Vorlesungsskripte im E-Learning-Zeitalter

Torsten Klie, Uwe Frommann
Forschungszentrum L3S Hannover

Christian Werner, Stefan Fischer
Institut für Telematik
Universität zu Lübeck

Astrid Weilert, Christoph Klinzmann, Dietmar Hosser
Institut für Baustoffe, Massivbau und Brandschutz
Technische Universität Braunschweig

Jan Zimmermann, Manfred Krafczyk
Institut für Computeranwendungen im Bauingenieurwesen
Technische Universität Braunschweig

Zusammenfassung

In der universitären Lehre dominieren Lernmaterialien, die hauptsächlich aus Text, Formeln, Tabellen und graphischen Illustrationen bestehen. Die Hersteller von E-Learning-Werkzeugen tragen dieser Tatsache jedoch nicht ausreichend Rechnung und bieten fast ausschließlich Autorensoftware an, die auf die Erstellung von audiovisuellen Inhalten ausgerichtet ist. Obwohl sich die Integration von solchen textbasierten Inhalten in Lernplattformen und andere E-Learning-Systeme anbietet, fehlen bislang tragfähige Konzepte zur technischen Umsetzung. Zum einen müssen diese fachspezifische Anforderungen berücksichtigen, wie etwa ausreichende Unterstützung für mathematische oder chemische Formeln, und zum anderen sollen die erstellten Inhalte auch wiederverwendbar und möglichst plattformunabhängig sein. Im Rahmen dieses Beitrags diskutieren die Autoren zunächst mögliche Lösungsansätze für diese Problemstellung, im Weiteren stellen sie geeignete Werkzeuge für die Inhaltserstellung vor und präsentieren schließlich ihre Erfahrungen mit XML-basierter Inhaltserstellung im universitären Praxiseinsatz.

1 Einleitung

Obwohl geeigneten Datenformaten zur Beschreibung von Lerninhalten offenbar eine fundamentale Bedeutung für die erfolgreiche Umsetzung von E-Learning-Konzepten beikommt, beschränkten sich Forschungsarbeiten in diesem Bereich lange Zeit auf die Entwicklung von neuen Metadatenstandards.

Besonders bekannte Beispiele sind hier der SCORM-Standard von Advanced Distance Learning (ADL) [1] sowie die verschiedenen Spezifikationen, die vom IMS Global Learning Consortium [5] veröffentlicht wurden. Sämtliche Funktionalitäten, die über die Metadatenbeschreibung hinausgehen, zielen dabei auf die Interoperabilität von technischen Vorgängen ab. Beispielsweise wird in der „IMS General Web Services“ Spezifikation zwar genau festfelegt, wie Lernsystene mit Hilfe von plattformunabhängigen Protokollen Lernmaterialien austauschen können, konkrete Datenformate für die Beschreibung von Lernmaterialien werden aber nicht angegeben.

Zunächst ging man nämlich davon aus, dass mit bereits vorhandenen Technologien wie HTML-Seiten, Postscript- oder PDF-Dateien sowie mit proprietären Datenformaten wie Microsoft Word oder Macromedia Flash bereits ausreichende Möglichkeiten für die Erstellung und Nutzung von E-Learning-Inhalten bestünden.

In der Praxis hat sich diese Annahme jedoch häufig nicht bestätigt: Anders als bei der Erstellung von Web-Seiten oder Dokumenten, die ausschließlich für den Druck bestimmt sind, beinhaltet die Erstellung von E-Learning-Anwendungen eine ganze Reihe von wichtigen Faktoren, deren jeweilige Gewichtung nach Einsatzbereich variiert:

Aktualisierbarkeit: Die Lebensdauer einer Kurseinheit in der universitären Lehre ist oftmals sehr kurz, typischerweise beträgt sie ein bis zwei Semester. Anschließend müssen die Materialien aktualisiert werden.

Modularisierung und Unterstützung für mehrere Autoren: Weiterhin sollte das Datenformat modular aufgebaut sein. Bei umfangreicheren Dokumenten ist es äußerst wichtig, dass mehrere Autoren gleichzeitig an einzelnen Modulen arbeiten können, ohne dass es zu Schwierigkeiten bei Querverweisen oder der Nummerierung von Abschnitten kommt.

