Terminologiebasierte, komponentenorientierte Entwicklung von Anwendungssystemen

Erich Ortner

Technische Universität Darmstadt, Institut für Betriebswirtschaftslehre, Fachgebiet Wirtschaftsinformatik I, Entwicklung von Anwendungssystemen, Hochschulstr. 1, 64289 Darmstadt, Tel.: +49 (6151) 16-4204, Fax: -4301, E-Mail: ortner@bwl.tu-darmstadt.de

Zusammenfassung: In dem Beitrag wird die These vertreten, dass die zukünftige Anwendungssystementwicklung nicht nur komponentenorientiert, sondern zur Integration der Anwendungen auch terminologiebasiert sein wird. Der in dieser Hinsicht heute vorherrschende, aus den 70er Jahren stammende, am Aufbau eines konzeptionellen Datenschemas orientierte Ansatz hat sich in zahlreichen größeren Projekten (z. B. Entwicklung und Einsatz von Branchendatenmodellen oder STEP: Standard for the Exchange of Product Model Data) als ineffizient und längerfristig als „undurchführbar" (inflexibel) erwiesen. Rekonstruierte und beispielsweise in Form eines Lexikons organisierte Terminologien aus den Anwendungsbereichen stellen ein flexibleres Instrument zur Integration verteilter Anwendungssysteme als „unternehmensweite Datenmodelle" dar. Während „Fach-Terminologien" (terminologiebasierter Ansatz) zu den domänenspezifischen (materialen) Entwicklungssprachen gerechnet werden, sind „konzeptionelle Datenschemata" (schemabasierter Ansatz) als Teile von Anwendungen oder Entwicklungsresultaten aufzufassen. Der Beitrag erläutert einige Entwicklungen, die nach Auffassung des Autors diese These untermauern. Dazu zählen Projekte und Produkte aus dem Bereich „Komponententechnologien", die „baukastenorientierte" Entwicklung von Anwendungssystemen mit Komponenten- und Lösungskatalogen sowie der Aufbau von Unternehmensrepositorien, die sowohl für die Entwicklung als auch für den Betrieb und die Administration (global) verteilter Anwendungen eingesetzt werden. Eine weitergehende Untermauerung der aufgestellten These befindet sich in Arbeit.

Schlüsselwörter: Fach-Komponenten, konzeptionelles Datenschema, Terminologie, Komponententechnologie, Stückliste, Repositorium, Workflow-Management.

1 Strategien der Anwendungssystementwicklung

Bei der Entwicklung von Anwendungssystemen können heute verschiedene Strategien (Bild 1) gewählt werden:

a) Standardsoftware wird durch „Customizing" beim Anwender an das Unternehmen angepasst.

b) Es wird „Standardsoftware" ausgewählt, und die Organisationsstrukturen werden an die Software angepasst.

c) Die Entwicklung der Anwendungen erfolgt aus „Bauteilen", die als „Frameworks" vorliegen und an die individuelle Situation der Anwender (z. B. durch Unterklassenbildung) angepasst werden.

d) Es werden individuelle Anwendungslösungen aus „Fach-Komponenten“ (sie werden auch „Business Objects“, „Anwendungselemente“ oder „Application Objects“ genannt), die in hohen Variantenzahlen vorliegen, entwickelt.

In dem Beitrag wird die letzte Variante (d, Bild 1) als die erfolgversprechendste Strategie für die Zukunft betrachtet. Es werden die Bedingungen ihrer Umsetzung behandelt. Dazu gehören beispielsweise die Nutzung von Komponententechnologien wie CORBA (Common

	Anpassung	**Auswahl**
Systeme	Customizing a)	Standardsoftware b)
Kompo-nenten	Frameworks c)	Fach-Komponenten d)

Bild 1: Strategien der Anwendungssystementwicklung

Object Request Broker Architecture) oder DCOM (Distributed Component Object Model), die „baukastenorientierte“ Entwicklung von Anwendungssystemen mit Komponenten- und Lösungskatalogen oder der Aufbau von Unternehmensrepositorien zur unternehmensübergreifenden Administration der Informationsverarbeitungen. Daneben wird der Integrationserfolg der Anwendungssysteme vor dem Hintergrund einer schemabasierten (Datenbankschemata) sowie einer terminologiebasierten (Lexikon) Verbindung der Anwendungen beleuchtet. Es wird an einem Beispiel demonstriert, dass die terminologiebasierte Integration der Anwendungen flexibler ist und für die Zukunft erfolgversprechender erscheint, als eine auf der Grundlage unternehmensweiter Fach-Referenzmodellen (z. B. Datenmodelle) erfolgte, schemabasierte Integration der Anwendungen.

2 Datenschema-orientierte oder terminologiebasierte Integration von Anwendungen?

Das Datenbank-Paradigma – mit einer Organisation der Datenressourcen nach der 3-Schema Architektur von ANSI/SPARC [ANX375] – hat zu Anwendungssystemen geführt, bei denen die Integration der Anwendungen über ein anwendungsübergreifendes Datendesign (ein „konzeptionelles Datenschema“ oder „Unternehmensdatenmodell“) erreicht werden soll. Bild 2 stellt eine integrierte Lösung für das Rechnungswesen eines Unternehmens, basierend auf einem „konzeptionellen Datenschema“ (Grundrechnung) und zahlreichen „externen Schemata“ (Sonderrechnungen), die Ende der 70er Jahre entwickelt und publiziert wurde [WeOr77], vor.

Mit zunehmender Verbindung der Informationsverarbeitungen von Unternehmen – auch über die Unternehmensgrenzen (globale Informationsverarbeitung) hinaus [KuSc99] – wird der Aufwand für den Aufbau und die Administration solcher „Datenschemata“ (z. B. Branchendatenmodelle, das Referenz-Datenmodell von R/3 oder STEP: Standard for the

Exchange of Product Model Data) unvertretbar groß. Heute gibt es kaum ein Unternehmen, das es geschafft hat, sein „Unternehmensdatenmodell“ komplett aufzubauen und die Konsistenz des „konzeptionellen Datenschemas“ über einen längeren Zeitraum aufrecht zu erhalten.

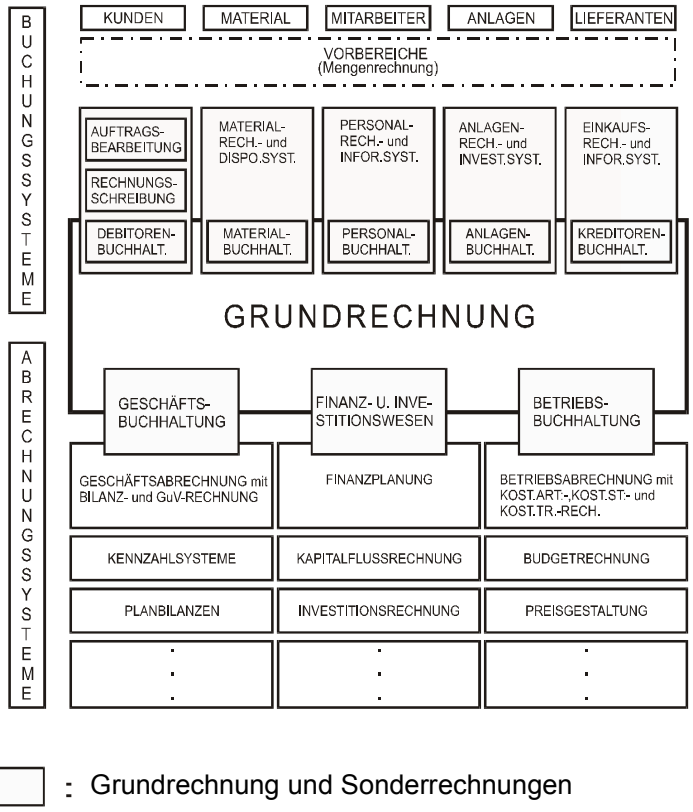

Bild 2: Datenbankbasiertes System des Rechnungswesens

Die zentrale Schwierigkeit resultiert aus der Tatsache, dass ein „konzeptionelles Datenschema“ ein gigantisches Entwicklungsergebnis auf der Anwendungsseite der Systeme darstellt und nicht zum Entwicklungssystem (zur Entwicklungssprache), mit dem die Anwendungen entwickelt wurden, gerechnet werden kann. Damit ist es in den Anwendungssystembetrieb sowie in den Ausbau der Anwendungen vollständig integriert, erweist sich auf Grund seiner Größe und Strukturierung als inflexibel und kann nur mit sehr hohem Aufwand mit den entwickelten Anwendungen im „Einklang“ (konsistent) gehalten werden.

