

Universität Stuttgart

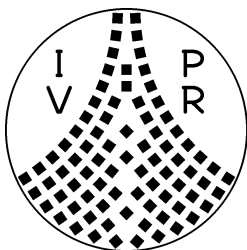
Fakultät Informatik

Prüfer: Prof. Dr. rer. nat. Kurt Rothermel
Betreuer: Dipl.-Inform. Wolfgang Theilmann
begonnen am 01.03.1999
beendet am 31.08.1999
CR-Klassifikation: H.3.2, H.3.3, D.3.4, H.3.0, H.3.5

Diplomarbeit Nr. 1762

Konzeption und Implementierung eines Frameworks für spezialisierte Suchmaschinen

Steffen Arnold



Institut für Parallele und Verteilte
Höchstleistungsrechner
Breitwiesenstraße 20-22
D-70565 Stuttgart

Danksagung

Ich möchte mich an dieser Stelle bei Wolfgang Theilmann für die sehr gute Betreuung meiner Diplomarbeit bedanken.

Kurzfassung

Spezialisierte Suchmaschinen sind Suchmaschinen für das World Wide Web, die auf ein Themengebiet beschränkt und zugeschnitten sind. Dadurch sind sie in der Lage, wesentlich genauere Ergebnisse auf Suchanfragen zu ermitteln, als dies universellen Suchmaschinen möglich ist.

Das Konzept der Domänen-Experten ist ein an der Fakultät Informatik der Universität Stuttgart entwickeltes Konzept für spezialisierte Suchmaschinen. Wesentlich für die Praxistauglichkeit des Konzeptes ist dabei, einen Domänen-Experten bis zu einem bestimmten Grad durch Generierung erstellen zu können.

Im Rahmen dieser Diplomarbeit wurde ein generisches Framework zur Erstellung von Domänen-Experten entworfen, das Teile des vorgegebenen Konzeptes der Domänen-Experten ausgestaltet und umsetzt. Es umfaßt ein generisches Dokumenten- und Anfragemodell, Realisierung einer CGI-Schnittstelle für Suchanfragen und Generierung einer themenspezifischen Suchmaske, sowie Funktionalität für die Bearbeitung von Suchanfragen.

Das erarbeitete generische Framework ist in der Lage Domänen-Experten mit den im Rahmen dieser Arbeit relevanten Teilen und Funktionalitäten vollständig generisch zu erstellen; manuelle Eingriffe sind nicht notwendig. Die Themengebietsunabhängigkeit des Konzeptes wurde anhand dreier Themengebiete überprüft.

Inhaltsverzeichnis

1	Einführung	1
1.1	Motivation und Übersicht	2
1.2	Bewertungskriterien für Suchwerkzeuge	3
1.3	Existierende Suchwerkzeuge	3
1.3.1	Klassifikation	4
1.3.2	Fachbegriffe	4
1.3.3	Zentralisierte Ansätze	5
1.3.4	Verteilte Ansätze	6
1.4	Analyse	9
1.4.1	Probleme und Mängel	9
1.4.2	Ursache	9
1.4.3	Voraussetzungen für Effektivität und Effizienz	10
1.5	Quellen	11
2	Domänen-Experten	13
2.1	Domänen-Experten – Einführung	14
2.1.1	Allgemeine Übersicht	14
2.1.2	Arbeitsweisen eines Domänen-Experten	15
2.2	Architektur von Domänen-Experten	16
2.3	Einsatz mobiler Filter-Agenten	18
2.4	Wissensbasis eines Domänen-Experten	20
2.5	Domänenspezifische Funktionalität	21
2.5.1	Domänenspezifische Teile	21

2.5.2	Optionale domänenspezifische Erweiterungen	21
2.6	Kooperation zwischen Domänen-Experten	21
2.7	Zugriff auf Domänen-Experten	22
2.8	Quellen	22
3	Grundlagen und Dokumentenmodell	23
3.1	Problem der Generizität	24
3.2	Aufgabenstellung der Diplomarbeit	25
3.3	Das Konzept - Grundlagen	25
3.3.1	Herleitung	25
3.3.2	Darstellung	26
3.3.3	Ausgestaltungsmöglichkeiten	26
3.4	Erstellung eines Domänen-Experten - Grundprinzip	27
3.5	Das generische Dokumentenmodell	28
3.5.1	Index	28
3.5.2	Dokumenten-Datensatz	29
3.5.3	Verworfenne Dokumente	31
4	Anfragebearbeitung	33
4.1	Benutzerdialog	34
4.2	CGI - Grundlegendes	34
4.3	CGI-Schnittstelle für Domänen-Experten	35
4.3.1	Prinzipielle Realisierung	35
4.3.2	Übertragungsmethode	37
4.4	Technisches zur CGI-Anbindung	37
4.4.1	CGI und HTML	37
4.4.2	Aufbau eines CGI-Parameterstrings	39
4.5	Generizität und themenspezifische Suchmaske	39
4.6	Weitere Funktionalität	42
4.6.1	Indexsuche unabhängig von Merkmalen	42
4.6.2	Anfragen an die konventionelle Suchmaschine	43

4.7	Die Anfragesprache	43
4.7.1	Beschreibung	44
4.7.2	Übersicht	44
4.8	Durchführung von Vergleichen	45
4.8.1	Ausprägungen im Index des Domänen-Experten	45
4.8.2	Ausprägungen in Dokumenten-Datensätzen des Domänen-Experten	45
4.9	Bewertung von Elementen der Domäne	46
4.10	Auswertung von Suchanfragen	47
4.10.1	Prinzipieller Algorithmus	47
4.10.2	Konkretisierung des Auswertungsalgorithmus	49
4.11	Bearbeiten von Suchanfragen	52
4.12	Formatierung der Suchergebnisse	53
4.13	Festlegung des Suchergebnis-Formats	53
4.13.1	Bezeichner	54
4.13.2	String	54
4.13.3	Zeilenumbruch	54
4.13.4	Funktionsaufrufe	54
4.14	Quellen	55
5	Generieren und Betriebsphase	61
5.1	Verschiedene Generierungsparameter	62
5.1.1	<code>ConventionalSearchEngineSearchString</code>	62
5.1.2	<code>QueryEvaluatingProgram</code>	63
5.1.3	<code>MaxNumberOfResults</code>	63
5.1.4	<code>DocumentAcceptanceThreshold</code>	63
5.2	Details zur Erstellung eines Domänen-Experten - Übersicht	64
5.3	Das Konfigurationsfile	64
5.4	Konsistenzbedingungen	67
5.5	Lexikalische Struktur und Syntax	68
5.5.1	Lexikalische Struktur	68

5.5.2	Syntax	69
5.6	Weiteres zum Konfigurationsfile	70
5.7	Betriebsphase - Vorbemerkung	70
5.8	Übersicht über die Betriebsphase	70
5.9	Ablauf des Wissenserwerbs	71
5.9.1	Allgemeiner Ablauf	71
5.9.2	Aufteilung der Funktionalität	73
6	Hinweise und Bewertung	75
6.1	Anwendungsunabhängigkeit des Ansatzes	76
6.2	Hinweise zur Realisierung	76
6.2.1	Index	76
6.2.2	Dokumenten-Datensatz	77
6.2.3	Interne Darstellung des Suchergebnis-Formats	78
6.3	Bewertung und Ausblick	80
6.3.1	Bewertung des Konzeptes der Domänen-Experten	80
6.3.2	Bewertung meines Konzeptes für ein generisches Framework	82
6.3.3	Gesamtbewertung	82
6.3.4	Ausblick	82
6.4	Quellen	83
A	Analyse dreier Sachgebiete	85
A.1	Wissenschaftliche Veröffentlichungen	85
A.1.1	Merkmale des Sachgebietes	85
A.1.2	Spezifikation eines Domänen-Experten	86
A.2	Abrufbare public domain Software	86
A.2.1	Merkmale des Sachgebietes	86
A.2.2	Spezifikation eines Domänen-Experten	87
A.3	Hotelinformationen	88
A.3.1	Merkmale des Sachgebietes	88
A.3.2	Spezifikation eines Domänen-Experten	89
B	Zeitplan	93
	Literaturverzeichnis	97

Kapitel 1

Einführung

Dieses Kapitel motiviert kurz die Aufgabenstellung meiner Diplomarbeit und gibt eine Übersicht über diese Ausarbeitung. Im Anschluß daran führt es in die Thematik und Problematik der Suchwerkzeuge ein.

Inhaltsangabe

1.1	Motivation und Übersicht	2
1.2	Bewertungskriterien für Suchwerkzeuge	3
1.3	Existierende Suchwerkzeuge	3
1.4	Analyse	9
1.5	Quellen	11

1.1 Motivation und Übersicht

Das World Wide Web enthält eine sehr große Menge an Informationen. Ist die Adresse einer Information bekannt, kann mittels eines sogenannten Browsers direkt darauf zugegriffen werden. Gängige Browser sind Netscape Navigator, Microsoft Internet Explorer und NCSA Mosaic.

Soll jedoch eine Information abgerufen werden, deren Adresse nicht bekannt ist, oder soll nach Informationen zu einem Stichwort oder Sachgebiet gesucht werden, so ist dies nicht so unmittelbar möglich. Für solche Aufgaben werden im World Wide Web *Suchwerkzeuge*¹ angeboten.

Die Qualität heute verfügbarer Suchwerkzeuge ist noch nicht befriedigend. So enthalten Ergebnisse auf Suchanfragen nicht alle Dokumente, die im World Wide Web verfügbar sind und zur Anfrage passen. Darüber hinaus werden viele Ergebnisse geliefert, die nicht zu den vom Benutzer beabsichtigten gehören. Auch sind manche Architekturen heutiger Suchwerkzeuge für die effiziente Verwaltung großer Datenmengen nicht geeignet.

W. Theilmann und K. Rothermel schlagen in [Theilmann98b] ein Konzept für ein neues Suchwerkzeug vor, das den Problemen heutiger Suchwerkzeuge Rechnung trägt: das Konzept der *Domänen-Experten*. Teil dieses Konzeptes ist u.a. die Forderung, solche Suchwerkzeuge bis zu einem bestimmten Grad *generieren* zu können.

Im Rahmen dieser Diplomarbeit soll nun ein Teil des Konzeptes der Domänen-Experten unter besonderer Berücksichtigung des Aspektes des Generierens realisiert werden.

Kapitel 1 dieser Ausarbeitung enthält eine Einführung in Thematik und Problematik der Suchwerkzeuge. Dabei gebe ich Bewertungskriterien für Suchwerkzeuge an und anschließend eine Übersicht über existierende Ansätze für Suchwerkzeuge. Es folgt eine Analyse heutiger Suchwerkzeuge mit Nennung von Mängeln und Ursachen für deren unbefriedigende Qualität, sowie Voraussetzungen für Effektivität und Effizienz dieser Art von Software.

Kapitel 2 stellt das Konzept der Domänen-Experten dar.

Kapitel 3 geht auf das Problem der Generizität ein, das letztlich diese Diplomarbeit motiviert, und nennt die Aufgabenstellung derselben. Im Anschluß daran beginnt die Darstellung des von mir erarbeiteten Konzeptes. Dabei erfolgt zunächst die Herleitung des Grundkonzeptes, welches die theoretische Grundlage meines Ansatzes bildet. Anschließend erkläre ich, wie ich beim Generieren prinzipiell vorgehe und stelle das generische Dokumentenmodell dar.

Kapitel 4 geht auf die technische Realisierung des Zugriffes auf Domänen-Experten ein, was zum Verständnis des im Anschluß dargestellten generischen Anfragemodells bei-

¹„Suchwerkzeug“ ist der Überbegriff für Programme zur Suche nach Informationen im World Wide Web. Diese Definition ist nicht allgemeingültig – in [Gudivada97] hat der Begriff eine andere Bedeutung. Ich verwende diese Definition aufgrund von [Theilmann98b] im Rahmen dieser Arbeit.

trägt. Zum Abschluß dieses Kapitels wird die Definition des Formats für die Ausgabe von Suchergebnissen erklärt.

Kapitel 5 nennt verschiedene Parameter, die bei der Erstellung eines Domänen-Experten angegeben werden können. Im Anschluß daran gehe ich auf das Konfigurationsfile ein, in dem ein Domänen-Experte definiert wird und anhand dessen die Generierung desselben erfolgt. Schließlich stelle ich die im Rahmen dieser Diplomarbeit für Domänen-Experten erstellte Betriebsphase dar.

Kapitel 6 schließlich geht auf die Domänenunabhängigkeit meines Konzeptes ein und gibt Hinweise zur Realisierung ausgesuchter Bereiche. Eine Bewertung der Konzepte mit kurzem Ausblick auf weitere Aufgaben schließt das Kapitel und die eigentliche Ausarbeitung ab.

Anhang A enthält die Daten von der Überprüfung der Domänenunabhängigkeit meines Konzeptes (Kapitel 6).

Anhang B enthält die Zeitpläne zu meiner Arbeit.

1.2 Bewertungskriterien für Suchwerkzeuge

Für die Bewertung der Qualität von Suchwerkzeugen ist es notwendig, Kriterien zu definieren, anhand derer die Bewertung erfolgen soll.