Mächtigkeit und Erweiterbarkeit: Im wissenschaftlichen Bereich stellt jede Fachrichtung andere Anforderungen an ein E-Learning-Datenformat. Im Bereich der Naturwissenschaften ist etwa die Darstellung von Formeln ein wesentliches Merkmal, in den Ingenieurswissenschaften sind es auch technische Zeichnungen. Ein Datenformat zu schaffen, das all diesen Anforderungen genügt, ist äußerst schwierig. Dennoch ist die Überbrückung dieser Heterogenität häufig erforderlich – z.B. weil eine Universität ein zentrales Lern-Management-System betreibt, auf dem alle Fachbereiche ihre Inhalte einstellen sollen. Ein möglicher Lösungsweg besteht hier in der Implementierung von Erweiterungsmechanismen. Das Datenformat ist also nicht statisch, sondern kann angepasst werden, wenn sich einzelne Anforderungen ändern oder neue hinzukommen.

Benutzerfreundlichkeit: Ein vielfach unterschätzter Aspekt ist die Qualität der

Benutzerschnittstelle – sowohl aus Sicht der Lernenden als auch aus Sicht der Inhaltsersteller. Die Wahrnehmung aus Lernersicht hängt maßgeblich von der Qualität der erstellten Inhalte ab. Das verwendete Datenformat wirkt sich hier nur indirekt aus, etwa durch die Beschränkung möglicher Inhalte wie Formeln, Grafiken oder multimedialer Elemente (s.o.). Viel direkter ist der Einfluss auf die Benutzerschnittstelle für Inhaltsersteller. Je nach dem, welches Datenformat verwendet wird, wird er mit anderen Werkzeugen zur Erzeugung seiner Inhalte konfrontiert.

All diese Faktoren müssen bei der Wahl eines E-Learning-Datenformats berücksichtigt werden. Die Frage nach einem optimalen Datenformat ist in der E-Learning-Forschung nicht neu und wurde bereits in etlichen Publikationen diskutiert [4, 2, 8, 9].

Ein häufig gewählter Lösungsansatz besteht in der Inhaltsbeschreibung mittels XML. Häufig wird XML jedoch als Allheilmittel für sämtliche Kompatibilitätsprobleme gesehen und die damit verbundenen Nachteile übersehen oder ignoriert. In diesem Beitrag wollen wir die in den Projekten PORTIKO und ELAN gesammelten Erfahrungen präsentieren und neue Lösungsansätze für XML-basierte Inhaltserstellung vorstellen [7]. Ausgangspunkt ist dabei die Frage, wie sich textbasierte Inhalte so in aktuelle E-Learning-Plattformen integrieren lassen, dass ein Mehrwert für Autoren und Lernende entsteht.

Im Abschnitt 2 hinterfragen wir zunächst den Sinn einer Integration von Textinhalten in E-Learning-Umgebungen und entwickeln dann ein Konzept für die technische Umsetzung. In Abschnitt 3 geben wir einen Überblick zu geeigneten Werkzeugen für die Inhaltserstellung. Die Stärken und Schwächen unseres Ansatzes werden in Abschnitt 4 analysiert und bewertet. Schließlich fassen wir unsere Ergebnisse in Abschnitt 5 zusammen und geben einen Ausblick auf weitere Arbeiten.

2 Vorlesungsskripte mit PORTIKO

2.1 Sinn und Machbarkeit

Zunächst stellt sich die Frage, ob Materialien vom Typus „Vorlesungsskript“ überhaupt noch in die durch Interaktivität und Multimedialität geprägte E-Learning-Welt passen. Macht es Sinn, das Konzept des Vorlesungsskripts in die E-Learning-Welt hineinzutragen oder sind längere zusammenhängende Text hier einfach nicht angemessen?

Würde man diese Frage verneinen, so wäre man sicherlich nicht in der Lage, eine zu herkömmlicher Lehre vergleichbare Menge an Informationen bereitzustellen. Das geschriebene Wort kann durch den Einsatz mutimedialer Elemente sicherlich ergänzt werden, es ist aber nicht dadurch zu ersetzen.

Es gibt viele Beispiele für den erfolgreichen Einsatz von textbasierten Materialien in Verbindung mit dem Medium Computer. Zu nennen wären hier beispielsweise e-Books oder auch Wikipedia. Längst haben textbasierte

Materialien auch in die E-Learning-Welt Einzug gehalten. Das „Online-Stellen“ von Text-Dateien im PDF- oder Postscript-Format gehört sicherlich zu den Vorformen von E-Learning im heutigen Sinne. Auch die Entwicklung von Hypertextdokumenten in HTML wird im E-Learning-Bereich häufig praktiziert. Immer bessere HTML-Editoren ermöglichten es auch ungeübten Anwendern, optisch ansprechende HTML-Seiten zu programmieren. Leider lassen sich HTML-Dokumente nicht besonders gut weiterverarbeiten und auch der Ausdruck gestaltet sich mitunter als schwierig.

