Thema:

Browsergestützte Visualisierung von XML Topic Maps

Diplomarbeit

Arbeitsgruppe Wirtschaftsinformatik

Themensteller: Prof. Dr. rer. pol. habil. Hans-Knud Arndt
Betreuer: Dipl.-Kfm. Henner Graubitz

vorgelegt von: Jens Rummler

Abgabetermin: 20. Oktober 2008

Inhaltsverzeichnis

Verzeichnis der Abkürzungen und Akronyme

AJAX	Asynchronous JavaScript And XML
API	Application Programming Interface
ASP	Active Server Pages
BOS	Bounded Object Set
bspw.	beispielsweise
bzw.	beziehungsweise
CDATA	Character Data
CGI	Common Gateway Interface
CSS	Cascading Style Sheets
DB	Datenbank
DOM	Document Object Model
DTD	Document Type Definition
etc.	et cetera
FTP	File Transport Protocol
GFX	Graphics
GML	Generalized Markup Language
HTML	Hypertext Markup Language
HTTP	Hypertext Transport Protocol
HyTime	Hypermedia/Time-based Structuring Language
ID	Identifier (eindeutiger Bezeichner)
IP	Internet Protocol
ISO	International Organisation for Standardization
JDBC	Java Database Connectivity
JSON	JavaScript Object Notation
JSP	Java Server Pages
MIME	Multipurpose Internet Mail Extension
OO	Objektorientierung
OWL	Web Ontology Language
PHP	PHP: Hypertext Preprocessor
PNG	Portable Network Graphics
RDF	Resource Description Framework
SAX	Simple API for XML
SGML	Standardized Generalized Markup Language
sog.	sogenannt
SQL	Structured Query Language
SVG	Scalable Vector Graphics
TCP	Transmission Control Protocol
UML	Unified Modeling Language
URL	Unified Resource Locator
vgl.	vergleiche
W3C	World Wide Web Consortium
WWW	World Wide Web
XHTML	eXtensible Hypertext Markup Language
XML	eXtensible Markup Language
XTM	eXtensible Topic Map

Symbolverzeichnis

#	Anzahl (der Elemente)
n	Natürliche Zahl > 0
X	Reellwertige Zahl ≥ 0, abgetragen auf der Abszisse eines kartesischen Koordinatensystems
Y	Reellwertige Zahl ≥ 0, abgetragen auf der Ordinate eines kartesischen Koordinatensystems

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einführung und Aufbau der Arbeit

1.1 Einführung

Die Suche nach Informationen im Internet oder in organisationsinternen Intranets ist ein wesentlicher Teil des Arbeitsprozesses in der heutigen Gesellschaft. Nahezu jede Information ist im World Wide Web (WWW) verfügbar, jedoch stellt es sich als ein kompliziertes Unterfangen dar, aus der Menge der Suchresultate diejenigen herauszufiltern, die die gesuchten Informationen enthalten. Eine Ursache für den Produktivitätsverlust, der durch das mühsame Beschaffen von Informationen entsteht, ist die Mehrdeutigkeit der natürlichen Sprache.

Einen möglichen Ausweg bieten Topic Maps. Mit ihnen ist es möglich, unstrukturierten Datenquellen eine Struktur zu verleihen. Dies geschieht durch das Ergänzen von Metadaten, jedoch ohne die Dokumente an sich zu verändern. Vielmehr wird eine Metadatenschicht über die Dokumente gelegt, die diese um Kontextinformationen, hierarchische und andere Beziehungen ergänzt. Erst mit diesen Metadaten wird eine semantische Suche in diesen Datenquellen möglich.

In den heutigen Volltextsuchmaschinen wie bspw. Google[1] existiert keine solche Metadatenschicht, aus diesem Grund wird eine Suche nach dem Begriff „Golf“ eine Vielzahl von verschiedenen Ergebnissen hervorbringen. Diese können jedem Kontext, den der Begriff in der natürlichen Sprache annehmen kann, wie bspw. einer Sportart, einem Automodell, einer Meeresbucht oder einer Sprachfamilie, entsprechen[2]. Der Suchende hat nun die Aufgabe, die wesentlichen Dokumente von den für seine Suche unwesentlichen zu trennen. Mit einer Metadatenschicht könnte er jedoch einen Kontext für seine Suche angeben, was dazu führt, dass, wenn gewünscht, nur die Begriffe im Bereich „Sport“ angezeigt werden.

Um diese Funktionalität für die Nutzer des WWW oder von Intranets verfügbar zu machen, wird als erster Schritt eine Darstellung von Topic Maps, welche die ergänzenden Daten zur Suche liefern, benötigt. Diese Visualisierung sollte möglichst im Werkzeug der Informationsbeschaffung, dem Web-Browser, erfolgen, so dass sich der Umgang mit diesen Informationen ohne einen Medienbruch natürlich für den Benutzer ergibt.

---

[1] http://www.google.de, aufgerufen am 27. Juli 2008.
[2] Vgl. Abbildung 1-1: Google-Suche nach "Golf".

Abbildung 1-1: Google-Suche nach "Golf"

Das Ziel dieser Arbeit ist die Entwicklung eines Anwendungsprogramms, das Topic Maps, welche im XTM 1.0 Format kodiert sind, in gängigen Web Browsern dynamisch darstellen kann, ohne weitere Hilfsmittel, wie bspw. Java Applets oder Adobe Flash[3], zu verwenden. Weiterhin soll eine dynamische Navigation innerhalb der Topic Map und zu den in ihr beschriebenen Dokumenten ermöglicht werden.

Zur Verdeutlichung des Potentials einer solchen Darstellung wird weiterhin die in (Twele (2008)) beschriebene Java Bibliothek[4] zur Transformation von *Wikipedia*[5]-Seiten in Topic Maps in die Anwendung integriert.

1.2 Aufgabenstellung

In dieser Arbeit wird die Entwicklung einer Web-Anwendung zur Darstellung von eXtensible Topic Maps thematisiert.

Laut Aufgabenstellung soll ein Anwendungsprogramm, welches im Web-Browser ohne jegliche weitere clientseitige Installation ablauffähig ist, entwickelt werden, das Topic Maps, kodiert mittels der XTM 1.0 Spezifikation, visualisieren kann. Dabei soll das Java Archiv *XTMGenerator* eingebunden und als Datenquelle genutzt werden.

[3] Vgl. http://www.adobe.com/de/products/flashplayer/, aufgerufen am 27. Juli 2008.
[4] Vgl. Twele (2008).
[5] Vgl. http://www.wikipedia.de, aufgerufen am 27. Juli 2008.

Weiterhin soll die Möglichkeit existieren, XTM-Dokumente, die lokal gespeichert dem Nutzer vorliegen, zu visualisieren.

Die sich daraus ergebenen Beschränkungen und Freiheitsgrade sind in folgendem morphologischen Kasten (vgl. Tabelle 1) zusammengefasst. Alternativen, die weder durch die Aufgabenstellung fixiert noch ausgeschlossen wurden, konnten bei der Entwicklung frei gewählt werden.

Tabelle 1: Beschränkungen und Freiheitsgrade der Aufgabenstellung

Kriterium	***Legende:*** **a ~ fixiert durch Aufgabenstellung b ~ ausgeschlossen durch Aufgabenstellung ✓ ~ umgesetzt in der Arbeit**			
Art der Anwendung	Desktop-Anwendung (b)		Web-Anwendung (a, ✓)	
Art der Client-Applikation (Web)	AJAX (a, ✓)	Adobe Flash (b)	Java Applet (b)	Reines HTML
Art der Server-Applikation (Web)	CGI (b)	PHP (b)	Java Servlets / JSP (a, ✓)	ASP (b)
Form der Visualisierung	Serverseitige Erzeugung von Graphikdatei	DOM-Manipulation (✓)	Graphikframework (b)	
Navigation in Topic Maps	Drag & Drop (✓)	Suchbegriff eingeben (✓)	Sprachsteuerung	
Übersichtlichkeit	Darstellung der gesamten Topic Map	Darstellung einer relevanten Teilmenge bzgl. eines Suchbegriffs (✓)	Ausblenden von Assoziationen (✓)	

1.3 Aufbau der Arbeit

Die Arbeit ist in vier Kapitel aufgeteilt. Das erste, einleitende Kapitel liefert einen kurzen Einstieg in die Thematik und definiert Aufgabenstellung, Ziel und Aufbau der Diplomarbeit.

In Kapitel 2 werden die grundlegenden Technologien zur Entwicklung einer Anwendung, die im Web-Browser ohne zusätzliche Anwendungsprogramme ablaufen soll, erläutert. Dabei steht besonders die Einhaltung von IT-Standards im Vordergrund, um eine möglichst große Anzahl an Client-Computersystemen zu unterstützen. Weiterhin werden semantische Netze thematisiert, die eine Verallgemeinerung der Topic Maps darstellen, um dann den ISO-Standard für Topic Maps zu beleuchten. Letztlich wird die XTM 1.0 Spezifikation, in welcher eXtensible Topic Maps kodiert werden, beschrieben.

Kapitel 3 erläutert die Entwicklung der Web-Anwendung anhand eines Software-Entwicklungsprozesses. Dabei steht die Entwicklung der Software für die Clientseite im Vordergrund, da auf der Serverseite lediglich Transformationen durchgeführt werden; von *Wikipedia*-Artikeln in XTM-Dateien und von großen XTM-Dokumenten in kleine den aktuellen Suchbegriff beschreibende Ausschnitte aus diesen großen XTM-Dateien. Die Clientseite nutzt diese XTM-Dokumente zur Visualisierung. Diese wird mittels der DOM-Manipulation der Website realisiert, welche die Anwendung repräsentiert. Dabei stehen die JavaScript-Objekte zum Parsen und Speichern der Inhalte des Dokumentes und zur Durchführung der Visualisierung in besonderem Fokus. Letztlich wird die Benutzung der Anwendung anhand eines Beispiels erläutert.

In Kapitel 4 wird ein Fazit gezogen und Verbesserungsmöglichkeiten der entwickelten Anwendung aufgezeigt. Weiterhin werden Vorschläge zur Weiterentwicklung und zum Einsatz der Software gegeben.

2 Grundlagen der Arbeit

Im folgenden Kapitel werden die grundlegenden Technologien und Architekturen erläutert, auf denen diese Arbeit beruht und mit denen die zu entwickelnde Software umgesetzt wird. Eine Übersicht der beschriebenen Grundlagen bietet die folgende Abbildung:

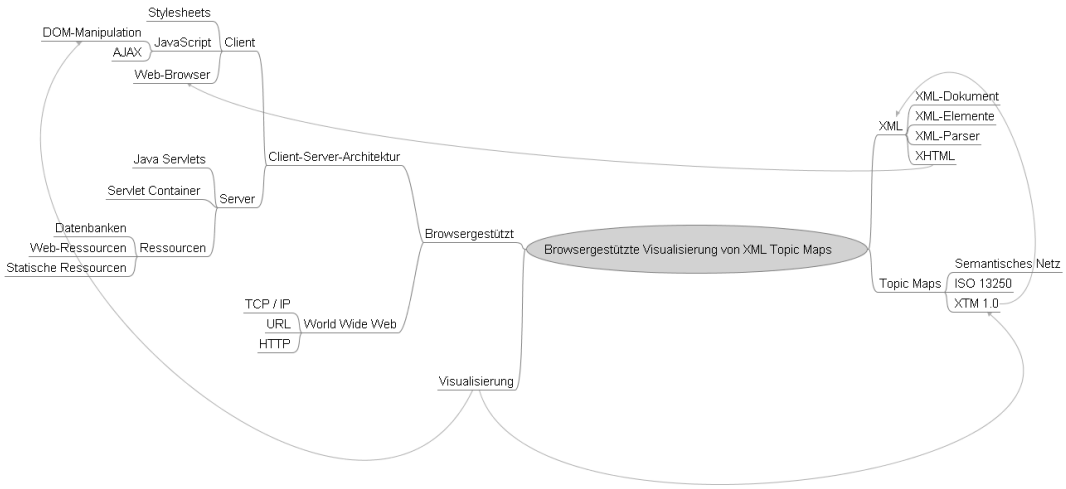


Abbildung 2-1: Mind Map der Grundlagen

In der Mind Map werden die einzelnen Stichwörter aus dem Titel dieser Arbeit mit den zugehörigen Technologien, Standards und Konzepten versehen. Diese sind teilweise miteinander verknüpft. So können Dokumente in XHTML, einer XML-Spezialisierung, formuliert und im Web-Browser angezeigt werden. Technologien, die durch die Aufgabenstellung ausgeschlossen wurden, werden nicht thematisiert (vgl. Tabelle 1).

Da durch die Verknüpfungen keine feste Reihenfolge zur Erläuterung abgeleitet werden kann, sollen zunächst die grundlegenden Techniken erläutert werden, um darauf aufbauend zu den komplexeren Technologien überzugehen.

Die Erläuterungen sollen einen Einstieg bieten und sind keinesfalls als vollständige Beschreibung zu betrachten.

2.1 Client-Server-Architektur

Als Client-Server-Architektur „bezeichnet man ein Systemdesign, bei dem die Verarbeitung einer Anwendung in zwei separate Teile“[6] aufgespalten wird. Die Anwendung wird also teilweise clientseitig und teilweise serverseitig verarbeitet.[7]

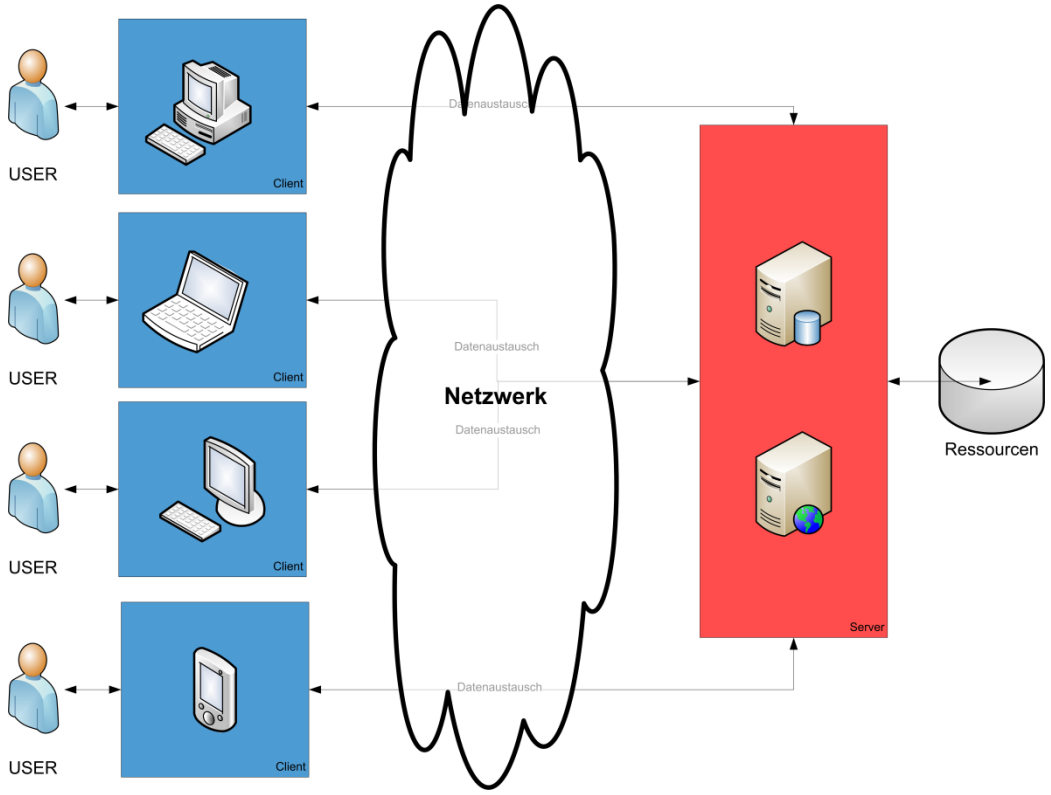


Abbildung 2-2: Client-Server-Architektur

Wie in Abbildung 2-2 dargestellt, interagiert der Benutzer (User) mit der Anwendung, indem er Daten in einen Client eingibt. Der Client ist über ein Netzwerk mit dem Server verbunden und gibt per Datenaustausch mit dem Server die Verarbeitung der gesendeten Daten in Auftrag. Der Server bearbeitet die Daten und nutzt dazu von Ressourcen bereitgestellte Informationen. Mithilfe dieser Daten generiert der Server eine Antwort. Diese wird an den Client, der den Auftrag zur Bearbeitung gab, per Datenaustausch zurückgesendet. Die Antwort des Servers wird auf dem Client verarbeitet und / oder dem User präsentiert.[8]

---

[6] Vgl. http://www.it-administrator.de/lexikon/client-server-architektur.html, aufgerufen am 27. Juli 2008.
[7] Vgl. Abbildung 2-2: Client-Server-Architektur.
[8] Vgl. Rotter et al. (2008).

Es ist anzustreben, möglichst mannigfaltige Ausprägungen von Client-Systemarchitekturen mit einer Anwendung zu unterstützen. Dies wird durch die Nutzung von IT-Standards, wie in Abschnitt 2.4 beschrieben, erreicht.

Die Daten werden per Datenaustausch an den Server übermittelt. Der Datenaustausch erfolgt in einer vom Entwickler spezifizierten Form, es können also eigene Datenformate entworfen werden. Es ist jedoch ratsam, auch hier auf bewährte Standards zurückzugreifen. Diese werden in Kapitel 2.2.4 erläutert.

Um die Informationen zwischen Client und Server auszutauschen wird ein Netzwerk benötigt. Dabei kann es sich um ein rechnerinternes Loopback-Netzwerk, ein firmeninternes Netzwerk oder um das Internet handeln. Letzteres wird stellvertretend für alle anderen Formen von Netzwerken in Abschnitt 2.2 erläutert, da die grundlegenden Technologien bei allen diesen Netzwerken gleich sind und das Internet bzw. seine graphische Oberfläche, das WWW, somit stellvertretend für die zahlreichen Intranets[9] beschrieben werden kann. Generell sind Intranets nur einem bestimmten Nutzerkreis zugänglich, um den Zugriff auf geschützte Informationen zu kontrollieren.

Damit der Server die übermittelten Daten entgegennehmen und verarbeiten kann, ist der Einsatz von Technologien wie bspw. Web-Servern und Java Servlets nötig. Diese werden in Abschnitt 2.5 erläutert. Mithilfe von Ressourcen, wie zum Beispiel Datenbanken oder Informationsquellen im WWW, wird eine Antwort auf die Anfrage des Clients erzeugt, die, umgewandelt in das Datenaustauschformat, an den Client zurückgesendet wird. Mögliche Ressourcen werden in Kapitel 2.5.3 genannt.

Die Client-Server-Architektur schreibt nicht generell vor, an welcher Stelle die Trennung der Verarbeitung zwischen Client und Server erfolgen soll. In (Niemann (1995)) werden die folgenden Schnittpunkte vorgeschlagen:

- Präsentation
- Steuerung
- Anwendungslogik
- Datenverwaltung
- Datenhaltung

[9] Vgl. Steffen et al. (2008).

So kann bspw. die Präsentation, Steuerung und Anwendungslogik clientseitig, die Datenverwaltung und Datenhaltung aber serverseitig erfolgen.

2.2 World Wide Web

Im Jahr „1990 verknüpfte Tim Berners-Lee zwei Technologien zu einer neuen Anwendung“[10], indem er die Technologien Internet und Hypertext kombinierte. Es entstand die Internet-Anwendung WWW. Die Begriffe WWW und Internet stellen im gängigen Sprachgebrauch Synonyme dar, sind es aber per Definition nicht.

Das Internet, welches bereits in den frühen 1960er Jahren entwickelt wurde, stellt eine „Infrastruktur für Netzwerk-Anwendungen“[10] bereit, die „beliebig implementierte lokale Netze auf beliebigen Rechner-Plattformen verknüpfen sollte.“[10] Die Kommunikation erfolgt dabei über das, dem Internet zugrundeliegende, TCP/IP-Protokoll[11]. Nachdem in Wissenschaftskreisen die Internet-Anwendungen E-Mail, USENET und FTP etabliert waren, stellte das WWW die Weichen zur Nutzung des Internet durch die breite Öffentlichkeit.[12]

Als Hypertext werden spezielle Textdokumente bezeichnet, „in denen besonders gekennzeichnete Textstellen nach Aktivierung (meist mit der Maus) Aktionen auslösen.“[13] Hypertext wird mittels eines Web-Browsers dargestellt[14], nutzt die Technologien URL[15] und HTTP[16] und wird mittels HTML[17] implementiert.

2.2.1 TCP/IP

„Die Kommunikation zwischen Client und Server erfolgt unter Nutzung der Internet-Infrastruktur. Die TCP/IP-Protokollfamilie stellt diese Infrastruktur bereit. Sie besteht im Wesentlichen aus den Protokollen TCP und IP.“[18]

Das IP-Protokoll ermöglicht es, durch die Zuweisung einer eindeutigen IP-Adresse, Daten an den richtigen Rechner im richtigen Netzwerk zu übermitteln. Dadurch werden die Protokolle, die das IP-Protokoll zur Kommunikation nutzen, davon entbunden, Informationen über den physikalischen Übertragungsweg zu kennen.

---

[10] Wöhr (2004), S. 3.
[11] Vgl. Kapitel 2.2.1.
[12] Vgl. Wöhr (2004), S. 3ff.
[13] Universität Zürich (2008).
[14] Vgl. Kapitel 2.4.1.
[15] Vgl. Kapitel 2.2.2.
[16] Vgl. Kapitel 2.2.3.
[17] Vgl. Kapitel 2.3.7.
[18] Wöhr (2004), S. 5.

Das IP-Protokoll kann jedoch nicht gewährleisten, dass die in Paketen versendeten Informationen vollständig und korrekt übertragen wurden. Diese Funktionalität ergänzt das TCP-Protokoll, welches für das richtige Zusammensetzen der Pakete beim Empfänger sorgt und Fehlerkorrekturen vornimmt. Weiterhin wird das Port-Konzept eingeführt, das definiert, welche Anwendung beim Empfänger für die Verarbeitung der Daten zu nutzen ist.[19]

2.2.2 URL

URL steht für Uniform Resource Locator. Mit URL wird die „Adressierung jeglicher Art von Internet-Ressource“[20] möglich. Die allgemeine Syntax für einen URL ist:

```
protokoll://host[:port]/pfad/zu/ressource[#sektion|?parameter]
```

Durch das *Protokoll* wird der auf dem *Host* angesprochene Dienst beschrieben. Dies kann u.a. `http`, `ftp` oder `ssh` sein. Durch die Angabe des Protokolls wird der Anfrage ein Standard*port* zugewiesen, der allerdings durch die explizite Angabe eines *Ports* übergangen werden kann.

Durch den *Host* wird der Zielrechner angegeben. Dies kann ein Name oder eine IP-Adresse sein.

Der *Pfad zur Ressource* wird, angelehnt an die Unix Verzeichnisstruktur, mit Schrägstrichen spezifiziert. Weiterhin kann eine spezielle *Sektion* im Dokument abgerufen oder Daten übermittelt werden, welche durch *Parameter* angegeben sind.[21]

2.2.3 HTTP

„Oberhalb von TCP und IP ist im Protokoll-Stack die Applikationsschicht angesiedelt. [...] Die Applikationsschicht ist für die konkrete Syntax und Semantik der zwischen Dienstprogrammen auszutauschenden anwendungsspezifischen Daten verantwortlich.“[22]

Das Hypertext Transfer Protocol (HTTP) ist das Protokoll, auf dem das WWW basiert. Es stellt die gemeinsame Sprache des Web-Browsers und des Web-Servers dar und basiert auf dem Request/Response-Paradigma. Eine HTTP-Transaktion erfolgt in 4 Schritten:

---

[19] Vgl. Wöhr (2004), S. 5f.
[20] Wöhr (2004), S. 13.
[21] Vgl. Wöhr (2004), 13f.
[22] Wöhr (2004), S. 6.

1. Aufbau einer TCP-Verbindung vom Client zum Server
2. Senden eines HTTP-Request an den Server
3. Senden einer HTTP-Response an den Client
4. Abbau der Verbindung durch den Server

Der HTTP-Request wird dabei durch einen URL erzeugt. Der Server sendet daraufhin per MIME[23] (Multipurpose Internet Mail Extension) typisierte Daten an den Client zurück. Dies können XML- oder HTML-Dokumente, aber auch Bilder, Anwendungsdaten, etc. sein.

2.2.4 Datenaustausch

Nutzen Client und Server ein gemeinsames Protokoll, also sprechen sie dieselbe Sprache, können Daten ausgetauscht werden. Dies reicht von einer einfachen Anforderung und Sendung von Daten, die vom Server nicht manipuliert werden, wie bspw. statische HTML-Seiten oder Dateien per FTP, bis zu hochspeziellen Anforderungen wie der Erzeugung dynamischer Inhalte per serverseitigen Skripts oder AJAX-Anfragen[24].

Zur Übertragung von strukturierten Daten, welche auf der Seite des Clients weiterverarbeitet werden, existieren verschiedene Datenaustauschformate, bspw. JSON[25] und XML; ebenfalls kann ein Entwickler eigene Datenformate erdenken. Es existieren allerdings einige Fakten, die für die Nutzung von dem sich zum Datenaustausch-Standardformat entwickelnden XML sprechen[26]:

- XML ist plattformunabhängig.
- XML ist durch den Unicode[27] Zeichensatz international verwendbar.
- XML-Dokumente liegen in Baumstruktur vor, deshalb sind sie leicht durchsuchbar.
- Viele Werkzeuge zur Verarbeitung sind verfügbar.
- Daten werden strukturiert übertragen.

[23] Wöhr (2004), S. 8ff.
[24] Vgl. Kapitel 2.4.3.2.
[25] http://json.org/, aufgerufen am 10. August 2008.
[26] Vgl. Kobligk (2008).
[27] Vgl. Widhalm/Mück (2002), S. 29

- Eine Metadaten-Übertragung erfolgt durch XML-Elemente.

2.3 XML

„XML steht für eXtensible Markup Language. Laut World Wide Web Consortium (W3C) ist XML ein *universelles Format für strukturierte Dokumente und Daten.*“[28]

Bei XML handelt es sich um eine Markup-Sprache. Charles Goldfarb entwickelte 1969 im Auftrag von IBM die erste Markup-Sprache, die Generalized Markup Language (GML). Sie besaß schon die wesentlichen Vorteile, die Markup-Sprachen für den Datenaustausch interessant machen:

- Grundsätzlich plattformneutral (da Textdokumente),
- Lesbar durch Mensch und Computer,
- Veränderbar durch variable Länge.

Im Jahr 1986 wurde SGML, die Standardized Generalized Markup Language, standardisiert. Sie war bereits wesentlich umfangreicher und komplexer als der Vorgänger GML.[29]

Mit der Einführung von HTML wurden die Markup-Sprachen bekannt, jedoch erzeugte das starre Dokumentenmodell von HTML eine Vielzahl von proprietären Erweiterungen und Quasistandards, die sogenannten ‚Browserkriege‘[30] begannen. An dieser Stelle soll nicht weiter darauf eingegangen werden.

„XML vereinigt Eigenschaften von SGML als auch von HTML. Es bietet eine Form der Einfachheit, die an HTML erinnern lässt, jedoch auch eine Komplexität, die nahezu an die von SGML heranreicht.“[31]

[28] Wöhr (2004), S. 50.
[29] Vgl. Widhalm/Mück (2002), S. 21f.
[30] Vgl. Wöhr (2004), S. 43.
[31] Widhalm/Mück (2002), S. 23.

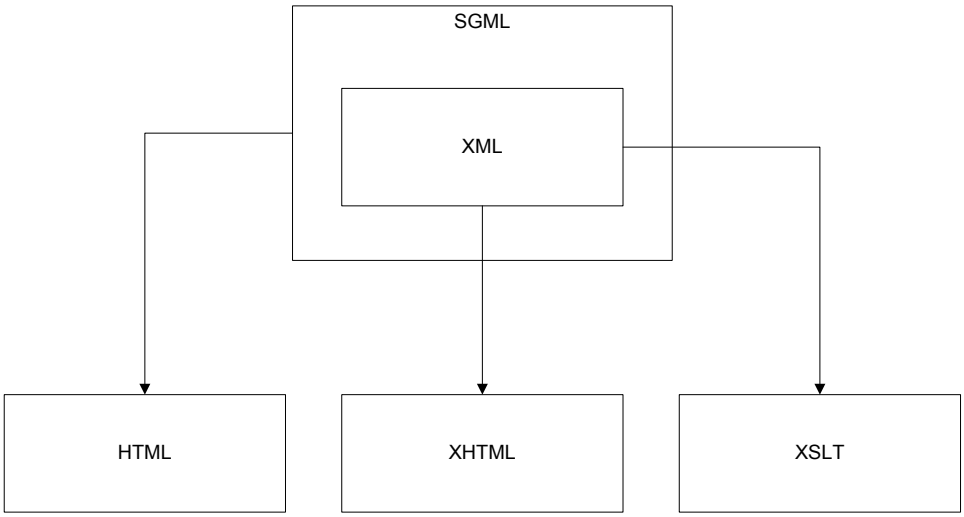


Abbildung 2-3: Markup-Sprachen (Wöhr (2004), S. 52)

XSLT erlaubt die Transformation von XML-Dokumenten. Für weitere Informationen siehe (Wöhr (2004), S. 107ff).

