Fachhochschule Giessen Friedberg
FB Mathematik Naturwissenschaften Informatik (MNI)

XML

Das ´Esperanto´ des E-Commerce?

Seminararbeit
- Großes Seminar -

Roger Zacharias
Matr.Nr.: 620354
7. Semester

10.06.2001
SS 2001

Seminarleiter: **Prof. Dr. Peter Kneisel**

XML

Das ´Esperanto´ des E-Commerce?

Seminararbeit
- Großes Seminar -

Roger Zacharias

10.06.2001

Vorwort

Technology of a sufficiently advanced level
will be indistinguishable from magic. (Arthur C. Clarke)

Blättert man zur Zeit in einer der vielen Fachzeitschriften für Softwareentwickler oder IT-Projektmanager, findet man sehr viele Artikel mit Überschriften wie “XML – die Lösung für ihr E-Business“, “XML – die Lösung aller Interoperabilitätsprobleme“, “XML auf dem Server“ oder “Endlich strukturierte Daten mit XML“. Das Schlagwort XML ist neben den Middleware-Technologien (J2EE, .NET, CORBA, DCOM, usw.) das momentan am häufigsten verwendete. Das übergeordnete Schlagwort ist E-Business: Der Kauf und Verkauf von Gütern inklusive Bezahlung mittels dem als Plattform dienenden Internet, der Informationsaustausch zwischen den Mitarbeitern einer Firma über das Kommunikationsmedium Intranet, sowie das Extranet zur Vernetzung eines Unternehmens mit seinen Lieferanten, Händlern und sonstigen Geschäftspartnern. Zur Zeit vollzieht sich der vom Informationszeitalter vorangetriebene Wandel vom ’normalen’ Geschäft zum E-Business. Dies verlangt jedoch eine auf Standards basierende, evolutionsfähige Softwarearchitektur. XML soll und kann diese Forderungen erfüllen.

Diese Arbeit soll einen Einblick in die Technologie, ihre Entstehungsgeschichte und in einige Anwendungen und Möglichkeiten im Bereich B2B geben.

DANKSAGUNGEN

Ich bedanke mich bei meiner Verlobten Alexandra Georg für Ihr liebevolles Verständnis, auch wenn es manchmal ’etwas später wurde’ und für ihre Hilfe beim Redigieren dieser Arbeit. Mein Dank geht auch an meine Eltern und Schwiegereltern für ihre Unterstützung. Weiterhin bedanke ich mich bei meinen Kommilitonen, Kollegen und Professoren, die mir durch ihr großes Interesse an diesem Thema, durch Fragen und Diskussionen geholfen haben, dieses Gebiet in seiner Gesamtheit zu begreifen und aus verschiedenen Blickpunkten zu betrachten.

Juni 2001 Roger Zacharias

Erklärung zur Urheberschaft

Hiermit versichere ich, die vorliegende Arbeit selbständig und unter ausschließlicher Verwendung der angegebenen Literatur und Hilfsmittel erstellt zu haben. Gleichzeitig versichere ich, diese Arbeit in gleicher oder ähnlicher Form weder veröffentlicht, noch einer anderen Prüfungsbehörde vorgelegt zu haben.

Dillenburg, den 10.06.2001

Inhaltsverzeichnis

1 Einführung

Das Internet als kostengünstige und weltweite Kommunikationsinfrastruktur entwickelt sich mehr und mehr zur Plattform des B2B. Außerdem steigt die Anzahl der enthaltenen Informationen ins Unermessliche. XML verfolgt hier drei grundlegende Ansätze:

- die Strukturierung der zur Zeit noch unstrukturierten Informationen im WWW
- den plattform- und systemübergreifenden Datenaustausch zwischen Geschäftspartnern und innerhalb von Firmen
- die Beschreibung von Metadaten

Durch konsequenten Einsatz der neuen ´Internet-Technologien´ wie J2EE, .NET und eben XML erhoffen sich Unternehmen steigende Gewinne und Kosteneinsparungen. So konnte die Frankfurter Flughafen AG durch die Umstellung ihres Bestellwesens auf ´Internet-Technologien´ eine Kosteneinsparung von 87% erzielen. Der ERP-Branchenführer SAP setzt in seinem R/3-Nachfolger ´mySAP.com´ auch auf neue Internet-Technologien. Die Wirtschaft hat aus diesen Gründen enormes Interesse an XML. In den meisten Software Firmen laufen Pilotprojekte in dieser Richtung, es werden verzweifelt Experten dieser jungen Technologie gesucht und selbst die Erzrivalen Sun und Microsoft arbeiten hier zusammen.

IT-Manager, Softwareentwickler und Softwareberater werden, unabhängig von Technologie und Sprache, die sie bislang für ihren Webauftritt und ihre Geschäftsabwicklung einsetzen, bald genauso wenig um XML herum kommen, wie heutzutage um HTML. Es ist also an der Zeit, sich näher mit XML auseinander zu setzen!

In dieser Arbeit soll nur ein grober Überblick und eine Einführung in die Technologien XML und XSL und die auf ihnen basierenden B2B-Technologien gegeben werden. Sie soll nicht als Vorlage zum Studium dienen, da dies den Rahmen dieser Arbeit sprengen würde. Hier möchte ich auf die zum Teil sehr guten Spezifikationen des W3C verweisen.

Kapitelübersicht:

Kapitel 2 – Was ist XML? – beschreibt die Entstehungsgeschichte, gibt einen Einblick in die Struktur und zeigt die Vorteile dieser Technologie gegenüber anderen auf.

Kapitel 3 – Was ist XSL? – gibt einen Einblick in die Sprache XSL, welche in Kombination mit XML zur geräteunabhängigen Präsentation verwendet wird.

Kapitel 4 – XML in der Praxis – zeigt Möglichkeiten, Entwicklungstrends und konkrete Anwendungen von XML im Bereich B2B auf.

Kapitel 5 – Zusammenfassung und Ausblick – fasst die wichtigsten Dinge dieses komplexen Themas noch einmal zusammen und beschreibt die enormen Entwicklungsmöglichkeiten dieser Technologie.

2 Was ist XML?

Dieses Kapitel soll die Grundlagen und Konzepte der Metasprache XML vorstellen. Hierzu wird zuerst auf die zugrundeliegende Plattform, das Internet und hier insbesondere das WWW, eingegangen, dessen rasante Entwicklung den Nährboden für eine Technologie wie XML darstellt. Im Zuge dieser Vorstellung soll das ´World Wide Web Consortium´ (W3C) als Begründer und ´Wächter´ dieser Technologie vorgestellt werden. Anschließend wird XML in das Umfeld der Meta- und Markup-Sprachen positioniert, dessen Vorteile dargestellt und diskutiert, warum XML im Begriff ist, die vorherrschende Sprache für die Datenbeschreibung, Datenspeicherung und den Datentransport zu werden.

2.1 Das Internet

Um die Struktur des Internets zu verstehen, ist es notwendig, dessen Geschichte zu kennen. Die Ursprünge des Internets liegen in den sechziger Jahren, als das amerikanische Verteidigungsministerium auf der Suche nach einem technischen Kommunikationssystem war, das auch nach einem Atomschlag noch funktionieren würde. Auch nach Ausfall einzelner Kommunikationspunkte, d.h. Städte oder Militärstützpunkte, sollte dieses Kommunikationssystem funktionsfähig bleiben. Aus diesem Anforderungsprofil begründet sich die auf den ersten Blick chaotische Struktur des Internets: unabhängig voneinander operierende Knotenpunkte und Datenpakete statt permanenten Datenströmen. 1992 wurde am Kernforschungszentrum CERN[1] in Genf das ´World Wide Web´ (WWW) auf Basis des Internets entwickelt. Hiermit wurde ein Standard geschaffen um Informationen aufzubereiten, Dokumente grafisch zu präsentieren und diese durch Hyperlinks miteinander zu verknüpfen. Die Basis dieser Entwicklung waren das Internetprotokoll HTTP und die Markup-Sprache HTML. Zunächst nur zur Verbindung von Universitäten und Forschungseinrichtungen genutzt, verbreitete sich das WWW, insbesondere seit 1994 mit der Einführung einfach zu bedienender Browser, rasant. Mittlerweile ist die Nutzerzahl auf über 300 Millionen Anwender gestiegen und der tägliche Zuwachs beträgt ca. 70.000.

[1] Conseil Européen pour la Recherche Nucléaire, Europäische Organisation für Kernforschung, 1954 gegründet, mittlerweile mit 20 Mitgliedstaaten das weltgrößte Partikelphysik-Zentrum der Welt.

Die Anzahl der Dokumente hat mittlerweile die 1 Milliarden Marke überschritten[1], da das Internet nun auch eine Fülle von Informationen für den ´normalen Endanwender´ bietet: Telefonverzeichnisse, Wörterbücher, Fahrpläne, Warenkataloge, Stadtpläne, Nachrichten, Aktienkurse, usw. Aufgrund dieser Zahlen erkannte auch die Wirtschaft recht schnell das riesige Marktpotential des Internets. Online-Shops sprießen aus dem Boden und der elektronische Datenaustausch zwischen Geschäftspartnern wird mehr und mehr über das Internet abgewickelt.

Mit dieser Entwicklung sind aber auch Nachteile verbunden:

- Die Menge an unstrukturierten Informationen ist so gigantisch, dass es sehr schwer ist, die relevanten Informationen zu finden. Zitat Simon Phipps[2]: ”The Web could be a much better business tool if its data was more structured!“.
- Ein weiteres Problem liegt in der unterschiedlichen Darstellung der Informationen. Es existiert kein absoluter Standard für die Speicherung von Daten und die Kommunikation. So durchläuft ein Datum aus der Datenbank viele unterschiedliche Umformatierungen und Konvertierungen um z.B. auf einem WAP-Handy angezeigt zu werden oder in einer PDF-Datei zur Verfügung zu stehen.
- Die Inhalte unterliegen einer hohen Änderungsrate, da die Aktualität ein wichtiges Kriterium der dargestellten Informationen ist. Diese Aktualität kann aufgrund der Informationsmenge nicht oder nur sehr schwer gewährleistet werden.
- Mit Hilfe der Darstellung und des Kontextes kann der Mensch die Semantik der Daten erkennen, eine Maschine (Software) kann jedoch keine semantischen Informationen erkennen und extrahieren.
- Das WWW war ursprünglich für die Mensch-Mensch- bzw. Mensch-Maschine-Kommunikation konzipiert. Durch die Verwendung dieses Mediums als B2B-Plattform wird die Maschine-Maschine-Kommunikation immer wichtiger, welches vollkommen andere Anforderungen an die verwendeten Kommunikationssprachen stellt.

2.2 Das World Wide Web Consortium (W3C)

Das W3C hat seinen Hauptsitz am Massachusetts Institute of Technologie (MIT) in Boston. Dieses unabhängige Konsortium sorgt für die Weiterentwicklung des Internets durch Schaffung neuer Grundlagentechniken wie z.B. XML, Erweiterung der bestehenden Technologien und Prüfungen von Standardisierungsanträgen (wie z.B. die Überleitung von EDI-Defintionen in XML-DTDs, siehe *Kapitel 4.3 E-Commerce mit BizTalk und ebXML*). Mit einer Reihe von Standards will das W3C den Anbietern von Websites die Möglichkeit bieten, ihre Inhalte genauer zu beschreiben und die präsentierten Informationen zu

[1] Zahlen nach www.W3C.org.

[2] Simon Phipps ist CTO (Chief Technology Officer) bei Sun Microsystems Inc.

strukturieren, um die maschinelle Verarbeitung der publizierten Daten zu erleichtern. Das Ziel ist ein ´semantisches Web´ (maschinell auswertbare Informationen über andere Informationen, d.h. Metadaten), in welchem auch Computer die Informationen aus dem Web lesen und auswerten können, was zur Zeit noch dem Menschen vorbehalten ist.

2.3 Meta-Sprachen, Markup-Sprachen und Dokumente

In vielen ´Fachzeitschriften´ wird XML direkt mit HTML verglichen oder gar als ´Nachfolger von HTML´ bezeichnet. Um eine klare Trennung zu erreichen, sollten zuerst die Begriffe der Meta- und Markup-Sprache sowie des Dokumentes geklärt werden.

2.3.1 Meta-Sprachen

Eine Meta-Sprache dient zur Definition der Grammatik und des Vokabulars anderer Sprachen, d.h. sie enthält Informationen über die in einer Sprache transportierten Informationen. So könnte in einer Metasprache, welche die deutsche Sprache beschreibt, zum Beispiel stehen:

- ´Guten Tag!´ und ´Auf Wiedersehen!´ begrenzen ein Gespräch
- Ein Gespräch besteht aus einer nichtleeren Anzahl von Sätzen
- Sätze bestehen aus einer nichtleeren Anzahl von Wörtern und Pausen
- Ein Wort besteht aus den Buchstaben A-Z
- usw.

Mit Meta-Sprachen können also Sprachen, wie zum Beispiel Markup-Sprachen, definiert werden. Bei XML und dessen Vorgänger SGML handelt es sich um solche Meta-Sprachen.

2.3.2 Markup-Sprachen

Bei Markup-Sprachen, auch als Auszeichnungssprachen bezeichnet, handelt es sich um Sprachen, die mittels eingestreuter Direktiven (Tags) die Struktur und/oder die Präsentation eines Dokumentes bestimmen. Markup-Sprachen (wie z.B. HTML, RTF) erzeugen Dokumente immer für eine bestimmte Anwendung, und werden meist für eine spezielle Anwendung (z.B. Textformatierung) entworfen. Bei HTML handelt es sich um eine solche Auszeichnungssprache. HTML selbst wurde durch die Meta-Sprache SGML definiert.

2.3.3 Dokumente

Ein Dokument wiederum ist mit Hilfe einer Markup-Sprache definiert und beinhaltet letztendlich die Daten, ist also der Informationsträger. Als Beispiel sind RTF-Dokumente, Microsoft Word Dokumente, PDF-Dokumente und natürlich mit HTML definierte WWW-Seiten zu nennen.

