**Bernd Blümel**

# Software Engineering

# 1 Einleitung

Im Grundstudium haben Sie sich (hoffentlich) mit der Erstellung kleinerer Computerprogramme beschäftigt. Sie haben gelernt, wie man an eine gegebene Problemstellung herangeht und sie mittels der Entwicklung eines Algorithmus löst und diesen Algorithmus in einer Programmiersprache realisiert.

Für die Abwicklung konkreter Projekte in der industriellen Praxis ist dies aber nicht immer hinreichend. Betrachten wir dies am Beispiel eines neu zu entwickelnden Reisevertriebssystems. Hier fällt es wegen der Komplexität der Aufgabenstellung schwer, einen Algorithmus zu finden und zu implementieren.

Vielmehr werden Sie mit Anwendern reden müssen, um die genauen Anforderungen an das System kennenzulernen. Diese Anforderungen müssen sodann strukturiert und mit den Anwendern verifiziert werden. Sind die Anforderungen bekannt (sofern es in diesem Fall überhaupt möglich ist, alle Anforderungen zu finden), können Sie dennoch nicht sofort mit der Erstellung des Programmcodes beginnen. Zunächst müssen Sie die einzelnen zu implementierenden Funktionen festlegen und das Zusammenspiel der Funktionen untereinander planen.

Diese Vorlesung beschäftigt sich mit oben dargestellten Thematiken.

Im ersten Teil des ersten Lehrbriefs werden die im Software-Entwicklungsprozeß durchzuführenden Aufgaben vorgestellt. Anhand zweier beispielhafter Vorgehensmodelle werden mögliche Abläufe beim Software-Entwicklungsprozeß beschrieben.

Der zweite Teil des ersten Lehrbriefs und der zweite Lehrbrief widmen sich klassischen Methoden zur Unterstützung der im Software-Entwicklungsprozeß durchzuführenden Aktivitäten.

Der dritte und vierte Lehrbrief führen neuere objektorientierte Lösungsansätze ein.

# 2 Software-Entwicklungsaktivitäten und Vorgehensmodelle

## 2.1 Aktivitäten

In Software-Entwicklungsprojekten müssen verschiedene Aktivitäten durchgeführt werden, so z. B:

- Ermittlung der Anforderungen der Anwender an die zu entwickelnde Software.
- Festlegung von Programmiersprache, Hardware-Umgebung, Betriebssysteme usw.
- Schreiben des Programm-Codes.
- Testen der Software.

Die Aktivität “Schreiben des Programm-Codes” ist sicherlich sehr viel konkreter, als Gespräche mit Anwendern zur Ermittlung der Anforderungen an das System. Andererseits erhält man bei Anwendergesprächen eine ganzheitlichere Sicht auf das zu entwickelnde System.

Ich werde nun die im Software-Entwicklungsprozeß durchzuführenden Aufgaben in fünf Abstraktionsebenen beschreiben. Dies bedeutet jedoch nicht, daß die folgenden Ebenen des Software-Entwicklungsprozesses sequentiell Top-Down durchlaufen werden müssen. Vielmehr sind durchaus andere Vorgehensweisen möglich und auch üblich (vgl. *Kapitel 2.2.1* und *Kapitel 2.2.2*).

### 2.1.1 Vorstudie (Einstieg)

Dem Einstieg in ein neues Projekt liegt immer ein erkannter Bedarf zugrunde. Dies kann eine Unterhaltung mit anschließender Aufgabenstellung beim Mittagessen sein, wie z.B.: “Können wir unsere interne Projekterfassung nicht über ein WWW-Interface abwickeln, damit wir unseren Designern neben ihrem Apple-Macintosh nicht auch noch einen Windows-PC für unser bisheriges Windows 95 basiertes Client-Server-System zur Vergügung stellen müssen?” bis hin zu komplexen Fragestellungen wie: “Unser Reisevertriebssystem wird den neuen Freizeit- und Reisegewohnheiten nicht mehr gerecht. Das Altsystem scheint ausgereizt und mit vernünftigem Aufwand nicht mehr erweiterbar. Wir müssen eine Neuentwicklung analysieren lassen.”

Umfang und Inhalt der Vorstudie hängen unmittelbar mit der Aufgabenstellung zusammen. Im Fall WWW-Projekterfassung könnte dies z.B. eine von folgenden Aussagen sein:

- Von der Umstellung würden nur unsere vier Designer profitieren. Allen anderen Mitarbeiter steht ein Windows 95 basierter PC zur Verfügung. Benutzer-Interface und Programmlogik unseres bestehenden Systems sind stark miteinander verflochten. Eine Umstellung unseres Systems würde ein völliges Redesign der Anwendung erfordern. Die Umstellung wird ca. zwei Mannmonate Arbeitszeit in Anspruch nehmen. Allerdings steht unseren Designern in ihrem Büro ein Windows 95 basierter PC zur Verfügung. Dieser Rechner wird auch weiterhin notwendig sein, da einige unserer Kunden Unterlagen in Formaten liefern, die nicht

direkt auf den Macintosh-Systemen lesbar sind und daher auf diesem Rechner in lesbare Formate umgesetzt werden. Ein PC reicht aber zur Projekterfassung für unsere Designer vollkommen aus.

- Die gleiche Problematik, die für unsere Design-Abteilung gesehen wurde, trifft auch für unsere Software-Entwickler und System-Administratoren zu. Beide Abteilungen arbeiten mit Linux-PC's und Sun-Rechnern. Insgesamt sind 50 Mitarbeiter betroffen. Benutzer-Interface und Programmlogik unseres bestehenden Systems sind strikt voneinander getrennt. Eine Umstellung des Benutzer-Interfaces auf WWW-Technologien wird unserer Meinung nach ca. zwei Manntage an Aufwand erfordern.

Im ersten Fall wird als Ergebnis der Vorstudie das Projekt wahrscheinlich abgelehnt, im zweiten Fall durchgeführt werden. In beiden Fällen kann die Vorstudie von einem Mitarbeiter in kurzer Zeit bearbeitet werden.

Anders sieht es im Falle eines Großprojektes wie der Analyse eines neuen Reisevertriebssystems für einen führenden Reiseveranstalter aus. Hierbei ist das Ergebnis des Einstiegs eine Machbarkeitsstudie, die durchaus 100 Seiten umfassen kann. Ich habe an der Vorstudie für ein Reisevertriebssystem mitgearbeitet, wo der Aufwand allein für die Vorstudie 1,5 Mannjahre betrug.

### 2.1.2 Analyse-Aktivitäten

Ziel der Analyse ist es, ein besseres Verständnis für das zu lösende Problem zu entwickeln. Es muß festgehalten werden, was die Anwendung im Einzelnen leisten soll. Dafür werden Anwendungsfälle definiert. Anwendungsfälle (auch Anforderungen genannt) beschreiben den Funktionsumfang des zu realisierenden Systems.

Ein Anwendungsfall eines Reisevertriebssystem ist z.B.:

Die Anwendung muß Pauschalreisen unterstützen. Eine Pauschalreise ist ein mit einem Flug verbundener Hotelaufenthalt. Der Hotelaufenthalt darf nicht ohne den Flug gebucht werden und der Flug nicht ohne den Hotelaufenthalt.

An diesem Beispiel sieht man, daß das Verständnis einer Anforderung das Verständnis der in der Anforderung enthaltenen Begriffe voraussetzt. Der Anwender, der die Anforderung formuliert, hat eine bestimmte Vorstellung von der Bedeutung der Begriffe Hotelaufenthalt und Flug. Mit Flug ist ein Hin- und Rückflug zum Urlaubsort (wo dann auch das Hotel sein sollte) gemeint, mit Hotelaufenthalt die Übernachtung in einem noch zu spezifizierenden Zimmertyp für eine zu spezifizierende Zeit. Das Verständnis der Anforderungen zwingt also zum Verständnis des Anwendungsgebietes.

Neben der Feststellung der Anforderungen ist die Abschätzung technologischer Risiken Bestandteil der Analyse-Aktivitäten. So kann eine technologische Festlegung Bestandteil der Anforderungen sein, z.B.:

Unser neues Reisevertriebssystem soll zukunftssicher entwickelt werden. Daher soll es objektorientiert in der Programmiersprache Java realisiert werden.

Hier muß nun geprüft werden:

- Kann ein so komplexes System in einer so neuen Programmiersprache realisiert werden, ist die Sprache performant genug, etc.?
- Stehen Mitarbeiter zur Verfügung, die schon Projekte mit Java durchgeführt haben oder zumindest Erfahrungen mit objektorientierter Entwicklung besitzen?

Bestandteile der Analyse sind also:

- Verständnis des Anwendungsgebietes.
- Erstellung des Anforderungskataloges.
- Abschätzung technologischer Risiken.

Insgesamt werden durch die Analyseaktivitäten die Kernanforderungen an das zu erstellende System erkannt. Ein Modell für das gewünschte Verhalten des Systems wird gebildet.

### 2.1.3 Entwurfs-Aktivitäten (DV-technisches Design)

In dieser Phase werden die Realisierungsentscheidungen getroffen. Dies beinhaltet u.a.:

- Wahl der Datenbanktechnologie (z.B. relational, objektorientiert)
- Auswahl der Datenbank (z.B. Oracle, Sybase, etc.)
- Auswahl des zugrundeliegenden Betriebssystems
- Auswahl der Anwendungsarchitektur (z.B. Client/Server oder 3-Ebenen-Architektur)
- Auswahl der Programmiersprache

sofern sie nicht durch strategische oder politische Unternehmensentscheidungen vorgegeben sind.

Darüberhinaus wird die Systemarchitektur entwickelt. Dies beinhaltet u.a.:

- Beschreibung des Funktionsmodells, des (logischen) Datenmodells (bzw. des Objektmodells) sowie schwieriger Algorithmen.
- Beschreibung des Zusammenspiels der Komponenten (oder Objekte).
- Schnittstellen zu anderen Anwendungen.

Insgesamt ist die Erzeugung einer Architektur für die Implementierung das Ziel der Entwurfsaktivitäten.

### 2.1.4 Realisierung

Durch die Realisierungsaktivitäten wird die im Entwurf dargestellte Anwendung in ein lauffähiges Programm umgesetzt. Dabei werden die im Entwurf definierten Komponenten detailliert und kodiert. Die einzelnen Komponenten werden getestet (Komponententest).

### 2.1.5 Integration

Die Integrationsaktivitäten umfassen u.a.:

- Integrationstest, d.h. die schrittweise Zusammenfügung der ausgetesteten Einzelkomponenten zum DV-Anwendungssystem.
- Systemtest, d.h. das DV-Anwendungssystem wird mit den vorgegebenen Testdaten getestet und damit auf den Abnahmetest vorbereitet
- Validierung der Anforderungsspezifikation
- Abnahmetest (unter Beteiligung der Fachabteilung)

## 2.2 Vorgehensmodelle

Die im vorherigen Kapitel dargestellten Aktivitäten müssen während des Software-Entwicklungsprozesses durchgeführt werden. Dabei werden bestimmte Aktivitäten zu sogenannten Phasen zusammengefaßt. Die Phaseneinteilung wiederum wird von Vorgehensmodellen (oder auch Prozeßmodellen) unterstützt. Grundsätzlich gibt es viele unterschiedliche Vorgehensmodelle. Gemeinsame Ziele aller Vorgehensmodelle sind:

- Gliederung des Gesamtprozesses in überschaubare Abschnitte zwecks Planung, Aufgabenverteilung und Kontrolle.
- Definition von aufeinander aufbauenden Ergebnissen.
- Minimierung der notwendigen Investitionen durch Festlegung von Entscheidungs- und Planungspunkten (Meilensteinen), an denen entschieden werden kann, ob das Software-Entwicklungs-Vorhaben fortgeführt werden soll.

### 2.2.1 Das Wasserfallmodell

Das Wasserfallmodell war das erste durchdachte Vorgehensmodell im Software-Engineering. Das Wasserfallmodell unterteilt den Software-Entwicklungsprozeß in eine Reihe von Phasen, die sequentiell durchlaufen werden. In jeder Phase werden Tätigkeiten gleichen Typs durchgeführt.

*Abbildung 1* zeigt eine mögliche Ausprägung des Wasserfallmodells. Die dort dargestellten fünf Phasen korrespondieren mit den im vorigen Kapitel vorgestellten Aktivitäten. In der Analyse-Phase (vgl. *Abbildung 1*) werden z.B. die in *Kapitel 2.1.2* dargestellten Analysetätigkeiten durchgeführt.

Jede Phase im Wasserfallmodell baut auf den Ergebnissen der vorhergehenden Phase auf. Am Ende einer jeden Phase kann entschieden werden, ob die nächste Phase in Angriff genommen werden soll (Projektfortsetzung) oder ob das Projekt abgebrochen werden soll.

Insbesondere aus Sicht des Managements schien das Wasserfallmodell eine attraktive Vorgehensweise zu sein. Es ist einfach zu verstehen und scheint eine gute Projektverfolgung und- koordinierung zu ermöglichen. So können Zeitpunkte festgelegt werden, an denen bestimmte Phasen abgeschlossen sein müssen (solche Zeitpunkte werden im folgenden Meilensteine genannt). Anhand der Erreichung der Meilensteine läßt sich kontrollieren, inwieweit ein Software-Entwicklungsprojekt seinen Zeitrahmen einhält.

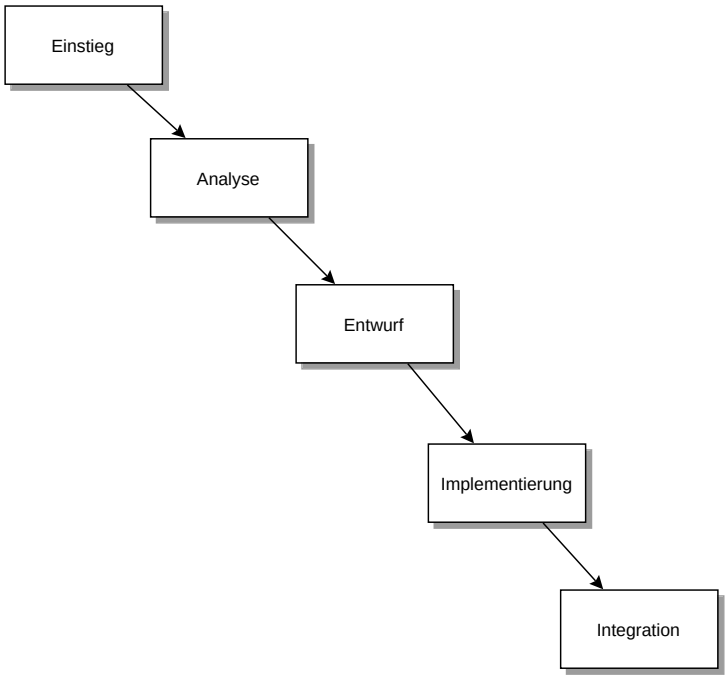


*Abbildung 1 Das Wasserfallmodell*

Das Wasserfallmodell in seiner ursprünglichen Form geht allerdings davon aus, daß die Ergebnisse jeder Phase korrekt sind. Stellt sich hingegen z.B. während des Entwurfs heraus, daß widersprüchliche Benutzeranforderungen existieren, gibt es nach diesem Modell keine Möglichkeit, solche Benutzeranforderungen zu verifizieren. Das Wasserfallmodell wurde daraufhin modifiziert. Wird während der Entwicklungsarbeit in einer Phase ein Fehler an den Ergebnissen einer vorgelagerten Phase entdeckt, werden die als falsch erkannten Ergebnisse an die vorgelagerte Phase zurückgegeben, die daraufhin erneut durchlaufen wird (vgl. *Abbildung 1*).

Im praktischen Einsatz des modifizierten Wasserfallmodells traten allerdings auch gravierende Mängel auf:

- Der Entwicklungsprozeß ist in dieser Form sehr aufwendig. Dies ist insbesondere der Fall, wenn die einzelnen Phasen von unterschiedlichen Teams bearbeitet werden. Erkannte Fehler müssen an ein anderes Team zurückgegeben werden, welches dann Korrekturen durchführen muß, obwohl die Team-Mitglieder mit hoher Wahrscheinlichkeit bereits an anderen Projekten arbeiten.

- Das Wasserfallmodell geht davon aus, daß alle Nutzeranforderungen in der Analyse-Phase erkannt werden. Meist ergeben sich aber im Projektverlauf neue Benutzeranforderungen, indem sich z.B. auch bei den Nutzern durch ihre Einbeziehung ein besseres Problemverständnis entwickelt und damit neue Anforderungen entstehen oder indem einfach durch während der Projektdurchführung vergehende Zeit neue Anforderungen an das zu entwickelnde System gestellt werden. So kann z.B. bei der Analyse der Anforderungen an ein Reisevertriebssystem keine Rede von Sitzplatzreservierungen bei der Buchung von Flugsitzen gewesen sein. Dann stellt ein Mitbewerber vor der Freigabe des neu-

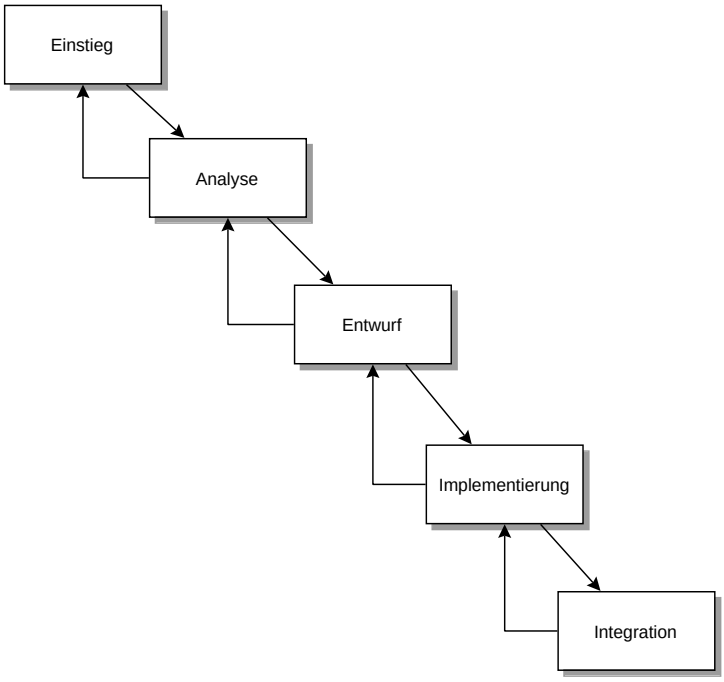


*Abbildung 2 Das modifizierte Wasserfallmodell*

en System dieses “Feature” in einer Werbekampagne heraus, und natürlich muß das neue System dann Sitzplatzreservierungen unterstützen.

- Risikobehaftete Tätigkeiten in der Realisierung oder die Integration finden erst zu einem sehr späten Zeitpunkt im Projektverlauf statt. Dies kann zu Nichteinhaltung von Terminen und Kostenexplosionen führen.

- Die strikt hierarchische Durchführungsorganisation kann zur Verfälschung der Benutzeranforderungen führen.

### 2.2.2 Inkrementell, iterative Modelle

Bei iterativ, inkrementellen Vorgehensmodellen wird die Software nicht “an einem Stück” am Ende des Projekts fertiggestellt, sondern in Teilen entwickelt und freigegeben. Das Software-System wird als eine Serie von aufeinander aufbauenden Teilprodukten mit ständig steigender Funktionalität (inkrementell) entwickelt.

Für jedes Teilsystem werden die in *Kapitel 2.1* dargestellten Aktivitäten durchgeführt (bis auf den Einstieg). Das Projekt ist also eine Folge wiederholter (iterativer) Durchführungen der Analyse-, Entwurfs-, Implementierungs- und Integrationsaktivitäten.

Die Anwender erhalten in einer frühen Phase Teile des Systems mit denen sie entweder bereits arbeiten oder doch zumindest ihre Anforderungen verifizieren können.

*Abbildung 1* zeigt ein Beispiel eines inkrementell, iterativen Prozeßmodells (vgl. [Fowler 1999]).

Die Aktivitäten der Phase Einstieg sind in *Kapitel 2.1.1* beschrieben. In der Analyse-Phase werden die Tätigkeiten aus *Kapitel 2.1.2* durchgeführt. Allerdings werden die

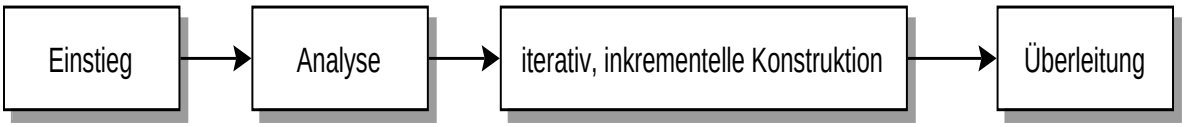


*Abbildung 3 Ein inkrementell, iteratives Modell (nach [Fowler 1999])*

Anwendungsfälle auf einer “abstrakten Ebene” erfaßt und erst während der Iteration detailliert. Danach wird jeder Anwendungsfall einer Iteration zugewiesen. Schwierige Anwendungsfälle oder Anwendungsfälle mit hoher Priorität sollten in frühen Iterationen abgehandelt werden.

In der Phase Konstruktion werden für jede Iteration sodann die Aktivitäten:

- Analyse der Anwendungsfälle der Iteration,
- Entwurf,
- Implementierung und
- Integration durchgeführt.

Jede Iteration wird mit einer Demonstration für die künftigen Benutzer und mit Systemtests, die die Korrektheit der in der Iteration abgebildeten Anwendungsfälle sicherstellen, abgeschlossen.

Die Anwender sehen also zu einem relativ frühen Projekt-Zeitpunkt Software-Teile, an denen sie feststellen können, ob ihre Anforderungen von den Entwicklern richtig verstanden wurden. Systemtests und Systemintegration sind Tätigkeiten, die nach dem Wasserfallmodell nur in späten Projektphasen durchgeführt werden, aber ein hohes Risiko beinhalten, so z.B.:

- Bei Tests gefundene Fehler müssen korrigiert werden.
- Das Projekt läuft aus dem Zeitrahmen, Tests werden verkürzt mit der Folge, daß fehlerhafte Software ausgeliefert wird, was wiederum zu hohen Wartungskosten führt.

Bei iterativ, inkrementellen Prozeßmodellen werden diese Tätigkeiten bei jeder Iteration durchgeführt, was das Risiko vermindert.

In der Überleitung wird das System beim Benutzer eingeführt.

Das Zusammenspiel zwischen Analyse und Iterationen veranschaulichen wir uns an folgendem Beispiel:

*Ausgangssituation:*

Ein Ingenieurbüro beschäftigt sich mit der Datenverarbeitung im Umweltschutz. Schwerpunkt bildet hier die Erzeugung von Grundwasserströmungs- und Stofftransportmodellen, mit deren Hilfe Aussagen und Prognosen bezüglich der Grundwasser-

quantität und -qualität einschließlich der Interaktion mit Oberflächengewässern gemacht werden können. Bedeutung haben diese Modelle beispielsweise für unterirdische Bauvorhaben (Tunnelbau, Tiefgaragen) und den Bergbau, indem die Auswirkungen auf den Grundwasserhaushalt erkannt und durch entsprechende Maßnahmen minimiert werden können.

Das Ingenieurbüro hat zur Berechnung dieser Grundwasserströmungs- und Transportmodelle das Programmsystem SPRING (**S**imulation of **Pr**ocesses **in G**roundwater) entwickelt. Diese Software erzeugt anhand von diversen Parametern zwei- und dreidimensionale Modelle, in denen durch graphische Elemente (z.B. Iso-Linien, Flächenfärbung) die entsprechenden Grundwasserdaten dargestellt werden.

Bei der Abwicklung von größeren Projekten müssen diese Darstellungen Bearbeitern aus unterschiedlichen Fachdisziplinen, unterschiedlichen Firmen (oder Verwaltungen) mit unterschiedlichen Zielsetzungen zugänglich gemacht werden. Aufgrund deren Beurteilungen sind oftmals neue Berechnungen bis hin zu Modelländerungen erforderlich, bis das endgültige Ergebnis verabschiedet werden kann.

Z. Zt. werden Ausdrucke (Plots) der Ergebnisse erstellt und versendet. Diese Vorgehensweise führt nicht nur zum Zeitverlust durch den Posttransport. Vielfach sind aufgrund der Plots neue vergrößernde Darstellungen erforderlich, die dann erst beim Ingenieurbüro berechnet und erneut versendet werden müssen.

*Einstieg:*

Der zeitaufwendige Transport über den Postweg soll durch Übertragung und Darstellung der Visualisierungen über das Internet ersetzt werden. Dabei soll den Bearbeitern die Möglichkeit gegeben werden, die Darstellungsform der Ergebnisse interaktiv zu ändern (z.B. Farben der Iso-Linien, vergrößerte oder verkleinerte Darstellungen).

Die Voranalyse ergab, daß diese Aufgabenstellung grundsätzlich lösbar ist. Dazu ist das Software-Paket des Ingenieurbüros modular aufgebaut. Es erzeugt u.a. eine Plotdatei mit allen Darstellungs-Informationen, auf die von der zu konzipierenden Anwendung zugegriffen werden kann. Daher wird die Anwendungs-Entwicklung wahrscheinlich nicht zu aufwendig.

*Analyse:*

In der Analyse wurden u.a. folgende Anforderungen an das System festgelegt:

1 Die Plotdatei muß eingelesen, über das Internet übertragen und in einem Browser dargestellt werden.

2 Eine Lupenfunktion muß vorhanden sein.

3 Die Berechnung für die Lupenfunktion muß im Browser durchgeführt werden, damit die Plotdatei nur einmal übertragen werden muß.

4 Darstellungsparameter müssen eingegeben werden können.

5 Benutzerberechtigungen müssen vergeben und kontrolliert werden.

6 Jeder Zugriff auf die Darstellungsfunktion muß protokolliert werden.

*Iteration:*

Iteration 1

Zuerst wurde Anforderung 1 realisiert. Dazu mußte derAufbau der Plotdatei analysiert werden (Analyse in der Iteration). Danach wurde das Objektmodell zur Aufnahme der Plotdaten entwickelt (Entwurf in der Iteration) und implementiert (Implementierung in der Iteration). Das Modell für die Übertragung zum Client (Browser) und das Grafikmodul für die Darstellung wurden entwickelt und implementiert. Die einzelnen Teilmodule wurden integriert (Integration in der Iteration).

