Entwicklung und Anwendung eines Webservice zur Konvertierung von XML-Dokumenten

Peter Buxmann, Erik Wüstner, Robert Barsch, Christian Rödel, Sven Schade

Technische Universität Freiberg

Lehrstuhl für Wirtschaftsinformatik

WWW: http://www.bwl.tu-freiberg.de/wi

E-Mail: peter.buxmann|erik.wuestner@bwl.tu-freiberg.de; robert.barsch|christian.roedel|sven.schade@x-act.org

Zusammenfassung: In diesem Beitrag zeigen wir, wie es auf Basis von offenen Standards und Open-Source-Software heute möglich ist, moderne EDI-Lösungen zu entwickeln. Eine Schlüsselrolle spielen dabei die Extensible Markup Language (XML) sowie komplementäre Technologien. Praxiserfahrungen haben jedoch gezeigt, dass auch die Nutzung von XML das Problem der Konvertierung zwischen unterschiedlichen Formaten, wie z. B. xCBL und OAGIS, nicht löst. Vor diesem Hintergrund stellen wir den Webservice <x:act> vor, der eine solche Übersetzung zwischen unterschiedlichen XML-Formaten durchführt.

Schlüsselworte: Standardisierung, Konvertierung, XML, XSLT, EDI, Webservice

1 Einleitung

Der elektronische Austausch von Geschäftsdokumenten per Electronic Data Interchange (EDI) hat in vielen Branchen zu erheblichen Kosteneinsparungen geführt. Trotz vieler „Erfolgsgeschichten“ [Emme93, S. 193ff.] ist EDI heute überwiegend in großen Unternehmen verbreitet. Gründe für den Rückstand in kleinen und mittelständischen Unternehmen sind insbesondere die hohen Implementierungskosten sowie fehlendes Know-how.

Vor diesem Hintergrund sind internet-basierte EDI-Formen entstanden, wie etwa XML/EDI. Hierbei werden die Geschäftsdokumente mithilfe von XML beschrieben und über Internet-Protokolle übertragen. Eine Zielsetzung, die mit solchen Lösungen verfolgt wird, ist die Senkung der Eintrittsbarriere für die Teilnahme an EDI-Netzwerken. Die Überlegung dahinter ist, dass Teilnehmer an XML/EDI-Netzwerken auf die Anwendung offener Standards und bei Bedarf auf Open-Source-Komponenten zurückgreifen können und keine weitergehenden Investitionen in EDI-Konverter notwendig sind. Der Austausch von Geschäftsdokumenten per XML ist jedoch auch mit Problemen verbunden, die daraus resultieren, dass mittlerweile auf Basis von XML eine Vielzahl von Business-Vokabularen entstanden ist [Weit+01]. Das Ziel des Beitrags besteht in der Vorstellung eines Webservice, der eine solche Konvertierung zwischen verschiedenen XML-Standards durchführt.

Im zweiten Kapitel zeigen wir zunächst, wie es mithilfe offener Standards und Open-Source-Komponenten heute möglich ist, praxistaugliche XML/EDI-Lösungen zu entwickeln. Gegenstand des dritten Kapitels ist die Vorstellung des Webservice <x:act> zur Konvertierung zwischen unterschiedlichen XML-Formaten. Dabei gehen wir sowohl auf die technische Realisierung als auch die Anwendung des <x:act> Webservice ein. Der Artikel schließt mit einer Zusammenfassung und einem Ausblick auf zukünftige Forschungsaktivitäten.

2 Grundlagen und Anwendung von XML/EDI

In diesem Kapitel wollen wir die Anwendung von XML/EDI untersuchen. Dabei verstehen wir unter XML/EDI den elektronischen Austausch von mit XML beschriebenen Geschäftsdokumenten über Netze [Stef00]. XML/EDI-Lösungen besitzen im Vergleich zu traditionellen EDI-Lösungen [Neub94] den Vorteil, dass sie vollständig auf Basis offener Standards und Open-Source-Komponenten erstellt werden können. Dies wiederum

führt zu Kosteneinsparungspotenzialen im Vergleich zu herkömmlichen Lösungen, bei denen etwa die Notwendigkeit besteht, einen EDI-Konverter zu beschaffen [Alpa02, S. 36]. Durch die Nutzung des Internet fallen auch die für traditionelles EDI typischen Kosten für die Nutzung von Value Added Networks (VAN) weg.

Neben XML kann es sich bei den offenen Standards zum Aufbau von XML/EDI-Lösungen etwa um HTTP zum Versenden oder Abholen der Geschäftsdokumente handeln. Die Verarbeitung der XML-Dokumente kann auf Basis komplementärer Standards, wie DOM oder SAX, sowie darauf aufbauender Open-Source-Software-Komponenten, wie etwa des Apache Xerces Parsers, erfolgen. Ein einfaches Anwendungsszenario ist in der folgenden Abbildung dargestellt: Der Sender einer Nachricht erzeugt eine mit XML beschriebene Bestellung. Diese wird mit HTTP, FTP oder SMTP bzw. deren sichere Pendants HTTPS, SFTP oder SSMTP an den Empfänger übertragen. Der Empfänger hat nun zwei Möglichkeiten: Entweder er kann sich das XML-Dokument in seinem Browser anzeigen lassen oder er kann das Dokument darüber hinaus mit einem XML-Parser verarbeiten und in seine Inhouse-Systeme integrieren.