Eine Alternative zu diesem Ansatz stellt eine terminologiebasierte (und komponentenorientierte) Entwicklung von Anwendungssystemen (Bild 3) dar. Der Ansatz basiert nicht nur auf der Datenbanktechnologie [ANX375], sondern auf einer weitergehenden Komponententechnologie [OMGr96], die im nächsten Abschnitt erläutert wird. Die (Fach-) Terminologie (ihr Aufbau und ihre Administration) gehört nicht zu den Anwendungen, sondern sie wird zu der eingesetzten Entwicklungssprache (zum Entwicklungssystem) gerechnet und kann wesentlich flexibler (z. B. in Form eines Lexikons) organisiert und mit geringerem Aufwand als beim schemaintegrierten Datenbank-Ansatz separat von den Anwendungen administriert werden.

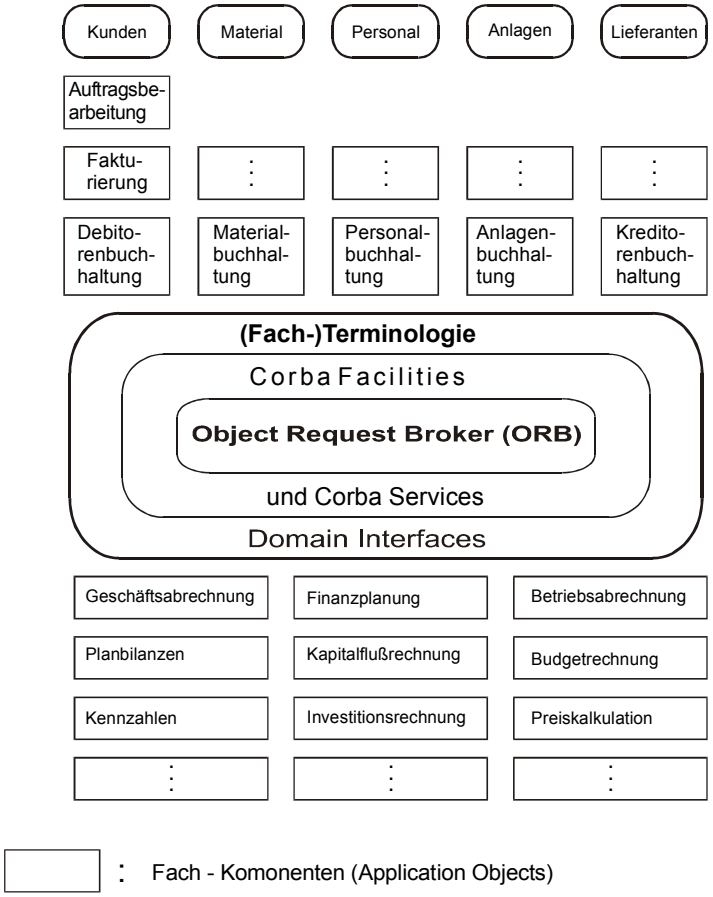

Bild 3: Komponentenorientiertes System des Rechnungswesens

Eine Terminologie für das Rechnungswesen (Bild 3) kann dabei beispielsweise mit „Prädikatorenregeln“ [Lore73] wie folgt aufgebaut werden:

$$x \in \text{Bilanzkonto} \Rightarrow x \in \text{Unternehmenskonto}$$

$$x \in \text{GuV-Konto} \Rightarrow x \in \text{Unternehmenskonto}$$

$$x \in \text{Unternehmenskonto} \wedge x \in' \text{Bilanzkonto} \Rightarrow x \in \text{GuV-Konto}$$

$$\vdots$$

$$x \in \text{Vermögenkonto} \Rightarrow x \in \text{Bilanzkonto}$$

$$\vdots$$

$$x \in \text{Bilanzkonto} \wedge x \in' \text{Vermögenkonto} \Rightarrow x \in \text{Kapitalkonto}$$

$$\vdots$$

ε ist als „ist ein“ und ε‘ als „ist kein“ zu lesen. „∧“ steht für das logische UND und der Regelpfeil „⇒“ bedeutet: Es ist erlaubt von... überzugehen zu... Die Verwendung (Bedeutung, Definition) der Wörter (Termini) muss natürlich in dem betreffenden Unternehmen

oder dem Anwendungsgebiet gemeinsam mit den Anwendern rekonstruiert und im Zusammenhang mit neuen Entwicklungsprojekten oder einer anderen Arbeitspraxis in den Anwendungsbereichen stets aktualisiert werden. Die Terminologie wird jedoch auf der Seite der Entwicklungssprachen und nicht auf der Seite der entwickelten Anwendungen, unternehmensweit administriert. Dadurch kann sie flexibler (verwendungsneutral) aufgebaut, erweitert oder geändert werden. Die Integration der Anwendungen erfolgt „terminologie-basiert“ aus den rekonstruierten (materialen) Entwicklungssprachen heraus. Es bleibt festzustellen, dass die überwiegende Zahl von Anwendungssystemen heute als Datenbank-Anwendungen – mit einem unternehmensweiten oder unternehmensübergreifenden Datenschema als Integrationsbasis – entwickelt und (als Standard-Lösungen) vertrieben werden.

Auch „Termini“ sind – modellierungstechnisch gesehen – als „Schemata“ (sprachliche Repräsentationen von Typen) aufzufassen. Sie stehen nur nicht wie Datenschemata oder Programmschemata als „Anwendungen“ bereits zur Verfügung, sondern müssen erst im Entwicklungsprozess aus dem Entwicklungssystem (materiale Entwicklungssprache, Fach-Normsprache) heraus zu „Anwendungen“ zusammengefügt werden.

3 Verteilte Informationsverarbeitung auf der Basis der CORBA

Die verteilte Informationsverarbeitung orientiert sich heute an CORBA (Common Object Request Broker Architecture). CORBA ist eine von der Object Management Group (OMG) spezifizierte Beschreibung eines verteilten Objektverwaltungssystems, die festlegt, wie verteilte Objekte (z. B. Anwendungen) mit Hilfe eines Object Request Brokers (ORB) miteinander kommunizieren können (Bild 4). Die Kommunikation [LorK92, S. 113] läuft dabei „modellierungstechnisch“ nach dem Konzept von „Schema“ (rekonstruiertes Wissen) und „Ausprägungen“ (kommunizierte Information) zwischen den Kommunikations-teilnehmern (Anwender und/oder Systemkomponenten) ab. Im Unterschied zum Microsoft „Distributed Component Object Model“ (DCOM) ist CORBA Plattformenneutral (unabhängig von Betriebssystemen) und unterstützt die Kommunikation zwischen heterogenen Anwendungen. Sowohl DCOM als auch CORBA sind von konkreten Programmiersprachen weitgehend unabhängig und sind daher eindeutig der als „Middleware“ bezeichneten Systemsoftware zuzuordnen.

Die in Bild 4 dargestellte Architektur wird OMA (Object Management Architecture) genannt. Sie stellt eine universelle Kommunikationsplattform dar, mit deren Hilfe Objekte zusammen-arbeiten können, deren Implementierungen sich auf unterschiedlichen Plattformen befinden. Den „Datenaustauschmechanismus“ bildet der Object Request Broker (ORB), der sich in der Mitte des Bildes 4 befindet. Er sorgt dafür, dass Objekte bzw. Software-Komponenten über Grenzen von Betriebssystemen und Hardwareplattformen hinweg miteinander durch Nachrichtenaustausch (nach dem Konzept von „Schema“ und „Ausprägungen“ modelliert) kommunizieren können.