Iteration 2

In der zweiten Iteration wurde die Lupenfunktion (Anforderungen 2 und 3) realisiert. In Zusammenarbeit mit dem Ingenieurbüro wurden die genauen Anforderungen an die Lupenfunktion erarbeitet (Analyse), konzipiert und implementiert.

Iteration 3

In der dritten Iteration wurden die Benutzeridentifikation und Zugriffsprotokollierung (Anforderungen 5 und 6) realisiert. Hierzu wurden wieder in Zusammenarbeit mit dem Ingenieurbüro die genauen Forderungen an die Struktur der dieser Funktionalität zugrundeliegenden Datenbank ermittelt. Danach wurde die Datenbank modelliert und implementiert.

Iteration 4

Analyse, Konzipierung und Implementierung von Anforderung 4 schlossen das Projekt ab.

Nach jeder Iteration wurde das dahin erstellte System dem Ingenieurbüro präsentiert.

## 2.3 Weiterführende Literatur:

Grady Booch (1994): Objektorientierte Analyse und Design, Addison Wesley, Bonn

Andreas Frick (1995): Der Software-Entwicklungsprozeß: Ganzheitliche Sicht, Hanser Verlag, München

Martin Fowler, Kendall Scott (1999): UML konzentriert, Addison Wesley, Bonn

Gunter Müller-Ettrich (1999): Objektorientierte Prozeßmodelle, Addison Wesley, Bonn

# 3 Ausgewählte Techniken des Software-Engineering

In diesem Kapitel werden nun ausgewählte Methoden des Software-Engineering, die bei den im vorherigen Kapitel vorgestellten Aktivitäten bei der Software-Entwicklung eingesetzt werden können, vorgestellt. Dieses Kapitel beschränkt sich zunächst auf klassische Methoden. Neuere objektorientierte Techniken werden in den Lehrbriefen drei und vier behandelt.

## 3.1 Entscheidungstabellen

Entscheidungstabellen sind Hilfsmittel, die bei allen Aktivitäten des Software-Entwicklungsprozesses eingesetzt werden können. Entscheidungstabellen entstanden 1957 in einer Projektgruppe der General Electric Company und beschreiben komplexe Entscheidungssituationen.

Aufgrund ihrer übersichtlichen Darstellung solcher Entscheidungssituationen sind sie ein excellentes Verständigungsmittel zwischen Software-Entwicklern und Anwendern.

Entscheidungstabellen bestehen aus Bedingungen, Bedingungsanzeiger, Aktionen, Aktionsanzeiger und Regeln (vgl. *Abbildung 4*).

| Bedingungen | Bedingungsanzeiger | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Bedingung 1 | J | J | J | J | N | N | N | N |
| Bedingung 2 | J | J | N | N | J | J | N | N |
| Bedingung 3 | J | N | J | N | J | N | J | N |
| Aktionen | Aktionsanzeiger | | | | | | | |
| Aktion 1 | x | x | x | | | x | x | x |
| Aktion 2 | x | | | x | x | | x | |

*Abbildung 4 Aufbau einer Entscheidungstabelle (3 Bedingungen, 2 Aktionen*

Bevor wir allgemeine Eigenschaften von Entscheidungstabellen betrachten, werden wir uns die Erstellung einer Entscheidungstabelle anhand eines Beispiels veranschaulichen.

### 3.1.1 Beispiel für das Erstellen einer Entscheidungstabelle

*Verbale Problembeschreibung (Ausführen von Überweisungen)*

Wenn die vereinbarte Kreditgrenze des Auftraggebers einer Überweisung überschritten wird, das bisherige Zahlungsverhalten des Auftraggebers aber einwandfrei war und der Überschreitungsbetrag kleiner oder gleich 1000,- DM ist, dann soll die Überweisung durchgeführt werden.

Liegt der Überschreitungsbetrag über 1000,- DM und das Zahlungsverhalten war bisher einwandfrei, soll die Überweisung durchgeführt werden, der Kunde aber gleichzeitig brieflich von dem Vorgang unterrichtet werden.

War das Zahlungsverhalten nicht einwandfrei und ist die Kreditgrenze überschritten, wird die Überweisung nicht durchgeführt.

Selbstverständlich wird die Überweisung durchgeführt, wenn der Kreditbetrag nicht überschritten ist.

*Umsetzung in Aktionen und Bedingungen*

Zunächst werden wir die Bedingungen ermitteln. Die Bedingungen dieses Textes sind:

- Kreditgrenze überschritten?
- Zahlungsverhalten einwandfrei?
- Überschreitungsbetrag <= 1000,- DM?

Danach stellen wir die Aktionen fest. In diesem Beispiel handelt es sich um:

- Überweisung durchführen
- Überweisung nicht durchführen
- Briefliche Benachrichtigung des Kunden

In der Entscheidungstabelle wird nun festgelegt, wie die Aktionen von den Bedingungen abhängen. Dazu werden die Bedingungen zunächst in den Bedingungsteil und die Aktionen in den Aktionsteil der Entscheidungstabelle übernommen (vgl. *Abbildung 4* und *Abbildung 5*).

Danach werden alle möglichen Bedingungskombinationen gebildet. Dies geschieht, indem die Bedingungsanzeiger im Bedingungsanzeigerteil der Entscheidungstabelle (vgl. *Abbildung 4* und *Abbildung 5*) auf J (Bedingung erfüllt), bzw. N (Bedingung nicht erfüllt). Da jede Bedingung zwei Werte annehmen kann (J und N) und es insgesamt drei Bedingungen gibt, ergeben sich insgesamt $2^3 = 8$ mögliche Kombinationen.

Nun wird im Aktionsanzeigerteil der Entscheidungstabelle für jede Aktion, die bei der jeweiligen Bedingungsanzeigerkombination durchgeführt wird, ein Kreuz gesetzt (Aktionsanzeiger). Eine Bedingungsanzeigerkombination zusammen mit ihren Aktionsanzeigern heißt Regel. Da es in unserem Beispiel acht Bedingungsanzeigerkombinationen gibt, existieren hier acht Regeln (vgl. *Abbildung 5* R1 bis R8).

*Abbildung 5* führt eine weitere in unser ersten Analyse nicht vorhandene Aktion, nämlich die Aktion “unlogisch” ein. Dies ergibt sich dadurch, daß Bedingungsanzeigerkombinationen in unserem Beispiel zu einem Widerspruch führen. Betrachten wir R8 in *Abbildung 5*. Hier stehen u.a. die Bedingungsanzeiger der Bedingungen “Kreditgrenze überschritten” und “Überschreitungsbetrag <= 1000,- DM” auf “N”. Dies bedeutet, die Kreditgrenze wurde **nicht** überschritten, gleichzeitig ist der Überschreitungsbetrag aber **größer** als 1000,- DM. Dies kann nicht sein, diese Bedingungsanzeigerkombination ist daher unlogisch.

| | R1 | R2 | R3 | R4 | R5 | R6 | R7 | R8 |
|---|---|---|---|---|---|---|---|---|
| Kreditgrenze überschritten | J | J | J | J | N | N | N | N |
| Zahlungsverhalten einwandfrei | J | J | N | N | J | J | N | N |
| Überschreitungsbetrag <= 1000,- DM | J | N | J | N | J | N | J | N |
| Überweisung durchführen | X | X | | | X | | X | |
| Überweisung nicht durchführen | | | X | X | | | | |
| Briefliche Benachrichtigung | | X | | | | | | |
| unlogisch | | | | | | X | | X |

*Abbildung 5 Entscheidungstabelle des Überweisungsbeispiels*

### 3.1.2 Eindeutigkeit von Entscheidungstabellen

Entscheidungstabellen können durch Formulierung der Bedingungen und Aktionen auf mehrere Arten aufgestellt werden. So könnten die Bedingungen unseres Beispiels auch auf folgende Weise notiert werden:

- Kreditgrenze eingehalten?
- Zahlungsverhalten einwandfrei?
- Überschreitungsbetrag > 1000,- DM?

Bei den Aktionen könnte “Überweisung nicht durchführen” weggelassen werden, da kein Kreuz bei “Überweisung durchführen” als “Überweisung nicht durchführen” verstanden werden kann. Dies würde zu folgender Entscheidungstabelle führen:

| | R1 | R2 | R3 | R4 | R5 | R6 | R7 | R8 |
|---|---|---|---|---|---|---|---|---|
| Kreditgrenze eingehalten | J | J | J | J | N | N | N | N |
| Zahlungsverhalten einwandfrei | J | J | N | N | J | J | N | N |
| Überschreitungsbetrag > 1000,- DM | J | N | J | N | J | N | J | N |
| Überweisung durchführen | | X | | X | X | X | | |
| Briefliche Benachrichtigung | | | | | X | | | |
| unlogisch | X | | X | | | | | |

*Abbildung 6 Alternative Entscheidungstabelle des Überweisungsbeispiels*

Während die Bedingungsformulierungen sicher keinen Anteil an der Verständlichkeit der Entscheidungstabelle haben, führt das Weglassen der (redundanten) Aktion “Überweisung nicht durchführen” zu zwei Regeln ohne Aktionsanzeiger (R7 und R8 in *Abbildung 5*). Die Aufgabenstellung wird durch das Hinzufügen dieser Aktion sicher klarer beschrieben.

### 3.1.3 Vorgehensweise bei der Erstellung von Entscheidungstabellen

Ich fasse hier die Vorgehensweise bei der Erstellung einer Entscheidungstabelle und die bis jetzt behandelten Begrifflichkeiten zusammen.

Zunächst werden die Bedingungen ermittelt, aufgelistet und in den Bedingungsteil der Entscheidungstabelle übernommen.

Danach werden die Aktionen ermittelt, aufgelistet und in den Aktionsteil der Entscheidungstabelle übernommen.

Alle formal möglichen Bedingungsanzeigerkombinationen werden gebildet. Dies geschieht durch Setzen der Bedingungsanzeiger auf J bzw. N. Bei n Bedingungen ergeben sich $2^n$ Bedingungsanzeigerkombinationen.

Für jede Bedingungsanzeigerkombination werden die Aktionsanzeiger durch ankreuzen der für diese Bedingungsanzeigerkombination durchzuführenden Aktionen gesetzt. Eine Bedingungsanzeigerkombination mit zugehörigen Aktionsanzeigern heißt Regel. Die Regeln sind also die Spalten der Entscheidungstabelle. Da es $2^n$ Bedingungsanzeigerkombinationen (bei n Bedingungen) gibt, gibt es in diesem Fall auch $2^n$ Regeln.

### 3.1.4 Konsolidierung (Verdichtung) von Entscheidungstabellen

Betrachten wir noch einmal die Entscheidungstabelle in *Abbildung 5*. Wir sehen uns hier die Regeln R3 und R4 an. Bei beiden Regeln sind die Aktionsanzeiger identisch gesetzt (Aktion “Überweisung nicht durchführen”). Betrachten wir nun die Bedingungsanzeiger:

Für die Bedingung “Kreditgrenze überschritten” stehen die Bedingungsanzeiger beider Regeln auf J, für “Zahlungsverhalten einwandfrei” hingegen auf N. Sie stimmen also auch überein. Der Bedingungsanzeiger von “Überschreitungsbetrag <= 1000,- DM” steht in Regel R3 auf J, in Regel R4 hingegen auf N.

Wir sehen also: Wenn die Bedingung “Kreditgrenze überschritten” erfüllt und die Bedingung “Zahlungsverhalten einwandfrei” nicht erfüllt ist, dann wird die Aktion “Überweisung nicht durchführen” durchgeführt, egal, ob “Überschreitungsbetrag <= 1000,- DM” erfüllt ist oder nicht. Die durchzuführende Aktion ist also bereits durch die Werte der Bedingungsanzeiger der beiden ersten Bedingungen festgelegt, “Überschreitungsbetrag <= 1000,- DM” spielt unter der gegebenen Konstellation der anderen beiden Bedingungen keine Rolle mehr.

In unserer Entscheidungstabelle kann dies durch Zusammenfassen der Regeln R3 und R4 zu einer Regel (Ra in *Abbildung 7*) deutlich gemacht werden. Anstelle J in R3 und N in R4 für den Bedingungsanzeiger von “Überschreitungsbetrag <= 1000,- DM” wird in der zusammengefaßten Regel Ra der Irrelevanz-Anzeiger “-” gesetzt. Hier-

durch wird deutlich, daß die auszuführenden Aktionen von der Bedingung, für die der Irrevelanzanzeiger gesetzt ist, nicht mehr abhängen.

Wenn wir unsere Entscheidungstabelle in *Abbildung 5* weiter analysieren, sind zwei weitere Irrelevanzen feststellbar:

- Wenn die Kreditgrenze nicht überschritten ist und der Überschreitungsbetrag kleiner als 1000,- DM ist, wird überwiesen, egal wie das Zahlungsverhalten war (Regeln R5 und R7 in *Abbildung 5* werden zu Rb in *Abbildung 7* zusammengefaßt.

- Wenn die Kreditgrenze nicht überschritten ist und der Überschreitungsbetrag größer als 1000,- DM ist, dann ist das unlogisch, egal wie das Zahlungsverhalten war (Regeln R6 und R8 in *Abbildung 5* werden zu Rc in *Abbildung 7* zusammengefaßt.

| | R1 | R2 | Ra | Rb | Rc |
|---|---|---|---|---|---|
| Kreditgrenze überschritten | J | J | J | N | N |
| Zahlungsverhalten einwandfrei | J | J | N | - | - |
| Überschreitungsbetrag <= 1000,- DM | J | N | - | J | N |
| Überweisung durchführen | X | X | | x | |
| Überweisung nicht durchführen | | | X | | |
| Briefliche Benachrichtigung | | X | | | |

*Abbildung 7 Konsolidierte Entscheidungstabelle des Überweisungsbeispiels*

Zusammenfassend können wir sagen:

Zwei Regeln einer Entscheidungstabelle können zusammengefaßt (konsolidiert) werden, wenn sie identische Aktionsfolgen beinhalten und sich im Bedingungsanzeigerteil an höchstens einer Stelle (Bedingung) unterscheiden. Die zusammengefaßte Regel erhält bei dieser Bedingung dann einen Irrelevanzanzeiger “-”.

Eine Regel, die nur J oder N enhält, nennt man einfache Regel, während duch die Konsolidierung komplexe Regeln entstehen.

### 3.1.5 Kaskadierte Entscheidungstabellen

Entscheidungstabellen mit vielen Regeln werden schnell unübersichtlich. Bei fünf Bedingungen erhält man z.B. $2^5 = 32$ Regeln. Um die Übersichtlichkeit zu bewahren, können Entscheidungstabellen kaskadiert werden. Dies bedeutet, daß im Aktionsteil weitere Entscheidungstabellen referenziert werden können.

Ich erläutere den Sachverhalt an folgendem Beispiel :

Ein Unternehmen wechselt seine Lieferanten nach folgendem Vorgehensmuster:

Werden von einem Lieferanten die Qualitätsnormen des Unternehmens nicht erfüllt, erhält dieser grundsätzlich keine Bestellungen mehr. Werden die Qualitätsnormen erfüllt, die Liefertermine jedoch überzogen, so erhält der Lieferant nur dann eine weitere Chance, wenn die bestellte Menge zum vereinbarten Preis geliefert wird. In diesem Fall werden jedoch bei künftigen Bestellungen Vergleichsangebote eingeholt. In allen anderen Fällen erhält der Lieferant bei Überziehung der Liefertermine keine weiteren Aufträge mehr. Werden auch die Liefertermine eingehalten, wird der Lieferant beibehalten, wenn die bestellte Menge zum bestellten Preis geliefert wird. Wird zum Liefertermin nicht die gesamte Liefermenge ausgeliefert, oder ein höherer Preis verlangt, werden bei künftigen Bestellungen Vergleichsangebote eingeholt.

Eine mögliche Lösung mit kaskadierten Entscheidungstabellen ist in *Abbildung 9* dargestellt.

| ET A | R1 | R2 | R3 | R4 |
|---|---|---|---|---|
| Termintreue ? | J | J | N | N |
| Qualitätsnormen erfüllt ? | J | N | J | N |
| Keine Bestellungen mehr | | X | | X |
| ET B durchführen | X | | | |
| ET C durchführen | | | X | |

| ET B | R1 | R2 | R3 | R4 |
|---|---|---|---|---|
| Preistreue ? | J | J | N | N |
| Mengentreue ? | J | N | J | N |
| Beibehaltung | X | | | |
| Vergleichsangebot | | X | X | X |

| ET C | R1 | R2 | R3 | R4 |
|---|---|---|---|---|
| Preistreue ? | J | J | N | N |
| Mengentreue ? | J | N | J | N |
| Keine Bestellungen mehr | | X | X | X |
| Vergeleichsangebot | X | | | |

*Abbildung 8 kaskadierte Entscheidungstabelle*

### 3.1.6 Überprüfung von Entscheidungstabellen

Eine weitere schöne Eigenschaft von Entscheidungstabellen ist, daß einige formale Kriterien für die Richtigkeit von Entscheidungstabellen überprüfbar sind. Formale Richtigkeit bedeutet natürlich nicht inhaltliche Richtigkeit.

*Vollständigkeit*

Im Rahmen des Vollständigkeitstests kann für eine Entscheidungstabelle geprüft werden, ob jede mögliche Bedingungsanzeigerkonstellation für die erkannten Bedingungen durch eine der vorhandenen Regeln abgedeckt wird. Dies läßt sich relativ einfach ausrechnen. Wie bereits in *Kapitel 3.1.3* dargestellt, muß eine Entscheidungstabelle mit n Bedingungen $2^n$ Regeln beinhalten. Dies läßt sich durch einfaches Abzählen verifizieren.

Selbst bei bereits konsolidierten (vgl. *Kapitel 3.1.4*) Entscheidungstabellen ist dies möglich. Regeln mit einem Irrelevanzanzeiger entsprechen zwei einfachen Regeln. Regeln mit zwei Irrelevanzanzeigern 4, Regeln mit n Irrelevanzanzeigern $2^n$ Regeln. *Abbildung 9* zeigt eine unvollständige konsolidierte Entscheidungstabelle.

| | R1 | R2 | R3 | R4 |
|---|---|---|---|---|
| B1 | J | N | J | N |
| B2 | J | - | J | J |
| B3 | J | J | N | - |
| A1 | X | | X | X |
| A2 | | X | X | X |

*Abbildung 9* *Unvollständige Entscheidungstabelle (Nur 6 statt 8 Regeln sind abgebildet, R2 und R4 beinhalten durch den Irrelevanzanzeiger zwei Regeln)*

*Redundanz*

Zwei Regeln sind redundant, wenn sie die gleichen Aktionsfolgen beinhalten und wenn es eine Bedingungskonstellation gibt, die durch beide Regeln abgedeckt wird (vgl. *Abbildung 9*). Redundanzen decken Fehler beim Konsolidieren auf. R1 kann es in *Abbil-*

*dung 9* eigentlich nicht mehr geben, denn R1 müßte beim Konsolidieren eigentlich in R2 aufgegangen sein.

| | R1 | R2 | R3 | R4 |
|---|---|---|---|---|
| B1 | J | J | J | N |
| B2 | J | - | J | - |
| B3 | J | J | N | - |
| | | | | |
| A1 | | | X | X |
| A2 | X | X | X | |

*Abbildung 10* *Redundante Entscheidungstabelle (Regeln R1 und R2 sind redundant, denn R1 ist in R2 enthalten)*

*Widerspruchsfreiheit*

Ein Widerspruch liegt vor, wenn es zwei Regeln gibt, die beide eine gemeinsame Bedingungskonstellation abdecken, aber unterschiedliche Aktionsfolgen haben. Dem Grund von Widersprüchen ist nachzugehen, da sie beseitigt werden müssen (vgl. *Abbildung 9*).

| | R1 | R2 | R3 | R4 |
|---|---|---|---|---|
| B1 | J | J | J | N |
| B2 | J | - | J | - |
| B3 | J | J | N | - |
| | | | | |
| A1 | | X | X | X |
| A2 | X | | X | |

*Abbildung 11* *Entscheidungstabelle mit Widerspruch (Regeln R1 und R2 sind widersprüchlich, denn R2 kann auch JJJ sein, aber die Aktionen in R1 und R2 sind unterschiedlich)*

### 3.1.7 Erweiterte Entscheidungstabellen

Bei den bisher betrachteten Entscheidungstabellen werden die Bedingungen so formuliert, daß im rechten oberen Quadranten, also dem Bedingungsanzeigeteil, nur die Eintragungen “J” für “ja”, “N” für “nein” und “-” für “nicht von Bedeutung” (irrelevant) und im Aktionsanzeigeteil nur die Eintragungen “x” für “Aktion ausführen” bzw. eine Leerstelle für “Aktion nicht ausführen” möglich sind. Diese Art Entscheidungstabelle heißt auch begrenzte Entscheidungstabelle.

Bei einer erweiterten Entscheidungstabelle sind die Bedingungen und Aktionen so formuliert, daß im Bedingungs- und Aktionsanzeigeteil nicht nur “J”, “N”, “-” bzw. “x” und “ “ stehen, sondern ganz oder teilweise konkrete Werte (vgl. *Abbildung 9*).

| Arbeitsgang Vorbohren | | | | |
|---|---|---|---|---|
| Durchmesser d der Bohrung in mm | <=10 | >10 u. <=50 | >10 u. <=50 | > 50 |
| Werkstoff | - | Al | St | - |
| Vorbohren mit einem Boher mit dem Durchmesser | - | 0,5 *d | 0,8 *d | 30mm |
| Weitere Prüfung nach ET | 18 | 12 | 12 | 20 |

*Abbildung 12 Erweiterte Entscheidungstabelle*

### 3.1.8 Vorteile von Entscheidungstabellen

Entscheidungstabellen sind leicht lesbar und unmißverständlich. Sie sind damit ein hervorragendes Verständigungsmittel zwischen Systemanalytikern, Programmierern und Mitarbeitern aus den Fachabteilungen.

Sie erlauben die systematische Analyse und einfache Darstellung komplexer Entscheidungssituationen.

Formale Fehler in begrenzten Entscheidungstabellen können durch

- Vollständigkeitstest
- Redundanztest
- Widerspruchstest

aufgedeckt werden.

### 3.1.9 Selbsttestaufgabe

Das alteingessene stahlverarbeitende Unternehmen Rost-Stahl & Co KG entschließt sich, im Rahmen von Rationalisierungsmaßnahmen die Reisekostenerstattungsverordnung zu vereinfachen. Folgender Vorschlag wird von der Verwaltung des Unternehmens erarbeitet und dem Vorstand zugeleitet:

Manager ab Abteilungsleiter aufwärts dürfen immer einen Tag vor Beginn des Dienstgeschäftes am Ort des Dienstgeschäftes anreisen. Andere Angestellte dürfen nur dann einen Tag vorher am Ort des Dienstgeschäftes anreisen, wenn die Dauer der für

das Dienstgeschäft erforderlichen Reise vom Ort des Unternehmenssitzes zum Ort des Dienstgeschäftes 4 Stunden überschreitet.

Manager ab Abteilungsleiter aufwärts dürfen Hotels beliebiger Preisklasse am Ort des Dienstgeschäftes benutzen. Andere Angestellte müssen am Ort des Dienstgeschäftes Hotels der Preisklasse bis zu 120,-- DM benutzen. Eine Ausnahme ist gegeben, wenn in einem Umkreis von 30 km um den Ort des Dienstgeschäftes kein Hotel dieser Preisklasse vorhanden ist. Dann dürfen auch diese Angestellten am Ort des Dienstgeschäftes ein Hotel beliebiger Preisklasse benutzen.

Manager ab Abteilungsleiter aufwärts dürfen immer ein Flugzeug zur Abwicklung ihrer Reise vom Ort des Unternehmenssitzes zum Ort des Dienstgeschäftes nutzen. Andere Angestellte dürfen nur dann mit dem Flugzeug vom Ort des Unternehmenssitzes zum Ort des Dienstgeschäftes reisen, wenn die Entfernung vom Ort des Unternehmenssitzes zum Ort des Dienstgeschäftes größer als 759 km ist.

Manager ab Abteilungsleiter aufwärts dürfen nach Erledigung des Dienstgeschäftes noch einen weiteren Tag am Ort des Dienstgeschäftes verbringen. Dies gilt aber auch für Manager ab Abteilungsleiter aufwärts nur dann, wenn die Reise vom Ort des Dienstgeschäftes zum Ort des Unternehmenssitzes länger als 4 Stunden dauert oder die Entfernung vom Ort des Dienstgeschäftes zum Ort des Unternehmenssitzes größer als 759 km ist. Andere Mitarbeiter müssen am Tag der Erledigung des Dienstgeschäftes die Reise vom Ort des Dienstgeschäftes zum Ort des Unternehmenssitzes antreten. Eine Ausnahme ist gegeben, wenn die Entfernung vom Ort des Dienstgeschäftes zum Ort des Unternehmenssitzes größer als 759 km ist und die Reise vom Ort des Dienstgeschäftes zum Ort des Unternehmenssitzes länger als 4 Stunden dauert. Dann dürfen auch diese Mitarbeiter nach Erledigung des Dienstgeschäftes einen weiteren Tag am Ort des Dienstgeschäftes verbleiben, bevor sie ihre Rückreise zum Ort des Unternehmenssitzes antreten müssen.

Allein: Die Logik dieses Vorschlags wird vom Management des Unternehmens Rost-Stahl & Co KG nicht recht verstanden. Daher sollen Sie eine vollständige Entscheidungstabelle dieses Vorschlags erstellen. Danach sollen Sie diese Entscheidungstabelle konsolidieren.