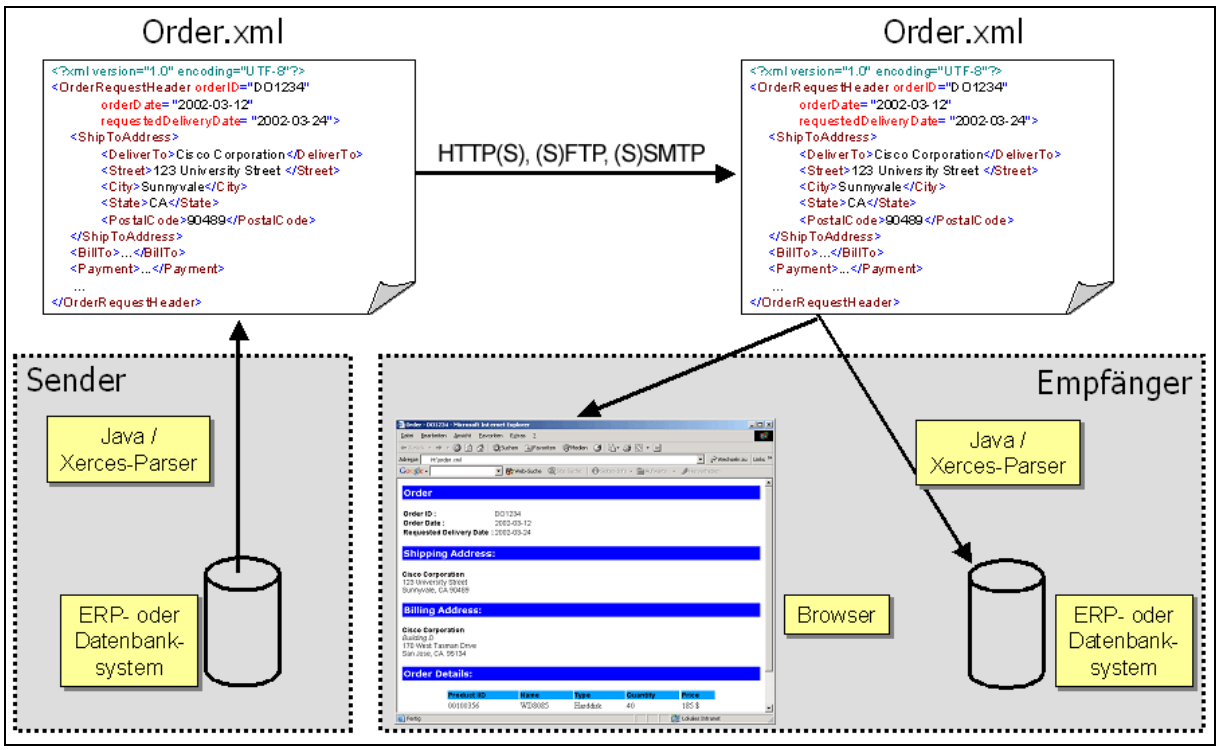

Abbildung 1: Einfache Anwendung von XML/EDI

Dieses Konzept bildete die Grundlage für den Einsatz von XML/EDI bei Lufthansa AirPlus Servicekarten GmbH für den Austausch von Rechnungen. Bei der Implementierung zeigte sich, dass das Konvertierungsproblem, also die Übersetzung zwischen verschiedenen Formaten und Standards, auch durch XML nicht ohne weiteres lösbar ist. Der Grund liegt darin, dass XML lediglich eine allgemeine Sprache zur Beschreibung von Dokumenten ist. Dabei ist mittlerweile eine nahezu unüberschaubare Vielzahl an Standardisierungsinitiativen entstanden, die es sich zum Ziel gemacht haben, Inhalte und Strukturen von Geschäftsdokumenten auf Basis von XML zu definieren [HoJu02, S. 34, Weit+01]. Beispiele sind xCBL, OAGIS oder SAP IDOC.

Grundsätzlich stehen nun zwei Alternativen zur Verfügung, um eine Kommunikation zwischen den Akteuren zu ermöglichen: Entweder die Beteiligten einigen sich auf ein Format oder es wird eine Konvertierung zwischen den verschiedenen Formaten durchgeführt. Bei beiden Alternative fallen Implementierungskosten an, die sich aus Aufwendungen für zusätzliche Hardware, Kosten für Software-Anpassungen und -Neuentwicklungen und Schulungskosten des Personals zusammensetzen [MuKe02, S. 1304].

Die erstgenannte Alternative der Standardisierung wird in der Regel zu hohen Koordinationskosten führen, die daraus resultieren, dass sich die Beteiligten auf einen Standard einigen [BeWe93, S. 123]. Dieser Einigungsprozess wird üblicherweise dadurch erschwert, dass die verfügbaren Standards für die verschiedenen Akteure unterschiedlich gut geeignet sind. Diese Interessenvielfalt erfordert meistens einen hohen Koordinationsaufwand. Zudem führt die Verwendung eines gemeinsamen Standards zu Opportunitätskosten aufgrund des Nutzenverzichts, der den Beteiligten dadurch entsteht, dass nicht alle individuellen Anforderungen an den Standard erfüllt werden können [BrWh85, BuKö98]. Dies kann sich beispielsweise darin äußern, dass der

Standard nicht alle benötigten Informationen enthält oder aber zwingend Informationen verlangt, die einzelnen Benutzern als unnötig erscheinen oder die von diesen nicht oder nur mit unverhältnismäßig hohem Aufwand generiert werden können. Die Opportunitätskosten der Standardisierung sind umso höher, je stärker die Anforderungen der Benutzer von der Funktionalität des gemeinsamen Standards abweichen.

Demgegenüber fallen bei der Alternative der Konvertierung zwischen unterschiedlichen Formaten Konvertierungskosten an, die sich aus Erstellungskosten für die zu verwendenden Konverter, den Betriebskosten für die Ausführung der Konvertierungsprozesse und den Kosten, die aus unzulänglichen Konvertierungsergebnissen entstehen, zusammensetzen.

Im Folgenden werden wir einen Webservice vorstellen, der die Konvertierung zwischen verschiedenen XML-Formaten unterstützt, und somit anstrebt, einen Beitrag zur Senkung der Konvertierungskosten zu leisten. Mithilfe solcher Konvertierungsansätze lässt sich die Bedeutung von Standards im EDI-Bereich zumindest teilweise relativieren [PiNe01, S.34f].

3 <x:act> – Ein Webservice für die Konvertierung von XML-Dokumenten

3.1 Offene Standards und Open-Source-Software als Grundlage der Entwicklung von <x:act>

Die Zielsetzung unseres Webservice <x:act> besteht darin, einen Dienst zur Konvertierung zwischen verschiedenen in XML beschriebenen Geschäftsdokumenten anzubieten. Wie die im letzten Abschnitt vorgestellte XML/EDI-Lösung, basiert <x:act> (http://www.x-act.org) vollständig auf offenen Standards und Open-Source-Software.

In der Literatur finden sich verschiedene Definitionsansätze für Webservices [Bett01, KrLu02]. Wir verstehen unter einem Webservice allgemein eine Anwendung oder Software-Komponente, die für andere Client-Anwendungen Dienste über das Web anbietet. Die Kommunikation zwischen Nutzer und Anbieter eines Webservice erfolgt mithilfe XML-basierter Nachrichten. Die in diesem Umfeld benötigten Standards sind im Wesentlichen:

- Webservice Description Language (WSDL) ist eine XML-Sprache zur maschinenlesbaren Beschreibung von Webservices, die aus einer Initiative der Unternehmen Microsoft, Ariba und IBM entstanden ist und an das World Wide Web Konsortium (W3C) übergeben wurde [W3C02b]. Das WSDL-Dokument eines Webservice spezifiziert unter anderem, welche Daten der Webservice erwartet, welche Daten vom Webservice zurückgeliefert werden sowie über welche Protokolle und unter welcher URL der Webservice erreichbar ist.
- SOAP [W3C02a] ist aus XML-RPC hervorgegangen und wird beim W3C unter Federführung von IBM und Microsoft weiterentwickelt. SOAP ist ein einfach gehaltenes XML-basiertes Protokoll zum Austausch von strukturierten Informationen in verteilten Umgebungen wie dem Internet. Ähnlich wie bei anderen verteilten Ansätzen, wie z. B. DCOM oder IIOP/CORBA, können Methoden, Services, Komponenten und Objekte auf entfernten Servern aufgerufen werden. Im Gegensatz zu diesen Ansätzen, die binäre Formate verwenden, nutzt SOAP das Textformat in Verbindung mit XML, um den Datenaustausch zu strukturieren. In Anlehnung an Schichtenmodelle übernimmt SOAP bei der Kommunikation zwischen Webservice-Nutzer und -Anbieter die Verpackung, Strukturierung und Kapselung der auszutauschenden Informationen. Physisch werden diese Informationen über beliebige Transportprotokolle, wie HTTP(S) oder (S)SMTP, übertragen. Ein Nachteil der Verwendung XML-basierter Protokolle ist der im Vergleich zu proprietären Protokollen höhere Bedarf an Bandbreite zum Verschicken der Daten [Beut02, S. 29].
- Universal Description, Discovery and Integration (UDDI) bezeichnet einen Verzeichnisdienst, bei dem Anbieter ihre Webservices registrieren lassen und Anwender nach entsprechenden Diensten suchen können [Over02].
- Webservice Choreography Interface (WSCI) beschreibt die Nachrichtenflüsse, die von verschiedenen, interagierenden Webservices [Minz+02, S. 11] produziert und verarbeitet werden. Durch diesen Standard werden Nachrichtenzusammenhänge abgebildet sowie Transaktionsbeschreibungen und Ortsangaben von Webservices ermöglicht. Dies lässt einen Blick auf die Gesamtheit der an einem komplexen Geschäftsprozess beteiligten Webservices zu [Lang03, S. 466 – 478].