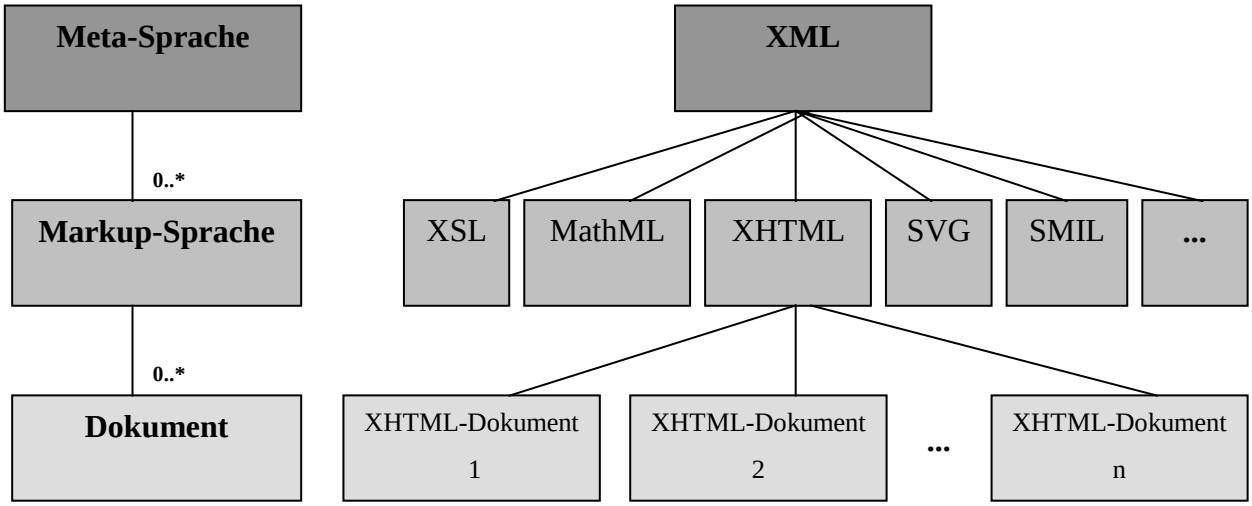

Abbildung 2-1 **Beziehung Meta-Sprache, Markup-Sprache, Dokument**

Abbildung 2-1 – Beziehung Meta-Sprache, Markup-Sprache, Dokument verdeutlicht den Zusammenhang zwischen diesen Instanzen. Auf der linken Seite ist die Beziehung allgemein dargestellt und auf der rechten konkret am Beispiel von XML. Den angeführten Vergleich von XML und HTML kann man also nur auf der Ebene der Dokumente durchführen. Dazu ist es wiederum notwendig den Aufbau eines Dokumentes zu kennen.

Ein Dokument besteht aus:

- Struktur (Markup-Elemente mit Attributen)
- Inhalt (die eigentlichen Informationen, d.h. Text- und Daten-Elemente)
- Präsentation (Definition der Ausgabe des Inhaltes)

Diese Verteilung soll an einem ´typischem´ HTML-Dokument dargestellt werden, welches aufgrund der Verbreitung von HTML sicher keiner Erläuterung bedarf:

```
<html>
        <head>
                <title> Typisches HTML-Dokument </title>
        </head>
        <body>
                <h1> Überschrift erster Ebene </h1>
                <h2> Überschrift zweiter Ebene </h2>
                <br>
                <br>
                Allgemeiner Text ohne besondere Formatierung
                <b> Fettgedruckter Text
        </body>
</html>
```

Abbildung 2-2 **Typisches HTML-Dokument**

Man sieht hier deutlich die Struktur des HTML-Dokumentes: Das gesamte Dokument ist durch die Direktiven <*html*> und </*html*> begrenzt, der Kopf des Dokumentes, welcher den Titel und gegebenenfalls Meta-Informationen beinhaltet, durch <*head*> und </*head*> und der Körper des Dokumentes, welcher den eigentlichen Inhalt des Dokumentes, also die Informationen, darstellt, ist durch das einführende <*body*> und das abschließende </*body*> begrenzt. Die Präsentation wird über Direktiven wie <*br*>, <*h1*>, <*h2*> oder <*b*> direkt gesteuert.

2.4 SGML, XML und HTML

Mit dem Wissen aus *Kapitel 2.3 – Meta-Sprachen, Markup-Sprachen und Dokumente* soll nun näher auf die Metasprachen SGML und XML und die Auszeichnungssprache HTML, sowie deren Entwicklung, eingegangen werden.

2.4.1 SGML

Der ´Vorgänger´ der Metasprache XML und die Basis aller bisherigen Auszeichnungssprachen des Webs liegt in der 1986 als ISO-Standard 8879 festgelegten Metasprache SGML (Standard Generalized Markup Language). Die Grundlage dieser Sprache bilden zum einen die 1969 bei IBM entwickelte Auszeichnungssprache GML (Generalized Markup Language) für Textformatierungen und zum anderen das 1968 von der Graphic Communications Association Organisation entwickelte Verfahren GenCode zur Definition generischer Formatierunscodes für Satzsysteme.

Der Gründer von SGML Charles Goldfarb, entwickelte diese Sprache mit dem Ziel, die logische Struktur von Textdokumenten abzubilden und die eigentlichen Informationen von deren Präsentation zu trennen. SGML fand und findet auch heute noch für das professionelle Präsentieren von Daten und Informationen in vielen großen Konzernen und Institutionen z.B. in Form von Bibliotheksverwaltungen, Redaktions- und Dokumentationssystemen Anwendung. Allerdings ist SGML sehr komplex, weshalb sich im WWW die von SGML abgeleitete sehr einfache Sprache HTML durchsetzte.[1]

2.4.2 HTML

Die Auszeichnungssprache HTML (Hypertext Markup Language) wurde 1989 von Tim Bernes-Lee mit Hilfe von SGML definiert, um Informationen im Bereich der Teilchenphysik mit Kollegen im Internet auszutauschen. Diese Sprache, die das 1950 von Ted Nelson entwickelte Hypertext-Konzept unterstützt und zusammen mit dem Transportprotokoll HTTP die Basis des sich rasant verbreitenden WWW bildet, hat sich heute, in der Version 4.0, zum absoluten Standard zur Beschreibung von Web-Seiten etabliert. HTML, welches vom W3C weiterentwickelt wird, zeichnet sich durch die Möglichkeit aus, Texte und andere Elemente wie Grafiken sehr einfach und unkompliziert auf einer Seite zu formatieren und diese Seite durch Hyperlinks mit eigenen oder fremden Seiten zu verknüpfen.

---

[1] Die Markup-Sprache HTML ist eine SGML-DTD.

Die im WWW auf HTML Basis angebotenen Informationen sind zwar umfangreich, informativ und optisch gut strukturiert, allerdings ist eine gezielte Suche nach bestimmten Informationen kaum möglich. Es liegt im Charakter von HTML begründet, die Informationen optisch aber nicht semantisch zu strukturieren und so die Inhalte mit der Präsentation verschmelzen zu lassen. Dabei existieren keinerlei Vorschriften für die Semantik eines HTML-Dokumentes, lediglich die Darstellung ist über Standardisierung der Auszeichnungssyntax durch das W3C festgelegt. Das bedeutet, die Tags beschreiben nur wie die Daten dargestellt werden sollen, aber nicht, was sie darstellen. So ist es nicht möglich, die dargestellten Daten aus einem HTML-Dokument zu extrahieren, um sie extern zu bearbeiten oder an anderer Stelle wiederverwenden zu können. Es kommt also zu einem Informationsverlust im Sinne der Weiterverarbeitung der Daten.

2.4.3 XML

Durch die unglaubliche Informationsmenge im Internet wuchsen die Probleme des Datenaustausches und der Repräsentation der Daten (siehe *Kapitel 2.1 – Das Internet*). Zum einen gibt es eine unüberschaubare Palette von Datenformaten für unterschiedlichste Einsatzgebiete, zum anderen den Standard HTML für die Darstellung von interaktiven Dokumenten, mit den bereits beschriebenen Problemen. Die Wichtigkeit der Präsentation wird heute durch Fragen der Datenbankanbindung, der redaktionellen Unterstützung und der Unternehmensintegration verdrängt. “Gefragt ist die Einfachheit von HTML mit der Mächtigkeit von SGML, jedoch ohne die Einschränkung der ersten und die Komplexität der letzteren.“[1] Für den durchschnittlichen Anwender ist die Metasprache SGML mit ihrer 500 Seiten umfassenden Spezifikation zu mächtig und zu kompliziert. Aufgrund dieser geringen Akzeptanz von Seiten der Mehrheit der Anwender, der Erkenntnis, dass viele Regelwerke von SGML überhaupt nicht angewendet wurden, den geschilderten Problemen der Auszeichnungssprache HTML und der rasanten Entwicklung des WWW, stellte das W3C eine SGML Expertengruppe unter Leitung von Jon Bosak, einem Systemarchitekten bei Sun, zusammen, welche ein SGML- Derivat auf HTTP Basis erstellen sollten.

Die wichtigsten Entwurfsziele waren hierbei:

- XML soll zu SGML kompatibel sein, damit vorhandene SGML-Werkzeuge und SGML-Anwendungen mit XML arbeiten können

[1] vgl. [MERZ99] S.334

- XML soll einfach im Internet benutzbar sein, d.h. es soll die Anforderungen an Anwendungen in verteilten Netzwerkumgebungen erfüllen
- Die Anzahl der optionalen Merkmale soll möglichst klein sein, um keinen unnötigen Overhead zu erhalten und Inkompatibilität zwischen Dokument und Prozessor zu vermeiden
- XML-Dokumente sollen leicht erstellbar und auch für Menschen lesbar sein
- XML soll in möglichst vielen Anwendungsgebieten einsetzbar sein, um seine Akzeptanz zu fördern
- Der XML-Entwurf soll präzise und formal sein, um zu erreichen, dass die Sprache leicht zu verstehen und zu gebrauchen ist
- Platzsparende Darstellung ist von untergeordneter Bedeutung, da Klarheit den Vorrang vor Kürze haben soll

Unter Berücksichtigung dieser Entwurfsziele wurde eine ´abgespeckte´ Version (30 Seiten Spezifikation) und Teilmenge der Meta-Sprache SGML entwickelt, die das Thema dieser Arbeit darstellt: XML, die Extensible Markup Language. XML wurde 1996 als Diskussionsvorschlag vom W3C vorgestellt und 1998 unter Leitung des W3C als Standard in der Version 1.0 verabschiedet.

Der ´Stammbaum´ von XML ergibt sich also in der folgenden Form:

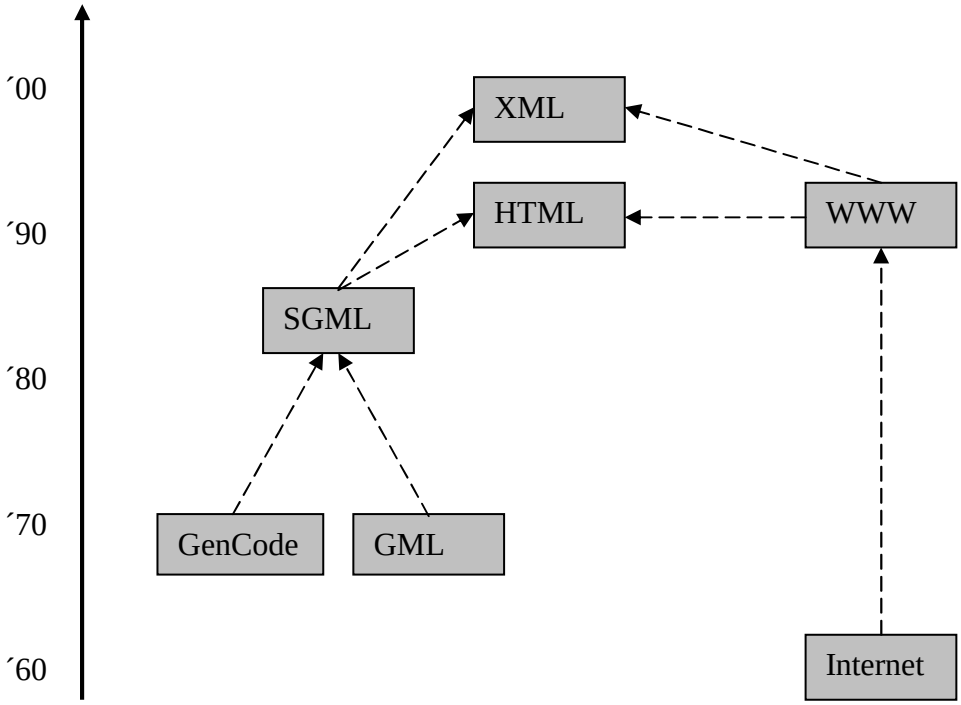

Abbildung 2-3 **Der XML-Stammbaum**

Eine sehr gelungene XML-Definition findet sich in *[JS1/00]*:

"XML: Eine standardisierte Möglichkeit strukturierte Informationen zu beschreiben, zu erfassen, zu speichern und weiterzugeben. Das Ganze flexibel, erweiterbar, selbsterklärend und da im reinen Textformat vorliegend, überall einsetzbar."

Dieses Schaubild soll den Zusammenhang zwischen SGML, XML und HTML noch einmal verdeutlichen:

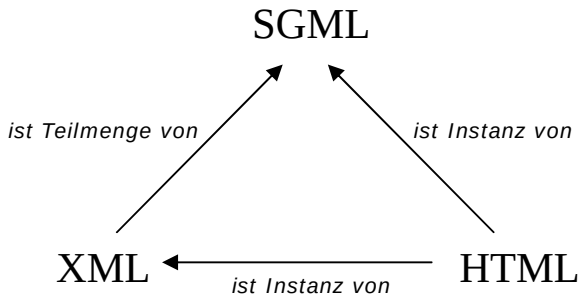

Abbildung 2-4 **Der Zusammenhang zwischen SGML, XML und HTML**

Weiterhin soll hier, der Vollständigkeit halber, der in *Kapitel 2.3 – Meta-Sprachen, Markup-Sprachen und Dokumente* angesprochene Vergleich zwischen XML und HTML stehen. Aufgrund der Definitionen sollen hier aber die Dokumente dieser unterschiedlichen Sprachen verglichen werden.

