

XML-Anwendungen (ANW)

Lernziele

- Sie haben einen soliden Überblick über die Verwendungsmöglichkeiten und Vorteile von XML auf verschiedenen Gebieten.
- Sie wissen, wo XML bereits erfolgreich eingesetzt wird und welche Trends in der Zukunft zu erwarten sind.
- Sie können einschätzen, welche Rolle XML im E-Business spielt und kennen die wichtigsten XML-Anwendungen und Initiativen im E-Business-Bereich.
- Sie kennen die relevanten Standards und das Konzept, das hinter den XML Web Services steht.
- Sie kennen die zukunftsweisenden XML-Standards aus dem Grafik- und Präsentationsbereich sowie für die Darstellung wissenschaftlicher Sachverhalte.
- Sie wissen, was Content Management ist und welche Rolle XML dabei spielt.

XML und E-Business

E-Commerce und E-Business

E-Commerce

E-Commerce = Electronic Commerce (*Elektronischer Handel*)

E-Commerce ist der Austausch von Gütern oder Dienstleistungen gegen Geld zwischen zwei oder mehr Wirtschaftseinheiten über das Internet.

E-Business

- von IBM 1998 geprägter Begriff
- erstreckt sich über alle Prozesse *innerhalb und außerhalb* des Unternehmens

IBM-Definition

The transformation of key business processes through the use of Internet Technologies.

Kategorisierung des E-Business

Business to Consumer (B2C)

- Bestell- und Verkaufsprozess eines Anbieters gegenüber einer großen, wechselnden Zahl an Kunden.
- Niedriges Transaktionsvolumen
- lockere Bindung zwischen Transaktionspartnern
- Kunde meist eine einzelne Person
- typische Anwendung: webbasierte Katalog- und Buchungsanwendungen
- **Beispiele:** *Amazon, Dell*

Business to Business (B2B)

- findet zwischen Unternehmen statt
- Handel entlang komplexer Wertschöpfungsketten
- typische Anwendungen: Beschaffungssysteme (*Procurement-Systeme*), Marktplätze und Broker-Systeme zur Zusammenführung von Anbietern und Nachfragern, jede Form von Extranet-Integration zwischen Unternehmen
- **Beispiele:** *T-Mart* (Marktplatz der Deutschen Telekom;
<http://www.t-mart.com>)

Zahlen

- B2C entwickelt sich nach der Krise der New Economy moderater
- B2B-Handel bis 2005: 6 Trillionen Dollar (Jupiter Media Metrix:
<http://www.jup.com/company/pressrelease.jsp?doc=pr001002>)

Mensch und Maschine im E-Business

Bei Geschäftsbeziehungen im E-Business sind immer Maschinen in Form von Computern beteiligt. Ziel ist allerdings, soviel als möglich zu automatisieren. Somit ergibt sich eine weitere wichtige Klassifizierungsmöglichkeit für E-Business:

- Mensch-Maschine
- Maschine-Maschine

Anforderungen an E-Business-Systeme

- **Interoperabilität** zwischen den beteiligten Systemen
- **Kohärenz** der Systeme - standardisiertes Vokabular ist notwendig, so dass die Software des Lieferanten den Bestellauftrag des Kunden "verstehet"
- **Flexibilität** - schnelle Anpassbarkeit an neue Situationen und Geschäftsbedingungen
- **Workflow-Management** - interne Prozesse müssen Daten aus externen Geschäftsprozessen weiterverarbeiten können

Der Urvater des E-Business: EDI

E-Business gibt es im Grunde bereits seit einem Vierteljahrhundert:

- Beginn Mitte der 70er-Jahre: UPC Barcodes
- Mitte der 80er-Jahre: EDI

Barcodes sind mittlerweile allgegenwärtig, aber die Durchschlagskraft von EDI war sehr viel geringer.

Trotz allem ist EDI im Grunde der Wegbereiter für Durchbruch des E-Business.

EDI = Electronic Data Interchange (*elektronischer Datenaustausch*)

Unter klassischem EDI wird seit den 70er Jahren der papierlose Austausch von strukturierten Geschäftsdaten, wie Bestellungen, Rechnungen etc., von Anwendung zu Anwendung über ein elektronisches Übertragungsmedium verstanden. Die Möglichkeit zur automatischen Weiterverarbeitung der ausgetauschten Nachrichten ohne menschliches Eingreifen ist dabei ein wesentliches Kennzeichen.

EDI-Standards und verantwortliche Organisationen

- Accredited Standards Committee X12 (ASC X12)

=> X12-Standard (USA und Kanada)

Der X12-Standard wird z.Zt. verwaltet von der DISA (Data Interchange Standards Association).

- United Nations Electronic Data Interchange for Administration, Commerce and Transport (UN/EDIFACT)

=> EDIFACT-Standard (außerhalb Nordamerikas) [ISO 9735]

Die beiden Standards sind nicht kompatibel zueinander!

Vorteile von klassischem EDI gegenüber papierbasierten Lösungen

- Kostenreduktion
- Zeitreduktion
(*Beispiele*: Verkürzung des Lieferzyklus, schnellere Zahlungsabwicklung, Beschleunigung der Zollabfertigung von Waren)
- Verringerung der Fehlerwahrscheinlichkeit
- engere Kundenbindung und erweiterter Kundendienst
(*Beispiele*: Paketverfolgung, 1-Click-Bestellung)
- bessere Planung durch Realisierung von "Just-in-Time"-Bestandssystemen
(*Beispiele*: Reduktion der Lagerbestände)
- Zugriff auf Echtzeit-Informationen zur Entscheidungsunterstützung
- Förderung der Konkurrenzfähigkeit
- schnellere Reaktionsmöglichkeiten auf den Markt

Das Konzept von EDI

- basiert auf dem Begriff der **Transaktion**
- Transaktionen bestehen aus **Nachrichten**. Nachrichten sind der elektronische Ersatz für die Papierdokumente eines Geschäftsprozesses (Beispiel: Bestellauftrag, Rechnung, Lieferschein).
- Nachrichten werden in standardisierten **Formaten** verfasst und mit vordefinierten Kommunikationsprotokollen übertragen
- Formate müssen zwischen den Geschäftspartnern vereinbart und ausgetauscht werden. Dies geschieht über **Schemata** (Beschreibung der Formate) und **Repositories** (Sammlung von Schemata)
- zu Beginn proprietäre Vernetzung über Standleitungen (VAN = Value Added Networks), heute zunehmend **Extranets**

Die Konzentration liegt eindeutig auf der Definition (branchenübergreifender) Nachrichtenformate.

Die Regeln für den Austausch der Nachrichten (Geschäftsprozesse) müssen zwischen den beiden Geschäftspartnern in einem sogenannten **Trading Partner Agreement (TPA)** vereinbart werden.

Das EDI-Konzept umfasst jedoch nicht nur die Spezifikation von Nachrichtenformaten, sondern auch Kommunikationsprotokolle und sogar einige Hardware-Anforderungen.

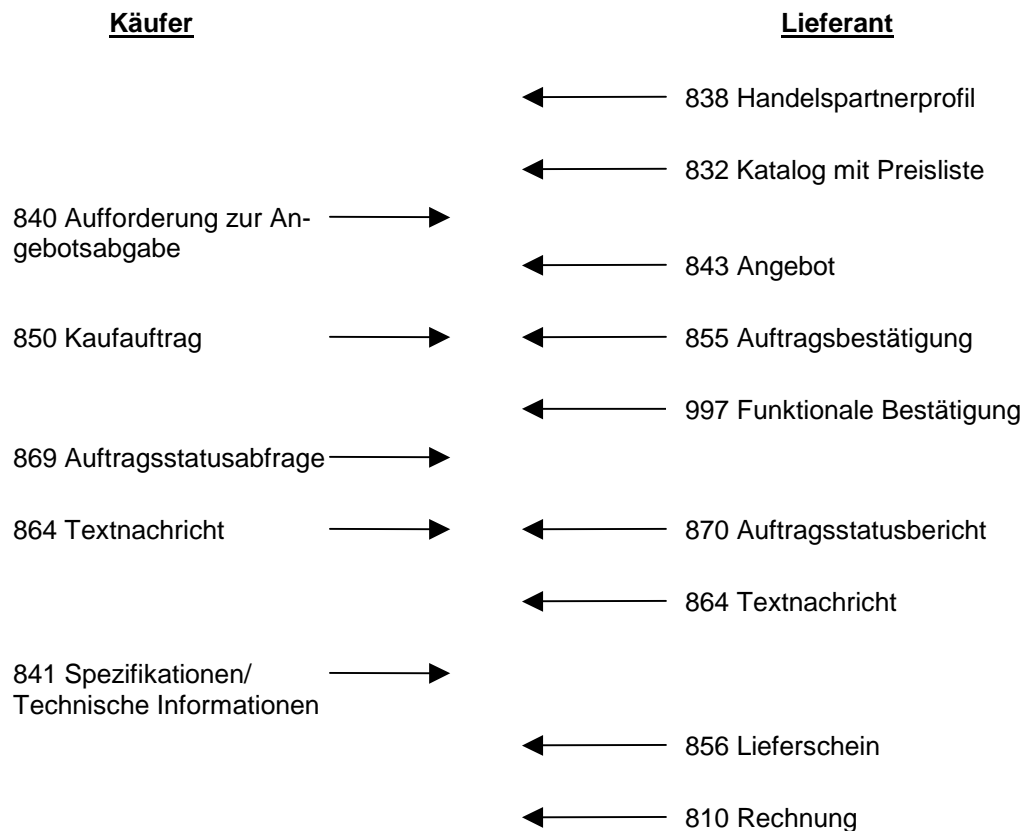


Abbildung ANW-1: EDI-Transaktion

Einige Kernprinzipien des EDI-Konzepts

- Nachrichten erhalten eine eindeutige Identifikation
- Definition von Basisstrukturen für branchenübergreifenden Standardnachrichten
- Definition von *wiederverwertbaren* Komponenten (Segmenten)
- TPAs
- Übertragung nur der jeweils variablen Daten in einer Nachricht
- Funktionale Bestätigungen

Eine EDIFACT-Nachricht

Beispiel ANW-1: EDI-Nachricht im Original

```
UNB+UNOC:3+4331111111008:14+4121212120005:14+990519:120+
525+++++EANCOM'UNH+785+ORDERS:D:93A:UN:EAN007'BGM+220+01
4501234567'DTM+137:19990519:102'NAD+BY+4909090909991::9+
+Kunde AG Zentraleinkauf Abt. 062+Richard-Wagner-Str.
2+Hamburg+40000++DE'NAD+SU+4999999123123::9++Lieferant
GmbH+Siemensring 90-
92+Muenchen+47877++DE'LIN+1++4999993123456:EN'IMD+F++:::
Heckenschere
500'QTY+21:50'PRI+AAA:28.9::LIU'PRI+AAB:54.9::SRP'
```

Bestellt wird in diesem Beispiel eine Heckenschere (Artikelnr. 4999993123456) zum 24.05.1999. Als Referenznr. (Auftragsnr.) wird 014501234567 übermittelt. Bei der Bestellung ist die *Kunden AG Zentraleinkauf* der Auftraggeber und *Kunden AG Warenverteilzentrum* der Waren- und Rechnungsempfänger. Die Heckenschere soll vom Lieferanten *Lieferant GmbH* besorgt werden. Zur besseren Lesbarkeit ist die EDI-Nachricht im folgenden Beispiel kommentiert.

Beispiel ANW-2: EDI-Nachricht kommentiert

Interchange Header:

UNB+UNOC:3+4331111111008:14+4121212120005:14+990519:120+525+++++EANCOM'

Message Header

UNH+785+ORDERS:D:93A:UN:EAN007'

Start der Nachricht (Begin of message)

BGM+220+014501234567'

Datum (Date/Time/Period)

DTM+137:19990519:102'

Adressen (Name and Adress) Auftraggeber

NAD+BY+4909090909991::9++Kunde AG ZEntraleinkauf
Abt. 062+Richard-Wagner-Str. 2+Hamburg+40000++DE'

Adressen (Name and Adress) Lieferant

NAD+SU+4999999123123::9++Lieferant GmbH+Siemensring
90-92+Muenchen+47877++DE'

Bestellposition (Line Item)

LIN+1++4999993123456:EN' **EAN-Artikelnummer**

Artikelbeschreibung

IMD+F+++::Hechenschere 500'

Bestellmenge

QTY+21:50'

Preisberechnung Netto

PRI+AAA:28.9::LIU'

Preisberechnung Brutto

PRI+AAB:54.9::SRP'

Ein Bestellauftrag in XML könnte dagegen etwa so aussehen:

Beispiel ANW-3: Bestellauftrag in XML

```
<?xml version="1.0" encoding="iso-8859-1"?>
<bestellauftrag>
  <header>
    <auftrags-nr>1234</auftrags-nr>
    <datum>12.03.2001</datum>
    <absender>www.meineFirma.de</absender>
    <empfänger>www.einkauen.de</empfänger>
  </header>
  <body>
    <lieferadresse>
      <name>Dieter Mann</name>
      <strasse>Kohlenweg 3</strasse>
      <plz>64321</plz>
      <ort>Eschborn</ort>
      <email>mann@freemail.de</email>
    </lieferadresse>
    <währung>DM</währung>
  </body>
</bestellauftrag>
```

```
<item>
  <itemID>234156</itemID>
  <itemBez>Laptop Computer D456</itemBez>
  <itemPreis>3459,89</itemPreis>
</item>
<accountID>7865</accountID>
</bestellauftrag>
```

Nachteile von traditionellem EDI

Die Information über die Bedeutung des Inhalts von EDI-Daten ist nicht in Tags festgelegt, sondern sie ergibt sich aus der Position bzw. der Reihenfolge der jeweiligen Element einer Nachricht:

Vorteil:

- kompaktes Format => niedrigeres Übertragungsvolumen

Nachteil:

- nicht menschenlesbar
- unflexibel
- Validierung problematisch
- schwierig zu implementieren
- teuer (in Betrieb und Implementierung)
- langsame Standardentwicklung
- keine Einbindung von Binärdaten (z.B. Röntgenbilder, CAD-Zeichnungen) oder Softwarekomponenten möglich
- schwierige automatische Weiterverarbeitung der Nachrichten
- umständliche Konvertierungsprozeduren (*Mapping*) in das interne Format aufgrund des komplexen Regelwerks

Folge:

- geringe Verbreitung (EDI beschränkt sich größtenteils auf Großunternehmen und Banken sowie kleinere Zuliefererfirmen, die EDI auf Kundenverlangen einsetzen müssen)
- Insellösungen, die sich um einen dominanten Spieler ranken
- "Balkanisierung" der Nachrichtenformate

Zahlen

95% von den 1000 größten Unternehmen nutzen EDI, aber nur 5% der kleineren Unternehmen

=> Rachel Foerster, "The Next Generation EC/EDI Standards", *The Executive Summary*, January 1999 (www.rfa-edi.com/00-edi.htm)

Der Einsatz von XML im E-Business

XML ist das ideale Datenformat für den Austausch von Geschäftsdaten über das Internet. Aufgrund der Auszeichnung ist Weiterverarbeitung möglich. Die DTD bzw. ein XML-Schema kann inhaltliche Strukturen festlegen und validierbar machen.

