Staab, Eva

Entwicklung einer XML-Struktur zur strukturierten Speicherung von Zeitschrifteninhalten

– Diplomarbeit –

Hochschule Mittweida – University of Applied Sciences (FH)

Fachbereich Medien

Mittweida, 2009

Hochschule Mittweida (FH)
University of Applied Sciences

MITTWEIDA
ROSSWEIN

Staab, Eva

Entwicklung einer XML-Struktur zur strukturieren Speicherung von Zeitschrifteninhalten

– eingereicht als Diplomarbeit –

Hochschule Mittweida – University of Applied Sciences (FH)

Fachbereich Medien

vorgelegte Arbeit wurde verteidigt am:

Erstprüfer: Prof. Dr. phil. Ludwig Hilmer
Zweitprüfer: Severin Taranko

Dresden, 2009

Staab, Eva:
Entwicklung einer XML-Struktur zur strukturierten Speicherung von Zeitschrifteninhalten . - 2009 - 101 S.
Dresden, Hochschule Mittweida (FH), Fachbereich Medien, Diplomarbeit

Referat

Die hier vorliegende Arbeit beschäftigt sich mit der Entwicklung einer Speicherstruktur für Zeitschriftenschrifteninhalte. Diese Speicherstruktur soll in der Lage sein, Inhalte so zu speichern, dass diese später unter Zuhilfenahme zusätzlicher Gestaltungsvorgaben in verschiedene Ausgabeformate exportiert werden können. Zur Analyse der Anforderungen, die die neue Speicherstruktur erfüllen muss, wird zunächst das Projekt vorgestellt in dem diese eingesetzt werden soll. Anschließend werden Unterschiede von Druck- und Printausgabe dargestellt, die sich direkt auf die Speicherstruktur auswirken. Nachdem die Funktionsweisen der Auszeichnungssprache XML und anderer Techniken zur strukturierten Speicherung von Inhalten erläutert wurden, folgen eine Anforderungsanalyse, die die gewonnenen Erkenntnisse zusammenführt sowie die Konzeption einer für den speziellen Anwendungsfall geeigneten Speicherstruktur auf Basis von XML.

Inhaltsverzeichnis

Abbildungsverzeichnis

Abkürzungsverzeichnis

CMS	Content Management System
CMYK	Cyan, Magenta, Yellow, Key (Schwarz), die vier Grundfarben der subtraktiven Farbmischung
CSS	Cascading Style Sheets
DTD	Document Type Definition
DTP	Desktop Publishing
PCL	Printer Control Language
PDF	Portable Document Format
RGB	Rot, Grün, Blau, die vier Grundfarben der additiven Farbmischung
SGML	Standard Generalized Markup Language
SVG	Scalable Vector Graphics
W3C	World Wide Web Consortium
XML	Extensible Markup Language
XSL	Extensible Stylesheet Language

Danksagung

Mein Dank gilt der queo GmbH in Dresden für die flexiblen Arbeitszeiten und das große Verständnis seitens der Geschäftsführung. Besonderer Dank gebührt dabei auch meinem Zweitprüfer Severin Taranko für die hervorragende Betreuung.

1. Zielstellung

Mit zunehmender Verbreitung des Internets werden Inhalte immer öfter in mehreren Ausgabeformaten veröffentlicht. Um den Aufwand für die Publikation in verschiedenen Medien zu verringern, ist es sinnvoll ein System zur Automatisierung der medienspezifischen Publikation zu entwickeln.In einem solchen Publikations-System müssen Inhalt und Gestaltungsvorgaben so gespeichert werden, dass sie flexibel in alle benötigten Ausgabeformate transformiert werden können. Um dabei die spezifischen Besonderheiten der einzelnen Ausgabeformate berücksichtigen zu können, werden Inhalte üblicherweise strukturiert gespeichert, ohne jedoch Informationen über die optische Repräsentation der Inhalte zu enthalten, da diese von Ausgabeformat zu Ausgabeformat wesentlich variieren kann. Die verwendete Struktur muss dabei so gestaltet sein, dass sie den darzustellenden Inhalt optimal wiedergibt.

Diese Arbeit beschäftigt sich mit der Entwicklung einer Speicherstruktur für den speziellen Anwendungsfall einer Internetplattform zur Erstellung und Publikation von Zeitschriften, dem sog. „wanZine“. Ziel dieser Arbeit ist die Entwicklung einer Speicherstruktur, mit der es möglich ist, Zeitschrifteninhalte strukturiert zu speichern, so dass diese später in verschiedene Ausgabemedien exportiert werden können.

Um den Verwendungszweck der zu erstellenden Speicherstruktur zu veranschaulichen, wird zunächst die Grundidee hinter dem wanZine-Projekt vorgestellt. Dabei wird auch der technische Aufbau des wanZine-Projekts erläutert, der den Rahmen dieser Arbeit bildet. Da das wanZine in der Lage sein soll, Zeitschriften sowohl für die Druck- als auch die Monitorausgabe zu exportieren, werden die unterschiedlichen Anforderungen analysiert, die Online- und Druckausgabe wegen ihrer spezifischen technischen Gegebenheiten und unterschlichen Nutzungsverhaltens an ein Layout stellen.

Im folgenden Kapitel werden die Eigenschaften und der Funktionsumfang der Auszeichnungssprache XML im Hinblick auf ihre Verwendbarkeit für

das wanZine-Projekt zusammenfassend dargestellt. Außerdem werden Möglichkeiten aufgezeigt, den Aufbau einer XML-Struktur zu beschreiben.

Im letzten Teil des Grundlagenteils werden bereits existierende Auszeichnungssprachen beispielhaft vorgestellt, mit denen es möglich ist, Daten und Layouts zu speichern. Der Schwerpunkt der Betrachtung liegt dabei auf den zwei grundsätzlichen Vorgehensweisen bei der Speicherung von Layouts: der flussorientierten Speicherung und der positionsorientierten Speicherung.

Anschließend wird analysiert, welche Anforderungen die neu zu erstellende Speicherstruktur aus technischer und gestalterischer Sicht erfüllen muss. Aufbauend auf den Erkenntnissen dieser Anforderungsanalyse wird zuletzt eine Struktur zur Speicherung von Inhalten definiert, die den zuvor gestellten Anforderungen genügt.

Im letzten Kapitel erfolgen eine Bewertung der in dieser Arbeit entwickelten XML-Struktur und ein Ausblick auf ihre zukünftige Verwendung innerhalb des wanZine-Projekts.

2. Grundlagen

Im folgenden Kapitel werden die technischen und gestalterischen Rahmenbedingungen dieser Arbeit erläutert. Die hier aufgeführten Aspekte wirken sich direkt oder indirekt darauf aus, wie der Inhalt eines wanZines strukturiert gespeichert werden muss.

Zunächst wird das wanZine-Projekt vorgestellt, in dem die neu zu erstellende Speicherstruktur eingesetzt werden soll. Ein für diese Arbeit besonders wichtiger Aspekt ist dabei die Veröffentlichung des Inhalts in verschiedene Ausgabemedien, die durch das wanZine ermöglicht werden soll. Um die Problematik bei der Veröffentlichung in mehrere Ausgabeformate zu verdeutlichen, werden einige grundlegende Gestaltungsprinzipien für die Druck- und Monitorausgabe vorgestellt. Der Fokus liegt dabei auf den Gestaltungsprinzipien, in denen sich Druck- und Monitorausgabe unterscheiden.

In den darauf folgenden beiden Unterkapiteln werden die technischen Grundlagen dieser Arbeit dargelegt. Dazu wird zunächst die Auszeichnungssprache XML vorgestellt, die die Basis der Speicherstruktur für das wanZine-Projekt bilden könnte. Unterschiedliche Vorgehensweisen und vorhandene Techniken zur strukturierten Speicherung von Inhalten werden vorgestellt.

2.1. Das wanZine-Projekt

2.1.1. Grundidee

Der Name „wanZine“ ist ein Neologismus aus WAN (kurz für wide area network, einem großflächigen Computernetzwerk) und der Endsilbe von magazine. Das wanZine ist also bereits seinem Namen nach ein Verbindungsglied zwischen traditionellen Zeitschriften und der modernen Kommunikation im Internet. Die Grundidee des wanZine-Projekts ist es, über eine Internetplattform die Erstellung und Veröffentlichung von Zeitschriften zu ermöglichen. Der Name „wanZine“ bezeichnet dabei gleichermaßen das ganze Projekt wie auch eine im Rahmen dieses Projektes erzeugte Zeitschrift.

Das wanZine-Projekt beinhaltet einen Online-Editor, in dem Inhalte zusammengestellt, gestaltet und strukturiert werden können, eine Datenbank, in der diese Inhalte und Informationen zu ihrer Gestaltung gespeichert werden, die wanZine-Engine, die aus diesen Informationen unterschiedliche Ausgabeformate generiert sowie eine Internetseite, auf der die so entstandenen Publikationen veröffentlicht werden.

WanZine richtet sich mit diesem Angebot an ein breites Publikum. Durch die Bereitstellung eines Online-Editors und die Distribution der Zeitschriften über das Internet sind die Einstiegshürden für den interessierten Nutzer gering. Er benötigt weder eine spezielle Software, noch fallen Kosten für Produktion und Distribution der Zeitschrift an.

2.1.2. Funktionsweise

Bei wanZine handelt es sich im weiteren Sinne um ein webbasiertes Content Management System (CMS). Unter Content Management versteht man in diesem Zusammenhang die „systematische und strukturierte Beschaffung, Erzeugung, Aufbereitung, Verwaltung, Präsentation, Verarbeitung, Publikation und Wiederverwendung von Inhalten.“[1] Ein Content Management System ist demnach eine Software, die den Nutzer bei der Lösung konkreter Content-Management-Aufgaben unterstützen kann. [2]

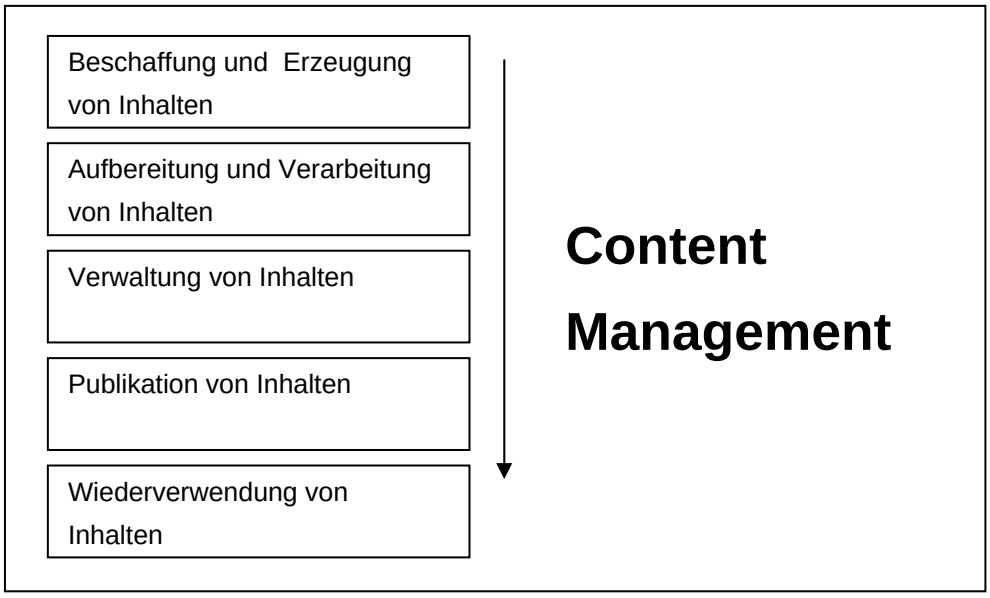

Abbildung 2.1: Grundkonzept des Content Managements

Verglichen mit anderen CMS zeichnet sich wanZine besonders durch ein integriertes Publikations-System aus, mit dem es möglich ist, Inhalte aus einer gemeinsamen Datenbasis für verschiedene Ausgabemedien zu veröffentlichen. Man bezeichnet ein solches Vorgehen bei der Publikation von Inhalten auch als Database Publishing.[3]

---

[1] Rothfuss/Ried 2001, 52

[2] vgl. ebenda

[3] vgl. Beer 2006, http://visualxmag.de, aufgerufen am 02.01.09

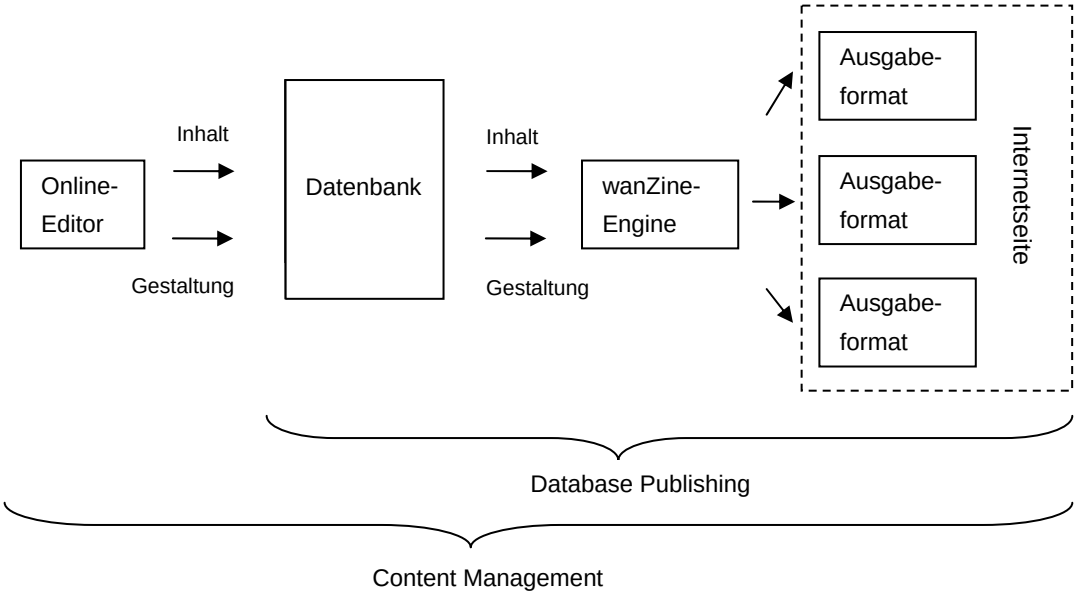

Abbildung 2.2: Grundaufbau des wanZine-Projekts

Der Aufgabenbereich eines klassischen CMS reicht normalerweise nur bis zur Publikation des Endproduktes. Da sich wanZine vor allem an Amateure und semi-professionelle Nutzer richtet, die mit der gezielten Verbreitung ihrer Publikationserzeugnisse wenig vertraut sind, wird die Funktionalität von wanZine auch um Teilaufgaben der eigentlichen Distribution erweitert: Über eine Internetplattform sollen die mit Hilfe des wanZines entstandenen Publikationen veröffentlicht und verbreitet werden. Durch die kostengünstige Online-Vermarktung und einen leicht zu bedienenden Editor ist es jedem möglich, Nischen-Magazine herauszugeben, die aufgrund ihrer geringen Leserzahl im etablierten Zeitschriftenmarkt kaum eine Chance hätten.

Eine zusätzliche Distribution unabhängig vom wanZine-Projekt ist natürlich dennoch möglich, beispielsweise wenn es sich um die gedruckte Ausgaben einer wanZine-Zeitschrift handeln soll.

2.1.3. Verwaltung von Inhalten

Für die Strukturierung und Formatierung von Inhalten stellt wanZine einen Editor zur Verfügung, in dem der Autor Texte, Bilder, Videos, Sounddateien und Multimediaobjekte zusammenstellen, strukturieren und formatieren kann. Dieser Editor ähnelt den Computerprogrammen, die bei der computergestützten Erzeugung von Printmedien (engl. desktop publishing, kurz DTP) zum Einsatz kommen.

Die Speicherung der wanZine-Inhalte erfolgt dabei auf zwei Ebenen: Die Gestaltungsvorgaben werden vom strukturierten Inhalt getrennt gespeichert. Die Trennung des Inhalts von seiner optischen Darstellung ist bereits seit den späten 60er Jahren ein in der Programmierung weit verbreiteter Grundsatz.[4] Durch die logische Trennung von Inhalt und Gestaltung, die sich häufig auch in der Speicherung in separaten Dateien widerspiegelt, lassen sich beide später unabhängig voneinander weiterverarbeiten. Erst im fertigen Publikationsergebnis bilden Inhalt und Formatierung eine Einheit.[5]

Auch im wanZine ist die Trennung des Inhalts von seiner Darstellung von großer Wichtigkeit, um identischen Inhalt mit unterschiedlichen Layouts in verschiedene Ausgabeformate exportieren zu können. Für die Einhaltung dieses Grundsatzes sollte der wanZine-Editor die Struktur des Inhalts deutlich erkennen lassen. Nur so lässt sich die saubere Trennung bei der Speicherung der Daten erreichen.

In gängigen Textverarbeitungsprogrammen erfolgt die Strukturierung eines Dokumentes häufig halbautomatisiert und dabei vom Nutzer unbemerkt über die Formatierung des Textes. Mit Formatvorlagen weist der Nutzer einem Text eine Formatierung, aber gleichzeitig auch eine zuvor festgelegte Stellung in der logischen Dokumentstruktur zu. So wird einer Überschrift

---

[4] vgl. Jendryschik, 35

[5] vgl. Rothfuss/Ried 2001, 10

über die Formatvorlage „Überschrift 1“ eine Formatierung zugewiesen. Gleichzeitig wird sie aber auch als Strukturelement „Überschrift der 1.Ordnung“ ausgezeichnet. Eine solche halbautomatisierte Strukturierung ist für die Strukturierung von wanZine-Inhalten nur eingeschränkt geeignet. Die Tatsache, dass die Strukturierung vom Autor unbewusst vorgenommen wird, führt schnell zu Fehlern.[6] Für die Strukturierung werden deshalb in diesem Fall spezifische Hilfsmittel benötigt, die in klassischer DTP-Software, deren Ausgabe nur auf ein spezielles Ausgabemedium ausgerichtet ist, so nicht zu finden sind. Beispielsweise muss es dem Nutzer im wanZine-Editor möglich sein, die Struktur eines Dokumentes unabhängig vom Layout anzeigen zu lassen, um sie überprüfen und anpassen zu können.

