Empfehlungen zur Nutzung eines Textverarbeitungswerkzeugs zur Erstellung von XML-basierten E-Learning-Inhalten

Friedrich Meincke [1], Ulrike Lucke [2], Djamshid Tavangarian [1]

[1] Universität Rostock, Institut für Informatik, A.-Einstein-Str. 21, 18059 Rostock
vorname.nachname@uni-rostock.de

[2] Universität Potsdam, Institut für Informatik, A.-Bebel-Str. 89, 14482 Potsdam
vorname.nachname@uni-potsdam.de

Abstract: Single Source Authoring von Lehrmaterial, welches auf einer abstrakten (XML-)Beschreibung basiert, ist sehr mächtig, jedoch auch sehr komplex in der Umsetzung. Eine große Flexibilität der erzeugbaren Ausgabemedien (z.B. Material für Bildschirm oder Print, für Lehrende oder Lernende oder für verschiedene Schwierigkeitsgrade) geht einher mit einem aufwändigen Authoring-Prozess. Es gibt eine Reihe von Werkzeugen für die Inhaltserstellung mit jeweils eigenen Vor- und Nachteilen. Auf der Basis früherer Arbeit an anderen Lösungen fiel unsere Wahl auf den OpenOffice.org Writer als ausgereifte und flexible Plattform. Eine Lösung für die Sprache <ML>³ wurde entwickelt. Der Artikel gibt einen Überblick über notwendige Implementationsschritte und diskutiert damit verbundene Problemstellungen, was zu generellen Richtlinien bezüglich der Nutzung eines Textverarbeitungswerkzeugs zum Single Source Authoring hinführt. Darüber hinaus wird auf Aspekte der Feinabstimmung einer Beschreibungssprache sowie auf die Organisation des Authoring-Prozesses eingegangen.

1 Einleitung

Mit der eXtensible Markup Language (XML) wurde ein de-facto Standard für die Beschreibung zukunftsorientierter Bildungsmaterialien etabliert. Wo wiederholte Themenbeschreibungen in Form von Webseiten (HTML), Manuskripten (DOC) oder Folien (PPT) ein hohes Maß an Redundanz produzieren, kann eine Obermenge dieser Inhalte als abstrakter und kompakter XML-Quellcode beschrieben werden, welcher sich leicht in alle erforderlichen Dokumentenformate transformieren lässt. Dies wird unter dem Begriff Single Source Authoring verstanden [Ha93]. Vorteile liegen nicht nur in redundanzfreien Beschreibungen, die sich leicht pflegen lassen, sondern auch in der allgemeinen Flexibilität des Materials, was zu einem hohen Grad an Adaptivität und Wiederverwendung führt [BM02]. Als Beispiel seien mächtige Mechanismen zur automatisierten Erstellung individueller Kursunterlagen genannt, die auf Inhalten arbeiten, welche sich lediglich durch eine parameterisierte Transformation an unterschiedliche Szenarien anpassen lassen.

Ein Basiskonzept dahinter ist die Definition des Lernobjekts (LO) als kleinere, eigenständige und wiederverwendbare Einheit von Lernmaterial [Fr04], welche für die Kursgenerierung einfach mit anderen LOs kombiniert werden. Die sogenannte Modularisierung kann sowohl auf der Ebene der Meta-Information (ein Set Meta-Daten pro LO, zusätzliche Meta-Daten-Untermengen für interne Teilgebiete) als auch auf der Ebene der Datei-Granularität (eine oder mehrere Dateien pro LO) angesehen werden. Bedauerlicherweise erfordert hohe Wiederverwendbarkeit eine geringe Kontextspezifität, was aus pädagogischer Sicht eher schädlich ist. Adaptive LOs, die auf einer abstrakten Beschreibung (wie XML) basieren, sind die Lösung für dieses Problem [Ro04].

Eine Reihe von Beschreibungssprachen wurden auf diesem Gebiet entwickelt, wie z. B. die eLearning Markup Language (eLML) [FB06], die Learning Material Markup Language (LMML) [Fr02], die Educational Modeling Language (EML) [KM04] oder die Multidimensional LearningObject and Modular Lectures Markup Language (<ML>³) [Lu03]. All diese Sprachen haben ihren eigenen Fokus und ein Set von Werkzeugen für den LO-Lebenszyklus.

Jedoch ist das hohe Abstraktionsniveau ein Nachteil des Single Source Authoring, was ein Problem während des Authoring-Prozesses aus technischer und pädagogischer Sicht darstellen kann. Letzteres wird durch Vorlagen und Wizards für die Nutzerunterstützung adressiert, wobei Lernobjekte sogar gleichzeitig für verschiedene pädagogische Strategien beschrieben werden können [LM10]. Die technischen Aspekte bei der Bearbeitung von XML-basierten Lerninhalten wurden in den vergangenen Jahren durch eine Reihe von Entwicklungen untersucht. Abbildung 1 bietet einen Überblick anhand einer einfachen Klassifikation der unterliegenden Ansätze.

