3.3. Books on Demand

> BoD steht für "Books on Demand" und ist gleichzeitig Firmenname, Technologie, Konzept und Produkt. "Wir machen Bücher" - Books on Demand. Mit der Geschäftsidee, Bücher erst bei Bedarf zu produzieren, kombiniert mit einem einzigartigen Anschluss an den klassischen Buchhandel, ist BoD binnen kürzester Zeit zum Marktführer dieser Branche avanciert. [...] Bei BoD stehen Bücher nicht mehr physisch im Regal, sondern werden als elektronische Druckvorlage im Server gespeichert. Erst mit der Bestellung werden die Bücher produziert und "just in time" ausgeliefert.
>
> — Vorstellung von Books on Demand auf deren Internetseiten, siehe http://www.bod.de/bod/index.html, Zugriff: 12.01.2003

Laut eigenen Angaben ist Books on Demand (kurz BoD) der weltweit erste Internetdienst, der eine Portalseite für den digitalen Buchdruck angeboten hat. Autoren erhalten die Möglichkeit, elektronische Bücher über diesen Dienst zu veröffentlichen. Die Erstellung erfolgt bei Bedarf (*Print on Demand*), und ab einer Auflage von einem Stück[13]. Bei diesem Konzept ist sichergestellt, dass Exemplare eines Buches erst bei Anfrage gedruckt werden; ein Autor wird bei der Veröffentlichung nicht zur Abnahme eines Eigenanteils verpflichtet. Dies ist bei herkömmlichen Verlagen oder Druckereien der Fall, sodass auf den Autor hohe fixe Kosten für die Abnahme dieser Exemplare entfallen. Für die Veröffentlichung eines Buches mit 200 Seiten und eigener ISBN des Autors verlangt BoD in der einfachsten Form einmalige Bruttokosten von 313,20 Euro[14]. Der Verkaufspreis des Dokumentes kann direkt vom Autor festgelegt werden. Als weiteres Feature muss sich dieser Autor nicht um den Absatz kümmern, sein Buch wird automatisch in das Sortiment von BoD aufgenommen.

BoD akzeptiert die Dokumentenvorlage in den Formaten PDF, Postscript oder auch als gedruckte Version, welche dann aufgrund nötiger Nachbearbeitung von BoD mit höheren Kosten verbunden ist. Das DigiPub-System ermöglicht ebenfalls ein Print on Demand, jedoch nur für DocBook-Dokumente. Die Aufbereitung dieser Dokumente, inklusive der Generierung eines ISBN Buchland-Strichcodes, geschieht automatisiert. Der Autor kann hierbei zwischen verschiedenen Layoutformaten wählen. Auch im DigiPub-System wird das Dokument automatisch mit in das Sortiment aufgenommen und durch die Speicherung im XML-Format auch in eine Volltextsuche einbezogen.

[13] Print on Demand wird bereits seit einigen Jahren als eine der größten Innovationen auf dem Buchmarkt gesehen (vgl. http://www.buchhandel.de/nrw/info/bun_98.htm, Zugriff: 18.01.2003)

[14] Errechnet über den Preiskalkulator von Bod (siehe http://www.bod.de/produkte/kalkulator.html, Zugriff: 18.01.2003)

3.4. Slicing Books Technology

Diese an der Universität Koblenz entwickelte Technologie wird von der Slicing Information Technology GmbH weiterentwickelt und auf der eigenen Webseite folgendermaßen beschrieben:

> Die S.I.T. GmbH entwickelt die Slicing Books Technology, mit deren Hilfe sich existierende Bücher und Dokumente in semantische Einheiten gliedern lassen. Diese werden anschließend entsprechend der Fragestellung und unter Berücksichtigung des Wissensstandes des Lesers wieder zusammengesetzt. Das so erzeugte Dokument stellt die aktuell relevanten Ausschnitte aus dem Gesamtwerk dar.
>
> — (http://www.slicing-infotech.de/de/index.php, Zugriff 30.12.2002)

Die Slicing Books Technology ermöglicht, wie auch das DigiPub-DMS, das Erstellen von neuen Dokumenten aus einzelnen Abschnitten verschiedener Dokumente. Es wird ebenfalls eine Suchoption angeboten, in der sowohl eine Volltext- als auch eine Schlüsselwortsuche möglich ist. Das erstellte Dokument wird im PDF-Format ausgeliefert. Die Funktionalität in Bezug auf das Erstellen neuer Dokumente zeigt in vielen Teilen Ähnlichkeiten zu der des DigiPub-DMS (vgl. Abschnitt 6.3.3).

Der Hauptunterschied in beiden Technologien liegt im Datenformat der Dokumente. Bei der Slicing Books Technology wird der Inhalt der Dokumente in einzelne Einheiten (Slices) zerlegt, welche zur Abbildung der logischen Beziehungen untereinander mit Metadaten versehen werden. Dies erfordert ein manuelles Eingreifen, ermöglicht allerdings auch die Nutzung unterschiedlicher Datenformate für diese Technologie. Das DigiPub-DMS speichert die Dokumente im strukturierten XML-Format. Dieses enthält noch keinerlei Aussagen über das Präsentationslayout. Kapitelnummern, Seitenzahlen und Inhaltsverzeichnis werden automatisch für die Struktur des neuen Dokumentes bei Abruf generiert. Diese automatische Generierung einer neuen Struktur ist in dieser Form nicht in den Beispielen für die Slicing Books Technology enthalten.

Ein beispielhafter Abruf für ein neu zu erstellendes Dokument aus dem Lehr- und Übungsbuch für Fachhochschulen (Band 1)[15] mit Auswahl einer Einheit auf unterster Ebene ist in Abbildung 3.2. *Auswahl Slicing Book* zu sehen. Ausgewählt wurde Einheit 1/4/4/3/1: Beispiel 46. Ein Ausschnitt des Ergebnisses ist in Abbildung 3.3. *Ergebnis Slicing Book* zu finden.

[15] Erreichbar unter http://www.slicing-infotech.de/de/sbooks.php (Zugriff 30.12.2002)

