Studiengang:	Informatik
Prüfer:	Prof. Dr.-Ing. habil. Bernhard Mitschang
Betreuer:	Dipl.-Inf. Matthias Großmann
begonnen am:	01. November 2001
beendet am:	30. April 2002
CR-Klassifikation:	H.2.8, H.3.1, H3.3

Studienarbeit Nr. 1834

Ortsbasierter Web-Zugriff

Mihály Sütö

Institut für Parallele und Verteilte
Höchstleitungsrechner (IPVR)
Abteilung Anwendersoftware
Universität Stuttgart
Breitwiesenstraße 20-22
D-70565 Stuttgart

Kurzfassung

Die Forschergruppe NEXUS entwickelt eine offene und globale Infrastruktur für ortsbezogene Anwendungen, die mobile Benutzer bei der Erledigung vielfältiger Aufgaben unterstützen sollen. Das Weltmodell, das diesen Anwendungen zu Grunde liegt, das *Augmented World Model*, ist eine, durch virtuelle Objekte erweiterte, Abbildung der realen Welt. Virtuelle Objekte wie Webseiten des World Wide Webs dienen der Informationsspeicherung und können durch mobile Geräte dargestellt werden. Der *Spatial Model Server*, der Objekte des *Augmented World Models* speichert und mobile Anwendungen mit Information versorgt, sollte nach Möglichkeit automatisch mit Daten gefüllt werden.

Diese Studienarbeit beschäftigt sich mit der Suche und der Speicherung von Webseiten mit Ortsinformation. Wichtige Aufgabenteile sind die Zusammenfassung von Metadatenstandards für Webseiten und die Untersuchung von Suchmaschinen auf ihre Fähigkeit hin, Webseiten mit Ortsinformation aufzuspüren. Die Hauptaufgabe ist die Implementierung eines Suchwerkzeuges, das Webseiten im World Wide Web nach Ortsinformation durchsucht und gefundene Metadaten in einen *Spatial Model Server* einträgt.

Danksagungen

An dieser Stelle möchte ich meinem Betreuer, Herrn Dipl.-Inf. Matthias Großmann und Frau Dipl.-Inf. Daniela Nicklas danken, die mir bei der Erstellung dieser Studienarbeit stets hilfreich zur Seite standen.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

KAPITEL 1 EINFÜHRUNG

Dieses Kapitel gibt einen Überblick über die zu realisierende Aufgabe und skizziert die zur Lösung führende Vorgehensweise.

1.1 Motivation

Die Technologieentwicklung der letzten Jahre hat die schnelle Verbreitung der Computer und die Ansammlung großer, elektronischer Datenbestände ermöglicht. Information gewinnt immer mehr an Bedeutung, sowohl in der Wirtschaft, als auch im Privatsektor. Die benötigten Daten sollten schnellstmöglich und überall zugänglich sein. Wo wir uns befinden, bestimmt oft, was wir tun, deswegen ist ortsbezogene Information besonders interessant.

Kleine, mobile, vernetzte Computer, ausgestattet mit den entsprechenden Applikationen, könnten diesen Wissensdurst befriedigen. Dabei wäre es sinnvoll, die mobilen Geräte mit bereits existierenden Datenquellen, wie den World Wide Web, zu verbinden. Die Suche und Bereitstellung von ortsbezogener Information im World Wide Web ist in diesem Zusammenhang eine wichtige Aufgabe.

1.2 Aufgabenstellung

Die Forschergruppe NEXUS entwickelt eine offene und globale Infrastruktur für ortsbezogene Anwendungen für mobile Benutzer. Im Zentrum der Infrastruktur steht das *Augmented World Model*. Das *Augmented World Model* ist eine durch virtuelle Objekte erweiterte Abbildung der realen Welt. Diese Objekte können der Einbindung externer Informationsquellen, wie zum Beispiel des World Wide Web dienen. Solche virtuelle Objekte werden im *Spatial Model Server* gespeichert und stehen für die mobilen Anwendungen bereit.

Das Ziel dieser Studienarbeit ist die Suche und die Bereitstellung von Webseiten des World Wide Web für die NEXUS Infrastruktur. Für mobile Anwendungen sind vorrangig Webseiten mit eingebetteter Ortsinformation interessant. Metadaten, die eine standardisierte Beschreibung der Webseiten erlauben und die gesuchte Ortsinformation tragen, sind für diese Arbeit von besonderer Bedeutung.

Folgende Teilaufgaben sind zu erledigen:

Die Zusammenfassung existierender Metadaten-Standards, die eine Angabe von Ortsinformation für Webseiten mittels HTML META-Tags und *Ressource Description Framework* ermöglichen, soll die Suchaufgabe vorbereiten.

Danach soll die Eignung von Suchmaschinen zum Auffinden von Webseiten mit Ortsinformation untersucht werden.

Die Erweiterung des *Augmented World Models* um ein Objekt zur Repräsentation von Webseiten, soll die Speicherung von Metadaten im *Spatial Model Server* ermöglichen.

Die Hauptaufgabe ist die Implementierung der Suchkomponente und einer Schnittstelle zum *Spatial Model Server.*

Abschließend soll eine Bewertung des implementierten Systems erfolgen.

1.3 Vorgehensweise

Die wichtigsten in der Aufgabenstellung erwähnten Aufgabenteile werden jeweils in einem eigenen Kapitel behandelt. Kapitel 2 befasst sich mit der Möglichkeit der Metadatendarstellung mit HTML META-Tags und dem *Resource Description Framework*. Dazu gehört sowohl die Betrachtung der grundsätzlichen Strukturen für die Metadatenangabe als auch die Untersuchung von Metadatenformaten, mit denen Metadaten dargestellt werden.

Der häufigste Weg für die Suche von Information, der im World Wide Web eingeschlagen wird, ist die Benutzung einer Suchmaschine. In Kapitel 3 werden verschiedene Arten von Suchwerkzeugen betrachtet, und auf ihre Eignung für das Auffinden von Webseiten mit Ortsinformation hin untersucht. Dabei werden sowohl kommerzielle als auch frei verfügbare und konfigurierbare Suchwerkzeuge unter die Lupe genommen.

Gefundene Webseiten mit eingebetteter Ortsinformation sollten, durch entsprechende Objekte dargestellt, in einem *Spatial Model Server* abgelegt werden. Kapitel 4 befasst sich mit der Erstellung von Datenstrukturen, die das Ablegen von Webseiten in einem solchen Server ermöglichen. Dazu ist sowohl die Betrachtung vorhandener Datenstrukturen des *Augmented World Models* als auch die Analyse der zu speichernden Daten notwendig.

Nachdem die Darstellung von Webseiten mit Ortsinformation in der NEXUS Infrastruktur geklärt ist, bleibt noch die Aufgabe der Suche und der Speicherung solcher Webseiten. In Kapitel 5 wird ein zum Teil selbst entwickeltes Programm vorgestellt, das diese Aufgabe vollständig löst.

Eine Bewertung des Systems an Hand mehrerer Testläufe befindet sich in Kapitel 6, während in Kapitel 7 eine kurze Zusammenfassung und ein Ausblick in die Zukunft die Arbeit abrundet.

KAPITEL 2 METADATEN FÜR WEBSEITEN

Dieses Kapitel fasst existierende Metadatenstandars für Ressourcen mit Ortsinformation zusammen und zeigt die Möglichkeiten der Speicherung von Metadaten in Webseiten.

2.1 Einführung

Im Allgemeinen sind Metadaten Daten über Daten [Day00]. Im Bereich der Informationsverarbeitung spricht man von Metadaten, wenn man strukturierte Information über elektronische oder nicht-elektronische Ressourcen meint. Das geläufigste Beispiel für Metadaten ist der Katalog einer Bibliothek. In dem Katalog findet man unter anderem den Titel, den Autor und das Erscheinungsjahr für jedes Buch. Dies sind Daten über die Bücher, die wiederum Daten enthalten.

Am Beispiel des Bibliothekkatalogs wird deutlich, dass Metadaten sehr hilfreich sind; man benutzt sie täglich ohne sich dessen bewusst zu sein. Ein strukturiertes Ablegen mit definierter Ordnung ist eine wichtige Vorbedingung für die erfolgreiche Suche in größeren Datenmengen. Ein weiteres Beispiel ist ein Telefonbuch, das den Namen, die Anschrift und die Telefonnummer vieler Personen enthält.

2.1.1 Darstellung von Metadaten

Ein Metadaten-Element, so wie es in dieser Arbeit verwendet wird, ist ein Tupel aus einer “Eigenschaft” (*property*) und aus einem Wert (*value*). Die “Eigenschaft”, beschrieben durch einen bestimmten Wert, modelliert einen Aspekt der Ressource. Die “Eigenschaft” (oder Name eines Metadatenelementes) stammt meist aus einem vordefinierten Namensraum (z.B. {Titel, Autor, Beschreibung}). Der dem Elementnamen zugeordnete Wert muss einem vordefinierten Datentyp und Format entsprechen (z.B. *ISO-3166* [IMA02] für die abgekürzte Darstellung von Ländernamen für den Fall Elementname = “Land”). Ein Satz von Metadaten-Elementen bildet die Gesamtbeschreibung einer Ressource.

Ein Metadatenelement, das Information über diese Studienarbeit trägt, könnte so aussehen:

<Autor: Mihály Sütö>

Der Elementname ist in diesem Fall “Autor” und der Wert ist “Mihály Sütö”. Eine Gesamtbeschreibung (sehr vereinfacht) dieser Studienarbeit könnte so aussehen:

<Title: Ortsbasierter Webzugriff>
<Autor: Mihály Sütö>
<Beschreibung: Eine Studienarbeit über ortsbasierten Webzugriff>

2.1.2 Metadaten im World Wide Web

Seiten im World Wide Web sind nicht strukturiert abgelegt. Die Zugehörigkeit mehrerer Webseiten zu einer Domain garantiert keineswegs eine inhaltliche Verwandtschaft. Das World Wide Web, verglichen mit einer Bibliothek, wäre eine, in die jeder nach Belieben ein Buch hineinstellen darf, ohne eine Ordnung beachten zu müssen. In dieser Bibliothek wären dann Angestellte (Entsprechung für eine Suchmaschine) tätig, die ein Teilbestand der Bücher durchgeblättert hätten und, ohne sie verstanden zu haben, Empfehlungen für das gesuchte Thema aussprechen würden. Eine Ausnahme bilden Webseitenkataloge wie z.B. *Yahoo!*. Sie ordnen Webseiten in verschiedene Kategorien ein, aus denen der Benutzer das entsprechende Themengebiet auswählen kann. Diese Kataloge decken einen eher kleinen Ausschnitt des World Wide Webs ab. Eine genaue Untersuchung von Suchmaschinen findet in Kapitel 3 statt.

Metadaten für Webseiten können strukturierte Information über den Inhalt der Seite enthalten. Sie erleichtern dadurch die nachträgliche Kategorisierung. Die Benutzung eines standardisierten Metadatenformats für Webseiten könnte eine hochspezifische Suche im World Wide Web ermöglichen. Suchmaschinen könnten an Hand der genaueren Beschreibung eine differenzierte Suche durchführen und dem Benutzer nur relevante Ergebnisse liefern.

Voraussetzung der maschinellen Verarbeitung von Metadaten (wie bei Suchmaschinen der Fall) ist die maschinelle Lesbarkeit dieser Daten. Eine Lesbarkeit für Menschen erleichtert das Verständnis und die Erstellung von Metadaten, ist aber nicht unbedingt erforderlich. Obwohl die meisten Metadatenformate für Menschen lesbar sind, und durch ihre einfache Syntax das Verständnis erleichtern, ist ihre Erstellung in den meisten Fällen auch durch Softwarewerkzeuge möglich.

Nach David Bearman und Ken Sochats [BS94] gibt es sechs Arten von Metadaten:

1. Metadaten zur Identifikation
2. Metadaten zu Nutzungskonditionen
3. Metadaten zu strukturellen Aspekten
4. Metadaten zum Kontext
5. Metadaten zum Inhalt
6. Metadaten zur Nutzungsgeschichte

Bisher finden zwei dieser Aspekte im Zusammenhang mit Webseiten eine Anwendung. Metadaten werden zur Identifikation (z.B. Titel oder Adresse einer Webseite) und zur Beschreibung (META-Tags*: keywords, description*) des Inhalts der Seiten benutzt. Existierende Metadaten-Formate für Webseiten erlauben eine umfassende Beschreibung der Seiten durch Metadaten. Obwohl HTML die Verwendung von Metadaten unterstützt, hat sich bisher kein Metadaten-Format durchgesetzt und sich als Standard etabliert.

2.2 Metadaten in HTML

In einer Bibliothek werden Metadaten zentral (Katalog) gehalten. Beim Eintreffen neuer Bücher wird der Katalog aktualisiert. Im World Wide Web ist diese Art der Metadatenspeicherung nicht möglich. Eine zentrale Instanz, die alle neuen Seiten indexiert, gibt es nicht. Metadaten der Webseiten werden aus diesem Grund an die jeweiligen Seiten gekoppelt abgelegt. Es gibt zwei Möglichkeiten der Speicherung:

1. Die Einbettung der Metadaten in den HTML-Code der Seite mit Hilfe des META-Elements.
2. Die Speicherung in einer externen Datei, die durch einen Verweis (Link) an die Seite gebunden ist.

2.2.1 Das META-Element

Seit der Version 2.0 [BC95] des HTML Standards ist die Angabe von Metadaten in HTML möglich. Ein Metadaten-Element wird durch das HTML-Element META dargestellt. In der HEAD-Sektion der HTML-Beschreibung der Webseite können Metadaten durch beliebig vielen META-Elementen angegeben werden.

```
<HEAD>
   ...
   <META ... >
   <META ... >
   ...
</HEAD>
```

Innerhalb des META-Elements tragen Attribut-Wert-Paare die Information. Folgende Attribute sind in HTML 4.0 [FHJ98] möglich:

- *name:* Attribut für den Namen eines Metadaten-Elementes.
- *content:* Attribut für den Wert eines Metadaten-Elementes.
- *scheme:* Dieses Attribut erleichtert die Interpretation des Wertes (*content*), indem es ein Kodierungsschema für den Wert angibt. Durch das Attribut-Wert-Paar *scheme="ISO8601"* kann zum Beispiel das Format für ein Datum angegeben werden.
- *lang:* Dieses Attribut spezifiziert die Sprache des Wertes in *ISO-639-1* [Lib02] (Zweibuchstaben-Kodierung für Sprachen in English). Dieses Attribut wird nur zusammen mit dem *content*-Attribut verwendet.
- *dir:* Dieses Attribut gibt die Leserichtung (*ltr*: "von links nach rechts" oder *rtl*: "von rechts nach links") für Texte und Tabellen an. Dieses Attribut wird nur zusammen mit dem content-Attribut verwendet.

- *http-equiv:* Dieses Attribut kann statt des *name*-Attributes verwendet werden und dient der automatischen Antwortgenerierung für HTTP-Server. Enthält eine HTML-Datei zum Beispiel das META-Tag

  ```
  <META http-equiv=content-type content="text/html”>
  ```

  dann kann der HTTP-Server, der die HTML-Datei verwaltet dieses *Tag* auslesen, und die Art des Dateiinhaltes, im Fall einer Anfrage, mitteilen. In diesem Beispiel ist der Inhaltstyp *text/html*.

Ein META-Tag für die HTML Version dieses Dokuments wäre:

```
<META name="Author" content="Mihály Sütö">
```

Das Attribut-Wert-Paar *name="Author"* bildet den Elementnamen. Das Attribut-Wert-Paar *content="Mihály Sütö"* bildet den zum Elementnamen gehörenden Wert. Durch Elementname und Wert ist ein Metadatenelement definiert. Das Hinzufügen weiterer Attribut-Wert-Paare kann die Semantik des Metadaten-Elementes verfeinern:

```
<META name="Author" lang="hu" scheme=”text/plain” dir=”ltr” content="Mihály Sütö">
```

Das *lang*-Attribut mit dem Wert "hu" gibt an, dass der Wert des *content*-Attributs in ungarischer Sprache angegeben wird. Das *scheme*-Attribut spezifiziert die Art des Inhalts; in diesem Fall handelt es sich um simplen Text. Das *dir*-Attribut gibt zusätzlich die Leserichtung des Inhalts an.

In der HTML 4.0 Spezifikation gibt es keinen vordefinierten Satz der akzeptierten Elementnamen . Die Entscheidung ob oder welche Metadaten eine Webseite verwendet, wird dem Gestalter überlassen. Folgende Elementnamen werden oft in Webseiten verwendet:

keywords

Werte für diesen Elementnamen sind die wichtigsten Schlüsselwörter, die eine Webseite charakterisieren.

description

Der Wert für diesen Elementnamen ist eine kurze Beschreibung der Webseite in natürlicher Sprache.

Die Elementnamen *keywords* und *description* konzentrieren sich nur auf die Beschreibung des Webseiteninhaltes. Weitere Eigenschaften der Webseiten werden selten durch weitere Metadaten beschrieben. Obwohl die Attribute *name* und *content* die Anwendung eines beliebigen Elementnamen-Satzes (Metadaten-Formates) erlauben, hat sich bisher kein Standard durchgesetzt.

2.2.2 Das LINK-Element

In HTML 4.0 gibt es durch das LINK-Element die Möglichkeit einen Verweis anzugeben, der auf eine, das aktuelle HTML-Dokument beschreibende, Ressource zeigt. Dadurch entsteht die Möglichkeit, Metadaten über eine Webseite anzugeben. Die Quell-Seite (der Link zeigt von dieser Seite aus), kann durch die Ziel-Seite (auf diese Seite wird gezeigt) beschrieben werden. Das LINK-Element steht in der HEAD-Sektion der HTML-Beschreibung und ist für den Betrachter der Webseite unsichtbar. Das folgende Beispiel zeigt, wie ein Inhaltsverzeichnis durch das LINK-Element angegeben werden kann.

```
<HEAD>
   <TITLE> Ortsbasierter Webzugriff </TITLE>
      <LINK rel="Contents" href="Inhaltsverzeichnis.html">
      ...
</HEAD>
```

Im Konstrukt *rel=”Contents”* kodiert die Abkürzung *rel* eine Vorwärtsreferenz (alternativ gibt es die Abkürzung *ref* für eine Rückwärtsreferenz), während *Contents* den aktuellen Linktyp angibt. Eine Vorwärtsreferenz bedeutet in diesem Zusammenhang, dass der LINK auf eine Ressource verweist, die das HTML-Dokument, das den LINK enthält, beschreibt. In diesem Fall ist diese Ressource ein weiteres HTML-Dokument, das ein Inhaltsverzeichnis enthält. Eine Rückwärtsreferenz würde auf eine Ressource zeigen, die durch das aktuelle HTML-Dokument beschrieben wird. Stammt beispielsweise die oben angegebene HEAD-Sektion aus einem HTML-Dokument namens *“Arbeit.html”*, dann könnte das Dokument *"Inhaltsverzeichnis.html"* in der HEAD-Sektion den folgenden LINK (Rückwärtsreferenz) aufweisen:

```
<LINK ref="Contents" href="Arbeit.html">
```

Nicht alle Link-Typen können zur Angabe von Metadaten verwendet werden. Link-Typen die Metadaten-Charakteristik haben, sind folgende:

- *Contents:* Dieser Linktyp zeigt auf eine Webseite mit einem Inhaltsverzeichnis.
- *Index:* Dieser Linktyp zeigt auf eine Webseite mit einem Index.
- *Glossary:* Dieser Linktyp zeigt auf eine Webseite mit einem Glossar.
- *Copyright:* Dieser Linktyp zeigt auf eine Webseite, die die Kopierschutzrechte angibt.

Dieser Weg Metadaten anzugeben benötigt mehrere Webseiten, um eine einzige zu beschreiben. Die vier Aspekte erlauben keine umfassende Beschreibung einer Ressource und der Satz der Linktypen ist nicht erweiterbar. Damit sind Linktypen für die Angabe von umfassenden Metadaten ungeeignet.

2.2.3 Das Resource Description Framework

Das *Ressource Description Framework (RDF)* [LS99] ist ein Modell für die Angabe von Metadaten über beliebige Ressourcen, die mit einem *Uniform Resource Identifier (URI)* identifiziert werden können. Es ist vom *World Wide Web Consortium (W3C)* entwickelt worden, um in erster Linie Objekte im World Wide Web zu beschreiben.

Die grundlegende Idee des RDF ist die Modellierung von Metadaten über Objekte, durch einfache Tripel (*statement*). Ein Subjekt (Ressource) wird durch ein Prädikat (Eigenschaft) und einem zugehörigen Objekt (Wert) beschrieben.

Subjekt	www.suetoe.de/mihaly/
Prädikat	Creator
Objekt	“Mihály Sütö”

Tabelle 2-1: RDF-Tripel

Die graphische Darstellung des Tripels könnte so aussehen:

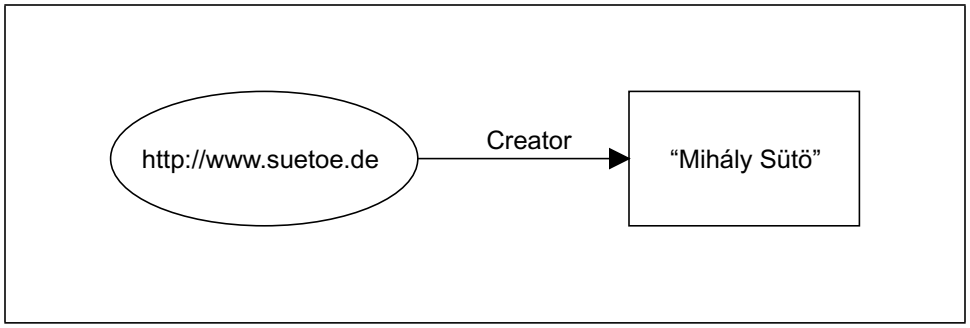

Abbildung 2-1: Einfacher RDF-Ausdruck

Das Subjekt ist typischerweise ein Objekt des World Wide Webs (z.B. eine Webseite). Das Prädikat ist eine Eigenschaft, die einen Aspekt der Ressource durch das Objekt beschreibt. Das Objekt kann ein einfacher (z.B. Text) oder ein zusammengesetzter Datentyp sein. Man spricht in diesem Zusammenhang von einem zusammengesetzten Datentyp, wenn das Objekt durch ein URI identifizierbar ist. In diesem Fall kann man es durch RDF beschreiben. Das vorherige Beispiel mit einem zusammengesetzten Objekt könnte wie in Abbildung 2 dargestellt werden.

Das Objekt, das die Webseite “www.suetoe.de” beschreibt, ist auch ein Subjekt, das durch Name und Email weiter spezifiziert wird.Sein URI ist “www.suetoe.de/mihaly/”, das eine Person beschreibt.

Das *Resource Description Framework* ist ein abstraktes Modell ohne eigene Syntax. Es benutzt das *Extensible Markup Language (XML)* und ist damit eine Anwendung dieser Sprache.

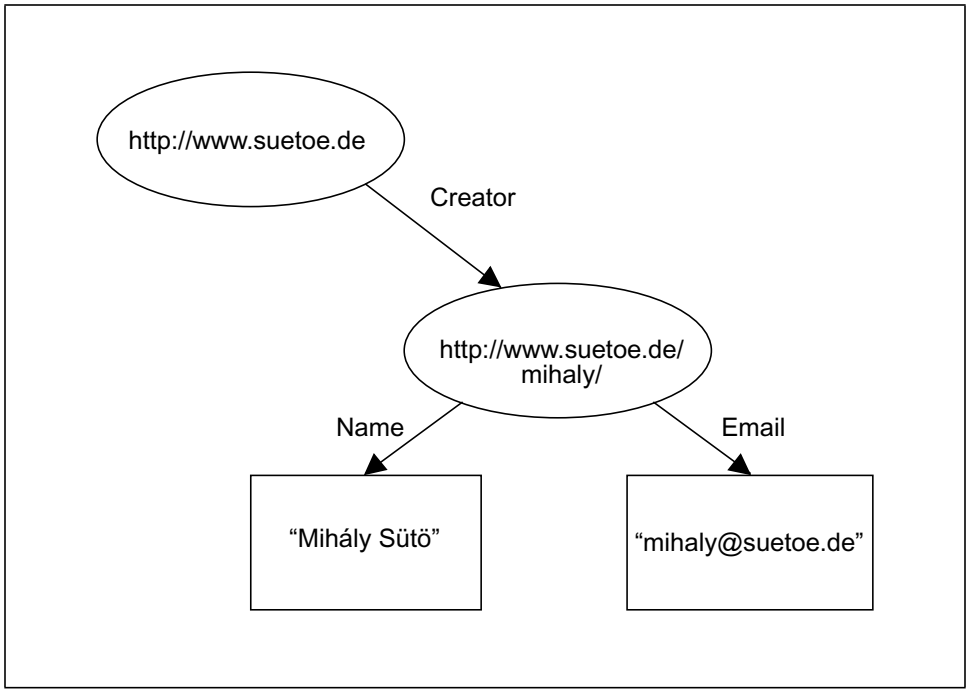

Abbildung 2-2: Zusammengesetzter RDF-Ausdruck

Zur Abbildung 2-1 passende XML / RDF-Fragment sieht folgendermaßen aus:

```
<rdf:RDF>
   <rdf:Description about="http://www.suetoe.de">
      <s:Creator>Mihály Sütö</s:Creator>
   </rdf:Description>
</rdf:RDF>
```

Zusammengesetzte Ausdrücke werden durch Verschachtelung in XML / RDF ausgedrückt. Zur Abbildung 2-2 passende XML / RDF-Fragment hat die Folgende Form:

```
<rdf:RDF>
   <rdf:Description about="http://www.suetoe.de">
      <s:Creator>
         <rdf:Description>
            <s:Name>Mihály Sütö</s:Name>
            <s:Email>mihaly@suetoe.de</s:Email>
         </rdf:Description>
      </s:Creator>
   </rdf:Description>
</rdf:RDF>
```

Ein wichtiges Konzept von XML ist die Benutzung von Namensräumen (*namespaces*). Ein Namensraum in XML ist eine Sammlung von Namen, die als Attributnamen verwendet werden [BHL99]. Solche Attributnamen sind *Description* oder *Creator*. Der Attributname *Description* stammt aus dem *rdf* Namensraum und signalisiert, dass eine Beschreibung einer Ressource oder eines Ressourcenteils erfolgt. Attributnamen wie *Creator* oder *Name* aus dem Namensraum *s*, repräsentieren verschiedene Eigenschaften. Ein vollständiges XML-Dokument muss immer die Spezifikationen aller benutzten Namensräume durch URIs angeben.

