

KONZEPTION UND UMSETZUNG EINER XML-  
BASIERTEN TERMINOLOGIEVERWALTUNG  
IN DER RUBICON GMBH

**D I P L O M A R B E I T**

IM FACHBEREICH INFORMATIK UND  
KOMMUNIKATIONSSYSTEME  
DER HOCHSCHULE MERSEBURG (FH)

vorgelegt von:  
Peggy Furch, KTM02

---

Erster Betreuer:	Herr Dr. Thomas Meinike
Zweiter Betreuer:	Herr Dipl.-Ing., Ing. Peter Kretschmann
Eingereicht am:	2006-12-14

## Inhalt

<b>1 Einleitung</b>	<b>1</b>
<b>2 Das Unternehmen</b>	<b>3</b>
2.1 <i>Problematik</i>	4
2.2 <i>Aufgabenstellung und Zielsetzung</i>	5
<b>3 Grundlagen der Terminologiearbeit</b>	<b>7</b>
3.1 <i>Grundbegriffe</i>	7
3.1.1 <i>Terminologie</i>	8
3.1.2 <i>Begriff und Benennung</i>	8
3.1.3 <i>Terminologielehre</i>	9
3.1.4 <i>Terminologiearbeit</i>	9
3.2 <i>Wirtschaftliche Bedeutung</i>	10
3.3 <i>Lösung des Terminologie-Problems</i>	12
3.3.1 <i>Normung durch Institute</i>	12
3.3.2 <i>Normung in Unternehmen</i>	13
<b>4 Systematisierung der Terminologie</b>	<b>14</b>
4.1 <i>Vorüberlegungen</i>	14
4.2 <i>Informationssammlung</i>	15
4.3 <i>Überarbeitung des Materials</i>	16
4.3.1 <i>Vereinheitlichen der Benennungen</i>	16
4.3.2 <i>Bearbeiten der Typenkennzeichnungen</i>	19
4.4 <i>Bereitstellung für den Benutzer</i>	20
4.4.1 <i>Übersicht der Maschinentypen</i>	20
4.4.2 <i>Handout zur Typenkennzeichnung</i>	22
4.4.3 <i>Terminologie-Anwendung</i>	23
<b>5 XML</b>	<b>24</b>
5.1 <i>Gründe für den Einsatz von XML</i>	24
5.1.1 <i>Strukturierte Daten</i>	24
5.1.2 <i>Trennung von Struktur und Layout</i>	25
5.1.3 <i>Flexible Sprachelemente</i>	25

5.1.4	<i>Universelles Austauschformat</i>	26
5.2	<i>Aufbau des XML-Dokuments</i>	26
5.2.1	<i>Prolog</i>	27
5.2.2	<i>Dokumentinstanz</i>	28
5.2.3	<i>Document Type Definition</i>	30
<b>6</b>	<b>Transformation des XML-Dokuments</b>	<b>34</b>
6.1	<i>Grundlagen der Extensible Stylesheet Language</i>	34
6.1.1	<i>XPath</i>	35
6.1.2	<i>XSLT</i>	38
6.2	<i>Realisierung</i>	40
6.2.1	<i>Aufbau des XHTML-Dokuments</i>	43
6.2.2	<i>Struktur des XSLT-Dokuments</i>	48
6.2.3	<i>Umsetzung weiterer Funktionen</i>	51
<b>7</b>	<b>AJAX</b>	<b>55</b>
7.1	<i>Grundlagen</i>	57
7.1.1	<i>Das Hypertext Transfer Protocol</i>	57
7.1.2	<i>Das XMLHttpRequest-Objekt</i>	59
7.1.3	<i>Senden einer HTTP-Anfrage</i>	60
7.2	<i>Document Object Model</i>	64
7.2.1	<i>Der Dokumentenbaum</i>	65
7.2.2	<i>Eigenschaften</i>	65
7.2.3	<i>Methoden</i>	66
7.2.4	<i>Selektieren einzelner Elemente</i>	67
7.3	<i>Realisierung</i>	68
7.3.1	<i>Aufbau der Seite</i>	68
7.3.2	<i>Verarbeitung der XML-Daten</i>	70
<b>8</b>	<b>Evaluation der beiden Varianten</b>	<b>82</b>
8.1	<i>Vergleich der Applikationen</i>	82
8.1.1	<i>Vor- und Nachteile von XSLT</i>	83
8.1.2	<i>Vor- und Nachteile von AJAX</i>	84
8.2	<i>Usability-Test</i>	85

8.2.1 Die Testaufgaben	86
8.2.2 Testauswertung und Korrektur	87
<b>9 Einsatz der Terminologie-Anwendung</b>	<b>89</b>
9.1 Benutzung	89
9.1.1 Navigation	90
9.1.2 Suchfunktion	90
9.1.3 Glossar	91
9.2 Terminologiepflege	92
9.2.1 XMLSpy Professional Edition Version 2007	92
9.2.2 XMLmind Standard Edition 3.4.0	95
9.2.3 XML Notepad 2007	97
9.2.4 Microsoft Excel	99
<b>10 Zusammenfassung</b>	<b>103</b>
<b>11 Literaturverzeichnis</b>	<b>105</b>
<b>12 Eidesstattliche Erklärung</b>	<b>109</b>
<b>13 Anhang</b>	<b>110</b>

## Anhangsverzeichnis

Anhang A	Übersicht der Maschinentypen
Anhang B	Richtlinie für die Schreibweise der Maschinentypenkennzeichnung
Anhang C	Usability-Test
Anhang D	Inhalt der CD

## Abbildungsverzeichnis

Abbildung 1	Folgen von Verständnisschwierigkeiten (Schulz 2003, S. 13)	11
Abbildung 2	Übersicht Maschinentypen	21
Abbildung 3	Schreibweise der Typenkennzeichnung	22
Abbildung 4	Dokumentenbaum terminologie.xml (In Anlehnung an: Shepherd 2002, S. 86).	31
Abbildung 5	Der XSLT-Prozessor (Ammelburger 2004, S. 332)	39
Abbildung 6	Einteilung der Webseite	41
Abbildung 7	Layout der Startseite	42
Abbildung 8	Übertragung ohne AJAX (Carl 2006, S. 3)	56
Abbildung 9	Übertragung mit AJAX (Carl 2006, S. 4)	56
Abbildung 10	Kommunikation zwischen Client und Server via HTTP (Gamperl 2006, S. 166)	57

## VII

Abbildung 11	Ausgabe der Suchergebnisse in der XSLT-Anwendung	87
Abbildung 12	Grid-Ansicht in XMLSpy	93
Abbildung 13	Einfügen von Elementen in XMLSpy	94
Abbildung 14	XMLmind	95
Abbildung 15	Einfügen von Elementen in XMLmind	96
Abbildung 16	XML Notepad	97
Abbildung 17	Suche nach Einträgen in XML Notepad	98
Abbildung 18	Erstellen einer XML-Liste in Microsoft Excel	100
Abbildung 19	XML-Liste in Microsoft Excel	100
Abbildung 20	Einfügen neuer Daten in eine XML-Liste	101

**Tabellenverzeichnis**

Tabelle 1	Überarbeitung der Benennungen	17
Tabelle 2	Achsen in XPath (Hauser 2006, S. 82)	37
Tabelle 3	Wertebereich der Statuscodes (Gamperl 2006, S. 168)	58
Tabelle 4	Methoden des XMLHttpRequest-Objekts (Gamperl 2006, S. 183)	61
Tabelle 5	Werte der Eigenschaft readyState (Gamperl 2006, S. 187)	63
Tabelle 6	Eigenschaften der DOM-Knoten (Gamperl 2006, S. 25 f.)	66
Tabelle 7	Methoden des node-Objekts (Gamperl 2006, S. 30)	67
Tabelle 8	Gezieltes Ansprechen von Elementen (Gamperl 2006, S. 44)	68

## Abkürzungsverzeichnis

<b>AJAX</b>	Asynchronous JavaScript and XML
<b>API</b>	Application Programming Interface
<b>CSS</b>	Cascading Style Sheets
<b>DOM</b>	Document Object Model
<b>DTD</b>	Document Type Definition
<b>HTML</b>	Hypertext Markup Language
<b>HTTP</b>	Hypertext Transfer Protocol
<b>PCDATA</b>	Parsed Character Data
<b>URL</b>	Uniform Resource Locator
<b>W3C</b>	World Wide Web Consortium
<b>XHTML</b>	Extensible Hypertext Markup Language
<b>XML</b>	Extensible Markup Language
<b>XPath</b>	XML Path Language
<b>XSL</b>	Extensible Stylesheet Language
<b>XSLT</b>	XSL Transformations

## 1 Einleitung

Eine konsistente Terminologie stellt die Grundlage für eine erfolgreiche Unternehmenskommunikation dar. Sie beeinflusst alle Unternehmensbereiche - von der Konstruktion über Produktion und Vertrieb bis hin zur Dokumentation. Vor allem in den technischen Industriezweigen, aber auch in den einzelnen Unternehmen selbst, haben sich individuelle Fachsprachen entwickelt. Neue Entwicklungen bringen und werden auch in Zukunft immer wieder neue Fachwörter mit sich bringen.

Werden diese Fachwörter im Unternehmen aber nicht einheitlich verwendet, kann es schnell zu Verständigungsproblemen bei der internen Kommunikation kommen. Aber auch die Erschließung internationaler Märkte bringt neue sprachliche Herausforderungen mit sich. Gibt es aber bereits in der Ausgangssprache Unstimmigkeiten in der Terminologie, wird sich dies auch auf die Übersetzungen und deren Qualität auswirken. Ein effizientes Terminologie-Management kann hier Abhilfe schaffen.

Diese Diplomarbeit soll einen möglichen Ansatz für die Einführung einer einheitlichen Terminologie aufzeigen. Im Rahmen dieser Diplomarbeit wurde eine Terminologienormung in der rubicon Gummitechnik und Maschinenbau GmbH durchgeführt, deren Ergebnisse in Form von Handouts und einer Terminologie-Anwendung umgesetzt wurden.

Einführend soll zunächst das Unternehmen kurz vorgestellt und die Notwendigkeit der Terminologiarbeit aufgezeigt werden. Schlussfol-

gernd daraus soll die Zielsetzung dieser Diplomarbeit formuliert werden.

In Kapitel 3 werden zum Einstieg in die Thematik die theoretischen Grundlagen der Terminologiearbeit erläutert. Das darauf folgende Kapitel beschäftigt sich mit der praktischen Umsetzung. Hier werden die Herangehensweise und die Durchführung der Systematisierung der Terminologie beschrieben.

Die darauf folgenden Kapitel beschäftigen sich mit der technischen Umsetzung einer Terminologie-Anwendung. In Kapitel 5 wird die Extensible Markup Language (XML) vorgestellt und ihr Einsatz zum Speichern der Terminologie beschrieben. Kapitel 6 und 7 stellen zwei Varianten vor, wie eine Webseite zum Ausgeben der gespeicherten Einträge realisiert werden kann. Die erste Variante soll mit XSL Transformations umgesetzt werden, die zweite mit AJAX. In dem darauf folgenden Kapitel werden diese Möglichkeiten verglichen und hinsichtlich ihrer Funktionalität bewertet.

Kapitel 9 beschäftigt sich mit dem Einsatz der Terminologie-Anwendung in der Praxis. Hier werden die Bedienung und die Funktionen der Anwendung erklärt. Außerdem beschäftigt sich das Kapitel mit der Pflege der Inhalte und zeigt verschiedene Möglichkeiten auf, wie das erstellte XML-Dokument editiert werden kann.

Die im Rahmen dieser Diplomarbeit erstellten Anwendungen befinden sich auf der beiliegenden CD.

## 2 Das Unternehmen

Die rubicon Gummitechnik und Maschinenbau GmbH (im Folgenden rubicon GmbH genannt) wurde im Oktober 1991 in Halle (Saale) gegründet. Das Betätigungsfeld des Unternehmens erstreckt sich von der Herstellung, über die Entwicklung und Konstruktion, bis hin zum Vertrieb von Anlagentechnik für die gummiverarbeitende Industrie.

Spezialisiert hat sich das Unternehmen dabei auf die Entwicklung und Herstellung von Extrusionstechnik. Die von der rubicon GmbH hergestellten Extruder sind Schneckenpressen, die dickflüssige Kautschukmassen gleichmäßig aus einer formgebenden Öffnung herauspressen. Einsatz finden diese Maschinen beispielsweise bei der Herstellung von Schläuchen, Profilen, Gummiwalzen und Kabeln. Demzufolge zählen zu den Abnehmern Hersteller technischer Gummiwaren, die Reifen- und die Kabelindustrie. (rubicon GmbH, Interne Materialien)

Im Mittelpunkt der Technologie- und Maschinenentwicklung steht dabei der Kunde. Aufgrund der engen Zusammenarbeit von Gummitechnologen und Konstrukteuren ist die rubicon GmbH in der Lage, auf die speziellen Wünsche und Bedingungen der Kunden zugeschnittene Maschinen zu produzieren. So kann dem Kunden auch durch Fertigung von Einzelmaschinen eine effiziente Produktion seiner Waren ermöglicht werden. ([www.rubicon-halle.de](http://www.rubicon-halle.de))

Zu den Produkten der rubicon GmbH gehören:

- Extruder zur Verarbeitung von Kautschukmischungen
- Extruder-Zahnradpumpen-Kombinationen

- Kontinuierliche Vulkanisationsanlagen
- Beschickungs- und Nachfolgetechnik
- Misch- und Laborwalzwerke
- Laborausrüstungen
- Sondermaschinen

Aber auch Industriemontagen und die Überholung von Gebrauchsmaschinen zählen zu den Leistungen der rubicon GmbH.

Aufgrund des ständig weiterentwickelten Fertigungsprogramms ist die rubicon GmbH sowohl am deutschen als auch am internationalen Markt erfolgreich. Der Großteil der Produktion wird ins Ausland verkauft. Exportverbindungen bestehen zu europäischen Ländern, wie z. B. Österreich, Niederlande, Polen und zu außereuropäischen Ländern wie Iran, Saudi Arabien, Indien, Malaysia und China. (rubicon GmbH, Interne Materialien)

## **2.1 Problematik**

Gemeinsam mit dem Unternehmen und seinen Entwicklungen ist auch der Umfang der verwendeten Fachwörter im Laufe der Jahre gewachsen. Doch nicht immer wurden von allen Mitarbeitern die gleichen Benennungen verwendet. Dies führt in einigen Bereichen zu großen Erschwernissen bei der täglichen Arbeit.

Ein großes Problem stellt sich beim Wiederauffinden von Dokumenten dar. Die Inhalte aller Dokumente bzgl. eines Kunden, z. B. Anfragen, Angebote und Auftragsbestätigungen, werden in einer Microsoft Access-Datenbank gespeichert. Der Benutzer hat über eine Suchfunktion die Möglichkeit, nach darin enthaltenen Zeichenketten zu

suchen. Dies kommt zum Beispiel vor, wenn er wissen möchte, ob für ein bestimmtes Produkt schon einmal ein Angebot erstellt wurde.

Die Datenbank liefert alle Datensätze, in denen der Suchbegriff exakt so, wie er eingegeben wurde, vorkommt. Aufgrund unterschiedlicher Benennungen, aber auch teilweise durch unterschiedliche Schreibweisen, wird der Benutzer dabei nicht alle vorhandenen Datensätze finden. Liefert die Suche nicht gleich das erwünschte Ergebnis, so muss er es mit anderen Benennungen versuchen. Dies kostet viel Geduld und auch Zeit.

Aber nicht nur bei den Benennungen fehlt die Konsistenz. Die hergestellten Maschinen besitzen Typenkennzeichnungen. Diese setzen sich aus einer Aneinanderreihung von Buchstaben und Zahlen zusammen. Dabei wurden bisher aber keine strengen Regeln für die Schreibweise erstellt. Teilweise werden sie mit Leerzeichen, teilweise ohne Leerzeichen geschrieben. Möchte der Benutzer Suchen anhand dieser Kennzeichnungen vornehmen, so ist er oft dazu gezwungen, mehrere Varianten durchzuprobieren, um so viele Ergebnisse wie möglich zu erhalten.

## **2.2 Aufgabenstellung und Zielsetzung**

Im Rahmen dieser Diplomarbeit werden die in der rubicon GmbH verwendeten Fachausdrücke in deutscher und englischer Sprache zusammengetragen, überarbeitet und vereinheitlicht. Zudem soll eine Systematisierung der Maschinentypenkennzeichnungen erfolgen.

Die dabei zusammengestellte Terminologie soll in Form von Handouts sowie in einer Web-Anwendung dargeboten werden, die von

jedem Mitarbeiter am eigenen PC abgerufen werden kann. Als Datenquelle soll ein XML-Dokument dienen.

Ziel der hier vorliegenden Arbeit ist es, die firmeninterne Terminologie zu normen und eine Terminologie-Anwendung zu entwickeln, die den Mitarbeitern des Unternehmens als hilfreiches Werkzeug für die tägliche Arbeit zur Verfügung steht. Diese wird ähnlich einem Online-Wörterbuch einsetzbar sein und ein schnelles Auffinden der richtigen Benennungen sowie deren Übersetzung ermöglichen. Damit soll eine Basis für die zukünftige Terminologearbeit geschaffen werden.

### **3 Grundlagen der Terminologiearbeit**

Große Fortschritte in den Bereichen Wissenschaft und Technik führten in den letzten Jahrzehnten dazu, dass die fachspezifische Kommunikation immer mehr an Bedeutung gewann. Mit der Erweiterung des menschlichen Wissens wuchs auch der Umfang der Fachwortbestände. Vielfach führte dies zu Verständigungsschwierigkeiten sowohl zwischen Laien als auch zwischen Experten. (Arntz, Picht, Mayer 2004, S. 1)

Weitere Schwierigkeiten treten bei der Erschließung des europäischen sowie des internationalen Marktes auf, die vielen Unternehmen neue Absatzmöglichkeiten für ihre Produkte und Dienstleistungen bietet. Die Unternehmen werden aufgrund der sprachlichen, kulturellen und gesellschaftlichen Vielfalt mit einer Vielzahl von Problemen konfrontiert. Ein wichtiger Aspekt ist dabei die Verständigung. Um erfolgreich und konkurrenzfähig zu sein, ist es notwendig spezifische Lösungen zu finden, um Sprachbarrieren zu überwinden. ([www.tekom.de](http://www.tekom.de))

Die Voraussetzung für eine Lösung dieser Verständigungsschwierigkeiten ist eine erfolgreiche Terminologiearbeit.

#### **3.1 Grundbegriffe**

Um eine erfolgreiche Terminologiearbeit in der Praxis durchführen zu können, ist die Kenntnis der wichtigsten Grundbegriffe notwendig.

### 3.1.1 Terminologie

Das Wort Terminologie ist eine Zusammensetzung aus terminus (lat. „festgelegte Bezeichnung“) und logos (griech. „Wort“ oder „Lehre“) und bezeichnet damit die „Lehre von den Bezeichnungen“. Auf dem Gebiet der Dokumentation und der Übersetzung bezeichnet man mit Terminologie oft eine Auflistung von Wörtern, Begriffen und Benennungen, die auf einem bestimmten Gebiet verwendet werden. (Schulz 2003, S. 21)

### 3.1.2 Begriff und Benennung

Die Kenntnis über die Bedeutung der Begriffe „Begriff“ und „Benennung“ ist die Voraussetzung für die Terminologiearbeit. Oft kommt es zu Problemen, da diese häufig verwechselt werden. (Schulz 2003, S. 23)

Der „Begriff“ bezeichnet eine

Denkeinheit, die aus einer Menge von Gegenständen unter Ermittlung der diesen Gegenständen gemeinsamen Eigenschaften mittels Abstraktion gebildet wird. (DIN 2342 Teil 1 1992, S. 1)

Demzufolge existieren Begriffe nur in gedanklicher Form. Sie stellen nichts Sprachliches dar, sondern existieren nur im Kopf (Schulz 2003, S. 23). Dadurch sind sie auch nicht an bestimmte Sprachen gebunden.

Die „Benennung“ hat zur Aufgabe, den „Begriff“ sprachlich zu bezeichnen. Sie kann aus einem oder mehreren Wörtern bestehen und

verleiht einem Gegenstand einen Namen. (DIN 2342 Teil 1 1992, S. 2)

Aufgabe der Terminologiearbeit ist es also, Benennungen zu sammeln und zu ordnen.

### **3.1.3 Terminologielehre**

Die Terminologielehre bezeichnet die Wissenschaft von Begriffen und ihren Benennungen auf dem Gebiet der Fachsprachen. Sie wird in die allgemeine und die spezielle Terminologielehre unterteilt.

Die allgemeine Terminologielehre ist fach- oder sprachübergreifend, die spezielle Terminologielehre umfasst nur ein Fachgebiet in einer oder mehreren Sprachen. (DIN 2342 Teil 1 1992, S. 4)

### **3.1.4 Terminologiearbeit**

Nach DIN 2342 Teil 1 „Begriffe der Terminologie - Grundbegriffe“ ist die Terminologiearbeit wie folgt definiert:

Auf der Terminologielehre aufbauende Erarbeitung, Bearbeitung oder Verarbeitung, Darstellung oder Verbreitung von Terminologie sowie Einarbeitung von Terminologie in Texte.

