XML und darauf basierende Standards: Die neuen Auszeichnungssprachen des Web

Robert Tolksdorf

Die XML-Technologie wird die Basis für zukünftig dominierende Darstellungsformen von Dokumenten und Daten im Web sein. Sie erlaubt die Definition einer anwendungsspezifischen Grammatik von Auszeichnungssprachen bestehend aus Elementen und deren Attributen. Aufbauend auf XML werden momentan verschiedene Auszeichnungssprachen, beispielsweise für Formeln, Multimediapräsentationen oder Datenbankanfragen, standardisiert. Für die Verarbeitung von XML-Dokumenten steht ein Objektmodell zu deren Repräsentation bereit und eine Transformationssprache zu ihrer Übersetzung. In diesem Beitrag geben wird einen Überblick auf die genannten Techniken und erläutern ihre Zusammenhänge.

Summary on page 435

Die Welt des Web verändert sich durch einen neuen Standard, XML. Rund um XML entstehen verschiedenste weitere Standards und Akronyme. Für den nicht in die Entwicklung involvierten Beobachter bleibt oft unklar, was XML, XSL, XQL, XHTML etc. eigentlich darstellen, in welchen Zusammenhängen die Konzepte stehen und welche Bedeutung sie haben. Dieser Betrag soll einen Überblick geben und begriffliche Klarheit schaffen.

Der Beitrag gliedert sich in drei Teile. Wir beschreiben zunächst die XML-Technologie mit ihren verschiedenen Konzepten. Im zweiten Teil stellen wir eine Reihe von Auszeichnungssprachen vor, die allesamt auf XML basieren und teilweise schon standardisiert sind oder kurz davor stehen.[1] Abschließend beschreiben wir die Verarbeitung von XML-Dokumenten durch regelgesteuerte Transformationen.

1 Von HTML zu XML

Anlass für die Entwicklung von XML waren die Beschränkungen von HTML. Die Auszeichnungssprache für Web-Dokumente lässt sich in Definitionen darstellungsorientierter Elemente wie < I > . . . < /I > für kursive Schrift und inhaltsorientierter Tags[2] wie < ADDRESS > . . . < /ADDRESS > unterteilen. Das *logische Markup* mit inhaltsorientierten Markierungen ist der rein layoutorientierten Auszeichnung überlegen, da sie eine gewissen semantischen Gehalt hat.

Es lag nahe, HTML mit einer Fülle weiterer logischer Tags anzureichern. Im HTML-4-Standard (http://www.w3.org/TR/REC-html40[3], [2]) wurde dies beispielsweise durch Tags wie < PERSON > versucht. Diese Entwicklungsrichtung ist aber zum Scheitern verurteilt, da die Menge der festgelegten Tags vom Umfang oder von der Granularität her – beispielsweise könnte man bei einer Person ja wiederum Vor- und Nachnamen separat

Robert Tolksdorf
Technische Universität Berlin, FB 13, Informatik,
KIT/FLP, FR 6–10, Franklinstraße 28/29, 10578 Berlin
e-mail: tolk@cs.tu-berlin.de

[1] Während der Standardisierung im World Wide Web Consortium (W3C) durchläuft ein Dokument verschiedene Stufen der Überarbeitung und Abstimmungen vom einfachen Vorschlag für eine Normierung als *Note* über *Working Draft* und *Proposed Recommendation* zur *W3C-Recommendation* als verabschiedetes Standarddokument.

[2] Die Web-Entwicklung ist englischsprachig dominiert. Die offiziellen W3C-Standards sind ausschließlich in englischer Sprache veröffentlicht (http://www.w3.org/Consortium/Translation), lediglich zu Informationsszwecken und unter Wahrung des W3C-Copyrights sind freiwillige Übersetzungen Dritter erlaubt. In diesem Beitrag verwenden wir die gebräuchlichen oder standardisierten englischsprachigen Begriffen.

[3] Die Web-Entwicklung ist online dokumentiert. W3C-Standards werden lediglich als Online-Dokumente veröffentlicht. In diesem Beitrag beziehen wir uns daher zumeist URLs statt auf herkömmliche Literaturstellen.

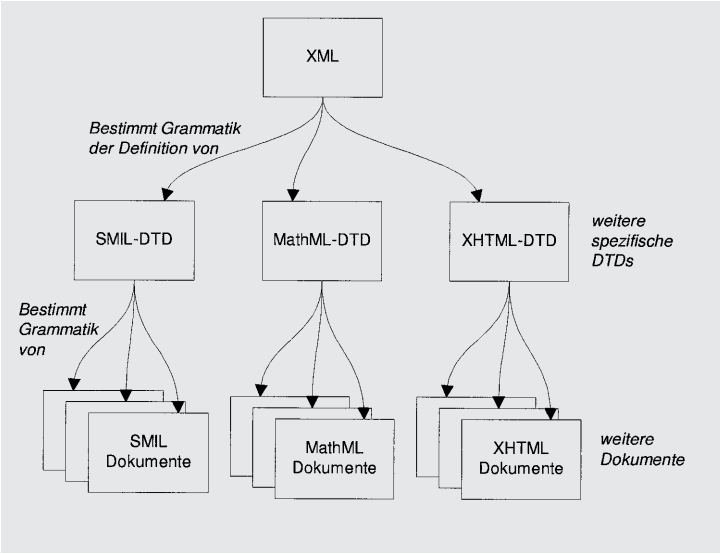

Abb. 1. XML als Metagrammatik für Auszeichnungssprache

auszeichnen – prinzipiell nicht allen Nutzeranforderungen genügen kann.

2 Grammatiken mit XML

Das World Wide Web Consortium (W3C) – die vom MIT, der INRIA und der japanischen Keio Universität getragene Standardisierungsorganisation des Web – erkannte die Sackgasse, in die die HTML-Weiterentwicklung geraten war. Sie erarbeitete einen Standard, mit dem sich eine Web-Auszeichnungssprache anwendungsspezifisch definieren lässt: XML, die *Extensible Markup Language*.

Mit diesem Begriff fängt gleichzeitig die Begriffsverwirrung an: XML ist keine erweiterbare Auszeichnungssprache – sie ist eine standardisierte Sprache in der sich die Syntax von Auszeichnungssprachen notieren lässt. Formaler ausgedrückt ist XML eine Metagrammatik für kontextfreie Grammatiken – also die Grammatik einer Sprache mit der sich die Regeln von Grammatiken notieren lassen (s. Abb. 1).

Diese Grammatikdefinition wird im XML-Umfeld DTD („d*ocument type definition*“) genannt. Dokumente werden so durch ihre Grammatik typisiert: Die Sätze der durch die DTD definierten formalen Sprache bilden mögliche Dokumente.

Der Begriff DTD ist keineswegs neu – er wurde für SGML geprägt [1], und in der Tat übernimmt XML die Konzepte von SGML, magert sie allerdings so weit ab, dass ein verständlicher und einfach zu verwendender Mechanismus entsteht.

HTML als eine Auszeichnungssprache für Dokumente besitzt eine bestimmte DTD. So sind auch die verschiedenen HTML-Versionen syntaktisch durch die Vorgabe einer SGML-DTD definiert worden. Mangels leicht verfügbarer SGML-Werkzeuge wurde dieser Aspekt allerdings nie weiterverfolgt und genutzt. Mit der Definition von XML wird ein Abschied von der bisherigen Sonderstellung von HTML eingeleitet: HTML wird eine unter sehr vielen Auszeichnungssprachen, die jeweils durch eine XML-DTD definiert sind. In der Tat ist mit XHTML eine entsprechende XML-DTD für HTML in Arbeit, auf die wir weiter unten eingehen werden.

2.1 Elemente und Attribute

Verfolgen wir als Beispiel die Problematik der Auszeichnung von Anschriften. Wir möchten, dass in einem Adressbuch die Namen von Personen markiert werden (aber nicht Vor- und Nachname getrennt ausgezeichnet wird). Daraufhin ist evtl. eine Straße mit Hausnummer oder ein Postfach zu nennen, dann ein Ort mit Postleitzahl, Namen und optional eine Landesangabe.

