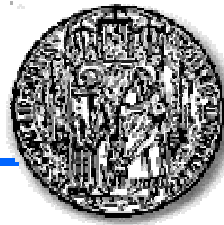


Vorlesung Softwaretechnik

Prof. Dr.-Ing.habil. Dipl.Math. Klaus-Peter Fährnich
(Wintersemester 2001/2002)

Universität Leipzig
Institut für Informatik





4. Die Definitionsphase

- 1. Funktionale Sicht**
 - 1. Funktionsbaum**
 - 2. Geschäftsprozess**
 - 3. Datenflussdiagramme**
 - 4. Zusammenfassung**
- 2. OO-Sicht**
 - 1. Objektorientierte Konzepte**
 - 2. Zusammenfassung**



4.1. Die Definitionsphase – Funktionale Sicht

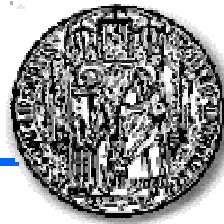
1. Konzepte Funktionsbaum, Geschäftsprozess und Datenflussdiagramm
2. Hierarchische Gliederung von Funktionen und Konstruktion eines Funktionsbaumes für eine gegebene Problemstellung
3. Checklisten
4. Strukturierung von Geschäftsprozessen
5. Ermittlung von Schnittstellen, Funktionen, Speicher und Informationsfluss für gegebene Problemstellung
6. Regeln für die Syntax und Semantik von DFD (Datenflussdiagramme)

4.1. Die Definitionsphase – Funktionsbaum



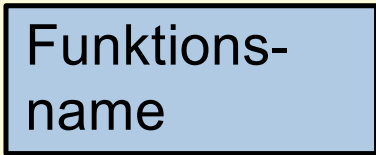
Konzepte und Sichten Alternative Notationen häufig verwendet sellen verwendet											
Funktionsbaum	Geschäftsprozess (1987)	Datenflussdiagramm 1966	Data Dictionary 1979	ER (Entity Relationship) 1976	Klassendiagramm 1980/ 1990	Pseudocode	Struktogramm 1973	PAP (Programmablaufplan) 1966	ET (Entscheidungstabelle) 1957	Aktivitätsdiagramm 1997	Kollaborationsdiagramm
Funktionale Hierarchie	Arbeitsablauf	Informationsfluss	Daten-Strukturen	Entitätstypen & Beziehungen	Klassen-Strukturen	Kontroll-Strukturen		wenn-dann-Strukturen	Endlicher Automat	Nebenläufige Strukturen	Interaktions-Strukturen
Funktionale Sicht			Datenorientierte Sicht		Objektorientierte Sicht	Algorithmische Sicht	Regelbasierte Sicht	Zustandsorientierte Sicht		Szenariobasierte Sicht	
LE 5			LE 8		LE 6-7	LE 9	LE 9-10	LE 11-12		LE 7	

Basiskonzepte : Funktionale Sicht



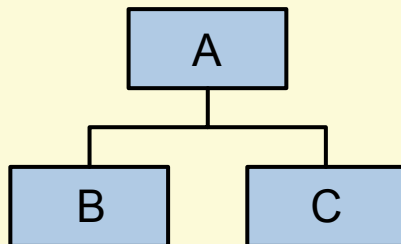
4.1.1. Die Definitionsphase – Funktionsbaum

- **Funktion** beschreibt Tätigkeit oder klar umrissene Aufgabe innerhalb eines größeren Zusammenhangs.
- Funktionshierarchie entsteht wenn eine allgemeine Funktion in spezielle Teilfunktionen gegliedert wird. Es entsteht ein **Funktionsbaum**.



