Publishing mit XML: Open eBook und DocBook

Richard Cyganiak

21. Februar 2003

Ausarbeitung im Rahmen des Seminars

XML–Technologien

von
Prof. Dr. Robert Tolksdorf
Freie Universität Berlin, Institut für Informatik

Inhaltsverzeichnis

Text-Markup und Publishing mit XML

Die Wurzeln von SGML und XML liegen im Markup von Texten. SGML wurde aus einem System zum Markup von gerichtlichen Texten entwickelt. Die populärste Anwendung von SGML ist eine Text-Markupsprache: HTML (auch wenn HTML nur im theoretischen Sinne eine SGML-Anwendung ist). HTML, die Sprache des Web, hat den Prozess des Publishings, also des Veröffentlichens von Informationen, revolutioniert.

Die Entwicklung von XML, das man als Nachfolger von SGML bezeichnen könnte, wurde stark durch Erfahrungen mit HTML und DocBook, einer weiteren SGML-Anwendung zum Auszeichnen von Texten, beeinflusst. Unter der Vielzahl von XML-Anwendungen, die seitdem entwickelt wurden, finden sich eine Reihe von Lösungen für den Text- und Publishing-Bereich, beispielsweise Open eBook, Open Office, NewsML und RSS. Letzteres hat vielleicht das Potential, die erste auch für den Endnutzer sichtbare Anwendung von XML zu sein.

Forschung, Erwartungen und Hype um XML konzentrieren sich auf die Bereiche Middleware und Wissensrepräsentation. Publishing und Text-Markup hingegen sind Bereiche, in denen XML und verwandte Technologien bereits etabliert sind und sich bewährt haben.

Die vorliegende Arbeit befasst sich mit Open eBook und DocBook. Open eBook ist ein offenes Standard-Format für elektronische Bücher. DocBook wurde ursprünglich als Austauschformat für technische Dokumentation entwickelt und wird inzwischen vor allem im Bereich des Single Source Publishing eingesetzt.

Open eBook

Open eBook ist ein offenes Standard-Format für elektronische Bücher. Zuerst soll der Begriff des eBooks eingeführt werden. Die Anforderungen dieses Marktes und einige existierende Standards und Formate werden aufgezählt. Es folgt eine Beschreibung des Open eBook-Standards. Open eBook baut stark auf anderen Standards wie XHTML und CSS auf. Bekanntheit mit diesen wird vorausgesetzt. Zum Abschluss sollen einige Vor- und Nachteile von Open eBook diskutiert werden.

Was sind eBooks?

Das klassische, aus toten Bäumen produzierte Buch ist überholt. So sehr seine Fans auch von Ledereinbänden und dem Charme alter Bibliotheken, von der Materialität und Konkretheit des gedruckten Buches schwärmen: Ihm fehlt die leichte Durchsuchbarkeit, die Hypertextualität, die Multimedialität, die allgegenwärtige Verfügbarkeit, die Flexibilität in der Darstellung und reversible Annotierbarkeit, die wir durch die modernen elektronischen Medien gewöhnt sind. Es wird sicher noch eine Weile dauern, bis das Buch den Weg des Grammophons gegangen ist, aber das Äquivalent zu Amazon im Jahre 2050 wird seinen Umsatz wahrscheinlich nicht mehr mit dem Verkauf von Büchern machen.

Der Autor kann sich auch nicht vorstellen, dass sie ihren Umsatz mit dem Verkauf von dem machen, was wir heute eBooks nennen. Dennoch sind eBooks das heutige elektronische Gegenstück zum gedruckten Buch.

Terminologie

Der Begriff des eBooks wird nicht einheitlich verwendet. Einerseits kann er ein elektronisches Dokument bezeichnen, andererseits ein Gerät zum Lesen dieser Dokumente. Hier sei mit „eBook“ immer ein elektronisches Dokument gemeint; die Lesegeräte seien als „eBook-Reader“ bezeichnet.

Verschiedene Geräte können zum Lesen von eBooks verwendet werden. Populär ist die Vorstellung eines tragbaren Gerätes in der Größe eines klassischen Buches, das nur zum Lesen von eBooks dient. Ebenso können es aber PDAs, Laptops und ganz gewöhnliche Desktop-PCs sein. Als eBook-Reader lassen sich auch neuartige mobile Geräte verwenden, die der Idee des eBooks möglicherweise bald zusätzlichen Auftrieb verschaffen, wie SmartPhones und Tablet PCs.

Eine genaue Abgrenzung des eBook-Begriffs gegenüber anderen elektronischen Dokumenten wie Word- oder PDF-Dokumenten, Hilfe-Dateien und Webseiten ist schwierig und soll an dieser Stelle nicht versucht werden.

eBook-Formate

Es existieren eine Reihe von Dokumentformaten für eBooks. Einige sind speziell für diesen Anwendungsbereich entwickelt worden, einige sind allgemeine Standard-Dokumentformate.

Anforderungen

Verwendbarkeit auf mobilen Geräten

Tragbare Geräte wie PDAs und SmartPhones verfügen meist über eingeschränkte Rechen- und Speicherkapazität. Kompakte und leicht zu verarbeitende Dateiformate sind von Vorteil.

Portabilität

Die Vielfalt der möglichen Endgeräte macht es wünschenswert, dass ein Dokument auf möglichst vielen Plattformen verwendbar ist. Andernfalls muss der Autor bzw. Verleger für jede Plattform eine eigene Version produzieren. Proprietäre Formate sind meist nur auf bestimmten Plattformen verfügbar, da der Hersteller das zur Anzeige benötigte Programm nicht portiert hat.

Package-Form

Ein klassisches Buch besteht nicht aus losen Seiten, sondern wird gebunden. Ebenso sollte ein eBook nicht aus losen Dateien (Text, Grafiken) bestehen, sondern aus einem klar definierten Paket. Dies erleichtert die Distribution der Dokumente und ihre Verwaltung auf dem Endgerät.

Rechtemanagement

Die natürlichen Content Producer für eBooks sind die klassischen Verlage. Sie verfügen über die Infrastruktur und den Pool an bestehenden Inhalten für die großflächige Umsetzung der eBook-Idee. Sie sind allerdings besorgt um ihr Copyright. Ein Dateiformat, das problemlos kopiert und online gestellt werden kann, wird kaum die nötige Unterstützung seitens der Content-Industrie bekommen. Sie bevorzugt Formate mit der Möglichkeit zur Beschränkung der Nutzung, beispielsweise auf eine bestimmte Person, ein bestimmtes Gerät oder einen bestimmten Zeitraum.

Einige populäre Formate

Es folgt eine Liste einiger wichtiger Dateiformate (nach [Kirvin2001]).

Microsoft Reader

Binäres und proprietäres Format für die gleichnamige Lesesoftware von Microsoft. Es ist nur für Desktop-Windows und Windows CE verfügbar und unterstützt verschiedene Varianten des Rechtemanagements.

MobiPocket

Auch das MobiPocket-Format, lesbar mit der MobiPocket Reader-Software, ist ein binäres Format mit Unterstützung für Rechtemanagement. Es ist insbesondere in der PDA-Welt verbreitet. Die Software ist für viele Plattformen verfügbar, unter anderem für Windows CE, Palm, Psion und Symbian.

PDF

Das Portable Document Format der Firma Adobe ist im Web ein Quasistandard für den Dokumentenaustausch. Die umfangreichen Möglichkeit zum Verschlüsseln, Schützen und Signieren von PDF-Dateien haben das Format auch für die Anwendung im eBook-

Bereich interessant gemacht. PDF-Dateien lassen sich jedoch auf kleinen Bildschirmen nur sehr schlecht anzeigen, da es sich um eine Seitenbeschreibungssprache handelt. Entsprechend ist der Acrobat eBook Reader nur für Windows und Mac verfügbar.