Als XML-DTD anschrift.dtd lässt sich diese Sprache wie folgt definieren:

```
< ?xml version = "1.0" encoding = "UTF-8"? >
< ! ELEMENT ADRESSBUCH ANSCHRIFT* >
```

```
< !ELEMENT ANSCHRIFT
  (NAME, (STRASSE | POSTFACH) ?, ORT) >
< !ELEMENT NAME ANY >
< !ELEMENT STRASSE ANY >
< !ELEMENT POSTFACH ANY >
< !ELEMENT ORT EMPTY >
< !ATTLIST ORT
   PLZ CDATA #REQUIRED
   NAME CDATA #REQUIRED
   LAND CDATA #IMPLIED >
```

Die XML-DTD beginnt mit einer Markierung, dass es sich um eine XML-DTD handelt. Die Grammatik besteht aus einer Reihe von *Elementen*, die einerseits Terminale der beschriebenen Sprache definieren – sie können als Tags auftreten – und andererseits Produktionen als Regeln festlegen können.

Eine Liste von Adressen wird mit <ADRESSBUCH > und < /ADRESSBUCH > umklammert und kann eine beliebige Anzahl von Adressen enthalten – der Stern * erlaubt 0. .n maliges Auftreten von ANSCHRIFT-Elementen. < !ELEMENT ANSCHRIFT leitet die Definition eines Elements ANSCHRIFT ein. Dadurch wird die Markierung < ANSCHRIFT > . . . </ANSCHRIFT> Bestandteil der definierten Sprache. Für das Element existiert eine Regel, die beschreibt, dass innerhalb der < ANSCHRIFT > Markierung als Erstes ein < NAME > Element stehen muss. Ihm kann ein Element < STRASSE > oder < POSTFACH > folgen. Abschließend muss eine Element <ORT> vorhanden sein.

Für die Formulierung dieser Regeln bietet XML Gruppierungen mit ( ), Alternativen (|) und Sequenz (,) und Kardinalität der Wiederholung (0. .*: *, 1. .*: + , 0. .1: ?).

Die im Beispiel folgenden Regeln legen fest, dass innerhalb von < NAME > . . . </NAME > und den weiteren zwei Elementen beliebiger Text stehen darf und verwenden dazu die konstante „Regel“ ANY.

Bei < ORT > bestimmt die < !ELEMENT-Regel zunächst, dass keine Elemente in einem ORT-Element vorkommen dürfen. Im XML-Dokument kann an Stelle eines leeren Markierungspaars < ORT > < /ORT > übrigens die Abkürzung < ORT/ > treten.

Um die weiter oben genannten Eigenschaften eines Ortes auszuzeichnen, verwendet die Grammatik eine weitere Möglichkeit in XML-Dokumenten, die *Attribute*. Die Regel < !ATTLIST ORT. . . beschreibt eine Reihe von Attributen, die für das Element verwendet werden können, deren Typ und ob das Attribut optional oder mandatorisch ist.

Das Beispiel verlangt die Attribute PLZ und NAME – durch #REQUIRED markiert – und optional das Attribut LAND – mit #IMPLIED definiert. NAME lässt sich ohne weiteres als Attributname verwenden, obwohl ein Element < NAME > vorher definiert wurde – lediglich bezogen auf ein Element müssen die Attribute eindeutig bezeichnet sein. Die Typen der Beispielattribute sind jeweils Zeichenketten; in XML bezeichnet CDATA diesen Typ.

Mögliche Typen für Attribute in XML umfassen neben anderen:

- Zeichenketten beliebigen Inhalts, genannt CDATA.
- Namen und eindeutige Bezeichner. Der Typ ID beispielsweise beschreibt einen aus Zeichen gebildeten Bezeichner, der nur einmal in einem Dokument auftreten darf. IDREF bezeichnet eine Referenz auf einen solchen Bezeichner.
- Aufzählungstypen mit optionalem Vorgabewert. Beispielsweise legt < !ATTLIST PERSON GESCHLECHT (mann|frau) "frau" > fest, dass im XML-Dokument < PERSON GESCHLECHT = "mann" > notierbar ist und implizit GESCHLECHT zunächst den Wert „frau“ hat.

Ein XML-Dokument, das der Beispiel-DTD entspricht, könnte wie folgt aussehen:

```
< ?xml version = "1.0"? >
< !DOCTYPE ADRESSBUCH SYSTEM "anschrift.dtd" >
< ADRESSBUCH >
  < ANSCHRIFT >
    < NAME > Robert Tolksdorf < /NAME >
    < STRASSE > Franklinstr. 28/29 < /STRASSE >
    < ORT PLZ = "10587" NAME = "Berlin"/ >
  < /ANSCHRIFT >
< /ADRESSBUCH >
```

Das erste Element besagt, dass dieses Dokument nach den Regeln von XML in der Version 1.0 geschrieben ist. Das < !DOCTYPE-Element gibt an, dass der Typ dieses Dokuments, also seine Syntax den Regeln in der XML-DTD anschrift.dtd folgt und das Startsymbol ADRESSBUCH ist. Dann erst folgt das eigentliche Dokument mit den oben definierten Elementen.

Der Verweis auf die DTD kann auch eine vollständige URL sein, die beispielsweise auf eigene, öffentliche Grammatiken oder standardisierte DTDs

verweist. Ein Dokument gilt als *valid*, wenn es eine DTD nennt und dieser entspricht. Es ist *well-formed*, wenn es einer Reihe allgemeiner Regeln für XML-Dokumente genügt, beispielsweise, dass die Schachtelung von Elementen balanciert ist. Eine XML-Dokument kann auch auf die Nennung einer DTD verzichten. Man unterscheidet zwischen validierenden und nichtvalidierenden XML-Parsern, je nachdem, ob sie die Regeln der DTD überprüfen oder nur auf Wohlgeformtheit prüfen.

2. 2 Entitäten

Elemente definieren Terminale der Sprache und zugleich Produktionen für den von ihnen umschlossenen Inhalt. *Entities* dienen in einer XML-DTD ebenfalls zur Definition von Terminalen und Nichtterminalen, allerdings werden sie lediglich textuell expandiert.

Die *General Entities* sind Kürzel, die im XML-Dokument verwendet werden können und zu dem für sie definierten Ersetzungstext expandieren. Ein Kürzel TubStrasse wird in einer DTD wie folgt definiert: < !ENTITY TubStrasse "Franklinstr. 28/29" > . Im Dokument verwendet man dieses Kürzel, indem es mit & eingeleitet und mit ; abgeschlossen wird:

```
< STRASSE > & TubStrasse; < /STRASSE >
```

Die *Parameter Entities* dagegen sind nur für die Verwendung innerhalb der DTD vorgesehen. Man kann sie als Nichtterminale verwenden, wenn ihre Expansionen einen Teil einer Regel ergibt. So ließe sich beispielsweise verlangen, dass alle Änderungen am Bestand der Adressen im Beispiel mit Datum und Bearbeiter versehen werden müssen. Dementsprechend soll es bei allen Elementen die Attribute CHANGEDATE und CHANGEPERSON geben. Man würde die Attributdefinitionen einmalig festlegen und dabei das Kürzel mit dem Zeichen % von einer General Entity abgrenzen:

```
< !ENTITY % ChangeAttr"
  CHANGEDATE CDATA #REQUIRED
  CHANGEPERSON CDATA #REQUIRED
" >
```

ChangeAttr kann nun mit %ChangeAttr; zu den beiden definierten Zeilen innerhalb der DTD expandieren:

```
< !ATTLIST POSTFACH %ChangeAttr; >
< !ATTLIST ORT %ChangeAttr;
  PLZ CDATA #REQUIRED
  NAME CDATA #REQUIRED
  LAND CDATA #IMPLIED >
```

Der XML-Standard definiert eine Reihe weiterer Konzepte wie Verweise auf externe Teile von DTDs, auf externe Kürzeldefinition, Kürzeln für Sonderzeichen, den Umgang mit verschiedenen Zeichenkodierungen und weitere Details zur Expansion von Entitäten und Regeln, sowie zum Umgang mit Leerraum.

