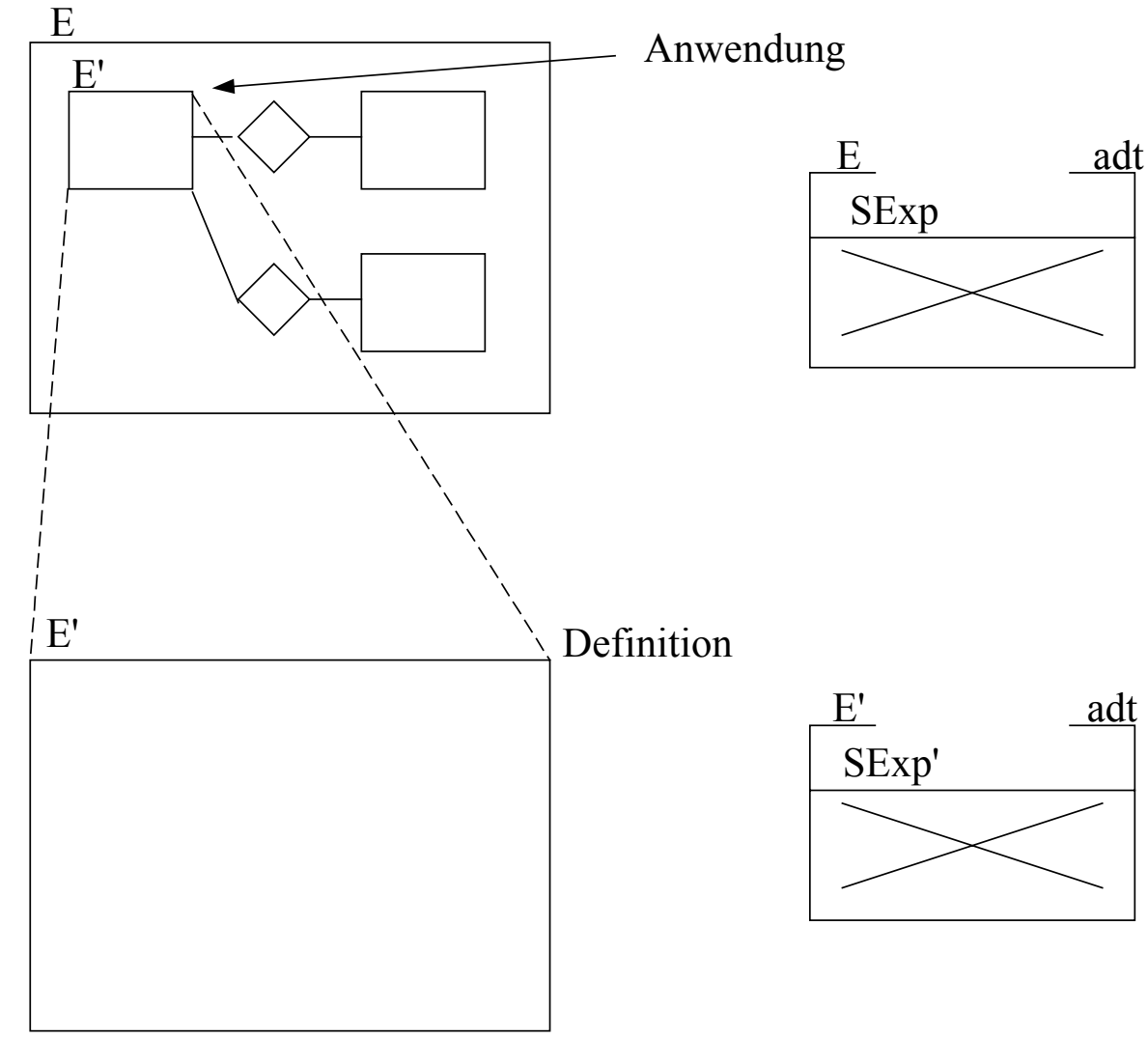


Das Problem der Verfeinerung von Prozessen



- saubere Unterscheidung Definition und Anwendung
- Definition müßte exakt sein: formale Datenflüsse von best. Typ, Semantik auf der Schnittstellenebene
- Balancierungsregeln sind dann kontextsensitive Syntax zwischen Def. und Anwendung, Schnittstelle und ”Rumpf”
- SA-Verfeinerung ist Rumpfverfeinerung (Verfeinerung der Implementierungen); sollte Schichtung von Schnittstellen sein
- Betrachtung der Benutzbarkeitsebene: zu welchem Zweck wird verfeinert: lokaler oder allg. Dienst

Das gleiche Problem bei Verfeinerung in EER

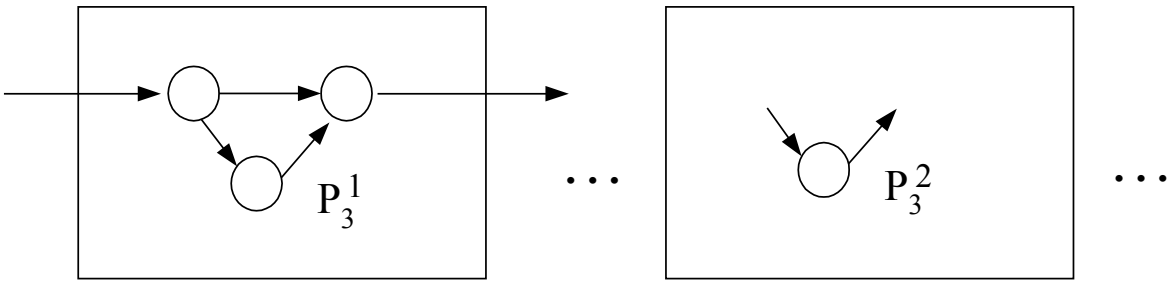


s. o.

- Definition mit Zugriffsoperationen
- Betrachtung Struktur- und Benutzbarkeitsebene
- Rümpfe bleiben ”unbeachtet”

Objekt- und Typinformation

1)



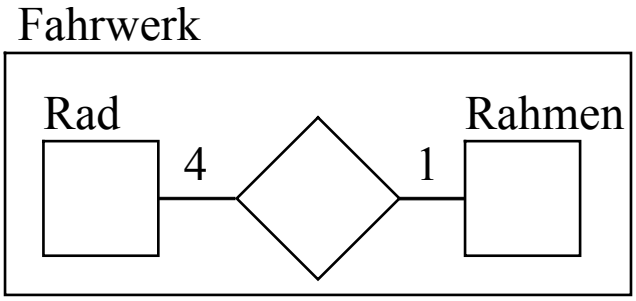
P_3^i haben das gleiche Verhalten:

entweder **ein** Prozeß oder mehrere Prozesse gleichen Verhaltens über Textvervielfältigung

⇒ SA hat nur Ausdrucksmöglichkeiten für Prozeßobjekte aber nicht für Prozeßtypen

Das gleiche gilt für Datentypen

2)

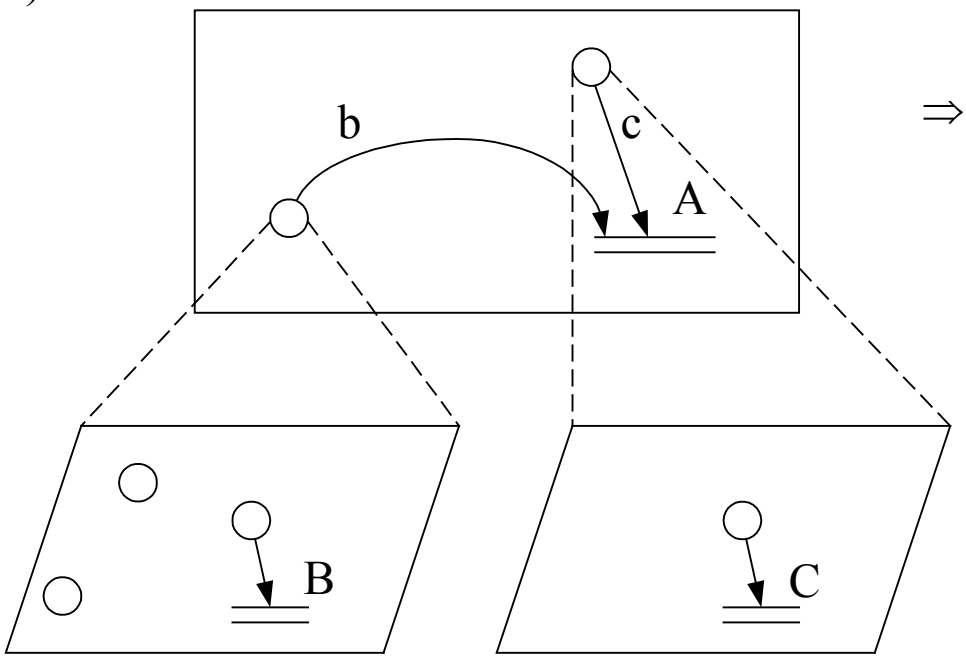


Auswechseln des linken Vorderrades schlecht ausdrückbar Vergleich mit Verbundtypdefinition

⇒ (E)ER hat nur Typ- und keine Objektinformationen

Datenspeicher nicht explizit definiert

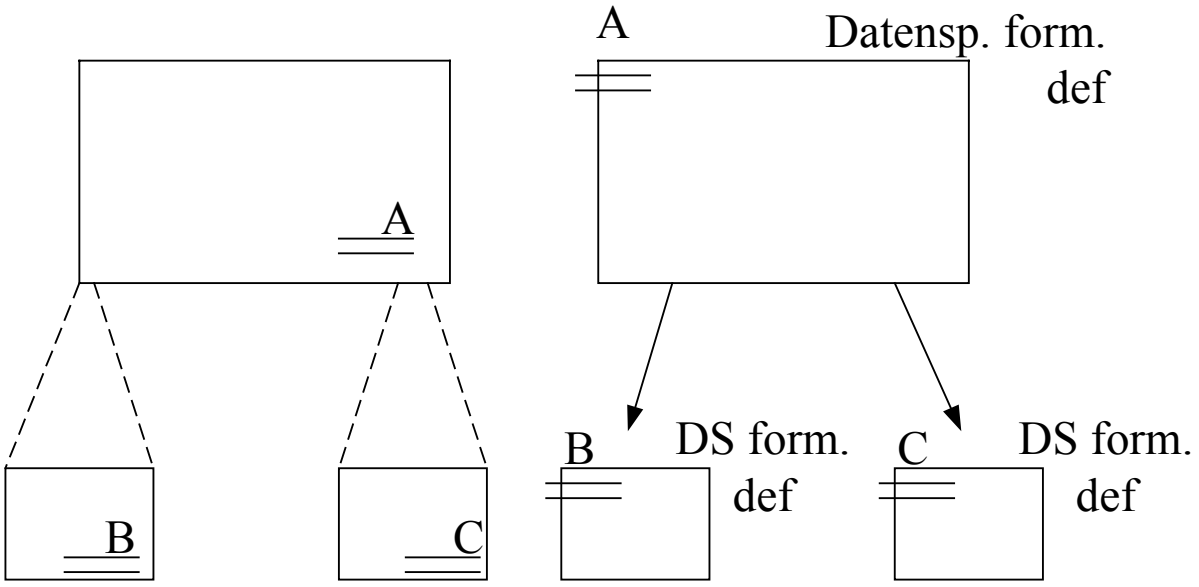
1)



⇒ Datenspeicherver-
feinerungen sollten
zugelassen werden

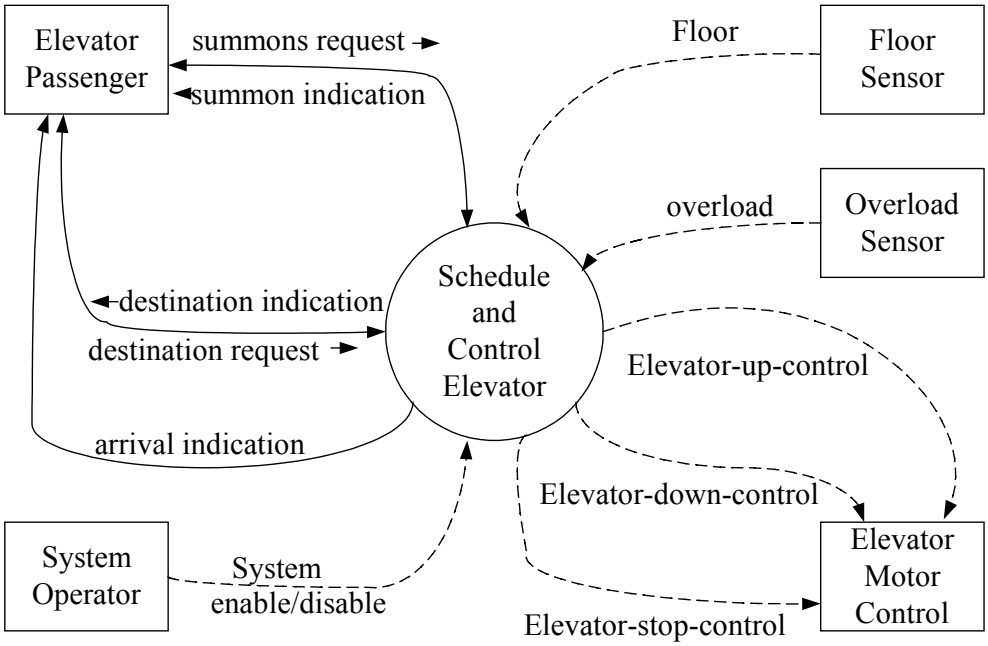
2) Datenspeicher sind stets angewandte Vorkommnisse:
Im Gegensatz zu Prozessen
Entitytypen
} ex. überhaupt kein Definitionsniveau

Das könnte eingeführt werden:



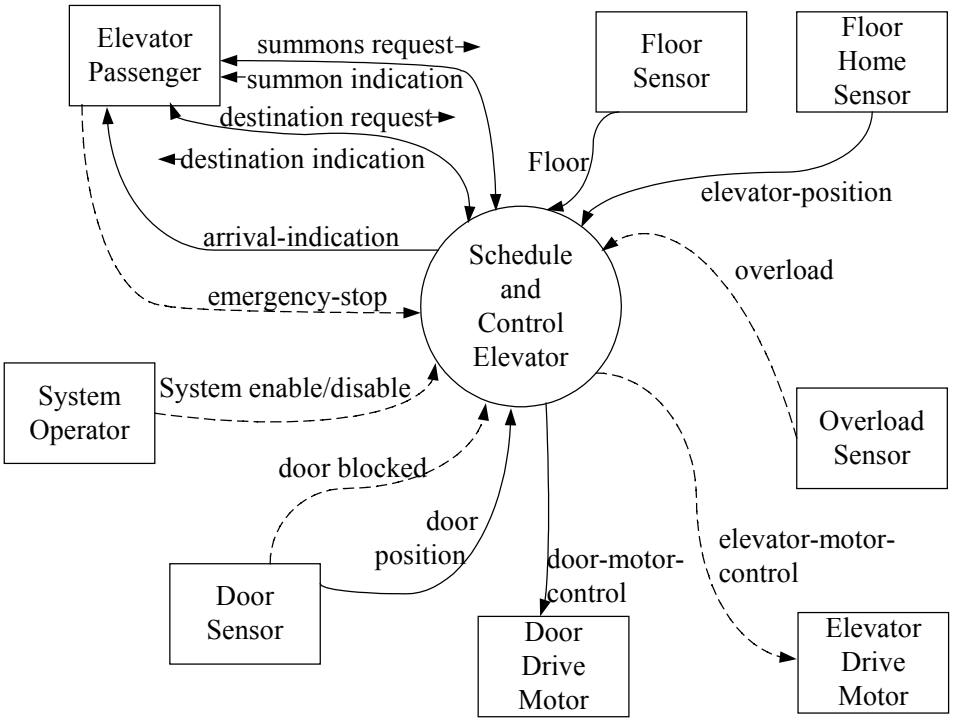
nicht vorhandene Verfeinerungen

- Terminatoren im Context Diagramm



nach /Yo 89/

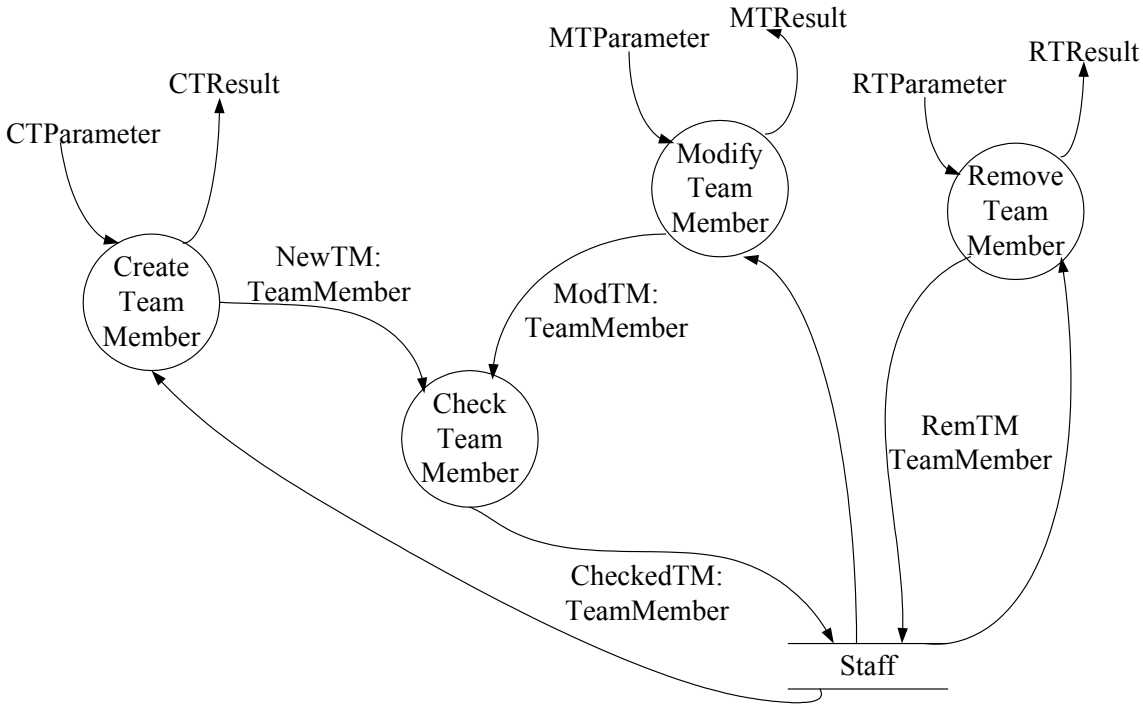
- Expanded Context Diagram



nach /Yo 89/

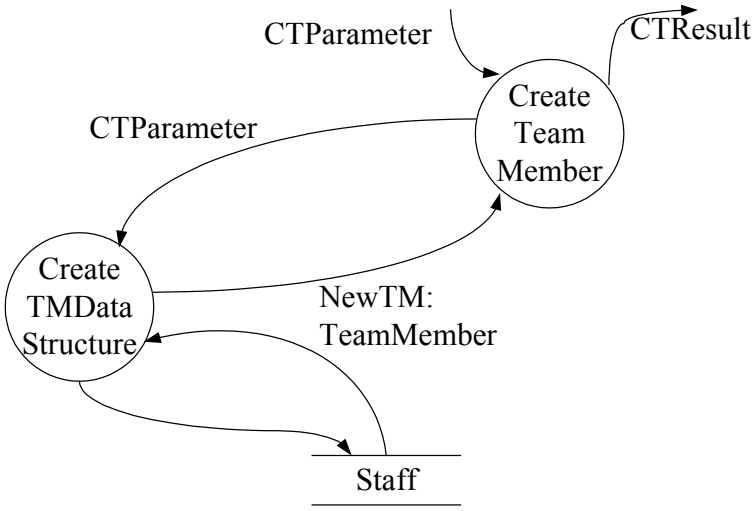
Mangelnde Methodikregeln

SA Diagramm mit unterschiedlichen Verfeinerungssemantiken



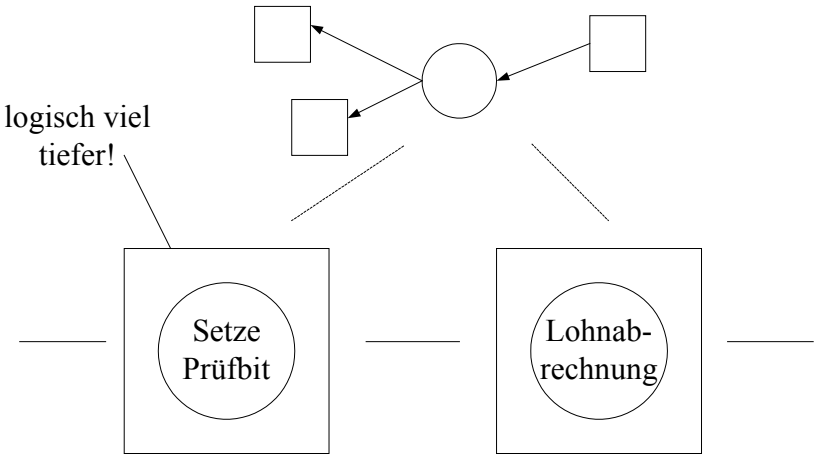
nach /Ja 92/

Modellierung von Prozessen mit der Funktionalität einer Zugriffsressource

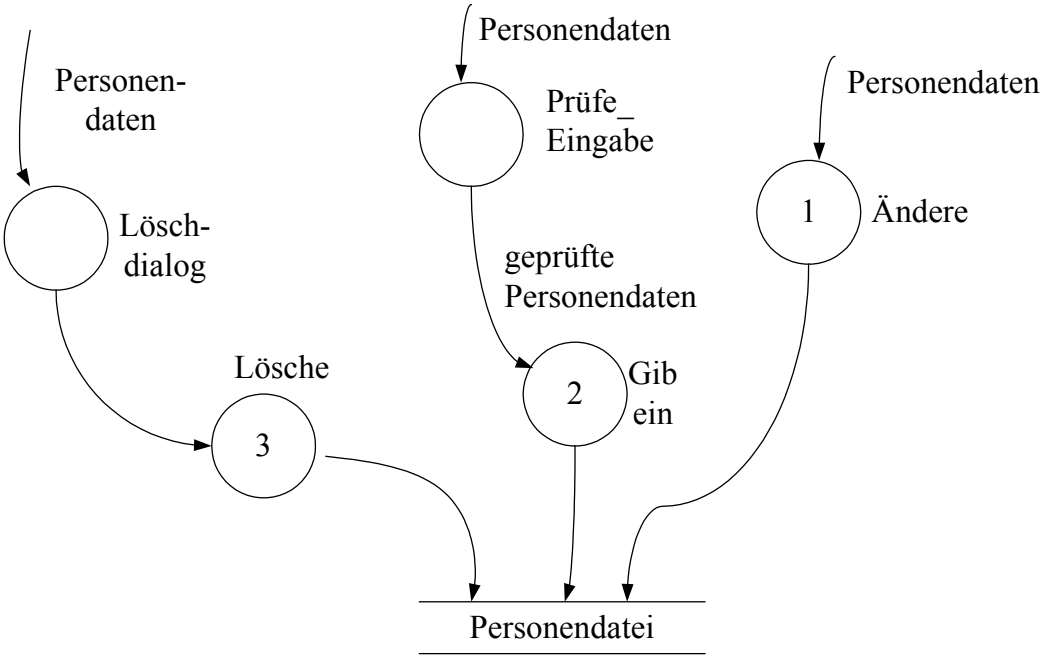


nach /Ja 92/

Gleiche Abstraktionsschicht auf gleichem Verfeinerungsniveau



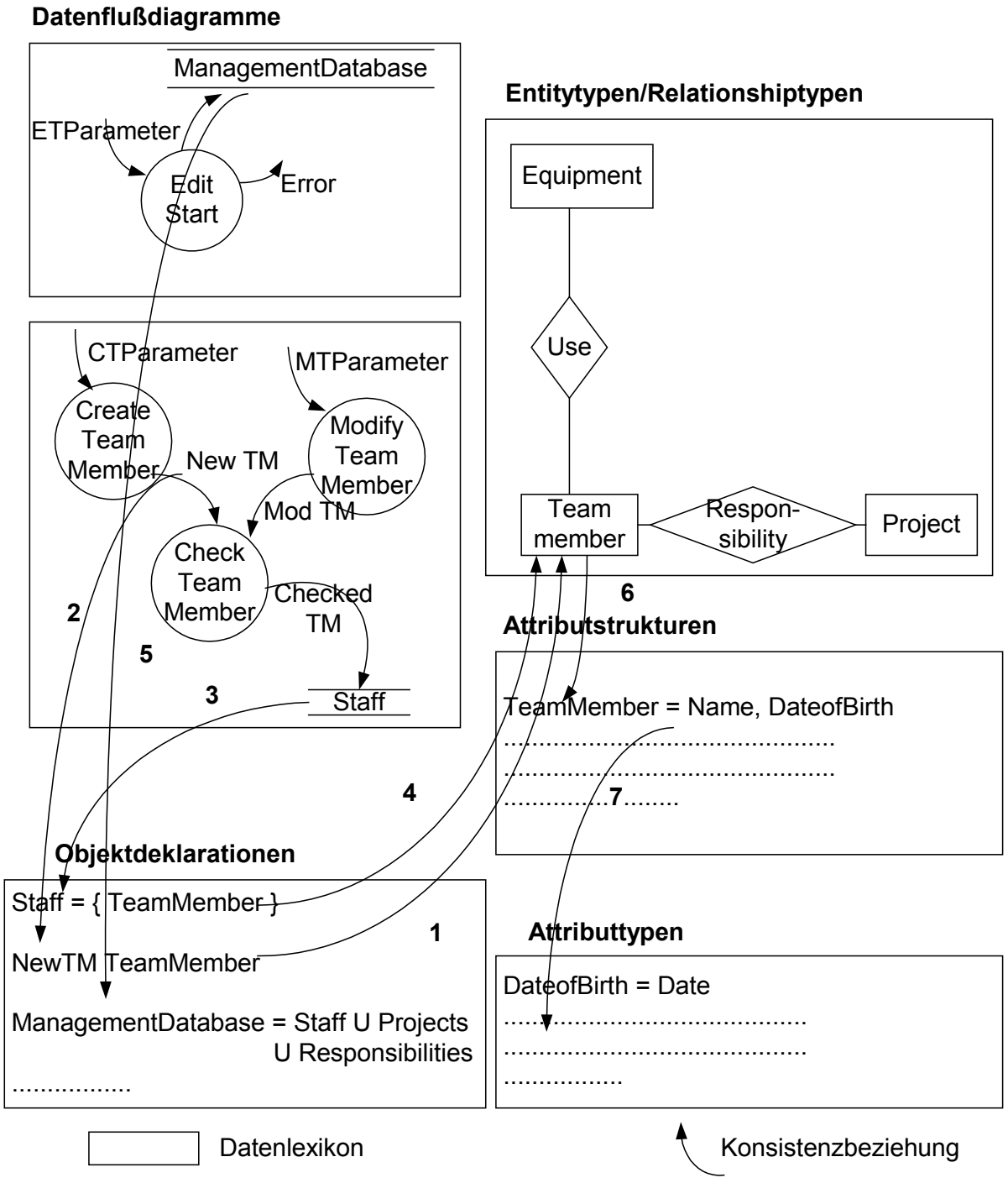
gleiches Verständnis über Wirkung eines Prozesses