Neben dem ORB sind in der OMA vier weitere Bereiche enthalten: „CORBAservices“, „CORBAfacilities“, „Domain Interfaces“ und „Application Objects“ [Schu99]. Bei den „CORBAservices“ handelt es sich um eine Sammlung elementarer Basisdienste, die im Falle der nichtverteilten Informationsverarbeitung von Compiler-Laufzeitsystemen, von Entwick-lungswerkzeugen oder von Betriebssystemen bereitgestellt werden. Die Dienste sind hochgradig modular, elementar und ohne Überdeckung (redundanzfrei) spezifiziert. Einerseits

können die Programmierer von verteilten Anwendungssystemen diese Dienste nutzen und beliebig miteinander kombinieren, andererseits sind CORBAservices aber auch als Grundbausteine für höherwertige Dienste, die sogenannten CORBAfacilities vorgesehen. Mittlerweile werden hier 15 Dienste (z. B. ein „Naming-Service“ als eine Art „Telefonbuch“ für CORBA-Objekte im System oder ein „Event-Service“, der neben der synchronen Kommunikation zwischen zwei Objekten auch das anonyme Broadcasting oder die asynchrone Kommunikation durch Einführung eines zusätzlichen Ereigniskanals zwischen Objekten erlaubt) in Form umfangreicher Spezifikationen [OMGr96] festgelegt.

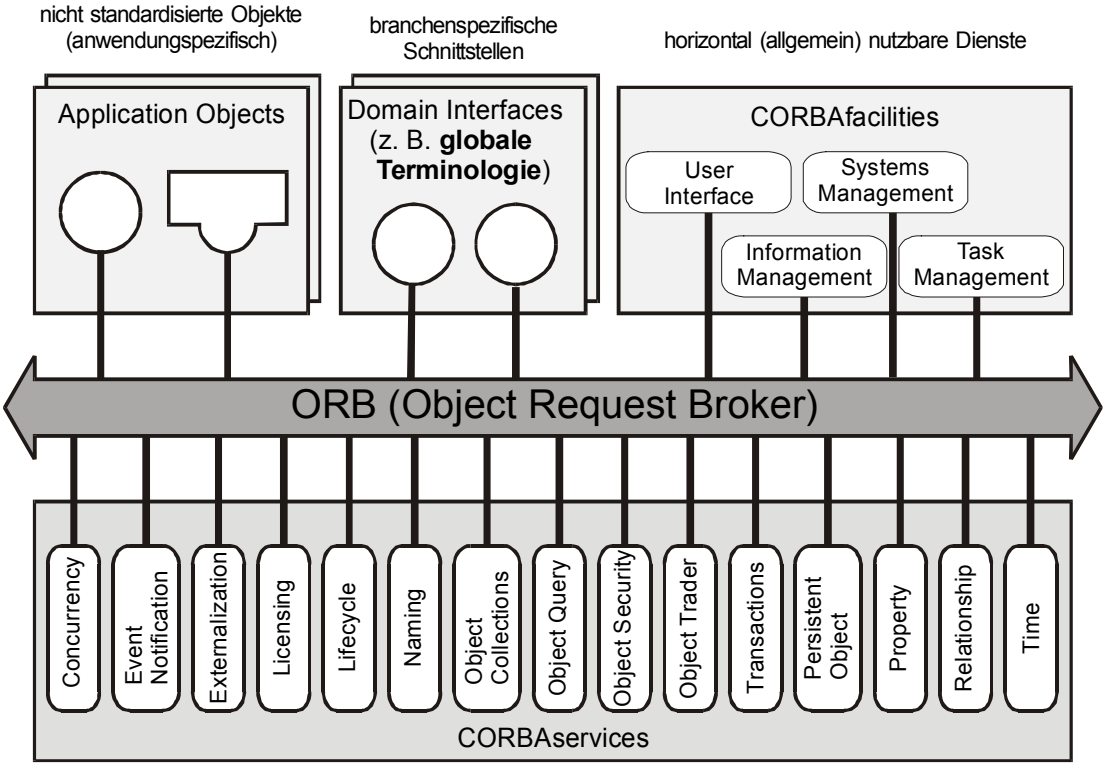

Bild 4: Architekturenmodell der Middleware nach OMG [OMG96a]

Erheblich mehr Funktionalität als CORBAservices bieten die „CORBAfacilities“, die auch als „Common Facilities“ bezeichnet werden. Hier handelt es sich um horizontale, daß heißt fachneutrale Komponenten, die einen klar umrissenen Funktionsumfang besitzen und von den Entwicklern verteilter Anwendungen ähnlich wie Basissysteme (z. B. Datenbank-Management-Systeme oder Workflow-Management-Systeme) genutzt werden können.

Die CORBAfacilities teilen sich in vier große Bereiche auf (Bild 4). In den „User Interface Common Facilities“ sollen Schnittstellen beschrieben werden, die weitestgehend mit der Benutzerschnittstelle von Anwendungen und insbesondere mit der Präsentation von Informationen zu tun haben. Die „Information Management Common Facilities“ befassen sich mit allen Aspekten der Datenhaltung. Im November 1997 wurde hier eine Meta-Object Facility (MOF) – ein Repository Management System – für die Verwaltung von Metadaten in verteilten Systemen standardisiert [OMGr97]. Die „Systems Management Common Facilities“ umfassen Dienste zur Verwaltung zur Konfiguration und zum Betrieb verteilter

Objektverwaltungssysteme. Und bei den „Task Management Common Facilities“ finden sich Infrastrukturdienste (Basissysteme), die den Benutzer direkt bei seiner Arbeit unterstützen, wie z. B. eine Workflow Management Facility [Schu99].

Im Bereich der „Domain Interfaces“ werden branchenspezifische Sachverhalte für die verteilte Informationsverarbeitung zwischen Unternehmen spezifiziert. Dabei wurden mit dem Ziel, die Standardisierung der Domain Interfaces voranzutreiben, von der OMG eigene Arbeitsgruppen (Task Forces) eingerichtet, die sich jeweils mit einem abgegrenzten Aufgabengebiet beschäftigen. Momentan gibt es innerhalb dieser „Domain Technology Commitee Arbeitsgruppen“ sechs Fachbereiche:

- **Business Objects:** Festlegung von Identifikations- und Beschreibungsstandards für Geschäftsobjekte (Business Objects), um Anwendungssysteme gemäß dem Baukastenprinzip aus Komponenten zusammensetzen zu können.
- **Electronic Commerce:** Klärung von Verkaufs-, Copyright– und Bezahlungsmodalitäten für den Handel mit elektronischen Inhalten.
- **CORBAfinancials:** Definition von Diensten zur sicheren Abwicklung von Finanztransaktionen sowie Entwicklung von Referenzmodellen für Bankgeschäfte.
- **CORBAmanufacturing:** Spezifikation von Anforderungen an CORBA-basierte Software im Produktdaten-Management (PDM)-Bereich und Identifikation geeigneter Referenzmodelle; z. B. Spezifikation einer produktneutralen Schnittstelle (PDM-Enabler) für PDM-Systeme.
- **CORBAmed:** Anwendungsübergreifende Standards im medizinischen und klinischen Bereich; z. B. Definition einer Standard-IDL (Interface Description Language) – Schnittstelle für den Austausch von Patientenakten.
- **CORBAtel:** Identifikation geeigneter Referenzmodelle und Schnittstellen für CORBA-basierte Anwendungen im Telekommunikationsbereich.

„Application Objects“ (Bild 4) bilden jenen Teil CORBA-basierter verteilter Anwendungssysteme, die fachbezogene Aufgaben aus den jeweiligen Anwendungsbereichen lösen und nicht ohne weiteres in anderen Kontexten wiederverwendet werden können. Aufgrund der Vielfalt und der individuellen Anforderungen ist hier eine Standardisierung durch die OMG **nicht** vorgesehen.

Wenn nachfolgend (Abschnitt 5) von „Fach-Komponenten“ und der Entwicklung von Anwendungen mit Hilfe von Komponenten- und Lösungskatalogen die Rede ist, so stehen in erster Linie „Application Objects“ (Bild 4) zur Debatte. Eine offene Frage ist dabei die Frage nach der „Granularität“ der „wieder-“, „weiter-“ oder „mehrfachverwendbarer“ Fach-Komponenten (Application Objects). Zur Zeit haben solche „Bauteile“ noch eine erhebliche Größe, so dass sie als eigenständige, ablauffähige Funktionseinheiten entworfen werden müssen. Ein „Fakturierungsprogramm“ oder die „Gehaltsabrechnung“ als Teil eines Personalinformationssystems werden beispielsweise als „Application Objects“ angesehen. Der Trend

geht aber zur Spezifikation, Wiederverwendung und Administration kleinerer Einheiten. Der Standardisierungsbedarf kann dabei durch den sich abzeichnenden Komponentenmarkt in den nächsten Jahren rasch anwachsen.