Funktions-
name

Grafische Darstellung



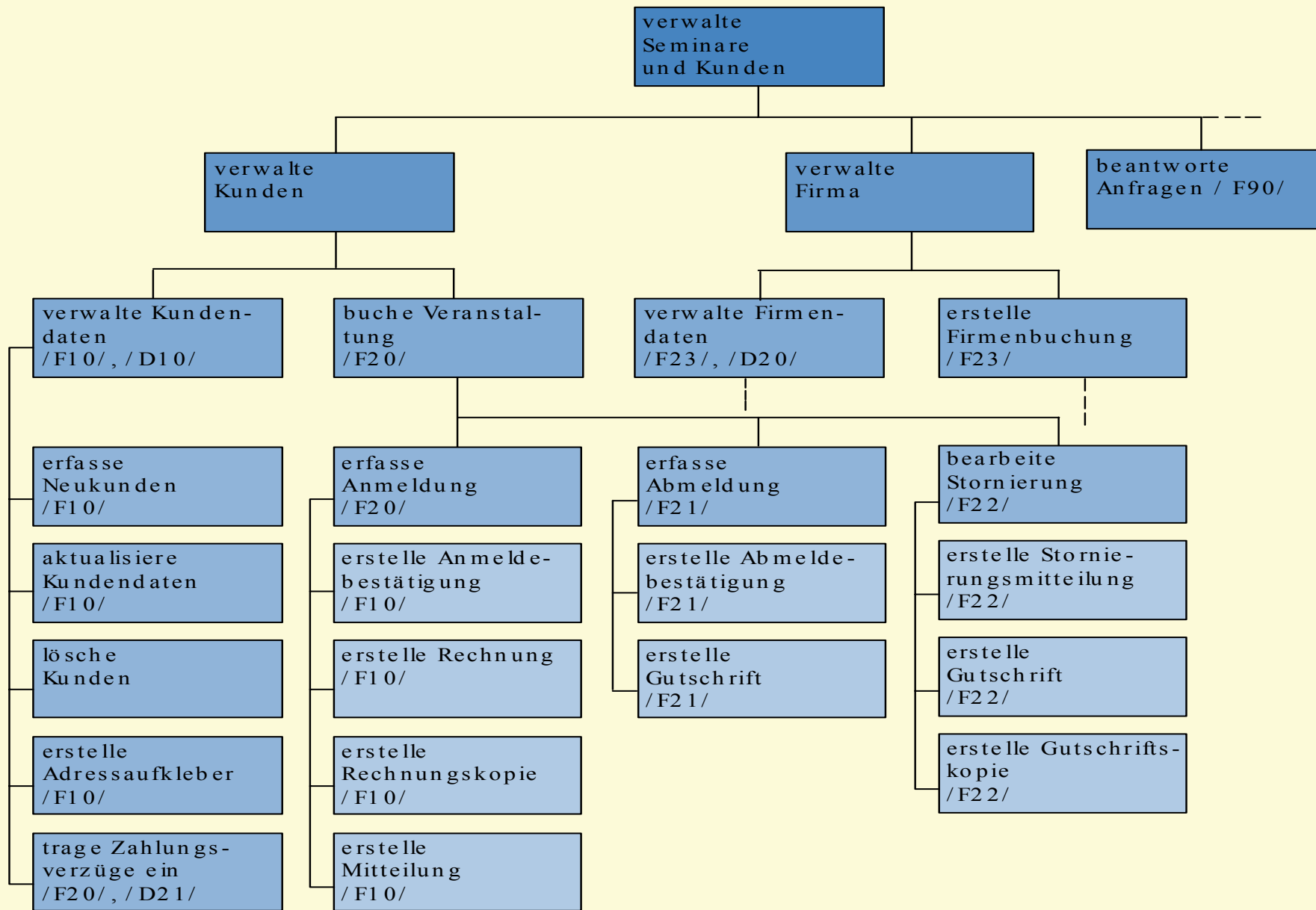
Hierarchie

Basis der Hierarchie

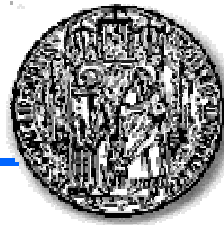
- **Besteht-aus** (meist in der Definitionsphase):
A besteht aus B und C
- **Ruft-auf** (meist in der Entwurfsphase):
A ruft B und C auf.

Erstellungsregeln:

- Unter einer gemeinsamen Vaterfunktion sollen nur Funktionen angeordnet sein, die fachlich eng zusammengehörende Tätigkeiten beschreiben
- Was eng zusammengehört kann nur mit Fachwissen entschieden werden
- Auf einer Hierarchieebene sollen Funktionen angeordnet sein, die sich auf gleichem Abstraktionsniveau befinden



Funktionsbaum einer Kundenverwaltung



4.1.2. Die Definitionsphase – Geschäftsprozesse

Geschäftsprozesse im Großen

- Ablauforganisation eines Unternehmens
- Unternehmensprozess (business process)
 - Besteht aus einer Anzahl von unternehmensinternen Aktivitäten, die durchgeführt werden, um die Wünsche eines Kunden zu befriedigen

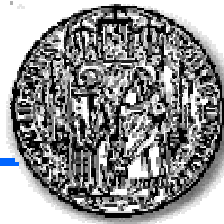
Akteure

- Rollen von Menschen oder Systeme, insbesondere Computersysteme, die als externe Beteiligte mit einem Unternehmen (Kunden) oder einem Software-Produkt (Benutzer) kommunizieren und Daten austauschen

Geschäftsprozesse im Kleinen

- Funktionalität eines Produkts
- Definiert einen Arbeitsablauf, der mit Hilfe von Software durchgeführt wird, aber manuelle und organisatorische Anteile besitzen kann
- **use case**
 - Teil eines Geschäftsprozesses, der die Benutzerkommunikation mit dem Software-System beschreibt
- Geschäftsprozess (*use case*) (Arbeitsablauf)
 - besteht aus mehreren zusammenhängenden Aufgaben, die von einem Akteur durchgeführt werden, um ein Ziel zu erreichen bzw. ein gewünschtes Ergebnis zu erstellen.

Wichtige Begriffe

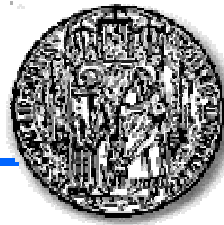


4.1.2. Die Definitionsphase – Geschäftsprozesse

Geschäftsprozess: Name (was wird getan?)

- **Ziel:** Globale Zielsetzung bei erfolgreicher Ausführung des Geschäftsprozesses
- **Kategorie:** primär, sekundär oder optional
- **Vorbedingung:** Erwarteter Zustand, bevor der Geschäftsprozess beginnt
- **Nachbedingung Erfolg:** Erwarteter Zustand nach erfolgreicher Ausführung des Geschäftsprozesses
- **Nachbedingung Fehlschlag:** Erwarteter Zustand, wenn das Ziel nicht erreicht werden kann
- **Akteure:** Rollen von Personen oder andere Systeme, die den Geschäftsprozess auslösen oder daran beteiligt sind.
- **Auslösendes Ereignis:** Wenn dieses Ereignis eintritt, dann wird der Geschäftsprozess initiiert
- **Beschreibung:**
 - o 1 Erste Aktion
 - o 2 Zweite Aktion
- **Erweiterungen:**
 - o 1a Erweiterung des Funktionsumfangs der ersten Aktion
- **Alternativen:**
 - o 1a Alternative Ausführung der ersten Aktion
 - o 1b Weitere Alternative zur ersten Aktion