Probleme:

- (1) hat Mischsemantik: Zugriffsoperation, Dialogverwaltung, Prüfung
- Zugriffsoperationen (2,3) entweder stets modellieren, oder abstr. Datenobjektverständnis des Datenspeichers
- Prüfe_Eingabe, Löschdialog sind Unterprozesse von ”Eingabe” oder ”Löschen”

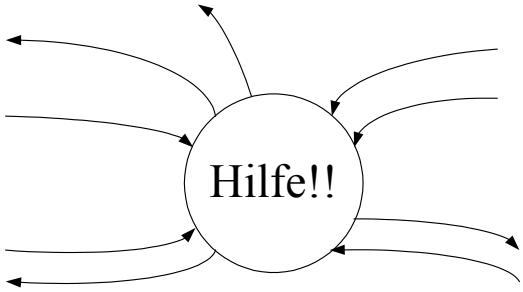
Mangelnde Integration durch Werkzeuge für RE



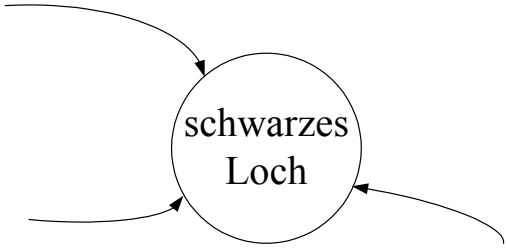
Konsistenzregeln zur Integration der beiden Teile der RE-Spache (nach /Co 90/)

Gute und schlechte DFDs; Beispiele 1

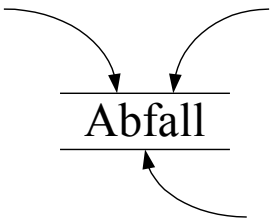
- zu komplexe Schnittstellen



- Informationssenken



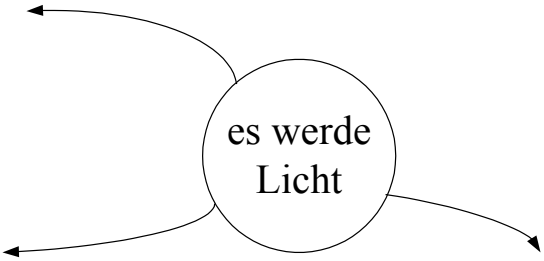
- Write-Only (oder Read-Only) Datenspeicher



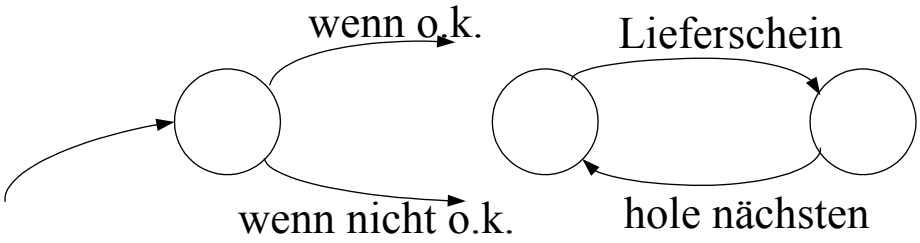
nach /Hr 90/

Gute und schlechte DFDs; Beispiele 2

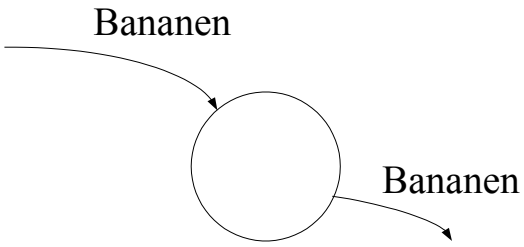
- wundersame Datenschaffung



- Anzeichen von Flußdiagrammen



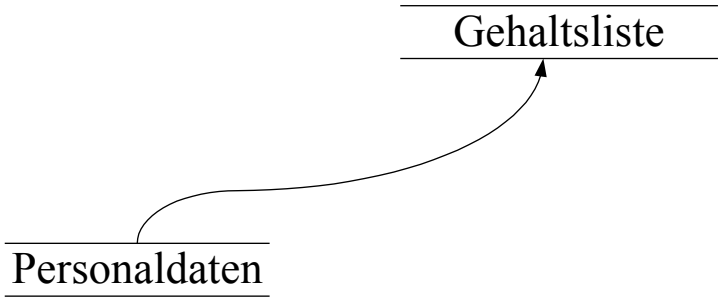
- Knoten ohne Funktion



nach /Hr 90/

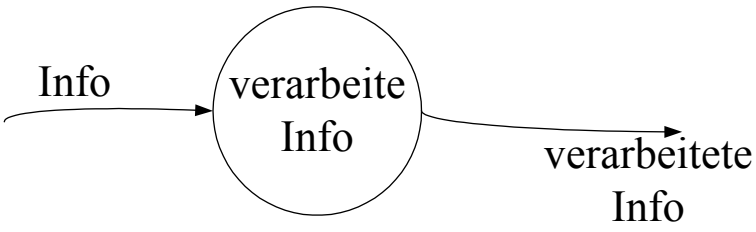
Gute und schlechte DFDs; Beispiele 3

- springende Daten



- Vorgänge außerhalb des Systems

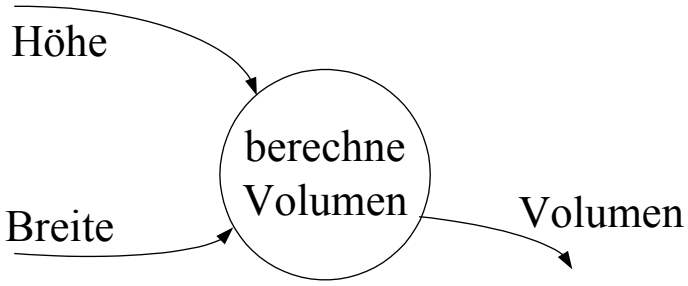
- nichtssagende Namen

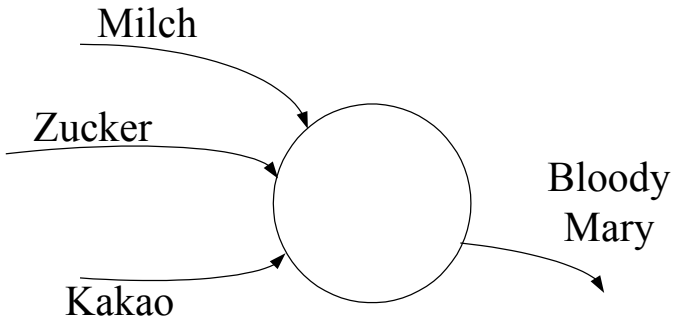


nach /Hr 90/

Gute und schlechte DFDs; Beispiele 4

- das Gesetz der Informationserhaltung

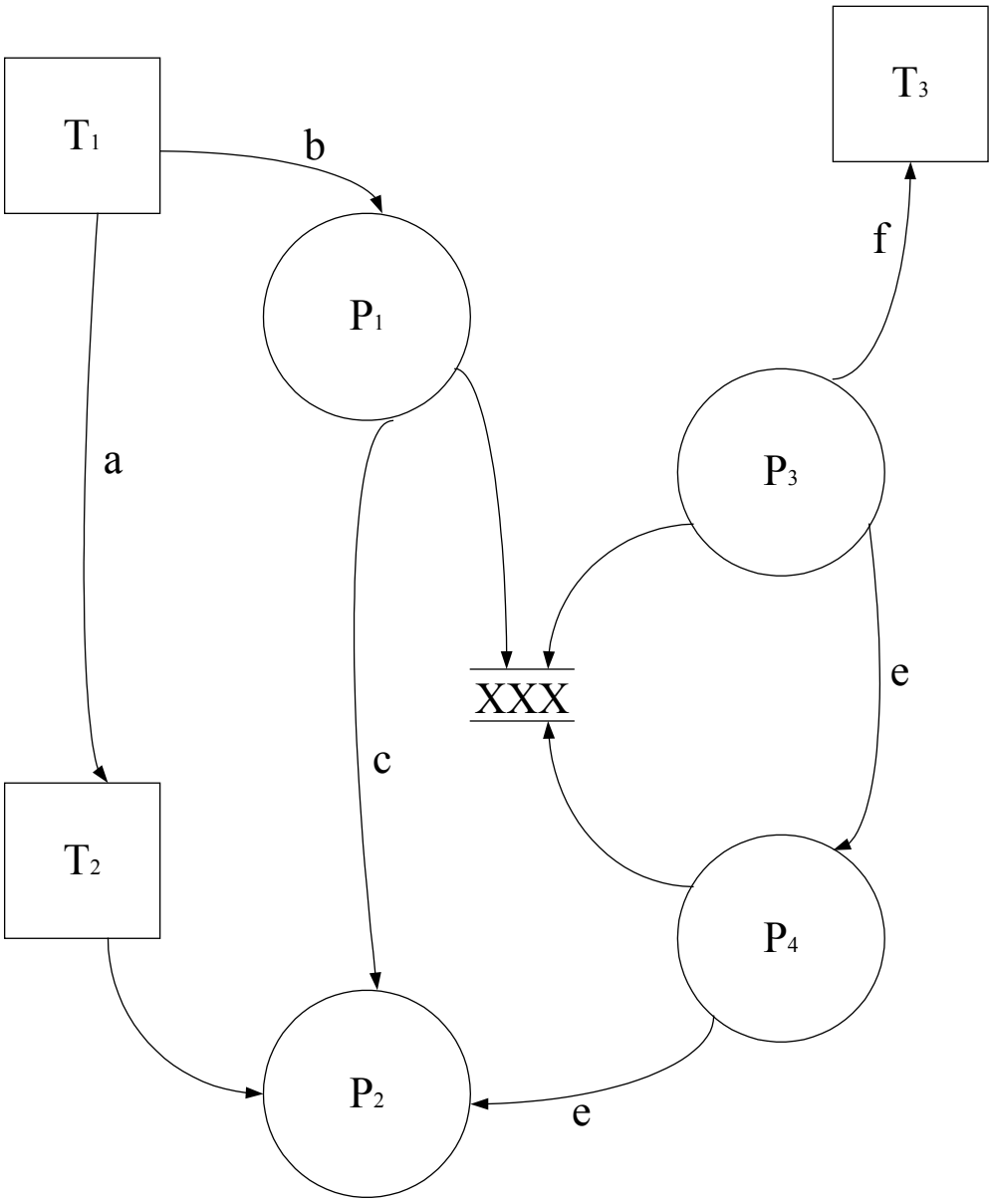




nach /Hr 90/

Zusammenfassung Datenflußdiagramme

Was ist hier falsch?



nach /Hr 90/

Ein Beispiel in SA-ER

Fallstudie Bibliothek

Eine Bibliothek soll zukünftig EDV-unterstützt arbeiten. Die dazu notwendigen Vorarbeiten bestehen u. a. darin, ein logisches Modell dieser Bibliothek zu entwickeln, d. h. ein ”formales” Modell als Hauptbestandteil der Anforderungsspezifikation zu erarbeiten.

Es sollen die Aufgaben, die bisher per Hand erledigt wurden, in einem Anforderungsmodell abgebildet werden, das zur Grundlage für eine EDV-Entwicklung dienen soll.

Aus der Sicht des Benutzers stellt sich die Bibliothek folgendermaßen dar:

- Die Bibliothek ist öffentlich, die Ausleihe ist kostenlos. Die Bücher stehen in Regalen, die den Benutzern frei zugänglich sind, außerdem gibt es einen umfangreichen Katalog, in dem alle Bücher nach Autor, Titel und Schlagworten sortiert sind.

- Jeder erfaßte Benutzer kann beliebig viele Bücher ausleihen und sie bis zu dreimal verlängern. Ein Benutzer wird dann von der Ausleihe vorübergehend ausgeschlossen, wenn er die Leihfristen um mehr als neun Wochen überschreitet, ohne um Verlängerung gebeten zu haben.

- Bücher, die im Moment ausgeliehen sind, können vorbestellt werden.

- Benutzer können Vorschläge für Neuanschaffungen machen.

Im einzelnen sollen drei Aspekte näher betrachtet werden, die Verwaltung der Bücher, die Verwaltung der Benutzer und die Verwaltung der Ausleihe.

- Jedes neue Buch wird hinsichtlich seiner Stammdaten erfaßt, ein bis drei Schlagworten zugeordnet, mit einer eindeutigen Nummer versehen, in den Katalog aufgenommen und ins Regal gestellt. Die Neuanschaffungen und Ersatzbeschaffungen werden über eine Buchhandlung abgewickelt.

- Hauptaufgabe der Benutzerverwaltung ist es, Namen und Anschrift neuer Benutzer zu erfassen und einen Ausweis mit einer eindeutigen Nummer zu erstellen. Natürlich können Benutzer auch gelöscht werden, allerdings nur dann, wenn sie alle ausgeliehenen Bücher zurückgegeben haben.

- Die Ausleihverwaltung registriert alle Ausleihvorgänge, außerdem werden hier die Rückgaben, Verlängerungen und Vorbestellungen erfaßt und bearbeitet. Vorbestellte Bücher werden bis zu drei Wochen nach Benachrichtigung der Benutzer aufbewahrt.