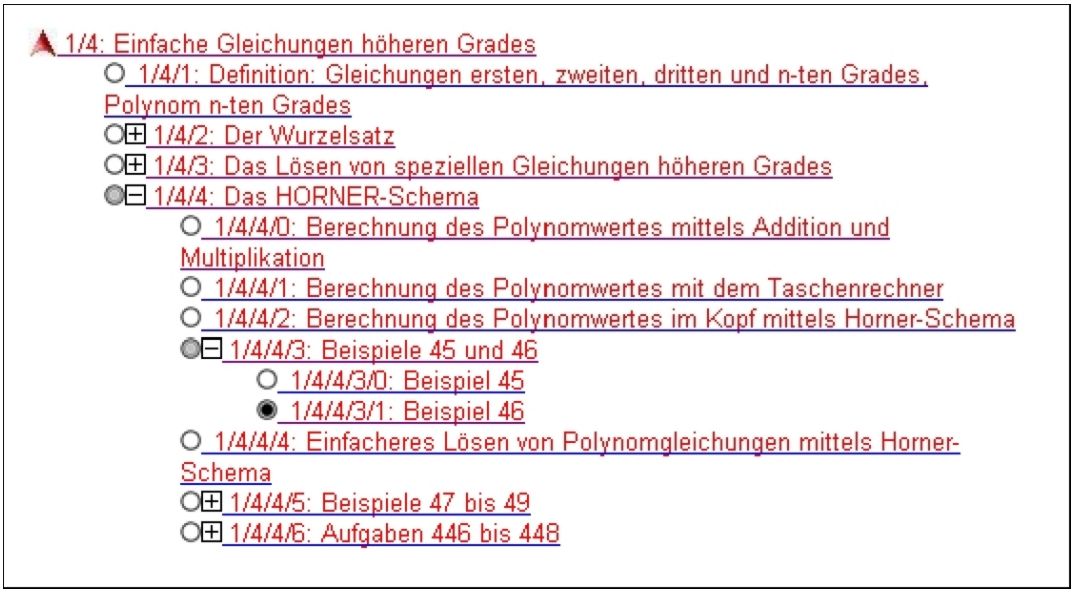

Abbildung 3.2. Auswahl Slicing Book

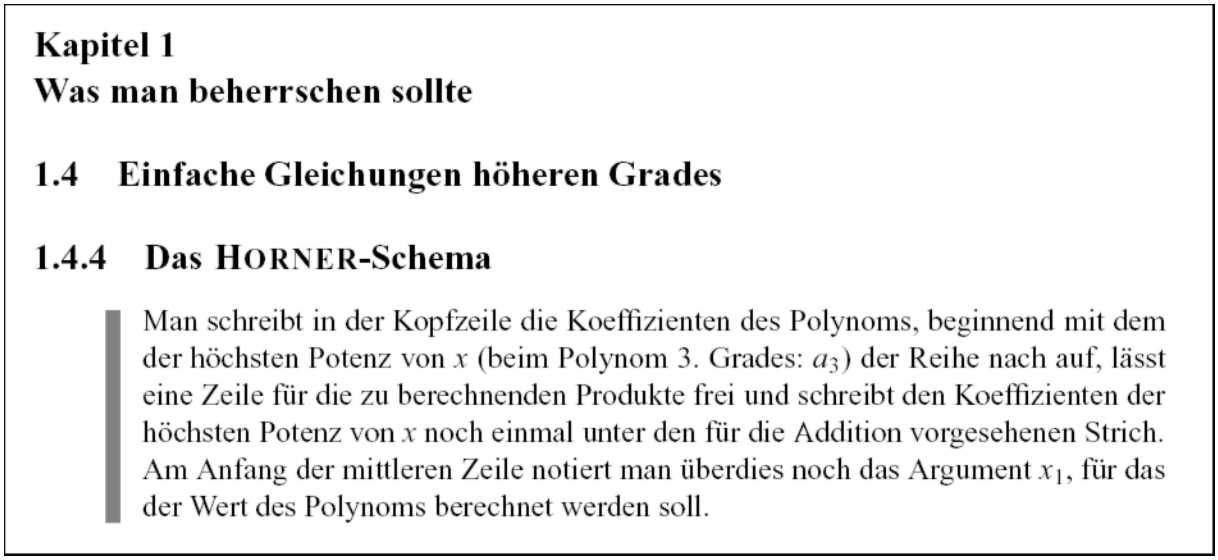

Abbildung 3.3. Ergebnis Slicing Book

Abbildung 3.3. *Ergebnis Slicing Book* zeigt, dass die Originalstruktur des Dokumentes übernommen und eine Neugliederung nicht vorgenommen wurde. Der erste Abschnitt in Kapitel 1 besitzt die Nummerierung 1.4, und auch Seitenzahlen sind im Dokument nicht enthalten. Diese Neugliederung ist durch die strikte Trennung von Inhalt und Präsentationsschicht im DigiPub-DMS automatisch gewährleistet.

Eine Beschreibung der Slicing Books Technology ist zu finden unter [Slicing01] oder auch [Slicing02].

3.5. Zope Book

Die Zope Corporation (Website unter http://www.zope.org/, Zugriff 29.12.2002) bietet ein Open-Source Framework für die Konzeption von Webanwendungen an. Eine Einführung in dieses Framework ist das Zope Book, welches online erreichbar ist unter http://www.zope.org/Documentation/Books/ZopeBook/current (Zugriff 29.12.2002). Ein Vergleich von Zope und dem DigiPub-System wird an dieser Stelle nicht vorgenommen, da beide Systeme unterschiedlichen Aufgaben dienen. Vielmehr wird in diesem Abschnitt das Zope Book betrachtet, welches dem Leser die Möglichkeit bietet, Annotationen zu einzelnen Textabschnitten einzufügen und somit einen Vergleich zu dem DigiPub-DMS ermöglicht. Abbildung 3.4. *Zope Book* zeigt einen Ausschnitt aus diesem Buch.

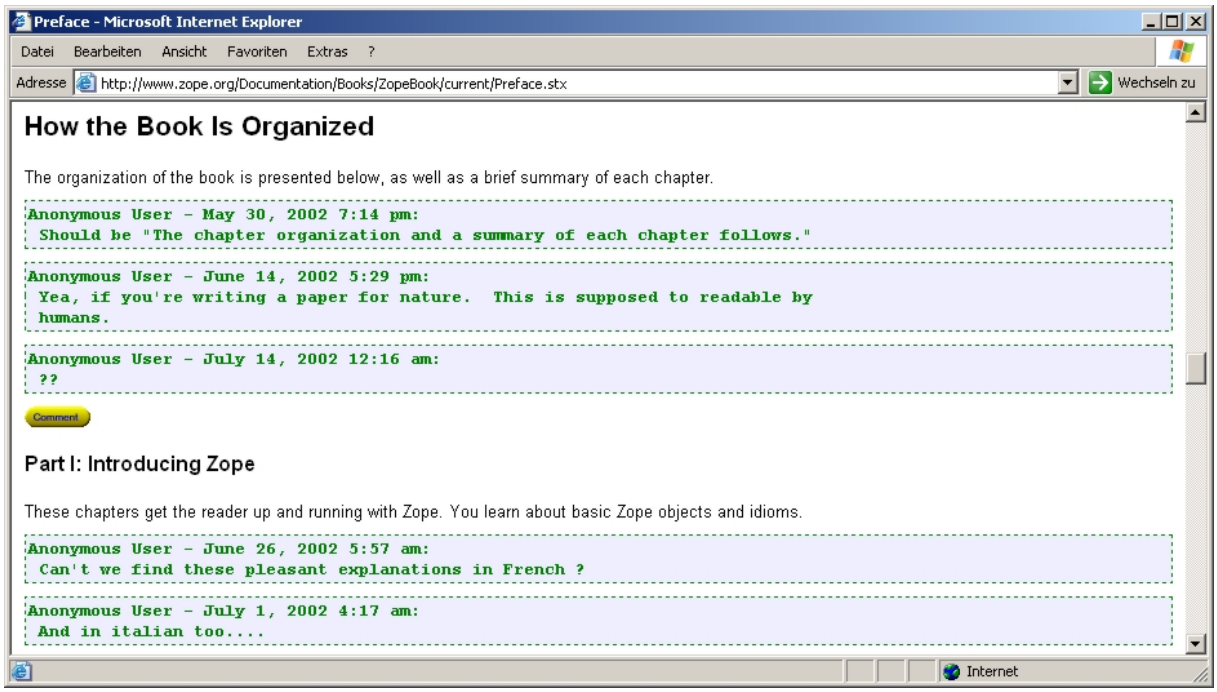

Abbildung 3.4. Zope Book

Das Zope Book basiert auf *StructuredText*, einem einfachen Format zum Strukturieren von Text. StructuredText benutzt Einrückung von Textpassagen und einfache Symbole zur Strukturierung des Textes, wodurch beispielsweise Listen oder unterstrichener Text gekennzeichnet werden können. Im Gegensatz zu Markupsprachen wie XML kann in diesem Format keine Inhaltsstruktur dargestellt werden, sondern lediglich eine Präsentationsstruktur, die vergleichbar ist mit HTML.

Der Quellcode für Abbildung 3.4. *Zope Book* ist dem Beispiel 3.1. *StructuredText im Zope Book* zu entnehmen. Eine Einführung in die Regeln von StructuredText ist in der Zope Community[16] erhältlich.

[16] Erreichbar unter http://www.zope.org/Members/jim/StructuredTextWiki/StructuredTextRules (Zugriff: 31.12.2002)

Beispiel 3.1. StructuredText im Zope Book

Quelle:
http://www.zope.org/Documentation/Books/ZopeBook/current/Preface.stx/document_src

```
How the Book Is Organized

  The organization of the book is presented below, as well as a brief summary
  of each chapter.

    % Anonymous User - May 30, 2002 7:14 pm:
     Should be "The chapter organization and a summary of each chapter follows."

    % Anonymous User - June 14, 2002 5:29 pm:
     Yea, if you're writing a paper for nature.  This is supposed to readable by
     humans.

    % Anonymous User - July 14, 2002 12:16 am:
     ??

  Part I: Introducing Zope

    These chapters get the reader up and running with Zope. You
    learn about basic Zope objects and idioms.

      % Anonymous User - June 26, 2002 5:57 am:
       Can't we find these pleasant explanations in French ?

      % Anonymous User - July 1, 2002 4:17 am:
       And in italian too....
```

Das Zope Book bietet eine ähnliche Funktionalität in Bezug auf Annotationen wie das DigiPub-DMS (vgl. Abschnitt 6.4). Die Annotationen werden direkt in das Dokument und genau an die entsprechende Position eingefügt und somit Bestandteil des Dokumentes. Dies ist im Zope Book auf Paragraphenebene möglich, das DigiPub-DMS erlaubt das Einfügen von Annotationen hingegen nur auf Kapitel- und Abschnittsebene oder direkt als Antwort auf eine bestehende Annotation.