2. 3 Schachtelung von XML-Sprachen

Jede XML-Grammatik, eine DTD, definiert durch die Elemente einen bestimmten Namensraum, in dem die Tag-Bezeichner eindeutig sind. Es kann Dokumente geben, die mehrere XML-Grammatiken verwenden, beispielsweise um in einem Textdokument eine spezielle Markup-Sprache für Formeln oder Grafiken zu verwenden. Damit die jeweils gültigen Namensräume identifiziert und Namenskonflikte vermieden werden können, definiert der W3C-Standard *Namespaces in XML* (http://www.w3.org/TR/REC-xml-names) Namensräume.

Namensräume werden in einem XML Dokument durch Zuweisen eines *Uniform Resource Name*, URN an ein Kürzel definiert. Diese Bindung entsteht durch die Verwendung eines Attributs xmlns:*Kürzel* bei dem Tag, innerhalb dessen dieser Namensraum verwendet werden kann. Ein Beispiel wäre die Definition eines Attributs xmlns: edi = "http://ecommerce.org/schema".

Das so definierte Kürzel edi bezeichnet nun einen Namensraum, dessen „Inhalt“ alle Tags sind, die in der durch den URN bezeichneten Grammatik definiert sind. Ein URN ist ein eindeutiger Name im Web-Umfeld, hinter dem sich im Gegensatz zu einer URL jedoch keine Web-Seite befinden muss. Um ein Tag einem Namensraum zuzuordnen, wird ihm das Kürzel als Präfix vorangestellt, also beispielsweise als < edi:price > 1.02 < /edi:price > . Auf diese Weise lassen sich in einem Dokument Tags aus unterschiedlichen Grammatiken gleichzeitig verwenden. Im Beispiel entsteht also kein Namenskonflikt mit anderen Price-Tags, wie < lottery:price > cheap toy < /lottery:price > .

In einem XML-Dokument lässt sich die Gültigkeit von Namensräumen an Elemente binden. Nur

innerhalb eines Tags, das ein xmlns-Attribut trägt, ist das entsprechende Präfix definiert. Zusätzlich kann ein Namensraum als Vorgabewert ohne Präfix definiert werden. Dazu ein Beispiel aus dem Standard:

```
< ?xml version = "1.0"? >
  < book xmlns = "urn:loc.gov:books"
    xmlns:isbn = "urn:ISBN:0-395-36341-6" >
      <title>Cheaper by the Dozen</title>
      < isbn:number > 1568491379
      < /isbn:number >
      <notes>
        < p xmlns = "urn:w3-org-ns:HTML" >
          This is a < i > funny < /i > book!
        </p>
      </notes>
  </book>
```

Innerhalb von < book > wird urn:loc.gov:books als Vorgabe-Namensraum definiert, indem das xmlns-Attribut kein Präfix definiert. Zusätzlich ist der Namensraum bei urn:ISBN:0–395–36341–6 an das Präfix isbn gebunden. Im < notes > Tag (das aus dem Vorgabe-Namensraum stammt), soll nun HTML verwendet werden können. Dazu definiert das xmlns-Attribut von < p > einen darin gültigen Vorgabe-Namensraum, der die HTML-Definition identifiziert. Dementsprechend bedeutet das < i > Tag die aus HTML gewohnte Auswahl kursiver Schrift.

4 XML-Darstellungsattribute festlegen

Da XML-Dokumente auf selbstdefinierten Grammatiken basieren, kann es keinen allgemeinen Standard für ihre Darstellung geben. Während bei HTML die Darstellung vom Standard zumindest ansatzweise vorgegeben war, existiert bei XML keinerlei Semantik von Tags in Bezug auf ihre Darstellung.

Mit den *Cascading Style Sheets* CSS (http://www.w3.org/Style/CSS) gibt es seit einiger Zeit ein Standard zur Festlegung von Attributen der Darstellung von HTML-Tags. Ein solches Style-Sheet kann beispielsweise die gewünschte Vorder- und Hintergrundfarbe von Tabellenzellen getrennt vom eigentlichen Dokument bestimmen. Damit wird die vorteilhaften Trennung von Inhalts- und Darstellungsauszeichnung begünstigt. Mit den letzten Versionen der wichtigsten Web-Browser hat CSS auch eine gewisse reale Verbreitung gefunden.

Für XML -Dokumente wurde mittlerweile der CSS-Ansatz adaptiert. Technisch ist dies auch sehr naheliegend – ein CSS gibt einen bestimmten Tag-Namen an und belegt einen Satz definierter Darstellungseigenschaften. Anstatt sich auf den Satz von HTML-Tags zu beschränken, erlaubt die CSS-Verwendung im XML-Kontext einfach beliebige Tag-Namen.

Soll von einem selbstdefinierten XML-Tag < price > umschlossener Text in weißer Schrift auf schwarzem Grund dargestellt werden, reicht die folgende CSS-Festlegung:

```
price {
 color: white;
 background-color: black;
}
```

Dabei ist color ein in CSS definiertes Attribut eines Tags und white ein vordefinierter Wert. Der W3C-Standard *Associating Style Sheets with XML documents* bei http://www.w3.org/TR/xml-stylesheet definiert dieses Verhalten und beschreibt wie sich ein solches Style-Sheet an ein XML-Dokument binden lässt.

2. 5 Referenzen auf Dokumente: XLINK/ XPOINTER

Hypermediale Dokumente zeichnen sich durch Verweise auf andere Dokumente aus. In HTML wird eine sehr einfache Form des Links mit dem < A > Element eingeführt, nämlich ungetypte, unidirektionale Beziehungen. HTML-Browser stellen die Beschriftungen solcher Links entsprechend dar und erlauben das Verfolgen des Links durch Anklicken. Zwei Standards beschäftigen sich im XML-Umfeld mit Links: *XLink* zur Definition von Verweisen und *XPointer* zur Beschreibung von Verweiszielen.

Im XML-Kontext stellt sich die Frage der Repräsentation von Links gesondert, da hier ja eigene Elemente definiert werden können und diese evtl. Link-Funktionalität haben sollen. Die *XML Linking Language* (XLink) beschreibt, wie Link-Elemente markiert werden und welches Verhalten Browser für sie implementieren sollen (http://www.w3.org/TR/WD-xlink).

2. 5. 1 XLink

Nach XLink ist ein Link in einem XML-Dokument genau daran zu erkennen, dass es sich um das Start-Tag eines Elements handelt, das das Attribut

xml:link trägt. Darauf bauen weitere Attribute auf, die verschiedene Arten von Links identifizieren und zusätzliche Informationen zu einem Link ermöglichen. Hat das xml:link-Attribut den Wert simple, handelt es sich um einen einfachen Link, der wie das < A > Element in HTML genau ein Dokument mit genau einem anderen verbindet. Ist der Wert extended, liegt ein erweiterter Link vor, der ein Dokument mit mehreren anderen verbinden kann.

Jedes Link-Element muss eine URL-Referenz im Attribut href definieren. Das Attribut inline legt fest, ob das Zielobjekt zu diesem Dokument gehört oder extern ist. Solche eingebetteten Verweise finden sich in HTML beispielsweise beim < IMG > Tag.

Weitere Attribute können Auskunft über die Bedeutung des Links geben. Das Attribut role typisiert den Link durch einen Namen. Das Verhalten eines Browsers kann das Show-Attribut steuern: Der Wert embed bettet das verwiesene Dokument bei Traversierung in das vorhandene ein, replace ersetzt die Anzeige des aktuellen Dokuments wie in normalen HTML-Browsern und new erzeugt eine neue Anzeige für das verwiesene Dokument, so wie das in HTML mit Framesets erreicht werden kann. Das Attribut actuate steuert, wann die verwiesene Ressource angezeigt wird: auto bedeutet sofortige Anzeige (wie bei </IMG> in HTML) und user legt fest, dass erst auf Nutzerinteraktion hin der Link verfolgt werden soll – z. B. um nicht sofort einen umfangreichen Videoclip zu laden. Für die anwendungsspezifische Steuerung des Umgangs mit Links ist das Attribut behaviour reserviert, das direkte Anweisungen an den Browser als Wert tragen kann.