Die Definition eines Namensraumes für das RDF ist ein *RDF-Schema*. Ein Schema gibt einen Satz von Attributnamen an, definiert die Bedeutung dieser Attributnamen und gibt den Kontext an, in dem sie verwendet werden [BG00]. Das RDF Modell ist also ein Rahmen für die Modellierung von Metadaten, der erst durch die verschiedene Schemata an Semantik gewinnt.

2.3 Dublin Core Metadata Initiative

Die Möglichkeiten, die sich durch die Verwendung von Metadaten in der Informationsverarbeitung erschließen, sind in vielen Bereichen erkannt worden. In vielen wissenschaftlichen Themengebieten existieren bereits oder werden zurzeit Metadatenformate entwickelt. Ein interessantes Beispiel ist das *National Biological Information Infrastructure (NBII)* [UDI96], das ein Metadatenformat für den Austausch von biologischen Daten entwickelt, oder das *Content Standard for Digital Geospatial Metadata (CSDGM)* [FGD98], ein Metadatenformat das die Dokumentation von geographischen Daten ermöglicht.

Das META-Element in HTML und das RDF ermöglichen zwar die Benutzung von Metadaten, sie geben aber kein Metadatenformat an, das benutzt werden soll. Die Entwicklung solcher Formate wird anderen Interessengruppen bzw. Organisationen überlassen.

Die Etablierung eines Standards ist ein langwieriger Prozess. Allgemein akzeptierte und verbreitete Metadatenstandards fehlen deswegen sogar in Bereichen, in denen offensichtlich ein immenser Bedarf für sie vorhanden ist. Ein Beispiel dafür wären Ressourcen des World Wide Webs. Obwohl gerade Suchmaschinen stark von der Verwendung von Metadaten profitieren könnten, hat sich noch kein Metadatenstandard für die Beschreibung von Webseiten verbreiten können.

Ein Grund dafür ist sicherlich die einfache Möglichkeit, Metadaten zu missbrauchen. Durch die falsche Angabe von Metadaten können Suchmaschinen, die von der Richtigkeit dieser Daten ausgehen, fehlgeleitet werden. Eine genaue Untersuchung von Suchmaschinen findet in Kapitel 3 statt. Unter anderem wird auch auf die Bewertung von Metadaten durch Suchmaschinen eingegangen.

Das *Dublin Core Metadata Initiative (DCMI)* ist eine Organisation, die die Verbreitung von Metadatenstandards fördert. Das meistverwendete Metadatenformat für Ressourcen des

World Wide Webs, das *Dublin Core Metadata Element Set*, stammt von dem *DCMI* und wird bereits von vielen Organisationen und Firmen eingesetzt. Es bietet unter anderem als einziges Metadatenformat für Webseiten ein Element, das eine Angabe von Ortsinformation bezüglich der beschriebenen Ressource unterstützt.

2.3.1 Dublin Core Metadata Element Set

Die Entwicklung des *DCMES* startete am 1. März 1995 in Dublin (Ohio USA) im Rahmen eines Workshops [WGM95], das vom dem *Online Computer Library Center (OCLC)* und von dem *National Center for Supercomputing Applications (NCSA)* gestiftet wurde. Ziel des Workshops war, einen grundlegenden Satz von Metadatenelementen zu finden, der die Beschreibung von netzwerktypischen Ressourcen ermöglicht. Seit dem 2. Juli 1999 ist die Version 1.1 des *DCMES* verfügbar. Der Status des Dokuments ist eine *"DCMI Recomendation"* was einem stabilen und empfohlenen Standard-Kandidaten entspricht.

2.3.1.1 Simple Dublin Core

Das *DCMES* spezifiziert 15 Elemente, die eine präzise Beschreibung von Ressourcen des World Wide Webs erlauben. Die Verwendung der Elemente ist allerdings auch in anderen Domänen denkbar. Jedes Element trägt das *DC* Präfix, das für *Dublin Core* steht, und gewährleistet dadurch ihre Eindeutigkeit im World Wide Web. Die 15 Grundelemente werden auch *Simple Dublin Core* genannt. Die Namen der Elemente und ihre sinnvolle Verwendung sind:

DC.Title

Name der Ressource, der üblicherweise vom Verfasser, Urheber oder Verleger vergeben wird.

DC.Creator

Person(en) oder Organisation(en), die den intellektuellen Inhalt der Ressource verantworten. Autoren bei Textdokumenten, Photographen, Komponisten oder Maler bei graphischen Dokumenten bzw. Kunstwerken.

DC.Subject

Das Thema der Ressource bzw. Stichwörter oder Phrasen, die das Thema oder den Inhalt beschreiben. Die Stichwörter können aus einem vordefinierten Vokabular bzw. Klassifikationsschema stammen.

DC.Description

Eine natürlichsprachliche Beschreibung des Ressourceninhalts. Die Beschreibung soll durch eine kurze Zusammenfassung oder ein Inhaltsverzeichnis einen groben Überblick über die Ressource geben.

DC.Publisher

Die Einrichtung, die verantwortet, dass diese Ressource in dieser Form zur Verfügung steht, wie z. B. ein Verleger, ein Herausgeber oder eine Universität.

DC.Contributor

Zusätzliche Person(en) bzw. Organisation(en), die einen bedeutsamen intellektuellen Beitrag zur Ressource geleistet haben und nicht unter *DC.Creator* aufgeführt sind. Die Leistung, der unter *DC.Contributor* aufgeführten Person(en) bzw. Organisation(en) ist im Vergleich mit denen in *DC.Creator* als sekundär zu betrachten (z.B. Herausgeber, Übersetzer oder Illustratoren).

DC.Date

Das Datum, an dem die Ressource in der gegenwärtigen Form zugänglich gemacht wurde. Das empfohlene Format des Datums ist das *International Standard Date and Time Notation (ISO 8601)*. Dabei wird das Jahr vierstellig, der Monat und der Tag zweistellig notiert (JJJJ-MM-TT). Andere Darstellungsschemata für das Datum sind erlaubt.

DC.Type

Die Kategorie, in die die Ressource einzuordnen ist. Beispiele für solche Kategorien sind Homepage, Roman oder Arbeitsbericht. Die Bezeichnung der Kategorie soll aus einem vordefinierten Satz von Ressourcenarten, wie z.B. das *DCMI Type Vocabulary* [DCM00b] stammen.

DC.Format

Das datentechnische Format der Ressource, z.B. *text/html*, *Postscript*-Datei, *JPEG*-Bilddatei usw. Die Angabe in diesem Feld gibt die erforderliche Information an, die Menschen oder Maschinen benötigen, um die richtige Verarbeitungsart auszuwählen (z.B. welche Hard- und Software benötigt werden, um diese Ressource anzuzeigen bzw. auszuführen). Ähnlich wie bei der Ressourcenart (*DC.Type*) soll die Angabe in diesem Feld aus einem vordefinierten Satz von Formattypen , wie z.B. die *Internet Media Types (MIME Types)* stammen.

DC.Identifier

Eine Zeichenkette oder Zahl, die die Ressource eindeutig identifiziert. Um die Eindeutigkeit der Identifikation zu gewährleisten, soll die Angabe in diesem Feld bewährte Identifikationsmethoden, wie URIs für vernetzte Ressourcen oder die ISBN für Bücher, benutzen.

DC.Source

Die Quellressource, aus der die aktuelle Ressource stammt. Zum Beispiel könnte hier bei einer HTML-Version eines Buches die gedruckte Version als Quelle angeben werden, die als Vorlage für die Erstellung der HTML-Version benutzt wurde.

DC.Language

Die Sprache(n) des intellektuellen Inhalts dieser Ressource. Der Inhalt dieses Feldes sollte möglichst mit der zweistelligen Sprachnotation (*ISO 639*) codiert werden. Eine Erweiterung des Sprachcodes durch einen zweistelligen Ländercode (*ISO 3166*) ist möglich. Ein Beispiel wäre die Notation des amerikanischen English als *"en-US"*.

DC.Relation

Dieses Feld ermöglicht es, Verbindungen unter verschiedenen Ressourcen darzustellen, die einen Bezug zueinander haben, aber als eigenständige Ressourcen existieren. Beispiele sind Bilder in einem Dokument oder Kapitel eines Buches. Eine formale Spezifizierung (Referenzliste) für dieses Element ist in Arbeit.

DC.Coverage

Räumliche Daten (z.B. geographische Koordinaten) und zeitliche Gültigkeit, die die Ressource charakterisieren. Die Werte für dieses Feld sollen einem einheitlichen Format entsprechen bzw. aus einem festen Vokabular, wie das *Thesaurus of Geographic Names (TGN)* [Get00], stammen.

DC.Rights

Angaben über die rechtlichen Bedingungen bzw. Bezugs- und Benutzungsbedingungen bezüglich des Ressourceninhaltes. Vorgesehen für den Inhalt dieses Elementes ist ein Verweis (angegeben durch ein URI) an einem Urhebervermerk, ein "Rights-Management"-Vermerk über die rechtlichen Bedingungen. Ist dieses Feld leer, sollten keine falsche Schlussfolgerungen bezüglich der oben genannten Rechten gezogen werden.

2.3.1.2 Qualified Dublin Core

Dei 15 Elemente des *DCMES* erlauben bereits eine umfassende Beschreibung einer Ressource. Es existieren trotzdem Anwendungsbereiche in denen einige Aspekte der Ressourcen eine genauere Angabe benötigen. Außerdem sind die Datentypen bzw. Formate, denen die Werte für die einzelnen Elementnamen entsprechen sollen, wenig spezifiziert.

Die *Dublin Core Qualifiers* dienen zweierlei Zwecken. Sie können eine Verfeinerung der Grundelemente realisieren oder verschiedene Kodierungsschemata für die Elementwerte angeben. *Qualifiers*, die der Verfeinerung dienen, schränken den ursprünglichen Verwendungsbereich des jeweiligen Elementes ein und geben damit eine genauere

Semantik an. *Qualifiers*, die ein Codierungsschema angeben, erleichtern dagegen die Interpretation des Elementwertes.

Ein “qualifiziertes” *Dublin Core Element* wird durch die Erweiterung des ursprünglichen Elementnamen angegeben, indem das *Qualifier* an den Grundnamen angehängt wird. Ein Beispiel dafür wäre die Verfeinerung von *DC.Title* durch *DC.Title.Alternative*.

Nachfolgend werden Qualifiers nur für diejenigen Elemente angegeben, die bei der Modellierung der Webseiten im NEXUS Projekt eine Rolle spielen.

DC.Title

Verfeinerung:

Alternative

Ein alternativer Titel für die Ressource. Dies kann zum Beispiel eine Abkürzung oder eine Übersetzung des Originaltitels sein.

DC.Creator

Es gibt bisher keine Verfeinerungen oder Kodierungsschemata für dieses Element.

DC.Subject

Kodierungsschemata:

LCSH

Das *Library of Congress Subject Headings (LCSH)* ist eine Sammlung von Stichwörtern, die Themengebiete beschreiben. Dieser Stichwortkatalog wird von vielen Bibliotheken weltweit als Standard benutzt.

MESH

Das *Medical Subject Headings (MeSH)* [Nel00] ist eine Sammlung von Stichwörtern, die medizinische Themengebiete beschreiben.

DDC

Das *Dewey Decimal Classification (DDC)* [For02] ist ein hierarchisches Klassifikationssystem für Themengebiete. Die einzelnen Klassen werden durch dreistellige arabische Zahlen mit eventuellen Nachkommastellen notiert. Die erste Zahl steht für eine Hauptklasse, die zweite für eine Division und die dritte für eine Sektion. Zum Beispiel steht *500* für Wissenschaften, *530* für Physik (als eine Division der Wissenschaften) und *531* für Mechanik (als eine Sektion der Physik).

LCC

Das *Library of Congress Classification (LCC)* [Lib01] ist ein dem *DCC* ähnliches Klassifikationssystem. Die ersten beiden Stellen der Notation sind Buchstaben und kodieren hierarchisch verschiedene Themengebiete. Die restlichen Stellen sind Zahlen, die ein Themengebiet weiter linear gliedern. Zum Beispiel steht *Q* für Wissenschaften, *QC* für Physik und *QC120-168.85* für den Teilbereich Mechanik.

UDC

Das *Universal Decimal Classification (UDC)* [UDC01] ist eine Erweiterung vom *DDC*. Es ermöglicht zusätzlich die Definition von Relationen zwischen den einzelnen Klassen.

DC.Description

Verfeinerungen:

TableOfContents

Diese Verfeinerung gibt ein Inhaltsverzeichnis für die gegebene Ressource an. Ein Objekt (Textfile, Webseite, ...), das das Inhaltsverzeichnis enthält, kann durch ein URI angegeben werden.

Abstract

Eine kurze Zusammenfassung des Inhalts, der Ressource.

DC.Publisher

Es gibt bisher keine Verfeinerungen oder Kodierungsschemata für dieses Element.

DC.Date

Verfeinerungen:

Created

Das Erstellungsdatum der Ressource.

Valid

Datum, bis zu dem die gegebene Ressource als gültig anzusehen ist. Alternativ kann auch ein Zeitrahmen spezifiziert werden.

Available

Datum, seit dem die Ressource verfügbar ist.

Issued

Datum, an dem die Ressource erschienen ist bzw. publiziert wurde.

Modified

Datum, an dem die Ressource das letzte Mal modifiziert wurde.

Kodierungsschemata:

Period

Dieses Schema ermöglicht die Spezifikation eines Zeitraums, der durch vier Komponenten dargestellt wird. Die Komponente *name* gibt einen Namen für die Periode an, die Komponenten *start* und *end* geben den Beginn und das Ende der Periode an und *scheme* spezifiziert ein Datenformat für *start* und *end*. Das Standardformat für die Darstellung eines Datums für *Period* ist *W3CDTF*.

W3CDTF

Dieses Schema ist das *W3C Date and Time Format* [WW98]. Es benutzt nur ein Teil des *ISO 8601* Standards um Fehleranfälligkeit zu vermeiden. Ein Beispieldatum wäre: 2002-04-30T9:20:30+01:00 für den 30. April 2002 9 Uhr 20 Minuten und 30 Sekunden, plus eine Stunde zur GMT.

DC.Identifier

Kodierungsschema:

URI

Dieses Schema ist das *Uniform Resource Identifier* Schema [BFI98].

DC.Coverage

Verfeinerungen:

Spatial

Räumliche Daten, die den Inhalt der Ressource charakterisieren. Beschreibt die Ressource zum Beispiel ein Gebäude, dann kann *Spatial* die Position des Gebäudes angeben.

Temporal

Daten über einen Zeitraum, der den Inhalt der Ressource charakterisiert. Berichtet eine Webseite zum Beispiel über eine Epoche der Weltgeschichte, dann kann man hier den zugehörigen Zeitraum angeben.

Kodierungsschemata für Spatial:

DCMI Point

Dieses Schema ermöglicht die Angabe eines Punktes im Raum durch geographische Koordinaten. Die drei Hauptkomponenten sind *east*, *north* und *elevation*. Die Komponenten *east* und *north* spezifizieren einen Punkt im zweidimensionalen Raum, *elevation* steht für die Höhe, also für die dritte Dimension. Alle drei Komponenten werden in Zahlen angegeben. Drei weitere Hilfskomponenten spezifizieren diese bisher "relative" Angabe. Die Komponente *units* steht für die Einheit (z.B. Grad) von *east* und *north*, *zunits* für die Einheit (z.B. Meter) von *elevation* und *projection* für die Einordnung in ein existierendes geographisches Koordinatensystem (z.B. Grandnetz der Erde). Zusätzlich kann durch *name* ein Name für die Position angegeben werden.

ISO 3166

Dieses Schema ist der internationale Standard für die Kodierung von Ländernamen durch zwei Buchstaben. Zum Beispiel steht *"DE"* für Deutschland und *"HU"* für Ungarn.

DCMI Box

Dieses Schema ermöglicht die Angabe einer Region, durch das kleinste umschließende Rechteck (*box*). Die Komponenten *northlimit*, *eastlimit*, *southlimit* und *westlimit* stehen für die jeweiligen Grenzen des Rechtecks. Die Komponenten *uplimit* und *downlimit* stehen für die dritte Dimension. Durch *units* (Standard: Grad) und *zunits* (Standard: Meter) werden, wie beim *DCMI Point*, die Einheiten für die Grenzen angegeben. Durch *projection* gewinnen die Angaben den Bezug zu einem existierenden geographischen Koordinatensystem. Ein Name kann zusätzlich durch *name* Spezifiziert werden.

TGN

Dieses Schema ist das *The Getty Thesaurus of Geographic Names (TGN)* [Get00], das wie der Name schon sagt, ein Katalog von geographischen Namen ist. Die über eine Million Einträge dokumentieren geographische Lokationen durch die Angabe des Namens, der Koordinaten und durch eine hierarchische Anordnung. Der Katalog kann durch einen speziellen Browser angeschaut werden, der die hierarchische Anordnung anschaulich darstellt. Das *TGN* ist außerdem in verschiedenen Formaten für eine maschinelle Verarbeitung zu erwerben.

Kodierungsschemata für Temporal:

Für die Verfeinerung *Temporal* des *DC.Coverage* Elements existieren das *DCMI Period* und das *W3C-DTF* Schemata. Sie wurden bereits bei dem Element *DC.Date* betrachtet.

2.3.2 Kodierung des DCMES in HTML

Wie in Kapitel 2.3.1 gezeigt wurde, kann man mit Hilfe der META-Tags Metadaten in HTML einfach darstellen. Das *DCMES* ist ein Satz von Elementnamen, der sich hervorragend für diesen Zweck eignet. Bei der Verwendung des *DCMES* kann man sich auf die Grundelemente (*Simple Dublin Core*) beschränken oder die verschiedene Verfeinerungen (*Qualified Dublin Core*) mitbenutzen.

2.3.2.1 Simple Dublin Core in HTML

Die Beschreibung von Metadaten mit *Simple Dublin Core* beschränkt sich auf die 15 Grundelemente ohne die Verfeinerungen mitzubenutzen. Dem Attribut *name* des META-Elementes wird ein *Dublin Core* Elementname zugeordnet. Das Attribut *content* erhält einen zum Elementnamen gehörenden Wert. Die generelle Syntax sieht folgendermaßen aus:

```
<META name=”Präfix.Elementname” content=”Wert”>
```

Mit “Präfix” spezifiziert man den entsprechenden Elementsatz. In diesem Fall ist es *DC* für *Dublin Core*. “Elementname” steht für ein beliebiges *Dublin Core* Grundelement, wie zum Beispiel *Creator*. Beispiel META-Tags, die diese Studienarbeit beschreiben, könnten so aussehen:

```
<META name="DC.Title" content="Ortsbasierter Webzugriff">
<META name="DC.Creator" content="Mihály Sütö">
<META name="DC.Date" content="2002-04-30">
```

Andere *Dublin Core* Grundelemente werden analog verwendet. Bei der Beschreibung der Ressource sind alle Elemente optional und jedes Element darf beliebig oft vorkommen.

Man kann und soll das LINK-Element in HTML benutzen, um die Spezifikationen der verwendeten Elementsätze anzugeben. Für das *DCMES* geschieht das indem man das URI der Spezifikation in der HEAD-Sektion der HTML-Beschreibung angibt:

```
<LINK rel="schema.DC" href= "http://purl.org/dc/elements/1.1/">
```

Die Verwendung von anderen Elementsätzen zusammen mit dem *DCMES* ist unproblematisch. Die konsequente Verwendung der Präfixe der einzelnen Elementnamen und

die Angabe der Elementsatzspezifikationen mit dem LINK-Element muss sichergestellt werden.

2.3.2.2 Qualified Dublin Core in HTML

Die Beschreibung von Metadaten mit *Qualified Dublin Core* verwendet die 15 durch entsprechende Verfeinerungen erweiterte Grundelemente. Schemata können zusätzlich, zur leichteren Interpretation der Werte angegeben werden. Dem Attribut *name* des META-Elementes wird ein *Qualified Dublin Core* Elementname zugeordnet. Das Attribut *content* erhält einen zum Elementnamen gehörenden Wert. Das Attribut *schema* kann zusätzlich ein Schema definieren. Die generelle Syntax sieht folgendermaßen aus:

```
<META name=”Präfix.Elementname.Verfeinerung” content=”Wert”
      schema=”Schema”>
```

“Präfix” spezifiziert wieder den Elementsatz (*DC*). “Elementname” steht für ein beliebiges *Dublin Core* Grundelement, wie zum Beispiel *Date*. “Verfeinerung” gibt eine Verfeinerung des Grundelementes, wie zum Beispiel *Available* an. Ein Beispiel META-Tag, das ein Datum für die Verfügbarkeit dieser Arbeit angibt, sieht damit so aus:

```
<META name="DC.Date.Available" content="2002-04-30" schema=”W3CDTF”>
```

Andere *Dublin Core* Verfeinerungen werden analog verwendet. Bei der Beschreibung der Ressource sind alle Elemente optional und jedes Element darf beliebig oft vorkommen, wobei ein Mehrfachauftreten mancher Elemente sinneswidrig ist.

Das LINK-Element dient wieder zur Angabe des verwendeten Elementsatzes. In diesem Fall ist es das *Qualified Dublin Core* Schema:

```
<LINK rel="schema.DC" href="http://dublincore.org/qdcmes/1.0/">
```

Im Anhang A befindet sich eine HTML-Datei, die weitere Beispiele zu *Simple* und *Qualified Dublin Core* Metadaten in META-Tags enthält.

2.3.3 Kodierung des DCMES in RDF

Die grundsätzliche Möglichkeit, Metadaten mit XML/RDF zu modellieren, wurde bereits in Kapitel 2.3.3 gezeigt. Um *Simple Dublin Core* mit XML/RDF auszudrücken, kann man einfach den *DC* Namensraum für die Darstellung der einzelnen Aspekte der Ressourcen benutzen. Die Modellierung von Ressourcen mit *Qualified Dublin Core* gestaltet sich jedoch komplizierter. Durch die Möglichkeit der Verschachtelung von Ausdrücken in RDF, kann man erheblich komplexere Zusammenhänge ausdrücken als es mit META-Tags möglich ist.

Bei der Verwendung von XML/RDF für die Beschreibung von Webseiten fasst man grundsätzlich alle Metadaten in einen einzigen XML-Dokument zusammen, das in eine separate Datei abgelegt wird. Durch das LINK-Element wird ein Verweis auf diese Datei in der HEAD-Sektion eines HTML Dokumentes folgendermaßen angegeben (z.B. für diese Studienarbeit):

```
<LINK rel="meta" href="studienarbeit.html.rdf">
```

Der Verweis hat den Linktyp “meta”, der darauf hinweist, dass dieser Verweis auf Metadaten zeigt.

2.3.3.1 Simple Dublin Core in XML/RDF

Die Darstellung von Metadaten mit Simple Dublin Core und XML/RDF ähnelt der linearen Darstellung von Metadaten mit META-Tags. Eigenschaften, die eine Ressource beschreiben, werden linear aufgelistet. Eine Ressource beschrieben durch mehrere Eigenschaften könnte graphisch, wie in Abbildung 2-3 zu sehen ist, dargestellt werden.

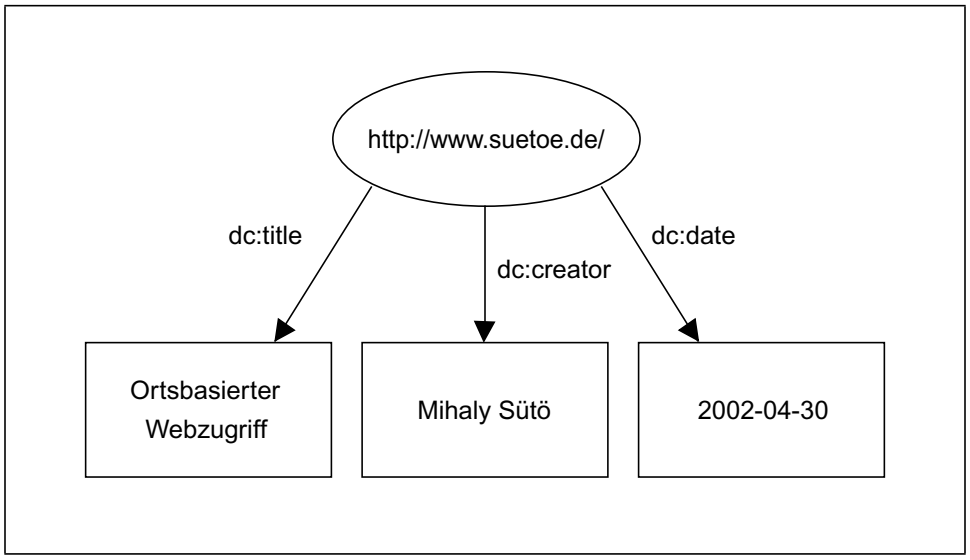

Abbildung 2-3: Simple Dublin Core in RDF

Das XML/RDF-Dokument, das die Zusammenhänge aus Abbildung 2-3 beschreibt, sieht folgendermaßen aus:

```
<?xml version="1.0"?>
  <rdf:RDF
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:dc="http://purl.org/dc/elements/1.1/">
     <rdf:Description rdf:about="http://www.suetoe.de/">
        <dc:title> Ortsbasierter Webzugriff </dc:title>
        <dc:creator> Mihály Sütö</dc:creator>
        <dc:date> 2002-04-30 </dc:date>
     </rdf:Description>
  </rdf:RDF>
```

In der ersten Zeile steht, welche XML-Version das Dokument verwendet. Zeilen drei und vier geben die Spezifikationen von RDF bzw. *Dublin Core* an. Wie bereits erwähnt, sollten alle Namensräume, die benutzt werden, am Anfang des jeweiligen XML-Dokumentes angegeben werden. Im *Description* Block werden die zu modellierende Eigenschaften linear aufgelistet. Ähnlich wie bei META-Tags, dürfen *Dublin Core* Elemente beliebig oft in beliebiger Reihenfolge vorkommen.

2.3.3.2 Qualified Dublin Core mit XML/RDF

Ein Modell für die Darstellung von Metadaten mit Qualified Dublin Core in XML/RDF befindet sich noch im Entwicklungsstadium. Es gibt eine Beschreibung in Form eines *"Working Draft"* vom *DCMI*, allerdings sind die verwendeten Namensräume noch nicht fertig entwickelt. Es gibt zurzeit weder eine offizielle Empfehlung noch offizielle Beispiele für die Implementierung.