Merkmal	HTML-Dokument	XML-Dokument
Inhalt und Präsentation	vermischt	getrennt; Transformation in das jeweils geeignete Zielformat
Struktur	Nicht vorhanden oder implizit erschließbar	wohldefiniert
Bezeichner (Tags)	fest	frei wählbar; erweiterbar
Kontrolle der Tag-Verwendung	`lose`	Streng
Zeichensatz	Westeuropa	Unicode
Kontextsensitivität	nein	ja
Validierung	nicht möglich	möglich
Whitespaces entscheidend	nein	ja

Tabelle 2-1 **Vergleich HTML- und XML-Dokument**

2.5 Warum XML?

Mit XML ist ein solcher Hype verbunden, dass man im ersten Moment einfach misstrauisch werden muss. ´Schon wieder eine neue Technologie – kann man sich nicht endlich auf etwas einigen?´. Mittlerweile hat sich aber herausgestellt, dass XML wirklich viele Probleme lösen kann und eine bestätigte Daseinsberechtigung in der IT-Welt hat. Die meisten Konzepte, die in XML verwendet werden sind nicht neu. Wirklich neu und bahnbrechend ist die breite Unterstützung dieses Standards durch die Software-Industrie. Alle ´global player´, Freunde und Rivalen wie IBM, Microsoft, Sun, Oracle, SAP, usw. haben sich in Bezug auf diesen herstellerunabhängigen, ´freien´ Standard der Datenspeicherung und –kommunikation geeinigt. Diese Standardisierung erlaubt die rasche Verbreitung und Akzeptanz dieser Technologie. Viele Hersteller haben mittlerweile ihre proprietären Datenformate aufgegeben und nutzen XML. XML ist neben Java der Hoffnungsträger der IT-Welt: ein plattform- und applikationsübergreifendes Datenaustausch-, Datenverwaltungs- und Datendarstellungs-format, das die divergierenden IT-Infrastrukturen weiter zusammenbringen und dem WWW zu mehr Struktur und Ordnung verhelfen soll. Mit Java ist erwiesenermaßen portabler Code möglich (´Write Once – Run anywhere´), mit XML nun auch portable Daten (´Write Once – Read anywhere´). Ein weiteres wichtiges Merkmal ist die strikte Trennung des Inhaltes von dessen Präsentation, d.h. in einem XML-Dokument werden Aussagen darüber gemacht, um welche Informationen es sich handelt und wie diese strukturiert sind, aber nicht wie der Inhalt darzustellen ist. Hierzu existiert die mit XML verwandte Technologie XSL, auf die im Rahmen dieser Arbeit auch näher eingegangen werden soll. Weiterhin erlaubt XML es den Anwendern, ihre eigene auf den entsprechenden Anwendungsbereich abgestimmte Markup-Sprache zu definieren. Wichtig ist hierbei, dass die Struktur der Dokumente jeder mit XML erzeugten Markup-Sprache gleich bleibt und somit weiterhin unabhängig verarbeitet werden kann. XML ist, im Gegensatz zu anderen Technologien, leicht zu lernen und HTML- und SGML-Benutzer können, aufgrund der Verwandtschaft (siehe *Kapitel 2.4.3 – XML*), sehr leicht auf diese neue Sprache umsteigen. Außerdem ist XML für Computer einfach zu verarbeiten, auszutauschen und darzustellen. Weiterhin sind sehr viele Entwicklungswerkzeuge in diesem Umfeld frei erhältlich und vor allen Dingen standardkonform.

2.6 Die XML Welt

Nachdem nun XML positioniert, die Notwendigkeit und die wirtschaftlichen Aspekte aufgezeigt sind, sollen nun das Umfeld und die technische Seite, d.h. Syntax und Semantik der XML, vorgestellt werden.

2.6.1 Die XML-Familie

Wie bereits angesprochen existiert der Standard XML nicht allein, sondern wird vielmehr von einer Anzahl von anderen Spezifikationen vervollständigt, die Dinge wie Formatieren, Verknüpfen, Darstellen, Abfragen, usw. erlauben. Als Gesamtheit betrachtet ergeben diese Technologien die sogenannte ´XML-Familie´. Basierend auf dieser entstanden und entstehen eine Vielzahl von XML Applikationen als konkrete Anwendungen.

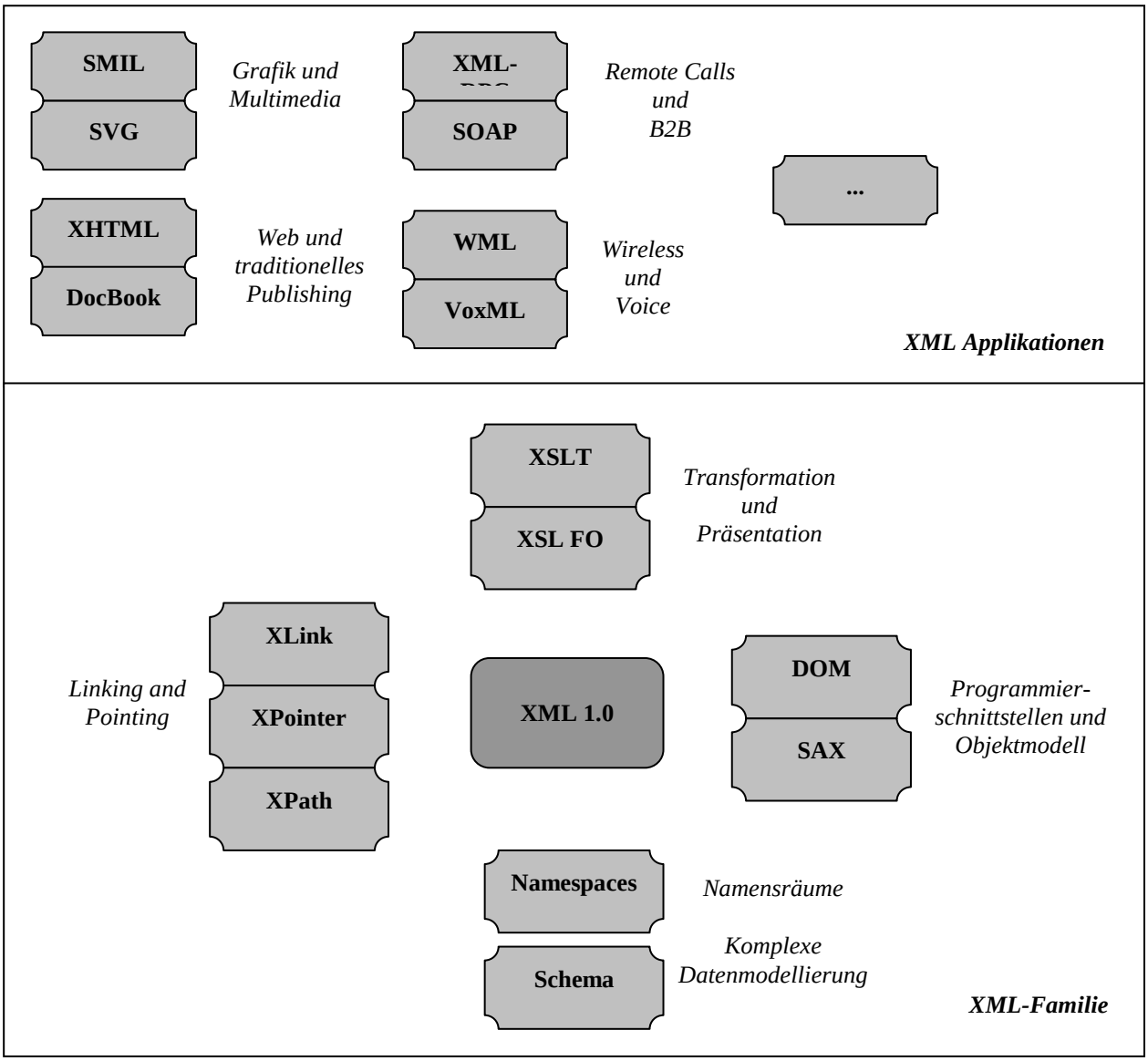

Abbildung 2-5 **XML-Familie und XML-Awendungen**

- XLink (XML Link Language):
  erlaubt multiple und bidirektionale Verknüpfungen von XML-Dokumenten
- XPointer (XML Pointer Language):
  definiert ein Adressierungsschema, mit dessen Hilfe innerhalb eines URI auf interne Strukturen eines Dokumentes zugegriffen werden kann um z.B. mittels Web-Browser auf einen bestimmten Dokumentenabschnitt zuzugreifen
- XPath (XML Path Language):
  definiert einen allgemeinen Mechanismus zur Navigation innerhalb von XML-Dokumentenbäumen
- XSLT (Extensible Stylesheet Language Transformations):
  XML-Sprache, welche über frei definierbare Regeln die Transformation von XML-Dokumenten in andere XML-Dokumente oder andere Formate erlaubt und damit als Vokabularkonverter zwischen den XML-Dialekten fungieren kann (siehe *Kapitel 3.3 – XSLT (Extensible Stylesheet Language Transformations)*)
- XSL FO (Extensible Stylesheet Language Formatting Objects):
  diese XML-Sprache bietet mediumunabhängige Aufbereitungsmöglichkeiten von XML-Dokumenten (siehe *Kapitel 3.4 – XSL FO (Extensible Stylesheet Language Formatting Objects)*)
- DOM (Document Object Model):
  ermöglicht den einheitlichen plattform- und sprachunabhängigen Zugriff auf XML-Dokumente (siehe *Kapitel 2.6.4.1 – DOM (Document Object Model)*)
- SAX (Simple API for XML):
  ermöglicht den einheitlichen plattform- und sprachunabhängigen Zugriff auf XML-Dokumente (siehe *Kapitel 2.6.4.2 – SAX (Simple API for XML)*)
- XML Namespaces:
  zur Vermeidung der Mehrdeutigkeiten von Metainformationen (siehe *Kapitel 2.6.6 – XML Namespaces*)
- XSchema:
  dient zur Definition von XML Schemata, die als Alternative zu DTDs dienen (siehe *Kapitel 2.6.7 - XSchema*)
- SMIL (Synchronized Multimedia Integration Language):
  eine XML-Sprache zur Beschreibung von Anwendungen, die Text, Bilder, Video- und Audiodaten kombinieren und synchronisieren
- SVG (Scalable Vector Graphics):
  DTD-Spezifikation zur XML-Codierung von Vektorgrafiken
- XHTML (XML Hypertext Markup Language):
  die XHTML 1.0 Spezifikation ist der offizielle Nachfolger der HTML 4.0 Spezifikation. Es handelt sich hierbei um eine abwärtskompatible Umsetzung von HTML auf eine XML-Applikation mit erweitertem Befehlsumfang
- DocBook:
  XML-Anwendung zur Erstellung von strukturierten Dokumenten, insbesondere von Büchern, Zeitschriften und Zeitungen
- XML-RPC:
  ein auf XML beruhender Standard für das Aufrufen entfernter Methoden über RPCs

- SOAP (Simple Object Access Protocol):
  XML-basiertes Interoperabilitätsprotokoll (siehe *Kapitel 4.2 – RPC mit SOAP*)
- WML (Wireless Markup Language):
  Mobilversion von HTML zum Zugriff auf das Internet über WAP-taugliche Handys
- VoxML (Voice Extensible Markup Language):
  dient als Grundlage eines Standards für die sprachgesteuerte Internet-Nutzung, durch den Anwender via Spracheingabe im Netz navigieren können

Die Anwendungen auf XML-Basis für bestimmte Fachgebiete verbreiten sich explosiv. Um die oben angeführten Anwendungen, die sich mittlerweile als Standards durchgesetzt haben, zu ergänzen, sollen hier einige weitere genannt werden:

- MathML - Mathematical Markup Language
- CML - Chemical Markup Language
- AML - Astronomy Markup Language
- BSML - Biosequence Markup Language
- MHTML (MIME Encapsulation of Aggregate HTML Documents):
  Neuer Standard für den E-Mail-Versand, welcher es ermöglicht eine komplette Website einschließlich aller Grafiken, Animationen, Rahmen, Tabellen und anderer Inhalte, anzuhängen
- XQL (XML Query Language):
  Abfrage-Sprache für XML-Datenbanken
- XMI (XML Metadata Interchange Format):
  ermöglicht das Serialisieren von Java-Objekten in XML-Dokumente zur Kommunikation verteilter Systeme

Als weitere sehr interessante Anwendung darf das ´Resource Description Framework´ (RDF) nicht fehlen. Es handelt sich hierbei um eine XML-basierte Sprache, die sich als Metasprache für Metadaten charakterisieren lässt. Mit deren Hilfe ist eine detaillierte Beschreibung von Dokumenten möglich, so dass Informationen im Internet katalogisiert und Dokumente von Maschinen gelesen und interpretiert werden können.

Begonnen werden soll hier mit der Syntax, Semantik und den Grundkonzepten der Sprache XML anhand eines einfachen Beispiels. Die Grundregeln, auf denen XML basiert, sind sehr weitreichend. Sind diese aber erst einmal verstanden, ist es nicht schwer, mit XML zu arbeiten.

2.6.2 Wohlgeformte und gültige XML-Dokumente

Alle XML-Dokumente, als konkrete Instanzen einer mit XML definierten Markup-Sprache, haben eine einfache universelle Grundstruktur, definiert durch die XML-Syntax. Nach dieser besteht ein XML-Dokument aus:

- Einem Prolog mit einer ´Startanweisung´, die das Element als XML-Dokument identifiziert und den verwendeten Zeichensatz festlegt sowie einem optionalen Verweis auf eine DTD (siehe *Kapitel 2.6.3 – DTD (Document Type Definition)*)
- Dem Hauptteil, bestehend aus einem Wurzelelement mit untergeordneten Elementen in Form einer Baumstruktur, wobei diese Elemente Attribute besitzen können.
- Einem optionalem Epilog, der Kommentare oder Verarbeitungsanweisungen enthalten kann.

Ein Element enthält eine Anfangsmarke (´Start-Tag´), eine Endmarke (´End-Tag´) und einen zwischen diesen Marken geklammerten Inhalt. Der Inhalt kann aus weiteren strukturierten Elementen oder reinem Text bestehen. Elemente können auch Attribute besitzen, welche in der Anfangsmarke definiert werden.

