Fachhochschule Braunschweig/Wolfenbüttel
Fachbereich Produktions- und Verfahrenstechnik
Studiengang Industrieinformatik

Entwicklung eines Verfahrens zur layoutgetreuen Dokumenterstellung unter Einsatz von XML- und Java-Technologien

Diplomarbeit zum Erlangen des Titels Dipl.-Ing. (FH)

Eingereicht von

Larissa Ochsner

Betreuender Dozent:	Prof. Dr. rer. nat. Klaus Harbusch
	Dipl.-Math. Anna Rudy
Abgabetermin:	August 2005
Unternehmen:	Hönigsberg&Düvel Datentechnik GmbH
Betreuer im Unternehmen:	Dipl.-Inf. Thomas Kanschik

Abkürzungsverzeichnis

API	Application Programming Interface
ASP	Active Server Pages
C	C Programmiersprache
C++	objektorientierte Programmiersprache
COM	Component Object Model
CORBA	Common Object Request Broker Architecture
DHTML	Dynamic Hypertext Markup Language
DOM	Document Object Model
DTD	Document Type Definition
Dublin Core(DCMI)	Dublin Core Metadata Standard
FOP	Formatting Objects Processor
GUI	Graphical User Interface
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IBM	International Business Machines
IDE	Integrated Development Environment
IO	Input und Output
J++	Programmiersprache
JAXP	Java API for XML Processing
JDK	Java Delelopment Kit
JRE	Java Runtime Environment
JVM	Java Virtual Maschine
KBSt	Koordinierungs-und Beratungsstelle der Bundesregierung für Informationstechnik in der Bundesverwahltung
LGPL	Lesser General Public Licence
MathML	Mathematical Markup Language
MOM	Message Oriented Middleware
MPL	Mozilla Public Licence
MS	Microsoft
MS SQL	Microsoft SQL Server
OASIS	Organization for the Advancement of Structured Information Standards
OLAP	Online Analytical Processing
OLE	Object Linking and Embedding
OOo	OpenOffice.org

PC	Personal Computer
PDF	Portable Document Format
PHP	Hypertext Preprocessor
PLG	Programming Language for Graphics
POP	Presentation Oriented Publishing
RTF	Rich Text Format
SAX	Simple API for XML
SGML	Standard Generalized Markup Language
SO	StarOffice
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
Sun	Sun Microsystems
SVG	Scalable Vector Graphics
UNO	Universal Network Objects
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
VB	Visual Basic
VB.NET	Visual Basic.NET
VCL	Visual Class Library
VM	Virtual Machine
VOS	Virtual Operating System
WEB	World Wide Web
WH2FO	Word HTML 2 Formatting Objects
WML	Wireless Markup Language
WYSIWYG	What You See Is What You Get
W3C	World Wide Web Consortium
XHTML	Extensible Hypertext Markup Language
XLINK	XML Link
XML	Extensible Markup Language
XPath	XML Path Language
XSL	Extensible Stylesheet Language
XSL-FO	Extensible Stylesheet Language-Formatting Objects
XSLT	XSL Transformation

Ehrenwörtliche Erklärung

„Hiermit erkläre ich an Eides statt, dass ich die vorliegende Arbeit selbstständig und ohne unerlaubte fremde Hilfe angefertigt habe, andere als die angegebenen Quellen nicht benutzt und die den benutzten Quellen wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe."

Wolfsburg, den 15.08.2005

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 „XML ist nicht gleich XML“

Sprache dient den Menschen zur Kommunikation. Sie sorgt für einen Austausch von Informationen. Ein ähnliches Ziel verfolgen die so genannten Computersprachen im Bereich der Informatik. Diese Computersprachen sind im Prinzip Programme, mit denen die Menschen Computer zu bestimmten Rechenschritten anleiten. Ähnlich wie bei den menschlichen Sprachen kommt es dabei häufig zu “Übersetzungsproblemen“: Nicht jeder spricht dieselbe Sprache – und auch innerhalb einer einheitlichen Sprachfamilie kann es unter Umständen Missverständnisse oder Anpassungsprobleme geben. So steht die Frage im Mittelpunkt dieser Diplomarbeit, welche Probleme es bei der Verwendung der Programmiersprache Java geben kann – und wie diese zu lösen sind. Dabei werden auch die Unterschiede und Interoperabilitäts-Probleme verschiedener XML-Formate deutlich. Interoperabilität ist dabei ein Ansatz, der eruieren soll, in welchem Maße zwei verschiedene Programme kompatibel sind. In dieser Diplomarbeit soll dieser Vergleich letztlich dazu beitragen - und sich vor allem daran messen lassen -, welche Druckergebnisse sich so etwa bei der Textverarbeitung auf XML-Grundlage erzielen lassen.

Wenn heute in einem Programm die Extensible Markup Language (XML)-Fähigkeit bescheinigt wird, geht der Anwender meist davon aus, dass es kompatibel zu jedwedem Standard wäre. Dieser Glaube ist trügerisch, die Realität sieht meistens anderes aus: Alleine die Vielzahl der so genannten XML-Dialekte und weiterer Programmiersprachen, die zu XML- Familie gehören weisen darauf hin wie unterschiedlich die XML-Ausgaben sein können. XML ist nicht gleich XML. Aus diesem Grund benötigen zwei XML-basierte Prozesse noch ein zusätzliches Übersetzungsprogramm, um sich verständigen zu können. Auch an einer Mensch-Prozess-Kommunikation sind hohe Anforderungen gestellt. Ein XML-Code ist zwar für Maschinen gut lesbar, für den menschlichen Betrachter aber äußerst unkomfortabel. Deswegen ist es eine große Herausforderung, die XML-Informationen in ein Ausgabeformat zu bringen.

Ziel ist es, das Verfahren zu verfeinern und so den Aufwand für mögliche Layout-Änderungen beim Druck von XML-basierten Dokumenten zu verringern. Dazu werden im Folgenden einige grundlegende Schnittstellenprogramme besprochen, die dazu dienen können, dass Java und XML problemloser zusammenarbeiten. In diesem Zusammenhang wird auf die zwei gängigsten, bereits existierenden Verfahrensweisen (siehe Kapitel 5.2) konzentriert und versucht beide zu einem multifunktionalen Verfahren auf der Grundlage von Java und XML zusammenzuführen. Dabei ist der Programmieraufwand bei den bisher gebräuchlichen Verfahren nicht unerheblich. Es soll versucht werden, diesen Aufwand zu minimieren, damit auch weniger versierte User mit dem Programm fehlerlos und ihren Bedürfnissen entsprechend umgehen können.

Trotz verbreiteter elektronischer Dokumentenverarbeitung ist es üblich, die Geschäftsinformation in Papierform zu verarbeiten und zu archivieren. Dabei werden an das Layout immer größere qualitative Anforderungen gestellt: Es soll den Normen entsprechend aufgebaut sein, korrekte Informationen liefern, repräsentativ wirken und natürlich soll das Firmen-Logo exakt entsprechen. Kleine Abweichungen im Druckergebnis können dabei große, negative Folgen haben. Diese Diplomarbeit soll aufzeigen, welche Möglichkeiten es gibt, mit möglichst geringem Aufwand ein stilvolles (Druck-) Ergebnis zu erzielen. Abschließend erfolgt die Entwicklung eines Programms zur Datenverarbeitung.

2 Grundlagen

2.1 Die Programmiersprache Java

2.1.1 Entstehung

Java (ursprünglich: „Oak“) wurde als strukturierte, objektorientierte und plattformunabhängige Programmiersprache entwickelt. Java basiert auf C, C++ Programmiersprachen, bietet hohe Sicherheit bei der Übertragung des Programmcodes durch das Netzwerk, und ist damit ideal für die Verwaltung größerer Projekte sowie zum Einsatz im Internet. Öffentlich wurde Java im Mai 1995 ins Leben gerufen. Im Dezember 1995 gelang Java-Technologie der Durchbruch als Implementierung des Netscape Navigators in Version 2. Die erste Java Development Kit (JDK) kam 1996 heraus und ermöglichte das Programmieren von Java-Applikationen und von Web-Applets. Zurzeit wird Java unter anderem für die Gestaltung von Internet-Applikationen eingesetzt. Die Kontrolle und Weiterentwicklung von Java Standards ist allein der Firma Sun Microsystems (Sun) vorbehalten.

2.1.2 Java Eigenschaften

a) Objektorientierung

Als Konzept der Objektorientierung wird die Vorstellung eines Computerprogramms als Kombination aus getrennten, aber miteinander interagierenden Objekten bezeichnet. Ein Objekt umfasst sowohl Informationen als auch Möglichkeiten, an die Information heranzukommen und sie zu ändern. So bietet Objektorientierung eine einfachere Verwaltung der Softwareprojekte und Wiederverwendung der Softwaremodule.[1]

b) Plattformunabhängigkreit

Die Plattformunabhängigkeit von speziellen Betriebssystemen und Hardware ist unter der Voraussetzung der Integration der Java Virtual Maschine (JVM) im jeweiligen System ebenfalls von großem Vorteil.

Die JVM von Sun Microsystems (Sun) ist Teil der Java Runtime Environment (JRE), der Java Plattform, die auch sämtliche benötigten Java-Klassenbibliotheken enthält. Neben Server- und Desktopbetriebssystemen wie Microsoft Windows, Linux, Solaris oder andere unterstützt JRE

[1] Vgl. Lemany, L./Cadenhead, R (2002), S39.

auch viele eingebettete (Computer-) Systeme wie Mobiltelefone und andere technische Plattformen, die zum Beispiel in Geräten der Unterhaltungselektronik integriert sind. Die möglichen Anwendungsgebiete sind also immens.

c) Sicherheitskonzept

Bei der Kompilierung aus Java Quelltext wird ein Bytecode erzeugt. Dieser Code ist in der Regel maschinenunabhängig und oft im Vergleich zum Quellcode und Maschinencode relativ kompakt. Die Virtuelle Maschine übersetzt Bytecode in Maschinencode für den jeweiligen Prozessor. Gleichzeitig trägt JVM einen wichtigen Teil zum Java-Sicherheitskonzept bei. Dieses Sicherheitskonzept besteht aus einem Dreischichtmodell:

- einem Code-Verifier, der sicherstellt, dass die Virtual Maschine (VM) keinen ungültigen Bytecode ausführen kann,
- einem Class-Loader, der die Zufuhr von Klasseninformationen zum Interpreter steuert,
- einem Security-Manager, der schließlich dafür sorgt, dass der Zugriff nur auf die freigegebene Programmobjekte erfolgt.[2]

[2] o.V.: Java (Programmiersprache), http://de.wikipedia.org/wiki/Java_%28Programmiersprache%29, 05.06.2005.

Um weitere Zusammenhänge besser erläutern zu können, wird nun kurz auf einige XML-Grundlagen eingegangen:

2.2 XML

XML ist ein universelles Format für die Beschreibung von Daten. XML ist ein Standard zur digitalen Erstellung maschinen- und menschenlesbarer Dokumente. Die XML- Sprache ist eine Weiterentwicklung von SGML. Sie definiert die Regel zum Aufbau aller auf der Baumstruktur basierenden Dokumente und bildet damit die Struktur von allen Auszeichnungssprachen ab. XML-Dokumente können zum Beispiel: Text, Grafiken, Videos, Präsentationen, Musik und andere Multimedia-Komponenten beschreiben.

2.2.1 Der XML-Dokumentaufbau

Jede XML-Datei entspricht einer Baumstruktur: Es gibt genau ein Wurzelelement. Die Verschachtelung der Elemente bildet dabei die Baumstruktur. Attribute sind den Elementen als Knoten untergeordnet. Elementwerte und die Werte der Attribute bilden schließlich die Blätter des Baums.

a) Element

Jedes Element besteht aus einem öffnenden und einem schließenden Tag. Dazwischen befindet sich der Elementinhalt.

Beispiel:

```
<name> Beispiel</name>     -  Element
Starttag  Inhalt(Text)  Endtag
```

b) Attribute

Neben dem Inhalt kann jedes Element beliebige viele Attribute enthalten. Sie werden in dem öffnenden Tag eingetragen. Jedes Attribut muss einen Wert in Anführungszeichen haben

Beispiel:

```
< bild quelle="bild.jpg"/>
        Attributname Attributwert
```

c) Verarbeitungsanweisungen

Verarbeitungsanweisungen werden definiert und befinden sich im Dokumentkopf.

Beispiel:

```
<?Ziel-Name Parameter ?>
```

d) Kommentare

Kommentare werden mit Zeichenfolge abgegrenzt. Sie können an jeder beliebigen Stelle im XML-Dokument stehen, dürfen aber nicht innerhalb von Tags erscheinen.

Beispiel:

```
<!- -.Kommentar- ->
```

2.2.2 XML-Deklaration

Mit dieser Deklaration wird eine Verbindung zwischen den XML-Daten und der DTD hergestellt. Die DTD kann in einer Datei stehen (externe DTD) oder Bestandteil der XML-Datei sein (interne DTD). Die kombinierte Verwendung zwischen externen und internen DTD-Deklarationen ist möglich. Die öffentlichen Deklarationen verweisen auf DTDs, die durch einen Standard festgelegt sind.

Beispiel:

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

2.2.3 Dokumententyp-Definition (DTD)

Durch die DTD wird die Form des Inhalts beziehungsweise die Erweiterung einer XML-Datei festgelegt, zum Beispiel: Welche Elemente dürfen verwendet werden, welche Werte können Elemente und Attribute annehmen, welche Elemente und Attribute sind vorgeschrieben oder müssen einen Wert haben? So können DTD benutzt werden, um die Namen für die Elemente einer XML-Datei festzulegen. Der Inhalt, die Verschachtelung und Häufigkeit der Elemente können ebenfalls festgelegt werden. Die DTD können extern und intern deklariert werden. Dabei erweitert eine interne DTD die externe DTD und überschreibt sie nicht. Eine Verwendung

von DTD ist nicht immer notwendig, außer wenn die Daten von Hand erfasst werden oder ausgetauscht werden müssen, ist eine DTD unbedingt erforderlich.

Vorteile:

- DTD ist eine ausgereifte und getestete Technologie
- DTD ist ein fester Bestandteil des XML-Standard
- Fast alle XML-Tool unterstützen die Verwendung von DTD
- Fast alle XML-Applikationen werden durch DTD festgelegt

Nachteile:

- DTD verwendet eine andere Syntax als die XML-Datei. Das heißt, dass Parser und Benutzer die Syntax von zwei verschiedenen Sprachen kennen müssen.
- Einsetzung eingeschränkter Datentypen, es sind zum Beispiel keine speziellen Datentypen wie Zahlen oder Daten möglich.
- DTD bietet wenig Möglichkeiten, eigene Datentypen zu definieren

2.2.4 XML-Schema als Alternative zur DTD

Das XML-Schema ist eine Applikation zur Beschreibung der Struktur von XML-Dateien. Das XML-Schema bietet auch wie DTD die Möglichkeit, den Inhalt von Elementen und Attributen zu beschränken. Die Entitäten-Definition wird dagegen vom XML-Schema nicht unterstützt. Sie unterstützt aber die gängigen Programmiersprachen und kann vorhandene Datentypen mit eigenen kombinieren.

Ebenfalls können Datentypen auf Basis vorhandener oder eigener Datentypen erweitert oder eingeschränkt werden. Komplexe Typen können Attribute, andere Elemente oder Text als Inhalt enthalten. Die simplen Typen enthalten als Inhalt nur Text. Die Beschreibung in einem Schema ist länger als in einer DTD - aber strukturierter und detaillierter. Die Verknüpfung mit der XML-Datei erfolgt auf zwei Weisen: Als eine Schema-Instanz oder mit dem Attribut xsi-schemaLocation kann eine Schema-Datei angegeben werden.[3]

[3] Vgl. Freund+Dirks S.1-15

Beispiel:

```
< adressbuch
xmlns:xsi=“http://www.w3.org/2001/XMLSchema-instance“
xsi:no NamespaceSchemaLocation=“adresse1.xsd“>
```

2.2.5 Markup

Das Markup ist ein Bestandteil des XML-Textes und wird in der Regel durch <-Zeichen und >-Zeichen oder durch &-Zeichen und Semikolon abgegrenzt. Das Markup zwischen < > Zeichen wird als Tag bezeichnet. Der Begriff XML-Text setzt sich nicht allein aus Zeichendaten, sondern aus einer Kombination von Zeichendaten und Markup zusammen.

2.2.6 Wohlgeformtheit und Gültigkeit

Eine XML-Datei ist wohlgeformt, wenn am Anfang die XML-Deklaration steht, es genau einen Wurzelelement gibt und Elemente korrekt verschachtelt sind.

Eine XML-Datei ist gültig, wenn sie wohlgeformt ist und die Struktur einer formalen Beschreibung folgt. Diese formale Beschreibung kann eine DTD sein. Andere Möglichkeiten zur Strukturbeschreibung sind XML-Schema (siehe oben) und RELAX (REgular LAnguage XML), also eine Strukturbeschreibung durch reguläre Ausdrücke.

2.2.7 Prinzip der Trennung von Inhalt, Struktur und Design

Das Prinzip der Trennung von Inhalt, Struktur und Design wird angewendet, wenn viele gleichartige Daten dargestellt werden. Damit eine eindeutige Zuordnung möglich ist, muss die Struktur der Daten eindeutig sein.

2.3 XSL

XSL ist eine der Hauptanwendungen von XML. XSL wird für umfangreiches Umformen, Sortieren und Auswählen von Elementen auf dem Server verwendet. Damit die XML-Dateien mit Hilfe der Stylesheets dargestellt werden können, müssen die beiden verknüpft werden. Die Verknüpfung geschieht durch eine Ausführungsanweisung in Prolog der XML-Datei.