Vorteile von XML

- einfache und kostengünstige Implementierung (offene Standards plus verfügbare Internet-Technologien)
- geringere Einstiegskosten als klassisches EDI
- einfache Syntax für Nachrichtenformate
- XML ist sowohl für manuelle als auch für maschinelle Verarbeitung geeignet.
- Flexibilität
- Einbindung von Skripten, Komponenten und Binärdaten
- preiswerte Werkzeuge zur Erstellung und zum Austausch von XML-Nachrichten sind verfügbar
- sehr große Marktbreite
- hervorragende Herstellerunterstützung (Microsoft, Sun, IBM...)
- Zahl der Programmierer, die XML unterstützen, nimmt zu
- der Hype um XML weckt das Interesse von immer mehr Unternehmen
- bereits heute eine Reihe von Brancheninitiativen mit dem Ziel, Repositories/Registries für DTDs und Schemata zu erstellen
- Beschleunigung der Standardisierung durch Trennung der Standardisierungsprozesse für Syntax, Semantik und Präsentation

- Trennung von Inhalt/Struktur und Geschäftsregeln ist möglich
- bessere Unterstützung des Workflow in die internen Systeme
- normative Kraft des Faktischen: XML hat sich durchgesetzt!

Folge:

E-Business wird auch für kleine und mittelständische Unternehmen attraktiv

aber:

E-Business-Systeme sind nicht alleine durch den Einsatz von XML leichter implementierbar. Es handelt sich immer um komplexe Systeme, die oftmals zu Beginn hohe Investitionen erfordern!

Nachteile von XML

- geringere Sicherheit und Qualität des Netzes
- größere Übertragungsvolumina (da neben den eigentlichen Nutzdaten auch deren semantische Beschreibung übertragen werden muss) und daher geringere Performance (aber: geschäftliche Nachrichten sind in der Regel nicht zeitkritisch)

Notwendigkeit zur Standardisierung

XML alleine ist keine Lösung für jegliche Modellierungs- und Implementierungsprobleme. Es handelt sich zunächst nur um eine technische Lösung. Voraussetzung für einen elektronischen Austausch, gleich ob per traditionellem EDI oder mittels XML, ist ein Nachrichtenformat, welches von allen beteiligten Geschäftspartnern verstanden wird.

Denn jedes Unternehmen kann DTDs nach seinen Erfordernissen mit dem jeweiligen Geschäftspartner vereinbaren. Jedoch führt die Vereinbarung von unternehmensspezifischen, bilateralen Formaten bei einer Vielzahl von Kommunikationspartnern zu Inkompatibilitäten und ist nicht praktikabel.

=> Notwendigkeit, sich auf eine einheitliche Definition der Tags und ihrer semantischen Interpretation zu einigen.

Entscheidend für den Erfolg von EDI/XML sind globale, herstellerunabhängige Standards für den geschäftlichen Austausch.

XML-Standardisierungsinitiativen

- "Übersetzung" von EDI-Standards in XML (EDI/XML)
- branchenspezifische (vertikale) und branchenübergreifende (horizontale bzw. funktionale) XML-Vokabulare
- XML-Repositories und XML-Registries
- XML-basierte E-Business-Frameworks

Branchenspezifische XML-Vokabulare (*Verticals*)

Branche	Standard
Finanzen	FinXML
Gesundheitswesen	HL7
Personalverwaltung	HR-XML
Chemie	CML

Beispiel ANW-1: Stellenbeschreibung in HR-XML

```
<JobInformation>
  <JobTitle>XML Entwickler</JobTitle>
  <Location>Karlsruhe, Deutschland</Location>
  <Description>
    <JobPurpose>
      <ul>
        <li>
          Weiterentwicklung eines datenbankbasierten In-
          ternet-Application-Frameworks
        </li>
        <li>
          Konzeption und Umsetzung neuer XML-Komponenten
        </li>
        <li>
          Gestaltung von XML-DTDs bzw. Schemas
        </li>
        <li>
          Umsetzung der Projekte mit externen Partnern
        </li>
      </ul>
    </JobPurpose>
  </Description>
</JobInformation>
```

```

        </ul>
    <JobPurpose>
    <QualifRequired>
        <SoftwareRequired>XML Spy</SoftwareRequired>
        <ExperienceRequired YearsOfExperience="2">
            Sie sollten Erfahrungen mit E-Business und XML
haben
        </ExperienceRequired>
    </QualifRequired>
    <Preferences>
        <p>Erfahrungen in der Bankenbranche sind von Vor-
teil</p>
    </Preferences>
    <Compensation>
        <p>Innovative Projekte</p>
    </Compensation>
</Description>
<HowToApply>
    <p>Schicken Sie uns Ihre aussagekräftigen Bewer-
bungsunterlagen und einige Referenzen</p>
</HowToApply>
</JobInformation>

```

Branchenübergreifende XML-Vokabulare (*Functions*)

Bezeichnung	Organisation	URL
xCBL	xCBL.org (Commerce One, SAP, Sun, HP ...)	http://www.xcbl.org
ICE	Graphic Communications Association (GCA) (Vignette, Adobe, Sun ...)	http://www.w3.org/TR/NOTE-ice
cXML	cXML:org (Ariba ...)	http://www.cxml.org
UBL	OASIS	http://www.oasis-open.org/committees/ubl/
BMEcat	BME, wiss. Institute	

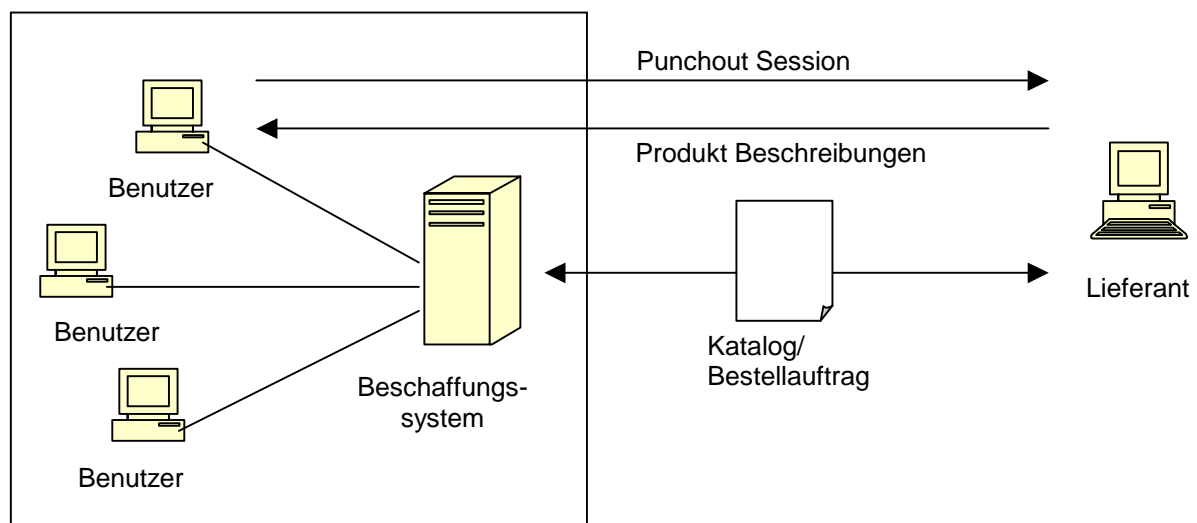
cXML

cXML = commerce XML

cXML ist ein Protokoll, das XML-basierte B2B-E-Commerce-Transaktionen über das Internet ermöglicht. Es wurde von der Firma Ariba mit dem Ziel der Integration weltweiter, verteilter E-Commerce-Anwendungen entwickelt.

cXML ist geeignet für Transaktionen im Rahmen von:

- elektronischen Produktkatalogen
- cXML-Punchout-Katalogen (Punchout-Kataloge = interaktive Kataloge auf einer Webseite)
- Beschaffungssystemen
- Kaufgemeinschaften



Struktur von cXML-Dokumenten

- Umschlag = <cXML>-Wurzelement (Sprachangabe, Zeitstempel u.a.)
- Header-Element (Adressinformationen, Authentifizierungsinformationen)
- Request/Response oder Message (bei unidirektionaler Kommunikation)

Beispiel ANW-2: Bestellung (Order-Request)

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cXML SYSTEM
"http://xml.cXML.org/schemas/cXML/1.1.009/cXML.dtd">

<cXML payloadID="3223232@ariba.acme.com"
      timestamp="1999-03-12T18:39:09-08:00"
      xml:lang="en-US">

  <Header>
    <From>
      <Credential domain="AribaNetworkUserId">
        <Identity>admin@acme.com</Identity>
      </Credential>
      <Credential domain="AribaNetworkUserId"
type="marketplace">
        <Identity>bigadmin@marketplace.org</Identity>
      </Credential>
      <Credential domain="BT">
        <Identity>2323</Identity>
      </Credential>
    </From>
    <To>
      <Credential domain="DUNS">
        <Identity>942888711</Identity>
      </Credential>
    </To>
    <Sender>
      <Credential domain="AribaNetworkUserId">
        <Identity>admin@acme.com</Identity>
        <SharedSecret>abracadabra</SharedSecret>
      </Credential>
      <UserAgent>Ariba.com Network V1.0</UserAgent>
    </Sender>
  </Header>

  <Request deploymentMode="test">

    <OrderRequest>

      <OrderRequestHeader orderID="DO1234" order-
Date="1999-03-12" type="new">
        <Total>
          <Money currency="USD">12.34</Money>
        </Total>
        <ShipTo>
          <Address>
            <Name xml:lang="en">Acme</Name>
            <PostalAddress name="foo">
```

```

        <DeliverTo>Joe Smith</DeliverTo>
        <DeliverTo>Mailstop M-543</DeliverTo>
        <Street>123 Anystreet</Street>
        <City>Sunnyvale</City>
        <State>CA</State>
        <PostalCode>90489</PostalCode>
        <Country isoCountryCode="US">United
States</Country>
    </PostalAddress>
</Address>
</ShipTo>
<BillTo>
    <Address>
        <Name xml:lang="en">Acme</Name>
        <PostalAddress name="foo">
            <Street>123 Anystreet</Street>
            <City>Sunnyvale</City>
            <State>CA</State>
            <PostalCode>90489</PostalCode>
            <Country isoCountryCode="US">United
States</Country>
        </PostalAddress>
    </Address>
</BillTo>
    <Shipping trackingDomain="FedEx" track-
ingId="1234567890">
        <Money currency="USD">12.34</Money>
        <Description xml:lang="en-us">FedEx 2-
day</Description>
    </Shipping>
    <Tax>
        <Money currency="USD">12.34</Money>
        <Description xml:lang="en">foo</Description>
    </Tax>
    <Payment>
        <PCard number="1234567890123456" expira-
tion="1999-03-12"/>
    </Payment>
    <Comments xml:lang="en-US">Anything well formed
in XML can go here.</Comments>
</OrderRequestHeader>

    <ItemOut quantity="2" requestedDeliveryDate="1999-
03-12">
        <ItemID>
            <SupplierPartID>1233244</SupplierPartID>
        </ItemID>
        <ItemDetail>
            <UnitPrice>
                <Money currency="USD">1.34</Money>

```

```

    </UnitPrice>
    <Description xml:lang="en">hello</Description>
    <UnitOfMeasure>EA</UnitOfMeasure>
    <Classification domain="SPSC">
      12345
    </Classification>
    <ManufacturerPartID>234</ManufacturerPartID>
    <ManufacturerName>foobar</ManufacturerName>
    <URL>www.foo.com</URL>
  </ItemDetail>
  <ShipTo>
    <Address>
      <Name xml:lang="en">Acme</Name>
      <PostalAddress name="foo">
        <Street>123 Anystreet</Street>
        <City>Sunnyvale</City>
        <State>CA</State>
        <PostalCode>90489</PostalCode>
        <Country isoCountryCode="US">United
States</Country>
      </PostalAddress>
    </Address>
  </ShipTo>
  <Shipping>
    <Money currency="USD">1.34</Money>
    <Description xml:lang="en-us">FedEx 2-
day</Description>
  </Shipping>
  <Tax>
    <Money currency="USD">1.34</Money>
    <Description xml:lang="en">foo</Description>
  </Tax>
  <Distribution>
    <Accounting name="DistributionCharge">
      <Segment type="G/L Account" id="23456" de-
scription="Entertainment"/>
      <Segment type="Cost Center" id="2323" de-
scription="Western Region Sales"/>
    </Accounting>
    <Charge>
      <Money currency="USD">.34</Money>
    </Charge>
  </Distribution>
  <Distribution>
    <Accounting name="DistributionCharge">
      <Segment type="G/L Account" id="456" de-
scription="Travel"/>
      <Segment type="Cost Center" id="23" descrip-
tion="Europe Implementation"/>
    </Accounting>

```

