Betriebliche Informationssysteme

Kurseinheit 2

Architektur von betrieblichen Informationssystemen

Autor:
Prof. Dr. Lars Mönch
unter Mitarbeit von:
Dr. Jens Zimmermann und
Dr. René Drießel

Architektur von betrieblichen Informationssystemen

Inhaltsübersicht

Lernziele

Die vorliegende Kurseinheit soll Sie neben dem Architekturbegriff für Informationssysteme mit einem entsprechenden generischen Architekturrahmen vertraut machen. Unter einer Architektur versteht man die grundlegende Organisation eines Systems, dargestellt durch dessen Komponenten, deren Beziehungen zueinander und zur Umgebung. Sie sollen verschiedene Beispiele für Architekturen von betrieblichen Informationssystemen kennenlernen. Kurseinheit 2 dient als Grundlage für die nachfolgenden Kurseinheiten zur Konstruktion von betrieblichen Informationssystemen sowie zu Anwendungssystemen.

Nach dem Studium von Abschnitt 2.1 sollen Sie sicher mit dem für die Wirtschaftsinformatik charakteristischen Architekturbegriff umgehen können.

Wenn Sie Abschnitt 2.2 durchgearbeitet haben, werden Sie in der Lage sein, den von Sinz [53] vorgeschlagenen Architekturrahmen charakterisieren zu können.

Nach dem Studium von Abschnitt 2.3 können Sie wesentliche Eigenschaften des Semantischen Objektmodells benennen und Geschäftsprozesse mit SOM modellieren. Den Zusammenhang zwischen dem Semantischen Objektmodell und der Architektur von Anwendungssystemen haben Sie erkannt. Sie sind sicher im Umgang mit den Modellebenen und Sichten der Architektur integrierter Informationssysteme. Die Prinzipien dienstorientierter Informationssysteme und service-orientierter Architekturen verstehen Sie.

Die Beschäftigung mit Abschnitt 2.4 der vorliegenden Kurseinheit wird Sie dazu befähigen, die Nutzung von Informationssystemarchitekturen in Unternehmen zu verstehen.

Das Studium von Abschnitt 2.5 wird Sie in die Lage versetzen, Qualitätsaspekte von Informationssystemarchitekturen zu verstehen und die Qualität von Informationssystemarchitekturen bewerten zu können.

Schließlich sollen Sie nach dem Durcharbeiten von Abschnitt 2.6 zum einen ein prinzipielles Verständnis für Softwarearchitekturen für betriebliche Anwendungssysteme entwickelt haben, andererseits auch in der Lage sein, eigenständig entsprechende Architekturen für Anwendungssysteme zu entwerfen.

2.1 Architekturbegriff

Architektur

In der Wirtschaftsinformatikliteratur besteht weitestgehend Einigkeit über die Bedeutung der Informationssystemarchitektur und über die Zweckmäßigkeit der Verwendung des Architekturbegriffes [4].

Scheer bezeichnet die Komponenten eines Informationssystems und ihre Beziehungen als Architektur [48]. Österle definiert die Informationssystemarchitektur als Grobstruktur der Organisation, der Geschäftsfunktionen, der Daten, der Anwendungen und der Datenbanken, die den konzeptionellen Rahmen für die Entwicklung der Organisation, der Anwendungen und der Datenbanken bilden [55]. In dieser Kurseinheit wird der von Sinz in [53] gegebenen Definition gefolgt.

Definition 2.1.1 (Informationssystemarchitektur) *Eine Informationssystemarchitektur ist*

- *ein Bauplan des Informationssystems im Sinne einer Spezifikation und Dokumentation seiner Komponenten und ihrer Beziehungen unter allen relevanten Blickwinkeln,*
- *sowie eine Menge der Konstruktionsregeln zur Erstellung dieses Bauplans.*

Modell und Modellsystem

Der Bauplan stellt dabei ein Abbild des Informationssystems dar. Er führt zu einem Modellsystem, dessen Objektsystem das eigentliche betriebliche Informationssystem darstellt (vergleiche Definition 1.1.12 für die Begriffe Objekt- und Modellsystem). Unter einem Modell verstehen wir dabei, wie in Definition 1.1.12 festgelegt, ein abstraktes immaterielles Abbild realer Strukturen bzw. des realen Verhaltens, das subjektiv auf den Modelladressaten zugeschnitten ist. Modelle sind zweckorientiert, da mit der Erstellung eines Modells typischerweise das Erreichen eines bestimmten Ziels verknüpft ist.

Die Konstruktionsregeln sind als Vorschriften zur Erstellung des Modellsystems aufzufassen. Sie werden typischerweise in Form von Metamodellen angegeben. Wir verweisen auf Definition 1.1.15 für Metamodelle aus Abschnitt 1.1.3.

Bei der Modellierung betrieblicher Informationssysteme unterscheiden wir zwei unterschiedliche Ebenen [53]:

Aufgaben- und Aufgabenträgerebene

1. **Aufgabenebene**: Diese Ebene umfasst die zu lösenden Informationsverarbeitungsaufgaben. Die Aufgaben sind durch Informationsbeziehungen sowie durch zeitliche Vorrangbeziehungen miteinander verknüpft.
2. **Aufgabenträgerebene**: Diese Ebene wird durch Menschen und durch Rechner- und Kommunikationssysteme gebildet. Die Aufgabenträger lösen Aufgaben und kommunizieren zu diesem Zweck miteinander. Die zu lösenden Informationsverarbeitungsaufgaben werden dabei gegebenenfalls kooperativ ausgeführt.

Wir betrachten das nachfolgende Beispiel zur Veranschaulichung der Aufgabenebene.

Beispiel 2.1.1 (Beziehungen zwischen Aufgaben) *Die Aufgabe, automatisch zu bestimmten Zeitpunkten einen Kapazitätsabgleich durchzuführen, setzt voraus, dass vorher die Aufgabe der Bereitstellung von Daten über die aktuelle Belastung der Maschinen im Produktionssystem gelöst wird.*

Eine Architektur zu besitzen, bedeutet für die Informationstechnologie, eine Konzeptualisierung vorgenommen zu haben, die eine zielgerichtete und effiziente Entwicklung, Wartung, Dokumentation und Nutzung von Informationssystemen ermöglicht.

Wir unterscheiden zwischen Informationssystemen, die der Leistungserstellung durch den Anwender dienen, Basissoftware wie relationale Datenbanken und Betriebssysteme sowie Hardwaresystemen. Die technische Informationsinfrastruktur eines Unternehmens wird durch Informationssysteme, Basissoftware und Hardware gebildet. Entsprechend der Einteilung in Informationssysteme, Basissoftware und Hardware unterscheiden wir zwischen drei Klassen von Architekturen [26]: Architekturklassen

1. **Informationsarchitekturen**: Diese Art von Architektur stellt die Beziehungen zwischen Teilen eines betrieblichen Informationssystems oder zwischen unterschiedlichen betrieblichen Informationssystemen her.

2. **Softwarearchitekturen**: Sie stellen den Zusammenhang zwischen unterschiedlichen Elementen eines Softwaresystems dar [24].

3. **Rechnerarchitekturen**: In Analogie zu einer Softwarearchitektur beschreibt eine Rechnerarchitektur den Zusammenhang zwischen Hardwarekomponenten.

Diese drei Architekturen bilden in ihrer Einheit die Informationssystemarchitektur. Diese ist die Grundlage für die Entwicklung, Pflege und Wartung der Informationsinfrastruktur eines Unternehmens. In Abschnitt 2.2 werden wir für den dort vorgestellten Architekturrahmen zeigen, dass die drei Architekturklassen bestimmte Blickwinkel der Informationssystemarchitektur darstellen.

2.2 Architekturrahmen

2.2.1 Modellebenen und Blickwinkel

Wir beschreiben im Folgenden den Architekturrahmen von Sinz [53]. Sinz schlägt vor, dass eine Informationssystemarchitektur aus unterschiedlichen Modellebenen besteht. Wir definieren dazu zunächst den Begriff der Modellebene.

Definition 2.2.1 (Modellebene) *Unter einer Modellebene verstehen wir die vollständige Beschreibung eines Informationssystems unter einem bestimmten Blickwinkel.* Modellebene

Blickwinkel

Jeder Modellebene muss somit ein bestimmter Blickwinkel zugeordnet sein. Ein einzelner Blickwinkel unterstützt mindestens eines der mit der Modellbildung verfolgten Ziele. Wir stellen verschiedene Ziele im nachfolgenden Beispiel vor.

Beispiel 2.2.1 (Ziele der Modellbildung) *Mögliche Ziele der Modellierung sind die Erfassung und Beschreibung des Datenhaushalts eines Informationssystems, die Darstellung von zeitlichen Abhängigkeiten zwischen der Aufgabenerfüllung von unterschiedlichen Subsystemen eines Informationssystems (Vermeiden von Dead- und Livelocks) sowie die Beschreibung der Kommunikationsbeziehungen innerhalb eines verteilten Informationssystems.*

Mögliche Blickwinkel sind nach Sinz [53]:

- Aufgabenblickwinkel,
- Aufgabenträgerblickwinkel,
- softwaretechnischer Blickwinkel,
- Blickwinkel der Außen- und Innenbetrachtung des Informationssystems.

Aufgabenblickwinkel

Nachfolgend werden die unterschiedlichen Blickwinkel genauer erläutert. Der **Aufgabenblickwinkel** stellt die Fachkonzeptebene dar. Ein Fachkonzept beschreibt wesentliche Aufgaben des Informationssystems unter fachlichem Gesichtspunkt, klammert aber technische Fragestellungen weitestgehend aus. Zu den technischen Aspekten gehören insbesondere auch Implementierungsfragestellungen. Das Fachkonzept spezifiziert Geschäftsprozesse, die durch das betriebliche Informationssystem unterstützt werden sollen. Ein Geschäftsprozess besteht aus mindestens einem Geschäftsvorfall. Ein Geschäftsvorfall dient der Veränderung von mindestens einem Geschäftsobjekt (z.B. durch eine Geschäftstransaktion). Jeder Geschäftsvorfall setzt sich aus einem oder mehreren Schritten zusammen. Jeder einzelne Schritt wird vollständig, partiell oder gar nicht von dem zu entwickelnden betrieblichen Informationssystem unterstützt.

Aufgabenträgerblickwinkel

Der **Aufgabenträgerblickwinkel** wird durch zwei Modellebenen repräsentiert. Die erste Modellebene dient der Modellierung maschineller Aufgabenträger. Maschinelle Aufgabenträger sind durch Computer und Software gegeben. Folglich gehören ein Softwarekonzept und ein Konzept für Systemplattformen zu dieser Modellebene. Die zweite Modellebene, die diesem Blickwinkel zugeordnet ist, ist die Modellebene für personelle Aufgabenträger. Diese Ebene wird durch das Organisationskonzept abgebildet.

Der **softwaretechnische Blickwinkel** umfasst die Softwarearchitektur des Informationssystems. Unter der Softwarearchitektur wird an dieser Stelle der Zusammenhang zwischen Elementen von Softwaresystemen verstanden. Daten- sowie Funktionsmodelle (bzw. Objektmodelle) werden durch diesen Blickwinkel erfasst. Außerdem wird die Unterteilung in fachliche und technische Komponenten beschrieben. Ein Softwaresystem besteht aus technischen Einheiten, die zur

Erstellung des Laufzeitsystems führen. Dazu gehören beispielsweise Quellcode, statische und dynamische Bibliotheken, Skripte und ausführbare Programme. Der softwaretechnische Blickwinkel erfasst stets auch diese Erstellungssicht.

Der Blickwinkel der **Außenbetrachtung** des Informationssystems ist durch einen Anwender des Informationssystems direkt oder zumindest indirekt einzunehmen. Die Dialog-Benutzerschnittstelle wird durch diesen Blickwinkel erfasst. Die Benutzerschnittstelle wird durch die Gesamtheit der möglichen Dialoge gebildet. Neben der Dialog-Benutzerschnittstelle existiert die Batch-Benutzerschnittstelle. Die Außenbetrachtung umfasst auch die Interaktion des Informationssystems mit Nachbarsystemen, da Informationssysteme in einem Unternehmen stets Bestandteil einer Systemlandschaft sind.

Außenbetrachtung

Der Blickwinkel der **Innenbetrachtung** des Informationssystems wird durch Architekten, Systementwickler bzw. Systembetreiber eingenommen. Die Innensicht wird unter anderem durch die Betreibersicht gebildet. Im Wesentlichen geht es dabei um Fragen der Installation und Deinstallation des Informationssystems, um Fragen der Datensicherung, der Sicherheit und um zu ergreifende Maßnahmen bei unerwarteten Zwischenfällen. Bei der physischen Sicht werden die physischen Geräte beschrieben, die ein Informationssystem entweder alleine oder gemeinsam mit anderen Informationssystemen benutzt. Dazu gehören unter anderem Leitungen, Speichermedien und Drucker. Informationssysteme bestehen zur Laufzeit aus einer Reihe von Prozessen sowie Dateien. Die das Informationssystem bildenden Prozesse kommunizieren miteinander. Die Steuerung der Prozesse wird ebenfalls im Rahmen der Innenbetrachtung des Informationssystems erfasst.

Innenbetrachtung

2.2.2 Modellierungskonzepte

Ein Modellierungskonzept beschreibt, wie eine bestimmte Klasse von Modellen systematisch erzeugt werden kann. Unterschiedliche Modellierungskonzepte werden in den Kurseinheiten über die Konstruktion von Informationssystemen ausführlicher dargestellt. Für die Zwecke dieses Abschnittes ist eine Unterscheidung in klassische, objektorientierte und agentenbasierte Modellierungskonzepte ausreichend.

Modellierungskonzept

Klassische Modellierungskonzepte sind durch Konzepte zur Daten- und zur Funktionsmodellierung gegeben. Dabei ist wesentlich, dass Daten- und Funktionsmodelle unabhängig voneinander entwickelt werden.

Definition 2.2.2 (Datenmodellierung) *Die Datenmodellierung dient der Beschreibung der Struktur der Datenbasis des Informationssystems. Die Datenbasis wird durch Datenobjekttypen, denen Attribute zugeordnet sind, repräsentiert. Datenobjekttypen sind durch Beziehungen untereinander verbunden.*

Datenmodellierung

Die Beschreibung der Struktur der Datenbasis des Informationssystems ist wesentlich für die Festlegung der Funktionen des Informationssystems. Die dazu

notwendige Beschreibung der Funktionen wird durch die funktionale Dekomposition und die Strukturierte Analyse unterstützt.

Funktionale Dekomposition

Definition 2.2.3 (Funktionale Dekomposition) *Unter der funktionalen Dekomposition verstehen wir die mehrstufige Zerlegung der Funktionen eines Informationssystems in Teilfunktionen. Dabei werden gleichzeitig Schnittstellen zu anderen Teilfunktionen, zu Komponenten des Basissystems sowie zur Umwelt des Informationssystems festgelegt.*

Die von den Funktionen bearbeiteten Daten werden als eigenständige Datenstrukturen unabhängig von den Funktionen modelliert. Das stellt eine wesentliche Schwäche der funktionalen Dekomposition dar.

Strukturierte Analyse

Definition 2.2.4 (Strukturierte Analyse) *Im Rahmen der Strukturierten Analyse wird ein Informationssystem als Menge von Datenflüssen modelliert. Datenflüsse sind Kanäle für Informationen, die zwischen zwei Funktionen, zwischen einer Funktion und einem Datenspeicher sowie zwischen der Systemumgebung und dem betrachteten System auftreten können. Datenflüsse werden durch hierarchisch verfeinerbare Funktionen transformiert. Datenspeicher werden als temporärer Aufenthaltsort für Daten genutzt. Funktionen, die nicht mehr weiter verfeinerbar sind, werden in Mini-Spezifikationen definiert.*

Objektorientierte Modellierung

Da Funktionen auf Daten leben, ist eine Trennung von Funktions- und Datenmodellierung nicht sinnvoll. Das führte zur Entwicklung objektorientierter Modellierungskonzepte.

Definition 2.2.5 (Objektorientierte Modellierung) *Bei der objektorientierten Modellierung wird die Aufgabenebene eines Informationssystems als Menge von Objekttypen angegeben. Ein Objekttyp wird durch Attribute sowie durch Methoden festgelegt. Eine Wertbelegung der Attribute dient der Beschreibung des Zustands eines Objektes. Durch Methoden werden Objekte eines Objekttyps verändert. Nachrichten werden für die Kommunikation zwischen Objekten spezifiziert. Wenn ein Objekt eine Nachricht erhält, wird eine entsprechende Methode des Objekts ausgeführt.*

Objektorientierte Modellierungskonzepte legen somit die Objekte bzw. deren Beziehungen untereinander fest. Außerdem wird das Verhalten der Objekte sowie der Nachrichtenaustausch der Objekte im zeitlichen Verlauf spezifiziert.

Agentenbasierte Modellierung

Softwareagenten sind eine natürliche Weiterentwicklung von Objekten [61, 62]. Im Gegensatz zu herkömmlichen Objekten sind Agenten durch eigene Ziele sowie durch die Möglichkeit von proaktivem Verhalten gekennzeichnet.

Definition 2.2.6 (Agentenbasierte Modellierung) *Agentenbasierte Modellierungskonzepte beschreiben den Aufbau des Informationssystems aus Agenten, den Lebenszyklus einzelner Agenten sowie die Interaktion der Agenten untereinander.*

Softwareagenten werden an verschiedenen Stellen dieses Kurses genauer behandelt. In Abschnitt 4.1 werden Objekte und Agenten verglichen.

Die Geschäftsprozessmodellierung ermöglicht den Übergang von einer statischen und strukturorientierten Sicht auf das Informationssystem zu einer dynamischen und stärker verhaltensorientierten Sichtweise. Im Gegensatz zu den bisher vorgestellten Modellierungskonzepten ermöglicht die Geschäftsprozessmodellierung eine Ausdehnung des Modellierungsumfangs von der Aufgaben- auf die Aufgabenträgerebene sowie auf das Basissystem.

Geschäftsprozessmodellierung

Definition 2.2.7 (Geschäftsprozessmodellierung) *Die Geschäftsprozessmodellierung dient der Modellierung eines ereignisgesteuerten Ablaufs von Aufgabendurchführungen. Dabei werden die auf die Erreichung von Unternehmenszielen ausgerichtete Leistungserstellung, deren Lenkung sowie die dazu benötigten Ressourcen mit in die Betrachtung einbezogen.*

Durch die später in Abschnitt 2.3.2 dargestellte Architektur integrierter Informationssysteme ist ein typischer Vertreter eines geschäftsprozessorientierten Modellierungsansatzes gegeben. Das in Abschnitt 2.3.1 beschriebene Semantische Objektmodell verbindet objektorientierte mit geschäftsprozessorientierten Modellierungsansätzen.

2.2.3 Darstellung des Architekturrahmens

Im weiteren Verlauf wird angenommen, dass die Modellebenen

Modellierungskonzept

$$M := (M_1, \ldots, M_n) \tag{2.1}$$

vorliegen. Das einer bestimmten Modellebene zugeordnete Modellierungskonzept wird durch ein Metamodell dargestellt. Wir bezeichnen die Menge der Metamodelle mit

$$MM := (MM_1, \ldots, MM_n)\,. \tag{2.2}$$

Somit wird jeder Modellebene genau ein Metamodell zugeordnet.

Wir betrachten zunächst ausschließlich eine feste Modellebene M_i. Die Modellierung einer bestimmten Modellebene wird durch die Einführung von Sichten unterstützt. Die zur Modellebene M_i zugehörigen Sichten werden mit

Sichten

$$S_i := (S_{i1}, \ldots, S_{ik}) \tag{2.3}$$

bezeichnet. Sichten werden ebenfalls wieder durch Metamodelle spezifiziert. Das Metamodell einer Sicht ergibt sich als Projektion auf das Metamodell der Modellebene. Es gilt:

$$Proj(MM_i) := (Proj_1(MM_i), \ldots, Proj_k(MM_i))\,, \tag{2.4}$$

wobei wir mit $Proj(MM_i)$ die Menge der Metamodelle der Sichten der Modellebene M_i bezeichnen. Die Notation $Proj_j(MM_i)$ wird für das Metamodell der Sicht S_{ij} verwendet. Wir betrachten das nachfolgende Beispiel für Metamodelle.

Beispiel 2.2.2 (Metamodelle für Sichten einer Modellebene) *In Abschnitt 2.2.1 wird festgestellt, dass eine Softwarearchitektur Daten- und Funktionsmodelle umfasst. Ein mögliches Metamodell für die Sicht Datenmodell ist durch das Entity-Relationship-Modell (ERM) [9] gegeben. Für Funktionsmodelle wird durch die Structured-Analysis-and-Design-Technique (SADT) [14] ein entsprechendes Metamodell bereitgestellt.*

Eine Sicht stellt die unvollständige Beschreibung einer Modellebene dar. Beispielsweise wurde in Abschnitt 2.2.1 die Innenbetrachtung eines Informationssystems in die Betreiber- und die Laufzeitsicht unterteilt. Während sich die Betreibersicht ausschließlich mit dem Betrieb des betrieblichen Informationssystems beschäftigt, setzt sich die Laufzeitsicht mit der Dekomposition des Informationssystems in kooperierende Prozesse und deren Steuerung auseinander.

Eine vollständige Beschreibung einer Modellebene erhalten wir durch die Angabe aller Sichten. Die Einführung von Sichten stellt somit in erster Linie ein Instrument zur Bewältigung der Modellkomplexität dar. Die verwendeten Sichten werden in Abhängigkeit vom gewählten Modellierungskonzept ausgesucht.

Entwurfsmuster

Für die einzelnen Modellebenen werden Strukturmuster (Pattern) betrachtet. Die ursprünglich aus der Architektur stammende Idee von Designpattern wurde von Coad [8] auf die objektorientierte Softwareentwicklung übertragen. Wir definieren zunächst den Begriff objektorientierter Entwurfsmuster.

Definition 2.2.8 (Entwurfsmuster) *Ein Entwurfsmuster beschreibt auf eine generische Art und Weise das Zusammenwirken von Objektklassen bzw. von einzelnen Objekten bei der Lösung einer speziellen Aufgabe.*

Wir betrachten die nachfolgenden Beispiele für Entwurfsmuster.

Beispiel 2.2.3 (Entwurfsmuster) *In der objektorientierten Softwareentwicklung wird häufig das Singleton-Pattern [18] angewandt. Dieses Muster stellt sicher, dass von bestimmten Klassen zur Laufzeit nur eine einzige Instanz erstellt werden kann. Ein weiteres bekanntes Pattern ist das Model-View-Controller-Pattern [18] für Anwendungssysteme, das die Trennung in Model, mit der Verantwortung für die Anwendung und insbesondere für die Daten, View für die sichtbare Darstellung sowie Controller für die Verarbeitung von Benutzeraktionen vorschlägt [51]. Für weitere Muster im Umfeld verteilter Softwaresysteme verweisen wir auf [49].*

Strukturmuster

Strukturmuster stellen strukturelle Integritätsbedingungen dar. Sie verringern die aufgrund des Metamodells einer Modellebene zulässigen Strukturen für das Modellsystem. Die zur Modellebene M_i zugehörigen Strukturmuster werden wie folgt bezeichnet:

$$P_i := (P_{i1}, \ldots, P_{im}) \,. \tag{2.5}$$

Im nachfolgenden Beispiel ordnen wir ausgewählte Entwurfsmuster als Strukturmuster bestimmten Modellebenen zu.

Beispiel 2.2.4 (Strukturmuster) *Wir betrachten das Model-View-Controller-Pattern, das maßgebend für Standard-Schichtenmodelle betrieblicher Informationssysteme ist und somit als Muster für die Softwarearchitektur zur Verfügung steht. Als zweites Beispiel soll das Gateway-Pattern [15] dienen. Dieses Muster schlägt die Schaffung einer generischen Schnittstelle vor, durch die der Zugriff auf ein externes System über ein systemspezifisches Application-Programming-Interface (API) gekapselt werden kann. Das Gateway-Pattern kann sowohl bei der Spezifikation der Softwarearchitektur als auch bei der Spezifikation der Innen- und Außensicht Verwendung finden.*

Ontologie

Als weitere mögliche Strukturmuster für die Fachkonzeptebene betrachten wir Ontologien.

Definition 2.2.9 (Ontologie) *Eine Ontologie dient der Beschreibung eines Wissensbereiches („Knowledge Domain") unter Verwendung einer standardisierten Teminologie sowie der Beziehungen und Ableitungsregeln zwischen den definierten Begriffen [21].*

Ontologien können nach ihrem Umfang wie folgt klassifiziert werden [22, 16]:

- generische, bereichsübergreifende Ontologien,
- auf bestimmte Anwendungsbereiche bezogene Ontologien (sogenannte Domänenontologien),
- bekannte konzeptuelle Daten- und Klassenmodelle, die mit dem Namen Ontologie aufgewertet werden.

Eine Ontologie umfasst die folgenden beiden Elemente [41]:

1. Konzeptualisierung des untersuchten Anwendungsbereichs, d.h., es wird beschrieben, unter welchem Blickwinkel und wie der Bereich modelliert wird,
2. eine Spezifikation der Konzeptualisierung, d.h. im Wesentlichen eine formale Beschreibung.

Sowohl die Konzeptualisierung des Bereiches als auch die Spezifikation der Konzeptualisierung werden durch die verwendete Modellierungstechnik beeinflusst. Es wird zwischen slot- bzw. frame-basierten Spezifikationen unterschieden. Bei Verwendung einer slot-basierten Ontologiemodellierung werden die Konzepte durch Klassen im objektorientierten Sinn (vergleiche hierzu Abschnitt 3.4.3 dieses Kurses) abgebildet. Die Attribute einer Klasse werden als Slots bezeichnet. Facetten dienen zur Beschreibung der Eigenschaften eines Slots. Durch Facetten wird eine Typisierung von Slots vorgenommen und Wertebereiche für die Slots vorgegeben.

Ontologien werden von Zelewski, Schütte und Siedentopf in [64] insbesondere unter dem Blickwinkel der Betriebswirtschaft und der Wirtschaftsinformatik diskutiert. Konkrete Ontologien für die Fertigungsdomäne sind unter anderem in [57] vorgeschlagen worden.

Beziehungsmetamodell

Nachdem wir bisher ausschließlich eine einzelne Modellebene M_i betrachtet haben, ist es im nächsten Schritt notwendig, die Beziehungen zwischen mehreren Modellebenen zu studieren. Dazu führen wir die nachfolgende Menge von Metamodellen ein:

$$BMM := \{BMM_{ij} \mid M_i \neq M_j\}\,. \tag{2.6}$$

Mit BMM_{ij} wird dabei das Beziehungsmetamodell bezeichnet, das Metaobjekte der Modellebene M_i mit Metaobjekten der Modellebene M_j verbindet. Durch BMM werden die Metamodelle der unterschiedlichen Modellebenen zu einem Gesamt-Metamodell des Modellsystems zusammengefasst. Beziehungsmetamodelle finden bei der Beschreibung von Zuordnungs- und Transformationsbeziehungen zwischen den unterschiedlichen Modellebenen in Abhängigkeit vom ausgewählten Modellierungskonzept Anwendung. Wir erinnern daran, dass Modellebenen zur Beschreibung eines Informationssystems unter einem bestimmten Blickwinkel dienen. Wir betrachten das nachfolgende Beispiel.

Beispiel 2.2.5 (Beziehungsmetamodelle) *Den Geschäftsobjekten, die zur Modellierung der Geschäftslogik der Anwendung dienen und die im Rahmen des Fachkonzeptes spezifiziert werden, können die entsprechenden Objekte der Datenmodellebene zugeordnet werden, die wiederum im Rahmen des DV-Konzeptes untersucht werden.*

In Analogie zu den Modellebenen schlägt Sinz [53] als letzten Bestandteil des generischen Architekturrahmens die Einführung von Beziehungsstrukturmustern vor, die als Muster für die Beziehungen zwischen unterschiedlichen Modellebenen dienen. Die zur Beschreibung der Beziehungen zwischen Modellebene M_i und M_j zuständigen Pattern werden mit $PBMM_{ij}$ bezeichnet. Für die Gesamtheit aller Beziehungsstrukturmuster gilt:

$$PBMM := \{PBMM_{ij} \mid M_i \neq M_j\}\,. \tag{2.7}$$

Wir geben nun ein Beispiel für ein Beziehungsstrukturmuster an.

PROSA

Beispiel 2.2.6 (Beziehungsstrukturmuster) *Wir betrachten die* **P***roduct-***R***esource-***O***rder-***S***taff-***A***rchitecture (PROSA) [58] für den Bau von holonischen Produktionssteuerungssystemen. Ein Holon wird definiert als eine autonome und kooperative Einheit eines Produktionssystems, die zur Transformation, dem Transport und dem Speichern von physikalischen und Informationsobjekten dient [34]. Holonen bestehen aus einem Steuerungsteil und einem (optionalen) Teil zum Ausführen von physischen Operationen [35]. Ein Holon kann selber aus anderen Holonen bestehen. Unter einer Holarchie wird ein aus Holonen bestehendes System verstanden, in dem die Holonen kooperieren, um ein bestimmtes Ziel zu*

erreichen. Holonische Produktionssteuerungssysteme werden typischerweise als Multi-Agenten-Systeme implementiert (vergleiche den Abschnitt 4.1 dieses Kurses für eine Einführung in Softwareagenten und Multi-Agenten-Systeme). PROSA definiert als Referenzarchitektur die abstrakten Entscheideragententypen

- *Auftragsagent,*
- *Ressourcenagent,*
- *Produktagent*

und legt deren Aufgaben fest. Neben Entscheideragententypen sind in PROSA Dienstagenten vorgesehen. Dienstagenten unterstützen Entscheideragenten bei der Lösung der formulierten Entscheidungsaufgaben. Das Verhalten von Agenten der einzelnen Agententypen wird beschrieben. Interaktionen zwischen den Agententypen werden dargestellt. PROSA schlägt einen hierarchischen Aufbau der Ressourcenagenten vor. Damit sind Hierarchien von Entscheideragenten möglich. PROSA beschreibt auf generische Art und Weise die Aufgaben, die durch ein holonisches Produktionssteuerungssystem erfüllt werden müssen. Diese Teile von PROSA sind dem Fachkonzept zuzuordnen. Außerdem wird, wenn auch nicht detailliert, die Komponentensicht eingenommen, die der Softwarearchitektur zuzurechnen ist. Das Beziehungsstrukturmuster ist dadurch gegeben, dass die im Fachkonzept beschriebenen Aufgaben den entsprechenden Softwareagenten zugeordnet werden.

Abbildung 2.1 dient der Darstellung des vorgestellten generischen Architekturrahmens.

Zusammenfassend kann die Festlegung einer konkreten Informationssystemarchitektur durch die folgende Schrittfolge erreicht werden:

1. Bestimmung der unterschiedlichen Modellebenen,
2. Festlegung der zu verwendenden Modellierungskonzepte,
3. Spezifikation der zu unterstützenden Sichten pro Modellebene,
4. Auswahl von geeigneten Strukturmustern zur Modellierung der einzelnen Modellebenen,
5. Beschreibung von Beziehungen zwischen den einzelnen Modellebenen,
6. Auswahl von geeigneten Beziehungsstrukturmustern.

Wir werden bei der Diskussion ausgewählter Architekturen sehen, dass es nicht notwendig ist, stets alle Schritte zu durchlaufen.

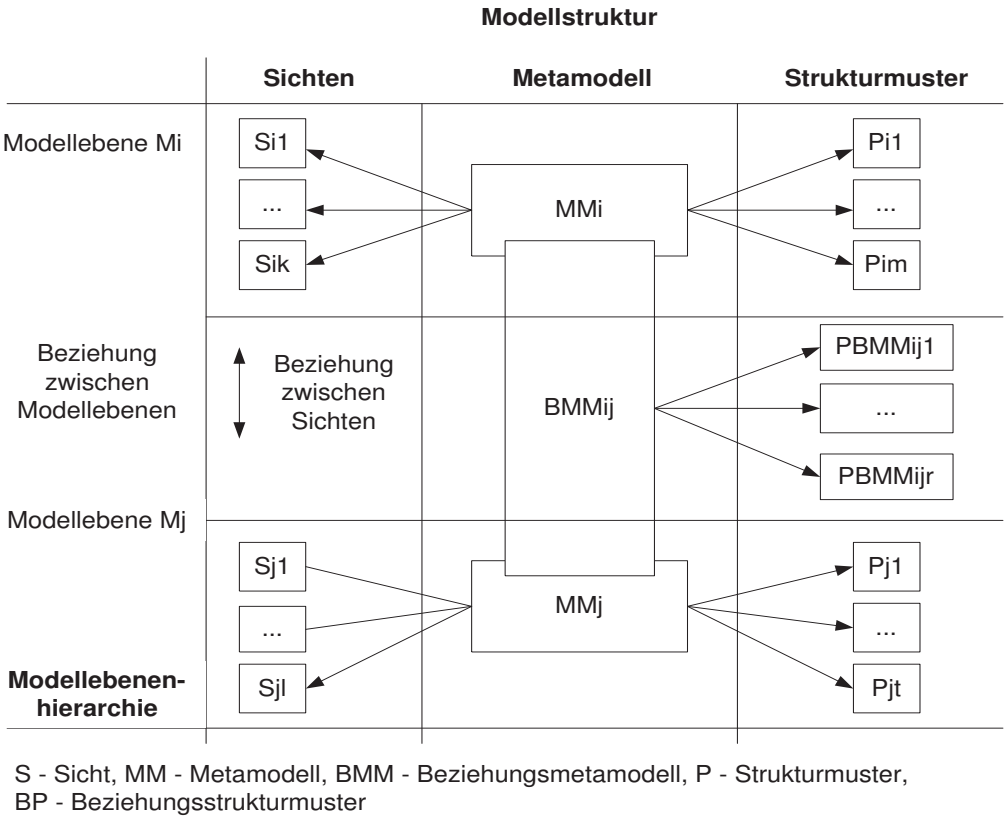

Abbildung 2.1: Generischer Architekturrahmen in Anlehnung an [53]

2.3 Ausgewählte Architekturbeispiele

Wir beschreiben in diesem Abschnitt das Semantische Objektmodell sowie die Architektur integrierter Informationssysteme. Anschließend führen wir den Begriff der dienstorientierten Informationssysteme ein. Als moderne Entwicklungsrichtung stellen wir abschließend service-orientierte Architekturen vor.

2.3.1 Semantisches Objektmodell

SOM

Das Semantische Objektmodell (SOM) wurde als generisches Modellierungskonzept für Informationssysteme von Ferstl und Sinz [12] entwickelt. SOM unterstützt die Analyse, das Design und Redesign von Unternehmen und den dazugehörigen Informationssystemen. SOM verbindet objektorientierte mit geschäftsprozessorientierten Modellierungsansätzen. Der Modellierungsansatz, die vorgeschlagene Unternehmensarchitektur sowie das Vorgehensmodell von SOM bilden zusammen die SOM-Methodik. Die betriebliche Leistungserstellung ist wesentlicher Bestandteil der durch SOM unterstützten Modellbildung.

