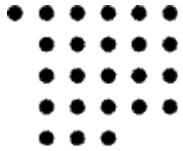

Fachhochschule Köln, Campus Gummersbach
Fachbereich Informatik
Studiengang Wirtschaftsinformatik

D i p l o m a r b e i t

Entwicklung eines Modells zur Integration eines Warenwirtschaftssystems in eine Webapplikation und Implementierung auf Grundlage von Web Services

von

Nikolas Plewa

Matrikelnummer: *11021288*

Erstprüferin: *Prof. Dr. Heide Faeskorn-Woyke*

Zweitprüfer: *Prof. Dr. Erich Ehses*

Sinzig, Februar 2004

Danksagung

Ich sehe es als meine Pflicht und gleichzeitig als große Freude, an dieser Stelle all jenen zu Danken, die mir bei der Erstellung dieser Arbeit zur Seite gestanden haben. Allen voran den Brüdern Christian und Ludger Michel, die mir als Eigentümer der Firma *Michel Development & Consulting* die Möglichkeiten und die Freiräume gegeben haben mich mit diesem Interessanten Thema intensiv befassen zu können. Frau Prof. Dr. Heide Faeskorn-Woyke, die mich während der Erstellung dieses Dokuments betreut hat und Prof. Dr. Ehrich Ehses, der gerne bereit war die Zweitkorrektur zu übernehmen.

Mein besonderer Dank gilt meinen Freunden und meiner Familie die mich durch diese Zeit und das Studium begleitet haben und in ganz besonderem Maße meiner Freundin Alexandra Unger, die über einen großen Zeitraum immer wieder auf mich verzichten musste.

Sinzig, 10.02.2004

Inhaltsverzeichnis

Abbildungsverzeichnis

Abkürzungsverzeichnis

A2A	Application to Application
ACID	Atomicity, Consistenzy, Isolation and Durability
AOL	America Online
API	Application Programming Interface
ARPA	Advanced Research Project Agency
ARPANet	Advanced Research Projects Agency Network
ASCII	American Standard Code for Information Interchange
B2B	Business to Business
BPEL4WS	Business Process Execution Language for Web Services
BPM	Business Process Management
COM	Component Object Model
CORBA	Common Object Request Broker Architecture
CRM	Customer Relationship Management
CSS	Cascading Style Sheets
DCE	Distributed Computing Environment
DCOM	Distributed Component Object Model
DNS	Domain Name System
DOM	Document Object Model
EAI	Enterprise Application Integration
EBCDIC	Extended Binary Coded Decimal Interchange Code
EJB	Enterprise Java Beans
ERP	Enterprise Resource Planning
GUI	Graphical User Interface

HTML	Hypertext Markup Language
HTTP	Hyper Text Transfer Protocol
IDL	Interface Definition Language
IIOP	Internet Inter Orb Protocol
IMAP	Internet Message Access Protocol
IP	Internet Protocol
J2EE	Java 2 Platform, Enterprise Edition
J2ME	Java 2 Platform, Micro Edition
J2SE	Java 2 Platform, Standard Edition
JAX-RPC	Java API for XML-based RPC
JAXP	Java API for XML Processing
JAXR	Java API for XML Registries
JDBC	Java DataBase Connectivity
JDK	Java Development Kit
JNDI	Java Naming and Directory Interface
JNI	Java Native Interface
JSP	Java Server Pages
JSSE	Java Secure Socket Extension
JTA	Java Transaction API
LDAP	Lightweight Directory Access Protocol
NCSA	National Center for Supercomputing Applications
NNTP	Network News Transfer Protocol
ODBC	Open Database Connectivity
OMG	Object Management Group

ORB	Object Request Broker
PHP	Personal Home Page, Professional Home Page, Hypertext Preprocessor
POP3	Post Office Protocol, Version 3
RFC	Request for Comment
RMI	Remote Method Invocation
RPC	Remote Procedure Call
SAAJ	SOAP with Attachments API for Java
SAX	Simple API for XML
SDK	Software Development Kit
SFA	Sales Force Automation
SNMP	Simple Network Management Protocol
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
SSL	Secure Socket Layer
TCP	Transmission Control Protocol
UDDI	Universal Description, Discovery & Integration
URI	Uniform Resource Identifier
URL	Uniform Ressource Locator
UTF-16	Universal Character Set Transformation Format 16-Bit
UTF-7	Universal Character Set Transformation Format 7-Bit
UTF-8	Universal Character Set Transformation Format 8-Bit
VM	Virtual Machine
VPN	Virtual Private Network
W3C	World Wide Web Consortium

WSDD	Web Service Deployment Descriptor
WSDL	Web Services Definition Language
WWW	World Wide Web
XML	eXtensible Markup Language
XSL	eXtensible Stylesheet Language
XSLT	XSL Transformations

1. Einführung

1.1 Einleitung

Customer Relationship Management (CRM) ist ein seit einiger Zeit in allen Managementpublikationen heiß diskutiertes Thema. Die Philosophie des *CRM* stellt den Kunden in den Mittelpunkt der Unternehmensaktivitäten und -prozesse, mit dem Ziel der größtmöglichen Ausschöpfung von Kundenpotentialen und damit einhergehender Umsatzmaximierung. Voraussetzung dafür ist die umfassende Kenntnis der Kundenbedürfnisse und eine daran orientierte Kommunikation mit dem Kunden. Dies Bedingt einen optimalen Überblick über alle zu einem Kunden verfügbaren Daten[1]. Dieses Ziel kann nur durch das unternehmensweite Zusammenspiel aller relevanter Unternehmensbereiche und -systeme erreicht werden. Das macht die Abkehr von der rein bereichsbezogenen bzw. funktionalen Organisation im Unternehmen, hin zu einem Denken in horizontalen, also abteilungsübergreifenden Geschäftsprozessen, notwendig[2]. Dabei ist es Aufgabe der Informationstechnik, die Prozesse mit den notwendigen Werkzeugen zu unterstützen. Neben der Unterstützung und Verbesserung der Kommunikation zwischen Kunden, Unternehmen und den an den unternehmensinternen Prozessen beteiligten, liegt dabei ein Schwerpunkt in der Ausnutzung bereits vorhandener Daten. Diese schlummern häufig in den unterschiedlichsten Systemen verschiedener Abteilungen entlang der betrieblichen Wertschöpfungskette. Es gilt, die vorhandenen Quellen zu erschließen und in die Unterstützung aller horizontal integrierten Geschäftsprozesse einzubinden.

Enterprise Application Integration (EAI) ist der Oberbegriff für eine ganze Reihe unterschiedlicher Integrationsstrategien und -lösungen. Er fasst alle jene Anstrengungen zusammen, die unternommen werden, um Softwarelösungen im Rahmen eines durchgängigen Geschäftsprozesses zur Kooperation zu bewegen, die getrennt voneinander, nicht zur gemeinsamen

1 S. [ITManagement11_2001], Der Kunde ist König, S.22
2 Vgl. [Meyer2003], S.V

Arbeit konzipiert wurden[3]. Das bedeutet, dass vorhandene Applikationen weiter genutzt werden können und damit bereits getätigte Investitionen geschützt bleiben. Es werden zusätzliche Schnittstellen ergänzt, die es ermöglichen, dass die vorhandenen Applikationen Daten automatisiert, ohne manuelle Eingriffe austauschen können[4].

1.2 Gegenstand und Aufbau dieser Arbeit

Das Ziel dieser Arbeit ist es, die notwendigen Überlegungen und die Schritte beim Aufbau einer konkreten Integrationslösung zu dokumentieren. Konkrete Aufgabe dabei ist die Entwicklung einer Integrationslösung für ein existierendes Warenwirtschaftssystems im Rahmen der Neuimplementierung einer webbasierten CRM-Applikation. Dabei soll die Basis einer Integrationsplattform geschaffen werden, die eine Wiederverwendung und Erweiterung in zukünftigen Projektschritten, über die in diesem Schritt miteinander zu integrierenden Systeme hinaus, ermöglicht. Es soll für den Entwickler der vorgelagerten Schichten, welcher die Integrationslösung in Zukunft verwendet, möglichst transparent sein, aus welchen Backend-Systemen die von ihm benötigten Daten stammen.

Zu Beginn soll der Begriff Integration genau definiert und Arten der Integration benannt werden. Anschließend wird dargelegt, welche Entwicklungen dazu geführt haben, dass dieses nicht ganz neue Thema wieder eine derartige Aktualität erlangt hat und welche Rolle eBusiness, eCommerce und CRM in diesem Zusammenhang spielen.

Im folgenden Kapitel werden die möglichen Lösungen für die Realisierung der Anbindung beschrieben. Anschließend wird der für das Projekt verwendete Lösungsweg skizziert und die zur Umsetzung benötigten Softwarekomponenten vorgestellt. Am Beispiel von aus dem Warenwirtschaftssystem benötigten Adress- und Rabattinformationen wird schließlich die Implementierung der Anbindung erläutert und im letzten Kapitel die gemachten Erfahrungen zusammengefasst. Abschließend wird

3 Vgl. [Keller2002], S.5
4 S. [Nußdorfer2000], 1.4.2 Investitionsschutz dank EAI

eine Einschätzung für die weitere Entwicklung dieses Problemfeldes gegeben und Ausblick für das beschriebene Projekte geschaffen.

1.3 Integration und Arten der Integration

Laut [Starke2002] versteht man unter *Integration* die Einbindung bestehender Systeme in einen neuen Kontext[5]. Dies erfolgt mit dem Ziel, automatisierter, unterbrechungsfreier Geschäftsprozesse, um dadurch Beschleunigungs- und damit Kosteneffekte zu erzielen. Man unterscheidet grundsätzlich zwei Arten der Systemintegration[6]:

Interne Integration beschäftigt sich mit dem Austausch von Daten zwischen den internen Geschäftsanwendungen eines Unternehmens und wird daher auch *Application to Application* (A2A) Integration genannt.

Externe Integration ist die Verbindung der betrieblichen Wertschöpfungsketten von Unternehmen mit Hilfe von Kommunikationstechnik, vorzugsweise dem Internet oder Internettechnik. Besser bekannt ist diese Art der Integration unter dem Begriff *Business to Business* (B2B) Integration. Im Rahmen von *B2B* ist es zum Beispiel möglich, dass ein Warenwirtschaftssystem eines Unternehmens automatisch eine Bestellung im System eines Zulieferers platziert.

Die Idee, die im betrieblichen Alltag verwendeten Softwareprodukte miteinander zu integrieren, ist bei weitem kein Novum. So konnten sich die heute weit verbreiteten *Enterprise Resource Planning*-Systeme vor allem deshalb am Markt durchsetzen, weil sie integrierte Lösungen für betriebliche Standardaufgaben wie Rechnungswesen, Logistik, Produktion und Personalwirtschaft darstellen. Bis zur Einführung von ERP-Systemen wurden diese Aufgaben häufig mit Hilfe völlig unabhängiger Systeme erfüllt. Bei dem in diesem Bereich führenden Produkt R/3 des Herstellers SAP werden alle für diese Zwecke benötigten Daten redundanzfrei in einer zentralen Datenbank gehalten[7]. Doch kein Softwarehersteller kann mit seinen Produkten alle

5 S. [Starke2002], S.166
6 S. [Nußdorfer2000], 1.1 Integration statt Anwendungs-Inseln
7 Vgl. [Victor2001], S.7

betrieblichen Tätigkeiten abdecken. Selbst SAP ist heute, trotz einer 30 jährigen Entwicklungsgeschichte des R/3 Systems, immer noch damit beschäftigt, Basisfunktionalitäten aus dem Bereich ERP zur Reife zu bringen[8]. Außerdem griffen die von den ERP-Herstellern verfolgten Integrationsansätze oft zu kurz. So war es mit R/3 lange Zeit nicht möglich, fremde, also andere Systeme als R/3, zu integrieren[9].

Durch die Verbreitung des Internet und die dadurch mögliche globale Kommunikation hat der Bedarf nach stärkerer Systemintegration in jüngster Zeit noch einmal massiv zugenommen. Die Art und die Geschwindigkeit der geschäftlichen Aktivitäten haben sich aufgrund der verfügbaren Technologien fundamental verändert. Die existierenden Geschäftsmodelle müssen überdacht, teilweise völlig neu gestaltet und an die neuen Möglichkeiten angepasst werden. Der weltumspannende Charakter des Internet hat nicht nur Einfluss auf die Geschwindigkeit der Abläufe, sondern auch auf die zu beobachtende Konkurrenz. Unternehmen sehen sich plötzlich mit Wettbewerbern konfrontiert, die vorher für sie keine Rolle spielten. Von dieser Situation profitiert in erster Linie der Kunde, der mit dem Anspruch auf eine größere Produktauswahl und -qualität, niedrigeren Preisen, eine schnellere Auslieferung und einen ständig erreichbaren Kundenservice reagiert. Eine Forderung, auf die von den Unternehmen nur mit Reduzierung der Kosten und Effektivierung des innerbetrieblichen Informationsflusses reagiert werden kann. Das zwingt sie, immer enger mit Lieferanten und Partnern zusammenzuarbeiten und sich mit diesen immer besser abzustimmen. Es entsteht ein Integrationsbedarf über Unternehmensgrenzen hinweg, der erst durch Ausnutzung der Möglichkeiten der internetbasierten Technologien befriedigt werden kann[10].

8 S. [WestPlosHoff2001/III], S.33
9 Vgl. [SAPINFO61]
10 Vgl. [ITManagement11_2001], Netze Knüpfen - Integration interner Applikationen, S.12

1.4 Internet und webbasierter Applikationen

Durch die zunehmende Rolle des Internet und der Internettechnologie für die interne und externe Kommunikation und die Entstehung neuer Kooperationsformen wie z.B. elektronische Märkte, elektronische Beschaffung aber auch Internet Präsenzen und Onlineshops, finden jene Technologien, auf denen das Internet basiert, zunehmend Einzug in die interne Unternehmensinfrastruktur. So ist es heute Gang und Gebe, dass auch firmeninterne Netze auf dem Internet Kommunikationsprotokoll TCP/IP basieren. Durch das Internetprotokoll wurden andere in der Vergangenheit vielfach verwendete Protokolle wie IPX/SPX und X.25 fast vollkommen aus den Unternehmen verdrängt. Inspiriert durch die Client/Server-Architektur des Internets finden immer mehr Applikationen Anwendung, die als Präsentationsschicht vom Server dynamisch generierte HTML-Seiten verwenden, die in einem Web Browser auf dem Client dargestellt werden. Die Kommunikation zwischen Client und Server erfolgt dabei über das zum TCP/IP-Protokollstack gehörende HTTP-Protokoll.

Im folgenden Abschnitt soll dargestellt werden, wie es zu dem kam, was wir heute als „State-of-the-Art" der Internettechnik verstehen. Den Vorzügen von webbasierten Applikationen wird sich im Anschluß der Abschnitt 1.4.2 widmen.

1.4.1 Geschichte und Techniken des Internet

Vom ARPA-Net zum Internet

Das Internet ist ein globales aber dezentral organisiertes Netzwerk, das aus vielen unabhängigen Teilnetzen besteht[11]. Diese Idee reicht in die 1960er Jahre zurück, die Zeit des Kalten Krieges, und lässt sich eindeutig auf das Engagement des US Militärs zurückführen. Das dezentrale Konzept des Internets entstand damals im Rahmen der Bemühungen, ein Netzwerk zu schaffen, dessen grundsätzliche Funktion auch dann noch gewährleistet bleibt, wenn Teile des Netzes ausgefallen sind. Eine Situation wie man sie

11 S. [WormGeschichteWeb], Was ist das Internet

im Kriegsfall befürchtete. Das von der US Air Force 1964 initiierte Projekt scheiterte aber und wurde nie realisiert.

Das Konzept wurde allerdings 1966 von der *Advanced Research Projects Agency* (ARPA) wieder aufgegriffen, einer wissenschaftlichen Einrichtung deren Forschungsergebnisse wiederum in militärische Projekte einfließen[12]. Unter dem Namen ARPA-Net wurden Ende 1969 vier Rechnersysteme an verschiedenen Standorten in den USA miteinander verbunden. Es handelte sich um eine SDS SIGMA 7, eine SDS940/Genie, eine IBM 360/75 und eine DEC PDP-10[13], also heterogene Hardwareplattformen mit unterschiedlichen Betriebssystemen[14]. Als Kommuikationsprotokoll fand zu Beginn noch das *Network Control Protocol* (NCP) Verwendung. Erst 1974 wurde der detaillierte Entwurf des *Transmission Control Protocol* (TCP) veröffentlicht[15]. In dessen Entwurf wird auch zum ersten Mal der Begriff *Internet* verwendet[16].

In den 1970er Jahren stieg die Anzahl der an das ARPA-Net angeschlossenen Rechner, vorwiegend aus dem Forschungsbereich, ständig. Nach wie vor handelte es sich dabei um sehr unterschiedliche Typen[17]. Im Jahre 1978 erfolgte die Trennung in TCP als verbindungsorientiertes Protokoll und dem *Internet Protokoll* (IP) als darunter liegendes verbindungsloses Protokoll zum Transport der einzelnen Datenpakete[18]. Dies wurde notwendig, um für die verschiedenen verwendeten Leitungswege - Wahlleitungen, Standleitungen etc. - des immer größer werdenden Netzes ein einheitliches, standardisiertes Datenübertragungsschema zu schaffen. Das TCP/IP Protokoll, so wie es heute verwendet wird.

Anfang der 1980er Jahre spaltete sich der militärische Teil des ARPA-Net ab[19]. Der immens gewachsene zivile Teil des Netzes umfasste mittlerweile

12 Vgl. [Münz2001], Entstehung des Internet
13 Vgl. [HobbesInternetTimeline], Stand: 20.12.2003
14 Vgl. [WormGeschichteWeb], Geschichte des Internet
15 S. [WormGeschichteWeb], TCP/IP - Transfer Control Protokoll/Internet Protokoll
16 Vgl. [GeschichteDesInternet.com], 05/1974
17 Vgl. [Münz2001], Entstehung des Internet
18 Vgl. [HobbesInternetTimeline], Stand: 20.12.2003
19 Vgl. [Münz2001], Entstehung des Internet

ca. 4000 Rechner. Im gleichen Jahr erfolgte die vollständige Umstellung auf TCP/IP und der *Domain Name Service* (DNS)[20], ein serverbasiertes System zur Abbildung von Hostnamen auf die zur Adressierung der einzelnen Rechner verwendeten IP-Adressen wurde vorgestellt. Ein neues Hauptnetz zum Anschluss kleinerer Netze, ein so genannter Backbone, entstand. Dieses Netz mit dem Namen NSF-Net durfte die Struktur der ARPA mit benutzen und wurde von der US-Regierung finanziert. Die Bezeichnung *Internet* ist seit dieser Zeit gebräuchlich, und im Folgenden wurden Teilnetze auf der ganzen Welt daran angeschlossen[21]. Im Jahr 1987 sind es 27.000 verbundene Rechner[22].

Das World Wide Web

Killerapplikationen des zu diesem Zeitpunkt bereits sehr stark gewachsenen Netzes waren Ende der 1980er Jahre E-Mail und Usenet. Letzteres entspricht einer Ansammlung elektronischer schwarzer Bretter zu den verschiedensten Themen[23]. Mit dem Ziel, Dokumente von allgemeinem Interesse für Mitglieder von Forschungseinrichtungen zugänglich zu machen und leicht miteinander verknüpfen zu können, entwickelte der gebürtige Brite Tim Berners-Lee 1991 am europäischen Kernforschungszentrum CERN in Genf ein Hypertextsystem mit einem sehr einfachen Bedienungskonzept. Das Projekt erhielt den Namen *Word-Wide-Web* (WWW), da es Verknüpfungen von Dokumenten möglich macht, die auf Servern im ganzen Internet und damit auf dem ganzen Globus verteilt liegen. Das von Berners-Lee entwickelte Konzept des WWW basierte von Anfang an auf den folgenden drei Säulen[24]:

1. Die Auszeichnungssprache für die im WWW verwendeten Hypertext-dokumente mit dem Namen *Hyper Text Markup Language*, kurz HTML.

20 Vgl. [GeschichteDesInternet.com], 1983 und 11/1983
21 Vgl. [Münz2001], Entstehung des Internet
22 S. [GeschichteDesInternet.com], 1987
23 Vgl. [WormGeschichteWeb], Geschichte des Internet
24 Vgl. [Münz2001], Entstehung des World Wide Web

2. Das *Hypertext Transfer Protokoll* (HTTP) definiert die Kommunikation zwischen Web Client und Web Server. Es ist ein einfaches verbindungsloses Protokoll, welches zum Transport der Inhalte auf TCP/IP aufsetzt.
3. Sogenannte *Universal Resource Identifier* (URI) spezifizieren dabei beliebige Datenquellen im Netz eindeutig.

Das Team um Berners-Lee bemühte sich in der folgenden Zeit unermüdlich um die Verbreitung des neuen Systems. Allerdings begann das WWW erst mit der Verfügbarkeit von grafikfähigen Web Clients seinen Siegeszug durch das Internet. Die von Studenten 1992 am National Center for Supercomputing Applications (NCSA) der Universität von Illinois entwickelte Software Mosaic war der erste verbreitete Web Client, auch Browser genannt, mit grafischer Oberfläche. Der an der Entwicklung von Mosaic beteiligte Marc Andreesen gründete schließlich die Firma Netscape, die einen Browser entwickelte, um diesen kommerziell zu vertreiben. In den Jahren 1995 und '96 hatte der Netscape Browser einen Marktanteil von über 90%.

Ende 1994 gründete Tim Berners-Lee das *World Wide Web Consortium* (W3C) am Computer Science Laboratory des MIT, welches auch heute noch für die Definition und Überwachung von *HTML* und anderen Internetstandards zuständig ist. Der Microsoft-Konzern der mit seinen Produkten Mitte der 1990er Jahre bereits den PC-Betriebssystemmarkt dominierte, unterschätzte zunächst die Potentiale des Internet und des WWW. Aufgerüttelt durch die Erfolge, die die Firma Netscape sowohl im Internet als auch an der Börse feierte, begann der Software-Riese mit großem Aufwand einen eigenen Web Browser zu entwickeln. Es entbrannte ein heftiger Kampf um Marktanteile zwischen den beiden Kontrahenten Netscape und Microsoft, in dem Netscape schließlich unterlag[25]. Im Jahre 1998 wurde der angeschlagene Browserhersteller für 4,2 Mrd. US Dollar vom mittlerweile weltweit führenden Onlinedienst *America Online* (AOL) übernommen. Unter dem Dach von AOL führte der Web Client aber jahrelang ein Schattendasein. Im Juli 2003 stellte AOL die Weiterentwicklung des Netscape-Browsers endgültig ein, nachdem der Quellcode bereits einige Zeit

25 Vgl. [Münz2001], Entstehung des World Wide Web

zuvor Eingang in das Open-Source-Project *Mozilla* gefunden hatte. In Rahmen dieses Projektes wird der Browser ebenfalls unter dem Namen *Mozilla* nun weiter entwickelt[26]. Am 30. Juni 2003 wurde die Version 1.4 offiziell veröffentlicht. Der Browser ist für eine ganze Reihe von Plattformen verfügbar: Win32, Mac OS X, Linux, AIX, Irix, OS/2 und Sun Solaris, um nur einige zu nennen. Außerdem werden Varianten in verschiedenen Sprachen wie englisch, deutsch und französisch, aber auch dänisch, griechisch, polnisch und sogar koreanisch und chinesisch angeboten[27].

Der Apache Web Server und die Folgen

Während der Web Browsermarkt durch den Kampf der beiden Konkurrenten Mitte der 1990er Jahre stark in Bewegung geriet, gibt es auf Seiten der Web Server seit damals bis heute einen klaren Marktführer. Im Februar 1995 war der *NCSA HTTP Daemon*, entwickelt am National Center for Supercomputing Applications der Universität Illinois, die am meisten verbreitete Serversoftware für das World Wide Web. Da der federführende Entwickler Rob McCool die NCSA aber bereits Mitte 1994 verlassen hatte, stockte die Weiterentwicklung der verbreiteten Software. Viele Administratoren entwickelten daher ihre eigenen Erweiterungen und Patches, die aber Gefahr liefen, niemals im Rahmen einer neuen Version des Servers veröffentlicht zu werden. Eine kleine Gruppe dieser Entwickler, die in regelmäßigem E-Mail Kontakt miteinander stand, beschloss schließlich, ihre Änderungen und Erweiterungen zu koordinieren. Im April 1995 veröffentlichte die Gruppe die erste Version ihrer Software auf Basis des *NSCA httpd*[28]. Der Name des Produkts ist ein Wortspiel. Aufgrund der vielen Flicken und Erweiterungen sprechen die Entwickler von „a patchy server", der von nun an *Apache HTTP Server* heißen sollte. Weniger als ein Jahr nach Gründung der *Apache Group* überholte der *Apache* den *NCSA httpd* als Nummer eins der Web Server im Internet und hält diese Position bis heute[29]. Diese marktführende Stellung verdankt der freie *Apache* vor allem

26 Vgl. [SpiegelNetzRuhe2003], Stand 27.07.2003
27 S. [Mozilla.org2003]
28 Vgl. [Apache.org_About], Stand: 20.12.2003
29 S. [Kefk.net_Apache], Stand: 20.12.2003

der Tatsache, dass es sich um ein robustes und zuverlässiges Produkt handelt, dass sich zu recht rühmt, kommerzielle Reife erlangt zu haben. Trotzdem ist der *Apache* kostenlos verwendbar. Darüber hinaus handelt es sich um ein *Open-Source*-Projekt, was bedeutet, dass der Quellcode ebenfalls zugänglich ist und von Jedermann in Augenschein genommen und verbessert werden kann, was zusätzlich zur Stabilität und Sicherheit des Produktes beiträgt.

Im Jahr 1999 gründeten Mitglieder der *Apache Group* die *Apache Software Foundation*, um zukünftigen Entwicklungen einen stabilen organisatorischen, rechtlichen und finanziellen Rahmen geben zu können. Was für den *Apache HTTP Server* entstand, gibt heute einer Vielzahl von ehrgeizigen *Open-Source*-Projekten ein Zuhause. Die *Apache Software Foundation* tritt mit dem Anspruch an, qualitativ hochwertige Softwareprodukte zu liefern, die in den jeweiligen Bereichen Maßstäbe setzen und trotzdem von Jedermann frei und kostenlos verwendet werden dürfen. Viele Produkte der Foundation werden zu Referenzimplementierungen für die jeweiligen Anwendungsprobleme[30].

Standardisierungsbemühungen im Wolrd Wide Web

Das *W3*-Konsortium setzt seine Arbeit über die Grenzen von *HTML* fort, indem es Standards für Erweiterungen wie *Cascading Style Sheets* (CSS) definiert und sich beispielsweise mit der Entwicklung neuer Architekturen für das WWW, wie der *Extensible Markup Language* (XML) als strukturiertes Datenaustauschformat und dem *Document Object Model* (DOM) als Plattform- und Programmiersprachenunabhängige Schnittstelle für den Zugriff auf Webdokumente, beschäftigt[31]. Für das *W3C* stehen dabei nach wie vor Ziele im Vordergrund, die auch schon in den Gründertagen des Internet verfolgt wurden. Das Web soll von allen Menschen nutzbar sein, gleichgültig welche Hard- oder Software sie verwenden[32]:

30 Vgl. [Apache.org_About], Stand: 20.12.2003
31 S. [W3CAchitecture_2003], Stand 27.07.2003
32 Vgl. [W3CSieben_2003], Stand 27.07.2003

„W3C als eine herstellerunabhängige Organisation, fördert die Interoperabilität, indem es offene, nicht proprietäre Computersprachen und Protokolle entwirft und damit die in der Vergangenheit vorherrschende Marktspaltung verhindert. Dies wird durch einen Konsens innerhalb der Industrie und durch die Ermunterung zu öffentlichen Diskussionen erreicht."

1.4.2 Vorzüge und Nachteile webbasierter Applikationen

Die immer größere Verbreitung von Webapplikationen führt zwangsläufig zu einer Verschiebung der Bedeutung von Client- und Server-Betriebssystemen. Dieser Trend wurde ausgelöst durch die Vorzüge, die die Verwendung der Techniken des World Wide Web erkennen lassen[33]:

- Die Installation von Client Software (GUI) entfällt, was im Fall von Tausenden von Arbeitsplätzen einen enormen Fortschritt bei der Wartung der Software darstellt.
- Durch Webtechnologie können einfachste Endgeräte zum Einsatz kommen, welche keine eigene Peripherie oder umfangreiche Speicherungsmechanismen aufweisen müssen.
- Für mobile oder von zu Hause aus arbeitende Angestellte stehen heute praktisch überall Internet-Zugangspunkte zur Verfügung, so dass "mobile", resp. "home office" sehr einfach zu realisieren ist.
- Dank einer sehr serverlastigen Architektur kann durch die Erweiterung der Web Server-Leistung die Gesamtleistung des Systems auf einfache Weise gesteigert werden (Skalierbarkeit).

Außerdem sind viele Nutzer mittlerweile vertraut mit der Benutzung eines Web Browsers und finden sich von daher leicht in einer browserbasierten Applikation zurecht. Allerdings bringen diese auch einige Restriktionen mit sich, über die man sich bei der Betrachtung im klaren sein muss. Bedienelemente können, je nach Browser, unterschiedlich dargestellt

33 Vgl. [ITManagement11_2001], Der Kunde ist König – Wie Internettechnologie hilft, Kundenwünsche optimal zu erfüllen, S.24

werden, was beim Benutzer durchaus zu Verwirrung führen kann. Nicht jede Applikation lässt sich auf der Basis von Standard HTML realisieren, da es für viele Dinge in HTML-Formularen keine Eingabemöglichkeiten gibt. So ist es z.B. mit reinem HTML weitgehend unmöglich, eine Layout-Software zu entwickeln, da es keine Möglichkeit gibt, grafische Elemente zu bearbeiten und diese frei auf einer Fläche zu positionieren[34].

