XML-Schemasprachen und W3Cs XML-Schema

Mario Jeckle

DaimlerChrysler Forschungszentrum Ulm
Wilhelm-Runge-Str. 11
89013 Ulm
mario.jeckle@daimlerchrysler.com

Abstract: Die generische Metasprache Extensible Markup Language (XML) hat seit ihrer Vorstellung 1998 beachtenswertes Interesse gefunden. Standen zur Formulierung von XML-Vokabularen zunächst ausschließlich die vom Vorgängerstandard SGML übernommenen Dokumenttypdefinitionen (DTD) zur Verfügung, so konnte sich mittlerweile eine eigene XML-Sprache zur Darstellung beliebiger XML-Sprachschemata etablieren.
Die Notwendigkeit einer solchen Sprache wird durch die Beschränkungen des DTD-Mechanismus offenkundig, der sich zunehmend als inadäquat offenbarte. Eine Ursache liegt im veränderten Einsatzkontext der Sprache. Während die Grundkonzeption der SGML-DTD primär auf die Abbildung präsentationsorientierter Information, wie in der technischen Dokumentation verwendet, ausgelegt war, werden XML-Dokumente überwiegend Datenorientiert eingesetzt und stellen daher andere Anforderungen hinsichtlich der angebotenen Strukturierungs- und Inhaltsprimitive.

1 Einleitung

Aktuell entsteht durch die Definition neuer XML-basierter Formate die XML-Sprachfamilie. Nahezu ausnahmslos verwenden diese XML-Vokabulare die durch das World Wide Web Konsortium (W3C) verabschiedete Sprache *XML-Schema* zur Definition der Sprachstruktur (Syntax) und des erlaubten Dokumentinhalts (Typisierung).
XML-Schema [Be01, BM01] schickt sich damit an, den bisher dieses Anwendungsfeld dominierenden Document Type Definition (DTD)-Mechanismus abzulösen, der aufgrund seiner Herkunft aus dem Bereich des präsentationsorientierten Publizierens denkbar schlecht für die heute verstärkt anzutreffende datenorientierten Betrachtungsweise geeignet ist.
Der Begriff *Schema* ist dabei der im Datenbankumfeld gebräuchlichen Terminologie entlehnt. Dort bezeichnet er Informations- oder Datenmodelle, die als Konstruktionsvorlage, zur Dokumentation oder zum Datenbankdesign herangezogen werden. Hierzu muß ein Schema nicht zwingend in einer graphischen Datenmodellierungssprache vorliegen, sondern kann beispielsweise auch die Tabellenstruktur einer relationalen Datenbank beschreiben.

2 Zur Notwendigkeit einer Schemasprache

Zum Zeitpunkt der Konzeption der Metasprache *Standard Generalized Markup Language* (SGML) [In86] Mitte der 1970er Jahre war deren Anwendungsfeld klar umrissen und im wesentlichen auf die Digitalisierung vormals papiergestützter Dokumentation festgelegt. Daraus erklärt sich die Mächtigkeit der XML-DTD [Br00, chap. 3.2ff], der angebotenen Grammatiksprache zur Darstellung von Dokumentstrukturen. Insbesondere war weder die datenorientierte Verwendung von SGML, noch die rund 30 Jahre später einsetzende Weiterentwicklung (eigentlich: Reduktion) zur Extensible Markup Language abzusehen.
Die inzwischen erfolgende breite Anwendung von XML-Vokabularen zur Darstellung beliebiger Inhalte läßt jedoch die Beschränkungen und Unzulänglichkeiten des DTD-Mechanismus für dieses Einsatzgebiet offenkundig werden.
Vom syntaktischen Standpunkt aus betrachtet gestatten Dokument Typ Definitionen die Festlegung der grundlegenden XML-Sprachprimititive *Element* (Werte, die durch benannte, von spitzen Winkelklammern umschlossene, Marken eingegrenzt sind) und *Attribute* (Namen-Wert-Paare).

Nachfolgend seien einige der durch Nutzung des DTD-Mechanismus zur Beschreibung datenintensiver Strukturen induzierten Einschränkungen zusammengestellt:

- Unzureichende Datentypunterstützung.
  DTDs erlauben für Elemente nur vier Inhaltsmodelle: explizit angegebene Kindelemente (*child elements*), unstrukturierten textartigen Inhalt (*parsed character data – PCDATA*), „gemischten Inhalt“ , der sowohl Texte als auch strukturierte Anteile kombiniert (*mixed content*), sowie das leere Inhaltsmodell (*EMPTY*).
  Für die Attributdefinition stehen verschiedene Datentypen zur Verfügung. Genaugenommen werden aber hier nur zeichenkettenartige Datentypen (explizit für *character data – CDATA*, implizit für identifizierende Schlüsselattribute (*ID*), deren Referenzen (*IDREF* und *IDREFS*) und Textersetzungsmuster (*ENTITY* und *ENTITIES*)), sowie Bezeichner (*NMTOKEN* und *NMTOKENS*) angeboten.[1]

- Unzureichende Strukturierungsunterstützung.
  Die Operatoren zur Steuerung der Auftrittshäufigkeit einzelner Kindelemente gestatten theoretisch die Codierung beliebiger Kardinalitäten. Allerdings sind die hierfür anzuwendenden Prinzipien teilweise umständlich in der Schreibung, und daher fehlerträchtig. Zusätzlich beeinträchtigen sie die Lesbarkeit der entstehenden DTD signifikant.