2.3.1.1 Systemtheoretische Charakterisierung von Unternehmen

Ein Unternehmen ist in der Außensicht ein offenes, zielorientiertes, sozioökonomisch-technisches System. Unternehmen sind als offene Systeme aufzufassen, da sie mit Kunden, Lieferanten und anderen Geschäftspartnern interagieren, um Güter und Dienstleistungen auszutauschen. Ein Unternehmen und seine Güter bzw. Dienstleistungen sind stets Teil eines Wertschöpfungsnetzwerkes, das mehrere unterschiedliche Unternehmen umfasst. Neben den Güter- und Dienstleistungsflüssen werden Informations- und Geldflüsse betrachtet. Das Verhalten eines Unternehmens wird von seinen Geschäftszielen entscheidend beeinflusst.

Ein Unternehmen ist ein sozioökonomisch-technisches System, da die Akteure in einem Unternehmen Menschen und Maschinen sind. Menschliche Akteure sind Personen, die verschiedene Rollen in einer Unternehmensorganisation einnehmen können. Unterschiedliche Produktionseinrichtungen wie Werke, Produktionsmaschinen, Transportsysteme und Computer fungieren als maschinelle Akteure. Anwendungssysteme (vergleiche Abschnitt 1.1.8 dieses Kurses für eine exakte Definition und Kurseinheit 5 für weitere Beispiele für Anwendungssysteme) dienen der Automatisierung des Informationssystems eines Unternehmens. Das unternehmensweite Informationssystem ist in diesem Sinne als informationsverarbeitendes Subsystem des Systems Unternehmen aufzufassen.

Verteiltes System

Ein Unternehmen kann gleichzeitig als verteiltes System aufgefasst werden, das aus autonomen, lose gekoppelten Komponenten besteht, die zur Erreichung ihrer Entwurfsziele kooperieren. Wir definieren den Begriff des verteilten Systems wie folgt [56].

Definition 2.3.1 (Verteiltes System) *Ein verteiltes System wird durch die folgenden Eigenschaften charakterisiert:*

- *Das verteilte System wirkt nach außen wie ein integriertes System. Insbesondere ist die Verfolgung gemeinsamer Ziele für das verteilte System typisch.*
- *Das verteilte System umfasst mehrere autonome Komponenten, die zur Erreichung der gemeinsamen Ziele des verteilten Systems miteinander kooperieren. Keine Komponente besitzt die globale Kontrolle über das Gesamtsystem.*

Koordination

Die autonomen Komponenten eines Unternehmens sind Geschäftsprozesse, die zur Herstellung von Gütern und Dienstleistungen dienen. Weitere Geschäftsprozesse werden durch die Güter und Dienstleistungen zur Verfügung gestellt. Jeder einzelne Geschäftsprozess besitzt spezifische Ziele, die aus dem Gesamtziel des Unternehmens durch die Unternehmensführung abgeleitet werden [25]. Diese Ableitung von Zielen für die einzelnen Geschäftsprozesse ist ein wesentliches Koordinationsinstrument zur Herstellung eines kooperativen Verhaltens. Der Koordinationsbegriff ist wie folgt definiert [38].

Definition 2.3.2 (Koordination) *Unter Koordination wird die direkte Einflussnahme auf unterschiedliche Systembestandteile mit dem Ziel eines integrierten, harmonischen Zusammenwirkens verstanden.*

Neben der Koordination über die Ziele der Geschäftsprozesse findet auch eine verhandlungsbasierte Abstimmung zwischen den unterschiedlichen Geschäftsprozessen statt.

Jeder Geschäftsprozess hat Bestandteile, die kooperieren und deshalb koordiniert werden müssen. Die erforderliche Koordination wird durch Anweisungen und Überwachung durch das für den jeweiligen Geschäftsprozess verantwortliche Management durchgeführt. Die unterschiedlichen Bestandteile eines Geschäftsprozesses sind im Gegensatz zu den einzelnen Geschäftsprozessen selber eng gekoppelt.

Funktionstypen

Die Geschäftsprozesse und ihre Unterprozesse implementieren Funktionen. Die folgenden Typen von Funktionen existieren:

- **Eingabe-Ausgabe-Funktionen**: Diese Art von Funktionen implementiert das eigentliche Produktionssystem.
- **Lieferfunktionen**: Diese Art von Funktionen liefert Materialien und Energie.
- **Wartungsfunktionen**: Die Sicherung des laufenden Betriebs wird durch Wartungsfunktionen vorgenommen.
- **Monitoringfunktionen**: Diese Funktionen sind für die Wahrnehmung von Störungen innerhalb und außerhalb des Unternehmens verantwortlich.
- **Koordinationsfunktionen**: Sie dienen der Koordination der unterschiedlichen Subsysteme und damit der Sicherstellung eines kooperativen Verhaltens.

Diese Unterscheidung von Funktionstypen soll bei der Identifizierung von Geschäftsprozessen und ihren Unterprozessen helfen.

2.3.1.2 SOM-Unternehmensarchitektur

Modellebenen für SOM

SOM [13, 53] verwendet drei unterschiedliche Modellebenen, die eine Unternehmensarchitektur bilden. Die drei Ebenen der Unternehmensarchitektur können wie folgt beschrieben werden:

1. Die oberste Ebene besteht aus einem Unternehmensplan zur Modellierung der Außensicht eines betrieblichen Systems.
2. Die mittlere Ebene wird durch ein Geschäftsprozessmodell zur Modellierung der Innensicht eines betrieblichen Systems gebildet.

3. Die unterste Ebene dient der Modellierung von Ressourcen, die zur Ausführung von Aktivitäten der Geschäftsprozesse benötigt werden.

Wir stellen diese drei Ebenen in Abbildung 2.2 dar.

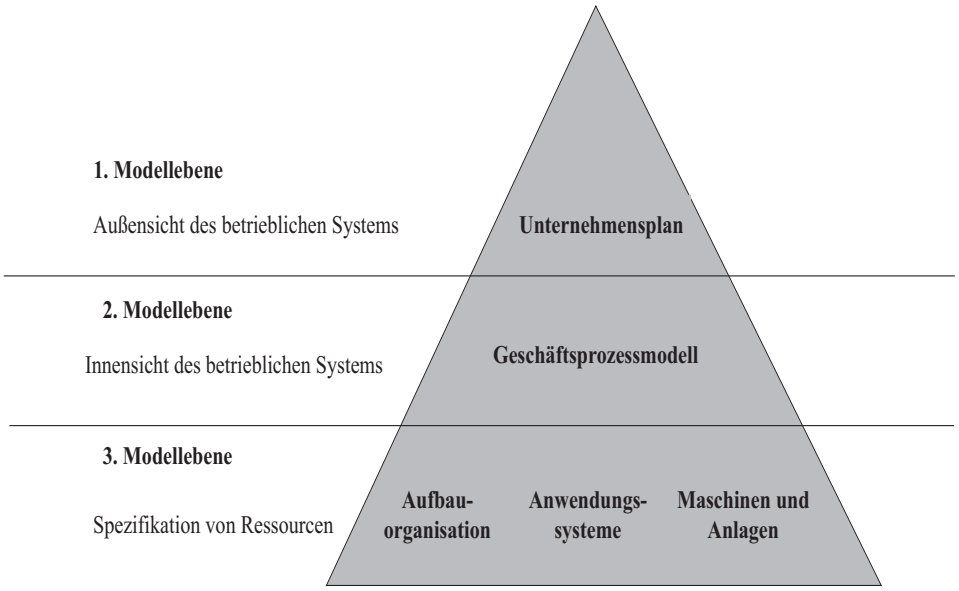

Abbildung 2.2: Unternehmensarchitektur nach SOM (in Anlehnung an [13])

Unternehmensplan

Der **Unternehmensplan** blickt von außen auf das Unternehmen. Der Plan beschäftigt sich mit den globalen Aufgaben und Ressourcen des Unternehmens. Globale Aufgaben beziehen sich auf die Festlegung von Zielen sowie auf Güter und Dienstleistungen, die hergestellt werden sollen. Die globalen Aufgaben dienen zur Festlegung der notwendigen Ressourcen. Umgekehrt sind die verfügbaren Ressourcen bei der Aufgabenfestlegung zu berücksichtigen.

Geschäftsprozessmodell

Das **Geschäftsprozessmodell** blickt von innen auf das System „Unternehmen“. Das Geschäftsprozessmodell spezifiziert Kern- und Hilfsprozesse. Kernprozesse dienen unmittelbar der Erfüllung der Unternehmensziele, während die Hilfsprozesse dies nur indirekt tun, indem sie ihre Ergebnisse Kernprozessen oder anderen Hilfsprozessen zur Verfügung stellen. Beziehungen zwischen unterschiedlichen Geschäftsprozessen sind nach dem Client-Server-Prinzip organisiert. Ein Client-Geschäftsprozess fordert andere Geschäftsprozesse auf, eine bestimmte Dienstleistung an ihn zu liefern. Die aufgeforderten Geschäftsprozesse fungieren somit als Server. Die Geschäftsprozesse eines Unternehmens bilden ein verteiltes System. Sie kooperieren, um die aus den Unternehmenszielen abgeleiteten Ziele für die einzelnen Geschäftsprozesse zu erreichen.

Spezifikation von Ressourcen

Die **Spezifikation von Ressourcen** bezieht sich auf Personal, Maschinen und Produktionsstätten sowie Anwendungssysteme. Wir betrachten in dieser Kurseinheit lediglich das informationsverarbeitende Subsystem eines Unternehmens. Jede Informationsverarbeitungsaufgabe kann in nicht weiter zerlegbare,

vollständig automatisierte oder vollständig nicht-automatisierte Unteraufgaben zerlegt werden. Wir bezeichnen derartige Unteraufgaben als atomare Aufgaben. Infolgedessen konzentrieren wir uns auf Anwendungssysteme und Personal als Ressourcen.

Im Sinne des in Abschnitt 2.2 beschriebenen generischen Architekturrahmens besteht die SOM-Architektur aus drei unterschiedlichen Modellebenen, die das zu modellierende Unternehmen aus unterschiedlichen Perspektiven betrachten. Die Beziehungen zwischen den Modellebenen werden beschrieben. Jede Modellebene stellt für sich ein verteiltes System dar, das aus autonomen, lose gekoppelten Komponenten besteht. Die vorgeschlagene Drei-Ebenen-Architektur erlaubt lokale Änderungen an den Modellebenen [13]. So ist es möglich, Änderungen am Geschäftsprozessmodell vorzunehmen, gleichzeitig aber die Spezifikation von Ressourcen und den Unternehmensplan nicht zu ändern.

Die Funktionalität und die Architektur von Anwendungssystemen auf der Ressourcenebene werden aus dem Geschäftsprozessmodell abgeleitet. Dabei wird die später in Abschnitt 5.3.7.4 näher erläuterte strukturelle Analogie zwischen Organisation und Informationssystemarchitektur ausgenutzt.

2.3.1.3 Geschäftsprozessmodellierung in SOM

SOM-Sprache

In diesem Abschnitt beschreiben wir die SOM-Sprache [12] zur Geschäftsprozessmodellierung. Die Sprache wird im Wesentlichen durch ein Metamodell sowie eine Menge von Dekompositionsregeln vollständig bestimmt.

Das Metamodell für Geschäftsprozesse beinhaltet Notationen und die Beziehungen zwischen diesen Notationen. Die Notationen und Beziehungen werden durch Entitäten und Relationen im Rahmen eines Entity-Relationship-Modells dargestellt. Das Geschäftsprozessmodell beschreibt eine Menge von Geschäftsprozessen und deren Beziehungen untereinander. Jeder Geschäftsprozess besitzt Ziele, deren Zielerreichung durch die Vertreter der verschiedenen Managementebenen eines Unternehmens kontrolliert wird.

Geschäftsobjekt

Das **Geschäftsprozessmodell** wird durch Geschäftsobjekte gebildet. Ein **Geschäftsobjekt** produziert Güter und Dienstleistungen und stellt diese anderen Geschäftsprozessen zur Verfügung. Jedem Geschäftsprozess ist mindestens ein Geschäftsobjekt zugeordnet. Jedes Geschäftsobjekt gehört entweder zu einem Geschäftsprozess oder zur Umwelt des betrachteten Unternehmens. Eine

Geschäftstransaktion

Geschäftstransaktion überträgt Güter, Dienstleistungen oder Nachrichten an einen anderen Geschäftsprozess oder erhält diese von Geschäftsprozessen. Geschäftstransaktionen, die mehrere Geschäftsprozesse miteinander verknüpfen, sind diesen Geschäftsprozessen zugeordnet.

Wir unterscheiden zwischen operativen Geschäftsobjekten und Geschäftsobjekten mit Managementfunktion. Ein operatives Geschäftsobjekt produziert oder transportiert Güter und Dienstleistungen unmittelbar. Geschäftsobjekte mit Managementfunktion hingegen steuern operative Geschäftsobjekte unter Verwendung von Nachrichten. Eine Geschäftstransaktion verbindet stets genau

zwei Geschäftsobjekte. Transaktionen (TA) können als Abbildungen zwischen Geschäftsobjekten aufgefasst werden. Es gilt:

$$TA(BO_1, BO_2) : X_{BO_1} \times Y_{BO_1} \times Z_{BO_1} \longrightarrow X_{BO_2} \times Y_{BO_2} \times Z_{BO_2}, \qquad (2.8)$$

wobei wir mit X_{BO_j} die Eingangsgrößen, mit Y_{BO_j} die Ausgangsgrößen und mit Z_{BO_j} den Zustand von Geschäftsobjekt BO_j bezeichnen.

Mehrere Geschäftstransaktionen können von einem Geschäftsobjekt initiiert werden. In diesem Fall bildet das jeweilige Geschäftsobjekt bei jeder Geschäftstransaktion die linke Seite in Beziehung (2.8). Umgekehrt kann ein Geschäftsobjekt direkt von Geschäftstransaktionen beeinflusst werden. Das entsprechende Geschäftsobjekt steht in diesem Fall auf der rechten Seite von Beziehung (2.8).

Für die Interaktionen zwischen Geschäftsobjekten auf der Basis von Geschäftstransaktionen existieren die nachfolgenden Möglichkeiten:

- Die Geschäftstransaktion überträgt Güter, Dienstleistungen oder Nachrichten zwischen zwei operativen Geschäftsobjekten.
- An der Interaktion sind genau zwei Geschäftsobjekte mit Managementfunktion beteiligt. Bei der Interaktion werden Nachrichten ausgetauscht.
- Außerdem können ein Geschäftsobjekt mit Managementfunktion und ein operatives Geschäftsobjekt durch Nachrichtenaustausch miteinander interagieren.

Geschäftsobjekte, die Geschäftstransaktionen zwischen ihnen, sowie die durch die Geschäftstransaktionen übertragenen Güter und Dienstleistungen werden als Interaktionsschema bezeichnet. Das Interaktionsschema stellt eine strukturelle Sicht dar.

Jedes Geschäftsobjekt beinhaltet eine oder mehrere Aufgaben. Jede einzelne Aufgabe löst eine oder mehrere Geschäftstransaktionen aus. Die einem Geschäftsobjekt zugeordneten Aufgaben arbeiten im objektorientierten Sinne (vergleiche hierzu Abschnitt 3.4.3) auf den dem Geschäftsobjekt zugeordneten Attributen. Die Aufgaben besitzen Ziele.

In SOM wird zwischen lose und eng gekoppelten Aufgaben unterschieden. Eng gekoppelte Aufgaben gehören zu einem Geschäftsobjekt und arbeiten folglich auf den durch die Attribute des Geschäftsobjekts dargestellten Daten. Lose gekoppelte Aufgaben gehören zu unterschiedlichen Geschäftsobjekten und arbeiten daher auf Attributen, die zu unterschiedlichen Geschäftsobjekten gehören. Die Aufgaben sind über Geschäftstransaktionen miteinander verbunden. Die Geschäftstransaktionen übertragen somit die Werte von Attributen des Geschäftsobjektes.

lose und enge Kopplung

Eng gekoppelte Aufgaben werden durch interne Ereignisse des Geschäftsobjektes ausgelöst. Interne Ereignisse hingegen werden durch eine Aufgabe initiiert

und stoßen die Ausführung einer anderen Aufgabe an. Wir betrachten dazu das nachfolgende Beispiel.

Beispiel 2.3.1 (Eng gekoppelte Aufgaben) *Wir betrachten ein Los als spezielles Geschäftsobjekt. Dieses Los muss verschiedene Arbeitsgänge durchlaufen, die zur Herstellung des mit dem Los assoziierten Produktes notwendig sind. Die Durchführung der einzelnen Arbeitsgänge sind Aufgaben des Geschäftsobjektes „Los“. Die Beendigung eines Arbeitsgangs stellt ein internes Ereignis dar, das die Durchführung des im Arbeitsplan nachfolgenden Arbeitsganges initiiert.*

Wir bemerken, dass im Gegensatz zu eng gekoppelten Aufgaben lose gekoppelte Aufgaben durch Geschäftstransaktionen angestoßen werden. Das nachfolgende Beispiel erläutert dies.

Beispiel 2.3.2 (Lose gekoppelte Aufgaben) *Wir betrachten das Geschäftsobjekt „Lager“ und das Geschäftsobjekt „Fertigungsauftrag“. Wenn im Lager der Mindestbestand für Schrauben durch Entnahmen unterschritten wird, wird eine Geschäftstransaktion ausgelöst, die einen Fertigungsauftrag initiiert. Dem Geschäftsobjekt „Fertigungsauftrag“ wird dazu sein Starttermin und die Menge an herzustellenden Schrauben mitgeteilt.*

Externe Ereignisse

Neben den bereits eingeführten internen Ereignissen existieren externe Ereignisse. Externe Ereignisse zeichnen sich dadurch aus, dass sie unabhängig von den Geschäftstransaktionen und Geschäftsobjekten ausgelöst werden. Wir betrachten dazu das nachfolgende Beispiel.

Beispiel 2.3.3 (Externes Ereignis) *Ein bezogen auf das System „Produzierendes Unternehmen“ externes Ereignis ist die Änderung der terminlichen Dringlichkeit eines Kundenauftrags durch den Kunden.*

Externe Ereignisse lösen ebenfalls Aufgaben aus. Die Abfolge von Aufgaben, Geschäftstransaktionen und Ereignissen, die neue Aufgaben auslösen, wird als Aufgaben-Ereignis-Schema eines Geschäftsprozessmodells bezeichnet. Das Aufgaben-Ereignis-Schema stellt somit eine Verhaltenssicht dar. Es beschreibt, welche Aufgabe infolge welcher Ereignisse bzw. Geschäftstransaktionen durchgeführt werden.

Der Zusammenhang zwischen Geschäftsobjekten, Geschäftstransaktionen, Aufgaben und Ereignissen ist als UML-Klassendiagramm in Abbildung 2.3 dargestellt. Wir verweisen auf Abschnitt 3.4.3 der nächsten Kurseinheit oder auf vorausgegangene Kurse für die Darstellung der UML-Notation.

2.3.1.4 Regeln zur Dekomposition von Geschäftsprozessmodellen

Die SOM-Methodik erlaubt die sukzessive Verfeinerung der Geschäftsprozessmodelle. Dabei werden ausgehend von Abbildung 2.3 das Interaktionsschema und

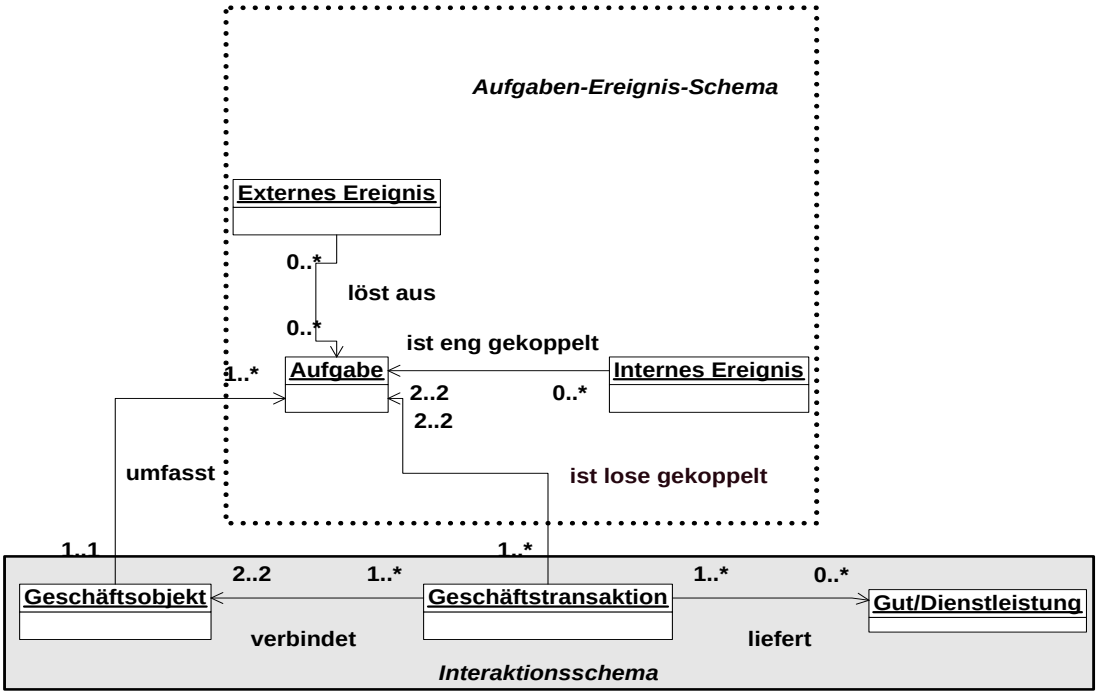

Abbildung 2.3: Metamodell für Geschäftsprozessmodelle [13]

das Aufgaben-Ereignis-Schema getrennt behandelt. Die Entitäten „Geschäftsobjekt“ und „Geschäftstransaktion“ des Interaktionsschemas werden unter Verwendung von Regeln geeignet dekomponiert. Für die Entität „Gut“ bzw. „Dienstleistung“ hingegen ist eine Betrachtung von entsprechenden Sub-Gütern und Sub-Dienstleistungen ausreichend. Das Aufgaben-Ereignis-Schema wird nicht dekomponiert, aber im Zuge der Betrachtung von Dekompositionslevels des Interaktionsschemas schrittweise feiner beschrieben. Wir zeigen, dass die Dekompositionsregeln zur Abbildung der beiden Koordinationsmechanismen „Vorgabe mit Rückkopplung“ und „Verhandlung“ dienen.

Dekomposition von Geschäftsobjekten

Wir geben zunächst Regeln zur Dekomposition von Geschäftsobjekten aus dem Interaktionsschema an:

$$BO ::= \{BO_1, BO_2, TA_c(BO_1, BO_2), [TA_f(BO_2, BO_1)]\}, \tag{2.9}$$

$$BO ::= \{BO_1, BO_2, [TA(BO_1, BO_2)]\}, \tag{2.10}$$

$$BO ::= \{spec\, BO_1\}^+, \tag{2.11}$$

$$BO_1|BO_2 ::= BO. \tag{2.12}$$

Dabei dient das Symbol ::= zur Kennzeichnung einer Ersetzung. Geschweifte Klammern {} werden für die Darstellung von Mengen verwendet. Listen von gleichartigen Elementen werden mit $\{\}^+$ bezeichnet. Optionale Bestandteile werden durch eckige Klammern [ ] beschrieben. Spezialisierungen werden durch das Symbol *spec* ausgedrückt. Alternativen durch einen senkrechten Strich |.

Regel (2.9) dient der Beschreibung des Rückkopplungsprinzips. Dazu wird das Geschäftsobjekt BO in zwei Unterobjekte BO_1 und BO_2 zerlegt. Das Geschäftsobjekt BO_1 stellt ein Managementgeschäftsobjekt dar, während BO_2 ein operatives Geschäftsobjekt bezeichnet. Mit TA_c bezeichnen wir eine Geschäftstransaktion („Control transaction"), die Anweisungen des Managementgeschäftsobjekts an das operative Geschäftsobjekt übergibt. Die Geschäftstransaktion TA_f („Feedback transaction") dient der Beschreibung der (optionalen) Rückkopplung.

Regel (2.10) besagt, dass Geschäftsobjekte in Untergeschäftsobjekte dekomponiert werden können und diese Untergeschäftsobjekte durch eine Geschäftstransaktion verbunden werden. Diese Tatsache wird auch durch Beziehung (2.8) ausgedrückt.

Die Regel (2.11) besagt, dass Geschäftsobjekte eines Typs in eine Menge von spezialisierten Geschäftsobjekten zerfallen. Wir betrachten dazu folgendes Beispiel.

Beispiel 2.3.4 (Spezialisierung von Geschäftsobjekten) *Aus dem Geschäftsobjekt „Auftrag" können die Geschäftsobjekte „Kundenauftrag", „Lagerauftrag" sowie „Fertigungsauftrag" gewonnen werden.*

Regel (2.12) macht deutlich, dass BO_1 und BO_2 Geschäftsobjekte sind. Diese Regel dient somit als Ersetzungsregel im Rahmen eines schrittweisen Dekompositionsprozesses.

Dekomposition von Geschäftstransaktionen

Anschließend beschreiben wir die Dekomposition von Geschäftstransaktionen durch die nachfolgenden Regeln:

$$TA(BO,BO_1) ::= [[TA_i(BO,BO_1)seq]TA_{con}(BO_1,BO)seq]TA_e(BO,BO_1), \quad (2.13)$$

$$TA_j ::= TA_j^{(1)}\{\ seq\ TA_j^{(2)}\}^+ \mid TA_j^{(1)}\{\ par\ TA_j^{(2)}\}^+, j \in \{i, con, e, c, f\}, \quad (2.14)$$

$$TA_j ::= \{\ spec\ TA_j^{(1)}\}^+, j \in \{i, con, e, c, f\}, \quad (2.15)$$

$$TA_i \mid TA_{con} \mid TA_e ::= TA, \quad (2.16)$$

$$TA_c \mid TA_f ::= TA. \quad (2.17)$$

Wir verwenden an dieser Stelle die Bezeichnungen „seq" und „par" für eine sequentielle bzw. parallele Abarbeitung von Transaktionen.

Dekompositionsregel (2.13) dient der Beschreibung des Verhandlungsprinzips. Eine Geschäftstransaktion wird dazu in drei Untergeschäftstransaktionen zerlegt. Die Geschäftstransaktion TA_i initiiert die Verhandlung. Informationen bezüglich zu liefernder Güter bzw. Dienstleistungen werden zwischen den durch die Transaktion gekoppelten Geschäftsobjekten ausgetauscht. In der eigentlichen Vertragsgeschäftstransaktion TA_{con} („Contracting Transaction") schließen die beiden Geschäftsobjekte einen Vertrag über die Bereitstellung bestimmter Güter bzw. Dienstleistungen. In der abschließenden Ausführungsgeschäftstransaktion TA_e („Enforcing Transaction") wird die Übertragung der Güter bzw. Dienstleistungen veranlasst. Durch Regel (2.13) wird ausgedrückt, dass die Unterge-

schäftstransaktionen TA_i, TA_{con} und TA_e unmittelbar hintereinander stattfinden. TA_i und TA_{con} sind dabei jeweils optional.

Regel (2.14) stellt sicher, dass eine Geschäftstransaktion in Untergeschäftstransaktionen eines bestimmten Typs zerlegt wird, die dann sequentiell oder parallel ausgeführt werden.

Regel (2.15) besagt in Analogie zu (2.11), dass Geschäftstransaktionen eines bestimmten Typs in Mengen spezialisierter Geschäftstransaktionen zerfallen. Durch Regel (2.15) wird somit eine Typisierung von Geschäftstransaktionen erreicht.

Regel (2.16) besagt, dass TA_i, TA_{con} und TA_e Geschäftstransaktionen sind. Entsprechend Regel (2.17) gilt das auch für TA_c und TA_f. Die Regeln (2.16) und (2.17) sind somit Ersetzungsregeln innerhalb eines schrittweisen Dekompositionsprozesses.

Sukzessive Dekomposition

Wir betrachten zur Veranschaulichung der Zerlegungsregeln das folgende Beispiel.

Beispiel 2.3.5 (Sukzessive Dekomposition) *Die schrittweise Dekomposition eines aus einem Geschäftsobjekt bestehenden Geschäftsprozessmodells ist in Abbildung 2.4 dargestellt. Rechtecke werden zur Darstellung von Aufgaben eines Geschäftsobjektes verwendet, Kreise zur Abbildung von dem Geschäftsobjekt zugeordneten internen Ereignissen. Zunächst wird ein Geschäftsobjekt BO in zwei Geschäftsobjekte BO_1 und BO_2 zerlegt, die durch eine Geschäftstransaktion TA miteinander in Beziehung stehen. Im nächsten Schritt wird TA weiter verfeinert. Die Geschäftstransaktion TA wird in die Untergeschäftstransaktionen TA_i, TA_{con} und TA_e zerlegt. TA_i initiiert die Verhandlung zwischen BO_1 und BO_2. Die Geschäftstransaktion TA_{con} dient dem Vertragsabschluss, während TA_e die eigentliche Übertragung der Güter und Dienstleistungen veranlasst. Die Untergeschäftstransaktionen führen zu einer Veränderung des Zustands von BO_1 und BO_2. Im nächsten Schritt wird zusätzlich BO_1 in die Untergeschäftsobjekte BO_{o1} und BO_{m1} zerlegt, um das Koordinationsprinzip „Vorgabe mit Rückkopplung“ anzuwenden. BO_{m1} überwacht die Ausführung des Vertrags durch BO_{o1}.*

Durch die Regeln (2.9) bis (2.17) wird das System „Unternehmen“ mit seinen Geschäftsprozessen in ein Steuerungssystem mit entsprechenden Steuerungsprozessen und ein operatives System mit entsprechenden operativen Prozessen aufgeteilt. Das operative System wird auch als Basissystem und die dazugehörigen Prozesse als Basisprozesse bezeichnet [39].

Übungsaufgabe 2.1 (Dekompositionsregeln von SOM) *Wenden Sie die Dekompositionsregeln (2.9) bis (2.17) auf das Beispiel 2.3.5 an. Dekomponieren Sie Geschäftsobjekte sowie Geschäftstransaktionen.*

2.3.1.5 Anwendung von SOM auf den Geschäftsprozess „Einkauf“

In diesem Abschnitt betrachten wir den Geschäftsprozess „Einkauf“ eines Industrieunternehmens. Das Interaktionsschema besteht aus den nachfolgenden drei

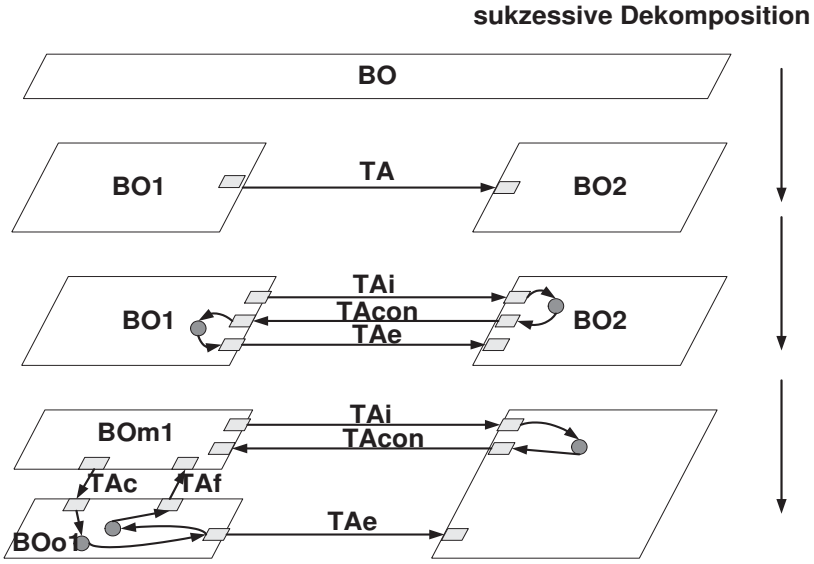

Abbildung 2.4: Dekomposition eines einfachen Geschäftsprozessmodells [13]

Geschäftsprozess „Einkauf“

Komponenten:

1. dem Geschäftsobjekt **„Einkäufer“**,
2. dem Geschäftsobjekt **„Lieferant“**,
3. der Geschäftstransaktion **„Einkaufen“**.

Das Geschäftsobjekt „Lieferant“ stellt ein externes Objekt dar, das zur Umgebung des Geschäftsprozesses „Einkauf“ gehört. Das Geschäftsobjekt „Einkäufer“ ist umgekehrt ein internes Objekt des Geschäftsprozesses. Die Geschäftstransaktion „Einkaufen“ ist für die Beschaffung von dinglichen Gütern verantwortlich, die für die Produktion des Unternehmens erforderlich sind.

In der ersten Iteration wird die Kooperation zwischen den Geschäftsobjekten „Einkäufer“ und „Lieferant“ durch die Geschäftstransaktion „Einkaufen“ beschrieben. Das Interaktionsschema und das Aufgaben-Ereignis-Schema für den Geschäftsprozess „Einkauf“ sind in Abbildung 2.5 gezeigt. Im Interaktionsschema werden Rechtecke und Ellipsen zur Darstellung von internen und externen Geschäftsobjekten verwendet. Die Bedeutung der Symbole im Aufgaben-Ereignis-Schema wurde bereits in Beispiel 2.3.5 beschrieben.

Das Aufgaben-Ereignis-Schema wird im rechten Teil von Abbildung 2.5 dargestellt. Das Schema zeigt die Ausführungsreihenfolge der Aufgaben, die auf dieser Modellierungsebene noch sehr einfach sind. Die Aufgabe „Einkaufen >“ wird vom Einkäufer erzeugt und vom Lieferanten als Aufgabe „> Einkaufen“ empfangen. Beide Aufgaben gehören zur Geschäftstransaktion „Einkaufen“.

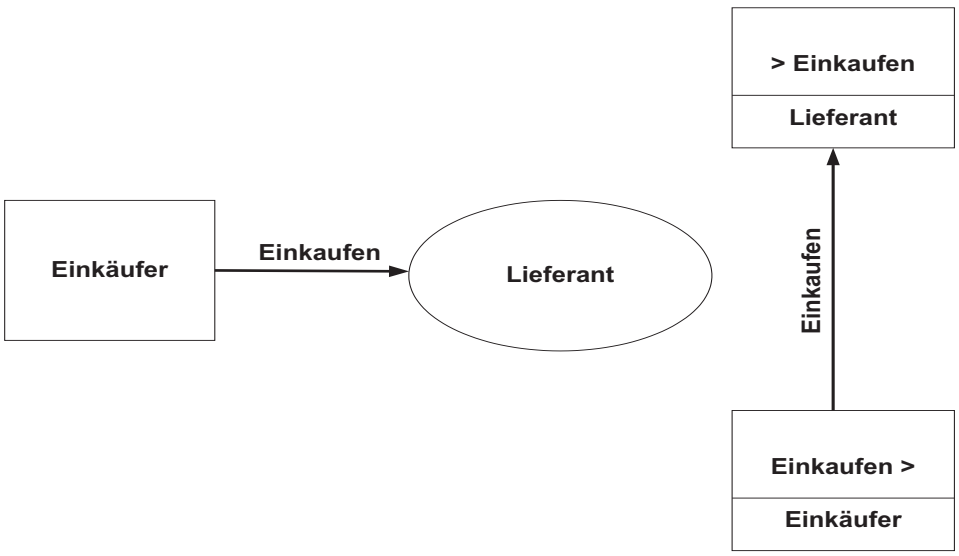

Abbildung 2.5: Geschäftsprozess „Einkauf“: Interaktions- und Aufgaben-Ereignis-Schema

In der zweiten Iteration verfeinern wir die Geschäftstransaktion „Einkaufen“. Die Geschäftstransaktion wird in drei Untertransaktionen entsprechend dem Verhandlungsprinzip zerlegt:

1. i: Einholen von Angeboten („initiating“),
2. con: Abgabe von Angeboten, Auswertung und Vertragsabschluss („contracting“),
3. e: Durchführung des eigentlichen Einkaufs („enforcing“).

