Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Diplomarbeit

Petra Rohwer-Ehlers

Ein Netzmodell für das Dokumentenmanagement im Collaborative Workplace

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Petra Rohwer-Ehlers

Ein Netzmodell für das Dokumentenmanagement im Collaborative Workplace

Diplomarbeit eingereicht im Rahmen der Diplomprüfung
im Studiengang Softwaretechnik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer : Prof. Dr. rer. nat. Jörg Raasch
Zweitgutachter : Prof. Dr. rer. nat. Kai von Luck

Abgegeben am 21. Juni 2007

Petra Rohwer-Ehlers

Thema der Diplomarbeit

Ein Netzmodell für das Dokumentenmanagement im Collaborative Workplace

Stichworte

Collaborative Workplace, Dokumentenmanagement, Metadaten, Semantische Netze, Topic Maps, Wissensrepräsentation

Kurzzusammenfassung

Thema der vorliegenden Arbeit ist die Entwicklung eines Wissensnetzes als anwendungsunabhängiges Dokumentenmanagementsystem zur Verbesserung computerunterstützter Gruppenarbeit in einem Konferenzraum (Collaborative Workplace). Die im Zuge einer Gruppenarbeit anfallenden relevanten Informationen und Dokumente sollen an einem zentralen Ort zusammengefasst werden. Die Aussagekraft bezüglich der Relevanz eines Dokuments in einem bestimmten Kontext, soll durch die flexible Anreicherung mit zusätzlichen Informationen (Metadaten) erhöht werden. Darüber hinaus erfolgt eine semantische Verknüpfung inhaltlich zusammenhängender Dokumente nach dem Hyperlinkprinzip. Realisiert wurde die Anwendung als Web-Applikation auf Basis von Topic Maps.

Petra Rohwer-Ehlers

Title of the paper

A simulated network for the document management in a collaborative workplace

Keywords

collaborative workplace, metadata, semantic net, Topic Maps, knowledge repräsentation

Abstract

Subject of the presented paper is the development of a knowledge net as application-independent document management system to improve of computer-aided teamwork in a conference room (collaborative workplace). The relevant information and documents resulting in teamwork are to be summarized at a central place. The expressiveness concerning the relevance of a document in a certain context is to be increased by the flexible enrichment with additional information (metadata). Furthermore a semantic linkage of contentwise connected documents takes place according the hyperlink principle. Application was realized as web application on basis of Topic Maps.

Inhaltsverzeichnis

Inhaltsverzeichnis

1. Einleitung

1.1. Motivation

Im Rahmen des Projekts „Collaborative Workplace“ im Department Informatik der *Hochschule für angewandte Wissenschaft*en in Hamburg, kurz HAW, wurde ein Gruppenarbeitsraum eingerichtet. Dieser Raum soll im Zuge von Diplom-, Bachelor- und Masterarbeiten mit Computertechnik und Software zur computerunterstützten Gruppenarbeit ausgerüstet werden.

Ziel ist der Aufbau einer kollaborativen Arbeitsumgebung, die verschiedenen Projektteams als Plattform für die gemeinsame Arbeit dienen soll. Die Arbeitsumgebung hat das vorrangige Ziel, die gemeinschaftliche Arbeit zu fördern und effizienter zu gestalten.

Maßgeblich bei der technischen Umsetzung ist dabei der von Mark Weiser geprägte Begriff des *Ubiquitous Computing*, wonach sich verschiedene vorrangig mobile Computer (Laptops, PDAs, Mobiltelefone, usw.) nahtlos in die allgemeine Umgebung integrieren [Burf 2006].

Aspekte der technischen Umsetzung wurden bereits in den Arbeiten [Bart 2006], [Mund 2006] und [Burf 2006] analysiert und prototypisch implementiert. Im Fokus dieser Arbeit steht die Entwicklung einer Benutzungsschnittstelle[1] und die Entwicklung eines Modells zur Organisation und Repräsentation des in einer Arbeitsgruppe vorhandenen Wissens.

We should no longer ask whether we have enough information,
we should rather ask if we can manage the information we have.

Leopold Sauermann

1.2. Problemstellung

Elementarer Bestandteil jeder Gruppenarbeit sind Projektbesprechungen, Diskussionen und Präsentationen. Gemeinsam werden Ideen entwickelt, Lösungen ausgearbeitet, Aufgaben verteilt und Beschlüsse gefasst. Hierbei entsteht eine Vielzahl unterschiedlicher Dokumente, wie Protokolle, Notizen, Skizzen, Dokumentationen, Ablaufpläne, Kostenschätzungen, Use-Cases, Glossare, Diagramme etc.

Zur Erzeugung und Bearbeitung all dieser Dokumente stehen den „Büroarbeitern“ heute komfortable Werkzeuge zur Verfügung. Diese sind jedoch vornehmlich für die Einzelplatzarbeit ausgelegt. Gruppenarbeit wird von diesen Systemen meist nur mäßig oder gar nicht unterstützt. Zudem fehlen in der Regel Strategien zur anwendungsübergreifenden Nutzung der Daten.

---

[1] Die Summe interaktiver Eingriffsmöglichkeiten für Benutzer wird als Benutzungsschnittstelle bezeichnet [Stay 1996]

Darüber hinaus finden sich in herkömmlichen Seminar- und Besprechungsräumen eine Reihe etablierter technischer Hilfsmittel, die Gruppenarbeit auf unterschiedliche Art und Weise unterstützen. Hierzu zählen Overheadprojektor, Tafeln, Whiteboard, Flipchart u. a., wobei der Overheadprojektor vieler Orts bereits durch den Beamer abgelöst wurde, so dass die Präsentation digitaler Dokumente möglich ist.

Der Einsatz dieser Geräte bringt neben Vorteilen auch zahlreiche Probleme mit sich. Problematisch ist hierbei zum Beispiel, dass sich „analoge" Tafelbilder schlecht in die ansonsten „digitale" Arbeitswelt einbinden lassen. Weitere Schwierigkeiten treten auf, wenn es darum geht, an Tafeln oder Whiteboards erzeugte Notizen zu archivieren, zu bearbeiten, wiederzuverwenden oder zur Info an andere Personen zu verteilen.

Ausführlich wurde diese Problematik, einschließlich möglicher Lösungsansätze, in den Arbeiten von Carola Neumann [Neu 2006] und Lars Burfeindt [Burf 2006] behandelt. Im Fokus dieser Arbeit soll jedoch eine andere Thematik stehen – die Repräsentation sowie die inhaltliche Beschreibung und Verknüpfung vorhanden Wissens auf Metaebene. Dieses Wissen existiert in Unternehmen und Organisationen bereits heute in großen Mengen von Daten, abgelegt in Datenbanken, Office-Dokumenten oder E-Mails.

Viele dieser Dokumente und Daten stehen inhaltlich in einem engen Zusammenhang. Doch anders als bei Internetseiten, die per Hyperlink miteinander verknüpft werden können, ist eine Verknüpfung zwischen den heterogenen Dokumenttypen von Desktop-Applikationen nicht ohne weiteres möglich. Hier erfolgt allenfalls eine Strukturierung der Daten über die Ablage in einem hierarchischen Dateisystem. Eine anwendungs- und prozessübergreifende Verknüpfung von Informationen wird hierdurch jedoch nicht erreicht.

Im folgenden wird dieser Arbeit ein Anwendungskontext zugrunde gelegt, der davon ausgeht, dass der Raum überwiegend als Arbeitsraum zur Entwicklung innovativer Konzepte und weniger als reiner Vortragssaal genutzt wird. Konkret soll das bedeuten, dass hier über einen längeren Zeitraum Projekte abgewickelt werden. In diesem Fall ist es wahrscheinlich, dass Teammitglieder von Zeit zu Zeit wechseln.

Für neu hinzukommende Personen ist es oft sehr schwer, sich in ein laufendes Projekt einzuarbeiten, selbst wenn sie über ein zentrales Netzwerk Zugriff auf alle relevanten Daten haben. Aufgrund der Fülle vorhandener Dokumente und fehlender Strukturen, die Zusammenhänge verdeutlichen, kann die Einarbeitung in ein laufendes Projekt so einige Zeit in Anspruch nehmen.

Ein weiteres Problem entsteht, wenn benötigte Informationen erst gar nicht verfügbar sind, weil sie nicht zentral sondern verstreut auf lokalen Festplatten oder mobilen Endgeräten, wie Notebooks, PDAs und Handys gespeichert sind. Dies könnte gerade bei den für diesen Arbeitsraum anvisierten, bei [Neu 2006][2] und [Burf 2006][3] beschriebenen, Anwendungsszenarien der Projektbesprechungen und Präsentationen zum Tragen kommen.

---

[2] Beispielszenario: Softwareentwicklungsprojekte
[3] Beispielszenario: Stadtplanung

Insbesondere wenn sich Personen aus unterschiedlichen Organisationen zu einem Projekt zusammen finden, kann es leicht passieren, dass Dokumente auf mitgebrachten Notebooks oder PDAs liegen und Kollegen so der Zugriff auf die dort enthaltenen Informationen verwehrt ist. Ein Wiederfinden bestimmter Informationen ist dann häufig unmöglich. Unter Umständen fehlt sogar das Wissen um die Existenz dieser Dokumente.

Gesamtziel des Projekts „Collaborative Workplace“ ist der Aufbau einer kollaborativen Arbeitsumgebung, die Projektteams aus unterschiedlichen Fachgebieten als Plattform für die gemeinsame Arbeit dienen soll. Die Arbeitsumgebung hat das vorrangige Ziel, die gemeinschaftliche Arbeit zu fördern und sie effizienter und nachhaltiger zu gestalten. Zudem sollen in diese Umgebung die Vorteile des klassischen Besprechungsraumes einfließen, ohne dabei neue Probleme zu schaffen [Burf 2006].

Die Schlüsselkomponenten des Gruppenarbeitsraumes bilden aus technologischer Sicht eine Reihe hochauflösender, etwa 42” großer, berührungssensitiver TFT-Wandbildschirme. Aufgrund der Größe sind sie bei Wahl entsprechender Schriftgröße von mehreren Personen auch aus einigen Metern Entfernung gut einsehbar. Die positiven Eigenschaften von Whiteboard und Beamerprojektion sind somit im Hinblick auf Gruppenarbeit ausreichend abgedeckt, weniger jedoch im Hinblick auf Vortragsveranstaltungen.

Aufgrund der Touchfunktionalität eignen sie sich zudem gut als Eingabegerät für alle mit der Maus durchführbaren Computerinteraktionen. Mit entsprechender Software ist auch handschriftliches Schreiben und das Erstellen von Skizzen an den Bildschirmen möglich. Somit sind weitere Vorteile von Whiteboard und Flipchart abgedeckt.

Weitere technische Funktionalitäten werden darin bestehen, dass mitgebrachte mobile Endgeräte, z. B. Notebooks, PDAs und Mobiltelefone, über Funktechnologien[4] mit den Geräten des Raums interagieren können. Außerdem werden ein Authentifizierungsmodul und ein Positionierungssystem [Herz 2007] integriert. Außer durch technische Komponenten, soll die Zusammenarbeit im Gruppenarbeitsraum durch eine Anwendung zur Verwaltung der anfallenden digitalen Dokumente unterstützt werden.

1.3. Zielsetzung

Zur Verwaltung digitaler Dokumente unterschiedlichen Formates werden im allgemeinen Dokumentenmanagementsysteme (DMS) eingesetzt. Je nach Einsatzgebiet und Implementierungsstand bieten professionelle Dokumentenmanagementsysteme unterschiedliche Funktionen wie:

- Archivierung digitaler Dokumente, d. h. „sehr“ sichere und langfristige Aufbewahrung (meist sind gesetzliche Vorschriften zu beachten)
- Reglementierung von Zugriff, Verteilung sowie Regelung von Arbeitsabläufen
- Katalogisierung, Kategorisierung, Indizierung und Verschlagwortung der archivierten Dokumente
- Versionierung zur Darstellung des Lebenszyklusses eines Dokuments
- Suchmöglichkeiten über die Ablagestruktur, über Metadaten oder Volltext-indiziert

[4] Drahtlose digitale Verbindung zur Datenübertragung, z. B. WLAN oder Bluetooth

1. Einleitung 4

Im Rahmen dieser Arbeit wird der Ansatz verfolgt, relevante digitale Dokumente an einem zentralen Ablageort zusammenzufassen, mit Metadaten zu beschreiben und die entstehenden „Knoten“ über attributierte Links zu einem Netz zu verbinden. Hier geht es ausdrücklich nicht darum, digitale Massendokumente wie Rechnungen oder Bestellungen zu archivieren. Stattdessen soll erreicht werden, den Informationsgehalt jedes einzelnen Dokuments durch die Anreicherung mit Metadaten und die Verknüpfung mit anderen Dokumenten zu steigern.

Kern dieser Arbeit ist der konzeptionelle Entwurf und die prototypische Entwicklung einer Benutzungsschnittstelle. Die Benutzungsschnittstelle soll den Nutzer unterstützen und befähigen, ein sogenanntes Informationsnetz, oder auch Wissenslandkarte genannt, aufzubauen und zu nutzen.

Die Anwendung soll die inhaltliche Verknüpfung beliebiger Dokumenttypen nach dem Hyperlinkprinzip unterstützen. Ein solches Netz, bestehend aus Knoten und Kanten, muss auf anschauliche Weise die Beziehungen zwischen inhaltlich zusammenhängenden Dokumenten wiedergeben.

Die zu entwickelnde Anwendung ist so anwendungsneutral zu entwerfen und in die technische Umgebung des Gruppenarbeitsraums einzubinden, dass sie von jeder Arbeitsgruppe genutzt werden kann. Das Öffnen und Bearbeiten der gängigen Dokumenttypen soll, sofern entsprechende Installationen vorhanden sind, sowohl an den öffentlich sichtbaren Wandbildschirmen, als auch nach erfolgtem Download, auf den mobilen Endgeräten möglich sein.

Funktionen, wie das Betrachten von Bildern oder Bearbeiten einfacher Textdateien können mit direkt im System implementierten Komponenten, wie Texteditor oder Viewer erfolgen. Komplexere Aufgabenstellungen, wie Tabellenkalkulation, Präsentations- oder Projektplanerstellung sollten über Drittanwendungen in das Gesamtsystem integrierbar sein.

Die Problematik ist prinzipiell nicht neu und seit langem erkannt. Für das WWW wird seit geraumer Zeit an Standards und Techniken, bekannt unter dem Begriff „Semantic Web“, gearbeitet. Und seit einiger Zeit schon gibt es Entwicklungen, diese Ansätze als „Semantic Desktop“ auf den Computerarbeitsplatz zu übertragen. Erreicht werden soll, den einzelnen Menschen zuerst mit seinen eigenen Daten in Einklang zu bringen, bevor darauf aufbauend Kommunikation und Zusammenarbeit in sozialen Netzwerken (dienstlich oder privat) verbessert werden kann [SWSch-b]. In Kapitel 4 werden einige dieser Projekte vorgestellt.

Diese Ansätze basieren in der Regel auf öffentlichen Standards. Einer dieser Standards ist Topic Maps, der in Kapitel 2 ausführlicher beschrieben und später als Basistechnologie eingesetzt wird.

1.4. Aufbau der Arbeit

Um den späteren Diskurs zu erleichtern, erfolgt in Kapitel 2 (Grundlagen) eine Einführung zum Thema Netzmodellierung und Metadaten. Hierzu zählt unter anderem die Vorstellung der später verwendeten Standards wie Dublin Core, Topic Maps und SVG.

Die Vision des KORA-Dokumentenmanagementsystems ist Gegenstand von Kapitel 3. Dieses Kapitel vermittelt einen detaillierteren Einblick in die Netzmodellierung sowie in Verfahren wie Navigation im Netz und Visualisierung. Des Weiteren liefert dieses Kapitel einen Überblick über die weiteren funktionalen und nicht-funktionalen Anforderungen an das KORA-Dokumentenmanagementsystem.

Im anschließenden Kapitel 4 (Existierende Lösungen), werden einige, z. T. noch in Entwicklung befindliche Projekte vorgestellt, die sich mit einem ähnlichen Thema, nämlich der Entwicklung so genannter PIM-Systeme[5], beschäftigen. Hierbei geht es im wesentlichen um die semantische Vernetzung persönlicher Informationen, die das Auffinden vorhandener Informationen auf dem eigenen PC erleichtern sollen.

In Kapitel 5 (Design und Realisierung) folgt, basierend auf den in Kapitel 3 identifizierten Anforderungen, die schrittweise Herleitung und Beschreibung der Softwarearchitektur sowie die Vorstellung der verwendeten Technologien.

Auf Einzelheiten der Implementierung wird in Kapitel 6 (Prototyp) näher eingegangen. Die wichtigsten Abläufe und Funktionen werden hier mit Hilfe von Usecases und Screenshots eingehender beschrieben.

Kapitel 7 soll einen abschließenden Einblick in weitere Einsatzmöglichkeiten und die Potentiale einer semantischen Beschreibung von Inhalten mit Topic Maps vermitteln. Im letzten Kapitel erfolgt dann eine Zusammenfassung der Arbeit sowie ein Ausblick auf mögliche Weiterentwicklungen des Prototypen hin zum Produktivsystem.

[5] PIM = Personal Information Management

2. Grundlagen

In diesem Kapitel erfolgt eine Beschreibung der für den Entwurf der Softwarelösung für den Collaborative Workeplace relevanten Begriffe und Basistechnologien.

2.1. Ontologie

In der Informatik versteht man unter einer Ontologie die konzeptuelle Formalisierung bestimmter Wissensbereiche (z. B. Marketing, Vertrieb, Produktion, etc.). Der Begriff „Ontologie“ wird häufig synonym zu „Semantisches Netz“ und „Topic Map“ oder allgemeiner „Wissensmodell“ und „Wissenslandkarte“ verwendet. Der wohl bekannteste Definitionsversuch stammt von T. Gruber (1993)[6]:

» An Ontology is an explicit specification of a conceptualization «

Die Strukturierung unternehmens- oder fachspezifischer Themen erfolgte in der Vergangenheit häufig mittels Thesauri oder sonstiger Wortlisten. Eine deutliche Verbesserung der Abbildung von Wissensstrukturen stellen seit Ende der 90er Jahre Topic Maps dar. Inhalte werden durch Metadaten angereichert und dadurch maschinell auswertbar, so dass die Bedeutung von Begriffen eindeutig beschrieben ist.

Die Ontologie stellt in diesem Sinne eine Beschreibung von Konzepten (Begriffen) und Beziehungen (Relationen) in einem Diskursbereich (Domäne) dar. Ontologien bilden eine Schlüsseltechnologie für semantische Netzwerke. Der Unterschied zur Taxonomie ist der, dass die Ontologie ein Netzwerk von Informationen mit logischen Relationen darstellt, während die Taxonomie eine einfache Hierarchie bildet.

Am Anfang einer jeden Wissensrepräsentation steht immer die Frage: „*Was wollen wir wissen*?“ Um Antworten auf diese Frage zu finden, muss das Wissen modelliert und strukturiert werden. Dies geschieht mit dem Aufbau einer entsprechenden Ontologie. Innerhalb einer Organisation dienen Ontologien auch dazu, einen gemeinsamen Sprachgebrauch zu etablieren.

Dies ist notwendig, da in der menschlichen Sprache oft verschiedene Symbole synonym für die selbe Sache verwendet werden, z. B.: Auto = Fahrzeug = PKW = Kraftfahrzeug = Wagen = Automobil. Andersherum bezeichnet ein Symbol manchmal auch gleichzeitig verschiedene Dinge, z. B. ein Jaguar ist sowohl ein Auto als auch eine Raubkatze oder ein Kampfjet und mit einer Bank kann ein Sitzmöbel oder ein Kreditinstitut bemeint sein.

---

[6] Tom Gruber war in den frühen 1990ern an der Stanford University ein Vorreiter bei der Nutzung des Web für Wissensteilung und Zusammenarbeit

Die folgende Abbildung 1 zeigt einen Ausschnitt aus dem Klassifikationsschema einer Ontologie, die das Tier „Jaguar“ eindeutig beschreibt.

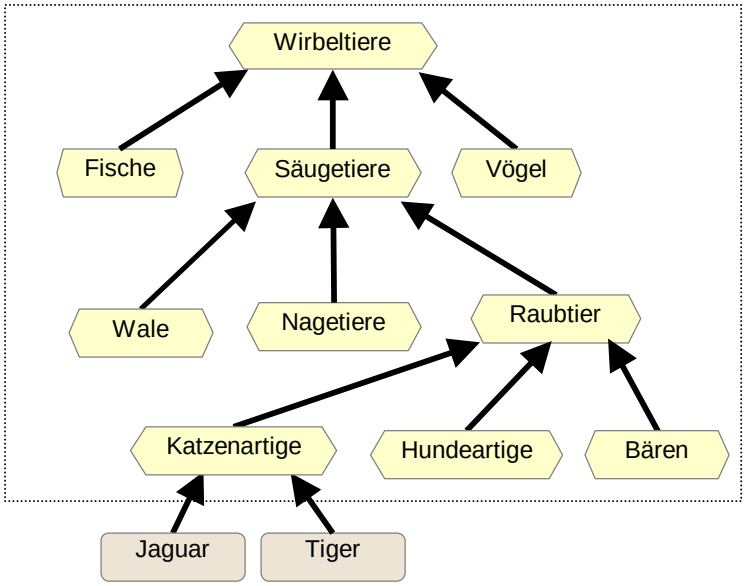

Abbildung 1: Ontologie Wirbeltiere

Eine Ontologie ist vergleichbar mit einem UML-Klassen-Diagramm. Dieses modelliert nach dem Konzept der objektorientierten Softwareentwicklung einzelne Klassen, deren Eigenschaften sowie die Beziehungen zwischen den verschiedenen Klassen. Ontologien haben die gleiche Aufgabe. Es werden jedoch keine Softwareklassen modelliert, sondern einzelne Konzepte.

Ontologiebasierte Lösungen können das Wissensmanagement wie folgt unterstützen [ontos]:

Wissenssuche
- Verbesserte Relevanz der Suchergebnisse
- Verbesserte Vollständigkeit der Suchergebnisse
- Auswertung von Beziehungen zwischen unterschiedlichen Suchergebnissen
- Erkennen nicht expliziter Zusammenhänge
- Grafische Navigation durch die Suchresultate
- Automatische Generierung von Navigationsstrukturen

Wissenspräsentation
- Visualisierung von Treffern
- Anzeige verwandter Dokumente
- Semantikbasierte Push-Dienste (Anzeige von Favoriten und Personalisierung)

Wissensvermittlung/-verteilung
- Bedarfsgerechte Mehrfachnutzung von Inhalten (z. B. für Abteilungen, Arbeitsgruppen, Partner und Kunden)

2.2. Topic Maps

Seit geraumer Zeit ist die Entwicklung semantischer Verfahren Gegenstand der Forschung. Diese Technologie soll Nutzern einen verbesserten Zugang zu großen Informationsmengen bieten [Hum 2004]. Eines dieser Verfahren ist das Konzept der Topic Maps. Topic Maps ist ein Standard zur Beschreibung von Informations- und Wissensstrukturen.

Topic Maps dienen der Verknüpfung von Wissensstrukturen mit existierenden Informationsressourcen [Pep 2001]. Die Idee hinter Topic Maps ist, ähnlich wie bei Lexika, Glossaren oder Indizes, externe Dokumente (etwa Bilder oder Artikel in einem digitalen Lexikon) zu referenzieren und die Themen dieses Wissens (z. B.: worüber ein Artikel in einem Lexikon handelt), miteinander in Verbindung zu bringen [Wid 2002].

Eine Topic Map bildet eine strukturierte Informationsschicht über den eigentlichen Informationsressourcen, die so mit weiteren Informationen angereichert werden können. Die Dokumente bleiben von der Topic Map unangetastet; dadurch sind Topic Maps losgelöst und austauschbar. Topic Maps wurden 1999 als ISO-Standard ISO/IEC 13250 normiert und später im Jahr 2000 als XML Topic Maps (XTM) in XML formuliert [Wid 2002].

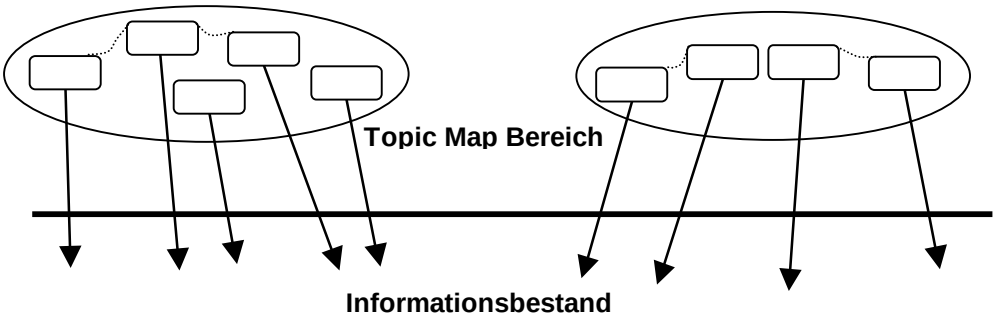

Abbildung 2: Topic Map - Strukturierte Informationsschicht über den Daten

Der Topic Map Standard basiert auf den gleichen Konzepten, die auch Indizes verkörpern; auf **T**opics, **A**ssociations und **O**ccurrences – kurz das TAO [Pep 2001]. Die nachfolgenden Abschnitte erläutern diese und weitere Konzepte bezüglich Identität, Rollen und Scope. Insgesamt umfasst das Datenmodell der Topic Maps 19 Elemente [XTM 2001].

Topic Maps sind eine Sammlung aus Topics und „semantischen“ Beziehungen, die diese zu einem Wortnetz verlinken. Occurrences (Vorkommensstellen) stellen in Topic Maps die Verbindung zwischen den Topics und externen Referenzen her, wie beispielsweise Ressourcen, die über URLs adressiert werden. Auf diese Weise ermöglichen Topic Maps außerhalb der eigentlichen Informationen eine einfache und selektive Navigation auf Metaebene zu den gewünschten Informationen.

Allgemein kann man sagen, dass Topic Maps ein standardisiertes Modell und eine Architektur für die gezielte Navigation, Suche und Auffindung von Informationen in großen Datenbeständen darstellen. Weiterführende Konzepte finden sich unter anderem in den Elementen Rollen und Scopes (Gültigkeitsbereich).

Die wesentlichen Bestandteile des XTM-Standards werden nachfolgend anhand eines Beispiels, das zum Teil der Topic Map „opera.xtm“[7] entstammt, erläutert. Die vollständige DTD zum XML Topic Maps Standard findet sich zusätzlich im Anhang dieser Arbeit.

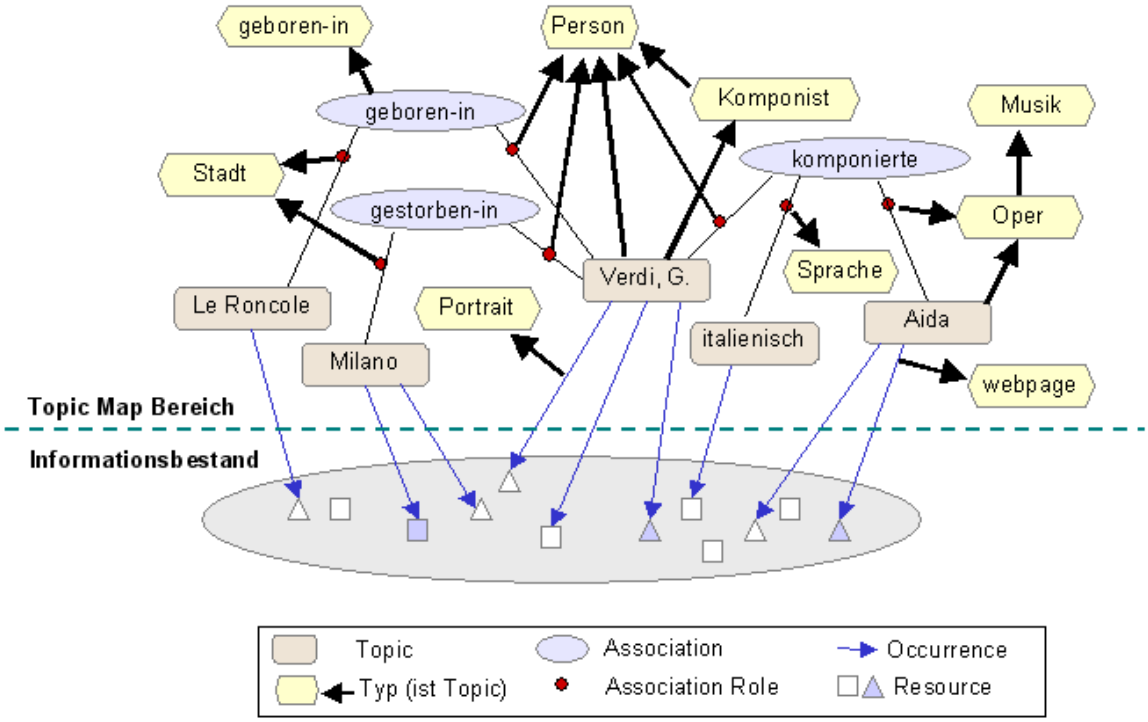

Abbildung 3: Topic Map-Schema "Guiseppe Verdi"

Die nachfolgende Abbildung 4 zeigt die Attribute und die Kindelemente des topicMap-Elements der XML Topic Map-DTD als Baumstruktur. Anschließend werden die einzelnen Elemente noch einmal textlich aufgeführt, beschrieben und abschließend mit einem XML-Beispiel belegt.