```

        <Charge>
            <Money currency="USD">1</Money>
        </Charge>
    </Distribution>
    <Comments xml:lang="en-US">Anything well formed
in XML can go here.</Comments>
</ItemOut>

</OrderRequest>

</Request>

</cXML>

```

Beispiel ANW-3: Eingangsbestätigung (Order-Response)

```

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE cXML SYSTEM
"http://xml.cXML.org/schemas/cXML/1.1.009/cXML.dtd">

<cXML payloadID="9949494@cxml.workchairs.com"
xml:lang="en-US"
timestamp="1999-03-12T18:39:09-08:00">

    <Response>
        <Status code="200" text="OK"/>
    </Response>

</cXML>

```

XML-Repositories

Es gibt mittlerweile Tausende von branchenspezifischen und branchenübergreifenden XML-Vokabularen. Falls Sie Ihre eigenen Geschäftsdaten mit XML modellieren müssen, sollten Sie in keinem Falle das Rad neu erfinden, sondern in einer der zahlreichen Repositories für XML-Vokabulare nach passenden DTDs oder Schemata suchen.

Bezeichnung	Organisation	URL
XML.org	OASIS	http://www.xml.org

BizTalk.org von Microsoft ist eingestellt worden!

XML-basierte E-Business-Frameworks

Bezeichnung	Organisation	URL
ebXML	UN/CEFACT u. OASIS	http://www.ebxml.org
RosettaNet	RosettaNet (Sun, ASC X12-Gruppe, IBM, SAP...)	http://www.rosettanet.org
eCo Framework	CommerceNet	http://eco.commerce.net
XML/EDI	XML EDI Group	http://www.xmledi-group.org/

ebXML

ebXML ist eine Initiative des *United Nations Body for Trade Facilitation and Electronic Business (UN/CEFACT)* und der *Organisation for the Advancement of Structured Information Standards (OASIS)*.

Die Organisationen

UN/CEFACT (<http://www.uncefact.org>)

globale Organisation, die verantwortlich ist für weltweite Handelsstrategien und die technische Entwicklung auf dem Gebiet der Handelserleichterung, insbesondere im E-Business. Die UN/CEFACT bringt jahrzehntelange Erfahrung in der Analyse von organisationsübergreifenden Geschäftstransaktionen und in der Schaffung kompatibler Systemschnittstellen.

OASIS (<http://www.oasis-open.org>)

Non-Profit-Organisation; bedeutendste Dachorganisation im XML-Sektor.; entwickelt Industriestandards für Interoperabilität im XML/SGML-Bereich; OASIS unterhält zudem das XML-Portal XML.org, eine Registry für XML-Schemas.

Ziel von ebXML

Entwicklung eines technischen, modularen Frameworks für das E-Business, mit dessen Hilfe die Eintrittshürden für kleine und mittelständische Unternehmen (KMUs) in das Electronic Business überwunden werden sollen:

It's goal is to develop a set of specifications to allow any business of any size in any industry to do business with any other entity in any other industry anywhere in the world.

- Start: Anfang Dezember 1999
- Abschluss: Mai 2001
- mehrere Arbeitsgruppen
- technische Architektur basiert auf öffentlichen Internet-Standards
- modularer Aufbau ermöglicht inkrementelle Implementierung

Anforderungen

ebXML soll Unternehmen in die Lage versetzen:

- zu entdecken, welche Produkte und Serviceleistungen andere Unternehmen anbieten
- festzulegen, welche gemeinsamen Geschäftsprozesse oder Informationsmodelle zu benutzen sind, um Produkte und Serviceleistungen auszutauschen zu können
- festzulegen, welche Form der Kommunikation für den Informationsaustausch zu wählen ist
- einen gegenseitigen Vertrag zu schließen
- Informationen und Dienstleistungen in automatisierter Form in Übereinstimmung mit dem geschlossenen Vereinbarungen auszutauschen

Architektur

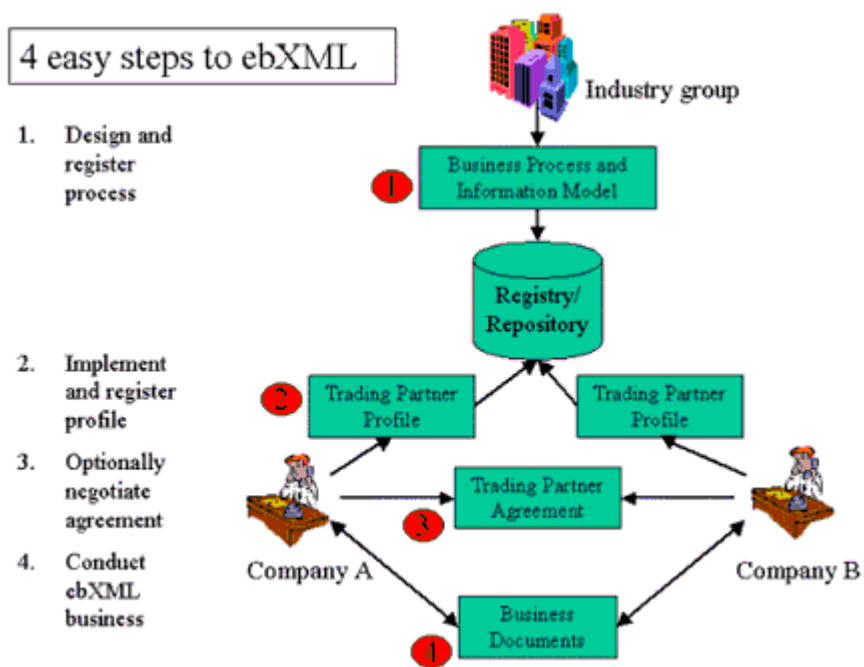


Abbildung ANW-3: ebXML-Architektur

ebXML Framework-Komponenten

- Business Process Specification Schema (BPSS)
- Core Components
- Registry/Repository
- Collaboration Protocol Profiles and Agreement
- Transport, Routing, and Packaging
- Security
- Architecture

XML Web Services

Was sind Web Services?

Web Services sind modulare, selbstbeschreibende Applikationen, die von überall im Internet (oder lokalem Netzwerk) veröffentlicht, lokalisiert und aufgerufen werden können. Anbieter und Nutzer eines XML Web Services müssen sich keine Gedanken über das Betriebssystem, die Sprachumgebung oder das Komponentenmodell machen, um einen Web Service anzubieten oder zu nutzen. XML Web Services basieren auf globalen und offenen Internet-Standards wie XML, HTTP und SMTP.

Web Services erweitern somit die traditionelle Client-Server-Architektur des Internet. Ein Client kann mit einem Server Kontakt aufnehmen und von ihm eine Dienstleistung anfordern, die dieser selbst bei anderen Diensten einfordern kann.

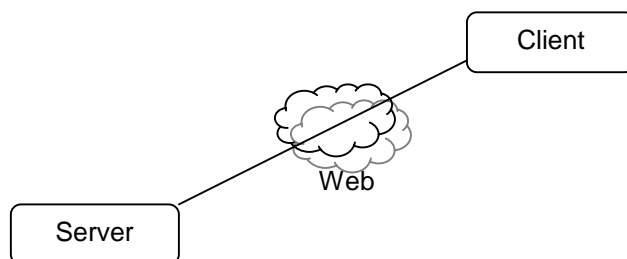


Abbildung ANW 4: Traditionelle verteilte Client/Server-Architektur

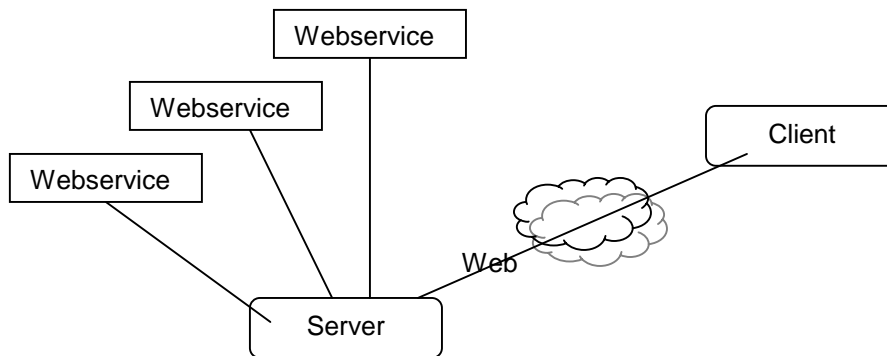


Abbildung ANW-5: Verteilte Client/Server-Architektur, die Web Services nutzt

Beispiele für Web Services:

- Buchungssystem im Reisebüro:
 Es muss Hotelzimmer und Flüge buchen, Bestätigungen über deren Verfügbarkeit einholen, Mietwagen besorgen und schließlich vermerken, wie und wann der Kunde seine Reise bezahlen will. Für jede dieser Aufgaben muss es Informationen von anderen Rechnern besorgen und dort Aktionen anstoßen.
 Implementiert man dieses Beispiel unter Zuhilfenahme von Web Services, so spricht das System per URL einen Service für Flugbuchungen an, der Flugpläne verschiedener Airlines abfragt und als einzelnes Dokument zur Verfügung stellt. Ein weiterer URL liefert einen Dienst, mit dem ein bestimmter Flug gebucht werden kann, und zuletzt wird dann ebenfalls mit einem Web Service die Hotel- oder Mietwagenbuchung durchgeführt.
- Börsennachrichten, Wetterdienste, Entfernungsberechnungen usw.
- Frachtkostenberechnung
 UPS könnte z.B. die Frachtkostenberechnung von der Post benutzen

Basisfunktionen eines Web Service

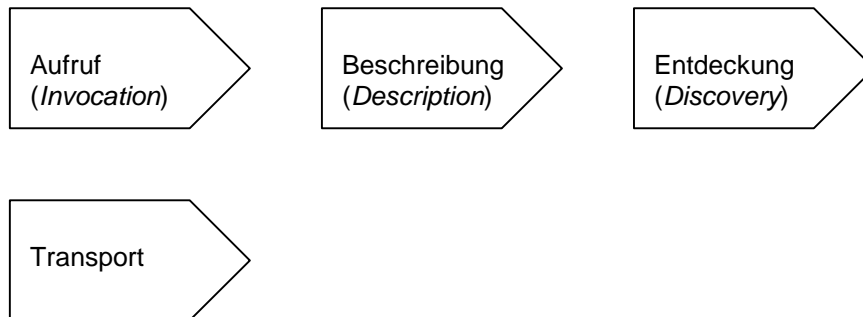


Abbildung ANW-6: Basisfunktionen eines Web Service

SOAP: XML-Dokumente als Nachrichten

SOAP = Simple Object Access Protocol

SOAP ist ein einfaches Protokoll zum Informationsaustausch in dezentralisierten, verteilten Umgebungen.

SOAP nutzt

- XML als eine Art Netzwerkprotokoll, um beliebige Daten und Methodenaufrufe an entfernte Systeme zu übertragen
- HTTP als Übertragungsprotokoll, kann aber auf jedem beliebigen Übertragungsprotokoll aufsetzen (Erweiterbarkeit)

Aufbau einer SOAP-Nachricht

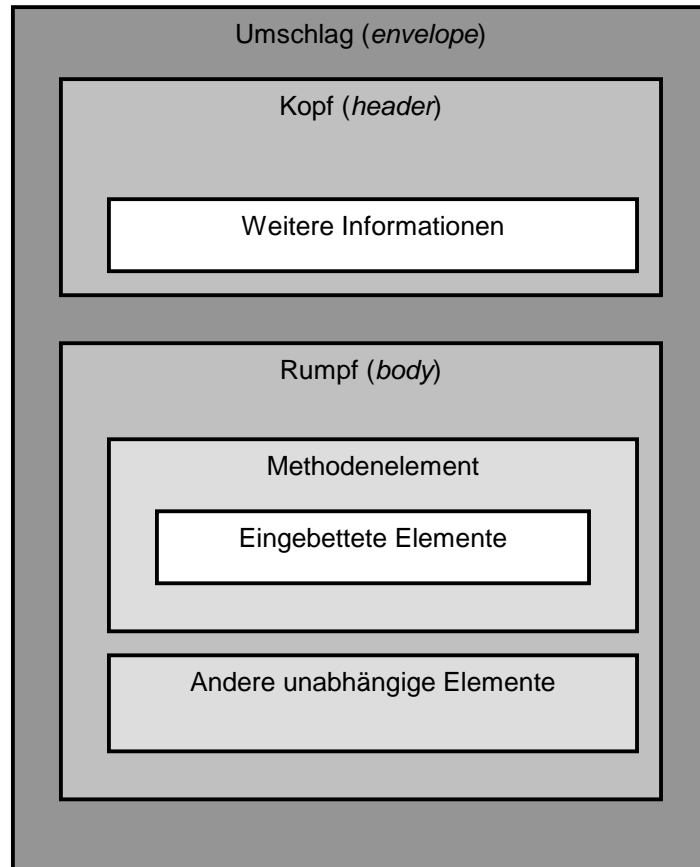


Abbildung ANW-7: Die Grundstruktur eines SOAP-Dokuments

Ein einfaches SOAP-Dokument sieht folgendermaßen aus:

Beispiel ANW-4: Einfache SOAP-Nachricht

