Seminar 2002
XML und Datenbanken

XML im Bereich EAI und B2B

Markus Schütze

Universität Kaiserslautern

0. Inhaltsverzeichnis

1. Einleitung

Diese Seminararbeit befasst sich mit den Möglichkeiten der Kommunikation innerhalb und zwischen Unternehmen. Sie hebt insbesondere die Rolle der Datenbeschreibungssprache XML hervor. Sie erläutert die neuen Anforderungen, Änderungen und Verbesserungen an bestehende Kommunikationstechnologien durch Einsatz dieser doch relativ jungen , noch immer in Entwicklung befindlichen Datenbeschreibungsprache.

Die Bedeutung des Internets und die zunehmende Vernetzung von Rechnern hat für die Wirtschaft immer mehr zugenommen. Wo anfänglich das 'sich Präsentieren', die Webpräsenz wichtig war, so rücken jetzt immer mehr Themen in den Vordergrund, die einerseits die Beziehung zwischen Consumern und Unternehmen, das sog. 'E-Commerce' betonen andererseits die Kommunikation innerhalb und zwischen Unternehmen, die Aneinanderkopplung bzw. Verzahnung ihrer Geschäftsprozesse und deren Optimierung mit Hilfe der sich immer weiter entwickelnden modernen Kommunikationstechnologien und -protokolle hervorheben.

Besonders das letztgenannte Thema ermöglicht es den Unternehmungen und Organisationen effizient und flexibel auf sich ändernde Marktbedingungen zu reagieren und so Kosten einzusparen und ökonomisch zu handeln.

Im Wesentlichen unterscheidet man beim e-business vier Kategorien :

- **B2C (Business-to-Consumer)** ,
  Handel zwischen Unternehmen und Endkunden, der im Wesentlichen geprägt ist durch Online-Shopping bzw. den Verkauf von Waren an den Endverbraucher.

- **B2E (Business-to-Employe)**
  Unternehmensinterne Kommunikation, Informationsaustausch aller in der Unternehmung beteiligten Personen zwecks besserer Zusammenarbeit. (Stichwort: Intranet)

- **B2B (Business-to-Business)**
  Verkauf von Gütern und Waren eines Unternehmens an andere Unternehmen. Dieses beginnt mit dem Beschaffungsprozess von Rohmaterialien und halbfertigen Gütern um eigene Güter und Waren zu produzieren, die man wieder weiterverkauft**.**

- **B2A (Business-to-Administration)**
  Informationsaustausch zwischen Unternehmung und staatlichen Organisationen z.B.: Zoll

Um E-Business, im Wesentlichen B2C und B2B, erfolgreich durchzuführen, d.h. Geld zu verdienen, benötigt man Geschäftsprozesse, die sich durch hohe Geschwindigkeit und Flexibilität auszeichnen. Dieses erreicht man durch Informationsflüsse ohne Zeitverzögerungen. Was wiederum bedeutet, dass die unterstützende Informationstechnologie immer weiter integriert werden muss. Auf der einen Seite müssen Datenformate entwickelt werden, die einen einfachen und effizienten Austausch von Geschäftsdokumenten z.B. Rechnungen, Bestellungen, Preislisten etc. ermöglichen. Hier wurden Erfahrungen auf dem Gebiet des Electronic Data

Interchange (EDI) gemacht, das sich aber nur in einem geringen Maß in der geschäftlichen Welt erfolgreich etabliert hat. Meistens wird EDI nur von grossen Unternehmen betrieben und u.U. dadurch kleineren Unternehmungen aufgezwungen. (siehe Kapitel 2) Gleichzeitig müssen diese Datenformate beherrschbar bleiben, das bedeutet, sie müssen einigermaßen lesbar für den Anwender sein, damit der sensible Datenaustausch zwischen den beteiligten Partnern einsehbar bleibt. Dieses kann man unter Einsatz der XML-Technologie hervorragend verwirklichen.
Auf der anderen Seite kann intra- und interbetriebliche Kommunikation nicht nur aus reinem Austausch von Daten über vorher wohldefinierte Schnittstellen bestehen, die den Einsatz solcher EDI-Systeme schwierig und teuer machen können.
Effektive Kommunikation zum Zweck der ökonomischen Verzahnung unterschiedlicher Geschäftsprozesse zwischen und innerhalb von Unternehmungen kann auch durch die Interoperabilität von unterschiedlichen Anwendungen und damit zwischen Geschäftsprozessen realisiert werden. Einzige Vorraussetzung ist, dass diese Anwendungen eine Schnittstelle zum Versand und Empfang von Daten haben. Enterprise Application Integration (EAI) stellt Technologien bereit , die den reibungslosen und effizienten Datenaustausch zwischen diesen Anwendungen ermöglicht. EAI schafft somit die Vorraussetzung einem oder mehreren Unternehmen eine Infrastruktur aufzubauen, die durch den zunehmenden ‘E-Business’ veränderten Integrationsanspruch gerecht wird.
Auch in diesem Bereich kann XML eine bedeutende Rolle spielen, wie es im dritten Kapitel dieser Arbeit aufgezeigt wird.

2. Electronic Data Interchange (EDI)

> *"Unter EDI (Abkürzung für engl.: electronic data interchange) versteht man den elektronischen Datenaustausch über Geschäftstransaktionen (Bestellungen, Rechnungen, Überweisungen, Warenerklärungen usw.) zwischen Betrieben. Die Daten werden in Form von strukturierten, nach vereinbarten Regeln formatierten Nachrichten übertragen. Dadurch ist es dem Empfänger möglich, die Daten direkt in seinen Anwendungsprogrammen weiterzuverarbeiten (Durchgängigkeit der Daten)"*
> [Hansen 1996]
> "*Die Übermittlung dieser strukturierten Daten mittels festgelegter Nachrichtenstandards von einer Computeranwendung in die andere und zwar auf elektronischer Weise erfolgt mit einem Minimum an menschlichen Eingriffen."*
> [Kölner Centrale für Coorganisation (CCG)]

EDI ermöglicht es vorher strukturierte Geschäftsdaten zwischen zwei oder mehreren Computersystemen auszutauschen, so dass diese von den beteiligten Systemen automatisch weiterverarbeitet werden können. Unter strukturierten Daten versteht man z.B.: digitalisierte Informationen, die in Formularen eintragbar, verarbeitbar und änderbar sind. Dieses sind meistens Daten aus dem Investition und Finanzwesen, z.B. Rechnungen, Buchungen, Lieferscheine, Zahlungsaufträge aber auch z.B. Zolldokumente.

Bedingt durch die zunehmende Europäisierung ist EDI auch in zunehmenden Maße immer mehr relevant für Institutionen und staatliche Behörden. Electronic Data Interchange ist keine neue Technologie, sondern wird seit nun mehr als ca. 20 Jahren praktiziert. Vorreiter der Idee 'Elektronischer Dokumentenaustausch' waren hauptsächlich Vertreter der amerikanischen Bekleidungsindustrien sowie große bedeutende Handelsketten. In unseren europäischen Breitengraden waren es die Automobilindustrie und das Finanzwesen, die frühzeitig den Nutzen einer automatisierten Dokumentübermittlung erkannten. EDI war für sie ein Mittel folgende Ziele zu erreichen :

- Konsistente und beschleunigte zwischenbetriebliche Workflows
- Vermeidung wiederholter Datenerfassung, Medienbrüche
- Intensivierung von Partnerbetreuung, Kommunikation
- Wettbewerbsvorteile durch schnellere Reaktion
- Kosteneinsparungen (z.B.: Abbau von Lagerbeständen via JiT)
- Reduzierung von Verwaltungskosten
- Verbesserte Lagerkontrolle
- Verbesserung des Kundenservice
- Unternehmenssicherung
- Erhaltung der Wettbewerbsfähigkeit

Da sich EDI nicht aus einer Quelle entwickelte, sondern seinen Ursprung in verschiedenen Branchen hatte , wurden viele verschiedene branchenspezifische Standards entwickelt, die untereinander inkompatibel waren. Auch wurde EDI nur von grösseren Unternehmen betrieben, da eine Einführung eines EDI-Systems hohe finanzielle Anschaffungen zur Folge hatte. So wurden unter anderem kleinere Firmen von grösseren Unternehmen dazu gedrängt EDI-Systeme einzusetzen, was für diese einen erheblichen Eingriff in Geschäfts- und Prozessabläufe darstellte. Noch heute widersetzen sich kleinere Firmen gegen den Einsatz von EDI, was zur Folge hat, dass sich die klassische Idee 'EDI' nicht so durchsetzte wie damals angenommen. Nach Schätzungen nutzen lediglich 5-10% der Unternehmen, für die der Einsatz von EDI vorteilhaft wäre, bereits heute das Potential dieser Technologie - und dies oft primär aufgrund des Drucks ihrer Geschäftspartner.
Auch die angesprochene Standardvielfalt tat ihr übriges. So war es z.B. nicht ohne weiteres möglich Dokumente aus der Automobilindustrie ins Finanzwesen zu transferieren und umgekehrt. Eine Auflistung existierender Standards liefert uns auszugsweise folgende Tabelle:

Bezeichnung	Branche	Kurzbeschreibung
CEFIC	Chemische Industrie	Conseil Europ´een de Federations del'Industrie Chemique
SEDAS	Konsumgüterindustrie	Standardregeln einheitlicher Datenaustauschsysteme
EDIFICE	Computer- und Elektroindustrie	Electronic Data Interchange Forum for Companies
TRADACOMS	Trade Data Communication Standard	Trade Data Communication Standard

RINET	Versicherungen	Reinsurance and Inscurance Net
SWIFT	Finanzgewerbe	Society for Worldwide Interbank Financial Telecom
ODETTE	Automobilindustrie	Organization for Data Exchange by Teletransmissions in Europe
VDA	Automobilindustrie	Verband der Automobilindustrie

Diese Standards erfreuen sich noch heute eines grossen Rückhaltes und werden zum Teil noch weiterentwickelt, obwohl es schon Standards gibt, die umfangreicher und umfassender sind.

So entwickelten die Vereinten Nationen einen global gültigen, branchenüberbrückenden Standard : UN/EDIFACT. Ihm gegenüber steht der amerikanische X12 Standard der zwar branchenübergreifend ist, aber nur nationale Bedeutung erlangt hat.

2.1 Der UN/EDIFACT-Standard

Der EDIFACT-Standard ist sehr komplex bedingt durch seine branchenüberbrückende und globale Gültigkeit, so dass er allein sehr schwer zu handhaben ist. Deswegen wurden Subsets gebildet, die ihn für gewisse Anwendungsbereiche handhabbarer machen, untereinander aber trotzdem kompatibel bleiben. So sind EDIFACT-Subsets wohldefinierte Untermengen des EDIFACT Standards.

Auch wurden obengenannte Standards in den EDIFACT Standard übernommen, sind aber zum Teil keine echten Untermengen, da sie teilweise eine ganz andere Syntax und Semantik verfolgen. ‘Richtige’ Subsets des UN/EDIFACT Standards sind der aus der Automobilbranche stammende ODETTE-Standard oder der EANCOM-Standard, mit dem man über 200 Geschäftsvorfälle behandeln kann.

Der EDIFACT-Standard gliedert die Geschäftsvorfälle in Messages oder Nachrichten.

Ein kleiner Auszug dieser Nachrichtentypen liefert uns folgende Tabelle:

Message	Beschreibung
APERAK	ANWENDUNGSFEHLER- UND BESTÄTIGUNGS-NACHRICHT
AUTHOR	AUTORISIERUNGS-NACHRICHT
.....	
FINCAN	STORNO-NACHRICHT
FINPAY	MULTIPLER INTERBANK-ZAHLUNGSAUFTRAG
FINSTA	BANKKONTOAUSZUG
GENRAL	ALLGEMEINE NACHRICHT
GESMES	ALLGEMEINE STATISTISCHE NACHRICHT
HANMOV	NACHRICHT FÜR DEN LADUNGS-/GÜTERUMSCHLAG UND -TRANSPORT
.....	
ORDERS	BESTELLUNG
ORDRSP	BESTELLANTWORT
....	
VESDEP	ABFAHRT DES SCHIFFES
WASDIS	MÜLLENTSORGUNGS-INFORMATION
WKGRDC	BESCHEID ÜBER ARBEITSERLAUBNIS
WKGRRE	ANTRAG AUF ARBEITSERLAUBNIS

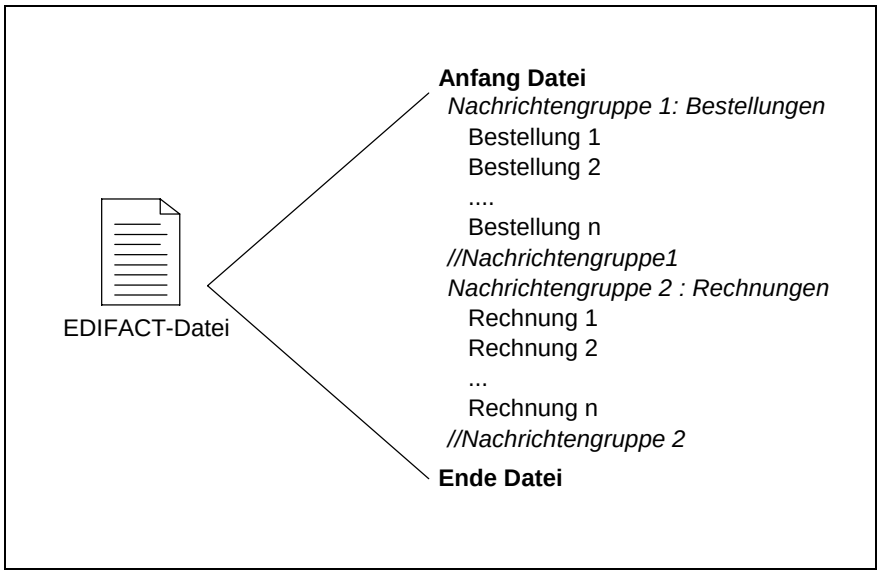
Ein komplettes Verzeichnis aller Messagetypen findet man unter

http://www.unece.org/trade/untdid/welcome.htm . Oder unter

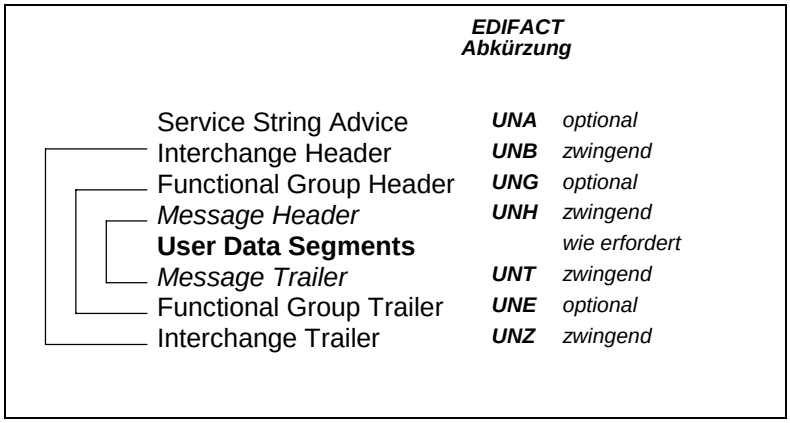
http://www.edifactory.de/edifact/D01A/msglist.html.

2.2 Aufbau einer EDIFACT Übertragungsdatei

Im folgenden wird der Aufbau einer Message beschrieben, wobei aber zu beachten ist, dass diese Arbeit nur einen Einblick in das EDIFACT-Format liefert, da sonst der Rahmen dieses Seminars gesprengt wird.
Die EDIFACT-Syntaxregeln, zusammengefasst in Message Implementation Guides sogenannten MIG's definieren den Aufbau der jeweiligen Message. Der Aufbau einer Message ist auch in der Norm ISO9735 festgelegt. Dort wird beschrieben wie zu verschickende Daten in einzelne Segmente zusammengefasst werden. Diese werden zu Nachrichten zusammengefasst, die zusammen in einer Übertragungsdatei gekapselt werden. In dieser Übertragungsdatei werden die einzelnen Nachrichtentypen nach Typ ihres Geschäftsvorganges gegliedert und sortiert. Zur Veranschaulichung sei auf folgende Grafik verwiesen, in der die einzelnen Syntaxeinheiten detailliert beschrieben sind:



Am Anfang jeder EDIFACT -Übertragungsdatei sind Steuersegmente (z.B. UNH, UNT etc.) definiert, die standardübergreifend für alle Nachrichten gelten.
Somit sieht der Aufbau einer EDIFACT-Nachricht s gemäß der oben genannten ISO-Norm wie folgt aus:



Wobei die einzelnen Abkürzungen für einzelne Messagsegmente folgende Bedeutung haben.

Tag	Name	Nutzung
UNA	Service String Advice (Servicesegment)	Eine feste Stringfolge, die in der Übertragungsdatei verwendete Trennzeichen definiert.
UNG	Functional Group Header	Definiert und spezifiziert eine funktionale Messagegruppe
UNB	Interchange Header (Servicesegment)	Steht am Beginn jeder EDIFACT-Übertragungsdatei und enthält Angaben zum verwendeten Zeichensatz, Nachrichtensender und Empfänger, eine eindeutige Referenzennummer und weitere Angaben.
UNH	Message Header (Servicesegment)	Steht am Beginn jeder EDIFACT-Nachricht und enthält Angaben zum Nachrichtentyp, dem verwendeten Standard, etc.
UNT	Message Trailer (Servicesegment)	Ist das letzte Segment jeder EDIFACT-Nachricht und enthält die Anzahl der Segmente der Nachricht und die selbe Nachrichtenreferenznummer wie das UNH.
UNE	Functional Group Trailer	Beendet eine funktionale Messagegruppem spezifiziert Nachrichtenmenge und spezifiziert funktionale Gruppenreferenz
UNZ	Interchange Trailer (Servicesegment)	Kennzeichnet das Ende einer EDIFACT-Übertragungsdatei und enthält die Anzahl der übertragenen Nachrichten und die Referenzennummer des UNB.