[7] Von Steve Pepper zu Demonstrationszwecken erstellte und frei verfügbare XTM http://www.ontopia.net/topicmaps/examples/opera/opera.xtm

Topic Map (Themennetz)

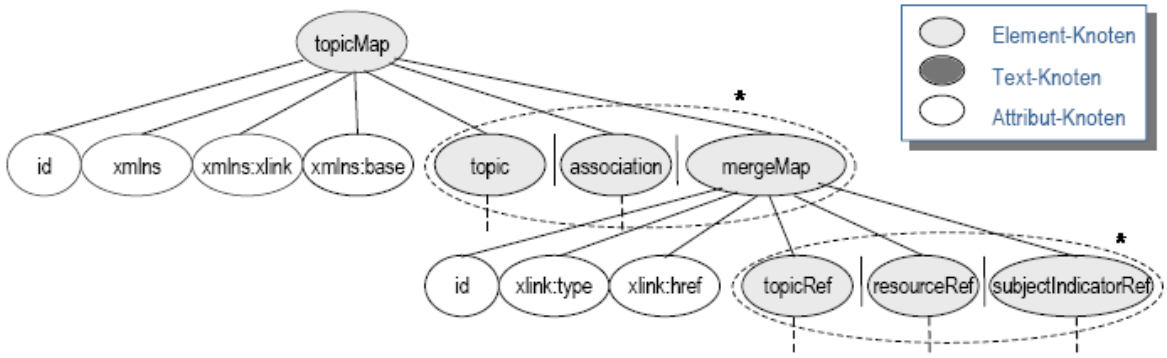

Abbildung 4: XML DTD - Baumstruktur Topic Map [Grün 2001]

Vater-Elemente	**---**		
Kinder-Elemente	`<topic>`		Topic ist ein netzinterner Repräsentant für ein Subject.
	`<association>`		Verknüpfung zwischen Topics.
	`<mergeMap>`		Dient der Verschmelzung der Topic Map mit anderen Topic Maps.
Attribute	`id`		Eindeutiger Identifizierer des Elements
	`xmlns`	CDATA	#FIXED 'http://www.topicmaps.org/xtm/1.0'
	`xmlns:xlink`	CDATA	#FIXED 'http://www.w3.org/1999/xlink'
	`xml:base`	CDATA	#IMPLIED

XML Beispiel: Topic Map

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

<topicMap
     id="tm-opera"
     xmlns="http://www.topicmaps.org/xtm/1.0/"
     xmlns:xlink="http://www.w3.org/1999/xlink"
>
<!-- Topics, Association und Verschmelzen von Topic Maps -->

</topicMap>
```

Topic (Thema)

Ein Topic repräsentiert ein beliebiges Objekt. Das kann von physikalischer Natur (ein reales Ding), eine Idee oder ein Konzept sein. Durch die Generalität des Topic-Konstruktes wird die Anwendbarkeit der Architektur für möglichst viele Gebiete garantiert und ein orthogonaler Aufsatz auf bestehende Wissensbasen ermöglicht. Topics ergeben sich aus dem jeweiligen Anwendungsbereich.

Die wesentlichen charakteristischen Merkmale eines Topics sind:

- Ein Topic besitzt einen internen, innerhalb einer Topic Map eindeutigen, Namen (***id***).
- Ein Topic hat mindestens einen für den Benutzer sichtbaren Namen (***baseName***).
- Ein Topic kann beliebig viele Vorkommensangaben auf externe Informationen enthalten (***occurrence***).
- Topics können Mitglieder (***member***) einer Beziehung (***association***) sein, in der sie dann eine Rolle (***roleSpec***) markieren.
- Topics können zur Klassifizierung als Typen anderer Topics referenziert werden (***instanceOf***). Auf diese Weise lässt sich eine Klassenhierarchie ähnlich einer Verzeichnisstruktur oder einer Vererbungsstruktur abbilden.

Die Gültigkeit jeder der drei Ausprägungen eines Topics als Topic, Occurrence oder Rolle einer Association, wird bestimmt durch den jeweils betrachteten Kontext (scope).

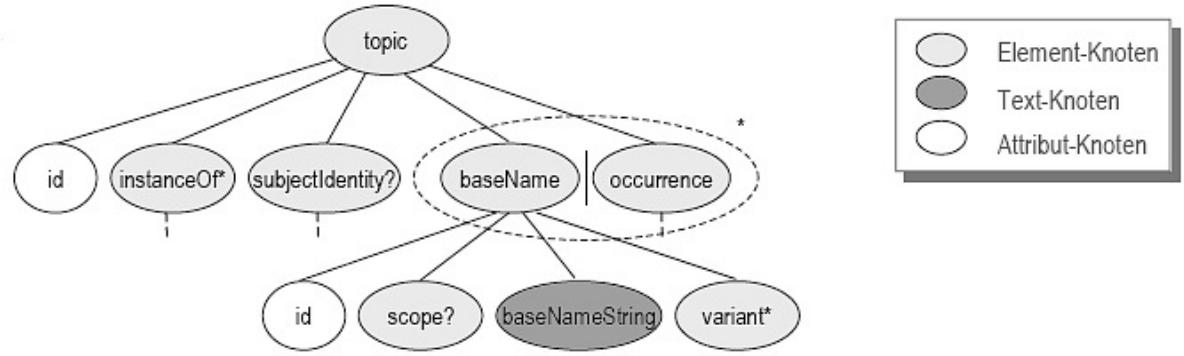

Abbildung 5: XTM-DTD Baumstruktur Topic [Grün 2001]

Vater-Elemente	`<topicMap>`	
Kinder-Elemente	`<instanceOf>`	Klasse-Instanz Beziehung. Verweis auf <topic> oder Subject Identity-Resource.
	`<subjectIdentity>`	Spezifiziert die Identität des Subjects des Topics.
	`<baseName>`	Spezifiziert die Namens-Charakteristiken des Topics.
	`<scope?>`	Scope eines <baseName> Elements für Synonyme.
	`<baseNameString>`	String für Topic-Name
	`<variant*>`	Andere Topic-Namen, z.B. zum Sortieren oder Anzeigen
	`<occurrence>`	Vorkommensstellen
Attribute	`id`	Eindeutiger Identifizierer des Elements

XML-Beispiel: Topic

```
<topic id="verdi">
  <!-- Topic „verdi“ ist Unterklasse von „composer“ und „person“ -->
  <instanceOf><topicRef xlink:href="#composer"/></instanceOf>
  <instanceOf><topicRef xlink:href="#person"/></instanceOf>

  <!-- Der angezeigte Name lautet „Verdi, Giuseppe“ -->
  <baseName>
    <baseNameString>Verdi, Giuseppe</baseNameString>
  </baseName>
</topic>
```

Association (Verbindung)

Eine Association stellt eine Beziehung zwischen zwei oder mehr Topics dar, die durch einen Verweis adressiert werden. Topics, die an einer Association beteiligt sind, heißen Member. Weitere Eigenschaften von Associations sind:

- Associations sind ungerichtete Kanten im Graphen. Um die Beziehung klar zu definieren, dient das Rollenkonzept.
- Über Typzuweisungen können Associations klassifiziert werden (*instanceOf*). Ein Typ ist wiederum ein Topic.
- Associations können beliebig viele Member haben.

Noch nicht im XTM Standard enthalten ist die Abbildung von Transitivität und Symmetrie.

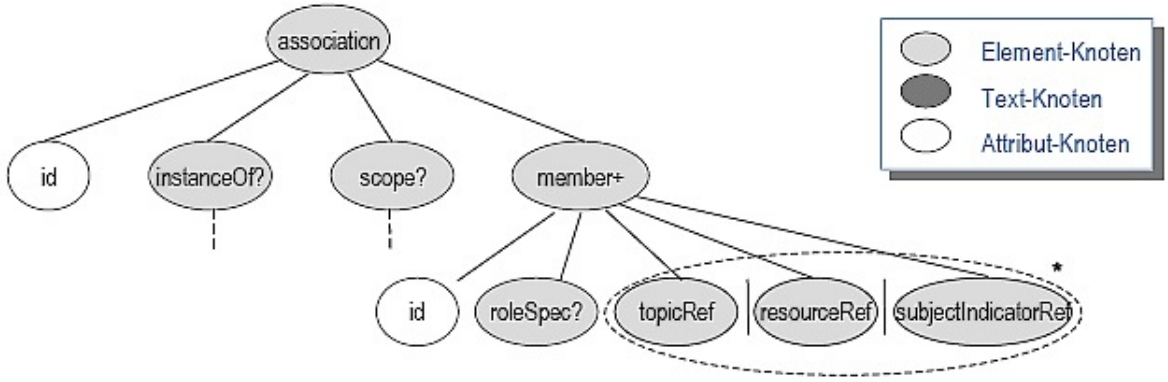

Abbildung 6: XTM-DTD Baumstruktur Association [Grün 2001]

Vater-Elemente	`<topicMap>`	
Kinder-Elemente	`<instanceOf>`	Klasse-Instanz Beziehung. Verweis auf <topic> oder Subject Identity-Resource.
	`<scope>`	Scope. Verweis auf <topic>, Resource oder Subject Identity-Resource.
	`<member>`	Spezifiziert die Topics, die eine Rolle in der Beziehung <association> einnehmen.
Attribute	`id`	Eindeutiger Identifizierer des Elements

XML Beispiel: Association

```
<association>
  <instanceOf><topicRef xlink:href="#composed-by"/></instanceOf>
  <member>
    <roleSpec><topicRef xlink:href="#composer"/></roleSpec>
    <topicRef xlink:href="#verdi"/>
  </member>
  <member>
    <roleSpec><topicRef xlink:href="#opera"/></roleSpec>
    <topicRef xlink:href="#aida"/>
  </member>
</association>
```

Occurrence (Vorkommensstelle)

Das occurrence-Element stellt die Verbindung zwischen den Strukturen der Topic Maps und externen Informationseinheiten her. Occurrences spezifizieren Ressourcen, die weiterführende, relevante Informationen zum Topic beinhalten. Hierbei werden zwei Arten von Ressourcen unterschieden:

a) Verweise auf externe Ressourcen via URLs, URNs *(resourceRef*)
b) Ressourcen, die selbst textuellen Inhalt enthalten (*resourceData*)

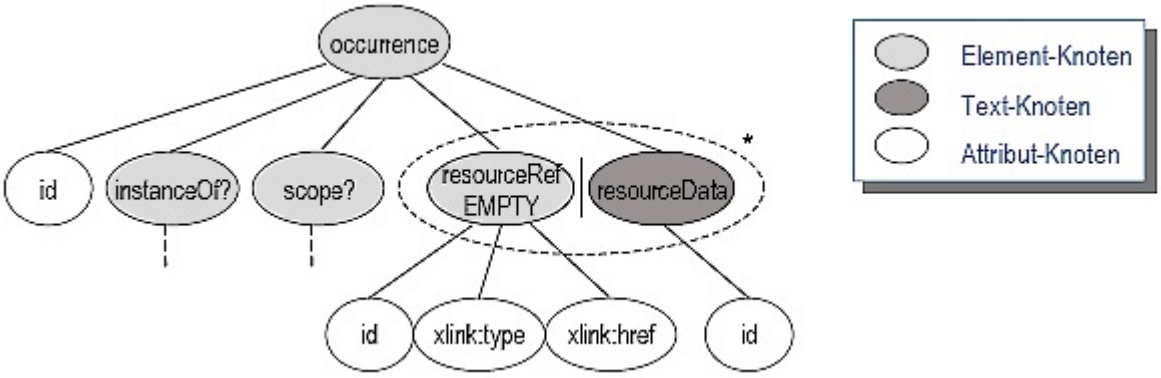

Abbildung 7: XTM-DTD Baumstruktur Occurrence [Grün 2001]

Vater-Elemente	`<topic>`	
Kinder-Elemente	`<instanceOf>`	Klasse-Instanz Beziehung. Verweis auf <topic> oder Subject Identity-Resource.
	`<scope>`	Scope. Verweis auf <topic>, Resource oder Subject Identity-Resource.
	`<resourceRef>`	Beinhaltet einen Link auf eine Ressource, die ein Vorkommnis von <topic> ist.
	`<resourceData>`	Inhalt ist eine Vorkommensstelle des Vater-Elements <topic>
Attribute	`id`	Eindeutiger Identifizierer des Elements

XML Beispiel: Occurrence

```
<topic id="verdi">
  <baseName>
    <baseNameString>Verdi, Giuseppe</baseNameString>
  </baseName>
...
  <!– Ein Bild vom Typ „portrait“ unter der angegebenen URL
  <occurrence>
    <instanceOf><topicRef xlink:href="#portrait"/></instanceOf>
    <resourceRef xlink:href=file:///E:\...\TopcMap\XTM\verdi.jpg/>
  </occurrence>
</topic>
```

Die besonderen Charakteristika von Occurrences sind:

- Externe Ressourcen können beliebige digitale multimediale Informationsobjekte sein, wie Web-Seiten, Dokumente, Bilder, Videos, Diagramme, Audiodateien.
- Occurrences können ebenfalls über Typzuweisungen klassifiziert werden (*instanceOf*). Der Typ einer Occourrence ist wiederum ein Verweis auf ein Topic.

Role (Assoziationsrolle)

Das Rollenkonzept in Topic Maps definiert die Art der Beteiligung, die ein Topic als Member einer Association inne hat. In einer Association „isPartOf“ könnte beispielsweise ein Topic die Rolle „Ganzes“ und ein anderes die Rolle „Teil“ verkörpern.

Scope (Gültigkeitsbereich)

Scopes begrenzen den Kontext für die Gültigkeit einer Topic Charakteristik. Formell bedeutet das, dass ein Scope eine Menge von Aussagen, die zusammen den Kontext definieren, zusammenfasst. Scopes werden beispielsweise häufig zur Abbildung von Mehrsprachigkeit eingesetzt oder zur Definition von Sichtbarkeiten im Zusammenhang mit Benutzerrechten. Scopes lassen sich neben dem Gültigkeitsbereich eines Topic BaseName auch für Associations und Occurrences festlegen.

Weiterführende Topic Map Konzepte sind:

InstanceOf

Klasse-Instanz Beziehung, dient der Typenbildung bei Topics, Associations, Assocation Roles, Occurrences und Scopes.

Topic Types

Jeder Typ selbst ist wiederum ein Topic, das entweder in der gleichen Topic Map oder in einem anderen Dokument definiert wird (*subjectIndicatorRef*). Topic Typen führen eine Typenhierarchie ein. Jede Topic Map hat ihre eigene Typenhierarchie. Es gibt kein globales Typensystem. Das Typensystem einer Topic Map bezeichnet die Ontologie des modellierten Wissensraums.

Variant

Varianten stellen alternative Formen des Basisnamens eines Topics bereit. In bestimmten Kontexten sind diese dann aussagefähiger als der Basisname. Variant-Namen dienen u. a. zur Abbildung von Mehrsprachigkeit. Der angezeigte Name für „Auto“ ist dann im deutschen Kontext „Auto“, im englischen „car“ und im französischen „voiture“. Ein Variant-Name kann eine Zeichenkette aber auch jede andere Art von Informationsressource sein.

2.3. Metadaten

Als Metadaten oder Metainformationen bezeichnet man allgemein Daten, die Informationen über andere Daten enthalten. Metadaten sind strukturierte oder beschreibende Angaben, die eine Ressource, eine Entität, ein Objekt oder andere Daten beschreiben. Sie können darüber hinaus dem Auffinden, der Verwendung oder der Verwaltung einer Ressource dienen.

Bei der Speicherung von Metadaten gibt es verschiedene Ansätze:

1. Im Dokument selbst. So ist in einem Buch stets auch der Autor und das Erscheinungsjahr verzeichnet. In HTML-Dokumenten werden mit Hilfe von Meta-Tags Sprache, Autor, Unternehmen und Schlagwörter angegeben, die beispielsweise von Suchmaschinen ausgewertet werden können.
2. In zugeordneten Nachschlagewerken, zum Beispiel für ein Buch in einer Bibliothek im Bibliothekskatalog.
3. Bei Computerdateien in den Dateiattributen. Die meisten Dateisysteme erlauben nur genau festgelegte Metadaten in Dateiattributen. Auch ist es üblich, die Meta-Information „Dateityp“ im Dateinamen unterzubringen; typischerweise in der Extension.
4. Als begleitende Dateien die als ,,Anhängsel" zu anderen Dateien gehören und meist weitere Informationen oder Einstellungen zu den Hauptdateien beinhalten. Verschiedentlich bezeichnet man diese als Sidecar-Dateien oder auch als „Filialdokumente“. Beispiel hierfür sind Bildformate, z. B. Adobe Photoshop™

Die meisten „Multimedia“-Formate[8] können bereits mit zusätzlichen Informationen versehen werden. Dabei handelt es sich in der Regel allerdings um Daten, die für den jeweiligen Typ signifikant sind. Im WWW dienen unter anderem die Standards Dublin Core[9] und RDF dazu, Ressourcen zu beschreiben.

---

[8] Der Begriff Multimedia bezeichnet Inhalte und Werke, die aus mehreren, meist digitalen Medien bestehen: Text, Fotografie, Grafik, Animation, Audio und Video.

[9] Die Initiative zur Gründung der Dublin Core Metadata Initiative (DCMI) entstand 1994 am Rande einer World-Wide-Web-Konferenz in Chicago. Der Name „Dublin Core“ rührt her vom ersten Tagungsort dieser Initiative im März 1995 in Dublin/Ohio - http://dublincore.org

2.1.1. Dublin Core

Dublin Core ist ein Metadatenformat zur Beschreibung von Dokumenten und anderen Objekten im Internet. In seiner einfachen Version als *Dublin Core Metadata Element Set* besteht es aus 15 Datenfeldern. Sämtliche Felder sind optional und können auch mehrfach vorkommen.

1. Title	6. Contributor	11. Source
2. Creator	7. Date	12. Language
3. Subject	8. Type	13. Relation
4. Description	9. Format	14. Coverage
5. Publisher	10. Identifier	15. Rights

Tabelle 1: Dublin Core Datenfelder

Im Gegensatz zu Standards wie MAB[10] und MARC[11] ist Dublin Core kein bibliothekarisches Metadatenformat. Da es keine genauen Regeln für die Belegung der einzelnen Felder gibt, ist Dublin Core eher eine Art „minimale Empfehlung“ für den Austausch von Metadaten.

2.1.2. RDF (Resource Description Framework)

Im WWW dient RDF dazu, Ressourcen mittels einer formalen maschinenlesbaren Sprache zu beschreiben, d. h. sie mit Metadaten zu versehen. Das RDF Datenmodell besteht aus den Kernkonzepten Ressource, Property und Statement [W3C RDF].

Ressource: Jedes Ding, das beschrieben werden kann, wird in der RDF-Spezifikation als Ressource bezeichnet und durch eine URI adressiert.
Property: Ein Property (Eigenschaft) beschreibt eine Ressource in irgendeiner Art näher.
Statement: Ein RDF-Statement ist eine Kombination aus Ressource mit einer benannten Eigenschaft (Property) und dem zugehörigen Wert der Eigenschaft.

Jeweils eine Ressource, eine Eigenschaft und ein Objekt bilden zusammen ein so genanntes RDF-Tripel. Die folgende Abbildung zeigt die Zusammenhänge zwischen den einzelnen Objekttypen.

Subjekt (Ressource)	http://home/xyz.html
Prädikat (Eigenschaft)	autoredBy
Objekt (Literal)	„Alice”

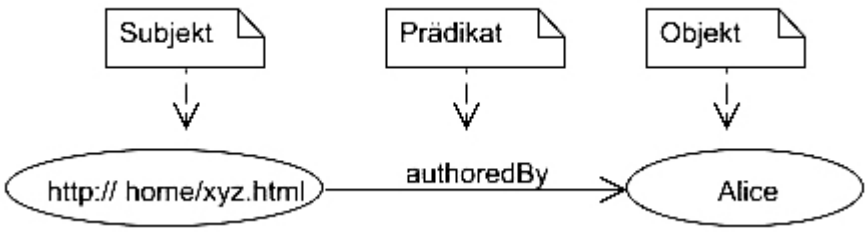

Abbildung 8: RDF-Statement in Graphensyntax-Notation

[10] http://mab.know-library.net/
[11] http://marc.know-library.net/

2.4. SVG (Scalable Vector Graphic)

SVG ist ein Standard zur Beschreibung zweidimensionaler Vektorgrafiken in XML-Syntax [W3C SVG]. SVG wurde im September 2001 vom W3C als Empfehlung veröffentlicht und viele aktuelle Webbrowser wie zum Beispiel Mozilla, Firefox oder Opera können von Haus aus einen Großteil des Sprachumfangs darstellen. Andere, z. B. der InternetExplorer, benötigen zur Darstellung jedoch noch die Installation eines Plug-In, zum Beispiel den SVG-Viewer von Adobe[12].

SVG ermöglicht das Erstellen strukturierter Dokumente und die Verarbeitung im Umfeld anderer XML-Technologien. SVG-Dokumente lassen sich statisch und dynamisch generieren und relativ einfach in serverseitige Anwendungen integrieren. Da SVG Graphiken XML-Dokumente sind, ist eine Transformation via XSLT in andere bzw. aus anderen XML-Dokumenten möglich.

Vektorgrafiken beschreiben nicht einzelne Pixel, wie beispielsweise Grafikformate (GIF, JPEG, PNG) sondern geometrische Elementarobjekte (Primitive). D. h. Bilder bestehen nicht aus einer Matrix einzelner Bildpunkte, sondern aus einer mathematischen Beschreibung einzelner Bildobjekte. Ein Kreis wird beispielsweise durch seinen Mittelpunkt und seinen Radius festgelegt .

Beispiel:

```
<circle   cx="600" cy="200" r="100"
          fill="red"
          stroke="blue"
          stroke-width="10"  />
```

Die weiteren Attribute beschreiben die Füllfarbe, die Farbe des Rands sowie dessen Stärke. Das Rasterformat, das ein Bildschirm oder ein Drucker für die Wiedergabe benötigt, kann aus den Angaben für jede beliebige Darstellungsgröße aktuell neu und exakt berechnet werden. Daraus folgt die beliebige und verlustfreie Skalierung. Die bei Rastergrafiken bekannten Treppenstufen- und andere Vergröberungseffekte treten nicht auf (siehe Abbildung 9).

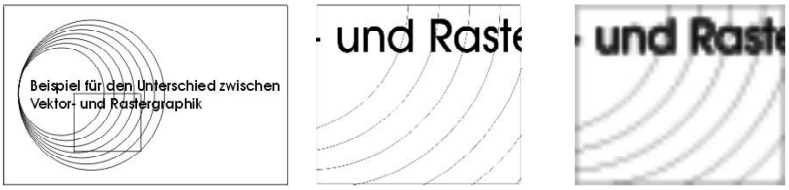

Abbildung 9: Die Ausschnittsvergrößerung einer Vektorgraphik bleibt exakt (Bild, mitte), im Gegensatz zum Fall einer Rastergraphik (Bild, rechts)

Die technische Grundlage für Interaktivität und deren Auswirkungen ist die Erzeugung und Manipulation von Elementen. Per JavaScript können sämtliche Elemente auch am Client neu kreiert werden. Die noch nicht standardisierten, aber in fast allen Viewern implementierten JavaScript Methoden `getURL()` bzw. `parseXML()` ermöglichen das Nachladen von

[12] http://www.adobe.com/svg/viewer/install/main.html

weiteren Daten in SVG Dateien [Hel 2003]. Sämtliche auf diese Art und Weise eingefügten Elemente sind nach dem Ladevorgang in der SVG Datei verfügbar, als wären sie ursprünglich schon vorhanden gewesen.

SVG zeichnet sich durch folgende Eigenschaften aus [Ber 2005]:

- SVG ist eine auf XML basierende Sprache
- SVG ist ein offizieller Standard des W3C
- SVG Grafiken verfügen über eine hohe Auflösung/Qualität
- SVG Grafiken (Vektorgrafiken) sind ohne Qualitätsverlust skalierbar
- SVG Dokumente zeichnen sich gegenüber Rastergrafiken durch geringe(re) Dateigröße aus
- SVG ermöglicht dynamische Interaktivität durch Unterstützung von Scriptsprachen (z.B. JavaScript[13], ECMAScript[14])
- SVG ermöglicht Detailansichten durch Zoomen und Schwenken in Grafiken
- SVG bietet zahlreiche Filtereffekte
- SVG Grafiken sind auf jeder Plattform mit XML-Werkzeugen erstellbar
- SVG unterstützt Stylesprachen (z.B. CSS)
- SVG wird weitgehend kompatibel zu HTML 4.0 sein
- SVG ist voll kompatibel zu XHTML
- SVG besitzt ein eigenes DOM
- universell einsetzbar [bei nativer Browser-Unterstützung]

Aufgrund der genannten Eigenschaften und Fähigkeiten lässt sich SVG sehr gut für Visualisierung und grafische Navigation in Web-basierten Anwendungen einsetzen.

2.5. Zusammenfassung

Topic Maps werden häufig als Alternative zu RDF-basierten Standards des W3C gehandelt. Aber im Gegensatz zu RDF, das eine computerverstehbare Formalisierung zum Ziel hat, sind Topic Maps eher zur Strukturierung von Wissen aus Sicht der Menschen konzipiert. Topic Maps sollen die bessere Navigation und Suche in Internet-Resourcen und anderen Dokumenten ermöglichen und dem Austausch von Metadaten dienen [XMLCH].

In KORA-Dokumentenmanagementsystem steht das Auffinden von Informationen durch den Benutzer im Vordergrund. Eine Wissensrepräsentation soll dabei helfen, Informationen zu strukturieren und so leichter auffindbar zu machen. Die Visualisierung der Zusammenhänge hängt dabei nicht unmittelbar von der Beschreibungssprache XTM ab, sondern kann an die jeweiligen Bedürfnisse des Anwenders angepasst werden.

Für die Implementierung des geplanten Systems sind Topic Maps daher das geeignetere Konzept, da sie eine eigene Informations- und Strukturierungsschicht über existierenden Informationsressourcen bilden und die Dokumente von der Topic Map unangetastet bleiben.

---

[13] JavaScript ist eine objektbasierte Skriptsprache. JavaScript ist eine Marke der Firma Sun Microsystems, Inc.

[14] JavaScript ist durch die Ecma International (ehemals ECMA-*European Computer Manufacturers Association*) unter dem Namen „ECMAScript“ standardisiert.

Für die Visualisierung der Netzstruktur wird hier auf SVG gesetzt. Für einen Prototyp sind die Möglichkeiten von SVG ausreichend. Dadurch, dass es sich bei SVG genau wie bei XTM um eine XML-basierte Sprache handelt, fügt es sich hervorragend in die geplante Architektur ein. Da die Visualisierung nicht von der Beschreibungssprache XTM abhängig ist, kann für eine spätere Version ggf. ein anderes, z. B. ein 3D, Visualisierungsverfahren gewählt werden.

3. Vision

3.1. Beschreibung der Vision des KORA-Systems

Mitglieder einer Arbeitsgruppe, evtl. sogar Personen aus unterschiedlichen Organisationen, kommen zu einem Arbeitstreffen zusammen. Die meisten Teilnehmer sind im Besitz eines oder mehrerer intelligenter „mobiler“ Endgeräte (z. B. Notebook, PDA oder Handy), auf denen benötigte Arbeitsmaterialien gespeichert sind.

Der Raum ist ausgestattet mit mehreren großen touchsensitiven TFT-Wandbildschirmen und ggf. weiteren, in Arbeitstische eingelassenen, touchsensitiven Monitoren. Des Weiteren agiert im Hintergrund eine Serverlandschaft, bestehend aus Applicationservern, Datenbank, Authentifizierungsservice u. a.

Abbildung 10: Der Gruppenarbeitsraum iRoom des Projekts Interactive Workspaces der Stanford University in Benutzung [Joh 2004]

Die Kernfunktionalität des KORA-Dokumentenmanagementsystems liegt in der inhaltlichen Vernetzung verschiedenartiger Dokumente. Dies geschieht auf die Weise, dass Nutzer mitgebrachte oder „on the fly“ auf einem mobilen Gerät erstellte digitale Dokumente kabellos ins System einbinden können.

3.2. Vision der Netzmodellierung

Erfahrungen mit betrieblichen Wissensmanagementsystemen haben gezeigt, dass nahezu jedes zentrale Wissensmanagementsystem von mangelnder Akzeptanz bedroht ist [Lud 2005]. Das liegt unter anderem daran, dass Wissen immer persönliches Wissen ist und mit einer persönlichen Anschauung und einem individuellem Kontext verknüpft ist. Wenn jemand sein Wissen nun in eine Ordnung pressen soll, die seiner Sicht der Dinge nicht entspricht, ruft dieses unweigerlich Widerstände hervor.

Bei der Erstellung des Datenmodells für das KORA-Netztool ist deshalb angestrebt, die erforderliche persönliche und individuelle Ausdrucksfähigkeit der Nutzer zu integrieren.

3.2.1. Datenmodell des Wissensnetzes

Dem Netztool zur Wissensrepräsentation im Collaborative Workplace soll ein möglichst einfaches Datenmodell zugrunde liegen, das möglichst intuitiv verständlich ist und eine gezielte Navigation durch die Informationen ermöglicht. Das Datenmodell muss dabei mindestens folgende Anforderungen erfüllen:

1. Es muss eine eindeutige Verbindung zu existierenden Ressourcen, in Form von beliebigen Dokumenten, hergestellt werden.
2. Diese Ressourcen sollen mittels aussagefähiger Links, ähnlich dem Hyperlink-Prinzip miteinander verknüpfbar sein.
3. Die Ressourcen sollen über die Dateiattribute hinaus mit weiteren beliebigen Metadaten angereichert werden können.

Ein Datenmodell beschreibt und beschränkt, wie jedes andere Schema auch, die Ausdrucksfähigkeit der Nutzer beim Abspeichern von Informationen. Demnach gilt es einen Kompromiss zu finden, der sowohl eine einheitliche Grundlage vorgibt, aber auch Spielraum für die fortlaufende Weiterentwicklung des Datenmodels lässt.

Mit Topic Maps steht ein entsprechend flexibles und noch dazu leistungsfähiges Datenmodell für die Informationsrepräsentation zur Verfügung. Topic Maps sind ein abstraktes Modell, mit dem sich die obigen Anforderungen umsetzen lassen, da sie zur Strukturierung von Wissen aus Sicht der Menschen konzipiert sind. Im Gegensatz zu RDF, das eine computerverstehbare Formalisierung zum Ziel hat [XMLCH]. Darüber hinaus haben Topic Maps den Vorteil, dass neue Topic- und Assoziations-Typen zu jeder Zeit in der Topic Map selbst erstellt werden können und nicht, wie bei RDF, in einem externen Schema definiert werden müssen.

3.2.2. Repräsentation

Das wesentlichste Anliegen eines Wissensmanagements ist die Nutzbarmachung von Informationen und Dokumenten aller Art. Dies beinhaltet die Sammlung und Ordnung von Dokumenten. Zur Wissensordnung und -repräsentation existieren verschiedene Techniken, die ihren Ursprung z. T. in der Bibliothekswissenschaft haben:

a) Katalog, Glossar, Taxonomie (einfache kontrollierte Vokabularien)
b) Klassifikation, Thesaurus (begrenzte Zahl von Relationen in der Regel ohne Vererbungsrelation)

c) Semantisches Netz, Ontologie, Frames, Produktionsregeln
d) Axiomensystem, Prädikatenlogik
e) Mehrschichtige erweiterte Semantische Netze (MultiNet)

Mit anderen Worten, bei der Wissensrepräsentation geht es um die Frage, wie Wissen in einem Informationssystem formal gespeichert werden kann, mit welchen Mechanismen darauf zugegriffen und wie darauf aufbauend Wissen visualisiert und somit nutzbar gemacht werden kann [SWSch-a].

Damit eine Eindeutigkeit der verwendeten Symbole und Begriffe und damit verbunden eine Wiederverwendung und Interoperabilität des repräsentierten Wissens möglich ist, muss sich ein solches System auf eine Ontologie stützen [Berg 2006]. Die Erstellung einer Ontologie bedarf jedoch einer gewissen redaktionellen Vorarbeit, hierzu zählt unter anderem die Abgrenzung des Diskursbereiches und die Definition der Zielsetzung.

Die Zielsetzung legt fest, welche Fragen und Problemstellungen der Nutzer mit Hilfe der erstellten Topic Map lösen können soll [Berg 2006]. Für eine Topic Map beinhaltet das Erstellen einer Ontologie die Suche nach allgemeingültigen Typen der vorkommenden Topics und Assoziationen. Damit werden gleichzeitig Kriterien für die Relevanz eines Topics festgelegt und über die Aufnahme eines Topics in die Map oder deren Ausschluss entschieden.

Das Klassifikationsschema einer Topic Map sieht so aus, dass alles auf ein Topic zurückgeführt wird, selbst ein Assoziationstyp verweist auf ein Topic. Die so entstehende Ontologie kann sowohl als eigenständige Topic Map modelliert oder als Teil der eigentlichen Repräsentation erstellt werden.

3.2.3. Navigation im Netz

Außer dem Sammeln und Ordnen von Dokumenten, gehört zur Nutzbarmachung von Informationen auch das zielsichere Wiederfinden gespeicherter Dokumente. Entsprechende Suchfunktionen basieren auf folgenden Konzepten [Baum 2002]:

a) **Keyword Search**:
Der Anwender hat eine klare Vorstellung wonach er sucht und stellt eine konkrete Anfrage an das System.

b) **Information Browsing**:
Der Anwender hat ein unvollständiges Wissen und startet die Informationssuche mit einer sehr allgemeinen Anfrage, um anschließend in Abhängigkeit der erhaltenen Resultate die Anfrage zu spezifizieren.

Beim Information Browsing Prozess wird das genaue Ziel der Suche dadurch erreicht, dass sich der Anwender über die interaktive Benutzungsschnittstelle an die gesuchte Information „herantastet“. Unterstützt wird dies dadurch, dass bei entsprechender graphischer Visualisierung auch der Kontext der jeweiligen Information angezeigt wird. Ein zusätzlicher positiver Effekt dieses Verfahrens besteht darin, dass durch die Einbeziehung thematisch nahestehender Informationen, ein Lerneffekt erzielt werden kann.

3.2.4. Visualisierung

Mit Hilfe von Topic Maps lassen sich Graphen aus Knoten und Kanten modellieren. Diese, auch als „Themennetze“ oder „Wissensnetze“ bezeichneten Graphen, spiegeln Wissensausschnitte der realen Welt wieder. Sie eignen sich gut, vernetztes Wissen und kontextabhängige Sachverhalte darzustellen. Die Vorteile dieser, auf Basis semantischer Technologien erstellten Netze, liegen in ihrer Flexibilität und Eingängigkeit.

Doch genau hierin liegt auch eines der Hauptprobleme. Die Übersichtlichkeit, die von der Darstellung kleiner semantischer Graphen suggeriert wird, geht mit zunehmender Zahl von Knoten und Kanten schnell verloren. Dies ist bei der Visualisierung sehr großer Topic Maps nicht anders. Denn zum einen besteht die Anforderung, dem Nutzer möglichst viele der im Netz modellierten Informationen auf einen Blick ersichtlich zu machen und gleichzeitig muss die Darstellung aber auch noch übersichtlich sein. Dieses Problem verschärft sich noch beim Einsatz von Geräten mit sehr kleinen Displays, wie z. B. PDAs oder Handys.

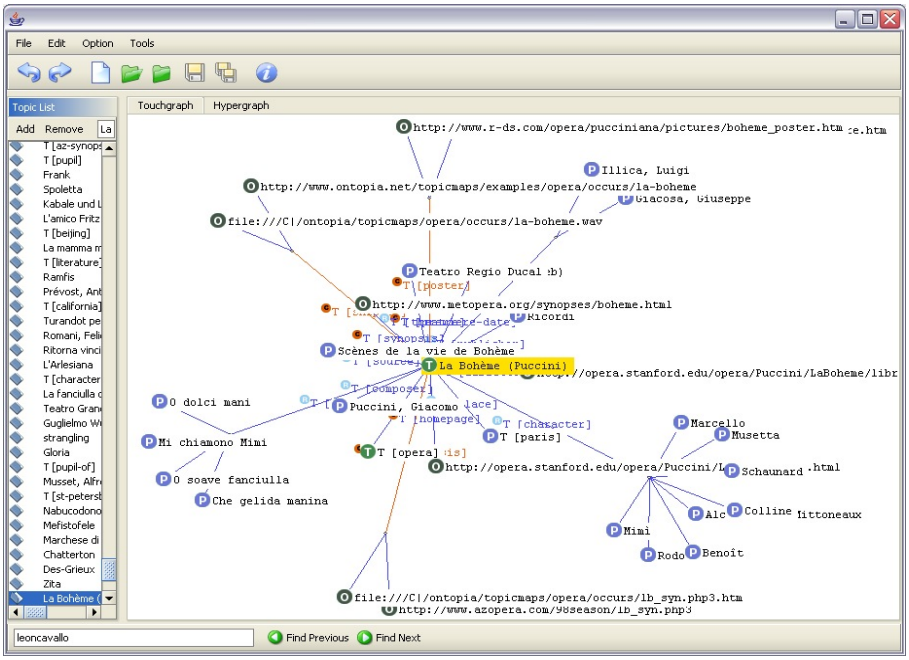

Abbildung 11: Große Anzahl an Knoten geht zu Lasten der Übersichtlichkeit

Bei der Gestaltung der Visualisierung gilt es daher einen Kompromiss zu finden, bzw. eine flexibel und durch den Benutzer anpassbare Darstellungsform zu entwerfen. Hier reichen die Ansätze von einfachen, als Hyperlinks vernetzten Strukturen, über explorerähnliche Visualisierungen, Hyperbolic Trees und Knowledge Maps, bis hin zu dreidimensional modellierten Visualisierungen [Park 2002].

3.2.5. Abgrenzung zu „Semantischen Netzen“

Semantische Netze sind Repräsentationen von Wissen in Form von Graphen. Diese bestehen aus Knoten, die Dinge aus der Realität darstellen und Kanten, die deren Beziehungen repräsentieren. Semantische Netze sind besonders in der Künstliche-Intelligenz-Forschung eingesetzt worden. Der Begriff „Semantische Netze“ ist ein Oberbegriff, der zunächst keine Aussage über die Art der Formalisierung, die Aussagekraft und die Implementierungssprache enthält. Topic Maps stellen eine Variante von Semantischen Netzen dar. Mit Semantischen Netzen verbunden ist ein komplexes Konzept

der Logik zur Fortschreibung von Inhalten. Topic Maps verhindern dies nicht, konzentrieren sich aber auf die Darstellung von Links als Basis für die linkbasierte Abfrage von Informationen.

3.3. Systemanforderungen

Aus den vorangegangenen Ausführungen geht bereits hervor, dass der Raum von Projektgruppen unterschiedlicher Fachgebiete genutzt werden soll. In den Diplomarbeiten [Neu 2006] und [Burf 2006] sind exemplarisch Szenarien aus den Bereichen Softwareentwicklung und Stadtplanung beschrieben.

Aufgrund der Gemeinsamkeiten der in diesen Szenarien extrahierten Anforderungen wird die Hypothese gewagt, dass diese auch für weitere Anwendungskontexte gelten könnten. Aus diesen und weiteren allgemein gültigen Softwareanforderungen soll nachfolgend ein Anforderungskatalog aufgestellt werden. Die identifizierten Anforderungen wären für den Einsatz der Software in einem Produktivsystem zwingend. Der im Rahmen dieser Arbeit entwickelte Prototyp wird einige Funktionalitäten jedoch nur andeutungsweise implementiert haben.

Das Hauptaugenmerk wird hierbei auf der Benutzungsschnittstelle und der Wissensrepräsentation liegen. Systemanforderungen bezüglich Technologiebasis, Persistenz und Middleware wurden bereits in den Diplomarbeiten [Bart 2006], [Mund 2006] und [Burf 2006] behandelt.

3.3.1. Nichtfunktionale Anforderungen

Die nichtfunktionalen Anforderungen beschreiben Anforderungen, die nicht unmittelbar ausführbare Dienste des Systems betreffen, aber dennoch für die Nutzbarkeit des Systems unabdingbar sind.

3.3.1.1. Benutzungsfreundlichkeit

Bedingt durch das möglicherweise geringe technische Vorwissen und die eingeschränkten Einarbeitungsmöglichkeiten der Anwender, ergeben sich für den Konferenzraum besonders hohe Anforderungen an die Bedienbarkeit der Software. Kriterien und Definitionen bezüglich Anforderungen an eine ergonomische Gestaltung der Benutzungsschnittstelle einer Software in der Bürokommunikation sind inzwischen in Normen, z.B. DIN 66 234 und Standards, z. B. ISO 9241, festgelegt [Stary 1996].

Zu den Bewertungskriterien zählen:

- Aufgabenangemessenheit
- Selbstbeschreibungsfähigkeit
- Steuerbarkeit
- Erwartungskonformität
- Erlernbarkeit
- u.a.

Erlernbarkeit ist beispielsweise gegeben, wenn es möglich ist, die Aufgabenbewältigung in einer angemessenen Zeitspanne zu erlernen.

3.3.1.2. Wahrung von Rechten

Eine weitere nichtfunktionale Anforderung in Verbindung mit Dokumenten ist auf juristischer Ebene zu sehen, nämlich der Wahrung von Rechten am geistigen Eigentum, dem Urheberrecht. Dieses kann in manchen Anwendungsfällen eine Rolle spielen. Das Urheberrecht bleibt in jedem Fall unangetastet. Unabhängig davon können einem Nutzer eingeschränkte Manipulationsrechte eingeräumt werden, die dann vom System entsprechend kontrolliert werden müssen.

3.3.1.3. Effizienz

Eine wesentliche Rolle für die Akzeptanz eines Systems bei den Nutzern spielt die Effizienz. Die Effizienz bezeichnet die Sparsamkeit bezüglich der Ressourcen, Zeit und Speicherplatz, die zur Lösung eines festgelegten Problems beansprucht werden. Die Tatsache, dass durch ein Netz navigiert wird, darf für den Nutzer zu keinen spürbaren Antwortzeitverlängerungen führen.

3.3.2. Funktionale Anforderungen

Die folgenden Absätze beschreiben im einzelnen die Fähigkeiten des Systems, die der Anwender erwartet, um seine fachlichen Probleme mit Hilfe des Systems zu lösen.

3.3.2.1. Authentifizierung und Autorisierung

Beim Betreten des Raumes muss eine Authentifizierung erfolgen. Hierfür sind unterschiedliche Verfahren denkbar. Es wäre sowohl ein manuelles LogIn mit Benutzernamen- und Passworteingabe oder der Einsatz eines Besitz- oder Merkmalauthentifizierungsverfahrens, per SmartCard oder Fingerprintscanning denkbar.

3.3.2.2. Unterstützung einfacher mobiler Endgeräte

Die Nutzung des Kora-Systems sollte ortsunabhängig und möglichst ohne spezielle Konfigurationen und ohne aufwendige Softwareinstallationen mit jedem internetfähigen Endgerät möglich sein. D. h. die Anzeige und Bearbeitung der Repräsentation des Wissensnetzes soll auf Geräten mit unterschiedlichen Eingabe- und Darstellungsmöglichkeiten angepasst sein.

3.3.2.3. Versionskontrolle

Wenn mehrere Personen gemeinsam ein Projekt bearbeiten, hierfür mehrere Computer benötigt werden und wenn es außerdem möglich sein soll, Änderungen im Projekt zurückzuverfolgen, ist der Einsatz von Versionskontrollsystemen sinnvoll. Ein Versionskontrollverfahren besteht aus zwei Mechanismen:

a) Transaktionsbearbeitung: Änderungen an Dokumenten werden wie Transaktionen behandelt, d. h. Ausschluss paralleler Updates
b) Versionierung: Rückgriff auf beliebige ältere Versionen ist möglich.

Das bekannteste der frei verfügbaren Versionskontrollsysteme ist das „Concurrent Version System“ (CVS). Grundsätzlich lässt sich jedes Dokument versionieren. In der Praxis werden die Verfahren der Versionskontrolle allerdings nur selten außerhalb der Softwareentwicklung eingesetzt. Hinsichtlich der angewendeten Versionierungsverfahren lassen sich zwei Typen von Versionskontrollsystemen unterscheiden:

a) Jede Änderung eines Dokuments wird als neue Version gespeichert. Die zeitliche Abfolge wird durch Versionsnummern gekennzeichnet. Frühere Fassungen bleiben erhalten. Dieses Verfahren erzeugt in änderungsintensiven Anwendungsgebieten einen hohen Speicherplatzbedarf, da jeweils eine vollständige Kopie des Dokuments gespeichert wird.
b) Nur die aktuelle Version wird als Vollversion gespeichert. Unterschiede zur Vorversion werden als Delta-Kodierung archiviert. Bei Bedarf wird eine beliebige Vorversion anhand des Änderungsprotokolls zurück berechnet. Dieses Verfahren ist wesentlich speicherplatzsparender.

3.3.2.4. Offlinebetrieb

Gegebenenfalls kann es nötig sein, dass Anwender Dokumente aus dem KORA-Dokumentenmanagement auch offline nutzen möchten. In diesem Fall sollte es möglich sein, ein Subnetz auszuwählen und herunterzuladen. Abbildung 12 zeigt ein solches Subnetz in den Ausprägungen 1. Ebene, 2. Ebene und 3. Ebene. Ein Download beinhaltet für die involvierten Knoten die Dokumente, die Repräsentation des Netzes und die Metadaten zu den Dokumenten. Um die Netzrepräsentation auch ohne Netzwerkverbindung nutzen zu können, muss allerdings auf dem Endgerät des Nutzers eine Desktopapplikation vorhanden sein, die zumindest die Visualisierung einer Topic Map ermöglicht.

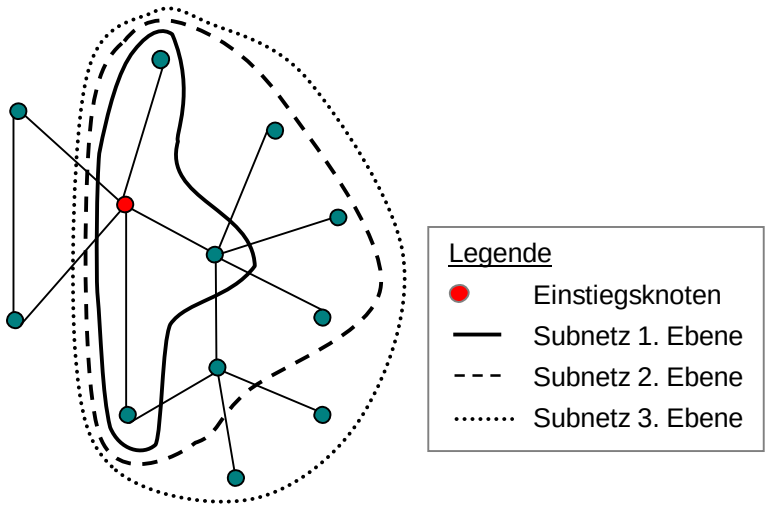

Abbildung 12: Subnetzebenen

3.3.2.5. Synchronisierung

Im Anschluss an ein erfolgreiches LogIn könnte dem Nutzer vom System angezeigt werden, welche Dokumente seit seinem letzten LogIn neu eingefügt wurden bzw. welche Dokumente zwischenzeitlich in einer neueren Version vorliegen. Auf Wunsch könnte dann eine Synchronisierung mit seinem mobilen Endgerät durchgeführt werden. Ebenso könnte das System einen Abgleich mit einem mitgebrachten mobiles Endgerät durchführen und prüfen, ob hier in einem bestimmten Ordner Dokumente vorliegen, die im System noch fehlen und importiert werden könnten.

Damit das KORA-System den Nutzern auch außerhalb der gemeinsamen Projektbesprechungen als Arbeitsplattform dienen kann, muss der Im- und Export von Dokumenten auch manuell angestoßen werden können. Insbesondere muss dies auch über das Internet möglich sein.

3.3.2.6. Rechteverwaltung

Für jede Art von Dokumentenmanagementsystem ist zu beachten, dass Dokumente eventuell vertrauliche Informationen enthalten oder dass sie urheberrechtlich zu schützen sind. In diesen Fällen wird der Zugriff über die Rechteverwaltung der Anwendung reglementiert oder gegebenenfalls verweigert. Der Zugang zu den Dokumenten wird über Access Control Lists (ACL) gesteuert. Einzelheiten hierzu wurden von Horst Mund [Mund 2006] erarbeitet.

3.3.2.7. Veröffentlichung von Dokumenten

Das Öffnen der auf dem Systemserver abgelegten Dokumente an einem der öffentlichen Wandbildschirme kann nur durch berechtigte Personen erfolgen. Für das Setzen der Rechte auf ein Dokument ist der Autor verantwortlich. Aus den Zugriffsrechten leiten sich weitere Restriktionen für die Verteilung und Synchronisation von Dokumenten ab:

- Ein Anwender kann nur Dokumente vom System auf sein mobiles Gerät ziehen, für die er ein entsprechendes Leserecht besitzt.
- Schreibrecht beinhaltet Leserecht.
- Konflikte, die durch legale Änderungen an Dokumenten entstehen, sind von den Nutzern selbst zu lösen. Solche Konflikte können entstehen, wenn zwei oder mehr berechtigte Personen außerhalb des Systems parallel Kopien des selben Dokuments bearbeitet und dieses dann zurück speichern wollen.

3.3.2.8. Anzeige der Ressourcen

Über die Benutzungsschnittstelle muss den Nutzern ein umfassender Überblick über die im System vorhandenen Ressourcen geboten sein. Zu den Ressourcen zählen die im System gespeicherten Dokumente, einschließlich der gespeicherten Metadaten.

Zur Visualisierung sollten hier zwei Ansätze zum Einsatz kommen. Zum einen sollte die Darstellung der Inhalte in Form einer Verzeichnisstruktur erfolgen. Diese Darstellungsform ist den meisten Computernutzern von ihren Betriebssystemen her vertraut und wird daher große Akzeptanz finden. Der zweite Ansatz basiert auf dem „Starting-Point-Ansatz“ [Eich 2006].

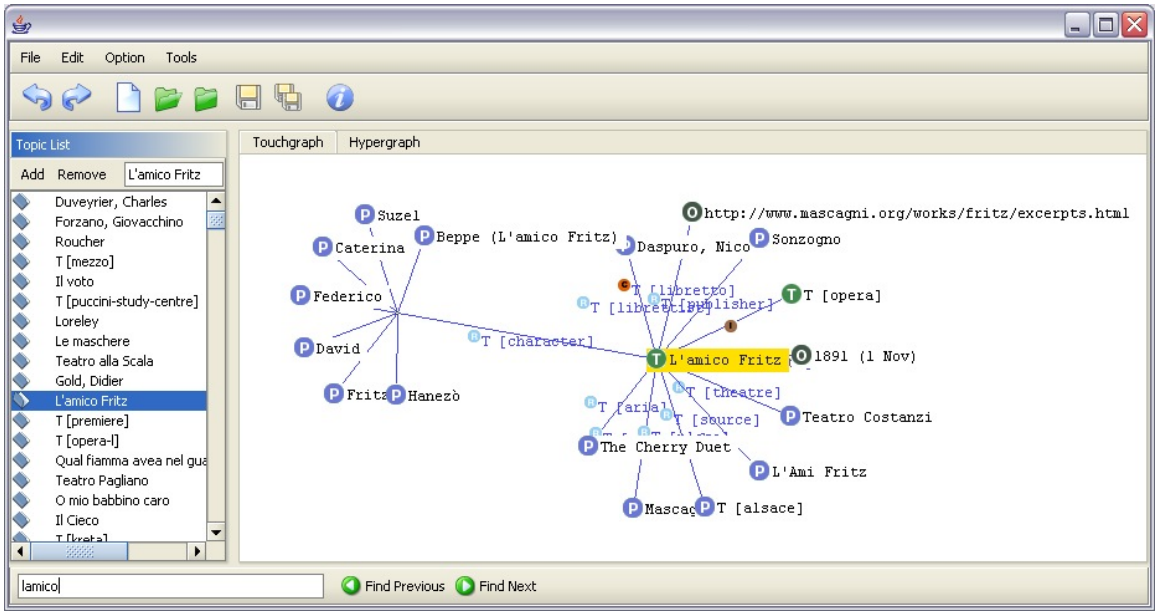

Abbildung 13: Durch den TouchgraphRenderer[15] dargestelltes Graphenmodell aus opera.xtm[16]

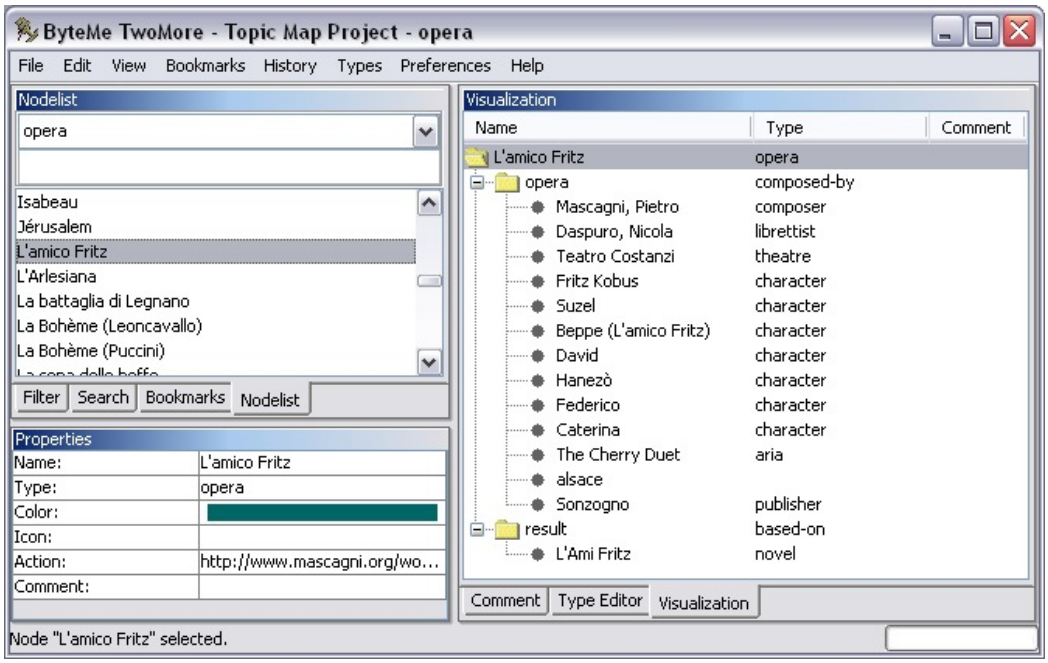

Abbildung 14: Mit TwoMore[17] als Baum dargestellter Ausschnitt aus opera.xtm

---

15 GraphRenderer aus der TMNav-Klassenbibliothek zur Visualisierung von TopicMaps

16 Von Steve Pepper zu Demonstrationszwecken erstellte, frei verfügbare XTM (http://www.techquila.com/topicmaps/tmworld/12066.html; http://www.ontopia.net/topicmaps/examples/opera/opera.xtm)