```
<?xml version="1.0" encoding="iso-8859-1"?>
<soap:Envelope
  xmlns:soap=
    "http://schemas.xmlsoap.org/soap/envelope/"
  soap:encodingStyle=
    "http://schemas.xmlsoap.org/soap/encoding/">
<soap:Header>
  <h:from xmlns:h="http://www.meinServer/Header">
    Franz-Josef.Herpers@spmtechnologies.com
  </h:from>
</soap:Header>
```

```
<soap:Body>
  <w:Gruß xmlns:w="http://www.meinServer/HalloWelt/">
    <w:Nachricht>Hallo Welt!</w:Nachricht>
  </w:Gruß>
</soap:Body>
</soap:Envelope>
```

Interessanter für den Bereich Web Services sind allerdings das Request-/Response-Paradigma oder auch ganze Nachrichtenketten. Beides wird von SOAP unterstützt. Das Grundprinzip ist dabei recht einfach: Jede Nachricht hat einen Sender und einen Empfänger und Empfänger können miteinander verkettet werden.

Beispiel ANW-4: SOAP-Request und -Response

Request

```
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:Body>
    <m:GetLastTradePrice xmlns:m="Some-URI">
      <symbol>DIS</symbol>
    </m:GetLastTradePrice>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Response

```
HTTP/1.1 200 OK
Content-Type: text/xml;
charset="utf-8"
Content-Length:
nnnn
```

```
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV=
    "http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-ENV:encodingStyle=
    "http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
    <m:GetLastTradePrice xmlns:m="Some-URI">
      <Price>34.5</Price>
    </m:GetLastTradePrice>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Modularer Aufbau von SOAP

SOAP ist konzipiert auf Erweiterbarkeit in folgenden Gebieten:

- *Nachrichtensyntax*
Erweiterungen der Nachrichtensyntax können über den SOAP-Header hinzugefügt werden
- *Daten*
Eine SOAP-Nachricht kann beliebige Daten enthalten. SOAP stellt eine Methode der Serialisierung zur Verfügung, aber Anwendungen können ebenso ihre eigenen Regeln definieren. Außerdem kann ein XML-Schema referenziert werden, um die Daten zu validieren.
- *Transport*
SOAP definiert, wie Nachrichten über HTTP ausgetauscht werden, aber es kann jedes beliebige alternative Transportprotokoll verwendet werden.
- *Verwendung*
SOAP legt nicht fest, wozu eine Nachricht dient. SOAP definiert aber eine Konvention, mit der Sie SOAP-Nachrichten für entfernte Methodenaufrufe verwenden können (Remote Procedure Calls = RPC).

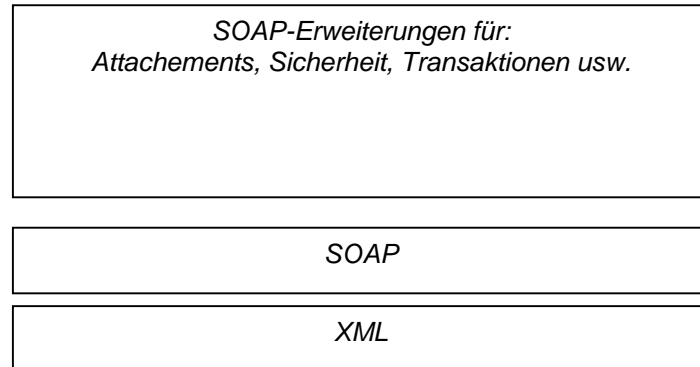
Unterstützung des Standards

- fast alle Branchenriesen unterstützen mittlerweile den Standard (z.B. Biz-Talk-Server von Microsoft); insgesamt bereits über 20 Implementierungen
- es gibt Toolkits in fast allen Programmiersprachen

TOOL-TIPP

Eine vollständige Liste aller SOAP-Implementierungen finden Sie unter:
<http://www.soapware.org/directory/4/implementations>.

Aufruf eines Web Services



WSDL

WSDL = Web Service Description Language

WSDL ist eine XML-Anwendung zur Beschreibung von Web Services. Sie wurde unter Federführung von Microsoft und IBM entwickelt.

Warum WSDL?

Web Services müssen selbstbeschreibend sein, wenn Sie von überall her aufgerufen werden können. Über WSDL können Web Services Ihre Schnittstellen veröffentlichen und ein Client kann den Web Service aufrufen, ohne Details über den individuellen Web Service, die Plattform, auf dem dieser läuft, oder das darunter liegende Betriebssystem wissen zu müssen.

Komponenten eines WSDL-Dokuments

- Typen (*types*)
- Nachrichten (*messages*)
- Operationen (*operations*)

- Port-Typen (*port types*)
- Bindings
- Ports
- Services

Ein WSDL-Dokument besteht aus Definitionen. Diese Definitionen definieren einen **Service** als einen Satz von einem oder mehreren Netzwerk-Endpunkten oder **Ports**. Jeder Port ist mit einem spezifischen **Binding** assoziiert. Ein Binding definiert, wie eine abstrakte Menge von **Operationen** und **Nachrichten** an einen Port und ein bestimmtes Protokoll gebunden sind. Ein Binding mappt ein spezifisches Protokoll auf einen **Port-Typ**. Ein Port-Typ besteht aus einer oder mehreren Operationen. Operationen repräsentieren eine Menge von Dingen, die ein Service "tun" kann. Jede Operation setzt sich aus einer Menge von abstrakten Nachrichten zusammen. Nachrichten repräsentieren die Daten, die während einer Operation kommuniziert werden. Jede Nachricht enthält ein oder mehrer (Daten-)Typen.

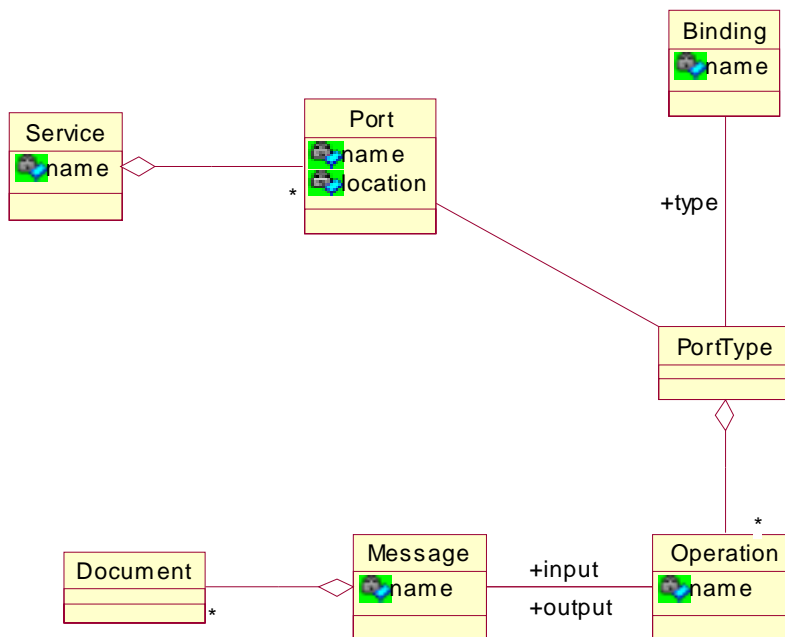


Abbildung ANW-8: Relationen zwischen den Komponenten eines WSDL-Dokuments

Protokoll-Bindings der WSDL-Spezifikation:

- SOAP
- HTTP GET/POST
- MIME

Schema-Sprache

XML-Schema

Ein Beispiel

Das folgende WSDL-Dokument definiert einen Web Service, der eine Operation besitzt: `GetCurrentTemperature`. `GetCurrentTemperature` erwartet eine Postleitzahl und gibt die momentane Temperatur für diese Stadt zurück.

Beispiel ANW-5: WSDL-Dokument und zugehöriger Request

WSDL

```
<?xml version="1.0"?>

<definitions name="CurrentTemp"
  targetNamespace="http://example.com/currenttemp.wsdl "
  xmlns:tns="http://example.com/currenttemp.wsdl "
  xmlns:xsd1="http://example.com/currenttemp.xsd"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns="http://schemas.xmlsoap.org/wsdl/">

  <types>
    <schema
      targetNamespace="http://example.com/currenttemp.xsd"
      xmlns="http://www.w3.org/2001/XMLSchema">
      <element name="CurrentTemperatureRequest">
        <complexType>
          <all>
            <element name="cityCode" type="string"/>
          </all>
        </complexType>
      </element>
      <element name="CurrentTemperature">
        <complexType>
          <all>
            <element name="temp" type="float"/>
          </all>
        </complexType>
      </element>
    </schema>
  </types>

  <message name="GetCurrentTemperatureInput">
    <part name="body">
```

```

        element="xsd1:CurrentTemperatureRequest" />
</message>
<message name="GetCurrentTemperatureOutput">
  <part name="body"
    element="xsd1:CurrentTemperature" />
</message>

<portType name="CurrentTemperaturePortType">
  <operation name="GetCurrentTemperature">
    <input message="tns:GetCurrentTemperatureInput" />
    <output
      message="tns:GetCurrentTemperatureOutput" />
  </operation>
</portType>

<binding name="CurrentTemperatureSoapBinding"
  type="tns:CurrentTemperaturePortType">
  <soap:binding
    style="rpc"
    transport="http://schemas.xmlsoap.org/soap/http" />
  <operation name="GetCurrentTemperature">
    <soap:operation
      soapAction="http://example.com/GetCurrentTemperature" />
    <input>
      <soap:body use="encoded" />
    </input>
    <output>
      <soap:body use="encoded" />
    </output>
  </operation>
</binding>

<service name="CurrentTemperatureService">
  <documentation>Mein erster Service</documentation>
  <port name="CurrentTemperaturePort"
    binding="tns:CurrentTemperatureSoapBinding">
    <soap:address
      location="http://example.com/currenttemp" />
  </port>
</service>

</definitions>

```

Request

```

POST /CurrentTemperature HTTP/1.1
HOST: www.tempserver.com
Content-Type: text/xml; charset="utf-8"
Content-Length: ###

```