Um die spätere Veröffentlichung in unterschiedlichen Medien zu ermöglichen, ist es nötig, den Inhalt von Beginn an für verschiedene Ausgabemedien aufzubereiten. Besonders wichtig ist dabei die Bereitstellung sog. Alternativinhalte, die benötigt werden, wenn Inhalte in einem der Ausgabemedien nicht in ihrer ursprünglichen Form angezeigt werden können. So werden beispielsweise für die Druckausgabe eines Filmes ein Alternativbild und eventuell auch ein Alternativtext benötigt, um denselben Inhalt auch dann vermitteln zu können, wenn der Film nicht wiedergegeben werden kann. Neben Formatierung und Strukturierung ist es also eine wichtige Aufgabe des Editors, dem Anwender die Möglichkeit zu geben, den Inhalt mit sog. Metadaten zu versehen. Metadaten sind Daten, die die eigentlichen Inhalts-Daten beschreiben und sie um Sekundärinformationen bereichern[7].

[6] vgl. Rothfuss/Ried 2001, 24 f.

[7] vgl. Köhler/Wittenbrink 2003, 69

2.1.4. Veröffentlichung der Inhalte

Mit Hilfe der wanZine-Engine soll der Autor/Herausgeber in der Lage sein, möglichst automatisiert und dadurch zeitsparend, auf derselben Datengrundlage verschiedene Ausgabeformate wie PDF und SWF zu erzeugen. Es handelt sich bei einem wanZine also um eine „Multi-Plattform-Publikation“[8].

Um Zeitschriften in verschiedene Ausgabeformate zu exportieren, ist ein Database Publishing System Bestandteil des wanZine-Projekts. Beim Database Publishing wird der Inhalt strukturiert in einer Datenbank gespeichert. Layouts für die verschiedenen Ausgabemedien werden dann mit Verweisen auf die strukturierten Daten festgelegt.[9] Das Database Publishing System führt diese beiden Datengrundlagen bei der Veröffentlichung zusammen und erstellt daraus die gewünschten Ausgabeformate.

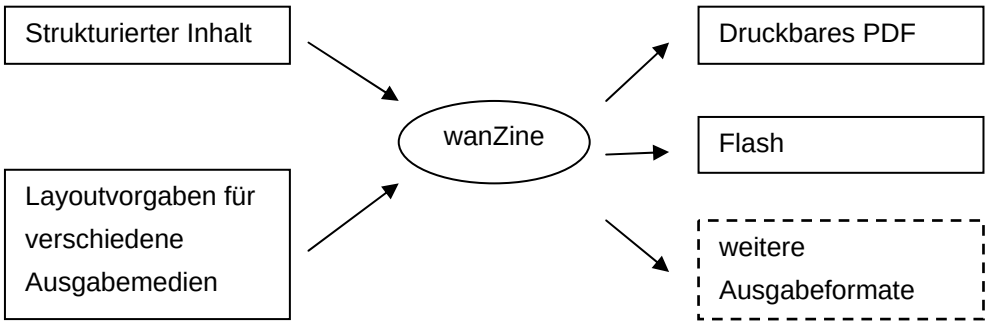

Abbildung 2.3: Schematische Darstellung des Database-Publishings am Beispiel des wanZines[10]

[8] vgl. Rothfuss/Ried 2001, 18

[9] vgl. ebenda

[10] vgl. Beer 2006, http://visualxmag.de, aufgerufen am 02.01.09

Da der Export aus einer gemeinsamen Datenbasis erfolgt, müssen Inhalte nur einmal in das System eingepflegt und ausgezeichnet werden, können später aber mehrfach weiterverwendet werden. Die Erstellung einer solchen wiederverwertbaren Datenbasis ist zwar im Normalfall aufwändiger als die Erstellung eines spezifischen Ausgabemediums, der Mehraufwand ist aber insgesamt geringer einzuschätzen als der Aufwand für die händische Erstellung mehrerer Ausgabeformate.

2.2. Layoutprinzipien

Das Ziel eines Layouts ist es, den zu vermittelnden Inhalt mit Hilfe grafischer Darstellung möglichst gut zu präsentieren. Die optimale Darstellung hängt dabei wesentlich von der Art des Mediums ab, über das die Inhalte transportiert werden sollen. Die für diese Arbeit relevanten Darstellungsmedien sind die gedruckte und die elektronische Ausgabe.

Die Gestaltung von Inhalten für Monitor- und Druckausgabe beruht in vielen Bereichen auf gemeinsamen Grundlagen. Besonders generelle Gestaltungsregeln, die auf den Eigenheiten der menschlichen Sinneswahrnehmung beruhen, behalten auch bei der Ausgabe auf unterschiedlichen Ausgabemedien ihre Gültigkeit: So wird die Empfindung des optischen Gleichgewichts beispielsweise nur unwesentlich vom darstellenden Medium beeinflusst. Ziel ist es für jedes Ausgabeformat eine spannende, optisch ausgewogene Komposition zu schaffen – unabhängig ob das Gesamtergebnis gedruckt oder am Bildschirm ausgegeben werden soll.

Auch Gestaltungsgrundregeln, die auf technischen Ursachen beruhen, finden sich sowohl in Druck- als auch in Online-Medien wieder. In beiden Fällen werden beispielsweise überwiegend rechteckige Formate verwendet. Auch die Gestaltung basiert in den meisten Fällen auf einem rechteckigen Gestaltungsraster, in das sich Fotos, Textblöcke und auch Multimediaobjekte in meist ebenfalls rechteckiger Form einfügen. Durch das Raster erhält der Gestalter ein „normiertes System für Planung und Berechnung“.[11] Nur mit einem festgelegten Raster als Grundlage der Gestaltung erhalten Zeitschriften und Bücher ihr jeweils typisches Aussehen. Auch in Details unterschiedlich gestaltete Seiten wirken durch ein gemeinsames Raster zusammengehörig und harmonisch. Ein festes Gestaltungsraster ist auch im Webdesign unverzichtbar: „Die gleichmäßige Ausrichtung auch der grafischen Elemente und Bilder beruhigt […] den Seitenaufbau und lenkt die Aufmerksamkeit auf den Inhalt“[12]

[11] Radtke/Pisani/Wolters 2004, 57

[12] Lankau 2000, 363

Die Anpassung an die speziellen Eigenheiten des gewählten Mediums ist wichtig, um einen Gesamteindruck zu erreichen, der den Leser zur Lektüre einlädt. Denn „Gestaltungsparameter und -techniken sind nicht einfach von Druck- auf Online-Medien übertragbar“[13] und müssen für jedes Medium einzeln bestimmt werden.

Gestaltungsmerkmale, die sich für Druck- und Monitorausgabe wesentlich unterscheiden und daher besondere Beachtung bei der Publikation für verschiedene Ausgabeformate finden müssen, werden in den folgenden Kapiteln aufgezeigt.

[13] Lankau 2000, 17

2.2.1. Druckausgabe

Formate und Abmessungen

Einen großen Vorteil bei der Gestaltung von Drucksachen bilden die exakten Abmessungen des fertigen Produkts, die von Beginn an bekannt sind. Nur so ist es möglich, das Grundraster und die einzelnen Seiten millimetergenau zu planen und zu definieren. Gedruckte Publikationen sind in der Regel rechteckig, wobei Hochformate überwiegen.[14] Querformate sind weniger verbreitet, aber ebenfalls möglich. Auch Sonderformate können in Einzelfällen eingesetzt werden, sind wegen der höheren Produktionskosten aber relativ selten.

Farbe

Üblicherweise wird im Druckbereich mit dem sog. CMYK-Farbmodell gearbeitet. CMYK steht hierbei für die im Vierfarbdruck verfügbaren Grundfarben Cyan, Magenta, Gelb (engl. Yellow) und Schwarz (Key).[15] Da das vom Papier reflektierte Licht mit jeder hinzugefügten Farbe verringert wird, spricht man auch von „subtraktiver Farbmischung“.

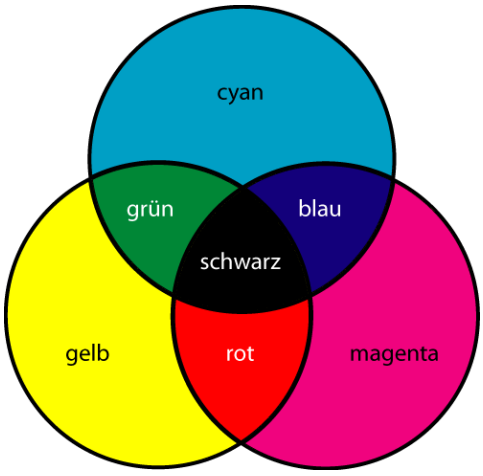

Abbildung 2.4: Subtraktive Farbmischung im CMYK-Modell

[14] vgl. Lankau 2000, 40

[15] vgl. Radtke/Pisani/Wolters 2004, 69

Beim Vierfarbdruck ist zu beachten, dass es durch die Mischung der Farben stets zu leichten Farbabweichungen kommen kann. Ebenso ist die Menge der tatsächlich unterscheidbaren Farben, die aus der Mischung der vier Druckfarben entstehen können, begrenzt. Ist eine besondere Farbtreue, beispielsweise für die Darstellung eines Logos, erwünscht, so kann diese durch die Verwendung von Sonderfarben gewährleistet werden.

Bilder

Wichtig bei der Gestaltung von Druckerzeugnissen ist die Aufbereitung der Bilder. Besonderes Augenmerk gilt hierbei der Auflösung des vorhandenen Bildmaterials. Bilder, die für die Druckausgabe vorgesehen sind, sollten mindestens in einer Auflösung von 300dpi[16] vorliegen. Eine höhere Auflösung ist jedoch von Vorteil, da die tatsächlich druckbare Auflösung im Offsetdruck deutlich höher ist.[17] Eine zu niedrig gewählte Auflösung führt unweigerlich dazu, dass das gedruckte Bild grob aufgerastert aussieht.

Die hohe Auflösung moderner Druckmaschinen ist auch ein Grund dafür, bei der Bildbearbeitung für Druckausgabe möglichst mit nicht komprimierenden Dateiformaten zu arbeiten. Bei der Komprimierung von Bildern werden bewusst Teile der Bildinformation aus der Datei entfernt. Dadurch sinkt zwar die Qualität des Bildes, gleichzeitig wird aber die Dateigröße deutlich verringert. Die Größe der Bilddateien ist bei der Erstellung von Druckerzeugnissen jedoch nur von geringer Relevanz, da diese Bilddateien bei der Auslieferung nicht auf digitalem Wege übertragen werden müssen. Das unkomprimierte TIFF-Format ist für Bilder aus diesem Grund ein häufig verwendetes Speicherformat in der Druckvorstufe.

[16] dots per inch = Bildpunkte pro Zoll

[17] vgl. Radtke/Pisani/Wolters 2004, 140

Typographie

Zusammen mit dem Grundraster ist das Schriftbild ausschlaggebend für den Gesamteindruck eines Druckerzeugnisses. Eine schlecht lesbare Schrift kann einem Leser schnell die Freude und das Interesse an einem Text nehmen.

Aus diesem Grund bietet DTP-Software ein breites Spektrum an Möglichkeiten, auf das Erscheinungsbild der Schrift Einfluss zu nehmen. Der Gestalter kann das Schriftbild über verschiedene Parameter wie Schriftart, Schriftschnitt und Schriftgröße beeinflussen und damit nach seinen Vorstellungen gestalten.

Schriftsatz

Obwohl DTP-Programme ständig weiterentwickelt wurden, ist ein komplett automatisierter Schriftsatz auch heute nicht möglich. „Selbst bei Printpublikationen dürfte der durchgängig automatische Mengensatz stark in der Minderheit sein [...] Mit automatischen Mengensatzmethoden lassen sich keine ästhetisch perfekten Satzergebnisse erzielen.“ [18] Infolgedessen wird auch bei der Gestaltung eines wanZines kein komplett automatisierter Schriftsatz möglich sein.

[18] Rothfuss/Ried 2001, 9

2.2.2. Bildschirmausgabe

Formate und Abmessungen

Eine große Herausforderung bei der Gestaltung von Onlinemedien ist es, den vorhandenen Raum optimal zu gestalten, ohne die Abmessungen der zu gestaltenden Fläche zu kennen. Variable Faktoren wie die Größe des Browserfensters, Bildschirmgröße und -auflösung machen es unmöglich, ein genaues Format vorherzusehen. War in früheren Zeiten immerhin noch ein Monitor mit dem Seitenverhältnis 4:3 zu erwarten, ist selbst diese Annahme angesichts der zunehmenden Verbreitung von 16:9-Bildschirmen kritisch zu bewerten. Gleichgeblieben ist das rechteckige, meist querformatige Grundformat der Monitore.

Ebenso wie bei Druckmedien ist es auch bei der Gestaltung von Onlinemedien sinnvoll, von Beginn an ein Grundraster für die Seitengestaltung anzulegen. Ein solches Raster sollte aus den zuvor genannten Gründen aber wesentlich flexibler und skalierbarer sein als ein vergleichbares Raster bei der Gestaltung von Druckerzeugnissen.

Farbe

Das RGB-Farbmodell, das bei Dokumenten für die Bildschirmausgabe verwendet wird, lehnt sich an die Funktionsweise von Monitoren an. Ein am Bildschirm als weiß wahrgenommener Pixel entspricht je einem Sub-Pixel in den additiven Primärfarben, Rot, Grün und Blau.[19] Je stärker die einzelnen Sub-Pixel leuchten, desto heller wird auch die resultierende Gesamtfarbe wahrgenommen. Man spricht hierbei von der sog. „additiven Farbmischung“.

[19] Radtke/Pisani/Wolters 2004, 67

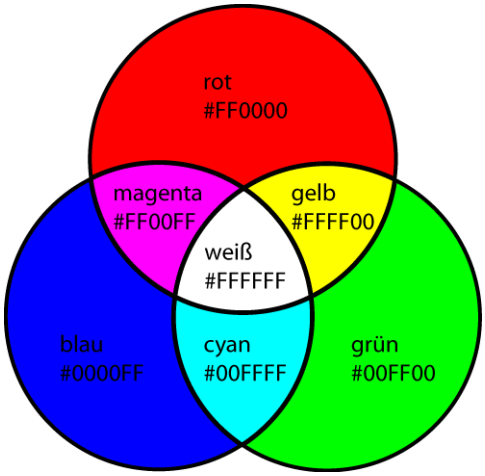

Abbildung 2.5: Additive Farbmischung im RGB-Modell

„Leistungsstarke Monitore können […] wesentlich mehr Farben anzeigen, als im Vierfarbdruck wiedergegeben werden.“[20] Allerdings unterscheiden sich Monitore in der Praxis stark in ihrer Farbwiedergabe. „Eine hundertprozentige Farbtreue auf jedem Gerät ist nicht möglich“[21], so dass das Aussehen eines Layouts von Monitor zu Monitor beträchtlich variieren kann.

Unabhängig davon wirkt sich auch die Funktionsweise von Bildschirmen auf die in einem Layout verwendeten Farben aus. Fast alle Bildschirm-Bauarten besitzen eine hinterleuchtete Monitorscheibe, die ein reines Weiß deutlich strahlender erscheinen lässt als auf Papier. Aus diesem Grund ist das Lesen von schwarzem Text auf weißem Hintergrund am Monitor schnell ermüdend, weil das Weiß die schwarze Schrift überstrahlt und diese so vor den Augen verschwimmt. Es empfiehlt sich bei der Gestaltung von Lesetexten für die Bildschirmausgabe daher die Verwendung von gedämpften Weißtönen als Hintergrundfarbe.[22]

---

[20] Radtke/Pisani/Wolters 2004, 67

[21] ebenda

[22] vgl. Lankau 2000, 368

Bilder

Angesichts einer geringen Monitorauflösung von nur 72 bzw. 96 dpi müssen Bilder für die Monitorausgabe nur in einer deutlich geringeren Auflösung vorliegen, als dies für eine Druckausgabe der Fall wäre. [23]

Ein beschränkender Faktor für die Verwendung von Bildern im Internet ist nach wie vor die Bandbreite bei der Datenübertragung. Auch wenn die Übertragungsgeschwindigkeiten in den letzten Jahren stark gestiegen sind, wären Bildgrößen von mehreren Megabytes, wie sie in der Druckvorstufe üblich sind, nach wie vor undenkbar. Aus diesem Grund werden für die Online-Verwendung häufig komprimierende Verfahren wie JPEG und GIF eingesetzt. Die Datei wird durch die Komprimierung zwar deutlich kleiner, allerdings müssen je nach Kompressionsgrad Verluste bei der Bildqualität in Kauf genommen werden.[24]

Typographie

Das Lesen langer Texte am Monitor ist für das menschliche Auge deutlich anstrengender als das Lesen auf Papier. Aus diesem Grund muss bei der Gestaltung von Onlinemedien die gute Lesbarkeit der Texte stets im Vordergrund stehen.[25]

Dabei ist eines der großen Probleme die geringe Anzahl der verfügbaren Schriften. Bei der Gestaltung von HTML-Seiten ist die Auswahl auf die Geräteschriften beschränkt, die standardmäßig auf den meisten Computern vorinstalliert sind. Soll zur Einhaltung von bestehenden Gestaltungsvorgaben (z.B. im Rahmen von Corporate-Design-Festlegungen) eine andere Schriftart verwendet werden, kann dies nur über die Anzeige des Textes in Form einer Bilddatei oder die Verwendung alternativer Techniken wie PDF oder Adobe Flash erfolgen. Für die Darstellung der so eingebundenen

[23] Radtke/Pisani/Wolters 2004, 140 sowie Lankau 2000, 43

[24] vgl. Lankau 2000, 237 ff.

