Technische Universität Ilmenau

Fakultät für Informatik und Automatisierung

Fachgebiet Prozessinformatik

Betreuer: Dipl.–Inf. Detlef Streitferdt

Projektarbeit

Sommersemester 2003 / Wintersemester 2003/2004

zum Thema

Anwendung von XSL(T) in OpenOffice.org Applikationen

Bearbeiter: Katharina Berg
Termin: 31. Oktober 2003

Inhaltsverzeichnis

1. Einleitung

1.1. Gegenstand der Arbeit

Die Extended Markup Language XML ist ein vom World Wide Web Consortium- W3C[1] entwickelter Standard zur Strukturierung von Daten und Dokumenten.
XML hat als offener Standard sowohl im Internet wie auch in Intranets wichtige Funktionen zur unabhängigen Darstellung von Daten in einem Browser. Zur Darstellung werden weitere Anweisungen benötigt, die unter anderem durch XSLT- Dokumente (Extensible Stylesheet Language Transformation) realisiert werden. Auf der anderen Seite lassen sich XML-Dokumente über XSLT- Anweisungen filtern, manipulieren, sortieren, berechnen und auch individuell in ihrem Aussehen verändern. Damit ergibt sich eine Vielzahl von Anwendungsmöglichkeiten, in denen XML- Daten einmalig vorliegen und nach Wunsch in andere Datenformate übertragen werden. Diese Möglichkeiten sind noch lange nicht ausgeschöpft. Ihre Entwicklung wird aber intensiv vorangetrieben. Ein Beispiel soll das hier thematisierte Openoffice.org[2] sein.
Ein besonderer Aspekt von XML ist die Fähigkeit, eigene beschreibende Elemente in Document Type Definitions oder Schemata definieren zu können, ihnen zuzuordnen, welche Informationen sie enthalten sollen und festzulegen, wie sie später dargestellt werden sollen.[3]

Durch die dargestellte Flexibilität liegt es nahe, XML in Zukunft generell für die Speicherung von Daten zu nutzen. Gerade in der Hinsicht, dass proprietäre Systeme immer mehr ins Hintertreffen gelangen und inkompatible Datenformate auf dem globalen Markt aufeinander treffen, wird es nötig, offene Standards zu etablieren und der wachsenden Community zugänglich zu machen.[4]
Dies geschieht unter anderem schon heute in Openoffice.org[5].
Große Organisationen wie das W3- Consortium haben es sich zur Aufgabe gemacht, diese Entwicklungen voranzutreiben und Standards zu veröffentlichen. Das W3C beschreibt die Bedeutung von XML: “XML is also playing an increasingly important role in the exchange of a wide variety of data on the Web and elsewhere.”[6]

1.2. Zielsetzung

Ausgehend von der möglichen Leistungsvielfalt des XSLT- Standards soll hier, als ein Beispiel der vielen möglichen XML- Einsatzgebiete, die Anwendung in Openoffice.org Applikationen untersucht werden. Es wird gezeigt, wie mit Hilfe von XSLT- Anweisungen Openoffice.org

[1] Das World Wide Web Consortium W3C ist die ofizielle Organisation zur Förderung und Entwicklung von Standards, die miteinander arbeiten können (Spezifikationen, Richtlinien, Software und Werkzeuge), vgl. [Shepherd 2002] S. 30, mehr dazu unter http://www.w3c.org

[2] Mehr dazu unter http://www.openoffice.org

[3] Vgl. [UseGroup 2001] „Was ist XSLT?“

[4] Vgl. [OO.o deutsch 2003] Startseite

[5] OpenOffice.org ist ein Opensource Projekt, welches im Kapitel 3 ausführlich behandelt wird

[6] Vgl. [W3C Introduction 2003]

Dokumente abgeändert und bestimmte Daten herausgefiltert werden können. Dafür werden Beispieldaten des „Digitalen Videoprojektes“ DVP[7] verwendet. Einzelne Daten sollen aus dem ursprünglichen XML- Dokument gefiltert und in die Grundstruktur von Openoffice.org überführt werden.

1.3. Aufbau der Arbeit

Im ersten Teil wird nach einer kurzen Entwicklungsgeschichte von XML erläutert, wie sich die Transformationssprache XSL(T) anwenden lässt und in welchen Bereichen sie denkbar ist. Dazu werden im zweiten Teil Mechanismen zur Ausgabe verschiedener Formate ausgehend aus einem XML- Dokument wie zum Beispiel von PDF- Dokumenten, Datenbanken und Bildern vorgestellt.
Im darauf folgenden Teil wird als konkretes Anwendungsbeispiel für XML das Opensource-Projekt Openoffice.org und seine wesentlichen Ideen vorgestellt. Dazu wird anschließend auf die grundlegende XML Struktur der Dokumente eingegangen. Zudem soll eine Reihe von möglichen XML- Parsern und ihre Funktionsweise vorgestellt werden. Im 4. Kapitel erfolgt nun die praktische Anwendung. Anhand von Daten des DVP wird die Umsetzung der XML- basierenden Daten durch XSLT- Templates näher erläutert. Dabei müssen bestimmte Daten nach Kriterien gefiltert und sortiert ausgegeben werden. Die heraus gelösten Daten werden dann in die Struktur eines Openoffice.org Dokumentes hineingearbeitet und somit im Zieldokument dargestellt.

Beispiele der XML- Schreibweise oder andere Quelltexte werden durch eine andere Schriftart deutlich gemacht:

```
Quelltext.
```

Alle Beispiele wurden mit Hilfe des XML- Editors XML Spy von Altova geschrieben[8].
Im Anschluss folgt eine Zusammenfassung und ein Ausblick der geplanten Methoden und Vorgehensweise von Openoffice.org.

Aus namensrechtlichen Gründen ist die richtige Bezeichnung der allgemein bekannten Openoffice Suite Openoffice.org[9].

[7] Mehr dazu unter [Streitferdt DVP 2003]

[8] Mehr dazu unter http://www.altova.com

[9] Vgl. [OO.o 2003] FAQs

2. Einführung in XML

2.1. Entwicklungsgeschichte

2.1.1. Industriestandard SGML

Ideen zur Strukturierung von Daten gibt es bereits seit längerer Zeit. So wurde die Standard Generalized Markup Language SGML schon 1986 in der ISO 8879 standardisiert. Sie ist eine Standard- Metasprache für die Grenzen und Inhalte von Textauszeichnungssystemen mit einem Definitionskatalog von über 500 Seiten.[10] Somit ist sie so monumental, dass sie heute hauptsächlich als Industriestandard für sehr große und komplexe Probleme angewendet wird.

2.1.2. HTML und Internet

Mir dem wachsenden Bekanntheitsgrad des Internets und damit auch dem immer größeren Wunsch Daten darzustellen, wurde 1989 die Hypertext Markup Language von Tim Berners Lee verabschiedet.[11] HTML war nunmehr eine Auszeichnungssprache, von SGML abgeleitet. Mit HTML ließen sich auf einfache Weise in einem Editor Anweisungen schreiben, wie die Daten in einem Browser darzustellen sind. Der Internetboom nahm damit seinen Lauf. Allerdings hat die HTML- Schreibweise zwei entscheidende Nachteile: Zum einen stellt jeder Browser den Inhalt anders dar. Damit ist die universelle Verwendbarkeit der in HTML formatierten Informationen wieder eingeschränkt[12]. Zum anderen geht der semantische Informationsgehalt der Daten verloren.

2.1.3. Semantik durch XML

Um die Probleme von HTML zu beseitigen, kam 1998 ein neuer Standard auf: XML. Auch XML ist eine sehr agile Ableitung der SGML. Sie beschränkt sich auf einen Umfang von 33 Seiten[13] und wurde daher für den alltäglichen Gebrauch, und damit kleineren Problemen, konzipiert. [14] XML wird inzwischen als eine neue Form von “Web- Sprache” bezeichnet, denn die Sprache dient als Vorlage weitere Auszeichnungssprachen wie zum Beispiel CML (Chemical Markup Language), WML (Wireless Markup Language) oder SMIL (Synchronized Multimedia Integration Language).
Die Abbildung 1 soll die Verhältnisse der Meta- und Auszeichnungssprachen verdeutlichen:

[10] Vgl. [Pott u.a. 2000] S. 34

[11] Durch Tim Berners- Lee, vgl. [W3C History 1998]

[12] Vgl. [Jung Background 2002] S. 3

[13] Vgl. [Pott u.a. 2000] S. 34

[14] Vgl. [Brosche SGML- XML 2002] S. 8

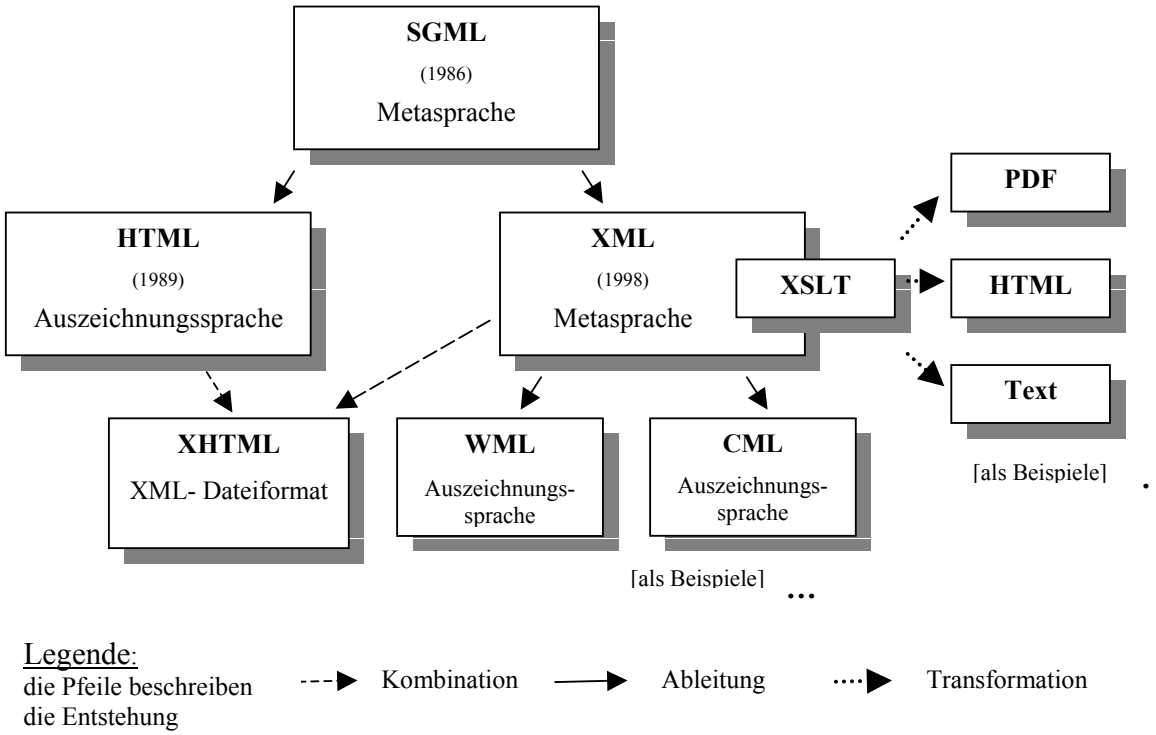

Abbildung 1: Sprachverwandtschaften[15]

