Strukturierte Analyse

Allgemeines

- Anfang in Mitte der 70er Jahre, bis vor ca. 10 Jahren wichtigste Standardmethode der Systemanalyse
- Entwicklung von Modellen des Anwendungsbereiches, geben Überblick über das betrachtete System und ermöglichen Darstellung jedes erkannten Details in einem Modell
- einfache Modellnotation, stark grafisch orientiert -> soll von jedem leicht verstanden werden können
- Grafiken stellen die Komponenten des Modells und ihre Schnittstellen dar
- SA-Modell enthält alle wesentlichen Informationen -> keine methodenexternen Dokumentationsmittel werden benötigt
- Informationen aus Gesprächen mit dem Anwender können einfach in das Modell eingefügt werden
- durch Konsistenzprüfung des Modells sind viele Fehler aufspürbar
- **induktiver Ansatz**: Modellierung des neuen Systems mit Studium eines oder mehrerer Vorgängersysteme
- **deduktiver Ansatz**: ohne Studium von Vorgängersystemen, also nur unter Beachtung der an das System gerichteten Ziele und Anforderungen -> am üblichsten da induktiver Ansatz wesentlich zeitaufwendiger und daher unwirtschaftlicher

Vorgehensweise bei der Strukturierten Analyse

- es werden zwei verschiedene Zerlegungsstrategien unterschieden:
  1. **funktionsorientierte Zerlegung:** ältere Variante; top-down-Ansatz vom System als Ganzes, zuerst wird Ereignistabelle aufgestellt, daraus wird Kontext-Diagramm erstellt, dann schrittweise Verfeinerung der Prozesse bis zur Ebene der elementaren Prozesse durch Datenflußdiagramme, elementare Prozesse durch Prozeßspezifikation (PSPEC) spezifiziert; bei diesem Vorgehen Identifizierung der Datenelemente -> Beschreibung im Datenkatalog; am Ende Normalisierung der Datenstrukturen -> Datenmodell entsteht (geeignet für Datenbank-Design); aber: mangelnde Betonung der Datenstrukturen als zentrale Informationsquelle für den Analytiker
  2. **essentielle Zerlegung:** besser, inside-out-Strategie; jedem Ereignis wird Prozeß zugeordnet der die zum Ereignis gehörende Systemantwort erarbeitet; Datenelemente werden entsprechend der Struktur der Objekte im Problemraum zusammengefaßt -> Datenmodell ist leicht zu normalisieren -> stellt Weiterentwicklung der funktionsorientierten Zerlegung dar, gleicht deren Nachteile aus, deshalb wird im folgenden auf die essentielle Zerlegung detaillierter eingegangen

Notationen der Strukturierten Analyse

Im folgenden werden die bei der strukturierten Analyse verwendeten Notationen/ Diagrammtypen beschrieben:

1. Ereignistabelle

- Ausgangspunkt für die essentiellen Zerlegung
- enthält Liste der Ereignisse und nennt die zugehörigen Auslöser und Antworten

Laufende Nummer	*Ereignis*	*Auslöser*	*Antwort*
...	...	...	...

- Auslöser und Antwort sind Datenflüsse: durch Auslöser erhält das System Kenntniss vom Ereignis; die Antwort ist der Datenfluß, den das System als Reaktion auf das Ereignis an die Umgebung abgibt

- Einträge der Ereignistabelle werden numeriert und im (vorläufigen) essentiellen Datenflußdiagramm als Bestandteil der Prozeßnummern weiterverwendet

2.Datenflußdiagramm (DFD)

- beschreiben Modelle grafisch
- enthalten kaum Informationen über die einzelnen Datenstrukturen
- zeigen Existenz von Speichern und die Funktionen zur Nutzung der Speicher
- Datenflüsse und Speicher werden im Datenkatalog (Datadictionary) spezifiziert
- DFD´s benutzen die folgenden grafischen Symbole:

Prozeß:

- Name: Prozeßname; gebildet aus Substantiv und einem starken Verb, das eine Aktion exakt beschreibt; Prozeßname kann gut nach den vom Prozeß erzeugten Ergebnissen vergeben werden (z.B. „Anschrift eintragen“); falls Namensfindung schwer dann Hinweis auf noch vorhandene Modellierungsprobleme (-> lokal neu fassen oder weiter verfeinern)
- Nr: hierarchisch gebildete Nummer
- Aufgabe: konvertieren Eingabedaten in Ausgabedaten, enthalten den dafür benötigten Algorithmus
- Prozeß in nächster Verfeinerungsebene durch DFD näher unterteilt, auf unterster Verfeinerungsebene Prozeßspezifikationen (PSPEC´s) erstellt

Datenspeicher:

- Name besteht aus Substantiv das auf den Inhalt hinweist
- temporärer Aufenthaltsort für Daten
- Attributierung wird im Datenkatalog spezifiziert
- passiv, d.h. Daten nur eingestellt oder entnommen, wenn Prozeß dies veranlaßt, beim Lesen wird Speicherinhalt nicht verändert (non destructive read)
- erscheinen erst auf der Ebene, wo sie von mindestens 2 Prozessen genutzt werden
- nicht erlaubt:
  - Datenspeicher die nur beschrieben aber niemals gelesen werden
  - Datenspeicher die nur gelesen aber niemals beschrieben werden
  - Ausnahme: Datenspeicher werden auch in die darunterliegenden Ebenen eingetragen auch wenn sie dort nur von einem Prozeß benutzt werden (scheinbare read-only-Speicher)