In dem RDF kann ein Subjekt durch ein zusammengesetztes Objekt beschrieben werden. Auch ein Prädikat, das mit einem Objekt als Wert ein Subjekt beschreibt, kann zusammengesetzt sein. Diese Eigenschaft wird bei der Modellierung von *Qualified Dublin Core* in XML/RDF ausgenutzt. Die Verfeinerung eines *Dublin Core* Grundelementes mit einem zusammengesetzten Prädikat und die Angabe eines Schemas für die Wertemenge könnte graphisch dargestellt wie in Abbildung 2-4 aussehen.

Bei diesem Beispiel werden zwei Namensräume verwendet: die Grundelemente (*Dublin Core = dc*) und die Verfeinerungen (*Dublin Core Qualifiers = qdc*). Hier wird das *DC.Date* Grundelement mit der Verfeinerung *Available* und dem Schema *W3CDTF* spezifiziert. Die Semantik des Ausdruckes ist, dass die Ressource unter "http://www.suetoe.de" seit dem 30. April 2002 verfügbar ist. Das Datum hat das *W3CDTF* Format.

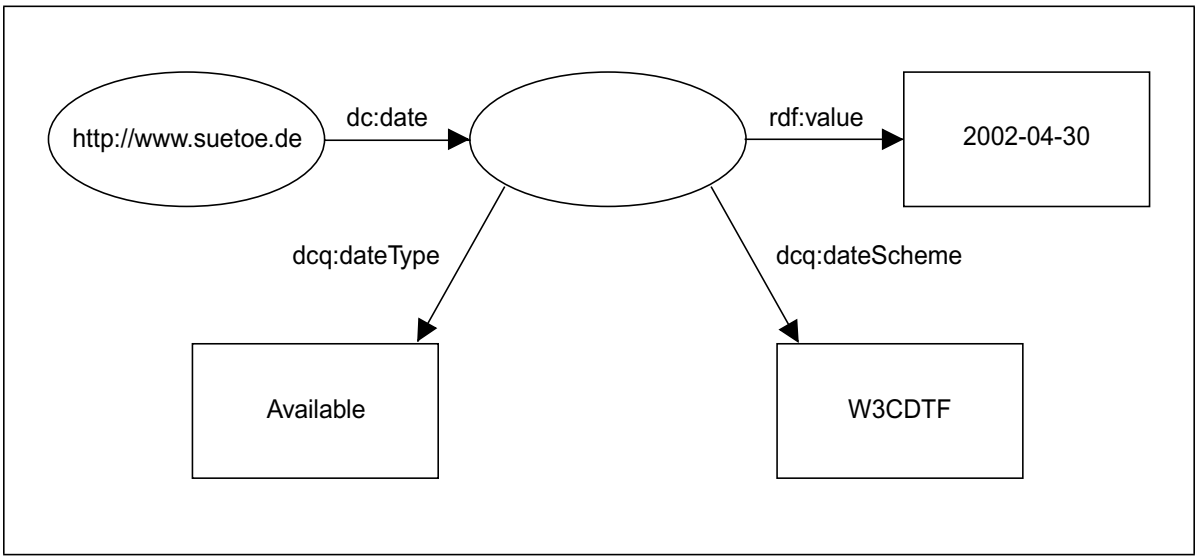

Abbildung 2-4: Qualified Dublin Core in RDF

Die Zusammenhänge in Abbildung 4 werden durch das folgende XML-Dokument beschrieben:

```
<?xml version='1.0'?>
   <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:dc = "http://purl.org/dc/elements/1.1/"
         xmlns:dcq = "http://purl.org/dc/qualifiers/1.0/">
      <rdf:Description rdf:about = "http://www.suetoe.de">
         <dc:date>
            <rdf:Description>
               <rdf:value> 2002-04-30 </rdf:value>
               <dcq:dateType> Available </dcq:dateType>
               <dcq:dateScheme> W3CDTF </dcq:dateScheme>
            </rdf:Description>
         </dc:date>
      </rdf:Description>
   </rdf:RDF>
```

Das zusammengesetzte Prädikat *date* besteht aus einem eigenen *Description*-Block, der einen Wert (*2000-04-30*), eine Verfeinerung (*Available*) und ein Schema (*W3CDTF*) angibt. Ein dritter Namensraum für Terme (*Dublin Core Terms = dct*), der die Verfeinerungen und die Schemata beschreibt, kann zusätzlich verwendet werden. Dies wird nötig, wenn eine Verfeinerung oder ein Schema nicht durch ein Literal, sondern durch ein URI oder einen weiteren zusammengesetzten Ausdruck angegeben werden.

2.4 Bewertung

Das Hauptziel dieser Studienarbeit ist die Einbindung von Webseiten in die NEXUS Infrastruktur. Mobile Anwendungen sollen von ortsbezogener Information in Form von Webseiten profitieren. Dabei ist es wichtig, in welcher Form Ortsinformation in Webseiten gespeichert ist und welche Formen bei der Einbindung eine Rolle spielen sollten.

Wie in diesem Kapitel gezeigt wurde, gibt es verschiedene Möglichkeiten der Metadatenspeicherung. Metatags sind grundsätzlich für die Darstellung von einfacher Metainformation in Form von Literalen geeignet. Sollen die beschreibenden Daten komplexer strukturierbar sein, dann ist das RDF für die Modellierung die bessere Alternative.

Obwohl Metadaten für die Informationssuche im World Wide Web sehr wichtig sind, werden Metatags nur selten umfassend eingesetzt. Die Elementnamen *keywords* und *description* sind zwar weit verbreitet, sie sagen aber wenig über eine Ressource aus. Das RDF wird noch seltener als META-Tags für die Angabe von Metadaten im World Wide Web benutzt. Die aufwendige Syntax und der zusätzliche Aufwand schreckt viele Webdesigner von der Verwendung ab. Kostenlose Tools zur Erstellung von RDF-Beschreibungen existieren zwar, doch eine verbreitete Verwendung wird nur dann stattfinden, wenn populäre HTML-Editoren die Erstellung von RDF-Metadaten unterstützen.

Für die Einbindung von Webseiten in die NEXUS Infrastruktur wird ein Webroboter, der in Kapitel 4 detailliert beschrieben wird, eingesetzt. Bei der Suche nach relevanten Webseiten soll er sowohl META-Tags, als auch eine eventuelle RDF-Beschreibung von Webseiten analysieren.

META-Tags und das RDF sind nur ein Rahmen für die Metadatenangabe. Metadatenformate, wie das *Dublin Core* werden zusätzlich zur Modellierung benötigt. Das *Dublin Core* ist das einzige Format, das ein Element für die Angabe von Ortsinformation bereitstellt und relativ verbreitet ist. Zahlreiche Regierungen und Firmen benutzen und empfehlen das *Dublin Core* Metadatenformat. Namhafte Beispiele sind die australische Regierung, die eine geringfügig erweiterte Version (*Australian Government Locator Service*) des *Dublin Core* für sämtliche australische Behörden empfiehlt, oder Firmen wie IBM, die das *Dublin Core* in ihren Webseiten teilweise benutzen. Der oben erwähnte Webroboter unterstützt deswegen das *Dublin Core* Format sowohl in META-Tags als auch in einer RDF-Beschreibung.

KAPITEL 3 SUCHMASCHINEN

In diesem Kapitel werden verschiedene Suchmaschinen auf ihre Eignung hin untersucht, Webseiten mit Ortsinformation aufzuspüren.

3.1 Einleitung

Im vorangegangenen Kapitel wurden Möglichkeiten der Metadatendarstellung für Webseiten ausführlich dargestellt. Welche Metadatenelemente oft für die Beschreibung von Webseiten verwendet werden, lässt sich per Entnahme von Stichproben aus dem World Wide Web nicht feststellen. Die manuelle Suche nach Webseiten mit speziellen Metadaten ist hoffnungslos; diese Arbeit müssen Suchwerkzeuge erledigen. Die meistverwendete Methode Webseiten im World Wide Web mit gewünschter Information zu finden, ist die Benutzung einer Suchmaschine oder eines Kataloges. Webseiten, die in Katalogen zu finden sind, werden von Menschen auf den Inhalt analysiert und in Kategorien eingeordnet. Eine Anfrage durchsucht nur die Kategorienamen nach den gesuchten Begriffen, was die gezielte Suche nach Metadaten unmöglich macht. Es stellt sich also die Frage, ob Suchmaschinen Metadaten bei der Analyse von Webseiten untersuchen und ob eine gezielte Suche nach bestimmten Metadaten möglich ist.

3.2 Generelle Funktionsweise

An Hand von *Google* [BP98] werden die wichtigsten Komponenten einer Suchmaschine und ihr Zusammenspiel in diesem Abschnitt vorgestellt. Abbildung 3-1 zeigt eine vereinfachte Darstellung der Komponenten dieser Suchmaschine.

Zu besuchende Webseiten werden von *Google* in einem so genannten *URL-Server* gespeichert. Die *Crawler* haben die Aufgabe, URLs aus dem *URL-Server* zu holen, die zugehörige Webseiten aus dem World Wide Web herunterzuladen und mittels dem *Storage-Server* abzuspeichern. In der Regel werden mehrere *Crawler* gleichzeitig verwendet, um möglichst viele Webseiten abspeichern zu können. Der *Storage-Server* legt die Webseiten komprimiert ab, um Speicherplatz zu sparen. Der *Indexer* ist das Herzstück der Suchmaschine und übernimmt gleich mehrere Aufgaben. Er holt nacheinander die heruntergeladenen Webseiten vom *Storage-Server*, extrahiert alle Verweise und übergibt sie dem *URL-Server*. Diese URLs werden später vom *Crawlern* besucht. Die Hauptfunktion des *Indexer* ist die Analyse der Seiteninhalte. Zu jeder Seite werden so genannte *hits* erzeugt, die sowohl die Vorkommenshäufigkeit als auch die Position von Wörtern, die in den Webseiten auftreten, speichern. Diese *hits* werden, nach *docIDs* sortiert, in dem *Hit-Storage* abgelegt. Ein *docID* ist die eindeutige Identifikation einer Webseite im System. Das *Lexicon* enthält einen *Index*, der ein schnelles Auffinden von Webseiten, die ein bestimmtes Wort enthalten, ermöglicht. Der *Searcher* ist die Suchkomponente des Systems, die unter Verwendung des *Hit-Storage* und des *Lexicon* Benutzeranfragen beantwortet.

Die genaue Funktionsweise anderer Suchmaschinen kann von diesem Beispiel selbstverständlich abweichen. Welche Technik die *Crawler* bei der Durchwanderung des World Wide Webs anwenden oder nach welchen Kriterien die *Indexer* Webseiten untersuchen, fällt jeweils unterschiedlich aus. Die große Anzahl und die Vielfalt von Suchmaschinen

ermöglichen dem Benutzer, eine Suchmaschine auszuwählen, die den jeweiligen persönlichen Anforderungen entspricht.

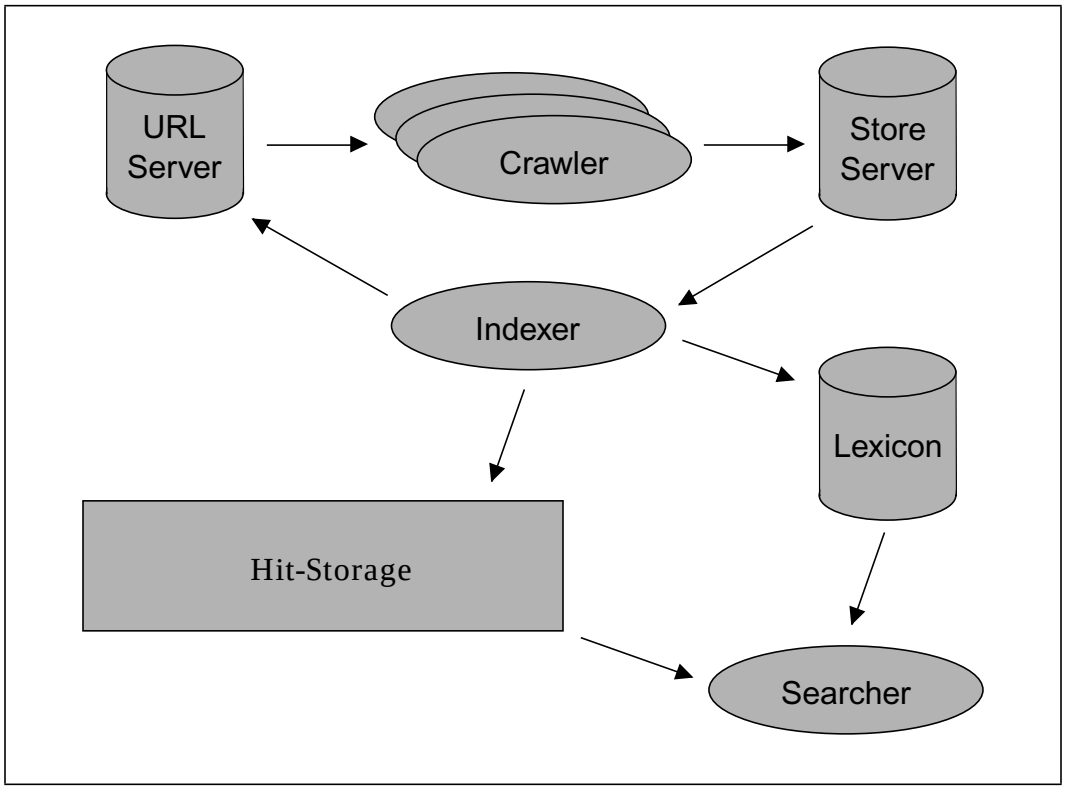

Abbildung 3-1: Vereinfachte Darstellung von Google

Ob eine Suchmaschine für das Auffinden von Webseiten mit speziellen Metadaten geeignet ist, lässt sich an Hand der näheren Betrachtung der *Indexer*- und der *Searcher*-Komponenten feststellen. Soll eine Suchmaschine in der Lage sein, nach speziellen Metadaten zu suchen, dann muss sie eventuell vorhandene META-Tags oder die RDF-Beschreibung der jeweiligen Webseite analysieren. Ist dies der Fall, muss der *Searcher* zusätzlich eine Möglichkeit bieten, genau nach diesen Metadaten zu fragen. Ob gängige Suchmaschinen diese Optionen bieten, wird an Hand der verwendeten Anfragesprache und der Indexierungstechnik drei großer Suchmaschinen in den folgenden Abschnitten untersucht.

3.3 Anfragesprache

Suchmaschinen stellen verschiedene Möglichkeiten bereit, eine Suche und die Darstellung der Ergebnisse zu gestalten. Die Effektivität der Suche hängt nicht nur von dem Crawling- oder Indexierungsverhalten der jeweiligen Suchmaschine ab, sondern auch von der Anfragemächtigkeit, die angeboten wird.

Eine Anfrage wird durch einen Suchstring spezifiziert, den die jeweilige Suchmaschine analysiert und in die Suche umsetzt. Die Möglichkeit, den Suchstring entsprechend viel-

fältig zu gestalten, ergibt die Anfragemächtigkeit einer Suchmaschine. Die Anfragesprache und damit die Anfragemächtigkeit variiert unter den Suchmaschinen; allerdings haben die meisten einen gemeinsamen Satz von Grundoptionen wobei die Anfragesyntax unterschiedlich sein kann.

3.3.1 Grundoptionen

Den folgenden Satz von Optionen bieten die meisten Suchmaschinen an.

Begriffsuche

Beispiel-String: *Hund*

Als Ergebnis werden Webseiten geliefert die das gesuchte Wort, in diesem Fall das Wort “Hund”, enthalten.

Verknüpfte Suche mit mehreren Begriffen.

Beispiel-String: *+Hund +Katze*

Als Ergebnis werden nur Webseiten geliefert, die alle gesuchten Begriffe enthalten. In diesem Fall die Wörter “Hund” und “Katze”.

Ausschließende Suche mit mehreren Begriffen.

Beispiel-String: *+Hund -Katze*

Mit dem *Minus*-Zeichen können Begriffe angegeben werden, die in der gesuchten Seite nicht vorkommen sollen. In diesem Fall werden nur Webseiten, die das Wort “Hund” nicht aber das Wort “Katze” enthalten, geliefert.

Phrasen-Suche

Beispiel-String: *“der Hund jagt die Katze”*

Es wird nach dem gesamten Ausdruck gesucht. Es werden nur Webseiten geliefert, die den gesamten Ausdruck enthalten.

3.3.2 Boolsche Ausdrücke

Für die meisten Suchabsichten sind die bisher vorgestellten Anfragemöglichkeiten völlig ausreichend. Viele Suchmaschinen bieten trotzdem zusätzliche Optionen an, um dem Benutzer eine genauere Anfrage zu ermöglichen. Dazu gehört die Verwendung von folgenden boolschen Operatoren:

AND-Verknüpfung

Beispiel-String: *Hund* AND *Katze*

Die logische UND-Verknüpfung entspricht der bereits vorgestellten verknüpften Suche und wird nur der Vollständigkeit halber aufgeführt.

OR-Verknüpfung

Beispiel-String: *Hund* OR *Katze*

Eine Anfrage mit der logischen ODER-Verknüpfung liefert Webseiten, die nur einen der angegebenen Begriffe enthalten. In diesem Fall wären es Seiten die entweder das Wort “Hund“ oder das Wort “Katze” enthalten.

NOT Präfix

Beispiel-String: *Hund* NOT *Katze*

Das logische NICHT-Präfix entspricht dem *Minus*-Zeichen der ausschließenden Suche und wird wegen der Vollständigkeit halber aufgeführt. In diesem Fall werden nur Webseiten zurückgeliefert, die das Wort “Hund” nicht aber das Wort “Katze” enthalten.

NEAR-Verknüpfung

Beispiel-String: *Hund* NEAR *Katze*

Durch die Verknüpfung von Begriffen mit NEAR wird die Suchmaschine angewiesen, nur nach Webseiten zu suchen, die die angegebenen Begriffe in enger Nachbarschaft enthält. Angesicht der Tatsache, dass zu findende Begriffe von den meisten Suchmaschinen im engen Kontext gesucht werden, ist die Verwendung von NEAR nur selten notwendig. Dies kann der Fall sein, wenn eine genaue Entfernung der gesuchten Wörter erwünscht ist, und die Suchmaschine entweder einen festen oder einen konfigurierbaren Wert für die Distanz verwendet.

Verschachtelung

Die Kombination der einzelnen logischen Operatoren ermöglicht eine sehr genaue Angabe der gesuchten Wörter. So kann man zum Beispiel nach Webseiten suchen, die das Wort “Hund” oder das Wort “Katze” nicht aber das Wort “Maus” enthalten. Zusammen gehörende Teilausdrücke werden geklammert.

Anfrage-String: (*Hund* OR *Katze*) NOT *Maus*

3.3.3 Erweiterte Suchoptionen

Die bisher vorgestellten Suchfunktionen ermöglichen nur Anfragen nach den Inhalt einer Webseite. Andere Merkmale, die durchaus aussagekräftig sind, wurden noch nicht betrachtet. Viele Suchmaschinen untersuchen nicht nur den Inhalt einer Seite sondern auch den Kontext, in dem sie auftritt. Der Titel einer Webseite, die Domain ihres URLs, der Inhalt benachbarter Seiten, ein- oder ausgehende Verweise oder der Text dieser Verweise sind viele weitere Aspekte, die wichtige Information über eine Webseite tragen können. Nachfolgend werden weitere Anfrageoptionen vorgestellt, die auch diese Aspekte berücksichtigen.

Suche im URL

Beispiel-String: url: *Ausdruck*

Diese Option bietet die Suche nach Begriffen in dem URL einer Webseite. URLs können durchaus Information über eine Seite enthalten, sind aber in vielen Fällen nicht aussagekräftig. Die Suche im URL kann trotzdem sehr nützlich sein, wenn man gezielt nach einem URL sucht, von dem nur ein Fragment bekannt ist.

Suche im Titel

Beispiel-String: title: *Ausdruck*

Jede Webseite sollte einen kurzen und aussagekräftigen Titel besitzen. Diese Suchoption ermöglicht die Suche von Webseiten, die bestimmte Begriffe in ihrem Titel enthalten.

Suche im Verweistext

Beispiel-String: anchor: *Ausdruck*

Auch Verweise, die von einer Webseite ausgehen, können Rückschlüsse auf den Inhalt der Seite ermöglichen. Diese Option bietet die Suche nach Webseiten, die die gesuchten Begriffe in ausgehenden Verweisen enthält.

Suche in der Domain

Beispiel-String: domain: *Domainname*

Wie der Name schon vermuten lässt, erlaubt diese Funktion die Suche von Webseiten, die unter einer bestimmten Domain zu finden sind. Dies ist nützlich, wenn der Benutzer die Domain der gesuchten Webseite kennt oder die Webseite unter einer bestimmten Domain vermutet.

Verweisanalyse

Beispiel-String: link: *URL*

Diese Funktion bietet nicht die Möglichkeit, nach einem bestimmten Begriff zu suchen, sondern betrachtet die Verweisstruktur des World Wide Webs als Informationsträger. Sie erlaubt Webseiten zu finden, die einen Verweis auf eine bestimmte Webseite enthalten. Der Verweistext spielt dabei keine Rolle. Möchte der Benutzer zum Beispiel Webseiten, die durch eine Hyperlink auf www.starwars.com verweisen, finden, dann müsste man den folgenden Anfrage-String angeben:

link: http://www.starwars.com

Die vorgestellten Suchoptionen werden von den meisten großen Suchmaschinen wie *Altavista* oder *Google* angeboten. Die Syntax der einzelnen Optionen ist suchmaschinenabhängig. Es gibt noch weitere Anfrageoptionen, wie zum Beispiel die Suche nach ähnlichen Webseiten oder eine Datumsbeschränkung gesuchter Webseiten, deren Auflistung hier nicht zweckmäßig ist. Besonders Suchmaschinen, die das World Wide Web auch nach nicht-HTML Dateien (Musik, Videos, Bilder, usw.) durchsuchen, können viele weitere sinnvolle Anfrageoptionen, wie beispielsweise die Beschränkung der Dateigröße, anbieten.

Die meisten Suchmaschinen bieten nicht nur eine spartanische Anfragesprache, sondern auch eine übersichtliche graphische Oberfläche, mit der man die aufgelisteten Suchoptionen bequem und intuitiv verwenden kann. Die Darstellung der Ergebnismenge ist oft individuell, unter der Verwendung von zusätzlichen Filtern, konfigurierbar.

Für das Auffinden von Webseiten mit Metadaten ist weder die Art der Anfragestellung (graphische Oberfläche oder komplexer Anfragesyntax) noch die Gestaltung der Ergebnismenge relevant. Wie die Untersuchung von Anfrageoptionen ergab, gibt es bei den “Standardsuchmaschinen” keine Möglichkeit, direkt nach dem Inhalt von META-Tags oder von RDF-Beschreibungen zu suchen. Die Frage, in welchem Maß oder ob überhaupt die großen Suchmaschinen META-Tags oder RDF-Beschreibungen bei der Indexierung von Webseiten beachten, muss noch geklärt werden.

3.4 Standard-Suchmaschinen

Die Güte der gefundenen Webseiten, die das Ergebnis einer Anfrage bilden, hängt neben der Anfragemächtigkeit von der Indexierungstechnik der jeweiligen Suchmaschine ab. Während die meisten Suchmaschinen einen ähnlichen Satz an Anfrageoptionen bieten, kann der jeweilige *Indexer* die Analyse unterschiedlich durchführen. Die wichtigsten Komponenten einer Webseite, die bei einer Untersuchung eine Rolle spielen können, sind nachfolgend aufgelistet:

- **URL** einer Webseite.
- **Titel** einer Webseite, der mittels des TITLE-Tags angegeben wird.
- V**erweise in der HEAD-Sektion** der HTML-Beschreibung
- META-Tags oder die **RDF-Beschreibung** einer Webseite.
- **Sichtbarer Text** einer Webseite, der den textlichen Inhalt der Seite bildet. Er befindet sich in der BODY-Sektion der HTML-Beschreibung.
- **Verweise in der BODY-Sektion** der HTML-Beschreibung und ihr Text.

Welche dieser Komponenten Suchmaschinen bei der Indexierung analysieren, wird am Beispiel drei bekannter Suchmaschinen untersucht.

3.4.1 Altavista

Altavista wurde von der Firma *Digital Equipment Corporation* im Dezember 1995 in Betrieb gestellt. Sie ist einer der größeren Suchmaschinen mit schätzungsweise 397.000 indexierten Webseiten [Not02].

Altavista bietet viele der oben genannten Anfrageoptionen kombiniert mit einer einfachen Oberfläche, die sowohl für unerfahrene als auch für geübte Suchmaschinenbenutzer geeignet ist.

Die genaue Indexierungstechnik und vor allem die Relevanzbeurteilung von Webseiten für eine bestimmte Anfrage (*Rating*) von *Altavista* wird geheimgehalten. Die Betreiber nennen als Hauptgrund für diese Entscheidung die entstehenden Missbrauchmöglichkeiten, die nach einer Veröffentlichung entstehen würden. Mit diesem Wissen ausgestattet wären Gestalter von Webseiten in der Lage, Webseiten zu erstellen, die *Altavista* in die Irre führen könnten, und sie dazu verleiten würden, Benutzern Webseiten mit unerwünschtem Inhalt zu präsentieren.

Obwohl die genaue Arbeitsweise der Indexierungskomponente von *Altavista* unbekannt ist, gibt es von den Betreibern Hinweise, wie Webseiten eine hohe Wertung bei der Beurteilung erreichen können. Demnach spielen der Titel und die ersten Paar Zeilen des textlichen Inhaltes der Webseite die wichtigste Rolle bei der Indexierung. Obwohl die ersten Paar Zeilen besonders wichtig sind, wird der gesamte Text einer Webseite untersucht. Dabei findet weniger die Vorkommenshäufigkeit einzelner Wörter sondern vielmehr ihre Anordnung Beachtung. Eine Weitere Komponente, die bei der Indexierung untersucht wird, ist der Text von Verweisen, die auf eine gerade betrachtete Webseite zeigen.

Nach Angabe von *Altavista* werden auch META-Tags untersucht, wobei nur zwei weit verbreitete Metadatenelemente beachtet werden. Das *keywords*-META-Tag wird als “normaler” Text der Webseite behandelt, während das *description*-META-Tag bei der Präsentation der Suchergebnisse als Beschreibung der Webseite Verwendung findet. Der Grund

für die Ignorierung der META-Tags ist, wie *Altavista* mitteilt, die oben genannte Missbrauchmöglichkeit.

