Studiengang: Master of Science an der HAW-Hamburg
Vorlesung: Anwendungen 1
Vortrag von: Thomas Steinberg

HAW HAMBURG

Zusammenfassung des Vortrags Syntaktische Transformation

Thomas Steinberg, 08. Juli 2005
thomas.steinberg@informatik.haw-hamburg.de

Dieser Vortrag wurde in der Vorlesung Anwendung I des Studiengangs Master of Science von Thomas Steinberg am 04.05.2005 gehalten. Er besteht aus einer allgemeinen Einführung in das Thema “Syntaktische Transformation“, der Implementierung von Transformationen anhand von XML, der Motivation und einem konkreten Teilnehmerangebot für das Projekt „Ferienclub“ des nächsten Semesters.

HAW HAMBURG

Inhaltsverzeichnis:

Allgemeines

Semantik vs. Syntax

In diesem Kapitel werden als erstes die Definitionen für die Ausdrücke Semantik, Syntax und Transformation aufgeführt, um auf die fachliche Richtung des Vortrags einzustimmen.

Transformation

Im Kapitel Transformation wird anhand von vielen Beispielen gezeigt, wo Transformationen in unserer Umwelt getätigt werden und welche Arten von Transformationen es gibt. Es wird im allgemeinen auf die Gründe und die Probleme von Transformationen eingegangen. An dieser Stelle werden die einzelne Unterpunkte aus den Folien noch mal kurz ergänzt:

Anwendung von Transformationen: Transformationen begegnen uns in unserer Umwelt überall. In der *Mathematik* kennt man Transformationen in Form der mathematischen Abbildung oder Funktion, aber auch in Koordinatentransformationen(affine Transformation) oder in den Funktionstransformationen(Fouriertransformation).Im Bereich *Elektrotechnik* ist Transformation bekannt in der Form der Spannungstransformation mit einem Transformator. Weiterhin ist Transformation auch in der *Informatik* in vielen Varianten eingeführt, um nur einige zu nennen, wäre da z.B. die Modelltransformation, die Codetransformation oder die Typentransformation. Es gibt viele andere Bereiche in denen Transformationen angewendet werden, die an dieser Stelle nicht weiter vertieft werden, da es den Rahmen der Zusammenfassung sprengen würde.

Arten der Transformation: Es gibt eigentlich nur zwei Arten der Transformation. Die erste Variante ist die *verlustbehaftete Transformation*. Diese Art der Transformation verliert Informationen bei der Transformation von einem Zustand zum Anderen. Somit ist sie nur in eine Richtung gültig und einer mathematischen Ableitung gleich zu setzen.
Die zweite Variante ist die *wahrheitswerterhaltende Transformation*. Bei dieser Transformationsart gehen keine Informationen verloren, somit ist sie in beide Richtungen gültig und anwendbar.

Gründe für Transformation: Es gibt viele Gründe für Transformationen, damit man nur einen kleinen Einblick bekommt wurde die Informatik als Bereich gewählt um einige Beispiele vorzustellen. Wie bereits erwähnt gibt es in der Informatik viele Anwendungsfälle für die Transformation, im Vortrag wurden drei Teilbereiche ausgewählt und vorgestellt. Als erstes wäre da die *Codetransformation* die Quellcode in eine andere Programmiersprache oder einen anderen Dialekt transformiert. Des weiteren gibt es die *Datentypentransformation* die Datentypen in andere Datentypen transformiert. Als dritten Teilbereich wurde die *Konvertierung* vorgestellt, diese ermöglicht die Überführung einer Datei von einem Dateiformat in ein anderes.

Probleme mit Transformationen: Transformationen bringen auch Probleme mit sich. Da die einzelnen Strukturen und/oder Formate von unterschiedlichen Personen entwickelt wurden, die unterschiedliche Denkweisen und unterschiedliche Konzepte verfolgten, kommt es zu Konflikten. Diese müssen gelöst werden, um eine saubere Transformation durchzuführen zu

können, um dabei so wenig wie möglich an Informationen zu verlieren. Einer der Konflikte wäre z.B. die unterschiedliche Baumstruktur(siehe Abbildung 1).

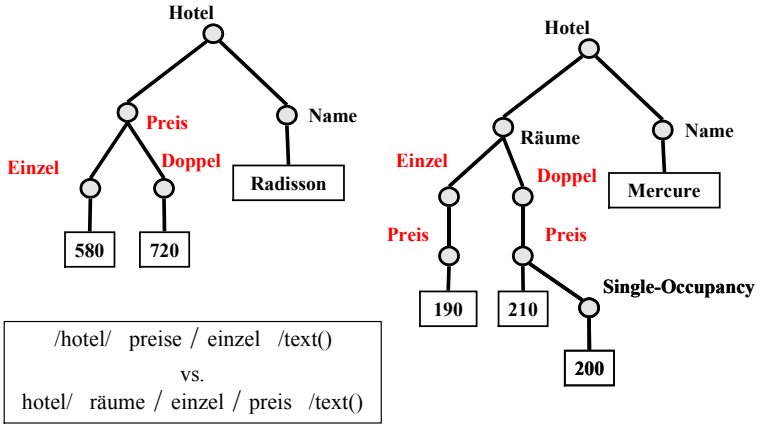

Abbildung 1

Ein anderes Problem wären da Namenskonflikte die in zwei Varianten auftreten können. Die erste Variante sind die sogenannten *Synonyme*. Diese Variante zeigt zwei identische bzw. semantisch äquivalente Objekte(Relationen, Attribute) mit unterschiedliche Namen (z.B. Kumpel == Freund). Die zweite Variante sind die sogenannten *Homonyme*. Diese Variante zeigt zwei unterschiedliche Objekte, aber diesmal mit demselben Namen (z.B. Bank [Geldbank] ≠ Bank [Sitzbank]).