2.3.1 XML-Dokument

„Ein XML-Dokument gliedert sich im Wesentlichen in einen *Prolog* und in eine *Dokumentinstanz*.“[32]

Der *Prolog*, mit dem XML-Dokumente beginnen, gibt die XML-Version und optional den Dokumenttyp und weitere Eigenschaften des Dokumentes an. Ein Beispiel für einen Prolog wäre:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
```

Elemente, die mit `<?` beginnen und mit `?>` enden, nennt man XML-Verarbeitungsanweisungen. In dem angegebenen Beispielprolog wird die XML-Version, die Kodierung und das Attribut `standalone`, welches angibt, ob ein Dokument von anderen Dokumenten abhängig ist oder für sich selbst steht, spezifiziert. Für weitere Informationen siehe (Widhalm/Mück (2002), S. 29ff).

[32] Widhalm/Mück (2002), S. 29.

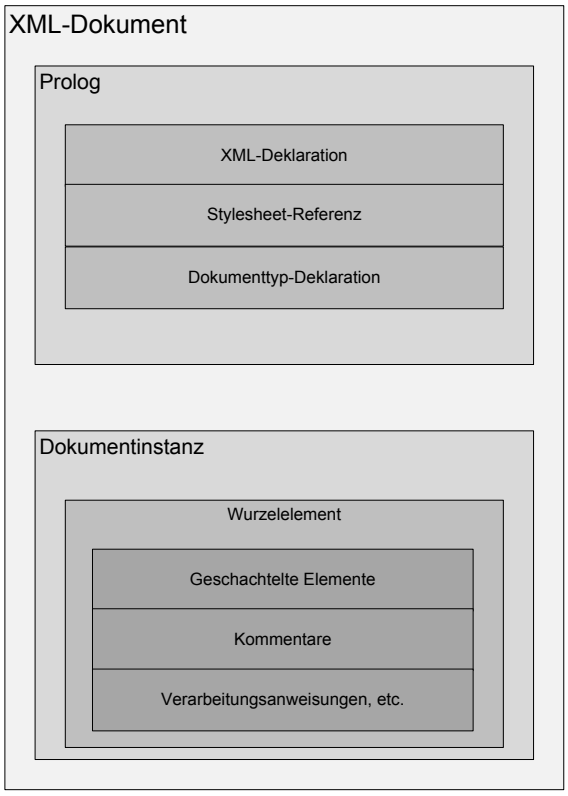


Abbildung 2-4: Aufbau eines XML-Dokumentes (vgl. Wöhr (2004), S. 53)

Weiterhin ist es vorteilhaft, wenn auch nicht zwingend erforderlich, eine Dokumenttyp-Deklaration vorzunehmen. Dies kann entweder eine Referenz auf eine externe Dokumenttyp-Definition (DTD) oder eine im Dokument eingebettete Definition sein. Beispiele für externe Referenzen sind die XHTML DTDs[33] oder die XTM 1.0 DTD[34].

Wird keine DTD angegeben, so ergeben sich folgende Nachteile[35]:

- Elemente, die nur weitere Elemente als Inhalt haben, dürfen keinen Leerraum zwischen dem Start-Tag des umhüllenden Elements und dem Start-Tag des ersten geschachtelten Elements enthalten. XML-Parser wissen ohne DTD nicht, ob die Information interpretiert werden soll oder nicht. Der Leerraum wäre als Text interpretierbar.

- Es müssen alle Attributwerte im XML-Dokument angegeben werden, Vorgabewerte sind nicht möglich.

[33] Vgl. W3C (2008a).
[34] Vgl. Pepper/Moore (2008).
[35] Vgl. North/Hermans (2000).

- Es wird keine Restriktion des Autors durch die vorgegebene Struktur des XML-Dokumentes gegeben. Solche Restriktionen sind zur Fehlervermeidung sinnvoll, da die Validierung gegen die DTD strukturelle Fehler offenbart.

Der eigentliche Inhalt des XML-Dokumentes findet sich in der *Dokumentinstanz*. Sie enthält, soweit angegeben, die Elemente, Attribute und Inhalte, welche in der DTD im *Prolog* als für das Dokument gültig angegeben wurden. Auf die einzelnen Sprachelemente gehen die folgenden Abschnitte ein.

XML-Beispieldatei:

```
<?xml version="1.0">
<!DOCTYPE HTML PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
          "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="de">
        <head>
            <title>Beispielseite</title>
        </head>
        <body>
            <p>Hallo!</p>
        </body>
</html>
```

2.3.2 Elemente

Eine XML-Datei kann verschiedene Elemente enthalten. Es ist darauf zu achten, dass diese in der DTD definiert wurden. Die Elemente bilden eine Baumstruktur, welche nur ein Wurzelelement besitzt, im Beispiel wäre dies `<html>`. Ein Element kann weitere Elemente, Attribute und Zeichenfolgen beinhalten. Es wird durch `</html>` geschlossen. Ein Start-Tag beginnt mit dem Zeichen <, ein End-Tag mit `</`. Start- und End-Tags werden durch das Zeichen > geschlossen. Das Element `<html>` darf laut DTD nur die Elemente `<head>` und `<body>` umschließen, und kann verschiedene Attribute tragen, hier `xmlns` und `xml:lang`. Die Reihenfolge der geschachtelten Elemente ist nicht beliebig; der Autor des Dokumentes muss sich an die in der DTD vorgegebene Abfolge halten.

Diese Struktur wird durch XML-Parser[36], die nach dem Document Object Model (DOM) vorgehen, als Baum im Speicher des Rechners abgelegt und ermöglichen dem Programmierer einen einfachen Zugriff auf die Attribute und Inhalte.

2.3.3 Attribute

„Elemente können auch über mehrere Attribute verfügen, diese werden dann hintereinander, durch Leerraum getrennt, innerhalb des Start-Tags angeführt“[37].

Die Attributnamen, ebenso wie die Elementnamen, dürfen nur mit einem Buchstaben oder dem Sonderzeichen ‚_‘ beginnen. Darauf folgend bestehen sie aus einer Reihe von beliebigen Zeichen, die entweder aus Buchstaben, Ziffern oder einem der Sonderzeichen aus der Menge (_ ; - ; .) gebildet werden. Es wird zwischen Groß- und Kleinschreibung unterschieden.

Weiterhin bestehen Attribute immer aus einem Paar, das nach dem Schema

```
Attributname = "Attributwert"
```

gebildet wird. Der Attributwert wird immer als Zeichenkette interpretiert, das heißt selbst bei numerischen Werten muss ein umschließendes einfaches oder doppeltes Anführungszeichen vorhanden sein. Jedes Attribut darf nur einmal pro XML-Element vorkommen, die Reihenfolge spielt dabei keine Rolle.[38]

Im Beispiel besitzt das Element `<html>` zwei Attribute. Das erste Attribut, dessen Attributname `xmlns` und dessen Attributwert `http://www.w3c.org/1999/xhtml` ist, gibt den Namensraum des Elements und der darunter geschachtelten Elemente an. Das zweite Element definiert die Sprache, sein Attributname ist `xml:lang`, der Attributwert gibt den Ländercode für Deutschland an.

Einer besonderen Erwähnung bedürfen die `XLink-` und `XPointer`-Attribute. Sie setzen die Referenzmechanismen von Hypertext-Links für XML-Dokumente in erweiterter Weise um. Für weitere Informationen siehe (Widhalm/Mück (2002), 55ff).

---

[36] Vgl. Abschnitt 2.3.6.
[37] Widhalm/Mück (2002), S. 26.
[38] Vgl. Dipper (2008).

2.3.4 Zeichenfolgen und CDATA-Bereiche

Elemente können, neben den umschlossenen Elementen und wenn es laut DTD vorgesehen ist, textuellen Inhalt besitzen. Im Beispiel wäre dies der Inhalt des Elements `<title>`, dieser hat den Text ‚Beispielseite' zum Inhalt.

Allerdings dürfen dabei „bestimmte Zeichen nicht verwendet werden, da sie durch den XML-Parser in besonderer Weise interpretiert werden."[39] Sollen bspw. spitze Klammern in einem Textblock verwendet werden, so müssen diese, damit sie durch den Parser als Text und nicht als Beginn eines neuen Elements betrachtet werden, entweder maskiert oder innerhalb eines sog. CDATA-Bereiches genutzt werden.

Um bestimmte Zeichen zu maskieren, definiert XML sog. *Entity Referenzen*. „Entity Referenzen sind Ersatzdarstellungen, vergleichbar mit symbolischen Konstanten in Programmiersprachen."[39] Für weitere Informationen siehe (W3C (2008b)).

„Durch die Definition eines CDATA-Bereiches (character data) wird dem Prozessor mitgeteilt, dass es sich um reinen Text handelt. Dieser wird vom Prozessor nicht interpretiert."[40] So könnte das Beispiel erweitert werden:

```
[...]
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="de">

  <head>
            <title>
                <![CDATA[
                      <html>
                            <head>
                                  <title>Beispielseite</title>
                            </head>
                      <<<?xml version="33.33'>>>
                ]]>
            </title>
   </head>
  [...]
</html>
```

[39] Wöhr (2004), S. 56.
[40] Wöhr (2004), S. 57.

Es ist anzumerken, dass das `<title>` Element laut DTD keine weiteren untergeordneten Elemente enthalten darf. Auf keinen Fall darf auch das Wurzelelement innerhalb der geschachtelten Elemente wiederholt werden. Dieses XML-Dokument ist aber, durch die Verwendung des CDATA-Abschnittes, als korrekt zu betrachten, da die im CDATA-Bereich enthaltenen Elemente nicht interpretiert werden.

2.3.5 XML-Namensräume

„Namensräume stellen ein Konzept dar, das bei höheren Programmiersprachen bereits Tradition hat. Das Ziel bei Einführung von Namensräumen ist die *Vermeidung von Namenskonflikten*."[41]

Da XML-Dokumente oftmals in verschiedenen Anwendungen Verwendung finden, werden auch verschiedene Elemente genutzt, wobei es nicht auszuschließen ist, dass Elemente gleich bezeichnet sind. Um deren eindeutige Zuordnung zu einer Anwendung gewährleisten zu können, werden Namensräume eingeführt. Innerhalb der Namensräume müssen die Elemente eindeutig bezeichnet sein.

Im Beispiel wird ein sog. *Default-Namensraum* definiert:

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

Alle geschachtelten Elemente, sofern nicht explizit ein anderer Namensraum für sie definiert ist, werden durch diese Angabe als zum Default-Namensraum zugehörig betrachtet. Für weitere Informationen zu Namensräumen siehe (Wöhr (2004), S. 61ff).

2.3.6 XML-Parser

Um ein XML-Dokument computergestützt zu verarbeiten, muss es ausgelesen und der Software, die es nutzen soll, zugänglich gemacht werden. Dieser Schritt wird *Parsen* (aus dem Englischen: to parse = analysieren) genannt; als Ergebnis erlangt die Software Zugriff auf die Bestandteile des Dokumentes.

Grundsätzlich unterscheidet man zwischen Parsern, die ein XML-Dokument *validieren* und solchen, die nur die *Wohlgeformtheit* eines Dokumentes prüfen. Ein XML-

[41] Vgl. Wöhr (2004), S. 61.

Dokument wird als *wohlgeformt* bezeichnet, wenn es die folgenden Forderungen erfüllt[42]:

- Es hat genau ein Wurzelelement,
- Alle XML-Elemente werden mit einem End-Tag geschlossen,
- Die Elemente sind korrekt geschachtelt,
- Attributwerte stehen in einfachen oder doppelten Anführungszeichen,
- Attribute sind als Name=Wert-Paare definiert.

„Entspricht die *Struktur eines Dokumentes* den Regeln des zugeordneten Dokumenttyps [DTD], wird das Dokument als gültig (valid) bezeichnet. Die Gültigkeit setzt darüber hinaus die Wohlgeformtheit voraus.“[43]

Es existieren zwei standardisierte Programmierschnittstellen zum Parsen von XML-Dokumenten:

- Simple API for XML (SAX)
- Document Object Model (DOM)

Während DOM das XML-Dokument in Form eines Baumes an das verarbeitende Programm liefert, erzeugt SAX eine Reihe von Ereignissen, die beim Auslesen des Dokumentes ausgelöst werden. Beide Techniken haben Vor- und Nachteile, deshalb ist es vom Einsatzzweck abhängig, welche Schnittstelle gewählt wird.

Der DOM-Parser, vom W3C 1998 erstmals spezifiziert und freigegeben, ist in Web-Browsern generell vorhanden und ermöglicht das Navigieren, Lesen und Schreiben[44] in HTML- und XML-Dokumenten. Für weitere Informationen zu XML-Parsern vgl. (Wöhr (2004), S. 68ff).

[42] Vgl. Wöhr (2004), S. 54.
[43] Wöhr (2004), S. 58f.
[44] Vgl. Kapitel 2.4.3.1.

Tabelle 2: Vergleich SAX / DOM (vgl. Wöhr (2004), S. 68ff))

Kriterium	SAX	DOM
Zugriff	Ereignisgesteuert	Baumstruktur
Ende des Parsens	Ende des Auslesens	Beendigung der Bearbeitung durch das Programm
Lesen / Schreiben	Ja / Nein[45]	Ja / Ja
Ressourcen-verbrauch	Niedrig	Hoch (da Baum komplett im Speicher gehalten wird)
Art des Zugriffs	Nach der Reihenfolge (des Auftretens im XML-Dokument)	Gezielt (auf bestimmten Knoten im Baum)

2.3.7 (X)HTML

Die Hypertext Markup Language (HTML) ist eine auf SGML basierende Markup-Sprache. Es liegt ihr das Hypertext-Prinzip zu Grunde, jedoch bestand die Neuerung „in globalen Hypertext-Referenzen. Das referenzierte Dokument konnte sich auf einem anderen Rechner im Netzwerk befinden. Für diese Referenzierung entwickelte Berners-Lee eigens das HTTP-Protokoll.“[46]

Nachdem mit Version 4.01 die letzte HTML-Version standardisiert wurde, begann das W3C, das Konzept von SGML auf XML zu übertragen. Die weite Verbreitung und die zahlreichen Werkzeuge, bspw. zur Validierung, waren Argumente für den Wechsel. Weiterhin ist durch das Namensraum-Konzept die Erweiterung des Tag- und Attributvorrats in standardkonformer Weise möglich. Auch die Verknüpfung mit Stylesheets[47] und der Zugriff durch JavaScript[48] auf das Dokument ist durch die qualitativ hochwertigen, wohlgeformten und gültigen XHTML-Dokumente zuverlässiger und sicherer als mit reinem HTML.

Bei o.g. *XML-Beispieldatei* handelt es sich um ein XHTML-Dokument. Für dieses gelten alle für XML besprochenen Regeln und es beschreibt eine Dokumentenstruktur,

[45] Es existieren Möglichkeiten zum Schreiben eines XML Dokumentes mit SAX (vgl. Marchal (2008)).
[46] Wöhr (2004), S. 42.
[47] Vgl. Kapitel 2.4.2.
[48] Vgl. Kapitel 2.4.3.

die durch Komponenten, Layout und Scripting ergänzt, die clientseitige Präsentation (vgl. Abbildung 2-5) bildet. Serverseitig kann XHTML erzeugt und ergänzt werden, um Informationen an den User zu übertragen. Für weitere Details siehe (W3C (2008a)).

2.4 Client Technologien

Auf der Seite des Clients ist ein Anwendungsentwickler einer heterogenen Infrastruktur ausgeliefert. Verschiedene Hardware- und Softwarearchitekturen, unterschiedliche Rechenleistung und Software-Versionen erschweren es, eine verlässliche Basis zur Entwicklung einer Client-Anwendung vorzufinden und diese fehlerfrei zu implementieren.

Technologien wie Java[49], welche eine sog. Virtual Machine, die eine künstliche, aber homogene, Infrastruktur liefern, sind ein Ansatz für die Entwicklung. Allerdings muss der Anwender, möchte er die Anwendung benutzen, eine zweite Software, welche die Infrastruktur für die Ausführung bereitstellt, installieren. Dazu sind viele Benutzer nicht bereit bzw. nicht in der Lage.

Naheliegend ist es also, nur Programme und Schnittstellen zu nutzen, die jeder Benutzer, trotz unterschiedlicher Voraussetzungen, auf seinem Computer, Mobiltelefon oder sonstigen Endgeräten bereits installiert hat. Die Anwendung ist somit sofort einsetzbar. Web-Browser sind solche Programme.

2.4.1 Web-Browser

Ein Web-Browser (Browser) ist ein Programm zum Lesen, Anzeigen und Speichern von Dokumenten im WWW.

„Die am gesamten clientseitigen Präsentationsprozess beteiligten Technologien können [...] in vier Bereiche eingeteilt werden.“[50] Wie aus Abbildung 2-5 ersichtlich ist die clientseitige Präsentation, vom Browser dargestellt, zusammengesetzt aus der Dokumentenstruktur, der Formatierung, verschiedenen Komponenten und dem Scripting.

[49] Vgl. Sun Microsystems, Inc. (2008).
[50] Wöhr (2004), S. 41.

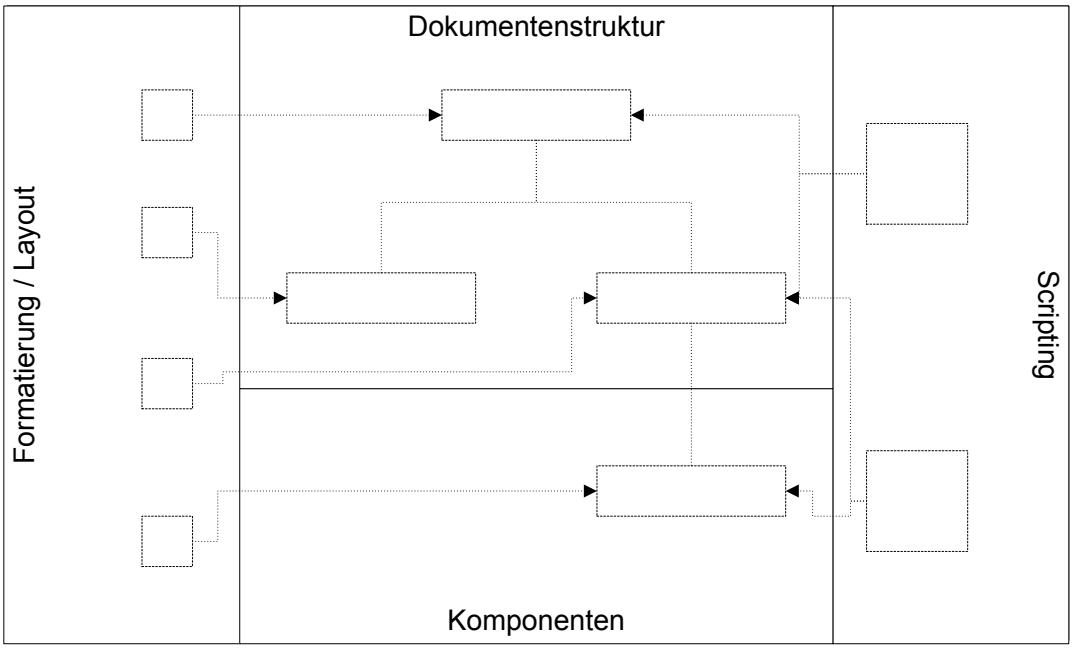


Abbildung 2-5: Bausteine der clientseitigen Präsentation (vgl. Wöhr (2004), S. 41)

Die Dokumentenstruktur wird durch HTML oder XHTML beschrieben. Da diese Dokumente allerdings auf verschiedenen Medien oder in verschiedenen Anwendungsbereichen einsetzbar sein sollen, ist es sinnvoll, die Formatierung, also das Layout, von den Strukturinformationen zu trennen. Dies kann durch Stylesheets erfolgen.

Ein weiterer Bestandteil sind die Komponenten, die sich in zwei Klassen aufteilen lassen. Die erste Klasse von Komponenten besteht aus Dateien, die, eingebettet in die Website, direkt vom Browser angezeigt bzw. ausgeführt werden können. Dies entscheidet der Browser nach Erkennung des MIME-Typs. Zu dieser Klasse gehören bspw. Bildformate (GIF, PNG, JPG, etc.), reiner Text oder JavaScript-Dateien.

Die zweite Klasse von Komponenten bilden Dateiformate, die nur durch die Installation einer Zusatzsoftware zur Erweiterung des Funktionsumfangs des Browsers (Plugin), scheinbar in die Website eingebettet, darstellbar sind oder gänzlich außerhalb des Browsers in einem Anwendungsprogramm auf dem Client ausgeführt werden. Zu dieser Klasse gehören unter anderem mit Flash und Java erstellte Websites oder Microsoft Word Dokumente, die in der zugehörigen Anwendung angezeigt werden. Nachteilig an dieser zweiten Klasse ist, dass im heterogenen Umfeld der verschiedenartigen Clients nicht sichergestellt werden kann, dass eine Anwendung oder ein Plugin installiert ist.[51]

[51] Vgl. Wöhr (2004), S. 42.

„Das *Skripting* fügt die Bestandteile zusammen und verleiht dem Dokument Dynamik.“[52] Skripte können durch Aktionen des Users oder andere Ereignisse ausgelöst werden, Komponenten austauschen, Knoten in die Dokumentenstruktur einfügen oder löschen und die Formatierung gezielt beeinflussen. Diese Aktionen werden komplett clientseitig ausgeführt. De-Facto-Standard für clientseitiges Scripting im Browser ist JavaScript.

Da Browser von verschiedenen Herstellern entwickelt werden (bspw. Internet Explorer[53], Firefox[54] oder Opera[55]), unterstützen sie auch verschiedene Befehle, die teilweise proprietärer Natur sind. Durch die Standards des W3C (XHTML, CSS) und JavaScript-Bibliotheken, die das Erstellen von Skripten für verschiedene Browser vereinfachen, existiert allerdings eine sichere Basis, auf der Web-Anwendungen programmiert werden können.

2.4.2 Stylesheets

„Cascading Stylesheets ermöglichen die Beschreibung von Layout und Präsentation für HTML- und XML-Dokumente.“[56] Dies ermöglicht die:

- Trennung von Inhalt und Präsentation,
- Reduktion der Dokumentengröße,
- Verwendung von (X)HTML ausschließlich zur Strukturbeschreibung.

Das (X)HTML-Dokument ist in einem Web-Browser generell als DOM-Baum verfügbar. Der Entwickler kann den verschiedenen Knoten, anhand ihres Tag-Namens, einer vergebenen ID oder Klasse, Stil-Informationen zuweisen. Dies geschieht mittels sich überschneidender, kaskadierender Stylesheets. So kann die Position des Elements, dessen Rahmen, dessen Hintergrundfarbe, die Schriftart des Inhaltes usw. manipuliert werden. Dabei werden die Stil-Attribute untergeordneten Elementen teilweise vererbt.

Die Stil-Informationen können sowohl extern (in separaten Stylesheets) oder auch direkt im (X)HTML-Dokument, über das Attribut `style`, vergeben werden. Prinzipiell ist die externe Zuweisung vorzuziehen, allerdings sind diese Informationen durch

---

[52] Wöhr (2004), S. 42.
[53] http://www.microsoft.com/germany/windows/products/winfamily/ie/default.mspx, aufgerufen am 6. August 2008.
[54] http://www.mozilla-europe.org/de/firefox/, aufgerufen am 6. August 2008.
[55] http://de.opera.com/, aufgerufen am 6. August 2008.
[56] Wöhr (2004), S. 86.

JavaScript nicht manipulierbar, so dass auch vereinzelt Stil-Informationen im (X)HTML-Dokument vergeben werden müssen.

Für weitere Informationen zu CSS und XSL (eXtensible Stylesheet Language) siehe (Wöhr (2004), S. 86ff).

2.4.3 JavaScript

Die Firma Netscape Communications entwickelte die interpretierte Sprache JavaScript für den Browser Netscape Navigator. Sie kann client- und serverseitig eingesetzt werden; an dieser Stelle soll nur die clientseitige Verwendung thematisiert werden.

Um dem Benutzer eine komfortable Bedienung einer Web-Anwendung im Browser zu ermöglichen, „wie er dies von GUI-basierten Anwendungen gewohnt ist“[57], muss JavaScript die folgenden Funktionalitäten bieten:[58]

- Zugriff auf die Bestandteile des geladenen Dokumentes,
- Dynamische Änderung des geladenen Dokumentes,
- Steuerung externer Komponenten,
- Reaktion auf Benutzeraktionen,
- Fenstermanagement.

Als Skriptsprache ist JavaScript typenlos, nutzt eine an Java angelehnte Syntax und ist objektbasiert. Allerdings ist „JavaScript [...] keine objektorientierte Sprache [...] [da einige] wesentliche OO-Paradigmen [...] nicht unterstützt“[59] werden. So unterscheidet sie „nicht zwischen Objekten und Klassen, sondern kennt nur das Konzept der Objekte“[60]. Dies nennt man Prototypen-Konzept. Jedes Objekt wird durch eine Vorlage, ein sog. prototypisches Objekt, erzeugt. Dies ist allerdings selbst bereits ein Objekt, so dass verschiedene Operationen mit ihm durchgeführt werden können. Weiterhin ist es möglich, Objekte „zur Laufzeit um beliebige Methoden und Attribute“[60] zu erweitern.

JavaScript wird im Browser auf drei verschiedene Weisen ausgeführt. Es kann innerhalb eines `script`-Elements im (X)HTML-Dokument, als Event-Handler oder als sog. JavaScript-URL ausgeführt werden. Letztere Möglichkeiten sind stilistisch jedoch nicht

[57] Wöhr (2004), S. 117.
[58] Vgl. Wöhr (2004), S. 117.
[59] Wöhr (2004), S. 118.
[60] Ziegler (2001).

zu empfehlen, da es dabei zu einer Vermischung von (X)HTML und JavaScript kommt. Generell sollte JavaScript in gesonderten Dateien abgelegt werden, die über das `script`-Element eingebunden werden.

Über das DOM erhält JavaScript Zugriff auf die geladene Website und alle Elemente darin. Diese werden vom Browser als Objekte zur Verfügung gestellt und können, wie im folgenden Abschnitt beschrieben, gelesen und auch manipuliert werden, was zu einer dynamischen Änderung des geladenen Dokumentes führt.

2.4.3.1 DOM-Manipulation

„Die [...] durch das W3C im Oktober 1998 freigegebene DOM-Spezifikation beschreibt ein Objektmodell für HTML und XML.“[61] Wie bereits bei den XML-Parsern beschrieben, erstellt der Browser in seinem Speicher einen Baum der Elemente eines HTML-Dokumentes. Dieser Baum wird dann vom Browser interpretiert und dem Benutzer präsentiert. Weiterhin enthält der Baum alle Informationen über den Zustand der Objekte, die angezeigt werden. Mittels CSS und JavaScript kann auf diesen Baum zugegriffen werden, wie Abbildung 2-6 zeigt:

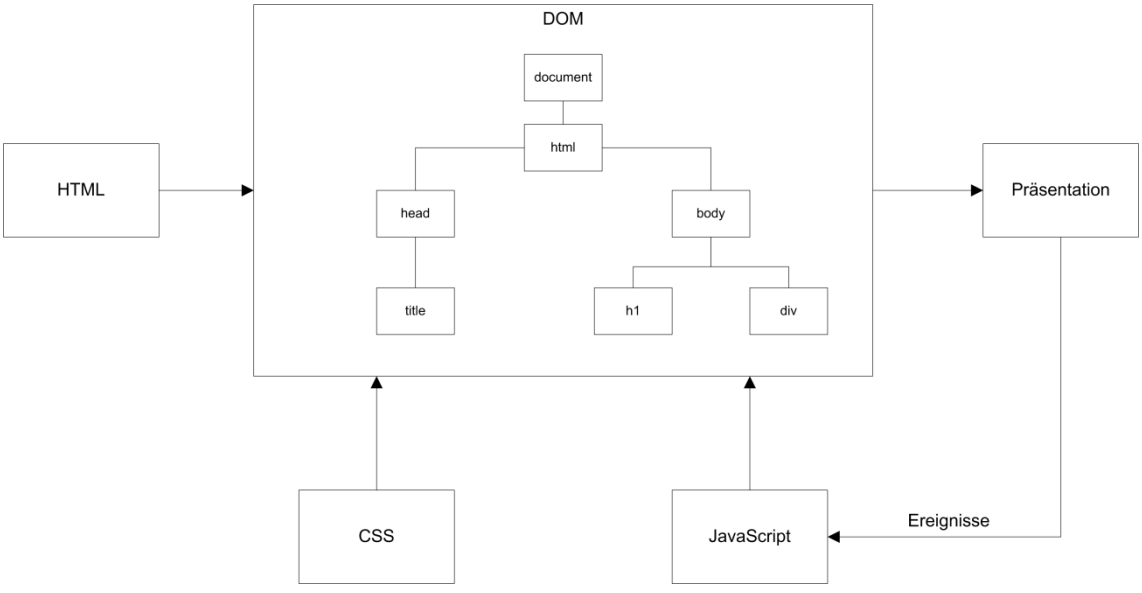


Abbildung 2-6: DOM im Kontext des Browsers (vgl. Wöhr (2004), S. 75)

Ein einfaches Beispiel, wie CSS den DOM beeinflusst, ist das Festlegen der Hintergrundfarbe des `body`-Elements und eines Rahmens für `div`-Elemente:

[61] Wöhr (2004), S. 69.

```
body {background-color: red;}
div {border-style: solid; border-color: #000000;}
```

In diesem Beispiel wird über den Tag-Namen (`body`, `div`) auf die Elemente zugegriffen und diesen dann Stil-Informationen zugeordnet. Soll nicht für alle Elemente mit gleichem Tag-Namen der gleiche Stil verwendet werden, so müssen Klassen oder IDs für Elemente definiert werden.

Mittels JavaScript soll nun die Farbe des Textes in allen `div`-Elementen auf die Farbe Grün geändert werden. Dies wird durch den folgenden Codeabschnitt realisiert:

```
var allDivs = document.getElementsByTagName ("div");
for (var i = 0; i < allDivs.length; i++)
        allDivs[i].style.color = "green";
```

JavaScript ist weiterhin dazu in der Lage, Knoten in den DOM-Baum einzufügen und zu löschen. Dies kann ereignisgesteuert geschehen, also als Event, das vom Benutzer durch eine Aktion ausgelöst wird. Dadurch wird der Eindruck erweckt, eine GUI-basierte Anwendung im Browser zu bedienen.

2.4.3.2 AJAX

„AJAX ist zwar keine neue beeindruckende Technologie, dennoch treibt sie eine aktuelle Generation von Web-Applikationen an, indem die volle Ausschöpfung und neue Kombination der schon lange vorhandenen Potentiale bestehender Webtechniken ausgenutzt wird.“[62]

Das Akronym AJAX bedeutet „Asynchronous JavaScript And XML“. Die Technologie ermöglicht, innerhalb einer bereits geladenen Website eine Server-Anfrage durchzuführen und erzeugt so den Eindruck einer schnelleren und dynamischeren Reaktion auf Nutzeranfragen. Die Ergebnisse liefert der Server in XML zurück. Diese werden mit JavaScript verarbeitet und per DOM-Manipulation in die geladene Seite eingefügt.

Es sei erwähnt, dass zum Datenaustausch nicht ausschließlich XML verwendet wird. Es existieren weitere Austauschformate, wie bspw. JSON[63]. Die Vorteile von XML wurden

[62] El Moussaoui/Zeppenfeld (2008), S. 111.
[63] http://json.org/, aufgerufen am 10. August 2008.

jedoch im Rahmen dieser Arbeit bereits erläutert[64], so dass auf andere Austauschformate an dieser Stelle nicht eingegangen werden soll.

Wie in Abbildung 2-7 ersichtlich, führt in klassischen Webanwendungen jede Benutzerinteraktion dazu, dass eine neue Website geladen werden muss, damit der Benutzer über das Ergebnis seiner Anfrage informiert wird. Dies ist mit Hilfe der AJAX-Technologie nicht mehr nötig, die Seite wird nach einer Anfrage entsprechend aktualisiert.

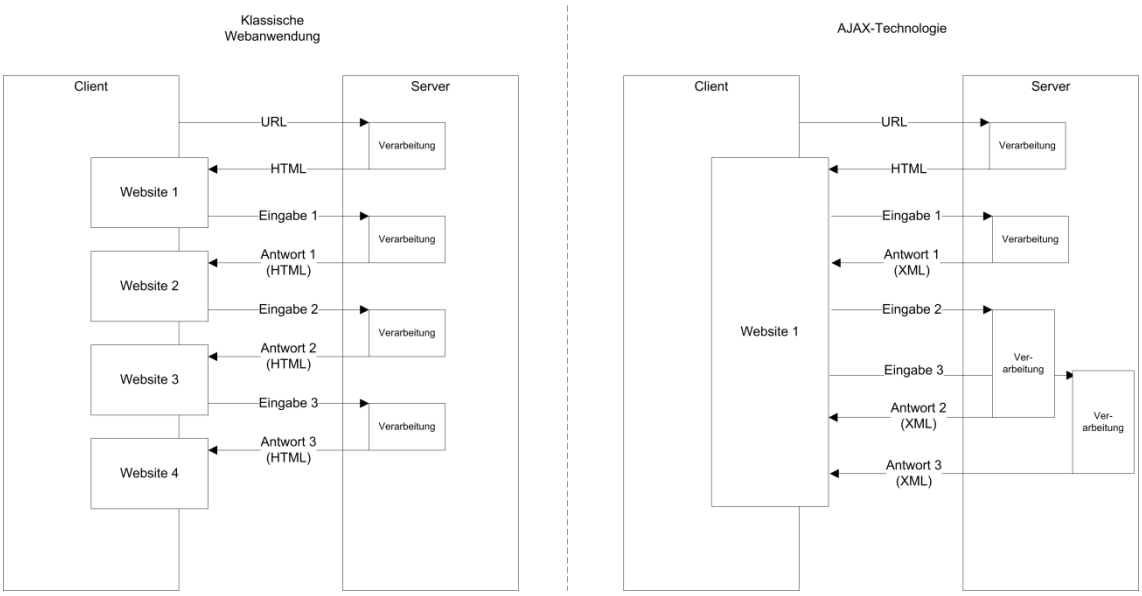


Abbildung 2-7: Vergleich Klassische Webanwendung / AJAX-Technologie

Ein weiterer Vorteil ist die ständige Interaktion des Nutzers mit der Website; selbst wenn eine Anfrage noch nicht beantwortet ist hat er die Möglichkeit, weitere Anfragen abzusenden, wie dies in Abbildung 2-7 bei Eingabe zwei und drei der Fall ist. Ob dies gewünscht ist, muss der Entwickler der Webanwendung entscheiden.

Weitere Vorteile der Technologie: [65]

- Unter der Voraussetzung, dass JavaScript vom Benutzer nicht deaktiviert wurde, ist die Technologie plattformunabhängig und mit allen Browsern benutzbar.
- Durch die verminderte Menge an ausgetauschten Daten ergibt sich eine Kostenersparnis für den Betreiber der Applikation.

[64] Vgl. Kapitel 2.2.4.
[65] Vgl. El Moussaoui/Zeppenfeld (2008), 111ff.

- Sprachunabhängigkeit auf der Serverseite, da die Anfragen per HTTP erfolgen und XML als Austauschformat genutzt wird.

2.4.3.3 AJAX-Frameworks

„AJAX-Frameworks können gleich zwei positive Effekte haben: Zum einen läuft der Code auf möglichst vielen Clients, zum anderen spart man sich durch die Verwendung vorgefertigter Effekte mitunter eine Menge Zeit und Aufwand.“[66]

Durch die Vielzahl an auf Clients eingesetzten Browsern, Betriebssystemen und Versionsständen ist es nahezu unmöglich, Web-Applikationen in all diesen Umgebungen zu testen. Auch die Ansteuerung von verschiedenen Technologien, wie AJAX-Abfragen oder das Reagieren auf Ereignisse (Eventhandling), sind auf viele verschiedene Arten in verschiedenen Browsern implementiert.

Dieser Komplexität begegnet man mit AJAX-Frameworks. Dabei handelt es sich um Bibliotheken, die in JavaScript implementiert sind. Diese beschleunigen die Entwicklung von Web-Applikationen durch Funktionen, welche die Inkompatibilitäten beseitigen. Meist liefern diese Frameworks auch eine Menge von nützlichen Hilfsfunktionen, mit deren Hilfe die Anwendungsentwicklung weiter vereinfacht wird.

Beispiele für AJAX-Frameworks sind das *DOJO Toolkit*[67], *jQuery*[68] oder das *Prototype-Framework*[69]. Der Funktionsumfang dieser Frameworks kann als gleichwertig betrachtet werden; es ist also eine Geschmacksfrage, welches Framework von einem Entwickler genutzt wird. Im Rahmen dieser Arbeit wurde das *Prototype-Framework* gewählt, da es einen hohen Bekanntheitsgrad besitzt und weit verbreitet ist.

Aufbauend auf AJAX-Frameworks existieren weitere Bibliotheken, die das Erzeugen von graphischen Effekten oder Navigationselementen erleichtern. Genutzt werden hier die Bibliotheken *script.aculo.us*[70] und *Control.Tabs*[71].

Für weitere Informationen zu AJAX-Frameworks (vgl. Wenz (2007), S. 465ff).

---

[66] Wenz (2007), S. 465.
[67] http://dojotoolkit.org/, aufgerufen am 12. August 2008.
[68] http://jquery.com/, aufgerufen am 12. August 2008.
[69] Vgl. Stephenson (2008).
[70] http://script.aculo.us/, aufgerufen am 12. August 2008.
[71] http://livepipe.net/control/tabs, aufgerufen am 12. August 2008.

2.5 Server Technologien

Eine dynamische Web-Applikation auf Basis der Client-Server-Architektur benötigt serverseitig, wie bereits beschrieben, einen Web-Server. Dieser liefert nach Aufruf eines ihm zugeordneten URL die Client-Applikation, bestehend aus XHTML-, CSS-, JavaScript-Dateien und AJAX-Frameworks, aus.

In der Folge müssen weitere Informationen vom Server an den Client gesendet werden, was auf Basis der Eingaben des Benutzers geschieht und dynamisch ist. Dies erledigen „*serverseitige Programme*, die Dokumente auf Client-Anfragen hin generieren. [...] Deren konkrete Inhalte stammen häufig aus stets aktualisierten Datenbanken.“[72] Allerdings können die eingesetzten Ressourcen zur Generierung der Seiten auch statische Dokumente oder Internet-Ressourcen sein. Ebenfalls möglich ist die dynamische Übertragung von ausführbarem Code in Form von JavaScript oder bspw. Java Applets.

„Die wichtigsten und am weitesten verbreiteten Technologien“[73] für serverseitige Programmierung sind:

- Common Gateway Interface (CGI),
- PHP: Hypertext Preprocessor (PHP),
- Java Servlets und Java Server Pages (JSP),
- Active Server Pages (ASP).

Im Rahmen dieser Arbeit wird der *Apache Tomcat* [74] Server eingesetzt, der sowohl einen klassischen Web-Server darstellt, als auch *Java Servlets* und *JSP* ausführen kann. Die eingesetzten Ressourcen sind eine Datenbank und die Java-Klassenbibliothek *XTMGenerator*[75], die *Wikipedia*-Seiten in XTM-Dateien umwandelt.

2.5.1 Java Servlets

Bei Java Servlets handelt es sich um Java-Klassen, welche die Servlet-Klassenbibliothek nutzen, die von der Java 2 Enterprise Edition[76] (J2EE) bereitgestellt

[72] Wöhr (2004), S. 23.
[73] Wöhr (2004), S. 25.
[74] http://tomcat.apache.org/, aufgerufen am 12. August 2008.
[75] Vgl. Twele (2008).
[76] http://java.sun.com/javaee/, aufgerufen am 12. August 2008.

Produkte, können die Servlets ausführen. Grund dafür ist die umfassende Standardisierung der Servlet-Bibliothek.

Die Funktionsweise dieser Servlet Container ist bei allen Produkten gleich und in Abbildung 2-8 verdeutlicht. „Für jedes Servlet muss zur Laufzeit mindestens eine Instanz existieren. Jede Client-Anfrage an diese Instanz wird in einem separaten Thread ausgeführt.“[78]

Die Servlets nutzen weitere Java-Klassen um die Client-Anfragen zu beantworten. Diese wiederum greifen auf Ressourcen zu, bspw. über die Java-Schnittstellen JDBC (Java Database Connectivity) und Java.NET.

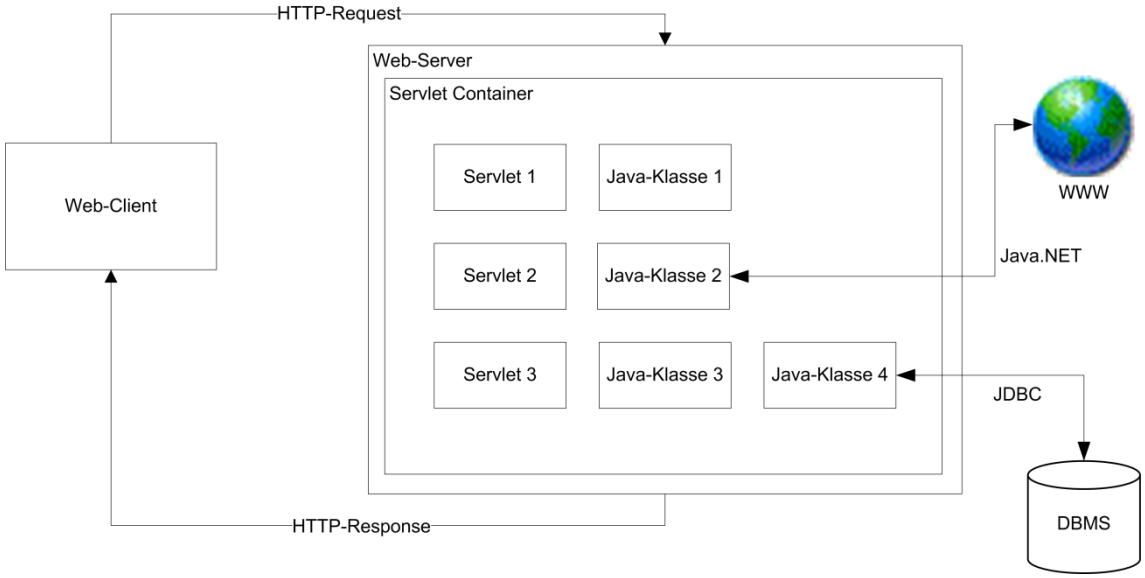


Abbildung 2-8: Funktionsweise eines Servlet Containers (vgl. Wöhr (2004), S. 317)

Mit diesen Ressourcen wird eine Antwort, die sog. HTTP-Response, generiert und an den Client zurückgesendet.

Für weitere Informationen zu Java Servlets und Servlet Containern siehe (Wöhr 2004, 317ff).

[78] Wöhr (2004), S. 317.

2.5.3 Ressourcen

Eine Anwendung, die auf Ressourcen zurückgreift, nennt man datenorientiertes System oder Informationssystem. Dieses „beschäftigt sich nicht nur mit der eigentlichen Verarbeitung von Daten, sondern mit deren effizienten Speicherung zur Befriedigung von Informationsbedürfnissen. [...] Die technologisch einheitliche Speicherung, Verarbeitung und Bereitstellung der für Informationen wichtigen Daten erfolgt im Allgemeinen in Form von ***Datenbanken*** *(data bases).*“[79]

Um von einer Java-Klasse auf eine Datenbank zuzugreifen wird die Schnittstelle *JDBC* genutzt (vgl. Abbildung 2-8). Diese verfügt über eine einheitliche Schnittstelle in Richtung Java-Klasse, muss jedoch über einen, für die genutzte Datenbank passenden, Treiber verfügen, um die Verbindung zum jeweiligen Datenbanksystem herzustellen. Zu den kommerziellen Datenbanksystemen gehören[80] *ORACLE*, *SYBASE*, *DB2*, *Informix*, *ACCESS* oder auch *INGRES*. Wichtige Open-Source Datenbanksysteme sind *MySQL*[81] und *PostgreSQL*[82].

Zur Kommunikation mit einem relationalen Datenbanksystem, das heißt zum Abrufen und Manipulieren der Daten und zur Definition von Datenbanken, wird die standardisierte Datenbanksprache *SQL* (Structured Query Language) genutzt. „Bei so gut wie allen Client-Server-Systemen wird die Kommunikation zwischen Client und Server über SQL abgewickelt und auch Desktop-Datenbanksysteme wie beispielsweise dBase oder Access lassen sich per SQL-Anweisung steuern. Wenn Sie mit Datenbanken zu tun haben, kommen Sie an SQL kaum vorbei.“[83]

Für weitere Informationen zu Datenbanken, vor allem in Zusammenhang mit Java, respektive JDBC, siehe (Saake/Sattler (2000)).

Eine weitere in Abbildung 2-8 angegebene Ressource ist das WWW. Dies ist jedoch ebenfalls eine statische Ressource, wie bspw. eine statische HTML-Seite, oder der indirekte Zugriff auf eine Datenbank. Dieser indirekte Zugriff erfolgt über eine Anfrage an ein serverseitiges Programm auf dem Zielrechner. Diese Anfrage wird per URL abgesendet; in diesem Fall ist das Servlet wiederum Client eines Servers im Internet und erhält dessen HTTP-Response, die dann, manipuliert oder nicht, an den eigentlichen Client weitergereicht wird.

---

[79] Dumke (2001), S. 214.
[80] Vgl. Dumke (2001), S. 216.
[81] http://www.mysql.de/, aufgerufen am 14. August 2008.
[82] http://www.postgresql.org/, aufgerufen am 14. August 2008.
[83] Ebner (2002), S. 9.

In dieser Arbeit werden als Ressourcen ein *MySQL*-Datenbanksystem, welches per *JDBC* mit dem Servlet Container verbunden ist und per *SQL* Befehle empfängt, und die *WWW*-Ressource *Wikipedia* eingesetzt.

Nachdem die grundlegenden Technologien besprochen sind, widmen sich die folgenden Abschnitte dem Konzept der semantischen Suche. In diesem Zusammenhang werden Topic Maps beschrieben und die auf XML basierende Spezifikation von Topic Maps, XTM 1.0, erläutert.

2.6 Semantische Netze

„Die grundlegende Idee [des Semantic Web] besteht darin, Inhalte im Web so anzureichern, dass sie nicht nur für Menschen verständlich sind, sondern auch von Maschinen zumindest soweit erfasst werden können, dass Automatisierung auch auf der Ebene der *Bedeutung* möglich wird.“[84] Das Semantic Web ist ein Ansatz von Tim Berners-Lee, das Konzept der semantischen Netze für das WWW umzusetzen.

Die Semantik als Teilgebiet der Sprachwissenschaft befasst sich mit Sinn und Bedeutung von Sprache und sprachlichen Zeichen.

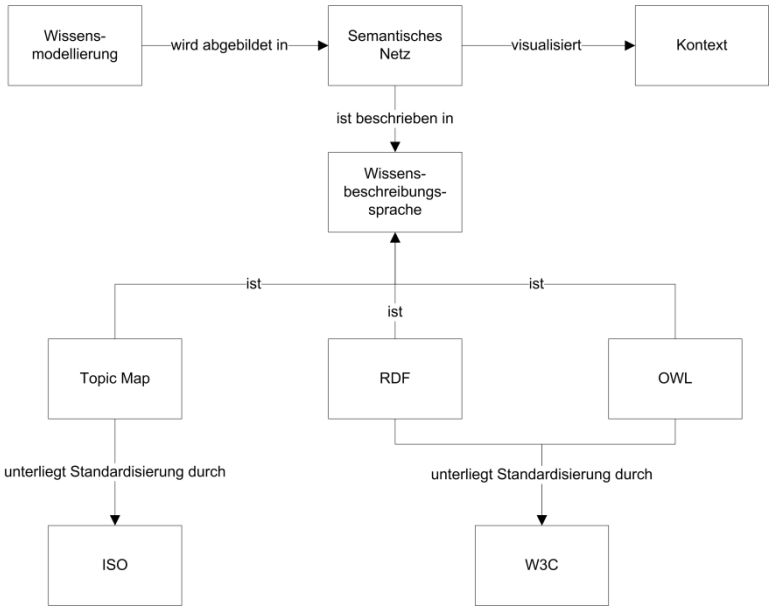


Abbildung 2-9: Sematisches Netz (vgl. punkt. netServices (2008))

[84] Tochtermann/Maurer (2006), S. 2.

„Ein semantisches Netz [...] kann man sich analog zu einem realen Netz denken, in dem eine Menge von Seilen miteinander verknotet (verbunden) sind. Im semantischen Netz entsprechen die Knoten den einzelnen Begriffen (Konzepten) und die Seile sind die Verbindungen oder Relationen zwischen diesen Konzepten.“[85] Semantische Netze sind also Werkzeuge zur Wissensmodellierung.

In Abbildung 2-9 ist das Konzept des semantischen Netzes visualisiert. Wie ersichtlich, werden semantische Netze[86] mit Wissensbeschreibungssprachen[87] beschrieben, für die hier drei Beispiele angegeben werden. Um das Wissen zu beschreiben, welches in das semantische Netz einfließt, und um den Kontext eines Begriffes abzubilden, kann man die folgenden Forderungen an Wissensbeschreibungssprachen ableiten:

1. **Wissensbeschreibungssprachen sollen einfache Taxonomien beschreiben können.**
   Taxonomien beschreiben einfache Hierarchien, bspw. werden in der Biologie Verwandtschaftsbeziehungen zwischen Tieren in einem hierarchischen System erfasst. Dies sind „ist ein“-Beziehungen, wie bspw. *ein Mensch ist ein Säugetier*.[88]

2. **Wissensbeschreibungssprachen sollen Thesauri beschreiben können.** Neben Ober- und Unterbegriffen (Taxonomie) werden weitere Relationen benötigt. Diese beschreiben Deskriptoren und Non-Deskriptoren über eine Äquivalenzbeziehung und verknüpfen Abkürzungen und Akronyme und andere verwandte Begriffe mit dem Deskriptor. Thesauri beschreiben eine Wissensdomäne möglichst vollständig und sind so zur Bildung einer Fachsprache geeignet.[89]

3. **Wissensbeschreibungssprachen sollen Metadatenmodelle beschreiben können.**
   Es sollte möglich sein, Begriffe in Kategorien einzuteilen, Klassen von Objekten zu definieren und Subklassen zu beschreiben. Eine Möglichkeit zur Definition von Werte- und Gültigkeitsbereichen von Eigenschaften einer Klasse sollte vorhanden sein.

Eine weitere Anforderung im Sinne der Automatisierung ist die Umsetzung von logischen Inferenzmechanismen, bspw. das Ermitteln von disjunkten Subklassen mit

[85] Krings (1996).
[86] Vgl. Kienreich/Strohmaier (2006).
[87] Auch: Wissensmodellierungssprachen.
[88] Vgl. Kienreich/Strohmaier (2006).
[89] Vgl. Wendt (1997).

transitiven Eigenschaften. Diese Forderungen werden jedoch nur von Ontologien gestellt, für deren Beschreibung das W3C die Ontologie-Beschreibungssprache OWL entwickelt und standardisiert hat.

Im nächsten Kapitel werden Topic Maps vorgestellt, die nicht zur Semantic Web Initiative des W3C gehören, sondern von der ISO zertifiziert werden und einen alternativen Ansatz zur Frame-basierten[90] Wissensmodellierung der W3C Semantic Web Standards bieten.

2.7 Topic Maps

Eine Topic Map „orientiert sich [...] an menschlichen Denkprozessen, in denen es [...] keinen zentralen Einstiegspunkt gibt, kein klares ‚oben und unten', sondern der Mensch erschließt sich neue Erkenntnisse aus verschiedenen Blickwinkeln – immer geprägt aus der aktuellen Situation seines Denkens und Handelns heraus."[91] Dennoch lassen sich mit Topic Maps hierarchische Strukturen beschreiben; sie erfüllen alle Forderungen an eine Wissensbeschreibungssprache.

„[...] die Idee hinter Topic Maps [ist], ähnlich wie bei Lexika, Glossaren oder Indexen, externe Dokumente (etwa Bilder oder Artikel in einem Lexikon) zu referenzieren und die Grundbestandteile dieses Wissens, die Topics (etwa, worüber ein Artikel in einem Lexikon handelt), miteinander in Verbindung zu bringen und so eine Wissensbasis aufzubauen, die die Suche und Navigation innerhalb dieses Wissens und der (oftmals großen) Mengen an zugehörigen Dokumenten erleichtern und die Geschwindigkeit und Qualität dieser Suchvorgänge verbessern soll."[92]

Topic Maps beschreiben semantische Netze. Sie existieren losgelöst von den in ihnen beschriebenen Wissensobjekten, sind verknüpfbar und austauschbar. Weiterhin erlauben Topic Maps eine Bottom-Up-Wissensmodellierung, die, im Gegensatz zum Entity-Relationship-Paradigma und den Frame-Modellierungen der Semantic Web Initiative des W3C, keine abstrakten Klassen auf der Meta-Ebene benötigen, bevor ein semantisches Modell formuliert werden kann.[93]

---

[90] Vgl. Birkenbihl (2006).
[91] Beier (2006), S. 263.
[92] Widhalm/Mück (2002), S. 6.
[93] Vgl. Beier (2006), S. 268f.

Tabelle 4: Konzepte des ISO 13250 Standards (vgl. Widhalm/Mück (2002), S. 6ff)

Konzept	Beschreibung
Topic	Ein Topic ist ein elementares Subjekt im Kontext des modellierten Wissens; eine Entität. Dies kann alles Beschreibbare sein, bspw. eine Person, ein Gegenstand, ein Wort oder ein Dokument. Topics können Instanzen von 0 bis n anderen Topics sein, sie sind dann „vom Typ“ des übergeordneten Topics.
Topic Names	Bezeichnungen für Topics können vielerlei Gestalt annehmen: volle Namen, Abkürzungen, Akronyme, Katalognummern, etc. Der Standard bietet drei Varianten: • **Base Name**: Ist der „eigentliche“ Name eines Topics. Jedes Topic benötigt einen Namen, es kann allerdings auch mehrere Base Names in verschiedenen Gültigkeitsbereichen (Scopes) geben. • **Display Name**: Optionale Zeichenfolge, die zur Darstellung eines Topics genutzt wird. • **Sort Name**: Optionaler Name, welcher zur Sortierung herangezogen wird. Wird kein Display oder Sort Name angegeben, übernimmt der Base Name diese Rolle.
Topic Occurrences	Durch Occurrences werden externe Web-Ressourcen oder Dokumente mit dem Topic verknüpft. Dies können bspw. Artikel oder Bilder sein.
Public Subject Descriptor	Ist eine eindeutige Beschreibung eines Topics. Dies kann bspw. eine Ident-Nummer für Produkte, eine ISBN-Nummer für Bücher oder eine Steuernummer für Personen sein.
Associations	Associations (Assoziationen) beschreiben Beziehungen zwischen Topics. Dabei wird keine Aussage über die Art der Assoziation (transitiv, symmetrisch, reflexiv) getroffen. Assoziationen können beliebig viele Topics verknüpfen, diese benötigen dann allerdings eine Rolle in der Assoziation, die als Topic definiert sein muss. Auch ein Assoziations-Typ kann als Topic definiert und mit ihr verknüpft werden. Beispiele hierfür wären „erbaute“, „grenzt an“ oder „komponierte“.

Konzept	Beschreibung
Scopes	Scopes legen Gültigkeitsbereiche fest. So können Topics den gleichen Namen, jedoch einen anderen Public Subject Descriptor besitzen. Mittels Scopes werden diese Topics voneinander abgegrenzt und ihrer jeweiligen Wissensdomäne zugeordnet. Auch für Topic Names (Beispiel: verschiedene Sprachen) und Assoziationen kann dieses Konzept genutzt werden.
Facets	Facets (oder Facetten) erlauben es, Informationsobjekten Eigenschafts-Wert-Paare zuzuordnen. Dies können Topics, aber auch Assoziationen sein. So könnte einem Topic, das ein Land beschreibt, die Eigenschaft „Einwohnerzahl“ und die konkrete Zahl als Wert zugeordnet werden. Werte unterstützen keine Datentypen und werden immer als Zeichenketten hinterlegt. Facetten können weitere Facetten besitzen, bspw. das Jahr der Erhebung der Einwohnerzahl.
Topic Maps	Die bereits beschriebenen Konzepte werden in Topic Maps zusammengefasst. Diese können selbst in einem bestimmten Gültigkeitsbereich definiert sein, der für alle in der Topic Map beschriebenen Informationsobjekte gilt. Es ist möglich Topic Maps zu kombinieren und, wie es die Entwickler des Standards vorschlagen, sog. Topic Map Templates, also Vorlagen, zu benutzen.
Bounded Object Sets	Bounded Object Sets (BOS) sind Dokumentenbäume, die der verarbeitenden Applikation bekannt sind und entspringen dem auf SGML aufbauenden HyTime-Standard. Da das BOS-Konzept in der XTM-Spezifikation aufgegeben wurde, soll es hier nicht näher erläutert werden.

Der Standard ISO 13250 zu Topic Maps gliedert sich in die in Tabelle 4 beschriebenen Bestandteile und baut im Wesentlichen auf der SGML-Erweiterung HyTime[94] auf. Da SGML, wie bereits beschrieben, von XML abgelöst wird, werden die Elemente des Standards hier nur kurz erwähnt und die ausführliche Beschreibung anhand der *XTM 1.0* Spezifikation im nächsten Kapitel durchgeführt.

[94] Vgl. Widhalm/Mück (2002), S. 123ff.

2.8 Die XTM-Spezifikation

„Im Dezember 2000 veröffentlichte die TopicMaps.Org Authoring Group, eine von den Autoren des ISO-Standards 13250, M. Biezunski und S. R. Newcomb, angeführte, unabhängige Autorengruppe, die XML Topic Maps (XTM) Spezifikation in der Version 1.0.“[95]

Dies bedeutete nicht nur die Portierung des Standards auf XML, sondern erweiterte diesen auch um einige Sprachkonstrukte. So ist feststellbar, dass „ISO 13250 und XTM [...] in einigen Aspekten eine unterschiedliche Betrachtungsweise des Themas Topic Maps“[95] aufweisen. Sowohl die Namen einiger Elemente, als auch die Dokumentstruktur ist im Vergleich zu ISO 13250 verändert worden.

Die XTM 1.0 DTD[96] bietet keine Möglichkeit zur Umbenennung von Elementnamen, wie es im ISO-Standard vorgesehen ist, und durch die Unterstützung von validen XML-Dokumenten bildet jede XTM-Datei genau eine Topic Map ab, die Bounded Object Sets aus HyTime werden nicht unterstützt. Das Verknüpfen mehrerer Topic Maps geschieht nunmehr durch das neu definierte `mergeMap`-Element. Weiterhin wurde das Facets-Konstrukt ersatzlos gestrichen, es kann durch Occurrences und Assoziationen ersetzt werden.

2.8.1 Das XTM-Dokument

Ein XTM-Dokument[97] ist, da es auf XML aufbaut, auch ein XML-Dokument. Deshalb besteht es ebenfalls aus einem *Prolog* und der *Dokumenteninstanz*. Für XTM-Dokumente muss der folgende Prolog definiert werden:

```
<?xml version="1.0"?>
<!DOCTYPE topicMap PUBLIC "-//TopicMaps.Org//DTD XML Topic Map
         (XTM) 1.0//EN" "xtm1.dtd">
```

Zu beachten ist hier, dass der Dokumententyp auf die XTM 1.0 DTD verweist, die alle gültigen Sprachelemente enthält und die Struktur der Dokumenteninstanz vorgibt.

Die Dokumenteninstanz besteht aus einem Wurzelelement und den geschachtelten Elementen, die in den folgenden Abschnitten beschrieben werden. Eine beispielhafte Dokumenteninstanz ist in Anhang E dieser Arbeit zu finden.

---

[95] Widhalm/Mück (2002), S. 369.
[96] Vgl. Pepper/Moore (2008).
[97] Vgl. Widhalm/Mück (2002), S. 369ff.

2.8.2 Das Wurzelelement: topicMap

Jedes XTM-Dokument muss als Wurzelelement das `topicMap`-Element enthalten. Dieses umklammert die Dokumenteninstanz und muss die folgenden Attribute beinhalten:

- `xmlns="http://www.topicmaps.org/xtm/1.0/"`
  Dieses Attribut definiert den Standard-Namensraum für das gesamte Dokument und verweist auf die XTM 1.0 Spezifikation.

- `xmlns:xlink="http://www.w3.org/1999/xlink"`
  Hier wird der Namensraum für die *XLink*-Attribute der geschachtelten Elemente definiert und auf die *XLink*-Spezifikation des W3C verwiesen.

Wie in der XTM DTD spezifiziert, dürfen als geschachtelte Elemente in der ersten Ebene nur die Elemente `topic`, `association` und `mergeMap` dem Wurzelelement untergeordnet werden. Nachdem die Referenzmechanismen in einer Topic Map beschrieben werden, wird einzeln auf diese Elemente eingegangen.

2.8.3 Referenzmechanismen

„Als *Subject Indicator* wird in XTM allgemein etwas definiert, das die Identität eines Subjekts eindeutig beschreibt bzw. eine *Subject Identity*, die Identität eines Subjekts, anzeigt.“[98] Dies wird durch die drei in XTM definierten Referenzmechanismen umgesetzt:

- Das Element: `topicRef`
  Über ein `xlink:href`-Attribut wird auf ein `topic`-Element verwiesen. Dies muss nicht in derselben Topic Map definiert sein.

- Das Element: `subjectIndicatorRef`
  Beschreibt ein Subjekt eindeutig und nutzt die gleiche Syntax wie das `topicRef`-Element. Der Unterschied zu diesem besteht darin, dass das Ziel des Links kein Topic, sondern eine Web-Ressource ist.

- Das Element: `resourceRef`
  Auch dieses Element nutzt als Attribut `xlink`, jedoch zeigt es auf Web-Ressourcen, die das Element nicht eindeutig beschreiben, wie dies bei `subjectIndicatorRef` der Fall ist.

[98] Widhalm/Mück (2002), S. 371.

2.8.4 Topics in XTM

Das `topic`-Element beschreibt ein elementares Subjekt im Kontext des modellierten Wissens, wie es das Konzept des ISO Topic Map Standards vorgibt. Das Element an sich enthält nur ein Attribut, die ID, die dem Topic einen eindeutigen Identifikator innerhalb der Topic Map zuordnet und über welche es referenziert werden kann. Weiterhin enthält das Element die folgenden Sub-Elemente, die das Subjekt näher beschreiben:

- Identität: das `subjectIdentity`-Element,
- Typ: beschrieben durch die `instanceOf`-Elemente,
- Name: das `baseName`-Element,
- Vorkommen (Ressource): das `occurrence`-Element.

„Zunächst bezeichnet eine Serie von `instanceOf`-Elementen [...] jene Topics, die den Typ des zu beschreibenden Topics charakterisieren.“[99] Hierfür werden die Referenzmechanismen `topicRef` oder `subjectIndicatorRef` genutzt; das `instanceOf`-Element kann ebenfalls als Sub-Element von `occurrence`, `association` und `member` genutzt werden. Durch die Typisierung von Topics werden Taxonomien in XTM formulierbar.

Das `subjectIdentity`-Element „spezifiziert das Subjekt der Realwelt, das ein Topic repräsentiert.“[100] Falls dieses Subjekt adressierbar ist, bspw. könnte es ein Dokument im Intranet einer Organisation sein, so nutzt man das `resourceRef`-Element. Ist das Subjekt nicht adressierbar, handelt es sich also zum Beispiel um eine Person, einen Gegenstand oder ein Land, so können die Elemente `topicRef` und `subjectIndicatorRef`, mehrfach und optional, zur Beschreibung herangezogen werden.

„Die Namen von Topics werden durch beliebig viele `baseName`-Elemente dargestellt.“[101] Diese enthalten das `baseNameString`-Element, das den Namen als Zeichenkette enthält, ein optionales `scope`-Element, welches den Gültigkeitsbereich des Namens angibt und beliebig viele `variant`-Elemente, welche die Konzepte des *Sort* und *Display Names* für XTM umsetzen, jedoch flexibel weitere Darstellungsarten ergänzen können. `scope`-Elemente sind auch als Kind-Elemente der `association`- und `occurrence`-Elemente einsetzbar.

---

[99] Widhalm/Mück (2002), S. 372.
[100] Widhalm/Mück (2002), S. 373.
[101] Widhalm/Mück (2002), S. 374.

Durch das `occurrence`-Element wird das gleichnamige Konzept in XTM adaptiert (vgl. Tabelle 4). Occurrences können typisiert (`instanceOf`-Element) und einem Gültigkeitsbereich (`scope`-Element) zugeordnet werden. Weiterhin kann für eine Occurrence ein Name in Form eines `baseName`-Elements vergeben werden. Eine Referenz mittels des `resourceRef`-Elements oder eine Zeichenfolge innerhalb des `resourceData`-Elements sind ebenfalls vorgesehen.

2.8.5 Assoziationen in XTM

Assoziationen, die Verknüpfungen zwischen Topics realisieren, werden durch das Element `association` beschrieben. Dies kann eine ID als Attribut enthalten, wird optional durch genau ein `instanceOf`-Kindelement typisiert und ebenfalls optional durch genau ein `scope`-Element einem Gültigkeitsbereich zugeordnet.

Die Teilnehmer werden durch das `member`-Element definiert. Dies enthält eine Referenz auf ein teilnehmendes Topic, welches durch das `topicRef`- oder das `subjectIndicatorRef`-Element referenziert wird. Weiterhin kann den Teilnehmern eine Rolle in der Assoziation zugewiesen werden. Diese, spezifiziert durch das `roleSpec`-Element, muss wiederum als Topic definiert sein oder als eindeutige Web-Ressource vorliegen.

2.8.6 Das mergeMap-Element und Public Subject Indicators

Beim `mergeMap`-Element „geht es darum, ein durch ein `href`-Attribut verlinktes Topic Map-Dokument mit jener Topic Map, in der sich das `mergeMap`-Element befindet, zu verschmelzen.“[102] In besonderem Maße ist dies notwendig, um als Topic Map Templates (Vorlagen) definierte und in XTM kodierte Wissensbasen einzubinden, so dass auf vordefinierte elementare Topics zurückgegriffen werden kann. Zur Verknüpfung wird wiederum ein *XLink* verwendet.

Angemerkt werden muss an dieser Stelle, dass, sollen nur einzelne Topics aus einer Vorlage verwendet werden, der Einsatz des `mergeMap`-Elements nicht notwendig ist. Vielmehr kann innerhalb eines `topicRef`-Elements direkt per URL auf dieses Topic zugegriffen werden, wie auf die *Public Subject Indicator* genannte Sammlung von Topics der Basis-Definition[103]. Mittels eines Sektionsaufrufs wird bspw. direkt auf die Basis-Definition einer Assoziation zugegriffen werden, die nötige Referenz lautet:

[102] Widhalm/Mück (2002), S. 378.
[103] http://www.topicmaps.org/xtm/core.xtm, aufgerufen am 19. August 2008.