Am deutlichsten wird jedoch die Struktur einer Message an einem Beispiel.

```
UNA:+,? 'UNB+UNOA:2+FHPEDAL+HUBERGMBH+990802:1557+
9908021557'UNH+INVOIC0001+INVOIC:D:96A:UN'BGM+380+
9908001+9'DTM+3:19990802:102'RFF+ON:O0010001'DTM+4
:19999715:102'NAD+SE++Fahrradhandel Pedal++Waginge
rstr. 5+München++81549'NAD+BY++Huber GmbH++Obstgas
se 2+München++81549'LIN+1++4711.001'IMD+F++:::Fahr
rad, Damen'QTY+47:1:PCE'MOA+66:750'PRI+AAA:750'LIN
+2++4711.002'IMD+F++:::Luftpumpe, Stand-'QTY+47:1:
PCE'MOA+66:19,9'PRI+AAA:19,9'LIN+3++4711.003'IMD+F
++:::Ersatzventil'QTY+47:3:PCE'MOA+66:7,5'PRI+AAA:
2,5'UNS+S'MOA+79:777,4'MOA+124:124,38'MOA+128:901,
78'TAX+7+VAT+++:::16+S'UNT+28+INVOIC0001'UNZ+1+990
8021557'
```

Hier werden zuerst das Datenelementtrennzeichen (+), das Dezimalkommazeichen (,), sowie das Releasezeichen(?) der Nachricht definiert. Nach einem reservierten Freizeichen wird dann das Segementtrennzeichen (‘) festgelegt. Danach wird die Nachricht, ähnlich wie bei einem Briefumschlag -eingeleitet durch ein UNB+UNOA- mit Sender (FHPEDAL) und Empfänger (HUBER GmbH) versehen. Dann bekommt die Nachricht einen Datum/Zeitstempel (02.08.99:1557-> 990802:1557) und eine von der Senderfirma festgelegten Nachrichtenreferenznummer (9908021557). Das erste Segment der Nachricht wird mit einem abschließenden Hochkomma (‘) geschlossen. Das nächste Segment, eingeleitet durch ein ‘UNH’, legt den EDIFACT Nachrichtentyp (hier INVOIC) und Nachrichtenversion fest (hier D.96A). Dieses kennzeichnet die verschickte Nachricht als Rechnung, deren Aufbau gemäß UND.93A-Standard erfolgt. Im nächsten Segment werden die benutzerdefinierten Daten vermerkt, die sich aber strikt an den Standart halten müssen. Zuerst werden, eingeleitet durch das Flag ‘BGM’ Informationen wie Rechnungsnummer übermittelt. Auch wird hier gekennzeichnet, ob es sich bei diesem übermittelten Dokument um ein Original oder eine Kopie handelt. Die ‘9’ am Ende des BGM Segmentes weist dieses Beispiel als Original aus. Dann kommen Rechnungsdatum und Bestellinformationssegmente, in denen wiederum Bestelldatum, Artikelnummer, Mengenbezeichnung und Preis codiert sind, wie es die Message Implementation Guides vorschreiben. So ist ein Datumssegment DTM in der betreffenden MIB wie folgt definiert:

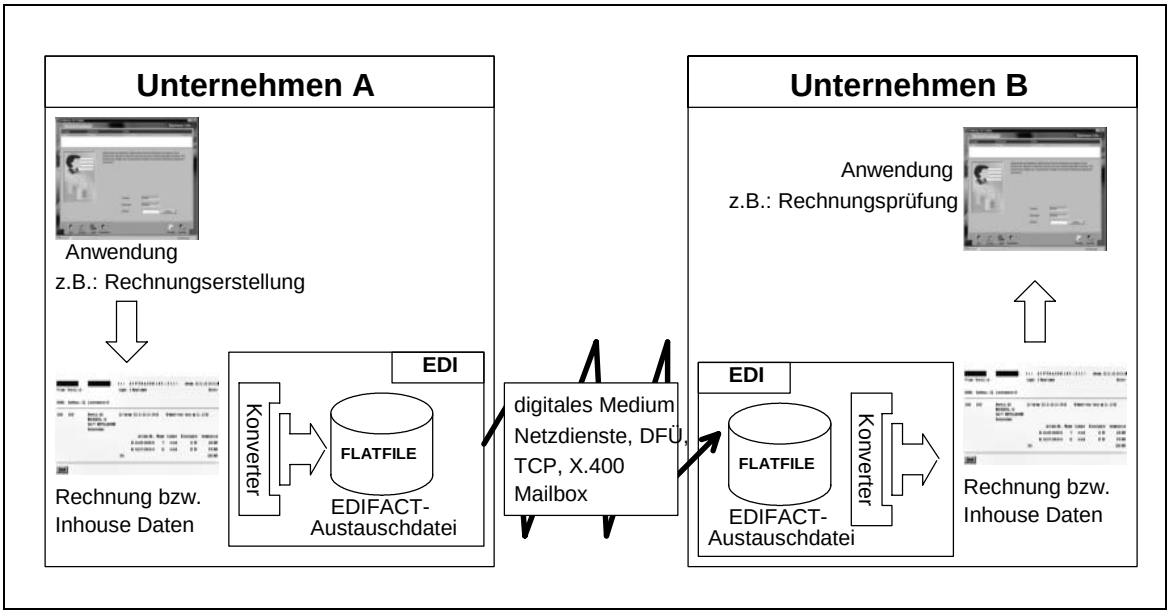
DTM - 1 Date/time/period				
UN/EDIFACT Directory 96.A M		D6 M		
A segment specifying the date and, if required, the time the message is created.				
Segment number : 3				
UN/EDIFACT Directory 96.A		D6 Description		
C507 **Date/time/period**	M	M		
2005 Date/time/period qualifier	M an..3	M	*R	137 = Document/message date/time
2380 Date/time/period	C an..35	R		A date in the future can be accepted given the creation of messages across different time zones.
2379 Date/time/period format qualifier	C an..3	R	*	102 = CCYYMMDD
			*	203 = CCYYMMDDHHMM

Das Segment DTM besteht aus der Gruppe 507 mit 3 Feldern , wobei das Feld mit der Bezeichnung 2005 eingetragen werden muss (zu sehen am M in Spalte 2 der letzten Zeile) und die Felder 2380 und 2379 eingetragen werden können (zu sehen am C) bzw. deren Einsatz empfehlenswert ist (R). Gleichzeitig werden

Datenformate der Felder festgelegt. So darf das Feld 2005 aus 3 alphanumerischen (an) Zeichen bestehen. Das Feld 2380 aus 35 alphanumerischen Zeichen und das Feld 2379 wiederum aus 3 Zeichen derselben Art. Wobei das erste und das letzte Feld Codes beinhalten die erstens Aussagen über Art des Datums machen (3 = Rechnungsdatum, 4 = Bestelldatum etc..) und zweitens die Datumscodierung im Feld 2380 festlegen.
So liefert DTM+3:19990802:102 die Information, dass die Rechnung vom 02.08.1999 ist. Dieses Datum wird nach CCYYMMDD (Jahrhundert-Jahundert-Jahr-Jahr-Monat-Monat-Tag-Tag) kodiert, was das letzte Feld durch den Code 102 festlegt.
Wer weitere Informationen über den Aufbau von EDIFACT-Nachrichten sucht, sei auf die Message Implementation Guides der UN, zu finden unter http://http://www.unece.org/etrades/download/downmain.htm#edifact verwiesen.
Wie werden und wann werden aber nun solche EDI-Austauschdateien aus Rechnungsformularen erstellt ? Die Antwort darauf liefert uns das nächste Teilkapitel.

2.3 Der klassische EDI-Vorgang

Den klassischen EDI-Dokumentaustauschvorgang zweier Unternehmungen kann man sich am besten wie folgt vorstellen :



Der EDI-Dokumentaustauschprozess lässt sich grundsätzlich in zwei Teilprozesse aufteilen:

1. **Konvertierungsprozess**
2. **Übermittlungsprozess**

Der Konvertierungsprozess übersetzt die anfallenden auszutauschenden Daten von einem unternehmensinternen Format (Inhouse - Daten) in das oben beschreibende EDIFACT-Format (FLATFILE) und umgekehrt. Hierzu

bedarf es eines Konverters. Dieser ist ein Teil des EDI-Systems , das sowohl die Konvertierung der Daten als auch die Kommunikation vollautomatisch abwickelt. Damit die EDIFACT-Daten von der Anwendung des Kommunikationspartners (Unternehmen B) ebenfalls automatisch weiterverarbeitet werden können , muss auch bei ihm ein EDI System installiert sein, das die übertragenen EDIFACT-Daten als FLATFILE wieder in ein für ihn verarbeitbares Inhouseformat automatisch umwandelt und an die weiterverarbeitende Applikation weiterreicht Diese Möglichkeit der automatisierten Weiterverarbeitung ermöglicht den Datenimport in betriebswirtschaftliche Anwendungen, wobei es im Regelfall keine Rolle spielt, welches Anwendungsprogramm gerade benutzt wird.

Beim Übermittlungsprozess werden die EDIFACT-Daten als Flatfile von dem Unternehmen A an das Unternehmen B bzw. an mehrere Kommunikationspartner übermittelt. In der Praxis kommen meistens zwei Arten der Übermittlung zustande :

- **Direkte Kommunikation**
- **Entkoppelte Kommunikation über Mailboxsysteme**.

Unter direkter Kommunikation versteht man die Einrichtung einer direkten Kommunikationsverbindung (Point-to-Point) zwischen Sender und Empfänger. Dieses wird gerade bei der Verarbeitung von zeitkritischen Daten praktiziert, weshalb sie zum Beispiel in der Automobilindustrie genutzt wird. Dazu wurde das Übertragungsprotokoll ODETTE File-Transfer-Protokoll (OFTP) entwickelt, das eine sichere und schnelle Punkt zu Punkt Verbindung ermöglicht.

Bei der entkoppelten Kommunikation werden Sende- und Empfangsvorgang der Daten durch die Verwendung eines Mailboxsystems (Store-and-Foreward) entkoppelt. Dies hat den Vorteil, dass keine permanente Verbindung zu den Kommunikationspartner aufrechterhalten werden muss, sondern die Daten lediglich in bestimmten Zeitabständen in die Mailbox der Empfängerseite bzw. eine eine eigene Mailbox, die für ihn eingerichtet wurde, platziert werden.

Dieses ermöglicht den gleichzeitigen Versand der Daten an mehrere Empfänger und Einsparung von Kommunikationskosten. Zur Übermittlung der Daten wird standardmäßig das X.400-Protokoll angewendet. Für dieses Protokoll bietet zum Beispiel u.a. die Telekom mit der Telebox 400 , einem Teildienst von T-BusinessMail so ein Mailboxsystem an. (zu finden unter : http://www.telekom.de/dtag/ipl1/cda/level2_a/0,,11471,00.html)

Konvertierungs- und Übermittlungsprozess werden zu einem vollautomatisierten Gesamtprozess konkateniert.

Die Umkonvertierung von einem Inhouse Format in das EDIFACT Format liefert den Vorteil, dass es weltweit gültig ist. Würde man die Umkonvertierung sein lassen , und statt dessen die Inhouse Fileformate verschicken, so würde der Informationsaustausch zwischen verschiedenen Unternehmen zu einem nicht lösbaren Problem werden.

Aber auch so existieren schon negative Faktoren die im nachfolgenden zur Verdeutlichung kurz aufgezählt werden.

2.4 EDI-Probleme

Wie man insbesondere im Kapitel 2.2 erahnen kann, sind die heutzutage gängigen EDI-Formate ziemlich starre Regelwerke, die sich eng an definierte Geschäftsprozesse anlehnen, um damit möglichst viele Geschäftsvorgänge abzudecken. Aus diesem Grund ist EDIFACT ein ziemlich komplexes Regelwerk, das von einer zentralisierten Kommission weiterentwickelt werden muss. Es ist nicht möglich neue Geschäftsprozesse zu implementieren, die nötig sind , um z.B.: neue Geschäftsvorfälle zu modellieren, die mit dem Einzug des E-Commerce in die Unternehmenswelt immer häufiger auftreten. Auch sind die meisten EDI-Lösungen plattformabhängig, was eine Einführung bzw. Erneuerung von EDI sehr teuer macht. Dies ist ein Grund, warum grössere Unternehmen EDI häufiger benutzen und den 'kleineren' Firmen dieses aufzwingen. Die Fähigkeit EDI einzusetzen ist manchmal sogar ein Faktor, der eher wettbewerbshindernd wirkt.

Es wäre vorteilhafter ein weniger starres EDI Format zu entwickeln, das sich leicht , ohne den vorher erwähnten langwierigen Standardisierungsprozess anpassen lässt, und zusätzlich über das World Wide Web genutzt werden kann, um Kommunikationskosten zu reduzieren.
Genau diese Idee greift Electronic Data Interchange in Verbindung mit der Metadatenbeschreibungssprache XML auf.

2.5 XML/EDI

In der Industrie gibt es bereits zahlreiche Maßnahmen zur Einführung von XML, u. a.:

- **Commerce XML (cXML)**
  ein vom Unternehmen ARIBA geschaffener Standart für den Austausch von Geschäftsnachrichten. Weltweite Verbreitung, aber überwiegend in Amerika. Framework für den Austausch von Produktkatalogen, Geschäftstransaktionen und komplexe Dokumente. Definiert direkte Kommunikation der Beschaffungsysteme über das Web. Wird heute von einem unabhängigen Konsortium weitergepflegt (www.cxml.org)

- **XML based CommonBusiness Library (xCBL**)
  von Commerce One initialisiert, Ermöglicht Austausch strukturierter Daten über das Web (Rechnungen, Bestellungen, Produktdaten), definiert aber nicht wie Nachrichten transportiert werden.

- **RosettaNet**
  XML-Framework, das auf Initiative von mehr als 40 Unternehmen der IT entwickelt wurde legt neben Nachrichtenformaten auch Richtlinien fest über Abläufe und Regeln von Transaktionen. Strebt Angleichung der Geschäftsprozesse in der IT an.

- **Electronic Catalog XML**
  von Requisite Technology angeboten, Hauptanwendungsgebiet von ecXML liegt im Austausch von Kataloginhalten und Katalogstrukturen, liefert Metadatenstruktur, um verschiedene Formate miteinander zu verbinden.

- **Open Catalog Format (OCF)**
  offener Standard für die Beschreibung von Produktkatalogen. Repräsentation, Speicherung und Abbildung von Produktinformationen. offener, allgemeiner Standard.

- **Produkt Data Markup Language**
  Format für den Austausch von Produktinformationen zwischen kommerziellen Systemen

Neben diesen Formaten ist XML/EDI ein interessanter Ansatz um kostengünstig die EDI-Idee zu verwirklichen und so einer grossen Menge an Unternehmen, Organisationen und Firmen zukommen zu lassen.

Die Vorteile von XML prädestinieren es für den Einsatz von Electronic Data Interchange. Diese sind :

- Aufgrund der Gliederung nach Inhalt ist eine sinnvolle Suche innerhalb des XML-Dokuments nach bestimmten Elementen oder Elementen mit bestimmten Eigenschaften möglich.
- Ein XML-Dokument kann durch die Verwendung von Stylesheets in unterschiedlichen Formaten oder Ansichten dargestellt und universell eingesetzt werden.
- XML-Dokumente können leicht durch (unterschiedliche) Programme ausgewertet werden.
- XML-Dokumente sind menschenlesbar.
- Anhand von DTDs kann ein Validieren der Dokumente vorgenommen werden.
- DTD's beschreiben Syntax und Semantik. XML-Dokumente sind selbsterklärend.
- Durch das Document Object Model (DOM) wird der Zugriff auf die Elemente der XML-Dokumente beschrieben. Mit dieser einheitlichen Schnittstelle wird die Programmierung von XML-Applikationen vereinfacht und damit kostengünstiger. Das unterschiedliche Format von Dokumenten ist kein so großes Hindernis mehr beim Datenaustausch und die Dokumente sind sehr flexibel einsetzbar.

Deshalb ist es sinnvoll die XML-Technologie mit EDI zu XML/EDI zu verbinden.

XML/EDI liefert gegenüber klassischem EDI folgende Vorteile.

2.6 Vorteile von EDI/XML

- XML/EDI ist kompatibel zu klassischem EDI.
- Durch Anbieten von Schnittstellen zu klassischem EDI sind bestehende schon finanzierte EDI-Systeme weiternutzbar.
- Es ist offenen Standards unterworfen, und kann somit leicht weiterentwickelt werden.
- Daten und Regeln, die den Aufbau der Daten beschreiben, sind eng miteinander verknüpft. Die XML/EDI-Dokumente sind somit selbsterklärend.
- Es unterstützt globale Repositorys, in denen global DTD's und z.B.: Elementbeschreibungen abrufbar sind. (sowohl für den Menschen als auch maschinell.)
- Es erlaubt einfache Erweiterung bestehender Anwendungen.
- Besteht nicht nur aus den Daten und den Beschreibungen, sondern liefert im Paket Konverter-, Worklow und Prozessmanangmentutilities dazu, was wiederum eine einfache Anpassung der Dokumente an Geschäftsvorfälle und -prozesse erlaubt.
- Es ermöglicht Interaktive Transaktionen im World Wide Web.
- browserfähig.
- Es ist einfacher und günstiger zu implementieren als klassisches EDI.