Des weiteren bietet der im Internetbereich übliche Einsatz von offenen Standards wie Java, HTML und XML die ideale Plattform für die schnelle und einfache Anbindung von weiteren Applikationen[35].

1.5 Electronic Business und Electronic Commerce

Die Bemühungen, Geschäftstätigkeiten mit Internettechnik zu unterstützen und auch über das Internet abzuwickeln, wird unter den Begriffen eBusiness und eCommerce subsummiert. Die Vorteile elektronischer Unterstützung von Geschäftsprozessen sollen im Folgenden erläutert werden. Zunächst werden aber diese und andere häufig verwendeten Begriffe erklärt und gegeneinander abgegrenzt[36]:

„*Electronic Business* bezeichnet als Oberbegriff unterschiedliche Formen des Einsatzes elektronischer Kommunikations- und Kooperationsmechanismen zur Realisierung, Unterstützung und Optimierung von Geschäftsprozessen. Die Begriffe *Electronic Business*, *eBusiness*, und *elektronisch realisierte Geschäftsabläufe* sind Synonyme."

Über die Erläuterung des Begriffs hinaus finden sich in der Literatur verschiedene Definitionen der Teilbereiche von eBusiness. Dabei wird eCommerce aber durchweg als wesentlicher Teilbereich von eBusiness betrachtet. So werden in [FriKarKno2001] die Konzepte und Anwendungen von eBusiness-Lösungen beispielsweise in die folgenden fünf Kerngebiete gegliedert[37]:

---

34 Vgl. [MicrosoftDesign4Web], Stand: 27.07.2003
35 S. [ITManagement11_2001], Der Kunde ist König – Wie Internettechnologie hilft, Kundenwünsche optimal zu erfüllen, S.25
36 Aus [Zwißler2002], S.9
37 S. [FriKarKno2001], S.7

- *Supply Chain Management* für das reibungslose Zusammenspiel aller betrieblichen Abläufe von der Bestellung bis zur Auslieferung,
- *Enterprise Resource Management* für die Optimierung aller administrativen Geschäftsprozesse,
- *Business Information Management* zur effizienten Nutzung und Vernüpfung aller relevanten Informationen im Unternehmen,
- *Customer Relationship Management* für die reibungslose Koordination von Vertrieb, Marketing und Kundenservice, zugeschnitten auf die individuellen Bedürfnisse der Kunden, und schließlich
- *eCommerce*, der elektronische Handel, der sich immer mehr in Richtung Mobile Commerce entwickelt, also mit Hilfe mobiler Endgeräte hin zu Mobile Shopping, Mobile Banking, Mobile Booking und Mobile Brokerage.

So wird klar, dass eCommerce, CRM und SCM Bezeichnungen für verschieden Teilaspekte des Themas eBusiness darstellen, die alle auf den unter 1.4.1 erläuterten technischen Voraussetzungen aufsetzen. Die Vorteile, die elektronisch realisierte Geschäftsprozesse mit sich bringen, werden im folgenden Abschnitt kurz zusammengefasst.

1.5.1 Nutzeffekte von Electronic Business

Die Einführung von eBusiness-Systemen geht zunächst mit nicht unerheblichem finanziellem, personellem und organisatorischem Aufwand einher. Häufig müssen vorhandene Prozesse angepasst oder sogar vollständig neu definiert werden. Auch wenn durch die Integration der vorhandenen Systeme die Kosten für die Umsetzung der eBusiness-Strategie reduziert werden können, stellt sich dabei unweigerlich die Frage nach Effekten, die die zu tätigenden Investitionen rechtfertigen.

Die Vorteile elektronisch realisierter Geschäftsabläufe lauten nach [Zwißler2002][38]:

38 S. [Zwißler2002], S.11

Höherer Durchsatz:
Durch die elektronische Verarbeitung kann die Leistungsfähigkeit extrem gesteigert werden.

Geringere Kosten:
Reale Dokumente können durch elektronische Datensätze ersetzt und damit kostengünstiger verarbeitet und verwaltet werden.

Größere Robustheit:
Die automatische Verarbeitung und Überprüfung ist häufig weniger fehleranfällig als eine manuelle Bearbeitung.

Höhere Transparenz:
Vorgehensweisen und Entscheidungsgrundlagen können auf Wunsch leicht offengelegt werden.

Bessere Interaktivität:
Elektronische Medien erlauben schnelle und direkte Reaktionen.

Expliziter Mehrwert:
Durch Verknüpfung unterschiedlicher Daten und Verarbeitungsschritte können innovative, bisher nicht verfügbare Funktionalitäten, realisiert werden.

Es ist also davon auszugehen, das die hier vorgestellten Effekte auch hohe Anfangsinvestiotionen in der Regel legitimieren.

1.5.2 CRM - Kerngebiet des E-Business

CRM ist ein neuer Trend im Management, eine neue Philosophie der Unternehmensführung und -ausrichtung. Diese lässt sich aber nur mit Hilfe elektronischer Medien sinnvoll und vollständig implementieren, was die Nähe zu eBusiness deutlich unterstreicht. Da die Orientierung an den Bedürfnissen des Kunden kein ganz neuer Gedanke ist, stellt sich die Frage, warum man also gerade jetzt wieder in so hohem Maße darauf aufmerksam wurde?

Das Internet bietet dem Kunden ganz neue Möglichkeiten: Niemals in der Vergangenheit war es so einfach, Preise zu vergleichen und sich über Produkte zu informieren wie seit der breiten Kommerzialisierung des neuen Mediums. Nie war es so einfach von einem Anbieter zu einem anderen zu wechseln. CRM gilt als die Antwort auf diese neue Herausforderung.

CRM - Managementtrend zur Kundenbindung

Die in München ansässige Meta Group definiert CRM wie folgt[39]: „Geschäftsphilosophie zur Optimierung der Kunden-Identifizierung, Kundenbestandssicherung sowie des Kundenwertes. Die Umsetzung der Philosophie erfolgt durch die Automatisierung aller horizontal integrierten Geschäftsprozesse, die über eine Vielzahl von Kommunikationskanälen Vertrieb, Marketing und Kundenservice involvieren."

Zunächst geht es im CRM also um die Identifizierung von Kunden und Interessenten, also festzustellen, wer sie und was ihre Bedürfnisse sind. Die möglichst vollständige Erfüllung der individuellen Bedürfnisse der bestehende und potentiellen Abnehmer soll zur Kundenbestandssicherung beitragen und damit zur Umsatzmaximierung anhand der vorhandenen Kundenbasis führen. CRM umfasst aber laut Definition auch die Optimierung des Kundenwertes. Dabei wird der Tatsache Rechnung getragen, dass Umsatzmaximierung nicht Gewinnmaximierung bedeutet. Tatsächlich ist es denkbar, dass ein Unternehmen alle Bedürfnisse seines Kundenstamms optimal befriedigt, aber trotzdem keine Gewinne erwirtschaftet. Das ist der Fall, wenn die Kosten die erzielten Gewinne übersteigen. Im nächsten Schritt muss es daher darum gehen, die vorhandene Kundenbasis nach Umsätzen, Deckungsbeiträgen und Potentialen zu segmentieren und die hochprofitablen von den weniger profitablen und den defizitären Kunden zu trennen. Hat man sich schließlich einen entsprechenden Überblick verschafft, erhält man die Möglichkeit aus verlustreichen wieder profitable Kunden zu machen, indem man die Kundenbindungsmaßnahmen an die verschiedenen Segmente anpasst, also darauf abgestimmte Angebots- und Servicepakete entwickelt. Ziel von CRM ist es also auch, das Verständnis zu

39 S. [CybizCRM2-6/2000], CRM?->Nein danke!, S.18

fördern, welche Kundenbeziehungsmodelle ökonomisch zum Erfolg führen; weg von undifferenzierten und damit unökonomischen Kundenbindungsmaßnahmen. Häufig bedeutet das auch weg vom klassischen Massenmarketing (siehe dazu auch [Rapp2001]).

Arten und Aufbau von CRM-Software-Lösungen

Aufbauend auf der oben genannten Definition, unterscheidet die Meta Group drei verschiedene CRM-Arten mit den entsprechenden Software-Lösungen[40]:

1. **Operationales CRM:** Umfasst Lösungen für Sales Force Automation (SFA), Marketing Automation, Call Center/Customer Interaction Center. Sie müssen in bereits vorhandene Back-Office-Lösungen integriert werden.
2. **Analytisches CRM:** Umfasst Lösungen im Umfeld von Data Mining für die Analyse der im operationalen CRM-Bereich generierten Daten. Ziel ist das Business Performance Management.
3. **Kollaboratives CRM:** Umfasst Kanäle (Kommunikationslösungen wie E-Mail. Fax, Web/E-Commerce, Computer Telephony Integration/CTI etc.). Sie ermöglichen eine direkte Interaktion zwischen Kunden und Unternehmen (inklusive des Managements und der Synchronisation der Kanäle).

Diese drei Kategorien umfassen alle beschriebenen Aspekte von CRM und gliedern diese sinnvoll, wobei in der Praxis mit Lösungen zu rechnen ist, die nur Teile aus diesen Kategorien implementieren.

2. EAI und Möglichkeiten der Umsetzung

Im ersten Kapitel wurden die Hintergründe für die anhaltende Aktualität des Themas Integration dargelegt. Dies umfasste eine allgemeine Betrachtung der Unterstützung von Geschäftsprozessen, der zugrundeliegenden Daten und der Internet- bzw. E-Business-Problematik, insbesondere des *Customer*

40 S. [CybizCRM2-6/2000], CRM?->Nein danke!, S.18

Relationship Managements. Im zweiten Kapitel folgt nun eine Beschreibung der architektonischen und funktionalen Aspekte von Integrationslösungen. Da es sich hierbei um ein sehr weites Feld handelt, werden immer wieder bestimmte, im Allgemeinen besonders wichtige Bereiche herausgegriffen und vertieft.

2.1 Architekturkonzepte

Zu Beginn werden einige grundlegende Konzepte behandelt, die für alle Formen von Integrationslösungen relevant sind. Dabei geht es insbesondere um die folgenden Fragen:

- Auf welcher Ebene findet die Integration statt?
- Wie läuft die Kommunikation ab und welche Infrastruktur wird ggf. benötigt?
- Welche zusätzlichen Probleme werden durch die Integrationslösungen inhärente Verteilungsproblematik aufgeworfen?
- Welche Sicherheitsaspekte spielen bei Integrationslösungen eine Rolle?
- Wie kann die vollständige Durchführung angestoßener Datenverarbeitungsprozesse garantiert werden?

2.1.1 Integrationsmodelle

Betrachtung von Mehrschichtarchitekturen

Es ist typisch für moderne Softwaresysteme, dass die Anwendung in logische Schichten gegliedert wird. Man spricht dabei auch von Mehrschichten-, oder *n-Tiere-Architekturen*. Diese logischen Schichten lassen sich in der Regel in eine der drei folgenden Kategorien einordnen[41]:

41 Vgl. [Wutka2001], S.40

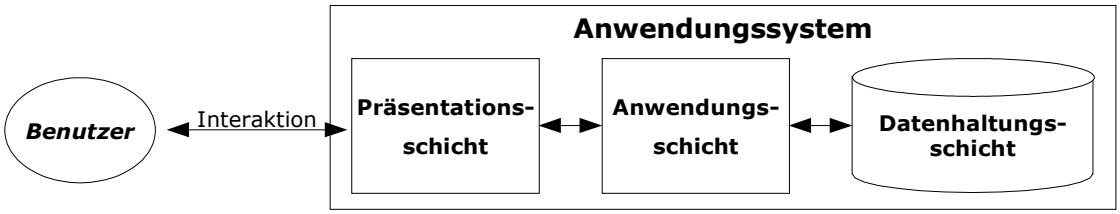

Abbildung 1: n-Tier Architektur: Typische Gliederung in Schichten

Wie *Abbildung 1* zeigt, interagiert der Benutzer ausschließlich mit der *Präsentationsschicht*. Typische Beispiele für diese Kategorie sind GUI-Applikationen oder Webseiten. Aber auch Reportgeneratoren zählen zu dieser Rubrik.

In der *Applikationsschicht*, Mittelschicht oder Fachkonzeptschicht befindet sich die Geschäftslogik. Diese implementiert alle für die Unterstützung der Geschäftsprozesse wichtigen Funktionen und stellt diese der Präsentations-schicht zur Verfügung. Im Idealfall abstrahiert sie damit vollständig vom Zugriff auf die niedrigeren Schichten. Gleichzeitig sorgt sie für die Wahrung der Datenintegrität, da nur über diese Schicht mit den darunterliegenden kommuniziert wird.

Die *Datenhaltungsschicht* sorgt für die Speicherung der von der Anwendung verwalteten Informationen. Hierzu werden vorzugsweise Datenbanken eingesetzt. Aber auch die Verwendung von flachen Dateien ist je nach Anwendungsfall gebräuchlich.

Diesen drei Ebenen eines Softwaresystems entsprechend unterscheiden sich die folgenden Integrationsmodelle[42]:

Integration über Präsentationsschicht

Hierbei erfolgt die Integration unter Verwendung der vorhandenen Benutzerschnittstelle. Entweder wird diese in eine andere eingebunden oder es wird ihr eine neue Oberfläche gewissermaßen 'übergestülpt'. Ein typisches Beispiel für die erste Variante sind Web Interfaces, die über ein Frameset in eine neue Webapplikation eingebunden werden. Für den

42 Vgl. [Kaib2002], S. 60

zweiten Fall ist die sogenannte GUIfizierung von Hostanwendungen typisch, wobei die textbasierten Bildschirmdarstellungen ausgewertet und entsprechende Masken z.B. im Rahmen einer Windowsanwendung generiert werden.

Abgesehen von dem im Internet-/Intranetbereich mittlerweile sehr weit verbreiteten Portalkonzept, bei dem die Teilanwendungen speziell auf diese Form der Integration ausgelegt werden, stellt dieses Modell eher eine Notlösung für den Fall dar, das keine anderen Möglichkeiten zur Verfügung stehen. Daher wird im Folgenden auch nicht weiter auf diese Variante eingegangen.

Integration über Datenhaltung

In diesem Falle wird die Integrationsproblematik durch gemeinsame Nutzung der Datenhaltungsschicht gelöst. Das bedeutet, dass verschiedene Anwendungen parallel auf dieselbe Datenbasis zugreifen. Da dabei keine Erweiterung der vorhandenen Applikationslogik erforderlich ist, stellt dies eine vergleichsweise einfache und schnelle Lösung dar. Gleichzeitig liegen die Nachteile aber in der Gefahr von Dateninkonsistenzen und der Schaffung von potentiell redundanter Applikationslogik.

Integration über Funktionsaufrufe

Bei der Funktionsintegration werden Teile der Geschäftslogik eines Anwendungssystems aus einem anderen System über definierte Schnittstellen aufgerufen. Die vorhandene Applikationslogik wird dabei wiederverwendet und sorgt für die Konsistenz der zu verwaltenden Daten. Damit werden die Hauptnachteile der Integration über Datenhaltung überwunden. Allerdings geht dies häufig mit größeren Anpassungen der vorhandenen Applikationslogik einher, was die Umsetzung dieses Modells generell zu einem sehr komplexen Vorhaben werden lässt.

Sowohl das Modell der Integration über Datenhaltung als auch die Integration über Funktionsaufrufe sind als Basis für die weiteren Ausführungen relevant. Obwohl integrierte Systeme auf Grundlage der

letztgenannten Methode heute 'state of the Art' sind, finden sich immer wieder Lösungen unter Verwendung gemeinsamer Datenhaltung. Oft ist die Alternative zu aufwendig, oder mangels Möglichkeit des Zugriffs über API's oder den Quellcode auf das vorhandene Anwendungssystem, ausgeschlossen.

2.1.2 Kommunikationsstile

Kommunikation im Sinne der Übertragung von Daten zwischen Informationssystemen oder Systemkomponenten ist ein vielschichtiges Thema. Angefangen bei der Codierung der Nachrichten, über den Aufbau der Datenstrukturen, bis hin zu der verwendeten Kommunikationsinfrastruktur. Gerade im Rahmen der Funktionsintegration ist die Frage nach dem genauen Verlauf der Kommunikation zwischen den beteiligten Partnern von Interesse. Hierbei muss grundsätzlich zwischen *synchroner* und *asynchroner Kommunikation* unterschieden werden[43].

Synchrone Kommunikation

Diese Art des Verständigung findet Anwendung bei Funktionsaufrufen in allen gängigen Programmiersprachen. Charakteristisch für dieses Verfahren ist, dass der aufrufende Prozess solange mit der Weiterarbeit abwartet, bis er eine Antwort in Form eines Ergebnisses (Request/Reply-Stil) oder einer Quittung erhalten hat (Synchrone Einwegkommunikation[44]). Die Verwendung von synchroner Kommunikation macht also nur Sinn, wenn auch mit einer umgehenden Antwort zu rechnen ist.

Asynchrone Kommunikation

In diesem Falle arbeitet der Prozess unmittelbar nach dem Versand der Nachricht weiter. Er wartet also keine Betätigung oder ein Ergebnis ab. Man spricht daher bei diesem Vorgehen auch von 'fire and forget'. Es wird vom Sender dabei also nicht sichergestellt, dass die Nachricht ihr Ziel erreicht und auch verarbeitet wird. Dies muss im Rahmen der Anwendungsarchitektur ggf. an anderer Stelle sichergestellt werden. Zu diesem Zweck

43 Vgl. [Starke2002], S.181
44 S. [Keller2002], S.33

kommen so genannte Broker zum Einsatz, die sich um den weiteren Versand der Nachrichten kümmern. Ein Vorteil einer solchen Lösung ist, das Sender und Empfänger nicht zwingend zur gleichen Zeit verfügbar sein müssen. Außerdem können Nachrichten unter Berücksichtigung unterschiedlicher Prioritäten vom Broker weiter versendet werden. Ein bekanntes Beispiel für ein solches Vorgehen sind E-Mail-Systeme.

2.1.3 Kommunikations-Infrastrukturen

Neben der Betrachtung des Kommunikationsverlaufs ist vor allem auch der zur Integration verwendete 'Unterbau' interessant. Auch hierzu gibt es verschiedene Ansätze:

Bei der Verwendung von *Punkt-zu-Punkt Verbindungen* besitzt jedes System direkte Schnittstellen zu jedem weiteren mit ihm integrierten System[45]. Die Kommunikation erfolgt unmittelbar zwischen den Systemen und damit nicht über eine zentrale Instanz. Diese, den Quell- und Zielsystemen gut angepasste Integrationslösung, hat den Nachteil, dass mit steigender Anzahl der zu integrierenden Systeme der Aufwand für die Menge der zu wartenden Schnittstellen enorm steigt[46].

Erfolgt die Kommunikation über einen *Softwarebus*, bedeutet dies zwar, dass diese Aufgabe immer von einer bestimmten Komponente der Systemarchitektur übernommen wird, allerdings hat diese keinen zentralen Charakter. Vielmehr muss sie auf jedem Rechnersystem auf dem eine Anwendung diesen Kommunikationsdienst nutzen möchte installiert sein[47].

Bei der Integration über eine zentrale Integrationsplattform besitzen sämtliche zu integrierende Systeme ausschließlich Schnittstellen zu diesem zentralen System, welches die gesamte Kommunikation abwickelt. Man spricht in diesem Fall auch von einer *Hub-and-Spokes-Architektur* (Narbe-und-Speiche). Sämtliche für die Unterstützung der Prozesse notwendige Logik, sowie alle Möglichkeiten für Kontrolle und Verwaltung, finden sich in

45 S. [Keller2002], S.23
46 Vgl. [Kaib2002], S.69
47 Vgl. [GruhnThiel2000], S.164

diesem einen Teilsystem. Seine größte Stärke ist aber gleichzeitig auch die größte Schwäche dieses Ansatzes: Diese Schaltzentrale ist im Zweifel auch der Flaschenhals und die empfindlichste Stelle des ganzen Systems. In der Praxis kann dies bedeuten, dass dieser Hub ggf. redundant ausgelegt werden muss[48].

2.1.4 Spezielle Probleme verteilter Systeme

Neben den zuvor behandelten möglichen Ansatzpunkten der Integrationsarchitektur und der Kommunikation zwischen den einzelnen Systemen, treten einige Probleme schon allein dadurch zu tage, dass die Bestandteile der Anwendungen oftmals auf unterschiedlichen Rechnersystemen betrieben werden. Die folgenden Punkte gilt es dabei zu beachten[49]:

- Ein Funktionsaufruf über Systemgrenzen hinweg erfordert einen erheblichen Mehraufwand als lokale Aufrufe.
- Es werden ggf. Mechanismen zum Auffinden (Namensdienste) und zur Beschreibung von Schnittstellen benötigt, die von Außen aufgerufen werden können.
- Es muss sichergestellt werden, das miteinander kommunizierende Systeme eine einheitliche Codierung der Daten, also von Zeichen, und numerischen Werten verwenden. Stichworte im Zusammenhang mit der Verwendung von Zeichensätzen sind EBCDIC, ASCII und Unicode bzw. UTF-2, UTF-8, UTF-16 usw.
- Die Verfolgung und Behandlung von Fehlern über Systemgrenzen hinweg gestaltet sich deutlich schwieriger als bei der Wartung eines einzelnen lokalen Systems.

All diese Punkte sollten im Rahmen der Integrationsarchitektur gelöst oder zumindest berücksichtigt werden.

48 Vgl. [Kaib2002], S.86
49 Vgl. [Starke2002], S.176

2.1.5 Sicherheitsanforderungen

Natürlich spielen gerade für verteilte Systeme Sicherheitsaspekte eine gewichtige Rolle. Zur Sicherheit in Informationssystemen wurden 1992 von der OECD, der Organisation für wirtschaftliche Zusammenarbeit und Entwicklung, Richtlinien erstellt, mit deren Hilfe Systeme entsprechend ihres Sicherheitsniveaus beurteilt werden können. Demnach sind die folgenden Eigenschaften bedeutend für die Systemsicherheit[50]:

- **Vertraulichkeit:** Geheimhaltung von Daten gegenüber dritten. Im wesentlichen kommen dazu Verschlüsselungsverfahren zum Einsatz.
- **Identifikation und Authentizität:** Mechanismen zur Bestimmung, ob die grundsätzliche Benutzung eines Systems durch einen Benutzer oder ein anderes System rechtmäßiger erfolgt. Zur Identifikation werden in der Regel Benutzernamen verwendet. Zur Authentifizierung, also der Bestätigung der angegebenen Identität, kommen Geheimnisse wie beispielsweise Passwörter zum Einsatz.
- **Integrität:** Schutz vor ungewollter oder vorsätzlicher Verfälschung von übermittelten Daten. Zu diesem Zweck werden Prüfsummen gebildet oder kryptographische Verfahren verwendet.
- **Zugangangskontrolle:** Prüfung der Zugangsberechtigung zu einzelnen Bereichen des Systems. Dies setzt eine vorige Identifikation und eine entsprechende Rechteverwaltung vorraus.
- **Nachvollziehbarkeit:** Protokollierung aller durchgeführten Aktionen mit der Möglichkeit diese im Nachhinein wieder dem Initiator zuzuordnen.
- **Verfügbarkeit:** Garantie der Nutzbarkeit eines Systems durch Skalierung der Systemressourcen und der renundanten Auslegung von Teilsystemen.

50 Vgl.[Zwißler2002], S.100

2.1.6 Transaktionsmechanismen

Neben den zuvor beschriebenen Sicherheitsanforderungen müssen die zur Unterstützung von Geschäftsprozessen verwendeten Systeme auch die Durchführung der angestoßenen Datenverarbeitungsprozesse garantieren können. Für die dazu benötigten Transaktionsmechanismen sind die folgenden Eigenschaften erforderlich[51]: Unteilbarkeit (englisch Atomicity), Konsistenz (engl. Consistenzy), Isoliertheit (engl. Isolation) und Dauerhaftigkeit (engl. Durability). Man spricht dabei, aufgrund der Anfangsbuchstaben der englischen Begriffe, auch von den ACID-Kriterien[52]. Das bedeutet im Einzelnen:

- **Unteilbarkeit:** Eine Transaktion, die aus mehreren Einzelaktionen bestehen kann, wird entweder vollständig oder gar nicht ausgeführt. Sollte es aus irgendeinem Grund zu einem Abbruch kommen, befindet sich das System, bezogen auf diese Transaktion, wieder im selben Zustand wie vor Beginn der Ausführung.
- **Konsistenz:** Wird eine Transaktion erfolgreich ausgeführt, befindet sich das System danach wieder in einem konsistenten Zustand.
- **Isoliertheit:** Alle Aktionen einer Transaktion laufen getrennt von anderen Transaktionen ab. Die von ihr durchgeführten, zunächst vorläufigen Änderungen sind erst nach Bestätigung (*commit*) und erfolgreichem Abschluß der Transaktion gültig und nach Außen sichtbar. Vorher können sie vollständig zurückgenommen werden (engl. *rollback*).
- **Dauerhaftigkeit:** Wurde eine Transaktion vom System als erfolgreich bestätigt, bleiben in ihrem Rahmen durchgeführten Änderungen dauerhaft erhalten. Das bedeutet insbesondere, dass sie auch nach einem Neustart des gesamten oder Teilen des Systems noch vorhanden sind.

Um diese Kriterien erfüllen zu können muss in verteilten Systemen ein zweistufiges Verfahren verwendet werden. Zunächst müssen die not-

51 Vgl. [Wutka2001], S.248
52 S. [Kaib2002], S.111

wendigen Änderungen angekündigt und von allen betroffenen Teilsystemen genehmigt werden. Im zweiten Schritt erfolgt dann persistente Modifikation, also die Festschreibung der Änderungen. Diese Methode wird als *Two-Phase-Commit* bezeichnet.

2.2 Integrationsplattformen

Insbesondere für die zuvor beschriebene Integration über Funktionsaufrufe stehen eine Reihe von möglichen Plattformen mit verschiedenen Konzepten zur Verfügung. Kapitel 2.2.1 wird zunächst einen Überblick über gängige Softwareprodukte zur Integration geben. Im den darauf folgenden Abschnitten werden dann unterschiedliche technische Lösungen erörtert.

2.2.1 Softwareprodukte zur Integration

Natürlich kann dieses Kapitel nur eine Auswahl der allgemein zur Integration verwendeten Produkte beschreiben. Es ist aber davon auszugehen, dass es sich dabei um die gängigsten Systeme im Zusammenhang mit Integrationslösungen handelt.

2.2.1.1 ERP-Systeme

Wie bereits in Abschnitt 1.3 dargelegt, bedeutete die Einführung von betriebswirtschaftlicher Standardsoftware in Form von ERP-Systemen einen gewaltigen Schritt nach vorn hinsichtlich der Bemühungen um integrative Unterstützung von Geschäftsprozessen. Die Module dieser Systeme zur Erledigung betrieblicher Standardaufgaben, wie Rechnungswesen, Personalwirtschaft und Logistik, sind von Herstellerseite bereits miteinander integriert. Darüber hinaus werden in der Regel auf technischen Standards basierende Schnittstellen zur Funktionsintegration angeboten, die es möglich machen unternehmensspezifische Lösungen anzubinden und die Lücken in jenen Bereichen zu schließen, die von der ERP-Software nicht abgedeckt werden. Damit nimmt das ERP-System häufig die Rolle der unter

2.1.3 beschriebenen zentralen Integrationsplattform ein. Man spricht in diesem Falle auch von einem *ERP Backbone*[53].

2.2.1.2 Messagebroker

Wie bereits unter 2.1.2 erwähnt, spielen insbesondere im Zusammenhang mit asynchronen Kummunikationsmechanismen Messagebroker eine bedeutende Rolle. Gerade wenn es nicht möglich ist die Nachricht eines Senders direkt zu verarbeiten oder Sender und Empfänger nicht gleichzeitig verfügbar sind, kommen diese, auch unter dem Namen *Message oriented Middleware* (MoM) bekannten Systeme, zur Anwendung. Die empfangenen Nachrichten werden in eine Warteschlange (engl. queue) eingestellt in der sie bis zur Weitergabe an den oder die Empfänger verbleiben. Außerdem werden Funktionen zum Auffinden, Erzeugen und Löschen der Message-Queues zur Verfügung gestellt[54].

Verbreitete Produkte aus diesem Gebiet sind der *Microsoft Message Queue Server* und *IBM MQSeries*, welches vorzugsweise zur Integration von IBM Großrechnern in Unix- und Windows-Umgebungen verwendet wird.

Eine Ergänzung des MoM-Ansatzes stellt das sogenannte *Publish/Subscribe* Verfahren dar, welches eine noch losere Kopplung der Systeme möglich macht. Dabei meldet sich ein Sender und Empfänger für Nachrichten zu bestimmten Themen (Topics) beim Messagebroker an. Empfängt dieser dann eine Nachricht zu einem dieser Themen, wird diese an alle dafür registrierten Empfänger weitergeleitet[55].

2.2.1.3 Transaktionsmonitore

Wie bereits unter 2.1.6 dargelegt, sind bei verteilten Systemen zur Erfüllung der aufgeführten ACID-Kriterien, systemübergreifende Mechanismen notwendig. Hauptaufgabe von Transaktionsmonitoren, im englischen *Transaction Processing-Monitore* (TP-Monitore) genannte ist die Bereitstellung globaler Transaktionsmechanismen. Aber auch Ressourcen-

53 S. [Kaib2002], S.70
54 S. [Zwißler2002], S.177
55 Vgl. [Keller2002], S. 83 u. S. 86

management über die Vergabe von Prioritäten und der Schutz des Zugriffs auf die zu verwaltenden Daten gehört zu ihren Aufgaben[56]. Sie wurden in den 1970er Jahren für den Mainframebereich entwickelt[57] und werden mittlerweile vorzugsweise für die Integration von Großrechnern eingesetzt[58]. Führende Produkte aus diesem Bereich sind *CISC* und *Encina* von *IBM*, *Tuxedo* von *BEA Systems*, sowie *Component Transaction Server* von *Sybase*[59].