Der wesentliche Unterschied zwischen den beiden Implementationen zum Einfügen von Annotationen besteht jedoch im Datenformat und der Speicherung der Daten. Durch die Nutzung des XML-Datenformates wird der Inhalt des Dokumentes nicht auf der Präsentationsebene, sondern auf der Strukturebene beschrieben. Eine Annotation wird vom DigiPub-DMS an einer eindeutig referenzierbaren Position eingefügt. Dies bietet den Vorteil, dass sogar aus neu erstellten Teildokumenten (virtuelle Dokumente) eine Annotation zu einem bestimmten Abschnitt eingefügt werden kann. Diese wird wiederum in das Originaldokument, welches in einer XML-Datenbank gespeichert wird, eingefügt und somit bei Suchabfragen im System oder neuem Abrufen des Dokumentes mit berücksichtigt. Das Zope Book ist auch im PDF-Format abrufbar, allerdings enthält dieses keine Annotationen[17].

[17] Abrufbar unter http://www.zope.org/Documentation/Books/ZopeBook/current/ZopeBook.pdf (Zugriff 30.12.2002)

Kapitel 4. Eingesetzte Technologien

In diesem Kapitel werden die Grundlagen einiger eingesetzter Technologien beschrieben, welche das Design des DigiPub-DMS maßgeblich beeinflusst haben und zum Verständnis der Umsetzung erforderlich sind. Diese sind vor allem X-Technologien und das Datenformat DocBook. Weiterhin wird auf Techniken zur Speicherung der anfallenden Daten eingegangen und eine Einführung in das Struts-Frameworks gegeben, welches als grundlegende Basis zur Implementation des DigiPub-DMS eingesetzt wurde.

4.1. X-Technologien

Als X-Technologien werden in diesem Dokument Technologien bezeichnet, die XML als Basis enthalten. Abschnitt 4.1.1 beschreibt die Darstellung der Dokumentenstruktur in XML; in Abschnitt 4.1.2 wird die Zuweisung eines Präsentationsformates für ein XML-Dokument beschrieben. Da grundlegendes Wissen über das XML-Format und die Deklaration eines Dokumententyps für das Verständnis weiterer X-Technologien erforderlich ist, werden diese Technologien anhand von Beispielen erläutert. Auf diese Weise wird auch einem Neuling auf diesem Gebiet ein erster Eindruck ermöglicht.

4.1.1. XML

XML bietet die Ausgangsbasis für X-Technologien. Es wurde vom *World Wide Web Consortium (W3C)* entwickelt und wird auf dessen Webseiten wie folgt beschrieben:

> Die Extensible Markup Language (XML) ist ein einfaches, sehr flexibles Textformat, abgeleitet von SGML (ISO 8879). Ursprünglich konstruiert um den Herausforderungen von elektronischen Veröffentlichungen in großem Umfang zu begegnen, spielt XML auch eine zunehmend wichtigere Rolle im Austausch eines großen Angebots an Daten im Internet und anderen Bereichen.
>
> — http://www.w3.org/XML/, Zugriff: 04.01.2003, Eigene Übersetzung.

Durch ihren Ursprung in der *Standard Generalized Markup Language* (SGML) ist auch XML eine Markup-Sprache. Markup-Sprachen haben im Gegensatz zu vielen herkömmlichen Datenformaten den Vorteil, dass sowohl inhaltliche Daten als auch beschreibende Informationen über die Bedeutung der Daten (Strukturinformationen) gemeinsam im Dokument gespeichert werden. XML-Dokumente beschreiben sich somit selbst[18].

In der aktuellen XML 1.0 Empfehlung des W3C[19] sind die Regeln dargestellt, die von einem XML-Dokument erfüllt werden müssen. XML-Dokumente bestehen aus Markup-Elementen, welche sich in folgende Komponenten gliedern lassen:

[18] Vgl. [Klettke2002], S 22

[19] Siehe [XML 1.0]

- Deklarationen: diese dienen zur Angabe von Metainformationen wie der verwendeten XML-Version oder des genutzten Zeichensatzes.
- Elemente: diese bilden den Hauptbestandteil von XML-Dokumenten.
- Kommentare
- Referenzen: diese dienen zur Referenzierung von vorab definierten Inhalten und werden in die Zeichen & und ; eingeschlossen. In XML sind die Zeichen >, <, ', " und & vordefiniert, letzteres wird beispielsweise referenziert durch *&*.
- Verarbeitungsanweisungen: diese können genutzt werden, um dem XML-Dokument Informationen für die Verarbeitung durch ein spezielles Programm mitzugeben.

XML-Dokumente werden durch eine Baumstruktur dargestellt. Sie bestehen neben Deklarationen, Kommentaren und Verarbeitungsanweisungen aus genau einem Element, welches wiederum Elemente enthalten kann. Dieses Element wird als Root-Element bezeichnet und bildet den Wurzelknoten der Baumstruktur. Leere Elemente und reiner Text bilden die Blätter der Baumstruktur[20]. Gekennzeichnet werden Elemente durch ein einleitendes und ein schließendes *Tag* in der Form <bezeichner> und </bezeichner>, wobei bezeichner für einen zu vergebenden Elementnamen steht. Leere Elemente bilden ein Blatt des Baumes und enthalten somit keine weiteren Inhalte. Diese Elemente können durch eine verkürzte Schreibweise durch ein einzelnes Tag dargestellt werden (Beispiel: <bezeichner/>). Tags können weitere Informationen in Form von Attributen enthalten, welche durch die Angabe *name="wert"* aufgezeigt werden. Ein Beispiel hierfür ist in Beispiel 4.1. *dokument.xml* zu finden.

Beispiel 4.1. dokument.xml

```
<?xml version="1.0"?>                                    ❶
<!DOCTYPE dokument SYSTEM "dokument.dtd">                ❷
<dokument>                                               ❸
  <version nummer="2.0"/>                                ❹
  <titel>Testdokument</titel>
  <absatz>Dies ist ein Testdokument.</absatz>
</dokument>                                              ❺
```

❶ optionale XML-Deklaration zur Spezifizierung der genutzten XML-Version.
❷ optionale Dokumententyp-Deklaration zur Spezifizierung der Dokumentenstruktur.
❹ Ein leeres Element mit dem Attribut *nummer*.
❸ Starttag des Root-Elementes.
❺ Endtag des Root-Elementes.

[20] Vgl. [Goldfarb2000], S. 68/69

Um eine Eindeutigkeit der Elementnamen zu gewährleisten, können Elemente einem *Namensraum* zugewiesen werden. Hierdurch kann die Quelle, und somit auch die Bedeutung, für gleichnamige Elemente eindeutig referenziert werden. Beispielsweise wird durch die folgende Angabe definiert, dass Elemente, denen die Bezeichnung xsl: voransteht, dem Namensraum des W3C zuzuordnen sind:

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"/>
```

Ein XML-Dokument, welches der [XML 1.0] Spezifikation entspricht, wird als *wohlgeformt* bezeichnet. Um eine Einschränkung der Dokumentenstruktur zu erreichen und zu überprüfen, kann einem XML-Dokument eine Dokumententyp-Definition (DTD) zugewiesen werden.

4.1.1.1. DTD

Eine Dokumententyp-Definition definiert Elemente und deren Attribute sowie die Struktur der Elemente, die für ein Dokument dieses Typs zulässig ist. Durch diesen Mechanismus kann geprüft werden, ob ein XML-Dokument einem bestimmten Dokumententyp zugeordnet werden kann. Beispiel 4.1. *dokument.xml* enthält eine Dokumententyp-Deklaration (siehe ❷), durch die festgelegt wird, dass dieses Dokument vom Dokumententyp *dokument* ist, welcher in Beispiel 4.2. *dokument.dtd* definiert ist. Eine Überprüfung dieser Angabe kann durch entsprechende Programme, genannt XML-Parser, überprüft werden. Ein XML Dokument, welches sich an die Regeln einer referenzierten DTD hält, wird als *gültiges* XML-Dokument bezeichnet.

Beispiel 4.2. dokument.dtd