Ein Beispiel für ein solches Dokument befindet sich auf der nächsten Seite. In diesem Beispiel wird ein XML-Dokument dargestellt, wie es (im einfachsten Fall) bei einer elektronischen Bestellung auftreten könnte. Der HTML-Kenner wird hier sofort die Gemeinsamkeiten mit HTML erkennen, allerdings gelten folgende Regeln, die in HTML nicht zwingend sind:

- Elemente dürfen sich nicht überlappen
- Leere Elemente können in Kurzform benutzt werden
- Jedes Element muss mit einer Endmarke abgeschlossen werden
- Groß- und Kleinschreibung der Element- und Attributnamen ist entscheidend
- Attributwerte müssen in Anführungszeichen gesetzt werden

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE Bestellung SYSTEM "Bestellung.dtd" >

<Bestellung>
        <Datum>14.04.2001</Datum>

        <Kunde>
                <Name>Karl Klammer</Name>
                <EMail>Karl.Klammer@t-online.de</EMail>
        </Kunde>

        <Artikel Artikelnr="S0019">
                <Beschreibung>AMD Athlon 1100/256/30,5 – System</Beschreibung>
                <Menge>7</Menge>
                <Preis>2100 DM</Preis>
        </Artikel>

        <Artikel Artikelnr="P00123">
                <Beschreibung>Agfa-Snapscan 310 Scanner</Beschreibung>
                <Menge>17</Menge>
                <Preis>200 DM</Preis>
        </Artikel>

</Bestellung>
```

Abbildung 2-6 **Ein einfaches XML-Dokument**

2.6.2.1 Wohlgeformtheit

Erfüllt ein Dokument diese Syntaxregeln der XML, wird es, wie das angeführte Beispiel, als ´wohlgeformt´ (´wellformed´) bezeichnet. Die Prüfung auf Wohlgeformtheit übernimmt ein ´nicht validierender Parser´, näheres dazu später in *Kapitel 2.6.4 – Programmierschnittstellen und Parser*.

2.6.2.2 Gültigkeit

Die ´Wohlgeformtheit´ gibt an, ob ein Dokument den Syntaxregeln entspricht. Doch woher weiß man, ob die logische Struktur eines Dokumentes stimmt, d.h. ob es die richtigen Elemente in der richtigen Reihenfolge enthält, ob die Anzahl der Elemente stimmt, ob keine Elemente fehlen, usw.? Hierzu wird die Semantik, d.h. Grammatik und das Vokabular der Sprache, in Form einer DTD (Document Type Definition) festgelegt (siehe *Kapitel 2.6.3 – DTD (Document Type Definition)*). Stimmt ein Dokument mit dieser vorgegebenen Grammatik überein, wird es als ´gültig´ (´valid´) bezeichnet. Die Prüfung auf Gültigkeit übernimmt ein ´validierender Parser´. Ein gültiges Dokument ist gleichzeitig immer auch wohlgeformt.

2.6.3 DTD (Document Type Definition)

Die DTD beschreibt mit Hilfe der erweiterten Backus-Naur Form (EBNF) die Grammatik und das Vokabular einer XML-Sprache, d.h. sie ist ein Konstrukt, in dem sämtliche Element-Deklarationen, Attribut-Deklarationen, Notations-Deklarationen, Kardinalitäten, usw. zusammengefasst werden. Sie kann somit als Schablone auf ein XML-Dokument angewendet werden oder, um das Ganze objektorientiert auszudrücken, die DTD ist die Klasse und das XML-Dokument eine konkrete Instanz dieser Klasse. Eine solche DTD kann zur Festlegung einer Corporate Identity für alle geschäftlichen Dokumente dienen. Mit der passenden Software (siehe *Kapitel 2.6.4 – Programmierschnittstellen und Parser*) können diese Dokumente ohne Probleme verarbeitet, gespeichert, publiziert und weltweit ausgetauscht werden.

Eine DTD, welche das angeführte Beispiel in ein gültiges Dokument ´verwandeln´ würde und die entsprechende Markup Sprache (z.B. die BML = ´Bestellungs Markup Language´) definiert, könnte z.B. folgendermaßen aussehen:

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

<!-- DTD (Document Type Definition) fuer eine Bestellung -->

<!ELEMENT Bestellung        (Datum, Kunde, Artikel+)      >

<!ELEMENT Datum             (#PCDATA)                     >

<!ELEMENT Kunde             (Name, EMail?)                >
<!ELEMENT Name              (#PCDATA)                     >
<!ELEMENT EMail             (#PCDATA)                     >

<!ELEMENT Artikel           (Beschreibung, Menge, Preis)  >
<!ATTLIST Artikel           Artikelnr ID #REQUIRED        >
<!ELEMENT Beschreibung      (#PCDATA)                     >
<!ELEMENT Menge             (#PCDATA)                     >
<!ELEMENT Preis             (#PCDATA)                     >
```

Abbildung 2-7 **Eine einfache DTD**

In dieser DTD werden für konkrete Instanzen folgende Vorgaben gemacht:

- Das Element Bestellung besteht aus folgenden Elementen, die in dieser Reihenfolge aufgeführt werden müssen: Datum, Kunde und eine nichtleere Menge von Artikeln
- Das Element Datum besteht aus einfachem Text
- Das Element Kunde besteht aus Name und optionaler EMail-Adresse, welche Textelemente sind

- Das Element Artikel besteht aus einer Beschreibung, einer Menge und einem Preis, welche jeweils Textelemente sind
- Jeder Artikel hat zwingend ein Attribut Artikelnummer

2.6.4 Programmierschnittstellen und Parser

XML Parser sind Bestandteile von Anwendungsprogrammen, die XML-Dokumente verarbeiten. Das heißt, XML Parser analysieren Dokumente und stellen der Anwendung die benötigten Informationen über diese zur Verfügung. Mittels einheitlicher, plattform- und sprachunabhängiger APIs kann der Parser auf die Dokumente und die darin enthaltenen Daten zugreifen. Hierbei existieren zwei grundlegende dieser Schnittstellen: Die objektmodell-orientierte DOM-Schnittstelle und die ereignisorientierte SAX-Schnittstelle.

Zur Verdeutlichung dient dieses Schaubild:

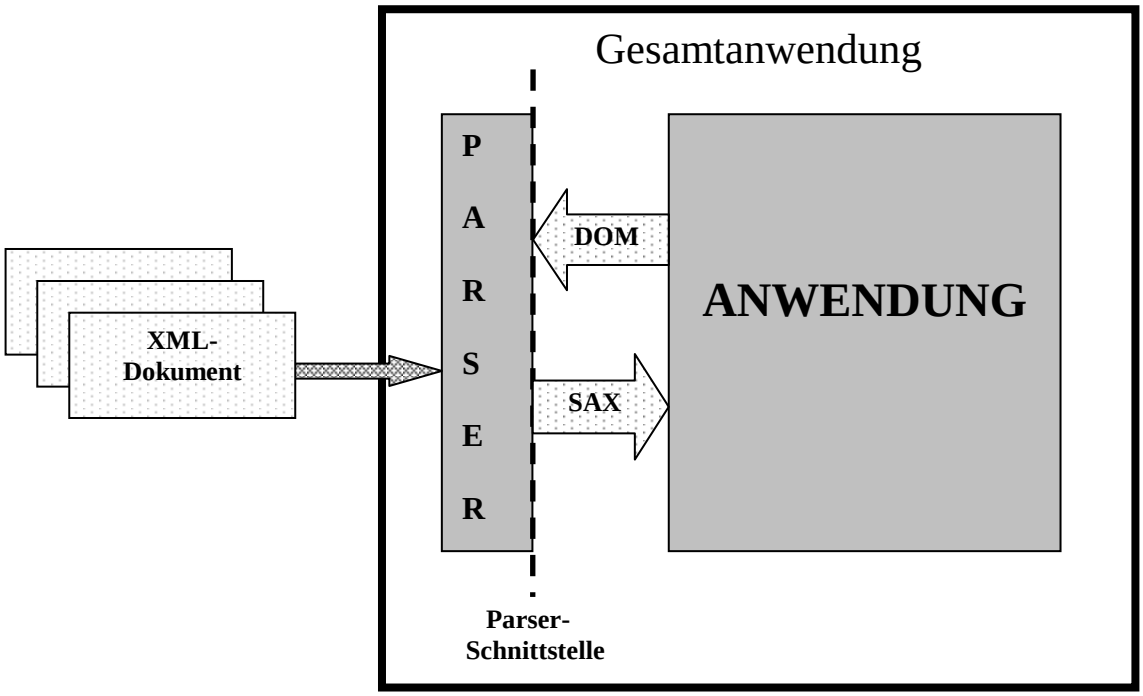

Abbildung 2-8 **Programmierschnittstellen und Parser**

2.6.4.1 DOM (Document Object Model)

Die DOM-Schnittstelle, auch als ´baumorientierte Schnittstelle´ bezeichnet, erlaubt, wie auch SAX, den standardisierten Zugriff auf beliebige HTML- und XML-Dokumente. Sie bietet Methoden zum Lesen und zur Manipulation der Baumstruktur eines Dokumentes an. Ein DOM-Parser verarbeitet (parst) zunächst das gesamte Dokument und erstellt intern einen ´Elementebaum´, der auch als ´Syntaxbaum´ oder ´DOM-Baum´ bezeichnet wird. Dieser Baum fungiert als Repräsentant des Dokumentes. Die an den Parser gekoppelte Anwendung kann nun über einheitliche Schnittstellen diesen Baum durchwandern und manipulieren, indem sie ihren Zugriff auf einzelne Knoten, deren Vorgänger und Nachfolger geschickt nutzt.

Das allgemeine Vorgehen bei Nutzung der DOM-Schnittstelle ist also die Objektmanipulation auf folgende Art und Weise:

1. Parser instanziieren
2. XML-Dokument vollständig lesen und dabei DOM-Baum aufbauen
3. Resultat-Baum durchwandern, durchsuchen und die Informationen manipulieren

Der Vorteil gegenüber der ereignisorientierten SAX-Schnittstelle liegt in der Möglichkeit der Navigation und Manipulation des Baumes, d.h. es ist hier möglich die Baumstruktur vorwärts und rückwärts zu durchwandern. Der große Nachteil dieser Methode besteht darin, dass das gesamte Dokument geparst werden muss, bevor man auf dessen Inhalt zugreifen kann und dieser Vorgang bei großen Dokumenten und somit großen Resultat-Bäumen sehr speicherintensiv ist[1]. Um dieses Problem zu lösen, wurde in der XML-Gemeinde die SAX-Schnittstelle vorgeschlagen und unter dem Motto “Don´t call the DOM, the parser calls you!“ standardisiert.

[1] Dieses Problem tritt insbesondere beim Einsatz von XML in Zusammenhang mit Streaming-Technologien auf.

Der DOM-Baum des Beispieles im *Kapitel 2.6.2 – Wohlgeformte und gültige XML-Dokumente* sieht folgendermaßen aus:

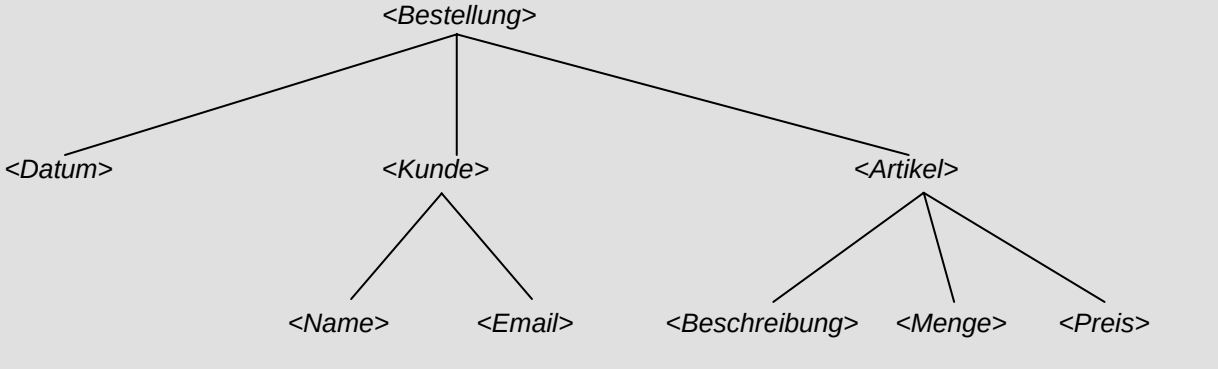

Abbildung 2-9 **DOM-Baum**

2.6.4.2 SAX (Simple API for XML)

Die SAX-Schnittstelle, auch als ´ereignisorientierte Schnittstelle´ bezeichnet, erlaubt, wie auch DOM, den standardisierten Zugriff auf beliebige XML-Dokumente. Hierbei wird jedoch nicht der DOM-Baum aufgebaut und anschließend bearbeitet. Ein SAX-Parser liest das XML-Dokument sequentiell und ruft bei Ereignissen, wie z.B. ´Dokumentenanfang´ oder ´End-Tag´ entsprechende Methoden (Event-Handler-Methoden) auf, deren Rümpfe der Anwendungsprogrammierer seinen Bedürfnissen entsprechend gestalten kann.

Das allgemeine Vorgehen bei Nutzung der SAX-Schnittstelle ist also die Ereignisbehandlung auf folgende Art und Weise:

1. Parser instanziieren
2. Event-Handler instanziieren
3. Event-Handler beim Parser registrieren
4. Dokument sequentiell einlesen
5. Bei jedem Ereignis den entsprechenden Event-Handler informieren

Der Vorteil gegenüber der baumorientierten DOM-Schnittstelle liegt in der vielfach höheren Verarbeitungsgeschwindigkeit der SAX-Parser, da kein vollständiger DOM-Baum aufgebaut werden muss und die an den Parser gekoppelte Anwendung die für sie unwichtige Teile des Dokumentes ignorieren kann. Weiterhin besteht hierdurch die Möglichkeit sehr große Dokumente und Streams zu verarbeiten. Allerdings ist die SAX-Schnittstelle aufgrund der fehlenden Möglichkeit der Baum-Manipulation nicht für jeden Anwendungsfall geeignet.

Die Ausgabe eines SAX-Parsers, der das o.a. Dokument parst und bei jedem Ereignis, das gefundene Element inklusive Inhalt auf dem Bildschirm ausgibt, könnte folgendermaßen aussehen:

```
Initialising SAX Parser ...

Start parsing Bestellung1.xml...
LOCATOR
 SYS ID: file:///D:/Uni/Giessen VII/Seminar/Tests/SAXParser/Bestellung1.xml

START DOCUMENT
<?xml version='1.0' encoding='UTF-8'?>
   ELEMENT: <Bestellung>
      ELEMENT: <Datum>
      CHARS:
            14.04.2001

      END_ELM: </Datum>
      ELEMENT: <Kunde>
         ELEMENT: <Name>
         CHARS:
                  Karl Klammer

         END_ELM: </Name>
         ELEMENT: <EMail>
         CHARS:
                  Karl.Klammer@t-online.de

         END_ELM: </EMail>
      END_ELM: </Kunde>
      ELEMENT: <Artikel
        ATTR: Artikelnr      "S0019"
      >
         ELEMENT: <Beschreibung>
         CHARS:
            AMD Athlon 1100/256/30,5 - System

         END_ELM: </Beschreibung>
         ELEMENT: <Menge>
         CHARS:
            7

         END_ELM: </Menge>
         ELEMENT: <Preis>
         CHARS:
            2100 DM

         END_ELM: </Preis>
      END_ELM: </Artikel>
      ELEMENT: <Artikel
        ATTR: Artikelnr      "P00123"
      >
         ELEMENT: <Beschreibung>
         CHARS:
            Agfa-Snapscan 310 Scanner

         END_ELM: </Beschreibung>


         ELEMENT: <Menge>
```

```
        CHARS:
           17

        END_ELM: </Menge>
        ELEMENT: <Preis>
        CHARS:
           200 DM

        END_ELM: </Preis>
     END_ELM: </Artikel>
  END_ELM: </Bestellung>
END DOCUMENT
press any key to exit...
```

Abbildung 2-10 **Ausgabe einer SAX-Parser Anwendung**

2.6.5 XML - Dokumente oder Daten?

Eine konkrete XML-Instanz, welche meist als XML-Dokument bezeichnet wird, fällt immer in eine der beiden Kategorien:

- XML-Dokument (im engeren Sinne)
- XML-Daten

XML unterscheidet nicht, ob eine Instanz ein Dokument oder Daten enthält, doch bei der Benutzung bestimmter Werkzeuge zur Bearbeitung von XML-Instanzen ist diese Unterscheidung wichtig. Meist kann die Kategorie durch bloßes Betrachten bestimmt werden:

Typische XML-Dokumente werden primär vom Menschen geschrieben und gelesen. Hierzu zählen zum Beispiel Bücher, Zeitschriften, Webseiten usw. Solche Dokumente enthalten hauptsächlich Fließtext, einige Überschriften, Tabellen oder Bilder. Das heißt der Inhalt dieser Elemente wird als reiner Text interpretiert.