Plain Text
Unstrukturierter ASCII-Text ist das portabelste aller Formate, allerdings aufgrund der fehlenden Möglichkeiten für Navigation und Layout kaum für das Veröffentlichen von Büchern geeignet. Aber für eine andere Meinung siehe [ProjectGutenberg].

(X)HTML
Technologien wie Wireless LAN und GPRS lassen mobile Geräte und das Internet immer mehr konvergieren. Die Bedeutung von HTML als Dokumentformat auf portablen Geräten wächst. Die Verwendung dieser offenen und portablen Sprache als eBook-Format liegt nahe. Allerdings gestaltet sich das offline-Lesen von HTML-Dateien schwierig, und die Anforderungen an die Software sind aufgrund vieler begleitender Dateiformate und inkompatibler Implementierungen hoch.

Das Open eBook-Format: Überblick

Die genannten Formate für eBooks sind entweder für andere Zwecke geschaffene, generische Dokumentformate (Plain Text, HTML, PDF), oder proprietäre Formate, deren Verfügbarkeit auf einer Plattform von der Portierung einer proprietären Client-Software abhängig ist (Microsoft Reader, MobiPocket). Das Fehlen eines offenen und plattformunabhängigen Standardformats wurde 1999 als Bedrohung für den eBook-Markt erkannt. Zur Schaffung eines solchen Formates wurde das Open eBook Forum initiiert ([OeBF]). Neben dem Hauptinitiator Microsoft sind Adobe, Palm und OverDrive die größten Sponsoren ([OeB-Sponsors]) dieser Industrieorganisation. Ihr Ziel:

> „to define a standard means of content description for use by purveyors of electronic books [...] allowing such content to be provided to multiple Reading Systems." ([OeB1.2])

Ergebnis dieser Bemühungen ist die Open eBook Publication Structure (OEBPS). Die erste Version dieser Spezifikation datiert auf September 1999. Die aktuelle Version ist 1.2 und datiert auf Juli 2002. Diese Spezifikation wird oft als OeB-Spezifikation bezeichnet, und das von ihr definierte Format wird als OeB-Format bezeichnet.

In der Terminologie dieser Spezifikation wird ein eBook – also eine in sich geschlossene Veröffentlichung, äquivalent zu einem klassischen Buch – als Publikation bezeichnet. Eine Publikation kann aus mehreren OeB-Dokumenten bestehen, beispielsweise eins je Kapitel. Neben den Dokumenten enthält eine Publikation ein so genanntes Package File, das die Gesamtheit der Publikation beschreibt. Optional können weitere Dateien wie Grafiken und Stylesheets enthalten sein.

Verwendete Standards

Die Open eBook-Spezifikation baut stark auf einer Reihe etablierter Standards auf. XML wird als Format für die OeB-Dokumente und das Package File verwendet. Eine Teilmenge von XHTML findet als Dokumentformat Anwendung. Das Layout der Dokumente erfolgt mittels einer CSS-Teilmenge. Metadaten werden mit Dublin Core ausgezeichnet. Auf einige Besonderheiten der Verwendung dieser Standards in Open eBook wird nun eingegangen.

XML und XML Namespaces
Die XML-Deklaration, laut XML-Standard optional, ist für OeB-Dokumente und Package Files zwingend erforderlich. Als Encoding ist nur UTF-8 oder UTF-16 erlaubt. Die

Verwendung interner DTD-Subsets ist nicht gestattet. Unterstützung von XML Namespaces wird von Lesegeräten nicht verlangt, ist aber erlaubt. Allerdings ist die Angabe bestimmter Namespaces zwingend vorgeschrieben, wie im Abschnitt über das `metadata`-Element genauer beschrieben wird. Ein anderer Default-Namespace als XHTML darf nicht gesetzt werden.

XHTML

Das Format für Dokumente ist eine echte Teilmenge von XHTML 1.1. Als Begründung für die Beschränkung auf eine Teilmenge heißt es in der Spezifikation:

> „[to] minimize the implementation burden on Reading System implementers (who may be working with devices that have power and display constraints) [...]“ ([OeB1.2])

Diese Teilmenge – sie umfasst 66 der 83 Elemente von XHTML – ist durch eine eigene DTD definiert, die OEBPS Document DTD. Dennoch wird von Dokumenten lediglich die Wohlgeformtheit, nicht jedoch die Validität nach dieser DTD gefordert.

Fallbacks

Die Spezifikation schränkt die unterstützten Formate auf folgende Dateitypen ein: XHTML für Texte, CSS als Stylesheets, JPEG und PNG für Grafiken. Publikationen dürfen Dateien in anderen Formaten enthalten, aber es muss immer eine Alternative in einem der genannten Kernformate verfügbar sein. Diese alternativen Versionen werden als Fallbacks bezeichnet. Beipsielsweise könnte ein Lesegerät auch PDF für Dokumente mit besonderen Layoutanforderungen, XML und XSLT für Inhalte, Java und Flash für interaktive Komponenten unterstützen. Eine Standardkonforme OeB-Publikation darf Dateien in diesen Formaten enthalten, muss aber immer auch alternative Versionen in den Kernformaten enthalten. Somit können Lesegeräte zusätzliche Features anbieten, aber Publikationen bleiben dennoch auf allen Geräten verwendbar.

Das Package File

Der Kern jeder OeB-Publikation ist das Package File. Es enthält Metadaten, eine Liste der Dateien der Publikation, Informationen über Fallbacks, und Angaben zur Navigationsstruktur.

```
<?xml version="1.0"?>
<!DOCTYPE package
  PUBLIC "+//ISBN 0-9673008-1-9//DTD OEB 1.2 Package//EN"
  "http://openebook.org/dtds/oeb-1.2/oebpkg12.dtd">
<package unique-identifier="xyz">
  <metadata><!-- ... --></metadata>
  <manifest><!-- ... --></manifest>
  <spine><!-- ... --></spine>
  <tours><!-- ... --></tours>
  <guide><!-- ... --></guide>
</package>
```

Das Beispiel zeigt ein Package File. Das Root-Element jedes Package Files ist das `package`-Element. Die folgenden Abschnitte gehen auf die möglichen Kindelemente ein.

Das metadata-Element

Das `metadata`-Element enthält Metainformationen über die Publikation. Es muss in jeder Publikation vorhanden sein.

```
<package unique-identifier="xyz">
  <metadata>
    <dc-metadata
        xmlns:dc="http://purl.org/dc/elements/1.1/"
        xmlns:oebpackage=
          "http://openebook.org/namespaces/oeb-package/1.0/">
      <dc:Title>Alice in Wonderland</dc:Title>
      <dc:Language>en</dc:Language>
      <dc:Identifier id="xyz" scheme="ISBN">
        123456789X
      </dc:Identifier>
      <dc:Creator role="aut">Lewis Carroll</dc:Creator>
    </dc-metadata>
    <x-metadata>
      <meta name="price" content="USD 19.99" />
    </x-metadata>
  </metadata>

  <!-- ... -->
</package>
```

Das Element `dc-metadata` ist erforderlich und enthält Metainformationen nach dem Dublin-Core-Standard ([DublinCore]). Die beiden Namespace-Angaben `dc` und `oebpackage` sind in genau dieser Form (Präfix und Namespace-URI) fest forgeschriebene Bestandteile der OeB-Spezifikation. Dadurch können auch Parser ohne Namespace-Unterstützung das Package File verarbeiten, indem sie den Namespace-Präfix als Teil des Elementnamens behandeln. Der `dc`-Namespace wird für Dublin Core-Elemente verwendet. Der `oebpackage`-Namespace ist für die Verwendung in zukünftigen Versionen der Spezifikation reserviert.