[25] vgl. Runk 2008, 186

Schriftarten benötigt der Anwender allerdings ein spezielles Zusatzprogramm, ein sog. Plug-In, zur Anzeige am Rechner.[26]

Zusätzlich wird die Auswahl der wirklich nutzbaren Schriften durch das Ziel der guten Lesbarkeit weiter eingeschränkt, da sich einige der standardmäßig vorhandenen Schriftarten nur schlecht für die Darstellung am Monitor eignen. Grotesk-Schriften, also Schriften ohne kleine dekorative Striche an den Enden der Buchstaben, die sog. „Serifen“, sind am Monitor in den meisten Fällen besser lesbar als Serifenschriften.[27]

Auch die sog. typographischen Auszeichnungsarten wie die Verwendung eines kursiven Schriftschnittes oder Kapitälchen eignen sich nur bedingt für die Verwendung bei der Bildschirmanzeige, da diese am Monitor nur schwer lesbar sind. [28] Besser eignet sich bei der Bildschirmausgabe eine farbige Auszeichnung oder die Verwendung eines halbfetten oder fetten Schriftschnittes. Diese Arten der optischen Auszeichnung sollten bei der Druckausgabe hingegen nur sparsam bis gar nicht verwendet werden. [29]

[26] vgl. Runk 2008, 186f

[27] vgl. Lankau 2000, 368 sowie Runk 2008, 193 f.

[28] vgl. Lankau 2000, 366 sowie Runk 2008, 197

[29] vgl. ebenda

Schriftsatz

Viele Computertechniken wie HTML und SWF, mit deren Hilfe Texte variabler Länge automatisiert angezeigt werden können, haben einen entscheidenden Nachteil: Sie bieten keine Möglichkeit zur automatisierten Silbentrennung, was die Möglichkeiten beim Schriftsatz deutlich eingeschränkt.[30]

Die Verwendung von Blocksatz sollte aus ästhetischer Sicht in diesen Formaten auf jeden Fall vermieden werden. Dadurch, dass Wörter nicht automatisiert getrennt werden können, kann der Flatterbereich, also der Bereich, in dem die einzelnen Zeilen enden, sehr groß ausfallen. Müssen die so entstandenen Freiräume am Ende der Zeile für den Blocksatz über Buchstaben- und Wortabstände ausgeglichen werden, entstehen im Text große Lücken und gesperrt geschriebene Zeilen, die das Gesamtbild der Schrift negativ beeinflussen.

Blocksatz ist für Internettechniken ohne automatische Silbentrennung nur schlecht geeignet. Blocksatz ist für Internettechniken ohne automatische Silbentrennung nur schlecht geeignet.

Blocksatz ist für Internettechniken ohne automatische Silbentrennung nur schlecht geeignet. Blocksatz ist für Internettechniken ohne automatische Silbentrennung nur schlecht geeignet.

Abbildung 2.6: Blocksatz mit und ohne Silbentrennung

Für die Lesbarkeit eines Textes am Monitor ist es von Vorteil, wenn die Zeilenlänge kürzer, die Laufweite der Schrift etwas weiter und der Zeilenabstand etwas größer ist als bei einem entsprechenden Druckerzeugnis.[31]

[30] Lankau 2000, 47 f.

[31] vgl. Runk 2008, 189

Anders als im Drucksatz ist im Internet der automatische Schriftsatz weit verbreitet. Aufgrund der unterschiedlichen Darstellung in einzelnen Browsern, skalierbaren Formaten und veränderbaren Schriftgrößen ist ein exakter Schriftsatz wie für die Druckausgabe nicht möglich. Da das Aussehen eines Textes von System zu System ohnehin variiert, werden die Mängel des automatischen Schriftsatzes hier in Kauf genommen.

2.3. Extensible Markup Language XML

XML, die Extensible Markup Language, ist eine Auszeichnungssprache, die es ermöglicht, Daten in Textform strukturiert darzustellen. Das Online-Lexikon von ITWissen definiert die Funktionalität von Auszeichnungssprachen folgendermaßen: Sie „definieren die Bausteine eines Dokuments und legen die Beziehung fest, in denen die einzelnen Dokumente zueinander stehen. Sie sind textbasiert und beschreiben je nach Anwendung den logischen Inhalt von Dokumenten, deren Struktur und Datenaustausch oder werden zur Definition anderer Auszeichnungssprachen benutzt.“ [32]

Es handelt sich bei XML um eine sog. Meta-Auszeichnungssprache. Meta-Auszeichnungssprachen geben Gestaltungsregeln vor, mit denen sich Auszeichnungssprachen für konkrete Anwendungsfälle entwerfen lassen, die den eigenen Bedürfnissen exakt entsprechen.[33] XML wiederum ist ein Anwendungsprofil des W3C-Standards SGML (Standard Generalized Markup Language), bei der es sich um eine Meta-Auszeichnungssprache mit deutlich größerem Funktionsumfang handelt.[34]

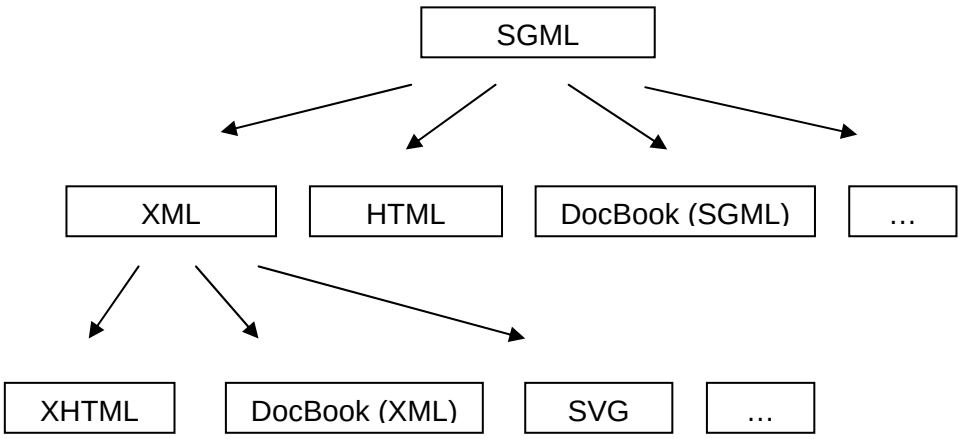

Abbildung 2.7: Übersicht über SGML-basierte Auszeichnungssprachen

[32] ITWissen: Auszeichnungssprache, www.itwissen.info, aufgerufen am 02.01.09

[33] ITWissen: XML (extended markup language), www.itwissen.info, aufgerufen am 17.03.09

[34] vgl. http://www.w3.org/TR/REC-xml, aufgerufen am 31.12.2008

XML-Dateien zeichnen sich dadurch aus, dass sie sowohl für Menschen als auch für Maschinen les- und generierbar sind. Es handelt sich bei XML um eine deskriptive, also beschreibende Markup-Sprache. Das heißt, die Auszeichnung des Dokumentes erfolgt über sog. Tags, kleine Textbausteine, die den Inhalt, den sie umschließen, beschreiben. Diese Textbausteine können frei und passend zum jeweiligen Verwendungszweck gewählt werden, und machen den Inhalt auf diese Weise auch für menschliche Leser verständlich.[35]

XML-basierte Auszeichnungssprachen haben einen weiteren Vorteil gegenüber proprietären Formaten wie PDF (Adobe) oder DOC (Microsoft): „XML ist ein offener Standard, der sich [...] ohne rechtliche Einschränkungen und ohne Lizenzgebühren verwenden lässt.“[36] Gleichzeitig ist der offene Standard XML vom World Wide Web Consortium (W3C) in einer Recommendation, also Empfehlung, standardisiert. So bietet XML für Entwickler ein hohes Maß an Sicherheit bei der Entwicklung, die prinzipielle Austauschbarkeit zwischen verschiedenen Systemen ist garantiert.

Bei der Erstellung einer XML-basierten Sprache sollte die Struktur des so entstehenden neuen Dokumenttyps definiert werden. Dies erfolgt üblicherweise über eine Document Type Definition (DTD) oder ein XML-Schema. Wie bei natürlichen Sprachen spricht man bei einer solchen Sprachdefinition von der Grammatik einer Sprache. [37]

Über DTD oder XML-Schema lässt sich jeder in XML neu generierte Dokumenttyp genau definieren. Auf diese Weise lässt sich auch maschinell überprüfen, ob ein vorliegendes Dokument den Anforderungen einer speziellen DTD oder eines Schemas entspricht und damit valide, also gültig ist.

---

[35] vgl. 2.3. Layoutspeicherung mit Auszeichnungssprachen

[36] Köhler/ Wittenbrink 2003, 39

[37] Köhler/Wittenbrink 2003, 18

Durch Standardisierung und maschinelle Validierbarkeit eignet sich XML gut zum Datenaustausch in verteilten Systemen, wenn unsicher oder unbekannt ist, mit welcher Software die Dokumente verarbeitet werden sollen. Ein XML-Format definiert nicht, mit welchem Programm ein Dokument verarbeitet werden soll, sondern gibt nur Informationen darüber, wie es zu verarbeiten ist.[38]

Mit Techniken wie der Extensible Stylesheet Language (XSL) lassen sich XML-Dateien gestalten und in andere Formate konvertieren.

[38] vgl. Köhler/ Wittenbrink 2003, 39

2.3.1. Aufbau einer XML-Anwendung

Bei der Nutzung von XML ist die Trennung von Inhalt, Struktur und Layout ein wesentlicher Gestaltungsgrundsatz.[39] Aus diesem Grund besteht eine typische Anwendung von XML aus drei Einzelteilen:

- Die eigentliche XML-Datei enthält die strukturierten Inhalte.
- Eine DTD oder wahlweise ein XML-Schema definiert den strukturellen Aufbau der XML-Datei
- Ein Stylesheet (entweder in Form von Cascading Stylesheets (kurz CSS) oder Extensible Stylesheet Language (kurz XSL)) beschreibt die Darstellung der Daten.

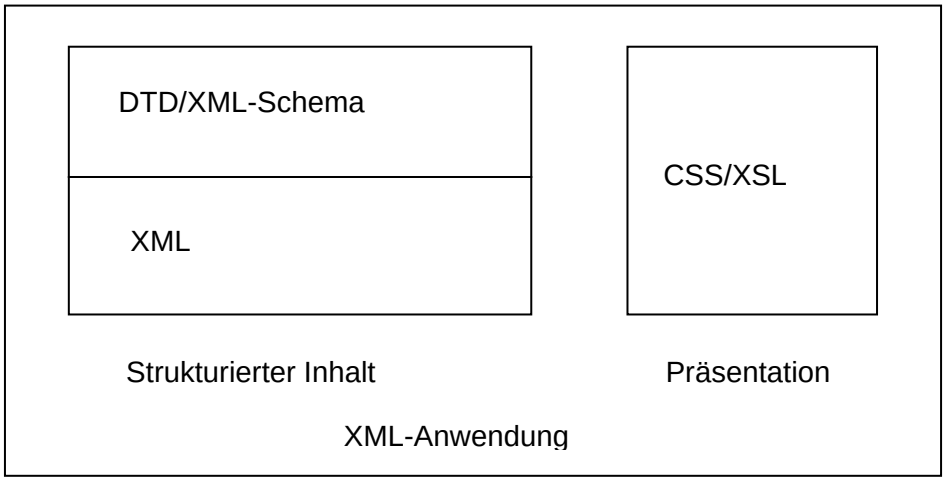

Abbildung 2.8: Aufbau einer typischen XML-Anwendung

[39] vgl. Rothfuss/Ried 2001, 163

2.3.2. Aufbau einer XML-Datei

2.3.2.1. Elemente und Attribute

Innerhalb einer XML-Datei werden die einzelnen Inhaltselemente mit Hilfe von Tags voneinander getrennt. Tags werden in XML durch spitze Klammern (< und >) gekennzeichnet. Zwischen diesen Klammern steht der Name des Elements, der nahezu frei definiert werden kann. Tags treten stets paarweise als Start- und End-Tag auf; zwischen diesen beiden befindet sich der Inhalt eines Elements. Das End-Tag ist dadurch gekennzeichnet, dass zwischen der einleitenden spitzen Klammer und dem Elementtypnamen ein Schrägstrich (/) steht.[40]

```
<element> Inhalt </element>
```

Abbildung 2.9: Beispiel eines einfachen XML-Elements

Der Inhalt eines Elements kann aus Text, untergeordneten Elementen oder einer Kombination dieser beiden Inhaltstypen bestehen. Auch Elemente ohne Inhalt sind möglich. Man spricht dabei von „leeren Elementen“. Für diesen Sonderfall gibt es eine verkürzte Schreibweise, die aus nur einem Tag besteht, das den Namen des Elementes und dahinter einen Schrägstrich enthält.

```
<element>
      Inhalt
      <leeresUnterlement/>
      <unterelement>
            Inhalt
      </unterelement>
</element>
```

Abbildung 2.10: Beispiel eines komplexen XML-Elements

[40] vgl. Schraitle 2004, 12 f.

Neben dem eigentlichen Inhalt, der innerhalb der beiden Tags gespeichert wird, ist es möglich, weitere Informationen über das Element in sog. Attributen zu speichern. Attribute enthalten üblicherweise Informationen, die „eher als ein Aspekt oder eine Eigenschaft des Elements dargestellt werden sollten, und nicht als eine Komponente des Elements.“[41] Attribute dienen also dazu, ein Element näher zu beschreiben. Sie eignen sich aus diesem Grund besonders gut für die Speicherung von Metadaten.[42]

Attribute werden innerhalb des Start-Tags in Form eines Name/Wert-Paares dargestellt, wobei der Attributwert stets in Anführungszeichen steht.[43] Mit Hilfe von Elementen und Attributen lassen sich Inhalte strukturieren und gleichzeitig mit unterschiedlichsten Zusatzinformationen versehen.

```
<element attributname="attributwert">
      Hier steht der Inhalt des Elements.
      <leeresUnterelement/>
</element>
```

Abbildung 2.11: Beispiel eines XML-Elements

[41] Arciniegas 2001, 35

[42] vgl. Köhler/ Wittenbrink 2003, 245

[43] vgl. Arciniegas 2001, 35 sowie Köhler/ Wittenbrink 2003, 99 ff.

2.3.2.2. Struktur

Der Aufbau einer XML-Datei ist strikt hierarchisch. Das End-Tag eines Elementes befindet sich also stets innerhalb desselben Elementes, in dem sich auch sein Start-Tag befindet.[44]

Zulässige Schachtelung von Elementen

```
<element>
      <Unterelement>
      </Unterelement>
</element><
```

Unerlaubte Schachtelung von Elementen

```
<element>
      <Unterelement>
</element>
      </Unterelement>
```

Abbildung 2.12: Schachtelung von Elementen

Dadurch ist von jedem Element innerhalb eines Dokumentes bekannt, zu welchem übergeordneten Element es gehört.[45] Alle Elemente innerhalb eines XML-Dokumentes sind untergeordnete Elemente eines gemeinsamen Wurzelelements, das den gesamten Inhalt umschließt.[46]

[44] vgl. Arciniegas 2001, 31

[45] vgl. Köhler/ Wittenbrink 2003, 96

[46] vgl. Köhler/ Wittenbrink 2003, 32

Für die anschauliche Darstellung der hierarchischen Struktur eines XML-Dokuments wird häufig eine Baumstruktur verwendet.

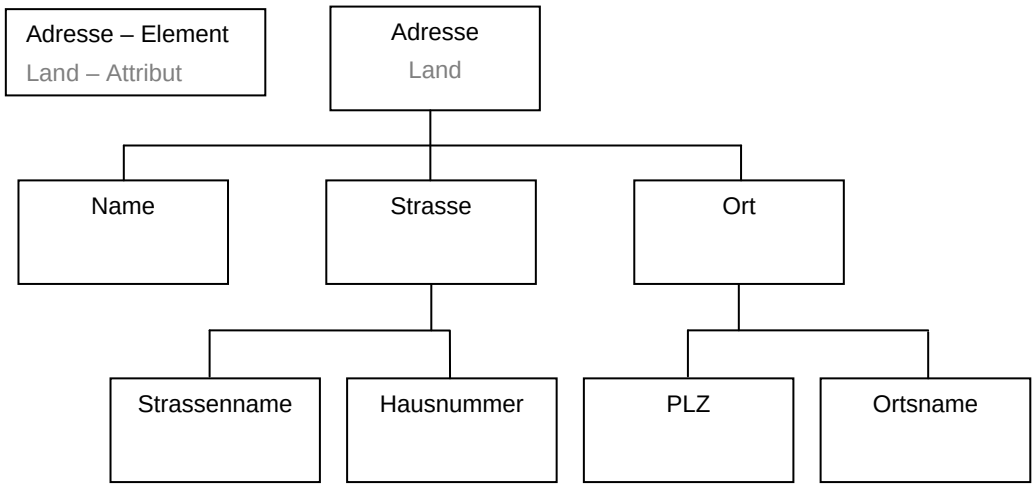

Abbildung 2.13: Baumstruktur einer XML-Datei

2.3.2.3. Kommentare und Processing Instructions

Neben Elementen und Attributen können XML-Dateien auch Informationen beinhalten, die sich direkt an den menschlichen Leser oder die verarbeitende Maschine richten.[47]

Kommentare beinhalten Informationen, die dem menschlichen Leser das Verständnis einer XML-Datei erleichtern. Sie werden mit der Zeichenfolge <!-- eingeleitet und mit --> beendet.[48] Processing Instructions enthalten Informationen für die Anwendung, die ein XML-Dokument verarbeiten soll. Sie werden durch die Zeichenfolgen <? und ?> gekennzeichnet.[49]

[47] vgl. Köhler/ Wittenbrink 2003, 101 ff.
[48] vgl. Köhler/ Wittenbrink 2003, 101 f.
[49] vgl. Köhler/ Wittenbrink 2003, 103 f.