Bei einer terminologiebasierten Integration der Anwendungen (Application Objects) ist die fachspezifische Terminologie der Anwendungsbereiche global zu rekonstruieren (und zu normieren) und beispielsweise unter „fiktiven Bezeichnern“ (z. B. „Buchstabenfolgen“) im Bereich „Domain Interfaces“ (Bild 4) der CORBA zu implementieren und zu administrieren. Dadurch wird eine Art „normsprachliches Wrapping“ kommunizierender Anwendungen ermöglicht, bei dem die global rekonstruierte Terminologie (einer universellen Normsprache oder einer „universal networking language“) als „Zwischensprache“ [Ortn99] auftritt, die hinsichtlich ihres Vokabulars von den Sprachteilnehmern (z. B. Application Objects) nicht beherrscht werden, jedoch auf global einheitlich rekonstruierten Begriffen beruhen muss.

Verschiedene „Wörter“ (z. B. Kunde, Customer, abcd, Client, etc.) stellen denselben Begriff dar und können, da es sich (normierter Weise) um synonyme Bezeichnungen handelt, gegeneinander ausgetauscht werden. In Verbindung mit dem logischen Gesetz der „Komparativität“ (Sind zwei Größen, z. B. „Kunde“ und „Client“, einer dritten, z. B. „abcd“, gleich (**R**), so sind sie auch untereinander gleich.) können Komponenten, die unterschiedliche (aber synonyme) Terminologien benutzen, mit einer aus „fiktiven Bezeichnern“ (z. B. abcd, abce, etc.) für die global rekonstruierten Begriffe aufgebauten universellen Normsprache so „gewrapped“ (eingehüllt, verkleidet) werden, dass die Kommunikation zwischen ihnen möglich ist, ohne dass die Normsprache (auf Ebene der „fiktiven Bezeichner“) von den „Kommunikationsteilnehmern“ (z. B. Application Objects) beherrscht werden muss [Ortn99, S. 326].

In dem gewählten Beispiel ist aus Gründen der „Komparativität“ folgender Zusammenhang gegeben:

Kunde	**R**	abcd	⟶	Kunde	**R**	Customer
∧ Customer	**R**	abcd		∧ Customer	**R**	Kunde
∧ Client	**R**	abcd		∧ Kunde	**R**	Client
etc.				∧ Client	**R**	Kunde
				∧ Customer	**R**	Client
				etc.		

Das Beispiel wird von links nach rechts gelesen. Dabei hat „∧“ Vorrang vor „⟶“. In einer Komponente A wird der Terminus „Kunde“ und in der Komponente B der Terminus „Client“ verwendet. Die Kommunikation läuft über den „fiktiven“ (aber synonymen) Terminus „abcd“ (Wrapping der Komponente A mit „Kunde **R** abcd“ und Wrapping der Komponente B mit „Client **R** abcd“) ab.

Das Beispiel verdeutlicht, dass beim Aufbau einer globalen Terminologie für die Kommunikation (Bild 4) zwischen verteilten, „verschiedensprachigen“ Anwendungen, die

Rekonstruktion und Normierung der (globalen) **Begriffe** und nicht die Rekonstruktion und Normierung ihrer Bezeichner (z. B. „abcd“) im Vordergrund steht.

4 Repositorien: Aufbau eines Komponenten- und Normsprachen-Servers

Repositorien [Bern99] ziehen heute ein großes Interesse aus Sicht der rechnerunterstützten Informationsverarbeitung in Unternehmen sowie aus Sicht der Informatik und Wirtschaftsinformatik auf sich. Berners Lee, Erfinder des World Wide Web (W3), spricht im Zusammenhang mit Repositorien und der zu entwickelnden „Metainformationsverarbeitung“ von einer „neuen Ära der Aufklärung“ [Taps98]. Gemeint ist die Tatsache, dass sich durch die weltumspannende Informationsverarbeitung eine Annäherung der Sprachen (auch der Gebrauchssprachen) bis auf „globale Uniformität“ zahlreicher Fachbegriffe (durch den systematischen Aufbau von „Fach-Normsprachen“, beispielsweise für den Electronic Commerce) bereits abzeichnet. Ein nützliches Instrument zur Bewältigung einiger Aufgaben in diesem Kontext stellen Repositorien [Ortn99] dar.

Die Anwendungssystementwicklung könnte auch aus dieser Perspektive in Zukunft komponentenorientiert und terminologiebasiert sein. Anwendungssysteme lassen sich nach dem „Baukastenprinzip“ mit Komponentensammlungen und Lösungskatalogen aus auf dem Markt verfügbaren Bausteinen (Fach-Komponenten) in hoher Variantenzahl - durch „Auswahl und Verbindung“ und nicht durch „Auswahl und Anpassung“ - zu individuellen Lösungen „zusammensetzen“. Dabei müssen jedoch bestehende Lösungen (die man vor diesem Hintergrund „Wissens-vor-Produkte“ nennen könnte) zunächst an die Komponentenorientierung „herangeführt“ werden. Ebenso ist das Konzept der „Schemaintegration“ (Fach-Referenzmodelle) um das Konzept der „Terminologiebasierung“ (Fach-Terminologien und Fach-Normsprachen) in der Anwendungssystementwicklung zu ergänzen. Einen besonderen Stellenwert erhalten hierbei Repositorien oder Metainformationssysteme, die als Komponenten- und Normsprachen-Server (Bild 5) implementiert werden können.

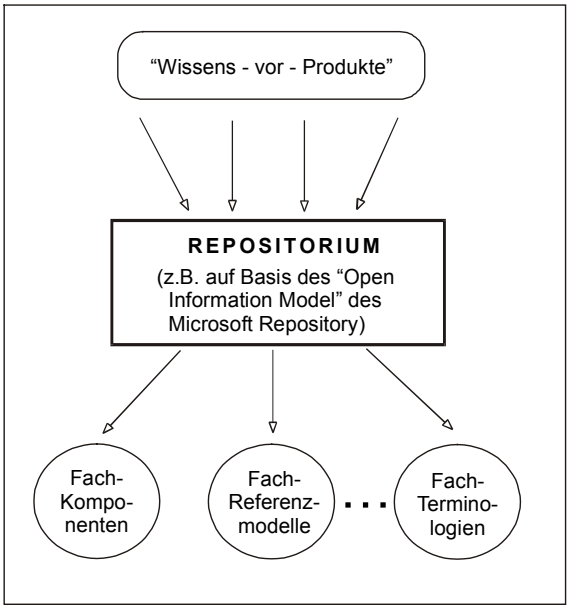

Bild 5: Komponenten- und Normsprachen-Server

In Bild 5 werden die Aspekte der zu leistenden Arbeit aus Sicht eines geeigneten Werkzeugs (Servers) dargestellt. Der Untersuchungsgegenstand, die „Wissens-vor-Produkte“, sind in großer Zahl vorhanden. Sie müssen jedoch noch gemeinsam mit den Anwendern sachkundig „aufbereitet“, d. h. sprachkritisch rekonstruiert und in Form von „Fach-Komponenten“, „Fach-Referenzmodellen“ oder „Fach-Terminologien“ (Bild 5) der Software-Industrie sowie den Anwender-Unternehmen zur Verfügung gestellt werden. Dabei wäre ein Werkzeug, das man fachlich „Normungsrepositorium“ nennen könnte und das beispielsweise auf Basis eines objektrelationalen DBMS [Ston96] und eines geeigneten (erweiterbaren) Metaschemas wie dem „Open Information Model“ des Microsoft Repository [Bern99] als W3-Server installiert würde, sehr nützlich.