Im Rahmen dieser Diplomarbeit verwende ich folgende Kriterien:

Effektivität: Die Effektivität eines Suchwerkzeuges wird durch die Kriterien *Abruf* (engl. *recall*) und *Genauigkeit* (engl. *precision*) bestimmt.

Abruf bezeichnet das Verhältnis zwischen den gelieferten Dokumenten, die zu den gewünschten gehören, und den insgesamt im Web vorhandenen, gewünschten Dokumenten. Angestrebt werden hohe Abruf-Werte.

Genauigkeit bezeichnet das Verhältnis zwischen den gelieferten Dokumenten, die zu den gewünschten gehören, und der Gesamtzahl der gelieferten Dokumente. Auch für die Genauigkeit werden hohe Werte angestrebt.

Effizienz: Effizienz bezieht sich sowohl auf Zeiteffizienz, als auch auf Ressourceneffizienz. Zeiteffizienz beschreibt die Ausführungsgeschwindigkeit und damit die Antwortzeit des Suchwerkzeuges, Ressourceneffizienz die Nutzung von Systemressourcen durch das Suchwerkzeug (z.B. Speicherbedarf, Netzwerkbelastung).

1.3 Existierende Suchwerkzeuge

In diesem Abschnitt wird eine Übersicht über existierende Ansätze bei Suchwerkzeugen gegeben. Nach einer Klassifikation in Abschnitt 1.3.1 gehen die anschließenden Abschnitte näher auf diese Ansätze ein.

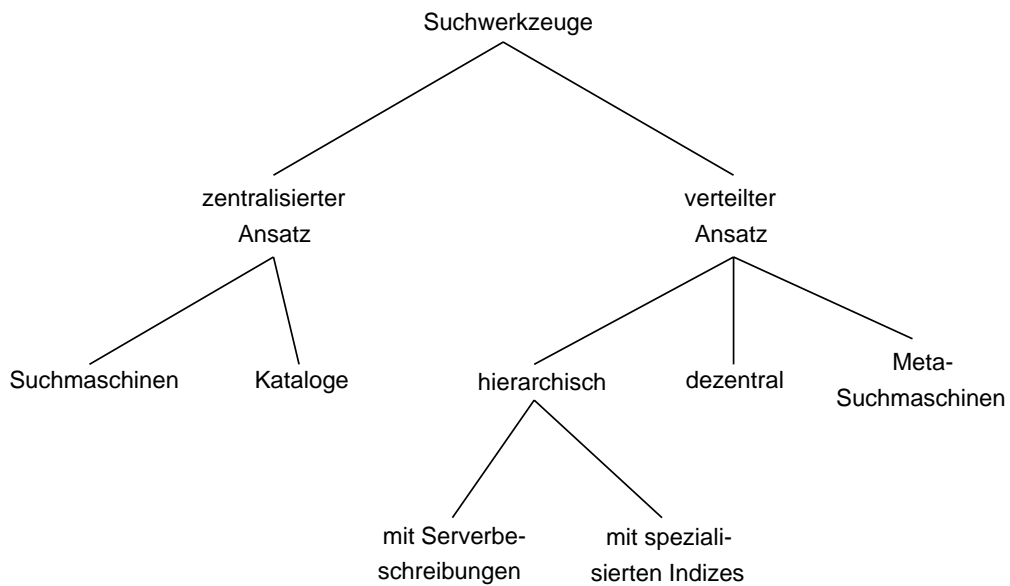


Abbildung 1.3.1: Klassifikation existierender Ansätze bei Suchwerkzeugen

1.3.1 Klassifikation

Derzeit existierende Suchwerkzeuge lassen sich bezüglich ihres Ansatzes der Datenverteilung und –organisation wie folgt klassifizieren²:

1. Suchwerkzeuge, die auf einem *zentralisierten Ansatz* beruhen, und
2. Suchwerkzeuge, die auf einem *verteilten Ansatz* beruhen.

Suchwerkzeuge, die auf einem *zentralisierten Ansatz* beruhen, lassen sich weiter in die Kategorien *Suchmaschine* und *Katalog* unterteilen. Bei Suchwerkzeugen, die auf einem *verteilten Ansatz* beruhen, gibt es die Kategorien *hierarchisch*, *dezentral* und *Meta-Suchmaschinen*. In der Kategorie *hierarchisch* lassen sich weiter Suchwerkzeuge, die mit *Server-Beschreibungen* arbeiten, und solche, die *spezialisierte Indizes* verwenden, unterscheiden.

Abbildung 1.3.1 stellt diese Klassifikation graphisch dar.

1.3.2 Fachbegriffe

Die Begriffe *Index* und *Phrasen-Suche* (engl. *phrase search*) werden im Zusammenhang mit Suchwerkzeugen sowohl in den folgenden Abschnitten, als auch sonst in dieser Arbeit immer wieder verwendet werden. Aus diesem Grund gehe ich in diesem Abschnitt kurz auf sie ein.

²Diese Klassifikation und Nomenklatur ist nicht allgemeingültig. Sie basiert auf [Theilmann98b] und [Sander-Beuermann98] und wird von mir im Rahmen dieser Arbeit verwendet. Andere Arbeiten oder Veröffentlichungen können – wie z.B. [Gudivada97] – davon abweichen.

Der *Index* ist ein Teil der Dokument–Wissensbasis eines Suchwerkzeuges. In ihm werden Dokumente wortbezogen abgelegt, d.h. zu jedem Wort, das in mindestens einem Dokument auftritt, wird vermerkt, in welchem Dokument/welchen Dokumenten es auftritt. Zu der Angabe des Dokumentes, in dem ein Wort auftritt, können weitere Zusatzinformationen gespeichert werden, z.B. die Position des Wortes im Dokument oder andere Informationen, die sich aus dem Kontext des Wortes im Dokument ergeben (z.B. ob das Wort in einer Literaturangabe steht). Man beachte, daß ein Wort in einem Dokument mehrfach enthalten sein kann.

Phrasen–Suche (engl. *phrase search*) steht für die Suche nach Wortfolgen. Mit einer Phrasen–Suche können Dokumente selektiert werden, die an mindestens einer Stelle die Wörter der Phrase in genau der genannten Reihenfolge enthalten (oder bei denen dies gerade nicht der Fall ist – je nach Formulierung der Anfrage). Man beachte, daß bei einem Index die Phrasen–Suche realisierbar ist, wenn zu jedem Wort die Position im Dokument gespeichert wird.

1.3.3 Zentralisierte Ansätze

Suchwerkzeuge, die heute weltweit im Einsatz sind, basieren alle auf einem zentralisierten Ansatz.

Suchmaschinen

Suchmaschinen zielen darauf ab, das ganze World Wide Web zu durchsuchen und die so gewonnenen Informationen in einem Index abzulegen. Durchgeführt wird dies von einem Programmteil der Suchmaschine, die *Roboter* genannt wird. Der Roboter nimmt eine Suche periodisch vor. Suchanfragen an die Suchmaschine werden anhand des so erstellten Indexes beantwortet.

Suchmaschinen arbeiten völlig automatisch. Trotzdem wird von ihnen höchstens ein Drittel der Dokumente des Webs erfaßt; dies mindert den Abruf–Wert von Suchanfragen.

Der größte Nachteil von Suchmaschinen ist jedoch, daß oft Dokumente geliefert werden, die nicht zu den vom Benutzer beabsichtigten gehören (Genauigkeits–Wert). Sie werden geliefert, weil sie formal der Anfrage entsprechen, sind jedoch aufgrund eines anderen Kontextes nicht relevant im Sinne der Anfrage (siehe Beispiel von Abschnitt 1.4.1 Punkt 2).

Ein weiteres Problem ist die Aktualität der Ergebnisse. So werden oft Verweise geliefert, die nicht mehr existieren.

Beispiele für Suchmaschinen sind AltaVista ([AltaVista]) und Lycos ([Lycos]).

Kataloge

Bei Katalogen werden die Dokumente in eine hierarchische, sie charakterisierende Struktur eingeordnet. Der Benutzer eines Kataloges kann diese hierarchische Struktur schrittweise absuchen oder auch Suchvorgänge auf dem kompletten Datenbestand anstoßen.

Klassifizierung eines Dokumentes und Aufnahme in den Katalog erfolgen manuell durch eine Person. Dadurch ist Informationssuche in Katalogen durch hohe Genauigkeitswerte gekennzeichnet. Die manuelle Pflege des Indexes macht es jedoch unmöglich, alle Dokumente des World Wide Webs zu erfassen; die Abruf-Werte hängen daher stark von dem Wissensgebiet ab, um das es geht. So wird der Betreiber eines Katalogs bei populären Themen eher Zeit darauf verwenden, den Katalog auf einem aktuellen Stand zu halten, während bei speziellen, nicht populären Themen möglicherweise überhaupt kein Eintrag im Katalog vorhanden ist.

Da der Datenbestand von Hand gepflegt wird, kann dessen Struktur leicht an den Datenumfang angepaßt werden und so auch eine Verteilung auf verschiedene Rechner vorgenommen werden.

Suchwerkzeuge, die nur Wissen aus einem bestimmten Gebiet enthalten oder sich nur einem bestimmten Thema widmen, sind normalerweise als Katalog realisiert.

Beispiel für einen Katalog ist das Suchwerkzeug Yahoo! ([Yahoo!]).

1.3.4 Verteilte Ansätze

Das Problem zentralisierter Ansätze bei Suchwerkzeugen ist, daß bei wachsendem Datenbestand die Effizienz abnimmt: Das Suchwerkzeug muß Anfragen anhand eines u.U. sehr großen Datenbestandes beantworten, zudem gehen alle Anfragen an dieses eine Suchwerkzeug. Dies bedeutet bei bestimmten Arten, den Index zu gestalten, längere Antwortzeiten, insbesondere aber dann, wenn mehrere Benutzer gleichzeitig Anfragen stellen. Um dieses Problem der Skalierbarkeit von Suchwerkzeugen zu lösen, wurden viele verteilte Ansätze für Suchwerkzeuge entwickelt.

Hierarchische Ansätze

- **Hierarchisch, mit Server-Beschreibungen:**

Die auf Servern verfügbaren Informationen werden durch charakterisierende Begriffe beschrieben. Diese Begriffe spezifizieren dadurch die Inhalte des Servers und stellen somit eine Serverbeschreibung dar. Die Begriffe werden an einen Server übergeben, der solche Serverbeschreibungen speichert, dieser wiederum kann die Serverbeschreibungen zusammenfassen und an einen dritten Server übergeben, der sie speichert und ebenso handeln kann, usw.. Charakterisierende Begriffe enthalten keine Verweise auf die entsprechenden Informationen eines Servers.

Suchanfragen werden an einen Server mit Serverbeschreibungen der obersten Ebene gerichtet, der diese dann an die entsprechenden Server der darunterliegenden Ebene weiterleitet. Diese wiederum handeln ebenso. Dadurch wird die Anfrage an geeignete Server geleitet.

Nachteile: Serverbeschreibungen durch Stichworte sind u.U. nicht eindeutig. Es können dadurch Anfragen an Server geleitet werden, die keine relevanten Dokumente enthalten. Auch sind Informationen zu einem thematischen Gebiet in der Regel nicht nur auf einem einzigen Server vorhanden. Daher wird eine Anfrage an viele verschiedene Server weitergeleitet werden.

Beispiele für solche Suchwerkzeuge sind WHOIS++ ([Weider96]) und Discover ([Sheldon95]).

- **Hierarchisch, mit spezialisierten Indizes:**

Ein Suchwerkzeug dieser Kategorie enthält viele Indizes, wobei jeder Index sich nur einem Wissensgebiet oder Thema widmet. Solche Indizes werden *spezialisierte Indizes* genannt. Die Indizes enthalten Informationen über die Dokumente.

Beim Suchwerkzeug Harvest ([Bowman95]) existiert eine spezielle Stelle, die Harvest Server Registry, die einem Benutzer hilft, den für seine Suchanfrage geeigneten Broker (jeder Index ist in einem sogenannten Broker enthalten) zu finden.

Harvest ist sehr effizient in der Verteilung von Index-Daten. Auch ist es in der Lage, die Daten effizient auf den neuesten Stand zu bringen. Ein Nachteil ist jedoch, daß Links von Brokern zu Informationsquellen manuell zu erstellen sind. Dadurch können Broker neue Informationsquellen nicht selbständig entdecken.

Das Suchwerkzeug Pharos ([Dolin97]) legt für die Beschreibung von Dokumenten und Indizes drei Taxonomien fest (Thema, Zeit, Ort). Ein Dokument wird auf seinem Server automatisch kategorisiert, diese Beschreibung an den geeigneten spezialisierten Index gesandt, der diese aufnimmt. Dieser Index wird mid-level-server genannt. Die Themengebiete, die mid-level-server enthalten, werden in sogenannten high-level-servern gespeichert, welche dann Benutzer für Anfragen an die geeigneten mid-level-server verweisen.

Bei Pharos können neue Informationsquellen automatisch, d.h. ohne manuellen Eingriff, genutzt werden. Allerdings erfordert dies einen weltweiten Standard zur Beschreibung von Dokumenten (obige Taxonomien), was einen geringeren erreichbaren Genauigkeit-Wert bewirkt, sowie die Konstruktion spezialisierter Indizes beeinflusst. So ist es nur möglich, für Themengebiete, die durch den definierten Standard zur Beschreibung von Dokumenten ausgedrückt werden können, einen spezialisierten Index zu erstellen.

