Diplomarbeit

Universität Hamburg

Fachbereich Informatik

Arbeitsbereich Softwaretechnik

April 1998

Untersuchung und Bewertung von Methoden zur Unternehmens- und Geschäftsprozeßmodellierung im Rahmen der Informationssystementwicklung am Beispiel von EP/KID

Jan Crüsemann

Betreuung

Prof. Dr. Heinz Züllighoven,

Prof. Dr. Arno Rolf

Universität Hamburg

Diese Diplomarbeit wurde dem Fachbereich Informatik der Universität Hamburg zur teilweisen Erfüllung der Anforderungen zur Erlangung des Titels Diplom-Informatiker eingereicht.

Erklärung

Ich versichere hiermit, diese Arbeit selbständig und unter ausschließlicher Zuhilfenahme der in der Arbeit aufgeführten Hilfsmittel erstellt zu haben.

Hamburg, 28. April 1998

_____

Jan Crüsemann
Helgolandstraße 101
22846 Norderstedt
Tel.: 040 - 535 80 28

Betreuung

Prof. Dr. Heinz Züllighoven
Arbeitsbereich Softwaretechnik

Prof. Dr. Arno Rolf
Arbeitsbereich Angewandte und Sozialorientierte Informatik

Fachbereich Informatik
Universität Hamburg
Vogt-Kölln-Straße 30
22527 Hamburg

1 Einleitung

Geschäftsprozeßmodellierung und *-optimierung*, *Workflow-Management-Systeme*, *Unternehmensmodellierung*, *Business Process Reengineering* sind Schlagworte, die heute sehr stark im Mittelpunkt des Interesses, sowohl in der Wirtschaft als auch im Bereich der universitären Forschung, stehen. In die Diskussion dieser Themen fließen dabei Gesichtspunkte verschiedenster Disziplinen, wie Wirtschaftswissenschaften, (Arbeits-) Psychologie oder Informatik, ein.

Der Themenbereich Geschäftsprozeßoptimierung ist von starkem Interesse in den Wirtschaftswissenschaften, denn immer mehr Unternehmen sind gezwungen, sich einem mehr und mehr globalen Wettbewerb zu stellen. Wie erfolgreich ein Unternehmen auf zunehmend kundenorientierten Märkten mit immer kürzer werdenden Produktinnovationszyklen ist, hängt dabei nicht allein von der Qualität seiner Produkte und Dienstleistungen ab, sondern auch von den Prozessen und Vorgehensweisen, mit denen diese Leistungen erbracht werden. Die Bewegung des „Total Quality Management“ sieht in der Qualität des Produktionsprozesses sogar das wichtigste Kriterium für ein erfolgreiches Produkt und seine Chancen am Markt.[1]

Die Informatik spielt dabei aus mehreren Gründen eine wesentliche Rolle. Zum einen versteht sie sich nicht zuletzt als die Wissenschaft von Abstraktion und Modellbildung, von Systemanalyse und -organisation. Informatiker sind schon von daher nicht selten damit befaßt, Unternehmen bei der Analyse und Reorganisation ihrer Geschäftsprozesse zu unterstützen. Zum anderen spielt die Unterstützung der Arbeit durch betriebliche Informations- und Anwendungssysteme eine zunehmend wichtige Rolle.

Durch eine stark anwachsende Vernetzung von Computersystemen, sowohl der Arbeitsplatzsysteme innerhalb der Unternehmen (LAN, „Intranet“), als auch zwischen den Unternehmen (WAN, „Extranet“) bis hin zu globaler Vernetzung (Internet), verschiebt sich der Betrachtungsschwerpunkt bei der Softwareentwicklung zunehmend auf die Unterstützung verteilter, kooperativer Arbeit. Dieses bedeutet aber zwangsläufig, daß zunehmend Aspekte sowohl der Ablauf- als auch der Aufbauorganisation von Unternehmen in den Softwareentwicklungsprozeß nicht nur einfließen, sondern von diesem auch verändert werden. A. Rolf sagt dazu:

> »Organisationsentwicklung ist heute nicht mehr ohne Softwarenutzung denkbar und Softwareentwicklung berührt fast immer Strukturen und Abläufe einer Organisation. [...] Je mehr sich die physikalischen Dinge wie Rechnungsformulare, Lieferscheine, Auf-

[1] Vgl. dazu etwa [MHS96].

> tragsbestätigungen etc. und auch die Wege dieser Dokumente durch die Organisation in der Software wiederfinden, desto mehr findet sich die Organisation, ihre Struktur und die Arbeitsabläufe in der Software wieder. Eine isolierte Betrachtung von Software und Organisation bedarf aus diesem Blickwinkel der Begründung.« ([Rol97], S. 5 f.)

Da einerseits das Gebiet der Geschäftsprozeßmodellierung und -optimierung noch sehr jung ist, andererseits von Seiten der Unternehmen ein sehr großer Handlungsbedarf in diese Richtung herrscht, ist es nicht verwunderlich, daß es eine Reihe sehr unterschiedlicher Sichtweisen und Produkte zu diesem Thema gibt. Auch hat sich bisher keine einheitliche Begriffsbildung konsolidieren können, was die herrschende Verunsicherung und Orientierungslosigkeit noch verstärkt.

Diese Arbeit verfolgt daher folgende Ziele:

- Im ersten Teil der Arbeit soll ein Einblick in die Themenlandschaft der Bereiche Analyse und Modellierung von Geschäftsprozessen und der Entwicklung betrieblicher Anwendungs- und Informationssysteme vor dem Hintergrund der am Arbeitsbereich Softwaretechnik des Fachbereiches Informatik der Universität Hamburg vertretenen Sichtweise auf diese Themenbereiche gegeben werden. Dazu werde ich zunächst diese Sichtweise kurz skizzieren. Anschließend soll eine kleine „Umschau“ vorgenommen werden. Vorgestellt und verglichen werden dabei unterschiedliche Ansätze und Sichtweisen sowohl aus dem Bereich der Forschung, als auch Produkte und Methoden, die zu diesem Themenbereich kommerziell erhältlich sind. Diese werde ich dann aus der Perspektive des am Arbeitsbereich Softwaretechnik vertretenen Standpunktes kritisch bewerten.

- Im Anschluß daran leite ich aus den vorgestellten Ansätzen und den dazu gegebenen Bewertungen einige Unterscheidungsmerkmale ab, anhand derer sich Methoden zu den genannten Themen aus dem Blickwinkel der am Arbeitsbereich Softwaretechnik vertretenen Sichtweise kategorisieren lassen.

- Im dritten Teil stelle ich ein Vorgehensmodell vor, welches ich im Rahmen dieser Diplomarbeit für die Geschäftsprozeßanalyse und Anwendungsentwicklung speziell in Verbindung mit dem Produkt EP/KID, einem Produkt der Unternehmensberatung Entitec, erstellt habe. Zur besseren Einordnung dieses Vorgehensmodells stelle ich dazu zunächst kurz die grundlegende Architektur von EP/KID vor.

- Abschließend unternehme ich dann den Versuch einer Bewertung des in dem davorliegenden Abschnitt vorgestellten anhand der in den ersten Abschnitten angestellten Betrachtungen. Dabei soll sowohl das Produkt EP/KID an sich, als auch das entwickelte Vorgehensmodell aus der Perspektive der am Arbeitsbereich Softwaretechnik vertretenen Sichtweise und den daraus abgeleiteten Bewertungskriterien kritisiert werden.

2 Einordnung

Zur besseren Einordnung des weiterhin Gesagten möchte ich zunächst den am Arbeitsbereich Softwaretechnik des Fachbereichs Informatik der Universität Hamburg vertreten Standpunkt verdeutlichen.

2.1 STEPS - Ein Methodenrahmen

Unter der Leitung von Christiane Floyd wird hier der *Methodenrahmen* STEPS (= Softwaretechnik für evolutionäre, partizipative Systementwicklung) gelehrt.[2] Die Definition der Begriffe Methodenrahmen und Methode übernehme ich in diesem Zusammenhang von Gryczan:

> »Ein **Methodenrahmen** verkörpert eine grundlegende Sichtweise der Softwareentwicklung, bezieht sich auf eine Klasse von Anwendungsbereichen und gibt Richtlinien vor für die Auswahl von *Methoden*, Werkzeugen und Organisationsformen.
>
> Eine **Methode** stellt einen Satz von Werkzeugen und Darstellungsmitteln zur Verfügung. Dazu beinhaltet sie eine Vorgehensweise, die den Einsatz der Werkzeuge und Darstellungsmittel als Menge von Regeln anleitet.« ([Gry95], S. 2)

In dieser Definition des Begriffs Methodenrahmen wird bereits zum Ausdruck gebracht, daß STEPS als Anleitung zur Verwendung von (geeigneten) Methoden zu verstehen ist. Es sollen keine Vorschriften, was das Vorgehen bei der Erstellung von Software anbelangt, gemacht werden.

STEPS konzentriert sich auf die Erstellung von Softwaresystemen, die in menschliches Arbeitshandeln eingebettet werden sollen und stellt in diesem Zusammenhang einen menschenzentrierten Ansatz dar.

Bei der in STEPS eingenommenen, grundlegenden Sichtweise wird Softwareentwicklung als kooperativer Kommunikations- und Lernprozeß aufgefaßt. Mehrere Personen verschiedenen fachlichen Hintergrunds mit unterschiedlichem Vorwissen und unterschiedlichen Interessen sind an dem Softwareentwicklungsprozeß beteiligt (Auftraggeber, Entwickler, Sachbearbeiter der Fachabteilungen, etc.). Sie entwickeln gemeinsam eine Folge von aufeinander aufbauenden Versionen des Softwaresystems, die immer wieder validiert und weiterentwickelt werden. Die Validierung des so entstehenden Softwaresystems erfordert viel Kommunikation zwischen den am Entwicklungsprozeß beteiligten Perso-

[2] Für eine detailiertere Einführung in STEPS siehe etwa [Flo91].

nen. Gleichzeitig evolviert so zunehmend ein gemeinsames Wissen über das Fachgebiet des jeweils anderen, insbesondere der Entwickler über den Anwendungsbereich.

Mit der in STEPS eingenommenen Perspektive sind bestimmte Wertvorstellungen verknüpft. Die entwickelten Softwaresysteme sollen den Benutzer in seinem Arbeitshandeln angemessen unterstützen, ihm nicht eine Arbeitsweise vorschreiben. Aus diesem Grund wird in STEPS die Partizipation des zukünftigen Benutzers an der Entwicklung des Softwaresystems als wesentlich angesehen.

Softwareentwicklung wird hier also nicht ausschließlich als Herstellung des Produktes, sondern gleichermaßen als ein Erkenntnisprozeß aller Beteiligten angesehen. STEPS will sich mit dieser prozeßorientierten Sichtweise abgrenzen von anderen, produktorientierten Ansätzen, die von fest vorgegebenen Anforderungen an das Softwaresystem ausgehen und darauf aufbauend ein „wasserfallartiges" Vorgehen mit verschiedenen Phasen von der Anforderungsermittlung bis hin zur Produktauslieferung vorgeben.

2.2 Leitbilder in der Softwareentwicklung

Von Züllighoven et al. wird dazu geraten, im Rahmen der Softwareentwicklung ein *Leitbild* zu verwenden. Ein solches Leitbild soll den Entwicklern und den am Entwicklungsprozeß beteiligten Benutzern gleichermaßen eine Orientierungshilfe sein bei der Vorstellung, wie das zukünftige System und der Umgang mit ihm aussehen soll. Ihm liegen bestimmte Vorstellungen über das Verhältnis zwischen dem Softwaresystem und dem Anwender zugrunde. Der Begriff Leitbild wird bei Gryczan folgendermaßen definiert:

> »Ein **Leitbild** ist eine benannte, mit Absicht eingenommene grundsätzliche Sichtweise. Es ist eine Orientierung, die von Menschen angenommen wird, anhand der sie einen Ausschnitt von Realität wahrnehmen, verstehen und gestalten. Das Leitbild bestimmt, wie Entwickler und Benutzer bei der Systementwicklung wechselseitig miteinander umgehen und beschreibt Gestaltungsziele bei der Softwareentwicklung.« ([Gry95], S. 99)

Ausgehend von der in STEPS vertretenen Sichtweise wird am Arbeitsbereich Softwaretechnik das Leitbild vom „gut ausgestatteten Arbeitsplatz für qualifizierte menschliche Tätigkeit"[3] gelehrt. Dieses ist von der Grundannahme geprägt, daß der Benutzer ein qualifizierter Experte seines Fachgebietes ist. Er besitzt genügend Kompetenz, die Tätigkeiten zur Erledigung seiner Aufgaben selber zu organi-

[3] Für eine detailliertere Beschreibung des Leitbildes und der in seinem Zusammenhang verwendeten Entwurfsmetaphern vergleiche u.a. [BZ90], [GZ92], [KGZ94], [Rie95], [Gry95].

sieren; es gibt keine Veranlassung, ihm durch vordefinierte Arbeitsabläufe Restriktionen aufzuerlegen, die seine flexible Arbeitgestaltung einschränken würden. Anwendungssysteme sollen so gestaltet sein, daß sie den Benutzer bei der Erledigung seiner Aufgaben unterstützen, nicht ihn anleiten. Der Benutzer trifft die Entscheidungen, mit welchen Mitteln er seine Handlungsziele erreicht. Züllighoven et al. sprechen in diesem Zusammenhang von *reaktiven Systemen*, die sie wie folgt charakterisieren:

> »Reaktive Systeme sind [...] so konstruiert, daß die wesentliche „Richtung" der Ereignisse von „außen nach innen" geht, d.h. die Benutzereingaben bestimmen, welche Komponenten des Systems wie aktiv werden. Hat eine Komponente ihre Dienstleistung erbracht, wartet das System auf das nächste äußere Ereignis. Traditionell werden Anwendungssysteme genau umgekehrt, d.h. von innen nach außen, konstruiert. Das System befindet sich in einem bestimmten Zustand, in welchem eine Menge von Eingaben zulässig ist.« ([KGZ94], S. 27)

In [Gry95] grenzt Gryczan das hier vorgestellte „Leitbild vom gut ausgestatteten Arbeitsplatz für qualifizierte menschliche Tätigkeit" ab gegenüber dem Leitbild der Fabrik, dem im Gegensatz zu der geforderten reaktiven Charakteristik des zu entwickelnden Anwendungssystems eine sehr ablaufsteuernde Sichtweise zugrunde liegt. Gestaltungsziel dabei ist es, eine möglichst hochgradige Automatisierung der Arbeitsabläufe zu erreichen. Die damit verbundene Wertvorstellung ist von der Annahme geprägt, daß menschliches Arbeiten stark fehleranfällig ist und nach Möglichkeit durch maschinelle Tätigkeit ersetzt werden sollte. Wir werden sehen, daß dieses eine weit verbreitete Sichtweise, gerade im Bereich von Workflow-Management Systemen, ist.

2.3 Entwurfsmetaphern in der Softwareentwicklung

Ein Leitbild wird konkretisiert und technisch wie fachlich umgesetzt durch geeignete *Entwurfsmetaphern*. Diese sollen Entwicklern und Benutzern gleichermaßen helfen, eine bildliche Vorstellung von der Benutzung des Softwaresystems zu entwickeln. Eine solche Metapher sollte dabei von der Analyse bis hin zur technischen Realisierung tragfähig bleiben, sie muß sowohl eine fachliche als auch eine technische Interpretation ermöglichen. In diesem Sinne wird der Begriff Entwurfsmetapher bei Gryczan folgendermaßen definiert:

> »Eine **Entwurfsmetapher** ist eine bildhafte Vorstellung, die ein Leitbild fachlich und konstruktiv konkretisiert. Eine Entwurfsmetapher hat sowohl eine fachliche als auch eine technische Interpretation. Die fachliche Interpretation ist die Interpretation in der Um-

gangskategorie. Die technische Interpretation ist die Interpretation in der Maschinenkategorie.[4]« ([Gry95], S. 100)

2.3.1 Die WAM-Methode

Von Züllighoven et al. werden für die objektorientierte Entwicklung interaktiver Softwaresysteme die Metaphern von Werkzeug, Automat und Material (kurz WAM) vorgeschlagen. Danach kann bei allen Objekten des Anwendungssystems unterschieden werden, ob es sich um ein Werkzeug, ein Material oder einen Automaten handelt, wobei die spätere Benutzung des Systems so aussieht, daß der Benutzer mit Hilfe der Werkzeuge die Materialien bearbeitet. Automaten formalisieren dabei „lästige Routinetätigkeiten“, sie laufen mit einer durch den Benutzer veränderbaren Einstellung im Hintergrund ab und führen bestimmte Aktionen automatisch durch.

Diese Entwurfsmetaphern decken sich mit dem „natürlichen“ Verständnis der menschlichen Tätigkeit - Menschen sind es gewohnt, Arbeitsmittel und -gegenstände zu unterscheiden. Die Arbeitsgegenstände sind dabei die Materialien, die sie bearbeiten oder für ihre Arbeit gebrauchen. Die Bearbeitung findet mit Hilfe bestimmter Arbeitsmittel, den Werkzeugen, statt.

Dementsprechend wird der Fokus der Betrachtung im Rahmen der Analyse auch auf die *Umgangsformen* gerichtet. Es wird untersucht, *womit*, also mit welchen Gegenständen, bei der Erledigung der Arbeit in dem Anwendungsbereich umgegangen wird, und welche Umgangsformen diese Gegenstände haben. Umgang bedeutet dabei sowohl eine Veränderung der Gegenstände, als auch das Ablesen von Informationen an ihnen.

Nachdem in der Analyse die Werkzeuge und Materialien, mit denen bei der untersuchten Tätigkeit umgegangen wird, identifiziert wurden, können diese „bruchlos“ in eine Implementierung überführt werden.[5] In seiner *Arbeitsumgebung* werden dem Benutzer dann die anwendungsfachlich motiviert zusammengestellten Werkzeuge, Automaten und Materialien zur Verfügung gestellt. Die Arbeitsweise des Benutzers wird durch ein in dieser Weise realisiertes Anwendungssystem kaum verändert, er hat nach wie vor einen Satz von Arbeitsmitteln zu Händen, die er nach eigenem Ermessen für die Erledigung seiner Arbeit benutzen kann.

2.3.2 Metaphern zur Unterstützung kooperativer Arbeit

Mit den bisher vorgestellten Entwurfsmetaphern läßt sich allerdings nur die individuelle Arbeit eines einzelnen beschreiben. Zur Unterstützung kooperativer, räumlich wie zeitlich verteilter Arbeit werden

[4] Für eine Erläuterung der Begriffe *Umgangskategorie* und *Maschinenkategorie* siehe [Gry95].

noch weitere Metaphern benötigt. Gryczan et al. zeigen dazu einen möglichen Ansatz auf, wie die WAM-Entwurfsmetaphern zur Unterstützung kooperativer Arbeitsprozesse erweitert werden können.[6]

Dabei werden die für einen kooperativen Prozeß erforderlichen Informationen in Form von *Prozeßmustern* vergegenständlicht. Diese Prozeßmuster sind ein Material im Sinne der WAM-Metaphern und beschreiben die Verantwortlichkeiten von Personen oder Rollenträgern für die auszuführenden Tätigkeiten, die Abhängigkeiten zwischen den Tätigkeiten, sowie die für den Prozeß erforderlichen Dokumente.

Analog zu den zur Koordination der Kooperation ohne die Unterstützung durch ein entsprechendes Softwaresystem verwendeten Gittermappen,[7] werden die für einen Prozeß benötigten Materialien sowie das entsprechende Prozeßmuster in einer *Vorgangsmappe*, ebenfalls einem Material, zusammengefaßt und transportiert.

Jedem Benutzer stehen in seiner Arbeitsumgebung *Postkörbe* für den Austausch von Vorgangsmappen mit anderen Arbeitsumgebungen zur Verfügung. Der Benutzer kann beispielsweise eine solche Mappe, mit der Empfängeradresse versehen, in seinen Ausgangskorb legen. Die Postkörbe werden von einem *Versandautomaten* bedient, der im Hintergrund, außerhalb der Arbeitsumgebung eines einzelnen Benutzers, aktiv ist. Dieser entnimmt die Vorgangsmappe dem Ausgangskorb und transportiert sie zu der Arbeitsumgebung des Empfängers, wo sie in den entsprechenden Eingangskorb gelegt wird.

Durch ein Prozeßmuster wird zwar in gewissem Umfang eine Ablaufsteuerung vorgegeben, allerdings ist diese nur als „Leitfaden“ für den Ablauf des Prozesses zu verstehen. Da es sich bei einem Prozeßmuster ebenfalls um ein Material handelt, kann dieses auch von dem Benutzer verändert und individuellen Bedürfnissen angepaßt werden. Außerdem unterliegt es der Entscheidung des Benutzers, an wen ein teilbearbeiteter Vorgang gesendet werden soll. Eventuelle Überprüfungen von einzuhaltenden Sicherheits- oder Datenschutzbestimmungen können zwar realisiert werden, aber der Umgang mit dem System soll, im Sinne des zugrundeliegenden Leitbildes, nicht dergestalt sein, daß ein eigenständig aktiver Automat die Abwicklung der Vorgänge übernimmt und den Bearbeitern ohne deren Einflußnahme Dokumente vorlegt und nach deren Bearbeitung wieder entzieht, um sie an den nächsten Sachbearbeiter weiterzuleiten.

---

[5] Für eine Beschreibung, wie eine technische Umsetzung der WAM-Metaphern aussehen könnte, siehe u.a. [KGZ94] oder [Rie95].

[6] Vergleiche dazu etwa [Wul95] oder [Gry95].

[7] Mit *Gittermappen* sind hier solche Vorgangsmappen gemeint, wie sie beispielsweise häufig für die interne Hauspost in größeren Firmen verwendet werden. Diese haben einen gitterförmigen Aufdruck, in den Datum, Absender, Adressat und ähnliches eingetragen wird.

Ein solches Vorgehen ist bei vielen kommerziellen Workflow-Management Systemen zu finden, es steht jedoch im krassen Gegensatz zu den mit dem hier vorgestellten Leitbild vom „Arbeitsplatz für qualifizierte, menschliche Tätigkeit“ verknüpften Wertvorstellungen.

3 Stand der Diskussion

Wie bereits eingangs gesagt, herrscht noch kein breiter Konsens über die inhaltlichen Schwerpunkte der Betrachtungen, es zeichnen sich allerdings gewisse Tendenzen diesbezüglich ab. Um den Stand der Diskussion zu skizzieren, möchte ich zunächst exemplarisch einige unterschiedliche Ansätze, Methoden und kommerzieller Produkte vorstellen und diese von dem eben dargelegten, am Arbeitsbereich Softwaretechnik der Universität Hamburg vertretenen, Standpunkt aus bewerten.

3.1 August Wilhelm Scheer: „ARIS“

Eines der bekanntesten, und nach einer Untersuchung der Gartner Group[8] das kommerziell erfolgreichste Produkt in diesem Marktsektor ist „ARIS“ (= Architektur integrierter Informationssysteme) der Firma IDS Prof. Scheer GmbH[9].

3.1.1 Vorstellung der Methode

Nach ARIS läßt sich die Architektur eines Anwendungssystems in drei Komponenten unterteilen:

- Die unterschiedlichen Betrachtungs**sichten**, die bei der Beschreibung des Anwendungssystems unterschieden werden können.
- Orthogonal dazu die verschiedenen Betrachtungs**ebenen**, auf denen das Anwendungssystem betrachtet werden kann, und schließlich
- die unterschiedlichen Beschreibungs**sprachen**, also die Darstellungstechniken, mit denen die Inhalte der unterschiedlichen Betrachtungssichten und -ebenen dargestellt werden.

In ARIS werden vier Betrachtungssichten unterschieden, die sich alle aus der „Funktionsweise“ eines Unternehmens ableiten lassen sollen: Jedes Unternehmen, unabhängig von seiner Größe und seiner Geschäftsbranche, läßt sich demnach folgendermaßen beschreiben: Es treten bestimmte *Ereignisse* auf, etwa „Eingang eines Auftrages“ oder „Ausfall einer Maschine“. Solche Ereignisse führen dazu, daß bestimmte *Funktionen*, wie beispielsweise „Auftrag bearbeiten“, ausgeführt werden (→ Funktionssicht). Zur Ausführung dieser Funktionen werden bestimmte *Daten*, hier beispielsweise „Artikel- oder Kundenstammdaten“, benötigt (→ Datensicht). Die Ausführung der Funktionen wird von gewis-

[8] [CZ95]

sen *Organisationseinheiten* („Vertrieb“ oder „Einkauf“) verantwortet (→ Organisationssicht). Die vierte Betrachtungssicht in ARIS stellt die Steuerungssicht dar. Sie stellt die Verbindungen zwischen den anderen Sichten her, hier wird beispielsweise beschrieben, welche Organisationseinheit welche Funktionen zu verantworten hat oder welche Daten von einer Funktion manipuliert werden. Außerdem findet hier auch die Verknüpfung der Ereignisse mit den Funktionen statt. Die Einführung dieser Steuerungssicht ermöglicht nach Scheers Auffassung ferner

> »die Darstellung objektorientierter Zusammenhänge [...], da hier die Verbindungen der Daten mit den Funktionen (Methoden) aufgezeigt werden.« ([SJ96], S. 31)

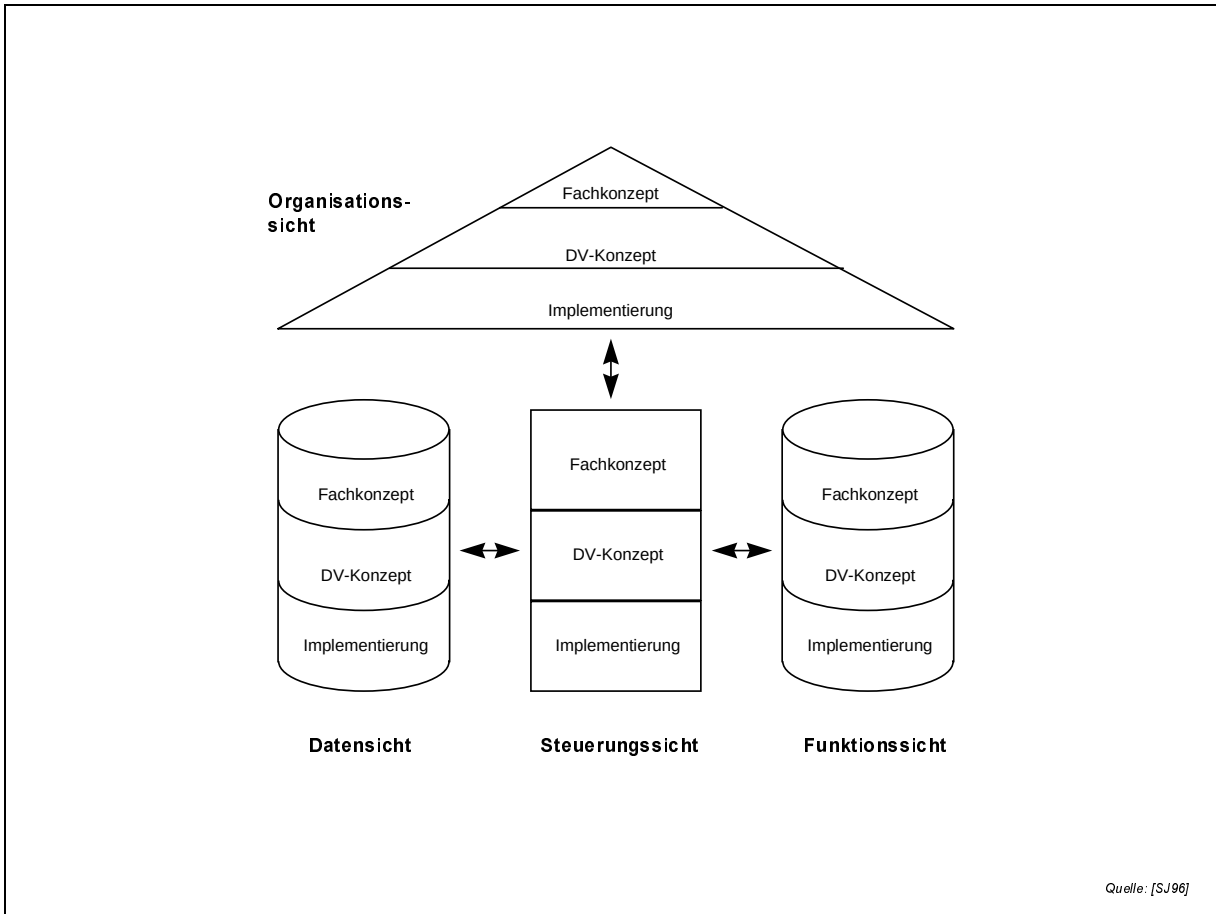

Abb. 1: Architektur integrierter Informationssysteme (ARIS) nach Scheer.

Die Unterscheidung dieser vier Beschreibungssichten soll die Komplexität des Problems vermindern, denn es soll möglich sein, sich bei der Beschreibung eines Unternehmens jeweils völlig auf eine gewählte Betrachtungssicht zu beschränken, ein Unternehmen kann also beispielsweise rein nach funktionalen oder organisatorischen Gesichtspunkten beschrieben werden.

[9] Für eine eingehendere Beschreibung der ARIS-Vorgehensweise vgl. etwa [Sch94] oder [SJ96].

Orthogonal zu den Betrachtungssichten werden in ARIS drei Betrachtungsebenen unterschieden:

- Auf Ebene des **Fachkonzept**es werden ausschließlich die betriebswirtschaftlichen Aspekte des Unternehmens betrachtet.
- Im Rahmen des **DV-Konzept**es wird das Fachkonzept abgebildet auf »DV-orientierte Strukturen«, diese werden auf Ebene der
- **Implementierung** programmiertechnisch umgesetzt.

Die Zusammenhänge zwischen den unterschiedenen Sichten und Ebenen werden in Abb. 1 dargestellt.

Als dritte Komponente der ARIS-Architektur werden die unterschiedlichen Beschreibungstechniken angesehen, mit denen die Inhalte der verschiedenen Sichten und Ebenen dargestellt werden können. Scheer bezeichnet die ARIS-Architektur in diesem Zusammenhang als Rahmenwerk, in das sich verschiedene Darstellungstechniken integrieren lassen. In jedem Fall wird allerdings dazu geraten, formale Darstellungsmittel zu verwenden:

> »Die Notwendigkeit, formalisierte Beschreibungstechniken zu verwenden, resultiert aus der Tatsache, daß eine rein textuelle Beschreibung aufgrund der Verständlichkeit sowie der Änderungsproblematik (Change-Management) nicht sinnvoll ist.« ([SJ96], S. 33)

Als wesentlich in diesem Zusammenhang wird der Integrationsaspekt der verwendeten Darstellungsmittel angesehen, daß heißt die Einheitlichkeit, mit der bestimmte Objekttypen, die in mehreren unterschiedlichen Darstellungen der verschiedenen Sichten oder Ebenen vorkommen, definiert sind.

Die verschiedenen von Scheer vorgeschlagenen Darstellungsmittel für die jeweiligen Sichten sowie ihre integrative „Vernetzung“ werden in Abb. 2 angedeutet. Im oberen Teil der Abbildung, im Bereich der Organisationssicht, ist die Aufbauorganisation des betrachteten Unternehmens in Form eines Organigramms dargestellt. Es beschreibt die Organisationseinheiten, also die Aufgabenträger, und ihre Verknüpfungen, also Weisungsbefugnis und Berichtswege. Zur Darstellung der Datensicht wird das von Chen vorgestellte, im Bereich der Datenmodellierung zum Standard gewordene, Entity-Relationship-Modell[10] verwendet. Im rechten Teil der Abbildung, der Funktionssicht, werden die Funktionen des Unternehmens in einem Hierarchiediagramm dargestellt. Dabei können komplexe Funktionen über mehrere Hierarchieebenen in Teilfunktionen zerlegt werden. In der Steuerungssicht werden diese getrennt entwickelten Modelle des Unternehmens durch ereignisgesteuerte Prozeßketten (EPK) integrativ verknüpft.

[10] Siehe dazu etwa [Che76].

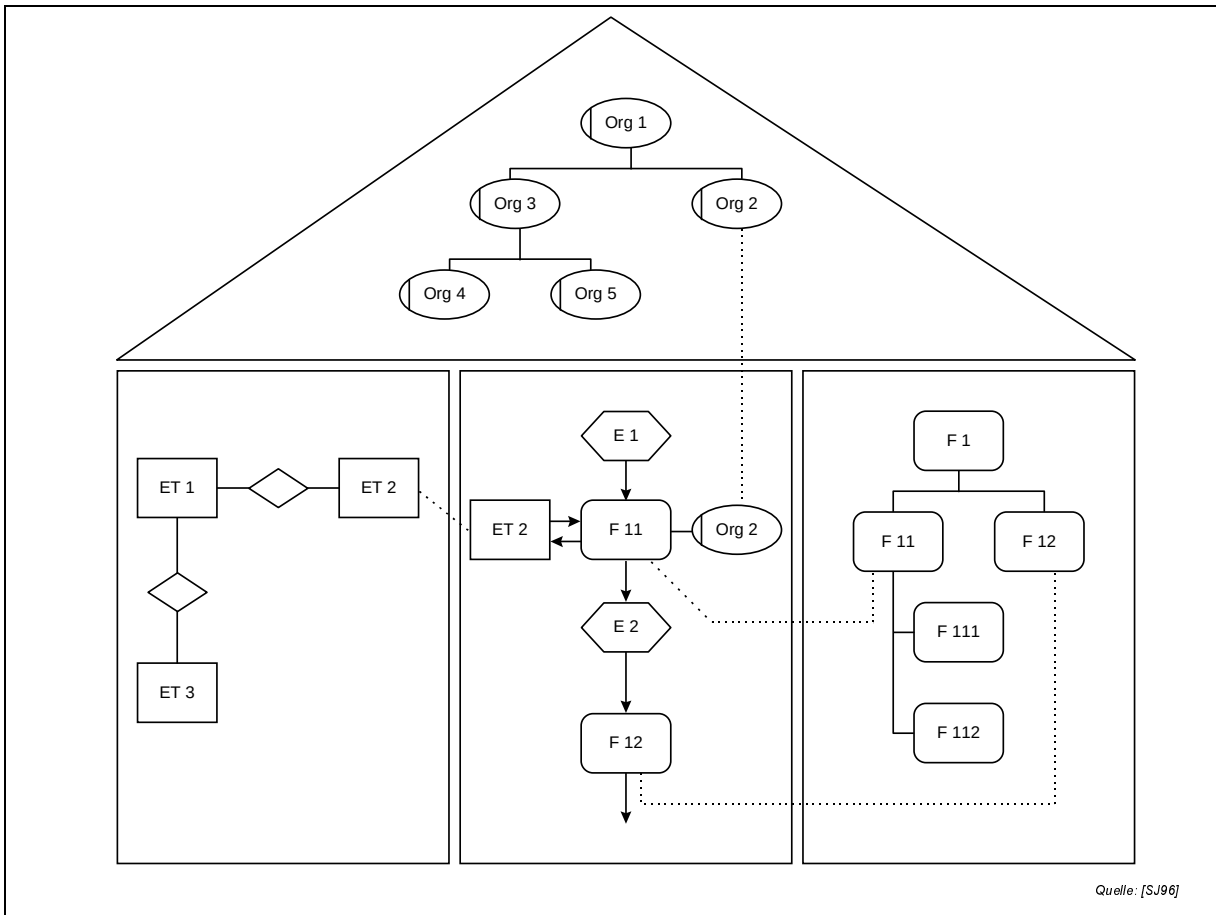

Abb. 2: Integrative Beschreibungstechniken in ARIS.

Solche Prozeßketten werden als zentrales Darstellungsmittel zur Visualisierung der Geschäftsprozesse in der Steuerungssicht in ARIS vorgeschlagen. Eine EPK besteht im wesentlichen aus zwei Objekttypen, den Ereignissen und den Funktionen, wobei letztere, je nach Detaillierungsgrad, wiederum als Prozesse angesehen werden können. Die Ereignisse bewirken Zustandsänderungen bei den Datenobjekten und können sowohl Auslöser als auch Ergebnis einer Funktionsausführung sein. Um abbilden zu können, daß Prozesse nicht immer sequentiell ablaufen müssen, werden zusätzliche „Verknüpfungsoperatoren“ (UND, (exklusives) ODER) als Darstellungsmittel hinzugenommen. Als weitere Bestandteile einer EPK können noch andere, „externe“ Objekte, etwa die betroffenen Datenobjekte oder die die Funktionen verantwortenden Organisationsbereiche, aufgenommen werden. Diese sollen dann im Sinne der geforderten Integrität der verwendeten Techniken entsprechend den Darstellungsmitteln der anderen Sichten (Datensicht, Organisationssicht) symbolisiert werden.

In diesem Zusammenhang werden bei der Vorgehensweise nach dem ARIS-Konzept objektorientierte Mechanismen integriert, um Vorteile, die der Objektorientierung zugeschrieben werden, etwa eine erhöhte Wiederverwendbarkeit des Fachkonzeptes, nutzen zu können. Allerdings sieht Scheer die gängigen Methoden der objektorientierten Modellierung nicht als geeignet an, um damit gemeinsam

mit einem Vertreter einer Fachabteilung, für den ein Anwendungssystem erstellt werden soll, ein Fachkonzept zu erstellen. Er stellt in diesem Zusammenhang fest,

> »..., daß die objektorientierte Sicht für den Anwender (Fachbereich), für den die Software letztendlich entwickelt wird, in der Regel nur sehr schwer verständlich ist. Da jedoch der Anwender an der Erstellung des Fachkonzeptes in starkem Maße beteiligt ist, kann dies ein nicht zu unterschätzendes Problem darstellen. Was fast allen objektorientierten Modellierungskonzepten fehlt, sind Konstrukte zur Beschreibung der von dem zu entwikkelnden Softwaresystem zu unterstützenden Geschäftsprozesse und der Aufbauorganisation.« ([SJ96], S. 38)

Um dieses Manko zu beseitigen, sollen objektorientierte Darstellungstechniken in die EPKs integriert werden, um so auch objektorientierte Techniken auf betriebswirtschaftliche Fragestellungen übertragen zu können.