- Keine Wiederverwendungsunterstützung.
  Während Elementstrukturen innerhalb der definierenden DTD beliebig wiederverwendet werden können, sind Attribute immer an das umgebende Element gebunden.

[1] Die manchmal anzutreffende (Nach-)Modellierung benötigter Datentypen durch anwenderdefinierte Aufzählungstypen ist zeitaufwendig und fehlerträchtig und soll hier nicht als praktikable und ernstzunehmende Alternative diskutiert werden.

Eine Nutzung in einer anderen als der definierenden DTD ist nicht vorgesehen. Zwar läßt sich dies durch die Definition von einfachen Textersetzungsmustern – in SGML/XML-Terminologie: sog. *Entitäten* – annähernd nachbilden, allerdings ist dies bereits zum Erstellungszeitpunkt der Grammatik geeignet zu berücksichtigen.
Modularisierung im Sinne einer Wiederverwendung der durch die Elemente definierten Typen innerhalb der DTD, beispielsweise zur Definition weiterer Typen, wird lediglich durch *Parameter Entitäten* unterstützt. Sie verhalten sich für die DTDs wie die Entitäten in XML-Dokumenten.

- Starres Typsystem.
  Die Erweiterung des vorgegebenen Typsystems durch den Anwender ist nicht vorgesehen und findet auch durch existierende Sprachmechanismen keine adäquate Unterstützung.

- Keine Unterstützung von Namensräumen.
  Namensräume [Br99b] für die Familie der durch die Vokabulardefinition festgelegten Dokumente können in der DTD nicht angegeben werden.[2]

- Rudimentärer Referenzierungsmechanismus.
  Die durch *ID-IDREF(S)*-Paare offerierten Verknüpfungen sind ausschließlich dokumentlokal möglich und gestatten keine Differenzierung hinsichtlich der Semantik des eindeutig identifizierten oder referenzierten Elements.

- DTD-Syntax ist nicht XML.
  Die von SGML übernommene Syntax der DTD bildet eine eigene Sprache, die durch Entwickler gesondert zu erlernen und durch Werkzeugen gesondert zu implementieren ist.

3 Ansätze

Prinzipiell lassen sich die in der Vergangenheit vorgeschlagenen Ansätze zur Definition einer XML-Schemasprache in vier Kategorien unterscheiden:

1. Orientierung am bestehenden DTD-Mechanismus.
   Erweiterungen des bestehenden Mechanismus um zusätzliche Sprachelemente.

2. Orientierung an der programmiersprachlichen Interpretation.
   Versuch, XML und ein Ausführungsmodell möglichst eng zu koppeln.

3. Orientierung an Wissensdarstellungen.
   Interpretation des Schemas einer XML-Sprache als Wissen über die Sprache.

[2] Die hierfür oft angeführten Hilfskonstrukte spiegeln weder die volle Mächtigkeit, noch die korrekte Semantik der Namensräume wieder.

4. XML-Sprachen zur Inhaltsbeschreibung.
   Da XML i.A. zur Beschreibung beliebiger Information herangezogen werden kann, ist die Verwendung auch für die Beschreibung von XML-Strukturen denkbar.

Die naheliegendste Option dürfte die Erweiterung des bestehenden DTD-Sprachumfanges sein. Durch geeignete Modifikationen und Ergänzungen ließen sich alle, mit Ausnahme der letzten, identifizierten Unzulänglichkeiten beheben.
Konzeptionell lassen sich innerhalb dieses Ansatzes zwei Erweiterungsvarianten aufzeigen. Zunächst die Möglichkeit, die XML-DTDs um Elemente der ursprünglichen SGML-DTD zu erweitern (dieser findet sich beispielsweise in den *Datatypes for DTDs* [Bu00] verwirklicht). In der Konsequenz nähert sich XML, positiv formuliert, wieder der Ausdrucksmächtigkeit der Ursprache SGML an. Negativ formuliert, kann jedoch XML auf diesem Wege niemals Inhaltsstrukturen ausdrücken, die nicht durch SGML ausdrückbar sind, da die Mächtigkeit des SGML-DTD-Mechanismus eine Grenze der Erweiterbarkeit darstellt.
Zusätzlich ist anzumerken, daß ein solcher Ansatz der ursprünglichen Intention der XML-Entwicklung – ein leichter einsetzbares SGML zu schaffen – entgegenläuft.

Alternativ zur Erweiterung hin zur SGML-Mächtigkeit ließe sich der bestehende XML-DTD-Mechanismus um neue zusätzliche Konstrukte anreichern, die nicht Bestandteil der SGML-DTD-Syntax sind. Dieser Ansatz zeichnet sich durch den Vorteil aus, den Vorgängerstandard unberücksichtigt lassen zu können und beliebige Erweiterungen in Syntax und Semantik einbringen zu können. Allerdings würde damit auf eine der zentralen Forderung der XML-Entwicklung, die sich bereits im Abstract der XML-Recommendation findet, kein Bezug genommen: die Untermengenbeziehung zu SGML. Durch eine Erweiterung, welche über die SGML-Mächtigkeit hinausreicht, entstehen korrekte XML-Dokumente, welche nicht mehr als gültige SGML-Dokumentinstanzen anzusehen sind.