Typische XML-Daten sind Rechnungen, Bestellungen, Kataloge, Preislisten und andere strukturierte Informationen, die primär für die maschinelle Verarbeitung und den Datenaustausch gedacht sind. Solche Dokumente enthalten oft ´hochstrukturierte´ Daten, die bestimmte Integritätsbedingungen erfüllen und typisiert sein müssen. (siehe dazu *Kapitel 2.6.7 - XSchema*).

2.6.6 XML Namespaces

Innerhalb eines XML-Dokumentes muss jeder Element- und Attributname eine eindeutige Bedeutung besitzen, ansonsten ist das Dokument nicht gültig (siehe *Kapitel 2.6.2.2 Gültigkeit*). Da man aber sprechende Namen für Elemente und Attribute verwenden sollte, kann dies bei komplexeren Systemen oft zu Mehrdeutigkeiten und somit zu Namenskollisionen führen. So kann z.B. ein Tag <*Titel*> sowohl für die Anrede als auch für den Titel einer Veröffentlichung stehen. Zur Auflösung dieser Mehrdeutigkeiten von Metainformationen, die durch die rasant steigende Anzahl von DTDs und XML Schemata unausweichlich wird, dienen XML Namespaces.

Um Mehrdeutigkeiten aufzulösen, qualifiziert man die mehrdeutigen Namen durch Bindung an URIs. Ein Namensraum ist eine eindeutig identifizierte Sammlung von Namen für Elemente und Attribute, qualifiziert durch einen URI. Ein Beispiel soll dies verdeutlichen:

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

<bestellung     xmlns:sender="http://www.sender.com"
                xmlns:empfänger="http://www.empfänger.de">

        <sender:artikel artikelnr="S0019">
                <beschreibung>AMD Athlon 1100/256/30,5 – System</beschreibung>
                <menge>7</menge>
                <preis>2100 DM</preis>
        </sender:artikel>

        <empfänger:artikel>
                <beschreibung> AMD Athlon 1100/256/30,5 – System</beschreibung>
                <menge>7</menge>
                <währung>DM</währung>
                <preis>2100</preis>
        </empfänger:artikel>

</bestellung>
```

Abbildung 2-11 **Ein einfaches Namensraum-Beispiel**

Hierbei werden innerhalb des Elementes <*bestellung*> zwei Namensräume in der Form <*elementname* xmlns:*namensraumprefix* = "*uri*"> deklariert. Die Verwendung eines Elementes geschieht durch Voranstellen des entsprechenden Prefix und die Trennung durch einen Doppelpunkt. So ist es, wie in diesem Beispiel beim Element <*artikel*>, möglich, zwei Elemente bzw. Attribute gleichen Namens in unterschiedlichen Kontexten zu benutzen.

2.6.7 XSchema

Das Konzept der DTD stammt aus der dokumentenorientierten Welt der SGML. XML wird aber nicht nur zur Dokumentenverwaltung, sondern auch für den Datenaustausch benutzt (siehe *Kapitel 2.6.5 – XML - Dokumente oder Daten?*). In DTDs können dagegen nur Textelemente definiert werden, was für den Datenaustausch nicht ausreichend ist. XSchema erweitert die Funktionalität der DTDs abwärtskompatibel und unterstützt viele gängige Datentypen wie String, Real, Boolean, Date, Language, sowie selbstdefinierte Typen, Vererbungsmechanismen, Integritätsbedingungen, Listen, Referenzen usw. Ein weiterer Vorteil der XSchema-Sprache besteht darin, dass sie selbst in XML formuliert ist und somit im Gegensatz zur DTD keinen speziellen Parser benötigt. Die Nachteile der XSchema gegenüber der DTD sind die mittlerweile sehr große Verbreitung und Kenntnis von DTDs, die Vielzahl von Werkzeugen und bereits auf Basis von DTDs defininierten Sprachen.

Bei Benutzung von XSchema wird ein Dokument nicht mehr auf Gültigkeit (siehe *Kapitel 2.6.2.2 Gültigkeit*) sondern auf ´Schema-Konformität´ geprüft. Weiterhin existieren im Gegensatz zu DTDs drei Ebenen der Definition:

- Typ-Definition (erfolgt im Schema)
- Element- und Attribut-Deklarationen, die diese Typen verwenden (erfolgt im Schema)
- Element-Definitionen, d.h. Element-Instanzen (erfolgt im Dokument)

Das folgende kurze Beispiel soll die Funktionsweise von XSchema verdeutlichen. Zuerst wird im Schema ein neuer Typ namens *nameT* definiert, der sich aus zwei Strings und einem optionalem Datum zusammensetzt. Im zweiten Schritt werden zwei Elemente auf Basis dieses Types deklariert, die im dritten Schritt im Dokument instanziiert werden.

```
<xsd:schema xmlns=http://www.w3.org/2000/08/XMLSchema>

        <!-- Den Typ ´NameT´ definieren -->
        <xsd:complexType name="nameT">
                <xsd:sequence>
                        <xsd:element name="vorname"   type="xds:string"/>
                        <xsd:element name="nachname" type="xds:string"/>
                </xds:sequence>
                <xsd:attribute name="geburtsdatum" type="xsd:date" use="optional"/>
        </xsd:complexTyp>

   <!-- Den neuen Typ in zwei Elementdeklarationen verwenden -->
   <xsd:element name="name" type="nameT"/>
   <xsd:element name="autor " type="nameT"/>

</xsd:schema>
```

Abbildung 2-12 Ein XSchema (´name.xsd´)

```
<!-- Elemente vom Typ nameT -->

<name geburtsdatum="1955-02-02">
   <vorname> Karl </vorname>
   <nachname> Klammer </nachname>
</name>

<autor>
   <vorname> Eva </vorname>
   <nachname> Eifrig </nachname>
</autor>
```

Abbildung 2-13 Benutzung der im Schema deklarierten Elemente

3 Was ist XSL?

In *Kapitel 2 – Was ist XML?* wurde der Vorteil der strikten Trennung zwischen Inhalt und dessen Präsentation hervorgehoben und gezeigt, dass ein XML-Dokument, im Gegensatz zu einem HTML-Dokument, keinerlei Informationen über die Präsentation beinhaltet. Nun stellt sich aber die Frage ´Wie präsentiere ich denn nun die Inhalte des XML-Dokumentes?´. Hierzu dient die selbst in XML definierte Sprache XSL (Extensible Stylesheet Language), die als entkoppelte Präsentationsebene fungiert und in diesem Kapitel vorgestellt und anhand von Beispielen erläutert werden soll.

3.1 Geschichte

Wie auch XML blickt die Sprache XSL auf eine lange Geschichte zurück. Ihr Vorgänger war der SGML Co-Standard DSSSL (Document Style Semantics and Specification Language, sprich: ´Dissel´), eine algorithmisch vollständige Skriptsprache, anhand derer die Visualisierung von SGML-Dokumenten realisiert wurde und wird. Analog zum Verhältnis SGML-XML ist XSL der ´kleine Bruder´ von DSSSL. XSL ist gegenüber DSSSL hinsichtlich der Mächtigkeit etwas reduziert, um es einfacher nutzbar und im WWW schneller einsetzbar zu machen. Es basiert auf ECMAScript, der standardisierten Form von Netscapes JavaScript.

3.2 Eine Übersicht

XSL ermöglicht es, ein XML-Dokument auf verschiedenen Medien darzustellen, oder um es anders auszudrücken, durch XSL werden verschiedene Sichten auf ein XML-Dokument ermöglicht (´Views-Konzept´). So kann z.B. ein XML-Dokument, welches aktuelle Börsennachrichten enthält, im Internet als HTML-Dokument publiziert werden, als PDF zur Druckausgabe zur Verfügung gestellt werden, an ein WAP-Handy in Form eines WML-Dokumentes geschickt werden, in Sprachausgabe umgewandelt werden oder im reinen Textformat als E-Mail oder SMS versendet werden. Andere Anwendungen sind das automatische Generieren einer Antwort aus einer Anfrage im Online-Shopping Bereich, die Transformation von XML-Dokumenten in EDI-Nachrichten (siehe *Kapitel 4.3 – E-Commerce mit BizTalk und ebXML*), die Darstellung eines Dokumentes in unterschiedlichen Anwendungskontexten (extern, intern, Management, Mitarbeiter), die Vereinheitlichung von Dokumenten, das Filtern von Informationen, usw.

Abbildung 3-1 – Trennung von Daten und Präsentation soll die Funktionsweise von XSL verdeutlichen. Soweit möglich soll sowohl Datenhaltung, als auch Datenaustausch auf XML-Basis durchgeführt werden. Ist es aber notwendig, Informationen aus XML-Dokumenten dem Menschen zugänglich zu machen, wird das XML-Dokument in das entsprechende Zielformat umgewandelt.

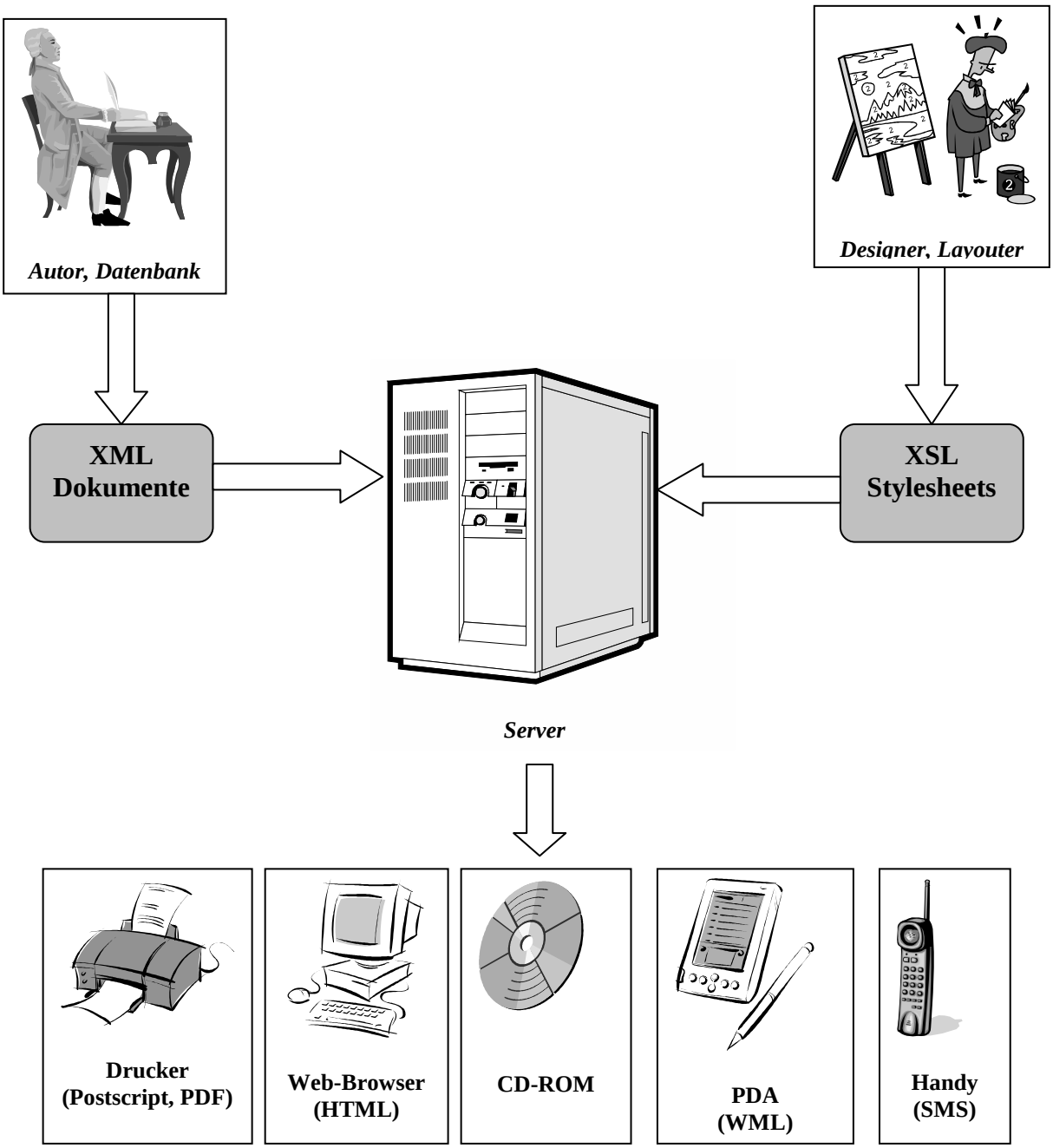

Abbildung 3-1 **Trennung von Daten und Präsentation**

XSL besteht aus zwei Sprachen: aus der Transformationssprache XSLT (Extensible Stylesheet Language Transformations) und der Formatierungssprache XSL FO (Extensible Stylesheet Formatting Objects).

3.3 XSLT (Extensible Stylesheet Language Transformations)

XSLT ist eine deklarative Sprache, mit welcher beschrieben wird, wie ein XML-konformes Dokument in ein anderes, meist auch XML-konformes Dokument, umgewandelt werden soll. Die Umsetzung kann das alte Vokabular übernehmen, verändern oder vollständig ersetzen. Die bekanntesten Anwendungen sind die Überführung von semantischen XML-Dokumenten in Anzeigestruktur-Dokumente wie HTML, um diese grafisch aufbereitet in einem Browser zu präsentieren, sowie das Konvertieren von XML-Dokumenten zwischen verschiedenen Vokabularen, also z.B. zwischen dem internen Format der Firma A und dem der Firma B. Eine beispielhafte Anwendung wäre des automatische Generieren eines ´Inhaltsverzeichnis-XML-Dokumentes´ aus den Kapitelüberschriften eines sehr großem XML-Dokumentes.

3.3.1 Das Prinzip

Die Transformation eines XML-konformen Dokumentes in ein anderes XML-konformes Dokument geschieht nach folgendem Prinzip: Ein XSLT-Prozessor liest das Quell-Dokument sequentiell ein und vergleicht in jedem Knoten, ob dieser einem, in der Stylesheet-Datei angegebenen, Muster entspricht. Trifft dies zu, wird die entsprechende, auch im Stylesheet angegebene, Aktion ausgeführt. Ansonsten wird das Element kopiert.

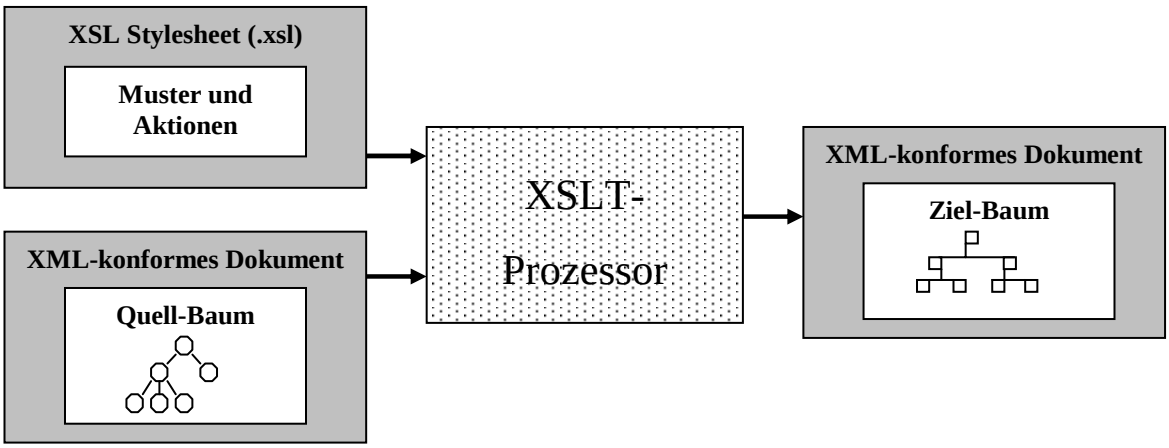

Abbildung 3-2 **Das Prinzip der XSLT**

Eine XSL Stylesheet-Datei besteht aus einem Satz von Regeln, welche festlegen, wie das Quelldokument, d.h. der Quellbaum, in das Zieldokument, also den Zielbaum, konvertiert werden soll.

Eine solche Konstruktionsregel gliedert sich in folgende Teile:

- Muster (pattern):
  Das Muster gibt an, auf welche Bezeichner innerhalb des XML-Dokumentes sich die darauf folgende Aktion beziehen soll, d.h., findet der Parser das angegebene Muster, wird die entsprechende Aktion ausgeführt.
- Aktion (action):
  Wurde das angegebene Muster erkannt, wird die darauffolgende Aktion ausgeführt. Diese Aktion kann aus der entsprechenden Umwandlung des betroffenen Elementes in die deklarierte Ausgabeform, d.h. das korrespondierende Element des Zielbaumes (Template, Schablone), oder aus einer dynamischen Anweisungen, z.B. Aktivierung eines Skriptes, bestehen.

Zur Verdeutlichung soll folgendes Beispiel dienen: Die bereits in *Kapitel 2.6.2 – Wohlgeformte und gültige XML-Dokumente* benutzte elektronische Bestellung, soll nun sowohl in ein Textdokument, als auch in eine HTML-Datei transformiert werden, um sie darzustellen.

Hier noch einmal die zu transformierende XML-Datei *Bestellung1.xml*:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE Bestellung SYSTEM "Bestellung.dtd" >

<Bestellung>
        <Datum>
                14.04.2001
        </Datum>

        <Kunde>
                <Name>
                        Karl Klammer
                </Name>
                <EMail>
                        Karl.Klammer@t-online.de
                </EMail>
        </Kunde>

  <Artikel Artikelnr="S0019">
        <Beschreibung>
                AMD Athlon 1100/256/30,5 - System
        </Beschreibung>
        <Menge>
                7
        </Menge>
        <Preis>
                2100 DM
        </Preis>
  </Artikel>
```

```
    <Artikel Artikelnr="S0017">
        <Beschreibung>
            P3 800/128/20,5 - System
        </Beschreibung>
        <Menge>
            3
        </Menge>
        <Preis>
            1500 DM
        </Preis>

    </Artikel>

    <Artikel Artikelnr="P00123">
        <Beschreibung>
            Agfa-Snapscan 310 Scanner
        </Beschreibung>
        <Menge>
            17
        </Menge>
        <Preis>
            200 DM
        </Preis>
    </Artikel>

</Bestellung>
```

