

Westfälische Wilhelms-Universität Münster  
Englisches Seminar Sommersemester 2003/04  
Proseminar: Syntax  
Lea Cyrus

---

Hausarbeit zu dem Thema:

# **XML in Treebanks**

Martin Pyka  
martin.pyka@gmx.de  
2. Fachsemester

Studiengang: Dipl. Informatik mit Text- und Sprachwissenschaft

Münster, den 29.09.2003

# Inhaltsverzeichnis

<b>I</b>	<b>Einleitung</b>	<b>3</b>
<b>II</b>	<b>Was ist XML?</b>	<b>3</b>
II.1	Die Idee von XML	3
II.2	Bestandteile von XML	5
II.2.1	Aufbau und Struktur eines XML-Dokumentes	5
II.2.2	Dokumenttyp-Definition	6
II.2.3	XML Schema	9
II.2.4	XSL und XSLT	10
II.3	Anwendungsmöglichkeiten	11
<b>III</b>	<b>Der Einsatz von XML in Treebanks</b>	<b>12</b>
III.1	Forschungsstand	12
III.1.1	Text Encoding Initiative	12
III.1.2	XCES	13
III.2	Diverse Projekte	16
<b>IV</b>	<b>Vorteile und Möglichkeiten von XML</b>	<b>19</b>
<b>V</b>	<b>Fazit</b>	<b>20</b>

# I Einleitung

Eine Fülle an Treebanks stehen der Forschung zur Verfügung, doch die verschiedenen Theorien, die dahinter stecken, und die damit verbundenen unterschiedlichen Speicherformate können einen annotierten Korpus zu einer Wissensinsel isolieren. Theorie- und sprachübergreifende Zusammenhänge zwischen den einzelnen Treebanks lassen sich nur dann untersuchen, wenn der Aufwand für die Darstellung und Konvertierung von diversen Korpusformaten zeitlich vertretbar und technisch finanzierbar ist. Helfen kann dabei ein standardisiertes Annotierungsschema - also ein Regelwerk das auf der einen Seite allen Annotierungsschemas gemeinsam ist, auf der anderen Seite aber flexibel genug für die spezifischen Anforderungen der einzelnen Projekte. So ein Standard könnte XML sein.

„XML in Treebanks“ ist das Thema dieser Hausarbeit und die Ideen, Konzepte und Möglichkeiten, die sich dahinter verbergen, ihr Inhalt. In den nachfolgenden Kapiteln soll die Beschreibungssprache XML erläutert werden, sowie ihre Anwendung in Treebanks. Dabei werden konkrete standardisierte Annotierungsverfahren und Projekte, die darauf aufbauen, vorgestellt, sowie Möglichkeiten, die bislang unausgeschöpft blieben, beschrieben. Ziel dieser Hausarbeit ist es, aufzuzeigen, welche Chancen und welches Potential sich hinter dieser Beschreibungssprache verbirgt, und wie die Korpuslinguistik davon profitieren kann.

## II Was ist XML?

### II.1 Die Idee von XML

Hinter der *Extensible Markup Language* (XML) verbirgt sich ein Konzept, dass sich mit der Speicherung und Darstellung von Daten beschäftigt. Der Grundgedanke dabei ist, die Speicherung der Daten so transparent und die Darstellung der Daten so flexibel wie möglich zu halten, so dass Dokumente, die von Programmen in Dateien gespeichert werden, nicht zwangsläufig auch an diese Programme gebunden sind. Während also zum Beispiel Microsoft Word-Dokumente nur von Microsoft-

Produkten oder von Programmen mit einer entsprechenden Lizenz gespeichert und gelesen werden können, sind Dokumente, die in HTML dargestellt werden, für jeden frei zugänglich. Dieser offene Standard, der durch das W3C<sup>1</sup> entwickelt wurde, ermöglicht es der Softwareindustrie eine Vielzahl von Programmen zu entwickeln, die alle in der Lage sind Dokumente des selben Formates auszulesen (auch parsen genannt), darzustellen, zu bearbeiten und zu speichern. Der Anwender sieht sich also nicht mehr an eine bestimmte Software gebunden, sondern kann sich entsprechend seinen Wünschen und Anforderungen das für ihn beste Produkt aussuchen, um seine Informationen verwalten und bearbeiten zu können.

Unabhängig von einer bestimmten Software zu sein, ist daher ein zentraler Vorteil eines offenen Standards. Eine flexible Darstellung von Informationen erscheint in der elektronischen Datenverarbeitung aber fast noch wichtiger. Die Print-Ausgabe eines Artikels muss beispielsweise anders gestaltet werden, als die dazugehörige Webseite oder Blindenschriftausgabe. Datenbankinhalte müssen für die Darstellung im Browser nach HTML konvertiert werden, während sich Tabellenkalkulationsprogramme selbst um die Visualisierung kümmern.

Diese beiden Beispiele zeigen, dass sich ein höherer Nutzen von Informationen und mehr Flexibilität erzielen lässt, wenn man die Daten von ihrer Darstellung strikt trennt. Wenn also die konkrete Darstellung von der Software oder einer Schnittstelle zwischen Software und Daten übernommen wird und die Informationen lediglich in irgendeiner Form strukturiert vorliegen.

Diese Struktur spiegelt sich im Speicherformat wieder, das für die Softwareentwicklung beschrieben werden muss. XML ist so eine Beschreibungssprache (= Markup Language) und in ihrem Umfang und ihren Möglichkeiten universell einsetzbar. Dabei werden die einzelnen Daten eines Dokumentes lediglich durch sogenannte Tags in einen semantischen Zusammenhang gebracht, so dass eine hierarchische Ordnung des gesamten Dokumentes erkennbar wird. Die nächsten Unterkapitel werden dies näher erläutern.

Des Weiteren ist es möglich eine Dokumenttyp-Definition (DTD) anzulegen, durch die sich Regeln bzw. Grammatiken beschreiben lassen, die für jedes XML-Dokument

---

<sup>1</sup> World Wide Web Consortium; <http://www.w3c.org>

gelten sollen. Auf diese Weise kann ein bestehendes XML-Dokument auf seine Gültigkeit hinsichtlich dieser DTD überprüft werden. Dadurch lässt sich eine konsequentere Einhaltung eines definierten Standards gewährleisten.