```
<!ELEMENT dokument (version?, titel, absatz+)>   ❶
   <!ELEMENT version EMPTY>                      ❷
   <!ATTLIST version nummer CDATA #REQUIRED>     ❸
   <!ELEMENT titel (#PCDATA)>
   <!ELEMENT absatz (#PCDATA)>
```

Der grundlegende Aufbau einer DTD wird im Folgenden anhand von Beispiel 4.2. *dokument.dtd* erläutert. Hierbei wird lediglich auf die Deklaration von Elementen und Attributen eingegangen, da dies zum Verständnis des vorliegenden Dokumentes ausreicht. Für eine vollständige Spezifikation wird auf [XML 1.0] verwiesen.

Innerhalb einer DTD werden Elemente durch das Konstrukt <!ELEMENT bezeichner inhalt> beschrieben. *bezeichner* enthält den Elementnamen (in ❶ dokument), *inhalt* den erlaubten Inhalt des Elementes. Möglicher Inhalt kann die Angabe *EMPTY* (vgl. ❷) für „keinen Inhalt“ oder eine Kombination von Elementen und Text (*#PCDATA* - Parsable Character Data) sein, welche innerhalb von Klammern angegeben wird. Als dritte Möglichkeit ist die Angabe

ANY erlaubt, die sowohl „keinen Inhalt“ als auch eine nicht spezifizierte Kombination von Text und Elementen erlaubt. Die Reihenfolge der Elemente und die Anzahl der Vorkommen wird durch spezielle Zeichen gekennzeichnet, die in folgender Liste dargestellt sind. Zum Gruppieren mehrerer Elemente werden Klammern benutzt (vgl. Beispiel 4.2. *dokument.dtd*).

Sequenz
: Eine Sequenz wird durch Kommata gekennzeichnet. Beispiel: (element1, element2)

Alternative
: Eine Alternative wird durch das Zeichen | gekennzeichnet. Beispiel: (element1 | element2)

Multiplikatoren
: Multiplikatoren stehen hinter den Elementen oder Blöcken, auf die sie bezogen werden. Folgende Multiplikatoren sind verfügbar:

  Kein Multiplikator
  : Vorkommen genau einmal, Beispiel: (element1)

  ?
  : Vorkommen null bis einmal, Beispiel: (element1)?

  +
  : Vorkommen mindestens einmal bis unendlich oft, Beispiel: (element1)+

  *
  : Vorkommen gar nicht bis unendlich oft, Beispiel: (element1)*

Attribute für deklarierte Elemente werden durch das Konstrukt <!ATTLIST bezeichner beschreibung> dargestellt. *bezeichner* enthält auch hier den Elementnamen, *beschreibung* die Beschreibung des Attributes. Die *beschreibung* des Attributes besteht aus der Attributbezeichnung, dem Attributtyp und einer Angabe, ob das Attribut zwingend vorhanden sein muss oder nicht und wie ein XML-Parser reagieren soll, falls dieses Attribut nicht vorhanden ist. Folgende Attributtypen sind in [XML 1.0] definiert:

CDATA
: Beliebiger Text.

ID
: Ein im Dokument eindeutiges Elementkennzeichen.

IDREF, IDREFS
: Ein (IDREF) oder mehrere (IDREFS, einzelne IDREF-Werte durch Whitespace-Zeichen getrennt) Wert(e) eines Attributes vom Typ ID; hierdurch wird eine Referenz auf dieses Element gebildet.

ENTITY, ENTITIES	Referenz auf eine (ENTITY) bzw. mehrere (ENTITIES, einzelne ENTITY-Werte durch Whitespace-Zeichen getrennt) Zeichenkette(n), welche in der DTD deklariert sein müssen (vgl. Referenzen in XML).
NMTOKEN, NMTOKENS	Ein Wort (NMTOKEN) oder mehrere (NMTOKENS) Wörter, welche als Bezeichner für Elemente zulässig sind. Eine Definition für erlaubte Elementbezeichner ist in [XML 1.0] zu finden.
eine Aufzählung	Mehrere Werte, getrennt durch das Zeichen | , von denen einer gewählt werden kann. Bei diesem Attributtyp ist die Angabe eines vorgegebenen Wertes aus der Aufzählung hinter dieser als Standardwert möglich.
NOTATION	Eine Notation, die in der DTD deklariert sein muss.

Mögliche Angaben zur Präsenz eines Attributes sind im Folgenden aufgelistet (vgl. [XML 1.0]):

#REQUIRED	Das Attribut ist zwingend erforderlich.
#IMPLIED	Das Attribut ist nicht zwingend erforderlich.
#FIXED	Das Attribut ist in der DTD fest vorgegeben und kann nicht verändert werden. Die Festlegung des Wertes erfolgt direkt hinter dieser Angabe.

Ein Beispiel für die Deklaration eines Attributes ist in Beispiel 4.2. *dokument.dtd* bei ❸ zu finden.

Zur Überprüfung auf Wohlgeformtheit oder Gültigkeit eines XML-Dokumentes stehen diverse frei verfügbare und auch kommerzielle XML-Parser zur Verfügung. Das DigiPub-DMS nutzt für diese Zwecke *Xerces 2 (Java)* der *Apache Software Foundation* (siehe Xerces-J 2.2.1).

Eine Dokumententyp-Definition bietet eine einfache Art, die Struktur von XML-Dokumenten festzulegen. Für viele Dokumentenformate, wie auch das im DigiPub-DMS genutzte Dokumentenformat DocBook (siehe Abschnitt 4.2), ist diese Technik ausreichend. Für exakte Datenbeschreibungen hingegen bietet eine DTD keine ausreichenden Strukturierungsmerkmale. Dies soll durch das XML Schema Format erreicht werden.

Eine kurze, aber hinreichende Anleitung zum Umgang mit XML ist unter [Eckstein2001] zu finden. Für eine umfassende Ausarbeitung wird auf [Goldfarb2000] verwiesen.

4.1.1.2. XML Schema

Ein XML Schema bietet im Vergleich zu einer DTD mehrere Vorteile. In einem XML Schema können explizit Datentypen für Elemente und Attribute festgelegt werden. Des Weiteren können Gültigkeitsregeln und Nebenbedingungen für Elemente definiert werden[21]. Das Format eines XML Schemas beruht selbst auf XML. Nachteilig hingegen ist die Komplexität im Gegensatz zu einer Dokumententyp-Definition.

Die Spezifikation von Dokumententypen für das DigiPub-DMS wurde, soweit möglich, mit Hilfe einer entsprechenden DTD vorgenommen. Aufgrund dessen ist die genaue Spezifikation für ein XML Schema für das Verständnis dieser Diplomarbeit nicht erforderlich. Für detaillierte Informationen wird auf die XML Schema Empfehlung des W3C verwiesen (siehe XML Schema). Um jedoch einen ersten Einblick der Syntax von XML Schema gewinnen zu können, wird in Beispiel 4.3. *dokument.xsd* das XML Schema abgebildet, welches äquivalent zu der DTD in Beispiel 4.2. *dokument.dtd* ist.

Beispiel 4.3. dokument.xsd

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="dokument">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="version" minOccurs="0"/>
        <xsd:element ref="titel"/>
        <xsd:element ref="absatz" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="version">
    <xsd:complexType>
      <xsd:attribute name="nummer" type="xs:string" use="required"/>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="titel" type="xs:string"/>
  <xsd:element name="absatz" type="xs:string"/>
</xsd:schema>
```

4.1.2. XSL

Durch die Selbstbeschreibung der Dokumente bietet die Verwendung von XML die Möglichkeit, den darzustellenden Inhalt von der Präsentationsschicht zu trennen. Eine Anwendung kann direkt aus dem Dokument dessen Struktur auslesen und den Dokumenteninhalt anhand eines auf diese Struktur abgestimmten Präsentationsformates ausgeben. Durch die Verwendung einer Dokumentendefinition (DTD oder XML Schema) kann eine gültige Struktur zugesichert werden.