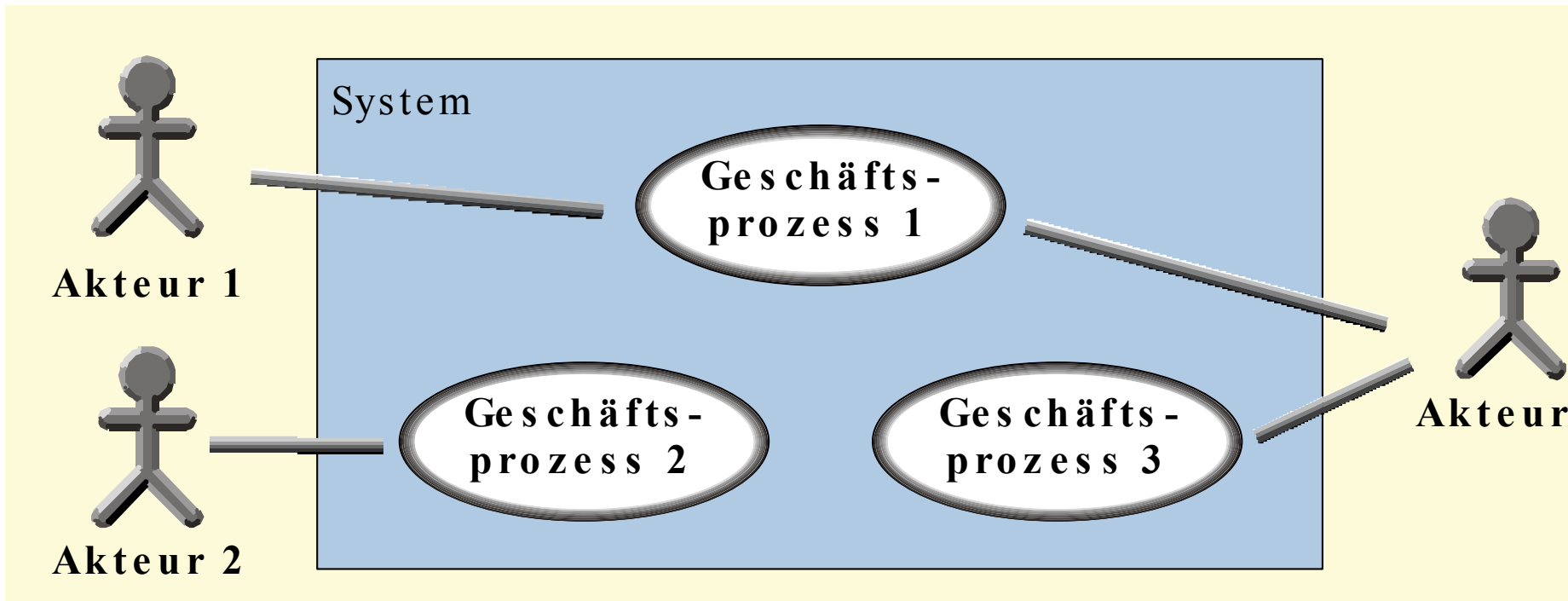
Schablone für Geschäftsprozesse



4.1.2. Die Definitionsphase – Geschäftsprozesse

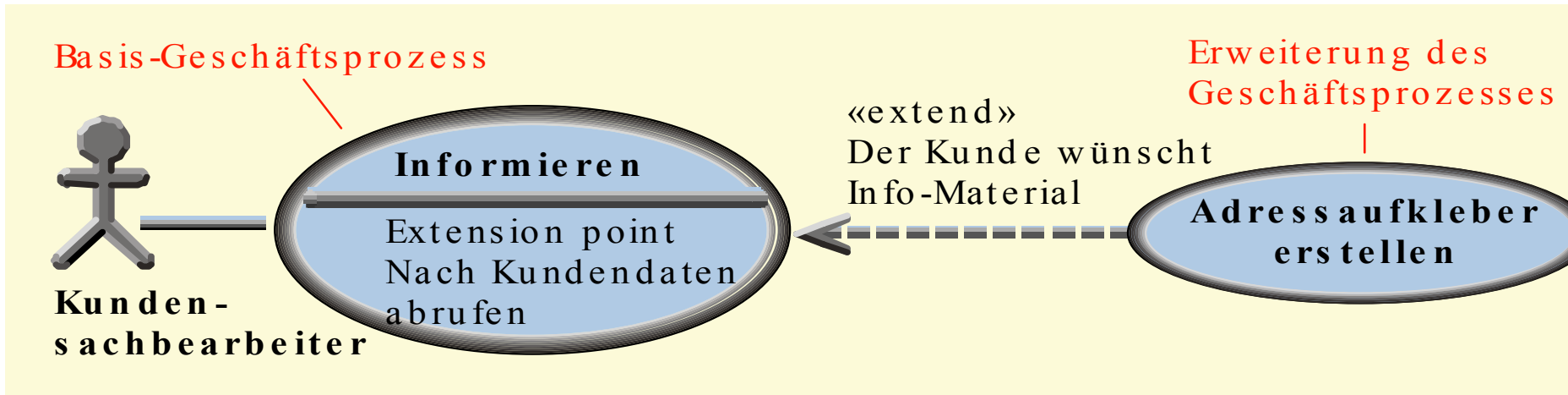
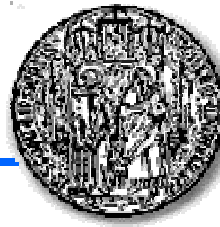
Geschäftsprozessdiagramm (use case diagram) :

- Beschreibt das Zusammenspiel mehrerer Geschäftsprozesse untereinander und mit den Akteuren
- In UML gibt es 3 Möglichkeiten, Geschäftsprozesse zu strukturieren : die *extend*-Beziehung, die *include*-Beziehung und die Generalisierungsbeziehung

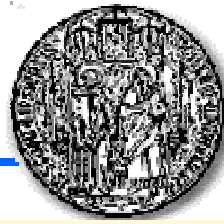


Notationen für ein Geschäftsprozessdiagramm

4.1.2. Die Definitionsphase – Geschäftsprozesse



Mache Prozesse besitzen die gleichen Unterprozesse. Um es nicht doppelt zu beschreiben wird als selbständiger Prozess modelliert. Mehrere Prozesse können ihn dann verwenden