Dezentrale Ansätze

Bei Suchwerkzeugen dieser Kategorie wird versucht, die Speicherung von und die Suche nach Informationen in vollständig dezentraler Weise zu realisieren.

Das Suchwerkzeug Ingrid ([Francis95]) ist ein Vertreter dieser Kategorie. Die Verteilung erfolgt hier auf Dokumentenebene. Jedes Dokument wird durch 15 Schlüsselwörter beschrieben, außerdem enthält die Beschreibung eines jeden Dokuments Verweise zu Dokumenten, die mindestens eines der Schlüsselwörter ebenfalls besitzen. Dies geschieht in der Art und Weise, daß zwei Dokumente mit mindestens einem gemeinsamen Schlüsselwort immer durch einen Pfad verbunden sind.

Problematisch für diesen Ansatz ist die einfache Beschreibung der Dokumente, die z.B. keine Phrasen-Suche zuläßt. Darüber hinaus ist die Bearbeitung von Suchanfragen aufwendig – was die Effizienz mindert.

Suchwerkzeuge, die auf einem dezentralen Ansatz beruhen, haben das generelle Problem, daß effektive Algorithmen zur Beantwortung von Suchanfragen ziemlich komplex sind und viel Eigenwissen (wie z.B. Wörterbücher) benötigen. Da es jedoch nicht effizient ist, dieses Eigenwissen vielfach zu replizieren – was notwendig wäre –, kommen nur sehr einfache Algorithmen zur Beantwortung von Suchanfragen zum Einsatz. Dies bewirkt eine geringere Effektivität solcher Suchwerkzeuge.

Meta-Suchmaschinen

Meta-Suchmaschinen bearbeiten Suchanfragen, indem sie andere Suchwerkzeuge und Informationsquellen beauftragen bzw. abfragen. Sie bieten dem Benutzer eine Maske, in der er die Suchanfrage eingeben kann. Die Suchanfrage wird von der Meta-Suchmaschine gleichzeitig an mehrere Suchwerkzeuge und Informationsquellen verteilt. Verwendet ein beauftragtes Suchwerkzeug oder eine befragte Informationsquelle für die Formulierung einer Anfrage eine andere Syntax, so nimmt die Meta-Suchmaschine die entsprechende Transformation vor. Die Ergebnisse der beauftragten Suchwerkzeuge und Informationsquellen werden von der Meta-Suchmaschine idealerweise zusammengeführt, außerdem werden Duplikate entfernt. Das Resultat wird anschließend dem Benutzer präsentiert.

Der Ansatz, den Meta-Suchmaschinen verfolgen, ist geeignet, die Abruf-Werte zu erhöhen, wie stark hängt jedoch davon ab, in wie weit sich die Datenbestände der beauftragten Suchwerkzeuge und Informationsquellen ergänzen und nicht überschneiden. Bei den Genauigkeits-Werten sind Meta-Suchmaschinen jedoch im wesentlichen davon abhängig, welche Genauigkeits-Werte die beauftragten Suchwerkzeuge und befragten Informationsquellen liefern.

Anstatt *Meta-Suchmaschine* werden auch die Bezeichnungen *Meta-Maschine*, *Multi-Searcher* oder *ParallelSearcher* verwendet.

Beispiele für Meta-Suchmaschinen sind MetaCrawler ([MetaCrawler]) und MetaGer ([MetaGer]).

1.4 Analyse

Dieser Abschnitt nennt Probleme und Mängel heutiger Suchwerkzeuge, gibt eine wesentliche Ursache für deren unbefriedigende Qualität an und nennt Voraussetzungen für effektive und effiziente Suchwerkzeuge.

1.4.1 Probleme und Mängel

Analysiert man heutige Suchwerkzeuge, so findet man keines, das eine sowohl effektive als auch effiziente Suche im World Wide Web ermöglicht. Ja mehr noch, es findet sich auch kein Konzept für ein Suchwerkzeug, das dies prinzipiell ermöglichen könnte.

Drei wesentliche Mängel heutiger Suchwerkzeuge sind:

1. Ergebnissen auf Suchanfragen liegt nur ein Teil der Informationen des World Wide Webs zugrunde, d.h. es werden nicht alle passenden, im World Wide Web verfügbaren Dokumente geliefert. Mit anderen Worten gesagt: die Abruf-Werte sind zu gering.
2. Viele der auf eine Anfrage gelieferten Dokumente sind nur aus formalen Gründen relevant im Sinne der Anfrage, nicht jedoch im vom Benutzer beabsichtigten Sinne. Ursache dafür kann sein, daß sie einem anderen thematischen Kontext entstammen (z.B. Suche nach Informationen zur Stadt Rom liefert auch Dokumente zu (CD-)ROMs). Unter Verwendung der Terminologie von Abschnitt 1.2: Die Genauigkeits-Werte sind zu gering.
3. Die Architekturen mancher Suchwerkzeuge sind nicht geeignet, den Datenbestand so zu verwalten, daß unabhängig von dessen Umfang effizienter Betrieb möglich ist. Mit anderen Worten: Manche Architekturen skalieren nicht oder nur schlecht.

Insgesamt kann die Qualität heutiger Suchwerkzeuge als noch nicht befriedigend eingestuft werden.

1.4.2 Ursache

Es stellt sich natürlich die Frage nach den Gründen für die unbefriedigende Qualität derzeit verfügbarer Suchwerkzeuge. Verwaltung von und Umgang mit Datenbeständen sind in der Informatik ja nicht neu.

Die wichtigsten Eigenschaften des World Wide Webs, die es von herkömmlichen Datenbanken unterscheidet, sind:

- Größe: Das World Wide Web hält weitaus mehr abrufbare Dokumente (z.B. Text-Dokumente, Software, Bilder) bereit als jede Datenbank.

- **Inhaltliche Dynamik:** Die Inhalte des World Wide Webs ändern sich ständig, jeden Tag kommen tausende Dokumente hinzu oder werden gelöscht.
- **Keine regelnde zentrale Stelle:** Niemand steuert zentral die inhaltlichen Veränderungen oder die Struktur, in der die Dokumente verknüpft sind.
- **Heterogenität:** Die Dokumente können unterschiedlichste Formate besitzen, neue Formate können jederzeit hinzukommen.

Für jede dieser vier Eigenschaften gibt es einen Ansatz, der die Probleme löst, die sich daraus ergeben. Es gibt jedoch keinen Ansatz, der dies für alle vier Eigenschaften gleichzeitig kann. Genau dies ist nun eine wesentliche Ursache der nicht befriedigenden Qualität heutiger Suchwerkzeuge.

1.4.3 Voraussetzungen für Effektivität und Effizienz

Aus der Analyse der bestehenden Suchwerkzeuge und ihrer speziellen Vor- und Nachteile (Abschnitt 1.3) lassen sich drei Punkte ableiten, die entscheidend für die Architektur eines effektiven und effizienten Suchwerkzeuges sind:

1. Der Architektur muß ein verteilter Ansatz zugrunde liegen. Angesichts der Menge von Dokumenten, die über das World Wide Web erreichbar ist, und der Anzahl der Benutzer des World Wide Webs ist es offensichtlich, daß ein zentralisierter Ansatz für ein Suchwerkzeug nicht geeignet ist, effektive und effiziente Suche zu ermöglichen. Große Benutzerzahlen könnten durch schlichte Replikation des Suchwerkzeuges bewältigt werden. Wenn es jedoch darum geht, die Informationen im Index der Suchmaschine aktuell zu halten, kommt man angesichts der inhaltlichen Dynamik des World Wide Webs nicht umhin, den Index auf mehrere Server zu verteilen.
2. Bei einem verteilten Index wird die Menge der Dokumenten-Beschreibungen auf mehrere einzelne Indizes verteilt. Um eine effiziente Suche zu ermöglichen ist es notwendig, die Dokumenten-Beschreibungen nach Inhalten zu gruppieren; eine Suchanfrage kann dann an die passenden Gruppen weitergeleitet werden. Ohne eine solche Gruppierung müssen für jede Suchanfrage viele Indizes durchsucht werden.
3. Die von Suchwerkzeugen z.B. bei der Suche nach Dokumenten verfolgten Strategien müssen an die Themengebiete der Dokumente angepaßt werden. Es ist nicht realistisch anzunehmen, ein einziger Suchalgorithmus sei in der Lage, für alle Themengebiete gleichermaßen gute Ergebnisse zu liefern. Dies wird z.B. deutlich, wenn man die Themengebiete „Public Domain Software“ und „Homepages von Personen“ betrachtet.

1.5 Quellen

Kapitel 1 beruht auf [Theilmann98b], [Theilmann98a], [Sander–Beuermann98] und [Gudivada97]. Liegen Textteilen andere Quellen zugrunde, so sind diese Quellen an den entsprechenden Stellen genannt.

Den Angaben und Ausführungen zu Suchwerkzeugen liegen ebenfalls die gerade erwähnten Quellen zugrunde und *nicht* die zu konkreten Suchwerkzeugen jeweils in Klammern angegebenen – jene sind für den interessierten Leser als weiterführende Literatur gedacht.

Kapitel 2

Domänen-Experten

In diesem Kapitel wird das Konzept der Domänen-Experten dargestellt.

Inhaltsangabe

2.1	Domänen-Experten – Einführung	14
2.2	Architektur von Domänen-Experten	16
2.3	Einsatz mobiler Filter-Agenten	18
2.4	Wissensbasis eines Domänen-Experten	20
2.5	Domänenspezifische Funktionalität	21
2.6	Kooperation zwischen Domänen-Experten	21
2.7	Zugriff auf Domänen-Experten	22
2.8	Quellen	22

2.1 Domänen-Experten – Einführung

Ein *Domänen-Experte* ist ein Suchwerkzeug, das auf einem neuen Konzept basiert. Bei diesem Konzept werden die in Abschnitt 1.4.3 beschriebenen Voraussetzungen für ein effektives und effizientes Suchwerkzeug berücksichtigt.

Der Begriff „Domäne“ steht im Rahmen dieser Arbeit für ein (beliebiges) Wissens- oder Informations-Gebiet, dessen Informationen semantisch zusammenhängen.

Abschnitt 2.1.1 gibt eine kurze Übersicht über Domänen-Experten. In Abschnitt 2.1.2 wird auf die verschiedenen Arbeitsweisen eines Domänen-Experten eingegangen.

2.1.1 Allgemeine Übersicht

Ein Domänen-Experte ist ein spezialisiertes Suchwerkzeug. Er eignet sich nur zur Suche nach Informationen einer Domäne, auf die er angepaßt ist.

Für diese Domäne besitzt der Domänen-Experte spezifisches Wissen, durch das er insbesondere auch in der Lage ist zu entscheiden, ob ein Dokument zu dieser Domäne gehört oder nicht. Er sammelt nur Beschreibungen von Dokumenten, die zu dieser Domäne gehören. Die Beschreibungen speichert er in seinem lokalen Datenbestand.

Hinweise auf Dokumente seiner Domäne erhält er von einer konventionellen Suchmaschine in Form von Uniform Resource Locators¹ (URLs) des World Wide Webs (WWW), indem er Suchanfragen von Benutzern an diese konventionelle Suchmaschine weiterleitet. Er ist aber auch in der Lage, Hinweise auf Dokumente z.B. aus Dokumenten abzuleiten, über die er Informationen in seinem Datenbestand besitzt. Darüber hinaus kann ein Domänen-Experte digitale Bibliotheken oder andere Domänen-Experten konsultieren.

Dokumente, deren URLs auf solche Weise erworben wurden, werden durch Filter-Agenten eingehender auf ihre Zugehörigkeit zur Domäne hin untersucht. Diese Filter-Agenten sind als mobile Agenten² realisiert, d.h. sie sind in der Lage, sich auf bestimmte Rechner des Internets zu transferieren und Dokumente von dort aus zu untersuchen. Dies geschieht, um eine effiziente Untersuchung der Dokumente zu ermöglichen.

¹Ein *Uniform Resource Locator* ist ein Link im World Wide Web. Er enthält die Zugriffsmethode, den Server und die Dokumentangabe mit Pfad. In einem URL sind alle notwendigen Informationen vorhanden, um ein Dokument des World Wide Webs abrufen zu können.

Beispiele für URLs:

`http://www.informatik.uni-stuttgart.de/kolloquium/kolloquium.html`

`ftp://ftp.uni-stuttgart.de/pub/`

http bzw. ftp bezeichnen in den zwei Beispielen jeweils die Zugriffsmethode, `www.informatik.uni-stuttgart.de` bzw. `ftp.uni-stuttgart.de` den Server und `kolloquium/kolloquium.html` bzw. `pub/` das Dokument mit Pfad. Im Fall von `pub/` wird dabei genaugenommen kein Dokument spezifiziert, sondern ein Verzeichnis. □

(basierend auf [Nitsche-Ruhland95])

²Ein *mobiler Agent* kann z.B. wie folgt definiert werden: „Ein Prozeß, der von sich aus von einem Rechner zu einem anderen Rechner migrieren kann und – aufgrund von Entscheidungen, die auf lokal durchgeführten Berechnungen/Operationen beruhen – von sich aus zu einem dritten Rechner“

Anhand der Untersuchungsergebnisse der Filter-Agenten entscheidet der Domänen-Experte, welche Dokumente zur Domäne gehören.