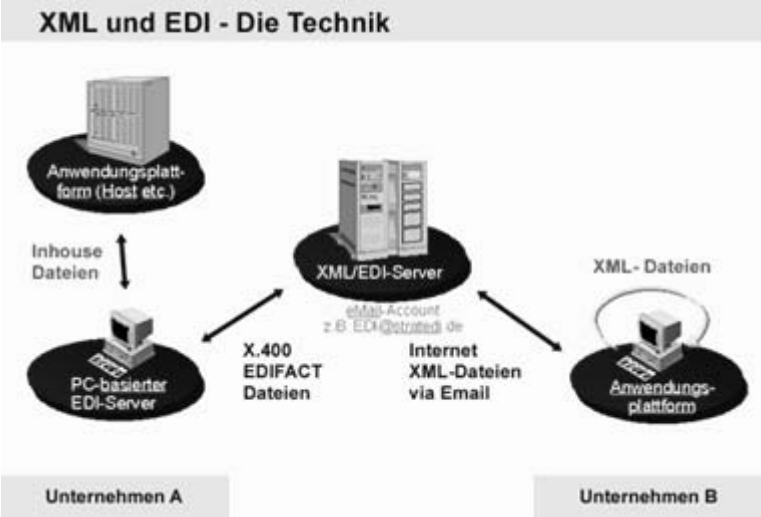
2.7 Änderung der EDI-Systemstruktur durch XML

Wie funktioniert aber nun XML/EDI?

XML/EDI ist ein Verbund von verschiedenen Technologien:

- **XML**
- **EDI-Standards**
- **Templates**
- **Agenten**
- **Repositories**

Die Metadatenbeschreibungssprache XML dient dazu die auszutauschenden Daten zu strukturieren. Jedes Element des Dokumentes kann problemlos über das DOM angesprochen werden und man kann gleichzeitig beim Extrahieren der Daten ihre Beschreibung bekommen. Durch Einsatz von XML-Parsern kann man die Korrektheit der hergeleiteten Dokumente nachweisen. In EDI-Standarts (z.B.: EDIFACT -MIBS) sind Geschäftsvorfälle beschrieben. Es ist sinnvoll, vorhandene, langwierig getestete Standards weiter zunutzen, anstatt sie neu zu entwickeln. Sie müssen nur an die neue Technologie angepasst werden. Templates definieren Regeln zur Verarbeitung von Daten, hierzu gehören DTD's, die die Daten strukturieren und beschreiben, wie sie zu analysieren sind. Gleichzeitig beinhalten Templates auch Vorschriften wie die Daten anzuzeigen sind. Dies wird mit Hilfe von XSL (Extensible Style Language) verwirklicht. Auch kann ein XML Dokument Informationen beinhalten, die die Verarbeitung der Daten vorschreiben. Die Daten werden damit selbsterklärend. Agenten führen von den Templates definierte Aktionen aus. Sie bestimmen die Ansichtsweise der übermittelten Informationen. Beispielsweise Java-Applet's , welche z.B. die Darstellungsinformationen der XSL Datei auslesen und die in einer XML überlieferten anzeigen. In Repositories werden alle benötigten Informationen z.B. zur Wiederverwendung gespeichert und global zur Verfügung gestellt. Hierzu gehören DTD'S, XSL's und andere wichtige Informationen. Der Sinn und Zweck dieser Bibliotheken liegt darin, dass Agenten benötigte Informationen durch sie abrufen können, die diese zur Verarbeitung von Information oder Dokumenten brauchen. Auch auf die oben genannten EDI-Prozesse hat XML/EDI Auswirkungen. So kann man sich den Austausch von Dokumenten wie folgt vorstellen :



Quelle: www.ecin.de

Gleichzeitig zeigt dieses Beispiel auch auf, wie man klassisches EDI mit der XML Technologie zusammenschliessen kann.

Unternehmen A benutzt noch die ältere EDIFACT-Schnittstelle zum Austausch. (Funktionsweise siehe Kapitel 2.3) Unter Nutzung der klassischen X.400 Schnittstelle, schickt es seine Daten an einen XML/EDI Server. Dieser ist in der Lage mit Hilfe von Repositories und den Agenten aus der EDIFACT Datei, eine XML/EDI-Datei zu generieren und zu verifizieren. Mit seinen Vorteilen. Dort werden auch die Daten verschlüsselt, so dass sie, wenn sie zum Beispiel über TCP/IP verschickt werden, von Dritten nicht eingesehen werden können. Danach erfolgt das Routing an die betroffenen Geschäftspartner (Unternehmen B) . Die Anzeige und die Bearbeitung der Daten beim Empfänger erfolgt in einem Browser, der mit Hilfe von Applets eine komfortable Arbeitsweise ermöglicht. Die Bearbeitung mit Hilfe einer Browser/Applet-Kombination eröffnet u.a. auch den Vorteil, dass der Empfänger der EDI-Daten nicht unbedingt ein EDI System besitzen muss. Er ist aber an diese Lösung nicht gebunden, sondern kann auch ein eigenes EDI System zur Bearbeitung benutzen, da die XML/EDI Daten leicht in seine eigenes INHOUSE-Format umgewandelt werden können. (Serverseitig oder auf der Clientseite (Unternehmen B))
Die vom Unternehmen B in XML generierte Antwort wird wieder an den XML Server zurückgesandt, der diese in EDIFACT Daten umwandelt und an den Sender zurücksendet. Sematik- und Syntaxüberprüfung erfolgt ebenfalls auf dem Server.

Wer sich für die Konvertierung von EDI auf XML interessiert sei auf das im Anhang Teil A angefügte Beispiel verwiesen, das exemplarisch eine mögliche Konvertierung von EDIFACT Daten in XML aufzeigt. Mit dem aufgezeigten Wissen aus Kapitel 2.2 sollte ein Vergleich leicht möglich sein.

2.8 Nachteile von EDI/XML

Beim Beispiel im Anhang wird womöglich auch ein gravierendes Problem bei der Ausführung von EDI mit Hilfe von XML deutlich. Die generierten XML Datei und deren Zusatzdateien beinhalten 6 bis 10 mal mehr

Metadaten als die Ausgangs-UN/EDIFACT-Datei. Das Verhältnis von Nutz- zu Strukturierungsinformation ist wesentlich schlechter. Es werden also somit höhere Anforderungen an die unterliegende Kommunikationsschicht gestellt. Was zu Problemen der Antwortzeiten führen kann, wenn der Verbindungsdurchsatz zu gering gewählt wird.

3. Enterprise Application Integration

3.1 Was ist EAI ?

Definition:

„Enterprise Application Integration (EAI) kombiniert die Technologien und Prozesse, die es ermöglichen, dass unterschiedliche Anwendungen eines und verschiedener Unternehmen Geschäftsvorfallinformationen in Daten und Contexten austauschen können, die sie alle verstehen können."

„Enterprise Application Integration macht es sich zur Aufgabe, dass unabhängig voneinander entwickelte Anwendungen sich verständigen können."

„Enterprise Application Integration ermöglicht uneingeschränktes Verteilen von Daten und Geschäftsprozessen unter jeglichen zusammengeschlossenen Anwendungen und Datenquellen in einem Unternehmen."

Diese drei Definitionen beschreiben das Wesen von EAI ziemlich genau. EAI ermöglicht es, dass sich unterschiedlichste Applikationen untereinander Daten austauschen können ohne vorher mit großen Aufwand an die Anforderungen, die der Bereich der Kommunikation erfordert, angepasst zu werden. Die einzige Vorraussetzung, die diese miteinander kommunizierenden Anwendungen erfüllen müssen, ist eine Möglichkeit bzw. Schnittstelle bereitzustellen, die überhaupt Kommunikation erst zulässt. Enterprise Anwendungs Integration hat es sich zur Hauptaufgabe gemacht, unterschiedlichste Anwendungen in einer heterogenen Systemumgebung zu vereinen. Dazu vereinigt EAI folgende Eigenschaften :

- EAI ist nicht nur der Aufwand Datenformate umzuwandeln und Geschäftsprozessinformationen verschiedenen Anwendungen zukommen zu lassen, EAI ist vielmehr auch das Abbilden und die Lifeintegration von Geschäftsprozessen durch beliebige Kombination von Geschäftsapplikationen auf die Kommunikationsinfrastruktur eines oder mehrerer Unternehmen.
- EAI ist keine 'Instant'-Lösung, sondern speziell auf die Belange spezifischer Geschäftsprozesse zugeschnitten. EAI ist kein statisches System, es ist dynamisch an sich ändernde Anforderungen anpassbar, und zeichnet sich durch eine ausgezeichnete Erweiterbarkeit aus.
- EAI bildet den 'Klebstoff' der die unterschiedlichsten Protokolle, Software und Hardware miteinander verbindet und mit Hilfe dieser Bausteine Geschäftsprozessmodelle nachbildet und automatisiert. Unternehmungen, die ein funktionierendes, leistungsfähiges EAI-System betreiben,

können somit bessere Effizienz ihrer Geschäftsprozesse und bessere ökonomische Handlungsfähigkeit erreichen.

3.1.2 Die verschiedenen Ebenen der Anwendungsintegration

Anwendungsintegration ist nicht gleich Anwendungsintegration. Im Bereich der EAI gibt es verschiedene Ebenen, um Anwendungsintegration zu erreichen.
Es gibt drei Integrationsstufen, die gleichzeitig die Leistungsfähigkeit von EAI-Systeme beschreiben:

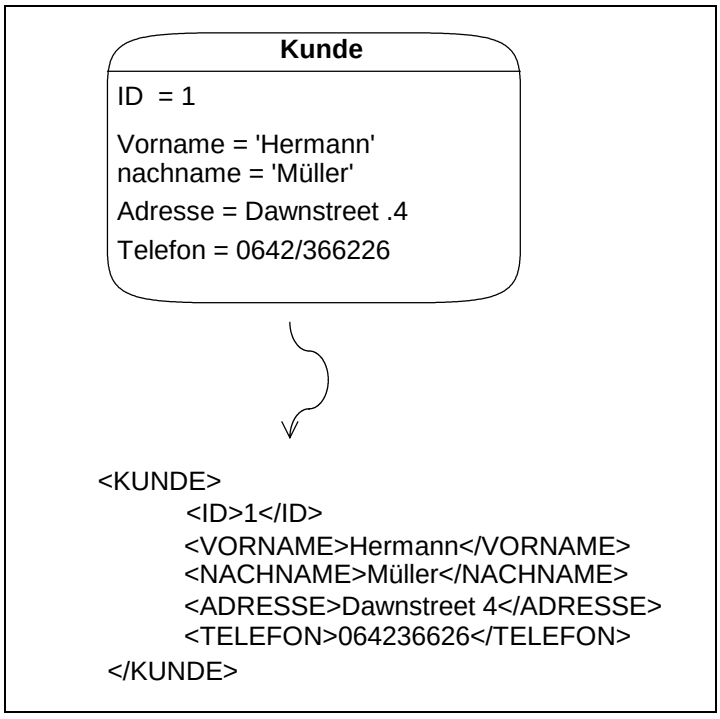
1. **Anwendungsintegration auf Datenebene**
2. **Anwendungsintegration auf Objektebene**
3. **Anwendungsintegration auf Prozessebene**

Bei der Anwendungsintegration auf Datenebene wird der Austausch von Informationen hauptsächlich als das Verfügbarmachen von Daten gesehen. Bei den meisten Unternehmungen ist Anwendungsintegration auf Datenebene der Einstiegspunkt ins Feld der EAI. Datenbasiertes EAI ermöglicht den Austausch von Daten zwischen Anwendungen. Unterstützt wird EAI dadurch durch Middlewaresysteme. Es existieren viele Datenintegrationstools, die einen Einstieg in EAI relativ günstig machen und Zugriff auf Datenbasen ohne Abänderung von Programmcode ermöglichen. XML spielt hier mit allen seinen Vorteilen eine große Rolle, da es einen flexiblen Austausch von strukturierten Daten ermöglicht.
Bei der Anwendungsintegration auf Objektebene wird versucht Daten und Gegenstände von Geschäftsvorfällen auf Objekte abzubilden. Diese Objekte kapseln Daten und definieren gleichzeitig Methoden auf ihnen. (vergleiche objektorientierte Programmierung). Sinn und Zweck dieser Ebene ist es die spezifischen in einem bestimmten Format vorhandenen Daten unter Einhaltung der korrekten Semantik in ein anderes, dem Zielsystem verständlichen Format umzuwandeln. Auch hier eignet sich der Einsatz von XML hervorragend und wird auch benutzt , um zu übertragene Objekte zu definieren. (siehe Enterprise Application Integration with Java und XML , JP Morgenthal Kapitel 9) Diese werden in Messages gekapselt und mit Hilfe der EAI-Middleware über spezielle Messagebehandlungsroutinen (CORBA, mqSeries, XML-RPC, SOAP, etc) verschickt. Gleichzeitig erfordert diese Art der Datenbehandlung spezielle Anpassungen an den Anwendungsschnittstellen der zu integrierenden Applikationen. Bei der Anwendungsintegration auf Prozessebene werden inner- und interunternehmerische Geschäftsprozesse unterstützt und existierende Anwendungen in einen Processflow integriert. Im Laufe dieser Prozessausführung werden wiederum Objekte generiert verschickt und weiterverarbeitet. EAI-Middleware dient hier nicht zur reinen Messageverarbeitung, sondern zusätzlich als eine ‘Workflowengine’ , die Prozessvorgänge innerhalb und zwischen Unternehmen vorschreibt und unterstützt. Diese Interationsstufe ist höchste der zu erreichenden Integration, da sie existierende eigentlich unvereinbare Anwendungen in ein zusammenhängendes System von Geschäftsprozessen integriert.
Diese Ausarbeitung wird hauptsächlich Aspekte der Anwendungsintegration auf Objektebene erläutern, und im folgendem die verschiedenen Konzepte (XML, XML-RPC, Message Queue) diskutieren, die diese Ebene der Integration realisieren.

3.1.3 Die Rolle von XML in EAI-Systemen

Da die Integration von Applikationen im Bereich EAI durch Austausch von Information auf Basis von Daten , Definition von Objekten und Kapselung dieser in Messages erfolgt. Ist es wichtig eine gemeinsame Informationsmodellierung zu finden , damit alle beteiligten Systeme sich untereinander verständigen können.

Mit XML wurde eine ausgezeichnete Lösung gefunden dieses zu bewerkstelligen, da sich Xml anbietet, jegliche Datenstrukturen strukturiert zu modellieren. XML bietet insofern ausgezeichnete Möglichkeiten eine Abbildung von komplexeren Datenstrukturen auf XML-Dokumente zu definieren, sowie deren Umkehrung. Somit entwickelt sich XML durch die massive Unterstützung von industriellen Grössen (z.B.: IBM, Microsoft, HP etc.) zu einem wichtigen Hilfsmittel, den Datenaustausch mit Hilfe von Middleware zwischen den unterschiedlich spezifizierten Schnittstellen unterschiedlichster Anwendungen zu realisieren.
So kann zum Beispiel das anwendungsinterne Objekt Angestellter leicht in XML transformiert werden. (siehe Abbildung 3.1.3.1)



Die Umwandlung von anwendungspezifischen Daten erfolgt mit Hilfe sogenannter Konnektoren, die Informationstrukturen in XML-Nutzdaten transformieren. Diese Konnektoren, müssen vom jeweiligen Anwendungshersteller bereitgestellt werden, damit die Anwendung mit dem EAI-System kommunizieren kann.
Diese Nutzdaten können vom EAI-System durch Integrationsserver weiterverarbeitet werden. So kann das System mit Hilfe von XLST ein XML-Datendokumentformat in ein anderes XML-Datenformat umwandeln. Desweiteren können diese XML Nutzdaten wiederum einfach in andere XML-Dokumente gekapselt werden, und so können hilfreiche Zusatzinformationen selbständig vom EAI System hinzugefügt werden. So kann das EAI-System iterativ von Netzwerkknoten zu Netzwerkknoten zusätzliche Informationen hinzufügen, die zum Beispiel Routinganweisungen beinhalten, oder ergänzende Informationen die für den eigentlichen Empfängerkonten bzw. Empfangsanwendung zur Interpretation wichtig sind.

So kann unser obiges Beispiel in Zusammenspiel mit einer Bestellannahmeapplikation durch Informationen z.B.: für eine Rechnungstelle oder/und für den Versand ergänzt werden.

```
<?xml version =”1.0”?>
<INVOICE>
        <SHIPTO>
                <KUNDE>
                        <ID>1</ID>
                        <VORNAME>Hermann</VORNAME>
                        <NAME>Müller</NAME>
                        <ADRESSE>Dawnstreet 4</ADRESSE>
                        <TELEFON>064236626</TELEFON>
                </KUNDE>
        </SHIPTO>
        <BILLTO>
                <KUNDE>
                        <ID>1</ID>
                        <VORNAME>Hermann</VORNAME>
                        <NAME>Müller</NAME>
                        <ADRESSE>Dawnstreet 4</ADRESSE>
                        <TELEFON>064236626</TELEFON>
                </KUNDE>
        </BILLTO>
        <TOTAL currency =”€”>200.50</TOTAL>
        <LINE_ITEMS>
                <LINE>
                        <BESCHREIBUNG>NVIDIA Geforce FX</BESCHREIBUNG>
                        <ARTNR>124345</ARTNR>
                </LINE>
        </LINE_ITEMS>
        <SHIPPING currency=”€”>12.50</SHIPPING>
</INVOICE>
```

Neben den hier vorgestellten Möglichkeiten, gibt es weitere verschiedene Einsatzmöglichkten von XML in EAI, die Erklärung dieser würde den Rahmen dieser Ausarbeitung sprengen.