Das Konzept von XML ist nicht neu. Schon 1986 wurde die *Standard Generalized Markup Language* (SGML) eingeführt, in der XML eine Untermenge darstellt. Für den Standard XML wurden viele optionale Einstellungsmöglichkeiten herausgenommen, um es schlanker und konsistenter zu machen, bei gleichbleibendem Leistungsumfang. Dadurch ist XML weit aus einfacher und für die Verwendung und Verbreitung im Internet besser geeignet.<sup>2</sup>

XML kann also beliebige Informationen (Text-Dokumente, Datenbank, Grafiken, Multimedia-Dateien) strukturieren und hierarchisch ordnen und dank einer Beschreibung dieser Struktur (DTD) Programmen zugänglich machen. Zukünftig soll es damit nicht mehr Programme für bestimmte Formate geben, sondern Programme für bestimmte Funktionen.

## II.2 Bestandteile von XML

### II.2.1 Aufbau und Struktur eines XML-Dokumentes

Eine XML-Datei ist eine reine Textdatei. Das heißt, ihr Inhalt ist mit einem beliebigen Texteditor editierbar. Die Struktur eines XML-Dokumentes lässt sich am besten als Baum beschreiben, bestehend aus Tags als semantische Kategorien und Daten in Form von gewöhnlichen Zeichenketten. Ein Tag ist dabei ein durch < und > umschlossenes Schlüsselwort. Jedes Tag wird auch als Anfangstag bezeichnet, das durch ein Endtag in der Form </Schlüsselwort> geschlossen werden muss. Innerhalb dieses Bereiches können weitere Tags oder Daten folgen. Ein Tag kann zu dem über Attribute verfügen, denen Werte zugewiesen werden. Dazu ein Beispiel:

```
<?xml version="1.0"?>
<Satz ID=1>
  <!-- dies ist ein Kommentar, der kein Endtag besitzt -->
  <NP>
    <Det>The</Det>
    <N>manager</N>
  </NP>
```

<sup>2</sup> Vgl. [http://www.oreilly.de/xml/xml\\_faq\\_a5.html](http://www.oreilly.de/xml/xml_faq_a5.html), aufgerufen am : 16.09.2003

```

<VP>
  <V zeit="pres">relies</V>
  <PP>
    <Pr>on</Pr>
    <NP>
      <Pron>his</Pron>
      <N>employees</N>
    </NP>
  </PP>
</VP>
</Satz>

```

Dieser XML-Code zeigt die syntaktische Aufschlüsselung eines englischen Satzes gemäss den Regeln von Phrasenstrukturbäumen. Dazu wird in der ersten Zeile die XML-Version deklariert, die für dieses Dokument verwendet wurde. Der eigentliche Inhalt des Dokumentes beginnt in der zweiten Zeile und ist von einem Satz-Tag umschlossen, das wiederum aus einem NP- und einem VP-Tag besteht, die sich weiter aufschlüsseln lassen. Die Tags Satz und V besitzen darüber hinaus noch Attribute, die den Inhalt dieses Tags näher beschreiben. So wird der Satz-Tag mit einer Identifikationsnummer ausgezeichnet und dem Inhalt des Tags V ein Zeit-Attribut zugewiesen.

Das Beispiel verdeutlicht, dass hier die eigentlichen Daten, nämlich die Zeichenfolge „*The manager relies on his employees*“ mit sogenannten Metadaten angereichert werden, also Informationen über den eigentlichen Inhalt. Zu beachten ist dabei, dass diese Metadaten keinen Bezug zu einem programmtechnischen Kontext haben. Das heisst, je nach Anwendung und Aufgaben, können diese Tags auf unterschiedliche Weise zur Weiterverarbeitung oder Darstellung des Inhaltes verwendet werden.

Die bereits angesprochene hierarchische Struktur eines XML-Dokumentes macht es zu dem leicht, einzelne Elemente eines Dokumentes zu Gruppen zusammen zu fassen, die sich als Elemente einer abstrakteren Kategorie sehen und verwenden lassen. Welche Vorteile diese Sicht auf Informationen und deren Darstellung hat, wird in den Kapitel Anwendungsmöglichkeiten **II.3** und Vorteile und Möglichkeiten **IV** erläutert.

## II.2.2 Dokumenttyp-Definition

Ein XML-Dokument, wie es im oberen Beispiel aufgeführt wird, wird als „wohlgeformt“ bezeichnet, da es die syntaktischen Regeln von XML einhält. Dazu gehört im wesentlichen die XML-Deklaration in der ersten Zeile sowie ein äußerstes

Tag (auch Dokument-Element genannt), das das gesamte Dokument umschließt. Ein Dokument, das lediglich diesen Ansprüchen genügt, lässt sich bereits von einem Programm korrekt parsen und weiterverarbeiten. Je nach Verwendungszweck eines XML-Dokumentes wird es allerdings nötig Tag-Namen, sowie die hierarchische Struktur, zu definieren und durch Regeln zu beschreiben. So ein Regelwerk wird als Dokumenttyp-Definition (DTD) bezeichnet und kann entweder Bestandteil des XML-Dokumentes selber sein oder in einer externen Datei ausgelagert werden. Letzteres ist der Regelfall, da dadurch die gesamte DTD für einen beliebigen Bestand von XML-Dateien nur einmal vorhanden sein muss. Innerhalb der XML-Datei reicht dann lediglich ein Verweis auf diese DTD, der folgendermaßen aussieht.

```
<?xml version="1.0"?>
<!DOCTYPE regeln SYSTEM "regeln.dtd">
<!-- Inhalte -->
```

Die innerhalb dieser DTD definierten Regeln dienen auch dazu, eine bestehende XML-Datei auf ihre „Gültigkeit“ hin zu überprüfen, bevor die eigentliche Verarbeitung der Daten beginnt. Dadurch lässt sich wesentlich leichter eine Standardisierung von Formatbeschreibungen durchsetzen.

