

# Schreibt strukturiert!

Dr. Hendrik Bunke

7. Dezember 2005

## Inhaltsverzeichnis

<b>1</b>	<b>Das Problem: Texte schreiben mit Word und anderen Textverarbeitungen</b>	<b>1</b>
<b>2</b>	<b>What you see is what you mean. Zum Nutzen von strukturiertem Text</b>	<b>3</b>
<b>3</b>	<b>Was bieten die XML-Markupsprachen?</b>	<b>4</b>
<b>4</b>	<b>Zur Eignung von Docbook XML für geisteswissenschaftliche Texte</b>	<b>6</b>
<b>5</b>	<b>Usability</b>	<b>7</b>
5.1	XML-Editoren mit WYSIWYM Ansicht . . . . .	8
<b>6</b>	<b>Fazit</b>	<b>9</b>

### Zusammenfassung

Der Artikel erörtert die Notwendigkeit von strukturiertem, wiederverwendbarem und standardisiertem Text und entsprechenden Dateiformaten für den Bereich der Geisteswissenschaften. Untersucht wird außerdem die Eignung speziell von Docbook für das Verfassen und Speichern von akademischen Texten aus diesem Bereich. Der Beitrag versteht sich als Plädoyer für bessere Textverarbeitung und -produktion (= nachhaltiger, strukturierter und zugänglicher) im digitalen Zeitalter und soll auch die nicht technisch Interessierten AutorInnen aus den Geisteswissenschaften ansprechen. Das Ganze basiert auf einem kleinen internen Paper, das ich mal für eine Arbeitsgruppe geschrieben habe und hat - außer meiner Vorliebe für XML - auch den Hintergrund leidvoller Erfahrungen bei der Herstellung von Büchern sowie einer geschichtswissenschaftlichen Zeitschrift, jeweils mit mehreren AutorInnen. Ich darf gar nicht drüber nachdenken, wieviel Arbeit mir erspart geblieben wäre, würden die alle standardmäßig (Docbook-)XML verwenden. Sie benutzen aber Word, und da fängt das Problem an.

## 1 Das Problem: Texte schreiben mit Word und anderen Textverarbeitungen

Wer schon längere Texte (Artikel, Bücher, Abschlussarbeiten etc.) mit Word geschrieben hat, kennt die akuten Probleme. Das Programm stürzt (immer in den falschen Mo-

menten) ab, Kopf- und Fußzeilen sind ein Albtraum, Bilder verschwinden spurlos, Seitenumbrüche machen sich selbstständig, Fußnoten erscheinen auf der falschen Seite etc.pp.

Stabilität und Usability sind also das eine Problem. Das zweite ist ein vielleicht noch viel schwerwiegenderes. Alle Textverarbeitungen trennen den Inhalt nicht vom Layout. Für einen einfachen Text wie ein Brief, ein kurzes Konzeptpapier etc. mag das noch ziemlich zweitrangig sein. Für längere Texte, die weiterverarbeitet, publiziert und für unterschiedliche Medien aufbereitet werden müssen oder sollen, ist es eine Katastrophe.

Das dritte große Manko von Textverarbeitungen ist das der Kompatibilität und proprietärer Dateiformate. Haben Sie mal versucht, einen Word-Text aus DOS-Zeiten in einer aktuellen Word-Version zu öffnen? Sehen Sie. Selbst zwischen aktuelleren Word-Versionen von Microsoft gibt es Kompatibilitätsprobleme. Für die Weiterverarbeitung von Dokumenten und ihre nachhaltige, langfristige Speicherung und Nutzbarkeit sind proprietäre Formate kontraproduktiv, und zwar in jeder Hinsicht.

Letzteres Problem wird mittlerweile immerhin dadurch gemildert, dass sich XML als grundlegendes Dateiformat für Office-Applikationen allmählich durchsetzt. Dies gilt vor allem für OpenOffice.org, aber auch für neue Word-Versionen von Microsoft, das allerdings auch hier sein eigenes Süppchen kocht, sich Standards wie dem OASIS-Format verweigert und das zugrunde liegende Schema nicht veröffentlicht. Generell gilt außerdem, dass durch die Nutzung von XML die beiden anderen oben genannten Probleme nicht automatisch gelöst werden. Nach wie vor finden sich auch im OpenOffice-Format Layout-Anweisungen, und nach wie vor sind Textverarbeitungen sehr komplexe und fehleranfällige Programme, die oft eher vom Schreiben ablenken denn die Konzentration auf Inhalte fördern.