### 3.1.10 Lösung der Selbsttestaufgabe

| | R1 | R2 | R3 | R4 | R5 | R6 | R7 | R8 | R9 | R10 | R11 | R12 | R13 | R14 | R15 | R16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Manager ab Abteilungs-leiter | J | J | J | J | J | J | J | J | N | N | N | N | N | N | N | N |
| Reiselänge > 4 Stunden | J | J | J | J | N | N | N | N | J | J | J | J | N | N | N | N |
| Hotel bis 120 DM im Umkreis bis zu 30 km | J | J | N | N | J | J | N | N | J | J | N | N | J | J | N | N |
| Entfernung größer 759 km | J | N | J | N | J | N | J | N | J | N | J | N | J | N | J | N |
| Anreise 1 Tag eher | X | X | X | X | X | X | X | X | x | x | x | x | | | | |
| Hotels über 150 DM | X | X | X | X | X | X | X | X | | | x | x | | | x | x |
| Anreise mit Flugzeug | X | X | X | X | X | X | X | X | x | | x | | x | | x | |
| Abreise 1 Tag später | x | x | x | x | x | | x | | x | | x | | | | | |

*Abbildung 13 Entscheidungstabelle der Selbsttestaufgabe*

| | R1 / R2 | R3/ R4 | R5/ R7 | R6/ R8 | R9 | R10 | R11 | R12 | R13 | R14 | R15 | R16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Manager ab Abteilungs-leiter | J | J | J | J | N | N | N | N | N | N | N | N |
| Reiselänge > 4 Stunden | J | J | N | N | J | J | J | J | N | N | N | N |
| Hotel bis 120 DM im Umkreis bis zu 30 km | J | N | - | - | J | J | N | N | J | J | N | N |
| Entfernung größer 759 km | - | - | J | N | J | N | J | N | J | N | J | N |
| Anreise 1 Tag eher | x | x | x | x | x | x | x | x | | | | |
| Hotels über 150 DM | x | x | x | x | | | x | x | | | x | x |
| Anreise mit Flugzeug | x | x | x | x | x | | x | | x | | x | |
| Abreise 1 Tag später | x | x | x | | x | | x | | | | | |

*Abbildung 14 Erste Konsolidierung der Selbsttestaufgabe*

| | R1 /R2 R3/R4 | R5/ R7 | R6/ R8 | R9 | R10 | R11 | R12 | R13 | R14 | R15 | R16 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Manager ab Abteilungs-leiter | J | J | J | N | N | N | N | N | N | N | N |
| Reiselänge > 4 Stunden | J | N | N | J | J | J | J | N | N | N | N |
| Hotel bis 120 DM im Umkreis bis zu 30 km | - | - | - | J | J | N | N | J | J | N | N |
| Entfernung größer 759 km | - | J | N | J | N | J | N | J | N | J | N |
| Anreise 1 Tag eher | x | x | x | x | x | x | x | | | | |
| Hotels über 150 DM | x | x | x | | | x | x | | | x | x |
| Anreise mit Flugzeug | x | x | x | x | | x | | x | | x | |
| Abreise 1 Tag später | x | x | | x | | x | | | | | |

*Abbildung 15 Zweite Konsolidierung der Selbsttestaufgabe*

## 3.2 Strukturierte Analyse

Die Strukturierte Analyse ist eine Methode, die insbesondere in der Analyse-Phase eingesetzt wird. Eigentlich besteht die Strukturierte Analyse aus drei miteinander verknüpften Methoden:

- Datenflußdiagramme.
- Datenkataloge.
- Prozeßspezifikationen.

Die Methode der Strukturierten Analyse hat sich seit der Veröffentlichung des Standardwerks “Structured Analysis and System Specification” von Tom DeMarco in den siebziger Jahren zu einem Standard bei der Systemanalyse entwickelt und ist Bestandteil vieler Software-Entwicklungs-Tools (CASE-Tools). Ein besonderer Vorteil der Strukturierten Analyse ist die einfache Modellnotation, die sich intensiv grafischer Objekte bedient, um ein Modell sowohl für Anwender als auch für Entwickler übersichtlich zu gestalten. Eine Beschreibung der Anwendung in freiem Fließtext, die häufig unverständlich und mehrdeutig ist, wird vermieden.

### 3.2.1 Datenflußdiagramme

Elementare Aufgabe jeder Software-Anwendung ist es, Eingaben in Ausgaben zu transformieren. So kann die Eingabe eines Buchungsvorgangs bei einem Reisevertriebssystem zu verschiedenen Ausgaben wie Flugscheinen, Hotelvouchers oder auch zu einer Meldung wie “gewünschte Reise ausgebucht” führen. Datenflußdiagramme sind eine Serie von Grafiken, die versuchen, diesen Tatbestand in einem Modell nachzubilden.

Wir veranschaulichen uns Datenflußdiagramme und die Vorgehensweise zu ihrer Erstellung zunächst an einem Beispiel:

#### 3.2.1.1Beispiel für Datenflußdiagramme:

Eine Anwendung zur Unterstützung der Klausuranmeldungen für Studenten soll erstellt werden. Dazu sollen zunächst zur Mitte des Vorsemesters die für das nächste Semester geplanten Klausuren in das System übernommen werden. Diese Eingaben (incl. Raum der Klausur, Länge der Klausur, Klausurtag etc.) werden vom Vorsitzenden des Prüfungsausschusses durchgeführt.

Wenn alle Klausuren erfasst worden sind, erzeugt das System zwei Prüfungspläne:

- Der Prüfungsplan für die Professoren und Mitarbeiter beinhaltet die Aufsichtsführenden der jeweiligen Klausur und den jeweiligen Erst- und Zweitprüfer. Er wird allen Professoren und Mitarbeitern zugestellt.
- Den Prüfungsplan für die Studenten ohne diese Informationen. Dieser Prüfungsplan wird den Studenten zugestellt.

Studenten können sich sodann an speziellen Terminals für ihre Klausuren anmelden. Das System überprüft, ob die Studenten für die Klausur zugelassen sind und ob es die Klausur überhaupt gibt. Zur Überprüfung der Vorleistung wird eine bereits existierende

Vorleistungsdatei genutzt. Ist der Student für die Klausur zugelassen, erhält er eine Bestätigung und wird in die Teilnehmerdatei für die Klausur übernommen. Ansonsten erhält er eine Ablehnung mit Begründung.

Nach Ablauf des Anmeldungszeitraums erhalten die jeweiligen Erstprüfer eine Liste mit den für ihre Klausur zugelassenen Studenten.

Der erste (und einfachste) Schritt zur Erstellung der Datenflußdiagramme für diese Anwendung besteht darin, einen Kreis zu malen und den Namen der Anwendung hineinzuschreiben. Dieser Kreis versinnbildlicht unsere Anwendung im Modell (vgl. *Abbildung 16*).


*Abbildung 16 Versinnbildlichung der Anwendung im Modell*

In den Datenflußdiagrammen soll wie bereits gesagt die Umwandlung von Eingabeströmen in Ausgabeströme modelliert werden. Eingaben und Ausgaben stellen wir uns also als Datenströme vor, die in das System hineinfließen bzw. aus dem System herausfließen. Sie werden durch gerichtete Pfeile symbolisiert.

Wir ermitteln nun die Eingaben in das System und die Ausgaben, die das System seinen Nutzern zur Verfügung stellt.

Eingaben sind:

- Klausurdaten.
- Klausuranmeldungen.

Ausgaben des Systems sind:

- Prüfungsplan für Professoren.
- Prüfungsplan für Studenten.
- Anmeldungsbestätigung.
- Anmeldungsablehnung.
- Teilnehmerlisten.

Wir nehmen nun die gefundenen Eingaben und Ausgaben in das System auf. Eingaben werden als Pfeile mit Spitze in Richtung unserer Anwendung, Ausgaben als Pfeile mit Spitze aus unserer Anwendung heraus modelliert. *Abbildung 17* entsteht.

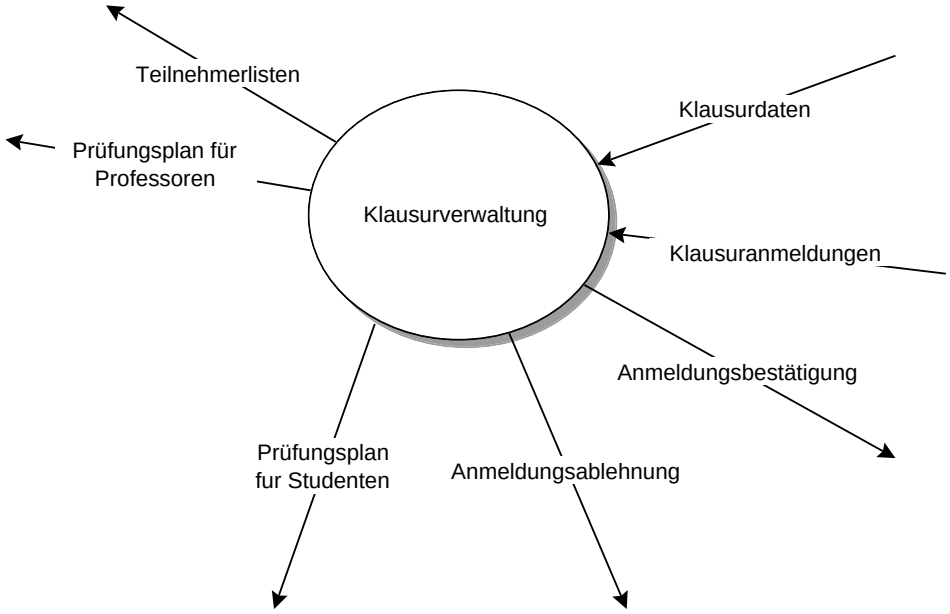


*Abbildung 17 Das DFD-Modell mit Eingabe- und Ausgabedatenströmen*

Als nächstes werden die Datenströme den Personen, die sie erzeugen oder erhalten, zugeordnet. So wird der Datenstrom “Klausurdaten” vom Vorsitzenden des Prüfungsamts erzeugt, den Datenstrom “Prüfungsplan für Professoren” erhalten die Professoren. Hierfür müssen wir die Personen ermitteln, die mit der Anwendung interagieren. Dies sind:

- Professoren.
- Studenten.
- Der Vorsitzende des Prüfungsausschusses.

Im Datenflußdiagramm werden die mit der Anwendung interagierenden Personenkreise durch Rechtecke symbolisiert. Sie heißen Schnittstellen oder Terminatoren. *Abbildung 18* ensteht.

Damit ist unser erstes Datenflußdiagramm erfolgreich erstellt. Dieses Diagramm heißt das Kontextdiagramm der Anwendung. Es zeigt:

- Die Anwendung selbst im Mittelpunkt.
- Alle Datenflüsse in die Anwendung hinein und aus der Anwendung heraus (Eingaben und Ausgaben).
- Die Umgebung der Anwendung. Dies bedeutet, Personenkreise oder andere Anwendungen (dies kam in unserem Beispiel nicht vor), die mit der Anwendung interagieren.

Aus unserem Kontextdiagramm wird allerdings nicht klar, welcher Datenstrom in welchen transformiert wird und wie dies vor sich geht. Dazu ist die “Funktion” in *Abbildung 18* (Klausurverwaltung) viel zu allgemein.

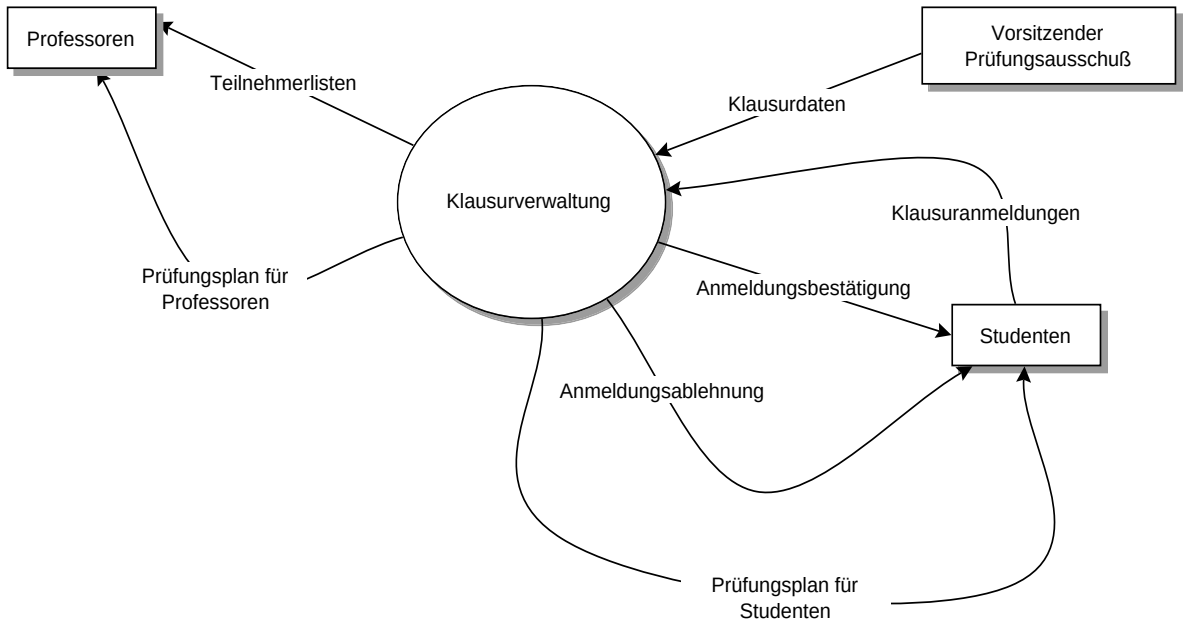


*Abbildung 18 Das DFD-Modell mit Ein-, Ausgabedatenströmen und interagierenden Personenkreisen*
*Das Kontextdiagramm des Beispiels*

Um auch dies modellieren zu können, ermitteln wir nun die Hauptfunktionen der zu realisierenden Anwendung. Eine Möglichkeit, die Funktionalität des zu entwickelnden Systems aufzuteilen, besteht in folgenden Funktionen:

- Klausuren planen.
- Klausuranmeldung überprüfen.
- Listen ausgeben.

Um dies in Datenflußdiagrammen abbilden zu können, zeichen wir Kreise in die wir die Namen der Hauptfunktionen schreiben (vgl. *Abbildung 19*)

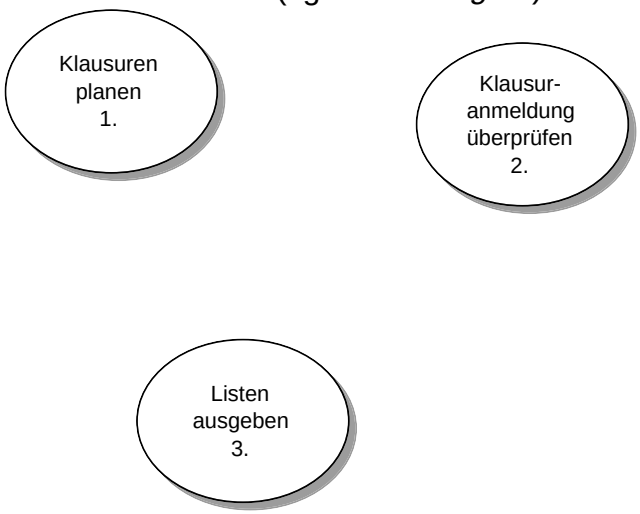


*Abbildung 19 Hauptfunktionen der Anwendung*

Die Hauptfunktionen werden durchnummeriert. Die Eingabeströme und Ausgabeströme unserer Anwendung sind bekannt (vgl. *Abbildung 18*). Wir verteilen die Ein- und Ausgabeströme auf unsere Hauptfunktionen. *Abbildung 20* ensteht.

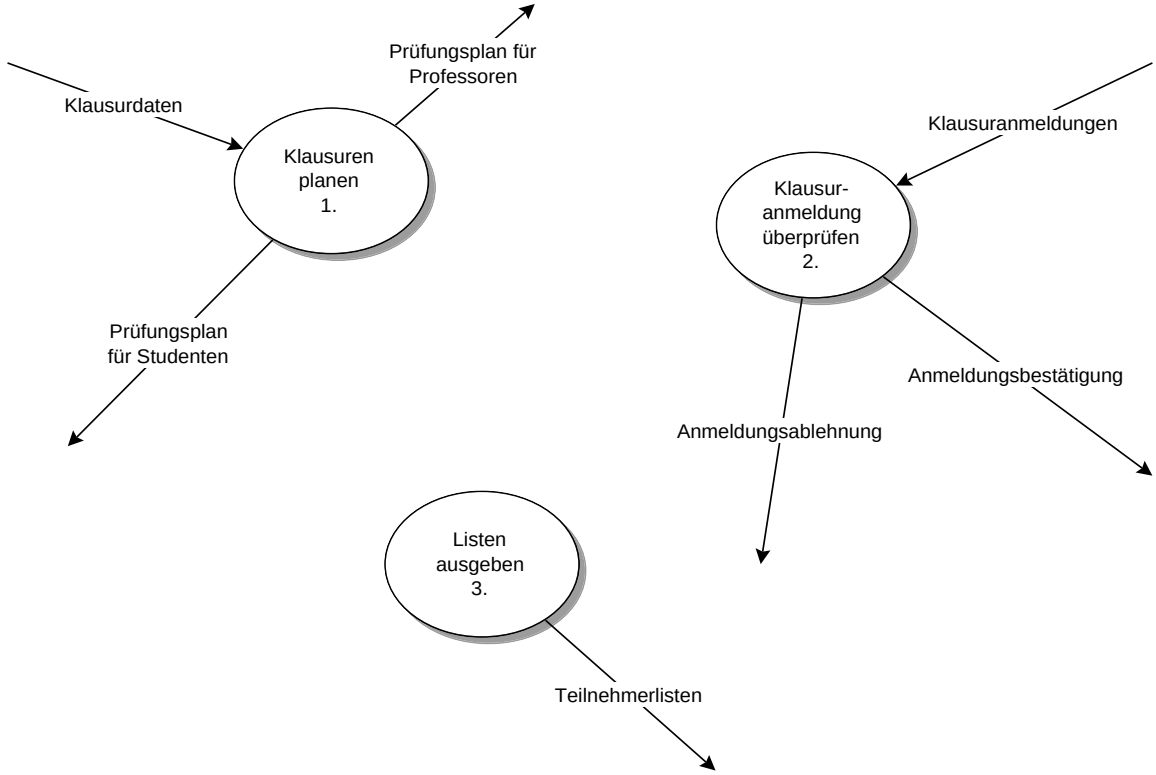


*Abbildung 20* *Die Hauptfunktionen der Anwendung mit den ihnen zugeordneten Datenströmen*

Die Funktionen in *Abbildung 20* sind seltsam unzusammenhängend. Zusammenhänge zwischen den Funktionen sind nicht erkennbar. “Klausuranmeldung überprüfen” benötigt z.B. Ergebnisse von “Klausuren planen” für seine Tätigkeit. Studenten können sich ja nur für eine Klausur anmelden, die auch angeboten wird. Das Klausurangebot wird aber von “Klausuren planen” festgelegt. Desweiteren benötigt “Klausuranmeldung überprüfen” externe Informationen, nämlich eine Datei mit den erbrachten Vorleistungen, um überprüfen zu können, ob der jeweilige Student zur Klausur zugelassen ist. Auch “Listen ausgeben” benötigt Informationen, um die Teilnehmerlisten ausgeben zu können.

In Datenflußdiagrammen werden Datenspeicher benutzt, um solche Zusammenhänge modellieren zu können. Datenspeicher oder Dateien bilden eine Ablagemöglichkeit für Daten, bei denen sich der Entstehungszeitpunkt vom Nutzungszeitpunkt unterscheidet. In unserem Beispiel existieren folgende Datenspeicher:

- Klausurdatei: Wird von “Klausuren planen” erstellt. Enthält alle Daten über die zu schreibenden Klausuren (z.B. Name der Klausur, Klausurnummer, Termin, Raum, etc.).
- Vorleistungsdatei: Enthält die Studenten, die die Vorleistungen für die jeweilige Klausur bereits erbracht haben.

- Teilnehmerdatei: Enthält die Teilnehmer für die jeweiligen Klausuren.

Datenspeicher werden durch zwei parallele Striche dargestellt. Die Aufnahme von Datenspeichern in *Abbildung 20* ergibt *Abbildung 21*.

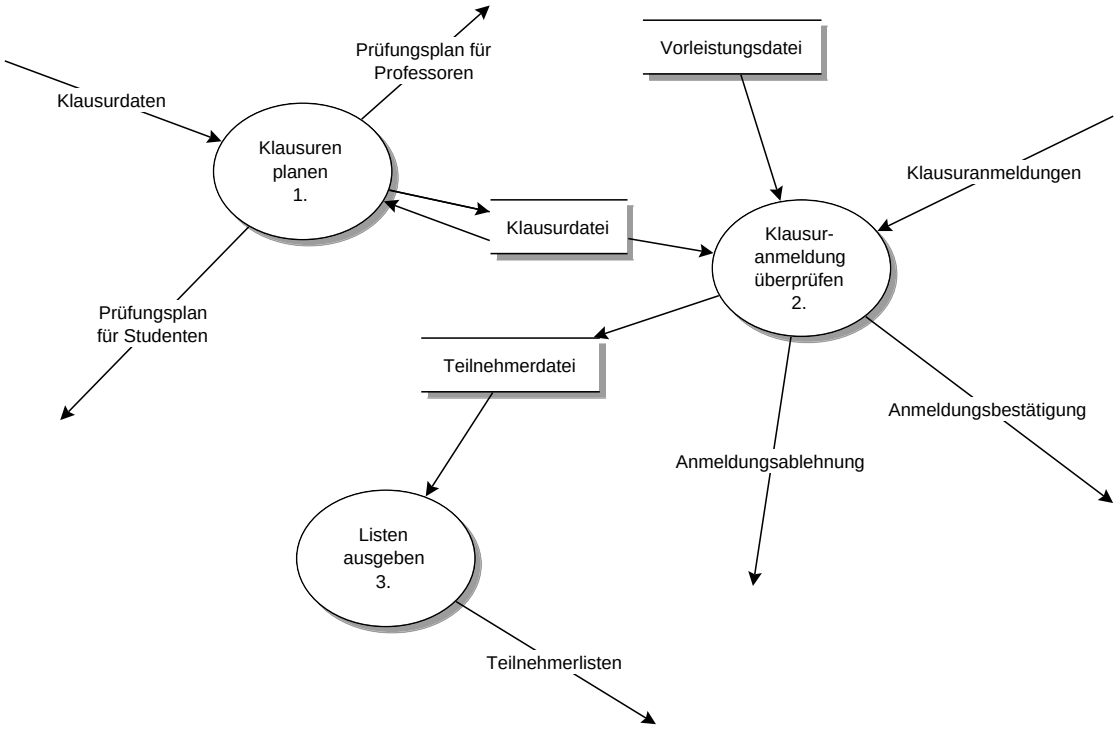


*Abbildung 21* *Die Hauptfunktionen der Anwendung mit den ihnen zugeordneten Datenströmen und den Datenspeichern*
*Das Diagramm 0 der Anwendung*

Schreibt eine Funktion Daten in einen Datenspeicher, wird dies durch einen auf den Datenspeicher gerichteten Datenstrom (Pfeil mit Spitze auf Datenspeicher) symbolisiert. Der Lesevorgang einer Funktion wird durch einen auf die Funktion gerichteten Datenstrom (Pfeil mit Spitze auf Funktion) modelliert. Datenflüsse von und zu Datenspeichern erhalten keine Namen. Sie sind durch den zugehörigen Datenspeicher charackterisiert.

*Abbildung 21* heißt das Diagramm 0 der Datenflußdiagramme zu unserem Anwendungsbeispiel. Das Diagramm 0 zeigt also:

- Die Anwendung aufgegliedert in ihre Hauptfunktionen. Die Hauptfunktionen werden durch Kreise im Diagramm 0 modelliert.
- Die bereits im Kontextdiagramm vorhandenen Datenflüsse. Sie werden auf die Hauptfunktionen verteilt. Hierbei ist zu beachten, daß kein Datenfluß verlorengehen oder hinzukommen darf (Datenflußkonsistenz).
- Die Datenspeicher. Sie werden durch Datenflüsse mit den sie nutzenden Funktionen verbunden.

Die Schnittstellen werden im Diagramm 0 nicht mehr aufgeführt.

Vom Diagramm 0 ausgehend werden die Funktionen oder Prozesse des Diagramm 0 weiter verfeinert. Eine Verfeinerung des Prozesses (Prozeß und Funktion werden im folgenden synonym benutzt) “Klausuren planen” könnte sein:

- Klausuren aufnehmen und
- Prüfungspläne ausgeben.

Der Prozeß “Klausuren planen” hatte im Diagramm 0 die Nummer 1. Seine Verfeinerung im Datenflußdiagramm-Modell heißt dann Diagramm 1. Das Diagramm ensteht völlig analog zum Diagramm 0.

Zunächst werden die Funktionen der Verfeinerung in Kreisen dargestellt (vgl. *Abbildung 22*). Die Funktionen werden strukturiert durchnummeriert. Da “Klausuren planen” im Diagramm 0 die Numer 1 hatte, heißen die im Diagramm 1 dargestellten Funktionen 1.1, 1.2, etc.

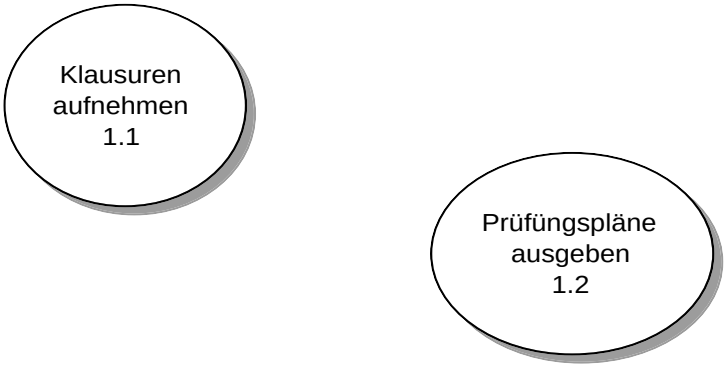


*Abbildung 22 Verfeinerung der Funktion “Klausuren planen”*

Danach werden die Datenflüsse der Funktion “Klausuren planen” auf ihre Teilfunktionen verteilt (vgl. *Abbildung 23*).

