

Bachelorarbeit

im Fachgebiet Wirtschaftsinformatik
am Lehrstuhl für Wirtschaftsinformatik und Interorganisationssysteme

Enterprise Application Integration - Grundlagen, Methoden und Techniken

Themensteller: Prof. Dr. Stefan Klein
Betreuer: Dr. Bernd Schneider

vorgelegt von: Angela Schwering
Steinfurter Str. 77
48149 Münster
0251-83875060
wiansc@wi.uni-muenster.de

Abgabetermin: 2001-09-18

Inhaltsverzeichnis

Abbildungsverzeichnis	III
Abkürzungsverzeichnis	IV
1 Einleitung	1
2 Grundlagen	3
2.1 Integration in der Wirtschaftsinformatik	3
2.2 Erklärung grundlegender Begriffe	6
2.3 Betriebswirtschaftliche Motivation zur Einführung von EAI	7
3 Integrationsarten	9
3.1 Darstellungsintegration	10
3.2 Datenintegration	12
3.3 Funktionsintegration	14
3.4 Bewertung der Methoden	17
4 Middlewarebasierte Architekturen	18
4.1 Nachrichtenorientierte Middlewarearchitektur	21
4.2 Objektorientierte Middlewarearchitektur	28
4.3 Transaktionsorientierte Middlewarearchitektur	35
5 Zusammenfassung und Ausblick	39
6 Literatur	41

Abbildungsverzeichnis

Abb. 1:	Herausforderung Anwendungsintegration	2
Abb. 2:	Ausprägungen der Integrierten Informationsverarbeitung	4
Abb. 3:	Entwicklung der Aufgaben in der Wirtschaftsinformatik	4
Abb. 4:	Middleware-Evolution	6
Abb. 5:	Stovepipes	7
Abb. 6:	Verschiedene Integrationsmethoden	10
Abb. 7:	Darstellungsintegration	11
Abb. 8:	Datenintegration	13
Abb. 9:	Funktionsintegration	14
Abb. 10:	Vergleich der verschiedenen Integrationsmethoden	18
Abb. 11:	Aufbau einer Nachricht	22
Abb. 12:	Message Passing	23
Abb. 13:	Message Queuing	24
Abb. 14:	Request/Reply	25
Abb. 15:	Kommunikationsablauf von Request/Reply	26
Abb. 16:	Publish/Subscribe	27
Abb. 17:	Queue Manager	28
Abb. 18:	Object Management Architecture	30
Abb. 19:	Object Request Broker	31
Abb. 20:	Enterprise Java Bean	32
Abb. 21:	EJB Architektur	33
Abb. 22:	Vergleich der Modelle	35
Abb. 23:	Distributed Transaction Processing (DTP) Modell	38

Abkürzungsverzeichnis

ACID	atomic, consistent, isolated, durable
ADO	Active Data Object
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
B2B	business to business
B2C	business to consumer
CCM	CORBA Component Model
CICS	Customer Information Control System
COM+	Component Object Model+
CORBA	Common Object Request Broker Architecture
DBMS	Database Management System
DII	Dynamic Invocation Interface
DNA	Distributed interNet Application Architecture
DOT	Distributed Object Technology
DTM	Distributed Transaction Monitor
DTP	Distributed Transaction Processing
DTS	Distributed Transaction Service
EAI	Enterprise Application Integration
EBCDI	Extended Binary Coded Decimal Interchange
EDI	Electronic Data Interchange
EJB	Enterprise Java Bean
ERP	Enterprise Resource Planning
FIFO	Fist In First Out
GUI	Grafical User Interface
HTML	Hyper Text Markup Language
IDL	Interface Definition Language
IK	Information und Kommunikation
IMS	Information Management System
IT	Informationstechnologie bzw. Information Technology
IV	Informationsverarbeitung
J2EE	Java 2 Plattform, Enterprise Edition
MOM	Message Oriented Middleware
MTS	Microsoft Transaction Server
ODBC	Open Database Connectivity
OLE DB	Object Linking and Embedding Database
OMA	Object Management Architecture
OMG	Object Management Group
OODB	Objektorientierte Datenbank
ORB	Object Request Broker
OTM	Object Transaction Monitor
RMI	Remote Method Invocation
RPC	Remote Procedure Call
SQL	Structured Query Language
TCP/IP	Transmission Control Protocol / Internet Protocol
TPM	Transaction Processing Monitor

1 Einleitung

Unternehmensorganisationen sind derzeit einem starken Wandel unterzogen. Während noch vor wenigen Jahren die Integration einzelner Arbeitsplätze in den Geschäftsablauf im Vordergrund stand - Büroautomation, kooperierendes Arbeiten unterschiedlicher Bereiche innerhalb einer Firma - hat sich die Dimension heute geändert. Durch härteren Wettbewerb und zunehmende Dynamik der Märkte getrieben, suchen Unternehmen nach effizienteren und umfassenderen Geschäftslösungen. Dabei werden Grenzen von Unternehmungen verschoben, neu definiert oder verschwimmen sogar. Interorganisatorische Beziehungen müssen optimiert und in ein Unternehmensnetzwerk integriert werden.¹⁾

Um im Wettbewerb bestehen zu können, müssen sich Unternehmen differenzieren. Neue Kanäle zur Interaktion mit dem Kunden wie das Internet verlangen zeitnahe Reaktion auf Kundenbedürfnisse. Flexibilität gegenüber Kundenwünschen, niedrige Kosten und perfekte Qualität sind notwendige Fähigkeiten, um im Markt konkurrenzfähig zu bleiben. Eine umfassende Integration eröffnet neue Potenziale zur verbesserten Planung und Steuerung des operativen Geschäfts und zur besseren Koordination über Unternehmensgrenzen hinweg. Die Einbindung von neuen Märkten wie E- und M-Commerce in das bestehende System ist eine Voraussetzung, um mit dem verschärften Wettbewerb auch in Zukunft mithalten zu können.

Versuche, eine einheitliche Integrationsplattform zu schaffen, scheiterten an einem fehlenden Konzept für die Gesamtintegration. Standards für eine Integration waren nicht vorhanden. Hersteller von ERP-Software machten den ersten Schritt in diese Richtung. Sie entwickelten Plattformen, die bereits weite Teile eines Unternehmens integrieren. Da die Schnittstellen zur Außenwelt jedoch herstellerabhängig sind, bieten sie nicht genug Flexibilität zur Integration von Fremd- und Altsystemen.

Enterprise Application Integration (EAI) ist eine Strategie, mit der Gesamtintegration schnell und koordiniert realisiert werden kann. EAI heißt, bestehende Anwendungen mit Schnittstellen zu versehen, damit sie über eine standardisierte Middleware integrationsfähig

1) Klein, S. (1996), S. 1-2.

werden. EAI heißt aber auch, Geschäftsprozesse neu zu definieren und bisher getrennte Geschäftsprozesse miteinander zu koppeln.

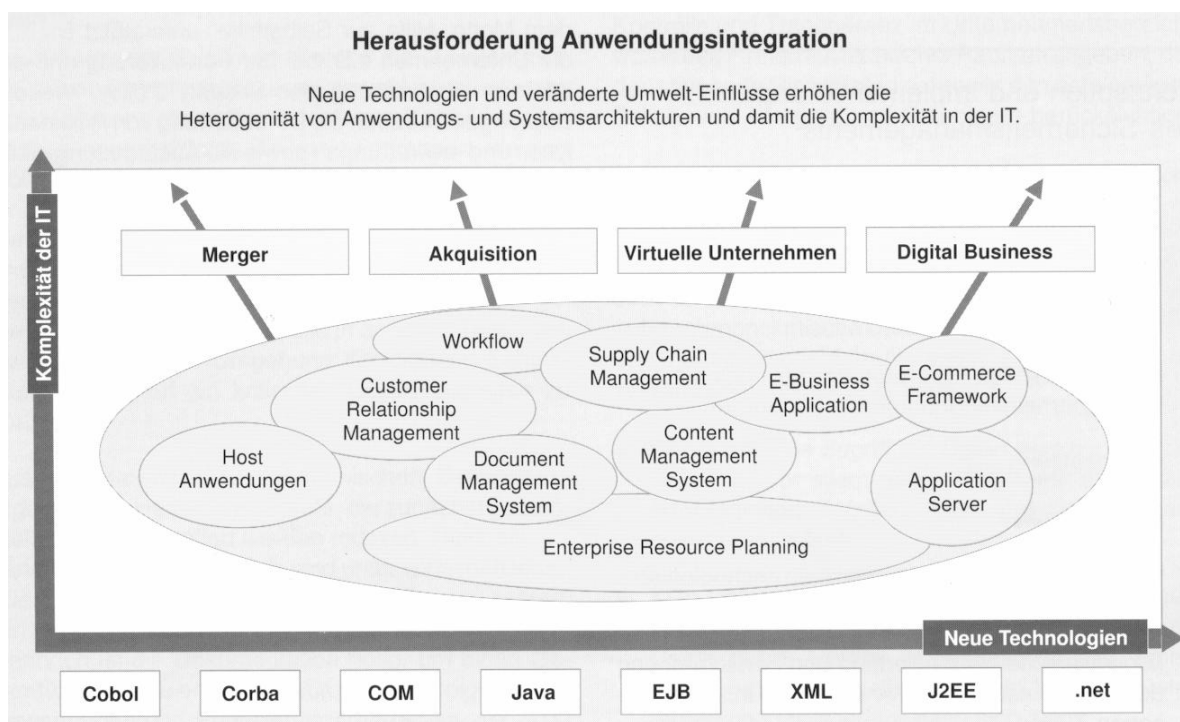


Abb. 1: Herausforderung Anwendungsintegration²⁾

Diese Arbeit soll das Konzept Enterprise Application Integration erläutern und verschiedene technische Realisierungsmöglichkeiten vorstellen.

Im zweiten Kapitel werden verschiedene Ausprägungen von Integration behandelt und ein historischer Überblick über die sich wandelnden Aufgabenstellungen der Integration gegeben. Zentrale Begriffe wie EAI und Middleware werden erklärt und aus der betriebswirtschaftlichen Perspektive betrachtet. In Kapitel 3 werden verschiedene Ansatzmöglichkeiten von EAI auf der Daten-, Funktions- und Präsentationsebene dargestellt und anschließend die unterschiedlichen Integrationsarten miteinander verglichen und bewertet. Unterschiedliche Strukturmöglichkeiten für das heterogene Erscheinungsbild von Middlewareprodukten werden im vierten Kapitel vorgeschlagen. Danach werden die verschiedenen Middlewarearchitekturen unter technischen Gesichtspunkten gegeneinander abgegrenzt.

2) Walter, J. (2001), S. 6.

2 Grundlagen

Zuerst wird der Begriff Integration nach dem allgemeinen Verständnis in der Literatur mit ihren verschiedenen Ausprägungen erläutert. In einem historischen Überblick werden die sich wandelnden Aufgabenstellungen der Integration unter dem Aspekt der verschiedenen Ausprägungen dargestellt. Anschließend wird Integration aus der technischen Perspektive beleuchtet und unterschiedliche Entwicklungsstufen von Middleware vorgestellt. Zum Schluss werden die Begriffe Enterprise Application Integration und Middleware näher erläutert und deren Potentiale betriebswirtschaftlich bewertet.

2.1 Integration in der Wirtschaftsinformatik

Mertens definiert Integration als die Wiederherstellung eines Ganzen. "In der Wirtschaftsinformatik ist Integration als Verknüpfung von Menschen, Aufgaben und Technik zu einer Einheit zu verstehen."³⁾

Mertens unterscheidet verschiedene Ausprägungen der Integrierten Informationsverarbeitung: Integrationsgegenstand, Integrationsrichtung, Integrationsreichweite und Automationsgrad (s. Abb. 2).

Gegenstand der Integration sind Daten, Funktionen, Prozesse, Methoden oder Programme. Hinsichtlich der Integrationsrichtung wird zwischen der horizontalen Integration, der Integration innerhalb eines Prozesses, und der vertikalen Integration, der Integration von Planungs- und Kontrollsystemen durch Administrations- und Dispositionssysteme, unterschieden. Bezüglich der Integrationsreichweite definiert Mertens drei Ausprägungen: Die Bereichsintegration beinhaltet die Integration innerhalb eines Sektors oder Prozesses. Unter der innerbetrieblichen Integration versteht man die Integration innerhalb eines Unternehmens, während zwischenbetriebliche Integration über Unternehmensgrenzen hinausgeht. Der Automationsgrad trennt voll- und teilautomatische Verkettung von Modulen.

3) Mertens, P. (2000), S. 1.