Datenflüsse:

- Kanäle für Informationsflüsse, übertragen Datenpakete und lösen den empfangenden Prozeß datengetrieben aus
- Name enthält ein Substantiv, kann durch „Modifier“ ergänzt werden der den Verarbeitungszustand beschreibt (z.B. <neu>); Datenflüsse zwischen Prozessen und Datenspeichern erhalten nur dann Name, wenn vom Prozeß nicht alle Attribute des Datenspeichers benutzt werden
- Datenflüsse müssen mit mindestens einem Prozeß verbunden sein-> somit nur zwischen Prozeß

und Terminator bzw. zwischen Prozeß und Speicher möglich
- Zusammensetzung der Datenpakete im Datenkatalog beschrieben

Terminator:

- stehen für andere Systeme gegen die sich das System abgrenzt
- nur im Kontext-Diagramm
- treten als Sender und Empfänger auf, werden nicht weiter beschrieben; Datenflüsse zwischen Terminatoren und dem System stellen die **Schnittstellen** (bzw. **Außenbeziehungen**) des Systems dar
- keine Beziehungen zwischen Terminatoren darstellbar

Verfeinerungshierarchie von Datenflußdiagrammen:
- Kontext-Diagramm:
  - oberste Ebene des Modells
  - stellt Beziehungen des Systems zu seiner Umwelt und auch die Abgrenzungen zu anderen Systemen dar
  - System wird durch einzigen Prozeß dargestellt, alle Terminatoren vorhanden, keine Datenspeicher erlaubt

  ➔ dient zur Dokumentation der Terminatoren und der Schnittstellen des Systems, Funktionsweise des Systems nicht darstellbar
- Verfeinerung jedes Prozesses durch DFD oder auf unterster Ebene durch PSPEC mit gleichen Außenbeziehungen (d.h. mit gleichen Eingabe/Ausgabe-Datenflüssen (Datenflußgleichgewicht))
- Verfeinerung solange fortgeführt bis Prozeß (primitive Prozesse/ elementare Prozesse) durch PSPEC beschreibbar (mindestens halbe, maximal ganze Seite)

  ➔ durch Beschreibung auf unterster Ebene Vermeidung von Redundanz
- Bezeichnung von DFD´s als **Parent-Diagramm** (das Diagramm welches verfeinert wird) bzw. **Child-Diagramm** (das verfeinernde Diagramm)
- Verfeinerungstiefe muß nicht für alle Prozesse einer Ebene gleich sein, ist aber meist annähernd gleich
- Numerierung von Prozessen und DFD´s:
  - DFD erhält Nummer des Prozesses den es verfeinert (**Diagrammnummer**)
  - **Prozessnummern** aus Diagrammnummer, Dezimalpunkt und einer DFD-lokalen Nummer gebildet; besitzt keine inhaltliche Bedeutung

<u>3. Prozeßspezifikationen (PSPEC)</u>
- auch als Mini-Spezifikationen (MSPEC) bezeichnet
- dient zur eindeutigen Beschreibung eines Prozesses d.h. die von einem Prozeß durchzuführende Transformation wird spezifiziert; PSPEC erhält Nummer des Prozesses den sie beschreibt
- dabei ist jedes Mittel zulässig z.B. strukturierte Sprache, Pseudocode, Entscheidungsbäume, Entscheidungstabellen...
- **strukturierte Sprache**:
  - ist Übersetzung von Pseudocode in die Sprache der Anwender, Anwender soll diese Sprache verstehen können, dient der Kommunikation zwischen Anwender und Analytiker
  - verwendet **Sequenz** (einfache Sätze), **Sequenz ohne Reihenfolge** (parallele Verarbeitung möglich da Reihenfolge der Anwendungen beliebig), **Fallunterscheidung**, **abgeschlossene Wiederholung**, **Blockstruktur** (Kontrollblöcke mit jeweils nur einem Ein- und Ausgang)
  - Anweisungszeilen werden zwecks guter Lesbarkeit gemäß Schachtelungstiefe eingerückt
  - vorkommende Worte:
    - im Datenkatalog erklärte Namen

- lokale Namen
- reservierte Logik-Worte
- Worte ohne Bedeutung (zur besseren Lesbarkeit)
- Makros (fassen mehrere Anweisungen zu einem Wort zusammen)

4. Datenkatalog (Data Dictionary)

- beschreibt jeden Datenfluß und jeden Speicher in seiner Zusammensetzung (nur inhaltliche Bedeutung, keine speziellen Datenstrukturen)
- verwendet Data Dictionary Syntax (vereinfachte Backus-Naur-Form)
- Einträge des Datenkatalogs sind alphabetsich geordnet
- **Data Dictionary Syntax**:

Symbol	***Name***	***Bedeutung***
name	Bezeichner	Entitytyp, Bezeichnungstyp, Attribut
=	Zusammensetzung	besteht aus
+	Verkettung	und (Sequenz)
"..."	Diskreter Wert	der Wert der Variablen
[..|..]	Selektion	entweder...oder
{}	Iteration	mehrfaches Auftreten
n{...}m	Iteration von n bis m	
()	Option	kann vorhanden sein
@name	Schlüsselkandidat	Zugriff über name erforderlich
<...>	Modifier	kommentierende Ergänzung zum Name
"..."	Kommentar	zusätzliche Information

5. ER-Diagramme

- verdeutlichen Relationen zwischen Datenspeichern, die anderenfalls nur in den Prozeßspezifikationen zu sehen wären
- jedes ER-Datenelement entspricht einem Datenspeicher im Datenflußdiagramm

6. Zustands-Transitions-Diagramme

- modellieren zeitabhängiges Verhalten; beschreiben Steuerungsprozesse oder die Zeitsteuerung für die Ausführung von Funktionen und Datenzugriffen die von Ereignissen ausgelöst werden
- werden hier nicht weiter behandelt ...

