

Universität Freiburg i. Ue. Institut für Informatik

XSL zur flexiblen Darstellung von XML-Dokumenten

Seminararbeit in Wirtschaftsinformatik

vorgelegt bei Prof. Dr. Andreas Meier

von Dominic Giger

Dominic Giger
Route du Varis 9
1700 Fribourg
026/321'15'49
Dominic.Giger@unifr.ch

Freiburg i. Ue. im Mai 2002

Inhaltsverzeichnis

Inhaltsverzeichnis	I
Abkürzungsverzeichnis	II
Abbildungsverzeichnis	II
1 Einleitung	4
2 XML - Extensible Markup Language	5
2.1 XML-Grundlagen	5
2.1.1 Entwicklungsgeschichte	5
2.1.2 XML Allgemein	6
2.1.3 XML versus HTML	6
2.1.4 Aufbau und Syntax eines XML-Dokuments	7
2.2 DTD und XMLSchema	9
2.2.1 Dokumenttyp-Deklaration (DTD)	9
2.2.2 XMLSchema	9
2.2.3 Namensräume	11
3 Die Formatierung von XML-Dokumenten	13
3.1 CSS - Cascading Stylesheet Language	13
3.2 XSL - Extensible Stylesheet Language	14
3.2.1 XSLT - XSL Transformations	14
3.2.2 XSL-FO - XSL Formatting Objects	16
4 Anwendungsbeispiel:	
Die flexible Darstellung von IDML-Reporting-Dokumenten	18
4.1 IDML Allgemein	18
4.2 IDML Reporting	18
4.3 Formatierung eines IDML-Reporting-Dokuments	20
5 Schlussbemerkung	23
6 Anhang	24
A.1 Projektreport	24
A.2 Stylesheet 1	25
A.3 Stylesheet 2	28
7 Literatur und Quellen	29

Abkürzungsverzeichnis

CSS:	Cascading Style Sheet
DTD:	Dokumenttyp-Deklaration
engl.:	englisch
IDML:	International Development Language
IT:	Informationstechnologie
HTML:	Hypertext Markup Language
MATHML:	Mathematical Markup Language
PCDATA:	Parsed Character Data
PDF:	Portable Document Format
RDF:	Resource Description Framework
SGML:	Standard Generalized Markup Language
URL:	Uniform Resource Locator
W3C:	World Wide Web Consortium
XHTML:	Extensible Hypertext Markup Language
XML:	Extensible Markup Language
XSL:	Extensible Stylesheet Language
XSL-FO:	Extensible Stylesheet Language Formatting Objects
XSLT:	Extensible Stylesheet Language Transformations

Abbildungsverzeichnis

Abbildung 1: Beispieldokument gliederung.xml

[Abbildung 2: Beispiel einer DTD](#)

Abbildung 3: Beispiel eines Schemas

Abbildung 4: Beispiel eines CSS-Stylesheets

Abbildung 5: Darstellung im Browser

Abbildung 6: Beispiel eines XSL-Stylesheets

Abbildung 7: HTML-Output

Abbildung 8: Stylesheet zur Umwandlung eine XML-Dokuments in ein XSL-FO-Dokument

Abbildung 9: Das erzeugte XSL-FO-Dokument

Abbildung 10: Überblick IDML-Reporting Schema

Abbildung 11: Beispielstylesheet 1

Abbildung 12: Beispielstylesheet 2

Abbildung 13 : Darstellung im Browser

1. Einleitung

Seit der Empfehlung des XML 1.0 Standards durch das W3C hat sich XML mit grosser Geschwindigkeit in der IT-Welt verbreitet. XML wird heute in Gebieten der IT eingesetzt, an die niemand bei der Entwicklung des Standards gedacht hätte. XML entwickelt sich immer mehr zum Format der Wahl für den Austausch und die Speicherung von Informationen im Internet [Harold/Means 2001, S. 243]. In dieser Arbeit soll dargestellt werden wie XML-Dokumente mit Hilfe von XSL flexibel präsentiert werden können.

In einem ersten Schritt sollen dazu zunächst die Grundlagen von XML erarbeitet werden. Als erstes wird die Entwicklungsgeschichte von XML betrachtet. Danach der grundlegende Unterschied zwischen XML und HTML kurz dargestellt. Im weiteren wird die Syntax und der Aufbau von XML-Dokumenten erläutert. Anschliessend werden die zwei grundlegenden Mechanismen zur Definition von Dokument-Strukturen vorgestellt (DTD und XMLSchema).

Im dritten Kapitel werden drei verschiedene Möglichkeiten der Präsentation von XML-Dokumenten kurz vorgestellt. Es wird sich dabei herausstellen, dass nur eine dieser drei Möglichkeiten zur Zeit für die Präsentation von XML-Dokumenten auf dem Web praktikabel ist.

Die so erarbeiteten Grundlagen von XML und der Präsentation von XML-Dokumenten werden in einem letzten Schritt mit einem konkreten Anwendungsbeispiel noch weiter verdeutlicht. Dazu werden zu einem IDML-Reporting-Dokument (IDML-Reporting ist eine XML-Applikation) einige Beispielstylesheets entworfen.

Diese Arbeit kann und soll keine vollständige Einführung in XML sein. Die vorgestellten Möglichkeiten zur flexiblen Formatierung von XML-Dokumenten sind nur ein kleiner Ausschnitt aus der Vielzahl von Darstellungsmitteln die XSL bietet. Für weitergehende Informationen zu XML selbst oder zur Darstellung von XML-Dokumenten sei auf die im Quellenverzeichnis angegebenen Titel und Webseiten verwiesen.

2. XML – Extensible Markup Language

Dieses Kapitel gibt einen kurzen Einblick in XML.

2.1 XML Grundlagen

Zunächst wird kurz die Entwicklungsgeschichte dargestellt. Im weiteren wird kurz besprochen, was XML eigentlich ist. Es folgt ein Abschnitt, der den Unterschied zwischen XML und HTML erläutert. Schliesslich wird ein Blick auf die Syntax von XML geworfen.

2.1.1 Entwicklungsgeschichte

XML ist eine Untermenge der Standard Generalized Markup Language (SGML).

„SGML ist der Standard zur Beschreibung von Dokumenten festgelegt von der International Standardization Organization (ISO), der unter der Nummer ISO 8879 im Jahr 1986 veröffentlicht wurde. Das Ziel und die Idee dieses Standards ist es, die Struktur des Inhalts eines Dokuments von seiner layoutorientierten Erscheinungsform zu trennen.“

[Hofmann/Raitelhuber 1998, S. 2]

SGML ist gut geeignet für grosse Organisationen, die anspruchsvolle Standards für ihre Dokumente benötigen. Aufgrund seiner Komplexität (die Spezifikation von SGML umfasst 155 Seiten) fand es aber ausserhalb von grossen Organisationen kaum Verbreitung. Mit dem Boom des Internet Anfangs der Neunziger Jahre kam das Bedürfnis nach einer einfachen Auszeichnungssprache für Webseiten auf. Dies führte zur Entwicklung der Hypertext Markup Language (HTML). Der erste Standard von HTML (HTML 1.0 DTD) wurde 1993 publiziert [Günther 1997]. HTML ist auf die Präsentation von Webseiten ausgerichtet, zu mehr ist es, als rein darstellungsorientierte Sprache, kaum in der Lage. Mit HTML wurde ein grundlegendes Konzept von SGML, die Trennung von Form und Inhalt eines Dokuments, aufgegeben. Mit der enormen Verbreitung von HTML fielen zunehmend auch die Unzulänglichkeiten der Sprache auf. HTML bietet bei weitem nicht mehr die Leistungsfähigkeit, die SGML besass. So begann 1996 die Entwicklung von XML, die zum Ziel hatte eine vereinfachte Version von SGML zu schaffen, ohne dabei die Leistungsfähigkeit von SGML einzubüssen.

[Harold/Means 2001, S. 8ff]. Die Version 1.0 von XML wurde am 10. Februar 1998 durch das World Wide Web Consortium (W3C) verabschiedet [W3CX 1998].

2.1.2 XML Allgemein

Die Extensible Markup Language (XML) ist eine Metasprache zur Definition von anwendungsspezifischen Dokumenttypen [Dünhölder 1998, S. 2]. Eine Metasprache ist eine Sprache mit der andere Sprachen definiert werden können. So bildet zum Beispiel die deutsche Grammatik zusammen mit dem deutschen Wörterbuch eine Metasprache für geschriebenes Deutsch [Astra 2002]. Dokumenttypen sind Dokumente die in ihrem Aufbau dem gleichen Grundmuster folgen (zum Beispiel HTML-Dokumente). HTML wurde vom W3C als Dokumenttyp in XML neu formuliert (vgl. dazu [W3CH 2000]).