Diese Diversität der Ansätze und Werkzeuge wird durch die Vielseitigkeit der Anforderungen in unterschiedlichen Anwendungsszenarien gerechtfertigt. Es gibt keine allgemein beste Lösung für alle Nutzertypen, was sich wie folgt aufschlüsseln lässt:

- Kommerzielle Applikationen erfordern Effizienz und Verlässlichkeit.
- Im öffentlichen Sektor sind geringe Kosten das entscheidende Kriterium.
- In der Forschung hingegen ist Open Source ein wertvolles Attribut.

Wie die Häkchen in der Grafik zeigen, liegen bereits eine Reihe von Erfahrungen von früheren Entwicklungen vor. Web-basierte Ansätze sind aufwändig zu entwickeln, können darüber hinaus auch zum Flaschenhals werden. Sie sind besonders in Kombination mit zentralen Inhalts-Repositories geeignet. Beispiele sind das eLML Plug-In für den Browser-basierten Firedocs-Editor [FB06] oder die Generierung XML-basierter LOs aus einem Wiki mit media2mult [GV08]. In der Kategorie der alleinstehenden Werkzeuge stellen XML-Editoren den direkten Ansatz dar, was jedoch für unerfahrene Nutzer unangebracht ist. Andererseits ist der Umgang mit Editoren, die speziell für eine Beschreibungssprache ausgerichtet sind wie die XMLeditools für <ML>³ [Gr09] für Nutzer mit wenig Technikkenntnissen einfach in der Bedienung, was

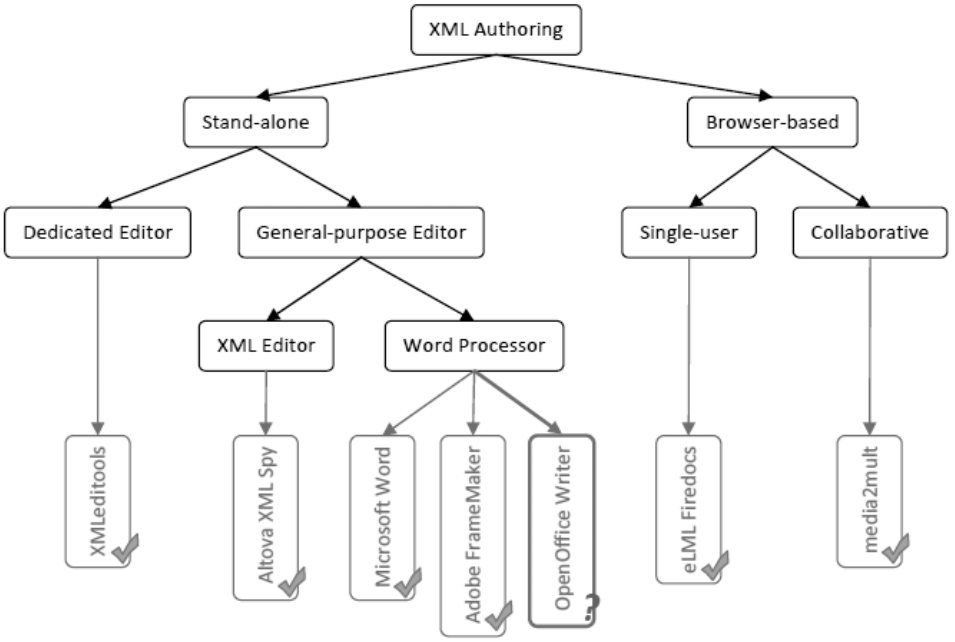

Abbildung 1: Eine einfache Klassifikation von Authoring-Ansätzen für XML-basierte Lehrmaterialien und Beispielwerkzeuge

jedoch mit einem hohen Entwicklungsaufwand verbunden ist, wobei sich viele Komponenten wiederverwenden lassen (wie Textformatierungen, Multimedia-Integration, Tabellen und Mathematische Formeln). Somit ist die Bereitstellung gebräuchlicher Textverarbeitungsmöglichkeiten ein nützlicher Ansatz. MS Word wird vielseitig und vielerorts eingesetzt, ist jedoch unzureichend dokumentiert. Daher werden heuristische Import/Export-Mechanismen benötigt. Ferner ist die Abhängigkeit vom Anbieter kritisch zu betrachten [GH04]. Adobe FrameMaker ist mächtig und stabil, ermöglicht eine direkte Abbildung von XML-Konzepten in der internen Dokument-struktur [Lu06], ist jedoch mit hohen Kosten und einer beachtlichen Einarbeitungszeit verbunden. Angesichts dieser Ausgangslage stellt der OpenOffice.org (OOo) Writer eine gute Alternative dar. Das Programm ist Word nachempfunden und damit leicht zu bedienen, weitverbreitet, gut dokumentiert sowie Open Source und stellt Mechanismen für den Import/Export verschiedener Formate bereit. Es gibt auch Lösungen für XML-basierte Beschreibungssprachen jenseits des Bildungssektors, z. B. für das DocBook-Format [Ri03].

Dieser Artikel beschreibt unsere Bemühungen, das Bearbeiten von <ML>[3]-Inhalten mit dem OOo Writer zu ermöglichen, sowie unsere Erfahrungen bei diesem Vorhaben. Dies führt zu allgemeinen Richtlinien für ähnliche Ansätze, nicht nur in Bezug auf die Implementierung des Werkzeugs, sondern auch für die Sprachenentwicklung und die Organisation des Authoring-Prozesses.