Nach Aufbau einer ausreichenden Wissensbasis im eigenen Index kann ein Domänen-Experte Suchanfragen allein anhand dieses Indexes beantworten.

Der Datenbestand eines Domänen-Experten enthält nicht nur Inhalte der entsprechenden Dokumente, sondern auch Meta-Informationen über die Dokumente, die implizit durch die Strukturen oder Umgebungen der entsprechenden Dokumente zum Ausdruck kommen.

Der Domänen-Experte ist ohne manuelle Eingriffe in der Lage, die eigene Wissensbasis aufzubauen, zu erweitern und zu pflegen.

2.1.2 Arbeitsweisen eines Domänen-Experten

Es lassen sich drei Arbeitsweisen eines Domänen-Experten unterscheiden:

1. Zu Beginn des Betriebes eines Domänen-Experten besitzt dieser in seinem Datenbestand keinerlei Informationen über Dokumente. Er besitzt lediglich domänen-spezifisches Wissen zur Erkennung relevanter Dokumente.

Der Domänen-Experte baut seinen Datenbestand auf folgende Weise auf: Die Suchanfrage eines Benutzers wird an eine konventionelle Suchmaschine weitergegeben, die eine Menge von URLs zurückliefert. Diese URLs werden mit Hilfe von mobilen Filter-Agenten auf ihre Zugehörigkeit zur Domäne hin bewertet. Vom Domänen-Experten als zugehörig eingestufte Dokumente werden von diesem geladen und eingehender analysiert. Dabei werden Beschreibungen der Dokumente erstellt und in den Datenbestand aufgenommen. Die Suchanfrage des Benutzers wird anschließend anhand des Datenbestandes beantwortet.

Solches Vorgehen soll im folgenden als *lernende Arbeitsweise* bezeichnet werden, weil der Domänen-Experte von der konventionellen Suchmaschine gewissermaßen lernt.

Der Domänen-Experte ist somit von Anfang an in der Lage, Suchanfragen zu beantworten.

migrieren kann, wird *mobiler Agent* genannt.“ (nach [Rothermel98, Seite 2, Absatz 6] übersetzt). Dies ist eine strenge, technische Definition aus der Literatur.

Für den Rahmen dieser Arbeit seien mobile Agenten (unter Verwendung von [Theilmann99, Seite 2, Fußnote 1]) etwas weitergehendender wie folgt erklärt: *Mobile Agenten* sind Prozesse, die imstande sind, eine an sie gestellte Aufgabe – auch autonom, d.h. ohne Kontakt zum Auftraggeber – auszuführen. Eine wesentliche Eigenschaft mobiler Agenten ist zudem, daß sie in der Lage sind, sich selbst von einem Rechner auf einen anderen Rechner zu verlagern und dort die Ausführung fortzusetzen. Voraussetzung für die Anwesenheit, Ausführbarkeit und jegliche Aktivität eines mobilen Agenten auf einem Rechner ist dabei das Vorhandensein einer *Agenten-Plattform* auf dem Rechner; ein mobiler Agent kann sich nur von Agenten-Plattform zu Agenten-Plattform verlagern.

2. Nach einer bestimmten Zeit der lernenden Arbeitsweise besitzt der Domänen-Experte eine ausreichende Wissensbasis, um die Mehrzahl der Benutzeranfragen autonom, d.h. ausschließlich anhand seines Indexes, zu beantworten. Die Suchmaschine wird nur benutzt, um zu überprüfen, ob neue Dokumente im World Wide Web verfügbar sind und ob bekannte Dokumente verändert worden sind (veränderte Zeitmarke des Dokumentes). Mobile Filter-Agenten werden nur ausgesandt, wenn von der Suchmaschine neue oder veränderte Dokumente gefunden wurden.

Dieses Vorgehen soll im folgenden als *autonome Arbeitsweise* bezeichnet werden.

3. Bei der dritten Arbeitsweise versucht der Domänen-Experte ohne den Kontext einer benutzerseitig gestellten Suchanfrage neue Dokumente zu finden. Dabei kann er auf verschiedene Weise vorgehen:
 - Er kann die Umgebung eines ihm bekannten Dokumentes untersuchen, d.h. z.B. Dokumente des gleichen Web-Servers.
 - Bekannte Dokumente können nach URLs, Namen von Personen oder Konferenzen, usw. durchsucht werden. Diese Hinweise können dann zur weiteren Suche nach relevanten Dokumenten genutzt werden, indem auf Suchmaschinen, geeignete Domänen-Experten oder andere Informationsquellen zurückgegriffen wird.
 - Der Domänen-Experte kann externe Wissensbestände durchsuchen.

Solchermaßen gefundene Dokumente werden – wie bei der lernenden Arbeitsweise beschrieben – eingeschätzt, ggf. geladen, analysiert und in den Index aufgenommen.

Diese Art vorzugehen soll im Rahmen dieser Arbeit *proaktive Arbeitsweise* genannt werden.

Die drei Arbeitsweisen können nicht drei zeitlich disjunkten Phasen der Lebensdauer eines Domänen-Experten zugeordnet werden. Ihr Auftreten ist vielmehr zu verschiedenen Zeiten der Lebensdauer des Domänen-Experten verschieden stark ausgeprägt. Die autonome Arbeitsweise wird zu Beginn des Einsatzes eines Domänen-Experten kaum eingesetzt, während die lernende Arbeitsweise mit wachsender Einsatzdauer an Bedeutung verliert. Die proaktive Arbeitsweise ist prinzipiell jederzeit durchführbar.

2.2 Architektur von Domänen-Experten

Abbildung 2.2.1 stellt Architektur und Funktionsweise eines Domänen-Experten graphisch dar. Die Nummerierung der Pfeile gibt die Reihenfolge der Schritte an, die zur Bearbeitung einer eingehenden Suchanfrage durchgeführt werden; bei optionalen Schritten ist die Nummer in Klammern angegeben.

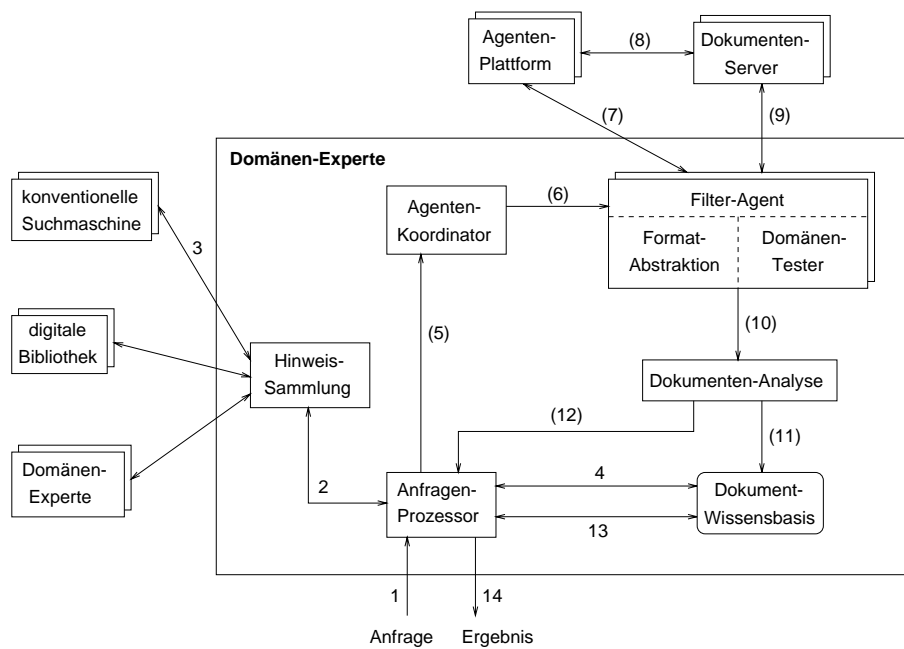


Abbildung 2.2.1: Architektur und Funktionsweise eines Domänen-Experten

Die Bearbeitung einer eingehenden Suchanfrage erfolgt folgendermaßen: Die Suchanfrage des Benutzers wird vom *Anfragen-Prozessor* entgegengenommen (1). Dieser beauftragt die *Hinweis-Sammlung*, initiales Wissen von einer externen Wissensbasis einzuholen (2). Die *Hinweis-Sammlung* erfüllt diesen Auftrag, indem sie auf eine konventionelle Suchmaschine zurückgreift (3). Die von dieser Suchmaschine gelieferten URLs werden an den *Anfragen-Prozessor* übermittelt.

Man beachte, daß die Dokumente, deren URLs übermittelt wurden, nur einigen der vom Benutzer eingegebenen Schlüsselwörter genügen. Wahrscheinlich gehört auch nur ein kleiner Teil dieser Dokumente zur Domäne des Domänen-Experten und ein noch kleinerer Teil eignet sich als Ergebnis auf die Suchanfrage des Benutzers.

Der *Anfragen-Prozessor* überprüft zunächst, welche der übermittelten URLs bereits in der *Dokument-Wissensbasis* vorhanden sind (4). Ein Dokument, dessen URL in der *Dokument-Wissensbasis* noch nicht vorhanden ist oder dessen Zeitmarke sich von der in der *Dokument-Wissensbasis* gespeicherten Zeitmarke unterscheidet, muß eingehender untersucht werden. Die URLs solcher Dokumente werden an den *Agenten-Koordinator* übergeben (5). Der *Agenten-Koordinator* entscheidet, zu welchen *Agenten-Plattformen* er mobile *Filter-Agenten* schicken will und welche Dokumente jeder mobile *Filter-Agent* untersuchen soll (6). Die mobilen *Filter-Agenten* migrieren zu den *Agenten-Plattformen*, denen sie zugewiesen worden sind (7) und untersuchen die ihnen zugewiesenen Dokumente (8). Es besteht auch die Möglichkeit, daß Dokumente von einem *Filter-Agenten* untersucht werden, der sich auf der *Agenten-Plattform* des Domänen-Experten befindet (9). Die *Filter-Agenten* liefern zu jedem untersuchten Dokument eine Bewertungszahl, die den Zusicherungsgrad angibt, daß das Dokument zu der Domäne des Domänen-Experten gehört; außerdem werden ermittelte Informa-

tionen über das jeweilige Dokument übergeben (10). Der Domänen-Experte entscheidet, aufgrund der übermittelten Informationen, welche der Dokumente er als seiner Domäne zugehörig betrachtet. Diese Dokumente werden vom Domänen-Experten geladen und weiter analysiert, um Dokumenten-Beschreibungen aus Schlüsselwörtern und Meta-Schlüsselwörtern zu erstellen. Die Dokumenten-Beschreibungen werden der Dokument-Wissensbasis hinzugefügt (11). Von Dokumenten, die der Domänen-Experte als nicht zu seiner Domäne gehörig bewertet, werden in der Dokument-Wissensbasis nur die für solche Dokumente notwendigen Informationen abgelegt (11). Die Ergebnisse der Analysen werden auch dem Anfragen-Prozessor mitgeteilt (12). Der Anfragen-Prozessor bestimmt das Ergebnis der vom Benutzer an ihn gestellten Suchanfrage, indem er die Dokument-Wissensbasis konsultiert (13). Schließlich liefert er dieses Ergebnis an den Benutzer (14).

Man beachte, daß der Einsatz der mobilen Filter-Agenten (Schritte 5 bis 12) nicht bei jeder Suchanfrage eines Benutzers erfolgt. Zu Beginn des Einsatzes eines Domänen-Experten ist die Wahrscheinlichkeit, daß diese Schritte ausgeführt werden, nahezu 1. Mit fortschreitendem Einsatz des Domänen-Experten wird die Wahrscheinlichkeit dann fast 0.

Andere Domänen-Experten oder digitale Bibliotheken werden nur bei proaktiver Arbeitsweise benutzt.

2.3 Einsatz mobiler Filter-Agenten

Der Domänen-Experte erhält von der konventionellen Suchmaschine Hinweise auf Dokumente, die möglicherweise für seine Domäne relevant sind. Dabei erhält er insbesondere die URLs dieser Dokumente. Die Dokumente, über die er noch keine Informationen im Index besitzt oder die verändert wurden, müssen auf Zugehörigkeit zur Domäne untersucht werden. Da davon ausgegangen werden kann, daß nur ein Teil dieser zu untersuchenden Dokumente zur Domäne gehört, besteht nun die Situation, daß eine große Menge von Dokumenten verteilt über das World Wide Web zu untersuchen ist, jedoch nur ein Teil davon wirklich lokal benötigt wird. Es bietet sich daher aus Effizienzgründen an, die Dokumente vor Ort zu untersuchen.

Da es ganz offensichtlich unmöglich ist, auf jedem Server des World Wide Web einen Filter-Agenten zu installieren – zumal für verschiedene Domänen auch verschiedene Filter-Agenten verwendet werden müssen –, muß in diesem Punkt anders vorgegangen werden.

Aus diesem Grund werden die Filter-Agenten eines Domänen-Experten als mobile Agenten realisiert, die sich zu entfernten Agenten-Plattformen bewegen und von dort aus Dokumente beliebiger Server untersuchen können. Durch dieses Konzept ist es auch nicht notwendig, daß auf allen Servern Agenten-Plattformen installiert sind.