Bezieht sich die Terminologiearbeit nur auf eine Sprache spricht man von terminologischer Einzelnormung. Dem gegenüber steht die übersetzungsorientierte Terminologiearbeit, die sich mit mehreren Sprachen beschäftigt. (DIN 2342 Teil 1 1992, S. 4)

### **3.2 Wirtschaftliche Bedeutung**

Mangelndes Terminologie-Management führt nicht nur bei der zwei- oder mehrsprachigen Kommunikation zu Problemen. Im Alltag eines Unternehmens kann es bereits zu Erschwernissen kommen, wenn Mitarbeiter verschiedene „Dialekte“ einer speziellen Fachsprache sprechen. So sind Verständnisprobleme schon in der Muttersprache keine Seltenheit. Ein Beispiel dafür ist die Kommunikation zwischen verschiedenen Abteilungen. So kann es vorkommen, dass die Entwicklung andere Benennungen für ein bestimmtes Bauteil benutzt als der Vertrieb. Folgen dieser Unterschiede können Missverständnisse, aber auch ein erhöhter Zeitaufwand durch notwendige Rückfragen sein (Schulz 2003, S. 8).

Abbildung 1 zeigt die Folgen auf, die aufgrund von Verständnisschwierigkeiten auftreten können.

**Typologie der Folgen von Verständnisschwierigkeiten****Rückfragen - Zeitverlust**

- innerhalb einer Abteilung
- zwischen Abteilungen oder Unternehmensbereichen
- bei der Aus- und Weiterbildung
- zwischen externen und dem Unternehmen

**Fehlentscheidungen - materieller Schaden**

- Fehlgebrauch, Unfall
- Falsches Teil oder Ersatzteil wird bestellt oder geliefert (plus Rücklieferung und Nachlieferung)
- Falsche Prozedur wird angewendet
- Anfrage wird falsch interpretiert
- Falsche Auskunft wird erteilt
- Fehler in der Produktion
- Fehler in der Lagerverwaltung

**Frustration - immaterieller Schaden**

- Verständnisstörungen innerhalb des Unternehmens
- Verminderte Lerneffizienz in der Ausbildung
- Wissensverluste
- Anfrager verliert Zeit durch Weiterverbinden
- Anfrager erhält negatives Bild der Unternehmenskompetenz
- Anfrager verliert Kaufinteresse

Abb. 1: Folgen von Verständnisschwierigkeiten

Solch eine Ineffizienz beim Sprachgebrauch wirkt sich selbstverständlich auch auf Übersetzungen aus. Vor allem, wenn Übersetzungen durch verschiedene Personen gemacht werden, sei es durch eigene Mitarbeiter oder durch Übersetzungsbüros. Ohne Vorgaben hinsichtlich einer eigenen Terminologie wird jede Übersetzung ein wenig anders aussehen. Versucht man nun verschiedene Übersetzungen zusammenzufügen, „kommt wild gemixte Terminologie im Dokument dabei heraus.“ (Schulz 2003, S. 15 f.)

### **3.3 Lösung des Terminologie-Problems**

Um den genannten Problemen zu entgehen, ist eine Terminologienormung unumgänglich. Diese kann durch Institute, aber auch im Unternehmen selbst erfolgen.

#### **3.3.1 Normung durch Institute**

In den meisten Ländern beschäftigen sich Normungsinstitutionen mit dieser Problematik. Dabei steht die Sicherung der eindeutigen Verständigung im Vordergrund. (Arntz, Picht, Mayer 2004, S. 135)

In den verschiedenen Ländern sind auf nationaler Ebene zentrale Institutionen für die Normungsarbeit zuständig. Beispiele dafür sind das Deutsche Institut für Normung e. V. (DIN), die Association Française de Normalisation (AFNOR) in Frankreich und das American National Standards Institute (ANSI) in den USA. Die Arbeit dieser Institute wird auf internationaler Ebene von der ISO und der IEC koordiniert. (Arntz, Picht, Mayer 2004, S. 141)

Beim Deutschen Institut für Normung ist der Normenausschuss „Terminologie“ (NAT) für die Grundsatzfragen auf dem Gebiet der Terminologie zuständig. Für die fachspezifische Terminologie sind die Normenausschüsse der jeweiligen Fachgebiete zuständig.

### **3.3.2 Normung in Unternehmen**

Terminologienormung kann auch in Unternehmen stattfinden. Und zwar durch die Erstellung von Richtlinien, die den *firmenrelevanten* Fachwortschatz festlegen. Diese Richtlinien haben den Vorteil, dass sie auf die Bedürfnisse und Anforderungen des Unternehmens zugeschnitten sind. Sie widersprechen nicht den Empfehlungen der DIN-Normen, sondern führen diese weiter. (Arntz, Picht, Mayer 2004, S. 145)

Im Rahmen dieser Diplomarbeit wurde eine firmeninterne Normung durchgeführt.

## 4 Systematisierung der Terminologie

Um einen nutzbaren Terminologie-Bestand schaffen zu können, ist es notwendig, die vorhandene Terminologie zu sammeln, zu überarbeiten und letztendlich zu normen. Dazu ist eine Reihe von Arbeitsschritten notwendig, die in diesem Kapitel anhand der durchgeführten Terminologiarbeit in der rubicon GmbH erläutert werden soll.

### 4.1 Vorüberlegungen

Bevor die eigentliche Terminologiarbeit beginnt, müssen einige grundsätzliche Festlegungen getroffen werden. Diese sollen die Grundregeln für die Erfassung der Terminologie bilden und somit die nachfolgende Arbeit erleichtern. (Schulz 2003, S. 49)

Zunächst ist es wichtig, sich klar zu machen, was genau gesammelt werden muss. Nur so kann von Anfang an effektiv gearbeitet werden. Die Erfassung der Terminologie soll zweisprachig erfolgen, in der Ausgangssprache (Deutsch) und einer Zielsprache (Englisch). Gleichzeitig werden die Typenkennzeichnungen zusammengetragen und vereinheitlicht.

In der Terminologie-Anwendung soll später eine Sortierung nach Sachgebieten möglich sein. Folgende Sachgebiete wurden dafür festgelegt:

- Abzugsbänder
- Ersatzteile
- Extruder

- Infrarot
- Kühlanlagen
- Linien
- Mikrowelle
- Salzbad
- Walzwerke
- Werkzeuge
- Wickler
- Zahnradpumpen

In der XML-Datei werden den Benennungen über Attribute die entsprechenden Sachgebiete zugeordnet. Um die zusammengetragene Terminologie später nicht erneut sortieren zu müssen, wird die Auflistung entsprechend der Sachgebiete erfolgen.

Zusätzlich zu den deutschen und englischen Benennungen wird es eine Möglichkeit geben, veraltete bzw. alternative deutsche Benennungen einzufügen. Dadurch hat der Benutzer die Möglichkeit, ein Suchergebnis zu erzielen, auch wenn er sich eine andere Benennung eingeprägt hat. Dieser Eintrag soll optional sein, d. h. er ist nicht zwingend erforderlich.

## **4.2 Informationssammlung**

Anhand der getroffenen Vorüberlegungen kann mit der Sammlung vorhandener Informationen begonnen werden. Die Sammlung erfolgte auf der Basis bereits erstellter Angebote. Dazu wurden alle Angebote der Jahre 2004 bis 2006 gesichtet, sowohl in deutscher als auch in englischer Sprache.

Beim Lesen der Angebote wurden die gefundenen Fachwörter, ihre jeweiligen Übersetzungen und Typenkennzeichnungen zusammengetragen und entsprechend ihrer Sachgebiete in Tabellen zu terminologischen Datensätzen zusammengestellt. (Arntz, Picht, Mayer 2004, S. 222)

### **4.3 Überarbeitung des Materials**

Im Anschluss an die Informationssammlung konnte mit der Überarbeitung des gesammelten Materials begonnen werden. Dazu waren zunächst das Vereinheitlichen der Einträge sowie das Aussortieren doppelter Benennungen notwendig.

#### **4.3.1 Vereinheitlichen der Benennungen**

Die Analyse des gesammelten Materials zeigte, dass in einigen Fällen mehrere Benennungen für ein und denselben Begriff verwendet werden. Andersherum wurden aber auch Benennungen gefunden, die für unterschiedliche Begriffe eingesetzt werden. Einige Beispiele solcher Benennungen und ihre Überarbeitung zeigt Tabelle 1.

Gefundene Benennungen	Überarbeitete Benennungen
Abzugsband Abzugsförderband	Abzugsförderband
Bandabzug Caterpillar Raupenabzug	Bandabzug
Geradeaus-Extrusionswerkzeug Geradeauswerkzeug Geradeaus-Werkzeug	Geradeaus-Werkzeug
Kühlwanne	Kühlwanne mit Sprühkühlung Kühlwanne, doppelwandig Kühlwanne mit Tauchrollen
Nasstrennmittelauftragsvorrichtung Trennmittelauftragsvorrichtung	Trennmittelauftragsvorrichtung
Querwerkzeug Quer-Werkzeug	Quer-Werkzeug
Umlenkwerkzeug	Umlenk-Werkzeug
Walzwerk	Mischwalzwerk Laborwalzwerk
rubicon Zahnradpumpe Zahnradpumpe	rubicon - Zahnradpumpe

Tab. 1: Überarbeitung der Benennungen

Bei den deutschen Benennungen war es von besonderer Bedeutung, neben einer einheitlichen Benennung auch eine einheitliche Schreibweise zu finden. Vor allem bei zusammengesetzten Wörtern spielte dies eine große Rolle. Hier wurden Regeln festgelegt, die genau vorschreiben, ob das Wort mit oder ohne Bindestrich sowie mit oder ohne Leerzeichen geschrieben werden soll. Die exakte Schreibweise ist

wichtig, damit diese Begriffe später in der Access-Datenbank wieder gefunden werden.

Das Zusammentragen der Standardextruder brachte beispielsweise folgende fünf Varianten:

1. rubicon Standardextruder
2. rubicon- Standardextruder
3. rubicon– Standardextruder
4. rubicon-Standardextruder
5. rubicon - Standardextruder

Als einheitliche Schreibweise wurde hier *rubicon - Standardextruder* festgelegt.

Bei den englischen Benennungen stellte sich ein ganz anderes Problem heraus. Hier stand nicht das Festsetzen der richtigen Schreibweise im Vordergrund, sondern das Finden der entsprechenden Benennung. Schlägt man einen deutschen Begriff im Wörterbuch nach, findet man meist nicht nur eine äquivalente Benennung. Dadurch, dass bisher keine einheitliche Terminologie definiert wurde, konnte jeder Mitarbeiter frei entscheiden, welche dieser Benennungen er benutzen möchte. Bei der Überarbeitung der gesammelten Materialien war es deshalb besonders wichtig, die englische Benennung zu finden, die der deutschen am ehesten entspricht. Dies wurde durch Recherchen und Vergleiche mit Wörterbüchern und Fachwörterbüchern erreicht. Durch die Vereinheitlichung der Benennungen soll eine terminologische Konsistenz erzielt werden.

### 4.3.2 Bearbeiten der Typenkennzeichnungen

Die zusammengetragenen Typenkennzeichnungen wurden meist durchgängig in einheitlicher Form genutzt. Probleme traten hier vorwiegend durch unterschiedlich gesetzte Leerzeichen und Bindestriche auf, wie das folgende Beispiel zeigt:

EEK 45.14 S - 11/70

EEK 45.14 S – 11/70

EEK 45.14 S-11/70

Die zweite Schreibweise mit dem Gedankenstrich kommt bei der Erstellung der Angebote in Microsoft Word zustande. Hier ändert die Autokorrektur während der Eingabe den Bindestrich in einen Gedankenstrich. Um dies in Zukunft zu vermeiden, sollte an allen Arbeitsplätzen diese Autokorrektur ausgeschaltet werden. Über das Menü *Extras/AutoKorrektur-Optionen* wird das Fenster AutoKorrektur geöffnet. Unter der Registerkarte „*AutoFormat während der Eingabe*“ kann nun das Kontrollkästchen „*Bindestriche durch Geviertstrich*“ deaktiviert werden.

Ein weiteres Problem, das sich aufzeigte, war die gleiche Typenkennzeichnung für verschiedene Begriffe. Folgende Benennungen liefen beispielsweise unter der Kennzeichnung *KW 6*:

1. Kühlstrecke (LCM)
2. Kühlwanne mit Sprühkühlung
3. Kühlwanne, doppelwandig
4. Kühlwanne mit Tauchrollen

Um auch hier ein schnellstmögliches Wiederfinden in der Datenbank zu ermöglichen, wurden die Kennzeichnungen wie folgt geändert:

Kühlstrecke (LCM)	KW-LCM 6
Kühlwanne mit Sprühkühlung	KW 6/SP
Kühlwanne, doppelwandig	KW 6/DW
Kühlwanne mit Tauchrollen	KW 6/T

#### **4.4 Bereitstellung für den Benutzer**

Die überarbeitete Terminologie soll nun allen Mitarbeitern im Unternehmen zur Verfügung gestellt werden. Dazu wurden drei Darreichungsformen gewählt, die je nach Arbeitsziel verwendet werden können.

##### **4.4.1 Übersicht der Maschinentypen**

Zunächst wurden alle gesammelten und vereinheitlichten Maschinentypen zu einer Übersicht zusammengetragen, die einen Überblick über das Fertigungssortiment geben soll.

Zusätzlich zu den Benennungen und Typenkennzeichnungen enthält diese Liste auch die Nummer des letzten Angebots. Dies ermöglicht dem Benutzer, bei Bedarf schnell auf ein existierendes Angebot zugreifen zu können, um nähere Informationen dazu zu bekommen.

Die Übersicht ist nach den einzelnen Sachgebieten unterteilt. Durch eine fortlaufende Nummerierung sowie ein Inhaltsverzeichnis ist der Nutzer in der Lage, gesuchte Begriffe schnell zu finden. Neue Benennungen bzw. Kennzeichnungen werden durch die Anmerkung

*neu!* am Zeilenbeginn gekennzeichnet. Dadurch wird der Benutzer nochmals darauf aufmerksam gemacht, dass hier Änderungen stattgefunden haben. Wie Abbildung 2 zeigt, werden die alten Bezeichnungen ebenfalls notiert, um die Begriffe auch in den alten Dokumenten auffinden zu können. Die komplette Übersicht ist in Anhang A dargestellt.

<b>5 Mikrowellen-Anlagen</b>			
<i>neu!</i>	Mikrowellentunnel	RC-MW 12/6 alt: RC-MW 12 kW	A065237
<i>neu!</i>	Mikrowellentunnel	RC-MW 18/6 alt: RC-MW 12 kW	A054959
<i>neu!</i>	Mikrowellentunnel	RC-MW 24/7 alt: RC-MW 12 kW	A054801
	Heißlufttunnel	RC-HLT 12 EL	A065237
	<i>Heißlufttunnel</i>	<i>RC-HLT 12 GAS</i>	<i>kein Angebot</i>
		...	

Abb. 2: Übersicht Maschinentypen

Die Daten liegen als Word-Dokument vor, sollen jedoch als PDF-Datei den Mitarbeitern zugänglich gemacht werden, damit nicht jeder das Dokument editieren kann und sich somit nicht wieder Ungeheimheiten einschleichen können. Ergänzungen und Änderungen sollte möglichst nur eine Person vornehmen.

#### 4.4.2 Handout zur Typenkennzeichnung

Zusätzlich zu der Übersicht wurde ein Handout entwickelt, das den Mitarbeitern dabei helfen soll, die richtige Maschinentypenkennzeichnung zu erstellen.

Das Handout legt die Regeln für die Schreibweise fest. Die exakte Schreibweise ist wichtig, um in der Access-Datenbank anhand der Kennzeichnung nach Datensätzen suchen zu können. Dabei ist vor allem die Getrennt- oder Zusammenschreibung zu beachten. Zum Verdeutlichen der Leerzeichen innerhalb einer Kennzeichnung wurde das Symbol „□“ verwendet.

Zusätzlich zu den Informationen über die Schreibweise enthält das Handout auch Erklärungen dazu, aus welchen Werten sich die Kennzeichnungen zusammensetzen. Wie Abbildung 3 zeigt, erfolgt das zum besseren Verständnis immer an einem Beispiel.

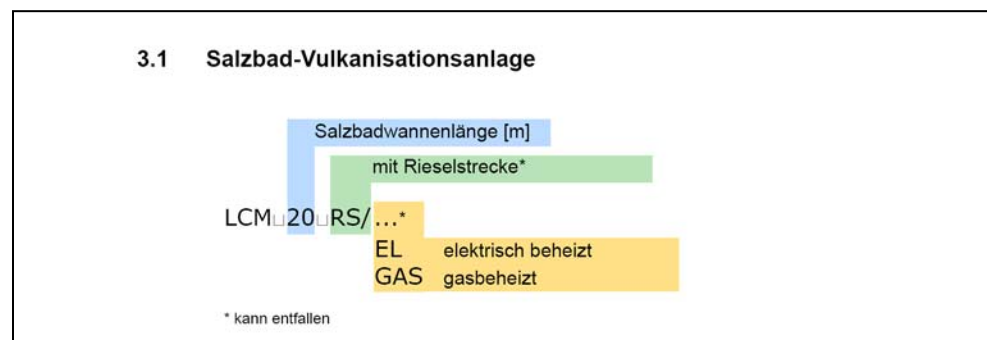


Abb. 3: Schreibweise der Typenkennzeichnung

Die veränderlichen Bestandteile werden durch eine farbige Kennzeichnung hervorgehoben, die gleichzeitig angibt, welcher Wert hier eingefügt wird. Das komplette Handout ist in Anhang B zu finden.

#### 4.4.3 Terminologie-Anwendung

Mit einer Terminologie-Anwendung soll den Mitarbeitern ein nützliches Werkzeug für die Übersetzung von Fachtexten gegeben werden. Die Anwendung soll im Intranet zur Verfügung gestellt werden, so dass alle Mitarbeiter Zugriff auf die Daten haben.

Die Datenbank wird für den Nutzer eine Art Wörterbuch darstellen, in dem er sowohl deutsche als auch englische Benennungen nachschlagen kann und die jeweilige Entsprechung der anderen Sprache angezeigt bekommt. Dabei werden verschiedene Vorgehensweisen möglich sein. Der Zugriff auf die Daten kann anhand ihres Sachgebietes, über ein Glossar und über eine Freitextsuche erfolgen.

Die Terminologie-Daten werden mit Hilfe der *Extensible Markup Language (XML)* erfasst und gespeichert. Als Benutzeroberfläche, über die die Daten abgerufen werden können, soll eine Web-Anwendung dienen. Diese kann mit ganz verschiedenen Technologien erzeugt werden. Für die Umsetzung der im Rahmen dieser Diplomarbeit erstellten Terminologie-Anwendung wurden deshalb zwei Möglichkeiten gewählt: Zum einen die Transformation der XML-Daten mittels der *Extensible Stylesheet Language for Transformations (XSLT)* und zum anderen über eine *Asynchronous Javascript and XML (AJAX)*-Anwendung.

Die einzelnen Technologien werden in den folgenden Kapiteln erklärt.

## **5 XML**

Die Abkürzung XML steht für Extensible Markup Language (erweiterbare Auszeichnungssprache). Im Jahr 1996 wurde XML konzipiert und zwei Jahre später zum W3C-Standard erhoben. (Shepherd 2002, S. 39)

XML wurde aus der Standard Generalized Markup Language (SGML) entwickelt. Das Ziel dabei war, die Funktionen von SGML speziell für den Einsatz im Internet zu optimieren. Im Vergleich zu SGML ist XML deutlich strenger, knapper und kompakter. (Ammelburger 2004, S. 53)

### **5.1 Gründe für den Einsatz von XML**

XML besitzt Funktionen und Fähigkeiten, die viele Möglichkeiten beim Strukturieren und Speichern von Daten eröffnen. Dieser Abschnitt soll die Eigenschaften von XML und die Vorteile beim Einsatz zur Speicherung der Terminologie zeigen.

#### **5.1.1 Strukturierte Daten**

Ein XML-Dokument besteht aus Daten, die in strukturierter Form vorliegen. Dies bietet den Vorteil, dass sich Daten aus vorhandenen Datenbanken leicht in XML verarbeiten lassen. Aber auch andere Anwendungen können XML-Dokumente nutzen. Dazu benötigen die Anwendungen eine Software-Komponente, den so genannten Prozessor. Dieser liest die Dokumente ein, analysiert die Struktur und gibt die Inhalte an die Anwendung weiter, die dann die weitere Ver-

arbeitung bestimmt. Ein Beispiel für solch eine Anwendung ist der Webbrowser. (Jänecke 2003, S. 407 f.)

### **5.1.2 Trennung von Struktur und Layout**

Wie bereits beschrieben beschränkt sich XML allein auf die Beschreibung der Struktur von Dokumenten. Es erfolgt keine Festlegung, wie diese Dokumente später dargestellt werden sollen.

Das Layout kann bei der Ausgabe der Daten ganz individuell bestimmt werden. Dazu werden spezielle Sprachen verwendet, mit denen Stylesheets gestaltet werden können. Beispiele für solche Sprachen sind die Extensible Stylesheet Language (XSL) oder die Cascading Style Sheets (CSS). (Jänecke 2003, S. 407)

### **5.1.3 Flexible Sprachelemente**

XML ist eine Metasprache. Das bedeutet, es handelt sich um eine allgemeine Sprache, mit der sich Untersprachen definieren lassen. (Hauser 2006, S. 9) Das bietet den Vorteil, dass sich aus XML Sprachen erzeugen lassen, die den Anforderungen spezifischer Einsatzgebiete angepasst sind (Jänecke 2003, S. 406).

Im Gegensatz zu anderen Auszeichnungssprachen, wie zum Beispiel HTML, sind in XML keine Elemente vordefiniert. Elementtypen können individuell erstellt werden und ebenfalls durch individuelle Attribute ergänzt werden. (Jänecke 2003, S. 406) Dadurch ist XML extrem flexibel und kann verschiedenen Verwendungszwecken genau angepasst werden. Anstelle einer vorgegebenen, begrenzten Anzahl von Elementen, die viele verschiedene Zwecke erfüllen müssen, lassen sich unbegrenzt eigene Elemente definieren. Diese müssen na-

türlich allgemeinen Regeln entsprechen, können aber sehr flexibel an die jeweiligen Ansprüche angepasst werden (Harold 2004, S. 31-33).