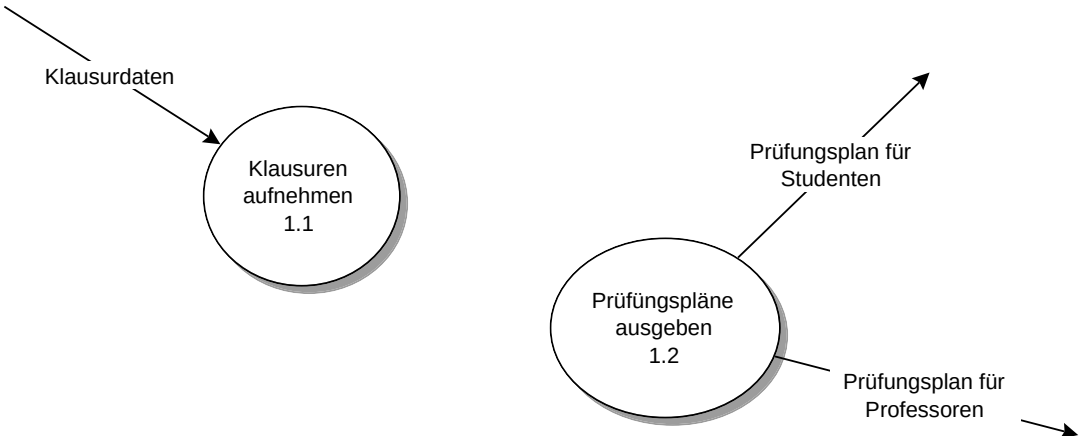


*Abbildung 23 Verfeinerung der Funktion “Klausuren planen” mit zugehörigen Datenflüssen*

Zum Abschluß werden die von “Klausuren planen” genutzten Datenspeicher in die Abbildung übernommen (vgl. *Abbildung 24*)

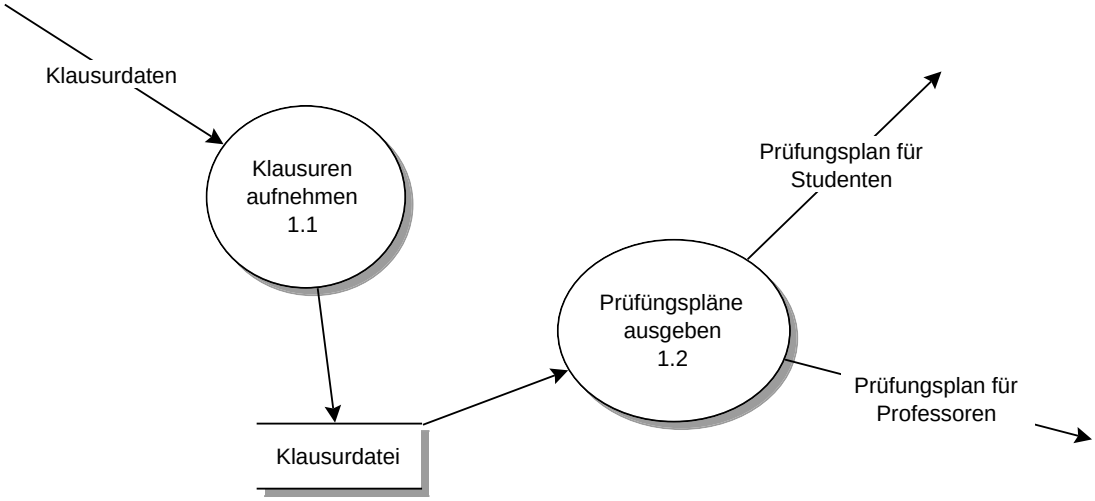


*Abbildung 24* *Verfeinerung der Funktion “Klausuren planen” mit zugehörigen Datenflüssen und Datenspeichern*
*Das Diagramm1 der Anwendung*

Nun verfeinern wir den Prozeß “Klausuranmeldung überprüfen”. Er besteht aus den Teilprozessen:

- Klausur wird angeboten überprüfen und
- Vorleistung ist erbracht überprüfen.

Daraus ergibt sich das Diagramm 2 der Anwendung (vgl. *Abbildung 25*)

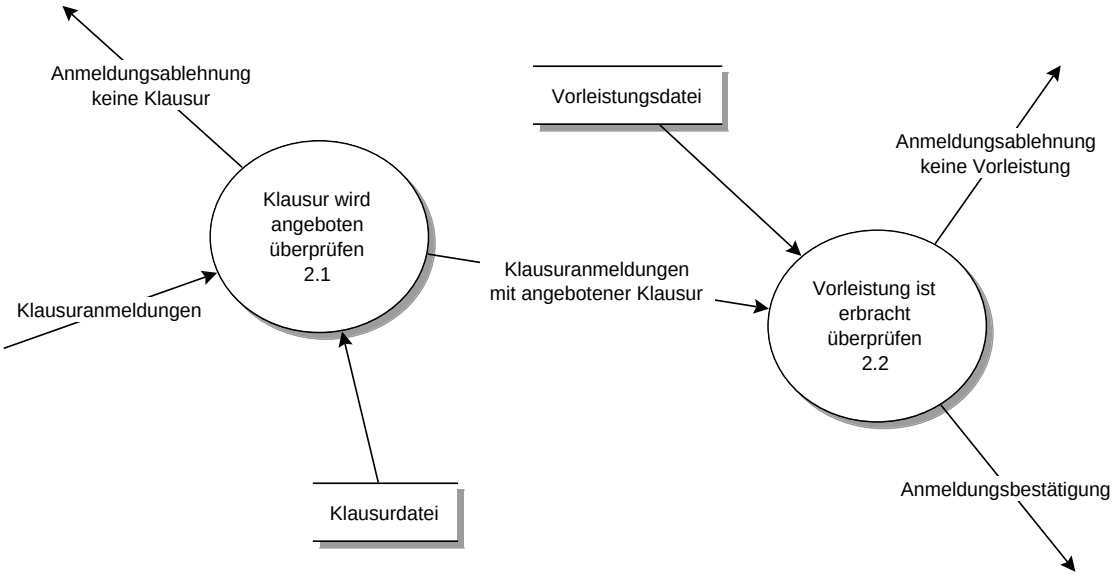


*Abbildung 25* *Das Diagramm 2 der Anwendung*

*Abbildung 25* führt zwei neue Konzepte ein:

- Mit der Funktion “Klausuranmeldung überprüfen” in *Abbildung 21* sind drei Datenströme verbunden (Klausuranmeldungen, Anmeldungsbestätigung, Anmeldungsablehnung). Die Gesamtanzahl der in *Abbildung 25* aus den Teilfunktionen hinaus- und hineinfließenden Datenströme ist aber vier (Klausuranmeldungen, Anmeldungsbestätigung, Anmeldungsablehnung keine Klausur, Anmeldungsablehnung keine Vorleistung). Dies widerspricht (scheinbar) der Datenflußkonsistenz. In Wahrheit wird aber der Datenfluß Anmeldungsablehnung in zwei Datenflüsse (Anmeldungsablehnung keine Klausur, Anmeldungsablehnung keine Vorleistung) aufgesplittet. Dies ist zulässig, muß aber im Diagramm 2 angemerkt werden (vgl. *Abbildung 26*).

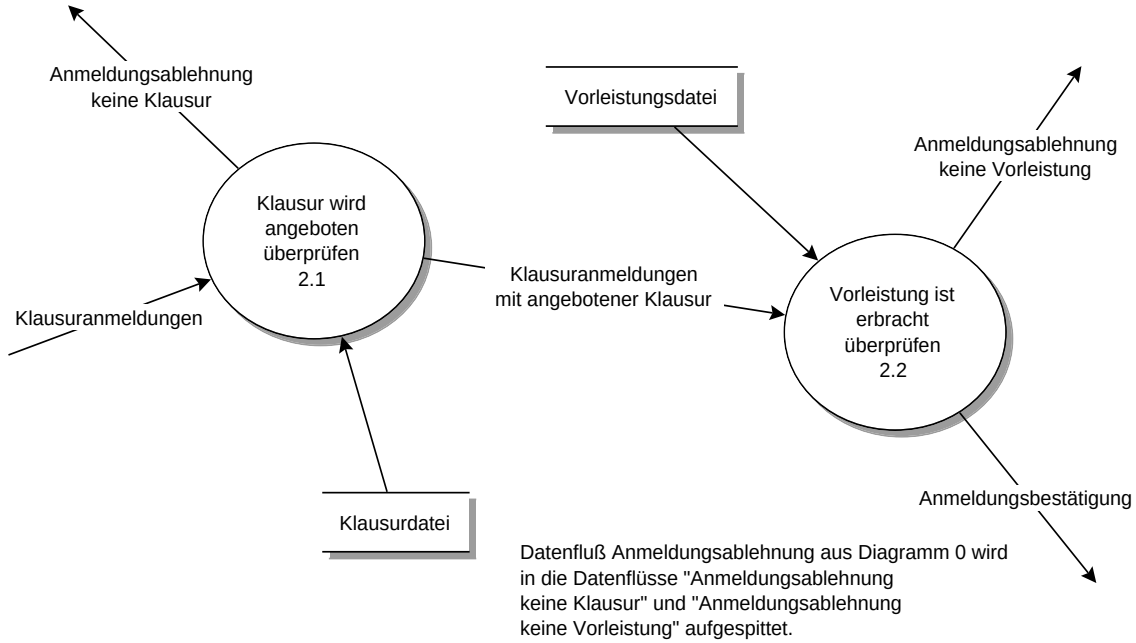


*Abbildung 26 Das Diagramm 2 der Anwendung (verbessert)*

- Ein Datenfluß zwischen zwei Prozessen kommt hinzu (Klausuranmeldung mit angebotener Klausur). Dies scheint ebenfalls der Datenflußkonsistenz zu widersprechen, ist aber wieder nicht der Fall. Der Datenfluß “Klausuranmeldung mit angebotener Klausur” befindet sich gänzlich innerhalb des schraffierten Bereichs in *Abbildung 27*. Dies bedeutet, daß er im nächsthöheren Diagramm (in diesem Fall Diagramm 0) nicht mehr zu sehen sein wird, da der gesamte schraffierte Bereich in *Abbildung 27* zum Prozeß “Klausuranmeldungen überprüfen” aggregiert wird. Datenflüsse zwischen zwei Prozessen einer Verfeinerung können also hinzukommen. Die Datenflußkonsistenz gilt nur für Datenflüsse, die nach außen gehen oder von außen kommen.

Den Prozeß “Listen ausgeben” zu verfeinern, macht wenig Sinn. Er ist bereits so elementar, daß eine Verfeinerung unnötig erscheint. Generell werden Prozesse solange verfeinert, bis sie übersichtlich in den Prozeßspezifikationen beschrieben werden können. Prozeßspezifikationen werden in *Kapitel 3.2.3* behandelt.

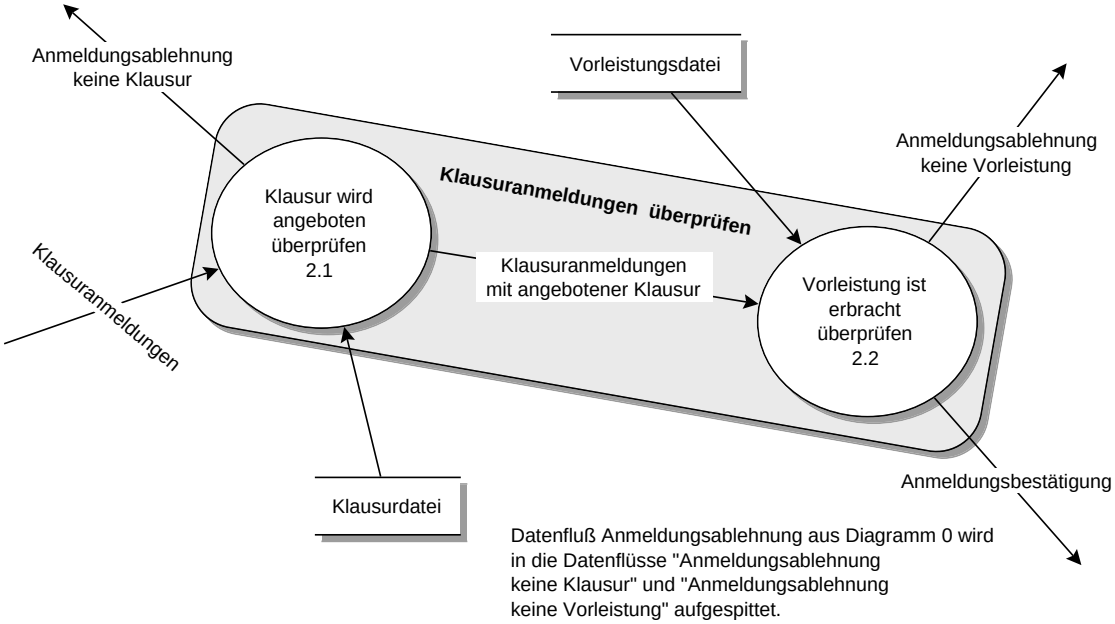


*Abbildung 27 Bei Aggregation verschwindender Datenfluß zwischen zwei Prozessen*

Auch die Prozesse in Diagramm 1 und Diagramm 2 erfordern keine weitere Verfeinerung. Wir erhalten also die Datenflußdiagramm-Struktur von *Abbildung 28*.

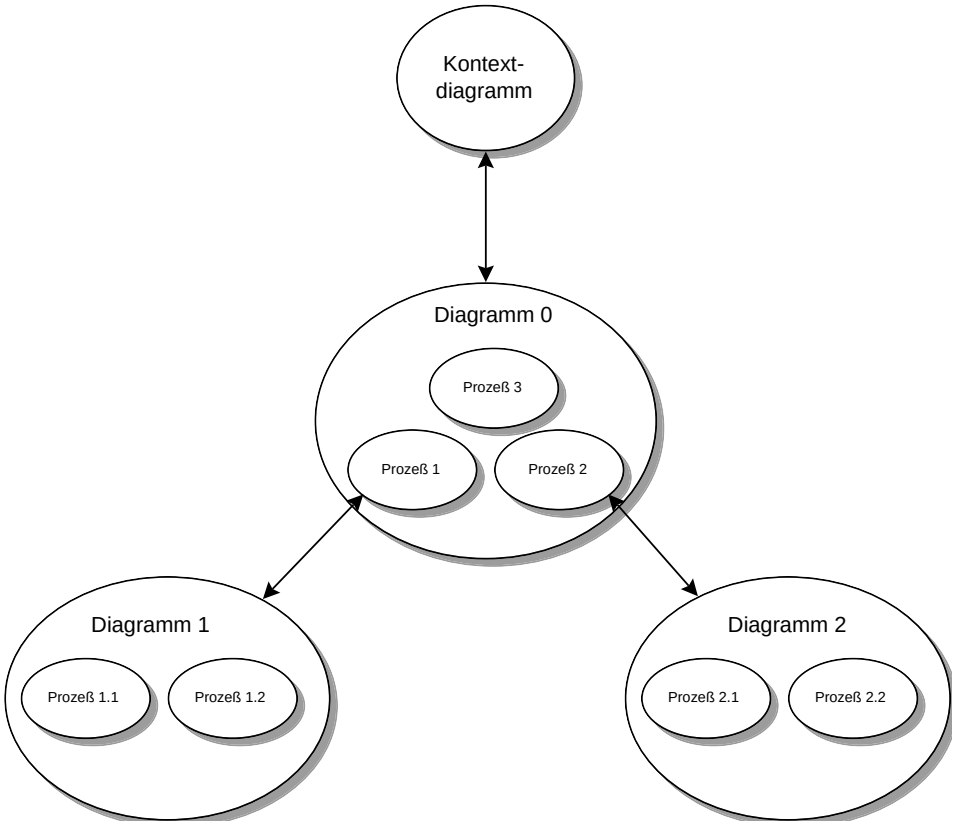


*Abbildung 28 Die Datenflußdiagramm-Struktur des Klausurverwaltungsprogramms*

### 3.2.1.2Zusammenfassung der Begrifflichkeiten

Datenflußdiagramme modellieren die Transformation von Eingabeströmen in Ausgabeströme. Dabei werden die in *Abbildung 29* dargestellten Symbole genutzt.

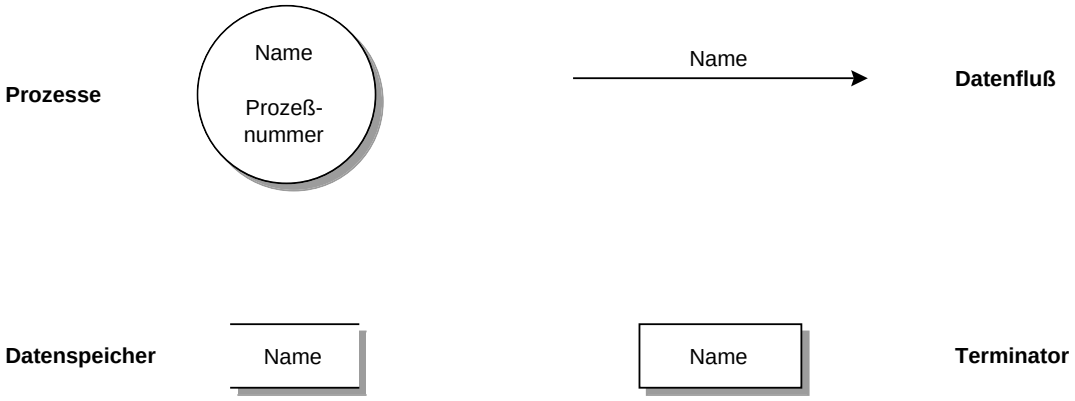


*Abbildung 29 Symbole in Datenflußdiagrammen*

**Prozesse** werden durch Kreise, sogenannte “Bubbles”, dargestellt. Sie enthalten einen Namen und eine hierarchisch gebildete Nummer. Prozesse haben die Aufgabe, Eingabedaten in Ausgabedaten zu verarbeiten und enthalten die dafür notwendigen Algorithmen. Ist der Inhalt eines Prozesses einfach formulierbar, so wird die Beschreibung in einer Prozeßspezifikation (kurz: PSPEC, vgl. *Kapitel 3.2.3*) vorgenommen, in allen anderen Fällen muß der Prozeß in der nächsten Ebene durch ein neues Datenflußdiagramm beschrieben werden.

**Datenflüsse** werden durch einen Vektorpfeil mit einem dazugehörigen Text repräsentiert. Die Richtung eines Vektors gibt an, ob Daten zu Prozessen, Datenspeichern und Terminatoren hin- oder wegfließen.

**Terminatoren** beschreiben die Schnittstellen bzw. die Beziehungen des betrachteten Systems zur Außenwelt. Sie treten nur als Sender bzw. Empfänger von Daten auf, haben also keine eigene Verarbeitungslogik und werden deshalb auch nicht beschrieben. Terminatoren dürfen nur im Kontextdiagramm verwendet werden. Beziehungen zwischen zwei Terminatoren werden nicht dargestellt.

**Datenspeicher oder Dateien** werden durch zwei parallele Striche dargestellt. Dateien bilden eine Ablagemöglichkeit für Daten, bei denen sich der Entstehungszeitpunkt von dem Nutzungszeitpunkt unterscheidet. Die Richtung eines Datenflusses auf einen Datenspeicher gibt an, ob aus der Datei gelesen wird (Pfeil zeigt vom Datenspeicher weg) oder ob in die Datei geschrieben wird (Pfeil zeigt in Richtung des Datenspeichers). Dateien sind im Kontextdiagramm noch nicht sichtbar.

Das **Kontext-Diagramm** bildet die oberste Ebene der Datenflußdiagramm-Hierarchie. Es beschreibt die Schnittstellen des Gesamtsystems zur Außenwelt. In einem Kontext-Diagramm wird das betrachtete System durch einen einzigen Prozeß dargestellt.

In der nächsten Stufe wird nun das zu betrachtende System weiter verfeinert, wobei darauf zu achten ist, daß die Regel des Datenflußgleichgewichts eingehalten wird, d.h. alle Eingabe-/ Ausgabe-Datenflüsse des zu verfeinernden Prozesses müssen auch im

verfeinerten Prozeß vorkommen. Die Verfeinerung wird solange durchgeführt, bis sich die Prozesse nicht weiter verfeinern lassen, d.h. elementar oder primitiv sind. Elementare Prozesse werden dann durch eine Prozeßspezifikation (s. *Kapitel 3.2.3*) beschrieben.

Untergeordnete, also verfeinerte Diagramme, werden als Kinder (engl. child-diagrams), übergeordnete Diagramme als Eltern (engl. parent-diagrams) bezeichnet. Jedes Datenflußdiagramm besitzt eine hierarchisch gebildete Prozeßnummer, die aus Diagrammnummer - Dezimalpunkt - lokale DFD Nummer gebildet wird. Zu Veranschaulichung diene *Abbildung 30*.

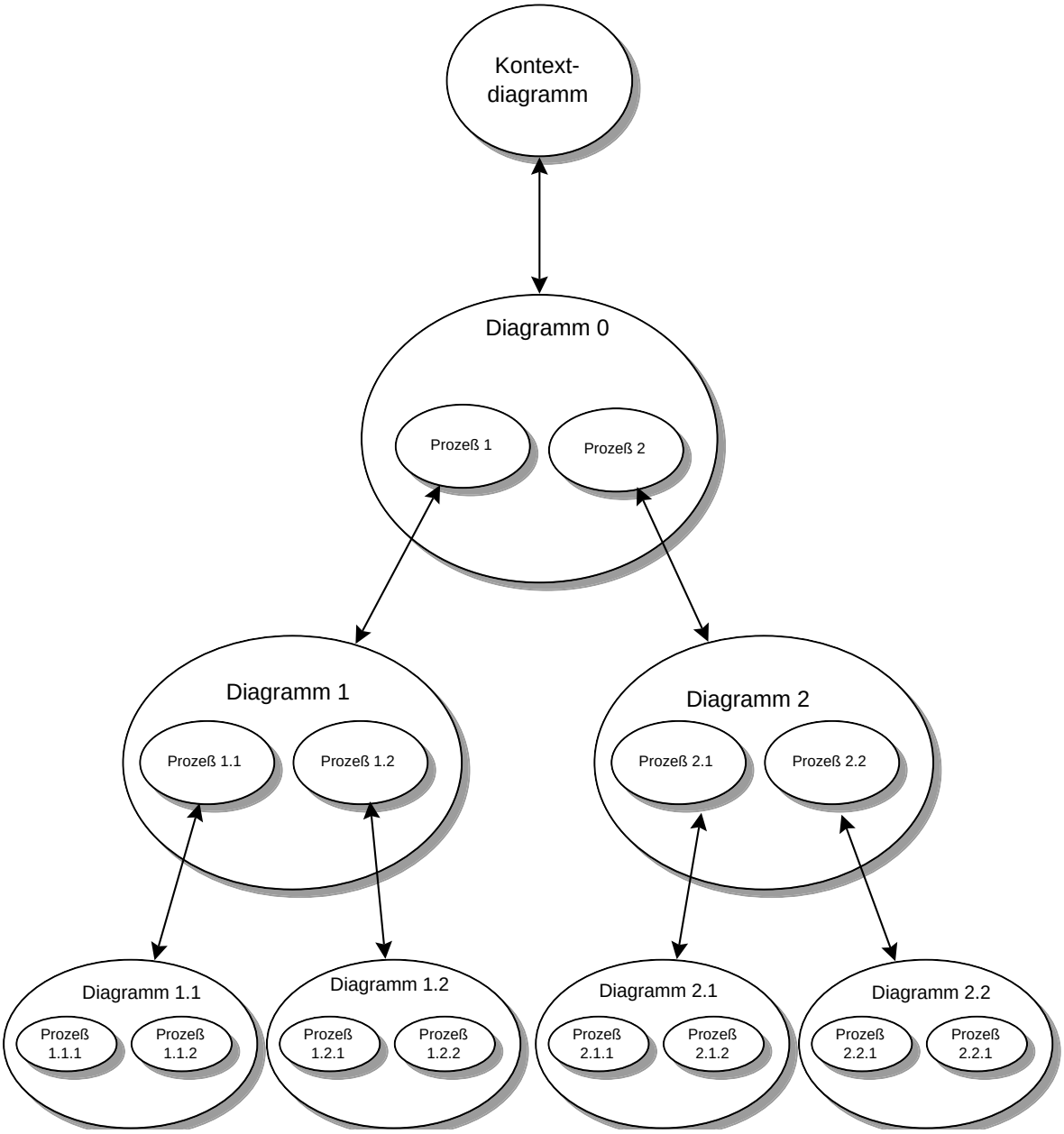


*Abbildung 30 Datenflußdiagramm-Hierarchie*

### 3.2.1.3Generelle Vorgehensweise

Eine generell richtige und immer zutreffende Vorgehensweise für die Erstellung von Datenflußdiagramm-Hierarchien gibt es nicht.

Eine (die im Beispiel angewendete Möglichkeit) ist, mit dem Kontextdiagramm zu beginnen. Man ermittelt bei dieser Vorgehensweise zunächst alle in das System hineinfließenden uns aus dem System herausfließenden Datenströme. Sodann werden die Schnittstellen (Terminatoren) identifiziert und mit den Datenströmen verbunden.

Nach der Erstellung des Kontextdiagramms werden die Hauptfunktionen der Anwendung und damit die Prozesse des Diagramm 0 ermittelt. Dies ist ein kreativer Prozeß und sollte zusammen mit den zukünftigen Anwendern des Systems geschehen. Die Datenflüsse des Kontextdiagramms werden auf die Prozesse des Diagramm 0 aufgeteil. Die Datenspeicher werden gefunden und mit den Prozessen verbunden. Eventuell hinzukommende Datenflüsse zwischen Prozessen werden aufgenommen.

Nun beginnt die Verfeinerung der Hauptfunktionen. Für jede Funktion im Diagramm 0 wird entschieden, ob sie einer Verfeinerung bedarf. Ist dies der Fall, wird die Verfeinerung durchgeführt. Dies bedeutet, die Teilfunktionen des Prozesses werden festgestellt und in das untergeordnete Diagramm aufgenommen. Die Datenflüsse, die im Kontextdiagramm mit dem zu verfeinernden Prozeß verbunden sind, werden auf die Teilprozesse verteilt (Datenflußkonsistenz). Die Datenspeicher, mit denen der zu verfeinernde Prozeß kommuniziert, werden in das Diagramm aufgenommen. Eventuell hinzukommende Datenflüsse zwischen Prozessen werden identifiziert.

In jedem entstandenen Diagramm wird nun überprüft, ob die dort vorhandenen Prozesse weiterer Verfeinerung bedürfen. Ist dies der Fall, wird wie oben beschrieben verfahren. Die Modellbildung endet, wenn keine weitere Verfeinerung erforderlich ist.

Eine weitere Möglichkeit besteht darin, mit dem Diagramm 0 zu beginenn. Man versucht zunächst, die Hauptfunktionalitäten der Anwendung festzustellen. Dann werden die Datenströme zu und von den Hauptprozessen ermittelt. Das Diagramm 0 wird fertiggestellt. Danach kann man aus dem Diagramm 0 das Kontextdiagramm durch Aggregation erstellen. Hier müssen eigentlich nur noch die Terminatoren ermittelt werden.