4.1.2. Die Definitionsphase – Geschäftsprozesse

Include-Beziehung



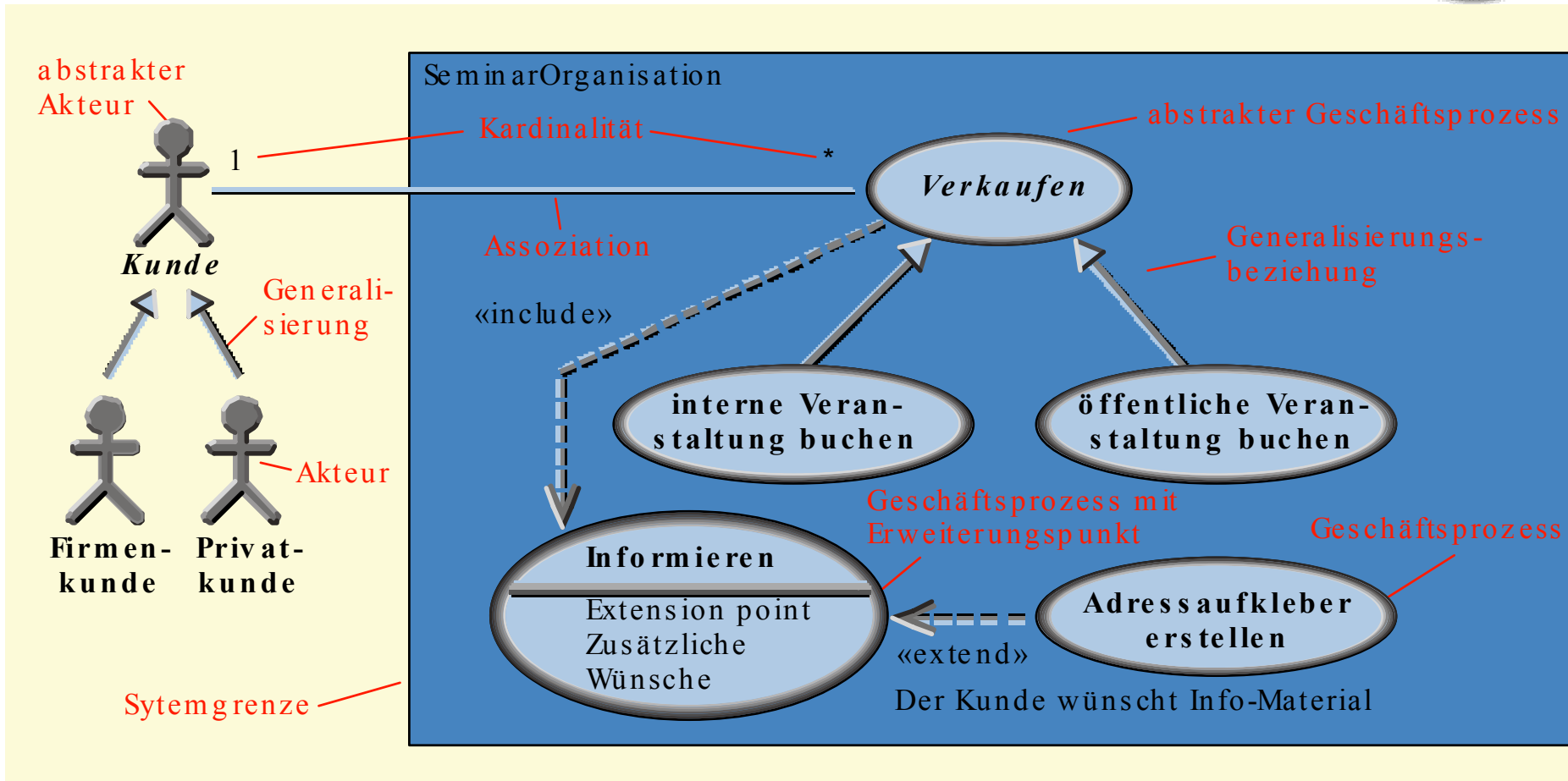
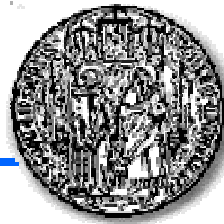
Generalisierung



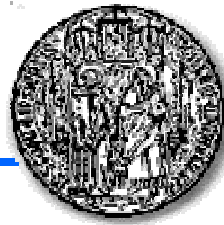
→ UML : siehe Programmierung und Programmiersprachen

Geschäftsprozesse in UML - Die Include- und Generalisierungsbeziehung

4.1.2. Die Definitionsphase – Geschäftsprozesse

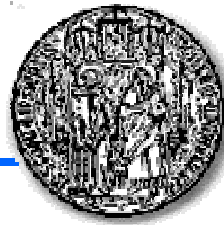


Geschäftsprozess vs. Funktionen



4.1.2. Die Definitionsphase – Geschäftsprozesse

- **Ergebnisse**
 - Geschäftsprozessdiagramm
 - o Alle Geschäftsprozesse und Akteure sind eingetragen
 - Beschreibung der Geschäftsprozesse
 - o Alle Geschäftsprozesse sind umgangssprachlich oder mittels Schablone beschrieben.
- **Konstruktive Schritte**
 - 1. Akteure ermitteln**
 - o Welche Personen führen diese Aufgaben zur Zeit durch und besitzen daher wichtige Kenntnisse über die durchzuführenden Arbeitsabläufe? Welche Rollen spielen diese Personen?
 - o Welche Personen werden zukünftig diese Aufgaben durchführen und auf welche Vorkenntnisse muss die Benutzungsoberfläche abgestimmt werden? Welche Rollen spielen diese Personen?
 - o Wo befindet sich die Schnittstelle des betrachteten Systems bzw. was gehört nicht mehr zu dem System?



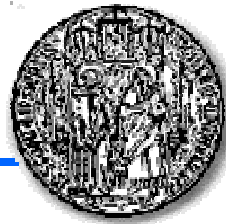
4.1.2. Die Definitionsphase – Geschäftsprozesse

2. Geschäftsprozesse für die Standardverarbeitung ermitteln

- o Primäre und ggf. sekundäre Geschäftsprozesse betrachten
- o Welche Standardverarbeitung besitzen sie?
 - a) mittels Akteuren
 - ❖ Sind die Akteure Personen?
 - ❖ Welche Arbeitsabläufe lösen sie aus?
 - ❖ An welchen Arbeitsabläufen wirken sie mit?
 - b) mittels Ereignissen (Akteure = externe Systeme)
 - ❖ Erstellen einer Ereignisliste
 - ❖ Für jedes Ereignis einen Geschäftsprozess identifizieren
 - ❖ Externe und zeitliche Ereignisse unterscheiden
 - c) mittels Aufgabenbeschreibungen
 - o Was sind die Gesamtziele des Systems?
 - o Welches sind die zehn wichtigsten Aufgaben?
 - o Was ist das Ziel jeder Aufgabe?

3. Geschäftsprozesse für Sonderfälle formulieren

- o Erweiterungen und Alternativen mittels Schablone erstellen
- o Aufbauend auf Standardfunktionalität mit extend die Sonderfälle formulieren, d.h. erweiterte Geschäftsprozesse beschreiben.



4.1.2. Die Definitionsphase – Geschäftsprozesse

4. Aufteilen komplexer Geschäftsfälle

- o Komplexe Schritte als eigene Geschäftsprozesse spezifizieren (include)
- o Komplexe Geschäftsprozesse (viele Sonderfälle) in mehrere Geschäftsprozesse zerlegen und Gemeinsamkeiten mit include modellieren
- o Umfangreiche Erweiterungen als Geschäftsprozesse spezifizieren (extend)

5. Gemeinsamkeiten von Geschäftsprozessen ermitteln

- o Auf redundanzfreie Beschreibung achten (include).