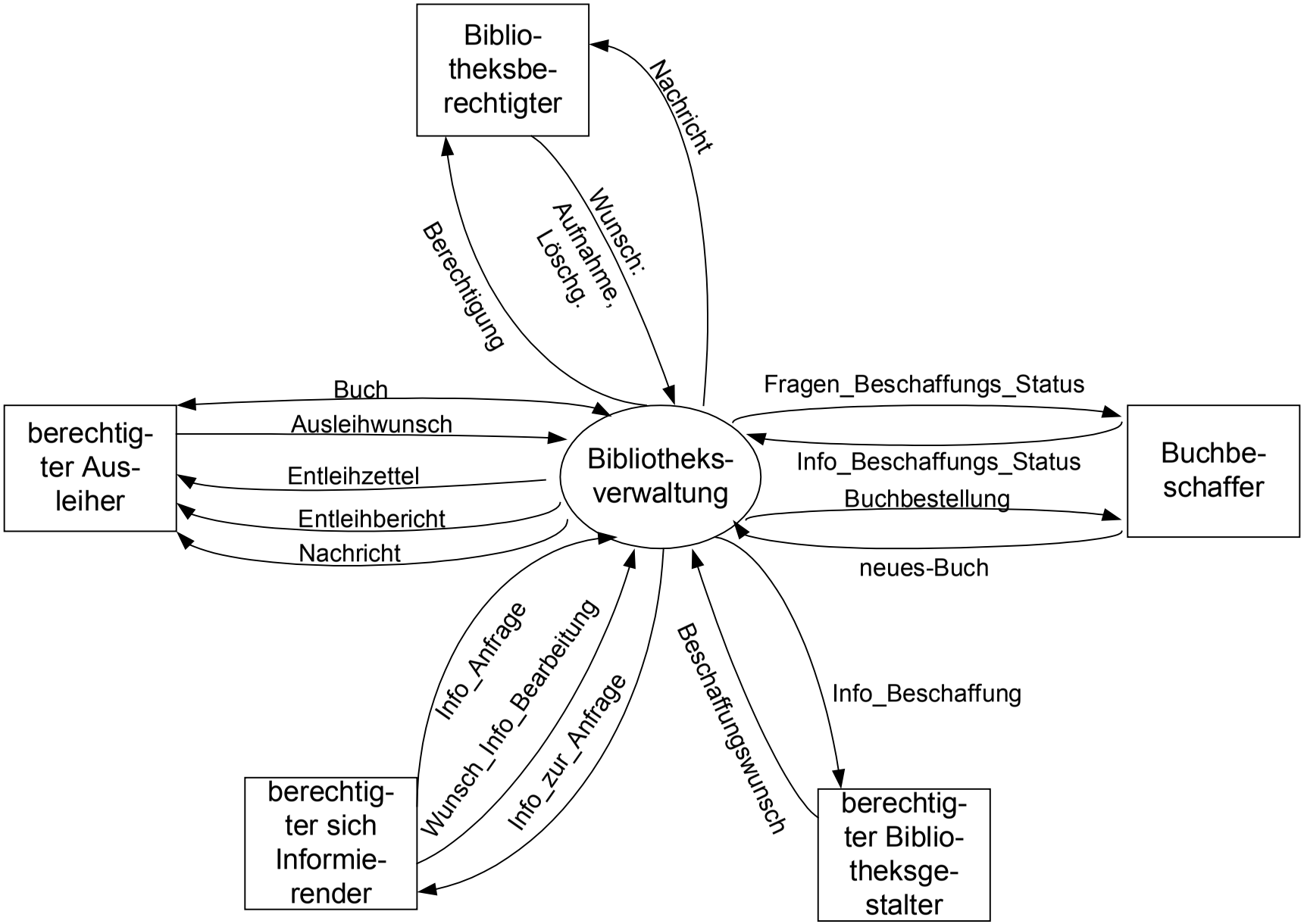
Was wird nicht modelliert?

- Verwaltung Bibliothekspersonal
- Beschaffungsstrategie
- Beschaffungsverwaltung (Überwachung der Lieferung, Bezahlung etc.)
- Verbindung in andere Bibliotheken (Fernleihe, allg. Infodienst (CD-ROM, Info-Server), Dienste anderer Bibliotheken, Nutzung einer virt. Bibliothek über Internet)
- Buchhaltung (Etatverwaltung)
- Buchtransport, Aufstellung, Heraussuchen, Magazin, etc.
- Katalogerstellung, Aktualisierung

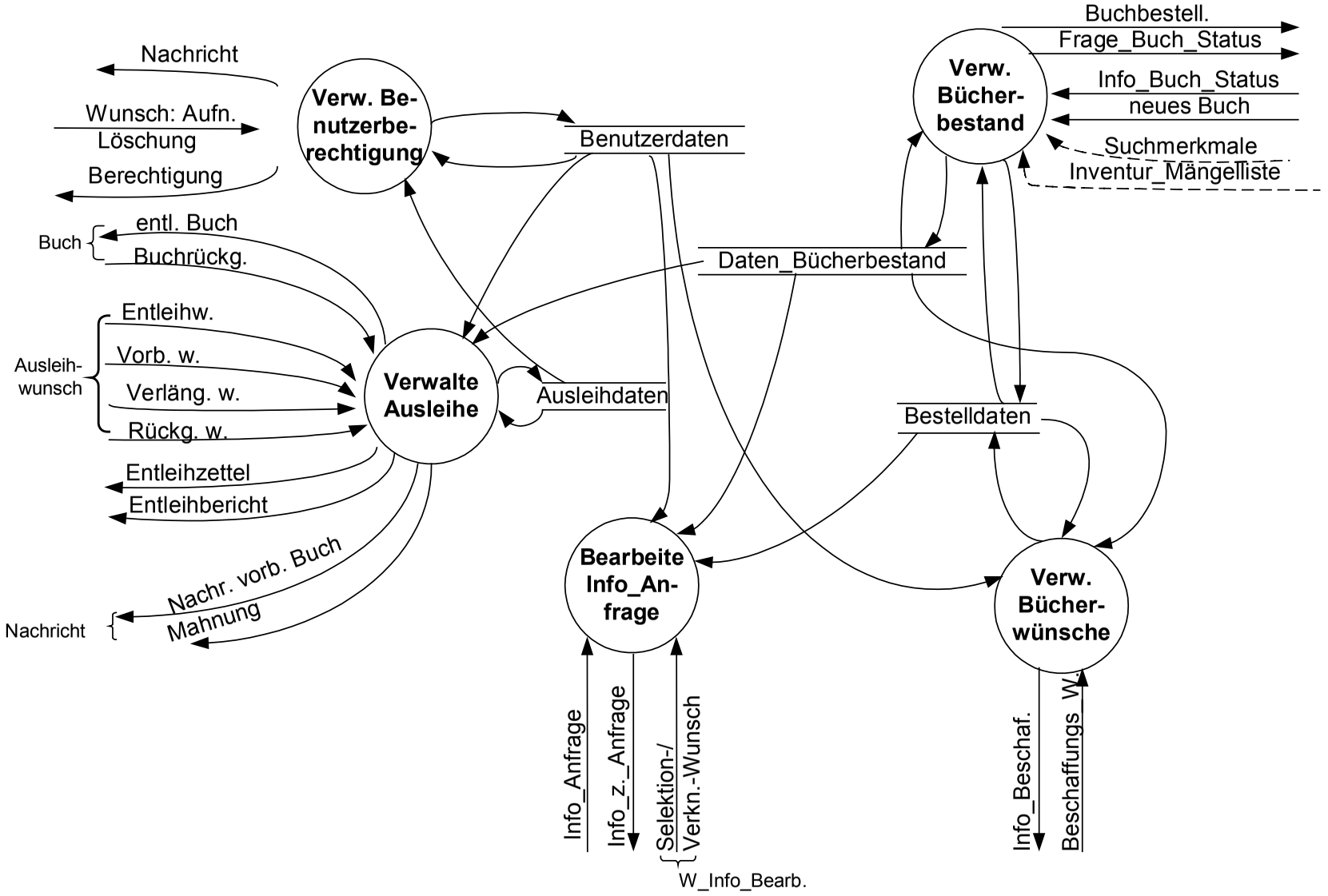
Wovon ist die Modellierung unabhängig?

- Bibliotheksprofil (wiss. Bibliothek, Freizeitbibliothek)
- Benutzer (Schüler, Studenten, Werksangehörige, Vereinsmitglieder, etc.)
- Größe der Bibliothek in gewissen Grenzen (Klein-, ..., Bereichsbibliothek)

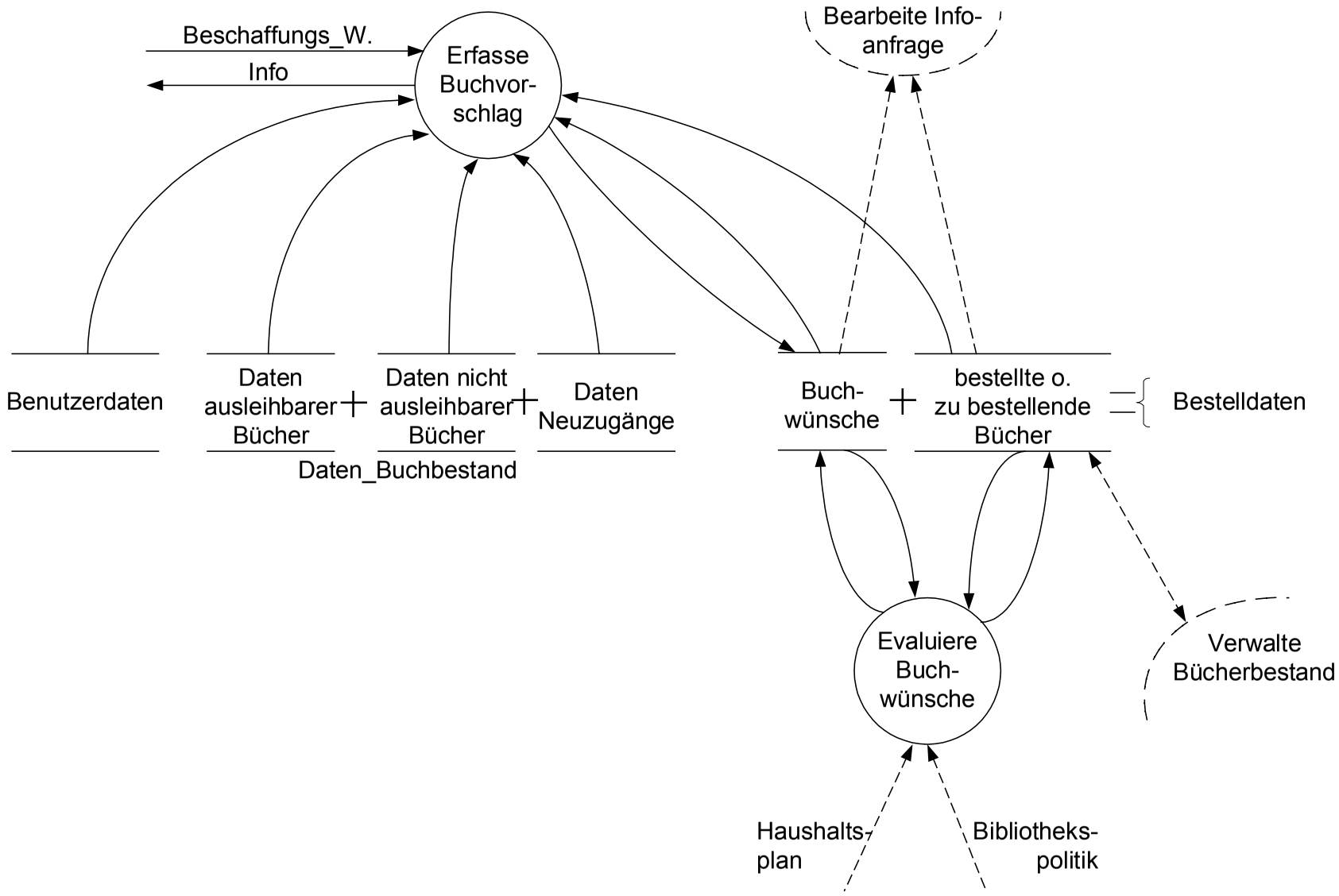
Funktionsmodell in SA Kontextebene: Ebene 0