```
SOAPAction: " http://example.com/GetCurrentTemperature "
```

```
<soapenv:Envelope  
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/  
">  
  <soapenv:Body>  
    <GetCurrentTemperature>  
      <cityCode>D-12345</cityCode>  
    </GetCurrentTemperature>  
  </soapenv:Body>  
</soapenv:Envelope>
```

Typen

Die Datenstrukturen innerhalb des `<types>`-Element sind abstrakte Typen, d.h. sie sagen nichts über das tatsächliche Format der Datentypen aus. Der Datentyp `CurrentTemperature` z.B. kann innerhalb einer Nachricht mit SOAP-Encoding als XML repräsentiert werden (`<CurrentTemperature>24.5</CurrentTemperature>`) innerhalb eines einfachen HTTP-Request, aber auch als `CurrentTemperature=24.5`.

Nachrichten

Die definierten Typen können nun zu (wiederum abstrakten) Nachrichten zusammengesetzt werden. Nachrichten bestehen aus Teilen, die durch das `<part>`-Element repräsentiert werden. Jeder Teil einer Nachricht ist mit einem bestimmten Datentyp über das `type`-Attribut des `<part>`-Elements assoziiert.

Port-Typen

Ein Port-Typ besteht aus einer Menge von Nachrichten, die Operationen zugeordnet sind. Eine Operation ist die kleinste Arbeitseinheit eines Web Services. Es gibt verschiedene Typen von Operationen (sogenannte **transmission primitives**). Jede Operation kann sich aus einer Input-, einer Output- und einer Fehler-Nachricht zusammensetzen. In unserem Beispiel besteht die Operation `GetCurrentTemperature` aus einer Input-Nachricht namens `GetCurrentTemperatureInput` und einer Output-Nachricht namens `GetCurrentTemperatureOutput`. Auch die Operationsdefinitionen sind abstrakter Natur, d.h. sie sagen nichts über das darunter liegende Protokoll aus. Wie die Operationen auf ein Protokoll gemappt werden, definieren die Bindings.

Bindings

Bindings definieren, wie eine Operation an ein Protokoll gebunden ist. Die Syntax ist ähnlich wie bei der Definition der Port-Typen. In obigem Beispiel wird der `CurrentTemperaturePortType` an das `CurrentTemperatureSOAPBinding` gebunden. Es wird der RPC-Stil (`style="rpc"`) über HTTP (`transport="http://schemas.xmlsoap.org/soap/http"`) genutzt. Die Nachrichten werden in Einklang mit den SOAP-Encoding-Regeln serialisiert (`use="encoded"`).

Ports und Services

Ein Port verknüpft ein Binding mit einer (protokollspezifischen) Adresse. Eine Adresse repräsentiert den Netzwerk-Endpunkt auf dem der Service zur Verfügung steht. Ein Service ist also nichts anderes als eine Kollektion von Ports. Ein über das `<port>`-Element repräsentierter Port referenziert mittels des `binding`-Attributs ein vorher definiertes `<binding>`-Element und mittels seines Inhalts eine Adresse. Im obigem Beispiel wird der Port `CurrentTemperaturePort` an das vorher definierte Binding `CurrentTemperatureSoapBinding` gebunden und über das SOAP-Element `<address>` wird dann die Adresse des Ports in Form einer URL angegeben: `<soap:address location="http://example.com/currenttemp"/>`.

TOOL-TIPP

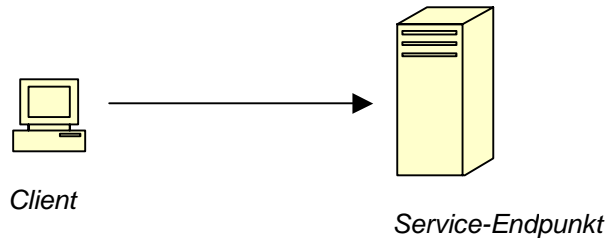
Web Services Toolkits wie z.B. das *IBM Web Services Toolkit* stellen Werkzeuge zur Verfügung, mit denen Sie aus WSDL-Dokumenten Code generieren können und umgekehrt.

URL:

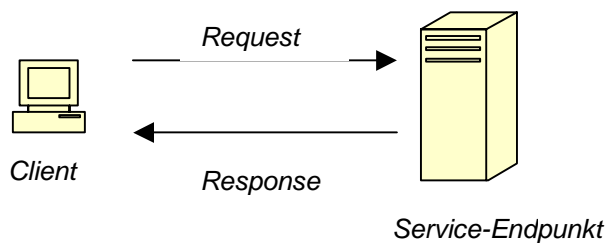
<http://www.alphaworks.ibm.com/tech/webservicestoolkit>

Transmission Primitives

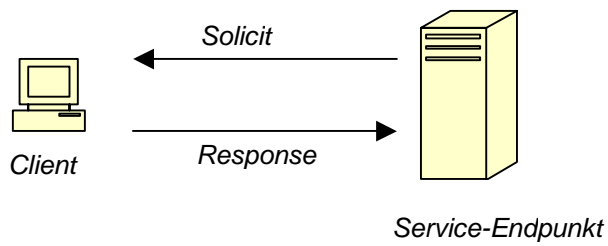
- One-way



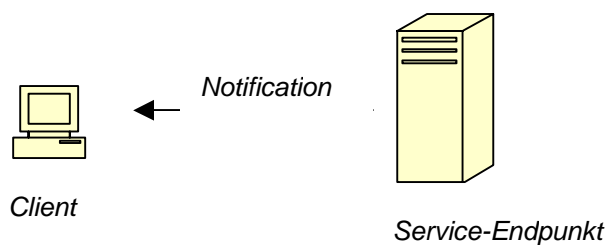
- Request/Response



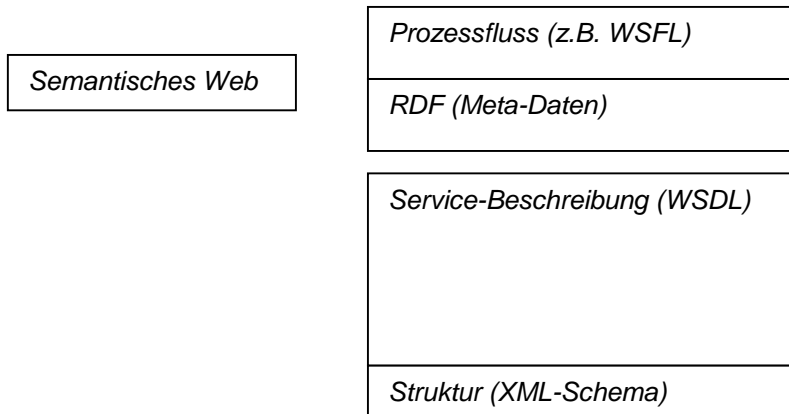
- Solicit/Response



- Notification



Beschreibung eines Web Services



UDDI

UDDI = Universal Description, Discovery and Integration

Das UDDI Projekt wurde im September 2000 von IBM, Microsoft und Ariba ins Leben gerufen (**URL:** <http://UDDI.org>).

Version 1 wurde im September 2000 veröffentlicht.

Die aktuelle Version ist die 3.0-Version vom 19. Juli 2002.

Seither haben sich 300 Firmen als Unterstützer eingetragen (*advisory group*).

Die Spezifikation wird weiterentwickelt von einer Arbeitsgruppe. Mitglieder sind z.Zt. IBM, Microsoft, Ariba, HP, SAP, Oracle, Intel, Accenture, Fujitsu, Verisign, Sun, Compaq, Commerce One und i2 Technologies.

Ziel:

Bereitstellung eines offenen Frameworks für das E-Business. Dadurch soll die Entwicklung des B2B-E-Commerce beschleunigt werden. Es besteht eine gewisse Konkurrenz zur ebXML-Initiative.

- UDDI ermöglicht die Registrierung und Suche nach Services/Produkten, die Firmen zur Verfügung stellen (White-Pages, Yellow-Pages, Green-Pages).
- Über ein API können die Services in eigene Anwendungen integriert werden.

- Die Beschreibung der Web Services erfolgt über WSDL (Web Services Description Language).
- Die Kommunikation erfolgt über SOAP (Simple Object Access Protocol).

UDDI definiert im Grunde einen Standard für webbasierte Registries. Unternehmen können in diese Registries Informationen über sich selbst, über die von Ihnen angebotenen Services und über die technischen Details, wie die Services erreichbar sind, einpflegen. Gleichzeitig können Sie die Registry nach anderen Unternehmen durchsuchen.

Die UDDI-Spezifikation definiert Datenstrukturen zur Modellierung von Geschäftsfeldern und deren Services in der Registry sowie APIs zur Veröffentlichung, Suche und Löschung dieser Informationen. Weitere Spezifikationen richten sich speziell an Betreiber von UDDI-Registries und legen z.B. fest, wie die Informationen zwischen den einzelnen Registries repliziert werden.

Das UDDI-Konzept

Ariba, IBM und Microsoft sowie seit kurzem auch HP und SAP betreiben bereits UDDI-Registries. Die Firmen werden daher auch als **Operators** bezeichnet. Die Registry ist verteilt auf mehrere **Operator Nodes**, die nach einem geregelten Mechanismus abgeglichen werden. Alle UDDI-Registries zusammen bilden die **Universal Business Registry (UBR)**, die salopp auch UDDI-Wolke (*UDDI cloud*) oder Service-Wolke (*service cloud*) genannt wird. Das bedeutet, dass ein einmal auf einem Operator Node veröffentlichter Service auf allen anderen Operator Nodes ebenfalls auffindbar ist ("*register once, publish everywhere*").

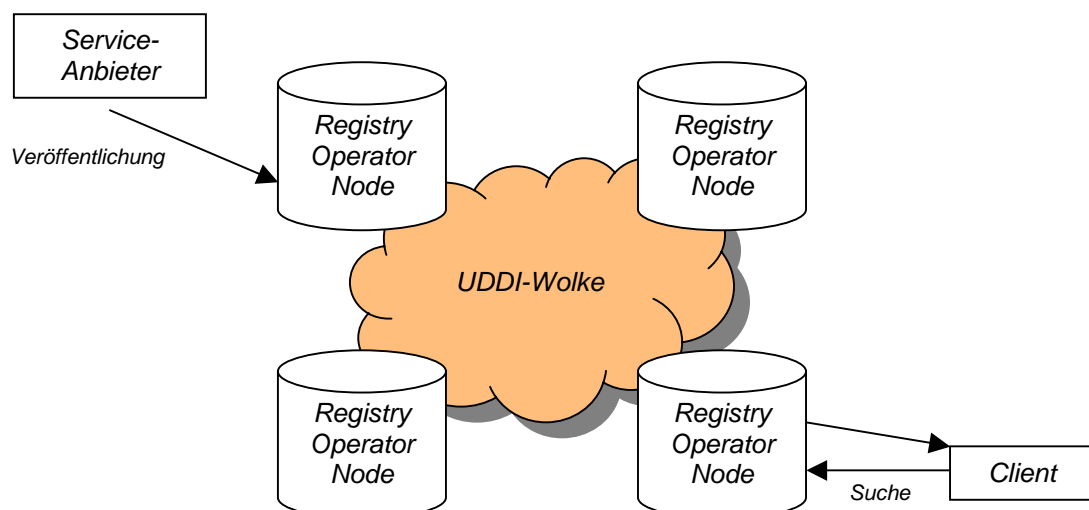


Abbildung ANW-9: Universal Business Registry

Alle Operator Nodes bieten ein Web-Interface an. Die folgende Abbildung zeigt das Web-Interface des MS Operator Nodes:

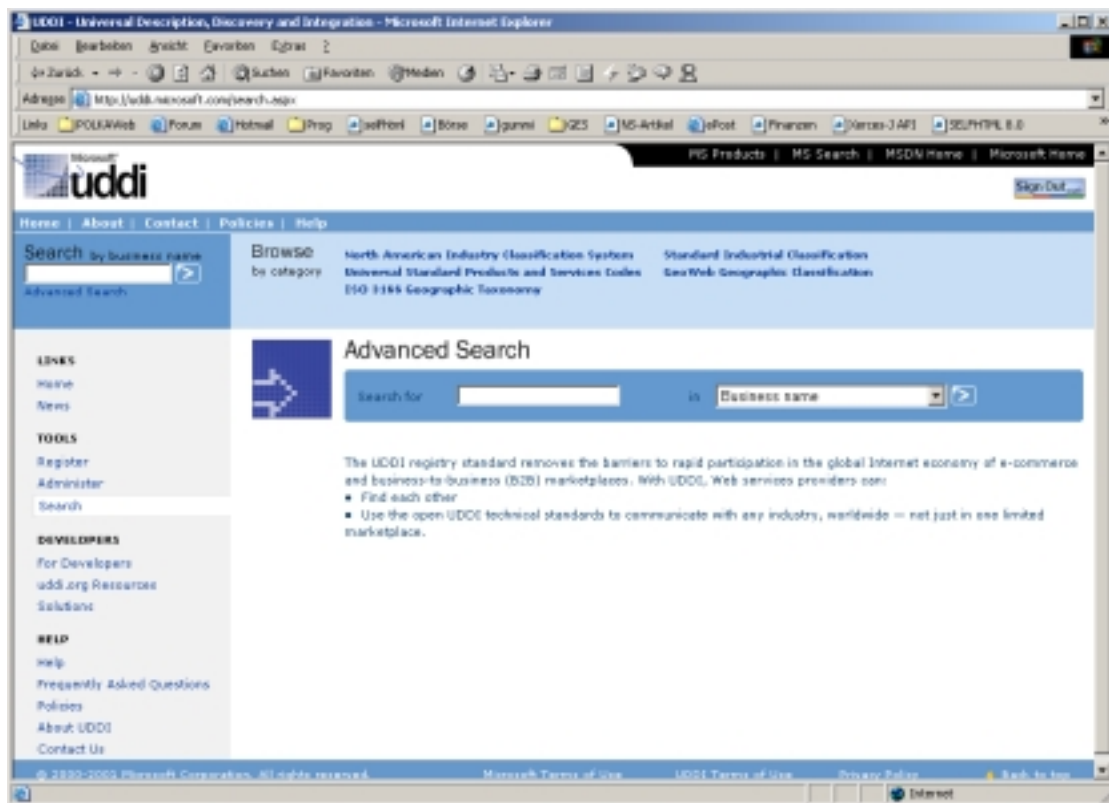


Abbildung ANW-10: Web-Interface des MS Operator Nodes

Neben dieser UBR gibt es auch private Deployment-Szenarien für UDDI-Registries (z.B. B2B-Marktplatz oder Intranet).

Nutzungsmodell von UDDI

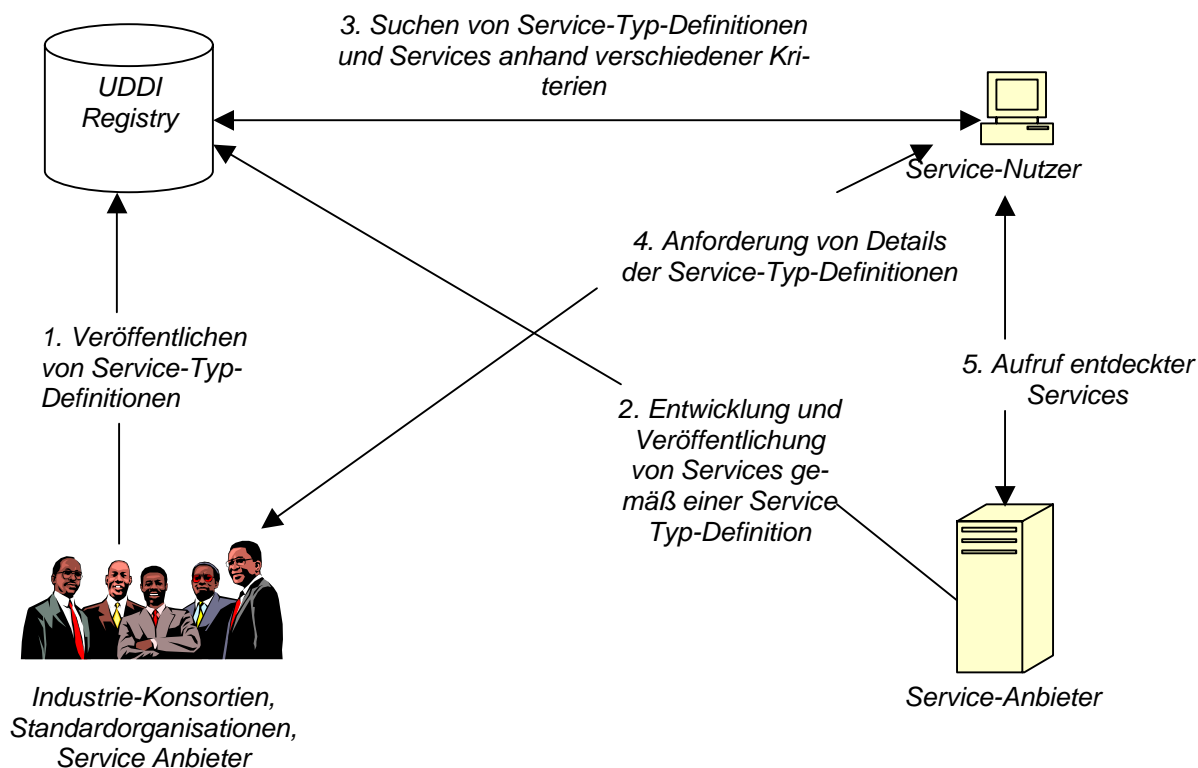


Abbildung ANW-11: Nutzungsmodell von UDDI

Datenarten in einer UDDI-Registry

- *White Pages*
enthalten Informationen wie den Namen der Firma, Kontaktinformationen und menschenlesbare Beschreibungen des Unternehmens.
- *Yellow Pages*
enthalten Informationen, die das Unternehmen klassifizieren. Diese Informationen basieren auf standardisierten Klassifikationsmechanismen wie NAICS (North American Industry Classification System) und UNSPSC (Universal Standard Products and Services Classification).
- *Green Pages*
enthalten technische Informationen über die Services, die ein Unternehmen anbietet. Außerdem enthalten sie Informationen über Geschäftsprozesse, Servicebeschreibungen und Binding-Informationen über den Service.

Das Informationsmodell von UDDI

Die UDDI-Spezifikation definiert 5 Datenstrukturen:

- Business Entity
- Business Service
- Binding Template
- tModel
- Publisher Assertion

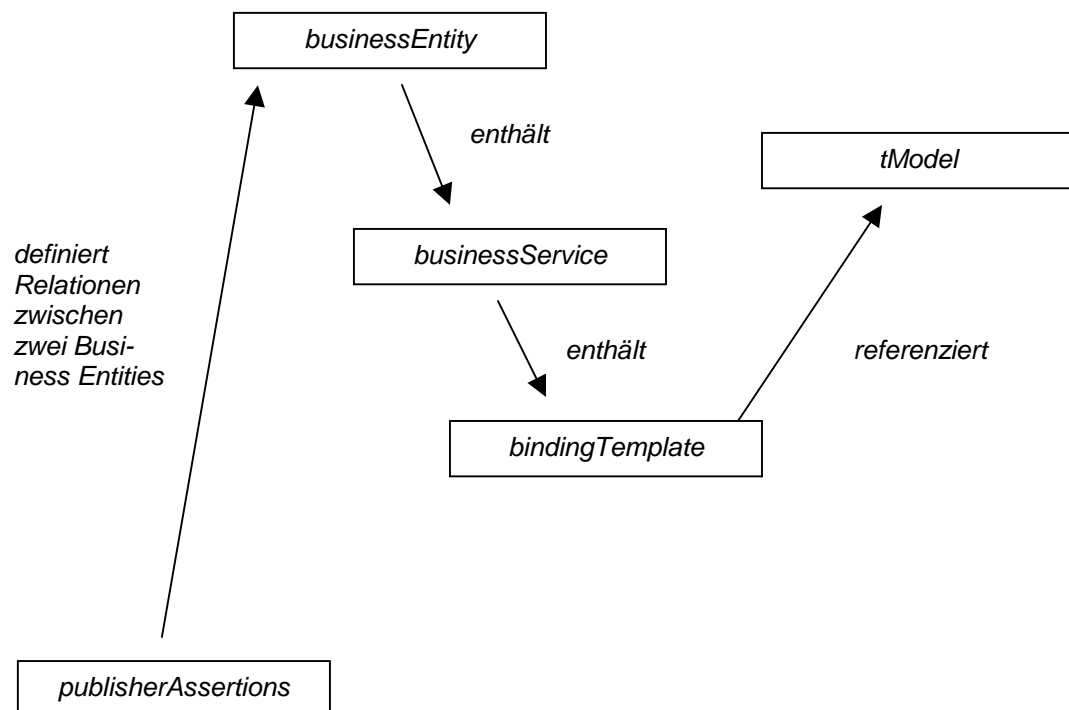


Abbildung ANW-12: Informationsmodell von UDDI

Beispiel ANW-6: Vollständige UDDI-Registrierung