Bisher nutzt <x:act> von diesen Standards lediglich WSDL und SOAP. Sobald der Webservice in Pilotprojekten erfolgreich getestet wurde, werden wir ihn auch in UDDI registrieren. Zur Nutzung von WSCI besteht derzeit keine Notwendigkeit, da von <x:act> kein anderer Webservice benutzt wird.

Neben diesen offenen Standards haben wir bei der Entwicklung auf Open-Source-Software gesetzt. Bei der Implementierung fiel die Wahl auf die Verwendung des Java Webservice Developer Packs (JWSDP) von Sun Microsystems [Röwe02, S. 126]. Die Konvertierung zwischen den Formaten, d. h. die originäre Funktionalität des Webservice, wurde als Java-Servlet implementiert. Dabei wurde die Java API for XML Messaging (JAXM) von Sun Microsystems verwendet. Wir haben uns gegen RPC und für einen dokumentenbasierten Ansatz entschieden, da dies mehr Flexibilität bezüglich der Kommunikation mit dem Webservice bietet. So wollen wir damit in Zukunft etwa auch das ebXML-Framework[1] unterstützen. Als Servlet-Container kommt Apache Tomcat zum Einsatz. Zudem haben wir den Apache Webserver sowie MySQL als Datenbank verwendet. Zum Verarbeiten der XML-Dokumente haben wir auf Apache Xerces und Xalan zurückgegriffen.

Neben der Verwendung dieser Komponenten ist <x:act> selbst auch ein Open-Source-Projekt, d. h., die gesamte Software und deren Quellcode ist auf Basis einer Apache-ähnlichen Lizenz (siehe http://www.x-act.org/LICENSE) frei verfügbar.

Um potenziellen Nutzern die Implementierung zu erleichtern, haben wir Clients in C, C#, PHP und Java entwickelt, die von der Web-Seite des Services heruntergeladen werden können.

3.2 Der Konvertierungsansatz: Nutzung von XSLT auf Basis eines Schichtenmodells

Die Zielsetzung des Webservice besteht darin, einen Dienst zur Konvertierung zwischen verschiedenen in XML beschriebenen Geschäftsdokumenten anzubieten. Grundlage ist ein Schichtenmodell, wie es in der folgenden Abbildung dargestellt ist. Dabei haben sich die Akteure auf unterer Ebene auf Kommunikationsstandards wie HTTP und SOAP sowie darauf aufbauend auf XML zur Beschreibung der Dokumente geeinigt. Auf höherer Ebene können sie jedoch die Formate nutzen, die ihren individuellen Anforderungen am besten entsprechen. Die Konvertierung erfolgt mithilfe von XSLT-Stylesheets [Tidw01, EdSt01].

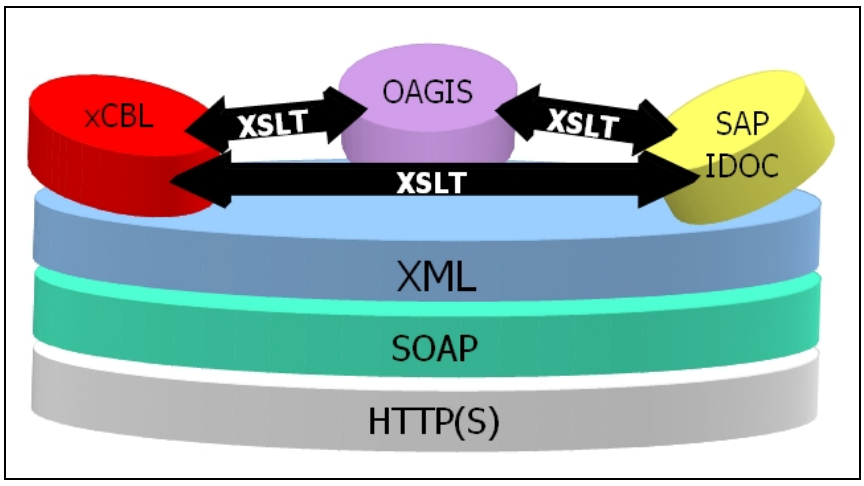

Abbildung 2: XML und XSLT als Hauptkomponenten einer beispielhaften Implementierung des Schichtenmodells

Mit dem Schichtenmodell versuchen wir, die Vorteile der beiden Alternativen „Standardisierung“ und „Konvertierung“ miteinander zu verbinden. Dabei gehen wir von der These aus, dass die Koordinations- und Opportunitätskosten einer Nutzung der Standards für HTTP(S), SOAP und XML relativ gering sind. Dafür sprechen die folgenden Gründe:

- HTTP (Hypertext Transfer Protocol) regelt die Kommunikation zwischen einem Client und einem Server und ist unabhängig von Betriebssystemen und Netzwerktypen. Als offener Standard ist HTTP weltweit anerkannt

[1] Die ebXML-Spezifikation ist von UN/CEFACT und OASIS entwickelt worden und definiert eine Kommunikationsarchitektur für den Austausch von in XML beschriebenen Geschäftsdokumenten. Als Framework für die XML-basierte Kommunikation zwischen Unternehmen bietet das ebXML-Framework über SOAP hinausgehende Funktionalitäten, die speziell auf den Austausch von Nachrichten zwischen Unternehmen ausgerichtet sind [Beim+02, S. 279f].

und in der letzten Version 1.1 seit mehreren Jahren unverändert im Einsatz. Die sichere Variante HTTPS greift zur Realisierung einer verschlüsselten Verbindung auf Secure Socket Layer (SSL) zurück.

- Die Einigung auf SOAP als Kommunikationsprotokoll erfordert vergleichsweise geringen Koordinationsaufwand. So gibt es inzwischen SOAP-Implementierungen für über 50 verschiedene Plattformen und Programmiersprachen (http://www.soapware.org/directory/4/implementations). Weiterhin lassen sich SOAP-Nachrichten über HTTP verschicken, wodurch unternehmensinterne Sicherheitsmechanismen, wie z. B. Firewalls, die Kommunikation nicht behindern [KrLu02, S. 53f].