Motivation und Vision für das Projekt

Am Ende des Kapitels wird ein Szenario aus dem Ferienclub dargestellt, das die unterschiedlichen Akteure und ihre individuellen Dokumente darstellt. Ziel des Szenarios ist es ein Gefühl zu bekommen, wo es in einem Ferienclub überall nötig ist Transformationen anzuwenden und welche Probleme auf die Projektteilnehmer im nächsten Semester zukommen.

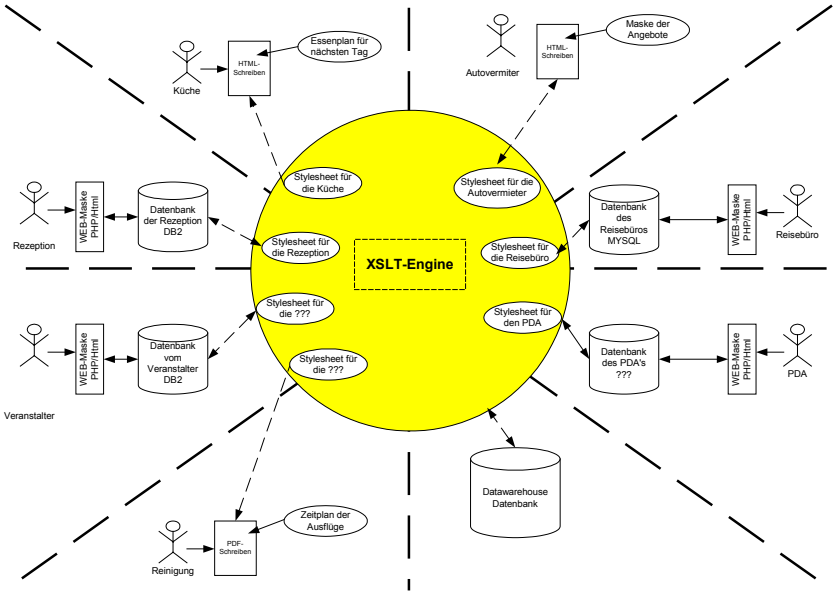

Abbildung 2

Trägersprachen

Nach dem allgemeinen Teil der "Syntaktischen Transformation", wird jetzt das Thema konkretisiert. Die Präsentation wurde in den Bereich Semantic Web eingruppiert, weil die Syntaktische Transformation als Fundament für weitere Techniken und Anwendungen des Semantic Webs dienen soll. Um hier kurz vorzugreifen, ist das experimentieren mit Ontologien, die im Vortrag von Artem Khvat behandelt wurden, das Ziel um einen vollautomatische Transformationsservice zu entwickeln.
Es lag nahe eine Trägersprache für Transformationen zu wählen die sich schon im "Semantic Web Stack"(siehe Abbildung 3)bewährt hat. Das Kapitel „Trägersprachen für die Transformation" wird am Beispiel von XML beschrieben. Das auf fast der untersten Ebene vom Modell liegt.

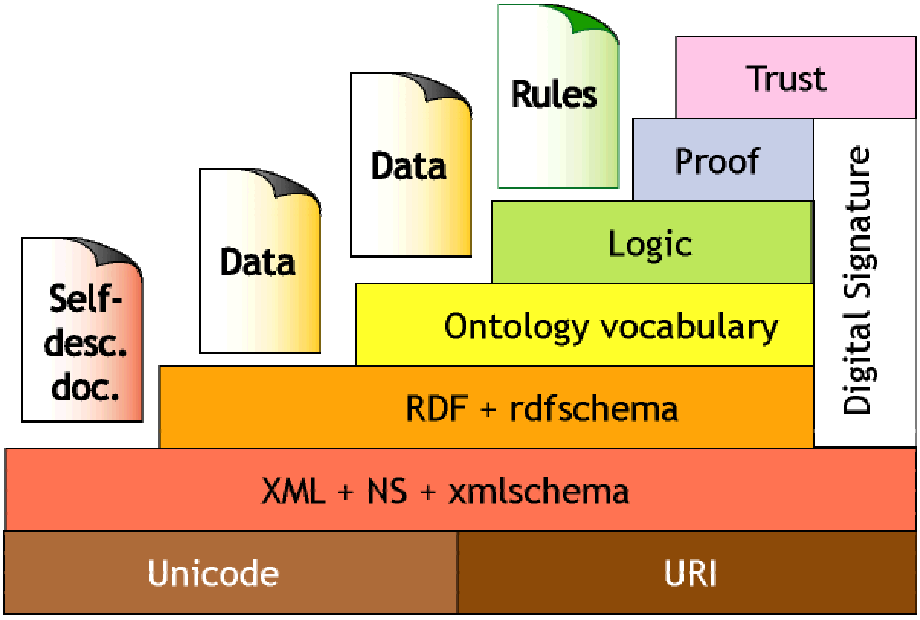

Abbildung 3

Als Einführung wird ein Rundumblick gegeben, was XML ist und welche Instrumente hilfreich waren, dass sich XML so schnell als Markup Sprache etabliert hat.