Einkäufer und Lieferanten verhandeln miteinander. Das entsprechende Interaktions- sowie das Auftrags-Ereignis-Schema sind in Abbildung 2.6 angegeben.

Der Einkäufer fordert zunächst mehrere Lieferanten auf, Angebote abzugeben. Der Einkäufer wertet die Angebote aus und schließt mit dem am besten geeigneten Lieferanten einen Vertrag ab. Anschließend wird der eigentliche Einkauf auf Basis des Vertrags abgewickelt.

Im nächsten Schritt wird das Rückkopplungsprinzip auf das Geschäftsobjekt „Einkäufer“ angewandt. Dazu werden die Untergeschäftsobjekte „Einkauf“ und „Einkaufssystem“ eingeführt. Das Geschäftsobjekt „Einkauf“ stellt dabei das Managementobjekt dar, während das Untergeschäftsobjekt „Einkaufssystem“ als operatives Geschäftsobjekt wirkt. Für das Rückkopplungsprinzip werden weiterhin die Geschäftstransaktion „Einkaufsauftrag“ für Vorgaben, mit c: Einkaufsauftrag bezeichnet, und die Geschäftstransaktion „Einkaufsbericht“, f: Einkaufsbericht genannt, für Rückkopplungen verwendet. Die Geschäftstransaktionen des

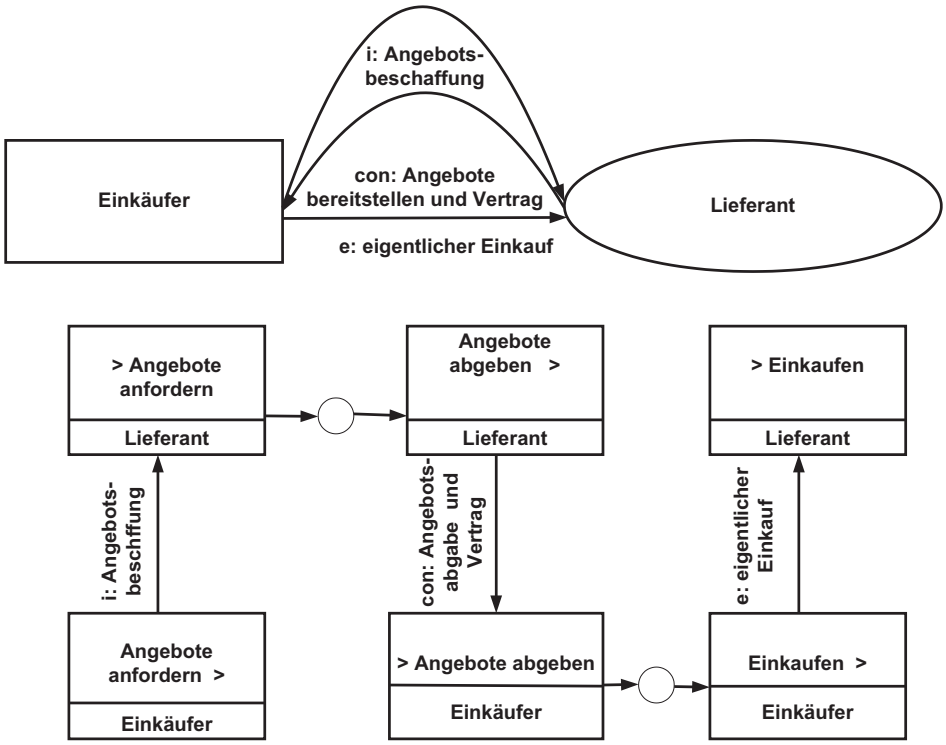

Abbildung 2.6: Geschäftsprozess „Einkauf“: Interaktions- und Aufgaben-Ereignis-Schema

Geschäftsobjekts „Einkäufer“ werden weiterhin den neuen Untergeschäftsobjekten zugeordnet. Die beschriebene Situation ist in Abbildung 2.7 in Form des Interaktionsschema dargestellt. Das zugehörige Aufgaben-Ereignis-Schema befindet sich in Abbildung 2.8.

Wir bemerken, dass weitere Verfeinerungen des in der dritten Iteration erhaltenen Interaktions- und Aufgaben-Ereignis-Schemas möglich sind. Beispielsweise kann das Einkaufssystem in weitere Subsysteme zerlegt werden.

Übungsaufgabe 2.2 (Geschäftsprozessmodellierung in SOM) *Betrachten Sie den Geschäftsprozess Vertrieb und modellieren Sie diesen mit SOM. Identifizieren Sie hierbei zuerst die Geschäftsobjekte und -transaktion und erstellen Sie das Interaktions- und das Aufgaben-Ereignis-Schema für den Geschäftsprozess Vertrieb. Zerlegen Sie im Anschluss daran die im ersten Schritt identifizierte Geschäftstransaktion in Untertransaktionen. Erstellen Sie dafür ebenfalls das Interaktions- und Aufgaben-Ereignis-Schema.*

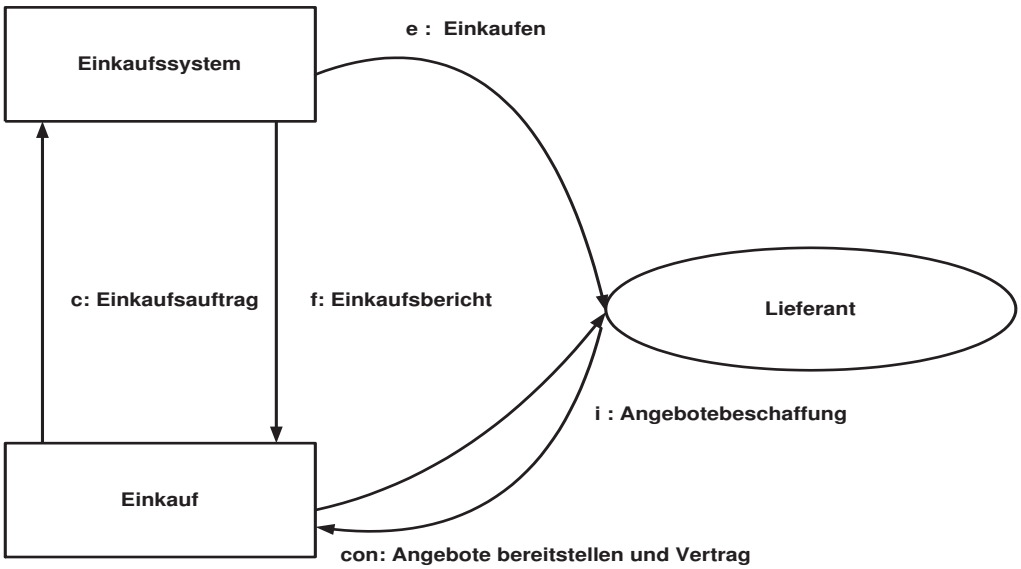

Abbildung 2.7: Geschäftsprozess „Einkauf“: Interaktionsschema

2.3.1.6 Zusammenhang von SOM und der Architektur von Anwendungssystemen

Objektorientierter Ansatz

Die SOM-Methodik schlägt einen objektorientierten Ansatz für die domänenspezifische Spezifikation von Anwendungssystemen vor [13]. Dazu wird zunächst ein Metamodell für Anwendungssysteme verwendet (siehe Abbildung 2.9), das dann in Beziehung zum Metamodell für Geschäftsprozesse aus Abbildung 2.3 gesetzt wird.

Im Mittelpunkt des Metamodells für Anwendungssysteme stehen ein Schema für konzeptuelle Klassen und ein Schema für Aufgabenklassen. Objekt-, dienst- sowie transaktionsspezifische Klassen sowie ihre Beziehungen bilden das Schema konzeptueller Klassen. Geschäftsobjekte aus dem Metamodell für Geschäftsprozessmodelle sind den objektspezifischen Klassen des Metamodells für Anwendungssysteme zugeordnet. Güter- und Dienstleistungen stehen in Beziehung zu den dienstspezifischen Klassen des Metamodells für Anwendungssysteme. Die Geschäftstransaktionen werden durch transaktionsspezifische Klassen des Metamodells für Anwendungssysteme abgebildet. Der in Abbildung 2.9 dargestellte Zusammenhang zwischen Geschäftsprozessmodell und Anwendungssystem hat Einfluss auf die Architektur des Anwendungssystems.

Wir erläutern im Folgenden diese Aussage genauer. Die SOM-Methode führt zu

- objektorientierten,

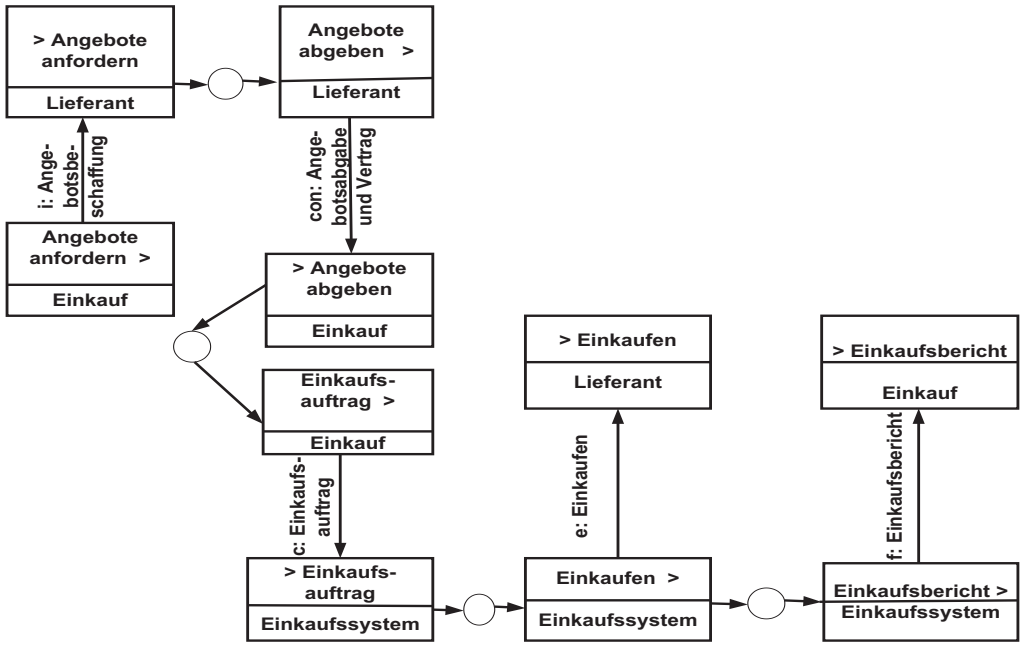

Abbildung 2.8: Geschäftsprozess „Einkauf“: Aufgaben-Ereignis-Schema

- verteilten,
- objekt-integrierten,
- adaptiven

Spezifikationen von betrieblichen Anwendungssystemen. Die Spezifikation ist **objektorientiert**, da das Schema der konzeptuellen Klassen und das Schema der Aufgabenklassen objektorientiert sind. Konzeptuelle Klassen kapseln den Zustand der Aufgaben der Geschäftsobjekte, der Geschäftstransaktionen sowie der Güter und Dienstleistungen. Außerdem werden Methoden definiert, die auf den die Zustände beschreibenden Attributen arbeiten. Ausgehend vom Interaktionsschema und dem Aufgaben-Ereignis-Schema des Geschäftsprozessmodells kann unter Verwendung der in Abbildung 2.9 dargestellten Beziehungen das Schema der konzeptuellen Klassen abgeleitet werden.

Workflow

Klassen des Aufgabenklassenschemas koordinieren das Zusammenwirken der Klassen des konzeptuellen Klassenschemas und des Aufgabenklassenschemas. Die Klassen des Aufgabenklassenschemas bestimmen somit den Workflow des Anwendungssystems. Wir definieren den Begriff Workflow wie folgt.

Definition 2.3.3 (Workflow) *Unter einem Workflow verstehen wir einen Teil eines Geschäftsprozesses, der aus sequentiell oder parallel ablaufenden Aktivitäten besteht [28].*

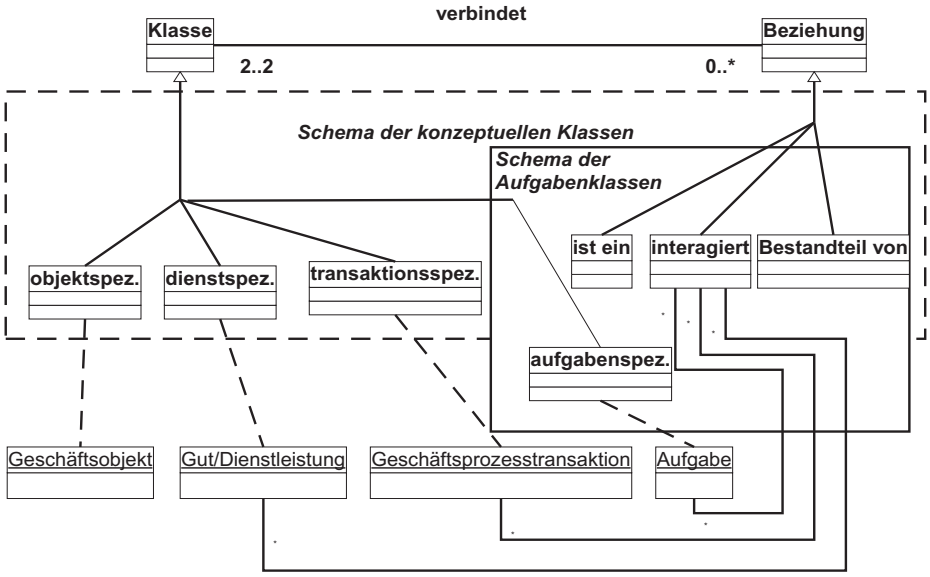

Abbildung 2.9: Metamodell für Anwendungssysteme [13]

Der Workflow wird alternativ auch als Ablaufsteuerung bezeichnet. Die Aktivitäten des Geschäftsprozesses sind durch Workflows auf Funktionen des Anwendungssystems abgebildet. Das führt zum Begriff des Workflow-Management-Systems. WFMS

Definition 2.3.4 (Workflow-Management-System) *Unter Workflow-Management-Systemen verstehen wir Systeme, die auf Basis von Workflow-Modellen die sowohl funktional als auch zeitlich korrekte Ablauffolge von Aktivitäten steuern und überwachen [59]. Die Aktivitäten sind dabei als Funktionen von Informationssystemen implementiert, deren Ablaufsteuerung anwendungsübergreifend ist und aus diesem Grund aus den einzelnen Anwendungssystemen herausgebrochen werden kann.*

Workflow-Management-Systeme (WFMS) dienen der Anpassung von Anwendungssystemen an die Ablauforganisation eines Unternehmens. Sie dienen somit dazu, Geschäftsprozesse zu automatisieren und sind das technische Gegenstück zu den stärker betriebswirtschaftlich orientierten Geschäftsprozessen.

Das Schema der Aufgabenklassen wird aus dem Aufgaben-Ereignis-Schema des zugehörigen Geschäftsprozessmodells abgeleitet.

Eine der SOM-Methodik folgende Spezifikation führt in natürlicher Weise zu einem **verteilten Anwendungssystem**. Ausgangspunkt der Spezifikation ist ein Geschäftsprozessmodell, das aus Geschäftsobjekten besteht, die durch Geschäftstransaktionen lose miteinander gekoppelt sind. In einem ersten Schritt wird jede konzeptuelle Klasse und Aufgabenklasse als autonome Einheit aufgefasst. In späteren Stadien des Spezifikationsprozesses können aus diesen autono-

men Einheiten größere Blöcke gebildet werden.

Die SOM-Methodik erweitert die Datenintegration (vergleiche hierzu die Ausführungen in Kurseinheit 1 und in [37]) um den Aspekt der **Objekt-Integration**. Anwendungssysteme, die entsprechend der SOM-Methodik entwickelt wurden, bestehen aus autonomen bzw. lose gekoppelten Subsystemen. Die Subsysteme entstehen dabei wie bereits beschrieben durch Verschmelzung autonomer Systemeinheiten. Die unterschiedlichen Subsysteme tauschen Nachrichten aus, um eine Konsistenz des gesamten Anwendungssystems zu erreichen. Die im Geschäftsprozessmodell beschriebenen Koordinationsmechanismen „Vorgabe mit Rückkopplung“ und „Verhandlungen“ können für die Spezifikation von Nachrichtenprotokollen verwendet werden.

Adaptivität

Die unter Verwendung der SOM-Methodik erhaltene Spezifikation ist **adaptiv**. Entscheidend dafür ist die Isomorphie zwischen Geschäftsprozessmodell und Anwendungssystem, die durch das Metamodell für Anwendungssysteme sichergestellt wird. Daraus folgt insbesondere, dass lokale Änderungen am Geschäftsprozess stets auch nur lokale Änderungen am auf dem Geschäftsprozessmodell basierenden Anwendungssystem nach sich ziehen. Diese Eigenschaft wird zusätzlich durch den verteilten Charakter des Anwendungssystems verstärkt.

2.3.2 Architektur integrierter Informationssysteme

2.3.2.1 Begriffsbildungen

ARIS

Die Architektur integrierter Informationssysteme (ARIS) [46, 47, 48] stellt ein Architekturkonzept für Informationssysteme da. Auf Basis von ARIS wurden Referenzmodelle für die wesentlichen Geschäftsprozesse in Industrieunternehmen erstellt.

Ebenen

ARIS schlägt entsprechend des generischen Architekturrahmens die drei Modellebenen

1. Fachkonzept,
2. DV-Konzept,
3. Implementierung

Sichten

vor. Für jede dieser drei Modellebenen wird eine Unterteilung in

1. eine Datensicht,
2. eine Funktionssicht,
3. eine Organisationssicht

vorgenommen. Zusätzlich wird als vierte Sicht eine Steuerungssicht vorgeschlagen, die zur Beschreibung des Ablaufs von Geschäftsprozessen dient. Eine fünfte Sicht stellt die Leistungssicht, die zur Beschreibung betrieblicher Leistungen

dient, dar. ARIS verwendet somit zur Untergliederung der Architektur im Wesentlichen Sichten. Auf allen Modellebenen werden mit Ausnahme der Leistungssicht die gleichen Sichten gebildet. Die im generischen Architekturrahmen vorgeschlagene Möglichkeit, auf den verschiedenen Ebenen unterschiedliche Sichten zu verwenden, wird somit nicht ausgenutzt. Die Daten-, Funktions- und Steuerungssicht sind dem Aufgabenblickwinkel zugehörig, während die Organisationssicht dem Aufgabenträgerblickwinkel zuzuordnen ist. Die Leistungssicht vereint Elemente des Aufgaben- und des Aufgabenträgerblickwinkels.

Datensicht

Definition 2.3.5 *Die Datensicht stellt die Datenverarbeitungsumgebung sowie Nachrichten, die Funktionen auslösen oder durch Funktionen ausgelöst werden, zur Verfügung.*

Grobe Informationen über die Funktion von Informationssystemen als Datenspeichermedium können durch Datenbezeichner ausgedrückt werden. Die Nachrichten repräsentieren Ereignisse wie „Kundenauftrag eingetroffen“ oder „Arbeitsgang abgeschlossen“. Diese Ereignisse sind Informationsobjekte, die durch Daten persistent gemacht werden. Der Zustand bestimmter betriebswirtschaftlicher Objekte wie „Kundenauftragsstatus“ oder „Kundenstatus“ wird ebenfalls durch entsprechende Daten dargestellt. Die Datensicht wird somit durch Ereignisse und Zustände gebildet [46].

Funktionssicht

Definition 2.3.6 *Die Funktionssicht beschreibt Vorgänge, die Eingangsgrößen in Ausgangsgrößen umwandeln. Funktionen werden zur Erreichung von Zielen entwickelt und ausgeführt. Außerdem wird die Erfüllung von Funktionen an ihren Zielen gemessen. Aus diesem Grund sind die Ziele auch ein Bestandteil der Funktionssicht [46].*

Anwendungssysteme automatisieren die Ausführung betrieblicher Funktionen. Aus diesem Grund ist die Anwendungssoftware auch der Funktionssicht zugeordnet.

Organisationssicht

Definition 2.3.7 *Die Organisationssicht dient der Abbildung der hierarchischen Unternehmensstruktur. Diese Sicht wird zur Festlegung von Verantwortlichkeiten und Stellen verwendet. Die Organisationssicht betrachtet die Elemente der Aufbauorganisation eines Unternehmens.*

Die verantwortlichen Entitäten „Menschen“, „Maschinen“, „Finanzen“ und „Computerhardware“ sind der Organisationssicht zugeordnet.

Steuerungssicht

Definition 2.3.8 *Die Steuerungssicht dient zur Beschreibung von Beziehungen zwischen den unterschiedlichen Sichten und zwischen den Objekten aller Geschäftsprozesse. Dadurch wird eine systematische Untersuchung der relevanten Beziehungen ermöglicht.*

Leistungssicht

Definition 2.3.9 *Die Leistungssicht wird zur Beschreibung betrieblicher Leistungen (Produkte) verwendet.*

Leistungen sind das Ergebnis von Prozessen. Umgekehrt veranlasst der Bedarf nach Leistungen die Prozessausführung [47]. Wir unterscheiden zwischen Sach- und Dienstleistungen. Informationsdienstleistungen sind spezielle Dienstleistungen. Zur Darstellung produktbezogener betrieblicher Leistungen werden beispielsweise Produktdatenmodelle benutzt [36, 37].

In Abbildung 2.10 wird die ARIS-Architektur veranschaulicht.

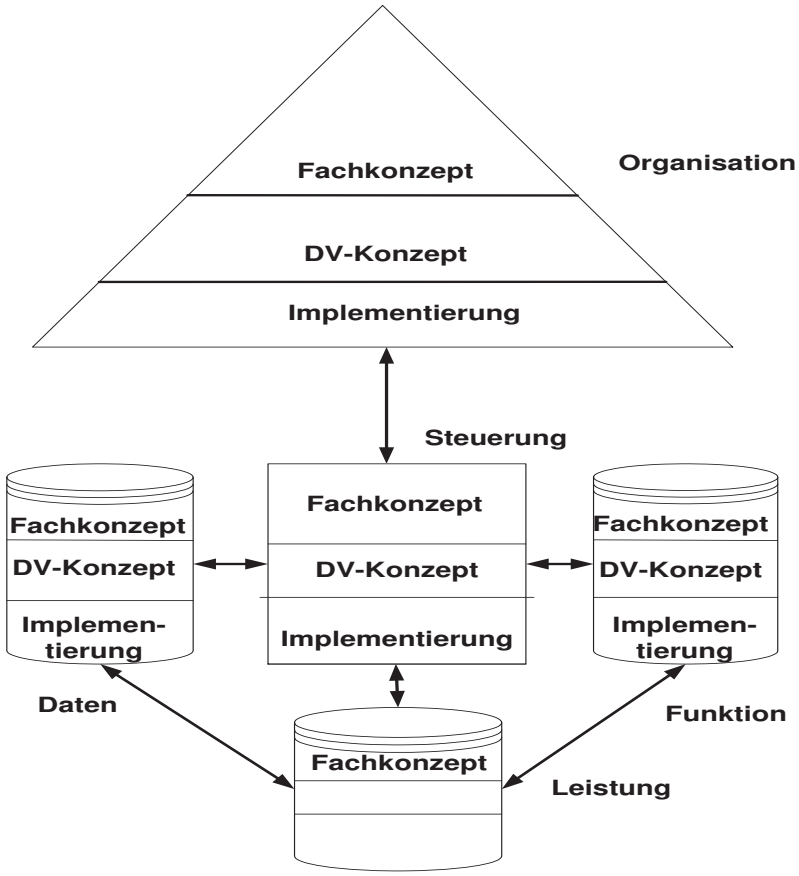

Abbildung 2.10: ARIS - Konzept [46]

Zur Komplexitätsreduktion werden innerhalb von ARIS Modellebenen eingeführt. Die Modellebenen werden als Beschreibungsebenen bezeichnet. Ausgangspunkt ist eine betriebswirtschaftliche Problemstellung.

- Auf der Fachkonzeptebene wird die Problemstellung exakt formuliert und unter Verwendung einer formalen Beschreibungssprache dargestellt. Das Fachkonzept beschäftigt sich ausschließlich mit der betriebswirtschaftlichen Problemstellung, Informationssysteme werden an dieser Stelle nicht in die Betrachtungen einbezogen.
- Die Ebene des DV-Konzepts dient dazu, die Begriffe des Fachkonzepts in entsprechende Konstrukte der Datenverarbeitung zu transformieren.

- Die Implementierungsebene enthält schließlich die zum DV-Konzept zugehörigen Hard- und Softwarekomponenten.

Wir konkretisieren im nächsten Schritt die Ausgestaltung der Modellebenen für die verschiedenen Sichten.

2.3.2.2 Fachkonzept

Funktionssicht Fachkonzept

Das Fachkonzept für die Funktionssicht umfasst

- Funktionsstruktur,
- Ablauffolge,
- Bearbeitungsformen.

Funktionen werden in Teilfunktionen zerlegt. Dadurch wird eine Komplexitätsreduktion erreicht. Als Ergebnis entstehen Elementarfunktionen, deren weitere Zerlegung betriebswirtschaftlich nicht mehr sinnvoll ist. Zur Modellierung von Funktionsstrukturen werden Funktionsbäume eingesetzt. Die so erhaltene Funktionsstruktur ist aber noch statisch. Das bedeutet, dass der zeitliche Ablauf, in der die identifizierten Teilfunktionen ausgeführt werden, nicht erkennbar ist.

Zur Beschreibung des zeitliches Ablaufes werden deshalb Anordnungsbeziehungen eingeführt. Zur Beschreibung der Ablaufsicht eignen sich unter anderem Interaktionsdiagramme der Unified-Modeling-Language (UML). Derartige Diagramme werden in Abschnitt 3.4.3.2 behandelt.

Im Fachkonzept für die Funktionssicht ist außerdem festzulegen, wie der Benutzer auf den Ablauf der Funktionen einwirken kann. Die automatische Verarbeitung ohne Eingriffe des Benutzers sowie die interaktive Verarbeitung werden unterschieden [46].

Datensicht Fachkonzept

Funktionen leben auf Daten. Die Funktionen müssen deshalb mit den passenden Daten, die in Datenstrukturen abgelegt sind, versorgt werden. Datenstrukturen können häufig nur mit erheblichem Aufwand geändert werden. Aus diesem Grund hat das Fachkonzept der Daten für die Gestaltung von Informationssystemen eine große Bedeutung. Wie aus dem Kurs über die Modellierung von Informationssystemen bekannt ist, sind für die Datenmodellierung Begriffe wie Entitätstypen, Beziehungstypen, Entitäten, Beziehungen sowie Attribute und Domänen wichtig. Ein erweitertes Entity-Relationship-Modell (ERM) wird als Modellierungskonzept für die Datensicht verwendet. Die Erweiterungen beziehen sich auf Möglichkeiten zur Abbildung von Generalisierung, Aggregation, Möglichkeiten zur Ereignis- und Zeitdarstellung, Abbildung von Existenzabhängigkeiten sowie Erweiterungen von Kardinalitäten.

Organisationssicht Fachkonzept

Fragen der Modellierung der Ablauforganisation wurden bereits im Rahmen von Ablauffolgen der Funktionssicht behandelt. In der Organisationssicht des

Fachkonzeptes wird deshalb nur die Aufbauorganisation betrachtet. Zur Darstellung dieser statischen Sicht auf die Organisation werden Organigramme verwendet. Ein Organigramm beschreibt die Organisationseinheiten und ihre Verknüpfungen. Die Weisungsbefugnis dient als Verknüpfungsart zwischen Organisationseinheiten. Durch sie werden die Berichtswege innerhalb der Organisation festgelegt.

Steuerungssicht Fachkonzept

Die Steuerungssicht verbindet Funktionen, Daten und Organisation [46]. Dazu ist es notwendig, die Modelle der Funktions-, Daten- und Organisationssicht miteinander zu verknüpfen. Das wird durch Ereignisgesteuerte Prozessketten (EPKs) [46, 45] geleistet. Eine Einführung in die EPK-Thematik wird in Abschnitt 3.4.2 dieses Kurses gegeben.

Leistungssicht Fachkonzept

Produktbäume bzw. -netze dienen der Modellierung der Leistungssicht. Sie bilden die Produktstruktur ab. In einem Produktstrukturmodell wird gegenüber einem Produktmodell eine reduzierte Semantik verwendet. Produktmodelle sind Datenmodelle. Leistungen werden mit den bei ihrer Herstellung anfallenden Kosten bewertet.

Übungsaufgabe 2.3 (Modellierung auf Fachkonzeptebene) *Das Fachkonzept der Datensicht beinhaltet die Datenmodellierung. Hierfür werden als Modellierungskonzept häufig Entity-Relationship-Modelle (ERM) eingesetzt. Erstellen Sie für die Produktionsdomäne ein ERM und verwenden Sie hierfür folgende Entitäten: Auftrag, Los, Produkt, Arbeitsplan, Arbeitsschritt, Maschine und Technologie und setzen Sie diese wie folgt in Beziehung. Ein Los kann genau einem Auftrag zugeordnet werden. Ein Auftrag wiederum gehört zu genau einem Produkt. Somit gehört ein Los auch genau zu einem Produkt. Ein Produkt gehört weiterhin zu einer Technologie und hat einen Arbeitsplan. Dieser Arbeitsplan besteht aus mehreren Arbeitsschritten, die wiederum unterschiedlichen Arbeitsplänen zugeordnet werden können. Um die Reihenfolge der Arbeitsschritte modellieren zu können, muss entsprechend eine Arbeitsschrittnummer mit abgelegt werden. Ein Arbeitsschritt kann auf unterschiedlichen Maschinen und auf einer Maschine können unterschiedliche Arbeitsschritte ausgeführt werden. Erstellen Sie ein ERM entsprechend der Beschreibung mit Entitäten, den wichtigsten Attributen der Entitäten und setzen Sie die Kardinalitäten. Kennzeichnen Sie die Primärschlüssel.*

2.3.2.3 DV-Konzept

Funktionssicht DV-Konzept

Das DV-Konzept in der Funktionssicht dient dem Modul- und Transaktionsentwurf. Alternativ wird der Name Softwareentwurf verwendet [47]. Zunächst werden die Funktionen gegenüber dem Fachkonzept feiner detailliert. Die Funktionen werden anschließend in Modulen (bzw. Komponenten) zusammengefasst. Vor der Einführung des im ARIS-DV-Konzept verwendeten Modulbegriffs führen wir zunächst den Begriff der Benutzertransaktion und des Dialogschrittes ein.

Definition 2.3.10 (Benutzertransaktion und Dialogschritt) *Unter einer Benutzertransaktion verstehen wir eine zusammenhängende Folge von Dialogschritten zur Bearbeitung einer Aufgabe. Das Ausführen eines Dialogschrittes bedeutet für den Benutzer die Bearbeitung einer Bildschirmmaske.*

Benutzertransaktionen

Obwohl fachlich motiviert, hat die Unterteilung in Benutzertransaktionen große Auswirkungen auf das DV-Konzept, da das Erreichen von Datenpersistenz eng mit den Benutzertransaktionen zusammenhängt. Wenn die Benutzertransaktion fehlerhaft ausgeführt wird, kann ähnlich wie bei Datenbanktransaktionen [9, 1] verlangt werden, dass die Daten auf den Stand vor der Benutzertransaktion zurückgesetzt werden.

Typischerweise sind mehrere Dialogschritte notwendig, um ein bestimmtes betriebswirtschaftliches Problem zu lösen. Die Bearbeitung einer Bildschirmmaske besteht für den Benutzer aus den Aktivitäten

- Empfangen der Maske,
- Dateneingabe,
- Absenden.

Daraus folgt, dass sich ein Dialogschritt im Allgemeinen in mehrere Systemschritte unterteilt.

Beispiel 2.3.6 (Systemschritt) *Ein möglicher Systemschritt ist durch die Eingabe von Daten in einer Maske gegeben. Weitere Systemschritte sind für das Umwandeln der eingegebenen Daten in ein bestimmtes Format und für das Speichern der Daten erforderlich. Speziell für das dauerhafte Speichern der Daten können auch Datenbanktransaktionen herangezogen werden.*

Aus Beispiel 2.3.6 ist deutlich zu erkennen, dass die eigentliche Verarbeitungslogik in den Systemschritten verankert ist, während Dialogschritte und die daraus zusammengesetzten Benutzertransaktionen Steuerungsfunktionalität bereitstellen.

Modul

Die Begriffe Modul und Benutzertransaktion werden in Abhängigkeit von der jeweiligen betriebswirtschaftlichen Standardsoftware unterschiedlich verwendet. Die in der Beschreibung des ARIS-DV-Konzepts verwendete Begriffsbildung lehnt sich stark an die des SAP R/3-Systems [6] an. Um trotzdem von konkreten Softwareprodukten relativ unabhängig zu sein und somit ein generisches Konzept vorzuschlagen, wird dem DV-Konzept von ARIS der folgende, relativ weitgefasste Modulbegriff zugrundegelegt [46, 47].

Definition 2.3.11 (Modul) *Eine Benutzertransaktion, ein Programm, Programmteil oder Bearbeitungsschritt heißt Modul, wenn das jeweilige Artefakt über die Moduleigenschaft verfügt, d.h. einen wohldefinierten Ein- und Ausgang für die Bearbeitung zur Verfügung stellt.*

Eine einzelne Funktion des Fachkonzepts wird typischerweise in mehrere Module umgesetzt. Umgekehrt können mehrere Funktionen zu einem Modul zusammengefasst werden.

Algorithmen werden innerhalb der Funktionssicht des DV-Konzepts mit Hilfe von Struktogrammen [46] veranschaulicht, der Kontrollfluss wird mit verschiedenen UML-Diagrammen dargestellt.

Datensicht DV-Konzept

Die Datensicht des DV-Konzepts dient dazu, die Datenhaltung zu beschreiben. Dazu wird aufbauend auf dem logischen Datenmodell in Form eines ERM ein Datenmodell des DV-Konzepts erzeugt. In Vorlesungen zu Datenbanken werden Verfahren behandelt, mit denen man ein ERM in ein entsprechendes Relationenmodell umwandeln kann (vergl. z.B. [50, 33]). Dieses Relationenmodell kann dann, nach entsprechender Normalisierung, dazu verwendet werden, die Tabellenstruktur für das jeweilige relationale DBMS abzuleiten.