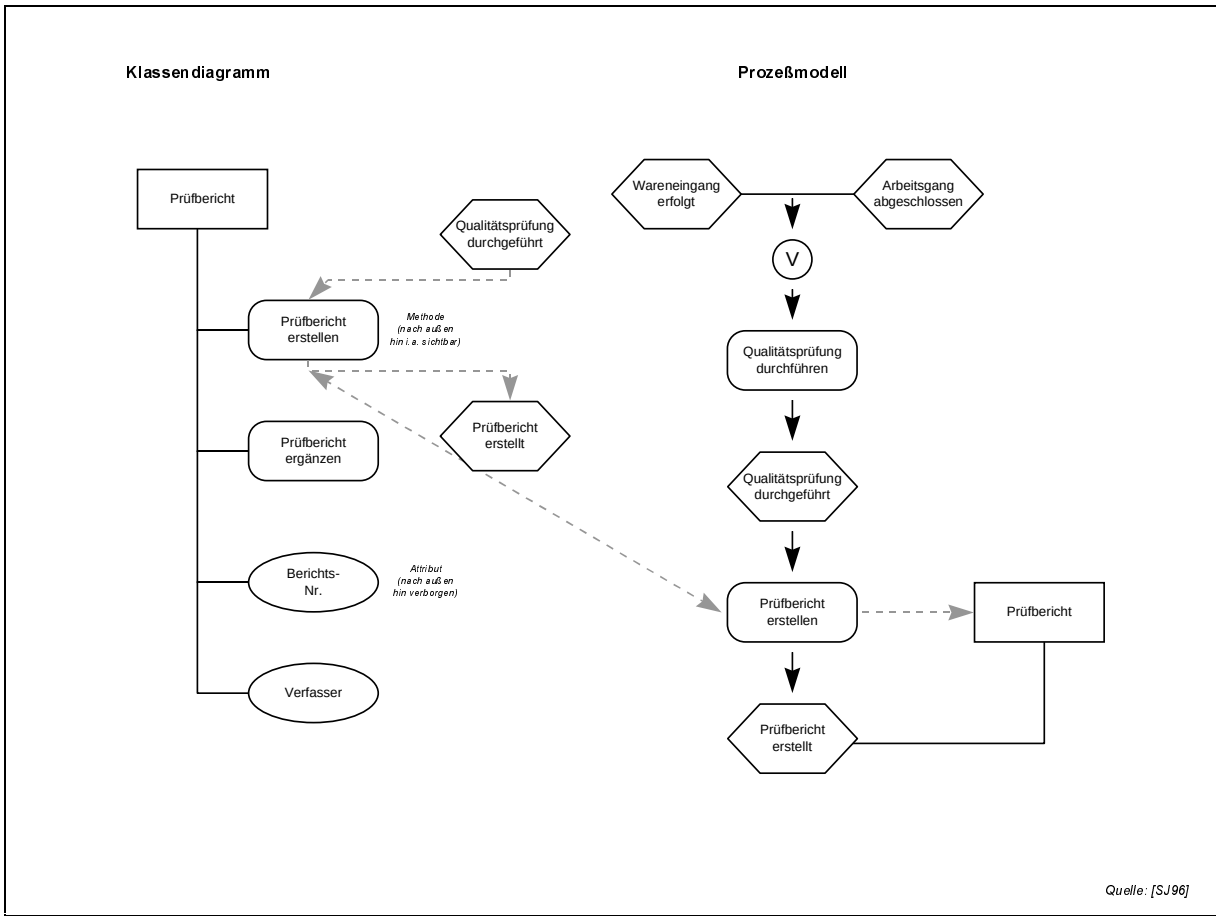

Abb. 3: Integration objektorientierter Modellierung in EPK.

Abbildung Abb. 6 veranschaulicht diese Integration. Im linken Bereich der Abbildung ist eine Klasse `Prüfbericht` dargestellt, die mehrere Methoden (`Prüfbericht erstellen`, `Prüfbericht ergänzen`) und Attribute (`Berichts-Nr.`, `Verfasser`) besitzt. Dieses Diagramm ist um die

eine Methode auslösenden (`Qualitätsprüfung durchgeführt`) beziehungsweise die von einer Methode erzeugten Ereignisse (`Prüfbericht erstellt`) erweitert. Dabei entsprechen die die Ereignisse darstellenden Symbole den Ereignissymbolen aus der EPK und die Symbole der Methoden der Klasse den Funktionssymbolen in der EPK beziehungsweise in der Funktionssicht. Im rechten Teil der Abbildung ist das Prozeßmodell eines Geschäftsprozesses `Qualitätsprüfung` in Form einer EPK dargestellt. Hier finden sich die genannten Ereignisse und Funktionen wieder, die Beziehungen zu dem Klassendiagramm sind durch entsprechende gestrichelte Pfeile gekennzeichnet.

3.1.2 Bewertung

Scheer stellt in [Sch94] explizit heraus, daß die ARIS-Methode einen objektorientierten Ansatz bei der Modellierung unterstützen soll:

> »Die ARIS-Konzeption ist ausdrücklich so entwickelt worden, daß sie den objektorientierten Modellierungsansatz umfaßt.« ([Sch94], S. 57)

Nach der Einteilung objektorientierter Analyse- und Entwurfsansätze nach Monarchi und Puhr[11] kann die ARIS-Methode aber allenfalls als *kombinierter Ansatz* angesehen werden, also als ein Ansatz, bei dem versucht wird, herkömmliche Sichtweisen mit objektorientierten Techniken zu vereinen.

Einer der grundlegenden Gedanken in der Vorgehensweise nach ARIS ist die explizite Trennung von Daten und Operationen bei der Modellierung, was sich in der Unterscheidung der vier Betrachtungssichten in ARIS ausdrückt.

Diese Trennung stellt zunächst einmal das genaue „Gegenteil“ der Objektorientierung im eigentlichen Sinne dar, denn ein Objekt soll gerade, sowohl in seiner fachlichen als auch in seiner (Software-) technischen Interpretation, eine zusammenfassende *Kapsel* um die Daten und die Operationen des beschriebenen Objektes darstellen, also einen abstrakten Datentypen. Die Daten repräsentieren den (inneren) Zustand des Objektes. Dieser soll von außen nur durch Funktionen abgefragt beziehungsweise durch Prozeduren verändert werden können. Dabei soll es für den Benutzer des Objektes unerheblich sein, wie dessen Zustand intern repräsentiert wird, die Implementation der Daten soll, ganz im Sinne des Geheimnisprinzips, vor der Außenwelt verborgen werden. Dieses weithin anerkannte Verständnis des Objektbegriffs spiegelt sich auch in der Definition des Objektbegriffs von Peter Wegner wieder:

[11] Siehe dazu [MP92].

> »Objects are collections of operations that share a state. The operations determine the messages (calls) to which the object can respond, while the shared state is hidden from the outside world and is accessible only to the object's operations.« ([Weg90], S. 8)

Diese grundlegende Eigenschaft eines Objektes im Sinne der Objektorientierung wird von Scheer bei seinem Ansatz völlig mißachtet. Die Daten und Funktionen der Objekte werden gewollt getrennt voneinander modelliert, die spätere Zusammenführung in der Steuerungssicht soll dann die Objektorientierung wiederherstellen.

Auch die Auffassung Scheers, eine objektorientierte Sichtweise sei für einen Anwender aus dem entsprechenden Fachbereich nur schwer verständlich, deckt sich nicht mit dem am Arbeitsbereich Softwaretechnik vertretenen Standpunkt. Nach der hier vertretenen Ansicht entspricht es dem „natürlichen" menschlichen Verständnis, die eigene Arbeit als den Umgang mit Objekten zu sehen. Der Mensch bedient sich bei der Erledigung seiner Aufgaben einer Reihen von Gegenständen, die er in irgendeiner Form verändert oder an denen er irgendwelche Informationen abließt. Werden diese Gegenstände mit Hilfe der von Züllighoven et al. vorgeschlagenen Entwurfsmetaphern von Werkzeug und Material in Objekte eines Anwendungssystems überführt, so daß die spätere Benutzung des Systems nach wie vor so aussieht, daß der Anwender Gegenstände, jetzt nämlich die Werkzeuge und Materialien des Anwendungssystems, benutzt, um seine Aufgabe zu erledigen, stellt die objektorientierte Sichtweise sehr wohl eine intuitiv leicht zu verstehende Sichtweise dar.

Vorausgesetzt, man erkennt der ARIS-Methode überhaupt eine gewisse Objektorientierung an, liegt dieser jedoch allenfalls das wenig plastische Leitbild der „Objektorientierung selbst" zugrunde, wie es Züllighoven et al. nennen. Dabei werden

> »...alle Objekte des Systems als aktive und miteinander kommunizierende Elemente eines Systems betrachtet [...]. Dieses Leitbild paßt gut in eine virtuelle Welt künstlicher, intelligenter „Artefakte" [...]. Es kann aber nur mit Mühe in den von uns betrachteten Anwendungsbereichen wiedergefunden werden.« ([KGZ94], S. 23)

Objektorientiertes Entwerfen bietet zudem die Möglichkeit eines bruchlosen Übergangs aus der fachlichen Welt des Anwendungsbereiches in die technische Welt der Implementation. Fachliche Begriffshierarchien, wie sie dem Verständnis der Experten des Anwendungsbereiches entsprechen, können unmittelbar, in Form entsprechender Vererbungshierarchien, in den technischen Entwurf übernommen werden.

Legt man das Leitbild vom „gut ausgestatteten Arbeitsplatz für qualifizierte menschliche Tätigkeit" mit der von Gryczan et al. vorgestellten Erweiterung um Prozeßmuster und Vorgangsmappen zugrunde, so erweist sich auch Scheers Kritik, bei einer objektorientierten Modellierung ließen sich die von

dem zu entwickelnden Softwaresystem zu unterstützenden Geschäftsprozesse nicht beschreiben, als nicht haltbar.

Auch Scheers Standpunkt bezüglich der zu verwendenden Darstellungsmittel ist konträr zu den hier vertretenen Ansichten. Scheer ist, wie oben bereits gesagt, der Auffassung, daß die Verwendung rein textueller Beschreibungen auch wegen ihrer schlechteren Verständlichkeit nicht sinnvoll sei, formalisiertere Darstellungstechniken seien in jedem Fall zu bevorzugen.

Nach der mit STEPS vorgelegten Sichtweise auf Softwareentwicklung wird eine Partizipation der zukünftigen Benutzer des Systems bei der Entwicklung gefordert, was auch im Einklang mit Scheers Auffassung steht. Allerdings wird von Züllighoven et al. darauf hingewiesen, daß es der Qualität der Ergebnisse von Analyse und Entwurf empfindlich schaden kann, wenn hierbei formalisierte Darstellungsmittel verwendet werden. Sie sprechen in diesem Zusammenhang von dem *Modellmonopol*, was zum Ausdruck bringen soll, daß die Anwender durch solche Darstellungstechniken gezwungen werden, sich in der Sprach- und Denkwelt der Entwickler zu bewegen, die ihnen nicht vertraut ist. Züllighoven et al. merken dazu an:

> »Anwender sind bereits bei ihrem Versuch, solche Ausdrucksmittel zu verstehen, von den Entwicklern abhängig. [...] Entsprechend werden die Chancen der Anwender, ihre eigenen Vorstellungen zur Systementwicklung einzubringen, durch softwaretechnisch motivierte Methoden minimiert. [...] Aber es fällt den Anwendern nicht nur schwer, ihre Vorstellungen in der Diskussion über Entwicklungsdokumente und über die Art des zu entwickelnden DV-Systems durchzusetzen. Sie haben zudem weniger Möglichkeiten, fehlerhafte oder verzerrte Darstellungen von Anwendungssituationen zu erkennen.« ([KGZ94], S. 94)

Stark formalisierte Darstellungstechniken haben sicherlich ihre Berechtigung, etwa die Verwendung von Petrinetzen für den formalen Nachweis von Korrektheit oder Verklemmungsfreiheit eines Algorithmusses. Solche Darstellungstechniken sind aus den oben genannten Gründen allerdings viel zu formal und intuitiv schwer verständlich, um damit im Rahmen der Analyse eines Anwendungsbereiches oder des Entwurfes eines Anwendungssystems Dokumente zu erstellen, die mit den Anwendern diskutiert werden sollen. Textuelle Beschreibungen, etwa in Form von Szenarios oder Systemvisionen, werden von Züllighoven et al. für diesen Zweck als wesentlich geeignetere Darstellungsformen angesehen. Über eine geringe Formalisierung der Entwicklungsdokumente, wie sie von Scheer explizit kritisiert wird, sagen Züllighoven et al.:

> »[...] - gerade das sehen wir als eine besondere Stärke dieses Dokumenttyps. Da die Einsicht in Anwendungssituationen im Vordergrund steht, können wir in Szenarios das „epische Moment“ von Schriftsprache nutzen, d.h. die Fähigkeit eines gut geschriebenen

> Textes, in uns lebendige Szenen und Vorstellungen auszulösen. Unklarheiten, Widersprüchlichkeiten, aber auch die Zusammenhänge einer geschilderten Handlung können uns in einem Prosatext sehr viel plastischer werden als etwa in einer Entscheidungstabelle.« ([KGZ94], S. 98)

Aus Sicht des am Arbeitsbereich Softwaretechnik vertretenen Standpunktes läßt sich die Kritik an der Vorgehensweise nach ARIS im wesentlichen in folgenden Punkten zusammenfassen:

- Es findet „nebeneinander“ eine getrennte Modellierung der Daten, der Operationen, der Ablauf- beziehungsweise der Aufbauorganisation des betrachteten Unternehmens statt. Es wird zwar versucht, durch einheitliche Verwendung der Darstellungsmittel der einzelnen Modelle eine gewisse Integrität in diesen Ansatz zu bringen, ein einheitliches, alle genannten Aspekte der Modellierung gleichermaßen abdeckendes Modell wird dadurch allerdings nicht erreicht.

- Die ARIS-Methode stellt einen technikzentrierten Ansatz dar, der nicht mit den Wertvorstellungen des oben vorgestellten Leitbildes von qualifizierter menschlicher Tätigkeit vereinbar ist. Der Benutzer wird in seiner Entscheidung, wie er seine Arbeit erledigen möchte, durch die in der Steuerungssicht vorgegebene Ablaufsteuerung stark eingeschränkt.

- Eine Partizipation der Benutzer an dem Entwicklungsprozeß, wie sie in STEPS gefordert wird, wird auch von Scheer als für unbedingt erforderlich angesehen. Allerdings wird die Einflußnahme des zukünftigen Benutzers auf die Entwicklung des Softwaresystems durch die formalisierten Darstellungsmittel der ARIS-Methode in oben beschriebener Weise stark reduziert.

3.2 Claudia Kohl: Objektorientierte Unternehmensmodellierung

Als weiteres Beispiel einer Sichtweise auf den hier betrachteten Themenbereich möchte ich nun den Ansatz vorstellen, wie er von Claudia Kohl et al. von der Universität Klagenfurt vertreten wird.[12]

3.2.1 Vorstellung der Methode

Im Gegensatz zu Scheer sieht auch Kohl einen objektorientierten Ansatz zur Unternehmensmodellierung als einen natürlicheren an, der keineswegs Aspekte der Ablauf- und der Aufbauorganisation eines Unternehmens vernachlässigen muß. Sie hält eine ganze Reihe gängiger Methoden für geeignet, Unternehmen objektorientiert zu modellieren:

[12] Für eine zusammenfassende Betrachtung vergleiche etwa [Koh96].

> »In der Literatur sind eine Vielzahl unterschiedlicher Methoden vertreten [RBP+91] [CY91] [Jac92] [Boo94]. Diese objekt-orientierten Analysemethoden formalisieren die kommunizierende Objektpopulation des betreffenden Systems in Form eines *konzeptuellen Schemas*, das sowohl statische als auch dynamische Aspekte des Systems beschreibt. Als Vorteile dieses Ansatzes werden hauptsächlich die Konzepte der Integration von Struktur und Verhalten, der Vererbung, der Abstraktion, der Kapselung und die Spezifikation komplexer Objekte genannt.
>
> Auch ein Unternehmen kann als System von kommunizierenden Objekten gesehen werden. Objektorientierte Analysemethoden bieten sich so auch für die Modellierung von konzeptuellen Schemata von Unternehmen an. [...] Die Abbildung der organisatorischen Aufbaustruktur und der dort bearbeiteten Geschäftsprozesse auf Objekte, die in bestimmter Weise miteinander kommunizieren, ist die natürlichste Art, die betriebliche Realität zu beschreiben.« ([Koh96], S.64 ff.)

Kohl bevorzugt nicht nur eine objektorientierte Modellierungsmethodik, sie kritisiert auch ganz explizit die Modellierung mit Hilfe der ereignisgesteuerten Prozeßketten, wie sie in ARIS vorgeschlagen wird, sowie die Untergliederung des Unternehmensmodells in die vier Sichten Daten-, Funktions-, Organisations- und Steuerungssicht. Sie zählt eine Reihe von Kritikpunkten auf, die ihrer Meinung nach durch das Vorgehensmodell von ARIS nicht ausreichend erfaßt werden, etwa die Wiederverwendung bereits modellierter Prozeßketten, die Integration heterogener Informationssysteme in das Modell, die Existenz nicht oder nur schlecht strukturierbarer Geschäftsprozesse, das Abweichen von tayloristischen Aufbaustrukturen und Funktionsteilungen, die Modellierung von Dokumentenflüssen, oder eine Komplexitätsreduktion durch eine schrittweise „Top-Down“ Zerlegung bei der Modellierung der EPKs. Bei einem objektorientierten Ansatz der Unternehmensmodellierung hingegen lassen sich solche Fälle adäquat darstellen.

Im Gegensatz zu Scheer unterscheidet Kohl zwei Ansätze der objektorientierten Modellierung, nämlich einen *struktur-* und einen *verhaltensorientierten*. Beim strukturorientierten Ansatz wird das System als Menge miteinander kommunizierender Objekte betrachtet, wobei die Objekte selbst „wissen“, was ihre Aufgabe im Rahmen der Geschäftsprozesse ist. Dementsprechend wird der Ablauf eines Geschäftsprozesses nicht explizit modelliert, sondern ergibt sich implizit aus der Kommunikationsstruktur der beteiligten Objekte, die Kontrollinformationen zu dem Geschäftsprozeß sind über die Objekte verteilt. Daraus ergibt sich das Problem, daß das solchermaßen erstellte Unternehmensmodell sehr inflexibel ist gegenüber Änderungen, ferner daß eine Wiederverwendung der Geschäftsprozeßlogik nicht oder nur in geringem Maße möglich ist. Diese Sichtweise entspricht offensichtlich Scheers Verständnis von objektorientierter Modellierung, hier ist seine Kritik, es fehle an Konstrukten zur Beschreibung der von dem zu entwickelnden Softwaresystem zu unterstützenden Geschäftsprozesse,

sicherlich angebracht. Auf dieses Leitbild der „Objektorientierung selbst“ wurde auch weiter oben bereits eingegangen.

Bei einem verhaltensorientierten Ansatz nach Kohls Verständnis hingegen werden die Geschäftsprozesse als eigenständige Objekte modelliert. Sie sind eine Art „Controller-Objekte“, die den Ablauf der Kommunikation zwischen den am Geschäftsprozeß beteiligten Objekten steuern. Die „Nicht-Geschäftsprozeß-Objekte“ sind jetzt nur noch Serverobjetke, die ihre Methoden den als Client fungierenden Geschäftsprozeßobjekten zur Verfügung stellen. Auf diese Weise soll die explizite Modellierung und damit eine modulare Änderung und Wiederverwendung der Geschäftsprozesse auch bei einer objektorientierten Modellierung möglich sein.

In [Koh96] stellt Kohl vor, wie mit Hilfe der „Object Modeling Technique“ von Rumbaugh et al. eine objektorientierte Modellierung eines Unternehmens nach ihren Vorstellungen aussieht. Sie unterscheidet dabei drei „Arten“ von Klassen, den *Organisationseinheits-*, den *Prozeß-* und den *Entity-*Klassen.

Organisationseinheits-Klassen sind dabei Bündelungen von Aufgaben, die dem Erreichen der Unternehmensziele dienen und von verschiedenen Aufgabenträgern, menschlichen oder maschinellen, erfüllt werden können. In diesem Sinne definiert Kohl eine Organisationseinheit auch als

> »[...] eine synthetische Zusammenfassung von Aufgaben, die im Unternehmen ausgeführt werden müssen. Eine Organisationseinheit stellt die Erfüllung dieser Aufgaben in der Rolle eines „Server“ zur Verfügung.« ([Koh96], S. 72)

Kohl unterscheidet zwischen *Stellen-Organisationseinheiten* und *Tool-Organisationseinheiten*. Erstere stellen Bündel von Aufgaben dar, die an menschliche Aufgabenträger zu übertragen und nicht notwendigerweise automatisierbar sind. Mit diesen Klassen wird im *Organisationseinheits-Teilobjektmodell* die Aufbaustruktur des Unternehmens modelliert und die Ziele und Aufgaben der einzelnen Organisationseinheiten festgelegt.

Abbildung 4 zeigt ein solches Organisationseinheits-Teilobjektmodell nach Kohl. Dargestellt werden zwei Organisationseinheiten, oder *Aufgabenbündel* nach Kohl, `Kellner` und `Restaurant`. Die Organisationseinheit `Kellner` stellt in diesem Sinne eine Aggregation der Aufgaben `Karte bringen`, `Bestellung notieren`, `servieren` und `kassieren` dar. Die Organisationseinheit `Restaurant` aggregiert die Aufgaben `Speiseservice`, `Candlelightdinner` und `Zustelldienst`, wobei in dem Diagramm ferner dargestellt ist, daß `Candlelightdinner` eine Spezialisierung von `Speiseservice` ist.

Der zu einer Stellen-Organisationseinheit im Prozeßmodell beschriebene Ablauf der Aufgabenerfüllung stellt eine Erwartung an den Aufgabenträger dar, soll ihm jedoch einen gewissen Handlungsfrei-

raum lassen. *Tool-Organisationseinheiten* werden nicht in die Aufbauorganisation eines Unternehmens integriert. Sie beschreiben automatisierbare Aufgaben, die an maschinelle Aufgabenträger übertragen werden können, wobei die Aufgabenverantwortung beim menschlichen Aufgabenträger verbleibt, der sich zur Erfüllung seiner Aufgabe einer solchen Tool-Organisationseinheit bedient.

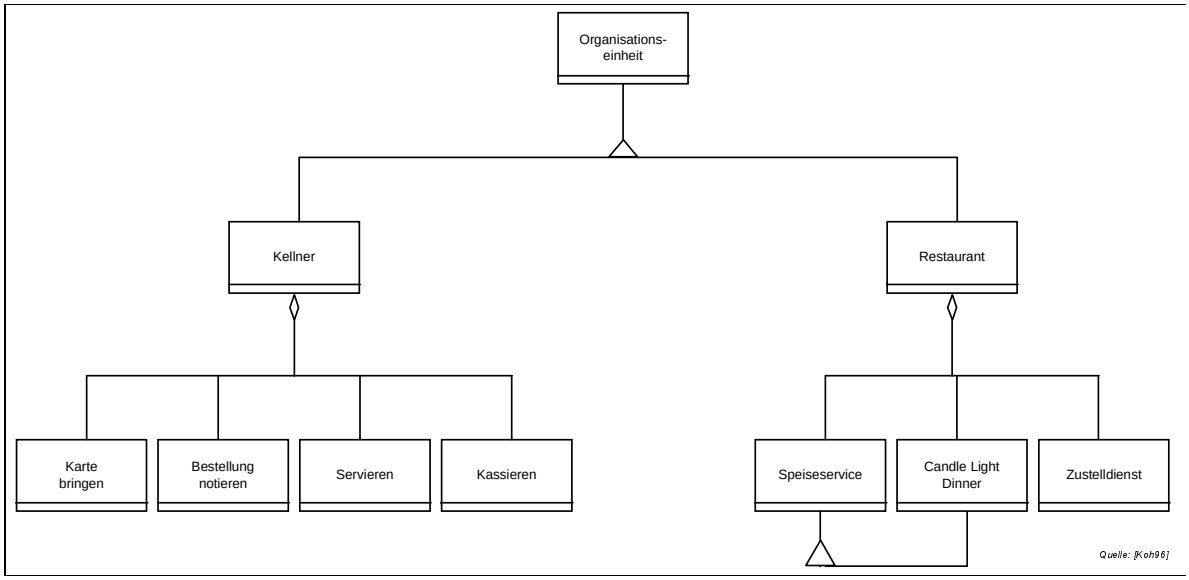

Abb. 4: Organisationseinheits-Teilobjektmodell mit Aufgabenbündeln.

Prozeß-Klassen beschreiben den Ablauf, der zur Erfüllung einer Aufgabe notwendig ist. Kohl unterscheidet bei einem Prozeß einen statischen und einen dynamischen Anteil. Der statische Anteil wird im *Prozeß-Teilobjektmodell* formalisiert, welches der Funktionssicht bei Scheer entspricht. Durch Prozeßaggregation (Zerlegung von Prozessen in Subprozesse) entstehen Prozeß-Aggregationshierarchien, in deren Blättern die durch die entsprechende Organisationseinheit zu erfüllenden Aufgaben stehen (siehe Abb. 5).

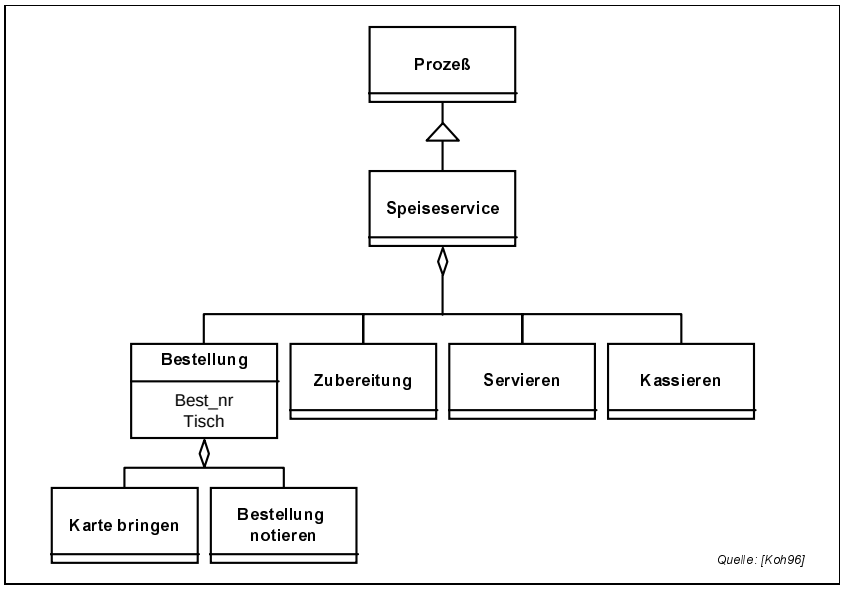

Abb. 5: Das Prozeß-Teilobjektmodell des Prozesses *Speiseservice*.

Die Dynamik des Prozesses, insbesondere die zeitlichen und logischen Abhängigkeiten, werden in dem Prozeß-Objektmodell noch nicht sichtbar, sie müssen in einem Dynamik-Modell veranschaulicht werden. Hierfür werden in OMT Zustandsdiagramme verwendet, die die möglichen Zustände einer Prozeßklasse darstellen. Diese Zustände werden durch *Events* miteinander verknüpft, wodurch sich die Operationsreihenfolge ergibt (siehe Abb. 6). Auch hier kann wieder Aggregation zur Komplexitätsreduktion eingesetzt werden.

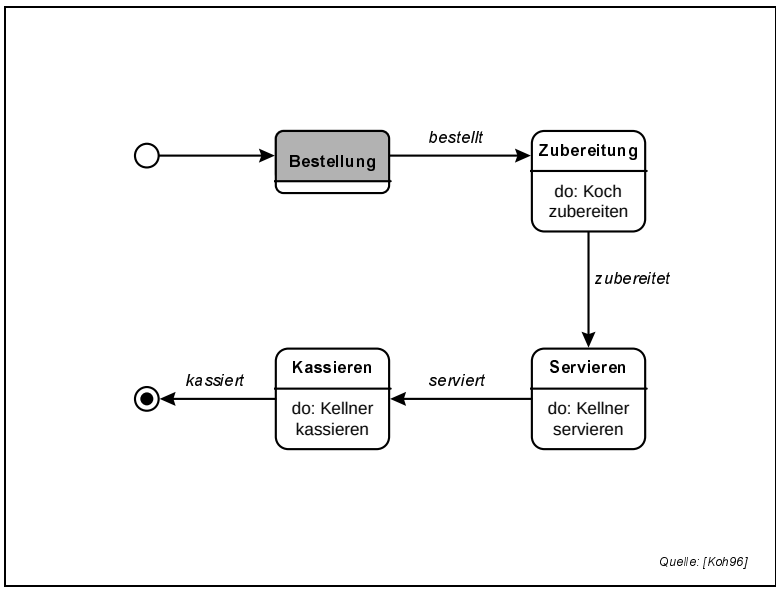

Abb. 6: Das Zustandsdiagramm der Prozeß-Klasse *Speiseservice*.

Entity-Klassen beschreiben diejenigen Objekte, welche durch die Prozesse produziert oder konsumiert werden, wobei hierbei das Augenmerk auf die, wie Kohl es nennt, *Informationen* der Objekte gerichtet ist. Es handelt sich um eine reine Datenmodellierung, das aus diesen Klassen resultierende *Entity-Teilobjektmodell* entspricht der Datensicht bei Scheer. Kohl legt allerdings Wert darauf festzustellen, daß über den Ansatz in ARIS hinaus auch eine »Aggregation von Informationen zu Dokumenten, die im Rahmen der Geschäftsprozesse zwischen den Organisationseinheiten fließen« ([Koh96], S. 76) stattfindet.

Die Abbildung 7 zeigt ein Entity-Teilobjektmodell nach Kohl, wobei hier die Darstellungsmittel für Objektmodelle aus OMT verwendet werden. Ein solches Diagramm ist folgendermaßen zu lesen: Zu jeder Rechnung existieren (möglicherweise mehrere) Bestellungen. Eine Bestellung kann mehrere Getränke, Menüs und Speisen umfassen. Umgekehrt kann ein Getränk, ein Menue oder eine Speise Bestandteil mehrerer Bestellungen sein. Die Verknüpfung zwischen Speise und Menue stellt dar, daß ein Menue eine Aggregation mehrerer Speisen ist.

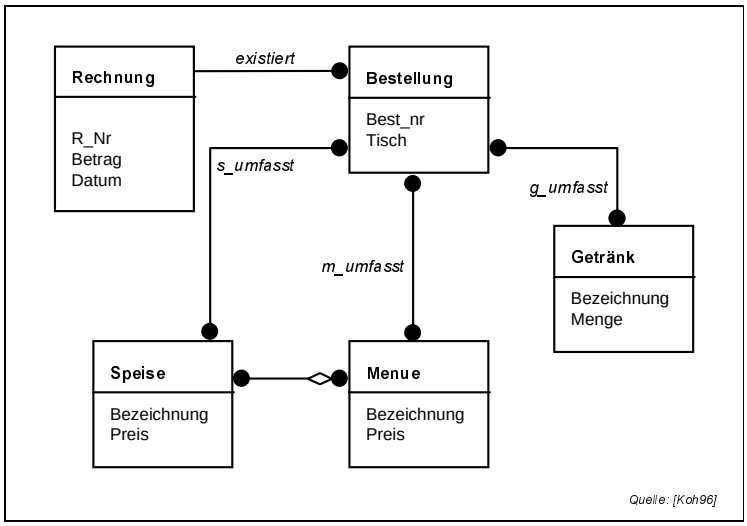

Abb. 7: Das Entity-Teilobjektmodell stellt die Datensicht dar.

3.2.2 Bewertung

Auf der einen Seite übt Kohl massive Kritik an Scheers Vorgehen, etwa an der Modellierung der Prozesse über ereignisgesteuerte Prozeßketten oder der Unterscheidung der vier Sichten Daten-, Funktions-, Organisations- und Steuerungssicht. Auf der anderen Seite unterscheidet sich ihr eigener Ansatz meiner Meinung nach wenig von dem in ARIS vorgenommenen. Dieses gilt zumindest für Kohls Versuch, eine objektorientierte Unternehmensmodellierung mit Hilfe der Object Modeling Technique von Rumbaugh vorzunehmen. Dabei ist sicherlich in Betracht zu ziehen, daß es schon stark in Frage zu stellen ist, inwieweit die OMT-Methode überhaupt als objektorientierte Methode zu bezeichnen ist. Nach der Einteilung von Monarchi und Puhr[13] ist auch dieser Ansatz ebenfalls den *kombinierten Ansätzen* zuzuzählen.

Durch die Unterscheidung der drei „Arten“ von Klassen, den Organisationseinheits-, den Prozeß- und den Entitiy-Klassen, wie sie von Kohl vorgenommen wird, und der anschließenden Bildung der drei entsprechenden Teilobjektmodelle, findet eine ähnliche Trennung von „Sichten“ statt, wie bei der ARIS-Methode.

Das Organisationseinheits-Teilobjektmodell entspricht dabei in etwa der Organisationssicht in ARIS. Lediglich die Erweiterung der in diesem Modell dargestellten Aufbauorganisation um die Aufgabenbündel findet sich bei ARIS in der Form nicht wieder. Sie ist allerdings vergleichbar mit der integrativen Darstellung in der Steuerungssicht von ARIS, wo in den EPKs zu den enthaltenen Funktionen die verantwortlichen Organisationseinheiten dargestellt werden (vgl. Abb. 2, Seite 13).

[13] Siehe dazu [MP92].

Das Entity-Teilobjektmodell entspricht der Datensicht in ARIS, es wird lediglich eine etwas andere Darstellungsform verwendet. Das mit Hilfe der Prozeßobjekte erstellte Prozeß-Teilobjektmodell ist inhaltlich dem Funktionsmodell der ARIS-Methode gleichzusetzen. Es beschreibt die, wie Kohl es nennt, statischen Komponenten der Prozesse. Die die dynamischen Komponenten beschreibenden Zustandsdiagramme entsprechen in ihrem Aufbau den ereignisgesteuerten Prozeßketten, wie sie in ARIS verwendet werden, wenn man die Kantenbeschriftungen und Zustände eines solchen Zustandsdiagramms den Ereignissen und Funktionen der entsprechenden EPK gleichsetzt.

Es findet hier also gleichermaßen wie in ARIS eine getrennte Modellierung von Daten und Operationen und damit ein Bruch mit den Prinzipien der Objektorientierung „an sich“ statt. Auch hier wird wieder der Fehler gemacht, daß objektorientiertes Modellieren mit der reinen Modellierung der Daten der Objekte gleichgesetzt wird. Der „Sinn“ eines Objektes, der eben gerade in der Kapsel um die Daten und den darauf zugreifenden Operationen liegt, wird hier genauso mißachtet.

Des weiteren stellt dieser Ansatz ebenfalls einen sehr technikzentrierten dar, was sich beispielsweise auch daran zeigt, daß Kohl es für legitim hält, Aufgaben sowohl menschlichen als auch *maschinellen* Aufgabenträgern zuzuordnen. In dem von Floyd et al. vorgestellten, menschenzentrierten Ansatz STEPS hingegen sind Aufgaben zwingend mit menschlichen Aufgabenträgern verbunden. Floyd dazu:

> »Eine *Aufgabe* ist das Ziel oder die Pflicht, die ein Mensch vor sich sieht (Brockhaus Enzyklopädie). Aufgaben bedürfen der aktiven Ausgestaltung durch den Menschen; sie hängen zusammen mit „Begriffen wie Verstehen, Sinngebung, Erfolg und Motivation, aber auch mit Verantwortung und Obliegenheit“[14], die als Hintergrund für die zielgerichtete Erfüllung von Aufgaben stets gegenwärtig sind. Dies macht deutlich, daß Aufgaben *weder modelliert noch automatisiert* werden können.« ([Flo91], S. 3.7)

Ebenso wie in ARIS, werden auch in diesem Ansatz stark formalisierte Darstellungsmittel verwendet. Allerdings muß man Kohl dabei zugute halten, daß der hier vorgestellte Modellierungsversuch mit den aus der OMT-Methode entliehenen Darstellungstechniken nur ein Beispiel darstellen sollte, an dem die Anwendbarkeit eben dieser Methode zur Unternehmensmodellierung untersucht werden sollte, woraus Kohl allerdings ein durchweg positives Fazit zieht.

Unabhängig von den verwendeten Darstellungsmitteln bleibt jedoch festzuhalten, daß auch Kohl versucht, in Form der „Steuerungsobjekte“ in gewissem Umfang eine Ablaufsteuerung vorzunehmen. Die freie Entscheidung des Benutzers, in welcher Form er seine Aufgaben erledigen möchte, wird hierbei ebenso eingeschränkt wie durch die in der Steuerungssicht in ARIS festgelegte Ablaufsteuerung.