Im zweiten Unterkapitel wird auf XSL(T) eingegangen, welches auf XML basiert und für die eigentliche Transformation von XML Dokumenten verantwortlich ist.

Im dritten Unterkapitel werden die Möglichkeiten vorgestellt die Java bietet, um mit XML Dokumenten umzugehen. Im Vortrag werden zwei auf Java basierende Parser vorgestellt, ein XSLT-Prozessor zur Transformation in einfache Formate und das Java-Projekt „FOP" zur Transformation in komplexe Formate. Zur Visualisierung der Daten(Knoten) wird zusätzlich auf das Java-Projekt Batik in Verbindung mit SVG kurz eingegangen.

XML (Extensible Markup Language)

XML (Extensible Markup Language) ist ein Produkt des W3C Konsortium das ein offenes und freies Konzept zum Speichern von Daten liefert. Es liefert wie der Name schon sagt die Möglichkeit die Sprache mit selbstgewählten Elementen zu erweitern und den eignen Bedürfnissen anzupassen.

XML basiert auf den Standard von SGML und arbeitet mit sogenannten “Tags“ die die Inhalte des Dokuments festhalten bzw. wiedergeben. Ein mögliches XML-Element wäre z.B.: <nachname>Meier</nachname> wobei das Start-Tag (<nachname>) und das End-Tag (</nachname>) den eigentlichen Inhalt des Elementes einkleiden. Dabei entsteht eine strukturierte Sammlung von Elemente, die dann gemeinsam ein XML Dokument ausmachen. Dieses XML Dokument hat die Struktur eines Baumes, wobei die einzelnen Elemente auch als Knoten bezeichnet werden.

Ein großer Vorteil von XML ist, dass alle Hilfetools auch auf XML basieren und man somit nicht auf teure Zusatzprogramme zur Verarbeitung oder neue Programmierkenntnisse angewiesen ist. Weiterhin hilft XML dazu, dass nicht mehr die geheime Dateiformate die Vermarktung von Software bestimmen, sondern die Qualität und Benutzerfreundlichkeit ausschlaggebend für deren Erfolg wird.
Eine weitere Stärke von XML ist die strikten Trennung zwischen Daten und Layout. Diese Stärke wird später im Kapitel “Visualisierung mit SVG“ noch mal deutlich gemacht.

Die Abbildung 4 zeigt den granulierten Aufbau des XML Standards.

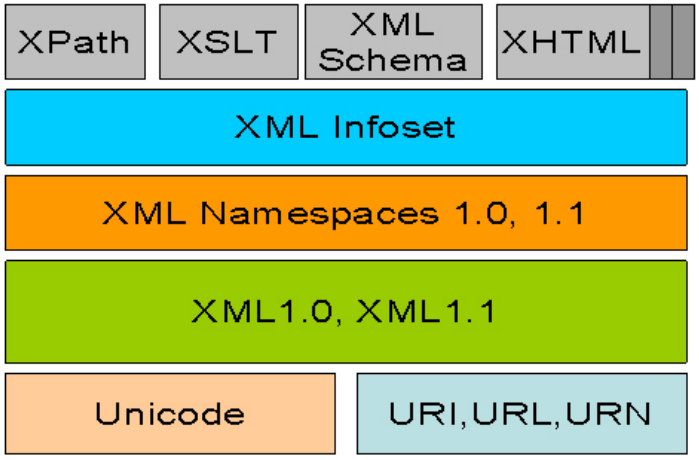

Abbildung 4

DTD (Dokumenttyp Type Definition)

Document Type Definitions (DTDs), das natürlich auch in XML Syntax erstellt ist, wird die Struktur der Daten eines XML Dokuments definiert. In der DTD müssen alle verwendete Elemente in ihrer Struktur beschrieben sein, ansonsten wird der Parser der die Gültigkeit des Dokuments prüft ein "ungültig" herausbekommen. Bei XML Dokumenten spricht man von Wohlgeformtheit und Gültigkeit. Ist ein XML Dokument wohlgeformt, entspricht die Syntax der Daten der XML Notation mit öffnenden und schließenden Klammern (< … >), in der bereits beschriebenen Anordnung. Spricht man von der Gültigkeit eines XML Dokuments, so entspricht die Struktur der angegebenen Elemente und Attribute der Definition in der zugehörigen DTD.

XML-Schema

Man hat recht schnell erkannt, das DTD nicht in jeden Fall ausreicht, somit haben einige Firmen unterschiedliche Schema-Sprachen entwickelt. Das W3C Konsortium hat eine Empfehlung für die Schema Sprachen verabschiedet, die sie einfach nur als XML-Schema bezeichnet. Grundsätzlich bietet XML-Schema gegenüber einer DTD einige Erweiterungen, wie z.B.

- Datentypenprüfung von Elementen und Attributen.
- Eigene Typdefinitionen sind möglich, auch mit Einschränkungen des Wertebereichs.
- Die Kardialitäten für das Auftreten von Elementen kann genau festgelegt werden.
- Die Definition von komplexen Elementen, die wiederum andere komplexe Elemente aufnehmen können.
- Die Bildung von Element- und Attributsgruppen ist möglich.
- Durch Namensräume wird das Zusammenführen von einander unabhängig entwickelten Schemata vereinfacht.
- Ein Schema kann mit Hilfe von XSLT in ein beliebiges anderes Schema umgewandelt werden.