Beispiel:

```
<?xml-stylesheet href="URI" type=" text/xsl"?>
```

Durch das Verknüpfen einer Stylesheet mit einer XML-Datei ist eine formatierte Ausgabe der Inhalte in einem modernen Browser möglich.
XSL gliedert sich in drei Bereiche:

- XSLT: Dient zum Umwandeln von XML-Dokumenten in andere textbasierte Formate
- XSL-FO: Dient der Beschreibung von Seiten für die Umwandlung in PDF, RTF.
- XPath: Dient zum Selektieren von Elementen einer XML-Datei.[4]

2.4 XSLT

Die XSLT (Extensible Stylesheet Language Transformation) macht das Umwandeln von XML-Dateien in andere textbasierte Dateien möglich. Für die Transformation sind die Eingaben einer XML-Datei und eines Stylesheets (XSL/ XSLT Stylesheet Datei) notwendig. Die Ausgabe erfolgt als beliebige Textdatei zum Beispiel HTML, XHTML, XSL-FO, SVG, SMIL oder DocBook.[5]
Dieses Verfahren nennt man Präsentation Oriented Publishing (POP). Ein weiteres Anwendungsziel ist Message Oriented Middlewar (MOM). Dabei wird die Transformation zum Zwecke des Datenaustausches verwendet. Aufgrund der vielen XML-Dialekte wird XSLT als Übersetzer von einer Sprache in die andere genutzt, da die Systeme für den Datenaustausch eine einheitliche XML-Sprache benutzen müssen.[6]

[4] Vgl. Freund+Dirks S.1-16
[5] Vgl. Freund+Dirks, S. 1-15
[6] „o.V.“, XSL Transformation, http://de.wikipedia.org/wiki/XSLT, 13.08.2005

2.4.1 Arbeitsweise eines XSLT-Prozessors

Der XSLT-Prozessor liest die gegebene XML-Datei, parst sie und stellt sie in einer Baumstruktur dar. Diese Struktur nennt man Eingabebaum. Anschließend liest und analysiert der XSLT-Prozessor die Stylsheet-Datei und stellt dort die festgelegten Transformationselemente ebenfalls in einer Baumstruktur dar. Diese Struktur nennt man Transformationsbaum. Das Ergebnis der Umformungen wird im so genannten Ergebnisbaum gespeichert und schließlich in eine neue XML-Datei ausgegeben.[7]

2.5 XSL-FO

XSL-FO (Extensible Stylesheet Language-Formating Objects) bietet eine Umwandlung in beliebige Dateiformate wie PDF und RTF. Dabei werden XML-Datei und Stylesheet in eine spezielle Form überführt (XSLT). Mit Hilfe von Format-Objekten werden das Aussehen, die Seitenaufteilung und sonstige Formatierungsmerkmale genau festgelegt. So wird die Trennung zwischen Darstellung, etwa durch XSL-FO, und Logik (XSLT) ermöglicht. Dies ist in den Textverarbeitungssystemen zu Gunsten einfacher Handhabung nicht möglich.
Zur Generierung der Printmedien stehen die FO-Prozessoren zur Verfügung. FOP ist eine Java-Anwendung aus dem Apache-Projekt, sie kann aus XSL-FO-Daten PDF-Dokumente, Postscript- und PLG-Dateien ausgeben (siehe Kapitel 3.1).[8]

2.5.1 Die Arbeitsweise des XSL-FO Prozessors

Den Ausgangspunkt für ein mit Hilfe von XSL erstelltes PDF-Dokument bildet eine XML-Datei. Aus ihr werden im ersten Schritt durch gezielte Transformationen die XML-Elemente ausgewählt, die im PDF-Dokument dargestellt werden sollen. Gleichzeitig werden alle erforderlichen XSL-FO-Elemente ergänzt, die Objekte (beispielsweise Seitengestaltung, Gestaltung von Spalten, Listen, Tabellen, ...) erzeugen. Das Ergebnis ist eine FO-Datei. In einem zweiten Schritt erzeugt ein geeigneter Formatierer (auch FO-Prozessor genannt) die gewünschte PDF-Datei.[9]

---

[7] Vgl. Mistlbacher A./Nussbaumer A. (2002), S. 377
[8] "o.V.", Extensible Stylesheet Language - Formatting Objects, http://de.wikipedia.org/wiki/XSL-FO, 12.08.2005
[9] "o.V.", Extensible Stylesheet Language - Formatting Objects, http://de.wikipedia.org/wiki/XSL-FO, 12.08.2005

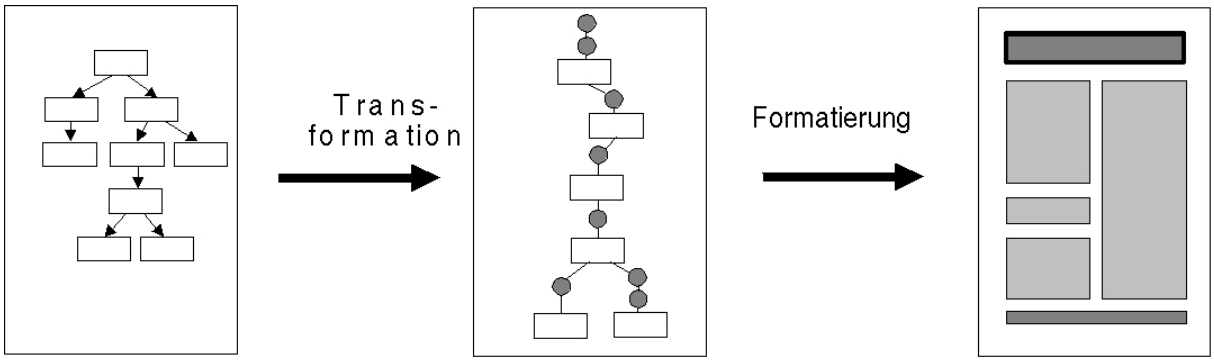

Abbildung1: Transformation mittels XSL-FO Prozessors
Quelle: Extensible Stylesheet Language - Formatting Objects, http://de.wikipedia.org/wiki/XSL-FO, 12.08.2005

2.6 XPath

XPath ist eine XML-Anwendung, welche die Knoten einer XML-Baumstruktur adressieren kann. Zusätzlich bietet XPath logische Ausdrücke und zusätzliche Funktionen, wie Bedingungen und Funktionen, die etwas über die Position aussagen. Die einfache Form der Adressierung orientiert sich an der Verzeichnisnotation, die eine einfache Form des Zugriffs auf die Knoten bietet. Durch die längere Form in XPath ist ein spezielles Auswählen von Knoten und Knotenmengen möglich. XPath-Operationen bieten viele Möglichkeiten für die Verarbeitung von XML-Dateien mit XSLT.[10]

[10] Vgl. Freund+Dirks

2.7 XML-Java Schnittstellen

Um eine XML-Datei bearbeiten zu können, benötigt man einen XML-Parser, der XML einliest und ihre Struktur über die XML-Programmierschnittstellen den anderen Programmen zur Weiterverarbeitung gibt.

2.7.1 Der XML-Parser

Ein XML Parser ist ein Programm, das die XML-Dokumente einliest und die im Dokument enthaltenen Daten für jede Applikation, die diesen Parser benutzt, zur Verfügung stellt. Damit bildet ein Parser eine zentrale Komponente in einer XML-fähigen Applikation. Beim Lesen der XML-Datei stellt der Parser sicher, ob das Dokument wohlgeformt und valide ist. Geschwindigkeit und Konformität der XML-Spezifikation sind dabei entscheidende Kriterien zur Wahl des Parsers. Besonderes bei der Verarbeitung der großen XML-Dokumente ist die Geschwindigkeit entscheidend für die Gesamtleistung des Systems. Zum Teil wird dabei die Konformität zu Gunsten einer höheren Leistung eingeschränkt. Die bekannten Parser-Prozessoren sind Apache Xerces, XP von James Clark, ProjecX von Sun und IBM XML4J.

2.7.2 SAX

Dieses und das folgende Kapitel vergleichen die beiden Schnittstellenprogramme SAX und DOM hinsichtlich ihrer Vor- und Nachteile für die XML-Bearbeitung und der Java-Anbindung. Die Simple API for XML (SAX) ist ein Standard-Programmierschnittstelle für XML-Parser. Der SAX-Parser arbeitet das XML-Dokument sequentiell (linear), Zeile für Zeile durch und ruft die definierten Ereignisse (Starttag, Endtag…) auf (siehe Tabelle1).

Vorteile:

- SAX bietet weitgehend Implementierungsunabhängigen Zugriff auf XML-Parser verschiedener Hersteller.
- SAX ist schnell und benötigt kaum Zeit für das Parsern, weil die Daten direkt an die Applikation geliefert werden.
- Da die Daten im Speicher nicht lange gehalten werden, benötigt SAX kaum Speicherkapazität und ist zur Verarbeitung großer Datenmengen geeignet.
- Wenn der XML-Reader installiert ist, lassen sich mehrere XML-Dateien hintereinander parsern.

Nachteile:

- Der Inhalt der bisher geparsten Datei wird nicht gespeichert und ist nach dem Parsern nicht mehr zugänglich.
- Parsern von mehr als einer Datei zu gleicher Zeit ist dagegen nicht möglich.
- Weil das Bewegen zwischen den Dokumentebenen nicht möglich ist, ist SAX für die XSL-Transformationen nicht geeignet.
- SAX ist aufwendig zu handhaben. Zum Beispiel führt die Verwendung der Locator-Instanz außerhalb der Gültigkeitsbereiches der Content-Handler-Implementation zu fehlerhaften Informationen und zur Verzehrung der geparsten XML-Dokumente.[11]

2.7.3 DOM

Das Document Object Model (DOM) ist eine Programmierschnittstelle und beschreibt, wie man unabhängig von speziellen Programmiersprachen auf HTML- oder XML-Dokumente zugreifen kann. Das DOM-Model besteht aus Knoten, die Dokumente, Elemente und Attribute enthalten, die in einer Baumstruktur durch Zeiger miteinander verbunden sind (siehe Tabelle1). DOM wurde zum Zwecke der vereinfachten Internetprogrammierung mit verschiedenen Browsern und Scriptsprachen entwickelt. Dom ist in Level eins bis drei und vom World Wide Web Consortium (W3C) standardisiert.[12]

[11] McLaughlin B.(2001),S.50-63

[12] Document Object Model, http://de.wikipedia.org/wiki/Document_Object_Model, 11.08.2005

Vorteile:

- DOM bietet ebenfalls weitgehend implementationsunabhängigen Zugriff auf XML-Parser verschiedener Hersteller.
- DOM baut im Speicher einen DOM-Baum auf und bietet damit Zugriff auf die Dokumentinhalte.
- DOM erlaubt die Bewegung zwischen den verschiedenen Dokumentebenen und ist für XSL-Transformationen geeignet.

Nachteile:

- Dom ist langsam und nur zum Parsern von XML-Dateien geeignet, die in den Arbeitsspeicher passen.
- DOM verbraucht viel Speicherkapazität.
- DOM ist schwer in der Handhabung[13]

DOM	SAX
<?xml version="1.0" encoding="UTF-8"?> <dokumet> <erstelldatum>14.07.2005</erstelldatum> <rechnungsnummer>34544/356</rechnungsnummer> <kunde> <kundennummer>35567645398</kundennummer> <name>Meier</name> <vorname>Dirk</vorname> <bestelldatum>12.07.2005</bestelldatum> </kunde> </dokumet>	startElement: dokumet startElement: erstelldatum characters: 14.07.2005 endElement: erstelldatum startElement: rechnungsnummer characters: 34544/356 endElement: rechnungsnummer … usw. endElement: dokumet

Tabelle 1: Vergleichende Darstellung der gleichen XML-Datei in DOM und SAX

2.7.4 JAXP: „Java API for XML Parsing“

JAXP ist eine Hilfs-Schnittstelle zwischen Java-Programmcode und XML-Schnittstellen (SAX, DOM). JAXP bildet eine Abstraktionsschicht, um die Erzeugung einer SAX- oder DOM-Parser-Instanz zu kapseln. Mit Verwendung von JAXP fallen das aufwändige Importieren und Referenzieren der Parser-Klassen aus dem Java-Code des jeweiligen Herstellers weg. Dieses ermöglicht den beliebigen Austausch zur Laufzeit. Es gibt jedoch einen entscheidenden

[13] McLaughlin B.(2001),S.184-206

Nachteil von JAXP: Jeder Parser, der JAXP unterstützt, muss eine Implementierung von JAXP-Klassen liefern, weil vier der sechs JAXP-Klassen abstrakt sind.[14]

2.7.5 JDOM

Die Vorteile der Plattformunabhängigkeit und Sprachenneutralität können durchaus auch in Nachteile umschlagen: Der Grund dafür liegt in der aufwändigen Handhabung von SAX und DOM aus der Applikation heraus. Daneben erweisen sich die herstellerspezifischen Implementierungen der SAX- und DOM-Interfaces an den XML-Parser als äußerst aufwändig. Aus diesem Grund ist ein Werkzeug nötig, das die Vorteile der Zielsprache möglichst konsequent nutzt.

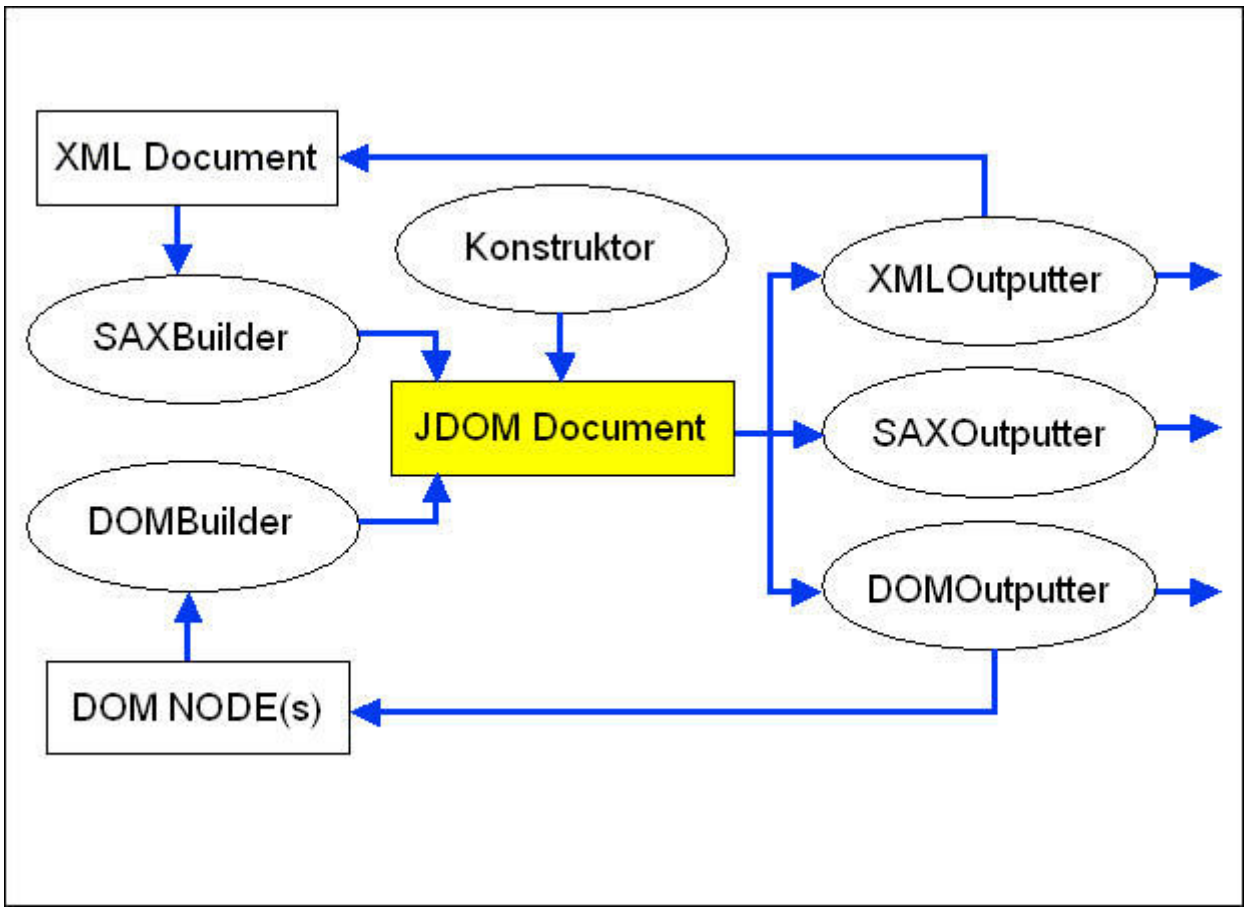

Abbildung 2: JDOM

Quelle: Hachmann B., Das JDOM Projekt, http://www.jdom.net/dom3.htm

JDOM ist eine vollständig Java 2-basierte Schnittstelle zur Arbeit mit XML. JDOM ist ein selbstständiger OpenSource-Projekt. Es basiert weder auf DOM noch auf SAX.

[14] McLaughlin B.(2001),S.208

Vorteile:

- Das XML-Dokument wird ähnlich wie in DOM als Baum im Hauptspeicher repräsentiert (siehe Tabelle 1).
- JDOM erlaubt die Bewegung zwischen den Dokumentebenen und ist für XSL-Transformationen geeignet.
- Gleichzeitig liefert es die gleiche hohe Performance wie SAX und ermöglicht ein zügiges Parsern sowie sehr schnelle Ausgaben.
- JDOM ist namensraumfähig, unterstützt DTD und die XML-Schema-Validierung.
- JDOM bietet eine komfortable IO-Schnittstelle, die XML-Dokumente nicht nur als Textdokument ausgeben kann, sondern sie auch in Form von SAX-Ereignissen und in einem DOM-Baum konvertieren kann (siehe Abbildung: JDOM). Für Ein- und Ausgaben benutzt es Java-Klassen.
- Weitere Vorteile der Java Programmierung:
  Ein XML-Dokument kann wie jeder andere Dokumenttyp als Einheit betrachtet werden.
  JDOM-Konstrukte werden durch eine direkte Objekt-Instanziierung erzeugt.
  Java-Methoden können für das Erzeugen, Entfernen und Verändern von XML-Konstrukten benutzt werden.[15]

Nachteil:

- JDOM ist keine Universelle Schnittstelle

JDOM ist in der Handhabung für den beschriebenen Einsatzfall einfacher als SAX und DOM.