2.3.3. Document Type Definition

Eine Document Type Definition (DTD) beschreibt das Aussehen einer Klasse von Dokumenten desselben Typs. In der DTD wird festgelegt, welche Elemente und Attribute eine XML-Datei dieses Typs enthalten darf, und wie diese Elemente angeordnet sein dürfen.[50] DTDs verwenden hierbei eine eigene Syntax, die nicht XML-konform ist.[51]

Ein Bestandteil jeder DTD ist die Elementtypdeklaration, in der für jedes Element festgelegt wird, welche Art von Inhalt es beinhalten und wie oft es auftreten darf. Die Möglichkeiten, die Häufigkeit des Auftretens einzelner Elemente zu beschreiben, sind dabei allerdings auf nur vier Aussagen beschränkt: Das Element kann entweder „genau einmal“, „null oder einmal“, „ein oder mehrere Male“ oder „null, ein oder mehrere Male“ auftreten. Eine Festlegung, dass ein Element mindestens zwei Mal auftreten muss, ist mit DTD beispielsweise nicht möglich.

Die zulässigen Inhalte, die einem Element mit Hilfe einer DTD zugewiesen werden können, sind Text oder weitere Elemente. Auch ein gemischter Inhalt aus Text und Elementen ist möglich.

In der Elementtypdeklaration einer DTD kann allerdings nicht festgelegt werden, welche Art von Text ein Element enthalten kann, so wie es in anderen Programmiersprachen üblich wäre. Textinhalte werden in einer DTD stets als #PCDATA bezeichnet, unabhängig davon, ob es sich bei dem Inhalt um eine Zahl des Typs Integer oder einen Fließtext handelt.[52]
So ist eine XML-Datei mit einem Element wie `<preis>Max Mustermann</preis>` valide, wenn in der DTD #PCDATA als möglicher Inhalt für das Element `<preis>` festgelegt wurde. Ob es sich bei dem Inhalt des Elements `<preis>` tatsächlich um die erwartete Zahl handelt, oder wie hier fälschlicherweise um einen Namen, lässt sich bei der Verwendung einer

[50] vgl. Arciniegas 2001, 49 sowie vgl. Köhler/ Wittenbrink 2003, 211

[51] vgl. Köhler/ Wittenbrink 2003, 514

[52] vgl. Arciniegas 2001, 51

DTD nicht verifizieren.[53] Eine DTD ist wegen dieser Ungenauigkeit unter Umständen nicht exakt genug, eine Klasse von XML-Dateien hinreichend genau zu beschreiben. Besonders, wenn es sich bei dem Inhalt eines Elements um speziell formatierten Text handeln soll, ist eine DTD ungeeignet.

Neben der Definition aller gültigen Elemente enthält eine DTD auch eine Deklaration der Attribute, die die einzelnen Elemente beinhalten dürfen oder müssen. In der sog. Attributlisten-Deklaration wird der Typ der einzelnen Attribute definiert. Allerdings gibt es hier ebenso wie in der Elementtypdeklaration für alle Zeichenketten nur einen Datentyp: CDATA. Daneben gibt es weitere Attributtypen wie beispielsweise ID oder ENTITY sowie die Möglichkeit, alle verfügbaren Attributwerte aufzulisten.[54]

Den einzelnen Attributen werden zuletzt noch Attributvorgaben zugewiesen. Mit Hilfe dieser Vorgaben wird festgelegt, ob ein Attribut vorkommen muss oder kann, es einen Standardwert hat oder nur einen bestimmten Wert annehmen kann.

```
<?xml version="1.0" encoding="utf-8"?>
<!ELEMENT Adresse (Name, Strasse, Ort)>
<!ELEMENT Name #PCDATA>
<!ELEMENT Strasse (Strassenname, Hausnummer)>
<!ELEMENT Strassenname #PCDATA>
<!ELEMENT Hausnummer #PCDATA>
<!ELEMENT Ort (PLZ, Ortsname)>
<!ELEMENT PLZ #PCDATA>
<!ELEMENT Ortsname #PCDATA>
<!ATTLIST Adresse Land (DE|AT|CH) #REQUIRED >
```

Abbildung 2.14: Beispiel einer Document Type Definition (DTD)

[53] vgl. Arciniegas 2001, 61 f.

[54] vgl. Sebestyen 2005, 81 ff.

2.3.4. XML-Schema

Durch die technischen Entwicklungen der letzten Jahre wurden neue Anforderungen an die Validierbarkeit von XML-Dateien gestellt, die mit DTDs nur unzureichend erfüllt werden konnten.[55] Aus diesem Grund wurde XML-Schema vom W3C standardisiert, das häufig als Nachfolger für DTD propagiert wird.[56]

Aufgrund der Komplexität von XML-Schema kann hier nur ein kurzer Überblick über die wichtigsten Eigenschaften verschafft werden.

Die Grundfunktionalität von XML-Schema und DTD ist sehr ähnlich: Beide beschreiben letztendlich die Beschaffenheit von Elementen und Attributen innerhalb einer XML-Datei. XML-Schema ist jedoch durch seine Ausrichtung an Konzepten der Objektorientierung der deutlich leistungsfähigere Mechanismus.[57] Anders als DTD basiert XML-Schema zudem auf der XML-Syntax und ist damit auch selbst validierbar.

Mit XML-Schema ist es möglich, genau zu definieren, wie häufig und in welcher Reihenfolge einzelne Elemente auftreten können. Außerdem ermöglicht XML-Schema eine wesentlich genauere Datentypisierung als eine DTD. Statt des einen Datentyps #PCDATA, der in DTDs zur Beschreibung von Texten verfügbar ist, bietet XML-Schema mehr als 30 einfache Datentypen, die nach eigenen Bedürfnissen weiter angepasst und damit beliebig erweitert werden können.

---

[55] vgl. Behme/Mintert: 21.1 XML-Schema, http://www.linkwerk.com, aufgerufen am 17.03.2009

[56] vgl. Köhler/ Wittenbrink 2003, 510 sowie
http://www.w3.org/TR/2004/REC-xmlschema-0-20041028/,
http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/,
http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/,
aufgerufen am 13.01.2009

[57] vgl. Arciniegas 2001, 311

Der Zugewinn an Funktionalität spiegelt sich allerdings auch in einer erhöhten Komplexität wider. XML-Schemata haben bei der Beschreibung derselben XML-Struktur einen deutlich größeren Umfang als eine DTD. Das folgende Beispiel definiert dieselbe XML-Struktur wie die DTD in Abb. 3.10.

```
<?xml version="1.0" encoding="utf-8" ?>
<xs:schema    elementFormDefault="qualified"
              xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="Adresse">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Name" type="xs:string"/>
        <xs:element name="Strasse">
          <xs:complexType>
            <xs:sequence>
              <xs:element   name="Strassenname"
                                    type="xs:string"/>
              <xs:element   name="Hausnummer"
                                    type="xs:positiveInteger"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="Ort">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="PLZ" type="PLZType"/>
              <xs:element name="Ortsname" type="xs:string"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="Land">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:enumeration value="DE"/>
            <xs:enumeration value="AT"/>
            <xs:enumeration value="CH"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
    </xs:complexType>
  </xs:element>
  <xs:simpleType name="PLZType">
    <xs:restriction base="xs:positiveInteger">
      <xs:fractionDigits value="0"/>
      <xs:totalDigits value="5"/>
    </xs:restriction>
  </xs:simpleType>
</xs:schema>
```

Abbildung 2.15: Beispiel eines XML-Schemas

2.4. Strukturierte Speicherung von Inhalten

Seit der Entstehung des Desktop Publishings besteht das Bedürfnis, die am Computer gestalteten Publikationen eindeutig beschreiben und damit rekonstruierbar speichern zu können. Im Laufe der Jahre sind zu diesem Zweck vielfältige Auszeichnungssprachen entstanden.

Auszeichnungssprachen nutzen eine von zwei grundlegenden Arten zur Auszeichnung von Inhalten (engl. markup). Je nach Anwendungsgebiet ist die prozedurale oder deskriptive Auszeichnung besser geeignet. Der wesentliche Unterschied zwischen diesen beiden Paradigmen der Dokumentauszeichnung besteht vor allem darin, dass prozedurales Markup nicht das Dokument an sich beschreibt, sondern vielmehr eine Änderung des Zustandes im verarbeitenden System.[58]

Typische Beispiele für die prozedurale Auszeichnung sind die Druckersteuersprachen PCL und PostScript, die einen Ablauf verschiedener Vorgänge während des Druckvorgangs speichern. So wird nicht gespeichert, wie eine Seite aussieht, sondern wie sich der Druckkopf bewegen muss, um diese Seite auszudrucken. Daran wird auch ein Nachteil des prozeduralen Markups deutlich: Es beinhaltet keinerlei Aussagen über die Struktur innerhalb des Dokumentes. Auch zusätzliche Informationen, die für eine Weiterverarbeitung in einem anderen Kontext als dem vorgesehenen notwendig wären, sind in der Datei nicht enthalten. Der Verarbeitungsprozess ist also bereits mit der Auswahl eines prozeduralen Vokabulars festgelegt.[59] Eine Konvertierung in andere Ausgabeformate ist nur schwer möglich.

Im Gegensatz dazu sind deskriptive Auszeichnungssprachen dadurch gekennzeichnet, dass sie Inhalte entsprechend ihres Typs mit aussagekräftigen, häufig in der Definition der Auszeichnungssprache bereits vordefinierten Tags beschrieben werden. „Einzelne Dokumentbestandteile werden

---

[58] vgl. Köhler/ Wittenbrink 2003, 31

[59] vgl. Kutscher/Ott/Bormann/Bergmann: 6.1.Auszeichnung von Dokumentbestandteilen, http://www.teialehrbuch.de, aufgerufen am 24.11.2008

entsprechend ihrer Bedeutung innerhalb des jeweiligen Anwendungsgebietes ausgezeichnet und so vollständig vom Prozeß der Ausgabeerzeugung entkoppelt."[60] Durch die strukturierte Speicherung unabhängig vom Ausgabeformat ist bei der Verwendung von deskriptivem Markup eine Konvertierung in andere Formate wesentlich leichter möglich.

Typisch für deskriptives Markup ist die Verwendung von Tags zur Auszeichnung. Das Vokabular, das dabei verwendet wird, kann der Terminologie des Anwendungsgebietes angepasst werden und ist auf diese Weise leicht verständlich und nachvollziehbar.[61]

Neben der Unterscheidung zwischen deskriptiver und prozeduraler Auszeichnung lassen sich Auszeichnungssprachen auch danach klassifizieren, ob die Lage einzelner Layoutelemente relativ zueinander oder absolut zum Seitenformat gespeichert wird. Sowohl die relative als auch die absolute Speicherung von Layoutelementen finden in der Praxis Anwendung, wie in den folgenden Kapiteln anhand von konkreten Beispielen kurz aufgezeigt wird. Hierbei soll jedoch der Fokus vor allem auf der Speichermethode und den daraus resultierenden Vor- und Nachteilen liegen. Der Funktionalitätsumfang des jeweiligen Formats findet in dieser Arbeit keine Berücksichtigung.

---

[60] Kutscher/Ott/Bormann/Bergmann: 6.2. Prozedurale Auszeichnungssprachen, http://www.teialehrbuch.de, aufgerufen am 24.11.2008

[61] vgl. ebenda

2.4.1. Flussorientierte Dokumentauszeichnung

Bei der flussorientierten Speicherung von Layouts wird die Lage der Inhaltselemente anhand ihrer relativen Position zueinander gespeichert. Am Beispiel der deskriptiven Auszeichnungssprachen HTML und DocBook wird im folgenden Kapitel die Funktionsweise der flussorientierten Dokumentauszeichnung erklärt.

2.4.1.1. Hypertext Markup Language HTML

Bei HTML, der Hypertext Markup Language, handelt es sich um eine der weltweit am häufigsten genutzten Auszeichnungssprachen. Ihre weite Verbreitung verdankt sie vor allem dem stetig wachsenden Internet, für das sie eine wichtige Grundlage bildet. [62] Für die Erstellung einer Internetseite ist HTML unverzichtbar.

Da in HTML nur ein sehr beschränktes Repertoire an Tags definiert ist, ist sie ein schnell erlernbares und dadurch häufig gebrauchtes Werkzeug.[63] Mit Hilfe von sog. WYSIWYG-Editoren[64], die die Erstellung der HTML-Dateien über eine Vorschau besonders einfach machen, ist es selbst Laien nach einer kurzen Einarbeitungszeit möglich, eigene HTML-Dateien zu erstellen.[65]

Eine der wesentlichen Eigenschaften von HTML, die je nach Verwendungszweck zum Vorteil oder Nachteil werden kann, ist die fließende Seitenbeschreibung. In mit HTML ausgezeichneten Dokumenten werden lediglich „relative Bezüge und voneinander abhängige Bedingungen [...] aufgestellt.“[66]

---

[62] vgl. Kutscher/Ott/Bormann/Bergmann: 6.1.Auszeichnung von Dokumentbestandteilen, http://www.teialehrbuch.de , aufgerufen am 24.11.2008

[63] vgl. Schraitle 2004, 10

[64] „what you see is what you get“ ,dt.: „du siehst das, was du bekommst“

[65] vgl. Lankau 2000, 61 f.

[66] Lankau 2000, 77

Durch diese Speicherung relativer Verhältnisse der einzelnen Layoutelemente ist es möglich, die Anzeige flexibel anzupassen, z.B. wenn vom Betrachter eine größere Schriftgröße gewünscht ist, oder die Bildschirmauflösung nur eine kleine Darstellung zulässt. Eben diese Flexibilität verhindert aber auch, dass für alle Ausgabesysteme ein identisches Aussehen garantiert werden kann. Eine Internetseite kann bereits durch die Verwendung von verschiedenen Browsern oder unterschiedlich großen Browserfenstern anders aussehen.

Bei HTML handelt es sich um eine reine Darstellungssprache, die dazu dient, „einem Browser zu sagen wie er eine Seite anzeigen soll, während er über die [inhaltliche] Struktur der Daten auf der Seite nichts erfährt.“[67] Reines HTML entspricht also nicht dem Grundsatz von Trennung von Inhalt und Präsentation.

Eine Mischform zwischen Beschreibung von Präsentation und Strukturierung stellen die Tags h1…h6 dar. Sie dienen zur Auszeichnung von Überschriften verschiedener Ebenen, bieten damit aber gleichzeitig auch eine Möglichkeit zur groben Strukturierung. Eine über diese vorhandenen Tags hinausgehende Strukturierung ist jedoch nicht möglich. In folgendem Beispiel kann man nicht automatisch erkennen, bei welchem der Elemente es sich um den Ortsnamen handelt, auch wenn dies für den menschlichen Betrachter aufgrund der für Adressen typischen Formatierung offensichtlich ist.

Quellcode:

```
<p>
Max Mustermann<br>
Musterstraße 1<br>
01234 Musterstadt
</p>
```

Browseransicht:

Max Mustermann
Musterstraße 1
01234 Musterstadt

Abbildung 2.16: Gegenüberstellung von HTML-Quellcode und Browseransicht

[67] Arciniegas 2001, 25

Ebenso wie XML und viele weitere Auszeichnungssprachen ist HTML eine Untermenge von SGML.[68] Über CSS (Cascading Style Sheets) lässt sich das Aussehen des mit HTML ausgezeichneten Dokumentes darüber hinaus detailierter gestalten. Durch die Verknüpfung von HTML mit CSS lässt sich so eine saubere Trennung von Inhalt und Präsentation erreichen.

[68] siehe auch 2.3 Extensible Markup Language XML

2.4.1.2. DocBook

DocBook ist eine der meistverbreiteten nicht proprietären Auszeichnungssprachen zur Erstellung von Druckerzeugnissen. Besonders für die Gestaltung von Büchern, Artikeln und anderen Prosadokumenten ist DocBook gut geeignet. Vor allem technische Dokumentationen werden häufig in DocBook gelayoutet, was in der Entstehungsgeschichte von DocBook begründet liegt: Als DocBook im Jahr 1991 entwickelt wurde, sollte es vor allem den Austausch der verschiedenen UNIX-Dokumentationen für die Entwickler vereinfachen.[69] Seitdem konnte sich DocBook aber auch in anderen Bereichen als fester Standard etablieren.

Neben der ursprünglichen DocBook Definition auf Basis der Metaauszeichnungssprache SGML gibt es seit einigen Jahren auch eine verschlankte, XML-basierte Version. Beide Varianten werden nach wie vor genutzt und weiterentwickelt.[70]

DocBook eignet sich besonders, „wenn aus dem Ursprungsdokument mehr als ein Format erzeugt werden soll“,[71] da es mit Hilfe unterschiedlicher Transformations-Werkzeuge möglich ist, die DocBook-Vorlage in verschiedene Ausgabeformate wie HTML, PDF oder RTF umzuwandeln.