3.4.2 HotBot

HotBot ist wie *Altavista* eine der ältesten Suchmaschinen. Sie wurde von *Inktomi Corporation* im Mai 1996 in Betrieb gestellt und sorgte damals mit einem enorm Leistungsfähigen (10 Millionen Webseiten pro Tag) *Crawler* und einer riesigen Datenbank für Furore. Mittlerweile befindet sich *HotBot* mit geschätzten 332.000 indexierten Webseiten eher im Mittelfeld der Größenrangliste bekannter Suchmaschinen.

Zur Indexierungstechnik von HotBot gibt es wenige konkrete Hinweise. Ähnlich wie bei *Altavista* gibt es Empfehlungen von den Betreibern von HotBot für die Gestaltung von Webseiten, die für bestimmte Begriffe eine hohe Bewertung bei der Analyse erhalten sollen.

Demnach spielt die Vorkommenshäufigkeit einzelner Wörter in Webseiten eine entscheidende Rolle. Je öfter ein gesuchter Begriff in einer Webseite auftaucht, desto höher wird sie in die Ergebnisliste eingestuft. Zusätzlich wird die Länge des Dokumentes ins Verhältnis zu den Worthäufigkeiten gestellt. Eine kürzere Webseite wird höher eingestuft als eine längere mit gleicher Worthäufigkeit. Dem HTML-Titel einer Webseite schenkt auch HotBot viel Aufmerksamkeit. Ein Wort, das im HTML-Titel vorkommt, wird höher geschätzt als ein Wort, das im Text der Webseite steht.

Den Betreibern von *HotBot* zufolge werden drei Metadatenelemente in META-Tags untersucht. Inhalte des *keywords*- und des *author*-META-Tags stehen zwischen dem HTML-Titel und dem Text der Webseite auf der Wichtigkeitsskala. Das *description*-META-Tag wird bei der Ergebnisgenerierung verwendet.

Obwohl *HotBot* die Benutzung von META-Tags empfiehlt, warnt sie gleichzeitig vor Missbrauch. Die Mitteilung auf der deutschsprachigen HotBot Webseite für Webmaster [Lyc02] *"Wir wissen, dass manche Leute Seiten erstellen, um Suchmaschinen böswillig in die Irre zu führen"*, wird von einer Warnung gefolgt, die die Entfernung von manipulativen Webseiten aus dem HotBot Index androht.

3.4.3 Google

Google ist mit geschätzten 968.000 indexierten Webseiten die größte existierende Suchmaschine. Sie geht aus einem Forschungsprojekt der *Stanford University California* hervor und wird in einem abschließenden Bericht [BP98] ausführlich beschrieben.

Google verwendet bei einer bestimmten Anfrage ein komplexes Bewertungssystem für die Relevanzbeurteilung einer Webseite. Die zwei Hauptkomponenten des Systems sind der Rang (*page rank*) und die Trefferpunktzahl (*hit score*) einer Webseite.

Rang einer Webseite (page rank)

Der Rang einer Webseite wird wie folgt berechnet:

$$PR(A) = (1-d) + d \cdot \left(\frac{PR(T_1)}{C(T_1)} + \ldots + \frac{PR(T_n)}{C(T_n)} \right)$$

In der Formel bezeichnen die Variablen T_1 ... T_n Webseiten, die einen Verweis auf die Webseite *A* enthalten. *C(X)* bezeichnet die Anzahl der ausgehenden Verweise und *PR(X)* den Rang einer Webseite *X*. Der Dämpfungsfaktor *d* (Standardwert *d= 0,85*) erlaubt eine Optimierung der Formel.

Einfach ausgedrückt spiegelt der Rang einer Webseite ihre Popularität wieder. Je mehr Verweise auf eine Webseite von anderen "populären" (d.h. mit hohem Rang) Webseiten zeigen, desto höher ist ihr Rang. Je mehr ausgehende Verweise eine Webseite besitzt, desto weniger "Wert" haben diese für die Bewertung von anderen Webseiten.

Trefferpunktzahl (hit score)

Google durchsucht folgende Komponenten einer Webseite bei der Suche nach relevanten Begriffen:

- **URL** der Webseite
- **Titel** der Webseite
- **Text von Verweisen**, die auf die aktuelle Webseite zeigen
- **Sichtbarer Text** der Webseite
- **META-Tags**

Ein Treffer (hit) wird erzielt, wenn *Google* einen gesuchten Begriff in einer der oben genannten Komponenten der Webseite findet. Die einzelnen Komponenten werden gewichtet; ein Treffer im Titel oder in einem Verweis ist wichtiger als einer im sichtbaren Text. Zusätzlich werden folgende Merkmale eines Wortes beachtet:

- **Vorkommenshäufigkeit**
- **Fontgröße** (je größer der Font, desto mehr Bedeutung hat das Wort)
- **Stil der Wortdarstellung** (z.B. Fettdruck)

An Hand oben genannter Komponenten wird eine Gesamtpunktzahl (*hit score*), die eine Webseite abhängig von den jeweiligen Suchbegriffen erzielt, berechnet.

Google kombiniert die Trefferpunktzahl einzelner Begriffe und den Rang zu einem Relevanzwert, der schließlich die Reihenfolge der Ergebnisseiten bestimmt. Welche Metadaten in META-Tags untersucht werden, wird in dem Bericht nicht weiter präzisiert; zu vermuten ist, dass neben den weit verbreiteten Metadatenelementen *keywords* und *description* keine weitere Elemente Beachtung finden.

Die Bilanz der Untersuchung von Standard-Suchmaschinen zeigt, dass sie nur wenige Metadatenelemente und ausschließlich META-Tags analysieren. Standardisierte Metadatenelemente wie die Elemente des *DCMES*, werden von keiner der großen Suchmaschinen beachtet. Der offensichtliche Grund dafür, ist die bereits erwähnte Missbrauchmöglichkeit. Damit sind Standard-Suchmaschinen für ein Auffinden von Webseiten mit Ortsinformation ungeeignet.

3.5 Meta-Suchmaschinen

Das Wort “Meta” in der Bezeichnung “Metasuchmaschine” könnte vermuten lassen, dass diese speziellen Suchmaschinen etwas mit Metadaten zu tun haben; aus diesem Grund ist eine kurze Nachforschung auf jeden Fall sinnvoll.

Metasuchmaschinen erscheinen auf den ersten Blick wie gewönliche Suchmaschinen; sie präsentieren sich durch graphische Oberflächen, von denen aus in gewohnter Weise Suchanfragen gestellt werden können. Sie liefern ihre Ergebnisse allerdings nicht aus der eigenen Datenbank, sondern leiten die Anfrage an andere bekannte Suchmaschinen wie *Altavista* oder *LookSmart* weiter. Der Anfragestring einer Suche wird dabei analysiert und in die jeweiligen Anfragesprachen der befragten Suchmaschinen übersetzt. Die Metasuchmaschine wartet bis die Antworten eintreffen und entfernt alle Duplikate aus der Ergebnismenge. Nach Bewertung und Sortierung der erhaltenen Webseiten werden diese dem Benutzer präsentiert.

Bei genauer Betrachtung stellt sich also heraus, dass Metasuchmaschinen gar keine Suchmaschinen im eigentlichen Sinne sind. Sie bilden eher eine generelle Schnittstelle zu anderen Suchmaschinen, was zwar eine zusätzliche Filterung der Ergebnisse ermöglicht, gleichzeitig wird jedoch die Anfragesprache beschränkt und weniger mächtig. Damit sind Metasuchmaschinen für die Suche nach Webseiten mit Metadaten weniger als ihre “normalen” Kollegen geeignet.

3.6 Metadata-Suchmaschinen

Neben Metasuchmaschinen gibt es eine große Anzahl an weiteren Suchmaschinen und Suchwerkzeugen, die sich auf spezielle Themenbereiche oder auf bestimmte Ressourcentypen konzentrieren. Solche Bereiche sind Nachrichten, Finanzinformation oder

medizinische Daten. Beispiele für ressourcentypspezifische Suchmaschinen sind *Ditto* oder *AudioGalaxy,* die eine Suche nach Bildern oder Audiodaten ermöglichen.

Wünschenswert wäre ein Suchwerkzeug, das auch nach Metadaten in HTML-Dokumenten und in dazugehörenden XML/RDF-Dateien sucht. *HotMeta* [DST99] ist eine Suchmaschine, die diesen Anforderungen sehr nahe kommt. Sie entstand 1998 während eines Forschungsprojektes des *Distributed Systems Technology Centre (DSTC)* in Australien, und ist mittlerweile in der Version 1.4.6 verfügbar. *HotMeta* sucht nach *Simple Dublin Core* Metdatatenelementen in Webseiten und Benutzer können gefundene Metadaten nach gewünschten Begriffen durchsuchen. Es werden nur einfache Anfragestrings, die eine verknüpfende oder ausschließende Suche (siehe Kapitel 3.3.1) ermöglichen, unterstützt; die Auswahl einzelner Metadatenelemente für eine Anfrage (z.B. Suche in *DC.Title*) ist möglich.

HotMeta ist mittlerweile ein kommerzielles Produkt, was eine freie Verwendung ohne Lizensierung ausschließt. Sie untersucht während der Metadatensammlung keine RDF-Beschreibungen von Webseiten, was im Falle eines eventuellen Einsatzes in der NEXUS Infrastruktur eine Modifizierung erfordern würde. Da der Source-Code von *HotMeta* nicht einmal bei einer Lizensierung verfügbar wird, kommt sie als Lösung nicht in Frage.

3.7 Open-Source-Suchmaschinen

Komerzielle Suchmaschinen sind aus mehreren oben erwähnten Gründen für die Integration in die NEXUS Infrastruktur ungeeignet. Während der Suche nach geeigneten, frei verfügbaren (open source) Lösungen sind vor allem zwei Suchwerkzeuge aufgefallen: *ht://Dig* und *SWISH-E*.

3.7.1 ht://Dig

ht://Dig ist eine frei verfügbare Suchmaschine, die 1995 an der San Diego State University entstand. Seit dem sind viele Personen an ihrer Weiterentwicklung im Rahmen eines open source Projektes beteiligt gewesen. Sie ist mittlerweile in der Version 3.1.6 verfügbar und hat folgende Hauptmerkmale:

- Suche im World Wide Web oder im Intranet nach HTML-Dokumenten oder nach Textdateien
- Unterstützung von boolschen Ausdrücken in Anfragestrings
- Auswahl von verschiedenen Suchalgorithmen
- Analyse von META-Tags mit wenigen Metdatatenelementen (*keywords*, *description*, *robots*)

- Möglichkeit der Beschränkung der Suchtiefe
- Unterstützung des *Standard for Robot Exclusion* (siehe Kapitel 5.5.2.6)
- Konfigurierbare Darstellung der Suchergebnisse mittels HTML-Templates
- Frei verfügbare Quelltext der gesamten Suchmaschine unter der *General Public License Version 2.0*

Da eine Untersuchung von META-Tags bei der Indexierung nicht im gewünschten Maß stattfindet und die Eintragung der Ergebnisse in ein *Spatial Model Server* zusätzlich implementiert werden muss, ist eine Anpassung des Programmes unumgänglich. Leider ist *ht://Dig* in der Programmiersprache C++ geschrieben; eine Softwarelösung sollte allerdings in der Programmiersprache Java entstehen, was eine Verwendung von *ht://Dig* ausschließt.

3.7.2 SWISH-E

SWISH-E ist ein freies Such- und Indexierungswerkzeug, das an der University of California seit mehreren Jahren unter der Beteiligung von vielen Einzelpersonen entwickelt wird. Es übertrifft in vielen Bereichen die Fähigkeiten von *ht://Dig*; unter anderem ist *SWISH-E* in der Lage neben HTML-Dokumenten und Textdateien auch XML-Dokumente zu analysieren. Damit ist es für die Suche und Analyse von Webseiten und RDF-Beschreibungen hervorragend geeignet; die einzige zu lösende Aufgabe wäre die entsprechende Formatierung und Abspeicherung gesammelter Daten in einem *Spatial Model Server. SWISH-E* ist in der Programmiersprache C geschrieben, was seine Verwendung für diese Studienarbeit aus dem gleichen Grund wie bei *ht://Dig* unmöglich macht.

Während der Suche nach geeigneten Softwarelösungen ist ein Webroboter, der während einer Dimplomarbeit an der Universität und Gesamthochschule Siegen entstand, aufgefallen. Der Webroboter, der in der Programmiersprache geschrieben wurde, schien Teilaspekte der gestellten Aufgabe lösen zu können und wurde schließlich für die Implementierung ausgewählt. Damit erübrigte sich die weitere Betrachtung von Suchwerkzeugen.

KAPITEL 4 ERWEITERUNG DES AUGMENTED WORLD MODELS

In diesem Kapitel werden Datenstrukturen zur Repräsentation von Webseiten mit Ortsinformation im Augmented World Model vorgestellt.

4.1 Augmented World Model

Das Weltmodell von NEXUS, das *Augmented World Model*, ermöglicht sowohl die Abbildung realer Objekte unserer Welt als auch die Modellierung von virtuellen Objekten, die der Informationsdarstellung dienen. Die Beschreibung dieser Objekte erfolgt durch das *Augmented World Class Schema*. Dieses Klassenschema ähnelt den Klassenschemata objektorientierter Programmiersprachen. Es modelliert Objekte, Objekteigenschaften sowie Beziehungen zwischen Objekten durch hierarchisch geordnete Klassen.

4.2 Virtual Web Portal

Im Zentrum des Interesses dieser Studienarbeit steht die Bereitstellung von Webseiten in der NEXUS Infrastruktur, die durch entsprechende Metadaten einen Zusammenhang zwischen ihrem Inhalt und einer geographischen Position darstellen. Anwendungen von NEXUS werden an Webseiten zu einer gegebenen Position interessiert sein. Die Gruppierung von Webseiten im *Spatial Model Server* nach der geographischen Position ist daher naheliegend.

Im World Wide Web werden Webseiten oft zu so genannten *Webportalen* zusammengefasst. Ein *Webportal* präsentiert dem Besucher Webseiten, die Information über ein gemeinsames Themengebiet präsentieren. In dem Spatial Model Server können mehrere Webseiten, die zu einer bestimmten geographischen Position gehören, ähnlich zusammengefasst werden. In das *Augmented World Class Schema* wird zu diesem Zweck eine neue Klasse, die *Webportale* modelliert, eingeführt. Ein *Webportal* ist kein reelles Objekt mit physischen Eigenschaften, deswegen wird es *VirtualWebPortal* genannt.

Jedes Objekt im dem *Augmented World Class Schema* wird aus der Oberklasse NEXUS*Object* abgeleitet. Die Klasse *DataObject* modelliert statische oder mobile Datenobjekte, die wiederum virtuell oder real sein können. Das *VirtualWebPortal* ist ein statisches, virtuelles Datenobjekt, das zusätzlich eine geographische Position repräsentiert. Es wird von *DataObject* über *SpatialObject* und *StaticObject* abgeleitet. Die Einordnung des *VirtualWebPortals* in den entsprechenden Teil des *Augmented World Class Schema* wird in Abbildung 4-1 dargestellt.

Jedes Objekt, das von *DataObject* abgeleitet wird, muss ein *id* und ein *kind*-Attribut besitzen. Das *VirtualWebPortal* soll außerdem das optionale Attribut *pos* vom *SpatialObject* erben. Damit hat es folgende Attribute:

- *id*: Die eindeutige Identifikation von jedem *VirtualWebPortal* in der NEXUS Infrastruktur. Der Wert für dieses Attribut ist ein NEXUS *Object Locator (NOL)*.
- *kind*: Die Art (*virtual* oder *real*) vom jeden *VirtualWebPortal*. Wie der Name schon verrät werden für dieses Attribut alle Webportale den Wert *virtual* besitzen.

- *pos*: Die Position, die das Webportal repräsentiert. Der Wert für dieses Attribut ist eine Position, die im *WGS84* Format dargestellt wird.

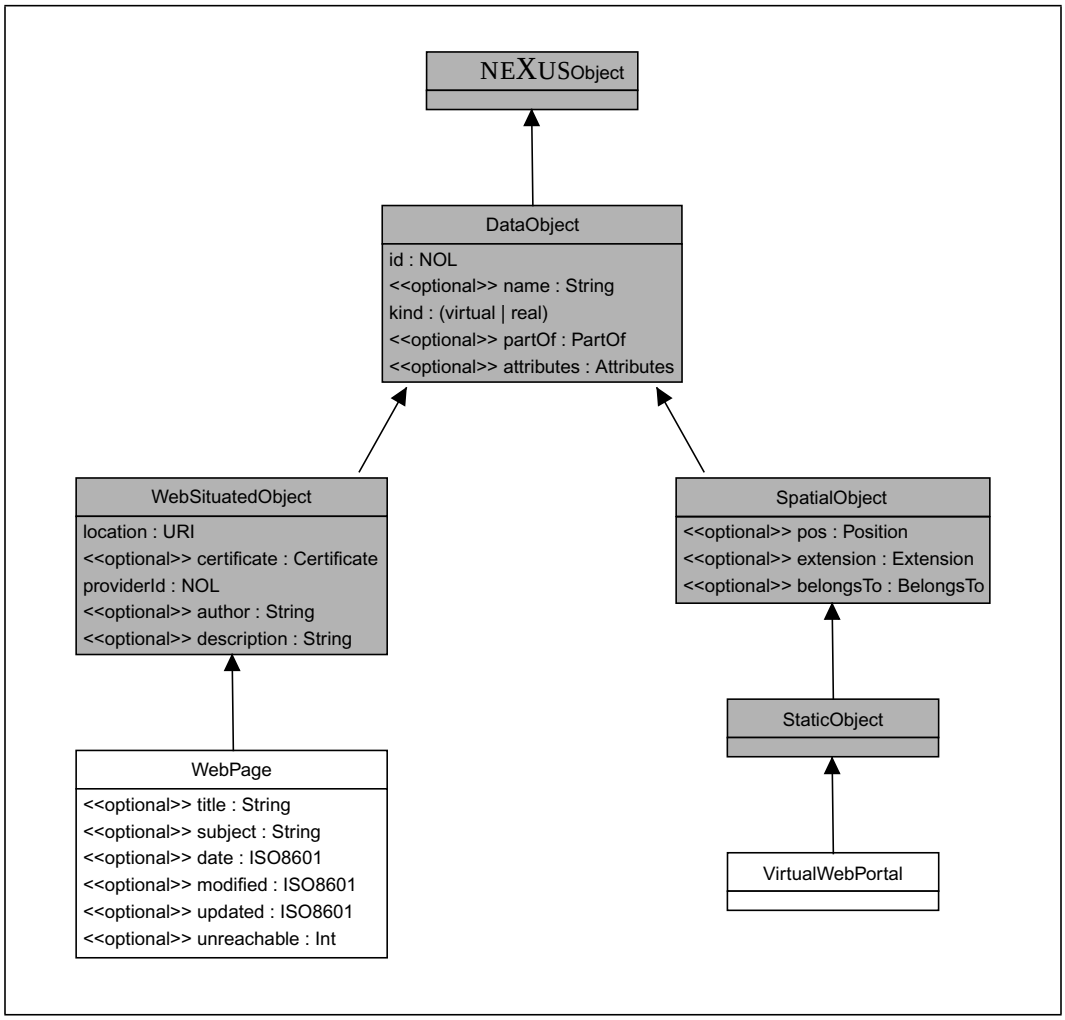

Abbildung 4-1: VirtualWebPortal und WebPage im AWCS

4.3 Web Page

Das *VitualWebPortal* hat nur eine zusammenfassende Funktion. Die tatsächliche Information wird von Objekten der Klasse *WebPage* getragen. Diese Klasse ist bereits im *Augmented World Class Schema* vorhanden, besitzt allerdings nicht alle nötige Attribute. Unter der Beibehaltung vorhandener Attribute wird *WebPage* mit neuen Merkmalen ausgestattet.

Die Klasse *WebPage* wird von *DataObject* und von *WebSituatedObject* abgeleitet. Sie besitzt die Attribute *id* und *kind*, die jedes von *DataObject* abgeleitete Objekt erbt. Von *WebSituatedObject* erbt *WebPage* die Attribute *location* und *providerId*, die jedes Web-Objekt besitzt. *WebPage* soll außerdem noch das optionale Attribut *partOf* vom *DataOb-*

ject und die optionalen Attribute *author* und *description* von *WebSituatedObject* erhalten. Alle geerbten und neuen Attribute sind die folgenden:

Von *DataObject* geerbte Attribute:

- *id*: Die eindeutige Identifikation von jeder WebPage in der NEXUS Infrastruktur. Der Wert für dieses Attribut ist ein NEXUS Object Locator.
- *kind*: *WebPage* ist ein virtuelles Objekt, deswegen erhält dieses Attribut den Wert *virtual*.
- *partOf*: Mit diesem Attribut wird auf das *VirtualWebPortal,* zu dem dieses *WebPage* gehört, verwiesen. Der Wert für dieses Attribut ist das *NOL* vom zugehörigen *VirtualWebPortal*.

Von *WebSituatedObject* geerbte Attribute:

- *location*: Die Adresse der Webseite angegeben durch ein URI. Dieses Attribut entspricht dem *Dublin Core* Element *DC.Indentifier.*
- *providerId*: Die Herausgeber der Ressource, die durch die Webseite dargestellt wird. Das entsprechende *Dublin Core* Element ist *DC.Publisher.*
- *author*: Der Autor der Ressource, die durch die Webseite dargestellt wird. Dieses Attribut entspricht dem *Dublin Core* Element *DC.Creator.*
- *description*: Die Beschreibung der Ressource, die die Webseite darstellt. Das entsprechende *Dublin Core* Element ist *DC.Description*.

Neue Attribute:

- *title*: Der Titel der Ressource. Dieses Attribut entspricht dem *Dublin Core* Element *DC.Title*.
- *subject*: Das Thema der Ressource. Das entsprechende *Dublin Core* Element ist *DC.Subject*.
- *date*: Das Datum der Ressource. Dieses Attribut entspricht dem *Dublin Core* Element *DC.Date*.
- *modified*: Das Datum der letzten Modifizierung der Ressource. Dieses Attribut unterstützt der Update-Modus von *DCbot*.
- *updated*: Das Datum an dem die gegebene Webseite im *Spatial Model Server* das letzte Mal aktualisiert wurde. Ist das Datum in diesem Feld älter als das Datum im *modified* Feld, dann wird die Seite aktualisiert.

- *unreachable*: Dieses Attribut gibt an, wie oft die Webseite bei einem Update-Versuch unerreichbar war. Erreicht der Wert in diesem Feld einen speziellen Schwellwert, dann wird die Webseite aus dem *Spatial Model Server* entfernt.

KAPITEL 5 ERWEITERUNG EINES WEBROBOTERS

In diesem Kapitel wird eine Softwarelösung für die Suche und die Speicherung von Metadaten über Webseiten vorgestellt.

5.1 Zielsetzung

Die Untersuchung der Suchmaschinen ergab, dass sie keine zufrieden stellende Möglichkeit für die Suche nach Metadaten in Webseiten bieten. Die Einbindung von Webseiten in die NEXUS Infrastruktur wird deswegen mit einer eigenen Softwarelösung realisiert.

Der zu entwickelnde Webroboter soll das World Wide Web durchwandern und Webseiten auf Metadaten hin untersuchen. Er soll sowohl in META-Tags als auch in XML / RDF-Beschreibungen nach *Dublin Core* Metadaten suchen. Webseiten, die Ortsinformation in dem *DC.Coverage* Element tragen, sollen in ein *Spatial Model Server* gespeichert werden. Um die Webseiten in den *Spatial Model Server* einzutragen, soll die *Augmented World Query Language (AWQL),* die Anfragesprache von NEXUS benutzt werden. Der Webroboter soll außerdem in der Lage sein, den aktuellen Datenbestand aus dem *Spatial Model Server* zu erfragen und die Daten zu aktualisieren.

5.2 Entwicklungsumgebung

Die Aufgabenstellung verlangt die Implementierung einer eigenen Softwarelösung in der Programmiersprache Java. *DCbot* wurde mit der Java Version 1.4.0 unter dem Betriebssystem Solaris 5.8 entwickelt. Die Wahl fiel unter anderem auf die neueste Java Version 1.4.0, wegen der neu eingeführten Möglichkeit, *Strings* mit regulären Ausdrücken zu vergleichen. Dieses Merkmal hat sich besonders bei der Entwicklung des *RDF-Parsers* als vorteilhaft erwiesen.

5.3 MediaFox

Die Entwicklung eines kompletten Webroboters mit allen geforderten Eigenschaften ist eine komplexe Aufgabe, die den Rahmen einer Studienarbeit sprengen würde. Aus diesem Grund wurde die Verwendung von bereits realisierten Komponenten in Betracht gezogen.

Bei der Suche nach einer passenden Softwarelösung wurden mehrere frei verfügbare Programme, wie *SWISH-E* oder *ht://Dig* untersucht. Sie waren jedoch teilweise zu komplex oder zu unspezifisch und daher ungeeignet. Schließlich wurde der Webroboter *MediaFox*, der während der Diplomarbeit “Resource Discovery und Information Retrieval” von Lars Eilebrecht in 1998 an der Universität - Gesamthochschule Siegen entstand, ausgewählt.

Der *MediaFox* Webroboter kann, ausgehend von einer Startwebseite, das World Wide Web durchwandern und die META-Tags der besuchten Seiten untersuchen. In einer Konfigurationsdatei werden Metadatenelementnamen angegeben, nach denen der

Webroboter suchen soll. Werden META-Tags mit der entsprechenden Information gefunden, so werden sie und der URL der Webseite in einer Datenbank gespeichert. Der Aufbau des Programmes ermöglicht den Einsatz einer beliebigen Datenbank für die Datenspeicherung. Die Anbindung der ausgewählten Datenbank erfolgt durch die Implementierung einer Schnittstelle.

Der *MediaFox* Webroboter ist freie Software unter einer *FreeBSD* ähnlicher Lizenzvereinbarung. Sie enthält eine Erklärung, die die Haftung des Autors und der Universität - Gesammthochschule Siegen für eventuelle Schäden, verursacht durch den Webroboter, in jedem Fall ausschließt. Die Lizenzvereinbarung ist in dem Quelltext der verwendeten Teile von *MediaFox*, wie in der Vereinbarung verlangt, vollständig enthalten. Neuimplementierte Programmteile enthalten eine entsprechende Lizenzvereinbarung. Auf ausdrücklichen Wunsch des Autors von *MediaFox*, wird der veränderte Webroboter den Namen *"MediaFox"* nicht mehr verwenden. Der weiterentwickelte Webroboter heißt *DCbot*.

5.4 Modifikationen

Der Webroboter *MediaFox* befriedigt nicht alle Wünsche, die in der Zielsetzung definiert wurden. Er kann zwar META-Tags von Webseiten analysieren, eine XML / RDF-Beschreibung lässt er allerdings außer Acht. Aus diesem Grund wird *DCbot* eine Komponente zur Analyse von XML / RDF-Dateien besitzen.