2.2.1.4 Datenbankzentrierte Produkte

Für das Modell der gemeinsamen Datenhaltung in Datenbanksystemen existieren verschiedene Produkte zur Unterstützung, die sich in Umfang und Komplexität unterscheiden. [Kaib2002][60] spricht in diesem Zusammenhang auch von *Datenzugriffsorientierter Middleware* und differenziert zwischen den folgenden drei Verfahren:

- **Proprietäre Schnittstellen** sind jene, die vom Hersteller eines Datenbanksystems für den Zugriff auf dieses Produkt zur Verfügung gestellt wird
- **Standardschnittstellen** stellen einheitliche Programmierschnittstellen, sogenannte APIs (Application Programming Interfaces), für den Zugriff auf verschiedene Datenbanksysteme zur Verfügung
- **Datenbank-Gateways** sind zwischengeschaltete Applikationen die es erlauben, auf eine breites Spektrum relationaler und nicht relationaler Datenbanken zuzugreifen

Populäre Vertreter für Standard APIs sind *Open Database Connectivity* (ODBC) und *Java Database Connectivity* (JDBC). *ODBC* wurde von der Firma Microsoft entwickelt um einheitliche Schnittstellen für Datenbanken unter Microsoft Windows zur Verfügung zu stellen. Mittlerweile existieren aber

56 S. [Lexitron2003], Eintrag: TP-Monitor, Stand: 29.11.2003
57 S. [Zwißler2002], S.177
58 Vgl. [Fischer1996], S.7
59 S. [Kaib2002], S.147
60 Vgl. [Kaib2002], S.107

auch Implementierungen von *ODBC* z.B. unter UNIX-Betriebssystemen[61]. Mit *JDBC* definiert *SUN Microsystems* eine entsprechende Schnittstellensammlung für die Programmiersprache *Java*[62].

2.2.1.5 Applikations-Server

Der Begriff des Applikations-Servers ist in der Literatur, nicht zwingend aber häufig, mit der Entwicklung von Webanwendungen verbunden[63]. Im Sinne einer Mehrschichtenarchitektur bildet er in diesem Falle eine dem Web Server nachgelagerte Schicht. Im Applikations-Server werden Geschäftsfunktionalitäten implementiert und nach außen zur Verfügung gestellt[64]. Er bietet dazu ein Laufzeitumgebung an, die für die Lösungen von Aufgaben wie Thread- und Prozessmanagement, Lastenverteilung, Transaktionsverwaltung, Verzeichnisdienste, Schnttstellen zur Backend-Anbindung etc. sorgt[65].

Zur Implementierung und Bereitstellung der Geschäftslogik werden häufig Komponententechnologien wie CORBA, EJB und COM/DCOM herangezogen. Die Kommunikation mit den vorgelagerten Applikationsschichten erfolgt entsprechend über Techniken für entfernte Funktions- bzw. Methodenaufrufe wie IIOP oder RMI[66]. Weil diese Technologien für die weiteren Ausführung von hohem Interesse sind, werden sie in den folgenden Abschnitten einer tieferen Betrachtung unterzogen.

2.2.2 Komponentenmodelle und RPC Mechanismen

Komponenten sind Software-Module, die einen erhöhten Grad der Wiederverwendbarkeit gewährleisten[67]. Eine Komponente ist nicht unbedingt ein einzelnes Objekt. Vielmehr stecken hinter einer Komponenten häufig eine Vielzahl von Objekten. Trotzdem bilden die objektorientierten Denkweisen

61 S. [Hage1996], Definition des ODBC Interfaces
62 Vgl. [Wutka2001], S.74
63 Vgl. [Zwißler2002], S.28
64 S. [Keller2002], S.22
65 S. [DenningerPeters2000], S.23
66 Vgl. [Kaib2002], S.143
67 S. [Wutka2001], S.123

und Techniken, aufgrund der engen Verwandtschaft, die ideale Basis für die Komponentenentwicklung. Die Schnittstelle ist, wie in der Objektorientierung, ein wesentliches Konzept des Komponentenparadigmas. Sie stellt einen durch die Komponente angebotenen Dienst nach außen zur Verfügung. Eine Komponente kann, wie ein Objekt, eine Vielzahl von Schnittstellen besitzen[68].

Für den Fall, dass sich eine Komponente nicht auf dem selben Rechnersystemen wie der Dienstnehmer befindet, werden Mechanismen benötigt, die es ermöglichen, Methodenaufrufe über ein Netzwerk zu versenden und das Ergebnis entgegen zu nehmen. Man spricht in diesem Fall von *Remote Procedure Calls* (RPC). RPC-Mechanismen umfassen auch spezielle Fehlerbehandlungen, die der Situation Rechnung tragen, dass Client und Server Fehler ggf. unabhängig voneinander behandelt werden müssen[69].

2.2.2.1 Stub und Skeleton

Alle im Folgenden besprochenen Techniken zur Erstellung verteilter Anwendungen basieren im Kern auf RPC-Mechanismen und verwenden die gleichen Konzepte, um von der eigentlichen Datenübertragung und der Nutzung eines entfernten Dienstes zu abstrahieren. Dazu werden zwei Teile benötigt: Der sogenannte Rumpf (engl. Stub) wird auf Seiten des Dienstnehmers wie eine lokale Funktion aufgerufen. In ihm erfolgt das als *marshaling* bezeichnete umwandeln (serialisieren) der Aufrufparameter zur Versendung übers Netz. Auf der gegenüberliegen Seite wird die Anfrage vom seinem Gegenstück, dem Skelett (engl. Skeleton), entgegengenommen, deserialisiert und an die tatsächliche Implementierung der aufgerufenen Methode weitergereicht. Beim Versenden der Antwort wird entsprechend verfahren. In der Regel existieren im Rahmen von RPC-Mechanismen Beschreibungen der Serverdienste, mit deren Hilfe Stub und Skeleton generiert werden könnten.

---

68 Vgl. [DenningerPeters2000], S.15
69 Vgl. [Starke2002], S.183

RPC Implementierungen basieren in der Regel auf synchronen Kommunikationsmechanismen. Demzufolge wartet der Stub mit der Weiterarbeit bis er eine Antwort vom Server erhalten hat. Bei asynchronen Varianten müssen hintereinander zwei Funktionen des Stubs aufgerufen werden. Die erste setzt die Anfrage an den Dienst ab und die zweite liefert das Ergebnis[70].

2.2.2.2 Distributed Computing Enviroment (DCE)

Der von der *Open Software Foundation* (OSF), der heutigen *Open Group*, verabschiedete *Distributed Computing Enviroment* (DCE)-Standard basiert ebenfalls auf RPC-Mechanismen und zielt auf die Erstellung verteilter Anwendungen ab. In diesem Standard ist die Übertragung von Zeichen- und numerischen Daten geregelt. Außerdem definiert er Sicherheitsmechanismen, Zeit- und Verzeichnisdienste und ein verteiltes Dateisystem[71]. Im UNIX-Umfeld stellt dieser Standard eine Schlüsseltechnologie dar[72]. Seit Ende der 1990er Jahre wurde DCE im kommerziellen Bereich aber zunehmend von Microsofts COM/DCOM, CORBA und der Java-2-Architektur verdrängt.

2.2.2.3 Component Object Model (COM/DCOM/COM+)

Das *Component Object Model* (COM) der Firma Microsoft definiert ein Komponentenmodell mit dem der Zugriff auf das Dienstangebot einer Softwarekomponente standardisiert wurde. Dabei bleibt es gleichgültig in welcher Programmiersprache die COM-Komponente implementiert ist, solange sie ein bestimmtes binäres Format aufweist[73].

Die Entwicklung von COM geht letztlich auf die von Microsoft *Object Linking und Embedding* (OLE) bezeichnete Technologie zurück, die es ermöglicht, Objekte aus einer Anwendung in einer anderen zu verwenden. Ein populäres Beispiel dafür ist die Verwendung einer mit Excel erstellten Tabelle in einem

70 Vgl. [Zwißler2002], S.179
71 Vgl. [Zwißler2002], S.180
72 S. [Kaib2002], S.106
73 Vgl. [GruhnThiel2000], S.51

Word Dokument. Demnach ist die Entstehung von COM eng mit der Geschichte von Microsoft Windows und Microsoft Office verbunden[74].

Zunächst waren COM-Komponenten ausschließlich darauf ausgelegt, miteinander zu interagieren, wenn sie auf dem selben Rechnersystem installiert waren. Mit Windows NT 4.0 wurde 1996 der *Distributed COM* (DCOM) Standard eingeführt, mit dem der Zugriff auf Softwarekomponenten anderer Rechner möglich wurde. Dieser Standard erlebte seine Weiterentwicklung mit *COM+*, welches integraler Bestandteil von Windows 2000 wurde. Im Rahmen von COM+ sollte der mitunter recht komplizierte Umgang mit COM-Komponenten erleichtert werden[75] und es wurden zusätzliche Dienste ergänzt. Es ist aber nach wie vor strittig, ob COM+ als neuer Oberbegriff für diese Technologie anzusehen ist, oder nur für die neu eingeführten Dienste steht[76].

Für den Zugriff auf entfernte Komponenten verwendet DCOM eine objektorientierte Erweiterung des DCE-RPC, indem es mehrere Methoden zu abstrakten Schnittstellen zusammenfasst. Diese so genannten Interfaces werden mithilfe der Microsoft *Interface Definition Language* (IDL) beschrieben. Über die Registry von Microsoft Windows können COM-Komponenten verwaltet und aufgefunden werden. Entfernte Objekte sind darin mit den notwendigen Adressangeben verzeichnet[77].

2.2.2.4 CORBA

Die *Common Object Request Broker Architecture* (CORBA) ist eine von der *Object Management Group* (OMG) definierte Basisarchitektur für die Entwicklung verteilter objektorientierter Anwendungen[78]. Bei der Entstehung von CORBA war es das Ziel, eine Architektur zu schaffen, die es ermöglicht, Objekte, die auf beliebigen Rechnerplattformen in beliebigen Programmiersprachen implementiert sind, miteinander interagieren lassen zu können[79].

74 Vgl. [GruhnThiel2000], S.47
75 Vgl. [GruhnThiel2000], S.49
76 S. [ComKomponentenDe2003], COM/DCOM/COM+
77 Vgl. [Zwißler2002], S.182
78 Vgl. [GruhnThiel2000], S.164
79 Vgl. [Wutka2001], S.418

Das Kernelement der CORBA-Architektur stellt der *Object Request Broker* (ORB) dar, bei dem sich jene Objekte anmelden, die ihre Dienste nach außen zur Verfügung stellen möchten. Die Dienstanbieter - in der CORBA Nomenklatur auch Server genannt - werden über die CORBA *Interface Definition Language* (IDL) für den Dienstnehmer, den Client, beschrieben. Damit erfüllt die CORBA IDL den selben Zweck wie die Microsoft IDL für COM-Objekte. Es handelt sich dabei aber um zwei unterschiedliche Sprachen[80].

Der ORB ist als Softwarebus konzipiert, der die gesamte Kommunikation zwischen Client und Server regelt (siehe dazu auch Kapitel 2.1.3). Hierbei kommt in der Regel das *Internet Inter Orb Protocol* (IIOP) zum Einsatz[81]. Zur Abstraktion der entfernten Methodenaufrufe werden die unter 2.2.2.1 beschriebene *Stub und Skeleton* Mechanismen verwendet, die anhand der IDL mit entsprechenden Hilfsmitteln für die als Client und Server verwendenden Programmiersprachen erzeugt werden können. Darüber hinaus bietet CORBA die Möglichkeit, ein dynamisches Verfahren anzuwenden, bei dem auch Objekte miteinander interagieren können, die zum Zeitpunkt der Compilierung der Applikation noch unbekannt waren[82].

2.2.2.5 Java RMI

Java *Remote Method Invocation* (RMI) ermöglicht die Erstellung von verteilten, Java-basierten Applikationen, in denen Methoden entfernter Objekte aus anderen Java *Virtual Machines* (VM) heraus aufgerufen werden können. RMI gehört zum Umfang aller Java Distributionen, von der *Java 2 Micro Edition*[83] bis hin zu *Java 2 Platform*, *Enterprise Edition*, in der es einen wesentlichen Teil der Enterprise JavaBeans Technologie darstellt (siehe dazu Kapitel 2.2.4 ab Seite 52).

RMI bedient sich dabei dem *Remote-Procedure-Call*-Paradigma und verwendet ebenfalls *Stub* und *Skeleton*, um von der Implementierung des

80 Vgl. [Zwißler2002], S.183
81 S. [Zwißler2002], S.183
82 S. [GruhnThiel2000], S.164
83 Vgl. [JavaSun2003], Java Remote Method Invocation, /products/jdk/rmi/, Stand 05.12.2003

Netzwerk nahen Codes zu abstrahieren[84]. Das Auffinden von entfernten Objekten erfolgt über die RMI-Registry, einem einfachen Nameserver, der Refferenzen auf verteilte Java RMI Objekte verwaltet[85].

Von 'Haus aus' verwendet Java RMI das *Java Remote Method Protocol* (JRMP) für die Kommunikation bei entfernten Methodenaufrufen[86]. Um Javaprogrammierern die einfache Verwendung von CORBA ohne Kenntnis der CORBA IDL zu ermöglichen, wurde mit Veröffentlichung der *Java 2 Standard Edition* Version 1.3 ein Mechanismus eingeführt, CORBA IDL Beschreibungen aus Java Klassen zu generieren und den Aufruf entfernter Methoden über das IIOP abzuwickeln[87].

Bei der Verwendung von RMI in 'reinen' Java Anwendungen, ist keine zusätzliche Schnittstellendefinition, wie mit IDL bei CORBA oder COM, notwendig. Diese ist bereits Teil jeder Java-Klasse und kann mithilfe der *Reflection API*, die ebenfalls zum Java Sprachumfang gehört, von anderen Java-Klassen ermittelt werden[88].

2.2.3 Web Services

Alle bisher in diesem Kapitel beschriebenen Techniken lösen das Problem der einfachen Integration von Komponenten über die Grenzen von Hardwareplattformen und Programmiersprachen hinweg möglicherweise nicht vollständig. Java RMI ist natürlicherweise nicht für alle Sprachen verfügbar, die Verwendung von DCOM bedeutet eine enge Verflechtung mit Strategie und Produkten der Firma Microsoft in Kauf nehmen zu müssen und CORBA ist bekannt für seine nicht gerade triviale Umsetzung. Außerdem gestaltet sich die Kommunikation zwischen verschiedenen gewachsenen CORBA Strukturen oft schwierig, da die ORB Implementierungen verschiedener Hersteller untereinander häufig nicht kompatibel sind[89].

---

84 S. [WilhelmsKopp1999], S.613
85 S. [ZDNetJavaRMI], Verteilte Services mit dem Java RMI-Framework
86 S. [JavaSun2003], RMI and IIOP in Java – FAQ, /pr/1997/june/statement970626-01.html, Stand 05.12.2003
87 S. [JavaSun2003], Java RMI over IIOP, /products/rmi-iiop/, Stand 05.12.2003
88 Vgl. [Zwißler2002], S.184
89 Vgl. [Knuth2002], S.13

In jüngster Zeit machen daher einige neue Technologien von sich reden, die unter dem Oberbegriff *Web Services* zusammen gefasst werden. Diesen Begriff definiert Keller wie folgt:

„Ein Web Service ist jene Anwendung oder Anwendugskomponente, die über Standard-Web-Protokolle aufgerufen werden kann."[90]

Als Standard-Web-Protokoll ist wohl zunächst das *Hypertext Transfer Protocoll* (HTTP) anzusehen. Aber auch das *File Transfer Protocol* (FTP) und das gängige E-Mail Protokoll SMTP gehören dazu.

2.2.3.1 XML als Basis für RPC-Mechanismen

Wie bereits in Kapitel 2.1.4 dargelegt, ist ein wesentlicher Aspekt bei der Entwicklung verteilter Systeme die Struktur der ausgetauschten Daten, da hierbei eine einheitliche Codierung der Informationen notwendig ist. Die oben vorgestellten Techniken verwenden zu diesem Zweck jeweils eigene Formate. In der Vergangenheit kam es bei der Entwicklung von Kommunikationsprotokollen und Speicherformaten vor allem auf einen effektiven Umgang mit vorhandenen Netzwerkbandreiten und Speicherplatz an. Daher wurden in der Regel maschienennahe Codierungen gewählt, die eine sehr kompakte Speicherung und Übertragung der Daten zulassen. In jüngster Zeit ist aber ein klarer Trend weg von maschienennahen Formaten und hin zu mehr Strukturierung, Flexibilität und Interoperabilität zu erkennen[91]. Die Auszeichnungssprache XML - ebenfalls ein W3C Standard - erfreut sich in diesem Zusammenhang ständig wachsender Beliebtheit. Sie gilt mittlerweile als das Speicherformat der Zukunft. Ein Beleg dafür dürfte sein, das Microsoft sogar das bis dahin geheime Dateiformat seiner Office Produkte auf XML umgestellt hat[92] und bereit ist, dieses offen zu legen[93].

XML verwendet für die Strukturierung von Daten, wie das auf ihr aufsetzende HTML, so genannte *Tags* und verwendet einfache Textdateien zur Speicherung. Dies hat den Vorteil, dass diese auch ohne die zur

90 [Keller2002], S.148
91 Vgl. [Zwißler2002], S.147
92 S. [Münz2001], Einführung in XML
93 S. [HeiseNews20031118hps]

Erstellung verwendete Software geöffnet und interpretiert werden können. Allerdings benötigt dieses Format aufgrund der für die Auszeichnung verwendeten *Tags* tendenziell mehr Speicherplatz, als maschienennahe Formate[94].

Wenn man sich in der Literatur zu Thema Web Services umsieht, fällt die Verbindung mit XML unmittelbar ins Auge. Alle Standards aus diesem Bereich, die sich mittlerweile durchgesetzt haben, verwenden XML als Basis[95]. Im Folgenden sollen populäre Web Service-Technologien einer kurzen Betrachtung unterzogen werden. Da die Darstellung von Grundlagen zu XML und den gängigen Internetprotokollen nicht Teil dieser Arbeit ist, muss an dieser Stelle auf entsprechende weiterführende Literatur verwiesen werden.

2.2.3.2 XML-RPC

Laut der offiziellen Webseite ist XML-RPC eine Spezifikation und ein Satz von Implementierung, die es Softwareprodukten auf verschiedenen Betriebssystemen erlauben, Prozeduraufrufe über das Internet durchzuführen. Dazu wird das HTTP als Transportprotokoll und XML zur Codierung der Aufrufe verwendet. Beim Entwurf von XML-RPC war es das Ziel, einen möglichst einfachen Mechanismus zur Übermittlung und Verarbeitung komplexer Datenstrukturen zu schaffen[96].

Ein Aufruf über XML-RPC erfolgt über ein 'wohlgeformtes' XML-Dokument, welches den Methodennamen und die Aufrufparameter enthält. Dieser Methodenaufruf wird an den XML-RPC Dienst per standard HTTP-POST Anfrage übertragen[97]. Die Spezifikation definiert darüber hinaus den Aufbau des Antwortdokuments, sowohl für einen erfolgreichen Methodenaufruf, als auch für den Fehlerfall. Sie definiert eine Reihe von Datentypen für ganze Zahlen, Wahrheitswerte, Zeichenketten, Fließkommazahlen etc., sowie eine mögliche Verschachtelung in strukturierten Datentypen und Listen[98].

94 S. [Lexitron2003], Eintrag: XML, Stand: 07.12.2003
95 Vgl. [Knuth2002], S.14
96 S. [XML-RPC.com], Startseite, Stand: 07.12.2003
97 Vgl. [Vaswani2002], S.163
98 Vgl. [XML-RPC.com], /spec, Stand: 07.12.2003

Aktuell existieren für XML-RPC eine Reihe von Implementierung für diverse Programmiersprachen wie ASP, C++, Delphi, Eiffel, Java, JavaScript, Lisp, Perl, PHP und Python, um nur einige zu nennen[99].

2.2.3.3 SOAP

Das *Simple Object Access Protocol* (SOAP) basiert auf einem ähnlichen Konzept wie XML-RPC, ist allerdings von seiner Zielsetzung breiter angelegt. Es handelt sich dabei um eine Empfehlung des *World Wide Web Consortium* (W3C) zum Zweck des dezentralen Informationsaustauschs über ein Netzwerk, die von Großunternehmen wie *Microsoft*, *IBM*, *HP*, *Lotus* und *SAP* unterstützt wird. Speziell entwickelt für die Informationsübermittlung zwischen verteilten Anwendungen und Objekten auf verschiedenen Plattformen, basiert es ebenfalls auf XML. Für den Transport der Nachrichten kommt in der Regel HTTP zum Einsatz. Aber auch zur Verwendung anderer Protokolle wie dem E-Mail Protokoll SMTP nimmt die Spezifikation Stellung[100].

Im Rahmen der SOAP-Spezifikation ist lediglich der Aufbau einer SOAP-Nachricht beschrieben. Außerdem definiert sie Regeln zur Verarbeitung, Übermittlung und Weiterleitung. Die Verwendung von SOAP ist daher nicht auf den RPC Bereich beschränkt. SOAP kann dazu verwendet werden, Daten jeglicher Art zu transportieren. Auch Szenarien bei denen Nachrichten über mehrere Knoten weitergeleitet werden sind vorgesehen. *Remote Procedure Calls* sind also nur eine mögliche Anwendung.

Ein SOAP-Nachricht ist ein wohlgeformtes XML-Dokument und besteht aus den folgenden drei elementaren Teilen:

- Der **Envelope** (engl. für Umschlag) beinhaltet Informationen über die Codierung der SOAP-Nachricht. Gleichzeitig stellt er den Wurzelknoten des XML-Dokuments dar.
- Der **Header** (engl. für Dateikopf) ist ein optionaler Teil in dem Steuer- bzw. Metainformationen zu der übermittelten SOAP-Nachricht

99 S. [XML-RPC.com], /directory/1568/implementations, Stand: 07.12.2003
100 S. [W3C_SOAP1.2_Part0]

transportiert werden können, wie z.B. Informationen zur Authentifizierung, Transaktionssteuerung etc.

- Der **Body** (engl. für Rumpf) umfasst die eigentlich zu übermittelnden Daten[101].

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
 <env:Header>
  <m:reservation xmlns:m="http://travel.example.org/reserv"
         env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
         env:mustUnderstand="true">
   <m:reference>uuid:093a2da1-q345-739r-ba5d</m:reference>
   <m:dateAndTime>2001-11-29T13:20:00.000-05:00</m:dateAndTime>
  </m:reservation>
  <n:passenger xmlns:n="http://mycompany.example.com/employees"
         env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
         env:mustUnderstand="true">
   <n:name>Åke Jógvan Øyvind</n:name>
  </n:passenger>
 </env:Header>
 <env:Body>
  <p:itinerary xmlns:p="http://travel.example.org/reserv/travel">
   <p:departure>
     <p:departing>New York</p:departing>
     <p:arriving>Los Angeles</p:arriving>
     <p:departureDate>2001-12-14</p:departureDate>
     <p:departureTime>late afternoon</p:departureTime>
     <p:seatPreference>aisle</p:seatPreference>
   </p:departure>
   <p:return>
     <p:departing>Los Angeles</p:departing>
     <p:arriving>New York</p:arriving>
     <p:departureDate>2001-12-20</p:departureDate>
     <p:departureTime>mid-morning</p:departureTime>
     <p:seatPreference/>
   </p:return>
  </p:itinerary>
  <q:lodging xmlns:q="http://travel.example.org/reserv/hotels">
   <q:preference>none</q:preference>
  </q:lodging>
 </env:Body>
</env:Envelope>
```

Abbildung 2: Beispiel einer typischen SOAP-Nachricht mit Envelope, Header und Body

Abbildung 2 zeigt eine mögliche SOAP-Nachricht zu einer Reisebuchung[102]. Das Beispiel zeigt alle drei beschriebenen elementaren Teile einer SOAP-Nachricht: Envelope, Header und Body. Der Inhalt von Header und Body ist

101 Vgl. [Vaswani2002], S.165
102 Aus [W3C_SOAP1.2_Part0], 2.1 SOAP Messages

applikationsspezifisch und nicht Teil der SOAP-Spezifikation. Bei der Verwendung von SOAP liegen die Entscheidungen den Aufbau dieser beiden Teile betreffend im vollen Umfang in der Hand des Entwicklers. Es werden in der Spezifikation auch keine Datentypen für die Übermittlung von Funktionsparametern wie bei XML-RPC festgelegt. Mit der Frage, wie der Aufbau einer SOAP-Nachricht beschrieben werden kann, wird sich der folgenden Abschnitt 2.2.3.4 beschäftigen.

Allerdings definiert die Spezifikation den Aufbau von Fehlernachrichten die ausgetauscht werden, wenn ein durch eine korrekte Nachricht angestoßener Verarbeitungsprozess einen Fehler ergeben hat, oder eine fehlerhafte Nachricht versendet wurde. Zum Abschluss dieses Abschnitts zeigt Abbildung 3 das Beispiel einer Antwortnachricht, die auf einen Fehler bei der Verarbeitung des Headers der Ursprungsnachricht hinweist[103]:

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope>
 <env:Header>
    <env:NotUnderstood qname="t:transaction"
                xmlns:t="http://thirdparty.example.org/transaction"/>
 </env:Header>
 <env:Body>
  <env:Fault>
   <env:Code>
    <env:Value>env:MustUnderstand</env:Value>
   </env:Code>
   <env:Reason>
      <env:Text xml:lang="en-US">Header not understood</env:Text>
      <env:Text xml:lang="fr">En-tête non compris</env:Text>
   </env:Reason>
  </env:Fault>
 </env:Body>
</env:Envelope>
```

Abbildung 3: Beispiel SOAP-Nachricht, die einen Fehler bei der Verarbeitung der eingegangenen Nachricht meldet

2.2.3.4 WSDL und UDDI

Wie in Kapitel 2.1.4 bereits erwähnt, werden im Zusammenhang mit verteilten Systemen und dem Aufruf entfernter Schnittstellen auch

103 Aus [W3C_SOAP1.2_Part0], 2.3 Fault Scenarios

Mechanismen zur Beschreibung und zum Auffinden dieser Schnittstellen benötigt. Diese Aufgaben übernehmen im Zusammenhang mit Web Services die *Web Services Description Language* (WSDL) und der Verzeichnisdienst *Universal Description, Discovery & Integration* (UDDI).

Web Services Definition Language

WSDL verwendet einen speziellen XML-Dialekt zur Beschreibung von Web Services. Dazu existieren in einer WSDL-Definition zunächst fünf verchiedene Konstrukte, deren Inhalte teilweise aufeinander Bezug nehmen: *message*, *types*, *portType*, *binding* und *service*.

Über ein *message* Konstrukt wird eine eingehende oder ausgehende Nachricht definiert. Diese kann wiederum mehrere *part* Elemente enthalten, die für die einzelnen Datenelemente stehen und für die demzufolge auch Typeninformationen spezifiziert werden müssen. Dafür können entweder elementare XML Schema Typen verwendet werden, oder es können im Rahmen des *types* Konstrukts komplexere Typen unter Verwendung von XML Schema definiert werden.

Ein *portType* fasst mehrere Operationen unter einem Namen zusammen. Eine Operation bezieht sich wiederum auf die Nachrichten Definition und legt fest, welche der beschriebenen Nachrichten für sie eine Eingangs- oder Ausgangsnachricht darstellt. Wobei es Operationen gibt, die auch nur aus einer Eigangs- oder Ausgangsnachricht bestehen. RPC Operationen benötigen typischerweise beides.

Über das *binding* Konstrukt wird das zu verwendete Protokoll und die Codierung festgelegt. Erst an dieser Stelle wird zum Beispiel die Verwendung von HTTP und SOAP definiert. Zu guter Letzt werden über die *Service* Definition mehrere sogenannte Ports zu einem Dienst zusammengefasst, wobei jeder Port auf eine *bindig* Definition verweist. Erst in diesem Schritt wird der Port und damit der Dienst einer URL zugeordnet[104].

104 Vgl. [Zwißler2002], S.193

```
<?xml version="1.0"?>
<definitions name="StockQuote"
    targetNamespace="http://example.com/stockquote.wsdl"
          xmlns:tns="http://example.com/stockquote.wsdl"
          xmlns:xsd1="http://example.com/stockquote.xsd"
          xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
          xmlns="http://schemas.xmlsoap.org/wsdl/">
 <types>
  <schema targetNamespace="http://example.com/stockquote.xsd"
              xmlns="http://www.w3.org/2000/10/XMLSchema">
   <element name="TradePriceRequest">
    <complexType>
     <all>
      <element name="tickerSymbol" type="string"/>
     </all>
    </complexType>
   </element>
   <element name="TradePrice">
    <complexType>
     <all>
      <element name="price" type="float"/>
     </all>
    </complexType>
   </element>
  </schema>
 </types>
 <message name="GetLastTradePriceInput">
  <part name="body" element="xsd1:TradePriceRequest"/>
 </message>
 <message name="GetLastTradePriceOutput">
  <part name="body" element="xsd1:TradePrice"/>
 </message>
 <portType name="StockQuotePortType">
  <operation name="GetLastTradePrice">
   <input message="tns:GetLastTradePriceInput"/>
   <output message="tns:GetLastTradePriceOutput"/>
  </operation>
 </portType>
 <binding name="StockQuoteSoapBinding" type="tns:StockQuotePortType">
  <soap:binding style="document"
            transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="GetLastTradePrice">
   <soap:operation
           soapAction="http://example.com/GetLastTradePrice"/>
    <input><soap:body use="literal"/></input>
    <output><soap:body use="literal"/></output>
  </operation>
 </binding>
 <service name="StockQuoteService">
  <documentation>My first service</documentation>
  <port name="StockQuotePort" binding="tns:StockQuoteBinding">
   <soap:address location="http://example.com/stockquote"/>
  </port>
 </service>
</definitions>
```

Abbildung 4: Beispiel einer WSDL Definition

Abschließend zeigt Abbildung 4 das Beispiel einer WSDL Definition mit den oben beschriebenen Konstrukten[105]. Mit den entsprechenden Werkzeugen kann anhand der WSDL Beschreibung entsprechender Programmcode generiert werden, über den der Dienst angesprochen werden kann. Eine dynamische Auswertung und Nutzung zur Laufzeit ist ebenfalls möglich.