Es ist erstaunlich und enttäuschend zugleich, dass die E-Learning-Gemeinde bis heute noch keine praxistauglichen Ausprägungen für textbasierte Inhalte gefunden hat, die über diese statischen Formen hinausgehen und die Texte sinnvoll in moderne E-Learning-Plattformen integriert. Ziel muss es sein, diese Integration technisch so zu gestalten, dass die Vorteile des Mediums Computer zum Tragen kommen und so einen Mehrwert für Autoren und Lernende entsteht.

2.2 Datenformat

Ausgehend von dieser Problemstellung haben wir im Rahmen des Projekts PORTIKO in Zusammenarbeit mit der TU Dresden einen technischen Lösungsansatz entwickelt und in den vergangenen drei Jahren praktisch umgesetzt. Ziel war es, ein Datenformat zu schaffen, das einfach genug ist, um von Inhaltserstellern mit rudimentären Computerkenntnissen verstanden zu werden. Bestehende Formate, wie z.B. DocBook [14] konnten nicht genutzt werden, weil sie trotz sehr großem Sprachumfang die Anforderungen nicht vollständig erfüllten.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<?xml-stylesheet type="text/xsl" href="style/contentobject.xsl"?>
<!DOCTYPE contentobject PUBLIC "-//Portiko//DTD Content v1.1
    20021018//EN" "../etc/dtd/content.dtd">
  <contentobject>
    <authors>
    <author id="ckli">
      <vcard xmlns="http://www.portiko.de/content/vcard">
        <fn>Christoph Klinzmann</fn>
        <email email.type="PREF">c.klinzmann@ibmb.tu-bs.de</email>
        <org><orgname>XML-AG, TU Braunschweig</orgname></org>
      </vcard>
    </author>
  </authors>
  <title>Beispieldokument Portiko DTD</title>
  <abstract .../>
  <chapter id="einleitung">
    <title>Beschreibung</title>
    <body>
      <sect1 id="abschnitt1">
        <para> Hier beginnt der Text weitere Tags sind möglich.
        <b> Dieser Teil z.B. ist fett </b>.</para>
      </sect1>
    </body>
```