Essentielle Zerlegung

- **Essenz**: logischer Kern des Systems; diejenigen Systemteile, die bei jeder denkbaren Implementierung vorhanden sein müssen; ändert sich im Gegensatz zur Implementierungstechnologie im Laufe der Zeit kaum
- die in der Systemanalyse entwickelten Modelle sollen zeitlos und implementationsunabhängig sein
  - ➔ **Modell der „perfekten internen Technologie":**
    - Prozessoren benötigen zur Aufgabenerledigung keine Zeit, sind wirklich zuverlässig und billig
    - Speicher benötigen keine Zeit für Datenzugriffe, haben unendliche Speicherkapazität, sind billig
    - ➔ erleichtert die Vorstellung der Implementationsunabhängigkeit; innerhalb des Systems müssen keine Vorkehrungen gegen die Grenzen der verfügbaren Technologie berücksichtigt werden; bei Änderung der Implementierung muß Essenz nicht geändert werden

Begriffsklärung:

- **grundlegende Aktivitäten**: Aktivitäten die dazu beitragen die eigentliche Zielsetzung des Systems zu erreichen; müssen bei jeder denkbaren Implementierung des Systems ausgeführt werden; werden durch externes oder zeitliches Ereignis ausgelöst
- **essentieller Speicher**: Gesamtheit aller Daten, die sich ein System merkt und die seine grundlegenden Aktivitäten brauchen; die gespeicherten Daten stammen aus der Umgebung oder entstehen bei Ausführung der grundlegenden Aktivitäten
- **Verwaltungsaktivitäten**: erstellen und verwalten den essentiellen Speicher (Daten einfügen, speichern, aktualisieren, löschen); externe Ereignisse veranlassen die Verwaltungsaktivität den essentiellen Speicher zu aktualisieren
- **essentielle Aktivitäten**: grundlegende Aktivitäten oder Verwaltungsaktivitäten
- **externes Ereignis**: Datenfluß von außerhalb des Systems zu Aktivität; folgendermaßen benannt: `externes_Objekt aktives_Verb Objekt` (z.B. Kunde wünscht Flugauskunft)
- **zeitliches Ereignis**: im SA-Modell nicht modelliert sondern in PSPEC des erkennenden Prozesses enthalten: `Zeit für <Datenfluß>`
- **Informanten**: zusätzliche Input-Datenflüsse des Systems; stehen immer zur Verfügung; müssen nicht im System gespeichert werden; lösen keine Ereignisse aus; werden innerhalb des Systems aktualisiert

Vorgehensweise:

- ausgehend von den Ergebnissen der ersten Gespräche mit dem Anwender werden die Ziele des Systems ermittelt und in einem Brainstorming die Ereignisse erkannt, auf die das System reagieren muß (=**grundlegende Aktivitäten finden**), dabei müssen spontane und geplante Reaktionen getrennt werden (Ereignisse auf die spontan reagiert werden muß werden auf separater Liste geführt; nur geplante Reaktionen von Interesse, aber: diese können sich in spontanen Reaktionen verstecken -> weiter analysieren und zerlegen) -> **Ereignistabelle** wird erstellt
- jedem grundlegenden Ereignis der Ereignistabelle wird Prozeß zugeordnet der die Aktivitäten ausführt die das System als Reaktion auf das Ereignis ausführen muß -> erster Teil der **essentiellen Ebene** entsteht
- das **Kontext-Diagramm** wird erstellt nach Erkennen der wahren Terminatoren
- **Speicher des Systems finden**: essentieller Speicher muß zerlegt werden da sonst zu komplexer Aufbau (würde komplizierte Such- und Pflegemechanismen erfordern); die Zusammensetzung der Auslöser und Antworten des Systems wird erfragt und im **Datenkatalog** spezifiziert; Vergleich der Ausgabe-Datenelemente mit den Eingabe-Datenelementen für jeden Prozess -> erkennen von Speichern die die Prozesse benötigen
  - ➔ zu jedem Objekt der Systemumgebung, über das das System sich Daten merken muß, wird Speicher eingeführt
    - ➔ erleichtert die Normalisierung des Datenmodells
- **Verwaltungsaktivitäten finden**, die den Speicher des Systems erstellen und die Zustandsübergänge auslösen

➔ **vorläufiges essentielles Modell** entsteht durch Integration aller essentieller Aktivitäten zu einem System; das Modell besitzt folgenden Eigenschaften:

- die essentiellen Aktivitäten sind voneinander entkoppelt, d.h. sie kommunizieren nicht direkt miteinander, sondern beziehen sich nur auf Inhalte essentieller Speicher
- Lokalität des Modells wird besonders unterstützt durch die Gliederung der Funktionalität des Systems in vollständige ereignisorientierte Prozesse