#### **5.1.4 Universelles Austauschformat**

Eine der wichtigsten Eigenschaften von XML ist die Plattformunabhängigkeit. Dadurch muss keine bestimmte Software vorhanden sein, um die Daten verarbeiten zu können. (Ammelburger 2004, S. 58)

In der Praxis bietet das viele Vorteile. XML kann als universelles und flexibles Daten- bzw. Dokumenten-Austauschformat verwendet werden. Man hat die Möglichkeit, beliebige Dokumentenarten in XML zu definieren, die dann zwischen verschiedenen Anwendungen ausgetauscht werden können. (Jänecke 2003, S. 408)

#### **5.2 Aufbau des XML-Dokuments**

Ein XML-Dokument weist eine logische Struktur auf. Es besteht aus zwei Teilen - dem Prolog und dem eigentlichen Inhalt, der Dokumentinstanz. (Born 2003, S. 766)

XML-Dokumente müssen immer *wohlgeformt* sein. Dies sind sie dann, wenn sie allen XML-Regeln für die Elemente und Attribute entsprechen. Zusätzlich können sie gegen eine DTD (Document Type Definition) bzw. ein Schema validiert werden. Das Dokument ist dann valide, wenn es eine korrekte Struktur aufweist und nur die in dieser Struktur erlaubten Elemente und Attribute verwendet. (Hauser 2006, S. 25)

### 5.2.1 Prolog

Am Anfang eines XML-Dokuments steht immer der Prolog. Dieser beginnt mit der XML-Deklaration.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

Anhand der Deklaration werden dem Programm, das das Dokument abarbeitet, wichtige Informationen übermittelt. Deshalb spricht man oft auch von Processing Instructions. (Hauser 2006, S. 11)

Die XML-Deklaration gibt Aufschluss über die verwendete XML-Version. Das Dokument entspricht der Version 1.0 der XML-Spezifikation. Durch die Angabe wird sichergestellt, dass das Dokument richtig gelesen wird. (Stein 2002, S. 11 f.)

Als zweite Information wird der verwendete Zeichensatz angegeben. Dieser gibt Aufschluss darüber, welche Schrift- und Sonderzeichen eingesetzt werden können. Der Zeichensatz ISO-8859-1 kodiert 256 Zeichen. (Hauser 2006, S. 12)

Nach der XML-Deklaration folgt die Dokumenttyp-Deklaration.

```
<!DOCTYPE terminologie SYSTEM "terminologie.dtd">
```

Diese Deklaration verweist auf eine externe Document Type Definition (DTD), anhand der der Parser die Dokumentinstanz überprüfen kann. (Born 2003, S. 771)

Die Dokumenttyp-Deklaration beginnt mit `<!DOCTYPE`. Danach folgt das Wurzelement `terminologie`. Das Schlüsselwort `SYSTEM` verweist auf eine externe DTD, deren Adresse in Anführungszeichen angegeben wird. (Hauser 2006, S. 31)

Bei der externen DTD handelt es sich um eine separate Datei. Sie besteht nur aus DTD-Anweisungen und wird mit der XML-Datei verknüpft (Stein 2002, S. 52). Der Inhalt der DTD wird in Abschnitt 5.2.3 beschrieben.

Es gibt auch noch die Möglichkeit, die DTD direkt in das XML-Dokument einzubinden. Dann spricht man von einer internen DTD. Diese steht direkt unter der XML-Deklaration im Dokument. Der Nachteil dabei ist, dass diese DTD nicht für andere Dokumente genutzt werden kann. (Stein 2002, S. 52) Außerdem kann es gerade bei umfangreicheren Regeln schnell unübersichtlich werden.

### **5.2.2 Dokumentinstanz**

Im Anschluss an den Prolog folgt nun der eigentliche Inhalt des XML-Dokuments - die Dokumentinstanz. Dieser Inhalt wird in *Elementen* abgelegt. (Born 2003, S. 776)

Die Elemente bestehen aus Markup-Tags und dem Elementinhalt, der zwischen dem Start- und dem Endtag steht. Die Dokumentinstanz wird durch das Wurzelement `terminologie` eingeleitet. Dieses umschließt alle weiteren Elemente. (Stein 2002, S. 12-14)

```
<terminologie>
  <term typ="werkzeug">
    <deutsch>Geradeaus-Werkzeug</deutsch>
    <englisch>straight extruder head</englisch>
    <alt>Geradeauswerkzeug</alt>
  </term>

  ...

</terminologie>
```

Die Strukturierung der Informationen erfolgt durch die Verschachtelung der Elemente. Dabei muss die Struktur mit den Deklarationen in der DTD übereinstimmen. (Born 2003, S. 776 f.)

Das Element `term` enthält drei Kindelemente - `deutsch`, `englisch` und `alt`. Es wird deshalb auch als Elternelement bezeichnet.

Mit Hilfe von *Attributen* können Elemente näher beschrieben werden. Dafür wird im Start-Tag des Elements der Attributname mit dem dazugehörigen Attributwert aufgeführt. (Born 2003, S. 778 f.) Dem Element `term` wurde auf diese Weise das Attribut `typ` mit dem Wert `werkzeug` zugewiesen.

Die Verwendung von Attributen bietet später die Möglichkeit, das XML-Dokument nach Elementen zu durchsuchen, die ein bestimmtes Attribut, z. B. `werkzeug`, haben.

### 5.2.3 Document Type Definition

Die Document Type Definition (DTD) legt die Struktur des XML-Dokuments fest. Sie stellt eine Sammlung von Regeln dar, an die sich das XML-Dokument halten muss. (Ammelburger 2004, S. 29 f.) Mit ihr werden sowohl Elemente, Attribute, Entitäten und Notationen als auch die Beziehungen zwischen ihnen festgelegt. Die DTD zeigt sozusagen die eigentliche Struktur des Dokumentes, unabhängig von den Daten. (Harold 2004, S. 219 f.)

Ein XML-Dokument ist ohne DTD nicht gültig. Sie wird gebraucht, um das Dokument zu validieren, d. h., auf seine Konformität zu überprüfen. (Pomaska 2005, S. 59)

In der Datei `terminologie.dtd` werden nun die Elemente und Attribute, die in dem XML-Dokument verwendet werden, sowie deren Inhaltsmodelle deklariert. Durch die *Elementtyp-Deklaration* wird auch gleichzeitig die Hierarchie der Elemente festgelegt. So gibt die DTD zum Beispiel Auskunft darüber, ob Elemente untergeordnete Kind-Elemente haben. (Jänecke 2003, S. 416)

Um die Elemente richtig festlegen zu können, ist es sinnvoll, sich das XML-Dokument zunächst als Baumstruktur abzubilden. (Abbildung 4)

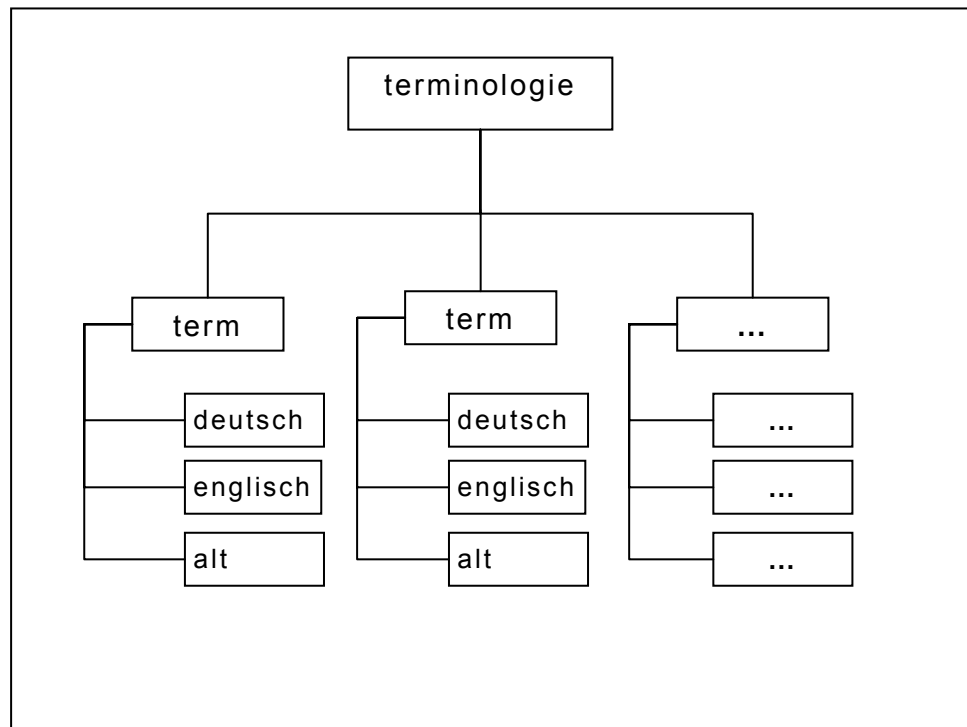


Abb. 4: Dokumentenbaum terminologie.xml

Das Dokument `terminologie.xml` enthält ein Wurzelement - `terminologie`. Dieses kann eine unbegrenzte Anzahl an `term`-Elementen enthalten. Ein Element `term` kann nun wiederum die Unterelemente `deutsch`, `englisch` und `alt` besitzen.

Die Elementtyp-Deklaration wird in der DTD durch das Schlüsselwort `!ELEMENT` eingeleitet. Daraufhin folgen der Name des Elements sowie das Inhaltsmodell. (Shepherd 2002, S. 87)

Als erstes wird das Wurzelement `terminologie` deklariert.

```
<!ELEMENT terminologie (term+)>
```

Das Element `terminologie` besitzt ein Unterelement `term`. Das Plus-Zeichen (+) legt fest, dass dieses Element mindestens einmal, aber auch beliebig oft vorkommen kann. (Hauser 2006, S. 38)

Nach dem gleichen Prinzip werden nun die Unterelemente `deutsch`, `englisch` und `alt` zugeordnet.

```
<!ELEMENT term (deutsch,englisch,alt*)>
```

Die Elemente `deutsch` und `englisch` sollen genau einmal vorkommen. Dies wird dadurch ausgedrückt, dass die Namen allein - ohne zusätzliche Zeichen - stehen. Bei dem Element `alt` hingegen kennzeichnet der Stern (\*), dass es keinmal bis beliebig oft vorkommen kann. (Hauser 2006, S. 38)

Die Elemente `deutsch`, `englisch` und `alt` enthalten keine weiteren Unterelemente sondern Zeichendaten.

```
<!ELEMENT deutsch (#PCDATA)>  
<!ELEMENT englisch (#PCDATA)>  
<!ELEMENT alt (#PCDATA)>
```

Das Schlüsselwort `#PCDATA` bedeutet `Parsed Character Data` und weist auf die Zeichendaten hin (Jänecke 2003, S. 416).

Elemente können durch Attribute mit zusätzlichen Informationen versehen werden. Attribute, die für die Elemente verwendet werden, werden in der DTD in so genannten *Attributlisten* deklariert. Dabei wird neben Elementtyp und Attributname auch noch ein Attributtyp definiert. (Jänecke 2003, S. 420)

Eine Attributliste wird mit dem Befehl `<!ATTLIST>` festgelegt. Sie gilt für ein Element, kann aber mehrere Attribute beinhalten. (Hauser 2006, S. 40)

```
<!ATTLIST term typ CDATA #REQUIRED>
```

Für das Element `term` wird hier ein Attribut mit dem Namen `typ` deklariert. Neben dem Attributnamen gibt es noch zwei weitere Angaben - den Attributtyp und einen Standardwert. Der Attributtyp `CDATA` ist der Standardtyp für normale Zeichendaten. Der Standardwert `#REQUIRED` besagt, dass das Element `term` immer das Attribut `typ` enthalten muss.

## 6 Transformation des XML-Dokuments

Im vorigen Abschnitt wurde beschrieben, dass XML ein geeignetes Format zur Datenspeicherung ist. Mit dem XML-Dokument allein kann jedoch der Benutzer nicht arbeiten. Daher ist es notwendig, das XML-Dokument in ein ausgabefähiges Format umzuwandeln.

Für diesen Zweck wurde die Extensible Stylesheet Language (XSL) geschaffen.

### 6.1 Grundlagen der Extensible Stylesheet Language

Die Extensible Stylesheet Language (XSL) umfasst drei Technologien, die unabhängig voneinander arbeiten können:

- XML Path Language (XPath)
- XML Transformations (XSLT)
- XSL Formatting Objects (XSL-FO)

XSL-FO wird dazu verwendet, Daten für die Druckausgabe zu formatieren. Da XSL-FO im praktischen Teil dieser Diplomarbeit keine Anwendung findet, soll hier nicht näher darauf eingegangen werden.

Die folgenden Abschnitte sollen einen Überblick über die Technologien XPath und XSLT geben.

### 6.1.1 XPath

Die XML Path Language (XPath) ist ein im Jahr 1999 verabschiedeter Standard, der es ermöglicht, auf Elemente innerhalb eines XML-Dokuments zuzugreifen (Hauser 2006, S. 77). XPath wird unter anderem dazu verwendet, Teile des XML-Dokuments zu identifizieren und für XSL-Transformationen bereit zu stellen (Stein 2002, S. 93).

XPath betrachtet ein XML-Dokument als Knotenbaum, der dem DOM-Baum (Document Object Model) ähnelt. Man unterscheidet sieben Arten von Knoten:

- Wurzelknoten
- Elementknoten
- Attributknoten
- Textknoten
- Kommentarknoten
- Verarbeitungsanweisungsknoten
- Namensraumknoten

Dabei enthält der Wurzelknoten das gesamte Dokument und besitzt im Gegensatz zu anderen Knoten keine Eltern. (Tidwell 2003, S. 44)

#### **Lokalisierungspfade**

XPath wird dazu verwendet, um *Lokalisierungspfade* zu erstellen, die die Position eines Eintrags im XML-Dokument beschreiben. Sie dienen dazu, bestimmte Knoten aufzufinden. (Tidwell 2003, S. 43-49)

Zur Adressierung der Knoten werden folgende XPath-Ausdrücke verwendet:

.	selektiert den aktuellen Knoten
/	selektiert den Wurzelknoten
elementname	selektiert das Element
@attributname	selektiert ein Attribut
..	übergeordneter Elternknoten
//elementname	greift auf Knoten mit dem Namen elementname in beliebig tiefer Verschachtelung zu

Die einzelnen Arten können dabei miteinander verbunden werden. (Hauser 2006, S. 79 f.)

Eine weitere Möglichkeit zur Adressierung von Knoten sind die *Achsen*. Mit ihnen werden die Beziehungen zwischen dem Ausgangsknoten zum gesuchten Knoten bestimmt. (Stein 2002, S. 95)

Tabelle 2 zeigt einige Achsen und ihre Bedeutung.

Achse	Beschreibung
ancestor	Liefert die übergeordneten Knoten.
ancestor-or-self	Liefert die übergeordneten Knoten inklusive des aktuellen Knotens.
attribute	Liefert die Attribute des aktuellen Knotens.
child	Wählt den Kindknoten des aktuellen Knotens.
descendant	Wählt alle Kindknoten des aktuellen Knotens.
descendant-or-self	Wählt alle Kindknoten des aktuellen Knotens und den aktuellen Knoten selbst.
following	Liefert alle Knoten nach dem Ende des aktuellen Knotens. Das Ende des Knotens ist das End-Tag.
following-sibling	Liefert alle Geschwisterknoten, die nach dem aktuellen Knoten folgen.
namespace	Liefert den Namensraum eines Elements.
parent	Wählt das Elternelement des aktuellen Elements.
preceding	Liefert alle Knoten vor dem Start-Tag des aktuellen Knotens.
preceding-sibling	Liefert alle Geschwisterknoten vor dem aktuellen Knoten.
self	Liefert das aktuelle Element selbst

Tab. 2: Achsen in XPath

Durch die Auswahl einer Achse kann man den Ausdruck einschränken. Es wird nur aus den Knoten ausgewählt, der durch die Achse angegeben ist. Diese Knotengruppe kann durch den so genannten *Knotentest* weiter eingeschränkt werden. Ein Knotentest steht durch

zwei Doppelpunkte (: :) getrennt hinter der Achse. Der Knotentest kann beispielsweise einen Elementnamen enthalten. (Harold 2004, S. 490)

In XSLT werden XPath-Ausdrücke in den Attributen `match` und `select` verwendet (Tidwell 2003, S. 43).

### 6.1.2 XSLT

XSLT steht für Extensible Stylesheet Language for Transformations und ist eine offizielle Empfehlung des W3C. Sie bietet die Möglichkeit, XML-Dokumente in andere Formate umzuwandeln. (Tidwell 2003, S. 1)

Es werden zwei Arten von Transformationen unterschieden. Die erste ist die *Seitwärtstransformation*, bei der von XML in ein weiteres XML-Format umgewandelt wird. Die zweite Variante ist die *Abwärts-transformation*. Dabei wird ein XML-Dokument in ein Ausgabeformat wie zum Beispiel HTML (Hypertext Markup Language), PDF (Portable Document Format) oder SVG (Scalable Vector Graphics) transformiert. (Hauser 2006, S. 66)

Eine XSLT-Transformation umfasst drei Dokumente:

1. das Quell-Dokument
2. das XSLT-Dokument
3. das Ergebnis-Dokument

Die Grundlage ist das XML-Dokument, das als Quell-Dokument dient. Das XSLT-Dokument beinhaltet die Regeln, nach denen das Ergebnis-Dokument erstellt wird. (Stein 2002, S. 135)

Für die Anwendung einer XSLT-Transformation auf ein XML-Dokument gibt es zwei Möglichkeiten. Die Erste ist, das XSLT-Dokument als Stylesheet einzubinden. Das geschieht mit folgender Anweisung im Prolog des XML-Dokuments:

```
<?xml-stylesheet type="text/xsl" href="beispiel.xslt"?>
```

Das Attribut `href` gibt hier den Ort des XSLT-Dokuments an.

Die zweite Möglichkeit ist die Transformation in eine neue Datei. Hierzu wird ein XSLT-Prozessor benötigt (Stein 2002, S. 142 f.). Dieser interpretiert die Angaben im Stylesheet gemeinsam mit den XPath-Anweisungen. Dadurch ist er in der Lage, die Daten zu extrahieren und mit dem XML-Dokument zu verknüpfen (Abbildung 5).

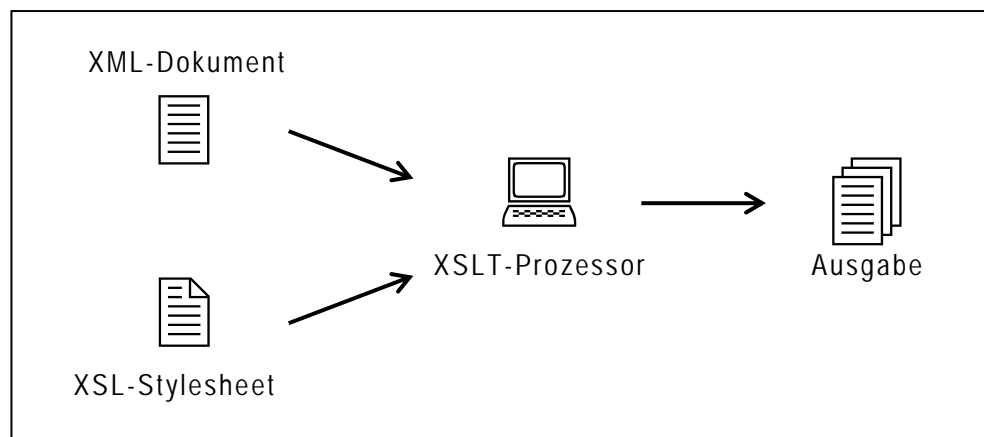


Abb. 5: Der XSLT-Prozessor

Der Internet Explorer enthält zum Beispiel XML-Module, zu deren Komponenten der XML- und der XSL-Prozessor (MSXML/MSXSL) gehören. Dadurch kann das Ausgabeformat sofort im Browser betrachtet werden.

XSLT bietet viele Möglichkeiten, die Inhalte des Quell-Dokuments zu verarbeiten. Einige davon sollen im folgenden Abschnitt am Beispiel der Terminologie-Datenbank erläutert werden.

## 6.2 Realisierung

Das Grundgerüst der Terminologie-Anwendung bildet ein Frameset. Der Anzeigebereich des Browsers wird über verschachtelte Frames eingeteilt, die wie folgt in der `index.html` definiert werden.

```
<frameset rows="120,50,*">
  <frame src="kopf.html" frameborder="0" name="logo"
    scrolling="No" noresize="noresize" />
  <frameset cols="250,*">
    <frame src="sprache.html" frameborder="0" name=
      "sprache" scrolling="No" noresize="noresize" />
    <frame src="suche.html" frameborder="0" name=
      "suche" scrolling="No" noresize="noresize" />
  </frameset>
  <frameset cols="250,*">
    <frame src="links_d.html" frameborder="0"
      name="links" scrolling="No" noresize="noresize"
      />
    <frame src="ausgabe.html" frameborder="0" name=
      "ausgabe" noresize="noresize" />
  </frameset>
</frameset> </noframes>
</frameset>
```

Dadurch ergeben sich fünf Fenster, in denen unterschiedliche XHTML-Dokumente angezeigt werden können. Mit dem Attribut `name` bekommen die einzelnen Frames Namen zugeordnet, die später den gezielten Zugriff über Links ermöglichen. (Born 2003, S. 380) Abbildung 6 zeigt das erzeugte Seiten-Layout und die Benennung der Frames.