```
  </chapter>
  <appendix .../>
</contentobject>
```

Abb. 1. Beispiel für das Grundgerüst eines PORTIKO-Dokuments

Als technische Basis bot sich hier die Datenbeschreibungssprache XML an, da man mit ihr (anders als mit HTML) Inhalte unabhängig von der späteren Darstellung beschreiben kann. Ein weiterer Vorteil von XML besteht darin, dass über eine XML-Grammatik (z.B. in Form einer DTD oder eines XML Schemas) genaue Vorgaben gemachen werden können, welche Struktur der zu erstellende Inhalt haben soll.

Die PORTIKO-DTD definiert eine einfache XML-Struktur, welche sich an den typischen syntaktischen Elementen aus Lehrbüchern bzw. Skripten orientiert (Kapitel, Abschnitt, Absatz, etc.). Anhand dieser Vorgabe können die erstellten Dokumente auch automatisch auf ihre Gültigkeit hin überprüft werden, d.h. es wird untersucht, ob sie die von der DTD bestimmte Struktur besitzen. Abb.1. zeigt die grundlegenden Elemente der PORTIKO-DTD anhand eines einfachen Beispiels.

2.3 Erzeugung des Zielformats

XML-Dokumente lassen sich mit Hilfe von XSL-Transformationen (XSLT) [16] in andere Formate überführen. Dabei kann es sich bei der Ausgabe wiederum um XML handeln, aber auch Seitenbeschreibungssprachen wie PDF, Postscript oder (X)HTML sind möglich.

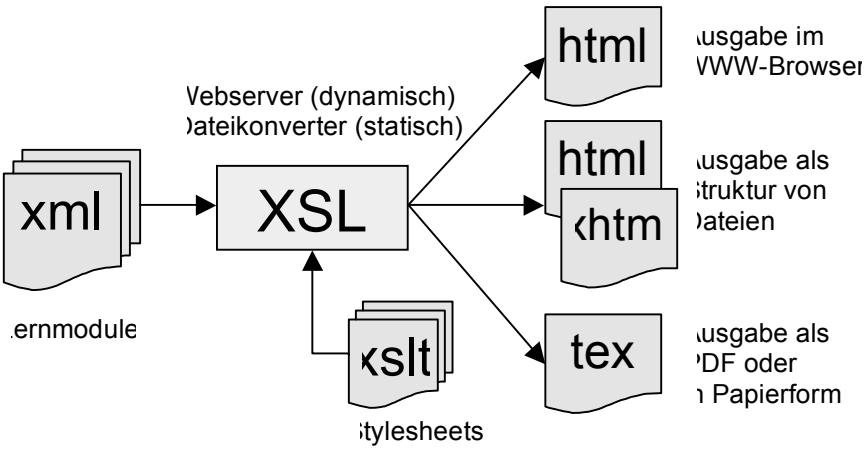

Abb. 2. Ein- und Ausgaben für den Transformationsprozess

Eine solche Transformation läuft wie folgt ab: Zunächst liest der XSLT-Prozessor ein XML-Dokument und ein XSLT-Stylesheet ein. In diesem Stylesheet

findet der Prozessor sog. Templates. Dies sind Formatvorlagen für die einzelnen Textelemente wie z.B. Überschriften oder Textblöcke. Darüber hinaus können weitere Sylesheets oder XML-Dokumente eingebunden werden. Dadurch können Erweiterungen eingebunden werden, ohne die grundlegende Formatierung verändern zu müssen. Der XSLT-Prozessor wendet die Templates dann auf die XML-Daten an und erzeugt die Ausgabedatei (siehe Abb. 2).

Im Rahmen der Projektarbeiten wurden vier verschiedene Stylesheets implementiert. Die Dokumente können zur Zeit entweder in einen Satz von HTML- oder XHTML-Seiten überführt werden, so dass sie von den Lernern per Web-Browser durchsucht und am Bildschirm gelesen werden können oder aber in ein LaTeX-Dokument, das als Vorstufe zum Druck dient. Die Erzeugung von HTML-Seiten kann dabei auch direkt auf dem Web-Server passieren, so dass der Transformationsprozess bei jedem Seitenaufruf ausgelöst wird.

Für die Inhaltsersteller ergibt sich hierdurch schon ein deutlicher Mehrwert, denn sie müssen nur noch die XML-Dokumente in einer einzigen Version pflegen und können daraus automatisch mehrere Ausgabeformate erzeugen.

Die Erzeugung der druckbaren Fassung funktioniert bei unserer Implementierung über einen Umweg: Es gibt im wesentlichen zwei Ansätze um XML-Daten in PDF zu überführen. XSL bietet mit der Erweiterung XSL:FO zunächst die Möglichkeit zur direkten Erzeugung von PDF-Dateien, d.h. der XSLT-Prozessor gibt direkt ein PDF-Dokument aus. Nachteile sind hierbei, dass der XSLT-Prozessor sämtliche Textsatzregeln, wie beispielsweise Silbentrennung, beherrschen muss und dass die XSLT:FO-Stylesheets sehr umfangreich und komplex werden. Wir haben uns daher für die zweite Variante entschieden: Wir transformieren die Portiko-Dokumente zunächst in das Zwischenformat LaTeX, das dann mit dem Programm pdflatex in ein PDF-Dokument übersetzt wird.

Das Ergebnis einer solchen Transformation ist unter *http://www.cab.bau.tu-bs.de/elan/kurzanleitung.pdf* zu sehen. Das selbe Dokument ist auch in HTML verfügbar: *http://www.cab.bau.tu-bs.de/elan/kurzanleitung/*.

Ein besonderes Augenmerk liegt jeweils auf Formeln in den XML-Dokumenten. Gerade im Bereich der universitären Lehre kommen diese sehr häufig in Lernmodulen und Skripten vor.