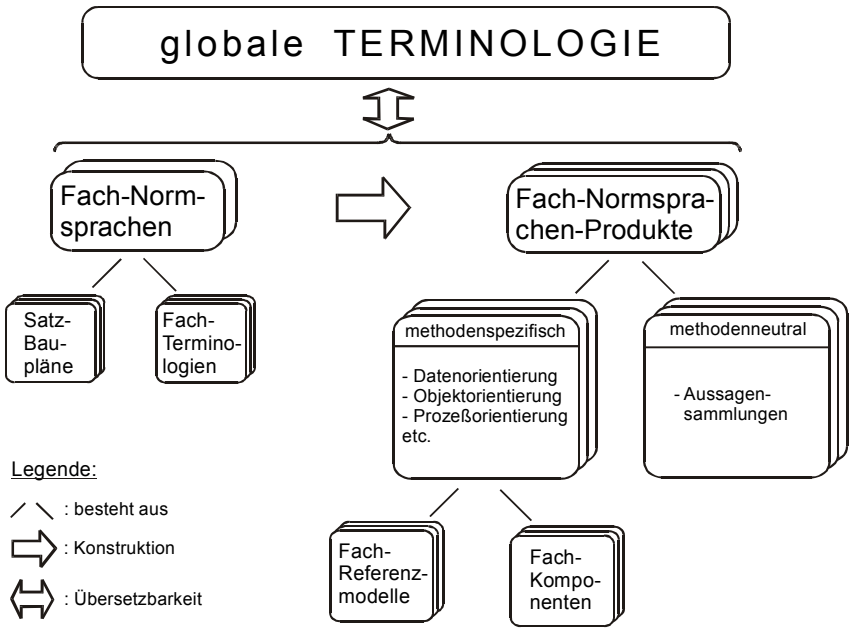

Bild 6: Bereiche normsprachlicher Anwendungssystementwicklung

Bei der Entwicklung von Anwendungssystemen aus Komponenten werden Sprachen, das sind heute i. d. R. Metasprachen, die z. T. formalisiert sind, eingesetzt. Es spricht nichts dagegen, auch rekonstruierte Objektsprachen (Fach-Normsprachen) oder „materiale Sprachen“ [Ortn97] zur Entwicklung von Anwendungssystemen einzusetzen. Dabei können die rekonstruierten Sprachkonstrukte beispielsweise in Form fachlicher Komponenten [Mert97], [KaLO99], Fach-Referenzmodellen [Sche95], [Fran97] oder fachlicher Terminologien [Schi97], [Ortn97] [Lehm99] vorliegen. Bild 6 stellt hier den Gesamtzusammenhang der relevanten Sprachen und Sprachartefakte dar und kennzeichnet dadurch die Komponenten einer auf der Grundlage einer „universellen Normsprache“ (globale Terminologie) organisierten Entwicklung von fachlich konsistenten „Wissensprodukten“. Dabei wird die „globale Terminologie“ zur transparenten Übersetzung des kommunizierten Wissens (das dann als „Information“ (Ausprägungen) aufzufassen ist) zwischen Wissensprodukten, die in verschiedenen Fach-Normsprachen entwickelt wurden, eingesetzt. Satz-Baupläne und Fach-Terminologien (Bild 6) bestimmen – neben einer spezifischen Gegenstandseinteilung – hauptsächlich eine Fach-Normsprache, die aus der Anwendungspraxis rekonstruiert wird. Die Entwicklungsergebnisse (Fach-Normsprachen-Produkte, Bild 6) gelten als „methoden-

neutral“, wenn sie (noch) nicht an einer spezifischen Anwendungsarchitektur – wie sie beispielsweise datenorientierte, objektorientierte oder prozessorientierte Lösungen aufweisen – ausgerichtet sind. Fach-Referenzmodelle und Fach-Komponenten (Bild 6) sind dagegen i. d. R. an einer spezifischen Gegenstandseinteilung (Anwendungsarchitektur) für den Systementwurf (z. B. Datenbasis/Anwendungen, Ausführungsprozesse/Steuerungsprozesse, Wissensbasis/Inferenzkomponente) orientiert. Sie werden aus der Aussagensammlung des „methodenneutralen Entwurfs“ [Ortn97] mit Hilfe von Werkzeugen [z. B. Vogle94] „generiert“.

Bei der Entwicklung von Anwendungssystemen mit „materialen Sprachen“ ist es möglich, ihre Korrektheit nicht nur in struktureller (Grammatik), sondern auch in inhaltlicher (Fach-Terminologie) Hinsicht anhand der eingesetzten Sprachen oder Sprachartefakte zu überprüfen. Dies setzt eine aus Sicht der einzelnen Anwendungen (Datenbank-Anwendungen, Workflow-Management-Anwendungen, etc.) neutrale Normung der Inhalte (Fachsprachen, Terminologien) voraus.

Um den Aspekt der Normsprachlichkeit von Repositorien zu erläutern, gehen wir von einer Repositoriumsarchitektur aus, wie sie Bild 7 zeigt. Die Architektur erstreckt sich über vier Sprachstufen. Auf der 1. Sprachstufe ist der Anwendungsbereich durch seine Fachsprache (Objektsprache) und die zu entwickelnden Anwendungen vertreten. Auf der 2. Sprachstufe wird die Dokumentationsstruktur für die entwickelten Sprachobjekte (Typen) der 1. Sprachstufe, das Repositoriumsmetaschema, modelliert. Im Prinzip geht es dabei um eine besondere „Implementierung“ der Methoden (Sprachen) und Werkzeuge (Tools i. w. S.), die bei der Entwicklung (z. B. als Modellierungstool), dem Betrieb (z. B. als Basissysteme) oder der Nutzung (z. B. als Anwendungen) von Informationssystemen zum Einsatz kommen, als Metaschema. Mit einem Metaschema können die in einer Entwicklungssprache (Methoden, Tools) erzielten Entwicklungsresultate strukturiert (in Teile und Beziehungen zwischen den Teilen zerlegt) beschrieben und stücklistenartig verwaltet werden.

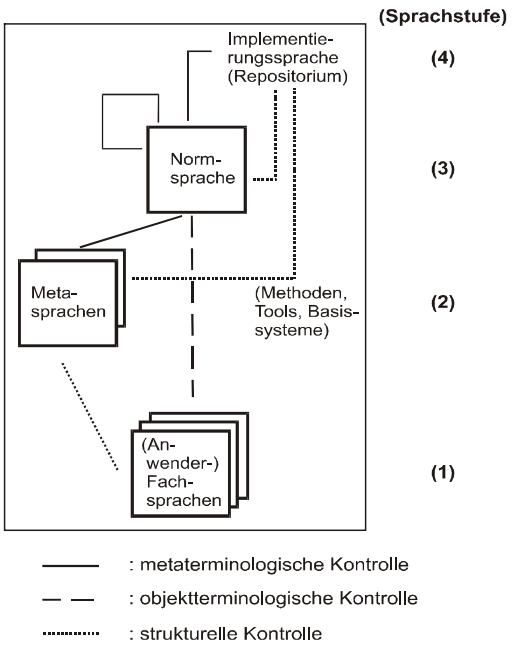

Bild 7: Aufbau eines normsprachlichen Repositoriums

Auf der 3. Sprachstufe (Bild 7) wird die auf **allen** Sprachstufen eingesetzte objekt- und metasprachliche Terminologie auf Basis eines Normsprachenschemas [Ortn99] verwaltet.

Von der 2. Sprachstufe aus gesehen (Bild 7) unterliegen die Entwicklungsergebnisse auf der 1. Sprachstufe einer strukturellen Kontrolle. Es wird z. B. bei der Entwicklung einer Tabelle (Relation) auf der 1. Sprachstufe auf der 2. Sprachstufe festgehalten, ob es sich um eine normalisierte Relation handelt, aus wie vielen Attributen die Relation besteht, welche Attribute Schlüssel- und welche Attribute Nichtschlüsselattribute sind, welcher Datentyp den Attributen zugeordnet ist, etc.

Auf der 4. Sprachstufe (Bild 7) ist schließlich der „Katalog“ bzw. das „Run-Time-Repository“ des zur Realisierung des normsprachlichen Repositoriums eingesetzten DBMS (Datenbank-Management-Systems) untergebracht. Von dieser Sprachstufe aus findet die strukturelle Kontrolle aller als Datenbank-Anwendung implementierten Sprachartefakte (Metaschema, Normsprachenschema) eines (normsprachlichen) Repositoriums statt.

Wenn man davon ausgeht, dass die Syntax dieser Sprachen (Sprachstufe 1 – 4) in einem Beispiel zu Bild 7 auf allen Sprachstufen dieselbe – nämlich das Relationenmodell - ist, dann wird ein Entwicklungsergebnis (z. B. die Tabellen (Relationen) KUR und URLAUB eines Personalinformationssystems) auf Basis einer normsprachlichen Terminologie mit einem Repositorium wie folgt dokumentiert:

1. Sprachstufe

URLAUB (UNUMMER; DAUER, PERSON, ...)

4711	2	Müller
4712	1	Meier
4713	2	Schulze
etc.		

KUR (KNUMMER; DAUER, PERSON, ...)

0815	6	Schulze
0816	8	Meier
0817	5	Huber
etc.		