[15] McLaughlin B.(2001),S.213-238

3 Die Alternativen

In diesem Kapitel werden einige XML-Tools gegenübergestellt. Diese Programme wurden mit dem Ziel entwickelt, ihren Anwendern das Design der XML-basierten Dokumente einfach und komfortabel zu gestalten. Im Weiteren wird diskutiert, in wie weit diese Tools zum Lösen der Aufgabenstellung beitragen können. Die Auswahlkriterien dabei sind: Aufwand der Dokumenterstellung, Vorkenntnisse der Anwender, erzielte Ergebnisse - und natürlich der Preis.

3.1 Das Apache-XML-Project

Man benötigt eine Menge unterschiedlicher Werkzeuge, um die XML-Dokumente in ihren Lebenszyklus zu begleiten. Angefangen mit der Erstellung über die Weiterreichung, das Einlesen, die Transformation und die Bearbeitung, bis zur Ausgabe für jeden Lebensabschnitt des XML-Dokumentes benötigt man ein so genanntes Werkzeug. Diese Werkzeuge kommen erst dann zur Geltung, wenn sie aufeinander abgestimmt sind. So vereint das Apache-XML-Project alle erforderlichen Tools zur Bearbeitung und Veröffentlichung von XML-basierten Inhalten. Aktuell umfasst das Projekt sieben Einzelprojekte. Diese sollen hier kurz aufgelistet werden. Dabei sind für diese Untersuchung Xerces, Xalan und FOP als Bestandteile des XML-Publishing-System-Cocoon von besonderem Interesse:

1. Xerces: XML-Parsers (Perl und COM-Bindings)
2. Xalan: XSLT-Stylesheet-Prozessor
3. Cocoon: XML-basiertes Web-Publishing
4. FOP: XSL-Formatierungsobjekte
5. Xang: Entwicklungsumgebung für dynamische Server Pages
6. SOAP: Simple Object Access Protocol
7. Batik: SVG-Toolkit

Zentrale Basistechnologien bilden der Parser Xerces und der XSLT-Prozessor Xalan. Insbesondere ermöglichen sie dem XML-Publishing-System-Cocoon, XML-Dokumente in diverse Zielformate wie HTML, RTF und PDF zu transformieren. Für das Erzeugen von PDF-Dokumenten greift Cocoon auf FOP zurück. Unter anderen kann Xalan mit SAX-(Level1) und

DOM-(Level2)-konformen Parsern zusammenarbeiten. Der Xalan-Prozessor ist in Java und C++-Version erhältlich und bietet ein hohes Maß an Stabilität für den professionellen Einsatz.[16]

3.1.1 Architekturübersicht Cocoon

Als nächstes wird am Beispiel der Apache XML Project Coccon die Darstellung der XML-basierten Dokumente vorgestellt.

Cocoon ist ein Framework mit einer offenen, komponentenbasierten Architektur. Diese Architektur gründet sich auf mehrere unabhängige, austauschbare Java-basierte Apache- XML-Projekte. Dieses Framework setzt Apache Xerces als XML -Parser, Apache Xalan als XSLT-Processor und Apache FOP zur PDF-Generierung ein. Genauso könnten aber auch andere XML-Parser oder XSLT-Prozessoren verwendet werden. Grundlage für die objektorientierte Architektur ist das Apache-Projekt Avalon, ein Java-Ansatz zur Software-Entwicklung (siehe Abbilbung 3). Cocoon kann in verschiedenen Umgebungen verwendet werden. Zum Beispiel kann Cocoon als Servlet in Servlet-Engines oder Application-Server eingebunden werden. Cocoon kann aber auch ein Skript aus der Kommandozeile ansteuern. Die Arbeitsweise des Cocoons ist umgebungsunabhängig.

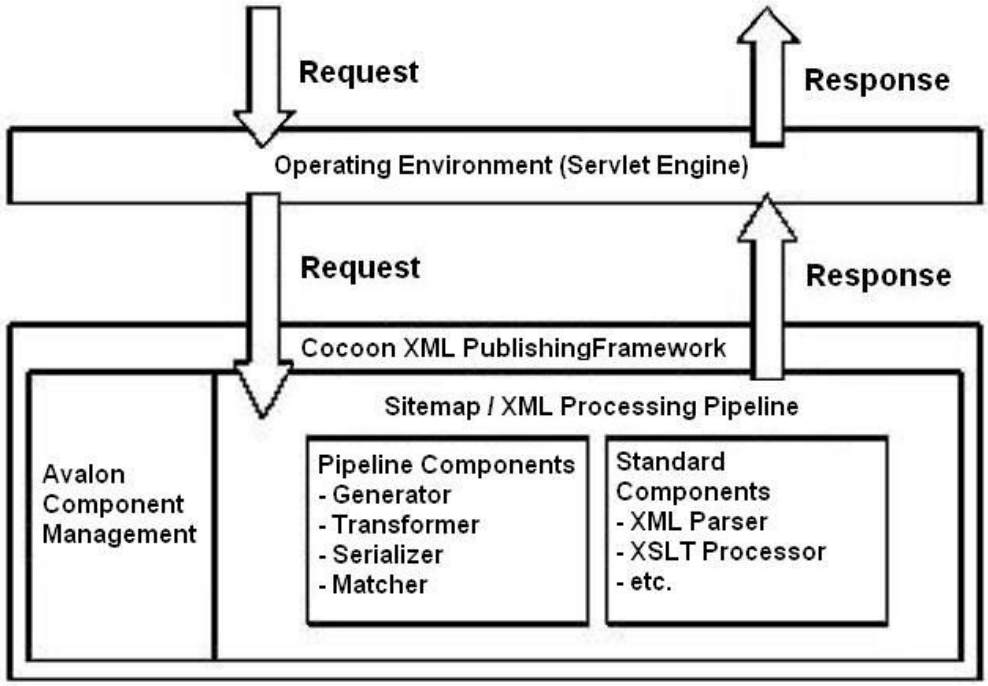

Abbildung 3: Cocoon-Architektur
Quelle: Langham M / Ziegeler C, Die Verpuppung, XML-Anwendungen erstellen mit dem Coccon.2, Teil1, http://www.javamagazin.de/itr/online_artikel/psecom,id,154,nodeid,11.html, 03.07.2005

[16] "o.V." XML Werkzeugkisten ,http://www.tecchannel.de/entwicklung/programmierung/401578/index9.html, 11.08.2005

Sollte Cocoon als Servlet eingesetzt werden, so antwortet er auf die Anfrage des Clients mit einer Antwort oder einem Dokument. Die Sitemap ist ein Bindeglied zwischen dem Dokumentnamen und der Antwort. Die Sitemap stellt ein XML-Dokument dar, in dem festgelegt wird, welche Arbeitsschritte für eine URI ausgeführt werden. Eine Darstellung der gesamten Funktionsweisen von Coccon2 würde den Rahmen dieser Diplomarbeit jedoch sprengen. Aus diesem Grund sind im Folgenden nur die in Coccon enthaltenen Projekte in Verbindung mit der Dokumentgenerierung vorgesehen.

Ein Dokument wird in der XML-Pipeline erzeugt. Die XML-Pipeline wird aus mehreren Komponenten wie Generator, Transformer, Serializer und Matcher zusammengestellt. Jede Pipeline beginnt mit einem Generator, dessen Aufgabengebiet die Umwandlung des Datenstroms aus verschiedenen Formaten nach XML ist. Als Datenquelle dienen also nicht nur XML-Dateien, sondern auch andere Daten, die ebenfalls transformiert werden können. Der Speicherort der Daten ist für Cocoon ebenfalls nicht von Bedeutung, da es die XML-Dateien von der Festplatte lesen oder von einer http-Verbindung laden kann. Als nächstes wird der XML-Datenstrom von einem XSL-Transformer verarbeitet und zur Output-Zwecken zum Serializer weitergeleitet. Der Serializer ist die letzte Komponente in der Pipeline und wandelt den XSLT-Baum in ein Ausgabeformat um. Dazu verwendet Coccon dem Ausgabeformat entsprechend HTML-, PDF- oder RTF-Serializer.[17]

Cocoon benutzt den klassischen Weg der XML-Transformation und Layout-Gestaltung mittels XSLT- und XSL-FO-Prozessoren. Dieser Weg ist weit verbreitet, wird in kommerziellen Lösungen sowie in OpenSource weltweit eingesetzt und nutzt die Vorteile der XML-basierten Dokumenterstellung. Die Trennung von Daten, Logik und Layout hat hierbei allerdings einen wesentlichen Nachteil: Sobald ein neues Feld dem Layout zugefügt oder das Feld Platzierung verändert werden soll, muss bei jeder Layout-Änderung ein neues Stylesheet geschrieben werden. Das setzt wiederum nicht unwesentliche Programmierkenntnisse voraus.

[17] Langham M / Ziegeler C., Die Verpuppung, XML-Anwendungen erstellen mit dem Coccon.2, Teil1, http://www.javamagazin.de/itr/online_artikel/psecom,id,154,nodeid,11.html, 03.07.2005

3.2 „XSL TemplateDesigner“

Der Gedanke von „XML zum PDF“ zu kommen ist nicht neu. Es gibt zahlreiche kommerzielle und OpenSource-Lösungen zu diesem Konzept. So hat die Firma Antenne House eine Erweiterung des bisherigen Produktes „XSL TemplateDesigner“ zur “XSL Report Designer“ auf den Markt gebracht. Dieses Software-Produkt ermöglicht die Gestaltung von XML-Formularen unter der Windows-Bedienoberfläche mit einem umfangreichen Werkzeugtool. Intern transferiert der Report Designer das Formular-Layout und XML-Formular-Inhalte zu XSL-FO Daten. Der XSL-Formater erzeugt daraus druckbare Ausgaben, die beliebig viele Seiten umfassen. Siehe (Tabelle 3) Der Komfort dieses Tools hat auch seinen Preis – er schlägt mit ca. 810,00 € zu Buche.

3.3 Crystal Reports

Die Crystal-Reports (ein Bestandteil der Visual Studio.NET) unterstützten ein leistungsfähiges Berichtsdesign aus eigenen Datenquellen sowie aus OLAP- und XML- Daten. Die Layouts können mit mehr als 100 Formatierungsoptionen erstellt werden. Die Anwender können in den Reports Drill-downs auf Detaildaten nutzen, Informationen sortieren oder filtern, Berichte aktualisieren, Reports drucken und sie in viele Formate wie PDF, Excel und Word exportieren.

Da die MS Office Software eine weltweite Verbreitung hat, könnte man die Vorteile des Crystal-Reports für Windows- und Web-entwickelte Software nutzen. Als integrierten Bestandteil der VB ab Version 5, der VisualStudio und Visualstudio.NET(VS.NET) entstehen keine zusätzlichen Kosten für die Anschaffung dieser Entwicklungsumgebungen. Spätestens im VisualStudio kann Crystal-Reports sprachenunabhängig eingesetzt werden, da VS.NET die Sprachen VB.NET, C++ und J++ unterstützt. Den Softwareentwicklern wird so übrigens das Erlernen einer neuen Programmiersprache erspart. Zudem können die vertraute Toolbox und die Entwurfsansicht als weitere Vorteile gelten. Zudem ähnelt die Berichterstellung in Crystal-Reports sehr der Vorgehensweise unter dem verbreiteten MS Access. Neben den MS Office internen Dateiformaten ist die Ausgabe in PDF-, HTML-, DHTML -Formaten möglich. Die erstellten Berichte lassen sich aktualisieren, nachbearbeiten und ermöglichen in Zusammenhang mit Crystal Enterprise den Zugriff für mehrere Benutzer. Benutzerfreundliche Berichtserstellung mit Crystal-Reports kann man als kommerzielle Lösung der gestellten Aufgabe annehmen, wenn

der Benutzer bereit ist, in diese Software zu investieren. Die OpenSource Softwarelösung ist dagegen zwar kostenfrei, setzt aber nicht unerhebliche Programmierkenntnisse voraus.[18]

3.4 Erstellung von XML-basierten Dokumenten mit iText

iText ist beispielsweise eine leistungsfähige Bibliothek zur Erstellung von PDF-Dateien. Sie steht unter LGPL- oder MPL-Lizenz. Das Erzeugen von PDF-Dateien mit iText erfolgt in sechs Schritten: Dokument anlegen, PDF-Writer zuordnen, Metadaten schreiben, Dokument öffnen, Seiten erzeugen und Dokument schließen. Der iText-Bibliothek liegt ein Modell zugrunde, das der SAX-Seitenerzeugung ähnlich ist. Dazu verwendet iText ein lineares Füllmodel, bei dem die Seiten nacheinander aufgebaut werden und jeweils nur die aktuelle Seite mit Text oder anderen Elementen gefüllt wird. Ein Löschen und Editieren von Elementen ist später nicht mehr möglich. Aus diesem Grund ist die Erstellung des Dokumentes auf traditionelle Art und Weise (XSLT und XSL-FO) sehr aufwändig. Erwähnenswert ist auch, dass iText-Dokumente von Java-Entwicklern erzeugt werden müssen. Das bedeutet, dass das Dokumentlayout (zum Beispiel Seitengröße, Paragraphen, Textfelder) fest in Java Code programmiert wird.

Ein komfortableres Parsern der XML-Dateien und Füllen der iText-Dokumente bietet die iText-Klasse *com.lowadie.text.xml.XmlParser*. Sie bietet einen Konvertierungsmechanismus, in dem er iText-spezifische Tags zur Abbildung von XML-Tags zur Verfügung stellt (siehe Tabelle 2). Eine genaue Definition der iText-Tags befindet sich in der gleichnamigen DTD-Datei.[19]

xml2pdf.xml	tagmap.xml
<?xml version="1.0" ?> *<DOKUMENT>* *<KAPITEL>* *<TITLE>Kapitel</TITLE>* *<NEWLINE />* *<TEXT>Dieser Text wurde mit* *<BOLDTEXT>iText</BOLDTEXT>* *in das PDF-Format konvertiert.* *</TEXT>* *</KAPITEL>* *<KAPITEL>* *<TITLE>Kapitel</TITLE>* *<NEWLINE />* *<TEXT>Noch mehr Text ...</TEXT>* *</KAPITEL>* *</DOKUMENT>*	<tagmap> <tag name="paragraph" alias="TEXT"> <attribute name="leading" value="14" /> <attribute name="size" value="10" /> </tag> <tag name="chunk" alias="BOLDTEXT"> <attribute name="style" value="bold" /> </tag> <tag name="itext" alias="DOKUMENT" /> <tag name="newline" alias="NEWLINE" /> <tag name="title" alias="TITLE"> <attribute name="size" value="24" /> </tag> <tag name="chapter" alias="KAPITEL"> <attribute name="numberdepth" value="2" /> </tag> </tagmap>

[18] „o.V." (2005): Crystal Reports, http://www.softguide.de/prog_m/pm_0989.htm,11.08.2005

[19] Jastram C., Java meets PDF, http://www.javamagazin.de/itr/online_artikel/psecom,id,441,nodeid,11.html, 11.08.2005

Tabelle 2: Vergleich zwischen iText-spezifische- und XML-Tags

Die Konvertierung von Dokumenten erfolgt mit einer *XmlParser.parse- (document, "xml2pdf.xml", "tagmap.xml")* Funktion. Dafür benötigt man eine zu konvertierende XML-Datei (hier ist es *"xml2pdf.xml"*, eine iText-spezifische XML-Layoutdatei (*"tagmap.xml")* und ein neu zu erzeugendes Dokument (*document*). Besonders bei großen Dateien bietet diese Art der Konvertierung eine bessere Orientierung innerhalb des Codes und erhöht die Wartbarkeit der Dokumente. Die Beschaffung von iText ist kostenfrei. Die Entwicklung mit iText erfordert allerdings die Einarbeitung in Java-Programmierung und die Auseinandersetzung mit der iText-Syntax. Somit ist dieses Programm nicht für einen gewöhnlichen Anwender geeignet.

3.5 WH2FO

WH2FO ist eine Java-Anwendung, ein OpenSource-Projekt von Fabio Giannetti. WH2FO erzeugt aus einem in Word 2000 erstellten und als HTML gespeicherten Dokument eine XML-Datei und eine XSLT-Datei. Diese werden dann weiter auf dem üblichen Wege mittels XSLT-Prozessor wieder zur HTML- und Text-Dateien transferiert oder mittels XSL-FO-Format in PDF-Dateien umgewandelt (siehe Abbildung 4)[20]

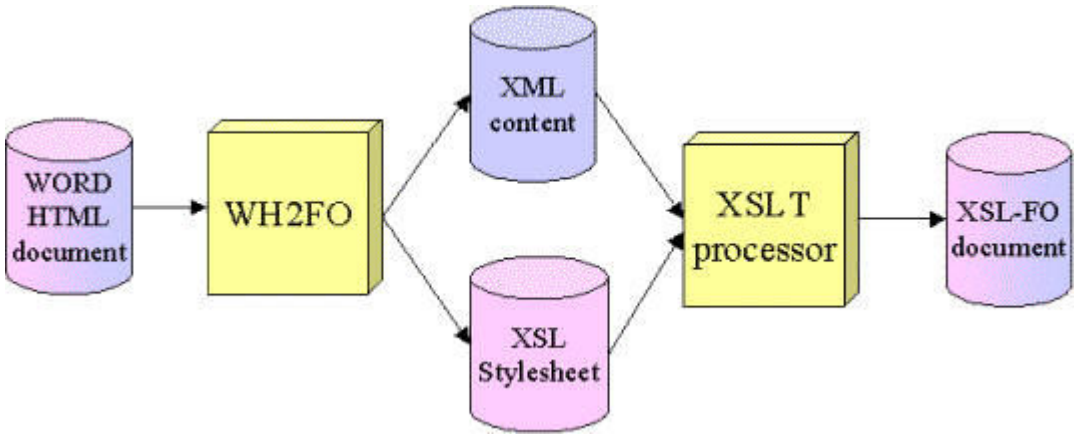

Abbildung 4: WH2FO

Quelle: Word HTML 2 Formatting Objects, http://wh2fo.sourceforge.net/, 11.08.2005

[20] Vgl. Giannetti F., Word HTML 2 Formatting Objects, http://wh2fo.sourceforge.net/, 11.08.2005

Die Anwendung erzeugt automatisch drei Dateien (*.xml, *.xsl, *Atts.xsl), die den gleichen Namen wie die Eingangs-HTML-Datei tragen. Dabei werden in der XSL-Datei nur der Grundbaum des XSL-Dokuments dargestellt und die Attribute in der Datei Atts.xsl ausgelagert. So wird der Inhalt des ursprünglichen Dokumentes mit XML-Tags umrahmt, die einzelne Absätze und Tabulatorstellungen darstellen. Die Transferierung von Bildern und des Textes, Tabellen, Kopf- und Fußnoten sowie Links wird ebenfalls unterstützt. Dagegen gehen vordefinierte, nicht ausgefüllte Felder sowie alles Unsichtbare bei der Transformation verloren. Dabei entstandene XML- und XSLT-Dateien sind starr und unveränderbar. Somit bietet WH2FO nur eine eingeschränkte Konvertierung statischer Worddokumente zum PDF.