In den XML-Dokumenten werden Formeln mit MathML [10] beschrieben. Die direkte Einbettung von MathML in HTML ist nicht möglich, da von den aktuellen Browsern lediglich Firefox [3] die MathML-Passagen ohne Fehler abbilden kann. Um dieses Problem zu umgehen, wurde auf ein Java-Applet der Firma DesignScience [13] zurückgegriffen. Dieses Applet verwendet den vorhandenen MathML Code und stellt die Formeln lesbar dar. Der entsprechende Aufruf wird bei der Transformation automatisch in die HTML-Dateien eingefügt.

Für die Nutzer des Firefox-Browsers wurde die Konvertierung in XHTML implementiert, da in diesem Format Formeln vom Browser direkt dargestellt werden können. Zur Betrachtung ist keine weitere Software notwendig. Die Vorteile der Verwendung von XHTML liegen daher in der deutlich höheren Geschwindigkeit der Anzeige, da nicht für jede einzelne Funktion ein Java-Applet gestartet werden muss. Des weiteren sind die Funktionen zusammen mit den Textinhalten, ohne Qualitätseinbußen beliebig in der Größe skalierbar. Ein

Beispiel dazu befindet sich unter *http://www.cab.bau.tu-bs.de/elan/mathml.htm* .

Bei der Transformation nach LaTeX werden die MathML-Formeln in LaTeX-Formeln umgewandelt, was weitestgehend problemlos funktioniert.

Weiterhin unterstützt unsere Implementierung durchgängig das vollständig automatische Erzeugen von Verzeichnissen: Abbildungsverzeichnis, Formelverzeichnis, Tabellenverzeichnis, Literaturverzeichnis, Stichwortverzeichnis, Autorenverzeichnis und Glossar. Lediglich die Indexwörter (also Autorennamen und Stichwörter) und die Erklärungen für die Glossareinträge müssen vom Inhaltsersteller im XML-Dokument manuell eingegeben werden.

3 Inhaltserstellung aus Nutzersicht

Obwohl die Struktur des gewählten XML-Formats sehr einfach ist, sind sicherlich nur die wenigsten Anwender bereit, XML-Dokumente mit einem Texteditor zu erstellen. Die Frage nach einem einfach zu bedienenden Werkzeug hatte daher für uns eine zentrale Bedeutung. Im Gegensatz zu Dokumenten in proprietären Binärformaten wie z.B. MS-Word, können XML-Dateien mit jedem beliebigen Editor erzeugt und bearbeitet werden. Es sind jedoch auch eine ganze Reihe spezifischer XML-Editoren verfügbar, die insbesondere die Arbeit mit umfangreichen Dokumenten erheblich erleichtern und für technisch weniger versierte Anwender deutliche Vorteile bieten. Die Ergebnisse unserer Untersuchung verschiedener Werkzeuge sind im Folgenden dargestellt.

3.1 XMLwriter

Der XMLwriter [15] ist ein leicht zu bedienender XML-Editor. Er hilft dem Autor durch farbige Hervorhebung (syntax highlighting) zwischen Inhalt und Objektbeschreibung zu unterscheiden. Das Werkzeug generiert aus einer eingebundenen DTD die so genannte TagBar, in der alle definierten Beschreibungsobjekte mit sämtlichen zugehörigen Attributen aufgelistet sind und per Drag und Drop dem Dokument hinzugefügt werden können. Gleichzeitig wird der Nutzer durch automatische Hilfen bei der Arbeit unterstützt. So werden geöffnete Tags sofort mit einem Endtag abgeschlossen und automatisch hinzugefügte Zeilenumbrüche und Einzüge helfen, die Inhalte übersichtlich zu gestalten. Umfangreiche Dokumente können in Form von Projekten organisiert werden, wodurch die Überschaubarkeit und damit der schnelle Zugriff auf einzelne Abschnitte erleichtert wird. Als weitere Funktion können eigene Kommandos definiert werden, die dann unter einem Knopf erreichbar sind. Beispielsweise kann man so den Start einer XSL-Transformation mit bestimmten Argumenten und Parametern in einem einzigen Mausklick bündeln. Zeitgleich mit der Transformation werden auftretende Fehlermeldungen ausgegeben, die durch Angabe der Zeilennummer, in der der Fehler auftritt, die Suche erleichtern und beschleunigen.