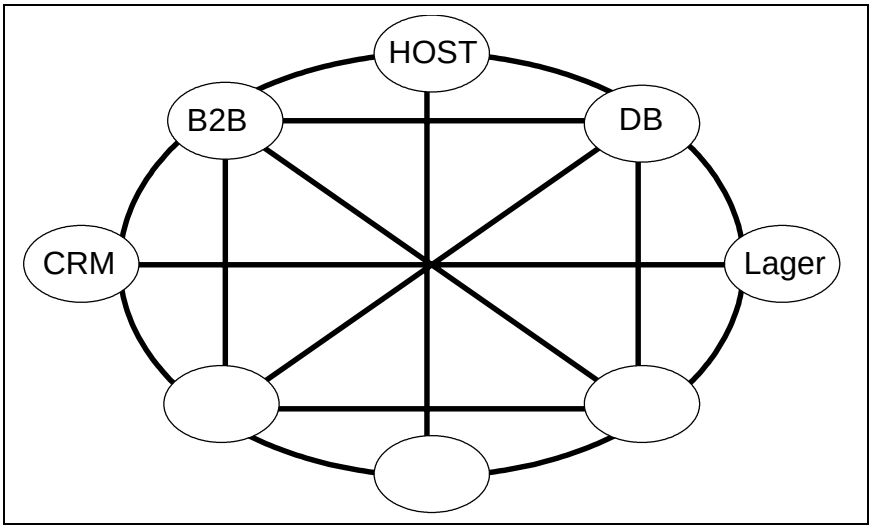
3.2 EAI-System-Ansätze

Da Anwendungsintegration nicht nur aus Anpassungen von softwaretechnischer Sicht geschehen , sondern auf Basis von geeigneter Hardware, stellt sich die Frage, wie EAI-System in der Realität strukturell aussehen, bzw. die einzelnen Komponenten verbunden sind.

Gleichzeitig wird durch die Visualisierung von verschiedenen Ansätzen deutlich, welche positiven Auswirkungen ein EAI-System auf eine firmeninterne heterogene Soft- und Hardwarelandschaft hat und man bekommt ein besseres Gefühl, wie die einzelnen Komponenten des Systems zur Zusammenarbeit bewegt werden.

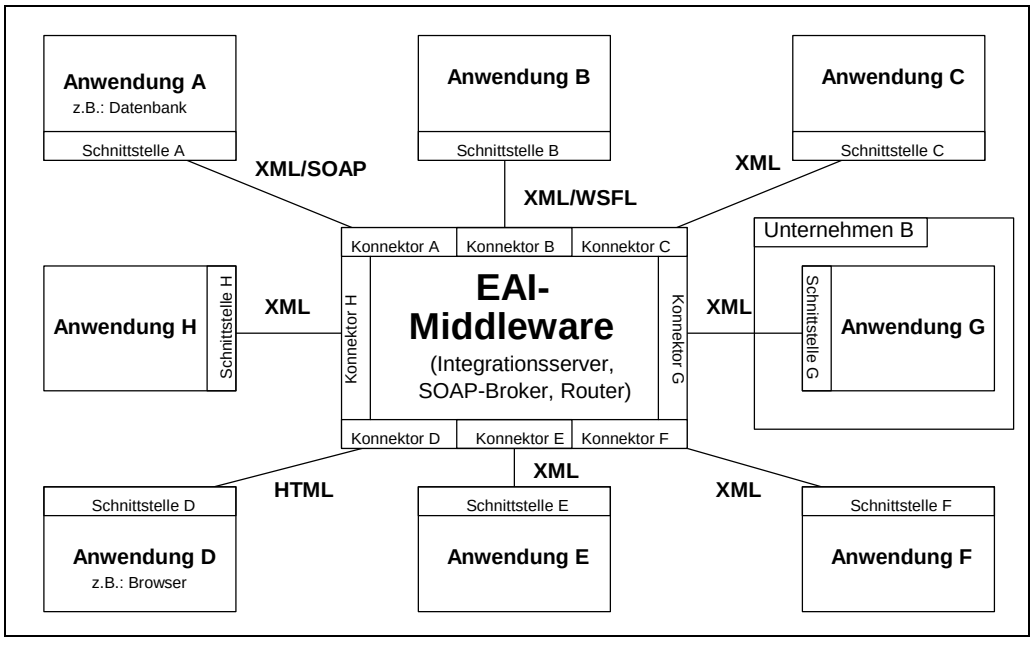
3.2.1 Point-to-Point Verbindungen

Ein negatives Beispiel einer gewachsenen Systemlandschaft bildet ein System, dessen Informationsflüsse durch Punkt-zu-Punkt-Verbindungen (engl. Point-to-Point) geschehen. Das Resultat einer solchen Kommunikationsstruktur ist mit ansteigender Kommunikationspartneranzahl sehr schnell unübersichtlich und nur sehr schwer bzw. unmöglich zu beherrschen, da bei n Kommunikationsteilnehmer n*(n-1) = n^2-n Schnittstellen zu verwalten sind.



3.2.2 HUB-Spoke- Ansatz

Ein beliebter Ansatz die Kommunikation in einem EAI-Konstrukt zu verwirklichen, ist der HUB-Spoke Ansatz. Man versucht der Schnittstellenexplosion das P2P-Ansatzes zu entkommen, indem man die Kommunikation über geeignete Middleware ablaufen lässt. Diese Integrationserver haben für jede Teilkomponente eine entsprechende Schnittstelle und wandeln, wie schon erwähnt, mit Hilfe der Konnektoren, anwendungspezifische Daten in ein EAI-systeminternes Datenformat um und je nach Erfordernis wieder in spezifische Datenformate zurück. Gleichzeitig sind sie 'Ansprechpartner' für vom Unternehmen ausgelagerte oder systemfremde Kommunikationssysteme. XML dient auch als Daten- und Objektaustauschformat.
Durch diesen Ansatz sinkt die Schnittstellenanzahl auf 2n Konnektoren.



Der Datenaustausch zwischen den einzelnen Anwendungen

So ist IBM mqSeries - Architektur ein Vertreter der HUB-und Spokearchitektur, und erweitert diese.

3.2.3 mqSeries von IBM

Die Integrationsplattform mqSeries von IBM implementiert die Kommunikation über zwei verschiedene Ansätze. Einmal die Kommunikation über MessageQueues , andererseits stellen die mqSeries Publish-Subscriberdienste zur Verfügung. Beide Ansätze werden hier kurz erklärt.
MqSeries offiziell 'WebSphere MQ family' genannt besteht aus 5 Teilkomponenten :

- Websphere MQ
- Websphere MQ Everyplace
- Websphere Adapters
- Websphere MQ Integrator Broker
- MqSeries Workflow

Die Komponente **'Websphere MQ'** realisiert Basisnachrichtenvermittlungsfunktionen, wie Message-Queue-Funktionalität, das im nachfolgendem erörtert wird. Die Komponente **'Websphere MQ Integrator Broker'** baut wie alle übrigen Komponenten auf der Basiskomponente auf, und erweitert die Basisfunktionalität um Publish-Subscriberdienste und höhere Routing-Mechanismen, die auch im Nachfolgenden erörtert werden. **'MqSeries Workflow'** ist ein 'Business-management-Flow'-System, dass der mqSeries Famile ermöglicht Geschäftsprozesse zu modellieren und diese zu integrieren. **'WebSphere Adapters'** stellen Konnektoren bereit, die es der Websphere MQ Family ermöglichen, die Kommunikation zwischen unterschiedlichsten Anwendungen bereitzustellen. **'Websphere MQ Everyplace'** ermöglicht die Anbindung von 'mobilen' -Systemen, die z.B. über Remoteverbindungen in das EAI-System eingebunden werden können. Das **Message-Queue-Prinzip** beruht auf der Idee der indirekten Kommunikation. Zwei Teilsysteme kommunizieren dabei indirekt asynchron über eine MessageQueue. Diese Queue dienst als Puffer und unterstützt Transaktionsmanagment. So wird garantiert , dass keine Transaktionen verlorengehen, sondern solange in der Queue gespeichert werden, bis ein Kommunikationpartner die in die Queue gestellte Nachricht auch konsumiert. Diese Art der Kommunikation bietet neben den ACID Eigenschaften, weitere Vorteile. So kann man grössere Flexibilität im Systemdesign erreichen , da Queues, one-to-one, one-to-many und many-to-many Kommunikation einfach realisieren. Desweiteren können Anwendungen im laufenden Betrieb runtergefahren und bei Bedarf wieder gestartet werden, ohne das Datenverlust und laufende Transaktionen unterbrochen werden müssen. Dieses bedeutet Ressourcenoptimierung. Netzwerk-Handling kann vor den Anwendungen versteckt werden, das Message-Queuesystem verwaltet die Queues und verwaltet die Kommunikationsprotokolle, die nötig sind, um in das System integrierte Anwendungen anzusprechen. Dadurch können sich wiederum Anwendungsentwickler auf die eigentliche Geschäftslogik der Anwendungen konzentrieren. Bei mqSeries von IBM kommunizieren zwei Anwendungen jeweils durch zwei unidirektionale Queues miteinander. Die eine verwaltet Anfragen vom Sender an den Empfänger, die andere wiederum

Antworten des Empfängers an den Sender. Die grösse der Messages kann variabel bestimmt werden. Die **Publish-/Subscriberfunktionalität** ist schnell erörtert. Der mqSeries-Server verwaltet Informationen darüber, welche Anwendung mit welcher kommuniziert. So tragen sich Anwendungen in eine Liste ein, die sie dazu berechtigt, Daten einer anderen Anwendung zu empfangen. Produziert diese Anwendung nun Informationen, so werden diese automatisch an alle anderen beteiligten , in diese Subscriberliste eingetragenen Anwendungen verschickt, ähnlichen einem Newsgroupsystems. Der größte Nutzen der mqSeries bleibt aber die konforme Transaktionsverwaltung. Es werden nicht einzelne Teilnachrichten gequeued, sondern gesamte Transaktionen. Dieses sollte nochmals betont werden. Die mq-Series Famile unterstützt einmal das Versenden der Nachrichten im XML Format. Dadurch wird dem MessageQueuing-Prinzip, die Vorteile (universelle Austauschbarkeit der Datenformate, etc.) dieser Technologie ermöglicht, gleichzeitig ermöglicht der mqSeries - Dienst die Transformation von proprietären, textbasierten Nachrichtenformate in XML und wieder zurück. Auch dient die mq-Series als Transformationsdienst zwischen den unterschiedlichen XML-NAchrichtentyopen, die in einem EAI-System verwendet werden können. Auch ist die behandlung von SOAP Messages vorgesehen. So kann Sie die SOAP -Nachrichten an verschiedene Anwendungsschnittstellen verschicken, so dass nicht unbedingt HTTP als Grundlage für den Versand von SOAP-Messages dienen muss.

3.3 XML-RPC

Durch XML können Remote-Procedure-Calls, das auführen Client fremder Methoden und Funktionen, die ein Server bereitstellt.

Dieses wird mit Hilfe von XML-RPC's verwirklicht.

Eine XML-RPC-Message ist ein HTTP-POST-Request. Der Körper dieses Requestes ist in XML spezifiziert. Ein durch den Request angesprochener Server führt eine Methode aus, und gibt deren Ergebnis in XML formatiert über HTTP wieder zurück. Ein Request kann wie folgt aussehen :

```
POST /RPC2 HTTP/1.0
User-Agent: Frontier/5.1.2 (WinNT)
Host: betty.userland.com
Content-Type: text/xml
Content-length: 181


<?xml version="1.0"?>
<methodCall>
   <methodName>examples.getStateName</methodName>
   <params>
      <param>
         <value><i4>41</i4></value>
         </param>
      </params>
   </methodCall>
```

Hierbei wird in dem Tag <methodCall> der Aufruf einer Methode namens getStateName initiiert, die ein einziger Parameter vom Typ32Bit-Integer mit einem Wert von '4' übergeben.

<methodCall> muss einen Methodennamen besitzen (<methodName>) und mindestens einen Parameter (<param>). Desweitern müssen User-Agent und der Host genannt werden, sowie zur Länge des Content und der ContentType auf text/xml gesetzt werden. RPC-XML definiert einfache Datentypen wie und Objektstrukturen, die nötig sind Daten auszutauschen, die für den Aufruf der spezifischen Remotemethode wichtig sind und Rückgabeparametertypen definieren.

Die Antwort vom Server wird wie folgt übermittelt:

```
HTTP/1.1 200 OK
Connection: close
Content-Length: 158
Content-Type: text/xml
Date: Fri, 17 Jul 1998 19:55:08 GMT
Server: UserLand Frontier/5.1.2-WinNT


<?xml version="1.0"?>
<methodResponse>
   <params>
      <param>
         <value><string>South Dakota</string></value>
         </param>
      </params>
   </methodResponse>
```

Hier wird gekapselt durch den Tag <methodResponse> die Antwort auf vorherigen RPC-Call gegeben. Sie besteht aus einem Parameter vom Typ String, der den von der RPC-Methode examples.getState(4) ermittelten Wert ‘South Dakota’ zurückliefet. Die erste Zeile des Headers der response informiert den Host über gelingen (200 OK) oder Nichtgelingen des RPC-Calls. Weiterhin können in der Responsenachricht , im Falle eines Fehlers, Informationen, die entsprechenden Fehlerrückgabewerte der RPC-Methode vermerkt werden.
Hierzu ein Beispiel :

```
<xml version="1.0"?>
<methodResponse>
   <fault>
      <value>
         <struct>
            <member>
               <name>faultCode</name>
               <value><int>4</int></value>
               </member>
            <member>
               <name>faultString</name>
               <value><string>Too many
parameters.</string></value>
               </member>
            </struct>
         </value>
      </fault>
   </methodResponse>
```

XML spezifiziert, folgende Datengrundtypen, die wiederum durch das Tag <struct> zu beliebig komplexeren Datentypen zusammengesetzt werden können.

Tag	Type	Example
<i4> or <int>	four-byte signed integer	-12
<boolean>	0 (false) or 1 (true)	1
<string>	ASCII string	hello world
<double>	double-precision signed floating point number	-12.214
<dateTime.iso8601>	date/time	19980717T14:08:55
<base64>	base64-encoded binary	eW91IGNhbid0IHJlYWQgdGhpcyE=

Zu beachten ist das ein Scalar jeweils durch ein zusätzliches <value>-Tag gekapselt wird. (siehe Bsp.)

Eine vollständige Spezifikation von XML-RPC findet man unter http://www.xmlrpc.com/spec

Ein zusammengesetzter Datentyp mit <struct> wird wie folgt gebildet :

```
<struct>
   <member>
      <name>lowerBound</name>
      <value><i4>18</i4></value>
      </member>
   <member>
      <name>upperBound</name>
      <value><i4>139</i4></value>
        </member>
   </struct>
```

Wobei die einzelnen Werte der einzelnen Elemente jeweils durch das <member>-Tag gekapselt werden. Das man wiederum durch ein <name-Tag bennenen kann>. <struct>-Strukturen können wiederum andere Struct Elemente beinhalten. XML-RPC bildet auch Arrays auf XML ab

:

```
<array>
   <data>
      <value><i4>12</i4></value>
      <value><string>Egypt</string></value>
      <value><boolean>0</boolean></value>
      <value><i4>-31</i4></value>
      </data>
   </array>
```

Ein wesentlicher Unterschied zum <struct>-Sprachmittel ist die Nichtbenennung der einzelnen Arrayelemente.

Der Nachteil von XML-RPC's ist, dass der beteiligten Applikation die Adresse des RPC-Servers, Methoden , deren Überparameter und Rückgabewerte bekannt sein müssen. Dieses erfordert bei einer Veränderung der 'System-Umwelt' empfindliche Eingriffe, in die Programmlogik. Eine elegantere Lösung sollen hier, die später erläuterten WebServices bieten. (siehe Kapitel 3.4 ff)

3.4 Die Rolle von XML in Web Services - UDDI - SOAP - WSDL

Es gibt viele unterschiedliche Definitionen, die versuchen das Wesen von der Technologie 'Web Services' zu charakterisieren.

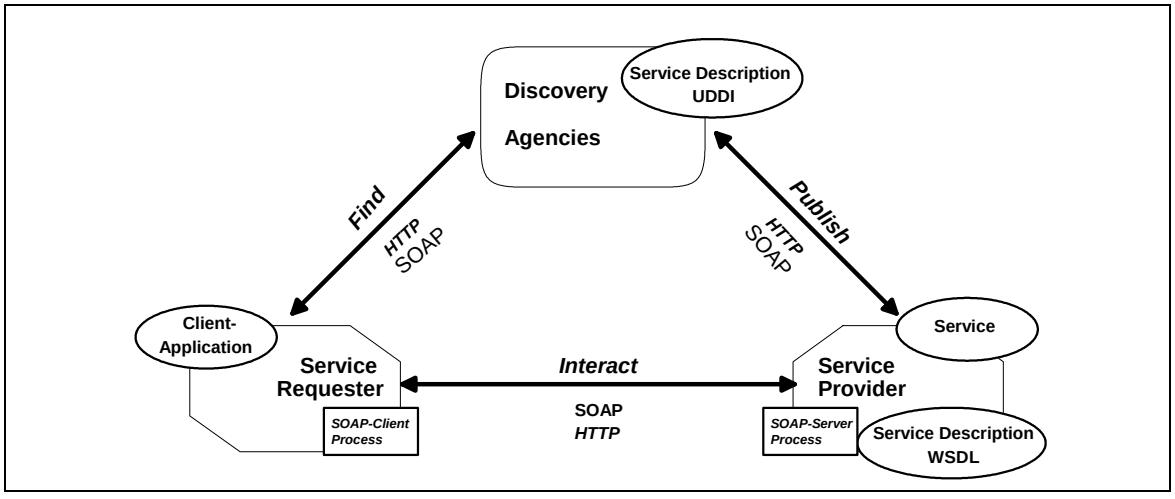
*„Web services are selfdescribing applications that can discover and engage other Web applications to complete complex tasks over the Internet. „ [*SUN]

„A Web service is a software system identified by a URI, whose public interfaces and bindings are defined and described using XML. Its definition can be discovered by other software systems. These systems may then interact with the Web service in a manner prescribed by its definition, using XML based messages conveyed by internet protocols." [W3C] http://www.w3.org/TR/2002/WD-ws-arch-20021114/#intro

Nach Definition sind Web Service sind physisch entkoppelte Softwarekomponenten, die eine Möglichkeit bereitstellen über Netzwerkprotokolle logisch zusammenzuarbeiten, um ein von einem Endbenutzer gewünschtes Ergebnis zu berechnen. In diesem Process greifen diese Softwarekomponenten auf Standards zurück, die ihre Beschreibung und Zusammenarbeit definieren und unterstützen. Diese Standards stützen zum überwiegenden Teil auf die verfügbaren Techniken, die die Datenbeschreibungssparache XML bietet.