- **Vergröberung der essentiellen Ebene** zur Vervollständigung der Diagrammhierarchie zwischen essentieller Ebene und Kontext-Diagramm:
  - Prozesse in Gruppen zusammenfassen wobei die Prozesse inhaltlich verwandt sein sollten, d.h. verwandte Systemantworten produzieren
  - Datenspeicher kapseln -> Reduzierung der Anzahl an Speichern oberhalb der essentiellen

Ebene
  - → Vergröberung bedeutet also Bildung von abstrakten Datentypen
- **Verfeinerung der essentiellen Aktivitäten**:
  - Verfeinerung eines essentiellen Prozesses erfolgt aufgrund einer funktionalen Zerlegung, wenn Prozeß zu komplex dann weitere Verfeinerung
  - PSPEC´s notieren
  - → Verfeinerung bedeutet Faktorisieren (funktional zerlegen), um den Zusammenhalt der Prozesse zu vergrößern und ihre Kopplung zu verkleinern

- später: Prüfbedingungen werden modelliert, die Fehler der Systemumgebung abwehren; Entscheidung mit welchen Prozessoren (Sachbearbeiter, Rechner) die einzelnen Aufgaben durchgeführt werden sollen:

Rechner:	Programme, Datenbanken realisieren
manuell:	Stellenbeschreibungen, Vorschriften zur Ablauforganisation

Ergebnis der essentiellen Zerlegung:
essentielles Modell:
- vollständig formal konsistentes SA-Modell das keine Implementationsdetails enthält
- die Speicherinhalte und Verarbeitungsfunktionen müssen vorhanden sein die unabhängig von der gewählten Implementierung vorhanden sein müssen
- besteht aus 2 Komponenten:
  1. **Umgebungsmodell**: Kontext-Diagramm, Ereignistabelle, Kurzbeschreibung der Aufgabe des Systems -> äußere Schnittstelen des Systems und Randbedingungen werden festgelegt; das System wird als black-box behandelt
  2. **Verhaltensmodell**: alle DFD´s außer Kontext-Diagramm, ER-Diagramm, PSPEC´s, Datenkatalog; beschreibt das Innere des Systems (**white-box**)

Strukturierte Analyse am Beispiel der „Silent Kitchen Company“

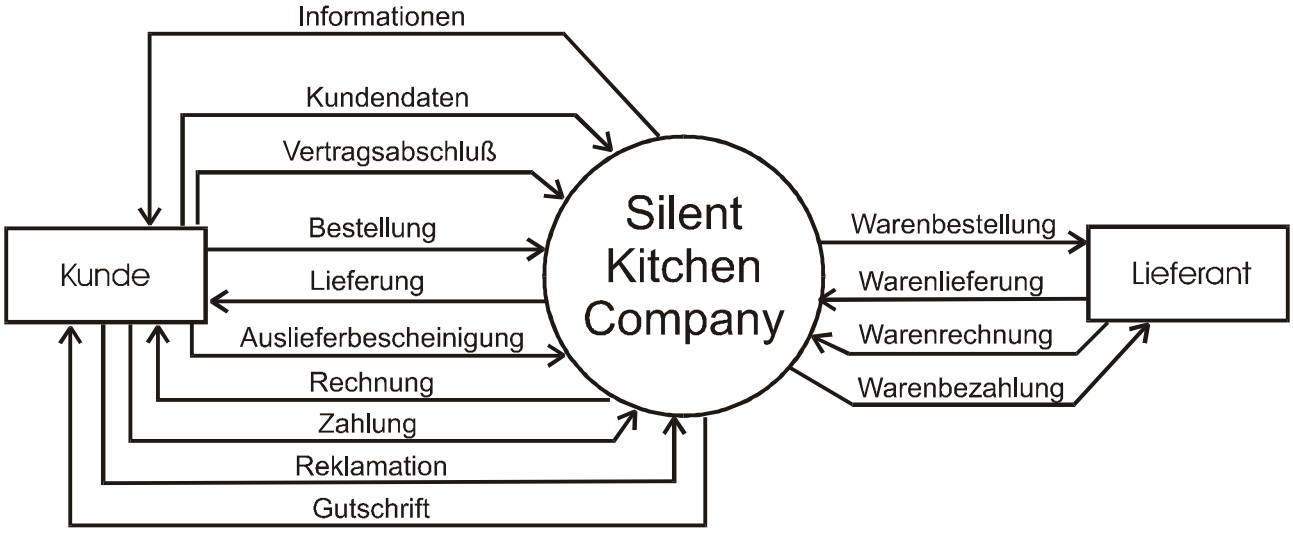
die **Ziele des Systems** werden ermittelt:

- Bestellungen sollen angenommen und bearbeitet werden
- nach Lieferung soll Rechnung verschickt werden
- Kunden sollen Verträge abschließen können
- Reklamationen sind möglich -> Gutschrift
- Zulassung neuer Kundenkonten
- Warenbestellung wenn Vorräte knapp werden
- regelmäßig werden Informationen an alle Kunden verschickt