Das `dc:Creator`-Element kann zusätzlich um ein `role`-Attribut erweitert werden, dessen mögliche Werte durch den MARC-Standard der Library of Congress der Vereinigten Staaten festgelegt sind ([MARC]).

Ein `dc:Identifier`-Element muss ein `id`-Attribut haben, dessen Wert mit `dem unique-identifier`-Attribut des `package`-Elements übereinstimmt. Dadurch wird eine Package-Identität festgelegt, die diese Publikation global eindeutig identifizierbar macht. Als mögliche Identifier werden URIs oder – wie im Beispiel – ISBN-Nummern vorgeschlagen.

Die Elemente `dc:Identifier`, `dc:Title` und `dc:Language` müssen angegeben werden.

Neben dem `dc-metadata`-Element kann ein optionales `x-metadata`-Element angegeben werden, in dem weitere, nicht standardisierte Metainformationen abgelegt werden können.

Das manifest-Element

Auch das `manifest`-Element ist zwingend erforderlich. Es enthält eine Liste aller Dateien, die Bestandteil der Publikation sind, ausgenommen das Package File selbst. Für jede Datei wird ein Dateiname (als URI) und ein MIME-Typ angegeben.

```
<manifest>
  <item id="toc" href="contents.xml"
    media-type="text/x-oeb1-document" />
  <item id="c1" href="chapter-1.html"
    media-type="text/x-oeb1-document" />
  <!-- ... -->
  <item id="titlepic" href="title.png"
    media-type="image/png" />
  <item id="titlepic_tiff" href="titlepic.tiff"
    media-type="image/tiff" fallback="titlepic" />
</manifest>
```

Dieses Beispiel zeigt die Funktionsweise des Fallback-Mechanismus: Die Datei `titlepic.tiff` gehört nicht einem der Standard-Dateitypen an. Daher wurde das Element mit der id `titlepic` als Fallback angegeben, also `title.png`.

Das spine-Element

Klassische Bücher haben zwangsweise eine lineare Struktur: Der Einband hält die einzelnen Seiten in einer festen Reihenfolge zusammen. Das `spine`-Element legt eine solche feste, sequentielle Reihenfolge für eine OeB-Publikation fest. Jedes Kindelement referenziert ein Dokument aus dem Manifest-Abschnitt. Dadurch kann zum Beispiel ein Lesegerät spezielle Tasten zum Vorwärts- und Zurückblättern analog zum klassischen Buch anbieten. Das `spine`-Element muss vorhanden sein.

```
<spine>
  <itemref idref="toc" />
  <itemref idref="c1" />
  <itemref idref="c2" />
  <itemref idref="c3" />
</spine>
```

Das tours-Element

Das `spine`-Element legt eine sequentielle Ordnung der Dokumente fest. Mit dem optionalen `tours`-Element können abweichende Ordnungen des Inhalts angegeben werden. Ein Kochbuch könnte beispielsweise eine Tour für alle Fischrezepte und eine Tour für alle Pastagerichte anbieten.

```
<tours>
  <tour id="tour1" title="Fischrezepte">
    <site title="Tunfischaufstrich"
      href="snacks.html#r3" />
    <site title="Lachs-Blattspinat-Lasagne"
      href="hauptgerichte.html#r14" />
    <!-- ... -->
  </tour>
  <tour id="tour2" title="Pastagerichte">
    <site title="Fettucine alla Carbonara"
      href="hauptgerichte.html#r11" />
    <site title="Lachs-Blattspinat-Lasagne"
      href="hauptgerichte.html#r14" />
    <!-- ... -->
  </tour>
</tours>
```

Das guide-Element

Schließlich kann das optionale `guide`-Element auf typische strukturelle Komponenten eines Buches verweisen.

```
<guide>
  <reference type="toc" title="Inhaltsverzeichnis"
    href="inhalt.html" />
  <reference type="index" title="Index"
    href="anhang.html#index" />
</guide>
```

Mögliche Werte für das type-Attribut sind: `cover`, `title-page`, `toc`, `index`, `glossary`, `acknowledgements`, `bibliography`, `colophon`, `copyright-page`, `dedication`, `epigraph`, `foreword`, `loi` (list of illustrations), `lot` (list of tables), `notes`, `preface`, und benutzerdefinierte Werte, die mit „`other.`“ beginnen müssen, beispielsweise `other.introduction`.

Diskussion

Vergleich mit XHTML

Open eBook baut stark auf XHTML auf. Es stellt sich die Frage, welche neuen Vorteile der Open eBook-Standard gegenüber XHTML bringt.

Bessere Eignung für „kleine“ Geräte

Die vorgenommenen Änderungen an XML, XHTML und CSS sollen das Rendering von OeB-Publikationen auf Geräten mit geringer Rechen- und Speicherkapazität erleichtern. Wenn diese Standards auf solchen Geräten aufgrund technischer Beschränkungen nicht vollständig implementiert werden, dann ist es sinnvoll, sich auf eine gemeinsame Teilmenge zu einigen. Dies erhöht die Interoperabilität und setzt die Schranke für Implementierer niedriger. Andererseits macht das Mooresche Gesetz diese Bemühungen vielleicht obsolet. Opera – ein vollständiger HTML-, XHTML- und CSS-fähiger Webbrowser mit vorbildlicher Unterstützung dieser Standards – läuft auch ohne solche Einschränkungen auf heutigen SmartPhones.

Navigation

Ein wichtiger Faktor für den Erfolg von HTML und des Webs waren die hochflexiblen Navigationsmöglichkeiten durch Hyperlinks. Bücher hingegen werden linear navigiert und haben eine relativ feste Struktur aus standardisierten Elementen wie Inhaltsverzeichnis, Vorwort, Kapitel, Anhänge, Literaturverzeichnis und Index. Open eBook stellt im Package-File spezielles Markup für die Auszeichnung dieser Strukturen zur Verfügung. Dies vereinfacht die Realisierung spezialisierter Navigationsfunktionen. Ein mobiles Lesegerät könnte beispielsweise über eine spezielle Taste verfügen, die eine Liste dieser Buch-Standardelemente aufruft und ihr einfaches Anwählen ermöglicht. Dieses Navigationsfeature würde in jedem eBook identisch aussehen und funktionieren. In XHTML würde wahrscheinlich jeder Autor oder Verleger sein eigenes Navigationsschema erfinden. Andererseits verfügt auch HTML über vergleichbares, wenn auch in der Praxis wenig genutztes Markup (das `link`-Element). Dieses Markup könnte ähnliche spezialisierte Navigationsfeatures ermöglichen.

Fallbacks

Der Fallback-Mechanismus von Open eBook fördert die Erweiterbarkeit des Standards. Lesegeräte können optional weitere Dateiformate unterstützen. Publikationen, die diese

Erweiterungen benutzen, werden trotzdem in allen standardkonformen Lesegeräten funktionieren. Doch auch hier bietet (X)HTML mit dem `object`-Element eine vergleichbare Alternative an.

Besser definierte Package

Eine Publikation im OeB-Format ist ein klar definiertes Paket. Ein OeB-Reader kann mit einen kurzen Blick in das Package-File die wichtigen Metadaten herausholen, die Navigationsstruktur sehen und feststellen, welche Dateien dazugehören. Bei (X)HTML-Dokumenten ist dies erst möglich, nachdem sämtliche Dokumente geparst wurden.