Es existieren in einem DocBook zwei Arten von Elementen. Strukturgebende Elemente wie Kapitel und Textabschnitte dienen einzig der Strukturierung des Inhalts. Inhaltsbezogene Elemente dienen beispielsweise zur Hervorhebung von Texten und Einbindung von Bildern und haben daher einen direkten Bezug zum untergeordneten Inhalt.[72] Die Trennung in strukturgebendes und inhaltsbezogenes Markup ist einer der großen Vorteile bei der Verwendung von DocBook.[73] Durch diese Trennung ist es möglich,

---

[69] vgl. Walsh/Muellner 1999, 13 f.

[70] http://www.docbook.org/schemas/, aufgerufen am 29.12.08

[71] Schraitle 2004, 87

[72] vgl. Köhler/ Wittenbrink 2003,298

[73] vgl. ebenda

auch maschinell die Struktur eines Dokumentes auszulesen und beispielsweise automatisiert Inhaltsverzeichnisse zu generieren, wie es bei anderen Auszeichnungssprachen wie HTML nicht möglich ist.[74]

Ein Nachteil von DocBook ist allerdings, dass es ebenso wie HTML ursprünglich für die Auszeichnung von wissenschaftlichen und technischen Texten entwickelt wurde.[75] Die verfügbaren Layoutoptionen sind daher auf diesen speziellen Anwendungsfall zugeschnitten. „Für Dokumente mit sehr komplexen Layout ([...] Zeitschriften u.a.) ist DocBook weniger geeignet.“[76] In Anwendungsfällen, in denen ein ästhetisch ansprechendes Layout eine große Rolle spielt, ist eine traditionelle DTP-Software mit grafischer Nutzeroberfläche dem textbasierten DocBook überlegen.

[74] vgl. Köhler/ Wittenbrink 2003, 301

[75] vgl. Kutscher/Ott/Bormann/Bergmann:12.1 Die DocBook-DTD, http://www.teialehrbuch.de, aufgerufen am 24.11.2008

[76] Schraitle 2004, 87

2.4.2. Positionsorientierte Dokumentauszeichnung

Es existieren auch Auszeichnungssprachen, welche die einzelnen Gestaltungselemente unabhängig von ihrer relativen Position innerhalb des Inhalts über absolute Koordinaten angeben. In diesem Kapitel werden zwei solche Formate beispielhaft vorgestellt, in denen Inhalte positionsorientiert gespeichert werden.

2.4.2.1. Portable Document Format PDF

Das Portable Document Format PDF ist seit seiner Veröffentlichung im Jahr 1993 zu einem Quasi-Standard für den Dokumentenaustausch über das Internet geworden. Besonders im Bereich der Druckvorstufe ist das PDF-Format bei der Übertragung eines fertigen Layouts an die ausführende Druckerei weit verbreitet. Dieser Entwicklung wurde Rechnung getragen, indem das vormals zwar offengelegte, aber noch proprietäre Dateiformat PDF als offener Standard in der Norm ISO 32000-1 definiert wurde.[77]

Die weite Verbreitung des PDF-Formates liegt vor allem darin begründet, dass das Aussehen einer Seite damit exakt gespeichert werden kann. Im Gegensatz zu anderen Auszeichnungssprachen ist das Aussehen einer PDF-Datei unabhängig vom Ausgabemedium. Seiteninhalte werden sowohl am Bildschirm als auch auf Papier identisch wiedergegeben.[78] Grundlage dafür ist das Adobe Imaging Model. Dieses von Adobe entwickelte Grafikmodell zeichnet sich im Zusammenhang mit der Zielstellung dieser Arbeit insbesondere durch seine Geräteunabhängigkeit aus. Das Adobe Imaging Model beschreibt die Seite abstrakt, ohne jedoch Bezug auf die Eigenschaften eines bestimmten Ausgabegerätes zu nehmen. Erst bei der Ausgabe wird die gespeicherte Information durch einen sog. Interpreter für das

---

[77] vgl. Adobe Systems Incorporated: Document management — Portable document format — Part 1: PDF 1.7. First Edition. S.VI
http://www.adobe.com/devnet/pdf/pdf_reference.html, aufgerufen am 13.11.2008

[78] vgl. Merz/Drümmer 2002, 609

aktuelle Ausgabegerät aufbereitet. Im Falle von PDF übernimmt die Acrobat-Software die Rasterung der Inhalte.[79]

Die interne Struktur eines PDF-Dokuments basiert auf ähnlichen Grundregeln wie auch HTML, SGML und XML. Zu nennen sind hierbei beispielsweise die hierarchische Struktur des Dokumentes und die Trennung in Struktur und Layout.[80]

Die Seitenbeschreibung innerhalb des PDF-Dokumentes erfolgt mit Hilfe eines Datenstroms oder mehrerer Datenströme, den sog. Content-Streams. Ein Content-Stream enthält Operatoren für Text, Vektor- und Rastergrafik und die zugehörigen Koordinatenangaben, über die der Inhalt positioniert werden kann.[81] Die Inhalte innerhalb der PDF-Datei werden unabhängig von der Seitenreihenfolge oder der Anordnung des Inhalts auf der Seite gespeichert.[82] Auch Texte werden bei diesem Vorgehen in einzelne Teilblöcke aufgebrochen. Ein solcher Teilblock kann dabei aus Einzelzeichen, Silben, Wörtern oder ganzen Zeilen und Absätzen bestehen. Aus diesem Grund sind Textblöcke häufig „in viele einzelne Fragmente zerrissen“, wenn man später versucht, sie innerhalb einer PDF-Datei auszuwählen.[83]

Die Speicherung in Einzelteilen ermöglicht eine Wiedergabe von einzelnen Seiten eines Dokumentes, ohne zuvor die Anzeige aller vorangegangenen Seiten abwarten zu müssen, wie es bei einer sequentiellen Speicherung notwendig wäre. PDF ermöglicht also einen sog. random-access, den di-

---

[79] vgl. Merz/Drümmer 2002, 6 f.

[80] vgl. Adobe Systems Incorporated: Document management — Portable document format — Part 1: PDF 1.7. First Edition. S.556
http://www.adobe.com/devnet/pdf/pdf_reference.html, aufgerufen am 13.11.2008

[81] vgl. Merz/Drümmer 2002, 545

[82] vgl. Merz/Drümmer 2002, 523

[83] vgl. Merz/Drümmer 2002, 546

rekten Zugang zu einzelnen Dokumentbestandteilen, ohne dabei eine vorgeschriebene Zugriffs-Reihenfolge einhalten zu müssen.[84]

Bei der Positionierung der einzelnen Inhaltsblöcke arbeitet PDF mit dem gleichen Standardkoordinatensystem, das auch PostScript verwendet. Es basiert auf der Einheit Punkt[85] und hat seinen Koordinatenursprung in der linken unteren Ecke. Die Positionen einzelner Elemente werden mit Hilfe dieses Koordinatensystem bestimmt.[86]

---

[84] vgl. Adobe Systems Incorporated: PDF Reference. Sixth Edition. Adobe Portable Document Format. Version 1.7. 11/2006, S.151
http://www.adobe.com/devnet/acrobat/pdfs/pdf_reference_1-7.pdf ,
aufgerufen am 08.01.2009

[85] Im DTP ist ein Punkt als das 72stel eines Zolls definiert , entspricht also 0,353mm, vgl. Runk 2008, 79

[86] vgl. Merz/Drümmer 2002, 539 f.

2.4.2.2. Scalable Vector Graphics SVG

Scalable Vector Graphics ist eine der - technisch gesehen - wichtigen Neuentwicklungen der letzten Jahre. Bei der Standardisierung von SVG durch das W3C im Jahr 2001 wurde SVG noch als Revolution im Bereich der Web-Grafik gefeiert. So sollten Flash-Dateien, Rastergrafiken, PDF und PostScript-Dateien durch SVG ersetzt werden können. Bislang konnte sich der SVG-Standard aber nicht in dem erwarteten Ausmaß durchsetzen, auch wenn SVG inzwischen von allen wichtigen Internetbrowsern auch ohne zusätzliches Plug-In unterstützt wird.

SVG ist eine Sprache zur Beschreibung von zweidimensionalen Vektorgrafiken. Auch kleinere Animationen wie Farbänderungen, Bewegungen entlang eines Pfades und das Drehen und Kippen von Elementen sind mit SVG möglich.[87] Dabei ist einer der großen Vorteile von SVG (und anderen Vektorgrafiken) der geringe benötigte Speicherplatz im Vergleich zu Rastergrafiken wie JPEG und GIF.[88] Außerdem sind Vektorgrafiken ohne Qualitätsverlust beliebig skalierbar.[89]

SVG basiert im Gegensatz zum PDF-Format auf dem XML-Standard. [90] Anders als viele andere XML-Anwendungen dient SVG aber nicht zur strukturierten Speicherung von Textinhalten, sondern zur Beschreibung von Vektorgrafiken. Vektoren werden dabei nicht binär gespeichert, sondern in Textform beschrieben, so dass Informationen auch mit einem Texteditor erstellt und bearbeitet werden können. [91]

SVG-Dateien verwenden zur Layoutspeicherung ebenso wie PDF das Adobe Imaging Model.[92] Auch im SVG-Format werden die Grafiken unab-

---

[87] vgl. Spona 2001, 16 f.

[88] vgl. Spona 2001, 11 sowie vgl. Köhler/ Wittenbrink 2003, 178

[89] vgl. Köhler/ Wittenbrink 2003, 178

[90] siehe auch Kapitel 2.3 Extensible Markup Language XML

[91] vgl. Spona 2001, 11

[92] Merz/Drümmer 2002, 6 f.

hängig vom Ausgabegerät gespeichert. Wie PDF speichert SVG die einzelnen Grafik-Elemente, indem es ihre absolute Position in Form von dinaten eines virtuellen Koordinatensystems speichert.[93] Die gespeicherte Position bezieht sich dabei nicht auf ein feststehendes Ausgabeformat, sondern auf die jeweils linke obere Ecke des Darstellungsbereichs.[94]