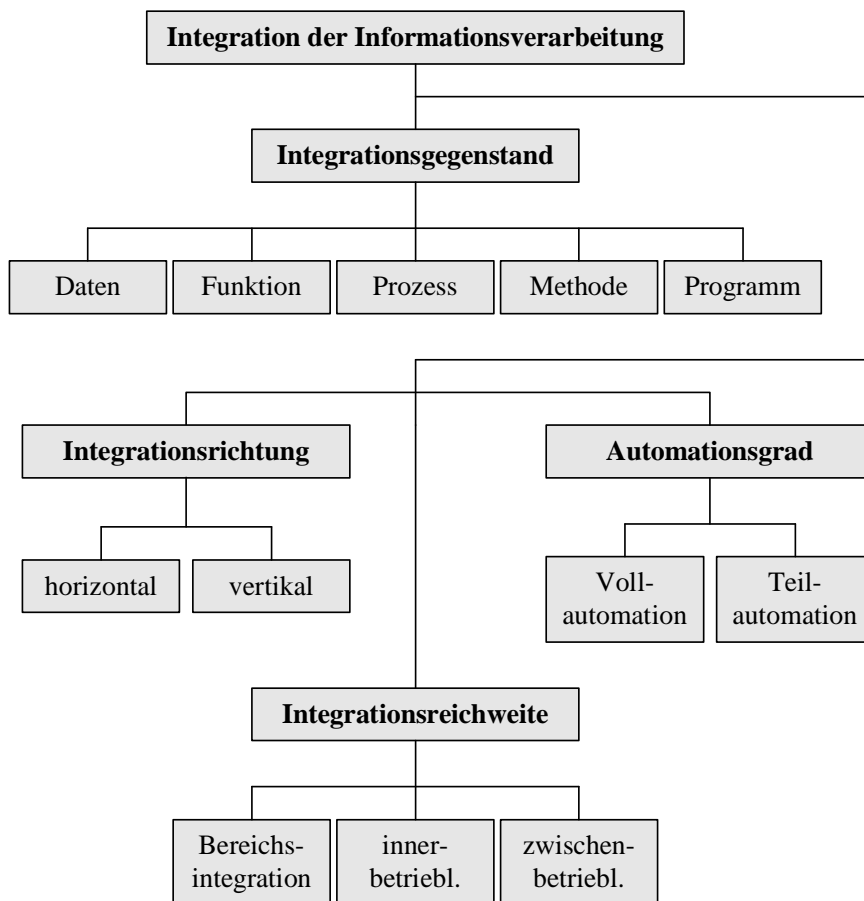


Abb. 2: Ausprägungen der Integrierten Informationsverarbeitung⁴⁾

Im Laufe der Geschichte der Wirtschaftsinformatik haben sich die Aufgabenstellung der Integration weiterentwickelt und das Ausmaß der oben aufgeführten Ausprägungen geändert.



Abb. 3: Entwicklung der Aufgaben in der Wirtschaftsinformatik

4) In Anlehnung an Mertens, P. (2000), S. 2.

Die ersten Ansätze der Integration sind in den späten 70er Jahren zu finden. Neue Betriebsformen wie Echtzeit- und Dialogbetrieb und bessere Speicher- und Verarbeitungskapazitäten der Rechner erlauben eine integrierte Datenverarbeitung mit dezentraler Ein- und Ausgabe. Durch die neue DV-Technik wird eine durchgängigere, am Arbeitsablauf orientierte Aufgabenintegration realisiert. In den 80er Jahren wird der Schritt von reiner Datenverarbeitung zur integrierten Informationsverarbeitung vollzogen. Erstmals werden Anwendungen nicht mehr isoliert, sondern die Interdependenzen von Anwendungssystemen betrachtet. Erhebliche Leistungssteigerungen von Computern führen zu einer Dezentralisierung und Individualisierung der Datenverarbeitung. Durch IK-Techniken findet eine Vernetzung der dezentralen Rechner statt.⁵⁾ In den späten 80er Jahren werden nun auch die strategischen Potentiale der Integration erkannt. Die Betrachtungsebene breitet sich auf die unternehmensweite Systemlandschaft aus - der Begriff des Information Engineering wird geprägt. IK-Technik wird integrierter Bestandteil der Organisation. Die reine softwaretechnische Betrachtung von IV-Systemen hat sich 1995 in eine betriebswirtschaftlich geprägte Sicht gewandelt. Integration bedeutet nun, Geschäftsprozesse und Informationsverarbeitung parallel und integriert zu planen. IK-Technik wird nicht mehr als reine Unterstützung eines vorgegebenen Geschäfts verstanden, sondern Geschäftsmodelle und Organisation von Unternehmen werden reorganisiert, um IK-Technik umfassender zu integrieren. Diese neue Dimension der Integration bietet Unternehmen neue Potentiale, aber auch neue Risiken. Netzwerke, die über Unternehmensgrenzen hinweg operieren, stellen hochkomplexe Anforderungen an deren Verwaltung. Kleine Änderungen in einer integrierten Welt können unbeabsichtigte Folgen haben. Diese Komplexität zu verwalten ist eine der wesentlichen, neuen Aufgaben von Integration.

Die Aufgaben der Integration haben sich rasant verändert. Während Datenintegration in den 80er Jahren durch Vernetzung der Rechner realisiert wurde, können integrierte Geschäftslösungen nicht mehr durch einfache Mechanismen realisiert werden. Die folgende Abbildung zeigt die Entwicklung der Technik bis zur heutigen Middleware.

5) In Anlehnung an Teubner, R. (1999), S. 47-52.

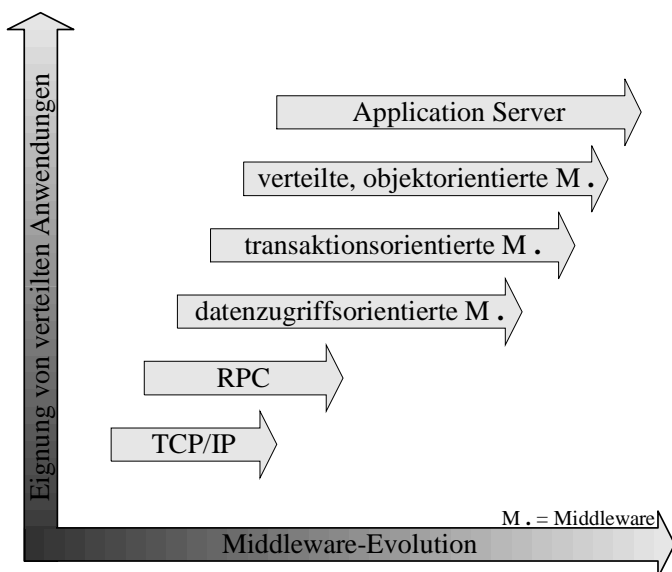


Abb. 4: Middleware-Evolution⁶⁾

TCP/IP ist ein einfaches Protokoll, um über ein Netzwerk zu kommunizieren. RPC ermöglicht den Methodenaufruf auf entfernten Systemen. Mit TCP/IP und RPCs kann die Interaktion von Anwendungen und somit die Grundfunktionalität von Middleware realisiert werden. Als erste tatsächliche Middleware wird die datenzugriffsorientierte Middleware bezeichnet. Jedoch ist sie auf die Datenebene beschränkt. Weiterentwickelte Funktionalität werden durch die transaktionsorientierte Middleware, die objektorientierte Middleware und in dem Produkt Application Server realisiert. Die technischen Eigenschaften werden in Kapitel 4 erläutert.

2.2 Erklärung grundlegender Begriffe

Enterprise Application Integration steht für umfassende Geschäftslösungen und das Konzept zur unternehmensweiten Integration von verschiedenartigen Anwendungen über eine gemeinsame Middleware. Sie ist also nicht als ad hoc Lösung, sondern als Strategie zu verstehen, die die Komplexität heterogener Systemlandschaften und Technologien reduziert und bewältigt. Funktionalität, Daten und Darstellung von Anwendungen werden gekapselt, sodass ihre Leistungen als Dienste anderer Software zur Verfügung gestellt werden können. Gleichzeitig bietet eine EAI-Plattform einen standardisierten Mechanismus zum

6) In Anlehnung an Ruh, W.; Maginnis, F.; Brown, W. (2000), S. 86.

Informationsaustausch zwischen IV-Systemen mit dem Ziel, jede mögliche Anwendung schnell und einfach in das System integrieren zu können.⁷⁾

Middleware ist eine anwendungsunabhängige, flexible Software, die zwischen Betriebssystemen und Verteilten Systemen einzuordnen ist. Sie stellt grundlegende Dienste zur Kommunikation zwischen Anwendungen über Netzwerk und Internet bereit mit dem Ziel, die Interaktion zwischen verteilten Softwaremodulen zu vereinfachen. Middleware bietet eine einheitliche Schnittstelle zu heterogenen Computerarchitekturen, Betriebssystemen, Programmiersprachen und Netzwerktechnologien, sodass die Entwicklung und die Verwaltung von Anwendungen erheblich vereinfacht werden.⁸⁾

2.3 Betriebswirtschaftliche Motivation zur Einführung von EAI

EAI ist "ein wesentlicher strategischer Baustein für die zukünftige Wettbewerbsposition vieler Unternehmen".⁹⁾ Geschäftsprozessen müssen effizienter gestaltet und koordiniert werden.

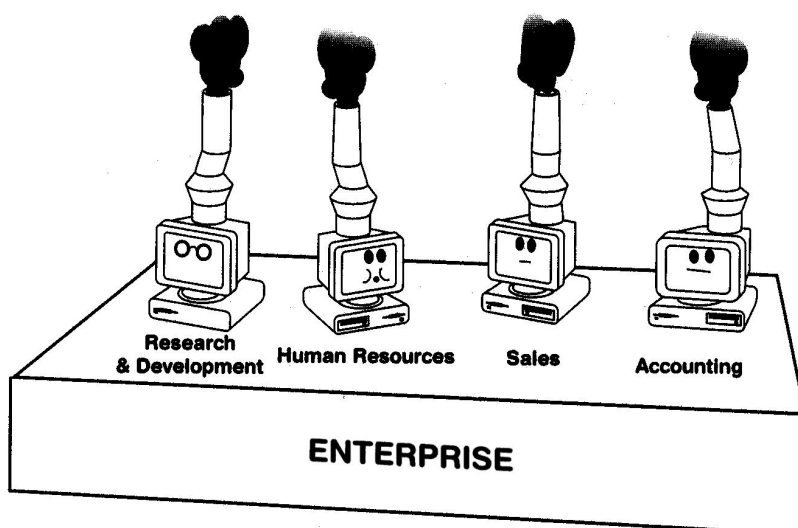


Abb. 5: Stovepipes¹⁰⁾

7) In Anlehnung an Ruh, W.; Maginnis, F.; Brown, W. (2000), S. 2-3.

8) In Anlehnung an Geihs, K. (2001), S. 24.

9) Bernotat, J.; Hoch, D.; Laartz, J.; Scherdin, A. (2001), S. 17.

10) In Anlehnung an Linthicum, D. (1999), S. 11.

Häufig sind Informationen verteilt über viele einzelne Anwendungen, sogenannte "Stovepipe Applications", gespeichert. Stovepipes - in Abb. 5 als nebeneinander angeordnete Ofenrohre bildhaft dargestellt - sind isolierte Anwendungen, die typischerweise für eine bestimmte Geschäftsfunktion innerhalb einer Organisationseinheit entwickelt wurden.¹¹⁾

Um beispielsweise eine umfassende Kundenbetreuung gewährleisten zu können, müssen diese Einzelanwendungen integriert werden, sodass relevante Informationen überall verfügbar sind und Medienbrüche vermieden werden. Ein weiterer wesentlicher Faktor für wettbewerbsfähige Unternehmen ist nicht nur die Beziehung zum Kunden, sondern auch zum Lieferanten. Durch Informationsaustausch mit Organisationen außerhalb des Unternehmens können Aktivitäten entlang der Versorgungskette effektiver koordiniert und zusätzliche Dienste wie Order Tracking angeboten werden.

Aber nicht nur im B2C- und B2B-Bereich besteht ein hoher Integrationsbedarf - auch der Bedarf an verbesserten unternehmensinternen Prozessen treibt EAI voran. Durch die weltweite Konkurrenz im Internetzeitalter ist es wichtiger denn je für ein Unternehmen, umfassend und schnell reagieren zu können. Durch EAI-Technologie kann ein stringenter Informationsfluss zwischen Datenspeicher und Stovepipe Applications realisiert werden. Eine einheitliche Datenbasis dient zur Entscheidungsunterstützung in der Unternehmung und macht komplexe Situationen transparent. Geschäftsprozesse werden automatisiert und redundante Daten eliminiert. Webinterfaces machen Informationen den Mitarbeitern weltweit zugänglich.

Des Weiteren stellt EAI eine Plattform bereit, in die Anwendungen einfach und schnell integriert werden können. Zum einen können so bereits getätigte Investitionen in Legacy Systeme und Packaged Applications sinnvoll genutzt werden¹²⁾, zum anderen kann der Aufwand für die Anwendungsentwicklung reduziert werden. Die Funktionalität einer Anwendung wird an das vorhandene Front-End System des EAI angebunden, wodurch der Zugang zu dieser Applikation über Web und andere Anwendungen ermöglicht wird, ohne eine eigene Benutzeroberfläche programmieren zu müssen. Eine zentrale Middleware

11) In Anlehnung an Ruh, W.; Maginnis, F.; Brown, W. (2000), S. 5.

12) In Anlehnung an Scheibenpflug, M. (2001), S. 60-61.

übernimmt die aufwändige Integration mit anderer Hard- und Software. Das reduziert den Entwicklungsaufwand und führt zu einer schnelleren Produkteinführung.

Ein effektiver Einsatz von EAI ist jedoch auch mit hohem Aufwand verbunden. Chaotische Architekturen mit verschiedensten Technologien vom Unix Mainframe bis zur Windows Workstation haben sich entwickelt. Dokumentationen für Programme existieren zumeist nicht. Häufig fehlen Mitarbeiter mit den nötigen Kenntnissen über die aktuellen Technologien. Die zunehmende Integration von Anwendungen, im Besonderen die Internettechnologie, stellt hohe Anforderungen an die Sicherheit. Es gilt also, die Komplexität zu managen, die Belegschaft auszubilden und den wachsenden Sicherheitsanforderungen gerecht zu werden. Bevor diese Voraussetzungen nicht geschaffen sind, ist ein erfolgreicher Einsatz von EAI nicht möglich.