Testen von XML Dokumenten (Parser)

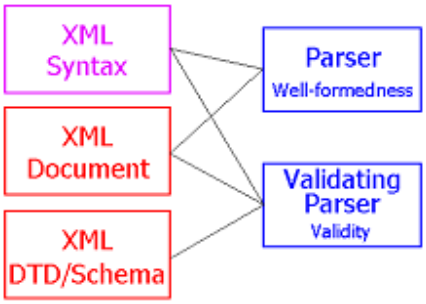

Abbildung 5

Eine der wichtigsten Komponenten bei XML ist der Parser(siehe Abbildung 5). Dieser stellt die Gültigkeit sowie die Wohlgeformtheit eines XML Dokuments fest. Hierzu wird die DTD benötigt um die Strukturen der Elemente abzugleichen. Nach dem Test des Parser, liefert dieser eine Datenstruktur mit der eine entsprechende Anwendung arbeiten kann. Dies kann je nach Anwendungsfall und Umgebung eine andere Datenstruktur sein.
Es gibt eine große Anzahl von Parsern, die Hauptsächlich nach der Bearbeitungsgeschwindigkeit klassifiziert werden. Je häufiger ein Zugriff auf das XML Dokument nötig ist, desto gravierender ist die Auswirkung des verwendeten Parsers auf die Gesamtperformance des Systems. So kann ein schlechter Parser schnell zur Systembremse werden.

Namespaces

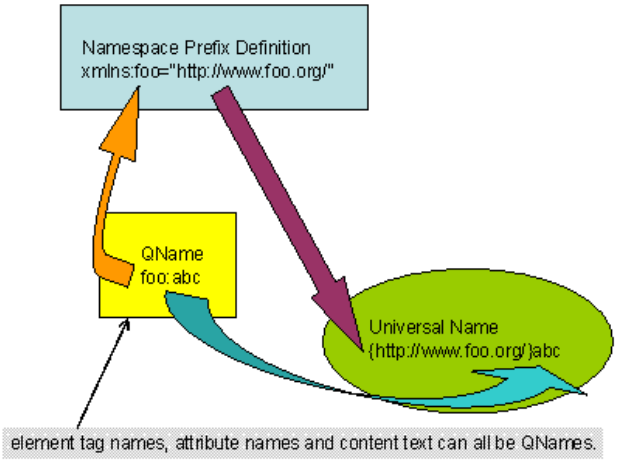

Abbildung 6

Namespaces haben mehre Funktionen, hier sollen sie hauptsächlich als Hilfsmittel, um bestimmte Elemente eines XML Dokumentes mit einer URI zu verknüpfen, eingesetzt werden. Somit wird ein Element nicht mehr einzig und allein durch seinen Namen, sondern zusätzlich durch sein dazugehörigen Namespace identifiziert. Namespaces sind besonders hilfreich bei Verwendung von mehreren DTD's die Konflikte verursachen würden durch *Synonyme* und/oder *Homonyme*. Dies kann vor allem dann geschehen, wenn mehrere Dokumente gleichzeitig verarbeitet werden müssen. Diese Konflikte können durch Namespaces gelöst werden, indem man die verschiedenen Elemente eindeutige unterscheiden kann.

XPath

XPath basiert natürlich auch auf XML und ist vom W3C Konsortium. Die Hauptaufgabe ist XSLT zu unterstützen. Wie erwähnt ist ein XML Dokument in einer Baumstruktur aufgebaut. Mit einem syntaktischen XPath Ausdruck lässt sich zum Beispiel ein Lokalisierungspfad auf einen Knoten im XML Dokument angeben. XPath stellt sozusagen einen Mechanismus zur Verfügung, um in einem XML Dokument zu navigieren. Die Hauptaufgaben von XPath lassen sich dabei in Adressierung von Daten, Auswertung und Definition von logischen Ausdrücken und Bereitstellung von Hilfsfunktionen gliedern. Dabei befasst sich die Adressierung von Daten mit dem gezielten Zugriff auf Elemente im XML Quelldokument. Bei der Definition von logischen Ausdrücken stellt XPath Mittel zur Verfügung, logische Ausdrücke auszuwerten und zu definieren. Der Bereich der Hilfsfunktionen beinhaltet Funktionen, die das Navigieren und das Abarbeiten bzw. bearbeiten erleichtern sollen.

XPath Hilfsfunktionen:

boolean() :
Überprüfung, ob ein Ausdruck Wahr oder unwahr ist.

true() bzw. false():
Liefert den logischen Ausdruck für „wahr“ zurück. Wird z.B. benutzt, um Variablen einen Wet zuzuweisen. Äquivalent zur Funktion true() funktioniert die Funktion false().

string() bzw. number()
Argument in eine Zeichenkette / Nummer umwandeln.

concat()
Zeichenketten zusammenfügen.

contains()
Überprüfung, ob eine bestimmte Teilzeichenkette in einer Zeichenkette enthalten ist.

translate()
Auffinden von Teilzeichenketten und Ersetzen dieser durch neue Zeichenketten.

current()
Liefert den aktuellen Knoten zurück.

name()
Ermittelt den Namen eines Knotens.

position()
Gibt die aktuelle Position im Baum als Nummer aus.

last()
Ermittelt die Positionsnummer des letzten Knotens auf einer Ebene.

XSL (Extensible Stylesheet Language)

„Genau wie XML ist XSL (Extensible Stylesheet Language) ein Standard, der vom W3C Konsortium entwickelt und veröffentlicht wurde. XSL basiert syntaktisch gesehen auf XML und ermöglicht die Generierung von beliebigen Dokumenten aus einer Basis von strukturierten XML Daten. Dabei werden die eigentlichen Formatierungsanweisungen als XSL und der Mechanismus der Überführung in ein neues Format als XSLT (Extensible Stylesheet Language Transformation) bezeichnet."