- Die Einigung auf XML als Sprache zur Beschreibung der Geschäftsdokumente erscheint ebenfalls relativ unproblematisch, da es sich bei XML um eine allgemeine Sprache handelt, mit der sich die von den Akteuren jeweils präferierten Formate abbilden lassen. Zudem wird XML inzwischen von den meisten ERP- und Datenbank-Anbietern unterstützt.

Während unser Schichtenmodell von der gemeinsamen Nutzung dieser Standards auf der unteren Ebene ausgeht, ist auf der oberen Ebene Vielfalt, d. h. die Nutzung der für die Akteure am besten geeigneten Formate zugelassen. So kann ein Akteur etwa, wie in Abbildung 2 dargestellt, xCBL, ein anderer OAGIS und ein Dritter beispielsweise SAP IDOC verwenden. Dabei bilden XML und verwandte Standards im Vergleich zu herkömmlichen EDI-Technologien eine gute Grundlage für die Konvertierung von Dokumenten. Für die Existenz eines Potenzials zur Reduzierung der damit verbundenen Konvertierungskosten sprechen die folgenden Gründe:

- Programmiersprachen wie Java und C++ arbeiten sehr gut mit XML und XSLT zusammen. Ein Beispiel hierfür ist die Integration der javax.xml.* Klassen zum Verarbeiten von XML-Dokumenten in das Java Development Kit (JDK) ab Version 1.4.0.

- Die Mehrzahl der im Unternehmensumfeld eingesetzten Datenbanken und ERP-Systeme bietet mittlerweile die Unterstützung von XML. Dies ermöglicht die Integration von XML-Daten in Inhouse-Systeme.

- XML-Dokumente sind für einen Menschen im Vergleich zu herkömmlichen EDI-Dokumenten einfacher zu verstehen und zu bearbeiten. Dies erleichtert das Extrahieren von Informationen aus den Dokumenten und damit auch die Entwicklung von Konvertern.

- Die Struktur von XML-Dokumenten lässt sich in DTD´s und XML Schemata exakt definieren [HoJu02, S. 34].

- Zum Thema XML existiert bereits eine große Entwicklergemeinschaft, die einen Großteil der entwickelten Software mit offenem Quellcode kostenlos verfügbar macht. Weiterhin steht für die XML-Standardfamilie auch eine Vielzahl an komplementären Standards, Technologien und Produkten zur Verfügung, die den Umgang mit XML-Dokumenten wesentlich vereinfachen.

Im Folgenden wollen wir nun untersuchen, wie die Konvertierung mit XSLT auf Basis des Schichtenmodells durchgeführt werden kann. Dabei sind für diese Konvertierung verschiedene Ansätze [Wüst+02] denkbar. Eine Möglichkeit besteht darin, jeweils zwei Formate durch zwei entsprechende Stylesheets ineinander zu konvertieren. Bei n Formaten werden dann maximal n(n-1) Stylesheets benötigt. Aus diesem Grund erscheint unter bestimmten Voraussetzungen die Nutzung eines intermediären Formats – im Folgenden auch Super-Standard genannt – als sinnvoll, wie in Abbildung 3 zu sehen ist:

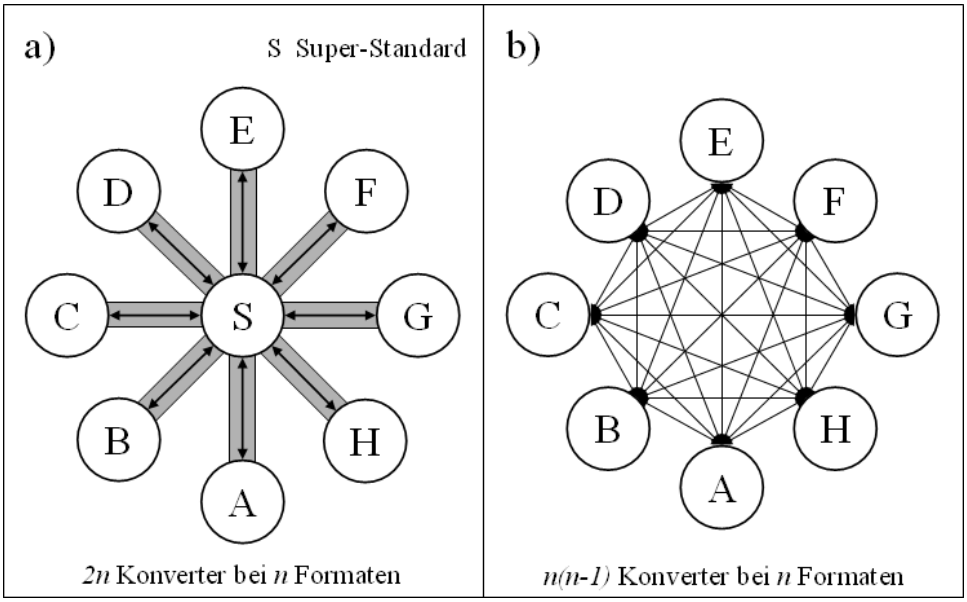

Abbildung 3: Konvertierung mittels eines intermediären Formats (a) und mittels eines vollständig vermaschten Netzes (b)[2]

Bei Nutzung eines intermediären Formats[3] kann die Anzahl der benötigten Stylesheets auf 2n begrenzt werden. Dabei ist zu beachten, dass die obigen Ausführungen jeweils immer für einen Dokumenttyp gelten. Da mehrere Dokumenttypen (PurchaseOrder, Invoice etc.) unterstützt werden, muss dies bei der Organisation des zum Webservice gehörenden Repositories bedacht werden. Dementsprechend gibt es mehrere intermediäre Formate, mindestens einen pro Dokumenttyp. Dabei arbeiten wir zurzeit daran, aus einer Menge von n Formaten annähernd automatisch einen solchen Super-Standard zu erzeugen. Erste Ansätze, an denen wir uns bei der Gestaltung der Super-Standards orientieren, sind:

- Die Semantik sollte möglichst weitgehend im Tagnamen abgebildet werden, um eine zu tiefe Struktur und damit verbundene Konvertierungsprobleme, die aus unterschiedlichen Gruppierungstiefen der Elemente resultieren, zu vermeiden. In Abbildung 4 sind zwei verschiedene Arten zu sehen, den gleichen Inhalt darzustellen. In 4a sind zwei Ebenen notwendig, während in 4b nur eine Ebene benötigt wird, da die Semantik im Tagnamen enthalten ist. Um eine flache Struktur zu gewährleisten, gehen wir als grobe Regel davon aus, dass der Super-Standard nicht mehr als vier Ebenen tief sein sollte.