Gefordert ist also eine von Hersteller, Plattform und verwendeter Applikation unabhängige Herstellung von Texten, die Inhalt und Layout strikt trennt sowie standardbasiert eine möglichst langfristige und vielfältige Les- und Nutzbarkeit ermöglicht. Das ist aber noch nicht alles. Hinzu kommt ein für 'normale' AutorInnen vielleicht erheblich relevanterer Aspekt: Textverarbeitungen kennen keine Semantik und geben keine (standardisierte) Struktur vor. Ein Absatz kann ein normaler Absatz sein, oder auch die Benennung des Autors, des Datums oder ein Zitat oder anderes im Zusammenhang mit dem Textinhalt Bedeutsames. Im Zentrum steht die Präsentation des Textes, nicht die Bedeutung. Die bleibt letztlich interpretationsfähig und ausschließlich abhängig von der Formatierung und Position im Text. Ebenso beliebig und dem Können (der Willkürlichkeit) der AutorInnen überlassen bleibt auch die Textstruktur. Niemand hindert mich daran, innerhalb eines Unterkapitels eine Kapitelüberschrift zu benutzen. Oder in einem Text die Überschriften für Unterkapitel verschieden zu formatieren. Das genau ist das Dilemma von Textverarbeitungen: Struktur und Bedeutungen können nur über die Formatierung vermittelt werden. Sie sind weder eindeutig noch irgendwie standardisiert, erlauben also Fehler und Beliebigkeit und sind außerdem auf die Interpretation durch den Leser bzw. die weiterverarbeitende Instanz angewiesen.

## 2 What you see is what you mean. Zum Nutzen von strukturiertem Text

Das Paradigma der gängigen Textverarbeitungen heißt *What you see is what you get* (*WYSIWYG*). Aus diesem Paradigma ergeben sich die o.g. Probleme: Im Vordergrund steht die Präsentation, nicht die Struktur und Bedeutung (ganz abgesehen davon ist der Anspruch *WYSIWYG* natürlich auch nie einzulösen und mit zahlreichen Problemen wie Druckertreibern, fehlenden Schriftarten etc. behaftet). Im Gegensatz dazu steht der hier propagierte Ansatz, für den sich das Kürzel *WYSIWYM* etabliert hat: *What you see is what you mean*.

*WYSIWYM* ist von den Programmierern des Latex-Frontends Lyx in die Welt gesetzt worden ('LyX is the first *WYSIWYM* document processor.'). und bezeichnet die skizzierte Prioritäten-Verschiebung von der Formatierung zu Inhalt und Struktur. LyX als *WYSIWYM*-Editor formatiert keinen Text, sondern er strukturiert Text. Was dann bspw. so aussieht:

```
\layout Section*
Überschrift
\layout Paragraph
Hier steht ein einfacher absatz mit viel blödem text.
Lyx und Latex sind schon toll, aber XML ist besser.
```

Die Überschrift wird also nicht dadurch zu einer solchen, dass der Text fett und groß formatiert wird, sondern indem ihm diese Funktion explizit zugewiesen wird und erst aus dieser Strukturvorgabe sich - am Ende der Kette - die Formatierung ergibt. Lyx ermöglicht, diesen Code nicht selbst schreiben zu müssen, sondern zeigt dem Benutzer eine einfache Oberfläche, mit der er schreiben und arbeiten kann wie mit der gewohnten Textverarbeitung. Dabei kann er sich konzentrieren auf die Struktur und den Text, die Formatierung erledigt LyX mittels der üblichen LaTeX-Werkzeugkette. Am Ende steht ein sauber strukturiertes und vor allem auch - das ist eben herausragend bei TeX - ein sehr professionell aussehendes Dokument.

Aber auf Lyx und LaTeX soll hier gar nicht weiter eingegangen werden. Wer primär an einer sauberen, printorientierten und immerhin ansatzweise strukturorientierten Textverarbeitung interessiert ist, sollte LaTeX unbedingt ausprobieren. Auch technikferne AutorInnen in den Geisteswissenschaften können Tools wie Lyx ganz wunderbar und erheblich problemloser als die üblichen Textverarbeitungen benutzen. Es handelt sich um eine sehr ausgereifte Technologie, die man bedenkenlos empfehlen kann.