Der Einsatz mobiler Filter-Agenten läßt sich in vier Abschnitte unterteilen:

3. **Zwischenergebnisse.** Die Agenten können Zwischenergebnisse an den Agenten-Koordinator schicken. Wenn z.B. ein Agent auf interessante Hyperlinks stößt, denen er momentan nicht selbst nachgehen will, kann er diese Informationen an den Agenten-Koordinator übermitteln, der dann entscheidet, ob er diese Aufgabe an einen Agenten delegiert, der bereits eine Aufgabe hat, oder ob er einen neuen Agenten losschickt.
4. **Ergebnismeldung.** Schließlich übermitteln die mobilen Filter-Agenten die Ergebnisse der Dokumenten-Bewertungen an den Domänen-Experten. Für jedes Dokument wird eine Bewertungszahl geliefert, die den Zusicherungsgrad, daß das Dokument zur Domäne des Domänen-Experten gehört, angibt. Wurden bei der Untersuchung eines Dokumentes bereits weitere Informationen zu dem Dokument ermittelt, so werden auch diese an den Domänen-Experten weitergegeben.

2.4 Wissensbasis eines Domänen-Experten

In der Wissensbasis eines Domänen-Experten werden sowohl Beschreibungen von Dokumenten, die zu der Domäne gehören, abgelegt, als auch Beschreibungen von Dokumenten, die nicht dazu gehören. Beschreibungen letzterer Dokumente beinhalten lediglich URL und Zeitmarke der letzten Modifikation des Dokumentes. Es ist notwendig, Beschreibungen zu Dokumenten, die nicht zur Domäne gehören, zu speichern, damit der Domänen-Experte Hinweise der konventionellen Suchmaschine auf solche Dokumente nicht jedesmal erneut durch Filter-Agenten untersuchen läßt.

Da der Domänen-Experte nur für Suchanfragen aus dem Kontext seiner Domäne benutzt wird, ist eine gewisse Vorauswahl getroffen, welche Suchanfragen an die konventionelle Suchmaschine gestellt werden. Die Wissensbasis bleibt dadurch auf einen Teil des World Wide Webs beschränkt. Es besteht also nicht die Gefahr, daß der Domänen-Experte letztlich doch das komplette World Wide Web in seiner Wissensbasis hält – die meisten Dokumente als nicht relevante gespeichert.

Das Wissen, das ein Domänen-Experte zu jedem Dokument seiner Domäne in den Dokumenten-Beschreibungen speichert, wird in *Facetten* angegeben. Mögliche Facetten sind z.B. „Schlüsselwort“, „Autor“ usw. Für jedes Dokument der Domäne werden für jede Facette die aktuellen Werte berechnet (durch die Komponenten Domänen-Tester und Dokumenten-Analyse des Domänen-Experten).

Dadurch, daß ein Domänen-Experte auf Dokumente einer Domäne beschränkt ist, bleibt seine Wissensbasis auf einen handhabbaren Umfang beschränkt. Dies ermöglicht dem Domänen-Experten, die Wissensbasis aktuell zu halten, indem er regelmäßig überprüft, ob Dokumente, zu denen er Beschreibungen besitzt, geändert (Zeitmarke) oder aus dem World Wide Web entfernt worden sind.

2.5 Domänenspezifische Funktionalität

2.5.1 Domänenspezifische Teile

Ein Domänen-Experte besitzt nur eine Komponente, die einen domänenspezifischen Teil zwingend benötigt: den Filter-Agenten.

Um die Konstruktion eines Filter-Agenten zu erleichtern, ist dieser aus zwei Komponenten aufgebaut: der allgemeinen *Format-Abstraktion* und dem domänen-spezifischen *Domänen-Tester* (siehe Abbildung 2.2.1). Die Format-Abstraktion abstrahiert von dem tatsächlichen Format eines Dokumentes (html, postscript, usw.), indem es das Dokument in eine einheitliche Darstellung bringt. Die Format-Abstraktion arbeitet wie ein Scanner, der eine Folge von Tokens erzeugt. Diese Tokens verwendet der Domänen-Tester um eine Bewertungszahl zu berechnen, die den Zusicherungsgrad, daß das Dokument zu der Domäne des Domänen-Experten gehört, angibt.

2.5.2 Optionale domänenspezifische Erweiterungen

Bestimmte Komponenten eines Domänen-Experten können (aber müssen nicht!) domänenspezifisch erweitert werden. Es handelt sich dabei um die Dokumenten-Analyse, die Hinweis-Sammlung und den Anfragen-Prozessor (siehe Abbildung 2.2.1).

Die Dokumenten-Analyse kann für die Ermittlung weiterer domänenspezifischer Merkmale eines Dokumentes (Schlüsselwörter und Meta-Schlüsselwörter) erweitert werden. Die Hinweis-Sammlung kann ergänzt werden, so daß sie auch auf andere externe Wissensquellen wie andere Domänen-Experten oder digitale Bibliotheken zugreift. Beim Anfragen-Prozessor kann die proaktive Arbeitsweise um domänenspezifische Vorgehensweisen erweitert werden.

2.6 Kooperation zwischen Domänen-Experten

Durch Kooperation zwischen Domänen-Experten lassen sich Effektivität und Effizienz der einzelnen, an der Kooperation beteiligten Domänen-Experten erhöhen.

Eine hierarchische Kooperation läßt sich relativ einfach dadurch realisieren, daß ein Domänen-Experte bei dessen Konstruktion/Konfiguration analog zum Zugriff auf eine konventionelle Suchmaschine auch eine Möglichkeit erhält, auf andere, bereits sich in Betrieb befindende Domänen-Experten zuzugreifen. Allerdings ist es bei dieser Form der Kooperation nicht möglich, daß jeder Domänen-Experte auf jeden zugreift.

Ein allgemeines Kooperationsprotokoll zur automatischen Kooperation zwischen Domänen-Experten würde es ermöglichen, daß jeder Domänen-Experte von jedem profitieren kann, und wäre darum sehr wünschenswert. Allerdings bedingte solches gewisse

semantische Festlegungen, an die sich alle Domänen-Experten – unabhängig von der Domäne – zu halten hätten. Facetten der Domänen-Experten wären dann z.B. nach diesen Festlegungen zu definieren. Ein allgemeines Kooperationsprotokoll würde dadurch die Flexibilität des Gesamtkonzeptes der Domänen-Experten ein gutes Stück einschränken. Diese Form der Kooperation ist darum seitens der Autoren der Originalliteratur für das Modell nicht geplant.

2.7 Zugriff auf Domänen-Experten

Um es Benutzern zu ermöglichen, alle Domänen-Experten leicht zu erreichen, benötigt man im Internet eine Stelle, an der sich alle Domänen-Experten registrieren. Dadurch kann in einem einfachen Ansatz ein Benutzer den für seine Suchanfrage geeigneten Domänen-Experten auswählen. Dieser einfache Ansatz ist jedoch im Grunde nicht ausreichend. Es ist notwendig, einen einheitlichen Zugriff auf alle Domänen-Experten zu realisieren, indem dem Benutzer eine einheitliche Suchmaske geboten wird, in die er seine Suchanfrage eingeben kann. Auswahl des geeigneten Domänen-Experten und Übersetzung der Suchanfrage für den ausgewählten Domänen-Experten erfolgen automatisch.

2.8 Quellen

Dieses Kapitel 2 beruht auf [Theilmann98b] und [Theilmann98a]. Wurde weitere Literatur verwendet, so ist dies an den entsprechenden Stellen mit Angabe des Verweises vermerkt.

Kapitel 3

Grundlagen und Dokumentenmodell

In diesem dritten Teil der Ausarbeitung erlaute ich das Problem der Generizität, das die im Anschluß genannte Aufgabenstellung meiner Diplomarbeit motiviert. Es folgen die Grundlagen für meinen Entwurf, sowie die Darstellung des generischen Dokumentenmodells.

Inhaltsangabe

3.1	Problem der Generizität	24
3.2	Aufgabenstellung der Diplomarbeit	25
3.3	Das Konzept - Grundlagen	25
3.4	Erstellung eines Domänen-Experten - Grundprinzip . . .	27
3.5	Das generische Dokumentenmodell	28

3.1 Problem der Generizität

Der kritische Punkt für die Praxistauglichkeit des in Kapitel 2 dargestellten Konzeptes der Domänen-Experten ist, in wie weit die Architektur des Domänen-Experten generisch gestaltet werden kann ([Theilmann98b, Abschnitt 3.5, erster Satz]), d.h. in wie weit die Erstellung der Domänen-Experten *generisch* erfolgen kann.

Diese Aussage läßt sich wie folgt erläutern: Beim Ansatz der Domänen-Experten ist vorgesehen, ein solches Suchwerkzeug auf eine Domäne zu beschränken und es mit domänenspezifischem Wissen auszustatten. Dies bedeutet, daß Domänen-Experten zweier (unterschiedlicher) Domänen sich voneinander unterscheiden. Die Frage ist damit: Wie wird ein Domänen-Experte mit domänenspezifischem Wissen ausgestattet? Wie werden die Teile eines Domänen-Experten erstellt, die von Domänen-Experte zu Domänen-Experte verschieden sind?

Ein einfacher Ansatz ist, ein *Framework* zu erstellen, d.h. ein Rahmenprogramm, das um wenige spezifische Komponenten ergänzt werden muß, um ein ablauffähiges Programm zu erhalten. Bei einem solchen Framework wären die domänenunabhängigen Teile eines Domänen-Experten vorgegeben, die domänenabhängigen zu ergänzen. Dieser Ansatz erfordert Kenntnisse der entsprechenden Programmiersprache. Das Hinzufügen der fehlenden Funktionalität ist potentiell fehlerträchtig.

Ein anderer Ansatz ist das *Generieren*. Hier wird die Konfiguration (möglichst vieler) spezifischer Aspekte der zu erstellenden Software durch eine relativ abstrakte, deklarative Beschreibung angegeben. Anhand dieser Beschreibung wird die Software automatisch erstellt. Bei Domänen-Experten wären bei Verwendung dieses Ansatzes die domänenspezifischen Teile deklarativ zu beschreiben. Dieser Ansatz ist relativ einfach in der Anwendung, d.h. auch weniger fehlerträchtig als ein Framework. Allerdings ist der Ansatz auch wesentlich aufwendiger und im allgemeinen Fall schwieriger in der Realisierung.

Wichtig für die Praxistauglichkeit des Konzeptes der Domänen-Experten ist, daß Domänen-Experten möglichst einfach erstellt werden können. Und hier liegen die Vorteile eindeutig beim Ansatz des Generierens.

Von daher erschließt sich auch obiger Satz, daß der kritische Punkt für die Praxistauglichkeit des Konzeptes der Domänen-Experten ist, in wie weit die Erstellung generisch gestaltet werden kann.

Der im folgenden gebrauchte Begriff *generisches Framework* steht für eine Kombination der Ansätze des Frameworks und des Generierens, d.h. ein Teil wird durch Generieren erzeugt, an bestimmten Stellen muß jedoch noch manuell Funktionalität ergänzt werden. Dieser Begriff wird verwendet weil noch nicht gesagt werden kann, ob Domänen-Experten vollständig durch Generieren erzeugt werden können.

3.2 Aufgabenstellung der Diplomarbeit

Im Rahmen dieser Diplomarbeit ist auf Basis des vorgegebenen Konzeptes der Domänen-Experten ein generisches Framework für die Erstellung von solchen spezialisierten Suchmaschinen auszuarbeiten und zu implementieren. Aufgrund der vorgegebenen Dauer der Diplomarbeit müssen damit erstellte Domänen-Experten allerdings nur über einen Teil der in Abschnitt 2.2 beschriebenen Architektur und internen Funktionalität verfügen.

Im einzelnen soll das generische Framework folgende Punkte abdecken:

- Erstellung eines generischen Dokumenten- und Anfragemodells,
- Aufbau eines themenspezifischen Indexes,
- automatische Generierung einer themenspezifischen Suchmaske
- Realisierung einer CGI-basierten Schnittstelle zur Abwicklung von Suchaufträgen und
- Abwicklung von Suchanfragen inklusive Berechnung der Relevanz der einzelnen Dokumente hinsichtlich der Suchanfrage.

Dabei sollen die speziellen Eigenschaften des Suchwerkzeuges so weit wie möglich deklarativ definierbar sein. Die Definition von themenspezifischen Filter-Verfahren ist nicht Teil dieser Arbeit.

3.3 Das Konzept - Grundlagen

3.3.1 Herleitung

„Jede Domäne besitzt Merkmale, die Elemente dieser Domäne auszeichnet.“

Schränkt man den Begriff „Merkmal“ nicht ein, sondern läßt auch triviale Merkmale zu, so scheint dieser Satz intuitiv wahr zu sein. Da jedoch ein Beweis fehlt, handelt es sich bei ihm lediglich um eine Behauptung.

Im Rahmen meiner Arbeit betrachte ich diese Behauptung als zutreffend und verwende sie als Grundlage meiner Arbeit. Diese Behauptung ist damit auch Grundlage des von mir erarbeiteten Konzeptes für ein Framework zur Erstellung von Domänen-Experten. Auf die Allgemeingültigkeit des von mir erarbeiteten Konzeptes gehe ich in Abschnitt 6.1 ein.