2 Eine prototypische Implementierung für OpenOffice.org

Unsere Strategie der Bearbeitung von <ML>³ im OOo Writer ergibt einen Kreislauf über Import, Bearbeitung und Export. Dies verweist auf das sogenannte XML-Round-Tripping [BO04]. Abbildung 2 vermittelt einen Eindruck dieses Prozesses sowie die Kombination mit anderen Inhaltsquellen und Anwendungsszenarien. Es sei angemerkt, dass die <ML>³-Dateien eines LOs als zentrale Schnittstelle zu anderen Komponenten und Werkzeugen fungieren. Deshalb ist die Erhaltung der Dokumentstruktur über Konvertierungen hinweg von hoher Bedeutung.

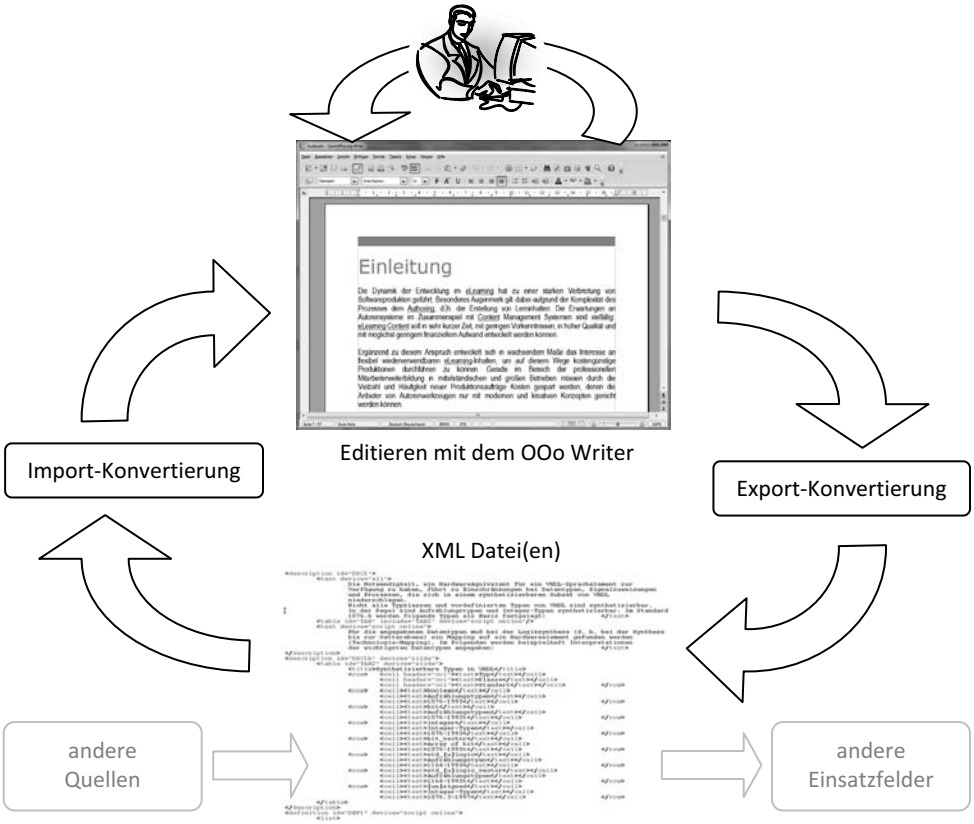

Abbildung 2: Die Strategie des XML-Round-Tripping mit OpenOffice.org

Die folgenden Abschnitte beschreiben unsere OOo-Applikation für <ML>³ als eine geeignete Beschreibungssprache für E-Learning-Inhalte. Ein Großteil der gewonnenen Erkenntnisse lässt sich jedoch auch auf andere XML-Sprachen übertragen. Wir nutzen hierfür grundlegende OOo-Mechanismen für unsere Erweiterung; dazu zählen deren API, Makro-Definitionen, Filter und Dokumentvorlagen.

2.1 Importmechanismen

Das interne Dokumentformat des OOo Writer ist XML-basiert. Daher ist die Abbildung von Elementen auf die <ML>³-Sprache über einen konventionellen Filter möglich, welcher die eXtensible Stylesheet Language (XSL) nutzt. Obwohl jede Konvertierung das Problem des möglichen Verlustes semantischer Information in sich birgt, ähneln sich die Eigenschaften von Lernobjekten und üblichen Textdokumenten sehr (abgesehen von den pädagogischen Informationen in einem LO), sodass die Transformation unmittelbar abgeleitet werden kann.

Wir implementierten eine XSL-Transformation als Import-Filter, der alle Elemente der <ML>³-Sprache in Konstrukte des OOo Writer verlustfrei abbildet. Zusätzlich ruft der Filter eine spezielle Dokumentenvorlage auf, die mit Formatierungsregeln und Makros erweitert wurde. Eine Konvertierung ist ein sequentieller Ablauf durch die <ML>³-Dateien, bei dem <ML>³-Strukturen und Elemente durch die des OOo Writer ersetzt werden. Dies ist eine Standard-XSL und soll nicht weiter im Detail erörtert werden. Die Abbildung erfolgt eineindeutig und kann über den Export wieder rückgeführt werden.