Zusammenfassend lässt sich feststellen, dass der OeB-Standard im Wesentlichen die gleiche Mächtigkeit wie XHTML besitzt. Die Einschränkungen an XML, XHTML und CSS reduzieren jedoch den Implementierungsaufwand. Das Package-File bietet einen zentralen Sammelpunkt für verschiedene Meta- und Verwaltungsinformationen an und erleichtert die effiziente Implementierung von Lesegeräten.

Ausblick

Trotz breiter Industrieunterstützung ist die praktische Relevanz von Open eBook bisher gering. Die großen Player pushen ihre eigenen proprietären Binärformate. Unter „Kompatibilität zu Open eBook“ verstehen sie, dass Tools zum Konvertieren von OeB in ihre proprietären Formate existiren (siehe Microsoft, Gemstar). Die einzige tatsächlich funktionierende Software (neben einigen Ankündigungen, wie [LiberGNU] und [OpenBERG]), die OeB-Publikationen lesen und anzeigen kann, ist [eMonocle] für Java. Bücher im OeB-Format findet man kaum, ob nun für Geld oder kostenlos. Die einzige große Sammlung scheint [GlobalMentor] zu sein, und ihre Veröffentlichungen sind automatische Konvertierungen der Plain-Text-Dokumente von [ProjectGutenberg]. „OeB-Standard“ wird daher gelegentlich scherzhaft mit „other eBook standard“ übersetzt.

Der Open eBook-Standard verzichtet auf eine Beschreibung der Art und Weise, wie diese Dateien physikalisch zu verpacken sind. Für den Endgerätemarkt entscheidende Fragen wie Kompression, Verschlüsselung und Rechtemanagement werden nicht berücksichtigt. Dieser Bereich wird von den proprietären Lösungen verschiedener Anbieter abgedeckt. So ist beispielsweise Microsofts Reader-Format eine binäre, komprimierte Codierung von Open eBook mit zusätzlichen Features für Rechtemanagement.

Content Provider werden ihre Inhalte nur in einem Format veröffentlichen, das ihre kommerziellen Interessen und Urheberrechte schützt. Das tut Open eBook nicht. Nichtkommerzielle Anbieter könnten Inhalte im OeB-Format anbieten, aber das naheliegende Vertriebsformat für sie ist HTML oder PDF, da diese für das weitaus größere Publikum vor Desktop-PCs leichter zugänglich sind. Es ist also nicht klar, woher die Inhalte im OeB-Format kommen sollen. Dies, und die relativ geringen Vorteile von OeB gegenüber HTML, machen es zweifelhaft, ob OeB als Client-Format auf den Endgeräten eine Zukunft hat.

Für Autoren und Verleger ist es sinnvoll, ein Standardformat für den Authoring-Prozess zu haben, und dieses zwecks Maximierung der potentiellen Zielgruppe in verschiedene Formate mit Rechtemanagement (Microsoft Reader, PDF, MobiPocket) zu konvertieren. Open eBook könnte sich als dieses Standardformat für das Authoring und den Austausch von eBooks etablieren. Die Verwendung bekannter Standards wie XHTML und CSS erleichtert das Authoring und ermöglichen die Weiterverwendung vorhandener Tools. Programme zur Konvertierung von OeB in gängige eBook-Formate sind vorhanden (z.B. [ReaderWorks], [MobiPocketPublisher]). Auf den Endgeräten werden wahrscheinlich weiterhin eher diese Formate zum Einsatz kommen, als Open eBook.

Literatur

[DublinCore] Dublin Core Metadata Element Set, Version 1.1. http://dublincore.org/documents/1999/07/02/dces/, 19.02.2003

[eMonocle] ION Systems eMonocle, http://www.ionsystems.com/ion/aboutemonocle.html, 20.02.2003

[GlobalMentor] GlobalMentor Bookstore. http://www.globalmentor.com/bookstore/, 20.02.2003

[Henke2002] Harold Henke: Survey on Electronic Book Features. http://openebook.org/doc_library/surveys/features/downloadformats/ebook_survey.pdf, 18.02.2003

[Kirvin2001] The Format Blues. In: Writing On Your Palm. http://www.writingonyourpalm.net/column011105.htm, 20.02.2003

[LiberGNU] LiberGNU: Free Software Development Group for the Open eBook Standard, http://www.libergnu.org/, 20.02.2003

[MARC] MARC Code List for Relators, Sources, Description Conventions. http://www.loc.gov/marc/relators/, 19.02.2003

[MobiPocketPublisher] MobiPocket Publisher, http://www.mobipocket.com/en/DownloadSoft/ProductDetailsPublisher.asp, 20.02.2003

[OeB1.2] Open eBook Forum: Open eBook Publication Structure 1.2. http://openebook.org/oebps/oebps1.2/download/oeb12-xhtml.htm, 18.02.2003

[OeBF] Open eBook Forum, http://www.openebook.org/, 18.02.2003

[OeB-Sponsors] Open eBook Forum: Gold Sponsors. http://www.openebook.org/about/goldsponsors.htm, 19.02.2003

[OpenBERG] OpenBERG, http://openberg.sourceforge.net/, 20.02.2003

[ProjectGutenberg] Project Gutenberg: History And Philosophy Of Project Gutenberg. http://promo.net/pg/history.html, 20.02.2003

[ReaderWorks] OverDrive ReaderWorks. http://www.overdrive.com/readerworks/, 20.02.2003

DocBook

DocBook ist ein Format zum Markup von Dokumenten. Ursprünglich für den Austausch technischer Dokumentation entwickelt, wird es heute meist zum Zweck des Single Source Publishings eingesetzt. Zuerst sollen dieser Begriff und einige damit verbundene Anforderungen eingeführt werden. Es folgt ein Abschnitt zur Entwicklung des DocBook-Standards, sowie ein Überblick über die in DocBook formulierbaren semantischen Konzepte. Anschließend werden die verfügbaren Tools sowie die Möglichkeiten zur Anpassung der DTD an spezifische Anforderungen beschrieben. Eine Diskussion der Bedingungen zur effektiven Nutzung von DocBook, sowie möglicher Alternativen, schließt die Arbeit ab.

Anforderungen des elektronischen Publishings

Heute werden die meisten Texte mit Hilfe von Computern geschrieben, verarbeitet und veröffentlicht. Dieser Prozess sieht im einfachsten Fall so aus:

- Autor öffnet Programm
- Autor schreibt Text
- Autor wendet einfache Formatierungen an
- Text wird veröffentlicht (z.B. ins Web gestellt, gedruckt oder per Email verschickt).

In vielen Fällen ist der Prozess komplizierter. Der Text soll in verschiedenen Medien mit verschiedenen Anforderungen veröffentlicht werden. Er soll später an anderer Stelle (natürlich mit anderem Layout) wiederverwendet werden. Die Bearbeitung immer gleicher inhaltlicher Strukturen von Hand ist unflexibel und fehleranfällig – Tools sollen diese Arbeiten übernehmen.

Diese Komplexitäten des elektronischen Publishing-Prozesses stellen zusätzliche Anforderungen nicht nur an den Autor, sondern auch an die verwendeten Programme und Dateiformate. Dazu gehören:

Trennung von Inhalt und Layout

Beim Lesen einer Zeitungsseite oder einer Hausarbeit erkennen wir die logische Struktur des Dokuments an den Überschriften. Diese Wortgruppe in großer Schrift ist der Titel des Dokuments. Jene fettgedruckte Zeile ist eine Zwischenüberschrift. Beim Schreiben von Dokumenten, beispielsweise in Microsoft Word, denken viele Autoren in den gleichen Bahnen: „Ich will eine Zwischenüberschrift, also mache ich diese Zeile fett und 16 Punkt“. Leider verstehen Computer solche visuellen Hinweise weit schlechter als Menschen. Damit Word automatisch ein Inhaltsverzeichnis erstellen kann, muss der Autor explizit angeben, was diese Zeile ist, nämlich eine Zwischenüberschrift, und nicht, wie sie aussieht.