[21] Siehe [Bach2000], S. 41/42

Das W3C hat für genau diesen Anwendungsfall *XSL*, die *Extensible Stylesheet Language*, entworfen, die ebenfalls auf XML basiert. Mit dieser Sprache ist es möglich, verschiedene Präsentationsformate für bestimmte Dokumentenarten zu definieren. *XSL* besteht aus drei Teilen: „XPath“ als Ausdruckssprache für den Zugriff auf einzelne Teilbäume, „XSLT“ zur Transformation eines Dokumentes und „XSL (Formatting Objects)“, einem XML-Vokabular zur Darstellung von Formatierungsangaben[22].

4.1.2.1. XPath

XPath wurde entworfen, um auf einzelne Knoten der Baumstruktur eines XML-Dokumentes zugreifen zu können. Im Gegensatz zu den meisten anderen X-Technologien weicht XPath von der XML Syntax ab, um auch innerhalb einer Webadresse (Uniform Resource Identifier, kurz *URI*) genutzt werden zu können, beispielsweise als Parameter für eine auf HTTP basierende XML-Datenbank-Abfrage. Basis für eine XPath Adressierung bildet ein Ausdruck, der als Ergebnis je nach seiner Art eine Menge von Knoten, einen Wahrheitswert (wahr oder falsch), eine Zeichenkette oder eine Fließkommazahl zurückliefert[23]. Als Knoten werden in XPath Elemente, Attribute, Kommentare, Textknoten, Verarbeitungsanweisungen und Namensraumangaben eines XML-Dokumentes bezeichnet. Alle Knotenarten können durch spezielle XPath Anweisungen gesondert adressiert werden[24].

Neben Pfadausdrücken, welche zur Navigation innerhalb der Baumstruktur dienen, enthält XPath Funktionen zum Bearbeiten von Zeichenketten, Zahlen und Wahrheitswerten. Mit Hilfe von Pfadausdrücken können Elemente innerhalb des XML-Dokumentes anhand ihres Namens, ihrer Position und ihres Bereiches relativ zu einem Referenzelement adressiert werden. Hierzu unterteilt XPath die umgebenden Bereiche eines Elementes in mehrere Teile, die durch entsprechende Angaben adressiert werden können. In Abbildung 4.1. *Navigationsachsen in XPath* ist diese Unterteilung illustriert. Kreise stellen hierbei Knoten des XML-Baumes dar, einzelne Bereiche sind durch gepunktete Linien abgegrenzt. Die Bezeichnung der Bereiche steht anbei, durchgezogene Linien beschreiben die Struktur des Baumes.

Durch die Kombination von Knotennamen mit Knotenart, Positions- und Bereichsangaben kann somit eine exakte Navigation in einem XML-Dokument erfolgen. Zusätzlich können die von XPath zur Verfügung gestellten Funktionen genutzt werden, um eine Einschränkung der Auswahl für bestimmte Bedingungen zu erzielen. Für eine vollständige Spezifikation von XPath und die enthaltenen Funktionen wird auf [XPath 1.0] verwiesen.

[22] Vgl. http://www.w3.org/Style/XSL/ (Zugriff: 05.01.2003)

[23] Vgl. [XPath 1.0]

[24] Vgl. [Bach2000], S. 76 ff.

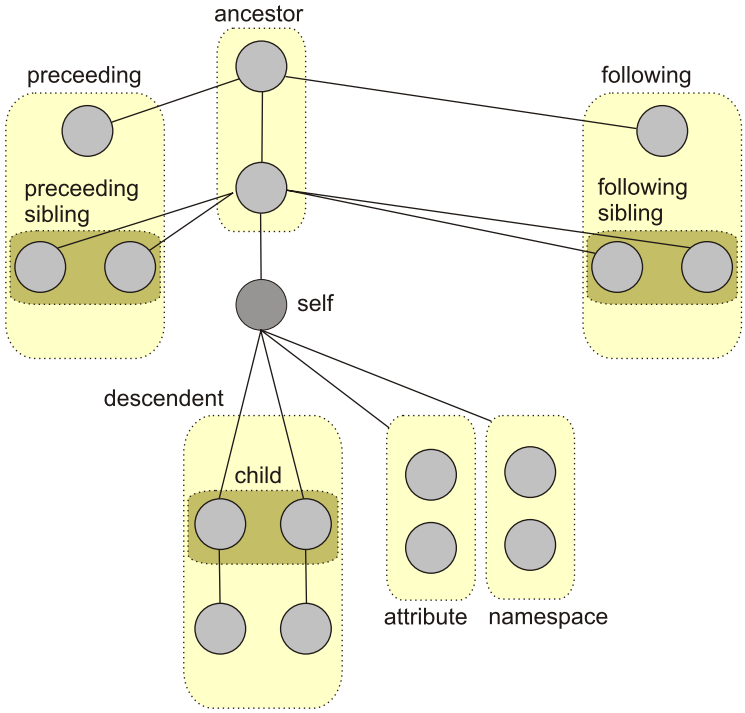

Abbildung 4.1. Navigationsachsen in XPath

Quelle für Abbildung 4.1. *Navigationsachsen in XPath*: Eigener Entwurf in Anlehnung an [Klettke2002], S. 260.

4.1.2.2. XSLT

XSLT definiert eine Sprache für die Transformation von XML in andere XML-Dokumente, wie beispielsweise *XHTML*[25], einer Neuformulierung von HTML in XML. Nachfolgend eine Beschreibung für eine XSLT Transformation:

> Eine in XSLT ausgedrückte Transformation beschreibt Regeln für die Transformation eines Quellbaums in einen Ergebnisbaum. Diese Transformation wird durch die Assoziation von Mustern mit Templates erreicht. Ein Muster wird gegen Elemente des Quellbaumes getestet. Ein Template wird instanziiert, um einen Teil des Ergebnisbaumes zu erstellen. Der Ergebnisbaum ist unabhängig vom Quellbaum. Die Struktur des Ergebnisbaums kann sich von der Struktur des Quellbaums komplett unterscheiden. Bei der Konstruktion des Ergebnisbaums können Elemente des Quellbaumes gefiltert und umgeordnet sowie beliebige Struktur hinzugefügt werden.
>
> — http://xml.klute-thiemann.de/w3c-de/REC-xslt-20020318/ (deutsche Übersetzung von [XSLT 1.0]), Zugriff: 05.01.2003

[25] Extensible HyperText Markup Language

XSLT Stylesheets sind selbst XML-Dokumente, welche neben den in XSLT definierten Elementen auch weitere, nicht in XSLT definierte Elemente enthalten können. Für den Zugriff auf einzelne Elemente bedient sich XSLT XPath. Das Zusammenspiel zwischen XSLT und XPath ist in Abbildung 4.2. *Zusammenspiel von XSLT und XPath* dargestellt.