Einen weiteren Vorteil von DTDs beschreibt Stefan Münz in seiner Dokumentation SelfHTML:

*"[...] die DTD ist - selbst wenn die Daten im Verarbeitungsprozess nicht auf Gültigkeit getestet werden - in jedem Fall eine Dokumentation des gewünschten Sollzustands der Daten. Vor allem, wenn Datenbestände über einen längeren Zeitraum aufbewahrt und von verschiedenen Personen weiterverarbeitet oder weitergepflegt werden, ist eine DTD eine wichtige normative Grundlage für die Korrektheit und Einheitlichkeit der Daten."<sup>3</sup>*

DTDs sind ebenfalls reine Textdateien die mit einem beliebigen Editor oder mit inzwischen erhältlichen Hilfsprogrammen bearbeitet werden können. Die Regeln für ein Tag, auch Element genannt, werden nach folgendem Schema definiert:

```
<!ELEMENT Name (Inhalt)>
```

---

<sup>3</sup> Vgl. <http://selfhtml.teamone.de/xml/dtd/allgemeines.htm#notwendigkeit>, aufgerufen am: 16.09.2003

Nach einem einleitenden `<!` folgt das Schlüsselwort `ELEMENT`, durch das die Definition eines neuen Elementes eingeleitet wird. Daraufhin folgt der Name des Elementes sowie, durch runde Klammern umschlossen, eine Liste der möglichen Inhalte, durch Kommata getrennt. Dies können weitere Elementnamen oder das Schlüsselwort `#PCDATA` für Textinhalte sein. Ein beispielhaftes (hier der Einfachheit halber unvollständiges) Regelwerk für das auf Seite 6 verwendete XML-Dokument könnte so aussehen:

```
<!ELEMENT Satz (NP, V)>
<!ELEMENT NP (Det?, N+)>
<!ELEMENT VP (V, PP*)>
<!ELEMENT Det (#PCDATA)>
<!ELEMENT N (#PCDATA)>
<!ELEMENT V (#PCDATA)>
<!-- weitere Elementdefinition... -->
```

Alle Elemente, die innerhalb der runden Klammern aufgelistet sind, müssen in der vorgeschriebenen Reihenfolge in der XML-Datei vorkommen. Ein Satz-Tag beinhaltet demnach zwingend ein NP-Tag und ein V-Tag. Das Fragezeichen hinter `Det` gibt an, dass dieses Element nur optional vorkommen muss. Ein Pluszeichen, wie es hier hinter dem `N`-Tag aufgeführt ist, markiert ein Tag, das beliebig oft aber mindestens einmal vorkommen muss. Ein Stern-Zeichen impliziert, dass das jeweilige Tag beliebig oft aber auch kein mal vorkommen darf. Auch das Schlüsselwort `#PCDATA` lässt sich mit anderen Elementnamen in eine Liste fassen. Es muss nicht zwingend als einziger Inhalt eines Tags angegeben werden. Darüber hinaus gibt es noch das Schlüsselwort `EMPTY`, das einen leeren Tag beschreibt.

Attribute lassen sich den definierten Elementen über folgendes Schema zuweisen:

```
<!ELEMENT Elementname (Inhalt)>
<!ATTLIST Elementname
  Attributname_1 Inhalt [#REQUIRED|#IMPLIED|#FIXED] "Wert"|Defaultwert
  Attributname_n Inhalt [#REQUIRED|#IMPLIED|#FIXED] "Wert"|Defaultwert
>
```

Ohne näher ins Detail zu gehen, da dies für das Thema der Hausarbeit nicht weiter nötig ist, sei an dieser Stelle kurz erwähnt, dass das Schema der DTD diverse Einstellungen bietet, die es beispielsweise möglich machen, Attribute als optional zu definieren oder vorgegebene Wertzuweisungen vorzuschreiben. Darüber hinaus können Tags über eindeutige Identifikationswerte verfügen, indem das Tag über ein

Attribut namens ID verfügt. Auf diese Tags kann von anderen Tags aus über das Attribut IDREF verwiesen werden, um so beispielsweise Querverweise zu ermöglichen.

Durch diese und einige andere Fähigkeiten der DTD-Sprache ist es dem Benutzer möglich selbst komplexe Zusammenhänge in abstrakte hierarchische Strukturen zu setzen und so eine konsistente Grammatik für die XML-Dateien zu entwickeln. Auf diese Weise können Programme bestehende XML-Dateien auf ihre Gültigkeit hin überprüfen und einlesen und neue XML-Dateien gemäß den in der DTD-Datei definierten Regeln speichern, ohne selbst irgendwelche zusätzlichen Informationen über das Speicherformat zu besitzen. Die DTD funktioniert also als eine Art Schnittstelle zwischen dem Programm und den Informationen. Dadurch können Programm und Informationen völlig unabhängig von einander weiterentwickelt und verändert werden.

### II.2.3 XML Schema

Die Beschreibung und Validierung einer Datenstruktur mittels einer DTD war lange Zeit der einzige Weg, um einen Standard für ein bestimmtes Format zu definieren. Allerdings fehlen DTDs einige Funktionen, die bei der Definierung von bestimmten Formaten wünschenswert sind. So bietet diese Sprache keine Datentypen, wie integer, string oder boolean. Des Weiteren besteht die DTD nicht selbst aus XML-Syntax.

Die damit verbundenen Schwierigkeiten bei der Format-Definition wurden durch einen im Jahr 2001 eingeführten Standard *XML Schema* gelöst. Dieser bietet neben den erforderlichen Datentypen und einer absolut XML-konformen Syntax, wesentlich feinere Einstellungsmöglichkeiten und die freie Kombinierbarkeit von XML Schemas innerhalb eines XML-Dokumentes. Nachfolgend sei eine DTD-Definition als XML Schema dargestellt.

#### DTD-Definition

```
<!ELEMENT Book (Title, Author, Date, ISBN, Publisher)>
```

#### XML Schema-Definition

```

<?xml version="1.0" encoding="UTF-8"?>
<!--W3C Schema generated by XML Spy v3.5 NT (http://www.xmlspy.com)-->
<xsd:schema xmlns:xsd="http://www.w3.org/2000/10/XMLSchema"
elementFormDefault="qualified">
  <xsd:element name="Book">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="Title"/>
        <xsd:element ref="Author"/>
        <xsd:element ref="Date"/>
        <xsd:element ref="ISBN"/>
        <xsd:element ref="Publisher"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>

```

4

Nichts desto trotz ist die Definition eines Formates über eine DTD immer noch der gebräuchlichste Weg. DTDs wurden (bisher) von XML Schemas nicht abgelöst.