Verschiedene Zielformate

Der Text eines Buches kann auch auf einer beigelegten CD auftauchen. Die ausgedruckte Hausarbeit soll auch im Web veröffentlicht werden. Der Referenzteil eines Handbuches soll in die Hilfefunktion einer Applikation integriert werden. Inhalte werden oft nicht für ein bestimmtes Zielmedium verfasst, sondern müssen in verschiedenen Umgebungen funktionieren. Doch andere Medien haben andere Anforderungen. Für den Druck optimierte elektronische Dokumente lassen sich leicht im PDF-Format ins Web stellen, aber diese Lösung ist selten optimal. Besser ist es, das Dokument in mehrere Teile aufzulösen und diese als verlinktes HTML online zu stellen. Die Automatisierung dieses

Prozesses wird als Single Source Publishing bezeichnet: Aus einem einzigen Quelldokument werden automatisch Zieldokumente in verschiedenen Formaten erzeugt.

Hohe Abstraktionsebene

Um den Anforderungen verschiedener Medien gerecht zu werden, und eine flexible Verwendung der Inhalte zu ermöglichen, muss die Beschreibung auf einer ausreichend hohen Abstraktionsebene erfolgen. Beschreibungen von Dokumentteilen auf einer niedrigen Ebene könnten so aussehen:

- „Dies ist ein Absatz mit der Überschrift "Zusammenfassung".“
- „Dies ist ein Absatz am Ende der Seite mit einer hochgestelten 1 davor.“

Auf einer höheren Abstraktionsebene könnten die gleichen Teile so aussehen:

- „Dieser Absatz ist eine kurze Zusammenfassung des Dokuments.“
- „Dieser Absatz ist eine Fußnote.“

Daraus ergeben sich Möglichkeiten wie:

- „Erstelle eine Seite mit den Zusammenfassungen aller Dokumente.“
- „In der Dokumentversion für den Druck platziere die Fußnoten ans Ende der Seite. In der Web-Version platziere sie in eine eigene Datei und verlinke sie mit der Stelle ihres Auftretens.“

Ein Beispiel für die Beschreibung eines Dokuments auf einer recht niedrigen Abstraktionsebene ist HTML. Diese Sprache versteht Konzepte wie Absatz, Überschrift, Liste, Hyperlink. Kompliziertere Ideen wie Fußnote, Abstract, Vorwort, Anhang, Indexeintrag müssen auf die einfachen Konzepte zurückgeführt werden, mit dem entsprechenden Verlust an Möglichkeiten zur flexiblen automatischen Verarbeitung.

Ein Beispiel für ein Dokumentformat, das die Erfüllung dieser Anforderungen erleichtert, ist DocBook.

Die Entwicklung des DocBook-Standards

Im Vergleich zu den meisten XML-Standards hat DocBook eine lange Geschichte. Sie beginnt Ende der 80er Jahre in der Computerindustrie.

Komplexe technische Systeme werden meist mit einer umfangreichen Dokumentation ausgeliefert. Ein Server mit Unix-Betriebssystem kann durchaus mit mehrere Regalmeter an Administrations-, Benutzer-, Referenz- und Programmierhandbüchern kommen. Neben der traditionellen Papierform kamen neue Vertriebswege für diese Materialien auf, beispielsweise CDs und erste Online-Dienste. Damit wurde das Bundling ein Problem: Bücher verschiedener Hersteller lassen sich problemlos in eine Kiste tun. Damit elektronische Versionen dieser Bücher auf einer CD unterkommen, ist ein gemeinsames Format empfehlenswert. Dazu mussten zuerst inkompatible Toolketten harmonisiert werden.

Vor diesem Hintergrund formierte sich 1991 die Davenport Group, ein Forum für Produzenten technischer Dokumentation, organisiert vom O'Reilly-Verlag. Eine erste Version des DocBook-Standards, basierend auf SGML, wurde 1992 veröffentlicht. 1994 wurde die Davenport Group eine offizielle Vereinigung mit eigener Satzung. Zu den Gründungsmitgliedern gehörten neben O'Reilly auch Novell, Fujitsu, HP, DEC, SCO, Hitachi, Sun und Unisys.

Die Arbeit der Davenport Group an DocBook war eine Art "training ground" ([Dougherty1999]) für die Entwicklung von XML. Fünf ihrer zehn Gründungsmitglieder wirkten in der XML Working Group des W3C mit.

Da die Weiterentwicklung von DocBook durch die Arbeit an XML stark gebremst wurde, übergaben die Mitglieder der Davenport Group die Verantwortung 1998 an die OASIS (Organization for the Advancement of Structured Information Standards, [OASIS]). Dort wurde eine Working Group für DocBook eingerichtet. Seit 2000 existiert eine offizielle XML-Zweigversion des DocBook-Standards. Die nächste große Version des Standards soll auf XML basieren, mit einer SGML-Zweigversion. Die aktuelle Version ist 4.2 ([DocBook4.2]), veröffentlicht im Juli 2002.

DocBook hat sich als ein Quasi-Standard für technische Dokumentation etabliert. Insbesondere in der Open-Source-Welt ist DocBook allgegenwärtig. Alle großen Linux-Distributionen außer Debian – Red Hat, SuSE, Mandrake, Caldera – nutzen DocBook, ebenso das Linux Documentation Project und viele Open-Source-Projekte wie FreeBSD, Darwin, KDE, GNOME, PHP, PostgreSQL. Die 150.000 Seiten Dokumentation auf docs.sun.com werden mit DocBook verwaltet. DocBook ist das bevorzugte Format, in dem Autoren ihre Bücher für den O'Reilly-Verlag schreiben sollen. (Quellen: [York2001], [Walsh2001], [Bosak2002], [DocBookWiki], [Robbins2002], [Daugherty1999])

Bei der Entwicklung von DocBook mussten viele verschiedene Interessen berücksichtigt werden. Jede beteiligte Organisation hat andere Erwartungen an die Features und andere Vorstellungen von der Struktur einer Publikation. Als Ergebnis ist DocBook "descriptive" und nicht "prescriptive" geworden. Das heißt, wenn in einem Punkt keine Einigung erzielt werden konnte (Welche Metadaten gehören zu einem Buch? Ist ein Abstract ein Metadatum oder Teil des Dokumentenkörpers? Kann ein Buch auch Manpages enthalten?), dann wurden beide Varianten ermöglicht.

Als Ergebnis kann jede Organisation die Struktur ihrer Publikationen beibehalten und muss sich nicht aufgrund eines rigiden Dokumentformats ihre Prozesse ändern. Die Schattenseite ist, dass DocBook ein verwirrend umfangreicher Standard mit über 300 XML/SGML-Elementen ist, und es an vielen Stellen mehrere Möglichkeiten zum Auszeichnen des gleichen Konzepts gibt. Entsprechend sind die Anforderungen an die Tools sehr hoch.

Syntax und Elemente von DocBook

Die zentrale Struktur in DocBook ist das Buch, aber auch ganze Sätze von Büchern, einzelne Artikel, API-Referenzen, UNIX-Manpages und FAQs lassen sich in DocBook schreiben.