Allerdings ist Latex immer noch primär druckorientiert, eine Weiterverarbeitung in andere Formate wie XHTML, Text, RTF etc. fürs Web oder andere Druckanforderungen ist kompliziert bis unmöglich. Hinzu kommt, dass auch in Tex-Dateien zahlreiche Formatierungsanweisungen enthalten sind (font-klasse, Papierformat etc.). Als universelleres Format für Texte ist deshalb mittlerweile definitiv XML vorzuziehen.

XML ist *die* (Auszeichnungs-)Sprache des Webs und des digitalen Zeitalters. Das kann man ohne allzuviel Pathos sagen, und es braucht hier eigentlich gar nicht weiter auf die Vorzüge eingegangen werden. Entscheidend für unseren Kontext sind vor allem zwei wesentliche Aspekte von XML: die hohe Transformationfähigkeit für verschiedene Ausgabemedien (Print, Web, Mobile etc.) sowie die Flexibilität und Präzision zur

Auszeichnung oder auch ‘Semantisierung’ von Text. Ein Absatz ist explizit genau als solcher erkenn- und lesbar, und zwar für Mensch und Maschine. Gleiches gilt noch viel mehr für mit hohem Bedeutungsgehalt versehenen Informationen, wie z.B. Autorennamen etc. Das obige kleine Beispiel könnte in XML so aussehen:

```
<section>
  <title>Das ist die Überschrift des Abschnitts</title>
  <para> hier steht ein einfacher absatz mit viel blödem ←
    text. Lyx und
    Latex sind schon toll, aber XML ist besser.</para>
</section>
```

Man sieht schon an diesem Mini-Beispiel, dass z.B. die Überschrift als Format sich zwingend aus der Struktur und Auszeichnung als `<title>` ergibt. Das ist die eigentliche Bedeutung von WYSIWYM und wird hier erheblich konsequenter umgesetzt als bei Lyx und Latex.

### 3 Was bieten die XML-Markupsprachen?

Nun ist XML an sich noch gar keine ‘Sprache’ oder ein System wie z.B. TeX. Es ist eine ‘Metasprache’, mit der sich für unterschiedlichste Zwecke Sprachen formulieren lassen (XHTML ist z.B. die bekannteste). Rein theoretisch wäre es auch möglich, dass ich mir selbst eine passende XML-Sprache erstelle und mit den entsprechenden Tools (XSLT, XSL-FO) eine entsprechende Transformations- und Publikationskette baue. Freilich muss oder kann mensch aber natürlich das Rad nicht immer wieder neu erfinden. Es gibt bereits einige XML-Sprachen, die speziell für Textpublikation und -verarbeitung in verschiedenen Bereichen gedacht sind.

Als erstes genannt werden und in diesem Beitrag noch genauer betrachtet werden soll **Docbook**. Docbook ist eine XML-Markup-Sprache, die primär für technische Dokumentationen gedacht ist und eingesetzt wird. Also solche ist sie sehr weit verbreitet. Docbook ist sehr mächtig, es lassen sich so ziemlich alle publikationsrelevanten Formate definieren (book, article, slides etc.) und mittels einer speziellen und ebenfalls sehr mächtigen Docbook-XSL in zahlreiche Formate transformieren (PDF, RTF, HTML etc.). Die Mächtigkeit bzw. Vielseitigkeit von Docbook hat zu einer sehr großen Zahl von Elementen geführt. Derzeit sind es ca. 400. Da diese große Zahl für die meisten Dokumente gar nicht unbedingt nötig ist, hat man mit ‘Simplified Docbook’ versucht, eine abgespeckte Version zu schaffen, die nur noch etwa 100 Elemente enthält. Ein ähnliches Ziel verfolgen Projekte wie **tbook**, das sich vor allem auf wissenschaftliche Texte konzentriert und mit ca. 80 Elementen auskommt, allerdings wohl auch eher ein Privatprojekt ist. Neuerdings, d.h. seit der Verwendung von RelaxNG als Schema-sprache für Docbook, existiert mit **docbook tiny** eine nahezu geniale Möglichkeit, die Zahl der Elemente von Docbook für den jeweiligen Einsatzzweck anzupassen. Wer will, kann so im Extremfall mit fünf(!) Elementen auskommen und valides Docbook schreiben.

