Institut für graphische Datenverarbeitung

XML Filter mit Openoffice 1.x

am Beispiel der EML und des E-Learning System ELAT

Axel Pospischil

Oktober 2004

Version 1.2

www.blue.it.org

Zusammenfassung

In dieser Arbeit soll es darum gehen, wie Schriftdokumente - insbesondere solche, die mit Officesystemen erstellt wurden - in E-Learningsysteme integriert werden können. Die inhaltlichen und strukturellen Informationen dabei zu bewahren, soll oberste Priorität haben. Es werden von mir nur offene Standards und Software eingesetzt, um eine innovative Weiterentwicklung und Integration in andere offene Software zu ermöglichen. Die Ergebnisse werden daher unter einer offenen Lizenz veröffentlicht. Das Modell soll leicht an gängige XML Standards adaptiert werden können und Schnittstellen zu gängigen E-Learningsystemen bieten. Als Referenzapplikation dient ein Konverter für OpenOffice/StarOffice Präsentationen für die E-Learning-Plattform ELAT.

Inhaltsverzeichnis

1 Changelog

Version 1.2 (Februar 2012) Kleinere Korrekturen und Fehlerbereinigung

Version 1.1 (Juli 2006) Überarbeitung. Kleinere Korrekturen. Titeländerung der Arbeit. Originaltitel: ”Schnittstellen bei Office Applikationen im Umfeld von E-Learning System”

Version 1.0 (Oktober 2004) Arbeit für das Institut für grafische Datenverarbeitung im Ramen eines 3 monatigen Praktikums.

2 Vorwort

Stellen Sie sich folgendes Szenario vor: Sie sind im Umfeld von Fort- und Weiterbildung tätig und möchten ihr Wissen elektronisch einem ausgesuchten Publikum zur Verfügung stellen. Sie besitzen das nötige Know How und auch die Kursmaterialien für die geforderte Aufgabe stehen Ihnen bereits zur Verfügung. Da sie bereits seit einigen Jahren im Geschäft sind und immer ihre Daten elektronisch erfasst haben, haben diverse Computersysteme und Dokumentformate Ihren Schreibtisch gesehen. Bisher ist also alles wie immer.

Ihr Auftraggeber möchte aber die Schulung innerhalb von 2 Wochen starten und besitzt bereits ein modernes Ausbildungszentrum mit entsprechender Ausstattung.

Zu ihren Auszubildenden gehören sowohl fest angestellte Mitarbeiter als auch freiberufliche, die sich jedoch nicht vor Ort befinden. Weil es sich um hoch bezahlte Spezialisten handelt, die nur in einem sehr begrenzten Zeitrahmen an Ort und Stelle zur Verfügung stehen, sollen der Kurs und die Leistungsnachweise zum Teil online stattfinden. Ein sehr heterogenes Publikum also.

Nun stellt Ihnen der zuständige Ingenieur das E-Learning System vor, in das Sie Ihre Kursdaten übertragen sollen. Es hört sich alles ganz vernünftig an, Wissensbausteine, Videounterstützung, interaktives Lernen. Beiläufig wird noch erwähnt, dass die Daten in XML-Dateien erfasst werden.

Sie denken sich vorerst nichts dabei, doch in den nächsten Tagen werden Sie graue Haare bekommen, wenn Ihnen die Inkompatibilitäten Ihrer Dokumenttypen und dieses E-Learning Systems bewusst werden.

Ihre didaktisch wohl durchdachten Powerpointpräsentationen werden zu einfachen Diashow ähnlichen Quicktimemovies reduziert. Ihre wohl strukturierten und mit zahlreichen Querverweisen versehenen Worddokumente sind mit der Version des Importfilters inkompatibel und Sie müssen alle Dokumente nochmals erstellen oder mit erheblichem Zeitaufwand korrigieren.

Teil I

Dokumententypen im Office und E-Learning Umfeld

Kapitel 1
Dokumententypen in Officeprogrammen

am Beispiel OpenOffice

Hinweis zu Abkürzungen:

- *OpenOffice.org* als Office Suite Softwarepaket wird im Folgenden als **OOo** bezeichnet.
- Die Textverarbeitung *Open Office Writer* möchte ich als **OOW** oder **OO Writer** bezeichnen.
- Das Präsentationsprogramm *Open Office Impress* wird mit **OOI** abgekürzt werden.
- Das Office Programm *Microsoft Word* wird als **Word** bezeichnet.
- Das E-Learning Programm *Environment for Learning and Teaching* ist **ELAT**.
- Die *Educational Modeling Language* ist **EML**.

1 Einleitung

Das Management von Dokumenten streift viele Teilgebiete der Informationswissenschaften, die nicht alle der Informatik angehören. Bedenkt man, dass erst Ende der 90er Jahre internationale Standards zur Speicherung von Dokumenten (XML) verabschiedet wurden und die große Masse der Dokumenten Erfassungswerkzeuge gerade erst beginnt, ernsthaft diese Standards zu nutzen, so ist es nicht verwunderlich, das wir Anfang des 20. Jahrhunderts in großen Teilen unserer

Dokumentenlandschaften immer noch mit geradezu steinzeitlich anmutenden Methoden arbeiten. Diese Arbeit hat mir gezeigt, wie viele verschiedene Aspekte für ein zukunftssicheres Dokumentenmanagement zu berücksichtigen sind und ich bin weit davon entfernt, eine allumfassende Lösung zu erkennen. Die größten Schwierigkeiten bestanden darin, die verschiedenen Teilbereiche Softwaretechnik, Programmierung und die Integration von Dokumentenstandards, wie XML - welches ein Universum für sich darstellt - unter einen Hut zu bringen und informationstechnisch korrekt umzusetzen.

In dieser Arbeit soll es darum gehen, wie Schriftdokumente - insbesondere solche, die mit Officesystemen erstellt wurden - in E-Learningsysteme integriert werden können. Die inhaltlichen und strukturellen Informationen dabei zu bewahren, soll oberste Priorität haben. Es werden von mir nur offene Standards und Software eingesetzt, um eine innovative Weiterentwicklung und Integration in andere offene Software zu ermöglichen. Die Ergebnisse werden daher unter einer offenen Lizenz veröffentlicht. Das Modell soll leicht an gängige XML Standards adaptiert werden können und Schnittstellen zu gängigen E-Learningsystemen bieten. Als Referenzapplikation dient ein Konverter für OpenOffice/StarOffice Präsentationen für die E-Learning Plattform ELAT. An vielen Stellen werde ich nur Ansätze zur Lösung liefern können, da ich hier selbst völliges Neuland betrete.

Diese Arbeit steht weiterhin ganz unter dem Motto *OpenSource* und *offene Standards*. Ich werde daher nicht nur auf die softwaretechnischen Details eingehen, sondern auch allgemeine Aspekte zu Anwendungsfällen bei der Erstellung und Konvertierung von Office Dokumenten aufzeigen. Mein vorrangiges Ziel ist es, Referenzplattformen wie OpenOffice und international einheitliche Standards in den Vordergrund zu rücken.

Als Werkzeuge kommt dabei vom Editor bis zur Entwicklungsumgebung und dem Betriebssystem ebenfalls nur Software zum Einsatz, die entweder der GPL oder ähnlichen offenen Lizenzen zur Verfügung stehen.

Ich muss leider auch an dieser Stelle - denn ich bin es schon gewohnt - darauf aufmerksam machen, dass dies nicht geschieht, um Geld zu sparen, sondern aus der Überzeugung heraus, dass langfristig eine moderne digitale Medienlandschaft nicht mit proprietären Software- oder Dokumentenstandards zukunftssicher gepflegt werden kann, auch wenn viele proprietäre Lösungen auf den ersten Blick scheinbar perfekte Lösungen anzubieten scheinen.

Vieles von dem was hier zur Sprache kommt, beschäftigt mich schon seit Ende der 80er Jahre, als ich auf einem Apple IIe und einem Commodore 64 (jeden Rechner hatte ich leihweise je 4 Wochen zur Verfügung), eine Facharbeit in Physik[1] zu schreiben begann. Der Leser mag sich ausmalen, welche Schwierigkeiten damals die Umstellung

[1] Online einsehbar unter apos.de -> Themen -> Theorie der Kernreaktoren http://www.apos.de

vom einen ins andere Format machte - besonders die Sonderzeichen und Formeln bereiteten große Schwierigkeiten - vom Abenteuer Ausdruck gar nicht zu reden. Von da an begleiteten mich PC's in vielen Tätigkeitsbereichen, vom Serienbrief über Tabellenberechnungen bis hin zu grafisch anspruchsvollen Dokumenten. Es gelang mir bisher, alle Dokumente bis auf den heutigen Rechner zu retten, jedoch nur, in dem ich Sie immer wieder regelmäßig konservierte, konvertierte, alte Software zur Not immer noch vorliegen hatte und niemals Word verwendete.

1.1 Internationale Standards versus proprietäre Standards

XML als internationaler Standard für Dokumente

Dokumente, die Jahre, vielleicht Jahrzehnte überdauern sollen, müssen in Formaten gespeichert sein, die weitblickend entworfen und international gepflegt werden. Industrielle Softwareprodukte mit meist kurzen Produkt- und Entwicklungszeiten werden meist nicht unter den Gesichtspunkten der Standardisierungsgremien, wie dem W3C `http://www.w3c.org` entworfen. Die wenigsten Softwarefirmen der Vergangenheit waren willens oder in der Lage, Mitarbeitern in die langwierigen Entscheidungs- und Evaluierungsprozesse über *RFC*'s (s. Glossar 6 auf Seite 59) einzubinden und so die Arbeit dieser Gremien in die Produkte einfinden zu lassen. Erst durch massiven Druck der Opensource Gemeinde und Firmen wie *SUN* (s. 1), aber auch durch gerichtliche Entscheidungen müssen nun rein kommerzielle Softwarehersteller im Interesse des Allgemeinwohls ihre Schnittstellen offen legen. Verständlich war das "closed source" Gebaren von Softwareherstellern in einer Zeit, in der Software alles und Service wenig bedeutete, in der Insellösungen an der Tagesordnung waren und der Kunde so gut wie alles kaufte, was seinen kurzfristigen Bedarf zufriedenstellte! Heute sind jedoch nicht nur die Kunden schlauer geworden, sondern Sie blicken auf den harten Alltag der vergangen Jahre zurück, in dem eine immer größere Anzahl wichtiger Firmendokumente aufbewahrt, weitergegeben und archiviert werden mussten und zahlreiche proprietäre Softwarestandards so manche Konvertierung vereitelten und die Pflege der Dokumente einem erneuten Erstellen an Aufwand fast gleichkam.

Zudem müssen angesichts immer umfangreicherer Standardsoftware aus dem *Opensource-Bereich* (s. Glossar 6 auf Seite 59) und dem *Microsoft* "de facto" Standard bereits namhafte Firmen, wie *Corel* (WordPerfect) oder *Lotus, jetzt IBM* (Smartsuite) erkennen, dass Sie mit Ihren Dokumentenformaten über ein Nischendasein nicht hinauskommen werden.

Die heterogene Landschaft von Informationsdokumenten und die zunehmende Notwendigkeit der Vermischung unterschiedlicher Dokumententypen macht zudem die firmeneigene, nicht öffentliche (proprietäre) Schnittstelle immer unsinniger. Ja sie war es eigentlich schon im-

mer, jedoch sahen viele Softwarefirmen ihre einzige Möglichkeit darin, ihr geistiges Eigentum zu schützen - meist auf Kosten der Endkunden - die nach einiger Zeit der Nutzung der Software feststellen mussten, dass Sie in eine Einbahnstraße geraten waren, aus der es nicht einmal mit Hilfe des Herstellers einen Ausweg gab. Zudem stellen immer noch viele Firmen die mannigfaltigen ”Fähigkeiten” Ihrer Software in den Vordergrund, ohne genügend Wert auf die Schnittstellenproblematik zu werfen.

1.2 Das Open Office Dokumentenformat als De Facto Standard?

Doch es ist Land in Sicht. Die Firma *SUN* - allen voran - schickte sich an, zusammen mit einer großen OpenSource-Gemeinde mit StarOffice/OpenOffice einen Dokumentenstandard zu erarbeiten, der den Grundforderungen an die heutigen Verfahren zu Verarbeitung und Speicherung von Dokumenten genügen sollte:

1. Der Standard sollte sich auf einfache Weise nutzen lassen.
2. Ein breites Spektrum von Anwendungen sollen unterstützt werden.
3. Er muss kompatibel sein.
4. Es soll einfach sein, Programme zu schreiben, die diese Dokumente verarbeiten.
5. Die Anzahl optionaler Merkmale soll minimal sein, idealerweise Null.
6. Dokumente sollten für Menschen lesbar und angemessen verständlich sein.
7. Die Beschreibung sollte formal und präzise sein.
8. Dokumente sollen leicht zu erstellen sein.
9. Knappheit des Markup ist von minimaler Bedeutung, da Textdateien sehr gut gepackt werden können.

Man einigte sich, wie sollte es anders sein, auf XML. Wem es noch nicht aufgefallen sein sollte, ich erlaubte mir die obigen 9 Punkte genau den Anforderungen zu entlehnen, demnach seinerzeit XML entwickelt wurde. Sie sind allgemein für das Speichern von Dokumenten im informationstechnischen Umfeld gültig. Ich möchte hier davon ausgehen, dass der Leser dieser Arbeit mit XML vertraut ist bzw. das Internet bemüht, um sich damit vertraut zu machen und werde auf diese Spezifikation nicht weiter eingehen (siehe Anhang: 1 auf Seite 62).

Was die Dokumente von StarOffice/OpenOffice angeht, so einigte man sich zusätzlich auf die Tatsache, das ausnahmslos alle Dokumente einer einzigen Spezifikation[2] genügen sollten. Man erkannte, dass die Zukunft des Dokumentenmanagements sehr heterogen sein wird, und möchte eine zentrale Schnittstellenspezifikation anbieten, die den Anforderungen zukünftiger Informationslandschaften genügt.

Eine Ausnahme bildet die Beschreibung mathematischer Inhalte. Hier greift man auf den speziellen und bereits hervorragend ausgereiften XML Dialekt, den MathML http://www.w3.org/Math/ zurück.

Auch die immer größer werdende Gemeinde der Koffice Suite des weit verbreiteten Linux KDE Desktop Environments und die Gnome-Office Gemeinde haben sich diesem Standard angeschlossen. Man hat hier also den Schritt hin zu einem einheitlichen Officeformat vollzogen. Die *IDA expert group* http://europa.eu.int/ISPO/ida/ der europäischen Kommission spricht sich ebenfalls für eben dieses Format aus (IDA [2004]):

> ”Industry has taken important steps to address the requirements and concerns of the public sector regarding the use of document formats. The publication of the OpenOffice.Org and WordML formats has greatly improved the potential for interoperability of document processing.”
>
> ”Die Industrie hat wichtige Schritte zurückgelegt, um die Anforderungen und Bedürfnisse des öffentlichen Bereichs zu berücksichtigen, wenn es um die Verwendung von Dokument Formaten geht. Die Veröffentlichung der OpenOffice.org und WordML Formate hat das Potential zur Zusammenarbeit im Dokumentenmanagement erheblich erhöht.”

Die *Organisation for the Advancement of Structured Information Standards* (OASIS) hat es sich zur Aufgabe gemacht, die Entwicklung eines offenen XML-basierenden Dokumentenstandards zu betreuen. Hier befindet sich die Kernzelle auf dem Weg zu einem wirklich offenen Standard. Auf den Seiten der OASIS Komitees http://www.oasis-open.org/committees/ kann man entsprechend die aktuellen Aktivitäten zur Open Office Spezifikation verfolgen. Die wichtigsten Ziele des Open Office Dokumentenformates sind demnach:[3]

1. Es muss den Anforderungen an Office Dokumente genügen, die Text, Tabellenkalkulationen, Diagramme und graphische Dokumente enthalten,
2. es muss kompatibel mit der W3C Extensible Markup Language (XML) v1.0 und den W3C Namespaces in der XML v1.0 Spezifikation sein,

[2] OOo XML Spezifikation http://xml.openoffice.org/xml_specification_draft.pdf, siehe auch Vogelheim [2004]

[3] Original in englischer Sprache, Übersetzt vom Autor, siehe Anhang 6 auf Seite 69.

3. es muss einen hohen Grad an Informationen beibehalten können, um für das editieren von Dokumenten nützlich zu sein,
4. es muss leicht sein, Transformationen mit XSLT oder anderen XML-basierten Sprachen und Tools durchzuführen,
5. es muss den Inhalt und das Layout von Dokumenten getrennt aufbewahren, so dass beide Teile unabhängig von einander verarbeitet werden können, und
6. es soll sich an ähnliche existierende Standards anlehnen, wo immer das möglich und erlaubt ist.

Eine interessante Untersuchung zum Thema findet der Leser in Thadden and Hetze [2002]. Das Bundesministerium für Sicherheit in der Informationstechnik hat eine Studie bei der Linux AG in Auftrag gegeben, deren über 100 seitiges Endergebnis viele Aspekte von Open-Source und offenen Standards betrachtet. Hier wird besonders auch der Sicherheitsaspekt eine Dokumentenformats detailliert erörtert.

An dieser Stelle sei nur angemerkt, dass Behörden eine Aufbewahrungspflicht von Dokumenten bis zu 60 Jahren haben und Standards dementsprechend lange lesbar sein müssen!

Und Microsoft?

Die Firma Microsoft hat nun endlich nach jahrelangem Tauziehen[4] mit Office 2003 drei verschiedene (!) Standards auf den Markt gebracht, die zwar offen liegen, leider nicht zueinander voll kompatibel sind, jedoch den Anforderungen an ein wirklich offenes Dokumentenformat im oben genannten Sinne weitestgehend entsprechen. Noch Ende des Jahres 2002 war keine endgültige Aussage aus Redmond zum Thema XML zu hören! Bisher war eine Konvertierung von Microsoft Officedokumenten nur durch proprietäre Blackbox DLLs möglich, die letztendlich die Dokumenteninhalte im Microsoft eigenen RTF Format speicherten. Andere Firmen, wie Lotus bildeten da keine Ausnahme.