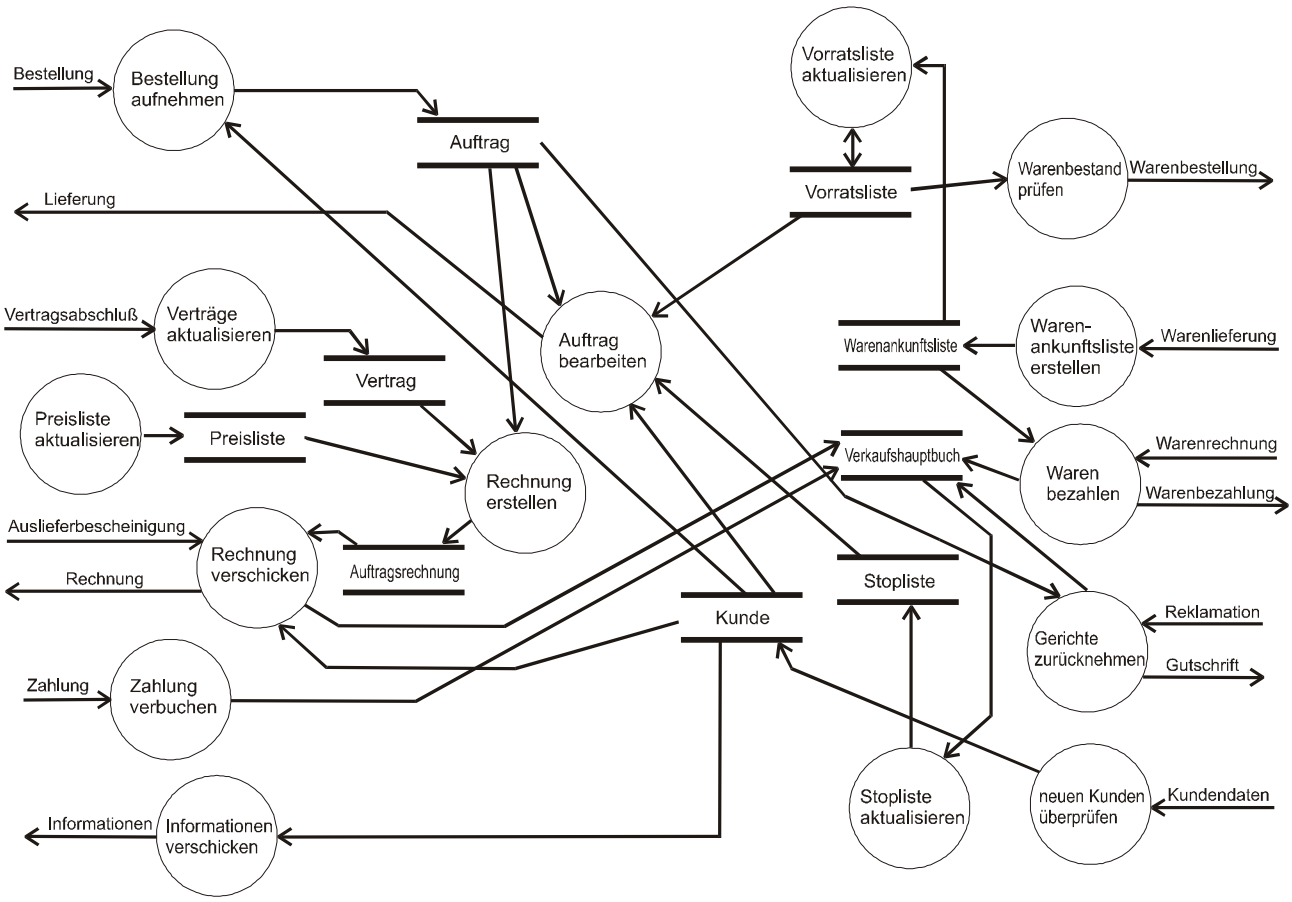
Es werden die Ereignisse erkannt, auf die das System reagieren muß. Diese Ereignisse werden in der **Ereignistabelle** dargestellt:

Lfd. Nr.	***Ereignis***	***Auslöser***	***Antwort***
1	Kunde gibt Bestellung auf	Bestellung	Lieferung
2	Kunde gibt Gerichte zurück	Reklamation	Gutschrift
3	Kunde schließt Vertrag ab	Vertragsabschluß	
4	Rechnung wird verschickt	Auslieferbescheinigung	Rechnung
5	Kunde bezahlt Rechnung	Zahlung	
6	neuer Kunde	Kundendaten	
7	Warenmangel		Warenbestellung
8	Waren werden geliefert	Warenlieferung	
9	Warenrechnung trifft ein	Warenrechnung	Warenbezahlung
10	Zeit, Informationen an Kunden zu verschicken		Information

Das **Kontextdiagramm** wird anhand der Ereignistabelle erstellt:



Das **essentielle Modell** wird entwickelt:

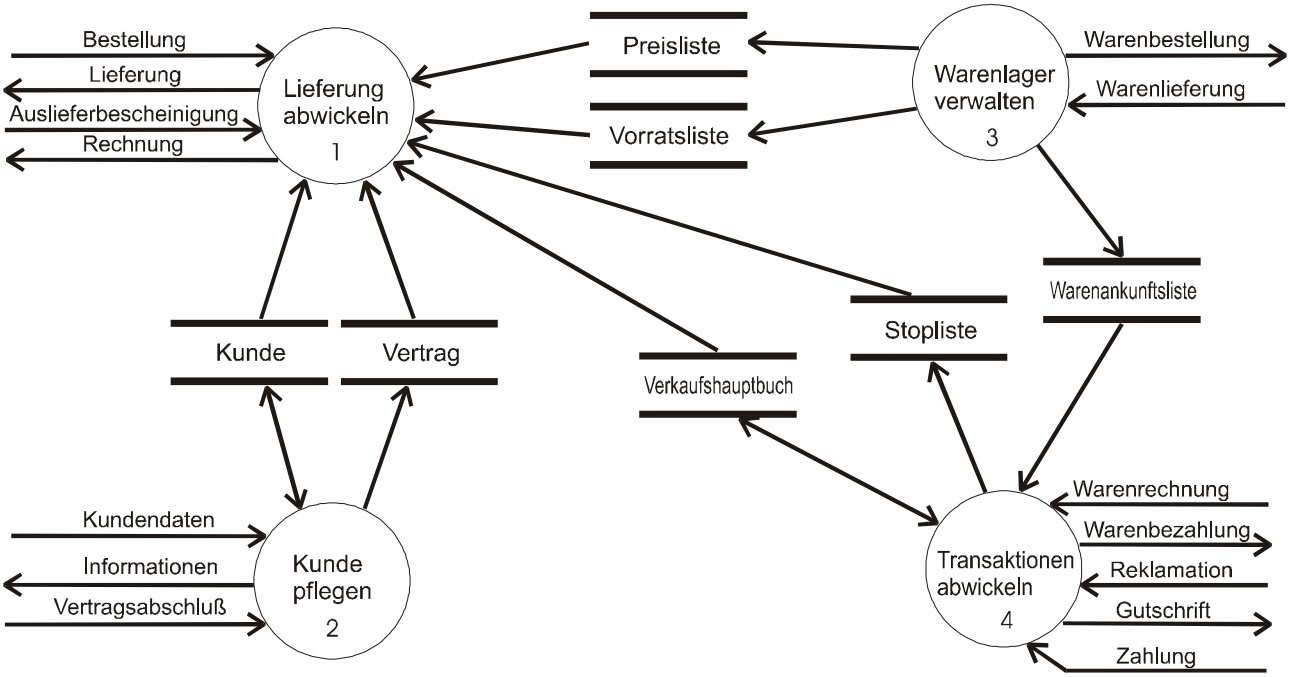


Der **Datenkatalog** wird erstellt:

Auftrag =Auftragsnummer + vorrauss_Liefertermin
Auftragsrechnung =Rechnungsnummer + Betrag + Kundenreferenznummer
Auslieferbescheinigung =Auftragsnummer
Bestellung =Kundennummer + {Gericht + Menge}
Gericht =Gerichtsnummer + Gerichtsbezeichnung + Einzelpreis
Gutschrift =Kundennummer + Kundenname + Anschrift + Betrag
Kunde =Kundennummer + Kundenname + Anschrift + weitere_Eigenschaften
Kundendaten =Kundenname + Anschrift + weitere_Eigenschaften
Lieferung =Auftragsnummer + Kundennummer + Kundenname + Anschrift + {Gericht + Menge}
Preisliste ={Gerichtsnummer + Preis}
Rechnung =Rechnungsnummer + Kundennummer + Kundenname + Anschrift + Betrag + Kundenreferenznummer
Reklamation =Auftragsnummer + Kundennummer + {Gericht + Menge}
Stopliste ={Kundennummer}
Verkaufshauptbuch ={Kundennummer + offener Betrag + {Auftragsnummer + Betrag + Kundenreferenznummer}} + {Bestellungsnummer + Preis}
Vertrag =Vertragsnummer + Menge + Preis
Vertragsabschluß =Kundennummer + Gericht + Menge + Preis
Vorratsliste ={Ware + Menge}
Ware =Warennummer + Warenbezeichnung
Warenankunftsliste ={Ware + Menge}
Warenbestellung =Bestellungsnummer + {Ware + Menge}
Warenbezahlung =Referenznummer + Betrag
Warenlieferung =Bestellungsnummer + {Ware + Menge}
Warenrechnung =Bestellungsnummer + Preis + Referenznummer

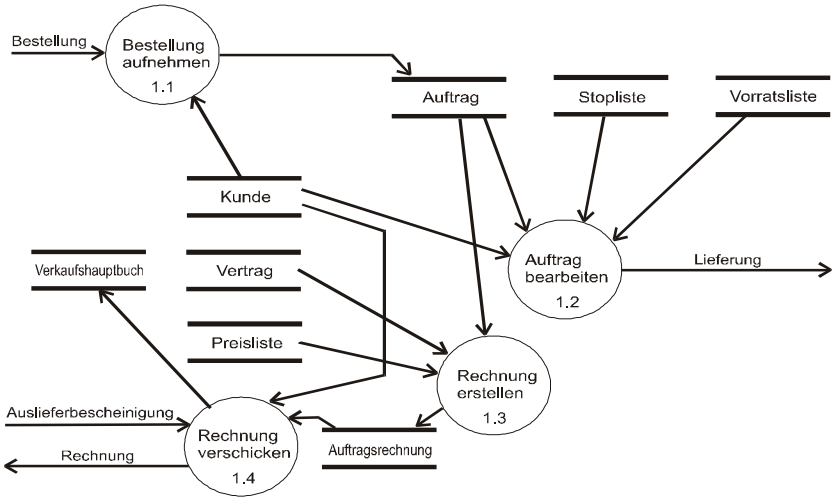
Zahlung =Kundenreferenznummer + Betrag

Eine **Vergröberung** der essentiellen Ebene wird vorgenommen (Ebene 0):