Quellcode: **Browseransicht:**

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 20001102//EN"
"http://www.w3.org/TR/2000/CR-SVG-20001102/DTD/svg-20001102.dtd">
<svg width="140" height="130">
    <rect   x="35" y="20" width="45"
            height="40" style="fill:rgb(255,0,0)"/>
    <circle cx="70" cy="60" r="20"
            style="fill:rgb(0,100,255)"/>
    <rect   x="65" y="65" width="45"
            height="40" style="fill:rgb(255,255,0)"/>
</svg>
```

Abbildung 2.17: Gegenüberstellung von SVG-Quellcode und Browseransicht

Für die Beschreibung der Grafiken gibt es für Vektor-Grundformen wie Rechtecke und Kreise vordefinierte Tags, die über Attribute nach eigenen Vorstellungen gestaltet werden können. Ein `text`-Tag ermöglicht außerdem die Darstellung von Texten.[95] Freie Formen lassen sich darüber hinaus über Pfade und Bezierkurven erstellen.[96] Ihre Erstellung ist aber ohne die Hilfe eines grafischen SVG-Editors schwierig und unübersichtlich.

[93] vgl. Köhler/ Wittenbrink 2003, 177

[94] vgl. Köhler/ Wittenbrink 2003, 199

[95] vgl. Spona 2001, 43 ff.

[96] vgl. Spona 2001, 11

2.5. Fazit

Folgende Schlussfolgerungen lassen sich aus dem vorangegangen Kapitel für die Erstellung einer Struktur zur Inhaltsspeicherung für das wanZine-Projekt ziehen:

Ein gutes Layout lässt sich nicht automatisiert erstellen, da keine universellen Regeln für die Erstellung eines geeigneten Layouts existieren. Das ästhetische Empfinden eines menschlichen Betrachters wird nicht allein durch rational begründbare Aspekte beschrieben, sondern beinhaltet auch die emotionale Wirkung eines Layouts. Dies lässt sich bisher nicht mathematisch beschreiben. Da es keine Formeln und Algorithmen gibt, mit deren Hilfe sich auf jeden Fall ein optimales Ergebnis erzielen lässt, ist eine komplett automatisierte Erstellung eines Layouts praktisch unmöglich. Ein computergeneriertes Layout bedarf daher in den meisten Fällen einer menschlichen Überarbeitung und Optimierung. Ein Computer(-Programm) kann also einen menschlichen Designer nur im Gestaltungsprozess unterstützen, da die menschliche Wahrnehmung letztendlich der Maßstab für die Qualität eines Layouts sein muss.

Für das wanZine bedeutet dies, dass es nur mit Einschränkungen möglich ist, aus einem Layout für ein bestimmtes Ausgabemedium automatisiert ein fehlerloses, optisch ansprechendes Layout für weitere Ausgabeformate zu generieren. Ein grobes Grundlayout kann generiert werden, bedarf aber einer menschlichen Überarbeitung.

Dies liegt unter anderem daran, dass sich Layouts für Monitor- und Druckausgabe nicht in allen, aber einigen wesentlichen Punkten unterscheiden. Ein Online-Layout lässt sich daher nicht eins zu eins in ein Druck-Layout übertragen. Ebenso wenig funktioniert eine direkte Transformation in die entgegengesetzte Richtung. Aus diesem Grund und aufgrund der Erkenntnis, dass automatisch generierte Layouts nur selten ästhetischen Ansprüchen genügen, müssen separate Gestaltungsvorgaben für einzelne Ausgabemedien erstellt und gespeichert werden.

Auch die im wanZine verwendeten Bilder müssen für die verschiedenen Ausgabemedien angepasst werden. Da vielen potentiellen Nutzern das Bewusstsein für die Problematik der unterschiedlichen Farbsysteme bei Druck- und Monitorausgabe fehlt, ist es sinnvoll, bereits beim Einfügen eines Bildes eine automatisierte Konvertierung vom CMYK- in den RGB-Modus oder umgekehrt durchzuführen und Bilddaten in beiden Varianten zu hinterlegen. Auf diese Weise können Farbverfälschungen durch Verwendung des falschen Farbmodells verhindert werden.

Die Nutzung bereits bestehender Auszeichnungssprachen für das wanZine-Projekt ist nur bedingt möglich. Existierende Auszeichnungssprachen sind entweder zu komplex oder aber nicht spezifisch genug. Der geringe Funktionsumfang von HTML eignet sich beispielsweise nicht, exakte Zeitschriftenlayouts zu speichern. DocBook hingegen ist zu komplex für diesen speziellen Anwendungsfall.

Aufgrund seiner Flexibilität und gleichzeitigen Validierbarkeit eignet sich XML gut für die strukturierte Speicherung von Inhalten. Mit Hilfe von XML ist es möglich, eine Auszeichnungssprache nach eigenen Vorstellungen und Bedürfnissen zu erstellen und die Einhaltung dieser Vorgaben mit Hilfe von DTD oder XML-Schema zu sichern. XML-Schema ist dabei die leistungsfähigere Möglichkeit, die Struktur einer XML-Datei festzulegen und zu überprüfen. Besonders aufgrund der erweiterten Unterstützung unterschiedlicher, auch eigener Datentypen ist XML-Schema für die Definition einer XML-Grammatik eine gute Wahl.

Für das wanZine-Projekt ist es sinnvoll, Inhalt und Präsentation in getrennten Dateien zu speichern. Ein solches Vorgehen folgt dem Grundgedanken der Trennung von Inhalt und Präsentation, der die Grundlage für die flexible Verwendung der vorhandenen Daten ist. Erfolgt die Speicherung in getrennten Dateien, ist auch eine Erweiterung auf weitere Ausgabeformate deutlich einfacher zu realisieren.

3. Anforderungsanalyse

Die im vorhergehenden Kapitel gewonnenen Erkenntnisse über die Funktionsweise von Auszeichnungssprachen und die Unterschiede in der Gestaltung für Monitor- und Druckausgabe finden nun praktische Anwendung. Das folgende Kapitel soll darstellen, wie die Struktur zur Speicherung der wanZine-Inhalte aussieht und welche Eigenschaften sie besitzen muss.

Bei den Anforderungen, welche die neue Speichersprache erfüllen sollte, müssen im Wesentlichen zwei Teilbereiche berücksichtigen werden. Die projektspezifischen Anforderungen sind dabei speziell auf die Speicherung von Zeitschrifteninhalten ausgerichtet. Die technischen Anforderungen liegen in der maschinellen Verarbeitung und Ausgabe über den Computer begründet und gelten daher auch für andere Anwendungsfälle.

3.1. Technische Anforderungen

3.1.1. Maschinenlesbarkeit

Die grundlegende Voraussetzung, die der strukturiert gespeicherte Inhalt erfüllen muss, ist die Maschinenlesbarkeit. Inhalte und Gestaltungsinformationen müssen so gespeichert werden, dass daraus automatisiert und ohne menschliches Zutun fertige Publikationen exportiert werden können. Die Veröffentlichung der Inhalte in Form eines animierten Magazins in Flash stellt dabei die höchsten Anforderungen an die Speicherstruktur der wanZine-Inhalte, weil Flash zur Laufzeit nur den Import von wenigen Dateiformaten unterstützt.

3.1.2. Konvertierbarkeit für verschiedene Ausgabeformate

Für den Einsatz im wanZine-Projekt muss der strukturierte Inhalt aber nicht nur maschinenlesbar sein, sondern auch in unterschiedliche Ausgabeformate konvertiert werden können. Dazu muss sowohl eine technische als auch eine logische Konvertierbarkeit der Inhalte gewährleistet sein.

Um aus dem gespeicherten Inhalt mehrere Ausgabeformate zu erzeugen, sollten bestenfalls bereits Techniken existieren, die in der Lage sind, diese Konvertierung automatisch durchzuführen. Gleichermaßen muss das wanZine aber auch logisch konvertierbar sein: Die Inhalte dürfen dementsprechend nicht ausschließlich auf ein Ausgabeformat ausgerichtet sein. Auch die je nach Ausgabemedium unterschiedlichen Farbmodelle sollten bei der Speicherung von Bildinhalten Berücksichtigung finden. In der Inhaltsstruktur sollten zwei Bilddateien hinterlegt werden können. Über die Arbeit mit zwei getrennten Bilddateien für Druck- und Monitorausgabe können Farbverfälschungen bei der Wiedergabe über unterschiedliche Medien vermieden werden.

Je nach gewünschtem Ausgabeformat, in das das wanZine exportiert werden soll, ist die fluss- oder positionsorientierte Speicherung der Layoutvorgaben besser geeignet. Auch eine Mischform kann sinnvoll sein. Aus diesem Grund sollte die Struktur, in der die Inhalte gespeichert werden, für beide Methoden gleichermaßen geeignet sein. Um die Gestaltungs- und Positionsvorgaben den frei positionierbaren Objekten zuordnen zu können, müssen diese innerhalb der Inhaltsstruktur mit eindeutigen Bezeichnern zur Identifikation versehen werden.

3.1.3. Einbindung externer Daten

Da ein wanZine nicht nur aus Text, sondern auch aus Bildern, Videos, multimedialen Objekten und Sound bestehen soll, muss die Speicherstruktur des wanZines die Möglichkeit bieten, externe Dateien einzubinden. Die im wanZine unterstützten Dateiformate müssen daher so gewählt werden, dass sie von möglichst vielen Anwendungsprogrammen, die zur Anzeige des wanZines genutzt werden können, unterstützt werden.

Weil aber nicht alle Arten von möglichen wanZine-Inhalten für alle Ausgabemedien geeignet sind, ist es notwendig, neben der Referenz auf die eigentliche Bilddatei auch eine Möglichkeit zur Speicherung von Alternativinhalten zu bieten. Diese werden zum Beispiel angezeigt, wenn ein Video bei der Druckausgabe nicht dargestellt werden kann.

3.2. Projektspezifische Anforderungen

Die neue Speicherstruktur muss in der Lage sein, sämtliche Bestandteile eines wanZines abzubilden. Diese einzelnen Bestandteile lassen sich in strukturgebende und inhaltsorientierte Elemente unterteilen. Wie bereits in Kapitel 2.4.1.2 festgestellt wurde, ist diese Unterteilung hilfreich, wenn es um eine automatisierte Weiterverarbeitung der Inhalte geht.

Listen und Tabellen bilden eine Art Mischform, da diese Elemente sowohl Merkmale des strukturgebenden als auch inhaltsorientierten Markups aufweisen.[97] Da im wanZine der strukturgebende Aspekt von Listen und Tabellen überwiegt, werden diese im Folgenden als strukturgebende Elemente behandelt.

3.2.1. Strukturelemente

Die Grundstruktur eines wanZines lehnt sich an die Struktur einer klassischen Zeitschrift an. Titelseite, Artikel und Rubriken bilden folglich das Grundgerüst eines wanZines. Die für das wanZine wesentlichen Eigenschaften dieser Strukturelemente werden im Folgenden aufgezeigt. Auch weitere strukturierende Elemente, die innerhalb eines Artikels auftreten können, werden im folgenden Kapitel berücksichtigt.

3.2.1.1. Artikel

Artikel sind die Hauptbestandteile eines wanZines. Da Artikel eines wanZines von unterschiedlichen Autoren und Herausgebern stammen können, muss es möglich sein, diese Informationen für die einzelnen Artikel zu speichern. Auch rechtliche Informationen zu Copyright- und Lizenzbedingungen und das Veröffentlichungsdatum müssen in der Speicherstruktur hinterlegt werden.

[97] Köhler/Wittenbrink 2003, 305

3.2.1.2. Rubriken

Innerhalb eines wanZines muss es möglich sein, einzelne aufeinanderfolgende Artikel in Rubriken einzuordnen. Rubriken dienen dabei ausschließlich der thematischen Gruppierung. Neben ihrem Namen besitzt eine Rubrik keine weiteren Eigenschaften.

3.2.1.3. Titelseite

Jedes wanZine hat genau eine Titelseite, die für jedes wanZine zwingend definiert werden muss. Das Titelblatt unterscheidet sich gestalterisch und damit auch inhaltlich wesentlich von den Innenseiten, was bei der Erstellung der Inhaltsstruktur berücksichtigt werden muss. Neben den allgemeinen Informationen über das wanZine und einem Hintergrundbild enthält es vor allem Hinweise in Bild-und Textform auf die in der Zeitschrift enthaltenen Artikel. Diese sog. Teaser müssen dabei direkt mit den jeweiligen Artikeln innerhalb des wanZines verknüpft sein, um daraus im fertigen Ausgabeformat Links auf die verknüpften Artikel generieren zu können.

3.2.1.4. Abschnitte

Ein wanZine-Artikel besteht aus untergeordneten Abschnitten. Diese können optisch durch Absätze gekennzeichnet und optional mit Zwischenüberschriften versehen werden. Abschnitte können wiederum unterschiedliche Arten Inhalte beinhalten, wobei jedoch Fließtexte die häufigsten Unterelemente eines Absatzes sind. Daneben können Absätze aber auch strukturierte Daten wie Listen und Tabellen oder binäre Daten wie Bilder und Multimedia-Objekte beinhalten.

3.2.1.5. Listen

Innerhalb eines Abschnittes können sowohl geordnete als auch ungeordnete Listen auftreten. Beide Listenformen unterscheiden sich nur in der Form ihrer Darstellung: Während ungeordnete Listen mit Aufzählungszeichen wie Strichen, Pfeilen oder Punkten gekennzeichnet sind, werden geordnete Listen mit aufeinanderfolgenden Ziffern durchnummeriert. Das genaue

Aussehen der Darstellung wird durch die externen Gestaltungsvorgaben bestimmt, die in Form von Stylesheets hinterlegt sind.

3.2.1.6. Tabellen

Strukturierte Daten können in einem wanZine mit Hilfe von Tabellen dargestellt werden. Grundlegend ist dabei natürlich die Speicherung des Inhalts in Form von Zeilen und Spalten. Ebenso wichtig ist die Möglichkeit, Kopfzellen für einzelne Zeilen und Spalten anlegen zu können, die Informationen über den Inhalt der untergeordneten Zellen enthalten. Mögliche Inhalte für Zellen einer Tabelle sind alle Arten von ausgezeichneten Fließtexten, Aufzählungen, Links sowie externe Dateien wie Bilder und Videos.

Wie bereits in Kapitel 2.1.3. festgestellt wurde, ist die Trennung von Struktur und Layout ein wesentlicher Grundsatz für Speicherung der wanZine-Inhalte. Die missbräuchliche Verwendung von Tabellen als Hilfsmittel zur Seitengestaltung, wie sie in HTML-Dateien verbreitet ist, sollte im wanZine möglichst ausgeschlossen sein.

3.2.2. Inhaltselemente

Neben den Strukturelementen muss die neue Speicherstruktur auch die eigentlichen Inhaltselemente eines wanZines abbilden können.

3.2.2.1. Fließtext

Fließtexte sind die wesentlichen Inhaltselemente eines wanZines. Da, wie in Kapitel 2.2 festgestellt wurde, für verschiedene Ausgabemedien unterschiedliche Textauszeichnungsarten geeignet sind, ist es nicht sinnvoll, bereits innerhalb der strukturellen Auszeichnung festzulegen, welche Art der Formatierung für die Hervorhebung von Text verwendet wird. Die Auszeichnung beschränkt sich daher auf die Aussage, dass es sich um einen hervorgehobenen Textabschnitt handelt. Informationen darüber, wie diese Auszeichnung dann genau aussehen soll, legen die Stylesheets für die einzelnen Ausgabeformate fest.

Neben der Kennzeichnung von hervorzuhebenden Textpassagen muss es die Möglichkeit geben, Zeilenumbrüche innerhalb des Fließtextes zu erzwingen.

3.2.2.2. Externe Dateien

Aus der Konzeption des wanZines[98] wird ersichtlich, welche Arten von externen Dateien durch die neue Inhaltsstruktur abgebildet werden müssen. Ein wanZine soll neben Fließtexten auch aus Bildern, Videos, Sounddateien und Multimediaobjekten bestehen. Dabei sind die vom wanZine unterstützen Dateiformate bislang nicht festgelegt und müssen daher im Rahmen der Konzeption definiert werden.

3.2.2.3. Links

Onlinemedien zeichnen sich besonders durch ihre netzartige Verknüpfung untereinander aus. Über sog. Links (engl. Verknüpfung) ist es möglich, durch die aufeinander verweisenden Seiten zu navigieren. Dabei sind sowohl Links innerhalb einer Internetseite als auch auf fremde Internetseiten möglich. Die Onlineausgabe des wanZines sollte die Verlinkung einzelner Objekte unterstützen, um die Stärke des Mediums Internet entsprechend zu nutzen. Aus diesem Grund muss die Speicherstruktur des wanZines die Möglichkeit bieten, Bilder und Texte mit Links auf andere Seiten innerhalb des wanZines oder externe Internetseiten zu versehen.

Die Verlinkung von multimedialen Objekten ist nicht sinnvoll, da diese bereits eine integrierte Link-Funktionalität besitzen können. Außerdem erfolgt die Steuerung multimedialer Objekte häufig über die Computermaus. Ein Mausklick auf ein Multimedia-Objekt wäre dadurch nicht eindeutig als gezielter Linkaufruf zu identifizieren.

---

[98] siehe auch 2.1 Das wanZine-Projekt

3.3. Fazit

Das wanZine stellt vielfältige Anforderungen an die Struktur, in der die Inhalte gespeichert werden sollen. Allgemeine technische Anforderungen an Speicherstrukturen müssen ebenso berücksichtigt werden wie projektspezifischen Anforderungen. Diese resultieren aus den Möglichkeiten der Text- und Inhaltsformatierung, die das wanZine dem Nutzer bieten soll.

Die Speicherstruktur muss daher so aufgebaut sein, dass sie maschinell verarbeitet werden und in verschiedene Ausgabeformate umgewandelt werden. Außerdem muss sie die Einbindung von externen Daten wie Bildern ermöglichen.

Daneben sollte die Speicherstruktur aber auch in der Lage sein, den Inhalt eines wanZines entsprechend seines logischen Aufbaus abzubilden. Da der wanZine-Editor auch von Laien bedient werden und dabei die Strukturierung der Texte zum Teil von Hand vorgenommen werden soll, ist es sinnvoll, die Anzahl der hierfür zur Verfügung stehenden Grundelemente, mit denen Text strukturiert werden kann, zu begrenzen. Eine freie Strukturierung der Inhalte durch den Nutzer ist nicht vorgesehen.

4. Konzeption

Die Konzeption der Inhaltsstruktur für das wanZine besteht aus zwei Bestandteilen. Zuerst muss eine Auszeichnungssprache gewählt werden, die die im vorherigen Kapitel dargestellten technischen Anforderungen erfüllt und sich damit für die Speicherung der wanZine-Inhalte eignet.

Aufbauend auf dieser Entscheidung können daraufhin der Aufbau und die einzelnen Elemente der Inhaltsstruktur sowie deren Eigenschaften so definiert werden, dass diese die projektspezifischen Anforderungen erfüllen können.

4.1. XML als Basis der wanZine-Struktur

Das wanZine-Projekt stellt spezifische Anforderungen an die Auszeichnungssprache, die zur Speicherung der Inhalte genutzt werden soll. Existierende Sprachen wie DocBook und HTML erfüllen diese Anforderungen nur unzureichend, da deren ursprünglicher Einsatzbereich, die technische Dokumentation, sich vom wanZine deutlich unterscheidet. Aus diesem Grund muss für das wanZine-Projekt eine eigene Struktur zur Inhaltsspeicherung entwickelt werden. Diese Inhaltsstruktur wird aus mehreren Gründen auf Basis von XML erstellt.

Valide XML-Dateien sind maschinenlesbar und werden von vielen Programmen als Datenaustauschformat unterstützt. Auch in Flash ist XML ein gut unterstütztes Dateiformat und aus diesem Grund auch hervorragend dafür geeignet, Inhalte zu speichern, die später in Flash weiterverarbeitet werden sollen. Die Wahl von XML als Grundlage der wanZine-Sprache gewährleistet außerdem die technische Konvertierbarkeit der gespeicherten Inhalte. Mit XML lassen sich die Inhalte strukturieren und danach in verschiedene Ausgabeformate konvertieren.[99] Es existieren bereits Techniken wie XSL zur Umwandlung von XML in andere Dateiformate wie PDF oder XHTML. Auch Flash kann XML-Inhalte formatieren und in Form einer

[99] vgl. Köhler/Wittenbrink 2003, 20

SWF-Datei wiedergeben. Durch die Einfachheit und Standardisierung von XML lässt sich die Publikation in verschiedene Ausgabemedien automatisieren. [100]

Nicht zuletzt eignet sich XML auch für die Einbindung externer Dateien. XML kann als textbasierte Auszeichnungssprache zwar externe Dateien nicht direkt einbinden, allerdings lassen sich Dateien wie Bilder und Filme innerhalb des strukturierten Inhalts über eine Referenz zu deren Speicherort einbinden. [101] Auch andere Auszeichnungssprachen wie HTML bedienen sich dieser Technik, um unterschiedliche Dateitypen zusammenzuführen. Bei diesem Vorgehen ist es sinnvoll, mit der Referenz auf den Speicherort auch den Typ der einzubindenden Datei anzugeben, um so die Weiterverarbeitung der Datei definieren zu können.

XML erfüllt damit die technischen Anforderungen, die im vorangegangenen Kapitel gestellt wurden: XML-Strukturen sind maschinenlesbar, lassen sich in andere Formate transformieren und können Referenzen auf binäre Dateien enthalten. XML ist daher als Basis für die strukturierte Speicherung der wanZine-Inhalte gut geeignet.

[100] vgl. Rothfuss/Ried 2001, 49

[101] vgl. Rothfuss/Ried 2001, 169

4.2. Aufbau der wanZine-Struktur

Die Elemente und Attribute der Speicherstruktur werden nun kurz in Aufbau und Funktion erläutert. Das XML-Schema, das die im Rahmen dieser Arbeit entstandene XML-Struktur definiert und damit auch für die Validierung von strukturierten wanZine-Inhalten eingesetzt werden kann, findet sich im Anhang dieser Arbeit.

4.2.1. Strukturgebendes Markup

Zunächst sollen diejenigen Elemente der Speicherstruktur beschrieben werden, die einen strukturierenden Charakter haben. Es wird dargestellt, wie diese Elemente verwendet werden und über welche Attribute sie verfügen. Anhand von Beispielen werden ihr Aufbau und ihre Position innerhalb der wanZine-Struktur veranschaulicht.

4.2.1.1. Das WanZine

Das Wurzelelement, das alle Elemente eines Magazins umschließt heißt `<wanzine>`. Es enthält Attribute, die Informationen zu den Eigenschaften des Magazins enthalten. Der Titel des wanZines steht dabei im Attribut `title`. Das Attribut `author` enthält die Namen der Autoren, `publisher` die Namen der Herausgeber. Das Veröffentlichungsdatum des wanZines kann im Attribut `pubdate` gespeichert werden. Daneben existieren Attribute zur Speicherung von Lizenzbedingungen und Copyright-Informationen. Die Attribute `license`, `copyright` und `author` sind optional, können also weggelassen werden, wenn sie nicht benötigt werden.

```
<wanZine    title="Titel meines wanZines"
            author="Vorname Nachname, Vorname Nachname"
            publisher="Vorname Nachname"
            pubdate="2009-04-01"
            license="Lizenzinformationen"
            copyright="Copyrightinformationen">
      ...
</wanZine>
```

Die möglichen Unterelemente eines wanZines sind Artikel und Rubriken, mit denen es möglich ist, einzelne Artikel thematisch zu gruppieren. Diese beiden Elemente können beliebig oft und in variabler Reihenfolge auftreten. Neben ihnen besitzt jedes wanZine eine Titelseite, die genau einmal zu Beginn der Inhaltsstruktur definiert werden muss.

4.2.1.2. Titelseite

Das erste Unterlement eines wanZines ist folglich das Element `cover`, das Informationen und Inhalte für die Titelseite des Magazins enthält. Im leeren Element `bgimg` wird das Hintergrundbild gespeichert. Da Bilder, wie in der Anforderungsanalyse festgestellt wurde, sowohl im RGB- als auch im CMYK-Modus vorliegen müssen, verfügt das Tag `bgimg` über die beiden Attribute `CMYKref` und `RGBref`, in denen die Speicherorte der beiden Bilder hinterlegt werden.

Daneben enthält das Element cover auch das Element `teaser`, in dem Bilder und Texte, die auf Artikel innerhalb des wanZines verweisen, gespeichert werden. Besonders wichtig ist für Teaser das Attribut `refID`. In diesem Attribut wird gespeichert, auf welchen Artikel innerhalb des wanZines sich die Teaser beziehen. So kann später ein Link bzw. Verweis auf die Innenseiten des wanZines generiert werden.

```
<cover>
    <bgimg  CMYKref="http://www.wanZine.com/bild1"
            RGBref="http://www.wanZine.com/bild2" />
    <teaser refID="AAAAE" id="AAAAA">
       <img CMYKref="http://www.wanZine.com/bild3"
            RGBref="http://www.wanZine.com/bild4"/>
            <text>
                  Ein <emp1>toller</emp1> Artikel!
            </text>
    </teaser>
</cover>
```

4.2.1.3. Artikel

Die Attribute eines durch das Tag `<article>` gekennzeichneten Artikels entsprechen denen eines wanZine-Elements. Auch ein Artikel kann über einen Titel, Autor, Herausgeber, Erscheinungsdatum, sowie Informationen zu Lizenz- und Copyright-Bedingungen verfügen. Zusätzlich erhalten Artikel eine ID, um Artikel bei der späteren Weiterverarbeitung eindeutig identifizieren zu können.