2. Sprachstufe

RELATION (RELNAME; ATTRIBUTANZAHL ...)

„Urlaub“	8
„Kur“	12
etc.	

DATENELEMENT (DENAME; DATENTYP, ...)

„Nummer“	integer
„Name“	character
etc.	

ATTRIBUT (ATNAME, RELNAME; DENAME, SCHLÜSSEL, ...)

„Unummer“	„Urlaub“	„Nummer“	ja
„Dauer“	„Kur“	„Nummer“	nein
„Person“	„Urlaub“	„Name“	ja
etc.			

3. Sprachstufe

TERMINUS (TNAME;	DEFINITION, ...	,SPRACHSTUFE)
„Relation“	Untermenge des Cartesischen Produktes der Wertebereiche von Attributen	2
„Datentyp“	Zusammenfassung von Wertebereichen und Operationen zu einer Einheit	2
„Urlaub“	Dienstfreie Zeit, die man zum Zwecke der Erholung in einem Betrieb erhält	1
„Person“	Mensch hinsichtlich seiner körperlichen, geistigen und rechtlichen Eigenschaften	1
etc.		

4. Sprachstufe

RELATION (RELNAME; ATTRIBUTANZAHL, SPRACHSTUFE, ...)

„Relation“	6	2
„Attribut“	8	2
„Terminus“	5	3
etc.		

Die Sprachartefakte der 1. Sprachstufe treten nicht explizit im Repositorium auf, sondern sie werden lediglich entsprechend ihrer Aufteilung gemäß Metaschema (2. Sprachstufe) im Repositorium dokumentiert. Man könnte auf dieser Sprachstufe natürlich auch ein (Meta-) Attribut „SOURCECODE“ in die Metaschemaelemente „RELATION“, „DATENELEMENT“ etc. aufnehmen. Dann könnten die beschriebenen Applikationsobjekte (z. B. die Relationen URLAUB und KUR) auch „physisch“ (als Sourcecode) zur Dokumentation hinzugerechnet werden. „In repositories all instances refer to types“ stellen Bernstein et al. in [Bern99] zutreffend fest. Deshalb sind die Ausprägungen der Attribute RELNAME (Relationenname), DENAME (Datenelementname), TNAME (Terminusname) etc. in Anführungszeichen geschrieben. Damit wird ausgedrückt, dass hier auf einen abstrakten Gegenstand, einen Typ, und nicht wie beispielsweise mit dem Namen „4711“ in der Relation (Tabelle) URLAUB, der auf ein konkretes Urlaubsgeschehnis referenziert, Bezug genommen wird.

5 Entwicklung von Anwendungssystemen aus Komponenten

Für die Entwicklung konfigurierbarer Anwendungssysteme (Bild 1, d) ist eine große Zahl von Fach-Komponenten mit spezifischen Funktionen erforderlich. Dadurch wird es möglich, flexible, modulare und für jeden Anwendungsfall individuell konfigurierbare Anwendungssysteme herzustellen. Je früher dabei im Entwicklungsprozess die Komponentenorientierung

beginnt, desto größer ist die Anzahl der Varianten, die von einer Komponente konstruiert, für die Suche adäquat dokumentiert und in Katalogen zugriffsbereit administriert werden müssen.

Bei der Entwicklung von Anwendungssystemen aus Komponenten, sind vor allem drei Problembereiche zu lösen:

1. Die „Suche“ der geeigneten Komponente in einer Entwicklungssituation muss durch den Aufbau von Komponenten- und Lösungskatalogen [Lang98] unterstützt werden.

2. Für die „Assemblierung“ der Komponenten zu Anwendungslösungen ist das Stücklistenverfahren [KaLO99] gut geeignet.

3. Zur „Verbindung“ der Komponenten zu Anwendungssystemen müssen geeignete Kopplungstechniken [Grif98] zur Verfügung gestellt werden.

Zu 1: Die Entwicklung von Fach-Komponenten auf der Grundlage des rekonstruierten Anwenderfachwissens reicht für die Konfigurierung eines Anwendungssystems aus Komponenten nicht aus. Es fehlt der „Bauplan“, nach dem die Fach-Komponenten zu Anwendungslösungen verbunden werden. Da die Fach-Komponenten (z. B. Bild 3) aufgabenorientiert (Funktionalität) definiert sind, muss ein Aufgabenplan [KaLO99] aus der Organisationsmodellierung (Aufbau- und Ablauforganisation) des Unternehmens für die Konfigurierung der Fach-Komponenten zu Anwendungslösungen (Bild 8) abgeleitet werden.

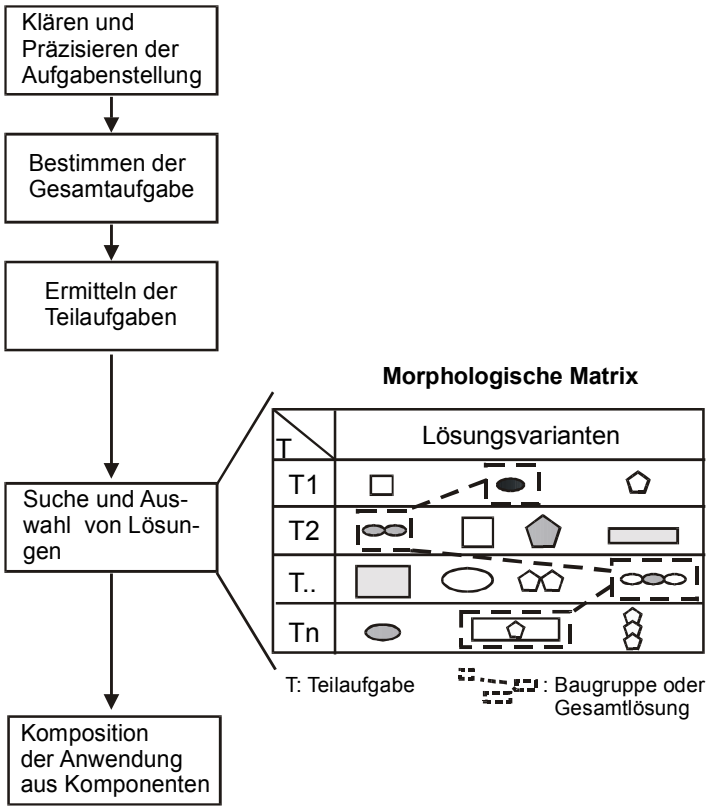

Bild 8: Entwicklung von Anwendungssystemen aus Komponenten

Aufgabenpläne sind zielorientierte Typisierungen und Gruppierungen von Arbeitsschritten (Handgriffen und/oder Sprachhandlungen) zu Aufgabeneinheiten, die einer Stelle, einer Arbeitsgruppe oder einer größeren organisatorischen Einheit ablauforganisatorisch zugeordnet werden.

Ist der Entwicklungsprozess soweit fortgeschritten, dass die Teilaufgaben feststehen, sind nur noch die beiden Konstruktionshandlungen „Auswahl und Komposition“ (Verbindung) zugelassen, um aus bestehenden Fach-Komponenten (Varianten) ein Anwendungssystem herzustellen. Zur Unterstützung der Auswahl und des Kompositionsprozesses können Komponenten- und Lösungskataloge [Lang98] sowie beispielsweise (Bild 8) eine „morphologische Matrix“ (Synonym: morphologischer Kasten) eingesetzt werden [BrFl93]. Sie dient dazu, das vorhandene Lösungsspektrum für einzelne Teilaufgaben (aus Sicht der Fach-Komponenten bzw. Varianten) vollständig und übersichtlich zu präsentieren. Ist im Sortiment der Fach-Komponenten (Komponenten-Kataloge) keine entsprechende „Variante“ zur Unterstützung oder Erfüllung einer Teilaufgabe vorhanden, kann mit dem vorliegenden Anforderungsprofil ein sehr konkreter Auftrag zur Konstruktion einer neuen Fach-Komponente (Variante) erteilt werden.