17 Open Source Visualisierungstool. "TwoMore" wurde Ende 2003 von einer Gruppe Studenten (ByteMe) der TU Darmstadt entwickelt

Der entscheidende Vorteil gegenüber Ordnerhierarchien liegt darin, dass die Suche nicht mit einem definierten Wurzelelement beginnend entlang eines festen Pfades geschieht, sondern auf beliebig strukturierten Wegen erfolgen kann. Visuell unterstützt wird dieser Ansatz durch den Einsatz einer grafischen Netzkomponente.

3.3.2.9. Erfassung von Metainformationen

Zur Charakterisierung digitaler Ressourcen mittels Metainformationen sind je nach Datentyp unterschiedliche Attribute denkbar. Daten wie Videos, Musikstücke oder Bilder werden zum Teil mit anderen Metadaten beschrieben wie Textdokumente. Eine Reihe vordefinierter Attribute, die zur Metabeschreibung von Text- und Bilddokumenten genutzt werden, basieren auf dem Dublin Core Standard. Dazu zählen u.a. Titel, Datum, Autor und Sprache.

Der zur inhaltlichen Beschreibung der Ressourcen verfügbare Satz an Attributen sollte jedoch flexibel an die jeweiligen Anforderungen angepasst werden können. Da diese Informationen vornehmlich zur Verarbeitung und weniger zur Repräsentation verwendet werden, spielt in diesem Zusammenhang der Gedanke der Standardisierung eine große Rolle [Eich 2006].

Um in Zukunft möglichst vielen bestehenden Applikationen das Lesen und Schreiben dieser Daten zu ermöglichen, ist der weit verbreitete und etablierte XML-Standard hierfür ein geeignetes Mittel. Eine breite Nutzung der vorgenannten Metadatentechnologien hat sich in der Dokumentenverwaltung zum jetzigen Zeitpunkt jedoch noch nicht eingestellt. Die Erstellung und Pflege der Metadaten könnte also zum „Flaschenhals“ des Systems werden.

Die Akzeptanz der Anwendung wird wesentlich davon abhängen, ob bei der Implementierung die Anforderungen eine einfache und intuitive Benutzbarkeit erreicht werden und ob der Einsatz der Software eine Steigerung der Arbeitseffizienz ermöglicht. Statt für organisatorische Zwecke soll mehr Zeit für produktive Arbeiten zur Verfügung stehen. Für die Erstellung der Metadaten bedeutet das, dass diese in Teilbereichen wenn möglich maschinell ausgelesen werden.

Technische Informationen, wie z. B. Dateiname, Dateiformat und Erstellungsdatum, die ohnehin für jede digitale Datei vorliegen, könnten unter Umständen transformiert und automatisiert in den repräsentierenden Knoten übertragen werden. Informationen, die auf die Bedeutung eines Dokuments abzielen, können nur manuell erstellt werden.

3.3.2.10. Informationssuche

Einzelne Inhalte müssen auch in einer Fülle von Daten und Informationen schnell und zuverlässig auffindbar sein. Der Prozess der Informationssuche setzt sich zusammen aus der Suche nach einem Einstiegsknoten im ersten Schritt und der weiteren Navigation über die Links zu benachbarten Knoten in der näheren Umgebung (Detailsuche).

Die Einstiegssuche soll ohne Angabe einer Datenquelle durch Textsuche über der Gesamtmenge der Knoten und Metadaten im Netz möglich sein. Dabei lässt sich die Ergebnismenge durch die Verwendung regulärer Ausdrücke und boolscher Verknüpfungen sinnvoll einschränken. Auf diese Weise lassen sich beispielsweise Suchanfragen folgender Art ausdrücken: „Finde alle Dokumente vom Typ X, bei denen das Attribut A dem Wert B entspricht.“

Ausgehend von der so ermittelten Ergebnismenge kann anschließend über eine grafische Komponente zu weiteren inhaltlich benachbarten Informationen navigiert werden. Ergänzend zur Suche über die Metadaten, könnte ein auf Volltextsuche basierender Suchalgorithmus zum Einsatz kommen, ähnlich dem indexbasierten Google-Suchdienst.

3.3.2.11. Bearbeitung von Dokumenten

Zentraler Anwendungsfall im Gruppenarbeitsraum ist das gemeinschaftliche Betrachten und Bearbeiten von Dokumenten an einem der öffentlichen Wandbildschirme oder an einem mobilen Gerät. Um dies zu ermöglichen, müssen erforderliche Werkzeuge und Anwendungen in die Infrastruktur des Gruppenarbeitsraums eingebunden werden.

3.3.2.12. Erstellung neuer Dokumente

Ähnlich signifikant wie das Bearbeiten von Dokumenten ist auch das Erstellen neuer Dokumente. Dies kann auf zweierlei Weisen geschehen. Entweder wird ein neues Dokument aus der Konferenzraumumgebung heraus erstellt, oder es wird an einem privaten PC oder mobilen Gerät erstellt und dann in das System importiert. In jedem Fall wird für das neue Dokument ein Netzknoten erzeugt. Hierfür wird vom System ein Dialogfenster geöffnet und der Nutzer wird aufgefordert, Metadaten zu editieren.

3.3.2.13. Verlinkung von Dokumenten

Ebenso bedeutsam wie die Schaffung und Bearbeitung von Inhalten und das Erfassen von Metadaten ist auch die Erstellung der Verknüpfungen zwischen den Dokumenten. Erst diese Links ermöglichen die semantische Navigation zu verwandten Informationen. Je besser dies genutzt wird und je vollständiger die Vernetzung der Dokumente ist, desto besser können später relevante Informationen durch Navigation im Datenbestand gefunden werden.

4. Existierende Lösungen

Derzeit wird in einer Reihe von Forschungs- und Entwicklungsprojekten an der Entwicklung *Persönlicher Informations- und Wissensmanagementsysteme* gearbeitet. Motivation dieser Projekte ist das Bestreben, die Auffindbarkeit existierender persönlicher Daten zu verbessern. Voraussetzungen dafür sind u. a. eine möglichst umfassende Einbeziehung, Verlinkung und Bedeutungszuordnung der vorhandenen persönlichen Informationen, die in einer Vielzahl unterschiedlicher Formate vorliegen.

Aufgrund der ähnlich gelagerten Problemstellung sollen in diesem Kapitel einige dieser Projekte vorgestellt werden. Im Wesentlichen werden hier drei unterschiedliche Ansätze verfolgt: Indexbasierte Suchsysteme, semantische Suchsysteme, ontologiebasierte Annotationswerkzeuge:

4.1. Indexbasierte Desktop-Systeme

Indexbasierte Systeme indizieren die auf dem Personal Computer gespeicherten Dokumente und nutzen den durch statistische Verfahren erzeugten Index, um Suchanfragen des Nutzer zu beantworten.

4.1.1. Google Desktop

Google Desktop Search[18] setzt moderne, auf Web-Suchmaschinentechnologie basierende, Suchfunktionalität in der PC-Umgebung ein. Google Desktop Search benutzt zur Indizierung des lokalen File-Systems Indexierungstechniken, die auf Erfahrungen der Internet-Indexierung basierenden. Dabei werden alle Dateien auf dem Computer, einschließlich Textdokumente ebenso auf einer Keyword-Basis indiziert wie E-Mails, Tabellenkalkulationen, Präsentationen, HTML-, PDF-, Grafik- oder Audiodateien.

Nach der Installation wird einmalig der komplette PC einer Indexierung unterzogen. Danach läuft im Hintergrund eine Indexierungs-Engine. Die Indexierungs-Engine überwacht alle Änderungen an Dokumenten und aktualisiert den Index laufend. Bei der permanenten Anpassung des Indexes werden nicht nur Dokumente und neu eingegangene E-Mails berücksichtigt, sondern auch vom Nutzer besuchte Internetseiten werden mit einbezogen und können bei einer Suche als Ergebnis zurückgeliefert werden.

Die Suchfunktionalität von Google Desktop Search basiert auf einer Volltextsuche. Die Anzahl der Resultate kann durch die Verwendung von Filtern und Suchoperatoren[19] eingeschränkt werden, z. B. Reduzierung der Resultate auf E-Mails.

---

[18] http://desktop.google.com/de/features.html

[19] Verknüpfung von Ausdrücken mit Hilfe boolscher Operatoren wie AND und OR

Abbildung 15: Google Desktop – Dateifilter

Angezeigt werden die Suchergebnisse im Browser. Ein Doppel-Klick genügt, um die Dokumente im entsprechenden Programm zu öffnen. Google Desktop Search unterstützt in der Basis-Version die gängigsten Dokumenttypen, wie zum Beispiel: MS Office (Word, Excel, PowerPoint, Outlook), Netscape Mail, Mozilla Mail, Thunderbird, Textdokumente sowie MP3, Grafik-, Audio- und Videodateien. Weitere Dokumentenformate können über ein AddIn-Konzept integriert werden.

4.1.2. MSN Suche Toolbar

MSN Suche Toolbar[20] ist das Desktop System der Firma Microsoft und basiert auf ähnlichen Prinzipien wie Google-Desktop. Daher soll es hier auch nicht im Detail vorgestellt werden.

MSN Suche Toolbar bietet als weitere Einschränkung der Suche die Einbeziehung von Keywords und Attributen an. So kann zum Beispiel nach Dokumenten eines bestimmten Autors gesucht werden. Durch den Einsatz von Keywords (Booleschen Ausdrücken) kann die Suche weiter verfeinert werden.

4.1.3. Spotlight

Spotlight[21] ist die Desktop-Variante der Firma AppleMacintosh und basiert ebenso wie der Google-Desktop und die MSN Suche Toolbar auf dem Prinzip der Indizierung des Personal Computers.

Ab der Version MAC OS 10.4 ist Spotlight fest ins Betriebssystem integriert. Suchergebnisse werden in einem neu erzeugten Fenster angezeigt. Die Ergebnisse werden in Kategorieren unterteilt und visualisiert, zum Beispiel nach Mail-Messages, Musik, Kontakte, Dokumente usw.

---

[20] http://desktop.msn.de/
[21] http://www.apple.com/macosx/features/spotlight/

Wie schon MSN Suche Toolbar, schließt auch Spotlight die Verwendung von Attributen in die Suche ein. Ein weiteres Feature von Spotlight bilden die s. g. „Smart Folders“. Diese Folder können vom Nutzer mit häufig verwendeten Suchkriterien verknüpft werden. Der Vorteil ist, dass sie ständig aktuell gehalten werden.

Für einen Smart Folder mit dem Suchkriterium „Präsentation“ würde dies bedeuten, dass hierin ein Verweis erzeugt wird, sobald eine neue Präsentation erstellt wurde.

4.1.4. Bewertung

Neben den genannten Produkten von Google, Microsoft und Apple werden zurzeit noch eine Reihe weiterer Desktop-Systeme entwickelt, die auf der gleichen Basistechnologie der Indizierung des Personal Computers basieren.

Alle Systeme verwenden einen Indexer, um die Dokumente des PCs zu scannen und so mit Hilfe einer Wortanalyse einen Index über alle Dokumente des Rechners zu generieren. Der Index wird genutzt, um Suchanfragen der Nutzer zu beantworten. Über die Verwendung von entsprechenden Filtern kann die Liste der Ergebnisse weiter eingeschränkt werden.

Unterschiede zwischen den Tools bestehen im Wesentlichen im Layout der Benutzungsschnittstelle und der Darstellung der Ergebnisse sowie bei größeren Datenmengen in der Effizienz der Indexer und der jeweiligen Suchmaschine.

4.2. Semantische Suchsysteme

Semantische Desktop-Systeme basieren auf Technologien, die im Zusammenhang mit der Entstehung des Semantic Web entwickelt wurden. Diese Tools verwenden Ontologien, um die auf dem PC gespeicherten Dokumente zu verwalten. Bei den Semantik-Desktop-Systemen gilt es, zwei unterschiedliche Ansätze zu unterscheiden a) den integrativen und b) den monopolistischen Ansatz.

Beim integrativen Ansatz werden „Plug-Ins“ und/oder Adapter verwendet, um bestehende Systeme, wie Textverarbeitungsprogramme, Mailprogramme oder Programme zum Erstellen von Präsentationen oder Grafiken, mit dem Semantik-Desktop zu verbinden. Bei diesem Ansatz stehen die Applikationen mehr im Mittelpunkt, denn sie werden verwendet, um die jeweiligen Dokumente zu bearbeiten.

Beim monopolistischen Ansatz stehen die Informationen und nicht die Anwendung mit der die Information erstellt wurde im Mittelpunkt. D. h. Daten werden in einem Standardformat z. B. XML oder RDF gespeichert und können so eher gemeinsam genutzt, zusammengeführt oder vereinheitlicht werden. Die vorherrschenden proprietären Lösungen auf Dateiebene lassen in der Regel eher ein Kopieren denn ein Referenzieren der Daten zu, was unumgänglich zu Konsistenzproblemen führt .

4.2.1. Haystack

Haystack[22] ist ein Semantik-Desktop-System, das den monopolistischen Ansatz verfolgt. Als Konsequenz vereint Haystack viele Applikationen in sich und ersetzt dadurch Programme, wie Textverarbeitungsprogramme, E-Mail und Messengingprogramme oder Programme zur Bildbearbeitung. Auf diese Weise ist der Nutzer nicht mehr gezwungen die Applikation zu wechseln. Er könnte zum Beispiel eine per E-Mail empfangene Präsentation direkt in Haystack bearbeiten und anschließend ebenfalls direkt mit Haystack wieder per E-Mail an andere Personen weiterleiten.

Mit diesem Ansatz möchte Haystack dem Nutzer die Möglichkeit bieten, seine Informationen auf eine Weise zu verwalten, die ihm am meisten nutzt. Der Wegfall willkürlicher applikationsbedingter Grenzen, die entstehen, wenn aufgrund unterschiedlicher Dokumententypen unterschiedliche Programme für die Bearbeitung eingesetzt werden müssen, soll dem Nutzer eine effektivere Verwaltung und einheitlichere Sicht seiner Daten ermöglichen.

Während in der Vergangenheit Informationen zwischen E-Mail-Client(s), Filesystemen, Kalender, Adressbüchern, dem Web und anderen spezifischen Behältern verstreut waren, löst Haystack diese Trennung auf, so dass Nutzer mit einer vereinheitlichten Art ihrer Informationen arbeiten. Eine derartige Personalisierung[23] des Informationsmanagement kann die Chancen des Nutzers, das zu finden, was er wirklich sucht und braucht, dramatisch verbessern.

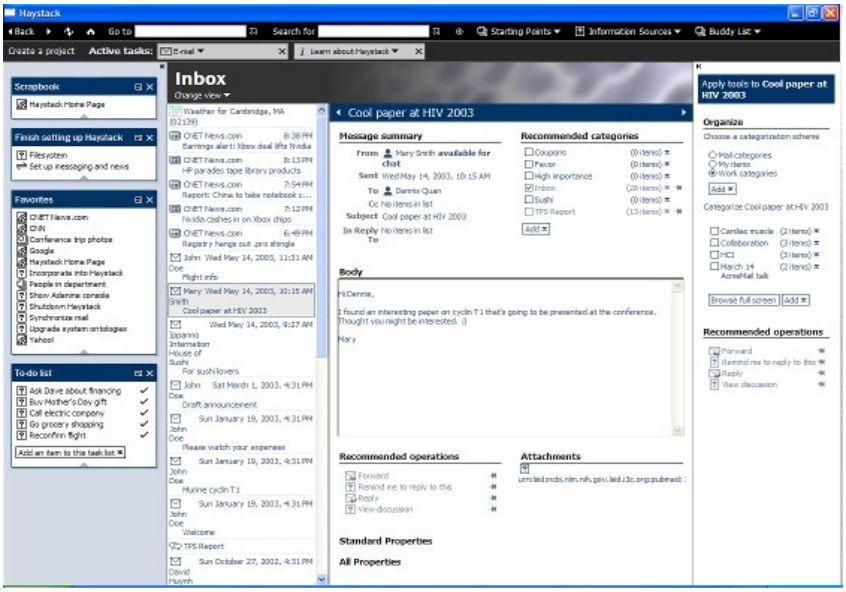

Abbildung 16: Haystack Oberfläche

Innerhalb des Haystack-System stehen die Information bzw. das Dokument im Vordergrund ebenso wie die Operationen, die mit dem Dokument ausgeführt werden können, z. B. Rechtschreibprüfung, Senden einer E-Mail, Drehen eines Bildes. Alle Metadaten über die

---

[22] Haystack wird vom Computer Science and Artificial Intelligence Laboratory am Massachsets Institute of Technology entwickelt;

[23] Hinter Personalisierung verbirgt sich der Grundgedanke, nur diejenigen Informationen anzubieten, die auf die persönlichen Interessen eines Users zugeschnitten sind.

Dokumente, die in Haystack zusammengeführt werden, sind innerhalb des Systems vereint. Informationen die in einem Zusammenhang stehen, können durch entsprechende Operationen verlinkt und zusammen verwaltet werden.

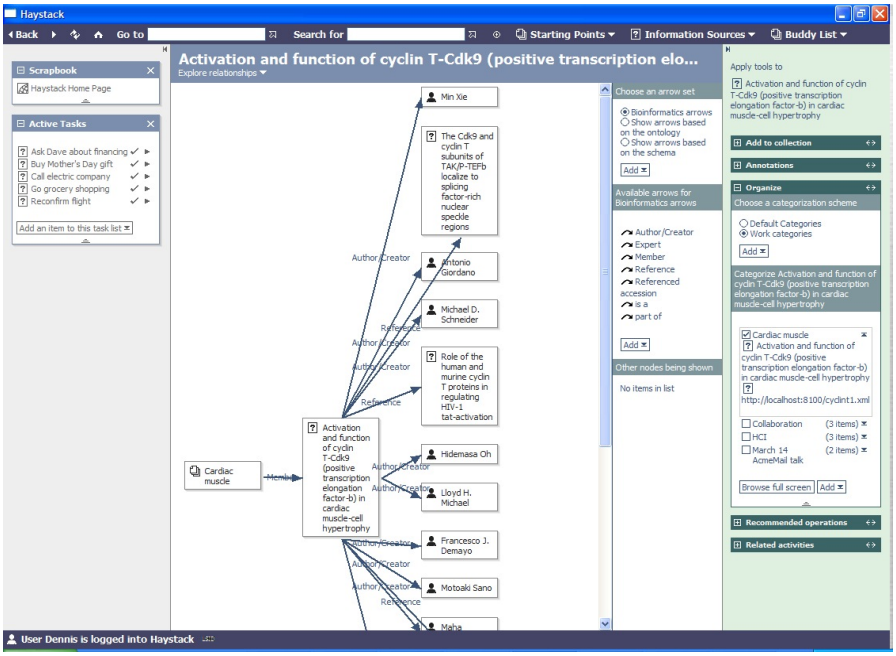

Abbildung 17: Haystack Beziehungen

Das System kann durch seine modular konzipierte Architektur leicht um neue Informations- und Dokumenttypen aber ebenso um neue Operationen erweitert werden.

4.2.2. IRIS Semantik Desktop

IRIS[24] ist ein Open Source Application-Framework, das Nutzern die Möglichkeit bieten soll, eine persönliche „Landkarte“ über ihre digitalen Daten zu erstellen. IRIS ist die Abkürzung für “Integrate. Relate. Infer. Share.” Die Hauptansätze des Systems werden bereits durch das Akronym beschrieben:

- *Integrate*: IRIS vereint vielfältige unabhängige Anwendungen, wie E-Mail (Mozilla), Internetexplorer (Mozilla), Dateimanager, Kalender (OpenOffice) und Chat (Japper).
- *Relate*: IRIS speichert die Daten als ontologiebasierte KB (Knowledge Base), die die Darstellung und Verknüpfung von Objekten des Arbeitsalltags unterstützt. In IRIS können Phrasen folgender Art ausgedrückt werden: „Suche Datei X, geschrieben von Y, präsentiert beim Meeting Z, über das Projekt T“.
- *Infer (folgern)*: IRIS integriert einen *online learning algorithmus,* der Daten eigenständig zum Beispiel klassifiziert, clustert, priorisiert, extrahiert und assoziiert. Zusätzlich zum automatisierten Lernen aus dem Beobachten der Benutzeraktivitäten, übernimmt IRIS über eine grafische Benutzungsschnittstelle auch Feedback der Nutzer.

[24] http://www.openiris.org

§ *Share*: Für die Zukunft ist geplant, dass das in IRIS erlangte Wissen eventuell auch verteilt genutzt werden kann.

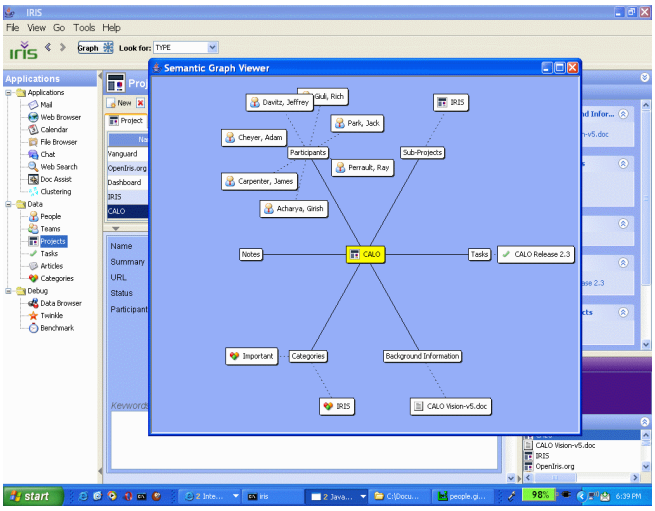

Abbildung 18: IRIS Semantic Graph

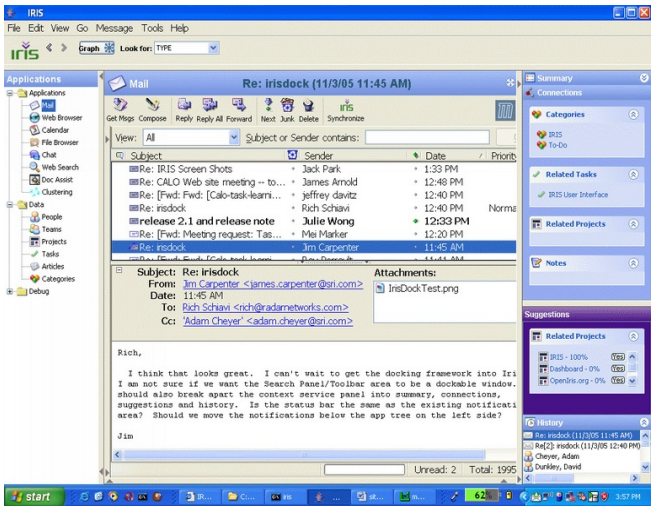

Abbildung 19: IRIS Mail

4.2.3. Gnowsis

Im Gegensatz zu Haystack und IRIS, verfolgt Gnowsis[25] den integrativen Ansatz eines Semantik-Desktops. Das heißt, die Daten, mit denen die Nutzer des Gnowsis arbeiten, kommen aus verschiedenen existierenden Anwendungen. E-Mails, Textdokumente, Kontakte (Adressbuch) und Kalender sind solche Datenquellen.

Basierend auf den Techniken und Erfahrungen des WWW, insbesondere unter Verwendung von Semantic-Web-Standards, verfolgt Gnowsis den Ansatz, Daten aus unterschiedlichen Quellen (SQL-Datenbanken, Office-Applikationen und andere gebräuchliche Büro-Anwendungen) zu integrieren.

Als erster Schritt zum Semantic-Desktop war hier zunächst ein Umdenken in der Philosophie nötig. Aus heutiger Sicht werden auf Desktop-PCs Dateien gespeichert. Aus Semantic-Desktop-Sicht werden die auf PCs gespeicherten Daten jedoch als Web-Dokumente auf einem Web-Server angesehen. Jede typische Datei auf einem Personal Computer (Dokument, Bild, E-Mail, Video, etc.) wird einfach als Ressource verstanden.

Den Kern des Gnowsis Semantic-Desktop-Systems bildet daher ein lokaler Webserver, der durch Semantic-Web-Technologien und durch Gnowsis-Implementierungen erweitert wird. Aufgabe des Webservers ist es, die Daten und Funktionen der unterschiedlichen Programme zu veröffentlichen und Softwareagenten den Zugriff auf diese Daten zu gewähren.

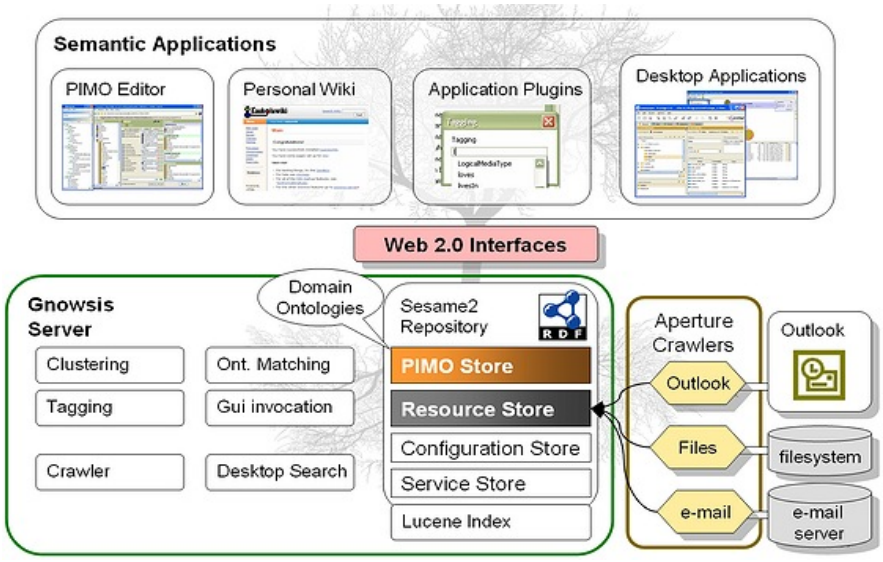

Abbildung 20: Gnowsis Architektur

Genau wie im WWW spielt auch beim Semantik-Desktop die Identifikation der Ressourcen eine entscheidende Rolle. Hierfür wird das bewährte Prinzip der URL *Uniform Resource Locator*, eine Untermenge der URI *Uniform Resource Identifier*, eingesetzt.

---

[25] http://www.gnowsis.org/

Ziel der Semantik-Desktop-Implementation ist eine Datenintegration mit der es möglich wird, alle Daten des PCs in einer Anwendung zusammenzuführen. Alle strukturierten Daten, die über eine URI identifiziert werden können, sind als RDF[26] ansprechbar und sollen durch einen RDF-Graphen beschrieben und integriert werden. Die Repräsentation der Daten als RDF erlaubt das Hinzufügen von Bedeutungen.

Gnowsis nutzt Ontologien zur Integration von Desktop-Applikationen. Dienste müssen die von ihnen verwalteten Daten mittels in RDF oder OWL ausgedrückten Ontologien beschreiben. Auf diese Weise wird es möglich, Dokumente über Applikationsgrenzen hinaus zu verbinden und einen geschlossenen Informationsraum zu erzeugen, der alle darin enthaltenen Dokumente verbindet.

4.2.4. Bewertung

Bei den vorgestellten semantischen Suchsystemen Haystack und IRIS liegt der Schwerpunkt auf der Verwaltung und Bearbeitung der Informationen und Dokumente unter einer einheitlichen Oberfläche. Dieser Ansatz erfordert allerdings, dass der Benutzer die Funktionalität des Systems nutzt und auf seine bisher verwendeten und gewohnten Tools verzichtet. Haystack und IRIS unterstützen zwar die Verknüpfung von Objekten; Annotation und Anreicherung von Dokumenten mit Metadaten werden jedoch nicht unterstützt.

Gnowsis ist ähnlich wie Haystack und IRIS ein Desktop-System mit dem es möglich ist, Dokumente semantisch zu verlinken. Die Anreicherung der Dokumente mit zusätzlichen Metadaten ist aber auch hier nicht möglich.

[26] RDF (Resource Description Framework) formale Sprache zur Bereitstellung von Metadaten im WWW

4.3. Ontologiebasierte Annotationswerkzeuge

Der Vollständigkeit halber sei hier auch noch die Gruppe der ontologiebasierten Annotationswerkzeuge vorgestellt. Diese Tools nutzen bestehende Ontologien, um automatisch oder halbautomatisch Dokumente mit Meta-Daten anzureichern. Dies geschieht z. B. dadurch, dass in das zu annotierende Dokument neue XML-Tags eingefügt werden, die das ausgewählte Textelement mit einem Attribut der Ontologie verbinden.

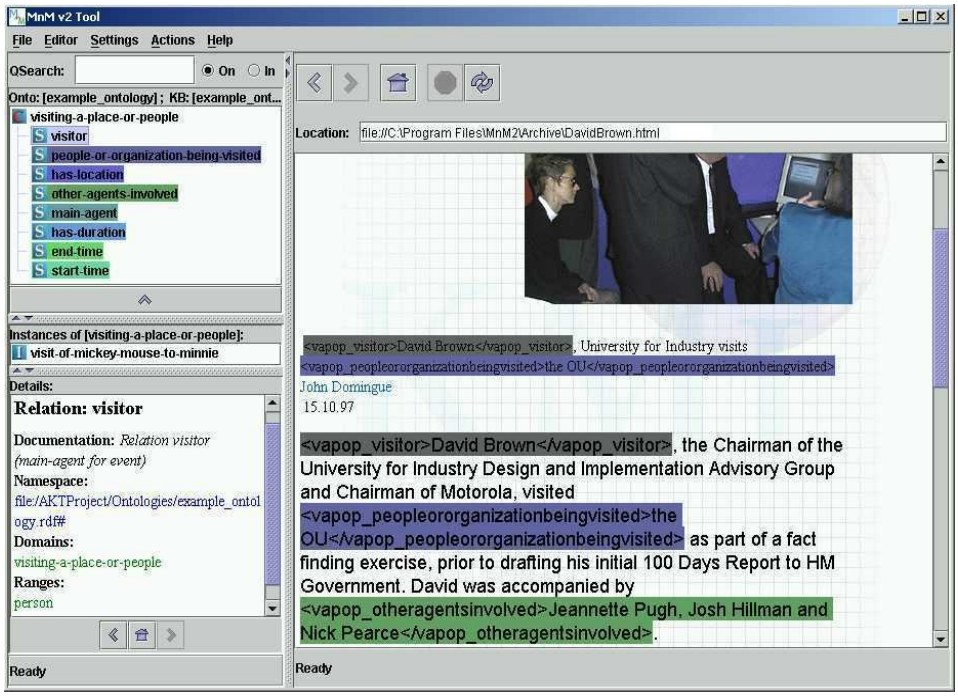

Abbildung 21: MnM: Integrierter Dokumentbrowser zur Annotation eines Dokuments

Wie auch schon in den vorhergehenden Abschnitten sind auch in dieser Kategorie, außer dem in Abbildung 21 abgebildeten MnM[27], eine Vielzahl ähnlicher Werkzeuge verfügbar, z. B. das COHSE (Conceptual Open Hypermedia Service Environment), ein Gemeinschaftsprojekt der Manchester Information Management Group (IMG) und der Southampton's Intelligence, Agents and Multimedia (IAM) group oder OntoMat, ein Produkt des Institut AIFB der Universität Karlsruhe.

[27] MnM wird entwickelt am: Knowledge Media Instiute, The Open University in WIten Hall (UK), http://kmi.open.ac.du/projects/akt/MnM/index.html

4.4. Zusammenfassung

Die vorgestellten indexbasierten Desktop-Systeme GoogleDesktop, MSN Suche Toolbar und Spotlight haben allesamt den großen Vorteil, dass sie auch von unerfahrenen Nutzern leicht zu bedienen sind und auf eine Anfrage schnell viele Resultate liefern.

Wie auch schon bei der Google Web-Suche wird auch hier ein Wortvergleich durchgeführt, d. h. eine Suchanfrage liefert ggf. zahlreiche Resultate die zwar das gesuchte Wort enthalten, aber nicht dem gesuchten Kontext angehören. Andere relevante Dokumente werden möglicherweise nicht gefunden, weil sie das gesuchte Wort nicht direkt enthalten, z. B. Eltern, statt Vater und Mutter.

Interessante Ideen für das geplante Konferenzraum-Dokumentenmanagementsystem bieten die vorgestellten semantischen Suchsysteme mit dem Konzept, Informationen und Objekte miteinander zu verknüpfen. Insbesondere der Semantic Desktop Gnowsis liefert mit der Philosophie, alle gespeicherten Daten als Web-Dokumente auf einem Web-Server anzusehen, gute Ansätze.

Manko aller vorgestellten Systeme ist jedoch, dass sie keinerlei Unterstützung bieten, um Dokumente mit zusätzlichen Metadaten zu beschreiben.

Die letzte Gruppe, die Annotationswerkzeuge beschränken sich ausschließlich auf die Annotation von HTML-Seiten, was für den angestrebten Anwendungskontext jedoch nicht ausreichend ist, da möglichst alle Dokumenttypen eingezogen werden sollen.

5. Design und Realisierung

Aspekte der technischen Realisierung und der Infrastruktur des Konferenzraums wurden bereits von [Bart 2006] und [Burf 2006] beschrieben. Im nachfolgenden Kapitel dieser Arbeit wird nun die Software-Architektur des KORA-Dokumentenmanagementsystem vorgestellt. Entwickelt werden soll eine Benutzungsschnittstelle für die benutzungsfreundlich strukturierte Ablage von Informationen. Die Repräsentation der Informationsobjekte soll vor allem dazu dienen, die Suche und Navigation innerhalb des Datenbestandes zu verbessern.

5.1. Fachliche Architektur

Aufgrund der bisherigen Analysen kann nun die Architektur des Systems festgelegt und das Gesamtsystem in Subsysteme aufgeteilt werden. Wie in Abbildung 22 zu sehen ist, kann das System in einzelne Module aufgeteilt werden. Die Aufteilung ergibt sich im einzelnen aus folgenden Überlegungen:

1. Der Konferenzraum soll ein Dokumentenmanagement (Modul I) erhalten
2. Dieses gliedert sich in Subleistungen zur Informationsgewinnung „DataExtraction“ und Informationsauffindung „DataRetrieve“ (Modul II-a und Modul II-b)
3. Diese Module bestehen wiederum aus verschiedenen Basisdiensten, die nach Bedarf genutzt werden. Hierzu zählen Metadatenextraktion, Kanten- und Knoteneditoren (Komponente III-a bis III-d).
4. Außerdem kommen externe Komponenten zum Einsatz, die das System vervollständigen, z. B. OntologieEditor oder Werkzeuge zum Bearbeiten der Dokumente.