Die im zweiten Punkt angedeutete Umsetzung ist durch eine programmiersprachliche Verarbeitung der XML-Dokumente motiviert. Aus Sicht dieser Anwendungsfacette ist ein Schemamechanismus idealerweise so ausgelegt, daß er die transparente Umsetzung in Applikationsdatenstrukturen ermöglicht. Dahinter steht der Wunsch, den *impedance mismatch*, mithin den zu leistenden Abbildungsaufwand zwischen XML-Konstrukten und Datenstrukturen, möglichst gering zu halten.
Beispielsweise greift der Vorschlag *Schema for Object-oriented XML* [Da99] (SOX) zur Definition der notwendigen Semantik der angebotenen Schemaprimitiven auf die Programmiersprache Java zurück.

Der dritte technische Ansatz weist auf eine alternative Interpretation der XML-Grammatikstruktur hin. So spiegelt ein Schema auch immer Wissen über Struktur und Inhalt eines betrachteten Problembereichs wieder.
Der bekannteste Vorschlag – die *Document Content Description* [Bra99a] (DCD) – nutzt zur Definition der Wissensstrukturen eines XML-Dokuments das *Resource Description Framework* (RDF) [LS99] des World Wide Web Consortiums.
Der Ansatz hat sich durch Referenzimplementierungen als tragfähig und, wegen der RDF-Basiertheit, als allgemein verwendbar erwiesen. Jedoch liegt hierin auch

die offensichtlichste Limitierung. RDF als Metasprache der Schemasprache legt bereits eine gewisse Strukturierung aller Schemata zugrunde[3], da jedes gültige DCD-Schema definitionsgemäß ein RDF-Dokument darstellt. Ebenso ist die Semantik der eingesetzten RDF-Elemente bereits durch diese Spezifikation vorgegeben. Beide Punkte zusammengenommen offenbaren eine ausgeprägte Abhängigkeit von den weiteren RDF-Aktivitäten des World Wide Web Consortiums, die bisher nicht auf die Wechselwirkung zwischen Schemasprache und Wissensbeschreibungsformat ausgerichtet sind.
Positiv fällt an DCD die Verwendung von XML zur Beschreibung von XML-Sprachen auf, womit auch die letzte der erhobenen Anforderungen zu erfüllen wäre.
Die Verknüpfung von RDF mit DCD als Schemasprache birgt allerdings ein potentielles Problem hinsichtlich der Validierbarkeit der entstehenden Strukturen. Durch den Rückgriff von DCD auf RDF entsteht bei der Angabe eines Schemas für RDF ein transitiver Zirkelschluß. In der Konsequenz wird zur Validierung eines XML-Dokuments, welches einer mittels DCD-formulierten Grammatik folgt, neben dem eigentlichen DCD-Schema des Dokuments auch das DCD-Metaschema und dessen Semantik-liefernde RDF-Beschreibung benötigt.

Diese Beschränkung mildert die vierte Familie von XML-Schemasprachen ab, der zahlenmäßig die meisten der veröffentlichten Vorschläge zuzurechnen sind. Technisch ist hier die Schemasprache selbst als als XML-Sprache ausgelegt, die sowohl durch ein Vokabular zur Definition eigener XML-Vokabulare als auch durch eine Semantikbeschreibung zur inhaltlichen Ausdetaillierung der beschriebenen Konzepte besitzt.
Als Konsequenz ergibt sich für die dadurch initiierte Meta-Schemaebene die Möglichkeit der Anwendung der Schemasprache auf sich selbst und damit die rekursive Selbstbeschreibbarkeit. Das bedeutet das Schema eines beliebigen Schemas (d.h. das Meta-Schema aller denkbaren Schemainstanzen) kann durch sich selbst validiert werden. Da dieser Prüfschritt nur einmal erfolgen muß, kann er durch Schemawerkzeuge vorweggenommen und daher fest umgesetzt werden.

Insgesamt, alle konkurrierenden Schema-Sprachansätze betrachtend, hat die Klasse der kontextfreien regulären Sprachen die größte Bedeutung erlangt.
Eine Sprache dieses Typs entwickelt auch die W3C-Arbeitsgruppe zur Definition eines XML-Schemasprachstandards, der inzwischen in der ersten Version der Öffentlichkeit vorgestellt wurde [Be01, BM01]. Insbesondere berücksichtigt diese Aktivität explizit die Vorgängersprachen XML Data [La98], DCD [Bra99a], SOX [Da99]sowie die Document Definition Markup Language [Bo99]. Diese in der Vorstandardisierungsphase am Markt konkurrierenden Vorschläge unterscheiden sich semantisch lediglich in Nuancen, bieten dem Anwender jedoch teilweise optisch stark differierende Konstrukte zur Syntaxspezifikation an.[4]

[3] im wesentlichen ihre Formulierbarkeit mit den Mitteln der Prädikatenlogik erster Stufe

[4] Einen strukturell unterschiedlichen Ansatz verfolgt die durch Rick Jelliffe vorgeschlagene Sprache *Schematron* [Je00]. Sie interpretiert ein Schema als Sammlung von Regeln, denen ein gegebenes Dokument genügen muß, um als gültig akzeptiert zu werden. Dies erlaubt die Formulierung mächtiger konktextsensitiver Einschränkungen, die während des Validierungsvorganges geprüft werden. Die Umsetzung dieser Schemasprache setzt auf den XML-Standards XPath [CD99] und XSLT [Cl99] auf.