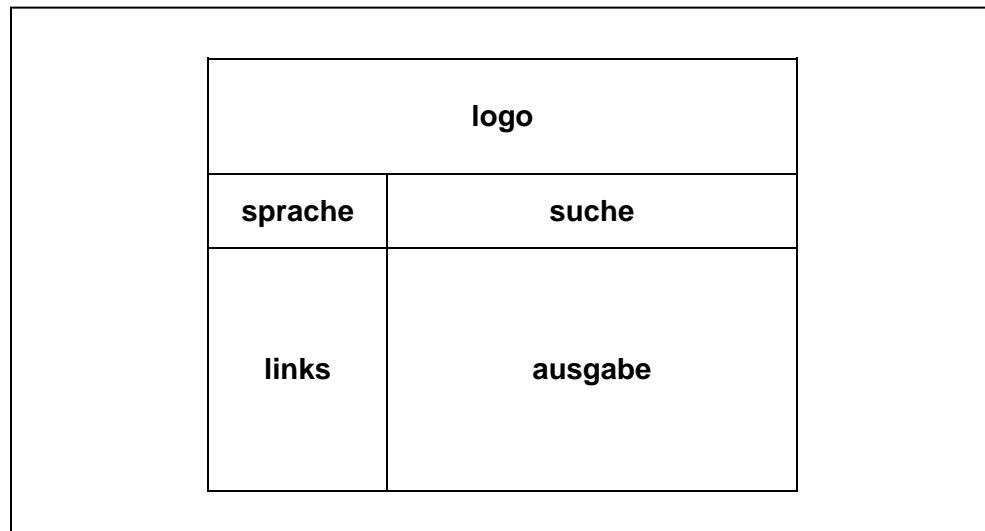


Abb. 6: Einteilung der Webseite

Die Verwendung von Frames bietet den Vorteil, dass die einzelnen Seiten einfach und unabhängig voneinander ausgetauscht werden können. Andere Bereiche hingegen, wie zum Beispiel der Frame `logo`, bleiben permanent eingeblendet. (Jacobsen 2002, S. 132) Für den Benutzer bleibt diese Aufteilung auf den ersten Blick unsichtbar. Dies wird durch das Setzen der Attribute `frameborder="0"`, das für einen rahmenlosen Frame sorgt, und `scrolling="No"`, das die Bildlaufleiste unterdrückt, erreicht.

Beim Aufruf der Datei `index.html` werden die im Frameset zugeordneten XHTML-Dokumente in den entsprechenden Frames ausgegeben. (Abbildung 7)

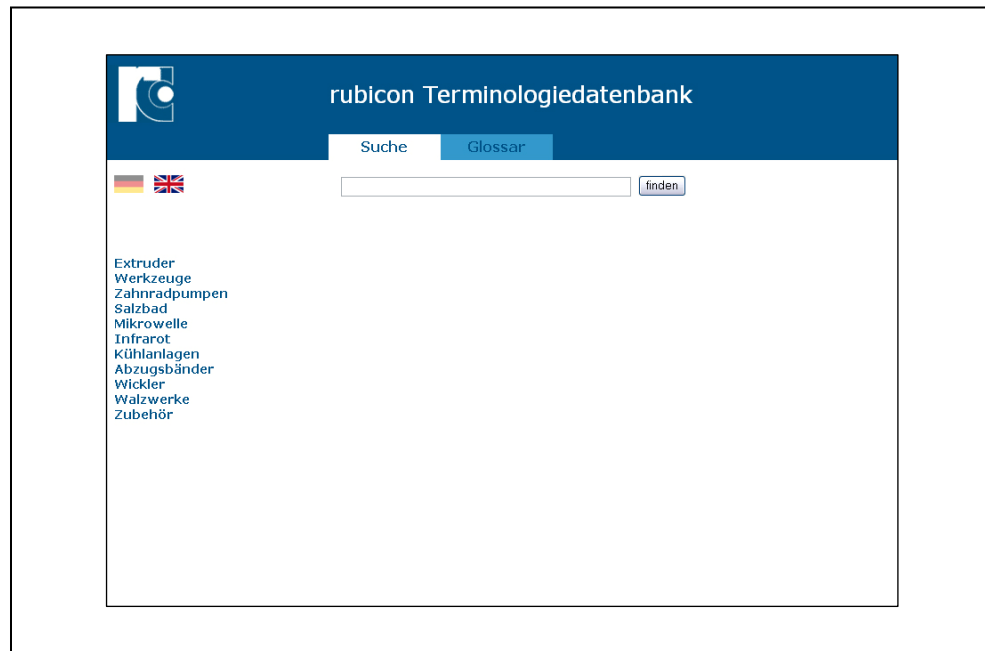


Abb. 7: Layout der Startseite

Die Seite bietet drei Suchmöglichkeiten. Der Benutzer kann sich über die vertikale Navigation alle Einträge entsprechend der Sachgebiete anzeigen lassen. Das Formularfeld im oberen Seitenbereich bietet die Möglichkeit einer Freitextsuche. Die dritte Variante ist das Glossar, das die Einträge in alphabetischer Reihenfolge auflistet.

Den im Frameset dargestellten HTML-Dokumenten ist jeweils ein XSLT-Dokument zugeordnet, das die Informationen für die Transformation enthält. Als Quell-Dokument dient die Datei `terminologie.xml`. Der Frame `ausgabe` soll später das Ziel für die Ergebnisdokumente der XSLT-Transformationen sein. Beim Aufruf der Seite wird hier zunächst die Datei `blank.html` angezeigt. Es handelt sich dabei um ein XHTML-Dokument mit leerem `body`-Element und dient hier als Platzhalter.

Die XSLT-Transformation soll in den nächsten Abschnitten anhand der vertikalen Navigation erläutert werden. Grundlage dafür sind die

Dateien `links_d.html` und `links_d.xsl`. Die weiteren Funktionen der Seite, wie zum Beispiel der Abruf der Daten über das Glossar, ähneln dieser Transformation und werden deshalb nicht nochmals ausführlich beschrieben. Ihre Besonderheiten werden im Abschnitt 6.2.3 dargestellt.

### 6.2.1 Aufbau des XHTML-Dokuments

XHTML-Dokumente haben einen streng festgelegten Aufbau. An erster Stelle steht die DOCTYPE-Deklaration. Sie enthält den öffentlichen Bezeichner, der auf die DTD verweist.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0  
Transitional//EN" "http://www.w3.org/TR/xhtml1/  
DTD/xhtml1-transitional.dtd">
```

Das Wurzelement des Dokuments ist das `html`-Element, das die `xmlns`-Deklaration für den XHTML-Namensraum enthält. (Mintert 2003, S. 371)

```
<html xmlns="http://www.w3.org/1999/xhtml">  
...  
</html>
```

Die weitere Struktur unterteilt sich in den Dokumentenkopf, den `head`, und den eigentlichen Inhalt, den `body`.

## Head

Im `<head>`-Tag finden die Anweisungen Platz, die für das komplette Dokument gelten. Dazu gehören beispielsweise der Titel des Dokuments, die Meta-Angaben und die Stylesheet-Definition. (Herold 2002, S. 11)

```
<head>
  <meta http-equiv="Content-Type" content="text/html;
    charset=iso-8859-1" />
  <title>Terminologie-Datenbank</title>
  <link rel="stylesheet" href="terminologie.css"
    type="text/css" />
  ...
</head>
```

Zusätzlich befindet sich hier ein JavaScript, das die Transformation auslöst und gleichzeitig eine Variable an das XSLT-Dokument übergibt.

## Skript-Bereich

Der Skript-Bereich wird durch das `<script>`-Tag eingeleitet. In diesem gibt das Attribut `type` die verwendete Skriptsprache, `text/javascript`, an. Die zusätzliche Verwendung des `language`-Attributs ist nicht zwingend erforderlich, gewährleistet aber, dass auch ältere Browser die Skriptsprache erkennen. (Herold 2002, S. 251 f.)

```
<script language="JavaScript" type="text/javascript">
<!--
...
//-->
</script>
```

Der Inhalt des Skript-Bereichs wird als besonderer Text behandelt, der nicht ausgegeben wird. Da es bei älteren Browsern jedoch passieren kann, dass dieser Bereich als Text ausgegeben wird, wird der Inhalt auskommentiert. (Herold 2002, S. 251)

Innerhalb der Kommentarzeichen befindet sich die JavaScript-Funktion `Ausgabe(name)`. Eine Funktion ist ein Programmblock, der nicht sofort ausgeführt wird. Der Aufruf erfolgt später über die Hyperlinks im `body`-Bereich. Im Funktionskopf wird der Parameter `name` definiert. Dieser kann in der Funktion direkt angesprochen werden. (Wenz 2007, S. 89)

```
function Ausgabe(name)
{
  Programmblock
}
```

Im Programmblock selbst werden zunächst die benötigten Variablen definiert. Sie werden dazu verwendet, um Daten zwischenspeichern. Die Zuweisung der entsprechenden Werte erfolgt durch das Gleichheitszeichen (=). Auf der linken Seite des Gleichheitszeichens steht der Name der Variablen und auf der rechten der zugewiesene Wert. (Wenz 2007, S. 65 f.)

```
var xmlfile,xslfile,xmlinput,xslinput,htmloutput,
    meineVariable,suchbegriff;

suchbegriff = name;
xmlfile = "terminologie.xml";
xslfile = "links_d.xsl";
```

Der Wert des Parameters `name` wird hier der Variablen `suchbegriff` zugeordnet. Die benötigten XML- und XSLT-Dokumente werden in den Variablen `xmlfile` und `xslfile` gespeichert.

Das Laden des XML- sowie des XSLT-Dokuments erfolgt durch den Aufruf des `ActiveXObject`-Objektconstructors. Dieser ermöglicht den Zugriff auf den MSXML-Parser.

```
Xmlinput = new ActiveXObject("MSXML2.DOMDocument");
xmlinput.async = false;
xmlinput.load(xmlfile);

xslinput = new ActiveXObject("MSXML2.DOMDocument");
xslinput.async = false;
xslinput.load(xslfile);
```

Mit dem Wert der Variablen `suchbegriff` sollen später die Inhalte der XML-Datei abgefragt werden. Dazu wird dieser zunächst an das XSLT-Dokument übergeben.

```
meineVariable =
    xslinput.documentElement.getElementsByTagName
        ("xsl:variable");
meineVariable[0].firstChild.nodeValue = suchbegriff;
```

Das bei der Transformation entstandene Ergebnis-Dokument wird in den Frame `ausgabe` ausgegeben.

```
htmloutput =
    xmlinput.transformNode(xslinput.documentElement);
parent.frames["ausgabe"].document.open();
parent.frames["ausgabe"].document.write(htmloutput);
parent.frames["ausgabe"].document.close();
```

## Body

Der Body umfasst den eigentlichen Inhalt des Dokuments. In ihm ist notiert, was beim Öffnen der Seite im Browser angezeigt wird. Bei der Datei `links_d.html` sind dies Links, die als Navigation dienen.

```
<body >
  <a href="javascript:Ausgabe('extruder')">Extruder</a>
  <br/>
  <a href="javascript:Ausgabe('werkzeug')">Werkzeuge</a>
  <br/>
  ...
</body>
```

Die HTML-Links verweisen auf das Protokoll "javascript:". Mit ihm wird durch Klicken des Links die JavaScript-Funktion `Ausgabe()` aufgerufen. (Wenz 2007, S. 60 f.) Die in Klammern angegebenen Werte (z. B. `extruder`) werden zur weiteren Verarbeitung an den Parameter `name` übergeben.

## 6.2.2 Struktur des XSLT-Dokuments

Ein XSLT-Dokument basiert auf XML, was bedeutet, dass es ebenfalls wohlgeformt sein muss. Genau wie ein XML-Dokument beginnt es mit einer XML-Deklaration.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

Das Wurzelement in XSLT ist `xsl:stylesheet`. Das Tag enthält die Versionsnummer sowie den entsprechenden Namensraum. (Stein 2002, S. 135 f.)

```
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
...
</xsl:stylesheet>
```

Das Format, in welches transformiert wird, wird mit dem `<xsl:output>`-Tag festgelegt. Mögliche Ausgabemethoden sind `xml`, `html` und `text`. (Tidwell 2003, S. 29) Da die Terminologie-Daten als HTML-Dokument ausgegeben werden soll, ist hier `html` gewählt.

```
<xsl:output method="html"/>
```

Mit dem Element `xsl:variable` wird eine Variable namens `suchbegriff` definiert.

```
<xsl:variable name="suchbegriff">...</xsl:variable>
```

Als Wert der Variablen werden an dieser Stelle nur drei Punkte eingesetzt. Sie dienen als Platzhalter für die Variable, die aus dem HTML-Dokument übergeben wird.

Der wichtigste Teil des XSLT-Dokuments ist das Template. Im `<xsl:template>`-Tag wird die gesamte Ausgabe festgelegt.

```
<xsl:template match="/">

</xsl:template>
```

Über das Attribut `match` wird geregelt, welcher Teil des XML-Dokuments in diesem Template verarbeitet wird. Der Attributwert `"/` ist ein XPath-Ausdruck für das Wurzelement. (Hauser 2006, S. 71)

Innerhalb des Templates befindet sich der komplette Inhalt des Ergebnis-Dokuments.

```
<html>
  <head>
    <title>Ergebnis</title>
    <link rel="stylesheet" href="terminologie.css"
          type="text/css" />
  </head>
  <body>
    <table>
      <xsl:for-each select="terminologie/term">
        <xsl:sort select="deutsch" />
        <xsl:if test="@typ = $suchbegriff">
          <tr>
            <td><xsl:value-of select="deutsch"/></td>
            <td><xsl:value-of select="englisch"/></td>
          </tr>
        </xsl:if>
      </xsl:for-each>
    </table>
  </body>
</html>
```

Neben dem erforderlichen HTML-Code gehören auch die XSLT-Elemente (fett markiert) dazu, die es ermöglichen, Daten auszulesen und in einem anderen Format auszugeben. Elemente, die innerhalb des Templates stehen, werden auch *Instruktionselemente* genannt. (Ammelburger 2004, S. 343 f.)

### **xsl:for-each**

Das Element `xsl:for-each` wird verwendet, um alle Knoten, die ein bestimmtes Kriterium erfüllen, zu verarbeiten. Man kann es auch als eine Art Schleife bezeichnen. (Tidwell 2003, S. 72) Sie durchläuft alle Elemente, die der XPath-Ausdruck im `select`-Attribut bestimmt. In diesem Beispiel handelt es sich um die `term`-Elemente des XML-Dokuments.

### **xsl:sort**

Die Schleife wird hier in Verbindung mit dem Element `xsl:sort` verwendet. Dieses bewirkt, dass die Ergebnisliste sortiert ausgegeben wird. Der XPath-Ausdruck im Attribut `select` gibt an, dass die Liste nach dem deutschen Begriff sortiert werden soll. (Hauser 2006, S. 89)

### **xsl:if**

Mit `xsl:if` kann eine Fallunterscheidung getroffen werden. Geprüft wird der XPath-Ausdruck im Attribut `test`. Wenn das Attribut `type` des `term`-Elements mit der XSL-Variablen `suchbegriff` übereinstimmt, liefert die Auswertung des Werts `test` den Wert `true`. Ist

dies der Fall, werden die Elemente verarbeitet. Beim Wert `false` wird der Inhalt des `xsl:if`-Elements ignoriert. (Tidwell 2003, S. 68)

### 6.2.3 Umsetzung weiterer Funktionen

Mit Hilfe von XSLT wurden weitere Funktionen der Webseite umgesetzt, deren Besonderheiten in diesem Abschnitt erläutert werden.

#### Suchformular

Bei der Suche über das Suchformular findet eine ähnliche Transformation statt wie zuvor beschrieben. Ein Unterschied besteht in der JavaScript-Funktion. Hier wird der Suchbegriff nicht anhand eines Parameters übergeben, sondern direkt dem Wert des Formularfeldes zugeordnet.

```
suchbegriff=document.forms[0].eingabe.value;
```

Der Aufruf der Funktion erfolgt beim Klicken des Submit-Buttons.

```
<form action="" onSubmit="Ausgabe()">  
  <input name="eingabe" />  
  <input type="submit" value="finden" />  
</form>
```

Ein weiterer Unterschied besteht im `xsl:if`-Element in der Datei `suche.xsl`. Bei der zuvor beschriebenen Variante wurden zwei Begriffe auf komplette Übereinstimmung überprüft. Bei einer Suchfunktion macht das aber nicht viel Sinn, da der Benutzer den zu suchenden Begriff exakt kennen müsste, um ein Ergebnis zu bekommen. Er soll aber zum Beispiel bei Eingabe des Begriffes „Extruder“

auch die Begriffe „Standardextruder“, „Stiftextruder“ usw. angezeigt bekommen.

Ermöglicht wird dies durch den Einsatz der Funktion `contains()`. Sie untersucht, ob das erste Argument das zweite enthält. Trifft das zu, wird der Wert `true` zurückgeliefert. (Tidwell 2003, S. 354)

```
<xsl:if
  test="contains(
    translate(deutsch, 'ABCDEFGHIJKLMNOPQRSTUVWXYZ',
      'abcdefghijklmnopqrstuvwxyz'), translate($suchbegriff,
      'ABCDEFGHIJKLMNOPQRSTUVWXYZ', 'abcdefghijklmnopqrstuvwxyz'))"
  >
  ...
</xsl:if>
```

Um zu gewährleisten, dass alle Begriffe unabhängig von Groß- und Kleinschreibung gefunden werden, wird hier zusätzlich die Funktion `translate()` eingesetzt. Diese Funktion konvertiert einzelne Zeichen einer Zeichenkette in einen anderen Wert. Dafür müssen drei Zeichenketten angegeben werden. Als erste wird die ursprüngliche Zeichenkette benannt, zum Beispiel die Variable `suchbegriff`. Die Zweite und die Dritte geben die zu konvertierenden Zeichen an. Stimmt nun ein Zeichen des Variablenwerts mit einem des zweiten Zeichenkettenarguments überein, wird es durch das entsprechende dritte Zeichen ersetzt. Das Zeichen „A“ wird zu „a“, „B“ wird „b“ usw. (Tidwell 2003, S. 431)

Gibt die `if`-Anweisung den Wert `false` zurück, d. h., es wurden keine Übereinstimmungen gefunden, soll im Ergebnis-Dokument die Meldung „Keine Suchergebnis.“ erscheinen. Im Gegensatz zu einer

Programmiersprache funktioniert die Fallunterscheidung jedoch nur in einem Fall. (Tidwell 2003, S. 68) Um herauszubekommen, wann die Anfrage kein Ergebnis hat, werden die gefundenen Knoten mit Hilfe der Funktion `count()` gezählt.

```
<xsl:if test="count(terminologie/term/deutsch
[contains(translate(.,'ABCDEFGHIJKLMNOPQRSTUVWXYZ',
'abcdefghijklmnopqrstuvwxy'), translate($suchbegriff,
'ABCDEFGHIJKLMNOPQRSTUVWXYZ', 'abcdefghijklmnopqrstuvwxy'))
]) = 0">
<p>Kein Suchergebnis.</p>
</xsl:if>
```

Ist die Anzahl der Knoten gleich null, wird ein Absatz mit dem entsprechenden Text als Inhalt transformiert.

## Glossar

Ähnlich wie bereits bei der Suchfunktion soll auch beim Glossar kein kompletter Begriff gesucht werden. Hier werden nur die Anfangsbuchstaben der Datenbank-Einträge benötigt. Dazu wird zunächst mit dem Element `xsl:key` ein Schlüssel definiert.

```
<xsl:key name="keyAlphabet"
match="terminologie/term/deutsch" use="substring(.,1,1)"/>
```

Das Attribut `name` gibt dem Schlüssel einen Namen. Der entsprechende Knoten wird über das Attribut `match` festgelegt. Mit dem Attribut `use` wird die Eigenschaft, die zum Erzeugen des Schlüssels verwendet wird, definiert. (Tidwell 2003, S. 283) Die darin verwendete Funktion `substring()` liefert den ersten Buchstaben des Eintrags.

Der festgelegte Schlüssel kann nun von der Funktion `key()` verwendet werden. Diese liefert alle Knoten, deren Werte für den angeforderten Schlüssel zum Suchargument passen.