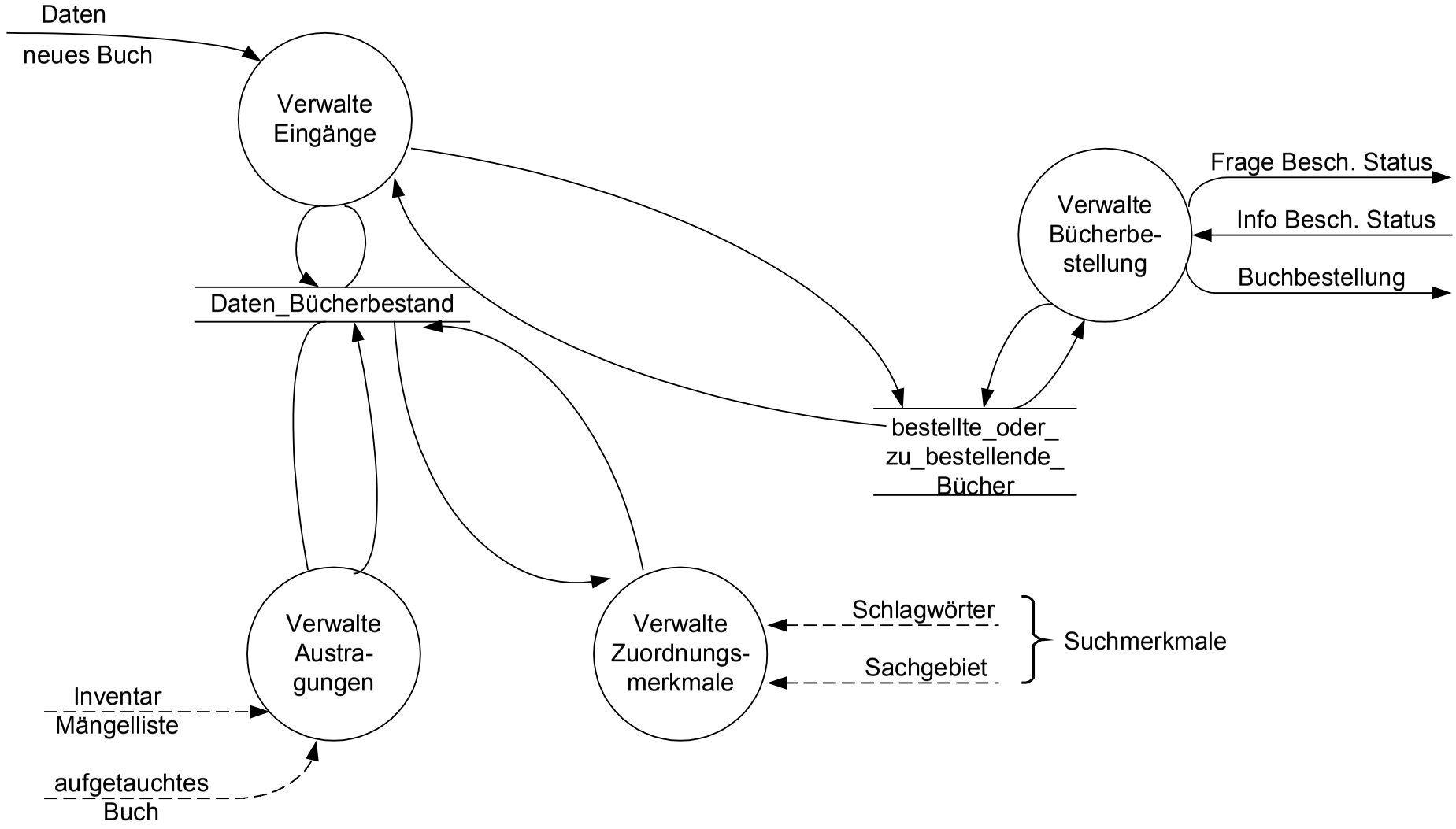
Bibliotheksverwaltung: Ebene 1



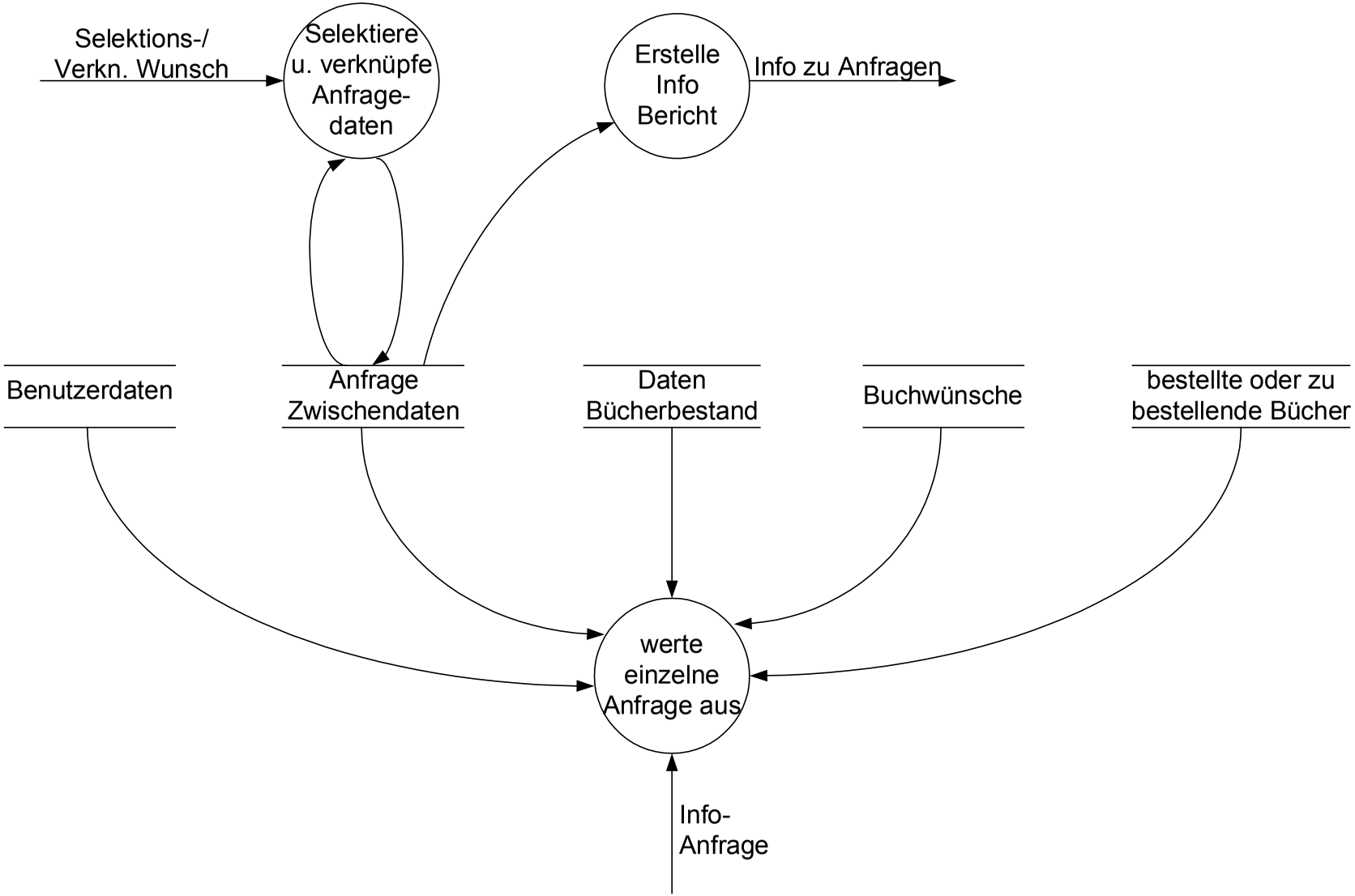
Verwalte Bücherwünsche: Ebene 2



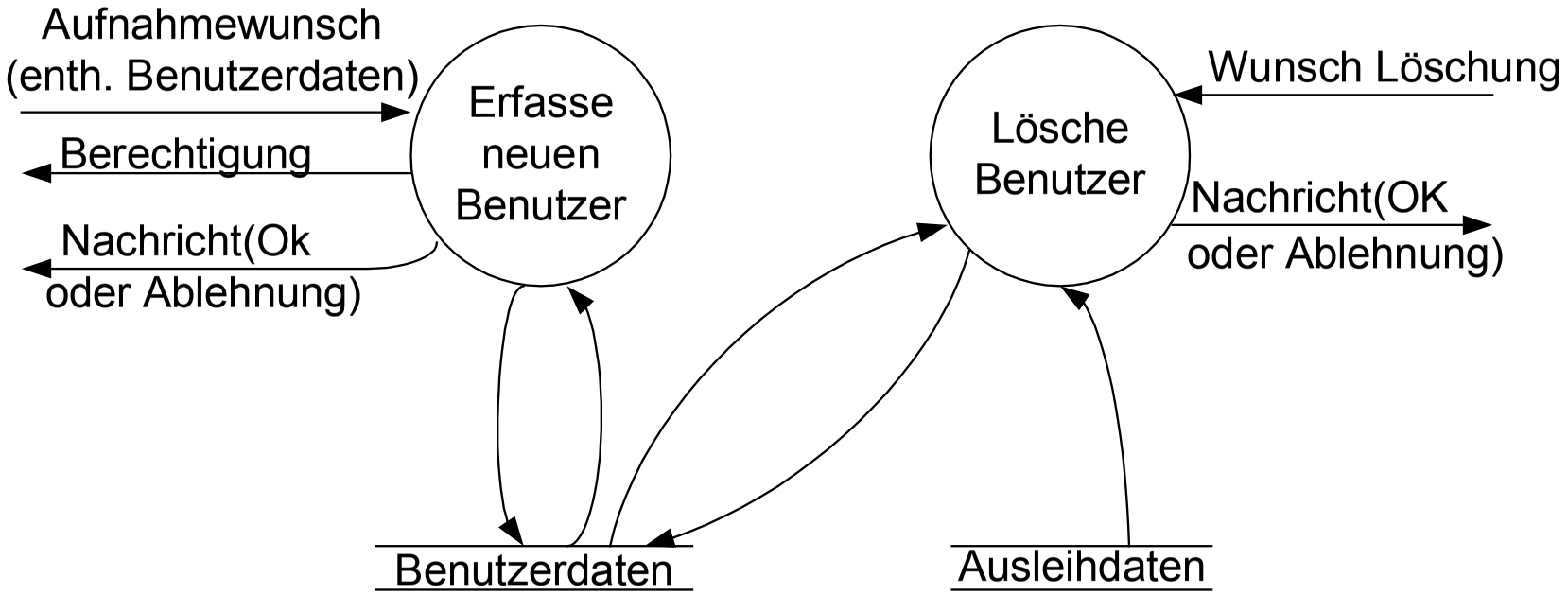
Verwalte Bücherbestand: Ebene 2



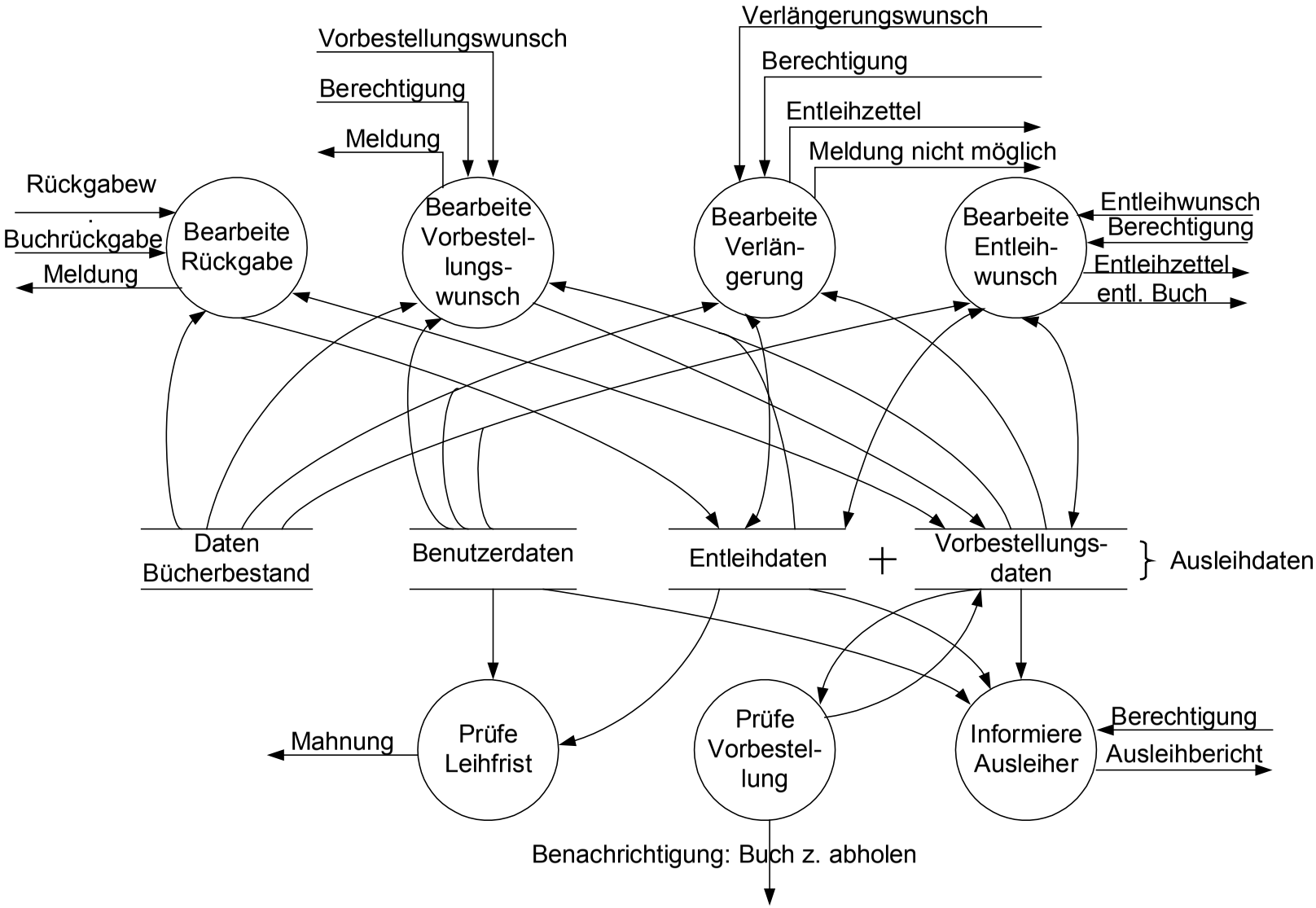
Bearbeite Info Anfrage: Ebene 2



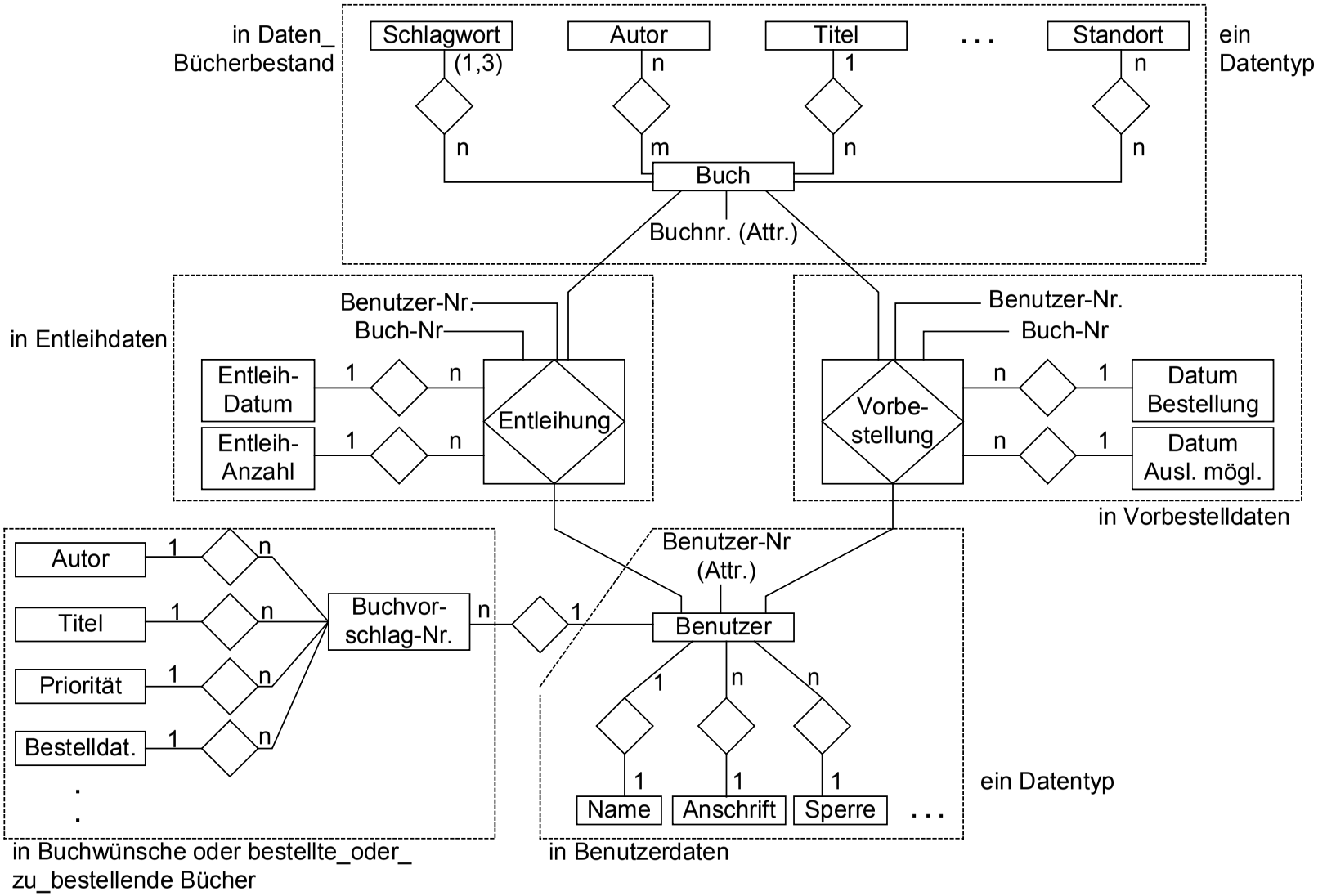
Verwalte Benutzerberechtigung: Ebene 2



Verwalte Ausleihe: Ebene 2



Informationsmodell in EER



Methodik-Regeln zur Modellierung mit SA/ER (nicht vollständig)

1) abstrakte, nach Funktionalität getrennte Terminatoren (nicht danach, ob sie auf einem Prozessor ablaufen), ggf. Terminatorverfeinerung
2) einzelfallorientierte Benutzerinteraktion (nicht Bücherstapel)
3) abstrakte Datenflüsse (nicht Realisierung) auf oberen Ebenen (z. B. Ausleihwunsch)
4) Beachtung: ”Was kann sich ändern?” auch bereits auf RE-Niveau führt zu abstrakterer Fassung, damit verbundener verbesserter Realisierung, Änderbarkeit von RE-Dokumenten
5) Erweiterungen im Kontext oder Systeminnenleben vorsehen
6) Einheitliche Semantik bei Verfeinerung: Summations- oder Implementierungssemantik, auf der ganzen Hierarchieebene
7) abstrakte, nach Funktionalität unterschiedene Datenspeicher (nicht ob sie gemeinsam realisiert werden)
8) genau Überlegung, auf welcher Ebene Datenspeicher in Teildatenspeicher aufgetrennt wird
9) viele eingehende Datenflüsse: kritische Beobachtung: liegt Asymmetrie vor, hängt diese mit Ausbauzustand des Systems zusammen?

10) Hinzudenken abstrakter Prozeßdefinition mit Formalparametern und Angabe einer Minispec für jeden Prozeß (auch nichtatomarer) liefert Erkenntnis, auf welchen Stufen der Hierarchie Datenflüsse verfeinert werden müssen (vgl. Prozeß Verwalte_Ausleihe)

11) Hinzudenken von Zugriffsoperationen zu Entity-Types bzw. von Navigationsoperationen für Relationship-Types

12) Sehen Datenspeicher als def. Vorkommen in Form eines abstrakten Datenobjekts vor: Hinzudenken abstrakter Zugriffsoperationen zu Datenspeichern

13) Auf unteren Ebenen treten neue Datenflüsse auf, wie Inventur_Mängelliste, die nach oben gezogen werden müssen (als abstrakter Datenfluß)