Ein Problem, welches uns beschäftigte, ist der Umgang mit LOs, die aus mehreren Dateien bestehen. Je größer ein LO wird, umso mehr tendiert ein Autor (oder eine Gruppe von Autoren) dazu, feinere Unterteilungen in Abschnitte vorzunehmen. Der OOo Writer kann damit jedoch nicht direkt umgehen. Daher entschieden wir uns dazu, alle XML-Quelldateien eines <ML>³-LOs in einem OOo Writer Dokument zu integrieren. Wir verwenden Markierungen, um die ursprüngliche Dateistrukturen samt Namen zu erhalten, damit diese über den Export wieder erzeugt werden können. Für den Fall, dass die ursprüngliche Dateistruktur nicht erhalten bleiben muss, kann dieser Schritt übergangen werden. Jedoch würde das die Interoperabilität mit anderen Werkzeugen des <ML>³-Frameworks reduzieren und war daher nicht unsere erste Wahl.

Ein Problem, das wir erwarteten, dem wir jedoch nicht begegneten, betrifft die Rekursion oder maximale Tiefe verschachtelter Strukturen. Anders als bei gewöhnlichen Textverarbeitungs-Werkzeugen sind in <ML>³ keine speziellen Formate für die erste/zweite/dritte... Überschriften-Ebene definiert. Substrukturen werden nur durch die Nutzung der entsprechenden Tags zur Verschachtelung modelliert, ohne dabei die Ebene der Vertiefung in der Struktur explizit zu definieren. Deshalb ist die Verschachtelungstiefe theoretisch unbegrenzt. Der OOo Writer zeigte jedoch keine Probleme damit.

Eine Herausforderung, an der noch gearbeitet wird, ist die Integration von MathML-Formeln und anspruchsvollen interaktiven Abläufen (wie Drag & Drop Aufgaben) im OOo Writer. Hier sind unter Umständen zusätzliche Plug-Ins erforderlich. Bisher werden derartig problematische Elemente zwar importiert und innerhalb der Dokumentstruktur referenziert, bleiben jedoch im OOo Writer nur als einfache Markierung sichtbar und sind damit begrenzt editierbar.

2.2 Visualisierungstechniken

Ähnlich wie der Import bestehender <ML>³-Dokumente werden die gleichen Definitionen zu Struktur, Formatierung und spezieller Funktionalität neuen <ML>³-Dokumenten hinzugefügt, welche vom OOo Writer aus erstellt werden. Den besonderen Merkmalen von <ML>³ wurde mit einer eigenen Dokumentenvorlage begegnet.

Ein primäres Feature dieser Vorlage ist die Vorgabe von Formatierungsrichtlinien für alle definierten Elemente und Attribute. Das betrifft Schriftarten, Größen, Stile, Farben, Abstände und so weiter. Dies ist ein elementarer Beitrag, um dem Autor eine Vorstellung über das finale Aussehen des LOs für seine Studenten zu geben (what you see is what you get, WYSIWYG), und sollte nicht unterschätzt werden. Da es keinen einzigen Stil für ein LO unabhängig vom gewünschten Ausgabeformat gibt, mussten wir eine Möglichkeit finden, die typische Erscheinung im Internet, in Print- und in Folienform abstrahieren zu können. Abbildung 3 zeigt ein einfaches Beispiel für eine solche Formatierung. Dieses LO beschreibt, wie man <ML>³-Dateien mit Adobe FrameMaker bearbeitet – nun im OOo Writer geöffnet.

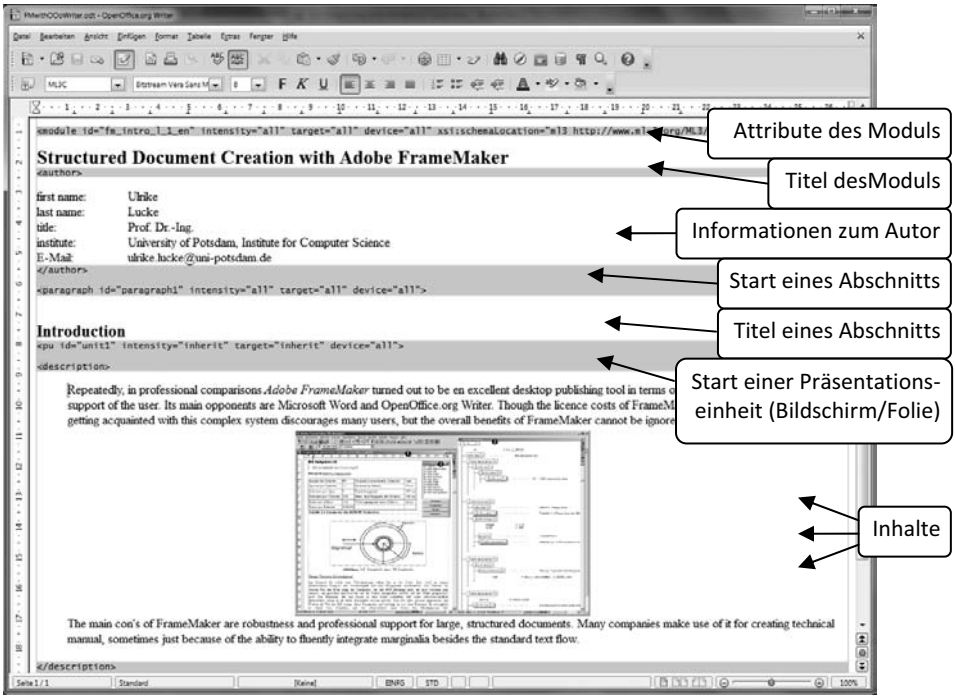

Abbildung 3: Die entwickelte Dokumentvorlage überträgt <ML>³-Eigenschaften in OOo-Stile