```
<wanZine>
      <cover />
      <article    title="Titel meines Artikels"
                  author="Vorname Nachname, Vorname Nachname"
                  publisher="Vorname Nachname"
                  pubdate="2009-04-01"
                  license="Lizenzinformationen"
                  copyright="Copyrightinformationen"
                  id="AAAAA">
            <img/>
            <para/>
      </article>
</wanZine>
```

Mögliche Inhalte eines Artikels sind Abschnitte und eine beliebige Anzahl von Bildern, die zur allgemeinen Seitengestaltung genutzt werden können. Je nach Ausgabeformat kann später mit Hilfe der Stylesheets bestimmt werden, ob und wo ein Bild auf einer Seite des Artikels zum Einsatz kommt. Bilder, die einem Artikel zugeordnet werden, dienen der allgemeinen Illustration und können beispielsweise als Hintergrundbild für die einzelnen Seiten genutzt werden. Genauere Informationen zur Einbindung von Bildern finden sich in Kapitel 4.2.2.2 Externe Dateien.

4.2.1.4. Abschnitte

Die Inhalte eines Artikels werden durch das Tag `<para>` in einzelne Abschnitte gegliedert. Jedem dieser Abschnitte kann eine eigene Zwischenüberschrift zugeordnet werden. Diese werden im Attribut `title` gespeichert.

```
<para title="Zwischenüberschrift 1">
      Fließtext
</para>
<para title="Zwischenüberschrift 2">
      Fließtext
</para>
```

Abschnitte können Fließtexte, aber auch Tabellen, Listen und alle Arten der im wanZine verfügbaren externen Dateien enthalten. Die Kapitel 4.2.1.7 Listen, 4.2.1.6 Tabellen sowie 4.2.2.2 Externe Dateien bieten weitergehende Informationen zur Einbindung dieser Inhaltstypen in ein wanZine.

4.2.1.5. Rubriken

Wie bereits angesprochen, können Artikel innerhalb des wanZines in unterschiedliche Rubriken (engl. heading) eingeordnet werden. Üblicherweise erscheinen alle Artikel einer Rubrik in der fertigen Publikation auf aufeinanderfolgenden Seiten. Auch in der wanZine-Struktur werden die zueinander gehörenden Artikel daher zusammen gespeichert. Die einzelnen Artikel werden dazu innerhalb des Tags `<heading>` zusammengefasst.

Da das Tag `<heading>` lediglich dazu dient, einzelne Artikel thematisch zusammenzufassen, verfügt es nur über zwei Attribute. Neben einem optionalen Titel erhält auch jede Rubrik eine eindeutige ID.

```
<wanZine>
      <cover />
      <article />
      <heading    title="Meine Rubrik"
                  id="AAAAA">
            <article />
            <article />
        </heading>
</wanZine>
```

4.2.1.6. Tabellen

Tabellen werden mit dem Tag `<table>` innerhalb eines Absatzes erstellt. Ein `<table>`-Tag verfügt über zwei Attribute, `rows` und `cols`, in denen die Anzahl von Zeilen und Spalten in der Tabelle gespeichert wird. Daneben besitzt eine Tabelle wie alle wanZines-Elemente eine eindeutige ID.

Die Inhalte selbst, die aus Fließtext, Listen und externen Daten bestehen können, werden zellenweise in Tags mit dem Namen `<col>` gespeichert. Die Zellen einer Zeile werden dabei innerhalb des Tags `<row>` gruppiert.

Für jede Zelle einer Tabelle lässt sich über das Attribut `header` bestimmen, ob eine Zelle als Kopfzelle formatiert werden soll oder nicht. Dazu reicht es aus, den Wert des Attributs auf `true` oder `false` zu setzten.

```
<table rows="3" cols="2" id="AAAAA">
      <row>
            <col header="true">Größe</col>
            <col header="true">Preis</col>
      </row>
      <row>
            <col>M</col>
            <col>29,95€</col>
      </row>
      <row>
            <col>L</col>
            <col>32,95€</col>
      </row>
</table>
```

Tabellen können im wanZine nicht ineinander geschachtelt werden. So kann zwar nicht gänzlich ausgeschlossen werden, dass Tabellen fälschlicherweise als Layoutelemente verwendet werden, ihre Flexibilität ist aber so eingeschränkt, dass sie für Gestaltungszwecke nahezu untauglich sind.

4.2.1.7. Listen

Im wanZine stehen zwei verschiedene Tags zur Auszeichnung von Aufzählungen zur Verfügung. Das Tag `<orderedlist>` dient zur Auszeichnung von geordneten, also nummerierten Listen, `<unorderedlist>` zur Auszeichnung von ungeordneten Listen. Innerhalb der Listen kennzeichnet das Tag `<listitem>` die einzelnen Listenelemente. Eine gültige Liste besteht mindestens aus zwei Elementen vom Typ `<listitem>`.

```
<orderedlist>
      <listitem/>
      <listitem/>
</orderedlist>
<unorderedlist>
      <listitem/>s
      <listitem/>
</unorderedlist>
```

4.2.2. Inhaltsbezogenes Markup

Neben den strukturgebenden Elementen enthält die wanZine-Struktur auch Elemente, die sich direkt auf den eigentlichen Inhalt beziehen. Diese Elemente werden in den folgenden Kapiteln kurz erläutert.

4.2.2.1. Fließtext

Fließtexte und die darin enthaltenen Elemente zur Textauszeichnung stehen auf der untersten Ebene der wanZine-Hierarchie. Über sie hinaus ist momentan keine weitere Unterstrukturierung der Inhalte angedacht.

Da aus gestalterischer Sicht innerhalb eines Textes nur eine begrenzte Anzahl unterschiedlicher Text-Auszeichnungen verwendet werden sollte, sind vier verschiedene Arten von Textauszeichnungen innerhalb eines Fließtextes für einen wanZine-Artikel ausreichend. Die hervorzuhebenden Textteile werden innerhalb der wanZine-Auszeichnungssprache mit den Tags `<emp1>` bis `<emp4>` gekennzeichnet. Mit Hilfe der Stylesheets kann das genaue Aussehen der Textauszeichnung für die einzelnen Ausgabeformate medienspezifisch festgelegt werden.

```
<para title="Zwischenüberschrift">
      Hier steht ein <emp1>hervorgehobener</emp1> Text!
</para>
```

Neben Abschnitten können Artikel auch mit dem `<br/>`-Tag erzwungene Zeilenumbrüche innerhalb des Fließtextes sowie eine beliebige Anzahl von Bildern beinhalten.

4.2.2.2. Externe Dateien

Es gibt vier Arten von externen Medien, die in ein wanZine eingebunden werden können. Es handelt sich um Videos im FLV-Format, Audiodateien im MP3-Format, Flash-Animationen im SWF-Format sowie Bilder im PNG-, GIF- oder JPEG-Format. Es erfolgt eine Reduzierung auf nur wenige unterstützte Dateiformate, um die Komplexität des wanZines nicht unnötig zu erhöhen. Bei der Wahl der vom wanZine unterstützen Dateiformate kommt es folglich zu Kompromissen: JPEG-Dateien sind für die Druckausgabe zwar nicht optimal, aber dennoch nutzbar, solange sie hochauflösend und wenig komprimiert vorliegen. Ein unkom-primiertes Bild im TIFF-Format wäre zwar für die Druckausgabe gut geeignet, aber weder für die Ausgabe in HTML noch in Flash nutzbar.

Für jeden der vier Medientypen steht ein eigenes Tag zur Einbindung zur Verfügung, da sich die Weiterverarbeitung der einzelnen Medien zum Teil stark unterscheidet und auch die möglichen Attribute variieren.

Allen externen Medien gemeinsam sind jedoch die folgenden Attribute: Wie alle Grundelemente eines wanZines haben auch externe Dateien eine eindeutige ID. Daneben besitzen sie die Attribute `title` und `source`, in denen der Titel des Elements und die Quellenangaben gespeichert werden können sowie spezifische Tags zur Speicherung der Alternativinhalte. Das wesentliche Attribut aller externen Dateien ist jedoch `ref`. In diesem Attribut wird der Link zum Speicherort der externen Datei hinterlegt.

```
<sound
      ref="http://www.wanZine.com/sound1"
      title="Eine Hörprobe"
      alttext="Musik"
      id="Sound1" />
```

```
<flash
      ref="http://www.wanZine.com/flash"
      title="Eine Flash-Animation"
      altimgRGB="http://www.wanZine.com/altRGB"
      altimgCMYK="http://www.wanZine.com/altCMYK"
      alttext="Beschreibung"
      source="istock"
      id="Flash1" />

<movie
      ref="http://www.wanZine.com/video"
      title="Ein Video"
      altimgRGB="http://www.wanZine.com/altRGB2"
      altimgCMYK="http://www.wanZine.com/altCMYK2"
      alttext="Beschreibung"
      source="istock"
      id="Video1" />
```

Einen Sonderfall bildet die Einbindung von Bildern. Da Bilder sowohl für die Druck- als auch Monitorausgabe genutzt werden können, muss die Datei sowohl im CMYK- als auch im RGB-Modus vorliegen. Das Tag `<img>`, das zur Einbindung von Bildern dient verfügt mit den Attributen `CMYKref` und `RGBref` über Verweise zu zwei Bilddateien. Auch Alternativbilder liegen über die Attribute `altimgRGB` und `altimgCMYK` und in zwei Versionen vor.

```
<img
      CMYKref="http://www.wanZine.com/imgCMYK4"
      RGBref="http://www.wanZine.com/imgRGB4"
      alttext="Bildbeschreibung"
      id="Bild4" />
```

4.2.2.3. Links

Es ist möglich, Texte innerhalb eines wanZines mit Links auf andere Internetseiten zu versehen. Dies erfolgt innerhalb von Fließtexten mit Hilfe des Tags `<link>`. Zwischen Start- und Endtag des `link`-Elements steht der Text, der im wanZine sichtbar ist. Der Link, der bei einem Klick auf diesen Text aufgerufen wird, steht im Attribut `URL` .

```
<link URL="www.wanZine.com">Hier klicken!</link>
```

Auch Bilder lassen sich im wanZine verlinken.

```
<img  CMYKref="http://www.wanZine.com/bild3"
      RGBref="http://www.wanZine.com/bild4"
      URL="http://www.wanZine.com"/>
```

Für die Druckausgabe, in der keine direkte Verlinkung möglich ist, ist eine alternative Darstellung der Links notwendig. Das Aussehen dieser alternativen Darstellung kann über die Stylesheets festgelegt werden.

4.3. Fazit

Mit Hilfe der in diesem Kapitel vorgestellten Elemente lässt sich der Inhalt von wanZine medienneutral und wiederverwendbar speichern. Ein einfaches Beispiel für einen in der neuen wanZine-Struktur beschriebenen Zeitschrifteninhalt befindet sich im Anhang dieser Arbeit.

5. Zusammenfassung und Ausblick

Ziel dieser Arbeit war die Entwicklung einer Struktur, mit deren Hilfe die Inhalte einer wanZine-Zeitschrift strukturiert gespeichert werden können.

Mit der Vorstellung des wanZine-Projekts wurde zunächst der Verwendungszweck dieser Speicherstruktur dargestellt. Die Ausgabe der wanZine-Zeitschriften in verschiedene Ausgabeformate wurde dabei als ein wesentlicher Aspekt herausgestellt. Da die zu erstellende Speicherstruktur die Anforderungen unterschiedlicher Ausgabeformate erfüllen soll, muss die Speicherung einerseits zwar medienneutral, also ohne medienspezifische Gestaltungsvorgaben, erfolgen, andererseits aber die wesentlichen inhaltlichen Informationen beinhalten, die für die einzelnen Ausgabeformate benötigt werden.

Um die voneinander abweichenden Anforderungen einzelner Ausgabeformate aufzuzeigen, wurden am Beispiel der für das wanZine relevanten Bildschirm- und Druckausgaben Unterschiede bei der Erstellung von Layouts untersucht. Dabei wurde deutlich, dass es besonders bei der optischen Gestaltung von Texten und den verwendeten Farbmodellen deutliche Unterschiede gibt, die auch bei der Speicherung der strukturierten Inhalte berücksichtigt werden müssen.

Die Vorstellung der Auszeichnungssprache XML und verschiedener Vorgehensweisen bei der strukturierten Speicherung von Inhalten schafft das für die Entwicklung einer eigenen Speicherstruktur benötigte Hintergrundwissen.

Auf Basis der Informationen wurden die von der neuen Speicherstruktur zu erfüllenden Anforderungen analysiert und erläutert. Im letzten Teil dieser Arbeit wurde darauf aufbauend eine Speicherstruktur für wanZine-Inhalte konzipiert, die sich für die strukturierte und medienneutrale Speicherung der Inhalte eines wanZines eignet. Sie erfüllt dabei sowohl die technischen als auch die gestalterischen und konzeptionellen Anforderungen, die das

wanZine-Projekt stellt: Sie ist maschinenlesbar und lässt sich in verschiedene Ausgabemedien konvertieren.

In Fortsetzung dieser Arbeit müssen Techniken definiert werden, um die in dieser Arbeit erstellte XML-Struktur mit Hilfe von Stylesheets in die gewünschten Ausgabeformate zu übertragen. Sobald dies gelungen ist, kann auch der Editor entwickelt werden, der die Inhalte in der wanZine-Struktur speichert.

Für die weitere Zukunft ist auch eine Erweiterung der wanZine-Speicherstruktur durch Einführung neuer Inhaltselemente möglich. Die Einführung eigener Elemente für die Einbindung von Zitaten oder Werbeelementen in Artikel sind denkbare Erweiterungen. Durch die Nutzung von XML zur Beschreibung der Inhalte ist die Einführung solcher neuen Tags problemlos möglich, solange diese Änderungen auch innerhalb des wanZine-Editors und der wanZine-Engine Berücksichtigung finden. Auch bestehende Elemente und Attribute innerhalb der aktuellen Struktur können bei Bedarf an geänderte Anforderungen angepasst werden.

Auch ohne diese möglichen zukünftigen Erweiterungen bietet die in dieser Arbeit entwickelte XML-Struktur jedoch bereits jetzt eine gute Lösung zur Speicherung von Inhalten in einem speziell auf das wanZine-Projekt zugeschnittenen Funktionsumfang.

Literaturverzeichnis

Bücher

Arciniegas, Fabio: XML Developer’s Guide. Poing 2001

Jacobsen, Jens: Website-Konzeption: Erfolgreiche Web- und Multimediaanwendungen entwickeln. 3., erweiterterte Auflage, München 2005

Köhler, Werner/Wittenbrink, Heinz (Hrsg.): XML. Berlin 2003

Jendryschik, Michael: Einführung in XHTML, CSS und Webdesign: Standardkonforme, moderne und barrierefreie Websites erstellen. München 2006

Lankau, Ralf: Webdesign und -publishing. Grundlagen und Designtechniken. 2. Auflage, München, Wien 2000

Merz, Thomas/Drümmer, Olaf: Die PostScript & PDF-Bibel. 2. Auflage, München 2002

Radtke, Susanne/Pisani, Patrizia/Wolters, Walburga: Handbuch Visuelle Mediengestaltung. 2. Auflage, Berlin 2004

Rothfuss, Gunther/Ried, Christian (Hrsg.): Content Management mit XML. Grundlagen und Anwendungen. Berlin, Heidelberg, New York 2001

Runk, Claudia: Grundkurs Typografie und Layout. 2., aktualisierte Auflage, Bonn 2008

Schraitle, Andreas: DocBook-XML. Medienneutrales und plattformunabhängiges Publizieren. 2004

Sebestyen, Thomas: XML. München 2004

Spona, Helma: Das Einsteigerseminar. SVG - Webgrafiken mit XML. Landsberg 2001

Walsh, Norman/Muellner, Leonard: DocBook. The Definitve Guide. Sebastopol, CA 1999

Wilke, Jürgen: Multimedia/Online-Medien. In: Noelle-Neumann, Elisabeth/Schulz, Winfried/Wilke, Jürgen (Hrsg.): Das Fischer Lexikon. Publizistik Massenkommunikation. Aktualisierte, vollständig überarbeitete und ergänzte Ausgabe. 2.Auflage, Frankfurt a.M. 2003

Internet – E-Books

Behme, Henning/Mintert, Stefan: XML in der Praxis, 2. Auflage, 2000, http://www.linkwerk.com/pub/xmlidp/2000/ xml-schema.html, aufgerufen am 17.03.2009

Kutscher, Dirk/Ott, Jörg/Bormann, Carsten/Bergmann, Olaf: Konzepte Content-Repräsentation & Markup-Sprachen, http://www.teialehrbuch.de/Kostenlose-Kurse/Markup-Sprachen, aufgerufen am 24.11.2008

Internet – Spezifikationen und Referenzen

http://www.docbook.org,
aufgerufen am 29.12.2008

World Wide Web Consortium: Extensible Markup Language (XML) 1.0. Second Edition. W3C Recommendation. 06.10.2000,
http://www.w3.org/TR/REC-xml,
aufgerufen am 31.12.2008

World Wide Web Consortium: 2 Introduction to HTML 4.
http://www.w3.org/TR/REC-html40/intro/intro.html,
aufgerufen am 13.01.2009

World Wide Web Consortium: Scalable Vector Graphics (SVG). 1.1 Specification. 14.01.2003,
http://www.w3.org/TR/SVG11/,
aufgerufen am 13.01.2009

World Wide Web Consortium: XML-Schema Part 0: Primer Second Edition. 28.10.2004,
http://www.w3.org/TR/2004/REC-xmlschema-0-20041028/,
aufgerufen am 13.01.2009

World Wide Web Consortium: XML-Schema Part 1: Structures Second Edition. 28.10.2004,
http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/ ,
aufgerufen am 13.01.2009

World Wide Web Consortium: XML-Schema Part 2: Datatypes Second Edition. 28.10.2004,
http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/,
aufgerufen am 13.01.2009

Adobe Systems Incorporated: Document Management. Portable Document Format. Part 1: PDF 1.7. First Edition. 01.07.2008,
http://www.adobe.com/devnet/pdf/pdf_reference.html,
aufgerufen am 13.11.2008

Adobe Systems Incorporated: PDF Reference. Sixth Edition. Adobe Portable Document Format. Version 1.7. 11/2006,
http://www.adobe.com/devnet/acrobat/pdfs/pdf_reference_1-7.pdf,
aufgerufen am 08.01.2009

Adobe Systems Incorporated: Adobe Supplement to the ISO 32000.
BaseVersion: 1.7. Extension Level: 3. 06/2008
http://www.adobe.com/devnet/acrobat/pdfs/adobe_supplement_iso32000.pdf,
aufgerufen am 08.01.2009

http://www.selfsvg.info, aufgerufen am 08.01.2009

Internet – Zeitschriften

Beer, Richard: Database Publishing mit Adobe InDesign CS2, 10/2006,
http://visualxmag.de/itr/online_artikel/psecom,id,827,nodeid,240.html,
aufgerufen am 02.01.2009

Internet – Nachschlagewerke

ITWissen. Das große Online Lexikon für Informationstechnologie. Auszeichnungssprache,
http://www.itwissen.info/definition/lexikon/Auszeichnungssprache-ML-markup-language.html,
aufgerufen am 02.01.2008

Anhang

Anlagenverzeichnis

Anlage 1: XML-Schema der wanZine-Inhaltsstruktur