3 Integrationsarten

Integration ist seit jeher ein vieldiskutiertes Thema in der IT-Branche. Während früher der Schwerpunkt auf Hardwareintegration mit einer von Grund auf neu entwickelten Software lag, stehen heute die Kombination von Hardware und komplexer Software zu einem System im Vordergrund. Dominiert wird der Prozess von der Software.

Integration kann grundsätzlich an zwei verschiedenen Kriterien gemessen werden: der Integrationsbreite und der Integrationstiefe.

Die Integrationsbreite bezeichnet die Integration der IV-Systeme in einem Geschäftsprozess. Ist Software der gesamten Wertschöpfungskette von der Produktion bis zur Fakturierung integriert, so spricht man von einer hohen Integrationsbreite, hingegen von einer geringen, wenn beispielsweise nur Programme der Ersatzteiledisposition integriert wurden.

Unter Integrationstiefe versteht man den Grad der semantischen Integration.¹³⁾ Hier werden drei Ebenen zur Integration von Software unterschieden: die Daten-, Funktions- und die Darstellungsintegration (vgl. Abb. 6).

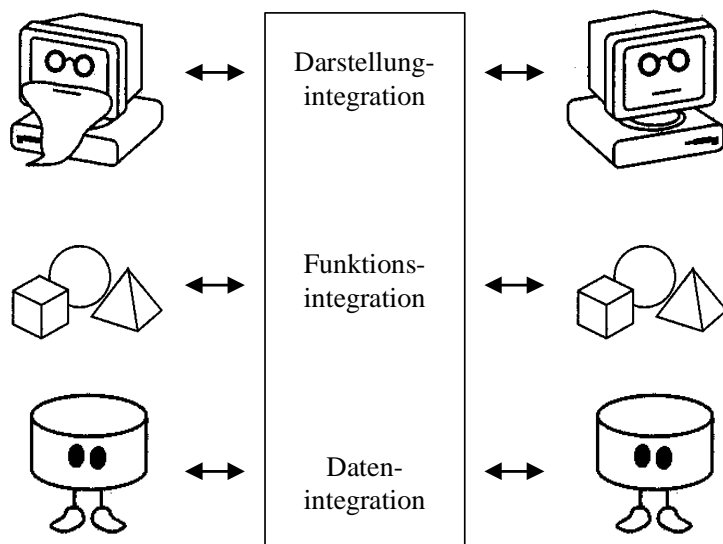


Abb. 6: Verschiedene Integrationsmethoden¹⁴⁾

Zunächst wird die Darstellungsintegration, die Methode der geringsten Integrationstiefe, ausführlich behandelt, danach wird näher auf die Daten- und zum Schluss auf das umfassendste Konzept, die Funktionsintegration, eingegangen.

An alle Arten der Integration werden folgende Anforderungen gestellt: Sie müssen sowohl mit einem akzeptablen Zeitaufwand durchführbar sein, als auch für verschiedene Konfigurationen wiederverwendbar sein.

3.1 Darstellungsintegration

Die Integration auf der Darstellungsebene ist die einfachste Integrationsmaßnahme. Verschiedene Anwendungen präsentieren sich dem Benutzer mit einer einheitlichen Anwendungsoberfläche, die über die Darstellungslogik der Anwendungen oder Legacy

13) In Anlehnung an Winkeler, T.; Westphal, L.; Raupach, E. (2001), S. 8.

14) In Anlehnung an Linthicum, D. (1999), S. 19.

Systeme kommunizieren. In der Abb. 7 wird durch die grau hinterlegten Felder die Ansatzpunkte der Darstellungsintegration bildlich illustriert.

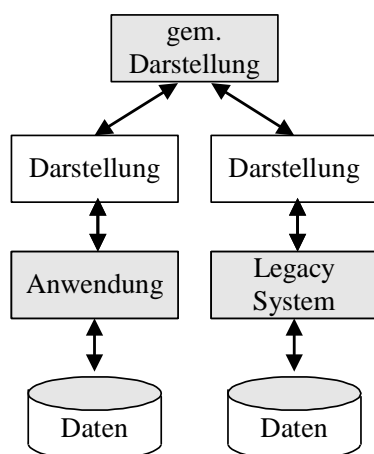


Abb. 7: Darstellungsintegration¹⁵⁾

Anhand von Screen-Scraping Tools werden alte Oberflächen kopiert und in der gemeinsamen Darstellung abgebildet. So können alte Systeme unter Einsatz von neuen Technologien wie XML sehr einfach mit neuen und modernen GUIs versehen werden. Da Integration auf dieser Ebene keine umfangreichen Kenntnisse der Softwareinterna erfordert, ist sie relativ schnell und billig durchführbar. Allerdings wurden Benutzeroberflächen nicht dazu entwickelt, Daten und Funktionslogik anderen Anwendungen zur Verfügung zu stellen, daher sind die Möglichkeiten sehr begrenzt.

Um Darstellungsintegration erfolgreich anwenden zu können, müssen die zu integrierenden Anwendungsoberflächen genau analysiert und deren Informationen identifiziert werden. Es ist wichtig, den Inhalt der Daten zu verstehen, um fehlerhafte Repräsentation in der zu entwickelnden Darstellungsumgebung zu vermeiden. Anschließend werden die einzelnen Bildschirme mit ihren Informationen katalogisiert und die Position von gleichen Daten erfasst, um Konsistenz zu gewährleisten und gesuchte Informationen schnell zu lokalisieren. Auf der nächsten Stufe müssen die Informationen der Bildschirme extrahiert werden. Dies kann entweder statisch realisiert werden, d. h. durch feste Angabe von Koordinaten, oder durch eine dynamische Extraktion. Beim letzteren kann durch Bedingungen und Logik

15) In Anlehnung an Ruh, W.; Maginnis, F.; Brown, W. (2000), S. 23.

automatisch auf Änderungen in der Darstellung reagiert werden und dieses Verfahren ist somit flexibler.¹⁶⁾

In der Darstellungsintegration können zwei verschiedene Methoden angewendet werden: Screen-as-Data interpretiert Bildschirme als rohen Datenfluss, wohingegen Screen-as-Object die Benutzeroberfläche als Objekte behandelt. Screen-as-Data liest den Datenstrom aus, analysiert, identifiziert und konvertiert die Informationen und verarbeitet sie. Es ist einfach zu realisieren, berücksichtigt aber im Gegensatz zum Screen-as-Object Konzept nur Daten, aber nicht die zugehörigen Methoden. Bei letzterem werden die Bildschirminformationen in Anwendungsobjekte wie Java- oder CORBA-Objekte übersetzt. Dadurch wird die Entwicklung komplizierter und teurer.¹⁷⁾

Die Darstellungsintegration wird angewendet, um beispielsweise für textbasierte Programme ein Java- oder HTML-basiertes GUI zu entwickeln oder verschiedene Mainframeanwendungen mit anderen Applikationen wie SAP/R3 über ein einheitliches Interface zu integrieren. Die Anforderungen variieren von einem einfachen GUI-Aufsatz bis zur Implementierung von zusätzlicher Logik zur Integration von Daten und Funktionen.¹⁸⁾

3.2 Datenintegration

Die Datenintegration integriert Anwendungen auf der Ebene der Datenstrukturen, indem sie den Datenaustausch zwischen verschiedenen Datenquellen wie Data Mining Systemen und Datenbanken ermöglicht. Fremde Anwendungen können über die EAI-Middleware neue Datensätze erzeugen, speichern und modifizieren. Die Abb. 8 veranschaulicht durch die grau hinterlegten Felder, dass die Middleware direkten Datenzugriff vorbei an der Software der Anwendung bzw. dem Legacy System hat.

16) In Anlehnung an Linthicum, D. (1999), S. 80-86.

17) In Anlehnung an Linthicum, D. (1999), S. 86-87.

18) In Anlehnung an Ruh, W.; Maginnis, F.; Brown, W. (2000), S. 22-24.

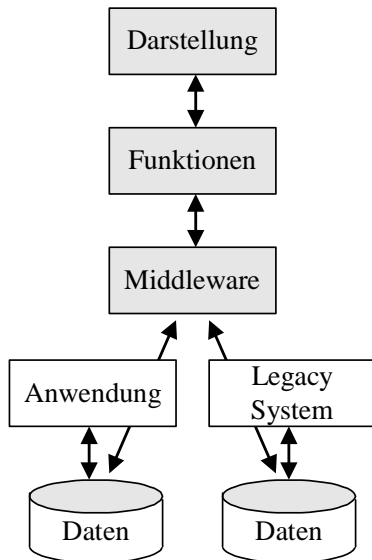


Abb. 8: Datenintegration¹⁹⁾

Datenintegration wird angewendet, um Daten aus verschiedenen Quellen für Analysen oder Entscheidungsprobleme zusammenzufassen. Daten können zentral in einem Data Warehouse gespeichert werden, um so verschiedenen Anwendungen Zugriff zu einheitlichen Daten zu ermöglichen. Über die Middleware können Daten aus einer Quelle ausgelesen und in einer anderen wiederverwendet werden. Redundanzen werden eliminiert und Daten synchronisiert.

Abhängig von unterschiedlichen Anforderung an Datenaktualität, Integrationskomplexität und gegebenen Zugriffsmöglichkeiten auf die Datenquellen gibt es verschiedene Realisierungsmöglichkeiten einer Datenintegration. Die älteste Methode ist der Transfer von Daten, die meist in Dateien gespeichert sind, über ein Batch File, wobei hier weitgehend Unabhängigkeit von der Art der Datenquelle besteht. Open Database Connectivity war der erste weit verbreitete Standard, um auf relationale Datenbanken zuzugreifen. Eine standardisierte API abstrahiert von verschiedenen Datenbanken. Durch die Heterogenität der zu integrierenden Datenquellen werden Hilfsprogramme, sog. Wrapper, benötigt, die Daten bzw. Dateien in verschiedene Formate konvertieren, zum Beispiel von ASCII in EBCDI-Code oder eine einfache Währungsumrechnung. Eine umfassende Datenintegration ist durch eine Middleware realisierbar. Die Middleware bietet eine Laufzeitumgebung, die durch Konnektoren den Zugriff auf die Datenquellen ermöglicht, Anfragen weiterleitet und

19) In Anlehnung an Ruh, W.; Maginnis, F.; Brown, W. (2000), S. 24.

entsprechende Ergebnisse zurückliefert. Ziel ist es, den Datenaustausch in Echtzeit zu ermöglichen, um die Reaktionsgeschwindigkeit eines Unternehmens zu erhöhen.²⁰⁾

3.3 Funktionsintegration

Die Funktionsintegration ist die wichtigste Integrationsmethode. Sie bezeichnet die Integration von Anwendungen, von Funktionen oder Objekten, auf der Code-Ebene. Einer Drittanwendung wird so ermöglicht, über die Middleware auf die Logik anderer Anwendungen oder Legacy Systeme zuzugreifen, um so Operationen auszuführen (s. Abb. 9). Durch Wiederverwenden der existierenden Geschäftslogik wird der Entwicklungsaufwand verringert und zusätzliche Fehler vermieden.

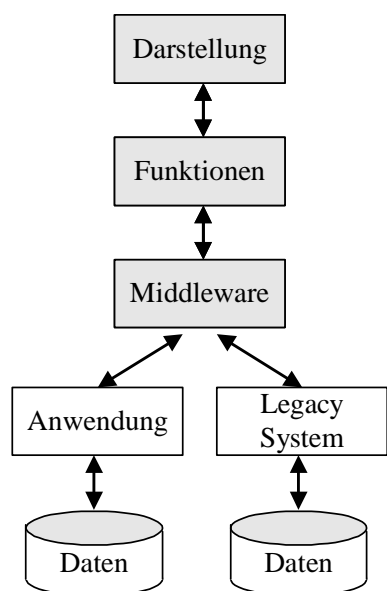


Abb. 9: Funktionsintegration²¹⁾

Funktionen aus Anwendungen in Verteilten Systemen können mit Remote Procedure Calls (RPCs) über ihre Signatur aufgerufen werden, jedoch ist dieses Konzept recht aufwändig in der Softwareentwicklung. Daher wird heute häufig der Einsatz einer Middleware, die das verteilte Arbeiten besser unterstützt, vorgezogen. Sie lässt sich in drei Kategorien unterteilen:

20) In Anlehnung an Ruh, W.; Maginnis, F.; Brown, W. (2000), S. 24-27.

In Anlehnung an Linthicum, D. (1999), S. 23-36.

21) In Anlehnung an Ruh, W.; Maginnis, F.; Brown, W. (2000), S. 29.

Message Oriented Middleware (MOM), Distributed Object Technology (DOT) und Transaction Processing Monitor (TPM). An dieser Stelle soll nicht näher auf die technischen Unterschiede eingegangen werden, da sie im Kapitel 4 näher erläutert werden.