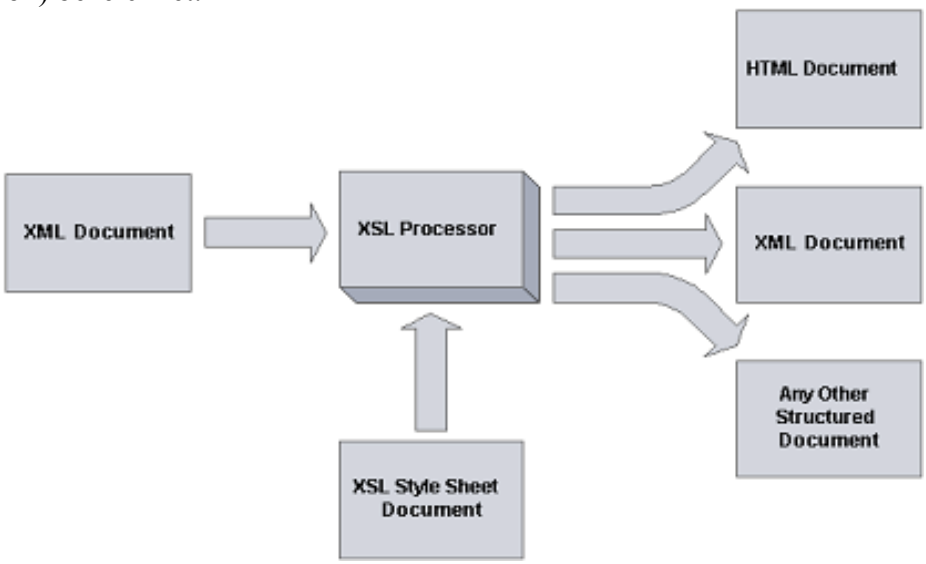

Abbildung 7

Zur Zeit sind XML Browser noch nicht weit verbreitet, deswegen ist es nötig die Daten in ein entsprechendes Betrachtungsformat zu transferieren. Ein mögliches Betrachtungsformat ist beispielsweise HTML, welches sich hervorragend für solche Transformationen eignet. Zur unterstützender Erklärung kann der Ablauf auch in der Abbildung 7 veranschaulicht werden. Es wird eine XML Datei erzeugt und mit einer DTD beschrieben. Die eigentliche Transformation wird mit Hilfe des so genannten Stylesheet durchgeführt. Dieses Stylesheet legt fest, wie die Daten umzuwandeln sind. Die Umwandlung selbst führt ein XSLT Prozessor durch, welcher aus Stylesheet und XML Daten ein entsprechend neues Format (hier HTML) generiert.

Der Ablauf der Transformation ist wie folgt, der XSLT Prozessor arbeitet Knoten für Knoten des Quelldokument ab. Zur Navigation bzw. Adressierung der Daten verwendet XSLT gültige XPath Ausdrücke und Funktionen. Mit Hilfe des Stylesheet wird nun anhand einer Schablone (<xsl:template…>) festgelegt was passieren soll, wenn ein Knoten gefunden wird. Im unteren XSL Stylesheet besagt die Schablone für den Knoten <nachname>, dass der Inhalt des Knotens an der Stelle zwischen den HTML Anweisungen <body> und </body> im Ergebnisdokument eingesetzt wird.

```
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/nachname">
 <html>
      <body>
            <xsl:apply-templates />
      </body>
 </html>
</xsl:template>
```

XSL-FO (Formating Objects)

Wie in Abbildung 7 gezeigt lässt sich eine Transformation mit einem XSLT Prozessor von einen XML Dokument in einfaches Format wie HTML bewerkstelligen, leider sind diese aber nur bedingt druckfähig. Deshalb wird auch der Anspruch gestellt komplexe Formate zu erzeugen.

Hierzu hat das W3C Konsortium mit FO (Formating Objects) eine Erweiterung der XSL Sprache geschaffen, die es ermöglicht komplexe Formatierungen durchzuführen. Mit Formating Objects wird genau beschrieben, wie das Dokument auszusehen hat und wie die Inhalte der jeweiligen XML Knoten formatiert werden sollen.

„Verwendet ein XSL Stylesheet die FO Erweiterung, spricht man auch von einem XSL-FO Stylesheet. Steht ein solches XSL-FO Stylesheet zur Verfügung, ist es nun möglich eine XML Datenbasis mittels einer XSLT Transformation durchzuführen. Das Ergebnis ist dann eine XML Ausgabe, welche die FO Erweiterung nutzt. Die FO Erweiterung beschreibt nun exakt, wie das Ergebnisdokument auszusehen hat. Anhand dieses FO Dokuments lässt sich dann ein beliebiges Format wie z.B. PDF erzeugen.“ (siehe Abbildung 8)