## II.2.4 XSL und XSLT

Die *Extensible Stylesheet Language* (XSL) ist eine XML-Sprache zur Formatierung und Konvertierung von XML-Dateien in andere XML-Formate. Eine grosse und mächtige Komponente innerhalb dieser Sprache ist dabei der Wortschatz der *XSL Transformation* (XSLT). Mit Hilfe dieser Komponente lassen sich für Tags, und in Abhängigkeit von ihren Attributen und Inhalten, Regeln aufstellen, nach denen der Inhalt in eine andere Sprache konvertiert werden soll. XSLT eignet sich daher beispielsweise besonders gut, um Inhalte nach HTML zu konvertieren und damit für einen Browser darstellbar zu machen.

Bezüglich der Korpusannotation kann diese Sprache sinnvoll sein, um Korpora von einem Format in ein anderes zu übertragen oder für ein Programm in eine lesbare Struktur umzuwandeln. Das XSLT-Framework bietet dazu Schlüsselwörter, mit denen sich Wenn-Bedingungen formulieren lassen, sortiert und nummeriert werden kann und eine umfangreiche Navigation im Strukturbaum erlaubt.

## II.3 Anwendungsmöglichkeiten

Das Konzept, Informationen zu abstrakten Gruppen zusammen zu fassen und zu beschreiben, lässt sich in fast allen Bereichen der elektronischen Datenverarbeitung anwenden. Vor allem im Webbereich gibt es neben HTML (einer nicht ganz konsequenten XML-Sprache) einige Standards, die für wissenschaftliche Zwecke

---

<sup>4</sup> Vgl. [http://www.devtrain.de/artikel\\_445.aspx](http://www.devtrain.de/artikel_445.aspx), aufgerufen am : 17.09.2003

relevant sind. Beispielsweise MathML, eine Beschreibungssprache für mathematische Formeln. Die meisten Browser (Internet Explorer ab 5.0, Netscape ab 6.1, Mozilla ab 0.9.9)<sup>5</sup> unterstützen diesen Standard bereits. Durch MathML lassen sich komplexe mathematische Terme ausdrücken und im Browser, oder einer speziellen Anwendung, mit den geeigneten mathematischen Zeichen und Größenverhältnissen anzeigen.

Doch auch für andere wissenschaftliche Bereiche gibt es bereits spezielle auf XML-basierende Frameworks. So ist die *Chemical Markup Language* (CML) eine Sprache "zur Beschreibung der Verwaltung von Molekularinformationen in Computernetzwerken"<sup>6</sup>. Mit der *Bioinformatic Sequence Markup Language* (BSML) gibt es einen Standard zur Darstellung von DNS-, RNS- und Proteinsequenz-Informationen.<sup>7</sup>

Grossen Anwendungsbedarf aber bietet XML vor allem in der gesamten Softwareindustrie, sei es in der Text-, Grafik- oder Filmverarbeitung, in Datenbankanwendungen oder Analysetools jeglicher Art. Die Vereinheitlichung und Standardisierung von Dateiformaten vereinfacht das Portieren von Daten zu anderen Anwendungen oder macht es sogar gänzlich überflüssig, da beispielsweise alle Textverarbeitungsprogramme das selbe Format verstehen und darstellen können. Stefan Münz schlussfolgert daher:

*"XML könnte dafür sorgen, dass es in Zukunft egal ist, womit Sie Ihre Briefe schreiben, Ihre Häuser konstruieren oder Ihre Plakatwerbung erstellen. Wenn die benutzte Software sich konsequent an den XML-Standard hält und nachweisbare Dokumenttypen (DTDs) benutzt, dann steht einem verlustfreien Portieren von Daten zwischen konkurrierenden Anwendungen mit gleichen Fähigkeiten eigentlich nicht mehr viel im Wege."<sup>8</sup>*

Dies ist auch der Grund, warum XML für Treebanks so interessant sein kann.

---

<sup>5</sup> weitere Informationen unter <http://www.w3.org/Math/XSL/>

<sup>6</sup> Vgl. Goldfarb, Prescod: "Das XML-Handbuch". 2. Auflage. München 2000 S.96

<sup>7</sup> Vgl. Goldfarb, Prescod: "Das XML-Handbuch". 2. Auflage. München 2000 S.96

<sup>8</sup> Vgl. <http://www.teamone.de/selfhtml/xml/intro.htm#softwaremarkt>, aufgerufen am: 18.09.2003

### III Der Einsatz von XML in Treebanks

In den nachfolgenden Abschnitten wird erörtert, in wie weit XML in der Annotierung von Korpora eingesetzt wird, welche Institutionen und Standards es bereits gibt, und welche Projekte darauf aufbauen.

#### III.1 Forschungsstand

Die Text Encoding Initiative und der XML Corpus Encoding Standard sind, auf XML/SGML basierend, die beiden wichtigsten Initiativen, die sich mit der Aufbereitung und Analyse von Texten beschäftigen und der Erstellung von Treebanks im speziellen. Die nächsten beiden Abschnitte stellen diese beiden vor.

##### III.1.1 Text Encoding Initiative

Die *Text Encoding Initiative* (TEI) wurde 1984 im Rahmen eines Forschungsprojektes von der *Association for Computers and the Humanities*<sup>9</sup>, der *Association for Computational Linguistics*<sup>10</sup> und der *Association for Literary and Linguistic Computing*<sup>11</sup> gegründet. Die TEI ist ein internationaler und interdisziplinärer Standard für die Aufbereitung von literarischen und linguistischen Texten zur Forschung und Lehre. Um dieser sehr allgemein gehaltenen Zielvorgabe gerecht zu werden, bemüht sich die TEI *"ein Regelwerk zu bestimmen, daß dem Anwender möglichst viel Freiheit überläßt und möglichst wenige Vorentscheidungen trifft."*<sup>12</sup> Dieses Regelwerk basiert auf SGML, dem Vorläufer von XML, und bietet diverse Base Tag Sets und Additional Tag Sets. In der ersten Gruppe finden sich Auszeichnungselemente für bestimmte Literaturgattungen, wie Prosa, Lyrik, Dramen oder Wörterbücher. Das Hauptaugenmerk liegt hierbei auf der Benennung einzelner Elemente dieser Gattungen, wie Verse für Lyrik oder Absätze und Überschriften für Prosa. Diese Tagsets können untereinander und mit den Additional Tag Sets kombiniert werden, in denen sich Elemente für spezielle Verwendungszwecke finden, beispielsweise Elemente zum Hyperlinking (also dem Verknüpfen von

---

<sup>9</sup> <http://www.ach.org/>

<sup>10</sup> <http://www.cs.columbia.edu/~acl/home.html>

<sup>11</sup> <http://www.allc.org/>

<sup>12</sup> Vgl. <http://computerphilologie.uni-muenchen.de/praxis/teiprax.html> (zweiter Absatz) aufgerufen am : 20.09.2003

Inhalten mit anderen Inhalten) oder Elementen für das Beschreiben eines textkritischen Apparates.