• Analytische Schritte

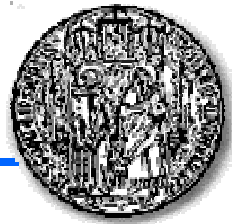
6. »Gute« Beschreibung

- o Verständlich für den Auftraggeber
- o Extern wahrnehmbares Verhalten
- o Fachliche Beschreibung des Arbeitsablaufs
- o Standardfall vollständig und Sonderfälle separat
- o Maximal eine Seite

7. Fehlerquellen

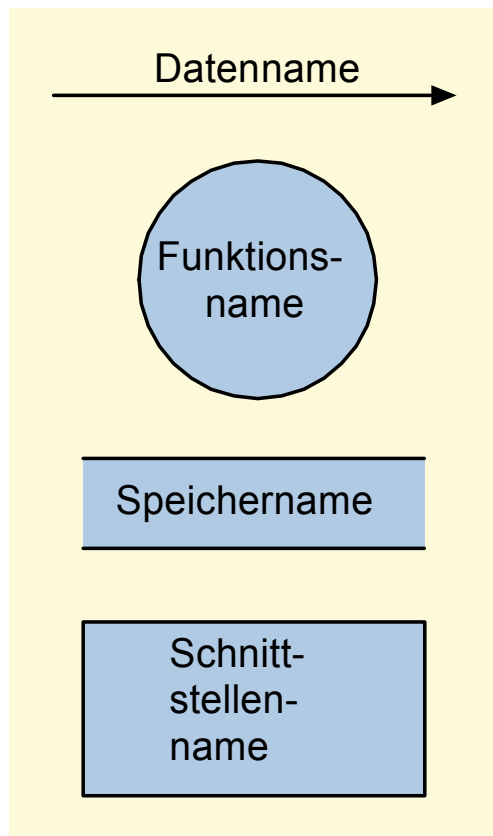
- o Zu kleine und damit zu viele Geschäftsprozesse
- o Zu frühe Betrachtung von Sonderfällen
- o Zu detaillierte Beschreibung

Checkliste Geschäftsprozess (4.-7.)



4.1.3. Die Definitionsphase – DFD

- Ein **DFD** (*data flow diagram*) beschreibt die Wege von Daten bzw. Informationen zwischen Funktionen, Speichern und Schnittstellen und die Transformation der Daten bzw. Informationen durch Funktionen.

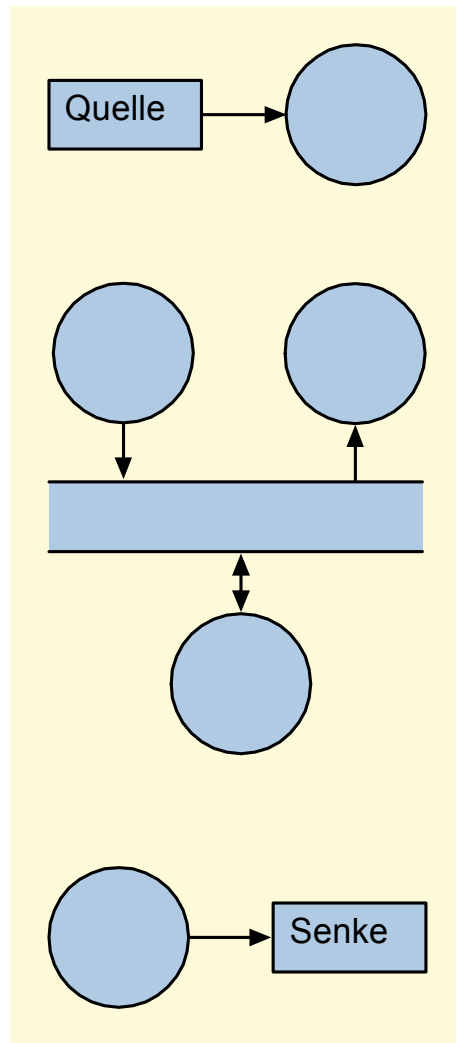


Notation nach / DeMarco 79 /

- 1. Datenfluss**
- 2. Funktion oder Prozess**
- 3. Datenspeicher**
- 4. Schnittstelle zur Umwelt**