In der Literatur wird die Funktionsintegration häufig noch weiter unterteilt: Sie kann auf Objektebene und Prozessebene stattfinden. Ein Objekt besteht aus Daten und zugehörigen Methoden, die als Schnittstelle zur Interaktion mit dem Objekt dienen. Das Objekt Kundenkonto speichert beispielsweise Daten über Konto und Kontoinhaber und stellt Methoden zur Verfügung, um den Kontostand auszulesen und zu modifizieren. Integration auf Prozessebene bezieht auch die Semantik der Prozesse, in denen Objekte be- und verarbeitet werden, mit ein. So ist es möglich, den Kontext der Funktionen zu berücksichtigen und eine umfassendere Integration zu gewährleisten.²²⁾

Mit Integrationsmaßnahmen auf Prozess- und Objektebene werden verschiedene Ziele wie Datenkonsistenz, Koordination von mehrstufigen Prozessen und Realisation von Plug-and-Play Komponenten verfolgt.

In vielen Unternehmen sind mit der Zeit zahlreiche Interdependenzen zwischen Anwendungen gewachsen. Häufig werden Daten zu verschiedenen Zwecken redundant gehalten, die in unterschiedlichen Bereichen der Geschäftsprozesse verwendet werden. Bei Aktualisierungen von Informationen handelt es sich daher häufig um Daten aus verschiedenen Quellen. Falls zum Beispiel eine Kundenadresse im System geändert werden soll, muss zuerst die eingegebene Adresse durch ein Kontrollsystem syntaktisch, eventuell sogar semantisch geprüft werden. Anschließend werden alle relevanten Anwendungen im System von der Änderung benachrichtigt und entsprechend aktualisiert. Dies ist je nach Verteilung der Daten ein komplexer Vorgang, der, um ihn automatisiert durchführen zu können, durch ein Programm unterstützt wird. Oft müssen Vor- und Nachbedingungen überprüft werden. Das Konzept der Datenintegration ist hier also nicht anwendbar, da es sich nicht um eine einfache Änderung eines Datensatzes handelt.

22) In Anlehnung an Winkeler, T.; Westphal, L.; Raupach, E. (2001), S. 8.

Integration von mehrstufigen Prozessen hat zum Ziel, alle Aktivitäten eines Prozesses in der richtigen Reihenfolge über alle relevanten Anwendungen hinweg automatisiert auszuführen. Zu diesem Zweck müssen eine Folge von Ablaufschritten identifiziert werden, die den Integrationsfluss zwischen den Softwarekomponenten darstellt. Ein Beispiel für einen mehrstufigen Prozess ist ein webbasiertes Bestellsystem, das alle notwendigen Aktionen zur Auftragsbearbeitung von der Entgegennahme der Kundenbestellung, über die Logistik bis hin zur Fakturierung selbstständig durchführt. Das System kann automatisch in verschiedene Status wechseln. Eine solche Integration hat also nicht nur die Aufgabe, Kommunikation und Anfragen zwischen verschiedenen Anwendungen weiterzuleiten, sondern auch die Koordination und das Management von diesen durchzuführen. Um eine effiziente Integration unter diesem Aspekt zu ermöglichen, sind detaillierte Kenntnisse über den Geschäftsprozess erforderlich.

Eines der kompliziertesten Ziele der Funktionsintegration ist die Plug-and-Play Komponentenintegration. Software soll als Komponente für ein "Stecksystem" entwickelt werden, sodass ohne Aufwand einzelne Anwendungen ausgewechselt und hinzugefügt werden können. Es müssen Interfaces für alle Anwendungen entwickelt werden, die so wohldefiniert sind, dass sämtliche Komponenten ohne Modifikation der Interfaces integriert werden können. Solche Interfaces müssen konsistent nach den gleichen Regeln und der gleichen Syntax entwickelt werden, die von jeder Anwendung interpretiert werden kann. Alle möglichen Funktionen einer Anwendung müssen aufgeführt und widerspruchsfrei definiert werden, damit gleiche Funktionen in allen Anwendungen über den gleichen Namen aufgerufen werden, z. B. muss ein Kundendatensatz einheitlich über `create_customer` und nicht `new_customer` erzeugt werden. Diese Interfaces spiegeln nicht nur die volle Funktionalität eines Programms wieder, sondern verwalten es auch, indem sie operationale Dienste wie Sicherheit oder Fehlerbehandlung übernehmen. Die Plug-and-Play Komponentenintegration ist sicherlich das komplexeste Integrationsziel, das bis heute nur von sehr wenigen Anbietern annähernd realisiert worden ist.²³⁾

23) In Anlehnung an Winkeler, T.; Westphal, L.; Raupach, E. (2001), S. 8.
In Anlehnung an Ruh, W.; Maginnis, F.; Brown, W. (2000), S. 27-38.
In Anlehnung Linthicum, D. (1999), S. 61-90.

3.4 Bewertung der Methoden

Die Darstellungsintegration ist die einfachste und kostengünstigste Methode. Unterstützende Tools lösen viele Probleme weitgehend automatisiert, sodass der Fokus auf Design der neuen Benutzeroberfläche und nicht auf der Integrationsproblematik liegt. Jedoch bietet dieses Konzept nur sehr begrenzte Möglichkeiten der Integration, da Daten und Funktionslogik überhaupt nicht berücksichtigt werden.

Integration auf Ebene der Daten ist flexibler als die Darstellungsintegration. Falls die Middleware Zugriff auf verschiedene Datenbanken unterstützt, kann auch hier schnell eine leistungsfähige Integration stattfinden, die die Wiederverwendung von sämtlichen Daten unterstützt. Als Nachteil ist jedoch zu bewerten, dass die Methoden bei den Anwendungen verbleiben. Die Geschäftslogik muss daher in jedem IV-System neu programmiert werden, wodurch der Entwicklungs- und Wartungsaufwand teilweise erheblich steigt. Jede Integration ist an ein Datenmodell geknüpft. Ändert sich das Modell, ändert sich die Basis der Integration und zieht umfangreiche Änderungen mit sich. Es ist von wesentlicher Bedeutung, ein Datenmodell sorgfältig zu erarbeiten, dass es Veränderungen im Zeitablauf standhält.²⁴

Die Funktionsintegration ist die stabilste und flexibelste Integration mit der umfangreichsten Wiederverwendung von Software. Jedoch ist sie auch die komplexeste und schwierigste aller Konzepte, da neben den technischen auch sehr hohe semantische Anforderungen gestellt werden. Die Integration der Geschäftslogik erfordert eine vollständige Analyse aller Anwendungen. Dies gestaltet sich im Besonderen schwierig, wenn die Software nicht dokumentiert ist. Leider ist dies sehr häufig der Fall. In der Abb. 10 sind die wesentlichen Charakteristika noch einmal übersichtlich aufgeführt.²⁵⁾

24) Weiterführende Literatur zur Schemaintegration: Vidal, V.; Winslett, M. (1994).

25) In Anlehnung an Ruh, W.; Maginnis, F.; Brown, W. (2000), S. 23, 26-27, 38.

	Darstellung	Daten	Funktion
Realisierungsdauer	kurz	mittel	lang
Komplexität	gering	mittel	hoch
Flexibilität	gering	mittel	hoch
Integration auf Darstellungsebene	ja	nein	nein
Integration auf Datenebene	nein	ja	ja
Integration auf Funktionsebene	nein	nein	ja

Abb. 10: Vergleich der verschiedenen Integrationsmethoden

4 Middlewarebasierte Architekturen

Die auf dem Markt verfügbaren EAI-Systeme basieren auf Middleware verschiedenster Art. Versuche, dieses heterogene Erscheinungsbild zu kategorisieren, führen zu vielen Klassifikationen mit unterschiedlichen Kriterien. Im Folgenden werden drei verschiedene Klassifikationsrichtungen vorgestellt.

Klassifikation anhand von Merkmalen

Serain schlägt eine Gliederung anhand folgender Kriterien vor:

- Verfügbarkeit auf verschiedenen Plattformen, da die angebotenen IV-Systeme häufig auf verschiedenen Plattformen laufen
- Zuverlässigkeit des Datentransfers, zum Beispiel die der Nachrichten
- Anpassung an die Bandbreite, da die Middleware als Rückgrat der Anwendung flexibel gegenüber hohem Nachrichtenaufkommen sein muss
- Unterschiedliche Kommunikationsstrukturen, zum Beispiel 1:1 und 1:n
- Namensräume, die Anwendungen unabhängig von Netzwerkadressen adressierbar machen
- Transaktionskonzepte, um im Besonderen bei mehrstufigen Transaktionen eine Kontrolle zu gewährleisten.²⁶⁾

26) In Anlehnung an Ließmann, H. (2000), S. 59f.
M. w. N. Serain, D. (1999), S. 4-5.

Klassifikation anhand von Diensten

Als weiteres Klassifikationsmerkmal werden häufig Dienste aufgeführt. Dienste werden als Funktionserweiterung zu den grundlegenden Eigenschaften einer Middleware gezählt. Kriterien sind beispielsweise: ²⁷⁾

- **Kommunikationsdienste:**
Peer-to-peer-Nachrichtenaustausch, RPC, Message Queuing, E-Mail, EDI
- **Systemmanagement:**
Ereignis- und Benutzerkontenverwaltung, Konfigurations- und Installationsmanagement, Fehlerbearbeitung sowie Authentifizierung und Überwachung
- **Informationsdienste:**
Verzeichnis-, Sicherheits- und Dateidienste sowie relationale DBMS, OODB und Repository-Verwaltung
- **Ablaufkontrolldienste:**
Thread-Verwaltung, Transaktionsverarbeitung, Ressourcen- und Prozessverwaltung
- **Präsentationsdienste:**
Verwaltung von Formularen, Masken, Grafiken und Druckern sowie Multimedia-Aufbereitung
- **Berechnungsdienste:**
Sortieren, mathematische Funktionen, Dienste zur Internationalisierung, Zeitberechnung und Konvertierung von Daten.

Klassifikation anhand von Produkttypen

Middleware dient als Basis für EAI. Derzeit existieren fünf verschiedene Realisierungsmöglichkeiten auf dem Markt: Remote Procedure Calls, Database Access Middleware, Message Oriented Middleware, Distributed Object Technology und Transaction Processing Monitors.

27) Ließmann, H. (2000) S. 59f.
M. w. N. Bernstein, P. A (1996), S. 86-98.
M. w. N. Riehm, R.; Vogler, P. (1996) S. 25-135.
M. w. N. Tresch, M. (1998), S. 249-256.

Mit Remote Procedure Calls²⁸⁾ können Grundfunktionalitäten von Middleware realisiert werden, die Bezeichnung Middleware wäre jedoch zu umfassend. Durch RPCs können entfernte Methoden und Prozeduren in Verteilten Systemen über ihre Signatur aufgerufen werden, ohne dass dem Aufrufenden die Logik der Funktion bekannt ist. RPC stellt eine einheitliche Syntax bereit, damit in unterschiedlichen Sprachen geschriebene Anwendungen kommunizieren können. In modernen EAI-Architekturen jedoch werden RPCs wegen ihres prozeduralen Schwerpunkts nur selten verwendet, da Objektorientierung beispielsweise ein wesentlich leistungsfähigeres System ist.²⁹⁾

Datenzugriffsorientierte Produkte, z. B. Database Access Middleware, ermöglicht den Zugriff auf verteilte, heterogene Daten wie Dateien oder Datenbanken und Programmlogik wie Stored Procedures³⁰⁾ auf der Datenebene. Sie findet häufig Anwendung in der Datenintegration. Jeder Hersteller hat eigene Standards entwickelt, der verbreitetste Standard jedoch ist ODBC von Microsoft. Da ODBC nur relationale Datenbanken unterstützt, werden weitere, flexiblere Standards wie OLE DB oder ADO entwickelt. Weil die EAI-Integration über die Datenebene hinausgeht, wird die Database Access Middleware nur in Kombination einer EAI-Architektur eingesetzt, die keinen Datenzugriff unterstützt.³¹⁾

Message Oriented Middleware basiert auf Messaging als Integrationskonzept und übernimmt die Koordination, das Erzeugen, Speichern und Kommunizieren von Messages. Durch das Nachrichtenkonzept realisiert sie eine Kommunikation zwischen den IV-Systemen durch eine lose Kopplung ohne ständige Verbindung. Die Anwendungen schicken Nachrichten an die Middleware, welche die Verteilung übernimmt. MOM eignet sich im besonderen dafür, Datenkonsistenz und mehrstufige Prozesse durchzuführen und ist einfach zu entwickeln. Da Messages jedoch verglichen mit Call Interfaces eher unübersichtliche Schnittstellen sind, ist

28) M. w. N. Britton, C. (2000), S. 25 ff..

M. w. N. Coulouris, G; Dollimore, J.; Kindbert, T. (2001), S. 165-205.

M. w. N. Silberschatz, A.; Galvin, P. (1999), S. 506 f..

29) In Anlehnung an Ruh, W.; Maginnis, F.; Brown, W. (2000), S. 53 f..

30) M. w. N. Froese, J.; Moazzami, M.; Rautenstrauch, C.; Welter, H. (1994), S. 245, 206, 298.