(a)
```
<supplier>
  <phone>0123456789</phone>
  <email>hans@mustermann.de</email>
</supplier>
```

(b)
```
<supplier_phone>0123456789</supplier_phone>
<supplier_email>hans@mustermann.de</supplier_email>
```

Abbildung 4: Semantik in der Struktur (a) bzw. im Tagnamen (b)

- Die Benutzung von Attributen sollte nach Möglichkeit vermieden werden, um den XSLT-Code einfach zu halten. Sämtliche Anwendungen von Attributen können auch durch Unterelemente realisiert werden.
- Daten sollten innerhalb des Super-Standards möglichst atomar und ohne Redundanzen gespeichert werden. Ausführliche Bezeichnungen wie „United States Dollar“ sind Abkürzungen wie „$“, „US$“, „USD“ oder „US Dollar“ vorzuziehen. Ebenso sollten bei einem Datum Jahr, Monat, Tag, Zeitzone, Stunde, Minute und Sekunde in einzelnen Tags gespeichert werden und nicht in einem Zeichenkette wie z.B. „yymmddThhmmss“. Dies erspart bei der Konvertierung aufwändige Trenn- und Verkettungsoperationen.

---

2 Das Stylesheet-Repository des <x:act> Webservices unterstützt sowohl die Nutzung eines intermediären Formats als auch vollständig vermaschte Netze (entspricht Direktkonvertierungen).

3 Dieses Szenario kann grundsätzlich auch durch einen Marktplatz abgebildet werden, der die beteiligten Formate direkt umwandelt [EsZu02, S. 257].

- Die Elemente des Super-Standards sollten keine Restriktionen in Bezug auf den Inhalt der Elemente enthalten. Anderenfalls wäre die Flexibilität des Super-Standards unnötig eingeschränkt.
- Weiterhin sollten die Elemente des Super-Standards genau dokumentiert sein und jede denkbare Möglichkeit der Ausgestaltung von Dokumentinhalten berücksichtigen. Damit soll sichergestellt werden, dass neu hinzukommende Formate möglichst schnell integriert werden können.

Ein möglicher alternativer Ansatz zur völligen Neuentwicklung des Super-Standards besteht darin, ein umfassendes Format wie xCBL 4.0 als Grundlage zu nehmen.

Bei der Konvertierung zwischen verschiedenen XML-Formaten kann nun eine Vielzahl von Problemstellungen auftreten. Dazu gehören unter anderem [Wüst+02]:

- Unterschiedliche Tagnamen, also beispielsweise das Speichern einer Postleitzahl als „Postcode“ in Format A und als „PostalCode“ in Format B.
- Unterschiedliche Strukturtiefen, wie in Abbildung 4 beispielhaft zu sehen.
- Gleicher Elementinhalt, unterschiedlicher semantischer Gesamtgehalt. Beispielsweise legt Format A fest, dass „01714567890“ eine Mobiltelefonnummer ist, Format B ermöglicht jedoch nur die Angabe von allgemeinen Telefonnummern. Beim Konvertieren von A nach B würde somit die Information, dass es sich um eine Mobiltelefonnummer handelt, verloren gehen.
- Redundante Daten, die eventuell in einem bestimmten Format enthalten sind, müssen beim Konvertieren in dieses Format „künstlich“ erzeugt werden. Wenn beispielsweise Format A das redundante Element „Gesamtbestellsumme“ enthält, so muss beim Konvertieren in A dieses Element durch Aufsummieren der einzelnen Bestellpositionen generiert werden.
- Unterschiedliche Formatierung bei gleichem Elementinhalt. Die äußert sich darin, dass in den zu konvertierenden Elementen die gleichen Inhalte gespeichert sind, diese sich aber in der Formatierung unterscheiden.

In der folgenden Abbildung wollen wir die Funktionsweise von XSLT-Stylesheets anhand des letztgenannten Konvertierungsproblems erläutern. Dabei ist ein amerikanisches Datum (a) in ein deutsches Datum (b) zu konvertieren. Ein komplexeres Beispiel, das die Konvertierung zwischen einer xCBL- und OAGIS-Bestellung zeigt, findet sich unter http://www.x-act.org/demo_po.php.

(a)
```
<?xml version="1.0" encoding="UTF-8" ?>
<root>
  <date>2002/11/13</date>
</root>
```

(b)
```
<?xml version="1.0" encoding="UTF-8" ?>
<root_element>
  <date>13.11.2002</date>
</root_element>
```

XSLT-Stylesheet (a) ==> (b)

```
<?xml version="1.0" encoding="UTF-8" ?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="xml" version="1.0" encoding="UTF-8" indent="yes" />
  <xsl:template match="/">
    <xsl:variable name="Date" select="//date" />
    <root_element>
      <xsl:element name="date">
        <xsl:value-of select="concat(substring($Date,9,2),'.',substring
          ($Date,6,2),'.',substring($Date,1,4))" />
      </xsl:element>
    </root_element>
  </xsl:template>
</xsl:stylesheet>
```

Abbildung 5: Beispiel für die Konvertierung zwischen zwei verschiedenen Datumsformaten

Die Erstellung eines Stylesheets kann mit einem hohen Aufwand verbunden sein. Vor diesem Hintergrund haben wir den <x:act> Mapper entwickelt, mit dem die Entwicklung dieser Stylesheets teilautomatisiert werden kann.

Der <x:act> Mapper nutzt das Konzept eines intermediären Formats und erlaubt eine grafische Generierung von Stylesheets für die Konvertierung von XML-Geschäftsdokumenten zu vorgegebenen Super-Standards. Dazu werden jeweils ein komplettes XML-Dokument und ein Super-Standard eingelesen und konvertierbare Elemente miteinander mittels Drag&Drop-Technik direkt oder mithilfe gegebener Operatoren grafisch verknüpft. Auf diese Weise decken wir einen Großteil von Besonderheiten und Eigenarten bei der XML-Konvertierung mittels XSLT-Stylesheets ab. In komplexen Fällen ist jedoch eine manuelle Nachbearbeitung des generierten Stylesheets notwendig. Abbildung 6 zeigt einen Screenshot des Mappers. Im linken Bereich der Oberfläche ist eine Instanz des zu konvertierenden Quellformats zu sehen, also z.B. eins der Formate A-H aus Abbildung 3a. Im rechten Bereich ist der Super-Standard S des betreffenden Dokumenttyps – z.B. Purchase Order – dargestellt. Im mittleren Teil befindet sich der Mapping-Bereich, in dem korrespondierende Elemente grafisch miteinander verbunden werden.