Löblich ist also das Ansinnen zu nennen, XML als Standard zu benutzen jedoch die Umsetzung zeigt, dass hier wieder Schranken errichtet werden, wo keine sein sollten. Die XML Referenz der *OfficeML*-Schema Familie, wie Microsoft seine ”patentierte” Schöpfung von ”Dokumenteninhalten basierend auf XML in *einer* Datei” nennt, finden sich auf einer eigenen MS Office Homepage `http://www.microsoft.com/office/xml`, welche leider zu 80% dazu einlädt, sich nicht mit dem eigentlichen Schema, sondern linzenzrechtlichen und patentrechtlichen Fragen auseinanderzusetzen[5]. Ich werde hier auf die Microsoft eigenen Schemas nicht eingehen, da Sie in meinen Augen auf Dauer den

[4] siehe Schüler [2002] und auch Thadden and Hetze [2002]

[5] Man möge dem Autor seine offensichtlich einseitige Meinung in diesem Punkte verzeihen, doch er ist der Ansicht, dass Firmen, die in diesen Größenordnungen Software entwickeln in noch größerem Mass auch zur lizenzrechtlichen Freigabe

Standardisierungsanforderungen der IDA, sowie der ISO nicht standhalten werden können und zudem immer Linzenzproblematiken für die Entwicklung freier Software einhergehen werden[6].

1.3 Was ein modernes Dokumentenformat leisten muss

Eine der wichtigsten Forderungen an ein zukunftsweisendes Dokumentenformat ist die modulare und damit eindeutige Trennung von Inhalt, Form und die Möglichkeit der Speicherung von Metadaten. Dies betrifft jedoch nicht allein Tatsache, dass - wie in allen Markupsprachen üblich - Textinhalte und Formatierungszuordungen eindeutig getrennt werden. Ein modernes Dokumentenformat muss neben dem Inhalt auch folgende Daten mit im Dokument abspeichern:

- Metadaten (über den Autor, Zeitlich relevante Daten)
- Stil (Formate, die den Dokumenten gespeichert werden)
- Applikationsabhängige Daten (z.B. Darstellung auf dem Bildschirm, Darstellungsebene, Druckeinstellungen)
- Zusätzliche Informationen über enthaltene Dateien (MIME Typ, Verschlüsselungsverfahren)
- Bilddateien in ursprünglicher binärer Form
- GUI Informationen zu Dokumentenmakros
- die Dokumentenmakros selbst in einer spezifischen Makrosprache
- Eingebettete Objekte (entweder Fremddokumente im Binärformat, oder XML Dokumente im eigenen Format)

Die genaue Dokumentation des OOo Dateiformats kann auf http://xml.openoffice.org/ eingesehen werden. In Abschnitt Abschnitt 2 auf Seite 10 gehe ich nochmals auf die einzelnen Dokumententypen ein, insbesondere auf Textdokumente und Präsentationsdokumente, da diese im E-Learning Umfeld die bedeutendste Rollen spielen. Ein ausgezeichnete Dokumentation kann man online einsehen: Eisenberg [2004][7]. Ich werde im Folgenden auf für die Konvertierung in andere

solch grundlegender Datenstrukturen gedrängt werden müssen. Strukturen, die ohne das Know How jahrzehntelanger öffentlich finanzierter Forschungs- und Entwicklungsarbeiten unmöglich wären!

[6] siehe Raymond [1998]

[7] Das Buch befindet sich in der Entstehung und ist, wie viele Werke dieser Art unter der Creative Common License http://creativecommons.org/licenses/by-nc-nd/2.0/ veröffentlicht, kann zur Zeit hier http://books.evc-cit.info/ online gelesen und als HTML heruntergeladen werden. In Zukunft wird es als GNU Free Documentation License http://www.fsf.org/licenses/fdl.html weiterhin frei erhältlich sein.

Formate ausschlaggebenden Definitionen des OOo Formates eingehen. Der Leser möge sich bitte die eben genannten Links zum besseren Verständnis zu Gemüte führen.

Die Firma SUN wird nach Aussage ihres Mitarbeiters Tim Bray (Sept. 2004) zudem nicht nur die OfficeML Konverter in StarOffice einbauen, sondern es soll versucht werden, das OOo Format bei der ISO standardisieren zu lassen. Ein, wie ich finde sehr zukunftsweisendes Ansinnen. Zu diesem Zweck wurde das OO-Dateiformat an ein Komitee namens OASIS http://www.oasis-open.org/committees/office/ zur Erarbeitung und Weiterentwicklung übergeben.

Eine Entscheidung...

Ich habe mich nicht nur auf Grund dieser Tatsachen, sondern auf Grund meiner langjährigen positiven Erfahrung mit *echten* offenen Standards und deren hervorragenden (wenngleich auch meist recht komplexen) Dokumentation für das Dokumentenformat von **OpenOffice** entschieden. Im Hinblick auf das Ziel Office Dokumente und E-Learning Applikation miteinander zu verbinden ist es sinnvoll, als Ausgangsbasis das Office Dokument zu nehmen und eine Exportfunktion zu integrieren. Im Gegensatz dazu scheint es mir weniger sinnvoll, die E-Learning Applikation selbst zu verändern oder dort einen Filter zu implementieren, da wohl in den seltensten Fällen ein direkter Eingriff in eine solch komplexe Software möglich ist und die heterogene Landschaft von E-Learningsystemen dem Entwickler kaum die nötigen Informationen zur Verfügung stellt. Doch dazu im zweiten Teil dieser Arbeit mehr.

Ich möchte mich zunächst mit dem OOo Dateiformat etwas detaillierter auseinandersetzen.

2 Open Office Dokumente im Detail

Mit Textverarbeitungs-, Tabellenkalkulations, einfachen Zeichen- bzw. Layoutprogrammen erstellte, sowie für Präsentationen aufbereitete Dokumente bezeichnet das Open Office Format als *Office Dokument*. Dies schließt spezielle Dokumenttypen, wie für mathematische Spezialtexte mit ein. Alle OOo Dokumente haben denselben grundsätzlichen Aufbau (Quelle: Vogelheim [2004]). Alle Inhalt oder Informationen über das Dokument enthaltende Dateien werden in XML-Dateien oder binär in einer Verzeichnishierarchie abgespeichert. Diese wird in eine ZIP Datei gepackt, die je nach Dokument eine unterschiedliche Dateiendung besitzt (siehe Tabelle 1 auf der nächsten Seite).

DATEINAME	INHALT
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, graphische Elemente)**
settings.xml	Einstellungen, die das Dokument und seine Betrachtung betreffen (etwa Darstellungsebene und der ausgewählte Drucker); diese sind in der Regel abhängig von der jeweiligen Applikation
META-INF/ manifest.xml	liefert zusätzliche Informationen über die anderen Dateien (etwa den MIME Typ oder die Verschlüsselungsmethode)
Pictures/	Verzeichnis, welches die Bilder enthält (in ihren ursprünglichen, binären Formaten)
Dialogs/	Verzeichnis, welches die Dialoge enthält, die von den Dokumentenmakros gebraucht werden
Basic/	Verzeichnis, welches die 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 1: Inhalt einer Open Office Datei

Optimierung, Komprimierung von OOo Dokumenten[8]

Es besteht die Notwendigkeit, den doch teils erheblichen Platzbedarf für das *Markup* (siehe Glossar 6 auf Seite 60) zu minimieren. Zudem enthalten Dokumente häufig sogenannte *BLOB's* (binary large objects). Die Daten liegen in unstrukturierter Form *binär* oder z.B. als *OLE* (Object Linking and Embedding) Daten vor. XML Dateien jedoch speichern nur strukturierte Daten oder Verweise auf solche Objekte. Das erfordert das Vorhandensein eines Dateibaums. Dokumente möchte man jedoch gerne in einer Datei vorliegen haben.

Die Anforderung an ein einheitliches Dokumentenformat in einer einzelnen Datei sind danach folgende:

1. *Effizienz*

   Geringer Platzbedarf, Laden von Teilen bei Bedarf (*on demand loading*), unabhängiges Sichern von Unterdokumenten

2. *Kompatibilität mit existierenden Anwendungen*

[8] siehe OpenOffice.org [2004]

Unterdokumente sollten mit existierenden Standardwerkzeugen bearbeitet werden können, dabei spielt der Focus auf XML-Werkzeuge (wie XSLT) eine erhebliche Rolle. Unterdokumente sollten mit existierenden Standardwerkzeugen erreicht werden können, um Sie zu extrahieren, einzufügen und zu verändern. ASCII Standard als Basis für alle Manipulationsaufgaben.

3. *Sicherheit*

   Verschlüsselung und Datenintegrität müssen sichergestellt sein.

4. *Zugriff auf alle Dokumentenbestandteile*

   Ein einfaches Hinzufügen zusätzlicher Unterdokumente und Daten, die nicht notwendigerweise vom XML Standard verstanden werden müssen, sondern proprietärer Software angehören, soll einfach möglich sein. So können Transformationsdaten einfach und direkt im Dokumentenformat mit abgespeichert werden. Die aktuelle Implementierung des OOo Standards ist durch die ***SAX API*** (s. Glossar 6 auf Seite 60) festgelegt. XML Filter können diese Schnittstelle ebenfalls nutzen und dadurch ohne Öffnen des Dokuments auf deren Inhalte zugreifen.

Folgende Kandidaten wurden für die Komprimierung in Betracht gezogen:

- ZIP oder JAR, XML mit Binärdaten - ASCII-encoded mit speziellen Tags (z.B. base64), MIME Dateien, gezippte TAR, BONOBO libefs

Man entschied sich nach einer nüchternen Anforderungsanalyse für eine **ZIP Datei als Grundlage**. Die Eigenschaften hinsichtlich der erzielten Kompression, des Ladeverhaltens wurden dabei mit gut bewertet, die Speicherung an sich, die Eignung für ein XML Dokument und den ASCII Standard als befriedigend.

Ausschlaggebend war letztendlich die Bewertung zur *Speicherung der Dokumente*, da ZIP Dateien indiziert sind, und somit einen effizienten und sicheren Zugriff auf Unterdokumente ermöglichen. Unterdokumente können zudem getrennt und dabei unkomprimiert gespeichert werden und ermöglichen so den vollen Zugriff via ASCII, ohne das Dokument zuvor entpacken zu müssen. Dies ist von entscheidender Bedeutung, wenn es um den schnellen Zugriff auf z.B. Metadaten oder binäre Daten geht.

2.1 OOo - Textverabeitungsdokumente

Textdokumente besitzen eine feste Gliederungsstruktur und zeichnen sich durch viele eingebettete Dokumente aus (Grafiken, Tabellen...). Sie werden später in der Regel für den Druck oder eine Webpräsentation aufbereitet. Diese Dokumente besitzen die Dateiendung *sxw*.

Ich verwende im Folgenden ein OOW Dokument, welches jedoch schon mit einem Template erstellt wurde, welches erst später im Kapitel 2 auf Seite 18 erklärt wird. Folgender Text wird dabei zu Grunde gelegt (bitte beachten Sie diesen lizenzrechtlichen Hinweis zum verwendeten Template):

Hirnschmalz

Experiment

Wirkung von Koffein auf die psychokinetischen Fähigkeiten

von Wanda

Das Experiment setzt sich aus einer Versuchsperson, einer Dose mit Koffein versetzter Cola und einem Goldfischglas zusammen. Die Fähigkeit, einen Goldfisch durch die menschliche Willenskraft im Kreis herumschwimmen zu lassen, ergibt sich aus der bekannten Gleichung:

$$P = m * M / d$$

Gleichung 1

P ist dabei die Wahrscheinlichkeit, dass der Fisch in einem vorgegebenen Zeitinterwall eine Kreisbahn schwimmt, m ist der IQ des Fisches, M ist der IQ der Versuchsperson und d ist die Entfernung zwischen dem Fisch und der Versuchsperson.

Erste Elementebene

Im diesem Unterabschnitt wollen wir uns die Interna der *content.xml* Datei dieses OOW Dokuments genauer ansehen. Sie ist die Grundlage für alle späteren Konvertierungen. Die grobe Grundstruktur dieser Datei (sie wurde dazu mit *unzip* entpackt) zeigt Abbildung 1 auf der nächsten Seite.

Das *Wurzelelement* dieser Datei ist das `office:document-content` Element. Es enthält alle Namespaces, die in diesem Dokument Verwendung finden. Als Attribut finden wir `office:class`, welches uns mitteilt, welchen Dokumententyp wir hier vorfinden - also `text` für ein OOW Dokument, `spreadsheat` für eine Tabellenkalkulation u.s.w.

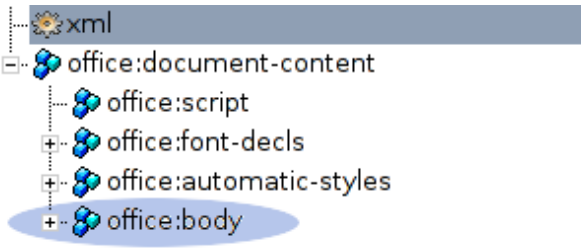

Abbildung 1: OO Writer Dokumentformat: Die erste Ebene von Elementen der Datei *content.xml.*

Die einzelnen Elemente unterhalb des Wurzelelements `document-content` kann man wie folgt aufschlüsseln:

office:script Dieses leere (!) Element besitzt als einziges Attribut den OOo Namensraum `xmlns:office="http://openoffice.org/2000/office".`

office:font-decls Hier wird das Mapping zwischen dem OOo Schriften Namensraum `xmlns:style="http://openoffice.org/2000/style"` und den XSL Formatierungsobjekten festgelegt.

office:automatic-styles Hier werden die Stile für Absatzlayouts, Ränder, Sprache u.v.m. des Dokuments als XSL Formatierungsobjekte abgespeichert.

office:body Der eigentliche Inhalt des Dokuments wird hier abgespeichert. Jeder einzelne Absatz wird mit zugehörigem Stil, Inhalt und eventuell untergeordneten XML Objekten (Links auf Bilder, samt Bildunterschriften...) erfasst.

Zweite Elementebene unterhalb von office:body

In Abbildung 2 auf der nächsten Seite sehen Sie die Struktur der Elemente unterhalb von des `office:body` Elements. Die Elemente des Typs `text:sequence-decl` können wir an dieser Stelle vernachlässigen. Uns interessieren nur die Elemente des Typs `text:p`, in welche die eigentlichen Inhalte des Textes und deren Stilinformationen gepackt sind.

Schauen wir uns zunächst das erste `text:p` Element an:

```
<text:p xmlns:text="http://openoffice.org/2000/text"
        text:style-name="P1"
        xmlns:text="http://openoffice.org/2000/text" >
    Hirnschmalz
</text:p>
```

Das erste Attribut `xmlns:text` kennzeichnet das Element dem `text`-Namensraum, so dass es einem OO Writer Dokument zugeordnet werden kann. Das zweite Attribut `text:style-name` kennzeichnet den Stil, mit dem es im Text dargestellt wird. Im Elementbaum von `office:automatic-styles` (s.o.) wurde dieser Stil festgelegt.

Schauen wir uns die zugehörige Stildefinition in `office:automatic-styles` einmal an:

```
<style:style
              xmlns:style="http://openoffice.org/2000/style"
              style:parent-style-name="Knowledge-object"
              xmlns:style="http://openoffice.org/2000/style"
              style:master-page-name="Standard"
              xmlns:style="http://openoffice.org/2000/style"
              style:family="paragraph"
              xmlns:style="http://openoffice.org/2000/style"
              style:name="P1"
              xmlns:style="http://openoffice.org/2000/style" >
   <style:properties
              xmlns:style="http://openoffice.org/2000/style"
              style:page-number="0"
```

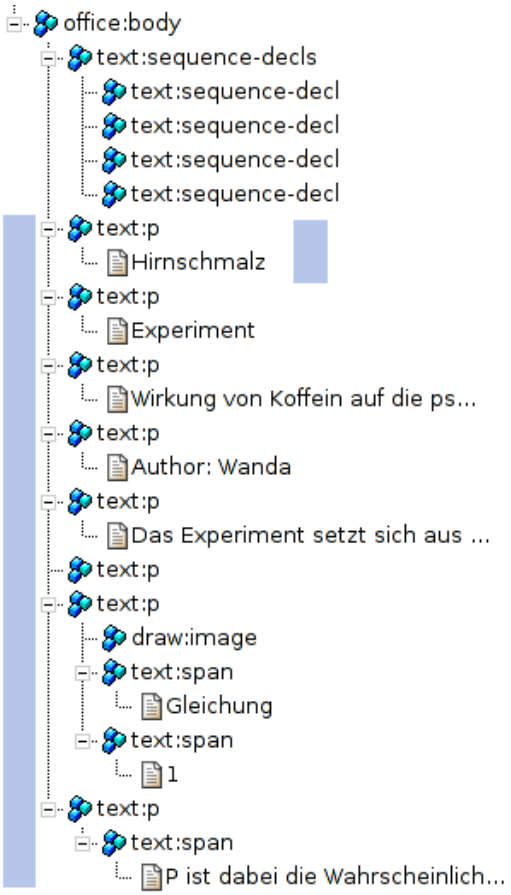

Abbildung 2: OO Writer Dokumentformat: Die erste Ebene von Elementen der Datei *content.xml*.

```
                    xmlns:style="http://openoffice.org/2000/style" />
    </style:style>
```

Die Zuordung des Absatzlayouts zum entsprechenden Eltern Stil geschieht an dieser Stelle. Dadurch wird das `style:name`-Attribut dem `style:parent-style-name` zugeordnet. In unserem Fall - doch dazu später mehr, einem *Knowledge-object*.

Leider sieht man an dieser Stelle, dass die XML Hierarchie, welche hier aufgespannt wird, relativ flach ist. Eine relative tiefe Dokumentenstruktur wird hier mit Hilfe von Attributen relativiert. Dies spielt später beim Exportieren eine große Rolle, da es immer schwieriger ist, mit Hilfe von XSLT (der XML Transformationssprache) flache Strukturen in geschachtelte zu transformieren als umgekehrt.

Dies war ein kurzer Abstecher ins OOW Dateiformat. In den weiter oben angegebenen Quellen über das OpenOffice Dateiformat ist ausführlich die XML Struktur dokumentiert. Für die später durchzuführenden Transformationen ist es ausreichend, die hier genannten Grundlagen zu verstehen.

Achtung: Den kompletten Quelltext des `office:body` finden Sie im Anhang (siehe 3 auf Seite 64).

2.2 Weitere OOo Dokumentformate

Alle anderen OOo Dokumentformate werden im Moment in dieser Arbeit nicht behandelt, zeigen aber prinzipiell den gleichen Aufbau, wie das OO Writer Dokument und bieten damit die selben Möglichkeiten zur Bearbeitung im Rahmen der XSLT Transformation.

OOo - Präsentationsdokumente

Diese Dokumentenart unterscheidet sich durch eine spezielle Eigenschaft von Textdokumenten: Sie besitzen eine **Ebenen- und eine Zeitkomponente**. Einen wesentlichen Teil einer gut aufbereiteten Präsentation macht es aus, dass Dokumentteile zeitlich versetzt vor oder hinter anderen erscheinen.