Universal Description, Discovery & Integration

UDDI definiert nicht nur eine Verzeichnisdienst zum Auffinden eines Web Services, es geht noch weit darüber hinaus. Es soll Unternehmen dabei unterstützen, potentielle Geschäftspartner zu finden und die elektronische Integration der Geschäftsprozesse über Webdienste möglich zu machen. Der Anstoß für die UDDI Architektur erfolgte aus gemeinsamen eBusiness Projekten von *Ariba*, *IBM* und *Microsoft* und es wurde großer Wert darauf gelegt, auf bestehende Standards wie DNS, XML und SOAP aufzusetzen[106].

UDDI definiert für die Beschreibung der so genannten „Business Entity" und deren Web Services[107] ein XML Datenmodell, bestehend aus sogenannten *Entities*. Im einzelnen werden die folgenden Entity-Typen definiert: Die *businessEntity* beschreibt ein Untenehmen oder einer anderen Organisation, die Web Services anbietet. Der *businessService* Typ charakterisiert eine Sammlung von verwandten Webdiensten, die von einer Organisation angeboten werden, die wiederum durch eine *businessEntity* beschrieben ist. Ein *bindingTemplate* liefert technische Informationen wie z.B. die URL oder eine sonstige Beschreibung, die notwendig ist, um mit dem Web Service zu interagieren und das sogenannte *tModel* beschreibt das 'technische Modell', also die Beschreibung der Standards auf denen der Web Service basiert, wie z.B. eine WSDL Beschreibung inklusive definition der verwendeten Protokolle etc[108].

Darüber hinaus definiert die Spezifikation mehrere APIs die Verhalten und Kommunikation mit und zwischen verschiedenen UDDI Implementationen standardisieren. Im Wesentlichen geht es dabei um den Zugriff und die

105 Aus [W3C_WSDL1.1], 1.2 WSDL Document Example
106 Vgl. [Zwißler2002], S.192
107 S. [Knuth2002], S.95
108 Vgl. [OASI_UDDI_3.0.1], 1. Introduction

Bereitstellung bzw. Pflege der UDDI Entities durch die Dienstnehmer bzw. Dienstanbieter. Aber auch für Aspekte wie Sicherheit und Replikation sind APIs spezifiziert.

2.2.4 Java 2 Platform, Enterprise Edition (J2EE)

Die *Java 2 Platform* ist eine reine Softwareplattform, die vollständig von den Eigenheiten der darunter liegenden Hardware abstrahiert. Im Kern der *Java 2 Platform* steht die Java VM (Virtual Machine) deren Portierung auf die verschiedenen Hardwareplattformen die Ausführung von Java Applikationen ohne weitere Anpassungen möglich macht. Die *Java 2* wird von *Sun Microsystems* in drei Varianten angeboten[109]:

- Die **Java 2 Platform, Standard Edition** (J2SE) bietet eine Laufzeitumgebung für Java Applikationen. Sie besteht aus dem Compiler, Werkzeugen und den Standard Java APIs zum entwickeln, testen und nutzen von Applets und Applikationen.
- Die **Java 2 Platform, Enterprise Edition** (J2EE) definiert einen Standard für die Entwicklung mehrschichtiger Unternehmensapplikationen. Sie setzt dabei auf J2SE auf und stellt zusätzliche Dienste, Werkzeuge und APIs bereit, um die Entwicklung von 'Enterprise' Applikationen zu unterstützen.
- Die **Java 2 Platform, Micro Edition** (J2ME) besteht aus einer Reihe von Technologien und Spezifikationen, die auf Consumer- und Embedded-Devices, wie Mobiltelephone, PDAs, Drucker, TV Set-Top-Boxen etc., abzielen.

Schon die Standardversion J2SE unterstützt Java RMI und ein Komponentenmodell für lokale Komponenten mit dem Namen JavaBeans[110]. J2EE definiert eine Applikationsmodell welches sich an dem klassischen drei Schichten Modell orientiert (siehe dazu Kapitel 2.1.1). Wie Abbildung 5 zeigt, gliedert es dabei die im J2EE Server beheimatete Applikationsschicht

109 Vgl. [JavaSun2003], Java Technology Overview, /overview.html, Stand: 13.12.2003
110 Vgl. [Zwißler2002], S.184

in zwei Bereiche: *Web Tier* und *Business Tier*[111]. Außerdem unterscheidet es drei elementare Elemente: Komponenten, Container und Konnektoren. Die Implementierung der Komponenten liegt im Fokus der Anwendungsentwicklung, während fertige Container und Konnektoren von der Komplexität systemnaher Programmierung abstrahieren sollen.

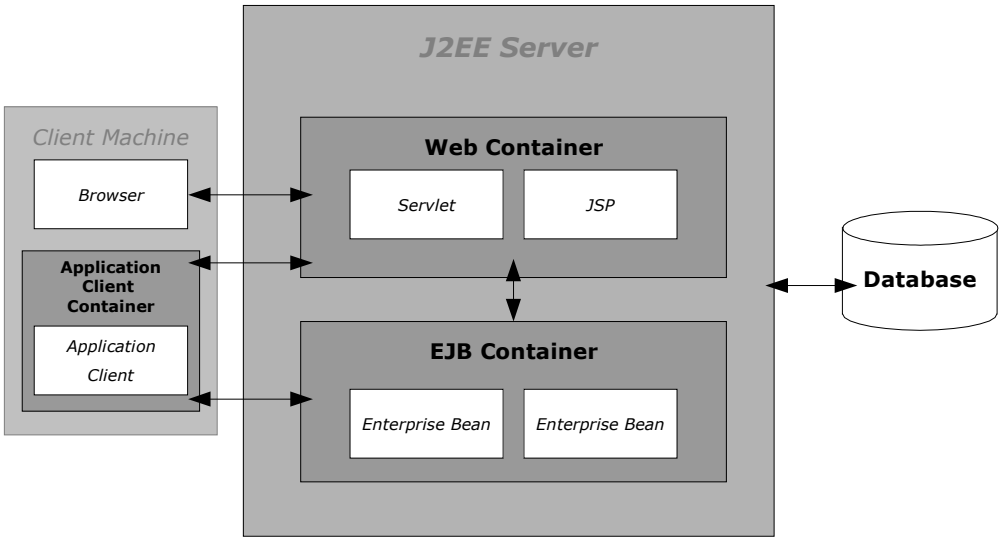

Abbildung 5: J2EE Server und Container

Container vermitteln dabei zwischen Client und Serverkomponenten und bieten Dienste in beide Richtungen, wie z.B. Transaktionsunterstützung und Resource Pooling. Außerdem erlaubt es der Container, bestimmte Verhaltensweisen einer Komponente bei ihrer Installation im Container zu bestimmen, anstatt sie im Quellcode festlegen zu müssen. Konnektoren sind am Rand der J2EE-Plattform angesiedelt. Sie erlauben die Kommunikation zu existierenden Unternehmensapplikationen[112]. Im Rahmen von J2EE-Komponenten wird zwischen drei Arten von Komponenten unterschieden[113]:

111 S. [JavaSun2003], The J2EE 1.4 Tutorial - Chapter 1: Overview, Distributed Multitiered Applications, /j2ee/1.4/docs/tutorial/doc/Overview2.html, Stand: 14.12.2003

112 Vgl. [JavaSun2003], Java 2 Platform, Enterprise Edition (J2EE) Overview, Application Model, /j2ee/appmodel.html, Stand: 13.12.2003

113 S. [JavaSun2003], The J2EE 1.4 Tutorial - Chapter 1: Overview, Distributed Multitiered Applications, /j2ee/1.4/docs/tutorial/doc/Overview2.html, Stand: 14.12.2003

- Komponenten wie **Applikationen** oder **Applets** laufen auf dem Client und werden daher auch als *Client-Komponenten* bezeichnet
- **Java Servletes** und **Java Server Pages** (JSP) sind sogenannte *Web-komponenten* die auf dem Server ihren Dienst verrichten
- **Enterprise Java Beans** (EJB) werden als *Business Komponenten* bezeichnet, die ebenfalls auf dem Server laufen

Für den Einsatz - auch als Deployment bezeichnet - der Serverkomponenten stehen auf einem J2EE Server die entsprechenden Container zur Verfügung.

2.2.4.1 Servlets, JSPs und Web Container

Eine J2EE-Webkomponente kann entweder ein *Servlet* oder eine *JSP* sein. Ein *Servlet* ist eine von einem entsprechenden Container gemanagte Webkomponente, welche dynamsch Inhalte erzeugt. Servlets interagieren mit dem Web Client über ein Anfrage/Antwort-Paradigma (engl. request/ response), welches sich an dem typischen Verhalten bei Verwendung des *Hyper Text Transfer Protocol* (HTTP) orientiert[114]. JSPs sind textbasierte Dokumente, die zu einem Servlet übersetzt und als solches ausgeführt werden. Sie ermöglichen einen einfacheren Umgang mit statischen Inhalten, insbesonderen mit HTML und XML[115].

Der Servlet Container bietet dem Servlet die notwendigen Netzwerkdienste über die Anfrage und Antwort empfangen und versendet werden. Darüber hinaus verwaltet er ein Servlet über seine gesamte Lebensdauer. Minimal muss ein Web Container HTTP zur Kommunikation unterstützen. Zusätzlich ist aber auch die Unterstützung von Protokollen wie HTTP über SSL (HTTPS) üblich.

---

114 Vgl. [SunServletSpec2.3], S.17

115 S. [JavaSun2003], The J2EE 1.4 Tutorial - Chapter 1: Overview, Distributed Multitiered Applications, /j2ee/1.4/docs/tutorial/doc/Overview2.html, Stand: 14.12.2003

2.2.4.2 Der EJB Container

Der EJB Container bietet eine Laufzeitumgebung und zusätzliche Dienste für Java Geschäftsobjekte, die sogenannten *Enterprise Java Beans*. Zusätzliche Dienste werden über Standardprogrammierschnittstellen, sogennante APIs, zur Verfügung gestellt. Zu den Aufgaben des EJB Containers gehören im Einzelnen[116]:

- Aktivierung und Passivierung von Beans
- die Bereitstellung von Namens- und Verzeichnisdiensten
- die permanente Erhaltung von Zustandsänderung einer Bean (Persistenz) auch über den Neustart des Containers hinaus
- Transaktionsmanagement
- Sicherheitsmechanismen

Einen kurzen Überblick der wichtigsten APIs in diesem Zusammenhang liefert der nächste Abschnitt.

Grundsätzlich gibt es zwei Arten von in einem EJB Container verwalteten *Enterprise Beans*, die für die weiteren Ausführungen von Interesse sind: *Entity-* uns *Session-Beans*. Auf den dritten Typ, die mit der EJB 2.0 Spezifikation eingeführte *Message Driven Bean*[117] soll hier nicht weiter eingegangen werden.

Mithilfe von *Session-Beans* werden Abläufe und Vorgänge modelliert, wie z.B. das Anlegen eines neuen Kunden in einem Warenwirtschaftssystem oder die Durchführung einer Buchung in einem Buchungssystem. Im Gegensatz dazu dienen *Entity-Beans* als Datenrepräsentationen realer Dinge, wie z.B. eines Kunden, eines Buchungskontos oder einer Betellung[118]. Sie stellen also letztenendes Datencontainer mit entsprechenden Zugriffsmethoden dar. Für sie muss der Container ggf. Persistenz- und Transaktionsmechanismen bereitstellen.

---

116 Vgl. [DenningerPeters2000], S.24
117 S. [SunEJBSpec2.0], S.25
118 Vgl. [DenningerPeters2000], S.30

2.2.4.3 JNDI, JDBC und andere APIs

Die im Folgenden beschriebenen APIs sind insbesondere deshalb von besonderer Wichtigkeit, als dass sie wesentlicher Teil der J2EE-Platform sind. Darüber hinaus schreibt die EJB-Spezifikation ihre Bereitstellung im Container vor[119].

Die *Java Naming and Directory Interface* API (JNDI) bietet Applikationen Methoden an, um Standardoperation auf Verzeichnisdienste durchzuführen, wie die Verknüpfung von Attributen mit Objekten und das Suchen innerhalb des Verzeichnisdienstes. Es kann verwendet werden, um beispielsweise Verbindungen zu Datenbankservern zur Laufzeit anzufordern. Die *Java Transaction API* (JTA) liefert eine Standardschnittstelle für die Abgrenzung von Transaktionen gegeneinander. Das Java Mail API kann verwendet werden, um Benachrichtigungen per E-Mail zu verschicken.

JDBC bietet eine API für Datenbankoperationen innerhalb von Java Applikationen[120]. Datenbankverbindungen werden dabei auf Objekte vom Typ *java.sql.Connection* abgebildet, die vom JDBC-Treiber-Manager verwaltet werden. Dieser kann mehrere Verbindungen zu unterschiedlichen Datenbanken gleichzeitig verwalten. Hierzu setzt er auf einem datenbankspezifischen Treiber auf, der nach den Vorgaben des JDBC Standards in der Regel vom Hersteller des Datenbankprodukts zur Verfügung gestellt wird. Man unterscheidet vier verschieden Typen von JDBC-Treibern: Typ 1 setzt auf einen nativen ODBC Treiber auf, der auf dem entsprechenden System installiert sein muss. Bei Typ 2 wird anstatt auf ODBC auf einen nativen herstellerabhängigen Treiber aufgesetzt. Im Falle von Typ 3 erfolgt der Zugriff an Stelle eines lokal installierten Treibers über eine serverseitige Datenbank-Middleware-Komponente. Typ 4 schließlich ist vollständig in Java implementiert und gilt daher auch als die modernste Variante[121].

Darüber hinaus definiert die J2EE-Spezifikation Standardschnittstellen zur Verarbeitung von XML. Im Einzelnen sind dies die *Java API for XML*

119 Vgl. [DenningerPeters2000], S.24
120 S. auch Kapitel 2.2.1.4: Datenbankzentrierte Produkte
121 Vgl. [Hartwig1998], Stand: 11.01.2004

Processing (JAXP), *Java API for XML Registries* (JAXR) und *Java API for XML-based RPC* (JAX-RPC), sowie die *SOAP with Attachments API for Java* (SAAJ). Neben den hier vorgestellen APIs werden noch weitere für den Entwurf von Standardschnittstellen zu Informationssystemen, zur Authentifikation und Autorisation von Benutzern und Benutzergruppen, sowie zur Interaktion mit JavaBeans definiert[122]. Diese haben aber keinerlei Relevanz für dies Arbeit und sollen von daher auch nicht namentlich genannt werden.

3. Konzept des Integrationsansatzes

Wie zu Beginn des ersten Kapitels bereits dargelegt, liegt dieser Arbeit die Aufgabe zugrunde, eine Integrationslösung für ein Warenwirtschaftssystem zu Entwickeln. Der Erörterung der genauen Problemstellung wird sich der folgende Abschnitt widmen. Im Anschluss werden dazu einige architektonische Überlegungen angestellt, um schließlich jene Werkzeuge und Produkte vorzustellen, die zur Lösung des Problems herangezogen werden sollen.

3.1 Erörterung der Problemstellung

Bei dem zu integrierenden Warenwirtschaftssystem handelt es sich um das Produkt *System21* der Firma *Geac*, insbesondere um das Vertriebsmodul *VER*. Als Hardwareplattform dient dem ERP-System eine iSeries, früher AS/400, von IBM. Da es zu *System21* nach Auskunft des Herstellers keine standardisierte APIs gibt, ist ein Integrationsansatz über Funktionsaufrufe, wie in Kapitel 2.1.1 erläutert, ohne weiteres nicht möglich.

Geac bietet zur webbasierten Nutzung von System21 eine Lösung an, bei der die ursprünglich textbasierten Masken des Systems interpretiert und in Form eines Java Applets in einem Browser dargestellt werden. Für den Hersteller des Systems mag das ein gangbarer Weg sein, insbesondere, da

122 S. [JavaSun2003], The J2EE 1.4 Tutorial - Chapter 1: Overview, J2EE APIs, / j2ee/1.4/docs/tutorial/doc/Overview7.html, Stand: 14.12.2003

das Applet die Informationen exakt so wiedergibt, wie sie in den textbasierten Masken dargestellt werden. Für die Integration im Rahmen des CRM-Projektes, die bei der Datenabfrage deutlich mehr Flexibilität erfordert, ist dies kein sinnvoller Ansatz.

Zur Datenhaltung verwendet das ERP-Produkt eine *DB2* Datenbank von IBM. Da zu dieser Datenbank eine Vielzahl von Schnittstellen, insbesodere standardisierte APIs wie ODBC und JDBC, verfügbar sind, ist die Integration über Datenhaltung in diesem Fall ein vernünftiger Weg.

Die Entwicklung der CRM-Lösung erfolgt auf Basis von PHP 4. Wie unter [IBM_PHP_DB2_2001][123] beschrieben, ist die direkt Anbindung von PHP an DB2 prinzipiell möglich. Zu Beginn von Kapitel 1 wurde allerdings bereits dargelegt, dass eine der Hauptforderungen an die Integrationslösung darin besteht, eine Basis zu schaffen, die auch in weiteren Projekten verwendet und weiter ausgebaut werden kann. Dies Forderung spricht gegen eine reine PHP Lösung und eine damit einhergehende enge Verzahnung mit der CRM-Applikation. Es muss ein geeigneterer Rahmen gefunden werden, der ausreichend Spielraum für zukünftige Anforderungen lässt.

In Kapitel 2 wurde ein allgemeiner Überblick über Techniken und Anforderungen an Integrationslösungen gegeben. Nicht alle dort genannten Aspekte sind in diesem ersten Schritt relevant. Trotzdem soll eine Plattform gewählt werden, die zu allen besprochenen Aspekten entsprechende Lösungen bereithält. Es wird nach den vorangegangenen Ausführungen nicht überraschen, dass man sich dabei für die *Java 2 Platform*, *Enterprise Edition* entschieden hat, da sie Konzepte, APIs und Tools für alle diskutierten Probleme bereithält und sich darüber hinaus am Markt bewährt hat.

Die Aufgabe besteht also darin, jene Funktionalität in Java zu implementieren, die benötigt wird, um mit dem ERP-System, respektive der darunter liegenden DB2 Datenbank zu interagieren. Darüber hinaus muss eine praktikabler Weg gefunden werden, um die implementierte Programmlogik im Rahmen der PHP Applikation nutzen zu können.

123 S. [IBM_PHP_DB2_2001], Stand: 29.12.2003

Insgesamt muss dies auf eine Art und Weise erfolgen, die spätere Erweiterungen sowohl der Funktionalität bezogen auf das ERP-System als auch die Integration weiterer Systeme möglich macht und unterstützt. Insbesondere die in Kapitel 2.1.5 aufgeführten Sicherheitsanforderungen gilt es, bei der Implementierung zumindest in wesentlichen Teilen zu berücksichtigen.

Um den Umfang dieser Arbeit nicht zu sprengen, wird die Umsetzung exemplarisch anhand von im Warenwirtschaftssystem angesiedelten Entitäten wie Adressen, Produkten und Rabattkonditionen dargelegt. Es erscheint zweckdienlich, ausschließlich lesende Zugriffe zu behandeln, da schreibende Operationen in ihrer Komplexität umfangreichere Darstellungen erfordern und dabei der Blick für den Gesamtzusammenhang leiden würde. Transaktionsmechanismen, wie unter 2.1.6 erläutert, werden dabei allerdings nicht zum Einsatz kommen. Insbesondere, da hiser nur lesende Operationen behandelt werden sollen. Es werden aber Techniken gewählt werden, die entsprechende spätere Erweiterungen möglich machen.

3.2 Architektonische Überlegungen

Auf Basis der Ausführungen des letzten Abschnitts ergibt sich für das Gesamtsystem, in Anlehnung an das in Kapitel 2.1.1 erörterte Schichtenmodell, das in Abbildung 6 dargestellte Schema. Auf die Betrachtung architektonischer Details der CRM-Applikation soll an dieser Stelle bewusst verzichtet werden. Vielmehr geht es im Folgenden um den Aufbau der Integrationslösung und die Anbindung an die PHP-Anwendung.

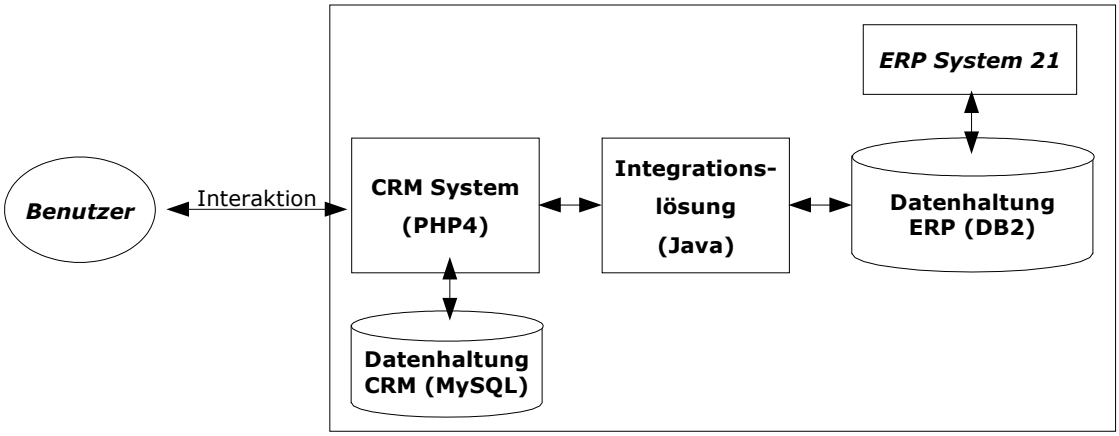

Abbildung 6: Schematische Darstellung des Gesamtsystems

3.2.1 Anbindungsmöglichkeiten von PHP und Java

PHP 4 biete eine Vielzahl von Möglichkeiten um mit seiner „Umwelt" zu kommunizieren. In der Dokumentation heißt es dazu:

„... PHP unterstützt auch die Kommunikation mit anderen Services, welche Protokolle wie LDAP, IMAP, SNMP, NNTP, POP3, HTTP, COM (auf Windows) und unzählige andere unterstützen. Sie können auch einfache Netzwerk Sockets öffnen, und unter Verwendung irgendeines Protokolls kommunizieren. ... Da wir gerade vom Zusammenwirken sprechen: PHP bietet auch Unterstützung zur Instantiierung von Java Objekten, und deren transparente Verwendung als PHP Objekte. Sie können auch unsere CORBA Erweiterung verwenden, um auf entfernte Objekte zuzugreifen. ..."[124]

Über die entsprechenden Möglichkeiten in Java wurde im vorangegangenen Kapitel bereits eingegangen. Es ist davon auszugehen, dass für die gängigen von PHP unterstützten Protokolle und Techniken auch APIs in Java zur Verfügung stehen.

3.2.1.1 RPC und Messaging

Wie bereits in Kapitel 2.2.2.5 dargelegt, verfügt die *Java 2 Platform* mit *RMI* über einen eigenen Mechanismus um Methoden entfernter Objekte

124 S. [PHPmanual], I. 1. Was kann PHP?, Stand: 02.01.2004

aufzurufen. Dies setzt die Verwendung entsprechender Java Stubs[125] voraus, was wiederum die Verwendung von Java Objekten in PHP notwendig machen würde. Mit dieser Möglichkeit wird sich der anschließende Abschnitt eingehender befassen.

Bleibt die Betrachtung gängiger, programmiersprachenübergreifender RPC-Mechanismen und deren Unterstützung in PHP. Gewissermaßen „von Haus aus" unterstützt PHP das *Component Object Model* (COM). Wie bereits dargelegt ist ein Nachteil dieser Technologie ihre enge Bindung an das Betriebssystem Windows. So stehen die sogenannten *COM Support Funktionen* auch leider nur für die Windowsvariante von PHP 4 zur Verfügung[126], was eine Verwendung im Rahmen dieses Projektes ausschließt.

Darüber hinaus ist in dem Zitat zu Beginn des letzten Abschnitts die Rede von einer CORBA Erweiterung für PHP. Eine direkte Unterstützung durch den PHP-4-Standardfunktionsumfang gibt es allerdings nicht. Fündig wird man hingegen im *PEAR Repository*, dem Verzeichnis für PHP Standard-erweiterungen. Die dort zu findende *Satellite CORBA client extension* gilt allerdings mittlerweile schon als 'abgelehnt'[127] (engl. deprecated). Ihre Nachfolge wurde von *Universe* angetreten, einem Projekt in dem aber noch sehr viel im Umbruch zu sein scheint[128].

Für eine Scriptsprache, deren Verwendung vorwiegend auf die Entwicklung von Webapplikationen abzielt[129] ist sicherlich ein Blick auf die entsprechenden Integrationsansätze von Interesse. Das Thema *Web Services* wird auch von PHP breit besetzt. Funktionen zur Verwendung von XML-RPC sind bereits Teil des Standardfunktionsumfangs[130]. Eine Liste mit weiteren Implementierungen dieses Web Service-Standards für PHP bietet darüber hinaus die XML-RPC Home Page[131].

---

125 S. Kapitel 2.2.2.1, *Stub und Skeleton*
126 S. [PHPmanual], V. VIII. COM Support Funktionen für Windows, Stand: 02.01.2004
127 Vgl. [PEARmanual], VII. Satellite CORBA client extension, Stand: 02.01.2004
128 Vgl. [Universe_PHPExtension], Stand: 02.01.2004
129 S. [PHPmanual], I. 1. Was ist PHP?, Stand: 02.01.2004
130 S. [PHPmanual], V. CXII. XML-RPC functions, Stand: 02.01.2004
131 S. [XML-RPC.com], /directory/1568/implementations, Stand: 07.12.2003

Auch für SOAP, den anderen großen Web Service-Standard existieren eine Reihe von Implementierungen. Dabei sind vor allem PEAR::SOAP, PHP-SOAP und NuSOAP zu nennen[132]. Auf das NuSOAP Toolkit wird im Folgenden noch weiter eingegangen.

3.2.1.2 Direkte Einbindung von Java in PHP

Laut Dokumentation bietet PHP in der Version 4 die Möglichkeit, Java Objekte zu instanzieren und deren Methoden aufzurufen. Dabei erfolgt der Zugriff auf die Java *Virtual Machine* (VM) über das *Java Native Interface* (JNI)[133], der Schnittstelle von Java zu der darunterliegenden Softwareplattform. Über dieses Interface kann ein Java Programm beispielsweise unter Windows eine dynamisch ladbare Bibliothek in Form einer .dll-Datei einbinden oder unter einem Unix Betriebssystem das entrechtende Pendant. Umgekehrt funktioniert dieser Weg allerdings auch: So kann z.B. ein C-Programm über die so genannte Invocation-API auf Java-Programme zugreifen[134]. Diese Tatsache haben sich die PHP Entwickler zu nutze gemacht um den Zugriff auf Java Objekte aus PHP zu realisieren. Es wäre daher denkbar, die benötigte Funktionalität für den Zugriff auf das ERP-System in Java zu Implementieren und diese Objekte entweder direkt, oder entsprechende Stub-Objekte für den Zugriff über RMI in PHP zu verwenden.

Leider ist die Implementierung der JNI-Anbindung nicht stabil. Vielmehr scheint das Gelingen der Instanzierung eines Java Objekts vom Zufall abzuhängen. Auf einen erfolgreichen Durchlauf folgen in der Regel einige in denen ein Versuch von PHP mit der Meldung 'Fatal error: Unable to create Java Virtual Machine' quittiert wird. Das Problem ist bekannt. In der *PHP 4 Bug Database* finden sich mehrere Einträge zu dem Thema[135]. Leider ist in absehbarer Zeit nicht mit einer Lösung zu rechnen, was ein entsprechendes Vorgehen nach diesem Ansatz für das laufenden Projekt unmöglich macht.

---

132 Vgl. [ActiveStateWsTutorial], Chart 35: PHP and SOAP, Stand: 02.01.2004
133 S. [PHPmanual], V. XLVII. Java, Stand: 02.01.2004
134 Vgl. [JavaInselOpenbook], 23.1 Java Native Interface und Invocation-API, Stand: 02.01.2004
135 S. [PHP4BugDatabase], Einträge 16690 und 18600, Stand: 02.01.2004

3.2.2 Gestaltung der Integrationsplattform

Neben den Fragen nach den Möglichkeiten der Anbindung von PHP und Java, muss auch der sinnvolle Einsatz der Container und APIs der J2EE-Plattform diskutiert werden. Wie in Kapitel 2.2.4 erläutert unterscheidet die J2EE-Spezifikation zwei Arten von Komponenten zur Modellierung der Applikationsschicht mit ihren entsprechenden Containern: Webkomponenten in Form von *Servlets* und *JSP*s auf der einen Seite und *Enterprise Java Beans* (EJBs) auf der anderen.