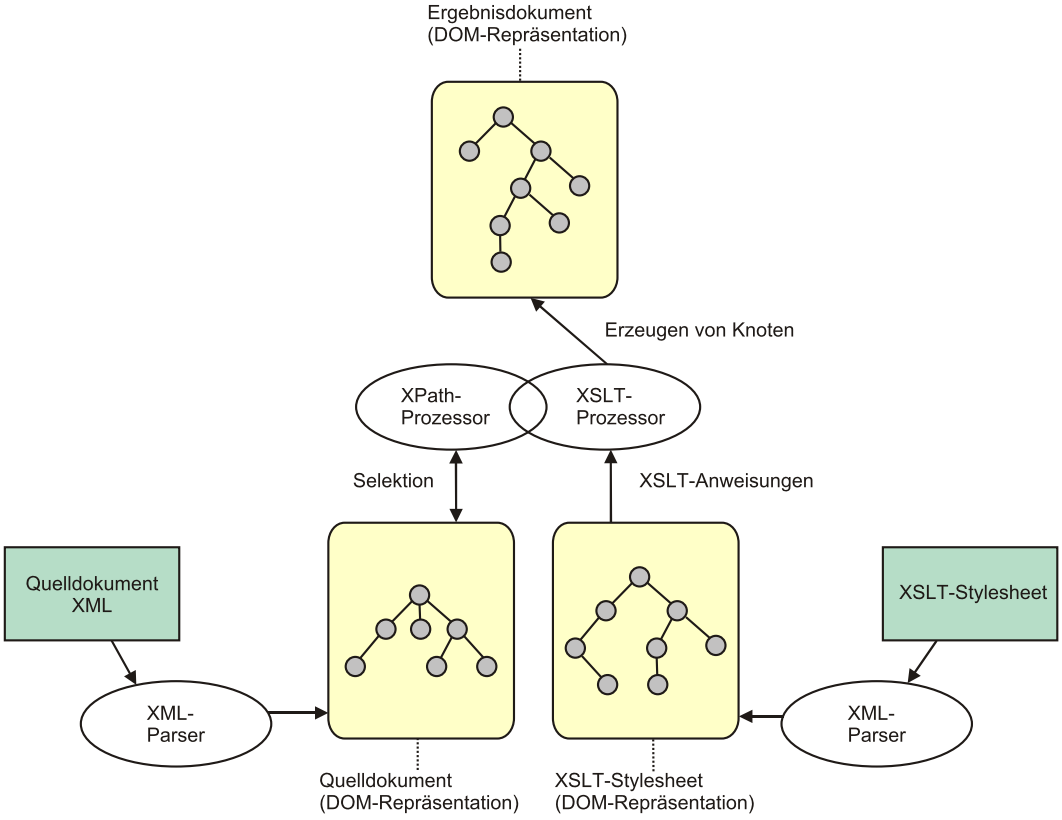

Abbildung 4.2. Zusammenspiel von XSLT und XPath

Quelle für Abbildung 4.2. *Zusammenspiel von XSLT und XPath*: Eigener Entwurf in Anlehnung an [Bach2000], S. 120.

Zur Transformation mittels XSLT wird sowohl ein XPath-Prozessor als auch ein XSLT-Prozessor benötigt. Das DigiPub-DMS nutzt für diese Zwecke *Xalan Java 2* der *Apache Software Foundation* (siehe Xalan-Java 2.4.1). Dieser Prozessor bietet eine komplette Implementation von [XSLT 1.0] und [XPath 1.0].

Für weitere Informationen zu XSLT wird auf [Bach2000] und [Kay2001] verwiesen. Die Spezifikation ist unter [XSLT 1.0] zu erhalten.

4.1.2.3. XSL (Formatting Objects)

XSL (Formatting Objects), im Folgenden durch *XSL-FO* abgekürzt, stellt den dritten Teil der XSL Spezifikation dar. XSL-FO bietet ein umfangreiches XML-Vokabular zur Formatierung

von XML-Dokumenten an und ist dazu gedacht, in Kombination mit XSLT eine Darstellung eines XML-Dokumentes zu erhalten[26], in dem zu verwendende Layoutangaben enthalten sind. Wie in „XSLT“ beschrieben, ist sowohl die Quelle für die XSLT-Transformation als auch das Ergebnis ein XML-Dokument. XSL-FO Elemente werden durch den Namensraum *fo* (http://www.w3.org/1999/XSL/Format) gekennzeichnet. Eine Darstellung der einzelnen Vokabeln in XSL-FO kann an dieser Stelle aufgrund der Komplexität nicht vorgenommen werden und ist zum Verständnis des DigiPub-DMS auch nicht erforderlich. Hierzu wird auf die Spezifikation unter [XSL 1.0] verwiesen.

Zur Darstellung eines mit XSL-FO formatierten XML-Dokumentes bedarf es einer Anwendung, die das Vokabular XSL-FO interpretieren kann. Das DigiPub-DMS nutzt für diese Zwecke *FOP* (Formatting Objects Processor) von der *Apache Software Foundation* (siehe FOP 0.20.5). Im Gegensatz zu XML- und XSLT-Parsern ist die Entwicklung im Bereich von XSL-FO-Prozessoren noch weit weniger ausgereift. Ein Vergleich der gegenwärtig am Markt verfügbaren Prozessoren zur Verarbeitung von XSL-FO ist unter [Kimber2002 Online] zu finden. FOP wird darin nicht als Prozessor für hochqualitative Dokumente empfohlen, da wesentliche Formatierungselemente noch nicht unterstützt werden. Im Rahmen des DPS-Projektes wurde jedoch entschieden, FOP zu nutzen, weil es wie Xerces-J 2.2.1 und Xalan-Java 2.4.1 von der Apache Software Foundation entwickelt wird und somit alle Produkte zur Verarbeitung von XML aus einem Hause stammen. Zudem ist FOP unter der Apache Software License frei verfügbar. Auch eine ständige Weiterentwicklung wird durch eine große Open-Source Gemeinde gewährleistet.

4.1.3. Umfang von XML

Nach Aussage der International Data Corporation (IDC) ist XML mittlerweile der „de facto“ Standard zur Integration[27]. Neben der Erstellung von Dokumenten ist ein wichtiger Aspekt von XML die Nutzung als Datenaustauschformat. Mit Hilfe von XML können für diesen Bereich einheitliche Sprachen definiert werden, ein Beispiel hierfür ist u.a. ebXML[28], eine XML-Implementation zur Standardisierung eines elektronischen Datenaustauschformates im Geschäftsumfeld. Mit den Spezifikationen *XML:RPC* [29] und *SOAP* [30] wurden Standards definiert, durch die eine Kommunikation verschiedener Anwendungen über ein einheitliches XML-Protokoll stattfinden kann. Letzteres dient als Basis der Microsoft-Architektur *.NET*. Selbst Grafiken lassen sich mittlerweile im XML-Format erstellen, als Scalable Vector Graphics (SVG). Ein Großteil der Grafiken in diesem Dokument wurde in dem SVG-Format erstellt.

Insgesamt betrachtet ist mit XML eine einfache und verständliche Technologie entstanden, deren Einzug und Verbreitung vor allem im Bereich des Internets kaum Einhalt zu gebieten

[26] Vgl. [Bach2000], S. 209 ff.

[27] Siehe http://www.idcresearch.com/getdoc.jhtml?containerId=pr2002_08_28_154542 (Zugriff: 20.01.2003)

[28] Siehe http://www.ebxml.org/ (Zugriff: 14.01.2003)

[29] XML Remote Procedure Call

[30] Simple Object Access Protocol

ist. In Abbildung 4.3. *Spezifikationen rund um XML* ist eine Übersicht dargestellt, in der verschiedene Spezifikationen und Empfehlungen zu XML aufgelistet sind. Auf diese Abbildung soll auf Grund ihrer Komplexität nicht weiter eingegangen werden, sie soll lediglich einen Überblick über den Umfang von XML und seinem Einzug in die Informatikwelt geben.