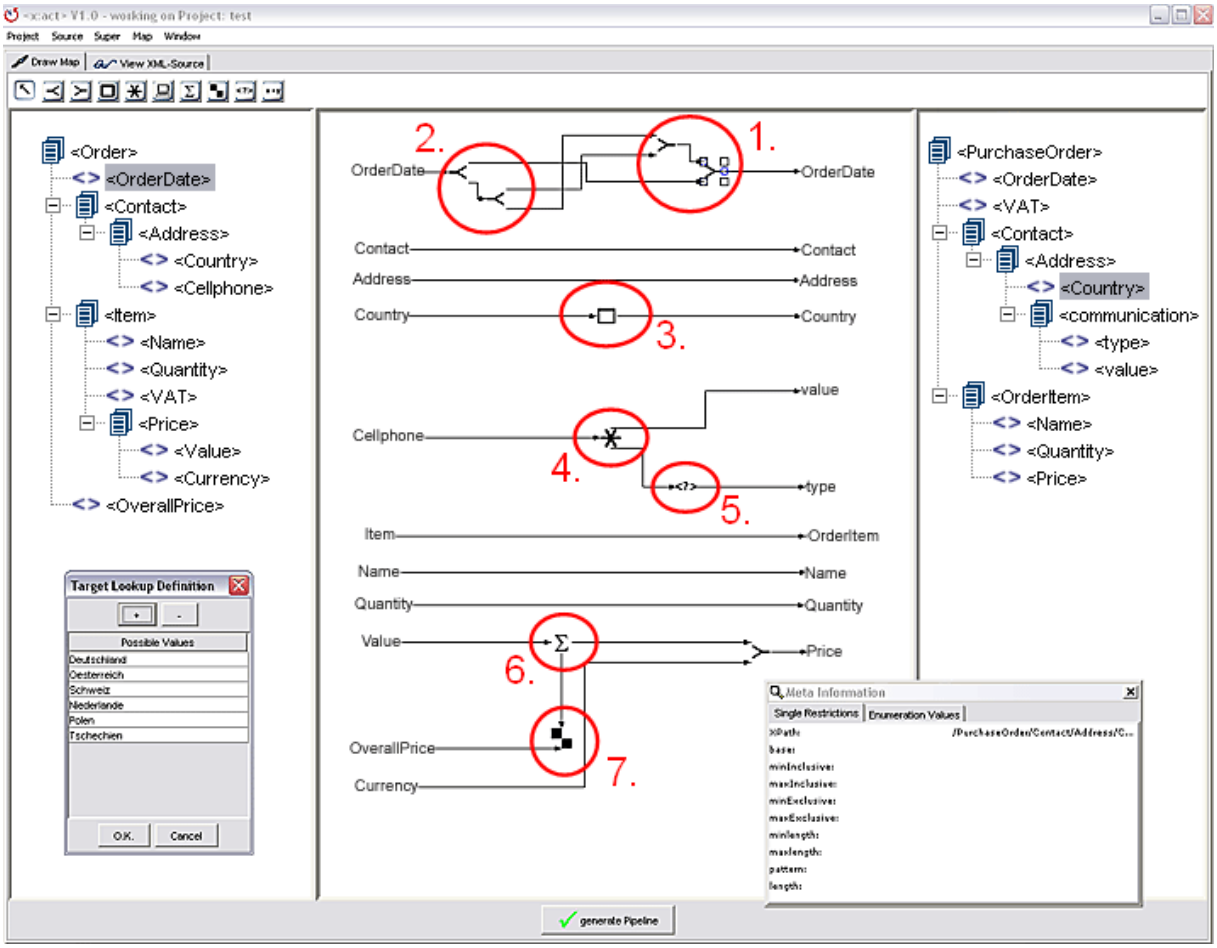

Abbildung 6: Screenshot des <x:act> Mappers

Folgende Operatoren stehen zurzeit zur Verfügung:

1. Merge: Damit lassen sich zwei Zeichenketten mit einem optionalen Verkettungszeichen verbinden.
2. Split: Dieser Operator trennt eine Zeichenkette anhand eines definierbaren Trennzeichens in zwei Teile. Dieser Operator ist also das Gegenstück zu „Merge“.

In Abbildung 6 werden die Operatoren „Merge“ und „Split“ benutzt, um eine Datumstransformation, wie in Abbildung 5 beschrieben, umzusetzen. Dabei soll „2002/11/13“ in „13.11.2002“ umgewandelt werden. Dazu wird „2002/11/13“ zunächst mittels des ersten Split-Operators in „2002“ und „11/13“ getrennt. Der zweite Split-Operator zerlegt danach „11/13“ in „11“ und „13“. Nun werden die Einzelteile wieder zu einem deutschen Datum zusammengesetzt. Dazu werden zunächst mit dem ersten Merge-Operator „13“ und „11“ verbunden. Das so entstandene „13.11“ wird im letzten Schritt mithilfe des zweiten Merge-Operators mit „2002“ zum Endergebnis „13.11.2002“ verbunden. Aus diesem grafischen Mapping wird dann automatisch der entsprechende, in Abbildung 5 zu sehende, XSLT-Code erzeugt.

3. Lookup Table: Anhand dieser Nachschlagetabellen kann man beispielsweise Abkürzungen und Äquivalenzwerte konvertieren (z. B. „de“ ist äquivalent zu „Deutschland“ und „at“ ist äquivalent zu „Österreich“).
4. Multiply: Dieser Operator erlaubt den Mehrfachzugriff auf Knoten des Quelldokuments (entspricht dem XSLT-Element „variable“).
5. Extract Tag: Mithilfe dieses Operators kann die XSLT-Funktion „name“ erzeugt werden. Dies ist notwendig, wenn Elementnamen in Elementwerte überführt werden sollen.

In Abbildung 6 werden die Operatoren „Multiply“ und „Extract Tag“ kombiniert verwendet, um

```
<cellphone>0177123456</cellphone>
```

in

```
<communication>
      <type>cellphone</type>
      <value>0177123456</value>
</communication>
```

umzuwandeln. Mithilfe des Operators „Extract Tag“ wird der Tagname des Elements „cellphone“ in das Element „type“ überführt. Der „Multiply“-Operator sorgt dafür, dass das Element „cellphone“ ein zweites Mal benutzt werden kann, diesmal wird der Inhalt des Elements „cellphone“ – also „0177123456“ – in das Element „value“ überführt.

6. Sum: Anhand dieses Operators können die Werte mehrerer Elemente eines Dokuments summiert werden.
7. Create Redundancy: Mit diesem Operator können semantisch redundante Elemente erzeugt werden.

In Abbildung 6 werden die beiden letztgenannten Operatoren dazu verwendet, das im Quelldokument vorhandene, redundante Element „OverallPrice“ zu erzeugen. Dazu werden alle, im Super-Standard gefundenen „price“-Elemente aufsummiert und das Ergebnis in das Element „OverallPrice“ übertragen.

Nach der grafischen Zuweisung aller relevanten Elemente folgt auf Wunsch des Benutzers die automatische Generierung zweier Stylesheets. Diese erlauben nun Konvertierungen sowohl vom jeweiligen XML-Dokument zum vorgegebenen Super-Standard als auch in der Gegenrichtung.

3.3 Architektur und Funktionsweise von <x:act>

Im Folgenden wollen wir nun die technische Funktionsweise des Webservice <x:act> vorstellen. Im ersten Schritt erzeugt eine Applikation bzw. ein Anwender einen SOAP-Request, der benötigte Meta-Konvertierungsinformationen, wie etwa Dokumenttyp, Quell- und Zielformat spezifiziert. Als Anhang ist das zu konvertierende XML-Dokument enthalten. Der SOAP-Request muss dabei den Spezifikationen des WSDL-Dokuments entsprechen, das unter http://www.x-act.org/conversion.wsdl verfügbar ist. Die beim Webservice eingehenden SOAP-Anfragen werden vom Conversion-Servlet entgegengenommen. Falls die in der Anfrage enthaltenen Informationen unvollständig bzw. fehlerhaft sind, wird ein standardisierter SOAP-Fehler generiert und an den Absender zurückgeschickt. Bei korrekter Anfrage ermittelt das Servlet zunächst per Datenbankabfrage, ob die gewünschte Konvertierung durch ein oder mehrere im Repository vorhandene Stylesheets durchgeführt werden kann.