```
<xsl:for-each
  select="key('keyAlphabet', $buchstabe) | key('keyAlphabet',
  translate($buchstabe, 'abcdefghijklmnopqrstuvwxyz',
  'ABCDEFGHIJKLMNOPQRSTUVWXYZ'))" >
...
</xsl:for-each>
```

## 7 AJAX

Der Begriff AJAX wurde erstmals im Februar 2005 in dem Artikel „Ajax: A New Approach to Web Applications“ von Jesse James Garrett erwähnt. Nach Garrett steht das Akronym für **A**synchronous **J**avaScript and **X**ML. ([www.adaptivepath.com](http://www.adaptivepath.com))

AJAX ist keine Programmiersprache. Es steckt auch keine neue Technik dahinter. Vielmehr geht es darum, verschiedene längst etablierte Technologien zu nutzen und damit neue Möglichkeiten zu eröffnen. (Carl 2006, S. 2) Der bisher für die nächste Generation der Webentwicklung verwendete Begriff „Web 2.0“ wurde durch AJAX abgelöst. Die Grundlagen sind also nicht neu, sie fanden bisher aber nicht in diesem Ausmaß Anwendung. (Gamperl 2006, S. 9)

Das Konzept, das dahinter steckt, kann man mit „Nachladen statt Neuladen“ betiteln. Bisher war es nur möglich, das Aussehen einer Webseite durch Neuladen der kompletten Seite zu verändern. Klickt man beispielsweise in der Navigation einer einfachen Seite einen Link an, der in einem anderen Bereich der Seite einen Text ändern soll, wird die Seite komplett neu geladen. Das einzige was sich jedoch geändert hat, ist der Text. (Carl 2006, S. 2-4)

Abbildung 8 stellt die Übertragung einer solchen Webseite dar.

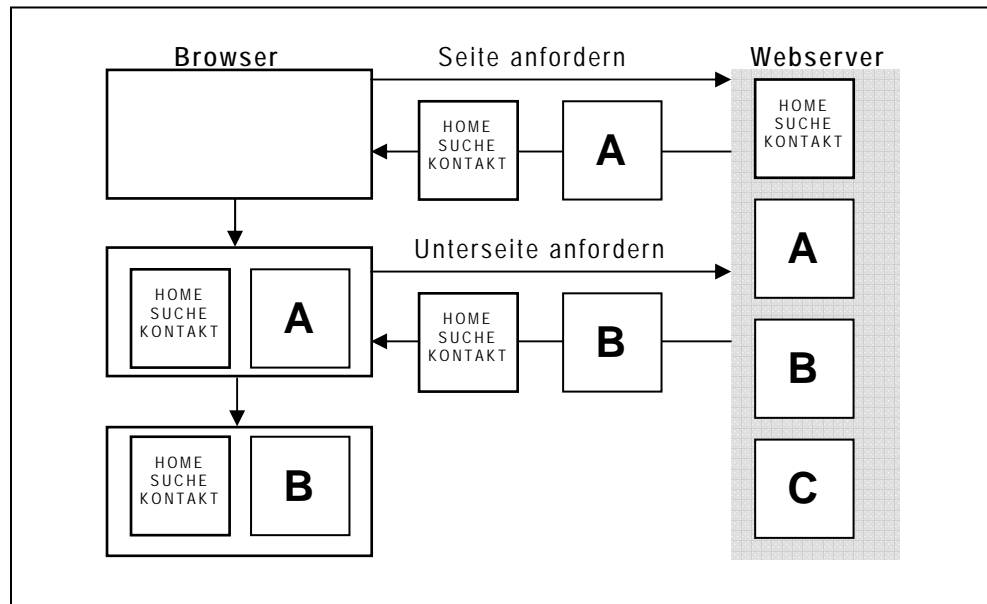


Abb. 8: Übertragung ohne AJAX

Mit AJAX ist es nun möglich, die Verbindung zum Webserver aufzubauen und nur den Text der Unterseite zu laden. Dieser wird dann an einer klar definierten Stelle im Browser eingefügt. (Abbildung 9) Das Ganze erfolgt ohne ein komplettes Neuladen der Seite. (Carl 2006, S. 3 f.)

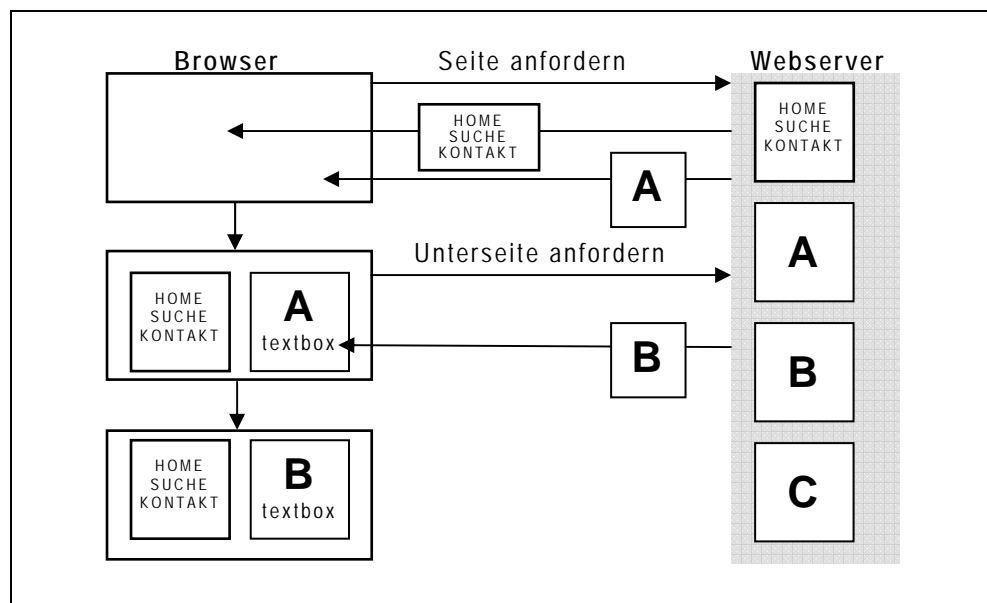


Abb. 9: Übertragung mit AJAX

## 7.1 Grundlagen

Die Basis einer AJAX-Anwendung stellt die Kommunikation zwischen Client und Server dar. Die Technik, die es ermöglicht, per JavaScript eine Anfrage an einen Server zu senden und dessen Antwort auszuwerten, nennt sich XMLHttpRequest-Objekt. (Gamperl 2006, S. 165)

Bevor nun hier diese Technologie näher erklärt wird, soll zunächst auf die Grundlagen des Hypertext Transfer Protocols, die notwendig für das Verständnis von Client/Server-Anwendungen sind, eingegangen werden.

### 7.1.1 Das Hypertext Transfer Protocol

Zum Austausch der Daten wird das Hypertext Transfer Protocol (HTTP) verwendet. Hauptsächlich wird es dazu eingesetzt, um Daten aus dem Internet in einen Browser zu laden. Wie Abbildung 10 zeigt, wird dabei über einen Webbrowser, dem Client, eine Anfrage an einen Server gesendet. Die Anfrage wird vom Server entgegengenommen und verarbeitet. Anschließend liefert er eine Antwort an den Client zurück. (Gamperl 2006, S. 166)

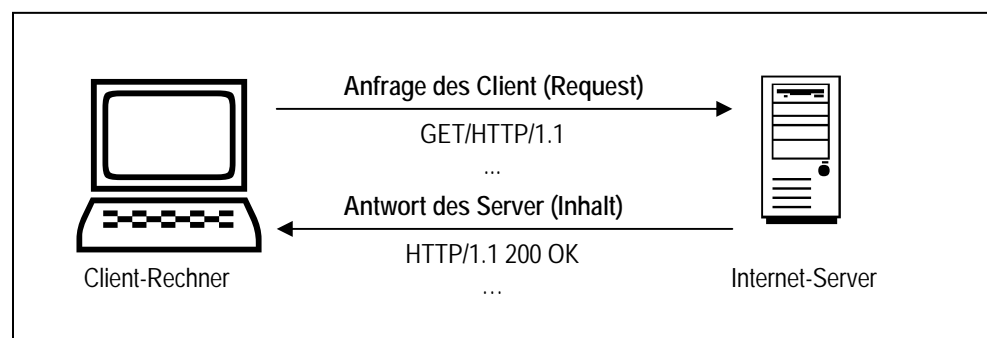


Abb. 10: Kommunikation zwischen Client und Server via HTTP

Der Server wird durch einen mitgesendeten Befehl, den Request, zu bestimmten Aktionen veranlasst. Dazu kennt HTTP verschiedene Methoden. Die am häufigsten verwendeten und gleichzeitig für AJAX am bedeutendsten sind GET und POST. Mit GET werden die entsprechenden Seiten vom Server angefordert. Ebenso arbeitet die Methode POST, bei der aber zusätzlich ein Datenblock übermittelt wird. Dieser wird beispielsweise bei der Übertragung von Formularinhalten verwendet. (Gamperl 2006, S. 167)

Auf jede Anfrage liefert der Server in der ersten Zeile des HTTP-Headers einen Antwortcode zurück. Dieser setzt sich aus der HTTP-Version des Servers, einem dreistelligen Statuscode und der Beschreibung des Ergebnisses zusammen. (Gamperl 2006, S. 168)

Eine erfolgreiche Anfrage könnte demnach so aussehen:

```
HTTP/1.1 200 OK
```

Die Statuscodes werden in Wertebereiche eingeteilt, die in Tabelle 3 dargestellt werden.

Wertebereich	Beschreibung
100-199	Informationen während der Anfrage. Die Anfrage wird dabei vom Server noch bearbeitet.
200-299	Erfolgreiche Anfrage. Die eigentliche Aktion kann ausgeführt werden.
300-399	Umleitung der Anfrage. Eine weitere Bearbeitung der Anfrage wird nötig.
400-499	Die Anfrage war unvollständig bzw. fehlerhaft und wurde daher abgebrochen.
500-599	Es ist ein Fehler auf dem Server aufgetreten.

Tab. 3: Wertebereich der Statuscodes

Die hier dargestellten Eigenschaften von HTTP sind wichtig für das Verständnis der Funktionsweise von AJAX, da sie auch bei XMLHttpRequest zum Einsatz kommen. (Carl 2006, S. 26)

### **7.1.2 Das XMLHttpRequest-Objekt**

Ein essentieller Bestandteil von AJAX ist XMLHttpRequest. Es handelt sich dabei um eine Programmierschnittstelle, auch API (Application Programming Interface) genannt. Sie steht in JavaScript in Form eines Objekts zur Verfügung. Durch dieses Objekt wird der Datenaustausch ohne ein Neuladen der Seite ermöglicht. (Carl 2006, S. 26)

Entwickelt wurde diese Möglichkeit von Microsoft. Die Realisierung erfolgte mit einer ActiveX-Komponente und ist im Internet Explorer ab Version 5.0 integriert. Der ActiveX-Ansatz war natürlich nicht betriebssystemunabhängig, weshalb die Hersteller anderer Browser XMLHttpRequest als natives Browserobjekt implementiert haben. Auch in der Version 7.0 des Internet Explorers ist kein ActiveX mehr notwendig. (Wenz 2006, S. 14-16)

Die Funktionsweise von XMLHttpRequest ist relativ einfach. Es wird eine HTTP-Anfrage an einen Server geschickt und die Antwort ausgewertet. Die Darstellung der Rückgabe erfolgt mit JavaScript. (Wenz 2007, S. 395)

#### **Das Objekt instanziiieren**

Um AJAX anwenden zu können, muss zunächst ein XMLHttpRequest-Objekt erzeugt werden. Für Browser, die dieses nativ unterstützen, sieht die Syntax wie folgt aus:

```
var XMLHTTP = new XMLHttpRequest();
```

Für den Internet Explorer bis zur Version 6.0 wird dazu das ActiveX-Objekt benötigt.

```
XMLHTTP = new ActiveXObject("Microsoft.XMLHTTP");
```

Für den browserunabhängigen Einsatz kann mit einem simplen Test das vorhandene JavaScript-Objekt herausgefunden werden:

```
if (window.ActiveXObject)
    XMLHTTP = new ActiveXObject("Microsoft.XMLHTTP");
else if (window.XMLHttpRequest)
    XMLHTTP=new XMLHttpRequest();
```

Die Überprüfung erfolgt für den Internet Explorer mit der Klasse `window.ActiveXObject` und für andere Browser mit der Klasse `window.XMLHttpRequest`. (Wenz 2006, S. 16-19)

### 7.1.3 Senden einer HTTP-Anfrage

Die im vorangegangenen Abschnitt beschriebenen Objekte enthalten gemeinsame Methoden und Eigenschaften. Diese werden zur Verarbeitung der Anfragen benötigt. (Wenz 2006, S. 19)

Tabelle 4 beschreibt die möglichen Methoden.

Method	Beschreibung
abort()	Bricht den aktuell laufenden Request ab.
getAllResponseHeaders()	Gibt eine Liste aller vorhandenen Header als key/value-Paare in einem String zurück.
getResponseHeader(name)	Gibt den Wert für den im Argument angegebenen Header zurück.
open(method, url [,syncFlag,username, password])	Hiermit wird der Request an die gewünschte Zieladresse gestartet und die Art der Übertragung vereinbart.
send(body null)	Sendet den Request an den Server mit einem optionalen Body.
setRequestHeader(key, value)	Setzt einen optionalen Header für den Request als key/value-Paar.

Tab. 4: Methoden des XMLHttpRequest-Objekts

Um einen gültigen Request starten zu können, werden mindestens die Methoden `open()` und `send()` benötigt. (Gamperl 2006, S. 183)

Die HTTP-Anfrage läuft in drei Schritten ab. Zuerst wird die Anfrage erzeugt, dann an den Webserver gesendet, und zum Schluss erfolgt die asynchrone Reaktion per Callback-Methode auf das vom Webserver gelieferte Ergebnis. (Wenz 2006, S. 19)

### Schritt 1

Erzeugt wird die HTTP-Anfrage mit der Methode `open()` des XMLHttpRequest-Objekts. Dabei wird weder eine Serververbindung aufgebaut, noch wird die Anfrage abgeschickt. Die Methode erfordert zwei Parameter. Zum einen die HTTP-Methode (`GET` oder `POST`) und zum anderen die URL, an die die Anfrage geschickt werden soll. (Wenz 2006, S. 20)

Der optionale dritte Parameter bestimmt, ob die Kommunikation *synchron* (`false`) oder *asynchron* (`true`) ablaufen soll. Bei der synchronen Kommunikation wird die Ausführung des Skripts angehalten, bis die Daten vom Server zurückkommen. Asynchron bedeutet, dass die HTTP-Anfrage im Hintergrund läuft, wodurch das Skript weiter ausgeführt werden kann. Letztere sollte in der Regel angewendet werden, damit nicht die komplette JavaScript-Anwendung zum Stehen gebracht wird, wenn der Server die Anfrage nur langsam verarbeitet. (Wenz 2007, S. 396)

```
XMLHTTP.open("GET", "terminologie.xml", true);
```

## Schritt 2

Mit der Methode `send()` wird die HTTP-Anfrage nun an den Server geschickt und somit der ganze Prozess in Gang gesetzt.

```
XMLHTTP.send(null);
```

## Schritt 3

Um bei der asynchronen Anfrage Bescheid zu bekommen, wenn ein Ergebnis vom Server ankommt, wird eine so genannte Callback-Funktion eingesetzt. Diese wird in der Eigenschaft `onreadystatechange` angegeben. (Wenz 2007, S. 397)

```
XMLHTTP.onreadystatechange = callback;

function callback() {
...
}
```

Die Callback-Funktion wird nun jedes Mal aufgerufen, wenn sich der Status der Abfrage ändert. Die Eigenschaft `readyState` gibt den jeweiligen Status zurück. (Wenz 2006, S. 22) Tabelle 5 zeigt die möglichen Werte dieser Eigenschaft.

Wert	Bedeutung	Beschreibung
0	uninitialized	Der Request wurde noch nicht durch die Methode <code>open()</code> ausgelöst.
1	loading	Der Request wird gestartet, wurde aber bisher noch nicht abgeschickt.
2	loaded	Der Request wurde durch die Methode <code>send()</code> ausgeführt. Eine Antwort des Servers steht noch aus.
3	interactive	Die Übertragung der Antwort des Servers läuft. Teile davon sind bereits im Buffer und mittels der Eigenschaften <code>responseText</code> oder <code>responseXML</code> verfügbar.
4	complete	Der Request wurde vollständig ausgeführt und beendet

Tab. 5: Werte der Eigenschaft `readyState`

Für die weitere Verarbeitung ist der Wert 4 bedeutend, denn er signalisiert, dass die Abfrage erfolgreich war. Dieser kann wie folgt zur Überprüfung abgefragt werden:

```
function callback()
{
    if(XMLHTTP.readyState == 4)
    {
        ...
    }
}
```

In der Callback-Funktion können nun die Daten abgefragt werden. Das geschieht über die Eigenschaften `responseText`, `responseXML`, `status` oder `statusText`. Mit `status` und `statusText` werden der vom Server gelieferte HTTP-Statuscode bzw. sein Beschreibungstext ausgegeben. Die Eigenschaft `responseText` gibt das Ergebnis der Anfrage als JavaScript-String aus. (Wenz 2006, S. 22)

Da das Austauschformat in AJAX-Anwendungen meist XML ist, hat die Eigenschaft `responseXML` eine große Bedeutung. Mit ihr wird aus dem Inhalt der Antwort ein XML-Objekt zurückgegeben, das über das *Document Object Model* weiterverarbeitet werden kann. (Gamperl 2006, S. 188 f.)

## 7.2 Document Object Model

Das Document Object Model (DOM) ist eine plattform- und sprachunabhängige Schnittstelle, die den Zugriff auf XML- und XHTML-Dokumente ermöglicht. Mit dem DOM bietet sich die Möglichkeit, auf bestimmte Elemente eines Dokuments dynamisch zuzugreifen und diese zu bearbeiten. Das DOM ist aus der Programmiersprache JavaScript hervorgegangen und wird vom W3C definiert. (Gamperl 2006, S. 17)

Da das DOM in Zusammenhang mit AJAX eine große Rolle spielt, sollen in diesem Abschnitt die wesentlichen Grundlagen erläutert werden.

### **7.2.1 Der Dokumentenbaum**

Gemäß dem W3C-DOM-Standard stellt eine vollständig übertragene Webseite einen hierarchisch geordneten Dokumentenbaum dar. Die Struktur entspricht dabei der Struktur der Seite. Jedes vorhandene Element und jeder Text wird im Dokumentenbaum als Knoten (engl. node) dargestellt. Diese Knoten enthalten neben dem Element auch entsprechende Attribute und den Elementinhalt. Die Merkmale dieser Objekte können über die DOM-API ausgelesen werden. (Gamperl 2006, S. 18)

Innerhalb eines Elements können sich weitere Elementknoten befinden, die als Kindknoten (engl. child nodes) bezeichnet werden (Wenz 2007, S. 350).

### **7.2.2 Eigenschaften**

Jeder Knoten des Dokumentenbaums besitzt bestimmte Eigenschaften, die die Beziehungen der Knoten untereinander beschreiben. Der oberste Knoten eines Dokuments, der Wurzelknoten, wird durch das `document`-Objekt gekennzeichnet und verweist auf das erste Element im Dokument. (Gamperl 2006, S. 21)

Das Array `childNodes` umfasst die Kindknoten eines Elements. Über entsprechende Eigenschaften kann nun von einem Knoten aus auf andere Knoten zugegriffen werden. (Wenz 2007, S. 351)

Tabelle 6 zeigt die wichtigsten Eigenschaften und ihre Beschreibungen.

<b>Eigenschaft</b>	<b>Beschreibung</b>
attributes	Array mit allen Attributen eines Knotens
childNodes	Array mit allen Kindknoten eines Knotens
firstChild	erstes Kind eines Knotens
lastChild	letztes Kind eines Knotens
nextSibling	rechter Nachbarknoten
nodeName	Name des Knotens
nodeType	Typ des entsprechenden Knotens
nodeValue	Wert des Knotens
ownerDocument	document-Objekt, zu dem der Knoten gehört
parentNode	Elternknoten
previousSibling	linker Nachbarknoten

Tab. 6: Eigenschaften der DOM-Knoten

### 7.2.3 Methoden

Mit den Methoden des DOM hat man nun die Möglichkeit, Elemente zu bearbeiten, zu entfernen oder auch neu einzufügen (Wenz 2007, S. 352).

Tabelle 7 zeigt eine Übersicht der wichtigsten Methoden.

<b>Methode</b>	<b>Beschreibung</b>
appendChild	Fügt einen Knoten an das Ende eines anderen Knotens ein.
cloneNode	Kopiert einen Knoten.
hasChildNodes	Prüft, ob ein Knoten weitere Kindelemente enthält.
hasAttributes	Prüft, ob ein Knoten Attribute besitzt.
insertBefore	Fügt einen Knoten vor einen anderen Knoten im Dokumentenbaum ein.
removeChild	Entfernt ein Element aus dem Dokumentenbaum.
replaceChild	Ersetzt ein Element des Dokumentenbaums durch ein anderes Element.

Tab. 7: Methoden des node-Objekts

#### 7.2.4 Selektieren einzelner Elemente

Ein Zugriff auf ein bestimmtes Element kann aber auch über die CSS-ID, den Namen des Elementtyps oder das `name`-Attribut erfolgen. Dies ermöglichen die in Tabelle 8 dargestellten Methoden des `document`-Objekts.

Methode	Beschreibung
getElementById	Selektiert ein Element anhand seiner CSS-ID.
getElementsByName	Selektiert alle Elemente als Array, die als <code>name</code> -Attribut das Argument der Methode enthalten.
getElementsByTagName	Selektiert alle Elemente als Array, die als <code>TagName</code> das Argument der Methode enthalten.

Tab. 8: Gezieltes Ansprechen von Elementen

Mit diesen Knoten wird der Wurzelknoten ausgewählt, der dann mit den Eigenschaften und Methoden des Dokumentenbaums weiter bearbeitet werden kann. (Gamperl 2006, S. 44)

### 7.3 Realisierung

Die Weiterverarbeitung der XML-Daten soll nun genauer am Beispiel der Terminologie-Datenbank beschrieben werden. Das Layout der Seite entspricht dem der XSLT-Variante, wird jedoch nicht über ein Frameset aufgebaut. Die Einteilung der einzelnen Bereiche der Seite erfolgt hier durch Tabellen und CSS-Formatierungen.

#### 7.3.1 Aufbau der Seite

Die Startseite ist die Datei `index.html`. Der Kopfbereich mit dem Logo sowie die Bereiche Sprachauswahl und Suchfunktion werden mit Hilfe einer Tabelle positioniert. Die darunter liegenden Bereiche der vertikalen Navigationsleiste und des Ausgabebereiches werden durch `div`-Elemente strukturiert.