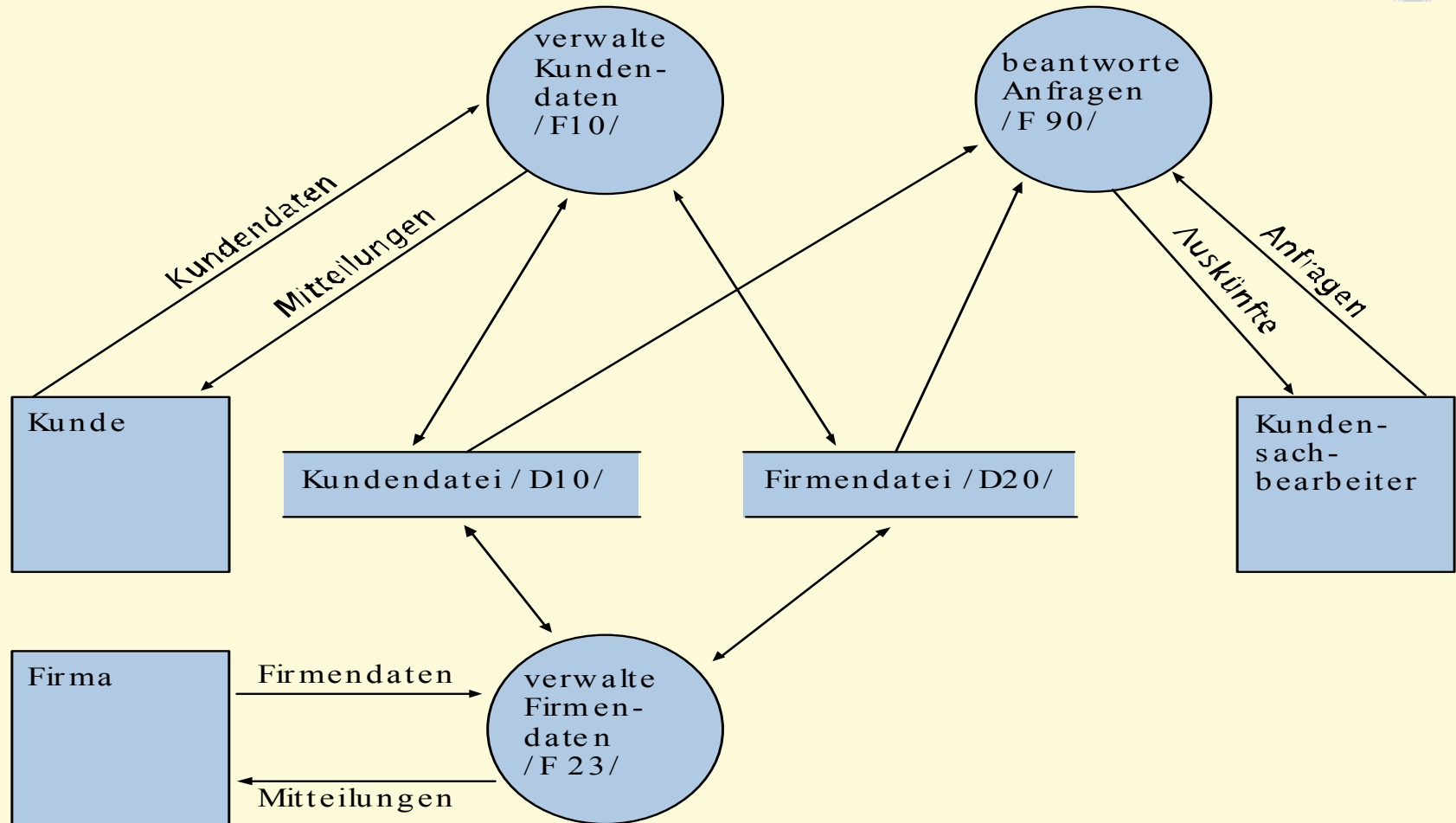


4.1.3. Die Definitionsphase – DFD

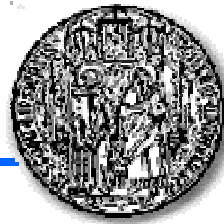


- **Grundidee** eines DFD besteht darin, dass man sich vorstellt, das zu entwickelnde System läuft bereits.
 - o Man macht sich keine Gedanken darüber, wie das System initialisiert und terminiert wird.
 - o Man konzentriert sich darauf, welche Informationen von wo nach wo durch das System fließen.
- System hat **Schnittstellen** mit seiner Umwelt.
- **Umwelt** besteht für das System aus Informationsquellen und Informationssensen.
- **Speicher** sind Hilfsmittel zur Ablage von Informationen. Informationen können hineinfließen oder herausgelesen werden.

4.1.3. Die Definitionsphase – DFD



DFD einer Kundenverwaltung

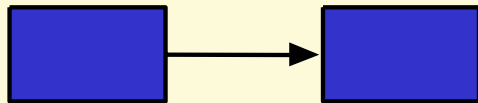


4.1.3. Die Definitionsphase – DFD

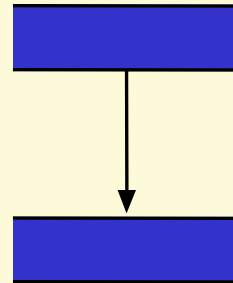
Beim Zeichnen eines DFD sind folgende syntaktische Regeln einzuhalten

- Ein DFD enthält mindestens 1 Schnittstelle
- Jede Schnittstelle ist i. Allg. nur einmal vorhanden
 - o Ausnahme: Wird das DFD unübersichtlich, dann kann eine Schnittstelle mehrfach gezeichnet werden
- Zwischen Schnittstellen gibt es keine Datenflüsse
- Jeder Datenfluss hat einen Namen
 - o 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

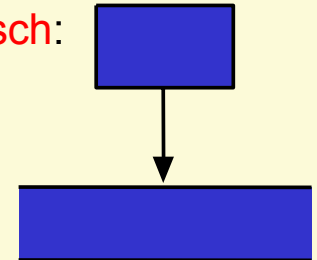
Falsch:

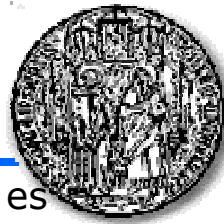


Falsch:



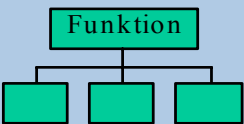
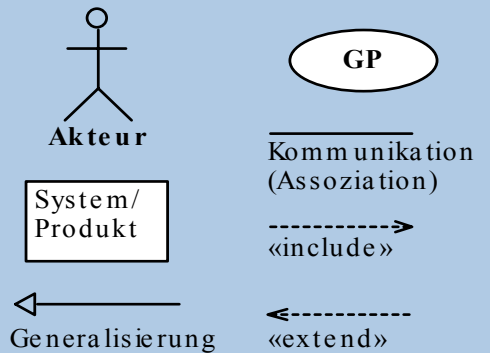
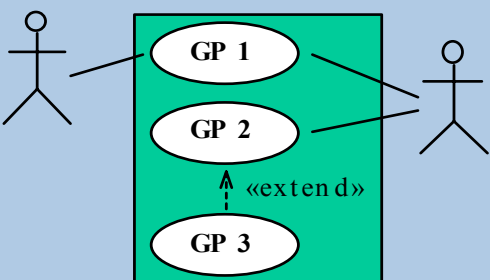
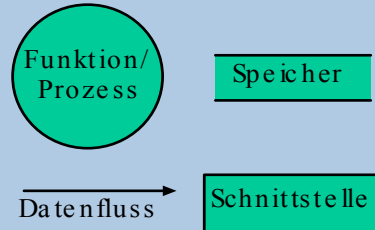
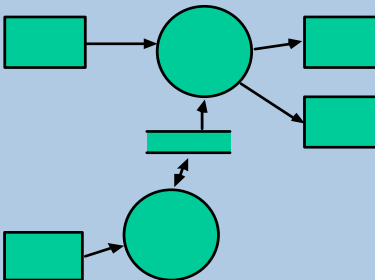
Falsch:

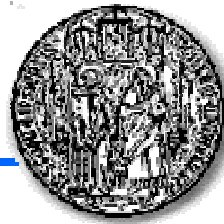




4.1.3. Die Definitionsphase – DFD

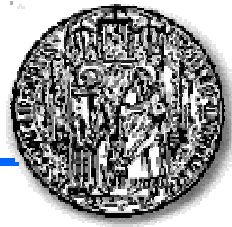
- Das DFD beschreibt den Datenfluss, nicht den Kontrollfluss, daher enthält es weder Entscheidungen noch Schleifen
- Steht eine Schnittstelle für eine Vielzahl von beliebig vielen Instanzen, dann wird sie als eine Schnittstelle dargestellt
- Wird das System durch eine eng begrenzte Anzahl von gleichartigen 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
- Bei der Wahl einer Schnittstelle wird von der konkreten Ein- oder Ausgabe einer Information in das System 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
- 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