```
<topicRef
xlink:href="http://www.topicmaps.org/xtm/core.xtm#association">
```

Nachdem nun alle Grundlagen zum Verständnis der Web-Technologien und semantischer Netze erläutert wurden, werden im folgenden Kapitel die Gründe zur Entwicklung einer auf AJAX-Techniken beruhenden, dynamischen Web-Applikation zur Darstellung von XML Topic Maps erläutert und die Applikation und deren Verwendung eingehend beschrieben.

3 Browsergestützte Visualisierung von XML Topic Maps

Im Folgenden wird die Entwicklung einer Applikation zur Darstellung von XML Topic Maps anhand eines Prozesses beschrieben.

Ein Software-Entwicklungsprozess besteht aus den Phasen Problemdefinition, Anforderungsanalyse, Spezifikation, Entwurf, Implementierung, Erprobung und Einführung.[104]

Die einzelnen Schritte werden, in verschiedener Ausführlichkeit, in diesem Kapitel beschrieben. Allerdings werden nur die ersten vier Phasen als solche behandelt, in der Folge wird auf die einzelnen Komponenten der Anwendung konkret Bezug genommen. Die Implementierung wird nicht vollständig erläutert, an dieser Stelle sei auf den Quellcode der Applikation verwiesen. Auf eine Dokumentation der Einführung wird verzichtet, da es sich um eine rein wissenschaftliche Arbeit handelt, die eine Art Prototyp darstellt.

Letztlich wird am Ende dieses Kapitels ein Anwendungsbeispiel des entwickelten Programms gegeben.

3.1 Problemdefinition

Zu Beginn einer Software-Entwicklung muss geklärt werden, ob die Entwicklung wirklich notwendig ist, das heißt, ob es nicht bereits ein Produkt gibt, welches die Anforderungen gänzlich umsetzt. Dazu ist es notwendig, die Anforderungen an das zu entwickelnde Produkt zu formulieren. Diese Anforderungen werden dann mit den Funktionalitäten der am Markt verfügbaren Applikationen verglichen und so festgestellt, ob die Entwicklung sinnvoll ist.

3.1.1 Anforderungen

Die „Anforderungen [...] an die Entwicklung von Software-Produkten [lassen sich] in funktionale [...], qualitative [...], systembezogene [...] und prozessbezogene Anforderungen“[105] unterteilen.

Funktionale Anforderungen beschreiben die Arbeitsweise und die problembezogene Funktionalität der zu entwickelnden Software. Dabei wird auch auf Schnittstellen und die zu verarbeitenden Daten Bezug genommen.

---

[104] Vgl. Dumke (2001).
[105] Dumke (2001), S. 24.

Die Anwendung soll:

1. Topic Maps, kodiert im XTM-Standard, dynamisch verarbeiten können,
2. Eine übersichtliche, graphische Darstellung der Topic Map, abhängig von einem Suchbegriff, erzeugen,
3. Dokumente, die im XTM-Dokument referenziert werden, verfügbar machen,
4. Datenquellen, die XTM-Dokumente generieren, einbinden können.

Qualitative Anforderungen beziehen sich auf die Produkt-Qualität der Software. Diese beziehen sich, abgeleitet aus dem ISO-Standard 9126 für Software-Produkte, auf die Kriterien Zuverlässigkeit, Benutzbarkeit, Effizienz, Wartbarkeit und Übertragbarkeit[106]. Qualitative Anforderungen an die Anwendung lauten:

5. Die Software soll intuitiv zu bedienen sein und eine einfache Nutzeroberfläche bieten.
6. Die Anwendung soll stabil und fehlertolerant sein.
7. Auf Suchanfragen soll die Anwendung in einem vertretbaren Zeitraum antworten.
8. Die Anwendung soll plattformunabhängig sein.
9. Die Software soll ohne Installation ablauffähig sein, das heißt auch keine Komponenten benötigen, die gesondert installiert werden müssen.

Die *systembezogenen Anforderungen* nehmen Bezug auf die konkrete Plattform und Programmiersprache der Entwicklung. Letztere wird an dieser Stelle noch nicht ausformuliert, da dies den Vergleich der bereits existierenden Software stark beeinflussen kann. Allgemein wird aber gefordert:

10. Die Anwendung soll mittels der Client-Server-Architektur umgesetzt werden.
11. Die Visualisierung soll im Web-Browser erfolgen.
12. Alle gängigen Web-Browser sollen die Anwendung ausführen können, ohne zusätzliche Plugins zu benötigen.

[106] Vgl. Dumke (2001), S. 26.

13. Die Serverkomponente soll möglichst plattformunabhängig und austauschbar sein.

Letztlich werden die *prozessbezogenen Anforderungen* formuliert. Diese beziehen sich auf das Projektmanagement und beinhalten die Aufwände und Zeitrahmen, die für die Entwicklung vorgesehen sind. Diese Anforderungen werden im Vergleich mit existierenden Anwendungen in diesem Fall ignoriert. Folgende Anforderungen lassen sich formulieren:

14. Die Software soll innerhalb der Bearbeitungszeit für eine Diplomarbeit, die 5 Monaten entspricht, entwickelt werden.

15. Die Anwendung soll von einem Entwickler, dem Verfasser der Arbeit, erstellt werden.

3.1.2 Vergleich mit verfügbaren Applikationen

Die bekannteste Applikation zur Darstellung von Topic Maps ist der *Ontopia Omnigator*[107], der derzeit in Version 3.3.0 vorliegt. Dieser kann nicht nur XTM-Dokumente anzeigen, sondern unterstützt viele weitere Formate wie HyTime, Linear Topic Map Notation (eine von Ontopia entwickelte Beschreibungssprache für Topic Maps) oder auch RDF. Die Funktionsweise des *Omnigator* wird in Abbildung 3-1 beschrieben.

Wie ersichtlich, werden viele Forderungen, die an die zu entwickelnde Applikation gestellt werden, bereits erfüllt. So erzeugt das Programm dynamisch HTML-Seiten, die im Web-Browser angezeigt werden können. Weiterhin wird die Client-Server-Architektur unterstützt; die serverseitigen Programme sind als Java Server Pages implementiert, die in einem Java Servlet Container ablaufen.

Auch eine graphische Visualisierung ist im *Omnigator* vorhanden. Dies wird von dem Java Applet *Vizigator* durchgeführt. Die Visualisierung soll der in Abbildung 3-2 dargestellte Screenshot verdeutlichen.

[107] http://www.ontopia.net/omnigator, aufgerufen am 21. August 2008.

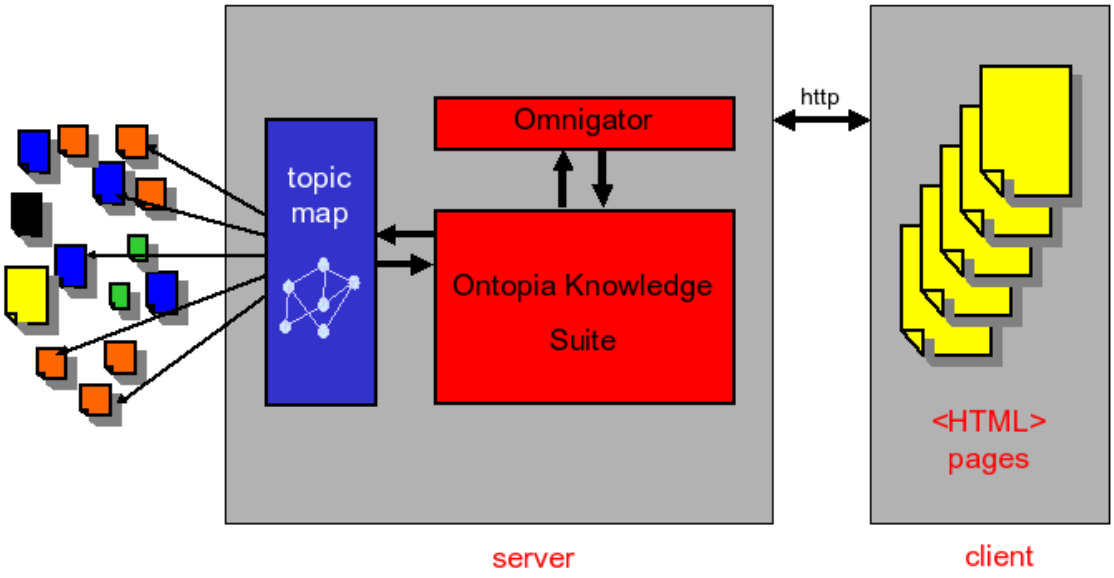


Abbildung 3-1: Ontopia Omnigator Architektur (Ontopia (2008))

Somit kann die Anwendung fast alle gesetzten Forderungen erfüllen. Jedoch verstößt sie im entscheidenden, die graphische Visualisierung betreffenden, Punkt gegen die Forderung, keine Zusatzsoftware oder Plugins zu benötigen. Der *Ontopia Vizigator* benötigt das Java Runtime Plugin zur Ausführung. Da dieses nicht zum Standardumfang von Client-Betriebssystemen gehört, ist an dieser Stelle eine Installation notwendig.

Der *Ontopia Omnigator* ist eine mächtige Plattform, die selbst das Verschmelzen von Topic Maps mit RDF beherrscht[108], dies macht sie jedoch relativ umfangreich und in der Bedienung kompliziert. Zahlreiche Plugins bieten viel Funktionalität, jedoch verstößt die Benutzerschnittstelle, aufgebaut aus Java Server Pages, Java Applets und zahlreichen Anpassungsmöglichkeiten, gegen die Forderung nach einer einfach zu bedienenden Nutzeroberfläche. Somit kann der *Ontopia Omnigator* nicht die Funktionen bieten, die die Anforderungen vorgeben.

Eine weitere Anwendung ist der *Topic Map Designer*[109] von Roland Heckel, ein Programm, welches im Rahmen einer Diplomarbeit entstanden ist und die Möglichkeit bietet, Topic Maps zu erstellen, zu editieren und anzuzeigen. Leider ist dieses Programm nur unter Microsoft Windows ablauffähig, dies widerspricht den Forderungen nach Plattformunabhängigkeit und Installationsfreiheit.

[108] Vgl. Ontopia (2008).
[109] http://www.topicmap-design.com/en/topicmap-designer.htm, aufgerufen am 21. August 2008.

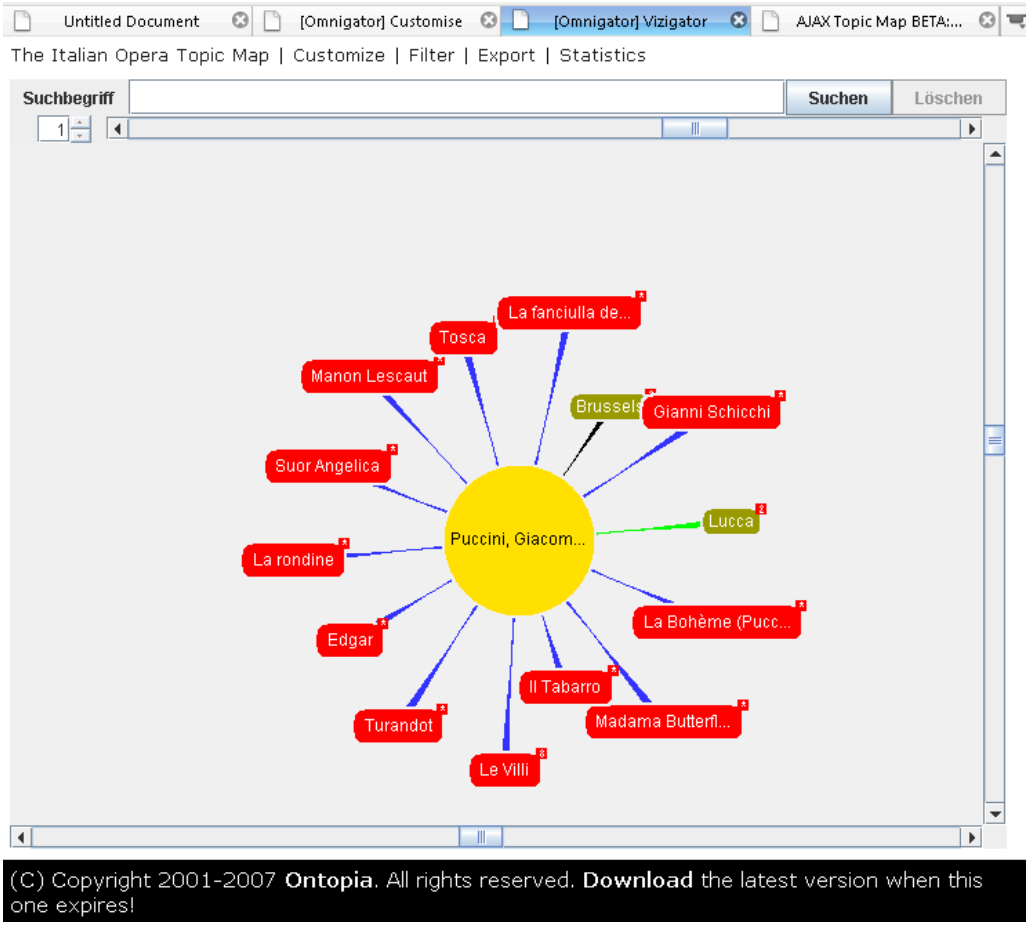


Abbildung 3-2: Screenshot Ontopia Vizigator

Zusammenfassend kann festgestellt werden, das derzeit keine Anwendung existiert, die alle gestellten Anforderungen erfüllt und somit eine Neuentwicklung sinnvoll ist.

3.1.3 Anforderungsanalyse

„Die Anforderungsanalyse [...] ist die Phase der Kontrolle von Anforderungen an ein zu entwickelndes Software-System hinsichtlich Korrektheit [...], Vollständigkeit [...], Sachgerechtheit [...], Konsistenz [...] und Machbarkeit“[110].

Anforderungen werden als *korrekt* bezeichnet, wenn sowohl der Auftraggeber, als auch der Auftragnehmer, diese als „zutreffend und richtig erkannt“[111] haben. Dies wurde im Rahmen dieser Arbeit durchgeführt, die Anforderungen sind also korrekt.

Die Prüfung auf *Vollständigkeit* umfasst die „den Nutzer betreffenden Funktionalitäten“[111], die *externe Vollständigkeit*, und die „das System selbst

[110] Vgl. Dumke (2001), S. 33.
[111] Vgl. Dumke (2001), S. 32.

betreffende“[111] *interne Vollständigkeit*. In diesem Zuge lassen sich weitere Forderungen formulieren:

16. Die Anwendung soll es bei der Darstellung ermöglichen, nur bestimmte Assoziationen einzublenden, da dies die Übersichtlichkeit erhöht.

17. Die Anwendung soll eine Funktionalität bieten die Suchanfragen der Vergangenheit (History), ohne erneuten Kontakt zum Server, wiederherzustellen.

18. Die Anwendung soll den Benutzer bei der Suche innerhalb von Datenquellen unterstützen.

19. Die Anwendung soll die folgenden Details der Topic Map darstellen können: Occurrences, Typisierungen, Assoziationen und Topics. Scopes und Verschmelzungen von Topic Maps können zu einem späteren Zeitpunkt hinzugefügt werden.

20. Das Navigieren innerhalb einer Topic Map soll benutzerfreundlich und einfach sein.

21. Die Darstellung von XTM-Dokumenten soll in Bezug auf die dargestellten Details vollständig sein.

22. Die vom Server gehaltenen Daten sollen nach einer gewissen Zeit verfallen, um Speicherplatz freizugeben und die Anwendung nicht zu verlangsamen.

Die Prüfung auf *Sachgerechtheit*, *Konsistenz* und *Machbarkeit*[112] soll an dieser Stelle ausgespart werden.

Das Ergebnis der Anforderungsanalyse ist der geprüfte Anforderungskatalog, welcher komplett im Anhang dieser Arbeit zu finden ist. Dieser nutzt aus Gründen der Übersichtlichkeit eine geänderte Nummerierung.

3.2 Spezifikation der Anwendung

Die Spezifikation befasst sich mit der „Umsetzung der Anforderungen in ein Modell, welches die künftige Funktionalität des zu entwickelnden Produktes vollständig beschreibt und damit auch die spätere softwareseitige Realisierung gewährleistet.“[113]

[112] Details zu den eingesetzten Technologien sind in Kapitel 2 beschrieben.
[113] Dumke (2001), S. 37.

Dabei werden die systembezogenen Forderungen, die die konkrete Hardware- und Softwarearchitektur spezifizieren, generell noch nicht eingebunden. Da die systembezogenen Forderungen hier jedoch recht allgemein formuliert wurden und noch nicht auf konkrete Programmiersprachen, etc. verweisen, werden die Merkmale in der Modellierung in Abbildung 3-3 berücksichtigt.

Die folgende Abbildung beschreibt vereinfacht die Suchunterstützung, den Abruf von Daten einer Quelle, das Senden des XTM-Dokumentes zum Client und die darauf folgende Visualisierung als UML-Sequenzdiagramm.

Es ist zu beachten, dass die Suchunterstützung und die Suchanfrage jeweils als asynchrone Anfragen formuliert sind, das bedeutet, dass während der Verarbeitung im Server die Anwendung ständig auf Benutzeranfragen reagieren kann.

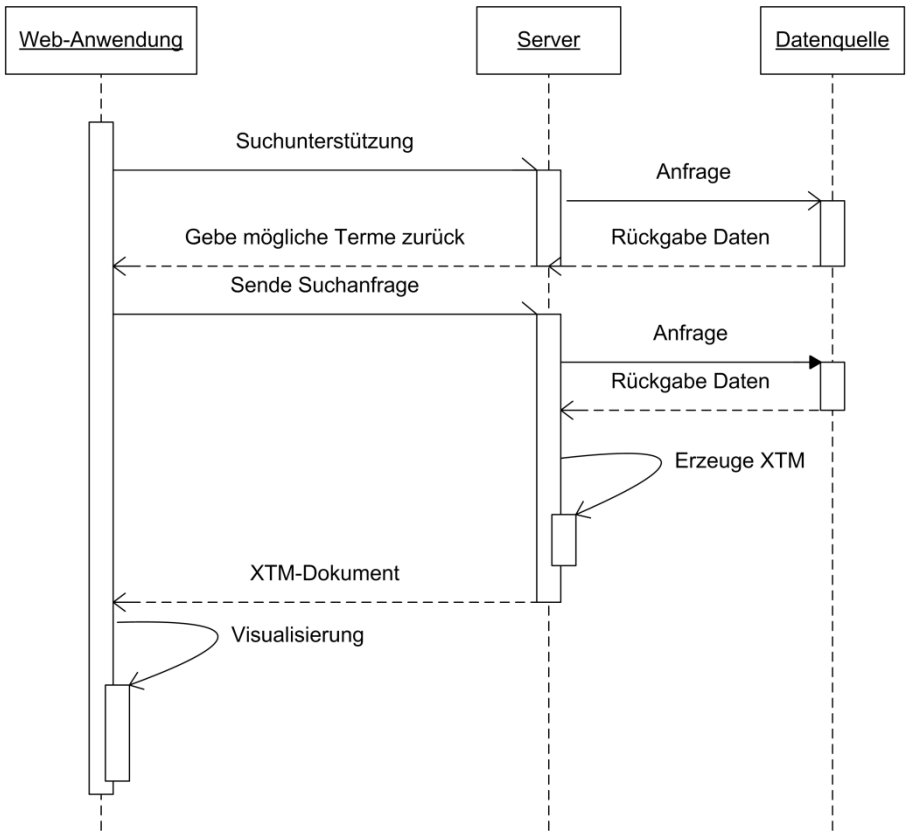


Abbildung 3-3: Sequenzdiagramm (vereinfacht)

Die Suchunterstützung erfolgt dynamisch bei der Eingabe einer Zeichenkette und kann auch wiederholt ausgeführt werden. Wurde die Suchanfrage abgesendet, so werden Daten von einer Datenquelle abgerufen. Dies geschieht durch 1 bis n Datenabrufe.

Das Vorhalten von alten Suchergebnissen, das Navigieren auf Basis einer bereits erzeugten Visualisierung, das Ausblenden von Assoziationen und das Hochladen (Upload) von XTM-Dokumenten sind nicht Bestandteil von Abbildung 3-3, die Funktionalität dieser Bestandteile der Anwendung wird jedoch in den folgenden Use-Case-Diagrammen deutlich.

Das erste Use-Case-Diagramm beschreibt den Anwendungsfall der Suchunterstützung durch das System.

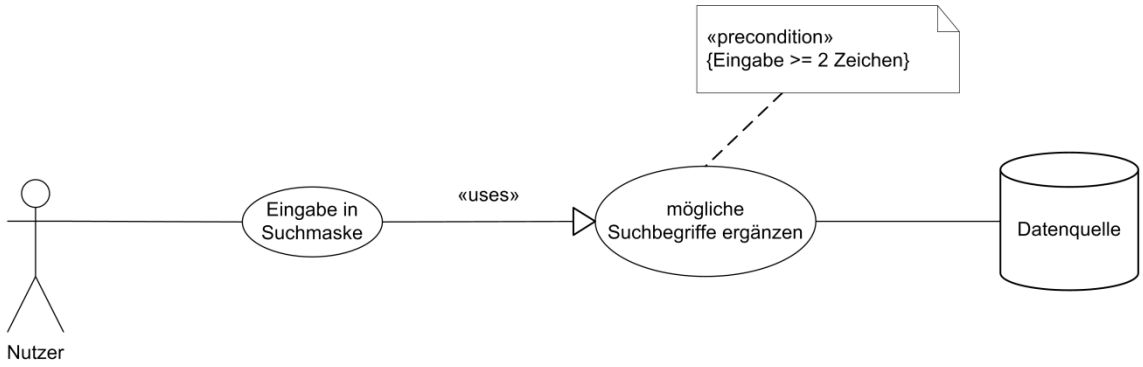


Abbildung 3-4: Anwendungsfall Suchunterstützung

Nachdem der Benutzer nun den gewünschten Suchbegriff spezifiziert hat, sendet er eine Anfrage nach diesem Begriff ab und erhält von der Anwendung eine Visualisierung der Suchbegriffs und der mit ihm verknüpften Begriffe. Diese werden in einer XTM-Datei, die dynamisch mit Hilfe der Datenquelle erzeugt wird, beschrieben und nach Erzeugung des Dokumentes entsprechend dargestellt (vgl. Abbildung 3-5). Weiterhin wird das Ergebnis der Suchanfrage abgespeichert, um ein komfortables Anzeigen der vergangenen Suchergebnisse zu ermöglichen.

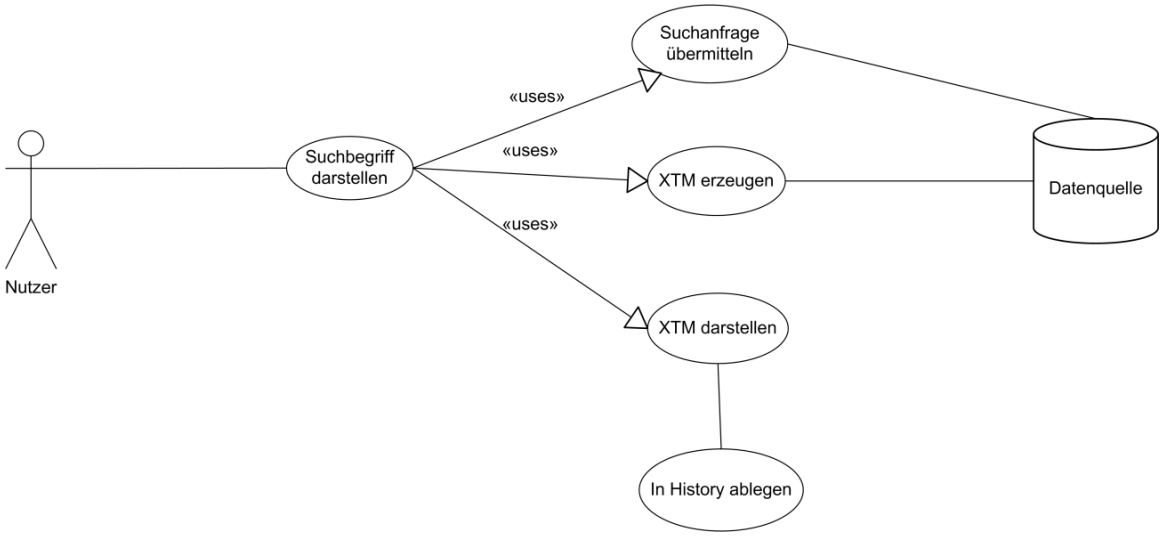


Abbildung 3-5: Anwendungsfall Suchbegriff darstellen (vgl. Abbildung 3-3)

Weiterhin soll es möglich sein, aus einer Darstellung heraus einen verknüpften Begriff als Suchterm zu übermitteln, ohne diesen erneut eingeben zu müssen. Dieser Anwendungsfall wird in Abbildung 3-6 dargestellt. Es sei zu beachten, dass der Anwendungsfall ‚Suchbegriff darstellen' in Abbildung 3-5 ausführlich beschrieben ist.

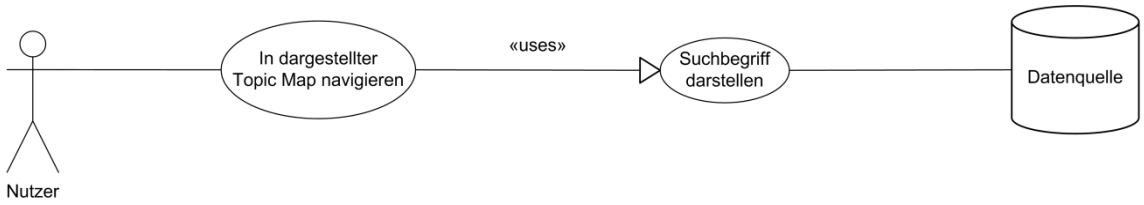


Abbildung 3-6: Anwendungsfall Navigation

Zum Aufruf der gespeicherten Ergebnisse in der History soll das folgende Diagramm dienen:

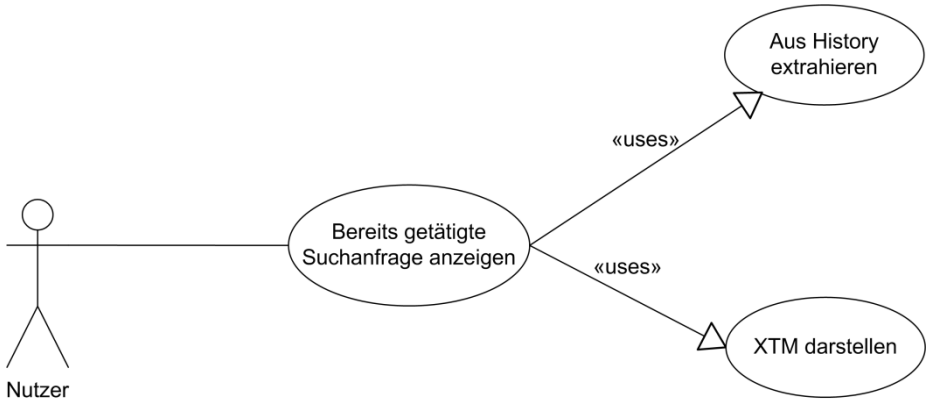


Abbildung 3-7: Anwendungsfall History abrufen

Es wurde gefordert, dass Assoziationen ausgeblendet werden können, um die Übersichtlichkeit bei der Darstellung der Topic Map zu erhöhen. Diese Anforderung soll so realisiert werden, dass entweder eine oder alle der in der Topic Map vorhandenen Assoziationen dargestellt werden. Wird also eine Assoziation ausgewählt, die angezeigt werden soll, so werden alle Topics, die nicht über diese Assoziation mit dem Suchbegriff verknüpft sind, nicht angezeigt.

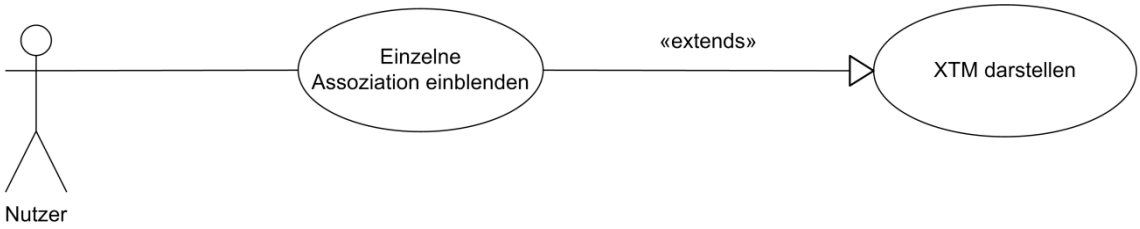


Abbildung 3-8: Anwendungsfall Assoziation einblenden

Schließlich bleibt noch das Hochladen (Upload) von XTM-Dokumenten, um in diesen zu navigieren. Dies verdeutlicht der letzte Anwendungsfall. Die Datenquelle ist in diesem Fall eine Datensenke, die Bezeichnung wird aber zur Wahrung der Konsistenz mit den vorangehenden Modellen nicht verändert.

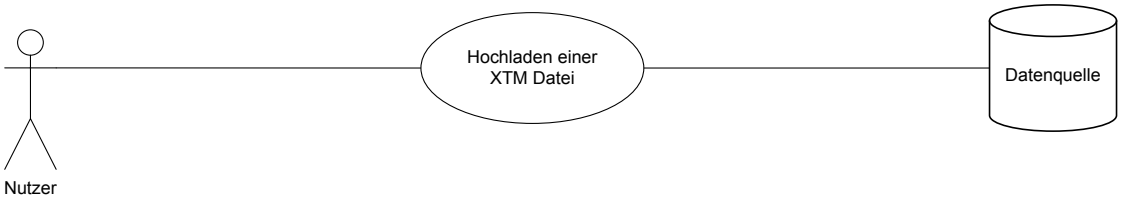


Abbildung 3-9: Anwendungsfall Upload

Zur Spezifikation gehören weiterhin ein Testkonzept, ein Abnahmekonzept, der erste Entwurf einer Anwendungsdokumentation und ein „Einsatzkonzept, welches den Inhalt und die Prioritäten für die Beschaffung der Einsatzmittel (Hard- und Software) und die Planung der erforderlichen Nutzerschulungen bestimmt und vorgibt.“[114]

Da andere Tools, wie bspw. der *Ontopia Vizigator*, zu Vergleichszwecken bzgl. der Vollständigkeit der Darstellung und der Übersichtlichkeit eingesetzt werden können, beschränkt sich das *Testkonzept* auf den Vergleich der Darstellung von Suchbegriffen zwischen diesen Tools und der zu entwickelnden Anwendung. Weiterhin werden die gestellten Anforderungen auf Erfüllung geprüft.

Die *Abnahme* erfolgt mit der Verteidigung dieser Arbeit.

Da das erzeugte Tool sehr einfach zu bedienen sein soll, wird im Rahmen der Spezifikation kein Entwurf einer *Anwenderdokumentation* durchgeführt. Ein Leitfaden zur Bedienung der Applikation ist in Abschnitt 3.6 zu finden.

Die eingesetzte Hard- und Software, also die *Einsatzmittel*, beschränken sich beim Betreiber der Anwendung auf einen Webserver, den Servlet Container *Apache Tomcat* und eine *MySQL*-Datenbank.

Damit ist die Spezifikation abgeschlossen und es kann mit dem Entwurf der Anwendung begonnen werden.

3.3 Entwurf der Anwendung

„Der ***Software-Entwurf*** (software design) ist die Umsetzung der in der Spezifikation erstellten Modelle in eine die hard- und softwarebezogenen Anforderungen berücksichtigende Form von Diagrammen, Schemata und Pseudocodes, die gleichzeitig

[114] Dumke (2001), S. 47.

die hierbei gültigen qualitativen Anforderungen mit einschließen und unmittelbar als Implementationsvorgabe verwendet werden kann.“[115]

In diesem Kapitel sollen vor allem architekturspezifische Details thematisiert werden. Die einzelnen Komponenten sind in den Abschnitten 3.4 und 3.5 erläutert. Dabei werden die Phasen der Softwareentwicklung verlassen und sowohl der Entwurf als auch die Implementierung thematisiert.

Tabelle 5: Systemspezifische Forderungen und Technologieentscheidungen

Forderung(en)	Bedeutung / Technologieentscheidung
Asynchrone Kommunikation / keine clientseitige Installation	Programm zur Darstellung muss auf allen Client-Systemen vorhanden sein: **Web-Browser**. Asynchrone Kommunikation mittels Web-Browser: **AJAX-Technologie**.
Serverkomponente soll plattformunabhängig und austauschbar sein	Als plattformunabhängige Technologie wird **Java** ausgewählt. **Java Servlets** sind serverseitige Programme und austauschbar zwischen Servlet Containern durch standardisierte Servlet-API.
Die Anwendung soll auf Suchanfragen in einem vertretbaren Zeitraum antworten.	Problematisch bei großen XTM-Dokumenten, da das clientseitige Parsen per DOM geschieht. Ausweg: Übertragen der XTM-Dokumente in eine **MySQL**-Datenbank und ‚Ausschneiden‘ der relevanten Objekte zur Suchanfrage. Die entstandene kleinere XTM-Datei kann clientseitig unproblematisch und schnell visualisiert werden.

Wie in den systemspezifischen Anforderungen festgehalten, soll die Anwendung mittels der Client-Server-Architektur umgesetzt werden. Weiterhin ist in Abbildung 3-3 eine asynchrone Kommunikation zwischen Client und Server gefordert. Letztlich wird verlangt, dass zur Ausführung der Applikation clientseitig keine Installation nötig sein soll. Tabelle 5 fasst die Forderungen und die daraus folgenden

[115] Dumke (2001), S. 61.

Technologieentscheidungen zusammen. Diese sind nicht willkürlich ausgewählt und als Antwort auf die gestellten Forderungen zu verstehen. So kann eine asynchrone Kommunikation mittels Web-Browser nur durch die AJAX-Technologie erfolgen. Weiterhin sind die Plattformunabhängigkeit und die Nutzung des Java-Archivs *XTMGenerator* entscheidende Argumente für den Einsatz der J2EE-Technologie auf der Serverseite.

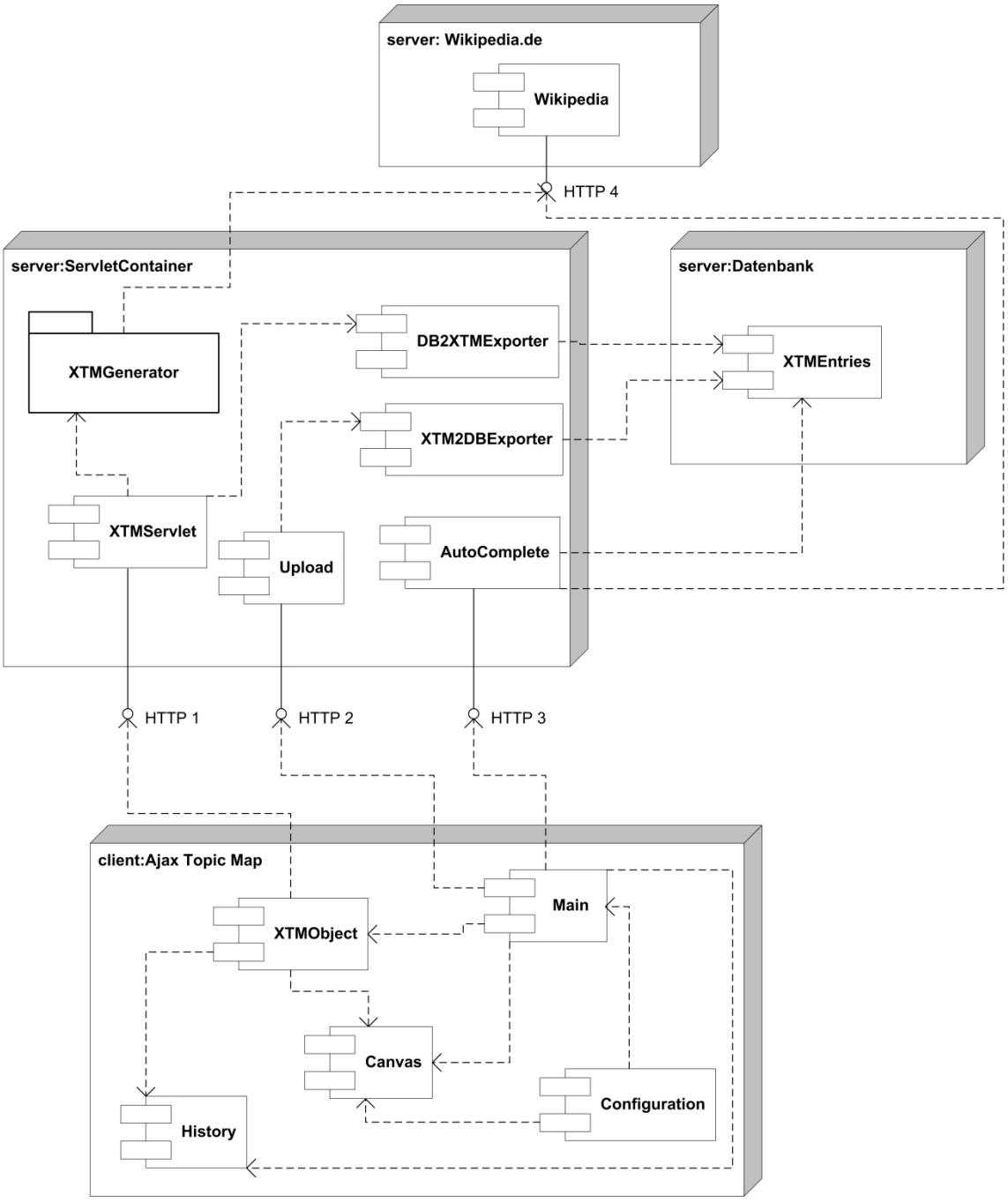


Abbildung 3-10: Komponentendiagramm der Applikation

Die Wahl des Datenbanksystems fiel auf *MySQL*, da dieses weit verbreitet und lizenzkostenfrei ist. Andere Datenbanksysteme können aber, da die standardisierten Technologien *JDBC* und *SQL* eingesetzt werden, leicht integriert werden.

Nachdem diese Entscheidungen getroffen wurden, kann ein Komponentendiagramm erstellt werden, das die Bestandteile der Anwendung und die Schnittstellen zwischen ihnen beschreibt. In Abbildung 3-10 ist die komplette Applikation, abzüglich einiger Hilfsklassen, die später beschrieben werden, dargestellt.

Es ist anzumerken, dass die Auslieferung der clientseitigen Komponenten per Web-Server erfolgt. Dies ist im Diagramm ebenfalls nicht enthalten; es beschreibt ausschließlich die Ausführung der Anwendung.

Die Datenquellen sind in der Darstellung als weitere Server verzeichnet. Die Web-Ressource *Wikipedia*, deren Artikel durch das Java-Archiv *XTMGenerator*[116] in XTM-Dokumente umgewandelt werden, und das Datenbanksystem, welches von Benutzern hochgeladene XTM-Dateien in Form einer relationalen Datenbank enthält, sind im Komponentendiagramm vorhanden.

Im Servlet Container wird zwischen zwei Arten von Komponenten unterschieden. Alle Komponenten, die eine HTTP-Schnittstelle enthalten, werden in Form von Java Servlets implementiert. Alle weiteren Komponenten innerhalb des Containers sind Java-Klassen, die von den Servlets zur Beantwortung der Requests eingebunden werden.

Die Ajax Topic Map Applikation, welche die Funktionalität für den Client bündelt, enthält JavaScript-Objekte. Diese werden durch den Browser ausgeführt, wenn bestimmte Ereignisse eintreten, können sich aber auch gegenseitig aufrufen. So wird eine Methode in der `Main`-Komponente bspw. gestartet, wenn die Website komplett geladen wurde, also die Client-Komponenten komplett vom Web-Server übermittelt wurden. Dieses ruft dann das `Configuration`-Objekt auf, welches die Konfiguration des Clients einliest und diese dem `Main`-Objekt zur Verfügung stellt.

Eine ausführliche Beschreibung der clientseitigen Applikation, ihren Komponenten und den eingesetzten Algorithmen und Datenstrukturen folgt im nächsten Abschnitt.

[116] Vgl. Twele (2008).

3.4 Entwurf und Implementierung des Client

Der Client wird, da es sich um eine Web-Applikation handelt, komplett im Browser ausgeführt. Die Ausführung beginnt mit dem Aufruf eines URL, der dem damit verknüpften Web-Server mitteilt, welche Dokumente und Komponenten übertragen werden sollen. Er verweist auf ein XHTML-Dokument, das typischerweise `index.html` oder `index.htm` heißt. Mit diesem Dokument, hier Basis-Website genannt, sind alle weiteren, zum Start der Applikation benötigten, Komponenten wiederum per URL verknüpft.

3.4.1 Basis-Website

Mit dem Empfang der Basis-Website beginnt die clientseitige Verarbeitung. Die Website ist ein XHTML-Dokument, somit besteht sie aus der für XML typischen Struktur aus *Prolog* und *Dokumenteninstanz*. Der Prolog lautet wie folgt und weist das Dokument als XHTML Transitional[117] aus:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html
     PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
     "DTD/xhtml1-transitional.dtd">
```

Die geladene DTD gibt die in der Dokumenteninstanz enthaltenen Elemente an. Das Wurzelelement lautet `<html>` und enthält die Elemente `<head>` und `<body>`.