14) Asymmetrien (z. B. viele ein-/ausgehende Datenflüsse bei einem Prozeß, z. B. unterschiedliche Verfeinerungshierarchien bei Prozessen über viele oder wenige Stufen etc.) sind immer Grund für eine kritische Überprüfung: Wird das Modell bei Erweiterungen symmetrisch (einige Punkte werden einfach nicht betrachtet)? Habe ich falsch modelliert? Sind die Asymmetrien also berechtigt oder nur Ergebnis schlechter Modellierung?

15) Eine Frage ist, auf welcher Ebene Datenflüsse zu verfeinern sind. Ein Prozeß in SA in einem DFD ist ein angewandtes Vorkommnis. Wir deuten das Diagramm als definierendes Vorkommnis mit formalen Datenflüssen. Somit sind Balancierungen nach oben (formale werden zu aktuellen Datenflüssen) sowie nach unten (formale Datenflüsse werden in Diagramm ggfs. verfeinert) zu beachten. Die Verfeinerungsstufe üblicher DFD, ergibt sich jetzt daraus, auf welcher Ebene für die Realisierung eines Prozesses der Datenfluß verfeinert werden muß.

16) Bei der Modellierung tauchen auf unteren Ebenen Datenspeicher auf, die in abstrakterer Form nach oben gezogen werden müssen (bis auf das passende Niveau).

17) Betrachten Sie vorhandene Prozesse und überprüfen sie diese (eingefahrene Abläufe: Nicht alles was gegeben ist, ist falsch, die Nutzer müssen sich in dem neuen System auch wieder- und zurechtfinden!

18) ER_Modellierung: sorgfältiger Überlegung, was Struktur ist und was Attributstruktur und Werte sind.

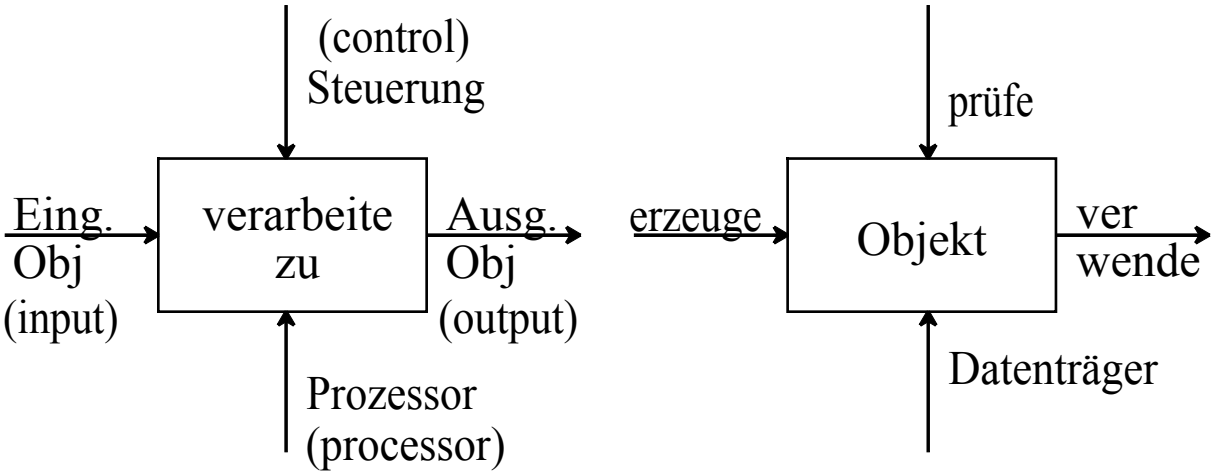
19) Ansonsten wären obige SA Regeln (jetzt angepaßt auf ER oder EER) zu wiederholen!

20) Überlegen: Was wird nicht modelliert? Wo in dem Modell wären entsprechende Erweiterungen (im Kontext oder Innenleben) vorzunehmen.

21) Welche Bestandteile des Anforderungsmodells sind wiederverwendbar, z. B. für eine Bibliothek ganz anderen Charakters (Großbibliothek, Fernausleihbibliothek, elektronische Bibliothek: Erhalt eines Buches heißt hier Zugriffsberechtigung auf den Datenbestand)

SADT (Structured Analysis and Design Technique) /Ro 85/

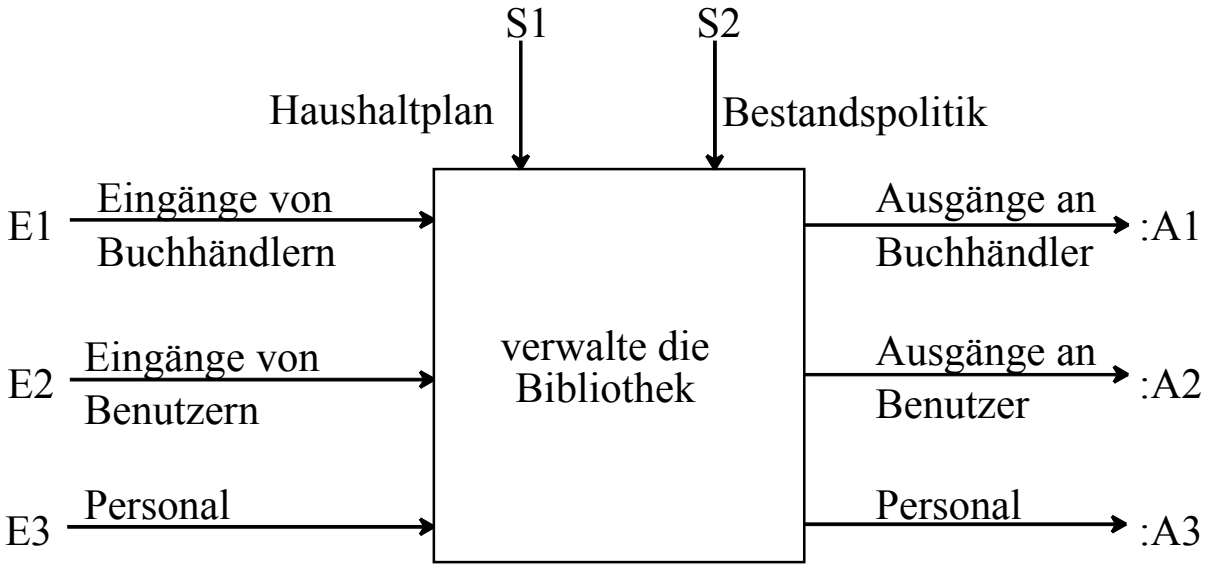
- Betrachtung eines Systems aus zwei Sichten:
- Abstraktionsstufen (Verfeinerung) ausdrückbar
- Zerlegung eines Teiles unabhängig von anderen Teilen der gleichen Abstraktionsstufe
  3 x Anzahl neuer Objekte x 6
- Jedes Modell zweifach darzustellen: funktionale Sicht
  Datensicht
  duale (redundante) Darstellung ermöglicht Vollständigkeits- und Konsistenzprüfungen
- keine Darstellung für Steuerfluß



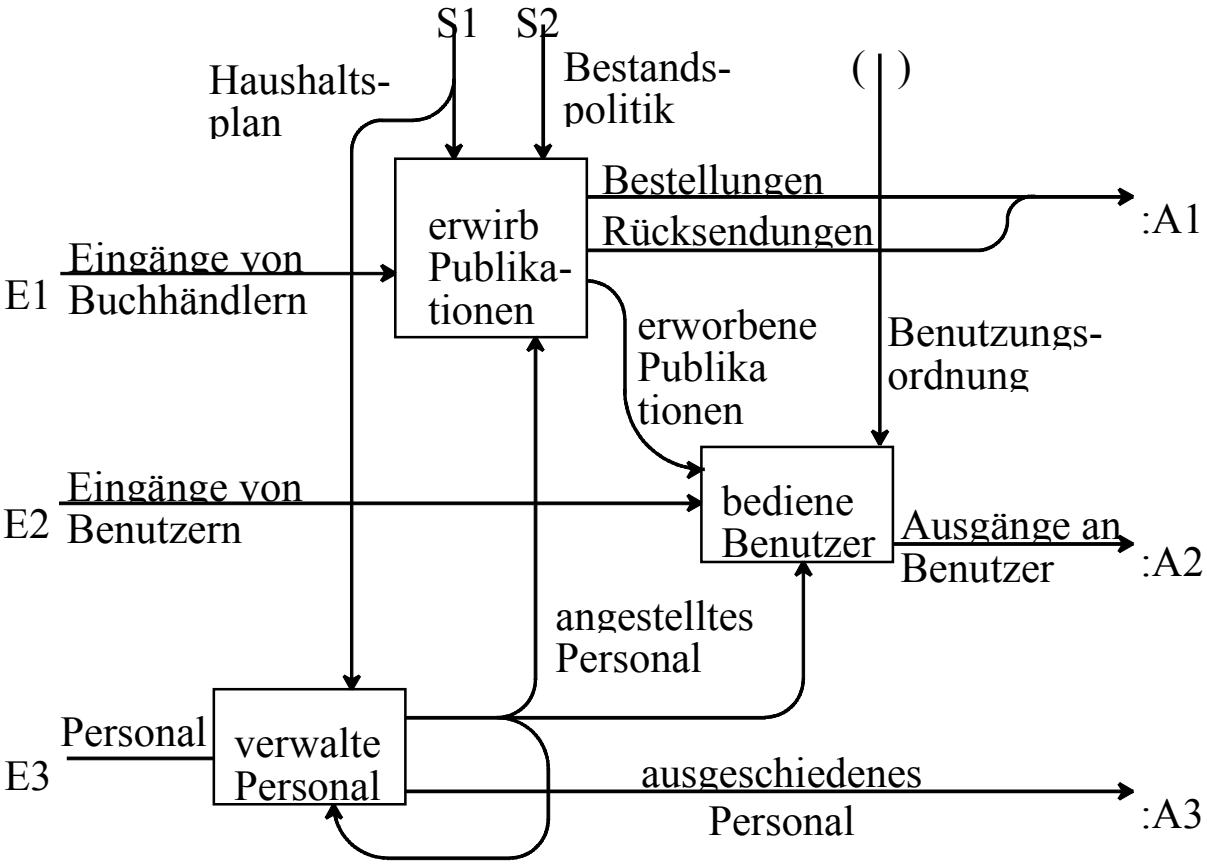
funktionale Sicht
(ICOP Objekte)