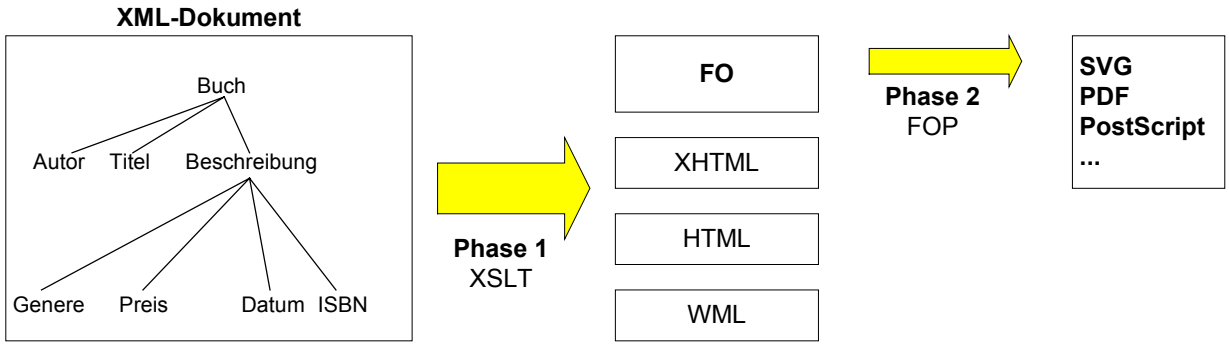

Abbildung 8

XML und Java

SAX-Parser / DOM

Wie im Kapitel "Testen von XML Dokumenten (Parser)" erwähnt ist die wichtigste Komponente der Parser. Im wesentlichen gibt es zwei wichtige Unterscheidungsformen bei Parsern. Zum einen gibt es die Klasse der SAX Parser (Simple API for XML) und zum anderen die Klasse der DOM Parser (Document Object Model). Welchen Parser man nutzt ist einzig und allein von der Anwendung abhängig. Im Prinzip arbeitet SAX ein XML Dokument immer sequentiell ab und kennt nur die Elemente an der aktuellen Stelle im XML Baum. DOM arbeitet hier ganz anders. DOM hält zur Laufzeit den kompletten XML Baum des Dokuments im Speicher. Dies hat vor allem einen großen Vorteil in der Performance der Abarbeitung. Hingegen kann sich dies auch zum Problem entwickeln, wenn sehr große oder sehr viele XML Dokumente gleichzeitig bearbeitet werden.

XALAN- XSLT-Prozessor

Wie im Kapitel "XSL (Extensible Stylesheet Language)" erwähnt, wird ein XSLT Prozessor benötigt, um die Transformation von XML Dokument in ein anderes einfaches Format durchzuführen. An dieser Stelle stellt die Apache Group mit „Xalan" einen XSLT Prozessor als Open Source Lösung zur freien Nutzung zur Verfügung, welcher komplett in Java erhältlich ist.

FOP (Formatting Objects Processing)

Wie im Kapitel "XSL-FO (Formating Objects)" erwähnt, benötigt man für eine Transformation von XML Dokument in ein koplexes Format wie PDF ein spezielles FO Objekt, das erzeugt werden muss. Hierzu hat die Apache Group mit FOP (Formation Objects Processor) ein weiteres Java Framework geschaffen, um den FO Output in beliebige Formate zu transformieren. FOP ist als Java „Jar" Bibliothek erhältlich, welche problemlos in bestehende Java Projekte integriert werden kann. FOP macht es möglich das gewünschte Ausgabeformat zu erzeugen. Als Input benötigt FOP lediglich ein XML Dokument, welches die Formating Objects Erweiterung verwendet. Zu Testzwecken oder in Serverumgebungen ist es auch möglich, FOP über eine Eingabekonsole zu benutzen.

Visualisierung mit SVG

SVG (Scalable Vector Graphics) ist natürlich auch ein Standard des W3C Konsortium. Dieser Standard ist zur semantischen Beschreibung von Vektorgrafiken und soll anderen Vektorgrafikformaten wie z.B. Flash Konkurrenz bieten. Dabei wird die Weiterentwicklung des Standards konsequent von der Firma Adobe vorangetrieben.

Was sind überhaupt die Vorteile einer Vektorgrafik?
Vektorgrafiken sind durch ihre verlustfreie Darstellung ideal für beliebige Skalierungen und Formatierungen geeignet. Sie benötigen einzig für die Darstellung einen Renderer, welcher die semantische Beschreibung der Formen, Geometrien oder Objekte in eine pixelorientierte Darstellung umwandelt. Dieser Renderer muss in Form eines PlugIns einem Browser zur Verfügung gestellt werden, um den SVG-Quellcode interpretieren zu können. Ein solches PlugIn stellt die Firma Adobe kostenlos auf ihrer Webseite zur Verfügung.

Die größte Stärke von SVG liegt vor allem darin,dass das gesamte Format auf XML basiert. SVG selbst stellt ein großes Repertoire zur Verfügung, um komplexe und ansprechende Grafiken zu erzeugen. Dazu zählen unter anderem Farbverläufe, Filter- und Schatteneffekte oder auch Animationen.
Auch die Realisierung von Interaktivität ist mit SVG möglicht. Somit erlaubt SVG z.B. die Verarbeitung von Mausklicks, Mausbewegungen oder Tastatureingaben. Gepaart mit der Möglichkeit Animationen der Objekte zu verwirklichen stellt SVG gute Möglichkeiten zur Verfügung um komplexe Anwendungen zu schaffen.