```
<div id="main">
  <div id="left"> Navigation </div>
  <div id="content"> Hauptteil </div>
</div>
```

Um das Ändern der vertikalen Navigation zu ermöglichen, ohne eine komplett neue Seite zu erstellen, stehen die Links nicht direkt im Dokument. Die Links für die deutsche Navigation befinden sich im Dokument `deutsch.html`, die der englischen im Dokument `englisch.html`. Sie werden über eine JavaScript-Funktion in das Hauptdokument geladen. Zunächst wird dafür mit der Funktion `ladeNavi(url)` ein `iframe`-Element im `div`-Element des Navigationsbereichs erzeugt.

```
function ladeNavi(url)
{
  document.getElementById("left").innerHTML =
  '<iframe name="iframe" src="' + url + '" onload="inhalt();"
  style="display:none;"></iframe>';
}
```

Eine weitere Funktion (`inhalt()`) fügt nun den Inhalt ein.

```
function inhalt()
{
  insert = iframe.document.body.innerHTML;
  document.getElementById("left").innerHTML = insert;
}
```

Beim Laden der `index.html` wird die Funktion `ladeNavi()` mit dem Parameter `deutsch.html` ausgeführt, wodurch die deutsche Navigation aufgerufen wird. Dies erfolgt durch den Event-Handler `onload` im einleitenden `<body>`-Tag. Ein Wechsel kann dann über die Buttons der Sprachauswahl erfolgen, die dieselbe Funktion mit dem entsprechenden Parameter aufrufen.

Die Verarbeitung des XML-Dokuments `terminologie.xml` mittels AJAX soll nun im folgenden Abschnitt beschrieben werden.

### 7.3.2 Verarbeitung der XML-Daten

Mit Hilfe einer AJAX-Anwendung sollen die Daten aus der XML-Datei auf der HTML-Seite ausgegeben werden. Dazu wird zunächst ein Platzhalter im `body`-Element benötigt. Um die Daten später strukturiert ausgeben zu können, soll dies in Form einer Tabelle erfolgen. Auf diese kann anhand der ID „Trefffer“ zugegriffen werden.

```
<div id="content">
  <table id="Trefffer">

  </table>
</div>
```

Da der Internet Explorer dynamisch erzeugte Tabellen nur in einem `tbody`-Bereich korrekt darstellen kann, wird dieses später dynamisch vom JavaScript-Code erzeugt. Die einzelnen Zellen werden dann in einem `tbody`-Element ausgegeben. (Wenz 2006, S. 34)

Die Abfrage soll immer dann erfolgen, wenn entweder ein Suchbegriff in das Formularfeld eingegeben oder ein Link der vertikalen Navigation ausgewählt wird. Die eingegebenen bzw. ausgewählten Werte werden mit der Funktion `getData()` geholt.

```
<a href="javascript:getData('extruder',true)>Extruder</a>
```

Bei der Suchfunktion wird diese Funktion erst aufgerufen, wenn der Suchbegriff mindestens drei Buchstaben hat, damit eine sinnvolle Suche erfolgen kann.

```
<form action="" onsubmit="return false">
  <input type="text" name="eingabe"
    onkeyup="if(this.value.length >= 3)
      getData(this.value,false)"
    value="Suchbegriff" onclick="this.value='' " />
</form>
```

In der Funktion `getData()` erfolgt die Instanziierung des `XMLHttpRequest`-Objekts. Dieses wird hier der Variable `XMLHTTP` zugeordnet. Bevor nun die Anfrage gestartet wird, wird zunächst überprüft, ob das Objekt wirklich erzeugt wurde. Dazu wird mit einer einfachen `if`-Anweisung der Wert der Variablen `XMLHTTP` überprüft. Sollte ein Fehler aufgetreten sein, hat diese den Wert `null`. Wurde das Objekt erzeugt, wird die Anfrage erstellt und gesendet.

```
var XMLHTTP=null, eingabe, attr=false;

function getData(val, at)
{
  attr=at;
  eingabe=val.toLowerCase();

  if (window.ActiveXObject)
    XMLHTTP=new ActiveXObject("Microsoft.XMLHTTP");
  else if (window.XMLHttpRequest)
    XMLHTTP=new XMLHttpRequest();

  if(XMLHTTP)
  {
    XMLHTTP.onreadystatechange=callback;
    XMLHTTP.open("GET", "terminologie.xml", true);
    XMLHTTP.send(null);
  }
}
```

Die weitere Verarbeitung erfolgt in der Funktion `callback()`. Diese soll erst gestartet werden, wenn die Anfrage erfolgreich war, weshalb zunächst überprüft wird, ob der Status den Wert `4` hat.

```
function callback()
{
var xml,tbody,begriff,ergebnisse,zeile,deutsch,englisch,alt,
    knoten,termDeutsch,termEnglisch,termAlt,i,j,navi,test;

    if(XMLHTTP.readyState==4)
    {
        ...
    }
}
```

Im Internet Explorer gibt es bei der Ausführung von AJAX-Anwendungen Probleme, wenn der Zugriff nicht auf einen Server, sondern lokal, d.h. über `file://...`, erfolgt. Der folgende Code ermöglicht in diesem Fall die Nutzung von `responseXML`.

```
if(!XMLHTTP.responseXML.documentElement &&
XMLHTTP.responseStream)
XMLHTTP.responseXML.load(XMLHTTP.responseStream);
{
}
```

Über den `GET`-Request wird der Inhalt der XML-Datei `terminologie.xml` zurückgeliefert. Dieser wird nun mit der Eigenschaft `responseXML` zur Weiterverarbeitung an die Variable `xml` übergeben.

```
xml = XMLHttpRequest.responseXML;
```

Da der Wert von `responseXML` bereits ein XML-DOM-Dokument ist, kann es direkt über das DOM weiterverarbeitet werden. Dabei kann man entweder das gesamte XML-Dokument durchlaufen oder direkt auf bekannte Knoten zugreifen. (Wenz 2006, S. 42 f.)

Als erstes muss der Tabelle ein `tbody`-Bereich hinzugefügt werden. Dies geschieht mit Hilfe der Methode `createElement()` des `document`-Objekts. Des Weiteren erhält das Element mit `setAttribute()` ein ID-Attribut mit dem Wert `inhalt`.

```
tbody = document.createElement("tbody");
tbody.setAttribute("id", "inhalt");
```

Anhand dieses Attributs kann nun überprüft werden, ob sich bereits Daten in der Tabelle befinden. Wenn das der Fall ist, soll der Tabelleninhalt gelöscht werden, sobald ein neuer Request gestartet wird. Würde man das nicht machen, würden alle neuen Ergebnisse immer wieder an die bereits angezeigten angehängen werden.

Zunächst wird der Variablen `treffer` das `table`-Element über die ID zugewiesen. Eine `if`-Anweisung überprüft jetzt, ob das Dokument ein Element mit der ID `inhalt`, also das `tbody`-Element, besitzt. Ist diese Bedingung erfüllt, wird das Element mit der Methode `removeChild()` entfernt.

```
treffer = document.getElementById("Treffer");

if(document.getElementById("inhalt"))
{
    treffer.removeChild(document.getElementById("inhalt"));
}
```

Jetzt kann die neue Ausgabe beginnen. Wenn eine Antwort erfolgt, werden alle `term`-Elemente des XML-Dokuments durchlaufen.

```
if(xml)
{
  ergebnisse = xml.getElementsByTagName("term");
  for(i=0;i<ergebnisse.length;i++)
  {
    begriff = ergebnisse[i];
```

Um später überprüfen zu können, welche Sprache der Nutzer gewählt hat, wird jetzt eine Variable `test` festgelegt, der der Wert des Attributs `name` des Sprachauswahl-Links übergeben wird. Auf den Link wird über die DOM-Methode `getElementById` zugegriffen. Der Wert des Attributs `name` wird über die Methode `getAttribute` zurück geliefert.

```
links = document.getElementById("link");
test = links.getAttribute("name");
```

Der Wert kann entweder `deut` oder `engl` sein und wird beim Klicken der Links über folgende Funktionen geändert:

```
function Attribut()
{
  var links = document.getElementById("link");
  links.setAttribute("name", "deut");
}

function Attribut2()
{
  var links = document.getElementById("link");
  links.setAttribute("name", "engl");
}
```

Über die Methode `setAttribute` wird dem Element dynamisch ein neues Attribut gesetzt. Dazu werden der Attributname sowie der entsprechende Wert als Argumente übergeben. Der bereits bestehende Wert des Attributs wird dabei überschrieben. (Gamperl 2006, S. 53)

## Abruf der Daten über Links

Nun soll die Abfrage für eine Auswahl über die linke Navigationsleiste erfolgen. Die Verarbeitung soll also starten, wenn die Variable `attr` den Wert `true` hat.

```
if(attr)
{
```

Zunächst werden die Inhalte der XML-Elemente ermittelt, die mit dem eingegebenen Suchbegriff verglichen werden sollen.

```
termDeutsch =
document.createTextNode(begriff.getElementsByTagName
("deutsch").item(0).firstChild.nodeValue);

termEnglisch =
document.createTextNode(begriff.getElementsByTagName ("eng-
lisch").item(0).firstChild.nodeValue);
```

Je nach Sprachauswahl variiert die weitere Verarbeitung. Ist die deutsche Sprache gewählt, sollen die deutschen Begriffe an erster Stelle ausgegeben werden. Bei der englischen Sprache genau umgekehrt. Deshalb wird zunächst überprüft, welchen Wert das Attribut `test` hat.

```
if(test == "deut")
{
```

Hat das Attribut den Wert `deut`, ist die deutsche Sprache gewählt und der deutsche Begriff soll in die erste Tabellenzelle.

Nun wird überprüft, ob das Attribut `typ` der gefundenen Elemente mit dem vom Link übergebenen Parameter übereinstimmt.

```
if(begriff.getAttribute("typ") == eingabe)
{
```

Anhand der gefundenen Übereinstimmungen werden nun die Textknoten erstellt, die in die Tabelle eingefügt werden sollen.

```
for (j=0; j<ergebnisse[i].childNodes.length; j++)
{
  knoten = ergebnisse[i].childNodes[j];
  switch (knoten.nodeName)
  {
    case "deutsch":
      termDeutsch = knoten.firstChild.nodeValue;
      break;
    case "englisch":
      termEnglisch = knoten.firstChild.nodeValue;
      break;
  }
}
```

Um nachher die gefundenen Benennungen in alphabetischer Reihenfolge ausgeben zu können, wird hier das Array `term_arr` verwendet. Arrays sind Variablencontainer, die mehrere Variablen beinhalten können. Zunächst wird die Array-Variable deklariert.

```
var term_arr=[];
```

Diesem Array werden nun die gefundenen Textknoten übergeben. Um die Zuordnung der zusammengehörenden deutschen und englischen Benennungen beizubehalten, werden diese getrennt durch einen Vertikalstrich (|) als Zeichenkette zusammengefügt.

```
term_arr[k] = termDeutsch+"|"+termEnglisch;
k++;
}
```

Ist die englische Sprache gewählt, erfolgt nun dieselbe Prozedur, mit dem Unterschied, dass dem Array die Textknoten in einer anderen Reihenfolge zugeordnet werden.

```
else if(test == "engl")
{
  if(begriff.getAttribute("typ") == eingabe)
  {
    for (j=0; j<ergebnisse[i].childNodes.length; j++)
    {
      knoten = ergebnisse[i].childNodes[j];
      switch (knoten.nodeName)
      {
        case "deutsch":
          termDeutsch = knoten.firstChild.nodeValue;
          break;
        case "englisch":
          termEnglisch = knoten.firstChild.nodeValue;
          break;
      }
    }
    term_arr[k] = termEnglisch+"|"+termDeutsch;
    k++;
  }
}
```

### Abruf der Daten über die Suchfunktion

Die Vorgehensweise für die Abfrage der über die Suchfunktion eingegebenen Begriffe ist ähnlich. Eine `if`-Anweisung überprüft zunächst, ob die Variable `attr` den Wert `false` hat. Ist dies der Fall, wurde die Anfrage nicht über die Links sondern über das Formularfeld gestartet.

```
else if(!attr)
{
```

Nun werden auch hier die Inhalte der XML-Elemente ermittelt, die mit dem eingegebenen Suchbegriff verglichen werden sollen. Die Methode `toLowerCase()` wandelt die erhaltene Zeichenkette in Klein-

buchstaben um. So können die Daten unabhängig von Groß- und Kleinschreibung miteinander verglichen werden.

```
termDeutsch = begriff.getElementsByTagName("deutsch").
    item(0).firstChild.nodeValue.toLowerCase();
termEnglisch = begriff.getElementsByTagName("englisch").
    item(0).firstChild.nodeValue.toLowerCase();
termAlt = begriff.getElementsByTagName("alt").item(0).
    firstChild.nodeValue.toLowerCase();
```

An dieser Stelle treten jedoch Probleme auf, wenn ein `term`-Element im XML-Dokument kein Kindelement `alt` besitzt bzw. dieses leer ist. In diesem Fall kann kein Elementinhalt ermittelt werden. Aus diesem Grund wurde allen Einträgen das Element `alt` mit einem festen Leerzeichen (`&#160;`) als Inhalt hinzugefügt. Da das Element immer vorhanden sein muss, wurde die entsprechende Kennzeichnung in der DTD vorgenommen.

Ist nun als Sprache deutsch gewählt, wird der eingegebene Suchbegriff mit den ermittelten Zeichenketten `termDeutsch` und `termAlt` verglichen. Dabei kommt die Methode `match()` zum Einsatz, die die Zeichenkette auf den Suchbegriff hin durchsucht.

```
if(test == "deut")
{
    if(termDeutsch.match(eingabe) || termAlt.match(eingabe))
    {
```

Die Erstellung der Textknoten sowie deren Zuordnung zu dem Array `term_arr` erfolgt entsprechend der Abfrage über die Navigation.

```
for (j=0; j<ergebnisse[i].childNodes.length; j++)
{
    knoten = ergebnisse[i].childNodes[j];
    switch (knoten.nodeName)
    {
        case "deutsch":
            termDeutsch = knoten.firstChild.nodeValue;
            break;
        case "englisch":
            termEnglisch = knoten.firstChild.nodeValue;
            break;
    }
    term_arr[k] = termDeutsch+"|"+termEnglisch;
    k++;
}
}
```

Ist als Sprache englisch gewählt, wird der eingegebene Suchbegriff nicht mit den Zeichenketten `termDeutsch` und `termAlt`, sondern mit `termEnglisch` verglichen. Ebenfalls werden hier die gefundenen Textknoten in umgekehrter Reihenfolge dem Array `term_arr` zugeordnet.

```
else if (test=="engl")
{
    if(termEnglisch.match(eingabe))
    {
        ...
    }
    term_arr[k] = termEnglisch+"|"+termDeutsch;
    k++;
}
```

## Ausgeben der Daten

Bevor die Daten in der Tabelle ausgegeben werden können, müssen sie alphabetisch sortiert werden. Das Array-Objekt stellt dazu die Methode `sort()` zur Verfügung.

```
term_arr.sort();
```

Die in dem Array gespeicherten Zeichenketten müssen nun wieder nach den deutschen und englischen Benennungen geteilt werden. Das `string`-Objekt besitzt dafür die Methode `split()`. Diese teilt die im Array zurückgegebenen Zeichenketten anhand des angegebenen Trennzeichens (`|`) in einzelne Zeichenketten, die der Variablen `termtext` zugewiesen werden.

```
for(i=0; i<term_arr.length; i++)  
{  
    termtext = term_arr[i].split("|");
```

Die Benennungen liegen jetzt wieder einzeln vor und können in die Tabelle eingefügt werden. Als erstes wird dafür eine Tabellenzeile (`<tr>`) erstellt. Daraufhin werden jeweils für den deutschen und englischen Begriff Tabellenzellen (`<td>`) eingefügt, an die die bereits erzeugten Textknoten angehängt werden.

```
zeile = document.createElement("tr");  
  
deutsch = document.createElement("td");  
deutsch.appendChild(document.createTextNode(termtext[1]));  
  
englisch = document.createElement("td");  
englisch.appendChild(document.createTextNode(termtext[0]));
```

Zu guter Letzt werden die Zellen an die Zeile und die Zeile an die Tabelle angehängt. Abschließend wird noch der `tbody`-Bereich an die Tabelle angehängt.

```
zeile.appendChild(englisch);
zeile.appendChild(deutsch);
tbody.appendChild(zeile);
}
treffer.appendChild(tbody);
```

## Glossar

Die Abfrage der Daten über das Glossar erfolgt analog der Abfrage über die Links. Da hier aber nicht ein kompletter Eintrag, sondern nur der Anfangsbuchstabe verglichen werden soll, wird über die Methode `substring()` der erste Buchstabe des Wortes ermittelt. Als Parameter werden die Position des ersten gewünschten Zeichens (0) sowie die Position des ersten Zeichens, das nicht mehr dazu gehören soll (1) übergeben.

```
termDeutsch = begriff.getElementsByTagName("deutsch").
item(0).firstChild.nodeValue.toLowerCase();

termEnglisch = begriff.getElementsByTagName("englisch").
item(0).firstChild.nodeValue.toLowerCase();

firstLetterD = termDeutsch.substring(0,1);
firstLetterE = termEnglisch.substring(0,1);