Attribute können entweder in der Dokumentansicht oder über einen Dialog bearbeitet werden. Schrifttypen (wie Schriftart, Stil oder Farbe) veranschaulichen Informationen über abstrakte Eigenschaften eines Elementes wie „nur für Druckausgabe“ oder „Lehrerversion“. Dialoge werden verwendet, um notwendige Informationen zu

bearbeiten, die nur schwer visualisiert werden können, wie beispielsweise Meta-Daten, IDs oder Referenzen.

Eine Schlüsseleigenschaft der XML-Dateien ist Strukturierung. Jedoch ist die Stuktur eines <ML>³-Dokuments nicht direkt ersichtlich, wenn es im OOo Writer geöffnet wird. Es gibt keine Baum-Darstellung. Dies stellt einen Unterschied zu anderen Ansätzen des strukturierten Dokumenten-Authorings dar, z. B. mit Adobe FrameMaker. Die Struktur lässt sich nur über neue Überschriften mit Nummerierungen, Zeileneinzüge oder Abgrenzungen (grauer Hintergrund) implizit wahrnehmen. Diese Strukturelemente sollen weitestgehend gegenüber Fehleingaben geschützt werden. Wir entschlossen uns, auf die Baum-Ansicht zu verzichten, um Verständnisproblemen, die Autoren bei früheren Realisierungen (wie mit Abobe FrameMaker) hatten, zu begegnen. Z. B. erkannten einige Autoren nicht, dass das Bewegen eines Sektionknotens alle Untersektionen darin mit einbezieht, weil sie es gewohnt waren, „flach“ zu denken wie in MS Word oder PowerPoint.

2.3 Bearbeitungsmöglichkeiten

Allgemein richten wir uns nach den nativen Eigenschaften des OOo Writer für das Bearbeiten von Dokumenten. Zusätzlich entwickelten wir eine <ML>³-Toolbar, welche spezifische Funktionalitäten hinsichtlich der Beschreibungssprache bietet. Die Toolbar wurde in BASIC implementiert, wozu die integrierte Entwicklungsumgebung (IDE) von OOo genutzt wurde. Die Toolbar beinhaltet Mechanismen, um zwischen verschiedenen Detail-Ebenen bezüglich der (normalerweise versteckten) Kontrollstrukturen zu wechseln, welche für die Reproduzierbarkeit von gültigem <ML>³ enthalten sind.

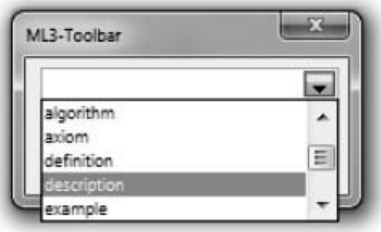

Abbildung 4: Die Toolbar für spezielle Authoring-Funktionen bezüglich <ML>³

Es gibt auch eine kontext-sensitive Eingabehilfe, die eine Liste von gültigen Elementen zu jeder gegebenen Position im Dokument bereitstellt. Dieser wird in Abbildung 4 dargestellt. Der Mechanismus stellt sicher, dass nur LOs produziert werden, die auch wieder in <ML>³ abgebildet werden können. Attribute werden entsprechend ihrer Visualisierung bearbeitet – entweder in der Dokumentansicht oder über Dialoge.

Diese <ML>³-spezifischen Funktionalitäten werden durch die entwickelte Dokumenten-vorlage bereit gestellt. Der Anwender muss keine Plug-Ins installieren oder einrichten. Die Vorlage wird automatisch geöffnet, wenn eine <ML>³-Datei geöffnet wird bzw. der Autor öffnet die Vorlage, wenn er ein neues Dokument erstellen will.

2.4 Exportmechanismen

Ähnlich zum Import wird der Export vom OOo Writer nach <ML>³ über eine XSL-Transformation realisiert. Wieder musste dem Problem der aus mehreren Dateien bestehenden LO begegnet werden, da die „Speichern Unter"-Funktion nur eine einzelne Datei erstellt. Die ursprüngliche Dateistruktur wird daher in zwei Schritten wiederhergestellt. Zunächst wird ein Zwischenformat erzeugt, welches die OOo-internen Datenstrukturen auf das Wesentliche reduziert. Danach wird diese temporäre Beschreibung nach dem <ML>³-Schema überprüft und unter Beachtung der Dateigrenz-Markierungen, die beim Import entstanden sind, in mehrere Dateien aufgeteilt. Daher wird diese Export-Routine als ein Makro aufgerufen (und nicht als Standard-Menübefehl).