In dem *Spatial Model Server* werden Ortsangaben in dem *WGS84* Format gespeichert. In Anbetracht der vielen verschiedenen Formate für geographische Positionsbestimmung ist es unwahrscheinlich, dass Webseiten, die Ortsinformation tragen, gerade dieses Format benutzen. Deshalb werden Werte der *DC.Coverage* Elemente, wenn möglich, in das *WGS84* Format gewandelt.

Um Datumsangaben im *Spatial Model Server* einheitlich zu halten, wird für Daten das *ISO8601* Format verwendet. *DCbot* wird versuchen andere Formate in *ISO8601* umzuwandeln, und Daten, die ein unbekanntes Format haben außer Acht zu lassen.

MediaFox hat seine Daten in einem *MySQL*-Datenbanksystem mittels einer *JDBC*-Schnittstelle gespeichert. *DCbot* wird zusätzlich in der Lage sein, die *Dublin Core* Metadaten in einem *Spatial Model Server* abzuspeichern. Die Fähigkeit eine beliebige Datenbank, die mittels *JDBC*-Treiber ansprechbar ist, für die Speicherung zu benutzen, wird beibehalten.

DCbot ist eine Weiterentwicklung von *MediaFox* mit einer erweiterten Suchmächtigkeit. Außer einem verbesserten Suchverhalten, werden weitere Funktionen wie die Gruppierung von mehrfach auftretenden Metadatenelementen (siehe Kapitel 5.5.3.3) oder eine Steuerung des Crawling-Verhaltens (siehe Kapitel 5.5.5.12) implementiert.

5.5 DCbot

Der Name des Webroboters ist ein Wortspiel aus der Bezeichnung *Dublin Core* und der zweiten Silbe des Wortes *robot*. *DCbot* realisiert die in der Zielsetzung (Kapitel 5.1) erwähnten Aspekte vollständig. Die generelle Funktionsweise, die Benutzerschnittstelle sowie die Hauptkomponenten (besonders die eigens entwickelte) werden nachfolgend detailliert beschrieben.

5.5.1 Generelle Funktionsweise

Die generelle Funktionsweise von *DCbot* wird in der folgenden Abbildung dargestellt:

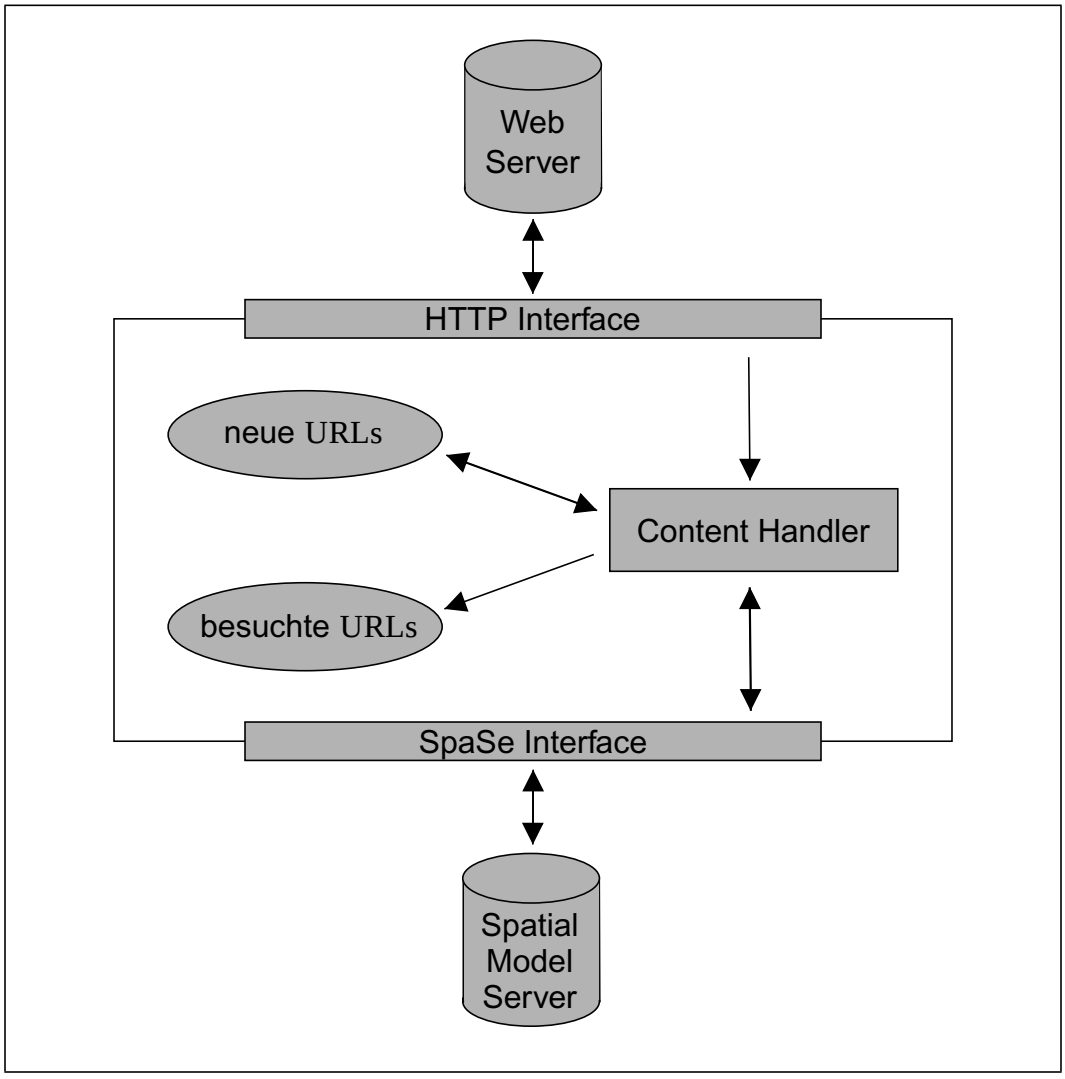

Abbildung 5-1: Generelle Funktionsweise von DCbot

DCbot verwaltet zwei URL-Listen. Die Liste der neuen URLs enthält URLs von Webseiten, die noch zu untersuchen sind. Bei einer normalen Sitzung erhält diese Liste am Anfang ein Start-URL. Befindet sich *DCbot* im “update” Modus, dann wird die Liste der neuen URLs am Anfang mit URLs aus dem *Spatial Model Server* gefüllt.

Der *Content-Handler* holt nacheinander URLs aus der Liste der neuen URLs. Er holt eine Webseite mittels des *HTTP-Interface* von einem bestimmten Web Server. META-Tags und die RDF-Beschreibung der Webseiten werden nach Metadaten durchsucht. Enthält die Webseite gesuchte Metadaten, dann wird sie in dem *Spatial Model Server* gespeichert. Der *Content-Handler* extrahiert außerdem alle Verweise aus einer Webseite und trägt sie in die Liste der neuen URLs ein. Schließlich wird der URL einer verarbeiteten Webseite in die Liste der besuchten URLs eingetragen.

DCbot wird beendet, wenn die Liste der neuen URLs leer ist oder die konfigurierbare maximale Anzahl der zu besuchenden Webseiten erreicht ist.

5.5.2 Funktionsumfang

5.5.2.1 Allgemeine Eigenschaften

- Alle wichtige Konfigurationsparameter des Roboters werden in Konfigurationsdateien gehalten. Dadurch ist eine Anpassung des Roboters an veränderte Bedingungen leicht zu bewerkstelligen.
- Der Suchbereich kann auf einen URL-Bereich (z.B. *http://www.domain.com/abc/*) oder auf eine Domain (z.B. *http://www.domain.com*) beschränkt werden.
- Die maximale Anzahl der zu untersuchenden Webseiten kann auf einen gewünschten Wert beschränkt werden.
- Die Entfernung (Anzahl der Verweise vom Start-URL), in der der Roboter Webseiten untersuchen soll, kann angegeben werden.
- Automatische Konvertierung von Datumsangaben. Die Formate *RFC 1123, RFC 850* und *POSIX/asctime()* werden in das Format *ISO 8601* [Kuh01] gewandelt.
- Automatische Konvertierung von Ortsangaben in das *WGS84* Format.
- Gruppierung von mehrfach auftretenden Metadatenelementen.
- Vollständige Unterstützung des *Standard for Robot Exclusion (SRE)* (siehe 5.5.5.14)
- Unterstützung von Robot META-Tags (*noindex, nofollow, usw.*)

5.5.2.2 HTTP Interface

DCbot realisiert die Verbindung zu einem Webserver mit dem *HTTPClient* (Version 0.3-3) von Ronald Tschalär. *HTTPClient* ermöglicht im Gegensatz zu dem *java.net.URLConnection*-Paket nur Verbindungen per HTTP. Als Ausgleich bietet er erweiterte Funktionen wie zum Beispiel das Setzen von Timeouts für eine Verbindung. Die wichtigsten Merkmale vom *HTTPClient* sind:

- Unterstützung von HTTP / 1.0 und HTTP / 1.1
- SOCKS Version 4 und Version 5 Unterstützung
- Unterstützung von dauerhaften Verbindungen
- Möglichkeit der Angabe von Timeouts
- Freies Software unter der *GNU LGPL*-Vereinbarung

5.5.2.3 HTML Parser

Zur Analyse der Webseiten wird das von Sun Microsystems angebotene Paket sun.net.www.html herangezogen. Das Paket besitzt folgende Hauptmerkmale:

- Unterstützung von META-Tags in der HTML *2.0* und HTML *4.0* Version.
- Analyse von Verweisen in der *BODY*-Sektion von HTML, sowie in Frames.
- Konfiguration der gesuchten Metadaten in einer Konfigurationsdatei.

5.5.2.4 RDF Parser

Der *RDF-Parser* von *DCbot* wurde vollständig in Rahmen der Arbeit implementiert. Die Haupteigenschaften sind:

- Korrektheitsüberprüfung von XML / RDF-Dokumenten.
- Unterstützung von *Simple Dublin Core* Metadaten in XML / RDF.
- Konfiguration der gesuchten Metadaten in einer Konfigurationsdatei.

5.5.2.5 Datenbankanbindung

Die Schnittstelle von *DCbot* für die Informationsspeicherung wurde vollständig im Rahmen der Arbeit entwickelt. Folgende Hauptmerkmale für die Datenspeicherung sind charakteristisch:

- Speicherung der Daten in einer Datenbank oder einem *Spatial Model Server.*

- Aktualisierungsmöglichkeit des Datenbestandes.
- Möglichkeit des Löschens aller Daten.

5.5.3 Konfigurationsparameter

DCbot unterstützt verschiedene Modi wie “normal” und “update” Betrieb, außerdem sind weitere Parameter, die das “Surf-Verhalten” und die Metadatenauswahl des Webroboters bestimmen, konfigurierbar. Die Auswahl einer Betriebsart und die Angabe eines Start-URLs geschehen an der Kommandozeile, während weitere Konfigurationsparameter in Konfigurationsdateien angegeben werden.

5.5.3.1 Kommandozeilenoptionen

DCbot ist ein Java Programm, das an der Kommandozeile aufgerufen wird.

Aufrufsyntax:

```
java URL  [Konfigurationsdatei]
```

Der übergebene URL dient als Start-URL für die Suche. In der Konfigurationsdatei stehen wichtige Einstellungen, die das Verhalten von DCbot maßgeblich bestimmen. Die Angabe einer Konfigurationsdatei ist optional. Wird keine angegeben, dann versucht das Programm standardmäßig die Datei *DCbot.config* zu laden. Ist die Datei *DCbot.config* nicht in dem Verzeichnis, indem sich das Hauptprogramm befindet, vorhanden, dann versucht *DCbot* die Datei */etc/DCbot.config* zu laden. Scheitern alle Versuche eine Konfigurationsdatei zu finden, dann wird das Programm mit einer Fehlermeldung beendet.

DCbot unterstützt außer dem Normalbetrieb noch weitere Betriebsarten, die mit dem folgenden Aufruf gestartet werden können:

```
java  DCbot  Option  [Konfigurationsdatei]
```

Die Angabe einer Konfigurationsdatei ist wieder freiwillig. Als Option darf man folgende Werte angeben:

- *version*: Der Name und die Versionsnummer des Programmes werden ausgegeben.
- *config*: Wenn diese Option angegeben wird, dann zeigt *DCbot* die Konfigurationseinstellungen, die in der Datei *DCbot.config* gespeichert sind.
- *initdb*: Die Verbindung zur Datenbank oder zum *Spatial Model Server* wird überprüft. Wird eine Datenbank zur Speicherung der Daten verwendet, dann legt *DCbot* die benötigte Tabelle an. Kommt ein *Spatial Model Server* zum Einsatz, dann wird die Möglichkeit, *Virtuelle Webportale* und *Webseiten* abzulegen, überprüft. Da der *Spatial*

Model Server die dynamische Erstellung und Veränderung von Datenstrukturen über *AWQL* nicht zulässt, müssen entsprechende Datenstrukturen vom Systemadministrator vor dem Betrieb von *DCbot* eingerichtet werden.

- *update*: Mit dieser Option aufgerufen, aktualisiert *DCbot* den Datenbestand der Datenbank oder des *Spatial Model Servers*. Er besucht alle gespeicherten Webseiten erneut und überprüft, ob die vorhandene Information aktuell ist. Ist dies nicht der Fall, dann werden die veralteten Daten durch aktuelle ersetzt.
- *deletedb*: Kommt eine Datenbank zum Einsatz, dann wird die Tabelle, die für die Speicherung angelegt wurde (*initdb*), gelöscht. Wird ein *Spatial Model Server* verwendet, werden alle *Virtuelle Webportale* und dazugehörende *Webseiten* aus dem Server entfernt.

5.5.3.2 DCbot Konfigurationsdatei

Durch Angaben in der Konfigurationsdatei kann man das Verhalten von *DCbot* maßgeblich bestimmen. Die verschiedene Parameter und die dazugehörende Werte stehen jeweils in einer Zeile. Damit ergibt sich die Form:

```
Parameter1      Wert1
Parameter2      Wert2
...
eof
```

Die Konfigurationsparameter und die erlaubten Werte sind:

- *Homepage*: Den Wert für diesen Parameter kann *DCbot* zusätzlich per HTTP-Header übermitteln, um sich bei einem Webserver zu identifizieren. Wird als Homepage *http://www.domain.com* angegeben dann schickt *DCbot* folgendes im HTTP-Header mit:

  ```
  User-Agent: DCbot/2.0 (http://www.domain.com)
  ```

  Wird dieser Parameter nicht angegeben, dann übermittelt das Programm nur seinen Namen und ihre Versionsnummer.

- *EMail*: Als Kontaktadresse für eventuelle Nachfragen kann eine Email Adresse spezifiziert werden. Wird als Wert *robotadmin@domain.com* angegeben, dann steht im HTTP-Header zusätzlich:

  ```
  From: robotadmin@domain.com
  ```

  Wird dieser Parameter nicht angegeben, dann übermittelt das Programm “unknown” als Adresse.

- *AcceptLanguage*: Mit diesem Parameter wird *DCbot* angewiesen, nur Webseiten von einem Webserver zu akzeptieren, die in einer bestimmten Sprache verfasst sind. Die Sprache einer Webseite sollte ebenfalls im vom Webserver geschickten, HTTP-Hea-

der stehen. Sollte *DCbot* zum Beispiel deutsche und englische Dokumente akzeptieren, wobei deutsche den Vorrang haben, dann muss dieser Parameter den Wert *de,en;q=0.8* erhalten.

- *ServerMode*: Mit diesem Parameter wird *DCbot* mitgeteilt, innerhalb welcher URL-Bereiche oder Domänen er sich bewegen darf. Gültige Werte für diesen Parameter sind:
  - *single*: Im *Single-Modus* verlässt *DCbot* nie den URL-Bereich in dem er gestartet wurde. Wird er mit dem URL *http://www.domain.com/abc/xzy*.html aufgerufen, dann werden nur im URL-Bereich *http://www.domain.com/abc/* Webseiten besucht. Die entsprechende Konfigurationszeile für diese Einstellung ist:

    ```
    ServerMode      single
    ```

  - *multi*: Im *Multi-Modus* verfolgt *DCbot* auch Verweise, die in andere URL-Bereiche oder Domänen führen. Eine Beschränkung auf eine spezielle Domain ist auch in diesem Modus möglich. Möchte man den Roboter auf die Domain *http://www.domain.com* beschränken, dann müssen dem Wert *multi* ein Doppelpunkt und die Domain folgen. Eine entsprechende Zeile in der Konfigurationsdatei hat die Form:

    ```
    ServerMode      multi:.domain.com
    ```

  - *indexpage*: Im diesem Modus wird der Start-URL als eine *Index-Seite* betrachtet. Alle Verweise dieser Seite werden im *Single-Modus* weiterverfolgt. Eine Beschränkung des Roboters auf eine Domain ist wieder möglich. Die Konfigurationszeile hat die Form:

    ```
    ServerMode      indexpage:.domain.com
    ```

- *RequestRate*: Mit diesem Parameter ist die Angabe einer Wartezeit zwischen zwei HTTP-Anfragen möglich. Eine zu kleine Wartezeit gilt als unhöflich, weil häufige Anfragen die Last am Zielwebserver zu stark erhöhen. Der Minimalwert für diese Einstellung ist eine Sekunde. Als Standardwert benutzt *DCbot* 15 Sekunden.
- *MaxDepth*: Der Wert für diesen Parameter bestimmt, wie weit sich *DCbot* vom Start-URL aus entfernen darf. Mit Entfernung ist hier die Anzahl der Verweise, die *DCbot* vom Start-URL aus bis zu einer bestimmten Webseite verfolgen muss, gemeint.
- *MaxRequests*: Der Wert für diesen Parameter bestimmt, wie viele Webseiten maximal besucht werden dürfen. Als Standardwert benutzt *DCbot* für diese Einstellung den Wert 1000.
- *MaxLinkStorage*: Dieser Wert reguliert die Größe des URL-Puffers. Damit lässt sich die “Breite” der Suche von *DCbot* regeln. Genauer gesagt lässt sicht die Anzahl von Webseiten festlegen, die in jeder Entfernung vom Start-URL aus betrachtet, untersucht werden sollen. Wird beispielsweise der Wert “100” angegeben, dann versucht *DCbot* 100 Webseiten mit der Entfernung “eins”, 100 Webseiten mit der Entfernung “zwei”, usw. vom Root-URL aus zu besuchen.

- *Mapfile*: Mit diesem Parameter wird der Pfad für die *map*-Datei angegeben. *DCbot* sucht standardmäßig im Hauptverzeichnis nach der Datei *DCbot.map*.
- *RequiredMetadata*: Wird ein Metadatenelement und ein dazugehörender Wert mit diesem Parameter angegeben, dann wird eine Seite im *Spatial Model Server* nur dann abgelegt, wenn entweder ein META-Tag oder die RDF-Beschreibung der Seite das geforderte Metadatenelement enthält und sein Wert mit dem geforderten Wert übereinstimmt. Sind zum Beispiel nur Webseiten interessant, die für das Metadatenelement *DC.Date* den Wert *2002-04-30* enthalten, dann sieht die erforderliche Zeile der Konfiguratiosdatei so aus:

  ```
  RequiredMetadata    DC.Date:2002-04-30
  ```

  Anstatt eines Wertes kann man das '*' Symbol angeben, wenn nur das Metadatenelement gesucht wird und der dazugehörige Wert keine Rolle spielt.
- *ForceOverwrite*: Für diesen Parameter können die Werte *yes* oder *no* angegeben werden. Wird eine Webseite besucht, die bereits im *Spatial Model Server* abgelegt ist, und der Wert *yes* ist gesetzt, dann wird die Seite auf jeden Fall aktualisiert. Ist der Wert *no* gesetzt, dann wird die Seite im *Spatial Model Server* nur aktualisiert, wenn das Datum der letzten Modifizierung veraltet ist.
- *Proxy*: Mit diesem Parameter kann ein *Proxy-Server* für die Webzugriffe von DCbot angegeben werden. Als Wert wird der Servername und die Portnummer, durch einen Doppelpunkt getrennt, erwartet. Beispiel:

  ```
  Proxy       webproxy.domain.com
  ```
- *NoProxy*: Wurde ein *Proxy-Server* definiert, dann können einzelne oder mehrere Domains, für die *DCbot* keinen *Proxy-Server* benutzen soll, angegeben werden. Beispiel:

  ```
  NoProxy   .mydomain.com, .ourdomain.com
  ```

DCbot ist in der Lage, für die Speicherung seiner Ergebnisse verschiedene Datenbanksysteme, die mittels einem *JDBC*-Treiber angesprochen werden können, zu verwenden. Daten können außerdem in ein *Spatial Model Server* abgelegt werden. Die *DCbot* Konfigurationsdatei bietet die Möglichkeit, die entsprechende Schnittstelle anzugeben.

- *DBInterface*: Mittels diesem Parameter wird *DCbot* mitgeteilt, welche Schnittstelle und damit welches Speicherungssystem benutzt werden soll. Mögliche Werte sind "DBI_JDBC" und "DBI_SpaSe". Mittels der *DBI_JDBC* Schnittstelle kann eine beliebige Datenbank, für die ein *JDBC*-Treiber existiert, angesprochen werden. In diesem Fall müssen Werte auch für die Konfigurationsoptionen *DBJDBCDriver*, *DBJDBCClass*, *DBName* und *DBTable* angegeben werden.

  Wird *DBI_SpaSe* als Schnittstelle angegeben, dann speichert *DCbot* seine Daten in einem *Spatial Model Server*.
- *DBJDBCDriver*: Der Name des zu verwendenden *JDBC*-Treibers.

- *DBJDBCClass*: Der vollständige Name der Java-Klasse des *JDBC*-Treibers. Zum Beispiel *COM.ibm.db2.jdbc.app.DB2Driver.*
- *DBName*: Der Name der zu verwendenden Datenbank.
- *DBTable*: Der Name der Datenbanktabelle.

5.5.3.3 DCbot map-Datei

In der *map*-Datei von *DCbot* kann man festlegen, welche in Webseiten gefundenen Metadaten gespeichert werden sollen. Man kann außerdem für jedes Metadatenelement festlegen, auf welches Attribut es im *Spatial Model Server* abgebildet werden soll. Die Grundsätzliche Struktur der *map*-Datei sieht so aus:

```
DC.Title        title
DC.Subject      subject
DC.Creator      creator
```

Die Namen in der ersten Spalte sind die Metadaten, nach denen *DCbot* in Webseiten suchen soll. Die zweite Spalte enthält den Attributnamen, der das jeweilige Metadatenelement im *Spatial Model Server* repräsentiert.

Die Möglichkeit der mehrfachen Angabe von Metadatenelementen besteht sowohl bei META-Tags als auch bei RDF-Beschreibungen. Die Behandlung solcher Gruppierungen war im Webroboter *MediaFox* nicht vorgesehen.

Bei *DCbot* besteht die Möglichkeit die Behandlung von Gruppierungen gesondert für jedes Metadatenelement einzuschalten. Soll die Gruppierung zum Beispiel für das Metadatenelement *DC.Creator* eingeschaltet werden, dann erhält dieses Metadatenelement in der dritten Spalte der *map*-Datei das Schlüsselwort “group”. Beispiel:

```
DC.Title        title
DC.Subject      subject
DC.Creator      creator         group
```

Mit einer solchen *map*-Datei fasst *DCbot* die META-Tags

```
<META name="DC.Creator" content="Homer Simpson">
<META name="DC.Creator" content="Bart Simpson">
```

zu dem META-Tag

```
<META name="DC.Creator" content="Homer Simpson; Bart Simpson">
```

zusammen. Die Einträge werden durch ein Semikolon und ein Leerzeichen getrennt. Die Anzahl der mehrfach auftretenden Metadatenelementen spielt keine Rolle. Die Behand-

lung von RDF-Beschreibungen geschieht analog. Zwei Zeilen eines XML/RDF-Dokumentes

```
<dc:creator>Homer Simpson</dc:creator>
<dc:creator>Bart Simpson</dc:creator>
```

werden zu einer Zeile

```
<dc:creator>Homer Simpson; Bart Simpson</dc:creator>
```

zusammengefasst. Enthält eine Webseite sowohl META-Tags, als auch eine RDF-Beschreibung, dann werden alle Metadaten zusammengefügt, und die Behandlung von mehrfachen Einträgen erfolgt für alle Metadaten.

Eine mehrfache Angabe eines Metadatenelementes ist nicht immer sinnvoll. Tritt ein solches Metadatenelement (z.B *DC.Date*) trotzdem mehrfach auf und die Gruppierung in der *DCbot* Map-Datei ist ausgeschaltet (sollte immer der Fall sein), dann wird nur das erste Vorkommen des Elementes beachtet.

5.5.4 Rückmeldungen

DCbot dokumentiert seine Arbeit durch Rückmeldungen auf das *Standard Output* des Systems in Form von kurzen Textmeldungen. Viele Meldungen wie “Connecting to Spatial Model Server ...” oder “Finished. Processed 23 URLs.” sind selbsterklärend und werden hier nicht alle behandelt. Statusmeldungen, die pro untersuchte Webseite ausgegeben werden, sind trotzdem erwähnenswert, weil sie Information über den Erfolg der Suche nach Metadaten bieten.

Nach jeder besuchten Seite gibt *DCbot* eine Meldung der folgenden Form aus:

```
URL: /  http://www.domain.com/index.html
     \  Status: OK, Depth: 3, Checked: 23, Queued: 529)
```

Nach der URL der Seite wird eine Statusmeldung (“Status: *Meldung*”), die Entfernung vom Start-URL (“Depth: *Zahl*”), die Anzahl der besuchten Webseiten (“Checked: *Zahl*“) und die Anzahl der noch zu besuchenden Webseiten (“Queued: *Zahl*“) ausgegeben. *DCbot* hat folgende Statusmeldungen:

- *OK*: Der besuchte URL referenziert ein HTML-Dokument, das weder in META-Tags noch in einer RDF-Beschreibung die gesuchten Metadaten enthält.
- *OK/METADATA*: Eine Webseite mit Metadaten wurde gefunden, und sie wurde im *Spatial Model Server* gespeichert.