[14] Floyd, C., Keil, R.: „Softwaretechnik und Betroffenenbeteiligung“ (In: Mambrey, P., Oppermann, R. (Hrsg.): „Beteiligung von Betroffenen bei der Entwicklung von Informationssystemen“; Frankfurt, New York, 1983)

Insgesamt läßt sich sagen, daß dem hier gewählten Ansatz ebenfalls das Leitbild von der „Objektorientierung selbst“ zugrunde liegt, allerdings in leicht abgewandelter Form, die Züllighoven et al. als Leitbild „von aktiven Steuerungsobjekten und Datenobjekten“ bezeichnen.[15] Die „klassische“ Vorgangssteuerung, wie sie etwa in der Steuerungssicht der ARIS-Methode vorliegt, wird dabei in Steuerungsobjekte „verpackt“, hier den Prozeßobjekten. Diese stellen dann die „aktiven“ Komponenten des Systems dar, die auf reinen Datenobjekten, die den Datenentitäten aus der konventionellen Datenmodellierung entsprechen, arbeiten. Auch dieses Leitbild ist in der Anwendungswelt wenig verständlich, da es kein Abbild einer „natürlichen“ menschlichen Arbeitsweise darstellt. Es eignet sich somit ebensowenig als Orientierungshilfe bei der Anwendungsentwicklung wie das Leitbild der „Objektorientierung selbst“.

Abschließend ist also festzuhalten, daß sich dieser Ansatz, zumindest aus Sicht des am Arbeitsbereich Softwaretechnik vertretenen Standpunktes, nicht wesentlich von der Vorgehensweise nach ARIS unterscheidet.

3.3 Otto K. Ferstl, Elmar J. Sinz: Semantisches Objektmodell („SOM“)

Als drittes Beispiel möchte ich das „Semantische Objektmodell“ (SOM) vorstellen, welches von Ferstl und Sinz vom Lehrstuhl für Wirtschaftsinformatik der Universität Bamberg entwickelt wurde.[16]

Sinz et al. betrachten die Modellierung *betrieblicher Systeme*[17] als zentrale Aufgabe der „Wirtschaftsinformatik“. Sie zeichnen in [FS95] die historische Entwicklung dieser Disziplin mit folgenden Stufen nach:

- Die Wurzeln dieser Entwicklung sehen sie in der Funktionsmodellierung der 70er Jahre, etwa die Methode *Structured Analysis and Design Technique (SADT)*. Solche Ansätze beschränken sich im wesentlichen auf die Modellierung einzelner Anwendungssysteme.
- Die 80er Jahre waren eine „Hochzeit“ der Datenmodellierung, vor allem repräsentiert durch die *Entity-Relationship-Modellierung*. Die Datenmodellierung geht bereits über die Betrachtung einzelner Anwendungssysteme hinaus und stellt eine unternehmensweite Datensicht dar.

[15] Vergleiche [KGZ94], S. 23 f.

[16] Für eine detailliertere Vorstellung des SOM-Ansatzes siehe etwa [FS95] oder [FM95].

[17] Unter *betrieblichen Systemen* verstehen Sinz et al. sowohl einzelne Unternehmen, als auch Unternehmensverbunde oder Geschäftsbereiche von Unternehmen.

- In den 90er Jahren werden verstärkt objektorientierte Methoden diskutiert, die die Möglichkeit bieten, die bis dahin getrennt modellierten Daten- und Funktionssicht auf Informations- und Anwendungssysteme zusammenzufassen.

Als nächsten Schritt dieser Entwicklung sehen Sinz et al. es an, die Betrachtung bei der Modellierung über einzelne Anwendungssysteme hinaus auf gesamte betriebliche Systeme auszudehnen. Dabei verschiebt sich der Fokus der Modellierung zunehmend auf das Verhalten der betrachteten Systeme und damit auf die *Geschäftsprozesse*.

3.3.1 Vorstellung der Methode

SOM versteht sich in diesem Sinne als ganzheitlicher Ansatz, bei dem versucht wird, über die Geschäftsprozesse eine vollständig objekt- und transaktionsorientierte Modellierung des gesamten betrieblichen Systems vorzunehmen, welches als System interagierender betrieblicher Objekte aufgefaßt wird.

Sinz et al. bemerken:

> »In der Literatur werden Geschäftsprozesse vielfach auf den ereignisgesteuerten Ablauf betrieblicher Aufgabendurchführungen in Form von Vorgängen reduziert. Geschäftsprozesse stellen damit eine verhaltensorientierte Ergänzung zu den herkömmlichen strukturorientierten Funktionsabgrenzungen dar.« ([FS95], S. 213)

In diesem Sinne legen Sinz et al. Wert darauf festzustellen, daß dem SOM-Ansatz »ein wesentlich umfassenderes Verständnis von Geschäftsprozessen zugrunde [liegt], bei dem die Modellierung von Struktur und Verhalten integriert wird.« ([FS95], S. 214) Diese Integration von Struktur und Verhalten wird im SOM-Ansatz durch eine objektorientierte Modellierung von Geschäftsprozessen umgesetzt.

Grundlage der Modellierung betrieblicher Systeme nach dem SOM-Ansatz ist ein Modell der *Unternehmensarchitektur*, bei der drei Modellebenen unterschieden werden (siehe Abb. 8, linke Abbildung).

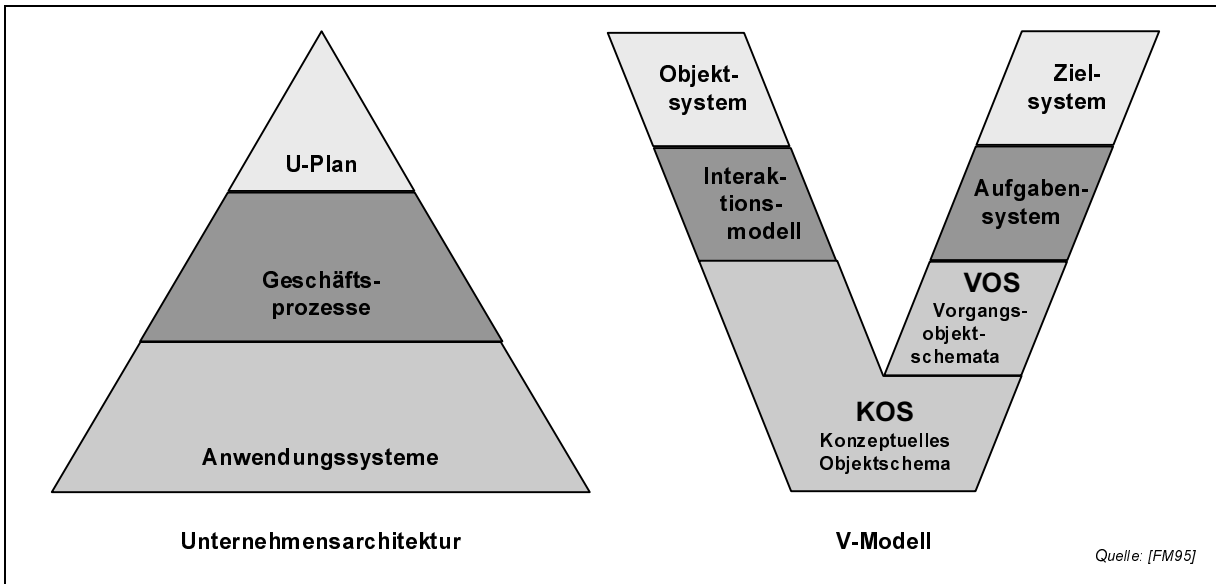

Abb. 8: Das SOM-Rahmenkonzept.

Der *Unternehmensplan (U-Plan)* stellt die Sicht von außen auf das Unternehmen dar und ist das Ergebnis der betriebswirtschaftlichen Unternehmensplanung. Hier werden die Unternehmensziele, die Unternehmens- und Marktstrategien sowie die Wertschöpfungsketten festgelegt. An dieser Stelle wird auch eine Abgrenzung vorgenommen zwischen der zu modellierenden Unternehmung, auch als *Diskurswelt* bezeichnet, und dem für die Unternehmung relevanten Teil der *Umwelt*, welcher ebenfalls, soweit erforderlich, modelliert wird.

Die *Geschäftsprozesse* stellen Lösungsverfahren für die Umsetzung des Unternehmensplanes dar, bilden also ein Modell der Innensicht des betrieblichen Systems. Geschäftsprozesse sind durch Leistungsbeziehungen miteinander verbunden und dienen der Erfüllung der im Unternehmensplan festgelegten Sach- und Formalziele. Sie können über mehrere Stufen verfeinert werden. Die Konsistenz und Gültigkeit dieser Verfeinerung kann dabei über ein „integriertes Metamodell“ (siehe Abb. 11, Seite 30) sowie einen dazugehörigen Satz von formalen „Produktionsregeln“, beides Bestandteil des SOM-Ansatzes, überprüft werden.

Die *Anwendungssysteme* als dritte Modellebene der Unternehmensarchitektur stellen im SOM-Ansatz, neben dem Personal des Unternehmens, einen Teil der Ressourcen zur Durchführung der Geschäftsprozesse dar. Ist das Geschäftsprozeßmodell hinreichend detailliert verfeinert, soll es als fachliche Spezifikation der Anwendungssysteme dienen können.

Die unterschiedlichen Ebenen der Unternehmensarchitektur werden jeweils in einem struktur- und einem verhaltensorientierten Modell dargestellt, die Beziehungen und Abhängigkeiten zwischen diesen einzelnen Modellen sind im *V-Modell* des SOM-Ansatzes beschrieben (Abb. 8, rechte Abbildung). Im linken „Schenkel“ des V-Modells sind die strukturorientierten Modelle aufgeführt, im rechten „Schenkel“ die verhaltensorientierten. Die Abstände zwischen den beiden Schenkeln des

V-Modells »symbolisieren die Freiheitsgrade hinsichtlich der Gestaltung der korrespondierenden Modellsichten der jeweiligen Ebene. Die Freiheitsgrade nehmen von oben nach unten ab.« ([FS95], S. 213)

Die oberste Ebene des V-Modells umfaßt die Modelle zur Darstellung des Unternehmensplans. Auf der strukturorientierten Seite steht das *Objektsystem*, in welchem auch die Abgrenzung zwischen den betrieblichen Objekten der Diskurs- und der Umwelt stattfindet. Die Festlegung der Sach- und Formalziele sowie der Strategien des Unternehmens stellen das verhaltensorientierte Modell *Zielsystem* dar.

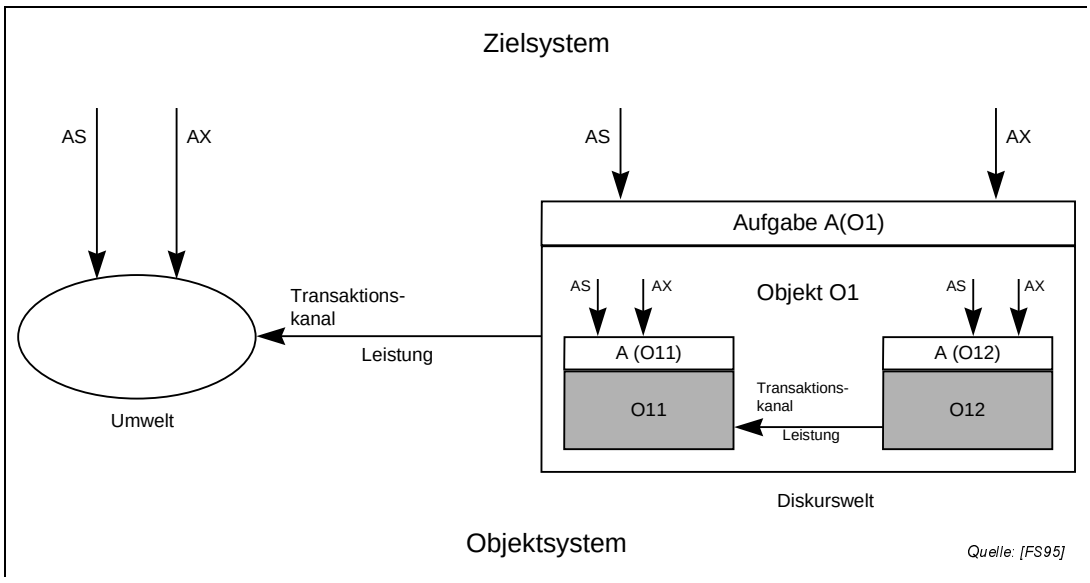

Abb. 9: Objektorientierte Modellierung im SOM-Ansatz.

Nach dem SOM-Ansatz setzt sich ein Geschäftsprozeß aus einem betrieblichen Objekt und den mit diesem verknüpften Transaktionen zusammen. Dabei können drei Sichten auf Geschäftsprozesse unterschieden werden:

- Die *Leistungssicht* stellt die Außensicht auf Geschäftsprozesse im Sinne einer Client-Server Struktur dar. Ein Geschäftsprozeß erbringt eine bestimmte Leistung und übergibt diese an den beauftragenden Geschäftsprozeß. Dazu benötigt er unter Umständen die Leistung anderer Geschäftsprozesse. Der Austausch der Leistungspakete erfolgt über Durchführungstransaktionen (siehe unten).

- Die *Ablaufsicht* zeigt, daß bei der Durchführung eines Geschäftsprozesses ereignisgesteuert Aufgaben betrieblichen Objekten zugeordnet werden. Die betrieblichen Objekte dienen damit der Erreichung bestimmter Ziele, die ihnen aus dem Zielsystem vorgegeben werden (siehe Abb. 9, AS für Aufgabensachziel bzw. AX für Aufgabenformalziel).

- Die Betrachtung der Koordination der an einem Geschäftsprozeß im Rahmen der Erstellung und Lieferung von Leistungen beteiligten betrieblichen Objekte nennen Sinz et al. die *Lenkungssicht*. Die zu erreichenden Ziele werden den Objekten ebenfalls in Form von Transaktionen, den Zieltransaktionen, vorgegeben.

Als mögliche Formen der Kooperation zwischen den betrieblichen Objekten werden in SOM die Strukturen *Regelung* für hierarchische Koordination und *Verhandlung* für nichthierarchische Koordination unterschieden (siehe Abb. 10).

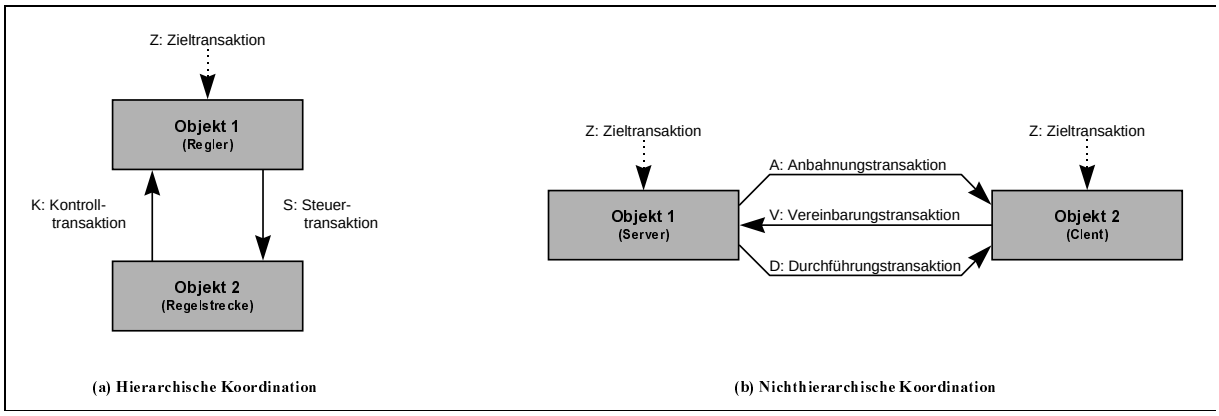

Abb. 10: Kooperationsformen zwischen betrieblichen Objekten nach dem SOM-Ansatz.

Mittels der Zieltransaktionen (Z-Transaktionen) werden dem Objekt Ziele vorgegeben, die es zu berücksichtigen hat. Kooperiert es zum Erreichen dieser Ziele mit anderen Objekten, kann dieses entweder in Form eines Regelkreises erfolgen, wobei das untergeordneten Objekt durch Steuertransaktionen (S-Transaktionen) gesteuert wird (siehe Abb. 10 (a)). Im Falle einer gleichberechtigten Beziehung zwischen den kooperierenden Objekten werden die „Details“ der Kooperation in Form von Anbahnungs- und Vereinbarungstransaktionen (A- und V-Transaktionen) zwischen den Objekten ausgehandelt, bevor die geforderte Leistung in Form einer Durchführungstransaktion (D-Transaktion) von dem Serverobjekt erbracht wird (siehe Abb. 10 (b)).

Über die Darstellungsmittel auf dieser Ebene der Modellbildung sagen Sinz et al.:

> »Für die Modellierung von Objektsystem und Zielsystem stehen häufig nur informale Darstellungsformen zur Verfügung. Die Überprüfung von Konsistenz und Vollständigkeit der Modellsichten ist in diesem Fall auf das Verständnis des Modellierers angewiesen.« ([FS95], S. 213)[18]

[18] Die Grafik in Abb. 9 ist in diesem Sinne auch nicht als Beispiel der Darstellungstechnik auf dieser Modellebene zu verstehen. Sie dient vielmehr der Erklärung der Sichtweise, wie sie im SOM-Ansatz vertreten wird.

Auf der nächsten Ebene des V-Modells stehen die Modelle zur Beschreibung der Geschäftsprozesse der Unternehmensarchitektur. Beide Sichten, struktur- und verhaltensorientiert, werden graphisch in Form von Diagrammen repräsentiert, wobei die methodische Grundlage der Geschäftsprozeßmodellierung ein „integriertes Metamodell“ ist (siehe Abb. 11).

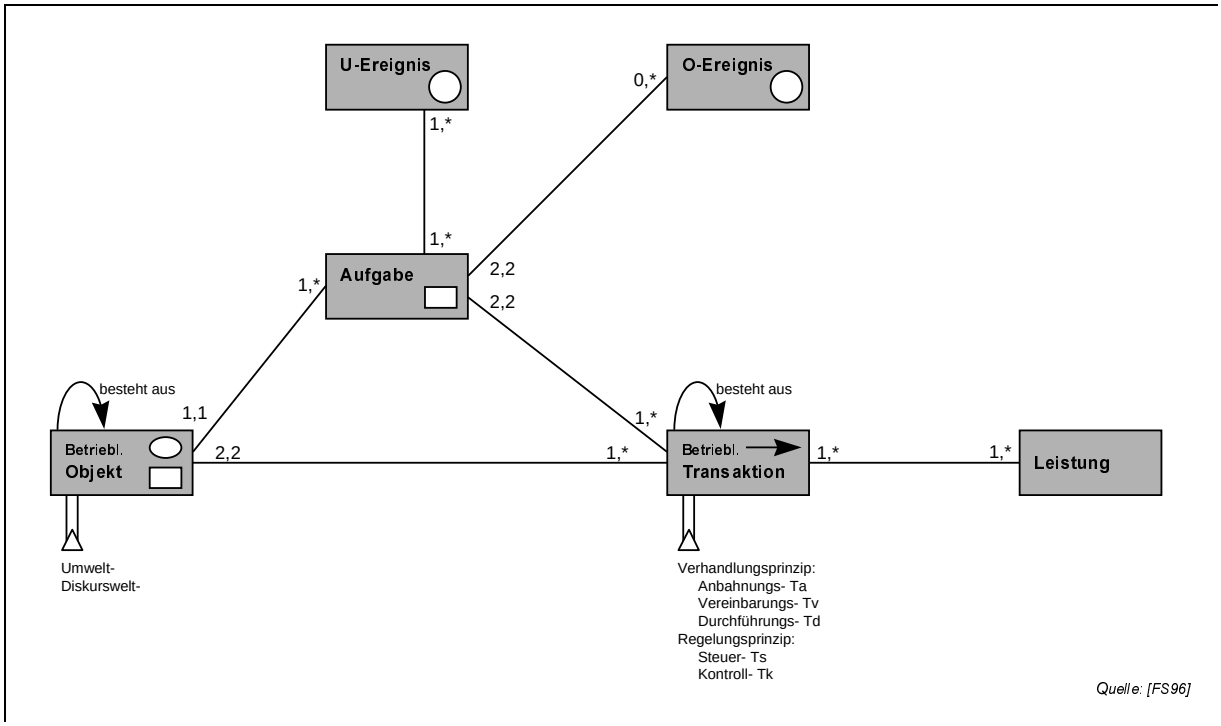

Abb. 11: Integriertes Metamodell der Geschäftsprozeßmodellierung nach SOM.

Dieses Metamodell zeigt die für die Geschäftsprozeßmodellierung zur Verfügung stehenden Modellkomponenten, die für ihre Darstellung zu verwendenden Symbole und die möglichen Spezialisierungen. Beispielsweise gibt es zwei Spezialisierungen betrieblicher Objekte, nämlich der Diskurswelt, symbolisiert durch Rechtecke, und der Umwelt, symbolisiert durch Ellipsen. Gültige Beziehungen zwischen Objekten verschiedenen Typs sind im Metamodell durch Kanten mit Angabe von Kardinalitäten dargestellt. Ein betriebliches Objekt steht beispielsweise mit beliebig vielen, mindestens aber mit einer Transaktion in Beziehung; andersherum steht eine Transaktion immer mit genau zwei betrieblichen Objekten in Beziehung.

Durch die Pfeile an den Metaobjekten wird eine mögliche Aggregation dargestellt; so kann ein betriebliches Objekt wieder in mehrere betriebliche Objekte zerlegt werden.

Um die Konsistenz und Vollständigkeit von Zerlegungen überprüfen zu können, ist ein Satz von Produktionen Bestandteil des SOM-Ansatzes, die angeben, welche gültigen Zerlegungsschritte es für Objekt- oder Transaktionszerlegungen gibt.

1. Produktionen zur Objektzerlegung

O	::= {O’, O", T_S (O’, O"), [T_K (O", O’)]}
O	::= {O’, O", [T (O’, O")]}
O’, O"	::= O

2. Produktionen zur Transaktionszerlegung

T (O, O’)	::= [[T_A (O, O’) seq] T_V (O’, O) seq] T_D (O, O’)
T_X	::= T_X {seq T_X}$^+$ | T_X {par T_X}$^+$
	(für X = A, V, D, S, K)
T_A | T_V | T_D	::= T
T_S | T_K	::= T

Abb. 12: Produktionen zur Objekt- und Transaktionszerlegung im SOM-Ansatz.

In Abbildung 12 sind diese Produktionen in Backus-Naur-Form angegeben. Eine Zerlegung eines betrieblichen Objektes in mehrere Objekte, die nach dem Regelungsprinzip wie in Abb. 10 (a) dargestellt kooperieren, wird beispielsweise durch die Produktion

O ::= {O’, O", T_S (O’, O"), [T_K (O", O’)]}

beschrieben. Sie definiert, daß ein betriebliches Objekt O in ein Reglerobjekt O' und ein Regelstrekkenobjekt O" zerlegt werden kann, zwischen denen es dann zumindest eine Steuerungstransaktion T_S von O' nach O" geben muß (T_S (O', O")). Eine Kontrolltransaktion T_K von O" nach O' ist bei dieser Zerlegung optional ([T_K (O", O')]).

Eine nichthierarchische Kooperation wie in Abb. 10 (b) dargestellt wird durch die Produktion

T (O, O') ::= [[T_A (O, O') seq] T_V (O', O) seq] T_D (O, O')

beschrieben. Sie besagt, daß eine Transaktion zwischen zwei Objekten O' und O" in eine Anbahnungstransaktion T_A, eine Vereinbarungstransaktion T_V und eine Durchführungstransaktion T_D in sequentieller Reihenfolge zerlegt werden kann. T_A und T_V sind dabei optional. Da sie dem „sich kennenlernen“ der beiden Kooperationspartner dienen, können sie wegfallen, wenn es sich um eine Zusammenarbeit zu bereits etablierten Bedingungen handelt.

Strukturorientiert werden die Geschäftsprozesse durch das *Interaktionsmodell* beschrieben, welches aus einer Folge von Interaktionsschemata besteht (siehe Abb. 13). Die Darstellungssymbole, Transaktionstypen und gültigen Verfeinerungsschritte lassen sich aus dem Metamodell beziehungsweise den Produktionen ableiten und überprüfen.

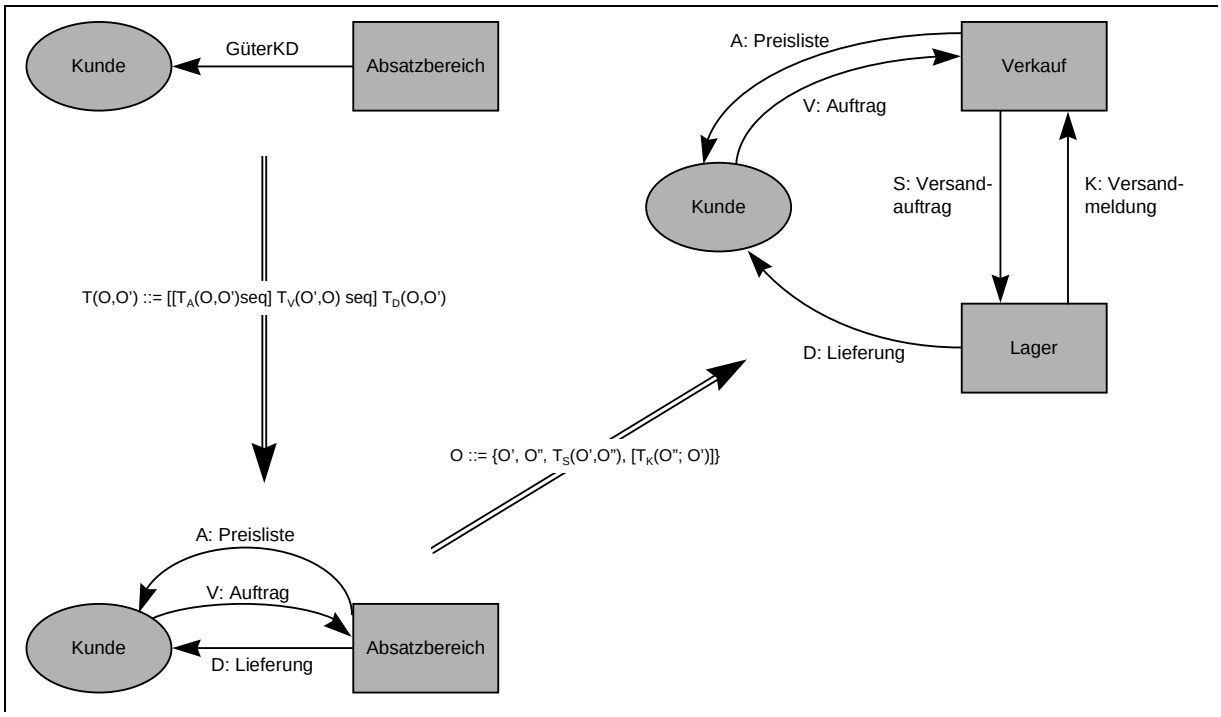

Abb. 13: Interaktionsmodelle eines Geschäftsprozesses *Güterdistribution* mit gültigen Zerlegungen.

Die verhaltensorientierte Darstellung der Geschäftsprozesse erfolgt in Form von Vorgangs-Ereignis-Schemata, welche in ihrem Detaillierungsgrad mit den jeweiligen Interaktionsschemata korrespondieren.

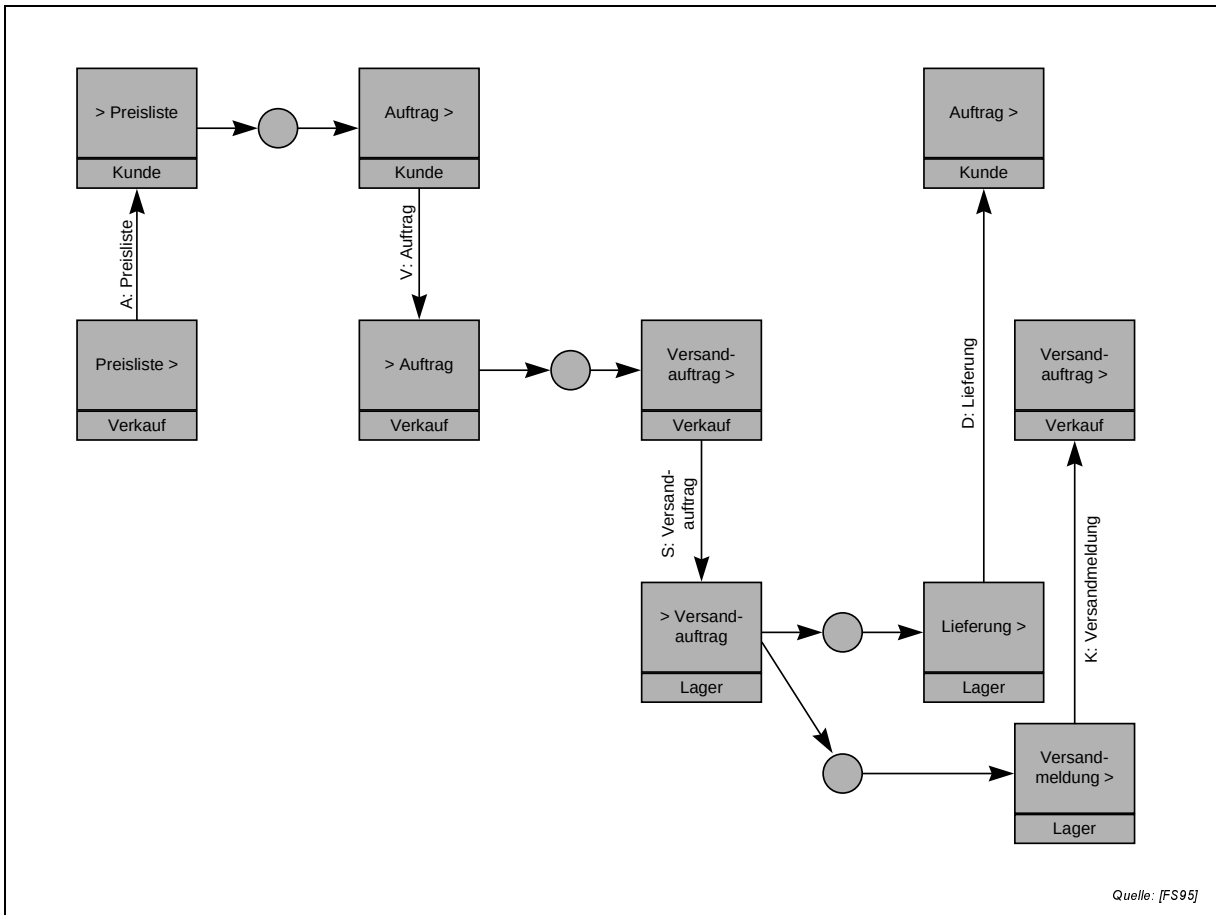

Abb. 14: Vorgangs-Ereignis-Schema zu dem Geschäftsprozeß *Güterdistribution* aus Abb. 13.

Auf der untersten Ebene der Unternehmensarchitektur, der Ebene der Anwendungssysteme, stehen das *konzeptuelle Objektschema (KOS)* und das *Vorgangsobjektschema (VOS)* (siehe Abb. 8, Seite 27).

Das konzeptuelle Objektschema bildet den „Knotenpunkt“ zwischen den struktur- und den verhaltensorientierten Modellen. Es beschreibt die konzeptuellen Objekttypen der Anwendungssysteme sowie deren Beziehungen. Dabei werden die Beziehungsformen Aggregation, Spezialisierung / Generalisierung sowie Interaktion unterschieden. Zur Darstellung eignen sich dementsprechend alle Techniken, bei denen sich diese Beziehungsarten unterschiedlich darstellen lassen. Dabei wird nach SOM Wert auf eine formale Darstellungsart gelegt, damit sich die angegebenen Beziehungen anhand des Metamodells auf Gültigkeit überprüfen lassen.[19]

Das Zusammenwirken der im konzeptuellen Objektschema beschriebenen Objekttypen bei der Durchführung betrieblicher Aufgaben wird in Vorgangsobjektschemata beschrieben, welche aus einem oder mehreren *Vorgangsobjekttypen (VOT)* bestehen. Vorgangsobjekttypen tauschen untereinander Nachrichten aus, die mit den vorgangsauslösenden beziehungsweise den von den Vorgängen produzierten

[19] In [FS91] stellen Sinz et al. eine eigene, geeignete Darstellungstechnik vor, auf die ich aber nicht näher eingehe, da sie mir für die konzeptionelle Vorstellung des SOM-Ansatzes nicht relevant erscheint.

Ereignissen, wie sie in den Interaktionsschemata angegeben sind, korrespondieren. Jeder Vorgangsobjekttyp beschreibt für eine oder mehrere verwandte Aufgaben, in welcher Weise die an der Aufgabendurchführung beteiligten Objekte zusammenwirken und welche Aktionenfolgen bei Empfang bestimmter Nachrichten ausgeführt werden, was das interne Verhalten der Vorgangsobjekte darstellt. Auf diese Weise werden auch Vorgänge im SOM-Ansatz objektorientiert modelliert.

3.3.2 Bewertung

Ein Unterschied zu den Ansätzen von Kohl und Scheer liegt bereits in dem dem SOM-Ansatz zugrunde liegenden Objektverständnis, hier liegt eine „wirklich“ objektorientierte Sichtweise vor. Die Daten und die auf diesen arbeitenden Operationen werden hier nicht getrennt modelliert, vielmehr wird von Sinz et al. eine zusammenfassende Modellierung der Daten und Operationen, also eine Kapselung im objektorientierten Sinne, gefordert:

> »Zentrale Forderungen für die Definition betrieblicher Informationssysteme gemäß SOM-Ansatz sind: - die ausschließliche Verwendung von Objekttypen, die Daten- und Funktionssicht vereinigen, [...].« ([FS91], S. 490)

Aus Sicht der am Arbeitsbereich Softwaretechnik eingenommenen Perspektive erscheint mir jedoch eine grundlegende Sichtweise im SOM-Ansatz als fragwürdig: Sinz et al. kritisieren an anderen Ansätzen, daß diesen keine ganzheitliche Sichtweise zugrunde liege, die auch die Tätigkeit des Menschen einbeziehe:

> »Die Konzentration auf die Anwendungsentwicklung verstellt den Blick auf eine ganzheitliche Sicht von IS, die auch nicht automatisierbare Aufgaben sowie den zugehörigen Aufgabenträger, den Menschen, einschließt.« ([FS91], S. 480)

Als Konsequenz daraus sehen sie es als erforderlich an, den menschlichen Aufgabenträger in die Modellierung der Geschäftsprozesse einzubeziehen. Allerdings sehen sie diesen dann auf der selben Stufe mit automatisierbaren, maschinell durchführbaren Teilen der Geschäftsprozesse:

> »Im Gegensatz zur betriebswirtschaftlichen Literatur, die als Träger der Aufgabenerfüllung ausschließlich Personen betrachtet, unterscheidet der SOM-Ansatz zwischen personellen und maschinellen Aufgabenträgern. Dies führt zur Abgrenzung eines automatisierten Teilsystems (Objekte mit maschinell durchgeführten Aufgaben) und eines nicht automatisierten Teilsystems (Objekte mit personell durchgeführten Aufgaben) des Objektsystems.« ([FS95], S. 212)

Sinz et al. rücken damit von der in STEPS explizit vertretenen Auffassung ab, daß Aufgaben zwingend mit menschlichen Aufgabenträgern verbunden und nicht automatisierbar sind. Im Rahmen der

Geschäftsprozessmodellierung nach SOM werden die menschlichen Aufgabenträger gleichberechtigt zu automatisierbaren Tätigkeiten in die Prozesse „hineinmodelliert“, sie sind Bestandteil der Modellierung.

In der Konsequenz steht dieses auch im Widerspruch zu der reaktiven Charakteristik der Anwendungssysteme, wie sie nach Züllighoven et al. gefordert wird. Die so modellierten Systeme erfüllen, neben den Menschen, einen Teil der im Rahmen der Geschäftsprozesse anfallenden Aufgaben und stehen nicht, wie durch das Leitbild des Arbeitsplatzes für qualifizierte menschliche Tätigkeit gefordert, dem Benutzer als Werkzeuge zur Verfügung.

Ein weiterer, grundlegender Unterschied zwischen dem mit SOM vorgestellten Ansatz und der Sichtweise von Scheer oder Kohl besteht in der Betrachtungsweise eines Unternehmens:

Wie weiter oben bereits gesagt, wird in der ARIS-Methode von einem grundsätzlichen, für alle Unternehmen gültigen Modell ausgegangen. Jedes Unternehmen läßt sich danach so beschreiben, daß auf bestimmte Ereignisse hin („Auftragseingang“) bestimmte Funktionen („Auftrag bearbeiten“) ausgeführt werden.

Im Gegensatz zu ARIS, wo also eine *funktionsorientierte* Sichtweise auf ein Unternehmen eingenommen wird, wird im SOM-Ansatz eine *prozeßorientierte* Perspektive gewählt. Ferstl et al. sehen diese Herangehensweise als Konsequenz aus dem Versuch, eine zunehmende Kundenorientierung bei gleichzeitiger Gewinnmaximierung zu erreichen:

> »Die Verwendung von Geschäftsprozessen als Grundbausteine für industrielle Systeme resultiert aus dem Übergang von der bisherigen statischen, funktionsorientierten Sichtweise auf ein Unternehmen hin zu einer dynamischen, ergebnisorientierten Sicht.« ([FM95], S. 447)

Die Unterscheidung der drei Sichten (Lenkungs-, Leistungs- und Ablaufsicht) auf Geschäftsprozesse mag auf den ersten Blick den Sichten, wie sie in ARIS oder auch bei Kohl unterschieden werden, ähnlich erscheinen. Ein wesentlicher Unterschied zu diesen liegt beim SOM-Ansatz jedoch darin, daß hier auch die Aufbauorganisation der Unternehmung über die Lenkungssicht mit modelliert werden kann. Bei einem Ansatz, wie er etwa in ARIS vorgenommen wird, finden die Organisationsstrukturen zwar in die Organisationssicht Eingang, sie werden dort allerdings als „gegeben“ hingenommen. Im SOM-Ansatz dagegen wird die Koordinationsstruktur des Unternehmens nicht nur mit modelliert in dem Sinne, daß sie in ihrer bestehenden Form abgebildet wird. Vielmehr gibt es in SOM auch für diese Strukturen verschiedene Möglichkeiten, sie zu strukturieren und zu zerlegen. Diese Regeln sind gleichfalls Bestandteil des Metamodells und der Produktionen, die den SOM-Ansatz theoretisch untermauern. Anders als bei einem Vorgehen nach ARIS wird im SOM-Ansatz ein „Überdenken“ der Ausgangs-Aufbaustruktur des modellierten Unternehmens also auch methodisch unterstützt.