Die Beschreibung der unterschiedlichen TEI-Auszeichnungssets umfasst weit über 1000 Seiten und ihr Umfang und ihre Komplexität erscheint selbst den Anwendern, die diesen Standard nutzen, zweifelhaft. So lassen sich in der Corpora List, einer offenen Diskussionsgruppe zum Thema Textkorpora, in dem Thread *"Is the TEI a waste of time"*<sup>13</sup> neben überzeugten Pro-TEI Statements auch kritische Stimmen nachlesen. Vor allen Dingen der Umfang der TEI-Dokumentation und fehlende Beispiele sollen dem Benutzer den Einstieg am meisten erschweren. Daraus resultiert unter anderem, dass sich die TEI trotz ihres guten und durchdachten Konzeptes nicht als das Standard-Framework etablierte, wie man es sich zum Austausch unterschiedlicher TEI-Dokumente erhofft hatte. Hinsichtlich der Annotierung von Korpora bemerkt Sylvain Loiseau in der Corpora List dazu:

*"[...] I agree that the TEI is perhaps "out to date" for some points: there is nothing for morphosyntactic or morphologic encoding, texts profiling, etc. The TEI remains perhaps not sufficiently adapted to linguistic corpora."*<sup>14</sup>

So ist die TEI in vielen Bereichen der Textspeicherung und -verwaltung für Bibliotheken, Museen und Verlage interessant, weil ihr Framework die nötige Grundlage bietet, um Texte vielseitig und programmunabhängig verwenden und weiterverarbeiten zu können. Den Anforderungen in der Korpus Annotation scheint sie jedoch nicht in dem Umfang erfüllen zu können, wie der folgende Standard.

### III.1.2 XCES

Der *Corpus Encoding Standard for XML* (XCES) ist die XML-Version des *Corpus Encoding Standard* (CES), das auf SGML basiert. XCES liegt derzeit allerdings nur in der Version Beta 0.2 vor. Diese Auszeichnungssprache ist ein Teil der Richtlinien, die von der *Expert Advisory Group on Language Engineering Standards* (EAGLES) entwickelt wurde, einer Organisation die sich in allen Bereichen der linguistischen Forschung und Technik um offene Standards und die Bereitstellung geeigneter Analysesoftware bemüht.<sup>15</sup>

<sup>13</sup> Vgl. <http://www.hit.uib.no/corpora/2003-1/0626.html>, aufgerufen am : 21.09.2003

<sup>14</sup> Vgl. <http://www.hit.uib.no/corpora/2003-1/0650.html>, aufgerufen am : 21.09.2003

<sup>15</sup> weitere Informationen unter: <http://www.tei-c.org/Applications/apps-ea01.html>

Der XCES übernimmt im Wesentlichen alle Segmente des CES und setzt diese in der XML-Umgebung konsequent fort. Er bietet eine Sammlung von DTDs und XML Schemas, die zur Speicherung von annotierten Korpora maßgeblich sind. Ziel dieses Standards ist es, ein theorie- und tagsetunabhängiges Grundgerüst zu schaffen, in dem Annotierungskonventionen einfach definiert werden können, anstatt diese vorzuschreiben.

Eine der Hauptvorteile in diesem auf XML basierenden Annotierungsstandard liegt in einer mächtigen Technik, Inhalte miteinander verknüpfen zu können. Wie SGML gibt es zwar auch in XML die Möglichkeit Tags eindeutige Identifikationswerte (IDs) zu zuweisen und über Identifikationsreferenzwerte (IDREFs) auf diese zu verweisen. Allerdings kristallisieren sich in der Praxis zwei Probleme heraus, die sich mit diesem Mechanismus nicht lösen lassen. Zum einen setzt das Verweisen auf Textstellen voraus, dass diese Textstellen auch mit einem Tag samt ID versehen sind. In Texten, die gelegentlich verändert werden erweist sich diese Bedingung als äußerst unpraktisch, zumal es in linguistischen Korpora nicht unüblich ist, auf nicht-markierte Textstellen zu verweisen. Ein vielleicht noch schwerwiegenderer Nachteil ist allerdings die Tatsache, dass Verknüpfungen nur innerhalb eines SGML/XML-Dokumentes gesetzt werden können. Verweise auf andere SGML/XML-Dokumente mit anderen DTDs oder XML-Schemas sind nicht möglich.

Die Lösung dieser beiden Probleme lautet *XML Path Language* (XPath), eine XML-konforme Verknüpfungssprache und eine Untermenge von XSLT, durch die es möglich ist, innerhalb eines Dokumentbaumes auf bestimmte Elemente und Zeichen des Textes zu verweisen. So enthält zum Beispiel das Tag

```
<tok xlink:href="substring(/p[2]/s[3]/text(),10,12)">
```

einen Verweis auf den zu annotierenden Text im zweiten <p> (Paragraph) Element innerhalb des dritten <s> (Satz) Elementes. Innerhalb dieses Textes wird auf 12 Zeichen verwiesen, beginnend mit dem zehnten. Über diesen Mechanismus ist es möglich, zeichengenaue Verweise zu verwenden, ohne die entsprechenden Textstellen zuvor durch Tags umrahmen zu müssen. Des weiteren kann der Text in einer externen Datei ausgelagert werden.