Wenn dies der Fall ist, wird die Konvertierung durchgeführt und die anfragende Anwendung bekommt das konvertierte XML-Dokument via SOAP-Response zugeschickt. Ist kein entsprechendes Stylesheet verfügbar, bekommt der Anwender den oben angesprochenen Mapper per Java Web Start (JWS) zur Verfügung gestellt. Auf dieser Basis kann er das entsprechende XSLT-Stylesheet erzeugen und dieses in das Repository hochladen. Nun kann das Servlet den Konvertierungsvorgang durchführen und dem Nutzer das konvertierte Dokument im Zielformat zuschicken. Die Zusammenhänge sind in der folgenden Abbildung dargestellt.

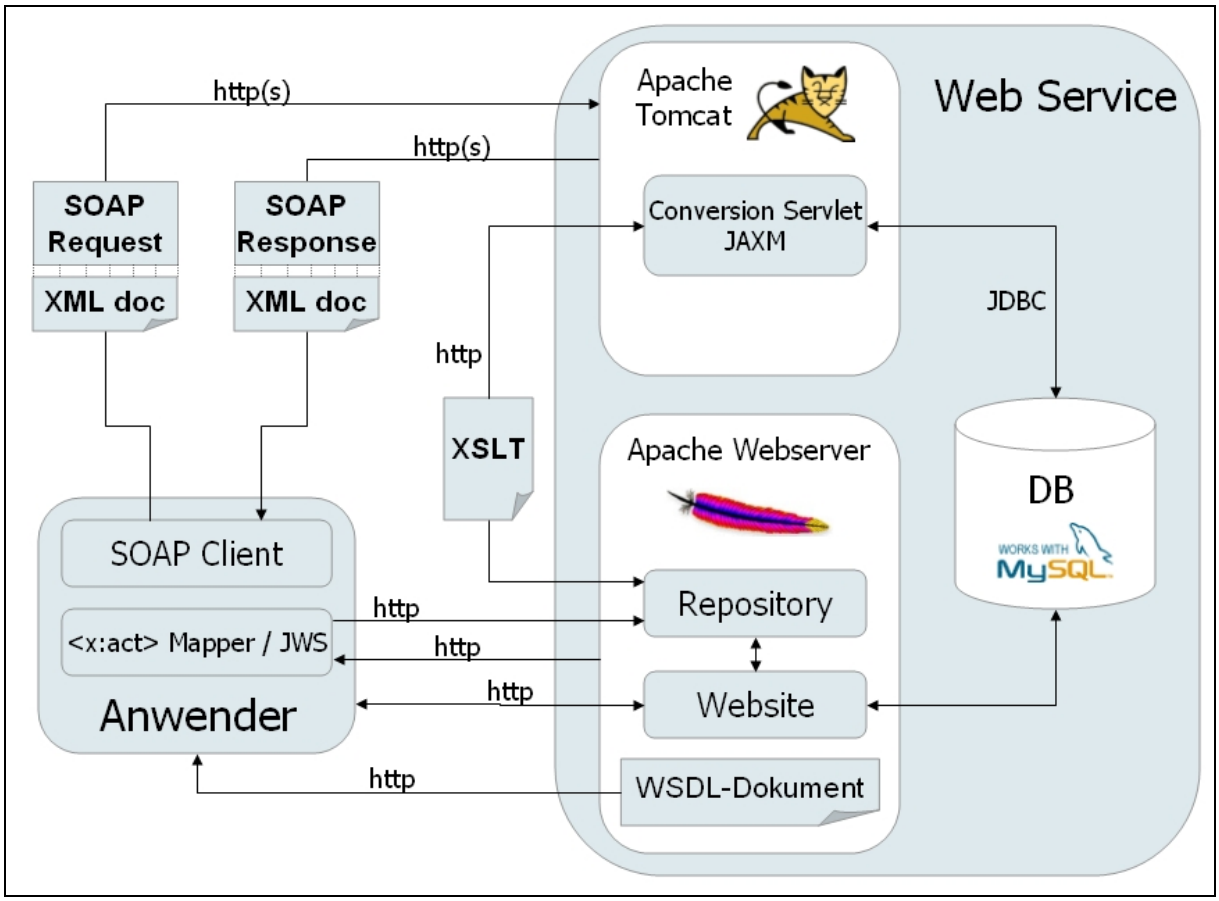

Abbildung 7: Die Architektur des <x:act> Webservice

Um Sicherheitsanforderungen [HoJu02, S. 35] Rechnung zu tragen, ermöglichen wir wahlweise einen gesicherten Datentransfer mittels HTTPS-Verbindung (HTTP über SSL). Darüber hinaus planen wir die Verschlüsselung der ausgetauschten XML-Dokumente mittels XML-Encryption. Denkbar ist auch die eindeutige Identifizierung von Webservices und deren Nutzern mittels digitaler Signaturen [JeZe02].

In Bezug auf den Webservice ist die Verfügbarkeit der XSLT-Stylesheets offensichtlich ein kritischer Erfolgsfaktor. Um zu einer entsprechenden installierten Basis zu kommen, führen wir daher zwei Maßnahmen durch: Zum einen sind wir dabei, selbst solche Stylesheets zu erzeugen und im Repository abzulegen. Zum anderen sollen die Anwender, die den Service benutzen und ein Stylesheet (z. B. mithilfe des Mappers) selbst erzeugen, dieses freiwillig (quasi als Gegenleistung für die kostenlose Nutzung) unserem Service zur Verfügung stellen.

4 Zusammenfassung und Ausblick

In diesem Beitrag haben wir gezeigt, dass sich auf Basis offener Standards und Open-Source-Komponenten XML/EDI-Lösungen entwickeln lassen. Eine Zielsetzung besteht darin, die Einstiegsbarrieren in EDI-Netze zu senken, da EDI-Konverter etwa durch offene und kostenlose Software ersetzt werden können. Bei der Anwendung von XML/EDI zeigt sich jedoch auch, dass das Problem der Konvertierung zwischen verschiedenen Formaten noch nicht gelöst ist. Vor diesem Hintergrund haben wir den Webservice <x:act> entwickelt, mit dem eine solche Konvertierung durchgeführt werden kann. Dabei kann der Umgang mit neuen Technologien auch zu Problemen führen. Dazu gehört die Unsicherheit in Bezug auf Konstanz und Stabilität der genutzten Standards. Einige Technologien sind noch nicht völlig ausgereift [Beut02, S. 29] und verschiedenen Veränderungen unterworfen. Besonders betroffen sind hierbei die mit Webservices verbundenen Technologien und XML-Businessvokabulare, wie xCBL und OAGIS. Dies erschwert die Software-Entwicklung, da permanent Anpassungen der verwendeten Software-Komponenten notwendig sind. Im Gegensatz dazu bereitet die Nutzung eines fest etablierten Standards, wie etwa HTTP, keinerlei Probleme.