Dioe Entwicklung der nachgeordneten Diagramme erfolgt wie im ersten Fall beschrieben.

### 3.2.1.4Semantische und syntaktische Regeln

Syntaktische Regeln:

- Ein Datenflußdiagramm sollte aus Übersichtlichkeitsgründen nicht mehr als neun Prozesse enthalten. Ist dies dennoch der Fall, sollte eine Zwischenebene eingezogen werden.
- Jeder Datenfluß muß mit mindestens einem Prozeß verbunden sein.
- Direkte Datenflüsse zwischen Terminatoren und Speichern sind verboten. Sie können auch nicht auftreten, da im Kontexdiagramm keine Datenspeicher vor-

handen sind. Terminatoren hingegen werden bei der klassischen Strukturierten Analyse nur im Kontexdiagramm aufgeführt.

- Jeder Prozeß und jeder Datenfluß, der nicht mit einem Datenspeicher verbunden ist, muß einen Namen haben
- Datenflüsse zwischen Speichern und Prozessen haben keinen Namen.
- Datenflüsse zwischen Terminatoren kann es nicht geben.
- Prozesse ohne Ausgänge kann es nicht geben. Prozesse ohne Eingänge sind ebenfalls zumeist falsch (eine Ausnahme ist z.B. ein Zufallszahlengenerator).

*Abbildung 31* stellt einige Fehlermöglichkeiten grafisch dar.

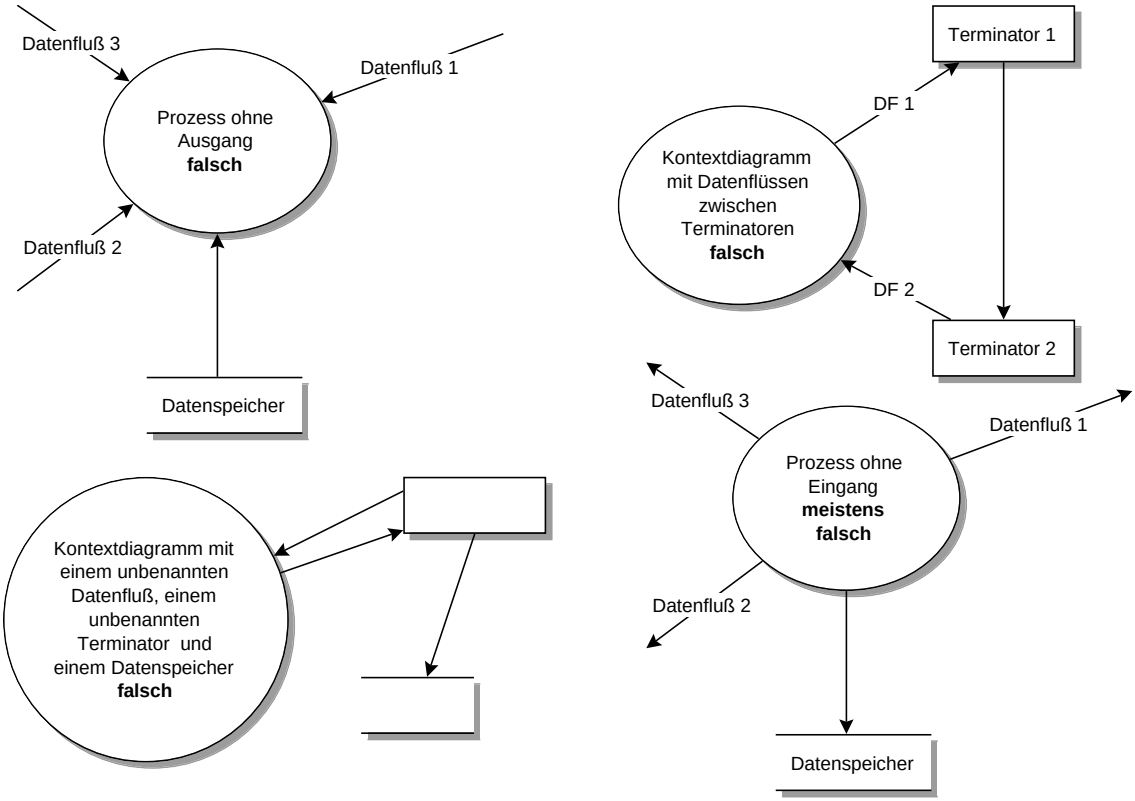


*Abbildung 31 Syntaktische Fehler in Datenflußdiagrammen*

Semantische Regeln:

In Datenflußdiagrammen sollten Prozesse, Datenspeicher und Datenflüsse mit einem möglichst kurzen, aber dennoch aussagekräftigen Namen versehen werden. Vermeiden sollte man sowohl technische Kürzel als auch Namen, die nicht klar und einwandfrei aussagen, welche Funktion das Objekt hat.

- Prozeßnamen sollten grundsätzlich aus einem starken Verb, das die Aktion exakt beschreibt und einem Substantiv gebildet werden (vgl. *Abbildung 32*).
- Datenflüsse sollten mit einem Substantiv bezeichnet werden, das inhaltlich keine Verarbeitung beschreiben darf.

| | Verben | Substantive |
|---|---|---|
| ok | prüfen, buchen<br>berechnen, eintragen<br>ermitteln, entfernen<br>suchen, erstellen | Buchungsnummer<br>Schnittpunkt<br>Strafmandat<br>Rechnung |
| Vorsicht! | erzeugen, verteilen<br>aufbereiten, drucken<br>ändern | Datensatz<br>Datei |
| vermeiden! | verarbeiten | Daten Informationen |

*Abbildung 32 Benennungen in Datenflußdiagrammen*

- Der Name eines Speichers sollte aus einem Substantiv bestehen, das auf den Inhalt des Speichers hinweist.

### 3.2.2 Datenkataloge

Für jeden Datenfluß und jede Datei erfolgt im Datenkatalog (engl. Data Dictionary) eine Beschreibung der Zusammensetzung des Datenflusses in einer speziellen Notation. Dabei wird nur auf inhaltliche Bedeutung der Datenelemente eingegangen, physikalische Datenformate werden nicht beschrieben. Mehrfach vorkommende Datenelemente werden über Hilfsdefinitionen beschrieben und können dann wie Konstanten weiterverwendet werden.

Die Notation für Datenkataloge ist wie in *Abbildung 33* gezeigt festgelegt.

| Name | Symbol | Bedeutung |
|---|---|---|
| Zusammensetzung | = | ist zusammengesetzt aus, ist äquivalent zu |
| Verkettung | + | und (Aufzählung) |
| Selektion | [..\|..] | entweder oder |
| Iteration | {...}<br>2 {...}<br>{...} 5<br>1 {...} 6 | mehrfaches Auftreten<br>von mindestens 2 mal<br>höchstens 5 mal<br>1 bis 6 mal |
| Option | ( ) | kann vorhanden sein |
| Diskreter Wert | "..." | Wert der Variablen |
| Kommentar | * .....* | zusätzliche Information |

*Abbildung 33 Notation im Datenkatalog*

Zunächst einige Beispiele für Einträge im Datenkatalog und ihre Bedeutung:

| | |
|---|---|
| Name | = Nachname + Vorname + (Vorname) |
| Adresse | = PLZ + Wohnort + [ Straße + Hausnummer \| Postfach ] |
| Kundensatz | = Kundennummer + Name + Adresse |
| Flugauskunft | = [ “Flug nicht verfügbar” \| “kein Platz frei” \| Preis + Startzeit] |
| Kundenanfrage | = Kundendaten + 1{Artikelnummer}10 |

Dies bedeutet:

**Name** setzt sich aus Nachname und Vorname und einem optionalen zweiten Vornamen zusammen.

**Adresse** setzt sich aus der PLZ, dem Wohnort und der Straße mit Hausnummer oder einem Postfach zusammen.

Ein **Kundensatz** besteht aus der Kundennummer und den Hilfsdefinitionen Name und Adresse.

Eine **Flugauskunft** kann entweder aus dem Text “Flug nicht verfügbar” oder “kein Platz frei” oder aus einem Preis mit Startzeit bestehen.

Eine **Kundenanfrage** besteht aus den Kundendaten mit mindestens einer und maximal zehn Artikelnummern.

Wir werden die Strukturierte Analyse unseres Klausurverwaltungsprogramms nun weiter vervollständigen, indem wir den Datenkatalog erstellen

| Name | Beschreibung |
|---|---|
| Klausurdaten | Klausurkürzel + Klausurdatum + Klausurstartzeit + Klausurdauer + Name Erstprüfer + Name Zweitprüfer + 2 {Name Aufsicht} + Raum |
| Klausuranmeldungen | Klausurkürzel + Matrikelnummer + Name |
| Prüfungsplan für Studenten | {Klausurkürzel + Klausurdatum + Klausurstart-zeit + Klausurdauer} + Raum |
| Prüfungsplan für Professoren | {Klausurdaten} |
| Anmeldungsbestätigung | “Sie sind für die Klausur zugelassen” |
| Anmeldungsablehnung | [“Die Klausur wird nicht angeboten” \| “Sie haben die Zulassung nicht erbracht”] |
| Teilnehmerlisten | Klausurkürzel + {Matrikelnummer + Name} |
| Klausuranmeldung mit ange-botener Klausur | Klausuranmeldungen |
| Anmeldungsablehnung keine Klausur | “Die Klausur wird nicht angeboten” |
| Anmeldungsablehnung keine Vorleistung | “Sie haben die Zulassung nicht erbracht” |
| Klausurdatei | {Klausurdaten} |
| Vorleistungsdatei | Klausurkürzel + Matrikelnummer |
| Teilnehmerdatei | {Klausurkürzel + {Matrikelnummer + Name}} |

*Abbildung 34 Der Datenkatalog zum Klausurverwaltungssystem*

### 3.2.3 Prozeßspezifikationen

Prozesse, die nicht weiter verfeinert werden sollen (elementare Prozesse engl. functional primitives), werden durch Prozeßspezifikationen beschrieben. Dazu kann jedes Mittel angewendet werden, das den Prozeß eindeutig beschreibt. Die Beschreibung sollte so gewählt werden, daß sie auch vom Anwender verstanden werden kann. Deshalb erfolgt die Darstellung häufig mit Hilfe der Strukturierten Sprache (ein anderes Wort für Pseudocode).

Die Erstellung von Pseudocode haben Sie bereits in den ersten Semestern Ihres Studiums (Grundlagen der Informatik) erlernt. Ich werde mich hier also kurz fassen.

Elemente der Strukturierten Sprache (des Pseudocodes) sind:

- einfache Sätze.
- Verben in Imperativform.
- reservierte Wörter (if, while, else, etc).
- Die Einträge im Datenkatalog.

Wir beenden nun die Strukturierte Analyse unseres Klausurverwaltungsprogramms, indem wir den Pseudocode für die unterste Ebene der Datenflußdiagramm-Hierarchie schreiben. Wir beginnen mit Diagramm 1 (vgl. *Abbildung 24*).

Prozeßspezifikation Klausuren aufnehmen

while Klausurdaten eingegeben werden

begin

    schreibe Klausurdaten in Klausurdatei

end.

*Prozeßspezifikation Prüfungspläne ausgeben*

while Ende von Klausurdatei nicht erreicht

begin

    lies Klausurdaten ein

end

erstelle Prüfungsplan für Studenten

stelle  Prüfungsplan für Studenten Studenten zu

erstelle  Prüfungsplan für Professoren

stelle  Prüfungsplan für Professoren Professoren zu

Die Prozeßspezifikationen zu Diagramm 2:

Prozeßspezifikation Klausur wird angeboten überprüfen

für jede Klausuranmeldung

begin

    if gewünschte Klausur in Klausurdatei

    then begin

        übermittle Klausuranmeldung an Vorleistung ist erbracht überprüfen

    end

    else begin

```
            erzeuge Anmeldungsablehnung keine Klausur
        end
end
```

Prozeßspezifikation Vorleistung ist erbracht überprüfen

```
Für alle von Klausur wird angeboten überprüfen übergebenenen Klausurdaten
begin
        if Vorleistung zur gewünschten Klausur in Vorleistungsdatei
        then begin
            erzeuge Anmeldungsbestätigung
        end
        else begin
            erzeuge Anmeldungsablehnung keine Vorleistung
        end
end
```

Es bleibt noch die Prozeßspezifikation des von uns nicht weiter verfeinerten Diagramm 0-Prozesses Listen ausgeben.

*Prozeßspezifikation* Listen ausgeben

```
Für jede Klausur
begin
        while Ende von Teilnehmerdatei nicht erreicht
        begin
            lies zur Klausur gehörige Teilnehmersätze ein
        end
        erzeuge Teilnehmerliste
        stelle Teilnehmerliste Erstprüfer zu
end
```

Grundsätzlich sollen die Prozesse in den Datenflußdiagrammen solange verfeinert werden, bis die zugehörige Prozess-Spezifikation auf einer DIN A4-Seite dargestellt werden kann. Dies zeigt, daß wir in unserem Klausurverwaltungsprogramm die Verfeinerungen Diagramm 1 und Diagramm 2 nicht benötigt hätten. Wir hätten schon für die Prozesse des Diagramm 0 Prozess-Spezifikationen schreiben können, wie der Pseudocode zum Prozess “Klausuranmeldung überprüfen” beispielhaft zeigt:

```
Prozeßspezifikation Klausuranmeldung überprüfen
für jede Klausuranmeldung
begin
    if gewünschte Klausur in Klausurdatei
    then begin
        if Vorleistung zur gewünschten Klausur in Vorleistungsdatei
        then begin
            erzeuge Anmeldungsbestätigung
        end
        else begin
            erzeuge Anmeldungsablehnung keine Vorleistung
        end
    end
    else begin
        erzeuge Anmeldungsablehnung keine Klausur
    end
end
```

Die Prozeßspezifikationen beschreiben grundsätzlich wie die Eingabedaten der Prozesse der untersten DFD-Ebene in die Ausgabedaten transformiert werden. Dies wird deutlich, wenn wir uns die beispielhaften Prozess-Spezifikationen dieses Kapitels betrachten. Eingabedatenströme und Ausgabedatenströme sind jeweils Bestandteil der Prozess-Spezifikation.

Neben Pseudocode sind Entscheidungstabellen ein gutes Werkzeug zur Erstellung von Prozess-Spezifikationen.

### 3.2.4 Selbsttestaufgabe

Das alteingessene stahlverarbeitende Unternehmen Rost-Stahl & Co KG entschließt sich, im Rahmen von Rationalisierungsmaßnahmen ein Auftragsabwicklungsprogramm realisieren zu lassen.

Dieses neue Softwaresystem soll die Angebotsbearbeitung, die Auftragsbearbeitung, die Versanddisposition und die Fakturierung umfassen.

Die Angebotserfassung soll auf Kundenanfragen reagieren. Bei Kundenanfragen soll das System Angebote mit Artikelnummer, Artikelname, kurzer Artikelbeschreibung, Nettopreis und Bruttopreis erzeugen. Werden mehrere Artikel angefragt, soll zusätzlich der Gesamtpreis (Brutto und Netto) ausgegeben werden. Angebote erhalten eine

Gültigkeitsdauer. Angebote deren Gültigkeitsdauer abgelaufen ist, werden aus der Angebotsdatei gelöscht.

Bei der Auftragsbearbeitung müssen Aufträge und Stornierungen entgegengenommen werden. Auftragsbestätigungen müssen erstellt werden, Aufträge aber auch abgelehnt wrden können. Rost-Stahl & Co KG nimmt nämlich nur Aufträge an, zu denen ihre Angebotsabteilung ein Angebot erstellt hat. Aufgrund schlechter Erfahrungen nimmt Rost-Stahl & Co KG auch nur noch Aufträge von Kunden an, deren Zahlungsverhalten bis jetzt einwandfrei war. Dies kann einem Kennzeichen in der Kundendatei entnommen werden.

Angebote, für die ein Auftrag vorliegt, werden gelöscht. Auftrags- und Stornierungsdaten müssen an die Produktionsplanung übermittelt werden. Selbstverständlich können Stornierungen nur vorgenommen werden, wenn ein nicht abgewickelter Auftrag existiert.

Die Versanddisposition soll die Auftragsabwicklung bei Rost-Stahl & Co KG automatisieren. Einmal täglich werden noch nicht ausgeführten Aufträge auf Ausführbarkeit überprüft. Ist ein Auftrag ausführbar, werden Lieferschein, Adreßaufkleber und Versandpapiere erstellt. Diese Dokumente gehen an das Lager.

Die Fakturierung wird ebenfalls automatisiert. Einmal täglich werden alle abrechenbaren Aufträge ermittelt. Für diese Aufträge werden Rechnungen erstellt. Der Rechnungsausgang wird an die Buchhaltung zur weiteren Bearbeitung übermittelt.

Erstellen Sie eine Strukturierte Analyse für diese Anwendung.

### 3.2.5 Lösung der Selbsttestaufgabe

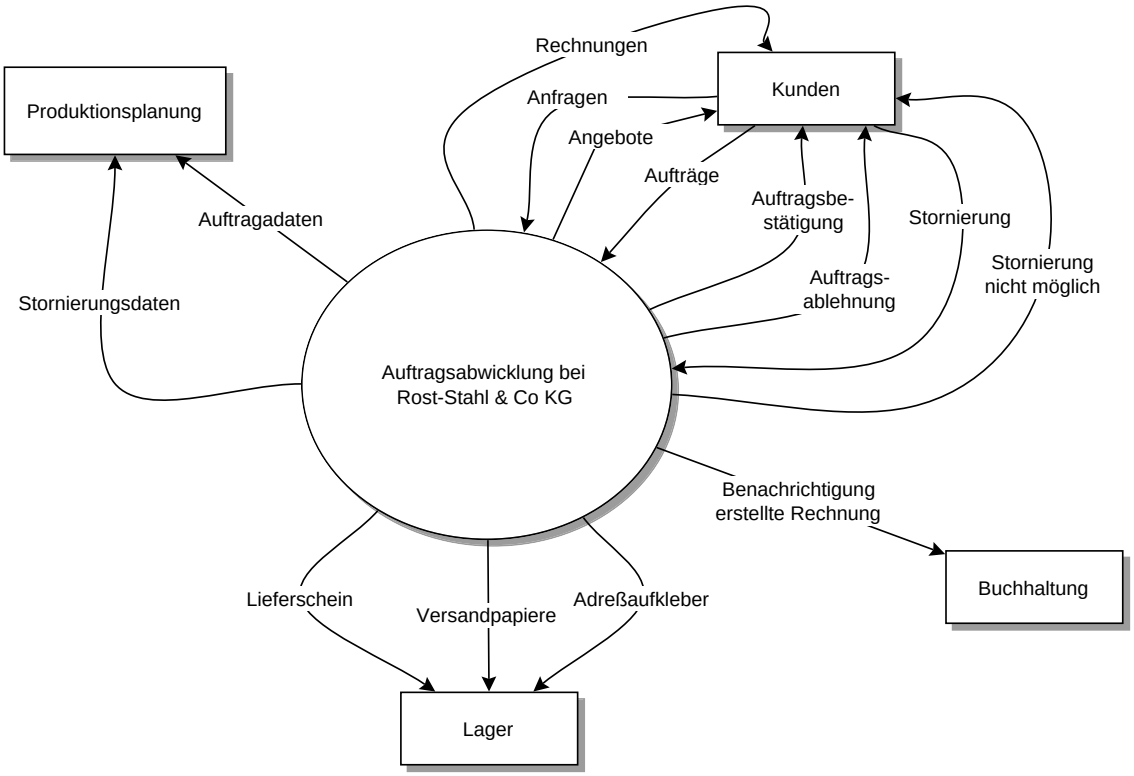


*Abbildung 35 Kontextdiagramm der Selbsttestaufgabe*

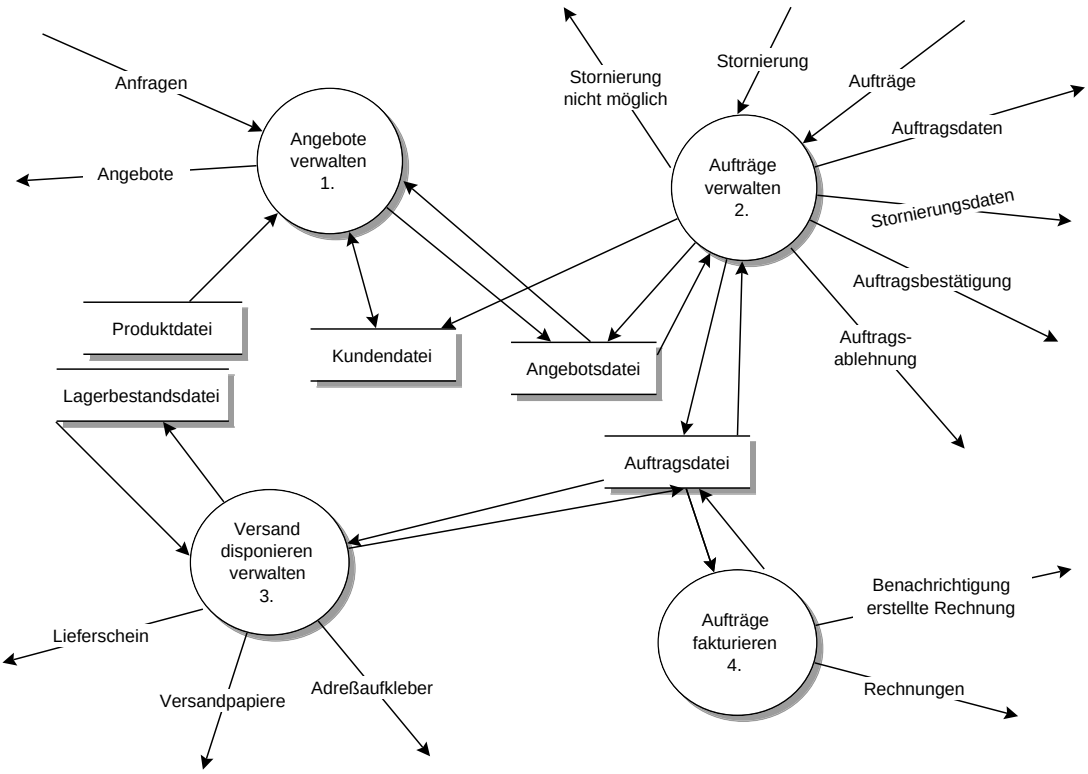


*Abbildung 36 Diagramm 0 der Selbsttestaufgabe*

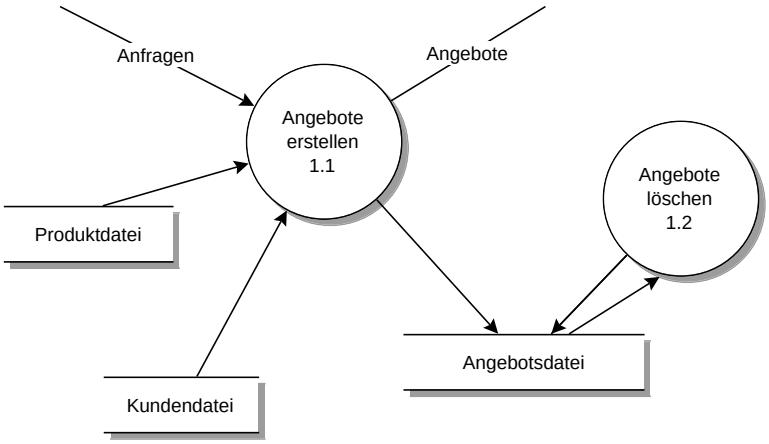


*Abbildung 37 Diagramm 1 der Selbsttestaufgabe*

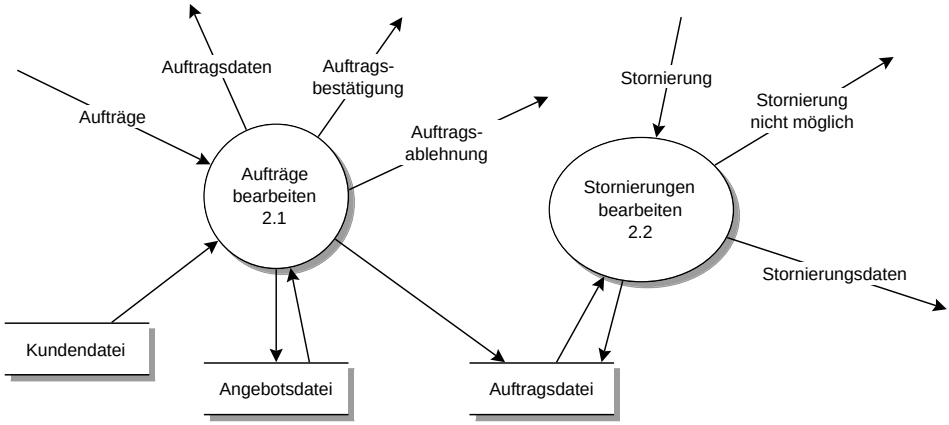


*Abbildung 38 Diagramm 2 der Selbsttestaufgabe*

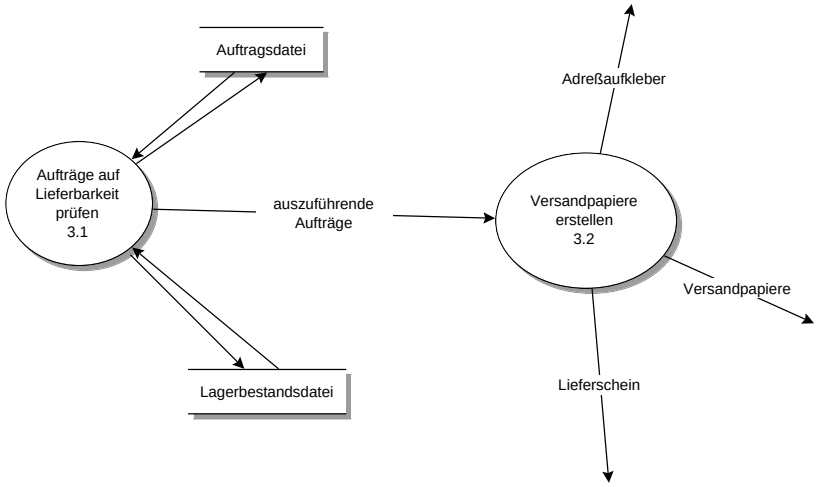


*Abbildung 39 Diagramm 3 der Selbsttestaufgabe*

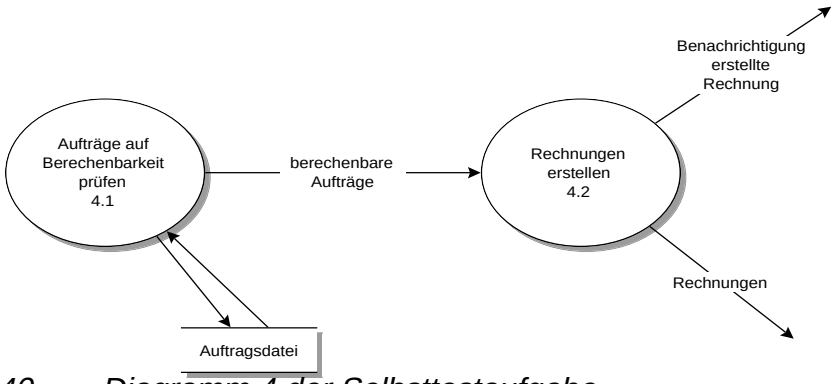


*Abbildung 40 Diagramm 4 der Selbsttestaufgabe*

| Name | Beschreibung |
|---|---|
| Hilfsdefinitionen | |
| Name | Nachname + Vorname + (Vorname) |
| Adresse | PLZ + Stadt + [Straße + Hausnummer \| Postfach] |
| Artikelkurzdaten | Artikelnummer + Artikelname + Artikelkurzbeschreibung |
| Kundendaten | Kundennummer + Name + Adresse |
| Kundensatz | Kundendaten + Tel.-Nr. + Fax-Nr + (email) + Bonitätsschlüssel |
| Bankverbindung | BLZ + Name des Kreditinstituts + Kontonummer |
| Datenkatalog | |
| Anfragen | Kundendaten + 1 {Artikelkurzdaten} |
| Angebote | Angebotsnummer + Kundendaten + 1{Artikelkurzdaten + Artikel-preis} + Gesamtprei-Netto + Gesamtpreis-Brutto |
| Aufträge | Bestellnummer des Kunden + Angebote |
| Auftragsbestätigung | Auftragsnummer + Aufträge |
| Auftragsablehnung | Aufträge + Ablehnungstext |
| Stornierungen | Bestellnummer des Kunden + Kundendaten + 1{Artikelkurzdaten} |
| Rechnungen | Rechnungsnummer + Bankverbindung + Auftragsbestätigung |
| Benachrichtigung erstellte Rechnung | Rechnungsnummer + Bankverbindung + Kundennummer + Rech-nungsbetrag + Rechnungsausgangsdatum |
| Auftragsdaten | Artikelnummer + Anzahl |
| Stornierungsdaten | Artikelnummer + Anzahl |
| Stornierung nicht möglich | Stornierungen + “Stornierung nicht möglich” |
| Lieferschein | Kundendaten + 1{Artikelkurzdaten} |
| Adreßaufkleber | Kundendaten |
| Versandanweisung | 1{Artikelnummer + Anzahl} |
| Kundendatei | {Kundensatz} |
| Produktdatei | {Artikelkurzdaten + Artikelpreis} |
| Angebotsdatei | {Angebote} |
| Auftragsdatei | {Auftragsbestätigung + Kennzeichen ausgeführt} |
| Lagerbestandsdatei | {Artikelnummer + Anzahl vorhanden} |