3.6 pcwSxwExport

Ein weiteres interessantes Programm ist das pcwSxwExport aus der Zeitschrift PC-Welt. Dieses Programm ermöglicht das Unwandeln von OOo-Writer- und XML-Dateien (DocBook) und MS-Word-Dateien in HTML, PDF und RTF. Außerdem wandelt das Tool die vorliegenden XML-Dateien in andere Formate. Für die Konvertierung verwendet das Tool XSL-Vorlagedateien und bietet die Parametereinstellungen wie Schriftgröße, Papierformat und das Festlegen von Parameterprofilen.

Es kommt beim Export von DOC-Dateien zu einigen Einschränkungen. Die im Dokument enthaltenen Bilder werden nicht berücksichtigt und Fußnoten erscheinen immer als Endnoten oder werden ganz ausgelassen. So werden die Ursprungstabellen in neue Tabellen konvertiert, deren Zeilenmaße sich dabei verändern und auch deren Hintergrundfarben vernachlässigt werden.

Die Text-Formatierungen sind abgewandelt. So sind fett-formatierte statische Felder aus dem Formular verschwunden. Die dynamischen Felder sind dagegen in kursiv und in Normaltext abgeleitet.

Zur Erstellung des eigenen DocBook-Dokumentes bietet das Tool eine Beispieldatei, die einen Einsatz von Absatz- und Zeichenvorlagen für die Strukturierung eines DocBook- Dokuments demonstriert.[21] Auch in diesem Format verschwinden die gespeicherten Formatierungen. So sind die eingefügten Bilder, zentrierte und rechtsbündige Zeilen sowie kursiv oder fett hinterlegte Felder nicht mehr aufzufinden.

[21] "o.V." Best of Praxis:Office plus(Teil2), http:www.pcwelt.de/ , 11.08.2005

Im Gesamtüberblick unterstützt das Programm pcwSxwExport den Export in viele Druckformate – allerdings mit erheblichem Datenverlust. Die Dokumente werden nicht gestaltungsgetreu abgebildet und sind somit lediglich zum Austausch von Informationen geeignet.

3.7 Überblick der angebotenen XML-Werkzeuge

Die Palette der kommerziellen Produkte und der XML-Werkzeuge aus der OpenSource-Gemeinde ist groß. Leider verhält sich häufig die Qualität dieser Produkte proportional zu ihrem Preis. So bieten Crystal-Reports sowie leistungsfähige und benutzerfreundliche Designer-Produkte höhere Qualitätsstandards – allerdings verbunden mit entsprechenden Kosten. Andererseits bietet das XML-Apache-Project mehrere Projekte kostenlos an, die allerdings wiederum erhebliche Programmierkenntnisse voraussetzen. Die iText-Lösung lässt zwar den XSLT-Schritt aus, vereinfacht damit den gesamten Konvertierungsprozess, ist aber ebenfalls an die Java-Programmierer gerichtet. Die anderen vorgestellten Programme WH2FO und pcwSxwExport ermöglichen jedem Benutzer, der sich an die einfachen Anweisungen hält, die Konvertierung von Dokumenten. Dabei werden allerdings starre und vom Ursprunglayout abweichende Dokumente generiert, die dem Zwecke des Informationsaustausches dienen - aber nicht für die layoutgetreue Dokumenterstellung verwendet werden können.

Diese Arbeit konzentriert sich auf Java OpenSource-Software für Microsoft Windows Betriebssysteme, da für die Umsetzung ein Programm der Microsoft Familie notwendig ist.

Produkt / Eigen-schaften	XSL Report Designer	Crystal Reports	XML-Projekt Apache	iText	WH2FO	pcwSxwExport
Hersteller	Antenna House	Microsoft	Apache Software Foundatin	Bruno Lowagie Paulo Soares	Fabio Giannetti	„PC-WELT“ Team
Umgebung	plattformunabhängig basiert auf JRE ab Version1.4.2	ASP,Win32 mit Zusatzsoftware plattformunabhängig	JRE	Java Klasse	MS VM JVM WINDOWS UNIX/LINUX	ab WINDOWS 98 JRE
Programmier-kenntnisse	nicht notwendig	nicht notwendig	XML-, Stylesheet Kenntnisse	XML-, Java-Kenntnisse	nicht notwendig	nicht notwendig
Layout Erstellung	Designprogramm	WYSIWYG	XSLT und XSL-FO	in Java programmiert	MS Word 2000	MS Word, OOo Writer, DocBook und interne XSL-Vorlagen
Output	PDF, Druckdateien	XML,DHTML,PDF,RTF, MS Excel, MS Word	XML, HTML, XHTML,PDF, WML,	PDF	HTML, PDF	PDF, HTML,RTF
Preis	ab 810,-€	ab 465,-€	kostenlos	kostenlos	kostenlos	kostenlos

Tabelle 3: Übersicht der Programmen zur Erstellung XML-basierten Dokumenten

4 Layouterstellung mit Textverarbeitungsprogrammen

Aus der Erkenntnis, dass keine der Tools den gestellten Anforderungen vollständig entspricht, ist es an der Zeit, ein eigenes Verfahren zu entwickeln. Dabei werden für die Layouterstellung bekannte Textverarbeitungsprogramme verwendet. Die Applikationen bieten einen zeitgemäßen Text-Output und eine Programmierschnittstelle, und machen das Steuern von Produkten zwischen Server und Client möglicht. Solche Software ist auf den meisten PCs bereits vorhanden und fordert damit keine zusätzlichen Investitionen ein. Die Textverarbeitungsprogramme unterstützen ebenfalls die XML-Dateiformate. Ein entscheidender Vorteil ist, dass jeder Kunde das Erstellen des Dokumentlayouts selbst vornehmen kann.

4.1 Wahl der Entwicklungsumgenung

„IDEs fassen verschiedene, bei der Software-Entwicklung benötigte Werkzeuge unter einer einheitlichen Oberfläche zusammen. Dabei sind die einzelnen, interoperierenden Werkzeuge in eine gemeinsame Oberfläche integriert und bieten eine einheitliche Benutzungsschnittstelle“.[22]

Es werden zahlreiche Entwicklungsumgebungen verschiedener Hersteller auf dem Markt angeboten: JBuilder von Borland, Eclipse von IBM, NetBeans und Sun ONE Studio von Sun entwickelt und viele andere. Nach dem ersten Auswahlkriterium „Verwendung von OpenSource“ stehen die beiden Entwicklungsumgebungen NetBeans und Eclipse zur Wahl. Nach der Gegenüberstellung der Basisfunktionen einer IDE (Editieren, Navigieren, Übersetzen und Debuggen) sind beide Entwicklungsumgebungen nach meiner Einschätzung als beinahe gleichwertig anzusehen. Beide IDEs bieten gute Werkzeuge wie Refaktoring, erweiterbare PlugIns, GUI-Unterstützung, „Syntaxhighliting“, und sind für mehrere Betriebssysteme verfügbar. Sie ermöglichen Codeänderung beim Debaggen und sind Open-Source-Produkte. Das entscheidende Kriterium für den IDE-Einsatz ist die Zusammenarbeit mit NetBeans seit der Version 3.6 mit OpenOffice.org. Aus der Tatsache, dass die beiden Tools von gleichem Hersteller Firma Misrosystem Sun stammen, kann man ableiten, dass die beiden Tools kompatibel sind. Office API (ein Teil der Office Development Kit) ist ein vollwertiger unabhängiger Tool und grundsätzlich für die Unterstützung aller Java IDEs konzipiert. Unter anderem bietet Office API Reference eine ausführliche Dokumentation zur Einbettung von Office-Klassen und Modulen, sowie der Verwendung von OpenOffice.org Macros in einer NetBeans-Entwicklungsumgebung. Weitere Vorteile der NetBeans IDE sind die leichte Erweiterbarkeit durch die externen Module und die XML-Unterstützung. Neben dem Erstellen

[22] Vgl. Melster R./ Mosconi M./ Pfeiffer C. u.a.(2003),S3-4

und dem Editieren von XML-Dokumenten unterstützt NetBeans die Erzeugung von XML-Bearbeitungsdateien wie zum Beispiel DTD, XML-Schema, Stylessheets (usw.). SAX- und DOM-Schnittstellen und Parser sind direkt in NetBeans IDE enthalten. Nach Angaben der Entwickler ist NetBeans ab Version 4.0 die erste IDE, die komplett auf Apache Ant basiert. Dieses Tool ist Java basiert und legt alle Konfigurationsfiles in XML ab.[23] Den oben erwähnten Vorteilen stehen lediglich das relativ schlechte Antwortverhalten und die langsame Geschwindigkeit gegenüber.

4.2 XML als Intern- und Externformat

Warum bemühen sich immer mehr Programme einen offenen (Austausch-)Standard zu unterstützen? Die Antwort liegt auf der Hand: XML bietet als Austauschformat viele Möglichkeiten, Dateien unabhängig von ihrer Herkunft zu verarbeiten. Dazu werden nun einige Vor-Überlegungen angestellt: Die XML-Fähigkeit der erweiterbaren Auszeichnungssprache ermöglicht es, viele Vorgänge so zu gestalten, wie es vor einigen Jahren noch nicht denkbar gewesen wäre. Die Inhalte der XML-Daten können von Applikationen und von Menschen erstellt, bearbeitet und gelesen werden. Bisher speicherten die Anwendungen ihre Dateiinformationen meist nur in einem Binärformat, dessen Bytefolge für das menschliche Auge unentschlüsselbar ist. Unzugänglich sind auch die mit der Dateiverarbeitung verbundenen Details. Mit dem XML-Einsatz sind die Daten nicht nur leicht zugänglich, sie bieten auch freien Zugriff auf die Dateiinhalte außerhalb der ursprünglichen Anwendungen. Die Dateninhalte können sowohl mit speziellem XML- als auch mit simplem Text-Editor bearbeitet werden. Mit dem Text-Editor können zum Beispiel auch beschädigte Dateien gerettet werden, wenn sie sich unter den nativen Anwendungen nicht mehr öffnen lassen. Es können auch anhand valider Dokumente verlorene und beschädigte Dateien in der Regel rekonstruiert werden. Diese Fähigkeiten tragen zur so genannten Robustheit des Systems bei.

Die Entwickler hoffen, durch die Unterstützung der offenen Standards wie XML und RTF ihren Programmen die Fähigkeiten zur unabhängigen und nahtlosen Zusammenarbeit mit anderen Programmen zu ermöglichen. Die Informationen können effizienter zwischen einzelnen heterogenen Prozessen ausgetauscht oder dem Benutzer zur Verfügung gestellt werden, ohne dass im Vorfeld besonderen Absprachen getroffen werden müssten. Diese Fähigkeit wird Interoperabilität genannt (siehe Kapitel 4.3). [24]

---

[23] „o.V“, http://www.golem.de/0412/35186.html Sun NetBeans 4.0 vorgestellt, 11.08.2005

[24] „o.V.” Interoperabilität, http://de.wikipedia.org/wiki/Interoperabilit%C3%A4t, 11.08.2005

Aus diesem Grund speichern Anwendungen ihre Dateien mittlerweile direkt in ein XML-Format, oder bieten ein XML-Format zum Austausch der Informationen an. Um Letzteres zu erzielen, übersetzen die Anwendungen die Daten aus einem eigenen in ein XML-Format.

Bei der Umsetzung stößt XML als durch und durch strukturierte Sprache an ihre Grenzen. In der Regel kann beim Übersetzen nur ein Teil des Dokuments eindeutig durch die XML-Struktur abgebildet werden, das andere Teil muss mit allen Mitteln in den Inhalt des Dokumentes eingebunden werden. Oft entstehen dabei umfangreiche, schlecht strukturierte und unlesbare XML-Dokumente, die noch dazu im XML-Dialekt geschrieben sind. Anhand dieser Tatsachen ist die Frage zu beantworten, in wie weit der Einsatz von XML eine bessere Interoperabilität zwischen den Anwendungen bewirken kann. Diese Problematik wird im Kapitel 4.5 behandelt.

4.3 Interoperabilität auf Softwareebene

Wie im Kapitel 4.2 schon besprochen spielt Interoperabilität eine große Rolle bei der Softwareentwicklung. In diesem Kapitel wird Interoperabilität als Begriff erklärt und ihr Einsatz in Textverarbeitungsprogrammen erläutert.

Was ist eigentlich Interopabilität? Der Begriff Interoperabilität bedeutet: „Zusammenarbeit in einem offenen System (gemäß dem Client/ Server-Modell). Unabhängig von der verwendeten Hardware, den eingesetzten Betriebssystemen, der verwendeten Netzwerktechnologie und der Realisierung einer Anwendung kann eine Zusammenarbeit zwischen diesen Anwendungen erfolgen.“[25]

„Im Zusammenhang mit Software spricht man vor allem dann von Interoperabilität, wenn mehrere Programme dasselbe Dateiformat oder dieselben Protokolle verwenden können.“[26]
Im Bezug auf Officeanwendungen bezieht sich Interoperabilität auf den Austausch von Dokumenten und wird „dateibasierte Interoperation“ genannt. Dabei kommt es darauf an, dass die Inhalte korrekt und vollständig bleiben, Darstellung und Form des Textes, der Formatierungen einzelner Felder, der Bilder und anderer Objekte innerhalb eines Dokumentes erhalten bleiben - und die einzelnen Funktionalitäten des Dokumentes als Ganzes unterstützt werden, zum Beispiel die identische Formatierung des Dokumentes für die Ausgabe an verschiedenen Medien. Ebenfalls wird Interoperabilität der Officesuiten aufgrund der XML-Basis untersucht.

---

[25] „o.V.” Interoperabilität, http://www.quality.de/lexikon/interoperabilitaet.htm, 11.08.2005
[26] „o.V.” Interoperabilität, http://de.wikipedia.org/wiki/Interoperabilit%C3%A4t, 11.08.2005

4.4 Kompatibilität der Office-Anwendungen

In der Konzeptumsetzung (siehe Kapitel 5) werden in MS Word erstellte Dokumentvorlagen in Openoffice.org Writer importiert. Die möglichen Konvertierungsfehler, nicht einheitliche Inhalte und abweichende Form der Darstellungen müssen in Vorfeld untersucht und ausgeschlossen werden. Um sicherzustellen, dass beim Import die Feldeinstellungen erhalten bleiben, wird in dieser Arbeit Kompatibilität der beiden Office-Programme untersucht.

4.4.1 Die Kompatibilität einer Office-Anwendung kann in folgenden drei Modellebenen dargestellt werden

a) Dokumentmodell

Jedes zu bearbeitende Dokument in einer Office-Anwendung wird aus einer Menge von Gestaltungsinstrumenten zusammengesetzt. Diese Konstruktionen, zum Beispiel Textboxen, Fußnoten oder Wasserzeichen, stellen ein Dokumentmodell dar, das je nach Textverarbeitungsprogramm unterschiedlich ausfällt. Die Kompatibilität auf der Dokumentmodellebene misst sich anhand ihrer Modellkompatibilität, also in welchem Maße die Modelle identisch sind, und der Modell-Abbildbarkeit, also in welchem Maße die Instrumente der Modelle übersetzt werden können.

b) Dateiformat

Office-Anwendungen lassen in der Regel verschiedene Dateiformate lesen und speichern. Die einzelnen Dateien enthalten zwar alle Informationen des Dokumentmodells, können aber durchaus anderes strukturiert sein. Die Strukturunterschiede begründen sich in der Art der Dateikodierung, dem Bearbeitungsstand oder auch in der Form der technischen Lesbarkeit. Der vollständige Informationsgehalt eines Dokumentsmodells ist in der Regel nur im ursprünglichen Dateiformat enthalten. Zum Beispiel werden in MS Word alle Dokumente binär-codiert gespeichert. Die Bearbeitung und der Zugriff auf die Objekte innerhalb der Dokumentmodellebene sind in diesem Fall nur aus MS-Anwendung heraus möglich. Die Konvertierung in andere Formate zum Zweck der Weiterverarbeitung bringt allerdings Abweichungen und Übersetzungsfehler mit sich. Folglich ist bei jeder Konvertierung mit dem Verlust von Informationen zu rechnen. Die Verwendung der standardisierten Dateiformate (zum Beispiel XML, RTF) oder dokumentierter Formate, als native Dateiformate erfüllen die

Voraussetzungen der Kompatibilität. Die technischen und semantischen Merkmale des Dateiformats sind bekannt, die Werkzeuge zur seiner Verarbeitung stehen zur Verfügung. Damit ist der Zugriff auf die Daten des Dokumentes möglich, ohne dabei die Dokumentstruktur zu verändern.

c) Gestaltungstreue

Ziel der Gestaltungstreue ist es, eine einheitliche Dokumentdarstellung auf allen Ausgabemedien, zum Beispiel Monitor oder Drucker, zu erreichen. Ein Benutzer erwartet, dass das Dokument immer so aussieht, wie er es zur Zeitpunkt der Benutzung auf Bildschirm wahrnimmt (WYSIWYG). Probleme der Darstellungstreue treten allerdings häufig auf und werden dem wünschenswerten Maßstab der Interoperabilität nur selten gerecht. Visuelle Unterschiede nach dem Datenexport oder -Import aus einer Anwendung beruhen oft fälschlicherweise auf der mangelnden Darstellungskompatibilität oder auf einem Dateiformat-Problem. Tatsächlich handelt es sich häufig um eine Inkompatibilität auf der Dokumentmodellebene, die durch einen Konvertierungswerkzeug falsch interpretiert wurde.