Tabellenkalkulationen

Tabellenkalkulationn werden im Rahmen dieser Version der Arbeit noch nicht behandelt. Zukünftige Versionen sollen diesen Bereich jedoch abdecken.

Mathematische Formeln MathML ist eine XML Beschreibungssprache für mathematische Inhalte. Open Office bietet ein eigenes Format

zur Erstellung komplexer mathematische Inhalte: StarML. Jede Formel, die in Open Office erstellt wird kann jederzeit mit einem Mausklick in unterschiedliche Ausgangsformate, darunter auch MathML exportiert werden. Im Zusammenhang mit E-Learning Systemen muss leider gesagt werden, dass derzeit das hier behandelte ELAT System, dass die Educational Modeling Language (siehe 3.2 auf Seite 24) als Basis nutzt, Inhalte zu speichern, MathML nicht nativ unterstützt. Das ELAT System kann leider MathML Formeln nicht nativ darstellen, was um so bedauerlicher ist, als MathML ein weltweit etablierter Standard ist. Formeln müssen deshalb leider umständlich in Bilder umgewandelt und als Abbildungen geführt werden!

Eine einzige Ausnahme ist mir bei meinen Recherchen untergekommen: Das Learning Material Markup Language Frameworks LMML des Institut für Informationssysteme und Softwaretechnik der Universität Passau.

Quellen zu MathML:

- W3C http://www.w3.org/TR/REC-MathML/
- MathML Conferences http://www.mathmlconference.org/
- LMML http://www.lmml.de/

Kapitel 2
Dokumenttypen in E-Learningsystemen

am Beispiel ELAT

In diesem Kapitel soll es um die Dokumenttypen gehen, welche in E-Learningsystemen, insbesondere im ELAT-System (Environment for Learning and Teaching) Verwendung finden. E-Learning ist informations- und kommunikationstechnologisch unterstütztes Lernen (Röder et al. [2002]).
Ich werde kurz auf die notwendigen Standards LOM und EML und anschließend konkret auf die Speicherung der einzelnen Dokumenteninstanzen eingehen. Leider muss ich den Leser dabei mit vielen Abkürzungen, Standards und Standardisierungsgremien konfrontieren, die sich aber - um es vorwegzunehmen - auf ein Gremium und einen Standard verdichten:

- den *LOM* und *EML* Standard zur Darstellung von Wissensinhalten
- das *LTSC* als Standardisierungsgremium für LOM und das *IMS* für EML

Eine detaillierte Beschreibung dieser Sachverhalte finden Sie in den folgenden Abschnitten.

1 ELAT
Environment for Learning and Teaching

oder: Wie erstellt man eine Kurseinheit?

ELAT ist eines von vielen bekannten E-Learningapplikationen. Es zeichnet sich durch einen extrem modularen Aufbau aus. ELAT ist eine

Client/Server Applikation, die komplett in Java geschrieben wurde. Der ELAT Client ist in zwei Versionen erhältlich. Er bietet sowohl dem Lernenden den Zugriff auf eine Datenbank mit vielfältigen Kursinhalten (Text, Grafik, Audio, Video), als auch den Lehrenden die Möglichkeit Kursinhalte anzulegen und in der Datenbank zu speichern (siehe Abbildung 3).

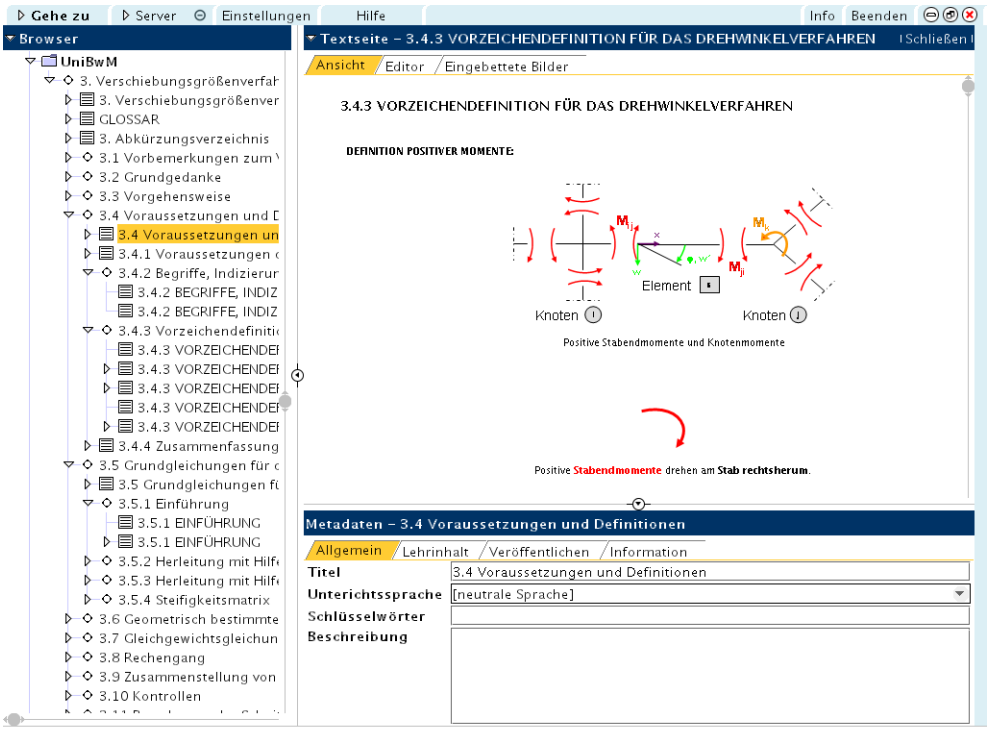

Abbildung 3: Autorenoberfläche des Client der E-Learningsoftware ELAT: Ansicht auf die Vorschau (Ansicht genannt) einer Kurseinheit.

Die Clientoberfläche kann nicht nur Kursinhalte visualisieren, sondern bietet alle Möglichkeiten einer modernen E-Learning Applikation, wie Messaging (Chat), Glossar, Zugang zu multimedialen Inhalten, Starten von externen Anwendungen u.v.m. Die Speicherung gesamten Kursdaten übernimmt das E-Learning System ELAT zentral auf einem Server, aber auch die lokale Vorhaltung der Daten ist möglich. Dies unterscheidet ELAT von den meisten anderen E-Learning Applikationen, die eine permanente Netzwerkverbindung voraussetzen.

Diese vielfältigen Möglichkeiten von ELAT oder E-Learningsystemen sind jedoch nicht Gegenstand dieser Arbeit, der Leser kann entsprechende Quellen zum Thema im Anhang (siehe Abschnitt 5 auf Seite 68) nutzen, um sich über die Thematik zu informieren.

Bei der Inspektion der Autorensoftware ergab sich folgendes Problem: Zwar war es möglich, Videos - sofern diese im Quicktime-Streaming Format vorlagen, Bilder und externe Anwendungen einzubinden, jedoch war es nicht möglich, eine mit einer Officeanwendung erstellten Text in das System zu integrieren. Zu diesem Zweck müssen in der

Autorenoberfläche die Kursdaten in einem speziellen XML Editor eingegeben, oder passende XML Dateien importiert werden (siehe Abbildung 4 auf der nächsten Seite). Die Eingabe wird dabei auf Korrektheit geprüft. Die Kurszusammenstellung ist damit jedoch ein Großteil Handarbeit.

Leider dürfte es den wenigsten Kurserstellern möglich sein, validen XML Text zu verfassen, auch wenn dies im Prinzip möglich wäre. Im Moment behilft man sich damit, einen zugekauften Konverter der Firma *Infinity Loop* `http://www.infinity-loop.de` einzusetzen, der auf der Befehlszeile Worddokumente in passende Kursdokumente (XML Dateien) übersetzen und aufteilen kann. Diese können dann mit Hilfe der Autorenoberfläche in einen Kurs importiert werden. Dabei werden auch Verweise auf Bilddateien korrekt integriert, welche anschließend dem Textbaustein hinzugefügt werden müssen. Es gibt bis jetzt keinen Exportfilter für Word oder eine andere Textverarbeitungssoftware, die nativ mit dem vorgegebenen XML Dialekt umgehen kann.

Die folgenden Abschnitte werden Sie mit den nötigen Grundlagen vertraut machen, um die Hintergründe zur Entstehung eines XML Textbausteins zu verstehen, wie er in ELAT Verwendung findet und gehen auf die Hintergründe der Standards ein, die dabei benutzt werden.

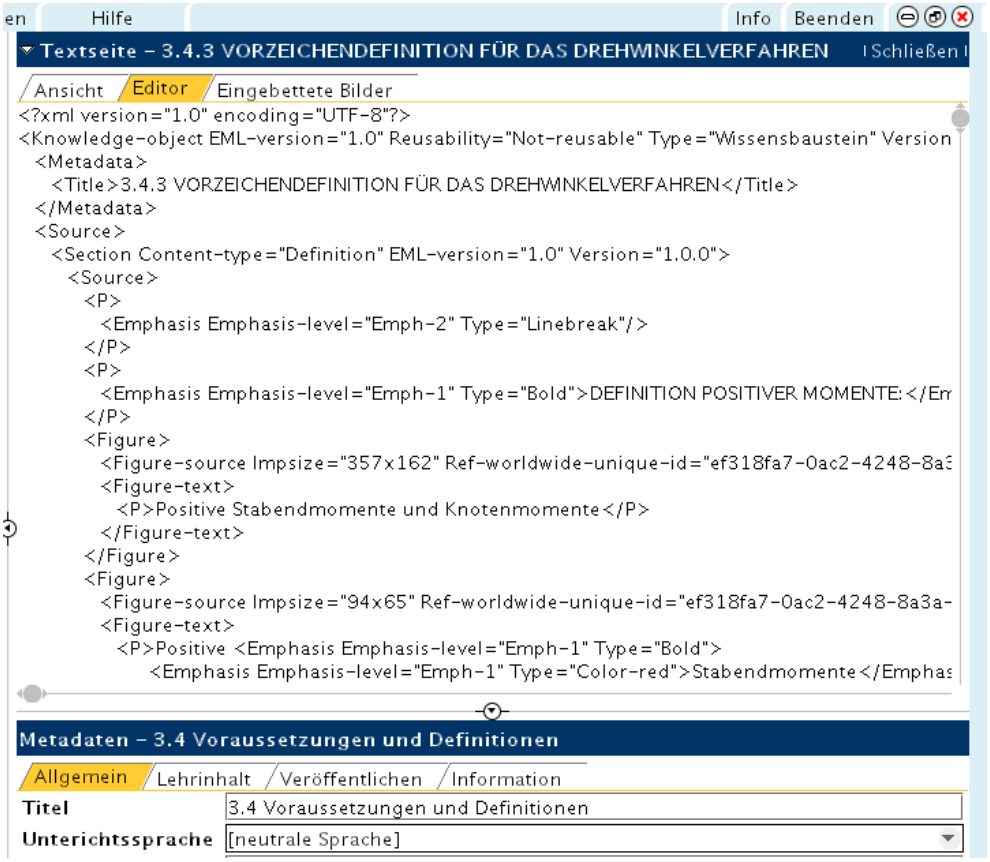

Abbildung 4: Autorenoberfläche des Client der E-Learningsoftware ELAT: Textuell basierte Kurse können nur in Form eines speziellen XML Dialektes importiert, oder mit Hilfe eines XML-Editors eingegeben werden.

2 Datenrepräsentation im E-Learning Bereich

oder: Was ist ein Learning Object?

E-Learning Systeme haben einen völlig anderen Ansatz zur Datenrepräsentation, als dies normale Textverarbeitungssysteme oder Präsentationsprogramme leisten könnten. Moderne Software zur Darstellung von Wissensinhalten (sog. *Courseware*) verfolgt das Paradigma des "*Learning Object Design*" (im folgenden kurz LOD genannt), manche sprechen auch von "*Educational modelling*". Dies war übrigens der Titel für ein Projekt, dass die dänische Regierung im Jahr 200XXXXX in Auftrag gab, um einen einheitlichen Standard für das elektronische Lernen zu erstellen.

. Frei übersetzt könnte man *Learning Objects* als Lernbausteine bezeichnen. Das *LOD Paradigma* könnte man in 4 Bereiche zusammenfassen, die in Tabelle 2 auf Seite 71 aufgeführt sind.

Tatsache ist, das hier an verschiedenen Fronten gekämpft wird. Auf der einen Seite möchten Anbieter proprietärer Software ihre Produkte möglichst so gestalten, dass Konkurrenten auf dem Feld von Lernsoft-

ware außen vor bleiben. Auf der anderen Seite kämpfen Organisationen, Universitäten und Bildungseinrichtungen auf der ganzen Welt darum, Wissensdatenbanken aufzubauen, die miteinander vernetzt sind und standardisierten Formaten genügen. Ich möchte an Stelle nicht genauer auf E-Learning oder entsprechende Systeme eingehen, sondern mich dezidiert mit den Teilen des Standards beschäftigen, der für das Dokumentenmapping zwischen normalen Office-Applikationen und Wissensbausteinen notwendig ist. Im Kapitel 5 auf Seite 58 finden Sie genügend Quellen, um sich einen Überblick zu verschaffen.

3 Relevante Schnittstellenstandards im E-Learning Bereich

3.1 LOM

Die Erfassung der Metadaten von Kursinhalten

Um der Wiederverwendbarkeit und dem modularen Prinzip gerecht zu werden hat das *Learning Technology Standards Commitee* (LTSC) das *Learning Object Model* (LOM) entwickelt. Es stell damit einen einheitlichen Standard bereit, durch dem Lernobjekte dargestellt und zu größeren Einheiten zusammen gefasst werden können. Die Metadaten werden dabei in XML Dateien abgespeichert. Um dem Aspekt der Wiederverwendung und Wiedererkennung gerecht zu werden, sorgt LOM dafür, dass besondere Eigenschaften von Lernobjekten, weil z.B. der Autor oder technische Details mit abgespeichert werden. Der LOM Standard ist detailliert auf den Seiten des IEEE `http://ltsc.ieee.org/wg12/` beschrieben. Da die genaue Spezifikation für dieses Projekt eine eher untergeordnete Rolle spielt, da in unserem Rahmen keine vollständigen Lerneinheiten erfasst werden sollen, sondern nur Teile davon, gehe ich hier nur kurz auf die wichtigsten Punkte ein.

LOM ist nur eine Spezifikation und daher von vornherein - wie eigentlich jeder XML Dialekt - auf Erweiterbarkeit ausgelegt. Der LOM Standard ist speziell auf elektronische Lernobjekte ausgerichtet. Besonders hervorzuheben ist die Möglichkeit, pädagogische Aspekte wie den Typ Lernobjekts (Übung, Simulation, Fragebogen, Diagramm, Tabelle, Text etc.), Grad der Interaktivität, didaktischen Kontext, Schwierigkeitsgrad, Durchschnittsalter der Zielgruppe und weiteres mehr zu erfassen.

LOM bildet dabei 9 Kategorien ab, die Sie in Tabelle 3 auf der nächsten Seite aufgelistet finden.

KATETEGORIE	BESCHREIBUNG
General Category	Grundlegende Informationen, die das Lernobjekt als Ganzes beschreiben
Lifecycle Category	Merkmale, die sowohl die Geschichte und den aktuellen Zustand des Lernobjekts als auch die beeinflussenden Lernobjekte beschreiben
Meta-Metadata Category	Informationen über die Metadaten-Instanz an sich
Technical Category	technische Voraussetzungen und Merkmale des Lernobjekts
Educational Category	pädagogische Merkmale und Bildungsmerkmale des Lernobjekts
Rights Category	Angaben zu Nutzungsbedingungen und Copyright-Fragen
Relation Category	Beziehungen zwischen dem Lernobjekt und anderen verwandten Lernobjekten
Annotation Category	Anmerkungen über den Bildungsnutzen des Lernobjekts und Informationen über die Entstehung der Kommentare (wann, von wem)
Classification Category	Einordnung des Lernobjekts in ein Klassifizierungssystem

Tabelle 3: Neun LOM Standard Kategorien nach IEEE

Die Erfassung und Bindung dieser LOM Metadaten an ein Dokument geschieht mit Hilfe der Autorenberfläche von ELAT und nicht (!) im Dokument selbst. So bleiben Lerneinheiten (Wissensbausteine) atomar und können in verschiedenen Kursen mit verschiedenem Schwierigkeitsgrad kombiniert werden. Es gibt auch externe Editoren für LOM Daten, wie z.B. den LOM Editor der TU Darmstadt: http://www.multibook.de/lom/de/.

Zusammenfassend lässt sich sagen, dass LOM in Zukunft sicher die Ausgangsbasis für viele zukünftige XML Dialekte für den E-Learning Bereich darstellen wird. Das gesamte Gebiet befindet sich in ständiger Weiterentwicklung und es ist empfehlenswert, die langfristigen Strömungen im Rahmen der o.g. Gremien (IEEE und LTSC) zu verfolgen.

3.2 EML

Die Erfasung von Lerninhalten in Studieneinheiten (”Study Units”)

Die *CEN/ISSS* (European Committee for Standardization / Information Society Standardization System) "Survey of Educational Modeling Languages (EMLs)" Version 1.0 definiert EML wie folgt:

> An EML is a semantic information model and binding, describing the content and process within a ’unit of learning’ from a pedagogical perspective in order to support reuse and interoperability. (CEN/ISSS, 2002)
>
> (*Übersetzt etwa:* Eine EML stellt ein semantisches Informationsmodell und eine semantische Bindung her, die den Inhalt und den Prozess innerhalb einer Studieneinheit beschreibt, wie er nötig ist, um von einer pädagogischen Perspektive aus die Wiederverwendung und Fähigkeit zur Interoperabilität zu unterstützen.)

Die Darstellung von Wissensinhalten und deren Prozesscharakter ist also die Aufgabe der *Educational Modelling Language* (EML). Sie wurde von der Open University of Netherlands (OUNL) entwickelt. EML hebt insbesondere die Didaktik und den Lernansatz hervor. Im Gegensatz zu LOM gibt es mehr Entities (im XML Sinne), die einen universellere Repräsentation von Wissensinhalten ermöglicht. Z.b. können Rollen, Interaktionen und Beziehungen zwischen Lernenden und Lehrenden modelliert werden.

Insbesondere im E-Learning Bereich wird ein *Subset* des EML verwendet, das vom IMS Global Learning Consortium `http://www.imsproject.org/` im Februar 2003 in der Version 1.0 veröffentlicht wurde.