Nachfolgend sei eine konkrete Annotierung eines Satzes nach dem XCES-Schema vorgestellt. Als Vorlage dazu dient ein Auszug aus der Penn Treebank:

```
((S (NP-SBJ-1 Jones)
  (VP followed)
    (NP him)
    (PP-DIR into
      (NP the front room))
    '
  (S-ADV (NP-SBJ *-1)
    (VP closing
      (NP the door)
      (PP behind
        (NP him))))))
.))
```

Innerhalb der XML-Datei, die sich an das XCES-Schema hält, wird die syntaktische Struktur des Satzes nach einem ähnlichen Muster aufbereitet. Allerdings wird über die weiter oben erläuterte Link-Technik lediglich auf die entsprechenden Worte (genauer gesagt: Zeichenketten) verwiesen. Der Grundtext selbst ist in diesem Schema nicht eingebettet. Lediglich die Kommentare zeigen, in welchem Tag welches Wort annotiert wird.

```
<chunk xml:base="http://www.loria.fr/doc.xml#">
  <struct id=s0>
    <feat type=CAT>S</feat>
    <struct id=s1 xlink:href="xptr(substring(/p/s[1]/text(),1,5))"/>
      <!-- Jones -->
      <feat type=CAT>NP</feat>
      <rel type=SBJ head=s2 />
    </struct>
    <struct id=s2 xlink:href="xptr(substring(/p/s[1]/text(),7,8))"/>
      <!-- followed -->
      <feat type=CAT>VP</feat>
      <struct xlink:href="xptr(substring(/p/s[2]/text(),16,3))"/>
        <!-- him -->
        <feat type=CAT>NP</feat>
        <!-- implicit OBJ relation here -->
      </struct>
      <struct xlink:href="xptr(substring(/p/s[2]/text(),20,4))"/>
        <!-- into -->
        <feat type="CAT">PP</feat>
        <rel type=DIR head=s2/>
        <struct xlink:href="xptr(substring(/p/s[2]/text(),25,14))"/>
          <!-- the front room -->
          <feat type=CAT>NP</feat>
        </struct>
      </struct>
    </struct>
    <struct>
      <feat type=CAT>S</feat>
      <rel type=ADV head=s2 />
      <struct ref=s1>
        <feat type=CAT>NP</feat>
      </struct>
      <struct id=s3 xlink:href="xptr(substring(/p/s[2]/text(),41,7))"/>
        <!-- closing -->
        <feat type=CAT>VP</feat>
        <struct> </struct>
      </struct>
    </struct>
    <!-- weitere Struct-Tags -->
  </chunk>
```

Durch das Tag `<struct>` wird die hierarchische syntaktische Struktur des Textes ausgedrückt. Es kann als Attribute unter anderem einen ID-Wert besitzen sowie einen Verweis auf die konkrete Zeichenfolge, die in einer anderen Datei ausgelagert sein kann. Mittels des `<feat>`-Tags wird das umrahmte Segment genauer beschrieben. Im oberen Beispiel bezeichnet es unter anderem den Namen der jeweiligen Phrase. Dieses Tag könnte allerdings auch einen Verweis auf einen Ort beinhalten, an dem sich genauere Informationen zum jeweiligen Segment befinden, beispielsweise Anzahl, Geschlecht etc.. Hier liegt es in der Freiheit des Anwenders, geeignete Kategorien zu definieren und Informationen bereit zu stellen.

XCES wird derzeit unter anderem um folgende Eigenschaften erweitert:

- Unterstützung unterschiedlicher Annotierungsarten für Sprachdaten
- mehrere Annotierungsebenen, die sich auf einander beziehen können
- gleichzeitiges Verwenden von mehreren Tagset-Schemas und Standards
- Integration der Standardisierungsbemühungen in den Vereinigten Staaten, Europa und Japan

Insgesamt stellt also XCES ein sehr umfangreiches und leistungsfähiges Framework dar, über das Nancy Ide resümiert:

*"The XML framework provides search, extraction, and transformation capabilities that answer most, if not all, of the current and foreseen needs for corpus-based language engineering. In particular, XML provides mechanisms for easily implementing the CES and XCES data architecture, which calls for modularization of resources by putting different kinds of annotation, different versions of the text and annotations, etc. in separate, linked documents. In addition, processing tools for the various XML recommendations (XPath, XPointer, XLink, etc.) are generally freely distributed, thus eliminating the need for costly and time-consuming tool development."<sup>17</sup>*

## III.2 Diverse Projekte

Angesichts der Tatsache, dass das XCES-Framework erst in der Version Beta 0.2 vorliegt, ließ sich im Internet kein namenhaftes Projekt ausfindig machen, das diesen Standard bereits nutzt. Der frühe Entwicklungsstand von XCES ist leider noch ein Grund dafür, warum einige neuere Projekte immer noch auf den älteren CES-

<sup>16</sup> Vgl. <http://www ldc.upenn.edu/exploration/expl2000/papers/ide/ide.pdf>, aufgerufen am : 16.09.2003

<sup>17</sup> Vgl. <http://www.cs.vassar.edu/~ide/papers/xces-lrec00.ps> (Conclusion), aufgerufen am: 16.09.2003

Standard oder die umständlichen TEI-Konventionen zurück greifen, wie das zum Beispiel beim Göteborg Spoken Language Corpus der Fall ist. Die haus eigene Beschreibungssprache Göteborg Transcription Standard (GTS) wurde durch eine XML-fähige Sprache ersetzt, die sich an die Richtlinien der TEI hält, trotz bekannter Nachteile und dem Fehlen der geeigneten Software.<sup>18</sup>

Ein auf CES aufbauendes Projekt ist *Preparatory Action for Linguistic Resources Organisation for Language Engineering* (PAROLE), einem EU-weitem Projekt mit dem Ziel, Korpora in allen Sprachen der EU-Länder anzubieten. Diese Korpora bestehen aus insgesamt 20 Millionen Wörtern aus Romanen und Zeitungsartikeln.<sup>19</sup>

Das *University of Maryland Parallel Corpus Project: Bible* verfolgt einen ähnlichen Ansatz. Es bietet einen multilingualen Korpus, bei dem als Textgrundlage die Bibel in derzeit 12 Sprachfassungen vorliegt. Auch diese Korpora halten sich an die CES-Richtlinien.<sup>20</sup>

Des Weiteren listen die Seiten des Corpus Encoding Standards acht weitere Projekte auf, die sich an dieses Framework halten. Namentlich genannt seien an dieser Stelle das französische Projekt *BAF : Corpus de bi-texte anglais-français*, sowie ein Korpus für gesprochene Sprache, der unter dem Projektnamen *MATE* entwickelt wird.<sup>21</sup>