Im `<head>` Element werden Metainformationen zum Dokument, der Titel der Website und Komponenten, die zur Ausführung benötigt werden, spezifiziert. Dabei handelt es sich um die Formatierungsinformationen, welche als CSS-Dokument verknüpft sind, die JavaScript- bzw. AJAX-Frameworks und die JavaScript-Dateien, welche den Code der Komponenten, wie sie in Abbildung 3-10 beschrieben sind, enthalten. Hierbei ist es wichtig, die Reihenfolge der Verknüpfungen zu beachten. Nutzt also ein Objekt, welches in einem Dokument beschrieben ist, Objekte in einer anderen JavaScript-Datei, so muss dieses bereits bekannt, also bereits geladen sein. Dies bezieht sich insbesondere auf Objekte aus Frameworks, die teilweise aufeinander aufbauend sind.

[117] Vgl. W3C (2008a).

Der Abschnitt in der Basis-Website lautet wie folgt:

```
<head>
      <meta content="text/html; charset=UTF-8" http-
      equiv="content-type"/>
      <title>AJAX Topic Map BETA</title>
      <script src="lib/prototype/prototype.js"
      type="text/javascript"></script>
      [...]
      <link rel="stylesheet" type="text/css" href="style.css"/>
</head>
```

Aus dem Code geht hervor, dass zuerst das AJAX-Framework `prototype.js` geladen wird. Danach werden die darauf aufbauenden und zusätzliche Funktionalitäten liefernden `scriptaculous.js` und `control.tabs.2.1.1.js`, eingebunden. Dies, wie auch das Laden der clientseitigen Komponenten, wie im nächsten Absatz beschrieben, ist im Beispiel ausgespart.

Die Dateien `xtm_spec.js`, die Objekte enthält, um XTM-Elemente im Speicher als JavaScript-Objekte halten zu können, und `visualizer.js`, welche die Canvas-Komponente (aus dem Englischen: canvas = Leinwand) und unterstützende Objekte enthält und sich um die eigentliche Visualisierung kümmert, werden dabei zuerst geladen. In der Folge wird der Parser, der die XTMObject-Komponente enthält und neben dem Einlesen von XTM-Dokumenten auch die Canvas-Komponente steuert, verknüpft. Die zum Auslesen der Konfiguration und zum Halten der vergangenen Abfragen benötigten Codeabschnitte, in den Dateien `config.js` und `history.js`, und die Datei `main.js` werden als letzte Bestandteile der, die Komponenten der Client-Applikation enthaltenden, JavaScript-Dateien geladen.

Am Ende wird durch das Element `link` ein Stylesheet verknüpft, das die Formatierung der Website definiert.

3.4.1.1 Strukturinformationen

Das `<body>`-Element, welches die Strukturinformationen der Basis-Website enthält, ist Thema dieses Abschnittes. Prinzipiell kann festgestellt werden, dass hier die Elemente der Website, beschrieben durch `<div>`-Tags[118], festgelegt werden und so eine Grundstruktur erstellt wird. Die Elemente habe IDs oder Klassenbezeichner in Attributform, mit deren Hilfe später per CSS die Formatierung durchgeführt wird und

[118] Vgl. W3C (2008a).

welche die DOM-Manipulation ermöglichen. Die IDs der wichtigsten so definierten Elemente sind:

- `navigation`
  Enthält eine Liste der Datenquellen der Applikation. Diese müssen als unsortierte Liste, die ebenfalls einen Bezeichner hat (hier: `nav`) hinterlegt sein.

- `search`
  Enthält pro Datenquelle ein weiteres `<div>`-Element mit Suchmasken und weiteren, auf die Datenquelle bezogenen, Funktionalitäten. Es ist zu beachten, dass lediglich eines dieser `<div>`-Elemente sichtbar sein darf, alle anderen müssen zwingend das Attribut `style="display: none;"` enthalten. Dies muss im XHTML-Dokument spezifiziert werden, da diese Information später manipuliert wird.
  Weiterhin enthalten ist das Element `choices`, welches allgemeine Funktionen wie das Aus- und Einblenden von durch Assoziationen beschriebenen Linien und ihren Bezeichnern und die Navigation in der History der Suchanfragen ermöglicht.

- `navigation-association-browser`
  Innerhalb dieses Elements werden die einzelnen Assoziationen der dargestellten Topic Map eingefügt. Diese sind dann per Mausklick einzeln einblendbar, alle anderen Assoziationen sind in der Darstellung ausgespart. Selbstverständlich ist es mit diesem Element auch möglich, wieder alle Assoziationen einzublenden. Es muss eine leere unsortierte Liste enthalten, die einen Bezeichner hat (hier: `nav2`) und später per DOM-Manipulation gefüllt wird.

- `main`
  Der Platzhalter für die durch das `Canvas`-Objekt erzeugte Darstellung von XTM-Dokumenten.

- `uploadwidget`
  Enthält die für den Datei-Upload benötigte Eingabemaske.

Angemerkt werden muss, das diese Bezeichnungen willkürlich und austauschbar sind. Beim Tausch der Bezeichner ist allerdings zu beachten, dass die Informationen per Konfigurations-Datei der Anwendung zur Verfügung gestellt werden und das Stylesheet entsprechend geändert wird.

3.4.1.2 Formatierung

Die Formatierung der Basis-Website wird mittels eines Cascading Stylesheets vorgenommen. Erst mittels dieses Dokumentes wird den einzelnen Elementen ein Bereich im Browserfenster und Informationen wie Hintergrundfarben und Rahmen zugewiesen. Auch müssen Elemente, die erst später per DOM-Manipulation eingefügt werden, hier bereits mit Stil-Informationen versorgt werden.

Der grundsätzliche Aufbau der Nutzeroberfläche ist in Abbildung 3-11 dargestellt. Dieser ist vom Betreiber der Applikation anpassbar und auch in veränderter Form benutzbar. So ist es bspw. möglich, die Positionen der Elemente zu verändern, keine Quellenauswahl zuzulassen oder die Möglichkeit der Ausblendung von Assoziationen zu entfernen. Der Platzhalter für den Assoziationsbrowser muss allerdings, um voll funktionsfähig zu sein, über dem Äquivalent des `main`-Elementes positioniert sein, da die Größe des Assoziationsbrowsers die des `Canvas`-Objekts beeinflusst.

Das Farbschema, welches in der Anwendung verwendet wird, basiert auf dem Corporate Design Handbuch der MIS-Arbeitsgruppe der Universität Magdeburg. Selbstverständlich sind die verwendeten Farben, durch eine Anpassung des Stylesheets, veränderbar.

Abbildung 3-11: Grundsätzlicher Aufbau der Nutzeroberfläche

Elemente, welche in dieser Ansicht nicht sichtbar sind, wie bspw. die Elemente mit den Identifikatoren `uploadwidget` und `navigation-association-browser`, werden entweder durch Nutzerinteraktionen erst sichtbar gemacht oder enthalten, da die Anwendung noch keine Topic Map darstellt, keine Einträge.

3.4.1.3 AJAX-Frameworks

AJAX-Frameworks erweitern den Sprachumfang von JavaScript durch Funktionen, die die Anwendungserstellung erleichtern, indem sie Browserinkompatibilitäten beseitigen und Hilfsfunktionen bereitstellen, um dem Programmierer die zeitraubende Implementierung häufig genutzter Funktionalitäten zu ersparen.

Die Erzeugung eines AJAX-Requests, einer Server-Anfrage mittels der AJAX-Technologie, ist ein Beispiel für eine Inkompatibilität, die durch die Frameworks gelöst wird. So ist eine, mit reinem JavaScript und ohne eine solche Bibliothek erzeugte, Abfrage für die Browser *Internet Explorer* und *Firefox* unterschiedlich zu initialisieren:

```
var xmlHttp = null;
//Internet Explorer pre 7.0
try {
    xmlHttp  = new ActiveXObject("Msxml2.XMLHTTP");
} catch(e) {
    try {
       xmlHttp  = new ActiveXObject("Microsoft.XMLHTTP");
    } catch(e)
       xmlHttp  = null;
    }
}

//Firefox, Opera, Safari, Internet Explorer 7...
if (typeof XMLHttpRequest != 'undefined') {
    xmlHttp = new XMLHttpRequest();
}

// jetzt gültig für alle Browser:
if (xmlHttp) {
    xmlHttp.open('GET', 'test.xml', true);
   // weitere Anweisungen zur Verarbeitung
}
```

Die gleiche Funktionalität bietet dieser mittels des *prototype.js*-Frameworks erstellte Aufruf:

```
var xmlHttp = new Ajax.Request (
         'test.xml', {
           method: 'GET',
           //weitere Anweisungen zur Verarbeitung
         }
);
```

Es wird in der Anwendung also das *prototype.js*-Framework genutzt, um AJAX-Abfragen zu initialisieren. Weiterhin bietet es für das Eventhandling, das Reagieren auf Ereignisse, wobei ähnliche Inkompatibilitäten bestehen, eine sichere Schnittstelle. Auch werden für das Erzeugen von Objekten und für die DOM-Manipulation Methoden aus dem Framework genutzt, wie sie in (Stephenson (2008)) beschrieben sind.

Aufbauend auf diesem Framework werden die Bibliotheken *script.aculo.us* und *Control.Tabs* genutzt.

Erstere bietet eine leichte Implementierung eines AJAX-Autocompleters, einer Funktionalität, die Eingabefelder regelmäßig auf Änderungen überprüft und gegebenenfalls eine AJAX-Abfrage einleitet und dem Nutzer auf Basis seiner Eingabe mögliche, in der Datenquelle vorhandene, Einträge zur Auswahl bietet. Weiterhin sind Funktionen zur Animation von DOM-Elementen und zur Implementierung einer *Drag & Drop*-Funktionalität, wie sie später beschrieben wird, vorhanden.

Control.Tabs unterstützt den Programmierer mit Funktionen zum Erzeugen von Reitern, wie in Abbildung 3-11 im Element `navigation` zu sehen. Diese blenden DOM-Elemente, abhängig vom gewählten Reiter, ein oder aus.

3.4.2 Die Komponente Main

In Anlehnung an die Java Main-Methode, die bei Java-Anwendungen standardisiert beim Programmstart aufgerufen wird, ist die Komponente, welche den Code enthält, der direkt nach dem Laden der Seite ausgeführt wird, in der JavaScript-Datei `main.js` gespeichert.

Diese enthält als Hauptbestandteil einen EventListener, eine Methode, die auf das Ereignis ‚die Seite wurde komplett geladen' reagiert. Dieses Ereignis nennt sich `load`, mittels des *prototype.js*-Frameworks wird diese Methode wie folgt implementiert:

```
Event.observe (window, 'load', function () {
        /* Anweisungen, die beim Eintritt des Ereignisses
           ausgeführt werden sollen. */
});
```

Dabei beschreibt das erste Argument das Objekt, welches überwacht werden soll, das zweite Argument das Ereignis, auf das es zu reagieren gilt und das dritte Argument die auszuführende Funktion (mit optionalen weiteren Argumenten)[119].

Dieses Ereignis markiert den Beginn der clientseitigen Verarbeitung. Mit dessen Eintritt ist die Anwendung komplett an den Benutzer übertragen und kann, nachdem sie konfiguriert ist, auf Nutzereingaben reagieren. Das Konfigurieren der Anwendung findet innerhalb der ausgeführten Funktion statt. Im Flussdiagramm in Abbildung 3-12 sind die Anweisungen, die innerhalb dieses EventListeners ausgeführt werden, beschrieben.

Nachdem die Konfiguration der Client-Anwendung eingelesen wurde, werden globale Referenzen für zwei Objekte erstellt. Eines enthält die JavaScript-Abbildung der Elemente der eingelesenen XTM-Datei und das andere deren graphische Pendants, die per DOM-Manipulation in den als `canvas-parent` definierten Abschnitt der Basis-Website eingefügt werden. Diese heißen `xtm` für das aktuelle `XTMObject` und `c` für das aktuelle `Canvas`-Objekt. `xtm` enthält zu diesem Zeitpunkt allerdings noch keine Daten, es wird mit `null` initialisiert.

Daraufhin wird das `History`-Objekt erstellt und das Tabbed Browsing mit *Control.Tabs*, wie bereits beschrieben, für das `navigation`-Element der Basis-Website aktiviert.

Ist dies geschehen, so werden die Datenquellen, wie sie in der Konfiguration angegeben werden, bearbeitet. Den Textformularen werden suchunterstützende automatische Ergänzungen auf AJAX-Basis, die Autocompleter, zugeordnet.

[119] Vgl. Stephenson (2008).

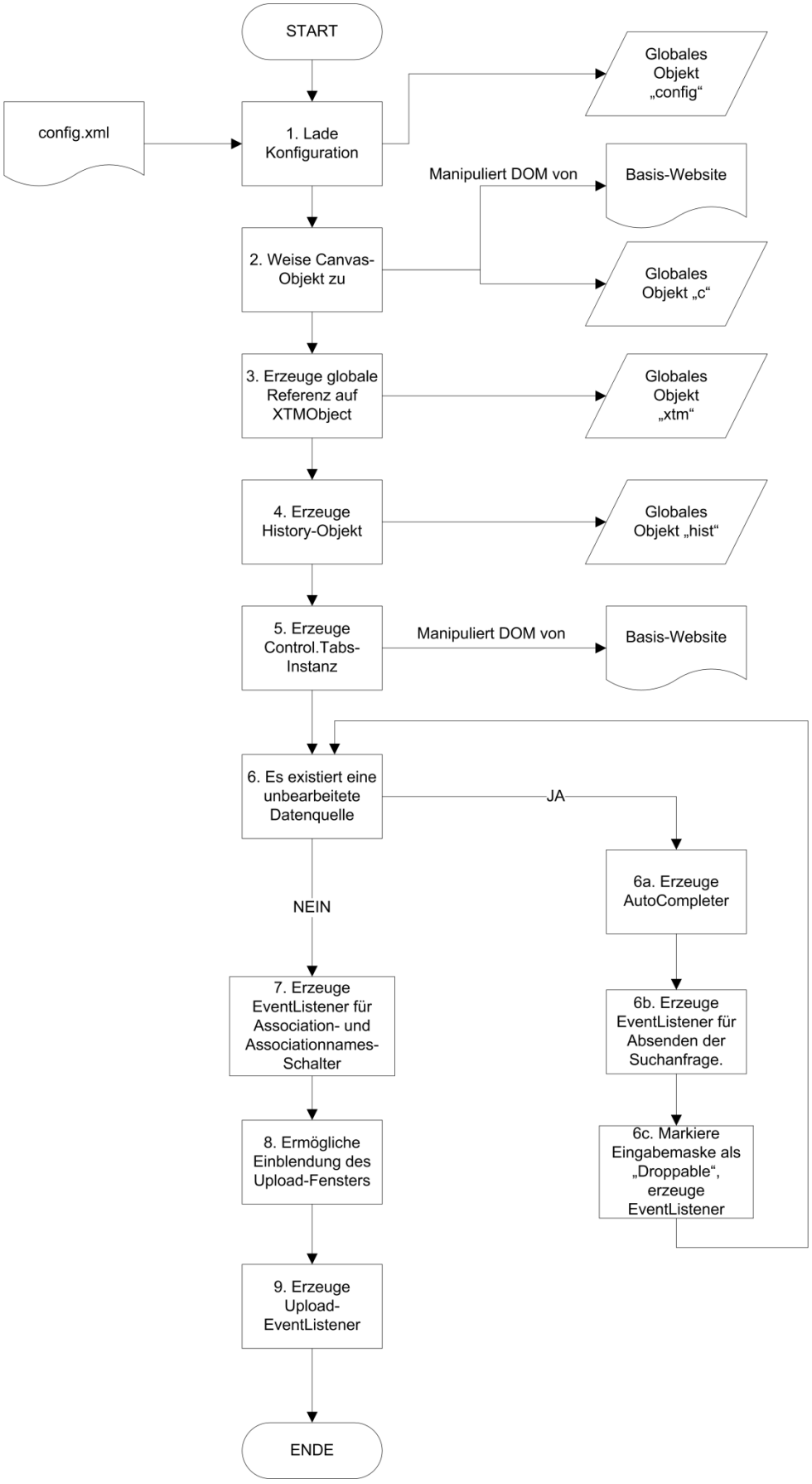


Abbildung 3-12: Flussdiagramm Ereignis "load"

Dann, in Punkt 6b des Flussdiagramms, wird die Routine, die als Reaktion auf das Ereignis ‚Suchanfrage absenden' ausgeführt wird, definiert. Hier wird das `XTMObject` der globalen Referenz `xtm` zugeordnet und dieses initialisiert. Dabei wird, wie später genauer erläutert, eine AJAX-Abfrage ausgeführt, die als Rückgabe eine XTM-Datei erhält. Diese wird im Objekt automatisch eingelesen und daraufhin die Visualisierung mittels der `c`-Referenz auf dem `Canvas`-Objekt eingeleitet. Auch wird das `XTMObject`, welches nun die Daten des XTM-Dokumentes enthält, dem `History`-Objekt hinzugefügt.

Weiterhin wird das Sucheingabefeld als „Droppable" markiert, dies macht es in diesem Fall zum Auslöser von bestimmten Ereignissen, falls Objekte mit der Maus auf dieses Feld gezogen werden.

Sind alle Datenquellen abgearbeitet, so wird den Schaltern, die in Abbildung 3-11 im Feld `search` zu sehen sind, Funktionalität verliehen (‚Show: Associations' und ‚Show: Association Names'). Dies geschieht, indem dem Ereignis des Mausklicks auf diese Schalter Verarbeitungsanweisungen zugeordnet werden, die dem `Canvas`-Objekt mitteilen, ob die Linien zwischen den dargestellten Topics angezeigt und ob diese mit dem zugehörigen Assoziations-Namen versehen werden sollen.

Letztlich wird dem Nutzer die Möglichkeit gegeben, XTM-Dokumente in eine Datenquelle hochzuladen. Dieses wird mit den Schritten 8 und 9 ermöglicht.

Die `main.js`-Datei enthält noch einen weiteren EventListener. Dieser reagiert auf das Vergrößern und Verkleinern des Anzeigefensters, das Ereignis `resize`, und gibt in diesem Fall die Neuskalierung der angezeigten Topic Map in Auftrag.

3.4.3 Konfiguration der Client-Applikation

Mittels des in der Datei `config.js` definierten `ConfigObject`, welches bei Initialisierung die Angabe einer Konfigurationsdatei, wie in Anhang B beispielhaft in dieser Arbeit zu finden, verlangt, werden der Anwendung die grundsätzlichen Informationen zur Verfügung gestellt, um die Arbeit aufzunehmen.

Die Konfiguration ist als XML-Dokument verfasst, jedoch wurde keine DTD spezifiziert. Deshalb ist es notwendig, auf Leerzeichen zu verzichten. Eine DTD kann jedoch leicht nachträglich erstellt werden. Das Dokument enthält das Wurzelelement `<config>`, untergeordnet finden sich die Abschnitte `general`, `pictures` und, beliebig oft wiederholbar, der Abschnitt `source`.

Unterhalb des <general>-Elementes werden allgemeine Informationen spezifiziert, wie der Seitentitel, die Anzahl der maximal darstellbaren Topics und die Elemente der Basis-Website, die als Träger der DOM-Elemente dienen, welche durch das Canvas-Objekt und die *Control.Tabs*-Bibliothek erzeugt und manipuliert werden. Im Element <pictures> werden die Graphikdateien festgelegt, die zur Erzeugung der Verbindungslinien zwischen dargestellten Topics oder zur Anzeige einer Lade-Aktivität dienen.

Mittels eines <source>-Abschnitts werden alle Angaben, die die Anwendung zur Abfrage von Informationen von einer Datenquelle benötigt, der Name der Datenquelle und die Elemente der Basis-Website, die zur Abfrage dieser Datenquelle erzeugt wurden, definiert. So werden die Schnittstellen für die Autocomplete-Funktionalität, der URL und ein Parameter für die eigentliche Datenabfrage vom Server spezifiziert.

Da AJAX-Abfragen aus Sicherheitsgründen nicht über Servergrenzen hinaus erfolgen dürfen, müssen die serverseitigen Programme, die durch den URL angesprochen werden, auf dem gleichen Host wie die Basis-Website liegen.[120]

Das ConfigObject selbst wird, wie mittels *prototype.js* üblich, durch den folgenden JavaScript-Code generiert:

```
var ConfigObject = Class.create ({
        initialize: function (url) {
          // bei Instantiierung ausgeführte Anweisungen
        },
        //weitere Funktionen
});
```

Erzeugt wird eine Instanz des Objektes in der oben beschriebenen main.js-Datei mittels des Aufrufs:

```
config = new ConfigObject ('config.xml');
```

Wird nun dieses neue Objekt erzeugt, so führt es die initialize-Funktion aus. Hier wird ein synchroner HTTP-Request ausgeführt, der die Daten in der Konfigurationsdatei als JavaScript-Objekte innerhalb der Instanz config speichert,

[120] Vgl. Crane, et. al (2006), S. 277ff.

sodass die Applikation bspw. den Seitentitel über das Objekt `config.pageTitle` einlesen kann. Die Anfrage muss hier synchron sein, da die folgenden Anweisungen die eingelesenen Konfigurationsdaten zur Ausführung benötigen und erst nachdem alle Einstellungen eingelesen sind ausgeführt werden dürfen.

3.4.4 Datenaustausch und Parser

Zur Datenabfrage von einer Datenquelle nutzt der Client der Applikation zwei Schnittstellen zum Server. Die erste Datenschnittstelle dient der Suchunterstützung, dem Autocomplete, die zweite der Abfrage von XTM-Dokumenten (vgl. HTTP 1 und 3 in Abbildung 3-10).

Innerhalb der Konfiguration werden diese Schnittstellen beschrieben. Bspw. wird die Schnittstelle zum Abruf von Daten einer Datenbank in der `config.xml` so definiert:

```
<source>
        <name>File-Upload</name>
        <submit-button>submit_db</submit-button>
        <input-form>searchtopic_db</input-form>
        <url-autocomplete>AutoComplete</url-autocomplete>
        <autocomplete-param>searchdb</autocomplete-param>
        <autocomplete-indicator>indicator1</autocomplete-
        indicator>
        <autocomplete-div>autocomplete_choices</autocomplete-
        div>
        <url-xtmrequest>XTMServlet</url-xtmrequest>
        <param>
                <value>DB</value>
        </param>
</source>
```

Die Quelle wird hier ‚File-Upload' genannt, da es diese Schnittstelle erlaubt, Daten, die hochgeladen wurden, zu visualisieren. Intern ist dies als der Zugriff auf die MySQL-Datenbank des Servers zu verstehen.

`<input-form>` spezifiziert die ID des DOM-Elements, in welchem die Texteingabe bei der Suche in der Datenquelle erfolgt. Dieses Feld ist Ausgangspunkt für die Suchunterstützung. An den im Element `<url-autocomplete>` angegebenen URL wird ein AJAX-Request gesendet, welcher als Parameter den Inhalt des Elements

<autocomplete-param> hat und als Wert den Inhalt des Texteingabefeldes übermittelt. Ein Ladesymbol wird im DOM-Element mit der ID indicator1 eingeblendet und das Ergebnis im DOM-Element autocomplete_choices anzeigt.

Abbildung 3-13: Beispiel der Suchunterstützung

Die Autocomplete-Funktionalität wird mit dem vordefinierten Objekt Ajax.Autocompleter aus der *script.aculo.us*-Bibliothek erzeugt. Die Suchunterstützung beginnt ab der Eingabe von mindestens zwei Zeichen, dies ist derzeit in der Konfigurationsdatei noch nicht einstellbar, jedoch wäre eine Anpassung der Anwendung leicht möglich.

3.4.4.1 Suchanfragen erzeugen

Hat der Nutzer sich mit Hilfe der Unterstützung für einen Suchbegriff entschieden, sendet er seine Anfrage mit dem im <submit-button> definierten Element ab. Auf dieses reagiert ein EventListener (vgl. Abbildung 3-12, Element 6b).

Wird der Schalter, der auf der Basis-Website mit der ID submit_db bezeichnet ist, aktiviert, so wird eine Funktion ausgeführt, die prüft, ob das zugehörige Texteingabefeld bereits Inhalt hat und ob gerade keine Suchanfrage ausgeführt wird. Ist dies der Fall, erzeugt diese eine neue Instanz des XTMObject und weist diesem die globale Referenz xtm zu. Das Objekt erwartet bei der Initialisierung als Attribute den URL, an den die Anfrage gesendet werden soll und der in der Konfiguration als <url-xtmrequest> spezifiziert wurde, und zwei Parameter:

- `search`
  Dieser Parameter enthält als Wert den Suchterm.

- `type`
  Dieser Parameter enthält als Wert den Typ der Suchanfrage, wie er in der Konfiguration unter `<param><value>` angegeben wurde.

Der komplette URL mit Parametern lautet im Beispiel wie folgt:

```
http://[Host_der_Basis-Website]/XTMServlet?search=[Term]&type=DB
```

3.4.4.2 Abbilden der XTM-Spezifikation mit JavaScript-Objekten

Um die Elemente der XTM-Spezifikation effizient nutzen zu können, bietet es sich an, Datenstrukturen zu schaffen, welche die Informationen aufnehmen und der Anwendung mittels Objekten zur Verfügung stellen können. Diese Objekte werden in der Datei `xtmspec.js` definiert.

Die Elemente, welche in XTM 1.0 standardisiert wurden, haben nun ein Äquivalent in Form eines JavaScript-Datentyps. Neben der Aufnahme von Informationen bieten diese die Möglichkeit, Funktionen zu definieren, die den Umgang mit den Daten erleichtern. Das Klassendiagramm des prototypischen Objekts `Topic` soll beispielhaft der Erklärung dienen:

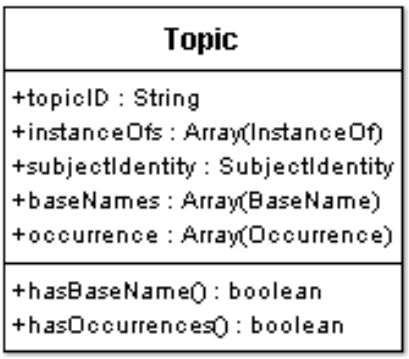


Abbildung 3-14: Klassendiagramm Topic-Objekt

Das Objekt benötigt also bei der Initialisierung fünf Attribute. Das Attribut `topicID` enthält, als Zeichenkette, den eindeutigen Bezeichner des Topics. Alle weiteren Attribute sind Instanzen der jeweiligen Datentypen, die ebenfalls in der `xtmspec.js`-

Datei definiert wurden. So wird bspw. eine Instanz des `subjectIdentity`-Objekts in dem `Topic`-Objekt gespeichert.

In dem in der XTM-Spezifikation standardisierten `Topic`-Element können untergeordnet mehrere `baseName`-, `instanceOf`- und `occurrence`-Elemente beschrieben sein. Diese Elemente werden im `Topic`-Objekt als *Arrays* gespeichert, die eine Liste von Instanzen der JavaScript-Äquivalente zu diesen Elementen enthalten.

Neben dem Speichern der Daten enthält der Objektprototyp `Topic` zwei Funktionen, welche der Applikation dienen, indem sie prüfen, ob ein Topic einen *BaseName* besitzt oder ob *Occurrences* gespeichert wurden.

Das interne Speichern eines Topics geschieht wie folgt:

```
var topicID = "testID";
var instanceOfs = new Array ();
instanceOfs.push (new InstanceOf ("topicRef",
        "subjectIndicatorRef"));
var subjectIdentity = new SubjectIdentity ("resourceRef",
        "subjectIndicatorRef", new Array ("topicRef1"));
var baseNames = new Array ();
baseNames.push (new BaseName ("baseNameString", null, null));
var occurrences = new Array ();
//...
var topic = new Topic (topicID, instanceOfs, subjectIdentity,
        baseNames, occurrences);
```

Im Codebeispiel wurde ein `BaseName`-Objekt initialisiert, das weder einen *Scope* noch ein *Variant*-Element besitzt. Aus diesem Grund erfolgt die Initialisierung mit dem leeren Objekt `null`.

Es sei angemerkt, dass Elemente der XTM-Spezifikation, die lediglich eine Zeichenfolge enthalten können oder nur über ein Attribut, jedoch keine untergeordneten Elemente, verfügen, direkt als Zeichenketten und nicht als gesonderte Objekte gespeichert werden. Dies ist bspw. bei den Pendants des *baseNameString*-Elements und der Referenzmechanismen der XTM-Spezifikation der Fall.

3.4.4.3 Das XTMObject

Die Datei `parser.js` enthält eine der Hauptkomponenten der Anwendung, das `XTMObject`. Dieses bietet die folgenden Funktionalitäten:

- Initialisierung: Erzeugen eines asynchronen AJAX-Request und Empfang der rückgesendeten Daten,
- Parsen: Auslesen des empfangenen XTM-Dokumentes und Speichern des Inhalts in JavaScript-Objekten,
- Erkennen von Topics, die lediglich von Assoziationen und anderen Topics instanziiert werden. Diese werden nicht als Topics dargestellt, sondern als Bezeichner von Assoziationen oder in der *InstanceOf*-Komponente der Topic Darstellung verwendet.
- Generieren des Assoziationsbrowsers auf Basis der eingelesenen Assoziationsnamen,
- Erzeugen von EventListenern, um auf Mausklicks im Assoziationsbrowser zu reagieren,
- Kommunikation mit dem `Canvas`-Objekt, um die Visualisierung zu erzeugen.

Das Flussdiagramm in Abbildung 3-15 beschreibt die Ausführung.

Ist eine AJAX-Abfrage erfolgreich, so erzeugt das *prototype.js*-Framework das Ereignis ‚complete', auf welches ein EventListener reagiert. Dieser erhält die Antwort des Servers, die als XML-Dokument in einem DOM-Baum der Anwendung zum Parsen zur Verfügung gestellt wird.

Innerhalb des `XTMObject` übernehmen die Methoden `parseTopics` und `parseAssociations` diese Aufgabe und erhalten als Eingabe das XML-Dokument. Dies wird nun, mit zahlreichen Hilfsmethoden, wie bspw. `getOccurrences` oder `getMembers`, in eine interne Repräsentation der XTM-Elemente, wie bereits beschrieben, umgewandelt. Der eingesetzte Parser ist der in allen Browsern verfügbare DOM-Parser. So können alle Topic-Elemente über den Aufruf `xml.getElementsByTagName ("topic")` eingelesen werden. Als Ergebnis des Parsens stehen dem `XTMObject` zwei *Arrays*, eines enthält die Topics, das andere die Assoziationen, zur Verfügung.

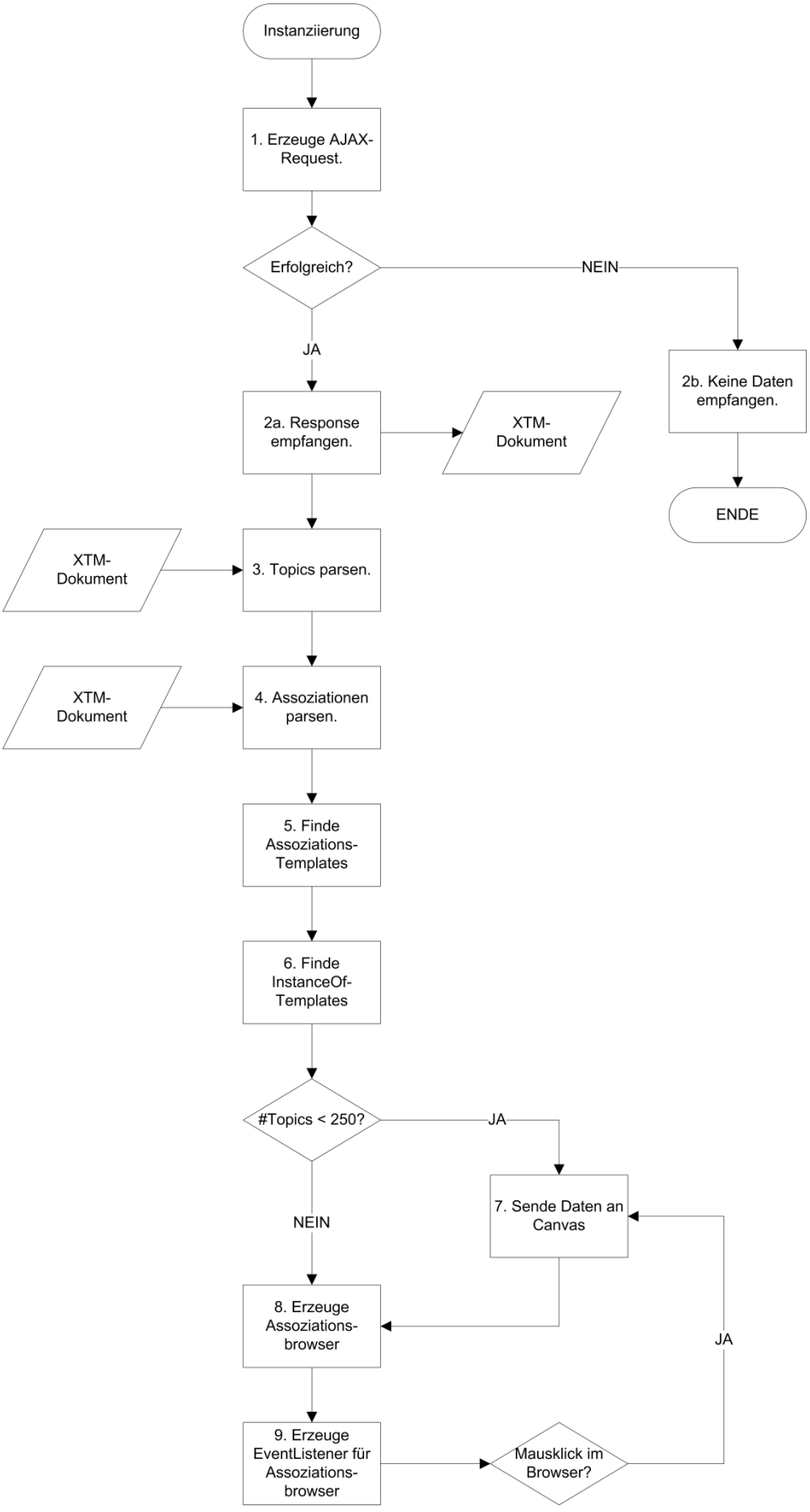


Abbildung 3-15: Flussdiagramm XTMObject

Da im XTM 1.0 Standard jedes Topic und jede Assoziation die Instanz eines Topics sein kann (Klasse-Subklasse-Beziehung), jedoch in der Visualisierung keine Topics dargestellt werden sollen, die lediglich nähere Beschreibungen zu Topics oder den Namen einer Assoziation liefern, müssen diese vor der Darstellung bekannt sein. Dazu durchsucht das `XTMObject` die geparsten Topics und Assoziationen auf solche Beziehungen und speichert zwei Arrays mit den IDs der Assoziations-Vorlagen und der lediglich instanziierten Topics.