Der klassische Ansatz bei der Verwendung von EJBs ist die Modellierung von Fach- und Anwendungslogik in den so genannten *Session-Beans*, getrennt von den als Datenhaltungsobjekte verwendeten *Entity-Beans*[136]. Dies erfordert bei der üblichen Datenhaltung in relationalen Datenbanken die Umsetzung entsprechender Konzepte und Mechanismen zur Abbildung der objektorientierten Strukturen auf Tabellen und Attribute der Datenbank. Wird der EJB Container für die Anbindung eines Altsystems verwendet, müssen die zur Kommunikation verwendeten Datenstrukturen ebenfalls auf Enitiy-Beans abgebildet werden und umgekehrt. Erfolgt die weitere Anbindung an die vorgelagerten Schichten nicht über Mechanismen, die objektorientierte Konzepte umfassen, müssen die modellierten Objekte an dieser Stelle wiederum auf andere Datenstrukturen abgebildet werden. Ein Aufwand, der spätestens dann als fragwürdig erscheint, wenn die objektorientierte Struktur, abgesehen von der internen Verwendung im EJB Container, niemals zum Einsatz kommen. Eine Situation wie sie beispielsweise gegeben ist, wenn der Client, der die Dienste des EJB Containers in Anspruch nimmt, keine objektorientierten Konzepte unterstützt oder diese nicht verwendet werden.

[ZieglerModernDesign] empfiehlt als Architekturansatz zur Lösung dieses Dilemmas die Verwendung der existierenden Datenstrukturen, ggf. in geringfügig überarbeiteter Form unter Verwendung von HashMaps und Arrays, insbesondere bei der Nutzung von XML zur Kommunikation mit den

136 Vgl. Kapitel 2.2.4.2: Der EJB Container

übrigen Applikationsschichten[137]. Schließt sich die Frage an, ob in einem solchen Fall die Verwendung eines EJB Containers, respektive die Verwendung von *Session-Beans* zur Implementierung der Fach- und Anwendungslogik noch Vorteile gegenüber einer reinen Implementierung mit Webkomponenten bietet? Zumal beide Containertypen in der Regel Dienste wie JNDI, Datenbankverbindungs- Pooling etc. bereitstellen. Außerdem bieten aktuelle Web Service-Werkzeuge, wie wir noch sehen werden, einen stabilen architektonischen Rahmen für strukturierte Anwendungsentwicklung. Ohnehin würde zur Erbringung der Web Service-Funktionalitäten der Web Container dem EJB Container vorgeschaltet.

3.2.3 Schlussfolgerungen der Überlegungen

Das Scheitern des Ansatzes der direkten Einbindung von Java Objekten in PHP und die Tatsache, dass die CORBA Unterstützung für PHP noch keinen Produktionsstatus erlangt zu haben scheint, lässt den Integrationsansatz über Web Services in den Mittelpunkt des Interesses rücken. Dieser Ansatz findet zusätzlich Unterstützung durch den Umstand, dass es sich bei dem Anlass der Integrationsbemühungen, sowieso um ein Webprojekt handelt. Demnach trifft er hier auf optimale Voraussetzungen. Darüber hinaus findet Integration über Web Service mittlerweile breite Unterstützung am Markt und muss daher als die Lösung mit der besten Zukunftsperspektive betrachtet werden.

In dem Kapitel über den Web Service-Standard SOAP[138] wurde erläutert, dass dieser nicht nur für die Realisierung von XML basierten RPC-Mechanismen, sondern weitaus breiter ausgelegt ist. Insbesondere in Zusammenhang mit dem zu Beginn auch bei Web Services verbreiteten Ansatz Datenhaltungsobjekte, so genannten „value Objects", in XML serialisiert zu übertragen, muss man sich die schon im Zusammenhang mit *Entity-Beans* aufgeworfene Frage stellen, ob die dazu benötigten Systemresourcen und der hohe Entwicklungsaufwand in einem ange-

137 Vgl. [ZieglerModernDesign]
138 S. Kapitel 2.2.3.3: SOAP

messenen Verhältnis zum Ergebnis stehen. Nachdem die ersten SOAP-Werkzeuge zunächst an dieser Idee festhielten, werden mittlerweile verstärkt Lösungen diskutiert, in denen XML Dokumente direkt ausgetauscht und verarbeitet werden. Die Vorteile dieses Ansatzes sind in [ZieglerModernDesign] wie folgt dargelegt[139]:

- Client und Applikation-Server sind vollständig entkoppelt. Das Hinzufügen oder Entfernen einzelner fachlicher Attribute ändert die Schnittstelle aus technischer Sicht nicht. Hierdurch wird eine unabhängige parallele Entwicklung von Client und Applikations-Server ermöglicht.
- XML-Datenströme sind selbstdokumentierend. Die textuelle Natur von XML-Tags und ihre Verwendung in einem hierarchischen Datenschema reduziert die Gefahr von Fehlinterpretationen während des Entwicklungsprozesses.
- Dem Client können als Ergebnis eines einzelnen Aufrufs gleichzeitig Daten und multiple Plausibilitätsverletzungen übermittelt werden.
- Über XML-basierte Schnittstellen können leicht zusätzliche technische Informationen wie Zeitstempel oder Journalisierungs- und Protokolldaten transportiert werden.
- SOAP als Kommunikationsprotokoll zwischen Client und Applikations-Server kann unterstützt werden, ohne anwendungsspezifische SOAP-Serializer schreiben zu müssen.

Ein gängiges Argument gegen diese Art der Schnittstelle gegenüber einer objektorientierten Lösung sind die fehlenden Compiler-Prüfungen für Ein- und Ausgangsparameter. Bei der Verwendung von PHP, einer nicht streng getypten Skriptsprache auf Seite des Klienten, greift dieses Argument allerdings von vorn herein nicht. Der Versuch strenger objektorientierte Konzepte auf PHP abbilden zu wollen ist auch in so fern kritisch zu bewerten, als das wesentliche objektorientierte Konzepte, wie das Überladen von Methoden, in PHP fehlen[140].

139 Aus [ZieglerModernDesign]
140 S. [PHP4UTutorial], 11. Polymorphie, Überladen von Funktionen, Stand: 04.01.2004

Bei der Entwicklung der Integrationsplattform wird als Schlussfolgerung daraus XML-basierte Kommunikation zwischen Client und Server verwendet. Neben der Verwendung von SOAP wird ein Ansatz favorisiert, bei dem die im SOAP Body versendeten XML-Dokumente gezielt erstellt und verarbeitet werden. Bei der Beschreibung der Schnittstellen wird der Fokus dabei in der Definition des Aufbaus der eingehenden und ausgehenden XML Dokumente liegen.

Die Entwicklung der Integrationsplattform wird als reine Webkomponente erfolgen. Dies verringert die Komplexität der Lösung und macht so eine weitgehend vollständige Darstellung im Rahmen dieser Arbeit möglich.

3.3 Darstellung des Lösungsansatzes

Aufbauend auf den bisherigen Ausführungen lässt sich nun ein schärferes Bild der Lösungsansatzes zeichnen.

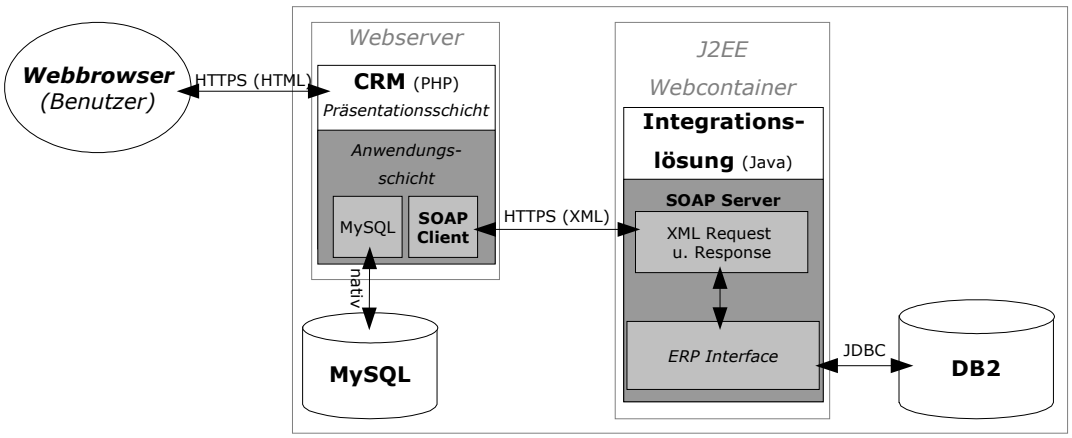

Abbildung 7: Schematische Darstellung der angestrebten Integrationslösung

Abbildung 7 zeigt schematisch das Zusammenspiel der CRM-Applikation und der Integrationslösung. Die Darstellung soll kurz erläutert und im Anschluss die zur Realisierung benötigten Werkzeuge vorgestellt werden.

Zunächst benötigt die CRM-Webapplikation eine angemessene Laufzeit-umgebung, einen Web Server mit PHP Unterstützung. Wie die Abbildung

andeutet, folgt ihr Aufbau dem klassischen Drei-Schichten-Modell. Zur Implementierung der Anwendungsschicht werden Werkzeuge zur Kommunikation mit der Integrationsplattform benötigt, die im Folgenden als *SOAP Client* oder *Toolkit* bezeichnet werden. Sie dienen dem Codieren, Versenden und Empfangen der SOAP-Nachrichten.

Die Integrationsplattform wird als Servletlösung in einem entsprechenden Container implementiert. Natürlich werden auch hier entsprechende Werkzeuge benötigt, welche die erforderliche Funktionalität zum Betreiben des *SOAP Servers* bereitstellen. Die weitere Kommunikation mit der DB2 Datenbank des ERP-Systems erfolgt über JDBC, die standardisierte Datenbankschnittstelle für Java. Hierzu wird ein entsprechendes Interface implementiert, welches alle Methoden zur Kommunikation mit dem Warenwirtschaftssystem bereitstellt.

Darüber hinaus benötigen beide Kommunikationspartner Werkzeuge zur Interpretation und Erstellung der XML-Nachrichten, die mittels SOAP ausgetauscht werden sollen. Zur Interpretation, dem so genannten Parsen, von XML existieren zwei populäre Standardschnittstellen: Das *Document Object Model* (DOM) und *Simple API for XML* (SAX). Mit dem DOM wird ein Baum als Abbild des XML Dokuments im Arbeitsspeicher erzeugt, durch den anschließend über entsprechende Methoden navigiert werden kann. SAX ist eine ereignisbasierter Parser[141]. Bei der Verwendung von SAX wird das XML Dokument wie ein Eingabestrom behandelt. Die Applikation kann dabei Funktionen beim Parser anmelden, die aufgerufen werden, wenn dieser bestimmte Elemente in dem XML-Dokument findet[142]. Beide APIs sind sowohl für Java als auch PHP verfügbar. Wie wir noch sehen werden unterstützen die Implementierungen zum Teil auch einige weitere nützliche Erweiterungen zur Bearbeitung von XML.

Um die in Kapitel 2.1.5 beschriebenen Sicherheitsanforderungen umzusetzen, müssen mehrere Techniken miteinander kombiniert werden. *Vertraulichkeit* wird über die Verschlüsselung der Daten erreicht. Hierfür

141 Vgl. [W3C_DOM_ActivityStatement], Stand: 05.01.2004
142 Vgl. [ExpatSourceforgeNet], Startseite, Stand: 08.01.2004

wird HTTPS, also HTTP über den *Secure Sockets Layer* (SSL) zum Einsatz kommen[143]. Zum Thema *Identifikation* und *Authentizität* liefert die SOAP-Spezifikation keine eigenen Konzepte[144]. Prinzipiell ist es möglich, Benutzernamen und Passwort z.B. im SOAP Header zu transportieren. Allerdings gibt es auch andere Weg: Die Möglichkeit der Versendung von Username und Passwort im Header des HTTP machen sich beispielsweise viele Webapplikationen zu nutze, um bestimmte Bereiche zu schützen. Diese Methode ist in RFC 2617 als *HTTP Basic* bzw. *Digest Access Authentication* beschrieben[145]. Sie wird von [Benoist2002], in Kombination mit SSL, als ein einfacher und effektiver Ansatz dargestellt, SOAP-Dienste sicher zu gestalten[146]. Daher wird sie auch im Rahmen dieses Projektes verwendet. Weitere *Zugangskontrollen*, also die Prüfung der Zugangsberechtigung zu einzelnen Diensten der Integrationsplattform, wird es insbesondere deshalb nicht geben, weil der Konsument der Dienste, in diesem Falle die CRM-Applikation, zunächst als einziger Benutzer auftritt, der generell Zugriff auf alle angebotenen Daten benötigt. Eine Prüfung, in wieweit der Benutzer der Client-Applikation Zugang zu Daten aus dem Backend-System erhalten darf, muss also in dieser Schicht, in diesem Falle in der CRM-Anwendung, erfolgen. Zur späteren *Nachvollziehbarkeit* können alle über die Integrationsplattform abgewickelten Aktionen in einer Datei protokolliert werden. Einen zusätzlichen Mechanismus zum Schutz der *Integrität* der Daten wird es nicht geben. Die Tatsache, dass diese verschlüsselt übertragen werden, wird in diesem Zusammenhang als hinreichend betrachtet. Die Verfügbarkeit des Systems wird schließlich durch eine ausreichende Dimensionierung der für die Integrationsplattform verwendeten Hardwareplattform erreicht.

3.4 Client, Server und Tools

Nachdem die technischen Anforderungen an die einzelnen Teile des Systems und wie diese miteinander interagieren erläutert wurde, werden im

143 Vgl. [NetscapeTechbriefsSSL], Stand: 08.01.2004
144 S. [W3C_SOAP1.2_Part1], 7. Security Considerations
145 S. [RFC2617], Stand: 08.01.2004
146 S. [Benoist2002], Stand: 08.01.2004

Folgenden die Werkzeuge zur Realisierung des Vorhabens vorgestellt. Den meisten Raum wird dabei die Beschreibung der Java-Komponenten für die Umsetzung der Integrationsplattform einnehmen. Es soll aber auch ein kurzer Blick auf die Schnittstellen zur DB2 Datenbank geworfen werden. Zu Beginn werden die zur Anbindung benötigten PHP Werkzeuge auf Seite der CRM-Applikation vorgestellt.

3.4.1 Werkzeuge für PHP

3.4.1.1 XML Parser

Auch wenn die Erzeugung von XML-Dokumenten ohne weitere Hilfsmittel durch die einfache Aneinanderreihung von Tags und Inhalten zu einem String möglich ist, erfordert die umgekehrte Interpretation von XML-Daten ausgefeiltere Werkzeuge. Zu diesem Zweck besitzt PHP 4 Erweiterungen die sowohl das SAX, als auch das DOM API implementieren.

Die Implementierung des ereignisgesteuerten SAX in PHP basiert auf der schnellen und robusten *eXpat Libary* von James Clark, die ursprünglich für den *Mozilla* Browser entwickelt wurde. Demnach gelten die SAX-Funktionen von PHP ebenfalls als tauglich und leistungsfähig[147].

Das DOM API für PHP macht Gebrauch von *LibXML*, der XML Bibliothek für den X-Windows Fenstermanager *Gnome*. Es folgt dabei so weit wie möglich der DOM Level 2 Spezifikation, mit leichten Anpassungen an PHP Programmierstandards. Außerdem wird die *XPath* Erweiterung zur direkten Ansprache von Unterknoten eines XML Dokuments unterstützt[148].

Warum diese beiden Bibliotheken als Basis für die Implementierung der APIs in PHP dienen, tritt bei einem Blick auf einen umfassenden XML-Benchmark zu tage. Bei einem Vergleich von *Expat*, *Xerces*, *LibXML* und den Bibliotheken von Oracle und Sun zeigt sich beide in dem Bereich, indem sie auch in PHP eingesetzt werden, als die schnellste Lösung[149].

147 Vgl. [Vaswani2002], S.21
148 S. [PHPmanual], V. XXIV. DOM XML Funktionen, Stand: 07.01.2004
149 S. [XMLbenchSourceforgeNet], Overall, Stand: 08.01.2004

Zur Interpretation der über SOAP von der Integrationsplattform gelieferten XML Dokumente durch die PHP Applikation, wird SAX ausreichen, insbesondere da es die effizientere Lösung darstellt.

3.4.1.2 SOAP Toolkit

Eine SOAP-Nachricht ist nichts anderes als ein XML-Dokument, dessen Aufbau den in der Spezifikation beschriebenen Regeln folgt[150]. Vorzugsweise wird sie über das HTTP Protokoll transportiert. Für die Implementierung eines SOAP Clients oder Servers benötigt man demnach nur einen XML-Parser und Funktionen zur HTTP Unterstützung. Wenn man so weit gehen möchte sich mit dem Aufbau des HTTP auseinanderzusetzen, reicht auch die Möglichkeit, Verbindungen über Netzwerk-Sockets herzustellen. Komfortabler ist selbstverständlich die Verwendung eines fertigen Werkzeugs zur SOAP-Unterstützung, wovon mehrere für PHP erhältlich sind. Für die Implementierung der Anbindung an die Integrationsplattform wird auf das *NuSOAP* Paket von Dietrich Ayala zurückgegriffen.

NuSOAP ist ein Toolkit, welches eine einfache API zur Verwendung von SOAP-basierten Web Services bietet. Mit Hilfe dieses Pakets können sowohl SOAP Clients als auch Server in PHP implementiert werden. Es unterstützt wichtige Feature wie die Erzeugung von WSDL, die Generierung von Proxy-Klassen, die Verwendung von SSL und HTTP Authentifizierung[151]. Damit stellt es alle Techniken zur Verfügung, die für die Entwicklung der Anbindung an die Integrationsplattform benötigt werden[152].

Die Installation des Toolkits gestaltet sich denkbar einfach. Es muss einfach in den *Includepath* von PHP kopiert werden, also in eines der Verzeichnisse in denen PHP nach Dateien zum Einbinden sucht. Ein Nachteil dieses Pakets ist seine etwas dürftig ausfallende Dokumentation[153]. Durch seinen gut strukturierten Aufbau können offene Fragen allerdings gut durch einen Blick in den Quellcode beantwortet werden. Außerdem stehen im Web einige

150 S. Kapitel 2.2.3.3: SOAP
151 Vgl. [PHPBuilderNuSOAP], Stand: 09.01.2004
152 Vgl. Kapitel 3.3: Darstellung des Lösungsansatzes
153 S. [ZendNuSOAP], Stand: 09.01.2004

weitere Quellen, wie Artikel und Weblogs, die sich speziell mit der Verwendung von *NuSOAP* beschäftigen, zu Verfügung.

3.4.2 Java Applikations-Server und Werkzeuge

Die Spezifikation der *Java 2 Platform*, *Enterprise Edition* definiert ein Komponentenmodell für die Entwicklung von Unternehmensapplikationen und entsprechende Anforderungen an die Container, die den Komponenten als Laufzeitumgebung dienen. Außerdem definiert sie eine Reihe von APIs zur Lösung von gängigen Problemen, wie sie bei der Entwicklung von Enterprise-Applikationen entstehen[154]. Der Urheber der Spezifikation *Sun Microsystems* bietet neben dem J2EE *Software Development Kit* (SDK) eine Reihe teilweise kostenloser und teilweise kostenpflichtiger Implementierungen der APIs und Container an. Es steht allerdings jedermann frei, ebenfalls entsprechende Implementationen des J2EE-Standards zu entwickeln und zu vertreiben.

Unter dem Dach der *Apache Software Foundation*[155] ist das *Jakarta Projekt* beheimatet. Im Rahmen dieses Projektes werden Open-Source-Lösungen für die *Java 2 Platform* entwickelt, die vorwiegend auf die serverseitige Verwendung abzielen. Die Lösungen erheben den Anspruch, kommerzielle Reife zu besitzen, werden aber trotzdem grundsätzlich kostenlos zur Verfügung gestellt[156]. Zur Implementierung der Integrationsplattform werden, neben dem SDK von Sun, ausschließlich Komponenten aus Apache Projekten verwendet. Der folgende Abschnitt wird darüber einen kurzen Überblick geben.

3.4.2.1 Apache AXIS

Grundsätzlich ist *Axis* eine SOAP Engine, also ein Framework zum Aufbau von Komponenten zur Verarbeitung von SOAP-Nachrichten, wie Clients, Server und Gateways. Darüber hinaus umfasst es auch Werkzeuge für die

154 Vgl. Kapitel 2.2.4: Java 2 Platform, Enterprise Edition (J2EE)
155 Vgl. Kapitel 1.4.1: Geschichte und Techniken des Internet: Der Apache Web Server und die Folgen
156 S. [ApacheJakartaProject], /site/mission.html, Stand: 10.01.2004

Unterstützung für WSDL, mit deren Hilfe z.B. Java Klassen aus WSDL-Beschreibungen erzeugt werden können. Zunächst wurde *Axis* in Java implementiert. Mittlerweile existiert aber auch ein Unterprojekt, welches an der Portierung der Axis-Engine auf C++ arbeitet. *Axis* ist die dritte Generation der *Apache SOAP*-Implementierung, welche ihre Anfänge als *SOAP4J* bei *IBM* nahm. Sie kann als einfacher stand-alone Server betrieben werden[157], üblich und entsprechend dokumentiert ist allerdings die Verwendung innerhalb eines Servlet-Containers[158]. *Axis* gilt als die verbreitetste SOAP Engine für Java. Einige kommerzielle Anbieter verwenden sie als SOAP-Implementierung im Rahmen ihrer eigenen Produkte, wie z.B. *Apple* in *WebObjects*, *Borland* im *Enterprise Server* und *IBM* in mehreren Produkten[159].

Axis basiert auf einer ausgefeilten Architektur, basierend auf sogenannten *Handlern*, die in Ketten (engl. Chains) hintereinander geschaltet werden. *Handler* verarbeiten z.B. die verschiedenen Transportprotokolle, den SOAP Header, oder dienen einfach zum Protokollieren von Informationen. Das Zusammenspiel und die sich daraus ergebenden Ketten können mit Hilfe von XML konfiguriert werden. Im mit *Axis* implementierten SOAP Server wird zuletzt immer ein *Service-Handler* aufgerufen, dessen Aufgabe es ist, die Methode des Objektes anzusprechen, welches die dienstspezifische Funktionalität umsetzt[160].

Die Implementierung eines Serverdienstes mit *Axis* kann auf verschieden Arten erfolgen. Diese sind grundsätzlich danach zu unterscheiden, in welcher Form der Inhalt des SOAP Bodys an die implementierende Service-Methode übergeben wird. Entweder es wird direkt als DOM-Objekt weitergereicht und so XML-nah verarbeitet, oder es erfolgt eine Abbildung (engl. mapping) auf ein spezifiziertes Java-Objekt, Standarddatentypen oder eine entsprechende JavaBean, die der Struktur des Dokuments entspricht. Bei der Abbildung auf Objekte unterstütz *Axis* die *JAX-RPC* Spezifikation der

157 Vgl. [ApacheAxisDocumentation], User's Guide, Stand: 10.01.2004
158 Vgl. [ApacheAxisDocumentation], Installation Instructions, Stand: 10.01.2004
159 S. [ApacheWikiAxisProjectPage], /Compare und /AxisBeingUsed, Stand: 10.01.2004
160 Vgl. [ApacheAxisDocumentation], Axis Architecture Guide, Stand: 10.01.2004

Java 2 Platform. Die Aufgabe des Entwicklers der einen SOAP Service mit *Axis* erstellt besteht darin, die Java Klasse mit der dienstspezifischen Funktionalität zu implementieren und das Zusammenspiel der Handler wie gewünscht zu konfigurieren[161].

3.4.2.2 Der Servlet-Container Tomcat

Der *Apache-Tomcat*-Servlet-Container in der Version 4 stellt die offizielle Refferenzimplementierung der *Java Servlet* 2.3 und *JavaServer Pages* (JSP) 1.2 Spezifikation von *Sun Microsystems* dar. Er ist vollständig in Java implementiert und damit für eine Vielzahl von Betriebssystemen verfügbar. Im Jahr 1999 begann die Entwicklung des Servlet-Containers mit der Kombination des *JServ*-Moduls, einer Servletengine für den Apache Web Server, und Ansätzen anderer Hersteller zu *Tomcat* in der Version 3. Der entstandene Code war allerdings so schwer zu warten, dass für die Version 4 unter dem Namen *Catalina* eine völlig neue Architektur entworfen wurde. Diese zeichnet sich durch leichte Erweiterbarkeit dank hoher Modularität aus. Die JSP-Engine mit dem Namen *Jasper* läuft beispielsweise ebenfalls als Servlet in *Tomcat 4*, kann aber auch unabhägig von ihm verwendet werden[162].

Tomcat wird in der *Axis* Dokumentation für den Einsatz als Container empfohlen[163]. Er unterstützt die Verwendung von HTTP über SSL und, spätestens durch die Anforderung der *Java Servlet 2.3* Spezifikation, auch die *HTTP Basic Authentication*[164]. Das bedeutet, dass er alle in Kapitel 3.3 definierten Bedingungen erfüllt und stellt damit die passende Umgebung für den Einsatz der Integrationsplattform dar.

3.4.2.3 Apache Xerces und Xalan

Mit *Xerces* stellt das *Apache XML Projekt* der Entwicklergemeinde einen leistungsfähigen XML Parser zur Verfügung. Neben der Java-Variante werden auch Implementierung für C++ und Perl angeboten. Die aktuelle

161 S. [ApacheAxisDocumentation], User's Guide, Stand: 10.01.2004
162 Vgl. [LampTomcat4], Stand: 11.01.2004
163 S. [ApacheAxisDocumentation], Installation Instructions, Stand: 10.01.2004
164 S. [SunServletSpec2.3], S.82

Version 2.6.0 der Java-Version unterstützt über die grundlegende XML Spezifikation hinaus auch wichtige Standards und APIs wie *XML Namespaces*, *DOM*, *SAX, Java APIs for XML Processing* (JAXP) und *XML Schema*[165].

Xalan ist die Realisierung eines *XSLT*-Prozessors, also eines Werkzeugs für die Transformationen von XML in andere XML oder nicht XML basierte Formate wie z.B. HTML. Für das hier vorgestellte Projekt ist *Xalan* vor allem wichtig wegen seiner Implementierung der *XML Path Language* (XPath)[166], einer Sprache zum direkten Adressieren von Unterknoten eines XML Dokumentes[167]. Die Verwendung von *XPath*-Ausdrücken vereinfacht den Zugriff auf Knoten tieferer Ebenen eines mit DOM erzeugten XML-Baumes erheblich, gegenüber der Möglichkeit, mit Hilfe der entsprechenden Methoden des DOM-Parsers durch den Baum zu navigieren. Neben der Java Implementierung von *Xalan* stellt das *Apache XML Projekt* auch eine Variante in C++ bereit.

3.4.2.4 Apache Log4J

Log4J ist eine interessante Ergänzung zu den vielen 'großen' Werkzeugen, die von der *Apache Foundation* angeboten werden. Es zielt auf eines der grundlegendsten Probleme bei der Softwareentwicklung ab: Das Protokollieren von Informationen zur Laufzeit zwecks Überwachung des Systems oder schlicht zur Fehlersuche. Mit *Log4J* ist es möglich, diesen Vorgang im laufenden System ein- und abzuschalten oder anzugleichen, ohne den Quellcode der Anwendung anpassen zu müssen. Es ist so aufgebaut, dass entsprechende Anweisungen im fertigen Code verbleiben können, ohne dass dadurch große Leistungseinbußen bei der Ausführung entstehen. Das Verhalten von *Log4J* beim Protokollieren kann vollständig über Konfigurationsdateien gesteuert werden[168].

---

165 Vgl. [ApacheXercesJava], Readme, Stand: 11.01.2004
166 Vgl. [ApacheXalanJava], Stand: 11.01.2004
167 S. [XMLPathLanguage1.0], Stand: 11.01.2004
168 Vgl. [ApacheLog4JDocs], Stand: 11.01.2004

3.4.3 IBM DB2 und JDBC

Das vom Hersteller IBM als objektrelationale Datenbank bezeichnete Produkt *DB2 Universal Database* ist eine der ältesten und damit am meisten ausgereiften relationalen Datenbanken am Markt. Ihre Entwicklungsgeschichte geht bis in die 1970er Jahre zurück[169]. Sie ist für eine ganze Reihe von Plattformen verfügbar: Windows 95/98/NT/2000, OS/2, Linux (inkl. Linux für z/OS), AIX, PTX, HP-UX, Sun Solaris, OS/400 und OS/390[170].

Die Auswahl des passenden JDBC Treibers für DB2 ist keine triviale Angelegenheit. Insbesondere für die in diesem Projekt verwendete OS/400-Variante der Datenbank, stehen mehrere Treiber zur Verfügung. Zunächst sind zwei Arten von JDBC Treibern für diese Variante zu unterscheiden. Der sogenannte „native"-Treiber wird von IBM mit seinen DB2 Produkten ausgeliefert. Es handelt sich dabei um einen Typ 2 Treiber der auf weitere Bibliotheken und Schnittstellen, die auf dem System installiert sein müssen, zurückgreift. Der alternative „Toolbox"-Treiber, ist Teil der *IBM Toolbox for Java*. Er ist so implementiert, dass er direkt Verbindung zum Datenbankserver aufnimmt und keine zusätzlichen Installationen auf dem Client benötigt (Typ 4).

IBM rät, je nach dem ob die Java Applikation auch auf der iSeries oder auf einer anderen Hardwareplattform läuft, zur Verwendung des *native*- oder des *Toolbox*-Treibers[171]. Hier wird der Toolbox-Treiber zum Einsatz kommen, u.a. auch weil dieser keine zeitaufwendigen Installationen und Konfigurationen notwendig macht[172].