Natürlich werden alle <ML>³-Elemente, die im OOo Writer versteckt waren, wie in der Ursprungsversion wiederhergestellt.

2.5 Verwaltungsaufgaben

Um den OOo Writer für die Bearbeitung von <ML>³-Dateien vorzubereiten, muss der Nutzer einmalig Konfigurationen vornehmen, um die XML-Filtereinstellungen bzw. den Dokumenttyp, die Import/Export-Filter und die Vorlage (mit den erstellten Makros und der Toolbar) auszuwählen.

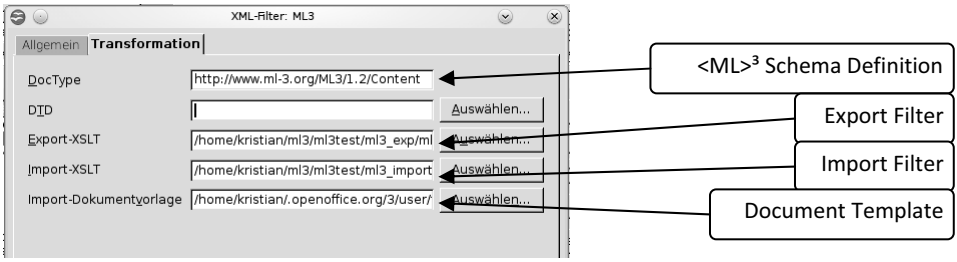

Abbildung 5: Spezifikation der XML-Filtereinstellungen für die Bearbeitung von <ML>³-Dateien im OOo Writer

Dieser Vorgang ist einmalig im Vorfeld auszuführen und nutzt den XML-Filter-Dialog vom Extras-Menü (wie in Abbildung 5 dargestellt). Wenn ein XML-Dokument mit der entsprechenden Schema-Definition geöffnet wird, werden die Einstellungen automatisch angewendet.

Wie bereits angemerkt muss kein Plug-In installiert werden. Die Konfiguration nach der obigen Anleitung ist der einzige Schritt, den ein Nutzer ausführen muss, um <ML>³ mit dem OOo Writer zu bearbeiten.

3 Erfahrungen und allgemeine Empfehlungen

Während unser primärer Fokus auf der Erweiterung des OOo Writer für die Beschreibungssprache <ML>³ lag, trafen wir auf Probleme, die über die Implementierung hinausgehen. Zunächst seien hier unsere Rückschlüsse bezüglich der Nutzung des OOo Writer für Single Source Authoring aufgeführt:

- Wie erwartet ist der OOo Writer ein mächtiges und ausgereiftes Werkzeug für XML-basiertes Authoring. Seine Architektur und Schnittstellen sind besonders geeignet für eigene Erweiterungen.
  → Wir können die Nutzung des OOo Writer für Single Source Authoring uneingeschränkt empfehlen.

- Die Integration von Formatanweisen mit Makros und einer Toolbar in eine Dokumentenvorlage reduziert den administrativen Aufwand für den Autor.
  → Wir befürworten die Definition einer Dokumentenvorlage mit Makros und/oder Toolbars im Vergleich zur Implementierung einer Reihe von Plug-Ins. Die Vorlage lässt sich für die Erstellung neuer LOs nutzen, während die Toolbar auch beim Bearbeiten existierender Dateien verfügbar ist.

- Die interne Komplexität des Programms ist hoch. Es gibt nur wenig Hilfestellung für das Schreiben von Makros. Oftmals werden Elemente oder Attribute im Fehlerfall einfach ohne eine Mitteilung übergangen, was das Debugging zu einer schwierigen Aufgabe macht.
  → Abgesehen von der offiziellen Dokumentation, der IDE und Hilfstools ist vor allem eine verständliche Sammlung ausführlicher Beispiele sehr hilfreich.

- OOo Writer kann auch für die spätere Erstellung von PDF-Versionen über die bereits integrierte Funktion genutzt werden, da der reguläre Weg einer Transformation von XML zu PDF unter Verwendung eines spezifischen Formatting Objects Processors (FOP) schwierig zu implementieren ist und nicht immer ansprechende Ergebnisse produziert. Jedoch kann die Darstellung in einem LO-Werkzeug für Lehrer aufgrund zusätzlicher Informationen oder Musterlösungen abweichen von der Version, wie es von Studenten bei der Kursarbeit betrachtet werden soll.
  → Ein von OOo erstelltes PDF kann den Betrachtungsprozess der Lerninhalte unterstützen, sollte jedoch vordergründig an Lerner gerichtet sein.

Darüber hinaus lernten wir mehr über <ML>³ als Beschreibungssprache, obwohl diese schon fast 7 Jahre alt ist:

- Die <ML>³-Spezifikation ist sehr komplex. Mehrere Optionen sind für unerfahrene Autoren schwierig zu begreifen und ließen sich über ein Makro zum besseren Verständnis nur unzureichend veranschaulichen. Besonders das Anlegen eines neuen LOs erfordert Einarbeitung in geeignete Konzepte und Strukturen. Dies erhöht die Wahrscheinlichkeit des Scheiterns beim Export nach <ML>³.