Der Vorteil von XML basiert darauf, dass der Nutzer selbst definieren kann, wie seine Daten strukturiert werden. Damit lassen sich in selbst erstellten Elementen die semantischen Informationen speichern und gehen nicht mehr, wie in HTML üblich, verloren. Erst in zusätzlichen Dokumenten wird die physische Ausrichtung festgelegt. Das XML- Prinzip in Abbildung 2:

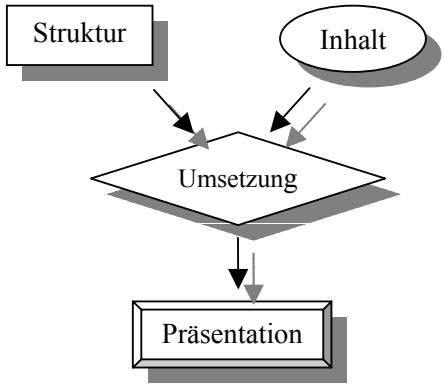

Abbildung 2: XML- Prinzip

[15] Jahreszahlen sind [Pott u.a. 2000] S. 33 entnommen

Ergänzend zu dem Informationsnutzen von XML über das Internet nimmt die Bedeutung von XML in der Wirtschaft exponential zu. XML dient als system-, sprachen- und netzübergreifender Datenhaltungsstandard, als Grundlage für semantisches Dokumentenmanagement und als Datenstruktur in Datenbanken. Auch wenn es hier noch Performanceprobleme gibt, sind einige Ansätze in Diskussion.[16] Grundlage der XML- Verwendung ist die Einigkeit der Handelspartner über die definierten Auszeichnungselemente, um Daten korrekt interpretieren und weiterverwenden zu können. XML kann somit als Werkzeugkasten für die Gestaltung interoperabler Lösungen gesehen werden. Gerade in Verbindung mit dem in der Wirtschaft geläufigen Electronical Data Interchange (EDI) wird durch XML der Datenaustausch erleichtert.[17]
Zudem ist durch die Kombination von XML mit HTML nach dem W3C „eine neue Familie bestehender und zukünftiger Dokumenttypen und Module, die HTML 4 reproduzieren, unterteilen und erweitern“ möglich und damit „der nächste Schritt in der Weiterentwicklung des Internets. Indem Inhaltsentwickler heute auf XHTML umsteigen, können sie in die XML- Welt mit allen dazugehörigen Vorteilen einsteigen, während sie sich auf die Abwärts- wie auch die zukünftige Kompatibilität ihrer Inhalte verlassen können.“[18]

2.1.4. Anwendungen und XSL(T)

XSL(T) erweitert die Anwendungspalette der XML- Welt. Mit Hilfe der Transformationssprache ist es möglich, XML- Daten in andere Formate umzuwandeln, sie zu berechen, zu sortieren oder zu filtern. Daten, die in einem XML- Dokument hinterlegt sind, können mittels XSL(T)-Anweisungen eine neue Form bekommen oder in andere Formate übertragen werden. Mit XML lassen sich ohne großen programmiertechnischen Aufwand anwendungsbezogen inhaltliche Strukturen von Dokumenten unterschiedlichster Art beschreiben.
Individuelle Änderungen an Internetseiten sind damit leichter realisierbar oder können je nach Kundengruppe und deren Interessen getrennt werden. Auch vorliegende Kataloge können in verschiedenen Formaten angeboten werden.
Bei den sich bietenden Anwendungsmöglichkeiten ist XML im Electronic Business und in Datenbankanwendungen nicht mehr wegzudenken.

2.2. XML- zugehörige Daten

Um ein XML- Dokument anzeigen zu können, werden zusätzliche Daten benötigt. Diese beschreiben, wie das Dokument im Browser darzustellen ist, oder welche Daten in welcher Form nach einer Transformation darzustellen sind.
Der Struktur- und der Inhaltsteil werden also wie im oben beschriebenen XML- Prinzip getrennt, können aber auch in einer einzelnen Datei gespeichert werden, der Separationsgedanke geht dadurch nicht verloren. Für die Wiederverwendbarkeit oder für größere Projekte ist es aber vorteilhafter, einzelne externe Dateien zu schreiben. Abbildung 3 zeigt eine Übersicht der möglichen externen Dateien:

[16] Vgl. [Jung Background 2002] S. 9- 12

[17] Mehr dazu unter http://www.ecin.de/edi/xml/

[18] Mehr dazu unter http://www.websitedev.de/xhtml/xhtml1/#xhtml

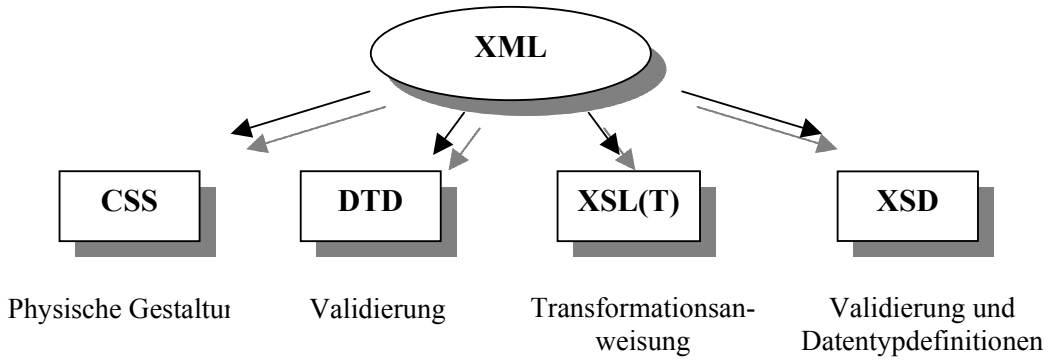

Abbildung 3: Einflussdateien auf ein XML- Dokument

Ein XML- Dokument ist, wenn es nach den Formvorschriften des XML- Standards formuliert wurde, wohlgeformt (wellformed). Dies gilt auch für alle anderen XML- Dokumente im weiteren Sinne, wie Schemata und XSL(T)- Dokumente.

2.2.1. CSS

Die einfachste und schnellste Lösung zur physischen Anordnung von XML- Daten sind die Cascading Style Sheets (CSS). Hier kann für jedes Tag eine entsprechende Regel erfolgen, wie Text zwischen den beschreibenden Elementen später im Browser darzustellen ist.
Möchte man zum Beispiel das Element <titel> groß in Fettschrift darstellen, wird das Tag folgendermaßen deklariert:

```
titel {font-size: 48pt; font-weight: bold;}
```

2.2.2. DTD

Document Type Definitions (DTD) sind Dateien, die beschreiben, welche Elemente mit welchen Attributen an welchen Stellen eines bestimmten XML- Formats erlaubt sind. Ebenso wird die Beziehung unter den Elementen, und welche Daten sie enthalten dürfen und welche nicht, geregelt. Alles was nicht in einer DTD definiert ist, ist folglich verboten. Dadurch wird das XML- Dokument gültig. Da die Korrektheit der Tags ganz erheblichen Einfluss darauf hat, ob die Dateien überhaupt angezeigt werden können, gibt es für fast jedes XML- Format ein DTD oder Schema.[19]
In DTDs werden auch Entities festgelegt. Das sind häufig vorkommende Textstücke oder Wörter, die in einer Abkürzung hinterlegt werden können. Im XML- Dokument muss somit nur das definierte Kürzel aufgerufen werden.
Ein XML- Dokument, welches konform mit den strengen Regeln einer DTD geht, ist somit gültig (valid).
Ein Ausschnitt einer DTD soll den Aufbau verdeutlichen:

```
<!DOCTYPE Firma [
<!ELEMENT Firma (Firmenname, Abteilung*|Person+)>
<!ELEMENT Firmenname (#PCDATA)>
```

[19] Vgl. [UseGroup 2001] „Was sind DTDs?“

```
<!ELEMENT Abteilung (Abteilungsname, Person+)>
<!ELEMENT Abteilungsname (#PCDATA)>
…
]>
```

Dieser Quelltext definiert das Wurzelelement Firma. Das Wurzelelement soll die Elemente „Firmenname“, 0 bis unendlich viele „Abteilungen“, oder mindestens eine „Person“ enthalten. Das Element „Firmenname“ ist vom Typ parsed character data (syntaktisch analysierte Zeichendaten). Das bedeutet, dass alle Zeichendaten, die keinem Element entsprechen, zwischen den Tags stehen können.
Das Element „Abteilung“ ist so definiert, das es ein Element „Abteilungsname“ und mindestes einmal das Element „Person“ enthält.[20]

2.2.3. XSD

Ein Schema (abgekürzt XSD) beschreibt genau wie eine DTD die Gültigkeit eines Dokumentes. Ein Schema tut dies aber auf eine datenorientierte Weise. Entscheidet man sich für die Verwendung von Schemata, kann man damit Datentypen und komplexe Strukturen definieren oder auch XSLT- Anweisungen schreiben, um andere Schemata zu manipulieren, da sie selbst XML- Dokumente sind. Auch hier wird ein XML- Dokument bei Befolgen der XSD- Regeln valid.[21]
Durch die frühzeitige Überprüfung der Gültigkeit von Dokumenten können Fehler schneller eliminiert oder vermieden werden.
Beispielsweise kann man definieren, dass das Attribut <postleitzahl> nur 5-stellige Zahlen zwischen 0 und 9 enthält.[22]

```
attribute name=“postleitzahl“>
…
<pattern value=“[0-9]{5}/>
```

2.2.4. XSL(T)

Daneben können XSL(T)- Dokuments geschrieben werden, die vorschreiben, wie die Daten für ein spezielles Format gerendert werden sollen.
Definiert werden Stilregeln und Konstruktionsregeln, die jeweils ein bestimmtes Muster ausdrücken. Werden entsprechend passende Stellen im XML- Dokument gefunden, wird dieses Muster auf sie angewendet.
Beispielsweise sucht folgendes Template den aktuellen Knoten, der durch das Tag <suche></suche> umschlossen wird, und gibt es als Überschrift in einer HTML- Seite aus:[23]

```
<xsl:template match=“suche“>
  <html>
   <body>
    <h1>
```

[20] Vgl. [Stein u.a. 2001] S. 77-79

[21] Vgl. [Tidwell 2002] S. 10- 11

[22] Vgl. [UseGroup 2001] „Was sind Schemata?“

[23] Vgl. [Tidwell 2002] S. 24- 25

```
    <xsl:value-of select=”.”/>
   </h1>
  </body>
 </html>
</xsl:template>
```