- *OK/UNCHANGED*: Die Webseite hat sich seit der letzten Aktualisierung nicht geändert. Die Daten im *Spatial Model Server* über diese Seite sind aktuell.
- *NO* HTML: Es handelt sich nicht um ein HTML-Dokument.
- *NO ACCESS*: Aufgrund der *robots.txt*-Datei oder eines entsprechenden META-Tags wird ein Zugriff auf diese Seite nicht durchgeführt.
- *FAILURE*: Das Dokument konnte nicht ordnungsgemäß heruntergeladen werden. Ursachen können dafür das Fehlen der Ressource (Status-Code: 404), ein Verbindungsabbruch oder die Überschreitung der vordefinierten Wartezeit (Timeout) sein.
- *PAGE GONE*: Das Dokument wurde vom Webserver entfernt (Status-Code: 410).
- *UNKNOWN HOST*: Der gegebene Webserver existiert nicht bzw. der benutzte DNS-Server haben keinen entsprechenden Eintrag.
- *ERROR*: Die HTTP-Anfrage des Roboters wurde mit dem Status-Code 400 (“bad request”) beantwortet. Tritt dieser Fehler häufiger auf, dann kann das auf ein Fehlverhalten des Roboters hindeuten. Eine genauere Untersuchung des Problems ist in diesem Fall erforderlich.

5.5.5 Implementierungsdetails

Nachfolgend wird die Zusammenarbeit der Hauptkomponenten von *DCbot* vorgestellt. Eine genauere Betrachtung von Teilaspekten erfolgt nur bei Programmteilen, die im Rahmen der Arbeit implementiert oder maßgeblich verändert wurden. Weitere Implementierungsdetails sind in [Eil98] zu finden.

Die *DCbot* Hauptklasse bildet den Rahmen des Programmes; sie enthält die Hauptschleife, die nacheinander die, in weitere Klassen untergliederte, Hauptfunktionen des Programmes aufruft. Mittels der Klassen *ConfigLineParser* und *ElementMapWrapper* werden die *DCbot* Konfigurationsdatei und die *map*-Datei eingelesen. Das *HTTPClient*-Paket sorgt für die Verbindung ins World Wide Web, während die *HTMLHandler*-Klasse die heruntergeladenen Webseiten und XML/RDF-Dateien untersucht. Für die Analyse der HTML-Dokumente und RDF-Beschreibungen sind das *sun.net.www.*html Paket und die *RDFParser*-Klasse zuständig. Nach einer erfolgreichen Suche können die gefundenen Metadaten wahlweise mittels der *DBI_JDBC*- oder der *DBI_SpaSe*-Klasse abgespeichert werden; beide Klassen implementieren das *DBI* Infterface von DCbot. Klassenzusammenhänge werden in Abbildung 5.2 dargestellt.

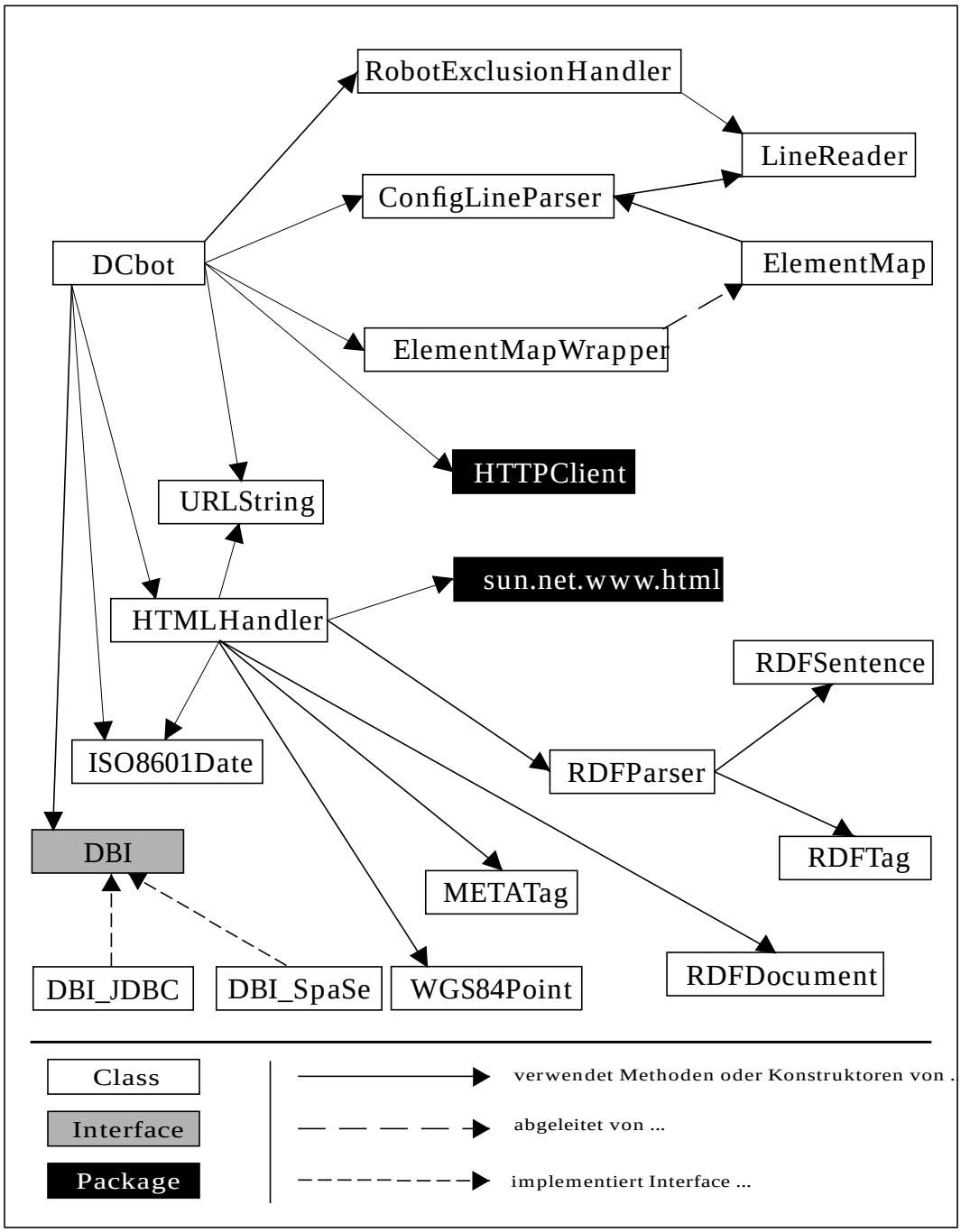

Abbildung 5-2: Klassenabhängigkeiten von DCbot

5.5.5.1 Methode main in DCbot.java

DCbot überprüft erst einmal, ob die übergebenen Parameter richtig sind. Im “normal”-Modus bedeutet dies die Syntaxüberprüfung des übergebenen URL und die Prüfung, ob

die angegebene Konfigurationsdatei existiert. Wird keine Konfigurationsdatei übergeben, dann wird die Datei *DCbot.config* gesucht. Das Programm lädt dann die Konfigurationseinstellungen aus der Konfigurationsdatei und aus der *map*-Datei. Nach dem Verbindungsaufbau zur Datenbank oder zum *Spatial Model Server* wechselt das Programm in den angegebenen Betriebsmodus (*normal*, *update* oder *indexpage*). Abhängig vom spezifizierten Betriebsmodus werden verschiedene Vorbereitungen getroffen. Im *update*-Modus wird zum Beispiel die Liste der zu besuchenden URLs mit den URLs aller Webseiten aus dem *Spatial Model Server* gefüllt. Nach den Vorbereitungen werden alle zu besuchenden URLs mit der Methode crawl(url) aufgerufen.

5.5.5.2 Methode crawl(url) in DCbot.java

Am Anfang der Methode erfolgt eine Syntaxüberprüfung des URLs. Die maximale Entfernung (Konfigurationsparameter *MaxDepth*) und die maximale Anzahl der zu besuchenden Webseiten (Konfigurationsparameter *MaxRequests*) werden überprüft. Wurde der Schwellwert für die Anzahl der zu besuchenden Webseiten erreicht, dann wird das Programm beendet. Je nachdem, ob eine Webseite die spezifizierte maximale Entfernung erreicht hat, wird die Methode retrieveSingleDocument(...) mit der Option *followlinks* oder *ignorelinks* aufgerufen, und die Verweise der Seite dementsprechend weiterverfolgt oder außer Acht gelassen. Die zu besuchende Links werden in einer Hash-Tabelle gespeichert, aus der die Seiten schubweise geholt werden. Dadurch ergibt sich ein der Breitensuche ähnliches Suchverhalten, das in Abbildung 5-3 dargestellt wird.

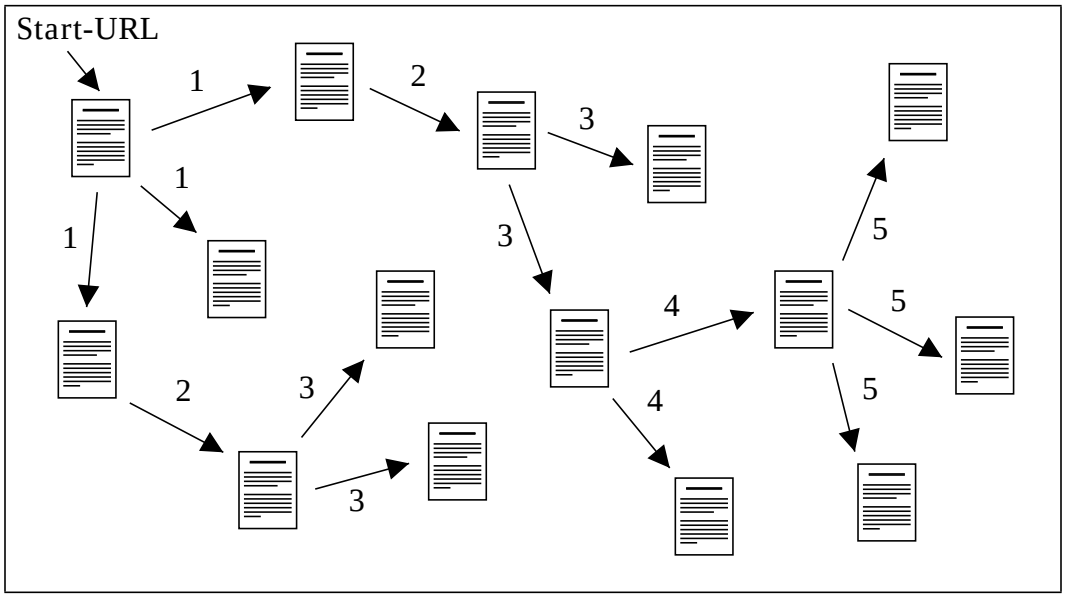

Abbildung 5-3: Suchverhalten von DCbot

5.5.5.3 Methode retrieveSingleDocument(...) in DCbot.java

Die Methode überprüft erst einmal ob der im URL gegebene Host existiert, und ob eine Robots.txt Datei den Zugriff auf die Webseite verbietet. Dafür wird die Verbindung zu dem Webserver mit dem Paket *HTTPClient* aufgebaut. Wird der Zugriff verweigert, dann wird eine entsprechende Meldung als Ergebnis des Methodenaufrufes zurückgeliefert. Ist ein Zugriff erlaubt, dann wird der HTML-Quelltext und eine eventuell vorhandene XML/RDF-Datei an den *HTMLHandler* übergeben. Wenn Verweise der aktuellen Webseite verfolgt werden sollen (*followlinks*), dann wird überprüft, ob die URLs sich im erlaubten URL-Bereich befinden (Bereichsbeschränkung durch verschiedene Modi). Gültige URLs werden in die Liste der zu besuchenden Links aufgenommen. Hat der *HTMLHandler* in der aktuellen Webseite gesuchte Metadaten gefunden, dann werden diese in den *Spatial Model Server* eingetragen.

5.5.5.4 HTMLHandler

Diese Methode ruft wichtige Programmteile auf, die die Analyse des HTML-Quelltextes und die Analyse der XML/RDF-Datei übernehmen. Die Metadaten werden in entsprechende Datenstrukturen abgelegt und gefiltert, bevor sie als Ergebnis zurückgeliefert werden.

Die Methode setDocTags(...) ruft den HTML-Parser (Paket *sun.net.www.*html) und den *RDF-Parser* auf. Der HTML-Parser extrahiert alle HTML-*Tags* aus dem HTML-Quelltext und speichert sie in das dafür vorgesehene Objekt *Document* ab. Der *RDF-Parser* extrahiert *RDFSentences* (siehe Kapitel 5.5.5.6) aus der XML/RDF-Datei und speichert sie in ein *RDFDocument*-Objekt ab.

Die Methode setMETATags() untersucht alle HTML-Tags und speichert alle META-Tags in der internen Variable *metatags* ab. Die Methode convertRDFSentences() wandelt die *RDFSentences* in META-Tags und speichert sie zusammen mit den eventuell vorhandenen META-Tags in der Variable *alltags* ab.

Bestimmte Metadaten haben nur dann Relevanz, wenn sie in dem geforderten Format vorliegen. Zum Beispiel ist dies bei dem Element *DC.Coverage* der Fall. Die Methode filterAllTags() sorgt dafür, dass nur Metadaten, die das richtige Format haben oder in das richtige Format konvertiert werden können, in *alltags* erhalten bleiben. Zur Zeit behandelt filterAllTags() die Metadatenelemente *DC.Coverage*, *DC.Date* und ihre Verfeinerungen. Wird also ein Metadatenelement *DC.Coverge* oder eine Verfeinerung des Elementes (z.B. *DC.Coverage.Spatial*) gefunden, dann untersucht *DCbot*, ob der Wert des Metadatenelementes dem *WGS84* Format entspricht oder in dieses Format umgewandelt werden kann (siehe Kapitel 5.5.5.10). Werte des Metadatenelementes *DC.Date* und Werte seiner Verfeinerungen (z.B. *DC.Date.Available*) sollten in dem *ISO8601* Format (siehe Kapitel 5.5.5.9) vorliegen. Wird ein entsprechendes Metadatenelement gefunden, dann wird es nur beibehalten, wenn sein Wert dem *ISO8601* Format entspricht oder in dieses Format

umgewandelt werden kann. Metadatenelemente, die weder in den geforderten Formaten vorliegen noch konvertiert werden können, werden aus *alltags* entfernt.

Die Methode handleMultipleEntries() erledigt die Gruppierung von mehrfach auftretenden Metadatenelementen (siehe Kapiel 5.5.3.3). Die Methode setFinalTags() filtert schließlich aus allen Metadaten nur die gesuchten Metadatenelemente heraus und speichert diese in der Variable *finaltags* ab. Ist ein Metadatenelement unbedingt erforderlich (siehe *RequiredMetadata* Kapitel 5.5.3.2), dann werden alle Metadaten nur gespeichert, wenn das Metadatenelement und der gesuchte Wert vorhanden sind.

5.5.5.5 RDFTag

Diese Klasse modelliert die einfachsten Tags von XML / RDF-Beschreibungen. Die allgemeine Form eines *RDFTags* sieht wie nachfolgend dargestellt aus:

```
<namespace:elementname>      für einen Start-Tag
</namespace:elementname>     für einen End-Tag
```

Der *Tag* enthält den Namen eines Metadatenelementes und die Bezeichnung eines Namensraumes, aus dem das Metadatenelement stammt. Der Konstruktor der Klasse erhält als Parameter den Metadatenelementnamen (*name*), die Bezeichnung des Namensraumes (*namespace*) und einen *int* Wert (0 für *Start-Tag* und 1 für *End-Tag)*, der den Typ (*type*) des *Tags* angibt:

```
RDFTag(String name, String namespace, int type)
```

Mit Hilfe der Zugriffsfunktionen getName(), getNamespace() und getType() kann man die entsprechenden Werte eines *Tags* erhalten. Beispiele für *RDFTags* sind:

```
<dc:title>     oder     </dc:title>
```

5.5.5.6 RDFSentence

Mit dieser Klasse wird ein *RDF-Satz* nachgebildet, der einem Attribut-Wert-Paar eines META-Tags gleichkommt. Die allgemeine Form sieht, wie nachfolgend dargestellt aus:

```
[Start-Tag] Wert [End-Tag]
```

Der *Start-Tag* und der *End-Tag*, die ein Metadatenelement angeben, schließen den dazugehörigen Wert ein. Der Konstruktor der Klasse erhält dementsprechend zwei *Tags* und einen Wert:

```
RDFSentence(RDFTag starttag, RDFTag endtag, String value)
```

Die Zugriffsfunktionen getStartTag(), getEndTag() und getValue() liefern die entsprechenden Werte zurück. Ein Beispiel eines *RDFSentence* ist:

<dc:title> Ortsbasierter Webzugriff </dc:title>

5.5.5.7 RDFDocument

Diese Klasse modelliert ein XML / RDF-Dokument, das Metadaten über eine Webseite enthält. Die interne Variable *source* speichert den gesamten Quelltext (*String*) einer XML / RDF-Datei. Die Variable *sentences* ist ein *Vektor,* der alle relevanten *RDF-Sätze* enthält.

Die Funktionen getSource(), setSource() oder getSentences() erlauben den bequemen Zugriff auf die entsprechenden Daten.

5.5.5.8 RDFParser

Der *RDFParser* speichert den Quelltext der XML / RDF-Datei in ein *RDFDocument*-Objekt ab. Die Syntax der RDF-Beschreibung überprüft die Methode checkRDFSyntax(...). Wenn es sich um eine gültige Beschreibung handelt, dann speichert die Methode extractSentences(...) alle *RDF-Sätze* in dem *RDFDocument* ab. Zusätzlich erfolgt eine Überprüfung der Werte (*value*) für jeden zu speichernden *RDF-Satz*. Der Wert selber darf kein *RDF-Satz* sein. Das würde auf eine Beschreibung mit *Qualified Dublin Core* hindeuten. Es existiert keine Spezifikation für die Verwendung von *Qualified Dublin Core* in *RDF,* deswegen wird es von *DCbot* nicht unterstüzt.

5.5.5.9 ISO8601Date

Das Format, welches *DCbot* zur Darstellung von Datumsangaben verwendet, entspricht dem internationalen Standard *ISO8601*. Es definiert die numerische Notation *YYYY-MM-TT* für die Datumsangabe, wobei *YYYY* die vierstellige Jahreszahl, *MM* die zweistellige Monatszahl und *TT* den Tag darstellt. Die Klasse *ISO8601Date* implementiert ein Datumsobjekt in dem entsprechenden Format. Nach Angaben des HTTP 1.1-Protokols [Fie97] sollen HTTP-Applikationen die folgenden drei Formate unterstützen:

- Sun, 06 Nov 1994 08:49:37 GMT; RFC 1123 [Bra98]
- Sunday, 06-Nov-94 08:49:37 GMT; RFC 1036 [HA87]
- Sun Nov 6 08:49:37 1994; ANSI C asctime() Format

Aus diesem Grund stellt die Klasse *ISO8601Date* Methoden zur Konversion aller drei Formate in das *ISO8601* Format bereit. Stößt *DCbot* während der Suche auf ein unbekanntes Format, dann wird eine entsprechende Fehlermeldung ausgegeben.

5.5.5.10 WGS84Point

Diese Klasse modelliert einen Punkt in dem *WGS84* Format, das für die Positionsangabe in der NEXUS Infrastruktur verwendet wird. Der Punkt besteht aus einer x-Koordinate und aus einer y-Koordinate. Die x-Koordinate steht für die Entfernung (*Breite* in Dezimaldarstellung) vom Äquator. Für die Dezimaldarstellung wird die nördliche Richtung als positiv und die südliche Richtung als negativ definiert. Die y-Koordinate bezeichnet die Entfernung (*Länge* in Dezimaldarstellung) des Punktes vom Null-Meridian. Die östliche Richtung ist positiv, während die westliche Richtung durch eine negative Zahl dargestellt wird. Die allgemeine Syntax eines Punktes ist:

Point (*x-Koordinate y-Koordinate*)

Die Fakultät Informatik der Universität Stuttgart hat zum Beispiel die Position:

Point (48.716667 9.116667)

Ein Objekt der Klasse *WGS84Point* hat drei Felder:

- *x*: die x-Koordinate eines Punktes in Dezimaldarstellung
- *y*: die y-Koordinate eines Punktes in Dezimaldarstellung
- *valid*: dieses Feld gibt an ob das Objekt gültig ist; der Wert ist entweder *true* oder *false*

Die Methoden getXCoordinate() und getYCoordinate() liefern die entsprechenden Koordinaten eines Objektes zurück, während die Methode isValid() überprüft, ob ein Objekt gültige Werte besitzt.

Der wichtigste Konstruktor der Klasse, der unter anderem die Umwandlung aus anderen Formaten vornimmt, ist:

WGS84Point(String *string*)

Dieser Konstruktor erwartet einen String der die folgenden Formate haben darf, damit er in ein gültiges *WGS84Point*-Objekt umgewandelt werden kann:

- Point(*fließkommazahl*, *fließkommazahl*)

  Dies ist die Repräsentation eines Punktes, wie sie in der NEXUS-Infrastruktur verwendet wird. Selbst wenn die Positionsangabe bereits im richtigen Format vorliegt, muss dies erkannt werden.

- Lat: *Grade Minuten N | S* Long: *Grade Minuten E | W*

  Wenn die Poitionsangabe Grade und Minuten statt Dezimalzahlen für die Darstellung der *Länge* und der *Breite* verwendet, dann sollte dieses Format vorliegen. Die *Breite* (*lat: latitude*) kann eine nördliche oder eine südliche (*N* oder *S*) Richtung haben.

Die *Länge* (*long: longitude*) bezeichnet eine östliche oder eine westliche Richtung (*E* oder *W*). Ein Beispiel wäre: “Lat: 39 45 N Long: 23 12 W”

5.5.5.11 DBI_JDBC

Die *DBI_JDBC*-Klasse ist eine der beiden Schnittstellen, die *DCbot* für die Abspeicherung gesammelter Metadaten verwenden kann. Sie implementiert das Datenbank-*Interface* (*DBI*) von *DCbot*. Mittels *DBI_JDBC* können alle Datenbanksysteme angesprochen werden, die einen *JDBC*-Treiber besitzen. Die wichtigsten Methoden sind die folgenden:

- *haveConnection()*: Die Verbindung zur Datenbank wird überprüft.
- *createTable()*: Die Datenbanktabelle für die Abspeicherung von Metadaten wird angelegt. Dies geschieht nur einmal, wenn *DCbot* mit dem *init*-Parameter aufgerufen wird.
- *set(String primarykey, Vector elements)*: Fügt eine neue Zeile in die Datenbanktabelle ein.
- *get(String primarykey)*: Liest eine Zeile der Datenbanktabelle.
- *delete(String primarykey)*: Löscht eine Zeile der Datenbanktabelle.
- *dropTable()*: Die angelegte Datenbanktabelle wird gelöscht.
- *close()*: Die Verbindung zur Datenbank wird geschlossen.

5.5.5.12 DBI_SpaSe

Diese Klasse soll die Speicherungsschnittstelle von *DCbot* zu einem *Spatial Model Server* implementieren. Wegen unvorhersehbaren Schwierigkeiten des *Spatial Model Server*-Prorotyps war bis zum Abgabetermin dieser Studienarbeit die Implementierung nicht möglich. Eine spätere Fertigstellung wurde in Aussicht gestellt, deswegen erfolgt hier die Angabe der wichtigsten zu implementierenden Methoden.

Der *Spatial Model Server* kann mittels dem *Augmented World Query Language (AWQL)* angesprochen werden. Die Hauptaufgabe besteht also darin, Funktionen die *DCbot* benötigt, auf entsprechende Anfragen in *AWQL* abzubilden bzw. die Antworten des *Spatial Model Servers* zu parsen und die benötigte Information *DCbot* bereitzustellen.

Die wichtigsten Methoden, die das Datenbank-Interface (*DBI*) verlangt, und wie jede Schnittstelle auch *DBI_SpaSe* bereitstellen muss, sind nachfolgend aufgelistet:

- *haveConnection()*: Die Verbindung zum *Spatial Model Server* wird überprüft. Im Falle eines Fehlers wird eine entsprechende Fehlermeldung ausgegeben.

- *set(String primarykey, Vector elements)*: Es wird überprüft, ob für die aktuelle Position ein entsprechendes *VirtualWebPortal* existiert. Ist dies nicht der Fall, dann wird ein neues *VirtualWebPortal* angelegt. Die gefundenen Metadaten werden in ein *WebPage*-Objekt gespeichert.
- *get(String primarykey)*: Eine Webseite (*WebPage*-Objekt) wird aus dem *Spatial Moder Server* geholt.
- *delete(String primarykey)*: Löscht eine Webseite (*WebPage*-Objekt) aus dem *Spatial Model Server.*
- *dropTable()*: Alle *VirtualWebPortal* und *WebPage* werden aus dem *Spatial Model Server* entfernt.
- *close()*: Die Verbindung zum *Spatial Model Server* wird geschlossen.

5.5.5.13 Konfigurationsparameter: MaxLinkStorage

Die Konfigurationseinstellung *MaxLinkStorage* ermöglicht die Beschränkung des URL-Puffers von *DCbot* auf einen bestimmten Wert. Diese Einstellungsmöglichkeit wurde eingeführt, um das *Crawling*-Verhalten von *DCbot* stärker beeinflussen zu können. Ohne die Beschränkung des URL-Puffers entfernt sich *DCbot* nur sehr langsam von dem Start-URL. Dies wird durch das der Breitensuche ähnliche Suchverhalten verursacht.

Während erster Testläufe hat sich herausgestellt, dass jede Webseite im Durchschnitt fünf Verweise enthält. Damit lässt sich die Anzahl n der Webseiten in Entfernung d vom Start-URL aus betrachtet durch die einfache Formel

$$n = 5^d$$

abschätzen. Die Summe s aller Webseiten in einer Entfernung d vom Start-URL aus betrachtet, beträgt

$$s = \sum_{n=1}^{d} 5^n$$

während im Falle einer unbeschränkten Breitensuche die durchschnittliche Entfernung von Webseiten vom Start-URL aus betrachtet mit der Formel

$$d_{durch} = \frac{\sum_{n=1}^{d} 5^n \cdot n}{\sum_{n=1}^{d} 5^n}$$

berechnet werden kann.

Soll *DCbot* alle Webseiten bis zur Entfernung *d=6* von einem Start-URL aus untersuchen, dann muss er bereits rund 20000 Webseiten betrachten. Die Breitensuche eignet sich also eher um die nähere “Umgebung” eines Start-URLs zu erkunden.