Das Institut für Maschinelle Sprachverarbeitung der Universität Stuttgart arbeitet derzeit an einem Projekt namens TIGER Treebank, einem Korpus, bestehend aus 700.000 Token (40.000 Sätze) aus deutschen Zeitungsartikeln der Frankfurter Rundschau. Der annotierte Korpus liegt unter anderem auch in einer XML-Version vor. Das Format wird TIGER-XML-Format genannt, und basiert weder auf TEI noch auf XCES. Die hierarchische Struktur eines Satzes spiegelt sich dabei nicht in dem XML-Schema wieder. Statt dessen werden für jeden Satz alle Terminal- und Nicht-Terminal-Zeichen benannt und mit ID- und IDREF-Attributen verknüpft. So hat zum Beispiel der Satz "Die Tagung hat mehr Teilnehmer als je zuvor" folgende hierarchische und syntaktische Struktur:

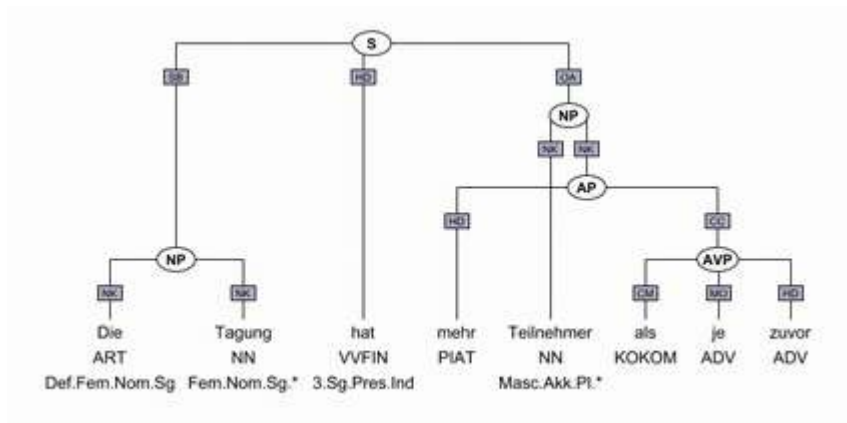
---

<sup>18</sup> weitere Informationen unter: [www.gslt.hum.gu.se/~leifg/gslt/doc/lingres.pdf](http://www.gslt.hum.gu.se/~leifg/gslt/doc/lingres.pdf)

<sup>19</sup> weitere Informationen unter: <http://www.dcs.shef.ac.uk/research/groups/nlp/funded/parole.html>

<sup>20</sup> weitere Informationen unter: <http://benjamin.umd.edu/parallel/bible.html>

<sup>21</sup> weitere Informationen unter: <http://www.cs.vassar.edu/CES/CES-P.html>



Innerhalb des Korpus wird der Text wie folgt annotiert:

```

<body>
<s id="s5">
  <graph root="s5_504">
    <terminals>
      <t id="s5_1" word="Die" pos="ART" morph="Def.Fem.Nom.Sg"/>
      <t id="s5_2" word="Tagung" pos="NN" morph="Fem.Nom.Sg.*"/>
      <t id="s5_3" word="hat" pos="VFIN" morph="3.Sg.Pres.Ind"/>
      <t id="s5_4" word="mehr" pos="PIAT" morph="--"/>
      <t id="s5_5" word="Teilnehmer" pos="NN" morph="Masc.Akk.Pl.*"/>
      <t id="s5_6" word="als" pos="KOKOM" morph="--"/>
      <t id="s5_7" word="je" pos="ADV" morph="--"/>
      <t id="s5_8" word="zuvor" pos="ADV" morph="--"/>
    </terminals>
    <nonterminals>
      <nt id="s5_500" cat="NP">
        <edge label="NK" idref="s5_1"/>
        <edge label="NK" idref="s5_2"/>
      </nt>
      <nt id="s5_501" cat="AVP">
        <edge label="CM" idref="s5_6"/>
        <edge label="MO" idref="s5_7"/>
        <edge label="HD" idref="s5_8"/>
      </nt>
      <nt id="s5_502" cat="AP">
        <edge label="HD" idref="s5_4"/>
        <edge label="CC" idref="s5_501"/>
      </nt>
      <nt id="s5_503" cat="NP">
        <edge label="NK" idref="s5_502"/>
        <edge label="NK" idref="s5_5"/>
      </nt>
      <nt id="s5_504" cat="S">
        <edge label="SB" idref="s5_500"/>
        <edge label="HD" idref="s5_3"/>
        <edge label="OA" idref="s5_503"/>
      </nt>
    </nonterminals>
  </graph>
</s>
</body>

```

23

Innerhalb des nonterminals-Tags wird die Struktur des Satzes konstruiert, indem auf bereits bestehende ID-Werte aus dem terminals-Tag verwiesen wird, als auch auf neu eingeführte ID-Werte bei den nonterminalen Symbolen.

<sup>22</sup> Vgl. <http://www.ims.uni-stuttgart.de/projekte/TIGER/TIGERSearch/doc/html/TigerXML.html> aufgerufen am : 24.09.2003

<sup>23</sup> Vgl. <http://www.ims.uni-stuttgart.de/projekte/TIGER/TIGERSearch/doc/html/TigerXML.html> aufgerufen am : 24.09.2003

## IV Vorteile und Möglichkeiten von XML

Unabhängig davon, welcher von den beiden oben beschriebenen Standards von den Sprachwissenschaftlern und Korpuslinguisten favorisiert wird und sich langfristig durchsetzen wird, wurde bei den Recherchen im Internet deutlich, dass eine einheitliche Standardisierung von annotierten Korpora die interdisziplinäre Zusammenarbeit zwischen den verschiedenen Forschungsgruppen und -projekten stark vereinfachen und beschleunigen würde. Ein auf XML basierendes Framework wäre für die Korpuslinguistik aus mehreren Gründen von Vorteil:

XML bietet die nötige Konsistenz, Flexibilität und Transparenz, um ein Annotierungsschema definieren und nachvollziehen zu können, und zwar nicht nur von Menschen, sondern insbesondere von Maschinen. Da sich das Annotierungsformat dank DTDs und XML Schemas selbst erklären kann, können Programme in der Lage sein, völlig fremde Datenstrukturen zu erfassen und zu visualisieren. Unzählige und völlig unterschiedliche Korpus-Projekte könnten so zu einer grossen Ressourcenquelle für die Wissenschaft zusammen geführt werden. XSL/XSLT bietet in diesem Zusammenhang eine standardisierte Sprache, um XML-Daten konvertieren zu können, sei es nach HTML zur Darstellung im Browser, nach PDF oder PostScript, oder in ein anderes Korpus-Format. Ein Korpus ließe sich damit also nicht nur als solcher begutachten und erforschen, sondern in einen direkten Zusammenhang mit anderen Korpora bringen.