2.1.3 XML versus HTML

Im Gegensatz zu HTML (Hypertext Markup Language) beschreiben Tags in XML die darin eingeschlossenen Daten und sind nicht nur Formatierungsanweisungen. Um diesen Unterschied deutlich zu machen folgt ein kurzes Beispiel.

In einem HTML-Dokument könnte folgendes stehen:

```
<td>1700</td>
```

Was aber repräsentiert die Zahl 1700?

- Ist es der Preis eines Computersystems?
- Die Postleitzahl von Fribourg?
- Das Geburtsjahr von Daniel Bernoulli?
- Die Anzahl Studenten an der Rechtswissenschaftlichen Fakultät der Universität Zürich?

Es könnte alles von dem oben genannten sein! Der Tag `<td>` kennzeichnet in HTML eine Zelle einer Tabelle. `<td>` ist also eine reine Formatierungsanweisung und beinhaltet keinerlei Information darüber, was die Zahl 1700 repräsentiert [Tidwell 2001, S. 5].

Das primäre Ziel von HTML ist die Formatierung eines Dokuments. Inhalt und Form eines Dokuments werden bei HTML nicht getrennt. Doch genau diese Trennung von Inhalt und Form ist ein etabliertes Prinzip in der Druckindustrie [Tidwell 2001, S. 5]. Im Gegensatz zu einem HTML-Dokument enthält ein XML-Dokument keinerlei Formatierungsanweisungen. So könnte der Tag

```
<Postleitzahl>1700</Postleitzahl>
```

in XML die Zahl 1700 als Postleitzahl kennzeichnen. Damit wird die Identifikation der inhaltlichen Bedeutung der Zahl 1700 möglich. Die Identifikation der inhaltlichen Bedeutung ist die Voraussetzung für eine maschinelle Weiterverarbeitung der Daten. Durch die Kennzeichnung von Inhalten wird es für Programme wesentlich einfacher Informationen in Dokumenten zu finden. So ist es zum Beispiel sehr einfach den Preis eines Produktes in einem XML-Dokument zu finden, wenn dieser beispielsweise mit einem Tag wie `<preis>20.99</preis>` gekennzeichnet ist. Sehr viel schwieriger gestaltet sich im Gegensatz dazu die Suche nach dem Preis eines Produktes in einem HTML-Dokument. XML ist im Gegensatz zu HTML nicht auf die reine Präsentation von Dokumenten ausgerichtet sondern bietet als strukturelle Auszeichnungssprache wesentlich mehr.

2.1.4. Aufbau und Syntax eines XML-Dokuments

Im weiteren wird anhand eines Beispiel der Aufbau von XML-Dokumenten erläutert.

Abbildung 1 zeigt wie ein XML-Dokument aussehen könnte.

<code><?xml version="1.0" encoding="UTF-8" standalone="no" ?></code>	← Prolog
<code><!DOCTYPE Gliederung SYSTEM "F:\idml\Beispiele\Gliederung2.dtd"></code>	← Dokumenttyp-Deklaration
<code><?xml-stylesheet href="F:\idml\Beispiele\Gliederung2.css" type = "text/css"?></code>	← Steueranweisung
<code><Gliederung></code>	← Wurzelement
<code><Kapitel AnzahlSeiten="8"></code>	← Kindelement von Gliederung
<code><Kapiteltitel>Kapitel 1</Kapiteltitel></code>	
<code><Unterkapitel></code>	
<code><Unterkapiteltitel>Unterkapitel 1</Unterkapiteltitel></code>	
<code><Abschnitt></code>	
<code><Abschnittstitel>Abschnitt 1</Abschnittstitel></code>	
<code><Abschnittstext>Hier beginnt der Text von Abschnitt 1</Abschnittstext></code>	
<code></Abschnitt></code>	
<code></Unterkapitel></code>	
<code></Kapitel></code>	
<code></Gliederung></code>	
<code><!--Zum Beispiel die Gliederung einer Arbeit--></code>	← Kommentar

Abbildung 1: Beispieldokument gliederung.xml

Der Prolog oder die XML-Deklaration hat die Attribute `version`, `encoding` und `standalone`. `Version` ist die XML Version, auf der das Dokument basiert (der bisher einzige mögliche Wert ist 1.0). `Encoding` gibt den Zeichensatz an, in dem das XML-Dokument kodiert ist. Die Angabe der Kodierung ist optional. Fehlt sie wird als Kodierung der Unicode-Zeichensatz¹ angenommen. Besitzt das Attribut `standalone` den Wert „no“, so beruht das XML-Dokument auf einer externen Dokumenttyp-Deklaration (DTD). Die XML-Deklaration muss als erstes

¹ Unicode ist ein Zeichensatz, der gross genug ist um sämtliche Schriftzeichen aller Sprachen aufzunehmen. Eine genauere Beschreibung findet sich in [Unicode 2002].

im XML-Dokument stehen, es dürfe keine Kommentare oder Steueranweisungen vor ihr auftauchen.

Die Dokumenttyp-Deklaration gibt an, dass „Gliederung“ das Wurzelement des Dokuments ist und dass die Dokumenttyp-Deklaration unter der URL `F:\idml\Beispiele\Gliederung2.dtd` zu finden ist.

Die Steueranweisung gibt in diesem Fall ein Stylesheet an, mit dem das Dokument in einem Browser dargestellt werden soll.

Ein Element beginnt mit einem Start-Tag (`<tag>`) und endet mit einem End-Tag (`</tag>`). Der Inhalt des Elements wird zwischen Start- und End-Tag eingeschlossen. Im Gegensatz zu HTML sind Elemente ohne End-Tags in XML nicht erlaubt (in HTML hat beispielsweise `
` keinen End-Tag). `<tag> </tag>` kann in XML wahlweise auch als `<tag/>` geschrieben werden. Elemente dürfen sich in XML nicht überschneiden, eine Kombination wie zum Beispiel ` fett <i> kursiv-fett kursiv </i>` [Tidwell 2001, S. 7], die in HTML erlaubt wäre, ist in XML nicht möglich. Jedes XML-Dokument muss genau ein Wurzelement enthalten, das die übrigen Elemente umschließt. Das Element „Gliederung“ ist das Wurzelement des Beispieldokuments. Es enthält das Kindelement Kapitel, das seinerseits wieder das Elternelement von Kapiteltitel und Unterkapitel ist. Ein XML-Dokument bildet folglich eine Art Baumstruktur, dies wird in Kapitel 3.2.1 eine wichtige Rolle spielen, wenn es um die Transformation eines XML-Dokuments in ein anderes Dokument geht.

Ein XML Dokument kann mit Kommentaren versehen werden. Kommentare beginnen mit `<!--` und enden mit `-->`. Kommentare dürfen nicht verschachtelt werden und nicht innerhalb eines Tags auftauchen [Harold/Means 2001, S. 14ff].

Entspricht ein XML-Dokument vollständig der Syntax von XML so spricht man von einem wohlgeformten (wellformed) Dokument.² XML-Dokumente müssen zwingend wohlgeformt sein.

Ein XML-Dokument mag zwar für einen Menschen lesbar sein, hat aber alleine keine Aussagekraft (für eine Maschine). Eine Maschine benötigt weitere Informationen über die Elemente die in einem XML-Dokument vorkommen. Diese Informationen werden der Maschine durch DTDs geliefert [Schädler 2001].

Das Beispieldokument (`gliederung.xml`) beruht auf einer DTD, welche im folgenden genauer untersucht werden soll.

² Die aufgezählten Syntax-Regeln sind bei weitem nicht vollständig. Eine vollständige Beschreibung der XML-Syntax findet sich in [W3CX 1998].

2.2 DTD und XMLSchema

In diesem Kapitel sollen zwei grundlegende Mechanismen zur Definition von Dokument-Strukturen vorgestellt werden.

2.2.1 Dokumenttyp-Deklaration (DTD)

Die Dokumenttyp-Deklaration des Beispieldokuments (gliederung.xml) ist in Abbildung 2 dargestellt.