Möchte man weiter in den World Wide Web vorstoßen, dann empfielt sich die Begrenzung des URL-Puffers (Suchbreite) mittels des *MaxLinkStorage* Konfigurationsparameters. Begrenzt man den Puffer auf *p* URLs, dann werden in jeder Entfernung (*d=1,2,3,...*) ungefähr *p* Webseiten untersucht. Da nicht sichergestellt werden kann, dass auch tatsächlich *p* Webseiten in jeder Entfernung *d* gibt, ist *p* nur ein Richtwert, der im Durchschnitt trotzdem erreicht wird. Wird für *MaxRequests* (Anzahl der zu besuchenden Seiten) die Zahl *m* angegeben, dann kann die maximale Entfernung , in die *DCbot* vorstoßen wird, mit der Formel

$$d_{max} = \frac{m}{p}$$

abgeschätzt werden. Wegen der Gleichverteilung beträgt die durchschnittliche Entfernung einer Webseite vom Start-URL aus betrachtet

$$d_{durch} = \frac{d_{max}}{2}$$

Verweise.

5.5.5.14 Ausschluß von DCbot von Webservern

Seit 1994 existiert der *Standard for Robot Exclusion* [Kos94], der den Administratoren von Webservern die Möglichkeit bietet, Webrobotern mitzuteilen, in welchen Bereichen eines Webservers sie unerwünscht sind. Dies geschieht durch die Angabe von URL-Bereichen in einer Datei namens *robots.txt* im Root-Verzeichnis des Webservers. Ein Beispiel *robots.txt*-Datei könnte so aussehen:

```
User-agent: *
Disallow: /mitarbeiter/finanzdaten/creditcardnumbers.txt
Disallow: /chef/
```

Diese Zeilen teilen Robotern mit, dass sie die Datei */mitarbeiter/finanzdaten/creditcardnumbers.txt* und alle Dateien oder Verzeichnisse, die unter */chef/* zu finden sind, nicht untersuchen dürfen.

Eine alternative Möglichkeit, um Webrobotern den Zugriff auf einzelne Dateien zu verbieten, ist die Angabe von speziellen META-Tags in der HTML-Quellcode einer Webseite. Dieses META-Tag muss das folgende Format haben:

```
<META name="robots" content=Befehl>
```

Für das *Befehl*-Feld sind folgende Werte erlaubt:

- *noindex*: Der Webroboter darf das HTML-Dokument nicht untersuchen.
- *nofollow*: Der Webroboter darf keine Verweise des HTML-Dokumentes verfolgen.
- *none*: Wirkt wie *noindex* und *nofollow* zusammen.

DCbot beachtet während der Suche im World Wide Web sowohl *robots.txt*-Dateien als auch *robots*-META-Tags, und hält sich vollständig an die jeweiligen Regeln.

KAPITEL 6 BEWERTUNG DES SYSTEMS

In diesem Kapitel erfolgt eine Bewertung des implementierten Systems an Hand zweier Testreihen .

6.1 Einleitung

Nach einer Implementierungsphase ist es immer wieder spannend zu testen, ob das Implementierungsziel erreicht wurde. *DCbot* soll im World Wide Web nach Webseiten suchen, die Ortsinformation enthalten. Aus diesem Grund wird *DCbot* nach der Fertigstellung einer ausführlichen Testreihe unterzogen, die als Grundlage für eine anschließende Bewertung dient.

Da der *Spatial Model Server* zum Zeitpunkt der Tests nicht einsatzfähig war, wurden gefundene Metadaten mittels der *DBI_JDBC* Schnittstelle von *DCbot* in eine *DB2* Datenbank abgespeichert. Dieses Vorgehen eröffnete die Möglichkeit, die Testdaten bequem per *SQL*-Anfragen untersuchen zu können.

6.2 Erste Testreihe

Um einen Überblick über die Verbreitung des meistverwendeten Metadatenstandards (*DCMES*) zu gewinnen, wird *DCbot* während den ersten drei Testläufen nach alle 15 *Dublin Core* Elementen suchen. Zusätzlich wird das Metadatenelement *DC.Coverage.Spatial*, das als Verfeinerung von *DC.Coverage* auch Ortsinformation enthalten kann, in die Liste der gesuchten Metadatenelemente aufgenommen. Die Filterung einzelner Metadaten (siehe filterAllTags() Kapitel 5.5.5.4) wird abgeschaltet. So erhält man Webseiten, die mindestens ein *Dublin Core* Metadatenelement in einem META-Tag oder in der RDF-Beschreibung enthalten. Die Inhalte der Metadatenelemente spielen dabei erstmal keine Rolle.

Anhang B enthält die komplette *map*-Datei, die für die ersten drei Testläufe verwendet wird und die oben erwähnten 16 Metadatenelemente verwendet

6.2.1 Erster Testlauf

Als Startseite bietet sich für den ersten Testlauf eine Webseite des *Dublin Core Metadata Initiative* an, die Links zu Organisationen, die *Dublin Core* Elemente bereits verwenden, angibt. So können möglichst viele Webseiten mit *Dublin Core* Metadatenelementen gefunden werden.

6.2.1.1 Konfigurationseinstellungen

Die wichtigsten Konfigurationseinstellungen für den ersten Testlauf in der *DCbot.config* Datei sind:

```
ServerMode      multi
MaxDepth        25
MaxRequests     25000
MaxLinkStorage  25000
```

DCbot soll also 25000 Webseiten untersuchen, die domainunabhängig in der näheren Umgebung von *www.dublincore.org* liegen. Für die Entfernung (*MaxDepth*) wird eine so große Zahl gewählt, damit diese Einstellung *DCbot* bei der Suche nicht einschränkt. Da *DCbot* schätzungsweise bis zur Entfernung *d=6* bereits

$$s = \sum_{n=1}^{6} 5^n = 5 + 25 + 125 + 625 + 3125 + 15625 = 19530$$

Webseiten untersuchen muss, wird er höchstens bis zur Entfernung *d=7* vorstoßen können. Der theoretische Wert für die durchschnittliche Entfernung von Webseiten vom Start-URL aus beträgt damit

$$d_{durch} = \frac{\sum_{n=1}^{6} 5^n \cdot n}{\sum_{n=1}^{6} 5^n} = \frac{5 + 50 + 375 + 2500 + 15625 + 93750}{5 + 25 + 125 + 625 + 3125 + 15625} = \frac{112305}{19530} = 5,75$$

Die komplette Konfigurationsdatei und *map*-Datei für diesen Durchlauf sind im Anhang B zu finden.

Der Testlauf wird mit den folgenden Aufrufen gestartet:

```
dcbot init
dcbot http://www.dublincore.org/news/adoption/
```

DCbot, aufgerufen mit der *init*-Option, initialisiert die Datenbankverbindung, in dem er die nötige Datenbanktabelle zur Abspeicherung der Metadaten anlegt. Der Aufruf von *DCbot* mit dem *URL* startet den Suchvorgang.

6.2.1.2 Ergebnisse

Die wichtigsten Kennzahlen des ersten Testlaufs sind in der Tabelle 6-1 zusammengefaßt. Die vollständige Liste der Statusmeldungen und ihren Vorkommenshäufigkeiten, die *DCbot* während der Suche ausgibt, sind im Anhang B zu finden.

Status	Webseitenanzahl	Anteil
Unerreichbar	2156	8,6 %
Kein HTML Dokument	1337	5,4 %
Kein Zugriff	1006	4,0 %
HTML ohne Metadaten	17996	72,0 %
HTML mit Metadaten	2505	10,0 %

Tabelle 6-1: Kennzahlen des ersten Testlaufs

Die durchschnittliche Entfernung der Webseiten, vom Start-URL aus betrachtet, betrug 5,04 Verweise, was in der Nähe des geschätzten Wertes (5,75 Verweise) lag. Von den 25000 untersuchten URLs waren 8,6 Prozent ungültig, d.h. der *Host*-Rechner war dem *DNS*-Server unbekannt oder die referenzierte Webseite existierte nicht. Weitere 5,4 Prozent der URLs referenzierten kein HTML-Dokument.

Bei 86 Prozent der URLs handelte es sich um gültige HTML-Dokumente, von denen 4,7 Prozent eine Untersuchung durch ein entsprechendes *robots*-META-Tag oder eine *robots.txt*-Datei ausschließen (Kein Zugriff). Von den übrig gebliebenen 20501 Webseiten, die ordnungsgemäß untersucht werden konnten, enthielten 12,2 Prozent *Dublin Core* Metadaten. Die meisten Metadaten (88,3 %) wurden in META-Tags gefunden, während nur wenige Webseiten (11,7 %) eine beschreibende XML / RDF-Datei besaßen.

Nachdem über zehn Prozent aller Webseiten *Dublin Core* Metadaten enthielten, stellt sich die spannende Frage, welche Metadatenelemente oft benutzt werden und ob Ortsinformation unter den Metadaten zu finden war. Die meistverwendeten Metadatenelemente waren bei diesem Testlauf *DC.Title* (86,8 %), *DC.Creator* (68,9 %) und *DC.Subject* (68,4 %); selten wurden die Elemente DC.Source (1,5 %), DC.Relation (1,8 %) angegeben. Die Vorkommenshäufigkeiten aller *Dublin Core* Metadatenelemente sind im Anhang B aufgelistet.

Ortsinformation in Webseiten war in diesem Lauf ebenfalls selten zu finden; 2,9 Prozent der Webseiten mit Metadaten enthielten einen *DC.Coverage* Element, während nur 1,1 Prozent ein *DC.Coverage.Spatial* Element verwendeten. Enttäuschend war die Tatsache, dass alle Ortsangaben Namen in natürlicher Sprache für die Bezeichnung einer Position verwendeten, die in allen Fällen sehr ungenau waren. Typische Beispiele sind “*Australia*” oder “*United Kingdom*”.

6.2.2 Zweiter Testlauf

Beim ersten Testlauf wurden nur Webseiten in der "Nähe" der Domain *www.dublincore.org* untersucht. Um die Verbreitung von Metadaten in einem größeren Bereich domainunabhängig zu untersuchen, wird beim zweiten Testlauf *DCbot* mit der Webseite des amerikanischen Nachrichtensenders CNN gestartet.

6.2.2.1 Konfigurationseinstellungen

Der Hauptunterschied zwischen dem ersten und dem zweiten Testlauf liegt in der Angabe eines URL-Puffers (*MaxLinkStorage*) für den zweiten Lauf. Eine Beschränkung der Suchentfernung (*MaxDepth*) sollte durch die Angabe eines hohen Wertes ausgeschlossen werden. Die wichtigsten Einstellungen in der Konfigurationsdatei sind damit:

```
ServerMode      multi
MaxDepth        250
MaxRequests     25000
MaxLinkStorage  250
```

Die Anzahl der zu untersuchenden Webseiten beträgt wieder 25000, wobei in jeder Entfernung nur 250 Webseiten angeschaut werden sollen. Die erwartete maximale Entfernung, in die DCbot vorstoßen soll beträgt

$$d_{max} = \frac{m}{p} = \frac{25000}{250} = 100$$

Verweise. Damit liegt die geschätzte durchschnittliche Entfernung der untersuchten Webseiten bei

$$d_{durch} = \frac{d_{max}}{2} = \frac{100}{2} = 50$$

Verweisen. Die hohe Entfernung der zu untersuchenden Webseiten vom Start-URL und die vielfältige Themenauswahl der Startseite sollte eine zuverlässige Einschätzung der Verbreitung von *Dublin Core* Metadaten im World Wide Web ermöglichen.

Der Testlauf wird mit den folgenden Aufrufen gestartet:

```
dcbot init
dcbot http://www.cnn.com/
```

6.2.2.2 Ergebnisse

Um den Vergleich mit dem ersten Testlauf zu ermöglichen werden in der Tabelle 6-2 die gleichen charakteristischen Kennzahlen für die aktuelle Suche angegeben. Die vollständige Liste der Statusmeldungen und der dazugehörigen Vorkommenshäufigkeiten sind im Anhang B zu finden.

Status	Webseitenanzahl	Anteil
Unerreichbar	1565	6,3 %
Kein HTML Dokument	644	2,6 %
Kein Zugriff	661	2,6 %
HTML ohne Metadaten	21653	86,6 %
HTML mit Metadaten	477	1,9 %

Tabelle 6-2: Kennzahlen des zweiten Testlaufs

Die durchschnittliche Entfernung der Webseiten, vom Start-URL aus betrachtet, betrug 55,8 Verweise, was verglichen mit dem Zielwert (50 Verweise) durchaus zufrieden stellend war. Von den 25000 untersuchten URLs waren 6,3 Prozent ungültig und der Anteil von nicht-HTML-Dokumenten betrug 2,5 Prozent.

Bei 91,2 Prozent der URLs handelte es sich um gültige HTML-Dokumente, von denen 2,9 Prozent eine Untersuchung durch einen Webroboter ausschließen. Insgesamt konnten 22130 Webseiten ordnungsgemäß untersucht werden und 2,2 Prozent von ihnen enthielten *Dublin Core* Metadaten. Die deutliche Abnahme der Vorkommenshäufigkeit von Metadaten verglichen mit dem ersten Testlauf lässt vermuten, dass die Ergebnisse des ersten Testlaufs durch die themenspezifische Startseite (www.dublincore.org) stark beeinflußt wurden.

Von den wenigen, im zweiten Lauf gefundenen Webseiten die Metadaten enthielten, benutzten 98,5 Prozent META-Tags und nur 1,5 Prozent eine XML/RDF-Datei zur Angabe von Metadaten. Häufig verwendete Metadatenelemente waren *DC.Title* (99,7 %), *DC.Description* (93,1 %) und *DC.Language* (87,6 %), während *DC.Source* (2,3 %) und *DC.Relation* (1,3 %) wieder selten auftraten.

Ähnlich wie beim ersten Testlauf wurde nur in wenigen Webseiten Ortsinformation gefunden; das Metadatenelement *DC.Coverage* trat in 2,7 Prozent der Fälle auf, während *DC.Coverage.Spatial* in 0,8 Prozent der Webseiten mit Metadaten vorkam. Für die Ortsangabe wurden wieder ausschließlich Namen in natürlicher Sprache verwendet.

6.2.3 Dritter Testlauf

Beim dritten Testlauf wurde die Verbreitung von *Dublin Core* Metadaten unter den Webseiten der Informatik Fakultät der Universität Stuttgart untersucht. Als Startseite wurde die Einstiegsseite der Fakultät gewählt.

6.2.3.1 Konfiguration

Nur Webseiten unter der Domäne *www.informatik.uni-stuttgart.de* sollten untersucht werden; deswegen wurde die entsprechende Beschränkung für den *ServerMode* von *DCbot* eingestellt. Die wichtigsten Zeilen der *DCbot* Konfigurationsdatei, die im Anhang B komplett angegeben wird, sind:

```
ServerMode          multi:www.informatik.uni-stuttgart.de
MaxDepth            50
MaxRequests         25000
MaxLinkStorage      25000
```

Die Suche wurde mit den folgenden Befehlen gestartet:

```
dcbot init
dcbot http://www.informatik.uni-stuttgart.de/
```

6.2.3.2 Ergebnisse

DCbot hat insgesamt 20899 URLs unter der Domäne *www.informatik.uni-stuttgart.de* gefunden und untersucht. Die wichtigsten Kennzahlen für diesen Testlauf sind in Tabelle 6-3 dargestellt.

Status	Webseitenanzahl	Anteil
Unerreichbar	1588	7,6 %
Kein HTML Dokument	6035	28,9 %
Kein Zugriff	178	0,9 %
HTML ohne Metadaten	13097	62,7 %
HTML mit Metadaten	1	0,005 %

Tabelle 6-3: Kennzahlen des dritten Testlaufs

Die durchschnittliche Entfernung der untersuchten Webseiten von der Starseite aus betrug 6,31 Verweise, während die größte Entfernung (48 Verweise) zeigt, dass die *MaxDepth*-Einstellung für diesen Lauf (50 Verweise) sehr knapp bemessen war.

Wie Tabelle 6-3 zeigt, waren 7,6 Prozent der URLs unerreichbar, und 28,9 Prozent zeigten auf kein HTML-Dokument. Die meisten URLs (62,7 %) referenzierten HTML-Dokumente von denen nur ein einziges Metadaten enthielt, was die Analyse der Vorkommenshäufigkeiten von verschiedenen Metadatenelementen überflüssig macht.

6.3 Zweite Testreihe

Während der ersten Testreihe hat *DCbot* nach allen *Simple Dublin Core* Metadatenelementen gesucht. Sein eigentlicher Aufgabenbereich ist allerdings die Suche nach Webseiten mit Ortsinformation. Sollte *DCbot* Metadaten nur über Webseiten abspeichern, die eine Ortsinformation enthalten, dann lässt sich das bequem durch die folgende Zeile in der *DCbot* Konfigurationsdatei einstellen:

```
RequiredMetadata        DC.Coverage:*
```

So konfiguriert, beachtet *DCbot* nur Webseiten, die das Metadatenelement *DC.Coverage* enthalten. Er akzeptiert standardmäßig nur diejenigen Positionsdaten für dieses Metadatenelement, die entweder einer Positionsangabe in *WGS84* Format entsprechen, oder in einem konvertierbaren Format vorliegen (siehe Kapitel 5.5.5.10).

Während des ersten Testlaufs wurden leider keine entsprechenden Webseiten gefunden. Aus diesem Grund wird *DCbot* mit eigenhändig erstellten Daten getestet.

6.3.1 Testdaten

Die Testdaten bestehen aus drei HTML-Dateien und einer XML/RDF-Beschreibung; die gesamten Daten sind im Anhang C enthalten. Sie wurden als Homepage dreier Figuren aus dem Cartoon “The Simpsons” gestaltet und sind zusätzlich unter http://www.suetoe.de/ zu finden. Die einzelnen HTML-Dateien enthalten jeweils Verweise aufeinander; damit kann jede der Seiten als Startseite gewählt werden, so dass alle Seiten für *DCbot* zu finden sind.

6.3.2 Konfigurationseinstellungen

Die Komplette *DCbot map*-Datei und Konfigurationsdatei sind im Anhang C abgebildet. Da es sich nur um drei Testseiten handelt, sind die folgenden Einstellungen völlig ausreichend:

```
ServerMode          multi
MaxDepth            5
MaxRequests         10
```

```
MaxLinkStorage          10
```

Die *map*-Datei enthält nur diejenigen Metadatenelemente, die im Nexus Kontext relevant sind (siehe Kapitel 4.3).

6.3.3 Erster Testlauf

Nach der Initialisierung wird DCbot mit der ersten Testseite aufgerufen:

```
dcbot init
dcbot http://www.suetoe.de/testseite1.html
```

Da alle Testseiten Verweise aufeinander enthalten, hat *DCbot* alle drei Webseiten gefunden und untersucht. Wie die Statusmeldungen *OK/METADATA* zeigen, fand *DCbot* zu den ersten zwei Testseiten gültige Positionsangaben (siehe Kapitel C.3). Die dritte Testseite enthielt ungültige Positionsdaten, deswegen gab *DCbot* die Statusmeldung *OK* aus, und speicherte keine Metadaten ab. Die Fehlermeldungen

```
WGS84Point: Warning, cannot parse "Springfield" - unknown position format!
ISO8601Date: Warning, cannot parse "Yesterday" - unknown date format!
```

dokumentieren, dass *DCbot* während der Untersuchung der dritten Testseite auf ungültige Datums- und Positionsangaben stieß.

Die abgespeicherten Metadaten (siehe Kapitel C.3) zeigen, dass DCbot sowohl in den META-Tags der ersten Testseite, als auch in der RDF-Beschreibung der zweiten Testseite erfolgreich Positionsangaben gefunden hat. Die Positionsangabe der ersten Testseite,

```
<meta name="dc.coverage" content="Lat: 48 49 N Long: 9 0 E">
```

wurde erfolgreich in den folgenden *WGS84* Punkt umgewandelt:

```
POINT (48.81666666666667 9.0)
```

Die Positionsangabe der zweiten Testseite,

```
<dc:coverage>POINT (36.08336 -115.16667)</dc:coverage>
```

wurde ebenfalls erfolgreich erkannt und als *WGS84* Punkt übernommen:

```
POINT (36.08336 -115.16667)
```

6.3.4 Zeiter Testlauf

Beim zweiten Testlauf wurde die gleiche *DCbot map*-Datei wie beim ersten Testlauf verwendet, während in der *DCbot* Konfigurationsdatei die *RequiredMetadata* Option auskommentiert wurde:

```
#RequiredMetadata
```

Durch diese Einstellung ist *DCbot* dazu angewiesen, nicht nur Webseiten mit einem speziellen Metadatenelement (wie *DC.Coverage*), sondern auch Webseiten mit einem beliebigen Metadatenelement aus der *DCbot map*-Datei abzuspeichern. Der Testlauf wurde wieder mit der folgenden Zeile gestartet:

```
dcbot http://www.suetoe.de/testseite1.html
```

Wie erwartet hat *DCbot* die ersten zwei Webseiten wieder erfolgreich gefunden und die entsprechenden Metadaten gespeichert. Die Statusmeldungen (siehe Kapitel C.3) *OK/ UNCHANGED* dokumentieren, dass diese Seiten bereits in der Datenbank vorhanden sind, und dass sie unverändert vorliegen.

Zusätzlich wurden in diesem Testlauf auch Metadaten in der dritten Testseite gefunden (siehe Kapitel C.3). Die Fehlermeldungen, des ersten Testlaufes,

```
WGS84Point: Warning, cannot parse "Springfield" - unknown position format!
ISO8601Date: Warning, cannot parse "Yesterday" - unknown date format!
```

treten wider auf; die Datumsangabe “Yesterday” kann genauso wenig wie die Positionsangabe “Springfield” richtig interpretiert werden.

Der eigentliche Zweck des zweiten Testlaufs ist die Demonstration der Fähigkeit von *DCbot*, mehrfach auftretende Metadatenelemente zu gruppieren (siehe Kapitel 5.5.3.3). Für die dritte Testseite werden drei Athoren mittels des *DC.Creator* Metadatenelementes in META-Tags angegeben:

```
<meta name="dc.creator" content="Bart Simpson">
<meta name="dc.creator" content="Lisa Simpson">
<meta name="dc.creator" content="Maggie Simpson">
```

Die Inhalte dieser META-Tags wurden erfolgreich zu einem Eintrag gruppiert und in der Datenbank abgespeichert:

```
dcCreator:          Bart Simpson; Lisa Simpson; Maggie Simpson
```

6.4 Zusammenfassung

Die beiden durchgeführten Testreihen belegen, dass *DCbot* sowohl in der Lage ist, mehrere zehntausend Webseiten zu untersuchen, als auch die erwünschten Metadaten abzuspeichern. Die Angabe eines speziellen Metadatenelementes mittels der *RequiredMetadata* Option in der *DCbot*-Konfigurationsdatei ermöglicht die Auswahl von Webseiten, die ein spezielles Metadatenelement (z.B. *DC.Coverage* für Ortsangaben) enthalten müssen. Damit ist *DCbot* in der Lage, gezielt nach Webseiten mit Ortsinformation zu suchen und gefundene Daten abzuspeichern; Datums- und Positionsangaben werden dabei, wenn möglich in die richtigen Formate konvertiert. Schlägt die Umwandlung fehl, dann werden keine Metadaten über die entsprechende Webseite gespeichert; damit ist sichergestellt, dass nach einem Suchvorgang nur Daten mit gültigen Positionsangaben in der Datenbank bzw. im *Spatial Model Server* vorhanden sind.

KAPITEL 7 ZUSAMMENFASSUNG

Dieses Kapitel enthält eine kurze Zuammenfassung und Empfehlungen zur künftigen Erweiterung der implementierten Softwarelösung.

Als Ergebnis der Untersuchung von Metadatenstandards für Webseiten lässt sich feststellen, dass der einzige Standard, der auch die Angabe von Ortsinformation vorsieht, das *Dublin Core Metadata Element Set* ist. Metadaten im World Wide Web sollten eine schnelle und automatisierte Suche von Ressourcen erleichtern. Leider besteht die Möglichkeit irreführende Metadaten über Ressourcen anzugeben, was die Tatsache mit sich bringt, dass die meisten Suchmaschinen und Suchwerkzeuge bei der Indexierung Metadaten keine Beachtung schenken. Ein weiterer Grund für die geringe Verbreitung von Metadaten im World Wide Web, ist der zusätzliche Aufwand, der bei der Webseitengestaltung mit Metadatenangabe, entsteht.

Während der Testphase von *DCbot* wurden 25000 URLs ausgehend von der Startseite des amerikanischen Nachrichtensenders CNN untersucht. Von 22130 Webseiten, die ordnungsgemäß analysiert wurden, enthielten lediglich 477 Webseiten Metadaten, wobei in 17 Fällen auch Ortsinformation angegeben wurde; dies entspricht 0,08 Prozent der untersuchten Webseiten. Alle gefundenen Ortsangaben waren in natürlicher Sprache Verfasst und äußerst ungenau (z.B. *Australia*), was eine Umwandlung in genaue Positionsangaben unmöglich macht. Im Endeffekt ist der implementierte Webroboter (*DCbot*) zwar wie durch Tests gezeigt, in der Lage, seine Aufgabe vollständig zu erfüllen; jedoch bringt eine Suche im World Wide Web praktisch keine brauchbaren Ergebnisse.

Leider war während der Implementierungsphase der *Spatial Model Server*-Prototyp nicht lauffähig, was die vollständige Realisierung einer entsprechenden Schnittstelle verhinderte. Im Kapitel 5.5.5.12 wurden der Rahmen sowie die wichtigsten Methoden einer Schnittstelle zum *Spatial Model Server* vorgestellt. Wahlweise kann der implementierte Webroboter eine herkömmliche Datenbank für die Speicherung verwenden, und ist damit vollkommen lauffähig.

Für die künftige Entwicklung von *DCbot* könnten, neben der Implementierung einer Schnittstelle zum *Spatial Model Server,* zwei weitere Erweiterungsmöglichkeiten in Frage kommen. Eine Möglichkeit für die Verbesserung des Webroboters besteht in der Realisierung einer graphischen Oberfläche, die eine bequemere Verwaltung der Konfigurationsoptionen und eine ansprechende Darstellung von Rückmeldungen realisieren könnte. Eine weitere Möglichkeit stellt die Beobachtung der zukünftigen Verbreitung von Metadaten und die Anpassung des Webroboters an neue Metadatenformate dar, um ein erfolgreiches Sammeln von Webseiten mit Ortsinformation künftig zu ermöglichen.

ANHANG A METADATENBEISPIELE

In diesem Anhang werden Beispiele für die Angabe von Metadaten über Webseiten dargestellt.