Datensicht
(Datenobjektsicht)

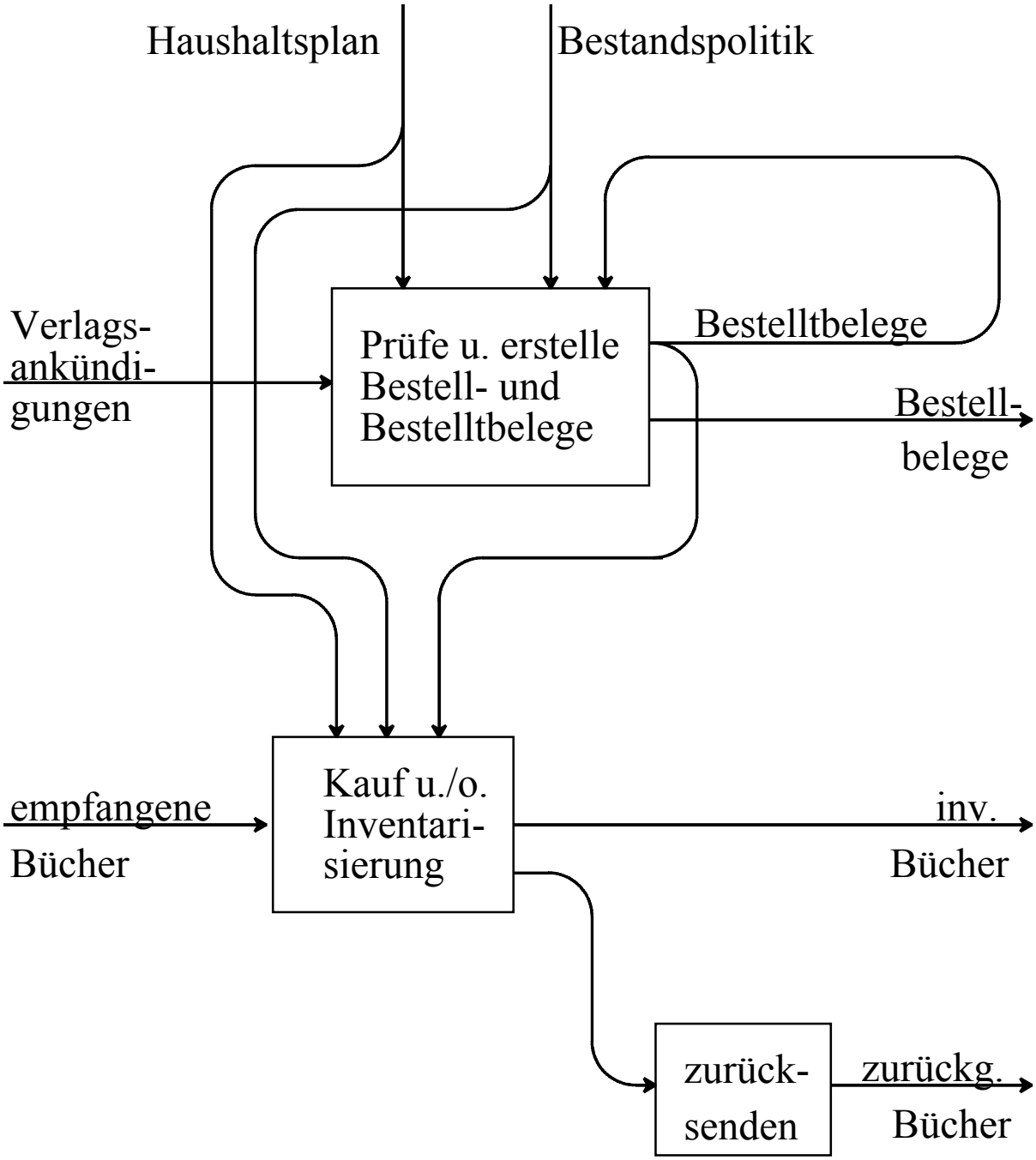
Beispiel (Verwaltung einer Bibliothek, Rohfassung)



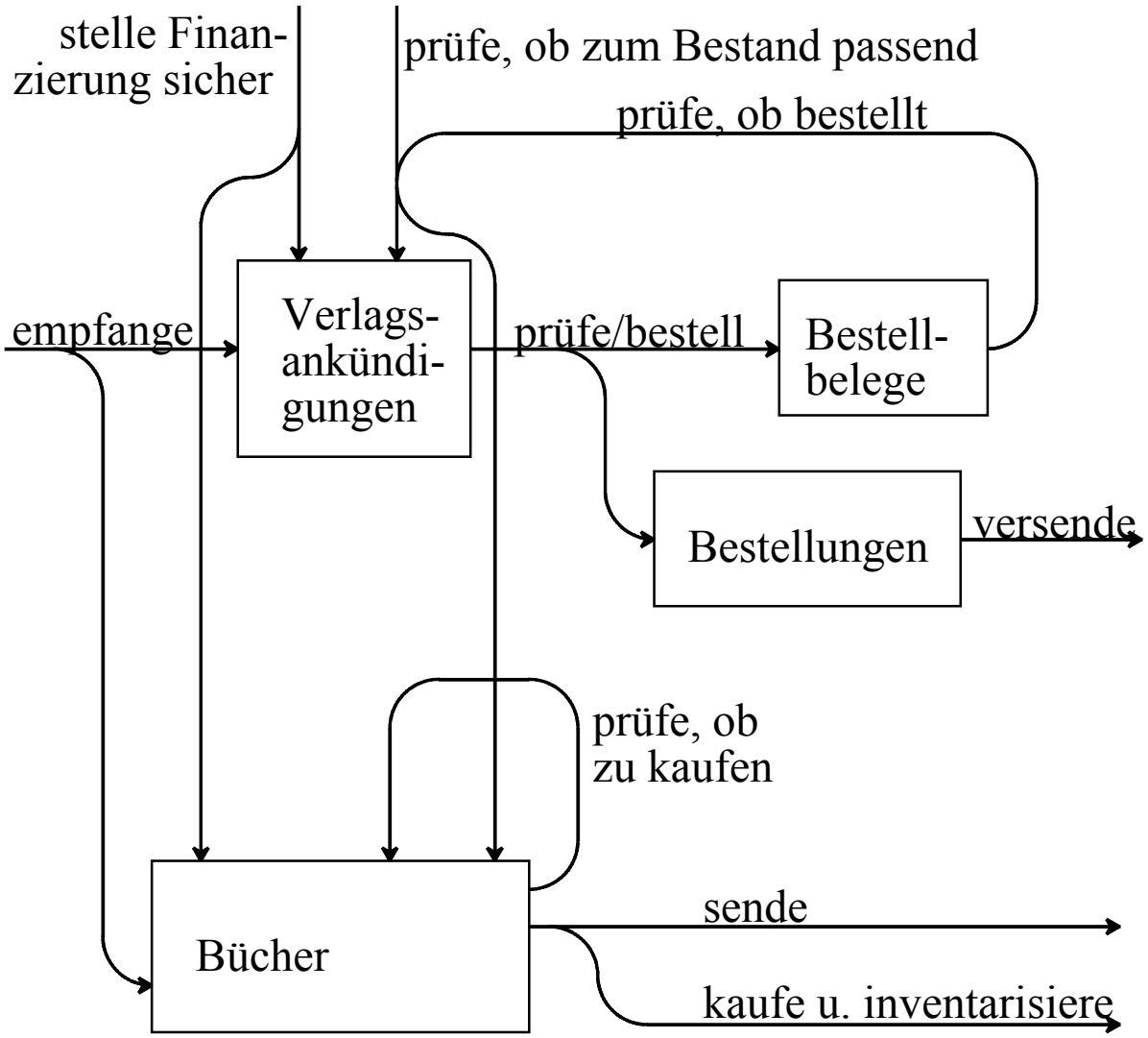
funktionale Darstellung: oberste Sicht f_0



funktionale Darstellung: f_1



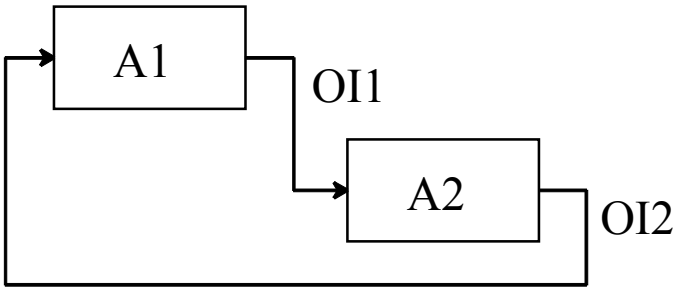
funktionale Sicht f_2: erwirb_Publikationen



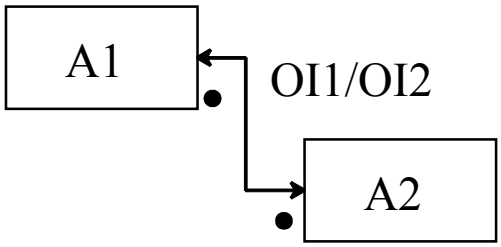
Ausschnitt datenorientierte Sicht d_2

Erweiterungen

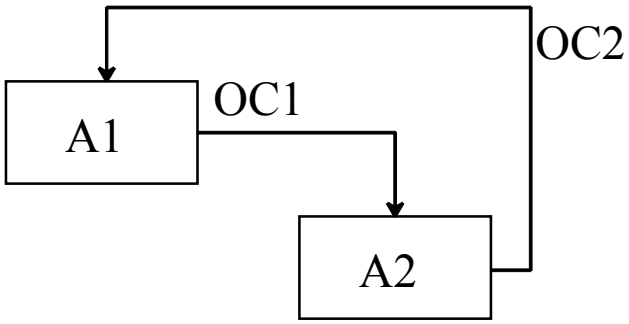
Iteration



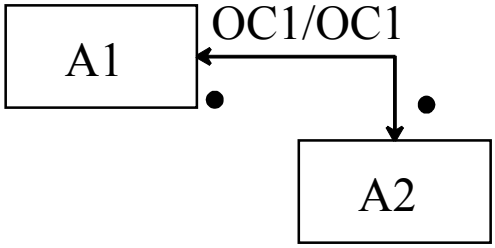
Kompaktform:



Rückkopplung:



Kompaktform:



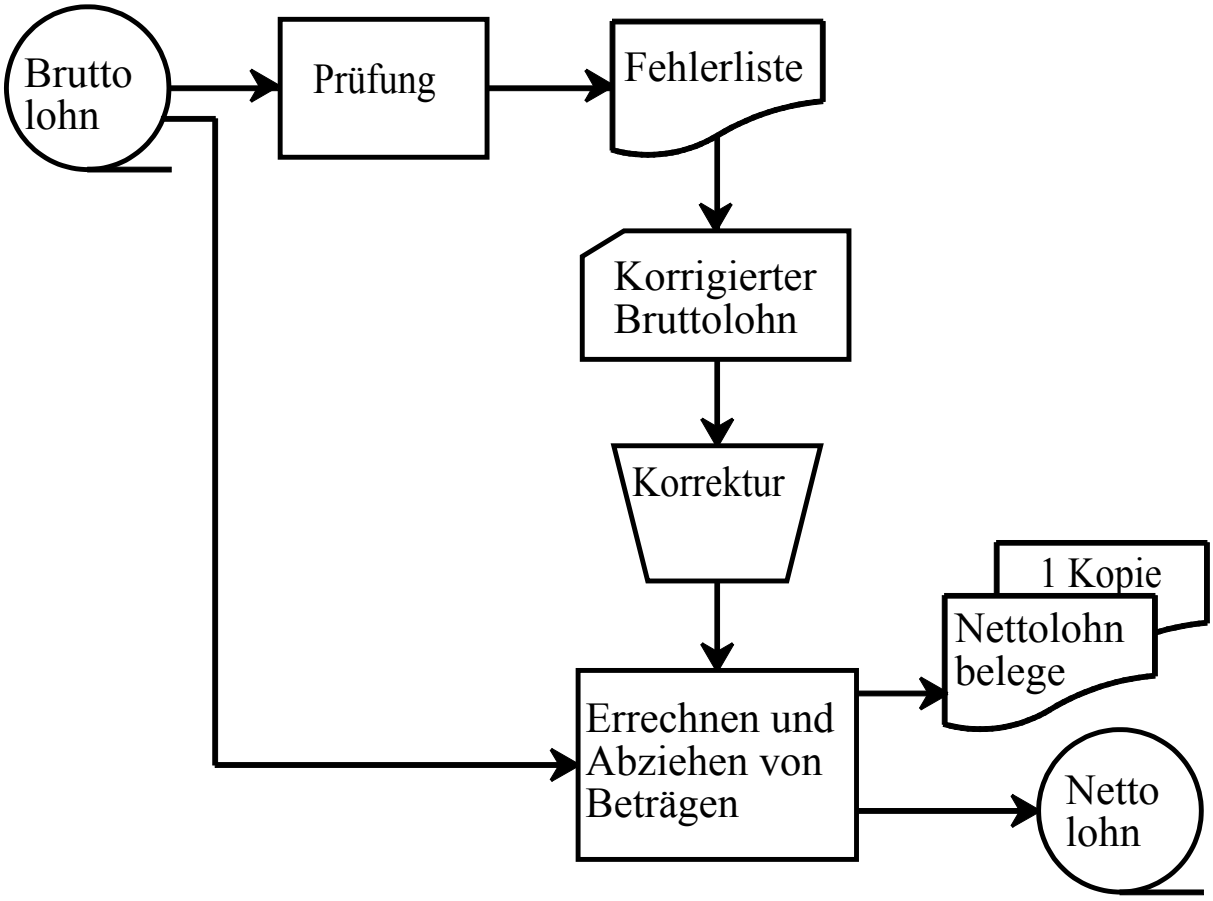
Historie

Datenflußpläne

— zur Darstellung von Bearbeitungsschritten
Datenträgern

- Kanten: Benutzen / Erzeugen von Daten

— bestimmte Symbole für best. Datenträger
für best. OP. z. B. Sortieren



Datenflußplan für eine Nettolohnabrechnung

- Abstraktion nicht gut darstellbar
- höchstens für Überblicke

Programmablaufpläne (Flußdiagramme)