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT Gliederung (Kapitel+)>
<!ELEMENT Kapitel (Kapiteltitle, Unterkapitel+)>
<!ELEMENT Unterkapitel (Unterkapiteltitle, Abschnitt+)>
<!ELEMENT Abschnitt (Abschnitttitle, Abschnittstext)>
<!ELEMENT Abschnitttitle (#PCDATA)>
<!ELEMENT Kapiteltitle (#PCDATA)>
<!ELEMENT Unterkapiteltitle (#PCDATA)>
<!ELEMENT Abschnittstext (#PCDATA)>
<ATTLIST Kapitel AnzahlSeiten CDATA #IMPLIED>
```

Abbildung 2: Beispiel einer DTD

In einer DTD werden alle Elemente deklariert, die in einem XML-Dokument, das sich an diese DTD hält, vorkommen dürfen. Element-Deklartionen besitzen die folgende Grundform:

```
<!ELEMENT Elementname (Inhaltsmodell)>
```

Das Inhaltsmodell beschreibt, welche Kinder ein Element in welcher Reihenfolge haben kann oder muss [Harold/Means 2001, S. 37]. Die zweite Zeile im Beispiel definiert ein Element mit dem Namen Gliederung, das als Kinderelemente ein oder mehrere Elemente vom Typ Kapitel enthalten kann. Die dritte Zeile gibt an, dass das Element Kapitel genau einen Kapiteltitle und ein oder mehrere Unterkapitel enthält. Die fünf letzten Zeilen definieren die Elemente Abschnitttitle, Kapiteltitle, Unterkapitel und Abschnittstext als "parsed character data" also reinen Text und ein Attribut "AnzahlSeiten" zum Element Kapitel, das optional (implied) ist. Eine ausführliche Beschreibung von DTDs findet sich in [Harold/Means 2001, S. 29ff] oder [W3CX 1998].

2.2.2 XMLSchema

Eine DTD sagt niemals etwas über die Bedeutung oder die erlaubten Werte des Inhalts eines Elements aus [Harold/Means 2001]. DTDs stellen lediglich einen Datentyp bereit (PCDATA, also Text). DTDs waren nur dazu gedacht Texte zu digitalisieren, wozu dieser eine Datentyp auch ausreichend war.

XML hatte aber einen derartigen Zuspruch erlangt, dass weitaus höhere Anforderungen an die Sprache gestellt wurden als nur Texte zu digitalisieren [Hoven/Liebrecht 2001]. So braucht zum Beispiel eine grössere Datentypvielfalt um Daten zwischen zwei Datenbanken zu transferieren.

XMLSchema ist eine Weiterentwicklung von DTDs. In XMLSchema können Datentypen und andere komplexe Strukturen beschrieben werden, die mit DTDs nur schwer oder gar nicht beschrieben werden könnten [Tidwell 2001]. XMLSchema bietet gegenüber DTDs folgende Vorteile: bessere Lesbarkeit (Schemas sind selber XML-Dokumente), Namensräume, die Möglichkeit der objektorientierten Programmierung (Vererbung), über 40 vordefinierte Datentypen (DTDs besitzen lediglich einen Datentyp) und die Möglichkeit eigene Datentypen genau zu definieren [Schädler 2001].

Ein Schema für das Beispieldokument (gliederung.xml, Abbildung 1 auf Seite 7) könnte wie in Abbildung 3 aussehen.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="F:\idml\Beispiele\Gliederung.xsd"
xmlns:gliederung="F:\idml\Beispiele\Gliederung.xsd" xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:element name="Gliederung">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Kapitel" type="gliederung:KapitelType" minOccurs="0"
maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:complexType name="KapitelType">
    <xs:sequence>
      <xs:element name="Kapiteltitel" type="gliederung:TextType"/>
      <xs:element name="Unterkapitel" type="gliederung:UnterkapitelType" minOccurs="0"
maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="AnzahlSeiten" type="xs:integer" use="optional"/>
  </xs:complexType>
  <xs:complexType name="UnterkapitelType">
    <xs:sequence>
      <xs:element name="Unterkapiteltitel" type="gliederung:TextType"/>
      <xs:element name="Abschnitt" type="gliederung:TextType" minOccurs="0"
maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="TextType">
    <xs:simpleContent>
      <xs:extension base="xs:string"/>
    </xs:simpleContent>
  </xs:complexType>
</xs:schema>
```

Abbildung 3: Beispiel eines Schemas

Das erste Element des Beispielschemas (<xs:schema>) setzt die Namensräume für das Schema. Das Wurzelement Gliederung besteht aus einer Sequenz von Kapitel-Elementen,

welche genau einen Kapiteltitel enthalten und eine Sequenz von Unterkapiteln. Die `minOccurs` und `maxOccurs` Attribute der einzelnen Elemente geben an wie oft ein Element vorkommen muss bzw. vorkommen darf (wenn nichts anderes angegeben wird haben `minOccurs` und `maxOccurs` standardmässig den Wert 1). Hat `maxOccurs` den Wert "unbounded" so darf ein Element beliebig oft vorkommen [W3CS 2001].)

In einem Schema werden im wesentlichen folgende Dinge deklariert:

- Die Typen der Elemente, die in einem Dokument vorkommen dürfen.
- Die Attribute, die ein Element haben kann, sowie deren Typen.
- Die Struktur eines auf dem Schema beruhenden Dokuments, d. h. wo und wie oft die einzelnen Elemente im Dokument vorkommen dürfen.

In XMLSchema gibt es zwei Arten von Typen: Komplexe Typen (`complexType`) und einfache Typen (`simpleType`). Komplexe Typen können Sequenzen (`sequence`), Elemente, Attribute und einfache Typen enthalten. Einfache Typen können weder Attribute noch Elemente enthalten. Bei einfachen Typen handelt es sich entweder um durch den Standard vordefinierten Typen³ oder um Erweiterungen (`extension`) dieser vordefinierten Typen.

Mit den im Rahmen des XMLSchema Standards vordefinierten Datentypen können Werte von Elementen eindeutig definiert werden. So wurde zum Beispiel im oben betrachteten Schema ein Attribut "AnzahlSeiten" zum Element Kapitel definiert, das nur eine Zahl (integer) sein kann.

Ein XML-Parser validiert ein XML-Dokument anhand der DTD oder des Schemas, das im Dokument referenziert ist. Ein XML-Dokument ist gültig (engl.: valid), wenn es keine anderen Elemente und Attribute enthält als in der DTD oder im Schema deklariert sind und wenn die Elemente und Attribute im richtigen, durch DTD/Schema vorgegebenen, Kontext benutzt werden (also zum Beispiel in der richtigen Reihenfolge im Dokument auftauchen, richtig geschachtelt sind etc.). Gültigkeit ist im Gegensatz zu Wohlgeformtheit optional [Harold/Means 2001, S. 30].

2.2.3 Namensräume

Namensräume (engl.: namespace) wurden mit XMLSchema eingeführt. Das Konzept von Namensräumen stammt aus der objektorientierten Programmierung. "Ein Namensraum ist ein Mechanismus zum Ausdrücken logischer Gruppierungen." [Stroustrup 2000, S.179]

³ eine Liste aller vordefinierten Typen findet sich in [W3CS 2001].

Namensräume haben in XML zwei Aufgaben: Zwischen Elementen und Attributen mit gleichem Namen zu unterscheiden und die Elemente und Attribute einer XML-Applikation zusammenzufassen [Harold/Means 2001, S. 63]. Eine XML-Applikation (z. B. XHTML oder MathML) ist eine Menge von Tags zur Auszeichnung von Dokumenten eines bestimmten Typs.

Wenn zwei XML-Applikationen Elemente definieren und beide den gleichen Namen für ein Element vergeben kann das zu einem Namenskonflikt führen (z. B. wenn in der einen XML-Applikation auch Elemente aus der anderen XML-Applikation benutzt werden sollen). Um in diesem Fall zwischen zwei Elementen unterscheiden zu können müssen sie unterschiedlichen Namensräumen zugeordnet werden. So enthält zum Beispiel die XML-Applikation RDF (Resource Description Framework, zur Beschreibung von Ressourcen mittels Metadaten) ein Element mit dem Namen "Description" [W3CR 1999]. Möchte man nun in einem XML-Dokument ein weiteres Element mit dem gleichen Namen benutzen (zum Beispiel zur Beschreibung eines Elementes "motorcycle") muss man diese zwei Elemente an zwei unterschiedliche Namensräume binden. Im Beispiel in Abbildung 3 werden die Namensräume mit

```
<xs:schema targetNamespace="F:\idml\Beispiele\Gliederung.xsd"
xmlns:gliederung="F:\idml\Beispiele\Gliederung.xsd" xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
```

gesetzt. targetNamespace ist der Namensraum für die im Schema deklarierten Elemente. Mit xmlns:xs wird die URL des Namensraums von XMLSchema an das Präfix xs gebunden.

Einem Elemente aus dem Namensraum von XMLSchema, das im Schema Gliederung.xsd verwendet werden soll, wird dann mit <xs:[Name des Elements]> als Element aus dem Namensraum von XMLSchema gekennzeichnet. Das Präfix kennzeichnet den Namensraum zu dem das Element gehört. Der Name des Elements identifiziert das Element im Namensraum. Das ganze Konstrukt wird qualifizierter Name genannt [Harold/Means 2001, S. 67]. Die zwei Attribute elementFormDefault und attributeFormDefault weiter oben geben an ob Elemente und Attribute standardmässig qualifiziert sein müssen (zu einem Namensraum gehören müssen) oder nicht.

Bisher wurde die Syntax eines XML-Dokuments erläutert und es wurden die zwei grundlegenden Möglichkeiten (DTD und XMLSchema) zur Beschreibung von XML-Applikationen kurz vorgestellt. Dieses Kapitel beschäftigte sich vor allem mit der Struktur eines XML-Dokuments. Das nun folgende Kapitel wird sich mit der Formatierung von XML-Dokumenten befassen.