2.3. Möglichkeiten von XSL(T)

XSLT ist der Transformationsbereich von XSL. So lässt sich zum Beispiel ein XML- Dokument in wohlgeformten HTML- Code transformieren. Zusätzlich zu XSLT enthält XSL ein XML-Vokabular, um eine Formatierung zu spezifizieren. Dies geschieht mit Hilfe der Formatting Objects (XSL- FO). XSL analysiert das Layout eines XML- Dokuments, um unter Benutzung von XSLT zu beschreiben, wie das Dokument in ein anderes XML- Dokument, welches das Formatierungsvokabular benutzt, transformiert wird.
Ein XSL(T)- Dokument enthält prinzipiell Deklarationen über Version, Namensraum, verwendeter Zeichensatz und Format des Zieldokumentes.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
3  <xsl:output method="html"/>
4   <xsl:template match="Zitat">
5    <html>
6     <head><title></title></head>
7      <body>
8           <b><xsl:value-of select="@ID"></xsl:value-of>.
9          <xsl:value-of select="Autor"></xsl:value-of></b>:
10         <xsl:value-of select="O-Ton"></xsl:value-of><br/>
11    </body>
12   </html>
13  </xsl:template>
14</xsl:stylesheet>
```


Abbildung 4: Schematischer Aufbau eines XSL(T) Dokumentes

Die erste Zeile enthält eine Processing Instruction[24], in der der Parser über Version und Zeichencode informiert wird. In diesem Fall soll er die aktuelle 1.0 Version von XML und den komprimierten UTF-8 Unicode verwenden, mit dem XML grundsätzlich arbeitet. Er erkennt damit die benötigten Zeichen. Dokumente mit dem UTF- 8 Zeichensatz sind allerdings etwa doppelt so groß, da Unicode zwei Byte für jedes einzelne Zeichen verwendet.
Jedes Unicode Zeichen wird durch eine Zahl zwischen 0 und 65535 repräsentiert.[25]
Die nächste Zeile 2 des XML- Dokuments deklariert die Version von XSL(T) und die zu verwendeten Namensräume. Diese werden benötigt, wenn Elemente die gleiche Bezeichnung bekommen sollen, aber etwas anderes meinen. So soll die Eineindeutigkeit gesichert werden. In

[24] Processing Instructions sind Anweisungen an den Prozessor für die Durchführung der Datenverarbeitung

[25] Vgl. [Stein u.a. 2001] S. 60-61

diesem Fall entsprechen Elemente mit vorgestelltem <xsl:...> dem offiziellen XSL(T)-Namensraum des W3C.[26]
Zeile 3 befasst sich mit dem Format des Zieldokumentes, in diesem Fall HTML. Anschließend wird im Quelldokument jedes Vorkommen des Musters „Zitat“ gesucht und verschiedene Anweisungen dafür ausgeführt. Hier werden entsprechende Knoten in dem Muster analysiert und von HTML- Tags eingebettet ausgegeben.
Am Ende des Dokumentes muss regelkonform das Stylesheet wieder geschlossen werden.

XSLT ist auch entworfen worden, um unabhängig von XSL verwendet zu werden. Jedoch ist XSLT nicht als vollständig universelle XML- Transformationssprache gedacht. Vielmehr ist sie primär für die Arten von Transformationen entwickelt worden, die benötigt werden, wenn XSLT als Teil von XSL benutzt wird.[27]
Mittels XSLT kann man nicht nur Darstellungsanweisungen definieren, sondern auch Schleifen und Berechnungen wie zum Beispiel Verzweigungs- und Steuerelemente realisieren. Dafür werden reservierte Elemente wie <xsl:if>, <xsl:choose>, <xsl:when> und <xsl:for- each> verwendet[28]
Mit Hilfe von XSLT lassen sich so nicht nur Formatbefehle für XML- Dokumente schreiben und HTML Seiten generieren. Man kann ebenso aus den strukturiert vorliegenden Daten eines XML-Dokuments ein PDF, ein Javaprogramm, eine Virtual Reality Modeling Language Datei (VRML) oder eine Scalable Vector Graphics Datei (SVG) entwerfen.
Besondere Bedeutung kommt XSLT in Verbindung mit Datenbanken zu. Das können relationale, aber auch objektorientierte Datenbanken sein. Andererseits kann auch ein XML- Dokument weiter modifiziert werden, indem Daten ausgewählt und weiterverarbeitet und schließlich in einem neuen XML- Dokument gespeichert werden. Das können zum Beispiel Filterfunktionen, Berechnungen oder auch Sortierungen für unterschiedliche Interessengruppen sein. Besonders wichtig wird das, wenn XML- Daten in anderen Applikationen weiterverwendet werden sollen.

[26] Vgl. [Stein ua. 2001] S. 63

[27] Vgl. [W3C XSLT deutsch 2002]

[28] Vgl. [Tidwell 2002] S. 68- 73

3. Openoffice.org

3.1. Vorstellung des Openoffice.org Projektes

Openoffice.org ist ein Opensource- Projekt, welches momentan große Anerkennung innerhalb der Open Source Gemeinde erfährt. Das Projekt ist unter anderem von Andy Hertzfeld, Miguel de Icaza, Tim O'Reilly und Brian Behlendorf öffentlich gutgeheißen worden. Gegründet wurde Openoffice.org von der Sun Microsystems Inc., indem Sun die Technologie für die populäre StarOffice Productivity Suite freigegeben hat. Die Entwicklung der Software nunmehr selbst liegt aber in der Hand der Openoffice.org Community. Das Ziel dieses Projektes ist es, eine international führende Office-Suite auf Basis eines frei zugänglichen Quellcodes zu entwickeln. Dieser soll als gemeinsame Code- Basis für zukünftige Open- Source- Applikationen für die weltweite Entwickler-Gemeinde dienen.

Die Ziele des Openoffice.org Projekts werden ausführlich auf der Homepage beschrieben. Die Evolution des Basiscodes soll Openoffice.org in neue und innovative Richtungen führen und neue Märkte öffnen, indem es auf allen wichtigen Plattformen funktionsfähig sein wird und durch transparente Schnittstellen und dem XML- basierenden Datenformat Zugang zu Funktionen und Daten ermöglicht. Zudem soll das erweiterbare, auf XML basierende Datei-Format und sprachunabhängige Komponenten, Application Programming Interfaces (API), etabliert werden und Innovationen für die nächste Generation offener Netzwerk-Systeme ermöglichen. Gesteigerte Kompatibilität und Austauschbarkeit, resultierend aus offenen, sprachunabhängigen APIs, dem offenen, auf XML basierenden Datenformat, und der Einbindung öffentlicher Anregungen, uneingeschränkte Portierungen und Sprachanpassungen sind einige wichtige Ziele für die Zukunft. Die APIs sollen es erlauben, die Suite in Form separater Programme oder auch eingebettet in andere Anwendungen zu verwenden.

Die aktuelle Version der OpenOffice.org- Suite ist die 1.0.3.1, sie steht unter http://de.openoffice.org/about-downloads.html zum Download bereit.
Auf den offiziellen Openoffice.org Seiten im Internet werden der Entwicklergemeinde der Quellcode, Projektinformationen, Diskussionsforum und zu erledigende Aufgaben vorgestellt, damit sich jeder an der Entwicklung beteiligen kann. Selbst Dokumentenvorlagen etc. werden gesammelt und veröffentlicht.
Daneben können zusätzliche Pakete herunterladen werden, zum Beispiel für Silbentrennung, Rechtschreibprüfung oder Anleitungen (how-tos).
OpenOffice.org ist ein komplettes Office-Paket mit allen Funktionen, die man von einem solchen erwartet. Es enthält eine Textverarbeitung (Writer), eine Tabellenkalkulation (Calc) mit der Möglichkeit Diagramme zu erstellen, ein Präsentations-Programm (Impress), ein Zeichenmodul (Draw) sowie einen Formeleditor und einem Dateiformat-Wandler (inklusive der Formate für Microsoft Office).
Allerdings ist Openoffice.org nicht gleichbedeutend mit dem bekannteren StarOffice, da Openoffice.org nicht den kompletten Quellcode beinhaltet. Da Sun Lizenzen an Dritte zahlen muss, gehören so zum Beispiel die Rechtschreibprüfung, die Hilfefunktionen oder bestimmte Schriftarten nicht dazu. Der OpenOffice.org Quellcode beinhaltet von Anfang an die Technik, die Sun Microsystems für die zukünftigen StarOffice- Versionen vorgesehen hat. Der Quellcode ist in C++ geschrieben und liefert sprachübergreifende und skriptierbare Funktionalität, inklusive

auf Java basierende APIs. Diese Quellcode-Technologie läutet eine neue Phase der Entwicklung ein, die es erlaubt, die Anwendungen sowohl einzeln als auch eingebettet in andere Applikationen zu nutzen. Etliche andere Features sind ebenfalls vorhanden, etwa die auf XML basierenden Dateiformate.
Im Mai 2002 wurde dann die erste offizielle Version freigegeben: OpenOffice.org 1.0 war geboren.[29]

3.2. Aufbau von Openoffice.org Dokumenten

3.2.1. Openoffice.org- Packages[30]

OpenOffice.org beinhaltet diverse Anwendungen von Textverarbeitung über Tabellenkalkulation. Alle diese Applikationen nutzen auf XML basierende Datei-Formate. Alle, mit Ausnahme von Math, dem Formeleditor, nutzen das gleiche Format, das in der veröffentlichten Spezifikation definiert wurde. Die Math Komponente nutzt zwar die gleiche Packstruktur und das Packformat, verwendet aber MathML, eine von XML abgeleitete Auszeichnungssprache für die spezielleren Zwecke in der Mathematik, innerhalb der gepackten Datei.
XML wird auch in anderen OpenOffice.org Dateien, wie etwa den Konfigurationsdateien, genutzt.
Die standardmäßigen Dateiendungen der auf XML basierenden OpenOffice.org- Dokumente lauten:

Textverarbeitung	sxw
Tabellenkalkulation	sxc
Draw (Zeichenprogramm)	sxd
Impress (Präsentation)	sxi
Math (Formel-Editor)	sxm
Textverarbeitung Global Dokument	sxg

Tabelle 1: Dateiendungen in Openoffice.org

Jedes Openoffice.org Dokument ist ein Bündel an gepackten Dateien. In diesem Package befinden sich neben den XML Daten auch Binärdaten, wie etwa Bilder. Die Dateien werden im weit verbreiteten ZIP-Format gepackt. Das Package erhält zusätzlich eine XML- Datei, die den Inhalt des Packages beschreibt und auch weitere Informationen wie zum Beispiel eine Verschlüsselungsmethode beinhalten kann. Alle Dokumente bis auf die Datei meta.xml sind somit komprimiert. Nur meta.xml bildet dabei eine Ausnahme, da so schneller Metadaten gesucht oder extrahiert werden können.[31]

[29] Vgl. [OO.o deutsch 2003] FAQs

[30] Die Angaben zur Speicherung der Packages wurden [OO.o deutsch 2003] FAQs und [Kaldewey 2003] Beiträge entnommen

Wird ein Openoffice.org Dokument mit einem Packprogramm geöffnet, sieht man die einzelnen Dateien des Packages:

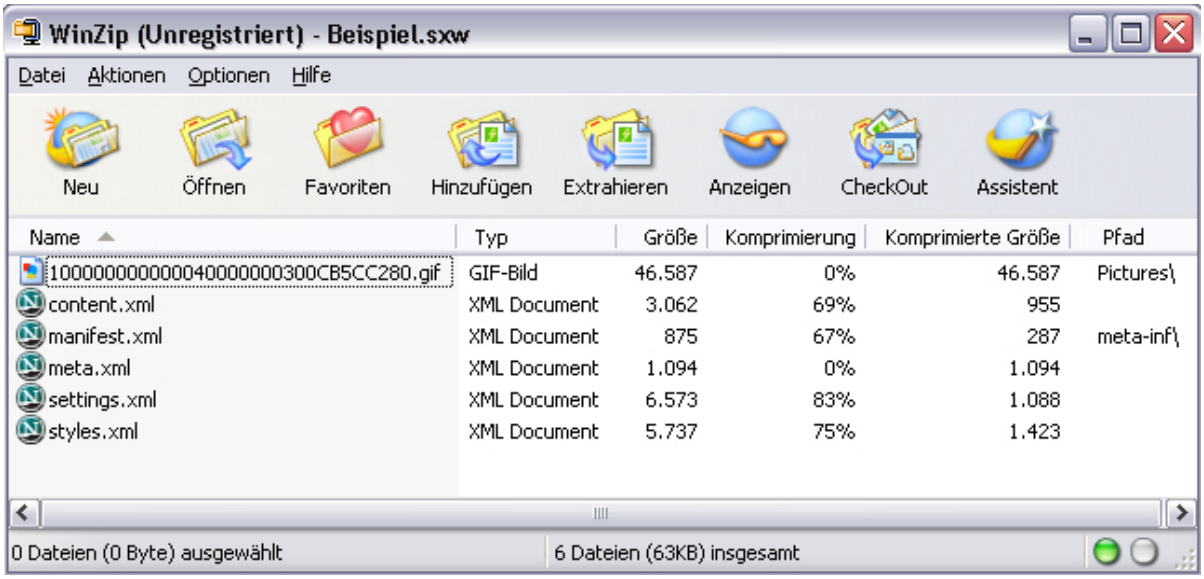

Abbildung 5: Geöffnetes Package

Jede dieser Dateien beinhaltet spezielle Informationen über das Dokument. Die folgende Tabelle fasst diese zusammen:

meta.xml	Informationen über das Dokument (Autor, Zeitpunkt der letzten Speicherung etc.)
styles.xml	Formate, die in diesem Dokument genutzt werden.
content.xml	Hauptinhalt des Dokumentes (Text, Tabellen, graphische Elemente)
settings.xml	Einstellungen, die das Dokument und seine Betrachtung betreffen (etwa Darstellungsebene und der ausgewählte Drucker); diese sind in der Regel abhängig von der jeweiligen Applikation
META-INF/manifest.xml	liefert zusätzliche Informationen über die anderen Dateien (etwa den MIME Typ oder die Verschlüsselungsmethode)
Pictures/	Verzeichnis, welches die Bilder enthält (in ihren ursprünglichen, binären Formaten)
Dialogs/	Verzeichnis, welches die Dialoge enthält, die von den Dokumentenmakros gebraucht werden
Basic/	Verzeichnis, welches die Makros (diese verwenden StarBasic) enthält

Obj.../	Verzeichnis, welches die eingebetteten Objekte, etwa Diagramme, enthält, jedes Verzeichnis enthält genau ein Objekt in seinem ursprünglichen Format. Für OpenOffice.org Objekte ist dies die XML Entsprechung. Andere Objekte liegen standardmäßig in einem binären Format vor.

Tabelle 2: XML- Daten eines Openoffice.org Dokumentes[32]

Diese Struktur von Openoffice.org Dokumenten ist jedoch nur eine Lösungsvariante.
Momentan gibt es nur wenige Ansätze, eigene Attribute, die nicht in der Openoffice.org DTD hinterlegt sind, in Dokumente mit einzufügen. In der Textverarbeitung ist es momentan möglich, eigenen Inhalt mit den <style:properties>- Elementen zu verknüpfen, alles andere wird von den internen Importfiltern entfernt. Solche selbst definierten Elemente können zum Beispiel Formatvorlagen für bestimmte Textpassagen sein. In späteren Versionen ist es jedoch geplant, eigenen Inhalt hinzufügen zu lassen. Ein anderer Ansatz verfolgt den Im- und Export von XML-Daten über die SAX- Schnittstelle[33]. Zudem soll es bald möglich sein, über die XMerge-Schnittstelle DocBook- Informationen austauschen zu können[34].
Zum anderem ist geplant, flache XML- Strukturen, also ohne Komprimierung, zu ermöglichen.
Alle diese Ansätze befinden sich aber noch im Experiment-Status.

3.2.2. Internes XML- Datenformat[35]

Die Openoffice.org Suite enthält eine Reihe von Document Type Definitions. Für die Dokumente selbst, außer Math- Dokumente, werden jedoch nur 2 DTDs referenziert: Alle XML- Dateien eines Packages beziehen sich auf die office.dtd, die Datei manifest.xml auf die manifest.dtd. Beide DTDs beinhalten Referenzen auf eine Vielzahl von externen Definitionen, die dann die speziellen Bereiche abdecken.
Alle Openoffice.org XML- Dateien beinhalten dasselbe standardisierte XML- Datenformat, welches in den DTDs spezifiziert wird. Dort wird also festgelegt, wie bereits in Abschnitt 2.2.2. erläutert, welcher Inhalt für ein gültiges Dokument erlaubt ist, und welcher nicht.
Openoffice.org hat für alle Komponenten im Bereich

- allgemeiner Dokumenteninhalt
- Textinhalt
- Tabelleninhalt
- graphischer Inhalt
- Formanweisungen
- Verzeichnisse
- Diagramme
- Benutzerschnittstellen
- Dialogkomponenten

[32] Vgl. [OO.o deutsch 2003] FAQs

[33] Mehr dazu unter http://xml.openoffice.org/filter/

[34] Mehr dazu unter http://xml.openoffice.org/xmerge/docbook/

[35] Vgl. [OO.o XML File Format 2003] S. 27-35

- und für das Package Format

eine Vielzahl von Stil- und Konstruktionsregeln.
Beispielhaft soll mit 3 Komponenten die interne XML- Struktur vorgestellt werden.

a. Definition eines Zeilenumbruchs[36]

Zeilenumbrüche gehören zu Textdefinitionen und werden in OpenOffice.org wie folgt festgelegt:

```
<!ELEMENT text:line-break EMPTY>
```

Das Schlüsselwort EMPTY gibt dabei an, dass das Element weder Kind-Elemente, noch Zeichendaten enthält.[37]
In den Dokumenten selbst wird dieses Element durch

```
<text:line-break/>
```

aufgerufen.

b. Definition einer Liste[38]

Listen können nummeriert sein oder auch ein einheitliches Zeichen wie einen Punkt besitzen.
Eine solche Liste hat in Openoffice.org bestimmte Strukturinformationen:

- Listentyp (nummeriert oder unnummeriert)
- Gliederungsebene der Liste
- Verwendetes Zeichen (Zahl, römische Zahl, Kreis, Quadrat etc.)
- Anzahl der Listeneinträge

Und Layoutinformationen:

- Einrückung der Listeneinträge
- Breite und Abstand zwischen Überschrift und Text
- Bild oder Zeichen für unnummerierte Listen
- Nummerformat für nummerierte Listen

Als Beispiel sind die Gültigkeitsregeln für die Entscheidung, ob eine Liste nummeriert oder unnummeriert sein soll, wie folgt geregelt:

```
<!ENTITY %list-items "((text:list-header,text:list-item*) | text:list-item+)">
<!ELEMENT text:ordered-list %list-items;>
<!ELEMENT text:unordered-list %list-items;>
```

Diese Definition beschreibt allgemein eine Entität, also eine Abkürzung, für Listenelemente. Diese werden selbst nochmals in der Definition verwendet, da es für Listen noch eine große Zahl möglicher Attribute gibt. Dabei ist es möglich, dass die Liste eine Überschrift und null bis unendlich viele Listeneinträge hat, oder aus 1 bis unendlich vielen Listeneinträgen besteht.[39]

---

[36] Vgl. [OO.o XML File Format 2003] S. 121 und Die Datei text.mod

[37] Vgl. [Stein u.a. 2001] S. 75

[38] Vgl. [OO.o XML File Format 2003] S. 130-138 und die Datei text.mod

[39] Vgl. [Stein u.a. 2001] S. 72

Diese Definition wird dann in den Elementen, die beschreiben, ob eine Liste nummeriert, oder unnummeriert ist, wiedergeben.
Im eigentlichen Dokument werden dann die Listeneinträge durch das Element

```
<text:ordered-list bzw. unordered-list>
…
<text:ordered-list bzw. unordered-list>
```

umschlossen. Daneben gibt es noch einige weitere Attribute für Listen, die in der veröffentlichten XML- Spezifikation von OpenOffice.org (zu finden unter http://xml.openoffice.org/xml_specification.pdf , oder direkt in der DTD für Text (text.mod im Verzeichnis \OpenOffice.org1.0.3\share\dtd\officedocument\1_0 der OpenOffice.org Installation) nachgelesen werden können.

c. Definition eine Tabelle[40]

Die offizielle OpenOffice.org- Spezifikation enthält zum Thema Tabellen 85 Seiten DTD-Definitionen, die unter anderem diese Bereiche behandeln:

- Dokumentensicherheit
- Berechnungen
- Tabellen
- Zeilen
- Spalten
- Tabellenzellen
- Inhaltsvalidierung
- Subtabellen
- Beschriftungen
- Namensausdrücke
- Filter
- Datenbank
- Verbinden
- Tabellenformatierungseigenschaften
- Spaltenformatierungseigenschaften
- Tabellenzellenformatierungseigenschaften

Diese können ebenfalls in der oben genannten Spezifikation oder in der Datei table.mod selbst nachgelesen werden. Deswegen soll hier nur ein kurzer schematischer Aufbau einer Tabelle erfolgen:

```
<!ENTITY % table-columns "table:table-columns | ( table:table-column | table:table-column-group )+">
<!ENTITY % table-header-columns "table:table-header-columns">
<!ENTITY % table-rows "table:table-rows | ( table:table-row | table:table-row-group )+">
<!ENTITY % table-header-rows "table:table-header-rows">
<!ENTITY % table-column-groups "((%table-columns;),(%table-header-columns;,(%table-columns;)?)?) |
(%table-header-columns;,(%table-columns;)?)">
<!ENTITY % table-row-groups "((%table-rows;),(%table-header-rows;,(%table-rows;)?)?) | (%table-
header-rows;,(%table-rows;)?)">
```

[40] Vgl. [OO.o XML File Format 2003] S. 255 ff. und die Datei table.mod