Wissenschaftliche Veröffentlichungen sind durch Angabe eines oder mehrerer Autoren, durch einen Titel usw. gekennzeichnet, abrufbare public domain Software durch Angabe eines Namens, einer Version, eines benötigten Betriebssystems usw.; dies alles sind Merkmale der jeweiligen Domäne.

Jedes Element einer Domäne besitzt eine Menge von konkreten Ausprägungen der Merkmale der Domäne. Diese Menge konkreter Ausprägungen der Merkmale der Domäne charakterisiert das Element.

Man kann diesen letzten Sachverhalt jedoch auch ein wenig anders formulieren, indem man sagt, daß die Menge konkreter Ausprägungen das Element der Domäne *spezifiziert*. Diese Spezifizierung kann eindeutig sein, d.h. nur dem einen Element zugeordnet werden können, oder mehrdeutig, d.h. für mehrere Elemente zutreffen.

In jedem Fall spezifiziert die Menge von Ausprägungen der Merkmale einen Teil der Elemente der Domäne. Betrachtet man diesen Sachverhalt nun dahingehend, daß die Angabe der Menge von Ausprägungen der Merkmale eine Suchanfrage darstellt, so liegt nun in Form dieses Konzeptes der Merkmale und Ausprägungen die theoretische Grundlage des Dokumenten- und Anfragemodells einer Suchmaschine vor, wobei von dem Anwendungsgebiet der Suchmaschine abstrahiert ist. Diese Abstraktion vom Anwendungsgebiet wiederum ermöglicht es, dieses Konzept der Merkmale und Ausprägungen als Grundlage eines Frameworks für spezialisierte Suchmaschinen zu verwenden.

3.3.2 Darstellung

Im letzten Absatz wurde das Konzept der Merkmale und Ausprägungen hergeleitet. Im folgenden soll nun noch eingehender dargestellt werden, wie das Konzept die theoretische Grundlage des Dokumenten- und Anfragemodells darstellt.

Ein Benutzer codiert seinen Wunsch durch Angabe einer Menge von Ausprägungen von Merkmalen einer speziellen Domäne, wodurch er eine Menge von Elementen der Domäne spezifiziert, nämlich die Elemente der Domäne, die durch die selben Ausprägungen der Merkmale charakterisiert sind, wie vom Benutzer angegeben. Die spezifizierten Elemente der Domäne sind einfach durch Vergleich der Ausprägungen der Merkmale der Elemente der Domäne und der vom Benutzer vorgegebenen Ausprägungen identifizierbar. Das Dokumentenmodell muß so gestaltet sein, daß es zu jedem gespeicherten Element der Domäne die für das Element geltenden Ausprägungen der Merkmale der Domäne speichert und auch entsprechende Vergleiche zuläßt.

Man beachte, daß die Aussagen des letzten Abschnittes anwendungsunabhängig sind.

3.3.3 Ausgestaltungsmöglichkeiten

Bei den bisherigen Aussagen zum Konzept der Merkmale und Ausprägungen wurde eine Suche dadurch beschrieben, daß die vom Benutzer angegebene Menge von Ausprägungen von Merkmalen eine Menge von Elementen der Domäne spezifiziert. Dies wurde durch die Aussage konkretisiert, daß die Menge der vom Benutzer angegebenen Ausprägungen von Merkmalen mit der Menge von Ausprägungen von Merkmalen jedes Elementes der Domäne verglichen wird. Beide Aussagen lassen Spielräume für die

Realisierung zu. So kann man im Rahmen dieser Aussagen festlegen, daß ein Element alle vom Benutzer angegebenen Ausprägungen besitzen muß, um zu den „spezifizierten“ zu gehören. Man kann aber genauso sagen, daß schon die Übereinstimmung einer Ausprägung genügt, um ein Element zu „spezifizieren“, Elemente, bei denen mehr Ausprägungen übereinstimmen, jedoch relevanter im Sinne der Anfrage sind. Des weiteren ist es möglich, Mischformen dieser zwei Festlegungen zu entwickeln: Ausprägungen, die ein Element der Domäne besitzen muß, um „spezifiziert“ zu werden, und Ausprägungen, die ein Element nicht besitzen muß, die jedoch bei Vorhandensein die Relevanz des Elementes bezüglich der Anfrage erhöhen.

Des weiteren ist es nicht notwendig, daß eine Benutzeranfrage für ein Merkmal nur eine Ausprägung nennen darf. Es ist möglich, die Angabe mehrerer Ausprägungen zu einem Merkmal zuzulassen, allerdings ist es auch hier nötig, Entscheidungen zur Semantik solcher Anfragen zu treffen: Muß ein Element alle angegebenen Ausprägungen eines Merkmals besitzen? Genügt das Aufweisen einer angegebenen Ausprägung, bei Enthalten von mehreren der angegebenen steigt die Relevanz des Elementes bezüglich der Benutzeranfrage?

Das Konzept der Merkmale und Ausprägungen liegt dem von mir erstellten Framework für spezialisierte Suchmaschinen zugrunde.

3.4 Erstellung eines Domänen-Experten - Grundprinzip

Die Erstellung eines Domänen-Experten mit Hilfe des von mir erstellten generischen Frameworks habe ich wie folgt realisiert: Das Framework ist als „domänenunabhängiger Domänen-Experte“ angelegt, der gestartet wird und ein Konfigurationsfile einliest. In diesem Konfigurationsfile sind die domänenspezifische Datenstruktur und Funktionalität beschrieben. Aufgrund dieser Angaben

- konfiguriert sich das Framework, wodurch es zum Domänen-Experten wird, und
- wird eine HTML-Seite für Benutzer-Suchanfragen generiert.

Im Anschluß an diese *Konfigurationsphase* geht das Framework - jetzt ein Domänen-Experte - in die *Betriebsphase*, in der der Domänen-Experte als Suchwerkzeug bereitsteht, Suchanfragen zu bearbeiten.

Abgesehen von dem gewählten Weg, einen Domänen-Experten zu erstellen, wurde auch noch ein anderer in Betracht gezogen. Bei diesem Weg war vorgesehen, einen *Domänen-Experten-Generator* zu erstellen, der aufgrund des oben erwähnten Konfigurationsfiles einen Domänen-Experten in Quellcode-Form erstellen würde. Der generierte Quellcode wäre dann lediglich noch mit Hilfe des Java-Compilers zu übersetzen gewesen.

Dieser alternative Weg wurde von mir als schwieriger zu realisieren eingeschätzt als der schließlich gewählte Weg und damit als aufwendiger. Aufgrund der begrenzten Zeit, die für eine Diplomarbeit zur Verfügung steht, habe ich mich darum für den zuerst geschilderten Weg entschieden.

Ein Domänen-Experte wird aus dem von mir im Rahmen meiner Diplomarbeit erstellten generischen Framework somit durch Konfigurieren erzeugt.

3.5 Das generische Dokumentenmodell

Bei dem von einem Domänen-Experten gespeicherten Elementen muß es sich nicht immer um Dokumente handeln. Ein Domänen-Experte für abrufbare public domain Software wird Informationen zu Software besitzen. Aus diesem Grund werde ich, wenn es um die von einem Domänen-Experten gespeicherten Entitäten geht, häufig von *Elementen* sprechen um damit diesen Sachverhalt zu betonen. Die Bezeichnung „Element“ steht damit für das, was mit dem Begriff „Dokument“ in diesem Zusammenhang genau genommen gemeint ist.

Das generische Dokumentenmodell ist die Grundlage dafür, daß mit Hilfe des Frameworks zwei Domänen-Experten erzeugt werden können, deren zwei Dokumenten-Typen sehr unterschiedlich sind. Es legt auch fest, wie ein Domänen-Experte bei dessen Erstellung auf den Dokumenten-Typ einer Domäne zugeschnitten wird.

Ein Domänen-Experte besitzt drei Datenstrukturen zur Speicherung von Informationen über Elemente seiner Domäne. Dabei handelt es sich zum einen um den

- Index. In ihm werden (wie schon in Abschnitt 1.3.2 erklärt) Dokumente *inhaltlich* wortorientiert abgelegt. Außerdem um den
- Dokumenten-Datensatz, in dem Informationen *zum* Dokument abgelegt werden (z.B. Sprache des Dokumentes). Die dritte Datenstruktur speichert
- welche Dokumente bereits untersucht und als nicht zur Domäne gehörig eingestuft worden sind.

Diese drei Datenstrukturen sind Teil des generischen Dokumentenmodells. Die folgenden Abschnitte gehen auf die Datenstrukturen unter diesem Gesichtspunkt ein.

3.5.1 Index

Ein Index speichert (wie schon in Abschnitt 1.3.2 erwähnt) Dokumente inhaltlich in der logischen Form

$$(\text{Wort}_i, \text{Dokument}_j, \text{Attribut}_{j_k}).$$

Da das Anfragemodell des zu erstellenden Frameworks auch Phrasen-Suche ermöglichen soll, muß eines der Attribute die Position des Wortes im Dokument sein. Weitere mögliche Attribute sind Merkmale der Domäne. Die logische Form der Einträge im Index ist demnach

$$(\text{Wort}_i, \text{Dokument}_j, \text{Position}_{j_i}, \text{Merkmal}_r).$$

Dieses Tupel zeigt, daß beim Index lediglich die Merkmale domänen-abhängig sind. Der Aspekt der Generizität schlägt sich beim Index darum nur darin nieder, welche Merkmale angegeben werden können.

Suchvorgänge im Index eines Domänen-Experten verwenden Wort und Merkmal als Parameter um Dokument und ggf. Position zu erhalten. Aus diesem Grund ist es sinnvoll, die Einträge des Index in der Form

$$(\text{Wort}_i, \text{Merkmal}_r, \text{Dokument}_j, \text{Position}_{j_i})$$

zu gestalten und zu ordnen. Diese Form entsteht aus der zuletzt angegebenen durch Umordnen der Stellen des Tupels.

3.5.2 Dokumenten-Datensatz

Ein Domänen-Experte legt für jedes ihm bekannte Element seiner Domäne einen Dokumenten-Datensatz (bei englischer Terminologie von mir *document data record* genannt) an. Im Dokumenten-Datensatz speichert er Informationen *über* das jeweilige Element. Es kann sich dabei sowohl um solche Informationen handeln, die explizit im Element enthalten sind, als auch um solche, die nicht explizit im Element enthalten sind.

Beispiele für solche Informationen sind bei wissenschaftlichen Veröffentlichungen das Datum derselben und das Format des Dokumentes, bei abrufbarer public domain Software der Autor der Software und das benötigte Betriebssystem.

Es gibt zwei Informationen, die jeder Domänen-Experte für die Elemente seiner Domäne in den jeweiligen Dokumenten-Datensätzen speichern muß. Es sind dies

1. die URL des jeweiligen Elementes der Domäne und
2. die Zeitmarke (bei englischer Terminologie *timestamp*) des jeweiligen Elementes der Domäne.

Die URL eines Elementes einer Domäne ermöglicht den Zugriff auf das Element auf dem jeweiligen Server, die Zeitmarke ermöglicht es einem Domänen-Experten zu entscheiden, ob seine Informationen zu dem Element der Domäne noch aktuell sind. Findet er bei dem durch die URL erreichbaren Element der Domäne nämlich eine neuere Zeitmarke, so weiß er, daß seine Informationen zu diesem Element veraltet sind.

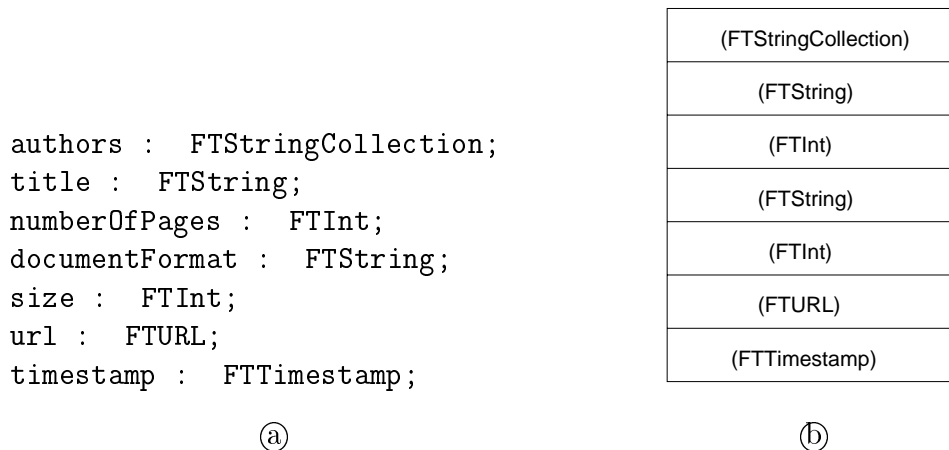


Abbildung 3.5.1: Beispiel für die Definition eines Dokumenten-Datensatzes (Ⓐ) und Veranschaulichung (!) der zugehörigen Datenstruktur (Ⓑ)

Diese beiden Informationen sind für jeden Domänen-Experten obligatorisch und damit domänenunabhängig. Abgesehen davon ist die Datenstruktur Dokumenten-Datensatz vollständig domänenabhängig und muß bei der Erstellung eines Domänen-Experten festgelegt werden.