Die Elemente der DocBook-DTD lassen sich grob in drei Gruppen aufteilen: Metainformationen, Hierarchie und Information Pool. Auf die drei Gruppen verstreut sind eine Vielzahl von Elementen für das Markup technischer Konzepte. Für eine detailierte Beschreibung der mehr als 300 Elemente von DocBook sei auf [DocBookReference] verwiesen. An dieser Stelle soll ein grober Überblick genügen.

Hierarchie-Elemente

Die Hierarchie-Elemente ermöglichen die grobe Strukturierung und Gliederung eines Dokuments. Dazu gehören die typischen Root-Elementen `book` und `article`. `preface`, `chapter`, `appendix`, `bibliography`, `glossary`, `index` und weitere unterteilen ein Dokument in grobe Abschnitte. Diese lassen sich jeweils durch verschachtelte `section`-Elemente oder durch die nach Tiefe nummerierten `sect1` bis `sect5` feiner unterteilen. Für Manpage-Einträge ist das `refentry`-Element mit seinen Kindelementen vorgesehen. Für FAQs gibt es das `qandaset`-Element. Neben diesen gibt es etwa 45 weitere Hierarchie-Elemente.

```
<book lang="de">
  <bookinfo>
    <title>Ein Beispiel-Buch</title>
    <author>
      <firstname>Richard</firstname>
      <surname>Cyganiak</surname>
    </author>
    <copyright>
      <year>2002</year>
      <holder>Freie Universität Berlin</holder>
    </copyright>
  </bookinfo>
  <preface><title>Einleitung</title>
    <para>...</para>
  </preface>
  <chapter><title>Das erste Kapitel</title>
    <para>...</para>
  </chapter>
<!-- ... -->
  <appendix><title>Ein Anhang</title>
    <para>...</para>
  </appendix>
</book>
```

Metainformations-Elemente

Die DocBook-DTD enthält über 100 Elemente zur Auszeichnung von Metainformationen. Beispiele sind `title`, `author`, `pubdate`, `revhistory`, `contractsponsor`, `legalnotice` und `issuenum`.

In DocBook 4.2 wurde eine Harmonisierung mit dem Dublin Core Standard vorgenommen. Die Elemente heißen in DocBook teilweise anders, haben aber äquivalente Semantik ([DocBook4.2#DC]).

Fast alle Hierarchie-Elemente können ein info-Element als Kind enthalten, also `book` enthält `bookinfo`, `section` enthält `sectioninfo` usw. Sie dienen als Container für Metainformationen. Das bedeutet, dass nicht nur das Root-Element Metainformationen haben kann. Wenn beispielsweise ein Kapitel von einem anderen Autor geschrieben wurde, so lässt sich dies kenntlich machen.

Alle Elemente für Metainformationen können auch im `bibliography`-Abschnitt verwendet werden, um Literaturangaben auszuzeichnen.

```
<bookinfo>
  <title>User's Guide for the DocBook DTD</title>
  <authorgroup>

<author><firstname>Terry</firstname><surname>Allen</surname>
    </author>
    <author><firstname>Eve</firstname><surname>Maler</surname>
      <affiliation><orgname>Arbortext, Inc.</orgname>
      </affiliation>
    </author>
    <author>
      <firstname>Norman</firstname><surname>Walsh</surname>
      <affiliation><orgname>Arbortext, Inc.</orgname>
      </affiliation>
    </author>
  </authorgroup>
  <edition>User's Guide version 1.0 for DocBook V3.0</edition>
  <pubdate>1997</pubdate>
  <copyright>
    <year>1992</year>
    <year>1993</year>
    <year>1994</year>
    <year>1995</year>
    <year>1996</year>
    <year>1997</year>
    <holder>
      Arbortext, Inc., HaL Computer Systems, Inc.,
      Fujitsu Software Corp.,
      and O'Reilly &amp;amp; Associates, Inc.
    </holder>
  </copyright>
  <legalnotice>
    <para>
      Permission to use, copy, modify and distribute
      the DocBook DTD and its accompanying documentation for
      any purpose and without fee is hereby granted in
      perpetuity, provided that the above copyright notice and
      this paragraph appear in all copies.
    </para>
  </legalnotice>
  <legalnotice>
    <para>
      The copyright holders make no representation about the
      suitability of this DTD for any purpose. It is provided
      <quote>as is</quote> without expressed
      or implied warranty.  If you modify the DocBook DTD in
      any way, except for declaring and referencing additional
      general entities and declaring additional notations,
      identify your DTD as a variant of DocBook.
    </para>
  </legalnotice>
</bookinfo>
```

Information Pool-Elemente

Natürlich muss es auch Elemente geben, in denen der eigentliche Inhalt des Dokuments steckt: Absätze, Listen, Tabellen, Bilder. Das wichtigste dieser Elemente ist `para`. Es enthält einen Absatz. Für Tabellen wird das CALS-Modell verwendet. Es entstand beim US-Verteidigungsministerium und ist in der SGML-Welt recht verbreitet.

```
<informaltable frame='none'>
  <tgroup cols='2'>
    <colspec colwidth='0.5in'/>
    <colspec colwidth='0.5in'/>
    <tbody>
      <row><entry>1</entry><entry>1</entry></row>
      <row><entry>2</entry><entry>4</entry></row>
      <row><entry>3</entry><entry>9</entry></row>
    </tbody>
  </tgroup>
</informaltable>
```

Die oben genannten Inhaltstypen sind im Prinzip aus HTML bekannt. DocBook hat noch etwa 110 weitere Elemente für Inhalte. Der Großteil dient entweder dem Markup technischer Begriffe, oder bezeichnet bestimmte Konzepte aus dem Bereich der Buchformatierung wie `footnote`, `indexterm`, `citation`.

Technische Markup-Elemente

Da DocBook für technische Dokumention entwickelt wurde, gibt es natürlich eine große Anzahl von Elementen zur Auszeichnung technischer Konzepte. Zu den abgedeckten Bereichen gehören Benutzerinterfaces (`keycombo`, `mousebutton`, `menuchoice`, `guiicon`), Programmierung (`methodname`, `ooexception`, `returnvalue`, `constructorsynopsis`), Betriebssysteme (`prompt`, `command`, `errorcode`, `manvolnum`, `filename`) und Hardware (`invpartnumber`, `hardware`). Natürlich ist auch ein `sgmltag`-Element vorhanden. Laut Dokumentation soll es auch für XML-Konstrukte verwendet werden. Spezielles Markup für bestimmte XML-Konstrukte ist für eine zukünftige Version des Standards vorgesehen.

```
<para>
  Und wenn nichts mehr hilft, drücken Sie
  <keycombo action="simul">
    <keycap>Strg</keycap>
    <keycap>Alt</keycap>
    <keycap>Entf</keycap>
  </keycombo>
</para>


<!--
public Wumpus getWumpus(int id)
        throws NoSuchWumpusException;
-->
<methodsynopsis>
  <modifier>public</modifier>
  <type>Wumpus</type>
  <methodname>getWumpus</methodname>
  <methodparam>
    <type>int</type><parameter>id</parameter>
  </methodparam>
  <exceptionname>NoSuchWumpusException</exceptionname>
</methodsynopsis>
```

Das role-Attribut

Für Fälle, in denen der semantische Reichtum des vorhandene Markups immer noch nicht genügt, ist das `role`-Attribut vorgesehen. Es ist auf jedem Element vorhanden. Der DocBook-Standard gibt keine möglichen Werte vor, sondern überlässt die Wahl dem Nutzer.

Dadurch lassen sich Subklassen aller Elemente bilden. Weiterverarbeitende Tools können das Element dann je nach Subklasse unterschiedlich behandeln. Dieser Mechanismus ähnelt dem class-Attribut in HTML und XHTML.