```
<businessDetail
  generic="1.0"
  operator="www.ibm.com/services/uddi"
  truncated="false" xmlns="urn:uddi-org:api">
<businessEntity authorizedName="0100000MDJ"
  businessKey="D2033110-3AAF-11D5-80DC-002035229C64"
  operator="www.ibm.com/services/uddi">
  <discoveryURLs>
    <discoveryURL useType="businessEntity">
      http://www.ibm.com/services/uddi
      /uddiget?businessKey=D2033110-3AAF-11D5-80DC-002035229C64
    </discoveryURL>
  </discoveryURLs>
  <name>IBM Corporation</name>
  <description xml:lang="en">
    At IBM, we strive to lead in the creation,
    development and manufacture of the industry's
    most advanced information technologies, including
    computer systems, software, networking systems,
    storage devices and microelectronics.
  </description>
  <contacts>
    <contact useType="US general">
      <personName>IBM Corporation</personName>
      <phone useType="">1 800 IBM 4YOU</phone>
      <email useType="">askibm@vnet.ibm.com</email>
      <address sortCode="" useType="">
        <addressLine>IBM</addressLine>
        <addressLine>New Orchard Road</addressLine>
        <addressLine>Armonk, NY 10504</addressLine>
      </address>
    </contact>
  </contacts>
  <businessServices>
    <businessService
      businessKey="D2033110-3AAF-11D5-80DC-002035229C64"
      serviceKey="894B5100-3AAF-11D5-80DC-002035229C64">
      <name>Buy from IBM</name>
      <description xml:lang="en">
        This service enables direct purchasing from
        IBM through the ShopIBM web site.
      </description>
      <bindingTemplates>
        <bindingTemplate
```

```

        bindingKey="6D8F8DF0-3AAF-11D5-80DC-
            002035229C64"
        serviceKey="894B5100-3AAF-11D5-80DC-
            002035229C64">
    <description xml:lang="en">
        Register to ShopIBM
    </description>
    <accessPoint URLType="https">
        https://commerce.www.ibm.com/cgi-bin/
        ncommerce/RegisterForm?Krypto=rux5N33YF
        4onmRRYrM2MK6JbZDJCQIqsYZjyy6s330dHEcqYH
        6sVJako2B%2FwXQuQ
    </accessPoint>
    <tModelInstanceDetails>
        <tModelInstanceInfo
            tModelKey="UUID:68DE9E80-AD09-
                469D-8A37-088422BFBC36"/>
        </tModelInstanceDetails>
    </bindingTemplate>
</bindingTemplates>
<categoryBag>
    <keyedReference
        keyName="UNSPSC: Database software"
        keyValue="43161501"
        tModelKey="UUID:DB77450D-9FA8-45D4-
            A7BC-04411D14E384"/>
    <keyedReference
        keyName="UNSPSC: Single optical
            rives"
        keyValue="43172310"
        tModelKey="UUID:DB77450D-9FA8-45D4-
            A7BC-04411D14E384"/>
    <keyedReference
        keyName="NAICS: Other Computer Related
            Services"
        keyValue="541519"
        tModelKey="UUID:C0B9FE13-179F-
            413D-8A5B-5004DB8E5BB2"/>
    </categoryBag>
</businessService>
</businessServices>
<identifierBag>
    <keyedReference keyName="D-U-N-S"
        keyValue="00-136-8083"
        tModelKey="UUID:8609C81E-EE1F-4D5A
            -B202-3EB13AD01823"/>
    </identifierBag>
</businessEntity>
</businessDetail>

```

Die UDDI-APIs

Die UDDI-Spezifikation definiert XML-basierte APIs für die Suche und die Veröffentlichung in UDDI-Registries. Diese APIs sind Schnittstellen, die in SOAP-Envelopes eingebettete XML-Nachrichten akzeptieren. Für die Manipulation dieser XML-Dokumente gibt es bereits Toolkits.

Beispiel ANW-7: Interaktion über die UDDI-API

Suche nach einem Business

```
<find_business generic="2.0"
                xmlns="urn:uddi-org:api_v2">
  <name>XMethods</name>
</find_business>
```

Über das `generic`-Attribut wird die API-Version angegeben. Dieser Aufruf lokalisiert alle Business Entities, die die Zeichenkette `XMethods` als Teil Ihres Namens beinhalten.

Antwort-Nachricht

```
<businessList
  generic="2.0"
  operator="www.ibm.com/services/uddi"
  truncated="false" xmlns="urn:uddi-org:api_v2">
  <businessInfos>
    <businessInfo
      businessKey="BA744ED0-3AAF-11D5-80DC-
                  02035229C64">
      <name>XMethods</name>
      <description xml:lang="en">
        Web services resource site
      </description>
      <serviceInfos>
        <serviceInfo
          businessKey="BA744ED0-3AAF-11D5-80DC
                    -002035229C64"
          serviceKey="D5B180A0-4342-11D5-BD6C
                    -002035229C64">
          <name>XMethods Barnes and Noble Quote</name>
        </serviceInfo>
        <serviceInfo
          businessKey="BA744ED0-3AAF-11D5-80DC
                    -002035229C64"
          serviceKey="ED85F000-4345-11D5-BD6C
```

```

-002035229C64">
  <name>XMethods Pacific Bell SMS Service</name>
</serviceInfo>
<serviceInfo
  businessKey="BA744ED0-3AAF-11D5-80DC
-002035229C64"
  serviceKey="D5921160-3E16-11D5-98BF
-002035229C64">
  <name>XMethods Delayed Stock Quotes</name>
</serviceInfo>
<serviceInfo
  businessKey="BA744ED0-3AAF-11D5-80DC
-002035229C64"
  serviceKey="618167A0-3E64-11D5-98BF
-002035229C64">
  <name>XMethods Currency Exchange Rates</name>
</serviceInfo>
</serviceInfos>
</businessInfo>
</businessInfos>
</businessList>

```

Beispiel ANW-8: Suche nach einem Service

Suchanfrage nach einem Service

```

<find_service
  generic="2.0" xmlns="urn:uddi-org:api_v2"
  businessKey="BA744ED0-3AAF-11D5-80DC-002035229C64">
  <name>XMethods Currency Exchange</name>
</find_service>

```

Antwort-Nachricht

```

<serviceList
  generic="2.0"
  operator="www.ibm.com/services/uddi"
  truncated="false" xmlns="urn:uddi-org:api_v2">
<serviceInfos>
  <serviceInfo
    businessKey="BA744ED0-3AAF-11D5-80DC
-002035229C64"
    serviceKey="618167A0-3E64-11D5-98BF

```

```

                                -002035229C64">
      <name>XMethods Currency Exchange Rates</name>
    </serviceInfo>
  </serviceInfos>
</serviceList>

```

Beispiel ANW-9: Anforderung von Service-Details über die UDDI-API

Anforderung von Detail-Infos

```

<get_serviceDetail generic="2.0"
                  xmlns="urn:uddi-org:api">
  <serviceKey>
    618167A0-3E64-11D5-98BF-002035229C64
  </serviceKey>
</get_serviceDetail>

```

Antwort-Nachricht

```

<serviceDetail
  generic="2.0"
  operator="www.ibm.com/services/uddi"
  truncated="false" xmlns="urn:uddi-org:api_v2">
  <businessService
    businessKey="BA744ED0-3AAF-11D5-80DC-002035229C64"
    serviceKey="618167A0-3E64-11D5-98BF-002035229C64">
    <name>XMethods Currency Exchange Rates</name>
    <description xml:lang="en">
      Returns exchange rates between 2
      countries&apos;currencies
    </description>
    <description xml:lang="en">
      SOAP binding for currency exchange rates service
    </description>
    <bindingTemplates>
      <bindingTemplate
        bindingKey="618474E0-3E64-11D5-98BF-
          002035229C64"
        serviceKey="618167A0-3E64-11D5-98BF-
          002035229C64">
        <description xml:lang="en">
          SOAP binding for currency exchange rates
          service

```