Bislang ist die Anwendung von <x:act> auf die Konvertierung zwischen XML-Dokumenten beschränkt. Dieses Konvertierungsproblem ist jedoch nur eine Instanz von einer Vielzahl ähnlicher Problemstellungen. Beispiele hierfür sind:

- Konvertierung zwischen XML-Formaten und klassischen EDI-Standards.
- Konvertierung zwischen verschiedenen Datenbanksystemen.
- Konvertierung zwischen unterschiedlichen Werkzeugen zur Geschäftsprozessmodellierung.

In unseren zukünftigen Arbeiten wollen wir uns unter anderem auf die Erweiterung des Anwendungsgebiets konzentrieren

Da Konvertierungsaufgaben auf diesen verschiedenen Gebieten in der Regel ein hohes Detailwissen erfordern, wird eine Arbeitsgruppe meist kaum in der Lage sein, diese vielfältigen Konvertierungsprobleme zu beherrschen. Eine sinnvolle Strategie, um zu einem solchen allgemeinen Konvertierungsservice zu kommen, besteht nun darin, Kooperationen einzugehen, um entsprechendes Wissen zu bündeln. Die Kooperationspartner würden in diesem Fall jeweils ihr eigenes Know-how in den Service einbringen – ein Modell, das sich bereits im Rahmen von vielen Open-Source-Projekten bewährt hat.

Danksagung

Wir danken der Deutschen Forschungsgemeinschaft, die unsere Arbeiten mit der Förderung des Projekts „Standardisierung und Kooperationen in Logistiknetzwerken“ (SKILNET) unterstützt (BU 1098/1-1).

Literatur

[Alpa02] Alpar, P.: Die kritischen Erfolgsfaktoren für EDI-Dienstleistungsanbieter – Eine Delphi-Studie. In: Wirtschaftsinformatik 44 (2002) 1, S. 29 – 40.

[Beim+02] Beimborn, D.; Mintert, S.; Weitzel, T.: WI-Schlagwort: Web Services und ebXML. In: Wirtschaftsinformatik 44 (2002) 3, S. 277 – 280.

[Bett01] Bettag, U.: Aktuelles Schlagwort: Web-Services. In: Informatik-Spektrum 24/5, 2001, S. 302 – 304.

[Beut02] Beutenmüller, U.: Web Services: Top oder Flop? In: Information Management & Consulting 17 (2002) 3, S. 26 – 30.

[BeWe93] Bernt, P.; Weiss, M.: International Telecommunications, Sams Publishing: Carmel, 1993.

[BrWh85] Braunstein, Y.M.; White, L.J.: Setting technical compatibility standards: an economic analysis, in: The Antitrust Bulletin Summer 1985, S. 337 – 355.

[BuKö98] Buxmann, P.; König, W.: Das Standardisierungsproblem: Zur ökonomischen Auswahl von Standards in Informationssystemen. In: Wirtschaftsinformatik 40 (1998) 2, S. 122 – 129.

[EdSt01] Eder, J.; Strametz, W.: Composition of XML-Transformations. In: Bauknecht, K. et al. (Hrsg.): EC-Web 2001, LNCS 2115, S. 71 – 80.

[Emme93] Emmelhainz, M.: EDI – A Total Management Guide. Van Nostrand Reinhold: New York, 2nd Edition, 1993.

[EsZu02] Esswein, W.; Zumpe, S.: Realisierung des Datenaustauschs im elektronischen Handel. In: Informatik-Spektrum 25/5. 2002. S. 251 – 261.

[HoJu02] Hoidn, H.-P.; Jungclaus, R.: Web Services aus Sicht der Unternehmens-Architektur. In: Information Management & Consulting 17 (2002) 3, S. 31 – 35.

[JeZe02] Jeckle, M.; Zengler, B.: Sicherheitsaspekte beim Einsatz von XML-basierten Web-Diensten am Beispiel SOAP. In: Information Management & Consulting 17 (2002) 3, S. 37 – 46.

[KrLu02] Krammer, A.; Luft, O.: Web Services – wie kleine und mittlere Unternehmen profitieren können. In: Information Management & Consulting 17 (2002) 3, S. 51 – 56.

[Lang03] Langner, T.: Webservices mit Java – Neuentwicklung und Refactoring in der Praxis. Markt+Technik: München, 2003.

[LöPi02] Löwer, U.; Picot, A.: Web Services – Technologie-Hype oder Strategie-Faktor? In: Information Management & Consulting 17 (2002) 3, S. 20 – 25.

[Minz+02] Minz, R.; Datel, A.; Wenzky, H.: Web Services – nur eine Schimäre? In: Information Management & Consulting 17 (2002) 3, S. 6 – 12.

[MuKe02] Mukhopadhyay, T.; Kekre, S.: Strategic and Operational Benefits of Electronic Integration in B2B Procurement Processes. In: Management Science 48 (2002) 10, S. 1301 – 1313.

[Neub94] Neuburger, R.: Electronic Data Interchange: Einsatzmöglichkeiten und ökonomische Auswirkungen. Dt. Univ.-Verl.: Wiesbaden, 1994.

[Over02] Overhage, S.: On Specifying Webservices Using UDDI Improvements. In: Dittmar, T.; Franczyk, B.; Hofmann, R.; Langhammer, F.; Lenz, B.; Müller-Schloer, C.; Nicklas, M.; Philippow, I.; Unland, R.; Weber, M.; Weissenbach, H. G., Westerhausen, J. (Hrsg.): Proceedings 2002 Net.Objectdays Workshops. Erfurt 2002, S. 535 - 550.

[PiNe01] Picot, A.; Neuburger, R.: Grundsätze und Leitlinien der Internet-Ökonomie. In: Eggers, B.; Hoppen, G. (Hrsg.): Strategisches E-Commerce-Management. Gabler: Wiesbaden, 2001.

[Röwe02] Röwekamp, L.: Java Webservice Tutorial – Teil 1: Grundlagen. In: iX 9/2002, 2002, S. 121 – 127.

[Stef00] Steffen, T.: Internet-Quellen zu XML/EDI. In: Wirtschaftsinformatik 42 (2000) 1, S. 78 – 86.

[Tidw01] Tidwell, D.: XSLT. O'Reilly: Sebastopol, 2001.

[Weit+01] Weitzel, T.; Harder, T.; Buxmann, P.: Electronic Business und EDI mit XML. dpunkt: Heidelberg, 2001.

[Wüst+02] Wüstner, E.; Hotzel, T.; Buxmann, P.: Converting Business Documents: A Classification of Problems and Solutions using XML/XSLT. In: Proceedings of the 4th International Workshop on Advanced Issues of E-Commerce and Web-based Information Systems (WECWIS 2002), Newport Beach.

[W3C02a] World Wide Web Consortium: SOAP Version 1.2. http://www.w3.org/TR/soap12-part0/ sowie http://www.w3.org/TR/soap12-part0/ und http://www.w3.org/TR/soap12-part1/, 2002, Abruf am 2003-01-13.

[W3C02b] World Wide Web Consortium: Webservices Description Language (WSDL) Version 1.2. http://www.w3.org/TR/wsdl12/, 2002, Abruf am 2003-01-13.