Erweiterte Links verweisen auf mehrere andere Ressourcen. Ein Element mit einem Wert von extended für das xml:link Attribut zeichnet eine Gruppe von Links aus – ein erweiterter Link stellt eine 1:n-Beziehung zwischen Dokumenten dar. In der Gruppe sind die einzelnen Links mit durch den Wert locator des xml:link-Attributs kenntlich. Ein Beispiel wäre ein Verweisgruppe für ein Dokument, das als Zusammenfassung und im Volltext vorliegt:

```
< zitat xml:link = "extended" inline = "false" >
  < verweis xml:link = "locator"
    role = "zusammenfassung"
    href = "http://reports.com/abs1.htm" >
  < verweis xml:link = "locator"
    role = "volltext"
    href = "http://reports.com/text1.htm" >
</zitat>
```

XLink hebt durch die erweiterten Links und der Typisierung die bisherigen Einschränkungen des Web als Hypertext teilweise auf. Die einfachen Links zeichnet einen klaren Migrationsweg von HTML aus vor. Allerdings gab es auch in HTML 4 durch das < LINK > Tag schon einen Versuch zu einer Anreicherung der Linkfähigkeiten des Web, der nicht von Browser-Herstellern aufgenommen wurde.

2. 5. 2 XPointer

Die *XML Pointer Language* (XPointer) ist eine Sprachdefinition für Ausdrücke zur Adressierung von Zielstellen eines Links innerhalb eines XML-Dokuments. In HTML werden Verweisziele mit dem Tag < A NAME = "Ziel" > markiert und können durch Anhängen des Namens als Fragment an die URL des Dokuments direkt adressiert werden: http://eine.url/dokument.html#Ziel. Mit XPointer ist nicht notwendig ein Zielnamen zu verwenden – die XPointer Ausdrücke erlauben eine Art Navigation im Dokument.

Ein Beispiel soll dies deutlich machen: id(Ziel).following(1,#element) ist ein Ausdruck in XPointer. Sein Wert ist ein Zeiger auf ein Element im Baum, den ein XML-Dokument beschreibt. Ausgangspunkt ist die Stelle des Elements, das das Attribut id mit dem Wert „Ziel“ trägt. Darauf angewandt wird die following Funktion, die hier das Erste danach folgende Element liefert. Anstelle mit #element auf ein Element zu verweisen, lassen sich mit entsprechenden Kürzeln beispielsweise so auch Stellen von Kommentaren oder Texten bezeichnen.

Aber auch durch Namen identifizierte Element lassen sich auswählen, so bezeichnet root().child(2,ANSCHRIFT) die zweite Adresse in einem Dokument nach unserer oberen Beispiel-DTD (Tabelle 1).

XPointer erlaubt eine sehr mächtige Beschreibung von Verweiszielen in XML Dokumenten, da durch die relative Adressierung der gesamte Dokumentenbaum traversierbar ist. Die Spezifikation befindet sich im Status Working Draft. Eine Implementierung der Interpretation von XPointer wird aber nicht sehr komplex sein, da das Aufsuchen von Stellen in einem XML-Dokument durch die Stan-

Die wichtigsten Ausdrücke, die XPointer definiert

Tabelle 1

	Schlüsselwort	Bedeutung
Absolute Ausdrücke	root()	Das Wurzelelement des Dokuments
	id(Name)	Die Stelle des Elements, das Name als Wert des id Attributs trägt
	html(Name)	Die Stelle eines < A > Elements, das Name als Wert des name Attributs trägt
Relative Ausdrücke	child	Die direkten Kinderknoten eines Elements
	descendent	Die Menge der direkten und indirekten Kinder eines Elements
	ancestor	Das Elter-Element eines Elements
	preceeding	Vorgängerknoten
	following	Nachfolgerknoten
	psibling	Kinder mit gleichem Elter-Element vor einem Element
	fsibling	Kinder mit gleichem Elter-Element nach einem Element

dard-APIs aus dem Document Object Model (s. unten) geleistet wird.

2. 6 XML-Werkzeuge

Mit XML als Standard für die Definition von Grammatiken eröffnen sich Möglichkeiten für sehr generische Werkzeuge und Bibliotheken.

XML-Parser sind inzwischen in großer Zahl vorhanden. Ihre jeweilige Mächtigkeit ergibt sich daraus, ob es sich um *validierende* oder *nichtvalidierende* Parser handelt. Erstere überprüfen die Validität eines XML-Dokuments bezüglich einer DTD, die anderen prüfen lediglich die Wohlgeformtheit des Dokuments. Einer der führenden validierenden XML-Parser ist XML4J von IBM. Das Paket ist in Java geschrieben und über die Adresse http://www.alphaworks.ibm.com erhältlich. Ein XML-Parser von Sun wird als Standard Extension in Java aufgenommen werden (s. http://www.javasoft.com/xml/ncfocus.html). Die inzwischen zahlreich vorhandenen DTD getriebenen Parserpakete finden zunehmend Ergänzung durch Parsergeneratoren für eine jeweils spezifische DTD.

Neben verschiedenen Paketen zur Darstellung beliebiger XML-Dokumente existieren inzwischen auch syntaxgesteuerte Editoren wie Xeena (ebenfalls über http://www.alphaworks.ibm.com), die entsprechend der DTD mögliche Eingabeelement in einer grafischen Oberfläche anbieten.

Die Anzahl von Werkzeuge zur Verwendung von XML-Dokumenten nimmt momentan massiv zu und umfasst alle Funktionalitäten, die auf der Basis einer kontextfreien Grammatik erstellt werden können.

2. 7 Dokumente als Objektbäume: DOM

XML-Dokumente repräsentieren aufgrund ihrer Wohlgeformtheit Bäume, deren Knoten, die laut DTD definiert Element oder Textblöcke sind. Die Hierarchie des Baums spiegelt die Schachtelung von Elementen im Dokument wider.

Ein Browser für XML-Dokumente wird das geladene Dokument nach der Parsierung in einem solchen Baum repräsentieren. Zur Darstellung des Dokuments traversiert er geeignet diesen Baum. Kennt der Browser Script-Sprachen wie Javascript oder Jscript, dann haben diese Sprachen üblicherweise Zugriff auf die interne Repräsentation und können beispielsweise dessen Struktur oder Knoten darin dynamisch verändern.

Notwendig ist also eine Objektdefinition, mit der einerseits die Baumstruktur selber und generische Operationen darauf, andererseits die spezifischen Knoten einer Auszeichnungssprache mit den jeweiligen Attributen repräsentierbar sind. Der W3C-Standard *Document Object Model* definiert eine solche Datenstruktur als Objektmodell (http://www.w3.org/DOM).

Das DOM-Level 1 besteht momentan aus zwei Schichten: *der DOM Core* definiert ein allgemeines Objektmodell zur Beschreibung von Dokumenten die einer XML-basierten Syntax folgen also Baumknoten und Elemente. *DOM HTML* konkretisiert dieses Modell für HTML-Dokumente, also beispielsweise für < P > oder < TABLE > Elemente.

Ein XML-Parser kann sich DOM-Core bedienen, um einen Baum zu erzeugen, in dem die Knoten beispielsweise Element-Objekte des Standards

sind. Er kann ein API benutzen, das die Navigation durch diesen Baum ermöglicht.