In Abbildung 3-15 ist nach Punkt 6 eine Abfrage zu finden, welche die Anzahl der geparsten Topics prüft. Die Zahl 250 entstammt der Konfigurationsdatei `config.xml`; sie ist im Element `<maxTopicsToShow>` hinterlegt. Dies ist eine Sicherheitsbarriere, da eine Visualisierung von zu vielen Topics einen Browser, der auf einem System mit wenig Arbeitsspeicher läuft, wegen eines Speicherüberlaufs zum Absturz bringen kann. Durch die Barriere werden die Topics nicht dargestellt, jedoch der Assoziationsbrowser generiert, so dass der Benutzer Teile der übertragenen XTM-Datei betrachten kann. Auf eigenes Risiko kann er so auch alle Topics darstellen lassen, dies wird jedoch nicht empfohlen.

Die Methode `update` ruft, nach Prüfung der Anzahl der Topics, die Funktion `paint` auf. Diese kommuniziert mit dem `Canvas`-Objekt, wenn die Anzahl der Topics dies zulässt. Andernfalls erzeugt die Funktion nur ein *Array*, das die Namen der Assoziationen enthält. Diese erhält die Funktion durch das Sammeln der *baseNameStrings* der Assoziations-Vorlagen oder, da das *mergeMap*-Element nicht unterstützt wird, die ID einer Assoziations-Vorlage außerhalb der Topic Map. Dieses *Array* dient zur Erzeugung des Assoziationsbrowsers. Ist die Kommunikation mit dem `Canvas`-Objekt erlaubt, werden die *Topic Widgets* (welche die Topics in der Darstellung repräsentieren) erstellt, diese miteinander verbunden, wie es durch die geparsten Assoziationen vorgegeben ist, und die Darstellung eingeleitet. Die Funktionen des `Canvas`-Objekts, die hier aufgerufen werden, werden später erläutert.

Letztlich ruft die Funktion `update` die Methode `generateAsBrowser` auf. Diese erzeugt den Assoziationsbrowser mittels der von der `paint`-Funktion ermittelten Assoziationsnamen und zusätzlich einem Link, der wieder die Gesamtdarstellung erzeugt. Weiterhin werden EventListener registriert, die auf die Links im Assoziationsbrowser reagieren und daraufhin die Methode `paintFromAsBrowser` aufrufen, welche der Methode `paint` nur eine Assoziation, deren Vorlage und die mittels dieser Assoziation verknüpften Topics übergibt. So wird nur ein Teil der geladenen XTM-Datei angezeigt.

Das Klassendiagramm des `XTMObject`, welches in Abbildung 3-16 dargestellt ist, enthält sowohl *private*- als auch *public*-Methoden. Es muss angemerkt werden, dass eine solche Unterscheidung mit JavaScript nicht möglich ist. Dennoch wurde in der Darstellung nicht darauf verzichtet, da dies die geplante Verwendung durch andere Methoden und Nutzer beschreibt. Bei einer geplanten Benutzbarkeit durch andere als in der Klasse selbst formulierte Methoden ist vor der Methode ein ‚+', bei ausschließlich interner Verwendung wurde vor dem Methodennamen ein ‚−' eingefügt.

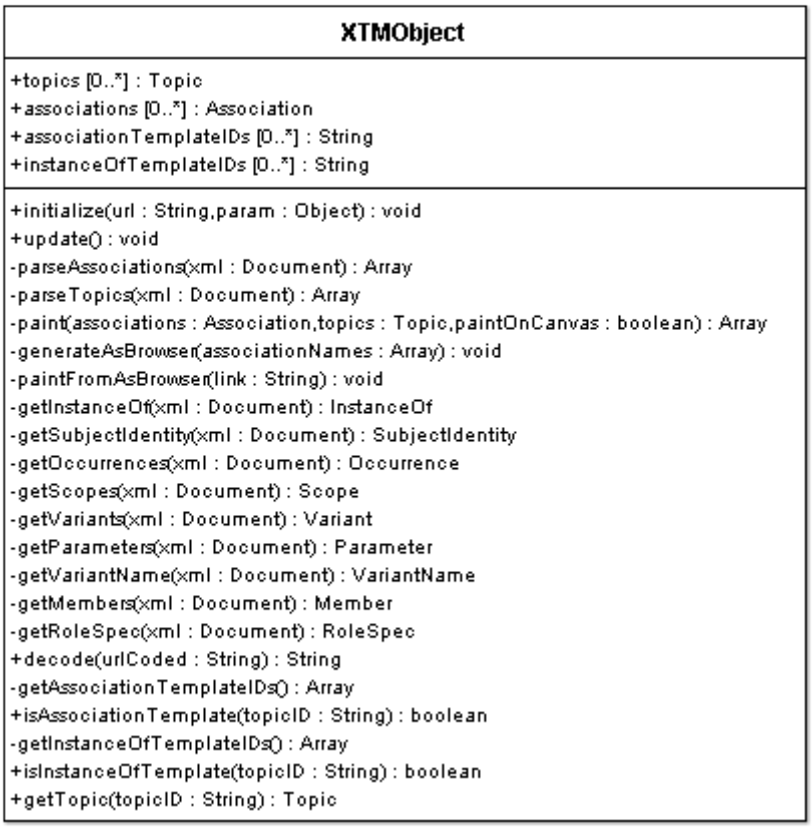


Abbildung 3-16: Klassendiagramm XTMObject

3.4.4.4 Speicherung der History der XTM-Objekte

Die Methode `update` einer Instanz von `XTMObject` löscht den Canvas, erzeugt den Assoziationsbrowser auf Basis der in der Instanz gespeicherten Assoziationen neu und beauftragt die `paint`−Methode mit der Erstellung der *Topic Widgets* und *Lines* im Canvas.

Sollen vergangene Abfragen gespeichert werden, so ist es sinnvoll, eine Art *Stack* mit `XTMObject`-Instanzen zu füllen, da diese alle zur Visualisierung benötigten Daten enthalten und einfach, durch den Aufruf der `update`-Methode, im `Canvas` angezeigt

werden können. Dies ermöglicht das `History`-Objekt. Es wird in der `Main`-Komponente instanziiert und bei jeder Erzeugung einer neuen Suchabfrage mit dem neu erstellten `XTMObject` befüllt. Wird eine Instanz des `History`-Objektes erzeugt, so muss das Element im DOM, welches die Links ‚Back' und ‚Forward' enthalten soll, spezifiziert werden und die Konfiguration in Form einer Instanz des `ConfigObject` übergeben werden.

Um einen neuen History-Punkt zu erstellen, kann in einem instanziierten `History`-Objekt die Funktion `push` aufgerufen werden. Diese benötigt als Argumente das zu speichernde `XTMObject` und den Suchterm, der die Anfrage einleitete. Diese Daten werden in einem Array gespeichert, das maximal 20 Elemente besitzen darf und bei Überschreitung dieser Grenze die ältesten Daten entfernt.

Die Methoden `hasPrev` und `isLast` dienen zur Orientierung innerhalb der gespeicherten Objekte. Ist das aktuell aktive `XTMObject` das erste in der Kette gespeicherte Objekt, so liefert `hasPrev` den Wert `false`, der Link ‚Back' wird also nicht angezeigt. Weiterhin wird der Link ‚Forward' nicht erzeugt, wenn das aktuelle Objekt das letzte in der Kette gespeicherte ist.

Wird einer der beiden Links angeklickt, so reagiert ein vom `History`-Objekt verwalteter EventListener. Dieser ruft die Methoden `changeBack` oder `changeForward` auf. Diese Methoden nutzen `getPrev` und `getNext`, um die gespeicherten Objekte aus dem *Stack* zu erhalten. Daraufhin werden die angezeigten Links aktualisiert und die globale Referenz `xtm` auf das neu ausgewählte Objekt eingestellt.

Zur besseren Integration in den Browser könnte das `History`-Objekt zukünftig auch die Browser-Schaltelemente für das Navigieren in den bereits aufgerufenen Seiten nutzen, um die Darstellung der gespeicherten Suchanfragen zu ermöglichen. Dies ist derzeit nicht implementiert.

3.4.5 Dynamische Visualisierung

Nachdem ein XTM-Dokument eingelesen und in eine interne Repräsentation umgewandelt wurde, wird die Canvas-Komponente genutzt, um die graphische Präsentation der übermittelten Topic Map zu erzeugen. In der Datei `visualizer.js` werden, neben dem `Canvas`-Objekt an sich, Objekte definiert, die dieser Präsentation dienen.

Das `Canvas`-Objekt nutzt die in dieser Datei definierten Objekte und bildet die Kommunikationsschnittstelle. Die in diesem Objekt definierten Methoden werden, sofern sie nicht zur internen Verwendung vorgesehen sind, durch das `XTMObject` aufgerufen.

Zunächst sollen aber die unterstützenden Objekte beschrieben werden, um danach deren Verwendung im `Canvas`-Objekt zu erläutern.

3.4.5.1 Die Klasse GraphElement

Da alle Objekte in der Datei `visualizer.js` mit der graphischen Präsentation befasst sind, gibt es einige Funktionalitäten, die alle diese Objekte benötigen. In der objektorientierten Programmierung bietet es sich deshalb an, Methoden, die von allen diesen genutzt werden, in einer Klasse zu definieren und diese dann zu vererben.

Diese Superklasse heißt `GraphElement`. Sie besitzt einen Konstruktor, welcher mit *prototype.js* durch die Methode `initialize` erzeugt wird. Dieser speichert die Höhe und Breite des Elements, die Verknüpfung zum im DOM angezeigten Element und weist diesem eine eindeutige ID zu.

Die Methoden `update` und `updateFromDOM` dienen der Aktualisierung der Höhe und Breite des Elements. Diese kann von zwei Seiten erfolgen, da das Element nicht nur durch den Code der Anwendung, sondern auch vom Browser selbst manipuliert werden kann. Sollen also Änderungen auf Basis von Berechnungen der Anwendung übernommen werden und das angezeigte Objekt aktualisiert werden, so wird die `update`-Methode genutzt. Wurde das angezeigte Objekt durch den Browser oder durch Funktionalitäten wie *Drag & Drop* aktualisiert und die Änderungen sollen im `GraphElement` übernommen werden, so wird die `updateFromDOM`-Methode aufgerufen, welche die Höhe und Breite des angezeigten Elements aus dem DOM ausliest und in den Variablen `width` und `height` speichert.

Letztlich wird eine Methode definiert, die es ermöglicht, dem `domElement` per DOM-Manipulation ein untergeordnetes Element hinzuzufügen. Diese Methode heißt `addContent` und erwartet als Parameter das vordefinierte Element, welches eingefügt werden soll.

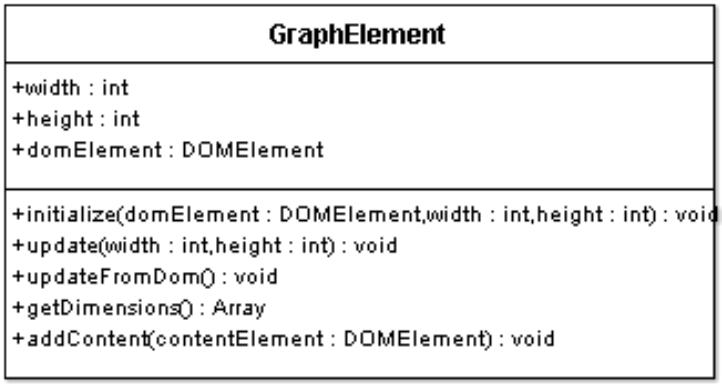


Abbildung 3-17: Klassendiagramm GraphElement

3.4.5.2 Topic Widgets

Die Klasse `TopicWidget` ist eine Subklasse von `GraphElement` und dient der Darstellung von Topics im `Canvas`. Bei seiner Instanziierung erzeugt es ein neues Element, welches in den DOM-Baum des Browsers eingehängt werden kann, vergibt an dieses `domElement` die Klasse ‚topicwidget' und speichert, zusätzlich zu den im `GraphElement` vorgesehenen Werten, die X- und Y-Koordinaten des Elements. Als *Widget* wird eine Komponente einer graphischen Benutzeroberfläche bezeichnet.

Durch die Vergabe einer Klasse können alle Elemente, die zu dieser Klasse gehören, über einen Eintrag im Stylesheet formatiert werden. Der zur Formatierung von *Topic Widgets* vorgesehene Abschnitt der `style.css` lautet:

```
.topicwidget {
  border: 1px solid black;
  background: #919191 none repeat scroll 0%;
  overflow: hidden;
  cursor: move;
  opacity: 0.9;
  z-index: 1000;
}
```

Zusätzlich zu den positionierenden Angaben wird bei der Erzeugung eines *Topic Widgets* das Topic an sich, gespeichert in der JavaScript-Repräsentation des Elements der XTM-Spezifikation, übergeben. Daraus können der Identifikator, die `topicID`, und der Name, der *baseName*, extrahiert werden. Hat das Topic einen *baseName*, so wird dieser innerhalb des `domElements` angezeigt. Andernfalls dient die `topicID` als angezeigter Name. Das gespeicherte Topic wird später ebenfalls vom `InfoWidget` genutzt, um Informationen wie *Occurrences*, Instanziierungen oder die *subjectIdentity* anzuzeigen. Weiterhin kann die Topic ID des in der Visualisierung hierarchisch

übergeordneten Topics angegeben und im `childOf` Attribut gespeichert werden; dies wird vom Positionierungsalgorithmus des `Canvas`-Objekts genutzt.

Letztlich gibt es eine Methode mit dem Namen `makeDraggable`. Durch diese wird das `domElement` mittels der *script.aculo.us*−Bibliothek per Maus bewegbar gemacht. So kann die Maus über eine, in den DOM-Baum eingehängte, Instanz der `TopicWidget` Klasse bewegt werden und dieses mittels der linken, gehaltenen Maustaste verschoben werden. Existiert nun ein Element, das als *Droppable* definiert wurde, und über welchem das gezogene Element ‚fallengelassen' wird, so können bestimmte Aktionen durch einen EventListeners ausgeführt werden. Dies nennt man *Drag & Drop*.

In der erzeugten Applikation sind dies die Formulare, in die eine Suchanfrage eingegeben werden kann. Nimmt man also eine `TopicWidget`-Instanz und zieht sie auf ein solches Formular, so wird direkt eine neue Suche nach dem im *Topic Widget* gespeicherten Topic gestartet. Die Suche geschieht in der Datenquelle, zu der das Suchformular gehört.

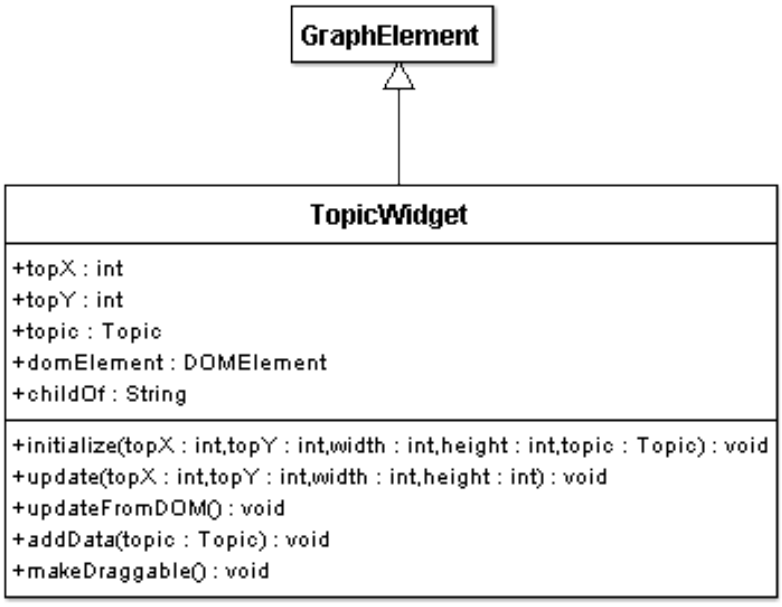


Abbildung 3-18: Klassendiagramm TopicWidget

3.4.5.3 Lines

Zur Darstellung der Beziehungen zwischen Topics, welche in der Topic Map durch Assoziationen beschrieben werden, werden Linien (Lines) benutzt. Diese verbinden zwei *Topic Widgets* und haben einen Bezeichner, der dem von der Assoziation

instanziierten Topic entspringt. Linien nutzen, wie bereits beschrieben, das `GraphElement` als Superklasse und werden durch die Angabe der zu verbindenden `TopicWidget`-Instanzen und einer Zeichenkette, welche den Namen der Assoziation enthält, instanziiert.

Da in Assoziationen innerhalb eines *member*-Elements mehr als ein Topic referenziert werden darf, also eine Assoziation mehr als zwei Knoten verknüpft, gelten sie in der Graphentheorie als Hyperkanten[121]. Ist dies in einer Topic Map der Fall, so müssen diese vor der Visualisierung entfernt werden. Dies übernimmt in der entwickelten Applikation die Serverseite. Dieser bildet aus Hyperkanten einfache, zwei Elemente verbindende, Kanten. Aus diesem Grund muss dies in der Visualisierung nicht berücksichtigt werden.

Die Klasse `Line` enthält die Methoden `initialize`, `update` und `changeParents`. Die beiden bei der Instanziierung übergebenen *Topic Widgets* dienen als Eltern-Elemente, mittels der in diesen gespeicherten Positionsangaben wird die Position der Linie im `Canvas` berechnet.

Die `update`-Methode übernimmt diese Berechnung. Dazu ermittelt sie, welches der *Topic Widgets* eine kleinere Y-Koordinate besitzt und speichert dieses, gegebenenfalls mit Hilfe der `changeParents`-Methode, als `parent1`. Dies dient der Komplexitätsreduktion, denn würde die Belegung völlig frei sein, müssten die im nächsten Absatz beschriebenen und in Abbildung 3-19 dargestellten Fallunterscheidungen deutlich umfangreicher sein.

Dadurch, dass das *Topic Widget* `parent1` immer oberhalb des `parent2` befindet, muss zur Linienberechnung nur noch der Halbkreis unterhalb von `parent1` untersucht werden. In Abbildung 3-19 sind nur 90° abgedeckt; dies dient der Übersichtlichkeit, denn die Flächen auf der rechten, unteren Seite sind lediglich Spiegelungen der beschriebenen Flächen A bis E. Als Spiegelachse dient die Gerade, welche durch die rechte Begrenzung der Fläche E definiert wird.

[121] Vgl. Löhnertz (2008).

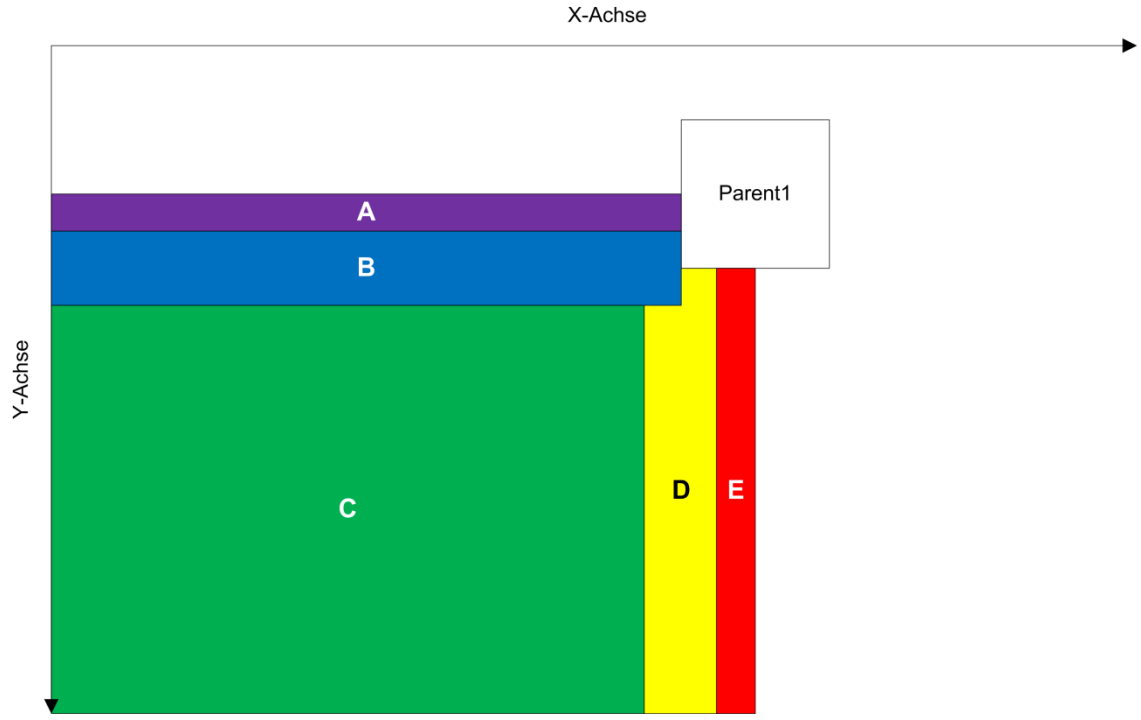


Abbildung 3-19: Fallunterscheidung Linienberechnung

Die Methode `update` prüft zunächst, ob das zweite, hier `parent2` genannte, *Topic Widget* auf gleicher Höhe (Mittelpunkt liegt auf oberer Begrenzung von Fläche A oder deren Spiegelung) oder dessen Mittelpunkt auf der rechten Begrenzung der Fläche E liegt. Ist dies der Fall, so erfolgt die Verbindung mit horizontalen oder vertikalen Linien, welche an den sich zugewandten Seiten jeweils in der Mitte der *Topic Widgets* eingepasst werden.

Befindet sich der Mittelpunkt des `parent2` innerhalb einer der Flächen A bis E, so werden die folgenden Verbindungen erstellt:

A) Diagonale Line von der Mitte der zugewandten Seite des `parent2` zur unteren Ecke der zugewandten Seite des `parent1`,

B) Diagonale Linie von der Mitte der zugewandten Seite der `parent1` zur Mitte der zugewandten Seite des `parent2`,

C) Diagonale Linie von der oberen Ecke der zugewandten Seite des `parent2` zur unteren Ecke der zugewandten Seite des `parent1`,

D) Diagonale Linie von der Mitte der oberen Seite des `parent2` zur Mitte der unteren Seite des `parent1`,

E) Diagonale Linie von der Mitte der oberen Seite des `parent2` zur linken unteren Ecke des `parent1`.

Zur Erzeugung der Linien werden die in der `config.xml` angegebenen Dateien genutzt. Die Elemente, in welchen Linien spezifiziert werden, heißen `<line-horizontal>`, `<line-vertical>`, `<line-upper-left-lower-right>` und `<line-upper-right-lower-left>`. Die Graphikdateien sind im PNG-Format gespeichert. *PNG*[122] erlaubt die Definition von transparenten Bereichen. Dies ist jedoch nicht mittels der sog. Alphakanaltransparenz gelöst, da ältere Browser teilweise Probleme mit der Anzeige dieser haben. Vielmehr wird eine Farbe der Palette als transparent deklariert.

Graphikdateien für Linien bestehen also aus der eigentlichen Linie und einem transparenten Hintergrund. Diese Graphiken werden dann, mittels DOM-Manipulation, an der richtigen Stelle im `Canvas` eingefügt und gestreckt oder gestaucht, um die richtige Länge und Breite zu erhalten.

Durch das Strecken und Stauchen erscheinen die Linien teilweise gepunktet oder unterscheiden sich in der Linienstärke. Dieses Problem wäre derzeit nur durch eine größere Menge an Vorlagen zu lösen. Da das Vektorgraphikformat *SVG* von älteren Browsern nativ nicht unterstützt wird, kann es erst in einigen Jahren als Alternative dienen.

3.4.5.4 InfoWidget

Info Widgets dienen der Anzeige von Informationen, die in einem Topic gespeichert sind, jedoch in der Standard-Anzeige keinen Platz finden. Bewegt der Nutzer seine Maus über ein *Topic Widget* und belässt den Mauszeiger länger als eine Sekunde innerhalb der Umgrenzung des *Topic Widgets*, so werden die zusätzlichen Informationen zum Topic eingeblendet.

Wie Abbildung 3-20 zeigt, wird der *baseName* als Überschrift angezeigt. Existiert eine *subjectIdentity*, so wird die Überschrift durch den Link zur hinterlegten Ressource ergänzt.

[122] Vgl. Völkel (2008).

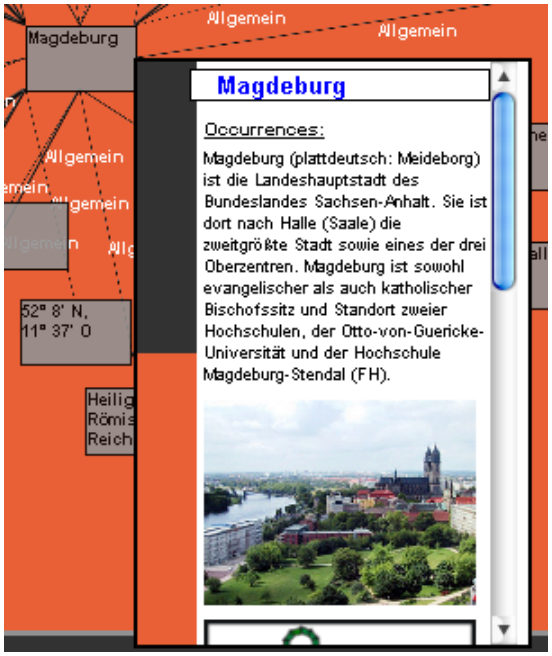


Abbildung 3-20: InfoWidget Topic Magdeburg

Im Textfeld werden Vererbungen (*instanceOf*), alternative Namen (weitere *baseName* Elemente) und *Occurrences* eingeblendet. Sind in den *Occurrences* Links vorhanden, so werden diese überprüft. Handelt es sich bei der referenzierten Ressource um ein Bildformat, so wird dies angezeigt. Andernfalls wird die Ressource durch eine XHTML-Referenz verknüpft.

Bewegt der Nutzer die Maus und verlässt das *Topic Widget* oder das *Info Widget*, so werden die Zusatzinformationen ausgeblendet, um die Standard-Ansicht der Topic Map wiederherzustellen.

3.4.5.5 Das Canvas-Objekt

Die Hauptkomponente der Visualisierung wird durch das `Canvas`-Objekt erzeugt. Diese bietet Methoden zur Erzeugung und Positionierung von *Topic Widgets* und zum Verbinden von *Topic Widgets* mittels Assoziationen. Weiterhin kann die komplette Visualisierung, aber auch ein einzelnes Topic, gelöscht werden. Die wichtigsten Methoden sind in Tabelle 6 angegeben.

Wird ein `Canvas`-Objekt instanziiert, so erzeugt es innerhalb seines zugeordneten DOM-Elements in der Basis-Website zwei neue Abschnitte, sogenannte `<div>`-Tags. Ersteres enthält später die *Lines* und letzteres die *Topic Widgets*. Es werden zwei Arrays, welche die *Topic Widgets* und die *Lines* speichern, und ein Drittes zum Aufbau

einer Hierarchie, wie in 3.4.5.6 beschrieben, erzeugt. Weiterhin wird ein Attribut zur Speicherung des aktuellen *Info Widgets* erzeugt und gespeichert, ob vom `Canvas` Linien und Assoziationsnamen angezeigt werden sollen.

Wurden bereits interne Repräsentanten von Topics erzeugt, wie in 3.4.4.2 beschrieben, so können diese dem `Canvas`-Objekt mittels der Methode `createTopic` übergeben werden. Diese Methode erzeugt ein neues *Topic Widget* und speichert dies. Ein Beispiel der Verwendung ist im Folgenden beschrieben.

```
var c = new Canvas ($(config.canvasParent));
c.createTopic (aTopic);
c.createTopic (bTopic);
```

Existiert nun eine Assoziation zwischen den Beispieltopics `aTopic` und `bTopic`, so kann diese ebenfalls dem `Canvas` bekannt gemacht werden. Durch die `connectTopics`-Methode wird eine neue `Line` erzeugt und gespeichert.

```
c.connectTopics (aTopic.topicID, bTopic.topicID, "Beispiel");
```

Sind alle darzustellenden Topics und Assoziationen übermittelt, so können diese positioniert und gezeichnet werden. Zu diesem Zweck muss zunächst die Methode `paintWidgets` und dann die Methode `paintLines` aufgerufen werden:

```
c.paintWidgets ();
c.paintLines ();
```

Es ist nötig, zuerst alle Topics und Assoziationen zu übermitteln, denn nur auf Basis dieser Daten kann eine Darstellungsform berechnet werden. Die Kommunikation und der Aufruf der Methoden erfolgt, wie beschrieben, durch die Funktion `paint` des `XTMObject`.

Tabelle 6: Wichtige Methoden der Canvas-Klasse

Methode	Erläuterung
initialize	Wird bei der Instanziierung aufgerufen. Erwartet als Parameter das zugeordnete DOM-Element, in der `config.xml` durch das Element `canvas-parent` spezifiziert.
showLoadSymbol	Erwartet einen boolschen Parameter, welcher angibt, ob im Zentrum des zugeordneten DOM-Elements ein Lade-Symbol angezeigt werden soll.
clear	Löscht sämtliche *Topic Widgets* und *Lines*.
createTopic	Erzeugt ein neues *Topic Widget*. Erwartet als Parameter ein Topic, wie in 3.4.4.2 beschrieben.
connectTopics	Verbindet zwei *Topic Widgets* miteinander und erzeugt eine neue Instanz des `Line`-Objekts. Erwartet als Parameter die IDs der beiden Topics und den Namen der abgebildeten Assoziation.
paintWidgets	Fügt die erzeugten *Topic Widgets* in den DOM-Baum des Browsers ein. Nutzt den Positionierungsalgorithmus, beschrieben in 3.4.5.6.
paintLines	Fügt die erzeugten *Lines* in den DOM-Baum des Browsers ein.
updateFromDOM	Wurde das Browser-Fenster, in dem der `Canvas` angezeigt wird, verkleinert oder vergrößert, so werden mit dieser Methode die *Topic Widgets* neu positioniert und deren Größe verändert. Weiterhin wird geprüft, ob sich der Assoziationsbrowser und das DOM-Element, welches den `Canvas` enthält, überlagern und ggf. das DOM-Element neu positioniert. Letztlich werden die *Lines* aktualisiert.
setLineVisibility	Legt fest, ob Lines angezeigt werden sollen. Erwartet als Parameter einen boolschen Wert.
setLineNameVisibility	Legt fest, ob die Namen der Assoziationen angezeigt werden sollen. Erwartet als Parameter einen boolschen Wert.

3.4.5.6 Positionierung der Topic Widgets

Die Positionierung der *Topic Widgets* besteht aus drei Teilen, welche in diesem Abschnitt besprochen werden sollen.

Zunächst muss über die Anzahl der *Topic Widgets* die Größe dieser bestimmt werden, also die Höhe und Breite ermittelt werden. Dies geschieht über die folgende Funktion:

$$widgetSize(\#TopicWidgets) = \begin{cases} \dfrac{canvasHeight}{10} & \text{für } \#TopicWidgets < 10 \\ \dfrac{canvasHeight}{10 + \sqrt{(\#TopicWidgets - 10) * 3}} & sonst \end{cases}$$

Die Funktion, welche die Breite der *Topic Widgets* berechnet, ersetzt lediglich `canvasHeight` durch `canvasWidth`. Diese Funktion ist konstant bis mehr als 10 *Topic Widgets* angezeigt werden. Bei einer höheren Anzahl von angezeigten Topics sinkt der Größenunterschied, da jedes einzelne *Topic Widget* ohnehin weniger Flächeninhalt besitzt als bei wenig angezeigten Topics.

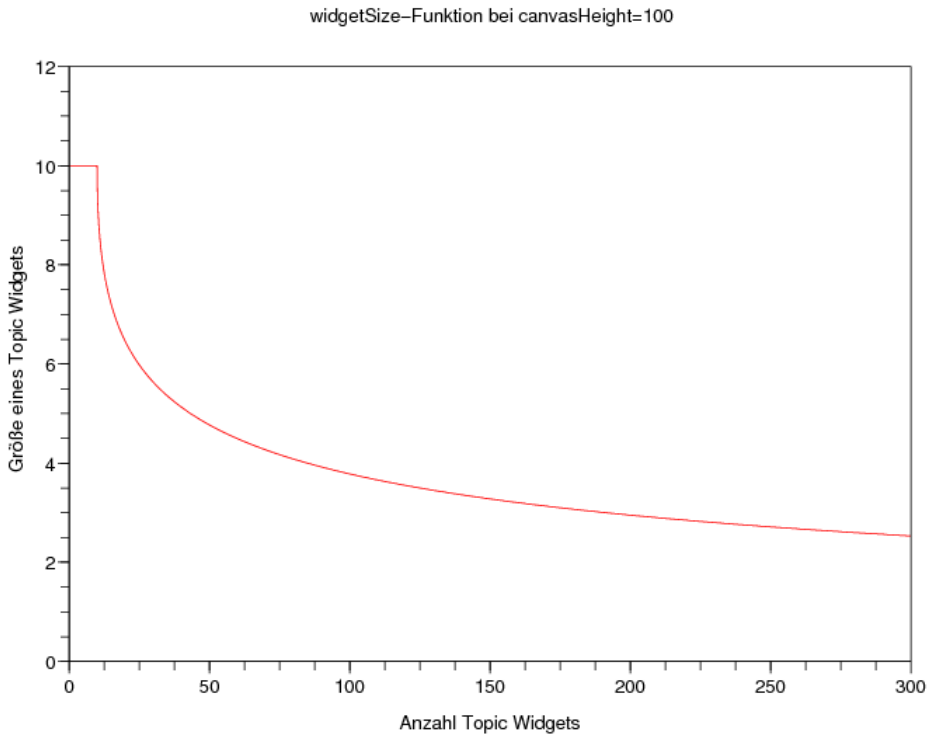


Abbildung 3-21: widgetSize-Funktion (Plot)

Der zweite Schritt ist der Aufbau einer Hierarchie in der Darstellung. Wie bereits beschrieben, handelt es sich bei Topic Maps um Netze ohne zentralen Einstiegspunkt und ohne ein klares ‚oben und unten'. Dies mag den menschlichen Denkprozessen nahe kommen, ist aber für die Visualisierung eine Herausforderung.