4 W3Cs XML-Schema

Jenseits aller existierenden verschiedenen Sprachvorschläge kommt dem W3C-Standard – zumeist als *XML Schema Description Language* (XSD) bezeichnet und entsprechend abgekürzt – die größte praktische Bedeutung zu.
W3Cs XML-Schema bildet zusammen mit XML v1.0 2^{nd} Edition und den XML-Namensräumen die Basis aller weiteren XML-Sprachstandards des Konsortiums.

Aus formalen Gründen ist nicht mit dem Ersatz der DTD durch Schema, in Form einer offiziellen Ablösungserklärung und Entfernung der DTD aus dem XML-Standard, zu rechnen. Dies liegt formaljuristisch in der postulierten Untermengenbeziehung zwischen XML und SGML begründet, die durch diesen Schritt aufgelöst würde. Als hieraus erwachsende praktische Konsequenz würde XML von der internationalen Standardisierung abgekoppelt und daher in vielen Bereichen, beispielsweise sicherheitskritischen Projekten, welche die Nutzung ISO-standardisierter Techniken vorschreiben, nicht mehr zur Verwendung offenstehen.
Jedoch ist mittelfristig damit zu rechnen, daß neu entwickelte XML-Vokabulare ausschließlich W3Cs XML-Schema nutzen werden.

XSD bildet eine vollständig in XML-Syntax formulierte kontextfreie reguläre Grammatik zur Formulierung beliebiger XML-Strukturen ab. Hierbei handelt es sich um die bekannten Grundprimitive *Element* und *Attribut*, andere – wie Entitäten oder Notationen – können hingegen nicht durch Schemata ausgedrückt werden. Somit erfolgt durch XSD implizit eine Weiterentwicklung von XML, dahingehend, daß ursprünglich von SGML übernommene – jedoch inzwischen als unpraktikabel oder potentiell fehlerträchtig angesehene – Sprachbestandteile nicht mehr durch den Grammatikmechanismus unterstützt werden.[5]
Gleichzeitig wurde, neben zahlreichen anderen Neuerungen, die Kommentarsyntax für Schemata neu definiert.

Inhaltlich gliedert sich der Standard in zwei Teilbereiche: *Part 1: Structures* [Be01] zur Definition von Inhaltsmodellen für Elemente, Attributstrukturen und wiederverwendbaren Strukturen sowie *Part 2: Datatypes* [BM01] zur Festlegung diverser inhaltlicher Charakteristika wie Datentypen und konsistenzgarantierender Einschränkungen.
In beiden Teilen werden XML-Namensräume explizit berücksichtigt. Konzeptionell rekonstruiert XSD-Part 1 zunächst die bekannte Mächtigkeit der DTD, um so die evolutionäre Weiterentwicklung bestehender XML-Sprachen zu ermöglichen.
Der zweite Teil der XSD-Spezifikation definiert ein eigenständiges Typsystem, das neben der Verwendung im ersten Teil der Schemasprache auch in anderen W3C-Arbeitsgruppen Verwendung findet. Inhaltlich baut auch Part 2 auf den in der DTD definierten Typen auf und erlaubt zunächst ihre direkte Angabe in Schemata. Zusätzlich wird eine Fülle verschiedenster Typen angeboten, die an verfügbare Typsysteme aus Programmiersprachen, Datenbankmanagmentsystemen und inter-

[5] In den meisten praktische bedeutsamen Fällen wird der vermeintliche Mächtigkeitsverlust durch eigenständige ergänzende Standards kompensiert und auch weiterhin der gewohnte Sprachumfang zur Verfügung gestellt.

nationalen Standards angelehnt sind.
Wegen des Umfanges der offiziellen Schemadokumente wird durch das W3C zusätzlich ein Dokument *Part 0: Primer* [Fa01] herausgegeben, das die Verwendung der beiden XSD-Teile in der Zusammenschau an Beispielen illustriert.

Durch die Einführung der Schemasprache als weiterer Grammatiksprache büßt der im Kontext der DTD geprägte Gültigkeitsbegriff an Qualität ein. So sind Dokumente denkbar, die zwar hinsichtlich einer gegebenen DTD als *valid* gemäß des XML-Standards eingestuft werden, jedoch ein zugehöriges Schema verletzen.
Daher führt die XSD-Spezifikation zusätzlich den Terminus der *schema validness* in Erweiterung der bisherigen (DTD-bezogenen) Gültigkeit ein. Dieser neue Gültigkeitsbegriff stützt sich explizit nicht auf dem bisherigen ab und läßt die die Konformität hinsichtlich einer eventuell existierenden DTD vollständig außer Acht.
XML-Schema etabliert daher einen vom DTD-gebundenen Gültigkeitsbegriff losgelösten Validierungsmechanismus.