Beide Standards - LOM und EML - weichen in einigen Punkten voneinander ab, diese Differenzen spielen in unserem Kontext aber keine grosse Rolle: In jedem Falle ist EML mächtiger und hat einen universelleren Ansatz als LOM. Beide Konsortien, das IMS (zuständig für EML) und das LTSC (zuständig für LOM) arbeiten eng zusammen.

EML eignet sich nicht nur für elektronische Lernobjekte, sondern für alle Arten von Wissenspräsentationen und Interaktion zwischen Lernenden und Lehrenden. Das Ziel ist ein zukünftiger Standard für die Repräsentation von Lerninhalten.

Ein von der *Open University of Netherlands* (OUNL) entwickelter Dialect der EML ist das sogenannte *Instructional Management System Learning Design* (IMS LD). IMS LD ist eine weitere Beschreibungssprache für Lehrinhalte.

Daneben gibt es eine andere IMS Spezifikation für die *Paketierung* von Lerninhalten: das sog. IMS Content Packaging, das jedoch keine didaktischen Konzepte implementiert.

Als Ergebnis kann man festhalten, dass eine Reihe von Standards existieren, die es nicht nur dem Laien schwer machen, eine zukunftssichere Entscheidung zu treffen. Besonders für den E-Learning Bereich ist aber EML mit Sicherheit eine gute Wahl, da sie einen Kompromiss hinsichtlich der Eignung und der Komplexität darstellt. Furcht, dass sich in den kommenden Jahren noch andere Standards herausbilden werden, muss man nicht haben, denn die XML Grundlage dieser Dialekte und die Tatsache, dass Sie über Jahre hinweg kontinuierlich international kommuniziert und entwickelt werden, macht kommende Konvertierungsprojekte gut planbar. Einen guten Überblick über diese Verfahren erhält man ebenfalls bei einer Suchanfrage auf `http://www.xml.org`. Dort sind alle nennenswerten Standards verzeichnet. Im Anhang habe ich einige Quellen für den Ausgang eigener Recherchen hinzugefügt.

3.3 EML in ELAT

Das E-Learning System ELAT verwendet den EML Standard zur Speicherung von unterschiedlichen didaktischen Konzepten im Rahmen einer sogenannten *Studieneinheit* (”Unit of Study”).

Es sind dies (IgDV FH-Darmstadt et al [2003]):

- Didaktische Methoden
- Aufgaben, andere Aktivitäten
- **Lernobjekte (Tests, Lehrmaterialien)**
- Konferenz Optionen (Chat, Email, etc.)
- Monitoring (Lernstatus)
- Tutor Funktionen, Such Optionen, Workflow Ooptionen und Anpassungen

Unser Fokus liegt auf den Lernobjekten und deren atomaren Bestandteilen, den sogenannten Wissensbausteinen. Dies sind die einzelnen atomaren Bestandteile einer jeden Studieneinheit (Texte, Videos, Audiodateien...).

ELAT verwendet wie schon gesagt einen Subset von EML Strukturen und unterscheidet *drei* verschiedene Repräsentationen in Form von *XML Schemas*, die jeweils einen Teilbereich der ELAT Systemspezifikation[9] abdecken.

1. Definition des kompletten ELAT LOM Modells (enthält 2 und 3)

[9] Interessenten und Softwareentwickler mögen sich bitte an das *Institut für grafische Datenverarbeitung Darmstadt der Fachhochschule Darmstadt* wenden, um genauere Informationen zur ELAT Software bzw. den dort verwendeten Standards einzuholen.

2. Definition des Zeitleistenobjektes
3. **Definition des Inhaltsmodells eines Knowledge Objekts**

Letzteres Schema beschreibt die Inhalte des kleinsten textuellen Bausteins, der als Kurselement im Autorenclient in einen Kurs eingebaut werden kann. Zur genauen Inhalten dieser Schemas im übernächsten Abschnitt.

4 Was sind Wissensbausteine (Knowledge Objects)?

ELAT unterscheidet zur Zeit folgende Medienobjekte, die in ein Kursprogramm importiert werden können (siehe 5):

- Videos
- Audiodateien
- Bilder
- Externe Anwendungen
- EML-Glossar
- **EML-Text**

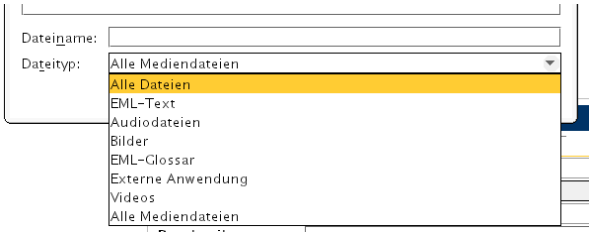

Abbildung 5: Medienimport in ein Zeitleistenobjekt von ELAT

Hier wird ersichtlich, dass das ELAT System die textuelle (XML-)Repräsentationen und Binädateien unterscheidet, wie es für XML Dateien üblich ist (siehe 2 auf Seite 10): Textuelle Wissenbaussteine werden in *EML-Text* abgelegt, Binäre Daten (Bilder, Videos, Audiodateien) im nativen Format. Die EML-Text verweisen dann mit Hilfe von X-Link auf die Bilddateien oder andere Binärformate.

In einem nächsten Schritt betrachten wir nun genauer den Aufbau eines *EML-Text* Wissensbausteins (Knowledge Objects genannt) und steigen damit tiefer in die XML Repräsentation ein des ELAT Systems ein.

4.1 EML-Text

Aufbau eines Wissensbausteins

EML-Text, wie er in ELAT Anwendung findet, genügt der Spezifikation 1.0, und kann entweder von der Open University of Netherlands http://eml.ou.nl/eml-ou-nl.htm oder auf XML.org http://www.xml.org/ heruntergeladen werden. Diese Dokument Typ Definition enthält alle Informationen, die nötig sind, um ein für ELAT erstelltes XML Dokument zu validieren.

In diesem Kontext interessieren in erster Linie die Teile von EML, welche in einem EML-Text Dokument Verwendung finden. Wie im Abschnitt 3.2 auf Seite 24 schon erwähnt wurde, bildet die EML eine viel komplexere Struktur. Auch die in ELAT zur Anwendung kommenden Zeitleistenobjekte.

Schauen wir uns die einfachst mögliche XML Form eines EML Textbausteins an. Zunächst in seiner Präsentation in der Ansicht des ELAT Autorenclients:

Hirnschmalz

Experiment

Ein (noch) ziemlich leeres Goldfischglas.

Hier dasselbe Dokument in der XML Repräsentation:

```
<?xml version="1.0" encoding="UTF-8"?>
<Knowledge-object EML-version="1.0" Reusability="Not-reusable"
                  Type="Wissensbaustein" Version="1.0.0"
                  Worldwide-unique-id="83a860be-229b-4459-8265-8f
  <Metadata>
     <Title> Hirnschmalz </Title>
     <Subtitle> Experiment </Subtitle>
  </Metadata>

  <Source>
     <Section Content-type="Task"
              EML-version="1.0"
              Version="1.0.0">
       <Source>
          <P>
           Ein (noch) ziemlich leeres Goldfischglas.
          </P>
        </Source>
    </Section>
  </Source>
</Knowledge-object>
```

Um nun den genaueren Aufbau eines `Knowledge-Object` zu verstehen, müssen wir uns die EML Spezifikation etwas genauer anschauen. Die EML Dokument Typ Spezifikation in der Version 1.0 beschreibt ein `Knowledge-Object` Element folgendermaßen:

```
<!ELEMENT Knowledge-object (Metadata, (Source | Internet-source)
    <!ATTLIST Knowledge-object %Component;>
```
[10]

Ein `Knowledge-Object` enthält damit in Reihenfolge `Metadaten` Elemente und anschließend entweder `Source` Elemente oder `Internet-source` Elemente.

Das Hauptelement `Source` ist in EML 1.0 wie folgt definiert (`Internet-source` wird im Rahmen eines ELAT `Knowledge-Object` nicht verwendet):

```
<!ELEMENT Source %Extra-p;>
   <!ENTITY % Extra-p
           "(P | Emphasis | List | Figure | Formula | Table
               | Lemma | Code-line | Literature | Audio
               | Video | Special   | View-property-value
               | View-property-group-values | Set-property-value
               | Set-property-group-values | Interactions
               | Internet-source | Section | Section-ref
               | Comment)*">
```

Als `Source` kommen nach EML alle denkbaren Inhalte, wie einfache Abschnitte (`P`), Listen (`List`), Tabellen (`Table`) u.s.w. zum Einsatz. Darin könne wiederum rekursiv weitere Sektionen enthalten sein. Somit ist die Abbildung eines größeren hierarchisch aufgebauten Dokuments gewährleistet.

Leider wird durch die Verwendung von Sektionen jedoch die komplexe Struktur eines hierarchischen Dokuments in z.B. Kapitel, Absätze, Unterabschnitte u.s.w. sehr verflacht, was der späteren Wiederherstellung des ursprünglichen Kontexts im Wege stehen dürfte. Eine Konvertierung ist daher nur *vom reicheren* Quelldokument einer Textverarbeitung *zur* EML Form hin sinnvoll.

Für ELAT - und dies dürfte für die meisten E-Learning Systeme zutreffen - wird anstelle einer DTD eine XML Schema Definition (XSD) verwendet[11].

Darin wird auch `Metadata` genauer spezifiziert. Dieses Element besteht im Grunde nur aus zwei möglichen Elementen: Titel (`Title`) und Untertitel (`Subtitle`), die am Anfang jedes Wissensbausteins stehen.

[10] Für das `Component` Entity siehe Anhang auf Seite 67.

[11] Die momentan verwendeten Schemata werden mit der Software Upcast zur Verfügung gestellt.

Nun besitzen wir alle nötigen Informationen, um mit Hilfe der OOo eigenen Werkzeuge ein Open Office Dokument in eine Reihe von EML-Text Dokumenten zu zerlegen. Wir werden dabei die XML Schema Language Transformation (XSLT) verwenden.

Teil II
Modell und Realisierung

Kapitel 3
Dokumentenmapping in OOo

Bevor die eigentliche Konvertierung stattfinden kann, will ich in diesem Kapitel zuerst die grundsätzlichen Möglichkeiten aufzeigen, wie in OpenOffice als Ausgangspunkt für das Dokumentenmapping zwischen OpenOffice und ELAT verwendet werden kann.

1 Anforderungsanalyse

Der Autor eines Kurses verfügt über zahlreiche Dokumente in verschiedenen Ausgangsformaten. Vom Textdokument, Präsentationsdokument über Bilder und Videosequenzen muss ein Sammelsurium von Ausgangsformaten bewältigt werden. Wir betrachten hier die Schnittmenge zwischen dem Autorenreservoir an Dokumenten und den Formaten, die ein Elearningsystem wie ELAT als Importfilter zur Verfügung stellen kann. Wie schon in Abbildung 5 auf Seite 26 dargestellt, beschränkt sich ELAT auf wichtige standardisierte Schnittstellenformate:

- Präsentationsdokumente (in Form von Quicktimedateien)
- Textdokumente (in Form von XML Dokumenten)
- Bilder (nativ eingebunden und verlinkt)
- Filmsequenzen (Quicktime)

Dem Anwender muss eine Möglichkeit haben, seine vielfältigen Ausgangsformate *einheitlich* zu erfassen, umzuwandeln und zu pflegen. Er muss dabei nicht nur ein E-Learning System bedienen lernen, sondern vielfältige Ausgangsformate, wie den Druck, Folienlayoute, Webseiten oder Präsentationen bewältigen. Für jede Aktivität in dieser Hinsicht ist meist ein Standard Büroprogramm - eine ”Office Suite” - die all diese Funktionen bietet vorhanden. Dem gerecht wird

$Microsoft\,Office^{TM}$oder $SUN's\,StarOffice^{TM}$ (bzw. *OpenOffice.org, genannt OOo* in der OpenSource Variante).

Ich gehen in der Folge davon aus, dass ein Autor sein Dokument im OOo Format vorliegen hat und zwar speziell als Text- bzw. Präsentationsdokument. Weiterhin setze ich voraus, dass das Dokument später als E-Learning Kurs Verwendung finden soll.

1.1 OO-Writer Dokument

Zwei didaktisch und softwaretechnisch verschiedene Ansätzen möchte ich angesichts der derzeitigen OpenOffice Version 1.1.4 näher betrachten:

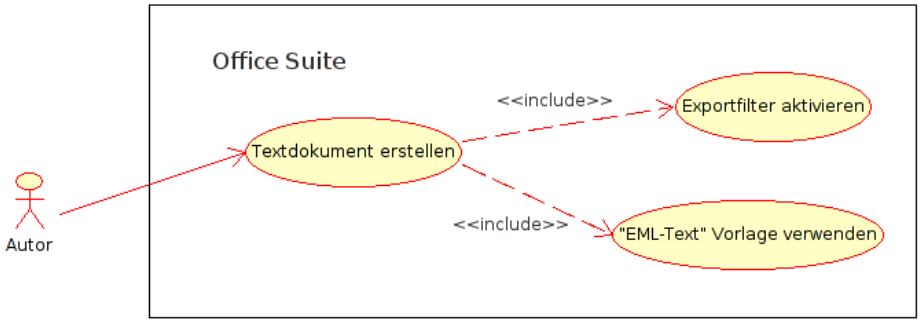

Abbildung 6: Anwendungsfall OO Textdokument für E-Learningsystem - nach OMG (Object Modeling Group Inc.) [1999].

Arbeiten mit einer speziellen Vorlage für EML-Text

Der Autor erstellt seine Dokumente mit einer speziellen Vorlage, die auf die Ausdrucksmöglichkeiten von EML Texten Rücksicht nimmt. Sie sollte dem Anwender z.B. nur solche Stile anbieten, die auch tatsächlich in EML umgesetzt werden können und andere vor Ihm verbergen.

Dies ist auf der Ebene der Stile (OOo Stylist, erreichbar in OOo durch drücken der Taste F11) möglich, die dem Anwender entsprechende Absatzvorlagen zur Verfügung stellt. Betrachtet man jedoch den Bereich Grafiken oder Tabellen, so sind leider die vielfältigen Möglichkeiten der Formatierung, welche OOW schwerlich umsetzbar, aber theoretisch mit hohem Aufwand möglich.

In jedem Fall erhält der Autor eine gute Vorlage für den Export in ein Format, welches vollständig im vorgesehenen XML Format des E-Learning Systems abgespeichert werden kann.

Arbeiten mit Standard OOo Dokumenten und einem Exportfilter für EML-Text

Möchte ein Anwender seine bisher in OOo erstellten Dokumente nach EML-Text konvertieren, wird der Anwender mit zwei Problemen konfrontiert.

So holt ihn zum einen nun leider die Nachlässigkeit der Textverarbeitungsprogramme ein, ihre Benutzer zur unstrukturierten Texterstellung erzogen zu haben. Es ist schlichtweg nicht gewohnt, mit Absatzvorlagen zu arbeiten.

OOo fördert mit der voreingestellten Ansicht eines Stylisten zwar das Ansinnen, Absatzformatierungen zu verwenden und sich damit mit einer strukturierten Art der Text***be***arbeitung auseinanderzusetzen, lässt aber dennoch vielfältige atomare Formatierungsmöglichkeiten zu, die in Ihrer Fülle nur schwer zu konvertieren sind.

Hier bleibt nur die Möglichkeit, darauf zu hoffen, dass ein Autor versteht, dass seine bisherigen Versuche, mit oft diversen Formatierungstricks unstrukturierte Dokumente zu erstellen, hier endgültig in einer Sackgasse münden. Kein noch so guter Filter kann alle beliebigen Formatierungen und deren Kombinationen exakt in einem anderen Dokumententyp abbilden.

Ein Kursautor wird sich beim Export in ein EML Dokument folgenden Problemen gegenüber sehen - je nach dem, wie strukturiert seine Texte erstellt wurden und wie intelligent ein Exportfilter OOo-Formatierungsstrukturen abbilden kann.

- Atomare Formatierungen verschwinden vollständig (Rahmen um Text, bestimmte Schriftarten, Farben...)
- Rein visuellen Strukturen, die durch atomare Textformatierungen entstanden sind, werden gar nicht abgebildet (dies betrifft z.B. Abschnitte, Kapitel...)

Für den unaufgeklärten Autor bricht an dieser Stelle erst einmal eine Welt zusammen, da er nicht verstehen kann, weshalb seine mühsam visuell formatierten Texte nun in einem völlig anderen ”Licht” erscheinen. Angesichts der Fülle der möglichen Formatierungs- und Absatzvorlagen und der Tatsache, dass jeder seine eigenen Vorlagendokumente beliebig erstellen kann, ist auf dieser Basis eine zukunftssichere Strategie zum Umwandeln in das EML-XML-Format aussichtslos.

Anmerkung Ich wünschte mir in diesem Zusammenhang, dass Textverarbeitungssysteme in Zukunft die Workflows zur Texterstellung deutlicher auf eine Absatzformatierung und hierarchische Gliederung hin ausrichten. Denn durch willkürliche Formatierungen werden Texte weder lesbarer, noch zukunftssicherer und ebensowenig verständlicher. Das Arbeiten mit Absatzvorlagen geht zudem einfacher und schneller vonstatten.

Um sich einen generellen Überblick über die Problematik zu verschaffen, empfehle ich den hervorragenden auch ins deutsche übersetzten Artikel von Cottrell [1999]. Zum Editieren von Markup für TEX, HTML oder die Erstellung von Docbook Formaten eignet sich auch hervorragend der Wysiwym[12] Editor Lyx, der leider seine Daten jedoch (noch) nicht in einem XML-Dialekt abspeichern kann.

Anderen Dokumentenstandards

Für das betriebliche Umfeld stellt sich in diesem Zusammenhang natürlich die Frage, ob nicht vorhandene XML Standard Dokumenttypen, wie DocBook http://docbook.org/ eine Alternative zu den Standardvorlagen der Textverarbeitungen darstellen. Für diese Option gibt es eine klare Antwort: in jedem Fall! DocBook ist eine international gepflegte und standardisierte Spezifikation und eignet sich hervorragend als Ausgangsbasis für jedwede Dokumententransformation in andere XML Dialekte, wie auch in EML.

Für OOo ist ein DocBook Filter vorhanden, der die Erstellung von XML DocBook Dokumenten als Ausgangsbasis theoretisch ermöglicht. Natives Arbeiten mit DocBook unter OOo ist jedoch nicht möglich, denn Ausgangsformat ist immer ein OOo Dokument und kein Docbook-Dokument. Der Docbookfilter wird in zukünftigen OOo Versionen ein nativer Bestandteil werden - siehe Zic [2003].