4.5 XML-Technologien zur Lösung der Inkompatibilitätsprobleme von Office-Anwendungen

Der Einsatz des XML-Dateiformates erleichtert zwar die Bearbeitung der Daten unabhängig von der ursprünglichen Anwendung, kann aber die Inkompatibilität auf Dokumentmodellebene nicht lösen. Um die Konvertierung eines Dokumentformates in das andere verlustfrei und umkehrbar zu gestalten, sollte man gewisse Einschränkungen in Kauf nehmen:

1. Durch die Festlegung einheitlicher Bearbeitungsrichtlinien der Dokumente und
2. durch die Aufteilung des Dokumentbearbeitungszyklus können erste Vorschritte erzielt werden.
3. Die Verwendung der Gestaltungsinstrumente, die identischen sind oder sich fehlerfrei übersetzen lassen.
4. Erst durch die konsequente Umsetzung der Richtlinien durch den Anwender ist die erfolgreiche Rücktransformation gewährleistet.

Die alternative Bearbeitung ist die Aufteilung der Dokumentbearbeitung in zwei Phasen: Die erste Phase beschränkt sich auf Inhaltbearbeitung, die zweite Phase beschäftigt sich nur mit der Gestaltung. Die unabhängige Bearbeitung des Dokumentinhalts und der

Dokumentgestaltung entspricht dem XML-Prinzip, der Trennung von Inhalt und Präsentation. Daraus lässt sich schließen, dass der Einsatz von XML in Verbindung mit XSL-Transformationen seinen Teil zur Interoperabilität beiträgt. Beide Office-Anwendungen bieten eigene Stylesheets zum Abbilden der XML-Dateien für die interne Präsentation. Sollte die anwenderbezogene Transformation an ihre Grenzen stoßen, bietet DOM, SAX standardisierte Programmierwerkzeuge zur Erstellung individueller Lösungen. (Kapitel 2.7.2 und Kapitel 2.7.3)

4.5.1 Vergleich: XML-Dateiformat, OOo/SO und MS Office2003

In diesem Kapitel werden die gegenwärtigen XML-Unterstützungen der Office-Anwendungen gegenübergestellt. Wie schon erwähnt, unterstützen beide Office-Anwendungen das XML-Dateiformat. OpenOffice.org oder StarOffice unterstützen das XML-Format ab Version OOo 1.0, der aktuellen Version 2.0 und SO ab Version 7(8).

	MS Office 2003	OOo/ SO
Dateiformat	MS eigenem binären Dateiformat (*doc, *xls)	Einheitliches XML-zentriertes Dateiformat
XML-Unterstützung	Word, Excel (anwendungsspezifisch)	alle Anwendungen
XML-Standardisierung	„Office 2003 XML Reference Schemata“	Office-Dateiformat OASIS
Grafikdarstellung	binär kodiert (über Zwischenablage austauschbar)	Verwendung nativer Formate
XML-Spezifikation	MS-Software abhängig	unabhängig
Speichern	Speichert die ganze Datei als *.xml	einzelne Zip -komprimierte XML-Dateien

Tabelle 4:Gegenüberstellung unter MS Office 2003 und OOo/SO verwendeter XML-Dateiformate.

Wie in der Tabelle ersichtlich, gibt es wenige Parallelen zwischen den beiden XML-Formaten. Die Unterschiede sind unter anderem auf Historie und Architektur der beiden Anwendungen zurückzuführen. MS Office bildet eine Familie unabhängig voneinander entwickelter Produkte. OOo/SO dagegen ist als integrierte Lösung konzipiert. Office 2003 bittet die Möglichkeit, die Dateien aus der Textverarbeitung und Tabellenkalkulation in einem XML-Format zu speichern. Das Standard-Format der Anwendungen bleibt das MS-eigene binäre Dateiformat (*doc, *xls

usw.). Dabei kann man auswählen, ob man die Dateien in XML-Format speichert, oder XML als Standard per Konfiguration einstellt.

Beim Speichern der Word oder Excel-Dateien im XML-Dateiformat ist wiederum mit Datenverlust zu rechnen. So werden zum Beispiel in Word eingefügte Excel Arbeitsblätter als druckbar kodierter Binärblock dargestellt. Beim Speichern einer Excel-Mappe in ein XML-Format gehen darin enthaltene Diagramme ebenfalls verloren. Die XML-Schemata sind Microsoft-spezifisch und werden in Office 2003 XML Reference Schemata dokumentiert. Die MS-Schemata umfassen eine Reihe einzelner Schemata: WordProcessingML (früher: WordML), SpreadsheetML, DataDiagrammingML (Visio, kein Bestandteil von Office 2003) und FormTemplate XML-Schemata (InfoPath).

OpenOffice.org oder Star Office verwenden dagegen ein einziges einheitliches XML-zentriertes Dateiformat für alle Anwendungen. Die Standardisierung eines Office-Dateiformates wurde durch OASIS festgelegt. Die Verwendung des XML-dokumentierten Dateiformates ist von der Anwendung unabhängig. Damit bietet OOo/SO die Sicherheit, langfristig und in beliebiger Weise über Daten zu verfügen. Die Grafiken werden im Gegenteil zu Office 2003 in nativen Formaten gespeichert, was sich kompakter als eine komprimierte XML-Präsentation oder binär-kodierter Block erweist. Die Aufteilung der Anwendungsdatei nach verschiedener Teilfunktionalität erleichtert zudem die zielgerechte automatisierte Bearbeitung von Dateien. Der Zugriff auf die Zip-komprimierten XML-Dateien ist mit etwas aufwändigem Packen und Entpacken verbunden, lässt aber zum Ausgleich schnell wachsende XML-Dateien kompakt darstellen. OOo/SO speichert prinzipiell alles in XML-Dateien und bietet daher eine schlüssige, vollständige XML-Ünterstützung als MS Office 2003. Office 2003 bietet dagegen nur für Word und Excel eine anwendungsspezifische XML-Untersctützung, die noch nicht ganz vollständig ist, und aus diesem Grund nicht als Standardformat eingestellt wird.

Aufgrund der weltweit marktführenden Position des MS Office ist OOo/SO gewissermaßen gezwungen, Interoperabilität zur MS Office herzustellen und bietet eine beinahe nahtlose Integration der XSLT-basierten Filter für Import/Export der Office-Dateien. Allerdings haben Excel und Word mit der Einführung benutzerdefinierter Schemata einen bedeutenden Schritt in Richtung der Erweiterbarkeit gemacht.

4.5.2 Bedeutung der Interoperabilität auf Basis der XML-Dateien/ XML Bedeutung

Das XML-Dateiformat gewinnt als Austausch-Format immer mehr an Bedeutung, nicht nur innerhalb der einzelnen Anwendungen, sondern auch zum Beispiel in Dokumentmanagement- oder Geschäftsprozess-Systemen. Aus diesem Grund ist die Normierung der XML-Formate zwingend notwendig. Welche von beiden - OpenDocument oder XML-Dateiformat – wird sich mit der Zeit zeigen. Solange Inkompatibilität auf der Dokumentmodellebene auftritt, kann man mit XML-Unterstützung nicht mehr viel erreichen. Allerdings bietet der Einsatz offener Standards mehr Wahlmöglichkeiten und Werkzeuge, die manche Probleme wirksam lösen können. XML-Technologien verfolgen das Ziel, die Dateiformate zu normieren.

Die Kompatibilität der Office-Programme ermöglicht die gemeinsame Bearbeitung der Dokumente und unterstützt die getreue Abbildung der Dokumente unter beiden Anwendungen. Die Unterstützung des XML-Formates als natives Dateiformat bietet den Zugriff auf Dokumente außerhalb der Anwendungen über XML-Schnittstellen und die automatisierte Verarbeitung der Dateien innerhalb der Geschäftsprozessysteme.

4.6 OpenOffice.org Dokumentformate auf Basis von XML

Nach SGNL und HTML ist mit XML eine weitere Sprache zur Auszeichnung von Textdokumenten verfügbar. Heute basiert eine Reihe von Dokumentformaten auf dieser Auszeichnungssprache.

Alle OOo-Anwendungen teilen sich eine gemeinsame DTD. Das dadurch entstehende Dokumentmodell bleibt in verschiedenen Applikationen konsistent und behält bei der Transformation aus einer Applikation in die andere das gleiche Format. Dieses Format bildet eine Struktur, die sich auf eine optimale Verwendbarkeit und leichtes Verständnis gründet und nicht nur einen in XML abgelegten Abzug der inneren Repräsentation abbildet.
Beim Design des OOo-XML-Formates wurde sehr viel Wert darauf gelegt, bestehende Standards und Empfehlungen des W3C einfließen zu lassen. Daher enthält das Format Elemente und Attribute von HTML, XSL-FO, XLINK, Dublin Core, MathML und SVG.
OpenOffice.org beinhaltet folgende Dateien in Office Packet.[27]

[27] Bundesministerium des Innern(2005) S.28-32

4.6.1 Das XML Dateiformat im OpenOffice.org

Alle OpenOffice.org-Applikationen nutzen ab Version 1.0 auf XML basierende Datei-Formate. Ab Version 2.0 verwendet OpenOffice.org auch den „OpenDocument-Format“ von OASIS. Dieses Format ist ein erweitertes OpenOffice.org XML-Dateiformat.

Alle in OpenOffice.org erstellten Dateien und auch Konfigurationsdateien liegen im weit verbreiteten ZIP-Format vor. Bei der Dekomprimierung erhält man neben reinen XML-Code auch zusätzliche Dateien, einige Dateien werden dabei in Verzeichnisse ausgelagert (siehe Tabelle 5). Die Funktionalität der Dateien wird durch die Namen der Verzeichnisse beschrieben. In einige Dateien enthalten die anderen beschreiben den Inhalt des Paketes (z.B. die Verschlüsselungsmethode oder im Dokument enthaltenen Bilder). Meta-Daten in meta.xml-Strang sind nicht komprimiert, damit die Suche und Extraktion nach Meta-Daten erleichtert wird. Die Bilder und eingebetteten Objekte werden in ihrem ursprünglichen Format gespeichert. Im entpackten Zustand findet man normalerweise alle Teile, die zusammen das komplette Dokument ergeben. [28]

[28] „o.V.“ Häufig gestellte Fragen, http://de.openoffice.org/doc/faq/xml/, 11.08.2005

Dateien	**Kommentar**
meta.xml	Informationen über das Dokument (Autor, Zeitpunkt der letzten Speicherung, ...)
styles.xml	Formate, die in diesem Dokument genutzt werden.
content.xml	Hauptinhalt des Dokumentes (Text, Tabellen, grafische 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 in ihren ursprünglichen, binären Formaten enthält
Dialogs/	Verzeichnis, welches die Dialoge enthält, die von den Dokumentenmakros gebraucht werden
Basics/	Verzeichnis, welches die StarBasic Makros 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 normalerweise in einem binären Format vor.

Tabelle 5 : Das XML Dateiformat im OpenOffice.org

Quelle: http://de.openoffice.org/doc/faq/x

4.6.2 Aufbau des OpenOffice.org XML-Dateiformates

In diesem Kapitel werden die Aufgaben und Inhalte der einzelner OpenOffice.org XML- Dateien beschrieben. Besonders ausführlich wird in Rahmen dieses Kapitels die Struktur der context.xml Datei dargelegt. Der Inhalt dieses Kapitels wird vorausgesetzt für das Verständnis des im Kapitel 5 erläuterten Konzepts und für den Ablauf der einzelnen Programme.

In OpenOffice.org XML-basierte Dokumente können auf zwei Wegen dargestellt werden: Als ein valides DTD konformes XML-Dokument und als ein Packet der Subdokumente. Diese Subdokumente sind jeweils valide und identifizieren sich durch ein eindeutiges Root-Element. Der Name des Root-Elements beschreibt welche der einzelne Aspekte des Gesamtdokumentes ein Subdokument in sich gespeichert hält (siehe die Tabelle: Subdokumente). Zum Beispiel enthält ein Subdokument die Seitenlayout-Informationen, während das andere Dokument die Information über den Dokumentinhalt speichert.

Name	Root-Element	Inhalt
manifest.xml	<manifest:manifest>	Angaben zu allen in Zip-Archiv enthaltenen Dateien
meta.xml	<office:document-meta>	Angaben zum Autor, Erstellungsdatum, letzter Bearbeitungsstand
styles.xml	<office:document-styles>	Seitenlayout und automatische Formatierungen
content.xml	<office:document-content>	Dokumentinhalt und automatische Formatierungen
settings.xml	<office:document-settings>	Anwendung spezifischer Formatierungen z. B.: Desktopdarstellungen und Druckinformationen

Tabelle 6: Subdokumente
Quelle: Sun Microsystem (2002)

Jedes der oben behandelten Subdokumente (außer manifest.xml) folgt im Wesentlichen der gleichen Dokumentstruktur, die innerhalb des Root-Elementes auf drei weiteren Elementen aufgebaut ist:

<office: meta>	enthält Dokument-Informationen
<office: styles>	beschreibt die Formatvorlagen
<office: body>	beschreibt Text-Darstellung, Tabellen und Bilder[29]

Weitere ausführliche Beschreibung des OpenOffice.org XML-Dateiformates ist in der gleichnamigen technischen Referenz (siehe Literaturverzeichnis) zu finden.
Die Inhalte, der in der Tabelle 6: aufgelisteten Subdokumente, werden weiter unten genauer beschrieben. Dabei muss man erwähnen, dass die Datei manifest.xml kein Bestandteil des Openoffice.org Dokumentes ist. Das erkennt man nicht an ihrem Root-Element, sondern erst durch das Entpacken des Dokumentes, zum Beispiel mit einem WinZip-Tool oder anderen Dekomprimierungs-Programmen.

a) Manifest

In META-INF/manifest.xml befindet sich die Beschreibung der äußeren Struktur des OOo-Dokumentes. Hier werden alle im Zip-Archiv vorkommenden Dateien samt ihrer Dateipfade und der anwendungspezifischen Attribute aufgelistet. Verschlüsselungsinformationen sind - falls vorhanden - ebenfalls enthalten.

b) Metadaten

Meta.xml enthält allgemeine Dokumentinformationen, die Aufschluss über den Initialen Autor, Datum, Sprache, Inhalt und Stichworte geben.

c) Formatvorlagen

Formatvorlagen werden in styles.xml abgelegt und können auch in content.xml eingebettet vorkommen. Die Informationen über die Darstellung werden über die Attribute des Elementes „style:properties“ gespeichert. Für die Nummerierungen und Listen gibt es ebenfalls Elemente, die deren Verhalten und Aussehen steuern. Die benutzerdefinierten und automatischen Vorlagen unterscheiden sich in ihrem Aufbau nicht.

[29] Vgl. Sun Microsystem (2002) S.28-35

d) Inhalt

In content.xml wird der eigentliche Dokumentinhalt gespeichert. Dabei wird zwischen Font-Deklarationen, automatischen Formatvorlagen und dem eigentlichen Dokumentkörper unterschieden.

Die Informationen über die Schriftartenfamilie, Pich-Wert und Zeichensatz sind in Font-Deklarationen, die Vorlagen für Schriftarten beschreiben, abgelegt. Über den Vorlagenamen können die Schriften einheitlich referenziert und Ersatzschriften leichter zugeordnet werden, wenn eine Schriftart nicht verfügbar ist. Automatische Formatvorlagen werden selbständig angelegt. Die Formatinformationen werden in Vorlagen abgelegt und dienen der strikten Trennung von Inhalt und Form. Sobald einem Teilinhalt eine bestimmte Form zugewiesen ist, wird direkt eine automatische Vorlage angelegt, die für eine immer gleiche Repräsentation von Format- und Layout-Informationen sorgt. Im Text wird die Formatvorlage über einen automatisch generierten Namen referenziert. Die wiederholt verwendeten Auszeichnungselemente müssen nur einmal definiert werden. Der Textkörper wird in einzelne Absätze des Dokumentes aufgeteilt. Die wichtigsten Elemente sind „text:h“ und „text:p“. Sie enthalten die Überschriften und die eigentlichen Texte in Form von Paragrafen. Diese Unterscheidung ist bei den sonst identischen Formaten notwendig, um zumindest teilweise Strukturinformationen zusätzlich zu Form und Inhalt transportieren zu können. Jeder Absatz stellt ein eigenes Element dar, das durch die Auszeichnung „text:p“ repräsentiert wird. Innerhalb dieses Elementes wird der eigentliche Text mit Bezug auf eine Formatvorlage angegeben. Die abweichenden Formate innerhalb des Elementes „text:p“ werden einer automatischen Vorlage mit „text:span“ zugewiesen. Die Texte werden als durch die Leerzeichen getrennte Worte gespeichert. Die weiteren wichtigen Elemente sind: „text:s“, der mehrere hintereinander folgende Leerzeichen ersetzt, „text:tab-stop“, der Tabulatoren implementiert und “text:line-break“, der die Zeilenumbrüche darstellt. Fußnoten, Endnoten und Referenzen sind ebenfalls Subelemente von Absätzen und werden dort direkt angegeben.

e) Einstellungen

In settings.xml werden Einstellungen zu den Applikationen gespeichert.[30]

[30] Vgl. Scherm B/ Stökel.J (2002) S.30-36

4.7 Richtlinien für gemeinsames Bearbeiten der Dokumente unter MS Word und OpenOffice.org

Wie oben schon besprochen sind die beiden Office-Programme nicht vollständig zueinander kompatibel. Die Strukturunterschiede sind auch in jeweiligen XML-Formaten wiederzufinden, die eigentlich zum Erreichen der Interoperabilität eingesetzt werden. Trotzdem machen die Überarbeitungsfunktionen (Filter) die gemeinsame Bearbeitung von Dokumenten in MS Word und OpenOffice.org Writer möglich. Wozu dieses notwendig ist, erfahren sie im Kapitel 5.2.3 dieser Arbeit. Eine entscheidende Rolle spielt dabei die Komplexität der zu bearbeitenden Dokumente.