Bei der Definition der Datenstruktur „Dokumenten-Datensatz“ für einen Domänen-Experten muß für jedes Feld dieser Datenstruktur ein symbolischer Bezeichner und ein Datentyp angegeben werden. Die Bezeichner sind frei wählbar, die möglichen Datentypen sind seitens des Frameworks vorgegeben. Mögliche Datentypen sind `FTString`, `FTStringCollection`, `FTInt`, `FTIntCollection`, `FTDate`, `FTDateCollection`, `FTInterval`, `FTIntervalCollection`, `FTTimestamp`, `FTTimestampCollection`, `FTURL` und `FTURLCollection`. Die Bezeichner der Datentypen geben an, für welche Arten von Werten sie stehen. „Collection“ steht dabei für eine geordnete Liste beliebiger Länge. So steht z.B. `FTStringCollection` für eine geordnete Liste von Zeichenketten. Weitere Datentypen sind denkbar.

Abbildung 3.5.1 zeigt eine (im Umfang reduzierte) Definition der Datenstruktur „Dokumenten-Datensatz“ und eine Veranschaulichung derselben¹ am Beispiel eines Domänen-Experten für wissenschaftliche Veröffentlichungen.

Wie in Abschnitt 2.4 erwähnt, werden die einzelnen Informationstypen zu einem Element einer Domäne bei Domänen-Experten auch *Facetten* genannt. Die Datentypen sind also gewissermaßen Facetten-Typen. Aus diesem Grund haben die für Felder eines Dokumenten-Datensatzes bereitgestellten Datentypen im Bezeichner alle das Präfix „FT“.

Das Framework ist so angelegt, daß die Ergänzung um weitere Facetten-Typen möglich ist - wenn auch nicht durch den Anwender des Frameworks.

¹Die Veranschaulichung soll dem besseren Verständnis dienen, sie soll jedoch keine bestimmte Realisierung nahelegen.

3.5.3 Verworfenene Dokumente

Elemente einer Domäne, die vom Domänen-Experten untersucht und als für die Domäne nicht relevant eingestuft worden sind, finden keine Aufnahme in den Datenbestand eines Domänen-Experten, in den akzeptierte Elemente aufgenommen werden. Damit diese verworfenen Elemente jedoch nicht bei jeder Nennung durch die konventionelle Suchmaschine erneut als unbekannte Elemente betrachtet und daher untersucht werden, müssen gewisse Informationen zu diesen Elementen gespeichert werden.

Jeder Domänen-Experte merkt sich von verworfenen Elementen seiner Domäne den URL und die Zeitmarke.

Durch diese Informationen kann bei einem Hinweis auf ein Dokument durch die konventionelle Suchmaschine vom Domänen-Experten festgestellt werden, ob das Element als verworfenes bekannt ist (URL), und - wenn ja - ob das Element auf seinem Server inzwischen verändert worden ist (Zeitmarke) und somit neu untersucht werden muß.

Dieser Teil des Dokumentenmodells ist domänenunabhängig.

Kapitel 4

Anfragebearbeitung

In diesem Kapitel wird zunächst darauf eingegangen, wie auf einen Domänen-Experten zur Beantwortung von Suchanfragen zugegriffen wird. Dies trägt zum Verständnis des im Anschluß dargestellten generischen Anfragemodells bei. Zuletzt erläutere ich, wie das Format definiert werden kann, in dem ein Domänen-Experte die Suchergebnisse dem Benutzer präsentiert.

Inhaltsangabe

4.1 Benutzerdialog	34
4.2 CGI - Grundlegendes	34
4.3 CGI-Schnittstelle für Domänen-Experten	35
4.4 Technisches zur CGI-Anbindung	37
4.5 Generizität und themenspezifische Suchmaske	39
4.6 Weitere Funktionalität	42
4.7 Die Anfragesprache	43
4.8 Durchführung von Vergleichen	45
4.9 Bewertung von Elementen der Domäne	46
4.10 Auswertung von Suchanfragen	47
4.11 Bearbeiten von Suchanfragen	52
4.12 Formatierung der Suchergebnisse	53
4.13 Festlegung des Suchergebnis-Formats	53
4.14 Quellen	55

4.1 Benutzerdialog

Ein Suchwerkzeug, das der Informationssuche im World Wide Web dient, muß in der Lage sein, Suchanfragen von Benutzern zu empfangen und zu beantworten, d.h. es muß für Benutzer zugreifbar sein.

Da ein Benutzer mit Hilfe eines solchen Suchwerkzeuges Hinweise auf im World Wide Web vorhandene Dokumente erhalten will, kann davon ausgegangen werden, daß er auch eine gewisse Anzahl der Dokumente, auf die er Hinweise erhält, abrufen will. Aus diesem Grund, sowie der einfachen Erreichbarkeit über das World Wide Web, bietet es sich an, den Zugriff auf Suchwerkzeuge ebenfalls über das WWW zu realisieren. Dieser Weg wurde für Domänen-Experten gewählt. Die notwendige Interaktivität wird dabei mit Hilfe des *Common Gateway Interface (CGI)* realisiert - was im World Wide Web ein gängiges Vorgehen ist.

Die folgenden Abschnitte gehen näher auf das Common Gateway Interface und den Zugriff auf Domänen-Experten ein.

4.2 CGI - Grundlegendes

Navigieren im World Wide Web (WWW) bedeutet, daß der Benutzer eine WWW-Seite von einem WWW-Server anfordert, der daraufhin diese Seite (falls vorhanden) dem Benutzer schickt, worauf dieser eine andere Seite von demselben oder einem anderen WWW-Server anfordert, usw. Diese Operationen verlaufen für den Benutzer transparent. Er steuert sie lediglich auf einer höheren Abstraktionsebene, in den meisten Fällen wahrscheinlich durch Mausklicks.

Sollen im WWW Suchdienste angeboten werden, so ist es nötig, daß ein Benutzer Daten an einen Server schickt, die über die Anforderung einer WWW-Seite hinausgehen. Es kann sich dabei z.B. um einen Begriff handeln, zu dem der Benutzer Informationen anfordern will. Dies kann über das *Common Gateway Interface (CGI)* erreicht werden.

Durch das Common Gateway Interface ist folgende Form der Interaktivität möglich (Abbildung 4.2.1): Der Benutzer fordert eine HTML-Seite an, die ein Formular enthält. Dieses Formular füllt er mit den gewünschten Angaben aus. Im Anschluß daran schickt er das Formular ab (1). Das Formular wird an den darin vermerkten Server übertragen (2), der ein ebenfalls im Formular vermerktes CGI-Skript startet (3). Das *CGI-Skript* ist ein Anwendungsprogramm. Es liest die Formular-Eintragungen des Benutzers, verarbeitet sie und generiert eine HTML-Seite die die Informationen enthält, die an den Benutzer gesendet werden sollen (4). Die HTML-Seite wird dem Server übergeben (5), der diese dann an den Benutzer weiterreicht (6).

Anmerkung: Zum besseren Verständnis sei noch auf zwei Sachverhalte hingewiesen.

- „Formular“ ist im letzten Absatz abstrakt gemeint, d.h. es sind nicht Formulare von HTML gemeint. Diese abstrakten Formulare werden jedoch durch HTML-Formulare realisiert.

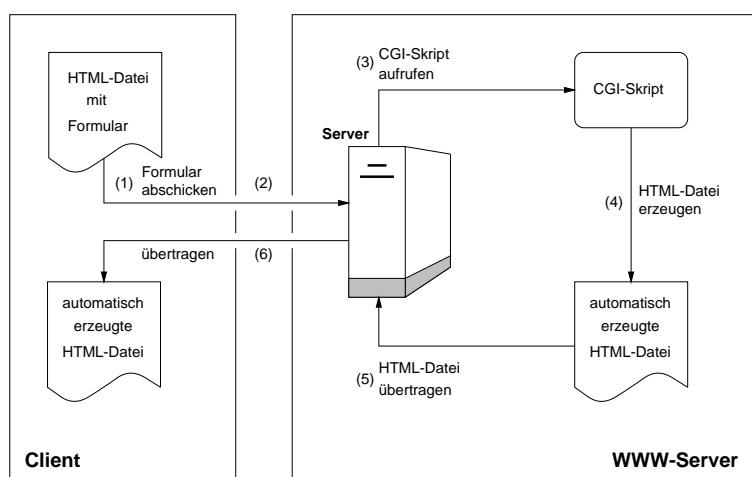


Abbildung 4.2.1: Interaktivität im World Wide Web mit dem Common Gateway Interface (CGI) (basierend auf [Schmidt98, Abbildung Seite 227])

- Es wird nicht das ganze Formular zum Server übertragen, sondern nur die vom Benutzer eingetragenen Daten zusammen mit internen Verwaltungsinformationen.

Es gibt zwei Wege, wie der WWW-Browser die Eingaben des Benutzers an den Server übergeben kann:

1. über die GET-Methode: Die vom Benutzer eingegebenen Daten werden an den URL des CGI-Skriptes angehängt. Dem CGI-Skript werden die Daten in der Umgebungsvariablen ‚QUERY_STRING‘ zur Verfügung gestellt.
2. über die POST-Methode: Die vom Benutzer eingegebenen Daten werden nicht an den URL angehängt, das CGI-Skript erreicht sie, indem es die Standard-Eingabe ausliest.

Da die meisten WWW-Server nur mit URLs umgehen können deren Länge 512 Zeichen nicht übersteigt, stößt die GET-Methode bei größerem zu übertragenden Datenvolumen an ihre Grenzen.

4.3 CGI-Schnittstelle für Domänen-Experten

4.3.1 Prinzipielle Realisierung

Das CGI ist im World Wide Web eine gängige Möglichkeit zur Realisierung von Interaktivität. Es bietet sich darum an, die Dienste eines Domänen-Experten über das CGI zugreifbar zu machen. Bei der Realisierung kann dann die vorgegebene, im Abschnitt 4.2 beschriebene Funktionalität verwendet werden.

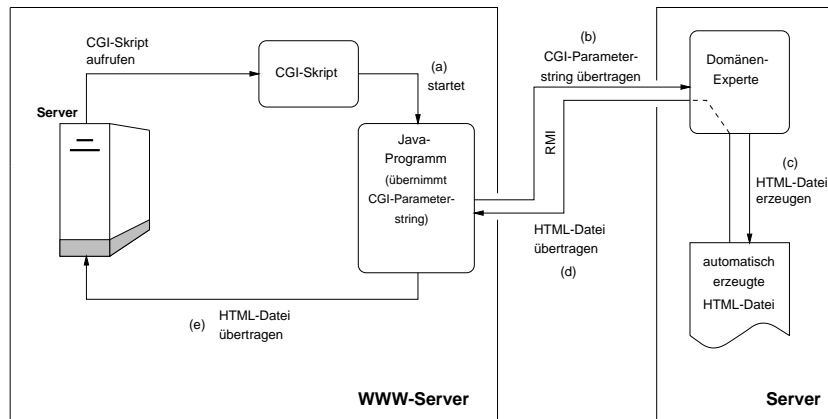


Abbildung 4.3.1: Datenaustausch zwischen CGI-Skript und Domänen-Experte

Der Ansatzpunkt für die Realisierung des Zugriffes auf einen Domänen-Experten durch das CGI unter Verwendung der vorgegebenen Funktionalität ist das CGI-Skript. Dabei stellt sich das konkrete Problem, wie die Anfrageparameter des Benutzers vom CGI-Skript an einen sich in Betrieb befindenden Domänen-Experten übergeben und die Suchresultate wieder von diesem übernommen werden können.

Mein Konzept löst dieses Kommunikationsproblem mit Hilfe der *Remote Method Invocation (RMI)* von Java. RMI gibt einem Objekt x die Möglichkeit, die Methode y eines Objektes z aufzurufen, wobei die Objekte x und z nicht zu dem gleichen Programm oder Prozeß gehören, ja nicht einmal auf dem gleichen Rechner lokalisiert sein müssen. Wie bei Aufruf von Methoden lokaler Objekte ist es bei RMI möglich, daß eine entfernte Methode bei Terminierung Daten zurückliefert. Der Aufruf einer entfernten Methode kann etwas abstrakter auch als Senden einer Nachricht mit ggf. anschließendem Empfangen einer Antwort angesehen werden.

CGI-Skript und Domänen-Experte tauschen die Daten, wie in Abbildung 4.3.1 dargestellt und im folgenden erklärt, aus.

Der Server ruft das CGI-Skript auf, welches ein kleines Java-Programm startet (a), das den CGI-Parameterstring übernimmt. Der Parameterstring wird von diesem Java-Programm (unter Verwendung von RMI) als Parameter einer Nachricht an den Domänen-Experten übermittelt (b). Der Domänen-Experte zerlegt den CGI-Parameterstring zur Ermittlung der Anfrageparameter, wertet die dadurch gegebene Suchanfrage aus, generiert eine HTML-Seite mit den Ergebnissen der Suche (c) und sendet diese als Antwort an das kleine Java-Programm zurück (d). Das Java-Programm übergibt die Antwortseite darauf an den Server (e), worauf wie gewohnt (s.o.) weiterverfahren wird.

Abbildung 4.3.2 stellt den Zugriff auf einen Domänen-Experten über das CGI vollständig dar.