Docbook als mögliches Format sichert zudem den Zugang für einfachen Export/Import für in Printmedien wichtige Formate ”Framemaker” oder das immer noch häufig verwendete ”RTF”.

Dokumentenformate wie *(La)TEX* werden in Zukunft wahrscheinlich als reine Drucksatzsprachen weiterbestehen, im Hinblick auf eine Pflege von Dokumenten werden Sie jedoch mit Sicherheit durch XML verdrängt. Die Ausgabealgorithmen von modernen Textverarbeitungssystemen kommen schon nahe an Systeme, wie LATEX heran.

HTML kann in vielen Bereichen jetzt schon als Grundlage für einfache, gegliederte Texte dienen. Der beschränkte Umfang an Gliederungs- und Formatierungsfunktionen und die gezielte Optimierung für den Webeinsatz, schränken den Anwendungsbereich sehr ein. HTML ist in erster Linie der Darstellungsebene zugeordnet, weniger der Gliederungs- oder Inhaltsebene. Auch die XML Form von HTML ändert daran grundsätzlich nichts.

Weiterführende Quellen zum Thema DocBook und OOo:

- ➥ OOo DocBook Filter http://xml.openoffice.org/xmerge/docbook/

[12] *What you see is what you **mean*** im Gegensatz zum allgegenwärtigen *What you see is what you get* - Prinzip. Lyx www.lyx.org ist ein solcher Texteditor, mit dem - nebenbei erwähnt - auch diese Arbeit verfasst wurde. Ich habe einen Artikel zu LyX, TEX und PDF verfasst: siehe Pospischil [2004].

- Zic [2003]
- DocBook-Publishing http://www.stefan-rinke.de/articles/publish/index.html
- DocBook-Filter in OpenOffice.org http://de.openoffice.org/doc/sonstiges/DocBook16de.html

1.2 Einfügen des Dokuments in ELAT

Möglichkeiten, Workflows und Stolpersteine

Wenden wir uns dem Anwendungsfall zu, dass ein Autor die Autorenoberfläche des ELAT Client geöffnet hat und direkt ein ihm vorliegendes Officedokument als Teil eines Kurses importieren möchte.

Die folgenden Szenarien sind denkbar:

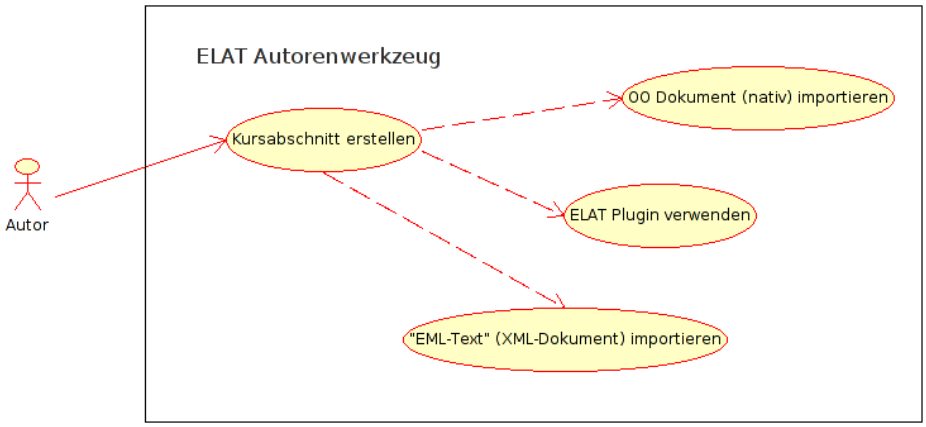

Abbildung 7: Anwendungsfall Dokumentenimport ELAT - nach OMG (Object Modeling Group Inc.) [1999].

1.2.1 Import des OOo-Dokuments

Diese Möglichkeit setzt das Vorhandensein eines Improtfilters in ELAT voraus. Hierbei können folgende Probleme auftreten:

- Das OOo Dokument ist nicht überhaupt nicht kompatibel
- Eine Konvertierung schlägt fehl, oder das Konvertierungsergebnis ist unbrauchbar
- Es treten Konvertierungsfehler auf, und das Dokument muss mühsam bei jedem Update der Kursinhalte mit Hilfe des vorhandenen XML-Editors bearbeitet werden

Ein Konvertierungsfehler muss nicht heißen, dass das Dokument nicht valide sein kann. Es kann auch bedeuten, dass es für den Anwender nicht brauchbar ist.

Bietet man also eine Importfunktion für den ELAT Autorenclient - sagen wir für OO-Textdokumente - an, dann muss sichergestellt sein, das die oben genannten Fehler abgefangen werden. Ich halte es für schwierig, bis unmöglich, angesichts der Fülle von Absatzvorlagen und Templates und der vielen Formatierungsoptionen in OOo, einem Anwender begreiflich zu machen, warum o.g. Fehler auftreten können. Frustration und Ärger sind vorprogrammiert. Zudem müssen zweierlei Softwarepakete gewartet werden: OOo und die ELAT Client Software. Ich empfehle nochmals an dieser Stelle, den reichhaltigeren Standard (OOo/EML) gegenüber der noch proprietären Software (ELAT) zu pflegen.

Besonders ”gefährlich” ist das Kopieren von Inhalten andere OOo Dokumente oder Importe von Word bzw. HTML in den OOW. Dabei werden die Styles in entsprechende OOW Styles konvertiert, ohne dass der Autor gewarnt wird. Dabei werden Formatierungen erstellt, die ebenfalls nicht durch den EML Exportfilter gemappt werden können und unser Anwender ist erstaunt, dass der Filter nicht wie erwartet funktioniert.

Das ELAT-Clientframework ist seit wenigen Wochen in Form eines Programmierhandbuchs in Anfängen dokumentiert. Der Dokumentationsprozess steckt jedoch noch in den Anfängen und ist für die nächste Monate vorgesehen. Softwaretechnisch geschieht der Zugriff auf den Server vom Elat-Client aus. D.h. dieser hat alle Möglichkeiten, komplette Kurse auf den Server hochzuladen. Es ist damit grundsätzlich möglich, einen Vollimport von OOo Dokumenten zu realisieren. Ein Ansatz in diesem Zusammenhang stellt sicher das Xmerge Framework von OOo zur Verfügung, welches alle nötigen Voraussetzungen dazu bietet (siehe 2.2 auf Seite 41), und sich auf Grund der Tatsache, dass sich daraus ein javabasierter Standalone Filter erstellen lässt hervorragend auch Clientseitig in ELAT integrieren ließe.

1.2.2 ELAT Plugin Modul

Auch über den Plugin Mechanismus des ELAT Systems könnte ein solcher Filter erstellt werden. Ich habe diesen Ansatz jedoch nicht weiter verfolgt.

Zu diesem Zweck kann ein neuer Lernobjekttyp entwickelt werden, um ihn als Modul in ELAT zu integrieren. Dies wäre jedoch auch wieder eine sehr spezielle und - nach Aussagen von einer Gruppe von Studenten, die in diesem Jahr den Plugin Mechanismus untersucht hat - eine zur Zeit sehr aufwändige Angelegenheit, der Vorgang ist zur Zeit jedoch relativ schlecht dokumentiert. In späteren ELAT Versionen ist der Plugin Mechanismus sicherlich einheitlicher aufgebaut, besser dokumentiert und einfacher zu implementieren.

1.2.3 Import des EML-Dokuments in ELAT

Eine *EML-Text* Importfunktion ist in ELAT schon vorhanden. Die Schnittstelle wird kontinuierlich vom ELAT Team gewartet. Welche Fehler können auftreten?

- ➥ Eine Konvertierung schlägt fehl und der Validator des Importfilters von ELAT verweigert die Annahme des Dokuments
- ➥ Es treten Konvertierungsfehler auf, und das Dokument muss mühsam bei jedem Update der Kursinhalte mit Hilfe des vorhandenen XML-Editors bearbeitet werden, weil es den Anforderungen des Autors nicht genügt

Wie wir sehen ist das Problem weggefallen, dass das OOo Dokument nicht konvertierbar ist (zumindest formal), da wir schon vorher sicher gestellt haben, dass unser EML Dokument ordentlich strukturiert und damit valide ist. im Zweifelsfall verhindert der Validator von ELAT den Import und antwortet mit einer entsprechenden Fehlermeldung.

Für die Wartung eines Importfilters liegt der Fokus des Entwicklers auf der Schnittstellenbeschreibung des von ELAT angebotenen Importfilters und der softwaretechnischen Umsetzung in OOo. Für OOo kann ein Template erstellt werden, das die genannten Fehler vermeiden hilft, in dem es dem Anbieter nur solche Formatierungsoptionen ermöglicht, die vom EML Standard unterstützt werden.

Es sei an dieser Stelle erwähnt, dass die Filterfunktionen von Open Office (aber auch von Word) noch in den Anfängen stecken und eigentlich nur von Anwendern sinnvoll benutzt und integriert werden können, die ein grundlegendes Verständnis von XML, der Konvertierung mit XSLT und den zugehörigen Dokumententypen besitzen - doch davon sollte man im entsprechenden Printmedienbereich ausgehen?[13] Der Normalanwender, der keine Zeit hat 2 Mannwochen in die Beschäftigung mit XML, XSL und XSLT in Zusammenarbeit mit einer Officesuite zu stecken, muss eine andere Möglichkeit geboten werden. Wie wir später noch sehen werden, lässt sich aber ein einfacher Filter mit relativ wenig Aufwand in OOo integrieren.

In jedem Fall sollte ein Exportfilter auch entsprechende Fehlermeldungen generieren, welche dem Anwender eine Hilfestellung geben, wenn er ungültige Dokumente produziert hat.

Für den XML Frischling gilt:

1. *Es muss dem Anwender klargemacht werden, dass ein 1 zu 1 Mapping aller in OOo vorhandenen Formatierungsfunktionen nicht möglich, wahrscheinlich sogar nie möglich sein wird, da wir es hier mit zwei völlig unterschiedlichen Konzepten zu tun haben:*

[13] Ausführlich und am Beispiel des DocBook Filters geht Zic [2003] auf die Probleme bei der Benutzung von OOW Filtern ein.

Formatierungsfunktionen von Textverarbeitungssystemen spielen gegenüber der kontextabhängigen Erfassung und Darstellung von Inhalten (getrennt vom Format) in Markupsprachen wie XML eine untergeordnete Rolle.

2. *Die Darstellung im Wysiwyg Editor OpenOffice Writer wird mit Sicherheit erheblich von der Darstellung im Ausgangsformat abweichen. Alle Formatierungsoptionen in OOW sollten im Sinne einer strukturellen Gliederung des Textes (Wysiwym) und nicht nach Aussehen vorgenommen werden.*

1.3 Conclusio

Aus dem voran gesagten ergibt sich leider - im Moment - nur eine sinnvolle Alternative für Autoren:

Sie müssen Ihre Dokumente auf *vor gelieferte Templates* für OOo abbilden, welche den EML-Standard unterstützen - was um so besser gelingt, je strukturierter die Texte im Vorfeld erstellt wurden. Beim Kopieren aus anderen Anwendungen ist dabei höchste Vorsicht geboten, da ”unsichtbare” Formatierungen in den Textpassagen eingebettet sein können.

Der direkte Import von Office Dokumenten jedweder Couleur direkt aus ELAT heraus wird auf Grund der hohen Kosten für die Entwicklung und Wartung wahrscheinlich Zukunftsmusik bleiben. Die OOo XML-Filter Schnittstelle ist und wird in Zukunft dagegen offen und gut dokumentiert sein, und stellt deshalb in meinen Augen die erste Wahl dar.

2 Konvertierung eines OO Dokuments in einen ELAT Wissensbaustein

Einen regelrechten Dschungel von Kombinationsmöglichkeiten erwartet den freudigen Softwareingenieur, wenn eine XML-Filter Lösung für das E-Learning System ELAT in Zusammenhang mit OOW Dokumenten implementieren möchte. Zudem werden die vorhandenen Filterfunktionen derzeit dem *Alpha* Stadium zugeordnet und sind damit noch stark in der Entwicklung begriffen.

Am Anfang meiner Überlegungen stand die Idee, einen Importfilter direkt in ELAT einzubauen. Nun, dieser erste Gedanke entstammte der Sicht auf das eigentliche Problem: eine fehlende Importfunktion für Officedokumente im ELAT-Autorenclient. Erst auf den zweiten - anwendungsorientierten - Blick hin erkennt man, dass ein EML konformes Dokument als Ausgangsbasis die Lösung darstellt.

Zur Erinnerung: EML-Text ist das XML Dokumentenformat, das dem ELAT-Autorenclient vorliegen muss, damit Texteinheiten wie z.B. Ka-

pitel, einem aktuellen Kursdokument beigefügt werden kann. Das E-Learning System dient nur als zusätzlicher Behälter für den importierten XML Quelltext. Während des Importvorgangs wird das Dokument auch auf syntaktische Korrektheit geprüft.

Die Entwicklung eines Office Importfilters - fest in das ELAT System integriert - ist meiner Ansicht nach nur sinnvoll, wenn ein intensiver Kontakt zum ELAT-Entwicklerteam gepflegt wird. Zudem steht die Lizenz, unter welcher die ELAT-Software vertrieben wird noch nicht fest und es dürfte daher problematisch sein, OpenSource direkt ins System einzubinden[14]. Es ist auch fraglich, ob dieser Ansatz zukunftsweisend ist, da dieser Softwareteil ELAT-seitig gepflegt werden muss und Seiteneffekte nicht ausgeschlossen werden können.

Als nächstes stellt sich also die Frage, welches Dokumentenausgangsformat verwendet werden sollte, da in Zukunft beliebige Software dieses Format lesen können muss. Wie ich in diesem Kapitel ausführlich dargestellt habe, spricht vieles gegen ein proprietäre Format. Leider muss man auch erwähnen, dass entsprechende proprietäre Konverter von z.B. Word in das EML-Text Format bereits existieren, gute Dienste verrichten, freie Software jedoch meines Wissens nach (noch) nicht existiert. Dies war auch die Motivation für diese Arbeit.

Tabelle 4 auf Seite 72 zeigt zusammenfassend, welche Varianten zur Konvertierung von Office Dokumenten in ELAT Textdokumente zur Diskussion stehen.

Für die Konvertierung in einen sogenannten **Wissensbaustein**, den das E-Learning-System ELAT als Grundeinheit verwendet, muss ein OOe Dokument mit Hilfe eines XSL Stylesheet umgewandelt werden. Glücklicherweise handelt es sich bei beiden Formaten um XML-Dateien.

OOo kann mit Unterstützung von Java installiert werden. Wird das Java Runtime Environment 1.4 `http://java.sun.com/j2se/index.jsp` und größer verwendet, dann sind alle nötigen Werkzeuge bereits installiert. Die Umwandlung von XML Dokumenten basiert im OOo auf der Nutzung des XSLT Prozessors Xalan `http://xml.apache.org/xalan-j/index.html`, welcher in das JRE ab Version 1.4.1 bereits integriert ist. Ältere Javaversionen (1.3) müssen mit Xalan nachgerüstet werden. Manche Autoren (s. Zic [2003]) empfehlen die zusätzliche Installation des XSLT Prozessors *Saxon* (Michael Kay [2004]) in der Version 6.5.x. Versionen größer 7 sollen jedoch demnach in Zusammenarbeit mit OpenOffice 1.1.x und z.B. dem DocBook Filter nicht funktionieren!

Es gibt grundsätzlich 2 verschiedene Ansätze, um OpenOffice als Editor für unterschiedliche Ausgangsformate zu verwenden:

1. Ausgangsbasis ist ein *OO-Template*, dass dem gewünschten Ausgangsformat entspricht. Das Template sorgt dafür, dass dem

[14] OpenSource Code muss klar zum übrigen Softwarecode abgrenzbar sein und als eigenständiger, d.h. vom proprietären Anteil unabhängig implementiert werden.

Benutzer entsprechende Formatierungsstile angeboten werden. Das Dokument wird mit Hilfe eines speziell erstellten Import- bzw. Exportfilters im gewünschten Format abgespeichert.

2. Ausgangsbasis ist das normale OpenOffice Writer Format. Mit Hilfe eines *Exportfilters*, der die *Xmerge* 2.2 auf der nächsten Seite-Engine von OOo nutzt, wird das Dokument in das gewünschte Format exportiert.

Ein Filter sollte die Möglichkeit bieten, größere Mengen von Dokumenten nacheinander und automatisiert umwandeln zu können. Dafür ist es nötig, dass der Filter auch unabhängig von Openoffice verwendet werden kann, d.h. als *Standalone* Lösung im Batchverfahren eingesetzt werden kann.

Die folgenden Abschnitte zeigen vom Prinzip her aus, wie diese beiden Ansätze in OOo umgesetzt werden können:

2.1 Standardfilter mit OO Dokumenttemplate

Um einen Standardfilter in OOo zu intergrieren, ist relativ wenig Aufwand nötig. Es müssen dazu zwei[15] XSL Stylesheets erstellt werden. Einer für den Import, einer für den Export ins gewünschte (XML) Format. Damit die Menge zu konvertierenden Absatzstile möglichst gering bleibt, empfiehlt sich das Erstellen eines geeigneten *OOo Dokument Templates*, welches der Autor als Vorlage verwenden kann.

OOo bietet dazu einen Filterassistenten, für den folgende Dateien bzw. Informationen bereit gehalten werden müssen - hier am Beispiel des EML Filters im OO Writer:

[15] Oder nur einer, wenn z.B. nur die ”Speichern unter” Funktionalität erwünscht ist.

Beschreibung	
Filtername	Beliebiger String, hier "*EML 1.0*"
Zugehörige OOo Anwendung	Dropdown (Writer, Calc, Impress, Draw), hier "*Writer*"
Name des Dateityps	Beliebig, wird im Öffnen/Speichern unter Dialog angezeigt, hier "*Education Modeling (eml)*"
Dateiendung	Beliebige Dateiendung für das zu exportierende bzw. importierende Dokument (z.B. xml, eml, txt ...), hier "*xml*"
Kommentar	Beliebiger Text, hier "*Filter für die Konvertierung von OpenOffice Writer Dateien (sxw) in einen Elat Textbaustein (xml). Bilder müssen als "verlinkte" externe Dateien im aktuellen Dokumentpfad vorliegen - integrierte Grafiken werden nicht extrahiert.*"
Transformation	
Doctype	XML Doctype, hier "*-//OUNL//DTD EML/XML binding 1.0/1.0//EN*"
DTD	XML Dokument Type Definition zur Validierung, hier "*/PfadZu/eml1.0.dtd*"
Export Schema	XSLT Schema zum Export (Speichern unter), hier "*/PfadZu/sofftoeml.xsl*"
Import Schema	XSLT Schema zum Import (Datei öffnen), hier "*/PfadZu/emltosoff.xsl*"
Dokumentvorlage	Open Office Writer Dokumentvorlage, die zur Erstellung dieses Dokumenttyps verwendet werden soll (dringend empfohlen), hier "*/PfadZu/eml_emplate.stw*"

Tabelle 5: OOW Filterdialog: nötige Angaben für den Transformationsfilter (siehe 2 auf Seite 63 im Anhang). **Fett** hervorgehobene Angaben sind zur Erstellung unbedingt nötig.