DOM ist sprachunabhängig definiert – dementsprechend wird zur Spezifikation dieses API als CORBA-IDL Schnittstellen aufgeschrieben. Ein Beispiel dafür ist die Definition der Knoten in einem Dokumentenbaum nach DOM:

```
interface Node { [...]
 readonly attribute unsigned short nodeType;
 readonly attribute Node      parentNode;
 readonly attribute NodeList childNodes;
 readonly attribute Node      firstChild;
 readonly attribute Node      lastChild;
 readonly attribute Node      previousSibling;
 readonly attribute Node      nextSibling;
 [. . .]
 Node insertBefore(in Node newChild,
                   in Node refChild)
      raises(DOMException);
 Node replaceChild(in Node newChild,
                   in Node oldChild)
      raises(DOMException);
 Node removeChild(in Node oldChild)
      raises(DOMException);
 Node appendChild(in Node newChild)
      raises(DOMException);
 boolean hasChildNodes(); [. . .]
};
```

Aufbauend auf den so allgemein definierten Knoten eines Objektbaums, kann DOM-Core spezifischer werden und die darin befindlichen Dokumentenelemente definieren:

```
interface Element : Node {
 readonly attribute  DOMString  tagName;
 DOMString getAttribute(in DOMString name);
 void   setAttribute(in DOMString name,
                     in DOMString value)
        raises(DOMException);
 void   removeAttribute (in DOMString name)
        raises(DOMException);
 Attr   getAttributeNode(in DOMString name);
 Attr   setAttributeNode(in Attr newAttr)
        raises(DOMException);
 Attr   removeAttributeNode(in Attr oldAttr)
        raises(DOMException);
 NodeList getElementsByTagName
                          (in DOMString name);
 void   normalize();
};
```

Der Kern enthält lediglich die Definitionen für die Repräsentation eines Dokuments als Baum aus mit Attributen versehenen Elementen und kann damit alle wohlgeformten XML-Dokumente repräsentieren.

Für eine spezifische Auszeichnungssprache können nun die Elemente entsprechend verfeinert werden, um ihre konkreten Tags und deren Attribute als Objekte zu repräsentieren. Für HTML geschieht dies im zweiten Teil des Standards mit DOM-HTML. Hier werden für alle HTML-Elemente entsprechende Schnittstellen definiert, die zusätzlich Methoden enthalten, mit denen das Element geeignet verändert werden kann. Für < TABLE > nach dem HTML-4-Standard ergibt sich:

```
interface HTMLTableElement : HTMLElement {
 attribute HTMLTableCaptionElement caption;
 attribute HTMLTableSectionElement tHead;
 attribute HTMLTableSectionElement tFoot;
 readonly attribute HTMLCollection rows;
 readonly attribute HTMLCollection tBodies;
 attribute DOMString               align;
 attribute DOMString               bgColor;
 attribute DOMString               border;
 attribute DOMString            cellPadding;
 attribute DOMString            cellSpacing;
 attribute DOMString               frame;
 attribute DOMString               rules;
 attribute DOMString               summary;
 attribute DOMString               width;
 HTMLElement createTHead();
 void deleteTHead();
 HTMLElement createTFoot();
 void deleteTFoot();
 HTMLElement createCaption();
 void deleteCaption();
 HTMLElement insertRow(in long index);
 void deleteRow(in long index);
};
```

DOM findet auf zweierlei Art Verwendung. Zum einen dient DOM als Repräsentation eines XML-Parserbaums zur Weiterverarbeitung. Zum anderen definieren die Methoden ein API zur Veränderung dieses Baums und ermöglichen so dynamische Web-Seiten. Eine Scriptsprache für Web-Browser kann sich auf DOM-HTML stützen und beispielsweise davon ausgehen, dass ein Tabellen-Objekt (erzeugt aus < TABLE >) eine Liste von Kinderknoten hat, die Tabellenzeilen repräsentieren (und aus < TR > Elementen erzeugt wurden). Diese Scriptsprache und ihre Benutzer können sich auf die in DOM definierte Standard-API stützen.

DOM-Level 2 wird weitere konkrete Objektmodelle einführen, so für Cascading Style Sheets Objekte oder ein standardisiertes Ereignismodell für Browser.

DOM hat sich mittlerweile als das Standard-API für die Repräsentation von XML-Dokumenten und für deren dynamische Modifikation durchgesetzt. Die meisten Werkzeuge im XML-Umfeld arbeiten mit DOM-Bäumen und die wichtigen Scriptsprachen in Browsern betten es ein.

3 Das XML-Babylon: Datenbanken, Formelsatz etc.

XML-DTDs sind anwendungsspezifisch definierbar. Sie lassen sich einfach aufschreiben und über das Netz bereitstellen. Dadurch kann jeder, der Informationen anbietet zusätzlich beschreiben in welcher Syntax er oder sie das tut. Die Semantik der ausgezeichneten Daten ist natürlich nach wie vor völlig offen gelassen. XML kann nicht eine Form der Darstellung nahe legen oder gar eine inhaltliche Semantik bereitstellen. Deren Festlegung muss außerhalb der Syntaxdefinition stattfinden, also beispielsweise in einem Textdokument.

Ein Standard kann ein solches Dokument sein und in der Tat hat das W3C seine HTML Weiterentwicklung aufgeteilt in eine Reihe verschiedener Standards für Auszeichnungssprachen die bestimmte Anwendungsdomänen abdecken sollen. Die W3C-Empfehlungen – die Standarddokumente in diesem Umfeld – bestehen in der Regel aus einer XML-DTD und der Festlegung deren Interpretation oder Darstellung.

3. 1 Weiterentwicklung von HTML

Die Weiterentwicklung der Sprache des Web, HTML war der eigentliche Auslöser für die Arbeit an XML. Spätestens mit HTML 4.0 wurde klar, dass es nicht möglich sein würde, eine einzige Auszeichnungssprache zu definieren die sämtliche Anwendungszusammenhänge in der richtigen Granularität mit Tags abdeckt. Die Syntax von HTML wurde in seinen unterschiedlichen Versionen jeweils durch eine SGML-DTD definiert. *XHTML* (http://www.w3.org/TR/xhtml1) ist eine Neuformulierung von HTML mit Hilfe einer XML-DTD. Als Working Draft steht XHTML noch vor der eigentlichen Standardisierung und die Software-Unterstützung ist momentan praktisch nicht vorhanden. Der Standard wird aber den Übergang von HTML zu XML markieren.

Technisch definiert XHTML einen XML-Namensraum, in dem die bekannten Tags und Attribute von HTML 4 definiert sind. Deren Verwendung wird im xmlns-Attribut des < html > Tags vermerkt: < html xmlns = “http://www.w3.org/TR/xhtml1“ > .

Für den HTML-Nutzer werden sich hauptsächlich syntaktische Änderungen ergeben – beispielsweise müssen nun alle Tag- und Attributnamen kleingeschrieben werden – während funktionale Ergänzungen des Sprachumfangs nicht vorgesehen sind.

3. 2 Formelsatz

Mit der *Mathematical Markup Language (MathML)* wird ein Vorhaben aus dem HTML-3-Entwurf verwirklicht: Eine Auszeichnungssprache für mathematische Formeln. Sie ist als W3C-Recommendation (http://www.w3.org/TR/REC-MathML) standardisiert.

Das folgende Beispiel zeigt eine einfache Auszeichnung einer Formel:

$$\sum_{i=0}^{100} x^i$$

```
<math>
 <apply>
  < sum/ >
  < bvar > < ci > i < /ci ></bvar>
  < lowlimit > < cn > 0 < /cn ></lowlimit>
  < uplimit > < cn > 100 < /cn ></uplimit>
  <apply>
   < power/ >
   < ci > x</ci>
   < ci > i</ci>
  </apply>
 </apply>
</math>
```

In der Formel werden zwei Elemente angewendet – das Summenzeichen mit *i* als Zähler zwischen der oberen und der unteren Grenze und die Potenzierung von *x* mit *i*.

Obwohl eine sehr ähnliche Tag-Menge schon für HTML 3 vorgesehen war, wurde sie in den damaligen Browsern nie wirklich implementiert. Bei MathML scheint dies anders zu sein: Es gibt Erweiterungen für vorhandene Browser – so das frei er-

hältliche Techexplorer Plugin von IBM (http://www.software.ibm.com/network/techexplorer) mit dem Netscape entsprechend erweitert wird –, Browser wie Amaya (http://www.w3.org/Amaya), die MathML direkt implementieren und erste Autoren-Werkzeuge, die MathML beherrschen – ein Beispiel ist die Nachfolgeversion des in Microsoft Office enthaltenen Formeleditors MathType (http://www.mathtype.com).