4. Umsetzung des Lösungsansatzes

Im Folgenden soll nun Schritt für Schritt die Umsetzung des zuvor beschriebenen Lösungsansatzes dargestellt werden. Zunächst werden jene Elemente beschrieben, aus denen sich der SOAP Server zusammensetzt. Im

169 S. [Wikipedia], Eintrag: SQL, Stand: 11.01.2004
170 S. [IBMDB2UDB_de], Stand: 11.01.2004
171 Vgl. [IBMiSeriesJDBCdriverFAQs], Stand: 11.01.2004
172 Vgl. Kapitel 5.2.5: Verwendung des native-JDBC-Treibers für DB2

Anschluss daran werden die Schritte zum 'Einsatz' der entwickelten Komponenten, dem so genannten *Deployment*, dargestellt. Abschließend wird die Erörterung der Client-Seite folgen und das Zusammenspiel beider Seiten dargestellt.

4.1 Design der Serverkomponenten

Um den Aufbau der Serverkomponenten darstellen zu können, werden zunächst die hierfür entworfenen Grundelemente vorgestellt. Ziel der Integrationsplattform ist es, nach und nach mehrere Backend-Systeme integrieren zu können. Bei dem hier verwendeten Ansatz werden zunächst die für den Zugriff auf das Backend-System benötigten Methoden in einer oder mehreren Klassen des selben Pakets implementiert. Dieses Pakete stellt damit die Gesamtheit der zur Integration verfügbaren Schnittstellen dar. Der Aufbau dieser Schnittstellen muss derart sein, dass es möglich ist, diese auch unabhängig von dem hier zur Kommunikation verwendeten Web Service-Protokoll zu verwenden, beispielsweise im Rahmen einer client-seitigen Java Applikation mit grafischer Benutzeroberfläche. Demnach sollten sie als eine Komponente gestaltet sein, die über klar definierte Mechanismen kommuniziert und nicht direkt auf die von ihr benötigten Dienste, wie Datenbankverbindungen und Protokollfunktionen, zugreift. Ein direkter Zugriff würde konkrete Implementierungen dieser Dienste voraussetzen und damit eine lose Kopplung unmöglich machen.

Um diesen Anforderungen gerecht werden zu können, werden Java-Interfaces definiert, über die der Komponente die zur Erledigung ihrer Aufgaben benötigten Dienste und Daten zur Verfügung gestellt werden. Diese erforderlichen Interfaces nachfolgenden im einzelnen mit Namen und Beschreibung:

- Die Implementierung des **Request** Interface kapselt die Anfrage der aus dem Backend-System benötigten Daten. Die angegebenen Bedingungen können über die im Interface definierten Methoden ausgelesen werden.

- Im Gegenzug dient die Implementierung des **Response** Interface als Datencontainer für das Anfrageergebnis, welches über die definierten Methoden befüllt werden kann.
- Das **DBConnectionPool** Interface dient ausschließlich der Unterstützung bei der Anbindung von *JDBC*-Datenquellen und definiert nur eine einzige Methode. Diese muss jeweils eine gültige Datenbankverbindung liefern, welche von der jeweiligen Klasse, die Backend-Schnittstellen implementiert, verwendet werden kann.
- Das **Logger** Interface ermöglicht es, im Rahmen der Backend-Schnittstellen Informationen zu protokollieren, ohne sich an dieser Stelle um die Implementierung dieses Mechanismus kümmern zu müssen.

Die Verwendung der Integrations-Schnittstellen setzt die Implementierung der in den Interfaces definierten Methoden entsprechend der Rahmenbedingungen voraus, unter denen die Integration erfolgen soll. Das kann z.B. für das *DBConnectionPool* Interface bedeuten, dass es im Fall der Integration in einer client-seitigen Java Applikation so implementiert wird, dass immer ein und die selbe Datenbankverbindung zurückgegeben wird. Wird das Backend-System in eine Serverapplikation integriert, werden möglicherweise mehrere Datenbankverbindungen gleichzeitig benötigt. Dies erfordert wiederum eine angemessene Umsetzung.

```
public interface Request {

   /*
    * Methoden für die Abfrage der gewünschten Attribute
    */

   public int numberOfRequestedAttributes();
   public Enumeration getAllRequestedAttributesNames()
   public String getRequestedAttributeName( int index
   public boolean isRequestedAttribute( String name );


   /*
    * Methoden zur Abfrage der Attributwert-Bedingungen
    */

   public int numberOfConditions();
   public String getNameOfCondition( int index );
   public int numberOfConditionValues( int index );
   public int numberOfConditionValues( String nodeName );
   public Enumeration getConditionValues( int index );
   public Enumeration getConditionValues( String nodeName );
   public boolean isSetConditionForAttribute( String nodeName );


   /*
    *  Methoden zur Abfrage der Sortier-Bedingungen
    */

   public int numberOfSortOrderConditions ();
   public String getSortOrderAttributeName( int index );
   public String getSortOrderDirection( int index );


   /*
    * Methoden zur Abfrage der Ausschnitt-Bedingungen
    */

   public boolean isSetLimit();
   public int getLimitFrom() throws Exception;
   public int getLimitAmount() throws Exception;
}
```

Abbildung 8: Request Java-Interface

Der Definitionen der *Request* und *Response Interfaces* liegt ein einfaches Prinzip zugrunde. Es orientiert sich an den homogenen Datenstrukturen in Form von Listen, wie sie relationalen Datenbanken zugrunde liegen. Es lässt sich aber in leicht abgewandelter Form auch auf komplexere Strukturen anwenden. Es basiert auf dem Umstand, dass die aus einem Backend-System angeforderten Daten-Einheiten - im Folgenden in Anlehnung an die Datenbanknomenklatur als Entitäten bezeichnet - aus einer bestimmten Menge von Attributen bestehen. Diese sind jeweils mit einem eindeutigen

Namen versehen. Das Ergebnis einer Anfrage besteht aus einer bestimmten Menge von Entitäten, deren Attributwerte den in der Anfrage formulierten Bedingungen entsprechen. Diese Bedingungen können über die im *Request*-Interface definierten Methoden ausgelesen werden. Sie bestehen aus einem oder mehreren möglichen Werten, die ein namentlich benanntes Attribut einer Entität annehmen kann, welche zur Ergebnismenge gehören soll. Außerdem kann spezifiziert werden, ob alle oder nur ein Teil der Attribute einer Entität benötigt werden. Darüber hinaus definiert das *Request Interface* weitere Methoden, über welche geforderte Eigenschaften der Ergebnismenge abrufbar sind. Diese Eigenschaften betreffen sowohl die gewünschte Sortierung als auch die Möglichkeit, nur einen Ausschnitt aus dem Gesamtergebnis anzufordern. Einen Überblick über die verfügbaren Funktionen bietet Abbildung 8, welche die vollständige Definition des *Request Interface* zeigt. Entsprechend definiert das *Response Interface* eine Methode, um jeweils eine Entität an die Ergebnismenge anzufügen.

4.1.1 Implementierung der SOAP Services

Wie im dritten Kapitel dargelegt wurde, sollen die SOAP-Dienste in einer Form implementiert werden, in der die XML-Dokumente direkt von der Dienstklasse verarbeitet werden. Dies erfordert eine eindeutige Festlegung auf die Struktur der eingehenden und ausgehenden Dokumente. Neben der eigentlichen Implementierung des SOAP-Dienstes steht daher im Folgenden die Beschreibung der mittels SOAP transportierten XML-Dokumente im Vordergrund. Der Aufbau orientiert sich an dem eingangs erläuterten Prinzip der Anfragebeschreibung und dem Aufbau einer Ergebnismenge aus Entitäten, Attributen und Attributwerten. Das Prinzip wird nur auf entsprechende XML-Dokumente abgebildet.

4.1.1.1 Anfrage und Antwort XML-Dokumente

Das XML-Dokument in Abbildung 9 zeigt Bedingungen für die Abfrage von Kundenadressen im Warenwirtschaftssystem. Der dargestellte Aufbau wird aber auch für Abfrage anderer Informationen verwendet. Der Wurzelknoten

ist immer mit *request* bezeichnet. Auf der nächsten Ebene folgen die Knoten *nodes*, *conditions*, *sort_order* und *limit*, mit deren Hilfe die oben beschriebenen Bedingungen formuliert werden.

Über die Textinhalte der Knoten unterhalb des *nodes*-Knoten werden die in der Ergebnismenge gewünschten Attribute der Entitäten spezifiziert. Die tatsächlichen Bedingungen für die angeforderten Entitäten sind unter *conditions* zu finden. Jeder Knoten auf der Ebene unterhalb dieses Knotens besitzt ein Attribut, über welches angegeben wird, worauf sich die Bedingung bezieht. In der Regel handelt es sich bei dieser Angabe um den Namen eines existierenden Attributs einer Entität. In Ausnahmefällen kann es aber sinnvoll sein, Bedingungen zu formulieren, die sich nicht unmittelbar oder nicht ausschließlich auf ein einziges Attribut beziehen. So werden z.B. in dem abgebildeten Dokument die Konditionen für die Attibute *name1*, *name2* und *name3* unter der Bezeichnung *name* zusammengefasst. Dies muss von der Dienst-Implementierung entsprechend umgesetzt werden. Jeder *condition*-Knoten umfasst einen oder mehrere Werte, jene Werten die ein das entsprechende Attribut einer Entität annehmen kann, welche zur Ergebnismenge gehören soll.

```
<?xml version="1.0" encoding="UTF-8"?>
<request>
   <nodes>
      <node>customer_id</node>
      <node>name1</node>
      <node>name2</node>
      <node>name3</node>
      <node>street_no</node>
      <node>zip</node>
      <node>corporate_zip</node>
      <node>po_box</node>
      <node>po_zip</node>
      <node>city</node>
      <node>phone1</node>
      <node>phone2</node>
      <node>fax</node>
      <node>E-Mail</node>
      <node>spkz</node>
      <node>lkz</node>
      <node>customer_since</node>
      <node>invoice_receiver</node>
      <node>sub_id</node>
      <node>blocked_status</node>
   </nodes>
   <conditions>
      <condition node="customer_id">
         <value>81750</value>
         <value>81751</value>
      </condition>
      <condition node="name">
         <value>michel</value>
         <value>consulting</value>
      </condition>
      <condition node="street_no">
         <value>adenauer</value>
      </condition>
      <condition node="zip">
         <value>53498</value>
      </condition>
      <condition node="city">
         <value>remagen</value>
      </condition>
   </conditions>
   <sort_order>
      <node direction="asc">name1</node>
      <node direction="asc">name2</node>
      <node direction="asc">name3</node>
   </sort_order>
   <limit>
      <from>100</from>
      <amount>0</amount>
   </limit>
</request>
```

Abbildung 9: Ein XML Anfragedokument für Adressen aus dem Warenwirtschaftssystem

Auf die Definition der Bedingungen für die im Ergebnis enthaltenen Entitäten folgt in dem Musterbeispiel die Angabe der Sortierreihenfolge. Jeder Unterknoten von *sort_order* bezieht sich auf eines der unter *nodes* angeforderten Attribute der Entitäten. Über ihre Reihenfolge in dem Anfrage-Dokument wird die Priorität des Attributes für die Sortierreihenfolge gesteuert. Durch die *direction*-Angabe wird die Sortierrichtung spezifiziert. Das letzte Element der oberen Ebene in dem dargestellten Beispiel dient der Einschränkung des Ergebnisses. Dafür sind zwei Angaben notwendig: Welches das erste Element des Ausschnitts zu sein hat und wie viele der folgenden dazu gehören sollen. Die Angaben begrenzen den Ausschnitt auf alle folgenden (Wert 0) ab und einschließlich dem einhundertsten Ergebniselement.

Der Aufbau des Antwort XML-Dokuments ist hierneben leicht nachzuvollziehen. Wie Abbildung 10 zeigt, besteht es aus einem *response*-Knoten, der die Unterknoten *metadata* und *content* umfasst. Ersterer dient dem Transport von Informationen über das Anfrageergebnis, wie z.B. die tatsächliche Anzahl der Entitäten im Gesamtergebnis - für den Fall, dass nur ein Ausschnitt angefordert wurde. Der mit *content* bezeichnete Knoten enthält das angefragte Ergebnis. Es umfasst entsprechend viele Unterknoten, wie entweder als Ausschnitt angefordert wurde, oder den spezifizierten Bedingungen tatsächlich entsprechen. Diese Unterknoten sind alle mit einem bestimmten Begriff bezeichnet, der zu den angeforderten Daten passt. In dem Antwortdokument zu dem obigen Adressen-Beispiel wird dazu der englische Begriff *address* verwendet. In diesen Unterknoten werden die angeforderten Attribute transportiert, wobei jedes mit seinem eindeutigen Namen versehen ist. Die Knotennamen unterhalb von *content* sind also alle von dem in Anspruch genommenen Dienst abhängig.

```
<?xml version="1.0" encoding="UTF-8"?>
<response>
   <metadata>
      <total_records>100</total_records>
   </metadata>
   <content>
      <address>
         <customer_id>117123</customer_id>
         <name1>Dr.med.Schulz</name1>
         <name2></name2>
         <name3></name3>
         <street_no></street_no>
         <zip>53424</zip>
         <corporate_zip></corporate_zip>
         <po_box></po_box>
         <po_zip></po_zip>
         <city>Remagen</city>
         <phone1></phone1>
         <phone2></phone2>
         <fax></fax>
         <E-Mail></E-Mail>
         <spkz>de</spkz>
         <lkz>DE</lkz>
         <customer_since>1989-12-18</customer_since>
         <invoice_receiver>117532</invoice_receiver>
         <sub_id></sub_id>
         <blocked_status>0</blocked_status>
       </address>
       <address>
         <customer_id>145579</customer_id>
         <name1>Klärwerk Remagen</name1>
         <name2></name2>
         <name3></name3>
         <street_no></street_no>
         <zip>53424</zip>
         <corporate_zip></corporate_zip>
         <po_box></po_box>
         <po_zip></po_zip>
         <city>Remagen</city>
         <phone1></phone1>
         <phone2></phone2>
         <fax></fax>
         <E-Mail></E-Mail>
         <spkz>de</spkz>
         <lkz>DE</lkz>
         <customer_since>1994-03-04</customer_since>
         <invoice_receiver>145750</invoice_receiver>
         <sub_id></sub_id>
         <blocked_status>0</blocked_status>
      </address>
   </content>
</response>
```

Abbildung 10: Ein XML Antwortdokument für eine Adressanfrage

4.1.1.2 Implementierungen der Dienst-Klassen

Die Erstellung der Klassen, die mittels SOAP ihre Dienste zur Verfügung stellen, setzt die Implementierung der Interfaces, die zur Benutzung der Backend-Schnittstellen benötigt werden, voraus. Im Prinzip müssen die Methoden der Dienst-Klassen die eingehenden XML-Dokumente interpretieren und die darin befindlichen Bedingungen über die Implementierung der *Request Interface*-Methoden zur Verfügung stellen. Entsprechend muss das über das *Response Interface* gesetzte Ergebnis auf das Antwort-Dokument abgebildet werden. Außerdem müssen die Interfaces *DBConnectionPool* und *Logger* angemessen umgesetzt werden. Dies bedeutet für die hier verwendete Umgebung beispielsweise, dass sicherzustellen ist, dass eine bestimmte Datenbankverbindung zu einem bestimmten Zeitpunkt immer nur einer Instanz einer Dienst-Klasse zur Verfügung steht. Die Umsetzung erfolgt hier in Form der Klassen *LayerDBConnectionPool* und *LayerLogger*.

Schließlich müssen die Vorgaben von *Apache Axis* für die Implementierung von so genannten „Message Style Services" berücksichtigt werden. Dieser Typ wird verwendet wenn gewünscht ist, dass *Axis* in den Hintergrund tritt und der Dienst-Implementierung die XML-Verarbeitung überlässt. Alternativ würden die Dokumente auf Java Objekte abgebildet. Für die Implementierung von „Message Styled Services" bietet *Axis* vier Alternativen an[173]. Hier wird die Variante verwendet, bei der das Anfrage Dokument als Objekt vom Typ *org.w3c.dom.Document*[174] übergeben und das Antwortdokument als Objekt des gleichen Typs zurückgegeben wird. Das bedeutet es kann als XML-Dokument mit den Möglichkeiten des DOM-Parsers verarbeitet werden.

Die Interpretation des als *Document*-Objekt an die Methode übergebene Anfrage-Dokuments erfolgt mit Hilfe der Klasse *XMLRequest*, die, wie der Name schon vermuten lässt, im Rahmen der SOAP-Dienste auch das *Request Interface* implementiert. Die entsprechenden Methoden navigieren dazu durch das XML-Dokument und liefern die geforderten Informationen in

173 S. [ApacheAxisDocumentation], User's Guide, Stand: 18.01.2004
174 S. [ApacheXerces-jAPIDocs], /org/w3c/dom/Document.html,Stand: 18.01.2004

Form der im Interface definierten einfachen Datentypen[175]. Das *Document*-Objekt wird bei der Erzeugung einer Instanz von *XMLRequest* zusammen mit einer Instanz von *LayerLogger* an den Konstruktor der Klasse übergeben und beide stehen damit innerhalb der Methoden über Instanzvariablen zur Verfügung. Das so erzeugte *XMLRequest*-Objekt wird zusammen mit einer Instanz der *XMLResponse*-Klasse, welche das *Response Interface* implementiert, bei Aufruf einer Methode der Backend-Schnittstelle übergeben. Zur Veranschaulichung dieses Vorgangs zeigt Abbildung 11 die Implementierung der Klasse *AddressesService*, die mit der Methode *getAddresses* die Operation zur Abfrage von Adressen im Warenwirtschaftssystem bereitstellt.

```
package org.michel.integration.layer.soap;

import org.w3c.dom.*;

import org.michel.integration.toolbox.*;
import org.michel.integration.toolbox.xml.*;
import org.michel.integration.layer.toolbox.*;
import org.michel.integration.systems.ver.VerInterface;

public class AddressesService {

   public Document getAddresses(Document body)
      throws Exception {

      Logger logger = new LayerLogger();

      Request request = new XMLRequest( body, logger );
      XMLResponse response = new XMLResponse( "address", logger );

      LayerDBConnectionPool pool = new LayerDBConnectionPool();

      VerInterface ver = new VerInterface( pool, logger );
      ver.getAddresses( request, response );

      return response.getResponseDoc();
   }
}
```

Abbildung 11: Implementierung des Dienstes zur Abfrage von Adressen aus dem ERP-System

Bei der Erzeugung eines Objektes der Klasse *XMLResponse* wird dem Konstruktor als erster Parameter ein String übergeben. Dieser enthält die Bezeichnung für den Knoten, der im Antwort-Dokument die einzelnen

175 S. Abbildung 8, S.78

Entitäten repräsentiert[176]. Die Methode der Backend-Schnittstelle, welche die entsprechende Logik für den Zugriff auf die Adressdaten im Warenwirtschaftssystem implemetiert, gehört zur Klasse *VerInterface* und trägt den gleichen Namen wie die Implementierung des SOAP-Dienstes. Diese Klasse wird eingehend im nächsten Abschnitt beschrieben. Vorher soll noch dargelegt werden, wie innerhalb der Methoden der Klasse *XMLRequest* vorgegangen wird, um Daten aus dem Anfrage-Dokument zu erhalten.

Die Implementierung der Methode *numberOfRequestedAttributes* soll dazu als Beispiel dienen[177]. Ihre Aufgabe ist es, die Anzahl der angeforderten Attribute zu bestimmen. Um diese Aufgabe zu erfüllen, bedient sie sich einiger nicht öffentlich zugänglicher Methoden der eigenen Klasse, welche Funktionalitäten kapseln, die in mehreren Methoden benötigt werden. Diese privaten Methoden sollen nicht im einzelnen vorgestellt werden. Vielmehr wird das von ihnen verborgene Vorgehen erläutert.

Mit Hilfe der *XPath API*[178] ist es über entsprechende Ausdrücke möglich, auf einzelne oder mehrere Unterknoten eines XML-Dokuments, welches z.B. durch ein *Document*-Objekt repräsentiert wird, zuzugreifen. Die Implementierung dieser API erfolgt in *Apache Xalan* durch die Klasse *org.apache.xpath.XPathAPI*. Die Klassenmethode *selectNodeList* liefert bei Übergabe eines Objekts vom Typ *org.w3c.dom.Node* und einem entsprechenden XPath-Ausdruck ein Objekt vom Typ *org.w3c.dom.NodeList*. Das *Node* Objekt repräsentiert einen beliebigen Knoten eines XML-Dokumentes und der XPath-Ausdruck kann wiederum mehrere Unterknoten adressieren. Die *NodeList* umfasst alle Unterknoten des übergebenen Knotens, auf die der Ausdruck passt. Konkret bedeutet dies, wird der Methode eine Referenz auf den Knoten *nodes*[179] des Anfragedokumentes in Kombination mit dem einfachen XPath-Ausdruck '*node*' übergeben, liefert diese eine Liste mit allen unmittelbaren Unterknoten von *nodes* mit der Bezeichnung *node*. Die Beispiel-Methode *numberOfRequestedAttributes* muss daraufhin nur noch die Methode

176 Vgl. Abbildung 10, S.83
177 Vgl. Abbildung 8, S.78
178 Vgl. Kapitel 3.4.2.3, Apache Xerces und Xalan
179 Vgl. Abbildung 9, S.81

getLength des Ergebnis-Objektes vom Typ *NodeList* aufrufen, um die Anzahl zu ermitteln[180]. Entsprechend lässt sich in anderen Methoden verfahren, die z.B. die Frage beantworten, ob ein Limit spezifiziert wurde, die Namen der angeforderten Attribute zurückliefern oder die Anzahl der für ein bestimmtes Attribut gesetzten Bedingungswerte.

Die Implementierung des *Response Interface XMLResponse*, welche nur die Methode *setLine* zum Anhängen einer Entität an das Ergebnis umfasst, kommt ohne zusätzliche Werkzeuge wie *XPath* aus. Noch im Konstruktor wird ein neues *Document*-Objekt erzeugt, welches das XML-Ergebnis-dokument repräsentiert. Da es bei den hier verwendeten Daten nur um Zeilen von Datenbanktabellen geht, können diese als Java-Hashtable an die Methode *setLine* übergeben werden. Für die Übergabe komplexere Entitäten kann das *Response Interface* später um zusätzliche Methoden erweitert werden. Anhand der Schlüssel-Wert-Paare der *Hashtable* und unter Zuhilfenahme der notwendigen Methoden des *Document*-Objekts werden dann die entsprechenden XML-Knoten im Antwort-Dokument erzeugt. Über den Aufruf der Methode *getResponseDoc* des *XMLResponse*-Objekts wird das Ergebnisdokument in der SOAP-Dienst-Methode dann schließlich ausgelesen[181].

4.1.2 Implementierung der ERP-Schnittstelle

In dem hier zugrunde gelegten Modell ist es Aufgabe der Backend-Schnittstellen, die Bedingungen, die über das *Request Interface* abgefragt werden können, auf die API des Backend-Systems abzubilden. Entsprechend müssen die Anfrageergebnisse über die Methoden des *Response Interface* weitergegeben werden.

In dem hier geschilderten konkreten Fall erfordert dies die Erzeugung von SQL-Kommandos zur Abfrage der Daten aus der dem Warenwirtschafts-system zugrunde liegenden *DB2* Datenbank. Dazu wurde zunächst ein Ansatz favorisiert, bei dem die Erzeugung der SQL-Befehle entsprechenden

180 Vgl. [ApacheXerces-jAPIDocs], Stand: 30.01.2004
181 S. Abbildung 11, S.85

Werkzeugen überlassen werden sollte. Es handelt sich bei dem zu integrierenden Warenwirtschaftssystem allerdings um ein über die Jahre gewachsenes Produkt mit entsprechenden Strukturen in der darunter liegenden Datenbank. So ist es bei der Abfrage eines Produktes beispielsweise keinesfalls selbstverständlich, dass die Produktbezeichnung immer aus ein und der selben Tabelle kommt. Vielmehr kann sich das in Abhängigkeit davon, in welcher Sprache die Produktbezeichnung ausgegeben werden soll, unterscheiden. Dies bedeutet aber nicht zwangsläufig, dass die Bezeichnungen zu verschiedenen Sprachen immer in verschiedenen Tabellen eingetragen sind. Der notwendige Umgang mit den Eigenheiten dieses über die Jahre gereiften Systems ließ die Verwendung von verfügbaren Werkzeugen schließlich nicht mehr sinnvoll erscheinen. Es soll an dieser Stelle nicht weiter auf die Tabellenstrukturen der DB2 Datenbank eingegangen werden, da dies den Umfang dieser Arbeit deutlich übersteigen würde. Vielmehr soll nun ein Werkzeug vorgestellt werden, welches im Rahmen des praktischen Teils dieser Arbeit entwickelt worden ist, um das Erzeugen der SQL-Kommandos zumindest zu vereinfachen.

Die Klasse *DBConnector* basiert ebenfalls vollständig auf den in Abschnitt 4.1 vorgestellten Beschreibungen von Entitäten, Attributen, Bedingungen für Attributwerte, Sortierungen und Ausschnitten aus dem Ergebnis. Zunächst einmal wird dabei davon ausgegangen, dass die Menge von Attributen aus denen sich eine Ergebnis-Entität zusammensetzt, aus einer oder mehreren Tabellen stammt. Die Klasse stellt Methoden zur Verfügung, über die jedem Attribut ein eindeutiger Name gegeben und das Attribut einer Spalte in einer Tabelle zugeordnet werden kann. Darüber hinaus können spezielle Bedingungen zum Verbinden der benötigten Tabellen (engl. *join*) und die außergewöhnliche Behandlung einzelner Spalten, z.B. unter Verwendung von SQL-Funktionen, definiert werden.

Aufbauend auf diesen Definitionen verfügt die Klasse über Methoden zur Generierung der Elemente eines *select*-Statements, dem Befehl zum Abfragen von Daten aus der Datenbank. Dieser besteht in seiner üblichen Form immer aus den folgenden Elementen: Aufzählung der Spalten, die zum

Ergebnis gehören sollen, Aufzählung der Tabellen, aus denen die Spalten stammen, Bedingungen für das Verbinden der Tabellen, Bedingungen, die auf alle Ergebnisdatensätze zutreffen sollen und Sortierung des Ergebnisses. Mit Hilfe der *DBConnector*-Klasse und den Informationen, die über das *Request Interface* zur Verfügung gestellt werden, können die erforderlichen SQL-Statements teilweise oder vollständig generiert werden. In einigen Fällen passt der durch die *DBConnector*-Klasse verallgemeinerte Aufbau allerdings nicht hundertprozentig. Dann lässt sie aber die Möglichkeit offen, dass zusätzlicher Code entwickelt wird, der Teile der SQL-Kommandos generiert, die sich mit den über die Hilfsklasse generierten Anteilen kombinieren lassen.

Die erzeugten SQL-Kommandos werden dann durch die Datenbank ausgeführt und das Ergebnis zeilenweise als Java-Hashtable an die Methode *setLine* der das *Response Interface* implementierenden Klasse übergeben. Zur Unterstützung dieses Prozesses besitzt die Hilfsklasse ebenfalls eine Methode. Für die Kommunikation mit der Datenbank wird eine Datenbankverbindung benötigt. Diese wird von dem Objekt bereitgestellt, welches das *DBConnectionPool*-Interface implementiert und vorher über den Konstruktor übergeben wurde. Darüber hinaus können Informationen über die Implementierung des *Logger*-Interface protokolliert werden[182].

4.1.3 Zusammenspiel der Serverkomponenten

Der Aufbau der Dienste *ProductsService* und *DiscoutsService* entspricht dem der dargestellten *AddressesService*-Klasse. Erstere verfügt dabei nur über die Methode *getProducts*, letztere über die drei Methoden *getDiscountsPerOrder*, *getDiscountsPerPosition* und *getStandardDiscounts*, was den verschiedenen Rabattarten im angebundenen Warenwirtschafts-system entspricht. Die Implementierungen der Methoden sind identisch mit *getAddresses* der *AddressesService*-Klasse, abgesehen davon, dass sie jeweils verschiedene Methoden der Backend-Schnittstelle aufrufen und sich in den angebotenen Attributen und der Bezeichnung des Entity-Knotens im

182 Vgl. Abbildung 11, S.85

XML-Ergebnisdokument unterscheiden[183]. Die eigentliche Spezialisierung dieser Dienste liegt in der Implementierung der entsprechenden Backend-Schnittstellen.

Interessant in diesem Zusammenhang ist noch die Tatsache, dass die Methoden nicht alle auftretenden Java-Exeptions selbst behandeln. Vielmehr werden die nicht behandelten Ausnahmen von *Axis* aufgefangen und per SOAP-Fehlernachrichten an den Client weitergegeben[184].

4.2 Server Deployment

Der Begriff *Deployment* bedeutet im Englischen so viel wie Aufmarsch, Aufstellung, Einsatz, Entsendung oder Stationierung[185]. Im Zusammenhang mit IT-Systemen wird er entsprechend verwendet, wenn es darum geht, diese für ihre Aufgaben vorzubereiten, also im wesentlichen Software zu installieren und zu konfigurieren. Im Folgenden soll unter dieser Überschrift erläutert werden, welche Schritte notwendig sind, um die zuvor beschriebenen Integrationskomponenten ihrem Zweck zuzuführen.

Dass es zur Verwendung der in diesem Rahmen entwickelten Java Klassen notwendig ist, diese mit einem entsprechenden Compiler zu übersetzen, wird als gegeben vorausgesetzt. Auf die Einzelheiten dieses Vorgangs, wie das Setzen des sogenannten CLASSPATH, unter dem die verwendeten Bibliotheken zu finden sind etc., soll daher hier nicht weiter eingegangen werden. Vielmehr wird es um die Einrichtung von *Tomcat* und *Axis*, also Installation und Konfiguration, gehen.