Abbildung 3-3 **Die mittels XSL umzuwandelnde XML-Datei**

Bei Benutzung des u.a. XSL Stylesheets (*Bestellung2Text.xsl*) wird die Bestellung im XML-Format in einfachen Text umgewandelt, was durch *output method="text"* deutlich wird. Man sieht hier sehr schön die Definition der Regeln und ihrer zugehörigen Aktionen. Der erste Punkt bedeutet: Wenn der aktuelle Knoten den Namen *Bestellung* hat, füge den Text *BESTELLUNG* und einen Strich zur Begrenzung ein, bearbeite alle weiteren Knoten und füge einen weiteren Strich zur abschließenden Begrenzung ein. Der zweite Punkt bedeutet: Wenn der aktuelle Knoten den Namen *Datum* hat, füge den Text *vom* ein und setze den Wert des aktuellen Knotens, d.h. das Datum der Bestellung, dahinter ein. Auf die gleiche Art und Weise wird mit den Knoten *Kunde* und *Artikel* verfahren.

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

<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">

        <xsl:output method="text" indent="no"/>

        <xsl:template match="Bestellung">
            BESTELLUNG
            ----------
            <xsl:apply-templates/>
            ----------
        </xsl:template>


        <xsl:template match="Datum">
```

```
            vom <xsl:value-of select="."/>
        </xsl:template>


        <xsl:template match="Kunde">
            Herr/Frau: <xsl:value-of select="."/>
        </xsl:template>


        <xsl:template match="Artikel">
            <xsl:number/> <xsl:value-of select="."/>
        </xsl:template>

</xsl:stylesheet>
```

Abbildung 3-4 **Ein XSL Stylesheet zur Umwandlung in Text**

Das Ergebnis der Umwandlung der XML-Datei in eine Text-Datei sieht folgendermaßen aus:

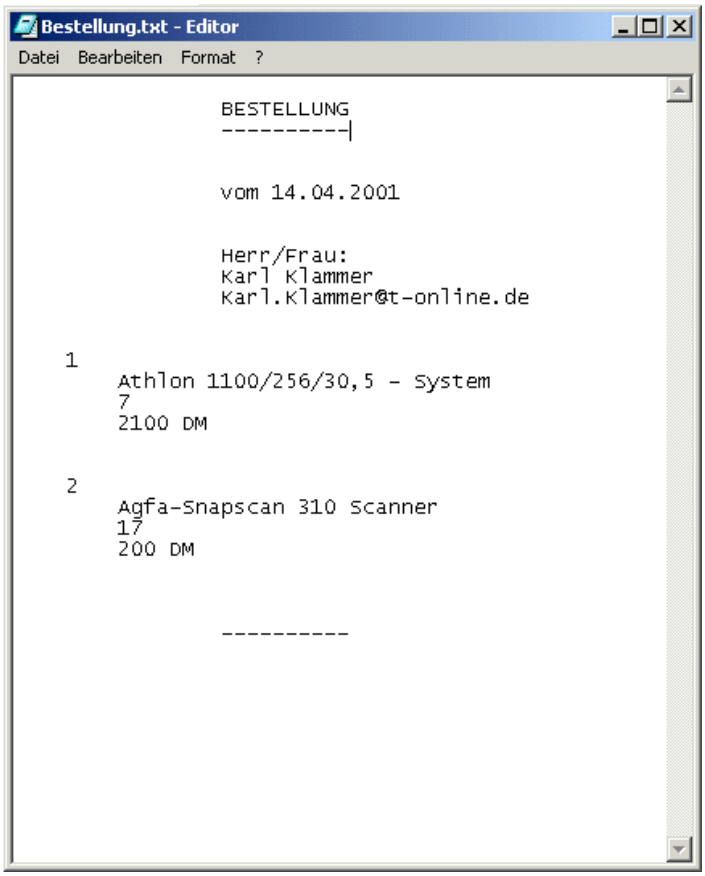

Abbildung 3-5 **Das Ergebnis der Umwandlung in das Text-Format**

Bei Benutzung des zweiten u.a. XSL-Stylesheets (*Bestellung2HTML.xsl*) wird die Bestellung im XML-Format in ein HTML-Dokument umgewandelt, was durch *output method="html"* deutlich wird.

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

<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">

	<xsl:output method="html" indent="yes"/>

	<xsl:template match="Bestellung">
		<html>
			<head>
				<h1>BESTELLUNG</h1>
			</head>

			<body>
				<xsl:apply-templates select="./Datum"/>
				<br></br>
				<br></br>
				<xsl:apply-templates select="./Kunde"/>
				<br></br>
				<table border="3">
					<tr>
						<th>Nummer</th>
						<th>Beschreibung</th>
						<th>Menge</th>
						<th>Preis</th>
					</tr>
					<xsl:apply-templates select="./Artikel"/>
				</table>
				<br></br>
			</body>
		</html>
	</xsl:template>

	<xsl:template match="Datum">
		<h2> vom <xsl:value-of select="."/> </h2>
	</xsl:template>

	<xsl:template match="Kunde">
		<h2> Herr/Frau: <xsl:value-of select="Name"/> </h2>
		<h3> E-Mail: <xsl:value-of select="EMail"/> </h3>
	</xsl:template>

	<xsl:template match="Artikel">
		<tr>
			<td><xsl:number/></td>
			<td><xsl:value-of select="./Beschreibung"/></td>
			<td><xsl:value-of select="./Menge"/></td>
			<td><xsl:value-of select="./Preis"/></td>
		</tr>
	</xsl:template>


</xsl:stylesheet>
```

Abbildung 3-6 Ein XSL Stylesheet zur Umwandlung in HTML

Das Ergebnis dieser Umwandlung der XML-Datei in eine HTML-Datei sieht folgendermaßen aus:

Abbildung 3-7 **Das Ergebnis der Umwandlung in das HTML-Format**

3.4 XSL FO (Extensible Stylesheet Language Formatting Objects)

XSL FO ist eine Sprache, mit der beschrieben wird, wie ein Formatierer Inhalte auf dem Ausgabemedium platzieren soll. Dies bedeutet, hier wird ein XML-konformes Dokument in ein nicht XML-konformes Dokument, wie z.B. einem Format für Printmedien wie PDF umgewandelt. Es können hierbei verschiedene Einstellungen zu den Rändern, Füll- und Hintergrundeigenschaften, Schrifteigenschaften usw. eingestellt werden. XSL FO zielt nicht auf die Fähigkeiten eines Programmierers ab, sondern die eines Designers oder Layouters.

3.4.1 Das Prinzip

Die Transformation eines XML-konformen Dokumentes in ein nicht XML-konformes Dokument für Printmedien geschieht nach folgendem Prinzip: Das Quelldokument wird hierzu zuerst per XSLT in einen FO-Baum umgewandelt und dieser dann mit einem FO-Prozessor in das nicht XML-konforme Formate übersetzt (PDF, RTF, Excel, Powerpoint, usw.).

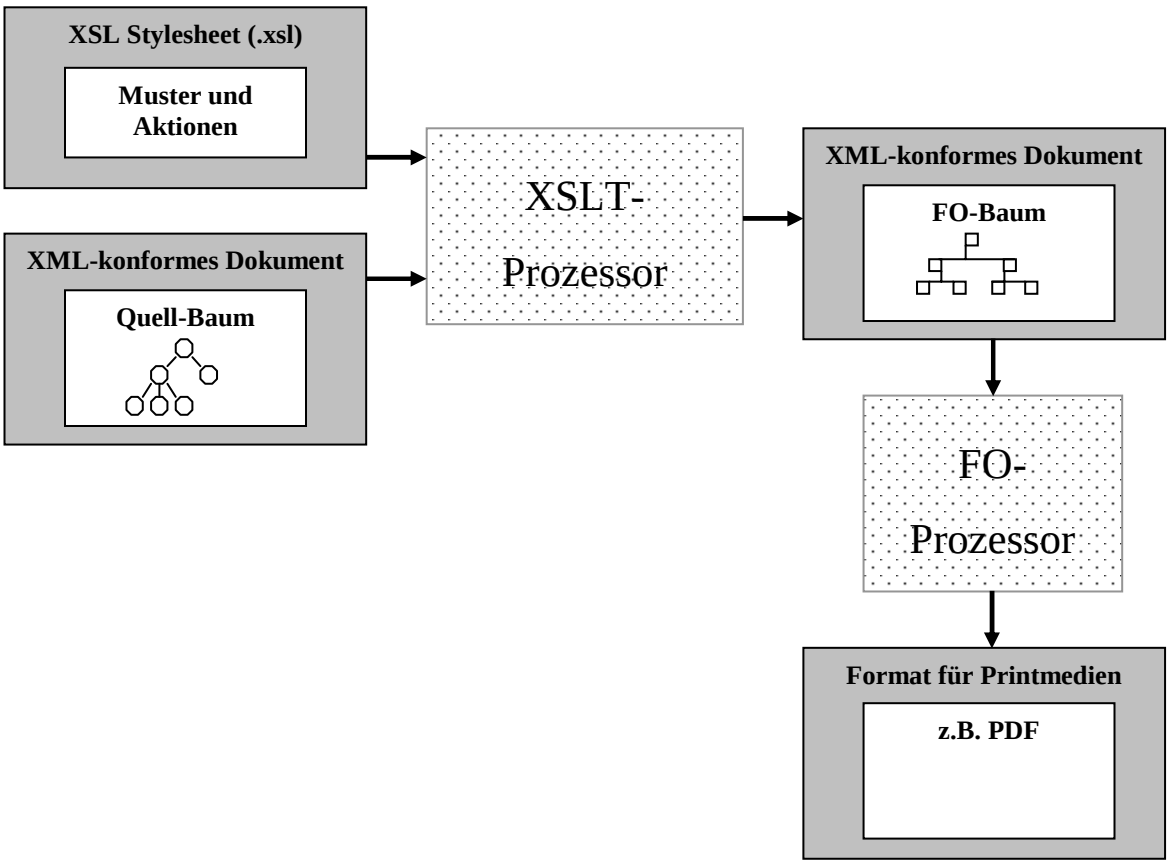

Abbildung 3-8 **Das Prinzip der XSL FO**

Zur Verdeutlichung soll folgendes Beispiel dienen: Die elektronische Bestellung soll nun in ein PDF-Dokument transformiert werden, um es drucken zu können.