if(test == "deu")
{
if(attr && (firstLetterD == eingabe))
...
}
```

## **8 Evaluation der beiden Varianten**

In den beiden vorangegangenen Kapiteln wurde gezeigt, dass ein XML-Dokument mit verschiedenen Ansätzen verarbeitet werden kann. Dieses Kapitel soll nun die entwickelten Anwendungen vergleichen, bewerten und hinsichtlich ihrer Benutzerfreundlichkeit beurteilen.

### **8.1 Vergleich der Applikationen**

Auf den ersten Blick stellt sich für den Benutzer kein Unterschied zwischen den beiden verschiedenen Anwendungen dar. Aufgrund des exakt gleichen Layouts und der identischen Funktionen sehen die beiden Webseiten absolut gleich aus.

Die Realisierung der XSLT-Variante mit Hilfe von Frames sorgt zusätzlich dafür, dass sich die Anwendungen gleichen. Da sich hier nur der Ausgabe-Frame ändert, wenn eine Aktion durchgeführt wird, hat der Benutzer das Gefühl, dass sich der Inhalt dynamisch ändert.

Der Unterschied wird erst bei der Suchfunktion deutlich. Während man bei der XSLT-Version die Eingabe entweder per Klick auf den Submit-Button oder per Enter-Taste abschicken muss, geschieht dies bei der AJAX-Anwendung automatisch. Ab dem dritten eingegebenen Buchstaben wird hier die Anfrage gestartet. Mit jedem weiteren Buchstaben, der eingegeben wird, beginnt eine neue Anfrage.

Für den Benutzer hat das den Vorteil, dass er nicht den vollständigen Begriff eingeben muss, sondern eventuell schon nach dem vierten Buchstaben das gewünschte Ergebnis angezeigt bekommt. Sollte

das nicht der Fall sein, gibt er den nächsten Buchstaben ein und die Ergebnisliste wird dynamisch geändert. Dadurch wird eine effizientere und schnellere Arbeitsweise ermöglicht.

Außerdem ermöglicht es dem Benutzer auch Begriffe zu finden, von denen er nicht die exakte Bezeichnung kennt. Anhand des Wortstamms werden während der Eingabe bereits mögliche Übereinstimmungen angezeigt, die ihm eventuell von Nutzen sein können. Bei der XSLT-Version würde der Benutzer aller Voraussicht nach erst den kompletten Begriff eingeben und dann absenden. Existiert der Begriff nicht exakt so in der XML-Datei, wird er kein Ergebnis angezeigt bekommen. Es besteht zwar auch hier die Möglichkeit, nach Übereinstimmungen von Teilzeichenketten zu suchen, jedoch müsste der Nutzer nach jedem eingegebenen Buchstaben die Eingabe absenden, um einen ähnlichen Effekt zu erreichen.

### **8.1.1 Vor- und Nachteile von XSLT**

XSLT ist eine leistungsfähige, flexible Sprache zum Transformieren von XML-Dokumenten. Sie ist so angelegt, dass viele verschiedene Stylesheet-Regeln gleichzeitig angewendet werden können. (Tidwell 2003, S. 2 f.) Zudem ist XSLT auch plattformunabhängig und kann somit unabhängig vom Betriebssystem eingesetzt werden.

Ein Vorteil, den der Einsatz von XSLT bietet, ist, dass die Daten in einem flexiblen Format vorliegen. Ihr Aussehen kann daher jederzeit beliebig verändert werden. (Tidwell 2003, S. 4) Auch Änderungen am Aussehen der Webseite selbst können ohne große Probleme durchgeführt werden. Da sich die Transformationsanweisungen in einer separaten Datei befinden, kann das Layout der Hauptseite einfach verändert werden, ohne dass die eigentliche Transformation davon betroffen ist.

XSLT bietet die Möglichkeit, XML-Daten in jedes beliebige Format umzuwandeln. Das kann zum Beispiel ein HTML-Dokument für die Darstellung im Web oder auch ein neues XML-Dokument für den Datenaustausch sein. Dadurch bietet sich ein mächtiges Werkzeug zum Manipulieren und Formatieren von Daten. (Ammelburger 2004, S. 30)

Die Tatsache, dass XSLT eine eigenständige Programmiersprache ist, kann aber auch einen Nachteil darstellen. Soll XSLT zum Einsatz kommen, müssen grundlegende Kenntnisse darüber vorhanden sein bzw. zunächst erlernt werden.

### **8.1.2 Vor- und Nachteile von AJAX**

Ein Vorteil, der für den Einsatz von AJAX spricht, ist die Verwendung bereits bekannter Technologien. AJAX ist keine eigene Programmiersprache, die zunächst erlernt werden müsste. Mit Kenntnissen in HTML, JavaScript und DOM kann man die Funktionsweise von AJAX schnell nachvollziehen.

Da die Inhalte einer AJAX-Anwendung dynamisch nachgeladen werden, verkürzen sich die Ladezeiten. Der Nutzer bekommt schneller eine Reaktion auf seine ausgeführten Aktionen als bei herkömmlichen Anwendungen. (Carl 2006, S. 4) Dadurch wird gleichzeitig die Usability erhöht.

Aber der Einsatz von AJAX bringt auch Nachteile mit sich. Einer ist die fehlende Vor- und Zurück-Funktionalität. Die dynamischen AJAX-Rückgaben werden nicht in der Historie des Browsers gespeichert. Klickt man auf den Zurück-Button, erhält man nicht die Ergebnisse der letzten Abfrage, sondern gelangt auf die zuvor besuchte Seite, die in der Browser-Historie gespeichert wurde. (Gamperl 2006, S. 219)

Beim Einsatz einer AJAX-Anwendung im Internet kann es aufgrund von JavaScript zu Problemen kommen. Aus Sicherheitsgründen schalten einige Nutzer JavaScript in ihren Browsern aus. Die Funktionalität der Seite wäre in diesem Fall nicht gegeben. Die genaue Anzahl der Browser mit deaktiviertem JavaScript lässt sich nicht genau ermitteln. Im Internet findet man verschiedene Statistiken, wie beispielsweise unter [www.w3schools.com](http://www.w3schools.com), die besagt, dass rund 10 % der Browser kein JavaScript aktiviert haben bzw. nicht unterstützen. ([www.w3schools.com](http://www.w3schools.com))

Die Daten, die diese Statistiken liefern, geben jedoch auch keinen Aufschluss darüber, welche Einstellungen die Nutzer einer bestimmten Webseite in ihren Browser vornehmen. Jede Seite spricht eine eigene Zielgruppe an, die ganz unterschiedliche Voraussetzungen mit sich bringen kann.

Im Fall der Terminologie-Anwendung spielt diese Problematik jedoch keine Rolle, da diese nur im Intranet veröffentlicht wird. Die Nutzer sind somit bekannt und an den Arbeitsplätzen können entsprechende Einstellungen vorgenommen werden.

## **8.2 Usability-Test**

Der Begriff Usability bezeichnet die Benutzerfreundlichkeit einer Webseite. Eine Webseite sollte so gestaltet sein, dass sich der Benutzer auf ihr leicht zurechtfindet. Das bedeutet, dass die Seiten gut lesbar, Buttons und Links erkennbar und die Navigation verständlich sein müssen. (Jacobsen 2002, S. 215 f.)

Auch wenn die Usability bei der Planung einer Seite berücksichtigt wurde, können für den Benutzer Probleme entstehen. Um diese

zuverlässig aufzudecken, ist die Durchführung eines Usability-Tests unerlässlich. Mit diesem wird getestet, ob der Benutzer sich auf der Seite zurechtfindet. (Jacobsen 2002, S. 216) Dazu muss die Testperson mit dieser Seite Aufgaben lösen. Anhand der Ergebnisse und Reaktionen kann später die Usability bewertet werden.

Um ein aussagekräftiges Ergebnis zu erzielen, müssen mindestens drei bis fünf Personen getestet werden. Sollte es schwerwiegende Probleme geben, werden diese sofort bei der ersten Testperson auftreten. Die Testperson sollte idealer Weise ein Vertreter der Zielgruppe sein. (Jacobsen 2002, S. 221 f.)

### **8.2.1 Die Testaufgaben**

In der Praxis müssen die Nutzer schnell bestimmte Begriffe in der Terminologie-Anwendung auffinden können. Der Usability-Test soll aufzeigen, ob das funktioniert. Aus diesem Grund werden den Testpersonen Aufgaben gestellt, in denen sie beispielsweise einen Begriff über das Glossar finden sollen.

Der Test beginnt mit einer Begrüßung der Testperson, in der ihr erklärt wird, worum es bei dem Test geht. Als Einstieg folgen nun ein paar einfache Aufgaben, die ohne Probleme beantwortet werden können. Das hat den Zweck, der Testperson Sicherheit zu geben. (Jacobsen 2002, S. 224)

Zunächst werden nur Aufgaben und Fragen zu der XSLT-Anwendung gestellt. Sie sollen die Funktionalität der Seite beurteilen. Im Anschluss daran soll die Testperson sich mit der AJAX-Anwendung befassen. Dabei geht es darum, Unterschiede festzustellen und zu beurteilen, welche Version besser ist. Der komplette Test ist in Anhang C zu finden.

## 8.2.2 Testauswertung und Korrektur

Der Usability-Test wurde mit drei Testpersonen durchgeführt. Es zeigte sich, dass alle Teilnehmer sehr gut mit der Bedienung der Webseite zurechtgekommen sind. Sie konnten sich sofort vorstellen, was sich hinter den einzelnen Navigationspunkten verbirgt. Dementsprechend führten sie die gestellten Aufgaben ohne Schwierigkeiten aus.

Die Lesbarkeit der einzelnen Seiten-Bereiche wurde wie folgt bewertet:

	Test 1	Test 2	Test 3
Obere Navigationsleiste	1	1	2
Linke Navigationsleiste	2	2	1
Mittleres Textfeld	2	1	2

Zwei Personen gaben an, dass der Abstand zwischen den Links in der linken Navigationsleiste zu klein ist und dadurch die Lesbarkeit erschwert wird. Als Konsequenz wurde der Abstand erhöht.

Ebenfalls zwei Teilnehmer bewerteten die Lesbarkeit des mittleren Textfeldes mit der Note 2. Die Bewertung erfolgte anhand der XSLT-Variante der Terminologie-Anwendung, in der die deutschen und englischen Benennungen direkt aneinander gereiht werden (Abbildung 11).

<b>Quer-Werkzeug</b> cross head
<b>Rheometer-Werkzeug</b> rheometer head
<b>Schlauchschlitz-Werkzeug</b> slit tube head
<b>Strainer-Werkzeug</b> strainer head
<b>Triplex-Extrusionswerkzeug</b> triplex extrusion head

Abb. 11: Ausgabe der Suchergebnisse in der XSLT-Anwendung

Beide Teilnehmer gaben hier an, dass eine Anordnung der Benennungen in Tabellenform die Lesbarkeit erhöhen würde. Die AJAX-Anwendung, in der die Ausgabe bereits in einer Tabelle realisiert wurde, empfanden die Testpersonen als angenehmer. Im Anschluss an den Test wurde die Formatierung auch für die XSLT-Version übernommen.

Beim Vergleich der Suchfunktionen der beiden Webseiten stellten alle Testpersonen Unterschiede fest. Nachdem sie eine Weile probiert haben, entschieden sich alle drei bei der Frage, welche Version ihnen besser gefällt, für die AJAX-Anwendung, die folglich in der rubicon GmbH zum Einsatz kommen soll.

Einen weiterer Verbesserungsvorschlag war, die linke Navigationsleiste alphabetisch zu sortieren, da das ein schnelleres Finden der Sachgebiete ermögliche. Da auf Nachfrage auch die anderen Testpersonen diese Anordnung als sinnvoll empfanden, wurde der Vorschlag umgesetzt.

## 9 Einsatz der Terminologie-Anwendung

Nachdem in den vorangegangenen Kapiteln die Erstellung der Terminologie-Anwendung im Vordergrund stand, soll nun auf deren Einsatz in der Praxis eingegangen werden. Zunächst soll die Benutzung aus Sicht des Anwenders erklärt werden. Der darauf folgende Abschnitt beschäftigt sich mit der Bearbeitung der XML-Daten.

### 9.1 Benutzung

Die Terminologie-Anwendung hat die Funktion eines Wörterbuches. Sie soll es dem Benutzer ermöglichen, nach deutschen bzw. englischen Benennungen zu suchen und die dazu entsprechende Übersetzung zu finden.

Das Öffnen der Anwendung erfolgt über die Datei `index.html`. Auf der Seite hat der Nutzer drei Möglichkeiten, den gesuchten Begriff zu finden:

- durch Auswählen eines Sachgebietes über die linke Navigation
- durch Suchen eines bestimmten Begriffs über die Suchfunktion
- durch Nachschlagen der Einträge über das Glossar

Beim Start der Webseite wird das Registerblatt *Suche* angezeigt, auf dem sich die Navigation und die Suchfunktion befinden. Durch Klicken des Registerblatts *Glossar* gelangt der Nutzer zum Glossar.

### 9.1.1 Navigation

Mit Hilfe der vertikalen Navigation auf der Startseite können Begriffe bezüglich ihres Sachgebietes angezeigt werden. Der Nutzer hat dabei die Auswahl zwischen folgenden Gebieten:

- Abzugsbänder
- Extruder
- Infrarot
- Kühlanlagen
- Linien
- Mikrowelle
- Salzbad
- Walzwerke
- Werkzeug
- Wickler
- Zahnradpumpen
- Zubehör

Durch Klicken des entsprechenden Links werden alle Einträge, die diesem Gebiet zugeordnet sind, in alphabetischer Reihenfolge angezeigt.

### 9.1.2 Suchfunktion

Die Suchfunktion befindet sich ebenfalls auf der Startseite. Über ein Formularfeld kann der Benutzer den gesuchten Begriff eingeben. Die Groß- und Kleinschreibung ist dabei nebensächlich, da die Schreibweise bei der Suche nicht berücksichtigt wird. Es können ebenfalls nur Wortfragmente eingegeben werden.

Ausgegeben werden alle Begriffe, die das eingegebene Wort oder den eingegebenen Wortteil an beliebiger Stelle enthalten. So ergibt zum Beispiel der Suchbegriff „extr“ folgende Ergebnisse:

Co**extr**usionswerkzeug  
**Ex**truder  
Geradeaus-Co**extr**usionswerkzeug  
rubicon - Hybrid-Labore**extr**uder, kaltbeschickt  
rubicon - Labore**extr**uder, kaltbeschickt  
rubicon - Silikon**extr**uder, kaltbeschickt  
rubicon - Standard**extr**uder, kaltbeschickt  
rubicon - Standard**extr**uder, warmbeschickt  
rubicon - Stift**extr**uder, kaltbeschickt  
rubicon - Vakuum**extr**uder, kaltbeschickt  
Triplex-**Ex**trusionswerkzeug

Je genauer der Suchbegriff ist, desto eindeutiger wird demzufolge das Suchergebnis sein.

Bei der XSLT-Variante wird die Suche durch Klicken des *finden*-Buttons gestartet. Die Abfrage der Suche bei der AJAX-Anwendung erfolgt dynamisch während der Eingabe.

### 9.1.3 Glossar

Zum Glossar gelangt man durch Klicken der Registerkarte Glossar. Hier können nun alle Begriffe anhand ihres Anfangsbuchstabens aufgelistet werden. Über Links werden die jeweiligen Buchstaben aufgerufen.

Dabei kann auch wieder über die Sprachbuttons die anzuzeigende Sprache ausgewählt werden. Beim Start der Seite ist immer die Sprache Deutsch aktiv.

## 9.2 Terminologiepflege

Im Laufe des Einsatzes der Terminologie-Datenbank wird es hin und wieder nötig sein, neue Begriffe in die XML-Datei einzufügen oder bestehende zu verändern. Diese Arbeiten müssen in der Datei `terminologie.xml` vorgenommen werden. Mit entsprechenden XML-Kenntnissen kann das Dokument in jedem beliebigen Texteditor bearbeitet werden.

Für den Fall, dass keine XML-Kenntnisse vorliegen, kann der Einsatz eines XML-Editors die Arbeit erleichtern. Im Gegensatz zu einfachen Texteditoren bieten XML-Editoren zusätzliche Eingabe- und Verarbeitungsmöglichkeiten. Dazu gehört beispielsweise auch die Überprüfung des Dokuments auf Wohlgeformtheit. ([www.teialehrbuch.de](http://www.teialehrbuch.de))

Es gibt eine Vielzahl an XML-Editoren, von denen eine Auswahl in den folgenden Abschnitten vorgestellt werden soll. Zusätzlich dazu soll die Bearbeitungsmöglichkeit mit Microsoft Excel vorgestellt werden.

### 9.2.1 XMLSpy Professional Edition Version 2007

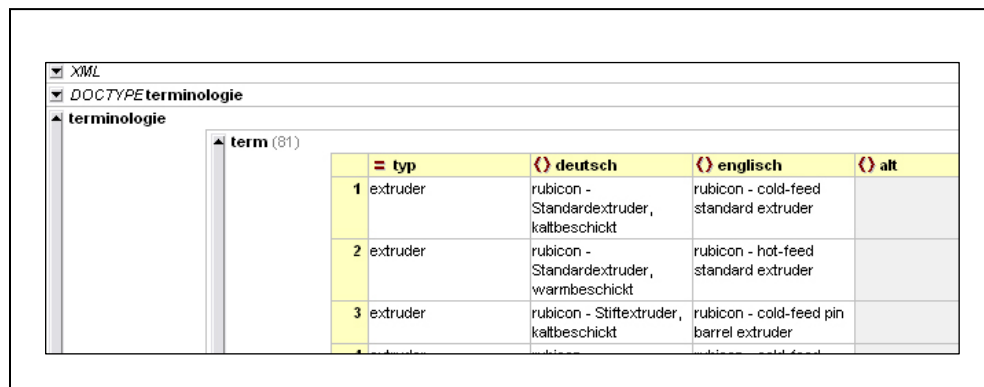
Einer der führenden XML-Editoren ist XMLSpy von Altova. Aufgrund seiner umfangreichen Funktionen findet er im professionellen Anwendungsgebiet Einsatz. Neben dem XML-Editor umfasst XMLSpy viele weitere Komponenten, wie zum Beispiel einen XSLT-Editor und -Prozessor. ([www.teialehrbuch.de](http://www.teialehrbuch.de))

XMLSpy ist sehr flexibel. Je nach Bedarf und Kenntnissen können geeignete Ansichten oder Optionen verwendet werden, um die Arbeit mit XML zu vereinfachen. Soll ein Dokument ohne grundlegende XML-Kenntnisse bearbeitet werden, empfiehlt sich die Arbeit in der

Grid-Ansicht. In dieser wird das Dokument in einer Tabellenform angezeigt. So kann der Inhalt einfach bearbeitet werden.

## Bearbeitung

Nach dem Öffnen der Datei in XMLSpy gelangt man durch Klicken der Schaltfläche *Grid* am unteren Rand des Dokumentenfensters in die Grid-Ansicht. (Abbildung 12)



	= typ	<> deutsch	<> englisch	<> alt
1	extruder	rubicon - Standardextruder, kaltbeschickt	rubicon - cold-feed standard extruder	
2	extruder	rubicon - Standardextruder, warmbeschickt	rubicon - hot-feed standard extruder	
3	extruder	rubicon - Stiftruder, kaltbeschickt	rubicon - cold-feed pin barrel extruder	

Abb. 12: Grid-Ansicht in XMLSpy

Die hierarchische Struktur des Dokuments wird nun verschachtelt dargestellt. Durch Klicken des kleinen schwarzen Pfeils können die einzelnen Ebenen ein- oder ausgeblendet werden. Die Attribut- und Elementinhalte werden in einer Tabelle dargestellt. Im Tabellenkopf werden Attribute durch das Symbol = und Elemente durch das Symbol <> gekennzeichnet.

Diese Ansicht hat für den Benutzer den Vorteil, dass er nicht mit dem Markup konfrontiert wird. Er sieht nur einen einfachen Text innerhalb der Tabellenzellen. Beim Editieren kann es nicht passieren, dass aus Versehen das Markup verändert wird.