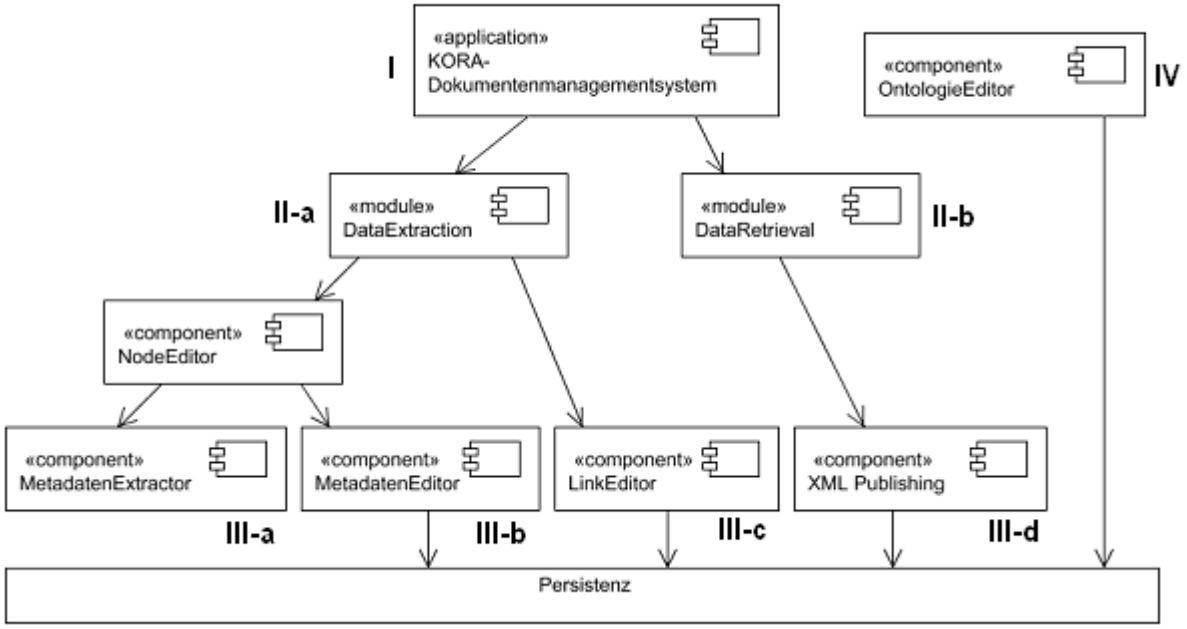

Abbildung 22: Komponenten-Diagramm

5.1.1. Komponenten

Die einzelnen Komponenten des KORA-Dokumentenmanagement werden in den nachfolgenden Abschnitten im Einzelnen ausführlicher beschrieben.

5.1.1.1. DataExtractionModule

Das DataExtractionModule implementiert Funktionalitäten zum Dateiimport und zum Aufbau des Wissensnetzes. Damit eine möglichst umfassende Repräsentation des im KORA-Dokumentenmanagementsystem vorhandenen Wissens erfolgen kann, muss sichergestellt sein, dass keine Dokumente ins System gelangen, ohne dass hierfür eine Repräsentation im Wissensnetz erzeugt wird. Mit anderen Worten, ein Nutzer kann ein Dokument nur importieren, wenn hierfür ein Knoten erzeugt werden kann. Für die Erzeugung eines solchen Knoten können erforderliche Metadaten entweder aus der Datei extrahiert werden oder der Nutzer muss fehlende Informationen mit Hilfe des `MetadatenEditor`'s nachpflegen (Abbildung 24).

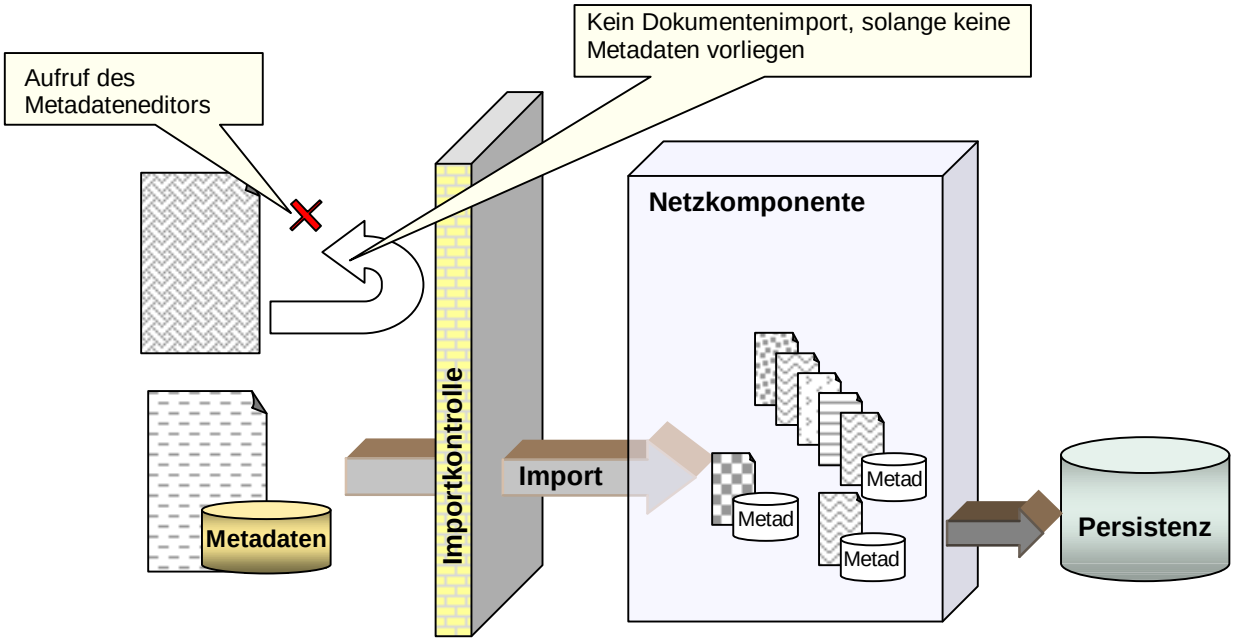

Abbildung 23: Dokumentenimport

Abbildung 23 zeigt schematisch das Prinzip der Importkontrolle. Die Importkomponente prüft, ob erforderliche Metadaten vorliegen. Das ist der Fall, wenn für dieses Dokument bereits ein Knoten existiert. Andernfalls muss innerhalb der Netzkomponente zunächst ein Knoten erstellt werden. Die hierfür erforderlichen Metadaten werden, wie nachfolgend beschrieben, ermittelt.

Der Prozess der Metadatengewinnung unterteilt sich in zwei Aktivitäten. Im ersten Schritt werden vom `MetadatenExtractor` Metadaten aus der Datei extrahiert. Im zweiten Schritt werden diese Informationen im `MetadatenEditor` angezeigt und können manuell vom Nutzer editiert und ergänzt werden, bevor sie dann in das Wissensnetz, also in die Topic Map, übernommen werden. Dieser Prozess der Metadatengewinnung wird in Abbildung 24 und Abbildung 25 detaillierter dargestellt und beschrieben.

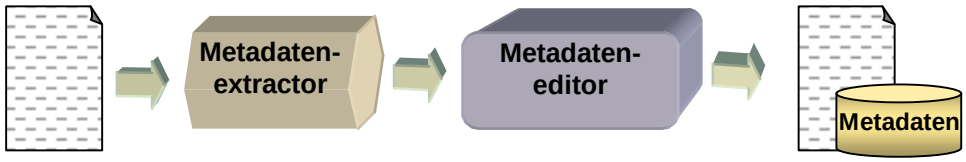

Abbildung 24: Prozess der Metadatengewinnung

MetadatenExtractor

Nachdem der Nutzer über die GUI den Import einer Datei angestoßen hat, wird diese dem `MetadatenExtractor` übergeben. Abbildung 25 veranschaulicht den Ablauf der Metadatenextraktion. Der `MetadatenExtractor` liest die Datei ein und extrahiert, sofern möglich, die darin enthaltenen Metadaten, wie Autor, Erstellungsdatum, Thema, usw. Die so gewonnenen Informationen übergibt er an den `MetadatenEditor`.

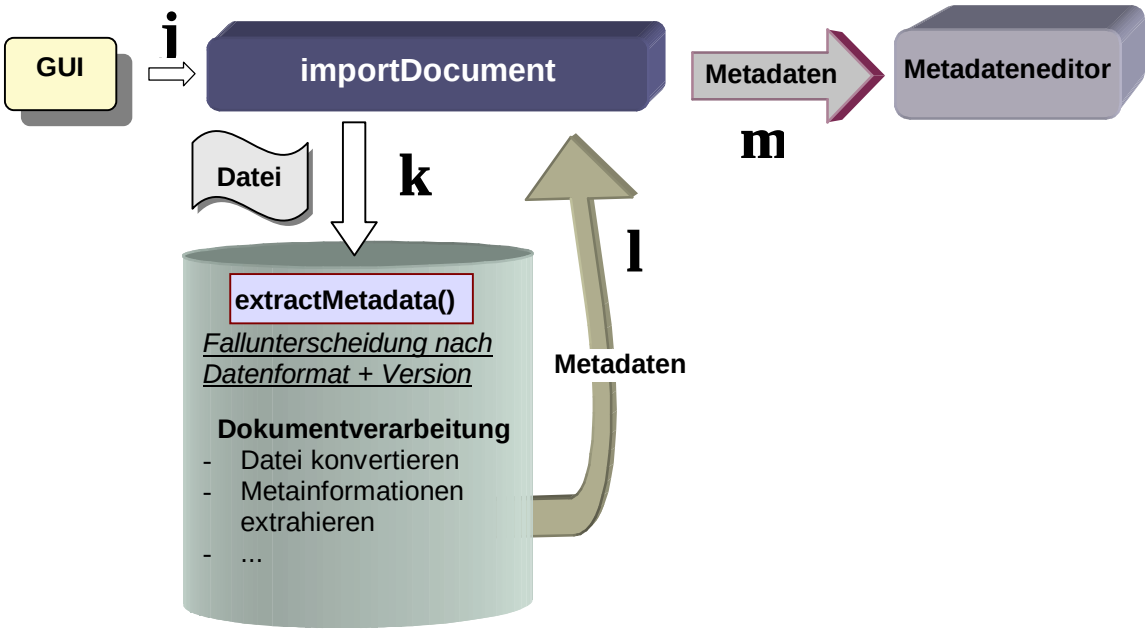

Abbildung 25: Prozess der Metadatenextraktion

Um das System bezüglich neuer Dateitypen erweiterbar und flexibel zu halten, ist an dieser Stelle eine Plugin-Architektur erforderlich. Bei Bedarf können auf diese Weise jederzeit neue Komponenten hinzugefügt werden, die die Extraktion von Metadaten aus neuen Dateitypen übernehmen.

MetadatenEditor

Der `MetadatenEditor` übernimmt die maschinell extrahierten Metadaten vom `MetadatenExtractor`, zeigt sie dem Nutzer in einem Formular an, nimmt Benutzereingaben entgegen, validiert gegebenenfalls die Eingaben und übergibt die Daten an den Knoteneditor (`NodeEditor`).

NodeEditor (Knoteneditor)

Der NodeEditor erstellt innerhalb der Netzrepräsentation aus den Attributdaten einen Knoten, d. h. er erzeugt ein Topic-Element in der Topic Map. Eine weitere Aufgabe des `NodeEditor` besteht darin, bei Bedarf, wenn Knoten bearbeitet werden müssen, die entsprechenden Topic-Elemente aus der Topic Map auszulesen und die Informationen an den `MetadatenEditor` zu übergeben.

LinkEditor (Kanteneditor)

Der `LinkEditor` ist die Komponente zur Erzeugung und Bearbeitung der Kanten (Associations) innerhalb der Netzrepräsentation.

5.1.1.2. DataRetrievalModule

Das `DataRetrievelModule` besteht im wesentlichen aus zwei Komponenten. Eine Komponente, die Daten der Netzrepräsentation so aufbereitet, dass sie von einer Client-Anwendung (z.B. einem Browser) angezeigt werden können. Die zweite Komponente hat die Aufgabe, ein Datenobjekt, sprich ein Dokument, das in der Wissensrepräsentation vom Nutzer ausgewählt wurde, zu identifizieren, so dass der Client des Nutzers die Daten von der Datenquelle abrufen kann. Auf diese Weise wird eine einheitliche, von der Datenquelle unabhängige Schnittstelle angeboten. Die eigentlichen Daten werden so zurückgegeben, wie sie auf der Datenquelle vorliegen.

5.1.2. Visualisierung

Während für die Darstellung einfacher Repräsentationsstrukturen, wie Verzeichnishierarchien, die Generierung eines zweidimensionalen Abbildes genügt, ist für die Visualisierung komplexer Strukturen ein mehrstufiger Prozess notwendig. Dieser Prozess wird Visualisierungspipeline bezeichnet [Schu 2000]. Die Pipeline umfasst, ausgehend von den Rohdaten, die Phasen Datenaufbereitung (Filtering), Datenabbildung (Mapping) und Bildgenerierung (Rendering).

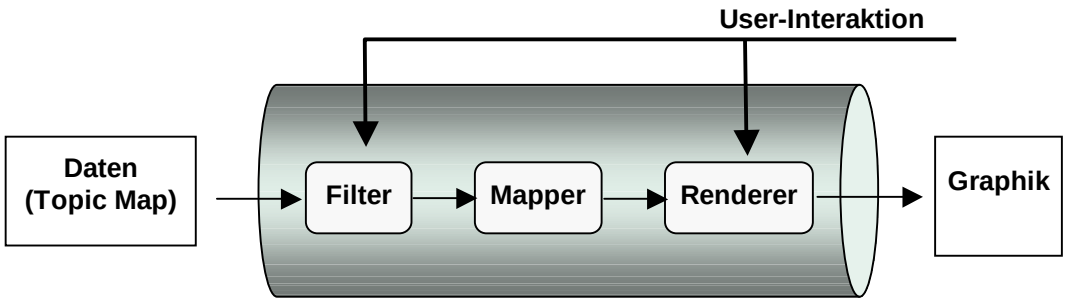

Abbildung 26: Stufen der Visualisierungspipeline

Der erste Schritt bei der Visualisierung einer Wissensstruktur besteht darin, die Menge der darzustellenden Knoten einzugrenzen. Dies geschieht in der Regel durch den Einsatz von Suchfiltern. In anderen Anwendungsumfeldern kann die Phase der Datenaufbereitung auch Operationen wie Strukturierung und Vervollständigung einer Datenmenge beinhalten.

Im zweiten Schritt der Visualisierungspipeline, dem Mapping, werden die zuvor aufbereiteten Daten in Geometriedaten überführt. Das heißt, es wird eine Daten-zu-Geometrie-Abbildung realisiert. Hierbei werden die Datenwerte auf geometrische Primitive, einschließlich zugehöriger Attribute (z.B. Farbe oder Position) abgebildet [Schu 2000].

Der letzte Schritt der Visualisierungspipeline realisiert die Abbildung der Geometriedaten auf Bilddaten. Das heißt, die geometrische Repräsentation der Daten wird durch ein Graphikmodul (Renderer) durch Projektion auf eine Bildebene in ein Rasterbild abgebildet. Je nach Anwendungsbereich sind verschiedene Darstellungsarten denkbar. Die Art der Darstellung wird in der Regel schon im Mappingschritt festgelegt. Sie beeinflusst die Wahl des Graphikpakets, das in der Lage sein muss, das entsprechende Bild auch zu erzeugen [Schu 2000].

5.2. Technische Architektur

In den folgenden Abschnitten soll die technische Architektur einer Client-Server-Architektur zur Benutzungsschnittstelle des KORA-Dokumentenmanagementsystems vorgestellt werden. Während von [Bart 2006] und [Burf 2006] verschiedene Middleware-Architekturen für die Konferenzraumsoftware betrachtet wurden, wird hier im weiteren Verlauf ein Entwurf auf Basis aktueller Web-Technologien ausgearbeitet.

5.2.1. Systemaufteilung

Bevor eine konkrete Architekturentscheidung gefällt wird, werden zunächst die unterschiedlichen Verteilungsansätze einer Client-Server-Architektur betrachtet. Diese reichen von Terminal-basierten Lösungen über HTML-basierte Anwendungen bis hin zu Fat-Clients (Abbildung 27).

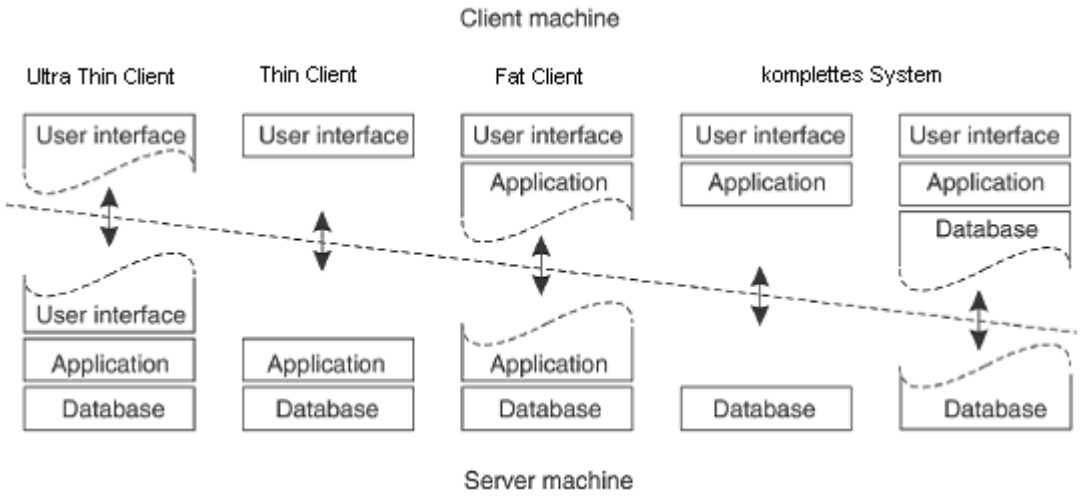

Abbildung 27: Verteilungsformen der Client/Server-Architektur [Tan 2002]

Client-Server-Architektur

Die Client-Server-Architektur ist ein Systemdesign bei dem die Funktionen zwischen Clientanwendung und Serveranwendung aufgeteilt werden. Die Verwaltung der Daten erfolgt dabei meist zentral auf der Serverseite, was Redundanzen vermeidet, während die Anwendungslogik je nach Ausprägung zum Teil oder sogar ganz auf dem Client ausgeführt wird. Die heutigen Client-Server-Anwendungen sind charakterisiert durch eine starke Server-Zentrierung mit Thin-Clients. HTML-basierte Anwendungen sind ein klassischer Vertreter dieser Gattung [Schä 2003].

Thin-Client-Architektur

Das Konzept des Thin-Client bezeichnet innerhalb einer Client-Server-Architektur eine Anwendung oder einen Computer als Endgerät, dessen funktionale Ausstattung auf die Ein- und Ausgabe beschränkt ist. Der Thin-Client steht damit zwischen dem Konzept des Fat-Client und dem des Ultra-Thin-Client, der eine Terminal-basierte Lösung darstellt.

Die Kernaufgaben des Thin-Clients bestehen in der Präsentation der Inhalte. Der Client verfügt über eine GUI mit Funktionalität und Möglichkeiten zur Steuerung, durch integrierte Dialogsteuerungslogik und Prüffunktionen. Die Aufgaben des Servers bestehen in der Ausführung der Applikations- und Businesslogik und der Verwaltung der Daten.

Die Vorteile der Thin-Client-Architektur lassen sich wie folgt zusammenfassen:

- Konsequente Umsetzung des MVC-Konzepts, durch Trennung von Logik und Präsentation möglich. Auf dem Client befindet sich nur die View. Controller und Model befinden sich auf dem Server
- Wiederverwendbarkeit der Anwendungslogik
- Verbesserte Wartbarkeit und Steigerung der Entwicklungseffizienz. Änderungen der Anwendungslogik erfolgen nur an einer Stelle, dem Server, und stehen allen Clients gleichzeitig zur Verfügung.

Ein Nachteil dieser Architektur kann sich allerdings daraus ergeben, dass sämtliche Rechenleistungen auf wenigen Servern ausgeführt werden und das sämtliche Daten über das Netzwerk transportiert werden. Dies kann bei einer großen Anzahl von Clients mit datenintensiven Aktionen zu einer hohen Netzbelastung und somit zu Performanzeinbußen führen. Ein weiterer Nachteil besteht darin, dass diese Form der Architektur eine fortwährende Netzverbindung voraussetzt.

Fat-Client-Architektur

Bei einer Client-Server-Architektur wird die Bezeichnung Fat-Client oder Rich-Client für einen Client verwendet, bei dem die eigentliche Verarbeitung der Daten vor Ort auf dem Client erfolgt. D. h. Teile der Businesslogik, der Datenprüfung oder die Konsistenzhaltung der Vorgänge einer Anwendung, liegen auf dem Client. Dadurch wird der Server entsprechend entlastet. In Client-Server-Architekturen mit Fat-Clients liefern diese einen Großteil der Prozessorleistung.

Ergonomisch weisen Fat-Clients erhebliche Unterschiede zu Web-Clients und hierbei vor allem zu reinen HTML-Clients auf Dies äußert sich unter anderem darin, dass sie in der Regel über einen wesentlich höheren Funktionsumfang verfügen [Zöll 2006][28].. Hierzu zählen z. B.:

- Drag & Drop
- Tastatur-Shortcuts
- Individuell einstellbare Benutzungsoberfläche
- Fat-Clients erreichen in der Regel eine bessere Konformität zum Windows-Style-Guide

Nachteile gegenüber einem Thin-Client ergeben sich durch hohe Redundanz zwischen Client und Server und hohen Administrations- und Integritätssicherungsaufwand. Außerdem entstehen durch entsprechenden Hardwarebedarf auf der Client-Seite hohe Kosten.

Zusammenfassung

Beide Technologien haben sowohl ihre Vor- als auch Nachteile. Welche Technologie letztendlich zum Einsatz kommen sollte, hängt im wesentlichen von den Anforderungen an das zu entwickelnde Produkt ab. Mit JavaScript und Java-Applets stehen Konzepte zur Verfügung, mit denen sich dynamische Elemente in statische HTML-Dokumente einbinden lassen, so dass funktionale und intuitive Browser-Anwendungen erstellt werden können, die sich nur noch wenig von reinen Desktop-Anwendungen unterscheiden.

Der Einsatz von JavaScript erlebt derzeit unter dem Akronym Ajax (**A**synchronous **J**avaScript **a**nd **X**ML) einen regelrechten Hype. Doch es werden auch kritische Stimmen laut, die davor warnen, dass sich dies schon bald als technologischer Holzweg entpuppen könnte [Müll 2006}[29]. Wenn aus einem „interaktiven Frontend“ ein komplexes, in JavaScript codiertes, Anwendungsprogramm wird, wächst die Gefahr der konkurrierenden Anwendungslogik zwischen dem Client und dem Server und damit die sich ergebenden Probleme.

---

[28] Bernhard Zöller: Nach langjähriger Tätigkeit als Berater im Bereich Dokumenten-Management gründete er Anfang 1997 die Zöller&Partner GmbH als neutrale, produkt- und anbieterunabhängige Beratungsfirma. Bernhard Zöller ist langjähriges Mitglied der AIIM und Vorstand im VOI.

[29] Björn Müller: Leiter des Bereichs "XMLi User Interfaces", bei der Software AG, der u.a. eine AJAX-basierte Frontend-Lösung für interaktive Unternehmensanwendungen verantwortet. Mit seiner Firma Casabac (heute Teil der Software AG) war Björn Müller einer der frühen AJAX-Pioniere in Deutschland.

5.2.2. Web-Applikationen

Als Web-Applikationen bezeichnet man Anwendungen, deren Benutzungsschnittstellen über das WWW (World Wide Web) bereitgestellt werden. Hierbei handelt es sich um verteilte Anwendungen, wobei sowohl Komponenten im Internetbrowser als auch auf Servern verwendet werden können. Webanwendungen unterscheiden sich von konventionellen statischen Internetseiten dadurch, dass sie dynamischen Response erzeugen. Eine Webanwendung kann mit Datenbanken und mit Fachlogik-Komponenten interagieren, um die Antwort anzupassen.

Allen serverseitigen Web-Technologien ist gemeinsam, dass Programme auf Informationen aus dem http-Request zugreifen können. Am Ende der Abarbeitung des jeweils spezifischen Programmcodes muss das serverseitige Programm einen Response generieren und diesen an den Web-Server liefern.

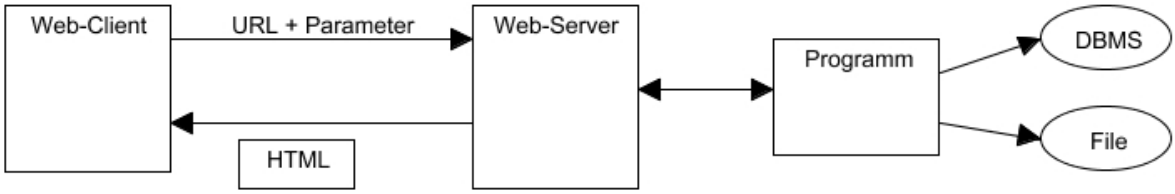

Abbildung 28: Architektur für dynamisch erzeugtes HTML durch serverseitige Programme

Webanwendungen lassen sich von anderen, ähnlichen Technologien gut abgrenzen, da die Interaktion mit dem Benutzer per Definition ausschließlich über einen Browser erfolgt. Auf dem Computer des Benutzers wird nur ein Webbrowser vorausgesetzt, was allgemein als gegeben angesehen werden kann.

Im Gegensatz zu herkömmlichen Client-Server-Anwendungen ist somit keine weitere Installation von Software auf den Computern der Benutzer notwendig (abgesehen von PlugIns, wie Flash-Player oder SVG-Viewer). Hierdurch erreichen Webanwendungen einen hohen Grad an Plattformunabhängigkeit, wenn bei der Entwicklung darauf geachtet wird, dass alle Browser unterstützt werden.

Durch die stetige Ausweitung der Browsertechnologie auf andere Endgeräte, wie Mobiltelefone oder PDAs, finden Webanwendungen inzwischen eine Verbreitung auch jenseits der klassischen Anwendungsgebiete.

Ein wesentlicher weiterer Vorteil dieser Technologie gegenüber herkömmlichen Client-Server-Architekturen besteht darin, dass Änderungen an der Geschäftslogik der Anwendung nur an einer „zentralen“ Stelle, nämlich auf dem Webserver, erfolgen. Dies wirkt sich günstig auf die Wartungskosten aus, aber vor allen Dingen sind alle Benutzer sofort auf dem aktuellen Stand.

5.2.3. XML Publishing

XML spielt bei der serverseitigen Implementierung von Web-Applikationen eine immer größere Rolle, z. B. bei der Konfiguration, der Datenrepräsentation und dem Datenaustausch. XML-Technologien, wie XML und XSLT, können in einer Web-Applikation auch für Kernfunktionalitäten, wie der Präsentation von Web-Dokumenten, eingesetzt werden [Wöhr 2004].

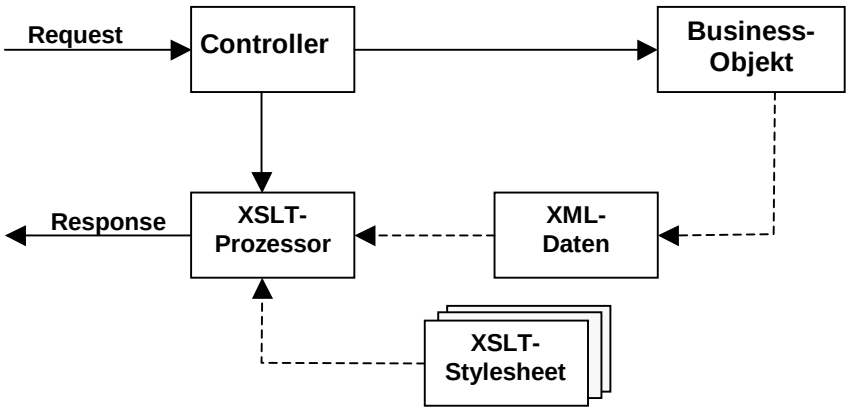

Abbildung 29: Architektur für den XML/XSLT-Einsatz in der Präsentationsschicht

Für das KORA-Dokumentenmanagementsystem wird in der Präsentationsschicht XML-Publishing eingesetzt, um aus der XML-basierten Repräsentation des Wissensnetzes eine vom Client darstellbare Visualisierung zu generieren. Die Transformation erfolgt durch ein XSLT-Stylesheet.

Dieser Ansatz unterstützt eine vollständige Trennung von Inhalt und Präsentation. Die aus der Business-Schicht gelieferten Daten repräsentieren den Inhalt und das XSLT-Stylesheet legt die Präsentation des Inhalts fest. Vor allem aber erfüllt XML-Publishing steigende Anforderungen an die Präsentation. Hierzu zählt unter anderem, dass parallel verschiedenartige Clients bedient werden können.

Ein Client muss nicht zwingend ein Web-Browser sein. Die Anfrage eines Handys wird mit einer Transformation der Daten nach WML beantwortet [Wöhr 2004]. Für jede weitere Client-Gattung werden lediglich entsprechende Stylesheets hinzugefügt. Bei Bedarf kann das Ergebnis einer Anfrage auch als PDF-Dokument an den Client geliefert werden.

5.3. Marktanalyse für Komponenten

Die Liste der verschiedenen Technologien zur Entwicklung von Web-Anwendungen ist mittlerweile lang. Sie reicht von A wie Ajax und ASP über JSP, PHP, Servlets, Web Services bis X wie XML und XSLT. Eine ausführliche Beleuchtung aller Technologien würde den Rahmen dieser Arbeit bei weitem sprengen. Nachfolgend wird daher nur eine Einführung in einige der wichtigsten, für dieses Projekt relevanten, Themen gegeben.

5.3.1. Web-Frameworks

Java Servlets und Java Server Pages (JSP) sind wohl die am weitesten verbreiteten Technologien in der Web-Entwicklung. Java Servlets bieten vielseitige Möglichkeiten zur Implementierung der Anwendungslogik. Bei JSP liegt der Fokus auf der Präsentation einer Webseite. Dynamische Inhalte und Anwendungslogik können mittels in das Online-Dokument eingebetteter Code-Fragmente ausgeführt werden. Mit diesem Ansatz werden komplexe Applikationen jedoch schnell unübersichtlich und können somit schlecht erweitert und gewartet werden. Abhilfe schaffen hier sogenannte, auf dem *Model-View-Control-Paradigma* basierende, Web-Frameworks, wodurch eine Trennung von Inhalt, Logik und Layout erreicht werden kann.

5.3.1.1. Struts

Jakata Struts ist ein Open Source Framework der Apache Software Foundation zur Erstellung von Java Web-Applikationen. Struts zählt zu den klassischen Vertretern der MVC-Architektur. Struts erlaubt die Aufteilung der Applikation in eine Reihe verschiedenartiger Komponenten. Das Zusammenwirken dieser Komponenten wird mit Hilfe von XML konfiguriert. Der Schwerpunkt liegt dabei auf dem Zusammenspiel von *Controller und View.*

Zur Realisierung des *Controllers* werden in Jakata Struts Servlets eingesetzt. Die Präsentation der Informationen (*View*) wird durch JSPs realisiert und für die Kapselung der Daten, als *Model* bezeichnet, dienen JavaBeans.

Das Jakata Struts Framework besteht aus einer Reihe von Komponenten (Klassen, Interfaces und Tag-Bibliotheken), die den Entwickler einer Web-Anwendung bei der Bewältigung von Standardaufgaben unterstützen. Abbildung 30 zeigt das allgemeine Architekturschema „Struts“.

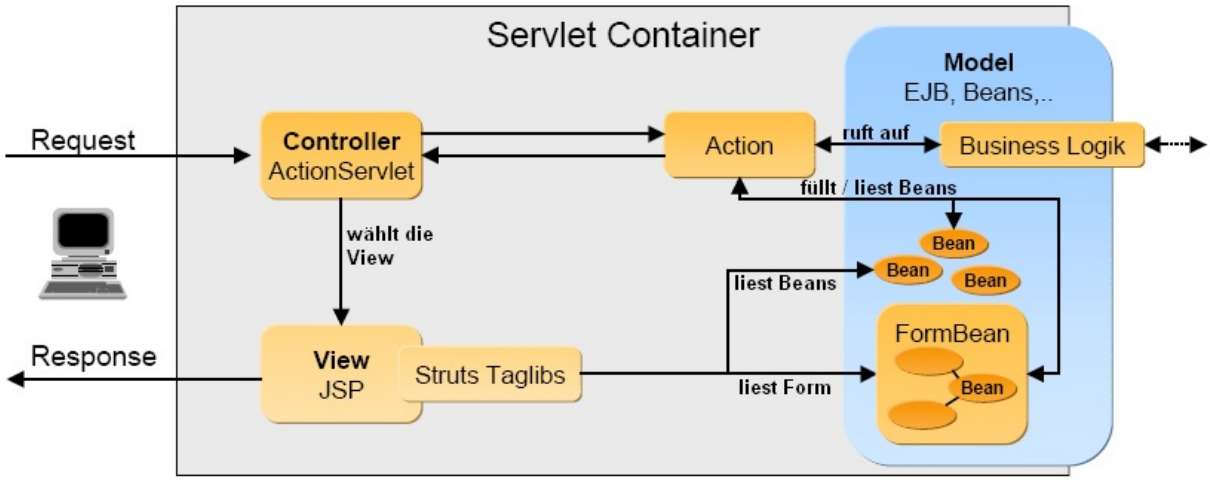

Abbildung 30: Struts – allgemein [30]

[30] Die Abbildung entstammt einer Präsentation „Vergleich der Model 2 Umsetzung von Struts und Cocoon“ von Tobias Kieninger, Orientation in Objects GmbH, www.oio.de

5.3.1.2. Cocoon

Cocoon[31] ist ein komponentenbasiertes XML-Publishing-Framework, das im Rahmen des XML-Projekts der Apache Software Foundation entwickelt wird. Cocoon ist Open-Source und somit frei erhältlich und beliebig erweiterbar. Cocoon wird eingesetzt, wenn Webseiten dynamisch generiert und publiziert werden sollen. Hierzu wird aus einem oder mehreren XML-Dokument(en) und dem zugehörigen XSL-Dokumente(en) ein beliebiges Ausgabeformat generiert und an beliebige Endgeräte ausgeliefert. Cocoon übernimmt also die Rolle des „Übersetzers“ in die jeweilige „Sprache“ des Empfängers. Mögliche Ausgabeformate sind unter anderem: HTML, WML, PDF, RTF oder SVG.

Obwohl Cocoon auch in lokalen Applikationen eingesetzt werden kann, wird es in den meisten Fällen als dynamischer Webside-Generator verwendet [Nie 2004]. In diesem Fall bildet ein reguläres Servlet die Schnittstelle zwischen Cocoon und dem Web. Cocoon wird dann in einem sogenannten Servlet-Container, wie z. B. Tomcat, betrieben.

Cocoon verkörpert das Filter-Entwurfsmuster zur Generierung statischer Dokumente. Jeder Filter konsumiert und produziert XML-Daten über definierte Schnittstellen. Mehrere Filter-Komponenten werden zu einer Pipeline verknüpft, entlang derer ein XML-Datenstrom verarbeitet wird. Die in der Pipeline genutzten Filter lassen sich in die Kategorien: Generatoren, Transformatoren und Serializer einteilen.

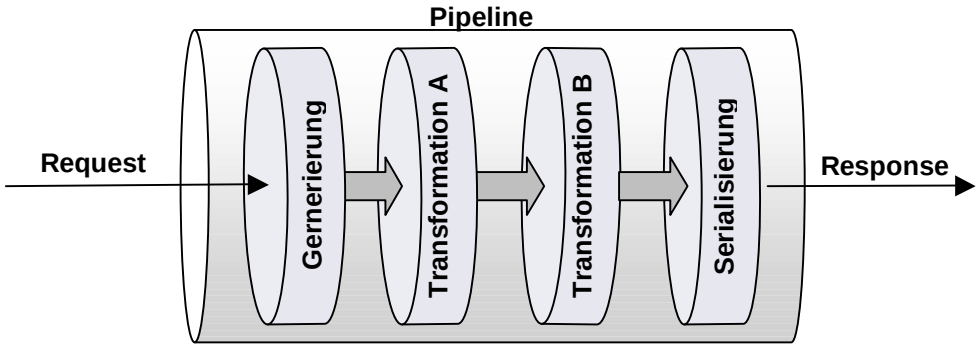

Abbildung 31: Cocoon-Pipeline mit einem Generator, zwei Transformatoren und einem Serialisierer

Der große Vorteil der Pipeline gegenüber anderen Mechanismen der Verarbeitung von XML-Dokumenten besteht in der Tatsache, dass die Definition der Schritte, die ein XML-Dokument zu durchlaufen hat, völlig unabhängig von den Dokumenten selbst und den verarbeitenden Komponenten angegeben werden kann [Nie 2007].

Die einzelnen Schritte des in Abbildung 31 dargestellten Verarbeitungsprozesses, die eine Pipeline durchläuft, werden nachfolgend erläutert:

[31] Cocoon wurde 1998 von dem italienischen Studenten Stefano Mazzocchi entwickelt, während er den Science-Fiction-Film „Cocoon“ sah, wonach das System benannt wurde.

Generierung (Erzeugung des SAX-Streams)
Nachdem eine eingehende Anfrage einer bestimmten Pipeline zugeteilt wurde (Matching), erfolgt im ersten Schritt die Erzeugung eines SAX-Streams aus einer angegebenen Datenquelle. Die hierfür zuständige Komponente ist der Generator.

Transformation (Umwandlung durch SAX-Events)
In diesem Schritt wird die Struktur eines XML-Dokuments, häufig unter Anwendung eines XSLT-Stylesheets, in eine andere Struktur transformiert. Die hierfür zuständige Komponente ist der Transformer. Innerhalb einer Pipeline können mehrere aufeinanderfolgende Transformationen durchgeführt werden, bis das gewünschte Ergebnis erreicht wurde.

Serialisierung (Ausgabe)
Im letzten Schritt einer Pipeline wird das bis zu diesem Punkt verarbeitete XML-Dokument in das Ausgabeformat verwandelt und ausgegeben. Ein solches Format kann neben jedem XML-basierten Format z. B. auch HTML, PDF, PostScript oder JPG sein. Die hierfür zuständige Komponente ist der Serializer.

Zusammenfassung
Cocoon verkörpert eine sehr flexible Präsentationsschicht und ist besonders geeignet für die Umwandlung in unterschiedliche Client-Formate. Die Stärken von Cocoon liegen in der Verarbeitung von XML-Dokumenten zur Präsentation von Inhalten. Cocoon ist kein Framework zur Implementierung komplexer Anwendungslogik. Hierfür können andere Konzepte, wie EJB, eingebunden werden.

Abbildung 32 zeigt das allgemeine Architekturmuster „Cocoon Flow“ im Falle der Integration externer Model-Komponenten.

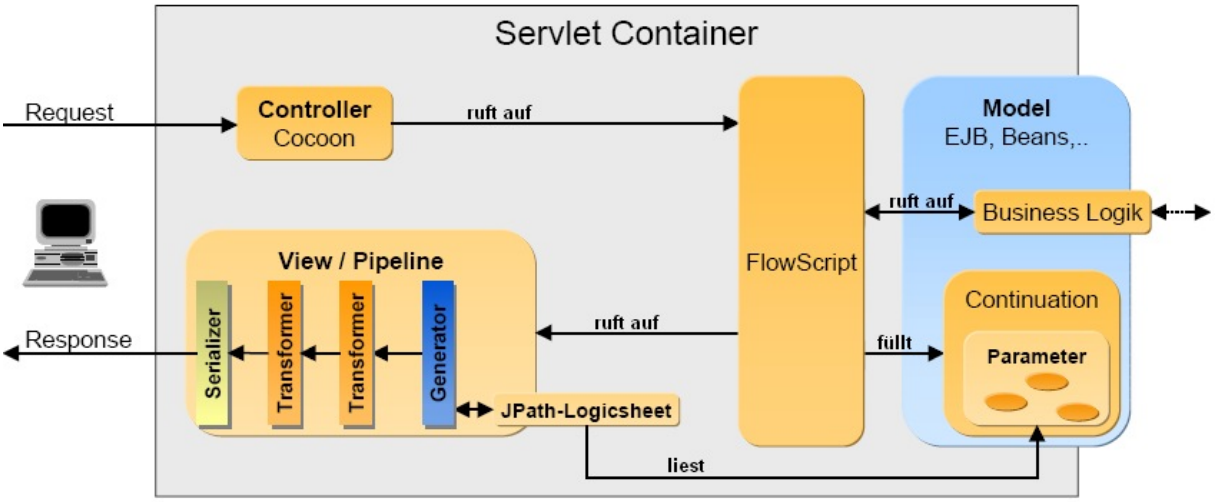

Abbildung 32: Cocoon Flow – allgemein [32]

[32] Die Abbildung entstammt einer Präsentation „Vergleich der Model 2 Umsetzung von Struts und Cocoon“ von Tobias Kieninger, Orientation in Objects GmbH, www.oio.de

5.3.1.3. Zusammenfassung

Jakarta Struts ist ein häufig eingesetztes und ausgereiftes Standard-Framework zur Entwicklung von Java Web-Anwendungen. Struts basiert auf weit verbreiteten Standardtechnologien, wie Java Servlets, JavaServer Pages und Java Beans. Es unterstützt den Model-View-Controller Ansatz und ermöglicht so die klare Trennung von Inhalt, Layout und Logik. Struts eignet sich besonders für Anwendungen mit einem hohen Anteil von Anwendungslogik, wie es häufig z. B. bei E-Business Anwendungen der Fall ist. Für kleinere Web-Applikationen ist der Einsatz von Struts mit relativ viel initialem Aufwand verbunden, es müssen sehr viele Komponenten in Java implementiert werden.