3. Die Formatierung vom XML-Dokumenten

Im folgenden werden verschiedene Möglichkeiten der Präsentation von XML-Dokumenten beschrieben.

3.1 CSS - Cascading Stylesheet Language

Die bei weitem einfachste und direkteste Möglichkeit XML-Dokumente zu formatieren ist die direkte Darstellung eines mit CSS formatierten XML-Dokuments in einem Browser (z. B. Internet Explorer 6.0). Abbildung 4 zeigt wie ein CSS-Stylesheet für das XML-Dokument aus Abbildung 1 (S. 7) aussehen könnte.

```
Gliederung {display: block;font-family: arial;font-weight: bold;text-align: left;
            line-height: 4eX;font-size: 18pt}
Unterkapiteltitle {display: block;font-size: 16pt}
Abschnittstitle {display: block;font-size: 14pt;font-weight: normal}
Abschnittstext {font-size: 12pt}
```

Abbildung 4: Beispiel eines CSS-Stylesheets

In einem CSS-Stylesheet wird für einzelne Elemente in einem XML-Dokument angegeben, wie sie im Browser dargestellt werden sollen. Auf die einzelnen Stilelemente soll nicht näher eingegangen werden. Die Stile gelten auch für die Abkömmlinge der Elemente für die sie vorgegeben wurden, sie bilden sozusagen eine Kaskade (daher auch der Name Cascading Stylesheets) [Harold/Means 2001, S. 210]. So kann zum Beispiel die Standard-Schriftart für das darzustellende Dokument in der Stilregel des Wurzelements (im Beispiel das Element "Gliederung") angegeben werden. Wenn ein Element in einer anderen Schriftart dargestellt werden soll, so wird in der Stilregel des betreffenden Elements diese Schriftart explizit angegeben. Mittels dem obigen Stylesheet formatiert, sieht das Beispieldokument aus Abbildung 1 (S. 7) folgendermassen aus (Internet Explorer 6.0):

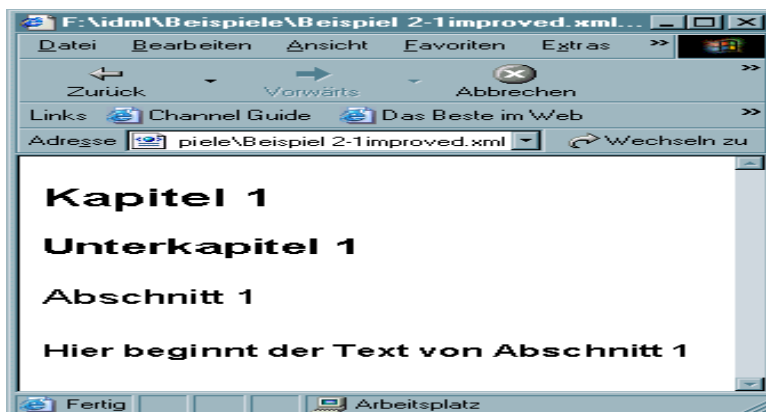


Abbildung 5: Darstellung im Browser

Der grösste Nachteil der direkten Darstellung eines XML-Dokuments mittels CSS in einem Browser ist, dass ältere Browser (ohne eingebaute Unterstützung für XML) diese nicht beherrschen. (Internet Explorer beispielsweise erst ab der Version 5.0) Soll ein Dokument im WWW publiziert werden ist diese Lösung nicht praktikabel, da Anwender mit älteren Browserversionen ausgegrenzt würden. Eine Lösung dieses Problems bietet die serverseitige Transformation von XML-Dokumenten in HTML-Dokumente mittels XSLT und die anschließende Formatierung mit CSS.

Desweiteren bietet CSS nur einige Selektoren um Elemente auszuwählen. Diese Selektoren sind aber bei weitem nicht so leistungsfähig wie die XPath-Syntax aus XSLT (siehe weiter unten) [Harold/Means 2001, S. 213].

3.2 XSL - Extensible Stylesheet Language

Die Extensible Stylesheet Language besteht aus zwei XML-Applikationen. XSLT zur Transformation von XML-Dokumenten in eine andere Form (z. B. HTML) und XSL-FO zur eigentlichen Formatierung von XML-Dokumenten.

3.2.1 XSLT - XSL Transformations

Wie in Kapitel 2.1.4 bereits erwähnt wurde, lässt sich ein XML-Dokument auf eine Baumstruktur abbilden. Die Transformation eines XML-Dokuments ist die Umwandlung eines Quellbaums in einen Ergebnisbaum mittels eines XSLT-Stylesheets. Abbildung 6 zeigt wie ein XSLT-Stylesheet für das Beispieldokument aus Abbildung 1 (S. 7) aussehen könnte.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="Gliederung">
  <html>
    <body>
      <head><title></title></head>
      <xsl:apply-templates/>
    </body>
  </html>
</xsl:template>

<xsl:template match="Kapitel">
  <h3>
    <xsl:value-of select="Kapiteltitel"/>
    ( <xsl:value-of select="@AnzahlSeiten"/> )
  </h3>
  <xsl:apply-templates select="Unterkapitel"/>
</xsl:template>
[...weitere Templates zur Darstellung der Elemente Unterkapitel und Abschnitt...]
```

Abbildung 6: Beispiel eines XSL-Stylesheets

Ein XSL-Stylesheet ist wie ein Schema ein XML-Dokument. In der ersten Zeile des Beispiels wird der Namensraum für die Elemente von XSL gesetzt, die im weiteren verwendet werden. Das restliche Stylesheet besteht aus einer Reihe von Regeln. Diese Regeln geben an, wie das XML-Dokument, auf das das Stylesheet angewendet wird, in ein anderes Dokument transformiert werden soll. Zur Umwandlung eines XML-Dokuments mittels eines Stylesheets ist ein XSLT-Prozessor notwendig, der die Elemente des Ausgangsdokuments mit den im Stylesheet angegebenen Vorlagen (engl. Templates) vergleicht. Findet der XML-Prozessor ein Template, das auf ein Element passt (engl: match) schreibt er den Inhalt dieses Templates in den Ausgabebaum [Harold/Means 2001, S. 139]. Im Beispiel passt das erste Template (<xsl:template match="Gliederung") auf das Wurzelement des XML-Dokuments aus Abbildung 1. Der Inhalt dieses Templates wird in das Ausgabedokument übernommen. Text wird vom XSLT-Prozessor 1:1 übernommen (so zum Beispiel die HTML-Tags oder die öffnende und schliessende Klammer im Template für das Element "Kapitel"). Mit <xsl:apply-templates/> wird angegeben, dass an dieser Stelle die übrigen Templates angewendet werden sollen. Die weiteren Templates für die Elemente Unterkapitel und Abschnitt wurden im Beispiel der Übersichtlichkeit halber weggelassen. Das vom XSLT-Prozessor ausgegebene HTML-Dokument ist in Abbildung 7 dargestellt.

```
<html>
<head>
<title></title></head>
<body>
<h3>Kapitel 1 ( 8 )</h3>
<h4>Unterkapitel 1</h4>
<h5>Abschnitt 1</h5>
<p>Hier beginnt der Text von Abschnitt 1</p>
</body>
</html>
```

Abbildung 7: HTML-Output

Das Ergebnis der Transformation ist ein vollständiges HTML-Dokument.

Im Stylesheet aus Abbildung 6 werden einige XPath-Ausdrücke verwendet. XPath-Ausdrücke werden in XSLT gebraucht um die jeweils gesuchten Elemente zu identifizieren [Harold/Means 2001, S.139]. Die in Stylesheets am häufigsten verwendeten XPath-Ausdrücke sind Lokalisierungspfade [Tidwell 2001, S. 47]. Der Lokalisierungspfad / wählt den Wurzelknoten des Dokuments (z.B. <xsl:template match="/") aus, / ist ein absoluter Lokalisierungspfad, d.h. er repräsentiert unabhängig vom aktuellen Kontextknoten (der Knoten, den der Parser gerade bearbeitet) immer den Wurzelknoten des Dokuments. Absolute Lokalisierungspfade können auch kombiniert werden so würde zum Beispiel ein

Lokalisierungspfad `"/auto/antrieb"` alle antrieb-Elemente, die Kinderelemente von auto-Elementen sind, repräsentieren. Der Vorteil von absoluten Lokalisierungspfadem ist, dass sie unabhängig vom Kontextknoten Elemente auswählen [Tidwell 2001, S. 49].

Der Wurzelknoten in XPath ist nicht identisch mit dem Wurzelement, er enthält auch die Steuerungsanweisungen und Kommentare, die vor dem Start-Tag des Wurzelements oder nach dem End-Tag des Wurzelementes stehen [Tidwell 2001, S. 44]. Alternativ dazu kann das Wurzelement auch einfach mit seinem Namen (wie im Beispiel `xsl:template match="Gliederung"`) ausgewählt werden. Bei einem Matching mit dem Namen des Elements werden dann allerdings nur die Kinderelemente des Kontextknotens ausgewählt, diese Lokalisierungspfade sind relativ (zum Kontextknoten). Das funktioniert im Falle der Auswahl des Wurzelementes "Gliederung" über seinen Namen deswegen, weil der Kontextknoten in diesem Fall der Wurzelknoten ist (der Wurzelknoten wird vom Parser zuerst bearbeitet) und das Wurzelement "Gliederung" folglich ein Kindelement des Kontextknotens. Mit `<xsl:value of select="Lokalisierungspfad">` wird der Wert eines Elements oder Attributs ausgewählt. So wurde im Beispiel aus Abbildung 6 der Wert des Attributknotens "AnzahlSeiten" des Elementes "Kapitel" ausgewählt mit:

```
<xsl:template match="Kapitel"/>
[...
  <xsl:value-of select="@AnzahlSeiten"/>
```

XPath bietet viele weitere Lokalisierungspfade und andere Mechanismen zur Auswahl von Elementen (Prädikate, Wildcards, vgl. dazu [Tidwell 2001, S. 47ff]).

3.2.2 XSL-FO - XSL Formatting Objects

Eine dritte Möglichkeit der Präsentation von XML-Dokumenten sind XSL Formatting Objects. Ein XML-Dokument wird mit einem XSL-Stylesheet in ein XSL-FO-Dokument umgewandelt und dieses wird anschliessend mit einem Formatierungsprogramm zum Beispiel in ein PDF-Dokument konvertiert [Harold/Means 2001, S. 223ff]. Folgendes Stylesheet (Abbildung 8) könnte benutzt werden um das XML-Dokument aus Abbildung 1 (S. 7) in ein XSL-FO-Dokument umzuwandeln:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:fo="http://www.w3.org/1999/XSL/Format">

<xsl:template match="/">
  <fo:root>

    <fo:layout-master-set>
      <fo:simple-page-master master-name="gliederung">
```

```

        <fo:region-body/>
    </fo:simple-page-master>
</fo:layout-master-set>

    <fo:page-sequence master-reference="gliederung" >
        <fo:flow>
            <xsl:apply-templates/>
        </fo:flow>
    </fo:page-sequence>
</fo:root>
</xsl:template>

<xsl:template match="Kapiteltitel|Unterkapiteltitel|Abschnittstitel|Abschnittstext">
    <fo:block><xsl:apply-templates/></fo:block>
</xsl:template>

</xsl:stylesheet>

```

Abbildung 8: Stylesheet zur Umwandlung eines XML-Dokuments in ein XSL-FO-Dokument

Das beim Anwenden dieses Stylesheets erzeugte XSL-FO-Dokument zeigt Abbildung 9.

```

<?xml version="1.0" encoding="UTF-8"?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
    <fo:layout-master-set>
        <fo:simple-page-master master-name="gliederung">
            <fo:region-body/>
        </fo:simple-page-master>
    </fo:layout-master-set>
    <fo:page-sequence master-reference="gliederung">
        <fo:flow flow-name="xsl-region-body">
            <fo:block>Kapitel 1</fo:block>
            <fo:block>Unterkapitel 1</fo:block>
            <fo:block>Abschnitt 1</fo:block>
            <fo:block>Hier beginnt der Text von Abschnitt 1</fo:block>
        </fo:flow>
    </fo:page-sequence>
</fo:root>

```

Abbildung 9: Das erzeugte XSL-FO-Dokument

Ein XSL-FO-Dokument besteht aus zwei Teilen:

- einem Vorspann, der das Layout für die zu füllenden Seiten festlegt (fo:layout-master-set)
- dem Inhalt für die Seiten (fo:page-sequence)

Zunächst wird im Beispiel eine einfache Seite definiert (mit fo:layout-master-set), die nur einen Hauptbereich (region-body) enthält. Danach wird mit fo:page-sequence eine Seite erzeugt. Der Text dieser Seite ist alles, was zwischen den Tags <fo:flow> und </fo:flow> eingeschlossen ist [Becker 2001].

Das erzeugte XSL-FO-Dokument könnte nun mit einem Formatierungsprogramm (etwa FOP [FOP 2002]) beispielsweise in ein PDF-Dokument umgewandelt werden.

Diese Möglichkeit der Präsentation von XML-Dokumenten ist im Gegensatz zu den zwei weiter oben erläuterten Möglichkeiten nicht besonders für Webseiten geeignet. Sie eignet sich eher für gedruckte Materialien (Bücher, Zeitschriften etc.) [Harold/Means 2001, S. 240].

4. Anwendungsbeispiel: Die flexible Darstellung von IDML-Reporting-Dokumenten

In diesem Kapitel soll (aufbauend auf die in den beiden vorangegangenen Kapitel erarbeiteten Grundlagen) gezeigt werden wie IDML-Dokumente mit Hilfe verschiedener XSL-Stylsheets flexibel dargestellt werden können.

4.1 IDML Allgemein

IDML ist eine Initiative zur Schaffung eines Datenaustauschstandards (basierend auf XML) für humanitäre Organisationen. Das Ziel dieser Initiative ist die Schaffung einer Markup Sprache (IDML) zur Vereinfachung des Austausches von Informationen innerhalb humanitärer Organisationen und zwischen humanitären Organisationen, ihren verschiedenen Partnern und der Öffentlichkeit [Bellanet 2002].

Das Schema von IDML definiert eine Struktur für Informationen über humanitäre Projekte [Hüsemann 2001]. IDML stellt eine Menge von Tags bereit, die dazu dienen Dokumente, die Projekte von humanitären Organisationen beschreiben, auszuzeichnen. Auf das Schema und weitere Einzelheiten von IDML soll im Rahmen dieser Arbeit nicht eingegangen werden. Informationen zu IDML finden sich in [IDML 2002], das Schema von IDML ist in [IDMLS 2001] beschrieben.

4.2 IDML-Reporting

IDML-Reporting ist eine Erweiterung von IDML. IDML-Reporting stellt eine Menge von Tags zur Verfügung, die zur Auszeichnung von Projektreports dienen. Das Wurzelement des Schemas ist reportsAndEvaluations, es ist im Schema wie folgt deklariert [IDMLR 2002]:

```
<element name="reportsAndEvaluations">
  <annotation>
    <documentation>The root element can contain any number of reports or
evaluations.</documentation>
  </annotation>
  <complexType>
    <sequence>
      <element name="reportAndEvaluation" type="idmlReporting:reportType"
maxOccurs="unbounded">
        <annotation>
          <documentation>A report or evaluation has a language, origin and creation date. If the
same report exists in different languages every language is a "report" element.</documentation>
        </annotation>
      </element>
    </sequence>
```

reportsAndEvaluations enthält eine Sequenz von reportAndEvaluation-Elementen. Abbildung 10 zeigt die für das weiter unten folgende Beispieldokument relevanten Teile dieses Elementes.