Die in XCES vorgestellte Möglichkeit über die XPath-Syntax auf externe Zeichenketten verweisen zu können, erlaubt es, die Annotierung losgelöst von dem eigentlichen Text zu verwalten. Das bedeutet, zukünftig kann ein Text, der über ein XML-Schema lediglich in Kapitel, Abschnitte und Sätze etc. unterteilt ist, Grundlage für mehrere Korpora sein, in dem die eigentliche Annotierung in einer externen Datei ausgelagert ist und über Links auf die betreffenden Textstellen verweist, wie es das Beispiel auf Seite 16 zeigt. Der Anwender kann sich dadurch über ein geeignetes Programm nur den Text, den Text mit den gewünschten Annotierungen oder nur die Annotierungen anzeigen lassen, ohne das zusätzliche Datenquellen benötigt werden und ohne, dass das Programm die Struktur des Korpus wirklich "versteht". Im übrigen hängt die Darstellung des Korpus am Bildschirm oder als Printversion

lediglich von den Fähigkeiten des Programms ab, aber nicht mehr von dem Format, in dem der Korpus abgespeichert ist, sofern es sich an einen XML-Standard hält.

Die Zusammenführung von Wissens- und Datenquellen innerhalb der Korpusanalyse ließe sich natürlich auch über Standards realisieren, die nicht auf XML basieren. Auf Grund der Tatsache aber, dass sich XML als Universalsprache für fast alle Belange der elektronischen Datenverarbeitung eignet, könnte die Computerlinguistik zukünftig nicht nur von ihren eigenen Ressourcen und Programmen profitieren, sondern auch von Datenquellen aus anderen wissenschaftlichen Bereichen oder der Softwareindustrie. Denkbar wäre beispielsweise ein auf XML-basierendes Format, das gesprochene Sprache in Form von Ton bzw. Video mit dem entsprechenden Text verknüpft.

Generell ließe sich der Informationsaustausch zwischen völlig unterschiedlichen wissenschaftlichen Disziplinen dank des XML-Standards also deutlich vereinfachen.

## **V Fazit**

XML ist noch ein sehr junger Standard und als Nachkömmling von SGML eine sehr leistungsstarke und vielseitig einsetzbare Beschreibungssprache. Allmählich wächst das Bewusstsein dafür, welches Potential sich hinter offenen Standards und der Trennung von Informationen und deren Darstellung verbirgt. Auf absehbare Zeit bleibt es jedoch zumindest fraglich, in wie weit dieses Potential tatsächlich ausgeschöpft werden kann. Sowohl in der Softwareindustrie als auch unter den verschiedenen wissenschaftlichen Institutionen wird eine stärkere Zusammenarbeit über allen kommerziellen und elitären Interessen hinweg nötig sein, um Standards auf XML basierend etablieren zu können, von denen sowohl der Kunde als auch der Anbieter profitieren kann.

Für die Korpuslinguistik bedeutet das: gibt es erst mal ein Standardframework für linguistische Korpora, das den unterschiedlichen Anforderungen in diesem Bereich gerecht wird, steht der standardisierten Verknüpfung und Darstellung unterschiedlicher Wissensquellen und einer daraus resultierenden Bereicherung für die Korpuslinguistik nichts mehr im Wege.

## Quellenangaben

- Goldfarb, Prescod: *Das XML-Handbuch*. 2. Auflage. München 2000.
- Ide, Nancy: *The XML framework and its implications for corpus access and use* in Proceedings of Data Architectures and Software Support for Large Corpora pp. 28-32 Paris.  
<http://www.cs.vassar.edu/~ide/papers/xml-lrec00.ps>  
Stand: 2000, abgerufen am: 16.09.2003
- Ide, Nancy: *The XML framework and its implications for the development of natural language processing tools* in Proceedings of the Workshop on Using Toolsets and Architectures to Build NLP Systems Luxembourg.  
<http://www.cs.vassar.edu/~ide/papers/coling00-ws-final.ps>  
Stand: 2000, abgerufen am: 16.09.2003
- Ide, Nancy; Bonhomme, Patrice and Romary, Laurent: *XCES: An XML-based encoding standard for linguistic corpora* in Proceedings of the Second International Conference on Language Resources and Engineering (LREC 2000) pp. 825-830 Athen.  
<http://www.cs.vassar.edu/~ide/papers/xces-lrec00.ps>  
Stand: 2000, abgerufen am: 16.09.2003
- Ide, Nancy and Romary, Laurent: *XML support for annotated language resources* in Workshop on Web-Based Language Documentation and Description pp. 148-153 Philadelphia.  
<http://www ldc.upenn.edu/exploration/expl2000/papers/ide/ide.pdf>  
Stand: 2000, abgerufen am: 16.09.2003
- *Welcome to the TEI Website*.  
<http://www.tei-c.org/>  
Stand: 06.09.2003, abgerufen am: 15.09.2003
- *XCES version 0.1*.  
<http://www.cs.vassar.edu/XCES/>  
Stand: 03.05.2003, abgerufen am: 15.09.2003
- *Corpus Encoding Standard*.  
<http://www.cs.vassar.edu/CES/>  
Stand: 20.03.2000, abgerufen am: 15.09.2003
- *SELFHTML: XML/DTDs*.  
<http://selfhtml.teamone.de/xml/index.htm>  
Stand: 2001, abgerufen am: 15.09.2003
- *TEI in der Praxis*.  
<http://computerphilologie.uni-muenchen.de/praxis/teiprax.html>  
Stand: 09.05.1999, abgerufen am: 15.09.2003

- Rauch, Andreas: *DTD vs. XML Schema - Wachablösung in der XML Validierung*.  
[http://www.devtrain.de/artikel\\_445.aspx](http://www.devtrain.de/artikel_445.aspx)  
abgerufen am: 15.09.2003
- *The TIGER-XML treebank encoding format*  
<http://www.ims.uni-stuttgart.de/projekte/TIGER/TIGERSearch/doc/html/TigerXML.html>  
abgerufen am: 25.09.2003