Docbook ist der Quasi-Standard, wenn es um technische Dokumentationen, Computerliteratur u.ä. geht, gewinnt aber zunehmend auch an Bedeutung für andere Bereiche. Das liegt auch daran, dass es mittlerweile eine Reihe von Tools und XSL-Dateien gibt, die den Umgang mit und die Transformation von Docbook leichter machen. Inwiefern sich Docbook auch für technikferne Bereiche wie die hier betrachteten Geistes- und Sozialwissenschaften eignet, werden wir unten noch genauer untersuchen. Hier muss noch zunächst noch der zweite Quasi-Standard für Dokument-Auszeichnungssprachen genannt werden, nämlich TEI-XML ([www.tei-c.org](http://www.tei-c.org)). TEI-XML ist primär für die Literaturanalyse und -bewahrung konzipiert worden und wird im wesentlichen zur Auszeichnung bereits existierender Literatur (Belletristik, Poesie etc.) verwendet. Es eignet sich deshalb weniger zum *Schreiben* von wissenschaftlichen Texten. Auch TEI-XML ist im übrigen sehr mächtig und umfasst derzeit sehr viele Elemente. Auch hier gibt es deshalb eine Lite-Version mit reduzierter Elementanzahl, die - so mein Eindruck - häufiger verwendet wird.

Neben diesen sehr umfangreichen und vielseitigen, mächtigen XML-Sprachen könnte man natürlich auch einfach XHTML verwenden. XHTML ist strukturiert, in Maßen transformierbar und - mit etwas Mühe und ordentlichem CSS - sogar für einfache Druckdokumente verwendbar. Hinzu kommt, dass es validierfähige Editoren wie Sand am Meer gibt. Aber XHTML ist zuallererst fürs Web gedacht. Eigentlich sogar ausschließlich dafür. Und deshalb sollte es nicht das primäre Format für strukturierte Texte sein, sondern vielmehr ein Transformationsformat. Hinzu kommt, dass XHTML nur sehr wenig Struktur und Semantik vorsieht. Metadaten lassen sich zwar implizit integrieren (Semantik), aber wer seinen Text bedeutungsvoll codieren will und z.B. in benannte automatische Sektionen strukturieren will, stößt früh an Grenzen. XHTML 1.0 kennt lediglich `<h1>`-`<h6>` als Elemente für Überschriften. Das sind - eigentlich ähnlich wie bei den Textverarbeitungen - nicht vielmehr als Formatierungsanweisungen, aber nur indirekt strukturierende Elemente. XHTML 2 wird hier nach heutigem Stand immerhin etwas Abhilfe schaffen und mit u.a. dem neuen Element `<section>` für mehr explizite Struktur und Strukturierungsmöglichkeiten sorgen. Dennoch sollte XHTML nur für einfache Texte (oder auch Präsentationen) im Netz verwendet werden.

Nicht ganz in unseren Fokus passen die neuen XML-Formate der Textverarbeitungen, vor allem von OpenOffice.org bzw. das neue OASIS-Standardformat OpenDocument (das OpenOffice ab Version 2.0 verwendet). Es passt deshalb nicht, weil es lediglich in XML speichert und - wie oben bereits erwähnt - immer noch eher an Formatierung und nicht an Struktur interessiert ist. Das offene(!) XML-Format für Textverarbeitungen ist also zwar ein sehr großer Fortschritt im Hinblick auf Transformatierbarkeit und Offenheit, bietet aber für unsere hier betrachteten Zwecke, nämlich wissenschaftliche Texte im Bereich der Geisteswissenschaften zu schreiben, immer noch zu wenig Struktur und Semantik.

Es bleibt also momentan vor allem Docbook. TEI-XML ist wie gesagt eher zur Kodierung bereits vorhandener Texte gedacht und geeignet, auch wenn es vielfach durchaus zum Schreiben eingesetzt wird. Docbook ist primär für Computerdokumentationen erfunden worden, ist allerdings heute auch darüber hinaus vielfach im Einsatz. Eignet es sich für geisteswissenschaftliche Texte?