Basis - konzept	Funktionale Hierarchie	Arbeitsablauf		Informationsfluss
Notation	Funktionsbaum	Geschäftsprozess (GP)		Datenflussdiagramm
»Muster« der Notation	Baumhierarchie	Netz (UML)	Muster/ Schablone	Netz
Elemente		 Beispiel: GP-Diagramm 	Geschäftsprozess: Ziel: Kategorie: Vorbedingung: Nachbedingung Erfolg: Nachbedingung Fehlschlag: Akteure: Auslösendes Ereignis: Beschreibung: 1 2 Erweiterungen: 1a Alternativen: 1a	 Beispiel : 
Methodisches Vorgehen	top-down	Von den Akteuren ausgehend (outside-in)		Von den Schnittstellen ausgehend (outside-in)
Regeln	■ Zu einer Vater-fkt. nur fachlich eng zusammengehörige Fkt. ■ Pro Hierarchieebene gleiches Abstraktionsniveau	■ GP-Name: Gerundium oder Substantiv-Verb oder Anfang-Ende ■ Zuerst Standardfall, dann Sonderfälle ■ Gemeinsames Verhalten: «include»-Beziehung ■ Erweitertes Verhalten: «extend»-Beziehung		■ Nur Datenfluss beschreiben, keinen Kontrollfluss ■ Schnittstelle ist Quelle/Senke von Informationen ■ Nur Funktionen/ Prozesse greifen auf Schnittstellen und Speicher zu



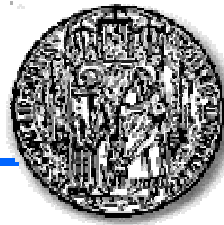
4.1.4. Die Definitionsphase – Funktionale Sicht

- Die **funktionale Sicht** auf ein zu entwickelndes Produkt kann durch die hierarchische Gliederung von Funktionen –angeordnet in einem **Funktionsbaum**- beschrieben werden.
- Durch **Geschäftsprozesse** können die Arbeitsabläufe von Akteuren mit dem Produkt auf einer hohen Abstraktionsebene beschrieben werden. Die Dokumentation erfolgt in **UML** durch **Geschäftsprozessdiagramme**. Zur Spezifikation einzelner Geschäftsprozesse kann eine **Geschäftsprozess-Schablone** eingesetzt werden.
- Durch **DFD** kann der Informationsfluss zwischen Funktionen, Speicher und Schnittstellen zur **Umwelt** dargestellt werden.



4.2. Die Definitionsphase – OO Sicht

1. Objektorientierte Software-Entwicklung
2. Klassifikation von Operationen
3. Aufstellen von UML-Diagrammen für gegebene Problemstellung
4. Einsatz von Paketen und Kriterien bei der Paketenbildung
5. Dynamische Abläufe
6. CASE-Werkzeuge

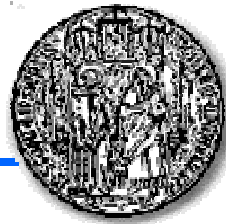


4.2.1. Die Definitionsphase – OO-Konzepte

→ [Dr. Leopold, Programmierung und Programmiersprachen](#)

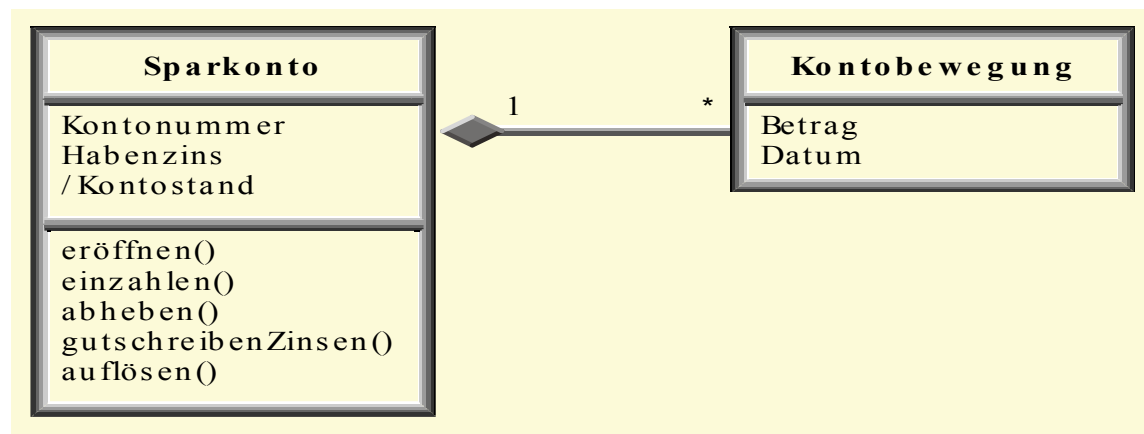
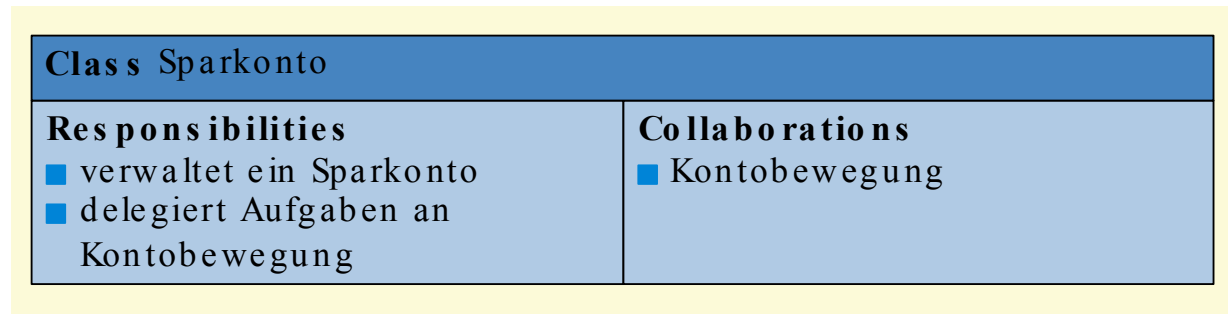
- Wichtigste **OO-Konzepte** sind :
 - o **Objekte**
 - o **Klassen**
 - o **Attribute**
 - o **Operationen**
 - o **Assoziationen**
 - o **CRC-Karten**
 - o **Vererbung**
 - o **Pakete**
 - o **Botschaften**
 - o **Szenarios**
- Wir behandeln nur **Pakete** und **CRC-Karten** da die anderen Konzepte in der oben genannten Vorlesung schon ausführlich behandelt wurden.