*Abbildung 41 Der Datenkatalog der Selbsttestaufgabe*

Prozeßspezifikationen

Prozeßspezifikationen Diagramm 1

Prozeßspezifikation Angebote erstellen

Für alle eingangenen Kundenanfragen

begin

lies Kundendaten aus Kundendatei

lies Produktdaten aus Produktdatei

berechne Gesamtnettopreis

berechne Gesamtbruttopreis

stelle Angebot Kunden zu

speichere Angebot in Angebotsdatei

end

end

*Prozeßspezifikation Angebote löschen*

while Ende von Angebotsdatei nicht erreicht

begin

lese Satz aus Angebotsdatei

if Gültigkeitsdauer überschritten

then begin

lösche Angebot

end

end

Prozeßspezifikationen Diagramm 2

Prozeßspezifikation Aufträge bearbeiten

für alle eingegangenen Aufträge

begin

if Angebot in Angebotsdatei und Bonitätsschlüssel Kunde (aus Kundendatei) ok

then begin

erzeuge Auftragsdaten

übergebe Auftragsdaten an Produktionsplanung

```
            erzeuge Auftragsbestätigung
            stelle Auftragsbestätigung Kunden zu
            lösche alle Produkte, für die ein Auftrag vorliegt aus Angebot
            if keine Produkte in Angebot
            then begin
                    lösche Angebot
            end
        else begin
            erzeuge Auftragsablehnung
            stelle Auftragsablehnung Kunden zu
        end
end
Prozeßspezifikation Stornierungen bearbeiten
für alle eingegangenen Stornierungen
begin
        if zu stornierender Auftrag in Auftragsdatei und Auftrag noch nicht abgewickelt
        then begin
            storniere alle zu stornierenden Produkte aus Auftrag
            erzeuge Stornierungsdaten
            übermittle Stornierungsdaten an Produktionsplanung
            if keine Produkte mehr im Auftrag
            then begin
                    lösche Auftrag aus Auftragsdatei
            end
        end
        else begin
            erzeuge Stornierungsablehnung
            stelle Stornierungsablehnung Kunden zu
        end
end
```

Prozeßspezifikationen Diagramm 3

Prozeßspezifikation Aufträge auf Lieferbarkeit prüfen

while Ende von Auftragsdatei nicht erreicht

begin

    if Auftrag nicht geliefert und alle Produkte auf Lager (aus Lagerbestandsdatei)

    then begin

        übergebe Auftragsdaten an Versandpapiere erstellen

        aktualisiere Lagerbestandsdatei

        aktualisiere Auftragsdatei (Auftrag als geliefert kennzeichnen)

    end

end

Prozeßspezifikation Versandpapiere erstellen

für alle auszuführenden Aufträge

begin

    erstelle Lieferschein

    erstelle Adreßaufkleber

    erstelle Versandanweisung

end

Prozeßspezifikationen Diagramm 4

Prozeßspezifikation Aufträge auf Berechenbarkeit prüfen

while Ende von Auftragsdatei nicht erreicht

begin

    if Auftrag geliefert

    then begin

        übergebe Auftragsdaten an Rechnung erstellen

        lösche Auftrag aus Auftragsdatei

    end

end

Prozeßspezifikation Rechnung erstellen

für alle zu berechnenden Aufträge

begin

erstelle Rechnung

stelle Rechnung Kunden zu

erstelle Benachrichtigung erstellte Rechnung

übermittle Benachrichtigung erstellte Rechnung an Buchhaltung

end

## 3.3 Moderne Struktrierte Analyse

Die Moderne Strukturierte Analyse ist eine Weiterentwicklung der Strukturierten Analyse. Sie fügt den bei der Strukturierten Analyse eingesetzten Techniken zwei weitere hinzu:

- Zustandsübergangsdiagramme (Ereignismodellierung)
- Entity Relationship Modellierung

### 3.3.1 Zustandsübergangsdiagramme (Ereignismodellierung)

Das zeitabhängige Verhalten eines Systems läßt sich mit den bisher dargestellten Verfahren nicht modellieren. In unserem Beispiel aus *Kapitel 3.2.1.1* (Klausurverwaltung) machen Klausuranmeldungen der Studenten erst dann Sinn, wenn die Klausurplanung abgeschlossen ist. Solange die Klausurplanung läuft, dürfen die Klausuranmeldungsfunktionen nicht genutzt werden. Dieser Tatbestand ist aber weder den Datenflußdiagrammen noch dem Datenkatalog oder den Prozeßspezifikationen zu entnehmen.

Zeitabhängiges Verhalten wird in der Modernen Strukturierten Analyse durch Zustandsübergangsdiagramme (Ereignismodelle) modelliert. Ein System (unsere zu konzipierende Anwendung) befindet sich zu jedem Zeitpunkt in genau einem Zustand. So befindet sich unser Klausurverwaltungs-System während einer bestimmten Zeit im Zustand Klausurplanung.

Systeme können von einem Zustand in einen anderen übergehen (die Klausurverwaltung z.B. vom Zustand “Klausurplanung” in den Zustand “Klausuranmeldung möglich”). Übergangsereignisse steuern die Systemübergänge. Übergangsereignisse bestehen aus:

- Startzustand
- Bedingung
- Aktion
- Folgezustand

In unserem Beispiel wäre der Anfangszustand “Klausurplanung”, der Folgezustand “Klausuranmeldung möglich”, die Bedingung “Klausurplanung abgeschlossen” und die Aktion “Klausuranmeldung möglich freigeben.”

Wie bei der Datenflußmodellierung aus *Kapitel 3.2.1* handelt es sich bei der Ereignismodellierung um eine grafikgestützte Methode. *Abbildung 42* zeigt die hierbei verwendeten Symbole.

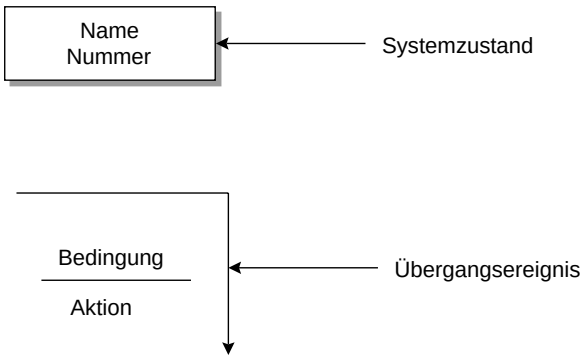


*Abbildung 42 Symbole bei Zustandsübergangsdiagrammen*

Wir wollen nun das Zustandsübergangsdiagramm für unser Klausurverwaltungsbeispiel aus *Kapitel 3.2.1.1* erstellen. Die Aufgabenstellung wird wie folgt erweitert:

- Anfang des Semesters wird vom Prüfungsamt ein Klausuranmeldezeitraum festgelegt und veröffentlicht. Zwei Wochen vor Beginn dieses Zeitraums ist die Klausurplanung abgeschlossen und das System druckt die Prüfungspläne. In den nächsten zwei Wochen bis zum Beginn des Prüfungszeitraums sind dann weder Klausuranmeldungen noch Änderungen im Prüfungsplan möglich.
- Während des Klausuranmeldezeitraum sind ausschließlich Klausuranmeldungen möglich. Einen Tag nach Ende des Klausuranmeldezeitraum druckt das System die Anmeldelisten. Danach beginnt die Planung des nächsten Klausurzeitraums.

Wir können fünf Zustände identifizieren:

- Klausurplanungsphase.
- Druck der Prüfungspläne (Planungsdruck).
- Warten auf den Anmeldezeitraum.
- Klausuranmeldezeitraum (Anmeldephase)
- Druck der Teilnehmerlisten (Teilnehmerlistendruck).

Die zugehörigen Bedingungen und Aktionen sind:

- 2 Wochen vor Anmeldezeitraum erreicht. Das System geht von der Klausurplanungsphase auf den Zustand “Planungsdruck” über. Die zugehörige Aktion ist “Planungsdruck veranlassen”.
- Pläne gedruckt. Das Systen geht vom Zustand Planungsdruck auf den Zustand “Warten auf Anmeldezeitraum” über. Die zugehörige Aktion ist “Anmeldezeitraum erwarten”.

- Anmeldezeitraum erreicht. Das Systen geht vom Zustand “Warten auf Anmeldezeitraum“auf die Anmeldephase über. Die zugehörige Aktion ist “Anmeldephase freigeben”.
- Ende Anmeldephase erreicht. Das System geht von der Anmeldephase auf den Zustand “Teilnehmerlistendruck” über. Die zugehörige Aktion ist “Teilnehmerlistendruck veranlassen”.
- Teilnehmerlisten gedruckt. Das Systen geht vom Zustand “Teilnehmerlistendruck“ auf die Klausurplanungsphase über. Die zugehörige Aktion ist “Klausurplanungsphase freigeben”.

*Abbildung 43* zeigt das Zustandsübergangsdiagramm dieses Beispiels:

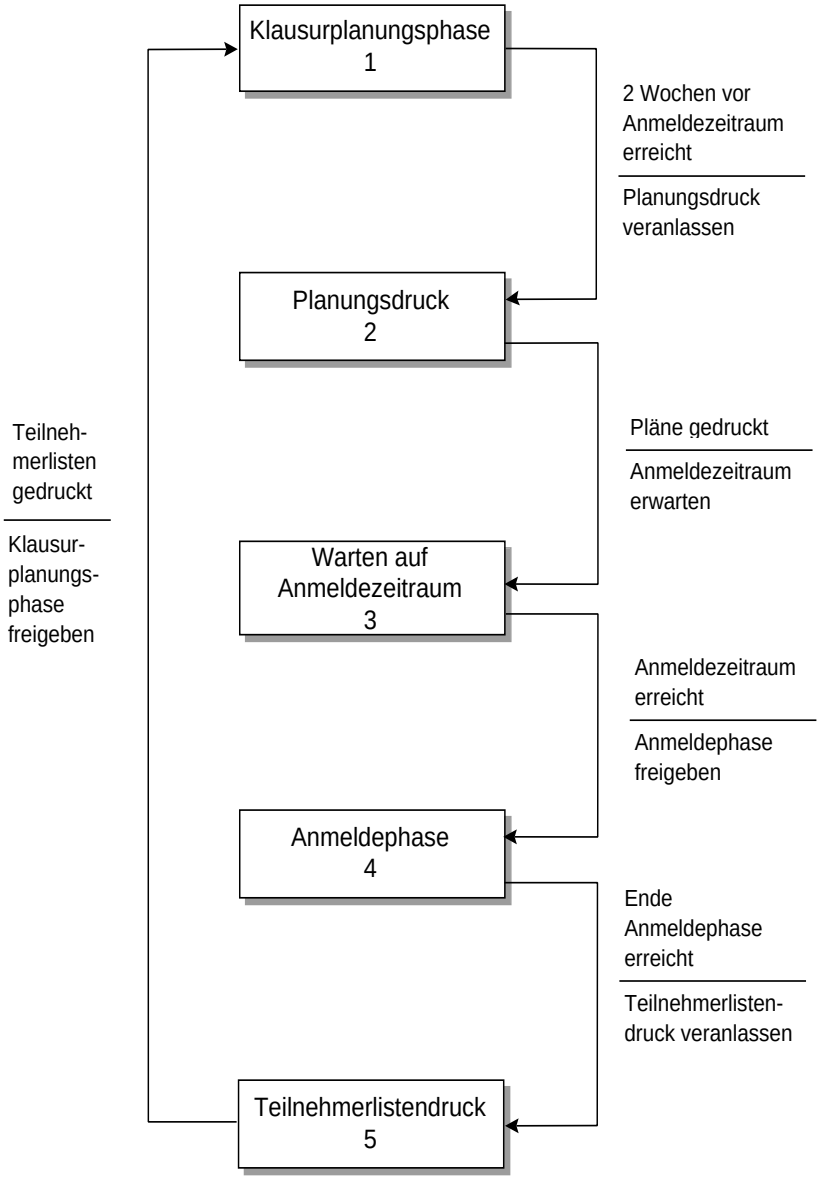


*Abbildung 43 Zustandsübergangsdiagramm des Klausurverwaltungsbeispiels*

Wir sehen, daß jeder Zustand genau einen Eingang und genau einen Ausgang besitzt. Dies ist nicht gezwungenermaßen so. Wir werden später ein Beispiel mit Zuständen mit mehreren Ein- und Ausgängen sehen.

Ebenso wie Datenflußdiagramme lassen sich Hierarchien von Zustansübergangsdiagrammen bilden. Wir veranschaulichen uns dies zunächst am Systemstart eines Unix-Computers (vgl. *Abbildung 44* und *Abbildung 45*)

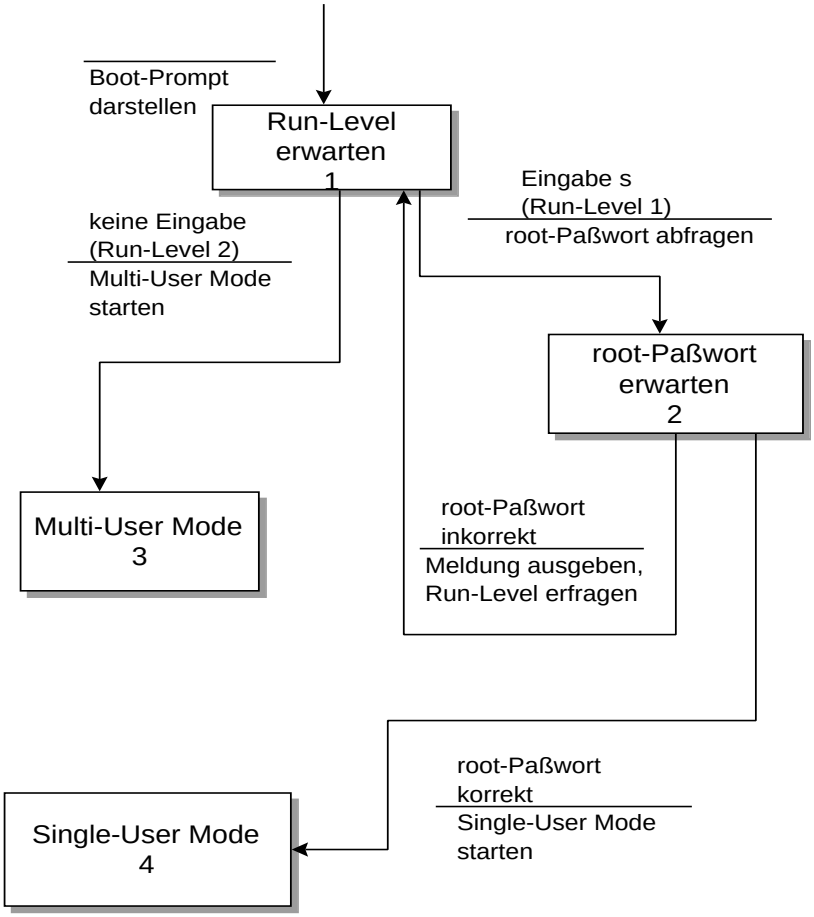


*Abbildung 44 Zustandsübergangsdiagramm für den Systemstart eines Unix-Rechners*

Zunächst betrachten wir *Abbildung 44*.

Nach dem Systemstart stellt ein Unix-Rechner einen Boot-Prompt dar. Erfolgt nach einer bestimmten Zeit keine Eingabe des Benutzers, fährt das System im Multi-User Mode hoch und Benutzer können sich am Rechner anmelden.

Wird “s” eingegen, muß der Rechner in den Single-User Mode starten. Da im Single-User Mode der Benutzer automatisch als System-Administrator “root” angemeldet wird (und damit alle Rechte auf dem Computer hat), fragt der Rechner das Administrator-Paßwort (root-Paßwort) ab. Wird das root-Paßwort korrekt eingegen, geht das System in den Single-User Mode. Im anderen Fall wir wieder der Boot-Prompt dargestellt. Überlegen Sie sich als Übung, warum es nicht sinnvoll ist, im Falle einer falschen Eingabe des root-Paßworts wieder zur Abfrage des root-Paßworts zu zurückzukehrten.

Wir sehen hier zu ersten Mal einen Zustand mit mehreren Ausgängen (“root-Paßwort erwarten”). In solchen Fällen müssen die Bedingungen, die zu den Folgezuständen führen sich gegenseitig ausschließen. Dies ist in *Abbildung 44* der Fall. Wurde das root-Paßwort korrekt eingegeben, kann die Eingabe nicht inkorrekt gewesen sein (und umgekehrt). Die Zustände Multi-User Mode und Single-User Mode sind Endzustände. Dies ist daran zu erkennen, daß kein Übergang zu einem Folgezustand möglich ist. Run-Level erwarten ist ein Anfangszustand. Anfangszustande erkennt man an der fehlenden Übergangsbedingung.

*Abbildung 45* stellt eine Verfeinerung des Zustands Multi-User Mode dar.

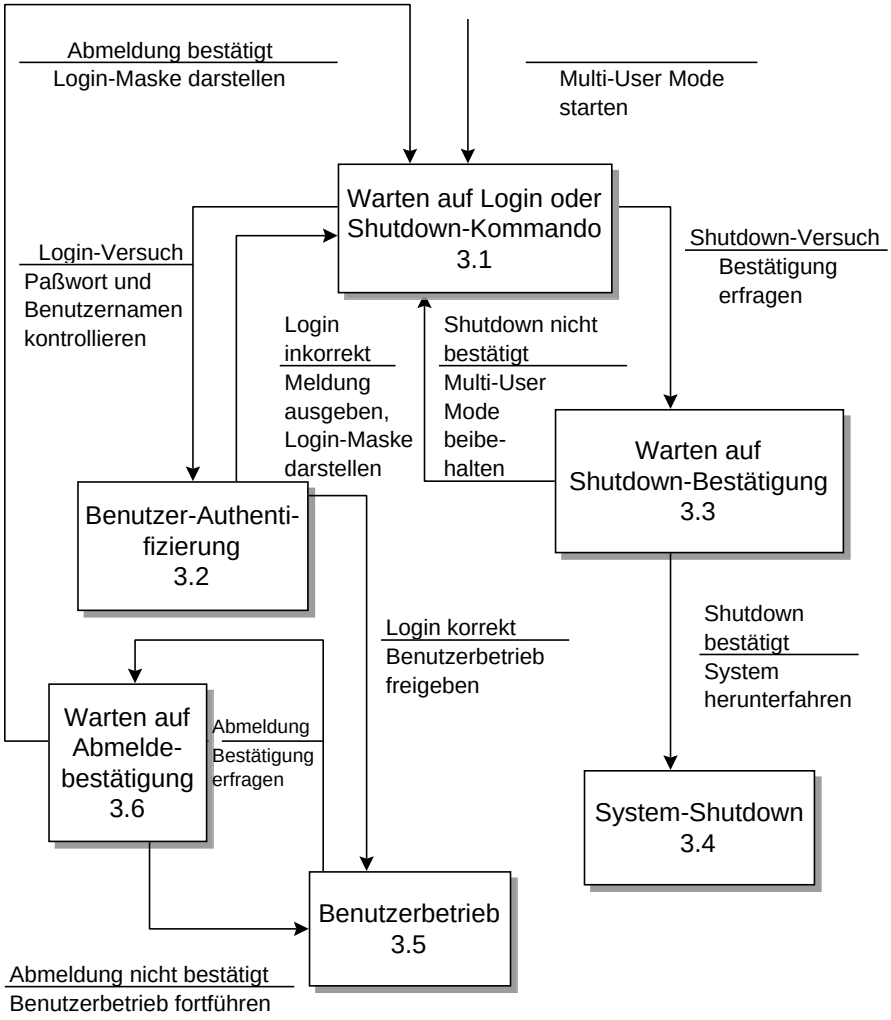


*Abbildung 45 Verfeinerung des Zustands Multi-User Mode aus Abbildung 44*

Strenggenommen widerspricht diese Vorgehensweise (Verfeinerungen durchzuführen) der Forderung, daß sich ein System zu jeder Zeit in genau einem Zustand befindet. Zustände, die verfeinert werden, heißen abstrakte Zustände. Befindet sich ein System in einem abstrakten Zustand, so fordern wir, daß sich das System in genau einem der Teilzustände des Abstrakten Zustandes befindet.

Für unser Beispiel bedeutet dies, daß der Rechner, wenn er den abstrakten Zustand Multi-User Mode (diesen Zustand haben wir ja in *Abbildung 45* verfeinert) erreicht hat, sich genau in einem der Teilzustände “Warten auf Login oder Shutdown-Kommando”, “Benutzer-Authentifizierung”, “Warten auf Shutdown-Bestätigung”, “System-Shutdown”, “Benutzerbetrieb” oder “Warten auf Abmeldebestätigung” befindet.

Namensgebung und Numerierung von Zuständen in hierarchischen Zustandsübergangsdiagrammen ist analog zur Vorgehensweise bei den Datenflußdiagrammen. Daher werde ich hier nicht weiter darauf eingehen.

*Generelle Vorgehensweise bei der Erstellung von Zustandsübergangsdiagrammen*

Zunächst werden die möglichen (einander ausschließenden) Zustände des Systems ermittelt. Danach versuchen wir die Bedingungen herauszufinden, unter denen ein System von einem Zustand in den anderen wechselt. Die Aktionen, die den Zustandswechsel herbeiführen, werden ermittelt.

Unter Einsatz der grafischen Beschreibungselemente wird das Zustandsübergangsdiagramm erstellt. Wird das Diagramm zu unübersichtlich, fassen wir zusammengehörige Zustände zu einem abstrakten Zustand zusammen und erhalten so eine Hierarchie von Zustandsübergangsdiagrammen.

### 3.3.2 Integration von Zustandsübergangs- und Datenflußdiagrammen bei der Modernen Strukturierten Analyse

Die Vorgehensweise bei der Integration von Zustandsübergangs- und Datenflußdiagrammen bei der Modernen Strukturierten Analyse machen wir uns zunächst an unserem Beispiel klar:

Als erstes erstellen wir das Zustandsübergangsdiagramm. Der Nummer des Zustands wird ein “Z” vorangestellt. *Abbildung 35* wird also zu *Abbildung 46*.

Vom Zustandsübergangsdiagramm gehen wir zu den Datenflußdiagrammen über. Aus dem Zustandübergangsdiagramm wird das Diagramm 0 der Datenflußdiagramme abgeleitet. Dabei wird:

- Jeder Zustand des Zustandsübergangsdiagramms zu einem Prozeß im Diagramm 0.
- Ein Kontrollprozeß eingeführt, der die Zustandsmodellierung in die Datenflußdiagrammhierarchie übernimmt.
- Datenflüsse zwischen dem Kontrollprozeß und den die Zustände repräsentierenden Prozessen, an denen wir erkennen können, wann welcher Prozeß vom Kontrollprozeß gestartet wird.

*Abbildung 47* zeigt das Diagramm 0 unseres Beispiels.

Wir sehen:

- Bei der Numerierung der Prozesse in der Datenflußdiagramm-Hierarchie wird den Prozeßnummern ein “D” vorangestellt.