Diese Form der Erstellung von Inhalten ähnelt der Arbeit mit Latex-Dokumenten oder einer Entwicklungsumgebung für Software. Das fertige Ergebnis wird erst nach dem Übersetzen / Konvertieren sichtbar. Für Nutzer, die über Erfahrungen im Programmieren oder der Arbeit mit Latex verfügen, ist daher der Einarbeitungsaufwand gering und sie können schnell zu einer effizienten Arbeitsweise gelangen.

3.2 WYSIWYG mit Textverarbeitungsprogrammen

Versuche, die PORTIKO-DTD in die gängigen Textverarbeitungsprogramme einzubinden, um diese dann als WYSIWYG-XML-Editor verwenden zu können, führten sowohl mit Microsoft Word [11] als auch mit OpenOffice Writer [12] zu keinem akzeptablen Ergebnis. Dies mag zunächst überraschen, da doch beide Programme in ihrer neusten Version (MS Word 2003 und OpenOffice 2.0 Beta) XML-Unterstützung bieten.

Zum einen ist die Entwicklung von separaten XSLT-Stylesheets und Makros für die WYSIWYG-Funktionalität notwendig, was sich als sehr aufwändig herausgestellt hat. Zum anderen vermuten wir jedoch einen noch größereren Nachteil dieses Ansatzes darin, dass viele Anwender versuchen würden, ihre Gewohnheiten beim Umgang mit der Textverarbeitung auf die Erstellung von XML-Dokumenten zu übertragen. Die Arbeit mit XML-Dokumenten erfordert es jedoch, sich auf die vorhandenen Formatierungselemente zu beschränken. Daher ist es äußerst schwierig zu vermitteln, warum Funktionalitäten, wie das Ändern der Schriftart für einzelne Wörter, nicht verfügbar sind, obwohl die Textverarbeitung dies normalerweise unterstützt.

3.3 XDoc

Im Rahmen einer studentischen Arbeit haben wir den Open Source Editor XDoc [6] entwickeln lassen. Die Grundidee besteht bei diesem Editor darin, dass der Anwender, bevor er mit der Inhaltserstellung beginnt, zunächst eine Grammatik für die zu erstellenden Inhalte in Form einer DTD oder XML-Schema-Datei angibt. Während der Bearbeitung überprüft der Editor permanent, ob die Grammatik eingehalten wird. Falls Teile des Dokuments dagegen verstoßen, markiert XDoc diese Dokumentteile rot und gibt einen Hinweistext aus (siehe Abb.3).

Will der Anwender ein neues Dokument erstellen, so kann XDoc ein minimales XML-Dokument der gewünschten Sprache automatisch erzeugen. Auf diese Weise kann man recht schnell ein gültiges Dokument erzeugen, ohne viel tippen zu müssen. Alternativ kann der Anwender natürlich auch bereits bestehende XML-Dateien öffnen und weiter bearbeiten.

Der Benutzer kann entweder im Text- oder Baummodus durch das Dokument navigieren und Elemente und Attribute löschen oder ergänzen. Abhängig von der aktuellen Cursorposition blendet XDoc die hier möglichen Elemente und Attribute

ein, und der Benutzer kann diese dann per Mausklick einfügen.

Obwohl sich XDoc noch in der Entwicklung befindet und die Oberfläche noch nicht auf die Bedürfnisse von Benutzern ohne XML-Expertenwissen zugeschnitten ist, zeigten erste Praxistests, dass die Bearbeitung von XML-Dokumenten mit XDoc auch für ungeübte Benutzer einfacher und schneller ist als mit Textverarbeitungsprogrammen im WYSIWYG-Modus. Die Einarbeitungszeit beträgt für Benutzer, die rudimentäre HTML-Kenntnisse haben, noch etwa einen halben Tag. Durch weitere Verbesserungen an der Benutzerschnittstelle lässt sich diese Zeit sicherlich weiter reduzieren.

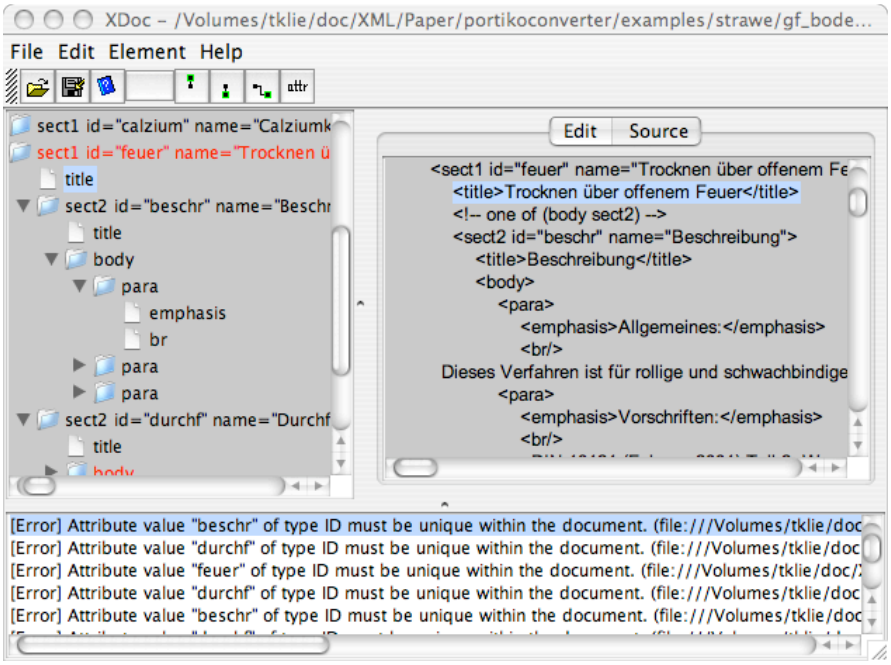

Abb. 3. Bildschirmfoto des XML-Editors XDoc

4 Evaluation