3.4.1 Was sind Web Services ?

In UDDI (Universal Discovery und Description Initiative) , SOAP (Simple Object Acess Protocol) und WSDL (Web Services Descripton Language) sind die Repräsentationen dieser Standards. Wie funktionieren Web Services und welche Rolle spielen die Standards UDDI, SOAP und WSDL? Diese Antwort soll folgende Abbildung liefern, die das Web Service Prinzip vereinfacht darstellt :



Der Anbieter eines Service (Service Provider) stellt bei sich eine im XML-Dialekt WSDL spezifizierte Beschreibung und Spezifikation seines Web Service Angebots/Dienst öffentlich zur Verfügung. In dieser Beschreibung sind Datenformate und Messageaustauschformate spezifiziert, die zwischen Service Requester und

Service Provider gemäß dem Protokoll SOAP ausgetauscht werden. Gleichzeitig veröffentlicht der Service Provider sein Serviceangebot bei einer oder mehreren Discovery Agencies, indem er eine UDDI-Service Description in eine Registry einträgt. In dieser Description stehen Informationen , die den angebotenen Dienst (Suchdienst, etc.) kurz beschreiben, den Anbieter charakterisieren, sowie Zugriffsinformationen beinhalten, die beschreiben, wo die genauere WSDL - Beschreibung des Dienstes beim Serviceanbieter zu finden ist (zum Beispiel durch eine URL).
Der Service Requester, dessen Anwendung einen bestimmten Fremddienst nutzen will, wendet sich nun zuerst an die oder eine Discovery Agency, und bekommt von ihr entsprechende Informationen über Anbieter und Dienstaccess übermittelt, die ihm ermöglichen bei diesem Anbieter den gewünschten Dienst anzusprechen. Der Informationsaustausch zwischen Service-Requester und Discovery Agency (UDDI-Registry) kann entweder über ein HTTP-Frontend oder -im Falle einer applikationsbasierten Nachfrage über das SOAP-Protokoll erfolgen. Hat der Service-Requester entsprechende Informationen bekommen, nutzt er diese , um vom Service-Provider (Dienstanbieter) die genaue, in WSDL spezifizierte, Dienstbeschreibung zu bekommen. Diese schreibt dem Service - Requester vor wie der Informationsaustausch über SOAP erfolgt. Die WSDL Spezifikation spezifiziert Schnittstellen und Datenformate, sowie Form der SOAP - Nachrichten, über die die Kommunikation zwischen Service-Requester-Client und Service-Requester-Dienst erfolgt. Die Kommunikation zwischen Applicationsclient, auf der Requester Seite, und Dienstprozess, auf der Providerseite, erfolgt ebenfalls über das SOAP-Protokoll. Der Client schickt nun benötigte Dienstaufrufe an den vom Service Provider zur Verfügung gestellten Dienst, in dem zuvor ermittelten Übertragunsprotokoll. Diese SOAP Nachrichten werden nun über HTTP z.B. an einen SOAP Server des Providers geschickt, der sie analysiert und entsprechende Dienstaufrufe generiert, um seinen Dienst anzusprechen. Der Dienst berechnet darauf das Ergebnis der Anfrage und schickt diese wiederum an den SOAP Server, der es wieder in eine SOAP-Message umwandelt und zurück an den Service-Requester-Client schickt, der mit Hilfe eines SOAP-Clientprozesses diese Auswerten kann. Die Kommunikation zwischen Service Requester und Service-Provider wird erst dadurch erfolgreich, da sie sich über ein zuvor bekanntgegebenes Protokoll ‘unterhalten’.

3.4.2 Die Rolle vom XML-Dialekten UDDI und WSDL in Web Services

Da es in der Vergangenheit kein adäquates Mittel gab, herauszufinden, welche Organisation oder Unternehmung, welche Dienste zur Verfügung stellen, geschweige denn standardisierte Beschreibungen dieser zu ermitteln , wurde UDDI eingeführt. Das UDDI-Protokoll untersteht einem Consortium, das sich darum kümmert, UDDI einheitlich zu gestalten. Dieses Consortium heißt Organization for the Advancement of Structured Information Standards (OASIS), und wird durch viele industrielle Größen unterstützt, wie z.B.: IBM, Microsoft.. Eine vollständige Liste aller Mitglieder findet man unter http://www.oasis-open.org/about/
Zur Zeit ist das UDDI-Protokoll bei Version 3 angekommen.

UDDI sind Dienste , die es Unternehmen ermöglichen, genau dieses zu tun. Das bedeutet, dass UDDI, sowohl Möglichkeiten liefert Beschreibungen der Dienste einfach zu publizieren, als auch eine standardisierte Speicherung dieser definiert. Eine UDDI Registry enthält somit Informationen über

- Firmennnamen und Kontaktdetails, den Webseitennamen und identifizierende Nummern. (White Pages)
- Kategorisierungen von Unternehmungen. (Yellow Pages)
- Technische Daten über die Services. (Green Pages)

Diese Informationen werden mit Hilfe von XML.Dokumenten strukturiert, z.B.:

```
<serviceDetail generic="1.0" operator="Microsoft Corporation"
    truncated="false" xmlns="urn:uddi-org:api">
  <businessService businessKey="0076B468-EB27-42E5-AC09-9955CFF462A3"
                   serviceKey="D2BC296A-723B-4C45-9ED4-494F9E53F1D1">
    <name>UDDI Web Services</name>
  <description xml:lang="en">UDDI SOAP/XML message-based programmatic web
  service interfaces.</description>
  <bindingTemplates>
    <bindingTemplate bindingKey="313C2BF0-021D-405C-8149-25FD969F7F0B"
                     serviceKey="D2BC296A-723B-4C45-9ED4-494F9E53F1D1">
      <description xml:lang="en">Production UDDI server,
      Publishing interface</description>
    <accessPoint URLType="https">https://uddi.microsoft.com/publish</accessPoint>
    <tModelInstanceDetails>
      <tModelInstanceInfo tModelKey="uuid:64C756D1-3374-4E00-AE83-EE12E38FAE63">
        <description xml:lang="en">UDDI SOAP Publication Interface</description>
      </tModelInstanceInfo>
      </tModelInstanceDetails>
    </bindingTemplate>

  ........

</serviceDetail>
```

In diesem Dokument werden unterschiedliche Dienste beschrieben und die URL deren Accesspoints gekennzeichnet z.B.:

```
<accessPoint URLType="https">https://uddi.microsoft.com/publish</accessPoint>
```

Anhand dieses Beispiel kann man leicht erkennen, das UDDI eher für toolunterstützte Anfragen konzipiert ist.

Hat man einen Accesspoint gefunden, kann man unter der Angegebenen URL die genaue Spezifikation in WDSL herunterladen . (z.B. Der Dienst `Test UDDI server, Publishing interface`, am Port : **`https://test.uddi.microsoft.com/publish`**)

Eine WSDL -Datei, die den Service beschreibt, Datenaustauschformate und Messageaufbau spezifiziert...

Eine WSDL-Datei ist im Prinzip auch eine XML Datei, in ihr werden Dienstmerkmale (<documentation>) beschrieben und Datenformatspezifikationen importiert (durch <definitions .....>):

In der WSDL-Datei werden Messagetypen des Dienstes definiert und deren Datentypen festgelegt :

```
……

<message name="add_publisherAssertions">
   <part name="body" element="uddi:add_publisherAssertions" />
 </message>

……
```

Sowie den Nachrichtentypen eine Funktion zugeordnet und Ausgabe und Eingabeformate spezifiziert.

```
……

<portType name="Publish">
                <operation name="add_publisherAssertions">
                        <input message="tns:add_publisherAssertions" />
                        <output message="tns:dispositionReport" />
                        <fault name="error" message="tns:dispositionReport" />
                </operation>
……
```

Als auch die Definition von SOAP-Austauschnachrichten und deren Bindung an vorher definierte Operationen :

```
<binding name="PublishSoap" type="tns:Publish">

        <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http" />

        <operation name="add_publisherAssertions">
          <soap:operation soapAction="add_publisherAssertions" style="document" />
          <input message="tns:add_publisherAssertions">
            <soap:body use="literal" parts="body" namespace="urn:uddi-org:api_v2" />
          </input>
        </operation>
```

Die komplette WSDL_XML_Datei findet man im Anhang B.

3.4.3 Die Rolle vom XML-Dialekt SOAP in Web Services

Eine weitere XML Prozessbeschreibungssprache wurde mit Simple Object Access Protocol (SOAP) entwickelt, deren Weiterentwicklung unter Aufsicht des W3C steht. Sinn und Zweck dieser Beschreibungssprache ist es Integration auf Prozessebene zu verwirklichen, indem sie ermöglicht Nachrichten zu kapseln, die zwischen Anwendungen und Einzelsystem im EAI-Gesamtsystem ausgetauscht werden.

Dadurch verwirklicht SOAP Information Hiding Prinzipien, die existentiell für den Aufbau eines EAI-Systems sind. SOAP kann im Zusammenspiel mit sog. SOAP-Routern als Wrapper dienen, der zwischen verschiedenen anderen Kommunikationssprachen vermittelt (z.B.: CORBA, EJB, Cobol , COM).

Eine SOAP-Nachricht ist wie folgt aufgebaut.

```
POST /StockQuote HTTP/1.1
Host: www.stockquoteserver.com
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn
SOAPAction: "Some-URI"

<SOAP-ENV:Envelope
        Xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
        SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
        <SOAP-ENV:Header>
                <t:Transaction xmlns:t="some-URI" SOAP-ENV:mustUnderstand="1">
                5
                </t:Transaction>
        </SOAP-ENV:Header>
        <SOAP-ENV:Body>
                <m:GetLastTradePrice xmlns:m="Some-URI">
                        <symbol>DEF</symbol>
                </m:GetLastTradePrice>
        </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Eine SOAP Message besteht aus mehreren ineinander verschachtelten Bausteinen.
Den äußersten Baustein bildet ein SOAP-Envelope. Dieses Element kann namespace-Deklarationen enthalten, sowie Informationen über Encoding der Message oder SOAP-Versionsangaben. Der SOAP-Envlope ist zwingend in einer Message erforderlich.
Des weiteren enthält der SOAP-Envelope zwei weitere größere Elemente:

- Den optionalen SOAP-Header, eingeleitet durch `<SOAP-ENV:Header>`, abgeschlossen durch `</SOAP-ENV:Header>`
- Und den zwingend erforderlichen SOAP-Body, eingeleitet durch `<SOAP-ENV:Body>`, abgeschlossen durch `</SOAP-ENV:Body>`

Im SOAP-Header sind Informationen vermerkt, wie ein Empfänger einer SOAP -Nachricht diese zu bearbeiten hat. (Ob einige Informationen zum Verständnis der Nachricht zwingend interpretiert werden müssen oder als Zusatzinfomration verarbeitet werden können.)
Im 'Pflicht-Element' SOAP-Body hingegen, steht die eigentliche Anweisung der Nachricht bzw. der definierte Transaktion. So werden im Body Transaktionen definiert, die zum Beispiel, eine Preisanfrage durchführen. (siehe im Body-Element) Des weiteren definiert SOAP Objekte, die nötig sind spezielle Dienste auszuführen. (z.B. ein Costumer Objekt, einer Bestelltransaktion.) Da weitere Erläuterungen des SOAP-Konstruktes, den Umfang dieser Ausarbeitung sprengen würden, sei auf Quelle http://www.w3.org/TR/SOAP/#_Toc478383491 verwiesen, die sehr verständlich das Simple Object Access Protokol erklärt.

4. Abschluss

Diese Arbeit sollte Möglichkeiten aufzeigen, die es Unternehmen ermöglichen, besser und effizienter miteinander zu kommunizieren, um den Anforderungen (ökonomisches Handeln, schnelleres Erkennung von Änderungen der Marktsituation, Kosteneinsparung etc.) der Märkte gerecht zu werden . Dazu wurde im ersten Teil dieser Arbeit das Prinzip des 'Electronic Dokument Interchange' erörtert und die Probleme (Formatvielfalt, Inkompatibilitäten, schwierige und teure Systemerweiterungen etc.) dieser klassischen Art des Informationsaustausches aufgezeigt. Desweitern wurde die Rolle der Metadatensprache XML in dem EDI Umfeld beschrieben. XML ermöglicht ältere bestehende EDI Systeme weiterzunutzen, bietet aber gleichzeitig eine günstigere Anbindung an EDI-Netzwerke für Unternehmen. Da EDI nur die Aufgabe des Informationsaustausches realisiert, nicht aber die Integration und Unterstützung von Geschäfts- und Prozessabläufen, wurde im zweiten Teil dieser Ausarbeitung Grundideen, Prinzipen und Methoden der Technologie 'Enterprise Application Integration' vorgestellt, die mit Hilfe von XML ermöglicht, heterogene Systeme innerhalb eines Unternehmens zusammenzuschliessen, Datenaustausch und Kommunikation zwischen diesen zu gewährleisten und es ermöglicht Prozessvorgänge im Unternehmen optimal zu unterstützen. (z.B. Durch XML - RPC Aufrufe)
Weitergehend wurde die 'relativ junge' , Technologie 'Web Services' genannt, die aufbauend auf der Idee 'EAI', unter Zuhilfename der XML-Technologien UDDI, WSDL und SOAP , das dynamische Finden, das

Ansprechen und die Nutzung logisch getrennter Dienste ermöglichen. ‘Web Services’ erweitern damit die Idee der ‘Enterprise Applikation Integration ‘ massgeblich.

5.Literatur

1. Enterprise Application Integration with XML and Java; JP. Morgenthal, Bill La Forge,
2. United Nations Directories for Electronic Data Interchange for Administration, Commerce and Transport; Http://www.unece.org/trade/untdid/welcome.htm
3. Nachrichten des EDIFACT-Standards D01A; EDIFACTORY, http://www.edifactory.de/edifact/D01A/msglist.html
4. UN/LOCODE by UNECE; http://www.unece.org/etrades/download/downmain.htm
5. UN/EDIFACT by UNECE; http://http://www.unece.org/etrades/download/downmain.htm#edifact
6. Deutsche Telekom AG; BusinessMaiL; http://www.telekom.de/dtag/ipl1/cda/level2_a/0,,11471,00.html
7. EDI, XML, and Seeing through Transparency*;* Steven O. Kimbrough
8. DEDIG XML/EDI Demo; http://www.gefeg.com/dedigxmldemo/
9. DEDIG AG; http://www.gefeg.com/de/standard/edifact/edifact.htm
10. GEFEG; Gesellschaft für Elektronischen Geschäftsverkehr GmbH; http://www.gefeg.com/index.htm
11. Guidelines for using XML for Electronic Data Interchange; Martin Bryan; http://www.geocities.com/WallStreet/Floor/5815/guide.htm
12. XML/EDI -the E-business framework; http://www.geocities.com/WallStreet/Floor/5815/startde.htm
13. XML/EDI -the E-business framework; http://www.geocities.com/WallStreet/Floor/5815/
14. Das E-Business der Zukunft erhält eine neue Architektur; Arnold Blömer; VisionConnect GmbH
15. bDie Zukunft von EDI – XML als Grundlage für den Aufbau zwischenbetrieblicher Geschäftsprozesse; http://lwi2.wiwi.uni-frankfurt.de/forschung/dokumente/FB-99-13.pdf
16. Wissenswerters Rund um EDIFACT; EDIFACTORY; http://www.edifactory.de/
17. EDIFACT-Beispiel; http://www.edifactory.de/openhtml.php3?ref=%2Fediexam1.htm
18. Http://www.jreckfort.de/texnet/edi/xml.htm
19. Http://www.ecin.de/edi/xml/
20. Http://www.edi-information.com/
21. Http://www.dedig.de/
22. Intelligent EAI; http://www.intelligenteai.com/
23. EAI: What’s All The Excitement About?; Colin Osborne; http://www.eaijournal.com/Article.asp?ArticleID=114
24. EAI Tools Ease ERP Integration Woes; Paul Korzeniowski; http://www.eaijournal.com/Article.asp?ArticleID=153

25. Achieving the Ultimate EAI Implementation. Part 2: Message-Level Integration; Boris Lublinsky**;** http://www.eaijournal.com/Article.asp?ArticleID=299
26. Achieving the Ultimate EAI Implementation. Part 3: Data-Level Integration; Boris Lublinsky; http://www.eaijournal.com/Article.asp?ArticleID=301
27. Eai Journal; http://www.eaijournal.com/
28. Http://www.openpsa.net/eai.htm
29. EAI-Forum; http://www.eaiforum.de
30. Http://www.competence-site.de
31. Http://www2.computerwoche.de/index.cfm?pageid=259&artid=41754&main_id=41754&category=201&currpage=1
32. Http://www.objectarchitects.de/eai/index.htm?eaihome.htm
33. IBM- mqSeries; Http://www.objectarchitects.de/eai/index.htm?books.htm
34. IBM- mqSeries; Http://www.de.ibm.com/entwicklung/presse/mqseries32.html
35. IBM- mqSeries; Http://www.de.ibm.com/pressroom/1999/990208e.html
36. Vorgehensweise, Ergebnisse und Erfahrungen bei der Evaluierung einer EAI Plattform; Jürgen Dreymann, SEB IT Software GmbH
37. Evaluierung und Einführung einer EAI-Plattform; Frank Weidmann (Union IT-Services GmbH), Jörg Christ (entory AG)
38. WebServices - EAI massgeschneidert; Dipl. Inf. Oliver Voßheinrivh
39. Web Services und Softwarearchitekturen; Hawlitzek Consulting
40. E-Business für Unternehmen und öffentliche Verwaltung Konzepte und Lösungen -EAI - Enterprise Application Integration-; Institut für Informatik der Universität Zürich, Dr. Harald Häuschen, 2002
41. EAI als Baustein von eBusiness; Dipl.- Inform. Ulrich Pape, Fraunhofer ALB in Paderborn
42. PCAM GT Private Banking; Stefan Gröger, Director of Technology & Standards
43. EAI im Wandel- Wertschöpfungsnetze durch kollaborative Geschäftsprozesse-; GFT Technologies AG; Seeburger AG; Software AG; Vitria
44. Schnittstellen: Das A&O für Enterprise Application Integration; dipl. El. Ing. HTL / M. Math; Andres Koch
45. Enterprise Application Integration, Erfahrungen aus der Praxis; dpunkt.verlag; ISBN 3-89864-186-4.
46. EAI – Enterprise Application Integration - Die Pflicht vor der Kür; Thomas Winkeler, Ernst Raupach, Lothar Westphal
47. Http://www.oasis-open.org/about/
48. Http://www.oasis-open.org/committees/
49. Http://www.w3.org/TR/SOAP/#_Toc478383491

50. Web Service Description Language; http://www.w3.org/TR/wsdl/

51. Web Services Architecture - W3C Working Draft 14 November 2002 -; http://www.w3.org/TR/2002/WD-ws-arch-20021114/#intro

Anhang A

Konvertierung EDIFACT in EDI/XML als Beispiel

DEDIG

Muss Software wirklich Fehler haben
RÜSCH

Bestell AG
PURCHASE OF ALL GOODS

Bestellung

Käufer:	**ILN:** 40 04712 00000 2	**Lieferant**	**ILN:** 40 02345 00000 8
Firma:	BESTELL AG	**Firma:**	AIKA
Strasse, Nr.:	MUSTERSTR. 43	**Strasse, Nr.:**	FABRIKSTRASSE 29
PLZ, Ort:	40237 BESTELLHAUSEN	**PLZ, Ort:**	15732 KLEINSTADT
Land:	DEUTSCHLAND	**Land:**	DEUTSCHLAND
		Währung:	EUR

Artikelbezeichnung	EAN	Bestell-menge	Einheit	Einzelpreis
INKSTREAM DRUCKER ML-5000A	40 01170 00004 9	0	Stück	200
NANOTECH DRUCKER SC640	40 03652 00001 1	0	Stück	250
YAKANA MONITOR S19A	40 05381 00002 7	0	Stück	850
YOGITECH MOUSE T312B	40 09185 00013 9	0	Stück	60
HIGHTECH TV-KARTE WINTY-DVBS	40 07345 00055 5	1	Stück	350
			Summe	350

Bestellen Abbrechen

EDIFACT-Nutzdaten

(ohne UNA und UNB UNE und UNZ Flags, da uninteressant)