```
<?xml version="1.0" encoding="utf-8" ?>
<xs:schema
      elementFormDefault="qualified"
      xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="wanZine">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="cover">
          <xs:complexType>
            <xs:sequence>
              <xs:element
                       minOccurs="1"
                       maxOccurs="1"
                       name="bgimg">
                <xs:complexType>
                  <xs:complexContent mixed="false">
                    <xs:extension base="img">
                      <xs:attribute
                              name="refID"
                              type="xs:IDREF" />
                    </xs:extension>
                  </xs:complexContent>
                </xs:complexType>
              </xs:element>
              <xs:element
                       maxOccurs="unbounded"
                       name="teaser">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element
                              name="img"
                              type="img" />
                    <xs:element name="text">
                      <xs:complexType mixed="true">
                        <xs:group
                              maxOccurs="unbounded"
                              ref="text" />
                      </xs:complexType>
```

```
                    </xs:element>
                  </xs:sequence>
                  <xs:attribute
                        name="refID"
                        type="xs:IDREF" />
                  <xs:attribute
                        name="id"
                        type="xs:ID" />
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:group ref="structure" />
      </xs:sequence>
      <xs:attributeGroup ref="Meta" />
    </xs:complexType>
  </xs:element>
  <xs:attributeGroup name="Meta">
    <xs:attribute
            name="author"
            type="xs:string" />
    <xs:attribute
            name="publisher"
            type="xs:string" />
    <xs:attribute
            name="pubdate"
            type="xs:date" />
    <xs:attribute
            name="title"
            type="xs:string" />
    <xs:attribute
            name="license"
            type="xs:string" />
    <xs:attribute
            name="copyright"
            type="xs:string" />
  </xs:attributeGroup>
<xs:group name="content">
    <xs:choice>
      <xs:element name="movie">
```

```
    <xs:complexType>
      <xs:attribute
              name="ref"
              type="xs:anyURI" />
      <xs:attribute
              name="alttext"
              type="xs:string" />
      <xs:attribute
              name="title"
              type="xs:string" />
      <xs:attribute
              name="altimgRGB"
              type="xs:anyURI" />
      <xs:attribute
              name="altimgCMYK"
              type="xs:anyURI" />
      <xs:attribute
              name="source"
              type="xs:string" />
      <xs:attribute
              name="id"
              type="xs:ID" />
    </xs:complexType>
  </xs:element>
  <xs:element name="flash">
    <xs:complexType>
      <xs:attribute
              name="alttext"
              type="xs:string" />
      <xs:attribute
              name="ref"
              type="xs:anyURI" />
      <xs:attribute
              name="title"
              type="xs:string" />
      <xs:attribute
              name="altimgCMYK"
              type="xs:anyURI" />
      <xs:attribute
              name="altimgRGB"
              type="xs:anyURI" />
```

```
      <xs:attribute
              name="source"
              type="xs:string" />
      <xs:attribute
              name="id"
              type="xs:ID" />
    </xs:complexType>
  </xs:element>
  <xs:element name="movie">
    <xs:complexType>
      <xs:attribute
              name="alttext"
              type="xs:string" />
      <xs:attribute
              name="ref"
              type="xs:anyURI" />
      <xs:attribute
              name="title"
              type="xs:string" />
      <xs:attribute
              name="altimgCMYK"
              type="xs:anyURI" />
      <xs:attribute
              name="altimgRGB"
              type="xs:anyURI" />
      <xs:attribute
              name="source"
              type="xs:string" />
      <xs:attribute
              name="id"
              type="xs:ID" />
    </xs:complexType>
  </xs:element>
  <xs:element name="sound">
    <xs:complexType>
      <xs:attribute
              name="ref"
              type="xs:anyURI" />
      <xs:attribute
              name="alttext"
              type="xs:string" />
```

```
          <xs:attribute
                  name="title"
                  type="xs:string" />
          <xs:attribute
                  name="source"
                  type="xs:string" />
          <xs:attribute
                  name="id"
                  type="xs:ID" />
        </xs:complexType>
      </xs:element>
      <xs:element name="img">
        <xs:complexType>
          <xs:complexContent mixed="false">
            <xs:extension base="img" />
          </xs:complexContent>
        </xs:complexType>
      </xs:element>
<xs:element name="table">
       <xs:complexType>
         <xs:sequence>
           <xs:element
                 maxOccurs="unbounded"
                 name="row">
             <xs:complexType>
               <xs:sequence>
                 <xs:element
                       maxOccurs="unbounded"
                       name="col">
                   <xs:complexType mixed="true">
                     <xs:sequence
                             minOccurs="0"
                             maxOccurs="unbounded">
                       <xs:group ref="simpleContent" />
                     </xs:sequence>
                     <xs:attribute
                             default="false"
                             name="header"
                             type="xs:boolean" />
                   </xs:complexType>
                 </xs:element>
```

```
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute
              name="rows"
              type="xs:positiveInteger" />
      <xs:attribute
              name="cols"
              type="xs:positiveInteger" />
      <xs:attribute
              name="title"
              type="xs:string" />
      <xs:attribute
              name="id"
              type="xs:ID" />
    </xs:complexType>
  </xs:element>
  <xs:element name="unorderedlist">
    <xs:complexType>
      <xs:sequence>
        <xs:element
              minOccurs="2"
              maxOccurs="unbounded"
              name="listitem"
              type="listitem" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="orderedlist">
    <xs:complexType>
      <xs:sequence>
        <xs:element
              minOccurs="2"
              maxOccurs="unbounded"
              name="listitem"
              type="listitem" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:group
```

```
          maxOccurs="unbounded"
          ref="text" />
  </xs:choice>
</xs:group>
<xs:group name="structure">
  <xs:choice>
    <xs:element
          minOccurs="0"
          maxOccurs="unbounded"
          name="heading">
      <xs:complexType>
        <xs:sequence>
          <xs:element
                maxOccurs="unbounded"
                name="article">
            <xs:complexType>
              <xs:complexContent mixed="false">
                <xs:extension base="article" />
              </xs:complexContent>
            </xs:complexType>
          </xs:element>
        </xs:sequence>
        <xs:attributeGroup ref="Meta" />
        <xs:attribute
                name="id"
                type="xs:ID" />
      </xs:complexType>
    </xs:element>
    <xs:element
          minOccurs="0"
          maxOccurs="unbounded"
          name="article">
      <xs:complexType>
        <xs:sequence>
          <xs:element
                minOccurs="0"
                maxOccurs="unbounded"
                name="img"
                type="img" />
          <xs:element
                minOccurs="1"
```

```
                maxOccurs="unbounded"
                name="para">
              <xs:complexType mixed="true">
                <xs:complexContent mixed="false">
                  <xs:extension base="para">
                    <xs:attribute
                              name="title"
                              type="xs:string" />
                  </xs:extension>
                </xs:complexContent>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
          <xs:attributeGroup ref="Meta" />
          <xs:attribute
                  name="id"
                  type="xs:ID" />
        </xs:complexType>
      </xs:element>
    </xs:choice>
  </xs:group>
  <xs:group name="text">
    <xs:choice>
      <xs:element
            name="emp1"
            type="xs:string" />
      <xs:element
            name="emp2"
            type="xs:string" />
      <xs:element
            name="emp3"
            type="xs:string" />
      <xs:element
            name="emp4"
            type="xs:string" />
      <xs:element name="link">
        <xs:complexType>
          <xs:simpleContent>
            <xs:extension base="xs:string">
              <xs:attribute
                        name="URL"
```

```
                        type="xs:anyURI" />
            </xs:extension>
          </xs:simpleContent>
        </xs:complexType>
      </xs:element>

      <xs:element name="br">
        <xs:complexType />
      </xs:element>
    </xs:choice>
  </xs:group>
<xs:complexType
            name="para"
            mixed="true">
    <xs:sequence
            minOccurs="0"
            maxOccurs="unbounded">
      <xs:group
            maxOccurs="unbounded"
            ref="content" />
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="article">
    <xs:sequence>
      <xs:element
            maxOccurs="unbounded"
            name="para"
            type="para" />
    </xs:sequence>
    <xs:attributeGroup ref="Meta" />
    <xs:attribute
            name="id"
            type="xs:ID" />
  </xs:complexType>
  <xs:complexType
            name="listitem"
            mixed="true">
    <xs:sequence
            minOccurs="0"
            maxOccurs="unbounded">
      <xs:group ref="text" />
```

```
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="img">
    <xs:attribute
            name="CMYKref"
            type="xs:anyURI" />
    <xs:attribute
            name="RGBref"
            type="xs:anyURI" />
    <xs:attribute
            name="alttext"
            type="xs:string" />
    <xs:attribute
            name="title"
            type="xs:string" />
    <xs:attribute
            name="source"
            type="xs:string" />
    <xs:attribute
            name="URL"
            type="xs:anyURI" />
    <xs:attribute
            name="id"
            type="xs:ID" />
  </xs:complexType>
<xs:group name="simpleContent">
    <xs:choice>
      <xs:element name="movie">
        <xs:complexType>
          <xs:attribute
                  name="ref"
                  type="xs:anyURI" />
          <xs:attribute
                  name="alttext"
                  type="xs:string" />
          <xs:attribute
                  name="title"
                  type="xs:string" />
          <xs:attribute
                  name="altimgRGB"
                  type="xs:anyURI" />
```

```
      <xs:attribute
              name="altimgCMYK"
              type="xs:anyURI" />
      <xs:attribute
              name="source"
              type="xs:string" />
      <xs:attribute
              name="id"
              type="xs:ID" />
    </xs:complexType>
  </xs:element>
  <xs:element name="flash">
    <xs:complexType>
      <xs:attribute
              name="alttext"
              type="xs:string" />
      <xs:attribute
              name="ref"
              type="xs:anyURI" />
      <xs:attribute
              name="title"
              type="xs:string" />
      <xs:attribute
              name="altimgCMYK"
              type="xs:anyURI" />
      <xs:attribute
              name="altimgRGB"
              type="xs:anyURI" />
      <xs:attribute
              name="source"
              type="xs:string" />
      <xs:attribute
              name="id"
              type="xs:ID" />
    </xs:complexType>
  </xs:element>
  <xs:element name="movie">
    <xs:complexType>
      <xs:attribute
              name="alttext"
              type="xs:string" />
```

```
        <xs:attribute
                name="ref"
                type="xs:anyURI" />
        <xs:attribute
                name="title"
                type="xs:string" />
        <xs:attribute
                name="altimgCMYK"
                type="xs:anyURI" />
        <xs:attribute
                name="altimgRGB"
                type="xs:anyURI" />
        <xs:attribute
                name="source"
                type="xs:string" />
        <xs:attribute
                name="id"
                type="xs:ID" />
      </xs:complexType>
    </xs:element>
    <xs:element name="sound">
      <xs:complexType>
        <xs:attribute
                name="ref"
                type="xs:anyURI" />
        <xs:attribute
                name="alttext"
                type="xs:string" />
        <xs:attribute
                name="title"
                type="xs:string" />
        <xs:attribute
                name="source"
                type="xs:string" />
        <xs:attribute
                name="id"
                type="xs:ID" />
      </xs:complexType>
    </xs:element>
   <xs:element name="img">
      <xs:complexType>
```

```
          <xs:complexContent mixed="false">
            <xs:extension base="img" />
          </xs:complexContent>
        </xs:complexType>
      </xs:element>
      <xs:element name="unorderedlist">
        <xs:complexType>
          <xs:sequence>
            <xs:element
                  minOccurs="2"
                  maxOccurs="unbounded"
                  name="listitem"
                  type="listitem" />
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="orderedlist">
        <xs:complexType>
          <xs:sequence>
            <xs:element
                  minOccurs="2"
                  maxOccurs="unbounded"
                  name="listitem"
                  type="listitem" />
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:group
            maxOccurs="unbounded"
            ref="text" />
    </xs:choice>
  </xs:group>
</xs:schema>
```

Anlage 2: Beispiel eines strukturierten wanZine-Inhalts

```
<?xml version="1.0" encoding="utf-8"?>
<wanZine
            author="Eva Staab"
            publisher="Eva Staab"
            pubdate="2009-04-01"
            title="Mein erstes wanZine">
      <cover>
            <bgimg
                  RGBref="http://www.wanZine.com/coverRGB"
                  CMYKref="http://www.wanZine.com/coverCMYK"
                  title="Mein Titelblatt"
                  alttext="Bildbeschreibung"
                  refID="Artikel1" />
            <teaser
                  refID="Artikel2"
                  id="Teaser1">
                  <img  id="Bild1" />
                  <text>
                        Mein <emp1>toller</emp1> Artikel!
                  </text>
            </teaser>
      </cover>
      <article
                  title="Meine wanZine-Premiere"
                  author="Eva Staab"
                  id="Artikel1">
            <img
                  RGBref="http://www.wanZine.com/imgRGB2"
                  CMYKref="htp://www.wanZine.com/imgCMYK2"
                  alttext="Bildbeschreibung"
                  id="Bild2" />
            <para title="Die erste Zwischenüberschrift">
                  Hier könnte ein besonders
                  <emp2>interessanter </emp2>wanZine-Artikel
                  stehen!<br />
                  Dabei können die Texte auch
                  <link URL="www.wanZine.com">verlinkt</link>
                  werden!
            </para>
```

```
        <para title="Die zweite Zwischenüberschrift">
              In diesem <emp3>Absatz</emp3> könnte auch
              eine Flash-Animation eingebunden sein:<br
              />
              <flash
                    ref="http://www.wanZine.com/flash"
                    title="Eine Flash-Animation"
                    altimgRGB="http://www.wanZine.com/altRGB"
                    altimgCMYK="http://www.wanZine.com/altCMYK"
                    alttext="Beschreibung"
                    source="istock"
                    id="Flash1" />
        </para>
  </article>
  <article
              title="Meine zweiter Artikel"
              author="Eva Staab"
              id="Artikel2">
        <img
              CMYKref="http://www.wanZine.com/imgCMYK3"
              RGBref="http://www.wanZine.com/imgRGB3"
              alttext="Bildbeschreibung"
              id="Bild3" />
        <para>
              Hier könnte ein anderer
              <emp4>interessanter</emp4> wanZine-Artikel
              stehen.<br />
              Dieser beinhaltet auch
              <unorderedlist>
                    <listitem> Tabellen</listitem>
                    <listitem>Sounds</listitem>
                    <listitem>Bilder</listitem>
              </unorderedlist>
        </para>
        <para>
              <table
                    rows="2"
                    cols="2"
                    title="Preisliste"
                    id="Tabelle1">
                    <row>
```

```
                              <col header="true">Größe</col>
                              <col header="true">Preis</col>
                        </row>
                        <row>
                              <col header="false">S</col>
                              <col hea-
der="false">19,95€</col>
                        </row>
                  </table>
                  <sound
                        ref="http://www.wanZine.com/sound1"
                        title="Eine Hörprobe"
                        alttext="Musik"
                        id="Sound1" />
                  <img
                        CMYKref="http://www.wanZine.com/imgCMYK4"
                        RGBref="http://www.wanZine.com/imgRGB4"
                        alttext="Bildbeschreibung"
                        id="Bild4" />
            </para>
      </article>
</wanZine>
```

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbstständig und nur unter Verwendung der angegebenen Literatur und Hilfsmittel angefertigt habe. Alle Teile, die wörtlich oder sinngemäß einer Veröffentlichung entstammen, sind als solche kenntlich gemacht.

Die Arbeit wurde noch nicht veröffentlicht oder einer anderen Prüfungsbehörde vorgelegt.

Dresden, den 31.03.09

Eva Staab