## 4 Zur Eignung von Docbook XML für geisteswissenschaftliche Texte

Betrachtet man die wesentlichen Elemente eines normalen (geisteswissenschaftlichen) Textes, lässt sich bei Docbook kein Mangel feststellen. Es gibt Absätze (<para>), Abschnitte (<section>), entsprechende Titel (<title>), Zitate (<quote>) und sogar Fußnoten (<footnote>) sowie Bibliographien (<bibliography>), die sich mittlerweile auch mit externen Programmen verwalten und integrieren lassen (s.a. [Open Source Literaturverwaltung](#)). Das alles (es gibt natürlich noch weitere nützliche Elemente) reicht für einen Standard-Text in den Geisteswissenschaften völlig aus. Gleichzeitig wäre aber bereits ein Dokument mit diesem kleinen und sehr leicht erlernbarem<sup>1</sup> Satz von Elementen erheblich leichter transformierbar und semantisch 'wertvoller', als mit üblichen Textverarbeitungen hergestellte.

Nehmen wir als Beispiel einen simplen Artikel, der z.B. so aussehen könnte.

```
<article lang="de_DE">

  <title>Schreibt strukturiert!</title>
  <subtitle>XML und Docbook in Sozial- und
    Geisteswissenschaften</subtitle>

  <author>Dr. Hendrik Bunke</author>

  <section>
    <title>Erster Abschnitt</title>

    <para> Duis autem vel eum iriure dolor in hendrerit in ←
      vulputate velit
      esse molestie consequat, vel illum dolore eu feugiat ←
      nulla facilisis at
      vero et accumsan et iusto odio dignissim qui blandit ←
      praesent luptatum
      zzril delenit augue duis dolore te feugait nulla ←
      facilisi. Lorem ipsum
      dolor sit amet, consectetur adipiscing elit, sed diam ←
      nonummy nibh
      euismod tincidunt ut laoreet dolore magna aliquam erat ←
      volutpat.</para>

  </section>

  <section>
    <title>Ein zweiter Abschnitt</title>
    <para>...weiterer Text...</para>
```

<sup>1</sup> Wenn ein entsprechender Editor eingesetzt wird, braucht mensch die Elemente noch nicht einmal selbst lernen, auch wenn das natürlich keinesfalls schadet. Siehe zu den Editoren folgenden Abschnitt.

```
</section>  
  
</article>
```

Das Beispiel zeigt einen validen Artikel in Docbook. Das ist sehr einfaches Markup, das zudem strikt inhalts- und strukturorientiert ist.

Für einen Standardartikel oder ein normales Buch (meine Dissertation bspw. lässt sich völlig problemlos mit einigen wenigen Docbook-Elementen erstellen) ist Docbook also mehr als geeignet. Die zahlreichen Elemente, die primär oder ausschließlich für technische (Computer-)Literatur gedacht und verwendbar sind, können dabei schlicht ignoriert werden. Interessanter oder auch problematischer wird es bei seltenen, aber eben zumindest für einige Bereiche durchaus regelmäßig relevanten Erfordernissen. Beispielsweise werden in der qualitativen Sozialforschung oder auch der Psychologie relativ viele Interviewsequenzen benutzt. Solche Textarten mit ganz spezifischen Erfordernissen aus einem eigenen semantischen Raum sind mit Docbook nur sehr schwer zu realisieren. Hier wäre vielleicht eine eigene Markup-Sprache außerordentlich sinnvoll, ähnlich wie TEI für die Linguistik.<sup>2</sup>

## 5 Usability

Die Nützlichkeit von XML-Dokumenten im allgemeinen wie auch speziell von Docbook ist also nun klar. Wie aber steht es denn mit der berühmten 'Usability'? Lässt sich den 'normalen' AutorInnen das Umsteigen von WYSIWYG-Textverarbeitungen auf Markup und Texteditoren vermitteln und wenn ja wie?