Die einfachen Dokumente können ohne größere Probleme bearbeitet werden, denn Übersetzungsfunktionen von OpenOffice.org und MS Office sind interoperabel.

Handelt es sicht um komplexe Dokumente, sollte ein gemeinsamer Bearbeitungsablauf festgelegt und eingehalten werden. Dabei ist zu empfehlen, die folgenden Punkte zu beachten:

1. Zur Bearbeitung ist ein Dokumentformat festzulegen.
2. Die Bearbeitung ist auf der rein inhaltliche Ebene durchzuführen.
3. Die Formatierungen erst nach Beendigung der Inhaltbearbeitung vornehmen.
4. Für die Formatierungen ist es zwingend notwendig, MS-Word-Formate festzulegen, weil unter MS Word keine OpenOffice.org-Filter verfügbar sind.
5. Die Formatierungen sind in der letzten Stufe der Dokumentbearbeitung vorzunehmen, da das Mapping zwischen OpenOffice.org und MS Office nicht eins zu eins funktioniert.
6. Das häufige Konvertieren von einem Format in das andere ist zu vermeiden.

Diese Punkte beziehen sich nur auf Austausch und Funktionalität der Textverarbeitungsprogramme. Bei Bearbeitung von Tabellenkalkulationen und komplexen Präsentationen sind zahlreiche weitere Punkte zu beachten, die in dieser Arbeit nicht erwähnt werden.

4.8 Prinzip der effektiven Layouterstellung in den Textverarbeitungsprogrammen

Trotz der in Kapitel 4.7 beschriebenen Einschränkungen liegen die Vorteile der Textverarbeitungsprogramme auf der Hand. Die MS Office und OpenOffice.org entsprechen den hohen Anforderungen bei der Layouterstellung, die an professionelle Dokumentverarbeitung gestellt werden. Die bequeme Vorlagenerstellung ist für alle Anwender zugänglich, die bereit sind, sich mit dieser Thematik auseinanderzusetzen. Die Vorgehensweisen sind in den jeweiligen Hilfe-Dateien beider Programme ausführlich dokumentiert und stehen dem Programmbenutzer zur Verfügung. Erst bei der Erstellung der komplexen Dokumentvorlagen sind die Kenntnisse der Macro-Erstellung und der VBA- beziehungsweise VBS-Programmierung gefragt. In dieser Arbeit werden die Vorlagen nur zum Zweck der Darstellung betrachtet. Die weiteren Dokumentfunktionalitäten wie das Füllen der Felder mit Inhalt oder automatisches Drucken werden entsprechend nicht mehr mit dem XML-Grundsatz der strikten Trennung zwischen Inhalt und Darstellung verfolgt.

Neben der Verfügbarkeit und der leichteren Bedienbarkeit haben die Textverarbeitungsprogramme einen weiteren Vorteil: Sie stellen eine Fülle der Gestaltungsmöglichkeiten in Rahmen eines WYSIWIG (What You See Is What You Get) Projekts zur Verfügung. Dieses präsentiert einzelne Punkte des Dokumentes auf dem Bildschirm genau so wie bei der Druckerausgabe oder bei der Browserdarstellung. Das Dokument wird nicht auf der Ebene des Druckbildes dargestellt, sondern als Text mit seinen Attributen. Zurzeit gibt es noch kein Programm, welches das WYSIWYG-Prinzip vollständig unterstützt. Für die Layouterstellung bedeutet das, dass der Ausdruck nicht genau, sondern lediglich sehr ähnlich abgebildet wird. Sollten die Abweichungen für das menschliches Auge zu erkennen sein, besteht die Möglichkeit, die einzelnen Dokumentobjekte so zu bearbeiten, bis die Ergebnisse zufriedenstellend sind. Jeder Bearbeitungsschritt ist für Anwender sofort sichtbar. Die Darstellungsfehler werden frühzeitig erkannt und können korrigiert werden. In Vergleich zur Layouterstellung mittels Stylesheets- oder iText-Programmen (siehe oben) erweist sich der Einsatz der Textverarbeitungsprogramme als simpel und trotzdem effektiv.

4.9 Erstellung Dokumentvorlagen

Im vorgestellten Konzept werden die im MS Word erstellten Vorlagen in OpenOffice.org Writer importiert und erst dann weiterbearbeitet. Es stellt sich die heuristische Frage, wozu man diesen Zwischenschritt benötigt. Denn Dokumentvorlagen können auch direkt in Writer erstellt werden. Dafür sind meiner Meinung nach zwei Bewegungsgründe vorhanden:

Erstens ist es anzunehmen, dass Word fast auf allen PC vorhanden ist und jeder Anwender, der einen PC benutzt, auch mit dem Word-Programm vertraut ist.

Zweitens werden die aktuellen Vorlagen zurzeit auch mit MS Word erstellt. So müssen sich die Anwender nicht in neue Software einarbeiten.

Natürlich ist auch OpenOffice.org weltweit vertreten und einfach zu bedienen. Die oben getroffenen Aussagen können auch widerlegt werden. Trotzdem greift man in der öffentlichen Verwaltung und Industrie immer noch auf altbewährte Mittel zurück. Der Trend zum Umstieg auf OpenSource-Software hat erst begonnen.

4.9.1 Dokumentvorlagen

„Alle Microsoft Word-Dokumente basieren auf einer Dokumentvorlage. Eine Dokumentvorlage legt die Grundstrukturen eines Dokuments fest und enthält Dokumenteinstellungen wie Autotext, Einträge, Schriftarten, Tastenbelegungen, Makros, Menüs, Seitenlayout, spezielle Formatierungen und Formatvorlagen.“[31]

Die Dokumentvorlagen sind zum Drucken von Seriendokumenten bestimmt. Aus diesem Grund beschränke ich die Layoutdarstellung auf die Felder der Kategorie Seriendruck, Datum und Uhrzeit.

[31] Microsoft Office Word Hilfe

4.9.2 Darstellung einiger in MS Word 2003 erstellter Felder samt Feldfunktionen in OpenOffice.org 1.1 XML-Format

Beim Einfügen eines Feldes mit Namen Rechnungsnummer aus der Kategorie Seriendruck mit dem Feldnamen MergieField generiert MS Office automatisch das Feld auf der Dokumentvorlage.

Beispiel:

«Rechnungsnummer» mit der die Feldfunktion:

{ MERGEFIELD Rechnungsnummer * MERGEFORMAT }

nach dem oben beschriebenen Konvertierungsvorgang sind die Feldeinstellungen in der OpenOffice.org-definierten XML-Datei wiederzufinden:

Beispiel:

<text:database-display text:table-name="" text:table-type="**table**" text:column-name=" **Rechnungsnummer** "><**Rechnungsnummer**></text:database-display>

Als nächstes wird das Feld mit dem Feldnamen Ask mit dem Textmarkennamen **Beispiel** dargestellt. Der OpenOffice.org-Eintrag sieht wie folgt aus (als Text wird die Zeile „das ist ein Beispiel“ eingefügt).

Beispiel:

```
<text:bookmark-start text:name="Beispiel" />
 <text:variable-input text:name="Beispiel" text:description="" text:display="none"
 <text:formula="das ist ein Beispiel" text:value-type="string" text:string-value="das
 ist ein Beispiel" />
<text:bookmark-end text:name="Beispiel" />
```

Das Datum-Feld gehört zur Kategorie Datum und Uhrzeit mit dem Dateiformat: „dd.MM.yyyy“
Im Word-Dokument erscheint das Feld in folgendem Layout 19.08.2005
Es hat die Feldfunktion:

Beispiel:

{ DATE @ “dd.MM.yyyy“ * MERGEFORMAT }

<text:date style:data-style-name="**N36**" text:date-value="**2005-07-13T17:39:14,37**">**13.07.2005**</text:date>

4.9.3 Richtlinien zur Erstellung der Dokumentvorlagen unter MS Word

Da die in MS Word erstellten Vorlagen schon existieren, werden die festgelegten Einstellungen zum größten Teil übernommen. Aus diesem Grund werden alle Felder einschließlich Datum einheitlich als MERGEFIELD definiert.

Beispiel:

```
{ MERGEFIELD  Feldname \* MERGEFORMAT }
```

Die Feldnamen sollen eindeutig sein, weil sie als Attribute und Text des XML-Tags übernommen werden und als Variablen im Programmcode eingesetzt werden.
Die Feldnamen werden aus jeweiligen Tagenamen und so genannten Elternknotennamen mit dem Unterstrich verbunden. Ist XML mehrstufig verschachtelt, muss ein Feldname zur eindeutigen Identifikation mindestens zwei Elternnamen enthalten.

Beispiel:

```
<root>
        <eltern1>
                <eltern2>
                        <eltern3>text</eltern3>
                </eltern2>
        </eltern1>
</root>
```

Feldname: eltern1_ eltern2_ eltern3 mit Feldfunktion:

```
{ MERGEFIELD  eltern1_ eltern2_ eltern3 \* MERGEFORMAT }
```

Aufgrund nicht hundertprozentiger Interopabilität zwischen MS Office- und OOo/SO-Textverarbeitungen sollten Formulare möglichst einfach gestaltet werden. So sollte man etwa auf Makros und spezielle Formatierungen verzichten. Die individuelle Gestaltung des Seitenlayouts, unabhängige Grafik- und Feldpositionierung, sowie die freie Wahl der Schriftart und der Schriftgröße bleiben davon unberührt.

4.9.4 Feld Seriendruck ist nicht gleich Seriendruckfeld

Um das Feld Seriendruck unter Word zu definieren, muss man sich einen Feldnamen unter 14 Angeboten aussuchen. Danach muss man die allgemeinen Formatierungen (wie Großbuchstaben oder erster Buchstabe groß) und spezifische Eigenschaften oder auch Formatierungsschalter festlegen. Sie steuern, ob das Seriendruckfeld ein zugeordnetes Feld ist oder die Zeichenkonvertierung vertikal zur Formatierung verläuft. Außerdem kann man das Feld selbst benennen und später in der Dateivorlage mit den Funktionen: Feld bearbeiten, Feldfunktionen ein/ausschalten und Feld aktualisieren nachbearbeiten.

Beim Seriendruckfeld in OpenOffice.org kann man nur bereits existierende Feldnamen einer Datenbanktabelle einfügen. In der Vorlage können zudem nur Layouteinstellungen dieses Feldes geändert werden. Der Feldname bleibt dagegen unveränderbar. Diese gravierenden Unterschiede sind auf die zugrunde liegende Feldstruktur zurückzuführen. So ist in Word das Database ein Feldname der Teilmenge Feld Seriendruck. In der OpenOffice.org ist das Seriendruckfeld eines der fünf unter der Option Datenbank zusammengefassten Felder. Auf diese Weise bleiben die Feldeigenschaften erhalten, wenn man das Word-Dokument unter der OpenOffice-Writer-Anwendung öffnet und als .SXW (alt) oder .OTD (neu) speichert. Dabei werden die Datenfelder im XML-Code auch als Database-Felder angezeigt. Verändert man die OpenOffice-Vorlagen, so verändert sich das ursprünglich definierte Feld Seriendruck in einen normalen Standard-Text. Aus diesem Beispiel erkennt man, wie wichtig die Kompatibilität der Textverarbeitungsprogramme neben der XML-Unterstützung ist.

4.9.5 Weitere Empfehlungen zur Erstellung der Textdokumentvorlagen unter OpenOffice.org

Wie bereits erläutert, kann unter Writer das Seriendruckfeld nicht verändert werden. Aus diesem Grund sollten andere Feldtypen zur Vorlagenerstellung verwendet werden. Die Gestaltung der Vorlagen ist jeden frei zu überlassen.

Mein Vorschlag ist, Feldtype Platzhalter zu benutzen. Die Feldeigenschaften ermöglichen es, den Feldnamen frei zu gestalten. Die Funktion des Feldes ist aus dem Namen abzuleiten, hält Platz für Informationen bereit. In der content.xml-Datei findet man das Feld auf gleiche Art und Weise wie das Seriendruckfeld als Tag <text:placeholder> dargestellt.
(zum Vergleich: Kapitel 4.9.4)

Weitere Informationen zum Erstellen von Textdokumentvorlagen unter OpenOffice.org Writer sind in der Hilfe-Datei zu finden. Näheres zur XML-Darstellungen einzelner Felder sind im Dokument „XML File Format 1.0“ ausführlich erklärt.

5 Umsetzung

5.1 Randbedingungen

Entwicklung eines Konzepts zur Darstellung XML-basierter Dokumente im druckbaren Standardformat, zum Beispiel im PDF-Format. Integrierung des Druckdienstes in das laufende Projekt mit dem Ziel:

1. Der Entwicklung eines einheitlichen Verfahrens für die Dokumenterstellung.
2. Den Anwendern ein schnelles und komfortables Gestalten der Dokumente nach vorgegebenen Richtlinien und Normen für Anwender zu ermöglichen.
3. Den Entwicklern das schnellere Implementieren der Layoutänderungen in den laufenden Prozess zu ermöglichen.
4. Den aktuellen Druckvorgang effektiv zu optimieren.
5. Die Erweiterung des Kundenservices darzustellen.

5.2 Aktueller Stand

Die Oracle-Datenbank oder MS SQL-Server-Datenbank exportieren Daten in Form von XML-Dateien und speichern sie auf dem konfigurierten Verzeichnis, üblicherweise auf dem Server ab. Die XML-Dateien enthalten jeweils einen Datensatz. Die Weiterbearbeitung der XML-Dateien erfolgt je nach Verwendungszweck und späteren Einsatzbereich.

5.2.1 Erster Verarbeitungsablauf

Die Dokumente stellen im Endergebnis die Rechnungsbelege dar (siehe Abbildung 5). Das Layout ist von den Softwareanwendern als MS Word-Dokumentvorlagedatei entworfen und steht zur Verfügung. Ein in VisualBasic geschriebener Tool parst die XML-Datei und importiert Ergebnisse in eine MS Dokumentvorlage-Datei. Die Rechnungen erfolgen in zwei Formaten als MS Word- und PostScript-Dateien. Mittels kommerzieller AdobeWriter- oder auch Freeware MAKEPDF2- Tools konvertiert man PostScript- zum druckbaren PDF-Format.

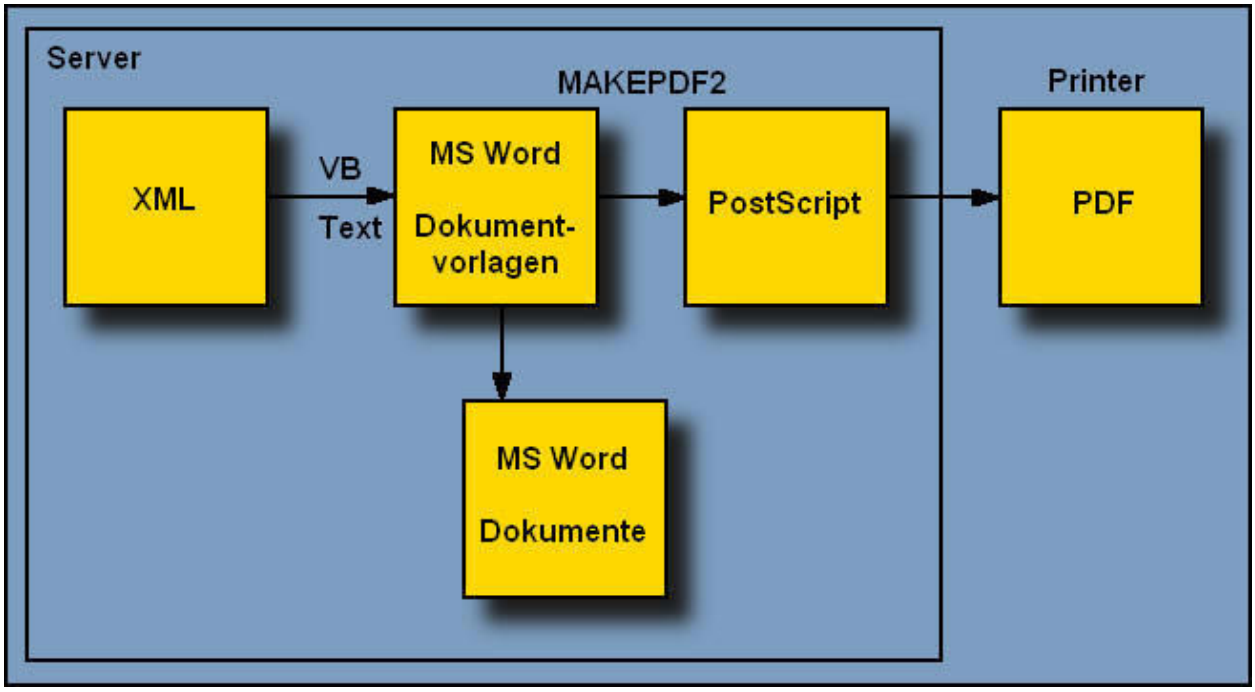

Abbildung 5: Erster Verarbeitungsablauf

Vorteile:

- Oben beschriebener Bearbeitungsablauf stellt ein funktionsfähiges Modul dar.
- Dieses Modul ist in der gleichen Programmiersprache entwickelt, wie das gesamte Projekt.
- Am Ende stehen zwei Ausgabeformate zur Verfügung: MS Word und PDF.

Nachteil:

- Die XML-Dateien werden als Textdateien bearbeitet, dabei gehen die Vorteile der XML-Technologie verloren.