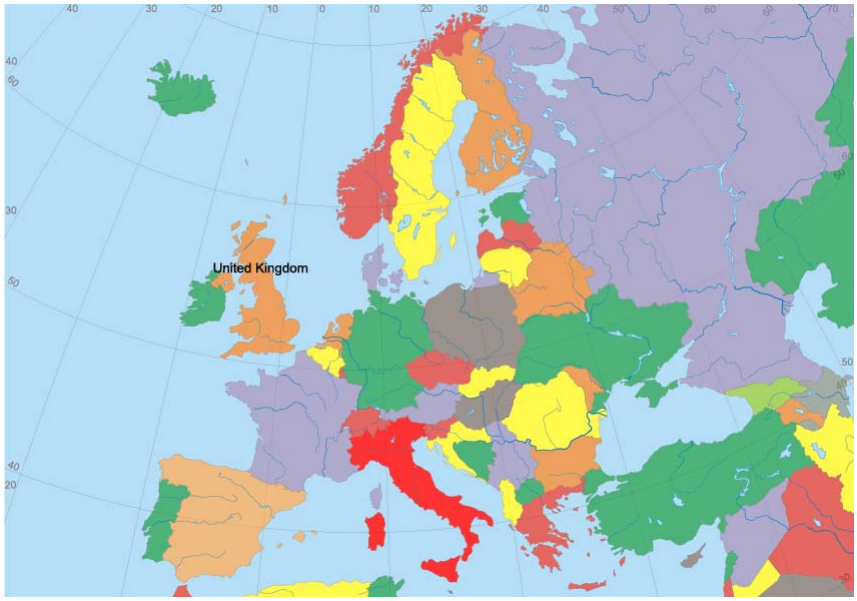

Abbildung 9

Die Einsatzbereiche von SVG reichen von interaktiven Landkarten(siehe Abbildung 9), Fluglinienbeschreibungen, Graphical User Interfaces bis zu 2D-Charts. 2D-Charts, welche Sachverhalte zu Präsentations- oder Reportingzwecken grafisch darstellen sollen, sind eine interessante Anwendungsmöglichkeit von SVG für das Reporting. Gerade um Daten optisch ansprechend aufzubereiten ist ein hervorragendes Einsatzgebiet von SVG.

Batik

Batik ist ein Open Source Projekt. Das Batik Framework wurde von Apache Group geschaffen und bietet eine Möglichkeit in Java das SVG Format zu unterstützen. Batik dient unter anderem zum Erzeugen und zur Manipulation des SVG Codes. Zudem ist es mit Batik möglich im Kontext einer Java-Applikation oder eines Java-Applets SVG darzustellen. Ein schönes Future von Batik ist eine SVG Vektorgrafik in beliebige Rastergrafikformate wie z.B. JPEG oder PNG konvertieren zu können. Batik wird einfach als externe „Jar“ Bibliothek in ein Java Projekt problemlos eingebunden und kann genutzt werden. Dazu sind an der internen Struktur keinerlei Anpassungen notwendig. Es ist eine eigenständige Bibliothek, welche die konkrete Kopplung von SVG und Java schafft.

Fazit

Mit der Fülle an Werkzeugen die XML unterstützen, lasst sich erkennen, das XML sich wirklich schon etabliert hat. Ein großer Vorteil ist, dass alle vorgestellten Werkzeuge auch kostenlos erhältlich sind und keinerlei Lizenzbedingungen unterliegen, was sehr förderlich für das Projekt ist.
Erfreulich ist auch, dass es so viele Werkzeuge gibt die in Java implementiert sind, dies erleichtert das Verständnis und lässt sich einfach in das Projekt einbinden.
Mit der Unterstützung von Batik und SVG, ergeben sich viele schöne Möglichkeiten auch Daten in Visuellobjekte umzuwandeln um besser die trocknen Informationen zu veranschaulichen.

Anwendung

Mein Angebot

Das Kapitel „Mein Angebot“ beinhaltet den Beitrag von Thomas Steinberg an das Projektteam, des nächsten Semesters. Es basiert darauf, eine XSLT-Engine in Java zu implementieren die Transformationen von einem XML-Dokument in ein anderes XML-Dokument oder ein anderes Format tätigt.

Alternative Produkte

Natürlich gibt es schon einige Produkte die Teilbereiche der vorgestellten Aufgabe auch erfüllen könnten (siehe Quelle: [22]), leider sind die meisten davon kostenpflichtig. Ein weiterer Grund für die eigenständige Implementierung, ist die Auseinandersetzung mit den Themen XML/XSLT und Java (bzw. den Java Tools) für Herrn Steinberg.

Implementation

- Die erste Implementation (**Version 0.1**) der XSLT-Engine, soll mit Hilfe einer Zuordnungsmaske (am Beispiel von Outlook-Synchronisation) die ersten Erfahrungen im Projekt erbringen. Es wird eine Oberfläche (siehe Abbildung 10) in Java gebaut, in die vom User-Hand die richtige Zuordnung der einzelnen Elemente zweier XML Dokumente (bzw. einen anderen Destination Dokument) darstellt und die Grundlage für eine richtige Transformation bieten soll. Mit der Zuordnungstabelle wird in der ersten Implementation dem Problem *Synonyme* und/oder *Homonyme* aus den Weg gegangen. Bei betätigen des “Create Buttons“ und der Auswahl des Ziel Dokuments wird eine Transformation ausgeführt und ein neues Dokument in dem entsprechenden gewünschten Format erzeugt.

  Als eine Erweiterung der ersten Implementierung (**Version 0.2**) ist ein “loadIn“ vorgesehen. Was ist ein loadIn? LoadIn soll ein Feature sein, um bei gleichen Zuordnungstabellen dem User den Aufwand abzunehmen und eine entsprechende Zuordnungstabelle(die z.B. in einer Textdatei gespeichert ist) in die XSLT-Engine reinzuladen.