3. 3 Multimediapräsentationen

HTML war zunächst eine Auszeichnungssprache für textorientierte Daten. Erst durch das < IMG > Tag und Browser mit grafischer Nutzeroberfläche kamen multimediale Elemente hinzu. Ausgehend von Bildern entstanden komplexere Techniken über animierte Formatvarianten von GIF und JPEG, Imagemaps, oder eingebettete Audio- und Videoströme. Die Lösungen waren teilweise nicht standardisiert (wie < BGSOUND > im Internet Explorer), oder in den führenden Browsern unterschiedlich oder nur halbherzig implementiert (so die < EMBED > und < OBJECT > Tags). Darüber hinaus fehlten Mittel zur Koordination dieser multimedialen Komponenten – sie erlaubten keine Festlegungen wie „zunächst Video trailer.mpg abspielen und danach gleichzeitig Video film.mpg und Audio sound.wav starten".

Auf XML-Basis wird dem durch die Auszeichnungssprache für Multimediapräsentation *SMIL („synchronized multimedia integration language")* abgeholfen. SMIL (sprich „smile") ist eine W3C-Recommendation und unter http://www.w3.org/TR/REC-smil nachzulesen.

Die in SMIL-definierten Elemente erlauben die Markierung des Ablaufs einer Präsentation. Die wichtigsten Elemente legen paralleles und sequentielles Abspielen von Strömen zusammen mit in Attributen festgelegten Zeitbedingungen fest.

Dazu ein Beispiel:

```
<seq>
 < img src = "title.gif" region = "screen"
   dur = "4s" / >
 <par>
  < video src = "film.avi" region = "screen"
    dur = "6s" / >
  < audio src = "sound.wav" dur = "6s" / >
 </par>
</seq>
```

In der Präsentation finden zwei mit < seq > . . . </seq > umfasste Darstellungen nacheinander statt. Zunächst wird auf einer vorher definierten Präsentationsfläche (screen) ein Bild 4 s lang dargestellt. Danach kommen zwei von < par > umschlossene Objekte parallel zur Darstellung, nämlich ein Videoclip und ein Audiostück. Beide werden für 6 s abgespielt, danach ist die Präsentation beendet.

Erste Unterstützung findet SMIL beispielsweise in dem RealPlayer G2 für Windows, der Player für die bislang populärsten proprietären Audio- und Videostromformate im Netz.

3. 4 Grafik im Netz

In HTML sind direkt lediglich Pixelgrafiken vorgesehen, wobei verschiedene Format wie GIF, JPG oder PNG eingesetzt werden. Für höherqualitative Zeichnungen mit Vektorgrafiken existieren zwar proprietäre Formate, die sich aber nicht durchgesetzt haben und jeweils nur von entsprechenden Plugins als Browser-Erweiterungen dargestellt werden können.

Auf XML-Basis erarbeitet das W3C die Spezifikation *Scalable Vector Graphics (SVG)* (http://www.w3.org/Graphics/SVG), um standardisierte Grafiken jenseits reiner Pixeldaten zu ermöglichen. SVG definiert eine Reihe von Elementen zur Auszeichnung von Koordinatensystemen, für einen Satz grundlegender Zeichnungselemente und deren Attribute, für die Einbindung von Pixelgrafiken in Zeichnungen und für die Verwendung von Text. Ein Beispiel aus dem aktuellen Spezifikationsentwurf zeigt eine typische SVG-Markierung:

```
< svg width = "3in" height = "3in" >
  < desc > A blue circle with a red outline
  < /desc >
  < g > < circle cx = "200" cy = "200" r = "100"
    style = "fill: blue; stroke: red" / > < /g >
</svg>
```

Die SVG-Grafik nimmt hier eine quadratische Fläche ein, und wird textuell innerhalb von < desc > beschrieben. Die eigentliche Zeichnung innerhalb von < g > besteht aus einem Kreis bei der Koordinate 200, 200 und einem Radius von 100 Einheiten. Sein Zeichenstil beschreibt das style-Attribut als blau gefüllt mit rotem Rand.

SVG ist noch nicht standardisiert und es existieren nur erste experimentelle Darstellungspro-

gramme, die über die obige URL erreichbar sind. Es ist aber abzusehen, dass ein solches Zeichnungsformat erfolgreich sein muss, da der Bedarf nach höherwertigen Darstellungen – beispielsweise für technische Dokumentationen – enorm ist.

3. 5 Datenbanken

XML ist eine Auszeichnungssprache für Texte. Während bei Elementen wie < FRAGE > . . .? < /FRAGE > und < ANTWORT > . . .! < /ANTWORT > das Gewicht tatsächlich auf der Markierung von Text liegt, wird sehr oft eher eine datenorientierte Markierungsabsicht deutlich, wie bei < PLZ > .

XML spielt im Bereich Datenbanken an drei Stellen eine wichtige Rolle:

– Aus herkömmlichen Datenbanken sollen Web-Seiten generiert werden, die eine gewisse semantische Auszeichnung behalten – bei einem RDBMS könnten die Elementnamen beispielsweise den Typnamen der Entitäten entsprechen. In der Kombination mit Style-Sheets kann auch die Darstellung besser gesteuert werden als bei reiner HTML-Erzeugung.

  Technisch notwendig sind hierfür XML-Exportformate für Datenbanken, die wie die heute üblichen HTML-Aufsätze für die gängigen Datenbanken verfügbar sein werden.
– XML soll die für den Nutzer sichtbare Datenrepräsentation darstellen. Dementsprechend wird er oder sie Datenbankanfragen in einer auf XML-basierenden Anfragesprache stellen, die eine Rolle ähnlich SQL für relationale DBMS haben würde.

  Momentan ist ein Standard für eine solche Sprache in Erarbeitung durch das W3C. Es gibt derzeit zwei konkurrierende Entwürfe: XQL und XML-QL.

  XML-QL: A Query Language for XML (http://www.w3.org/TR/NOTE-xml-ql) definiert eine Syntax und eine Semantik für die Auswahl von Daten aus XML-Dokumenten und die Erzeugung neuer XML-Dokumente daraus. Das folgende Beispiel soll dies klar machen:

```
WHERE
  < ANSCHRIFT >
      < NAME > $n < /NAME >
      < ORT NAME = „Berlin"/ >
  < /ANSCHRIFT >
IN „www.info/liste.xml"
CONSTRUCT
  < BERLINER > $n < /BERLINER >
```

  Für das Beispiel wird eine Anfrage bezüglich des Dokuments www.info/liste.xml formuliert. Dabei soll die Anfrage in der WHERE-Klausel auf alle Dokumententeile passen, die durch ein Tag < ANSCHRIFT > markiert sind, und das ein < NAME > Tag und ein < ORT > Tag umschließt. Bei < ORT > soll das Attribut NAME den Wert Berlin haben. Der von < NAME > umschlossene Dokumententeil soll an $n gebunden werden. Für jedes so gefundene $n soll ein neues XML-Fragment erzeugt werden, nämlich der gefundene Name markiert mit dem Tag < BERLINER > .

  XQL (http://www.w3.org/TandS/QL/QL98/pp/xql.html) dagegen nimmt einen etwas anderen Ausgangspunkt, der weiter unten beschrieben ist. XSL ist eine Sprache zur Transformation eines XML-Dokuments in ein anderes. Dazu werden Muster definiert, die auf einzelne Knoten des Quelldokuments passen und Regeln, die die Transformation dieses Knotens und die weitere Verarbeitung beschreiben.

  XQL nimmt die zur Beschreibung der Muster in XSL verwendete Sprache als Ausgangspunkt und erweitert sie zu einer Abfragesprache. Die Verarbeitung der Abfrage entspricht dann dem Auffinden eines auf das Muster passenden Teildokuments. Eine der ersten Implementierungen von XQL findet sich bei http://xml.darmstadt.gmd.de/xql.