So muss das Programm eine Möglichkeit finden, Elemente, die von hohem Interesse sind, zu zentralisieren, ohne zu wissen, welches Element gesucht wurde. Denn die gespeicherten *Topic Widgets* sind gleich, es gibt zwischen ihnen keinen Unterschied. Der Ansatzpunkt, der in der entwickelten Applikation verwendet wurde, lautet wie folgt.

Jedes *Topic Widget* ist mit 0 bis n weiteren *Topic Widgets* verknüpft. Im `Canvas`-Objekt existiert ein Zähler, der `topicAsCounter`, welcher die Anzahl der Verknüpfungen pro Topic ID speichert. Sind nun zwei Topics verknüpft und ist der Zähler für ein Topic höher als der für das andere, so ist das Topic mit der geringeren Anzahl an Verknüpfungen als Kindelement des anderen Topics zu betrachten. Die Instanz der Klasse `TopicWidget`, welche weniger Verknüpfungen hat, erhält als Attribut `childOf` die Topic ID des übergeordneten Elements.

Dies erzeugt eine Hierarchie; es existiert eine Methode in der `Canvas`-Klasse, welche `getChildsOf` heißt und als Parameter eine Topic-ID erwartet. Wird nun eine leere Zeichenkette übergeben, so erzeugt die Methode ein Array von *Topic Widgets*, welche kein Kindelement sind, also der höchsten Hierarchiestufe angehören. Diese Elemente werden vom Positionierungsalgorithmus zentralisiert.

Nachdem eine Hierarchie aufgebaut wurde, kann die Positionierung erfolgen. Da es sich bei Topic Maps um semantische Netze handelt, die mathematisch durch Graphen repräsentierbar sind, wurde zunächst ein Ansatz auf Basis von selbstorganisierenden Karten, sogenannten Kohonen-Netzen[123], gewählt. Dieser Ansatz führte zu guten Ergebnissen, doch das Erlernen der Karte führte zu einer schlechten Antwortzeit der Anwendung.

Aus diesem Grund wurde ein eigener Algorithmus zur Visualisierung entwickelt. Basis dieses Algorithmus sind die Hierarchiestufen, welche durch die `getChildsOf`-Methode gegeben sind.

Für die Berechnung werden alle Koordinaten in ein Koordinatensystem überführt, welches die Dimensionen 100 * 100 besitzt. Da der Positionierungsalgorithmus mit

[123] Vgl. Kohonen (1995).

Kreisen und in diesem Koordinatensystem arbeitet, wird nach Rücktransformation die gegebene Fläche durch Ellipsen optimal ausgenutzt. Die Transformation wird von den Methoden `scaleDown` und `scaleUp` durchgeführt, diese erwarten als Parameter ein `TopicWidget` und verändern dessen Koordinaten.

Der Ablauf des Algorithmus erfolgt dann wie in Abbildung 3-22 dargestellt. Der Aufruf der `positionWidget`-Funktion findet in der `paintWidgets`-Methode statt. Hier wird für jedes Element der obersten Hierarchiestufe die Funktion einmal aufgerufen.

```
var topLevel = this.getChildsOf (null);
for (var i = 0; i < topLevel.length; i++) {
        this.positionWidget (topLevel[i], null, i,
          topLevel.length, 0, 100);
}
```

Der erste übergebene Parameter ist das *Topic Widget* an sich, der zweite spezifiziert das Eltern-Element. In der Folge werden die aktuelle Position auf der jeweiligen Ebene, hier die Laufvariable `i`, und die Anzahl der Elemente, die auf dem gleichen Niveau liegen, hier `topLevel.length`, übergeben. Die letzen Parameter weisen den zu positionierenden Elementen einen Bereich des `Canvas` zu; hier sind die gesamten 100% zur Verfügung gestellt. Auf niedrigeren Ebenen können dies bspw. die zweiten 50% des `Canvas` sein, die Parameter wären entsprechend 50 und 100. Die Aufteilung des `Canvas` erfolgt derzeit nur horizontal, da die Elemente der obersten Hierarchieebene entsprechend horizontal verteilt werden.

Wie aus Abbildung 3-22 ersichtlich, handelt es sich bei der `positionWidget`-Methode um einen rekursiven Algorithmus. Einmal für ein Element der höchsten Hierarchiestufe aufgerufen, werden sämtliche Kind-Elemente ebenfalls positioniert. Diese ordnen sich auf konzentrischen Kreisen um das Eltern-Element an, die Berechnung der Position erfolgt anhand der zur Verfügung stehenden Fläche, der Gesamtzahl an Elementen, die um das Eltern-Element angeordnet werden sollen und der Nummer des aktuell zu positionierenden Elements.

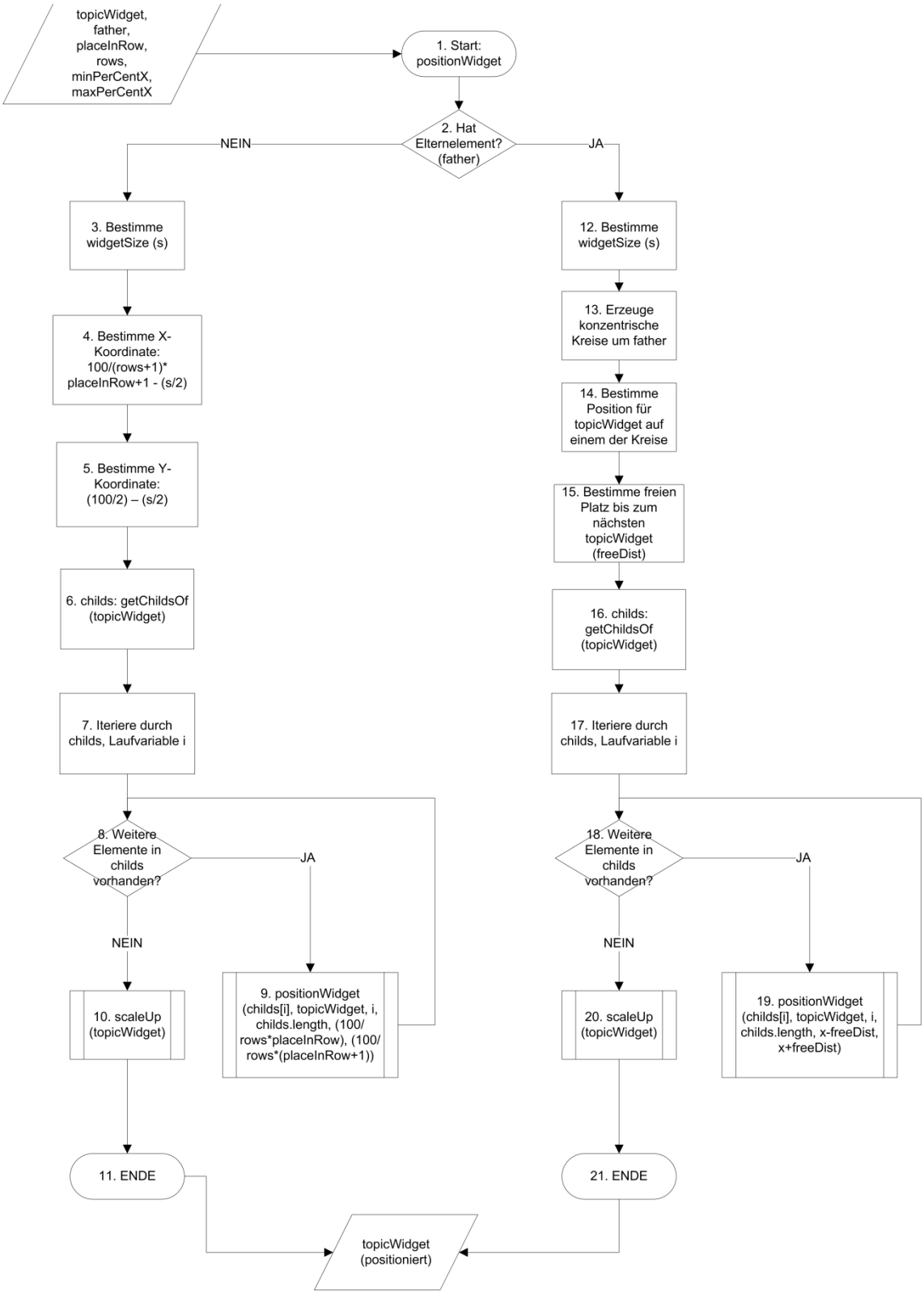


Abbildung 3-22: Flussdiagramm Positionierungsalgorithmus

Hat dieses Kind-Element selbst Kind-Elemente, so wird die `positionWidget`-Methode für diese erneut aufgerufen. Der zur Verfügung stehende Platz wird anhand der Distanz, die zwischen den bereits platzierten *Topic Widgets* noch besteht, an die Methode weitergegeben. Dies ist für zwei bis drei Hierarchie-Ebenen ausreichend, sollen jedoch weitere dargestellt werden, so wäre eine Abhängigkeit der Größe der *Topic Widgets* von der Hierarchie-Ebene, auf der sie sich befinden, sinnvoll, um den Platz zur Positionierung freizugeben. Da jedoch die Anwendung nur Topic Maps mit einer Hierarchiestufe darstellt, weil der Server nur diesen Teil aus großen Topic Maps ausschneidet, ist der Algorithmus in seiner dargestellten Form ausreichend.

Nachdem der Positionierungsalgorithmus abgeschlossen ist und alle Elemente auf die tatsächliche Größe des `Canvas` zurückskaliert wurden, werden alle *Topic Widgets* und *Lines* in den DOM-Baum eingefügt.

Mit diesem Abschnitt ist der clientseitige Teil der Anwendung komplett beschrieben, es folgt die Beschreibung der serverseitigen Komponenten.

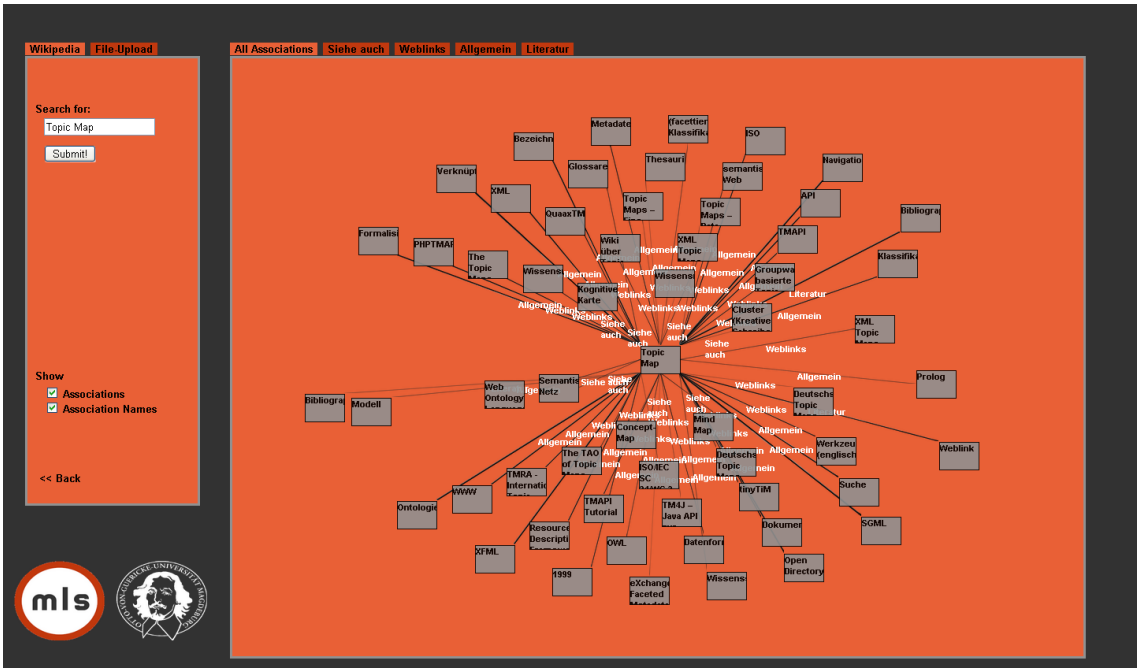


Abbildung 3-23: Anzeige einer Topic Map im Client

3.5 Entwurf und Implementierung des Servers

Die Serverkomponenten werden in diesem Abschnitt thematisiert. Die Beschreibung fällt nicht so umfangreich wie die der Clientkomponenten aus. Hauptsächlich wird die Komponente `XTMServlet` und die Klassen, welche diese benutzt, beschrieben. Komponenten wie der `Autocompleter` und das `Upload`-Servlet werden lediglich erwähnt, jedoch nicht in aller Ausführlichkeit erläutert.

In den Anlagen F bis J dieser Arbeit sind die Klassendiagramme der serverseitigen Anwendung abgebildet.

3.5.1 Datenempfang

Client und Server kommunizieren über das HTTP-Protokoll. Mittels eines URL übergibt der Client Suchanfragen und spezifiziert die Ressource, in welcher die Suche erfolgen soll. Diese Parameter können von Servlets ausgewertet werden und dienen als Ausgangspunkt der Beschreibung des `XTMServlet`. Da die Anfrage durch einen HTTP-Request mit der Methode GET[124] erzeugt wurde, wird die Funktion `doGet` ausgeführt, die automatisiert zwei Parameter übergibt. Der erste Parameter beschreibt die Anfrage, den *Request*. Mittels des zweiten Parameters kann eine Antwort auf die Anfrage erzeugt werden, diese heißt *Response*.

Die vom Client durch den URL übermittelten Parameter heißen `type` und `search`. Zunächst prüft das `XTMServlet` die Existenz des ersten Parameters, um die Suchanfrage an die passende Ressource zu übermitteln. Die erlaubten Werte des `type` Parameters sind:

- `DB`
  Das `XTMServlet` leitet die Suchanfrage an die Klasse `DB2XTMExporter` weiter. Diese extrahiert aus einer Datenbank, die ein per Upload hochgeladenes XTM-Dokument in einem relationalen Datenbankschema enthält, die zur Suchanfrage relevanten Teile.

- `WIKI`
  Die Suchanfrage wird an das *XTMGenerator*-Paket[125], die *Wikipedia*-Artikel in XTM-Dokumente umwandelt, weitergeleitet.

[124] Vgl. Wöhr (2004), S. 220
[125] Vgl. Twele (2008).

Der Aufruf der eingebundenen Klassen erfolgt jedoch nur dann, wenn auch der `search` Parameter einen Wert übermittelt hat. Von den eingebundenen Klassen wird ein Dokument zurückgeliefert. Diese enthalten das zur Suchanfrage generierte XTM-Dokument; es wird als Antwort an die Client-Applikation gesendet und kann dort visualisiert werden.

3.5.2 Erzeugen von XTM-Dokumenten aus Wikipedia-Artikeln

Wurde als `type`-Parameter der Wert `WIKI` übergeben, so leitet das `XTMServlet` die Anfrage an die, in dem *XTMGenerator* Java-Archiv gebündelten, Klassen und Methoden weiter. Diese erzeugen mittels der JDOM-Bibliothek[126] aus *Wikipedia*-Artikeln ein XTM-Dokument. Dies ist beschrieben in (Twele (2008)).

Der Aufruf geschieht durch den folgenden Code:

```
IXTM xtm = new XTM();
Document doc = xtm.getXTM(searchTerm, 25);
```

Es wird eine neue Instanz der `XTM`-Klasse mittels des Interface `IXTM` erzeugt. Dieser Instanz, `xtm` genannt, wird der Suchterm, gespeichert in der Variable `searchTerm`, übergeben. Weiterhin wird dem Objekt mitgeteilt, dass, sollte zu dem Suchterm kein passender *Wikipedia*-Artikel gefunden werden, die Suche innerhalb der *Wikipedia* nach ähnlichen Artikeln maximal 25 Ergebnisse zurücksenden soll. Diese Begriffe übermittelt das Objekt ebenfalls in Form eines XTM-Dokumentes.

Abschließend übergibt das `XTMServlet` erhaltene Dokument per *Output-Stream* und HTTP-Response an den Client.

3.5.3 Upload von XTM-Dokumenten

Um, wie in den Anforderungen an die zu entwickelnde Software formuliert, Topic Maps dynamisch verarbeiten zu können, ist die Möglichkeit eines Datei-Uploads notwendig. Topic Maps, welche eine Wissensdomäne beschreiben, werden schnell mehrere Megabyte groß. Der DOM-Parser, der vom Client verwendet wird, ist bei solchen Datenmengen überlastet. Weiterhin wäre eine Darstellung der gesamten Topic Map unübersichtlich, da teilweise mehrere tausend Topics dargestellt werden würden.

[126] Vgl. Hunter (2008).

Aus diesen Gründen werden hochgeladene Dateien von einer Java-Klasse mittels des effizienten SAX-Parsers eingelesen und in ein relationales Datenbankschema verwandelt. Innerhalb dieser Datenbank ist dann eine gezielte Suche nach einem Suchbegriff und den Assoziationen, welche das, den Suchbegriff repräsentierende, Topic verknüpfen, möglich. So wird ein Ausschnitt aus der Topic Map erstellt, der darstellbar ist. Dieser Prozess wird in den kommenden drei Abschnitten erläutert.

3.5.3.1 Umwandlung von XTM-Dokumenten in relationales Datenbankschema

Nachdem ein XTM-Dokument durch den Nutzer hochgeladen wurde, übergibt es das `Upload`-Servlet an den in der JDOM-Bibliothek definierten `SAXBuilder`. Dieser erzeugt den Datentyp `Document`, der die XTM-Datei für das anschließende Parsen vorbereitet. Weiterhin wird die *Session*, eine eindeutige Zeichenkette, die für jeden Nutzer der Web-Applikation automatisiert vom Servlet Container erstellt wird, gespeichert. Beide Variablen werden daraufhin an eine Instanz der Java-Klasse `XTM2DBExporter` weitergeleitet.

Nach Erhalt der Daten erzeugt die `XTM2DBExporter` Instanz eine Verbindung zu der Datenbank, in welcher der Inhalt des XTM-Dokumentes gespeichert werden soll. Diese Verbindung wird durch die JDBC-Schnittstelle generiert. Die Datenbank muss bereits erzeugt worden sein, dies geschieht durch den im Anhang C aufgeführten SQL-Code.

Die erzeugten Spalten in der Tabelle lauten:

- `XTM_SESSION_ID`
  Enthält die eindeutige Zeichenkette zur Identifikation des Nutzers, welcher die Daten eingespeist hat.

- `XTM_ENTRY_ID`
  Eindeutige Nummer innerhalb einer Session, wird für jedes *topic*-, *association*- und *mergeMap*-Element neu vergeben.

- `XTM_ENTRY_TYPE`
  Spezifiziert, ob es sich bei der `XTM_ENTRY_ID` um ein *topic*-, *association*- oder ein *mergeMap*-Element handelt.

- `XTM_PATH_ID`
  Beschreibt das in der aktuellen Zeile gespeicherte XTM-Element. Jedes Element der XTM-Spezifikation, bspw. *baseNameString* oder *member*, bekommt dabei eine eindeutige ID.

- `XTM_TEXT`
  Enthält ein XTM-Element eine Zeichenkette oder einen CDATA-Bereich, wie bspw. das *baseNameString*-Element, so wird dies in dieser Spalte abgelegt.

- `XTM_ATTRIBUTE_ID`
  Enthält das gespeicherte XTM-Element ein Attribut, welches ihm eine eindeutige ID zuweist, wird diese hier gespeichert. Innerhalb einer `XTM_ENTRY_ID` wird dieses Feld nicht gelöscht, so dass bspw. ein *baseNameString* die ID des zugehörigen Topics führen kann.

- `XTM_ATTRIBUTE_HREF`
  Enthält mittels der XTM-Referenzmechanismen erzeugte Links auf Topics oder andere Ressourcen in Form eines URL.

Innerhalb der `XTM_ENTRY_TYPE` und `XTM_PATH_ID` Spalten werden die Elemente der XTM-Spezifikation durch identifizierende Zahlencodes gespeichert. Diese Codes liegen der Klasse `XTM2DBExporter` in Form der in Anhang D hinterlegten `HashMap` vor. Die Methode `listNodes` erzeugt beim Parsen des XTM-Dokumentes einen Pfad, der durch die `HashMap` in den entsprechenden Code umgewandelt wird.

Die `XTM_ENTRY_ID` identifiziert innerhalb einer Session jedes *topic*-, *association*- und *mergeMap*-Element eindeutig. Aus diesem Grund fragt die `XTM2DBExporter`-Instanz vor dem Einfügen eines neuen XTM-Dokumentes in die Datenbank ab, ob der Nutzer bereits Daten eingespeist hat. Ist dies der Fall, so wird die maximale `XTM_ENTRY_ID` des Nutzers bestimmt und ab dieser weitergezählt.

Ist das XTM-Dokument komplett eingelesen, so kann der Nutzer innerhalb der Topic Map navigieren. Dazu gibt er in das entsprechende Suchfeld in der Client-Anwendung einen Begriff ein, der als *BaseName* in der hochgeladenen Topic Map vergeben wurde. Natürlich erhält er von dem `Autocomplete`-Servlet dabei Unterstützung. Hat er einen Suchterm ausgewählt, so wird aus der Datenbank eine relevante Teilmenge der hochgeladenen XTM-Datei extrahiert.

Da die Anwendung derzeit frei im WWW verfügbar ist, verfallen die hochgeladenen Daten mit dem Ablauf der *Session*, dieser Zeitraum ist zunächst auf zwei Stunden, in denen der Benutzer nicht aktiv war, festgesetzt worden. Bei einer großen Anzahl an Nutzern verlangsamt sich die Suche in der Datenbank, wie im nächsten Abschnitt beschrieben, da sie dann sehr viele Einträge enthält. Dies könnte durch das Anlegen von gesonderten Tabellen für jede *Session* gelöst werden.

3.5.3.2 Erzeugen einer relevanten Teilmenge aus der Datenbank

Um zu beschreiben, wie eine zu einem Suchbegriff relevante Teilmenge aus einer Topic Map in der Datenbank extrahiert wird, ist es nötig zu definieren, was eine relevante Teilmenge ist.

Zur relevanten Teilmenge gehören:

1. Topics, die einen *BaseName* besitzen, der Äquivalent zum Suchbegriff ist,
2. Alle Assoziationen, die in 1 gefundene Topics als *Member* enthalten,
3. Alle Topics, die mit 1 durch eine in 2 gefundene Assoziation verknüpft sind,
4. Alle Topics, deren Instanzen die vorher gefundenen Topics und Assoziationen sind.

Während die Topics möglichst komplett aus der Datenbank wiederhergestellt werden, werden die Assoziationen verändert. Diese können mehr als zwei Topics verknüpfen, es sind sog. Hypergraphen. Da aber die Client-Applikation nur Assoziationen mit zwei *member*-Elementen und jeweils einer Referenz auf ein Topic verarbeiten kann, werden diese hier generiert.

Liegt also die Referenz auf das Topic, das den Suchbegriff repräsentiert, innerhalb eines *member*-Elements, welches mehrere Topic-Referenzen enthält, so werden alle anderen Referenzen ignoriert, da nur die Verknüpfung mit dem gesuchten Topic der Teilmenge entspricht. Enthält das zweite *member*-Element, welches das gesuchte Topic nicht enthält, ebenfalls mehrere Topic-Referenzen, so werden entsprechend der Anzahl an Referenzen neue Assoziationen erzeugt, die das gesuchte Topic und je eines der damit verknüpften Topics enthalten.

Alle gefundenen Topics und Assoziationen werden in einer internen Datenstruktur abgelegt, die den bereits erläuterten JavaScript-Datenstrukturen ähnlich ist[127]. Es sei angemerkt, dass dazu eine Bibliothek basierend auf TMAPI[128] für Java existiert. Diese, TM4J genannt, kann leicht integriert werden, wurde aber für die vorliegende Implementierung nicht genutzt.

Die Klasse `DB2XTMExporter` erhält also den Suchbegriff und die aktive *Session* vom `XTMServlet`. Daraufhin wird die Methode `getMainTopics` aufgerufen, welche zwei Parameter besitzt, die beide vom Datentyp `String` sind. Ersterer kann einen

[127] Vgl. Anlage F.
[128] Vgl. Ahmed et al. (2008).

Suchterm übermitteln, um die entsprechende Topic-ID in der Datenbank zu finden. Ist dies geschehen, ruft die Methode sich selbst wieder auf, allerdings mittels des zweiten Parameters, der Topic-ID. Auf Basis der Topic-ID können die zum gespeicherten Topic zugehörigen Einträge in der Datenbank identifiziert, extrahiert und in der internen Datenstruktur für Topics abgelegt werden.

Sind die Topics eingesammelt und in einem `Set`, einem Datentyp, der nur einzigartige Elemente speichert, so dass keine doppelten Einträge erzeugt werden, gespeichert, wird die Methode `getAssociations` mit dem gespeicherten `Set` als Parameter aufgerufen. Diese durchsucht die in der Datenbank gespeicherten Assoziationen und filtert die der relevanten Teilmenge entsprechenden heraus. Diese werden, wie beschrieben, in einem neuen `Set`, bestehend aus der internen Repräsentation der Assoziationen, abgelegt.

Letztlich wird die Methode `getAllTopics` aufgerufen. Diese sammelt auf Basis der übergebenen `Sets`, welche Topics und Assoziationen enthalten, alle Topics aus der Datenbank, die durch ein *instanceOf*-Element die Assoziation näher beschreiben oder innerhalb eines *member*-Elements referenziert werden, ein und ergänzt das aus Topics bestehende `Set` um diese. Dazu nutzt die Methode die beschriebene `getMainTopics`-Methode.

Als Ergebnis erhält die Anwendung zwei `Sets`. Eines enthält alle Topics, das andere alle Assoziationen der relevanten Teilmenge.

3.5.3.3 Umwandlung der relevanten Teilmenge in XTM

Da der Client ein valides XTM-Dokument vom Server als Antwort erwartet, muss die relevante Teilmenge, welche bereits als interne Datenstruktur vorliegt, in XTM zurückgewandelt werden.

Dies geschieht ebenfalls in der `DB2XTMExporter` Klasse. Zunächst wird spezifiziert, dass es sich bei den gesendeten Daten um den MIME-Typ `text/xml` handelt. Der *Prolog* des Dokumentes wird mit der Methode `writeHeader` erzeugt, welche die XML-Version und die Kodierung des Dokumentes festlegt.

In der Folge wird ein sog. SAX-Transformer zur Erzeugung eines XTM-Dokumentes genutzt, wie in (Marchal (2008)) beschrieben. Die `writeXML`-Methode erzeugt dazu das Wurzelelement `topicMap` und führt den `XLink`-Namensraum ein. In der Folge schreiben die Methoden `writeTopics` und `writeAssociations` die relevante

Teilmenge in das XTM-Dokument, welches dann an das `XTMServlet` übergeben und von diesem an den Client gesendet wird.

3.6 Anwendungsbeispiel

Nachdem die Entwicklung generell und der Entwurf und die Implementierung für die einzelnen Komponenten speziell beschrieben wurden, soll an dieser Stelle ein Anwendungsbeispiel der entwickelten Applikation gegeben werden. Dies geschieht anhand einer schrittweisen Anleitung, die, wenn sie vom Leser komplett ausgeführt wird, ein komplettes Bild der Funktionalität der Anwendung liefert.

Zur Ausführung der Anleitung wird eine Topic Map im XTM 1.0 Format benötigt. Es kann die im WWW verfügbare *TMWorld Topic Map* genutzt werden, diese ist unter dem folgenden Link verfügbar:

`http://www.techquila.com/tmsamples/xtm/tmworld.xtm`

Zunächst soll die Anwendung gestartet werden. Dazu muss der Nutzer seinen Web-Browser öffnen, in welchem die Ausführung von JavaScript erlaubt ist. Daraufhin muss der folgende URL in die Adresszeile des Browsers eingegeben werden:

`http://bauhaus.cs.uni-magdeburg.de/wikitm/`

Ist die Seite komplett geladen, so können diese Schritte ausgeführt werden:

1. **Suchanfrage an die *Wikipedia***
   Wurde die Seite geladen, so ist das Suchfeld für Suchanfragen an die Wikipedia im linken Teil der Anzeige zu sehen. Nach einer Eingabe der Zeichenkette ‚Topic' beginnt die Suchunterstützung und zeigt die verfügbaren Artikel an. Innerhalb dieses Feldes kann der Artikel mit der Bezeichnung ‚Topic Map' ausgewählt werden. Mit einem Klick auf den Knopf mit der Aufschrift ‚Submit!' beginnt die Anfrage. Nach wenigen Momenten wird die aus der *Wikipedia* generierte Topic Map visualisiert.

2. **Navigieren innerhalb einer Topic Map**
   Die erste Ansicht der Topic Map ist relativ unübersichtlich, dies ist durch die Anzahl der angezeigten Topics unvermeidbar. Zunächst soll aber das Topic, welches zentralisiert dargestellt wurde und den Titel ‚Topic Map' trägt, genauer untersucht werden. Dazu wird die Maus in das Topic hineinbewegt und dort belassen. Nach einer Sekunde erscheint das sog. *Info Widget*, welches weitere

Informationen zum Topic anzeigt. Es fällt auf, dass die Überschrift eine Hypertext-Referenz trägt. Diese verweist hier auf den *Wikipedia*-Artikel, der zur Erzeugung der Topic Map diente. Nachdem die Maus aus dem Topic (oder dem *Info Widget*) heraus bewegt wurde, verschwinden die zusätzlichen Informationen. Ist der Nutzer mit der Visualisierung unzufrieden, kann er jedes Topic mit der Maus verschieben und so eine eigene Ansicht erstellen.

3. **Nutzung des Assoziationsbrowsers**
   Wie anhand der erzeugten Linien zu sehen, ist das Topic ‚Topic Map' durch verschiedene Assoziationen mit den verknüpften Topics verbunden. Diese Linien und die Spezifizierung der Verknüpfungen können durch die Schalter ‚Show: Associations' und ‚Show: Association Names' aus- und eingeschaltet werden. Interessiert nur die Verknüpfung ‚Weblinks', so kann im Assoziationsbrowser, welcher über der Visualisierung eingeblendet wird, auf diese Verknüpfung geklickt werden. Es erscheint ein Ausschnitt aus der Topic Map, welcher nur die durch diese Verknüpfung verbundenen Topics enthält.

4. **Upload von XTM-Dokumenten**
   Um lokal gespeicherte XTM-Dokumente visualisieren zu können, müssen diese der Anwendung per Upload zur Verfügung gestellt werden. Dazu muss in dem linken Feld der Reiter ‚File-Upload' aktiviert werden. Dann kann auf den Link ‚Upload File' geklickt werden und ein neues Fenster erscheint. Darin ist es möglich, eine XTM-Datei auf dem Computer auszuwählen und über den Knopf ‚Upload!' hochzuladen. Ist dies komplett geschehen, so kann das Fenster über den Link ‚Close Widget' geschlossen werden. Nun steht die Topic Map der Anwendung zur Verfügung.

5. **Nutzen der Drag & Drop-Funktionalität**
   Die Ansicht der Topic Map hat sich durch den Upload nicht verändert. Es ist noch immer die in Punkt 3 erzeugte Ansicht zu sehen. Hier ist das Topic ‚The TAO of Topic Maps' zu finden. Dieses wird mit der Maus auf das Eingabefeld, welches auf der linken Seite zu sehen ist, bewegt und genau darüber losgelassen. Nun wird automatisch in der hochgeladenen Datei nach einem Topic mit diesem Namen gesucht, eine neue Ansicht erscheint. Diese ist Teil der hochgeladenen Topic Map. Natürlich ist auch die Suchunterstützung, wie in Punkt 1 beschrieben, innerhalb der hochgeladenen Topic Map möglich.

4 Fazit und Ausblick

Durch den Einsatz der AJAX-Technologie ist eine Anwendung entwickelt wurden, mit der es weltweit das erste Mal möglich ist, ohne jegliche Installation Topic Maps im XTM-Format graphisch anzuzeigen. Weiterhin wurde durch die *Drag & Drop*-Funktionalität ein effizientes Navigieren innerhalb einer Topic Map oder auch über deren Grenzen hinaus ermöglicht.

Die Einschränkungen, welche der technische Ansatz mit sich bringt, konnten nur teilweise zufriedenstellend gelöst werden. Assoziationen werden durch Linien visualisiert, die durch Graphikvorlagen, welche in einer bestimmten Höhe und Breite gespeichert sind, erzeugt werden. Diese erscheinen aufgrund des Streckens und Stauchens der Originalgraphik nicht einheitlich und erzeugen so den Eindruck verschieden starker Verknüpfungen. Ein Graphik-Framework für Browser, wie erste Ansätze des DojoX GFX[129] oder das Format *SVG* zeigen, könnte hier eine Lösung bieten. Diese sind jedoch nur mit Browsern der neuesten Generation darstellbar, daher muss abgewogen werden, ob eine bessere Darstellung zu Ungunsten der Kompatibilität der Anwendung erfolgen soll.

Generell ist der Ansatz, aus einer Topic Map immer die relevante Teilmenge zum gesuchten Begriff anzuzeigen, im Einklang mit der Theorie der Topic Maps, nach der diese dem menschlichen Denken entsprechen und ihre Inhalte immer aus einer gewissen Sichtweise darstellen. Letztlich dient dies der Übersichtlichkeit, denn der Mensch kann eine Topic Map mit hunderten von Topics und deren Verknüpfungen nicht erfassen. Ein Ausschnitt ist jedoch hilfreich und schnell verinnerlicht. Dem Zweck der Übersichtlichkeit dient auch der Assoziationsbrowser, der aus der relevanten Teilmenge einen weiteren Teil extrahiert und gesondert anzeigt.

Ein grundlegendes Problem stellt beim Upload eines XTM-Dokumentes die darauf folgende Anzeige dar. Da es kein ‚oben und unten' in Topic Maps gibt, ergibt sich auch kein Topic, das generell zur Anzeige geeignet wäre. Aus diesem Grund erzeugt die Anwendung derzeit keine Graphik. Es wäre jedoch sinnvoll, dem Nutzer einige Topics, bspw. diejenigen mit den meisten Verknüpfungen innerhalb des XTM-Dokumentes, anzuzeigen, um ihm den Einstieg in die hochgeladene Topic Map zu ermöglichen.