- nur Operationen darstellbar
- spez. Symbole für EA, man. Tätigkeit, Unterprogr., Verzw., Zusammenführung (DIN-Norm)
  nur funktionale Abstraktion durch UPe
- keine Datenobjekte, Datenflüsse o. ä.
- $\Rightarrow$ für PIK, dort besser Struktogramme

Entscheidungstabellen

- für best. Probleme mit vielen Fallunterscheidungen
- Gegenargumente siehe Flußdiagramme

Regeln / Eingegangene Publikationen	1	2	3	4	5	6	7	8	9	sonst
Art des Eingangs	A	A	A	A	G	G	G	G	B	
In ”Bestellt”-Kartei registriert?	J		N	N	J		N	N	J	
Schon vorhanden?		J	N	N		J	N	N		
Paßt zum Bestand?			N	J			N	J		
Bestellen und in ”Bestellt”-Kartei registrieren										
Rechnung zur Bearbeitung weiterleiten				X					X	
Eintrag in ”Bestellt”-Kartei löschen									X	
Zurücksenden	X	X	X							
Aussondern (Einstampfen oder Verschenken)							X			
Zur Titelaufnahme weiterleiten				X	X			X	X	
An Sonderfallbearbeitung weiterleiten					X	X				X

Erläuterung der Abk.:
A zur Ansicht
B bestellt
G Geschenk
J Ja
N Nein

- einfache ET
- komplex ET
- erweiterte ET
- geschachtelte ET

ISDOS-Projekt (PSL/PSA)

Wertung:

- von hierarchischer Datenbankwelt geprägt
- PSA spiegelt Batch-Werkzeugwelt wieder
- von Modellierungselementen her reichhaltig
- PSL: Problem Statement Language
  (Istanalyse, Sollkonzept: Miniwelt)
- PSA: Problem Statement Analyzer für PSL-Beschreibungen
  - Konsistenzprüfung
  - Abspeicherung in Datenbank
  - Generierung von Berichten f. untersch. Perspektiven

Die Sprache PSL

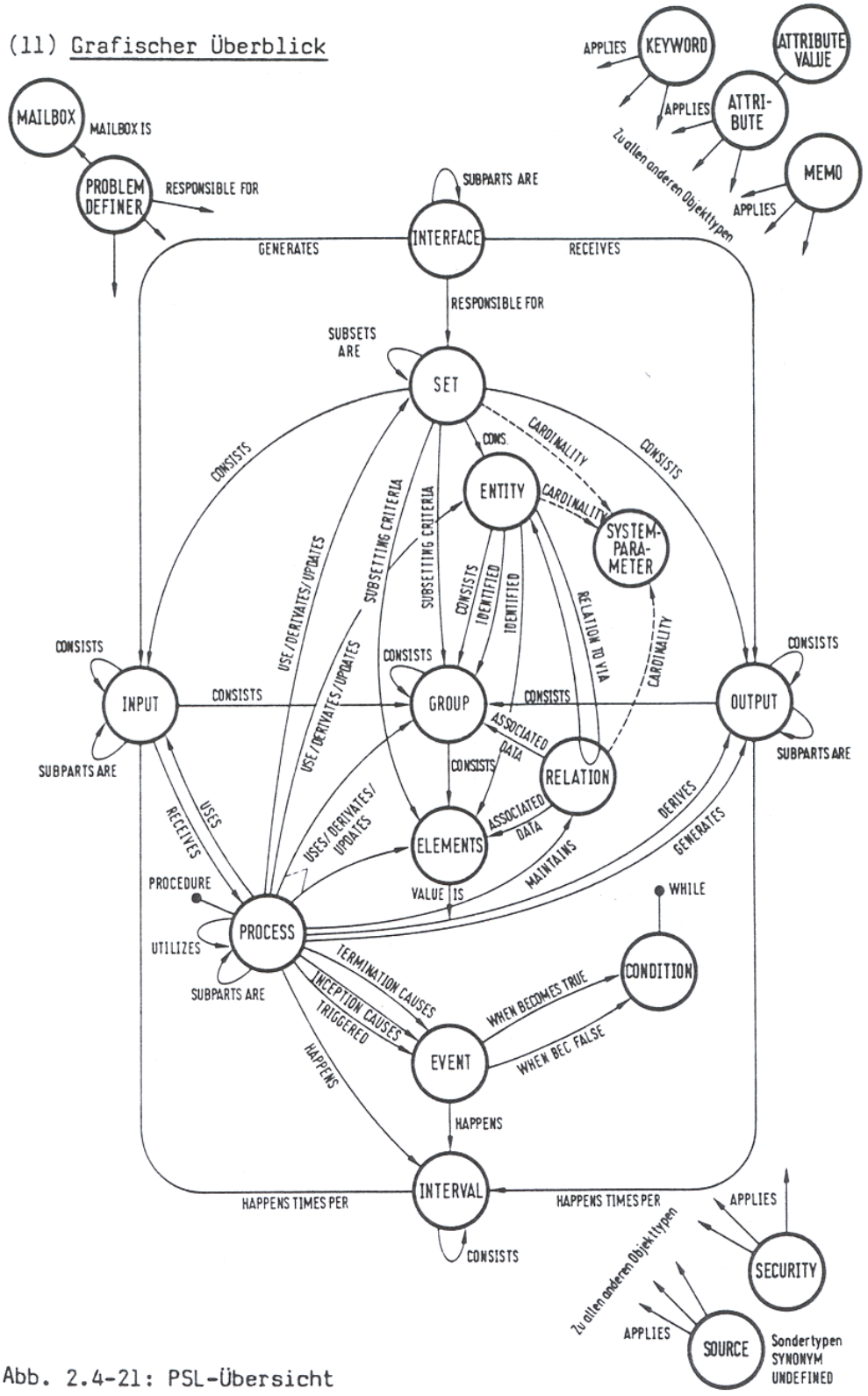


Abb. 2.4-21: PSL-Übersicht

11 Balzert, Software-Systeme

Aufgaben

1) Die in diesem Kapitel vorgestellten Notationen stammen aus dem Bereich der kommerziellen Systeme und dort hauptsächlich aus dem Bereich der interaktiven Systeme oberhalb einer Datenbank (Informationssysteme). Betrachten wir als anderes Beispiel ein Transformationssystem in Form eines Compilers. Die Struktur der Ein-/Ausgabedaten ist zu ”primitiv” für ein ER-Datenmodell. Der Berechnungsteil eines händisch erstellten Compilers kann mit SA formuliert werden. Ein Kontrollteil ist kaum nötig. Für die Formulierung fortgeschrittener Compileralgorithmen ist SA kaum geeignet. Andererseits sind die Möglichkeiten der Datendefinition und ihres Zusammenspiels mit der Verarbeitung für das vorliegende Problem unbefriedigend (Grammatikfestlegung auf unterschiedlichen Ebenen, Semantikfestlegung auf unterschiedlichen Ebenen, enges Zusammenspiel zwischen Verarbeitung und Datenstrukturdefinition). Diskutieren Sie diesen Sachverhalt an einem kleinen Compilerbeispiel.
2) Für Transformationssysteme auf einfachen Ein-/Ausgabestrukturen eignet sich JSP, was wir später kennenlernen werden. JSP ist weniger für die Analyse als für die Konstruktion vorgesehen, ”eignet sich” aber auch für die Analyse. Der Vorteil ist der, daß das Zusammenspiel von Verarbeitung und Datenstrukturdefinition deutlich wird. Betrachten Sie ein JSP-Beispiel und diskutieren Sie, inwieweit sich dieser Ansatz für die Anforderungsspezifikation eignet.

3) Die Notationen dieses Kapitels sind für technische Anwendungen nur bedingt geeignet. Betrachten Sie dazu eine Festigkeitsberechnung aus dem Bereich der Ingenieuranwendungen (z. B. Crash-Test bei Automobilen). Kontrollmodellanteil im Sinne von Reaktivität gibt es hier nicht. Was sind die zugrundeliegenden Datenstrukturen (Gitter) und Berechnungsverfahren (Differential- bzw. Differenzengleichungen). Warum können die Anforderungen an ein solches Softwaresystem mit SA/IM schlecht ausgedrückt werden?
4) Für Realzeitsysteme wurde SA/RT entwickelt (/HP 87; WM 85/). Hier werden Kontrollprozesse eingeführt, deren Innenleben wird durch endliche Automaten oder durch verallgemeinerte endliche Automaten /Ha 87/ beschrieben. Zusätzliche Kontrollflüsse dienen der Modellierung reaktiver Systeme. Modellieren Sie ein kleines Beispiel mit einer der Notationen /HP 87; WM 85/.
5) Fassen Sie in diesem Kapitel angesprochenen Notationserweiterungen bzw. Methodikregeln für SA/ER zusammen: Was haben wir bezüglich Terminatoren, Prozessen, Entity- oder Relationshiptypen, Verfeinerung, definierendes Vorkommnis von Prozessen, ER-Typen, Datenspeichern, angewandten Vorkommnissen bzw. vorangehender Benutzbarkeit kennengelernt? Was haben wir an Methodikregeln kennengelernt?

Literatur zu Kap. 5:

/Ba 89/ C. Batini (Ed.): Entity-Relationship-Approach, North Holland, 1989

/Be 95/ M. von der Beeck: Ein Kontrollmodel für die Strukturierte Analyse, Dissertation RWTH Aachen, 1995

/Bo 94/ G. Booch: Object-oriented Analysis and Design with Applications, 2nd Ed., Benjamin Cummings, 1994

/BR 96/ G. Booch/J. Rumbaugh: Unified Method for Object-oriented Development, Documentation Set Version 0.8

/Ch 83/ P. P. Chen: Entity Relationship Approach to Information Modelling and Analysis, North Holland, 1983

/Co 90/ S. Coors: Entwurf einer Integrierten Requirements-Engineering-Umgebung, Diplomarbeit RWTH Aachen, 1990

/Ja 92/ Th. Janning, Requirements Engineering und Programmieren im Großen, Dissertation RWTH Aachen, Deutscher Universitätsverlag, 1992

/DeM 79/ T. DeMarco: Strucured Analysis and System Specification, Prenctice Hall, 1979

/Ha 87/ D. Harel et al.: Statecharts: A Visual Formalism for Complex Systems, Sci. Comp. Progr., 8, 231-274, 1987

/HP 87/ D. J. Hatley/I. A. Pirbhai: Strategies for Real-Time Systems, Dorset House, 1987

/Hr 90/ P. Hruschka: Requirements Engineering, Skript einer Vorlesung an der RWTH Aachen, 1990

/IE 77/ IEEE Transactions on Software Engineering (Sonderheft), Vol. SE-3, Januar 77

/IE 85/ IEEE Computer (Sonderheft über Requirements Engineering), Computer, April 85

/Ko 96/ Ch. Kohring: Ausführung von Anforderungsdefinitionen zum Rapid-Prototyping, Dissertation RWTH Aachen, Shaker Verlag, 1996

/Le 95/ M. Lefering: Integrationswerkzeuge in einer Software-Entwicklungsumgebung, Dissertation RWTH Aachen, Shaker Verlag, 1995

/MP 84/ S. M. McMenamin/J. F. Palmer: Essential System Analysis, Yourdon, 1984

/Pa 91/ H. Partsch: Requirements Engineering, München: Oldenburg Verlag, 1991

/RB 91/ J. Rumbaugh, M. Blaha, W. Premerlani, F. Edely, W. Lorenzen: Object-oriented Modelleling and Design, Prentice Hall, 1991

/Ro 85/ D. T. Ross: Applications and Extensions of SADT, Computer, April 85, 25-34

/TH 77/ D. Teichroew/E. A. Hershey: PSL/PSA: A Computer-aided Technique for Structured Documentation and Analysis of Information Processing Systems, TOSE 3, 1, 41-48, 1977

/WM 85/ P. T. Ward/S. J. Mellor: Structured Development for Real-Time Systems, Vol. 1, 2, 3, Yourdon, 1985

/Yo 89/ E. Yourdon: Modern Structured Analysis, Yourdon Press, 1979