  Es ist offen, welcher der beiden Entwürfe sich durchsetzen wird, dabei ist zu beachten, dass XQL sehr starke Unterstützung durch Microsoft hat.
– Schließlich kann XML auch als internes Datenformat für ein DBMS dienen. Hersteller wie Poet, Oracle oder Object Store haben direkte XML-Unterstützung angekündigt; dabei wird hauptsächlich mit dem Wegfall der Konvertierung von Daten von und nach XML geworben.

Zugleich werden auch Standardisierungsbemühungen unternommen, XML durch entsprechende Schemadefinitionen zu einer vollwertigen Datenrepräsentationssprache zu erweitern, beispielsweise mit ersten Dokumenten wie *XML-Schema* (http://www.w3.org/TR/xmlschema-1 und http://www.w3.org/TR/xmlschema-2) oder der *Schema for Object-oriented XML* (http://www.w3.org/TR/NOTE-SOX).

Die Unterstützung von XML durch die großen Datenbankhersteller hebt die Bedeutung von XML für diesen Bereich hervor, zumindest was den Bereich der Web-basierten Darstellung von Daten angeht. In Sachen Anfragesprachen kann man allerdings noch nicht absehen, welche Lösung Marktdominanz erreichen wird.

3. 6 Geschäftlicher Datenaustausch

Noch erheblich vielfältiger sind die Ansätze für die XML-Verwendung im Bereich Electronic Commerce. Klar scheint, dass XML die Basis für verschiedenste Anwendungsplattformen im Bereich Datenaustausch und elektronische Bezahlung werden wird.

Welche technische Lösung sich letztlich durchsetzt ist aber noch nicht absehbar, da momentan noch keine Ansätze für herstellerübergreifende Standards vorhanden sind. Dementsprechend konkurrieren eine Reihe von Herstellern mit unterschiedlichen Lösungen um eine Dominanz bei der Erstellung eines Quasi-Standards, um darauf eine Markdominanz für Infrastrukturen aufzubauen (Tabelle 2).

7 Metadaten

Metadaten sind einer der entscheidenden Schlüssel zur einfacheren Navigation im Web. Während Suchmaschinen den Inhalt einer Seite durch Volltextindizes textuell erfassen, existieren momentan kaum Suchmöglichkeiten auf der Basis von Metadaten um beispielsweise Dokumente eines bestimmten Autors zu finden.

In HTML 4 waren durch die Tags < LINK > und < META > erste Möglichkeiten dafür vorgesehen, etwa um mit < META NAME = "author" CONTENT = "Robert Tolksdorf" > den Autor eines Dokuments bekannt zu machen. Die Tags haben sich aber weder bei den Informationsanbietern noch bei Nutzern von Suchmaschinen durchgesetzt. So bietet die Suchmaschine Fireball (http://www.fireball.de/detail.html) zwar die Suche auf < META > Tags, allerdings wird diese Möglichkeit kaum genutzt.

Tatsächlich beschränkt sich das < META > Tag lediglich darauf, zusätzliche Informationen zu einer Seite beispielsweise als Schlagwörter zu vergeben. Die Bildung von Relationen zwischen „Objekten“ im Netz ist nicht möglich. Mit dem < LINK > Tag war dies vorgesehen, es existiert aber kaum ein Browser, der es ausnutzt.

Das *Resource Description Framework* (RDF) (s. http://www.w3.org/RDF) ist eine Standardisierungsaktivität des W3C mit dem Ressourcen mit anderen in Relation gesetzt werden können.

RDF ist ein Rahmenwerk zur Notation von *Eigenschaften* von *Ressourcen*. Eine Web-Seite ist dabei eine Ressource, für die als Beispiel der Autor vermerkt werden soll. Sie kann also als Eigenschaft eine Beziehung zu ihrem „Autorenobjekt“ haben, was im RDF-Rahmen wie folgt notiert werden könnte:

```
< RDF >
  < Description about = "http://www. cs.
  tu-berlin.de/ ~ tolk/index. html" >
    < Creator > Robert Tolksdorf < /Creator >
  < /Description >
< /RDF >
```

Als Rahmenwerk legt RDF aber nicht etwa fest, dass jede Seite eine Eigenschaft Creator tragen kann, sondern definiert lediglich die dafür notwendigen Datentypen.

Ressourcen sind die Entitäten, deren Eigenschaften beschrieben werden. Das Beispiel beschreibt eine einzelne Web-Seite – es könnten aber auch Fragmente einer Seite oder eine andere durch eine URL beschriebene Entität sein. Eine Ressource

Aktuelle XML-basierte Entwürfe im Bereich Electronic Commerce

Tabelle 2

Entwicklung	Informationen bei
CBL eCO Systems	www.commerce.net www.veosystems.com
Open Trading Protocol	www.otp.org
XML/EDI	www.xmledi.com
Rosettanet	www.rosettanet.org
OBI Open Buying Initiative	www.openbuy.org
ICE	www.w3.org/TR/1998/NOTE-ice-19981026
Open Financial Exchange	www.ofx.net

kann *Eigenschaften* haben, im Beispiel ist dies Creator. Und schließlich kennt RFD Aussagen *(Statements)* die besagen, dass bei einer bestimmten Ressource eine bestimmte Eigenschaft einen bestimmten Wert trägt. Im Beispiel ist der Wert für die Creator Eigenschaft der Name des Autors.

Über diese Basistypen hinaus definiert RDF zusätzlich Behälter *(Container)*, mit denen Ressourcen mehrfache Werte für eine Eigenschaft tragen können. So gibt es geordnete *(Sequence)* und ungeordnete *(Bag)* Listen sowie Alternativen.

Das um Aussagen höherer Ordnung ergänzte Datenmodell von RDF ähnelt stark einem Entity-Relationship-Modell. Was ihm fehlt sind Möglichkeiten zur Festlegung und Typisierung der Eigenschaften von Ressourcen. Für die Festlegung eines solchen Schemas erarbeitet W3C einen entsprechenden Standard mit dem *Resource Description Framework Schema Specification* (http://www.w3.org/TR/1998/WD-rdf-schema).

Mit diesen Hilfsmitteln können nun standardisierte Schemata für Metadaten festgelegt und auf der Basis von RDF spezifiziert und verarbeitet werden. Während RDF lediglich einen Rahmen für die Notation von Metadaten bildet, legen diese Schemata die tatsächlich beim Nutzer sichtbaren Informationen fest.

Vom W3C sind momentan zwei solcher Schemadefinition in Arbeit. Die 1996/97 definierte *Platform for Internet Content Selection* PICS soll sich in einer zweiten Version auf RDF stützen. PICS (http://www.w3.org/PICS) erlaubt die Notation von Metadaten zur Art des Seiteninhalts. Sie dienen zur Ausfilterung unerwünschten Inhalts (Beispiel: Gewalttätiger Inhalt) für Nutzergruppen (Beispiel: Kinder). Als zweite RDF basierte Aktivität soll die *Platform for Privacy Preferences* P3P (http://www.w3.org/P3P) entwickelt werden. Hier dient RDF als Mittel zur Notation persönlicher Ansprüche an die Vertraulichkeit im Umgang mit übermittelten Daten bzw. der Zusage von Vertraulichkeit durch einen Anbieter.

RDF ist ein sehr mächtiger Standard, der seine Wirkung aber nur durch die Akzeptanz von standardisierten Metadaten-Schemata entfalten können wird. Die gesellschaftliche Bedeutung von PICS und P3P betont allerdings das Gewicht der RDF-Technologie.

4 XML-Dokumente mit XSL verarbeiten und darstellen

Wohlgeformte XML-Dokumente stellen durch die Schachtelung von Tags Bäume dar. Die Darstellung eines solchen Dokuments in einem Browser ist in der Regel die Transformation dieses Baums in einen anderen „Baum“: Den durch visuell geschachtelte Darstellungsflächen repräsentierten Baum.

Die *Extensible Stylesheets Language* XSL stellt einen allgemeinen Mechanismus für die Transformation von XML-Bäumen bereit und wird momentan als Standard entwickelt (http://www.w3.org/Style/XSL).