31) M. w. N. Britton, C. (2000), S. 27 ff..

die Integration neuer und alter Anwendungen mit MOM eher schwierig. MOM wurde zum Beispiel von IBM in der MQSeries und in den Smart Sockets von Talarian eingesetzt.³²⁾

Die neueste Middleware-Technologie ist die Distributed Object Technology. Sie erweitert die Objektorientierung um das Konzept der verteilten Verarbeitung, indem Interfaces für Anwendungen entwickelt werden, die Software nach außen wie Objekte erscheinen lässt. DOT verwendet Integration über APIs und ist dadurch geeignet für schnelle Integration von neuen Komponenten. Jedoch entstehen durch Interfaces stärkere Verbindungen zwischen den Anwendungen, durch die die Entwicklung einer DOT- im Gegensatz zur MOM-Middleware komplizierter wird. Hersteller wie OMG mit CORBA, Microsoft mit COM+ und Sun mit der J2EE Umgebung haben DOT verwendet.³³⁾

Transaction Processing Monitors erlauben "Verwaltung und Durchführung von Transaktionen über verschiedene Datenquellen hinweg".³⁴⁾ Die Integrität von Transaktionen kann mit Hilfe von TPMs sichergestellt werden, da sie die vier Eigenschaften Atomarität, Konsistenz, Isolation und Dauerhaftigkeit gewährleisten. Es werden die gängigen Merkmale von transaktionsbasierten Systemen wie Rollback, Error Logging u. a. unterstützt. Selten wird TPM allein als Basistechnologie verwendet, aber die TPM-Funktionalität wird in MOM und DOT integriert. BEA hat mit Tuxedo einen reinen TPM entwickelt.³⁵⁾

In Kapitel 4.1, 4.2 und 4.3 werden unterschiedliche Architekturen vorgestellt, die diese verschiedenen Middleware-Technologien als Basis haben.

4.1 Nachrichtenorientierte Middlewarearchitektur

In Middlewarearchitekturen wird zwischen zwei verschiedenen Integrationsmethoden unterschieden: dem Messaging und den Application Programming Interfaces. Messaging arbeitet mit Nachrichten, die sowohl die Daten als auch darauf anzuwendende Aktionen enthalten. Der Sender schickt die Nachricht in einem geeigneten Format über das

32) In Anlehnung an Ruh, W.; Maginnis, F.; Brown, W. (2000), S. 55, 28.

33) In Anlehnung an Ruh, W.; Maginnis, F.; Brown, W. (2000), S. 56., 28.

34) In Anlehnung an Ließmann, H. (2000), S. 61.

35) In Anlehnung an Ruh, W.; Maginnis, F.; Brown, W. (2000), S. 56 f., 28.

Kommunikationssystem an den Empfänger, der sie decodiert, um seine Anweisungen zu eruiieren. Durch Kombination von Daten und darauf auszuführenden Aktionen wird eine hohe Flexibilität erreicht, da zur Funktionserweiterung nur eine neue Nachricht erzeugt werden muss. Schwierigkeiten treten jedoch dadurch auf, dass der Sender nicht im vorhinein ermitteln kann, ob die Anwendung seine Aufgaben bearbeiten kann.³⁶⁾

Ein typischer Aufbau einer Nachricht wird in Abb. 11 gezeigt.

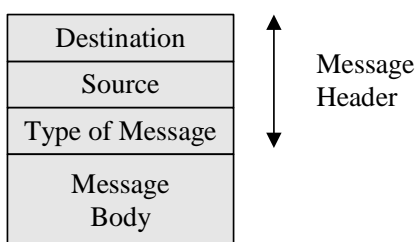


Abb. 11: Aufbau einer Nachricht³⁷⁾

In dem sogenannte Message Header wird das Ziel und die Quelle angegeben und die Art der Nachricht spezifiziert. Der Message Body enthält die eigentliche Nachricht, deren Inhalt zwischen Sender und Empfänger abgestimmt ist.³⁸⁾

In nachrichtenorientierten Middlewarearchitekturen ist Messaging die zentrale Methode, um verschiedene IV-Systeme, Client, Server, clientbasierte Anwendungen, serverbasierte Anwendungen, Packaged Applications und Legacy Systeme, zu verbinden. Messaging ist die technische Grundlage für Datenbewegungen zwischen Systemen.

Zwischen zwei verschiedenen Kommunikationsarten ist zu entscheiden, der synchronen und der asynchronen, wobei MOM meist die asynchrone Kommunikation verwendet.

Die synchrone Kommunikation wird verwendet, wenn Sender und Empfänger vom System kontrolliert kommunizieren sollen. Da auf eine Reaktion des Kommunikationspartners gewartet wird, hat sie jedoch, falls er nicht reagiert oder überlastet ist, den Nachteil, dass der Sender der Nachricht bis auf unbestimmte Zeit blockiert ist. Dadurch kann die Performance

36) M. w. N. Ruh, W.; Maginnis, F.; Brown, W. (2000), S. 49 f..

37) Bacon, J. (1993), S. 286.

38) In Anlehnung an Bacon J. (1993), S. 285 ff..

eines Systems stark herabgesetzt werden. Bei der asynchronen Kommunikation dagegen wird diese Abhängigkeit verringert, da der Sender einer Nachricht nicht mehr auf die Reaktion des Empfängers warten muss. Er schickt seine Information und arbeitet weiter. Seine Nachricht wird entweder bearbeitet, oder er wird nach Ablauf eines vom Entwickler gewählten Time Out benachrichtigt. Bei der asynchronen Kommunikation kann sich die fehlende Kontrolle der Kommunikation nachteilig auf die Schnelligkeit auswirken.³⁹⁾

Verschiedene Arten der Kommunikation sind Message Passing und Message Queuing. Auf Message Queuing basieren verschiedene Kommunikationsmechanismen wie Request/Reply, Publish/Subscribe und Message Queuing mit mehreren Queues und einem zentralen Router.

Kommunikation über Message Passing

Message Passing ist der einfachste Mechanismus der asynchronen Kommunikation. Es wird verwendet, um Daten über ein Netzwerk zu versenden oder Methoden auf entfernten Rechnern aufzurufen.

Die Kommunikationspartner verständigen sich über ein Netzwerk (Abb. 12) ohne jegliche Kontroll- oder Bestätigungsmechanismen.

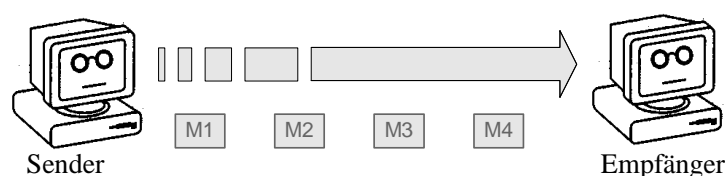


Abb. 12: Message Passing

Der Sender einer Information schickt seine Nachricht in Form einer Message in das Netzwerk. Falls der Empfänger nicht bereits auf die Nachricht wartet, wird sie zunächst im System zwischengespeichert, bis sie entgegengenommen wird. Voraussetzung für diesen Mechanismus ist, dass die Kommunikationspartner sich gegenseitig kennen, anhand der Angaben in der Nachricht den jeweiligen Partner identifizieren können und die

39) M. w. N. Ruh, W.; Maginnis, F.; Brown, W. (2000), S. 40 - 48.
M. w. N. Nehmer, J.; Sturm, P. (1998).

Kommunikation auf dem beidseitigen Einverständnis über Größe und Inhalt der Nachricht beruht. Dieses Kommunikationsmodell ist einfach zu realisieren, sollte jedoch nur in einem sehr zuverlässigen Netzwerk verwendet werden. Falls die Kommunikation scheitert und Messages verloren gehen, kann dies mit dieser Methode nicht bemerkt werden.⁴⁰⁾

Kommunikation über Message Queuing

Ähnlich wie beim Message Passing verständigen sich die Kommunikationspartner auch beim Message Queuing über ein Netzwerk. Der Sender einer Nachricht schickt diese jedoch nicht direkt an den Empfänger, sondern reiht sie in eine Queue ein. Eine Folge von Nachrichten wird als Queue bezeichnet, "wenn Elemente nur am Ende eingefügt und am Anfang entfernt werden dürfen."⁴¹⁾ Messages werden also im Netzwerk nach dem FIFO zwischengespeichert, bis der Empfänger sie der Reihe nach abrufen.

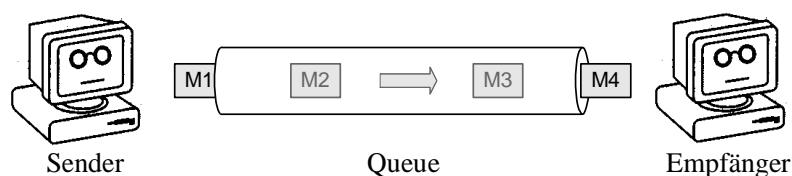


Abb. 13: Message Queuing

Der Sender muss also als erstes eine passende Queue identifizieren, seine Nachricht in das korrekte Format bringen und in der Queue speichern. Die Messages werden an einen Server weitergeleitet, der die Nachrichten empfängt und verarbeitet. Beim reinen Message Queuing sind Anfrage- und Antwortmechanismen getrennte Transaktionen, für deren Durchführung der jeweilige Sender verantwortlich ist.⁴²⁾

40) In Anlehnung an Bacon, J. (1993), S. 286 f..

In Anlehnung an Coulouris, G.; Dollinmore, J.; Kindberg, T. (2001), S. 126 ff.

In Anlehnung an Ruh, W.; Maginnis, F.; Brown, W. (2000), S. 46.

41) O. V. Duden Informatik (1993), S. 620.

42) In Anlehnung an Ruh, W.; Maginnis, F.; Brown, W. (2000), S. 62 f..

M. w. N. Schumacher, M.; Schneider, K (2000).

M. w. N. Leymann, F.; Gossmer, M. (2000).

Auf dem Modell des Message Queuing basiert die Middleware FALCONMQ von Level 8 Systems. Ihre Stärke liegt in der nahtlosen Integration von Microsoft Messaging Technologien in heterogenen Systemen.⁴³⁾

Kommunikation über Request/Reply

Der Request / Reply Mechanismus basiert wie das Message Queuing auf Schlangen, durch die die Kommunikationspartner sich gegenseitig Nachrichten schicken können (s. Abb. 14). Es ist eine synchrone Kommunikationstechnik. Die meisten RMI und RPC Systeme bauen auf Request/Reply Kommunikation auf.

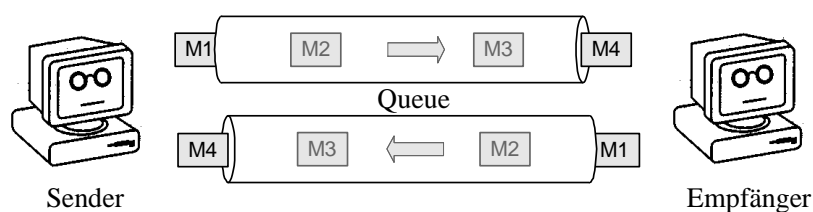


Abb. 14: Request/Reply

Beim Request/Reply Mechanismus werden jedoch im Gegensatz zum Message Queuing Anfrage- und Antwortmechanismen als eine einzige Transaktion betrachtet. Die Abb. 15 zeigt auf der linken Seite den gewöhnlichen Kommunikationsablauf und auf der rechten Seite die beim Request/Reply übliche Realisierung.

In Abb. 15a) arbeitet der Sender, nachdem er den Request an den Empfänger geschickt hat, weiter, bis er die Antwort erhält. Um die Komplexität zu senken und die Kommunikation besser kontrollieren zu können, wird bei Request/Reply (Abb. 15b) Anforderung und Antwort zu einer Transaktion, dem Request Service, zusammengefasst. Der Sender wartet also die Antwort des Empfängers ab, bevor er weiterarbeitet. Die Nachricht des Empfängers ist so nicht nur eine Antwort sondern gleichzeitig auch die Bestätigung für einen gelungenen

43) In Anlehnung an Ruh, W.; Maginnis, F.; Brown, W. (2000), S. 72-73.

Weitere Informationen und Produktspezifikationen unter <http://www.level8.com>.

Request. Dieser Mechanismus ist erforderlich, falls eine direkte Antwort für das korrekte Ausführen einer Transaktion erforderlich ist.⁴⁴⁾

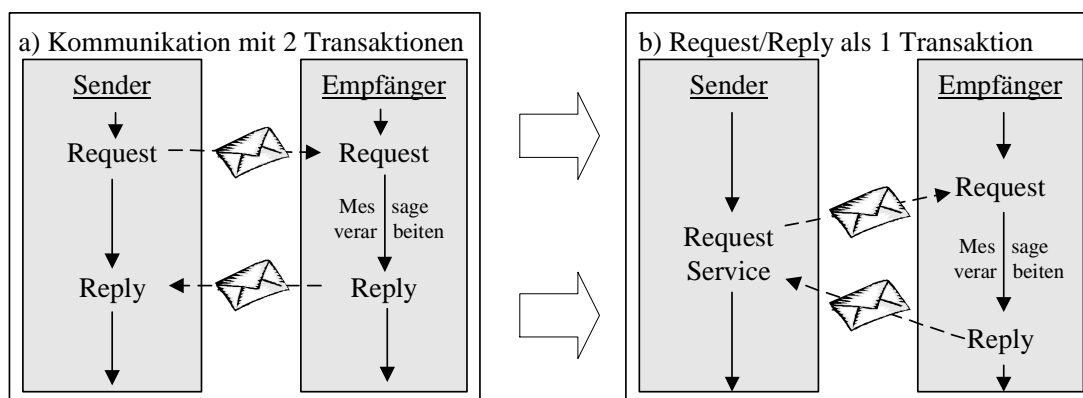


Abb. 15: Kommunikationsablauf von Request/Reply⁴⁵⁾

Die Middleware MQSeries von IBM basiert auf dem Request / Reply Kommunikationsmodell. Sie wird vornehmlich für Echtzeittransaktionen im Finanzsektor und für mobile Applikationen eingesetzt.⁴⁶⁾

Kommunikation über Publish/Subscribe

Das Publish/Subscribe Paradigma wird häufig in eventbasierten Systemen verwendet. Es kann zur Broadcastkommunikation genutzt werden.