Tools

Viele der gängigen Tools für DocBook sind für die SGML-Version des Standards entwickelt worden. Da XML eine echte Teilmenge von SGML ist, lassen sie sich auch mit XML-Dokumenten verwenden. Der Schwerpunkt dieses Abschnitts soll auf den reinen XML-Tools liegen.

Beim praktischen Einsatz von DocBook sollten in folgenden Bereichen auch die nicht-XML-Tools berücksichtigt werden:

- Konverter von anderen Formaten in DocBook, insbesondere aus MS Word ([MajiX], [Logictran], [YAWC])
- Konvertierung von DocBook in RTF, wofür keine XSLT-Lösung existiert
- Konvertierung von DocBook in PDF, wofür die SGML-Lösung (DSSSL-Stylesheet und jade) bessere Ergebnisse bringt als die XSLT-Lösung über FO

Editoren

Beim Erstellen und Bearbeiten von DocBook-Dokumenten kommen meist generische XML-Editoren zum Einsatz. Die meisten unterstützen die DocBook-DTD out-of-the-box. Sehr beliebt sind verschiedene Lösungen auf Basis von Emacs. XAE scheint das empfehlenswerteste Programm zu sein ([XAE]). Die meisten grafischen XML-Editoren sind noch recht unkomfortabel für eine so umfangreiche Sprache wie DocBook. Marktführer sind Corel XMetaL ([XMetaL]) und Arbortext Epic ([Epic]). Epic scheint die ausgereifteste Lösung zu sein. Leider konnte sich der Autor kein Bild davon machen, da Arbortext keine Evaluationsversion anbietet.

Eine Alternative ist natürlich immer der reine Texteditor. Hier sollte, je nach persönlicher Präferenz, Emacs oder vi gewählt werden.

XSLT-Stylesheets

Das wichtigste Tool im DocBook-Werkzeugkasten sind die XSLT-Stylesheets von Norman Walsh. Sie übernehmen die Konvertierung von DocBook in die meisten Zielformate: HTML, XHTML, FO, Windows Help, Javahelp und Manpages. Für die Konvertierung ist ein XSLT-Prozessor nötig. Für die Verwendung mit diesen Stylesheets wird meist xsltproc oder Saxon empfohlen.

Die DocBook-XSLT-Stylesheets sind eine sehr komplexe Anwendung. Das ist das Resultat von DocBooks enormen Umfang. Die Stylesheets enthalten etwa 100.000 Zeilen XSLT-Code. Ihr Output kann über XSLT-Parameter konfiguriert werden. Der Mechanismus zur Anpassung der Titelseite an individuelle Bedürfnisse erscheint dem Autor etwas kurios: Die Titelseite wird in einer eigenen XML-Sprache beschrieben. Ein XSLT-Stylesheet erzeugt aus dieser Beschreibung ein neues XSLT-Stylesheet. Dieses erzeugt dann aus dem DocBook-Dokument den Output im gewünschten Format. Dieser Prozess ist in [Stayton2002] beschrieben.

Trotz dieser enormen Konfigurierbarkeit sind in vielen Fällen die vorhandenen Anpassungsmöglichkeiten nicht genug. Dann müssen eigene XSLT-Templates geschrieben werden. Dies überfordert viele Nutzer. Ihre Hilfeanfragen zur Anpassung der Stylesheets machen einen Großteil des Traffics auf den Docbook-Mailinglisten ([DocBook-Apps]) aus.

DocBook anpassen

Bei der Entwicklung von DocBook wurde großen Wert auf Erweiterbarkeit und Anpassungsfähigkeit gelegt. Die DTD ist so strukturiert, dass sich leicht neue Elemente einfügen oder unnötige entfernen lassen. Um eine angepasste Version der DTD zu erstellen, legt man eine neue DTD an, das so genannte Driver File. In dieser DTD kann man neue Elemente definieren und Parameter-Entitäten der DocBook-DTD überschreiben, und schließlich die DocBook-DTD importieren.

Teilmengen

Nicht für jede Anwendung wird die volle Flexibilität von DocBook benötigt. Das Entfernen einiger Elemente oder das Erzwingen optionaler Elemente kann den Standard übersichtlicher machen und die Einhaltung bestimmter Richtlinien fördern (z.B. "jedes Dokument braucht einen Abstract").

Sun verwendet mit SolBook eine solche Teilmenge von DocBook. Eine andere Teilmenge ist Simplified DocBook. Diese Version des Standards – offiziell von der DocBook Working Group verabschiedet – reduziert die Anzahl der Elemente von etwa 300 auf etwa 100.

DocBook-Teilmengen sind immer noch gültige DocBook-Dokumente. Daher können die üblichen Tools zur Weiterverarbeitung genutzt werden.

Erweiterungen

Die Working Group betreut eine Reihe offizieller Erweiterungen:

- MathML
- SVG
- EBNF
- HTML Forms

Die Erweiterungen für MathML und SVG fügen diese Standards zum Wortschatz von DocBook hinzu. Die EBNF-Erweiterung ermöglicht das Markup von Grammatiken in erweiterter Backus-Naur-Form.

Neue Struktur

Für einige Anwendungen möchte man den reichen Schatz an Information-Pool- und Meta-Elementen verwenden, aber statt der normalen für Bücher entworfenen Grundstruktur einen anderen Satz von Hierarchie-Elementen verwenden. Ein populäres Beispiel ist die Website-DTD ([WebsiteDTD]), die auch von der DocBook-Website (docbook.org) benutzt wird. Sie erlaubt das Publizieren statischer Webseiten. Der Autor erspart sich hauptsächlich die aufwändige Erstellung von Navigationsmenüs auf jeder Inhaltsseite.

Ein anderes Beispiel ist die Slides-DTD. Sie erlaubt die Erstellung von Präsentationen. Für eine solche Präsentation, siehe [Walsh2001].

Die DocBook Extension For XSLT Stylesheet Documentation ([Ball2001]) ist eine Erweiterung zum Dokumentieren von XSLT-Stylesheets. Dabei wird die Dokumentation im DocBook-Format über einen eigenen Namespace direkt in die XSLT-Dateien geschrieben. Es lässt sich daraus eine umfangreiche hyperverlinkte Referenz erzeugen. Diese Extension wird beispielsweise von der XSLT Standard Library ([XSLTSL]) und Norman Walsh's DocBook-Stylesheets verwendet.

Diskussion

DocBook ist ein stabiler, ausgereifter Standard mit einer reichen Auswahl an Tools. Die große Auswahl an Markup-Elementen ermöglicht die Erstellung komplexer Dokumente auf einer hohen Abstraktionsebene. DocBook-Dokumente können in hoher Qualität in eine Reihe wichtiger Ausgabeformate konvertiert werden.

Andererseits gibt es keine allseits überzeugende Lösung zum Erstellen und Bearbeiten von DocBook-Dokumenten. Entweder wird der Benutzer direkt mit XML-Markup konfrontiert (Emacs), oder die Benutzung des Programms ist umständlich (Low-end XML Editoren), oder es handelt sich um hochspezialisierte und teure Lösungen (Arbortext Epic), oder es steht nur ein Teil der DocBook-Funktionalität zur Verfügung (Konvertierung aus Word, Corel, FrameMaker). Grundkenntnisse in XML sind letztendlich erforderlich, um als Autor mit DocBook zu arbeiten.