Mit Hilfe des Filtedialogs kann man nun auf Wunsch anhand der gemachten Angaben ein *Jar* File erstellen, welches in etwa dem entspricht, dass wir später zur Registrierung eines Plugins benötigen. Tatsächlich basiert auch hier die Registrierung des Filters auf einem Framework, dass OOo zur Verfügung stellt und die UNO Schnittstelle benutzt.

2.2 Xmerge basierter Exportfilter

Ursprünglich für die Konvertierung von Dokumenten für kleine Geräte (PocketPCs und Co.) entwickelt, stellt Xmerge `http://xml.openoffice.org/xmerge/` ein javabasiertes universelles Konvertierungsframework zum Bau von Import- und Exportfiltern zur Verfügung. Im Gegensatz zu einem Standardfilter, der nur auf der XSLT Konvertierung mit Hilfe eines Stylesheets beruht, bietet das Xmerge Frame-

work die Möglichkeit zur vielfältigen Manipulation. Unter anderem ist es möglich, Formatierungen für die spätere Verwendung im Exportformat zu speichern, so dass sie für die später im OOo Dokument wieder zur Verfügung stehen. Ein Anwendungsfall sind z.B. Dokumente für den Pocket PC, welche man nach einer Kopie auf dieses Gerät und zurück auf den Rechner natürlich im ursprünglichen Format weiter bearbeiten will. Dies ist z.B. bei älteren Worddokumenten nicht möglich, da die Formatierungsoptionen bei der Kopie verloren gehen! In weiteren Verlauf werde ich diese speziellen Möglichkeiten von Xmerge jedoch nicht verwenden, sondern mich allein auf den Konvertierungsmechanismus von XML-OOW nach XML-EML konzentrieren.

2.2.1 Ein Anwender Workflow:

Mit einem einheitlichen Officeformat arbeiten und dann das Dokument *exportieren...*

Stellen Sie sich den Anwender vor, der nicht so sehr mit den verschiedenen Dokumentenformaten vertraut ist. Er wird vorrangig mit zwei Applikationen konfrontiert: Seine Dokumente schreibt, korrigiert und speichert er mit Hilfe einer Office Suite. Publizieren oder weiterreichen möchte er diese Dokumente, in dem er Sie in das entsprechende Format exportiert. Meine eigenen Erfahrungen im Umgang mit Kunden zeigen, dass Anwender den Anwendungsfall ”Dokument exportieren” als grundlegend anderen Vorgang erfassen, als den Anwendungsfall ”Dokument speichern unter”. Dazu kommt, dass im Workflow das Dokument nach letzterem Vorgehen im OOo externen Format verbleibt und wichtige Formatierungsfunktionen verloren gehen können. Der Anwender müsste nach erfolgtem ”speichern unter” also nochmals die Datei im OOo Format abspeichern, ich halte dies für wenig produktiv.

Aus diesem Grunde wäre es angebracht, einen Exportfilter zu verwenden, der als Plugin in OOo eingebunden wird. Es gibt bereits einen Open Source Filter für LaTeX, Xhtml und MathML names Writer2latex http://www.hj-gym.dk/%7Ehj/writer2latex/ von Henrik Just - Just [2004]. Der Autor hat leider wesentliche Teile des Xmerge-Frameworks derart verändert und angepasst, dass eine Adaption an vielen Punkten scheiterte. Das Xmerge-Xslt Framework ist leider ebenfalls zu großen Teilen nicht nutzbar. Hendrik Just’s Filter sind Handarbeit, und basieren nicht auf einer Xslt Transformation, wie Sie hier gewünscht ist.

Von kosmetischer Auswirkung - aber auch von Nachteil für das Handling - ist es, den Xslt Filter des XMerge Frameworks dazu zu bewegen, als registriertes *Export Plugin* zu fungieren. Der Anwender Workflow *”Dokument exportieren als”* wird deshalb zur Zeit für einen XSLT basierenden OOo Filter nicht unterstützt, obwohl er problemlos in den Quellcode eingebaut werden kann. Die Ausführung des Test Treibers auf der Befehlszeile zeigt zwar die einwandfreie Funktion des Plugins, ebenso wird des Exportfilter sichtlich eingebunden, bleibt jedoch anschließend die die Konvertierung schuldig. Bis zum Abschluss die-

ser Arbeit und auch nach tagelangen erfolglosen Versuchen konnte ich leider der Ursache nicht auf den Schliche kommen. Offensichtlich werden nur *nicht*-Xslt basierte *PluginFactoryImpl*-Classes durch den Plugin Mechanismus von Xmerge eingebunden. Es ist mir kein Beispiel bekannt (auch nicht beim DocBook-Filter), wo ein Xslt-Filter als *OOo Export-Dialog* installiert werden konnte.

2.2.2 Xmerge Framework

Xmerge ist eine Zusammenfassung verschiedener Software Pattern zur Realisierung folgender Funktionen:

- Editieren von RTF Dokumenten auf Palm oder PocketPC's und ähnlichen kleinen mobilen Geräten.
- Erhaltung der dabei gemachten textuellen Veränderungen bei der Rückkonvertierung in die reichhaltigere Office Dokumentformate.
- **Nutzung des OOo XML Formats.**
- **Möglichkeit zur Konvertierung, Mischung und Erkennung von Veränderungen.**
- **Die Möglichkeit, die oben genannten Funktionen als Plugin für OOo zu Verfügung zu stellen.**
- Die Möglichkeiten eines Konverters können zur Laufzeit festgestellt werden.

Es stellt damit ein Modell für die Konvertierung von OOo Dokumenten (zunächst für die Textverarbeitung) in beliebige Ausgangsformate dar.

Für die Implementierung eines Exportfilters sind die hervorgehobenen Aspekte von Bedeutung. Bisher gibt es noch nicht viele Filter für OOo, die auf diesem Framework basieren. Die Dokumentation ist im Xmerge Development SDK `http://xml.openoffice.org/xmerge/docs/XMerge_sdk.pdf`[16] gut dokumentiert. Dasselbe gilt für den Xmerge Quellcode `http://xml.openoffice.org/xmerge/xmerge_src.zip`[17], der in Javadoc (siehe OpenOffice.org [2002]) ausführlich beschrieben ist.

Um all diese Funktionen zu realisieren, sind softwaretechnisch 3 Patternframeworks vorhanden, die u.a. einen prototypischen Filter realisieren, welchen der Programmierer entsprechend implementieren muss. Es sind dies:

1. der XmlFilterAdapter `http://xml.openoffice.org/xmerge/downloads/index.html#XmlFilterAdapter`

[16] Cameron et al. [2002]

[17] Via CVS ist der aktuelle Sourcecode ebenfalls zu bekommen: *pserver:anoncvs@anoncvs.services.openoffice.org:/cvs*

2. die *XMergeBridge*
3. verschieden auf Xmerge basierende Plugins `http://xml.openoffice.org/xmerge/index.html#plug-ins`

Alle OOo Filter machen dabei ergiebig Gebrauch vom *Factory Pattern* (Gamma et al. [1994]). Dadurch kann zur Laufzeit der Dokumenttyp bestimmt, geprüft[18] und eine spezielle Konversion angestoßen werden. Die nötigen Informationen über den Konverter werden in einer XML-Datei (*converter.xml*) beschrieben, die zusammen mit anderen Daten (z.B. Stylesheets) in einer *Jar-Datei* gespeichert, und anschließend als Plugin in OOo registriert werden.

Theorie und Praxis

Leider ist es - wie schon erwähnt - im Moment nicht möglich, das XSLT-Plugin von OpenOffice dazu zu bewegen, als Export Workflow zu arbeiten (zur Erinnerung = *"Datei->Exportieren"*). Dieser Workflow wird zur Zeit nicht unterstützt. Der Anwender hat darüber hinaus jedoch zwei Möglichkeiten eine Exportfunktionen mit Hilfe eines vorhanden Stylesheets zu nutzen:

1. Im Dialog "speichern unter"
2. Auf der Befehlszeile mit Hilfe des Xmerge Filters (*"TestDriver"* genannt).

Ad 1: Problematisch ist, dass nach dem speichern in das XML Format nicht alle Formatierungen erhalten bleiben. Der Anwender sollte also umgehend das Dokument wieder im OOW Format abspeichern und dann erst weiterarbeiten. Er würde sonst beim Öffnen des XML Dokumentes eine böse Überraschung erleben. Hat man sich an diese etwas umständliche Vorgehensweise gewöhnt und benutzt man diesen Arbeitsschritt selten, d.h. nur am Ende eines längeren Editiervorganges, dann kann man den OOWriter tatsächlich effektiv als Texteditor für beliebige XML Dokumente einsetzen.

Ad 2: Der XSLT Filter wird im Xmerge Framework integriert bzw. registriert. Die Javaklasse, welche den Filter anstößt, heißt "*Driver.class*" und befindet sich an folgender Stelle der *xmerge.jar* Datei:

`org.openoffice.xmerge.test.Driver`

Um den Konverter nutzen zu können, muss der Java CLASSPATH zudem die Datei *xmerge.jar* beinhalten. Dann und nur dann (!) kann auf der Befehlszeile mit dem Befehl

[18] Im Moment kann leider noch nicht gegen eine DTD geprüft werden (Cameron et al. [2002])!

```
java org.openoffice.xmerge.test.Driver
      -from application/sxw
      -to   application/eml
      Dateiname_der_zu_konvertierenden_OOW_Datei.sxw
```

der OOW Filter angestoßen werden. Die Ausgabe erfolgt in eine Datei namens *Dateiname_der_zu_konvertierenden_OOW_Datei.sxw.xml*.

Damit steht auch einer Batch-Konvertierung für mehrere Dokumente nichts im Wege.

Kapitel 4
Exportfilter für den OOWriter

Vorweg sei gesagt, dass ein Exportfilter nur prototypisch realisiert werden konnte.

1 Xslt Stylesheet zum Export von EML

XSLT ist eine deklarative und funktionale Programmiersprache. Sie eignet sich daher nicht so sehr für die klassische Programmierung von Anwendungen, die einen strukturierten Programmablauf voraussetzen. XSLT dient der Erstellung eines Regelwerks, dass die Ausgabe als Funktion der Eingabe beschreibt.

Die Programmlogik befindet sich somit nicht in einem Java Programm, wie im vorigen Abschnitt, sondern im XSLT Regelwerk, dass in einer XML Datei, dem XSLT Stylesheet abgespeichert wird. Darum können auch keine darüber hinausgehenden Manipulationen durch z.B. die Verwendung von Variablen erzeugt werden. Eine definierte Eingabe erzeugt eine - und nur eine - wohldefinierte Ausgabe. Dies ist ein Vorteil gegenüber einer Programmiersprache, wie Java: XSLT kennt keine Seiteneffekt, d.h. die Ausgabe funktioniert, oder sie funktioniert nicht, dazwischen gibt es nichts! Sie ist dadurch sicher und extrem robust.

Wir müssen in unserem Fall zwei Varianten einer XSLT Logik unterscheiden: eine Programmlogik zur Transformation eines EML Dokuments in ein OpenOffice Dokument (Importfilter) und eine Logik für den umgekehrten Weg (Exportfilter).

1.1 Besserer XSLT Code...

Ein Wort sei mir an dieser Stelle zum Programmierstil von XSLT Skripten erlaubt, denn bei der Analyse bereits vorhandener proprietärer Konverter von z.B. Word nach EML, musste ich feststellen, dass diese

Filter leider sehr flach programmiert waren. Gerade bei solch komplexen Filtern, wie Sie für Office Dokumente nötig werden, sind ordentliche Konzepte zur Erstellung von XSL Stylesheets unerlässlich.

Man kann XSLT Stylesheets programmieren, so dass sie wie ein Lochkartenstreifen funktionieren: endlos lang, unübersichtlich und flach. Solche Stylesheets sind leider weit verbreitet. Der Nachteil: Sie sind weder besonders schnell noch sind gut wartbar, d.h. erweiterbar. Die Entwicklung eines ordentlichen Stylesheets sollte aber die Struktur des Dokumentes modular wiedergeben können. Ich habe folgende Methoden angewendet um die in diesem Kapitel vorgestellten und von mir entwickelten[19] XSLT Stylesheets umzusetzen.

Die Grundstruktur jeden modular aufgebauten XSLT Stils ist das `xsl:template`. Templates kann man auf verschiedene Weise darauf hin präparieren, dass Sie nur in einem speziellen Kontext aufgerufen werden:

- Call by *name,* Call by *context,* Call by *mode,* Globale Variablen

1.1.1 Call by name

Das hier vorgestellte Pattern benutzt die Eigenschaft des `xsl:param` Elements und des `name` Attributs eines Templates. Auf diese Weise können einzelne Templates gezielt gestartet werden.

Definition

```
<xsl:template name="name-of-template">
   <xsl:param name="parameter_1"/>
      <!-- Implementation hier -->
</xsl:template>
```

Aufrufmuster

```
<xsl:call-template name="name-of-template">
   <xsl:with-param name="parameter_1" select="'node'"/>
</xsl:call-template>
```

1.1.2 Call by mode

Der XSLT Prozessor ist eine Zustandsmaschine. Den Zustand des XSLT Prozessors kann man mit Hilfe des `mode` Befehls steuern. Ein Template, dass mit dem `mode` Attribut erstellt wurde, wird nur dann aufgerufen, wenn es im selben Kontext definiert wurde.

[19] Alle in dieser Arbeit vorgestellten Stylesheets sind unter die GLPL gestellt worden, siehe Anhang 7 auf Seite 70

Definition

```
<xsl:template match="xml-node-name" mode="your-special-mode">
        <!-- Implementation hier -->
</xsl:template>
```

Aufrufmuster

```
<xsl:apply-templates select="context-node/xml-node-name"
                                        mode="your-special-mode">
    <xsl:value-of select="'node'"/>
</xsl:apply-templates
```

1.1.3 Call by context

Dies ist der Normalfall, wie templates verwendet werden. Der Nachteil ist, dass immer wenn das `select` Kriterium zutrifft, das Template ausgelöst wird.

Definition

```
<xsl:template match="xml-node-name">
    <xsl:param name="parameter1"/>
        <!-- Implementation hier -->
</xsl:template>
```

Aufrufmuster

```
<xsl:apply-templates select="context-node/xml-node-name">
    <xsl:with-param name="parameter1" select="'node'"/>
</xsl:apply-templates>
```

1.1.4 Globale Variablen

Definition

```
<xsl:variable name="global-variable">
   <!-- Implementation der Variable hier, auch xsl-Anweisungen --
</xsl:variable>
```

Aufrufmuster

```
<xsl:value-of select="$global-variable" />
```

Diese vier Arten XSLT Code zu modularisieren stellen eine exzellente Möglichkeit dar, die unterschiedlichen Aspekte von Dokumentteilen herauszuarbeiten. Ich habe in meiner Arbeit diese Vorgehensweise konsequent eingesetzt, um die Möglichkeit der Erweiterbarkeit und die Wartungsfreundlichkeit zu erhöhen.

Im nächsten Abschnitt möchte ich die OOo spezifischen Aspekte bei der Erstellung eines XSLT Stylesheets diskutieren.

1.2 Erstellen des XSLT Filters

Zur Erstellung eines Knowledge Objects mit OOo, aber auch zum Import eines solchen ist es notwendig, ein eigenes OO Writer Template als Vorlagendatei mit der Endung *stw* zu erstellen bzw. zu verwenden. Dadurch ist sichergestellt, dass der Benutzer tatsächlich mit den Absatzvorlagen arbeitet, die für ein *Knowledge-Object* sinnvoll sind.

> Dies ist wichtig, denn die XSLT Stylesheets, welche wir in OOW als Filter integrieren werden sind in hohem Masse abhängig von den Namen der Absatzlayouts, die mit Hilfe des Stylisten für die Vorlage eines *Knowledge-Objects* erstellt wurden. Konkret heißt dass: die Umsetzung der einzelnen Dokumententeile geschieht an Hand der Namen der Absatz- bzw. Zeichenlayouts. Im Anhang sind die von mir umgesetzten Stile dokumentiert (siehe 4.1). Solange die Namen der Absatzvorlagen im Stylist von OpenOffice nicht geändert werden, funktioniert der Import/Export einwandfrei. Haben Sie jedoch die Absicht diese Namen zu ändern, dann müssen Sie die XSLT Stylesheets ebenfalls anpassen.

Wie wir auf auf Seite 40 gesehen haben, werden die Absatzstile in Open Office in einer separaten XML Datei gespeichert. Arbeiten wir mit einer benutzerdefinierten Vorlage für *Knowledge-object*s (für den OO Writer mit der Endung **.stw*) stehen diese für die Dateivorlage und damit auch für unseren XSLT Filter zur Verfügung.

Bei Bedarf kann eine Definition natürlich auch innerhalb des Filters geschehen! Da, wie wir noch sehen werden, alle Dateien jedoch zu einem Filterpaket geschnürt werden, ist dies nicht nötig.

Die Vorlagendatei wurde mit OpenOffice erstellt. Dadurch ist der Code dieser Datei frei verfügbar, denn er ist, wie die von mir erstellten zugehörigen XSLT Stylesheets unter der LGPL Lizenz veröffentlicht (siehe Anhang Abschnitt 7 auf Seite 70).

Es gibt ebenfalls ein XML Autorenpaket der Firma Infinity Loop `http://www.infinity-loop.de/`, darin enthalten ein Stylesheet für Word. Es ist Teil der Software Upcast Single Plus und steht - zur Zeit - allen

Nutzern von ELAT zur Verfügung[20]. Ich habe aus Kompatibilitätsgründen die - nicht sehr schönen - z.T. englischen, z.T. deutschen Layoutnamen beibehalten. Aber Vorsicht: eine in OOo importierte Absatzvorlage von *Upcast Single Plus* funktioniert nicht (!) im Zusammenhang mit dem vorhandenen Stylesheets! Worddokumente können Sie natürlich ohne Probleme in OOW importieren, um Sie anschließend entsprechend zu formatieren.