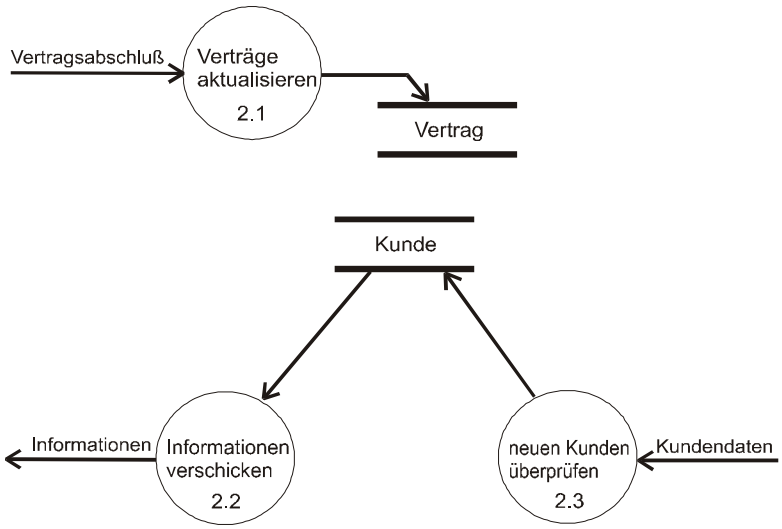


Die vier Prozesse dieses Datenflußdiagramms werden nun wieder verfeinert, so daß das Diagramm der essentiellen Ebene in vier Teile für die Verfeinerungen der Aktivitäten 1 bis 4 zerfällt (Ebene 1):

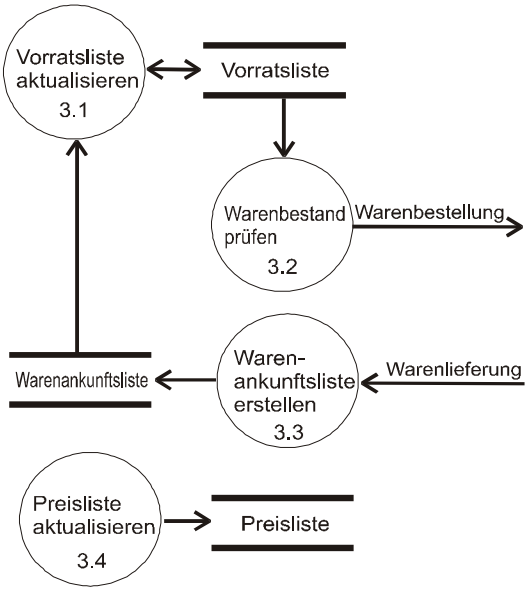
Prozeß „**Lieferung abwickeln**“:



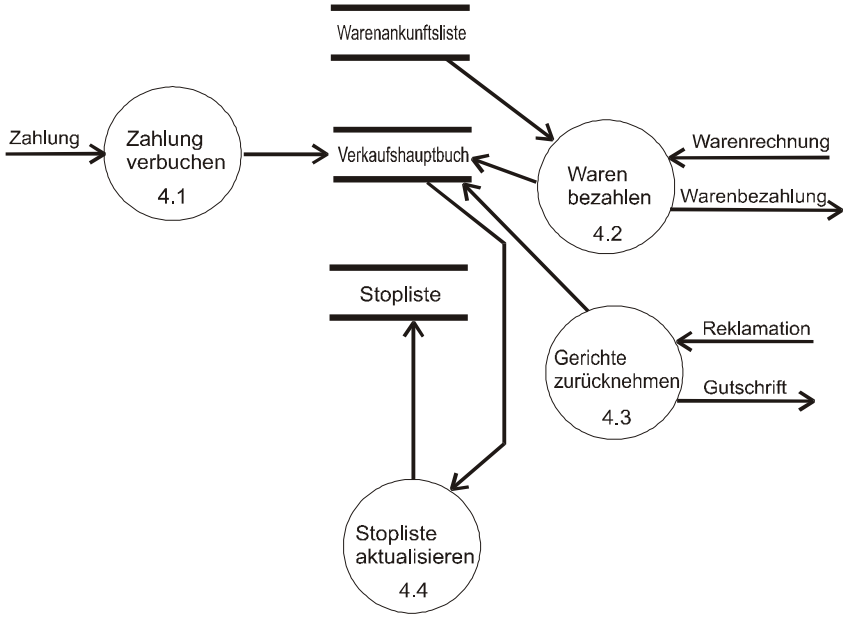
Prozeß „**Kunde pflegen**“:



Prozeß „**Warenlager verwalten**“:



Prozeß „**Transaktionen abwickeln**“:



Nun würde man die essentielle Ebene verfeinern. Hier ist dies nicht notwendig, da sich alle Prozesse durch Mini-Spezifikationen beschreiben lassen.