In der Tat bedeutet so ein Umstieg natürlich eine gewisse Anstrengung und erfordert ein erhebliches Umdenken bei der Texterstellung. Das ist aber auch beabsichtigt, weil notwendig. Einige Begründungen, aber auch die ganz persönlichen Profite für die AutorInnen habe ich oben bereits versucht zu skizzieren. Es ist außerdem notwendig, weil das Zeitalter der Schreibmaschine nun mal endgültig vorbei ist. Texte werden heute zunehmend für mehrere Ausgabemedien produziert, Texte sind zunehmend vernetzt oder gar interaktiv. Die Bedeutung von Suchmaschinen, von 'Information Retrieval' und Wissensmanagement kann gar nicht mehr hoch genug eingeschätzt werden, und das 'Semantic Web' ist mittlerweile von der Vision zur zumindest partiellen Realität geworden. Über Sinn und Unsinn dieser Entwicklungen kann man vielleicht streiten, aber die Textproduktion wird sich ihr dennoch anpassen müssen. Oder nehmen Sie heute noch handgeschriebene Seminararbeiten an?

Der notwendige Umlernprozess muss dabei ein gar nicht so gewaltiger sein. Nehmen wir mal die Frage der Umwandlung und Formatierung aus (auf die als reiner Autor ja eigentlich auch verzichtet werden kann und sollte), dann ist der Schreib- und Textproduktionsprozess eigentlich sogar viel einfacher, als bei herkömmlichen Textverarbeitungen. Es braucht wie gesehen, nur einige wenige Elemente, um Standardtexte mit gültigem Docbook-XML zu produzieren. Im Wesentlichen kann sich der Autor

<sup>2</sup> Inwiefern TEI-XML sich bereits für solche speziellen Textarten eignet, wäre noch genauer zu prüfen.

also wieder auf seine eigentliche Aufgabe, nämlich den Inhalt, konzentrieren. Hinzu kommt, dass man nicht auf ein spezielles Programm festgelegt ist (oder von dessen Markern abhängig ist). Jeder normale Texteditor (vielleicht noch mit Syntaxhighlighting) ist ausreichend, wenn auch nicht unbedingt komfortabel. Mittlerweile gibt es außerdem eine Reihe von speziellen XML-Editoren, die eine WYSIWYM-Ansicht bieten, den Nutzer also mit keinerlei Markup konfrontieren. Mit diesen Editoren wird XML für die breite Masse, die AutorInnen und WissenschaftlerInnen auch in nichttechnischen Bereichen nutzbar. Hier nur eine kleine Auswahl.

## 5.1 XML-Editoren mit WYSIWYM Ansicht

**XXE von XMLmind** scheint mir bislang noch der ausgereifteste der frei erhältlichen Editoren zu sein. Das Programm ist als leicht eingeschränkte (aber ausreichende) ‘Standard Edition’) umsonst, allerdings nicht Open Source. Der Editor verbirgt konsequent den XML-Code und lässt den Benutzer nur einen per CSS erzeugten, formatierten Text sehen. Das ist dann natürlich nicht WYSIWG, erleichtert aber Menschen ohne Affinität zu spitzen Klammer durchaus sehr die Bearbeitung des Dokuments und die Konzentration auf den Inhalt. Docbook wird standardmäßig unterstützt (neben anderen DTDs wie XHTML). Im Open Source Bereich verfolgen **Conglomerate** und **Vex** ähnliche Konzepte. Conglomerate ist bis dato noch etwas unausgereift und nicht unbedingt empfehlenswert, wenn auch vielversprechend. Vex baut auf Eclipse auf und macht in der Beschreibung und den Screenshots einen sehr guten Eindruck. Leider habe ich es unter FreeBSD nicht zum Laufen bekommen. Im kommerziellen Bereich noch sehr interessant ist **Serna** (Syntext).

Wer gerne mit dem reinen Quelltext arbeitet (was viel mehr Spaß macht und meistens auch erheblich schneller und vor allem flexibler geht!) oder das gerne ausprobieren möchte, dem seien als Editoren jEdit (mit XML-Plugin) oder Emacs (mit nxml-mode) empfohlen. Wichtig (wenn natürlich auch nicht unbedingt nötig) ist in diesem Fall, dass das Programm Syntaxhighlighting hat und XML-Dokumente validiert werden können.