Organisationssicht DV-Konzept

Im Fachkonzept dient die Organisationssicht der Darstellung der Aufbauorganisation. Daraus folgt, dass im DV-Konzept ein dazugehöriges Kommunikations- und Informationssystem dargestellt werden muss, das die unterschiedlichen Organisationseinheiten miteinander vernetzt. Dazu dient im Rahmen des DV-Konzepts aus Organisationssicht die Netztopologie. Den einzelnen Organisationseinheiten des Organisationsmodells aus dem Fachkonzept werden Rechnerknoten zugeteilt. Die dezentralen Organisationseinheiten bauen eigenständige Informationssysteme auf, die für ein Zusammenwirken im Sinne des Gesamtunternehmens koordiniert werden. Für die Beschreibung der Organisationssicht werden in der ARIS-Beschreibung keine speziellen Darstellungsformen vorgeschlagen.

Steuerungssicht DV-Konzept

Das DV-Konzept unter Steuerungsgesichtspunkten wird ähnlich wie bereits beim Fachkonzept dazu verwendet, um Funktionen, Daten und Organisation unter IT-Gesichtspunkten miteinander zu verbinden. Wir stellen zunächst die Verbindung von Funktionen und Organisation dar. Die durch das DV-Konzept in der Funktionssicht bereitgestellten Module werden auf die einzelnen Knoten des Rechnernetzes aufgeteilt.

Bei der Darstellung der Verbindung von Daten und Organisation wird beschrieben, wie Daten auf die Knoten eines Rechnernetzes verteilt werden können. Außerdem werden Zugriffsberechtigungen für die einzelnen Organisationseinheiten auf die verteilten Daten organisiert.

Der Zusammenhang von Funktionen und Daten wird durch die Beschreibung des Zugriffs auf die Daten aus den Funktionen heraus angegeben. Unter Verwendung des in der Datensicht des DV-Konzepts entwickelten Relationenschemas werden die Daten in relationalen Datenbanken abgelegt. Auf die Datenbank wird unter Verwendung von Datenbanktransaktionen zugegriffen. Wie bereits beschrieben, kann eine Benutzertransaktion des DV-Konzepts in funktionaler Sicht Datenbanktransaktionen enthalten.

Nachdem bisher für die Beschreibung der Steuerungssicht die Paare Funktionen-Organisation, Daten-Organisation sowie Funktionen-Daten darge-

stellt worden sind, beschreiben wir nun das Zusammenwirken von Funktionen, Organisation und Daten. Die im Rahmen der Steuerungssicht des Fachkonzepts entwickelten Prozessketten sind so zu steuern, dass die entsprechenden Teilvorgänge zum richtigen Zeitpunkt von den richtigen Akteuren, repräsentiert durch Organisationseinheiten, angestoßen werden. Wir unterscheiden zwischen programm- und dialoggesteuerter Bearbeitung. Aktionsnachrichten dienen der Übermittlung von Informationen an menschliche Aufgabenträger und lösen die Bearbeitung von Teilvorgängen aus. Triggernachrichten hingegen übermitteln Informationen an Anwendungsprogramme als maschinelle Aufgabenträger. Der Begriff „Trigger“ ist in diesem Zusammenhang wie folgt definiert: Trigger

Definition 2.3.12 (Trigger) *Ein Trigger ist ein Paar*

$$T := (E, A), \tag{2.18}$$

wobei Aktion A unmittelbar nach dem Auftreten von Ereignis E ausgeführt wird. Eine Aktion bedeutet dabei die Ausführung von bestimmten Funktionen.

Der eingeführte Triggerbegriff ermöglicht es, die durch EPKs beschriebenen Zusammenhänge zwischen Ereignissen und Funktionen in Konzepte der Trigger- und Aktionssteuerung umzuwandeln.

Auf die Darstellung des DV-Konzepts unter Leistungsgesichtspunkten wird verzichtet, da die Leistungssicht keine spezifischen Implementierungsverfahren bereitstellt [47].

Übungsaufgabe 2.4 (Modellierung auf DV-Konzeptebene) *Auf der DV-Konzeptebene der Datensicht erfolgt die Umsetzung des Datenmodells der Fachkonzeptebene. Überführen Sie das in Aufgabe 2.3 erstellte ERM in eine Tabellenstruktur. Setzen Sie dabei die Beziehungen der Entitäten entsprechend um. Berücksichtigen Sie Kardinalitäten. Eine Normalisierung ist nicht durchzuführen.*

2.3.2.4 Implementierung

Die Umsetzung der Fachinhalte in DV-Programme ist Gegenstand der Implementierung. Dabei setzt die Implementierung unmittelbar auf das DV-Konzept in den Sichten Organisation, Funktionen, Daten und Steuerung auf. Da verschiedene Möglichkeiten zur Konstruktion von Anwendungssystemen in den Kurseinheiten zur Konstruktion genauer beschrieben werden, gehen wir an dieser Stelle nur sehr knapp auf die prinizipiellen Möglichkeiten zur Realisierung der unterschiedlichen Sichten des DV-Konzepts ein.

Die durch das DV-Konzept in der Organisationssicht bereitgestellte Netzwerktopologie wird während der Implementierung durch die Auswahl und anschließende Bereitstellung entsprechender Hard- und Software realisiert. Organisationssicht Implementierung

Bei der Implementierung der Datensicht wird die im DV-Konzept bereitgestellte Tabellenstruktur mit Hilfe eines relationalen DBMS implementiert. Dazu

wird zunächst eine „leere“ Datenbank erzeugt, in die dann sukzessive die notwendigen Datenbankobjekte, insbesondere Datenbanktabellen, eingefügt werden.

Datensicht

Bei der Implementierung der Funktionssicht werden entsprechende Module softwaretechnisch umgesetzt, d.h. in einer geeigneten Programmiersprache implementiert.

Implementierung

Im Rahmen der Implementierung der Steuerungssicht wird die Frage behandelt, wie die in der Funktions-, Daten- und Organisationssicht entwickelten Artefakte (Funktionen, Datenbanken, Rechner) auch physisch miteinander verbunden werden können. Wir verweisen an dieser Stellen auf entsprechende Kursabschnitte zu Middleware und zur Kopplung von Anwendungssystemen.

Steuerungssicht Implementierung

Die Leistungssicht wird mit der gleichen Begründung wie beim DV-Konzept an dieser Stelle nicht dargestellt.

2.3.3 Dienstorientierte Informationssysteme

Konventionelle Informationssysteme sind Softwaresysteme, die als unteilbare Softwaresysteme entworfen, konstruiert, ausgeliefert und in den Unternehmen installiert werden. Konventionelle Informationssysteme werden in Form von CDs, DVDs und entsprechender Dokumentation an die Kunden ausgeliefert. Die Software wird dann von den Kunden auf ihrer Hardware installiert und kann nach Bezahlen der Lizenzgebühr genutzt werden [26]. Wir betrachten das nachfolgende Beispiel.

Konventionelle Informationssysteme

Beispiel 2.3.7 (Konventionelles Informationssystem) *ERP-Systeme wie das System SAP R/3 sind konventionelle Informationssysteme.*

Charakteristisch für konventionelle Informationssysteme ist deren Realisierung in Form von betriebswirtschaftlicher Standardsoftware.

Für betriebswirtschaftliche Standardsoftware ist ein großer Funktionsumfang typisch (vergleiche die Ausführungen in Abschnitt 5.3.2 dieses Kurses). Dieser wird benutzerspezifisch geeignet eingeschränkt. Trotzdem wird aber auch von der verbleibenden Funktionalität ein bestimmter Teil nur temporär genutzt. Nicht oder selten benutzte Funktionalität verbraucht Speicherplatz und belastet die Hardware, außerdem werden auch die Betriebskosten erhöht, da mehr Personal für Nutzung und Betrieb des Softwaresystems notwendig ist und ein größerer Aufwand bei einem Umstieg auf eine neue Version der Software anfällt.

Dienstorientierte Informationssysteme versuchen, die Nachteile konventioneller Informationssysteme zu verringern. Wir definieren dazu zunächst den Dienstbegriff.

Dienste

Definition 2.3.13 (Dienst) *Dienste sind wohldefinierte, eigenständige Module, die Standardgeschäftsfunktionalität zur Verfügung stellen. Sie sind unabhängig vom Zustand oder dem Kontext anderer Dienste. Dienste werden durch Standarddefinitionssprachen beschrieben, stellen öffentliche Schnittstellen zur Verfügung und kommunizieren mit den Dienstenutzern, um die Erfüllung einer Geschäftsaufgabe im Rahmen von Geschäftsprozessen sicherzustellen [17].*

Zur Implementierung von Diensten werden heute typischerweise Webservice-Standards wie Web-Service-Description-Language (WSDL), Simple-Object-Access-Protocol (SOAP) und Universal-Description-Discovery-and-Integration (UDDI) eingesetzt. Diese Technologien werden deshalb in Abschnitt 4.3 dieses Kurses behandelt.

Nach der Einführung des Dienstebegriffes erläutern wir nun den Begriff eines dienstorientierten Informationssystems.

Dienstorientierte Informationssysteme

Definition 2.3.14 (Dienstorientiertes Informationssystem) *Unter einem dienstorientierten Informationssystem verstehen wir ein System, das vom Benutzer als Portal wahrgenommen wird, bei dem er sich anmeldet und die benötigten Dienste in Anspruch nimmt. Ein Portal ist ein technischer Rahmen, der nur wenig fachliche Logik bereitstellt.*

Portal

Das Portal ist dafür verantwortlich, dass die vom Benutzer benötigten Dienste, zu deren Ausführung er berechtigt ist, zusammengesucht und durch die Portaloberfläche einheitlich angeboten werden. Die Dienste können dabei sowohl von lokalen Systemen als auch über das Internet angeboten werden. Der Benutzer verwendet die Funktionen sowie die Ablaufsteuerung (Workflow) und bemerkt keinen Unterschied, ob lokale Funktionen oder Funktionen, die über das Internet verfügbar sind, angeboten werden. Diese Situation ist in Abbildung 2.11 dargestellt.

Wir betrachten das nachfolgende Beispiel für Dienste.

Beispiel 2.3.8 (Dienste eines ERP-Systems) *Ein Mitarbeiter kann in einem Produktionsunternehmen den Dienst „Terminierung eines Kundenauftrags“ oder den Dienst „ Kapazitätsplanung“ in Anspruch nehmen. Für den Mitarbeiter muss in diesem Fall im ERP-System des Unternehmens ein Profil „Produktionsplanung“ hinterlegt sein.*

Da in dienstorientierten Informationssystemen die Trennung zwischen lokal installierten und entfernten Funktionen und Diensten aufgehoben ist, müssen lediglich noch häufig benutzte Funktionen sowie Funktionen, die Wettbewerbsvorteile versprechen, lokal vorgehalten werden. Weniger häufig benutzte und weniger Wettbewerbsvorteile versprechende Funktionen können über das Internet bezogen werden.

Im Gegensatz zur Lizenzierung ganzer Informationssysteme im konventionellen Fall können bei dienstorientierten Informationssystemen lediglich die benötigten Dienste abonniert werden. Ein Abonnement kann zeitlich befristet sein. Die Dienste müssen nicht notwendigerweise vom Unternehmen des Anwenders betrieben werden.

Die Funktionalität von betrieblichen Informationssystemen ist traditionell eng an die Aufbauorganisation eines Unternehmens angelehnt, z.B. an die aufbauorganisatorischen Einheiten Produktion und Vertrieb. Andererseits berücksichtigen die Geschäftsprozesse nur partiell die Aufbauorganisation. Aufgrund

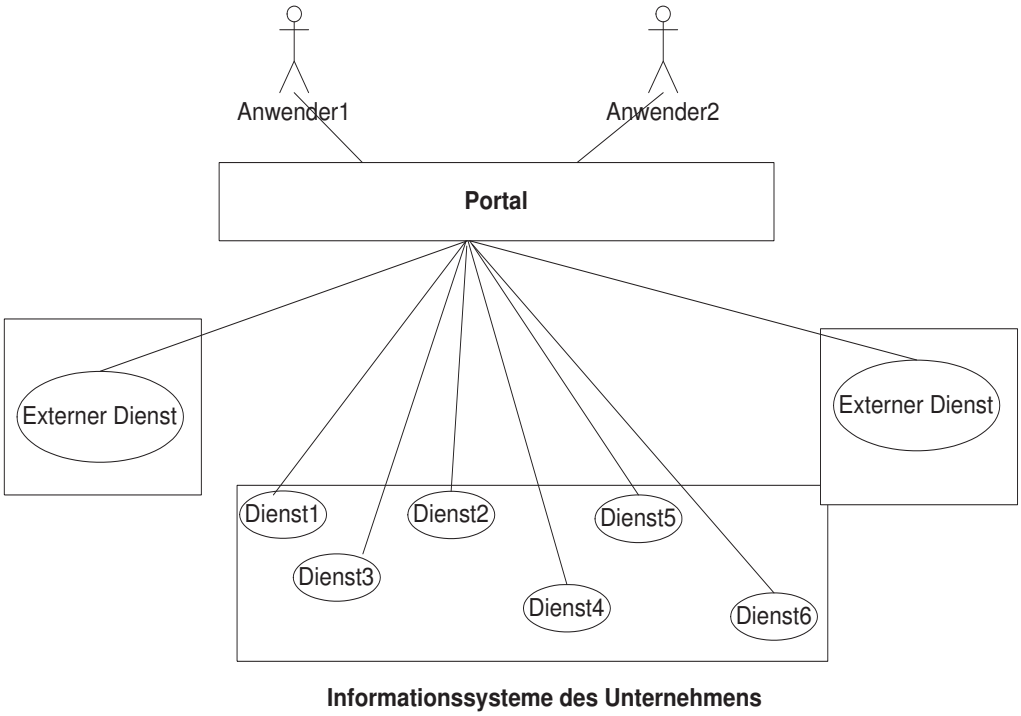

Abbildung 2.11: Zugriff auf ein dienstorientiertes Informationssystem über ein Portal

der Geschäftsprozessorientierung ist folglich die an der Aufbauorganisation angelehnte Zuordnung von Funktionen zu Anwendungssystemen nicht mehr zeitgemäß. Auch dieser Gesichtspunkt legitimiert und begründet die weitere Ausbreitung dienstorientierter Informationssysteme.

Durch das Internet begünstigt, verschwinden die Unternehmensgrenzen. Netzwerke aus Produktionsunternehmen, Lieferanten, Handelsunternehmen und Dienstleistern nehmen die Wertschöpfung vor. Der Endkunde ist ebenfalls wesentlicher Bestandteil derartiger Netzwerke [19, 30]. Supply-Chain-Management- und Customer-Relationship-Management-Systeme sind Informationssysteme, die die Grenzen der betrieblichen Informationsverarbeitung verlassen und folglich dienstorientiert realisiert werden. Dienstorientierte Informationssysteme verwischen die Grenzen zwischen inner- und zwischenbetrieblicher Integration.

Übungsaufgabe 2.5 (Identifikation von Diensten) *Modellieren Sie mit Hilfe ereignisgesteuerter Prozessketten (EPK) den Geschäftsprozess der Auftragsabwicklung vom Eingang des Kundenauftrags bis zur Freigabe für die Fertigung. Nach dem Eingang des Kundenauftrags wird dieser bearbeitet und grob terminiert. Danach wird der Kunde über den möglichen Fertigstellungstermin und den Preis unter Berücksichtigung möglicher Rabatte informiert. Nimmt der Kunde das Angebot an, so wird eine Kapazitätsplanung durchgeführt und der*

Auftrag in die Fertigung eingesteuert. Erstellen Sie die EPK und identifizieren Sie mögliche Dienste, die bei der Abarbeitung des Geschäftsprozesses in Anspruch genommen werden.

2.3.4 Service-orientierte Architekturen

2.3.4.1 Begriffsbildungen

Service-orientierte Architekturen dienen dem Entwurf von Systemlandschaften. Wir führen dazu zunächst den Begriff der Systemlandschaft ein.

Systemlandschaften

Definition 2.3.15 (Systemlandschaft) *Eine Systemlandschaft besteht aus einer Menge von Hardware- und Softwarebausteinen, die zur Realisierung der Anwendungssysteme eines Unternehmens verwendet werden.*

Systemlandschaften besitzen die nachfolgenden Eigenschaften [52]:

- Systemlandschaften bilden sich im zeitlichen Verlauf heraus. Sie sind in den meisten Fällen historisch gewachsen. Fachliche und technische Anforderungen verändern sich. Systemlandschaften befinden sich deshalb in einem ständigen Wandel.
- Für Systemlandschaften sind redundante Daten und Funktionen typisch.
- Systemlandschaften sind häufig heterogen, d.h., sie enthalten Systeme unterschiedlicher Hersteller und unterschiedlichen Alters, die unter Verwendung verschiedener Technologien entwickelt wurden.

Die heutigen Systemlandschaften enthalten häufig eng gekoppelte Altsysteme mit redundanten Daten und Funktionen. Neue fachliche und technische Anforderungen können nur mit großem Aufwand umgesetzt werden. Das SOA-Konzept verspricht eine Lösung für die Probleme bei der Gestaltung heutiger Systemlandschaften.

Wir definieren zunächst den Begriff der engen und losen Kopplung von Anwendungssystemen [52].

Enge und lose Kopplung

Definition 2.3.16 (Enge und lose Kopplung von Systemen) *Ein System S_1 heißt eng gekoppelt an ein System S_2, wenn S_1 voraussetzt, dass S_2 verfügbar ist. S_1 und S_2 sind eng gekoppelt, wenn S_1 eng an S_2 gekoppelt ist und umgekehrt. Die Systeme S_1 und S_2 sind lose gekoppelt, wenn weder S_1 an S_2 noch S_2 an S_1 eng gekoppelt ist.*

Eine enge Kopplung von Systemen wird oft durch synchrone Aufrufe verursacht. Eine lose Kopplung von Systemen wird durch asynchrone Aufrufe unterstützt.

Dienstorientierte Informationssysteme können in Form service-orientierter Architekturen implementiert werden. Systeme, die Dienste zur Verfügung stellen, werden als eigene Softwarekomponenten aufgefasst, die eine beliebige Anzahl

von Schnittstellen implementieren, die das System seinen Partnern zur Verfügung stellt. Das System nimmt umgekehrt seine Partner nur über definierte Schnittstellen wahr. Wir erläutern den Komponentenbegriff in Abschnitt 2.6.3 dieser Kurseinheit genauer.

Dienstorientierte Informationssysteme kommunizieren asynchron, behandeln Fehler geeignet und verkraften den Ausfall anderer Systeme. Das führt zu einer Entkopplung von Systemen. Die einzelnen Systeme müssen wenig voneinander wissen, da die Ablaufsteuerung in eine eigenständige Workflowkomponente ausgelagert wird.

SOA

Eine Architektur, die wie dargestellt Komponentenorientierung, lose Kopplung und eigenständige Workflowkomponenten beinhaltet, wird als serviceorientiert bezeichnet. Das führt zu nachfolgender Definition.

Definition 2.3.17 (Service-orientierte Architektur) *Unter einer serviceorientierten Architektur (SOA) verstehen wir ein Architekturkonzept für betriebliche Informationssysteme, durch dessen Umsetzung eine erhöhte Interoperabilität zwischen Anwendungen und Wiederverwendbarkeit von Softwareartefakten erreicht wird [40, 63, 42]. Service-orientierte Architekturen dienen dazu, die Anforderungen einer lose gekoppelten, auf Standards basierenden verteilten Informationsverarbeitung zu erfüllen. Das wird erreicht, indem das betriebliche Informationssystem eine ähnliche Struktur wie die zu unterstützenden Geschäftsprozesse besitzt. Die Ablaufsteuerung ist bei einer SOA-Architektur in eine eigene Workflowkomponente ausgelagert.*

Im Rahmen des Konzeptes werden Funktionen in Form von wiederverwendbaren, geschäftlich abgestimmten, voneinander unabhängigen und lose gekoppelten Diensten (Services) implementiert. In Abbildung 2.12 stellen wir ein monolithisches Anwendungssystem, das die Funktionalitäten A, B und C bereitstellt, einer SOA mit den Diensten A, B, und C gegenüber. Eine Systemlandschaft besteht typischerweise aus vielen monolithischen Anwendungssystemen.

SOA-Modellebenen

Im Sinne des generischen Architekturrahmens aus Abschnitt 2.2 werden SOAs durch drei unterschiedliche Modelle, genauer Modellebenen, repräsentiert [52]:

1. Prozessmodell,
2. Dienstemodell,
3. Technikmodell.

Das Prozessmodell dient der Beschreibung von Geschäftsprozessen. Eine Modellierung von Geschäftsprozessen ist erforderlich, um geeignete Dienste zu definieren. Das Dienstemodell dient zur Beschreibung von Diensten und deren Bezug zu Geschäftsprozessen. Das Dienstemodell ermöglicht eine Durchführung der Geschäftsprozesse. Es ist wünschenswert, wenn das Prozessmodell das Dienstemodell bestimmt. Umgekehrt beeinflusst das Dienstemodell zu einem gewissen

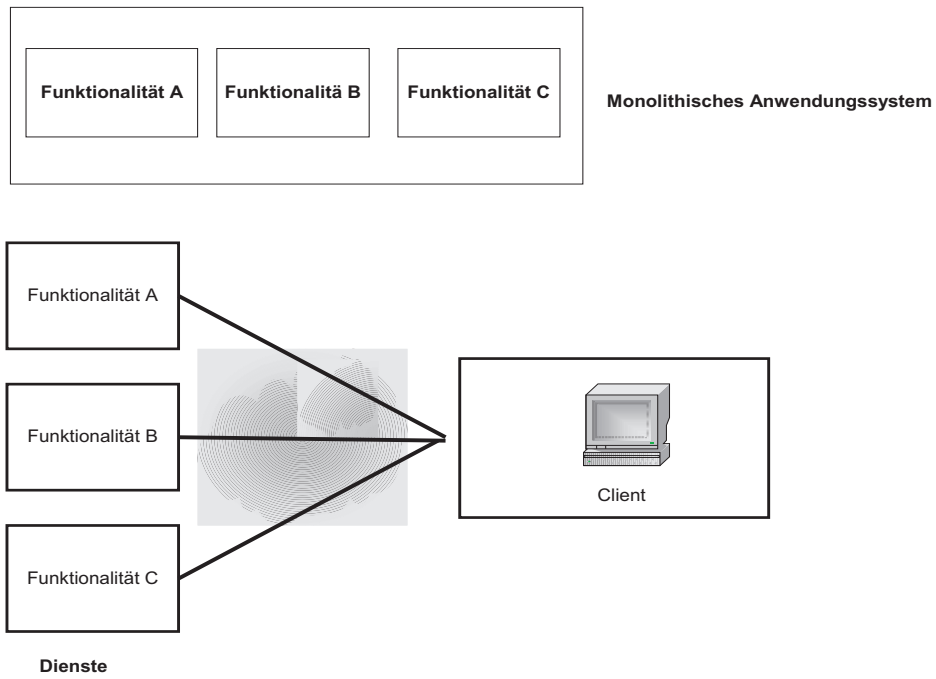

Abbildung 2.12: Monolithisches Anwendungssystem vs. SOA

Grad auch das Prozessmodell, da insbesondere betriebswirtschaftliche Standardsoftware und Altsysteme häufig bestimmte Geschäftsprozesse nur unzureichend unterstützen.

Das Technikmodell dient erwartungsgemäß dazu, festzulegen, mit welchen Technologien Dienste effizient implementiert werden können. Wir beschreiben entsprechende Möglichkeiten in Abschnitt 4.2. Das Technikmodell ist somit plattformabhängig. Durch das Dienstemodell wird die Anwendung von der Technik entkoppelt. Das Dienstemodell bestimmt das Technikmodell. Umgekehrt beeinflusst das Technikmodell geringfügig auch das Dienstemodell. Wir stellen die wechselseitigen Abhängigkeiten von Prozess-, Dienste- und Technikmodell in Abbildung 2.13 dar.

Das nachfolgende Beispiel verdeutlicht, wie sich Dienste- und Technikmodell wechselseitig beeinflussen können.

Beispiel 2.3.9 (Zusammenhang von Dienste- und Technikmodell) *Die Verfügbarkeit und die Antwortzeit eines Dienstes bestimmen in entscheidendem Maße, wie der Dienst technisch zu implementieren ist. Umgekehrt können zum Beispiel Konzernentscheidungen bezüglich der Verwendung bestimmter Kommunikations- und Softwaretechnologien Einfluss auf die Auswahl von Diensten haben.*

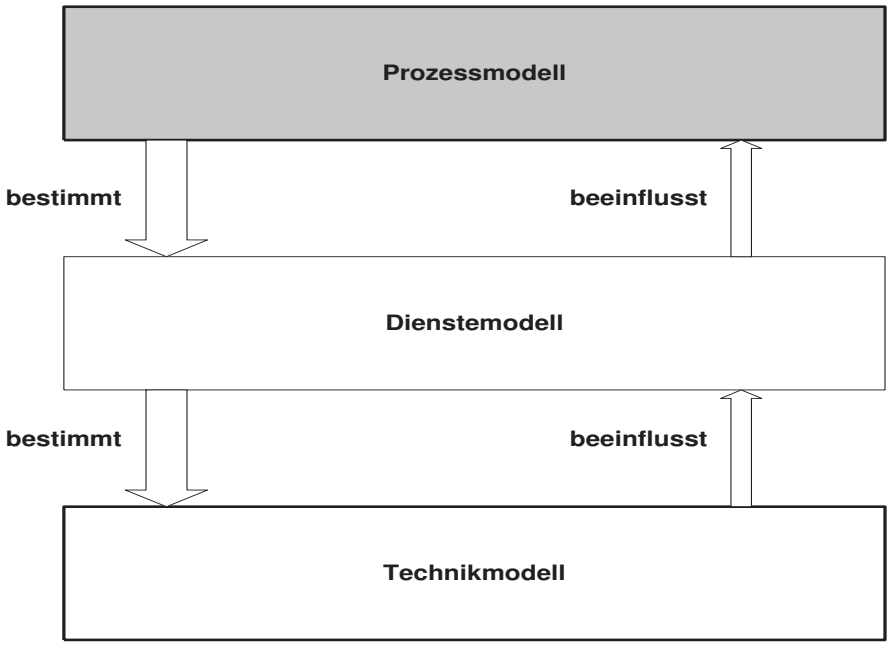

Abbildung 2.13: Modellebenen einer SOA

Dienste-eigen-schaften

Durch das SOA-Konzept wird die Implementierung dienstorientierter Informationssysteme vollzogen. Ein Dienst in einer SOA ist logisch ein Paar

$$< s\_inf, s\_impl > \tag{2.19}$$

bestehend aus der Schnittstelle s_inf des Dienstes sowie einer Diensteimplementierung s_impl. Die Schnittstelle s_inf beschreibt die Identifizierungs- und Aufruflogik des Dienstes. Die Implementierung s_impl implementiert die Funktionalität, die der Dienst bereitstellen soll. Ein Dienst innerhalb einer SOA besitzt die nachfolgenden wesentlichen Eigenschaften [42]:

- Die gesamte durch eine SOA zur Verfügung gestellte Funktionalität besteht aus Diensten. Darin enthalten sind Dienste, die ausschließlich der Unterstützung fachlicher Anforderungen dienen, Dienste, die Geschäftstransaktionen abbilden sowie Dienste, die reine Funktionalität für den Betrieb des Anwendungssystems zur Verfügung stellen.

- Die Dienste verfügen während der Interaktion mit dem Dienstnehmer über einen eigenen Kontext. In diesem Sinne sind Dienste in einer SOA in sich abgeschlossen.

- Die Dienste sind autonom. Externe Komponenten interessieren sich nur für das Ergebnis, das der Dienst zurückliefert. Die interne Verarbeitungslogik ist für die externen Komponenten dabei unbekannt.

- Dienste in einer SOA sind plattformunabhängig. Das bedeutet insbesondere, dass der durch die Schnittstelle des Dienstes repräsentierte Vertrag zwischen Dienstnutzer und Dienst nur plattformunabhängige Zusicherungen enthalten kann.

- Die Dienste in einer SOA können dynamisch lokalisiert, aufgerufen und miteinander kombiniert werden. Wesentlich ist dabei lediglich die Aufruffähigkeit. Es ist nicht interessant, ob Dienste lokal oder entfernt ausgeführt werden können oder welche Infrastrukturkomponenten benötigt werden, um die Verbindung zum Dienst herzustellen.

Da die Schnittstellen plattformunabhängig sind, kann ein Client, der für ein beliebiges Betriebssystem in einer beliebigen Programmiersprache entwickelt wurde, im Prinzip den Dienst benutzen. Das durch eine SOA verfolgte Ziel einer losen Kopplung von Diensten eröffnet Möglichkeiten für eine leichtere Implementierung und Wartung von unternehmensweiten und zwischenbetrieblichen Informationssystemen.

SOA stellt eine Entwurfsphilosophie dar, bei der die eigentlichen Geschäftsprozesse im Vordergrund stehen und die realisierende Technologie in den Hintergrund rückt [7]. Der Grundgedanke dieses Konzeptes ist die Trennung der Zuständigkeiten nach fachlichen Gesichtspunkten und die Kapselung technischer Details in Diensten. Das Architekturkonzept gibt für die Erstellung der Dienste Strukturmuster vor [3]. Das Konzept hat einen architektonischen Fokus, der Steuerung, Prozesse, Modellierung und Werkzeuge beinhaltet. Es besitzt einen günstigen Abstraktionsgrad, um Geschäftsbelange und technische Ressourcen miteinander zu verbinden und wiederverwendbare, grobgranulare Geschäftsfunktionen zu erstellen. SOA bietet eine Entwicklungsinfrastruktur, mit deren Hilfe neue Anwendungssysteme schnell und einfach erstellt werden können. Die Software-Wiederverwendung wird durch die Möglichkeit des Aufbaus einer Bibliothek von wiederverwendbaren Diensten für Geschäftsprozess- und Informationsverarbeitungsfunktionalität unterstützt [40].

2.3.4.2 Diensterollen in service-orientierten Architekturen

Rollen in einer SOA

Im Rahmen einer SOA werden, wie in [3, 42] vorgeschlagen, die nachfolgenden zwei Rollen unterschieden :

1. Service-Provider,

2. Service-Requester (Service-Consumer).

Der Service-Requester (Dienstnachfrager, Dienstnehmer) möchte den angebotenen Dienst nutzen. In diesem Sinne fungiert er als Client. Der Service-Provider (Dienstanbieter) stellt den gewünschten Dienst zur Verfügung. Er agiert somit

als Server. Die Anforderung eines Dienstes erfolgt durch Nachrichten. Da Dienste einer SOA in der Mehrzahl der Fälle softwaretechnisch als Webservices realisiert werden, werden die der Kommunikation von Webservices zugrundegelegten SOAP-Nachrichten ausführlich in Abschnitt 4.3 beschrieben. Wir weisen an dieser Stelle erneut darauf hin, dass der Dienstnehmer sich nicht für entsprechende Implementierungsdetails des Dienstes zu interessieren braucht. Es wird lediglich vorausgesetzt, dass der Dienst die erforderliche Funktionalität in einer entsprechenden Qualität zur Verfügung stellt. Dienstanbieter und Dienstnehmer schliessen einen Vertrag ab, der festlegt, wie ein Dienstnehmer mit dem Dienstanbieter zusammenwirkt. Der Vertrag legt unter anderem Syntax und Semantik der Anfrage und der Antwort fest. Dabei wird auch beschrieben, wie die Nachrichten zwischen Dienstnehmer und -anbieter zu transportieren sind. Vorbedingungen sowie Nachbedingungen der Dienstnutzung werden angegeben. Außerdem wird festgelegt, in welcher Qualität der Dienst zur Verfügung gestellt werden muss („Quality-of-Service (QoS)“) und wie lange der Vertrag gültig sein soll. Als Binding wird die Angabe des durch den Dienst verwendeten Nachrichtenformats, die Angabe des Protokolls zur Nachrichtenübertragung sowie die Angabe der Netzwerkadresse, unter der man den Dienst erreichen kann, bezeichnet.

Beispiel 2.3.10 (Dienstgüte) *Die Zeit, die zur Ausführung eines Dienstes benötigt wird, ist ein Beispiel für die Qualität eines Dienstes. Weitere Beispiele sind Verfügbarkeit, Zuverlässigkeit sowie Sicherheit des angebotenenen Dienstes.*

Falls Dienstnehmer und Dienstanbieter direkt miteinander in Beziehung treten, ist der Dienstnehmer für die Auswahl eines entsprechenden Dienstanbieters, anschließende Verhandlungen und Reservierungen selber verantwortlich. Dies stellt eine hohe Belastung für die Clients dar. Aus diesem Grund ist es naheliegend, diese Funktionen in einer zusätzlichen Rolle zu kapseln. Die neue Rolle fasst die Rolle eines Dienstanbieters und Dienstnehmers zusammen. Wir sprechen in diesem Zusammenhang von einer Dienstaggregation. Die neue Rolle wird als **Dienstaggregator** bezeichnet. Der Dienstaggregator spielt eine duale Rolle:

1. Der Dienstaggregator fungiert als Application-Service-Provider, der eine Lösung für ein bestimmtes Problem anbietet. Diese Lösung kann dadurch erreicht werden, dass neue Dienste aus bereits existierenden Diensten zusammengesetzt werden. Dabei kommen typischerweise Orchestrierungs- und Choreographiesprachen wie zum Beispiel die Business-Process-Execution-Language (BPEL) [32] zum Einsatz. Auf BPEL wird genauer in der Kurseinheit über die Konstruktion von dienstorientierten Informationssystemen eingegangen.

2. Der Dienstaggregator agiert als Dienstnachfrager, der Dienste bei anderen Dienstanbietern nachfragt und gegebenenfalls diese Dienste reserviert. Der Dienstaggregator muss dazu geeignete Dienstanbieter ausfindig machen.

Abbildung 2.14 dient dazu, das Zusammenwirken von Dienstanbieter, -nachfrager und -aggregator in Form eines UML-Klassendiagramms darzustellen. Wir verweisen in diesem Zusammenhang auf Abschnitt 3.4.3 der nächsten Kurseinheit für die Darstellung der UML-Notation sowie auf vorangegangene Kurse.