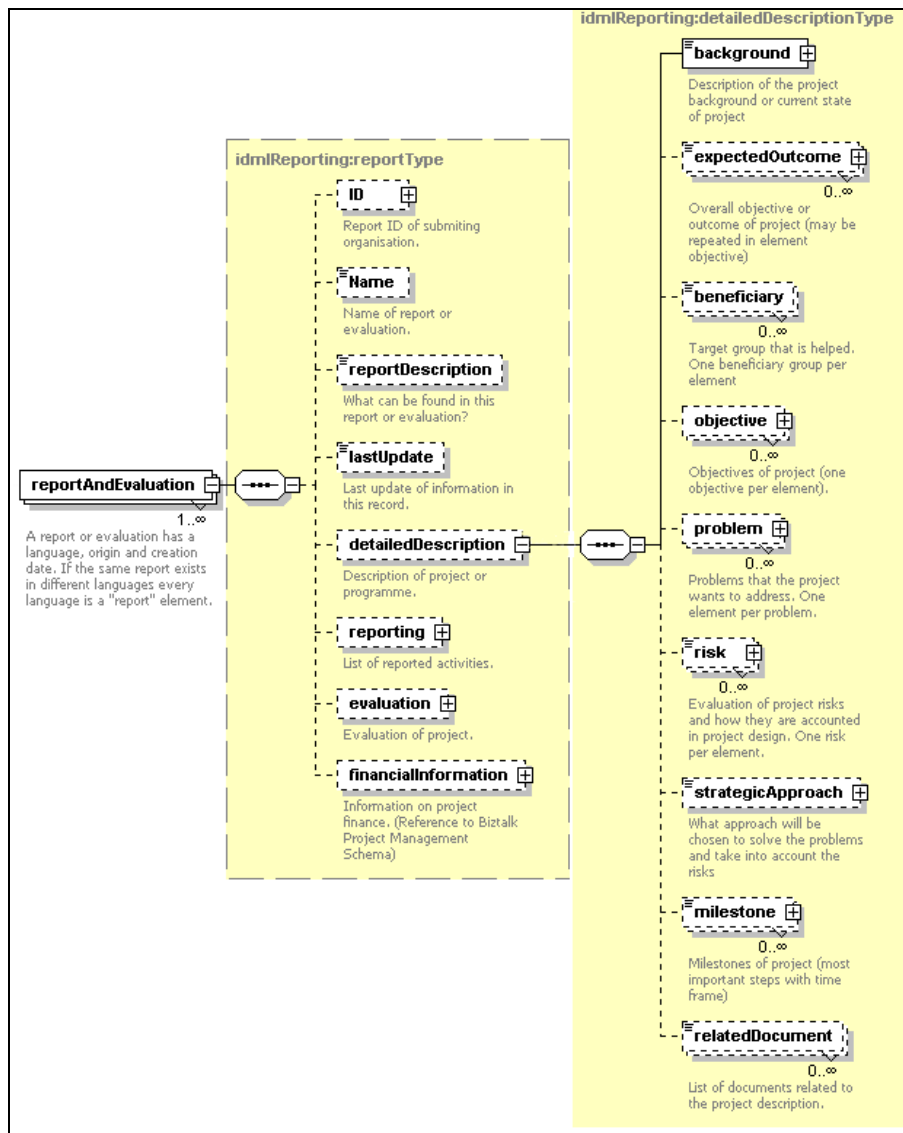


Abbildung 10: Überblick IDML-Reporting Schema

In dieser Abbildung ist nur das Element **detailedDescription** erweitert worden, sämtliche Elemente die in der Abbildung mit einem +-Symbol versehen sind enthalten selber wieder Elemente. Weiter ist aus der Abbildung ersichtlich wie oft die einzelnen Elemente vorkommen dürfen (beispielsweise steht beim Element **relatedDocument** in der rechten unteren Ecke "0..∞"). Alle in der Abbildung gestrichelt umrahmten Elemente sind optional, d.h. sie müssen nicht zwingend in einem Dokument vorkommen. Das IDML-Reporting Schema sowie eine Dokumentation dazu finden sich in [IDMLR 2002].

4.3 Formatierung eines IDML-Reporting-Dokuments

Das Beispieldokument, welches im weiteren mit Hilfe von Stylesheets unterschiedlich dargestellt werden soll, ist ein Projektreport über ein Projekt der Weltbank in Bangkok. Es folgt ein Auszug aus dem Dokument, die eckigen Klammern kennzeichnen ausgelassenen Text (der mit den verschiedenen Stylesheets bearbeitete Teil des Dokuments findet sich im Anhang):

```
<reportAndEvaluation lang="EN" reportOrigin="World Bank">
  <Name>Thailand:
  Bangkok Air Quality Management Project
  </Name>
  <reportDescription>GEF Concept Note</reportDescription>
  <detailedDescription>
    <background>The concentration
    [...]
  </background>
```

Mit einem ersten Beispielstylesheet (Abbildung 11) soll der Text, der im background-Element enthalten ist, in der Schriftart Arial dargestellt werden. Die übrigen Teile des Beispieldokuments werden nicht in die HTML-Datei, die das Stylesheet erzeugt, aufgenommen.

Der Prolog und die Namensraumdeklarationen wurden im Stylesheet der Übersichtlichkeit halber weggelassen.

```
<xsl:template match="/">
  <html>
    <head />
    <body>
      <xsl:for-each select="idmlReporting:reportsAndEvaluations">
        <xsl:for-each select="idmlReporting:reportAndEvaluation">
          <xsl:for-each select="idmlReporting:detailedDescription">
            <xsl:for-each select="idmlReporting:background">
              <span style="font-family:Arial; font-size:small">
                <xsl:apply-templates />
              </span>
            </xsl:for-each>
          </xsl:for-each>
        </xsl:for-each>
      </xsl:for-each>
    </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

Abbildung 11: Beispielstylesheet 1

Das background-Element wird in dem vorgestellten Stylesheet mit xsl:for-each selektiert. Mit xsl:for-each kann ein Template auf jeden Knoten einer (mit dem select-Attribut) angegebenen Knotenmenge angewendet werden [Harold/Means 2001]. In diesem Fall bewegt sich der Parser durch die geschachtelten xsl:for-each-Anweisungen in der Baumstruktur des XML-Dokuments abwärts bis er auf das background-Element stösst. Auf dieses wird das Template angewendet, d. h. dass der Inhalt des background-Elements mit einem -Tag umschlossen

wird, der die Schriftart (Arial) und die Schriftgröße (small) vorgibt. (der ``-Tag ist ein CSS-Befehl). Das Stylesheet bewirkt also nichts weiter, als dass das `background`-Element des Reports formatiert wird. Interessant ist in diesem Fall allerdings auch, was das Stylesheet nicht tut: Das Stylesheet übernimmt nur den Text, der innerhalb der `background`-Tags steht, in die erzeugte HTML-Datei. Die übrigen Teile des Reports werden nicht übernommen. Es ist also möglich Informationen, die im ursprünglichen XML-Dokument vorhanden waren, im erzeugten HTML-Dokument "auszublenden". HTML-Dokumente, die mittels eines XSLT-Stylesheets aus XML-Dokumenten erzeugt werden, können folglich nicht nur unterschiedlich formatiert werden, sondern auch unterschiedliche Inhalte haben.

Beispielstylesheet 2 (Abbildung 12) setzt den Inhalt des `namen`-Elements als Titel für das erstellte HTML-Dokument, ausserdem setzt es einen Zwischentitel "Background" und unterlegt den Textinhalt des `background`-Elements mit einer Hintergrundfarbe (silber). Im weiteren werden Schriftarten und -größen für die einzelnen Elemente vorgegeben.

```
<xsl:template match="/">
  <html>
    <head />
    <body>
      <xsl:for-each select="idmlReporting:reportsAndEvaluations">
        <xsl:for-each select="idmlReporting:reportAndEvaluation">
          <xsl:for-each select="idmlReporting:Name">
            <span style="font-size:large; text-decoration:underline">
              <xsl:apply-templates />
            </span>
          </xsl:for-each>
          <xsl:for-each select="idmlReporting:detailedDescription">
            <span style="text-decoration:underline">Background</span>
            <xsl:for-each select="idmlReporting:background">
              <span style="background-color:silver">
                <xsl:apply-templates />
              </span>
            </xsl:for-each>
          </xsl:for-each>
        </xsl:for-each>
      </xsl:for-each>
    </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

Abbildung 12: Beispielstylesheet 2

Die beiden Beispielstylesheets wurden mit dem Tool XML SpyXSLT Designer aus der XML Spy Suite erstellt [XMLSpy 2002]. Abbildung 13 zeigt wie das Beispieldokument mittels Beispielstylesheet 2 in einem Browser dargestellt aussieht.