In [FM95] zeigen Ferstl und Mannmeusel unter anderem auf, welche Gestaltungsgrundsätze es für die Strukturen des Lenkungssystems gibt. Sie führen beispielhaft vor, wie mit Hilfe der in SOM gültigen Transaktions- und Objektzerlegungen die Lenkungsstruktur, also die Aufbaustruktur eines Unternehmens, schrittweise verfeinert werden kann. Je nachdem, welche Zerlegungsformen dabei gewählt werden, können so unterschiedliche Koordinationsformen innerhalb des Unternehmens realisiert werden, sowohl „herkömmliche“, hierarchische Strukturen, als auch flachere, dezentrale Lenkungsstrukturen, wie sie beispielsweise in Unternehmen mit „Profit-Center-Strukturen“ zu finden sind.

Als eine Möglichkeit, die eine solche ganzheitliche Sichtweise bietet, sehen Sinz et al. unter anderem die Entwicklung von *Referenzmodellen* für Koordinationsstrukturen. Der Begriff Referenzmodell ist hierbei deutlich anders zu verstehen, als er beispielsweise bei SAP/R3 verwendet wird, wo versucht wird, für jeden denkbaren Geschäftsvorfall einen allgemeingültigen Ablauf bereits „vorausgedacht“ zu haben. Hier sind vielmehr Modelle in ähnlichem Sinne gemeint, wie sie in der Softwaretechnik unter dem Begriff *Entwurfsmuster* bekannt sind. Sinz et al. sehen die Möglichkeit, für bestimmte, durch Umwelteinflüsse vorgegebene, Leistungsanforderungen „passende“ Koordinationsstrukturen in Form von Referenzmodellen vorrätig zu haben. Diese können dann beispielsweise herangezogen werden, um erforderliche Anpassungen der Aufbaustruktur an veränderte Leistungsanforderungen zu ermitteln.

Aus dem Blickwinkel der am Arbeitsbereich Softwaretechnik eingenommenen Perspektive läßt sich zusammenfassend festhalten:

- Im SOM-Ansatz wird eine „wirklich“ objektorientierte Modellierung der betrieblichen Objekte vorgenommen. Die Geschäftsprozesse werden folgerichtig als eine Menge interagierender Objekte, die sich über Nachrichten, hier den Transaktionen, austauschen, modelliert.

- Dem SOM-Ansatz liegt kein tragfähiges Leitbild zugrunde, das die menschliche Arbeit in angemessener Weise unterstützt. Vielmehr wird die Tätigkeit menschlicher Aufgabenträger im Rahmen der Geschäftsprozeßmodellierung gleichberechtigt zu maschinellen, automatisierbaren Tätigkeiten angesehen.

- Die Unterscheidung verschiedener Koordinationsstrukturen in der Lenkungssicht eröffnet interessante Möglichkeiten der Modellierung der Organisationsaufbaustruktur. Dieses stellt einen Schritt einer Erweiterung des Betrachtungshorizontes der Wirtschaftsinformatik in Richtung der Verbindung von Softwaretechnik und Organisationstheorie dar, wie er zunehmend gefordert wird.

4 Bewertungskriterien

Wie bereits in diesen wenigen Beispielen deutlich wird, lassen sich bei den Ansätzen verschiedene Bewertungskriterien oder, neutraler gesagt, Merkmale unterscheiden, anhand derer sie sich kategorisieren lassen. Dabei lassen sich wiederum zwei Gruppen von Merkmalen unterscheiden, zum einen solche, die die zugrundeliegende Technik oder Sichtweise, zum anderen solche, die den Umfang des Ansatzes charakterisieren. Zu jeder dieser Gruppen möchte ich einige Unterscheidungsmerkmale aufzeigen, die insbesondere im Hinblick auf die am Arbeitsbereich Softwaretechnik vertretene Sichtweise interessant sind.

4.1 Merkmale der zugrundeliegenden Technik oder Sichtweise

4.1.1 Objektorientierung

Ein grundsätzliches Unterscheidungsmerkmal stellt die „Objektorientiertheit“ des betreffenden Ansatzes dar. Viele Ansätze schmücken sich mit dem Begriff Objektorientierung, nicht zuletzt aus Marketinggründen. Bei genauerem Hinsehen lassen sich jedoch eine Reihe von „Reifegraden“ bei der Umsetzung objektorientierter Mechanismen unterscheiden.

So gibt es Ansätze, denen keinerlei objektorientierte Grundlage anzuerkennen ist, obwohl diese von sich behaupten, objektorientiert zu sein. Daten und Operationen werden völlig getrennt voneinander modelliert, die grundlegendste Objekteigenschaft, das Geheimnisprinzip, wird völlig außer acht gelassen.

Findet eine Kapselung von Daten und Operationen eines Objektes statt, so ist die nächste Frage, inwieweit die Wahl der Objektmethoden fachlich begründet ist. Sieht die Modellierung so aus, daß es zu jedem (Daten-) Attribut des Objektes die Operationen GET und SET gibt, so ist sicherlich trotz einer Kapselung eher von Datenmodellierung denn von Objektmodellierung zu sprechen. Eine solche Methodenwahl setzt ebenfalls das Geheimnisprinzip außer Kraft, da der Benutzer eines Objektes nicht nur dessen Schnittstelle, sondern auch seine innere Datenstruktur kennen muß und von einer Änderung derselben auch betroffen ist.

Ferner stellt sich die Frage, inwieweit Mechanismen wie Vererbung und Polymorphie zur Modellierung eingesetzt werden; die Kapselung von Daten und Operationen allein stellt noch keine Verwendung der Objektorientierung in vollem Umfang dar.

4.1.2 Funktions- / Prozeßorientierung

Ein weiteres Unterscheidungsmerkmal besteht in der Orientierung, die bei der „Verteilung“ der Funktionalität der entwickelten Anwendungssysteme zugrunde gelegt wird. Dabei kann funktions- oder prozeßorientiert vorgegangen werden.

Bei der *funktionsorientierten* Sichtweise werden Aufgaben immer weiter in Teilaufgaben zerlegt, bis diese so „einfach“ sind, daß sie problemlos zu erledigen sind. Analog zu dieser Vorgehensweise findet auch die „klassische“, hierarchische Gestaltung der Aufbauorganisation von Unternehmen statt. In solcher Weise entworfene Anwendungssysteme stellen gewissermaßen Funktionsarbeitsplatzsysteme dar, an denen einzelne Teilstücke übergeordneter Aufgaben bearbeitet werden, ohne daß deren Einbettung in den Gesamtzusammenhang ersichtlich wäre.

Im Gegensatz dazu findet bei einer *prozeßorientierten* Sichtweise, wie der Name schon sagt, die Abgrenzung der Funktionalität der Anwendungssysteme entlang der unterschiedlichen Prozesse des Unternehmens statt. Die Gestaltung der Prozesse wiederum sollten an den Erwartungen des Prozeßkunden orientiert sein. Hess et al. dazu:

> »Ein Prozeß sollte jene Aufgaben, Aufgabenträger, Sachmittel und Informationen zusammenfassen, die aufeinander abgestimmt werden müssen, um aus Sicht eines Kunden zusammenhängende Leistungen zu erbringen, z.B. von der ersten Beratung bis zur Auslieferung der bestellten Ware.« ([HBÖ95], S. 481)

Innerhalb der Prozeßorientierung lassen sich wiederum verschiedene Sichtweisen unterscheiden. Bei einer *verrichtungsorientierten* Prozeßbeschreibung wird modelliert, welche Bearbeitungsschritte in welcher Reihenfolge zu vollziehen sind. Ein solcher Ansatz wird beispielsweise im SAP R/3 System verfolgt. Wird ein Prozeß hingegen *objektorientiert* beschrieben, so werden „nur“ die erforderlichen Tätigkeiten durch die Beschreibung der Umgangsformen in Objektmethoden vorgegeben, es findet keine Festlegung der Bearbeitungsreihenfolge statt. Diese Sichtweise entspricht dem mit WAM vorgestellten Ansatz und wird beispielsweise auch von Jacobson et al. in [JEJ95] vorgeschlagen.

4.1.3 Ablaufsteuerung / Ablaufunterstützung

Die Form der Arbeitsgestaltung, die durch die Methode verursacht wird, stellt ein weiteres Bewertungskriterium dar. Aus Sicht der mit STEPS und dem Leitbild vom Arbeitsplatz für qualifizierte menschliche Tätigkeit verbundenen Wertvorstellungen ist hier von besonderem Interesse, wie stark Arbeitsschritte und -abläufe dem Benutzer von dem entwickelten Anwendungssystem vorgeschrieben werden, beziehungsweise wie frei der Benutzer in der Ausgestaltung seiner Arbeit ist. Eine Gestaltung des Anwendungssystems nach WAM stellt hier eine sehr flexible Form der Arbeitsunterstützung dar.

Ein weiterer interessanter Punkt in diesem Zusammenhang ist, ob Formen der kooperativen Arbeit durch die Methode unterstützt werden, und wenn ja, in welcher Form. Auch hierbei ist zu untersuchen, inwieweit die Kooperation vom System reglementiert wird. So kann beispielsweise im Extremfall die Kooperation nach einem vorgegebenen Schema von einem Workflow-Management System erzwungen werden, ohne daß die beteiligten Akteure Einfluß darauf nehmen können, in welcher Reihenfolge von wem welche Bearbeitungsschritte vollzogen werden müssen. Eine solche „Unterstützung“ von Kooperation steht allerdings im krassen Gegensatz zu dem oben erwähnten Leitbild; danach wäre eher eine Unterstützung der Kooperation in einer Form, wie sie etwa von Gryczan in [Gry95] vorgeschlagen wird, gefordert.

4.2 Merkmale des Ansatzumfanges

Unabhängig von der zugrundeliegenden Technik oder Sichtweise können Methoden auch nach dem Ansatzumfang kategorisiert werden. Hierbei ist zu untersuchen, was alles Gegenstand der Betrachtung eines Ansatzes oder einer Methode ist.

4.2.1 Organisationsentwicklung

Ein sehr grundlegendes Merkmal des Ansatzumfanges ist die Frage, in welchem Maße die Aufbaustruktur des Unternehmens im Rahmen des Ansatzes Beachtung findet. Dieses kann in sehr unterschiedlicher Weise erfolgen. So kann ein Ansatz sich ausschließlich auf die Gestaltung des Informationssystems beschränken und die Organisationsstruktur dabei völlig außer acht lassen.

Eine erste Form der Einbeziehung der Aufbauorganisation in die Modellierung wird beispielsweise in ARIS vorgenommen, wo den modellierten Aufgaben, Tätigkeiten oder Prozessen die verantwortenden Organisationseinheiten zugeordnet werden, wobei allerdings die Organisationsstruktur selbst als „gegebene“ hingenommen wird und ihre (Um-) Gestaltung nicht Bestandteil des Ansatzes ist.

Soll eine weitreichendere Form der Einbeziehung der Organisationsstruktur in die Betrachtung erreicht werden, so muß die Modellierung der Aufbauorganisation des Unternehmens selbst Bestandteil der Methode oder des Ansatzes sein. Diese Form der Integration von Organisationsentwicklung und Informationssystementwicklung findet in der Literatur zunehmend Beachtung.[20]

[20] Vergleiche dazu etwa [Rol97] oder [FKR+97].

4.2.2 Beachtung externer Systeme

Eine weitere Möglichkeit, Methoden der Geschäftsprozeßmodellierung und Anwendungs- und Informationssystementwicklung zu unterscheiden, liegt in dem Umfang, in welchem die an das modellierte Unternehmen angeschlossenen externen Systeme, etwa Kunden oder Lieferanten, und deren für das Unternehmen relevanten Prozesse mit modelliert werden.

Eine solche Einbeziehung kann in einer Form erfolgen, wie sie beispielsweise im SOM-Ansatz vorgenommen wird, wo lediglich die relevanten betrieblichen Objekte der externen Systeme in die Modellierung einfließen, nicht jedoch die in diesen Systemen ablaufenden Prozesse.

Eine weitere Einbeziehung externer Systeme würde allerdings auch die dort vorhandenen Prozesse im Modell abbilden. Hess et al. legen in diesem Zusammenhang sogar nahe, die eigenen Prozesse denen der externen Systeme anzupassen, um so eine höhere Kundenbefriedigung zu erreichen.

4.2.3 Methodische Unterstützung

Weitere zu betrachtende Aspekte eines Ansatzes sind Art und Umfang der methodischen Unterstützung bei der Modellierung. Erfolgt eine solche Unterstützung, ist zu untersuchen, inwieweit die mit dieser Methode verbundenen Wertvorstellungen mit der am Arbeitsbereich Softwaretechnik vertretenen Sichtweise vereinbar sind.

Grundlegend hierfür ist beispielsweise eine Partizipation des Anwenders bei Entwurf und Gestaltung der zu entwickelnden Anwendungssysteme. In diesem Zusammenhang ist auch zu untersuchen, ob die durch die Methode vorgeschlagenen oder vorgeschriebenen Dokumenttypen für eine solche Partizipation geeignet sind. Hier lassen sich sehr unterschiedliche Auffassungen feststellen, etwa die von Scheer et al. in der ARIS-Methode vertretene, daß ausschließlich formale Darstellungsmittel zu verwenden sind.

Ein weiterer Aspekt in diesem Zusammenhang ist die Frage, inwieweit der Methode ein sinnstiftendes Leitbild zugrunde gelegt und auch explizit genannt beziehungsweise beschrieben ist, welches im Entwicklungsprozeß Benutzern und Entwicklern gleichermaßen eine Orientierungshilfe sein soll. Ist dieses der Fall, sollte die Methode auch geeignete Entwurfsmetaphern und Entwurfsmuster für eine konstruktive Umsetzung bereitstellen.

Zu betrachten ist des weiteren, ob auch ein Vorgehensmodell Bestandteil des Ansatzes ist und wie dieses aussieht. In STEPS wird ein evolutionäres, zyklisches Vorgehen gefordert, in welchem pro Zyklus eine Ausbaustufe des Systems erstellt wird, deren Evaluierung Ausgangspunkt des nächsten Zyklus ist. Durch ein „wasserfallartiges“ Phasenmodell läßt sich dieser von STEPS vorgegebene (Methoden-) Rahmen nicht ausfüllen.

5 Vorstellung der EP/KID-Architektur

In dem nun folgenden Abschnitt dieser Arbeit stelle ich ein Vorgehensmodell für die Geschäftsprozeßanalyse und Anwendungs- und Informationssystementwicklung in Verbindung mit dem Produkt EP/KID[21] vor, welches ich im Rahmen dieser Diplomarbeit erstellt habe. Ziel dieses Vorgehensmodells ist es, ein methodisches Vorgehen zu beschreiben, welches in dem durch die zugrundeliegende Technik und Architektur von EP/KID vorgegebenen Rahmen zu einem möglichst „guten" objektorientierten Entwurf im Sinne der vorgestellten Bewertungskriterien und des am Arbeitsbereich Softwaretechnik vertretenen Verständnisses hiervon führt.

Da dieses Vorgehensmodell stark auf den besagten, durch EP/KID vorgegebenen Rahmen abgestimmt ist, werde ich das Produkt zunächst etwas detaillierter als die verschiedenen Ansätze in den vorigen Abschnitten dieser Arbeit vorstellen.

Um die grundlegende Architektur von EP/KID, die Implementierung eines Anwendungssystems basierend auf dieser Architektur, sowie das Vorgehensmodell und die dort vorgestellten Dokumenttypen besser darstellen zu können, verwende ich eine kleine Beispielanwendung. Als Grundlage für dieses Fallbeispiel dient dabei die „Vorstudie über ein Bibliothekssystem", die bereits mehrfach in den Übungsgruppen zu der Lehrveranstaltung „Einführung in die Softwaretechnik" von Christiane Floyd Verwendung fand.[22] Zusätzlich werde ich vereinzelt Beispiele aus konkreten KID-Projekten verwenden, sofern es mir zur Erklärung einzelner Sachverhalte sinnvoll erscheint.

Es sei noch darauf hingewiesen, daß ich alle Begriffe zunächst so verwende, wie sie auch in der KID-Literatur benutzt werden.[23] Um die explizite Belegung der verwendeten Begriffe in der „KID-Welt" zu verdeutlichen, sind diese bei ihrer Einführung *kursiv* dargestellt. Sollten mir diese Begriffe zu ungenau oder für den jeweiligen Verwendungszusammenhang nicht passend erscheinen, so werde ich an entsprechender Stelle darauf eingehen.

5.1 Was ist EP/KID?

Als Unternehmensberatung hatte die Firma Entitec mehrfach den Auftrag, für ein Handels- oder Dienstleistungsunternehmen Anwendungssysteme zu erstellen, die dessen Geschäftsprozesse unter-

[21] EP steht für „Entitec-Produkt", KID für „Konzept für Innovatives Design".

[22] Die Vorstudie befindet sich im Anhang zu dieser Arbeit.

stützen sollten. Grundlage solcher Anwendungssysteme ist in der Regel ein Geschäftsdaten- oder Geschäftsobjektmodell des Unternehmens. Mit zunehmender Erfahrung in dem Bereich der Geschäftsprozeßanalyse und -modellierung fielen den Systemanalytikern dabei zwei wesentliche Punkte auf:

- Das Aufstellen eines Geschäftsobjektmodells, welches für alle Geschäftsprozesse und alle Fachabteilungen des Unternehmens Gültigkeit haben soll, ist ein sehr schwieriges und vor allem aufwendiges Unterfangen.
- Je mehr Geschäftsobjektmodelle für Unternehmen aus dem Handels- und Dienstleistungssektor aufgestellt wurden, desto mehr fielen die strukturellen Ähnlichkeiten und Gemeinsamkeiten zwischen diesen auf.

Vor dem Hintergrund dieser Erkenntnisse wurde daher beschlossen, ein allgemeingültiges, wiederverwendbares Geschäftsobjektmodell zu entwickeln, mit dessen Hilfe sich jedes beliebige Handels- oder Dienstleistungsunternehmen modellieren lassen soll. Basis eines KID-Anwendungssystems soll eine objektorientierte Anwendungsarchitektur sein, die auch verteiltes Arbeiten auf unterschiedlichen Plattformen gestattet. Lediglich eine unternehmensspezifische Namensgebung bei den *KID-Objekten* soll erforderlich sein, um dieses generalisierte Geschäftsobjektmodell für eine Anwendungsentwicklung in einem Unternehmen einsetzen zu können.

Der Begriff *KID-Objekt* wird in diesem Zusammenhang in der KID-Literatur folgendermaßen eingeführt:

> »Die KID-Objekte bilden das fachliche Herz der KID-Architektur. Es sind Objekttypen mit allen Eigenschaften und Funktionen, die fachliche Objekte in kommerziellen Anwendungssystemen benötigen.« ([KID96c], S.21)

Da mir diese Begriffsbildung etwas ungenau erscheint, möchte ich an dieser Stelle zunächst eine genauere Einordnung des eingeführten Begriffs vornehmen. Ich übernehme dazu die Definitionen für die Begriffe Klasse und Objekt von Züllighoven et al.:

> »**Klasse:** Auf der Ebene des Entwurfs dasjenige Modellelement, durch das die Konzepte der Anwendung beschrieben werden. Auf der Ebene eines objektorientierten Programms Erzeugungsmuster für die Objekte des Systems. In typisierten objektorientierten Sprachen entspricht die Klassendefinition der Typdefinition. [...]
>
> **Objekt:** Datenkapsel, die einen veränderbaren Zustand besitzt. Der Zustand eines Objektes kann durch Aufruf von Funktionen abgefragt und durch Aufruf von Prozeduren

[23] Siehe dazu etwa [KID96a-c].

verändert werden. Die Menge der aufrufbaren Prozeduren und Funktionen wird durch die erzeugende Klasse festgelegt.« ([KGZ94], S. 204f)

Im Sinne dieser Definitionen verwende ich im weiteren die Begriffe KID-Objekt und Objekt synonym. Inwieweit es sich bei den KID-Objekten tatsächlich um Objekte im objektorientierten Sinne handelt, soll weiter unten noch diskutiert werden.

Aufbauend auf diesem generalisierten Geschäftsobjektmodell wurde EP/KID um eine *Workbench* erweitert. Sowohl die Anwendungsentwicklung als auch die spätere Pflege und Wartung des KID-Anwendungssystems geschieht aus einer integrierten Oberfläche heraus, alle erforderlichen Informationen (Klassenbeschreibungen, Steuerungsinformationen, Fachfunktionalität) werden über diese Oberfläche gepflegt. Aus diesen Informationen wird das gesamte Anwendungssystem automatisch generiert, die Anwendungsentwickler brauchen keinen Programmcode mehr von Hand schreiben. Bei der Entwicklung der KID-Architektur war weniger die Performance als vielmehr eine möglichst flexible Änderbarkeit der KID-Anwendungssysteme maßgebend.

5.1.1 Das „generalisierte“ Unternehmen

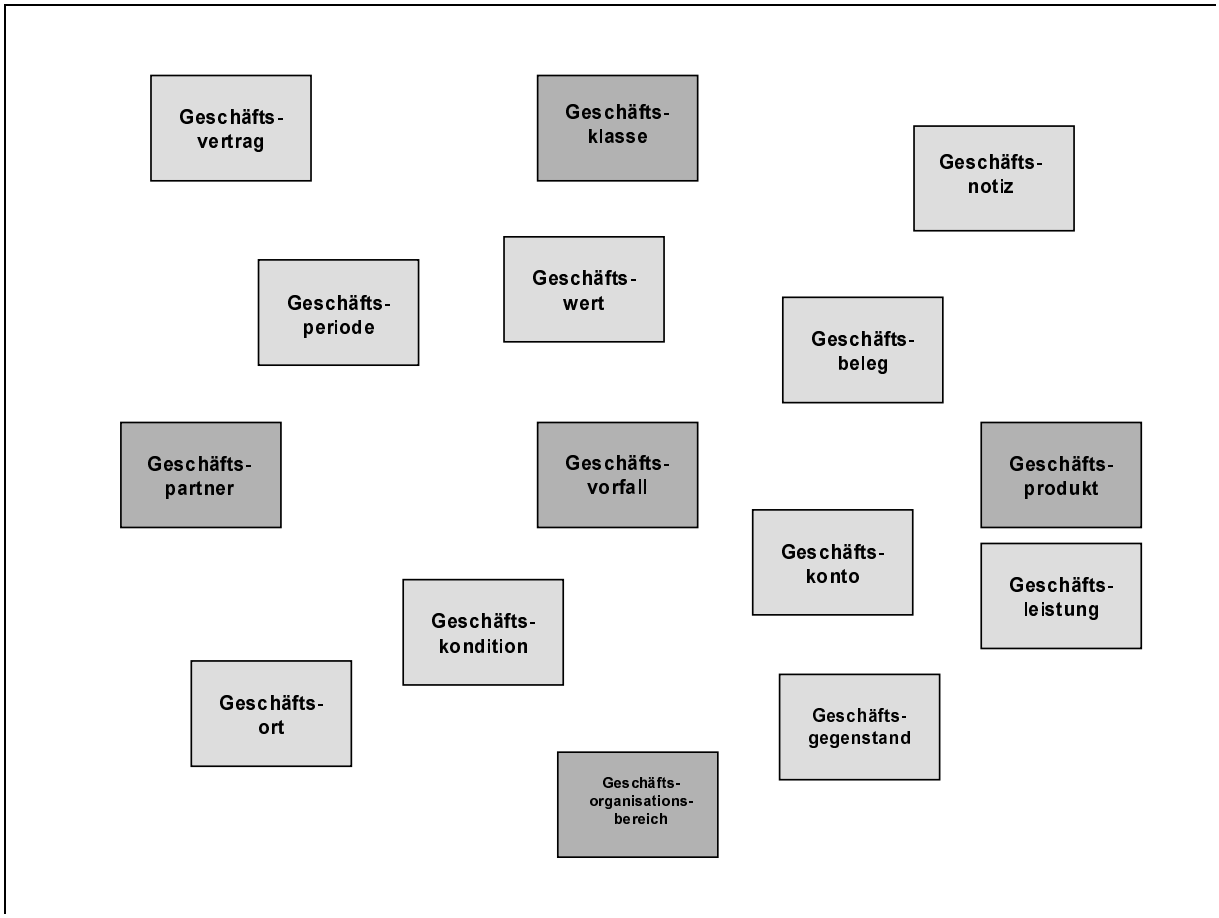

Abb. 15: Basisklassen der KID-Architektur.

Zielsetzung der KID-Architektur ist wie gesagt, ein Geschäftsobjektmodell (Abb. 15) zur Verfügung zu stellen, mit dem sich jedes Handels- oder Dienstleistungsunternehmen modellieren läßt und auf dessen Grundlage sich unternehmensweit Informations- und Anwendungssysteme entwickeln lassen:

Jedes Unternehmen erbringt im Rahmen von *Geschäftsvorfällen Geschäftsleistungen* für *Geschäftspartner*, hierüber werden *Geschäftsverträge* abgeschlossen. Eine solche Geschäftsleistung wird von einem bestimmten *Geschäftsorganisationsbereich* erbracht und stellt entweder die konkrete Lieferung eines angebotenen Produktes, oder, im Falle einer Dienstleistung, die konkrete Durchführung eines angebotenen Dienstes dar. Die angebotenen Produkte oder Dienste lassen sich dabei verallgemeinert als *Geschäftsprodukte* beschreiben.

Darüber hinaus gibt es einige Dinge, die zwar „eigenständige“ Objekte sind, die für das Unternehmen aber erst durch ihren Zusammenhang mit anderen Objekten relevant sind. So gibt es eine Reihe von *Geschäftskonten*, etwa Firmenkonten oder die Bankverbindungen der Geschäftspartner. Dazu zählen ferner alle Arten von Adressen, oder allgemein *Geschäftsorten*. Bei der Abwicklung der Geschäftsvorfälle können *Geschäftsbelege* anfallen, beispielsweise Rechnungen oder Zahlungsbescheide, und es werden *Geschäftsnotizen*, etwa Telefon- oder Gesprächsnotizen, erstellt. Falls sich ein Geschäftsvertrag außer auf ein (angebotenes) Geschäftsprodukt auch noch auf andere Dinge bezieht, beispielsweise das Auto bei einer Kfz-Versicherung, wird dieses durch einen *Geschäftsgegenstand* repräsentiert.

Die Zuordnung von *Geschäftsklassen* zu Objekten dient zum einen als Hilfskonstrukt für die Abbildung objektorientierter Mechanismen in die KID-Architektur, ich werde hierauf weiter unten noch genauer eingehen. Zum anderen wird durch die Klassifikation von Objekten die Möglichkeit geschaffen, im Rahmen des Controlling aus den Geschäftswerten Daten klassifikationsweise zu verdichten. So können beispielsweise Auswertungen zu bestimmten Klassen von Geschäftsprodukten oder -partnern erstellt werden. Sind die Geschäftsorganisationsbereiche klassifiziert, so lassen sich möglicherweise Auswertungen je Abteilung erstellen. Die Zeitachse wird dabei durch die *Geschäftsperiode* dargestellt.

5.1.2 Die KID-Architektur - technische Aspekte

Aus technischer Sicht werden zwei wesentliche Anforderungen an ein KID-Anwendungssystem gestellt:

- Es soll möglichst plattformunabhängig sein, sowohl bezogen auf die Hardware- als auch auf die Softwareumgebung. Das heißt, es muß unter verschiedenen Betriebssystemen, auf verschiedenen Rechnerarchitekturen, sowohl „stand alone“ als auch Client-Server fähig, in heterogenen Netzlandschaften, auch mit Großrechneranbindung, laufen können.

- Es soll möglichst einfach an technologische Veränderungen angepaßt werden können.

Um diesen Anforderungen gerecht werden zu können, werden KID-Anwendungssysteme modular aufgebaut. Dabei lassen sich je nach Betrachtungsschwerpunkt verschiedene Komponentenbildungen vornehmen.

5.1.2.1 Funktionale Komponentenbildung

Jedes KID-Modul läßt sich beispielsweise seiner Funktionalität entsprechend einer der fünf *Anwendungsaspekte Datenzugriff, Steuerung, Präsentation, Anwendungskern* oder *Kommunikation* zuordnen (siehe Abb. 16).

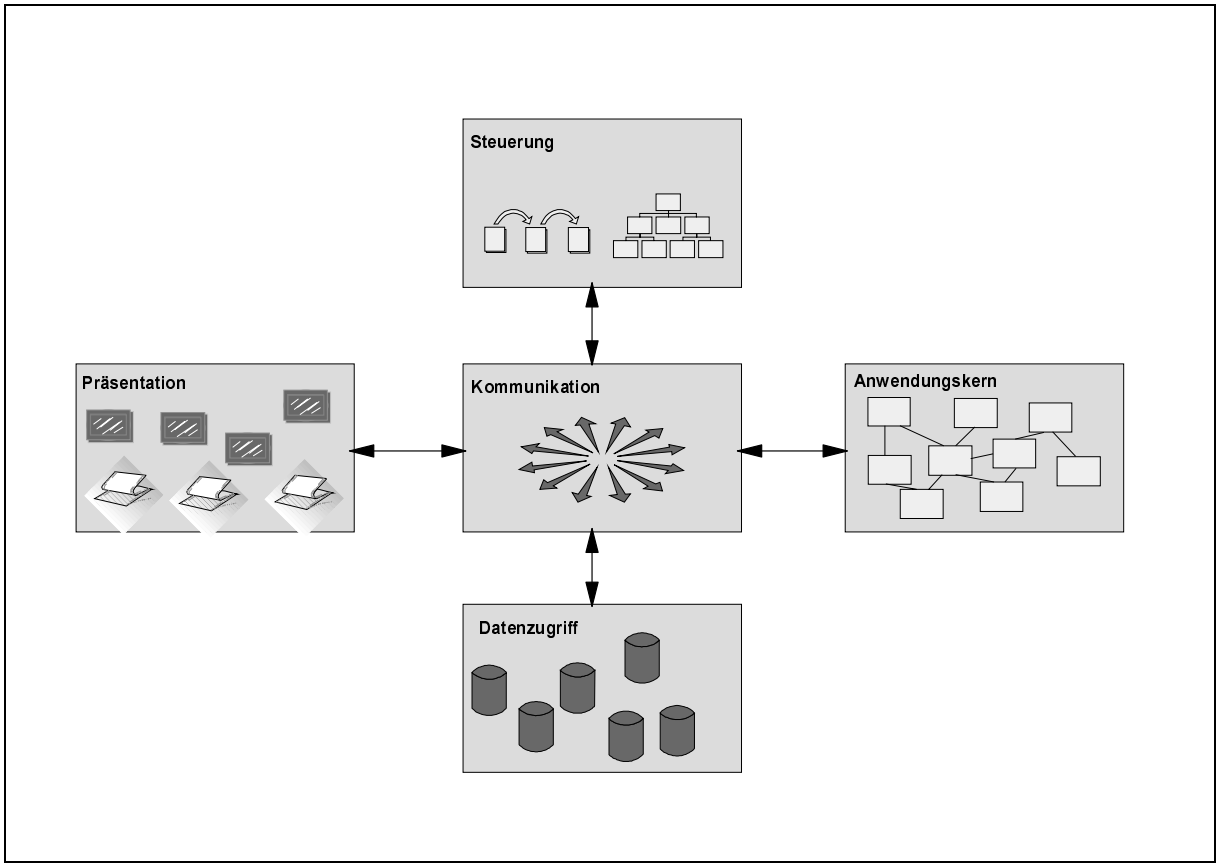

Abb. 16: Die fünf KID-Komponenten.

Zu dem Aspekt *Präsentation* gehören dann alle Module der Benutzungsschnittstelle. Das Aussehen von Bildschirmfenstern und -masken fällt hierunter ebenso wie die Gestaltung von Listen, Reports und Grafiken. Aber auch externe Schnittstellen, also die „Präsentation“ des KID-Anwendungssystems gegenüber anderen Anwendungen, gehören zur Präsentationskomponente.

In dem Aspekt *Datenzugriff* werden alle Module zusammengefaßt, welche die Anbindung des KID-Anwendungssystems an das zugrundeliegende Datenbanksystem realisieren. Nur die Datenzugriffs-

schicht „kennt“ die physische Datenbank und stellt den übrigen Komponenten das generalisierte Unternehmensdatenmodell zur Verfügung.

Zum *Anwendungskern* gehören alle fachlichen Module der Anwendung. Dabei sind die fachlichen Funktionen im Anwendungskern vollkommen unabhängig voneinander. Sie werden durch die *Steuerung* ereignisgesteuert zu Geschäftsvorfällen verknüpft. Die Geschäftsvorfallssteuerung stellt das Bindeglied zwischen dem Anwendungskern und der Präsentationskomponente dar, hier wird festgelegt, welche Masken zur Abwicklung einzelner Geschäftsvorfälle bearbeitet werden müssen, welche Benutzerinteraktionen welche Systemreaktionen auslösen, etc. Ich werde hierauf weiter unten noch detaillierter eingehen.

Um eine Verteilung der Objekte zu ermöglichen, kommunizieren alle Objekte über den *Kommunikator*, einer Art Object-Request-Broker. Ein Objekt ruft eine Methode eines anderen Objektes auf, indem es den Namen der gewünschten Operation sowie die erforderlichen Parameter an den Kommunikator schickt. Dieser macht das gerufene Objekt ausfindig, auch wenn es nicht lokal vorhanden ist, und ruft die entsprechende Methode auf. Die beim Methodenaufruf übergebenen Parameter sowie die erwarteten Rückgabeparameter werden in einem selbstbeschreibenden Format übergeben, um eine zusätzliche Entkoppelung von rufendem und gerufenem Objekt zu erreichen, wodurch die Änderbarkeit des Systems erhöht werden soll.

5.1.2.2 Benutzungshierarchie der Komponenten - fachliche Abstraktionsebenen

Durch die funktionale Entkoppelung der Module ergibt sich eine hierarchische Benutzungsstruktur zwischen den Komponenten, deren Schichten die unterschiedlichen Abstraktionsebenen von der zugrundeliegenden Technik widerspiegeln (siehe Abb. 17).

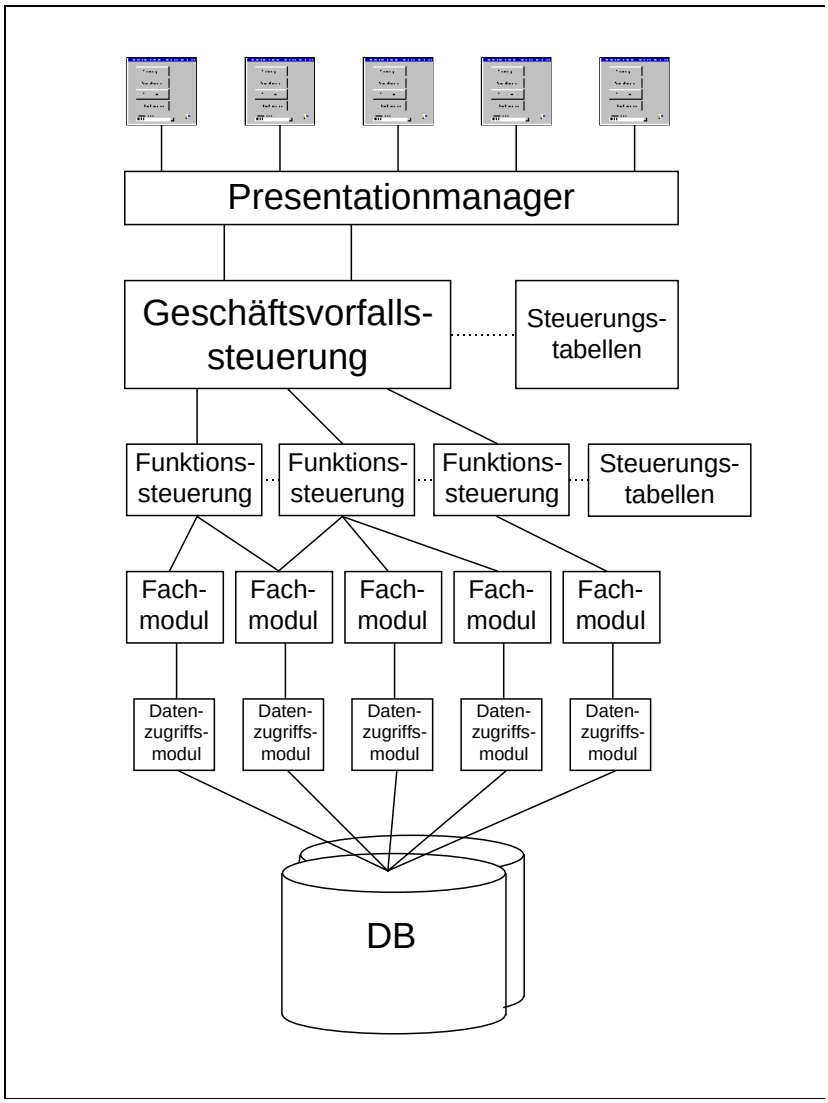

Abb. 17: Benutzungshierarchie der KID-Module.

Die unterste Ebene, die Ebene der *Datenzugriffsmodule*, abstrahiert von der physischen Datenbank. Nur über die Datenzugriffsmodule ist es der nächst höheren Schicht, der Ebene der Fachmodule, möglich, auf die Daten zuzugreifen.