```
<!ELEMENT table:table (table:table-source?, table:scenario?, office:forms?, table:shapes?, (%table-column-groups;), (%table-row-groups;))>
```

Auch hier werden wieder später zu verwendende Abkürzungen, Entitäten, festgelegt. Eine Tabelle besteht also aus Spalten (columns) und Zeilen (rows), die alleine oder (|) in einer Gruppe stehen können, und zwar einmal oder mehrmals (+). Auch Zeilen- oder Spaltenüberschriften sind kein- oder einmal (?) möglich.
Wird nun das Element <table:table...> im Dokument aufgerufen, müssen Einstellungen, Spalten, Zeilen und können Quelle, Aufbau, Formen und Gestaltung übergeben werden.
Im folgenden können und müssen noch eine Vielzahl Attribute für die Tabelle folgen.

3.3. XML- Parser

Um ein XML- Dokument in ein anderes Format oder in ein spezielleres XML- Dokument umzuwandeln, benötigt man zusätzlich einen Parser. Er liest nach Überprüfung der Wohlgeformtheit und je nach Parser auch der Gültigkeit das Quelldokument und die XSLT-Anweisungen ein und verarbeitet sie zu einem neuen Dokument. Die Abarbeitungsfolge hängt von der verwendeten API ab. Ein SAX-Parser ist kein Standard, wird aber oft verwendet, da er auf ereignisorientierte Weise das XML- Dokument sequentiell einliest. Ein DOM- Parser (Document Object Model) dagegen ist bei größeren Projekten langsamer und braucht mehr Speicherplatz, da er zuerst das komplette Dokument einliest.

3.3.1. Vorstellung aktueller Parser

Auf dem Markt gibt es momentan über sechzig Parser vor allem für Java, C, C++ und Delphi. Es gibt Parser, die lediglich Dokumente validieren und Fehlermeldungen zurückgeben können. Erst durch die Verwendung von Application Programming Interfaces können auf die Dokumente selbst zugegriffen werden. Fast alle Parser unterstützen SAX, viele DOM, und einige von ihnen inzwischen auch Schemata.
Der Grossteil der XML- Parser steht kostenlos zum Download bereit. Nur wenige Ausnahmen wie zum Beispiel CMarkup für C++ und Delphi sind kommerziell. [41]

Einige der kostenlosen Parser seien hier vorgestellt. Ausgewählt wurden sie anhand der Nennungshäufigkeit in der Literatur und ihrer Leistungsfähigkeit (die hier vorgestellten Parser unterstützen alle SAX, DOM und Schemata).

[41] [XML Magazin 2003] S. 77-78

Name, Version	Hersteller	SAX	DOM	Schema	Beschreibung	Download unter
Apache Xerces-J 2.5.0.	Apache Software Foundation	X	X	X	Der Apache Xerces-J ist eine Weiterentwicklung des IBM XML4J-Parsers. Er ist in Java geschrieben und hat einen sehr hohen Funktionsumfang. Er kann nach DTDs und XML- Schema validieren, XSLT- Transformationen durchführen und ist der am weitesten verbreitete und umfangreichste XML-Parser für Java. Xerces ist der featurereichste Java- XML- Parser.	Auf den XML- Seiten der Apache-Software Foundation http://xml.apache.org/xerces2-j Zusatzprogramm XIncluder für Xinclude-Anwendungen http://xincluder.sourceforge.net
LibXML2	Daniel Veillard	X	X	X	Bei LibXML handelt es sich um einen sehr schellen, in C geschriebenen XML-Parser, der für Unix, Linux und Cygwin verfügbar ist. LibXML2 kann XIncludes verarbeiten. LibXML2 ist auf fast jedem Linux-System installiert.	Ursprünglich für das Gnome-Projekt geschrieben, kann man LibXML2 unter http://xmlsoft.org als Einzel-Version downloaden.
MSXML 4.0	Microsoft	X	X	X	Microsoft MSXML ist der schnellste XML- Parser für die Windows-Betriebssysteme und ist gleichzeitig ein XSLT- Prozessor. Er kann ebenso für C, C++, JavaScript, JScript und Visual Basic aufgerufen werden. Mittlerweile sind vier verschiedene Versionen des MSXML- Parsers verfügbar, je nach Betriebssystemversion und Version des installierten Microsoft Internet Explorer.	Auf den Microsoftseiten unter http://msdn.microsoft.com/xml/ Zusatzprogramm von Frank Wessels für XInclude-Anwendungen unter http://www.codeproject.com/soap/xinclude.asp

Xalan 2.5.1.	Apache Software Foundation	X	X	X	Xalan ist ein javabasiernder Parser, der von Apache zur Verfügung gestellt wird. Er implementiert XSLT 1.0 und XPath mittels DOM und SAX. Er ist besonders anwenderfreundlich, da er kommandozeilenbasierend, genauso wie als Applet oder Applikation in einem anderen Programm eingebunden werden kann.	Auf den XML- Seiten von Apache unter http://xml.apache.org/xalan-j/index.html kann die aktuelle 2.5.1. Version kostenlos bezogen werden.
XML4J 4.2.2	IBM	X	X	X	Auch der Parser von IBM basiert vollständig auf Java: Grundlage ist der Xerces Parser.	Die aktuelle 4.2.2. Version des IBM- Parsers kann unter alphaworks.ibm.com/tech/xml4j kostenlos bezogen werden.
V2 9.2.0.5.0.	Oracle	X	X	X	Dieser Parser von Oracle ist eine Neuentwicklung, die ebenfalls auf –Java basiert und ein effizienteres XML- Parsen verspricht. Auch er kann mit SAX oder DOM arbeiten und ist gleichzeitig ein XSLT- Prozessor.	Der Oracle Parser kann unter http://otn.oracle.com/tech/xml/xdk/staxpreview.html kostenlos bezogen werden.

Tabelle 3: Einige aktuelle XML- Parser[42]

3.3.2. Vorgehensweise von Parsern

Der Parser verarbeitet die Anweisungen des XSLT- Dokumentes.
Dabei können beide Dokumente in Baumstruktur auf abstrahierte Weise betrachtet werden.
So kann jedes XML- Dokument als Baum dargestellt werden. Da sowohl Quell- aus auch Zieldokument XML- Dokuments sind, arbeitet der Parser letztlich mit 3 Bäumen (siehe Abbildung 5).
Bei einem DOM- analysierten Dokument erzeugt der Parser Objekte und ordnet diese in einer Baumform an. Erst wenn der Parser das komplette Dokument analysiert hat, gibt er die Kontrolle über den Code wieder zurück. Bei sehr großen XML- Dokumenten kann es dabei zu deutlichen Verzögerungen kommen.[43]
Da XSLT eine deklarative Sprache ist, wird beschrieben, wie das Ergebnis nach der Transformation aussehen soll. Die Templates des Stylesheets geben an, wie ein Knoten aus der Quelle im Zieldokument aussehen soll.[44] Der Parser macht also tatsächlich nichts anderes, als jeden Knoten in der Quelle zu durchlaufen und nach entsprechenden Templates zu suchen. Findet er eine solche passende Verarbeitungsanweisung, wird diese auf den Knoten angewendet.
Die Funktionsweise des Parsers als Schema:

[42] Die Tabelle wurde mit Hilfe von [DocBook- XML 2003] S. 14-15, [XML Magazin 2003] S. 77 ff. und [XML Magazin Online 2002] zusammengestellt

[43] Vgl. [Tidwell 2002] S. 13- 16

[44] Vgl. [Paul u.a. 2000] S. 376- 377

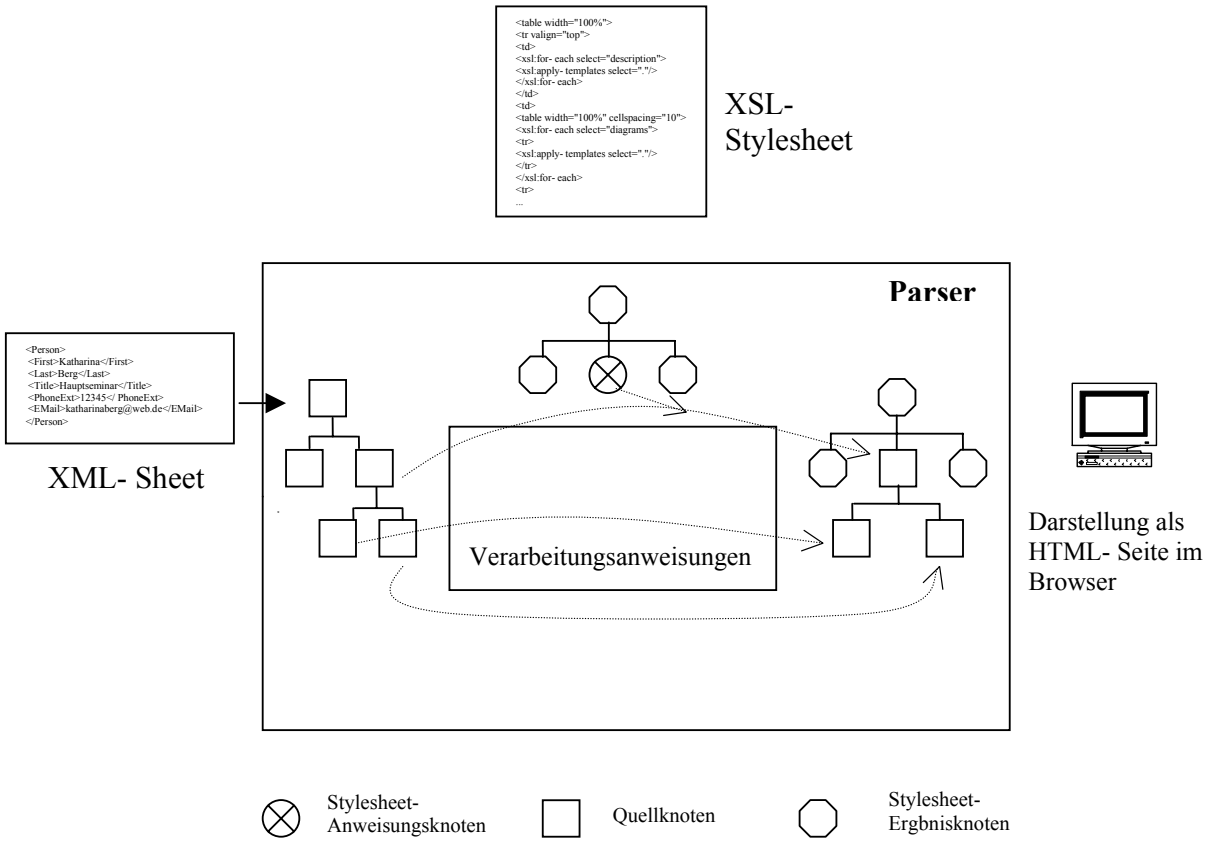

Abbildung 6: Umsetzung durch den Parser[45]

Parser nutzen für die Abarbeitung zwei verschiedene Application Programming Interfaces (API). Diese stellen zwei unterschiedliche Methodiken dar. Die erste ist das bereits erwähnte SAX. SAX, die aktuelle Version ist die 2.01[46], ist zwar kein offizieller Standard des W3C, wird aber als De- Facto- Standard oft in der Industrie verwendet. SAX- Parser arbeiten ereignisorientiert und interaktiv und sind damit schneller und wesentlich einfacher, da sie XML- Dokumente sequenziell abarbeiten und Elemente wie Ereignisse behandeln.
Das Document Objekt Model, kurz DOM, wird offiziell vom W3C standardisiert. Der Parser liest im Gegensatz zu SAX zuerst das ganze Dokument ein und erzeugt Objekte in einem neuen Dokument (Elemente, Attribute, Text, Kommentare etc.) und ordnet sie dann in einer Baumstruktur an. Erst nach kompletter Analyse gibt der Parser das Zieldokument aus. Damit wird für größere Dokumente ein höherer Speicherbedarf nötig und die Performance erreicht nicht

[45] Abbildung nach [Paul u.a. 2000] S. 377

[46] Vgl. [XML Magazin 2003] S. 77

die Qualität von SAX- Parsern, ist aber für manches Problem die einzige Lösung, da so die Baumstruktur vorwärts und rückwärts durchlaufen werden kann.[47]

[47] Vgl. [Tidwell 2002] S. 13- 17

4. Beispiel im DVP- Projekt

4.1. Vorstellung des Beispiels

Das „Digital Video Project“ ist ein Projekt von Studenten der TU- Ilmenau unter Leitung von Detlef Streitferdt.
Ziel ist es, einen digitalen Videorekorder, oder vielmehr ein Multimedia- System, zu entwerfen. In dieser Projektarbeit sollen dabei Anforderungen und Eigenschaften dieses Systems in einem OpenOffice.org- Dokument dargestellt werden. Die Daten liegen bereits in strukturierter Form, in einem XML- Dokument, vor.
Da OpenOffice.org Dateien in komprimierter Form speichert, wird als Beispiel im Rahmen dieser Projektarbeit die content.xml durch ein Template erzeugt und später in das Package eingebunden. Das Dokument, welches den Inhalt beinhaltet, bringt schließlich eine Vielzahl Informationen mit sich, die ebenfalls im Ausgabedokument festgelegt sein müssen. Das sind zuerst der Verweis auf die DTD von OpenOffice.org :