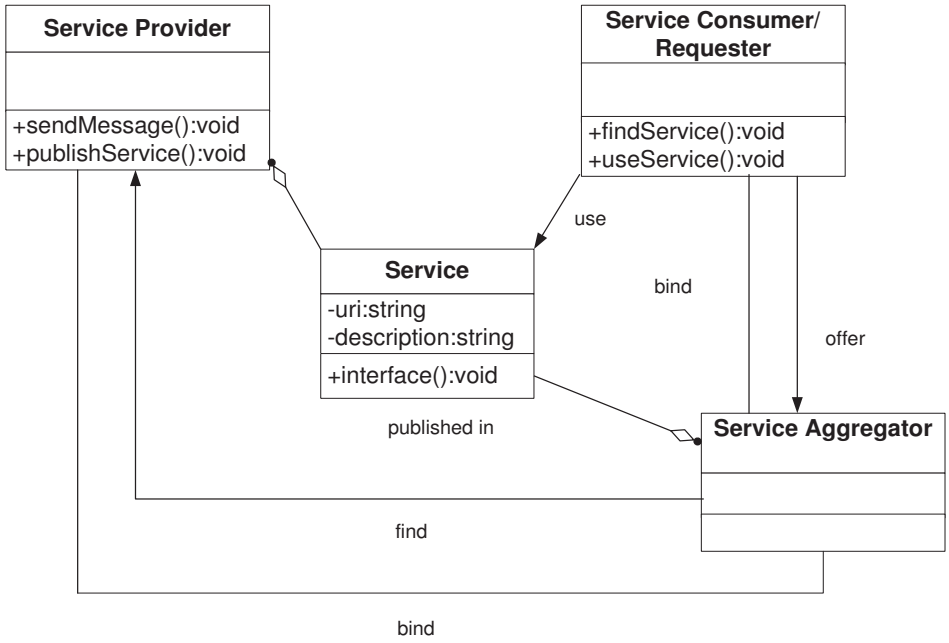

Abbildung 2.14: Zugriff auf Dienste durch Dienstaggregation

Broker für Dienste

Aus Abbildung 2.14 wird deutlich, dass ein Dienstaggregator eine Form des Dienste-Brokering darstellt, der insbesondere dem Dienstnachfrager nutzt. Allerdings ist noch nicht geklärt, wie ein Dienstnehmer potentielle Dienstanbieter im ersten Schritt identifiziert und anschließend auswählt. Dazu ist es erforderlich, die Rolle eines **Dienst-Brokers** innerhalb einer SOA einzuführen.

Der Dienstanbieter veröffentlicht die Dienstbeschreibung in einer Dienste-Registry beim Dienst-Broker und stellt den Dienst zur Ausführung bereit. Wesentlicher Bestandteil eines Dienst-Brokers ist somit ein Repository, in dem insbesondere neben einem Namensverzeichnis auch Dienstbeschreibungen abgelegt sind. Dadurch wird der Dienstnehmer in die Lage versetzt, Auswahlentscheidungen zwischen den Diensten vorzunehmen. Der Dienstnehmer nutzt die zur Verfügung gestellten Dienste. Er kann sie beim Dienst-Broker auffinden, indem er entweder direkt den Uniform-Resource-Identifier (URI) des Dienstes oder eine Dienstbeschreibung in der Dienste-Registry nutzt. Der Dienst-Broker wiederum stellt die Dienste-Registry zur Verfügung und verwaltet diese. Das Zusammenwirken von Dienstanbieter, Dienstnachfrager und Dienst-Broker wird in Abbildung 2.15 als UML-Klassendiagramm dargestellt.

Wir werden in Abschnitt 4.3 sehen, dass der Verzeichnisdienst UDDI als spezieller Dienst-Broker für Webservices aufgefasst werden kann. In diesem Fall werden die Beschreibungen der Dienste als WSDL-Dateien abgelegt.

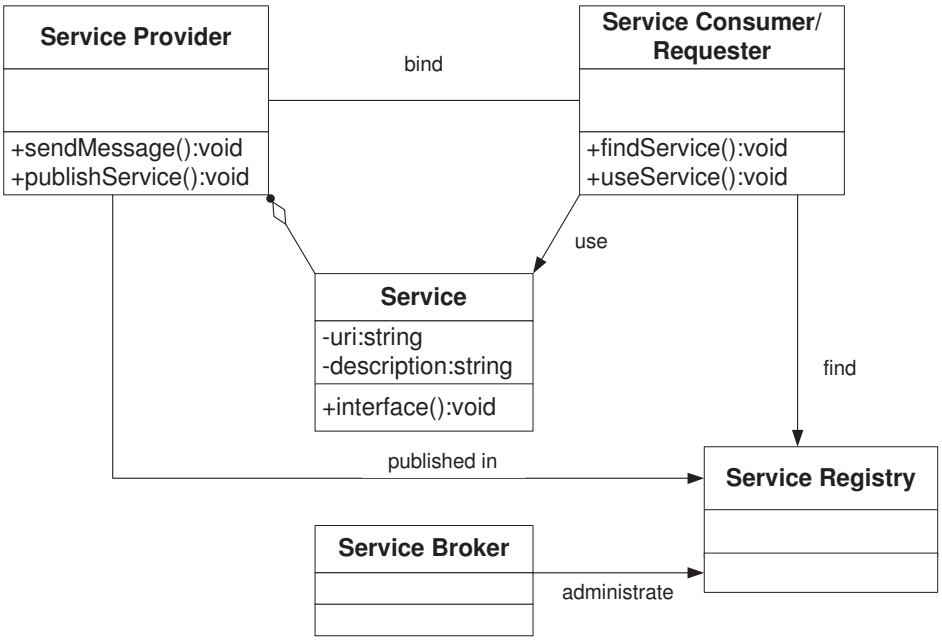

Abbildung 2.15: Zugriff auf Dienste über einen Broker

Wir bemerken, dass auch Kombinationen der in Abbildung 2.14 und 2.15 dargestellten Vorgehensweisen möglich sind. Dabei wird ausgenutzt, dass die Dienstaggregatoren ebenfalls wieder einen Dienst zur Verfügung stellen.

Wir möchten an dieser Stelle die Sicht eines einzelnes Dienstes verlassen und darauf eingehen, wie verschiedene Dienste zusammenwirken können. Bisher wurde das bei der Einführung des Dienstaggregators nur angedeutet.

2.3.4.3 Komposition von Diensten

Komposition

Wir führen dazu zunächst in Anlehnung an [44] einen Kompositionsbegriff ein.

Definition 2.3.18 (Komposition von Diensten) *Unter einer Komposition von Diensten versteht man das Zusammenfügen mehrerer Dienste zu einer neuen Anwendung. Komponierte Dienste können wieder selber als Dienst zur Verfügung gestellt werden.*

Ansätze zur Komposition von Diensten können den ganzen Lebenszyklus von Diensten umfassen. Eine Komposition von Diensten kann somit bereits beim Auffinden von Diensten vorgenommen werden, ist aber oft auch noch bei der

Überwachung von Diensten (Monitoring) sinnvoll. Wir unterscheiden zwischen proaktiver und reaktiver Komposition in Abhängigkeit davon, wann feststeht, welche Dienste in der Anwendung Verwendung finden sollen. Das führt zu folgender Definition.

Proaktive Komposition

Definition 2.3.19 (Proaktive Komposition) *Wenn alle zu verwendenden Dienste a priori bekannt sind und die Anwendung bereits zur Entwicklungszeit vollständig zusammengestellt werden kann, sprechen wir von einer proaktiven Komposition.*

Eine proaktive Komposition eignet sich besonders für Situationen, in denen die Anwendung oft benutzt wird, stabil sein muss sowie die Verarbeitungsgeschwindigkeit eine Rolle spielt. Im Gegensatz dazu ist eine reaktive Komposition besonders für dynamische Situationen geeignet. Wir definieren deshalb den Begriff der reaktiven Komposition wie folgt.

Reaktive Komposition

Definition 2.3.20 (Reaktive Komposition) *Wenn ein neuer Dienst erst zu dem Zeitpunkt, zu dem er benötigt wird, komponiert wird, sprechen wir von einer reaktiven Komposition.*

Reaktive Komposition kommt zum Einsatz, wenn der zu komponierende Dienst nur selten eingesetzt wird und die einzelnen Dienste, die benötigt werden, nicht a priori bekannt sind. Außerdem besteht bei reaktiver Komposition die Möglichkeit, noch zur Laufzeit der Anwendung Dienste hinzuzufügen oder auszutauschen.

Beispiel 2.3.11 (Reaktive Komposition) *Ein Unternehmen erstellt ein neues Produkt. Die Materialbeschaffung für dieses Produkt soll durch einen Dienst unterstützt werden. Wenn das Unternehmen die Komponenten des Produkts ermittelt hat, die durch Fremdbeschaffung bezogen werden sollen, kann ein Einkaufsagent automatisch Angebote geeigneter Anbieter einholen. Dabei greift der Einkaufsagent auf eine geeignete Datenbank zurück. Die Angebotsbeschaffung ist ebenfalls als Dienst implementiert.*

Konversation

Wir erläutern nun verschiedene Möglichkeiten zur Ausgestaltung von Dienstkompositionen. Wesentlich sind dabei die Begriffe Konversation, Choreographie und Orchestrierung. Wir führen dazu zunächst den Begriff der Konversation ein.

Definition 2.3.21 (Konversation) *Unter einer Konversation verstehen wir die Kommunikation zwischen zwei oder mehreren Diensten.*

Choreographie

Die Konversation muss bestimmten Regeln folgen. Das führt uns zum Begriff der Choreographie.

Definition 2.3.22 (Choreographie von Diensten) *Unter einer Choreographie verstehen wir das Herstellen zulässiger Nachrichtenabfolgen zwischen einem oder mehreren Diensten.*

Der Choreographiebegriff für Dienste greift das Bild der Inszenierung von Bewegungsabläufen beim Ballett auf. Choreographie wird alternativ auch als Koordination bezeichnet. Die in Abschnitt 7.1.5.4 dieses Kurses vorgestellten Interaktionsprotokolle dienen in diesem Sinne der Choreographie. Wir betrachten das nachfolgende Beispiel zum besseren Verständnis des Choreographiebegriffes.

Beispiel 2.3.12 (Choreographie) *Wir nehmen ein Szenario an, das aus einem Kunden besteht, der bei einem Unternehmen ein bestimmtes Produkt bestellt. Damit dieses Produkt ausgeliefert werden kann, ist es notwendig, dass bestimmte Teile von zwei weiteren Unternehmen bezogen werden. Aus diesem Grund sendet das Herstellerunternehmen Anfragen an die beiden Lieferanten. Erst wenn beide Lieferanten ihre Lieferbereitschaft angezeigt haben, kann der Kunde über eine erfolgreiche Bestellung benachrichtigt werden. Der beschriebene Geschäftsprozess ist in Abbildung 2.16 als UML-Sequenzdiagramm dargestellt. In Abbildung 2.17 wird das dazugehörige Koordinationsprotokoll als UML-Aktivitätsdiagramm dargestellt.*

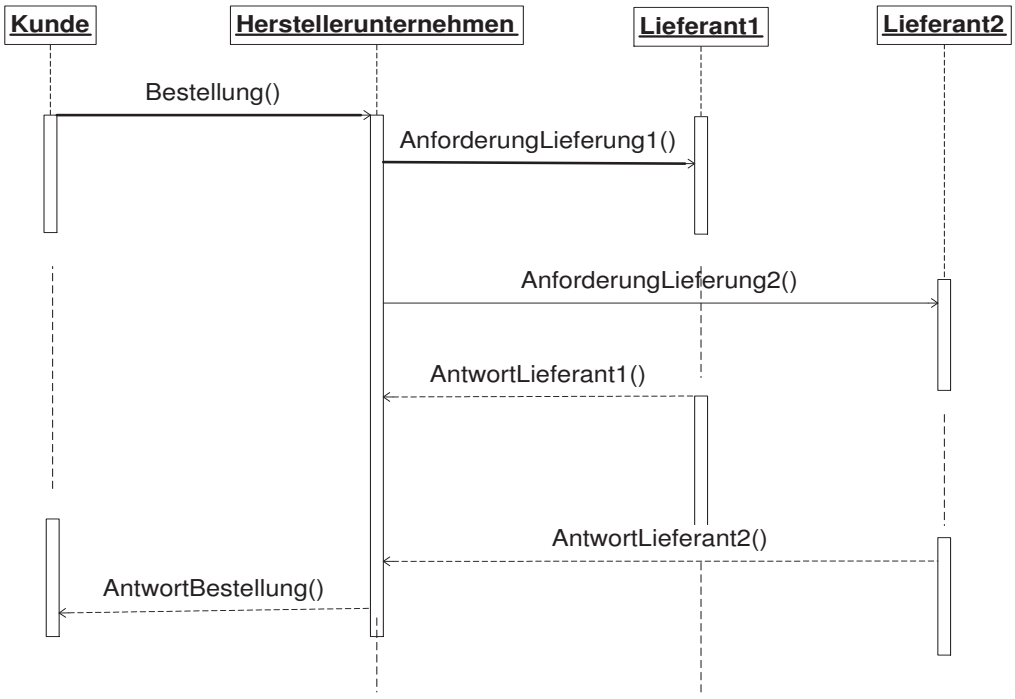

Abbildung 2.16: Konversation zwischen Kunde, Hersteller und zwei Lieferanten

Eine Choreographie ist nicht als Prozess ausführbar. Sie dient der Darstellung des Nachrichtenaustauschs aus einer öffentlichen Sicht, die interne Prozesslogik der Teilnehmer wird aber nicht abgebildet.

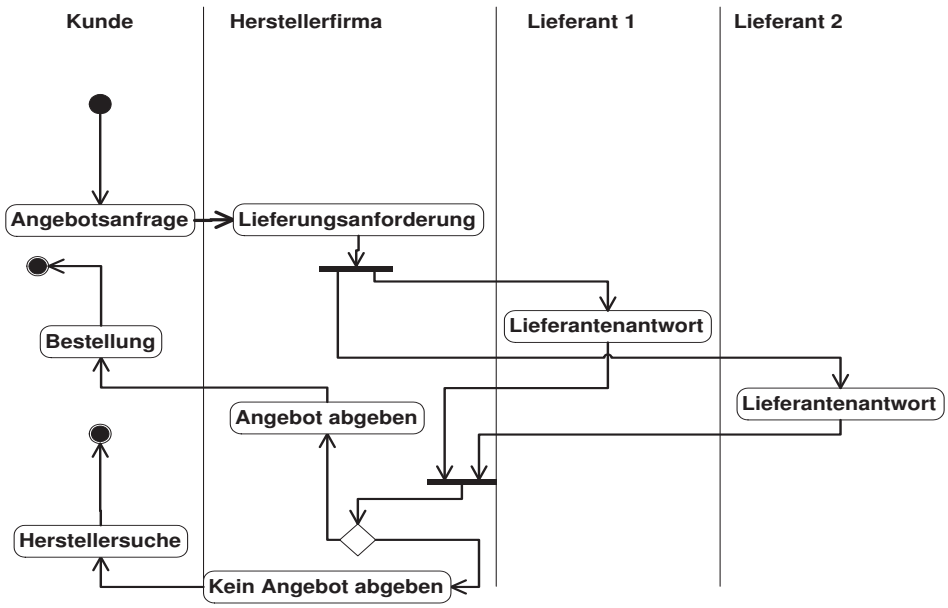

Abbildung 2.17: Koordinationsprotokoll für Kunde, Hersteller und Lieferanten

Beispiel 2.3.13 (Fehlende Abbildung der Prozesslogik) *Im vorher angegebenen Beispiel 2.3.12 wird nicht dargestellt, unter welchen Bedingungen die Herstellerfirma kein Angebot abgibt.*

Häufig ist es aber wünschenswert, die Geschäftsprozesslogik aus der Sicht eines Teilnehmers abzubilden. Dazu dient die Orchestrierung von Diensten. Orchestrierung

Definition 2.3.23 (Orchestrierung von Diensten) *Eine Orchestrierung beschreibt einen Geschäftsprozess, der den Aufruf mehrerer Dienste beinhaltet, aus der Sicht eines Teilnehmers am Geschäftsprozess.*

Der Orchestrierungsbegriff greift somit auf die Metapher eines Dirigenten zurück, der dafür sorgt, dass das Orchester wie von ihm gewünscht tätig wird.

Wir beschreiben den Unterschied von Choreographie und Orchestrierung. Dazu wird der Geschäftsprozess aus Beispiel 2.3.12 herangezogen. Abbildung 2.18 zeigt eine mögliche Orchestrierung, die passend zum Koordinationsprotokoll aus Abbildung 2.17 ist.

Beispiel 2.3.14 (Unterschied Choreographie und Orchestrierung) *Die Abbildung 2.18 zeigt den Geschäftsprozess „Angebotserstellung“ aus Beispiel 2.3.12 aus Herstellersicht. Die durch das Koordinationsprotokoll vorgegebenen Aktivitäten sind dick umrandet. Der interne Geschäftsprozess der Herstellerfirma wird durch gestrichelt umrandete Aktivitäten dargestellt. Nach der Angebotsanfrage durch den Kunden verschickt der Hersteller entsprechend dem*

Koordinationsprotokoll zunächst zwei Anfragen an Lieferanten. Die Daten des anfragenden Kunden werden intern durch den Hersteller gepflegt. Ebenfalls intern werden die Antworten der beiden Lieferanten bewertet. Entsprechend dem Koordinationsprotokoll wird anschließend entschieden, ob ein Angebot an den Kunden abgegeben wird oder nicht.

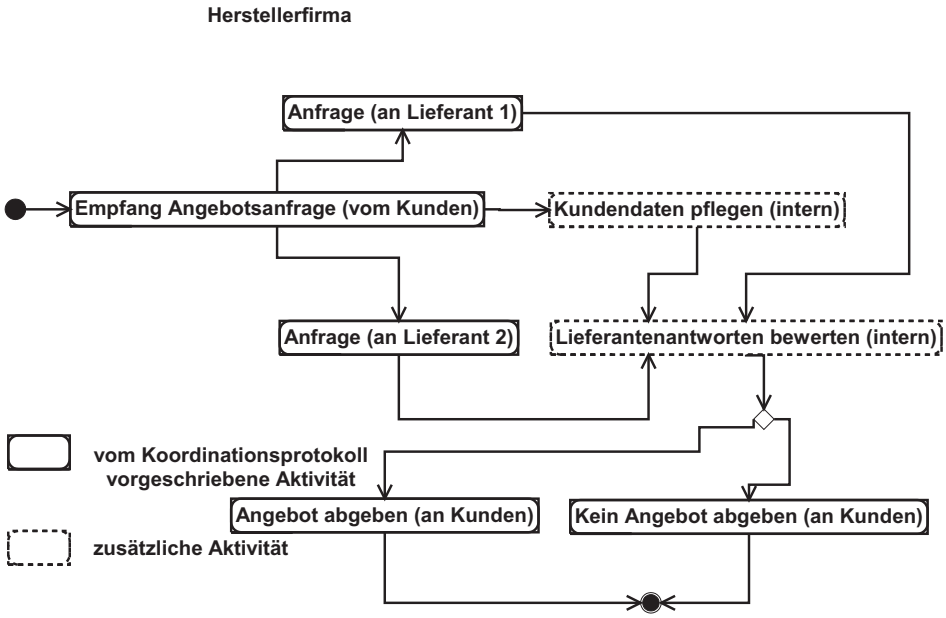

Abbildung 2.18: Orchestrierung von Diensten mehrerer Partner

Aus Beispiel 2.3.14 und Abbildung 2.18 wird deutlich, dass die bei der Orchestrierung auftretenden Aktivitäten, die gestrichelt umrandet sind, für die anderen Teilnehmer des Koordinationsprotokolls nicht sichtbar sind. Ein Unternehmen ist tyischerweise nicht daran interessiert, seine internen Geschäftsprozesse offenzulegen. Die von der Orchestrierung betroffenen Dienste sind somit nur teilweise auch Gegenstand der Choreographie. In Beispiel 2.3.14 ist es nicht von Interesse, dass die Kriterien zur Bewertung der Lieferantenantworten bekannt sind.

Übungsaufgabe 2.6 (Choreographie und Orchestrierung) *Stellen Sie den Unterschied zwischen Choreographie und Orchestrierung am Beispiel der Produktion eines Auftrags dar. An diesem Geschäftsprozess sind der Vertrieb, die Produktion, die Beschaffung und die Buchhaltung beteiligt. Stellen Sie zuerst die Choreographie der Dienste als Sequenzdiagramm dar. Anschließend stellen Sie das Zusammenspiel der Dienste in geeigneter Weise graphisch dar. Betrachten Sie im Anschluss daran sowohl den Dienst der Produktion als auch der Beschaffung. Stellen Sie jeweils die Orchestrierung dieser beiden Dienste graphisch dar. Die Darstellung muss nicht notwendigerweise als Aktivitätsdiagramm erfolgen.*

2.3.4.4 Enterprise-Service-Bus

ESB

Ein Enterprise-Service-Bus (ESB) führt die Kommunikation zwischen unterschiedlichen Diensten durch. Außerdem wird die Kommunikation verwaltet sowie überwacht. ESBs dienen der Übermittlung von Nachrichten von einem Quellsystem zu einem Zielsystem. Das Quellsystem wartet nicht auf die Rückmeldung des Zielsystems. In einer SOA wird durch den ESB ein Export von Diensten durchgeführt. Ein ESB stellt die nachfolgenden Dienste zur Verfügung [52]:

ESB-Dienste

- **Routing:** Ein ESB leitet ähnlich zu einem Object-Broker einer Middleware Nachrichten vom Sender an den Empfänger weiter. Die Nachrichten können sich dabei gegenseitig überholen.

- **Persistenz:** Der ESB ist dafür verantwortlich, dass gesendete Nachrichten nicht verloren gehen. Es wird sichergestellt, dass alle Nachrichten die vorgesehenen Empfänger erreichen. Probleme bereitet die Verarbeitung schreibender Nachrichten. Da der Sender keine Annahmen über die Laufzeit der Nachricht an den Empfänger treffen kann, besitzen Sender und Empfänger häufig einen unterschiedlichen Informationsstand. Es liegen somit Inkonsistenzen vor.

- **Fachliche Transformation:** Die beteiligten Systeme verfügen typischerweise über eine eigene Begriffswelt. Der ESB versteckt die unterschiedlichen Begrifflichkeiten, in dem er die Dienste in der Sprache ihrer Begrifflichkeiten aufruft.

- **Technische Transformation:** Ein ESB versteckt die konkrete Form der Kommunikation vor der Anwendung. Dadurch können Systeme miteinander kommunizieren, die auf unterschiedlichen Plattformen ablaufen und in unterschiedlichen Programmiersprachen erstellt worden sind.

- **Workflow:** Moderne ESB enthalten eine Workflowkomponente. Falls mehrere Dienste in einer bestimmten Reihenfolge auszuführen sind, übernimmt der ESB die Ablaufsteuerung. Wir gehen später in Abschnitt 4.3 auf die Business-Process-Execution-Language ein, die sich als Standard für die Choreographie bzw. Orchestrierung herauskristallisiert hat.

- **Monitoring:** Der ESB speichert, welche Dienste aufgerufen werden. Dabei sind Aufrufhäufigkeit sowie -parameter interessant.

- **Lastverteilung, Ausfallsicherheit und Skalierbarkeit:** Jeder Dienst kann mehrfach registriert werden. Der ESB übernimmt die Lastverteilung. Außerdem besteht die Möglichkeit, dass EBSs in mehreren Instanzen laufen. Eine Masterinstanz übernimmt dann die Aufgabe der Lastverteilung. Dadurch werden eine gute Skalierbarkeit, die prinzipielle Möglichkeit einer parallelen Verarbeitung sowie eine hohe Ausfallsicherheit erreicht.

- **Fehlerbehandlung:** Der ESB berücksichtigt, dass die Aufrufe von Diensten fehlschlagen können. Im Falle eines fehlgeschlagenen Aufrufes werden alle betroffenen Systeme benachrichtigt. Jedes System ist eigenverantwortlich für eine entsprechende Reaktion zuständig.

In Abbildung 2.19 ist die Architektur eines ESB schematisch dargestellt. Der ESB integriert Datenquellen, Java-Platform-Enterprise-Edition (J2EE)-Anwendungen unter Verwendung von Java-Message-Services (JMS) sowie .NET- und WebSphere-Anwendungen. Unter Verwendung von Message-Queuing (MQ)-Gateways wird auf in Dienste gekapselte Altanwendungen zugegriffen. Auf unternehmensweite Anwendungssysteme wie ERP und CRM wird über Adapter zugegriffen. Adapter dienen dazu, die Application-Programming-Interfaces (APIs) der Anwendungssysteme und die Infrastruktur zum Nachrichtenaustausch zu verbinden.

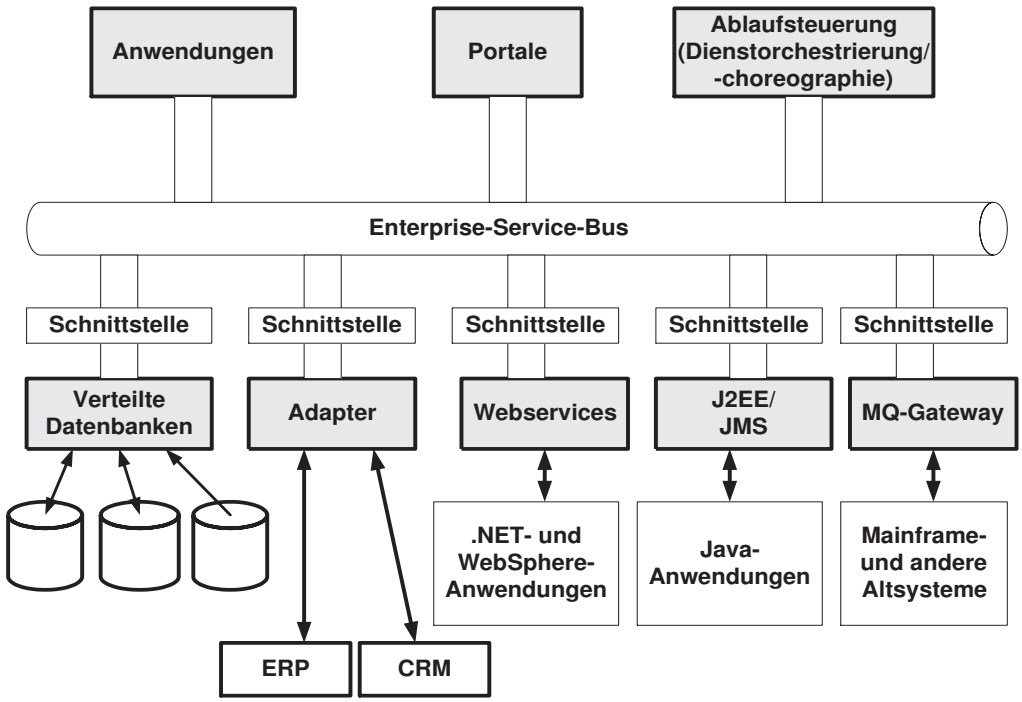

Abbildung 2.19: ESB zur Verbindung von Anwendungen (in Anlehnung an [42])

2.3.4.5 Nutzen service-orientierter Architekturen

SOA-Vorteile

Die Verwendung des SOA-Konzeptes verspricht die folgenden Vorteile:

- SOA ermöglicht es, neue oder geänderte Geschäftsprozesse ohne großen technischen Aufwand durch Neukombination bereits bestehender Dienste

im Rahmen von Orchestrierung und Choreographie schnell zu realisieren. Dies bewirkt eine gesteigerte Flexibilität der Unternehmen bezüglich neuer Anforderungen der Unternehmensplan- bzw. Geschäftsprozessebene. Entscheidend ist dabei, dass die Ablaufsteuerung in einer separaten Workflowkomponente ausgelagert ist.

- Durch die Kapselung von Implementierungsdetails durch wohldefinierte Schnittstellen wird eine Verringerung der Komplexität unternehmensweiter Informationssysteme und somit eine verbesserte Beherrschbarkeit erreicht.

- Durch die Wiederverwendbarkeit bestehender fachlicher Komponenten und durch die Integration über eine einheitliche service-orientierte Architektur können mittelfristig Kosten bei der Entwicklung und Wartung der Informationssysteme reduziert werden. Dabei wird ausgenutzt, dass die Infrastruktur der SOA bereits vorhanden ist und stets auf die gleiche Art und Weise verwendet wird. Durch die angestrebte lose Kopplung von Systemen in einer SOA sind auch weitestgehend entkoppelte Entwicklungsarbeiten möglich. Durch die schrittweise Freigabe neuer Dienste wird eine „weiche“ Einführung („Soft rollout“) ermöglicht, indem man bestimmte Dienste anfänglich nur leer zusätzlich mitlaufen lässt.

- Die SOA bietet einen hohen Investitionsschutz, da im Rahmen dieses Konzeptes erprobte Altsysteme weiter betrieben werden können, ohne dass die neuen Systeme von technologischen Altlasten abhängig sind.

- Eine evolutionäre Entwicklung der Anwendungslandschaft ist möglich, da eine SOA es gestattet, eine bestehende, gewachsene Systemlandschaft fachlich aufzuteilen und so in softwaretechnisch erneuerbare Bestandteile zu zerlegen, so dass eine sukzessive Ablösung auch von monolithischer und historisch gewachsener Software möglich ist.

- Durch die Standardisierung von Diensten wird das Outsourcing einzelner Geschäftsprozesse sowie die Dekomposition von Geschäftsprozessen erleichtert.

- Anwendungen, die auf SOA basieren, sind besonders gut für das Dynamic-Computing geeignet. Beim Dynamic-Computing werden Rechenleistung und Basissoftware als eine Ware betrachtet, die in beliebiger Menge zur Verfügung steht. Leistungsfähige, aber gleichzeitig preiswerte Rechner ohne Kühlung und Netzteil, die in goßer Anzahl in Rahmen eingesteckt werden, werden als Blades bezeichnet. Jedes Blade ist Träger eines Betriebssystems. SOA ist gut für Dynamic-Computing geeignet, da rechenintensive Dienste in vielen Instanzen beim ESB registriert werden können. Jedem Dienst wird dann ein eigenes Betriebssystemexemplar zugeordnet.

Das SOA-Konzept stellt somit ein Modellierungskonzept für betriebliche Informationssysteme dar, mit dem dienstorientierte Informationssysteme implementiert werden können.

Architekturrahmen SOA

Der service-orientierte Architekturrahmen wird in sieben Modellebenen untergliedert, die in Abbildung 2.20 dargestellt sind [3].

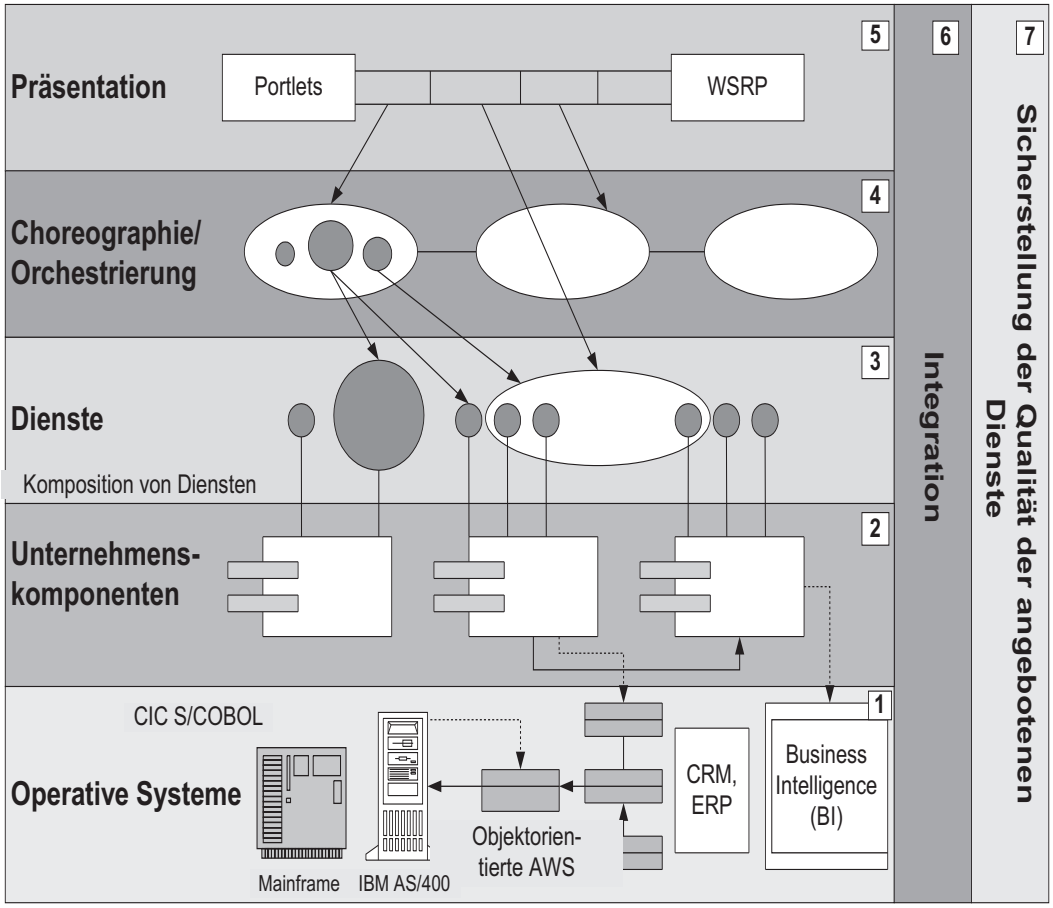

Abbildung 2.20: Modellebenen der SOA nach [3]

Modellebenen

Wir erläutern der Reihe nach die unterschiedlichen Modellebenen.

1. **Ebene der operativen Systeme**: Diese Modellebene umfasst existierende Enterprise-Resource-Planning- und Customer-Relationship-Management-Softwarepakete, ältere objektorientierte Softwaresysteme, Business-Intelligence-Anwendungen, unterschiedliche Datenhaltungssysteme sowie andere Altsysteme. Die Modellebene ermöglicht es, bestehende Anwendungssysteme eines Unternehmens in eine SOA zu integrieren. Im Sinne des generischen Architekturrahmens von Sinz [53], der in Abschnitt 2.2 dieses Kurses behandelt wurde, wird auf dieser Ebene eine Daten- und eine Außensicht des betrieblichen Informationssystems spezifiziert.

2. **Ebene der Unternehmenskomponenten**: Auf dieser Ebene wird die Hard- und Softwareinfrastruktur für die einzelnen Fachabteilungen eines Unternehmens spezifiziert. Diese Architekturebene verwendet typischerweise Technologien wie Applikationsserver für die Implementierung der betrieblichen Anwendungssoftware sowie für das Management der Lastverteilung auf den unterschiedlichen Servern. Durch diese Ebene wird im Sinne des generischen Architekturrahmens eine funktionale Sicht spezifiziert.

3. **Ebene der Dienste**: Die Dienste, die ein Unternehmen für seine SOA auswählt, sind in dieser Ebene angeordnet. Die Dienste können in der Dienste-Registry aufgefunden oder statisch eingebunden und dann aufgerufen oder zu einem zusammengesetzten Dienst angeordnet werden. Diese dienstveröffentlichende Schicht stellt auch Schnittstellen in Form von Dienstbeschreibungen zur Verfügung. Dadurch können Unternehmensbestandteile Dienstrealisierungen zur Laufzeit zur Verfügung stellen und Dienste nutzen, deren Funktionalität durch die Dienstschnittstelle bereitgestellt wird. Wie bereits auf der Ebene der Unternehmenskomponenten wird auch hier ebenfalls eine funktionale Sicht spezifiziert.