```
</description>
<accessPoint URLType="http">
  http://services.xmethods.net:80/soap
</accessPoint>
<tModelInstanceDetails>
  <tModelInstanceInfo
    tModelKey="UUID:E092F730-3E63-11D5-98BF-
      002035229C64" />
  </tModelInstanceDetails>
</bindingTemplate>
</bindingTemplates>
</businessService>
</serviceDetail>
```

Architektur von Web Services

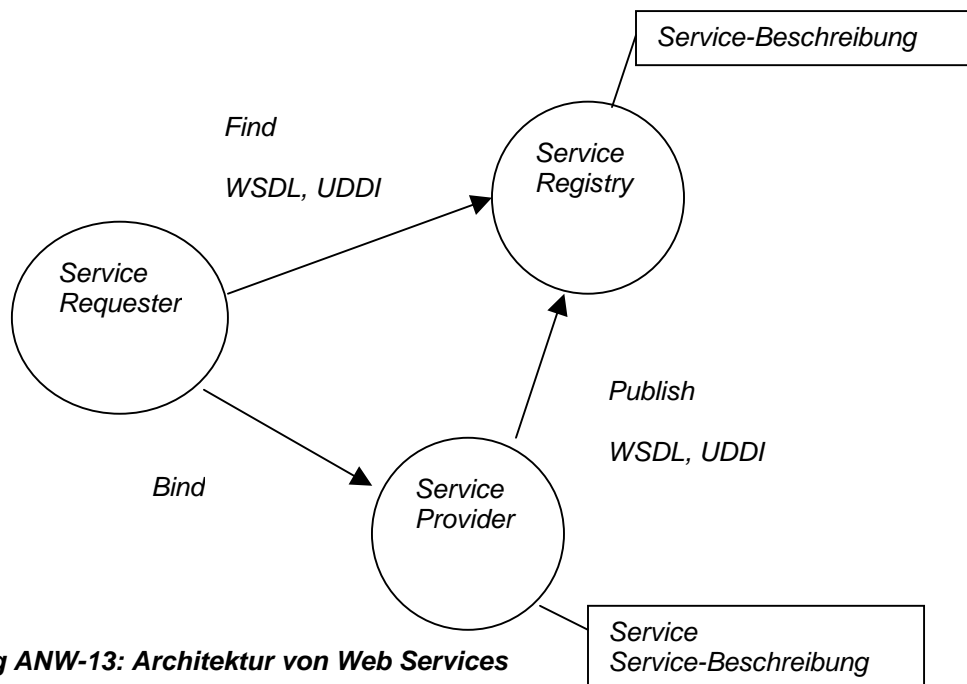


Abbildung ANW-13: Architektur von Web Services

Entwicklungslebenszyklus eines Web Service

- Build
- Deploy
- Run
- Manage

Web Services Stack

WSFL (Web Service Flow Language)	Service-Fluss
UDDI (Universal Description, Discovery an Integration)	Service-Entdeckung
UDDI (Universal Description, Discovery an Integration)	Service-Veröffentlichung
WSDL (Web Services Description Language)	Service-Beschreibung
SOAP (Simple Object Access Protocol)	XML-basiertes Messaging
Transportprotokolle (HTTP/FTP/SMTP/JMS/IIOP usw.)	Netzwerk

Abbildung ANW-14: Web Services Stack

XML und Multimedia: SVG, SMIL

Grafik mit XML: SVG

Was ist SVG?

SVG = Scalable Vector Graphics

SVG ist eine Markup-Sprache, die zweidimensionale Vektorgrafiken in XML-Syntax beschreibt. SVG ist eine Empfehlung des W3C (vom 04.09.2001).

An der Entwicklung von SVG waren alle großen Firmen aus dem Graphikbereich beteiligt (Adobe, Apple, Corel, HP, IBM, Kodak, Macromedia, Netscape, Quark, Sun u.a.)

Ein einfaches Beispiel

Beispiel ANW-5: Einfaches Beispiel

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC " //W3C//DTD SVG 20010904//EN"
"http://www.w3.org/TR/2001/REC SVG
20010904/DTD/svg10.dtd">

<svg xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink">
  <desc>SVG Hello World Example</desc>
  <rect x="50" y="50" width="200" height="150"
style="fill:blue;"/>
  <circle cx="200" cy="200" r="100"
style="fill:yellow;stroke:red;"/>
  <text x="150" y="350" style="font-size:40;">
    Hello World
  </text>
</svg>
```

Das `<svg>`-Element ist das Standard-Wurzelement einer SVG-Datei. Es enthält den eigentlichen Inhalt der SVG-Datei. Hier sind der Namensraum für SVG (<http://www.w3.org/2000/svg>) und der Namensraum für XLink deklariert. Mit XLink wird auf externe Bilder und Elemente verwiesen.

Die Attribute `width` und `height` legen die Abmessungen des Grafikbereichs (Zeichenfläche) in Zeicheneinheiten fest. Werden keine Maßangaben nachgestellt (z.B. cm, mm) wird standardmäßig Pixel angenommen.

Die Konstruktion der geometrischen Basisobjekte erfolgt über Koordinaten (`x` und `y`-Attribute), die Start- und Endpunkt der Linie festlegen. Der Nullpunkt des Koordinatensystems befindet sich in der linken oberen Ecke des Zeichenbereichs.

Ein SVG-Browser kann aufgrund dieser Anweisungen einen entsprechenden Grafiktyp (bzw. Text) anzeigen.

Zusätzliche Formatierungen können über das CSS-Attribut `style` und CSS-Eigenschaften erfolgen.

TOOL-TIPP

*Adobe stellt einen kostenlosen **SVG-Viewer** als Plugin für den Internet Explorer und den Netscape Navigator zur Verfügung.*

URL:

<http://www.adobe.com/svg/viewer/install/main.html>

*Auf der Adobe Webseite können Sie außerdem online ein SVG-Zeichenprogramm (**SVG Draw**) ausprobieren.*

URL:

<http://www.adobe.com/svg/demos/main.html>

Funktionalität von SVG

SVG-Grafikobjekte können:

- skaliert
- mit Bezeichnungen versehen
- gruppiert (in Ebenen, hierarchisch)
- formatiert
- transformiert
- kombiniert
- animiert (=> SMIL)
- gefiltert (=> Lichteffekte, Unschärfe/Schärfe, Überblenden usw.)

- mit Farb- und Schattenverläufen belegt
- dynamisch verändert (SVG-eigenes DOM, Scripting-Unterstützung)

werden.

SVG kennt darüber hinaus einfache Kontrollstrukturen (switch-case), über die Browserabfragen bzw. mehrsprachige Beschriftung von Grafiken realisierbar sind.

Bedeutung von SVG für die Web-Entwicklung

- SVG ist reiner Text.
- SVG-Dateien sind reines XML.
- SVG kann in HTML eingebunden werden.
- SVG-Grafiken können vom Benutzer manipuliert werden.
- Dynamische Größenveränderungen erfolgen ohne Qualitätsverlust.
- Das Format ist kompakt und kann schnell über das Netz übertragen sowie auf dem Bildschirm dargestellt werden.
- Positionierung der Grafiken erfolgt mit CSS 2.
- Grafiken kann ein Titel zugewiesen werden, über den dann Grafikobjekte suchbar sind.

Anwendungsszenarien für SVG

- selbständige SVG-Anwendungen zur Beschreibung ganzer Bilder (insbesondere geeignet für schematische Darstellungen wie Stadtpläne und Wetterkarten)
- SVG-"Fragmente", d.h. grafische Elemente im Körper eines Internet-Dokuments, etwa Piktogramme im Fließtext
- plattformunabhängiges Austauschformat zwischen verschiedenen Grafikprogrammen
- Visualisierung von dynamischen Daten (z.B. grafische Entwicklung der Börsenkurse)
- webbasierte Animationen (z.B. Wetterkarte, die den Durchzug eines Niederschlagsgebiets auf den Monitor zeichnet und dabei die neuesten Messergebnisse einbaut, zudem könnte die Karte interaktives Zoomen zulassen)

Tool-Unterstützung

- SVG-Export: Adobe (Illustrator), Corel (CorelDRAW)
- Autorensystem: Jasc (WebDraw)
- Open Source: Sketch, Mayura Draw
- verschiedene Konverter
- Browserunterstützung: direkte Unterstützung nur Amaya, alle anderen Adobes SVG-Viewer als Plugin

Einbindung von Multimedia-Elementen in XML: SMIL

Was ist SMIL?

SMIL = Synchronized Multimedia Integration Language

SMIL (gesprochen: "smile") ist der Standard des W3C für Multimedia im Web. SMIL-Dokumente sind XML-Dokumente. SMIL erlaubt die Einbindung mehrerer Multimedia-Objekte in eine gemeinsame Präsentation.

Mit SMIL kann der Autor:

- den zeitlichen Ablauf der Präsentation steuern
- die Darstellung auf dem Bildschirm festlegen
- einzelne Objekte mit Hyperlinks versehen

SMIL allein definiert keinen medialen Inhalt, kodiert also weder grafische, akustische oder textuelle Informationen. Die Inhalte selbst befinden sich außerhalb eines SMIL-Dokuments.

Die Empfehlungen

Mittlerweile liegt SMIL bereits in der Version 2.0 vom 07. August 2001 vor. Zudem gibt es seit dem 04. September 2001 eine eigene Empfehlung für SMIL-Animationen, die besonders für die Einbettung in SVG interessant sind.

Einfache Beispiele

Beispiel ANW-6: Struktur einer SMIL-Datei

```
<?xml version="1.0"?>

<smil>
  <head>
    <meta name="copyright" content="Name" />
    <layout>
      <!-- hier kommen die Layout-Tags -->
    </layout>
  </head>
  <body>
    <!-- hier folgen die Media-/Synchronisations-Tags-->
  </body>
</smil>
```

Die Struktur einer SMIL-Datei ähnelt stark der einer HTML-Datei.

Das Wurzelement `<smil>` fasst ähnlich wie das `<html>`-Element in HTML die Elemente `<head>` und `<body>` ein.

Im `<head>`-Element können Sie `<meta>`-Tags mit Informationen über das Dokument hinterlegen. Innerhalb des `<layout>`-Elements können Sie Bereiche im Dokumentfenster festlegen, die später zur Anzeige der Medienelemente (Bilder, Videos...) genutzt werden.

Im `<body>`-Element folgen die Tags, die zur Darstellung der Medienelemente benutzt werden.

Beispiel ANW-7: Grafik 8 Sekunden anzeigen mit SMIL

```
<?xml version="1.0"?>

<smil>
  <!-- Positioniere Grafik für 8 Sekunden -->
  <head>
    <layout>
      <!-- Layout festlegen -->
      <root-layout width="300" height="200" background-
color="F0FCF0" />
      <region id="grafik" left="75" top="50" width="32"
height="32" />
    </layout>
  </head>
  <body>
    <par>
      <!-- Zeige Grafik 8 Sekunden lang an -->
```

```

</par>
</body>
</smil>
```

Über das `<root-layout>`-Element legen Sie die Abmessungen des Fensters fest, in dem die Elemente anzuzeigen sind.

Die Positionierung der einzelnen Medienelemente erfolgt über das `<region>`-Tag. Alle Positionsangaben beziehen sich auf das vom Anzeigebereich (`<root-layout>`) aufgespannte Koordinatensystem. Der Nullpunkt liegt in der linken oberen Ecke und alle Abstandsangaben erfolgen in Pixel oder % des Anzeigebereichs.

Eine Region ist ein rechteckiger Bereich innerhalb der Anzeigefläche und dient als Container für ein Medienelement. Es können mehrere Regionen definiert sein, die alle über das `id`-Attribut ansprechbar sind.

Die erforderlichen SMIL-Elemente werden im `<body>`-Bereich aufgeführt. Dabei kommen entweder die Tags `<seq>...</seq>` oder `<par>...</par>` zum Einsatz. Sie geben an, ob die Elemente sequenziell (`<seq>`) oder parallel (`<par>`) auszuwerten sind.

Eine Grafik fügen Sie über ein ``-Tag ein und mit dem Attribut `dur` bestimmen Sie die Dauer der Anzeige. Neben `dur` kennt SMIL auch noch ein `begin`- und ein `end`-Attribut, mit dem sich die Anfangs- und Endzeit auf der Zeitachse spezifizieren lässt.

TOOL-TIPP:

Sowohl der *Windows Media Player* als auch der *Real Player* unterstützen SMIL. Beiden Programmen ist jedoch gemeinsam, dass die betreffenden SMIL-Dateien von Servern heruntergeladen werden müssen, ein umständliches Verfahren, wenn es um das Testen von SMIL-Dateien geht. SOJA ist ein kostenlos verfügbares Java-Applet der Firma Helio, über welches sie SMIL-Präsentationen in HTML-Dateien integrieren können.

URL:

<http://www.helio.org/products/smil/>

Einbindung multimedialer Objekte: HTML versus SMIL

HTML	SMIL
Dateiauswahl über <code>src</code> -Attribut	Dateiauswahl über <code>src</code> -Attribut
<code>type</code> -Attribut legt Format der Datei fest	<code>type</code> -Attribut legt Format der Datei fest
Einbindung rechteckiger Bildausschnitte über CSS in Webpräsentationen	Einbindung rechteckiger Bildausschnitte über CSS in Webpräsentationen
deklarativ: Bilder können in Fließtext integriert werden, aber die Ausgaben temporärer Daten wie Audio und Video müssen explizit durch Benutzerinteraktion ausgelöst werden	zeitliche Steuerung möglich
Videos laufen in einem Extra-Fenster	alle Ausgaben erfolgen innerhalb desselben Displays
nur räumliche Ausschnitte möglich	auch zeitliche Abschnitte möglich
Hyperlinks können auf räumliche Teile von Dokumenten verweisen	SMILs <code>href</code> -Attribut kann auch auf zeitliche Teile eines SMIL-Dokuments verweisen (z.B. "vorspulen")

Herstellerunterstützung

- Microsoft (Media Player, Internet Explorer 5.5)
- Real (Real Player)
- Apple (QuickTime-Player)

XML und Content-Management

Klassisches Dokumentenmanagement

Dokumentenmanagement ist die menschliche Tätigkeit der Erfassung, Bearbeitung, Verwaltung und Speicherung von Dokumenten.

Dokumentenmanagement-Systeme bieten u.a. folgende Unterstützung an:

- Versionsverwaltung
- Archivierung
- individuelle Rechtevergabe
- zunehmend Workflow-Management
- Indexierung
- Extraktion

Dokumentenmanagement-Systeme behandeln das Dokument als Black Box.

Folge:

Im klassischen Dokumentenmanagement-System müssen für jedes Ausgabemedium unterschiedliche Formate vorgehalten werden, d.h. für jedes gewünschte Ausgabemedium muss es eine vollständige, fertige Kopie des gesamten Dokuments geben. Ein Update des Inhalts erfordert die Erstellung einer neuen Version des Dokuments für jedes Ausgabeformat.

XML kann als Format zur Beschreibung der Metadaten eines Dokuments bzw. der Beziehungen der Dokumente untereinander zum Einsatz kommen.

Content-Management

Was ist Content-Management?

Unter Content-Management versteht man die systematische und strukturierte Beschaffung, Erzeugung, Aufbereitung, Verwaltung, Präsentation, Verarbeitung, Publikation und Wiederverwendung von Inhalten.

Ziel:

Beim Content-Management werden - anders als beim Dokumentenmanagement - einzelne Komponenten eines Dokuments verwaltet. Zentraler Aspekt ist die Wiederverwendung der Komponenten in verschiedenen Dokumenten.