183 Vgl. Abbildung 10, S.83 und Abbildung 11, S.85
184 Vgl. Kapitel 2.2.3.3: SOAP
185 S. [LeoWörterbuch], Eintrag engl. deployment, Stand: 25.01.2004

4.2.1 Tomcat Installation und Konfiguration

Für die hier vorgestellte Lösung findet die aktuelle Version von Tomcat 4.1.x Verwendung[186]. Voraussetzung dafür ist, dass ein *Java Development Kit* (JDK) Version 1.2 oder höher auf dem System installiert ist[187]. Die Installation von *Tomcat* ist einfach: Nach dem Herunterladen der Binär-Distribution[188], also der bereits kompilierten Variante, muss das Paket einfach in ein beliebiges Verzeichnis im Filesystem ausgepackt werden. Dieses Verzeichnis wird im Folgenden der Einfachheit halber mit dem Alias *TOMCAT* bezeichnet. Anschließend kann der Server dann über ein Skript im Verzeichnis *TOMCAT/bin* gestartet werden und sollte dann auf Port 8080 über HTTP ansprechbar sein.

Wie im vorangegangenen Kapitel bereits ausführlich dargelegt, werden für die Implementierung und den Betrieb der Integrationsplattform einige zusätzliche Werkzeuge verwendet, die im Web Container *Tomcat* zur Verfügung gestellt werden müssen. Neben Apache *Axis*, mit dessen Einrichtung sich der nächste Abschnitt befassen wird, handelt es sich dabei um die folgenden Komponenten: Den XML-Parser *Apache Xerces 2 Java*[189], XSLT und XPath Werkzeuge aus *Apache Xalan 2 Java*[190] und *JTOpen*, die Open-Source Variante der *IBM Toolbox for Java*, insbesondere die darin enthaltenen JDBC-Treiber[191].

Die Verzeichnisstruktur von *Tomcat* umfasst eine Reihe von Verzeichnissen, um zusätzliche Bibliotheken und Komponenten zu installieren. Bei den APIs *Xerces* und *Xalan* handelt es sich um Bibliotheken, die per Definition bereits Teil des *Java 2 Platform* sind, die aber von Dritten, in diesem Falle der *Apache Foundation*, gepflegt und weiterentwickelt werden. Für solche APIs definiert *Java 2* den sogenannten *Endorsed Standards Override Mechanism*, der es auf komfortable Art und Weise erlaubt, ältere Bibliotheken durch

186 Vgl. Kapitel 3.4.2.2: Der Servlet-Container Tomcat
187 S. [ApacheJakartaProject], /tomcat/tomcat-4.1-doc/RUNNING.txt, Stand: 25.01.2004
188 S. [ApacheJakartaProject], /site/binindex.cgi, Stand: 25.01.2004
189 S. [ApacheXercesJava]
190 S. [ApacheXalanJava]
191 S. [IBMToolboxJTOpen]

aktuellere Versionen zu ersetzen[192]. Dies bedeutet für die Installation der aktuellen Versionen dieser APIs in *Tomcat*, dass sie in ein entsprechendes Verzeichnis, getrennt von anderen zusätzlichen APIs, installiert werden müssen. Konkret bedeutet es, dass die Dateien *xercesImpl.jar*, *xml-apis.jar* und *xmlParserAPIs.jar* aus der *Xerces*-Distribution, sowie *xalan.jar* als Hauptbibliothek von *Xalan* in das Verzeichnis *TOMCAT/common/endorsed* kopiert werden.

Im Falle der mit *JTOpen* gelieferten JDBC-Treiber ist es sinnvoll diese, anstatt in das entsprechende Verzeichnis der *Axis*-Applikation, in das Verzeichnis für Bibliotheken zu kopieren, die für alle in *Tomcat* installierten Anwendungen zur Verfügung stehen. Daher wird das Paket *jt400.jar* in das Verzeichnis *TOMCAT/common/lib* kopiert. In diesem Paket sind spezielle Typ 4 JDBC-Treiber für den Zugriff auf die OS/400 Variante der DB2 Datenbank enthalten[193].

4.2.2 Deployment von Apache Axis in Tomcat

Neben den zuvor beschriebenen Werkzeugen muss das Deployment der SOAP Engine in den Web Container vorgenommen werden. *Axis* wird hier als Servlet-Applikation betrieben[194]. Für die Verwendung müssen zunächst einige Implementierungen der J2EE-Standard-APIs zur XML-Verarbeitung[195] als Bibliotheken verfügbar gemacht werden. Dazu werden *jaxrpc.jar*, *wsdl4j.jar* und *saaj.jar* aus dem *Axis*-Paket nach *TOMCAT/common/lib* kopiert. Anschließend wird das Verzeichnis *axis* aus dem Unterverzeichnis *webapps* des Pakets nach *TOMCAT/webapps* verschoben. Dieses enthält die gesamte *Axis*-Servlet-Applikation.

Die gesamte Konfiguration von *Tomcat* und den installierten Webapplikationen erfolgt über XML-Dateien, auch weil die Servlet-Spezifikation dies vorschreibt. Nach der Installation der *Axis*-Servlet-Applikation in das *webapps*-Verzeichnis von *Tomcat* ist es vorteilhaft, diese auch in die

192 Vgl. [Java2Endorsed], Stand: 25.01.2004
193 Vgl. [IBMToolboxJTOpen], /overview.htm, Stand: 25.01.2004
194 Vgl. Kapitel 3.4.2.1: Apache AXIS
195 Vgl. Kapitel 2.2.4.3: JNDI, JDBC und andere APIs

zentrale Konfigurationsdatei des Servers einzutragen und zu konfigurieren. Übergeht man diesen Schritt, wird *Axis* mit Standardeinstellungen beim Starten von *Tomcat* in Gang gebracht. Die zentralen Konfigurationsdateien des *Tomcat*-Servers liegen im Verzeichnis *TOMCAT/conf*. Im Mittelpunkt des Interesses steht hier die Datei *server.xml*. In ihr werden alle Einstellungen für *Tomcat* vorgenommen. Einen passenden Eintrag für *Axis* zeigt Abbildung 12. Dieser muss unterhalb des XML-Knotens *Host* vorgenommen werden. Damit wird ein eigenes Logfile für die SOAP Engine definiert und darüber hinaus macht es die Angabe *reloadable="true"* möglich, zur Laufzeit Änderungen an den in *Axis* installierten Service-Klassen vorzunehmen, ohne den Server neu starten zu müssen. Dies ist insbesondere in der Entwicklungs- und Testphase interessant.

```
<Context path="/axis" docBase="axis" debug="1" reloadable="true">
   <Logger className="org.apache.catalina.logger.FileLogger"
      prefix="localhost_axis_log." suffix=".txt"
      timestamp="true"/>
</Context>
```

Abbildung 12: Axis-Konfiguration in server.xml

Für das Deployment der Service-Klassen empfiehlt die *Axis*-Dokumentation die Verwendung des Werkzeugs *AdminClient*, welches zum Umfang des Pakets gehört[196]. Zunächst muss dazu allerdings das *AdminServlet* in Gang gebracht werden, das aus Sicherheitsgründen per Voreinstellung deaktiviert ist. Zur allgemeinen Konfiguration einer Servlet-Applikation dient die Datei *web.xml* im Unterverzeichnis *WEB*-INF des Applikationsverzeichnisses. Hierin muss schlicht der vorhandene *servlet-mapping* Eintrag zu dem *AdminServlet* von Kommentarzeichen befreit werden.

196 S. [ApacheAxisDocumentation], User's Guide, Stand: 25.01.2004

Nach diesem Schritt kann *Tomcat* gestartet bzw. neu gestartet werden. Ob das Deployment von *Axis* bis hierhin erfolgreich war, kann mit dem Aufruf von *http://hostname:8080/axis/happyaxis.jsp* getestet werden. Als Ergebnis sollte eine Webseite zurückgeliefert werden, die darüber ausführlich Auskunft gibt.

Für das weitere Deployment der Service-Klassen mit Hilfe des *AdminClient* wird ein *Web Service Deployment Descriptor* (WSDD) benötigt. Dieser beschreibt, welche Service-Klasse entsprechende Dienste zur Verfügung stellt. Ein Beispiel dafür zeigt Abbildung 13. Darin wurde für jede der drei Dienste *AddressesService*, *DiscoutsService* und *ProductsService* ein *service*-Eintrag vorgenommen, der die den Dienst implementierende Klasse inklusive ihres Pakets definiert. Außerdem ist angegeben, welche Methoden, in der Web Service-Sprache dann als Operationen bezeichnet, die Klassen über SOAP zur Verfügung stellen. Es ist in diesem Zusammenhang insbesondere darauf zu achten, dass die Dienste als „Message Style Services" definiert sind[197].

```
<deployment name="test" xmlns="http://xml.apache.org/axis/wsdd/"
     xmlns:java="http://xml.apache.org/axis/wsdd/providers/java"
     xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance">
   <service name="AddressesService" provider="java:MSG"
     style="message" use="literal">
      <parameter name="allowedMethods" value="getAddresses"/>
      <parameter name="className"
         value="org.michel.integration.layer.soap.AddressesService"/>
   </service>
   <service name="DiscountsService" provider="java:MSG"
     style="message" use="literal">
      <parameter name="allowedMethods" value="getDiscountsPerOrder,
         getDiscountsPerPosition, getStandardDiscounts"/>
      <parameter name="className"
         value="org.michel.integration.layer.soap.DiscountsService"/>
   </service>
   <service name="ProductsService" provider="java:MSG"
     style="message" use="literal">
      <parameter name="allowedMethods" value="getProducts"/>
      <parameter name="className"
         value="org.michel.integration.layer.soap.ProductsService"/>
   </service>
</deployment>
```

Abbildung 13: Beispiel für eine WSDD-Datei

197 Vgl. Kapitel 4.1.1.2: Implementierungen der Dienst-Klassen

Im Folgenden wird davon ausgegangen, dass diese Datei unter dem Namen *deploy.wsdd* vorliegt. Bei dem zuvor beschriebenen *AdminClient* handelt es sich um eine Java-Applikation, die von der Kommandozeile aus zu starten ist. Dafür sind zunächst einige Einstellungen notwendig, wie das Setzen der Pfade zu den benötigten Bibliotheken und natürlich den Implementationen der Service- und Hilfsklassen. Abbildung 14 verdeutlicht die dazu notwendigen Schritte unter dem Betriebssystem Windows. Unter einem UNIX Derivat ist die Syntax allerdings nur geringfügig anders. Es wird in dem Beispiel davon ausgegangen, dass die Kommandos in einem Verzeichnis ausgeführt werden, in dem sich auch die Datei *deploy.wsdd* befindet und welches ein Unterverzeichnis *bin/* mit den übersetzten Service-Klassen besitzt.

```
set AXIS_LIB="C:\Programme\Java\Tomcat 4.1\webapps\axis\WEB-INF\lib"
set TOMCAT_LIB="C:\Programme\Java\Tomcat 4.1\common\endorsed"
set PROJECT_PATH=.
set AXISCLASSPATH=%PROJECT_PATH%\bin;%AXIS_LIB%\axis.jar
set AXISCLASSPATH=%AXISCLASSPATH%\%AXIS_LIB%\commons-discovery.jar
set AXISCLASSPATH=%AXISCLASSPATH%\%AXIS_LIB%\commons-logging.jar
set AXISCLASSPATH=%AXISCLASSPATH%\AXIS_LIB%\jaxrpc.jar
set AXISCLASSPATH=%AXISCLASSPATH%\%AXIS_LIB%\saaj.jar
set AXISCLASSPATH=%AXISCLASSPATH%\%AXIS_LIB%\log4j-1.2.8.jar
set AXISCLASSPATH=%AXISCLASSPATH%\%TOMCAT_LIB%\xml-apis.jar
set AXISCLASSPATH=%AXISCLASSPATH%\TOMCAT_LIB%\xercesImpl.jar

java -cp %AXISCLASSPATH% org.apache.axis.client.AdminClient
-lhttp://localhost:8080/axis/services/AdminService -umanager -wmanager
deploy.wsdd
```

Abbildung 14: Deployment der Service-Klassen mit dem AdminClient

Der *AdminClient* setzt sich daraufhin mit dem *AdminServlet* unter der angegebenen URL in Verbindung und übergibt den Deployment Desriptor. Bei erfolgreicher Ausführung sollte sich der Prozess mit der folgenden Ausgabe zurückmelden: 'Processing file deploy.wsdd <Admin>Done processing</Admin>'.

Daraufhin befindet sich im Verzeichnis *TOMCAT\webapps\axis\WEB-INF* eine neue Datei mit dem Namen *server-config.wsdd*, die weit mehr als den Inhalt von *deploy.wsdd* enthält. Für zukünftige Änderungen ist es allerdings

einfacher, diese direkt in der generierten Datei durchzuführen. Zu guter Letzt müssen noch die kompilierten Serviceklassen *AddressesService*, *DiscountsService* und *ProductsService* inkl. aller Hilfsklassen in das Verzeichnis *TOMCAT\webapps\axis\WEB-INF\classes* kopiert werden.

4.2.3 Konfiguration der Sicherheitsmechanismen

Wie bereits zuvor erläutert, werden für die benötigten Sicherheitsmechanismen im wesentlichen zwei Techniken miteinander kombiniert[198]: die Verschlüsselung des *HTTP* mittels *SSL* und die *HTTP Basic Access Authentication*. Wie diese Mechanismen in dem hier verwendeten Kontext eingerichtet werden, soll in den folgenden beiden Abschnitten erörtert werden.

4.2.3.1 Konfiguration der SSL Unterstützung in Tomcat

Die Einrichtung der SSL Unterstützung in *Tomcat* setzt die Installation der *Java Secure Socket Extension* (JSSE)[199] voraus. Seit der Version 1.4 ist diese allerdings Teil des JDK und musste im vorliegenden Fall daher nicht explizit installiert werden.

Für die Verwendung von SSL muss zunächst ein sogenanntes Zertifikat erstellt werden, d.h. eine Art Ausweis, mit dem der Server seine Identität gegenüber dem Client beurkundet. Üblicherweise müssen diese Zertifikate für die öffentliche Nutzung von entsprechenden Zertifizierungsstellen vergeben und bestätigt werden[200]. In diesem Falle handelt es sich aber um eine firmeninterne Nutzung, die keine Bestätigung der Identität durch eine dritte Institution erfordert. Daher kann das benötigte Zertifikat einfach selbstständig mit den entsprechenden Werkzeugen generiert werden.

Die verwendeten Zertifikate auf einem Server werden in einem entsprechenden Behälter aufbewahrt, dem sogenannten *keystore*. Das ist eine Datei, die ein definiertes Format aufweisen muss. *Tomcat* 4 unterstützt

198 Vgl. Kapitel 3.3: Darstellung des Lösungsansatzes
199 S. [JavaSun2003], Java Secure Socket Extension, /products/jsse/, Stand 28.01.2003
200 S. [TrustedShopsSSLZertifikate], Stand: 28.01.2004

dabei ausschließlich die Verwendung des *Java KeyStore* Formats, also dem Format, welches von den JSSE-Werkzeugen erzeugt wird.

Die Erzeugung eines entsprechenden Zertifikats erfolgt mit dem Shell Programm *keytool*, welches sich im *bin/* Verzeichnis des JDK befindet. Ein Beispiel für den Aufruf dieses Werkzeuges in einer Windows Umgebung zeigt Abbildung 15. Zur Erzeugung des Zertifikats müssen dann verschiedene Angaben gemacht werden, z.B. die Vergabe eines Passworts, Firmenname, Name der zuständigen Kontaktperson etc. Diese Informationen können bei der Verbindung über HTTPS vom Client beim Server abgerufen werden. Sollte man sich bei dem verwendeten Passwort nicht für das von *Tomcat* verwendete Standardpasswort entscheiden, muss es später in der *Tomcat*-Konfiguration angegeben werden. Wird, wie in dem Beispiel, kein Pfad zum *keystore* spezifiziert, wird dieser unter dem Namen *.keystore* im Home-Verzeichnis des ausführenden Benutzers angelegt.

```
%JAVA_HOME%\bin\keytool -genkey -alias tomcat -keyalg RSA
```

Abbildung 15: Erzeugung eines SSL-Zertifikats mit JSSE

Im Anschluß wird ein Port in der *server.xml* von *Tomcat* für den Zugriff über HTTPS konfiguriert. Dies erfolgt durch die Definition eines entsprechenden *Connectors* wie es Abbildung 16 zeigt. Im dem *Factory* Knoten müssen ggf. die Angaben *keystoreFile* und *keystorePass* ergänzt werden, wenn hierfür nicht die Standardwerte verwendet wurden. Nach einem Neustart des Containers sollte dieser unter dem angegebene Port - in dem Beispiel ist das 8443 - per HTTPS zu erreichen sein[201].

201 Vgl. [ApacheJakartaProject], /tomcat/tomcat-4.1-doc/ssl-howto.html, Stand: 29.01.2004

```
<-- Define an SSL HTTP/1.1 Connector on port 8443 -->
<Connector
     className="org.apache.catalina.connector.http.HttpConnector"
           port="8443" minProcessors="5" maxProcessors="75"
           enableLookups="true"
           acceptCount="10" debug="0" scheme="https" secure="true">
  <Factory className="org.apache.catalina.net.SSLServerSocketFactory"
           clientAuth="false" protocol="TLS"/>
</Connector>
```

Abbildung 16: Definition eine SSL-Connectors für Tomcat

4.2.3.2 Sicherung gegen unerlaubten Zugriff

Zur *Identifikation* und der Bestimmung der *Authentizität*[202], also der Abfrage des Benutzernamen und Passworts, soll *HTTP Basic Access Authentication* zum Einsatz kommen. *Tomcat* verwendet zur applikationsübergreifenden Rechteverwaltung das Konzept von Rollen und Benutzern. Zunächst wird eine Rolle mit dem Zugriff auf eine Webapplikation oder einen bestimmten Bereich verbunden. Einem User können dann eine odere mehrere Rollen zugeordnet werden. Demnach sollte zu Beginn eine Rolle für den Zugriff auf die *Axis*-Web Services definiert und ein Beispieluser angelegt werden, der dieser Rolle zugeordnet wird. Die Definition von Rollen und Usern erfolgt üblicherweise in der Datei *TOMCAT/conf/tomcat-users.xml*. Abbildung 17 zeigt die beiden Einträge die in dieser Datei vorzunehmen sind.

```
...
<role rolename="webservices"/>
...
<user username="wsuser" password="wsxxx123" roles="webservices"/>
...
```

Abbildung 17: Definition einer Rolle und eines Benutzers in Tomcat

Darüber hinaus muss eine entsprechende Sicherheitsbedingung (engl. security constraint) in die Konfigurationsdatei der Webapplikation eingefügt werden. In Abbildung 18 ist dargestellt, wie die notwendigen Elemente aussehen. Sie sind als Unterknoten von *web-app* einzutragen. Der Knoten

202 Vgl. Kapitel 2.1.5: Sicherheitsanforderungen

security-constraint umfasst dabei einen oder mehrere Knoten vom Typ *web-resource-collection*. Darin ist es möglich, wie hier über *url-pattern*, Teile der Webapplikation zu definieren, für die die Sicherheitsbedingung greift. Außerdem wird unterhalb von *security-constraint* die zuständige Rolle assoziiert. Mit Hilfe von *login-config* wird darüber hinaus die Authentifizierungsmethode festgelegt[203], in diesem Falle natürlich *BASIC*. Diese Angaben haben zur Folge, dass beispielsweise die Startseite der *Axis*-Engine von jedem aufgerufen werden kann. Die Verwendung der eigentlichen Dienste erfolgt aber über das *Axis*-Servlet, welches wiederum über den Alias *services* angesprochen wird. Für entsprechende Zugriffe greift aber die definierte Sicherheitsbedingung, die zur Angabe von Username und Passwort verpflichtet.

```
<security-constraint>
   <web-resource-collection>
      <web-resource-name>Axis Web Services </web-resource-name>
      <url-pattern>/services/*</url-pattern>
   </web-resource-collection>
   <auth-constraint>
      <role-name>webervices</role-name>
   </auth-constraint>
</security-constraint>
<login-config>
   <auth-method>BASIC</auth-method>
   <realm-name>Axis Web Services</realm-name>
</login-config>
```

Abbildung 18: Security constraint für die Verwendung von HTTP Basic Access Authentication

4.3 Verwendung der Dienste in PHP

Die in Kapitel 3.4.1 vorgestellten Werkzeuge welche benötigt werden, um die Anbindung der PHP Applikation über SOAP basierte Web Services an die Integrationslösung zu realisieren, sind im wesentlichen bereits Teil des Funktionsumfangs von PHP. Nur das SOAP Toolkit *NuSOAP* muss hier zusätzlich installiert werden. Dies erfolgt, indem es einfach in eines der Verzeichnisse kopiert wird, in denen PHP nach zusätzlichen Bibliotheken sucht. *NuSOAP* ist ein Paket aus mehreren Klassen, die den Programmierer

203 Vgl. [SunServletSpec2.3], S.248

bei der Verwendung von SOAP basierten Web Services unterstützen. Aus dem Paket wird in dem hier vorgestellten Projekt ausschließlich die Klasse *soap_transport_http* verwendet. Mit ihrer Hilfe ist es möglich, den SOAP *Envelope* mit *Header* und *Body* zu erzeugen und diesen über HTTP oder HTTPS zu versenden. Im Gegenzug erhält man den XML-Inhalt der Antwortnachricht. Darüber hinaus unterstützt sie die *HTTP Basic Access Authentication*. Alle weiteren Klassen aus dem *NuSOAP* Paket finden hier keine Verwendung, da ihre Aufgaben sich hauptsächlich auf den Bereich der Nutzung von RPC-Mechanismen über SOAP beziehen.

Ein weiteres wichtiges Werkzeug zur Anbindung der PHP basierten CRM-Applikation an die SOAP-Dienste ist der SAX-Parser. Mit seiner Hilfe werden die XML-Antwortdokumente in entsprechend verschachtelte PHP-Arrays überführt. Dazu ist zu anzumerken, dass Arrays in PHP beliebig ineinander verschachtelt werden können. Ein Array-Element kann also einen einzelnen Wert darstellen oder wiederum auf einen Array verweisen. Außerdem unterscheidet PHP nicht zwischen Arrays, die numerische Werte zur Indexierung ihrer Elemente verwenden und den sogenannten assoziativen Arrays, die dieses über Zeichenketten erledigen.

Auf Basis dieser beiden Werkzeuge erfolgt die Implementierung von Komponenten, welche die Tatsache, dass es sich bei den verwendeten Datenquellen um SOAP-Dienste handelt, für die übrige PHP-Anwendung vollkommen transparent werden lassen.

4.3.1 Komponenten zur Anbindung der SOAP-Dienste

Für die transparente Anbindung der SOAP-Dienste innerhalb der CRM-Applikation sind die folgenden drei Komponenten verantwortlich:

- Der **SoapMessenger** dient ausschließlich zur vereinfachten Nutzung der Klasse *soap_transport_http*. Die für die Kommunikation mit den SOAP-Diensten notwendigen Methodenaufrufe dieser *NuSOAP*-Klasse werden in den beiden Methoden *sendMessage* und *getResponse* gekapselt. Für die Nutzung sind die folgenden drei Informationen erforderlich: Die URL des

SOAP Services, der Name der aufzurufenden Operation und das zu versendende XML-Anfragedokument. Außerdem erfüllt diese Klasse zusätzliche Aufgaben, die in *NuSOAP* nicht implementiert sind, z.B. die Behandlung eines HTTP Protokollfehlers. Sie erfüllt also ausschließlich Aufgaben, für die man sie auch in jedem anderen Projekt verwenden könnte.

- Die **ERPMessager**-Klasse steht der hier konkret zu lösenden Aufgabe schon deutlich näher. Sie besitzt ebenfalls nur zwei Methoden. Innerhalb der *call*-Methode wird ein verschachtelter PHP Array in ein XML-Anfragedokument der Form überführt, wie sie Abbildung 10 zeigt. Anschließend wird das generierte Dokument mit Hilfe des *SoapMessengers* verschickt und das Antwortdokument unter Verwendung des *ERP Sax Parser* ausgewertet. Der *content*-Knoten Inhalt des XML-Ergebnisses wird von der Methode als Array zurückgegeben. Der *metadata* Teil kann über die zweite Methode dieser Klasse *getMetadata* ebenfalls als Array abgerufen werden.
- Der **ERP Sax Parser** ist nicht als Klasse, sondern als Funktionsbibliothek implementiert. Der Grund dafür liegt in der Umsetzung des SAX-API in PHP. Wie bereits erläutert, basiert SAX auf Ereignissen bei der seriellen Verarbeitung eines XML-Dokuments[204]. Für solche Ereignisse, wie z.B. das Auftreten eines öffnenden Tags, können beim Parser Funktionen zur Behandlung registriert werden. PHP erwartet zur Registrierung den Funktionsnamen als Zeichenkette. Dies setzt voraus, dass die entsprechende Funktion global verfügbar und nicht in einer Klasse gekapselt ist. Die für die entsprechenden Ereignisse registrierten Funktionen des *ERP Sax Parsers* sind so aufgebaut, dass sie auf die immer vorhandenen Knoten *response*, *metadata* und *content* eines Ergebnisdokuments reagieren und die Unterknoten in eine entsprechende Array-Struktur überführen.

204 Vgl. Kapitel 3.3: Darstellung des Lösungsansatzes

Im folgenden Abschnitt soll hierauf aufbauend die Nutzung der beschriebenen Komponenten zur Durchführung einer Anfrage dargestellt werden.

4.3.2 Ablauf der Anfrageverarbeitung

Die Implementierung der benötigten Logik zur Anbindung der Backend-Systeme erfolgt, wie Abbildung 7 bereits andeutete, im Rahmen der CRM-Applikation durch Funktionen bzw. Methoden. Diese kapseln vollständig die von der Darstellung getrennte Anwendungslogik. Zum jetzigen Zeitpunkt wird die Anwendung aus zwei Backend-Systemen gespeist, einer MySQL-Datenbank, die direkt über entsprechende PHP-Funktionen angesprochen wird, und dem über die Integrationsplattform angebundenen ERP-System. Für die Präsentationsschicht bleibt es völlig transparent, ob eine von ihr aufgerufene Methode die benötigten Daten aus der MySQL-Datenbank oder über SOAP von der Integrationsplattform einholt. Sie übergibt die benötigten Parameter und erhält als Ergebnis einen verschachtelten Array.

Während die Funktionen, die zur Datenbeschaffung aus der MySQL-Datenbank zuständig sind, diese Funktionsparameter auf entsprechende SQL-Kommandos abbilden müssen, haben ihre Pendants, die für das ERP-System zuständig sind, diese nur ggf. in das richtige Format zu bringen und die *call*-Methode eines *ERPMessager*-Objekts aufzurufen. Daraus wird von dieser dann das XML-Dokument generiert und dieses mit Hilfe des *SOAPMessengers* verschickt. Das Antwortdokument wird mit der *ERP Sax Parser* Bibliothek interpretiert und das Ergebnis als Array an die aufrufende Funktion zurückgegeben. Diese reicht ihn an die Präsentationsschicht weiter.

Da PHP keinen nativen Mechanismus zur Fehlerbehandlung besitzt, implementiert die CRM-Applikation hierzu einen eigenen, der sicherstellt, dass aufgetretene Fehler dem Benutzer in einheitlicher Darstellung mitgeteilt werden. Im Falle eines Fehlers bei der Übertragung der SOAP-Nachrichten greift die Funktion auf diesen Mechanismus zurück, um das Problem zu kommunizieren.

5. Fazit

Die Einbindung der Web Services in die PHP-Anwendung verlief im großen und ganzen erfreulich unproblematisch. Vor allem überzeugte die technische Einfachheit der Lösung auf Seiten des Clients. Mit wenigen allgemein verfügbaren und kostenlosen Werkzeugen ist die Integration an eine Plattform gelungen, die beliebig weiter ausgebaut werden kann. Ihre Einsatzmöglichkeiten sind dabei nahezu unbegrenzt. Da die notwendigen Werkzeuge auf gängigen Standards beruhen, stehen sie für nahezu jede Programmiersprache zur Verfügung, was eine zukünftige Nutzung weiter begünstigt.

Mit dem vorgestellten Ansatz werden alle Ziele erreicht, die zu Beginn formuliert wurden. Es wurde die Basis für eine Integrationsplattform geschaffen, die in zukünftigen Projekten wiederverwendet und erweitert werden kann. Ihre Architektur ermöglicht die modulare Erweiterung zur Integration zusätzlicher Backend-Systeme. Die bisher definierten Dienste zeigen dabei, in welche Richtung es weitergehen könnte. Dienste werden anhand der von ihnen bereitgestellten Informationen und nicht nach Datenquellen gegliedert. Außerdem besteht die Möglichkeit, die Integrationsplattform um zur Zeit noch nicht unterstützte Eigenschaften zu erweitern. So war beispielsweise die Unterstützung von Transaktionsmechanismen in dieser Ausbaustufe weder gefordert noch war sie für die Implementierung der ausschließlich lesenden Zugriffe zwingend erforderlich. Sowohl das SOAP-Protokoll als auch die verwendete Plattform und die angewandten Techniken stellen Möglichkeiten für die spätere Erweiterung in diese Richtung bereit.

Ob sich der entwickelte Ansatz in der Praxis bewähren wird, muss an dieser Stelle offen bleiben. Wie auch der folgende Abschnitt unterstreicht, sind die Aussichten aber vielversprechend und er kann aufgrund der bisher gesammelten Erfahrungen bereits als Erfolg gewertet werden.