4. **Choreographie- und Orchestrierungsebene**: In dieser Ebene wird festgelegt, in welcher Reihenfolge unter Beachtung welcher Abhängigkeiten die Dienste genutzt werden können, um die gewünschte Funktionalität bereitzustellen. Ein Prozess kann dabei aus mehreren sequentiell oder parallel ablaufenden Diensten bestehen, wobei die Abfolge der einzelnen Dienste genau festgelegt sein muss. Auf dieser Ebene wird die Sicht der Geschäftsprozesse der Organisation spezifiziert.

5. **Zugriffs- und Präsentationsebene**: Diese Ebene wird benötigt, um zukünftige Lösungskonzepte zu berücksichtigen. Es ist hier wichtig zu wissen, dass eine SOA die Benutzerschnittstelle der einzelnen Komponenten entkoppelt und die Bereitstellung einer “end-to-end“-Lösung zwischen einem Zugriffskanal und einem Dienst oder einer Anordnung von Diensten benötigt wird. Die Ebene stellt diese Lösung zur Verfügung und ermöglicht den Zugriff auf einzelne Dienste oder auf einen Prozess aus komponierten Diensten. In Abbildung 2.20 wird gezeigt, dass auf Portale über Portlets und Webservices-for-Remote-Portlets (WSRP) zugegriffen werden kann. Portlets sind Bestandteile der Benutzeroberfläche eines Portals, die vom Portalserver angezeigt und verwaltet werden. Im Sinne des generischem Architekturrahmens wird auf dieser Ebene die Präsentationssicht beschrieben.

6. **Integrationsebene**: Diese Ebene ermöglicht die Integration von Diensten durch die Einführung von zuverlässigen Funktionen wie intelligentes Routing, protokollare Vermittlung und andere Übersetzungsmechanismen, die

üblicherweise als ESB-Funktionalität beschrieben werden. WSDL spezifiziert eine Bindung der Dienste, die den Ort einschließt, an dem der Dienst zur Verfügung gestellt wird. Andererseits stellt der ESB einen vom Ort der Bereitstellung des Dienstes unabhängigen Mechanismus für die Integration der Dienste zur Verfügung. Eine Spezifikation der Integrationssicht findet auf dieser Ebene statt.

7. **Ebene zur Sicherstellung der Qualität der angebotenen Dienste**: Diese Ebene ist für die Überwachung, Steuerung und Aufrechterhaltung der Qualität der Dienste sowie für deren Sicherheit und Nutzbarkeit verantwortlich. Die Überwachung, Steuerung und Aufrechterhaltung der QoS wird durch einen Hintergrundprozess in Form der Erfassung von Signalen und Reaktion auf diese Signale realisiert. Werkzeuge werden zur Verfügung gestellt, die den Zustand der SOA-Anwendungen überwachen und somit die Qualität der Dienste gewährleisten. Auf dieser Ebene wird die Sicht der Qualitätssicherung spezifiziert.

Das Ziel bei der Modellierung einer service-orientierten Architektur besteht darin, Dienste zu identifizieren, zu spezifizieren und zu realisieren, die global im gesamten Unternehmen zur Verfügung stehen.

Im Abschnitt 4.3 zeigen wir, wie eine SOA unter Verwendung der Webservicetechnologie realisiert werden kann. In Abschnitt 7.2 gehen wir im Rahmen einer Fallstudie auf die prototypische Implementierung einer SOA für die Produktionsdomäne ein.

Übungsaufgabe 2.7 (Generischer Architekturrahmen) *Wenden Sie den generischen Architekturrahmen aus Abschnitt 2.2 auf die Architekturbeispiele*

a) SOM,

b) ARIS,

c) SOA

an. Identifizieren Sie dabei die jeweiligen Modellierungskonzepte, die Modellebenen, die zugehörigen Sichten und mögliche Metamodelle.

2.4 Anwendung von Architekturen

Nutzung von Architekturen

In diesem Abschnitt beschreiben wir, wie Architekturen von Informationssystemen im Unternehmen genutzt werden können. Wir gehen zunächst auf die Nutzung der Architekturen aus Managementsicht ein. Dabei sind drei Anwendungsfelder zu unterscheiden [53]:

1. Gestaltung der Organisation,

2. Informationsmanagement,

3. Qualitätsmanagement.

Gestaltung der Organisation

In den letzten Jahren ist ein Trend hin zu flexiblen Organisationsformen erkennbar. Diese Entwicklung ist durch eine Aufweichung der Aufbauorganisation sowie durch einen Wechsel von funktionsorientierten Organisationsformen zu dezentralen und prozessorientierten Organisationsformen gekennzeichnet [60]. Den veränderten Organisationsformen wird durch entsprechende Informationssystemarchitekturen Rechnung getragen. Als vorteilhaft hat sich im Zusammenhang mit der **Gestaltung der Organisation** die explizite Trennung zwischen Aufgaben- und Aufgabenträgerebene erwiesen. Die Aufgabenebene dient der Spezifikation der Leistungserstellung eines Unternehmens sowie deren Lenkung. Die Aufgabenträgerebene hingegen beschreibt die personellen und maschinellen Ressourcen zur Durchführung der Aufgaben. Die Ressourcen können dabei flexibel den Aufgaben zugeordnet werden. In diesem Zusammenhang wird auf die in Abschnitt 5.3.7.4 beschriebene anzustrebende Isomorphie zwischen Organisationsstruktur und Informationssystemarchitektur verwiesen.

Informationsmanagement

Die Aufgabe des **Informationsmanagements** besteht darin, das Management eines Unternehmens bei der Gestaltung des unternehmensweiten Informationssystems, bei der Lenkung des Betriebs der IT-Infrastruktur und der integrierten Anwendungssysteme in enger Abstimmung mit den Fachabteilungen zu unterstützen. Die Informationssystemarchitektur stellt den Gesamtplan für das unternehmensweite Informationssystem sowie die dazugehörigen Anwendungssysteme mit einem hohen Abstraktionsgrad sowie verschiedene detaillierte Pläne für unterschiedliche Subsysteme zur Verfügung.

Qualitätsmanagement

Informationssystemarchitekturen unterstützen das **Qualitätsmanagement** eines Unternehmens, da die Informationssystemarchitektur die Geschäftsprozesse eines Unternehmens, deren Lenkung sowie die personellen und maschinellen Ressourcen, die zur Durchführung der Geschäftsprozesse benötigt werden, dokumentiert. Die Qualitätssicherung kann auf dieser Dokumentation aufbauen.

2.5 Qualitätsaspekte für Informationssystemarchitekturen

An Informationssystemarchitekturen sind Qualitätsanforderungen zu stellen. Ein wichtiges Qualitätsmerkmal von Informationssystemarchitekturen ist deren Flexibilität. Flexibilität besitzt in diesem Zusammenhang insbesondere die wesentlichen Ausprägungen Änder- und Erweiterbarkeit [5]. Ein betriebliches Informationssystem muss im Laufe der Zeit an neue Anforderungen und Rahmenbedingungen angepasst werden. Für die Bewertung der Flexibilität eines Informationssystems sind die folgenden Kriterien heranzuziehen:

Kriterien für Flexibilität

1. **Änderbarkeit**: Ein Informationssystem heißt änderbar, wenn es an veränderte Unternehmenspläne und Geschäftsprozesse angepasst werden kann. Veränderungen von Unternehmensplänen und Geschäftsprozessen werden durch Konkurrenten am Markt oder durch den Einfluss des Gesetzgebers verursacht. Die Grenzen zwischen Änderung und Wartung sind häufig fließend. Eine Änderbarkeit von Informationssystemen setzt voraus, dass die Modellsysteme der Informationssystemarchitektur über alle Modellebenen hinweg abgestimmt sind. Eine Änderung von Geschäftsprozessen erzwingt häufig eine Änderung der Anwendungssysteme eines Unternehmens. Die Anwendungssysteme können aber nur dann mit vertretbarem Aufwand geändert werden, wenn die Geschäftsprozessmodelle und die Spezifikation der Anwendungssysteme aufeinander abgestimmt sind. Das wird beispielsweise durch SOM geleistet.

2. **Wiederverwendbarkeit**: Eine Informationssystemarchitektur wird als wiederverwendbar bezeichnet, wenn Teile des Modellsystems mehrfach eingesetzt werden können. Die Zielsetzung der Wiederverwendung hängt dabei von der betrachteten Modellebene der Architektur ab.
   Eine Unterstützung der fachlichen Analyse bzw. eine normative Gestaltungsvorgabe wird durch Wiederverwendung von Bestandteilen eines Fachkonzepts auf der Fachkonzeptebene erreicht. Wir betrachten dazu das folgende Beispiel.

   Beispiel 2.5.1 (Wiederverwendung von Fachkonzepten) *Fachliche Klassenhierarchien für bestimmte Anwendungsdomänen unterstützen die Wiederverwendung. Ein derartiges fachlich motiviertes Objektmodell ist die PROSA-Referenzarchitektur (siehe [58] und die Ausführungen in Abschnitt 2.2.3). Weitere konkrete Beispiele sind durch die Enterprise-Ontologie [57] und die OZONE-Ontologie [54] gegeben.*

   Eine Wiederverwendung von Softwarekomponenten wird auf der Softwarearchitekturebene angestrebt. Beispiele für diese Wiederverwendungsart sind durch objektorientierte Rahmenwerke für konkrete Anwendungsdomänen gegeben (vergleiche die Ausführungen in Abschnitt 5.4 dieses Kurses).
   Systemplattformen hingegen stellen spezielle Entwicklungsobjekte für einen breiten Anwendungsbereich zur Verfügung [53]. Das nachfolgende Beispiel für Softwareplattformen wird angegeben.

   Beispiel 2.5.2 (Softwareplattformen) *Middleware wie der Common-Object-Request-Broker (CORBA) [43] unterstützt die Wiederverwendung von Softwarekomponenten. Middlewarekonzepte werden in Abschnitt 4.2 dieses Kurses dargestellt.*

3. **Testbarkeit und Integrierbarkeit**: Informationssysteme sind aus unterschiedlichen Komponenten aufgebaut. Ein Informationssystem heißt testbar, wenn seine Komponenten separat getestet werden können, eine systematische Identifizierung der Testfälle möglich ist, die Tests auf Basis der Testfälle mehrfach durchgeführt werden können und die Testergebnisse beobachtet werden können [29]. Die Komponenten eines testbaren Informationssystems werden somit zunächst getrennt getestet. Im zweiten Schritt werden größere Systemeinheiten aus den bereits getesteten Komponenten zusammengesetzt. Dieser Schritt wird als Integration bezeichnet. Ein Informationssystem wird als integrierbar bezeichnet, wenn es aus unterschiedlichen Teilsystemen zusammengesetzt werden kann.

4. **Skalierbarkeit**: Unter Skalierbarkeit versteht man die Fähigkeit eines Informationssystems, ohne nennenswerte Laufzeiteinbußen eine stärkere Belastung zu verkraften. Wir betrachten dazu das folgende Beispiel.

   Beispiel 2.5.3 (Belastung) *Eine höhere Anzahl von Transaktionen pro Sekunde, eine größere Anzahl gleichzeitig angemeldeter Benutzer oder eine Vervielfältigung des Datenvolumens stellen eine größere Belastung dar.*

   Skalierung wird stets nur durch gemeinsame Maßnahmen auf Hardware-, Basissoftware- und Anwendungssoftwareebene erreicht. Eine gute Architektur ist dadurch gekennzeichnet, dass Skalierbarkeit ohne starke Eingriffe in die Anwendungssoftware möglich ist. Das ist aber nur möglich, wenn die Abhängigkeiten der Anwendung von der Basissoftware minimal und exakt definiert sind [5]. Das kann beispielsweise durch mehrstufige Client-Server-Architekturen erreicht werden (vergleiche dazu die Ausführungen in Abschnitt 2.6.2).

5. **Portierbarkeit**: Unter Portierbarkeit verstehen wir die Fähigkeit eines Informationssystems, Veränderungen der Basissoftware (Betriebssystem, Datenbanksystem, Transaktionsmonitor, Middleware) zu verkraften, ohne dass man es neu entwickeln muss [5]. Eine gute Architektur beschreibt exakt die Abhängigkeiten der Anwendung von der Basissoftware und versucht, diese durch geeignete Zwischenschichten der Software zu verringern.

6. **Wartbarkeit**: Jedes Informationssystem weist Fehler auf. Ein Informationssystem wird wartbar genannt, wenn man Fehler mit vertretbarem Aufwand lokal reparieren kann, ohne dass neue Fehler an unerwarteten Stellen infolge der Fehlerkorrektur auftreten [5]. Eine durchdachte Komponentenstruktur als Bestandteil einer Architektur unterstützt eine Wartbarkeit des Informationssystems.

Eigenschaften für eine gute Architektur

Zusammenfassend ist festzustellen, dass ein Informationssystem mit diesen Eigenschaften eine gute Architektur besitzt. Die eben dargestellten Kriterien sind weich, da sie nicht quantitativ messbar sind.

Wir beschreiben nun eine Reihe von Eigenschaften eines Informationssystems, die indirekt Maße für die Qualität einer Informationssystemarchitektur darstellen. Wir bezeichnen in Anlehnung an [5] eine Architektur als gut, wenn die im Folgenden aufgezählten Eigenschaften durch lokale Reparatur oder Ergänzungen im Rahmen der vorhandenen Struktur mit vernünftigem Aufwand hergestellt werden können. Die meßbaren Kriterien werden nun im Einzelnen diskutiert.

1. **Performanz**: Die Performanz eines Informationssystems wird durch die geforderten Antwortzeiten im Dialog mit dem Benutzer sowie durch die Transaktionshäufigkeit bestimmt. Eine Transaktion, die nur selten gestartet wird, kann mehr Zeit zur Ausführung benötigen. Neben den Antwortzeiten im Dialog bestimmen auch die Laufzeiten der Batches im Batchbetrieb die Performanz des Informationssystems. Eine gute Architektur zeichnet sich dadurch aus, dass wichtige Transaktionen mit teilweise deutlich über 100000 Transaktionen pro Tag und die wichtigen Batches auf Kosten der weniger wichtigen und weniger zeitkritischen Funktionalität beschleunigt werden [5].

2. **Verfügbarkeit und Zuverlässigkeit**: Die Verfügbarkeit eines Informationssystems und allgemeiner die eines technischen Systems wird als Quotient aus der tatsächlichen und der zugesicherten Verfügbarkeit ermittelt. Die mittlere Zeit zwischen zwei Ausfällen („Mean Time to Failure (MTF)“) dient als Maß für die Zuverlässigkeit.

3. **Robustheit**: Unter Robustheit eines Informationssystems versteht man dessen Fähigkeit, falsche Benutzereingaben bzw. fehlerhafte Daten von Nachbarsystemen geeignet zu behandeln. Ein robustes Informationssystem erkennt diese fehlerhaften Konstellationen und beendet seine Arbeit korrekt. Das ist insbesondere auch bei der Versorgung von Nachbarsystemen im Rahmen von Kopplungsarchitekturen (vergleiche hierzu die Ausführungen in Abschnitt 3.5 dieses Kurses) von Bedeutung. Die Robustheit eines Informationssystems wird wesentlich durch die Gestaltung der Innen- und Außensicht (vergleiche Abschnitt 2.2.1) beeinflusst.

4. **Funktionsumfang**: Bei diesem Qualitätsmerkmal wird gemessen, ob das Informationssystem die gewünschte Funktionalität tatsächlich zur Verfügung stellt. Durch ein unter dem Aufgabenblickwinkel entwickeltes Fachkonzept wird das Erreichen der notwendigen Funktionalität wesentlich unterstützt.

5. **Benutzbarkeit**: Durch eine entsprechende Gestaltung der Benutzerschnittstelle des Informationssystems wird die Benutzbarkeit eines Infor-

mationssystems sichergestellt. Unter architektonischen Gesichtspunkten ist davon die Außendarstellung des Informationssystems betroffen.

6. **Sicherheit**: Ein Informationssystem wird als sicher bezeichnet, wenn es einen Missbrauch auf jeder Ebene verhindert. Unter Missbrauch wird dabei ein unautorisierter Zugriff verstanden. Für die Benutzerschnittstelle ist ein Berechtigungskonzept vorzusehen, auf Ebene der Kommunikation zwischen Rechnern sind Maßnahmen zur Verschlüsselung und Firewalls notwendig. Davon ist wieder die Außen- und Innensicht des Informationssystems betroffen.

In den Kurseinheiten 3 und 4 dieses Kurses wird an verschiedenen Stellen darauf eingegangen, wie Informationssysteme, die den Qualitätskriterien dieses Abschnitts genügen, konstruiert werden können. In Kapitel 5 wird dargestellt, dass die hier diskutierten Qualitätsmerkmale auch bei der Auswahl betriebswirtschaftlicher Standardsoftware eine große Rolle spielen.

2.6 Softwarearchitekturen betrieblicher Anwendungssysteme

In diesem Abschnitt konkretisieren und spezialisieren wir die Aussagen der bisherigen Abschnitte dieses Kapitels. Wir betrachten ausschließlich Softwarearchitekturen für betriebliche Anwendungssysteme. Nach der Diskussion früherer monolithischer Anwendungssysteme stellen wir die in der Praxis gegenwärtig weit verbreiteten mehrstufigen Client-Server-Architekturen für betriebliche Anwendungssysteme dar. Anschließend werden zukünftige komponentenbasierte Anwendungssystemarchitekturen vorgestellt.

2.6.1 Monolithische Anwendungssysteme

Monolithen

Betriebliche Anwendungssysteme wurden in der Vergangenheit als monolithische Systeme auf zentralen Großrechnern (Mainframes) betrieben. An Mainframes wurden Terminals angeschlossen, die es Benutzern erlaubten, mit dem Anwendungssystem zu interagieren. Das Anwendungssystem verwendete dabei ein proprietäres Betriebssystem.

Monolithische Anwendungssysteme sind dadurch gekennzeichnet, dass die einzelnen Systembestandteile eine untrennbare Einheit bilden. Insbesondere sind auch Anwendungslogik und anwendungsübergreifende Datenhaltung nicht sauber getrennt. Das erschwert und behindert die Wartung, Weiterentwicklung und Integrierbarkeit derartiger Systeme.

Großrechnersysteme findet man heute nur noch bei großen Unternehmen wie Banken und Versicherungen oder Dienstleistern vor. Der Einsatz von Großrechnern wurde im Vergleich zu Workstation- oder PC-basierten Lösungen zunehmend unwirtschaftlich. In der Konsequenz führte das zum Einsatz von auf dem

Client-Server-Prinzip basierenden betrieblichen Anwendungssystemen. Diese Architekturen unterstützten insbesondere auch Dezentralisierungsbestrebungen für die Informationssystemarchitektur.

2.6.2 Mehrstufige Client-Server-Architekturen

Schichten

Betriebliche Anwendungssysteme bestehen prinzipiell aus den drei nachfolgenden Systembestandteilen [5, 23, 2, 26]:

1. **Präsentationsschicht**: Diese Schicht dient der Interaktion zwischen dem Anwendungssystem und den Benutzern.

   Beispiel 2.6.1 (Präsentationsschicht) *Eine Stückliste für ein bestimmtes Produkt kann auf dem Bildschirm angezeigt werden. Dem Benutzer wird es ermöglicht, über die Benutzeroberfläche Auftragsdaten in das Anwendungssystem zu übertragen.*

2. **Anwendungsschicht**: Die Anwendungsschicht wird auch als Verarbeitungsschicht bezeichnet. In dieser Schicht ist die Verarbeitungslogik des Anwendungssystems implementiert. Sie bildet somit den Kern des betrieblichen Anwendungssystems.

   Beispiel 2.6.2 (Anwendungsschicht) *Fertigungsaufträge in der Produktion werden durch diese Schicht terminiert. Es wird automatisch ein Kapazitätsabgleich durchgeführt.*

3. **Datenhaltungsschicht**: Dieser Bestandteil des Anwendungssystems dient der Verwaltung der Daten des Anwendungssystems.

   Beispiel 2.6.3 *Auftragsdaten, Arbeitspläne sowie Stücklisten werden in der Datenhaltungsschicht in relationalen Datenbanken vorgehalten.*

C/S-Architektur

Die eben beschriebene Dreischichtenarchitektur ermöglicht zum einen das Aufbrechen der monolithischen Architektur. Andererseits kann das Anwendungssystem auf mehrere, miteinander vernetzte Rechner verteilt werden. Die Verarbeitung wird nach dem Client-Server-Prinzip zwischen den unterschiedlichen Rechnern aufgeteilt. Dienstnehmer (Clients) fragen bei einem Diensterbringer (Server) nach der Erbringung eines bestimmten Dienstes nach. Dabei wird der Interaktion zwischen Client und Server ein bestimmtes Protokoll zugrundegelegt. Man spricht von Client- und von Server-Prozessen im Sinne von verteilten Systemen, um zu verdeutlichen, dass sowohl Client als auch Server in Ausführung befindliche Programme sind, die auf einem der virtuellen Prozessoren des Betriebssystems ausgeführt werden [56].

Ein Server bedient typischerweise mehrere Clients. Server können prinzipiell

auch als Clients wirken, d.h., ein Server kann in der Rolle als Client bei anderen Servern bestimmte Informationen nachfragen. Wir betrachten dazu das nachfolgende Beispiel.

Beispiel 2.6.4 (Client-Server-Architekturen) *In Abbildung 2.21 ist ein Beispiel angegeben, in dem ein Client-Prozess CP_1 Anfragen an einen Prozess schickt (1), der gleichzeitig als Server und als Client fungiert. Dieser Prozess wird mit SCP_1 bezeichnet. Prozess SCP_1 kann die Anfrage von Prozess CP_1 nicht alleine bearbeiten und fragt deshalb beim Server-Prozess SP_1 an (2). SP_1 teilt Prozess SCP_1 mit, dass er die Anfrage auch nicht beantworten kann (3). SCP_1 fragt deshalb zusätzlich bei dem weiteren Server-Prozess SP_2 an (4). Dieser Prozess ist in der Lage, die Anfrage zu beantworten. Nach dem Erhalt dieser Antwort (5) ist jetzt SCP_1 befähigt, das Ergebnis der Anfrage an den Prozess CP_1 zu übermitteln (6).*

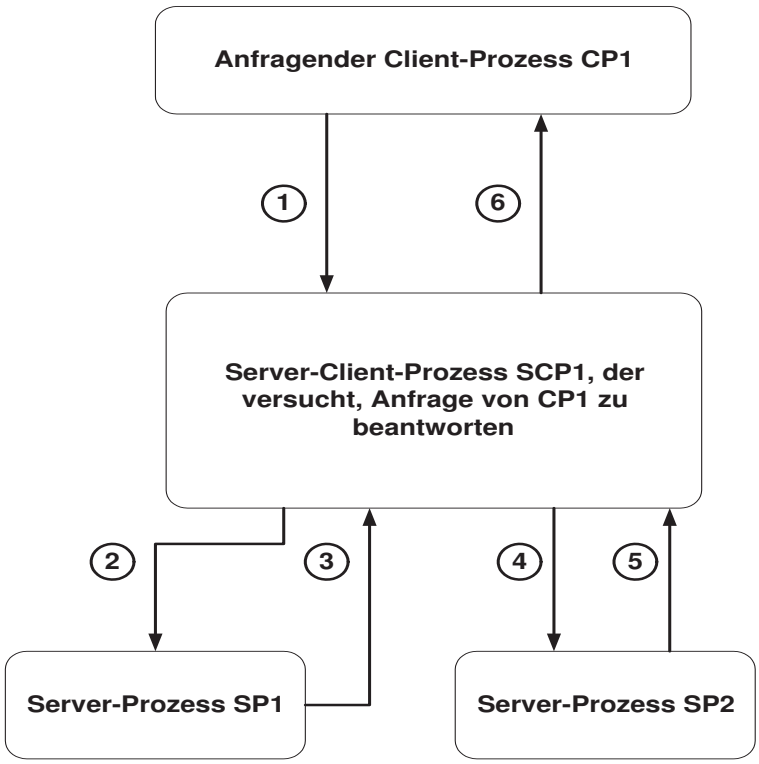

Abbildung 2.21: Interaktion von mehreren Client- und Serverprozessen

Client- und Server-Prozesse können mit gewissen Einschränkungen beliebig unterschiedlichen Rechnersystemen zugeordnet werden. Sowohl Client- als auch Server-Prozesse können auf einem oder auch verteilt auf mehreren Rechnern laufen. Das Loadbalancing (Lastverteilung) spielt eine wichtige Rolle bei der

Zuordnung von Prozessen zu Rechnern. Es ordnet parallel abzuarbeitende Prozesse bzw. Prozess-Schritte geeigneten Prozessoren zu. Dabei wird das Ziel verfolgt, die Prozesse möglichst effizient abzuarbeiten. Ein Prozess-Schritt kann eine Benutzersitzung, eine Transaktion oder die Verarbeitung einzelner Benutzereingaben darstellen [2].

Client-Server-Strukturen unterstützen die in Abschnitt 2.5 geforderte Skalierbarkeit von Anwendungssystemen. Falls die Hardware-Ressourcen nicht mehr ausreichen, kann die entsprechende Schicht auf mehrere Server verteilt werden.

Dezentralisierung

Client-Server-Architekturen entsprechen den organisatorischen Dezentralisierungsbestrebungen. Sie stellen somit ein Beispiel für die strukturelle Analogie zwischen Organisation und Informationssystemarchitektur dar, die in Abschnitt 5.3.7.4 eingeführt wird.

Verschiedene Formen der Dezentralisierung sind möglich [26]. Zunächst ist es sinnvoll, die Datenhaltung auszulagern. Außerdem können sowohl Datenhaltung als auch die Präsentationsschicht ausgelagert werden. Bei diesen beiden Dezentralisierungsstrategien steht das Herauslösen von einzelnen Systembestandteilen aus dem monolithischen Gesamtsystem im Vordergrund, man spricht in diesem Zusammenhang auch von einer Trennung der Zuständigkeiten („separation of concern“). Der Weg vom monolithischen Gesamtsystem zur Client-Server-Architektur, in der eine Trennung nach Zuständigkeiten realisiert ist, wird in Abbildung 2.22 gezeigt.

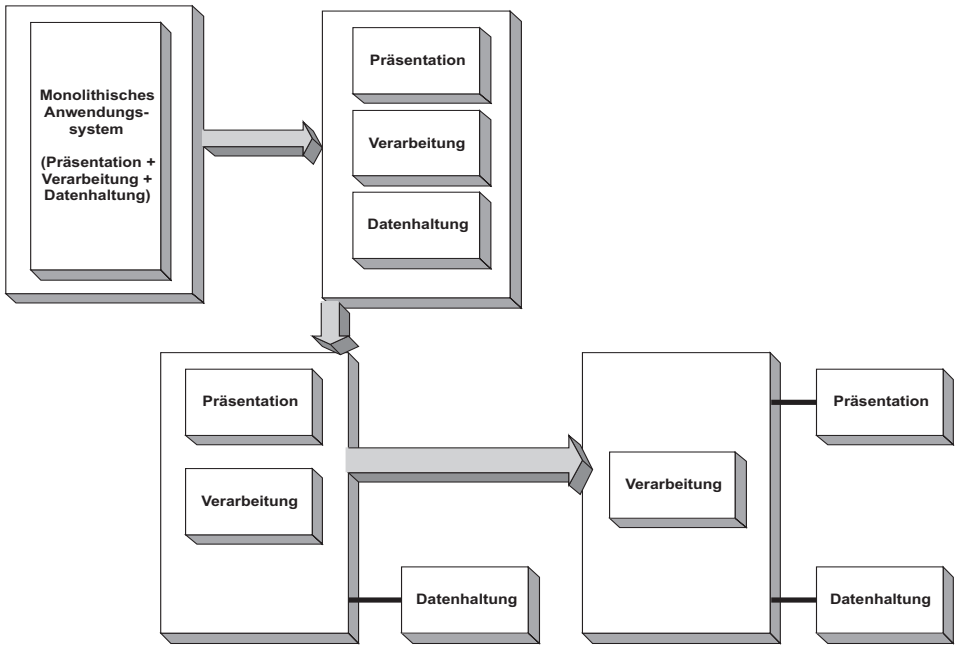

Abbildung 2.22: Weg vom Monolithen zur Client-Serverarchitektur [26]

Die beschriebenen Dezentralisierungsstrategien können weiter verfeinert wer-

den, indem verschiedene Formen der physischen Verteilung sowie der Replikation der herausgelösten Systembestandteile betrachtet werden. Für das Beispiel des Herauslösens der Datenhaltung aus dem monolithischen System bedeutet das, dass eine verteilte Datenhaltung möglich ist, bei der die Datenhaltung auf mehrere Rechner ausgelagert ist. Im Gegensatz dazu spricht man von einer entfernten Datenhaltung, wenn genau ein Server mit dieser Aufgabe betraut wird. Wir sprechen von verteilter Präsentation, wenn der Systembestandteil Präsentation repliziert auf den individuellen Rechnern der einzelnen Nutzer vorliegt.

Präsentation

Dreistufige Client-Server-Architekturen für betriebswirtschaftliche Standardsoftwaresysteme verteilen und replizieren damit typischerweise die Präsentationskomponente auf jeden einzelnen Anwendungsrechner. Dazu ist auf jedem dieser Rechner ein separates Programm zu installieren, das dem einzelnen Benutzer eine Dialogschnittstelle zum eigentlichen Anwendungssystem zur Verfügung stellt. Wir geben dazu das folgende Beispiel an.

Beispiel 2.6.5 (Replikation der Präsentationskomponente) *Das ERP-System SAP R/3 erfordert typischerweise die Installation des Programms SAP GUI für Windows bzw. SAP GUI für Java direkt auf dem Computer des SAP R/3-Nutzers oder in einem Rechnernetz.*

Datenhaltung

Der Systembestandteil Datenhaltung wird auf einem eigenen Server (einem sogenannten Datenbankserver) installiert. Der Systembestandteil Verarbeitung enthält zum einen die eigentliche Verarbeitungslogik, daneben wird auch Koordinationsfunktionalität für die unterschiedlichen Bestandteile der Anwendungsschicht bereitgestellt.

Anwendungslogik

Die Anwendungsschicht verteilt keine untergeordneten Systembestandteile, sondern organisiert die Anwendungsschicht als separate Betriebssystemprozesse, denen ein gemeinsamer Speicherbereich zur Verfügung steht. Durch einen Dispatcher werden den einzelnen Prozessen bestimmte Aufgaben zugewiesen. Der Dispatcher fungiert als zentrale Koordinationsinstanz. Zur Beschleunigung der Anwendung können die einzelnen Prozesse auf unterschiedliche Rechner verteilt werden.

C/S-Architekturformen

Heutige Client-Server-Architekturen sind überwiegend technisch motiviert [2]. Sie weisen typischerweise eine Baumstruktur auf. Clients zur Interaktion mit dem Benutzern bilden die Blätter des Baumes. Einer oder gegebenenfalls mehrere Anwendungsserver stellen den Clients Dienste zur Verfügung. Die Anwendungsserver verhalten sich gegenüber einem zentralen Datenbankserver, der die Wurzel des Baumes darstellt, wie Clients. In Abhängigkeit von der Struktur des Baumes können die nachfolgenden drei Ausprägungsstufen unterschieden werden:

1. Es existiert genau ein Anwendungsserver, der sich mit dem Datenbankserver auf einer gemeinsamen Maschine befindet. Prinzipiell sind mehrere Präsentationsrechner möglich, auf denen wiederum mehrere Clients für Präsentationszwecke ablaufen.

2. Bei der zweiten Ausgestaltungsform sind Anwendungs- und Datenbankserver auf unterschiedlichen Rechnersystemen installiert. Ansonsten unterscheidet sich diese Ausgestaltungsform nicht von der ersten.

3. In der letzten Ausgestaltungsform werden mehrere Anwendungsserver auf jeweils eigenen Maschinen betrieben. Die unterschiedlichen Anwendungsserver greifen auf genau einen Datenbankserver zu. Mehrere Präsentationsrechner sind möglich, auf denen ähnlich wie in der ersten und zweiten Ausgestaltungsform unterschiedliche Präsentationsclients laufen. Diese Ausgestaltungsform führt zu horizontal gegliederten, fachlich motivierten Architekturen, in denen eigenständige betriebliche Aufgaben auf unterschiedliche Rechner verteilt werden.

Die drei behandelten Ausgestaltungsformen sind in Abbildung 2.23 dargestellt.

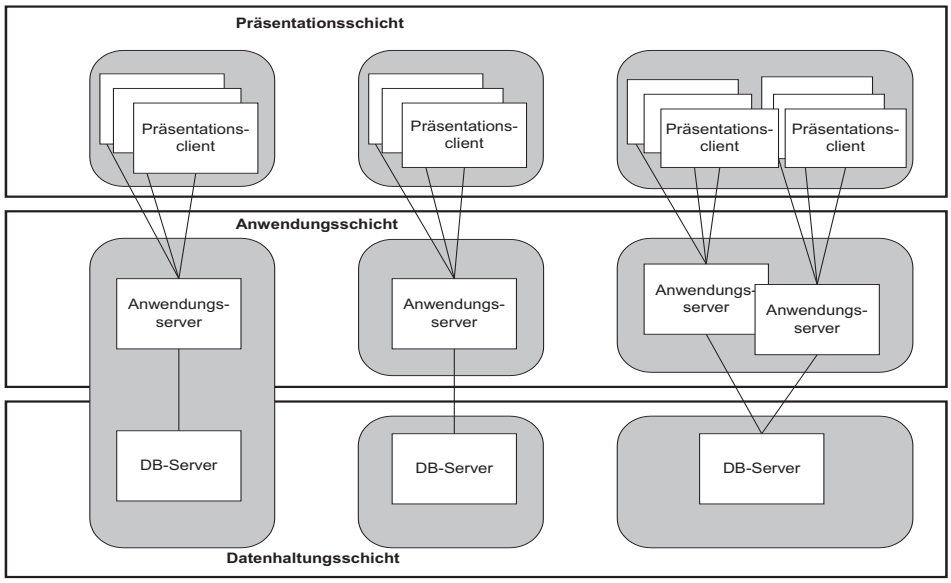

Abbildung 2.23: Ausgestaltungsvarianten von dreistufige Architekturen [2]

Neben den bereits diskutierten Ausgestaltungsformen existieren weitere Möglichkeiten. Eine weitere Ausgestaltungsform wird exemplarisch in Beispiel 2.6.6 diskutiert.