Und noch ein wichtiger Hinweis:

> Eine mit OOW geöffnete XML Datei speichert keinerlei (!) OO Formatierungs- und Absatzstile! Sie müssen das Dokument in einem SXW Format abspeichern, damit Sie beim nächsten Öffnen entsprechend weiterarbeiten können. D.h. für Sie konkret, dass sie das XML nur als Metaformat zum Abspeichern, aber *niemals* zum Bearbeiten des Dokumentes benutzen sollten. OO kann XML Dateien nicht nativ abspeichern. Die ”Speichern unter” und ”Öffnen” Dialoge sind im Sinne einer Import bzw. Exportfilters zu verstehen!

Besonderheiten zusammengefasst:

- Als Vorlagendatei muss ein spezielles Template eingesetzt werden, welches den Aufbau eines Knowledge-object im EML 1.0 Sinne unterstützt. Die Namen der Layouts dürfen nicht verändert werden.
- Bilder müssen als externe Referenzen eingebettet werden (verlinkt) und dürfen nicht *inline* angelegt werden.
- Mathematische Formeln lassen sich *nicht* mit MathML / StarMath einbinden, sondern müssen als Bilddateien abgespeichert werden. Sie werden daher als Abbildung geführt! OpenOffice definiert eine Formel eindeutig als MathML-Objekt. Leider müssen deshalb Formeln, als Abbildungen geführt und zusätzlich der zugehörige Absatz als *Formula* Stil gekennzeichnet werden. Zugehörige Beschriftungen sollten mit dem Absatzstil *Formula-text* gekennzeichnet sein.

1.2.1 Template als Benutzervorlage

Für die Arbeit mit OpenOffice ist eine Vorlage nötig, die eine benutzerdefinierte Absatzumgebung bereitstellt.

[20] Rechtlicher Hinweis: Diese Software ist jedoch kommerzielle Software und unterliegt entsprechenden Lizenzbedingungen. Mit OOo erstellte Dokumente, die dieses Template benutzten (d.h. dieses Stylesheet importieren), dürfen nicht widerrechtlich benutzt oder weitergegeben werden, wenn Sie nicht diese Software besitzen oder das Recht haben, diese zu verwenden. Seien Sie also vorsichtig, wenn Sie das Upcast Single Plus Template verwenden und im kommerziellen Umfeld tätig sind. In diesem Fall empfehle ich meine Vorlage, sie ist vollkommen frei im Rahmen der LGPL! Das Upcast Template wurde von mir in keiner Weise verwendet oder verändert.

Sie können diese Datei hier downloaden: EML Beispiel Gehirnschmalz http://www.blue-it.org/sites/default/files/EML_Beispiel_Wanda.sxw

1.2.2 Importfilter EML nach SXW

Der Importfilter versucht auf bestmögliche Weise die eben genannten Probleme zu umschiffen. Nur wenn Sie sich an diese Regeln halten, können Sie davon ausgehen verlustfrei zwischen EML und SXW Format hin- und her zu konvertieren.
Bitte beachten Sie, dass in diesem Prototyp nur einige wenige Aspekte umgesetzt werden konnten.

Sie können den Filter hier downloaden: sofftoeml.xsl http://www.blue-it.org/sites/default/files/sofftoeml.xsl

1.2.3 Exportfilter SXW nach EML

Vorbereitungen: Ein Exportfilter lässt sich nicht ohne weiteres auf der Befehlszeile mit *xalan* oder *saxon* testen. Dazu müssen folgende Vorbereitungen getroffen werden:

1. Extrahieren der Datei content.xml aus der OOW Datei
2. Aktualisieren des Pfades zur DTD Datei *office.dtd* in der `!DOCTYPE` Deklaration dieser Datei (z.B. /opt/OpenOffice.org1.1.2/share/dtd/officedocument/1_0/office.dtd). Sie können zu Testzwecken diese Datei auch einfach in das aktuelle Verzeichnis kopieren.
3. Je nachdem, auf welche Dateien die DTD bzw. die content.xml Datei referenziert kann dies, wie in 2. auch noch andere Dateien betreffen.

Jetzt erst ist gewährleistet, dass der Transformer über alle verwendeten DOM Elemente des OOo Formats, wie z.B. `office:document` oder `text:h` informiert ist, so dass wir Sie im Rahmen des XSLT Skriptes ansprechen können.

Sie können den Filter hier herunterladen: emltosoff.xsl http://www.blue-it.org/sites/default/files/emltosoff.xsl

1.3 Installation in Open Office

Wie ein Filter in OOo installiert wird, ist nicht Teil dieser Arbeit. Sie finden dazu in den von mir genannten Quellen genügend Hinweise. Im Anhang (siehe 2) habe ich jedoch ein paar Bildschirmschnappschüsse zur Installation beigelegt.

2 Diskussion

Der Filter deckt nur eine Teilmenge der im ELAT System bzw. in EML 1.0 vorgesehenen Elemente ab. Er ist auch nur für den Import eines *Knowledge-Objects* vorgesehen. Doch Sinn dieser Arbeit war es nicht, einen perfekten Filter zu bauen, sondern die Möglichkeiten aufzuzeigen, die Open Office bietet.

Von Benutzerfreundlichkeit - hinsichtlich der Integration in OO - sind die Filter noch weit entfernt. Mein Ziel, so viel Informationsgehalt wie möglich vom einen ins andere Format zu retten ist in Massen gelungen. Konvertiert man vom weniger reichhaltigeren Format EML zum reichhaltigeren OOW ist dies ohne Einschränkung machbar. Umgekehrt wird es schwieriger, da OOW kein Markup Editor ist, sondern dem Anwender zu viele Möglichkeiten zur Texztformatierung liefert.

Kapitel 5
Conclusio und Ausblick

Der Open Office Writer ist und bleibt eine Whysiwyg-Textverarbeitung.

Es wird schwer bleiben, dem Anwender die vielfältigen absoluten Formatierungsmöglichkeiten abzugewöhnen, wenn er sie permanent angeboten bekommt. Hier steht noch zu sehr der Layout Charakter bei der Texterstellung im Vordergrund. Für die Erstellung von Metadaten und Markuptexten ist ein grundsätzlich anderes Vorgehen notwendig.

OOo bietet mit Hilfe der ”Online-Ansicht” eine Möglichkeit, den Modus der Bearbeitung von ”A4” - und das macht nun im Zusammenhang mit Markup nun wirklich überhaupt keinen Sinn - auf eine Fließtextansicht umzuschalten. Wenn es jetzt noch gelänge, dem Benutzer die Sicht auf die Formatierungsmöglichkeiten zu beschränken, und ihm eine sinnvolle Auswahl zu Verfügung zu stellen, dann kämen wir einem für unsere Zwecke nützlichen Editor schon sehr nahe.

Wie schon am Anfang erwähnt, sind mir nur wenige Werkzeuge bekannt, die das Editieren von Markup konsequent auch in der Anwenderoberfläche umsetzen. Es sind dies:

- Lyx, TexMacs
- Adobe Framemaker und ähnliche Produkte anderer kommerzieller Anbieter

Leider bieten diese Anwendungen nicht immer die nötigen Import- bzw. Exportfilter oder entsprechende Schnittstellen an, um externe XML Dialekte einzubinden.

Ich bin sicher, dass die ernsthafte Erstellung und Pflege von Markuptexten nicht mit einer Whysiwyg Textverarbeitung sinnvoll ist. Inhalt und Layout sind zu sehr miteinander verknüpft - besonders aus der Sicht des Anwenders.

Es wäre wünschenswert, wenn Open Office es in zukünftigen Versionen ermöglicht, dem Programmierer in noch größerem Mass Einfluss

darauf zu nehmen, welchen Layoutmöglichkeiten der Autor im Zusammenhang mit einem konkreten Template zu sehen bekommt. Die Erstellung eines XSLT Filters dagegen ist eine reine Fleißarbeit.

Es wäre wünschenswert, wenn ausgezeichnete Markup Editoren, wie Lyx oder der Adobe Framemaker entsprechende Filtermechanismen, wie sie denen von OOo entsprechen, integrierten, um es Programmieren leicht zu machen, entsprechende Konvertierungstools zu entwickeln.

ELAT ist in der Entwicklung, hat aber gute Chancen, sich zu einem ernsthaften Konkurrenten auf dem E-Learning Sektor zu entwickeln. Vier Hindernisse sehe auf dem Weg dorthin: ELAT muss performanter werden. ELAT muss es schaffen, andere XML Formate zu integrieren, in dem die Browserkomponente entsprechende Funktionen anbietet. ELAT sollte für Entwickler so offen wie möglich gehalten werden und sich so nahe am EML Standard halten, wie möglich. Die Benutzerfreundlichkeit in Punkte Dokumentenmapping muss deutlich zunehmen!

Mein persönliches Ziel ist es, Kunden ein funktionierendes System von Editorkomponente und E-Learningapplikation zu bieten. Mit den bisherigen Ergebnissen kann man nur in Ansätzen zufrieden sein. Einem Kunden, der von XML nichts versteht oder kein entsprechendes Fachpersonal im Hause hat, kann man die hier vorgestellte Lösung ELAT/OO oder ELAT/Word für den produktiven Einsatz noch nicht empfehlen.

Ich habe für die kommende Zeit drei Ziele ins Auge gefasst auf dem Weg zum idealen (Opensource) Editor für Markuptexte:

1. Ich werde mich mit den Entwicklern des Open Source Editors Lyx in Verbindung setzen und helfen, dort an einem einheitlichen XML basierten Format mitzuarbeiten. Dies wäre eine exzellente Möglichkeit, Markuptexte für beliebige XML Formate zu erstellen und dem Anwender eine hervorragende, genau für diesen Zweck abgestimmte Programmoberfläche anzubieten.

2. Ich werde die Möglichkeiten des OOo Filtersystems weiter erforschen, insbesondere die des XMerge Frameworks. Auf der anderen Seite will ich sehen, welche Möglichkeiten es gibt, das Aussehen von OOo an die Zwecke eine Markupeditors anzupassen.

3. Ich werde weiter nach E-Learning Systemen suchen, die offen sind, beliebige XML Dialekte (wie MathML) problemlos integrieren, und möglichst modular aufgebaut ist.

Es gibt viel zu tun!

Verzeichnisse

Hinweis für die PDF Version: Bei einem Mausklick auf *Internetverweise* (http://...) öffnet der im Acrobat Reader eingestellte Webbrowser die zugehörige Seite.

Literaturverzeichnis

David Brownell. *SAX2*. O'Reilly, http://www.oreilly.com/catalog/sax2, 2002. ISBN 0-596-00237-8.

Brian Cameron, Martin Maher, and Mike Hayes. Xmerge Document conversion SDK. *http://xml.openoffice.org/xmerge/docs/XMerge_sdk.pdf*, 3 July 2002.

Allin Cottrell. Word Processors: Stupid and Inefficient. *http://ricardo.ecn.wfu.edu/ cottrell/wp.html*, 1999.

David Eisenberg. OpenOffice.org XML Essentials. *http://books.evc-cit.info/*, 30 July 2004.

Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Book of Design Patterns*. Addisson-Wesley, 1994.

IDA. IDA expert group conclusions and recommendations on open document formats. *http://europa.eu.int/ISPO/ida/*, 25 May 2004.

IgDV FH-Darmstadt et al. Growth of eLearning in higher education. *FH-Darmstadt*, 2003. Dipl.-Inform. Ulrich Gojny, Dipl.-Informationswirt Christian Vejrazka, Dipl.-Medienpraktikerin Angela LeichtweiÃ, Dipl.-Informationswirtin Cornelia Trebing, Prof. Dr. Horst F. RÃder, Prof. Dr. Gerhard Knorz, Prof. Dr. Udo Bleimann, Prof. Dr. Guo Li.

Henrik Just. Writer2latex. *http://www.hj-gym.dk/%7Ehj/writer2latex/*, October 2004.

Michael Kay. SAXON - The XSLT and XQuery Processor. *http://saxon.sourceforge.net/#F6.5.2*, 8 October 2004.

OMG (Object Modeling Group Inc.). *OMG Unified Modeling Language Specification*. OMG, 1999.

OpenOffice.org. Xmerge API Javadoc. *http://xml.openoffice.org/xmerge/docs/xmerge_javadoc.zip*, 2002.

OpenOffice.org. XML Packaging. *http://xml.openoffice.org/package.html*, 2004.

Axel Pospischil. LyX - ein Wysiwym Editor für LaTeX. *http://apos.blue-it.org -> Linux*, 21 January 2004.

Eric Steven Raymond. The cathedral and the basar. *The authors homepage*, 16 February 1998. URL http://www.catb.org/~esr/writings/homesteading/cathedral-bazaar/.

Prof. Dr. Röder, Horst, Dipl.-Inform. Gojny, Ulrich, and Dipl.-Informationswirt Vejrazka, Christian. *e-Learning Entwicklung im Hochschulbereich*. IgDV, 2002.

Peter Schüler. Hickhack um Office-XML. *CT Nr. 25*, 2002.

Joachim von Thadden and Sebastian Hetze. Offener Standard für Dokumentenformate. *http://www.linux-ag.de/linux/texte/dfs/index.html*, 29 August 2002.

Daniel Vogelheim. *Openoffice.org FAQ zu XML*. sun, http://de.openoffice.org/doc/faq/xml/, übersetzt von andreas hausmann edition, 1 September 2004. Daniel.Vogelheim@germany.sun.com.

Sandro Zic. OpenOffice and DocBook Tutorial. *http://www.zzoss.com/projects/oowdbk/*, 30 April 2003.

Abbildungsverzeichnis

Tabellenverzeichnis

Anhang

Kapitel 6
Glossar

RFC

Request for comment. Dies sind nummerierte Internetdokumente, die alle Belange von Internetstandards abdecken. RFC's sind das Hauptarbeitswerkzeug der Internet Engineering Steering Group, Internet Architecture Board, und anderer Internetgremien. RFCs werden durch das Internet Architecture Board (IAB) veröffentlicht und können auf u.a. http://www.ietf.org/rfc.html eingesehen werden. Das Verfahren, Standards über diesen unbürokratischen Weg zu erarbeiten, zu diskutieren und zu verbreiten unterscheidet das RFC Verfahren grundlegend von der Arbeitsweise anderer Gremien, wie z.B. der ISO.

Quelle: Wikipedia http://en.wikipedia.org/

Open Source

Der englische Ausdruck Open Source steht für quelloffen, einerseits in dem Sinne, dass der Quelltext eines Programms frei erhältlich ist, andererseits für "offene Quelle", also dass ein Werk frei zur Verfügung steht. Software gilt als Open Source, wenn sie bestimmte Kriterien erfüllt, die in ihrer Open-Source-Lizenz geregelt sind.

Der Ausdruck "Open Source Software" (OSS) wird auch oft als Synonym für freie Software verwendet, jedoch bezeichnen diese beiden Ausdrücke nicht unbedingt das Gleiche [...].

Das charakteristische für OSS sind vor allem die dem Anwender eingeräumten weitläufigen Verwertungsrechte. Unabhängig von den einzelnen Lizenzverträgen sind für OSS die folgenden drei charakteristischen Merkmale wesentlich:

1. Die Software (d. h. der Programmcode) liegt in einer für den Menschen lesbaren und verständlichen Form vor. [...]
2. Die Software darf beliebig kopiert, verbreitet und genutzt werden. [...]
3. Die Software darf verändert und in der veränderten Form weitergegeben werden. [...]

[...]
Diese Charakteristika werden detailliert in der Open Source Definition (OSD) der Open Source Initiative http://www.opensource.org festgelegt.
[...]

Quelle: Wikipedia http://en.wikipedia.org/

Markup

Beschreibt im Kontext einer *Markupsprache* die Struktur und das Aussehen eines spezifischen Dokuments. Inhalt und Form sind getrennt. Typische Markupsprachen sind HTML, TEX, XML...

MIME

Multipurpose **I**nternet **M**ail **E**xtensions . Ursprünglich für die Spezifizierung von E-Mail Anhängen gedacht. Die MIME-Typen beschreiben die Art der übertragenen Daten und stellen dadurch sicher, dass eine Software eindeutig Dokumententypen identifizieren kann. Ein *MIME Typ* besteht aus dem *Medientyp* (z.B. Text, Bild, Video...) und einem *Subtyp*, dem Dateiformat (z.B. html, gif, mpg).

Servereigene Dateiformate, die meist auf einem solchen ausgeführt werden, sind dabei meist mit eine *"x-"* eingeleitet .

Quelle: u.a. Selfhtml http://www.tu-chemnitz.de/docs/selfhtml/diverses/mimetypen.htm, IANA http://www.isi.edu/in-notes/iana/assignments/media-types/, Requests for Comments (RFCs) 2045, 2046 und 2077 http://www.tu-chemnitz.de/docs/selfhtml/intro/hilfsmittel/dokus.htm#rfcs

SAX API

SAX ist eine an Java adaptierte API für XML. Sie stellt mittlerweile eine "de facto" Standard für den Zugriff auf XML Dokumente dar und ist für viele Programmiersprachen implementiert.

Quelle: u.a. SAX http://www.saxproject.org/, O'Reilly Online http://www.oreilly.com/catalog/sax2/, Brownell [2002]

Kapitel 7
Erläuterungen

1 Historie von XML

An dieser Stelle ein kurzes Zitat über die beteiligten Personen, welche bei der Entstehung von XML eine Rolle gespielt habe. Es hebt die Rolle von SUN Microsystems hervor.

Quelle: Henning Behme & Stefan Mintert, 2000, http://www.linkwerk.com

> ”XML wurde von einer XML-Arbeitsgruppe (ursprünglich bekannt als das SGML Editorial Review Board) entwickelt, die 1996 unter der Schirmherrschaft des World Wide Web Consortium (W3C) gegründet wurde. Den Vorsitz hatte Jon Bosak von Sun Microsystems inne, unter aktiver Beteiligung einer XML Special Interest Group (ehemals SGML-Arbeitsgruppe), die ebenfalls vom W3C organisiert wurde. Die Mitglieder der XML-Arbeitsgruppe sind in einem der Anhänge genannt. Dan Connolly fungierte als Kontaktperson zwischen der Arbeitsgruppe und dem W3C.
>
> [...]
>
> Diese Spezifikation enthält, zusammen mit den verwandten Standards (Unicode und ISO/IEC 10646 für Zeichen, Internet-RFC 1766 für Codes zur Identifikation der Sprache, ISO 639 für Codes von Sprachnamen und ISO 3166 für Ländernamen-Codes), alle Informationen, die notwendig sind, um XML in der Version 1.0 zu verstehen und um Programme zu schreiben, die sie verarbeiten.”