```
<!DOCTYPE office:document-content PUBLIC "-//OpenOffice.org//DTD OfficeDocument 1.0//EN"
"office.dtd">
```

und die Namensraum- Deklarationen von OpenOffice.org für:

Allgemeine Anweisungen	xmlns:office="http://openoffice.org/2000/office"
Styleanweisungen	xmlns:style="http://openoffice.org/2000/style"
Textdateien	xmlns:text="http://openoffice.org/2000/text"
Tabellen	xmlns:table="http://openoffice.org/2000/table"
Grafiken	xmlns:draw="http://openoffice.org/2000/drawing"
Formatting Objects (XSL)	xmlns:fo="http://www.w3.org/1999/XSL/Format"
Verlinkung	xmlns:xlink="http://www.w3.org/1999/xlink"
Nummerierung	xmlns:number="http://openoffice.org/2000/datastyle"
SVG (Scalable Vector Graphics)-Dateien	xmlns:svg=“http://www.w3.org/2000/svg“
Diagramme	xmlns:chart="http://openoffice.org/2000/chart"
3D- Grafiken	xmlns:dr3d="http://openoffice.org/2000/dr3d"
OpenOffice.org- Math	xmlns:math="http://www.w3.org/1998/Math/MathML"
Gestaltung	xmlns:form="http://openoffice.org/2000/form"
Aufbau	xmlns:script="http://openoffice.org/2000/script”

Tabelle 4: Namensraum- Deklarationen von OpenOffice.org[48]

[48] Die Angaben wurden der Datei content.xml eines beliebigen OpenOffice.org Dokumentes entnommen

Dazu kommen eine Reihe von Style- Anweisungen für verschiedene Schriftanwendungen (gemäß dem Fall, dass die Textkomponente verwendet wird). Beispielhaft der Aufbau:

```
<style:font-decl style:name="Arial Unicode MS" fo:font-family="&apos;Arial Unicode MS&apos;" style:font-pitch="variable"/>
```

Um jedoch ein solches Dokument zu erzeugen, werden zuerst ein umzuwandelndes XML-Dokument und anschließend das XSL- Template benötigt.

4.2. XML- Dokument

Das XML- Dokument besteht aus einem 1. Teil, dem RequirementsModel. Jede geforderte und somit dort beschriebene Anforderung an den digitalen Videorekorder hat das Attribut ID. Daneben werden sie über die Elemente Description, ShortDescription, Source, History mit dem Atrribut dateTime, Person und Change_Text näher beschrieben.
Besonders zu beachten ist, dass nicht jede Anforderung eine Description hat. Deswegen muss dafür gesorgt werden, dass bei einer leeren Description die ShortDescription erscheint.
Zudem haben viele Anforderungen untergeordnete Punkte, die diese näher beschreiben.
Ein Ausschnitt der strukturierten Daten:

```
1 <?xml version="1.0" encoding="UTF-8"?>
2
3 <FORE>
4   <RequirementsModel ID="22">
5       <requirement ID="1" classification="must-be">
6           <history dateTime="2002-10-01T14:00:22+01:00">
7               <person>Detlef Streitferdt</person>
8               <description>erzeugt</description>
9           </history>
10          <short_desc>Das System soll fernsteuerbar sein.</short_desc>
11          <description>Hier ist die Grundidee einer Fernbedienung gemeint.</description>
12          <source pers_id="11">Domänenanalyse</source>
13          <requirement ID="1.1" classification="indifferent">
14              <history dateTime="2002-10-01T14:00:22+01:00">
15                  <person>Detlef Streitferdt</person>
16                  <description>erzeugt</description>
17              </history>
18              <short_desc>Das System soll durch eine Infrarot-Fernsteuerung bedienbar
19               sein</short_desc>
20              <description>Es geht nur um die Art der Technologie.</description>
21                                  <source pers_id="11">Domänenanalyse</source>
22…
```

Bei genauer Betrachtung dieser Strukturen erkennt man, dass in einem XML- Dokument zuerst deklariert werden muss, um welche XML- Version es sich handelt und welcher Zeichensatz zu verwenden ist. Der hier verwendete UTF- 8 ist der komprimierte Unicode. In Zeile 3 wird das Wurzelelement des Dokuments festgelegt. Jedes wohlgeformte XML- Dokument hat ein Wurzelelement, da XML nur genau ein Element in der obersten Ebene erlaubt. Alle anderen Elemente müssen sich zwischen dem Start- und Endtag des Wurzelelementes befinden. Danach

steht das erste eigentliche Element: RequirementsModel. Alles was zu diesem Element gehören soll, muss zwischen seinem Start- und Endtag stehen. Ab Zeile 5 folgt nun die Beschreibung der Anforderung mit der ID 1 und im Anschluss die Definition einer untergeordneten Anforderung 1.1. und so weiter.

Im 2. Teil der Datei werden die Eigenschaften im FeatureModel beschrieben. Das Feature selbst wird näher durch die Attribute ID, Name, optional, und Type dargestellt. Das Element Description beschreibt die Eigenschaften näher.
In FeatureSets werden Unterfunktionen mit denselben Attributen und Elementen zusammengefasst.
Schema der Daten:

```
1<FeatureModel ID="1">
2       <feature ID="00" type="concept" name="Digital Video Project" optional="false">
3           <description>Das gesamte System</description>
4           <feature ID="1" type="concept" name="TV" optional="false">
5               <description>Alles zum simplen Fernsehen.</description>
6               <feature ID="1.1" type="functional" name="EPG" optional="false"/>
7               <feature ID="1.2" type="functional" name="EPG+" optional="true"/>
8               <feature ID="1.3" type="functional" name="Kanalprofil" optional="true"/>
9 …
10              <feature ID="1.12" type="concept" name="Ton" optional="false">
11                  <feature_set ID="1.12.1" type="concept" name="TonSet" optional="true" min="1"
12                      max="*">
13                      <feature ID="1.12.1.1" type="functional" name="Stereo" optional="true"/>
14                      <feature ID="1.12.1.2" type="functional" name="Dolby Digital" optional="true"/>
15                      <feature ID="1.12.1.3" type="functional" name="Dolby 5.1" optional="true"/>
16                  </feature_set>
17              </feature>
18          </feature>
19          <feature ID="2" type="concept" name="Abspielen" optional="false">
20 …
21      </Feature>
22 </FeatureModel>
23 </FORE>
```

Damit ist die semantische Ausdrucksweise der Daten fertig. Zu beachten ist, dass bei XML jedes Tag in der richtigen Reihenfolge geschlossen werden muss. In Zeile 23 endet das Wurzelelement.

4.3. XSL(T)- Dokument

Das XSL(T)- Dokument dient dazu, das XML- Dokument umzuwandeln.
Als erster Teil müssen die oben genannten allgemeinen OpenOffice.org Anweisungen erfolgen, sowie anschließend die Schriftanweisungen für die Tabellenzellen. Beispielhaft die Festlegung einer Tabelle, einer Spalte und einer Zelle:

```
<style:style style:name="Tabelle1" style:family="table">
    <style:properties style:width="16.999cm" table:align="left"/>
</style:style>

<style:style style:name="Tabelle1.A" style:family="table-column">
    <style:properties style:column-width="1.39cm"/>
```

```
</style:style>
…
<style:style style:name="Tabelle1.A1" style:family="table-cell">
    <style:properties fo:vertical-align="middle" style:border-line-width-left="0.002cm 0.088cm 0.002cm"
    style:border-line-width-bottom="0.002cm 0.088cm 0.002cm" fo:padding="0.049cm" fo:border-  left="0.092cm
double #808080" fo:border-right="none" fo:border-top="none" fo:border-bottom="0.092cm double #808080"/>
</style:style>
…
```

Und ein Beispiel für die Style- Anweisungen eines Listenelementes in einer ungeordneten Liste:

```
<text:list-style style:name="L1">
        <text:list-level-style-bullet text:level="1" text:style-name="Bullet Symbols" style:num-suffix="."
        text:bullet-char="">
                <style:properties text:space-before="0.748cm" text:min-label-width="0.499cm" style:font-
                name="StarSymbol"/>
         </text:list-level-style-bullet>
```

Im hier zu besprechenden Beispiel sollen in Tabellenzellen die Requirements in entsprechender Reihenfolge mit ihrer Kurzbeschreibung ausgegeben werden. Bei beiden Auflistungen besteht das Problem, per Rekursion die untergeordneten Ebenen, also die jeweiligen Kind- Knoten der Requirements und der Features anzusprechen. Und schließlich sollen die verschachtelten Features in den FeatureSets ausgegeben werden. Dazwischen müssen die entsprechenden Tabellen- und Listenanweisungen von OpenOffice.org implementiert werden.

```
1    <xsl:output
2        method="xml" version="1.0"
3        encoding="UTF-8"
4        omit-xml-declaration="no"
5        doctype-public="-//OpenOffice.org//DTD OfficeDocument 1.0//EN"
6        doctype-system="office.dtd"/>
7
8         <xsl:template match="FORE">
9…
10<!—Hier erfolgen verwendete Namensräume und Style- Anweisungen für die Tabelle und die Liste, wie bereits in
den Kapiteln zuvor erläutert -->
11…
12</office:automatic-styles>
13            <office:body>
14                <text:sequence-decls>
15                    <text:sequence-decl text:display-outline-level="0" text:name="Illustration"/>
16                    <text:sequence-decl text:display-outline-level="0" text:name="Table"/>
17                    <text:sequence-decl text:display-outline-level="0" text:name="Text"/>
18                    <text:sequence-decl text:display-outline-level="0" text:name="Drawing"/>
19                </text:sequence-decls>
20
21      <xsl:apply-templates select="RequirementsModel"/>
22      <xsl:apply-templates select="FeatureModel"/>
23
24            <text:p text:style-name="Text body"><office:annotation office:create-date="2003-11-09"><text:p>&lt;!--
              EndFragment--25&gt;</text:p></office:annotation></text:p>
26            </office:body>
27 </office:document-content>
28
29  </xsl:template>
30
31  <xsl:template match="RequirementsModel">
32
33      <text:h text:style-name="Heading 2" text:level="2">DVP – Projektarbeit von Katharina Berg</text:h>
```