Aufgrund der Realisierung der Schemasprache als XML-Sprache ist jedes Schema auch ein XML-Dokument. Daher eröffnet sich die Möglichkeit, das Schema selbst durch ein Schema zu beschreiben. Dieses Metaschema, in der Originalveröffentlichung als *Schema für Schema* bezeichnet, gestattet die Validierung jedes Schemas. Damit erfüllt sich eine der Anforderungen an den Schemamechanismus: die Validierbarkeit der erstellten Schemata selbst, was für DTDs nicht gegeben war.
In der praktischen Anwendung zeigt sich dies in der Möglichkeit, erstellte Schemata mit denselben Werkzeugen zu analysieren, verarbeiten und zu prüfen, die auch für Instanzdokumente verwendet werden.
Da das Metaschema selbst wiederum ein XML-Dokument ist, folgt, daß auch hierfür ein Schema angegeben werden kann. Die XML-Standardisierung hat hier – nicht zuletzt um eine unendliche Reihung zur Validierung notwendiger Schemata zu vermeiden – den Ansatz gewählt, das Schema für Schema durch sich selbst zu beschreiben.

4.1 Schemareferenz

Jedes XML-Schema bildet als XML-Dokument eine eigenständige Speichereinheit, üblicherweise eine Datei. Mithin ist seine Einbettung in ein Instanzdokument, also die Bildung eines *internal subset*, nicht möglich.
Die Verbindung zwischen Schema und beschriebenem Dokument wird durch das in der XSD-Spezifikation vordefinierte Attribut `schemaLocation` bzw. `noNamespaceSchemaLocation` hergestellt. Eines dieser Attribute muß zwingend im Wurzelelement des XML-Dokuments angegeben werden.
Legt das Schema keinen Namensraum für die enthaltenen Deklarationen fest, d.h. alle darin deklarierten Elemente befinden sich im Vorgabenamensraum, so findet sich die Schemareferenz in `noNamespaceSchemaLocation`; andernfalls in `schemaLocation`.

4.2 Schemadefinition

Wuzelknoten jedes XSD-Dokuments ist das Element `schema`. Alle in einem Schema gegebenen Definitionen sind direkt oder transitiv als Kindknoten des `schema`-Elements angegeben. Durch die Attribute dieses Elements werden verschiedene Eigenschaften global festgelegt, die für alle im Schema definierten Elemente und Attribute gelten.

Zunächst wird durch eine Reihe von Attributen das Verhalten des Schemas in Bezug auf Namensräume festgelegt. Als Besonderheit eines XML-Schemas fällt hier die ständige Berücksichtigung von mindestens zwei Namensräumen ins Auge. Während ein Schema mit Elementen des Schemanamensraumes aufgebaut wird, trifft es zeitgleich Aussagen über einen zweiten Namensraum – den Namensraum des Vokabulars für welches das Schema erstellt wird. Dieser Namensraum wird *Zielnamensraum* (*target namespace*) genannt.
Daher findet sich im Attribut `targetNamespace` die Identifikation durch einen Universal Resource Identifikator (URI) des Zielnamensraumes. In diesen werden automatisch alle durch das Schema deklarierten Elemente und Attribute übernommen. Als Konsequenz müssen diese in jedem Schema-gültigen XML-Dokument im entsprechenden Namensraum auftreten.

4.3 Element und Attributdefinition

Als Obermenge der Ausdrucksmächtigkeit der DTD unterstützt auch XSD die Element-Inhaltsmodelle:

- unstrukturierter typisierter Inhalt
- beliebig (jedes Element kann als Kindelement angegeben werden)
- leer (keine Kindelemente)
- explizit angegebene Kindelemente
- gemischter Inhalt (gleichzeitiges Auftreten von Elementen und unstrukturiertem typisiertem Inhalt)

Unstrukturierter typisierter Inhalt:
Generell wird jedes Element durch das XSD-Element `element` und jedes Attribut durch `attribute` ausgedrückt.
Während durch den Dokument-Typ-Definitionsmechanismus für unstrukturierten Element-Inhalt und für Attribute insgesamt lediglich zeichenkettenartige Formen des Typs `(P)CDATA` angeboten wurden, definiert XML-Schema insgesamt mehr als vierzig Primitivtypen. Darunter finden sich alle aus der DTD bekannten Element- und Attributtypen, sowie eine Fülle Neuer.

Im Kern zerfallen die XSD-Typen in drei Klassen:

1. Atomare und aggregierte Typen
   Atomare Typen bestehen aus unteilbaren Werten. Der Begriff der Unteilbarkeit soll dabei auf die für den Anwender nicht erkennbare Unterstrukturierung

verweisen.[6]
Aggregierte Typen teilen sich in listenartige und Vereinigungstypen (*Union*). Listen werden dabei rekursiv als geordnete Menge atomarer Typen aufgefaßt. Vereinigungstypen erlauben hingegen die Verknüpfung typverschiedener Datentypen zu einem Neuen.