5.1 Verhalten im Testbetrieb

Der vorgestellte Integrationsansatz hat sich im Testbetrieb als durchaus stabile Lösung präsentiert. Im Dauerbetrieb über mehrere Wochen gab es keinerlei Ausfälle oder unverhofft aufgetretene Probleme. In stichprobenartigen Tests wurden die Abfragegeschwindigkeiten über die Integrationsplattform mit den Laufzeiten verglichen, die eine direkte Abarbeitung der generierten Kommandos über ODBC benötigt. Dabei kam man zu dem Ergebnis, dass die Anfrage über SOAP und JDBC nur unwesentlich langsamer ist als die über ODBJ. Allerdings erfolgten im Tests sowohl die JDBC- als auch die ODBC-Verbindungen zu der dem ERP-Systems zugrundeliegenden DB2 Datenbank über ein *Virtual Private Network* (VPN) quer über das Internet. Die Anfragegeschwindigkeiten schwankten dabei erheblich zwischen einigen zehnteln bis hin zu mehreren Sekunden, was ein klares Indiz dafür ist, dass der größte Flaschenhals in der Testumgebung an anderer Stelle zu suchen ist als bei der Generierung und Auswertung der XML-Dokumente. Was die Performance betrifft, besteht also noch Optimierungsbedarf. Für den Produktionsbetrieb ist allerdings bereits vorgesehen, die Integrationsplattform im selben physikalischen Netzwerk zu betreiben wie die integrierte Datenbank, was eine erhebliche Zunahme der Geschwindigkeit nach sich ziehen dürfte. Insbesondere, da sich in den Tests gezeigt hat, dass bei einer Anfrage der Datenverkehr zwischen Datenbank und Integrationsplattform etwa zehnmal größer ist als der Umfang der Ergebnisdaten. Der Overhead bei der Nutzung von SOAP über HTTPS dürfte deutlich darunter liegen.

Zu weiteren Einbußen könnte aber auch die nicht gerade auf Geschwindigkeit optimierte Datenbankstruktur des ERP-Systems führen. Es bleibt daher abzuwarten, wie sich das System insgesamt unter Last im Produktionsbetrieb verhalten wird.

5.2 Verworfene Ansätze

Der Weg zu einem gesteckten Ziel ist nicht immer so geradlinig, wie es die Darstellung in einem Dokument wie diesem erfordert. Das bereits ausführlicher diskutierte Thema EJB und die gescheiterte direkte Einbindung von Java-Klassen in PHP waren dabei nur zwei aus einer ganzen Liste von Ansätzen, die im Laufe dieser Arbeit aufgegriffen und wieder verworfen wurden. Um einen entsprechenden Eindruck zu vermitteln, sollen in diesem Abschnitt noch einige dieser verworfenen Ansätze dargestellt werden.

5.2.1 Torque als API für das ERP-System

Wie bereits in Kapitel 3.2.2 erläutert wurde, stellt die Abbildung der Tabellenstrukturen auf Klassen und Objekthierarchien eine wesentliche Herausforderung bei der Verwendung von relationalen Datenbanken im objektorientierten Kontext dar. Man kann sich diesem Problem allerdings auch mit entsprechenden Werkzeugen annähern, die dem Entwickler viel Arbeit ersparen. Die *Apache Foundation* hält auch zu diesem Zweck mehrere Lösungen bereit.

Zunächst war für dieses Projekt anvisiert, die benötigten SQL-Kommandos nicht mit selbstentwickelten Werkzeugen zu produzieren, sondern *Apache Torque* für diese Aufgabe zu verwenden. *Torque* ist in der Lage, anhand von in XML definierten Tabellen-Strukturen sowohl diese in der Datenbank zu erzeugen als auch entsprechende Klassen für den Zugriff darauf zu generieren. Dabei bildet jede generierte Klasse eine Tabelle und ein Objekt dieser Klasse einen Datensatz aus dieser Tabelle ab. Auf die einzelnen Attribute kann über *get*- und *set*-Methoden nach dem Vorbild einer JavaBean zugegriffen werden. *Foreign-Key*-Beziehungen der Tabellen werden über Assoziationen zwischen den Klassen abgebildet[205].

Der sinnvolle Einsatz dieses Werkzeugs setzt allerdings eine weitgehend akademisch einwandfreie Struktur der Tabellen in der Datenbank voraus, wie sie bei dem hier zu integrierenden, über die Jahre gewachsenen

205 S. [ApacheTorque]

System, nicht vorliegt. Die unter Last zu erwartenden Laufzeiten der generierten SQL-Kommandos bleiben abzuwarten. Sollten sich diese wider Erwarten in eine nicht tragbare Richtung entwickeln, ist es bei dem hier vorgestellten Ansatz durchaus möglich, zu optimieren. Bei der Verwendung von *Torque* wäre eine Einflussnahme zumindest nur mit deutlich größerem Aufwand und unter Umgehung der Vorteile des Werkzeugs möglich.

5.2.2 XML-RPC Lösung

Die Verwendung von SOAP als Web Service-Plattform wurde nicht von Anfang an favorisiert. Vielmehr erschien zu Beginn eine XML-RPC Lösung angemessener. Ein Paket für die Verwendung von XML-RPC in Java wird ebenfalls von *Apache* angeboten[206]. Schon bald zeigte sich aber, dass die enge Orientierung an klassischen Funktionsaufrufen für das Vorhaben hier eher unpraktikabel sein würde. XML-RPC definiert eine Reihe von Datentypen und Möglichkeiten, komplexere Strukturen aufzubauen[207]. Der erste Schritt bei der Verwendung in Java bestand nun darin, eine Möglichkeit zu entwickeln, um alle JDBC-Datentypen auf eine viel geringere Zahl von XML-RPC Typen abzubilden. Ein Abfrageergebniss wurde dann mitsamt der Informationen über seine Datentypen zum PHP Client befördert, einem Konsumenten also, der sich für Datentypen ganz und gar nicht interessiert. Bei komplexeren Typen galt es wiederum, sich an den relativ engen Vorgaben von XML-RPC zu orientieren, was unter Umständen bedeutet, vorhandene und auch nachvollziehbare Strukturen aufzugeben.

Letztlich überzeugte SOAP mit der Möglichkeit, sich bei der Gestaltung eines Dienstes einfach auf die Definition der eingehenden und ausgehenden Nachrichten zu konzentrieren. Die Herausforderung bei der Verwendung liegt jetzt in der Erzeugung eines XML-Dokuments mit der richtigen Struktur und nicht im wesentlichen darin, Datentypen und -strukturen innerhalb der verwendeten Programmiersprachen sauber aufeinander abzubilden. Dies besitzt insbesondere bei der Fehlersuche enorme Vorteile.

---

206 S. [ApacheXML-RPC]
207 Vgl. Kapitel 2.2.3.2: XML-RPC

Wenn man auf der Suche nach einer schnellen und unkomplizierten Möglichkeit ist, vergleichsweise einfache Methodenaufrufe mit weitgehend primitiven Datentypen plattformunabhängig über ein Netzwerk abzuwickeln, ist XML-RPC sicherlich das Werkzeug der Wahl. Für komplexere Vorhaben ist es nach den hier gesammelten Erfahrungen allerdings eher ungeeignet.

5.2.3 XSLT in PHP

Letzten Endes erfolgt in der CRM-Applikation bei der Verarbeitung des Ergebnisses einer ERP-Anfrage nichts anderes, als dass ein XML-Dokument mit Hilfe eines Templates in HTML überführt wird. Allerdings über den Umweg, als Array in PHP verarbeitet zu werden. Stellt sich die Frage: Warum das Ganze, wenn im XML-Umfeld ausgefeilte Werkzeuge zur Erledigung dieser Transformationsaufgabe existieren? Wie bereits in Kapitel 3.4.2.3 angesprochen, handelt es sich bei XSLT um ein solches Werkzeug, welches auch für PHP verfügbar ist[208]. So wurde zu Beginn der Entwicklung der CRM-Applikation durchaus einige Zeit lang die Auffassung vertreten, dass man auch die Daten aus anderen Datenquellen, z.B. der MySQL-Datenbank, in XML überführen und ausschließlich XSLT verwenden sollte. Allerdings wurde bald klar, dass ein solcher Ansatz nur vertretbar wäre, wenn er konsequent verfolgt würde. Dies würde wiederum bedeuten, dass die gesamte Anwendung auf dieser Basis entwickelt werden müsste, da viele Elemente sonst redundant implementiert würden.

Neben der offenen Frage, wie es mit der Performance einer solchen Lösung bestellt sein dürfte, mußte aber festgestellt werden, dass die Einführung dieser Technik keineswegs eine triviale Aufgabe darstellt. Ein in PHP eingespieltes Entwicklerteam auf eine derartig neue Technik einzuschwören, die gegenüber klassisch strukturierter Programmierung eine ganz andere Denkweise erfordert, ist ein durchaus zeitaufwendiges Vorhaben. So wurde schließlich wieder von diesem eleganten Ansatz abgerückt und, um ein einheitliches Vorgehen zu gewährleisten, wurde die XML-Dokumente der Integrationsplattform in PHP-Arrays überführt.

208 Vgl. [Vaswani2002], S.105

5.2.4 JDBC Connection Pooling

Die Verwaltung von mehreren offenen Datenbankverbindungen in sogenannten Pools ist eine Eigenschaft, die jeder J2EE Server unterstützen sollte. Natürlich bietet auch *Tomcat* eine entsprechende Lösung an[209]. Die notwendigen Verbindungsinformationen wie Benutzername, Passwort, URL etc. werden in den Konfigurationsdateien hinterlegt. Die Datenbankverbindungen werden dann von Tomcat aufgebaut und können über den Java-Verzeichnisdienst JNDI aus der Webanwendung heraus angefordert werden.

Leider wurden die Verbindungen im hier vorliegenden Fall regelmäßig nach einer gewissen Zeit der Nichtbenutzung von der Datenbank beendet. Diese Tatsache scheint der Implementierung des Connection-Pools allerdings zu entgehen, was dazu führt, dass die Integrationsplattform versucht, ihre Anfragen über bereits geschlossene Datenbankverbindungen durchzuführen. Das Problem ließ sich dadurch umgehen, dass die Verbindung bei Bedarf explizit vom der Servlet-Anwendung hergestellt und anschließend wieder beendet wird. Für die Zukunft sind allerdings weitere Nachforschungen an dieser Stelle geplant.

5.2.5 Verwendung des native-JDBC-Treibers für DB2

Neben der nunmehr hier verwendeten Typ 4 Ausführung des JDBC-Treibers, stehen, wie oben bereits ausgeführt, auch andere Varianten zur Verfügung[210]. So wurde die Arbeit zunächst unter Verwendung des sogenannten „native"-Treibers von IBM begonnen[211], einem Typ 2 Treiber, der neben der Installation der DB2 Client Software unter dem entsprechenden Betriebssystem auch umfangreiche Konfigurationen erfordert. So muss jede Datenbank, zu der eine Verbindung aufgebaut werden soll, zunächst in einen sogenannten Katalog eingetragen werden. Dieser Eintrag muss, je nach Wirtsbetriebssystem der Datenbank, mit einer Vielzahl unter-

209 S. [ApacheJakartaProject], /tomcat/tomcat-4.1-doc/jndi-datasource-examples-howto.html, Stand:05.02.2004
210 Vgl. Kapitel 2.2.4.3: JNDI, JDBC und andere APIs
211 Vgl. Kapitel 3.4.3: IBM DB2 und JDBC

schiedlicher Parameter versehen werden. Nun gestaltete sich die Installation unter Windows noch relativ einfach, da hier eine ganze Palette von grafischen und menügeführten Werkzeugen zur Einrichtung der Verbindungen zur Verfügung stehen. Bei einer späteren Portierung des Projektes nach Linux sah es aber leider ganz anders aus. Hier bestand die Schwierigkeit nun darin, die vielen Einstellungen, die das grafische Werkzeug unter Windows in dem Katalog vorgenommen hatte, mit kommandozeilenbasierten Clients durchzuführen. Eine Aufgabe, an der, wie sich herausstellte, auch DB2 erfahrene Entwickler und Administratoren scheitern können.

Die Verwendung des „Toolbox"-Treibers gestaltet sich daneben, unabhängig vom Betriebssystem, denkbar einfach. Wie man es als Entwickler einfacher PHP/MySQL Datenbankanwendungen gewohnt ist, genügt hier die Angabe von URL, Username und Passwort, um schließlich zu dem selben Ergebnis zu kommen, wie nach unmfangreichen Installations- und Konfigurationsarbeiten bei der alternativen Variante.

5.3 Ausblick

Zu Beginn des ersten Kapitels wurde ausgeführt, dass die zentrale Aufgabe der Informationstechnologie in Unternehmen darin besteht, Geschäftsprozesse integrativ, also über die Grenzen einzelner Organisationseinheiten hinweg, zu unterstützen. Ein nicht unwesentlicher Aspekt bei der Betrachtung dieses Sachverhalts besteht in der Tatsache, dass es immer wichtiger wird, Prozesse nicht nur effizient unterstützen, sondern Abläufe flexibel auf sich ständig verändernde Bedingungen anpassen zu können. Dieser stetige Prozess wird als *Business Process Management* (BPM) bezeichnet[212]. Eine effiziente IT-Infrastruktur zeichnet sich also nicht nur durch eine effektive Unterstützung der vorhandenen Geschäftsprozesse aus, sondern auch durch die Möglichkeit, die notwendigen Anpassungen mit möglichst geringem Aufwand umsetzen zu können.

212 Vgl. [JohannDerProzess]

Die Verwendung eines einheitlichen Mechanismus zur Kommunikation mit den unterstützenden Systemen ist dabei mit Sicherheit der Schritt in die richtige Richtung. Eine Überlegung, der mit der Verwendung von Web Services optimal Sorge getragen wird. Auf dieser Grundlage baut die *Business Process Execution Language for Web Services* (BPEL4WS) auf und definiert einen Standard für die formale Beschreibung der übergeordneten Prozesse. Mit Hilfe dieser Workflow-Sprache lassen sich Prozesse in XML definieren und festlegen, welche Web Services zur Durchführung verwendet werden sollen. Darüber hinaus lassen sich weitere Bedingungen, z.B. für den zeitlichen Ablauf, formulieren. Mit BPEL4WS kann der Aufbau der Nachrichten, die mit einem Dienst ausgetauscht werden können, hinterlegt und so die notwendigen Dokumente zum Aufruf generiert und Antworten entsprechend ausgewertet werden. Dazu werden bekannte Hilfsmittel wie XPath und WSDL verwendet[213].

In dem Zusammenhang lässt sich eine mögliche Schwäche der hier vorgestellten Lösung aufzeigen. Die Verwendung der *HTTP Basic Access Authentication* ist einfach und erfüllt ihren Zweck. Allerdings führt ihre Verwendung zwangsläufig dazu, dass wesentliche Merkmale für die Verwendung der Dienste, nämlich Benutzername und Passwort, getrennt von der eigentlichen Anfrage transportiert werden. Aus der Tatsache, dass diese Daten nicht Teil des XML-Dokuments sind, ergibt sich eine enge Bindung an das darunterliegende HTTP-Protokoll. Eine Situation, die man bei der Definition des SOAP-Standards eigentlich vermeiden wollte, in dem man die Wahl des Transportprotokolls freigestellt hat. Die Verwendung einer Workflow-Sprache wie BPEL4WS würde aber wohl vorraussetzen, dass alle notwendigen Informationen für den Aufruf eines Web Services in XML zu formulieren sind. Für den hier vorgestellten Lösungsansatz ist das Konzept aber sicherlich akzeptabel.

Einige Hersteller von EAI- und Workflow-Werkzeugen zeichnen eine goldene Zukunft, in der es einem Manager möglich sein wird, über eine grafische Oberfläche seine Geschäftsprozesse und die damit verbundenen Backend-

213 Vgl. [FischerBPEL4WS]

Systeme in immer neuen Varianten zu kombinieren. Die Idee, die sich dahinter verbirgt, basiert auf den hier vorgestellten Techniken. Sprachen wie die BPEL4WS zielen auf die Definition von Geschäftsprozessen und die in diesem Rahmen möglichst flexible Zusammenstellung der benötigten Dienste ab. Ob dies allerdings jemals so beliebig und spontan möglich sein wird, wie die Marketingabteilungen einiger Softwareunternehmen sich dies heute vorstellen, darf in Zweifel gezogen werden. Die Erfahrungen in diesem Projekt haben ein weiteres Mal gezeigt, das die Interaktion von Softwaresystemen auch unter der Verwendung von Web Services ein komplexes Thema bleibt, bei dem die Herausforderungen häufig im Detail liegen. So ist die Einführung von zusätzlichen Ebene zur Abstraktion eines konkreten Problems zwar in der Regel ein sinnvoller Schritt hin zu einer größeren Übersichtlichkeit und weg von einer Vielzahl von Speziallösungen. Letztlich kommt man bei der Frage nach dem Laufzeitverhalten eines Systems aber doch nicht umhin, sich mit den einzelnen Komponenten und ihren speziellen Anforderungen zu beschäftigen.

6. Literaturverzeichnis

[ActiveStateWsTutorial] ActiveState Corporation: Web Services Tutorial, http://talks.php.net/show/soap-phpcon-ny2003/

[Apache.org_About] Apache Software Foundation: About Apache, http://httpd.apache.org/ABOUT_APACHE.html

[ApacheAxisDocumentation] Apache Web Services Projekt: Axis Documentation, http://ws.apache.org/axis/java/index.html

[ApacheJakartaProject] The Apache Jakarta Project, http://jakarta.apache.org

[ApacheLog4JDocs] Apache Logging Services: Log4J Projekt, http://logging.apache.org/log4j/docs/index.html

[ApacheWikiAxisProjectPage] Apache Wiki: Axis Project Pages, http://nagoya.apache.org/wiki/apachewiki.cgi?AxisProjectPages

[ApacheTorque] Apache Torque: Datenbank Persistenz Mechanismus von Apache, http://db.apache.org/torque/

[ApacheXalanJava] The Apache XML Project: Xalan-Java version 2.5.2, http://xml.apache.org/xalan-j/index.html

[ApacheXerces-jAPIDocs] The Apache XML Project: Xerces2 Java Parser Full API documentation, http://xml.apache.org/xerces-j/apiDocs

[ApacheXercesJava] The Apache XML Project: Xerces2 Java Parser Readme, http://xml.apache.org/xerces2-j/

[ApacheXML-RPC] Apache XML-RPC, http://ws.apache.org/xmlrpc/

[Benoist2002] Benoist, Sylvain: Security with Apache SOAP, http://www.soapuser.com/sb_02jul02.html

[ComKomponentenDe2003] com-komponenten.de: Die Info-Website von Holger Schwichtenberg, Infos: COM/DCOM/COM+, http://www.com-komponenten.de/, Stand: 04.12.2003

[CybizCRM2-6/2000] Cybiz Magazin: CRM? -> Nein danke! (Ausgabe 6/2000)

[DenningerPeters2000] Denninger, Stefan und Peters, Ingo: Enterprise JavaBeans – Schneller Einstieg in die Konzepte von EJB, München: Addison-Wesley Verlag, 2000

[ExpatSourceforgeNet] The Expat XML Parser, http://expat.sourceforge.net/

[Fischer1996] Fischer, Bernhard: Entwurf und Implementierung eines Objektmodells für das Management von TP-Monitoren - Diplomarbeit, München: Technische Universität München, 1996

[FischerBPEL4WS] Fischer, Jörg: Gut beschrieben – BPEL4WS: Geschäftsprozesse mit Web Services, iX Magazin, Ausgabe 02/2004, Hannover: Heise Zeitschriften Verlag GmbH & Co. KG

[GeschichteDesInternet.com] Geschichte-des-Internet.com, http://www.geschichte-des-internet.com/, Stand: 20.12.2003

[GruhnThiel2000] Gruhn, Volker und Thiel, Andreas: Komponentenmodelle – DCOM, JavaBeans, Enterprise JavaBeans, CORBA, München: Verlag Addison-Wesley Verlag, 2000

[Hage1996] Hage, Sven: Eine Kurzüberblick über ODBC – Vortrag, Dresden: Hochschule für Technik und Wirtschaft, http://wwwbs.informatik.htw-dresden.de/svortrag/ai93/hage/odbc.html, Stand: 29.11.2003

[Hartwig1998] Hartwig, Jens: Einführung in JDBC, FHTW Berlin, http://web.f4.fhtw-berlin.de/hartwig/JDBC/jdbc.html

[HeiseNews20031118hps] News vom Heise Verlag: Microsoft öffnet Office-Formate vom 18.11.2003, http://www.heise.de/newsticker/data/hps-18.11.03-000/

[HobbesInternetTimeline] Zakon, Robert: Hobbes' Internet Timeline v6.1, http://www.zakon.org/robert/internet/timeline/

[IBM_PHP_DB2_2001] Scott, Dan: Connecting PHP Applications to IBM DB2 Universal Database, http://www-106.ibm.com/developerworks/db2/library/techarticle/scott/0614_scott.html, IBM 2001

[IBMDB2UDB_de] IBM Deutschland: DB2 Universal Database, http://www-5.ibm.com/de/software/data/sw/db2udb.html

[IBMiSeriesJDBCdriverFAQs] IBM Technical resources: Java - iSeries Native JDBC Driver – FAQS, http://www-1.ibm.com/servers/enable/site/java/jdbc/jdbcfaq.html

[IBMToolboxJTOpen] IBM: Toolbox for Java & JTOpen, http://www-1.ibm.com/servers/eserver/iseries/toolbox

[ITManagement11_2001] IT Management Magazin, Ausgabe 11/2001, Höhenkirchen: IT Verlag für Informationstechnik, 2001

[Java2Endorsed] Sun Microsystems: Java Endorsed Standards Override Mechanism, http://java.sun.com/j2se/1.4.2/docs/guide/standards/index.html

[JavaInselOpenbook] Ullenboom, Christian: Java ist auch eine Insel - Programmieren für die Java 2-Plattform in der Version 1.4 (2. Aufl.), Openbook: Verlag Galileo Computing, http://www.galileocomputing.de/openbook/javainsel2/index.htm

[JavaSun2003] Sun Microsystems: java.sun.com - The Source for Java Technology, http://java.sun.com/

[JohannDerProzess] Johann, Michael: Der Prozess – Grundlagen zu Business Process Management und Enterprise Application Integration, Java Magazin, Ausgabe 04/2003, Frankfurt am Main: Software & Support Verlag GmbH

[Kaib2002] Kaib, Michael: Enterprise Application Integration – Grundlagen, Integrationsprodukte, Anwendungsbeisiele, Magdeburg/Wiesbaden: Deutscher Universitäts-Verlag, 2002

[Kefk.net_Apache] Kefk Network: Apache, http://kefk.net/Open_Source/OS/Apache/index.asp

[Keller2002] Keller, Wolfgang: Enterprise Application Integration – Erfahrungen aus der Praxis, Heidelberg: dpunkt.verlag, 2002

[Knuth2002] Knuth, Michael: Web Services – Einführung und Übersicht, Frankfurt: Software & Support Verlag, 2002

[LampTomcat4] Lamp, Volker: Apache Jakarta Tomcat 4 - Architektur und Funktionsweise, http://www.fh-wedel.de/~si/seminare/ws02/Ausarbeitung/a.tomcat/tomcat1.htm

[LeoWörterbuch] Link Everything Online: Deutsch - Englisches Wörterbuch, http://dict.leo.org/

[Lexitron2003] Lexitron – Das Fachlexikon der IT-Begriffe, http://www.lexitron.de/

[Meyer2003] Matthias Meyer: CRM- Systeme mit EAI. Konzeption, Implementierung und Evaluation, Vieweg Verlag 2002

[MicrosoftDesign4Web] Microsoft Developer Network: Designing for Web or Desktop?: http://msdn.microsoft.com/netframework/using/building/web/default.aspx?pull=/library/en-us/dndotnet/html/designwebdesk.asp#designwebdesk_topic2

[Mozilla.org2003] mozilla.org: Webseite der Gruppe die sich um die Weiterentwickung des Mozilla-Projektes bemüht: http://www.mozilla.org (Stand 10.10.2003)

[Münz2001] Münz, Stefan: SelfHTML 8.0, http://selfhtml.teamone.de/, Stand: 27.10.2001

[NetscapeTechbriefsSSL] Netscape Network, Tech Briefs: Secure Sockets Layer, http://wp.netscape.com/security/techbriefs/ssl.html

[Nußdorfer2000] Nußdorfer, Richard: Das EAI-Buch (online), http://d4p.csaconsult.de/WWWROOTD4/eai-v1/buch/buch.html, Stand: 30.06.2003

[OASI_UDDI_3.0.1] Organization for the Advancement of Structured Information Standards (OASIS): UDDI Version 3.0.1 - UDDI Spec Technical Committee Specification, http://uddi.org/pubs/uddi-v3.0.1-20031014.htm, Stand: 14.10.2003

[PEARmanual] PEAR - PHP Extension and Application Repository: PEAR Manual, http://pear.php.net/manual/en/

[PHP4BugDatabase] PHP 4 Bug Database, http://bugs.php.net/

[PHP4UTutorial] PHP4U: Objektorientierte Programmierung mit PHP 4, http://www.php4u.net/index.php?main=tut2

[PHPBuilderNuSOAP] PHPBuilder: Building Web Services Using NuSOAP Toolkit, http://www.phpbuilder.com/columns/kramberger20031226.php3

[PHPmanual] PHP.net: PHP Handbuch, http://de.php.net/manual/de/

[Rapp2001] Reinhold Rapp: Customer Relationship Management, Das neue Konzept zur Revolutionierung der Kundenbeziehungen, Campus Verlag 2001

[RFC2617] Network Working Group: Request for Comments 2617 - HTTP Authentication: Basic and Digest Access Authentication, http://www.faqs.org/rfcs/rfc2617.html

[SAPINFO61] SAPINFO Magazin: Harmonisches zusammenwirken, Interview mit SAP-Vorstandsmitglied Peter Zencke zur Komponenten-Integration am 03.02.2000, Online Ausgabe: http://www.sap.info/public/de/print.php4/article/comvArticle-193353c63ad7488385/de

[SpiegelNetzRuhe2003] Spiegel Online, Netzwelt: Ruhe in Frieden, Netscape, http://www.spiegel.de/netzwelt/netzkultur/0,1518,257331,00.html

[Starke2002] Gernot Starke: Effektive Software-Architekturen, Carl Hanser Verlag 2002

[SunEJBSpec2.0] Sun Microsystems: Enterprise Java Beans (TM) Specification – Version 2.0, Palo Alto (CA, USA): Sun Microsystems, 2001

[SunServletSpec2.3] Sun Microsystems: Java Servlet Specification – Version 2.3, Palo Alto (CA, USA): Sun Microsystems, 2001

[TrustedShopsSSLZertifikate] TRUSTED SHOPS: Was ist ein SSL-Zertifikat?, http://www.trustedshops.de/de/help/help_de_263.html

[Universe_PHPExtension] Universe PHP Extension für CORBA, http://universe-phpext.sourceforge.net/

[Vaswani2002] Vaswani, Vikram: XML and PHP, USA: New Riders Publishing, 2002

[Victor2001] Frank Victor: Vorlesungsskript zu Wahlpflichtfach SAP R/3 Teil I (Sommersemester 2001), Fachhochschule Köln, Abteilung Gummersbach

[W3C_DOM_ActivityStatement] World Wide Web Consortium: Document Object Model (DOM) Activity Statement, http://www.w3.org/DOM/Activity.html

[W3C_SOAP1.2_Part0] World Wide Web Consortium: SOAP Recommendation Version 1.2 Part 0: Primer, http://www.w3c.org/TR/2003/REC-soap12-part0-20030624/, Stand: 24.06.2003

[W3C_SOAP1.2_Part1] World Wide Web Consortium: SOAP Recommendation Version 1.2 Part 1: Messaging Framework, http://www.w3.org/TR/2003/REC-soap12-part1-20030624/, Stand: 08.01.2004

[W3C_WSDL1.1] World Wide Web Consortium: Web Services Description Language 1.1, http://www.w3.org/TR/wsdl , Stand: 15.03.2001

[W3CAchitecture_2003] W3-Konsortium: Einstiegsseite zum Bereich Architektur: http://www.w3.org/Architecture/

[W3CSieben_2003] W3-Konsortium: Das W3C in sieben Punkten: http://www.w3.org/Consortium/Offices/Germany/sieben.html

[WestPlosHoff2001/III] Hartmut Westenberger, Jan Ploski, Rene Hoffmann: Vorlesungsskript zu Wahlpflichtfach Betriebliche Anwendungssysteme (Sommersemester 2001) Teil III: Entwicklung des Softwaremarktes für Unternehmensweite Lösungen, Fachhochschule Köln, Abteilung Gummersbach

[Wikipedia] Wikipedia – Die freie Enzyklopädie, http://de.wikipedia.org/

[WilhelmsKopp1999] Wilhelms, Gerhard und Kopp, Markus: Java professionell, Bonn: MITP-Verlag, 1999

[WormGeschichteWeb] Worm, Stefan: Grundlagen und Geschichte des Web, http://rnvs.informatik.tu-chemnitz.de/proseminare/www01/doku/web/index.htm

[Wutka2001] Wutka, Mark: J2EE Developers Guide - JSP, Servlets, EJB 2.0, JNDI, JMS, JDBC, Corba, XML, RMI, München: Markt+Technik Verlag, 2001

[XML-RPC.com] XML-RPC Home Page, http://www.xml-rpc.com

[XMLbenchSourceforgeNet] XML Benchmark Results, http://xmlbench.sourceforge.net/results/benchmark/

[XMLPathLanguage1.0] World Wide Web Consortium: XML Path Language (XPath) Version 1.0, http://www.w3.org/TR/xpath

[ZDNetJavaRMI] ZDNet.de: Verteilte Services mit dem Java RMI-Framework, http://www.zdnet.de/builder/program/0,39023551,39116198,00.htm?h, Stand: 05.12.2003

[ZendNuSOAP] Zend.com: Web Services with NuSOAP, http://www.zend.com/zend/tut/tutorial-campbell.php

[ZieglerModernDesign] Ziegler, Claus Dr.: Modern Design – Ein XML-basierter Designansatz für J2EE-Applikationen im EAI Umfeld, Java Magazin, Ausgabe 10/2002, Frankfurt am Main: Software & Support Verlag GmbH

[Zwißler2002] Zwißler, Sonja: Electronic Commerce, Electronic Business, Berlin/Heidelberg: Springer Verlag (Xpert.press), 2002