Damit Anwendungen auf Änderungen in anderen Systemen reagieren können wurde ein eventbasierter Kommunikationsmechanismus entwickelt. Falls eine Anwendung eine Änderung in einem ihrer Objekt registriert, schickt sie ein sogenanntes Event in das Netzwerk, über das "interessierte" Anwendungen benachrichtigt werden. Die Meldung von Ereignissen ist asynchron und wird vom Empfänger, dem Server, kontrolliert (s. Abb. 16).

44) In Anlehnung an Bacon, J. (1993), S. 288 ff..

M. w. N. Coulouris, G.; Dollimore, J.; Kindbert, T. (2001), S. 146-153.

45) In Anlehnung an Bacon, J. (1993), S. 76.

46) In Anlehnung an Ruh, W.; Maginnis, F.; Brown, W. (2000), S. 72-73.

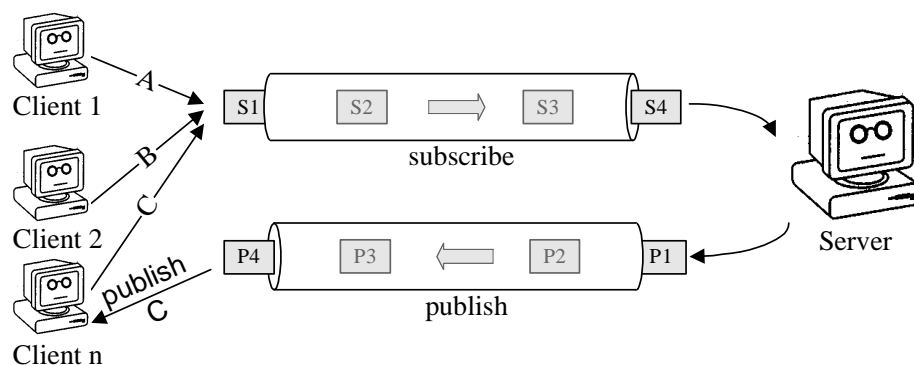


Abb. 16: Publish/Subscribe

Beim Publish/Subscribe Mechanismus abonniert die Clientanwendung einen bestimmten Typ Nachrichten, der aufgrund festgelegter Events oder Kriterien versendet wird. Wenn eine Serveranwendung ihre Nachricht veröffentlicht, wird sie an alle Abonnenten dieses Nachrichtentyps geschickt. Verschiedene Arten von Nachrichten können beispielsweise verschiedene Methoden sein, die auf einem für das IV-System relevanten Objekt ausgeführt wurden.⁴⁷⁾

Talarian produziert die Middleware Smart Sockets, die auf dem Kommunikationsmodell Publish/Subscribe aufbaut. Sie zeichnet sich durch besonders effiziente Nutzung der Bandbreite aus und eignet sich im Einsatz von Netzwerken, wo Server permanent verfügbar sind.⁴⁸⁾

Kommunikation mit mehreren Queues und Routing

In der Realität sind die meisten Systeme zu komplex, um mit den oben beschriebenen Kommunikationsmodellen realisiert werden zu können. Fortgeschrittene Systeme verwenden daher oftmals ein System von mehreren Queues, um Durchsatz und Performance zu optimieren.

Daher wurde ein Modell entwickelt, das dem Queue Management die Aufgaben der Organisation und Koordination der Nachrichten überträgt (s. Abb. 17).

47) In Anlehnung an Coulouris, G.; Dollimore, J.; Kindbert, T. (2001), S. 187 ff..

48) In Anlehnung an Ruh, W.; Maginnis, F.; Brown, W. (2000), S. 72-73.

M. w. N. o. V.: Talarian: all you need to know about Middleware (2000), <http://www.talarian.com/industry/middleware/whitepaper.pdf>.

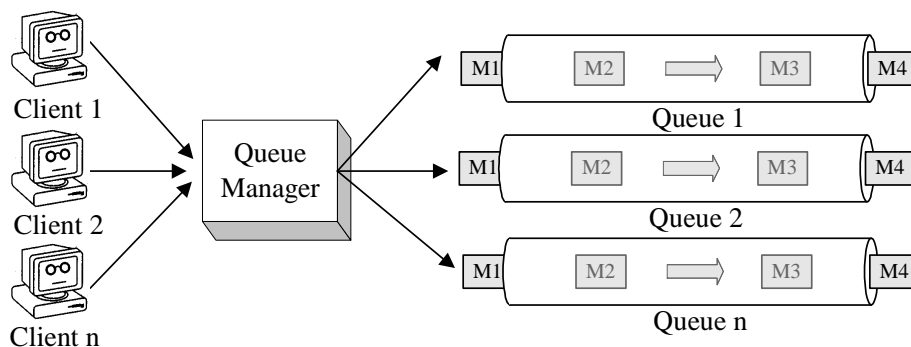


Abb. 17: Queue Manager

Der Sender einer Information vermerkt den Namen des Empfängers in der Nachricht und schickt sie an einen zentralen Queue Manager. So braucht der Client seinen Kommunikationspartner nicht mehr zu lokalisieren, da dies von dem Queue Manager übernommen wird. Genauso werden Antwortzeiten von Servern, Nachrichtengröße, -format und Prioritäten der unterschiedlichen Nachrichten und Queues für das ganze System zentral kontrolliert. Da Queues nur eine endliche Kapazität haben, müssen sie gegen Überlauf und somit den Verlust von Messages geschützt werden. In komplexen Systemen werden mehrere Queues der gleichen Art eingesetzt, um die Kommunikationsbandbreite zu erhöhen. Daher existieren viele verschiedene Wege, eine Nachricht durch das Netzwerk an seinen Empfänger zu versenden. Das Routing der Messages durch das System wird ebenfalls zentral geleitet.⁴⁹⁾

4.2 Objektorientierte Middlewarearchitektur

Objektorientierte Middlewarearchitekturen nutzen zur Integration das Application Programming Interface (API). Bei API, in der Literatur auch Interface Definition oder Call Interface genannt, kommuniziert der Sender von Informationen nicht über Nachrichten sondern mit Aufruf-Schnittstellen, die die Anwendung zur Verfügung stellt. Hier sind alle möglichen Aktionen definiert. Die Daten werden durch Interfaces an den Empfänger geschickt. Um API als Integrationsmethode nutzen zu können, muss jede Anwendung wohldefinierte Schnittstellen haben. Neue Funktionalität kann in Echtzeit hinzugefügt werden, indem das Interface geändert wird. Im Gegensatz zu Nachrichten sind APIs mit einem speziellen Programm verbunden, daher sind bei Änderungen komplexe

49) In Anlehnung an Ruh, W.; Maginnis, F.; Brown, W. (2000), S. 64-71.

Programmieraufwendungen notwendig.⁵⁰⁾ Jedoch ermöglicht API durch seine explizite Architektur einfache Wiederverwendung und bessere Wartung.⁵¹⁾

Die Distributed Object Technology ist das Fundament, auf dem eine objektorientierte Middleware aufbaut. Man unterscheidet zwischen reinen objektorientierten Systemen und komponentenbasierten Systemen. Reine objektorientierte Technologien sind z. B. CORBA in der Version 2.0, DCOM und RMI. Auf diese soll aber im Folgenden nicht näher eingegangen werden, da beide Technologien bereits zu den Komponentensystemen CORBA 3.0 und COM+ weiterentwickelt wurden. Im Wesentlichen existieren derzeit Produkte von drei verschiedenen Herstellern auf dem Markt:

die Component Object Request Broker Architecture von OMG

- die Enterprise Java Beans von Sun Microsystems
- das Component Object Model + von Microsoft.

Das zugrundeliegende Produkt prägt wesentlich die Architektur der Middleware, die im Folgenden eingehend erläutert wird.

Unabhängig vom Hersteller ermöglicht eine objektorientierte Middleware die Speicherung von Objekten auf verschiedenen Rechnern in Verteilten Systemen. In der Praxis hat sich gezeigt, dass objektorientierte Architekturen, da sie auf standardisierten Komponentenarchitekturen basieren, eine besonders schnelle Entwicklung von Unternehmensanwendungen unterstützen. Komponenten sind die wesentlichen Bausteine der meisten modernen Systeme. Sie liegen verteilt im System und bieten bestimmte Funktionalität und Dienste an.

CORBA

Die Basis für den de facto Standard CORBA bildet die Object Management Architecture (OMA), die von der Object Management Group (OMG) entwickelt wurde. Sie besteht aus den folgenden Komponenten (s. Abb. 18):

- Application Interfaces, nicht standardisierte Interfaces von Clientobjekten oder Serverobjekten

50) In Anlehnung an Winkeler, T.; Raupach, E.; Westphal, L. (2001), S. 9.

51) M. w. N. Ruh, W.; Maginnis, F.; Brown, W. (2000), S. 50 f..

- Object Request Broker (ORB), der die Kommunikation zwischen den beiden Objekten vermittelt
- Object Services, Objektdienste, die den ORB unterstützen, z. B. Naming Service, Life Cycle Service
- Common Facilities, bietet allgemeine Objektdienste, wie Hilfe-, Druck- und Sicherheitsdienste
- Domain Interfaces, bereichsspezifische Interfaces aus dem Finanz-, Gesundheits-, Telekommunikationsbereich u. a.

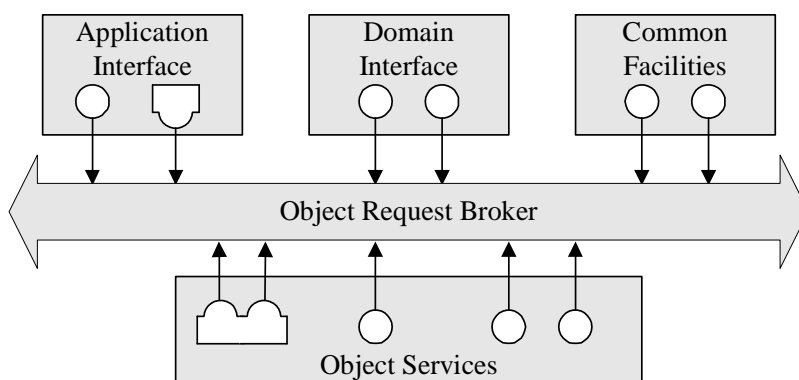


Abb. 18: Object Management Architecture⁵²⁾

Der ORB nimmt eine Vermittlerposition zwischen verteilten Anwendungen, dem Client und dem Server, in der OMA ein. Die Kommunikationspartner können auf unterschiedlichen Plattformen in unterschiedlichen Programmiersprachen verteilt im System existieren. Sie kommunizieren über eine definierte Schnittstelle mit dem ORB, der die Operationsaufrufe vom Client zum Server übermittelt und deren Ergebnisse zurückliefert. Außerdem übernimmt der ORB, unterstützt durch den Lebenszyklusdienst (s. Object Services), das Erzeugen, Aktivieren und Löschen von Serverinstanzen.

Die Abb. 19 veranschaulicht die Kommunikation zwischen Client- und Serverobjekten. Die ORB core Schicht fungiert hier als reine Transportschicht zwischen den Kommunikationspartnern.

52) O.V.: A Discussion of the Object Management Architecture (1997),
http://www.omg.org/technology/documents/formal/object_management_architecture.htm, S. 4-2.

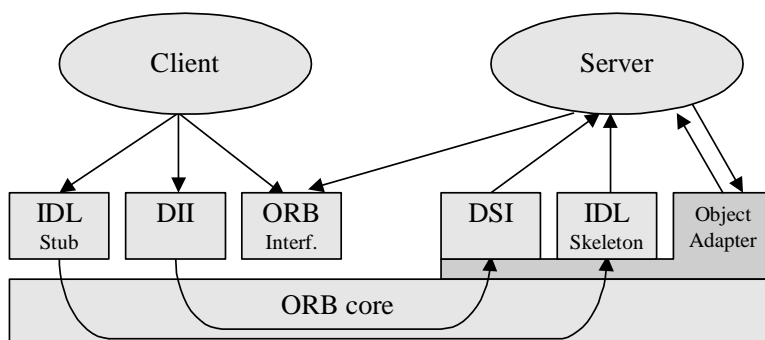


Abb. 19: Object Request Broker⁵³⁾

Ein Server muss, um Dienstleistungen zur Verfügung stellen zu können, Operationen in einem Interface bereitstellen, die durch die Interface Definition Language (IDL) spezifiziert wird. Eine IDL-Schnittstelle basiert auf dem OMG-Objektmodell und enthält folgende Informationen:

- IDL-Stub: "Stellt Routinen, ähnlich wie Bibliotheksroutinen, zur Verfügung, die der Client benutzt, um Dienstleistungen des Servers anzufordern." ⁵⁴⁾
- Dynamic Invocation Interface (DII) wird für verwendet, um dynamische Anforderungen an den Server aufzubauen.
- IDL-Skeleton: Von der OMA-Spezifikation vorgegebener "Rahmen, der vom Programmierer mit Code gefüllt werden muss. Dieser Code wird ausgeführt, wenn eine Anforderung eintrifft." ⁵⁵⁾

Das ORB-Interface und der Object-Adapter stellen Schnittstellen zu ORB-Diensten dar. An dieser Stelle wird nicht näher auf die Implementierungsdetails eingegangen. ⁵⁶⁾

EJB

Enterprise JavaBeans ist eine auf Java basierende Spezifikation, die von Sun Microsystems entwickelt wurde.