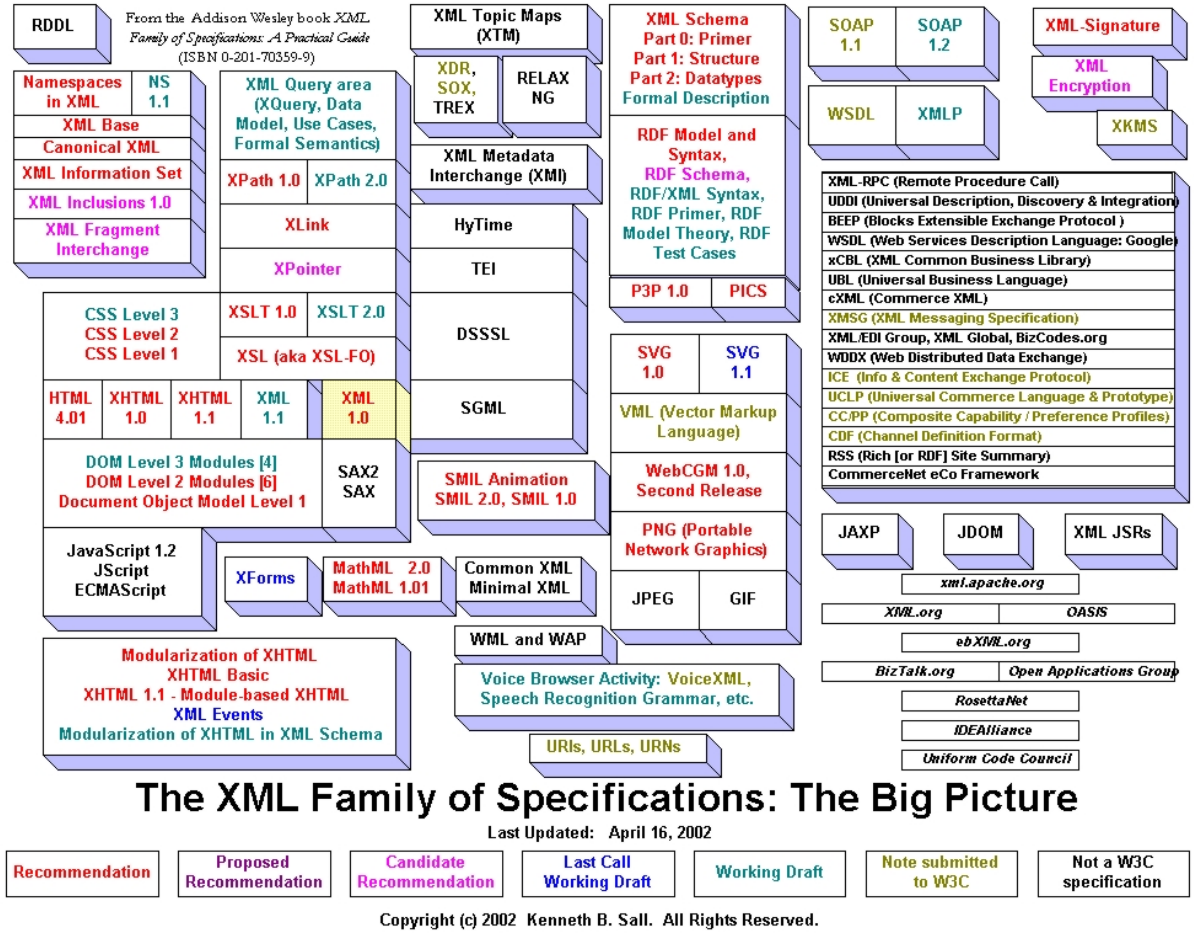

Abbildung 4.3. Spezifikationen rund um XML

Quelle für Abbildung 4.3. *Spezifikationen rund um XML*: http://wdvl.internet.com/Authoring/Languages/XML/XMLFamily/BigPicture/bigpix20a.html, Zugriff: 12.01.2003.

4.2. DocBook

Wie in Abschnitt 4.1 beschrieben, bietet der Einsatz von XML in Kombination mit XSL ein mächtiges Werkzeug zur Erstellung von Dokumenten. Diese Technik macht sich das DigiPub-DMS zu Nutze. Um XML-Dokumente formatieren zu können, müssen deren Strukturen bekannt sein. Zu diesem Zweck gibt das DigiPub-System eine DTD vor, an die sich Autoren beim Schreiben von XML Dokumenten halten müssen: die DocBook DTD.

> DocBook stellt ein System zum Schreiben strukturierter Dokumente in SGML oder XML bereit. Es ist besonders gut für Bücher und Dokumente über Computerhardware oder -software geeignet, auch wenn es auf keinen Fall hierauf beschränkt ist. DocBook ist eine SGML Dokumentendefinition (DTD). Eine XML Version ist [...] erhältlich, eine offizielle Veröffentlichung ist in Vorbereitung.
>
> [Eigene Anmerkung: am 17.06.2002 wurde DocBook Version 4.2 sowohl für XML als auch für SGML vom DocBook Technical Committee veröffentlicht (siehe http://www.oasis-open.org/committees/docbook/, Zugriff: 08.01.2003).]
>
> Da es sich um eine umfangreiche und robuste DTD handelt, und weil ihre Hauptstrukturen mit der allgemeinen Ansicht übereinstimmen, wie ein Buch zusammengesetzt sein sollte, wurde DocBook von einer großen und wachsenden Autorengemeinde übernommen. DocBook wird von einer Reihe kommerzieller Programme [...] unterstützt, und auch in einer Anzahl freier Softwareumgebungen wächst die Unterstützung zusehends. Kurz gesagt, DocBook ist eine einfach zu verstehende und weit verbreitete DTD. Jede Menge Unternehmen benutzen DocBook für Millionen von Dokumentationsseiten in verschiedenen Druck- und Onlineformaten, weltweit.
>
> — [Walsh1999], S. ix, eigene Übersetzung

DocBook wurde ursprünglich von HaL Computer Systems und O'Reilly & Associates konzipiert und wird seit 1998 vom DocBook Technical Committee weiterentwickelt[31]. Die Entscheidung, DocBook als zentrales Dokumentenformat im DigiPub-System einzusetzen, beruht vor allem auf der weiten Verbreitung[32] und der großen Anzahl von bereits existierenden Programmen, die das Erstellen von DocBook-Dokumenten vereinfachen.

Auf eine Eigenentwicklung einer DTD, wie im LIVE-System vorgenommen, wurde daher verzichtet. Für DocBook existieren bereits verschiedene XSL-Stylesheets[33] zur Erzeugung von HTML oder PDF sowie diverse Anleitungen und Einführungen (Suchergebnis mit Google (http://www.google.de) am 08.01.2003 nach „docbook tutorial“: ungefähr 25.300 Treffer). Die XSL-Stylesheets sind mehrsprachig ausgelegt (internationalisiert, vgl. Abschnitt 5.3), und DocBook wurde bereits zur Veröffentlichung von Büchern mit ISBN erfolgreich eingesetzt. Ein Beispiel hierfür ist die offizielle Dokumentation von DocBook [Walsh1999] (auch online in einer aktualisierten Version abrufbar unter [Walsh1999 Online]).

[31] Siehe http://www.oasis-open.org/committees/docbook/intro.shtml (Zugriff: 08.01.2003)

[32] Als Beispiele wären hier u.a. Sun Microsystems (siehe http://wwws.sun.com/software/xml/developers/online-pub/, Zugriff: 08.01.2002), IBM (siehe http://www-106.ibm.com/developerworks/library/l-docbk.html, Zugriff: 08.01.2002) und The Linux Documentation Project (http://www.tldp.org/, Zugriff: 08.01.2003) zu nennen.

[33] Erhältlich unter http://sourceforge.net/project/showfiles.php?group_id=21935 (Zugriff: 08.01.2003)

4.3. Datenspeicherung

Zur Speicherung von Daten benutzt das DigiPub-DMS zwei unterschiedliche Ansätze für jeweils verschiedene Datenarten. XML-Dokumente werden in einer XML-Datenbank gespeichert, auf diese Technik wird in Abschnitt 4.3.1 eingegangen. Weitere Daten, die nicht im XML-Format vorliegen, wie beispielsweise Benutzerdaten, werden in einer relationalen Datenbank gespeichert. Dieser Ansatz wird in Abschnitt 4.3.2 erläutert.

4.3.1. XML Datenspeicherung

Durch die Nutzung eines XML-Dokumentenformates wie DocBook bedarf es einer entsprechenden Speicherungsmöglichkeit, um eine Durchsuchbarkeit der Inhalte und einen optimalen Zugriff zu gewährleisten. Um eine Entscheidung für eine optimale Speicherung treffen zu können, muss zunächst analysiert werden, welche Art von XML-Dokumenten gespeichert werden soll: daten- oder dokumentenzentrierter XML-Inhalt. Bei datenzentriertem XML stehen die Daten im Vordergrund, die Strukturinformationen dienen zur Unterteilung dieser Daten. In dokumentenzentriertem XML enthalten sowohl die Daten selbst als auch die Struktur die nötigen Informationen[34]. Ist es nicht möglich, XML-Dokumente eindeutig einem dieser Bereiche zuordnen zu können, so werden diese als semistrukturierte Dokumente bezeichnet.

Im DigiPub-DMS ist die Hauptanforderung an eine XML-Datenbank die Speicherung von DocBook-Dokumenten. Bei diesem Format liegt das Hauptaugenmerk nicht allein auf den Daten, sondern ebenfalls auf der Struktur des Dokumentes. Daher ist für das DigiPub-DMS ein Speicherungssystem zu wählen, welches auf dokumentenzentriertes XML ausgelegt ist.