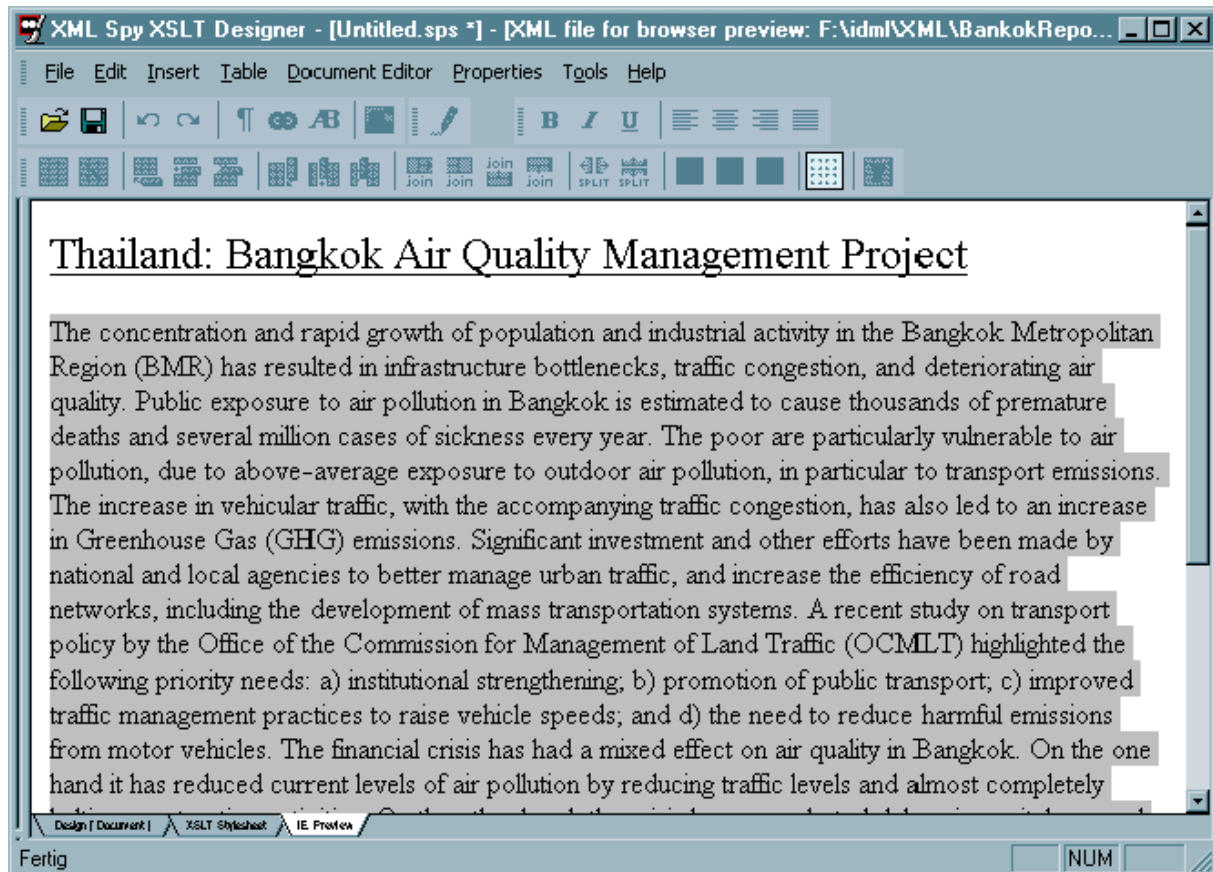


Abbildung 13 : Darstellung im Browser

Der "Umweg" über HTML-Dokumente ist zur Zeit für die Präsentation von XML-Dokumenten mit Sicherheit die beste Lösung. Sie erlaubt eine flexiblere Darstellung als die direkte Präsentation von XML-Dokumenten in Browsern mittels CSS. Dabei sei insbesondere auf die Möglichkeit der "Ausblendung" von bestimmten Teilen eines XML-Dokuments (die weiter oben beschrieben wurde) verwiesen. Desweiteren gibt es bei der direkten Präsentation im Browser keine Möglichkeit die Reihenfolge der einzelnen Textelemente zu verändern. Im Gegensatz dazu erlaubt die vorgestellte Lösung eine beliebige Zusammenstellung der im XML-Dokument enthaltenen Informationen. Zwei weitere im Anhang aufgeführte Stylesheets zeigen andere Möglichkeiten zur Formatierung des Beispieldokuments.

5. Schlussbemerkung

Auf den ersten Blick scheint die Umwandlung von XML-Dokumenten in HTML-Dokumente ein Widerspruch zu sein. Das ist aber nicht der Fall. HTML wird in diesem Fall nur als Präsentationsformat gewählt, weil es durch gängige Browser besser unterstützt wird als die direkte Darstellung von XML-Dokumenten. Die Präsentation mittels HTML-Dokumenten macht die Vorteile von XML-Dokumenten nicht zunichte, da die XML-Dokumente als Quelle für die erzeugten HTML-Dokumente dienen. So können die XML-Daten unabhängig davon wie sie präsentiert werden, einfach wiederverwendet und verändert werden. Ändert sich die XML-Quelle können die zur Präsentation erstellten HTML-Dokumente sehr einfach angepasst werden (erneute Erzeugung mittels den dafür geschriebenen Stylesheets). Auch das Aussehen der erzeugten HTML-Dokumente kann auf einfachste Art verändert werden, in dem die Stylesheets angepasst werden mittels derer die HTML-Dokumente erzeugt wurden. HTML-Seiten, die aus XML-Quellen erzeugt wurden, sind also wesentlich einfacher wartbar als direkt erzeugte HTML-Seiten.

XML ist ein mächtiges Instrument zur Präsentation von Informationen auf dem Web. XSLT und XSL-FO bieten eine Vielzahl von Möglichkeiten zur Darstellung von XML-Dokumenten.

Bereits heute setzen viel Software-Entwickler auf XML als offenes Datenaustauschformat (beispielsweise Microsoft mit der .Net-Strategie). XML entwickelt sich mit grosser Dynamik weiter, was an verschiedensten Standards für Erweiterungen der Sprache, die vom W3C verabschiedet wurden, erkennbar ist (XMLQuery, XMLKey Management, XMLEncryption u. a.). Auf die weitere Entwicklung und Verbreitung von XML darf man gespannt sein!

A.1 Projektreport

([...] kennzeichnet ausgelassenen Text)

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.2 (http://www.xmlspy.com) by Hanswurst (Globobruell) -->
<reportsAndEvaluations origin="12344" xmlns="F:\idml\Schemas\idmlReporting.xsd"
xmlns:idml="F:\idml\Schemas\IDML_091_local.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:schemaLocation="F:\idml\Schemas\idmlReporting.xsd
F:\idml\Schemas\idmlReporting.xsd">
  <reportAndEvaluation lang="EN" reportOrigin="World Bank">
    <Name>Thailand:
Bangkok Air Quality Management Project
</Name>
    <reportDescription>GEF Concept Note</reportDescription>
    <detailedDescription>
      <background>The concentration and rapid growth of population and industrial activity in the
Bangkok Metropolitan Region (BMR) has resulted in infrastructure bottlenecks, traffic congestion, and
deteriorating air quality. Public exposure to air pollution in Bangkok is estimated to cause thousands of
premature deaths and several million cases of sickness every year. The poor are particularly
vulnerable to air pollution, due to above-average exposure to outdoor air pollution, in particular to
transport emissions. The increase in vehicular traffic, with the accompanying traffic congestion, has
also led to an increase in Greenhouse Gas (GHG) emissions.
```

Significant investment and other efforts have been made by national and local agencies to better manage urban traffic, and increase the efficiency of road networks, including the development of mass transportation systems. A recent study on transport policy by the Office of the Commission for Management of Land Traffic (OCMLT) highlighted the following priority needs: a) institutional strengthening; b) promotion of public transport; c) improved traffic management practices to raise vehicle speeds; and d) the need to reduce harmful emissions from motor vehicles.

The financial crisis has had a mixed effect on air quality in Bangkok. On the one hand it has reduced current levels of air pollution by reducing traffic levels and almost completely halting construction activities. On the other hand, the crisis has exacerbated delays in capital renewal and investment in cleaner technologies. The Government budget for environmental investment has been reduced since the beginning of the crisis, and it is estimated that, without specific targeted interventions aimed at curbing emissions, air pollution will surpass pre-crisis levels in Bangkok soon after the economic recovery. Similar conclusions are drawn by the Bank's Bangkok UTS as far as traffic levels, congestion and trends in vehicle growth are concerned.

The proposed Bangkok Air Quality Management Project (AQMP) is a cross-sectoral operation involving environment, transport and urban dimensions that is intended to be a quick response to the increase in air pollution. The project design focuses on key issues such as the improvement of bus passenger transport, design and implementation of long-term transport management plans, and the strengthening of local government capabilities, to facilitate the decentralization process.

```
</background>
```

```
[...]
```

```
</reportsandEvaluations>
```

A.2 Stylesheet 1

```

<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:n1="F:\idml\Schemas\idmlReporting.xsd" xmlns:idml="F:\idml\Schemas\IDML_091_local.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <xsl:template match="/">
    <html>
      <head/>
      <body>
        <xsl:for-each select="n1:reportsAndEvaluations">
          <xsl:for-each select="n1:reportAndEvaluation">
            <br/>
            <xsl:for-each select="n1:Name">
              <span style="font-family:Arial; font-size:large; text-align:center; text-
decoration:underline">
                <span style="background-color:white; font-family:Arial; font-size:large; text-
align:center; text-decoration:underline">
                  <xsl:apply-templates/>
                </span>
              </span>
            </xsl:for-each>
            <br/><xsl:for-each select="n1:reportDescription">
              <span style="font-family:Arial; font-size:medium; text-align:center">
                <span style="font-family:Arial; font-size:medium; text-align:center">
                  <xsl:apply-templates/>
                </span>
              </span>
            </xsl:for-each><br/>
            <br/>
            <span style="background-color:aqua; font-family:Arial; font-size:small; text-
decoration:underline">Background</span>
            <br/>
            <br/>
            <br/>
            <xsl:for-each select="n1:detailedDescription">
              <xsl:for-each select="n1:background">
                <span style="font-family:Antigoni">
                  <span style="font-family:Times New Roman">
                    <xsl:apply-templates/>
                  </span>
                </span>
              </xsl:for-each>
              <br/>
              <br/>
              <br/>
              <span style="background-color:aqua; font-family:Arial; font-size:small; text-
decoration:underline">Expected Outcome</span>
              <br/>
              <br/>
              <br/>
              <xsl:for-each select="n1:expectedOutcome">
                <xsl:apply-templates/>
              </xsl:for-each>
              <br/>
              <br/>
              <br/>
              <span style="background-color:aqua; font-family:Arial; font-size:small; text-
decoration:underline">Objectives</span>
              <br/>

```

Anhang