Die *Fachmodule* stellen gewissermaßen „fachliche Datenzugriffsmodule“ dar. Sie ermöglichen der darüberliegenden Schicht den Zugriff auf die einzelnen Entities der Datenbank, zusätzlich findet aber eine fachliche Vorverarbeitung der Aufrufe statt. So werden etwa Konsistenz- und Plausibilitätsprüfungen innerhalb der einzelnen Entities, aber auch Prüfungen aufgrund fachlicher Abhängigkeiten zwischen ihnen, beispielsweise bei *Entityclustern*, durchgeführt. (Auf den Mechanismus der Entityclusterung werde ich weiter unten noch genauer eingehen.)

Die Ebene der *Funktionssteuerung* stellt den Zugriff auf die Objekte über fachliche (Objekt-) Methoden zur Verfügung. Oberhalb dieser Schicht ist es nicht mehr möglich, die zugrundeliegende relationale Datentechnik zu „sehen“.

In der *Geschäftsvorfallssteuerung* findet die Abbildung der Geschäftsprozesse des Unternehmens statt. Hier wird festgelegt, auf welche Präsentationsereignisse (Eingaben des Benutzers an der Benutzungsoberfläche) hin welche Funktionen oder Folgen von Funktionen ausgeführt werden sollen.

Der *Presentationmanager* ist für die Abwicklung der Benutzungsdialoge zuständig. Er wird von der Geschäftsvorfallssteuerung zur Anzeige der entsprechenden Masken aufgerufen und meldet die vom Benutzer ausgelösten Ereignisse (Drücken von Buttons, Schließen des Fensters, etc.) in Form von sogenannten *Präsentationsereignissen* an die Geschäftsvorfallssteuerung zurück.

Diese Aufteilung in hierarchische Schichten unterstützt die Möglichkeit, die Anwendungsentwicklung in Form eines inkrementellen Prototypings vorzunehmen: Um einen ablauffähigen Oberflächenprototypen zu erhalten, müssen „lediglich“ die Masken der Benutzungsschnittstelle beschrieben und die Steuerungstabellen der Geschäftsvorfallssteuerung gefüllt werden. Dieser Prototyp braucht dann „nur noch“ um die Fachfunktionsmodule ergänzt zu werden, um ein vollständiges Anwendungssystem zu erhalten. Ich werde hierauf bei der Vorstellung des Vorgehensmodells noch einmal eingehen.

5.2 Umsetzung eines objektorientierten Modells in die KID-Architektur

Da die KID-Architektur einerseits den Anspruch erhebt, objektorientiert zu sein, andererseits aber weder eine objektorientierte Programmiersprache noch eine solche Datenbank verwendet wird, wurde versucht, objektorientierte Mechanismen wie Polymorphie oder Vererbung über geeignete Hilfskonstrukte abzubilden.

Zunächst wird jeder „Vererbungsbaum“ des fachlichen Klassenmodells der Anwendung einer der KID-Basisklassen des generalisierten Objektmodells (siehe Abb. 15, S. 43) zugeordnet. Die Attribute der Wurzelklasse dieses Vererbungsbaumes werden in einem *Kernentity* beschrieben, alle Attribute der von dieser abgeleiteten Klassen werden jeweils in einem eigenen *Datenentity* abgelegt. Die Zusammenfassung des Kernentities und der dazugehörenden Datenentities wird in der KID-Terminologie ein *Entitycluster* genannt. In einem solchen Cluster sind also alle Exemplare der Wurzelklasse sowie sämtlicher von dieser erbenden Klassen enthalten.

Die Vererbungshierarchie an sich wird in einem eigenen Entitycluster, den *Geschäftsklassifikationen*, beschrieben. Über *Zuordnungsentities* können Assoziationsbeziehungen zwischen den in den Entityclustern abgelegten Objekten beschrieben werden, hierüber wird auch die Zuordnung der einzelnen Datensätze (also der Objekte) zu den Geschäftsklassen (also den Klassen des fachlichen Modells) abgebildet. Ich werde hierauf weiter unten noch näher eingehen.

5.2.1 Das Anwendungsbeispiel „Bibliothekssystem“

Diese Vorgehensweise bei der Abbildung einer objektorientierten Klassenhierarchie in die Strukturen des generalisierten Unternehmensmodells von EP/KID möchte ich nun anhand eines kleinen Anwendungsbeispiels näher beleuchten, um anschließend zu untersuchen, inwieweit sich mit diesen Hilfsmitteln eine „wirklich objektorientierte“ Modellierung mit KID vornehmen läßt.

5.2.1.1 Das fachliche Klassenmodell

In Abb. 18 ist das fachliche Klassenmodell des Fallbeispiels Bibliothekssystem abgebildet.

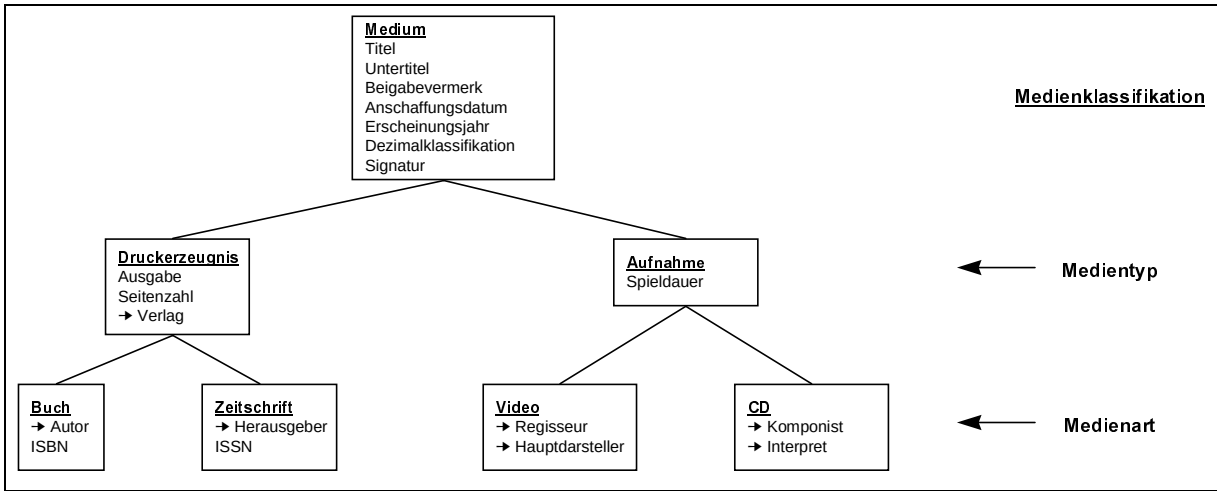

Abb. 18: Fachliches Klassenmodell mit Vererbungshierarchie und fachlichen Attributen des Fallbeispiels Bibliothekssystem.

Zum besseren Verständnis der Überführung eines solchen Klassenmodells in die KID-Architektur habe ich das Beispiel gegenüber der Vorstudie um ein paar Punkte erweitert.

Die Bibliothek verwalte allgemein *Medien*, die eine Reihe von Attributen gemein haben. So hat jedes Medium einen Titel, einen Untertitel und ein Erscheinungsjahr. Im Rahmen der Verwaltungsaufgaben der Bibliothek kommen noch einige Attribute hinzu, nämlich die in der Vorstudie beschriebenen Signatur und Dezimalklassifikation, sowie das Anschaffungsdatum und ein Beigabevermerk.

Innerhalb aller Medien lassen sich wiederum verschiedene *Medientypen* unterscheiden, nämlich Druckerzeugnisse und Aufnahmen, die weitere spezifische Attribute besitzen, beispielsweise haben alle Druckerzeugnisse einen herausgebenden Verlag und eine Seitenzahl, Aufnahmen hingegen eine Spieldauer.

An Druckerzeugnissen lassen sich die *Medienarten* Buch und Zeitschrift unterscheiden, an Aufnahmen Videokassetten und Compact Discs; auch diese Spezialisierungen besitzen wieder eigene Attribute. Die in der Abbildung verwendeten Pfeile in den Klassensymbolen stellen Verweise dar. So ha-

ben beispielsweise alle Objekte der Klasse Buch einen Verweis auf ein Objekt Autor, welches ein Objekt der Klasse Person ist.

5.2.1.2 Abbildung des Klassenmodells in das generalisierte KID-Geschäftsobjektmodell

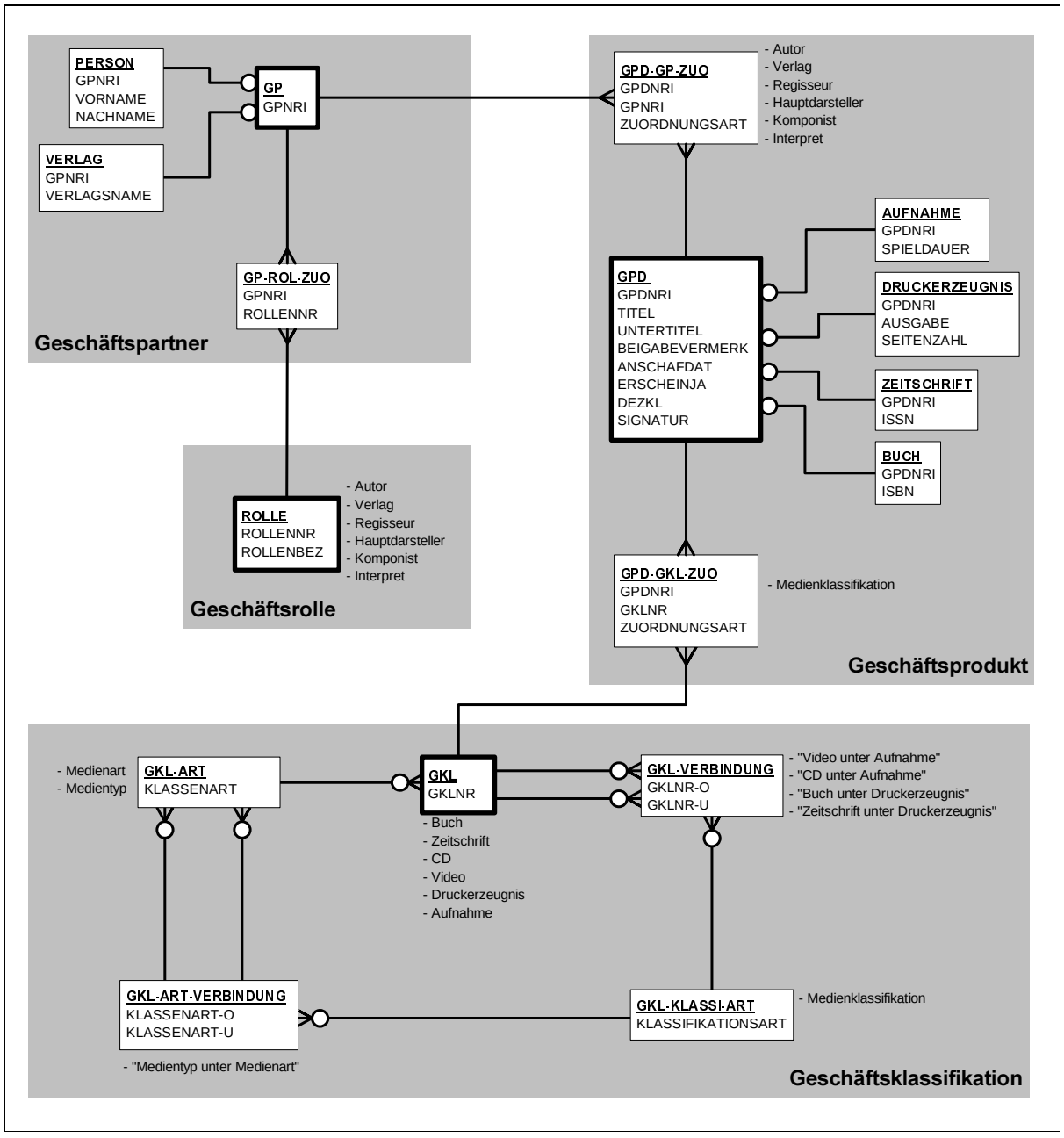

Abb. 19: Abbildung der fachlichen Vererbungshierarchie aus Abb. 18 in die KID-Architektur.

Bei der Überführung des Klassenmodells der Bibliothek in das generalisierte Datenmodell der KID-Architektur, wie sie in Abb. 19 dargestellt ist, werden die verwalteten Medien dem Entitycluster *Geschäftsprodukt* zugeordnet. Dementsprechend werden alle Attribute aus der in Abb. 18 abgebildeten Oberklasse Medium in dem Kernentity Geschäftsprodukt (GPD) abgelegt, alle Attribute, die durch Spezialisierungen dieser Klasse hinzukommen, in entsprechenden Datenentities. Das technische At-

tribut GPDNRI repräsentiert eine eindeutige Objekt-ID, die in allen Entities des Entityclusters verwendet wird und über welche die richtige Zuordnung von Einträgen im Kernentity zu den jeweiligen Datenentities sichergestellt wird.

Die Vererbungshierarchie an sich wird, wie oben bereits erwähnt, in dem Entitycluster *Geschäftsklassifikation* beschrieben. In dem Kernentity (GKL) werden zunächst alle Klassennamen „lose gesammelt“. In dem Entity *Klassifikationsarten* (GKL-KLASSI-ART) wird festgehalten, welche Arten von Klassifikationen es im gesamten KID-Anwendungssystem gibt, in diesem Beispiel also nur eine einzige, nämlich die Medienklassifikation. In dem Entity *Klassenarten* (GKL-ART) müssen die unterschiedlichen Ebenen der Vererbungshierarchie benannt werden, hier also Medienart und Medientyp. In den beiden Entities *Klassenart-Verbindung* (GKL-ART-VERBINDUNG) und *Klassen-Verbindung* (GKL-VERBINDUNG) werden schließlich die hierarchischen Anordnungen zum einen zwischen den verschiedenen Klassenarten, zum anderen zwischen den einzelnen Klassen, beschrieben. Auf diese Weise wird der gesamte Vererbungsbaum der fachlichen Vererbungshierarchie beschrieben.

Analog zu der Abbildung der Medien in dem Entitycluster Geschäftsprodukt, werden die Klassen Verlag, Autor, Herausgeber, Regisseur, Hauptdarsteller, Komponist und Interpret in dem Entitycluster *Geschäftspartner* abgebildet. Das einzige Attribut, was bei dieser Generalisierung allen Objekten gemein ist, ist die Objekt-ID, hier GPNRI genannt. Fachlich lassen sich zwei Klassen von Geschäftspartnern unterscheiden, nämlich zum einen natürliche Personen (Autor, Regisseur, Interpret, etc.), zum anderen die Verlage. Die für diese Klassen spezifischen Attribute werden auch hier wieder in entsprechenden Datenentities (PERSON bzw. VERLAG) abgelegt.

Die unterschiedlichen Rollen, die Objekte in einem KID-System innehaben können, werden, ähnlich wie die Geschäftsklassifikationen, in einem eigenen Entitycluster (ROLLE) abgelegt und können über entsprechende Zuordnungsentities in den Entityclustern der Objekte diesen zugeordnet werden. In dem hier vorliegenden Beispiel gibt es nur die unterschiedlichen Rollen, in denen Geschäftspartner auftreten können, diese werden dann über das Rollenzuordnungsentity GP-ROL-ZUO dem entsprechenden Geschäftspartnerobjekt zugeordnet. Im Gegensatz zu einer Klassifikation, wie sie bei den Medien vorgenommen wurde, können hier einem Objekt auch mehrere Rollen zugeordnet werden. Beispielsweise kann ein und die selbe Person sowohl Autor eines Buches als auch Hauptdarsteller in einem Video sein.

Über die Zuordnungsentities eines Entityclusters werden sowohl die Klassifizierungen und Rollen der Objekte als auch die Verweise auf andere Objekte, wie sie in Abb. 18 zu sehen sind, abgebildet. Bei Klassifikationen und Verweisen auf andere Objekte wird über eine Zuordnungsart angegeben, um was für eine Zuordnung es sich fachlich handelt. Über die *Geschäftsprodukt-Geschäftsklasse-Zuordnung* (GPD-GKL-ZUO) werden die Geschäftsproduktobjekte zunächst klassifiziert, beispielsweise als Ob-

jekte vom Typ Buch über eine Zuordnung zu der Geschäftsklasse `Buch`. Eine solche Zuordnung hat die Zuordnungsart `Medienklassifikation`. Über die *Geschäftsprodukt-Geschäftspartner-Zuordnung* (`GPD-GP-ZUO`) wird jedem Buch ein Autor zugeordnet. Eine solche Zuordnung hat die Zuordnungsart `Autor` und ordnet dem Objekt vom Typ Buch, identifiziert über seine Objekt-ID `GPDNRI`, ein Objekt vom Typ Person, identifiziert über die Objekt-ID `GPNRI`, zu, wobei dieses Objekt die Rolle `Autor` innehaben muß. Über eine Zuordnung der Zuordnungsart `Verlag`, wird dem Buch ein Geschäftspartner mit der Rolle `Verlag` zugewiesen.

Der fachliche „Zusammenhalt" zwischen den Zuordnungen, Klassifikationen, Kern- und Datenentities kann in KID über das *Regelwerk* sichergestellt werden. In diesem Regelwerk werden Sachverhalte beschrieben wie beispielsweise: *„Wenn ein `Geschäftsprodukt` eine Zuordnung der Zuordnungsart `Medienklassifikation` zu der Geschäftsklasse `Buch` besitzt, dann muß es zu diesem ein Datenentity vom Typ `Buch` geben"* oder *„Wenn ein `Geschäftsprodukt` eine Zuordnung der Zuordnungsart `Medienklassifikation` zu der Geschäftsklasse `Buch` besitzt, dann muß es zu diesem eine Zuordnung der Zuordnungsart `Autor` zu einem `Geschäftspartner`, welcher die Rolle `Autor` inne hat, geben."* Wie an diesem Beispiel bereits zu sehen ist, wird in KID über dieses Regelwerk auch der Bezug zwischen den unterschiedlichen Datensätzen in den einzelnen Entities und der Vererbungshierarchie des fachlichen Modells wiederhergestellt.

5.2.1.3 Beschreibung einer Vererbungshierarchie mit Hilfe der „KID-Workbench"

Zur besseren Veranschaulichung der Klassifikationsmechanismen in KID möchte ich kurz vorstellen, wie diese mit Hilfe der *KID-Workbench* beschrieben werden. Die Abbildung der fachlichen Vererbungshierarchie in eine entsprechende Hierarchie von Geschäftsklassifikationen ist in Abb. 20 dargestellt.

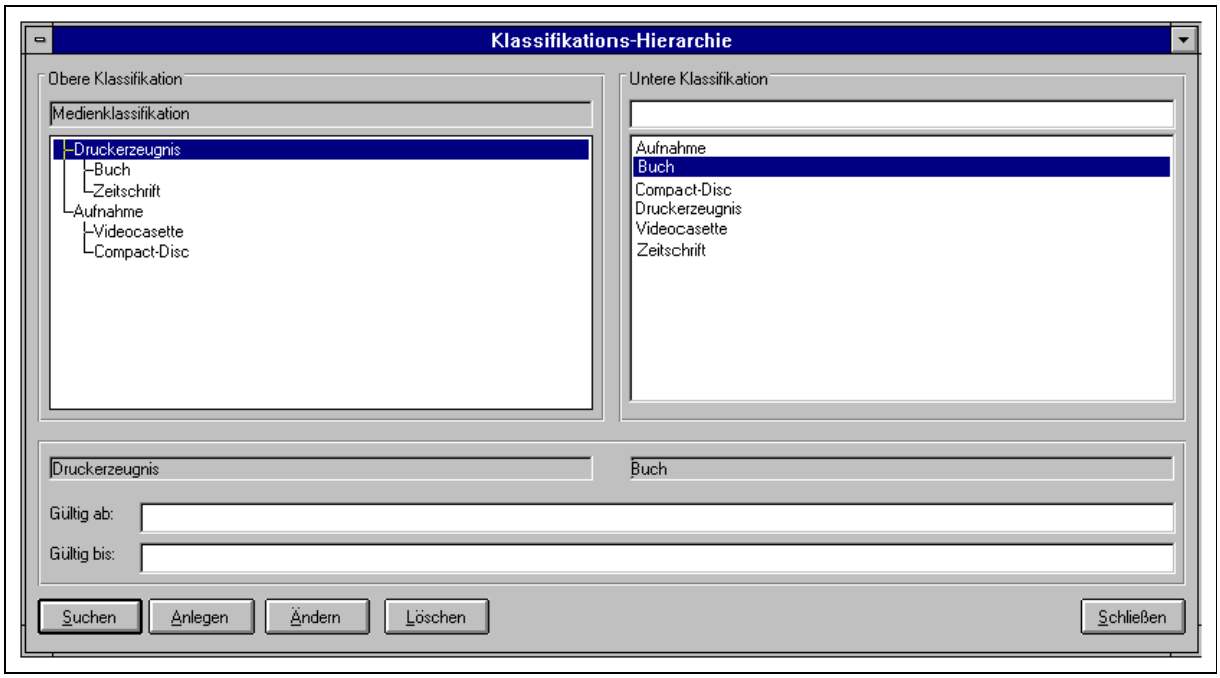

Abb. 20: Erstellung einer Klassifikationshierarchie mit der KID-Workbench.

Im rechten Teil der dargestellten Maske sind alle Geschäftsklassen des Systems aufgelistet beziehungsweise können hier hinzugefügt werden, im linken Teil sind die Geschäftsklassenverbindungen baumförmig dargestellt. Durch Markieren eines Knotens im linken Teil sowie einer Geschäftsklasse im rechten Teil der Maske und dem anschließenden Drücken des Buttons `Anlegen` kann eine neue Geschäftsklassenverbindung angelegt werden. Auf diese Weise läßt sich die Vererbungshierarchie, wie sie in Abb. 18 dargestellt ist, abbilden.

5.2.1.4 Beschreibung von Objektbeziehungen in Zuordnungsentities

Wie weiter oben bereits erwähnt wurde, werden die möglichen Verbindungen, die es zwischen den Klassen des fachlichen Klassenmodells geben kann, in der KID-Architektur über Zuordnungsentities zwischen den entsprechenden Entityclustern des KID-Datenmodells abgebildet. So muß für das Fallbeispiel der Bibliotheksanwendung unter anderem ein Zuordnungsentity für die Verweise von Geschäftsprodukten, hier also den Medien, auf die entsprechenden Geschäftspartner, Autoren, Interpreten, etc. (`GPD-GP-ZUO` in Abb. 19) geben. Wie ein solches Zuordnungsentity in der KID-Workbench definiert wird, ist in Abb. 21 zu sehen.

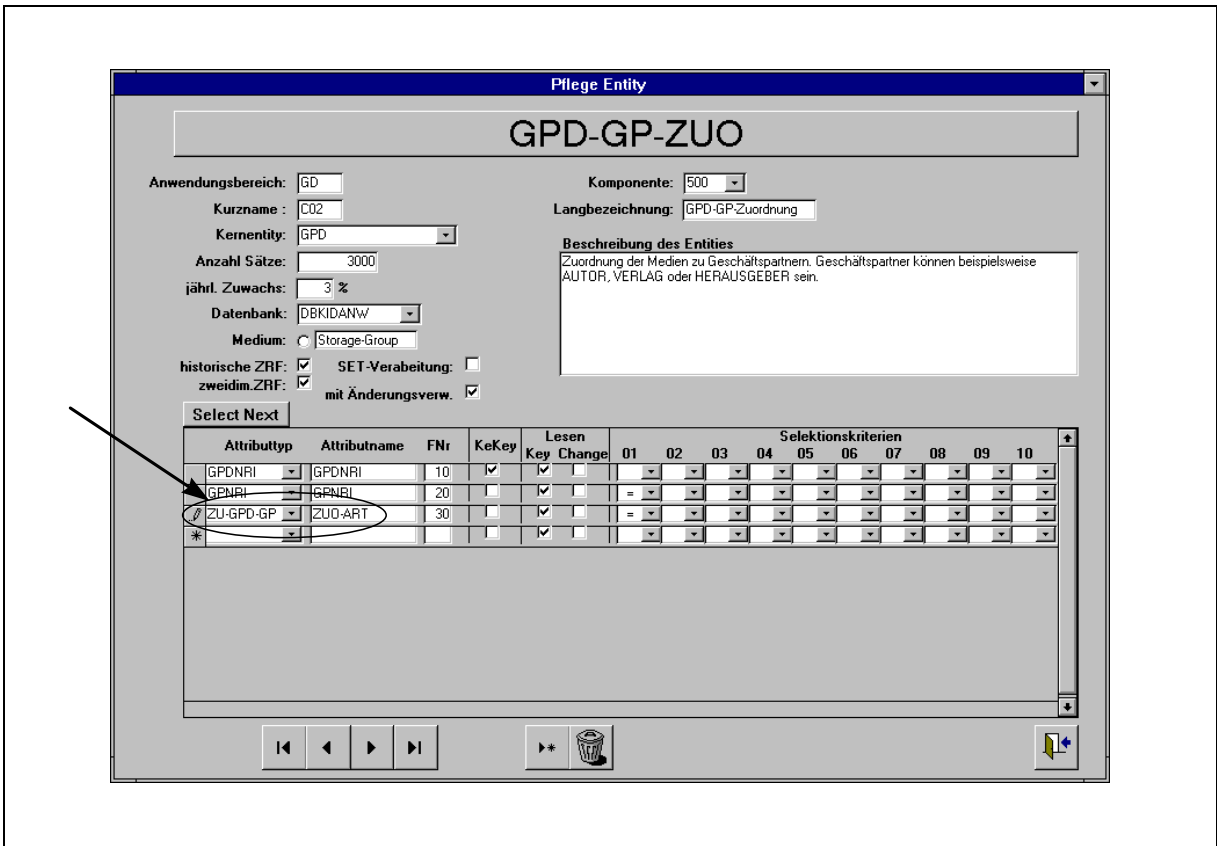

Abb. 21: Pflege einer Zuordnungsnetitybeschreibung mit der KID-Workbench.

Dieses Zuordnungsentity beinhaltet neben den Objekt-IDs der beiden zu verbindenden Objekte (`GPDNRI` und `GPNRI`) noch dasjenige Attribut, welches die Zuordnungsart angibt (`ZUO-ART`, siehe Pfeil in Abb. 21).

Der Attributtyp, von dem dieses die Zuordnungsart angebende Attribut ist, muß, wie alle in dem KID-Anwendungssystem verwendeten Attributtypen, im *Attributtypkatalog* deklariert werden (siehe Abb. 22).

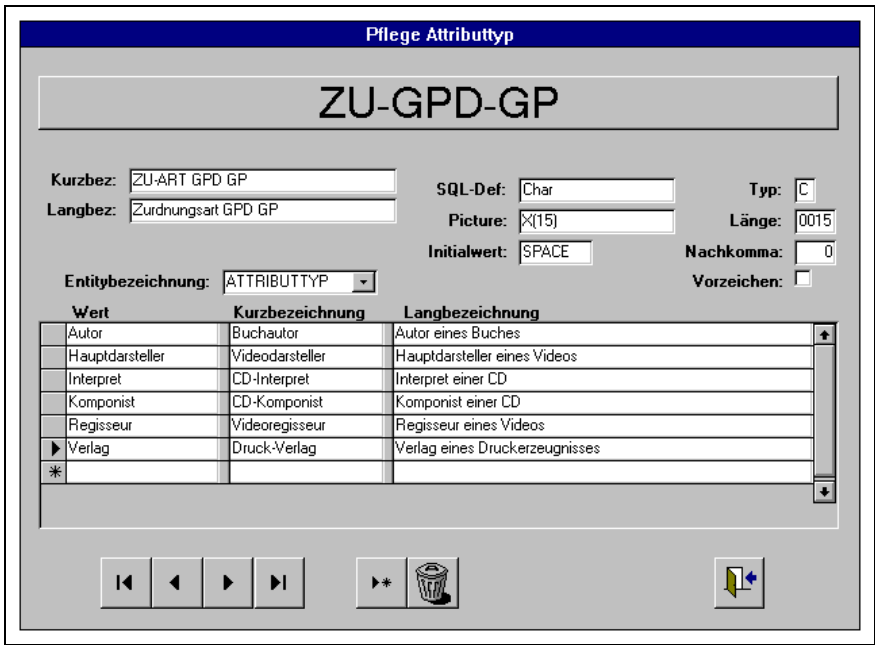

Abb. 22: Beschreibung eines Attributtyps eines KID-Anwendungssystems mit gültigen Ausprägungen.

Da dieses Attribut fachlich bedingt nur bestimmte Werte annehmen darf, müssen diese in der Spalte `Wert` der Attributtypdeklaration explizit angegeben werden.

5.2.1.5 Beschreibung von (fachlichen) Regeln im Regelwerk

Was für Zuordnungen Objekte eines bestimmten Typs haben müssen, und von welcher Zuordnungsart dieses sein dürfen, wird, wie bereits erwähnt, im KID-Regelwerk festgehalten. Hier wird also, übertragen auf das objektorientierte Klassenmodell, festgelegt, auf was für Objekte Exemplare einer bestimmten Klasse Verweise haben müssen.

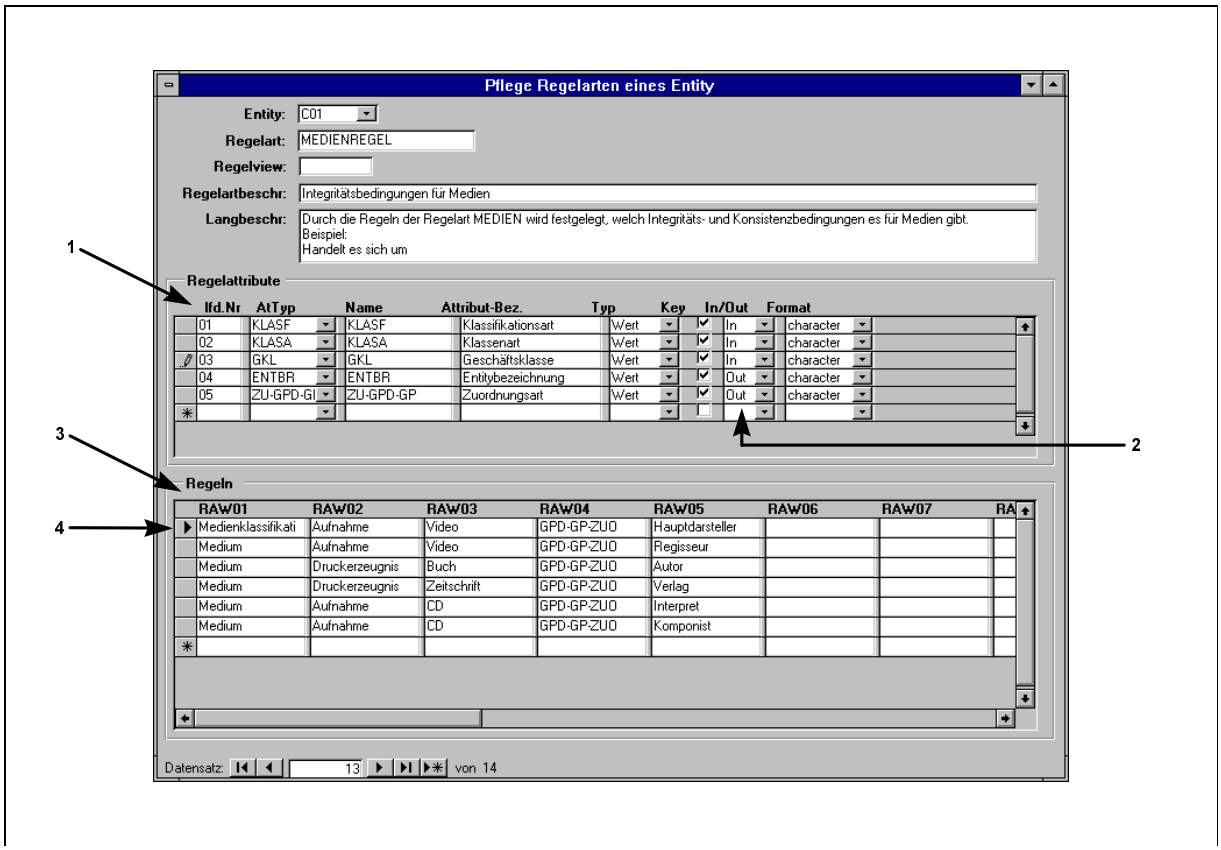

Abb. 23: Pflege des Regelwerkes eines KID-Anwendungssystems.

In Abb. 23 ist die Formulierung der weiter oben aufgeführten Regel aus dem Fallbeispiel zu sehen. Die Beschreibung geschieht ähnlich einer logischen Implikation. Im oberen Teil der Maske werden die Attribute der Regel festgelegt (Pfeil 1), hier wird zwischen den *In-* und *Out-Werten* der Implikation unterschieden (Pfeil 2).

Im unteren Abschnitt der Maske sind die einzelnen Regeln dieser Regelart definiert, jede Zeile der Tabelle entspricht dabei einer Regel (Pfeil 3). Die Reihenfolge der Regelattributwerte (`RAW01`, `RAW02`, ...) in einer solchen Zeile korrespondiert mit der Reihenfolge der im oberen Abschnitt angegebenen Regelattribute. Eine Regel stellt also eine gültige Kombination von Ausprägungen der Regelattribute dar, wobei durch die oben genannte Unterscheidung von In- und Out-Werten die „Richtung" der Implikation angegeben wird („*Wenn In-Werte gleich soundso, dann Out-Werte gleich soundso*").

Die Regel, die beispielsweise in der ersten Zeile beschrieben ist (Pfeil 4), ist folgendermaßen zu lesen: „*Wenn ein Objekt im Rahmen einer Klassifikation der Klassifikationsart* `Medium` *(RAW01) mit der Klassenart* `Aufnahme` *(RAW02) und der Geschäftsklasse* `Video` *(RAW03) klassifiziert ist, dann muß es zu diesem Objekt auch einen Eintrag in dem Entity* `GPD-GP-ZUO` *(RAW04) mit der Zuordnungsart* `Hauptdarsteller` *(RAW05) geben.*"

Die Einhaltung dieser so im Regelwerk formulierten Regeln wird durch die *Freigabeprüfung* automatisch überprüft, um die fachliche Konsistenz der Daten im Rahmen der angegebenen Regeln sicherzustellen.

Nachdem auf diese Weise alle statischen Komponenten des Anwendungssystems beschrieben wurden, können daraus alle Datenbankschemata, Datenzugriffsmodule und Fachmodule (siehe Abb. 17) automatisch generiert werden.

5.2.1.6 Abbildung von Objektmethoden in Funktionssteuerungen

Die fachlichen (Objekt-) Methoden werden in KID auf der Ebene der *Funktionssteuerung* in die entsprechenden Aufrufe der *Fachmodule* umgesetzt (siehe 5.1.2.2 Benutzungshierarchie der Komponenten - fachliche Abstraktionsebenen). Beispielsweise wird die Umgangsform „katalogisieren“ der Klasse Buch aus dem Anwendungsbeispiel Bibliothekssystem in eine Funktionssteuerung mit dem Namen `Buch katalogisieren` abgebildet. Diese Funktionssteuerung ruft dann die entsprechenden Fachmodule auf, mit denen die entsprechenden notwendigen Schritte für das Abbild der Klasse Buch in dem generalisierten KID-Datenmodell vollzogen werden. In diesem Fall wären das die Fachmodule `GPD anlegen`, `Buch anlegen`, `GPD-GKL-Zuordnung anlegen`, `GP anlegen`, `GPD-GP-Zuordnung anlegen` usw.. Die Freigabeprüfung überwacht dabei automatisch die Einhaltung der im Regelwerk formulierten Konsistenzkriterien.

5.3 Modellierung von Geschäftsvorfällen in KID

Über das Klassenmodell hinaus werden in einem KID-Anwendungssystem noch *Geschäftsvorfälle* modelliert, in welchen beschrieben wird, auf welche *Präsentationsereignisse* hin welche (Folgen von) Funktionssteuerungen ausgeführt beziehungsweise welche Masken angezeigt werden sollen. Präsentationsereignisse stellen dabei Eingaben des Benutzers an der Benutzungsschnittstelle dar.

Auch bei der Beschreibung der Geschäftsvorfälle können, ähnlich wie in den Funktionssteuerungen, erfolgsabhängige Verzweigungen im Ablauf vorgenommen werden.