Von Autoren multimedialer Lernmodule wird in Gesprächen häufig der Vorbehalt geäußert, dass für die Erstellung von XML-basierten Lernmaterialien nur wenig komfortable Werkzeuge zur Verfügung stehen, denn ein wesentlicher Teil der Arbeit müsse im Quellcode vorgenommen werden. Dies sei gerade in den Fachdisziplinen ein erheblicher Nachteil, in denen überwiegend mit WYSIWYG-Editoren gearbeitet würde. Auch im Projektumfeld von PORTIKO war dies der Fall. Es handelte sich um Bauingenieure, die als Autoren für die Erstellung von Lernmaterialien verantwortlich waren. Die überwiegende Mehrheit besaß

allenfalls rudimentäre HTML-Kenntnisse.

Um den Adressaten den Einstieg zu erleichtern, wurde neben einem Tutorial auch eine zweistündige Schulung durchgeführt. Sie erwies sich für alle Beteiligten als ausreichend, um anschließend die eigenständige Arbeit mit dem Werkzeug zu ermöglichen. Die zuvor geäußerten Befürchtungen hinsichtlich einer zu schwierigen und techniklastigen Inhaltserstellung stellten sich als unbegründet heraus. Der angebotene Support über eine Mailingliste wurde nur geringfügig genutzt. Eine spätere Projektevaluation ergab, dass die Inhaltsersteller mit dieser Art der Inhaltserstellung weitestgehend zufrieden waren. Gerade technische Probleme traten nach Aussagen der Befragten überhaupt nicht auf und auch die Gewöhnung an die für die meisten Befragten ungewohnte Arbeitsumgebung stellte sich schnell ein.

Die Erfahrungen mit der XML-basierten Inhalteentwicklung für Lehr-/Lernmaterialien lassen drei wesentliche Schlussfolgerungen zu:

1. Eine beschränkte Menge an Elementen und Formatierungsattributen beschleunigt die Eingewöhnung und den Entwicklungsprozess.
2. Inhaltserstellern wird durch die beschränkte Anzahl an Elementen und Attributen aus didaktischer Sicht eine klare Struktur vorgegeben.. So lässt es sich von vorn herein vermeiden, dass Inhaltsersteller schlecht strukturierte Lernmodule produzieren.
3. Nachdem Autoren den Einstieg in eine oftmals zunächst befremdlich wirkende Entwicklungsumgebung geschafft haben, wissen sie die Vorteile eines sehr stark strukturierenden Rahmens – auch ohne WYSIWYG – zu schätzen.

5 Zusammenfassung und Ausblick

In diesem Beitrag haben die Autoren einen Ansatz zur XML-basierten Erstellung von Vorlesungsskripten vorgestellt. Kernbestandteil ist die im PORTIKO-Projekt entstandene, gleichnamige Inhaltsbeschreibungssprache, die im Rahmen von ELAN ergänzt und verfeinert wurde. Durch den Einsatz von XSL-Transformationen kann die Darstellungsform der Inhalte nahezu beliebig angepasst werden. Damit ist einerseits eine nahtlose Integration in verschiedene E-Learning-Plattformen möglich, andererseits lässt sich auch ein für den Druck geeignetes Format generieren.

Die praktischen Erfahrungen zeigen, dass es – anders als etwa in [8] dargestellt – nicht zwingend erforderlich ist, den Inhaltserstellern eine WYSIWYG-Benutzeroberfläche zur Verfügung zu stellen. Vielmehr wissen es die Inhaltsersteller zu schätzen, sich nicht um gestalterische Dinge kümmern zu müssen. Sicherlich hängt der Nutzen einer WYSIWYG-Umgebung auch stark von der Art der erzeugten Inhalte ab. Für stark textbezogene Inhalte, bei denen gestalterische Elemente vollständig in die Stylesheets ausgelagert werden können, erscheint der WYSIWYG-Ansatz fragwürdig. Bei der Gestaltung von Lerneinheiten mit wenig Text und sehr vielen optischen Elementen, wie es wohl in [8] der Fall war, kann eine WYSIWYG-Entwicklungsumgebung sicherlich

sinnvoll sein.

Weiterhin haben wir mögliche Ansätze für die Gestaltung von XML-Autorenwerkzeugen anhand von drei Beispielen diskutiert, darunter auch den selbstentwickelten Open Source Editor XDoc. Er bietet zusätzlich zur normalen Textansicht eine kontextsensitive Baumansicht, in der der Benutzer die an dieser Stelle gültigen Elemente und Attribute einfügen kann. Die Validierung gegen eine XML-Schema-Datei erfolgt bei diesem Editor im Hintergrund permanent, wodurch Fehler in der XML-Syntax sofort behoben werden können. Die Verwendung einer Textverarbeitungssoftware als WYSIWYG-XML-Editor erwies sich dagegen als derzeit ungeeignet.