5.2.2 Zweiter Verarbeitungsablauf

Ziel ist es, XML-Dateien mittels XSL-Stylesheets nach HTML-Dateien zu transferieren und im Browser darzustellen.

Der zweite Ablauf stellt zwar eine optimale Wandlung von XML-Dateien dar, setzt dabei aber Programmierkenntnisse beim Softwareanwender voraus. Außerdem ist jede Layoutänderung mit aufwändigen Korrigieren oder Neuerstellen der Stylesheets verbunden.

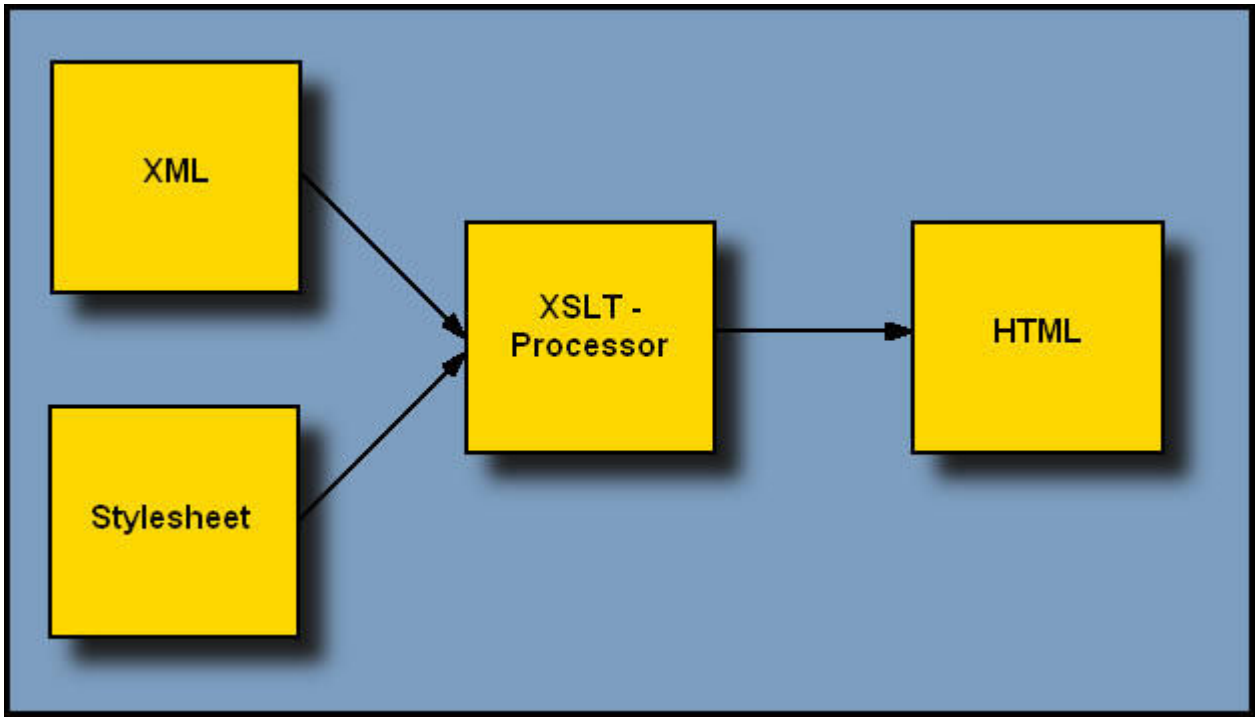

Abbildung 6: Zweiter Verarbeitungsablauf

Um die Inhalte in ein druckbares Format zu bringen, wird die ähnliche Methode wie in Kapitel „Erster Verarbeitungsablauf“ angewendet. Die XML-Inhalte werden in Dokumentvorlagedateien in dafür vorgesehene Platzhalter eingefügt.

Vorteil:

- Verwendung traditioneller XSL-Transformation.

Nachteile:

- Stylesheets müssen programmiert werden.
- Die Ausgabe im Druckformat ist nur bedingt möglich.

5.2.3 Neues Konzept

Seit Version 1.0 speichert OpenOffice.org die Dokumente in XML-Format.
Die Textverarbeitung speichert die SWX-Datei, die in ein ZIP-Archiv enthalten ist und mehrere zusammengefasste XML-Dateien enthält. Die Dateien können im einem Editor oder Browser angezeigt werden. Auf diesem technischen Stand basiert das neue Konzept. Es teilt sich in zwei Phasen auf: einer manuellen- und einer automatisierten Phase.

Zur manuellen Phase gehört Schritt eins dieses Konzeptes (siehe Abbildung 7):

Schritt 1:

In MS Word erstellte Dokumentvorlagen werden in OpenOffice.org importiert und als SXW-Datei gespeichert. Die SXW-Datei wird entzippt. Aus dem Inhalt des Zip-Archivs wird die content.xml-Datei zur Weiterverarbeitung entnommen und in ein vorgesehenes Verzeichnis eingefügt. Die restlichen Dateien des Zip-Archivs werden in das andere, ebenfalls vorgesehene Verzeichnis abgelegt. Dieser Schritt wird nur gelegentlich durchgeführt, zum Beispiel beim ersten Einrichten der Prozessumgebung oder nach der Layoutänderung eines bereits vorhandenen Dokumentes.

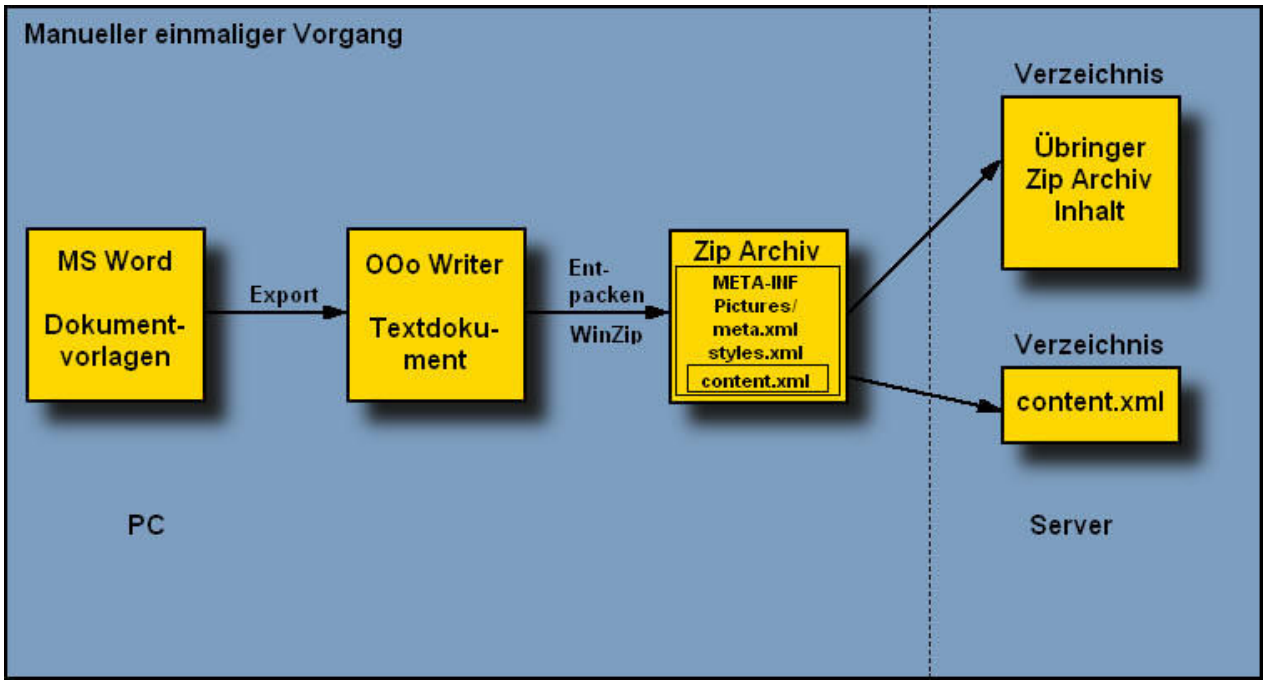

Abbildung 7: Manueller einmaliger Vorgang

Zur automatisierten Phase gehören die Schritte zwei bis sechs (siehe Abbildung 8):

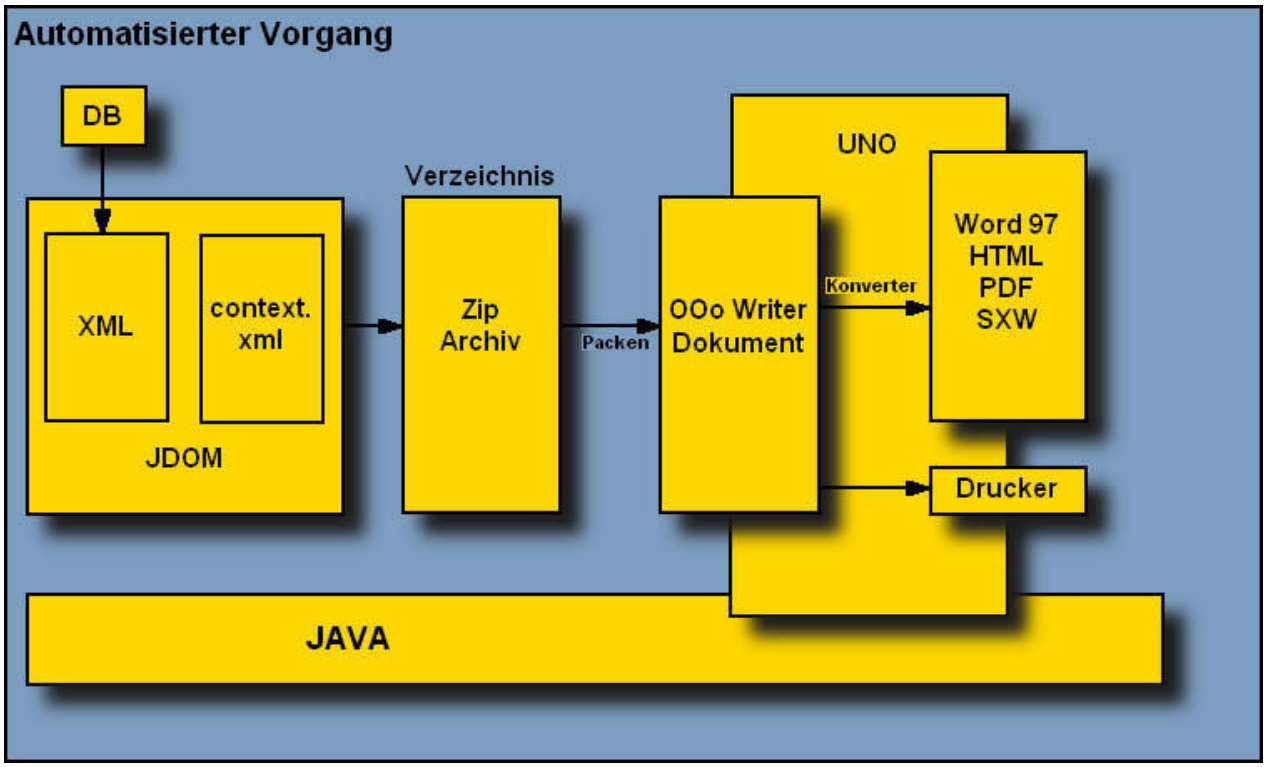

Abbildung 8: Automatisierter Vorgang

Schritt 2:
Eine XML-Datei, welche die Informationen zum Beispiel aus einer Datenbank liefert, wird geparst. Die dabei gewonnenen Daten werden zwischengespeichert.

Schritt 3:
Parsern einer context.xml-Datei und Einfügen in die durch Schritt zwei gewonnenen Informationen.

Schritt 4:
Erstellung des Druckformates. Die content.xml wird in das ursprüngliche Zip-Archiv eingefügt und gepackt. Dabei entsteht ein OpenOffice.org-Textdokument.

Schritt 5:
Im vorletzten Schritt erfolgt die Konvertierung der bearbeiteten Datei in weitere druckbare Formate wie Word97 oder PDF.

Schritt 6:
Als letzter Schritt folgt das Drucken des Dokuments.

Vorteile:

- Layout wird auf der Basis der Word-Dokumentvorlage oder auch in OpenOffice.org-Dokumentvorlagen erstellt.
- Layouterstellung basiert sich auf dem WYSIWYG-Prinzip.
- Die XML-Dateien werden mit XML-Technologien bearbeitet.
- Während des Prozesses entsteht sofort ein druckbares Dokument als SXW-Datei.
- Die automatisierten Schritte laufen ohne Installation von MS Word oder OpenOffice.org Software.
- Das Drucken und Konvertieren der Dokumente erfolgt über OpenOffice.org Schnittstelle.

Nachteile:

- Umständliches Zippen und Entzippen
- Für die Neugestaltung von Dokumentvorlagen sind nicht unerhebliche Kenntnisse der OpenOffice.org XML-Formates notwendig.

5.3 Programmablauf

Wie oben beschrieben, enthält eine XML-Datei unabhängiger Herkunft (die zum Beispiel aus einer Datenbank stammt) Informationen, die in einem Bericht, einem Formular oder in einer Rechnung ausgegeben werden sollen. Dafür müssen die Informationen aus einer Datei ausgelesen und zwischengespeichert werden. Genau das macht das Programm TextDokument.java

5.3.1 Parsen eines XML-Dokumentes mit JDOM

Zum Parsen von XML-Dokumenten bietet JDOM zwei Builder-Implementationen: SAXBuilder und DOMBuilder. Sie bilden ein Dokument-Objekt ohne zusätzliche Unterstützung zu ihren DOM- und SAX-Funktionalitäten. Trotzdem ist es zu empfehlen, den SAX-Builder zu nutzen, weil er eine bessere Performance bietet und wenig Speicherplatz benötigt. Außerdem benutzt der aktuelle DOM-Parser im Hintergrund SAX, um den Dom-Baum aufzubauen. Die Dokument-Objektklasse bietet die Möglichkeit, ein existierendes XML-Dokument von seinem Speicherort mittels org.jdom.input-Interface zu laden, einzulesen und ein JDOM-Dokument zu erzeugen. Dafür sind nur zwei Zeilen im Programm-Code erforderlich:

SAX Builder builder = new SAXBuilder();
Ducument doc = builder.build(new File());

Das Dokument kann auch aus einer URL oder einem InputStream stammen. In einem Dokument-Objekt sind alle Bestandteile des XML-Dokumentes durch die Elemente, Attribute und ProcessingInstructions etc. äquivalent abgebildet und stehen zu Bearbeitung durch Java-Methoden bereit. So sind interessante Informationen der XML-Datei als Text von einem Element dargelegt. Die Klasse JDOM-Element ermöglicht den Zugriff auf die Daten des Elements. Weil das Programm alle XML, deren Aufbau bekannt oder unbekannt sind, lesen soll, verwende ich die Methode *getChildren().* Diese Methode liefert alle eingeschachtelten Kinderelemente eines Elementes samt ihrer Informationen.

Die Vorüberlegungen für den Programmablauf sind Folgende: Es liegt eine unbekannte XML-Datei vor. Es ist nicht bekannt, wie tief die Elemente verschachtelt sind, wie die Elemente benannt wurden und welche Informationen sie enthalten. Die Java *Interator Schnittstelle* ermöglicht es, in einem XML-Dokument Ebene für Ebene einzusteigen und mit der Funktion *public boolean hasNext()* zu ermitteln, ob das Dokument weitere Verschachtelungen hat. Mit

der *public Objeckt next()*-Funktion ist es möglich, das nächste Element auf der gleichen Ebene zu ermitteln. Bei der Verwendung der rekursiven Funktion zusammen mit der Interator-Schnittstelle gelingt es, an alle Stufen und Elemente einer unbekannten XML-Datei zu gelangen. Von Interesse sind dabei nur diejenigen Elemente, die einen Text enthalten. Sobald zwischen den Element-Tags ein Text steht, so wird der Text zusammen mit dem Elementnamen und den Elternnamen in einem Zwei-Element-Array gespeichert (siehe auch Kapitel 2.2.2). Anschließend wird das Array in einem Vector gespeichert und zusammen mit dem internen Zähler an das nächste Programm übergeben.

Das zweite Programm parst die content.xml-Datei mit dem Ziel, die Vector-Inhalte einzufügen. Wie in Kapitel 4.6 beschrieben, enthält content.xml-Datei den eigentlichen Inhalt eines unter OOo erstellten Dokumentes. Aus den Kapitel 4.6.2 ist bekannt, dass die Textabschnitte durch das Elementpaar <text:p> </text:p> als Kinderelemente des Elements <office:body> dargestellt werden und diese wiederum ein Element <database-display> enthalten, wenn dieses Feld ursprünglich als Seriendruckfeld mit der MERGIEFUNKTION erstellt wurde. Mit diesen Vorüberlegungen schränkt sich die Suche auf ein paar bestimmte Elemente ein, wodurch sich die Programmgeschwindigkeit erhöht.

JDOM enthält zudem Funktionen, die einen Zugriff auf die Elemente mittels der Namenspace oder den Elementnamen bietet. Wird ein Element <database-display> gefunden, so wird der Text dieses Elements mit dem im Vector enthaltenen Elternelementnamen verglichen. Stimmen sie überein, so wird der im Vector gespeicherte Text den <database-display>-Text überschreiben. Die Dokumentebenen werden mit Funktionen der Interator-Schnittstelle durchsucht. Die Vector-Informationen werden dabei nur in die vordefinierten Felder eingefügt. Unter anderem erzeugt JDOM bei der Erstellung des DokumentObjectes automatisch ein DocTyp-Object, der mit der DTD-Reference aus der Quelldatei übereinstimmt.