2 Filtertemplate erstellen in Open Office Writer

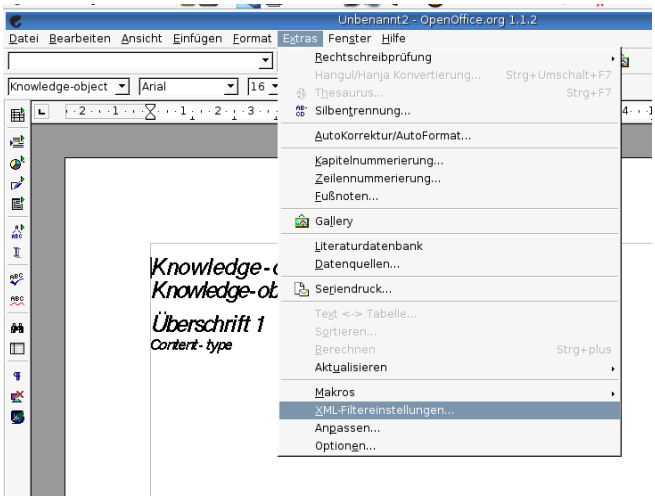

Abbildung 8: Open Office Writer: Filterdialog aufrufen

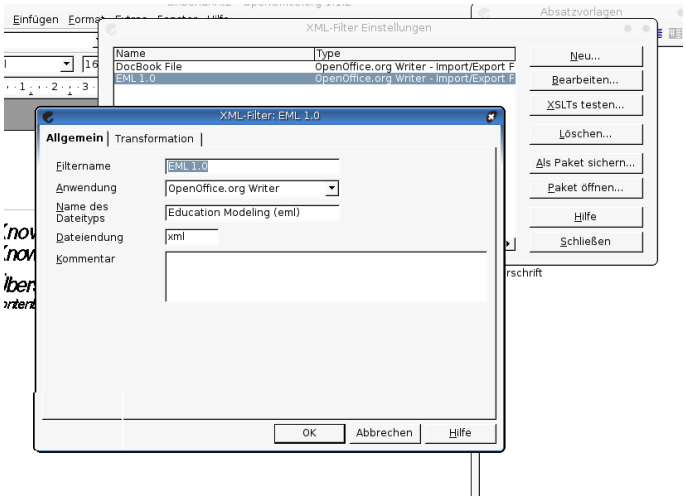

Abbildung 9: Open Office Writer: Filterdialog
Allgemeine Einstellungen zum Filter: Erscheinungsbild im Öffnen/Speichern unter Dialog, Dateiendung

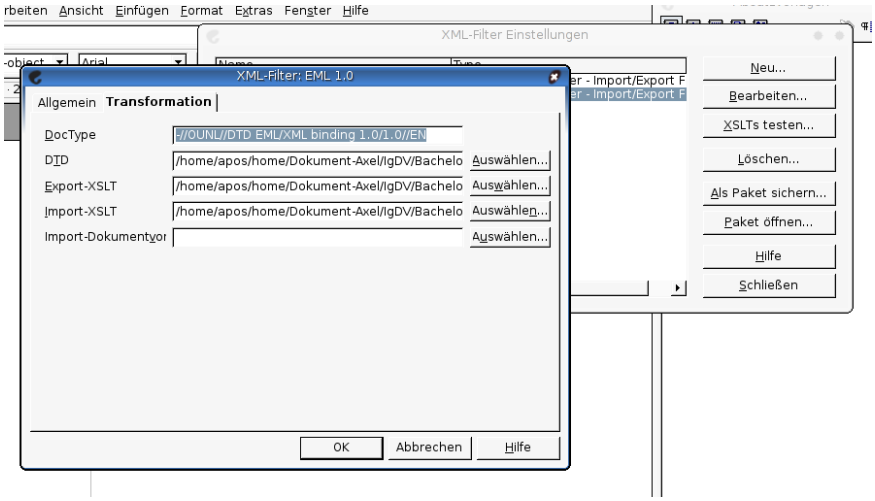

Abbildung 10: Open Office Writer: Filterdialog aufrufen
Beschreibung der Transformation durch: XML Doctype, DTD zur Validierung, Import- bzw. Exportschema und zur verwendende OO Writer Vorlagendatei (*.stw).

3 XML Beispiele

”Hirnschmalz” Beispiel

Vollständiger Quelltext zum Inhalt der *content.xml* Datei des OO Writer Dokuments ”Hirnschmalz” (siehe auf Seite 13 und auf Seite 27).

```
<office:body xmlns:office="http://openoffice.org/2000/office">
    <text:sequence-decls xmlns:text="http://openoffice.org/2000/text">
      <text:sequence-decl xmlns:text="http://openoffice.org/2000/text"
                          text:display-outline-level="0"
                          xmlns:text="http://openoffice.org/2000/text"
                          text:name="Illustration"
                          xmlns:text="http://openoffice.org/2000/text"
      [... weitere Deklarationen...]
    <text:p xmlns:text="http://openoffice.org/2000/text"
            text:style-name="P1"
            xmlns:text="http://openoffice.org/2000/text" >
             Hirnschmalz
    </text:p>
    <text:p xmlns:text="http://openoffice.org/2000/text"
            text:style-name="P2"
            xmlns:text="http://openoffice.org/2000/text" >
             Experiment
    </text:p>
    <text:p xmlns:text="http://openoffice.org/2000/text"
            text:style-name="Section-Ueberschrift-1"
            xmlns:text="http://openoffice.org/2000/text" >
             Wirkung von Koffein auf die psychokinetischen Fähigkeiten
```

```
    </text:p>
    <text:p xmlns:text="http://openoffice.org/2000/text"
            text:style-name="Author"
            xmlns:text="http://openoffice.org/2000/text" >
            von Wanda
    </text:p>
    <text:p xmlns:text="http://openoffice.org/2000/text"
            text:style-name="P"
            xmlns:text="http://openoffice.org/2000/text" />
    <text:p xmlns:text="http://openoffice.org/2000/text"
            text:style-name="P"
            xmlns:text="http://openoffice.org/2000/text" >
             Das Experiment setzt sich aus einer Versuchsperson,
einer Dose mit Koffein versetzter Cola und einem Goldfischglas zusamme
Die Fähigkeit, einen Goldfisch durch die menschliche Willenskraft im
Kreis herumschwimmen zu lassen, ergibt sich aus der bekannten Gleichun
    </text:p>
    <text:p xmlns:text="http://openoffice.org/2000/text"
            text:style-name="P"
            xmlns:text="http://openoffice.org/2000/text" />
    <text:p xmlns:text="http://openoffice.org/2000/text"
            text:style-name="P3"
            xmlns:text="http://openoffice.org/2000/text" >
      <draw:image xmlns:draw="http://openoffice.org/2000/drawing"
                  svg:width="2.962cm"
                  xmlns:svg="http://www.w3.org/2000/svg"
                  text:anchor-type="paragraph"
                  xmlns:text="http://openoffice.org/2000/text"
                  svg:height="0.741cm"
                  xmlns:svg="http://www.w3.org/2000/svg"
                  xlink:type="simple"
                  xmlns:xlink="http://www.w3.org/1999/xlink"
                  xlink:href="gleichung_1.png"
                  xmlns:xlink="http://www.w3.org/1999/xlink"
                  draw:z-index="0" xmlns:draw=
                            "http://openoffice.org/2000/drawing"
                  draw:name="Grafik1"
                  xmlns:draw="http://openoffice.org/2000/drawing"
                  xlink:show="embed"
                  xmlns:xlink="http://www.w3.org/1999/xlink"
                  xlink:actuate="onLoad"
                  xmlns:xlink="http://www.w3.org/1999/xlink"
                  draw:style-name="fr1"
                  xmlns:draw="http://openoffice.org/2000/drawing" />
      <text:span xmlns:text="http://openoffice.org/2000/text"
            text:style-name="T1"
            xmlns:text="http://openoffice.org/2000/text" >
             Gleichung
      </text:span>
      <text:span xmlns:text="http://openoffice.org/2000/text"
                 text:style-name="Formula-text"
```

```
                    xmlns:text="http://openoffice.org/2000/text" >
                     1
          </text:span>
        </text:p>
        <text:p xmlns:text="http://openoffice.org/2000/text"
                text:style-name="P4"
                xmlns:text="http://openoffice.org/2000/text" >
          <text:span xmlns:text="http://openoffice.org/2000/text"
                     text:style-name="Formula-text"
                     xmlns:text="http://openoffice.org/2000/text" >
                      P ist dabei die Wahrscheinlichekeit, dass der Fisch
in einem vorgegebenen Zeitinterwall eine Kreisbahn schwimmt, m ist
der IQ des Fisches, M ist der IQ der Versuchsperson und d ist die
Entfernung zwischen dem Fisch und der Versuchsperson.
          </text:span>
        </text:p>
      </office:body>
```

4 ELAT Knowledge Object

4.1 Absatzvorlagen für OOo

Dies sind die Namen der Abstzvorlagenstile, wie sie mit Hilfe des Stylists von OpenOffice erstellt wurden und im XSLT Stylesheet für den Import bzw. Exportfilter genannt sind. Für spätere Versionen sollten Übersetzung angefertigt werden. Die Namen entsprechen aus Kompatibilitätsgründen denen des Wordtemplates der Firma *Infinity Loop* http://www.infinity-loop.de/. Jedoch sind nicht alle Absatzvorlagen umgesetzt, nur die fett gedruckten Stile wurden implementiert.

Absatzstile (in Alphabetischer Reihenfolge)

Author
Book
Code-block
Content-type
Creator
Description
Figure
Formula
Knowledge-object
Knowledge-object-subtitle
Lemma
Metadata
P

Section-Ueberschrift-1
Section-Ueberschrift-2
Section-Ueberschrift-3
Section-Ueberschrift-4
Section-Ueberschrift-5
Section-Ueberschrift-6
Section-Ueberschrift-7
Section-Ueberschrift-8
Table-title
WW-Dokumentenstruktur

Zeichenvorlagen

Formula-source
Formula-text
Figure-source
Figure-text

4.2 Ausschnitte aus der EML 1.0 DTD für das `Knowledge-object`

Als Attributwert für das `Knowledge-object` ist eine `Component` Entity definiert:

```
<!ENTITY % Component
        "%May-have-link-name;
         %Reusable;
         Reusability (Reusable | Not-reusable) &#34;
         Not-reusable&#34;
         Type CDATA #IMPLIED ">
```

Die Entities *%May-have-link-name* und *%Reusable* sind dabei wie folgt deklariert:

```
<!ENTITY % May-have-link-name
          "Link-name CDATA #IMPLIED ">
<!ENTITY % Reusable
           "%May-have-identifier;
           EML-version CDATA #FIXED &#34;1.0&#34;
           Version CDATA &#34;1.0.0&#34;
           Worldwide-unique-id CDATA #IMPLIED ">
```

4.3 Ausschnitt aus der XSL Schema Definition für die Definition des `Metadata` Elements.

Quelle und Kontakt: ELATnet http://www.elatnet.de

```
<xs:element name = "Metadata">
  <xs:complexType>
   <xs:sequence>
     <xs:element name = "Title">
      <xs:complexType mixed = "true">
        <xs:choice minOccurs = "0" maxOccurs = "unbounded">
          <xs:element ref = "Internet-ref"/>
          <xs:element ref = "Emphasis"/>
          <xs:element ref = "Term"/>
          <xs:element ref = "Special-inline"/>
          <xs:element ref = "Figure-source"/>
          <xs:element ref = "Formula-source"/>
        </xs:choice>
      </xs:complexType>
     </xs:element>

   <xs:choice minOccurs = "0" maxOccurs = "unbounded">
     <xs:element name = "Subtitle">
      <xs:complexType mixed = "true">
          [... wie Title ...]
      </xs:complexType>
      </xs:element>
   </xs:choice>
  </xs:sequence>
 </xs:complexType>
</xs:element>
```

5 E-Learning Resourcen

Um sich einen Überblick über die Relevanz von E-Learning im Bildungssektor, aber auch der Industrie zu machen, empfehle ich eine intensive Internetrecherche. Als Ausgangspunkt können folgende Links dienen, es handelt sich nur um einen kleinen Ausschnitt interessanter Anlaufstellen zum Thema:

5.1 Standards

- IEEE LOM Standard http://ltsc.ieee.org/wg12/
- Vortrag zur Historie und Entstehung von LOM http://www.informatik.uni-erlangen.de
- IMS Global Learning Consortium http://www.imsglobal.org/
- Metadatenstandards http://www.mmcc.charite.de/technologie/technologie.htm, Charité
- EML http://eml.ou.nl/eml-ou-nl.htm, an der *Open University of Netherlands*

5.2 Allgemein

- MIT Media Laboraty http://learning.media.mit.edu, das MIT startet einen Grossversuch in Sachen E-Learning. Nicht nur Inhalte, sondern auch ein passendes Web Content Management sollen MIT Kurse einer breiten Öffentlichkeit zugänglich machen.
- Wissensnetz http://www.wissensnetz.de/index.html, hier gibt es einen interessanten, speziellen Bereich für E-Learning.
- Trainerlink - die besten Adressen im Weiterbildungsweb http://www.trainerlink.de/l
- Internet Basiswissen http://www.wast2000.de/xml/quellen.html

5.3 Presse / Unternehmen

- e-learning-presseclub 21. - 23. Januar 2004 in Hagen http://www.e-learning-presseclub.de/elpc/hagen2004/roundtableshagen.htm
- IBM - Im Fokus: E-Learning http://www-5.ibm.com/services/de/cnslt_pov/e-learning_qa.html

5.4 Hochschulbereich

- Studieren im Netz http://www.studieren-im-netz.de/

6 OASIS Anforderungen an ein Dokumentenformat

Zitat aus OASIS http://www.oasis-open.org/committees/ -> OpenOffice

> [...]
> The resulting file format must meet the following requirements:
> 1. it must be suitable for office documents containing text, spreadsheets, charts, and graphical documents,
> 2. it must be compatible with the W3C Extensible Markup Language (XML) v1.0 and W3C Namespaces in XML v1.0 specifications,
> 3. it must retain high-level information suitable for editing the document,
> 4. it must be friendly to transformations using XSLT or similar XML-based languages or tools,

5. it should keep the document's content and layout information separate such that they can be processed independently of each other, and

6. it should 'borrow' from similar, existing standards wherever possible and permitted.

[...]

7 GNU Lesser General Public License Version 2.1

Nachfolgend die Linzenz für alle in dieser Arbeit erstellten Quelltexte. Der von mir verwendete Code wurde entweder vollständig selbst erstellt, oder entstammen aus Sourcen, die anderen Lizenzen unterliegen. Dies geschah immer unter Nennung der Autoren und entsprechenden Lizenzformen .

Copyleft (Deutsche Version) http://www.gnu.org/copyleft/copyleft.de.html

GLPL (Deutsche Version) http://www.gnu.org/copyleft/lesser.de.html

```
# The Contents of this file are made available subject to the terms of
# of the following license
#
# GNU Lesser General Public License Version 2.1
# =============================================
#
# author: Axel C. R. Pospischil, 2004, email: pospischil@blue-it.org
#
# This library is free software; you can redistribute it and/or
# modify it under the terms of the GNU Lesser General Public
# License version 2.1, as published by the Free Software Foundation.
#
# This library is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
# Lesser General Public License for more details.
#
# You should have received a copy of the GNU Lesser General Public
# License along with this library; if not, write to the Free Software
# Foundation, Inc., 59 Temple Place, Suite 330, Boston,
# MA 02111-1307 USA
```

Wiederverwendbarkeit	Lerninhalt soll modularisiert vorliegen. Kleine Einheiten von Wissensbausteinen sollen zu in verschiedenen Kursen wiederverwendet werden können.
Zusammenarbeit	Wissensbausteine sollen miteinander agieren können, unabhängig vom Ersteller oder der verwendeten Software.
Dauerhaftigkeit	Wissensbausteine sollen widerstandsfähig sein gegen Veränderungen, die durch wechselnde Darstellungstechniken oder Übertragungsmedien hervorgerufen werden könnten. Sie dürfen dadurch nicht unbrauchbar werden.
Zugänglichkeit	Lerninhalte sollen überall und jederzeit zugänglich sein. Insbesondere sollen Lerninhalte netzwerkübergreifend zugänglich sein und wiederverwendet werden können.

Tabelle 2: Learning Object Paradigma

ANSATZ	VORTEILE / NACHTEILE
Importfilter in ELAT Autorenoberfläche	(+) Für den Kunden am einfachsten zu bedienen (-) Aufwendige Integration in ELAT Software (-) Entwicklungszeit (-) Zusätzlicher Aufwand bei Wartung durch enge Kopplung von OpenSource Code (externe Entwicklergemeinde) und proprietärem Code (ELAT) (-) Lizenzproblematik
ELAT Plugin	Siehe Importfilter
Standalone Filter zur Konvertierung von Worddokumenten	(+) Sofort einsetzbares, kommerzielles und supportetes Produkt das funktioniert! (+) Batchverarbeitung möglich (-) Umständlich zu bedienen, da nicht in Word integriert (-) nicht direkt in Word integriert
Exportfilter für Word	(-) Hoher Entwicklungsaufwand, Neuland
(XSLT-basierter) Import- bzw. Exportfilter für OOo	(+) Filtererstellung, Schnittstellen und Dateiformate ausgezeichnet dokumentiert (+) Volle Integration in OOe möglich durch vorhanden Filterkonfigurationsdialog (+) Relativ einfache Implementierung mit XSLT Stylesheets (+) Ist gleichzeitig auch als Java Kommandozeilenfilter konstruierbar, d.h. Batchverarbeitung möglich (+) gut dokumentiert (+) Keine Abhängigkeiten von ELAT Entwicklern und OpenSource Gemeinde
Xmerge basierter Import-, Export- bzw. Mergefilter für OOo	(+) Als Exportfilter in OOo integriert (bessere Workflowunterstützung) (+) Ist gleichzeitig auch als Java Kommandozeilenfilter konstruierbar, d.h. Batchverarbeitung möglich (+) Standardanwendung in C++ oder Java mit entsprechenden Möglichkeiten (+) gut dokumentiert (-) höherer Entwicklungsaufwand

Tabelle 4: Varianten des Imports von Office Dokumenten in das ELAT System