4.3.1.1. Datenbankauswahl

Neben speziellen XML-Datenbanken existieren für viele herkömmliche und bewährte relationale[35] oder objektorientierte[36] Datenbanksysteme Erweiterungen zur Speicherung von XML-Daten, eine Beschreibung einiger Systeme ist in [Klettke2002], Kapitel 11, einzusehen. Diese Datenbanksysteme sind nicht von Grund auf für die Speicherung von XML-Daten ausgelegt, die Hersteller machen jedoch erhebliche Fortschritte bei der Implementation von XML-Funktionalitäten[37]. Da der Fokus im DigiPub-DMS auf der Speicherung von dokumentenzentrierten Daten liegt, wurden für die Auswahl nur Datenbanken in Betracht gezogen, die speziell auf XML ausgelegt sind (diese Datenbanken werden auch als *native* XML-Datenbanken bezeichnet). Empfohlen wird der Einsatz einer nativen XML-Datenbank für dokumentenzentrierte XML-Dokumente u.a. in [Klettke2002] und [Bourret Online].

[34] Vgl. [Klettke2002], S. 100/101

[35] Beispiele hiefür sind u.a. Oracle 9i (siehe http://www.oracle.com/ip/deploy/database/oracle9i/) oder der Microsoft SQL-Server 2000 (siehe http://www.microsoft.com/sql/default.asp), beide Zugriff: 06.01.2003

[36] Ein Beispiel hierfür ist FastObjects der Poet Software GmbH (siehe http://www.fastobjects.com/de/FO_DE_Produkte_FastObjectsT7_Body.html, Zugriff: 06.01.2003)

[37] Vgl. [IS 12/01]

Für das DigiPub-DMS wurde der *Tamino XML Server* der Software AG (siehe Tamino 3.1.2.1) gewählt, der speziell zur Speicherung von XML ausgelegt ist. Die Vorteile dieses Datenbanksystems kommen insbesondere bei der Speicherung von dokumentenzentriertem XML zum Tragen[38]. Dies wiederum ist genau das Einsatzgebiet dieser Datenbank im DigiPub-System.

Tamino wurde bereits im LIVE-Projekt erfolgreich und mit positiver Erfahrung eingesetzt, ein Vergleich zu verschiedenen XML-Datenbanken wurde ebenfalls in diesem Projekt vorgenommen[39]. Als Alternative zu Tamino wurde im Rahmen des DPS-Projektes jedoch ein weiteres System in Betracht gezogen, und zwar *Infonyte DB* der Firma Infonyte GmbH. Dieses System basiert auf *PDOM*, einer abgewandelten Version vom *Document Object Model* (DOM, vgl. Abschnitt 5.4).

> Infonyte-PDOM: Ein skalierbares DOM
>
> Infonyte-PDOM ist eine persistente Implementierung des W3C DOM-API für XML. Mit Infonyte-PDOM können Sie XML-Dokumente im Gigabyte-Bereich erzeugen, bearbeiten, abfragen und modifizieren - alles bei gleichbleibender moderater Hauptspeicherbelastung.
>
> — http://www.infonyte.com/de/prod_pdom.html, Zugriff: 08.01.2003

DOM ermöglicht den kontextabhängigen Zugriff auf einzelne XML-Knoten der Baumstruktur. Für diesen Zweck wird der XML-Baum im Hauptspeicher abgebildet, wodurch eine Speicherauslastung von ungefähr dem zehn- bis zwanzigfachen der Größe des Originaldokumentes erfolgt[40]. Bei großen Dokumenten kann es somit schnell zu einer erheblichen Speicherauslastung kommen. Das von der Infonyte GmbH entwickelte *PDOM* erlaubt die dauerhafte Speicherung dieser DOM-Strukturen (z.B. auf einem Datenträger) und bietet dieselben Zugriffsschnittstellen wie DOM an[40] [41]. Eine auf *PDOM* basierende Technik ist das ebenfalls von der Infonyte GmbH entwickelte *PXSLT*.

> Infonyte-PXSLT: XSLT für sehr große Datenvolumen
>
> Infonyte-PXSLT ist ein XSLT-Prozessor, der Datenvolumen im Gigabyte-Bereich verarbeitet. [...] Infonyte-PXSLT setzt auf einer persistenten Implementierung der DOM-Schnittstelle auf (Infonyte-PDOM).
>
> — http://www.infonyte.com/de/prod_pxslt.html, Zugriff: 08.01.2003

PXSLT basiert auf dem XSLT-Prozessor Xalan und bietet erstmals die Möglichkeit, XSLT-Verarbeitung direkt auf einer persistenten Repräsentation der XML-Daten vorzunehmen[40].

[38] Vgl. [Klettke2002], S. 347

[39] Verglichen wurden die Datenbanken eXist, DBXML (jetzt Apache Xindice), Excelon und Tamino. Der Vergleich ist erhältlich in der LIVE Dokumentation (siehe http://live.fhnon.de:8080/live2002/Doku.pdf, Zugriff: 08.01.2003).

[40] Vgl. [WI 5/02]

[41] Vgl. [Klettke2002] S. 223 ff.

Durch die Möglichkeit, auf kontextabhängige Teile eines XML-Baumes zugreifen zu können, ohne das komplette Dokument über DOM in den Hauptspeicher zu laden, können die Hardwarevoraussetzungen für die XSL-Transformation erheblich verringert werden. Die *Infonyte DB* zeigt ihre Vorteile allerdings vor allem bei der Bearbeitung von sehr großen Dokumenten (in [WI 5/02] werden Größenordnungen im Bereich mehrerer 100 Megabyte und sogar Gigabyte genannt). Das DigiPub-System ist zur Speicherung von XML-Dokumenten und zur Auslieferung dieser als PDF-Datei gedacht, wobei die Dokumente weitaus kleiner sind. Diese Diplomarbeit hat im XML-Format eine Größe von 416 Kilobyte und bei der Verwendung eines Stylesheets zur Erstellung von Büchern im DIN A4 Format 120 Seiten. Eine gedruckte Version (beidseitiger Druck) auf 80g/m2 Papier erreicht somit eine Gesamtdicke von ca. 0,65 cm. Ein Dokument im DigiPub-System sollte eine Größe haben, in der ein Druck und eine Bindung des Dokumentes technisch noch realisierbar ist. Die maximale Dicke eines druckbaren Dokumentes ist beispielsweise für das Klebebindesystem BQ-140 Perfect Binder[42], welches zum jetzigen Zeitpunkt (08.01.2003) als einer der möglichen Kandidaten für den Einsatz im DigiPub-System in Betracht gezogen wird, 3,05 cm. Somit könnte diese Diplomarbeit theoretisch ca. 4,7-mal innerhalb eines Buches vorkommen, wodurch der XML-Inhalt eine Größe von annähernd 2 Megabyte erreichen würde. Dokumente dieser Größe können problemlos durch den Einsatz von DOM bearbeitet werden. Auch der Einsatz von PXSLT ist bei dieser Größe nicht erforderlich.

4.3.1.2. Datenbankzugriff

Als Schnittstelle für den Zugriff auf Tamino wird die API der *XML:DB* Initiative genutzt. Die Software-AG bietet, neben den weiteren Java API's „TaminoClient API“ und „Tamino API for Java“, eine geeignete Implementation hierzu an. Die XML:DB Working Group, die sich mit der Konzeption der XML:DB-API beschäftigt, beschreibt das Ziel dieser API wie folgt:

> Das Ziel der Arbeitsgruppe ist die Entwicklung einer API für XML Datenbanken. Diese API soll produktunabhängig sein, um die Nutzung mit einer größtmöglichen Anzahl von Datenbanken zu unterstützen.
> — http://www.xmldb.org/xapi/index.html, Zugriff 05.01.2003, eigene Übersetzung.

Der Vorteil der XML:DB-API liegt somit darin, dass diese nicht auf eine spezielle Datenbank abgestimmt ist, sondern eine einheitliche Zugriffsschnittstelle bietet, ähnlich wie JDBC für Java und relationale Datenbanksysteme. Obwohl die XML:DB-API noch den Status eines Working-Drafts (Entwurf) hat, existieren neben Tamino mehrere weitere XML-Datenbanksysteme, welche Implementationen der XML:DB-API anbieten. Beispiele hierfür sind u.a. *Apache Xindice*[43] oder *eXist*[44], welches beide Open-Source Datenbanken sind.

[42] Angeboten durch Océ Deutschland GmbH

[43] Apache Xindice: siehe http://xml.apache.org/xindice/ (Zugriff: 05.01.2003)

[44] eXist: siehe http://exist-db.org/ (Zugriff: 05.01.2003)