53) Krämer, B.; Papazoglou, M.; Schmidt, H.-W. (1998), S. 24.

54) Balzert, H. (2000), S. 927.

55) Balzert, H. (2000), S. 927.

56) In Anlehnung an Balzert, H. (2000), S. 924-931.

In Anlehnung an Krämer, B.; Papazoglou, M.; Schmidt, H.-W. (1998), S. 13-34.

In Anlehnung an O.V.: A Discussion of the Object Management Architecture (1997),

http://www.omg.org/technology/documents/formal/object_management_architecture.htm.

Weitere Informationen zu Details der Spezifikation unter <http://www.omg.org>.

Die Enterprise JavaBeans-Architektur ist eine serverbasierte Komponentenarchitektur für komponentenbasierte Anwendungen. Sun verspricht, dass Applikationen, die unter Verwendung von EJBs geschrieben werden, skalierbar, transaktionsorientiert und mehrbenutzergerecht sind. Anwendungen werden einmal geschrieben und können dann auf jeder EJB-Serverplattform in Betrieb genommen werden.⁵⁷⁾ EJB unterstützt internetbasierte Applikationen und die Entwicklung von Komponenten über die Java Plattform einer Unternehmung hinweg.

Die EJB-Spezifikation baut auf Komponenten, sogenannten Beans, auf. Es werden zwei verschiedene Bean-Typen unterschieden: die Entity Bean und die Session Bean. Die persistenten Entity Beans repräsentieren langlebige Daten. Session Beans sind hingegen transient. Sie modellieren Interaktionen und verwalten den Ablauf eines Kommunikationsprozesses.

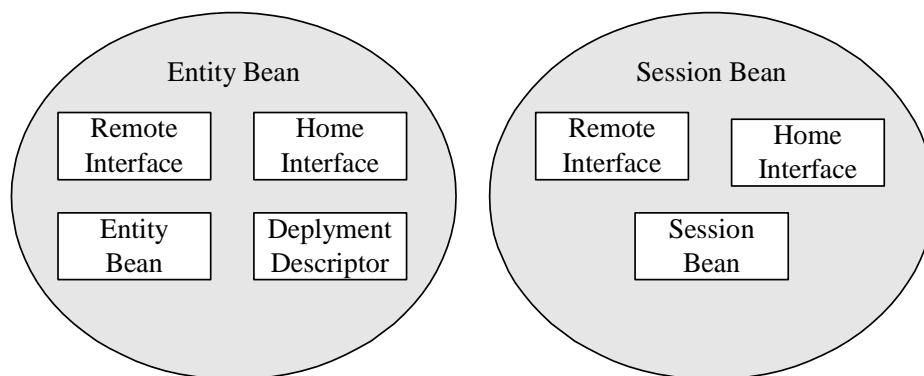


Abb. 20: Enterprise Java Bean

Beans bestehen aus Klassen, die in Abb. 20 dargestellt sind.

- Das Remote Interface ist eine Aufrufschnittstelle. In dem Interface werden alle Dienstleistungen der Komponente, die dem Client angeboten werden, spezifiziert.
- Das Home Interface ist eine Verwaltungsschnittstelle. Sie organisiert den Lebenszyklus einer Bean und bietet beispielsweise Methoden zum Erzeugen oder Suchen eines Objekts.
- Die Bean-Klasse, je nach Art der Bean eine Entity- oder Session Bean-Klasse, beinhaltet die eigentliche Implementierung der Operationen, die in den Schnittstellenklassen dem Client zur Verfügung gestellt werden.

57) In Anlehnung an DeMichiel, L.; Yalçinalp, L.; Krishnan, (2001); S. 1.

- Für die Implementierung einer Entity Bean ist noch eine vierte Klasse notwendig. Der in XML programmierte Deployment Descriptor ist eine Auslieferungsbeschreibung. Er enthält Informationen über die Schnittstellen, Attribute und Operationen der Entity Bean, die bei der Installation benötigt werden.

Die Beans können nicht selbstständig ausgeführt werden, sondern müssen in die EJB-Architektur integriert werden. Sie besteht aus dem Server und einem Container, in dem die Beankomponenten gespeichert sind (s. Abb. 21). Die drei Komponenten haben klar definierte Aufgaben und können daher von verschiedenen Herstellern sein.

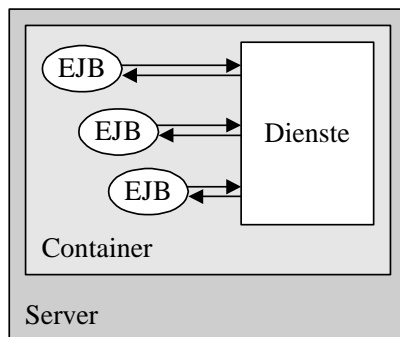


Abb. 21: EJB-Architektur

Der Server ist die Plattform, auf der der EJB-Container installiert wird, und muss Netzwerkanbindung, Skalierungsfunktionen zum Beispiel in einem Cluster und ein Prozess- und Ressourcenmanagement bereitstellen. Ein Container setzt auf dem Server auf und realisiert eine Benutzerverwaltung, Transaktionsmanagement und die Persistenz von EJBs. Er bietet außerdem Unterstützung bei der Installation von Beans. In der EJB-Spezifikation implementiert der Container klar definierte Schnittstellen. In der Praxis werden meist Server und Container von einem Hersteller produziert, da dadurch die Produkte besser aufeinander abgestimmt werden können und eine bessere Performance realisiert werden kann. Die Beankomponenten werden von dem Anwendungsentwickler erstellt und beinhalten die Geschäftslogik. Sie setzen auf den standardisierten Schnittstellen des Containers auf, der das technische Grundgerüst bereitstellt. Das hat den Vorteil, dass ein Beanentwickler nur über das

entsprechende Fachwissen, aber nicht über technische Implementierungsdetails von Transaktionssicherheit, Persistenz u. a. verfügen muss.⁵⁸⁾

COM+

Das Component Object Model+ von Microsoft hat sich aus den Vorläufern COM und DCOM entwickelt. Component Object Model (COM) ist ein binärer Standard zur Kommunikation zwischen Objekten. COM ist sowohl sprachunabhängig als auch prinzipiell betriebssystemunabhängig, wird jedoch nur mit dem Betriebssystem Windows vertrieben. Um die Kommunikation von Objekten auch in verteilten Systemen zu ermöglichen, hat Windows COM auf den DCOM-Standard weiterentwickelt. Anhand von RPCs können Methoden auf entfernten Computersystemen aufgerufen werden.

Verglichen mit Technologien wie CORBA und EJB sind jedoch beide Standards noch nicht weit genug entwickelt, um komplexe Unternehmenslösungen zu entwickeln. Daher hat Microsoft aus COM und DCOM kombiniert mit dem Microsoft Transaction Server (MTS) eine neue Technologie entwickelt. COM+ basiert auf der Distributed interNet Application Architecture (DNA) und wird mit Windows 2000 vertrieben.

Microsoft DNA 2000 beruht im Wesentlichen auf fünf Microsoftprodukten, dem Windows 2000 Server, dem Visual InterDev, dem SNA Server, der als Gateway und Integrationsplattform dient. Der SQL Server ist eine relationale Datenbank, der Site Server hat in der Architektur die Aufgabe, Informationen im Intranet zu veröffentlichen, zu finden und zu teilen.⁵⁹⁾

58) In Anlehnung an Balzert, H. (2000), S. 916-924.

In Anlehnung an Ruh, W.; Maginnis, F.; Brown, W. (2000), S. 101-104.

M. w. N. Monson-Haefel, R. (2000), S. 1-52.

M. w. N. Britton, C., (2000), S. 70-71.

M. w. N. DeMichie, L.; Yalçinalp, L.; Krishman, (2001).

Weitere Informationen und Details der Spezifikation unter <http://java.sun.com>.

59) In Anlehnung an Balzert, H. (2000), S. 934-936.

In Anlehnung an Britton, C., (2000), S. 52-56, 69-70.

In Anlehnung an Ruh, W.; Maginnis, F.; Brown, W. (2000), S. 98-101.

In Anlehnung an Krämer, B.; Papazoglou, M.; Schmidt, H.-W. (1998), S. 39-69.

Vergleich der Technologien

Die Komponentenmodelle sind nicht leicht zu vergleichen - zu verschieden sind die Anforderungen einer Unternehmenslösung. Grundsätzlich kann man eine positive Entwicklung in der Einsatzhäufigkeit der Technologien feststellen. Komponentenbasierte Techniken ermöglichen durch ihre zusätzlichen Dienste eine schnellere, flexiblere und somit kostengünstigere Entwicklung als einfache Verteilungsplattformen wie zum Beispiel frühere Versionen von CORBA, DCOM oder RMI.

Um die oben vorgestellten Technologien zu bewerten, wird hier ein Vergleich unter den Kriterien Plattformunabhängigkeit, Sprachunabhängigkeit und Herstellerunabhängigkeit herangezogen und in der Abb. 22 strukturiert dargestellt.

Kriterium	CORBA	EJB	COM+
Plattformunabhängigkeit	gegeben	gegeben	im Wesentlichen auf Windows beschränkt
Sprachunabhängigkeit	auf Java beschränkt	gegeben	gegeben
Herstellerunabhängigkeit	gegeben, aber Spezifikation von Sun	gegeben	Microsoft

Abb. 22: Vergleich der Modelle⁶⁰⁾

Das CORBA Component Model (CCM) ist von der Architektur den EJBs sehr ähnlich. Beans können in einem CORBA-System als Komponenten genutzt werden. Die Integration von COM+-Komponenten ist in der Spezifikation zumindest prinzipiell auch möglich. Daher ist die Abgrenzung und Bewertung der Technologien sehr schwierig.

4.3 Transaktionsorientierte Middlewarearchitektur

Um vollständige Integration zu gewährleisten, muss die Integrität von mehrstufigen Geschäftsprozessen gesichert sein - auch, wenn sie unvorhergesehen im Ablauf abbrechen. Das Hauptaugenmerk von transaktionsorientierter Middleware richtet sich auf diese Problematik der Transaktionssicherheit.

⁶⁰⁾ In Anlehnung an Balzert, H. (2000), S. 937.

Bei Middleware wird unter einer Transaktion eine Folge von Geschäftsfunktionen verstanden, die nur dann vollständig abgeschlossen ist, wenn alle Geschäftsfunktionen in der richtigen Reihenfolge durchgeführt und beendet wurden. Folgende vier Anforderungen werden an Transaktionen gestellt: Transaktionen müssen

- unteilbar (Atomicity)
- konsistent (Consistency)
- isoliert (Isolation)
- dauerhaft (Durability)

sein. In der Literatur werden diese Merkmale häufig in dem Schlagwort ACID zusammengefasst. Die Eigenschaft der Unteilbarkeit verlangt, dass entweder alle Geschäftsfunktionen einer Transaktion ausgeführt werden oder keine. Eine Transaktion darf niemals nach Ausführung nur eines Teils der Aktionen beendet werden. Die Anforderung der Konsistenz verlangt, dass das System sich vor und nach der Transaktion in einem konsistenten Zustand befindet. Während der Abarbeitung der Transaktion können inkonsistente Systemzustände auftreten. Das Ergebnis von parallel ausgeführten Aktionen muss dem einer sequentiellen Ausführung entsprechen. Im Besonderen müssen Inkonsistenzen während des Ablaufs isoliert, d. h. für andere Transaktionen nicht sichtbar, sein. Das Ergebnis einer Transaktion muss dauerhaft sein, d. h. persistent auf einem nichtflüchtigen Medium gespeichert werden.

Ein vielzitiertes Beispiel einer Transaktion ist eine Kontoüberweisung. Um eine Überweisung durchzuführen, muss zuerst Konto 1 mit einer bestimmten Summe belastet werden. Im zweiten Schritt wird diese Summe dem Konto 2 gutgeschrieben. Es wäre fatal, wenn die Transaktion nach der Hälfte durch einen Fehler abbrechen würde und die Gutschrift nicht mehr stattfände.⁶¹⁾

Transaction Processing Monitor

Um Transaktionen besser kontrollieren und verwalten zu können, wurden Transaction Processing Monitors (TPM) entwickelt. Sie sichern die Integrität von Geschäftsprozessen,

61) In Anlehnung an Balzert, H. (2000), S. 907-909.

In Anlehnung an Coulouris, G.; Dollimore, J.; Kindbert, T. (2001), S. 471.

indem sie die vier ACID-Eigenschaften gewährleisten. Vor einer Folge von Geschäftsfunktionen, die als Transaktion durchgeführt werden muss, startet der TPM eine neue Transaktion, die er erst nach der letzten Funktion der Transaktion beendet. Falls während der Abwicklung ein Fehler auftritt, durch den die Transaktion abgebrochen wird, sorgt der TPM für die Rückabwicklung der Transaktion.

Transaction Processing Monitors können lokal auf einem Rechner, z. B. einem Mainframe, oder in Verteilten Systemen laufen. In IBM Mainframe Umgebungen sind zwei Mainframe Transaction Monitors weithin verbreitet: das Information Management System (IMS) und das Customer Information Control System (CICS). Das IMS basiert auf der Messaging-Integrationsmethode wohingegen CICS auf der API-Integrationsmethode beruht. Mainframe Transaction Monitors zeichnen sich in der Regel durch eine besonders hohe Leistungsfähigkeit bezüglich der Skalierbarkeit, der Performance und der Stabilität aus.