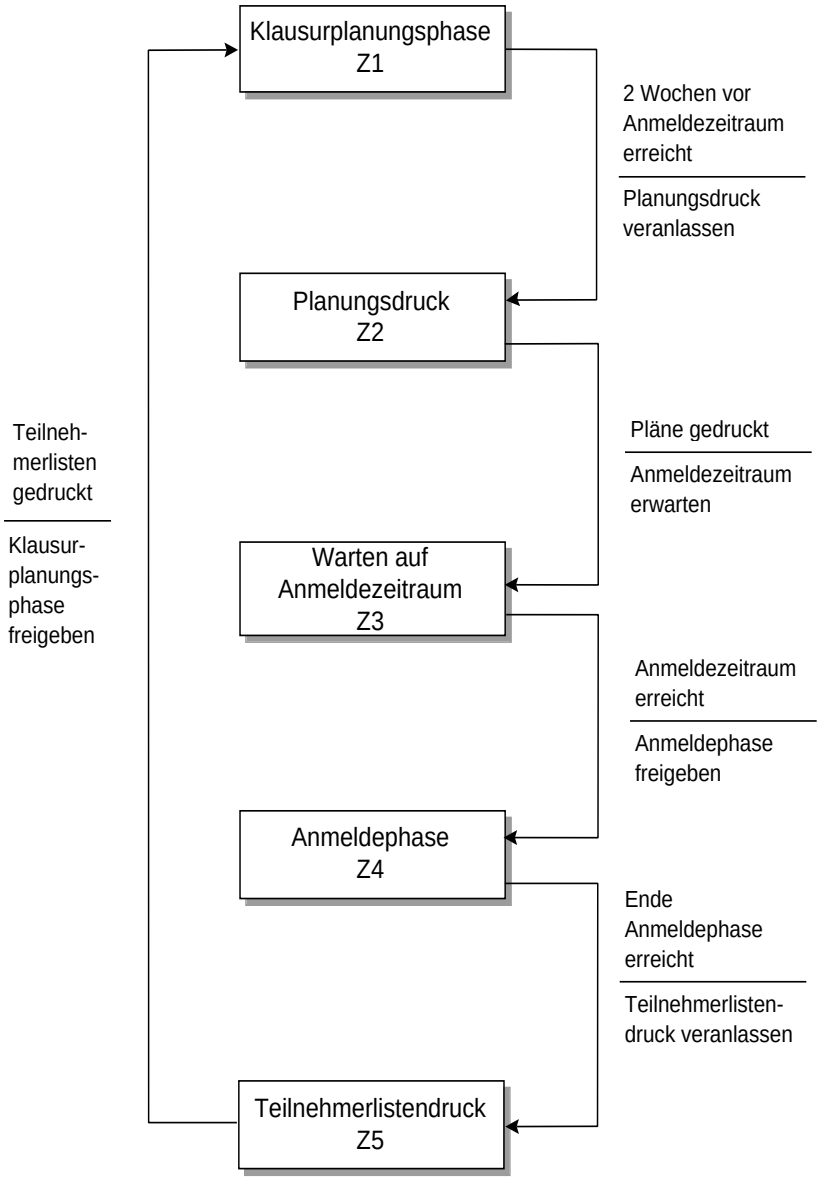


*Abbildung 46 Zustandsübergangsdiagramm des Klausurverwaltungsbeispiels bei der Modernen Strukturierten Analyse*

- Die Namen der Datenflüsse von den die Zustände repräsentierenden Prozessen zum Kontrollprozeß sind die Bedingungen, die zu einem Übergang zu einem anderen Zustand führen (vgl. *Abbildung 46* und *Abbildung 47*).
- Die Namen der Datenflüsse vom Kontrollprozeß zu den die Zustände repräsentierenden Prozessen sind die Aktionen, die zum Übergang auf den entsprechenden Prozeß führen (vgl. *Abbildung 46* und *Abbildung 47*).
- Die Ein- und Ausgabeströme der einzelnen Prozesse werden hinzugefügt. Diese kennen wir ja bereits aus *Kapitel 3.2.1.1*, so daß ich auf eine weitere Diskussion verzichte. Im Gegensatz zum Diagramm 0 der Strukturierten Analyse (vgl. *Abbildung 21*) werden die Terminatoren mit in das Diagramm aufgenommen.
- Die Datenspeicher werden hinzugefügt. Auch dies ist identisch mit der Vorgehensweise in *Kapitel 3.2.1.1*.

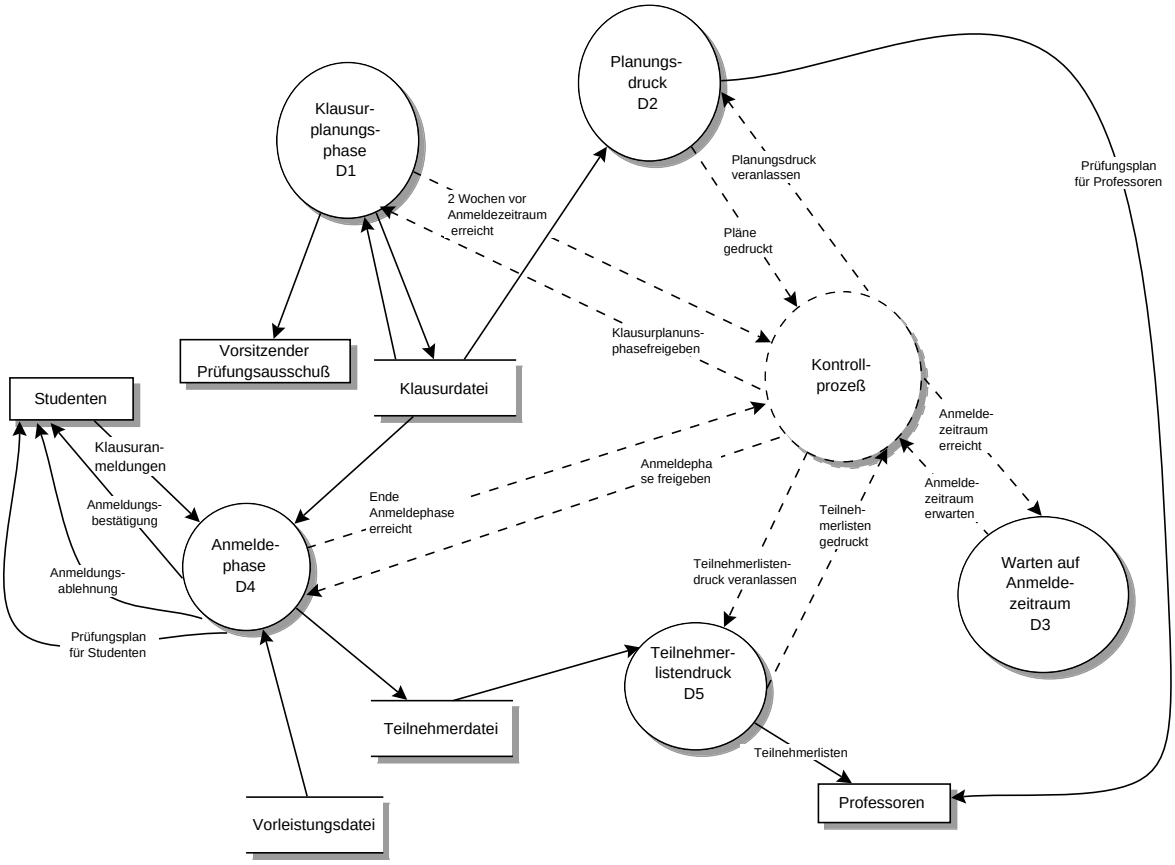


*Abbildung 47 Das Diagramm 0 des Klausurverwaltungsbeispiels bei der Modernen Strukturierten Analyse*

Das Diagramm 0 der Modernen Strukturierten Analyse ist dem Diagramm 0 der Strukturierten Analyse ähnlich. Die Prozesse “Klausurplanungsphase”, “Anmeldephase” und “Teilnehmerlistendruck” der Modernen Strukturierten Analyse entsprechen im wesentlichen den Prozessen “Klausuren planen”, “Klausuranmeldung überprüfen” und “Listen ausgeben” der Strukturierten Analyse (wir hätten auch diese Namen verwenden können). Hinzugekommen ist der Kontrollprozeß, aus dessen Kontrollflüssen ablesbar ist, unter welchen Bedingungen welche Funktionen ausgelöst werden und zwei Prozesse, die das zeitliche Verhalten des Systems vervollständigen (“Warten auf Anmeldezeitraum” und “Planungsdruck“).

Aus dem Diagramm 0 erstellen wir nun durch Aggregation das bereits bekannte Kontextdiagramm (vgl. *Abbildung 48*). Das Kontextdiagramm entspricht dem Kontextdiagramm der Strukturierten Analyse, so daß sich eine weitere Diskussion erübrigt.

Nun beginnen wir mit der Verfeinerung unserer Prozesse. Die Prozesse “Teilnehmerlistendruck”, “Warten auf Anmeldezeitraum” und “Planungsdruck“ des Diagramm 0 sind bereits so elementar, daß sie nicht weiter verfeinert werden müssen. Hierfür können wir sofort Prozeßspezifikationen schreiben.

Auch der Prozeß Klausurplanungsphase ist nicht weiter verfeinerbar. Im Diagramm 0 der Strukturierten Analyse (vgl. *Abbildung 24*) hatten wir “Klausuren planen” in “Klausuren aufnehmen” und “Prüfüngspläne ausgeben“ aufgeteilt. Durch unserer Ereignis-

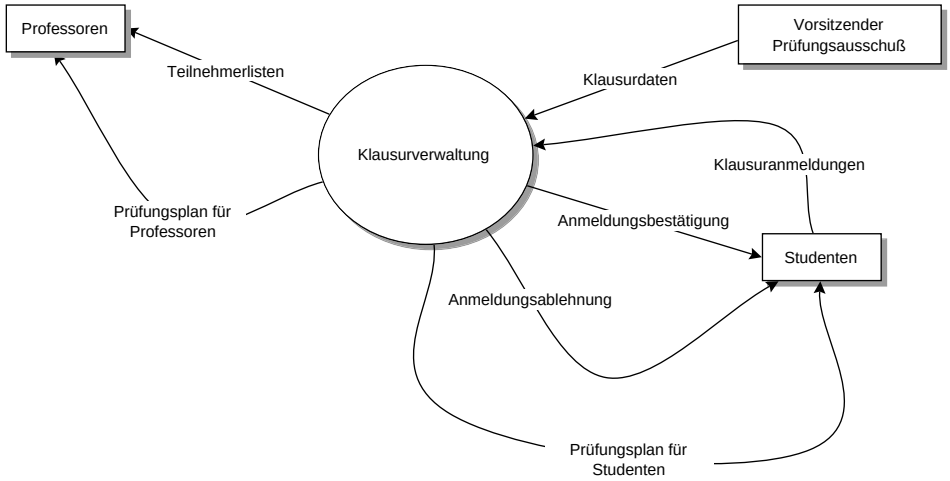


*Abbildung 48 Das Kontextdiagramm des Klausurverwaltungsbeispiels bei der Modernen Strukturierten Analyse*

modellierung ist “Prüfüngspläne ausgeben“ jedoch bereits im Diagramm 0 sichtbar (Planungsdruck).

Wir erstellen also nur noch das Diagramm D2, indem wir den Prozeß “Anmeldephase” verfeinern (vgl *Abbildung 49*)).

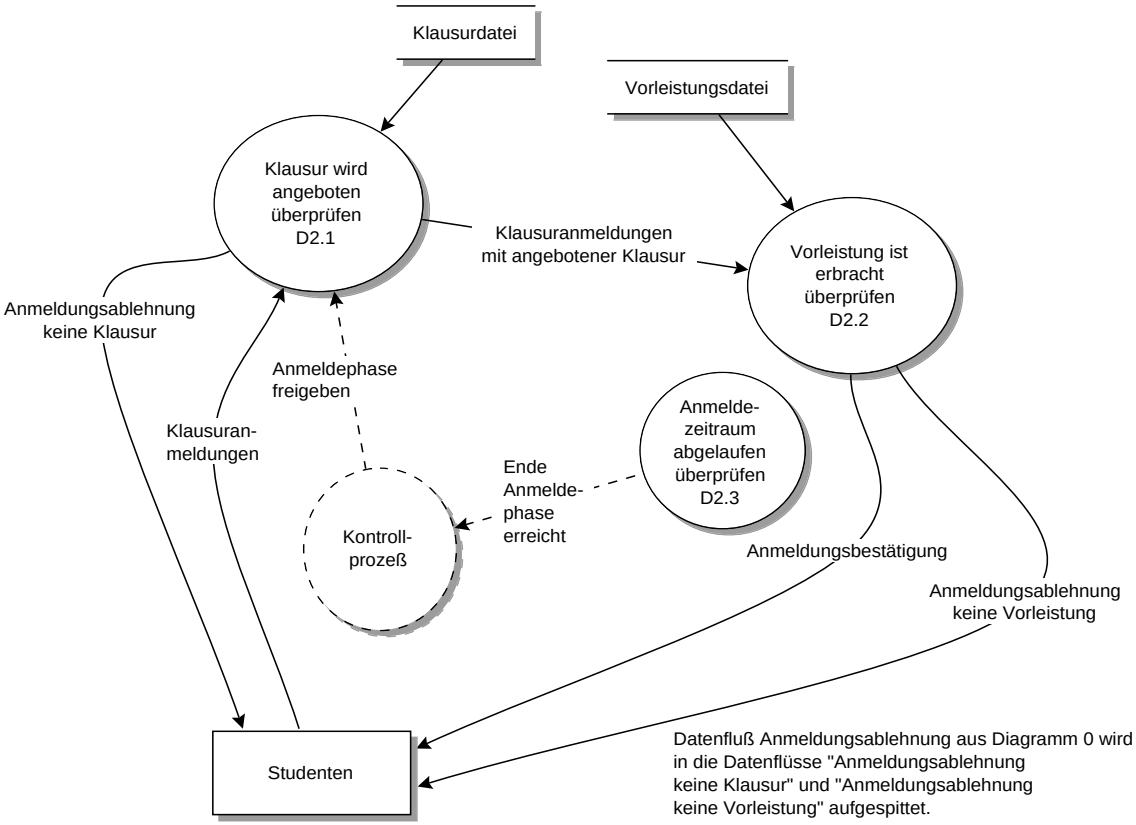


*Abbildung 49 Das Diagramm D2 des Klausurverwaltungsbeispiels bei der Modernen Strukturierten Analyse*

Das Diagramm D2 entspricht im wesentlichen dem Diagramm 2 der Strukturierten Analyse (vgl. *Abbildung 26*). Hinzu kommt:

- Der Kontrollprozeß und seine Kontrollströme.
- Der Prozeß “Anmeldezeitraum abgelaufen überprüfen”. Diesen Prozeß benötigen wir, um festzustellen, wann der Anmeldezeitraum endet. Bei Ende des Anmeldezeitraum teilt dieser Prozeß dem Kontrollprozeß über den Kontrollfluß “Ende Anmeldephase erreicht” das Ende der Anmeldephase mit. Der Kontrollprozeß kann nun auf die nächste Phase übergehen. “Anmeldezeitraum abgelaufen überprüfen” benötigt keine anderen Ein- oder Ausgaben, da nur die Systemzeit des Rechners abgefragt werden muß.
- Die Schnittstelle (Terminator) Studenten. Die Terminatoren werden in allen Diagrammen aufgeführt.

Damit ist die Zustandsübergangs- und Datenflußmodellierung unseres Beispiels beendet.

### 3.3.3 Entity-Relationship Modellierung

Ein weiterer Schwachpunkt der klassischen Strukturierten Analyse ist die kaum vorhandene Datenorientierung. Außer dem Datenkatalog wird keine weitere Datenmodellierung vorgenommen. In der Modernen Strukturierten Analyse wird Datenmodellierung nach dem Entity-Relationship Prinzip aufgenommen.

Entity-Relationship Modellierung haben Sie bereits in Datenbanken kennengelernt. Ich werde mich hier daher auf eine kurze Wiederholung beschränken.

Die Entity-Relationship Modellierung erfolgt in Entity-Relationship Diagrammen. Die dort erlaubten Symbole zeigt *Abbildung 50*. Es gibt allerdings weitere Möglichkeiten, Entity-Relationship Modelle darzustellen.

Wir behandeln nun die Symbole aus *Abbildung 50*.

Ein Entitätstyp (Objekttyp) beschreibt eine Menge materieller oder nicht materieller Dinge der realen Welt. Die Objekte des Objekttyps müssen voneinander unterscheidbar (ein einzelner PKW vom Objekttyp PKW z.B. anhand der Fahrgestellnummer) und notwendig für das zu erstellende System sein.

Entitätstypen verfügen über Datenelemente, die in diesem Zusammenhang oft Attribute genannt werden. Die Attribute des Objekttyps müssen für jede Instanz (jedes Objekt) des Objekttyps zutreffen.

*Abbildung 51* zeigt einen beispielhaften Objekttyp (PKW) mit einigen Attributen.

Objekte sind durch Relationen miteinander verbunden. *Abbildung 52* zeigt eine mögliche Relation zwischen Kunden und Artikel.

Relationen (Beziehungstypen) werden durch Rauten dargestellt und erhalten einen Namen. Die Kardinalität der Beziehung wird durch Zahlen an der Verbindung zwischen Objekt und Relation notiert. Kardinalität bedeutet in *Abbildung 52*:

- Mehrere (n) verschiedene Artikel können an einen Kunden geliefert werden,

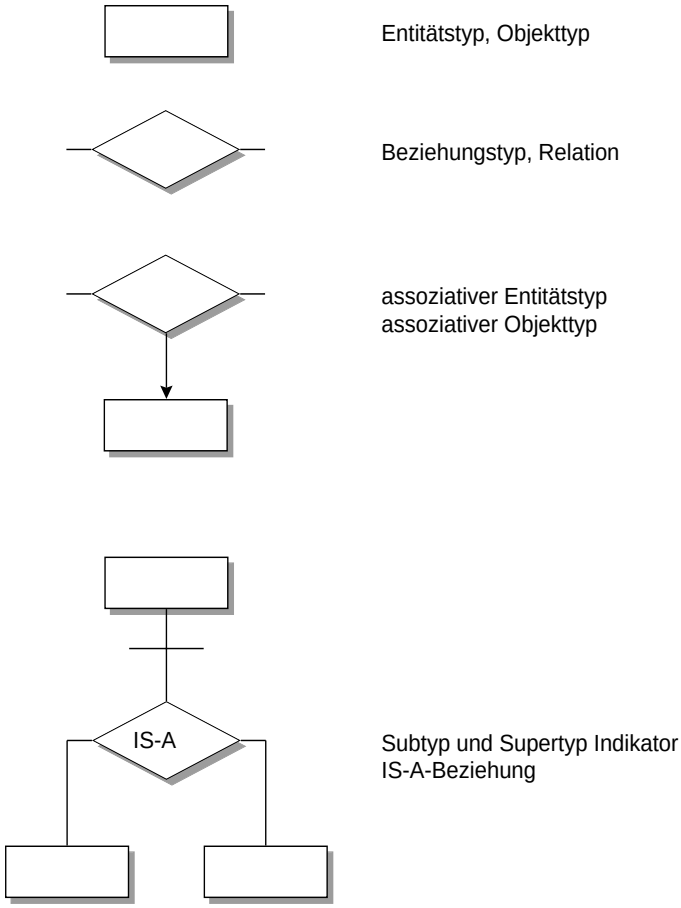


*Abbildung 50* Symbole bei der Entity-Relationship Modellierung

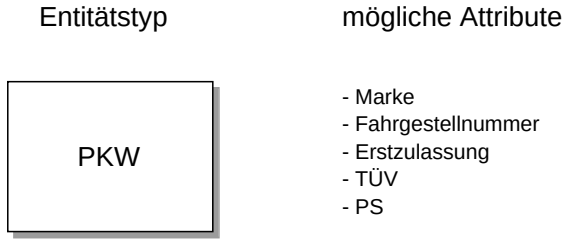


*Abbildung 51* Ein Entitätstyp mit möglichen Attributen

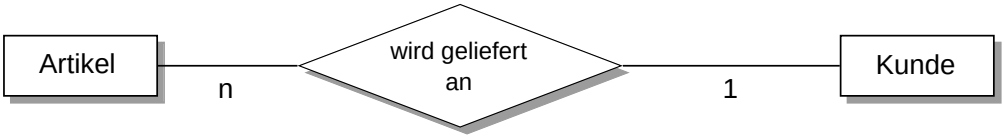


*Abbildung 52* Relation zwischen zwei Entitätstypen

- ein bestimmter Artikel wird aber an genau einen (1) Kunden geliefert.

Zwischen Entitätstypen können mehrere Beziehungen bestehen, wie *Abbildung 53* zeigt:

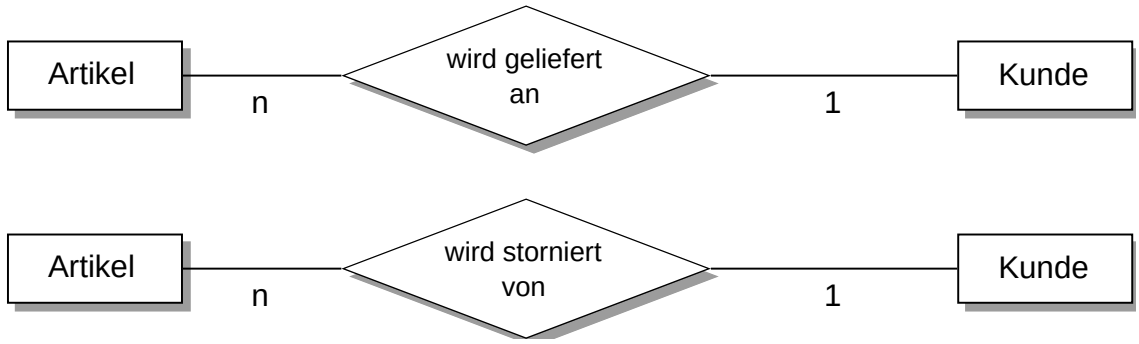


*Abbildung 53* Zwei Relationen zwischen zwei Entitätstypen

Die Kardinalität der zweiten Relation ist wie folgt zu lesen:

- Ein Kunde kann mehrere Artikel stornieren,
- ein Artikel kann aber nur von einem Kunden storniert werden.

Durch Relationen zwischen Entitätstypen können Objekte entstehen. Wenn ein Kunde beispielsweise eine Lieferung erhält, ist ein Lieferschein Bestandteil der Lieferung. Der Entitätstyp Lieferschein entsteht also durch unsere Relation aus *Abbildung 52*. Solche Entitäten heißen assoziative Entitäten. Sie sind Zwitter zwischen Objekttypen und Beziehungstypen. *Abbildung 54* zeigt ein Beispiel.

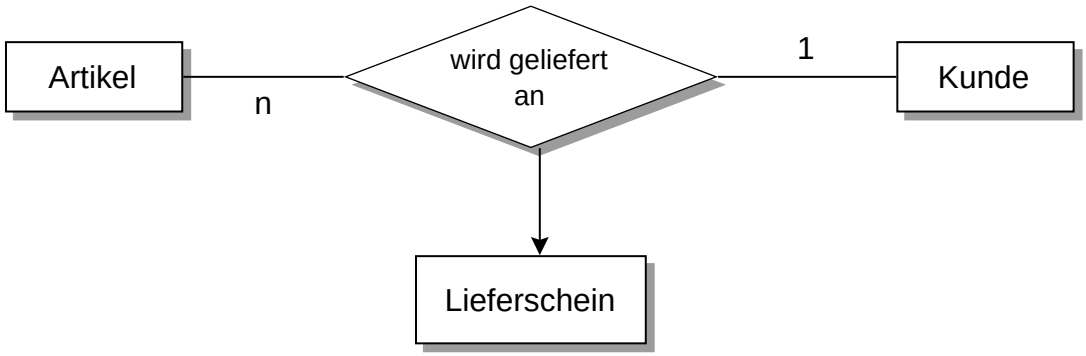


*Abbildung 54* Ein assoziativer Entitätstyp

Subtypen ordnen einem Objekttyp einen oder mehrere Unterobjekttypen zu. Unterentitätstypen übernehmen alle Attribute des ihnen zugeordneten Entitätstyp. Unterentitätstypen müssen weitere neueAttribute besitzen. *Abbildung 55* zeigt ein Beispiel.

### 3.3.4 Selbsttestaufgabe

Wir erweitern die Selbsttestaufgabe aus *Kapitel 3.2.4*. Rost-Stahl & Co KG ist ein konservatives Unternehmen ohne Verkauf über das Internet. Angebote und Aufträge wer-

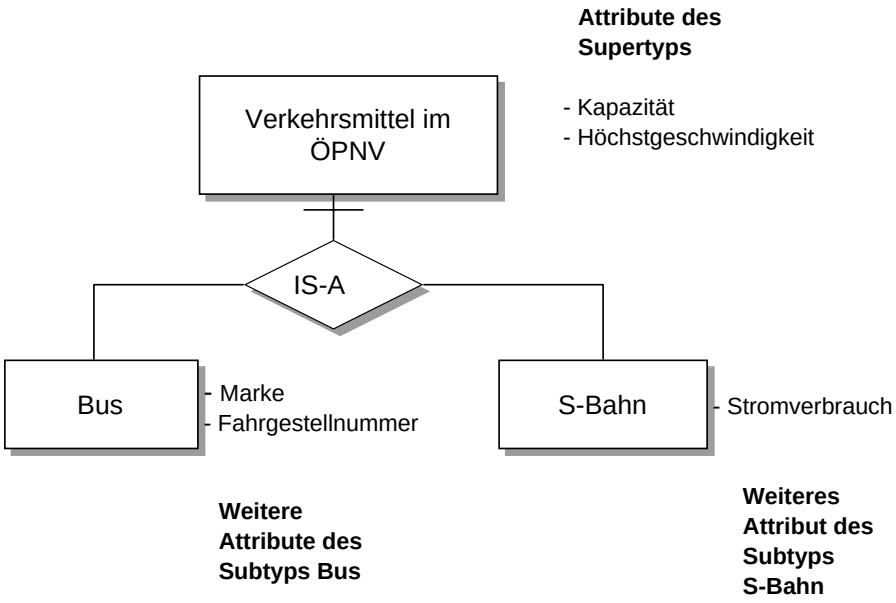


*Abbildung 55* Sub- und Supertypen

den nur während der Geschäftszeiten von 8.00 Uhr bis 17:00 bearbeitet. Während dieser Zeit dürfen aber keine anderen Teile der Auftragsbearbeitungssoftware (gemeint sind Versanddisposition und Fakturierung) benutzt werden, um die Performance des Systems sicherzustellen. Erstellen Sie eine Moderne Strukturierte Analyse für die geänderte Aufgabenstellung.