```
UNH+346455398+ORDERS:D:96A:UN'BGM+220+11731'DTM+137:20021230:102'NAD+SU+40 02345
 00000 8++AIKA+FABRIKSTRASSE 29+KLEINSTADT++15732+DE'NAD+BY+40 04712 00000 2++BE
STELL AG+MUSTERSTR. 43+BESTELLHAUSEN++40237+DE'CUX+2:EUR'LIN+1++40 07345 00055 5
'IMD+F++:::TV-KARTE HIGHTECH WINTV-DVBS'QTY+21:1:PCE'MOA+146:350'MOA+203:350'UNS
+S'MOA+86:350'UNT+14+346455398'
```

Konvertierung in EDIFACT in XML (biztalk Dialekt)

Die .dtd zur XML Datei

```
<?xml version="1.0" encoding='ISO-8859-1'?>

<!-- automatically generated by GEFEG EDIFIX -->
<!-- http://www.gefeg.com -->

<!ELEMENT M_ORDERS
(Bestellnummer?,Bestelldatum*,Besteller*,Lieferant*,Waehrung*,Bestellpositionen*,Gesamtbetrag*)>

<!ELEMENT Besteller (Adresse)>

<!ELEMENT Lieferant (Adresse)>

<!ELEMENT Bestellpositionen (EAN?,Artikelname*,Menge?,Einheit?,Einzelpreis*,Gesamtpreis*)>

<!ELEMENT Adresse (ILN?,Name?,StrasseNr?,Ort?,Postleitzahl?,Land?)>

<!ELEMENT Bestellnummer (#PCDATA)>

<!ELEMENT Bestelldatum (#PCDATA)>

<!ELEMENT Waehrung EMPTY>
<!ATTLIST Waehrung
        Value CDATA #FIXED "EUR">

<!ELEMENT Gesamtbetrag (#PCDATA)>

<!ELEMENT ILN (#PCDATA)>

<!ELEMENT Name (#PCDATA)>

<!ELEMENT StrasseNr (#PCDATA)>

<!ELEMENT Ort (#PCDATA)>

<!ELEMENT Postleitzahl (#PCDATA)>

<!ELEMENT Land EMPTY>
<!ATTLIST Land
        Value
(AD|AE|AF|AG|AL|AM|AO|AR|AT|AU|AZ|BA|BB|BD|BE|BG|BH|BI|BJ|BM|BO|BR|BS|BW|BY|BZ|CA|CF|CG|
```

```
CH|CI|CL|CM|CN|CO|CR|CU|CV|CY|CZ|DE|DK|DO|DZ|EC|EE|EG|ES|ET|FI|FJ|FR|GA|GB|GE|GH|GI|GM|GR|
GT|GY|HK|HN|HR|HT|HU|ID|IE|IL|IN|IQ|IR|IS|IT|JM|JO|JP|KE|KG|KH|KP|KR|KW|KZ|LB|LI|LK|LR|LS|LT|LU
|LV|LY|MA|MC|MD|MK|MR|MT|MX|MY|MZ|NA|NG|NI|NL|NO|NZ|OM|PA|PE|PG|PH|PK|PL|PR|PT|PY|QA|R
O|RU|RW|SA|SD|SE|SG|SI|SK|SM|SN|SR|SV|SY|SZ|TH|TJ|TM|TN|TR|TW|UA|UG|US|UY|UZ|VE|VN|YU|ZA|Z
M|ZR|ZW) #REQUIRED>

<!ELEMENT EAN (#PCDATA)>

<!ELEMENT Artikelname (#PCDATA)>

<!ELEMENT Menge (#PCDATA)>

<!ELEMENT Einheit EMPTY>
<!ATTLIST Einheit
          Value
(59|001|002|003|004|297|366|2N|2X|4K|4P|A25|A86|ACR|AMH|AMP|AMT|ANN|APX|ASM|ASU|BAR|BTU|C0|
CAN|CEL|CLT|CMK|CMQ|CMT|DAY|DD|DMT|DZN|EA|FAH|FOT|FTQ|GL|GLI|GM|GRM|GRO|GV|GWH|H
LT|HTZ|HUR|INH|JOU|KAH|KB|KBA|KEL|KGM|KHZ|KJO|KTM|KVT|KWH|KWT|LNE|LTR|LUX|MAL|MA
W|MFA|MGM|MHZ|MIN|MLT|MMK|MMT|MON|MTA|MTK|MTQ|MTR|MWH|NAR|NRL|ONZ|OZA|OZI|P1|
PCE|PND|PR|PTI|PTN|QAN|QTI|RPM|RTO|SEC|ST|TNE|VLT|WHR|WRD|WTT|YRD|ZE|ZP) #REQUIRED>

<!ELEMENT Einzelpreis (#PCDATA)>

<!ELEMENT Gesamtpreis (#PCDATA)>
```

Die XML-Datei zur .dtd

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<biztalk_1 xmlns="urn:biztalk-org:biztalk:biztalk_1">
   <header>
      <delivery>
         <message>
            <messageID>346455398</messageID>
            <sent>30.12.02 16:00:06</sent>
            <subject>Purchase Order</subject>
         </message>
         <to>
            <address>http://www.AIKA.com/</address>
         </to>
         <from>
            <address>http://www.bestell-ag.de</address>
         </from>
      </delivery>
      <manifest>
         <document>
            <name>PO</name>
            <description>Purchase Order</description>
         </document>
      </manifest>
   </header>
   <body>

    <!-- automatically generated by GEFEG EDIFIX -->
    <!-- http://www.gefeg.com -->

    <!DOCTYPE M_ORDERS SYSTEM "orders.dtd">
    <?xml-stylesheet type="text/xsl" href="orders.xsl"?>
```

```
    <M_ORDERS  xmlns="urn:biztalk-org:biztalk:M_ORDERS">
      <Bestellnummer>101731</Bestellnummer>
      <Bestelldatum>30.12.02 15:56:40</Bestelldatum>
      <Waehrung Value="EUR"/>
      <Besteller>
        <Adresse>
          <ILN>40 04712 00000 2</ILN>
          <Name>BESTELL AG</Name>
          <StrasseNr>Musterstraße 43</StrasseNr>
          <Ort>Bestellhausen</Ort>
          <Postleitzahl>40237</Postleitzahl>
          <Land Value="DE"/>
        </Adresse>
      </Besteller>
      <Lieferant>
        <Adresse>
          <ILN>40 02345 00000 8</ILN>
          <Name>AIKA</Name>
          <StrasseNr>FABRIKSTRASSE 29</StrasseNr>
          <Ort>KLEINSTADT</Ort>
          <Postleitzahl>15732</Postleitzahl>
          <Land Value="DE"/>
        </Adresse>
      </Lieferant>
      <Bestellpositionen>
        <EAN>40 07345 00055 5</EAN>
        <Artikelname>HIGHTECH TV-KARTE WINTV-DVBS</Artikelname>
        <Menge>1</Menge>
        <Einheit Value="PCE"/>
        <Einzelpreis>350</Einzelpreis>
        <Gesamtpreis>350</Gesamtpreis>
      </Bestellpositionen>
      <Gesamtbetrag>350</Gesamtbetrag>
    </M_ORDERS>
  </body>
</biztalk_1>
```