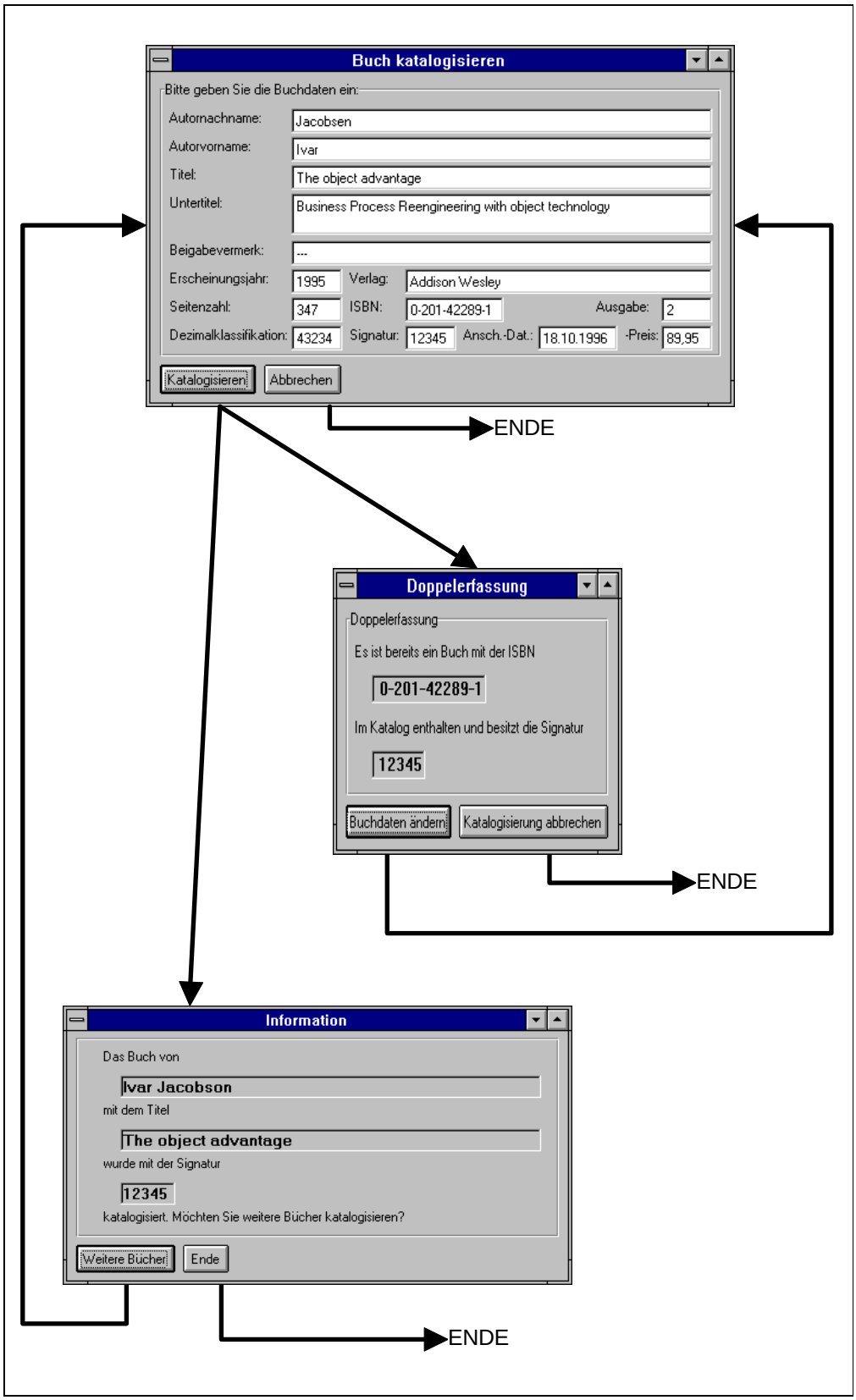

Abb. 24: Maskenablaufplan des Geschäftsvorfalls „Buch erfassen“.

In Abb. 24 ist ein „Prosy“ für den Geschäftsvorfall „Buch erfassen“ aus dem Fallbeispiel der Bibliotheksanwendung zu sehen. Ein solches Prosy beschreibt die möglichen „Wege“ durch einen Geschäftsvorfall, wie sie sich an der Benutzungsoberfläche darstellen.

Auf die Modellierung von Geschäftsvorfällen in EP/KID sowie auf den Dokumenttyp Prosy werde ich weiter unten, im Rahmen der Vorstellung des Vorgehensmodells, noch näher eingehen.

5.4 Untersuchung: Inwieweit ist EP/KID objektorientiert?

Inwieweit mit dem KID-Ansatz unter Einhaltung der genannten Punkte entwickelte Anwendungssysteme wirklich eine objektorientierte Architektur besitzen, soll im folgenden genauer untersucht werden.

Mit Hilfe der Entityclusterung ist es in KID zwar prinzipiell möglich, eine objektorientierte Klassenhierarchie in ein relationales Datenschema zu überführen. Allerdings sind dazu eine Reihe recht komplizierter Schritte notwendig. Zunächst muß das Klassenmodell in entsprechende Strukturen von Kern- und Datenentities abgebildet werden, ähnlich dem von Züllighoven et al. vorgeschlagenen Vorgehen.[24] Anschließend müssen die Klassenhierarchien des fachlichen Modells in der am Anwendungsbeispiel aufgezeigten Form in dem Entitycluster *Geschäftsklassifikationen* beschrieben werden. Zu einer vollständigen Abbildung der Klassenhierarchien müssen außerdem die Zusammenhänge zwischen den Zuordnungen der den Klassen entsprechenden Rollen und Klassifikationen einerseits, sowie den damit verbundenen Datenentities andererseits, im KID-Regelwerk abgelegt werden.

Diese Vorgehensweise ermöglicht es zwar, ein objektorientiertes Klassenmodell relativ „sauber“ in ein relationales Datenschema abzubilden. Da die Schritte aber, die zu einer solchen Abbildung eines Klassenmodells in die entsprechenden Relationen der KID-Architektur vonnöten sind, einerseits recht komplex sind, auf der anderen Seite diese Vorgehensweise jedoch nicht erzwungen, sondern lediglich ermöglicht wird, bedarf die konsequente Einhaltung dieser Vorgehensweise einer starken Selbstdisziplin und vor allem auch sehr guter Kenntnisse der Objektorientierung seitens der Entwickler.

Kritisch anzumerken ist in meinen Augen allerdings, daß durch die KID-Architektur eine Aufweichung der Objektkapsel stattfindet. In begrenztem Rahmen werden zwar Daten und Funktionen zu einem Objekt gekapselt, da auf die Objektattribute nur über die in den Funktionssteuerungen abgebildeten Objektmethoden zugegriffen werden kann. Die Methoden und die Daten eines Objektes werden allerdings getrennt voneinander modelliert. Dieses ermöglicht unter anderem, Funktionssteuerungen zu implementieren, die „objektübergreifend“ wirken, also auf die Attribute mehrerer Objekte zugreifen. Eine weitere Folge ist, daß zur Laufzeit bei einem Methodenaufruf die Zuordnung dieser Methode zu den tatsächlich auszuführenden Instruktionen nicht anhand der Klassendefinition, also innerhalb der Kapsel, sondern außerhalb von dieser stattfindet. Eine ähnliche Vorgehensweise findet sich auch

in der Programmiersprache CLOS (Common Lisp Object System), die in diesem Zusammenhang auftretenden kritischen Punkte werden auch von Wegner diskutiert:

> »The requirement that all methods owe their primary allegiance to a single class and operate at execution time on a single object greatly strengthens encapsulation. Weakening this requirement increases flexibility but can create unmanageable (spaghetti-like) object structures. [...] The price of this added flexibility is a loss of security and encapsulation. A late method definition for a class may change the behavior of already created instances. Methods referring to instance variables in multiple classes weaken encapsulation, just as a person working for multiple masters may give away the secrets of one to another.« ([Weg90], S. 12)

Die in KID für die Abbildung objektorientierter in relationale Strukturen verwendete Technik hat auch Auswirkungen auf den Umfang sowohl der Polymorphie als auch der Vererbung. Vorausgesetzt, daß die Vererbungshierarchie auf die im Anwendungsbeispiel Bibliothekssystem angedeutete Weise im Regelwerk beschrieben wurde, besitzt ein KID-Objekt alle Datenentities seiner Oberklassen und kann dadurch in begrenztem Rahmen polymorph verwendet werden. Da die Methoden jedoch, wie oben bereits angemerkt, „außerhalb“ der Klassenbeschreibung implementiert werden, also auch nicht der Vererbung entsprechend hierarchisch angeordnet werden können, müssen sie mit unterschiedlichen Namen versehen werden. Diese Tatsache führt dazu, daß ein polymorphes Überschreiben einer geerbten Methode nicht möglich ist.

Zusammenfassend läßt sich also sagen, daß mit EP/KID eine objektorientierte Modellierung eines Anwendungssystems vorgenommen werden kann, allerdings nur mit den hier genannten Einschränkungen. Da die Möglichkeiten der Abbildung objektorientierter Mechanismen in die KID-Architektur nicht erzwungen, sondern allenfalls ermöglicht werden, muß auf die Einhaltung aller dazu notwendigen Schritte sehr genau geachtet werden.

Eine Anwendungsarchitektur, in der Materialien unterschiedlicher Arten mit einem Satz an Werkzeugen bearbeitet werden, wie sie mit dem WAM-Ansatz erreicht werden soll, ist meines Erachtens mit KID nicht möglich.

[24] Vergleiche dazu [KGZ94], S. 78ff.

6 KID - Das Vorgehensmodell

Nachdem ich den durch die KID-Architektur vorgegebenen Rahmen für die Anwendungs- und Informationssystementwicklung mit KID abgesteckt habe, möchte ich nun zu der Vorstellung des von mir entwickelten Vorgehensmodells kommen. Ziel hiervon ist es, ein Vorgehen und mit diesem verbundene Dokumenttypen zu beschreiben, welche innerhalb dieses durch die KID-Architektur vorgegebenen Rahmens einen möglichst „guten“ objektorientierten Entwurf im Sinne der im ersten Teil dieser Arbeit vorgestellten Bewertungskriterien sowie der am Arbeitsbereich Softwaretechnik vertretenen Sichtweise ermöglichen.

6.1 Zyklische, evolutionäre Vorgehensweise

Wie bereits in der Einordnung dieser Arbeit gesagt wurde, hat der objektorientierte Softwareentwurf einige Vorteile gegenüber der „herkömmlichen“ Art der Softwareerstellung. Mit herkömmlich ist in diesem Zusammenhang die Trennung zwischen den Phasen Analyse des Anwendungsbereiches und Entwurf des Anwendungssystems, sowie damit einher gehend ein einmaliges, wasserfallartiges Durchlaufen dieser Phasen mit dem Anwendungssystem als Endprodukt, gemeint.

Bei der hier vertretenen Sichtweise hingegen wird Softwareentwicklung in erster Linie als ein gemeinsamer Kommunikations- und Lernprozeß von Entwicklern und den am Entwicklungsprozeß partizipierenden Anwendern verstanden, durch den ein gemeinsames Verständnis von dem Anwendungsbereich und der entwickelten Software evolviert. Die Anwendungsentwicklung ist eine Folge von Zyklen, in denen sich die Weiterentwicklung der Software und ihr Einsatz abwechseln. Diese Vorgehensweise soll, besser als die herkömmliche, ermöglichen, die Hürden im Entwicklungsprozeß zu überwinden, die Züllighoven et al. als „ein zentrales Dilemma der Softwareentwicklung“ bezeichnen:

> »Softwareentwickler haben immer nur eine begrenzte Fachkenntnis desjenigen Anwendungsbereichs, den sie gerade modellieren. Durch distanzierte „beobachtende“ Analyse oder Auswertung von Vorschriften und Arbeitsplatzbeschreibungen lassen sich die tatsächlichen Arbeitsformen und die Details einer Betriebswirklichkeit nicht erkennen. Andererseits erledigen Fachleute ihre tägliche Arbeit so routiniert, daß sie darüber auf Anforderung nicht einfach und systematisch Auskunft geben können. In dieser Situation, die durch die Kluft zwischen der Welt der Entwickler und der der Anwender gekennzeichnet ist, kann u.E. die enge Verzahnung von *objektorientiertem Entwurf* und *evolutionärer Systementwicklung* weiterführen. Im Mittelpunkt stehen dabei die *zyklische Verbindung von*

> *analysierenden, entwerfenden und bewertenden Tätigkeiten* und die explizite Berücksichtigung von Kommunikations- und Lernprozessen.« ([KGZ94], S. 91)

Dieser Fokussierung auf die im Rahmen der Softwareentwicklung stattfindenden Lern- und Kommunikationsprozesse und der hiermit verbundenen Evolution von Sichtweise und Fachkenntnis der beteiligten Personen muß natürlich auch das Vorgehensmodell Rechnung tragen. Es muß ein entsprechender Satz an Dokumentationsformen zur Verfügung gestellt werden, die diese Prozesse in geeigneter Form unterstützen und fördern. Ferner dürfen im Vorgehensmodell nicht im Vorwege Annahmen über den Verlauf des Entwicklungsprozesses gemacht werden. Es kann also keine Festlegung mehr auf eine (vor-) bestimmte Reihenfolge der erforderlichen Schritte geben. Vielmehr ist eine Vorgehensweise gefordert, die flexibel und situativ an die sich im Verlaufe des Entwicklungsprozesses ändernden Anforderungen adaptierbar ist.

6.2 Das KID-Vorgehensmodell im Überblick

Im vorigen Kapitel habe ich bereits gezeigt, in welcher Form und in welchem Umfang ein objektorientierter Entwurf mit EP/KID technisch in ein Anwendungssystem überführt werden kann. Zur Vervollständigung des KID-Vorgehensmodells fehlen noch der besagte Satz an Dokumenttypen, welche die genannten Lern- und Kommunikationsprozesse fördern und eine objektorientierte Anwendungsentwicklung, die den formulierten Anforderungen genügt, ermöglichen, sowie Techniken für die Steuerung und Kontrolle des Entwicklungsprozesses.

Im Rahmen des WAM-Ansatzes haben Züllighoven et al. eine Reihe von Dokumenttypen vorgestellt, die sich ihrer Meinung nach insbesondere für die objektorientierte Anwendungsentwicklung nach dem WAM-Ansatz gut eignen.[25] Einen Teil dieser Dokumenttypen habe ich mein Vorgehensmodell für die Anwendungsentwicklung mit EP/KID übernommen.

Da sich KID-Anwendungssysteme jedoch, wie oben bereits besprochen, in einigen Punkten grundsätzlich von WAM-Anwendungssystemen unterscheiden, habe ich nur solche Dokumenttypen übernommen, die sich meiner Meinung nach auch für die Anwendungsentwicklung mit EP/KID sinnvoll einsetzen lassen. Darüber hinaus habe ich einige weitere Dokumenttypen entwickelt, die ich aufgrund der genannten Unterschiede für hilfreich oder sogar erforderlich angesehen habe.

[25] Siehe dazu beispielsweise [LZ97], [GWZ97], [KGZ94] oder [GZ92].

Ich möchte das von mir vorgeschlagene Vorgehensmodell für die Anwendungsentwicklung mit EP/KID zunächst in einem kurzen Überblick in seiner Gesamtheit vorstellen, bevor ich im weiteren auf die einzelnen Abschnitte und die darin verwendeten Dokumenttypen genauer eingehe.

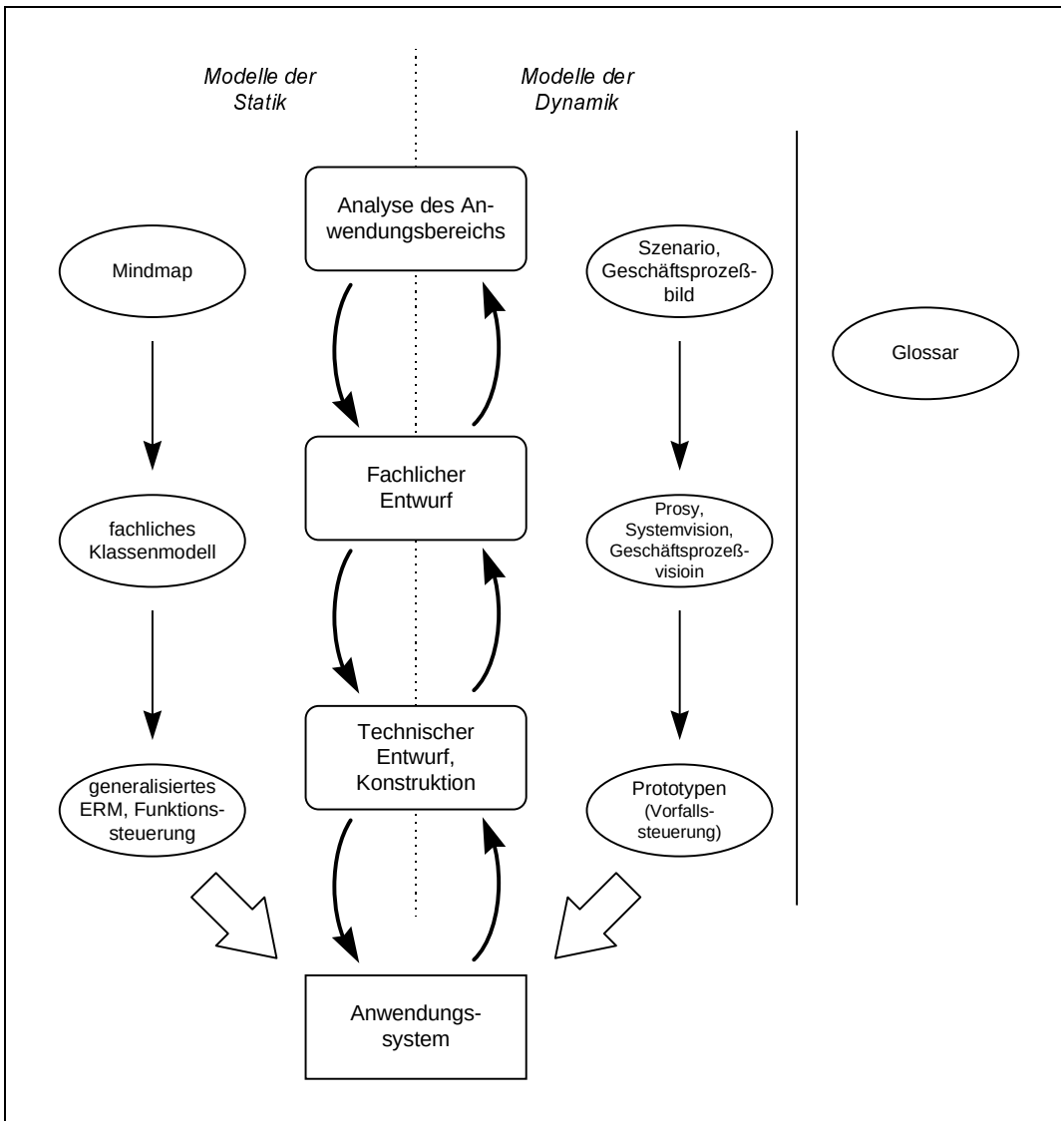

Abb. 25: Schematische Darstellung des KID-Vorgehensmodells.

In Abb. 25 ist das von mir vorgeschlagene KID-Vorgehensmodell schematisch dargestellt. Die zyklischen Pfeile in der Mitte der Abbildung sollen andeuten, daß es keine fest vorgegebene Abfolge der unterschiedlichen Entwicklungsphasen gibt, sondern daß vielmehr ein Wechseln zwischen den unterschiedlichen Abschnitten und den mit diesen verbundenen Tätigkeiten jederzeit möglich ist. Der große, umschließende Zyklus stellt dar, daß die Anwendungsentwicklung in Form der oben angesprochenen zyklischen Verschränkung von Entwicklung und Einsatz der Software stattfindet und nicht nach einmaligem Durchlaufen des Vorgehensmodells abgeschlossen ist.

Begonnen wird mit der Analyse des Anwendungsbereiches. Ziel dabei ist es, bei den Entwicklern und den zukünftigen Benutzern des Anwendungssystems, also den Sachbearbeitern der betroffenen

Fachabteilungen, zu einem gemeinsamen fachlichen Verständnis des durch das Anwendungssystem zu unterstützenden Anwendungsumfeldes zu gelangen. Für den Entwickler bedeutet dieses in erster Linie, daß er die fachlichen Zusammenhänge verstehen und erkennen lernen muß.

Ein Dokumententyp, der diesen ersten Schritt der Annäherung unterstützen soll, ist das *Mindmap*, das der Entwickler zusammen mit den Anwendern der Fachabteilungen in einer Art „Brainstorming" erstellt. In einem solchen Mindmap werden funktionelle Rollen, Aufgaben und Tätigkeiten aus dem Arbeitsumfeld des Anwendungsbereiches systematisch „gesammelt".

Da in einem Mindmap allerdings keine dynamischen Aspekte abgebildet werden können, beschreibt der Entwickler die im Arbeitsumfeld anfallenden Tätigkeiten, Kooperations- und Koordinationsformen anschließend mit Hilfe von *Szenarios* und *Geschäftsprozeßbildern*. Diese Dokumente dienen sowohl der Dokumentation der erkannten Abläufe als auch, im Rahmen des oben angesprochenen Lernprozesses, zur Rückkopplung mit den Anwendern, um festzustellen, ob die beschriebenen Sachverhalte von den Entwicklern richtig erkannt und verstanden wurden. Die Geschäftsprozeßbilder dienen dabei zur Darstellung von übergreifenden Aufgaben und Kooperationen und helfen zum einen, Schwachstellen in den bestehenden Geschäftsprozessen aufzudecken, zum anderen, sinnvolle Einsatzmöglichkeiten einer Computerunterstützung zu finden.

Im Rahmen des fachlichen Entwurfes kann aufbauend auf diesen Analyseergebnissen als statisches Modell des Anwendungssystems ein *fachliches Klassenmodell* erstellt werden, in welchem die Gegenstände des Anwendungsbereiches, materiell wie immateriell, und die Art und Weise, wie diese benutzt werden, also ihre Umgangsformen, beschrieben werden.

Ebenfalls auf der Ebene des fachlichen Entwurfs anzusiedeln sind die *Systemvisionen* und *Prosys*, in denen die Benutzung des zu entwickelnden Anwendungssystems, also seine Dynamik, beschrieben und mit den zukünftigen Benutzern abgestimmt wird, sowie die *Geschäftsprozeßvisionen*, mit denen die zukünftigen Geschäftsprozesse modelliert werden können.

Eine erste konstruktive Umsetzung des so spezifizierten Systementwurfs stellt die Erstellung entsprechender *Oberflächenprototypen* dar. Im Rahmen des technischen Entwurfes müssen noch weitere Dinge geleistet werden: Das fachliche Klassenmodell muß in das generalisierte Entitymodell der KID-Architektur abgebildet werden. Dazu werden zum einen die Klassen mit ihren Attributen in der bereits gezeigten Form in entsprechende Entityclusterstrukturen überführt, zum anderen die Klassenmethoden in Form entsprechender Funktionssteuerungen abgebildet. Ferner müssen die erstellten Prototypen sukzessive um Funktionalität erweitert werden, bis hin zum voll funktionsfähigen Anwendungssystem.

Parallel zu den genannten Entwicklungsschritten und -dokumenten wird ein *Glossar* erstellt, in dem alle relevanten Begriffe des Anwendungsbereiches, vor allem die im Rahmen der Entwicklung ver-

wendeten Fachtermini der untersuchten Anwendungsdomäne sowie des entwickelten Anwendungssystems, festgehalten und beschrieben werden. Ein solches Glossar dient als „Nachschlagewerk“ für die Entwickler; darüber hinaus bietet es eine weitere Möglichkeit, das fachliche Verständnis der Entwickler vom Anwendungsbereich mit den Anwendern rückzukoppeln und abzustimmen.

Da durch die notwendigerweise sequentielle Vorstellung der unterschiedlichen Abschnitte, die sich in einem Softwareentwicklungsprozeß unterscheiden lassen, möglicherweise der Eindruck entsteht, daß durch das Vorgehensmodell eine ebenso sequentielle Vorgehensweise nahegelegt werden soll, sei an dieser Stelle noch einmal ausdrücklich darauf hingewiesen, daß ein wesentlicher Bestandteil des hier vorgestellten Vorgehensmodells sein zyklischer Aufbau ist. Gerade im Hinblick auf den angesprochenen Lernprozeß, der bei allen an dem Entwicklungsprozeß beteiligten Personen stattfindet und explizit gefördert werden soll, ist eine zyklische und flexible Vorgehensweise erforderlich.

Ich werde nun die unterschiedlichen Bestandteile dieses Vorgehensmodells sowie die dort verwendeten Dokumenttypen im einzelnen detaillierter betrachten.

6.3 Dokumente der Analyse des Anwendungsbereiches

Der erste Schritt bei der Entwicklung eines Anwendungssystems ist zunächst einmal die Analyse des betrachteten Anwendungsbereiches. Zielsetzung dabei ist in erster Linie, daß die Anwendungsentwickler ein zunehmendes Verständnis von diesem erlangen. Dabei geht es nicht nur um das bloße Erkennen der zu erfüllenden Aufgaben, sondern auch darum, zu untersuchen, *wie* diese erledigt werden.

Eine gängige und bewährte Technik für diesen Abschnitt der Systementwicklung ist das Erstellen von Szenarios, ich werde hierauf weiter unten noch genauer eingehen.

6.3.1 Mindmaps

Grundlage von Szenarios sind in der Regel Interviews, welche die Entwickler mit den Anwendern führen. Als unterstützendes Darstellungsmittel für diesen Abschnitt der Anwendungsentwicklung wurden in einigen Projekten der Entitec *Mindmaps* eingesetzt, welche im allgemeinen auf positive Resonanz sowohl bei den Entwicklern als auch bei den interviewten Anwendern stießen.

Ein Mindmap ist ein Mischung aus einer Stichwortliste und einem baumartigen Graphen. In der Mitte des Mindmaps steht in einem Kreis der Name des Anwendungsbereiches, den es darstellt. Von diesem Kreis gehen eine Reihe von Ästen aus, die ihrerseits wiederum über mehrere Stufen verzweigen können. An diese Zweige werden Begriffe aus dem betrachteten Anwendungsbereich geschrieben, die

Äste mit ihren Verzweigungen dienen dabei als Mittel zur Sortierung und Strukturierung der Begriffe. In Abb. 26 ist ein Mindmap aus dem Fallbeispiel des Bibliothekssystems wiedergegeben. In dem zentralen Kreis in der Mitte steht der Name des betrachteten Anwendungsbereiches. Der kleinere Kreis links darunter soll darstellen, daß dieser „Teilbaum“ des Mindmaps aus Platzgründen in ein eigenes Diagramm ausgelagert ist.

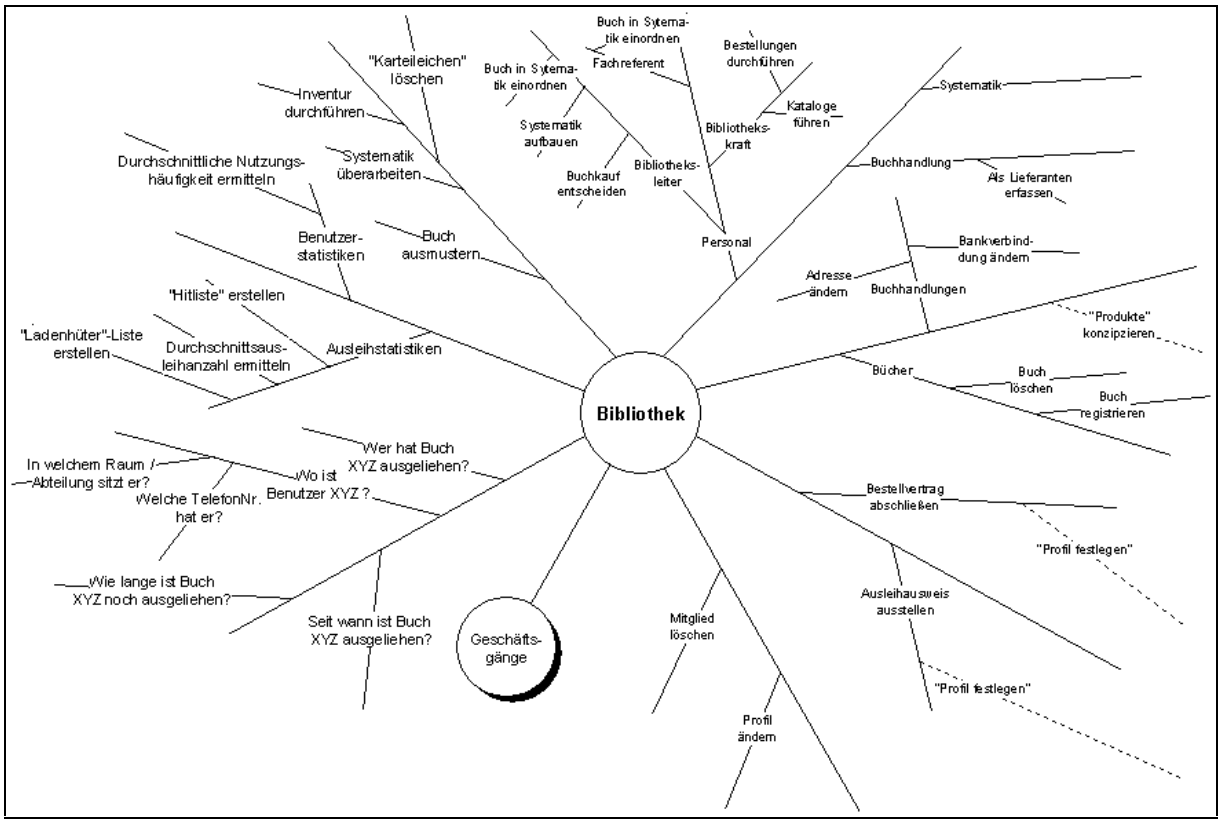

Abb. 26: Mindmap aus dem Fallbeispiel „Bibliothekssystem“.

Das Mindmap soll in dieser Form als Hilfsmittel dienen, die in einem Interview aufgedeckten Dinge, etwa bestehende Aufgaben, anfallende Tätigkeiten oder benötigte Gegenstände, in einer „irgendwie“ strukturierten Form zu sammeln, festzuhalten und zu sortieren. Es gibt den beteiligten Personen die Möglichkeit, das Bild, welches durch das Interview vor dem geistigen Auge entstehen mag, zu vergegenständlichen und zu „ver-gemeinsamen“, es sozusagen zu *kartographieren.*

Die Tatsache, daß die Art und Weise, in welcher dabei eine Strukturierung und Sortierung vorgenommen wird, nicht genau spezifiziert und formalisiert ist, mag auf den ersten Blick nachteilig erscheinen. Dies hat nämlich zur Konsequenz, daß ein solches Mindmap als Dokumentationsmittel, etwa zur Einarbeitung neuer Teammitglieder, nur bedingt geeignet ist, da es in vollem Umfang sicherlich nur den an seinem Erstellungsprozeß beteiligten Personen (-gruppen) verständlich ist. Deswegen wurde ursprünglich auch versucht, eine wesentlich stärker formalisierte Form der Mindmaps einzusetzen, in der genau vorgeschrieben war, wieviele Äste von dem Anwendungskreis in der Mitte

ausgehen dürfen, in wieviele Stufen ein solcher Zweig verzweigen darf, welcher Ast welche Bedeutung hat, und so weiter.

Solch eine starke Formalisierung ermöglicht sicherlich eher, mit Mindmaps eine recht detaillierte Dokumentation des Ist-Zustandes anzufertigen. Die Praxis zeigte jedoch, daß die zu beachtenden Regeln derart komplex waren, daß selbst die Entwickler sie in der Regel nicht vollständig kannten und beherrschten - geschweige denn die am Entwicklungsprozeß partizipierenden Anwender. Dieses führte letztlich dazu, daß sich die hier beschriebene, weniger formale Variante etablierte.

Ich sehe daher den Einsatz des Mindmaps auch weniger im Bereich der dauerhaften Dokumentation, sondern eher, wie oben dargestellt, als direktes Hilfsmittel während der Interviews sowie deren anschließender Auswertung als sinnvoll an. Dabei sehe ich gerade in der Tatsache, daß der Aufbau eines Mindmaps so wenig formalisiert ist, die besondere Stärke dieses Darstellungsmittels; es bietet so den nötigen Freiraum, es flexibel an die jeweiligen Bedürfnisse der Interviewpartner situativ anzupassen und sich auf die wesentlichen, inhaltlichen Dinge des Interviews und weniger auf die Einhaltung der Formalien zu konzentrieren.

6.3.2 Szenarios

Als Mittel, gewonnene Informationen während eines Interviews festzuhalten und zu strukturieren, dient ein solches Mindmap dann als Grundlage für die am Interview beteiligten Entwickler bei der Erstellung der Szenarios und der Geschäftsprozeßbilder, welche auch für die oben genannten Dokumentationszwecke wesentlich besser geeignet sind.

Szenarios dienen der Analyse der Dynamik des Ist-Zustandes. In ihnen beschreiben Entwickler in Prosatext, unter Verwendung der Fachsprache der Anwendung, die Aufgaben und alltäglichen Arbeitssituationen der in dem untersuchten Anwendungsbereich tätigen Personen sowie den Ablauf und Zweck der durchgeführten Handlungen.

Das Erstellen von Szenarios dient im Rahmen des Entwicklungsprozesses im wesentlichen folgenden Zielen:

- Es hilft den Entwicklern, sich in die Fachsprache der jeweiligen Anwendung einzuarbeiten.
- Durch Rückkopplung mit den Anwendern ist es möglich sicherzustellen, daß die Entwickler die betrachteten Sachverhalte und Zusammenhänge richtig erkannt und verstanden haben.
- Den am Entwicklungsprozeß beteiligten Anwendern bieten die Szenarios im Gegenzug die Möglichkeit, ihre eigene Arbeit einmal aus einer anderen Perspektive als ihrer eigenen dargestellt zu bekommen. Dadurch soll unter anderem erreicht werden, daß nicht kleine aber wichtige Details der in den Interviews besprochenen Tätigkeiten übersehen werden, weil sie den Anwendern auf Grund

der Routine bei der Erledigung ihrer alltäglichen Arbeit so selbstverständlich sind, daß sie vergaßen, diese zu erwähnen.

- Im Hinblick darauf, daß eine objektorientierte Modellierung des Anwendungsbereiches vorgenommen werden soll, geben die Szenarios durch die darin benannten Gegenstände, die zur Aufgabenerledigung eingesetzt werden, sowie die Art und Weise, wie sie verwendet werden, wichtige Hinweise auf potentielle fachliche Klassen des Anwendungsbereichs.

Durch eine Reihe von Autor-Kritiker-Zyklen kann so ein gemeinsames Verständnis vom Anwendungsbereich bei allen am Entwicklungsprozeß beteiligten Personen erreicht werden.

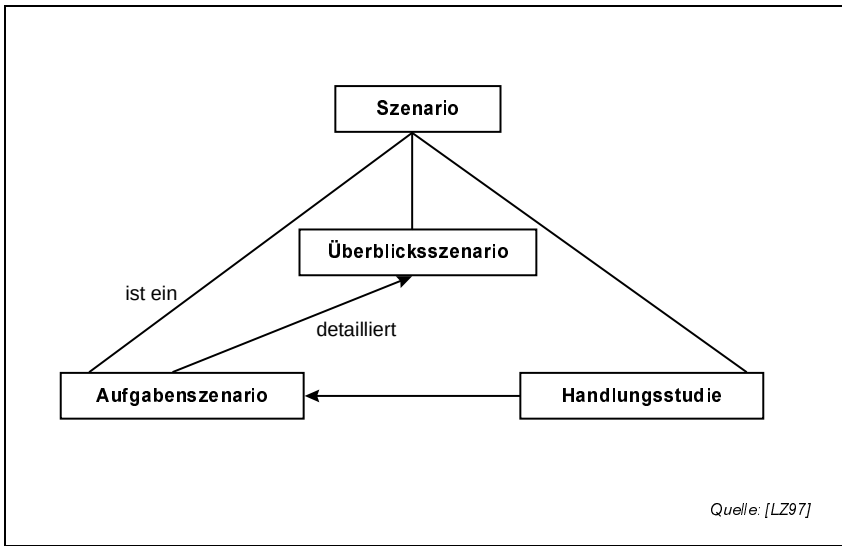

Abb. 27: Szenariotypen und ihre Beziehungen.

In [LZ97] werden drei Szenariotypen unterschieden und zueinander in Beziehung gesetzt, die ich hier kurz vorstellen möchte (siehe Abb. 27).

6.3.2.1 Überblicksszenarios

Die Überblicksszenarios geben, wie der Name schon sagt, einen Überblick über den betrachteten Aufgabenbereich. Inhaltlicher Schwerpunkt dieses Szenariotyps ist das Benennen der Aufgaben. Ein Überblicksszenario soll dem Leser einen Eindruck vermitteln, *was* in dem untersuchten Aufgabenbereich *von wem* getan wird. Ebenfalls an dieser Stelle betrachtet wird das Mengengerüst der untersuchten Aufgaben, auf das Anwendungsbeispiel bezogen also etwa die Fragen, wieviele Bücher am Tag ausgeliehen werden oder wieviele Bestellungen die Verwaltungskraft im Monat zu bearbeiten hat. Sie dienen in erster Linie dazu, einen „Einstieg“ in den Anwendungsbereich zu ermöglichen, und Gruppen zusammengehöriger Szenarios einzuleiten.

6.3.2.2 Aufgabenszenarios

In einem Aufgabenszenario werden die unterschiedlichen, in einem Überblicksszenario bereits angesprochenen, Aufgaben genauer betrachtet. Der Schwerpunkt liegt hierbei auf den zur Erfüllung dieser Aufgaben erforderlichen Tätigkeiten, ein solches Szenario soll dem Leser den Eindruck vermitteln, *wie* eine Aufgabe erledigt wird. Die Beschreibung erfolgt in Form von kleinen szenischen Schilderungen, die beispielhaft einen möglichen Ablauf wiedergeben.

Um die Beschreibung dabei nicht zu unübersichtlich werden zu lassen, kann über Handlungen, die näher betrachtet werden sollen, oder die in unterschiedlichen Aufgabenszenarios vorkommen, abstrahiert werden. Dazu werden solche wiederkehrenden Tätigkeiten benannt und in einem eigenen Szenariotyp, den Handlungsstudien, beschrieben, auf die dann in den verschiedenen Aufgabenszenarios referenziert wird. Dieses Vorgehensweise spart Zeit und Arbeit, da ein und dieselbe Handlung nicht mehrfach beschrieben werden muß, und verhindert zudem, daß die gleiche Handlung auf verschiedene, möglicherweise widersprüchliche Weisen beschrieben wird.

6.3.2.3 Handlungsstudien

Eine solche Handlungsstudie stellt den „detailgenauesten“ Szenariotyp dar. Sie beschreibt nur einen Ausschnitt aus den zur Erledigung einer oder möglicherweise mehrerer Aufgaben nötigen Schritten. Schwerpunkt der Betrachtung bei diesem Szenariotyp liegt darauf, *wie* etwas genau erledigt wird und *womit*, also unter Zuhilfenahme welcher Mittel und Gegenstände, dieses geschieht und wie dabei mit diesen umgegangen wird.