Beispiel 2.6.6 (Ausgestaltung von Client-Server-Architekturen) *Wir betrachten Szenarien, in denen neben mehreren Anwendungsservern auch eine Reihe von Datenbankservern Verwendung findet oder in denen den unterschiedlichen Anwendungsservern jeweils eine eigene Datenbank zugeordnet ist. In diesem Fall kann die Bewahrung der Atomicy-Consistency-Isolation-Durability (ACID)-Eigenschaft für Folgen von Lese- und Schreiboperationen (Datenbanktransaktionen) nicht mehr ohne weiteres sichergestellt werden.*

Vierstufige Architektur

Neben unternehmensweiten Anwendungssystemen ist es häufig notwendig, Anwendungssysteme unternehmensübergreifend zu integrieren. Diese Situation ist beispielsweise für das Lieferkettenmanagement typisch [30, 31]. Da Daten in den einzelnen Unternehmen örtlich getrennt anfallen, ist der Betrieb eines einzelnen, zentralen Datenbankservers in dieser Situation nicht sinnvoll. Die unterschiedlichen Anwendungssysteme werden deshalb auf der Anwendungsebene miteinander gekoppelt (vergleiche hierzu auch Abschnitt 3.5). Häufig ist es an dieser Stelle sinnvoll, neben den bereits diskutierten dreistufigen Client-Server-Architekturen vierstufige Client-Server-Architekturen zu betrachten. Die vierte Stufe wird durch Internetserver gebildet, die zwischen Anwendungs- und Präsentationsschicht angesiedelt sind. Durch Webbrowser kann von einem bestimmten Unternehmen aus über das Internet und die Anwendungsserver auf die Daten eines anderen Unternehmens zugegriffen werden. Die beschriebene Situation ist in Abbildung 2.24 gezeigt. Wir erhalten eine vierstufige Client-Server-Architektur.

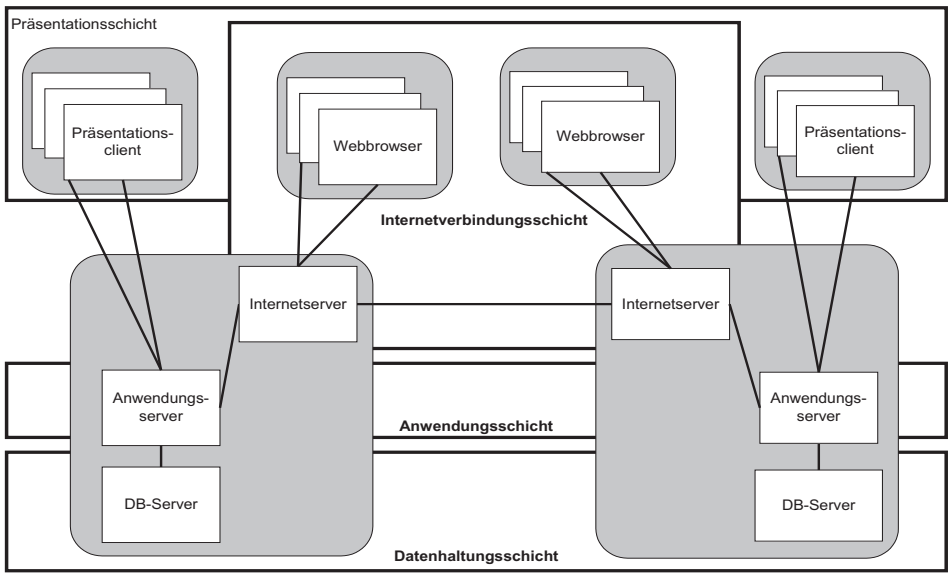

Abbildung 2.24: Vierstufige Client-Server-Architektur [2]

SAP R/3-Architektur

Wir konkretisieren die Architektur der Anwendungsschicht am Beispiel des R/3-Systems.

Beispiel 2.6.7 (Architektur der Anwendungsschicht) *Das R/3-Laufzeitsystem besteht aus einer Menge paralleler, kooperierender Systemprozesse [6, 27]. Ein zentraler Dispatcher wird verwendet. Daneben existiert eine variable Anzahl von sogenannten Workprozessen. Workprozesse werden für*

- *die Dialogverarbeitung,*

- *zum Verbuchen von Änderungen,*
- *die Batchverarbeitung,*
- *zum Drucken (Spooling)*

verwendet. Die Aufgabe des (zentralen) Dispatchers besteht in der Verteilung anstehender Verarbeitungsaufträge an die zuständigen Workprozesse. Pro Anwendungsserver gibt es genau einen Dispatcher. Workprozesse können direkt auf die Datenbank via SQL zugreifen. Eine Kommunikation mit der Systemumgebung erfolgt über den Dispatcher. Jedem Workprozess ist ein Taskhandler zugeordnet, der Aktivitäten des Workprozesses koordiniert. Die beschriebene Architektur ist in Abbildung 2.25 dargestellt.

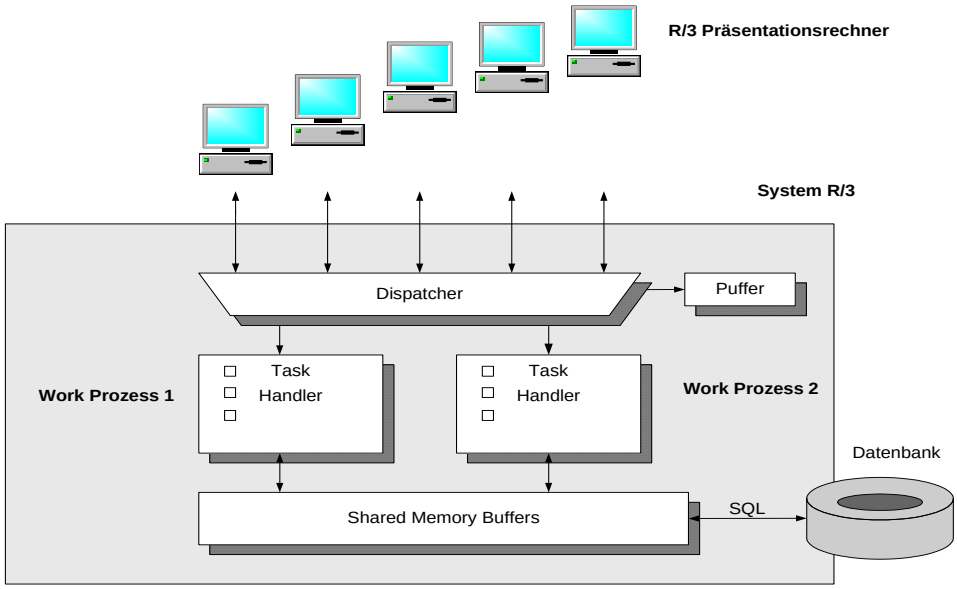

Abbildung 2.25: Dreistufige Architektur des SAP R/3-Systems [6]

Datenhaltungs- und Präsentationsschicht sind über Kommunikationskanäle mit der Anwendungsschicht verbunden. Die vollständige Trennung von Präsentations- und Anwendungsschicht ist nicht immer gegeben. Beispielsweise treten

- eingabeabhängige Plausibilitätsprüfungen und
- dynamische Umgestaltung von Masken aufgrund von Benutzereingaben

auf, die typischerweise von der Funktionalität der Anwendungsschicht softwaretechnisch realisiert werden, obwohl es sich um Funktionalität der Präsentationsschicht handelt. Diese auf die Präsentation abgestimmte Anwendungsschicht

führt zu Nachteilen und verhindert eine Umsetzung des Prinzips der Trennung der Zuständigkeiten.

Dreistufige Client-Server-Architekturen für betriebliche Anwendungssysteme sind in Abbildung 2.26 dargestellt.

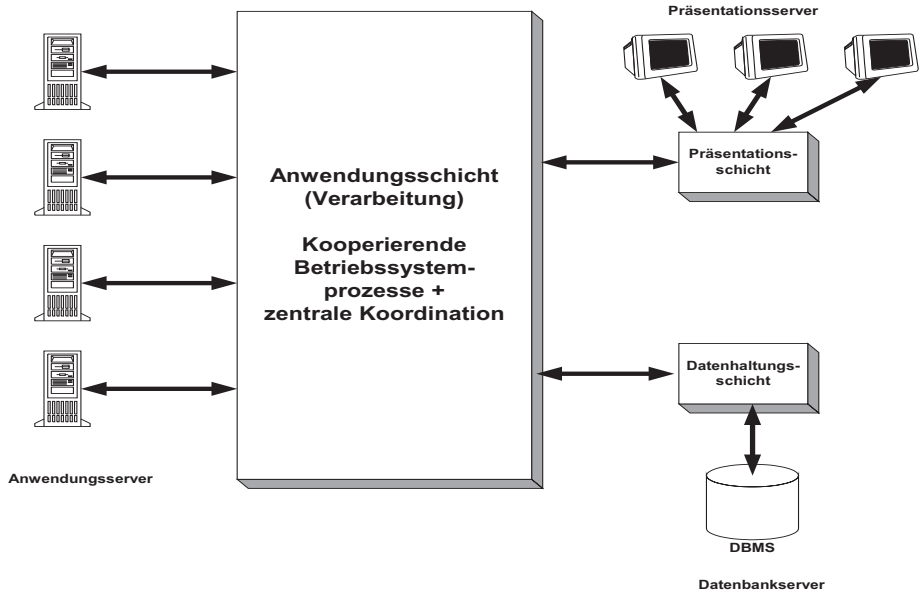

Abbildung 2.26: Dreistufige Architektur aus Betriebsmittelsicht

Die eben beschriebene dreistufige Client-Server-Architektur findet heute in den meisten großen betrieblichen Standardsoftwaresystemen Anwendung.

Bisher wurde zur Anwendungsschicht nur die Aussage getroffen, dass diese als Menge kooperierender Betriebssystemprozesse organisiert ist und damit prinzipiell auf mehrere Rechner verteilt werden kann. Unter funktionalen Gesichtspunkten kann aber zusätzlich eine weitere Dekomposition der Anwendungsschicht und damit der Verarbeitungslogik vorgenommen werden. Die Verarbeitungskomponente kann in die Subkomponenten Funktion und betriebliche Ablauflogik unterteilt werden [26].

Funktion und Ablauflogik

Die Subkomponente Funktion stellt Dienste bereit, die Tätigkeiten der Benutzer des Anwendungssystems automatisieren. Wir betrachten dazu folgendes Beispiel.

Beispiel 2.6.8 (Automatisierung von Aufgaben) *Durch ERP-Systeme werden die Tätigkeiten der Ablaufplanung und der Stücklistenauflösung partiell automatisiert durchgeführt.*

Die Subkomponente betriebliche Ablauflogik erlaubt jetzt im zweiten Schritt das Verknüpfen einzelner Funktionen zu Workflows und überwacht die korrekte

Ausführung der Workflows.

Vermittlung

Durch das Herauslösen der Ablauflogik aus der Anwendungsschicht entsteht zusätzlicher Kommunikationsbedarf zwischen der Subkomponente Funktion und Ablauflogik. Dieser Kommunikationsbedarf wird durch die Einführung einer zusätzlichen Vermittlungskomponente gedeckt. Die Aufgaben dieser Vermittlungskomponenten sind insbesondere in einer heterogenen Systemlandschaft nicht trivial. Unter einer heterogenen Systemlandschaft verstehen wir ein Umfeld, in dem die verschiedenen Komponenten des Anwendungssystems unter Verwendung unterschiedlicher Betriebssysteme und Hardware sowie unter Verwendung unterschiedlicher Technologien realisiert werden [26]. Die Aufgabe der Vermittlungskomponenten besteht darin, Dienstnehmer und -anbieter zu verbinden und eine Kommunikation in einem gebenenenfalls heterogenen Systemumfeld zu ermöglichen. Die Aufgabe einer Vermittlungskomponente wird typischerweise durch Middleware erledigt. Wir verweisen in diesem Zusammenhang auf Abschnitt 4.2 dieser Kurseinheit für eine Darstellung von Middlewarekonzepten. Die Entwicklung einer dreistufigen Client-Server-Architektur bis hin zu einer Architektur mit herausgelöster Ablauflogik und zusätzlicher Vermittlungsfunktionalität ist in Abbildung 2.27 dargestellt.

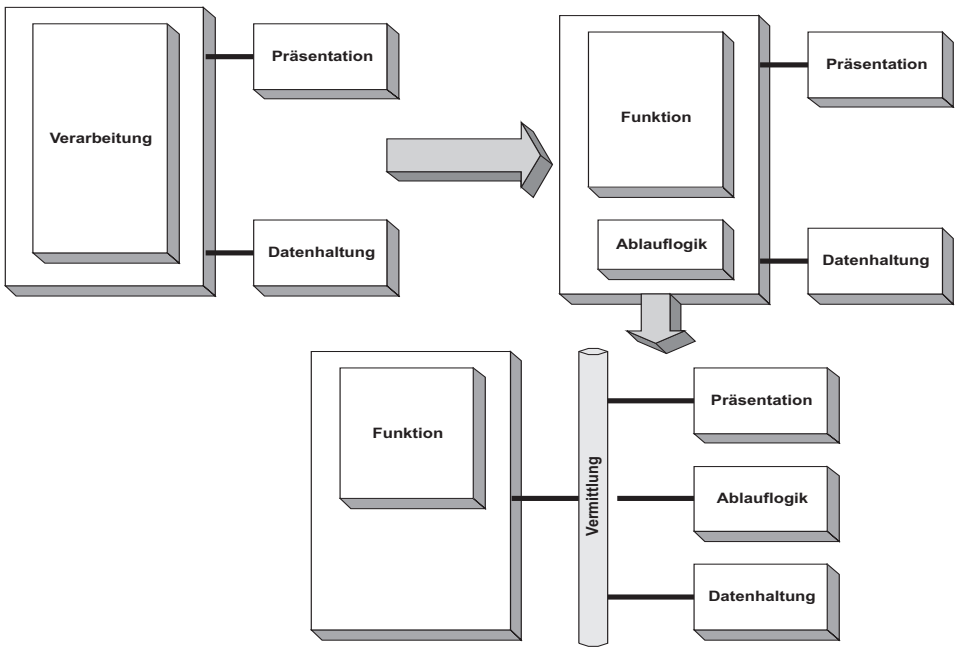

Abbildung 2.27: Dreistufige Architektur mit Vermittlungskomponente [26]

SOA

Wir bemerken, dass die Unterteilung der Verarbeitung in die Subsysteme Funktion und Ablauflogik die Grundideen einer SOA unterstützt. Das Subsystem für die Funktionen stellt Dienste bereit, während die Ablauflogikkomponente im Wesentlichen der Orchestrierungs- bzw. Choreographiefunktionalität entspricht.

Allerdings stehen die Dienste im eben beschriebenen Kontext, anders als bei der SOA, nur innerhalb des Anwendungssystems zur Verfügung. Das entspricht einer Mikro-Perspektive. SOA bezieht sich hingegen immer auf eine Systemlandschaft und dient der Gestaltung von Makro-Abläufen [52].

Die Anwendungsfunktionalität lässt sich zusätzlich in weitere kleinere Einheiten, sogenannte Fachkomponenten, zerlegen. Das führt zu komponentenbasierten Anwendungssystemarchitekturen, die im nächsten Abschnitt behandelt werden. Wir bemerken, dass wir bereits in Abschnitt 2.3.4 auf die Bedeutung von Komponenten für SOA hingewiesen hatten.

Übungsaufgabe 2.8 (Muster für Client-Server-Architekturen) *Zwei Arten von Mustern für Client-Server-Architekturen existieren. Zum einen müssen die Aufrufe der Clients durch den Server entsprechend behandelt werden und zum anderen gibt es Muster für die Bereitstellung der Ergebnisse für den Client durch asynchrone Aufrufe.*
Ein Server muss ständig erreichbar sein. Trifft eine Anfrage von einem Client beim Server ein, so kann er diese nicht selbst bearbeiten, da er sonst für andere Clients nicht mehr erreichbar wäre. Hierfür wurde das Server-Handler-Muster entworfen, bei dem die Aufrufe von Clients, die beim Server eingehen, durch diesen an sogenannte Handler delegiert werden. Die Handler übernehmen für den Server die Bearbeitung der Anfrage des Clients und die Kommunikation mit dem Client.
Es gibt generell zwei verschiedene Arten von Server-Aufrufen: synchrone und asynchrone Aufrufe. Bei synchronen Aufrufen wartet der Client solange, bis der Server die Ergebnisse bereitstellt und setzt erst danach die Bearbeitung fort. Da die Bearbeitung einer Anfrage durch den Server häufig länger dauert, kann der Client in dieser Zeit eine andere Aufgabe bearbeiten, für welche die Antwort des Servers nicht relevant ist. Man kommt zu asynchronen Aufrufen, die serverseitig durch Caller behandelt werden. Hierbei werden zwei Möglichkeiten der Bereitstellung der Ergebnisse durch den Caller unterschieden. Der Client kann beim Caller von Zeit zu Zeit nachfragen, ob die Ergebnisse der Anfrage bereitstehen. Das wird als Polling bezeichnet. Sendet der Caller die Ergebnisse nach der Bearbeitung direkt an den Client, so nennt man dies Callback.
Stellen Sie die beschriebenen Muster für Client-Server-Architekturen geeignet graphisch dar.

2.6.3 Komponentenbasierte Anwendungssysteme

Wir führen an dieser Stelle zunächst den Komponentenbegriff ein [20].

Komponenten

Definition 2.6.1 (Komponente) *Unter einer Komponente verstehen wir ein eigenständiges, wiederverwendbares, abgeschlossenes Stück Software. Sie ist unabhängig von bestimmten Anwendungssystemen. Eine Komponente stellt ihrer Umgebung wohldefinierte Dienste zur Verfügung.*

Komponenten verfügen über die nachfolgenden Eigenschaften:

- **Unabhängigkeit**: Komponenten und die durch sie angebotenen Dienste existieren unabhängig von bestimmten Anwendungen.
- **Abgeschlossenheit**: Die Funktionalität einer Komponente ist unabhängig von anderen Komponenten ablauffähig.
- **Offenheit**: Komponenten sind in der Lage, mit anderen Komponenten in beliebigen Systemumgebungen zusammenzuarbeiten. Sie stellen dazu Schnittstellen zur Verfügung.

Die bisherigen Aussagen und Erkenntnisse zur Softwarearchitektur betrieblicher Anwendungssysteme werden, wie in Abbildung 2.28 dargestellt, zu einer komponentenbasierten Architektur weiterentwickelt. Die kleineren Quadrate im Kasten Funktionalität symbolisieren die unterschiedlichen Komponenten. Diese Architektur ist heute die grundsätzliche Architektur moderner betrieblicher Anwendungssysteme. Wir bemerken, dass dieser Architektur auch in wesentlichen Teilen die in Abschnitt 2.3.4 beschriebenen SOA folgt. Dabei wird allerdings die Perspektive einer Systemlandschaft eingenommen.

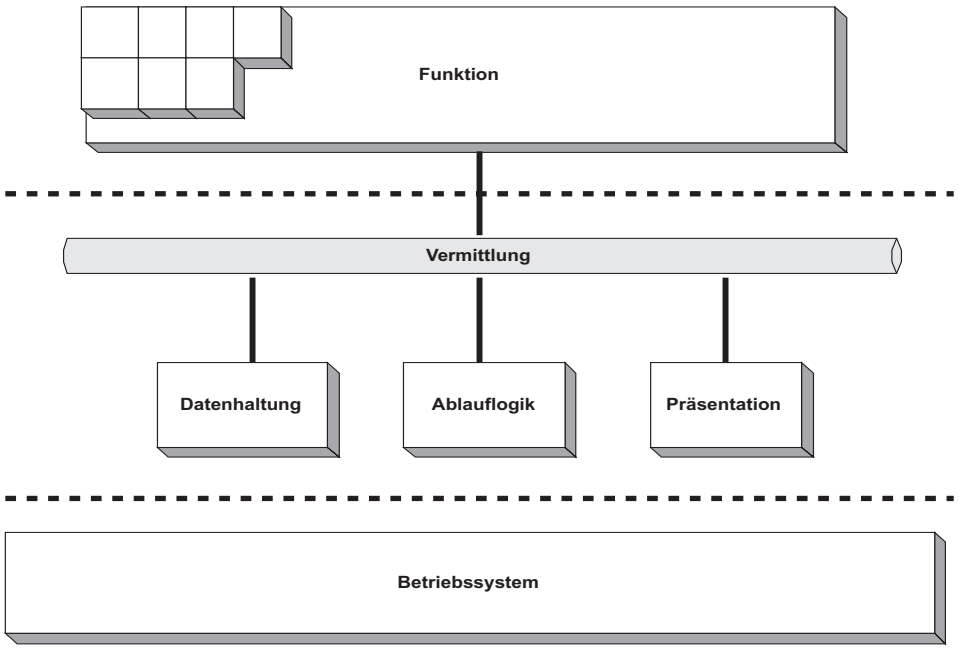

Abbildung 2.28: Bestandteile moderner betrieblicher Anwendungssysteme

Neben den allgemeinen Komponenten werden zusätzlich Fachkomponenten betrachtet.

Fachkomponenten

Definition 2.6.2 (Fachkomponente) *Unter einer Fachkomponente verstehen wir in Anlehnung an [11] eine Komponente, die*

1. *prinzipiell vermarktbar ist,*
2. *über eine wohldefinierte Schnittstelle verfügt und deshalb in zur Entwicklungszeit unvorhersagbaren Kombinationen mit anderen Komponenten einsetzbar ist,*
3. *an unternehmensspezifische Besonderheiten anpassbar ist,*
4. *eine wohldefinierte Menge von Aufgaben einer betrieblichen Anwendungsdomäne löst.*

Fachkomponenten sind spezielle Komponenten, die auf die Lösung von Aufgaben einer bestimmten betrieblichen Anwendungsdomäne abzielen. Fachkomponenten können selber wieder aus Fachkomponenten bestehen. Wir betrachten dazu das nachfolgende Beispiel.

Beispiel 2.6.9 (Rekursive Fachkomponenten) *Eine Fachkomponente zur Produktionsplanung wird typischerweise die Fachkomponenten „Ablaufplanung“ und „Bedarfsprognose“ enthalten.*

Übungsaufgabe 2.9 (Rekursive Fachkomponenten) *Geben Sie ein Beispiel für eine rekursive Fachkomponente an und stellen Sie die darin enthaltenen Fachkomponenten dar.*

Softwaretypen

Heutige betriebliche Anwendungssysteme umfassen, wie in Abbildung 2.28 dargestellt, stets drei Typen von Software:

1. das Betriebssystem, das eine wohldefinierte Schnittstelle für Anwendungssysteme zur Verfügung stellt, das technische Einzelheiten der Rechner-Hardware vor dem Benutzer verbirgt sowie die Koordination und Zuteilung von Ressourcen bei konkurrierend ablaufenden Programmen übernimmt,
2. Komponenten, die unabhängig von den betrieblichen Aufgaben sind,
3. Komponenten, die zur Lösung betrieblicher Problemstellungen verwendet werden.

Diese Typen lassen sich den nachfolgenden Ebenen zuordnen:

- Hardwareebene,
- Betriebssystemebene,
- Middlewarebene,
- Anwendungsebene.

BCA

Eine aus diesen Ebenen bestehende Architektur wird als Business-Component-Architecture (BCA) bezeichnet [26]. Unter Middleware verstehen wir in diesem Zusammenhang eine Zusammenfassung derjenigen anwendungsunabhängigen Systembestandteile, die nicht zum Betriebssystem gehören. Middlewarekonzepte werden in Abschnitt 4.2 genauer dargestellt.

Der BCA-Ansatz wird in Anlehnung an [26] in Abbildung 2.29 dargestellt. Abbildung 2.29 veranschaulicht somit den prinzipiellen Aufbau komponentenbasierter Anwendungssysteme. In dieser Abbildung werden mit dem

- Middleware-Integrationsrahmenwerk,
- System-Infrastrukturrahmenwerk,
- Komponenten-Anwendungsrahmenwerk

Rahmenwerke

drei zusätzliche Systembestandteile angegeben. Das Middleware-Integrationsrahmenwerk und das System-Infrastrukturrahmenwerk bilden zusammen das Komponenten-System-Rahmenwerk. Das Komponenten-Anwendungsrahmenwerk wird auch als Enterprise-Application-Framework [10] bezeichnet. Wir geben an dieser Stelle eine erste, noch ziemlich grobe Definition für Rahmenwerke an.

Definition 2.6.3 (Rahmenwerk) *Unter einem Rahmenwerk (Framework) verstehen wir einen Quellcode-Rahmen, der für eine bestimmte Klasse von Anwendungen ein Skelett vorgibt.*

Rahmenwerke werden ausführlich in Abschnitt 5.4 behandelt. Dort wird auch Definition 2.6.3 verfeinert werden. Wir beschäftigen uns nun mit Komponenten-System-Rahmenwerken.

Komponenten-System-Rahmenwerk

Definition 2.6.4 (Komponenten-System-Rahmenwerk) *Komponenten-System-Rahmenwerke stellen der aus Fachkomponenten bestehenden Anwendung middlewarenahe, anwendungsunabhängige Dienste zur Verfügung.*

Dienste

Wir verweisen in diesem Zusammenhang insbesondere auf Vermittlungsdienste. Unter Vermittlung verstehen wir an dieser Stelle die Möglichkeit zu entfernten Methodenaufrufen. Daneben werden Persistenzmechanismen sowie Möglichkeiten zur Überwachung von speziellen Ereignissen durch Komponenten-System-Rahmenwerke angeboten.

Spezielle Dienste wie die Unterstützung von Transaktionen werden ebenfalls durch das Komponenten-System-Rahmenwerk abgedeckt. DBMS sowie WFMS sind als spezielle Ausprägungen von Komponenten-System-Rahmenwerken aufzufassen. Vermittlungsdienste sind der wesentliche Bestandteil des Middleware-Integrationsrahmenwerks. Das System-Infrastrukturrahmenwerk beschäftigt sich dagegen beispielsweise mit dem Zugriff auf Dateien und unterstützt Persistenzmechanismen.

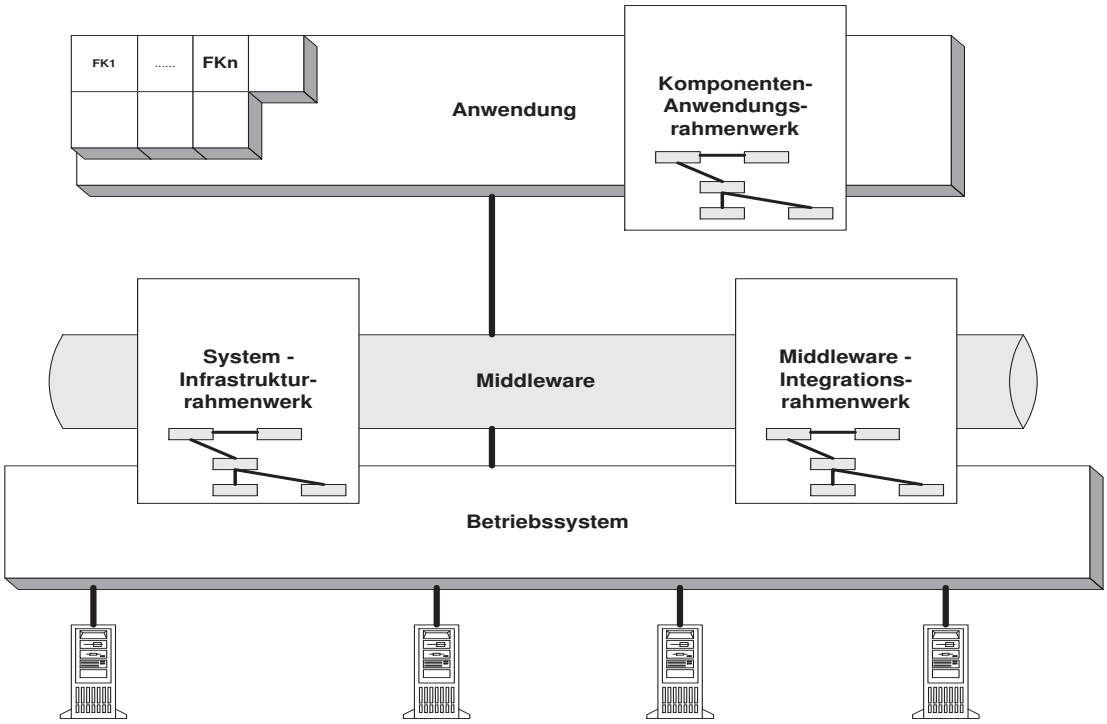

Abbildung 2.29: BCA-Bestandteile

Komponenten-Anwendungsrahmenwerke

Komponenten-System-Rahmenwerke sind von der konkreten Domäne unabhängig. Komponenten-Anwendungsrahmenwerke nehmen im Gegensatz dazu starken Bezug auf die Domäne. Wir definieren deshalb den Begriff der Komponenten-Anwendungsrahmenwerke wie folgt.

Definition 2.6.5 (Komponenten-Anwendungsrahmenwerk) *Das Komponenten-Anwendungsrahmenwerk ist ein Systembestandteil, der domänenspezifische Dienste bereitstellt sowie die unterschiedlichen Dienste zusammenführt und integriert.*

Wir betrachten das folgende Beispiel für anwendungsdomänenspezifische Dienste.

Beispiel 2.6.10 (Anwendungsdomänenspezifische Dienste) *Verfahren zur Behandlung fachlicher Konflikte oder zur Erzeugung von Geschäftsobjekten, die von mehreren Fachkomponenten genutzt werden, sind anwendungsdomänenspezifische Dienste.*

Die Verwendung mehrerer Anwendungsrahmenwerke innerhalb eines einzelnen Anwendungssystems ist prinzipiell möglich und oft auch sinnvoll.

In Tabelle 2.1 stellen wir konventionelle Anwendungssysteme komponentenbasierten Lösungen gegenüber und greifen damit die Diskussion aus Abschnitt 2.3.3 erneut auf.

Tabelle 2.1: Konventionelle vs. komponentenbasierte Anwendungssysteme

Merkmal	Konventionelle Anwendungssysteme	Komponentenbasierte Anwendungssysteme
Architektur	Client-Server-Architektur	SOA in der Mikroperspektive
Integrationsmodell	datenzentriert	nachrichtenzentriert
Zugriff	transaktionsorientiert	geschäftsprozessbezogene Benutzer
Anbindung an das Internet	via Webserver	via Portal
Prozessorientierung	Unternehmensprozesse	unternehmensübergreifende Prozesse

Vergleich konventioneller und komponentenbasierter Systeme

Es wird deutlich, dass insbesondere die Vermittlungskomponente dazu führt, dass auf nachrichtenbasierte Art und Weise Geschäftsprozesse sowohl unternehmensweit als auch unternehmensübergreifend abgebildet werden können. Damit besitzen komponentenbasierte Anwendungssysteme die typischen Eigenschaften dienstorientierter Informationssysteme.

Es wird deutlich, dass aufgrund des nachrichtenzentrierten Integrationsmodells komponentenbasierte Anwendungssysteme unternehmensübergreifende Geschäftsprozesse unterstützen. Konventionelle Anwendungssysteme sind in einem stärkeren Maße datenzentriert. In der Konsequenz muss mehr Aufwand betrieben werden, um unternehmensübergreifend arbeiten zu können.

Komponenten vs. SOA

Wir weisen an dieser Stelle erneut auf die Analogien zwischen komponentenbasierten Anwendungssystemen und der SOA hin. Komponentenbasierte Anwendungssysteme implementieren eine SOA im Kleinen, während umgekehrt SOA als Komponentenorientierung im Großen aufgefasst werden kann.

Wir bemerken an dieser Stelle, dass das System SAP R/3 bis zum Release 4.7 im Sinne der Gegenüberstellung in Tabelle 2.1 und der Ausführungen in Abschnitt 2.3.3 dieser Kurseinheit ein konventionelles Anwendungssystem ist, während sein Nachfolger mySAP ERP sich in Richtung eines modernen, komponentenbasierten Anwendungssystems entwickelt.

Lösungen zu den Übungsaufgaben

Übungsaufgabe 2.1
Im ersten Schritt wird das Geschäftsobjekt BO zerlegt. Die entstehenden Geschäftsobjekte stehen durch die Geschäftstransaktion TA miteinander in Beziehung:

$$BO ::= \{BO_1, BO_2, TA(BO_1, BO_2)\}.$$

Danach wird die Geschäftstransaktion TA in Untergeschäftstransaktionen zerlegt:

$$TA(BO_1, BO_2) ::= TA_i(BO_1, BO_2)seq, TA_{con}(BO_2, BO_1)seq, TA_e(BO_1, BO_2).$$

Im letzten Schritt wird das Geschäftsobjekt BO_1 zerlegt und mit den Geschäftstransaktionen entsprechend in Beziehung gesetzt:

$$BO_1 ::= \{BO_{m1}, BO_{o1}, TA_c(BO_{m1}, BO_{o1}), TA_f(BO_{o1}, BO_{m1})\}.$$

Abschließend müssen die Geschäftstransaktionen TA_i, TA_{con} und TA_e angepasst werden:

$$\begin{aligned} TA_i(BO_1, BO_2) &::= TA_i(BO_{m1}, BO_2), \\ TA_{con}(BO_2, BO_1) &::= TA_{con}(BO_2, BO_{m1}), \\ TA_e(BO_1, BO_2) &::= TA_e(BO_{o1}, BO_2). \end{aligned}$$

Übungsaufgabe 2.2
Für den Geschäftsprozess „Vertrieb“ werden die Geschäftsobjekte „Kunde“ und „Verkäufer“ identifiziert. Das Geschäftsobjekt „Kunde“ ist ein externes Objekt. Die Geschäftstransaktion ist der Verkauf, welche durch den Kunden durch eine Angebotsanforderung initiiert wird. Es ergibt sich das in Abbildung 2.30 dargestellte Interaktions- und Aufgaben-Ereignis-Schema.

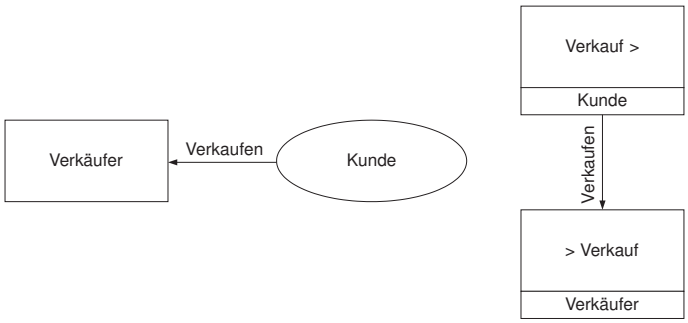

Abbildung 2.30: Geschäftsprozess „Vertrieb“: Interaktions- und Aufgaben-Ereignis-Schema

Die Geschäftstransaktion „Verkauf“ wird in vier Untergeschäftstransaktionen zerlegt:

- Angebot einholen,
- Angebot abgeben,
- Vertragsabschluss,
- Durchführung des eigentlichen Verkaufs.

Hierbei ergibt sich das in Abbildung 2.31 dargestellte Interaktions- und Aufgaben-Ereignis-Schema.