Distributed Transaction Monitors (DTM) unterstützen die Verwaltung von Transaktionen in Verteilten Systemen. Da eine Transaktion auf verschiedenen Plattformen von unterschiedlichen Herstellern abläuft, werden die Anforderungen an die Koordination komplexer und schwieriger. Verteilte Transaktionen werden auf einem System, dem sogenannten Master, angestoßen. Der Master verteilt einzelne Teilaufgaben der Transaktion auf andere Systeme, die dort als Teiltransaktion abgearbeitet werden. Um verteilte Transaktionen zu verwalten, wurde das Zwei-Phasen-Commit entwickelt. Eine Transaktion darf nur dann abgeschlossen werden, wenn jeder der Teiltransaktionen korrekt und vollständig durchgeführt wurden. Ansonsten muss die Transaktion mit allen ihren verteilten Teiltransaktionen zurückgesetzt werden. Der DTP Standard der Open Group definiert drei Teilprogramme, die für verteilte Transaktionen notwendig sind:

- Anwendung
- Ressourcenverwaltung (RV)
- Transaktionsverwaltung (TV)
- Kommunikationsressourcenverwaltung (KRV)

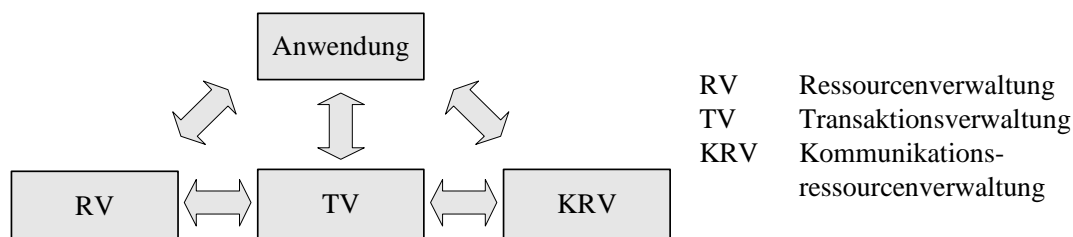


Abb. 23: Distributed Transaction Processing (DTP) Modell⁶²⁾

Die Anwendung ist ein Programm, das an der Transaktion beteiligt ist. Zur Initiierung und Ausführung einer Transaktion verwendet die Anwendung die Transaktionsverwaltung, die die Kontrolle und Koordination übernimmt. Die Ressourcenverwaltung kontrolliert verteilte Ressourcen wie beispielsweise Datenbanken, die während der Abarbeitung benötigt werden. Für das Management der Kommunikation unter den verschiedenen Transaktionsteilnehmern ist die Kommunikationsressourcenverwaltung zuständig.

Die bekanntesten Distributed Transaction Monitors sind der Tuxedo von BEA Systems und Encina von IBM/Transarc. Jedoch sind reine DTMs recht selten zu finden. Häufig wird die Technologie in andere, z. B. komponentenorientierte Systeme, integriert, sogenannte Object Transaction Monitors.

Object Transaction Monitor

Object Transaction Monitors (OTM) stellen Transaktionsintegrität für verteilte Objekte zur Verfügung. Sie sind in Komponentenmodellen wie CORBA, EJB und dem Microsoft Transaction Server (MTS), die in Kapitel 4.2 bereits eingehend beschrieben wurden, integriert. An dieser Stelle soll nur noch einmal kurz auf diese Technologien unter dem Gesichtspunkt der Transaktionsorientierung eingegangen werden.

CORBA Object Transaction Service ist der CORBA Standard für verteilte Objekttransaktionen. Er basiert auf dem Open Group DTP Modell. Sun bietet für Enterprise JavaBeans den Java Transaction Service an. Java Transaction Service ist eine Spezifikation einer Transaktionsverwaltung, die die Java Transaction API unterstützt. Er verwendet die

62) In Anlehnung an Ruh, W.; Maginnis, F.; Brown, W. (2000), S. 98-101.

Weitere Informationen und Details der Spezifikation unter <http://www.opengroup.org>.

CORBA Object Transaction Service Interfaces und das Internet Inter-ORB Protokoll, ein Protokoll, das die Kommunikation zwischen CORBA und CORBA-unterstützende Applikationen definiert.

Der Microsoft Transaction Server (MTS) ist ein wichtiger Vertreter von Object Transaction Monitors. Wie der Object Transaction Service von CORBA und von EJB basiert auch der MTS auf dem Open Group DTP Modell und integriert Microsofts COM+ und DCOM-Komponenten.⁶³⁾

5 Zusammenfassung und Ausblick

Auf dem Markt ist kaum ein einziges, integriertes Softwarepaket zu finden, das den heutigen Ansprüchen von Unternehmen genügt. Viele Einzellösungen prägen die Systemlandschaft, deren Erscheinungsbild durch die sich schnell ändernden Anforderungen einem ständigen Wandel unterzogen ist. Um dem harten Wettbewerb standhalten zu können, müssen die Einzellösungen unternehmensweit über eine Integrationsplattform, eine Middleware, integriert werden, um eine flexible und leistungsfähige IT-Infrastruktur zu schaffen.

Enterprise Application Integration ist zum einen eine reine Softwareintegration, die auf der Daten-, Funktions- und Darstellungsebene stattfindet. Zum anderen erfordert EAI eine Reorganisation der Geschäftsprozesse. Es wird eine Architektur entwickelt, die standardisierte Schnittstellen zur Verfügung stellt. Anhand von Produktbeispielen werden die technischen Eigenschaften der verschiedenen Architekturen und deren Einflüsse auf die unterschiedlichen Gestaltungskonzepte erläutert. Auf dem Markt derzeit vorherrschende Middleware sind datenzugriffsorientierte Architekturen, die jedoch auf die reine Datenebene beschränkt sind, nachrichtenorientierte, objektorientierte und transaktionsorientierte Architekturen.

In der Literatur ist man übereinstimmend der Meinung, dass vollständige Integration mit EAI wegen dem immer stärkeren Bedürfnis zur interorganisatorischen Organisation in der Zukunft ein bestimmender Wettbewerbsvorteil sein wird. Die geschäftsstrategische Perspektive von

63) In Anlehnung an Ruh, W.; Maginnis, F.; Brown, W. (2000), S. 112-129.

EAI, Prozesse und Geschäftslogik noch stärker mit der technischen Infrastruktur zu integrieren, wird in der Praxis an Bedeutung zunehmen. EAI als offene Integrationsplattform ist der Schlüssel für die Fähigkeit von Unternehmen, sich selbst und ihre IT so flexibel zu gestalten, dass sie mit dem Tempo der technologischen Entwicklungen mithalten und die daraus resultierenden Marktchancen wahrnehmen können. Durch unternehmensübergreifende Reorganisation und Optimierung von Geschäftsprozessen können Reaktionszeiten verringert und Medienbrüche vermieden werden.⁶⁴⁾

Auch die Aufgabe der IV wird sich in Zukunft ändern. Systemlandschaften setzen sich aus verschiedenen IV-Systemen zusammen, die nicht nur den funktionalen Anforderungen des Unternehmens gerecht werden, sondern auch miteinander kompatibel sein müssen.⁶⁵⁾ Eine zunehmende Entwicklung wird daher das Outsourcing von Wartung und Verwaltung der IT sein.

64) Bernotat, J.; Hoch, D.; Laartz, J.; Scherdin, A. (2001), S. 23.

65) Ließmann, H. (2000), S. 134.

6 Literatur

- Bacon, J.: Concurrent Systems - An Integrated Approach to Operating Systems, Database, and Distributed Systems, Addison Wesley New York u. a., 1993.
- Balzert, H.: Lehrbuch der Software-Technik, Spektrum Akademischer Verlag Heidelberg u. a., 2000.
- Bernotat, J.; Hoch, D. J.; Laartz, J.; Scherdin, A.: EAI - Elementarer Treiber der zukünftigen Wettbewerbsposition. In: Information Management & Consulting. 16 (2001) 1, S. 17 – 23.
- Bernstein, P. A.: Middleware: A Model for Distributed System Services. In: Communications of the ACM. 39 (1996) 2, S. 86-98.
- Britton, C.: IT Architectures and Middleware - Strategies for Building Large, Integrated Systems, Addison Wesley New York u. a., 2000.
- Coulouris, G.; Dollimore, J.; Kindbert, T.: Distributed Systems - Concept and Design, Addison Wesley New York u. a., 2001.
- Froese, J.; Moazzami, M.; Rautenstrauch, C.; Welter, H.: Effiziente Systementwicklung mit Oracle 7, Addison Wesley Bonn u. a., 1994.
- Geihs, K.: Middleware Challenges Ahead. In: Computer. (2001) June, S. 24 – 31.
- Klein, S.: Interorganisationssysteme und Unternehmensnetzwerke, Deutscher Universitätsverlag Wiesbaden, 1996.
- Krämer, B.; Papazoglou, M.; Schmidt, H.-W.: Information Systems Interoperability, John Wiley & Sons Inc. New York u. a., 1998.
- Ließmann, H.: Schnittstellenorientierung und Middleware-basierte Busarchitekturen als Hilfsmittel zur Integration heterogener betrieblicher Anwendungssysteme, Nürnberg, 2000.
- Linthicum, S.: Enterprise Application Integration, Addison Wesley New York u. a., 1999.
- Mertens, P.: Integrierte Informationsverarbeitung 1, Gabler Wiesbaden, 2000.
- Monson-Haefel, R.: Enterprise JavaBeans; O'Reilly Sebastopol u. a., 2000.
- O. V.: Duden Informatik, Dudenverlag Mannheim u. a., 1993.
- Riehm, R.; Vogler, P.: Middleware: Infrastruktur für die Integration. In: Österle, H.; Riehm, R.; Vogler, P.: Middleware - Grundlagen, Produkte und Anwendungsmöglichkeiten für die Integration heterogener Welten, Braunschweig u. a., 1996, S. 25-135.
- Ruh, W.; Maginnis, F.; Brown, W.: Enterprise Application Integration - A Wiley Tech Brief, John Wiley & Sons Inc. New York u. a., 2000.
- Scheibenpflug, M.: IT-Investitionen in der E-Business-Ära sichern. In: Information Management & Consulting. 16 (2001) 1, S. 60 – 61.
- Serain, D.: Middleware, Springer Berlin u. a., 1999.

Silberschatz, A.; Galvin, P.: Operating Systems Concepts, John Wiley & Sons Inc. New York u. a., 1999.

Teubner, R.: Organisations- und Informationsgestaltung, Gabler Wiesbaden, 1999.

Tresch, M.: Middleware: Schlüsseltechnologie zur Entwicklung verteilter Informationssysteme. In Informatik-Spektrum 19 (1996) 5, S. 249 - 256.

Vidal, V.; Winslett, M.: Presaving Update Semantics in Schema Integration. In Communications of the ACM. (1994) 11, S. 263 - 271.

Walter, J.: Enterprise Application Integration: Anwendungsintegration durch Integrationsanwendungen. In: Diebold Management. (2001) Report Nr. 1, S. 6 – 10.

Winkeler, T.; Raupach, E.; Westphal, L.: Enterprise Application Integration als Pflicht vor der Business-Kür. In: Information Management & Consulting. 16 (2001) 1, S. 7 – 16.

Internetadressen:

A Discussion of the Object Management Architecture. Auf:

http://www.omg.org/technology/documents/formal/object_management_architecture.htm,
publiziert im Januar 1997. [06.09.2001].

DeMichiel, L.; Yalçinalp, L.; Krishnan, S.: Enterprise JavaBeans Specification Version 2.0. Auf:

<http://java.sun.com/products/ejb/docs.html>, publiziert im April 2001. [05.09.2001].

Leymann, F.; Gossmer, M.: Enterprise Application Integration - Thema Message Broker. Auf:

<http://www.informatik.uni-stuttgart.de/ipvr/as/lehre/seminar/docSS00/message-broker.pdf>,
publiziert im SS 2000. [09.08.2001].

Nehmer, J.; Sturm, P.: Middleware - Betriebssysteme der Zukunft. Auf: [http://www.syssoft.uni-](http://www.syssoft.uni-trier.de/systemsoftware/Download/Seminare/Middleware/middleware.book.html)

[trier.de/systemsoftware/Download/Seminare/Middleware/middleware.book.html](http://www.syssoft.uni-trier.de/systemsoftware/Download/Seminare/Middleware/middleware.book.html), publiziert 1998.
[09.08.2001].

Schumacher, M.; Schneider, K.: Enterprise Application Integration - Message Queuing. Auf:

<http://www.informatik.uni-stuttgart.de/ipvr/as/lehre/seminar/docSS00/mq.pdf>, publiziert im SS
2000. [09.08.2001].

Talarian: Everything you need to know about Middleware. Auf:

<http://www.talarian.com/industry/middleware/whitepaper.pdf>. [04.09.2001].

Erklärung

Ich versichere hiermit, dass ich meine Bachelorarbeit über Enterprise Application Integration – Grundlagen, Methoden und Techniken selbstständig und ohne fremde Hilfe angefertigt habe, und dass ich alle von anderen Autoren wörtlich übernommenen Stellen wie auch die sich an die Gedankengänge anderer Autoren eng anlegenden Ausführungen meiner Arbeit besonders gekennzeichnet und die Quellen zitiert habe.

Münster, 17. September 2001

(Angela Schwering)