2. Primitive und abgeleitete Typen
   Primitive Datentypen existieren „aus sich heraus“ unabhängig von anderen Datentypen, während abgeleitete Datentypen von der Definition ihres Elterntyps abhängig sind.

3. Vorgegebene und anwenderdefinierte Typen
   Die vorgegebenen Typen sind ausschließlich diejenigen, die durch XML-Schema Part 2 eingeführt werden. Alle weiteren Typen eines Schemas sind durch den Anwender definierte abgeleitete Typen.

Durch Erweiterungs- und Aggregationsmechanismen ergibt sich ein Typsystem, das neben aus den Programmiersprachen bekannten Skalartypen wie `integer, byte` oder `boolean` auch inhärent strukturierte Primitivtypen wie beispielsweise einen Datumstyp umfaßt.

Die einfachste Form zur Definition eines Elements mit unstrukturiertem typisierten Inhalt lautet: `<xsd:element name=“`*elementName*`“ type=“`*elementType*`“/>`; entsprechend die Formulierung für Attribute `<xsd:attribute name=“`*attributeName*`“ type=“`*attributeType*`“/>`.

XSD definiert ferner spezifische Charakteristika für Elemente und Attribute, die durch Attribute der entsprechenden Deklarationen im XML-Schema ausgedrückt werden. So können einzelne Elemente als `abstract` deklariert werden, um ihre Verwendung in XML-Dokumentinstanzen zu verbieten. Ihnen kommt damit dieselbe Rolle wie abstrakten Klassen in der objektorientierten Programmierung zu.
Für Attribute und skalartypisierte Elemente ist es zusätzlich möglich, konstante Inhalte sowie Vorgabebelegungen festzulegen.
Mechanismen zur numerischen Steuerung der Auftrittshäufigkeit einzelner Kindelemente reformulieren die symbolische Syntax der DTD für Element-Inhaltsmodelle.
Abgerundet wird die Mächtigkeit der Deklarationen von Elementen und Attributen durch Zuweisungen, welche die Kontrolle evtl. im Schema plazierter Vererbungsstrukturen gestatten.

Beliebiger Inhalt:
Das bereits in SGML präsentierte, jedoch in XML deutlich restriktiver ausgelegte, freie Inhaltsmodell, welches Dokumentinstanzen gestattet, wahlfrei beliebige Elemente des durch die DTD vorgegebenen Sprachumfanges als Kindelemente eines XML-Elements zu verwenden, wird durch W3Cs XML-Schema präzisierend aufgegriffen.

[6] Beispielsweise werden Ausprägungen des Typs `string` zwar aus einzelnen Zeichen gebildet, jedoch sind diese nicht separat zugreifbar bzw. falls die Unterstrukturierung erkennbar ist, werden – wie bei Datumstypen – die Zugriffe auf die einzelnen Komponente nicht explizit unterstützt.

Der neue Grammatikmechanismus gestattet auf Ebene des Vokabulars durch explizite Berücksichtung des Herkunftsnamensraums die Differenzierung hinsichtlich des elementliefernden Vokabulars.
Zusätzlich wird das Prinzip auch auf Attribute übertragen, wodurch XML-Standards wie die bekannten erweiterten Hyperlinks (XLink [De01]), die ausschließlich eine aus XML-Attributen bestehende Sprache definieren, erst nutzbar werden.

Leerer Inhalt:
XML-Schema prägt zusätzlich den bereits im Kontext der DTD entwickelten rudimentären Typbegriff strenger. Dies zeigt sich deutlich in der Existenz des XSD-Elements `complexType`. Es führt die Möglichkeit einer expliziten, d.h. von der Verwendung losgelösten, Typbildung für Elemente ein. Syntaktisch kann diese Aussage sowohl innerhalb einer Elementdefinition, als auch separat erfolgen. Den einfachsten Anwendungsfall bildet die eingebettete leere `complexType`-Definition zur Darstellung des leeren Inhaltsmodells.

Ein XML-Schema-validierender Parser verhält sich in diesem Falle identisch zu einem (DTD-)validierenden Parser für das Inhaltsmodell `EMPTY`. Daher werden für die obige Festlegung ausschließlich die beiden Darstellungsformen zur Angabe eines leeren Elements (`<elementName/>` bzw. `<elementName></elementName>`) akzeptiert. Diese Interpretation wird auch durch die XML-Semantikfestlegung des *XML Information Sets* [CT01] gestützt, in der keine Unterscheidung der beiden Syntaxvarianten getroffen wird.