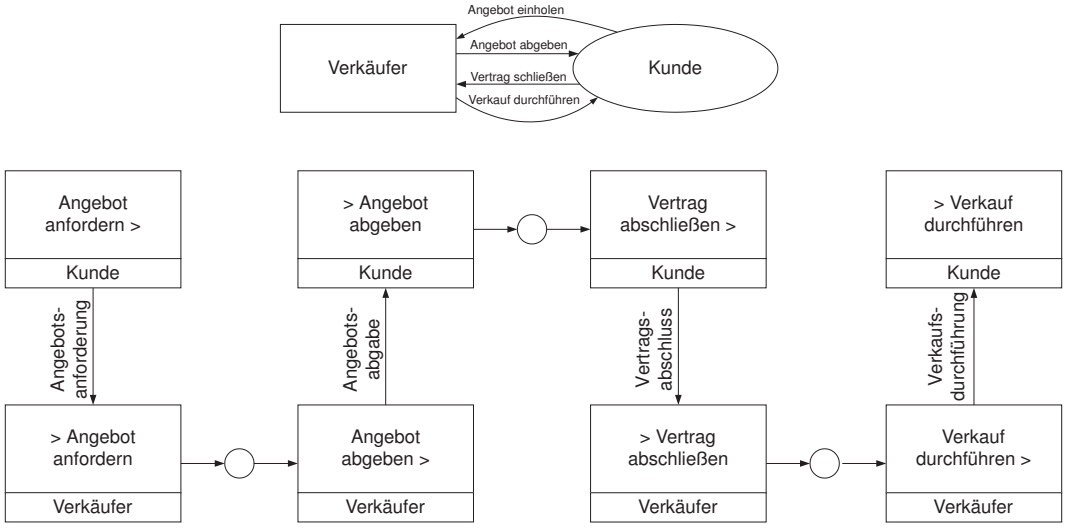

Abbildung 2.31: Geschäftsprozess „Vertrieb“: Interaktions- und Aufgaben-Ereignis-Schema nach Zerlegung

Übungsaufgabe 2.3
Aus der Beschreibung kann das in Abbildung 2.32 dargestellte ERM modelliert werden.

Übungsaufgabe 2.4
Für jede Entität wird eine Tabelle angelegt. Zusätzlich wird für die N:M-Beziehungen der Entitäten Arbeitsplan und Arbeitsschritt, zur Abbildung der Arbeitsschrittnummern sowie Arbeitsschritt und Maschine jeweils eine Tabelle verwendet. Bei den 1:1- und 1:N-Beziehungen muss bei mindestens einer der beiden entstehenden Tabellen der Primärschlüssel der jeweils anderen Tabelle als Feld eingefügt werden. Bei der aus einer N:M-Beziehung resultierenden Beziehungstabelle werden die beiden Primärschlüssel der beteiligten Tabellen als Primärschlüssel der Beziehungstabelle verwendet.

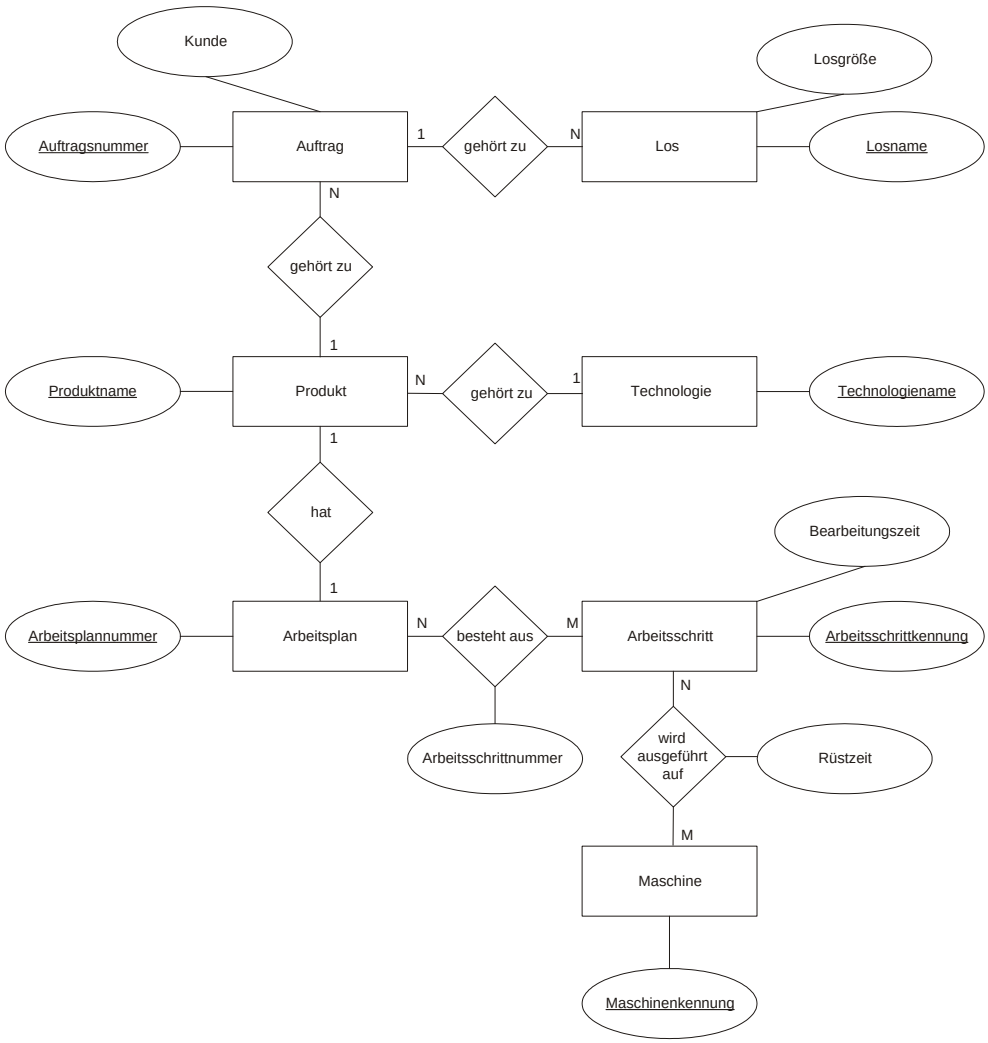

Abbildung 2.32: ERM für die Fachkonzeptebene

Folgende Tabellenstruktur entsteht bei der Überführung des ERM:

Auftrag		
Auftragsnummer	Kunde	Produktname

Technologie
Technologiename

Los		
Losname	Losgröße	Auftragsnummer

Produkt		
Produktname	Technologiename	Arbeitsplannummer

Arbeitsplan	
Arbeitsplannummer	Produktname

Arbeitsschritt	
Arbeitsschrittkennung	Bearbeitungszeit

Maschine
Maschinenkennung

Beziehungstabellen:

Arbeitsschrittnummern		
Arbeitsplannummer	Arbeitsschrittkennung	Arbeitsschrittnummer

Arbeitsschritt-Maschinen		
Arbeitsschrittkennung	Maschinenkennung	Rüstzeit

Übungsaufgabe 2.5
Durch die in Abbildung 2.33 dargestellte EPK kann der Geschäftsprozess der Auftragsverarbeitung modelliert werden. In der nachfolgenden Tabelle sind mögliche Dienste, die bei der Abarbeitung des Geschäftsprozesses in Anspruch genommen werden, dargestellt.

Aktion	Dienst
Auftragsbearbeitung	Kundendatenverarbeitung Auftragserfassung
Auftragsbestätigung erstellen	Auftragsterminierung Preisfindung
Kapazitätsplanung	Kapazitätsplanung
Auftragsfreigabe	Auftragseinsteuerung Mengenplanung der Teile anhand der Stückliste

Übungsaufgabe 2.6
Ein möglicher Ablauf der Produktion eines Auftrags kann wie im in Abbildung 2.34 dargestellten Sequenzdiagramm erfolgen.
Nachdem der Auftrag durch den Kunden bestätigt wird, erfolgt die Meldung des Auftrags an die Buchhaltung. Danach wird der Auftrag für die Produktion freigegeben, welche die Materialbeschaffung initiiert. Danach erfolgt die Grobterminierung des Auftrags und die Meldung des Termins an den Vertrieb. Dieser bestätigt den Termin. Die Produktion des Auftrags kann erfolgen. Nach der Fertigstellung werden sowohl der Vertrieb als auch die Buchhaltung zur Erstellung der Rechnung darüber informiert.
In Abbildung 2.35 ist die Choreographie der Dienste dargestellt, die für dieses Szenario komponiert werden. Bei der Choreographie erfolgt die Darstellung des Zusammenspiels der einzelnen Dienste zur Erfüllung einer bestimmten Aufgabe. Die internen Vorgänge werden nicht dargestellt.
Bei der Orchestrierung wird aus Sicht eines einzelnen Dienstes die Komposition der Dienste dargestellt. Die Orchestrierung für die Produktionsausführung ist in Abbildung 2.36 gezeigt.

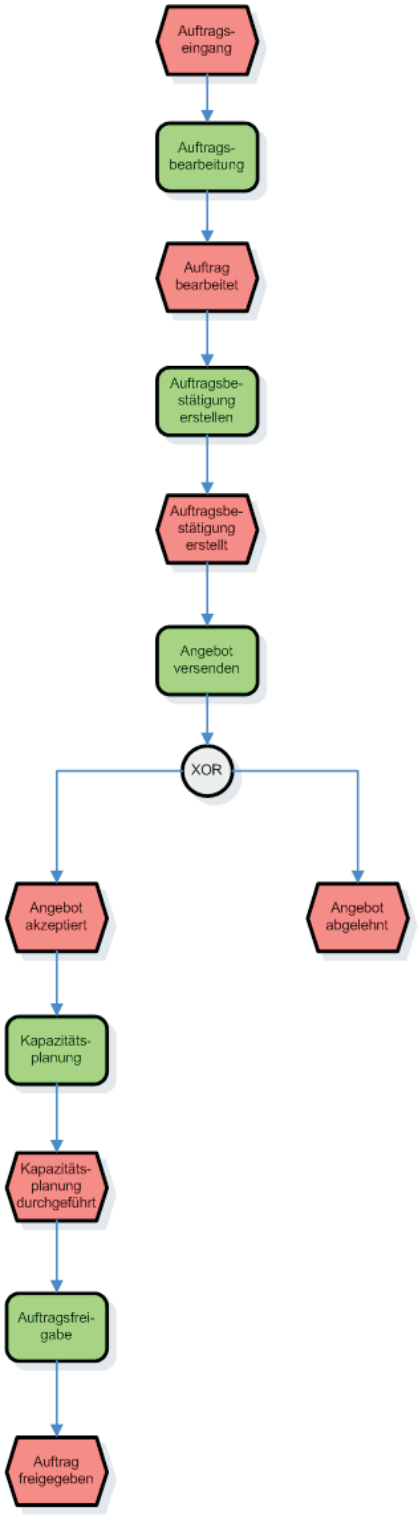

Abbildung 2.33: EPK zur Auftragsabwicklung

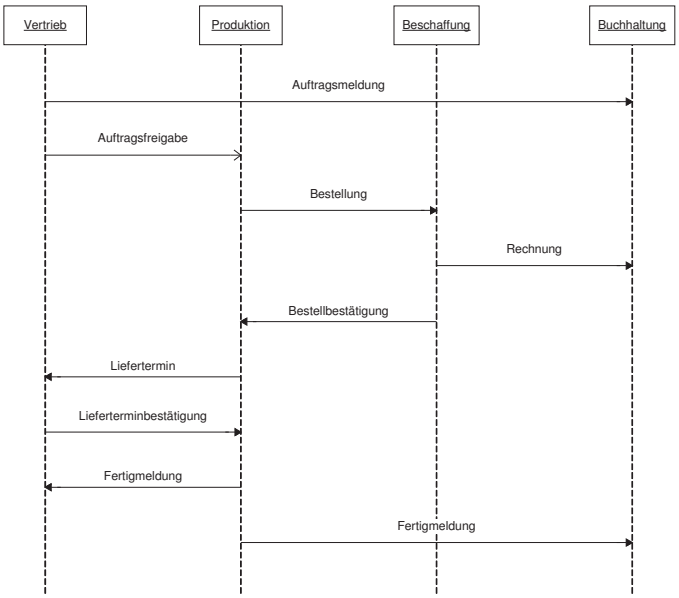

Abbildung 2.34: Sequenzdiagramm für die Produktion eines Auftrags

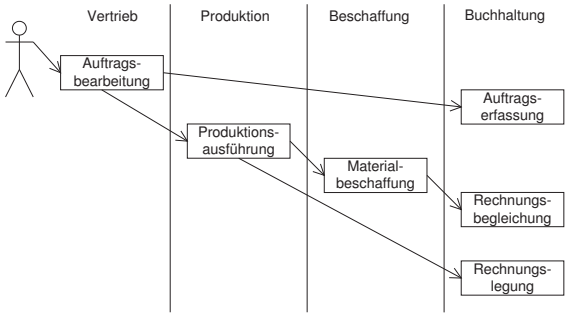

Abbildung 2.35: Choreographie der Dienste in der Produktion

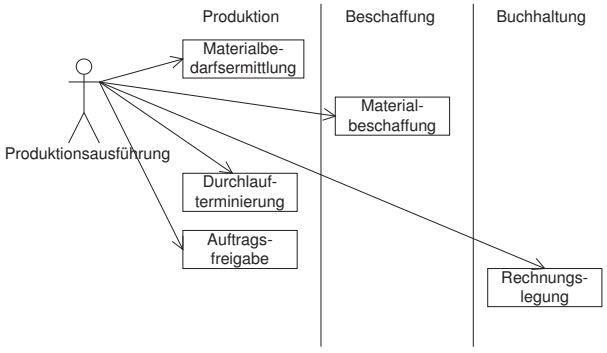

Abbildung 2.36: Orchestrierung der Produktionsausführung

Intern werden die Dienste Materialbedarfsermittlung, Durchlaufterminierung und Auftragsfreigabe genutzt. Weiterhin werden, wie auch aus der Choreographie ersichtlich, die Dienste Materialbeschaffung und Rechnungslegung in Anspruch genommen. Die Orchestrierung des Dienstes Materialbeschaffung ist der Abbildung 2.37 zu entnehmen.

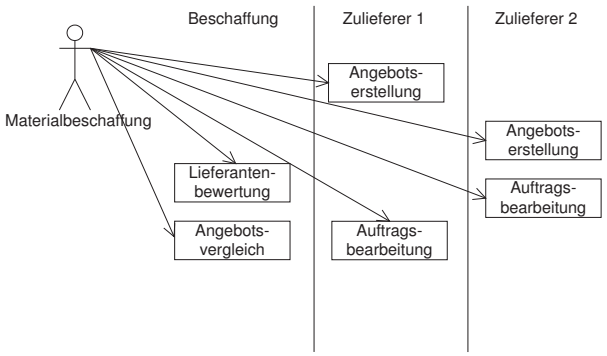

Abbildung 2.37: Orchestrierung der Materialbeschaffung

Intern werden die Dienste Lieferantenbewertung und Angebotsvergleich in Anspruch genommen. Weiterhin werden die Dienste Angebotserstellung und Auftragsbearbeitung der Zulieferer 1 und 2 genutzt.

Übungsaufgabe 2.7

zu a)

In SOM werden objektorientierte und geschäftsprozessorientierte Modellierungskonzepte verwendet. Folgende Modellebenen, Sichten und Metamodelle werden beschrieben.

Modellebenen	Sichten	Metamodelle
Unternehmens-planebene	Außensicht	Objektsystem Zielsystem
Geschäftsprozess-modellebene	Innensicht	Interaktionsschema Vorgangs-Ereignis-Schema
Ressourcenebene	Aufgabenträger-sicht	Konzeptuelles Objektschema Vorgangsobjektschema Metamodell für Anwendungssysteme

Als Beispiel für die Beziehung zwischen den Ebenen kann die Beziehung zwischen Geschäftsprozessmodellebene und der Ressourcenebene betrachtet werden. Hier erfolgt eine Abbildung der Elemente des Aufgaben-Ereignis-Schemas auf Klassen des Metamodelles für Anwendungssysteme, das ein Schema für konzeptuelle Klassen und ein Schema für Aufgabenklassen beinhaltet.

zu b)
In ARIS werden folgende Modellebenen, Sichten und Metamodelle betrachtet.

Modellebenen	Sichten	Metamodelle
Fachkonzept	Funktionssicht	Funktionsbäume Interaktionsdiagramme
	Datensicht	ERM
	Organisationssicht	Organigramme
	Steuerungssicht	EPK
	Leistungssicht	Produktbäume Produktnetze
DV-Konzept	Funktionssicht	Struktogramme UML-Diagramme
	Datensicht	Relationenmodelle
	Organisationssicht	
	Steuerungssicht	
Implementierung	Funktionssicht	Programmiersprachen
	Datensicht	Relationale DBMS
	Organisationssicht	Hard- und Software
	Steuerungssicht	Middleware

Die Beziehung zwischen den Ebenen kann am Beispiel der Datensicht erläutert werden. Beim Fachkonzept erfolgt eine Modellierung mit Hilfe von Entity-Relationship-Modellen. Diese werden auf der DV-Konzeptebene in Relationenmodelle überführt und letztendlich bei der Implementierung in relationalen Datenbanken als Tabellen umgesetzt.

zu c)
Bei SOA gibt es zwei unterschiedliche Sichtweisen, die nachfolgend dargestellt werden.

Modellebenen	Sichten	Metamodelle
Prozessmodellebene		EPK
Dienstmodellebene		Choreographie Orchestrierung
Technikmodellebene		Workflow-BPEL Middleware-Konzepte

Auf der Prozessmodellebene werden die Geschäftsprozesse mit Hilfe von EPKs abgebildet. Anschließend werden auf der Dienstmodellebene die Dienste identifiziert und mit Hilfe der Choreographie und Orchestrierung die notwendigen Dienste komponiert. Auf der Technikmodellebene wird die Ausführung der Dienste über die BPEL realisiert.

Modellebenen	Sichten	Metamodelle
Operative Systeme	Daten- und Außensicht	Softwaresysteme
Unternehmenskomponenten	funktionale Sicht	Hard- und Software
Dienste	funktionale Sicht	WSDL
Choreographie und Orchestrierung	Geschäftsprozesssicht	Choreographie Orchestrierung Workflow-BPEL
Zugriff und Präsentation	Präsentationssicht	
Integration	Integrationssicht	
Sicherstellung der Qualität angebotener Dienste	Qualitätssicherungssicht	DIN/ISO 9000

Die Beziehung zwischen den Ebenen ist in Abbildung 2.20 dargestellt. Auf den unteren beiden Ebenen sind die Hard- und Softwaresysteme dargestellt. Auf der darüberliegenden Ebene werden die Dienste identifiziert, die in der nachfolgenden Ebene komponiert werden. Auf der obersten Ebene wird der Zugriff auf die Dienste spezifiziert.

Übungsaufgabe 2.8
Das Server-Handler-Muster kann, wie in Abbildung 2.38 gezeigt, dargestellt werden.

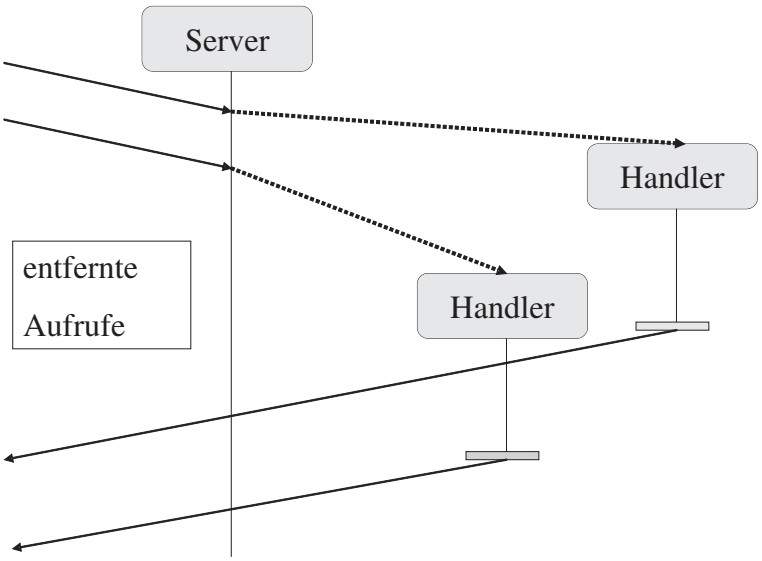

Abbildung 2.38: Server-Handler-Muster

Die Polling-Variante des asynchronen Aufrufmusters wird in Abbildung 2.39 gezeigt.

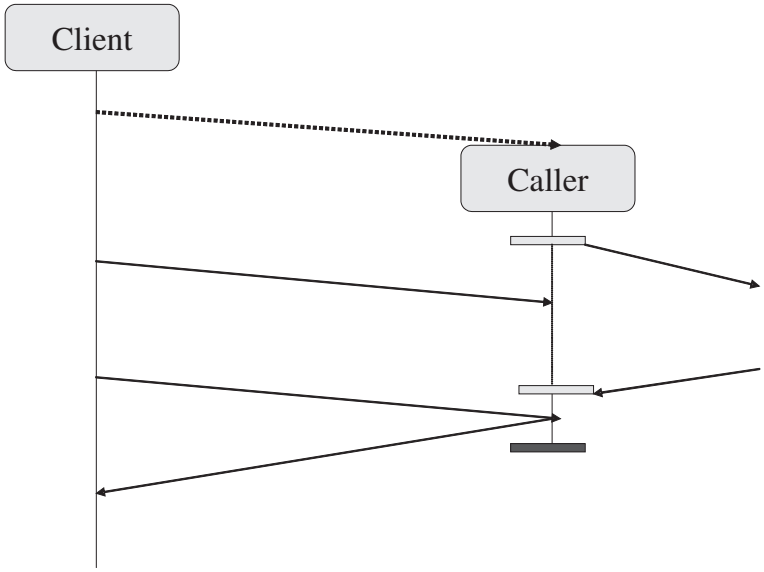

Abbildung 2.39: Asynchrones Aufrufmuster - Polling

Abschließend wird die Callback-Variante des asynchronen Aufrufmusters in Abbildung 2.40 gezeigt.

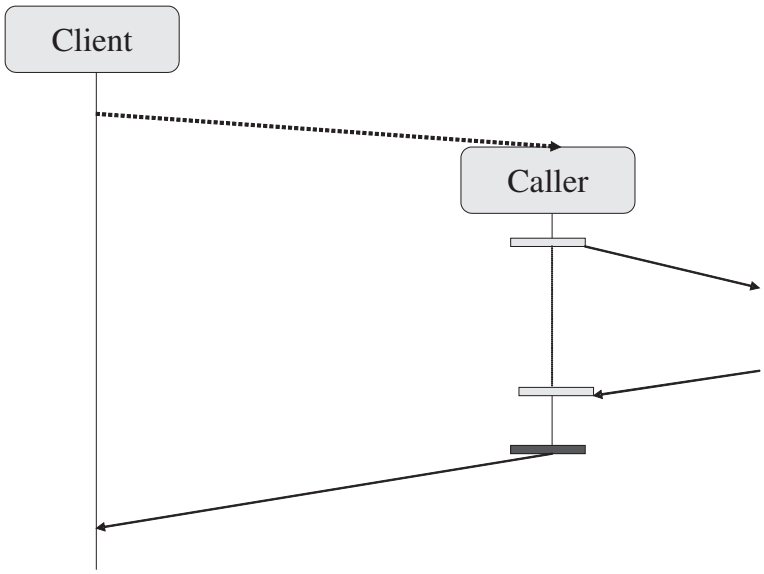

Abbildung 2.40: Asynchrone Aufrufmuster - Callback

Übungsaufgabe 2.9
Als Beispiel für eine rekursive Fachkomponente wird das SAP-Portal betrachtet. Dieses stellt die Fachkomponente Unternehmensweites Softwaresystem dar und enthält die Fachkomponenten:

- XI (Datenbus),
- APO (Produktionsplanung),
- BW (Data Warehouse),
- R/3 (ERP-System).

Literatur

[1] S. Abeck; P.C. Lockemann; J. Seitz. *Verteilte Informationssysteme: Integration von Datenübertragungstechnik und Datenbanktechnik*. dpunkt.Verlag, Heidelberg, 2003.

[2] H.-J. Appelrath; J. Ritter. *R/3-Einführung - Methoden und Werkzeuge*. Springer, 2000.

[3] A. Arsanjani. How to identify, specify and realize services for your SOA. http://81.29.72.180/index.php/ws/content/ view/full/ 55047. Abruf am 2005-11-07.

[4] P. Bernus; K. Mertins; G. Schmidt. *Handbook on Architecture of Information Systems*. 2. Auflage. Springer, Berlin, Heidelberg, New York, 2006.

[5] P. Brösler; J. Siedersleben. *Softwaretechnik*. Carl Hanser, München, 2000.

[6] R. Buck-Emden. *Die Technologie des R/3-Systems*. 4. Auflage. Addison-Wesley, Longmann, 1998.

[7] B. Burkhard; G. Laures. SOA - Wertstiftendes Architektur-Paradigma. *OBJEKTspektrum*, 6, 16–23, 2003.

[8] P. Coad. Object-oriented pattern. *Communications of the ACM*, 35(9), 152–159, 1992.

[9] K. Dittrich. Datenbanksysteme. P. Rechenberg; G. Pomberger (Hrsg.), *Handbuch der Informatik*, 3. Auflage, Hanser, München, 875–908, 2002.

[10] M. E. Fayad; Schmidt D. C.; R. E. Johnson. *Building Application Frameworks: Object-Oriented Foundation of Framework Design*. Wiley, New York, 1999.

[11] K.J. Fellner; C. Rautenstrauch; K. Turowski. Fachkomponenten zur Gestaltung betrieblicher Anwendungssysteme. *Information Management & Consulting*, 14(2), 25–34, 1999.

[12] O.K. Ferstl; E.J. Sinz. Der Ansatz des Semantischen Objektmodells (SOM) zur Modellierung von Geschäftsprozessen. *Wirtschaftsinformatik*, 37, 209–220, 1995.

[13] O.K. Ferstl; E.J. Sinz. Modeling of Business Systems Using SOM. P. Bernus; K. Mertins; G. Schmidt (Hrsg.), *Handbook on Architecture of Information Systems*, Springer, 347–367, 2006.

[14] C. Floyd; H. Züllighoven. Softwaretechnik. P. Rechenberg; G. Pomberger (Hrsg.), *Handbuch der Informatik*, 3. Auflage, Hanser, München, 763–790, 2002.

[15] M. Fowler. *Pattern of Enterprise Application Architecture*. Addison-Wesley, Pearson-Education, Boston, 2003.

[16] M.S. Fox; M. Grüninger. On ontologies and enterprise modelling. *Proceedings of the International Conference on Enterprise Integration Modelling Technology 97*. Springer-Verlag, 1997.

[17] P. Fremantle; S. Weerawarana; R. Khalaf. Enterprise Services. *Communication of the ACM*, 45, 77–82, 2002.

[18] E. Gamma; R. Helm; R. Johnson; J. Vlissides. *Design Patterns. Elements of Resusable Objectoriented Software*. Addison-Wesley, Longmann, Reading, 1997.

[19] H.-O. Günther. Supply chain management and advanced planning systems: a tutorial. H.-O. Günther; D.C. Mattfeld; L. Suhl (Hrsg.), *Supply Chain Management und Logistik: Optimierung, Simulation, Decision Support*, Physica-Verlag, Heidelberg, 2005.

[20] F. Griffel. *Componentware - Konzepte und Techniken eines Softwareparadigmas*. dpunkt-Verlag, Heidelberg, 1998.

[21] T.R. Gruber. Towards principles for the design of ontologies used for knowledge sharing. *International Journal Human-Computer Studies*, 43(5/6), 907–928, 1995.

[22] N. Guarino. Formal ontology and information systems. N. Guarino (Hrsg.), *Proceedings of the First International Conference (FOIS'98)*, Trento, IOS Press, Amsterdam, 3–15, June 1998. http://ontology.ip.rm.cnr.it/Papers/FOIS98.pdf.

[23] H. R. Hansen; G. Neumann. *Wirtschaftsinformatik 1: Grundlagen und Anwendungen*. 9. Auflage. Lucius & Lucius, Stuttgart, 2005.

[24] W. Hasselbring. Softwarearchitektur. *Informatik Spektrum*, 29(1), 48–52, 2006.

[25] M. Havey. *Essential Business Process Modeling*. O'Reilly, Sebastopol, 2005.

[26] S. Herden; J. Marx Gómez; C. Rautenstrauch; A. Zwanziger. *Software-Architekturen für das E-Business*. Springer, Berlin, Heidelberg, 2006.

[27] K. Hildebrand; M. Rebstock. *Betriebswirtschaftliche Einführung in SAP R/3*. Oldenbourg Verlag, München, 2000.

[28] S. Jablonski; M. Böhm; W. Schulze. *Workflow-Management - Entwicklung von Anwendungen und Systemen*. dpunkt-Verlag, Heidelberg, 1997.

[29] B. Kahlbrandt. *Software-Engineering: objektorientierte Software-Entwicklung mit der Unified Modeling Language*. Springer, Heidelberg, Berlin, New York, 1998.

[30] G. Knolmayer; P. Mertens; A. Zeier. *Supply Chain Management Based on SAP Systems - Order Management in Manufacturing Companies*. Springer, Berlin, 2002.

[31] S. Kreipl; M. Pinedo. Planning and scheduling in supply chains: An overview of issues in practice. *Production and Operations Management*, 13(1), 77–92, 2004.

[32] F. Leymann; D. Roller. Modeling business processes with BPEL4WS. *Information Systems and E-Business Management*, 4, 265–284, 2005.

[33] G. Matthiessen; M. Unterstein. *Relationale Datenbanken und SQL: Konzepte der Entwicklung und Anwendung*. ADDISON-WESLEY, München, 2003.

[34] D.C. McFarlane; S. Bussmann. Developments in holonic production planning and control. *Production Planning and Control*, 11(6), 522–536, 2000.

[35] D.C. McFarlane; S. Bussmann. Holonic manufacturing control: Rationales, developments and open issues. M. S. Deen (Hrsg.), *Agent-Based Manufacturing - Advances in the Holonic Approach*, Springer-Verlag, Heidelberg, 303–326, 2003.

[36] P. Mertens. *Integrierte Informationsverarbeitung 1: Operative Systeme in der Industrie*. Gabler, Wiesbaden, 2001.

[37] P. Mertens; J. Griese. *Integrierte Informationsverarbeitung 2: Planungs- und Kontrollsysteme in der Industrie*. Gabler, Wiesbaden, 2002.

[38] M.D. Mesarović; D. Macko; Y. Takahara. *Theory of Hierarchical, Multilevel, Systems*. Academic Press, New York, London, 1970.

[39] L. Mönch. *Agentenbasierte Produktionssteuerung komplexer Produktionssysteme*. Gabler, DUV-Verlag, Wiesbaden, 2006.

[40] E. Newcomer; G. Lomow. *Understanding SOA with Web Services*. Pearson Education Inc., Upper Saddle River, 2005.

[41] M. Obitko; V. Mařík. Ontologies for multi-agent systems in manufacturing domain. *Proceedings of the 3rd International Workshop on Industrial Applications of Holonic and Multi-Agent-Systems*, Aix en Provence, 2002.

[42] M. P. Papazoglou; W.-J. van den Heuvel. Service oriented architectures: Approaches, technologies and research issues. *Erscheint in VLDB Journal*, 2007.

[43] A. Puder; K. Römer. *Middleware für verteilte Systeme. Konzepte und Implementierungen anhand der CORBA-Plattform MICO.* dpunkt.verlag, Heidelberg, 2001.

[44] M. Reichert; D. Stoll. Kompositition, Choreographie und Orchestrierung von Web Services - ein Überblick. *EMISA Forum*, 24(2) 2004.

[45] F. Rump. *Geschäftsprozeßmanagement auf der Basis ereignisgesteuerter Prozeßketten.* Teubner Verlag, Stuttgart, Leipzig, 1999.

[46] A.-W. Scheer. *Wirtschaftsinformatik - Referenzmodelle für industrielle Geschäftsprozesse.* 6. Auflage. Springer-Verlag, Berlin, Heidelberg, New York, 1995.

[47] A.-W. Scheer. *ARIS - Modellierungsmethoden, Metamodelle, Anwendungen.* Springer, Heidelberg, Berlin, New York, 4. Auflage , 2001.

[48] A.-W. Scheer; K. Schneider. ARIS - Architecture of Integrated Information Systems. P. Bernus; K. Mertins; G. Schmidt (Hrsg.), *Handbook on Architecture of Information Systems*, Springer, 605–623, 2006.

[49] D. Schmidt; M. Stal; H. Rohnert; F. Buschmann. *Patternorientierte Softwarearchitektur: Muster für nebenläufige und vernetzte Objekte.* dpunkt.Verlag, Heidelberg, 2002.

[50] M. Schumann; H. Schüle; U. Schumann. *Entwicklung von Anwendungssystemen: Grundzüge eines werkzeuggestützen Vorgehens.* Springer, Berlin, Heidelberg, New York, 1994.

[51] J. Siedersleben. *Moderne Softwarearchitektur - Umsichtig planen, robust bauen mit Quasar.* dpunkt.Verlag, Heidelberg, 2004.

[52] J. Siedersleben. SOA revisited: Komponentenorientierung bei Systemlandschaften. *WIRTSCHAFTSINFORMATIK*, 49, 110–117, 2007.

[53] E.J. Sinz. Architektur von Informationssystemen. P. Rechenberg; G. Pomberger (Hrsg.), *Handbuch der Informatik*, 3. Auflage, Hanser, München, 1035–1047, 2002.

[54] S. Smith; M. Becker. An ontology for constructing scheduling systems. *Working Notes of the 1997 Symposium on Ontological Engineering.* AAAI Press, 1997.

[55] H. Österle; W. Brenner; K. Hilbers. *Unternehmensführung und Informationssystem: der Ansatz des St. Galler Informationssystem-Managements*. 2. Auflage. Teubner, Stuttgart, 1992.

[56] A. Tanenbaum; M. Steen. *Verteilte Systeme. Grundlagen und Paradigmen*. Prentice Hall, München, 2003.

[57] M. Uschold; M. King; S. Moralee; Y. Zorgios. The enterprise ontology. *The Knowledge Engineering Review*, 13(1), 31–89, 1998.

[58] H. Van Brussel; J. Wyns; P. Valckenaers; L. Bongaerts; P. Peeters. Reference Architecture for Holonic Manufacturing Systems: PROSA. *Computers in Industry, Special Issue on Intelligent Manufacturing Systems*, 37(3), 255–274, 1998.

[59] G. Vossen; J. Becker (Hrsg.). *Geschäftsprozeßmodellierung und Workflowmanagement: Modelle, Methoden, Werkzeuge*. International Thompson Publishing Company, Bonn, Albany, 1996.

[60] F. Wall. Kostenwirkungen der Prozessorientierung. *WIRTSCHAFTSINFORMATIK*, 42(3), 210–221, 2000.

[61] G. Weiss. *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. MIT Press, Cambridge, Massachusetts, 1999.

[62] M. Wooldridge. *An Introduction to Multiagent Systems*. Wiley, Chichester, 2002.

[63] R. Zacharias. Serviceorientierung: der OO-König ist tot, es lebe der SOA-König. *OBJEKTspektrum*, 2, 43–52, 2005.

[64] S. Zelewski; R. Schütte; J. Siedentopf. Ontologien zur Repräsentation von Domänen. G. Schreyögg (Hrsg.), *Wissen im Unternehmen - Konzepte, Maßnahmen, Methoden*, Erich Schmidt Verlag, Berlin, 183–221, 2001.