```
<br/>
<br/>
<xsl:for-each select="n1:objective">
  <xsl:for-each select="n1:objectiveName">
    <span style="font-family:Arial; font-size:small; text-
decoration:underline">
      <span style="font-family:Times New Roman; font-size:small; text-
decoration:underline">
        <xsl:apply-templates/>
      </span>
    </span>
  </xsl:for-each>
<br/>
<br/>
<xsl:for-each select="n1:objectiveDescription">
  <span style="font-family:Times New Roman">
    <xsl:apply-templates/>
  </span>
</xsl:for-each>
<br/>
<br/>
</xsl:for-each>
<br/>
<span style="background-color:aqua; font-family:Arial; font-size:small; text-
decoration:underline">Budget</span>
<br/>
<br/>
<xsl:for-each select="/">
  <html>
    <head/>
    <body>
      <xsl:for-each select="n1:reportsAndEvaluations">
        <xsl:for-each select="n1:reportAndEvaluation">
          <xsl:for-each select="n1:financialInformation">
            <xsl:for-each select="n1:budget">
              <xsl:for-each select="n1:comment">
                <xsl:apply-templates/>
              </xsl:for-each>
            </xsl:for-each>
          </xsl:for-each>
        </xsl:for-each>
      </xsl:for-each>
    </body>
  </html>
</xsl:for-each>
<br/>
<br/>
<br/>
<span style="background-color:aqua; color:black; font-family:Arial; font-
size:small; text-decoration:underline">Problems</span>
<br/>
<br/>
<xsl:for-each select="n1:problem">
  <xsl:for-each select="n1:problemDescription">
    <xsl:apply-templates/>
  </xsl:for-each>
</xsl:for-each>
<br/>
<br/>
<span style="background-color:aqua; font-family:Arial; font-size:small; text-
decoration:underline">Proposed Solution</span>
<br/>
```

Anhang

```
<br/>
<xsl:for-each select="n1:problem">
  <xsl:for-each select="n1:proposedSolution">
    <xsl:apply-templates/>
  </xsl:for-each>
</xsl:for-each>
<br/>
<br/>
<br/>
<span style="background-color:aqua; font-family:Arial; font-size:small; text-
decoration:underline">Risks</span>
<br/>
<br/>
<xsl:for-each select="n1:risk">
  <xsl:apply-templates/>
</xsl:for-each>
</xsl:for-each>
</xsl:for-each>
</xsl:for-each>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

A.3 Stylesheet 2

```

<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:n1="F:\idml\Schemas\idmlReporting.xsd" xmlns:idml="F:\idml\Schemas\IDML_091_local.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <xsl:template match="/">
    <html>
      <head/>
      <body>
        <xsl:for-each select="n1:reportsAndEvaluations">
          <xsl:for-each select="n1:reportAndEvaluation">
            <xsl:for-each select="n1:Name">
              <span style="font-family:Arial Black; text-decoration:underline">
                <xsl:apply-templates/>
              </span>
            </xsl:for-each>
            <br/>
            <br/>
            <span style="background-color:red; text-decoration:overline">Problems</span>
            <br/>
            <br/>
            <xsl:for-each select="n1:detailedDescription">
              <xsl:for-each select="n1:problem">
                <xsl:for-each select="n1:problemDescription">
                  <xsl:apply-templates/>
                </xsl:for-each>
                <br/>
              </xsl:for-each>
              <br/>
              <span style="background-color:green; text-decoration:overline">Possible
Solutions</span>
              <br/>
              <br/>
              <xsl:for-each select="n1:problem">
                <xsl:for-each select="n1:proposedSolution">
                  <xsl:apply-templates/>
                </xsl:for-each>
              </xsl:for-each>
            </xsl:for-each>
          </xsl:for-each>
        </xsl:for-each>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>

```

Literatur und Quellen

- [Astra 2002] Astra Datentechnik GmbH, IT-Glossar
verfügbar: <http://www.astra-gmbh.de/astrawebsite/glossar/glossar.html>
Zugriffsdatum: 25.03.2002
- [Becker 2001] Becker, Oliver: XSL Formatting Objects
verfügbar: www.informatik.hu-berlin.de/~obecker/Lehre/XML-Kurs/xml6.html
Zugriffsdatum: 27.03.2002-05-07
- [Bellanet 2002] Bellanet: IDML Initiative
verfügbar: <http://www.bellanet.org/xml/about.cfm>
Zugriffsdatum: 03.04.2002
- [Bosak/Bray 1999] Bosak, Jon; Bray, Tim: Mehr Tempo auf der Datenautobahn, Spektrum der Wissenschaft, Ausgabe Juli 1999, Seite 74.
- [Dönhölder 1998] Dönhölder, Kuno: Das Web automatisieren mit XML.
verfügbar: <http://members.aol.com/xmldoku/>
Zugriffsdatum: 26.03.2002
- [FOP 2002] Formating Objects Processor Version 0.20.3, Apache XML Project
verfügbar: <http://xml.apache.org/fop/index.html>
Zugriffsdatum: 03.04.2002
- [Günther 1997] Günther, Richard: Historische Entwicklung des HTML-Standards
verfügbar: <http://www.tat.physik.uni-tuebingen.de/~rguenth/html/node9.html>
Zugriffsdatum: 27.03.2002
- [Harold/Means 2001] Harold, Elliott R.; Means, W. Scott: XML in a nutshell, O'Reilly, 1. Aufl., Köln 2001
- [Hofmann/Raitelhuber 1998] Hofmann, Thomas; Raitelhuber, Ursula: SGML/XML.
verfügbar : <http://th-o.de/sgml/sgmlv.htm>
Zugriffsdatum: 20.03.2002
- [Hoven/Liebrecht 2001] Hoven, Philipp; Liebrecht, Mike: DTD & XML-Schema
verfügbar: www.uni-paderborn.de/cs/ag-taentzer/Dtdschema.pp
Zugriffsdatum: 01.04.2002
- [Hüsemann 2001] Hüsemann, Stefan: Einführung in XML, Vorlesungsunterlagen der Vorlesung Informationssysteme A an der Universität Fribourg, 2001
- [IDML 2002] IDML Initiative
verfügbar: <http://www.idmlinitiative.org/index2.cfm>
Zugriffsdatum: 03.05.2002
- [IDMLR 2002] Reporting Schema for humanitarian Projects
verfügbar: <http://iiufpc06.unifr.ch/Huesemann/Diss>
Zugriffsdatum: 15.05.2002

- [IDMLS 2001] AIDA IDML Schema version 0.9 - draft
verfügbar: <http://www.idmlinitiative.org/index2.cfm>
Zugriffsdatum: 03.05.2002
- [Schädler 2001] Schädler, Thomas, XML als Datenaustauschformat, Vorlesungsunterlagen der Vorlesung Informationssysteme A an der Universität Fribourg, 2001
- [Stroustrup 2000] Stroustrup, Bjarne: Die C++ Programmiersprache, Addison-Wesley, 4. Aufl., München 2000.
- [St. Laurent 1998] St. Laurent, Simon: XML a primer, MIS Press, Foster City 1998
- [Tidwell 2001] Tidwell, Doug: XSLT: Mastering XML Transformations, O'Reilly, 1. Aufl., Sebastopol 2001
- [Unicode 2002] The Unicode Standard
verfügbar: <http://www.unicode.org/>
Zugriffsdatum: 09.05.2002
- [W3CH 2000] World Wide Web Consortium, XHTML™ 1.0: The Extensible HyperText Markup Language
verfügbar: <http://www.w3.org/TR/xhtml1/>
Zugriffsdatum: 27.03.2002
- [W3CP 1999] World Wide Web Consortium, XML Path Language (XPath)
verfügbar: <http://www.w3.org/TR/xpath>
Zugriffsdatum: 07.04.2002
- [W3CR 1999] World Wide Web Consortium, Resource Description Framework
verfügbar: <http://www.w3.org/RDF/>
Zugriffsdatum: 04.04.2002
- [W3CS 2001] World Wide Web Consortium, XML Schema
verfügbar: <http://www.w3.org/TR/xmlschema-0>
Zugriffsdatum: 30.03.2002
- [W3CX 1998] World Wide Web Consortium, Extensible Markup Language 1.0 Recommendation.
verfügbar: <http://www.w3.org/TR/1998/REC-xml-19980210>
Zugriffsdatum: 26.03.2002
- [XMLSpy 2002] XML Spy
verfügbar: <http://www.xmlspy.com/>
Zugriffsdatum: 15.05.2002