Das benötigte XSL Stylesheets (*Bestellung2HTML.xsl*) zur Umwandlung in einen FO-Baum ist weiter unten dargestellt und soll hier nicht näher erläutert werden. Man sieht deutlich, welcher Aufwand in einem solchen Stylesheet steckt, weshalb sich die automatische Generation mit Hilfe von Tools anbietet, von denen zur Zeit noch relativ wenige existieren.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- XSL zur Erzeugung eines FOs (Formatting Objects) zur PDF-Generierung-->

<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"

xmlns:fo="http://www.w3.org/1999/XSL/Format">

        <!-- MATCH: "Bestellung" -->
        <xsl:template match="Bestellung">

                <!-- FO INFORMATIONEN -->
                <fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">

                        <!-- Definition der Seiten -->
                        <fo:layout-master-set>

                        <!-- Seitenlayout Informationen -->
                        <fo:simple-page-master master-name="Einfache Seite"
                                        page-height="29.7cm"
                                        page-width="21cm"
                                                margin-top="1cm"
                                                margin-bottom="2cm"
                                                margin-left="2.5cm"
                                                margin-right="2.5cm">
                                <fo:region-body margin-top="3cm"/>
                                <fo:region-before extent="3cm"/>
                                <fo:region-after extent="1.5cm"/>
                                </fo:simple-page-master>

                        </fo:layout-master-set>
                <!-- Ende: Definition der Seiten -->

                <!-- Definition des Seiteninhaltes -->
                <fo:page-sequence master-name="Einfache Seite">

                        <!-- Seitenmitte -->
                        <fo:flow flow-name="xsl-region-body">

                                <!-- Seiten-Titel -->
                                <fo:block font-size="18pt"
                                  font-family="sans-serif"
                                  line-height="24pt"
                                 space-after.optimum="15pt"
                                  background-color="blue"
                                  color="white"
                                  text-align="center"
                                  padding-top="3pt">
                                 Bestellung
                                </fo:block>
```

```
                                        <xsl:apply-templates/>

                                    <!-- Ende: Seitenmitte -->
                                    </fo:flow>

                                <!-- Ende: Definition des Seiteninhaltes -->
                                </fo:page-sequence>

                        <!-- Ende: FO INFORMATIONEN -->
                        </fo:root>

        </xsl:template>
        <!-- ENDE: MATCH: "Bestellung" -->

        <!-- MATCH: "Datum" -->
        <xsl:template match="Datum">

                <!-- Standard Text -->
        <fo:block font-size="15pt"
                  font-family="sans-serif"
                  line-height="15pt"
                  space-after.optimum="3pt"
                  text-align="justify">
        vom <xsl:value-of select="."/>
        </fo:block>

        </xsl:template>

        <!-- MATCH: "Kunde" -->
        <xsl:template match="Kunde">
                <!-- Standard Text -->
        <fo:block font-size="14pt"
                  font-family="sans-serif"
                  line-height="15pt"
                  space-after.optimum="15pt"
                  text-align="justify">
        Herr/Frau: <xsl:value-of select="."/>
        </fo:block>

        </xsl:template>


        <!-- MATCH: "Artikel" -->
        <xsl:template match="Artikel">
                <!-- Standard Text -->
        <fo:block font-size="12pt"
                  font-family="sans-serif"
                  line-height="15pt"
                  space-after.optimum="3pt"
                  text-align="justify">
        <xsl:number/> <xsl:value-of select="."/>
        </fo:block>