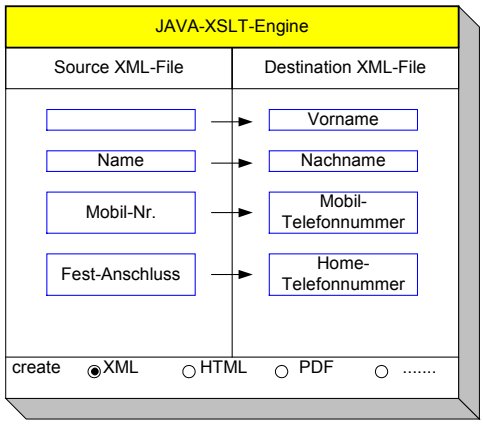

Abbildung 10

- Ein zusätzliches Angebot an das Datawarehouse/Dataminig Team, soll die XSLT-Engine in der **Version 0.3** als Modul eine Visualisierung der im XML-Dokument enthaltenen Daten(Knoten) als Torten/- Kuchendiagramme oder sonstige Diagrammtypen implementiert bekommen. Mit Hilfe von Batik und SVG wird der alternative Weg (siehe Abbildung 11) eingeschlagen, hier sollen wiederum neue Erkenntnisse auf der Visualisierung von Daten / Informationen im Projekt gesammelt werden.

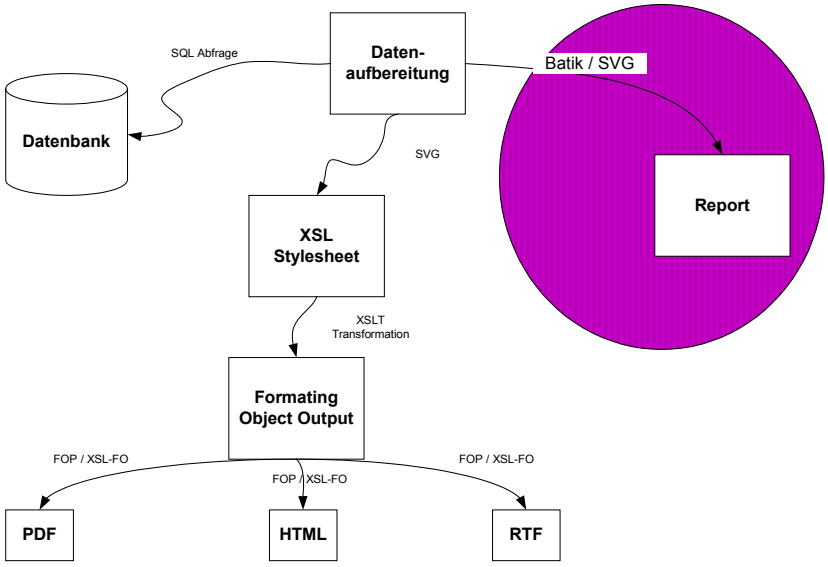

Abbildung 11

Die Implementation (**Version 1.0**) der XSLT-Engine wird in Kooperation mit Artem Khvat stattfinden, sie soll mit Hilfe von Semantik durch Ontologien automatisch eine Transformation tätigen. Dies setzt zu mindestens eine vollständig implementierte Grundstruktur durch die Version 0.1 voraus. Es wird untersucht, in wie weit man Daten semantisch kennzeichnen kann, sodass man ohne fremde/zusätzliche Hilfe Dokumente transformieren kann. Weitere Infos nach einer Menge Recherche und einer Menge an Meetings im nächsten Semester.

Quellen

Bücher:

[1.] Diplomarbeit: Steffen Otto
[2.] Buch: Simon North & Paul Hermans ,XML in 21 Tagen
[3.] Buch:Elliotte Rusty Harold, XML
[4.] Buch: RRZN, Java und XML 1 Auflage
[5.] Buch: Alexander Adam, SVG Das Praxisbuch, Franzis 2002

Skripte/Folien:

[6.] *Ubbo Visser*, Heiner Stuckenschmidt, and Holger Wache (Intelligent Information Integration for the Semantic Web)
[7.] Vorlesung Semantic Web von Prof.Owsnicki-Klewe

URL:

[8.] http://xml.apache.org/ (14.04.2005)
[9.] http://www.w3c.org/ (14.04.2005)
[10.] de.wikipedia.org/wiki/Semantik (14.04.2005)
[11.] www.schriftdeutsch.de/orth-li4.htm (14.04.2005)
[12.] de.wikipedia.org/wiki/Syntax (14.04.2005)
[13.] www.programmierschule-dortmund.de/Woerter.html (14.04.2005)
[14.] de.wikipedia.org/wiki/Transformation_(Sprachwissenschaft) (14.04.2005)
[15.] de.wikipedia.org/wiki/Transformation (14.04.2005)
[16.] http://www.matheboard.de/lexikon/Transformation,definition.htm (04.07.2005)
[17.] http://dbs.uni-leipzig.de/buch/mrdbs-129.html (04.07.2005)
[18.] http://www.w3.org/2002/Talks/SVG-HongKong-IH/45.html (04.07.2005)
[19.] http://devresource.hp.com/drc/resources/xmlfundCklist/index.jsp (04.07.2005)
[20.] http://java.sun.com/xml/jaxp/dist/1.1/docs/tutorial/overview/3_apis.html (04.07.2005)
[21.] http://www.cambridgedocs.com/products_wordml2FO.htm (04.07.2005)

Produkte:

[22.] http://www.antennahouse.com/product/designer/ (04.07.2005)