Explizit spezifizierter Inhalt:
Die inhaltliche Ausdetaillierung des `complexType`-Elements leitet direkt zum in der Praxis vorherrschenden Inhaltsmodell über, dem explizit angegebener Kindelemente.
Während die DTD das sequentielle Auftreten der Kindelemente in der angegebenen Reihenfolge nur implizit zu Grunde legt, stellt XML-Schema diesen Sachverhalt durch ein eigenes Sprachkonstrukt klar dar. Hierfür werden alle Kindelemente in ein weiteres Element `sequence` eingebettet.
Das Auswahlinhaltsmodell (auch: Selektionsmodell) – in der DTD durch Oder-Verknüpfung der einzelnen Elemente einer Elementgruppe ($element_1|element_2|\ldots|element_n$) ausgedrückt – wird entsprechend durch das XSD-Element `choice` ausgedrückt.
Eine besondere Variante des Selektionsmodells stellt die `all`-Gruppe dar. Sie kürzt das häufig genutzte DTD-Inhaltsmodell ($element_1|element_2|\ldots|element_n$)* ab. Es erlaubt die Angabe der Kindelemente in beliebiger Reihenfolge.
In Entsprechung zur Klammerdarstellung in der DTD muß zwingend einer der drei Reihenfolgetypen `sequence`, `choice` oder `all` als Element spezifiziert werden. Analog der Klammer-Schachtelung in der DTD können auch die verschiedenen Modellgruppen ineinander kombiniert werden.
Syntaktisch erfolgt die anwenderdefinierte Typbildung durch die Benennung des `complexType`-Elements mittels des Attributs `name`. Um die mehrfache Verwendung eines Typs zu ermöglichen, muß seine Definition zwingend auf einer Hierarchiestufe erfolgen, die für alle nutzenden Elemente erreichbar ist. Üblicherweise werden

daher diese Definitionen auf der ersten Stufe, direkt unterhalb des Wurzelknotens, plaziert. Zur Unterscheidung dieser benannten komplexen Typen werden die bisher genutzten – namenlosen Typen – als anonyme komplexe Typen bezeichnet.

Zur Unterstützung von Wiederverwendung und Erhöhung der Strukturierung des Entwurfs definiert XSD ein Vererbungskonstrukt zur Bildung neuer komplexer Typen auf der Basis bereits bestehender.
Zwei Ableitungssemantiken werden angeboten:

- Restriktion – Ableitung durch Einschränkung (*derivation by restriction*).
  Der erbende Subtyp gibt eine engere Definition des Supertypen
- Extension – Ableitung durch Erweiterung (*derivation by extension*).
  Der erbende Subtyp erweitert die Definition des Supertypen

Neben der anwenderdefinierten Bildung komplexer Typen steht es dem XSD-Modellierer auch offen, eigene (primitive) Datentypen festzulegen oder eigene Typen von bestehenden abzuleiten.
Hierfür definiert XML-Schema Part 1 das Element `simpleType`. Für einfache Typen ist jedoch nur die Restriktion zugelassen. Dies liegt in der praktischen Beherrschbarkeit des Typsystems begründet. Durch die strikte Semantik ergibt sich die Möglichkeit kontravarianter Substitution, wie sie bei objektorientierten Typsystemen und Vererbungsstrukturen anzutreffen ist. Dies bedeutet, daß an jeder Stelle, an der eine Ausprägung eines Supertyps erwartet wird, auch – unter Erhalt der Typrestriktion – eine Ausprägung eines Subtypen auftreten darf. Beispielhaft: Wird an einer Stelle des Instanzdokumentes durch das Schema das Auftreten einer Ausprägung von `integer` verlangt, so kann der Anwender auch Ausprägungen der Subtypen `int`, `short` oder `byte` angeben ohne die Gültigkeit des XML-Dokuments zu beeinträchtigen.

Zur Realisierung anwenderdefinierter Primitivtypen gestattet XML-Schema sowohl die durch sog. *einschränkende Facetten* kontrollierte Beschränkung bestimmter Charakteristika – wie etwa gültige Dezimalziffern, Gesamtlänge des Ausdrucks oder auch Domänenrestriktionen – als auch die mustergetriebene Definition durch Angabe eines regulären Ausdrucks.

5 Zusammenfassung und Ausblick

Die Grundidee des XML-Schemaansatzes erweist sich – insbesondere unter Berücksichtigung der im Umfeld moderner datenorientierter XML-Anwendungen immer klarer zu Tage tretenden Limitierungen der Document Type Definitions – als notwendiger Schritt.
XML zur Realisierung einer universell einsetzbaren Grammatikdefinitionssprache bildet hierbei nur einen logisch konsequenten Schritt, der die Allgemeingültigkeit des Ansatzes einer generischen Metasprache eindrucksvoll unterstreicht. Zusätzlich führt er zu einer Bereinigung des technischen Umfeldes, da sowohl Sprache als auch Sprachbeschreibung auf dieselben Realisierungsbausteine zurückgreifen können.

Rückblickend ist festzuhalten, daß die Existenz verschiedener am Markt offen konkurrierenden Sprachvorschläge zwar einerseits zur (sicherlich vermeidbaren) Verwirrung seitens der Anwender führte, anderseits barg sie jedoch die Chance der Bewußtseinsbildung hinsichtlich der Notwendigkeit eines gegenüber der DTD erweiterten Grammatikmechanismus, und gleichzeitig die Einsicht in die Möglichkeiten der Metasprache XML auch für diesen Anwendungsfall.
Mit dem verabschiedeten Standard des W3C Konsortiums liegt eine umfassende und auf den bekannten Strukturierungsprinzipien aufbauende Sprache zur Definition eigener XML-Vokabulare vor, die durch Werkzeuge und Systeme unterstützt wird. Sie stellt das erste Element einer neuen Generation von Web-Standards dar. Als Weiterentwicklung der Basisspezifikation XML v1.0 2^{nd} Edition fundiert sie die Grundlagen wichtiger Sekundärformate wie HTML, XSLT und XQuery.
Als ein notwendiger nächster Schritt sollte innerhalb der inzwischen gewachsenen Standardfamilie des World Wide Web Consortiums eine Konsolidierung eintreten, um die wechselseitigen Abhängigkeiten, insbesondere zwischen dem durch XML-Schema eingeführten Typsystem und dem innerhalb der Arbeitsgruppe für eine XML-Anfragesprache erarbeiteten XML-Datenmodell, aufzulösen und durch Weiter- und Wiedernutzung bereits erzielter Ergebnisse die möglichen Synergieeffekte auszunutzen.