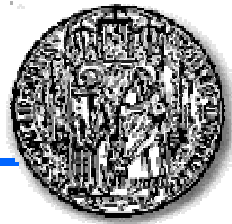
Die wichtigsten Konzepte



4.2.1. Die Definitionsphase – OO-Konzepte

- CRC-Karten (Class/Responsability/Collaboration-Karten) sind Karteikarten. Oben auf der Karte wird der Name der Karte eingetragen. Restliche Karte wird in 2 Hälften geteilt. Auf der einen Hälfte werden die **Verantwortlichkeiten** (responsabilities) der Karte notiert und auf der anderen die **Kollaborationen** (notwendige Klassen)

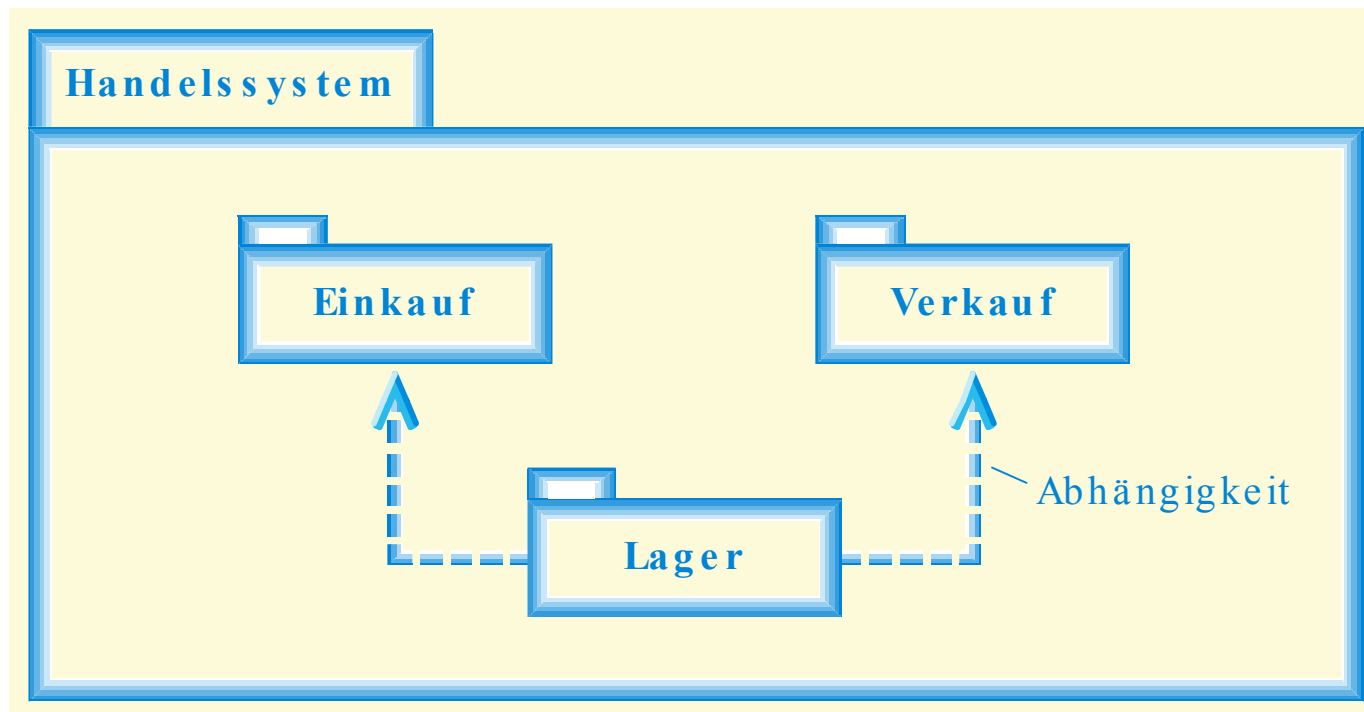


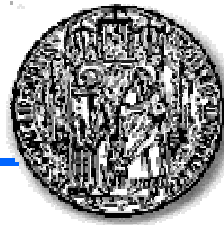


4.2.1. Die Definitionsphase – OO-Konzepte

- **Pakete** sind **Strukturierungsmechanismus** und erlauben es, Komponenten zu einer größeren Einheit zusammenzufassen
- Begriff ist allerdings nicht eindeutig definiert

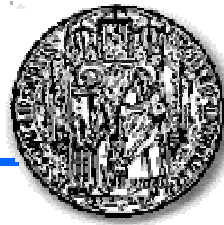
Pakete eines Handelssystems





4.2.1. Die Definitionsphase – OO-Konzepte

- **Kriterien**
 - o Das Paket soll eine logische Einheit bilden, d.h. es soll so strukturiert sein, dass es...
 1. den Leser durch das Modell führt
 2. einen Themenbereich enthält, der für sich allein betrachtet und verstanden werden kann
 3. Klassen enthält, die logisch zusammengehören
z.B. Artikel, Lieferant und Lager
 4. für sich entworfen und evtl. implementiert werden kann, wobei eine wohldefinierte Schnittstelle zur Umgebung vorhanden ist.
- Die **Schnittstelle** soll...
 - o Vererbungsstrukturen nur in vertikaler Richtung schneiden, d.h. zu jeder Unterklasse sollen alle Oberklassen in dem Paket enthalten sein
 - o keine Aggregation durchtrennen
 - o möglichst wenig Assoziationen enthalten
- Die Kopplung zwischen den Klassen – innerhalb eines Pakets – soll möglichst groß und zwischen Paketen möglichst gering sein
 - o Jedes potenzielle Paket prüft man auf seine Kopplungseigenschaft.



4.2.1. Die Definitionsphase – OO-Sicht

- **Objektorientierte SW-Entwicklung** basiert auf einer Reihe von Grundkonzepten. Hier werden verschiedene Diagrammartentypen benutzt, um unterschiedliche Aspekte darzustellen. Als grafische Notation wird UML verwendet.
- OO-Konzepte gut geeignet für Datenkapselung
- Für die Wartbarkeit des entstehenden Software-Systems ist es entscheidend, dass das Konzept der Vererbung sinnvoll eingesetzt wird