Parallel zu PORTIKO wurden etliche weitere Sprachen entwickelt (bzw. weiterentwickelt), deren Beschreibungsmöglichkeiten sich mit denen von PORTIKO überschneiden. Zu nennen sind hier vor allem DocBook [14], die Learning Material Markup Language (LMML) [4] und die Multidimensional LearningObjects and Modular Lectures Markup Language (<ML>^{3}) [9].

Es ist unerlässlich, dass nun, nachdem in den letzten Jahren praktische Erfahrungen mit all diesen Sprachen gesammelt wurden, ein Konsolidierungsprozess einsetzt, der schließlich in einem standardisierten E-Learning-Datenformat mündet. Die bisherigen Bemühungen in diese Richtung vom IMS Global Learning Consortium [5] und Advanced Distance Learning (ADL) [1] sind zwar durchaus zu begrüßen, jedoch bleiben die vorgeschlagenen Lösungen nach wie vor auf einem sehr abstrakten Niveau. Damit E-Learning-Strategien langfristig erfolgreich sein können, muss ein modulares und erweiterbares Datenformat für textbasierte E-Learning-Materialien standardisiert werden. Einzelne Projekte haben gezeigt, dass die XML-Technologie hierfür ideale technische Möglichkeiten bietet. Der nächste logische Schritt wäre die Zusammenfassung der erarbeiteten Lösungen in einem gemeinsamen Standard.

Literatur

1. Advanced Distributed Learning. Homepage. http://www.adlnet.org/.
2. Ulf-Daniel Ehlers, Wolfgang Gerteis, Torsten Holmer und Helmut W. Jung, Hrsg. E-Learning-Services im Spannungsfeld von Pädagogik, Ökonomie und Technologie. W. Bertelsmann-Verlag, 1. Auflage, 2003.
3. The Mozilla Organization. Firefox. http://www.mozilla.org/products/firefox.
4. Burkhard Freitag. LMML - Eine Sprachfamilie für eLearning Content. In 32. Jahrestagung der Gesellschaft für Informatik, Dortmund, 2002.
5. IMS Global Learning Consortium, Inc. Homepage. http://www.imsglobal.org/.
6. Jörg Kiegeland. XDoc, The Official Homepage. http://xdoc.sourceforge.net/.

7. Carsten Karkola, Christoph Klinzmann und Astrid Weilert. Technische Realisierung. In Abschlussbericht PORTIKO, Seiten 24 – 55. TU Braunschweig und TU Dresden, Juli 2004.
8. Lars Kornelsen, Ulrike Lucke, Djamshid Tavangarian, Matthias Waldhauer und Natalia Ossipova. Strategien und Werkzeuge zur Erstellung multimedialer Lehr- und Lernmaterialien auf Basis von XML. In Delfi 2004: Die 2. e-Learning Fachtagung Informatik, Lecture Notes in Informatics, P-52. Gesellschaft für Informatik, September 2004.
9. Ulrike Lucke, Djamshid Tavangarian und Denny Voigt. Multidimensional Educational Multimedia with <ML>^{3}. In Proceedings of E-Learn 2003, World Conference on E-Learning in Corporate, Government, Healthcare, & Higher Education, November 2003.
10. World Wide Web Consortium. Mathematical Markup Language (MathML) Version 2.0 (Second Edition), Oktober 2003. http://www.w3.org/TR/2003/REC-MathML2-20031021.
11. Microsoft. Microsoft Office Online: Word, 2005. http://office.microsoft.com/de-de/FX010857991031.aspx.
12. Nick Richards, John McCreesh und Louis Suárez-Potts. OpenOffice.org 1.0.x Features – Writer. OpenOffice.org, Mai 2003. http://www.openoffice.org/dev_docs/source/features.html#writer.
13. Design Science. WebEQ. http://www.dessci.com/en/products/webeq/.
14. Norm Walsh und Leonard Muellner. DocBook: The Definitive Guide with CD-ROM. O'Reilly & Associates, Inc., 1999.
15. Wattle Software. XMLwriter, Februar 2005. http://XMLwriter.net.
16. World Wide Web Consortium. XSL Transformations (XSLT), November 1999. http://www.w3.org/TR/xslt.