Serverseitig sollte ein flexibleres Datenbankschema verwendet werden. Das derzeit eingesetzte ist zwar funktional, kann jedoch nicht mit allen, durch die XTM-Spezifikation als valide erklärten, Konstrukten umgehen. So ist eine komplette

[129] Vgl. http://dojotoolkit.org/book/dojo-book-0-9/part-3-programmatic-dijit-and-dojo/drawing-gfx, aufgerufen am 16. September 2008.

Wiederherstellung der Topic Map aus der Datenbank derzeit nicht möglich. Zur Entwicklung kann das ER-Diagramm der TM-Engine[130] herangezogen werden. Weiterhin sollte für jeden Nutzer und jede XTM-Datei eine eigene Tabelle angelegt werden, dies führt zu schnelleren Zugriffszeiten bei vielen Benutzern und beugt der Vermischung von mehreren, durch einen Nutzer hochgeladenen, XTM-Dokumenten vor.

Ein mittelfristiges Ziel der Weiterentwicklung der Anwendung muss die Integration von Gültigkeitsbereichen, den *Scopes* sein. Dies ist derzeit noch nicht implementiert. Auch wenn die Client-Anwendung diese grundsätzlich bereits auslesen kann reagiert die graphische Präsentation nicht auf diese. Der Grund ist die Vielschichtigkeit des Konzeptes; *Scopes* beschreiben Gültigkeitsbereiche, dies können unterschiedliche Sprachen, Nutzerklassen, Zugriffsmedien, Einsatzbereiche, etc. sein. Die Benennung obliegt jedoch dem Erzeuger der Topic Map, so dass eine Zuordnung zu den genannten Kategorien nur teilweise möglich ist und das Programm auf Nutzerangaben bzgl. der Art des Gültigkeitsbereiches angewiesen wäre. Hier muss eine Strategie entwickelt werden, um mit dieser Herausforderung umzugehen.

Auch *mergeMap*-Elemente sollten von der Anwendung beachtet und, falls die verknüpften Topic Maps nicht im WWW zu finden sind, vom Nutzer nach dem Upload eines XTM-Dokumentes abgefordert und dann integriert werden.

Langfristig ist sicherlich eine Weiterentwicklung in Richtung eines Topic Map Editors sinnvoll. Die grundlegenden Methoden sind bereits vorhanden; der *Canvas* kann Topics erzeugen, verknüpfen und löschen. Durch den Einsatz des DOM-Parsers ist auch eine Änderung des geladenen XTM-Dokumentes und die Integration in die serverseitig gespeicherte komplette Topic Map ohne weiteres möglich.

Ein weiterer Einsatzbereich kann eine abgewandelte Form der entwickelten Anwendung als Navigationshilfe in Intranets sein. So könnte die Suchleiste ausgespart werden und ein `XTMObject` auf Basis einer fest innerhalb der Datenbank abgelegten XTM-Datei und der aktuellen Position innerhalb der *Sitemap* (‚Landkarte' eines Web-/Intranet-Auftritts) erzeugt werden. So wäre der Nutzer stets über die Einordnung der Website im Gesamtkonstrukt des Intranets informiert. Dokumente, welche einer bestimmten Kategorie angehören, können über deren Kontext, bspw. Personalwesen, auffindbar sein; in diesem Fall Urlaubsanträge, Zeiterfassungsbögen, etc.

[130] vgl. Widhalm/Mück (2002), S. 247

Anhang

A Anforderungskatalog

Funktionale Anforderungen:

1. Die Anwendung soll Topic Maps, kodiert im XTM-Standard, dynamisch verarbeiten können.
2. Sie soll eine übersichtliche, graphische Darstellung der Topic Map, abhängig von einem Suchbegriff, erzeugen.
3. Weiterhin soll sie Dokumente, die im XTM-Dokument referenziert werden, verfügbar machen.
4. Sie soll Datenquellen, die XTM-Dokumente generieren, einbinden können.
5. Die Anwendung soll es bei der Darstellung ermöglichen, nur bestimmte Assoziationen einzublenden, da dies die Übersichtlichkeit erhöht.
6. Die Anwendung soll eine Funktionalität bieten die Suchanfragen der Vergangenheit (History), ohne erneuten Kontakt zum Server, wiederherzustellen.
7. Die Anwendung soll den Benutzer bei der Suche innerhalb von Datenquellen unterstützen.
8. Die Anwendung soll die folgenden Details der Topic Map darstellen können: Occurrences, Typisierungen, Assoziationen und Topics. Scopes und Verschmelzungen von Topic Maps können zu einem späteren Zeitpunkt hinzugefügt werden.

Qualitative Anforderungen:

9. Die Software soll intuitiv zu bedienen sein und eine einfache Nutzeroberfläche bieten.
10. Die Anwendung soll stabil und fehlertolerant sein.
11. Auf Suchanfragen soll die Anwendung in einem vertretbaren Zeitraum antworten.

12. Die Anwendung soll plattformunabhängig sein.

13. Die Software soll ohne Installation ablauffähig sein, das heißt auch keine Komponenten benötigen, die gesondert installiert werden müssen.

14. Die Navigation innerhalb einer Topic Map soll benutzerfreundlich und einfach sein.

15. Die Darstellung von XTM-Dokumenten soll, in Bezug auf die dargestellten Details, vollständig sein.

16. Die vom Server gehaltenen Daten sollen nach einer gewissen Zeit verfallen, um Speicherplatz freizugeben und die Anwendung nicht zu verlangsamen.

Systembezogene Anforderungen:

17. Die Anwendung soll mittels der Client-Server-Architektur umgesetzt werden.

18. Die Visualisierung soll im Web-Browser erfolgen.

19. Alle gängigen Web-Browser sollen die Anwendung ausführen können, ohne zusätzliche Plugins zu benötigen.

20. Die Serverkomponente soll möglichst plattformunabhängig und austauschbar sein.

Prozessspezifische Anforderungen:

21. Die Software soll innerhalb der Bearbeitungszeit für eine Diplomarbeit, die 5 Monaten entspricht, entwickelt werden.

22. Die Anwendung soll von einem Entwickler, dem Verfasser der Arbeit, erstellt werden.

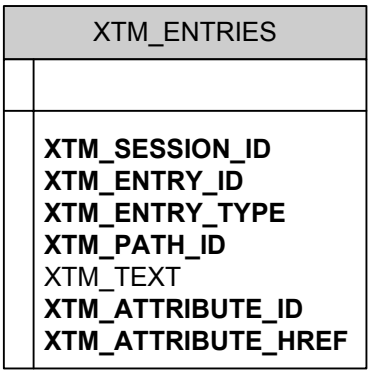
B Client-Konfigurationsdatei config.xml (Beispiel)

```
<?xml version="1.0" encoding="UTF-8"?>
<config>
	<general>
		<title>AJAX Topic Map BETA</title>
		<canvas-parent>main</canvas-parent>
		<navigation-parent>nav</navigation-parent>
		<navigation-parent2>nav2</navigation-parent2>
		<max-topics-to-show>250</max-topics-to-show>
	</general>
	<pictures>
		<ajax-load-small>img/ajax-loader.gif</ajax-load-small>
		<ajax-load-big>img/ajax-loader-big.gif</ajax-load-big>
		<line-horizontal>img/line_horiz.png</line-horizontal>
		<line-vertical>img/line_vertik.png</line-vertical>
		<line-upper-left-lower-right>img/line_ul-lr.png</line-
upper-left-lower-right>
		<line-upper-right-lower-left>img/line_ur-ll.png</line-
upper-right-lower-left>
	</pictures>
	<source>
		<name>Wikipedia</name>
		<submit-button>submit_wiki</submit-button>
		<input-form>searchtopic_wiki</input-form>
		<url-autocomplete>AutoComplete</url-autocomplete>
		<autocomplete-param>searchwiki</autocomplete-param>
		<autocomplete-indicator>indicator2</autocomplete-indicator>
		<autocomplete-div>autocomplete_choices2</autocomplete-div>
		<url-xtmrequest>XTMServlet</url-xtmrequest>
		<param>
			<value>WIKI</value>
		</param>
	</source>
	<source>
		<name>File-Upload</name>
		<submit-button>submit_db</submit-button>
		<input-form>searchtopic_db</input-form>
		<url-autocomplete>AutoComplete</url-autocomplete>
		<autocomplete-param>searchdb</autocomplete-param>
		<autocomplete-indicator>indicator1</autocomplete-indicator>
		<autocomplete-div>autocomplete_choices</autocomplete-div>
		<url-xtmrequest>XTMServlet</url-xtmrequest>
		<param>
			<value>DB</value>
		</param>
	</source>
</config>
```

C Datenbankerzeugung (SQL) und Datenbankmodelldiagramm

```
CREATE TABLE  `XTM`.`XTM_ENTRIES` (
  `XTM_SESSION_ID` text NOT NULL,
  `XTM_ENTRY_ID` int(11) NOT NULL,
  `XTM_ENTRY_TYPE` int(11) NOT NULL,
  `XTM_PATH_ID` int(11) NOT NULL,
  `XTM_TEXT` text character set utf8,
  `XTM_ATTRIBUTE_ID` varchar(255) NOT NULL default '',
  `XTM_ATTRIBUTE_HREF` varchar(255) NOT NULL default ''
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
```

Das zugehörige Datenbankmodellschema:



D HashMap zur Erzeugung der XTM_PATH_ID

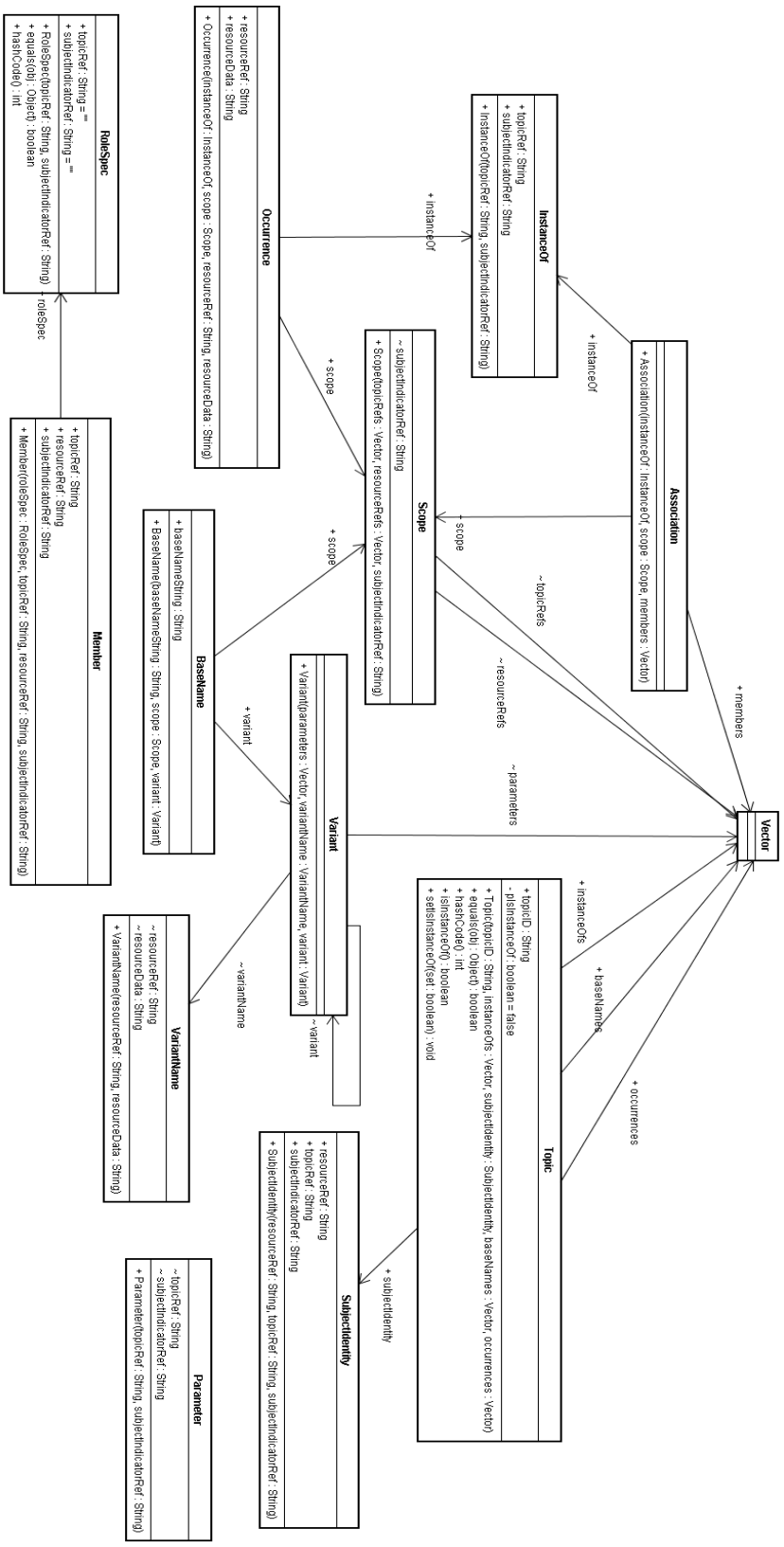
```
hashMapOfXMLPathes = new HashMap();
hashMapOfXMLPathes.put("/topicMap", "1");
hashMapOfXMLPathes.put("/topicMap/association", "2");
hashMapOfXMLPathes.put("/topicMap/association/instanceOf", "3");
hashMapOfXMLPathes.put("/topicMap/association/instanceOf/topicRef",
        "4");
hashMapOfXMLPathes.put("/topicMap/association/member", "5");
hashMapOfXMLPathes.put("/topicMap/association/member/roleSpec", "6");
hashMapOfXMLPathes.put("/topicMap/association/member/roleSpec/topicRef
        ", "7");
hashMapOfXMLPathes.put("/topicMap/association/member/topicRef", "8");
hashMapOfXMLPathes.put("/topicMap/association/scope", "9");
hashMapOfXMLPathes.put("/topicMap/association/scope/topicRef", "10");
hashMapOfXMLPathes.put("/topicMap/mergeMap", "11");
hashMapOfXMLPathes.put("/topicMap/topic", "12");
hashMapOfXMLPathes.put("/topicMap/topic/baseName", "13");
hashMapOfXMLPathes.put("/topicMap/topic/baseName/baseNameString",
        "14");
hashMapOfXMLPathes.put("/topicMap/topic/baseName/instanceOf", "15");
hashMapOfXMLPathes.put("/topicMap/topic/baseName/instanceOf/topicRef",
        "16");
hashMapOfXMLPathes.put("/topicMap/topic/baseName/scope", "17");
hashMapOfXMLPathes.put("/topicMap/topic/baseName/scope/topicRef",
        "18");
hashMapOfXMLPathes.put("/topicMap/topic/baseName/variant", "19");
hashMapOfXMLPathes.put("/topicMap/topic/baseName/variant/parameters",
        "20");
hashMapOfXMLPathes.put("/topicMap/topic/baseName/variant/parameters/to
        picRef", "21");
hashMapOfXMLPathes.put("/topicMap/topic/baseName/variant/variantName",
        "22");
hashMapOfXMLPathes.put(
        "/topicMap/topic/baseName/variant/variantName/resourceData",
        "23");
hashMapOfXMLPathes.put("/topicMap/topic/instanceOf", "24");
hashMapOfXMLPathes.put("/topicMap/topic/instanceOf/topicRef", "25");
```

```
hashMapOfXMLPathes.put("/topicMap/topic/occurrence", "26");
hashMapOfXMLPathes.put("/topicMap/topic/occurrence/instanceOf", "27");
hashMapOfXMLPathes.put("/topicMap/topic/occurrence/instanceOf/topicRef
          ", "28");
hashMapOfXMLPathes.put("/topicMap/topic/occurrence/resourceData",
          "29");
hashMapOfXMLPathes.put("/topicMap/topic/occurrence/resourceRef",
          "30");
hashMapOfXMLPathes.put("/topicMap/topic/occurrence/scope", "31");
hashMapOfXMLPathes.put("/topicMap/topic/occurrence/scope/topicRef",
          "32");
hashMapOfXMLPathes.put("/topicMap/topic/subjectIdentity", "33");
hashMapOfXMLPathes.put("/topicMap/topic/subjectIdentity/subjectIndicat
          orRef", "34");
```

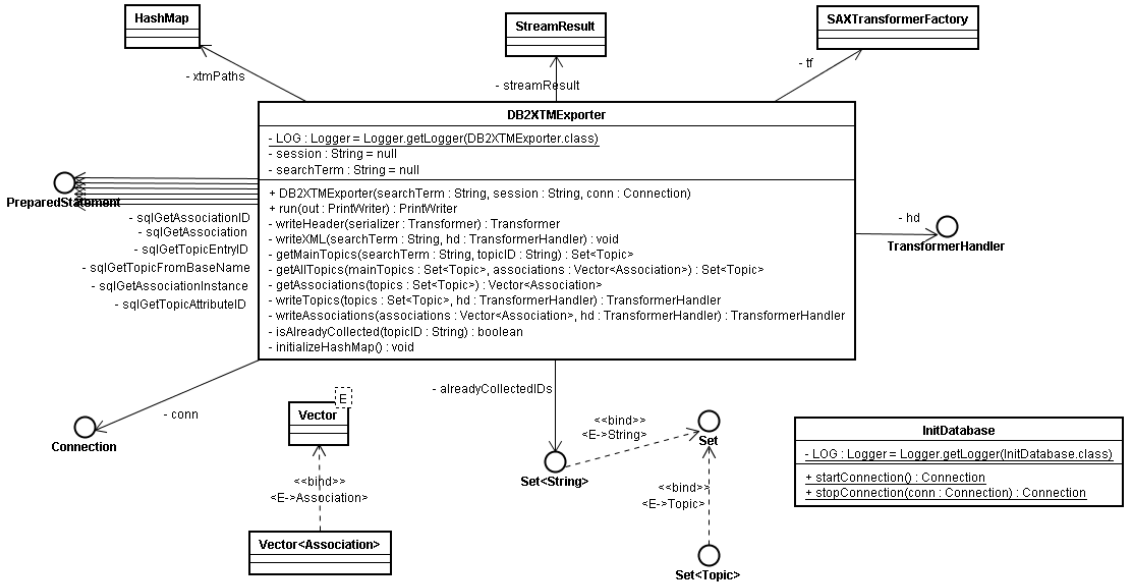
E Beispiel eines XTM-Dokumentes

```
<topicMap id="beispiel-topicMap"
      xmlns="http://www.topicmaps.org/xtm/1.0/"
      xmlns:xlink="http://www.w3.org/1999/xlink">
        <topic id="henry-iv">
          <instanceOf>
               <topicRef xlink:href="#play"/>
          </instanceOf>
          <baseName>
               <baseNameString>
                         King+Henry+IV
               </baseNameString>
          </baseName>
        </topic>
        <topic id="falstaff">
          <instanceOf>
               <topicRef xlink:href="#opera"/>
          </instanceOf>
          <baseName>
               <baseNameString>Falstaff</baseNameString>
          </baseName>
          <occurrence>
               <instanceOf>
                    <topicRef xlink:href="#premiere-date"/>
               </instanceOf>
               <resourceData>1893-02-09</resourceData>
          </occurrence>
        </topic>
        <association>
          <instanceOf>
               <topicRef xlink:href="#based-on"/>
          </instanceOf>
          <member>
               <topicRef xlink:href="#henry-iv"/>
          </member>
          <member>
               <topicRef xlink:href="#falstaff"/>
          </member>
        </association>
        [...]
</topicMap>
```

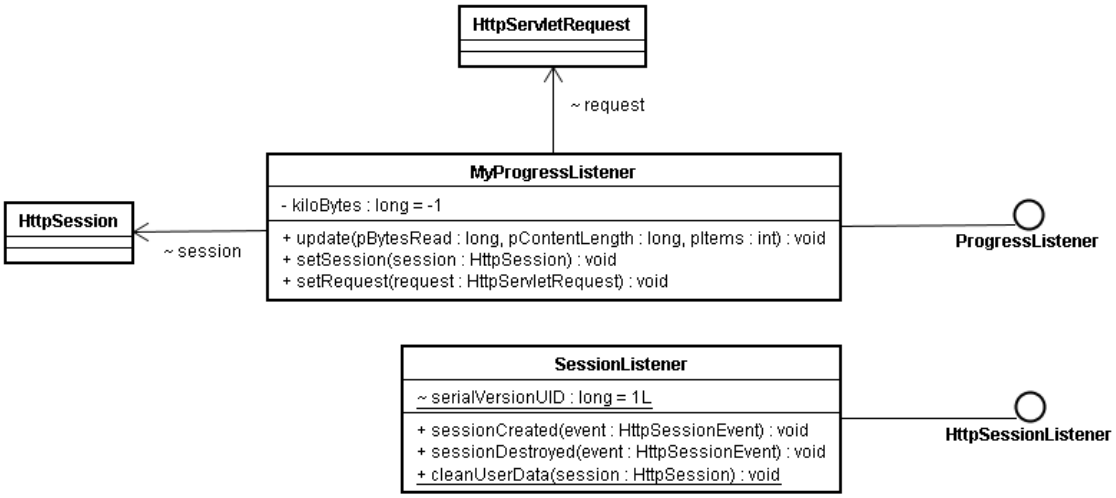
F Klassendiagramm: Java-Klassen zur Abbildung der XTM-Spezifikation



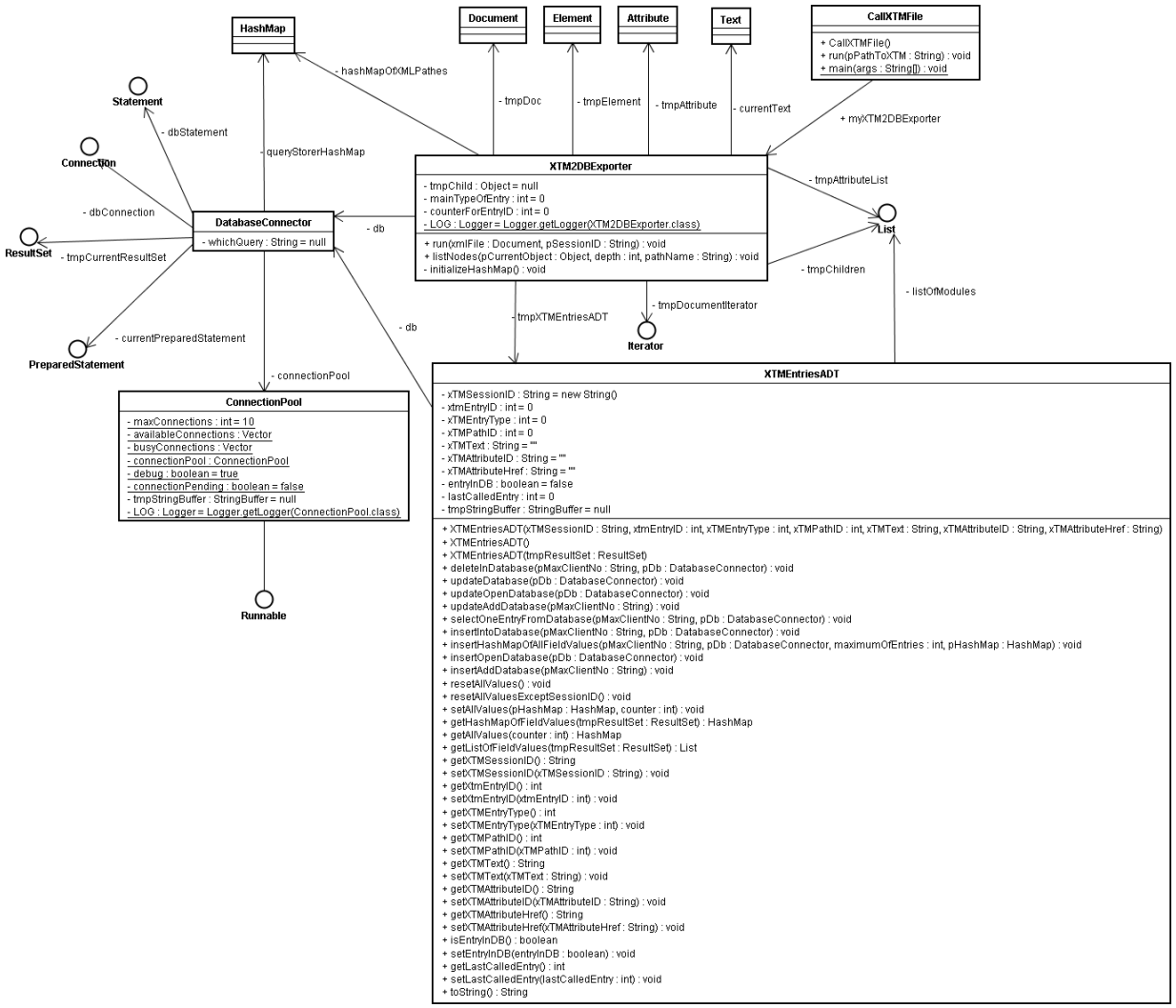
G Klassendiagramm: DB2XTMExporter und Hilfsklassen



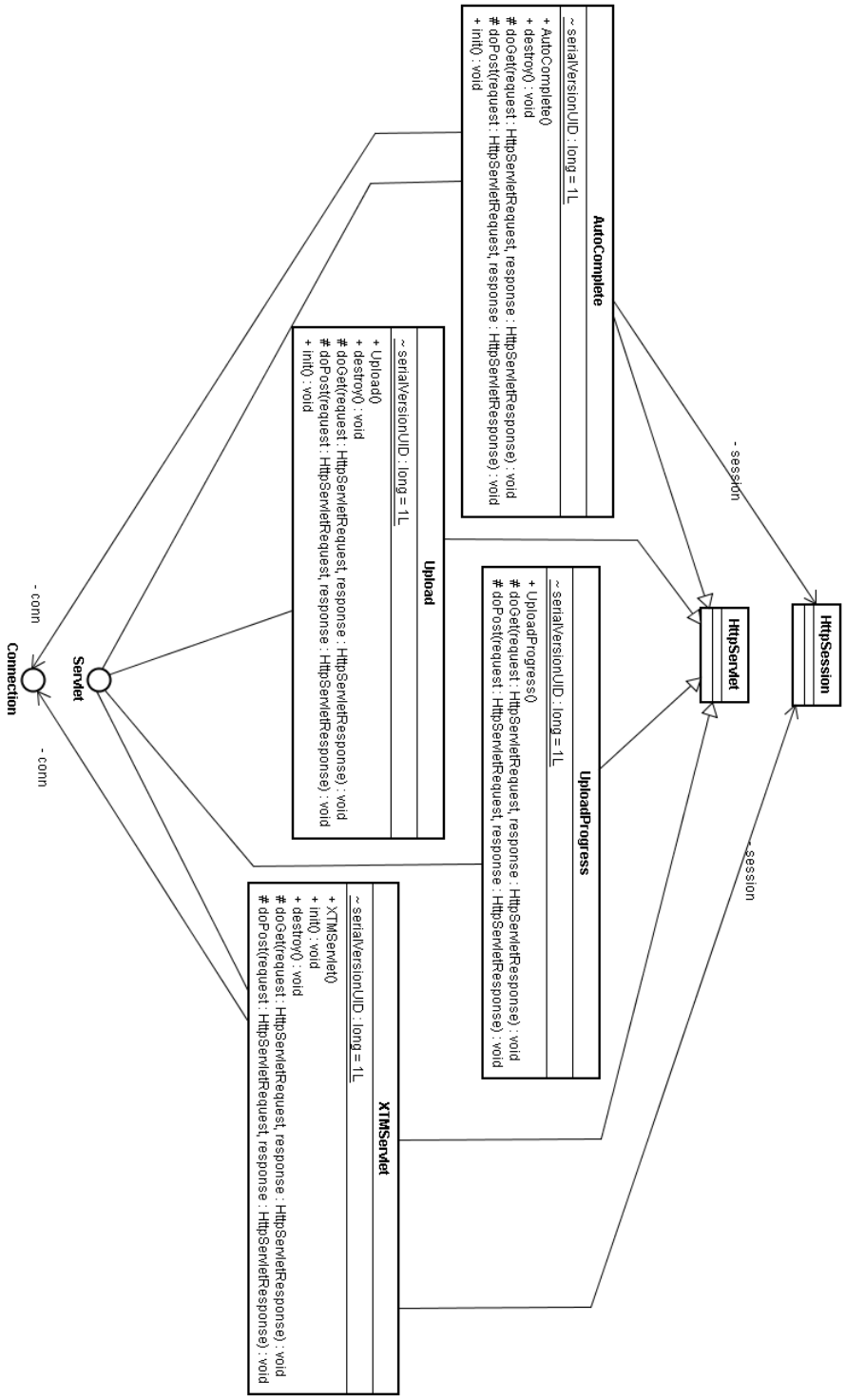
H Klassendiagramm: Session-/ProgressListener



I Klassendiagramm: XTM2DBExporter und Hilfsklassen



J Klassendiagramm: Servlets



Literaturverzeichnis

Ahmed, K.; Garshol, L. M.; Grønmo, G. O.; Heuer, L.; Lischke, S.; Moore, G. (2008): TMAPI - Common Topic Map Application Programming Interface. http://www.tmapi.org. 15. September 2008.

Beier, H. (2006): Betriebliches Wissensmanagement: Rollen, Prozesse, Instrumente. In: Pellegrini, T.; Blumauer, A. (2006) S. 257-272

Birkenbihl, K. (2006): Standards für das Semantic Web. In: Pellegrini, T.; Blumauer, A. (2006), S. 73-88.

Crane, D.; Pascarello, D.; James, D. (2006): Ajax in Action. Greenwich

Dipper, S. (2008): XML Grundlagen: Elemente, Attribute. http://www.ling.uni-potsdam.de/~dipper/teaching/ws04_xml/xml_basics.pdf. 30. Juli 2008.

Dumke, R. (2001): Software Engineering. 3. Aufl., Braunschweig-Wiesbaden.

Ebner, M. (2002): SQL lernen. München.

El Moussaoui, H.; Zeppenfeld, K. (2008): AJAX. Heidelberg

Hunter, J. (2008): JDOM. http://www.jdom.org. 10. September 2008.

Kienreich, W.; Strohmaier, M. (2006): Wissensmodellierung: Basis für die Anwendung semantischer Technologien. In: Pellegrini, T.; Blumauer, A. (2006) S. 359-372

Kobligk, M. (2008): Entwicklung und Nutzen von XML. http://www.fhwedel.de/~si/seminare/ss04/Ausarbeitung/9.Kobligk/datenaustausch.html. 29. Juli 2008.

Kohonen, T. (1995): Self Organizing Maps. Berlin et al.

Krings, H. P. (1996) Wissenschaftliche Grundlagen der technischen Kommunikation. Tübingen

Löhnertz, M. (2008): Graphentheorie. http://www.graphentheorie.de/Glossar.html. 9. September 2008.

Marchal, B. (2008): Tip: Using SAXTransformerFactory.
http://www.ibm.com/developerworks/xml/library/x-ipsxtf/?open&l=136,t=grx.
31. Juli 2008.

Niemann, K. D. (1995) Client/Server-Architektur, Organisation und Methodik der Anwendungsentwicklung. Braunschweig.

North, S.; Hermans, P. (2000): XML in 21 Tagen. München.

Ontopia (2008): Omnigator: The Topic Map Browser.
http://www.ontopia.net/omnigator/docs/navigator/userguide.html.
21. August 2008.

Pellegrini, T.; Blumauer, A. (Hrsg.) (2006): Semantic Web. Berlin u. a.

Pepper, S.; Moore, G. (2008): XML Topic Maps (XTM) 1.0.
http://www.topicmaps.org/xtm/index.html. 29. Juli 2008.

punkt. netServices (2008): Was sind Topic Maps?
http://www.topic-maps.at/topicmap.html. 15. August 2008.

Rotter, S.; Tremmel, A.; Hametner, S.; Jandl, W.; Eschlböck, M.; Gruber, R. (2008): Client-Server-Architektur. http://gd.tuwien.ac.at/study/hrh-glossar/1-2_17.htm.
30. Juli 2008.

Saake, G.; Sattler, K.-U. (2000): Datenbanken & Java: JDBC, SQLJ und ODMG.
Heidelberg

Steffen, R.; Staschinski, C.; Rössig, M. (2008): Glossar eLearning.
http://www.laum.uni-hannover.de/elearning/arbeitshilfen/glossar.htm.
30. Juli 2008.

Stephenson, S. (2008): Prototype 1.6. http://www.prototypejs.org/api.
13. August 2008.

Sun Microsystems, Inc (2008): Developer Resources for JAVA Technology.
http://java.sun.com. 01. August 2008.

Tochtermann, K.; Maurer, H. (2006): Semantic Web - Geschichte und Ausblick einer Vision. In: Pellegrini, T.; Blumauer, A. (2006) S. 1-6

Twele, L. (2008): Transformierung von HTML-Daten in eXtensible Topic Maps zur Visualisierung von Informationen am Beispiel des Online-Lexikons Wikipedia. Diplomarbeit, Otto-von-Guericke-Universität. Magdeburg.

Universität Zürich (2008): ZInfo Internet: Glossar. http://www.id.unizh.ch/cl/zinfo/old/glossar.html. 28. Juli 2008.

Völkel, M. (2008): Vortrag 12: Das Grafikdateiformat PNG. http://goethe.ira.uka.de/seminare/redundanz/vortrag12. 08. September 2008.

W3C (2008a): XHTML™ 1.0 The Extensible HyperText Markup Language (Second Edition). http://www.w3.org/TR/xhtml1/Cover.html. 29. Juli 2008.

W3C (2008b): Predefined Entities. http://www.w3.org/TR/REC-xml/#sec-predefined-ent. 30. Juli 2008.

Wendt, S. (1997): Terminus - Thesaurus - Text. Tübingen.

Wenz, C. (2007): JavaScript und AJAX. Bonn.

Widhalm, R.; Mück, T. (2002): Topic Maps. Berlin.

Wöhr, H. (2004): Web-Technologien. Heidelberg.

Ziegler, C. (2001): Gralssuche - JavaScript und Objektorientierung. In: iX, Nr. 04/2001. S. 194

Abschließende Erklärung

Ich versichere hiermit, dass ich die vorliegende Diplomarbeit selbständig, ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht.

Magdeburg, den 20. Oktober 2008