        </xsl:template>


</xsl:stylesheet>
```

Abbildung 3-9 **Ein XSL Stylesheet zur Umwandlung in einen FO-Baum**

Das Ergebnis dieser Umwandlung der XML-Datei in eine PDF-Datei sieht folgendermaßen aus:

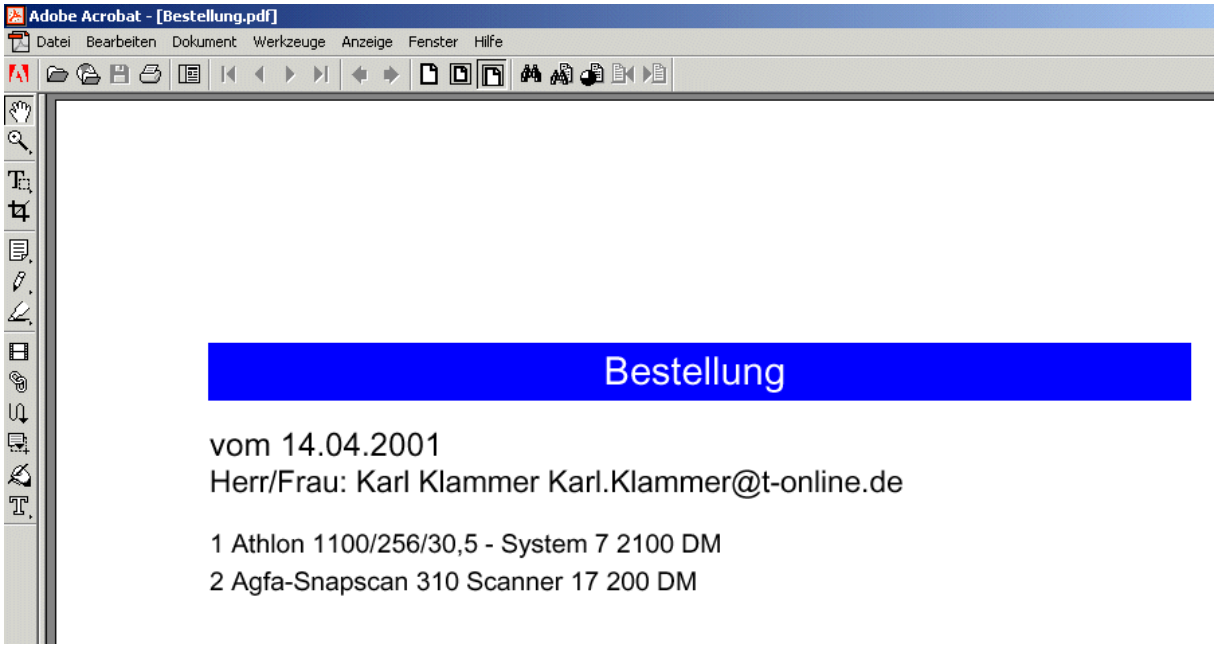

Abbildung 3-10 **Das Ergebnis der Umwandlung in das PDF-Format**

4 XML in der Praxis

In diesem Kapitel soll ein Querschnitt der interessantesten, am ehrgeizigsten verfolgten bzw. aussichtsreichsten XML-Projekte vorgestellt werden. Dazu zählen sowohl Open-Source-, als auch proprietäre Projekte in den verschiedensten Anwendungsbereichen.

4.1 Das Apache XML Projekt

Die Apache Software Foundation (ASF) hat 1999 ein Projekt ´aus der Taufe gehoben´, welches an XML-Entwicklungswerkzeugen und –Frameworks arbeitet, die kommerzielle Qualität bieten, aber Open-Source sind, um den Standard XML vor einem eventuellen Zugriff eines einzelnen Unternehmens zu bewahren. Hierzu werden der Apache-Entwicklergemeinde von Firmen wie IBM und Sun zum Teil Technologien zur Weiterentwicklung zur Verfügung gestellt. Sind diese Technologien entsprechend weit entwickelt, werden sie dem W3C zur Standardisierung vorgelegt.

Das Apache XML Projekt ist in sieben Subprojekte gegliedert, wobei sich jedes mit einem anderen Aspekt bzw. einer anderen Anwendung der Metasprache XML befasst:

- *Xerces* :

  Xerces ist ein XML-Parser für die Programmiersprachen Java, C++ und Perl, der folgende Standards unterstützt: DOM (Level 1 und Level 2), SAX (Version 2), XML Schema, XML Namespaces.

- *Xalan* :

  Xalan ist ein hochperformanter XSLT-Prozessor für Java und C++, der bisher die Transformation nach XML, Text und HTML beherrscht.

- *FOP :*

  FOP ist ein Java-basierter FO-Prozessor, der bis zum jetzigen Zeitpunkt nur die Transformation nach PDF ermöglicht.

- *Cocoon:*

  Cocoon stellt ein vollständig auf XML basiertes Publishing-Framework für mittlere bis komplexe Websites dar. Cocoon erlaubt die Erzeugung von XML-Dateien als Inhalt, die Verarbeitung der XML-Dateien und die Darstellung des Inhaltes mittels XSL-Stylesheets. Die Werkzeuge Xerces, Xalan und FOP werden hier intensiv genutzt.

- *Xang:*

  Im Gegensatz zu Cocoon stellt Xang ein einfaches XML-basiertes Publishing-Framework zur Verfügung, welches ausschließlich für einfache Websites konzipiert ist und JavaScript verwendet.

- *SOAP:*

  Eine Referenzimplementation des W3C Protokolls SOAP (siehe *Kapitel 4.2 RPC mit SOAP*).

- *Batik:*

  Batik ist ein Java-basiertes Werkzeug für Anwendungen, die Scalable Vector Graphics (SVG) für diverse Möglichkeiten (Anzeigen, Erzeugen und Verarbeiten des SVG-Formates) nutzen wollen. SVG-Parser, -Generator, -DOM-Implementation stellen die Hauptbestandteile von Batik dar.

- *Crimson:*

  Ein Java XML Parser entstanden aus dem Sun Project X Parser.

Bei Betrachtung dieser Subprojekte zeigt sich der allgemeine Trend, für die Verarbeitung von XML-Daten und XML-Dokumenten, die Programmiersprache Java zu verwenden. Java mit der Möglichkeit, portable Programme zu erstellen und XML mit der Möglichkeit, portable Daten zu gewährleisten ergänzen sich hier in perfekter Weise: Java besorgt die Programmlogik und XML übernimmt die Strukturierung der Daten – und beide sind an keine Plattform gebunden. Jon Bosak, der ´Vater der XML-Technologie´, meinte zu dieser Symbiose: “XML and Java technology are the yin and yang of cross-platform XML-technology“.

4.2 RPC mit SOAP

Als Spezialanwendung kann XML als ´verteiltes Protokoll´ fungieren, d.h. es wird als Datentransportmittel zwischen verteilten Applikationen verwendet. Das hierfür eingeführte XML-basierte SOAP (Simple Object Access Protocol) wird als der ´neue Star unter den Interoperabilitätsprotokollen´ bezeichnet und tritt in direkte Konkurrenz zu ´traditionellen´ RPC-Protokollen wie DCOM, DEC, IIOP und RMI. SOAP ist im Prinzip ein einfacher Paketierungs- und Routing-Mechanismus für XML-Daten auf Basis von etablierten Transportprotokollen wie HTTP, SMTP, usw. Der RPC wird in ein XML-Dokument, in diesem Kontext als SOAP-Envelope bezeichnet, verpackt und an den Empfänger versendet. Dieser extrahiert den RPC, ruft die entsprechende Methode der entfernten Instanz auf und verpackt die Antwort wiederum, um sie an den Aufrufenden zu senden.

Die Vorteile dieses Protokolls gegenüber dem Querschnitt[1] der ´traditionellen´ Protokolle:

- Keine Probleme mit Firewalls, da Kommunikation über HTTP
- Client und Server können in jeder Programmiersprache implementiert sein
- Plattformunabhängigkeit
- Keine Lizenzkosten; nicht proprietär
- Streaming über SAX möglich
- Unterschiedliche Serialisierungs-Versionen sind kompatibel
- Vom Menschen lesbar (bei Fehleranalyse, Systemtests)

Die Nachteile dieses Protokolls, welches zumindest im jetzigen Stadium sicher nicht für jedes beliebige Einsatzszenario geeignet ist:

- Geschwindigkeitsprobleme (bis Faktor 1:10), aufgrund Text, Parsen und Validieren
- Keine Integration in das OO-Modell der Applikation (Mapping notwendig!)
- Keine Transaktionslogik, da HTTP-basiert

[1] Auch CORBA ist programmiersprachenunabhängig.

4.3 E-Commerce mit BizTalk und ebXML

Zur Zeit bilden sich viele unterschiedliche Frameworks, welche den reibungslosen elektronischen Handel, den Austausch von Geschäftsnachrichten auf XML- bzw. SOAP-Basis und die problemlose EAI (Enterprise Application Integration) ermöglichen sollen. Die folgende Tabelle soll die wichtigsten dieser Frameworks auflisten:

Hersteller	Framework/ Sprache	Beschreibung
Ariba	cXML	Procurement-orientiert mit Schnittstellen zu anderen Herstellern
Commerce One	CBL	Definiert Geschäftsdokumente und Transaktionen
Microsoft	BizTalk	Framework mit Schemata und Repository für den Datentransport
Herstellerunabhängig	BMEcat	Austauschformat für Produktkataloge
Herstellerunabhängig	ebXML	Framework für B2B-Plattformen
Herstellerunabhängig	Rosettanet	Spezifiziert Prozesse im B2B-Austausch und legt Nachrichtenstandards fest

Tabelle 4-1 **B2B-Frameworks und -sprachen auf XML-Basis**

Im Moment scheint sich die Industrie auf zwei dieser Frameworks zu konzentrieren, denen die besten Chancen zum Standard gegeben werden: Microsofts BizTalk-Initiative und ebXML, eine herstellerunabhängige Initiative der UN und OASIS, der weltgrößten unabhängigen und gemeinnützigen Organisation. Beide werden von ´Schwergewichten´ wie SAP, Baan, Peoplesoft, J.D. Edwards, IBM und Sun unterstützt.

Die Idee dieser E-Commerce Frameworks für den Datenaustausch zwischen verschiedenen Applikationen sieht folgendermaßen aus: Es existiert ein öffentlicher Server, der als Sammelbecken für XML-Schemata (siehe *Kapitel 2.6.7 XSchema*) dient. Firmen können hier ihre XML-Schemata ablegen und verifizieren. Durch diese Veröffentlichung der Schemata ist es anderen Firmen nun möglich, mit ihnen Geschäftsnachrichten (Bestellungen, Produktinformationen, Preise, Zahlungen, usw.) auszutauschen. Es werden also nicht mehr die Schnittstellen einer Anwendung normiert und offengelegt, sondern das Format der zu bearbeitenden Daten. Die folgende Grafik soll dies verdeutlichen.

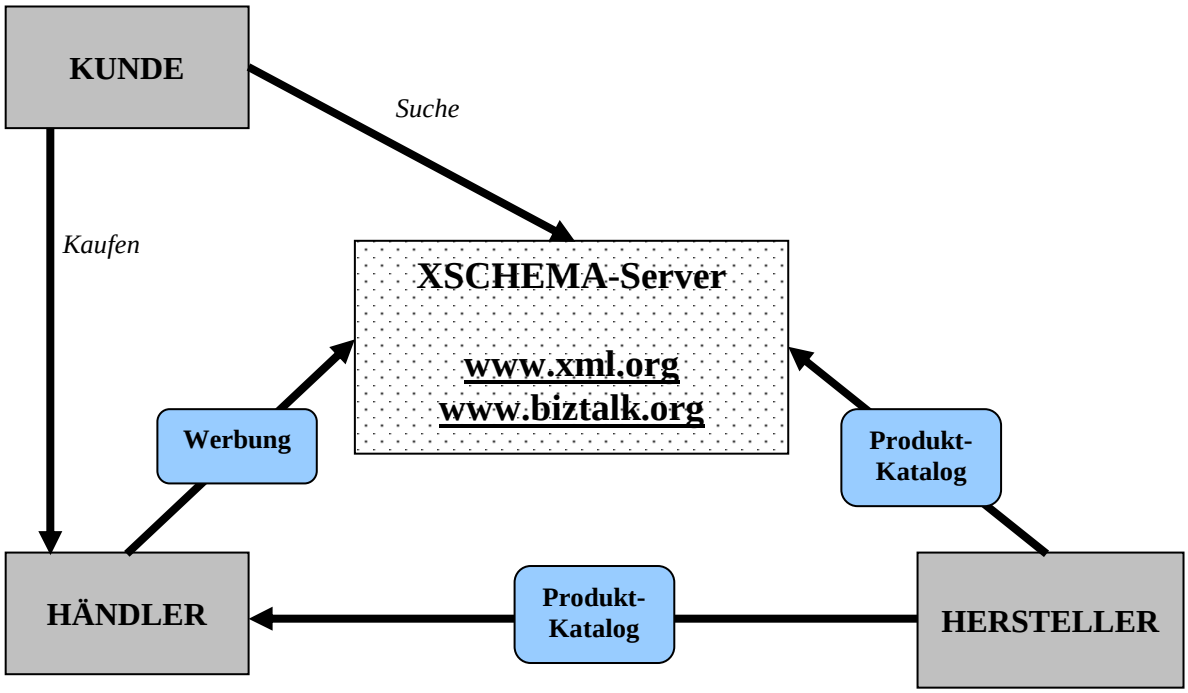

Abbildung 4-1 E-Commerce-Framework auf XML-Basis

“Damit sich über eine Million neuer Unternehmen am elektronischen Handel beteiligen können, muss deren Software lernen, eine einheitliche Sprache zu sprechen.“

Bill Gates, CEO, Microsoft

”ebXML enables a global electronic marketplace where enterprises of any size and in any geographical location can meet and conduct business with each other through the exchange of XML based messages.”

EbXML-Initiative, Homepage, “What is ebXML?”

4.4 XML-Persistenz mit Tamino

XML hat sich mittlerweile für die Datenbeschreibung und in Form von SOAP für den Datentransport etabliert. Mit Tamino, einem XML-basierten DBMS der Software AG, versucht es dies nun auch im Bereich der Datenspeicherung.

Der erste Ansatz des Mappings von XML-Daten in Tabellen eines RDBMS zeigte insbesondere bei der Verarbeitung von XML-Dokumenten im engeren Sinne (siehe *Kapitel 2.6.5 – XML - Dokumente oder Daten?*) gravierende Probleme. Alle gängigen Datenbank-Hersteller, darunter IBM, Oracle und Microsoft, bieten für ihre Produkte entsprechende Mapping-Aufsätze an, um ihr Datenbanksystem als ´XML enabled´ bezeichnen zu können. Allerdings handelt es sich hier in den meisten Fällen um niederperformante Ad-hoc-Lösungen, die XML-Dokumente unanalysiert als BLOBs ablegen, die nur durch Volltextsuche abfragbar sind. Eine andere Version ist die durch Tamino realisierte Möglichkeit die XML-Dokumente nativ zu speichern und entsprechende Abfragesprachen wie das standardisierte XPath zu verwenden. Ob sich diese Art der Datenbanksysteme durchsetzen kann, wird sich zeigen müssen. Es handelt sich um eine sehr junge Technik, die den seit Jahrzehnten kontinuierlich weiterentwickelten und verbesserten relationalen Datenbanksystemen gegenübersteht. Den gleichen Angriffspunkt verfolgen auch die objektorientierten Datenbanksysteme nun seit einigen Jahren ohne überragende Erfolge.

5 Zusammenfassung und Ausblick

Die Bedeutung des Internets zur Integration elektronischer Geschäftsprozesse aller Art, d.h. der innerbetrieblichen Vorgänge und der Kommunikation mit den Geschäftspartnern, wächst exponentiell. Während E-Business zur Zeit die Art, Geschäfte abzuwickeln, verändert, verändert XML das E-Business selbst. XML erlaubt erstmals die einfache und einheitliche Kommunikation in heterogenen Umgebungen wie dem Internet und überbrückt durch seine Plattformunabhängigkeit, freie Verfügbarkeit und Textbasiertheit alle Inkompatibilitäten. Durch den Einsatz der ´Partnertechnologie´ XSL ist weiterhin die mehrfache Verwendung einer Publikation möglich, d.h. eine Veröffentlichung wird ein Mal geschrieben und steht nun (durch die Verwendung von XSL-Stylesheets) auf allen verfügbaren Medien bereit. Da XML auf Unicode basiert, ist weiterhin die Internationalisierung der Datenbeschreibung, -übertragung und -speicherung kein Hindernis mehr. Der wichtigste Punkt ist aber die hohe Akzeptanz und breite Unterstützung dieser Technologie durch die Unternehmen. XML ist neben Java der Hoffnungsträger der IT.

Aufbauend auf der Basistechnologie entstehen zur Zeit eine Vielzahl von Anwendungen, die eine neue Art des E-Business und des Internets allgemein zur Folge haben werden. Die wichtigsten sind sicherlich SOAP, UDDI und WSDL, welche die Basis für den E-Commerce bilden sollen und in die neu vorgestellten Architekturen ´Sun ONE´ der Firma Sun und ´. NET´ der Firma Microsoft als fester Bestandteil einfließen, um die langverfolgte Idee der ´smart web services´ endlich Wahrheit werden zu lassen.

XML wird sich neben TCP/IP und HTTP als absoluter Standard durchsetzen und in absehbarer Zeit genauso selbstverständlich wie diese beiden Protokolle sein. Jede Middleware wird in Zukunft den Datenzugriff und –austausch über XML erledigen und jede Publikation wird als XML-Dokument vorliegen, um sie auf jedem Medium verbreiten zu können.

Zur Zeit sind noch nicht alle XML verwandten Technologien als Standard vom W3C verabschiedet. So befinden sich z.B. XSL FO, XLink, Xpointer und XPath noch im Entwicklungsstadium. Aufgrund der Bedeutung die die Metasprache XML heute schon hat, dürfte dies jedoch kein Hindernis darstellen, so dass in absehbarer Zeit das Fundament dieses neuen Standards geschaffen sein wird.

Abkürzungsverzeichnis

AML	Astronomical Markup Language
API	Application Programming Interface
ASF	Apache Software Foundation
B2B	Business-to-Business
B2C	Business-to-Customer
BLOB	Binary Large Object
BSML	Biosequence Markup Language
CDATA	Character Data
CERN	Conseil Européen pour la Recherche Nucléaire
CML	Chemical Markup Language
CORBA	Common Object Request Broker Architecture
CSS	Cascading Style Sheets
DBMS	Database Management System
DBVS	Datenbankverwaltungssystem
DCOM	Distributed Component Object Model
DOM	Document Object Model
DSSSL	Document Style and Semantics Specification Language
DTD	Document Type Definition
EAI	Enterprise Application Integration
EBNF	Erweiterte Backus Naur Form
ebXML	Electronic Business XML
EDI	Electronic Data Interchange
EIS	Enterprise Information System
ERPS	Enterprise Resource Planning System

GCA	Graphic Communications Association
GML	Generalized Markup Language
GUI	Graphical User Interface
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	HTTP über SSL
IIOP	Internet Inter-ORB Protocol
J2EE	Java 2 Enterprise Edition
JSP	JavaServer Pages
JVM	Java Virtual Machine
MathML	Mathematical Markup Language
MHTML	MIME Encapsulation of Aggregate HTML Documents
MIT	Massachusetts Institute of Technologie
OASIS	Organization for the Advancement of Structured Information Standards
ODBMS	Object Oriented Database Management System
OO	Objektorientierung
PCDATA	Parsable Character Data
PDA	Personal Digital Assistant
PDF	Portable Document Format
PI	Processing Instruction
RDBMS	Relational Database Management System
RDF	Resource Description Framework
RMI	Remote Method Invocation
RPC	Remote Procedure Call
RTF	Rich Text Format

SAX	Simple API for XML
SGML	Standard Gerneralized Markup Language
SMIL	Synchronized Multimedia Integration Language
SMS	Short Message Service
SMTP	Simple Message Transfer Protocol
SOAP	Simple Object Access Protocol
SQL	Standard Query Language
SSL	Secure Socket Layer
SVG	Scalable Vector Graphics
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
VoxML	Voice Extensible Markup Language
W3C	World Wide Web Consortium
WAP	Wireless Application Protocol
WML	Wireless Markup Language
WWW	World Wide Web
XHTML	XML Hypertext Markup Language
XLink	XML Link Language
XMI	XML Metadata Interchange Format
XML	Extensible Markup Language
XPath	XML Path Language
XPointer	XML Pointer Language
XQL	XML Query Language
XSL	Extensible Stylesheet Language
XSL FO	Extensible Stylesheet Language Formatting Objects
XSLT	Extensible Stylesheet Language Transformations

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[MERZ99] Merz, Michael (1999):
Electronic Commerce – Marktmodelle, Anwendungen und Technologien.
1. Aufl. 1999, Heidelberg, dpunkt Verlag

[iX1/00] Mandl, Peter, Nikolai Bauer (2000):
Rollenspiel – Chancen und Risiken der EJB-Komponententechnik.
In: iX, 1/2000, S.108ff

[JS3/00] Hettel, Jörg, Zink, Thomas (2000):
Enterprise-JavaBeans im Praxistest.
In: Java Spektrum, 3/2000, S.67ff

[OBJS1/00] Erbrich, Thomas (2000):
Integrierte E-Business-Lösungen mit “Windows DNA“, XML und “SAP R/3“.
In: Objekt Spektrum, 1/2000, S.74ff

[iX6/01a] Behme, Henning (2001):
Angelpunkt – Wo die Extensible Markup Language derzeit steht.
In: iX, 6/2001, S.52ff

[iX6/01b] Ziegler, Cai (2001):
Weltendämerung – XML und Datenbanken: Einblick in Tamino.
In: iX, 6/2001, S.56ff

[iX6/01c] Klever, Nik (2001):
Elementarteilchen – XML-Schema: objektorientierte Dokumenttyp-definitionen.
In: iX, 6/2001, S.62ff

[iX6/01d] Popper, Andreas (2001):
Daten eingewickelt– Dynamische Webseiten mit XML und SQL.
In: iX, 6/2001, S.67ff

[iX6/01e] Wegelin, Michael (2001):
.NET baut auf XML – Die Kommunikationsinfrastruktur der .NET-Server-Familie.
In: iX, 6/2001, S.67ff

[RZSEJB] Zacharias, Roger (2001):
EJB-Applikationsserver.
Seminararbeit
FH Gießen-Friedberg

[JM5/01a] Holubek, Andreas (2001):
Objekte aufbewahren – Persistenz zum Nulltarif: Kurz- und Langzeitpersistenz mit Java.
In: Java Magazin, 5/2001, S.16ff

[JM5/01b] Knuth, Michael (2001):
Serviceorientiert – Web Service Definition Language – was ist das?.
In: Java Magazin, 5/2001, S.91ff

[JM5/01c] Seemann, Michael (2001):
Fisch sucht Fahrrad? – Mit UDDI fit für B2B?.
In: Java Magazin, 5/2001, S.95ff

[JM4/01] Klein, Manuel (2001):
Mehr Stil für Java – Xalan – XSL-Prozessor für Java.
In: Java Magazin, 4/2001, S.108ff

[JM10/00] Großwendt, Volkmar (2000):
Präsentation im Web – Datenbank-Publishing mit XML.
In: Java Magazin, 10/2000, S.105ff

[JS4/00] Hranitzky, Norbert (2000):
XML-basiertes Content-Management.
In: Java Spektrum, 4/2000, S.56ff

[JM12/00a] Mathy, Frank (2000):
SOAP Opera – Verteilte Dienste mit SOAP unter Java.
In: Java Magazin, 12/2000, S.20ff

[JM12/00b] Reichel, Volker (2000):
Eingesponnen – Webpublishing mit Cocoon 1.8.
In: Java Magazin, 12/2000, S.91ff

[JS2/01] Frotscher, Thilo (2001):
Kommunikation mit dem Simple Object Access Protocol.
In: Java Spektrum, 2/2001, S.60ff

[JR11/00] Mannion, Mike (2000):
Rapid GUI Development for the Web: Exploiting Java, XML and XSLT.
In: Java Report, 11/2000, S.24ff

[JR10/99] Deadman, Richard (1999):
XML as a distributed application protocol.
In: Java Report, 10/1999, S.16ff

[OBJS6/00] Vasters, Clemens (2000):
SOAP – die Lösung aller Interoperabilitätsprobleme?.
In: OBJEKTspektrum, 6/2000, S.26ff

[JM3/99] Holubek, Andreas (1999):
Extravagant – XML und XSL in der Praxis anwenden.
In: Java Magazin, 3/1999, S.12ff

[JS1/00] Hranitzky, Norbert (2000):
XML- Der Stein der Weisen?.
In: Java Spektrum, 1/2000, S.49ff

[JM3/00a] Großwendt, Volkmar (2000):
Ausgezeichnet – XML – Eine Einführung.
In: Java Magazin, 1/2000, S.79ff

[JM3/00b] Großwendt, Volkmar (2000):
Reine Formsache – Elementstruktur von XML-Dokumenten definieren.
In: Java Magazin, 3/2000, S.81ff

[JM3/00c] Großwendt, Volkmar (2000):
DOMinikaner – DOM - Objektorientierte Dokumentenstruktur.
In: Java Magazin, 1/2000, S.94ff

[JBLUE] Sun Microsystems Inc.
J2EE Blueprints.
www-Dokumente vom 02.01.2000:
http://www.javasoft.com/j2ee/blueprints

[JAVAW1] Kadel, Rich:
Components for the rest of us.
www-Dokument vom 10.08.2000:
http://www.javaworld.com/javaone98/j1-98-ejb.html

[JBRAU] Brauer, Jürgen:
Die IT-Branche – Beschreibungssprachen HTML, XML ...
www-Dokument vom 04.04.2001:
http://home.t-online.de/home/juergen.brauer/seite03a.htm

[SAPT] SAP Deutschland
mySAP Technologie.
www-Dokument vom 23.05.2001:
http://www.sap-ag.de/germany/technology/index.asp

[JONE1] Sun Microsystems, Inc.
Sun Open Net Environment (SUN ONE).
www-Dokument vom 23.05.2001:
http://www.sun.com/software/sunone/

[MSNET] Microsoft Corporation

.NET Defined.
www-Dokument vom 23.05.2001:
http://www.microsoft.com/net/

[SWARM1] Phipps, Simon
Building the Swarm.
www-Dokument vom 25.05.2001:
http://java.sun.com/features/2001/04/swarm.html

[SWARM2] Seemann, Michael
Spürhunde im Web.
www-Dokument vom 25.05.2001:
http://entwickler.com/jm/ausgaben/2001/6/artikel/8/online.shtml

[W3C] World Wide Web Consortium
www-Dokument vom 27.05.2001:
http://www.w3.org

[APACHE] Apache XML Project
www-Dokument vom 21.05.2001:
http://xml.apache.org

[XCOM] XML.com
www-Dokument vom 21.05.2001:
http://www.xml.com