### 3.3.5 Lösung der Selbsttestaufgabe

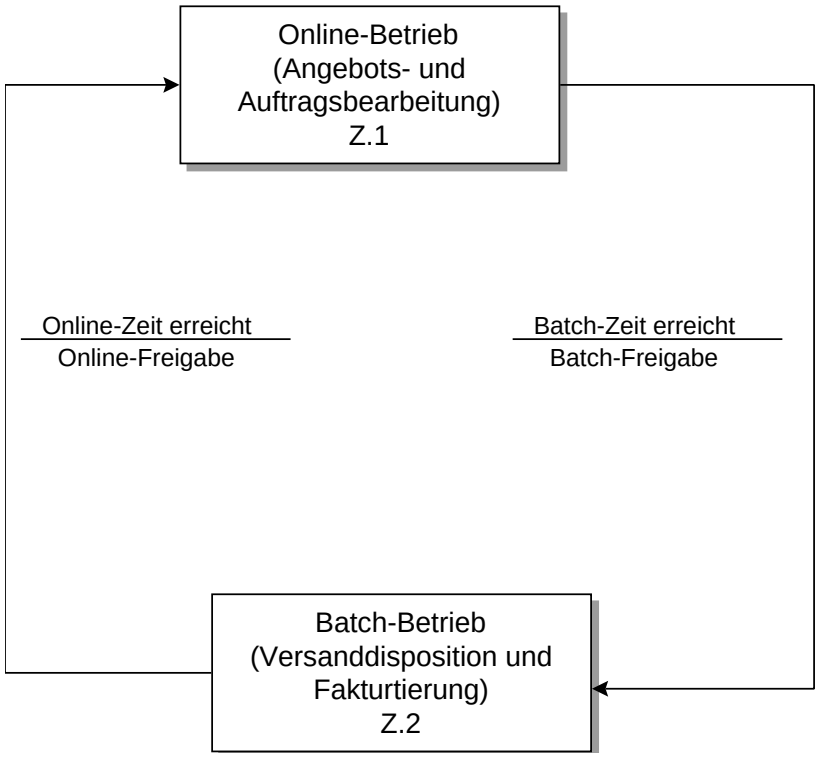


*Abbildung 56* Zustandsübergangsdiagramm der MSA des Anwendungsbeispiels

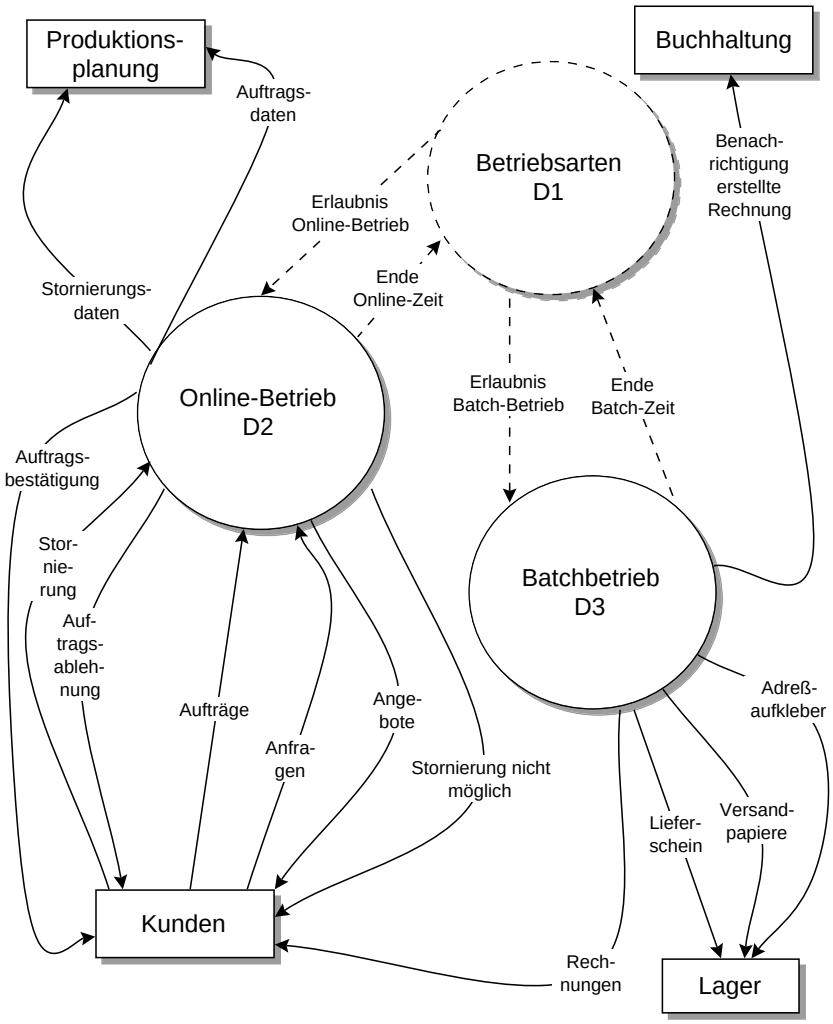


*Abbildung 57* Diagramm 0 der MSA des Anwendungsbeispiels

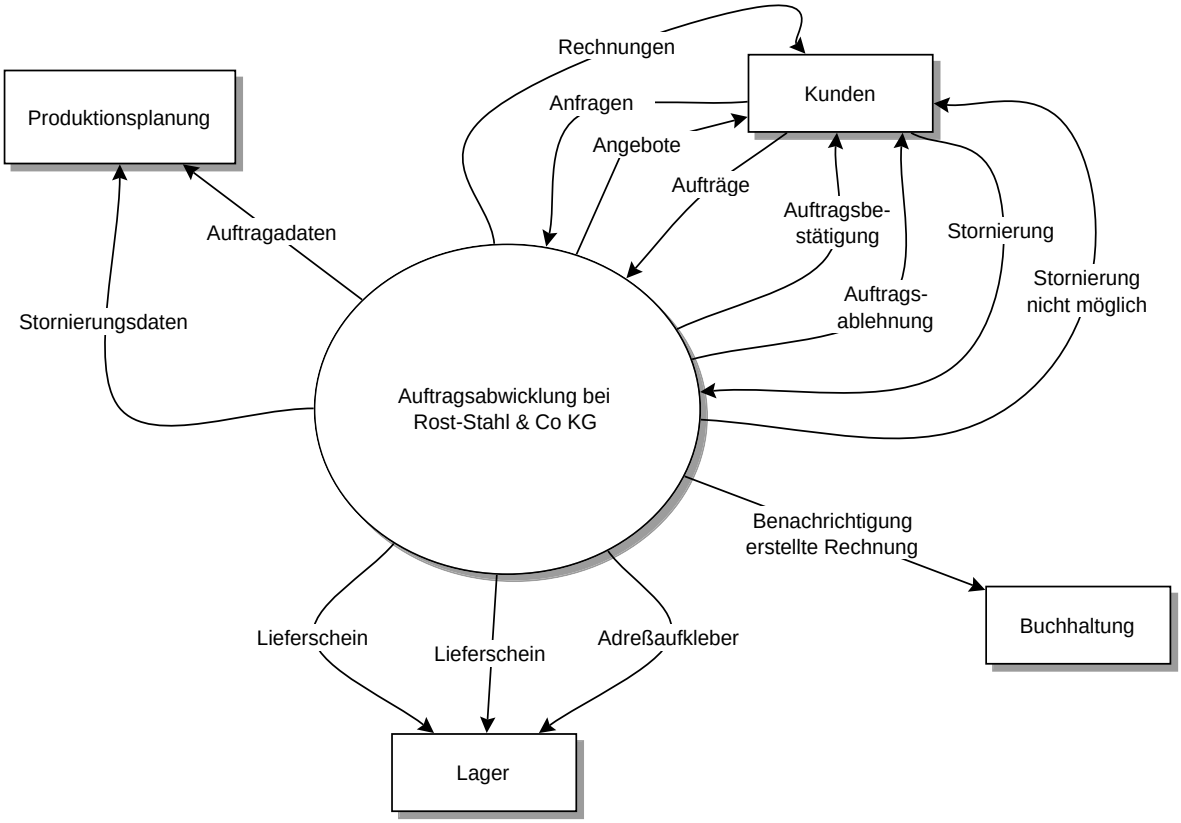


*Abbildung 58* Kontextdiagramm der MSA des Anwendungsbeispiels

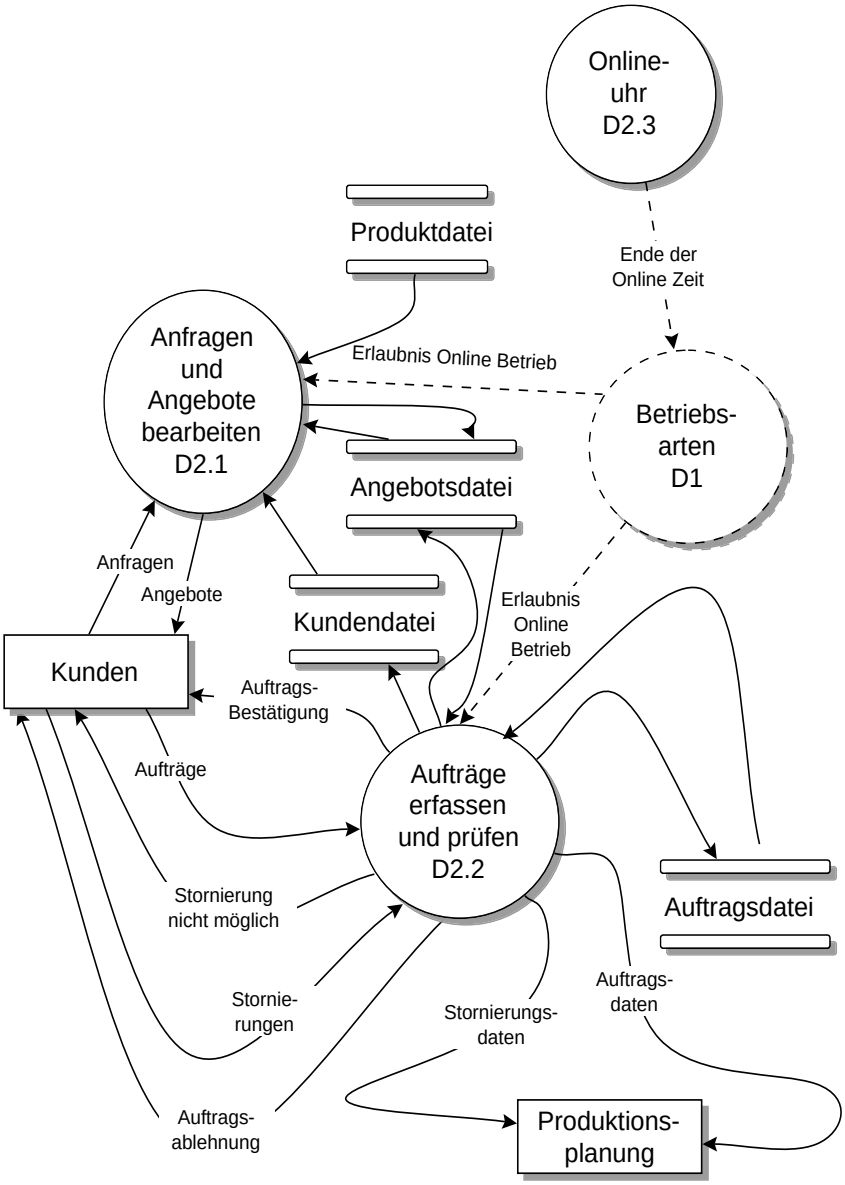


*Abbildung 59* Diagramm D2 der MSA des Anwendungsbeispiels

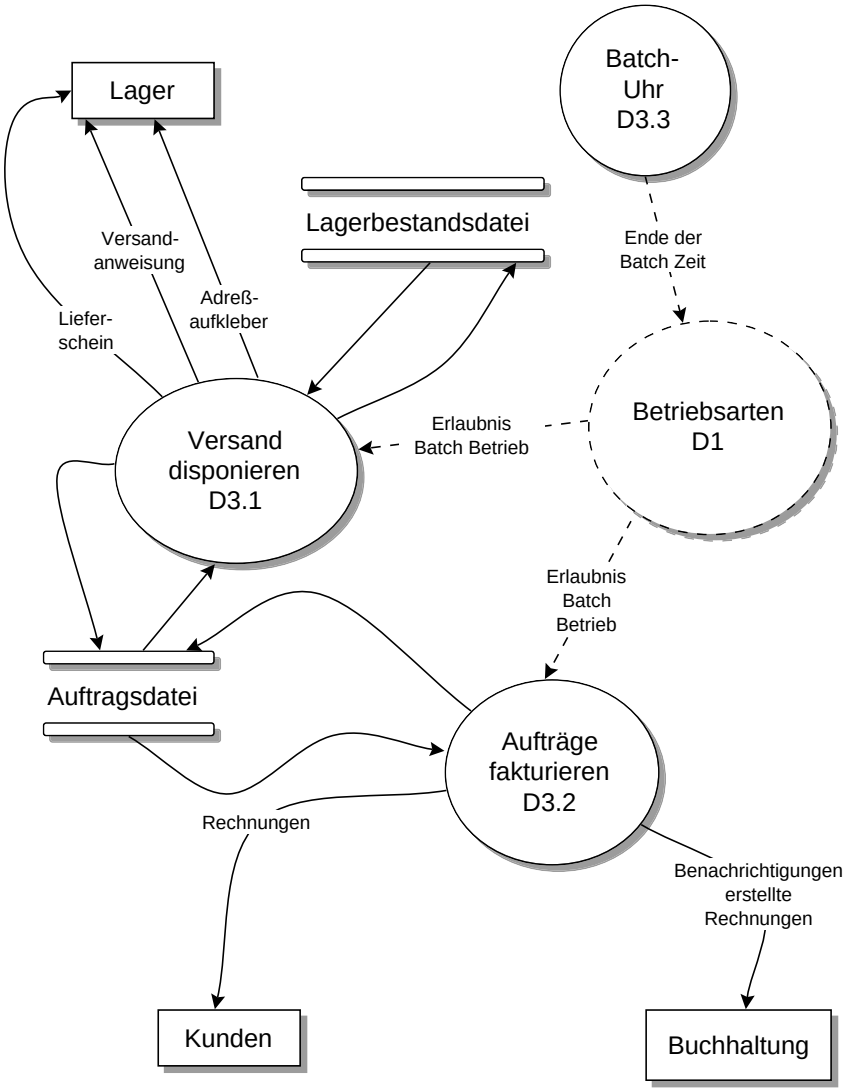


*Abbildung 60* Diagramm D3 der MSA des Anwendungsbeispiels

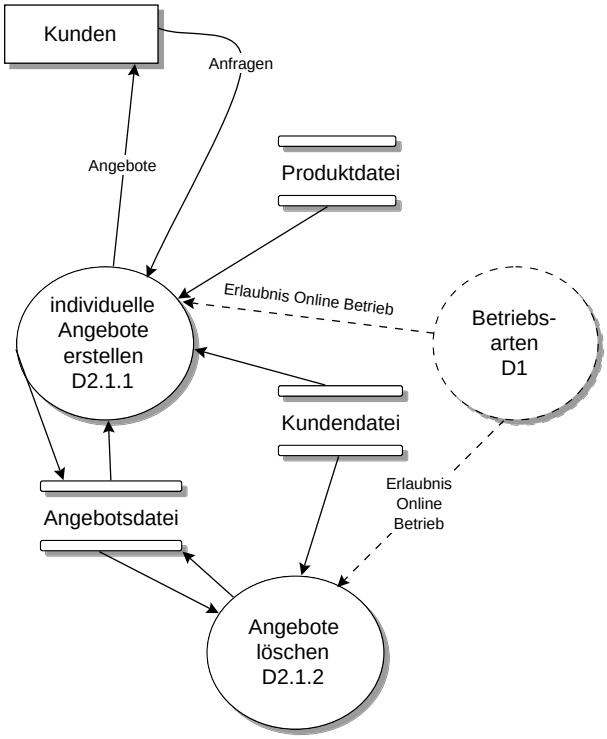


*Abbildung 61* Diagramm D2.1 der MSA des Anwendungsbeispiels

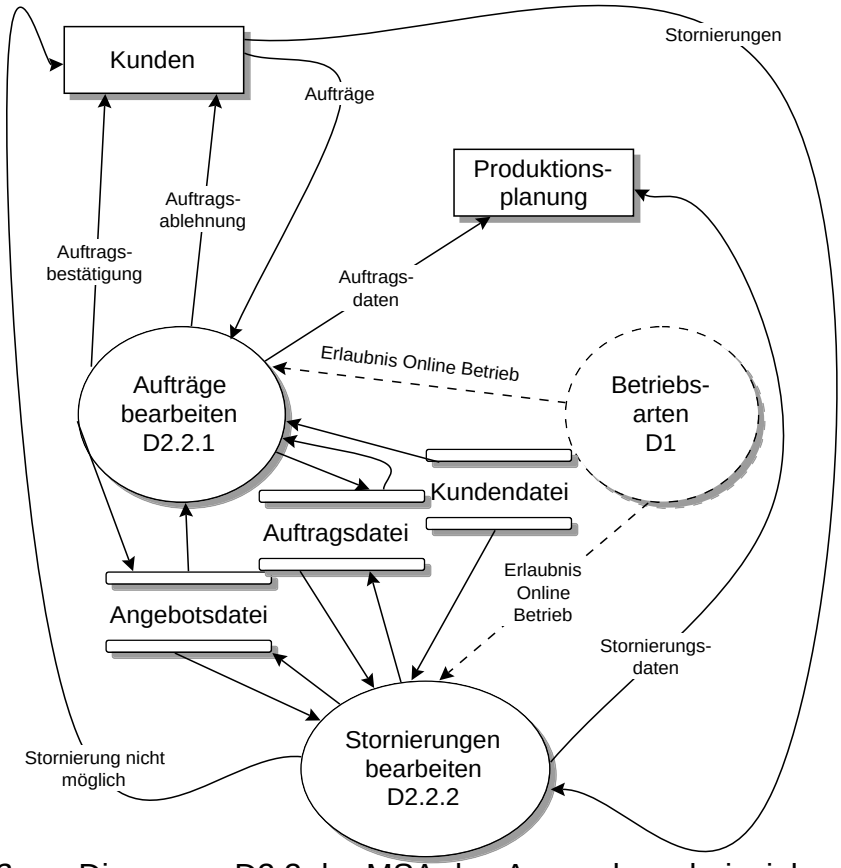


*Abbildung 62* Diagramm D2.2 der MSA des Anwendungsbeispiels

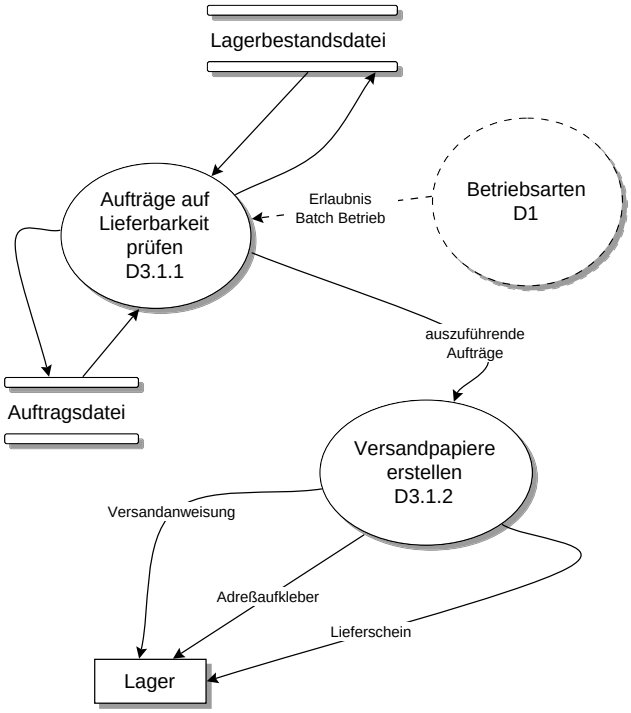


*Abbildung 63* Diagramm D3.1 der MSA des Anwendungsbeispiels

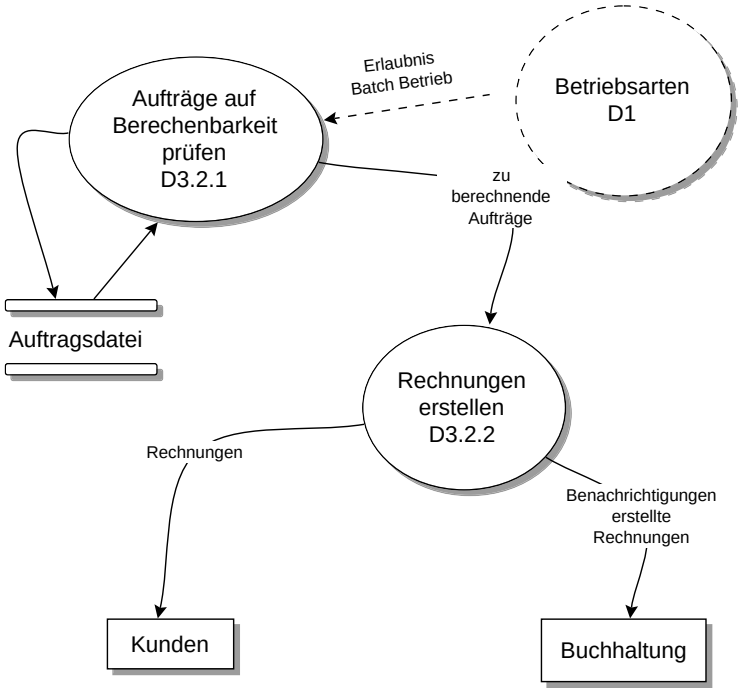


*Abbildung 64* Diagramm D3.2 der MSA des Anwendungsbeispiels

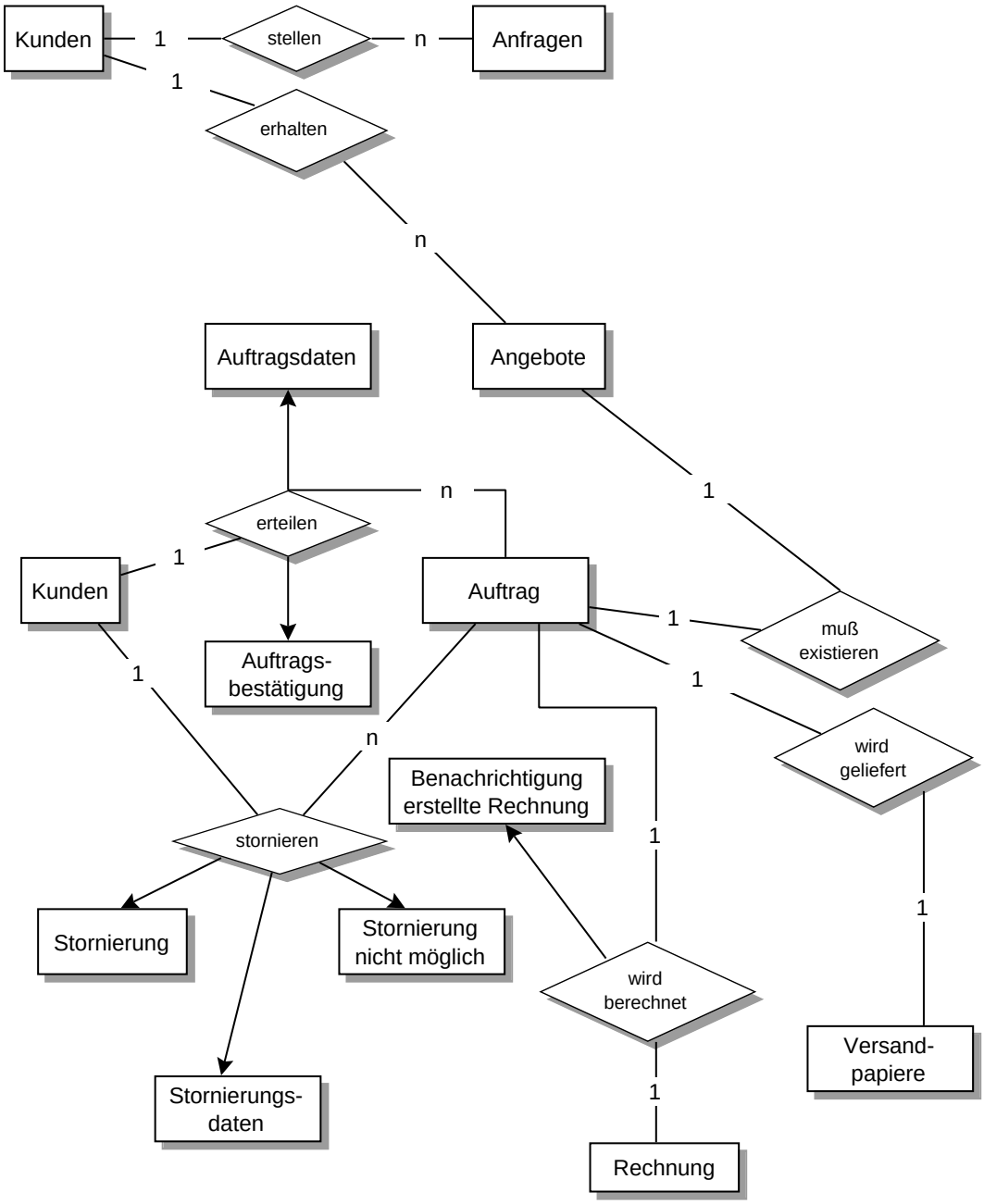


*Abbildung 65* Das Entity-Relationship Modell der MSA des Anwendungsbeispiels

Datenkatalog und Prozeßspezifikationen sind mit denen in *Kapitel 3.2.5* identisch.