A.1 META-Tags

META-Tags einer Webseite, die die HTML-Version dieser Studienarbeit beschreiben, sind nachfolgend dargestellt.

```
<HTML>

   <HEAD>

      <META name="DC.Title" content="Ortsbasierter Webzugriff" lang="de">
      <META name="DC.Creator" content="Mihály Sütö">
      <META name="DC.Subject" content="Metadaten, Suchmaschinen, Webroboter">
      <META name="DC.Description" content="Diese Studienarbeit beschäftigt sich mit
         den Möglichkeiten der automatisierten Suche von Webseiten mit
         Ortsinformation und ihre Einbindung in die Nexus Infrastruktur." lang="de">
      <META name="DC.Date.Available" content="30-04-2002" scheme="ISO8601">
      <META name="DC.Date.Modified" content="29-04-2002" scheme="ISO8601">
      <META name="DC.Identifier" content="http://www.suetoe.de/studienarbeit.html">
      <META name="DC.Type" content="Studienarbeit" lang="de">
      <META name="DC.Format" content="text/html">
      <META name="DC.Language" content="de">
      <META name="DC.Rights" content="Copyright (c) Mihály Sütö. Alle Rechte
         vorbehalten." lang="de">

   </HEAD>

   <BODY>

      ...

   </BODY>

</HTML>
```

A.2 RDF-Beschreibung

Ein XML / RDF Dokument, das die HTML-Version dieser Studienarbeit beschreibt, wird nachfolgend dargestellt.

```
<?xml version="1.0"?>
   <rdf:RDF
         xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:dc="http://purl.org/dc/elements/1.1/">
      <rdf:Description rdf:about="http://www.suetoe.de/studienarbeit.html">
         <dc:title> Ortsbasierter Webzugriff </dc:title>
         <dc:creator> Mihály Sütö </dc:creator>
         <dc:subject> Metadaten, Suchmaschinen, Webroboter </dc:creator>
         <dc:description> Diese Studienarbeit beschäftigt sich mit
            den Möglichkeiten der automatisierten Suche von Webseiten
            mit Ortsinformation und ihre Einbindung in die Nexus
            Infrastruktur.</dc:description>
         <dc:date> 2002-04-30 </dc:date>
         <dc:identifier> http://www.suetoe.de/studienarbeit.html </dc:identifier>
         <dc:type> Studienarbeit </dc:type>
         <dc:format> text/html </dc:format>
         <dc:language> de </dc:language>
         <dc:rights> Copyright (c) Mihály Sütö. Alle Rechte vorbehalten.</dc:rights>
      </rdf:Description>
   </rdf:RDF>
```

ANHANG B ERSTE TESTREIHE

In diesem Anhang werden die Konfigurationseinstellungen des Webroboters und die Ergebnisse der ersten Testreihe dargestellt.

B.1 DCbot map-Datei

Die *map*-Datei von *DCbot* (*DCbot.map*), die bei der ersten Testreihe zum Einsatz kam wird nachfolgend dargestellt. Während der drei Läufe dieser Testreihe hat der Webroboter nach allen 15 *Simple Dublin Core* Elementen und nach dem *Qualified Dublin Core* Element *DC.Coverage.Spatial* gesucht. Mehrfach auftretende *DC.Creator* Werte wurden gruppiert.

```
#
# DCbot map file
#

DC.Title                 title
DC.Creator               creator           group
DC.Subject               subject
DC.Description           description
DC.Publisher             publisher
DC.Contributor           contributor
DC.Date                  date
DC.Type                  type
DC.Format                format
DC.Identifier            identifier
DC.Source                source
DC.Language              language
DC.Relation              relation
DC.Rights                rights
DC.Coverage              coverage
DC.Coverage.Spatial      spatial
```

B.2 DCbot Konfigurationsdateien

Erster Testlauf: Eine Webseite der *Dublin Core Metadata Initiative* als Startseite für *DCbot*.

```
#
# DCbot configuration file
#

Homepage            http://www.suetoe.de
EMail               mihaly@suetoe.de
AcceptLanguage      de,en;q=0.8
ServerMode          multi
RequestRate         5
MaxDepth            50
MaxRequests         25000
MaxLinkStorage      25000
#Proxy
#NoProxy
MapFile             DCbot.map
#RequiredMetadata
DBInterface         DBI_JDBC
DBJDBCDriver        db2
DBJDBCClass         COM.ibm.db2.jdbc.app.DB2Driver
DBName              exercise
DBTable             dublincore
ForceOverwrite      no
```

Der Webroboter wurde mit den folgenden Kommandos aufgerufen:

```
dcbot init
dcbot http://www.dublincore.org/news/adoption/
```

Zweiter Testlauf: Startseite des Nachrichtensenders CNN als Startseite für *DCbot*.

```
#
# DCbot configuration file
#

Homepage            http://www.suetoe.de
EMail               mihaly@suetoe.de
AcceptLanguage      de,en;q=0.8
ServerMode          multi
RequestRate         5
MaxDepth            250
MaxRequests         25000
MaxLinkStorage      250
#Proxy
#NoProxy
MapFile             DCbot.map
#RequiredMetadata
DBInterface         DBI_JDBC
DBJDBCDriver        db2
DBJDBCClass         COM.ibm.db2.jdbc.app.DB2Driver
DBName              exercise
DBTable             cnn
ForceOverwrite      no
```

Der Webroboter wurde mit den folgenden Kommandos aufgerufen:

```
dcbot init
dcbot http://www.cnn.com/
```

Dritter Testlauf: Homepage der Informatik Fakultät der Universität Stuttgart als Startseite für *DCbot*.

```
#
# DCbot configuration file
#

Homepage              http://www.suetoe.de
EMail                 mihaly@suetoe.de
AcceptLanguage        de,en;q=0.8
ServerMode            multi:www.informatik.uni-stuttgart.de
RequestRate           5
MaxDepth              50
MaxRequests           25000
MaxLinkStorage        25000
#Proxy
#NoProxy
MapFile               DCbot.map
#RequiredMetadata
DBInterface           DBI_JDBC
DBJDBCDriver          db2
DBJDBCClass           COM.ibm.db2.jdbc.app.DB2Driver
DBName                exercise
DBTable               uni_informatik
ForceOverwrite        no
```

Der Webroboter wurde mit den folgenden Kommandos aufgerufen:

```
dcbot init
dcbot http://www.informatik.uni-stuttgart.de/
```

B.3 Statusmeldungen

Die Tabellen enthalten die Häufigkeiten einzelner Stautsmeldungen, die während der drei Testläufe auftraten.

Erster Testlauf: Eine Webseite der *Dublin Core Metadata Initiative* als Startseite für *DCbot*.

StatusCode	Webseitenanzahl	Anteil
UNKNOWN HOST	125	0,50 %
FAILURE	2029	8,12 %
ERROR	2	0,01 %
NO ACCESS	1006	4,02 %
NO HTML	1337	5,35 %
OK	17996	71,98 %
OK / METADATA	2505	10,02 %

Tabelle B-1: Statusmeldungen des ersten Testlaufs

Zweiter Testlauf: Startseite des Nachrichtensenders CNN als Startseite für *DCbot*.

StatusCode	Webseitenanzahl	Anteil
UNKNOWN HOST	67	0,27 %
FAILURE	1492	5,97 %
ERROR	6	0,02 %
NO ACCESS	661	2,64 %
NO HTML	644	2,58 %
OK	21653	86,61 %
OK / METADATA	477	1,91 %

Tabelle B-2: Statusmeldungen des zweiten Testlaufs

Dritter Testlauf: Homepage der Informatik Fakultät der Universität Stuttgart als Startseite für *DCbot*.

StatusCode	Webseitenanzahl	Anteil
UNKNOWN HOST	1	0,005 %
FAILURE	1586	7,59 %
ERROR	1	0,005 %
NO ACCESS	178	0,85 %
NO HTML	6035	28,88 %
OK	13097	62,67 %
OK / METADATA	1	0,005 %

Tabelle B-3: Statusmeldungen des dritten Testlaufs

B.4 Metadatenhäufigkeiten

Die Tabellen enthalten die Häufigkeiten einzelner Metadatenelemente, die während der drei Testläufe gefunden wurden. Das Feld “Anteil” zeigt in wieviel Prozent der Webseiten ein bestimmtes Metadatenelement gefunden wurde.

Erster Testlauf: Eine Webseite der *Dublin Core Metadata Initiative* als Startseite für *DCbot*.

Metadatenelement	Webseitenanzahl	Anteil
DC.Title	2174	86,79 %
DC.Creator	1727	68,94 %
DC.Subject	1713	68,38 %
DC.Description	1422	56,77 %
DC.Publisher	1633	65,19 %
DC.Contributor	189	7,54 %
DC.Date	701	27,98 %
DC.Type	237	9,46 %
DC.Format	1065	42,51 %
DC.Identifier	1273	50,82 %
DC.Source	38	1,52 %
DC.Language	1349	53,85 %
DC.Relation	45	1,80 %
DC.Rights	1450	57,88 %
DC.Coverage	73	2,91 %
DC.Coverage.Spatial	27	1,08 %

Tabelle B-4: Metadatenelemente des ersten Testlaufs

Zweiter Testlauf: Startseite des Nachrichtensenders CNN als Startseite für *DCbot*.

Metadatenelement	Webseitenanzahl	Anteil
DC.Title	442	99,66 %
DC.Creator	173	36,27 %
DC.Subject	403	84,49 %
DC.Description	444	93,08 %
DC.Publisher	383	80,29 %
DC.Contributor	54	11,32 %
DC.Date	51	10,69 %
DC.Type	356	74,63 %
DC.Format	369	77,36 %
DC.Identifier	49	10,27 %
DC.Source	11	2,31 %
DC.Language	389	87,55 %
DC.Relation	6	1,26 %
DC.Rights	58	12,16 %
DC.Coverage	13	2,73 %
DC.Coverage.Spatial	4	0,84 %

Tabelle B-5: Metadatenelemente des zweiten Testlaufs

Dritter Testlauf: Homepage der Informatik Fakultät der Universität Stuttgart als Startseite für *DCbot*.

Metadatenelement	Webseitenanzahl	Anteil
DC.Title	0	0 %
DC.Creator	0	0 %
DC.Subject	1	100 %
DC.Description	0	0 %
DC.Publisher	0	0 %
DC.Contributor	0	0 %
DC.Date	1	100 %
DC.Type	1	100 %
DC.Format	1	100 %
DC.Identifier	0	0 %
DC.Source	0	0 %
DC.Language	0	0 %
DC.Relation	1	100 %
DC.Rights	1	100 %
DC.Coverage	0	0 %
DC.Coverage.Spatial	0	0 %

Tabelle B-6: Metadatenelemente des dritten Testlaufs

ANHANG C ZWEITE TESTREIHE

In diesem Anhang werden die Konfigurationseinstellungen des Webroboters und die Ergebnisse der zweiten Testreihe dargestellt.

C.1 Testdaten

Während der zweiten Testreihe wurden folgende HTML-Dokumente und XML/RDF-Dateien mit *DCbot* untersucht.

Erste HTML-**Testseite:** Homer's Homepage

```
<HTML>
  <HEAD>
    <TITLE> Testseite 1 (Homer's Webseite) </TITLE>

    <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
    <meta http-equiv="content-language" content="en">

    <meta name="dc.title" content="Homer's Homepage">
    <meta name="dc.creator" content="Homer Simpson">
    <meta name="dc.subject" content="Homer">
    <meta name="dc.description" content="This page is about Homer Simpson.">
    <meta name="dc.contributor" content="Bart Simpson">
    <meta name="dc.date" content="2002-04-25">
    <meta name="dc.type" content="Homepage">
    <meta name="dc.format" content="text/html">
    <meta name="dc.identifier" content="http://www.suetoe.de/testseite1.html">
    <meta name="dc.language" content="en">
    <meta name="dc.relation" content="http://www.suetoe.de/testseite3.html">
    <meta name="dc.rights" content="Copyright (c) Homer Simpson.
                                    All rights reserved.">
    <meta name="dc.coverage" content="Lat: 48 49 N Long: 9 0 E">
  <BODY>
    <P><FONT face="Arial" size="5">Hi, i am Homer. </FONT></P>
    <P><img border="0" src="homer.gif" width="163" height="257"></P>
    <P><FONT face="Arial" size="5">This is my page.</FONT></P>
    <a href= "testseite2.html"> Link to Marge's Homepage</a> <p>
  </BODY>
</HTML>
```

Zweite HTML-Testseite: Marge's Homepage

```
<HTML>
   <HEAD>
      <TITLE> Testseite 2 (Marge's Homepage) </TITLE>

      <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
      <meta http-equiv="content-language" content="en">
      <link rel="meta" href="testseite2.html.rdf">
   </HEAD>
   <BODY>
      <P><FONT face="Arial" size="5">Hi, i am Marge.</FONT></P>
      <P><img border="0" src="marge.gif" width="90" height="183"></P>
      <P><FONT face="Arial" size="5">This is my page.</FONT></P>
      <a href= "testseite3.html"> Link to Bart's Homepage</a> <p>
   </BODY>
</HTML>
```

XML/RDF-Dokument mit Metadaten für die zweite Testseite:

```
<?xml version="1.0"?>
   <rdf:RDF
         xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:dc="http://purl.org/dc/elements/1.1/">
      <rdf:Description rdf:about="http://www.suetoe.de/testseite2.html">
         <dc:title> Marge's Homepage </dc:title>
         <dc:creator> Marge Simpson </dc:creator>
         <dc:subject> Marge </dc:subject>
         <dc:description> This page is about Marge Simpson. </dc:description>
         <dc:date> 2002-04-21 </dc:date>
         <dc:identifier> http://www.suetoe.de/testseite2.html </dc:identifier>
         <dc:type> Homepage </dc:type>
         <dc:format> text/html </dc:format>
         <dc:language> en </dc:language>
         <dc:rights>Copyright (c) Marge Simpson. All rights reserved.</dc:rights>
         <dc:coverage>POINT (36.08336 -115.16667)</dc:coverage>
      </rdf:Description>
   </rdf:RDF>
```

Dritte HTML-Testseite: Bart's Homepage

```
<HTML>
   <HEAD>
      <TITLE> Testseite 3 (Bart's Homepage) </TITLE>

      <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
      <meta http-equiv="content-language" content="en">

      <meta name="dc.title" content="Bart's Homepage">
      <meta name="dc.creator" content="Bart Simpson">
      <meta name="dc.creator" content="Lisa Simpson">
      <meta name="dc.creator" content="Maggie Simpson">
      <meta name="dc.subject" content="Bart">
      <meta name="dc.description" content="This page is about Bart Simpson.">
      <meta name="dc.date" content="Yesterday">
      <meta name="dc.coverage" content="Springfield">
   </HEAD>
   <BODY>
      <P><FONT face="Arial" size="5">Hi, i am Bart.</FONT></P>
      <P><img border="0" src="bart.gif" width="283" height="283"></P>
      <P><FONT face="Arial" size="5">This is my page.</FONT></P>
      <a href= "testseite1.html">Link to Homer's Homepage</a> <p>
   </BODY>
</HTML>
```

C.2 Konfigurationseinstellungen

DCbot *map*-Datei für die zweite Testreihe. Sie enhält nur Metadatenelemente, die im NEXUS Kontext von Bedeutung sind.

```
#
# DCbot map file
#

DC.Title                dcTtitle
DC.Creator              dcCreator           group
DC.Subject              dcSubject
DC.Description          dcDescription
DC.Publisher            dcPublisher
DC.Date                 dcDate
DC.Coverage             dcCoverage
```

DCbot Konfigurationsdatei für die zweite Testreihe.

```
#
# DCbot configuration file
#

Homepage                http://www.suetoe.de
EMail                   mihaly@suetoe.de
AcceptLanguage          de,en;q=0.8
ServerMode              multi
RequestRate             5
MaxDepth                5
MaxRequests             10
MaxLinkStorage          10
#Proxy
#NoProxy
MapFile                 DCbot.map
RequiredMetadata        DC.Coverage:*
DBInterface             DBI_JDBC
DBJDBCDriver            db2
DBJDBCClass             COM.ibm.db2.jdbc.app.DB2Driver
DBName                  exercise
DBTable                 dcbot
ForceOverwrite          no
```

C.3 Ergebnisse

Komplette Statusmeldungen des ersten Aufrufs:

```
Connecting to database... Ok.
Opening HTTP connection to "www.suetoe.de:80"... OK.
URL: / http://www.suetoe.de/testseite1.html
    \ Status: OK/METADATA, Depth: 0, Checked: 1, Queued: 1)
URL: / http://www.suetoe.de/testseite2.html
    \ Status: OK/METADATA, Depth: 1, Checked: 2, Queued: 1)
WGS84Point: Warning, cannot parse "Springfield" - unknown position format!
ISO8601Date: Warning, cannot parse "Yesterday" - unknown date format!
URL: / http://www.suetoe.de/testseite3.html
    \ Status: OK, Depth: 2, Checked: 3, Queued: 0)
DCbot: Finished. Processed 3 URLs.
Closing database connection... Ok.
```

Gefundene Metadaten zu testseite1.html:

dcTitle:	Homer's Homepage
dcCreator:	Homer Simpson
dcSubject:	Homer
dcDescription:	This page is about Homer Simpson.
dcDate:	2002-04-25
dcCoverage:	POINT (48.81666666666667 9.0)

Gefundene Metadaten zu testseite2.html:

dcTitle:	Marge's Homepage
dcCreator:	Marge Simpson
dcSubject:	Marge
dcDescription:	This page is about Marge Simpson.
dcDate:	2002-04-21
dcCoverage:	POINT (36.08336 -115.16667)

Komplette Statusmeldungen des zweiten Aufrufs:

```
Connecting to database... Ok.
Opening HTTP connection to "www.suetoe.de:80"... OK.
URL: / http://www.suetoe.de/testseite1.html
    \ Status: OK/UNCHANGED, Depth: 0, Checked: 1, Queued: 1)
URL: / http://www.suetoe.de/testseite2.html
    \ Status: OK/UNCHANGED, Depth: 1, Checked: 2, Queued: 1)
WGS84Point: Warning, cannot parse "Springfield" - unknown position format!
ISO8601Date: Warning, cannot parse "Yesterday" - unknown date format!
URL: / http://www.suetoe.de/testseite3.html
    \ Status: OK/METADATA, Depth: 2, Checked: 3, Queued: 0)
DCbot: Finished. Processed 3 URLs.
Closing database connection... Ok.
```

Gefundene Metadaten zu testseite3.html:

dcTitle:	Bart's Homepage
dcCreator:	Bart Simpson; Lisa Simpson; Maggie Simpson
dcSubject:	Bart
dcDescription:	This page is about Bart Simpson.

Literaturverweise

[BC95] Berners-Lee, T.; Conolly, D. : *Hypertext Markup Language - 2.0.* World Wide Web Consortium, 1995, Available from: http://www.ietf.org/rfc/rfc1866.txt

[BFI98] Berners-Lee, T.; Fielding, R.; Irvine, U.C.; Masinter, L.: *Uniform Resource Identifiers (URI): Generic Syntax.* The Internet Society, 1998

[BG00] Brickley D.; Guha, R.V.: *Resource Description Framework (RDF) Schema Specification 1.0.* World Wide Web Consortium, 2000, Available from: http://www.w3.org/TR/2000/CR-rdf-schema-20000327

[BHL99] Bray, T.; Hollander, D.; Layman, A.: *Namespaces in XML.* World Wide Web Consortium, 1999, Available from: http://www.w3.org/TR/1999/REC-xml-names-19990114

[BP98] Brin, S.; Page, L.: T*he Anatomy of a Large-Scale Hypertextual Web Search Engine.* Computer Science Department, Stanford University, 1998

[BPS00] Bray, T.; Paoli, J.; Sperberg-McQueen, C. M.; Maler, E.: *Extensible Markup Language (XML) 1.0 (Second Edition).* World Wide Web Consortium, 2000, Available from: http://www.w3.org/TR/2000/REC-xml-20001006

[Bra89] Braden, R.: *Requirements for Internet Hosts -- Application and Support.* Internet Engineering Task Force, 1989

[BS94] Bearman, D.; Sochats, K.: *Metadata Requirements for Evidence.* 1994, Available from: http://www.archimuse.com/papers/nhprc/BACartic.html

[Cox00a] Cox, S.: *DCMI Box Encoding Scheme: specification of the spatial limits of a place, and methods for encoding this in a text string.* Dublin Core Metadata Initiative, 2000, Available from: http://www.dublincore.org/documents/2000/07/28/dcmi-box/

[Cox00b] Cox, S.: *DCMI Point Encoding Scheme: a point location in space, and methods for encoding this in a text string.* Dublin Core Metadata Initiative, 2000, Available from: http://www.dublincore.org/documents/2000/07/28/dcmi-point/

[Cox00c] Cox, S.: *DCMI Period Encoding Scheme.* Dublin Core Metadata Initiative, 2000 Available from: http://www.dublincore.org/documents/dcmi-period/

[Cox00d] Cox, S.: *Recording qualified Dublin Core metadata in HTML meta elements.* Dublin Core Metadata Initiative, 2000 Available from: http://www.dublincore.org/documents/2000/08/15/dcq-html/

[Day00] Day, M.: *Metadata in a nutshell.* Information Europe 6, 2000

[DCM99] DCMI: *Dublin Core Metadata Element Set, Version 1.1: Reference Description.* Dublin Core Metadata Initiative, 1999, Available from: http://www.dublincore.org/documents/1999/07/02/dces/

[DCM00a] DCMI: *Dublin Core Qualifiers.* Dublin Core Metadata Initiative, 2000, Available from: http://www.dublincore.org/documents/dcmes-qualifiers/

[DCM00b] DCMI: *DCMI Type Vocabulary.* Dublin Core Metadata Initiative, 2000, Available from: http://dublincore.org/documents/dcmi-type-vocabulary/

[DST99] DSTC: *HotMeta Demonstration Site.* Distributed Systems Technology Centre, 1999, Available from: http://www.dstc.edu.au/Research/Projects/hotmeta/search.html

[Eil98] Eilebrecht, L.: *Resource Discovery und Information Retrieval.* Universität-Gesamthochschule Siegen, 1998

[FGD98] Federal Geographic Data Committee: *Content Standard for Digital Geospatial Metadata.* 1998, Available from: http://www.fgdc.gov/metadata/csdgm/

[FHJ98] Raggett, D.; Le Hors, A.; Jacobs, I.: *HTML 4.0 Spezification.* World Wide Web Consortium, 1998, Available from: http://www.w3.org/TR/REC-html40

[Fie97] Fielding, R. et al: *Hypertext Transfer Protocol -- HTTP/1.1.* Network Working Group, 1997

[For02] Forest Press: *Dewey Decimal Classification System.* OCLC Online Computer Library Center, 2002

[Get00] The J. Paul Getty Trust: *Getty Thesaurus of Geographic Names.* 2000, Available from: http://www.getty.edu/research/tools/vocabulary/tgn/index.html

[HA87] Horton, M.; Adams, R.: *Standard for Interchange of USENET Messages.* Network Working Group, 1987

[IMA02] ISO 3166 Maintanance Agency: *English country names and code elements.* Deutsches Institut für Normung e. V., 2002

[Kos94] Koster, M.: *A Standard for Robot Exclusion.* 1994, Available from: http://www.robotstxt.org/wc/norobots.html

[Kuh01] Kuhn, M.: *A Summary of the International Standard Date and Time Notation.* 2001, Available from: http://www.cl.cam.ac.uk/~mgk25/iso-time.html

[Kun99] Kunze, J.: *Encoding Dublin Core Metadata in HTML.* Dublin Core Metadata Initiative RFC 2731, 1999

[Lib01] Library of Congress: *Library of Congress Classifikation Outline.* 2001, Available from: http://lcweb.loc.gov/catdir/cpso/lcco/lcco.html

[Lib02] Library of Congress: *Codes for the Representation of Names of Languages-Part 1.* Library of Congress, 2002

[LS99] Lassila, O.; Swick, R. R.: *Resource Description Framework (RDF) Model and Syntax Specification.* World Wide Web Consortium, 1999, Available from: http://www.w3.org/TR/1999/REC-rdf-syntax-19990222

[Lyc02] Lycos Europe: *Webmaster FAQs.* 2002, Available from: http://www.HotBot.lycos.de/help/webmasterfaq.html

[MMB99] Miller, E.; Miller, P.; Brickley, D.: *Guidance on expressing the Dublin Core within the Resource Description Framework (RDF).* Dublin Core Metadata Initiative, 1999, Available from: http://www.ukoln.ac.uk/metadata/resources/dc/datamodel/WD-dc-rdf/

[Nel00] Nelson, S.: *Medical Subject Headings Fact Sheet.* National Library of Medicine, 2000

[Rus96] Rusch-Feja, D.: *Auf dem Weg zur Entwicklung eines Internet-Standards - 4. Dublin Core Metadata Workshop in Canberra.* Ehemaliges Deutsches Bibliotheksinstitiut, Bibliotheksdienst Heft 4, 1996

[Rus97] Rusch-Feja, D.: *Mehr Qualität im Internet: Entwicklung und Implementierung von Metadaten.* In Online-Tagung der DGD, 19, (pp. 113-130). [Berlin]: Deutsche Gesellschaft für Dokumentation, 1997

[Rus00] Rusch-Feja, D.: *International metadata initiatives: the latest developments.* Russian Digital Libraries, 3, 2000

[Sul01] Sullivan, D.: *The Search Engine Report.* 2001 Available from: http://www.searchenginewatch.com/

[THG02] The ht://Dig Group: *Features and System requirements.* 2002 Available from: http://www.htdig.org/

[Tsc01] Tschalär, R.: *HTTPClient Version 0.3-3.* Innovation GmbH, 2001, Available from: http://www.innovation.ch/java/HTTPClient/

[UDC01] UDC Consortium: *About Universal Decimal Classification and the UDC Consortium.* 2001 Available from: http://www.udcc.org/about.htm

[UDI96] U.S. Department Of The Interior: *Development of a Metadata Content Standard for Biological Resource Data.* 1996 Available from: http://www.computer.org/conferences/meta96/frondorf/ieeemeta.html

[UOC01] University Of California: *Simple Web Indexing Software for Humans - Enhanced.* 2001 Available from: http://swish-e.org/

[WGM95] Weibel, S.; Godby, J.; Miller, E.: *OCLC/NCSA Metadata Workshop Report. Office of Research.* OCLC Online Computer Library Center, Inc. 1995

[WKL98] Weibel, S.; Kunze, J.; Lagoze, C.; Wolf, M.: *Dublin Dore Metadata for Resource Discovery. IETF #2413.* The Internet Society, 1998

[WW98] Wolf, M.; Wicksteed, C.: *Date and Time Formats.* World Wide Web Consortium, 1998, Available from: http://www.w3.org/TR/NOTE-datetime

Erklärung

Ich versichere, dass ich diese Arbeit selbständig verfasst und nur die angegebenen Quellen verwendet habe.

_____

(Mihály Sütö)