Erwähnt werden muss eigentlich auch noch die Textverarbeitung von OpenOffice. Es ist durchaus möglich, OpenOffice als Editor für Docbook-Dokumente zu verwenden. Das liegt auch nahe, speichert OO doch ohnehin in XML, auch wenn das nicht sichtbar ist. Der Trick dabei ist der Einsatz von genau definierten Formatvorlagen, die mittels eines XSL-Filters in Docbook XML umgewandelt werden. Meine bisherigen Erfahrungen damit sind allerdings nicht so gut. Kann sein, dass sich jetzt mit der neuen Version 2.0 gebessert hat, ich hab’s noch nicht ausprobiert. Ohnehin kann eine solche Variante m.E. nur eine Not- oder Zwischenlösung sein. Das gilt auch für die gerade veröffentlichten XSL-Stylesheets zur Umwandlung von Docbook in WordML und vice versa (<http://www.explain.com.au/oss/docbook/>).

Alle erwähnten Programme sind übrigens entweder Open Source (Conglomerate, Vex, OpenOffice, jedit, emacs), frei erhältlich (XXE) oder zu einigermaßen zivilen Preisen verfügbar (Serna). Außerdem laufen alle Editoren auf Linux/FreeBSD, Mac und Windows.

## 6 Fazit

Docbook eignet sich durchaus auch für das Verfassen und die Weiterverarbeitung von geisteswissenschaftlichen Texten. Die hohe Zahl von ursprünglich primär für den IT-Bereich gedachten Elementen kann dabei ignoriert werden. Wesentliche Elemente wie para, footnote, section, title, quote etc. sind für geisteswissenschaftliche Texte genauso gültig und geeignet. Mit 'Docbook tiny' existiert außerdem neuerdings die Möglichkeit, sich seinen eigenen Elementesatz, quasi sein eigenes Docbook, zurecht zu schneiden. Die Weiterverarbeitung von Docbook-XML ist durch zahlreiche Tools und (XSL-)Vorlagen sehr sicher, robust und inzwischen vielfach erprobt. Gegenüber Word und anderen klassischen Textverarbeitungen sind sowohl diese Weiterverarbeitungsmöglichkeiten wie auch die hohe Strukturalität und Semantik von XML allgemein und Docbook im Besonderen ein enormer Vorteil. Diesen sollte man als ernsthafter Textproduzent nicht ignorieren.

Bleibt die Frage der Usability und des Lernaufwands. In der Tat würde es natürlich für einen lediglich Word gewohnten Autoren schon eine enorme Umstellung, plötzlich ohne Seitenansicht, Formatierungsmöglichkeiten etc. schreiben zu müssen. Aber - das kann ich auch aus eigener Erfahrung sagen - es bedeutet auch einiges an Befreiung von eben genau diesen Aspekten. Autoren können sich zunächst auf Inhalt und auch die Struktur ihres Textes konzentrieren. Sie brauchen (für manche gilt sicher auch: sie können) sich nicht mehr um Schriftart, Absatzabstand oder sonstige Formatierungsfragen zu kümmern. Wenn sie es wollen, ist diese Formatierung streng von der eigentlichen Textproduktion wie auch dem eigentlichen Text getrennt. Das sollte auch als persönlicher Fortschritt, als Arbeitserleichterung begriffen werden. Die für Standard-Texte wenigen Elemente sind schnell gelernt und entpuppen sich als geradezu lächerlich im Vergleich zu dem nicht nur persönlichen sondern auch - wie oben skizziert - für den gesamte Entwicklung von Textproduktion, Weiterverarbeitung und Integration z.B. ins Web. Die jetzt schon erhältlichen Editoren ermöglichen es sogar, weitgehend ohne Quellcode und angenähert an vertraute Textverarbeitungsflächen zu arbeiten. Und zu guter Letzt: Word zu lernen war schließlich auch jede Menge Arbeitsaufwand und Word zu benutzen und Workarounds für die diversen Probleme, Bugs etc. zu finden noch viel mehr.

Es gibt also viele gute Gründe, sich als geistes- oder sozialwissenschaftliche AutorIn mit XML, Docbook und strukturiertem Text zu beschäftigen. Die ersten Schritte erfordern lediglich etwas Veränderungswillen und Offenheit, dann erschließt sich ganz schnell eine völlig neue Sichtweise der Texterstellung und -verarbeitung. Natürlich wurde auch dieser Artikel in Docbook geschrieben. Wer will, kann sich die [Originalversion](#) oder auch die daraus generierte [PDF-Datei](#) ansehen.