Literatur

[BM01] Biron, P. V.; Malhotra, A. (editors): XML Schema Part 2: Datatypes. World Wide Web Consortium, Boston, USA, 2001. W3C Recommendation. URL: http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/.

[Be01] Beech, D.; Lawrence, S.; Maloney, M.; Mendelsohn, N.; Thompson, H. S. (editors): XML Schema Part 1: Structures. World Wide Web Consortium, Boston, USA, 2001. W3C Recommendation. URL: http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/.

[Bo99] Bourret, R.; Cowan, J.; Macherius, I.; St. Laurent, S. (editors): Document Definition Markup Language (DDML) Specification. World Wide Web Consortium, Boston, USA, January 1999. W3C Note. URL http://www.w3.org/TR/1999/NOTE-ddml-19990119.

[Br99a] Bray, T.; Frankston, C.; Malhotra, A. (editors): Document Content Description for XML. World Wide Web Consortium, Boston, USA, July 1999. W3C Note. http://www.w3.org/TR/1998/NOTE-dcd-19980731.

[Br99b] Bray, T.; Hollander, D.; Layman, A. (editors): Namespaces in XML. World Wide Web Consortium, Boston, USA, January 1999. W3C Recommendation. URL: www.w3.org/TR/REC-xml-names.

[Br00] Bray, T.; Paoli, J.; Sperberg-McQueen, C. M.; Maler, E. (editors): Extensible Markup Language (XML) 1.0 (Second Edition). World Wide Web Consortium, Boston, USA, October 2000. W3C Recommendation. URL http://www.w3.org/TR/2000/REC-xml-20001006.

[Bu00] Buck, L:, Goldfarb, C. F.; Prescod, P. (editors): Datatypes for DTD. World Wide Web Consortium, Boston, USA, January 2000. W3C Note. URL: http://www.w3.org/TR/2000/NOTE-dt4dtd-20000113.

[CD99] Clark, J.; DeRose, S. (editors): XML Path Language (XPath). World Wide Web Consortium, Boston, USA, November 1999. W3C Recommendation. http://www.w3.org/TR/1999/REC-xpath-19991116.

[CT01] Cowan, J.; Tobin, R. (editors): XML Information Set. World Wide Web Consortium, Boston, USA, October 2001. W3C Recommendation. http://www.w3.org/TR/2001/REC-xml-infoset-20011024.

[Cl99] Clark, J. (editor): XSL Transformations (XSLT) Version 1.0. World Wide Web Consortium, 1999. W3C Recommendation. URL: http://www.w3c.org/TR/xslt.

[Da99] Davidson, A.; Fuchs, M.; Hedin, M.; Jain, M.; Koistinen, J.; Lloyd, C.; Muloney, M.; Schwarzhof, K. (editors): Schema for Object-Oriented XML 2.0. World Wide Web Consortium, Boston, USA, July 1999. W3C Note. URL: http://www.w3.org/1999/07/NOTE-SOX-19990730.

[De00] DeRose, S.; Maler, E.; Orchard, D. (editors): XML Linking Language (XLink) Version 1.0. World Wide Web Consortium, Boston, USA, 2000. W3C Recommendation. URL: http://www.w3.org/TR/2001/REC-xlink-20010627/.

[Fa01] Fallside, D. C. (editor): XML Schema Part 0: Primer. World Wide Web Consortium, Boston, USA, 2001. W3C Recommendation 2001-05-02, URL: http://www.w3.org/TR/2001/REC-xmlschema-0-20010502/.

[In86] International Organization for Standardization (editor): Information processing – Text and online systems – Standard Generalized Markup Language (SGML). International Organization for Standardization, 1986. ISO 8879:1986, Geneva, CH.

[Je00] Jelliffe, R.: The Schematron Assertion Language. R. Jelliffe, 2000. URL: http://www.ascc.net/xml/schematron.

[LS99] Lassila, O.; Swick, R. R. (editors): Resource Description Framework (RDF) Model and Syntax Specification. World Wide Web Consortium, Boston, USA, February 1999. W3C Recommendation. URL: http://www.w3.org/TR/1999/REC-rdf-syntax-19990222.

[La98] Layman, A.; Jung, E.; Maler, E.; Thompson, H. S.; Paoli, J.; Tigue, J.; Mikula, N. H.; DeRose, S. (editors): XML-Data. World Wde Web Consortium, Boston, USA, 1998. W3C Note. URL: http://www.w3.org/TR/1998/NOTE-XML-data-0105.