Zu 2: Ein weiteres Instrument zur Entwicklung von Anwendungssystemen aus Fach-Komponenten, die in einer hohen Variantenzahl vorliegen, sind Stücklisten [Grup95]. In Bild 9 werden bei „Fach-Komponenten“ (Application Objects) „Anwendungselemente“ (sie spezifizieren den fachfunktionalen Teil, die „Fachsemantik“ einer Komponente) und „Creatorelemente“ (sie dienen der Implementierung einer fachlichen Lösung) unterschieden [KaLO99]. Creatorelemente bestehen aus ausführbarem, gekapseltem Programmcode und/oder aus kleiner granulierten Creatorelementen (Bild 9). Daraus lässt sich eine Erzeugnisstruktur für Anwendungssysteme angeben, die mit Variantenstücklisten [WeMü81] in der technischen Konstruktion vergleichbar ist.

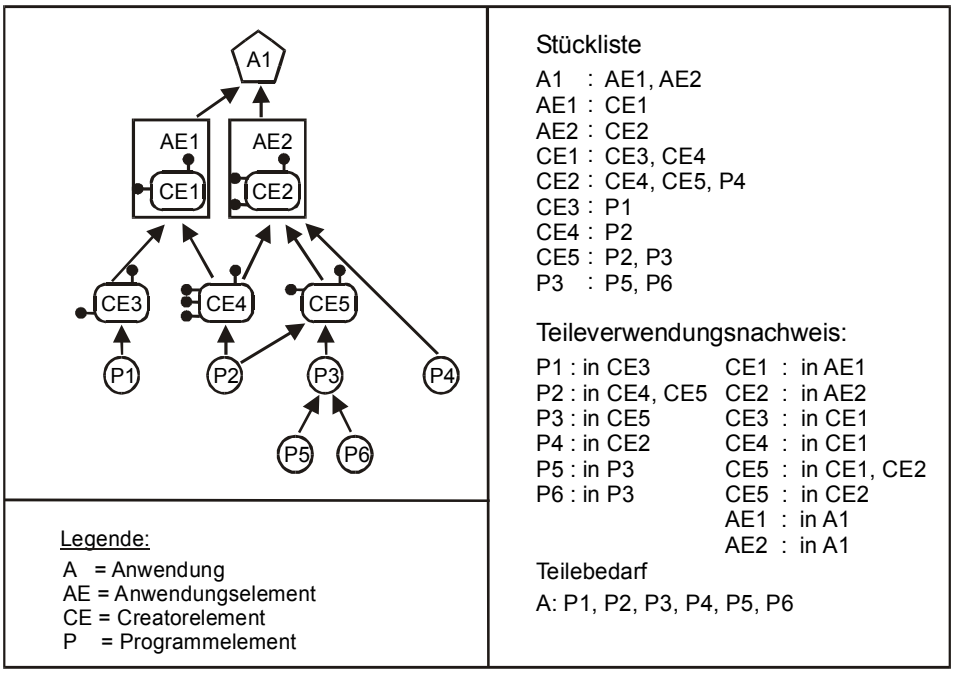

Bild 9: Erzeugnisstruktur einer kompentenbasierten Anwendung

Erzeugnisstrukturen (Stücklisten) unterstützen das Management von Varianten, die alternative Teillösungen für Teilprobleme anbieten. Geschlossene Variantenstücklisten mit Ergänzungsstücklisten oder Grundstücklisten mit Plus-Minusergänzungsstücklisten, sowie offene Variantenstücklisten werden heute [Grup95] unterschieden. Bei der Speicherungsorganisation von offenen Variantenstücklisten wird zwischen einer auftragsneutralen Stammstückliste mit dem maximalen Stücklistenumfang (einschließlich Platzhalter für noch nicht vollständig definierte Positionen) und einer Kundenauftragsstückliste unterschieden. Kombinationsmöglichkeiten und Kombinationsrestriktionen werden ebenfalls in den Stücklisten verwaltet, so dass mögliche Fach-Komponenten mit Hilfe eines Variantengenerators evaluiert und zu Anwendungssystemen komponiert werden können.

Zu 3: Für die Kopplung von Fach-Komponenten zu Anwendungssystemen kommen verschiedene „Techniken" in Frage. In der objektorientierten Anwendungssystementwicklung bietet sich beispielsweise in der Programmiersprache „Java" die sogenannte „Infobus-Technologie" durch das Versenden und Empfangen von "Ereignissen" zwischen den Komponenten an. Daneben stehen Kopplungsmechanismen wie der ORB (Object Request Broker) bei CORBA oder DCOM (von Microsoft) zur Verfügung. Skriptsprachen können gewissermaßen als „Kleber" zwischen den Komponenten eingesetzt werden, oder die Verbindung zwischen den Komponenten wird über „Namens- und Verzeichnisdienste" (Repositorien) realisiert.

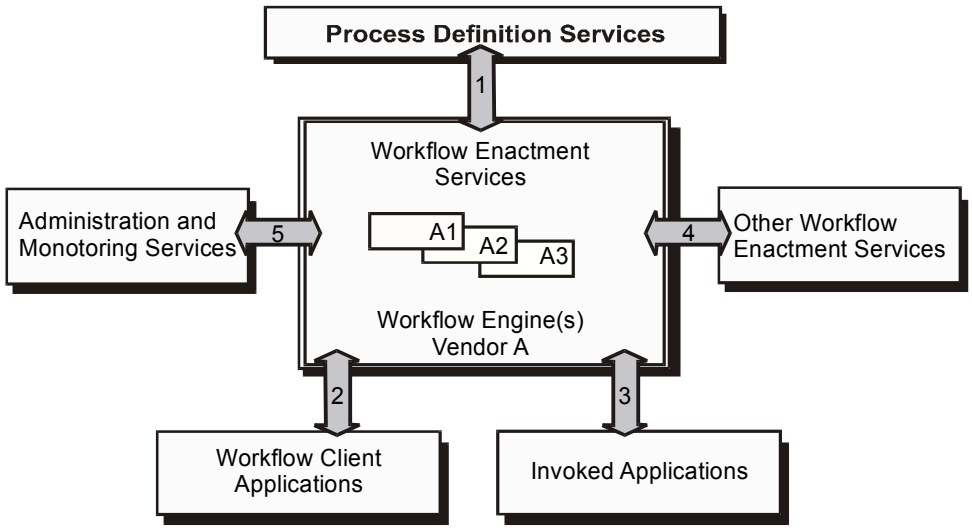

Bild 10: Referenzmodell der Workflow-Management Coalition (WfMC) für ein Workflow-Management-System [Lawr97]

Ein flexibles Instrument zur Verbindung von (Fach-) Komponenten zu Anwendungssystemen stellen „Workflow-Management-Systeme" [JaBS97] dar. Mit einem Workflow-Management-System (Bild 10) können nicht nur die Fach-Komponenten (Invoked Applications), sondern auch ihre „Benutzer" (Workflow Client Applications) zu ganzen Anwendungssystemen (u.a. auch mit Arbeitspersonen als „Komponenten") verbunden werden. Eine Verbindung zwischen den Komponenten wird dabei auf der „Steuerungs-" (Aktivitätenreihenfolgen kooperierender Komponenten) und auf der „Datenebene" (Kommunikation oder Nachrichtenaustausch zwischen den Komponenten) realisiert. „Workflow Client Applications" sind Einheiten, die die Kommunikation der „Workflow Engine" (Steuerungseinheit) mit einem Aufgabenträger

(z. B. einem Sacharbeiter) in Form einer Arbeitsvorratsliste (worklist) übernehmen. „Invoked Applications“ (Bild 10) sind dagegen relativ selbstständige „Fach-Komponenten“ (z. B. ein Rechnungsschreibungsprogramm), die im Rahmen der Ausführung von Workflows von der „Engine“ aufgerufen werden, wobei das Workflow-Management-System (WfMS) auf die konkrete Ausführung der Teilaufgabe keinen Einfluss hat, sondern lediglich die Ergebnislieferung kontrolliert und die Verbindung „Daten- und Kontrollfluss“ zwischen den Komponenten realisiert.

„Process Definition Services“ (Bild 10) sind Werkzeuge zur Entwicklung von Workflow-Management-Anwendungen. Die „Administration and Monitoring Services“ stehen dem WfMS-Administrator zur Planung, Kontrolle, Optimierung und Steuerung des WfMS-Betriebs zur Verfügung. Über das „Interface 4“ (Other Workflow Enactment Services) wird die Möglichkeit geboten, verteilte WfMS und WfM-Anwendungen – z. B. über ein unternehmensübergreifendes Repositorium [KuSc99] - zu entwickeln und zu betreiben.