```
34      <text:p text:style-name="P1">Anforderungen an den digitalen Videorekorder</text:p>
35      <table:table table:name="Tabelle1" table:style-name="Tabelle1">
36          <table:table-column table:style-name="Tabelle1.A"/>
37          <table:table-column table:style-name="Tabelle1.B"/>
38
39      <xsl:apply-templates select="requirement"/>
40
41      </table:table>
42  </xsl:template>
43
44  <xsl:template match="requirement">
45      <table:table-row>
46              <table:table-cell table:style-name="Tabelle1.A1" table:value-type="string">
47                  <text:p text:style-name="P2">
48                      <xsl:value-of select="@ID"/>
49                  </text:p>
50              </table:table-cell>
51              <table:table-cell table:style-name="Tabelle1.B1" table:value-type="string">
52                  <text:p text:style-name="Table Contents">
53                      <xsl:value-of select="short_desc"/>
54                  </text:p>
55              </table:table-cell>
56          </table:table-row>
57          <xsl:apply-templates select="requirement"/>
58  </xsl:template>
59
60  <xsl:template match="FeatureModel">
61
62      <text:p text:style-name="Text body"/>
63      <text:p text:style-name="Text body"/>
64      <text:h text:style-name="Heading 2" text:level="2">
65          <office:annotation office:create-date="2003-11-09">
66              <text:p>&lt;!--StartFragment --&gt;</text:p>
67                  </office:annotation>Eigenschaften des digitalen Videorekorder</text:h>
68      <text:unordered-list text:style-name="L1">
69
70      <xsl:apply-templates select="feature"/>
71
70      </text:unordered-list>
72
73  </xsl:template>
74
75  <xsl:template match="feature">
76
77      <text:list-item>
78          <text:p text:style-name="P3">
75              <text:span text:style-name="T1">
76                  <xsl:value-of select="@ID"/>
77              </text:span>
78              <text:line-break/>
79                  <xsl:value-of select="@name"/>
80          </text:p>
81      </text:list-item>
82
83  <xsl:apply-templates select="feature"/>
84
85  </xsl:template>
86
87</xsl:stylesheet>
```

Dieses Stylesheet beginnt mit einer erweiterten Output- Definition, in der in den Zeilen 1 bis 6 diverse Informationen übergeben werden. In Zeile 8 beginnt das Haupttemplate, in dem das Wurzelelement angesprochen wird. Darauf folgen eine Reihe Style- Anweisungen für die Tabellen- und Listenstruktur. Diese sollen aufgrund ihres Umfangs hier verkürzt dargestellt werden. Der schematische Aufbau erfolgte bereits am Anfang des Kapitels 4.3.
Anschließend wird auf die allgemeingültigen Styles der verschiedenen OpenOffice.org Anwendungen verwiesen, die zusätzlich benötigt werden.
Die Zeilen 21 und 22 rufen dann entsprechend Templates für die Knoten „RequierementsModel“ und „FeatureModel“ auf. Dem folgend wird eine Schlussnote übergeben und das OpenOffice.org-Dokument und das Template geschlossen.
Findet der Parser im Quelldokument den Knoten „RequirementsModel“, so soll er eine Überschrift ausgeben und dann eine Tabelle beginnen. Um die Felder mit Daten zu füllen, wird in Zeile 39 eine Template für den Knoten „Requirement“ aufgerufen, welches dann die Inhalte wiedergibt.
Danach werden Tabelle und Template geschlossen.
In den Zeilen 31 bis 59 erfolgen nun Anweisungen für die Behandlung von Requirements-Knoten. Die Daten erhalten Verweise auf den zu verwendenden Style und eine Absatzformatierung. Zellen und Zeilen werden entsprechend geschlossen.

Das Template für das FeatureModel ist analog aufgebaut. Hier bekommt der Parser ab Zeile 60 die Anweisung, beim Auffinden des Knotens mit dem Namen „FeatureModel“ eine Überschrift auszugeben und anschließend eine unnummerierte Liste zu beginnen. Die Werte werden mittels eines weiteren Templates in den Ausgabetext eingefügt. Sie erhalten ebenfalls Style-Anweisungen und eine Absatzformatierung.
Wenn keine Knoten mehr gefunden werden, wird die Liste und das Template beendet und schließlich in Zeile 87 das Stylesheet ganz geschlossen.

4.4. Ergebnis in Openoffice.org

Da OpenOffice.org momentan keine direkten XML- Strukturen verarbeiten kann, ist es nötig, das Zieldokument in einem Package zu speichern und anschließend zu komprimieren.
Da einige Parser Probleme mit der Verarbeitung von Whitespaces haben, mussten per Hand einige Formatierungsänderungen vorgenommen werden.
Das Ergebnis ist ein eigenständiges OpenOffice.org- Dokument, welches eine Tabelle und eine Liste mit den Daten des DVPs enthält.

Das Ergebnisdokument in OpenOffice.org in zwei Screenshots:

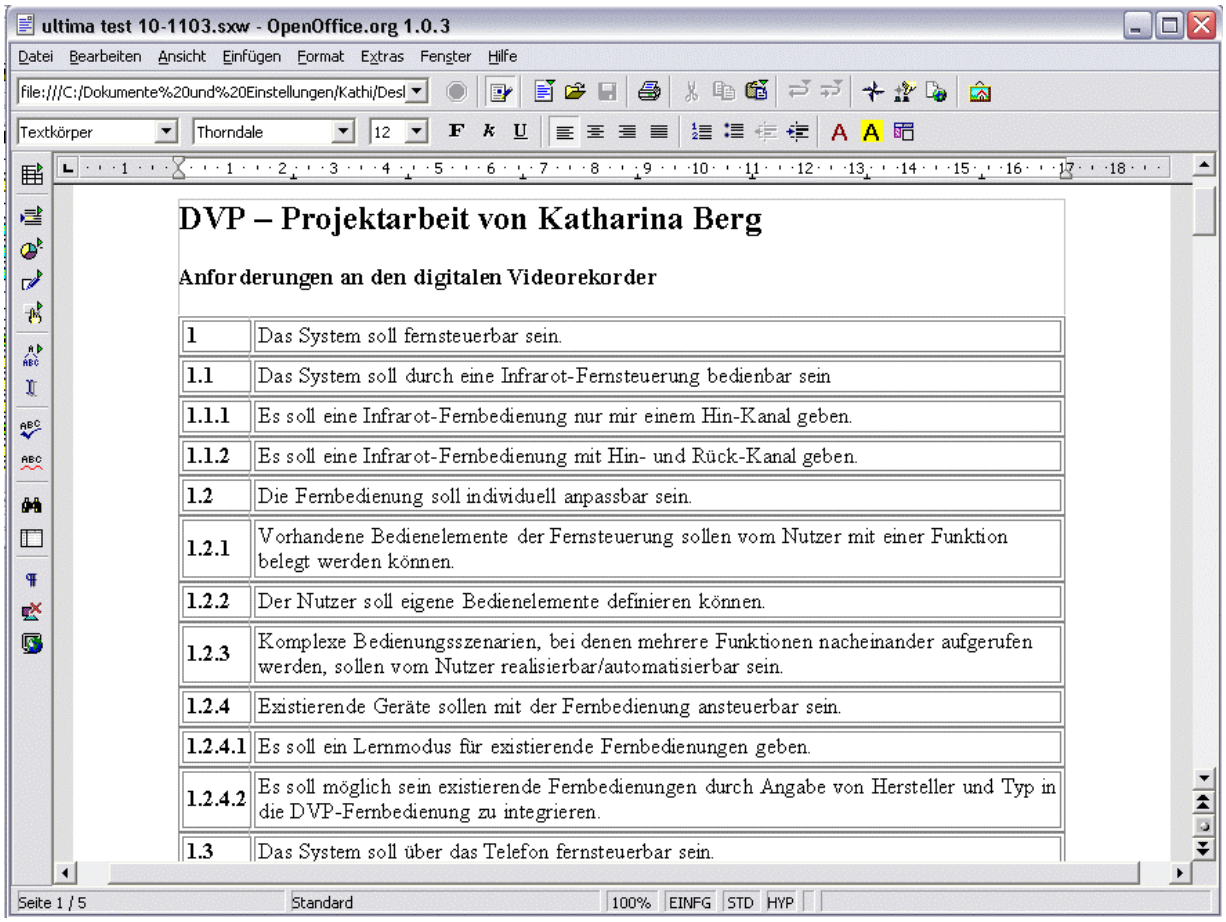

Abbildung 7: Screenshot der Ergebnistabelle in OpenOffice.org

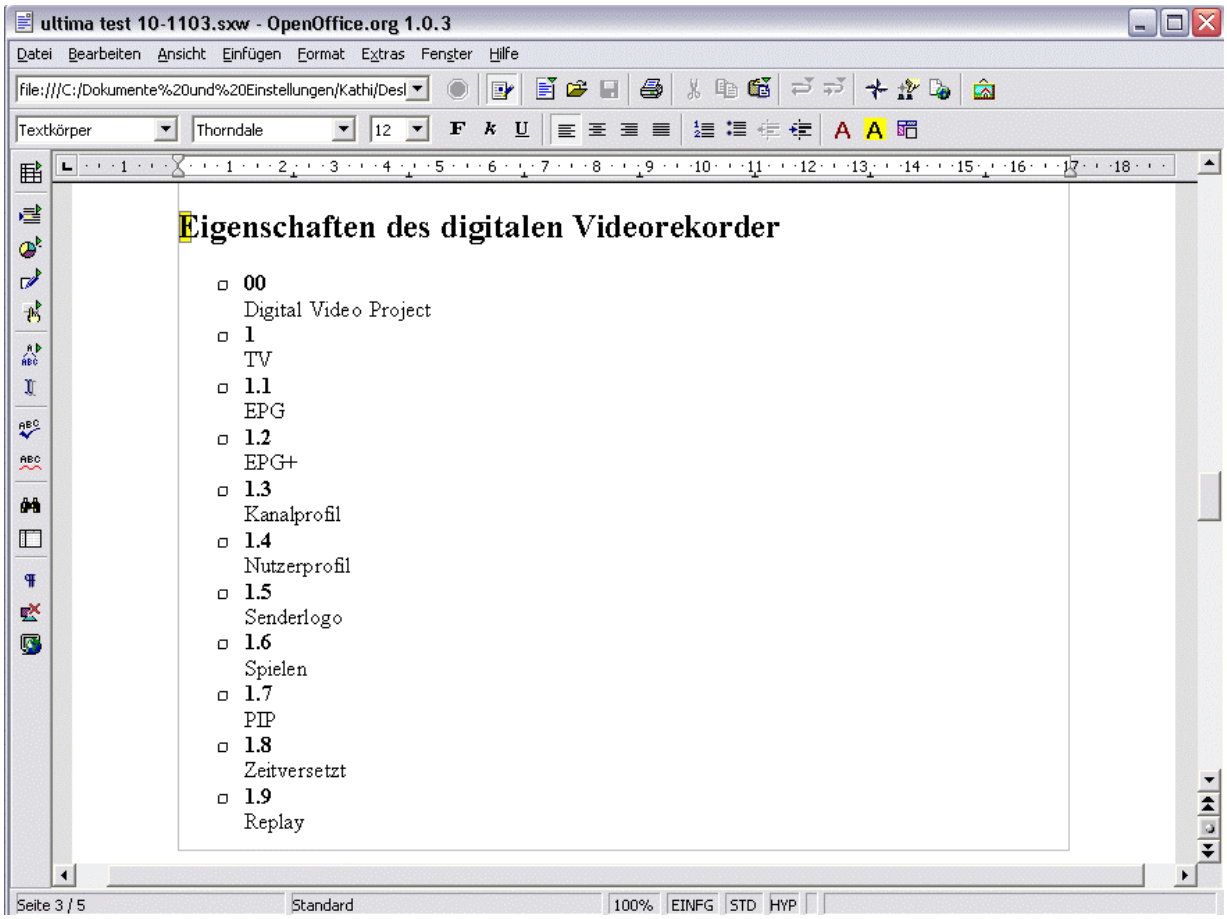

Abbildung 8: Screenshot der Ergebnisliste in OpenOffice.org

5. Zusammenfassung und Ausblick

5.1. Zusammenfassung

Die Entwicklung von XML befindet sich in einem rasanten Wachstum. Ein geschichtlicher Überblick wurde dazu im Kapitel 2 gegeben. Als Beispiele wurden hier einige Ideen und Arbeitsvorschläge des W3C, wie zum Beispiel XSL und XSLT, vorgestellt. Auch OpenOffice.org treibt die Evolution von XML und neuer Standards voran. Hier wurden das Verwenden von flachen XML- Strukturen und die xMerge- Schnittstelle genannt. Dazu wurde im Kapitel 3 ausführlich auf OpenOffice.org und seine grundlegenden XML- Struktur eingegangen. Der Fokus wurde dazu auch auf die Ideen und Ziele des Opensource Projektes gelegt und inwiefern diese anhand von XML umgesetzt wurden und auch weiterhin realisiert werden.
Um den aktuellen Stand beschreiben zu können, wurden im 2. Kapitel neben der Geschichte der Aufbau der Auszeichnungssprachen behandelt und anschließend das Grundschema von XML erklärt. Konkretisiert wurde XML dann anhand einer Umwandlung von XML- Daten in das OpenOffice.org Format. Zudem wurde erklärt, wie genau XML verarbeitet wird, und welche Möglichkeiten es dazu am Markt gibt.
Und schließlich hat eine praktische Umsetzung gezeigt, wie sich Daten in OpenOffice.org Format umwandeln lassen.

5.2. Ausblick

Diese Arbeit zeigte nur einen sehr kleinen Teilbereich der vielen Anwendungsmöglichkeiten von XML, doch die Metasprache XML gewinnt in vielerlei Hinsicht an Bedeutung.
Dieser Ausblick soll weitere Anwendungen und Weiterentwicklungen vorstellen.

Für viele, gerade Vertreter der Opensource Community, ist XML quasi zu einer Lebensphilosophie geworden. Der Erfolg von OpenOffice.org ist dementsprechend durchschlagend. So wurde bis Oktober 2003 bereits 18 Millionen Mal die OpenOffice.org Suite allein von der offiziellen Webseite heruntergeladen[49].