Da im Detail beschrieben wird, in welche Aktionen an denen im Anwendungsbereich vorhandenen Gegenständen die im Rahmen der Aufgabenerledigung anfallenden Tätigkeiten umgesetzt werden, leisten diese Szenarios wertvolle Hilfe sowohl für das Verständnis der fachlichen Konzepte in der Anwendungswelt als auch für den Entwurf des fachlichen Klassenmodells. Schließlich sollte der Fokus bei einer objektorientierten Anwendungsentwicklung in erster Linie auf die Umgangsformen der modellierten Objekte gerichtet sein.

6.3.3 Geschäftsprozeßbilder

Da es Zielsetzung der Anwendungsentwicklung mit EP/KID ist, einheitliche Anwendungssysteme für die unternehmensweite Unterstützung der Geschäftsprozesse eines Unternehmens zu entwickeln, ist es notwendig, über die Grenzen der einzelnen Szenarios hinaus deren Einbettung in den Zusammenhang dieser übergreifenden Geschäftsprozesse zu betrachten.

An der Durchführung der Geschäftsprozesse eines Unternehmens sind eine Vielzahl von Akteuren mit unterschiedlichen Aufgaben und Verantwortlichkeiten beteiligt, die auf verschiedene Weisen mitein-

ander kooperieren. Dabei kann es sich sowohl um „interne“ Akteure handeln, die einen Geschäftsprozeß erbringen, als auch um „externe“ Akteure, beispielsweise die Kunden oder Lieferanten des Unternehmens. Um die unterschiedlichen Beziehungen, die bei Durchführung eines Geschäftsprozesses zwischen diesen Akteuren bestehen, darstellen und analysieren zu können, habe ich den Dokumenttyp der *Geschäftsprozeßbilder* entwickelt.

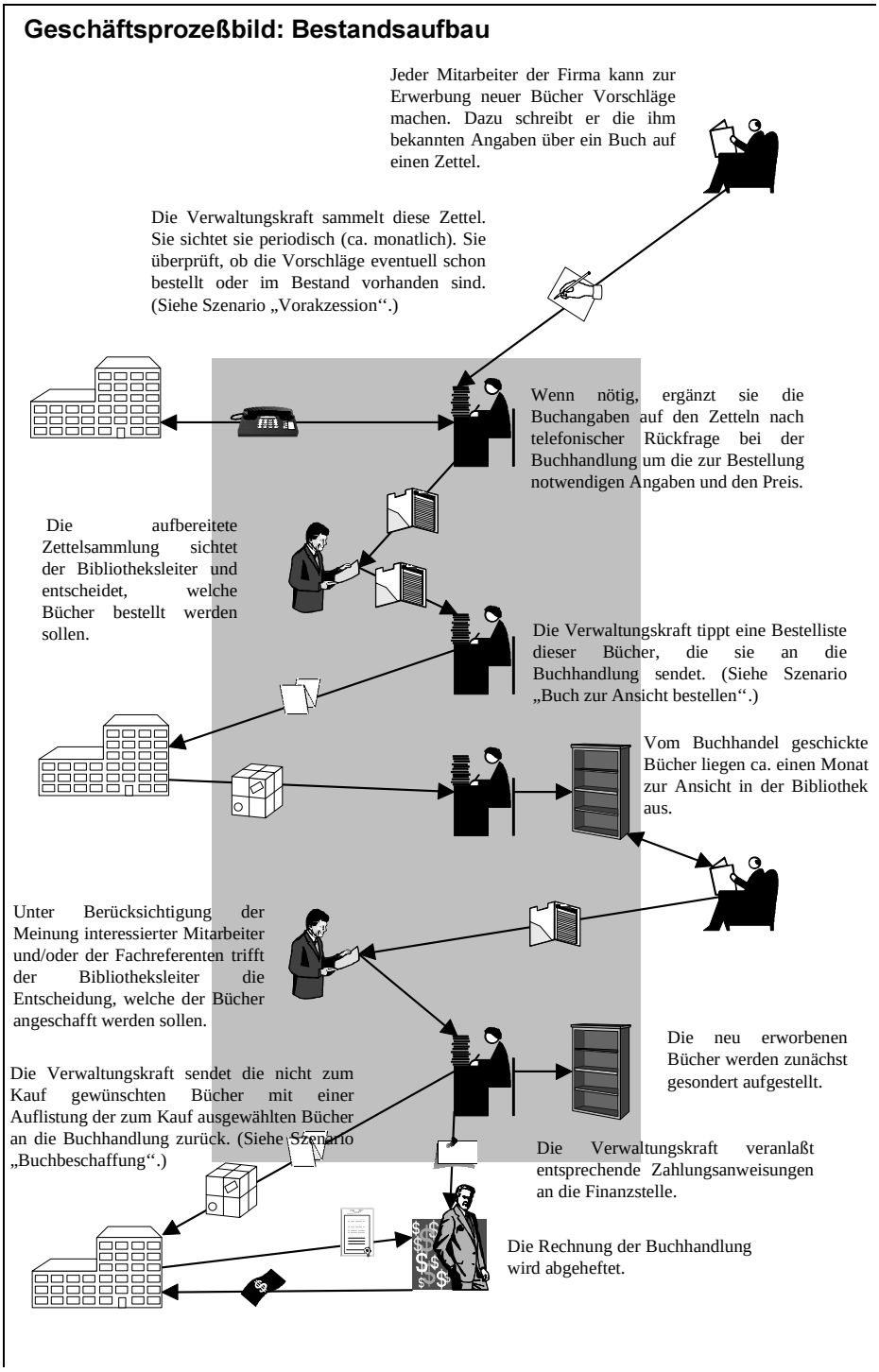

Abb. 28:Geschäftsprozeßbild des Geschäftsprozesses „Bestandsaufbau“.

In Abb. 28 ist ein solches Geschäftsprozeßbild für den Geschäftsgang „Bestandsaufbau“ der Bibliothek aus dem Fallbeispiel zu sehen, es zeigt einen möglichen Ablauf des Geschäftsprozesses.

Die beteiligten Akteure werden durch unterschiedliche Symbole dargestellt. Der Austausch von Gegenständen und Informationen zwischen ihnen wird durch entsprechende gerichtete Pfeile gezeigt, wobei Symbole an den Pfeilen angeben, auf welchem Wege dieser Austausch stattfindet. Durch den grau hinterlegten Kasten werden die Grenzen des betrachteten Systems angezeigt. Akteure innerhalb dieses Kastens sind die internen Akteure, die anderen entsprechend die externen.

Durch textuelle Beschreibungen können die jeweils im Rahmen des Geschäftsprozesses anfallenden Aufgaben kurz beschrieben werden. Diese Beschreibungen können dann durch entsprechende Szenarios weiter „verfeinert“ werden. Auf diese Weise wird die Einbettung der in den Szenarios beschriebenen Aufgaben und Tätigkeiten in den übergeordneten Zusammenhang der Geschäftsprozesse deutlich.

Ich sehe für den Einsatz von Geschäftsprozeßbildern mehrere Zielsetzungen:

- Sie helfen, Zusammenhänge zwischen den in den Szenarios beschriebenen Aufgaben und Tätigkeiten zu klären. Dabei verschaffen sie nicht nur den Entwicklern des Anwendungssystems ein besseres Verständnis des Anwendungsbereichs. Auch den am Entwicklungsprozeß beteiligten Anwendern können sie möglicherweise helfen, eine anschaulichere Vorstellung über die Einbettung ihrer Tätigkeiten und Aufgaben in die übergeordneten Geschäftsprozesse des Unternehmens und ein tieferes Verständnis für die Aufgaben von Mitarbeitern in anderen Tätigkeitsbereichen und Abteilungen zu entwickeln.

- Im Zusammenhang mit der Analyse der Geschäftsprozesse können die Geschäftsprozeßbilder außerdem wertvolle Ansatzpunkte für eine sinnvollen Unterstützung durch ein Anwendungssystem liefern.

- Im Hinblick auf die angestrebte objektorientierte Modellierung des Anwendungsbereiches geben die in den Geschäftsprozeßbildern durch die Symbole an den Pfeilen dargestellten Gegenstände und Informationen, die zwischen den beteiligten Akteuren ausgetauscht werden, einen weiteren Hinweis auf möglich Klassen des fachlichen Klassenmodells.

- Darüber hinaus können Geschäftsprozeßbilder möglicherweise helfen, Schwachstellen in den Abläufen der Geschäftsprozesse des Unternehmens aufzudecken, vor allem, wenn vielleicht erstmalig versucht wird, die Abläufe innerhalb des Unternehmens in einer solchen Form in ihrem Gesamtzusammenhang darzustellen.

6.4 Dokumente des fachlichen Entwurfs

Im Rahmen des fachlichen Entwurfs gilt es zunächst zu untersuchen und festzulegen, wie die Geschäftsprozesse sinnvoll durch Anwendungssysteme unterstützt werden können. Davon ausgehend muß ein fachliches (Klassen-) Modell der Anwendung erstellt werden, das dann in der bereits gezeigten Form in ein KID-Anwendungssystem überführt werden kann.

6.4.1 Fachliches Klassenmodell

Ziel des fachlichen Klassenentwurfes ist es, ein Klassenmodell des Anwendungssystems zu erstellen, das die Gegenstände und Konzepte, die bei der Erledigung der in dem Anwendungsbereich bestehenden Aufgaben verwendet werden, in ein entsprechendes *fachliches Klassenmodell* abbildet. Die in der Anwendungswelt vorgefundenen Begriffshierarchien werden dazu über Vererbungsbeziehungen den Generalisierungs- und Spezialisierungsbeziehungen entsprechend in das Klassenmodell überführt. Im Rahmen eines objektorientierten Entwurfs sollte der Fokus dabei in erster Linie auf die charakteristischen *Umgangsformen* der Objekte und weniger auf die Attribute gerichtet sein.

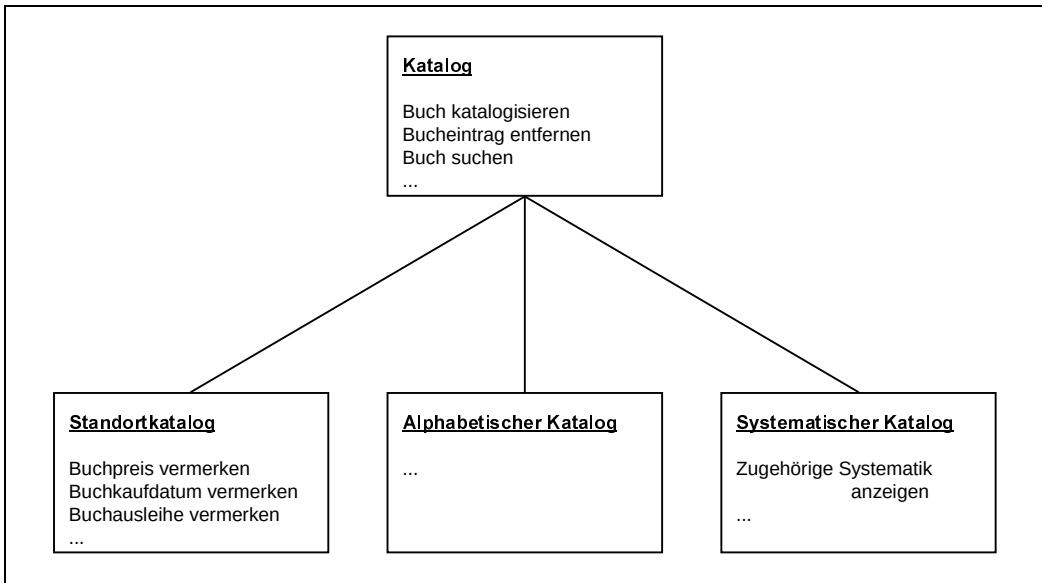

Abb. 29: Fachliches Klassenmodell aus dem Anwendungsbeispiel „Bibliothekssystem".

In Abb. 29 ist ein Ausschnitt des fachlichen Klassenmodells aus dem Anwendungsbeispiel des Bibliothekssystems wiedergegeben. Es zeigt, wie die Begriffshierarchie der unterschiedlichen, im Anwendungsbereich vorgefundenen, Kataloge der Bibliothek in eine entsprechende Klassenhierarchie abgebildet wurde. Diese speziellen Kataloge weisen zunächst ein Reihe von gemeinsamen Umgangsformen auf. Beispielsweise kann man in allen Katalogen Bücher hinzufügen, also ein Buch katalogisieren, oder nach einem Buch suchen. Diese allen Katalogen gemeinen Umgangsformen werden in der Oberklasse Katalog modelliert. Neben diesen weisen die unterschiedlichen Typen von Katalogen noch spezifische Umgangsformen auf. Beispielsweise wird nur in dem Standortkatalog die Ausleihe

eines Buches vermerkt. Diese speziellen Umgangsformen werden in den von der Oberklasse erbenden, spezialisierenden Klassen hinzugefügt.

6.4.2 Systemvisionen

Für einen fachlichen Entwurf der Dynamik des zukünftigen Anwendungssystems bietet sich im ersten Schritt die Erstellung von Systemvisionen und Prosys an. Die Systemvisionen sind ein den Szenarios recht ähnlicher Dokumenttyp. In ihnen beschreiben die Entwickler, ebenfalls in Prosatext, wie sie sich die Benutzung des Anwendungssystems vorstellen. Genau wie in den Szenarios schildern sie dazu, wie die Erledigung ausgewählter Arbeitssituationen mit Hilfe des zu entwickelnden Anwendungssystems aussehen wird. Züllighoven et al. bezeichnen die Systemvisionen daher auch als »„imaginäre“ Szenarios« ([KGZ94], S. 99f).

Als Ausgangspunkt für die Erstellung der Systemvisionen dienen die vorhandenen Szenarios. Die in diesen beschriebenen Arbeitsabläufe sollten zunächst durch entsprechende Systemvisionen „in die Zukunft“ abgebildet werden, d.h. es wird noch einmal beschrieben, wie die in den Szenarios geschilderten Aufgaben und Tätigkeiten erledigt werden, diesmal jedoch unter Zuhilfenahme des (zukünftigen) Anwendungssystems.

Systemvisionen stellen den Übergang von der Analyse zur Konstruktion dar. Sie sollen den an der Entwicklung beteiligten Personen dabei helfen, eine gemeinsame Vision vom zukünftigen System zu erhalten. Um die Vorstellung von Aussehen und Handhabung des entwickelten Anwendungssystems plastischer zu gestalten, sollten die Systemvisionen um Skizzen und Entwürfe der Benutzungsschnittstelle ergänzt werden. Dieses hilft insbesondere, bereits im Vorwege unterschiedliche Vorstellungen von Aussehen und Benutzung des Systems zu erkennen und zu klären.

6.4.3 Prosys

In der *Geschäftsvorfallssteuerung* der KID-Architektur wird eine feste Verknüpfung der *Präsentationsereignisse*, also der Eingaben des Benutzers an der Benutzungsoberfläche, mit dem Aufruf von auszuführenden Funktionssteuerungen und dem Aufblenden einer vom Ergebnis der ausgeführten Funktionssteuerung abhängigen Folgemaske, vorgenommen. Dadurch gestaltet sich die Benutzung eines KID-Anwendungssystems zwangsläufig in Form einer mehr oder weniger sequentiellen Abarbeitung der entsprechenden Eingabemasken.

Da sich in diesem Punkt Anwendungssysteme, die nach WAM entwickelt werden, grundlegend von KID-Anwendungssystemen unterscheiden, findet sich unter den von Züllighoven et al. für die Anwendungsentwicklung mit WAM vorgestellten Dokumenttypen auch keiner, mit dem sich eine solche Benutzungscharakteristik adäquat darstellen läßt. Ich habe daher an dieser Stelle einen weiteren Do-

kumenttyp in das Vorgehensmodell aufgenommen, der sich in der hier vorgestellten Form bereits in KID-Entwicklungsprojekten der Firma Entitec bewährt hat. Ich nenne diesen Dokumenttypen „Prosy“, was für prototypische Systemvision steht.

Ein solches Prosy ist eine Mischung aus einer in Prosatext verfaßten Systemvision und einer Art Maskenablaufplan. In ihm wird gezeigt, welche möglichen Pfade es, je nach Benutzeraktion und Funktionssteuerungsergebnis, bei der Systembenutzung gibt. In Abb. 30 ist ein solches Prosy für den Geschäftsvorfall „Buch katalogisieren“ aus dem Anwendungsbeispiel zu sehen.

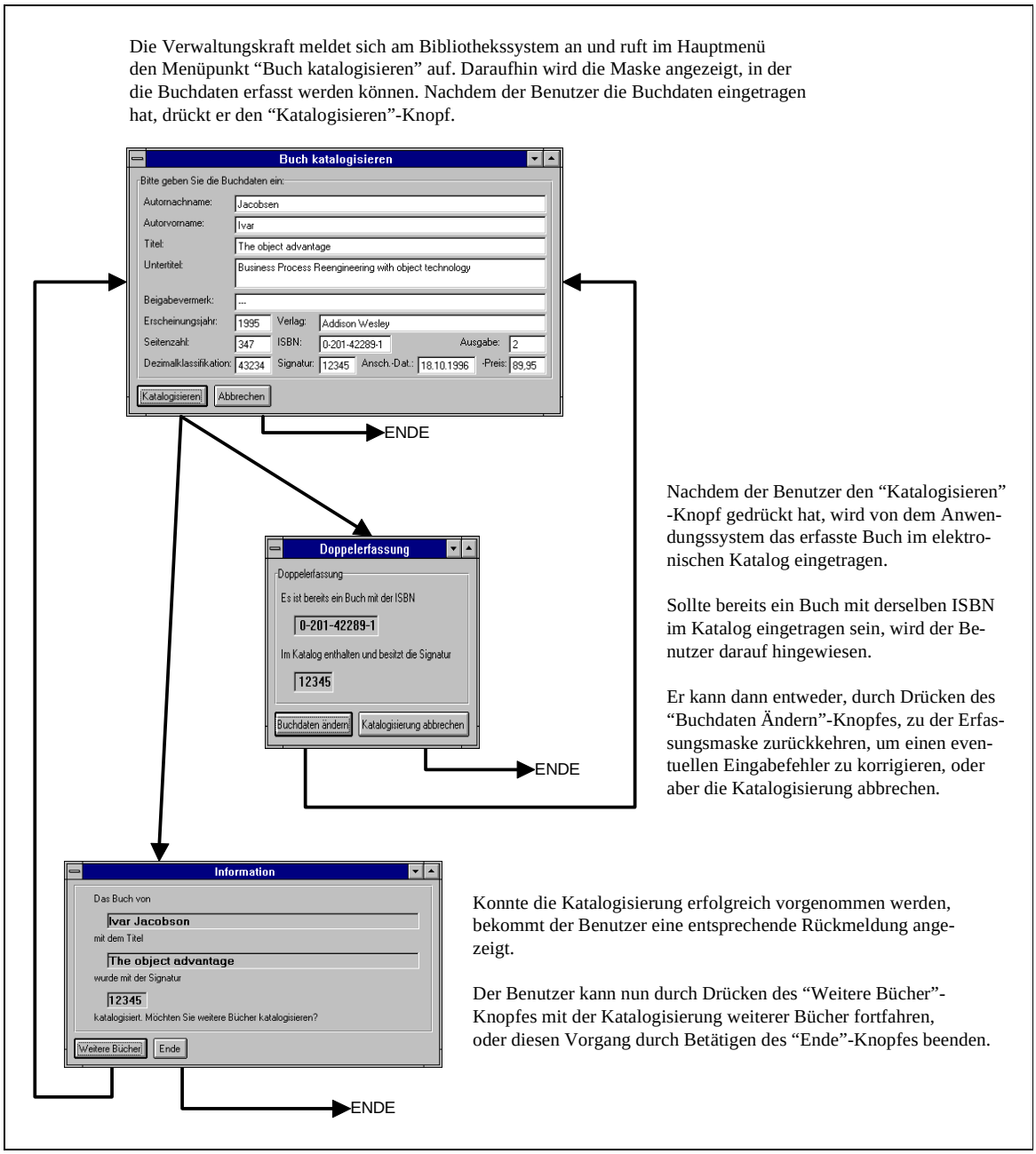

Abb. 30: Prosy für den Geschäftsvorfall „Buch katalogisieren“.

Ein Prosy stellt gewissermaßen eine Erweiterung der im WAM-Ansatz verwendeten Systemvisionen um die für KID-Anwendungssysteme charakteristische Ablaufsteuerung bei der Vorgangsbearbeitung

dar. In diesem Sinne ist die Zielsetzung beim Einsatz dieses Dokumenttyps auch die gleiche, wie bei den Systemvisionen. Sie sollen allen am Entwicklungsprozeß beteiligten Personen helfen, eine Vorstellung davon zu entwickeln, wie sich die Benutzung des zukünftigen Anwendungssystems gestalten wird. Einer solchen gemeinsamen Vision aller Entwickler von der zu entwickelnden Software bedarf es insbesondere, wenn trotz arbeitsteiliger Entwicklung des Anwendungssystems eine konsistente Benutzungsweise erreicht werden soll.

Sowohl die Systemvisionen als auch die Prosys dienen über die genannten Zielen während des Entwicklungsprozesses hinaus auch hervorragend als Ergänzung der Benutzungsdokumentation des Anwendungssystems, da sie in anschaulicher Weise die Benutzung des Systems zeigen und so eine gute Unterstützung bei der Einarbeitung in die Systembenutzung darstellen.

6.4.4 Geschäftsprozeßvisionen

Da der Einsatz eines Informations- und Anwendungssystems zur Unterstützung von Geschäftsprozessen zwangsläufig dazu führt, daß diese verändert werden, ist es erforderlich, diese Wirkung auf sie näher zu untersuchen. Dazu schlage ich den Dokumenttyp *Geschäftsprozeßvision* vor.

Analog zu dem Dokumentenpaar Szenario und Systemvision, stellen die Geschäftsprozeßvisionen das Pendant zu den Geschäftsprozeßbildern dar. In ihnen antizipieren die Entwickler den Ablauf der Geschäftsprozesse unter Berücksichtigung des Einsatzes eines unterstützenden Softwaresystems. Durch diesen Dokumenttyp ist es möglich, die Auswirkungen des Softwareeinsatzes auf die Geschäftsprozesse zu visualisieren und diese so diskutierbar zu machen.

In Abb. 31 ist eine Geschäftsprozeßvision für den in Abb. 28 dargestellten Geschäftsprozeß aus dem Fallbeispiel wiedergegeben. Wie hier bereits zu sehen ist, sind Geschäftsprozeßvisionen vom Inhalt und Aufbau mit den Geschäftsprozeßbildern identisch. Durch die geschwungenen Pfeile wird gezeigt, daß dieser Informationsaustausch auf elektronischem Wege durch den zukünftigen Einsatz des Anwendungssystems realisiert werden soll.

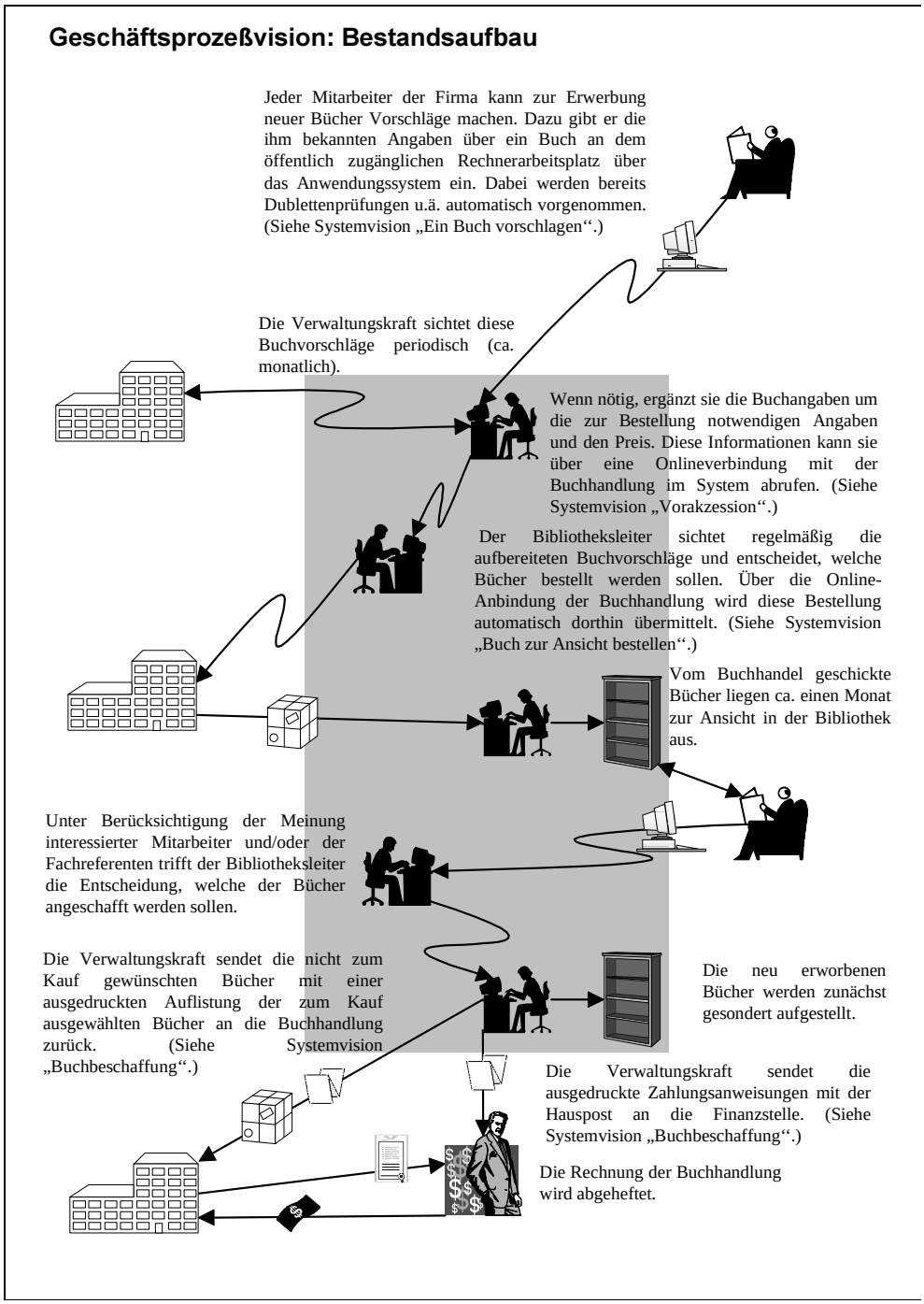

Abb. 31: Geschäftsprozeßvision des Geschäftsprozesses „Bestandsaufbau“.

Wie bereits an diesem Beispiel zu sehen ist, kann der Einsatz eines unterstützenden Softwaresystems, über die Veränderung der zu einer Aufgabenerfüllung erforderlichen Tätigkeiten hinaus, zu einer Veränderungen im Ablauf des Geschäftsprozesses und der Aufgaben der beteiligten Akteure führen. So ist es in dem hier gezeigten Beispiel durch die Online-Anbindung der Buchhandlung an das Softwaresystem der Bibliothek nicht mehr notwendig, daß die Verwaltungskraft, nachdem der Bibliotheksleiter entschieden hat, welche Bücher zur Ansicht bestellt werden sollen, eine Bestelliste tippt und diese zur Buchhandlung schickt. Statt dessen ist der Bibliotheksleiter jetzt selbst dafür verantwortlich, daß eine entsprechende Bestellung an die Buchhandlung übermittelt wird.

Die Geschäftsprozeßvisionen dienen dazu, die Auswirkungen der Einführung eines Softwaresystems auf die Arbeitsgestaltung bereits im Entwurf abschätzbar zu machen. Wie sich die Benutzung des Anwendungssystems im einzelnen gestalten wird, kann auch hier wieder durch die entsprechenden Systemvisionen und Prosys detaillierter betrachtet werden. Umgekehrt wird durch die Geschäftsprozeßvisionen die Einbettung der Systemvisionen und Prosys in den übergeordneten Gesamtzusammenhang, ähnlich wie bei den Szenarios und den Geschäftsprozeßbildern, transparenter.

6.4.5 Prototypen

Ein erster Schritt zur konstruktiven Umsetzung der mit Hilfe der bisher vorgestellten Dokumenttypen formulierten Spezifikation des Anwendungssystems stellt die Entwicklung von Prototypen dar. So anschaulich Systemvisionen und Prosys die Dynamik, insbesondere die der Benutzung des Systems, verdeutlichen mögen, es sind nichts desto trotz „nur“ Dokumente; sie können lediglich eine Vorstellung davon vermitteln, wie sich die tatsächliche Benutzung des Systems gestalten wird. Prototypen hingegen bieten die Möglichkeit, Dynamik und Handhabbarkeit des zu entwickelnden Anwendungssystem erfahrbar zu machen.

Wie ich bereits bei der Vorstellung der KID-Architektur angesprochen habe, bin ich der Meinung, daß diese sich sehr gut für die Anwendungsentwicklung mit Hilfe von inkrementellem Prototyping eignet. Bereits zur Erstellung der Systemvisionen und Prosys können mit dem „Vodka“[26], dem Werkzeug der KID-Workbench zur Erstellung der Benutzungsschnittstelle, die Masken erzeugt und in den Dokumenten verwendet werden.

Aufbauend darauf kann ohne großen Aufwand ein Prototyp ohne Funktionalität, also ein reiner *Oberflächenprototyp*, erstellt werden. Dazu müssen lediglich die Vorfallssteuerungen entsprechend den Prosys über die KID-Workbench angelegt werden. Konkret bedeutet dies, daß die Präsentationsereignisse der mit dem Vodka erstellten Masken in den Steuerungstabellen mit den entsprechenden Folgemasken verknüpft werden müssen. Ein solcher Oberflächenprototyp stellt dann eine sehr gute Diskussionsgrundlage für den weiteren Entwurf dar.

Die KID-Architektur ermöglicht es in sehr einfacher Weise, diesen Prototypen sukzessive mit Funktionalität anzureichern. Dazu müssen lediglich die einzelnen Funktionssteuerungen, ebenfalls mittels der KID-Workbench, implementiert und an entsprechender Stelle in die Vorfallssteuerung „eingehängt“ werden. Auf diese Weise kann das gesamte KID-Anwendungssystem inkrementell entwickelt werden.

[26] VODKA steht für Visueller Oberflächen-Designer für KID-Anwendungen.

6.5 Glossar

In einem Glossar werden die in dem Anwendungsbereich vorkommenden (Fach-) Begriffe und Konzepte sowie deren Zusammenhang dokumentiert. Dementsprechend sollte bereits während der Analyse damit begonnen werden, ein Glossar zu erstellen. Allerdings umfaßt ein Glossar neben den bereits im Anwendungsbereich existierenden Begriffen auch solche, die möglicherweise erst durch den Einsatz des Anwendungssystems hinzukommen.

Wesentlicher Bestandteil eines Glossareintrages ist es, den Sinn und Zweck des beschriebenen Objektes zu erklären. In Abb. 32 ist ein Beispiel für einen Glossareintrag aus dem Anwendungsbeispiel Bibliothekssystem gezeigt.

Vorakzession	Vorbereitung der Bestellung eines Buches. Im Rahmen der Vorakzession wird festgestellt, ob ein Buch bereits im Buchbestand vorhanden oder bestellt ist. Die Vorakzession wird daher auch als ->"Dublettenprüfung" bezeichnet.

Abb. 32: Beispiel für einen Glossareintrag.

Genau wie die Szenarios basieren auch Glossare auf mit den Anwendern geführten Interviews. Sie werden ebenfalls von den Entwicklern erstellt und anschließend mit den andern am Entwicklungsprozeß beteiligten Personengruppen diskutiert. Sie dienen somit genau wie die Szenarios sowohl dazu, das Verständnis der Entwickler vom betrachteten Anwendungsbereich durch Rückkopplung mit den Anwendern zu überprüfen als auch, den Mitarbeitern der Fachabteilungen die Möglichkeit zu geben, die Bedeutung der von ihnen verwendeten (Fach-) Begriffe und Konzepte zu reflektieren und abzugleichen.

Wichtig für den Nutzen, den ein Glossar erbringen kann, ist, daß die Erklärungen und Definitionen der enthaltenen Begriffe keinesfalls ohne Rückkopplung mit den Anwendern aus irgendwelchen bereits vorhandenen Dokumenten wie etwa Arbeitsplatzbeschreibungen oder Nachschlagewerken übernommen werden. Vielmehr müssen alle Glossareinträge kooperativ abgestimmt werden, damit sichergestellt ist, daß das im Anwendungsbereich *aktuell* vertretene Verständnis der verwendeten Begriffe wiedergegeben wird. Nur so kann ein Glossar den bereits erwähnten Lernprozeß fördern und zudem verhindern helfen, daß falsche Entwurfsentscheidungen aufgrund von Mißverständnissen bei der Begriffsbildung getroffen werden.

Ein wesentlicher Unterschied zu den Szenarios ist die Tatsache, daß die Erstellung von Glossaren nicht auf den Bereich der Analyse beschränkt ist; vielmehr werden sie während der gesamten Entwicklung und unter Umständen noch darüber hinaus fortgeführt und gepflegt, da sie nicht nur Dokumente des Entwicklungsprozesses sind, sondern auch Grundlage und Bestandteil der Benutzungsdokumentation des entwickelten Anwendungssystems sein sollten.

Darüber hinaus kann es sogar wünschenswert sein, projektübergreifend ein unternehmensweites Glossar zu erstellen. Dadurch wird zum einen die Arbeit zukünftiger Entwicklungsprojekte erleichtert, da auf bereits konsolidierte Kenntnisse zurückgegriffen werden kann. Zum anderen wird verhindert, daß innerhalb eines Unternehmens derselbe Begriff für unterschiedliche fachliche Konzepte verwendet wird, oder dieses zumindest dokumentiert, in Fällen, wo es nicht verhindert werden kann.

6.6 Steuerung und Kontrolle des Entwicklungsprozesses

Damit der Entwicklungsprozeß auch zu dem gewünschten Ziel führt, nämlich unter Vorgabe bestimmter äußerer Rahmenbedingungen, wie etwa zur Verfügung stehender Ressourcen oder einzuhaltender Termine, ein Anwendungssystem zur Unterstützung eines bestimmten Anwendungsbereiches zu entwickeln, muß dieser Prozeß gesteuert und kontrolliert werden. Dazu bedarf es geeigneter Hilfsmittel und Techniken, die sowohl eine qualitative Steuerung als auch eine zeitliche Fortschrittskontrolle ermöglichen. Wichtig ist, daß auch diese steuernden und kontrollierenden Maßnahmen auf das zyklische und evolutionäre Vorgehensmodell abgestimmt sind und eine solche Vorgehensweise nicht behindern oder gar unmöglich machen.

6.6.1 Qualitätssicherung durch Referenzlinien

Eine Möglichkeit, eine auf das hier vorgestellte Vorgehensmodell und die darin verwendeten Dokumenttypen anwendbare Qualitätssicherung durchzuführen, stellt das Aufstellen von sogenannten *Referenzlinien*[27] dar.

Eine Referenzlinie definiert einen zu erreichenden Zustand des Entwicklungsprozesses durch Angabe von Qualitätsmerkmalen der im Rahmen der Entwicklung erstellten Dokumente, sowohl bezogen auf ihren inneren Aufbau als auch auf ihre Abhängigkeiten untereinander. Sie gibt ferner an, wer für die Einhaltung dieser Qualitätsmerkmale verantwortlich ist und wie diese bewertet werden können. In Abb. 33 sind beispielhaft einige mögliche Referenzlinien wiedergegeben.

[27] Vgl. dazu beispielsweise auch [KGZ94] oder [Flo91].

Name der Referenzlinie	Dokumentenzustand	Verantwortlich
Szenarios 1	Erste von den Benutzern akzeptierte Version der Szenarios	Prototyping-Gruppe
Systemvision 1	Systemvisionen enthalten erste Skizzen der Benutzungsschnittstelle	Oberflächenentwickler
Systemvision 2	Systemvisionen decken vollständig alle in Szenarios beschriebenen Vorgänge ab	Entwickler
Prototyp 1	Oberflächenprototyp deckt alle Systemvisionen ab	Prototyping-Gruppe
Glossar 1	Glossar stimmt mit Szenarios überein	Alle Entwickler

Abb. 33: Referenzlinien zur Qualitätskontrolle.

Durch das Aufstellen solcher Referenzlinien ist es möglich, Projektziele, -aktivitäten und Verantwortlichkeiten festzuschreiben, ohne Vorgaben für die Reihenfolge der zum Erreichen der angegebenen Qualitätsmerkmale notwendigen Schritte zu machen. Diese Vorgehensweise steht in Einklang mit der oben formulierten Forderung, kein antizipatives, sondern vielmehr ein situativ adaptierbares Vorgehensmodell aufzustellen.

6.6.2 Fortschrittskontrolle durch Projektstadien

Außer der qualitativen Kontrolle der im Entwicklungsprozeß erstellten Dokumente ist natürlich in jedem kommerziellen Projekt eine Fortschrittskontrolle zur Einhaltung terminlicher Rahmenbedingungen vonnöten.

Mit Hilfe der aufgestellten Referenzlinien kann dazu eine Folge von *Projektstadien* definiert werden. Diese stellen eine Einteilung des Entwicklungsprozesses in zeitliche Abschnitte dar, die durch wesentliche Ereignisse im Projektablauf abgeschlossen sind. Zu den einzelnen Projektstadien kann angegeben werden, bis zu welchem Termin sie zu erreichen sind und welche Referenzlinien erfüllt sein müssen, damit sie erreicht sind. In Abb. 34 sind Beispiele für solche Projektstadien wiedergegeben.