6 Ausblick

Die industrielle Herstellung von Anwendungssystemen aus (Fach-) Komponenten wird sich dann durchsetzen, wenn sich ein Komponenten-Markt [KaHa99] entwickelt. Dazu müssen die „Marktplätze“ für eine Komponentenindustrie in der sich abzeichnenden „globalen Wissensbranche“ organisiert werden. Hierbei können das Internet und das World Wide Web eine wichtige Rolle spielen. Eine zentrale Komponente solcher Marktplätze sind Repositorien [z. B. Ortn99]. Dies wird zu einem Bedeutungsanstieg der Metainformationsverarbeitung in den Unternehmen führen.

Zur Erzielung integrierter Lösungen werden neben Unternehmensrepositorien vor allem Normungsrepositorien an Bedeutung gewinnen. Die „Verbindung“ der Anwendungen zu global interagierenden Anwendungssystemen wird sich dabei von einer Datenschema-orientierten Integration zu einer terminologiebasierten Integration [z. B. LiKS99] der Anwendungen verlagern.

Der Standardisierungsbedarf für die „Inhalte“ der Informationsverarbeitung wird hinsichtlich der Rechnerunterstützung in den Unternehmen zunehmen. Dies geben beispielsweise Projekte zur Entwicklung „domänenspezifischer Sprachen“ im Zusammenhang mit der Web-Sprache XML (Extensible Markup Language) klar zu erkennen. Der Wirtschaftsinformatik fällt in diesem Zusammenhang die Aufgabe zu, diese „Inhalte“ [z. B. Wede93] auf der Ebene der Anwenderfachsprachen in Form von „Fach-Terminologien“, „Fach-Normsprachen“, „Fach-Referenzmodellen“ oder „Fach-Komponenten“ methodisch zu rekonstruieren.

Literatur:

[ANX375] *ANSI/X3SPAC:* Study Group on Data Base Management, Systems – Interim – Report, in: Bulletine of ACM SIGMOD, 7(1975)2.

[Bern99] *Bernstein, P. A.; Bergstraeser, T.; Carlson, J.; Pal, S.; Sanders, P.; Shult, D.:* Microsoft Repository Version 2 and the Open Information Model, in: Information Systems, 24 (1999) 2, S. 71 – 98.

[BrFl93] *Breiing, A.; Flemming, M.:* Theorie und Methoden des Konstruierens, Springer–Verlag, Berlin 1993.

[Fran97] *Frank, U.:* Enriching Object–Oriented Methods with Domain Specific Knowledge: Outline of a Method for Enterprise Modeling, Arbeitsbericht des Instituts für Wirtschaftsinformatik, Nummer 4, Koblenz 1997.

[Grit98] *Griffel, F.:* Componentware: Konzepte und Techniken eines Softwareparadigmas, dpunkt–Verlag, Heidelberg 1998.

[Grup95] *Grupp, B.:* Aufbau einer optimalen Stücklistenorganisation: offene Stücklisten, Variantengenerator, PPS – Rahmen, CAD – Connection, Praxisbeispiele, expert–Verlag, Rennigen – Malmsheim 1995.

[JaBS97] *Jablonski, S.; Böhm, M.; Schulze, W. (Hrsg.):* Workflow–Management, Entwicklung von Anwendungen und Systemen, dpunkt–Verlag, Heidelberg 1997.

[KaHa99] *Kaufmann, T.; Hau, M.:* Entwurf eines Marktplatzes für betriebswirtschaftliche Software–Bauteile, in: Oberweis, A.; Sneed, H.: (Hrsg.); Software–Management ’99, B. G. Teubner – Verlag, Stuttgart/Leipzig 1999, S.117 – 135.

[KaLO95] *Kalkmann, J.; Lang, K.-P.; Ortner, E.:* Anwendungssystementwicklung mit Komponenten, in: Information Management & Consulting, 14 (1999) 2, S.35 – 45.

[KuSc99] *Kurbel, K.; Schwarz, Ch.:* Unterstützung des zwischenbetrieblichen Workflowmanagements durch unternehmensübergreifendes Repository, in: Information Management & Consulting, 14 (1999) 4, S. 75 – 81.

[Lang98] *Lang, K.-P.:* Variantenkonstruktion betriebswirtschaftlicher Anwendungssoftware, Arbeitsbericht 98/02 des Fachgebiets Wirtschaftsinformatik I, Entwicklung von Anwendungssystemen, Technische Universität Darmstadt, Darmstadt 1998.

[Lawr97] *Lawrence, P.(Hrsg):* Workflow–Handbook der Workflow–Management–Coalition, Wiley, Chicester 1997.

[Lehm97] *Lehmann, F. R.:* Fachlicher Entwurf von Workflow–Management–Anwendungen, B. G. Teubner Verlag, Stuttgart/Leipzig 1999.

[LiKS99] *Ließmann, H.; Kaufmann, T.; Schnitzer, B.:* Bussysteme als Schlüssel zur betriebswirtschaftlich–semantischen Kopplung von Anwendungssystemen, in: Wirtschaftsinformatik, 41 (1999) 1, S. 12 – 19.

[Lore73] *Lorenzen, P.:* Semantisch normierte Orthosprachen, in Kambartel, F.; Mittelstraß, J. (Hrsg.): Zum normativen Fundament der Wissenschaften, Athenäum – Verlag, Frankfurt/M 1973, S. 231 – 249.

[LorK92] *Lorenz, K.:* Einführung in die philosophische Anthropologie, 2., unveränderte Auflage, Wissenschaftliche Buchgesellschaft, Darmstadt 1992.

[Mert97] *Mertens, P. et al.:* Formen integrierter betrieblicher Anwendungssysteme zwischen Individual- und Standardsoftware, Forschungsbericht des Bayrischen Forschungszentrums für wissensbasierte Systeme (FORWISS), FR–1997–005, Erlangen 1997.

[OMGr96] *Object Management Group:* Object Management Architecture, Chapter 5.2 of the new OMG Guide, OMG Document AB/96–08–01, August 1996.

[OMGr97] *Object Management Group:* Meta Object Facility Specification, Joint Revised Submission, OMG Document ad/97–08-14, September 1997.

[Ortn97] *Ortner, E.:* Methodenneutraler Fachentwurf, Zu den Grundlagen einer anwendungsorientierten Informatik, B. G. Teubner Verlag, Stuttgart/Leipzig 1997.

[Ortn99] *Ortner, E.:* Repository Systems – Aufbau und Betrieb eines Entwicklungsrepositoriums, in: Informatik – Spektrum, 22 (1999) 4, S. 235 – 251 und in: Informatik – Spektrum, 22 (1999) 5, S.351 – 363.

[Sche95] *Scheer, A.-W.:* Wirtschaftsinformatik – Referenzmodelle für industrielle Geschäftsprozesse, Springer – Verlag, Berlin/Heidelberg/New York 1995.

[Schi97] *Schienmann, B.:* Objektorientierter Fachentwurf, ein terminologiebasierter Ansatz für die Konstruktion von Anwendungssystemen, B. G. Teubner Verlag, Stuttgart/Leipzig 1997.

[Schu99] *Schulze, W.:* Ein Workflow–Management–Dienst für ein verteiltes Objektverwaltungssystem, Dissertation, Technische Universität Darmstadt, Dresden, Fakultät Informatik, März 1999.

[Ston96] *Stonebraker, M.:* Object–Relational DBMSs – The Next Great Wave, Morgan Kaufmann, San Francisco 1996.

[Taps98] *Tapscott, D.:* Net Kids: Die digitale Generation erobert Wirtschaft und Gesellschaft, Verlag Dr. Th. Gabler GmbH, Wiesbaden 1998.

[Vogl94] *Vogler, M.:* OTMAR – Ein automatischer Übersetzer zur Konstruktion von Objekttypen und Beziehungen aus Aussagen, Diplomarbeit, Universität Konstanz, Fachgruppe Informationswissenschaft, November 1994.

[Wede93] *Wedekind, H.:* Kaufmännische Datenbanken, B. I. – Wissenschaftsverlag, Mannheim 1993.

[WeMü81] *Wedekind, H.; Müller, T.:* Stücklistenorganisation bei einer großen Variantenzahl, in: Angewandte Informatik, Heft 9, Sept. 1981, S. 377 – 383.

[WeOr77] *Wedekind, H.; Ortner, E.:* Der Aufbau einer Datenbank für die Kostenrechnung, in: Die Betriebswirtschaft, 37 (1977) 4, S. 533 – 542.