Bei Cocoon steht ebenfalls die strikte Trennung von Inhalt, Layout und Logik im Mittelpunkt. Erreicht wird dieses jedoch auf eine andere Art. Die Programmierung von Web-Applikationen erfolgt bei Cocoon hauptsächlich durch die Verwendung von XML und XSLT, was bei Anwendungen mit geringem Logikanteil den Aufwand verringert. Eine weitere Stärke von Cocoon besteht in der Fähigkeit, Inhalt ohne großen zusätzlichen Programmieraufwand in verschiedensten Formaten darzustellen (multichannel publishing). Dies könnte bei dem vorliegenden Anwendungsfall mit vielen verschiedenen Endgeräten eine wichtige Rolle spielen.

Ausschlaggebend für die Entscheidung, Cocoon als Framework für die Entwicklung der geplanten Webanwendung einzusetzen waren:

- die Anwendungsdaten liegen mit XML Topic Maps bereits in XML vor
- gegebenenfalls müssen unterschiedliche Ausgabeformate generiert werden
- es ist wenig Anwendungslogik zu integrieren

5.3.2. Topic Map APIs

XML Topic Maps sind XML Dateien. Um sie mit einer Programmiersprache wie Java bearbeiten zu können, benötigt man Software, die diese Daten ausliest und als Topic Map Datenstruktur bereitstellt. Eine solche Anwendung nennt man Topic Map-Engine. Die Funktionen, mit denen eine Topic Map bearbeitet werden kann, sind in der zugehörigen Topic Map-API definiert.

5.3.2.1. TMAPI

Mit TMAPI existiert ein API-Standard für die Verarbeitung von Topic Maps. Diese Schnittstellendefinition wird von verschiedenen Topic Map-Engines als gemeinsames Topic Map-Interface implementiert. Entwicklern von Topic Map-Anwendungen steht somit eine einzige einheitliche Schnittstelle zur Verfügung, die unabhängig ist von der konkreten Implementierung der verwendeten Engine. Anwendungen werden somit unabhängig von der eingesetzten Topic Map-Engine und die Wiederverwendbarkeit des Programmcodes wird unterstützt. Engines, die TMAPI unterstützen sind z.B. TM4J, tinyTIM und XTM4XMLDB.

5.3.2.2. tinyTIM

tinyTIM ist eine eher kleine frei verfügbare Topic Map-Engine mit nur wenig zusätzlichen Funktionen. Neben den in TMAPI definierten Schnittstellen bietet tinyTIM noch Dienste wie z. B. XTM Parser und XTM Serializer zum Lesen und Schreiben von XTM 1.0 Dateien, Konvertierung von RDF nach Topic Map und eine IndexAPI mit der ein Index über alle Occurrences, TopicNames, Topics, Associations usw. erstellt werden kann, der einen schnelleren Zugriff ermöglicht, als es durch die Iteration über alle Elemente möglich ist. Komplexere Operationen, wie zum Beispiel das Extrahieren eines Teils einer Topic Map sind nicht vorhanden.

5.3.2.3. TM4J

TM4J ist ebenfalls ein open-source Produkt und beinhaltet ein umfangreiches Funktionspaket zur Verarbeitung von Topic Maps. Neben einer eigenen Schnittstellendefinition implementiert TM4J auch das TMAPI. Den Kern von TM4J bilden Funktionen zum parsen, manipulierten und speichern von Topic Maps. TM4J ermöglicht auch das Speichern einer Topic Map in einer Datenbank und dafür kann entweder die objektrelationale Datenbank Ozone[33] oder eine relationale Datenbank über Hibernate[34] eingesetzt werden. Ferner bietet TM4J Query-Funktionen unter Benutzung der Abfragesprache Tolog[35].

5.3.2.4. Zusammenfassung

Für die Implementierung des KORA-Dokumentenmanagementsystems wird die API von TM4J genutzt, da sie über einen größeren Funktionsumfang verfügt als TMAPI oder tinyTIM. Insbesondere die Option, die Topic Map zu einem späteren Zeitpunkt statt in einer XML-Datei in einer Datenbank zu speichern, gaben hierfür den Ausschlag. Auch wird hierdurch die Möglichkeit offen gehalten, in einer späteren Version Tolog als Abfragesprache zu nutzen. Der höhere Ressourcenverbrauch zur Laufzeit dürfte hier keine Rolle spielen, da TM4J nur auf dem Server eingesetzt wird. Sollten für die offline-Nutzung des Netztools später Client-Anwendungen entwickelt werden, kann hierfür ohne weiteres ein schmaleres API eingesetzt werden.

---

33 Ozone ist ein Java basiertes objektrelationales Datenbankmanagementsystem (http://www.ozone-db.org)

34 Hibernate ist eine Software-Komponente für die Abbildung von Java-Objekten in ein relationales SQL-Schema (http://www.hibernate.org)

35 Tolog ist eine von Ontopia entwickelte Abfragesprache für Topic Maps (http://www.ontopia.net/topicmaps/materials/tolog.html)

6. Prototyp (proof of concept)

Der Lösungsansatz wird auf Basis aktueller Web-Technologien realisiert. Ausschlaggebend für die Technologieentscheidung war, dass hierdurch die Versorgung der vielen verschiedenen mobilen Geräte mit einer Clientsoftware entfällt, da internetfähige Geräte in der Regel bereits standardmäßig über einen Browser verfügen. Durch die Verwendung des Browsers als Benutzungsschnittstelle vereinfacht sich auch der Zugriff auf den Dokumentenbestand aus dem Inter- und Extranet.

6.1. Ziele des Prototypen

Parallel zur Analyse des Anwendungsbereichs wurde mit der Implementierung eines Software-Prototypen begonnen. Basierend auf der in Kapitel 3 beschriebenen Systemversion soll im Rahmen einer prototypischen Implementierung die Funktionsweise einiger ausgewählter Eigenschaften gezeigt und getestet werden. Prototyping ist allgemein zu einem festen Bestandteil im Softwareentwicklungsprozess geworden. Diese Vorgehensweise hat den Nutzen, dass sie schnell zu ersten Ergebnissen führt. Unter Einbeziehung der späteren Anwender ermöglicht Prototyping ein frühzeitiges Feedback bezüglich der Eignung des Lösungsansatzes.

In der Anfangsphase trägt der Demonstrationsprototyp dazu bei, Aspekte der Benutzungsschnittstellen oder Teile der Funktionalität zu veranschaulichen. Weiterhin kann ein Prototyp dazu dienen, den Anwendungsbereich zu analysieren. Sollen bei der Umsetzung einer Softwarelösung neue bisher unbekannte Technologien eingesetzt werden, wird ein Prototyp verwendet, um im Vorwege die technische Umsetzbarkeit zu testen.

In der Praxis existieren, je nach Zweck und Entwicklungsphase des Projekts, verschiedene Arten von Prototypen [Müll 2007]:

1. Demonstrationsprototyp (erster Eindruck)
2. Prototyp im engeren Sinne (erste Funktionalität)
3. Labormuster (technische Umsetzbarkeit)
4. Pilotsystem (Kern des Systems)

Ein anderer Klassifikationsansatz, der so genannte 3E-Ansatz nach Christiane Floyd, unterscheidet folgende drei Prototypingverfahren [Rsc 2006] [WiSoWiWiKi]:

explorativ: Beim explorativen Prototyping dient der Prototyp als Kommunikationsmedium zwischen Entwickler und Endbenutzer. Die Ermittlung von Anforderungen an ein Softwaresystem kann beispielsweise durch den Einsatz von Benutzungsschnittstellen-Prototypen erleichtert werden.

experimentell: Das experimentelle Prototyping hat den Zweck, die Tauglichkeit der bisherigen Problemlösung unter Beweis zu stellen, bevor in eine ressourcenaufwendige Implementierung investiert wird. In diesem Fall entspricht der Prototyp dem oben genannten Labormuster zum Testen der Umsetzbarkeit.

evolutionär: Evolutionäres Prototyping ist die schrittweise Weiterentwicklung des Prototypen. Auf diese Weise entsteht im Laufe des Projekts ein funktionsfähiges Pilotsystem.

Außerdem lassen sich Prototypen hinsichtlich ihrer Dimension in horizontale und vertikale Typen unterscheiden.

Horizontaler Prototyp
Als horizontaler Prototyp werden Prototypen bezeichnet, die nur einzelne Schichten des Systems realisieren, beispielsweise die Benutzungsoberfläche. Diese wird dann aber möglichst vollständig abgebildet.

Vertikaler Prototyp
Ein vertikaler Prototyp implementiert ausgewählte Teile des Zielsystems vollständig durch alle Ebenen hindurch.

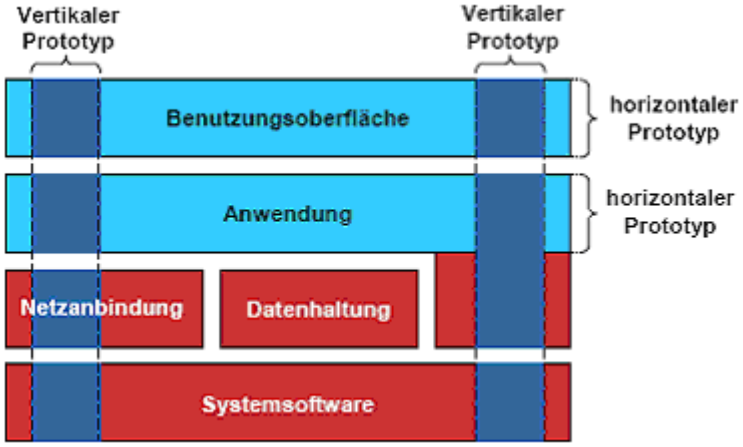

Abbildung 33: Vertikales und horizontales Prototypingmodell [Lig 2006]

In der Regel wird ein Prototyp nie einzig und allein einer der vorgenannten Kategorien zugeordnet werden können. Meist wird es sich hierbei um eine „Mischform“ handeln, die Merkmale mehrerer Kategorien berücksichtigt. Im Rahmen dieser Arbeit wird ein Prototyp implementiert, der ebenfalls mehreren der genannten Zwecke dient.

In dieser Arbeit soll mit Hilfe des Prototypen soll untersucht werden, ob sich die diskutierten Ansätze bezüglich eines Dokumentenmanagements auf die geplante Weise realisieren lassen und ob die gewählte Technologie, Web-Applikation auf Basis des XML-Publishing-Framework „Apache Cocoon“, hierfür geeignet ist. Hier wird bereits deutlich, dass der entwickelte Prototyp sowohl explorativen als auch experimentellen Zwecken dient.

Weiterhin sollen Visualisierungsmöglichkeiten des Netzes aufgezeigt werden. Insgesamt soll die Basis für die Entwicklung eines produktiven Systems gelegt werden.

6.2. Spezifikation

Die Systemspezifikation bildet die Grundlage der Konstruktion und Realisierung des geplanten Softwaresystems. In diesem Abschnitt erfolgt eine formale Beschreibung der benötigten Eigenschaften und Funktionen. Die Systemspezifikation dient dazu, die Systemgrenzen festzulegen und beschreibt das zu bauende Softwaresystem aus fachlicher Sicht. Sie gibt dabei Antworten auf folgende Fragen [Höf 2003]:

1. Welche Informationen über Dokumente verwaltet das System?
2. Welche Workflows bietet es?
3. Wie sieht die Benutzungsschnittstelle aus?
   - Dialoge und Dialogabläufe
   - Druckausgaben
4. Wie arbeitet es mit Nachbarsystemen zusammen?

6.2.1. Interne Repräsentation der Dokumente

Die interne Repräsentation der Dokumente erfolgt losgelöst von den eigentlichen Dokumentdateien innerhalb der Topic Map in Form einer XML Struktur. Aus den in der Topic Map-Datei gespeicherten Informationen, kann eine graphische Darstellung als Netz mit Knoten und Kanten abgeleitet werden. Die Metabeschreibungen und Verweise zu den Dokumenten finden sich ebenfalls innerhalb der Topic Map.

Die Dokumentdateien werden unverändert und unabhängig von der Repräsentation auf einem Fileserver des Dokumentenmanagementsystems abgelegt. Benutzerdaten, Gruppenzugehörigkeit und Benutzerrechte werden verschlüsselt und geschützt in einer Datenbank oder einem Verzeichnisdienst verwaltet.

6.2.2. Workflows

Die Kernfunktionalitäten können grob in die Prozesse Daten- bzw. Informationsgenerierung und Informationswiedergewinnung unterteilt werden. Zur Informationsgenerierung zählen:

Dokumentimport
Der Dokumentimport umfasst folgende Aktionen

- upload des Dokuments vom Endgerät des Nutzers auf den Fileserver des KORA-Systems
- extrahieren von Metadaten aus der Datei (soweit Metadaten in der Datei verfügbar sind)
- formularbasierte Anzeige der extrahierten Metainformationen zur weiteren Bearbeitung durch den Nutzer
- Erzeugung eines repräsentierenden Knotens im Wissensnetz

Infolge der Bearbeitung eines Dokuments, das sich bereits im System befindet, ändern sich gegebenenfalls auch die Metadaten. In dem Fall muss eine Anpassung des repräsentierenden Knotens erfolgen.

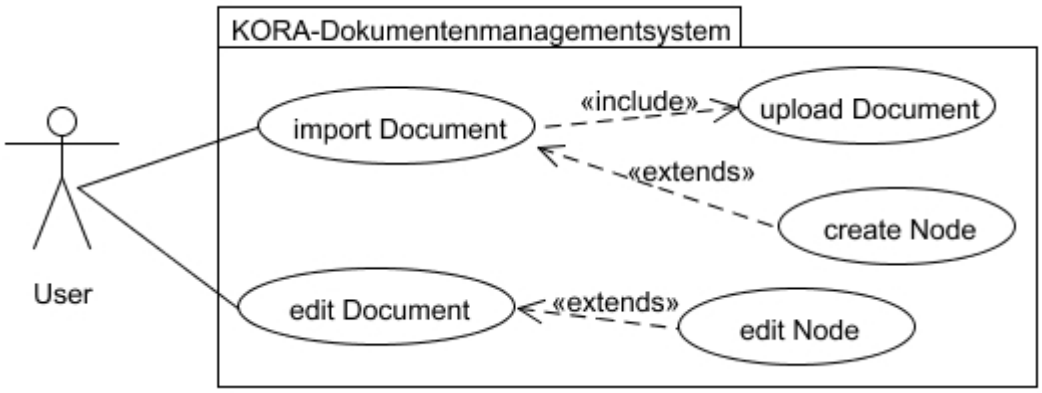

Abbildung 34: Dokument importieren und bearbeiten

Abbildung 34 zeigt die oben beschriebenen Anwendungsfälle des Dokumentimports und des Editierens eines Dokuments.

Dokumentexport
Der Nutzer wählt unter Verwendung eines entsprechenden Dialogs die gewünschte Datei aus und stößt die Downloadprozedur an. Diese bedingt folgende Aktionen:

- Damit dem Nutzer die im Netz gespeicherten Metainformationen zum Dokument auch offline zur Verfügung stehen, wird hieraus eine Metadatendatei generiert. Die Datei ist ein separates XML-Dokument (Sidecar-Dokument) mit den entsprechenden Informationen und wird parallel zum eigentlichen Dokument ausgeliefert. Löscht der Nutzer diese Datei auf seinem Gerät, verzichtet er explizit auf die Metainformationen.
- Download des Dokuments vom Fileserver des KORA-Systems auf das Endgerät des Nutzers.
- Download der Sidecar-Datei.

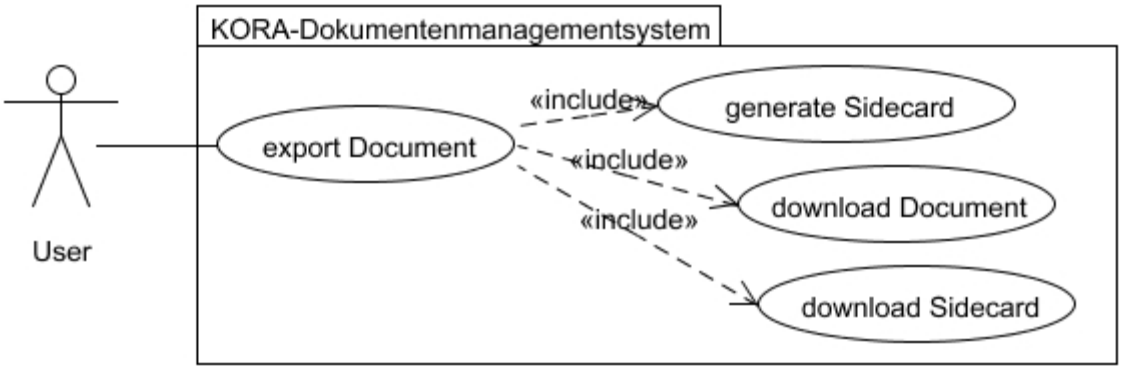

Abbildung 35: Dokument exportieren

Abbildung 35 zeigt den Anwendungsfall des Dokumentenexports und die damit verbundenen Aktionen. Diese Vorgehensweise führt zu einer nichtdestruktiven Dokumentenverarbeitung. Das Dokument wird nicht verändert, sondern nur durch eine Datei mit Metadaten ergänzt. Diese Vorgehensweise ist beispielsweise auch in Produkten wie Adobe Photoshop™ und Adobe Camera Raw™ umgesetzt.

Metadatenerfassung
Jedes Dokument respektive jeder Knoten im Netz soll mit zusätzlichen Metainformationen näher beschrieben werden. Das Schema für die Knotentypen und Metadaten kann sowohl in der Topic Map selbst oder in einer externen Topic Map beschrieben werden. Dieses Schema kann jederzeit an veränderte Anforderungen angepasst werden. Bei Änderung des Attributsatzes für einen Knotentyp behalten bestehende Knoten ihre Gültigkeit. Gegebenenfalls müssen hier Metadaten nachgepflegt werden.

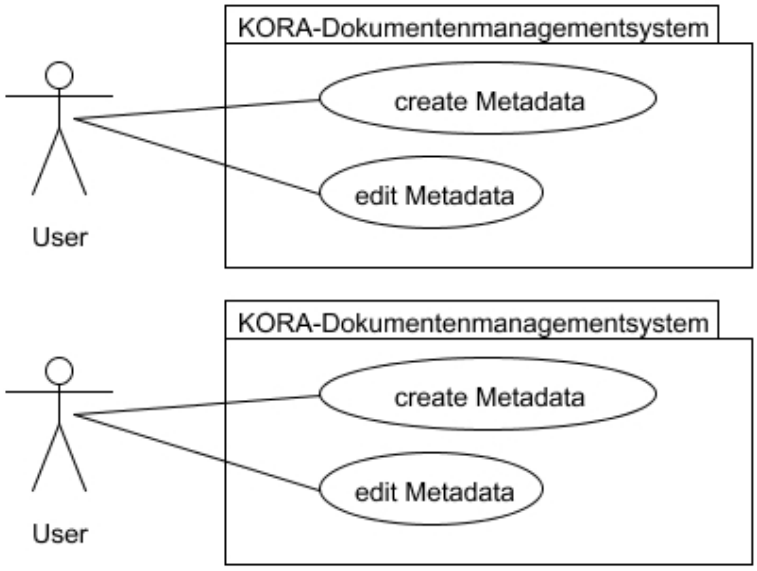

Abbildung 36: Metadaten erfassen und ändern

Abbildung 36: Metadaten erfassen und ändern zeigt die möglichen Anwendungsfälle zum Erfassen und Editieren von Metadaten.

Erstellung von Assoziationen
Die Erzeugung von Beziehungen zwischen den Dokumentknoten erfolgt manuell durch den Benutzer. Hierzu wird ein Dialogfenster geöffnet und die zu verbindenden Knoten und der Beziehungstyp bestimmt.

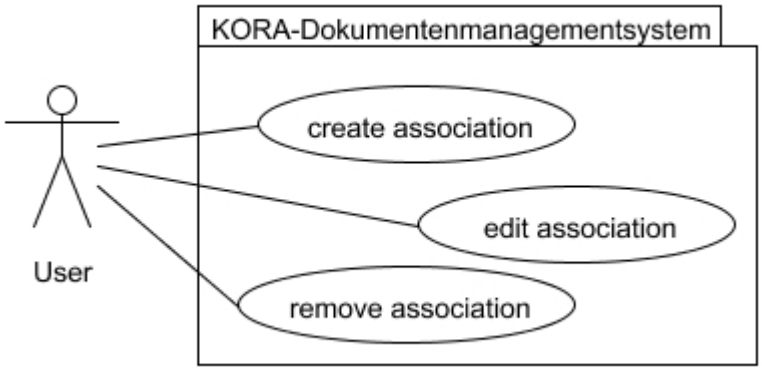

Abbildung 37: Verknüpfung erstellen und bearbeiten

Navigation im Netz
Die Navigationskomponente reagiert interaktiv auf Mouseclicks des Nutzers. Wählt der Nutzer per Mouseclick einen neuen Knoten aus, wird ein aktueller Ausschnitt des Netzes in den Browser geladen, bei diesem wird der ausgewählte Knoten im Zentrum der Visualisierung angezeigt (Abbildung 39). Metadaten des fokussierten Knoten werden seitlich in einem Frame angezeigt. Referenzierte Dokumente können von hier durch anklicken geöffnet werden.

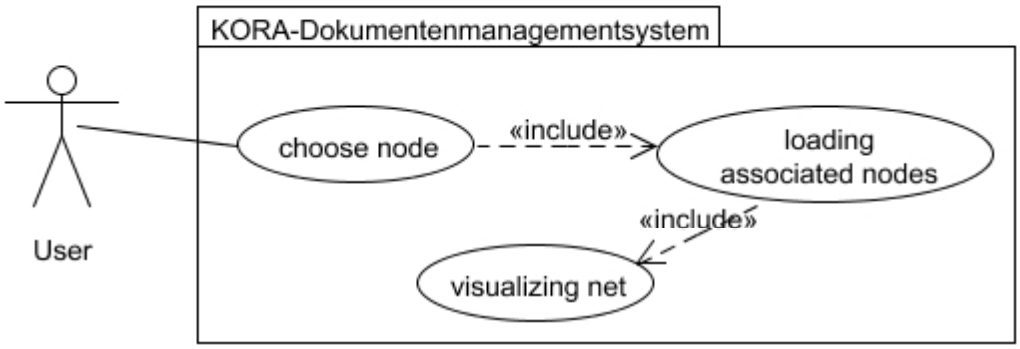

Abbildung 38: Navigation im Netz

6.2.3. Benutzungsschnittstelle

Die Benutzungsoberfläche besteht aus einem framebasierten Hauptfenster und einigen Formularen zum Anlegen und Editieren von Knoten und Assoziationen sowie Formularen für den Im- und Export von Dokumenten.

Hauptfenster
Das Hauptfenster besteht aus einem oberen Frame, der Menüleiste, einem Frame am linken Rand und dem Hauptframe. In den Untermenüs der Menüleiste verbergen sich Aktionen wie Dokumentimport und –export, Öffnen eines Formulars zum Erstellen oder Bearbeiten der Knoten und Assoziationen. Der Frame an der linken Seite teilt sich nochmals in drei Unterframes. Der obere dient der textbasierten Suche. Der mittlere ermöglicht eine Einschränkung der Knoten nach ihrem Typ, z. B. alle Assoziationstypen. Der untere Frame gleicht einem Propertyframe, hier werden die Metadaten und Ressourcen des fokussierten Knoten tabellarisch in zwei Spalten als Attribut-Wert- Paar aufgelistet.

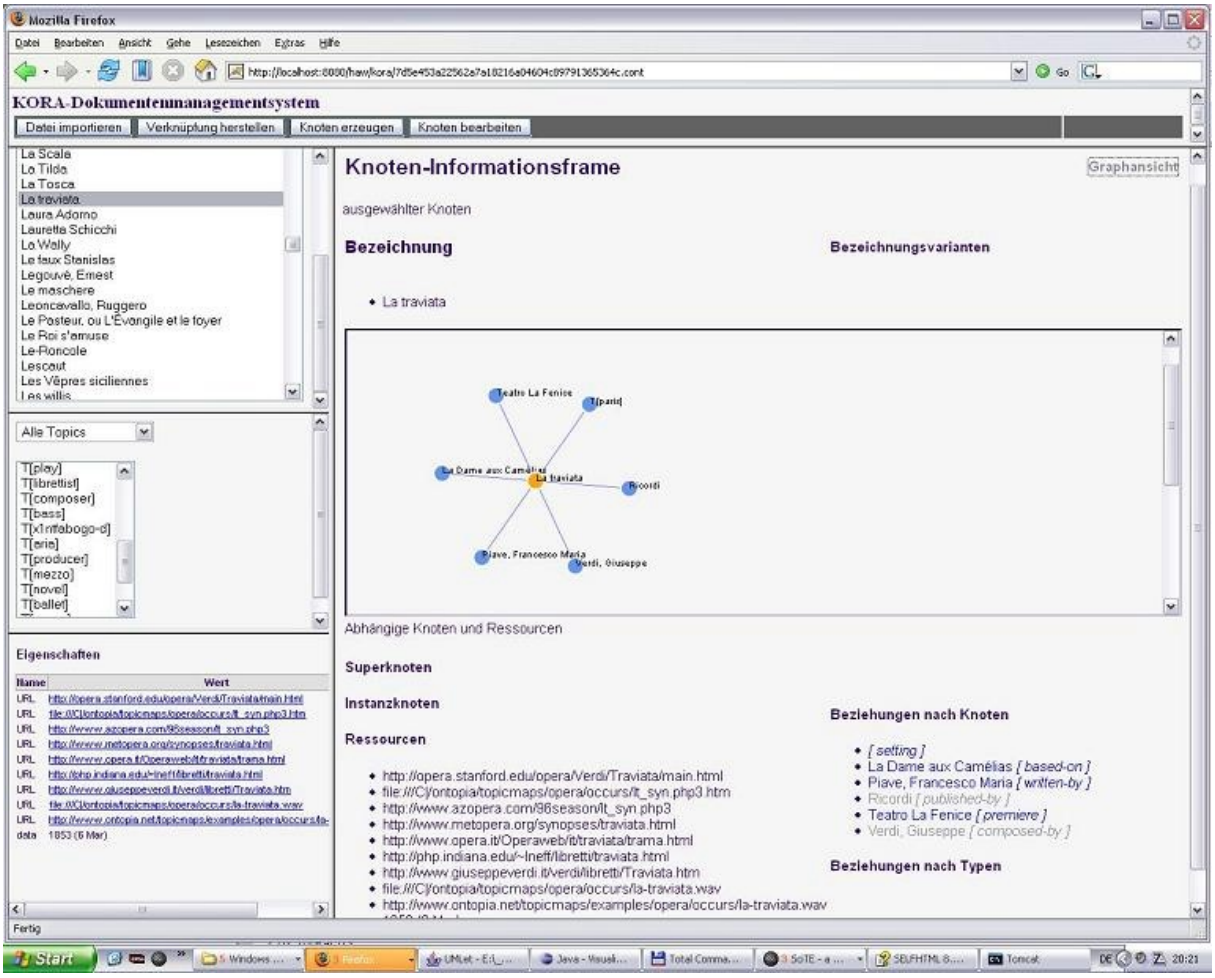

Abbildung 39: Benutzungsoberfläche des KORA-Dokumentenmanagmentsystem

Die eigentliche Hauptkomponente ist jedoch der Netz-Visualisierungsframe. In der Mitte dieses Frame befindet sich das SVG-Dokument, das die Visualisierung des aktuell ausgewählten Netzausschnitts darstellt. Auf Wunsch kann diese Sicht auch ausgeblendet und bei Bedarf wieder eingeblendet werden. Darüber und darunter sind die Beziehungen und Nachbarknoten alternativ in Form von Listen zu finden.

Sowohl die textliche Darstellung des Netzes als auch die graphische Version verfügen über Interaktionsfähigkeit in der Form, dass durch Anklicken ein neuer Knoten geladen wird und alle Datenframes aktualisiert werden.

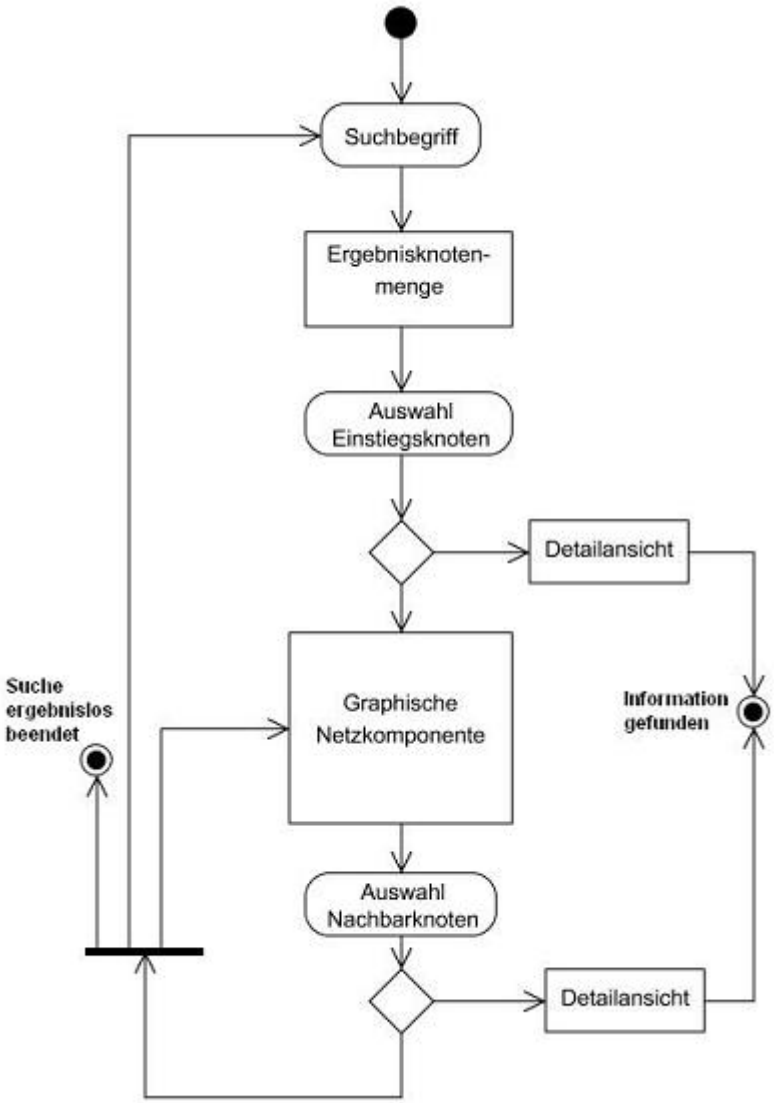

Abbildung 40: Informationssuche

Das obige Diagramm zeigt den Ablauf des Standardanwendungsfalls „Informationssuche“.

6.3. Realisierung

Bei der Realisierung des Prototypen wurde auf Java als Programmiersprache gesetzt. Java ist plattformunabhängig, aktuell und es existieren zahlreiche frei verfügbare Standardkomponenten und APIs. Des Weiteren wurde Tomcat als Web-Server und Cocoon für die Transformation von XML-Daten eingesetzt. Für die Implementierung von Fachklassen wurde das JDK 1.5 verwendet sowie die Topic Map-Enginge „TM4J“.

Für die Transformation der Netzausschnitte aus der Topic Map in ein HTML-Dokument wurden für jeden Frame entsprechende XSLT-Stylesheets entworfen. Die Transformation erfolgt in der Regel in mehreren Schritten. Im ersten Schritt werden die Daten aus der Topic Map selektiert und dann im zweiten Schritt in ein HTML- oder SVG-Dokument eingebettet.

6.4. Erfahrungen aus dem Prototypen

Der Prototyp ist weitgehend fertig gestellt. Einige Dinge stehen jedoch noch aus. Neben der Implementierung ausgesparter Funktionalitäten, zählt hierzu die Durchführung von Benutzungstests. Entsprechende Tests sollten in zwei Phasen durchgeführt werden.

1. Die Benutzbarkeit der Oberfläche sowie die Effektivität der Systemfunktionen kann in Einzeltests in einem Usabilitylabor beobachtet und nachgewiesen werden. In diesen Tests werden die Trabanten bei der Lösung von Einzelaufgaben beobachtet und im Anschluss hierzu befragt.
2. In einer zweiten Testphase ist die Tauglichkeit des Prototypen hinsichtlich der Gruppenarbeit im Konferenzraum zu untersuchen. Hierbei wird eine Gruppe von Testpersonen an den großen Wandbildschirmen arbeiten und Dokumentenmanagement betreiben. Im einzelnen werden nun die Funktionen zur Evaluierung des Wissensnetzes und die Informationssuche im Netz gemeinschaftlich durchgeführt. Hierzu zählt ferner der Im- und Export von Dokumenten von mobilen Geräten.

Diese Tests stehen zwar noch aus und die Benutzbarkeit des Systems muss noch nachgewiesen werden, dennoch kann man sagen, dass der Prototyp die generelle Machbarkeit des Ansatzes bereits bestätigt.

7. Methodische Abstraktion

Der Ansatz, Wissen mit Topic Maps in Form eines semantischen Netzes zu strukturieren und Dokumente mit Metadaten zu beschreiben, kann durchaus auch auf andere Szenarien übertragen und nutzbringend eingesetzt werden. Im konkret betrachteten Anwendungsfall wurde auf Basis von Topic Maps ein Dokumentenmanagement für den Collaborative Workplace entworfen. Ein solches System könnte praktisch in jeder Organisation zur abteilungsübergreifenden Verwaltung gemeinsam genutzter Informationen und Dokumente eingesetzt werden. Andere Beispiele für den Einsatz von Topic Maps zur Anreicherung von Inhalten mit semantischen Werten liefern E-Learning-Systeme oder Suchmaschinen.

Die direkte Verlinkung inhaltlich zusammenhängender Dokumente nach dem Hyperlink-Prinzip hat für die Navigation in großen Datenbeständen wesentliche Vorteile. Im Vergleich zur herkömmlichen Ablage von Dokumenten im Dateisystem, sind benachbarte Informationen so wesentlich schneller zu erreichen als über lange Wege entlang einer starren Verzeichnisstruktur.

Weitere Potentiale in der Verwendung von Topic Maps als Metabeschreibungssprache für Dokumente sind in der einheitlichen und flexiblen Beschreibung der gespeicherten Dokumente mit Metadaten zu sehen. Dokumente können unabhängig vom Typ mit zusätzlichen beschreibenden Informationen versehen werden. Hierdurch kann bereits auf Metaebene die Aussagekraft eines Dokuments bezüglich des Inhalts und gegebenenfalls des Werts für die eigene Arbeit erhöht werden.

Auf diese Weise lassen sich Dokumente unabhängig vom jeweiligen Typ auf einheitliche Weise mit Metadaten beschreiben. Die Menge der zur Beschreibung eines Dokumentes zur Verfügung stehenden Attribute kann frei gewählt und jederzeit ergänzt werden. Dies ist bei der Mehrzahl von Dateitypen von Haus aus nicht möglich. Zum einen ist der Satz an Beschreibungsattributen meist sehr typspezifisch und zum anderen kommt man anwendungsübergreifend an diese Daten gar nicht oder nur sehr schwer heran.

Der Ansatz, Metadaten auf XML-Basis mit Topic Maps zu codieren bedeutet einerseits, die Metadaten von den Dokumenten zu trennen, sie andererseits aber in einem engen Zusammenhang zu halten. Probleme bei dieser Vorgehensweise treten jedoch auf, wenn Dokumente durch Kopieren oder Verschieben dem Einflussbereich der Topic Map entzogen werden, in dem Fall gehen die Metadaten verloren. Um dies zu verhindern, sind gesonderte Maßnahmen nötig, z. B. das Anheften der Metadaten an die Datei oder das Übertragen der Daten in einer separaten Datei, einer so genannten Sidecar-Datei.

8. Zusammenfassung und Ausblick

8.1. Zusammenfassung

Ziel dieser Arbeit war der Entwurf einer Benutzungsschnittstelle für ein Dokumentenmanagementsystem in einem Gruppenarbeitsraum. Die Anwendung dient hier dem Zweck, die Mitglieder einer Projektgruppe bei der gemeinschaftlichen Arbeit zu unterstützen und deren Arbeitsabläufe effizienter zu gestalten. Insbesondere soll das Auffinden vorhandener Informationen im Dokumentenbestand erleichtert und beschleunigt werden.

Hierzu sollte eine Schnittstelle entwickelt werden, welche die Möglichkeit bietet, digitale Dokumente unterschiedlichen Datentyps flexibel mit Metadaten zu beschreiben. Außerdem sollte es möglich sein, inhaltlich zusammenhängende Informationen nach dem Hyperlink-Prinzip beliebig und unabhängig vom Datentyp zu verlinken und entlang dieser Links zu navigieren. Weitere Anforderung war, dass das System von unterschiedlichen überwiegend mobilen Clients nutzbar sein sollte.

Im Anschluss an die Identifizierung der Anforderungen wurden zunächst Lösungsansätze thematisch verwandter Projekte, dem Personalinformationmanagement, betrachtet. Hiervon erfüllte jedoch kein Lösungsansatz die Anforderungen in ausreichender Weise, obwohl auch diese Produkte zum Ziel haben, Informationen zu strukturieren und ein gezielteres und schnelles Auffinden benötigter Informationen zu unterstützen.

Umgesetzt wurden diese Anforderungen als Web-Anwendung, basierend auf Standardtechnologien, Java als Plattform und frei verfügbaren Java-Komponenten. Eine Schlüsselkomponente bilden hierbei die Topic Maps. Topic Map ist ein XML-basierter Standard zur Abbildung semantischer Netze, mit schwacher semantischer Ausdrucksstärke. Die von Topic Maps gebotenen Konzepte sind jedoch ausreichend um einfache Beziehungen, wie sie hier benötigt wurden, auszudrücken.

Die Visualisierung der mit Topic Maps abgebildeten Netzstruktur wurde auf zweierlei Arten realisiert. Zum Einen als textliche Version in Form von Listen. Hyperlinks führen zu den benachbarten Knoten, die dann dynamisch nachgeladen werden und eine neue Sicht auf das Netz bieten. Ergänzend zur textlichen Variante kann auch ein Netzausschnitt als Sterngraph erzeugt werden. Dies wurde durch Transformierung in ein SVG-Dokument realisiert. Diese Sicht erfordert jedoch, dass der Browser SVG darstellen kann. Gegebenenfalls muss ein SVG-Plugin installiert werden.

8.2. Stand der Entwicklung

Besonders aufwendige Teilaufgaben innerhalb dieser Arbeit und der Entwicklung des Prototypen waren:

- Die Suche und Bewertung geeigneter Standards und Technologien zur Realisierung der Vision einer Dokumentenverwaltung im Stil eines Semantischen Netzes.
- Die Implementierung der GUI, einschließlich der graphischen Visualisierung eines Netzausschnitts in Form eines Sterngraphen.