03/98 Akzeptanz durch das Management

Zustand:	Demonstrationsprototyp für ausgewählte Anwendungsfälle.
Referenzlinien:	Prototyp 3, Systemvision 1
Zielgruppe:	Entwicklerteam und DV-Management

07/98 Demonstration vor Anwendern

Zustand:	Ablauffähiger Oberflächenprototyp zur Präsentation der Konzepte des Bibliothekssystems
Referenzlinien:	Prototyp 4, Szenarios 3, Glossar 2
Zielgruppe:	Management, Bibliotheksbenutzer

10/98 Rückkopplung mit Anwender

Zustand:	Neuer Prototyp mit erweiterter Funktionalität und überarbeiteter Benutzungsschnittstelle
Referenzlinien:	Prototyp 5, Szenarios 4, Glossar 3, Systemvisionen 2
Zielgruppe:	Bereits angesprochene Benutzer

Abb. 34: Projektstadien zur Fortschrittskontrolle.

Eine derartige Verbindung von Referenzlinien und Projektstadien ermöglicht, wie eingangs gefordert, eine Vorgehensweise, die keinen Ablauf des Entwicklungsprozesses vorschreibt sondern sich an den Qualitätsmerkmalen der erstellten Dokumente orientiert und berücksichtigt gleichzeitig die Einhaltung äußerer, terminlicher Rahmenbedingungen.

7 Bewertung und Ausblick

Mit Blick auf die Vorstellung und Bewertung anderer Ansätze im ersten Teil dieser Arbeit ist sicherlich festzustellen, daß einige der dort angeführten Kritikpunkte auch für die hier entwickelte KID-Methode gelten. Dieses gilt insbesondere für die durch die KID-Architektur vorgegebene, ablaufsteuernde Benutzungscharakteristik von KID-Anwendungssystemen, die sich in der Form auch in den anderen vorgestellten Ansätzen finden läßt.

Ich habe aber mit dem hier vorgeschlagenen Vorgehensmodell den Versuch unternommen, so gut es in diesem durch die KID-Architektur vorgegebenen Rahmen möglich ist, die am Arbeitsbereich Softwaretechnik und auch von mir vertretenen (Wert-) Vorstellungen in die Entwicklung von Anwendungs- und Informationssystemen mit EP/KID einfließen zu lassen. Dieses gilt insbesondere für die grundlegende Sichtweise, was den Stellenwert der menschlichen Arbeit und die Qualifikation der tätigen Menschen und zukünftigen Benutzer des zu entwickelnden Anwendungssystems anbelangt. Diese Sichtweise erfordert ein Vorgehen bei der Anwendungsentwicklung, das eine hohe Partizipation der künftigen Benutzer des entwickelten Anwendungssystems vorsieht.

Eine solche Beteiligung der Anwender wird zwar beispielsweise auch in der von Scheer vorgestellten ARIS-Methode explizit gefordert, ist meiner Auffassung nach aber, aufgrund der dort verwendeten, stark formalisierten Darstellungsmittel, nur begrenzt umsetzbar. Ich habe daher Dokumenttypen vorgeschlagen, die es ermöglichen sollen, die Anwenderpartizipation für alle Beteiligten möglichst befriedigend und gewinnbringend zu gestalten.

Bezogen auf den Ansatzumfang läßt sich feststellen, daß mit der KID-Architektur der heutigen Form eine Abbildung der unternehmensweiten Geschäftsprozesse nicht unterstützt wird. Es können bislang lediglich Informationssysteme entwickelt werden, die zwar auf einem unternehmensweit einheitlichen Geschäftsobjektmodell beruhen, eine Unterstützung der übergreifend ablaufenden Geschäftsprozesse und der in diesem Zuge anfallenden Kooperationen ist jedoch bisher nicht möglich.

Durch die für das Vorgehensmodell entwickelten Dokumenttypen Geschäftsprozeßbild und Geschäftsprozeßvision wird es nun möglich, den Betrachtungshorizont in diese Richtung auszudehnen. Eine entsprechende Erweiterung der KID-Architektur in einer Form, wie sie beispielsweise von Gryczan et al. vorgeschlagen wurde, halte ich für möglich und erstrebenswert.

8 Schlußwort

Anliegen dieser Arbeit war es, im wesentlichen zwei Themenschwerpunkte zu behandeln.

Im ersten Teil der Arbeit habe ich versucht, einen Überblick über die vielgestaltige und weit gefächerte Themenlandschaft, die durch Begriffe wie *Geschäftsprozeßmodellierung, Workflow-Management,* oder *unternehmensweite Anwendungs- und Informationssystementwicklung* aufgespannt wird, zu geben.

Dieser Themenbereich sollte vor allem vor dem Hintergrund der am Arbeitsbereich Softwaretechnik des Fachbereichs Informatik der Universität Hamburg vertretenen Sichtweise beleuchtet werden. Dazu habe ich diesen Standpunkt zunächst kurz umrissen. Anschließend habe ich einige unterschiedliche Ansätze aus dem genannten Themenkreis vorgestellt und kritisch bewertet. Aufgrund dieser angestellten Betrachtungen habe ich dann einige Bewertungs- oder Unterscheidungskriterien herausgearbeitet, anhand derer sich Methoden und Produkte, insbesondere im Hinblick auf die hier vertretene Sichtweise, kategorisieren lassen.

Im zweiten thematischen Schwerpunkt der Arbeit habe ich das Produkt EP/KID der Unternehmensberatung Entitec vorgestellt, welches ebenfalls in dem betrachteten Themenbereich angesiedelt ist. An diesem Produkt habe ich untersucht, inwieweit hiermit eine objektorientierte Anwendungsentwicklung nach den vorgestellten Erwartungen und Ansprüchen an eine solche Entwicklung möglich ist.

Ich habe aufgezeigt, daß es mit gewissen Einschränkungen, die in der zugrundeliegenden Technik und Architektur des Produktes begründet sind, sehr wohl möglich ist, eine objektorientierte Entwicklung durchzuführen. Allerdings sind einige Abstriche gegenüber der am Arbeitsbereich Softwaretechnik vertretene Sichtweise zu machen, insbesondere was die mit KID in der Form nicht realisierbare reaktive „Werkzeug-Material-Charakteristik“ von Anwendungssystemen anbelangt.

Im weiteren Verlauf habe ich ein Vorgehensmodell und mit diesem verbundene Dokumenttypen vorgestellt, das ich für die Softwareentwicklung mit EP/KID entwickelt habe. Dieses Vorgehensmodell soll ermöglichen, eine evolutionäre und partizipative Anwendungsentwicklung zu betreiben. Die in diesem Vorgehensmodell verwendeten Dokumenttypen sollen dabei insbesondere den sich aus einer solchen Vorgehensweise ergebenden Anforderungen gerecht werden und die angestrebten Lern- und Kommunikationsprozesse unterstützen und fördern.

In Teilen habe ich einige der von Züllighoven et al. für die Anwendungsentwicklung mit WAM vorgeschlagenen Dokumenttypen übernommen, da diese sich für eine solche Vorgehensweise sehr gut eignen und sich ihr Einsatz in der Praxis bereits bewährt hat. Darüber hinaus habe ich mit den Prosys einen weiteren Dokumenttypen entwickelt, den ich aufgrund der genannten charakteristischen Unter-

schiede zwischen KID-Anwendungssystemen und solchen, die nach WAM entwickelt werden, für erforderlich gehalten habe.

Als ergänzenden Dokumenttypen habe ich die Mind Maps vorgeschlagen. Sie sollen in der hier gezeigten Form den Entwicklern wie den am Entwicklungsprozeß partizipierenden Anwendern gleichermaßen helfen, sich bereits zum Zeitpunkt der geführten Interviews ein gemeinsames Bild vom Anwendungsbereich zu machen.

Ferner habe ich die Geschäftsprozeßbilder und die Geschäftsprozeßvisionen als weitere Dokumenttypen vorgestellt, die eine Darstellung und Modellierung der Geschäftsprozesse eines Unternehmens ermöglichen. Dieses ist insbesondere im Hinblick auf den vorgeschlagenen weiteren Ausbau der EP/KID-Architektur zur Unterstützung der unternehmensweit ablaufenden Geschäftsprozesse von Bedeutung. Hierbei handelt es sich um einen Themenbereich, der nach wie vor im Mittelpunkt sowohl wissenschaftlichen als auch wirtschaftlichen Interesses steht und somit gute Ausgangspunkte für weiterführende Arbeiten im Bereich von EP/KID darstellen könnten.

Anhang A Vorstudie über ein Bibliothekssystem

Vorbemerkungen

Als Fallstudie für die Übungen dient ein dialogorientiertes Bibliothekssystem zur Unterstützung von Verwaltungs- und Informationsaufgaben einer Fachbibliothek. In Aufgabe 1 soll dazu eine aufgabenbezogene Anforderungsermittlung, in Aufgabe 2 ein Softwareentwurf nach den Maßgaben von STEPS (/1/) erarbeitet werden.

Die Vorstudie dazu baut auf einem an der Technischen Universität Berlin von Horst Hunger, Frank Itter, Richard Künzig und Zafer Tosun im Rahmen ihrer Diplomarbeiten entwickelten Text auf. Dort diente das Bibliothekssystem zur Erprobung verschiedener Methoden (/2/, /3/, /4/ und /5/ sind auch an der Bibliothek des Fachbereichs Informatik in Hamburg verfügbar). Die Erfahrungen, die dabei gemacht wurden, sind in /6/ und /7/ veröffentlicht.

Zur Einarbeitung in das Problemfeld wurde ein Standardbuch für Bibliothekare verwendet (/8/).

Gegenstand der Vorstudie ist die bibliotheksinterne und die ihr übergeordnete Organisationsform. Die beschriebenen Sachverhalte sind fiktiv. Sie sind jedoch an die Verhältnisse von Fachbibliotheken in Firmen, Behörden und Schulen angelehnt. Der Text geht von einer (Software-) Firma aus. Folgenden Aspekte werden behandelt:

1. Die Definition der Bibliotheks- und Umfeldorganisation

2. Die Beschreibung der Einsatzmöglichkeiten der EDV

Der Ausgangspunkt für die Übungen ist damit vergleichbar mit der Entwicklung von Standardsoftware: Für die Systementwickler bleiben die Systemanwender anonym. Ihr Bedarf muß durch sorgfältige Erarbeitung des Anwendungsgebiets antizipiert werden. Entscheidungsalternativen können nicht durch Rückfragen geklärt, sondern nur an denkbaren Anwendungsfällen studiert werden.

Der Begriff Vorstudie wurde bewußt alternativ zu den in der Softwaretechnik gebräuchlichen Namen für Dokumente gewählt, da auf der Grundlage dieses Textes zunächst nur ein gemeinsames Verständnis der Übungsgruppe hergestellt werden soll.

1. Annahmen zur Organisation der Bibliothek

1.1 Einordnung der Bibliothek in die Firma

Aufgabe der Fachbibliothek ist das Sammeln, Erschließen und Benutzbarmachen von Büchern für die Mitarbeiter der Firma. Damit nimmt sie die Stellung eines firmeninternen Dienstleistungsbetriebes ein. Die fachliche Ausrichtung bestimmt sich nach den Fachgebieten der einzelnen Firmenabteilungen.

Die Firma besteht aus autonomen Abteilungen. Eine Abteilung ist die Zusammenfassung von Mitarbeitern, die einen bestimmten Bereich gemeinsam bearbeiten. Jeder Abteilung steht ein Abteilungsleiter vor. Die Bibliothek liegt organisatorisch auf der Ebene der Abteilungen. Es existiert ein Rahmenplan über den Einsatz der Finanzmittel zum Auf-/Ausbau des Buchbestandes.

Benutzer sind alle Mitarbeiter der Firma.

1.2. Bibliothekspersonal

Bibliotheksleiter

Ein Abteilungsleiter übernimmt die Leitung der Bibliothek. Er ist der Geschäftsleitung für den Mitteleinsatz unter Einhaltung des Rahmenplanes für die Neuanschaffung verantwortlich. Er hat bibliothekarische Kenntnisse und übernimmt verantwortlich die Katalogführung (speziell Einordnung in die Systematik). Er ist Gesprächspartner für die Buchhaltung.

Fachreferenten

Alle Abteilungsleiter sind Fachreferenten für die Bibliothek. Als Spezialisten in ihrem Gebiet werden sie vom Bibliotheksleiter bei der Anschaffung neuer Bücher und deren Einordnung in die Systematik zur Beratung hinzugezogen.

Bibliotheksverwalter

Anfallende Verwaltungsaufgaben (Katalogführung, Bestellungen usw.) führt eine Verwaltungskraft aus.

Alle Arbeiten in der Bibliothek übernehmen die Personen neben ihrer eigentlichen Tätigkeit in der Firma. Der Arbeitsaufwand für die Bibliotheksaufgaben ist nicht bekannt.

1.3. Buchbestand

Die Bibliothek besitzt ca. 2000 Bücher (einschließlich Mehrfachexemplare). Daneben gibt es Zeitschriften und Projektberichte.

1.4. Räumliche Gegebenheiten

Der Buchbestand befindet sich in einem für alle Mitarbeiter der Firma zugänglichen Raum (Freihandaufstellung). Es existiert ein gemeinsamer Arbeitsplatz für das Personal. Er wird ausschließlich für die Bibliothekstätigkeiten benutzt.

1.5. Geschäftsgang

Der Geschäftsgang der Bibliothek wird angelehnt an die Begriffsbildung in /8/ beschrieben (Seiten 35ff, 66ff, 138ff). Die Zeitschriften sind nicht erfaßt und dürfen aus diesem Grund auch nicht ausgeliehen werden. Sie werden daher auch nicht im Geschäftsgang beschrieben. Die Projektberichte werden wie Bücher behandelt.

1.5.1. Bestandsaufbau (Erwerbung)

Auswahl:

Jeder Mitarbeiter der Firma kann zur Erwerbung neuer Bücher Vorschläge machen. Dazu schreibt er die ihm bekannten Angaben über ein Buch auf einen Zettel. Die Verwaltungskraft sammelt diese Zettel. Sie sichtet sie periodisch (ca. monatlich). Sie überprüft, ob die Vorschläge eventuell schon bestellt oder im Bestand vorhanden sind. Diese der Bestellung vorausgehende Überprüfung nennt man Vorakzession (zu einer Detaillierung dieses Begriffs siehe /8/). Wenn nötig, ergänzt sie die Buchangaben auf den Zetteln nach telefonischer Rückfrage bei der Buchhandlung um die zur Bestellung notwendigen Angaben und den Preis. Die aufbereitete Zettelsammlung sichtet der Bibliotheksleiter und entscheidet, welche Bücher bestellt werden sollen. Alle Bücher werden zunächst zur Ansicht bestellt. Die Verwaltungskraft tippt eine Bestelliste dieser Bücher, die sie an die Buchhandlung sendet. Vom Buchhandel geschickte Bücher liegen ca. einen Monat zur Ansicht in der Bibliothek aus. Unter Berücksichtigung der Meinung interessierter Mitarbeiter und/oder der Fachreferenten trifft der Bibliotheksleiter die Entscheidung, welche der Bücher angeschafft werden sollen.

Beschaffung:

Die Verwaltungskraft sendet die nicht zum Kauf gewünschten Bücher mit einer Auflistung der zum Kauf ausgewählten Bücher an die Buchhandlung zurück. Sie veranlaßt entsprechende Zahlungsan-

weisungen an die Finanzstelle. Die Rechnung der Buchhandlung wird abgeheftet. Die neu erworbenen Bücher werden zunächst gesondert aufgestellt. Pro Jahr werden ca. 100 bis 150 Bücher angeschafft.

1.5.2. Bestandserschließung (Katalogisierung)

Aufgabe der Katalogisierung ist die formale und inhaltliche Erschließung des Buchbestandes. Es existieren drei verschiedene Katalogarten: Standort-, Alphabetischer und Systematischer Katalog.

Kataloge

Standortkatalog:

Er verzeichnet die Bücher in der Reihenfolge, in der sie aufgestellt sind. Er ist damit ein genaues Spiegelbild der Aufstellung des Buchbestandes. Die Aufstellung der Bücher erfolgt nach Zugang (mechanische Aufstellung).

Unter mechanischer Aufstellung versteht man das Aufstellen der Bücher in der zufälligen Reihenfolge, in der sie in die Bibliothek kommen, also ohne Rücksicht auf den Inhalt (/8/ S. 142). Die Signatur, d.h. die Standortnummer legt den Standort des Buches innerhalb des Bestandes fest. Dazu wird eine vierstellige Zahl verwendet.

Der Standortkatalog dient als Hilfsmittel für die Signaturgebung und für die jährlich stattfindende Revision des Buchbestandes. Da der Standortkatalog auch als Inventarverzeichnis Verwendung findet, werden auf der entsprechenden Karteikarte noch Preis und Kaufdatum vermerkt. Zusätzlich dient er als Ausleihverzeichnis (siehe 1.5.3.).

Alphabetischer Katalog:

Er verzeichnet die Bücher nach formalen Gesichtspunkten (Autor bzw. Herausgeber) in alphabetischer Reihenfolge.

Systematischer Katalog:

Er verzeichnet die Bücher ihrem Inhalt entsprechend nach einer Systematik. Er vereinigt sachlich zusammengehörige Literatur und weist sie im Zusammenhang ihres größeren Sachgebietes nach. Die im Anwendungsfall unterschiedlich ausfallende Systematikbegriffsbildung ist formal durch Dezimalklassifikation gekennzeichnet.

Jedes Buch besitzt eine Dezimalklassifikation, welche es in ein Sachgebiet einordnet. Dies ist eine Zahl, in der jede angehängte Ziffer eine weitere Untergliederung des Sachgebietes kennzeichnet (z.B. “1” Informatik, “13” Softwaretechnik, “134” Entwurf).

Einordnen von Neuerwerbungen

Zu jedem erworbenen Buch fertigt die Verwaltungskraft eine Karteikarte an. Buch und Karteikarte erhalten die Signatur: Ein angefertigter Aufkleber wird auf den Buchrücken geklebt. Der Bibliotheksleiter bekommt das Buch und die zugehörige Karteikarte. Er ordnet das Buch unter Mithilfe der Fachreferenten in die Systematik der Bibliothek ein (legt die Dezimalklassifikatoren fest). Die Verwaltungskraft vervielfältigt die Karteikarte, ordnet sie in die Kataloge ein und stellt das Buch in das Regal. Alle Kataloge werden als Zettelkataloge geführt. Dabei werden die einzelnen Buchtitel auf Zettel, d. h. auf Katalogkarten aufgeführt, die in Form einer Kartei geordnet werden. Die Karteikarte hat folgendes Aussehen:

```
-------------------------------------------------------------------------------
 Autorenname, Vorname                                                Signatur
 Sachtitel
 Untertitel
 Ausgabenbezeichnung
 Beigabevermerk (Abbildungen, Karten)
 Erscheinungsvermerk (Ort, Verlag, Jahr)
 Seitenzahl
 ISBN                                                 Dezimalklassifikatoren
-------------------------------------------------------------------------------
```

1.5.3. Bestandsvermittlung (Benutzung)

Der Bibliotheksbenutzer kann sich anhand der Kataloge über den Buchbestand informieren. Die Systematikbegriffsbildung wird durch eine neben den Katalogen ausliegende Übersicht erläutert.

Ausleihe

Die Ausleihe wird vom Benutzer selbständig vorgenommen. Er verzeichnet die Ausleihe auf der Rückseite der zum Buch gehörenden Karteikarte im Standortkatalog mit Datum, Name und Telefon.

Ein Mahnwesen gibt es nicht.

2. Annahmen über Einsatzmöglichkeiten von EDV

2.1. Zielbestimmung des Bibliothekssystems

Die Arbeit mit der Bibliothek soll rationalisiert werden. Der Verwaltungsaufwand ist soweit als möglich zu reduzieren. Dabei sind die Routinearbeiten, speziell der Schreibaufwand zu minimieren. Das Personal soll mit möglichst wenig Zeitaufwand, auch bei einer kontinuierlichen Bestandserweiterung in der Lage sein, die Bibliothek weiterhin neben seinen eigentlichen Tätigkeiten in der Firma zu führen. Dem Benutzer sollen sich bessere und schnellere Informationsmöglichkeiten eröffnen. Die Ausleihe soll schnell und einfach sein. Eine Ausleihstatistik soll ermöglicht werden. An der prinzipiellen Aufgabenverteilung des Personals soll sich nichts ändern.

2.2. Rahmenbedingungen für den Einsatz von EDV

Das System soll im Einbenutzerbetrieb laufen. Dazu wird ein Bildschirmgerät und ein Drucker benötigt. Die Buchinformationen sind bisher nicht EDV-mäßig erfaßt.

2.3. EDV-Kenntnisse der Bibliotheksbenutzer und des Personals

Die Bibliotheksbenutzer haben im allgemeinen keine EDV-Kenntnisse. Die Verwaltungskraft hat ebenfalls keine EDV-Kenntnisse. Sie ist im Schreibmaschineschreiben geübt. Der Bibliotheksleiter besitzt EDV-Kenntnisse. Alle Mitarbeiter sind in der Lage, (evtl. nach Einführung) mit einem Bildschirmgerät und einem Drucker umzugehen.

2.4. Buchbestand und räumliche Gegebenheiten

Der Buchbestand wird kontinuierlich weiter ausgebaut. Anschaffungen pro Monat betragen ca. 10 Bücher. Die Bücher stehen in Freihandaufstellung im Bibliotheksraum. Das Signatursystem bleibt unverändert bestehen. Bildschirmgerät und Drucker werden im Bibliotheksraum aufgestellt. Die Karteikästen werden durch Bandkataloge ersetzt. Bandkataloge haben die Form eines großen Buches, in dem die Titel nacheinander aufgeführt sind.

2.5. Geschäftsgang mit EDV

Die Zeitschriften werden wie bisher nicht berücksichtigt.

Bei der Gestaltung eines möglichen EDV-Systems müssen verschiedene Aspekte berücksichtigt werden.

Die Bibliothek sollte auch bei Ausfall der EDV-Anlage benutzbar bleiben. Das System sollte datenschutzrechtlichen Bestimmungen genügen.

Die Benutzung des Systems sollte den Mitarbeitern der Firma als EDV-Laien so unproblematisch wie möglich sein. Qualitätskriterien der Softwareergonomie sind zu beachten.

Das EDV-System muß sinnvoll in die in 1.5 beschriebenen Geschäftsgänge eingebettet werden.

2.5.1. Bestandsaufbau

Zu beachten ist, wie die im existierenden Betrieb maßgeblichen Gegenstände, insbesondere die Zettel, Karteikarten und Listen, am Rechner modelliert werden können. Hilfreich ist es, ihre Verwendung und Veränderung in den unterschiedlichen Arbeitsprozessen der Beteiligten zu betrachten und zu beachten, für wen sie wann in welcher Form zur Verfügung stehen sollen.

Benutzenden muß die Möglichkeit gegeben werden, neue Buchvorschläge mit Kommentar einzubringen sowie zur Ansicht ausliegende Bücher zu kommentieren.

Zum Gewinn eines fachlichen Verständnisses und möglicher Anforderungen im Umgang mit dem EDV-System bietet sich eine differenzierte Betrachtung der Leistungen an, welche die Vorakzession erbringen soll.

2.5.2. Bestandserschließung

Der fachliche Prozeß der Einordnung neuer Bücher in die Systematik und in die existierenden Kataloge soll gleich bleiben. Jedoch werden die Kataloge am Rechner geführt. Hinzu kommt die Möglichkeit der Erstellung weiterer Kataloge.

2.5.3. Bestandsvermittlung

Durch den Einsatz des EDV-Systems sollten Möglichkeiten, sich über den Buchbestand zu informieren, geschaffen bzw. verbessert werden. Dazu können verschiedene sinnvolle Suchkriterien und Gruppierungskriterien (z.B. für Sachgebiete) zur Verfügung gestellt werden.

Offen ist auch die Form der Ausleihe bzw. die der dafür notwendigen Identifikation, sowie die der Rückgabe.

Für nicht auffindbare Bücher oder sonstige Ambiguitäten des Buchbestandes sind adäquate Meldungen an die Verwaltungskraft sinnvoll.

2.5.4. Verwaltungsarbeiten

Offenkundig notwendig ist die Unterstützung der Bearbeitung und Erweiterung von Bucheinträgen sowie der Systematik.

Wichtige automatisierungswürdige Bereiche der Verwaltungsarbeiten sind ferner ein mögliches Mahnwesen, die Revision und Aktualisierung des Buchbestandes sowie Statistiken.

2.6 Leistungsanforderungen

Umfang der bearbeiteten Vorgänge

Anzahl der verwalteten Bücher	bis 3000
Anzahl der Ausleihvorgänge	bis 10/Tag
Neuerwerbungen	ca. 10/Monat
Aussonderungen	ca. 5/Jahr

Zeitanforderungen

Alle Antwortzeiten des System müssen unter 10 Sekunden liegen. Ausnahme bei Listenerstellung auf dem Drucker.

Literatur

/1/ Christiane Floyd: Arbeitsunterlagen zur Lehrveranstaltung Einführung in die Softwaretechnik. Universität Hamburg. 1993.

/2/ Horst Hunger: Darstellung, Erprobung und Bewertung der Methode "Structured Analysis/Structured Design" anhand der Fallstudie "Bibliothekssystem", TU Berlin 1983.

/3/ Frank Itter: Darstellung, Erprobung und Bewertung der Software-Entwicklungsmethode Jackson System Development anhand der Fallstudie "Bibliothekssystem", TU Berlin 1983.

/4/ Richard Künzig: Darstellung, Erprobung und Bewertung der Software-Entwicklungsmethode Structured Analysis and Design Technique anhand der Fallstudie "Bibliothekssystem", TU Berlin 1983.

/5/ Zafer Tosun: Darstellung, Erprobung und Bewertung der IBM-Verfahrenstechnik anhand der Fallstudie "Bibliothekssystem", TU Berlin 1983.

/6/ Christiane Floyd: Eine Untersuchung von Software-Entwicklungsmethoden, in: Horst Morgenbrod, Werner Sammer (Hrsg.): Programmierungebungen und Compiler. Berichte des German Chapter of the ACM, Band 18, Teubner Verlag, Stuttgart 1984, S. 248-274.

/7/ Christiane Floyd: A Comparative Evaluation of System Development Methods, in: T.W. Olle, H.G. Sol, A.A. Verrign-Stuart (eds.): Information Systems Design Methodologies: Improving the Practce, North-Holland 1986, pp. 19-54.

/8/ Rupert Hacker: "Bibliothekarisches Grundwissen", Verlag Dokumentation München, UTB 148, 3. neubearbeitete Auflage 1976.

Anhang B Literaturliste

- **[BGZ95]**
  Dirk Bäumer, Guido Gryczan, Heinz Züllighoven:
  „Objektorientierte Software-Entwicklung für Banken - Methodik und Erfahrungen aus einer mehrjährigen Projektpraxis“
  OBJEKTSpektrum, (1995) 3, S. 45 - 49

- **[BKG+95]**
  Dirk Bäumer, Rolf Knoll, Guido Gryczan, Wolfgang Strunk, Heinz Züllighoven:
  Objektorientierte Entwicklung anwendungsspezifischer Rahmenwerke“
  OBJEKTSpektrum, (1995) 6, S. 48 - 59

- **[Boo94]**
  Grady Booch:
  „Object Oriented Analysis and Design with Applications“
  Menlo Park, CA: Benjamin / Cummings (1994)

- **[BZ90]**
  Reinhard Budde, Heinz Züllighoven:
  „Software-Werkzeuge in einer Programmierwerkstatt - Ansätze eines hermeneutisch fundierten Werkzeug- und Maschinenbegriffs“
  Bericht der Gesellschaft für Mathematik und Datenverarbeitung; Nr. 182
  München, Wien: Oldenbourg-Verlag (1990)

- **[Che76]**
  P. P. Chen:
  „Entity-Relationship Model: Towards a Unified View of Data“
  ACM Transactions on Database Systems, 4 (1979) 4, S. 397 - 434

- **[CHP+95]**
  Jan Crüsemann, Andreas Hartmann, Thomas Poggendorf, Jan Spiess:
  „Objektorientierte Entwicklung eines Bibliothekssystems“
  Hamburg, Universität Hamburg, Fachbereich Informatik, Studienarbeit, Oktober 1995

- **[CY91]**
  P. Coad, E. Yourdon:
  „Object Oriented Analysis“
  Englewood Cliffs, NJ: Prentice Hall (1991)

- **[CZ95]**
  „Prozeßmodellierung aus Europa hat weltweit Zukunft“
  ComputerZeitung (1995) 5, S. 13

- **[Dud93]**
  „Duden »Informatik«: ein Sachlexikon für Studium und Praxis“
  2., vollst. überarb. und erw. Aufl.; Mannheim, Leipzig, Wien, Zürich: Dudenverlag; 1993

- **[Eic95]**
  Peter Eichhorst:
  „Objektorientierte Unternehmensmodellierung“
  OBJEKTSpektrum, (1995) 2, S. 35 - 37

- **[Eng95]**:
  Thomas Engelmann:
  „Business Process Reenigneering; Grundlagen - Gestaltungsempfehlungen - Vorgehensmodell“
  Wiesbaden: Gabler Verlag, Deutscher Universitäts-Verlag (1995)

- **[FKR+97]**
  Christiane Floyd, Anita Krabbel, Sabine Ratuski, Ingrid Wetzel:
  „Zur Evolution der evolutionären Systementwicklung:
  Erfahrungen aus einem Krankenhausprojekt“
  Informatik Spektrum, 20 (1997) 1, S. 13 - 20

- **[Flo91]**
  Christiane Floyd:
  „Einführung in die Softwaretechnik“
  Skript zur gleichnamigen Vorlesung
  Hamburg, Universität Hamburg, Fachbereich Informatik, 1991

- **[FM95]**
  Otto K. Ferstl, Thomas Mannmeusel:
  „Gestaltung industrieller Geschäftsprozesse“
  Wirtschaftsinformatik, 37 (1995) 5, S. 446 - 458

- **[FS91]**
  Otto K. Ferstl, Elmar J. Sinz:
  „Ein Vorgehensmodell zur Objektmodellierung betrieblicher Informationssysteme im Semantischen Objektmodell (SOM)“
  Wirtschaftsinformatik, 33 (1991) 6, S. 477 - 491

- **[FS93]**
  Otto K. Ferstl, Elmar J. Sinz:
  „Geschäftsprozeßmodellierung“
  Wirtschaftsinformatik, 35 (1993) 6, S. 589 - 592

- **[FS94]**
  Otto K. Ferstl, Elmar J. Sinz:
  „Programmiermodell für objektorientierte, erweiterbare Anwendungssysteme“
  Aus den „Informationen zum Exponat CeBIT’94“; Universität Bamberg, 1994

- **[FS95]**
  Otto K. Ferstl, Elmar J. Sinz:
  „Der Ansatz des Semantischen Objektmodells (SOM) zur Modellierung von Geschäftsprozessen“
  Wirtschaftsinformatik, 37 (1995) 3, S. 209 - 220

- **[FS96]**
  Otto K. Ferstl, Elmar J. Sinz:
  „Geschäftsprozeßmodellierung im Rahmen des Semantischen Objektmodells“
  (in [VB96], S. 47 - 61)

- **[Gry95]**
  Guido Gryczan:
  „Situierte Koordination computergestützter qualifizierter Tätigkeit über Prozeßmuster“
  Hamburg, Universität Hamburg, Fachbereich Informatik, Dissertationsschrift, Juli 1995^

- **[GWZ97]**
  Guido Gryczan, Falk Wiegand, Heinz Züllighoven:
  „Objektorientierte Dokumenttypen & Prototypen zur Unterstützung kooperativer Tätigkeiten“
  Hamburg, Universität Hamburg, Fachbereich Informatik, 1997

- **[GZ92]**
  Guido Gryczan, Heinz Züllighoven:
  „Objektorientierte Systementwicklung, Leitbild und Entwicklungsdokumente“
  Informatik Spektrum (1992) 15, S. 264 - 272

- **[HB96]**
  Thomas Hess, Leo Brecht:
  „State of the Art des Business Process Redesign:
  Darstellung und Vergleich bestehender Methoden“
  2., überarb. und erw. Aufl.; Wiesbaden: Gabler (1996)

- **[HBÖ95]**
  Thomas Hess, Leo Brecht, Hubert Österle:
  „Stand und Defizite der Methoden des Business Process Redesign“
  Wirtschaftsinformatik, 37 (1995) 5, S. 480 - 486

- **[Hru98]**
  Peter Hruschka:
  „Ein pragmatisches Vorgehensmodell für die UML“
  OBJEKTSpektrum (1998) 2, S. 34 - 45

- **[Jac92]**
  Ivar Jacobson:
  „Object-Oriented Software Engineering“
  Reading, MA: Addison-Wesley (1992)

- **[JB94]**
  Ivar Jacobsen, Doug Bennet:
  „Der Softwareentwicklungsprozeß - das Leben jenseits aller Methoden“
  OBJEKTSpektrum (1994) 2, S. 42 - 44

- **[JEJ95]**
  Ivar Jacobson, Maria Ericsson, Agneta Jacobson:
  „The Object Advantage; Business Process Reengineering wirth Object Technology“
  Reading, MA: Addison-Wesley (1995)

- **[KGZ94]**
  Klaus Kilberth, Guido Gryczan, Heinz Züllighoven:
  „Objektorientierte Anwendungsentwicklung“
  2. Auflage; Braunschweig, Wiesbaden: Friedr. Vieweg & Sohn Verlagsgesellschaft mbH (1994)

- **[KKM93]**
  Roland Kaschek, Claudia Kohl, Heinrich C. Meyr:
  „Grenzen objekt-orientierter Analysemethoden am Beispiel einer Fallstudie mit OMT“
  (in [MW93], S. 135 - 154)

- **[KID96a]**
  „KID-Methode - Komponenten“
  ENTITEC GmbH (1996)

- **[KID96b]**
  „KID-Methode - Realisierung“
  ENTITEC GmbH (1996)

- **[KID96c]**
  „KID-Methode - Design“
  ENTITEC GmbH (1996)

- **[Koh96]**
  Claudia Kohl:
  „Objektorientierte Analysekonzepte in der Unternehmensmodellierung“
  (in [VB96], S. 63 - 79)

- **[LZ97]**
  Carola Lilienthal, Heinz Züllighoven:
  „Dokumenttypen im Detail - Dokumente im objektorientierten Entwicklungsprozeß“
  Hamburg, Universität Hamburg, Fachbereich Informatik, 1997

- **[MHH93]**
  Günther Müller-Luschnat, Wolfgang Hesse, Norman Heydenreich:
  „Objektorientiert Analyse und Geschäftsvorfallsmodellierung“
  (in [MW93], S. 74 - 90)

- **[MHS96]**
  Werner Mellis, Georg Herzwurm, Dirk Stelzer:
  „TQM der Softwareentwicklung; Mit Prozessverbesserung, Kundenorientierung und Change Management zu erfolgreicher Software“
  Braunschweig, Wiesbaden: Friedr. Vieweg & Sohn Verlagsgesellschaft mbH (1996)

- **[MP92]**
  David. E. Monarchi, Gretchen. I. Puhr:
  „A Research Typology for Object-oriented Analysis and Design“
  Communications of the ACM, 35 (1992) 9, S. 35 - 47

- **[MW93]**
  Heinrich C. Mayr, Roland Wagner (Herausgeber):
  „Objektorientierte Methoden für Informationssysteme“
  Berlin, Heidelberg: Springer-Verlag (1993)

- **[Oes98]**
  Bernd Oestereich:
  „Objektorientierte Geschäftsprozeßmodellierung mit der UML“
  OBJEKTSpektrum (1998) 2, S. 48 - 52

- **[RBP+91]**
  J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, W. Lorensen:
  „OBJECT-Oriented Modeling and Design“
  Englewood Cliffs, NJ: Prentice-Hall (1991)

- **[Rie95]**
  Dirk Riehle:
  „Muster am Beispiel der Werkzeug und Material Metapher“
  Hamburg, Universität Hamburg, Fachbereich Informatik, Diplomarbeit, März 1995

- **[Rol97]**
  Arno Rolf:
  „Moderne Zeiten für die Wirtschaftsinformatik - Eine Bestandsaufnahme der Stärken und Defizite der aktuellen WI“
  Universität Hamburg, Fachbereich Informatik, Arbeitsbereich „Angewandte und sozialorientierte Informatik (ASI)“, 1997

- **[Sac95]**
  Patricia Sachs:
  „Transforming Work: Collaboration, Learning and Design“
  Communications of the ACM, 38 (1995) 9, S. 36 - 44

- **[Sch94]**
  August-Wilhelm Scheer:
  „Wirtschaftsinformatik - Referenzmodelle für industrielle Geschäftsprozesse“
  4., vollst. überarb. und erweiterte Auflage; Berlin: Springer-Verlag (1994)

- **[SH92]**
  August-Wilhelm Scheer, Alexander Hars:
  „Extending Data Modeling to Cover the Whole Enterprise“
  Communications of the ACM, 35 (1992), S. 166 - 171

- **[SJ96]**
  August-Wilhelm Scheer, Wolfram Jost:
  „Geschäftsprozeßmodellierung innerhalb einer Unternhemensarchitektur“
  (in [VB96], S. 29 - 46)

- **[Suc95]**
  Lucy Suchman:
  „Making Work Visible“
  Communications of the ACM, 38 (1995) 9, S. 56 - 64

- **[Tay95]**
  David A. Taylor:
  „Business engineering with object technology“
  New York, Chichester, Brisbane, Toronto Singapore: John Wiley & Sons Inc. (1995)

- **[VB96]**
  Gottfried Vossen, Jörg Becker:
  „Geschäftsprozeßmodellierung und Workflow-Management. Modelle, Methoden, Werkzeuge“
  Bonn, Albany: International Thomson Publishing GmbH (1996)

- **[Weg90]**
  Peter Wegner:
  „Concepts and Paradigms of Object-Oriented Programming“
  OOPS Messenger, 1 (1990) 1, S. 8 - 87

- **[Wul95]**
  Martina Wulf:
  „Konzeption und Realisierung einer Umgebung zur Koordination rechnergestützter Tätigkeiten in kooperativen Arbeitsprozessen“
  Hamburg, Universität Hamburg, Fachbereich Informatik, Diplomarbeit, 1995