Die bestehenden Einträge können nun einfach in den Zellen geändert werden. Zum Hinzufügen neuer Datensätze markiert man zunächst einen Eintrag durch Klicken auf dessen Nummer. Im Fenster Element (Abbildung 13) erscheint daraufhin auf dem Register Anhängen das entsprechende Element `term`. Das Einfügen eines weiteren Elements wird nun durch Doppelklicken des Elements `term` erreicht.

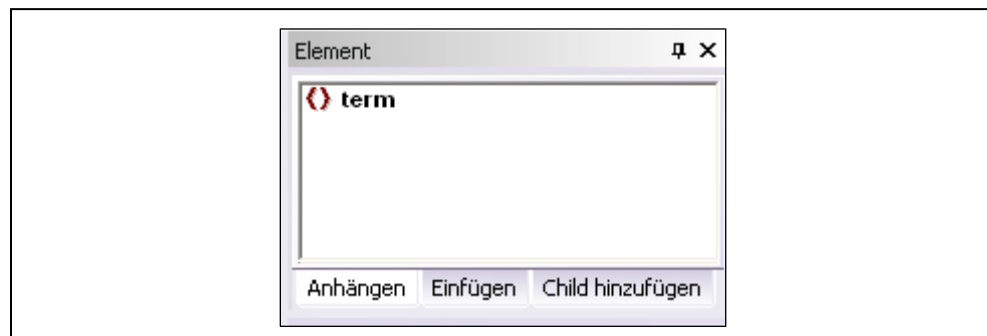




Abb. 13: Einfügen von Elementen in XMLSpy

Zwei weitere nützliche Funktionen von XMLSpy sind die Prüfung des Dokuments auf Wohlgeformtheit und das Validieren. Durch Klicken des Symbols  wird das Dokument nach der Definition der XML-Spezifikation auf seine Wohlgeformtheit überprüft. Über das Symbol  wird das XML-Dokument gegen die DTD validiert. Das Ergebnis beider Aktionen wird unterhalb des Hauptfensters angezeigt.

## Fazit

Die Grid-Ansicht bietet eine gute Möglichkeit, ein XML-Dokument ohne jegliche XML-Kenntnisse zu bearbeiten. Die einfache Bedienung gestattet dem Nutzer einen schnellen Einstieg in das Programm.

XMLSpy bietet etliche Funktionen, von denen allerdings nur die bereits genannten relevant für den Einsatz in der rubicon GmbH sind.

Der komplette Leistungsumfang des Programms würde mit der einfachen Editierung eines bereits bestehenden Dokuments nicht ausgenutzt werden. Die hohen Anschaffungskosten wären daher nicht gerechtfertigt.

### 9.2.2 XMLmind Standard Edition 3.4.0

Als nächstes soll der XMLmind-Editor näher betrachtet werden. Dieser ist in der Standard Edition kostenlos. Das Programm stellt das XML-Dokument in einer Baumansicht dar. (Abbildung 14)

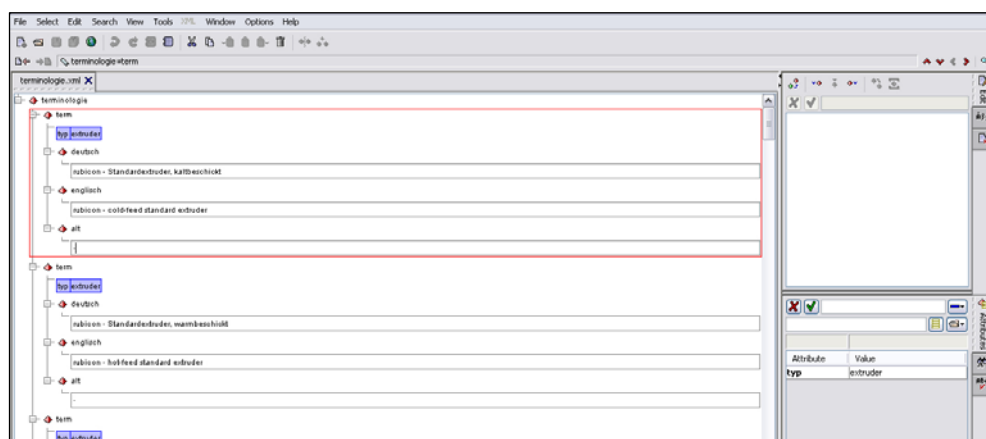


Abb. 14: XMLmind

### Bearbeitung

Änderungen an den Einträgen können einfach in den entsprechenden Feldern vorgenommen werden. Der Vorteil ist, dass der Benutzer zwar den Elementinhalt bearbeiten kann, nicht aber die Elementnamen. Das verhindert ein versehentliches Verändern des Markups.

Das Einfügen neuer Daten ist in XMLmind hingegen etwas komplizierter. Zunächst muss das letzte Element `term` ausgewählt werden. Die Auswahl wird durch einen roten Rahmen markiert. Über das Edit-Fenster, das sich rechts neben dem Dokumentenfenster befindet,

kann nun über Symbole ausgewählt werden, wo ein neues Element eingefügt werden soll. Durch Klicken des Symbols "insert after" werden die Elemente, die an dieser Stelle eingefügt werden können, angezeigt. Das Einfügen erfolgt durch einen Klick auf das grüne Häkchen. (Abbildung 15)



Abb. 15: Einfügen von Elementen in XMLmind

Das erstellte Element `term` besitzt das Attribut `typ`, dem aber noch kein Wert zugewiesen ist. Um dies zu tun, muss das Attribut im Fenster *Attributes* ausgewählt werden. Dadurch erscheint der Attributwert in der darüber liegenden Zeile, in der er bearbeitet werden kann. Durch das Bestätigen über das grüne Häkchen wird der eingegebene Wert dem Attribut zugeordnet.

XMLmind verfügt ebenfalls über eine Funktion zum Überprüfen der Validität. Über das Menü *Tools/Validity* kann diese ausgewählt werden. Das Ergebnis wird unterhalb des Dokumentenfensters angezeigt.

## Fazit

XMLmind bietet alle Möglichkeiten, ein XML-Dokument ohne Vorkenntnisse zu bearbeiten. Im Vergleich zu XMLSpy ist die Bearbeitung hier jedoch umständlicher und erfordert damit wahrscheinlich

eine längere Einarbeitungszeit. Erschwerend kommt noch hinzu, dass das Programm auf Englisch ist und es somit zu Problemen bei der Bedienung kommen kann. Nimmt man diese kleine Erschwernis in Kauf, bietet XMLmind eine gute Alternative zu anderen, teuren Programmen.

### 9.2.3 XML Notepad 2007

Ein weiterer kostenloser Editor, der hier vorgestellt werden soll, ist XML Notepad 2007 von Microsoft. Abbildung 16 zeigt die Ansicht des XML-Dokuments in XML Notepad.

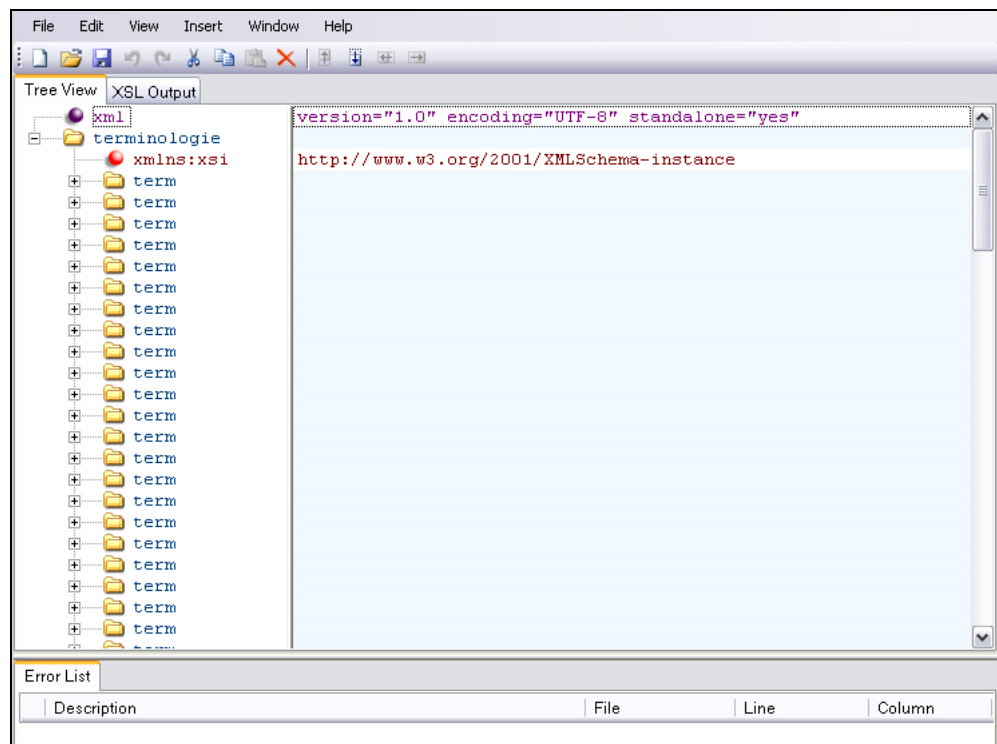


Abb. 16: XML Notepad

Die Struktur des Dokuments ist im linken Bereich in einer Bauman-sicht dargestellt. Diese Ansicht enthält die einzelnen Knoten, deren Kindknoten durch Klicken auf das Plus-Zeichen eingeblendet werden

können. Auf der rechten Seite werden Attributwerte und Elementinhalte angezeigt.

## Bearbeitung

Soll ein bestimmter Eintrag geändert werden, muss zunächst der entsprechende Knoten gefunden werden. Am schnellsten geschieht dies über die Suchfunktion, die mit der Tastenkombination „Strg + F“ aufgerufen wird. In dem sich öffnenden Suchfenster kann nun der gewünschte Suchbegriff eingegeben werden. Dieser kann sowohl die Benennung selbst als auch ein Attribut sein. Das Element mit der gefundenen Übereinstimmung wird nun aufgeklappt angezeigt und kann im rechten Fenster bearbeitet werden. (Abbildung 17)

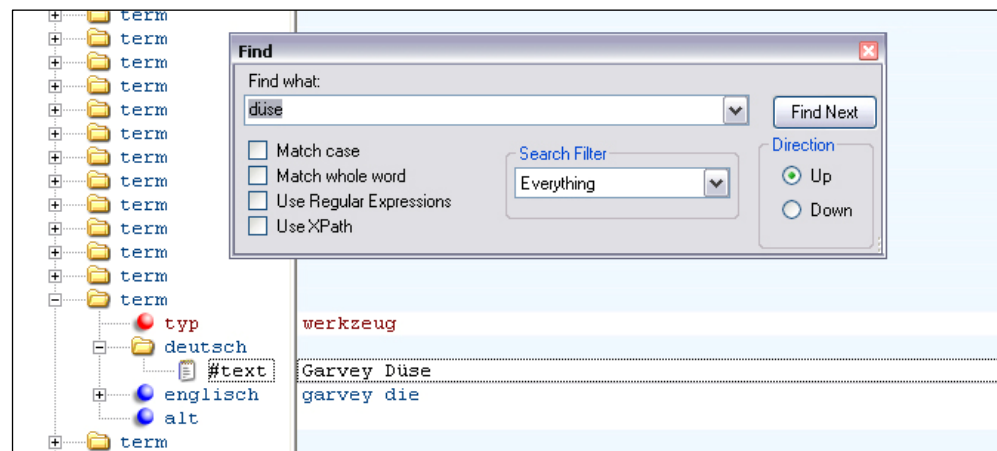


Abb. 17: Suche nach Einträgen in XML Notepad

Zum Einfügen neuer Daten können über das Menü *Insert* neue Elemente und Attribute angelegt werden. Dabei muss zunächst ein neues Element `term` und danach das Attribut `typ` sowie die Unter-elemente `deutsch`, `englisch` und `alt` einzeln erstellt werden.

Einfacher und schneller geht dies durch Duplizieren eines bereits vorhandenen `term`-Elements. Dazu wird zunächst das Element gewählt, das kopiert werden soll. Um die Übersicht zu behalten, ist es empfeh-

enswert, dafür das letzte Element zu wählen, da das Duplikat immer danach eingefügt wird. Dupliziert wird nun über das Menü `E-dit/Duplicate`. Jetzt müssen nur noch die Elementinhalte und der Attributwert geändert werden.

### **Fazit**

XML Notepad 2007 ist ein einfacher Editor, der sich gut zum schnellen Überarbeiten eines XML-Dokuments eignet. Durch die einfache, selbsterklärende Bedienung findet sich der Benutzer schnell zurecht und kann ohne lange Einarbeitungszeit mit dem Editieren beginnen.

Ein weiterer Vorteil ist, dass XML Notepad bereits während der Eingabe das Dokument validiert. Treten Fehler bei der Bearbeitung auf, werden diese in der *Error List* im unteren Fensterbereich angezeigt.

### **9.2.3 Microsoft Excel**

Zum einfachen Bearbeiten von XML-Dokumenten können oft auch bereits vorhandene Programme, die XML unterstützen, verwendet werden. Microsoft Office Professional Edition 2003 und Microsoft Office Excel 2003 stellen zum Beispiel XML-Funktionen zur Verfügung.

#### **Bearbeitung**

Zum Bearbeiten des XML-Dokuments muss dieses zunächst in Excel über das Menü Datei/Öffnen geöffnet werden. Daraufhin wird die Dialogbox „XML öffnen“ angezeigt. Durch das Aktivieren des Formats „Als eine XML-Liste“ (Abbildung 18) und Klicken auf „OK“ wird nun das Dokument als XML-Liste geöffnet.

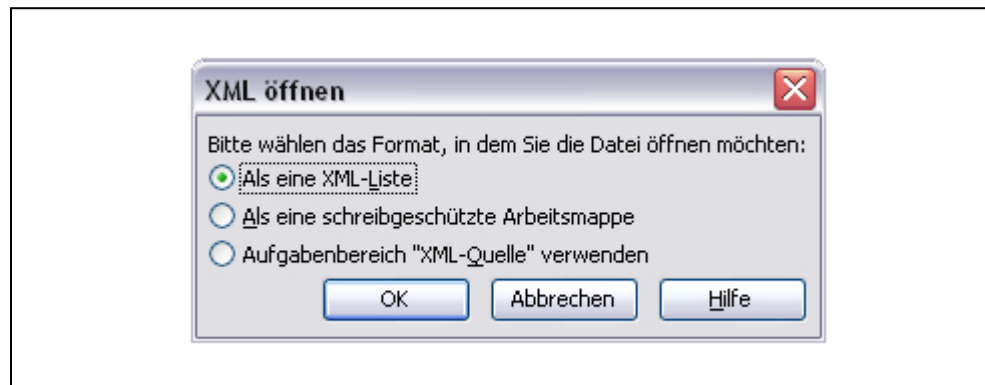


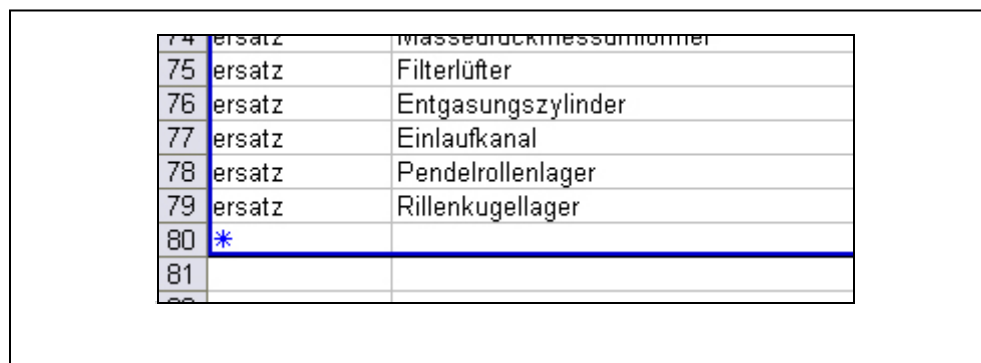
Abb. 18: Erstellen einer XML-Liste in Microsoft Excel

Diese Liste ähnelt einer Excel-Liste. Jedes Element und Attribut ist einer Spalte zugeordnet. Die Elementnamen werden dabei automatisch als Spaltenüberschriften übernommen. Der Listenbereich wird durch einen blauen Rahmen gekennzeichnet. (Abbildung 19)

	A	B	C	D
1	<b>typ</b>	<b>deutsch</b>	<b>englisch</b>	<b>alt</b>
2	extruder	rubicon - Standardextruder, kaltbeschickt	rubicon - cold-feed standard extruder	
3	extruder	rubicon - Standardextruder, warmbeschickt	rubicon - hot-feed standard extruder	
4	extruder	rubicon - Stifextruder, kaltbeschickt	rubicon - cold-feed pin barrel extruder	
5	extruder	rubicon - Vakuumextruder, kaltbeschickt	rubicon - cold-feed vent extruder	
6	extruder	rubicon - Silikonextruder, kaltbeschickt	rubicon - cold-feed silicone extruder	

Abb. 19: XML-Liste in Microsoft Excel

Die Inhalte der XML-Datei können hier wie Daten einer normalen Excel-Tabelle bearbeitet werden. Zum Hinzufügen weiterer Daten klickt man eine Zelle innerhalb des blauen Rahmens an. Dadurch wird am Ende der Liste eine neue Zeile eingefügt, die durch ein Sternchen gekennzeichnet ist. (Abbildung 20) In dieser Zeile können nun neue Daten in die Liste eingegeben werden.



74	ersatz	Wasserdrukmessumformer
75	ersatz	Filterlüfter
76	ersatz	Entgasungszyylinder
77	ersatz	Einlaufkanal
78	ersatz	Pendelrollenlager
79	ersatz	Rillenkugellager
80	*	
81		
82		

Abb. 20: Einfügen neuer Daten in eine XML-Liste

### Fazit

Der Einsatz von Microsoft Excel hat den Vorteil, dass keine neue Software angeschafft werden muss. Für den Nutzer bietet sich die Möglichkeit, in einer vertrauten Umgebung zu arbeiten. Er muss nicht erst lernen, eine neue Software zu bedienen, sondern kann relativ schnell und effektiv damit arbeiten.

Die Bearbeitung des XML-Dokuments ist sehr einfach, da nur die Element- und Attributinhalt zu sehen sind, ohne das für einen XML-Laien "verwirrende Drumherum". Excel bietet zwar keine Überprüfung auf Wohlgeformtheit oder Validität, ist aber für den Einsatzzweck, nämlich ein bestehendes XML-Dokument zu editieren und zu ergänzen, vollkommen ausreichend.

Zudem können die in der XML-Datei gespeicherten Daten durch die Excel XML-Liste leicht zwischen verschiedenen Dokumenten ausgetauscht und für andere Anwendungen in der rubicon GmbH bereitgestellt werden. Einen großen Nutzen brächte dies bei der Sprachanpassung der Anlagensteuerungen. Je nach Kundenwunsch bzw. -standort können die Bildschirme der Steuerung sprachlich angepasst werden. Die Übersetzung führt die Anlage anhand einer Excel-Tabelle durch, in der die entsprechenden Benennungen aufgelistet

sind. Zur Aktualisierung dieser Tabelle können nun einzelne Bereiche der XML-Liste durch Kopieren übernommen werden. Dies macht ein zeitaufwändiges Eintragen der einzelnen Benennungen überflüssig. Eine andere Möglichkeit ist auch, die XML-Liste in eine normale Excel-Tabelle zu konvertieren, die dann selbst, nach kleinen Anpassungen, als Grundlage für die Übersetzung verwendet werden kann. Die Umwandlung erfolgt einfach über das Menü `Daten/Liste/In Bereich umwandeln`.

## 10 Zusammenfassung

Die Terminologearbeit stellt eine wichtige Grundlage für die reibungslose Kommunikation innerhalb eines Unternehmens, aber auch zwischen dem Unternehmen und Außenstehenden, dar. Die hier vorliegende Diplomarbeit zeigt, wie man mit relativ einfachen Mitteln eine firmeninterne Terminologienormung in einem Unternehmen einführen kann. Durch das Zusammentragen und Vereinheitlichen bestehender Terminologie konnte die Grundlage für eine erfolgreiche Terminologearbeit geschaffen werden.

Eine einheitliche Terminologie bietet einem Unternehmen viele Vorteile. Vor allem der wirtschaftliche Faktor spielt dabei eine große Rolle. Durch die verbindliche Zuordnung einer Benennung zu einem Begriff werden den Mitarbeitern zeitraubende Suchen in Wörterbüchern oder anderen Medien erspart. Ebenso kann vermieden werden, dass ein und derselbe Begriff von mehreren Personen erneut nachgeschlagen werden muss, was einen sinnlosen Mehraufwand bedeutet. Der Aufwand - und damit auch die Kosten - werden gesenkt.

Obwohl die Einführung eines Terminologie-Managements oft aufwendig ist, lohnt sie sich. Die Vereinheitlichung der Terminologie kann Verbesserungen in allen Bereichen des Unternehmens mit sich bringen. Die firmeninterne Kommunikation wird vereinfacht. Ebenso kann nach außen hin eine einheitliche Unternehmenssprache repräsentiert werden. Auch Übersetzungen werden davon beeinflusst. Besteht bereits in der Ausgangssprache eine Konsistenz wird sich dies auch positiv auf die Qualität der Übersetzungen auswirken.

Im Rahmen dieser Arbeit wurde eine Terminologie-Anwendung entwickelt, die ein schnelles Nachschlagen von Begriffen ermöglicht. Die

Anwendung ist über das Intranet zugänglich und somit immer für alle Mitarbeiter verfügbar. Die Pflege der Daten kann zentral erfolgen, wodurch allen die aktuelle Version zur Verfügung steht.

Die Basis der Anwendung stellt ein XML-Dokument dar, das die zusammengetragenen Benennungen enthält. Für die Realisierung einer Webseite, mit der die Daten aus dem XML-Dokument abgerufen und ausgegeben werden können, wurden zwei Technologien vorgestellt - XSLT und AJAX. Mit beiden war es möglich, eine Anwendung zu erstellen, die auf die firmenspezifischen Anforderungen zugeschnitten ist. Die Anwendungen sind flexibel änderbar und können somit leicht an zukünftige Anforderungen angepasst werden.

Unabhängig von der eigentlichen Anwendung können die in der XML-Datei gespeicherten Terminologie-Daten jederzeit bearbeitet und ergänzt werden. Diese Tatsache ist von besonderer Bedeutung für die Terminologiearbeit, da diese einen fortwährenden Prozess darstellt. Die Fachwortbestände werden sich auch in Zukunft weiterentwickeln und vergrößern und müssen somit kontinuierlich erfasst werden.

## 11 Literaturverzeichnis

### Printmedien:

#### **Ammelburger 2004**

Ammelburger, Dirk: XML - Grundlagen der Sprache und Anwendungen in der Praxis. München: Hanser, 2004.

#### **Arntz, Picht, Mayer 2004**

Arntz, Reiner; Picht, Herbert; Mayer, Felix: Einführung in die Terminologearbeit. Hildesheim: Olms, 2004.

#### **Born 2003**

Born, Günter: HTML Kompendium. München: Markt+Technik, 2003.

#### **Carl 2006**

Carl, Denny: Praxiswissen AJAX. Köln: O'Reilly, 2006.

#### **Gamperl 2006**

Gamperl, Johannes: AJAX. Web 2.0 in der Praxis. 1., korrig. Nachdruck. Bonn: Galileo Press, 2006.

#### **Harold 2004**

Harold, Elliotte R.: XML. 2. Aufl. Bonn: mitp, 2004.

#### **Hauser 2006**

Hauser, Tobias: XML-Standards schnell + kompakt. Frankfurt: Software & Support, 2006.

**Herold 2002**

Herold, Helmut: Das HTML/XHTML-Buch mit Cascading Style Sheets und einer Einführung in XML. Nürnberg: SuSE-PRESS, 2002.

**Jacobsen 2002**

Jacobsen, Jens: Website-Konzeption. Erfolgreiche Web- und Multimedia-Anwendungen entwickeln. München: Addison-Wesley, 2002.

**Jänecke 2003**

Jänecke, Michael: XML. In: Avci, Oral; Trittmann, Ralph; Mellis, Werner (Hrsg.): Web-Programmierung. Softwareentwicklung mit Internet-Technologien - Grundlagen, Auswahl, Einsatz - XHTML&HTML, CSS, XML, JavaScript, VBScript, PHP, ASP, Java. Wiesbaden: Vieweg, 2003. S. 406-430.

**Mintert 2003**

Mintert, Stefan (Hrsg.): XHTML, CSS & Co. Die W3C-Spezifikationen für das Web-Publishing. München: Addison-Wesley, 2003.

**Pomaska 2005**

Pomaska, Günter: Grundkurs Webprogrammierung. Interaktion, Grafik und Dynamik - Mit XHTML und CSS, XML, JavaScript, Applets, SVG, PHP. Wiesbaden: Vieweg, 2005.

**Schulz 2003**

Schulz, Matthias: Terminologie als wirtschaftlicher Faktor. Abtsgmünd: Schulz, 2003.

**Shepherd 2002**

Shepherd, Devan: XML. München: Markt+Technik, 2002.

**Stein 2002**

Stein, Magnus: workshop XML. München: Addison-Wesley, 2002

**Tidwell 2003**

Tidwell, Doug: XSLT. 1. korr. Nachdruck. Köln: O'Reilly, 2003

**Wenz 2006**

Wenz, Christian: AJAX schnell + kompakt. Frankfurt: Software & Support, 2006.

**Wenz 2007**

Wenz, Christian: JavaScript & AJAX. Das umfassende Handbuch. 7., aktualis. Auflage. Bonn: Galileo Press, 2007.

**DIN-Normen:****DIN 2342 Teil 1**

DIN 2342 Teil1: Begriffe der Terminologie. Grundbegriffe. Oktober 1992.

**Internetquellen:****www.rubicon-halle.de**

rubicon Gummitechnik und Maschinenbau GmbH: Wir über uns.

URL: <http://www.rubicon-halle.de/ger/wir.htm> (20.06.2006)

**www.tekom.de**

Schmitz, Klaus-Dirk: Tagungsflyer. Erfolgreiche Terminologiearbeit - von Anfang an. 2006.

URL: [http://www.tekom.de/index\\_neu.jsp?url=/servlet/ControllerGUI?action=voll&id=1749](http://www.tekom.de/index_neu.jsp?url=/servlet/ControllerGUI?action=voll&id=1749) (16.11.2006)

**www.w3schools.com**

W3 Schools: Browser Statistics. 2006.

URL: [http://www.w3schools.com/browsers/browsers\\_stats.asp](http://www.w3schools.com/browsers/browsers_stats.asp)

(09.12.2006)

**www.teialehrbuch.de**

TEIA AG - Internet Akademie und Lehrbuch Verlag: Kurs: XML Grundlagen.

URL: <http://www.teialehrbuch.de/XML1-KK/20530-XML-Editoren.html>

(28.11.2006)

**www.adaptivepath.com**

Garrett, Jesse J.: Ajax: A New Approach to Web Applications. 2005.

URL: <http://www.adaptivepath.com/publications/essays/archives/000385.php>

(08.12.2006)

## **12 Eidesstattliche Erklärung**

Ich versichere, dass ich die Arbeit selbständig angefertigt, keine anderen als die angegebenen Hilfsmittel benutzt und alle wörtlichen oder sinnge-  
mäßigen Entlehnungen deutlich als solche gekennzeichnet habe.

Merseburg, 14.12.2006

---

## **13 Anhang**