- Die Evaluierung einer Ontologie im Sinne des geplanten Dokumentenmanagements unter Berücksichtigung der Metadatenattribute für gängige Dokumenttypen.

Im Rahmen dieser Arbeit konnten daher nicht alle Anforderungen entsprechend umgesetzt werden. Verschiedene Funktionen mussten ausgeklammert werden. Realisiert wurde eine Web-Benutzungsschnittstelle mit Funktionen zur textbasierten Suche nach Knoten innerhalb der Topic Map. Die Darstellung der assoziierten Informationen zu einem ausgewählten Knoten, wie Eltern- und Instanzknoten sowie die über Assoziationen verbundenen Knoten, ist wahlweise als Sterngraph oder textlich in Listenform verfügbar.

Des Weiteren werden die zum ausgewählten Knoten gehörenden Metainformationen und Ressourcen tabellarisch in Form von Eigenschaft-Wert-Paaren angezeigt. Über die Menüleiste können Formulare zum Erstellen eines neuen Knotens oder Links geöffnet oder ein Dateiupload angestoßen werden.

8.3. Fazit

Im Verlauf dieser Arbeit hat sich gezeigt, dass mit Topic Maps ein flexibles Netzmodellierungswerkzeug erstellt werden kann. Die Anfangsidee, Informationen und Dokumente anwendungsunabhängig unter einer einheitlichen Oberfläche in Form eines Semantischen Netzes zu verwalten, stellt eine interessante Alternative zur herkömmlichen Ablage im Dateisystem dar. Mit der Entwicklung des Prototypen für das KORA-Dokumentenmanagementsystem wurde die Basis geschaffen, beliebige inhaltlich zusammenhängende Dokumente sinnvoll miteinander verlinken zu können.

Mit Hilfe entsprechender Visualisierungskomponenten lassen sich diese Zusammenhänge optisch ansprechend und intuitiv verständlich darstellen. Eine weitere anfängliche Systemidee, den Informationsgehalt eines Dokuments auf Metaebene durch flexible Anreicherung mit Metainformationen zu steigern, konnte ebenfalls mit Hilfe der Topic Maps umgesetzt werden.

Allerdings ist die Erstellung und Pflege eines solchen semantischen Modells sehr aufwendig und teuer und es bleibt abzuwarten, ob etwaige Nutzer ein solches System entsprechend nutzen und vor allem nutzbringend fortführen. Der Nutzen eines solchen Netzes hängt maßgeblich von Qualität und Quantität der Assoziationen ab und die müssen manuell von den Nutzern erstellt werden. Dies kann auch nicht durch das System erzwungen werden.

Weniger zum Tragen käme dieses Problem, wenn die Mehrzahl der Nutzer nur lesend auf den Datenbestand zugreift und Erweiterungen von einer begrenzten Zahl von Redakteuren vorgenommen werden. In diesem Fall kann sicher ein entsprechendes Bewusstsein für den Nutzen und die Notwendigkeit der Pflege vorausgesetzt werden.

8.4. Ausblick

Bei dem implementierten System handelt es sich um einen ersten Prototypen. Dieser besitzt noch keine Produktreife. Der Prototyp diente in erster Linie dazu, die Möglichkeiten von Topic Maps bezüglich der Verwaltung von Dokumenten auszuloten. Des Weiteren sollte anhand des Prototypen getestet werden, ob die Anforderungen bezüglich Benutzbarkeit des Systems mit einer Web-Applikation erreicht werden können.

Im Falle einer Weiterentwicklung des Prototypen zu einem Produktivsystem sollten weitere Aspekte berücksichtigt werden, z.B. Autorisierung und Rechteverwaltung, so dass nutzerspezifische Sichten auf die Daten generiert werden können und ein Nutzer nur die Teile des Datenbestandes sieht, für die er auch leseberechtigt ist. Folgende Funktionen wurden noch nicht realisiert: Metadatenextraktion, Versionskontrolle, Synchronisierung und Im- und Exportkontrollen. Des Weiteren sollten auch vorhandene Funktionen wie Suche und Visualisierung weiter verfeinert werden.

Bezüglich der Wahl, die Benutzungsschnittstelle als Web-Applikation zu realisieren, muss sich bei weiterführenden Tests zeigen, ob diese von den Nutzern als benutzungsfreundlich und flexibel genug eingestuft wird, oder ob ein Desktop-Client hier bessere Ergebnisse erzielen würde. Insbesondere für den Konferenzraum selber wäre die Realisierung eines Clients sinnvoll, da hierdurch zusätzliche Funktionen angeboten werden können, die mit einem Webbrowser als Client nicht zu realisieren sind.

In diesem Fall könnten Werkzeuge integriert werden, die das direkte Bearbeiten einfacher Dokumente unterstützen, wie z. B. ein Texteditor oder auch ein Viewer zum Darstellen von Bildern und Graphiken. Bezüglich der Steuerung der Wandmonitore bestehen weitere Anforderungen, die mit einer Clientanwendung bedeutend flexibler gelöst werden könnten.

Literaturverzeichnis

Bart 2006 BARTNIK, Roman: Weiterentwicklung einer Technologiebasis für interaktive Gruppenarbeitsräume, Hochschule für Angewandte Wissenschaften Hamburg, Diplomarbeit (2006)

Bau 2002 BAUMÜLLER, Andreas: Topic Maps zur semantischen Strukturierung von Wissensbasen im Knowledge Management, Technischen Fachhochschule Wildau, Diplomarbeit (2002)

Ber 2005 BERKESSEL, Sascha; LIU, Rong: Proseminar XML Thema: SVG, Universität Koblenz-Landau (19.01.2005)

Berg 2006 BERGMANN, Johannes: Einheitliche Repräsentation heterogener Datenquellen mit Topic Maps, Technische Universität Darmstadt, Diplomarbeit (Januar 2006)

Burf 2006 BURFEINDT, Lars: Konstruktion einer Middleware für computergestützte Gruppenarbeit in ubiquitärer Systemumgebung, Hochschule für Angewandte Wissenschaften Hamburg, Diplomarbeit (2006)

DCMI The Dublin Core Metadata Initiative
http://dublincore.org/; http://dublincore.org/documents/dcmi-terms/
(Letztes Zugriffsdatum: 15.06.2007)

Eich 2006 EICHHORN, Jörg; LAMBECK, Christian; SCHIEKEL, Tom: Metadaten und semantische Verknüpfung im Personal Information Management; Technische Universität Dresden (2006)

Gero 2005 GEROIMENKO, V. und CHEN, C.: Visualizing Information Using SVG and X3D; Springer-Verlag London 2005 – ISBN 1852337907

Gnowsis http://www.gnowsis.org/
(Letztes Zugriffsdatum: 15.06.2007)

Google Google Desktop Search
http://desktop.google.com/de/features.html
(Letztes Zugriffsdatum: 15.06.2007)

Grün 2001 GRÜNWIED, Gudrun: Wissensrepräsentation mit XML Topic Maps (2001)
http://www.is-frankfurt.de/

Haystack http://haystack.lcs.mit.edu/
(Letztes Zugriffsdatum: 15.06.2007)

Held 2003 HELD, G.; NEUMANN, A. u. a.: SVG für die Webkartographie - Aktuelles und Zukünftiges (2003) - http://www.carto.net/
(Letztes Zugriffsdatum: 15.06.2007)

Herz 2007 HERZOG, Tobias: Dienste zur historisierbaren Positionsmessung lokalisierbarer Objekte in abgegrenzten Räumen, Hochschule für Angewandte Wissenschaften Hamburg, Bachelorarbeit (2007)

Literaturverzeichnis

Höf 2003 HÖFLER, Dr. Kai: Universität Karlsruhe (21.11.2003), Vorlesungsfolien: Software-Engineering Systemspezifikation (I / II) – http://www.aifb.uni-karlsruhe.de/Lehre/Winter2003-04/SE/Folien/Folien_SE05.pdf (Letztes Zugriffsdatum: 15.06.2007)

Hum 2004 HUMMEL, Benedikt: Einsatz und Nutzenpotenziale von Topic Maps: Ein State of the Art Bericht, Fachhochschule Potsdam – Diplomarbeit (2004)

IITB 2004 Fraunhofer-Institut Informations- und Datenverarbeitung IITB: visIT Wissensrepräsentation 2/2004; http://www.iitb.fraunhofer.de/servlet/is/8254/visIT_02_04.pdf (Letztes Zugriffsdatum: 15.06.2007)

IRIS http://www.openiris.org/ (Letztes Zugriffsdatum: 15.06.2007)

Joh 2002 JOHANSON, Brad ; FOX, Armando: The Event Heap: A Coordination Infrastructure for Interactive Workspaces, Stanford University (2002)

Joh 2004 JOHANSON, Brad; FOX, Armando: Extending Tuplespaces For Coordination in Interactive Workspaces. In: Journal of Systems and Software 69 (2004), Nr. 3

Lig 2006 Liggesmeyer, Prof. Dr., Technische Universität Kaiserslautern (2006), Vorlesungsfolien: Grundlagen Software Engineering – http://agde.informatik.uni-kl.de/teaching/gse/ws2006/material/folien/kapitel2/GSE_02_Prozesse_2s.pdf (Letztes Zugriffsdatum: 15.06.2007)

Lud 2005 LUDWIG, Lars (02-03-2005): Wie das Semantic Web das Persönliche Wissensmanagement revolutioniert http://www.semantic-web.at/36.24.24.article.lars-ludwig-wie-das-semantic-web-das-persoenliche-wissensmanagement-revolutioniert.htm (Letztes Zugriffsdatum: 15.06.2007)

Mei 2002 MEINIKE, Dr. Thomas: Scalable Vector Graphics (SVG) - ein XML-basierter Grafikstandard für 2D-Vektorgrafiken (2002) – http://www.et.fh-merseburg.de/person/meinike/PDF/TdF2002_Meinike.pdf (Letztes Zugriffsdatum: 15.06.2007)

MSNST MSN Suche Toolbar – http://desktop.msn.de/ (Letztes Zugriffsdatum: 15.06.2007)

Müll 2006 MÜLLER, Björn: Totally Ajax: Zurück zum "Fat Client"? (10/2006) – http://entwickler.de/zonen/portale/psecom,id,99,news,31727.html (Letztes Zugriffsdatum: 15.06.2007)

Müll 2007 MÜLLER, Christiane: IT-Infotek, Grundlagen des Software-Engineering, www.it-infothek.de (Letztes Zugriffsdatum: 15.06.2007)

Mund 2006 MUND, Horst: Berechtigungsstrukturen in kollaborativen Umgebungen, Hochschule für Angewandte Wissenschaften Hamburg, Diplomarbeit (2006)

Neu 2006 NEUMANN, Carola: Effizienzsteigerung von Diskussionsprozessen in einem neu gestalteten Konferenzraum, Hochschule für Angewandte Wissenschaften Hamburg, Diplomarbeit (2006)

Literaturverzeichnis

Nie 2004 NIEDERMEIER, Stephan: Das Cocoon Tutorial – http://cocoontutorial.logabit.com/ (Letztes Zugriffsdatum: 15.06.2007)

Nie 2007 NIEDERMEIER, Stephan: Cocoon 2 und Tomcat, Galileo Press Bonn (2007) – ISBN 3-89842-656-4

ontos ONTOS INTERNATIONAL AG: Ontologiebasiertes Wissensmanagement – http://www.ontos.ch (Letztes Zugriffsdatum: 15.06.2007)

Park 2002 PARK, Jack; HUNTING, Sam: ”XML Topic Maps: Creating and Using Topic Maps for the Web” , Addison Wesley Professional, 16.07.2002 (Nach Referenz durch Bau 2002)

Pep 2001 PEPPER, S.: The TAO of Topic Maps - finding the way in the age of infoglut (2001) - http://www.ontopia.net/topicmaps/materials/tao.html (Zugriffdatum: 15.06.2007)

Roß 2002 ROßBCH, HOLUBEK, PÖSCHMANN, RÖWEKAMP u. TABATT: Java Servlets und JSP mit Tomcat 4x; Software &Support Verlag GmbH (2002) – ISBN 3-935042-2

Rsc 2006 KAHLBRAND, RAASCH, ZUKUNFT: Hochschule für Angewandte Wissenschaften Hamburg (April 2006), Vorlesungsfolien: 025v-Benutzer-Entwickler-Kommunikation

Schä 2003 SCHÄFFER, Bruno: Durch dick und dünn - Probleme bei Thin-Clients, Javaspektrum (01/2003) - http://www.javaspektrum.de (Letztes Zugriffsdatum: 15.06.2007)

Schu 2000 SCHUMANN, H., MÜLLER, W.: Visualisierung – Grundlagen und allgemeine Methoden, Springer-Verlag Berlin Heidelberg (2000) – ISBN 3-540-64944-1

Spotlight http://www.apple.com/macosx/features/spotlight/ (Letztes Zugriffsdatum: 15.06.2007)

Stary 1996 STARY, C.: Interaktive Systeme – Software-Entwicklung und Software-Ergonomie. 2. Auflage, Braunschweig: Vieweg Verlag (1996) - ISBN 3-528-15384-9

Sult 2005 SULTANOV, Eldar: Integration und Visualisierung von XML Topic Maps, http://www.sigs.de/publications/js/2005/02/sultanow_JS_02_05.pdf (Letztes Zugriffsdatum: 15.06.2007)

SWSch-a SEMANTIC WEB SCHOOL Zentrum für Wissenstransfer: Wissensrepräsentation – http://www.semantic-web.at/10.36.90.catchword.kontext.wissensrepraesentation.htm (Letztes Zugriffsdatum: 15.06.2007)

SWSch-b SEMANTIC WEB SCHOOL Zentrum für Wissenstransfer: Wissensrepräsentation: Leo Sauermann: „Spätestens 2008 wird der Semantic Desktop in Europa Realität.“ (13.02.2006) – http://www.semantic-web.at/36.78.78.article.leo-sauermann-semantic-desktop.htm (Letztes Zugriffsdatum: 15.06.2007)

Tan 2002 TANENBAUM, Andrew; STEEN, Marten van: Distributed Systems: Principles and Paradigms. Upper Saddle River, NJ : Prentice Hall, 2002. – ISBN 0131217860

Literaturverzeichnis

tinyTIM http://tinytim.sourceforge.net/
(Letztes Zugriffsdatum: 15.06.2007)

TM4J http://tm4j.sourceforge.net/
(Letztes Zugriffsdatum: 15.06.2007)

TMAPI http://www.tmapi.org/
(Letztes Zugriffsdatum: 15.06.2007)

W3C RDF W3C: Resource Description Framework (RDF), W3C Recommendation (10.02.2004) – http://www.w3.org/RDF/
(Letztes Zugriffsdatum: 15.06.2007)

W3C SVG W3C: Scalable Vector Graphics (SVG) 1.0 Specification, W3C Recommendation (04.09.2001) - http://www.w3.org/TR/SVG/; http://www.w3.org/2000/svg
(Letztes Zugriffsdatum: 15.06.2007)

Wenz 2002 WENZ, Christian: JavaScript; Galileo Press GmbH 2002 – ISBN 3-89842-234-8

Wid 2002 WIDHALM, Richard, MÜCK, Prof. Dr. Thomas: Topic Maps: Semantische Suche im Internet, Springer-Verlag Berlin Heidelberg 2002 – ISBN3-540-41719-2

WiSoWiWiKi Universität Erlangen, WiKi der Wirtschaftsinformatik:
http://www.wi3.uni-erlangen.de/anwendungen/wiwiki/wiki/
Das_WiKi_der_Wirtschaftsinformatik
http://www.wi3.uni-erlangen.de/anwendungen/wiwiki/wiki/ Kategorie:Prototyping
(Letztes Zugriffsdatum: 15.06.2007)

Wöhr 2004 WÖHR, Heiko: Web-Technologien; dpunkt.verlag 2004 – ISBN 3-89864-247-X

XMLCH XML Clearinghouse: XML im deutsch-sprachigen Raum, Standards und Entwicklungen, XML Topic Maps (XTM) –
http://www.xml-clearinghouse.de/standards/20/
(Letztes Zugriffsdatum: 15.06.2007)

XTM 2001 PEPPER, S. and Moore, G: *XML Topic Maps (XTM) 1.0;* TopicMaps.org (2001) - http://www.topicmaps.org/xtm/1.0/
(Letztes Zugriffsdatum: 15.06.2007)

Zöll 2006 ZÖLLNER, Bernhard: Thin Client vs. Fat Client (11/2006) –
http://www.contentmanager.de/magazin/
artikel_1241_thin_client_fat_client.html
(Letztes Zugriffsdatum: 15.06.2007)

A. Anhang

A.1. Abkürzungen

#PCDATA	Parsed Character Data (vom Parser erkannte Zeichenfolgen)
API	Application Programming Interface
CMS	Content Management System
CSS	Cascading Style Sheets
DC	Dublin Core
DMS	Document Management System
DOM	Document Object Model
DTD	Document Typ Definition
ECMA	European Computer Manufacturers Association
EJB	Enterprise JavaBean
GUI	Graphical User Interface
HTML	Hypertext Markup Language
ISO	Internation Standards Organisation
JSP	Java Server Pages
KM	Knowledge Management
KORA	Konferenzraum
LAN	Local Area Network
OWL	Web Ontology Language
PDA	Personal Digital Assistant
PDF	Portable Document Format
RDF	Ressource Description Framework
SAX	Simple API for XML
SVG	Scalable Vector Graphics
TMAPI	Topic Map Application Programming Interface
TMQL	Topic Map Query Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locator

A.1 Abkürzungen

W3C	World Wide Web Consortium
WLAN	Wireless Local Area Network
WML	Wireless Markup Language
WWW	World Wide Web
XLink	XML Linking Language
XML	eXtensibel Markup Language
XSL	Extensible Stylesheet Language
XSLT	XSL Transformations
XTM	XML Topic Maps

A.2. Glossar

Axiom
Axiome sind Aussagen, die immer wahr sind. Diese werden normalerweise dazu verwendet Wissen zu repräsentieren, das nicht aus anderen Konzepten abgeleitet werden kann, z.B. zwischen Amerika und Europa existiert keine Zugverbindung

Client
Als Client (engl. = Kunde) bezeichnet man ein Softwareprogramm, eine Hardware (Computer) oder eine Kombination aus beidem, welche auf einen Server zugreift um die dort angebotenen Dienste zu nutzen.

Dublin Core
Dublin Core ist ein Metadaten-Schema zur Beschreibung von Dokumenten und anderen Objekten im Internet. Urheber dieses Schemas ist die „Dublin Core Metadata Initiative“ (DCMI).

Frames
Frames (engl.: "Rahmen") sind Konstrukte zur Wissensrepräsentation, die komplementär zur Repräsentation von Wissen mittels Logik (zum Beispiel Prädikatenlogik) sind.

Glossar
Ein Glossar ist eine Liste von erklärungsbedürftigen Wörtern mit Erläuterung.

Hyperlink
Als Hyperlink (kurz Link) bezeichnet man einen Verweis auf ein anderes Dokument in einem Hypertext, der durch das Hypertextsystem automatisch verfolgt werden kann.

Java-Applets
Ein Java-Applet ist ein Computerprogramm, das in einem Webbrowser ausgeführt wird. Java-Applets wurden eingeführt, um clientseitig in Web-Seiten Programme ablaufen lassen zu können, die im Webbrowser arbeiten und direkt mit dem Benutzer interagieren können, ohne Daten über die Leitung zum Server versenden zu müssen.
Üblicherweise werden Java-Applets von HTML-Seiten aufgerufen. Um sie ausführen zu können, muss der jeweilige Webbrowser allerdings über eine entsprechende Laufzeitumgebung (JVM – Java Virtuelle Maschine) verfügen.

JavaScript
JavaScript ist eine objektbasierte Skriptsprache mit Elementen aus den funktionalen Programmiersprachen. Die Ausführung des JavaScipt-Codes erfolgt durch einen im Browser integrierten Interpreter. JavaScript wird eingesetzt, um ansonsten statisches HTML mit dynamischen Elementen anzureichern. JavaScript wurde ursprünglich für den Netscape Navigator entwickelt. Mittlerweile finden sich allerdings in praktisch allen grafischen Browsern entsprechende Interpreter.

Katalog
Ein Katalog ist ein, nach einer bestimmten Systematik (alphabetisch, nach Schlagworten o. ä.) aufgebautes Verzeichnis, das vor allem der Übersicht über Sammlungen von Gegenständen (z. B. Bücher, Werkzeug, Bilder) dient.

A.2 Glossar

Klassifikation
Eine Klassifikation ist eine planmäßige Darstellung von Klassen, Kategorien oder anderen abstrakten Konzepten, die nach bestimmten Ordnungsprinzipien (einem System) gestaltet ist.

Metadaten
Als Metadaten oder Metainformationen bezeichnet man allgemein Daten, die Informationen über andere Daten enthalten. Bei den beschriebenen Daten handelt es sich oft um größere Datensammlungen (Dokumente) wie Bücher oder Dateien. Typische Metadaten zu einem Buch sind beispielsweise der Name des Autors, die Auflage, das Erscheinungsjahr, der Verlag und die ISBN. Zu den Metadaten einer Computerdatei zählen u. a. der Dateiname, die Zugriffsrechte und das Datum der letzten Änderung.

MVC (Model-View-Controller-Paradigma)
Der Begriff Model-View-Controller (MVC) bezeichnet ein Architekturmuster zur Aufteilung von Softwaresystemen in die drei Einheiten: Datenmodell (engl. Model), Präsentation (engl. View) und Programmsteuerung (engl. Controller).
Ziel des Modells ist ein flexibles Programmdesign, um u.a. eine spätere Änderung oder Erweiterung einfach zu halten und die Wiederverwendbarkeit der einzelnen Komponenten zu ermöglichen. Außerdem sorgt das Modell bei großen Anwendungen für eine gewisse Übersicht und Ordnung durch Reduzierung der Komplexität.

Ontologie
Ontologiesprachen, wie RDF oder OWL, sind wie Datenmodelle zu verstehen und somit durchaus mit einem Relationalen Datenbankmodell oder einem XML-Schema vergleichbar. Eine Ontologie ist: "Ein formales Modell eines Wissensraumes". Ontologien dienen aber auch dazu, einen gemeinsamen Sprachgebrauch z. B. innerhalb einer Organisation zu etablieren.

Ontologien:
Datenkonstrukte, die die Struktur einer Wissensdomäne widerspiegeln und Kategorien, Vokabulare und Informationen über Beziehungen enthalten.

Semantisches Netz
Ein semantisches Netz ist ein formales Modell von Begriffen und Beziehungen (Relationen). Informatik und die künstliche Intelligenz nutzen semantische Netze zur Wissensrepräsentation. Gelegentlich spricht man auch von einem *Wissensnetz*. Meist wird ein semantisches Netz durch einen verallgemeinerten Graphen repräsentiert. Die Knoten des Graphen stellen dabei die Begriffe dar. Beziehungen zwischen den Begriffen werden durch die Kanten des Graphen realisiert.

Server
Der Begriff Server (engl.: to serve = bedienen) bezeichnet eine Softwareprogramm, eine Hardware (Computer) oder eine Kombination aus beidem, welche einem mit ihm verbundenen Client Zugang zu speziellen Diensten verschafft.

Servlet
Als Servlets bezeichnet man Java-Klassen, deren Instanzen innerhalb eines Applicationservers Anfragen von Clients entgegen nehmen und beantworten.

A.2 Glossar

Sidecar-Datei
Eine Sidecar-Datei ist eine begleitende Datei, die als „Anhängsel“ zu einer anderen Datei gehört und weitere Informationen oder Einstellungen zur Hauptdatei beinhaltet. Verwendet wird dieses Verfahren häufig bei Bilddateien. Die Sidecar-Datei enthält in diesem Fall Formatierungseinstellungen, z. B. Helligkeit, Kontrast und Farbwerte.

SMIL (Synchronized Multimedia Integration Language)
SMIL ist ein auf XML basierender, vom W3C entwickelter Standard für eine Auszeichnungssprache für zeitsynchronisierte, multimediale Inhalte. SMIL ermöglicht die Einbindung und Steuerung von Multimedia-Elementen wie Audio, Video, Text und Grafik in Webseiten; SMIL-Dateien können mit Java-Applets und Servlets oder CGI-Skripten verknüpft werden und so beispielsweise auf eine Datenbank zugreifen. Als Dateierweiterung wird .smi oder .smil verwendet.

SVG (Scalable Vector Graphic)
SVG ist ein Standard zur Beschreibung zweidimensionaler Vektorgrafiken in XML-Syntax. SVG wurde im September 2001 vom W3C als Empfehlung veröffentlicht. Ein Großteil des Sprachumfangs von SVG kann bereits von aktuellen Webbrowsern, wie z. B. Mozilla Firefox oder Opera dargestellt werden. Andere können durch Plug-Ins erweitert werden.

Taxonomie
Die Taxonomie ist primär die sprachwissenschaftliche Klassifikation aller Gegenstände und Ereignisse in Gruppen bzw. Kategorien. Naturwissenschaftliche Disziplinen verwenden den Begriff der Taxonomie allgemein für eine in der Regel hierarchische Klassifikation (Klasse, Unterklasse usw.).

Thesaurus
Ein Thesaurus ist ein kontrolliertes Vokabular (Wörterbuch), dessen Begriffe durch Relationen miteinander verbunden sind.

Topic Map
Topic Maps ist ein abstraktes Modell sowie ein XML-basiertes Datenformat zur Formulierung von Wissensstrukturen, auch Ontologien genannt. Topic Maps wurden 1999 als ISO-Standard ISO/IEC 13250 normiert und später als XML Topic Maps (XTM) in XML formuliert.

Ubicomp
Siehe Ubiquitous Computing

Ubiquitous Computing
Der Begriff wurde erstmals 1988 von Mark Weiser verwendet und 1991 in seinem Aufsatz „The Computer for the 21st Century“ geprägt. Nach seiner Vision wird der (Personal-) Computer als Gerät verschwinden bzw. durch "intelligente Gegenstände" ersetzt werden. Ziel ist dabei, dass die immer kleiner werdenden Computer bei der normalen Arbeit unterstützen und dabei an den Rand der menschlichen Aufmerksamkeit rücken.

URI (Uniform Resource Identifier)
Eine URI ist eine Zeichenfolge, die zur Identifizierung einer abstrakten oder physischen Ressource dient. URIs werden zur Bezeichnung von Ressourcen (wie Webseiten, sonstigen Dateien, Aufruf von Webservices, aber auch z. B. E-Mail-Empfängern) im Internet und dort vor allem im WWW eingesetzt.

A.2 Glossar

URL (Uniform Resource Locator)
Als URL bezeichnet man eine Unterart von URIs. URLs identifizieren eine Ressource über ihren primären Zugriffsmechanismus (häufig http oder ftp) und den Ort der Ressource in Computernetzwerken.

Webbrowser
Ein Webbrowser ist ein HTTP-Client-Programm, das HTML-Dokumente darstellen kann. Viele moderne Webbrowser können darüber hinaus noch weitere Medienfomate darstellen, häufig auch XML.

Webserver
Ein Webserver ist ein Prozess, der über das Hypertext Transfer Protocol (HTTP) mit anderen Prozessen, den Clients, Daten austauscht.

XML (Extensible Markup Language)
Die XML (engl. für „erweiterbare Auszeichnungssprache“), ist eine Auszeichnungssprache zur Darstellung hierarchisch strukturierter Daten in Form von Textdateien. XML wird bevorzugt für den Austausch von Daten zwischen unterschiedlichen IT-Systemen, speziell über das Internet, eingesetzt.

A.3. XML Topic Map DTD