Die **Mini-Spezifikationen** werden erstellt (hier nur für zwei Beispiele demonstriert):

MSPEC: Auftrag bearbeiten

```
Für Kunde mit Kundennummer = Kundennummer im Auftrag
    wenn Kundennummer nicht auf der Stopliste steht
        prüfe Vorratsliste ob Bestellung erfüllt werden kann
        wenn Vorräte reichen
            Auftrag ausführen
        sonst
            Auftrag zurücklegen
    ansonsten
        Bestellung ablehnen
```

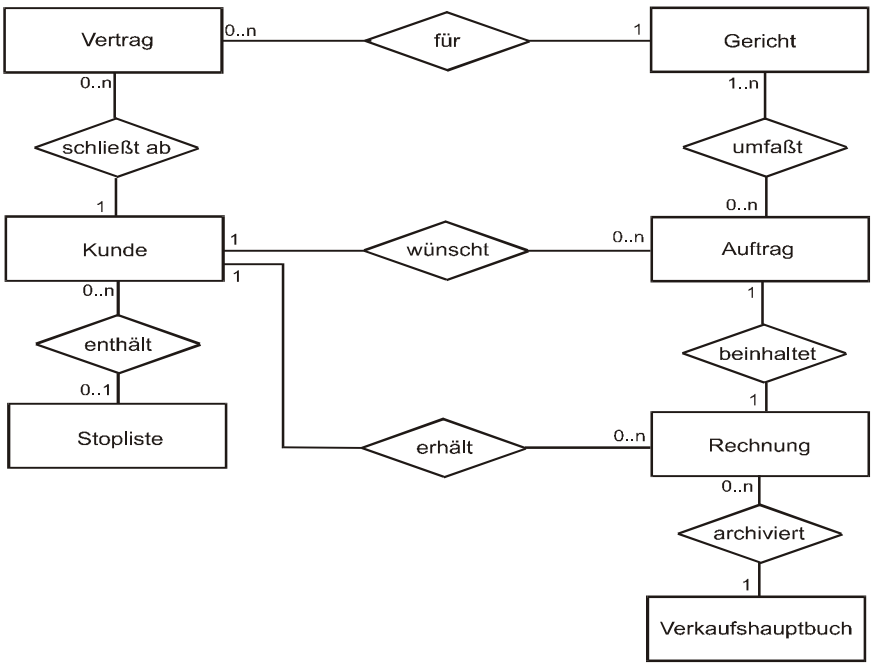
MSPEC: Informationen verschicken:

```
wenn Zeit für <Informationen>
    Informationen an alle Kunden verschicken
```

Das **Entity-Relationship-Diagramm** wird erstellt:

Dazu wird zunächst der Datenkatalog um folgende Beziehungstypen ergänzt:

beinhaltet	=Auftragsnummer + Rechnungsnummer
erhält	=Kundennummer + Rechnungsnummer
für	=Vertragsnummer + Gerichtsnummer
schließt_ab	=Kundennummer + Vertragsnummer
umfaßt	=Auftragsnummer + Gerichtsnummer + Anzahl
wünscht	=Kundennummer + Auftragsnummer



Vergleich Strukturierte Analyse versus Objektorientierte Analyse

Wichtung der 3 orthogonalen Sichten auf ein System:

SA	**OOA**
1. **funktionales Modell** (Datenflußdiagramme)	**Objektmodell** (Klassendiagramm)
2. **dynamisches Modell** (Zustands-Transitions-Diagramme)	**dynamisches Modell** (Zustands-, Sequenz-, Kollaborationsdiagramme)
3. **Objektmodell** (ER-Diagramm)	**funktionales Modell** (DFD´s)

Objektmodell: spezifiziert, auf wen oder was etwas geschieht
dynamisches Modell: spezifiziert, wann etwas geschieht
funktionales Modell: spezifiziert, was geschieht (zeigt, wie bei einer Berechnung die Ausgabewerte aus den Eingabewerten abgeleitet werden)

Bei der strukturierten Analyse macht man die Annahme, daß Funktionen wichtiger und komplexer sind als die Daten. Doch in der Praxis beziehen sich die meisten Anforderungsänderungen auf Funktionen, nicht auf Objekte. Deshalb besitzt die objektorientierte Analyse entscheidende Vorteile, die im folgenden aufgelistet werden:

• Funktionsänderungen können katastrophale Auswirkungen haben	• Problemlose Funktionsänderungen (Objekte verändern sich nicht, nur Methoden verändern)
• Systemgrenzen können nur schwer erweitert werden da die Struktur des SA-Entwurfs teilweise von den Systemgrenzen abgeleitet wird	• Systemgrenzen können leicht erweitert werden indem in die Nähe der Systemgrenze Objekte und Relationen hinzugefügt werden
• Unterschiedliche Entwickler produzieren unterschiedliche Modelle, da die Dekomposition eines Prozesses in Teilprozesse beliebig durchgeführt werden kann	• Unterschiedliche Entwickler spezifizieren ähnliche Objekte, da die Analyse auf den Objekten der Anwendungsdomäne basiert -> dies erhöht die Wahrscheinlichkeit, daß Komponenten aus einem Projekt wiederverwendet werden können
• Semantische Lücke (Strukturbruch) zwischen Analyse und Design	• Übergang von Analyse zu Design problemlos möglich

- Schlechte Integration von Datenbanken (es ist schwierig, Programmcode, der auf Funktionen aufsetzt, mit einer Datenbank zu verschmelzen, die auf Daten aufsetzt)
- Integriert Datenbanken besser mit dem Programmcode (Objekt kann Datenbank- und Programmstruktur modellieren)