XSL hat zwei Bestandteile: Eine allgemeine Sprache zur Transformation von Bäumen XSLT und eine Sprache zur Notation von in Flächen dargestellten Bäumen, die aus *Formatted Objects* – kurz FO – gebildet sind.

XSL-Regeln haben einen deklarativen Aspekt. Beginnend beim Wurzelknoten werden alle Kinderknoten im Hinblick auf ein Muster untersucht. Passt das Muster, wird eine angegebene Schablone darauf angewandt.

Sei als Beispiel eine Liste von Produkten wie folgt mit XML ausgezeichnet:

```
<liste>
 <hersteller>ABC</hersteller>
 <produkt>
  Lenkrad
  <verkauf>
   <inland>100</inland>
   <ausland>200</ausland>
  </verkauf>
 </produkt>
 <produkt>
...
</liste>
```

Eine XSL-Transformation soll daraus zwei Tabellen erstellen, wobei die eine die Inlands-, die andere die Auslandverkäufe enthalten soll. Die folgende XSL-Regel leistet dies:

```
<xsl:template match="product">
  <tabelle>
    <xsl:apply-templates
     select="verkauf/inland"/>
  </tabelle>
  <tabelle>
    <xsl:apply-templates
     select="verkauf/ausland"/>
</xsl:template>
```

Alle < produkt > -Kinder von < liste > sind also von Interesse. Sie werden mit einem Muster ausgewählt. Das angegebene Template erzeugt zwei neue Unterbäume, die jeweils mit < tabelle > ausgezeichnet sind. In diese Tabellen fügt der XSL-Prozessor die Kinder der Knoten ein, die mit < inland > und < ausland > jeweils innerhalb eines < verkauf > Tags markiert sind.

XSLT beschreibt weitere und umfangreiche Konstrukte für Muster und Templates, mit denen Bäume transformiert werden können. Ein operationaler Aspekt von XSL-Regeln ergibt sich aus Markierungen, die die Verarbeitung des Baums steuern. So traversiert < xsl:apply-templates/ > rekursiv den Baum, indem die Muster auf alle Kinderknoten eines passenden Knotens erneut angewendet werden. Hinzu treten beispielsweise Anweisungen für eine bedingte Transformation. An den Beispielen wird eine weitere Eigenschaft von XSL-Regeln deutlich: Sie sind selber als XML-Dokumente notiert und verwenden den Namensraum xsl.

Nachdem zwischen XML-Bäumen transformiert wird, ist der Zielbaum genauso wenig mit einer Semantik versehen wie der Quellbaum. Im Beispiel bleibt offen, was < tabelle > bedeutet oder wie es dargestellt werden soll. Der zweite Bestandteil von XSL, die *Formatted Objects*, hat eine Semantik in Form einer definierten Darstellung in einem Browser.

Durch Formatted Objects kann eine XSL-Regel beschreiben wie ein XML-Dokument dargestellt werden soll. Dazu transformiert sie den Quellbaum aus einem anwendungsspezifischen XML-Namensraum in den standardisierten Namensraum der Formatted Objects (Präfix fo). Er enthält einen Satz Baumknoten mit Attributen, die der Darstellungsmächtigkeit heutiger Web-Browser entsprechen.

Um dem obigen < tabelle > eine Semantik in der Darstellung zu geben, kann folgende XSL-Regel benutzt werden:

```
< xsl:template match = "tabelle" >
  < fo:table >
    < xsl:apply-templates/ >
  < fo:table-caption > Tabelle
  < /fo:table-caption >
  < /fo:table >
< /xsl:template >
```

Befindet sich in der aktuellen Knotenliste ein < tabelle > Element, so wird es in das Formatted Object < fo:table > transformiert, sein textueller Inhalt dort hineinkopiert und eine Tabellenunterschrift mit dem Formatted Object < fo:table-caption > erzeugt. Formatted Objects enthält eine umfangreiche Liste von Objekten, deren Darstellung jeweils textuell definiert wird.

XSL ist ein sehr komplexer Ansatz, gleichzeitig aber sehr mächtig. Es ist sehr fraglich, ob der Endnutzer eigene Darstellungsregeln formulieren wird. Auf der Seite der Informationsanbieter wird XSL aber eine größere Rolle spielen, da die Darstellungssemantik eigener XML-Grammatiken so festgelegt werden kann. Die Bedeutung in der Verarbeitung von XML-Daten durch Baumtransformationen wird momentan noch unterschätzt.

XSL-Prozessoren sind in ersten Versionen verfügbar, beispielsweise Koala (http://www.inria.fr/koala/XML/xslProcessor), XT (http://www.jclark.com/xml/xt.html) oder LotusXSL (http://www.alphaworks.ibm.com/tech/LotusXSL). Der InDelv XML Browser (http://www.indelv.com) ist einer der Ersten mit vollständiger XSL-Unterstützung einschließlich implementierter FO. Der Internet Explorer 5 von Microsoft implementiert XSL sehr eingeschränkt. Er kann zwar XSL-Regeln einlesen und XML transformieren, produziert als Ergebnis aber lediglich HTML. Somit beherrscht er weder beliebige XML-Namensräume noch Formatted Objects.

5 Ausblick

Die verschiedenen X-Standards im Web-Umfeld sind momentan erst in Entwicklung. Es existiert erste Software zu ihrer Unterstützung – XML wird als erste Technologie einen stabilen Status und große Verbreitung erreichen. Die angekündigte Unterstützung von Browser-Herstellern und die vielen Versuche, Standards aus Anwendungsdomänen in XML-Form zu bringen lassen die Vorhersage zu, dass XML mit Sicherheit das zukünftig wichtigste Format für Daten und Dokumente wird.

Tatsächlich überwiegt die schiere Tatsache der Akzeptanz einer solchen Standardfamilie das Bild, das XML im Hinblick auf Qualität und Innovation der Technologie abgibt.

XML bietet praktisch keinerlei technologische oder konzeptuelle Neuigkeiten. XML ist lediglich eine Sprache zur Definition einer kontextfreien

Grammatik, die selber SGML konform ist. Die Verwendung von XML-basierten Sprachen zum Austausch von Informationen zeichnet sich eigentlich nur dadurch aus, dass sie nicht binär repräsentiert sind und die Grammatik des Austauschformats als DTD mitgesandt werden kann.

Die Entwicklung XML-basierter Standards findet zwar im Rahmen des W3C statt, damit ist aber noch keineswegs eine zielgerichtete und sachorientierte Standardisierung gewährleistet. Wie in diesem Beitrag deutlich wurde, führt die Offenheit von XML zu einem Wildwuchs von Vorschlägen für verschiedenste Bereiche. Die Bedeutung eines Vorschlags misst sich zumeist nicht an seiner Qualität, sondern an dem Ausmaß der Unterstützung durch wichtige Hersteller. Hier versuchen die großen Firmen mittlerweile an möglichst allen Entwicklungen teilzuhaben und an die Stelle sachorienterter Spezifikationen treten Marktinteressen und Firmenpolitik.

Schließlich entsteht auch ein interessanter zeitlicher Aspekt. Während der Anstoß zur Entwicklung einer XML-basierten Technologie recht leicht gegeben werden kann und somit eine vermeintlich hohe Dynamik in diesem Bereich herrscht – die sicherlich auch einige der hier beschriebenen Entwicklungen betreffen wird –, bestimmt die reale Unterstützung eines Standards durch verbreitete Software das wahre Tempo. Die Geschwindigkeit der Umsetzung eines Standards ist aber das Maß für den Endnutzer der von neuen Technologien profitieren will.

Die enorme Bedeutung von XML ergibt sich somit nur aus seiner allgemeinen Akzeptanz als Standard. Diese Entwicklung ist vergleichbar mit der von HTML als einer Hypertext-Sprache mit sehr einfacher Mächtigkeit, deren Bedeutung sich aus ihrer globalen Verwendung hergestellt hat.

Literatur

1. Goldfarb, C. F., Rubinsky Y.: The SGML handbook. 1990
2. Tolksdorf, R.: Die Sprachen des Web: HTML4 und XHTML. Heidelberg: dpunkt.verlag 1999. http://www.dpunkt.de/html/