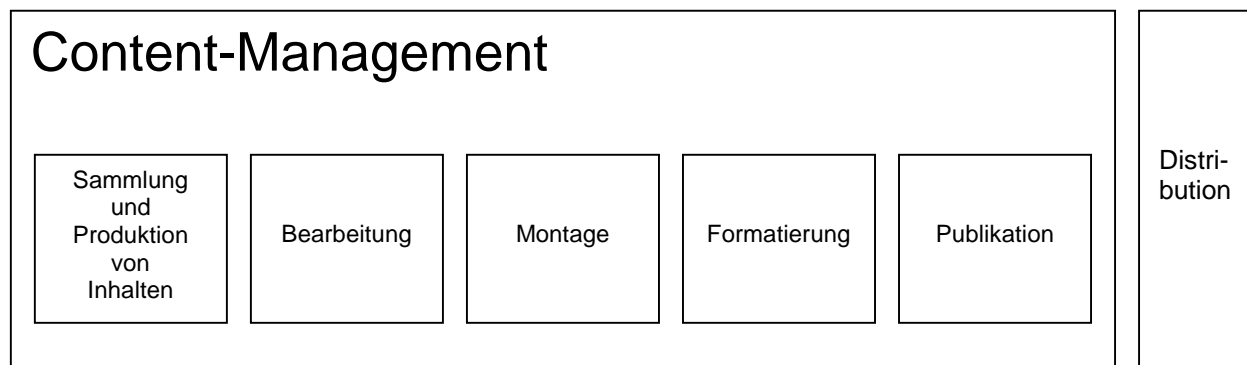


Abbildung ANW-15: Content-Management als Prozess

Content-Management-System

Ein Content-Management-System, das seinen Namen verdient, geht über ein Dokumentenmanagement-System in zweierlei Hinsicht hinaus.

- Content-Management-Systeme halten eine einzige Kopie eines Dokuments vor, die den Inhalt in abstrahierter Form darstellt. Diese wird verwendet um die verschiedenen Ausgabeformate zu erzeugen.
- Content-Management-Systeme verwalten kleinere Einheiten als Dokumente, sie zielen auf die Verwaltung der einzelnen Komponenten eines Dokuments.

Content-Management-Systeme basieren auf der Trennung von Form und Inhalt und XML spielt dabei die zentrale Rolle als standardisiertes Daten- bzw. Dokumentenformat.

=> im Idealfall müssen sich Autoren nur noch um die Inhalte kümmern.

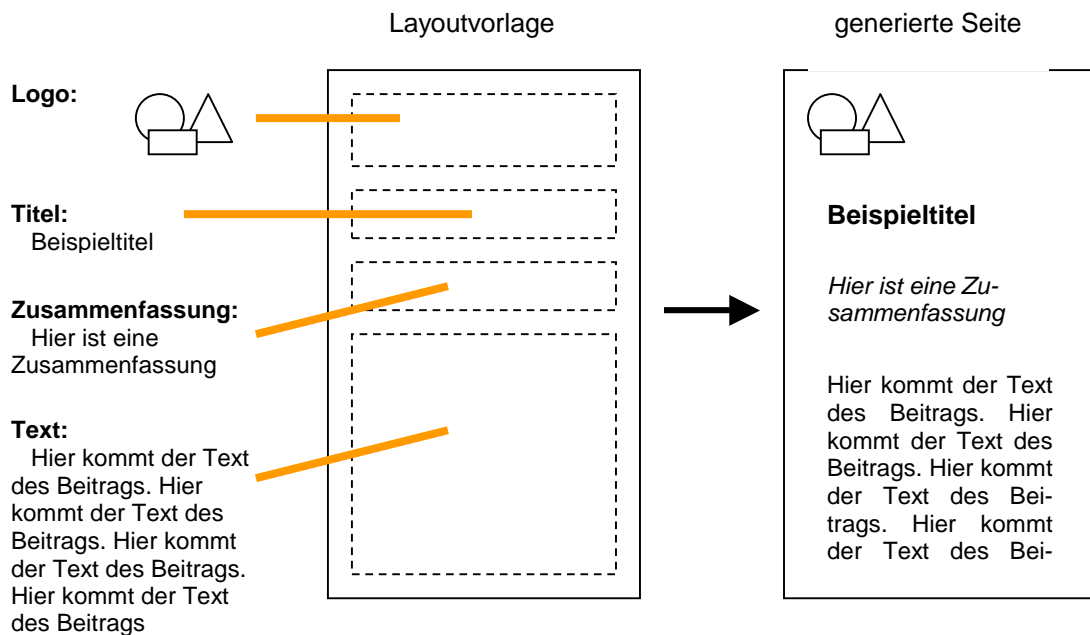


Abbildung ANW-16: Trennung von Form und Inhalt bei Content-Management-Systemen

Mögliche Funktionalitäten eines Content-Management-Systems

- strukturierte Speicherung von typisiertem Inhalt
- Bereitstellung von Metainformationen über die Inhalte
- Zugangskontrolle
- Protokollfunktionen
- Datensicherung
- Rollback
- Mehrplatzfähigkeit
- Check-In und Check-Out
- Anfragefunktionen
- Massenoperationen
- Bearbeitung und Verifikation
- Aufzeichnung atomarer Änderungen
- Aggregation und Beziehungen
- Versionsverwaltung

- Mehrsprachenfähigkeit
- Workflow
- Gestaltung
- Verarbeitungsfunktionen
- Fremdformatwandlung

Vorteile eines Content-Management-Systems

- ein beschleunigter redaktioneller Workflow
- die erhöhte Qualitätssicherung des verwalteten Contents
- die Wiederverwendbarkeit von Content
- der vereinfachte Im- und Export von Content
- die Crossmedia Nutzung
- sowie die Einsparungen im künftigen Betrieb und in der Wartung des Informationsportals.

Ein Nachteil ist die hohe Investitionssumme.

Bei der Auswahl eines Content-Management-Systems sollten Sie v.a. folgende Funktionen genauestens evaluieren:

- *Daten-Repository*
- *Workflow*
- *Benutzerschnittstelle*

Web-Content-Management-Systeme

Web-Content-Management-Systeme sind eine inzwischen sehr große Gruppe von Content-Management-Systemen, die sich auf die optimierte Verarbeitung von Inhalten für das Internet spezialisiert hat.

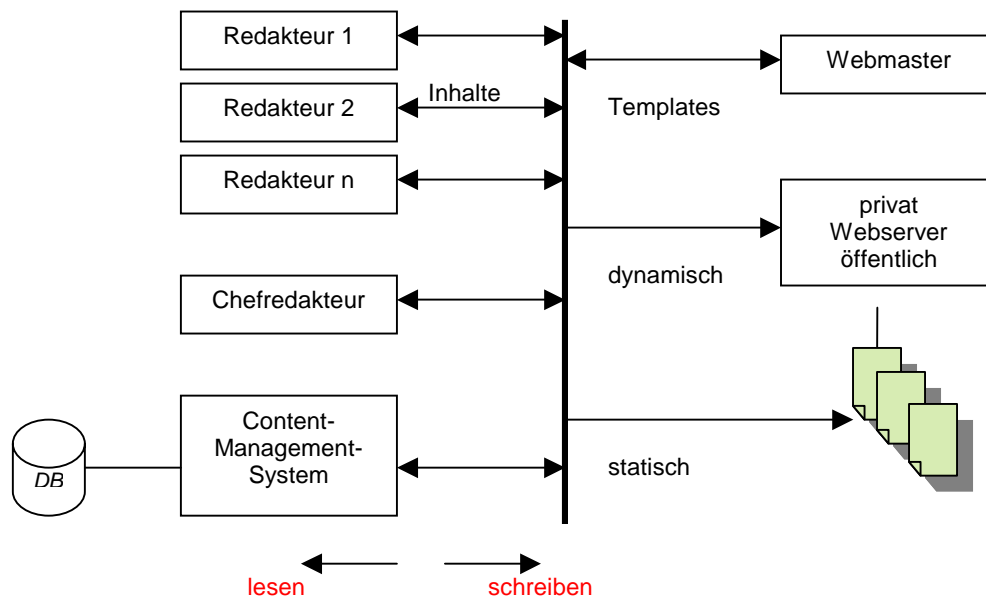


Abbildung ANW-17: Schematischer Aufbau eines Web-Content-Management-Systems

Beispiele für Web-Content-Management-Systeme

Produkt	Hersteller
Imperia	Imperia Software Solutions
InfoOffice, Professional	InfoOffice AG
NPS	Infopark AG
pirobase 4	Pironet
SixCMS 3	Six Offene Systeme GmbH
Storyserver	Vignette
VIP ContentManager	Gauss Interprise SG
OpenCMS	Open CMS Group

Eine aktuelle Marktübersicht von Content-Management-Systemen finden Sie unter <http://www.contentmanager.de/>!

XML und die Wissenschaft: MathML und CML

MathML

Was ist MathML?

MathML = Mathematical Markup Language

MathML ist eine XML-basierte Sprache, mit der sich mathematische Formeln erstellen lassen. Mit MathML können Sie - einen entsprechenden Browser vorausgesetzt - Ausdrücke wie $x + 2$, x^2 usw. in Web-Dokumente integriert werden.

Zielgruppe:

Mathematiker, Techniker, Ingenieure

TOOL-TIPP:

*Der kostenlose experimentelle Browser des W3C **Amaya** kann MathML darstellen und stellt auch einen grafischen Editor zur Erstellung von MathML-Formeln zur Verfügung. Sobald Sie ein neues Dokument anlegen, können Sie mathematische Zeichen in Form von Schablonen über Schaltflächen einer Symbolleiste abrufen.*

URL:

<http://www.w3.org/Amaya/>

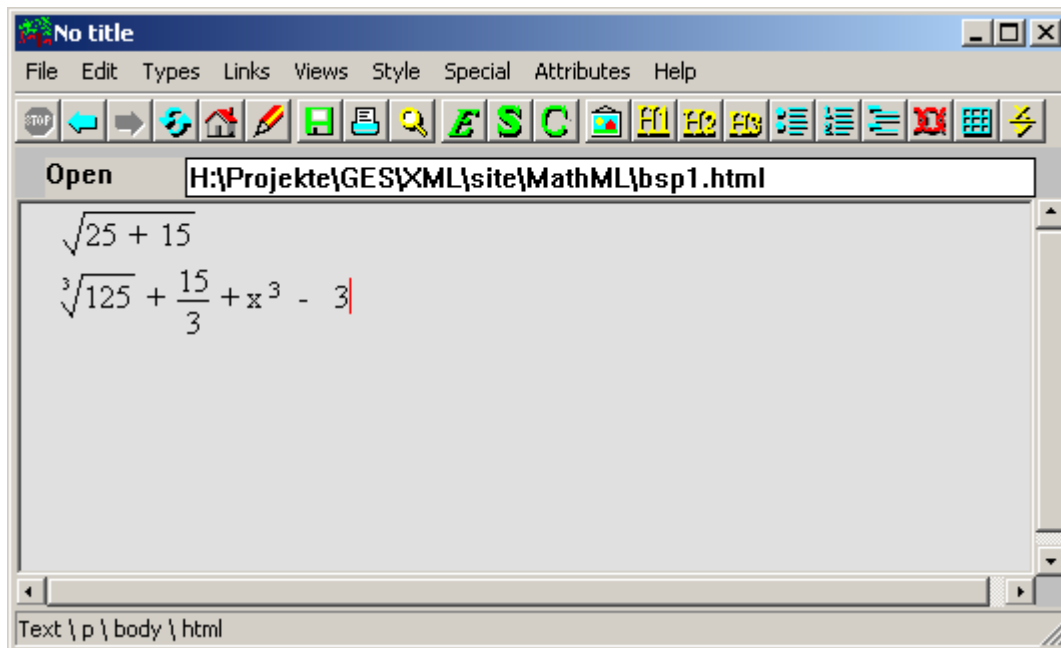


Abbildung ANW-18: Mathematische Formeln im Amaya

Beispiel ANW-8: Ausdruck $(a+b)^2$ in MathML

```

<msup>
  <mfenced>
    <mrow>
      <mi>a</mi>
      <mo>+</mo>
      <mi>b</mi>
    </mrow>
  </mfenced>
  <mn>2</mn>
</msup>

```

Das `<mrow>`-Element wird benutzt, um einen Ausdruck, der in einer Zeile stehen soll, aufzunehmen. Das Element `<mfenced>` erlaubt, in MathML Klammern zu setzen. Über `<mi>` werden Variablen abgebildet, während `<mo>` Operatoren definiert. Zur Darstellung der Exponentiation der Klammer dient das Element `<msup>`. Es besitzt zwei Unterelemente, die den Basisausdruck (hier der geklammerte Ausdruck) und den Exponenten (`<mn>`) festlegen.

MathML unterscheidet zwischen

- präsentationsorientiertem Markup (*Presentation Markup*)
- inhaltsorientiertem Markup (*Content Markup*)

Beispiel ANW-9: $(a+b)^2$ in inhaltsorientiertem Markup

```
<msup>
  <mfenced>
    <mrow>
      <mi>a</mi>
      <plus/>
      <mi>b</mi>
    </mrow>
  </mfenced>
  <mn>2</mn>
</msup>
```

Programmunterstützung

- Maple
- Mathematica
- Mathcad
- Mozilla 1.0

CML

Was ist CML?

CML = Chemical Markup Language

CML ist das Pendant zu MathML (*HTML with Molecules*) auf dem Feld der Chemie. CML dient der Beschreibung und Darstellung von Informationen aus dem Bereich der Chemie, speziell auf molekularer Ebene. So kann man CML z.B. dazu verwenden, um zwei- oder dreidimensionale Modelle von molekularen Strukturen zu zeichnen. CML wurde in Zusammenarbeit mit der Open Molecule Foundation (OMF) entwickelt.