→ Eine ausgereifte Dokumentenvorlage sollte wesentliche Strukturelemente sowie Platzhalter für die erforderlichen Informationen über ein LO beinhalten.

- Die geforderte Trennung von Inhalt und Layout (verbunden mit dem Single Source Authoring) ist ausschlaggebend für die Verwendung verschiedener Bearbeitungswerkzeuge, da der Import-Filter und Renderer sich nicht um die Formatierungs-Spezifikationen kümmern muss, die ein Autor eingestellt haben mag. So können Inhalte über verschiedene Plattformgrenzen hinweg ohne Nachteile bearbeitet werden.
  → Eine Beschreibungssprache für wiederverwendbare LOs darf keine Formatierungsmechanismen enthalten, unabhängig wie sehr Autoren auf das Festlegen fetter/kursiver Buchstaben oder linksbündiger/zentrierter/ rechtsbündiger Abbildungen bestehen. Solche Belange sind komplett der entsprechenden Stilvorgaben zu überlassen.

- Flexibles Verschachteln von Abschnitten erzeugte keine Probleme mit Rekursionen, jedoch mit Auslastung und Layout. Wir versuchten, Abschnitte und Unterabschnitte über Tabs zu veranschaulichen, was jedoch aufgrund ungeahnter Verschachtelungstiefen nicht brauchbar war. Somit wurde eine sequentielle Auflistung der Abschnitte mit Zeileneinzug und Nummerierung zur einzig effizienten Lösung. Dies führte je nach Anzahl der einzelnen XML-Dateien des LOs zu verlängerten Ladezeiten und hohen Speicheranforderungen.
  → Eine explizite Restriktion zu vordefinierten Abschnittsebenen innerhalb der Beschreibungssprache vereinfacht die Implementierung von Werkzeugen.

Schließlich konnten noch Erkenntnisse zur allgemeinen Organisation des Authoring-Prozesses gewonnen werden:

- Wir stellten fest, dass Autoren nicht über die unterliegende Baumstruktur ihrer LOs Bescheid wussten. Wenn möglich blendeten sie XML-Tags aus.
  → Eine Veranschaulichung des Dokumenten-Baums muss nicht der Schlüssel einer Interaktion sein. Stattdessen ist auch eine sequentielle Ansicht mit versteckten (und geschützten) Strukturinformationen akzeptabel.

- Generell verursachen große Dokumente oft Probleme bezüglich Performance und Robustheit. Ebenso wird kooperative Inhaltserstellung behindert, wenn keine parallelen Bearbeitungen möglich sind. Außerdem stellt die Dateiverwaltung eines LOs eine komplexe Aufgabe für die Implementierung mit dem OOo Writer dar.
  → Es sollte klare Regeln in der Dateiverwaltung geben, z. B. immer eine Datei pro LO. Auch die Größe eines LOs sollte begrenzt werden. Dies kann jedoch nicht in der Beschreibungssprache festgelegt werden, sondern erfordert separate Anweisungen an die Autoren.

- Immer wieder steht die automatische Aufteilung der Abschnitte eines LOs auf mehrere Folien oder Bildschirm-Seiten zur Diskussion. Bis heute fanden wir keinen Mechanismus oder Werkzeug, das diese Aufgabe zufriedenstellend löst.

→ Für die Menge an Inhalt pro Präsentationseinheit (Bildschirm, Folie) bleibt der Autor verantwortlich.

Obwohl mit viel Sorgfalt zusammengestellt erheben wir mit dieser Auflistung nicht den Anspruch auf Vollständigkeit. Wir werden unsere Arbeit an diesen Punkten fortsetzen, damit weitere Erfahrungswerte für die Zukunft gesammelt werden.

4 Zusammenfassung und Ausblick

Basierend auf einen Überblick über die Vielfalt der Ansätze für Single Source Authoring präsentiert dieser Artikel unser Konzept und die Implementation eines <ML>³-Plug-Ins für den OOo Writer sowie allgemeine Erfahrungen, die wir bei der Entwicklung des Prototyps sammeln konnten, und daraus resultierende Empfehlungen. Wir gruppierten diese Erkenntnisse in drei Bereiche hinsichtlich des Werkzeugs selbst, der unterliegenden Beschreibungssprache und der Verwaltung des gesamten Authoring-Prozesses. Der OOo Writer eignet sich trotz komplexer Entwicklungsarbeiten als Werkzeug für das Single Source Authoring, wobei Dokumentenvorlagen mit Makros gegenüber Plugins befürwortet werden. So lässt sich zudem der Einarbeitungsaufwand für unerfahrene Autoren durch vordefinierte Beispiele und Strukturelemente reduzieren.

Wir gehen davon aus, dass diese Ergebnisse leicht auf andere Ansätze des Single Source Authorings übertragen werden können (unabhängig, ob vom Bildungsbereich ausgehend oder nicht, unabhängig, ob mit OOo Writer oder nicht) und somit für zahlreiche Forscher und Entwickler in diesem Bereich von Vorteil sind.