5.3.2 JDOM-Dokument-Ausgabe

Die Verwendung von JDOM hat weitere Vorteile: Sobald ein DokumentObject aufgebaut ist, bestehen keine weiteren Verbindungen zu SAX, DOM und keine Kopplungen an das Ausgabeformat. Das heißt: Auch wenn die verwendeten Dateien aus OOo stammen, können sie wie jede andere Datei verarbeitet und bei der Ausgabe an die Applikationskomponente als eine wohlgeformte XML-Datei ausgegeben werden. Für die Dokumentausgabe hat JDOM

ebenfalls eine Hilfsklasse *org.jdom.output*. Ein Dokument kann dann in diversen Zielformaten (z.B. in eine Datei, auf die Konsole oder in ein URI) ausgegeben werden.[32]
Um ein Dokument in eine Datei auszugeben, müssen ein JDOM DokumentObjekt und eine Datei content.xml mit dem FileOutputStream erzeugt werden. Der JDOM XML-Outputter gibt den Inhalt des JDOM-Dokumentes in das File content.xml aus. Vorzugsweise wählt man das Verzeichnis in dem die restlichen Dateien des ursprünglich entzippten SXW-Dokumentes ebenfalls gespeichert sind. Denn um ein Dokument in im Druckformat zu bekommen, müssen die Dateien gezippt werden. Damit beschäftige ich mich im folgenden Kapitel.

5.3.3 Daten-Kompression und -Dekompression in Java.

Um an die XML Ressourcen der OOo-Writer zu gelangen, müssen die Dateien zunächst entzippt werden. Diese Aufgabe erfüllt ein Kompressionstool wie zum Beispiel WinZip unter Windows oder andere Tools unter Unix/Linux-Betriebssystemen. Die Komprimierung unter Java unterstützen die Klassen die java.util.zip und java.util.tar, java.util.jar als Erweiterung von Zip-Paketen. GZip/GunZip setzt man zum Komprimieren und Dekomprimieren für Datenströme ein und Zip nutzt man zum Behandeln von Archiven und Komprimieren von Dateien.

Tar-Tool komprimiert die Dateien nicht, sondern bündelt mehrere Dateien zur einen Gesamtdatei. Um eine höhere Kompressionsrate zu erzielen, werden die Tar-Archive anschließend mit gzip bzw. mit bzip2 noch mal gepackt. Bei der Komprimierung eines Tar-Archivs kann man die Redundanz besser ausnutzen. Dieses Komprimierverfahren bietet keinen freien Zugriff auf Archivdateien, so dass man das komplette Tar- Archiv dekomprimieren muss, um an eine Datei zu gelangen. Gleichzeitig bietet das GZip-Verfahren bessere Komprimierungsraten, als wenn die Dateien nach dem Zip-Verfahren einzeln gepackt würden.

Gründe für die Wahl des Zip Paketes: Tar ist kein Komprimierungstool, GZip/GunZip kann jeweils nur eine Datei komprimieren oder dekomprimieren. Somit fallen die beiden Tools weg. Eindeutige Vorteile des Zip-Verfahrens sind die Archivierung mehrerer Dateien und der wahlfreie Zugriff auf den Archivinhalt. OOo-Dateien lassen sich mit dem WinZip-Tool bearbeiten, das intern auch die Zip-Komprimierung unterstützt. Weitere Vorteile sind die Kombination von Dekomprimieren mit WinZip und Komprimieren mit dem Java-Zip-Programm.[33]

---

[32] Vgl. McLaughlin B.(2001) S. 229

[33] [33] Ullenboom C.,Java ist auch eine Insel,
http://www.galileocomputing.de/openbook/javainsel4/javainsel_12_011.htm#Rxx365java12011040004011F02C100, 11.08.2005

5.3.4 Arbeitsweise der Komprim.java Programm

Mit einem ZipOutputStream wird in der Regel eine Zip-Datei erzeugt. In diesem Programm wird eine SXW-Datei erstellt, die alle OOo-Dateien intern in einem Zip-Archiv darstellt. Die Dateien aus dem Zielverzeichnis werden mit dem Programm Komprim.java in einen Array eingelesen und an einen FileInputStream weitergegeben. Mit dem ZipEntry-Objekt wird eine gleichnamige Datei in ZipArchive erstellt und der Komprimierungsgrad der Zip-Datei mit der Methode *setLevel();* auf acht eingestellt. Anschließend wird der Datenstrom aus dem InputStream in den ZipOutputStream umgeleitet. Sobald alle Dateien aus dem Zielverzeichnis eingelesen wurden, werden die beiden Streams geschlossen, dabei entstehet eine SXW-Datei, die neben dem Layout auch die Informationen aus der ersten XML-Datei enthält.

Nach diesem Schritt sind die grundlegenden Anforderungen an meine Aufgabe erfüllt:
Ein von Kunden erstelltes Layout wurde mittels weltweit verbreiteter Software in einem Zwischenschritt (Konvertierung zum OOo-Format mit Hilfe von Standard-XML-Technologien) verarbeitet und schließlich in ein Druckformat ausgegeben.

Die folgenden Kapitel erläutern das Konvertieren von Writer-Dokumenten zu den anderen gängigen Formaten Word 97, PDF und HTML - und die Ausgabe dieser an einem Drucker.

5.3.5 Drucken und Konvertieren von Dokumenten

Drucken und Konvertieren von Dokumenten ist auf zwei Wegen Möglich.

Erstens: Aus der openOffice.org Writer-Anwendung heraus, wenn das Programm auf dem lokalem Rechner läuft.

Alternativ: Wenn Java Applikation und Office Packet nach einem Client Server Prinzip miteinender kommunizieren. Dafür benötigt man ein in Office API enthaltene UNO-Komponentenmodell.

Die Office API besteht aus sechs Ebenen. Die Interaktionen mit OpenOffice-Objekten sind unter den Betriebssystemen Linux, Windows und Solaris möglich und basieren auf Java Runtime Maschine. Für diese Betriebssystem-Plattformen werden VOS (Virtual Operating System) und VCL (Visual Class Library) als abstrakte Systemschicht verwendet. VOS versteckt alle Details

der Implementierung des darunter liegenden Betriebssystems, während VCL die Schnittstelle zu den verschiedenen Fenstersystemen darstellt. Unterhalb der abstrakten Systemschicht werden verschiedene Objekttechnologien unterstützt. Dies ist neben dem Microsoft-Windows-Standard COM/OLE und neben CORBA vor allem das Universal Network Object Model (UNO). Das UNO-Komponent-Modell verfügt über Basis-Bibliotheken für die Sprachen C++, Java, Phyton, und OpenOffice Basic, ebenfalls ist die Programmierung über die Microsoft-COM-Schnittstelle mit VB, Delphi, PHP möglich.

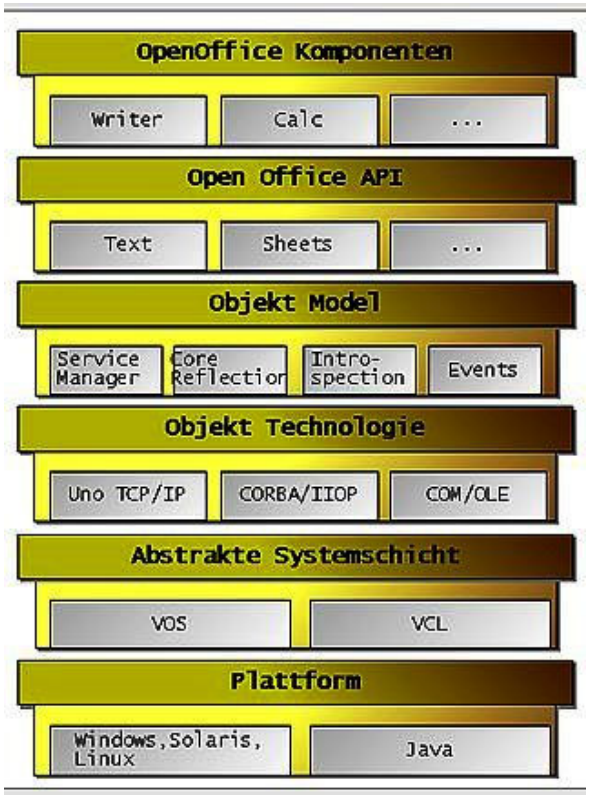

Abbildung 9: OpenOffice API

Das Komponentenmodell hat das Ziel, die Kommunikation zwischen verschiedenen Programmiersprachen, Objektmodellen, Maschinenarchitekturen und verschiedenen Prozessen zu ermöglichen. Als Einsatzgebiet für UNO sollen neben einer lokalen Maschine auch ein Netzwerk oder das Internet genutzt werden können. UNO ist kostenlos erhältlich und unterliegt der LGPL-Lizenz.[34]

[34] Scherm B./ Sröckel J.(2002), Java meets Office, Nutzung von Office Produkten als Server-Anwendung durch Java-Clients, http://www.javamagazin.de/itr/online_artikel/psecom,id,224,nodeid,11.html, 11.03.225

5.3.6 Zugriff auf OpenOffice.org-Objekte

Die Objekte, die das System repräsentieren, werden unter UNO über Factory Service Manager erzeugt. Jeder Service Manager existiert wiederum in einem Component-Context, das jeweils eine OpenOffice.org Applikation (z.B. Writer, Calc etc.) darstellt. Die einzelnen Services sind die abstrakten Objektinstanzen samt ihrer Schnittstellen und Eigenschaften. Es kommen im Wesentlichen zwei Services zum Einsatz: *com.sun.star.frame.Desktop* und *com.sun.star.document.OfficeDocument.*

Der Service *com.sun.star.frame.Desktop* ist ein Stammservice von OpenOffice.org. Er stellt die Funktionen für das Rahmenobjekt von allen OOo-Dokumenten zur Verfügung, auch das Erzeugen, Öffnen und Importieren von Dokumenten läuft über diesen Service. *Com.sun.star.document.OfficeDocument* Service liefert die Basisfunktionalität für einzelne Dokument-Objekte. Dazu gehören die Methoden zum Speichern, Exportieren und Drucken von Dokumenten. Ein Service ist etwa das com.sun.star.document.Office-Document. Es stellt einen Writer-Dokument dar, dass ein Interface com.sun.str.view.XPrintable besitzt. Die Kommunikation in UNO definierten Objekten erfolgt durch Aufruf der in den Interfaces definierten Methoden *getPrinter(), setPrinter()* und *print().*

Zum Laden eines Dokuments erzeugt OOo ein Objekt mit globalen Namen *StarDesktop*. Der eine Schnittstelle *com.sun.star.frame.XComponentLoader* liefert die zum Erzeugen, Exportieren und Öffnen von Dokumenten geeignet ist. Mit der Methode *StarDesktop.loadComponentFromURL();* lassen sich beliebig viele Dokumente laden und bearbeiten. Die Funktion loadComponentFromURL liefert wiederum ein Dokument-Objekt zurück, der den Service *com.sun.star.document.OfficeDocument* unterstützt. Die beiden Schnittstellen sind für das Speichern *com.sun.star.frame.XStorable* und Ausdrucken *com.sun.star.view.XPrintable* von Dokumenten zuständig.

Das Speichern von Dokumenten erfolgt über die Funktion *storeAsURL*, wenn dem Dokument noch kein Speicherplatz zugewiesen wurde, sonst mit der Methode *store().* Den Optionen der *storeAsURL*-Funktion kann man den Zeichensatz, Filter oder die Übergabe des Passwortes für die geschützte Datei zugrunde legen. Ähnlich wie das Speichern, erfolgt auch das Ausdrucken von Dokumenten direkt über das Dokumentobjekt. Die Methode *print()* bietet die Optionen zur Auswahl, welche die Einstellungen des Druck-Dialoges von OpenOffice.org widerspiegelt. Damit sind Druckerauswahl, Festlegung zur Seitenausrichtung, Papierformat usw. möglich.[35]

[35] Vgl. Sun Microsystem S.69-87

6 Fazit

Der Computer ist mittlerweile aus unserem Leben nicht mehr weg zu denken und ist als ein Kommunikationsmedium anerkannt. Natürlich wird von einem Computer, wie von einem Gesprächspartner auf eine Anfrage auch eine Antwort erwartet. Dauert die Rückmeldung zu lange, verläuft das Gespräch langwierig. Ähnlich verhalten sich die Menschen bei dem Umgang mit dem Computer. Mögen Programme ausgeklügelt und Technologien noch so ausgereift sein, sie bleiben uneffektiv, solange sie anwenderunfreundlich sind. Anwenderfreundlichkeit ist daher das erste Gebot.

In dieser Arbeit wurde hauptsächlich Augenmerk darauf verwendet, dass Datenabnehmer nicht mit der Ausgabeformatierung überfordert werden. Schließlich hat man sich bereits mit Einführung des SAA-Standards daran gewöhnt, dass die meisten Programme sich ähneln und das soll natürlich gemäß dem WYSIWYG Prinzips auch bei der Ausgabe so sein.

Die Aufgaben der Anwender beschränken sich auf sinnvolle Platzierung von Feldern in der Dokumentvorlage. Damit verläuft die Phase der Layouterstellung anwenderfreundlich und zeitsparend. Also haben die Kunden noch mehr Zeit für die eigentlich kreative Phase. So manches Layout nimmt schließlich einen langen Entstehungsweg bis es häufig verworfen wird.

Die in dieser Arbeit gemachten Erkenntnisse führen zum Schluss, dass für den Zweck der „Entwicklung eines Verfahrens zur layoutgetreuen Dokumenterstellung unter Einsatz von XML- und Java- Technologien“ das zusammen gefasste Verfahren am sinnvollsten ist. Es dient der zeitgemäßen anspruchsvollen Layouterstellung der XML-basierten Dokumente.

In dem Verfahren kann das Design des Dokuments unmittelbar mit den Ergebnissen auf dem Bildschirm verglichen werden. Die bequeme Erstellung der Dokumente anhand von Dokumentvorlagen ist dabei sehr einfach. Der Export von MS Word zum OpenOffice.org Writer ist auch problemlos umsetzbar und bietet letzten Endes eine getreue Layoutdarstellung. Vorteilhaft erweist sich, dass das interne XML-Dateiformat von Openoffice.org sich mit allen XML Werkzeugen auf traditionellen Wegen bearbeiten und wieder ins XML Format ausgeben lässt. Das Komprimieren und Dekomprimieren von OpenOffice.org Dateien lässt sich einfach mit WinZip oder mit dem integrierten Komprimierungsprogramm handhaben. Die Dokumente erscheinen sofort in einem druckbaren Format. Über die OpenOffice.org Schnittstelle werden die Dokumente in weitere Formate PDF, MS Word 97 konvertiert und gedruckt.

Beim Einsetzen dieses Verfahrens fallen keine zusätzlichen Investitionen an. Das Konzept bietet Erweiterungs- und Optimierungsmöglichkeiten des Tools, sowie flexible Gestaltung des Konvertierungs- und Druckvorgangs und Kombinationen mit anderen OpenSource-Programmen. Zum Beispiel kann Konvertierung nach PDF auch mit dem MAKE2PDF Programm erfolgen.

Automatisierungen des Mauellen Vorganges und Unterstützung weiterer Feldtypen bei der Dokumentvorlagenerstellung sind als zukünftige Entwicklungsschritte vorzuschlagen. Zum Beispiel die Entwicklung eines Tools, das aus dem Inhalt einer XML-Datei die Dokumentvorlagen Felder generiert. Die Feldnamenerstellung sowie Festlegung der Feldfunktionen sollten dabei automatisiert werden.

Literaturverzeichnis

Bundesministerium des Innern(2005):
Interoperabilität von Office-Anwendungen auf Basis von XML, 1. Aufl., Band 73, Berlin 2005

Freund + Dirks, Focusing on IT Professionals:
Teilnehmerunterlagen, XML Grundlagen

Lemay L./ Cadehead R.(2002):
JAVA™ 2 in 21 Tagen, 2. Aufl., München u.a. 2001

McLaughlin B.(2001):
Java und XML, 1. Aufl., Köln u.a. 2000

Melster R./ Mosconi M./ Pfeiffer C. u.a.(2003):
Einsatz und Nutzung Integrierter Software- Entwicklungsumgebungen (IDEs)

Mistlbacher A./Nussbaumer A. (2002):
XML ENT-PACKT, Paperback 2002

Scherm B./ Stökel J. (2002):
Offener Standard für Dokumentenaustausch, 1.Aufl, Berlin 2005

Sun Microsystem:
StarOffice™ 7 Office Suite, ASun™ ONE Software Offering, Basic Programmierhandbuch, Santa Clara

Sun Microsystem (2002):
OpenOffice.org XML File Format 1.0, Technical Reference Manual, 2. Version
San Antonio Road 2001

Quellen aus dem Internet

Hachmann B.:
Das JDOM Projekt, http://www.jdom.net/dom3.htm, 03.05.2005

Jastram C., (2003):
Java meets PDF, http://www.javamagazin.de/itr/online_artikel/psecom,id,441,nodeid,11.html, 11.08.2005

Langham M. / Ziegeler C.(2002):
Die Verpuppung, XML-Anwendungen erstellen mit dem Coccon.2, Teil1, http://www.javamagazin.de/itr/online_artikel/psecom,id,154,nodeid,11.html, 03.07.2005

Scherm B./ Sröckel J.(2002):
Java meets Office, Nutzung von Office-Produkten als Server-Anwendung durch Java-Clients, http://www.javamagazin.de/itr/online_artikel/psecom,id,224,nodeid,11.html, 11.03.225

Ullenboom C.(2004):
Java ist auch eine Insel, http://www.galileocomputing.de/openbook/javainsel4/javainsel_12_011.htm#Rxx365java12011040004011F02C100, 11.08.2005

„o.V.“:
XSL Report Designer (Antenna House) – Produktbeschreibung, http://www.mid-heidelberg.de/produkte/entwicklung/XSL-template-designer.htm, 02.03.2005

„o.V.“:
Java (Programmiersprache)**,** http://de.wikipedia.org/wiki/Java_%28Programmiersprache%29, 05.06.2005.

„o.V.“(2005):
http://www.golem.de/0412/35186.html Sun NetBeans 4.0 vorgestellt, 11.08.2005

“o.V.”:
XML Werkzeugkisten, http://www.tecchannel.de/entwicklung/programmierung/401578/index9.html, 11.08.2005

„o.V.“ (2005):
Crystal Reports, http://www.softguide.de/prog_m/pm_0989.htm,11.08.2005