XSL-Stylsheet zur Beschreibung der Ansicht

```
<?xml version='1.0' encoding='ISO-8859-1'?>

 <!-- automatically generated by GEFEG EDIFIX -->
 <!-- http://www.gefeg.com -->

 <!DOCTYPE xsl:stylesheet [
  <!ENTITY nbsp   " ">
  <!ENTITY Auml   "&#196;">
  <!ENTITY Ouml   "&#214;">
  <!ENTITY Uuml   "&#220;">
  <!ENTITY szlig  "&#223;">
  <!ENTITY auml   "&#228;">
  <!ENTITY ouml   "&#246;">
  <!ENTITY uuml   "&#252;">
 ]>
 <xsl:stylesheet xmlns:xsl="uri:xsl">
  <xsl:template match="/">
   <HTML>
   <HEAD>
   <TITLE>EDIFIX Report</TITLE>
   <STYLE type="text/css">
    #f0 { font: bold 12pt "Verdana"; }
    #f1 { font: bold 9pt "Verdana"; }
    #f2 { font: 9pt "Verdana"; }
    #f3 { font: bold 8pt "Verdana"; }
```

```
      #f4 { font: 8pt "Verdana"; }
      #f5 { font: 8pt "Arial"; }
      #b0 {border-width:1.5pt;border-style:double;border-color:#000000; }
      #b1 {border-color:#000000; }
      #b2 {border-left-width:0.8pt;border-top-width:0.8pt;border-left-style:solid;border-top-style:solid;border-color:#000000; }
      #b3
{border-left-width:0.8pt;border-top-width:0.8pt;border-right-width:0.8pt;border-left-style:solid;border-top-style:solid;border-right-style:soli
d;border-color:#000000; }
      #b4
{border-left-width:0.8pt;border-top-width:0.8pt;border-bottom-width:0.8pt;border-left-style:solid;border-top-style:solid;border-bottom-style
:solid;border-color:#000000; }
      #b5 {border-width:0.8pt;border-style:solid;border-color:#000000; }
      #b6
{border-top-width:0.8pt;border-right-width:0.8pt;border-bottom-width:0.8pt;border-top-style:solid;border-right-style:solid;border-bottom-st
yle:solid;border-color:#000000; }
      #b7 {border-right-width:0.8pt;border-bottom-width:0.8pt;border-right-style:solid;border-bottom-style:solid;border-color:#000000; }
    </STYLE>
    <script language="JavaScript">
     function setTopAd(adPage) {
      parent.BannerOben.location.replace(adPage);
     }
     function setBottomAd(adPage) {
      parent.BannerUnten.location.replace(adPage);
     }
    </script>
   </HEAD>
    <BODY BGCOLOR="#FFFFFF" onload="setTopAd('adverts_oben7.htm')">
      <TABLE  WIDTH="100%" CELLSPACING="0" CELLPADDING="0" BORDER="0">
       <TR>
        <TD VALIGN="TOP" WIDTH="100%" ID="b1"><IMG SRC="bestellag.gif" WIDTH="174" HEIGHT="58"
BORDER="0"/></TD>
       </TR>
       <TR>
        <TD VALIGN="TOP" WIDTH="100%" ID="b1" NOWRAP = "nowrap"><FONT  COLOR="#008000"><SPAN
ID="f0">Bestellung</SPAN></FONT></TD>
       </TR>
      </TABLE>
      <TABLE  WIDTH="100%" CELLSPACING="0" CELLPADDING="0" BORDER="0">
       <TR>
        <TD VALIGN="TOP" WIDTH="17%" ID="b2">
        <TABLE WIDTH="100%" BORDER="0" CELLSPACING="0" CELLPADDING="0">
         <TR>
          <TD WIDTH="3%"> </TD>
          <TD WIDTH="94%" NOWRAP = "nowrap" ID="f1">K&auml;ufer:</TD>
          <TD WIDTH="2%"> </TD>
         </TR>
        </TABLE>
        </TD>
        <TD VALIGN="TOP" WIDTH="6%" ID="b2">
        <TABLE WIDTH="100%" BORDER="0" CELLSPACING="0" CELLPADDING="0">
         <TR>
          <TD WIDTH="9%"> </TD>
          <TD WIDTH="90%" NOWRAP = "nowrap" ID="f1">ILN:</TD>
         </TR>
        </TABLE>
        </TD>
        <TD VALIGN="TOP" WIDTH="26%" COLSPAN="3" ID="b3">
        <xsl:choose>
         <xsl:when test="M_ORDERS/Besteller/Adresse/ILN[0]">
          <TABLE WIDTH="100%" BORDER="0" CELLSPACING="0" CELLPADDING="0">
           <TR VALIGN="TOP">
            <TD WIDTH="2%"> </TD>
            <TD WIDTH="97%" ID="f2"><xsl:value-of select="M_ORDERS/Besteller/Adresse/ILN"/></TD>
           </TR>
          </TABLE>
         </xsl:when>
         <xsl:otherwise>
           
         </xsl:otherwise>
        </xsl:choose>
        </TD>
        <TD VALIGN="TOP" WIDTH="5%" ID="b1"> </TD>
        <TD VALIGN="TOP" WIDTH="17%" COLSPAN="2" ID="b2">
```

```
<TABLE WIDTH="100%" BORDER="0" CELLSPACING="0" CELLPADDING="0">
 <TR>
  <TD WIDTH="3%"> </TD>
  <TD WIDTH="94%" NOWRAP = "nowrap" ID="f1">Lieferant:</TD>
  <TD WIDTH="2%"> </TD>
 </TR>
</TABLE>
</TD>
<TD VALIGN="TOP" WIDTH="5%" ID="b2">
<TABLE WIDTH="100%" BORDER="0" CELLSPACING="0" CELLPADDING="0">
 <TR>
  <TD WIDTH="9%"> </TD>
  <TD WIDTH="90%" NOWRAP = "nowrap" ID="f1">ILN:</TD>
 </TR>
</TABLE>
</TD>
<TD VALIGN="TOP" WIDTH="24%" COLSPAN="3" ID="b3">
<xsl:choose>
 <xsl:when test="M_ORDERS/Lieferant/Adresse/ILN[0]">
  <TABLE WIDTH="100%" BORDER="0" CELLSPACING="0" CELLPADDING="0">
   <TR VALIGN="TOP">
    <TD WIDTH="2%"> </TD>
    <TD WIDTH="98%" ID="f2"><xsl:value-of select="M_ORDERS/Lieferant/Adresse/ILN"/></TD>
   </TR>
  </TABLE>
 </xsl:when>
 <xsl:otherwise>
   
 </xsl:otherwise>
</xsl:choose>
</TD>
</TR>
<TR>
 <TD VALIGN="TOP" WIDTH="17%" ID="b2">
 <TABLE WIDTH="100%" BORDER="0" CELLSPACING="0" CELLPADDING="0">
  <TR>
   <TD WIDTH="3%"> </TD>
   <TD WIDTH="94%" NOWRAP = "nowrap" ID="f1">Firma:</TD>
   <TD WIDTH="2%"> </TD>
  </TR>
 </TABLE>
 </TD>
 <TD VALIGN="TOP" WIDTH="32%" COLSPAN="4" ID="b3">
 <xsl:choose>
  <xsl:when test="M_ORDERS/Besteller/Adresse/Name[0]">
   <TABLE WIDTH="100%" BORDER="0" CELLSPACING="0" CELLPADDING="0">
    <xsl:if test="M_ORDERS/Besteller/Adresse/Name[0]">
     <TR VALIGN="TOP">
      <TD WIDTH="1%"> </TD>
      <TD WIDTH="98%" ID="f2"><xsl:value-of select="M_ORDERS/Besteller/Adresse/Name"/></TD>
     </TR>
    </xsl:if>
   </TABLE>
  </xsl:when>
  <xsl:otherwise>
    
  </xsl:otherwise>
 </xsl:choose>
 </TD>
 <TD VALIGN="TOP" WIDTH="5%" ID="b1"> </TD>
 <TD VALIGN="TOP" WIDTH="17%" COLSPAN="2" ID="b2">
 <TABLE WIDTH="100%" BORDER="0" CELLSPACING="0" CELLPADDING="0">
  <TR>
   <TD WIDTH="3%"> </TD>
   <TD WIDTH="94%" NOWRAP = "nowrap" ID="f1">Firma:</TD>
   <TD WIDTH="2%"> </TD>
  </TR>
 </TABLE>
 </TD>
 <TD VALIGN="TOP" WIDTH="29%" COLSPAN="4" ID="b3">
 <xsl:choose>
  <xsl:when test="M_ORDERS/Lieferant/Adresse/Name[0]">
   <TABLE WIDTH="100%" BORDER="0" CELLSPACING="0" CELLPADDING="0">
```

```
      <xsl:if test="M_ORDERS/Lieferant/Adresse/Name[0]">
       <TR VALIGN="TOP">
        <TD WIDTH="1%"> </TD>
        <TD WIDTH="98%" ID="f2"><xsl:value-of select="M_ORDERS/Lieferant/Adresse/Name"/></TD>
       </TR>
      </xsl:if>
     </TABLE>
    </xsl:when>
    <xsl:otherwise>
      
    </xsl:otherwise>
   </xsl:choose>
   </TD>
  </TR>
  <TR>
   <TD VALIGN="TOP" WIDTH="17%" ID="b2">
   <TABLE WIDTH="100%" BORDER="0" CELLSPACING="0" CELLPADDING="0">
    <TR>
     <TD WIDTH="3%"> </TD>
     <TD WIDTH="94%" NOWRAP = "nowrap" ID="f1">Stra&szlig;e, Nr:</TD>
     <TD WIDTH="2%"> </TD>
    </TR>
   </TABLE>
   </TD>
   <TD VALIGN="TOP" WIDTH="32%" COLSPAN="4" ID="b3">
   <xsl:choose>
    <xsl:when test="M_ORDERS/Besteller/Adresse/StrasseNr[0]">
     <TABLE WIDTH="100%" BORDER="0" CELLSPACING="0" CELLPADDING="0">
      <TR VALIGN="TOP">
       <TD WIDTH="1%"> </TD>
       <TD WIDTH="98%" ID="f2"><xsl:value-of select="M_ORDERS/Besteller/Adresse/StrasseNr"/></TD>
      </TR>
     </TABLE>
    </xsl:when>
    <xsl:otherwise>
      
    </xsl:otherwise>
   </xsl:choose>
   </TD>
   <TD VALIGN="TOP" WIDTH="5%" ID="b1"> </TD>
   <TD VALIGN="TOP" WIDTH="17%" COLSPAN="2" ID="b2">
   <TABLE WIDTH="100%" BORDER="0" CELLSPACING="0" CELLPADDING="0">
    <TR>
     <TD WIDTH="3%"> </TD>
     <TD WIDTH="94%" NOWRAP = "nowrap" ID="f1">Stra&szlig;e, Nr:</TD>
     <TD WIDTH="2%"> </TD>
    </TR>
   </TABLE>
   </TD>
   <TD VALIGN="TOP" WIDTH="29%" COLSPAN="4" ID="b3">
   <xsl:choose>
    <xsl:when test="M_ORDERS/Lieferant/Adresse/StrasseNr[0]">
     <TABLE WIDTH="100%" BORDER="0" CELLSPACING="0" CELLPADDING="0">
      <TR VALIGN="TOP">
       <TD WIDTH="1%"> </TD>
       <TD WIDTH="97%" ID="f2"><xsl:value-of select="M_ORDERS/Lieferant/Adresse/StrasseNr"/></TD>
       <TD WIDTH="1%"> </TD>
      </TR>
     </TABLE>
    </xsl:when>
    <xsl:otherwise>
      
    </xsl:otherwise>
   </xsl:choose>
   </TD>
  </TR>
  <TR>
   <TD VALIGN="TOP" WIDTH="17%" ID="b2">
   <TABLE WIDTH="100%" BORDER="0" CELLSPACING="0" CELLPADDING="0">
    <TR>
     <TD WIDTH="3%"> </TD>
     <TD WIDTH="94%" NOWRAP = "nowrap" ID="f1">PLZ, Ort:</TD>
     <TD WIDTH="2%"> </TD>
```

```
          </TR>
         </TABLE>
         </TD>
         <TD VALIGN="TOP" WIDTH="32%" COLSPAN="4" ID="b3">
         <xsl:choose>
          <xsl:when test="(M_ORDERS/Besteller/Adresse/Postleitzahl|M_ORDERS/Besteller/Adresse/Ort)[0]">
           <TABLE WIDTH="100%" BORDER="0" CELLSPACING="0" CELLPADDING="0">
            <TR VALIGN="TOP">
             <TD WIDTH="1%"> </TD>
             <TD WIDTH="20%" ID="f2"><xsl:choose><xsl:when test="M_ORDERS/Besteller/Adresse/Postleitzahl[0]"><xsl:value-of
select="M_ORDERS/Besteller/Adresse/Postleitzahl"/></xsl:when><xsl:otherwise> </xsl:otherwise></xsl:choose></TD>
             <TD WIDTH="78%" ID="f2"><xsl:choose><xsl:when test="M_ORDERS/Besteller/Adresse/Ort[0]"><xsl:value-of
select="M_ORDERS/Besteller/Adresse/Ort"/></xsl:when><xsl:otherwise> </xsl:otherwise></xsl:choose></TD>
            </TR>
           </TABLE>
          </xsl:when>
          <xsl:otherwise>
            
          </xsl:otherwise>
         </xsl:choose>
         </TD>
         <TD VALIGN="TOP" WIDTH="5%" ID="b1"> </TD>
         <TD VALIGN="TOP" WIDTH="17%" COLSPAN="2" ID="b2">
         <TABLE WIDTH="100%" BORDER="0" CELLSPACING="0" CELLPADDING="0">
          <TR>
           <TD WIDTH="3%"> </TD>
           <TD WIDTH="94%" NOWRAP = "nowrap" ID="f1">PLZ, Ort:</TD>
           <TD WIDTH="2%"> </TD>
          </TR>
         </TABLE>
         </TD>
         <TD VALIGN="TOP" WIDTH="29%" COLSPAN="4" ID="b3">
         <xsl:choose>
          <xsl:when test="(M_ORDERS/Lieferant/Adresse/Postleitzahl|M_ORDERS/Lieferant/Adresse/Ort)[0]">
           <TABLE WIDTH="100%" BORDER="0" CELLSPACING="0" CELLPADDING="0">
            <TR VALIGN="TOP">
             <TD WIDTH="1%"> </TD>
             <TD WIDTH="20%" ID="f2"><xsl:choose><xsl:when test="M_ORDERS/Lieferant/Adresse/Postleitzahl[0]"><xsl:value-of
select="M_ORDERS/Lieferant/Adresse/Postleitzahl"/></xsl:when><xsl:otherwise> </xsl:otherwise></xsl:choose></TD>
             <TD WIDTH="78%" ID="f2"><xsl:choose><xsl:when test="M_ORDERS/Lieferant/Adresse/Ort[0]"><xsl:value-of
select="M_ORDERS/Lieferant/Adresse/Ort"/></xsl:when><xsl:otherwise> </xsl:otherwise></xsl:choose></TD>
            </TR>
           </TABLE>
          </xsl:when>
          <xsl:otherwise>
            
          </xsl:otherwise>
         </xsl:choose>
         </TD>
        </TR>
        <TR>
         <TD VALIGN="TOP" WIDTH="17%" ID="b4">
         <TABLE WIDTH="100%" BORDER="0" CELLSPACING="0" CELLPADDING="0">
          <TR>
           <TD WIDTH="3%"> </TD>
           <TD WIDTH="94%" NOWRAP = "nowrap" ID="f1">Land:</TD>
           <TD WIDTH="2%"> </TD>
          </TR>
         </TABLE>
         </TD>
         <TD VALIGN="TOP" WIDTH="32%" COLSPAN="4" ID="b5">
         <TABLE WIDTH="100%" BORDER="0" CELLSPACING="0" CELLPADDING="0">
          <TR>
           <TD WIDTH="1%"> </TD>
           <TD WIDTH="98%" NOWRAP = "nowrap" ID="f2"><xsl:choose><xsl:when
test="M_ORDERS/Besteller/Adresse/Land/@Value[0]">DEUTSCHLAND</xsl:when><xsl:otherwise> </xsl:otherwise></xsl:choos
e></TD>
          </TR>
         </TABLE>
         </TD>
         <TD VALIGN="TOP" WIDTH="5%" ID="b1"> </TD>
         <TD VALIGN="TOP" WIDTH="17%" COLSPAN="2" ID="b4">
         <TABLE WIDTH="100%" BORDER="0" CELLSPACING="0" CELLPADDING="0">
```

```
            <TR>
              <TD WIDTH="3%"> </TD>
              <TD WIDTH="94%" NOWRAP = "nowrap" ID="f1">Land:</TD>
              <TD WIDTH="2%"> </TD>
            </TR>
           </TABLE>
           </TD>
           <TD VALIGN="TOP" WIDTH="29%" COLSPAN="4" ID="b5">
           <TABLE WIDTH="100%" BORDER="0" CELLSPACING="0" CELLPADDING="0">
            <TR>
              <TD WIDTH="1%"> </TD>
              <TD WIDTH="97%" NOWRAP = "nowrap" ID="f2"><xsl:choose><xsl:when
test="M_ORDERS/Lieferant/Adresse/Land/@Value[0]">DEUTSCHLAND</xsl:when><xsl:otherwise> </xsl:otherwise></xsl:choos
e></TD>
              <TD WIDTH="1%"> </TD>
            </TR>
           </TABLE>
           </TD>
          </TR>
          <TR>
           <TD VALIGN="TOP" WIDTH="100%" COLSPAN="12" ID="b1"> </TD>
          </TR>
          <TR>
           <TD VALIGN="TOP" WIDTH="17%" ID="b4">
           <TABLE WIDTH="100%" BORDER="0" CELLSPACING="0" CELLPADDING="0">
            <TR>
              <TD WIDTH="3%"> </TD>
              <TD WIDTH="94%" NOWRAP = "nowrap" ID="f1">Bestellnummer:</TD>
              <TD WIDTH="2%"> </TD>
            </TR>
           </TABLE>
           </TD>
           <TD VALIGN="TOP" WIDTH="11%" COLSPAN="2" ID="b4">
           <xsl:choose>
            <xsl:when test="M_ORDERS/Bestellnummer[0]">
              <TABLE WIDTH="100%" BORDER="0" CELLSPACING="0" CELLPADDING="0">
               <TR VALIGN="TOP">
                 <TD WIDTH="5%"> </TD>
                 <TD WIDTH="89%" ID="f2"><xsl:value-of select="M_ORDERS/Bestellnummer"/></TD>
                 <TD WIDTH="5%"> </TD>
               </TR>
              </TABLE>
            </xsl:when>
            <xsl:otherwise>
              
            </xsl:otherwise>
           </xsl:choose>
           </TD>
           <TD VALIGN="TOP" WIDTH="6%" ID="b4" NOWRAP = "nowrap"><SPAN  ID="f1">Vom:</SPAN></TD>
           <TD VALIGN="TOP" WIDTH="15%" ID="b5">
           <xsl:choose>
            <xsl:when test="M_ORDERS/Bestelldatum[0]">
              <TABLE WIDTH="100%" BORDER="0" CELLSPACING="0" CELLPADDING="0">
               <TR VALIGN="TOP">
                 <TD WIDTH="4%"> </TD>
                 <TD WIDTH="96%" ID="f2"><xsl:value-of select="M_ORDERS/Bestelldatum"/></TD>
               </TR>
              </TABLE>
            </xsl:when>
            <xsl:otherwise>
              
            </xsl:otherwise>
           </xsl:choose>
           </TD>
           <TD VALIGN="TOP" WIDTH="5%" ID="b1"> </TD>
           <TD VALIGN="TOP" WIDTH="17%" COLSPAN="2" ID="b4">
           <TABLE WIDTH="100%" BORDER="0" CELLSPACING="0" CELLPADDING="0">
            <TR>
              <TD WIDTH="3%"> </TD>
              <TD WIDTH="94%" NOWRAP = "nowrap" ID="f1">W&auml;hrung:</TD>
              <TD WIDTH="2%"> </TD>
            </TR>
           </TABLE>
```

```
 </TD>
 <TD VALIGN="TOP" WIDTH="29%" COLSPAN="4" ID="b5">
 <xsl:choose>
  <xsl:when test="M_ORDERS/Waehrung/@Value[0]">
   <TABLE WIDTH="100%" BORDER="0" CELLSPACING="0" CELLPADDING="0">
    <TR VALIGN="TOP">
     <TD WIDTH="1%"> </TD>
     <TD WIDTH="97%" ID="f2"><xsl:value-of select="M_ORDERS/Waehrung/@Value"/></TD>
     <TD WIDTH="1%"> </TD>
    </TR>
   </TABLE>
  </xsl:when>
  <xsl:otherwise>
    
  </xsl:otherwise>
 </xsl:choose>
 </TD>
</TR>
<TR>
 <TD VALIGN="TOP" WIDTH="100%" COLSPAN="12" ID="b1"> </TD>
</TR>
<TR>
 <TD VALIGN="TOP" WIDTH="28%" COLSPAN="3" ID="b2">
 <TABLE WIDTH="100%" BORDER="0" CELLSPACING="0" CELLPADDING="0">
  <TR VALIGN="TOP">
   <TD WIDTH="2%"> </TD>
   <TD WIDTH="97%" ID="f3">Artikelbezeichnung</TD>
   <TD WIDTH="1%"> </TD>
  </TR>
 </TABLE>
 </TD>
 <TD VALIGN="TOP" WIDTH="21%" COLSPAN="2" ID="b2">
 <TABLE WIDTH="100%" BORDER="0" CELLSPACING="0" CELLPADDING="0">
  <TR VALIGN="TOP">
   <TD WIDTH="2%"> </TD>
   <TD WIDTH="96%" ID="f3">EAN</TD>
   <TD WIDTH="1%"> </TD>
  </TR>
 </TABLE>
 </TD>
 <TD VALIGN="TOP" WIDTH="18%" COLSPAN="2" ID="b2">
 <TABLE WIDTH="100%" BORDER="0" CELLSPACING="0" CELLPADDING="0">
  <TR VALIGN="TOP">
   <TD WIDTH="3%"> </TD>
   <TD WIDTH="82%" ID="f3">Bestellmenge</TD>
   <TD WIDTH="14%"> </TD>
  </TR>
 </TABLE>
 </TD>
 <TD VALIGN="TOP" WIDTH="9%" COLSPAN="2" ID="b2">
 <TABLE WIDTH="100%" BORDER="0" CELLSPACING="0" CELLPADDING="0">
  <TR VALIGN="TOP">
   <TD WIDTH="6%"> </TD>
   <TD WIDTH="80%" ID="f3">Einheit</TD>
   <TD WIDTH="13%"> </TD>
  </TR>
 </TABLE>
 </TD>
 <TD VALIGN="TOP" WIDTH="12%" COLSPAN="2" ID="b3">
 <TABLE WIDTH="100%" BORDER="0" CELLSPACING="0" CELLPADDING="0">
  <TR VALIGN="TOP">
   <TD WIDTH="96%" ALIGN="RIGHT" ID="f3">Einzelpreis</TD>
   <TD WIDTH="4%"> </TD>
  </TR>
 </TABLE>
 </TD>
 <TD VALIGN="TOP" WIDTH="12%" ID="b6">
 <TABLE WIDTH="100%" BORDER="0" CELLSPACING="0" CELLPADDING="0">
  <TR VALIGN="TOP">
   <TD WIDTH="94%" ALIGN="RIGHT" ID="f3">Gesamtpreis</TD>
   <TD WIDTH="5%"> </TD>
  </TR>
 </TABLE>
```

```
 </TD>
</TR>
<xsl:for-each select="M_ORDERS/Bestellpositionen">
  <TR>
   <TD VALIGN="TOP" WIDTH="28%" COLSPAN="3" ID="b4">
   <xsl:choose>
    <xsl:when test="Artikelname[0]">
     <TABLE WIDTH="100%" BORDER="0" CELLSPACING="0" CELLPADDING="0">
      <TR VALIGN="TOP">
       <TD WIDTH="2%"> </TD>
       <TD WIDTH="97%" ID="f4"><xsl:value-of select="Artikelname"/></TD>
       <TD WIDTH="1%"> </TD>
      </TR>
     </TABLE>
    </xsl:when>
    <xsl:otherwise>
      
    </xsl:otherwise>
   </xsl:choose>
   </TD>
   <TD VALIGN="TOP" WIDTH="21%" COLSPAN="2" ID="b4">
   <xsl:choose>
    <xsl:when test="EAN[0]">
     <TABLE WIDTH="100%" BORDER="0" CELLSPACING="0" CELLPADDING="0">
      <TR VALIGN="TOP">
       <TD WIDTH="4%"> </TD>
       <TD WIDTH="96%" ID="f4"><xsl:value-of select="EAN"/></TD>
      </TR>
     </TABLE>
    </xsl:when>
    <xsl:otherwise>
      
    </xsl:otherwise>
   </xsl:choose>
   </TD>
   <TD VALIGN="TOP" WIDTH="18%" COLSPAN="2" ID="b4">
   <xsl:choose>
    <xsl:when test="Menge[0]">
     <TABLE WIDTH="100%" BORDER="0" CELLSPACING="0" CELLPADDING="0">
      <TR VALIGN="TOP">
       <TD WIDTH="5%"> </TD>
       <TD WIDTH="94%" ID="f4"><xsl:value-of select="Menge"/></TD>
      </TR>
     </TABLE>
    </xsl:when>
    <xsl:otherwise>
      
    </xsl:otherwise>
   </xsl:choose>
   </TD>
   <TD VALIGN="TOP" WIDTH="9%" COLSPAN="2" ID="b4">
   <TABLE WIDTH="100%" BORDER="0" CELLSPACING="0" CELLPADDING="0">
    <TR>
     <TD WIDTH="11%"> </TD>
     <TD WIDTH="73%" NOWRAP = "nowrap" ID="f4"><xsl:choose><xsl:when
test="Einheit/@Value[0]">St&uuml;ck</xsl:when><xsl:otherwise> </xsl:otherwise></xsl:choose></TD>
     <TD WIDTH="15%"> </TD>
    </TR>
   </TABLE>
   </TD>
   <TD VALIGN="TOP" WIDTH="12%" COLSPAN="2" ID="b5">
   <xsl:choose>
    <xsl:when test="Einzelpreis[0]">
     <TABLE WIDTH="100%" BORDER="0" CELLSPACING="0" CELLPADDING="0">
      <TR VALIGN="TOP">
       <TD WIDTH="96%" ALIGN="RIGHT" ID="f4"><xsl:value-of select="Einzelpreis"/></TD>
       <TD WIDTH="4%"> </TD>
      </TR>
     </TABLE>
    </xsl:when>
    <xsl:otherwise>
      
    </xsl:otherwise>
```

```
          </xsl:choose>
          </TD>
          <TD VALIGN="TOP" WIDTH="12%" ID="b7">
          <xsl:choose>
            <xsl:when test="Gesamtpreis[0]">
              <TABLE WIDTH="100%" BORDER="0" CELLSPACING="0" CELLPADDING="0">
                <TR VALIGN="TOP">
                  <TD WIDTH="94%" ALIGN="RIGHT" ID="f4"><xsl:value-of select="Gesamtpreis"/></TD>
                  <TD WIDTH="5%"> </TD>
                </TR>
              </TABLE>
            </xsl:when>
            <xsl:otherwise>
               
            </xsl:otherwise>
          </xsl:choose>
          </TD>
        </TR>
      </xsl:for-each>
      <TR>
        <TD VALIGN="TOP" WIDTH="83%" COLSPAN="10" ID="b4">
        <TABLE WIDTH="100%" BORDER="0" CELLSPACING="0" CELLPADDING="0">
          <TR VALIGN="TOP">
            <TD WIDTH="80%"> </TD>
            <TD WIDTH="19%" ID="f3">Gesamtbetrag</TD>
          </TR>
        </TABLE>
        </TD>
        <TD VALIGN="TOP" WIDTH="17%" COLSPAN="2" ID="b5">
        <xsl:choose>
          <xsl:when test="M_ORDERS/Gesamtbetrag[0]">
            <TABLE WIDTH="100%" BORDER="0" CELLSPACING="0" CELLPADDING="0">
              <TR VALIGN="TOP">
                <TD WIDTH="97%" ALIGN="RIGHT" ID="f4"><xsl:value-of select="M_ORDERS/Gesamtbetrag"/></TD>
                <TD WIDTH="2%"> </TD>
              </TR>
            </TABLE>
          </xsl:when>
          <xsl:otherwise>
             
          </xsl:otherwise>
        </xsl:choose>
        </TD>
      </TR>
    </TABLE>
    <TABLE  WIDTH="100%" CELLSPACING="0" CELLPADDING="0" BORDER="0">
    </TABLE>
  </BODY>
  </HTML>
 </xsl:template>
</xsl:stylesheet>
```