Der Standard ist umfangreich und komplex. Die Tools sind oft schwer zu installieren und nicht gerade intuitiv (Java, TeX). Für den effektiven Einsatz müssen oft Stylesheets angepasst werden, so dass auch Kenntnisse in XSLT empfehlenswert sind (auch wenn diese nicht bei den Autoren liegen müssen). Möglicherweise muss auch die DTD selbst angepasst werden. Alles in allem erfordert der Einsatz von DocBook erhebliche Vorarbeit, bevor das erste Dokument zur vollen Zufriedenheit in mehrere Zielformate konvertiert ist. Diese Vorarbeit wird in vielen Szenarios ein großes Argument gegen DocBook sein. Tendenziell lohnt sie sich mehr, wenn große Mengen an Dokumenten verarbeitet werden, oder wenn mehrere Autoren eine zentral aufgesetzte Infrastruktur nutzen können.

Alternativen

DocBook ist Teil der Schnittmenge aus den Anwendungsbereichen "technische Dokumentation", "Single Sourcing" und "Textmarkup mit XML". Abschließend soll ein Blick auf einige andere Anwendungen aus diesen Bereichen geworfen werden.

TeX
: Das TeX-Satzsystem von Donald Knuth ist besonders unter Akademikern beliebt und für seine hochqualitative Druckausgabe bekannt. Für andere Zielmedien ist es allerdings nicht gut geeignet. Die Konvertierung in andere Formate ist schwierig, da TeX-Dokumente eine sehr komplexe Mischung aus Inhalt, Layoutinformationen und Makro-Programmcode sein können.

texinfo
: texinfo basiert auf TeX. Es wurde vom GNU-Projekt als Ersatz für die UNIX-Manpages entwickelt und ist eine sehr saubere, gut von Hand zu schreibende Markup-Sprache. texinfo besitzt nicht den semantischen Reichtum von DocBook, ist aber laut [Robbins2002] eine gute Alternative.

FrameMaker
: Die Software der Firma Adobe ist die wichtigste kommerzielle Alternative zu DocBook im Bereich der technischen Dokumentation.

XHTML
: Die XML-Version der Standardsprache des Webs vereint die Vorteile von XML mit einem hohen Bekanntheitsgrad und vielen guten Tools. Auch wenn die gängige Webdesign-Praxis anders aussieht, ist es möglich, mit XHTML und CSS sehr sauber zwischen Inhalt und Struktur zu trennen. Auch das Publizieren für verschiedene Medien – insbesondere Web und Print – ist inzwischen mit CSS sehr gut möglich. Die nächsten Generationen der beiden Sprachen (XHTML 2.0 und CSS3) werden weitere Verbesserungen bringen. Damit wird XHTML einige Bereiche abdecken können, in

denen bis vor Kurzem noch eine andere Lösung nötig war. Für umfangreiche Dokumente, und bei Bedarf nach umfassenden Metadaten, stellt XHTML allein aber weiterhin keine befriedigende Lösung dar.

TEI
Diese DTD der Text Encoding Initiative ([TEI]) stammt wie DocBook noch aus SGML-Zeiten, ist ausgereift und bewährt, und ist gut mit Tools versorgt. Sie wurde für das Markup literarischer Texte entwickelt und bietet besonderes Markup für linguistische Zwecke.

XML Spec
Diese DTD wurde vom W3C für die Veröffentlichung der XML-Spezifikation entwickelt und wird seitdem dort für Spezifikationen und Technische Berichte verwendet. Es gab kurzzeitig Bestrebungen, sie auch außerhalb des W3C zu etablieren. Die DTD ist in [XMLSpec] dokumentiert.

Entwicklung einer eigenen DTD
Wenn ein sehr eingeschränktes, spezialisiertes Vokabular benötigt wird, und das erforderliche XML-Fachwissen vorhanden ist, dann kann die Entwicklung einer eigenen DTD eine gute Lösung sein. Diese lässt sich genau auf die eigenen Bedürfnisse zuschneidern.

Literatur

[Ball2001] Steve Ball: DocBook Extensions For XSLT Stylesheet Documentation, `http://xsltsl.sourceforge.net/docbook-extensions.html`, 20.02.2003

[Bosak2002] Jon Bosak: The Birth of XML: A Personal Recollection. `http://java.sun.com/xml/birth_of_xml.html`, 20.02.2003

[DocBook.org] O'Reilly DocBook.org, `http://www.docbook.org/`, 20.02.2003

[DocBookIntro] O'Reilly DocBook.org: What Is DocBook? `http://www.docbook.org/oasis/intro.html`, 20.02.2003

[DocBook4.2] OASIS: The DocBook Document Type. Committee Specification 4.2, 16 July 2002. `http://www.oasis-open.org/docbook/specs/cs-docbook-docbook-4.2.html`, 20.02.2003

[DocBook4.2#DC] Relationship to the Dublin Core. In [DocBook4.2]. `http://www.oasis-open.org/committees/docbook/specs/cs-docbook-docbook-4.2.html#d0e652`, 21.02.2003

[DocBook-Apps] `docbook@lists.oasis-open.org` und `docbook-apps@lists.oasis-open.org`. In: OASIS Mailing Lists, http://www.oasis-open.org/docbook/mailinglist/, 21.02.2003

[DocBookReference] DocBook Element Reference, in [Walsh2002]. `http://www.docbook.org/tdg/en/html/part2.html`, 21.02.2003

[DocBookWiki] Who Uses DocBook. In: DocBook Wiki, `http://www.docbook.org/wiki/moin.cgi/WhoUsesDocBook`, 20.02.2003

[Dougherty1999] Dale Dougherty: The Making of the DocBook DTD. http://www.xml.com/pub/a/1999/10/docbook/docbook-making.html, 20.02.2003

[Epic] Arbortext: Epic: Single Source Publishing with XML, http://www.arbortext.com/Products/Epic/epic.html, 21.02.2003

[Logictran] Logictran RTF Converter, http://www.logictran.com/, 21.02.2003

[MajiX] Tetrasix MajiX, http://tetrasix.com/majix.html, 21.02.2003

[Robbins2002] Arnold Robbins: The Making of Effective Awk Programming. http://www.oreillynet.com/pub/a/oreilly/linux/2002/11/01/awk3.html, 20.02.2003

[Stayton2002] Robert Stayton: Using the DocBook XSL Stylesheets. http://www.sagehill.net/xml/docbookxsl/HtmlCustomEx.html#HTMLTitlePage, 20.02.2003

[TEI] Text Encoding Initiative, http://www.tei-c.org/, 20.02.2003

[Walsh2001] Norman Walsh: Introducing DocBook. (Präsentation) http://www.nwalsh.com/docs/tutorials/winwriters2001/, 20.02.2003

[Walsh2002] Norman Walsh und Leonard Muellner: DocBook: The Definitive Guide. http://www.docbook.org/tdg/en/html/docbook.html, 21.02.2003

[WebsiteDTD] DocBook Open Repository: Website DTD. http://docbook.sourceforge.net/projects/website/, 20.02.2003

[XAE] XML Authoring Environment for Emacs, http://xae.sunsite.dk/, 21.02.2003

[XMetaL] Corel XMetaL, http://www.corel.com/xmetal4/, 21.02.2003

[XMLSpec] Guide to the W3C XML Specification DTD, http://www.w3.org/XML/1998/06/xmlspec-report.htm, 20.02.2003

[XSLTSL] XSLT Standard Library, http://xsltsl.sourceforge.net/, 20.02.2003

[YAWC] YAWC Pro: Yet Another Word Converter. http://www.yawcpro.com/, 21.02.2003

[York2001] Dan York: Chasing Documentation's Holy Grail. (Präsentation) http://www.lodestar2.com/people/dyork/talks/2001/xugo/docbook/, 20.02.2003