```
<!-- topicMap: Topic Map document element ..... -->

 <!ELEMENT topicMap ( topic | association | mergeMap )* >
 <!ATTLIST topicMap
     id           ID             #IMPLIED
     xmlns        CDATA          #FIXED 'http://www.topicmaps.org/xtm/1.0/'
     xmlns:xlink  CDATA          #FIXED 'http://www.w3.org/1999/xlink'
     xml:base     CDATA          #IMPLIED
 >

 <!-- topic: Topic element ..... -->

 <!ELEMENT topic ( instanceOf*, subjectIdentity?, ( baseName | occurrence )* )>
 <!ATTLIST topic
     id           ID             #REQUIRED
 >

 <!-- instanceOf: Points To a Topic representing a class ..... -->

 <!ELEMENT instanceOf  ( topicRef | subjectIndicatorRef ) >
 <!ATTLIST instanceOf
     id           ID             #IMPLIED
 >

 <!-- subjectIdentity: Subject reified by Topic ..... -->

 <!ELEMENT subjectIdentity ( resourceRef?, ( topicRef | subjectIndicatorRef )* )>
 <!ATTLIST subjectIdentity
     id           ID             #IMPLIED
 >

 <!-- topicRef: Reference to a Topic element ..... -->

 <!ELEMENT topicRef  EMPTY >
 <!ATTLIST topicRef
     id           ID             #IMPLIED
     xlink:type   NMTOKEN        #FIXED 'simple'
     xlink:href   CDATA          #REQUIRED
 >

 <!-- subjectIndicatorRef: Reference to a Subject Indicator ..... -->

 <!ELEMENT subjectIndicatorRef  EMPTY >
 <!ATTLIST subjectIndicatorRef
     id           ID             #IMPLIED
     xlink:type   NMTOKEN        #FIXED 'simple'
     xlink:href   CDATA          #REQUIRED
```

A.3 XML Topic Map DTD

```
>

<!-- baseName: Base Name of a Topic ..... -->

<!ELEMENT baseName  ( scope?, baseNameString, variant* ) >
<!ATTLIST baseName
    id          ID          #IMPLIED
>

<!-- baseNameString: Base Name String container ..... -->

<!ELEMENT baseNameString  ( #PCDATA ) >
<!ATTLIST baseNameString
    id          ID          #IMPLIED
>

<!-- variant: Alternate forms of Base Name ..... -->

<!ELEMENT variant  ( parameters, variantName?, variant* ) >
<!ATTLIST variant
    id          ID          #IMPLIED
>

<!-- variantName: Container for Variant Name ..... -->

<!ELEMENT variantName  ( resourceRef | resourceData ) >
<!ATTLIST variantName
    id          ID          #IMPLIED
>

<!-- parameters: Processing context for Variant ..... -->

<!ELEMENT parameters  ( topicRef | subjectIndicatorRef )+ >
<!ATTLIST parameters
    id          ID          #IMPLIED
>

<!-- occurrence: Resources regarded as an Occurrence ..... -->

<!ELEMENT occurrence ( instanceOf?, scope?, ( resourceRef | resourceData ) ) >
<!ATTLIST occurrence
    id          ID          #IMPLIED
>

<!-- resourceRef: Reference to a Resource ..... -->

<!ELEMENT resourceRef  EMPTY >
<!ATTLIST resourceRef
    id          ID          #IMPLIED
    xlink:type  NMTOKEN     #FIXED 'simple'
    xlink:href  CDATA       #REQUIRED
>
```

A.3 XML Topic Map DTD

```
<!-- resourceData: Container for Resource Data ..... -->

<!ELEMENT resourceData  ( #PCDATA ) >
<!ATTLIST resourceData
    id            ID              #IMPLIED
>

<!-- association: Topic Association  ..... -->

<!ELEMENT association ( instanceOf?, scope?, member+ ) >
<!ATTLIST association
    id            ID              #IMPLIED
>

<!-- member: Member in Topic Association ..... -->

<!ELEMENT member ( roleSpec?, ( topicRef | resourceRef | subjectIndicatorRef )* ) >
<!ATTLIST member
    id            ID              #IMPLIED
>

<!-- roleSpec: Points to a Topic serving as an Association Role .. -->

<!ELEMENT roleSpec  ( topicRef | subjectIndicatorRef ) >
<!ATTLIST roleSpec
    id            ID              #IMPLIED
>

<!-- scope: Reference to Topic(s) that comprise the Scope ..... -->

<!ELEMENT scope  ( topicRef  | resourceRef | subjectIndicatorRef )+ >
<!ATTLIST scope
    id            ID              #IMPLIED
>

<!-- mergeMap: Merge with another Topic Map ..... -->

<!ELEMENT mergeMap  ( topicRef | resourceRef | subjectIndicatorRef )* >
<!ATTLIST mergeMap
    id            ID              #IMPLIED
    xlink:type    NMTOKEN         #FIXED 'simple'
    xlink:href    CDATA           #REQUIRED
>

<!-- end of XML Topic Map (XTM) 1.0 DTD -->
```

A.4. XML-Beispiel “Verdi"

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<topicMap
  id="operatm-tm"
  xmlns="http://www.topicmaps.org/xtm/1.0/"
  xmlns:xlink="http://www.w3.org/1999/xlink"
>
<!-- ********* Types ***********-->
<topic id="person">
  <baseName>
    <baseNameString>Person</baseNameString>
  </baseName>
</topic>

<topic id="composer">
<instanceOf><topicRef xlink:href="#person"/></instanceOf>
  <baseName>
    <baseNameString>Komponist</baseNameString>
  </baseName>
</topic>

<topic id="place">
  <baseName>
    <baseNameString>Stadt</baseNameString>
  </baseName>
</topic>

<topic id="language">
  <baseName>
    <baseNameString>Sprache</baseNameString>
  </baseName>
</topic>

<topic id="music">
  <baseName>
    <baseNameString>Musik</baseNameString>
  </baseName>
</topic>

<topic id="opera">
  <instanceOf><topicRef xlink:href="#music"/></instanceOf>
  <baseName>
    <baseNameString>Oper</baseNameString>
  </baseName>
</topic>

<topic id="webpage">
  <baseName>
    <baseNameString>webpage</baseNameString>
  </baseName>
</topic>

<topic id="portrait">
  <baseName>
```

A.4 XML-Beispiel “Verdi"

```
    <baseNameString>Portrait</baseNameString>
  </baseName>
</topic>

<topic id="born-date">
  <baseName>
    <baseNameString>Geburtsdatum</baseNameString>
  </baseName>
</topic>

<topic id="died-date">
  <baseName>
    <baseNameString>Todestag</baseNameString>
  </baseName>
</topic>

<topic id="born-in">
  <baseName>
    <baseNameString>geboren-in</baseNameString>
  </baseName>
</topic>

<topic id="died-in">
  <baseName>
    <baseNameString>gestorben-in</baseNameString>
  </baseName>
</topic>

<topic id="composed-by">
  <baseName>
    <baseNameString>komponierte</baseNameString>
  </baseName>
</topic>

<topic id="premiere-date">
  <baseName>
    <baseNameString>Premiere</baseNameString>
  </baseName>
</topic>

<!-- ********* Topics ***********-->
<topic id="verdi">
  <instanceOf><topicRef xlink:href="#composer"/></instanceOf>
  <instanceOf><topicRef xlink:href="#person"/></instanceOf>
  <!-- born-in: le-roncole (1813 (10 Oct)) -->
  <!-- died-in: milano (1901 (27 Jan)) -->
  <baseName>
    <baseNameString>Verdi, Giuseppe</baseNameString>
  </baseName>
    <occurrence>
    <instanceOf><topicRef xlink:href="#portrait"/></instanceOf>
    <resourceRef xlink:href="file:///E:\...\TopcMap\XTM\verdi.jpg"/>
  </occurrence>
  <occurrence>
    <instanceOf><topicRef xlink:href="#born"/></instanceOf>
    <resourceData>1813 (10 Oct)</resourceData>
  </occurrence>
```

A.4 XML-Beispiel “Verdi"

```
  <occurrence>
    <instanceOf><topicRef xlink:href="#died"/></instanceOf>
    <resourceData>1901 (27 Jan)</resourceData>
  </occurrence>
</topic>

<topic id="aida">
  <instanceOf><topicRef xlink:href="#opera"/></instanceOf>
  <!-- premiere: cairo-opera (1871 (24 Dec)) -->
  <!-- written-by: ghislanzoni locle bey -->
  <!-- published-by: ricordi-p -->
  <!-- aria-in-opera: ritorna-vincitor -->
  <!-- character-in-opera: aida-c amonasro amneris king-of-egypt radames
ramfis -->
  <!-- setting: egypt -->
  <baseName>
    <baseNameString>Aida</baseNameString>
  </baseName>
  <occurrence>
    <instanceOf><topicRef xlink:href="#premiere-date"/></instanceOf>
    <resourceData>1871 (24 Dec)</resourceData>
  </occurrence>
</topic>

<topic id="milano">
<instanceOf><topicRef xlink:href="#place"/></instanceOf>
  <baseName>
    <baseNameString>Milano</baseNameString>
  </baseName>
    <occurrence>
      <instanceOf><topicRef xlink:href=""/></instanceOf>
      <resourceData></resourceData>
    </occurrence>
</topic>

<topic id="le-roncole">
<instanceOf><topicRef xlink:href="#place"/></instanceOf>
  <baseName>
    <baseNameString>Le Roncole</baseNameString>
  </baseName>
    <occurrence>
      <instanceOf><topicRef xlink:href=""/></instanceOf>
      <resourceData>Geburtsort Giuseppe Verdi</resourceData>
    </occurrence>
</topic>

<topic id="italienisch">
<instanceOf><topicRef xlink:href="#language"/></instanceOf>
  <baseName>
    <baseNameString>italienisch</baseNameString>
  </baseName>
    <occurrence>
      <instanceOf><topicRef xlink:href=""/></instanceOf>
      <resourceData></resourceData>
    </occurrence>
</topic>
```

A.4 XML-Beispiel “Verdi"

```
<!-- ********* Associations ***********-->
<association>
  <instanceOf><topicRef xlink:href="#born-in"/></instanceOf>
  <member>
    <roleSpec><topicRef xlink:href="#person"/></roleSpec>
    <topicRef xlink:href="#verdi"/>
  </member>
  <member>
    <roleSpec><topicRef xlink:href="#place"/></roleSpec>
    <topicRef xlink:href="#le-roncole"/>
  </member>
</association>

<association>
  <instanceOf><topicRef xlink:href="#died-in"/></instanceOf>
  <member>
    <roleSpec><topicRef xlink:href="#person"/></roleSpec>
    <topicRef xlink:href="#verdi"/>
  </member>
  <member>
    <roleSpec><topicRef xlink:href="#place"/></roleSpec>
    <topicRef xlink:href="#milano"/>
  </member>
</association>

<association>
  <instanceOf><topicRef xlink:href="#composed-by"/></instanceOf>
  <scope><topicRef xlink:href="#music"/></scope>
  <member>
    <roleSpec><topicRef xlink:href="#composer"/></roleSpec>
    <topicRef xlink:href="#verdi"/>
  </member>
  <member>
    <roleSpec><topicRef xlink:href="#opera"/></roleSpec>
    <topicRef xlink:href="#aida"/>
  </member>
</association>


</topicMap>
```

Versicherung über Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §24(5) ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 21.06.2007 _____
Ort, Datum

_____
Unterschrift