Wer genauere Information bzw. XML/EDI in Action sehen will, sei auf www.dedig.de verwiesen, die eine wirklich anschauliche Demo auf ihrer Seite haben.

Anhang B

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions
 name="UDDI_Publication_API_V2"
 targetNamespace="urn:uddi-org:publication_v2"
 xmlns="http://schemas.xmlsoap.org/wsdl/"
 xmlns:tns="urn:uddi-org:publication_v2"
 xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
 xmlns:uddi="urn:uddi-org:api_v2">

        <documentation>
                WSDL Service Interface for UDDI Publication API V2.0

                This WSDL document defines the publication API calls for interacting with
                the UDDI registry.  The complete UDDI API specification is available
                at http://www.uddi.org/specification.html.

                Author: Peter Brittenham, IBM Corporation, peterbr@us.ibm.com
                Author: Eric Guthmann, Microsoft Corporation, ericguth@microsoft.com
                Author:     Chuck Reeves, Microsoft Corporation, creeves@microsoft.com
        </documentation>

        <import namespace="urn:uddi-org:api_v2" location="http://www.uddi.org/schema/uddi_v2.xsd" />

 <message name="add_publisherAssertions">
  <part name="body" element="uddi:add_publisherAssertions" />
 </message>

        <message name="assertionStatusReport">
  <part name="body" element="uddi:assertionStatusReport" />
 </message>

        <message name="authToken">
  <part name="body" element="uddi:authToken" />
 </message>

        <message name="bindingDetail">
                <part name="body" element="uddi:bindingDetail" />
        </message>

        <message name="businessDetail">
                <part name="body" element="uddi:businessDetail" />
        </message>

        <message name="delete_binding">
          <part name="body" element="uddi:delete_binding" />
        </message>

        <message name="delete_business">
          <part name="body" element="uddi:delete_business" />
        </message>

        <message name="delete_publisherAssertions">
  <part name="body" element="uddi:delete_publisherAssertions" />
 </message>

        <message name="delete_service">
          <part name="body" element="uddi:delete_service" />
        </message>

        <message name="delete_tModel">
          <part name="body" element="uddi:delete_tModel" />
        </message>

 <message name="discard_authToken">
  <part name="body" element="uddi:discard_authToken" />
 </message>

        <message name="dispositionReport">
                <part name="body" element="uddi:dispositionReport" />
        </message>

 <message name="get_assertionStatusReport">
  <part name="body" element="uddi:get_assertionStatusReport" />
```

```
</message>

<message name="get_authToken">
  <part name="body" element="uddi:get_authToken" />
</message>

<message name="get_publisherAssertions">
  <part name="body" element="uddi:get_publisherAssertions" />
</message>

<message name="get_registeredInfo">
  <part name="body" element="uddi:get_registeredInfo" />
</message>

        <message name="publisherAssertions">
                <part name="body" element="uddi:publisherAssertions" />
        </message>

        <message name="registeredInfo">
                <part name="body" element="uddi:registeredInfo" />
        </message>

        <message name="save_binding">
          <part name="body" element="uddi:save_binding" />
        </message>

        <message name="save_business">
          <part name="body" element="uddi:save_business" />
        </message>

        <message name="save_service">
          <part name="body" element="uddi:save_service" />
        </message>

        <message name="save_tModel">
          <part name="body" element="uddi:save_tModel" />
        </message>

        <message name="serviceDetail">
                <part name="body" element="uddi:serviceDetail" />
        </message>

        <message name="set_publisherAssertions">
                <part name="body" element="uddi:set_publisherAssertions" />
        </message>

        <message name="tModelDetail">
                <part name="body" element="uddi:tModelDetail" />
        </message>

        <portType name="Publish">
          <documentation>
                        This portType defines all of the UDDI publish operations.
          </documentation>

                <operation name="add_publisherAssertions">
                        <input message="tns:add_publisherAssertions" />
                        <output message="tns:dispositionReport" />
                        <fault name="error" message="tns:dispositionReport" />
                </operation>

                <operation name="delete_binding">
                        <input message="tns:delete_binding" />
                        <output message="tns:dispositionReport" />
                        <fault name="error" message="tns:dispositionReport" />
                </operation>

                <operation name="delete_business">
                        <input message="tns:delete_business" />
                        <output message="tns:dispositionReport" />
                        <fault name="error" message="tns:dispositionReport" />
                </operation>
```

```
		<operation name="delete_publisherAssertions">
			<input message="tns:delete_publisherAssertions" />
			<output message="tns:dispositionReport" />
			<fault name="error" message="tns:dispositionReport" />
		</operation>

		<operation name="delete_service">
			<input message="tns:delete_service" />
			<output message="tns:dispositionReport" />
			<fault name="error" message="tns:dispositionReport" />
		</operation>

		<operation name="delete_tModel">
			<input message="tns:delete_tModel" />
			<output message="tns:dispositionReport" />
			<fault name="error" message="tns:dispositionReport" />
		</operation>

		<operation name="discard_authToken">
			<input message="tns:discard_authToken" />
			<output message="tns:dispositionReport" />
			<fault name="error" message="tns:dispositionReport" />
		</operation>

		<operation name="get_assertionStatusReport">
			<input message="tns:get_assertionStatusReport" />
			<output message="tns:assertionStatusReport" />
			<fault name="error" message="tns:dispositionReport" />
		</operation>

		<operation name="get_authToken">
			<input message="tns:get_authToken" />
			<output message="tns:authToken" />
			<fault name="error" message="tns:dispositionReport" />
		</operation>

		<operation name="get_publisherAssertions">
			<input message="tns:get_publisherAssertions" />
			<output message="tns:publisherAssertions" />
			<fault name="error" message="tns:dispositionReport" />
		</operation>

		<operation name="get_registeredInfo">
			<input message="tns:get_registeredInfo" />
			<output message="tns:registeredInfo" />
			<fault name="error" message="tns:dispositionReport" />
		</operation>

		<operation name="save_binding">
			<input message="tns:save_binding" />
			<output message="tns:bindingDetail" />
			<fault name="error" message="tns:dispositionReport" />
		</operation>

	<operation name="save_business">
			<input message="tns:save_business" />
			<output message="tns:businessDetail" />
			<fault name="error" message="tns:dispositionReport" />
		</operation>

		<operation name="save_service">
			<input message="tns:save_service" />
			<output message="tns:serviceDetail" />
			<fault name="error" message="tns:dispositionReport" />
		</operation>

		<operation name="save_tModel">
			<input message="tns:save_tModel" />
			<output message="tns:tModelDetail" />
			<fault name="error" message="tns:dispositionReport" />
		</operation>

		<operation name="set_publisherAssertions">
```

```
                        <input message="tns:set_publisherAssertions" />
                        <output message="tns:publisherAssertions" />
                        <fault name="error" message="tns:dispositionReport" />
                    </operation>
</portType>


    <binding name="PublishSoap" type="tns:Publish">
                    <documentation>
                     This is the SOAP binding for the UDDI publish operations.
                    </documentation>

                    <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http" />

                    <operation name="add_publisherAssertions">
                     <soap:operation soapAction="add_publisherAssertions" style="document" />
                     <input message="tns:add_publisherAssertions">
                       <soap:body use="literal" parts="body" namespace="urn:uddi-org:api_v2" />
                     </input>
                     <output message="tns:dispositionReport">
                       <soap:body use="literal" parts="body" namespace="urn:uddi-org:api_v2" />
                     </output>
                     <fault name="error" message="tns:dispositionReport">
                       <soap:fault name="error" use="literal" />
                     </fault>
                    </operation>

                    <operation name="delete_binding">
                     <soap:operation soapAction="delete_binding" style="document" />
                     <input message="tns:delete_binding">
                       <soap:body use="literal" parts="body" namespace="urn:uddi-org:api_v2" />
                     </input>
                     <output message="tns:dispositionReport">
                       <soap:body use="literal" parts="body" namespace="urn:uddi-org:api_v2" />
                     </output>
                     <fault name="error" message="tns:dispositionReport">
                       <soap:fault name="error" use="literal" />
                     </fault>
                    </operation>

                    <operation name="delete_business">
                     <soap:operation soapAction="delete_business" style="document" />
                     <input message="tns:delete_business">
                       <soap:body use="literal" parts="body" namespace="urn:uddi-org:api_v2" />
                     </input>
                     <output message="tns:dispositionReport">
                       <soap:body use="literal" parts="body" namespace="urn:uddi-org:api_v2" />
                     </output>
                     <fault name="error" message="tns:dispositionReport">
                       <soap:fault name="error" use="literal" />
                     </fault>
                    </operation>

                    <operation name="delete_publisherAssertions">
                     <soap:operation soapAction="delete_publisherAssertions" style="document" />
                     <input message="tns:delete_publisherAssertions">
                       <soap:body use="literal" parts="body" namespace="urn:uddi-org:api_v2" />
                     </input>
                     <output message="tns:dispositionReport">
                       <soap:body use="literal" parts="body" namespace="urn:uddi-org:api_v2" />
                     </output>
                     <fault name="error" message="tns:dispositionReport">
                       <soap:fault name="error" use="literal" />
                     </fault>
                    </operation>

                    <operation name="delete_service">
                     <soap:operation soapAction="delete_service" style="document" />
                     <input message="tns:delete_service">
                       <soap:body use="literal" parts="body" namespace="urn:uddi-org:api_v2" />
                     </input>
                     <output message="tns:dispositionReport">
```

```
    <soap:body use="literal" parts="body" namespace="urn:uddi-org:api_v2" />
  </output>
  <fault name="error" message="tns:dispositionReport">
    <soap:fault name="error" use="literal" />
  </fault>
</operation>

<operation name="delete_tModel">
  <soap:operation soapAction="delete_tModel" style="document" />
  <input message="tns:delete_tModel">
    <soap:body use="literal" parts="body" namespace="urn:uddi-org:api_v2" />
  </input>
  <output message="tns:dispositionReport">
    <soap:body use="literal" parts="body" namespace="urn:uddi-org:api_v2" />
  </output>
  <fault name="error" message="tns:dispositionReport">
    <soap:fault name="error" use="literal" />
  </fault>
</operation>

<operation name="discard_authToken">
  <soap:operation soapAction="discard_authToken" style="document" />
  <input message="tns:discard_authToken">
    <soap:body use="literal" parts="body" namespace="urn:uddi-org:api_v2" />
  </input>
  <output message="tns:dispositionReport">
    <soap:body use="literal" parts="body" namespace="urn:uddi-org:api_v2" />
  </output>
  <fault name="error" message="tns:dispositionReport">
    <soap:fault name="error" use="literal" />
  </fault>
</operation>

<operation name="get_assertionStatusReport">
  <soap:operation soapAction="get_assertionStatusReport" style="document" />
  <input message="tns:get_assertionStatusReport">
    <soap:body use="literal" parts="body" namespace="urn:uddi-org:api_v2" />
  </input>
  <output message="tns:assertionStatusReport">
    <soap:body use="literal" parts="body" namespace="urn:uddi-org:api_v2" />
  </output>
  <fault name="error" message="tns:dispositionReport">
    <soap:fault name="error" use="literal" />
  </fault>
</operation>

<operation name="get_authToken">
  <soap:operation soapAction="get_authToken" style="document" />
  <input message="tns:get_authToken">
    <soap:body use="literal" parts="body" namespace="urn:uddi-org:api_v2" />
  </input>
  <output message="tns:authToken">
    <soap:body use="literal" parts="body" namespace="urn:uddi-org:api_v2" />
  </output>
  <fault name="error" message="tns:dispositionReport">
    <soap:fault name="error" use="literal" />
  </fault>
</operation>

<operation name="get_publisherAssertions">
  <soap:operation soapAction="get_publisherAssertions" style="document" />
  <input message="tns:get_publisherAssertions">
    <soap:body use="literal" parts="body" namespace="urn:uddi-org:api_v2" />
  </input>
  <output message="tns:publisherAssertions">
    <soap:body use="literal" parts="body" namespace="urn:uddi-org:api_v2" />
  </output>
  <fault name="error" message="tns:dispositionReport">
    <soap:fault name="error" use="literal" />
  </fault>
</operation>

<operation name="get_registeredInfo">
```

```
  <soap:operation soapAction="get_registeredInfo" style="document" />
  <input message="tns:get_registeredInfo">
    <soap:body use="literal" parts="body" namespace="urn:uddi-org:api_v2" />
  </input>
  <output message="tns:registeredInfo">
    <soap:body use="literal" parts="body" namespace="urn:uddi-org:api_v2" />
  </output>
  <fault name="error" message="tns:dispositionReport">
    <soap:fault name="error" use="literal" />
  </fault>
</operation>

<operation name="save_binding">
  <soap:operation soapAction="save_binding" style="document" />
  <input message="tns:save_binding">
    <soap:body use="literal" parts="body" namespace="urn:uddi-org:api_v2" />
  </input>
  <output message="tns:bindingDetail">
    <soap:body use="literal" parts="body" namespace="urn:uddi-org:api_v2" />
  </output>
  <fault name="error" message="tns:dispositionReport">
    <soap:fault name="error" use="literal" />
  </fault>
</operation>

<operation name="save_business">
  <soap:operation soapAction="save_business" style="document" />
  <input message="tns:save_business">
    <soap:body use="literal" parts="body" namespace="urn:uddi-org:api_v2" />
  </input>
  <output message="tns:businessDetail">
    <soap:body use="literal" parts="body" namespace="urn:uddi-org:api_v2" />
  </output>
  <fault name="error" message="tns:dispositionReport">
    <soap:fault name="error" use="literal" />
  </fault>
</operation>

<operation name="save_service">
  <soap:operation soapAction="save_service" style="document" />
  <input message="tns:save_service">
    <soap:body use="literal" parts="body" namespace="urn:uddi-org:api_v2" />
  </input>
  <output message="tns:serviceDetail">
    <soap:body use="literal" parts="body" namespace="urn:uddi-org:api_v2" />
  </output>
  <fault name="error" message="tns:dispositionReport">
    <soap:fault name="error" use="literal" />
  </fault>
</operation>

<operation name="save_tModel">
  <soap:operation soapAction="save_tModel" style="document" />
  <input message="tns:save_tModel">
    <soap:body use="literal" parts="body" namespace="urn:uddi-org:api_v2" />
  </input>
  <output message="tns:tModelDetail">
    <soap:body use="literal" parts="body" namespace="urn:uddi-org:api_v2" />
  </output>
  <fault name="error" message="tns:dispositionReport">
    <soap:fault name="error" use="literal" />
  </fault>
</operation>

<operation name="set_publisherAssertions">
  <soap:operation soapAction="set_publisherAssertions" style="document" />
  <input message="tns:set_publisherAssertions">
    <soap:body use="literal" parts="body" namespace="urn:uddi-org:api_v2" />
  </input>
  <output message="tns:publisherAssertions">
    <soap:body use="literal" parts="body" namespace="urn:uddi-org:api_v2" />
  </output>
  <fault name="error" message="tns:dispositionReport">
```

```
                    <soap:fault name="error" use="literal" />
                  </fault>
                </operation>
        </binding>
</definitions>
```