XML hat schon lange die Fähigkeiten von HTML hinter sich gelassen, da die Sprache permanent erweitert wird und als offener Standard im Gegensatz zu propritären Systemen von jedermann weiterentwickelt werden kann. Zudem braucht man aufgrund der Einfachheit der Sprache nur wenige Programmierkenntnisse und fast alle benötigten Dateien werden in ein und derselben Sprache geschrieben. Zur Unterstützung der Anwender gibt es inzwischen Editoren, die die Anwendung von XML erleichtern. Das macht die Weiterentwicklung schneller als je zuvor. Viele Anwendungen ziehen entsprechend nach und rüsten mit Filtern, Editorfähigkeiten für XML oder XML als grundlegenden Datenhaltungsstandard nach.
Auch hier wurde im Rahmen von OpenOffice.org die xMerge- Schnittstelle für den Austausch von Docbook Informationen genannt.
Zusätzlich soll es in OpenOffice.org bald möglich sein, flache XML- Strukturen zu verwenden, und damit leichter änderbare Dokumente zu entwickeln.
Dazu nennt das XML Magazin über sechzig anwendbare Parser, die fast ausschließlich kostenfrei zu beziehen sind.

[49] Diese Zahl wurden den deutschen Presse- FAQs entnommen: http://de.openoffice.org/presse/presse_faq_1.1.html

XML gewinnt vor allem auf der Seite der Webentwickler Anerkennung, die somit große Projekte leichter variieren und mit HTML und anderen Websprachen kombinieren können. Gerade SMIL ermöglicht neuartige Ablaufstrategien von Multimediainhalten. Zudem lassen sich Dokumente einfach filtern, berechnen und in andere Formate umwandeln. So kann beispielsweise ein Hersteller einen datenbankgestützten Produktkatalog auf seiner Seite veröffentlichen und gleichzeitig Kataloge für einen bestimmten Kundenkreis zusammenstellen und im PDF- Format zum Download anbieten.
Auch für den Print- on- Demand Bereich ist damit XML von großer Relevanz. Viele Softwarehersteller setzen heute auf anschauliche digitale Dokumentationen und Handbücher. Mit einem entsprechenden Schema lässt sich auch dies leicht bewerkstelligen.

Vor allem in unternehmensweiter und unternehmensübergreifender Hinsicht wird und ist XML von immenser Bedeutung, da mit XML Daten plattform- und systemunabhängig dargestellt werden können. Nicht nur die Übertragungsmöglichkeiten, auch die Datenintegration benötigt inzwischen den Einsatz von XML.
Die Vernetzung von Unternehmen und ihrer Geschäftstätigkeit mit Partnern und Zulieferern benötigt eine qualitativ hochwertige Datenintegration aus allen relevanten Quellen in lesbarer, konsistenter und verständlicher Weise. Diese Aufgabe hat sich unter anderem das Xperanto–Projekt von IBM vorgenommen. Dieses Projekt will mit Hilfe von XML nicht nur das „Realtime-Unternehmen“, sondern auch die Unterstützung von ERP (Enterprise Resource Planning), CRM (Customer Relationsship Management) und SCM (Supply Chain Management) ermöglichen.
Diese Ziele benötigen eine automatisierte Datenintegration, die die Sicherheit der Datenqualität gewährleistet und die Beziehungen zwischen Daten sicherstellt. Und schließlich müssen Daten transformiert und transportiert werden. All dies sind Aufgaben, die durch die Verwendung von XML abgedeckt werden können.[50]
Es ermöglicht somit auch ein verbessertes Dokumentenmanagement und Erweiterungen von Datenbankanwendungen, denn mit XLST lassen sich Datenströme in Datenbanken speichern, öffnen und bearbeiten. Diese Datenströme selbst können wieder XML- Dokumente sein. In diesem Bereich gibt es noch viele Ansätze, die weiterverfolgt werden.

Momentan arbeitet das W3C an der XSLT Version 2.0. Hier sind eine Reihe von Änderungen vorgenommen worden. Vor allem wurden viele Bugs entfernt.
Inwiefern diese Version erweitert oder abgeändert wurde, kann auf den offiziellen Seiten des W3C nachgelesen werden. [51]

[50] Vgl. [XML Magazin 2003] S. 17 ff.

[51] Der aktuelle Stand dieses Arbeitsvorschlags des W3Cs kann unter http://www.w3.org/TR/xslt20/ nachgelesen werden

Abkürzungsverzeichnis

API	Application Programming Interface
CML	Chemical Markup Language
CRM	Customer Relationsship Management
CSS	Cascading Style Sheet
DVP	Digital Video Project
DTD	Document Type Definition
EDI	Electronic Data Interchange
ERP	Enterprise Resource Planning
HTML	Hypertext Markup Language
DOM	Document Object Model
PCDATA	Parsed Character Data
PDF	Portable Document Format
SAX	Simple API for XML
SCM	Supply Chain Management
SGML	Standard Generalized Markup Language
SMIL	Synchronized Multimedia Integration Language
SVG	Scalable Vector Graphics
VRML	Virtual Reality Modelling Language
W3C	World Wide Web Consortium
WML	Wireless Markup Language
XHTML	Extensible Hypertext Markup Language
XML	Extended Markup Language
XSD	Schema
XSL	Extensible Stylesheet Language
XSL- FO	Extensible Stylesheet Language for Formatting Objects
XSLT	Extensible Stylesheet Language Transformation

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[Akademie 2002] Akademie für Lehrerfortbildung und Personalführung- Glossar, 2002. http://nt1.alp.dillingen.de/telumm/schilfglossar/ Abruf: 5.5.2003

[DocBook- XML 2003] Dokumentationen mit DocBook-XML Einführung in die Erstellung und Publikation von DocBook-Dokumenten, Lars Trieloff, 2003. http://trieloff.net/docbook/tutorial/ Abruf: 23.8.2003

[Jung Background 2002] Frank Jung, Software AG– Background XML, 2002. http://www.softwareag.com/tamino/download/d-XML_Backgrounder_XML- WP04D0502.pdf Abruf: 26.4.2003

[Kaldewey 2003] Das deutsche OpenOffice.org Portal, 2003. http://www.kaldewey-online.de/postnuke/index.php Abruf: 4.9.03

[OO.o 2003] OpenOffice.org Homepage http://www.openoffice.org Abruf: ab 12.8.2003

[OO.o deutsch 2003] OpenOffice.org 1.1. Office Suite für den deutschsprachigen Raum, 2003. http://de.openoffice.org/ Abruf: 23.9.03

[OO.o XML File Format 2003] OpenOffice.org XML File Format 1.0, Technical Reference Manual, 2002. http://xml.openoffice.org/xml_specification.pdf Abruf: 23.8.03

[Paul u.a. 2000] Eduard Paul, Barbara Jaekel, Uwe Jaeckel und Reinhard Engel: XML professionell. MITP Verlag, 2000.

[Pott u.a. 2000] Oliver Pott, Gunter Wielage: XML – Praxis und Referenz. 2. Auflage, Markt& Technik Verlag.

[Shepherd 2002] Devan Shepherd: XML in 21 Tagen. Markt + Technik Verlag, 2002.

[Stein u.a. 2001] Magnus Stein, Ingo Dellwig: XML.
Addison- Wesley Verlag, 2001.

[Streitferdt DVP 2003] Digital Video Project, 2003.
http://proinf.de/DVP
Abruf: 12.5.03

[Tidwell 2002] Doug Tidwell: XSLT.
O'Reilly Verlag, 2002.

[UseGroup 2001] useGroup Software- und Webentwicklung, 2001.
http://www.usegroup.de/software/xmltutorial/
Abruf: 20.4.2003

[W3C History] W3C, 1998.
http://www.w3.org/History/1989/proposal.html
Abruf: 5.5.2003

[W3C Introduction 2003] Einführung in XML, 2003.
http://www.w3c.org/XML/
Abruf: 5.5.2003

[W3C XSLT 1999] W3C, 1999.
http://www.w3.org/TR/xslt
Abruf: 16.4.2003

[W3C XSLT deutsch 2002] W3C – Deutsche Übersetzung, 2002.
http://xml.klute- thiemann.de/w3c- de/REC- xslt- 20020318/
Abruf: 12.6.2003

[XML Magazin Online 2002] XML Magazin & Webservices Online, 2003.
http://jax2003.de/itr/online_artikel/psecom,id,94,nodeid,69.html
Abruf: 15.10.03

[XML Magazin 2003] XML Magazin & Webservices, Ausgabe 6 2003.
Software & Support Verlag GmbH