Wir werden unsere Implementation weiter verfolgen, um das komplette Spektrum der <ML>³-Eigenschaften abzudecken, besonders hinsichtlich Mathematik-Formeln und interaktiven Elementen. Einige Dialoge für besondere Bearbeitungsaspekte sind aktueller Gegenstand unserer Arbeit, wie das Definieren von Attributen, welche ein Element nach dem dreidimensionalen Modell von <ML>³ für verschiedene Zielgruppen, Schwierigkeitsgrade und Ausgabemedien klassifizieren. Außerdem denken wir auch über die Verwendung der Referenzierungsmechanismen des OOo Writer nach, um Elemente innerhalb und sogar außerhalb eines LOs zu verlinken. Dies ist von Interesse, da das <ML>³-Konzept nicht nur Inhalte vom Layout trennt, sondern auch didaktische Aspekte eines LO in unabhängigen und austauschbaren Dateien. Anwendung durch Autoren wird das entwickelte System im Rahmen eines Verbundprojektes mit fünf norddeutschen Universitäten im Bereich der Lerninhaltserstellung finden. Sicherlich werden wir dabei weitere Erfahrungen zur Verbesserung des Systems sammeln können.

Darüber hinaus verfolgen wir Ansätze, die <ML>³-Autorenwerkzeuge um eine web-basierte Lösung zu erweitern. Wir stellen uns dabei ein Wiki zum kollaborativen Bearbeiten vor, da es einen dezentralisierten Ansatz unterstützt, welcher aus Architektursicht vielversprechend hinsichtlich Robustheit und Fehlertoleranz ist. Leistungsfähige Plattformen wie Semantic Mediawiki sollten dafür aufgrund ihrer eingebauten Attributierungs- und Referenzierungs-Mechanismen eine gute Basis für das Abbilden von <ML>³-Konzepten sein.

Danksagung

Diese Arbeit erfolgte mit Unterstützung des Ministeriums für Bildung, Wissenschaft und Kultur des Landes Mecklenburg-Vorpommern im Rahmen des Verbundprojektes „Technische Informatik Online“. Die Autoren bedanken sich bei den Studenten Tobias Zimmer und Kristian Schultz für ihre Implementierung.

Literaturverzeichnis

[BO04] Brehm, J., Ossipova, N.: <ML>³ Authoring mit FrameMaker. Structured eLearning 2004. University of Rostock, S. 83-92

[BM02] Brusilovsky, P., Maybury, M. T.: From adaptive hypermedia to adaptive Web. Communications of the ACM, 45(5), 2002, S. 31-33

[Da05] Dagger, D., Wade, V., & Conlan, O.: Personalisation for All: Making Adaptive Course Composition Easy, 2005, Educational Technology & Society, 8 (3), S. 9-25

[FB06] Fisler, J., Bleisch, S.: eLML, the eLesson Markup Language: Developing sustainable e-Learning Content Using an Open Source XML Framework. Web Information Systems and Technologies (WEBIST), 2006

[Fr02] Freitag, B.: LMML – Eine Sprachfamilie für eLearning Content. Informatik 2002. Köllen Verlag, S. 349-353

[Fr04] Friesen, N.: Three Objections to Learning Objects. Online Education Using Learning Objects. RoutledgeFalmer, London, 2004. S. 59-70

[GH04] Gecks, T., Henrich, D. (2004). Von MS Word zu <ML>³ - Ein Konvertierungswerkzeug. Structured eLearning 2004. University of Rostock, S. 93-102

[GV08] Gieseking, M., Vornberger, O.: media2mult - A Wiki-based authoring tool for collaborative development of multimedia documents, International Conference on eLearning 2008, IADIS. S. 295-303

[Gr09] Gries, V., Lucke, U., & Tavangarian, D.: Werkzeuge zur Spezialisierung von XML-Sprachen für die vereinfachte, didaktisch unterstützte Estellung von eLearning-Inhalten. Die 7. e-Learning Fachtagung Informatik (DeLFI) 2009. Köllen Verlag, Germany. S. 211-222

[Ha93] Hardman, L., van Rossum, G. & Bulterman, D.: Structured multimedia authoring. Multimedia 1993. ACM, New York, USA.S. 283-289

[KM04] Koper, R., Manderveld, J.: Educational modelling language: modelling reusable, interoperable, rich and personalised units of learning. British Journal of Educational Technology, 35(5), 2004, S. 537-551

[Lu03] Lucke, U., Tavangarian, D., Voigt, D.: Multidimensional Educational Multimedia with <ML>³, E-Learn 2003. AACE Charlottesville, VA. S. 101-104

[Lu06] Lucke, U.: An Algebra for Multidimensional Documents as Abstraction Mechanism for Cross Media Publishing. Automated Production of Cross Media Content for Multi-channel Distribution (Axmedis) 2006. IEEE, Los Alamitos, CA, USA. S. 165-172

[LM10] Lucke, U., Martens, A.: Utilization of Semantic Networks for Education: On the Enhancement of Existing Learning Objects with Topic Maps in <ML>³. Informatik 2010. Köllen Verlag, Germany. S. 91-96

[Ri03] Rinke, S.: DocBook Publishing. Online publishing made easy. online at http://www.stefan-rinke.de/articles/

[Ro04] Robson, R. Context and the Role of Standards in Increasing the Value of Learning Objects. Online Education Using Learning Objects. Routledge/Falmer, London. 2004, S. 159–167