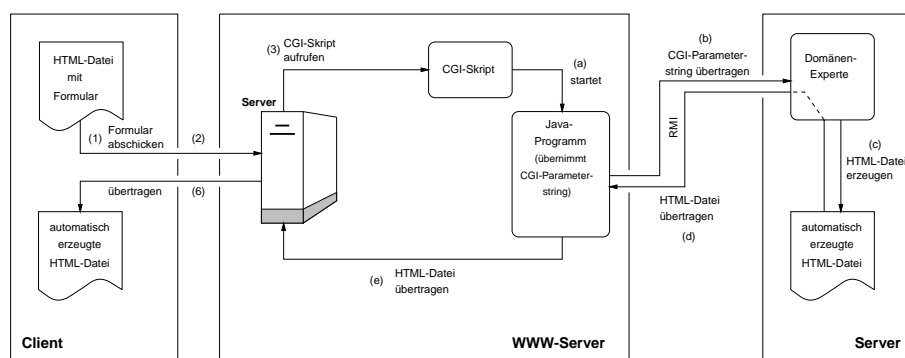


Abbildung 4.3.2: Zugriff auf einen Domänen-Experten

4.3.2 Übertragungsmethode

Es besteht die Möglichkeit, daß ein Domänen-Experte erzeugt werden soll, bei dem viele Merkmale der Domäne berücksichtigt werden. Berücksichtigt man weiter daß bei einer Suchanfrage an einen solchen Domänen-Experten für einen Großteil der Merkmale Ausdrücke angegeben werden, so ist es denkbar, daß 512 Zeichen zur Übertragung der Ausdrücke (inklusive URL und interner Verwaltungsinformationen) nicht ausreichen könnten.

Um das Konzept wegen dieser Grenze von 512 Zeichen Länge nicht einschränken zu müssen, wird zur Übertragung der Benutzereingaben an den Server die POST-Methode verwendet, die diese Beschränkung nicht kennt (siehe [Schmidt98, Seite 228]).

4.4 Technisches zur CGI-Anbindung

Die Ausführungen in diesem Abschnitt sind technisch und dienen zum besseren Verständnis des Abschnittes 4.5.

4.4.1 CGI und HTML

Die Hypertext Markup Language (HTML), mit der Seiten des World Wide Web erstellt werden, besitzt Anweisungen, durch die Interaktivität mit Hilfe des CGI sehr bequem realisiert werden kann - zumindest was das Eingeben und Abschicken von Daten seitens des Benutzers des World Wide Webs angeht.

HTML bietet die Möglichkeit, innerhalb einer HTML-Seite ein *Formular* zu definieren. Dieses Formular stellt den Rahmen für die Interaktivität mit dem CGI dar. Innerhalb des Formulars können u.a. Eingabefelder und Buttons definiert werden. Eingabefelder erlauben es dem Benutzer, Eingaben vorzunehmen. Ein möglicher Button sorgt bei Betätigung durch den Benutzer des Formulars dafür, daß die eingegebenen Daten an das entsprechende CGI-Skript abgeschickt werden, ein anderer löscht bei Betätigung

```

(1) <FORM ACTION=
      "http://www.informatik.uni-stuttgart.de/vsbin/search.cgi"
      METHOD="POST">
(2)   Authors:
(3)   <INPUT TYPE="text" SIZE="40" NAME="intid0">
(4)   <br>
(5)   Title:
(6)   <INPUT TYPE="text" SIZE="40" NAME="intid1">
(4)   <br>
(7)   <INPUT TYPE="submit" VALUE="Start Search">
(8)   <INPUT TYPE="reset" VALUE="Clear Fields">
(9) </FORM>

```

Abbildung 4.4.1: Beispiel für ein HTML-Formular

The image shows a browser window with a search form. The form has two text input fields. The first is labeled 'Authors:' and the second is labeled 'Title:'. Below the input fields are two buttons: 'Start Search' and 'Clear Fields'.

Abbildung 4.4.2: Darstellung (der logischen Struktur) des Formulars von Abbildung 4.4.1 in einem WWW-Browser

durch den Benutzer des Formulars alle Eintragungen in Eingabefeldern des Formulars.

Die HTML-Befehle zur Realisierung von Formular, solchen Eingabefeldern und Buttons sind sehr einfach.

Abbildung 4.4.1 enthält ein Beispielformular in HTML-Quellcode. Abbildung 4.4.2 zeigt dieses Formular wie ein WWW-Browser dessen logische Struktur darstellt.

Erläuterungen zum Code von Abbildung 4.4.1: Anweisung (1) definiert den Kopf des Formulars mit Angabe des CGI-Skriptes, das die eingegebenen Daten verarbeiten soll (**ACTION**), und der Art, wie die Daten zum CGI-Skript geschickt werden sollen (**METHOD**). Anweisungen (3) und (6) definieren Eingabefelder mit Länge (**SIZE**) und internem Bezeichner (**NAME**). (7) ist der Befehl für einen Button, dessen Betätigung bewirkt, daß die Daten an das bearbeitende CGI-Skript abgeschickt werden. Auch (8) definiert einen Button, jedoch bewirkt dieser, daß die Eingabefelder des Formulars gelöscht werden. **VALUE** legt bei den beiden Buttons jeweils die Beschriftung der Buttonfläche fest. Anweisung (9) steht für das Ende des Formulars. Anweisung (4) bewirkt einen Zeilenumbruch. Durch (2) und (5) wird genau die genannte Zeichenfolge ausgegeben.

4.4.2 Aufbau eines CGI-Parameterstrings

Im Rahmen dieser Diplomarbeit beschränke ich mich auf die in Abbildung 4.4.2 verwendeten Eingabefelder. Dieser Abschnitt beschreibt den CGI-Parameterstring, wie er unter dieser Voraussetzung aufgebaut sein wird.

Wird ein Formular „abgeschickt“, so werden die Inhalte der Eingabefelder mit den jeweiligen internen Bezeichnern der Eingabefelder in einen String geschrieben, der dann an das im Formular angegebene CGI-Skript geschickt wird.

Der String ist nach folgender Syntax aufgebaut (die Syntax ist in der Extended Backus-Naur-Form notiert):

```
CGIParameterstring = EinFeld {"&" EinFeld}.  
EinFeld = InternerBezeichnerDesFeldes "=" [BenutzereingabeDesFeldes].
```

Beispiel: Für das Formular von Abbildung 4.4.2 und die Eingabe

Knuth

im Feld nach „Authors:“, wäre

intid0=Knuth&intid1=

der CGI-Parameterstring. □

Es sei an dieser Stelle besonders darauf hingewiesen, daß die Inhalte der Eingabefelder in genau der Reihenfolge in den CGI-Parameterstring geschrieben werden, in der die Eingabefelder im Formular definiert sind. Diese Tatsache wird in Abschnitt 4.5 verwendet werden.

4.5 Generizität und themenspezifische Suchmaske

Die folgenden Abschnitte behandeln das generische Anfragemodell. Dies ist eine umfangreiche Thematik. Sie umfaßt Festlegungen, wie Anfragefunktionalität für konkrete Domänen-Experten vom Benutzer des Framework definiert werden kann, die Generierung einer themenspezifischen Suchmaske, Syntax und Semantik der Anfragesprache, sowie Auswertungsfunktionalität mit Bestimmung des Relevanzgrades eines Dokumentes bezüglich einer Anfrage.

Dieser Abschnitt 4.5 führt die Konzepte zur Definition der Anfragefunktionalität und zur Generierung der themenspezifischen Suchmaske für einen Domänen-Experten ein.

Man beachte, daß es im folgenden um die Erstellung von Domänen-Experten geht. Ausführungen, die die Bearbeitung von Anfragen betreffen, dienen zur Motivierung der Konzepte.

In Abschnitt 3.3 wurden die Grundlagen meines Konzeptes zur Realisierung eines Frameworks für Domänen-Experten dargestellt. Dabei wurde aufgezeigt, daß es bei der Formulierung von Anfragen an einen Domänen-Experten darum geht, Angaben für Merkmale der Domäne zu machen.

Es bietet sich von daher an, für Merkmale, die dem Domänen-Experten bekannt sind, dem Benutzer in der Suchmaske Eingabefelder zu präsentieren. In ein Eingabefeld x kann der Benutzer dann die Angaben eintragen, die für die von ihm gewünschten Elemente der Domäne in punkto Merkmal x gelten sollen.

Erhält der Domänen-Experte die Suchanfrage-Parameter, so enthalten diese dann genau diese Informationen bezüglich den gesuchten Elementen der Domäne: für welche Merkmale welche Ausprägungen gelten sollen. Um diese Ausdrücke auswerten zu können muß jedoch bekannt sein,

1. wo die Ausprägungen der Merkmale von Elementen der Domäne abgelegt sind, d.h. im Index oder im Dokumenten-Datensatz des Domänen-Experten, und
2. ggf. wie die Vergleiche durchzuführen sind, d.h. durch
 - exakten Stringvergleich oder
 - durch Stringvergleich unter Ignorieren der Groß- und Kleinschreibung oder durch
 - Patternmatching.

Diese Informationen sind von der Domäne und den einzelnen Merkmalen abhängig, wobei es für den Ersteller eines Domänen-Experten gewisse Freiheitsgrade gibt (bei einem Merkmal „Schlüsselwörter“, d.h. bekannte Wörter mit definierter Bedeutung, ist die Vergleichsmethode mit exaktem Stringvergleich genauso denkbar wie die mit Ignorieren der Groß- und Kleinschreibung).

Mein Konzept sieht vor, daß der Ersteller eines Domänen-Experten Eingabefelder für die Suchmaske definiert und zu jedem Eingabefeld angibt,

1. auf welches Merkmal der Domäne sich die Eingaben beziehen werden,
2. ob die Ausprägungen der Merkmale im Index oder in Dokumenten-Datensätzen zu suchen sind und
3. ggf. wie die Vergleiche durchzuführen sind.

Abbildung 4.5.1 zeigt ein Beispiel für solche Definitionen anhand eines Domänen-Experten für wissenschaftliche Veröffentlichungen (Ausschnitt). Dabei steht jede Zeile für die Definition eines Eingabefeldes mit der zugehörigen Auswertungsfunktionalität. Die erste Angabe definiert den Bezeichner eines Eingabefeldes, er erscheint in der Suchmaske vor dem Eingabefeld. Die vom Bezeichner durch das Komma getrennte Zahl legt

```
"Authors", 40 : indexSearch{author}
                , conventionalSearchEngineCriterion;
"Title", 40 : indexSearch{title};
"Number of Pages", 20 : documentDataRecordSearch{numberOfPages};
"Language", 20 : documentDataRecordSearch{language}{matching};
"Document Format", 20 : documentDataRecordSearch{docformat}{liberal};
"URL", 30 : documentDataRecordSearch{url}{matching};
"Timestamp", 30 : documentDataRecordSearch{timestamp}
```

Abbildung 4.5.1: Beispiel für eine Definition von Anfragefunktionalität und themenspezifischer Suchmaske

die Länge des Eingabefeldes fest (Abbildung 4.5.2 zeigt die von den Angaben in Abbildung 4.5.1 definierte Suchmaske). Nach dem Doppelpunkt legt `indexSearch` oder `documentDataRecordSearch` fest, ob die Ausprägungen des entsprechenden Merkmals im Index oder in Dokumenten-Datensätzen zu suchen sind. Der Bezeichner in dem folgenden Paar geschwungener Klammern gibt an, um welches Merkmal es geht. Er spezifiziert damit im Fall von `indexSearch` ein im Index gespeichertes Merkmal mit dem gleichen Bezeichner, im Fall von `documentDataRecordSearch` das Teilfeld jedes Dokumenten-Datensatzes, das den gleichen Bezeichner besitzt. Das zweite Paar geschwungener Klammern ist nur im Fall von `documentDataRecordSearch` möglich und definiert wie die Vergleiche durchzuführen sind. Im Beispiel von Abbildung 4.5.1 steht `liberal` für das Ignorieren von Groß- und Kleinschreibung und `matching` für Vergleich mit Patternmatching. Ein Strichpunkt trennt die Definitionen. (Die erste Definition in Abbildung 4.5.1 enthält nach der schließenden geschwungenen Klammer ein Komma, gefolgt von `conventionalSearchEngineCriterion`. Was es damit auf sich hat, wird in Abschnitt 4.6.2 erklärt werden. An dieser Stelle hier sind Komma und dieses Schlüsselwort ignorierbar.)

Der Domänen-Experte benötigt diese Informationen (bis auf die Länge der Eingabefelder) für die Bearbeitung von Suchanfragen während der Betriebsphase. Er legt sie darum in einer Tabelle ab, wie sie in Abbildung 4.5.3 für die Definitionen von Abbildung 4.5.1 dargestellt ist.

Die Bezeichner der Eingabefelder der Suchmaske werden benötigt, um einen Benutzer des Domänen-Experten auf eventuelle Fehler in seiner Suchanfrage hinweisen zu können. Durch Angabe des jeweiligen Eingabefeldes werden Fehlermeldungen hilfreicher und damit ein Domänen-Experte benutzerfreundlicher.

Es sei darauf hingewiesen, daß der Reihenfolge der Definitionen im Konzept eine wichtige Bedeutung zukommt. Die Reihenfolge der Definitionen bestimmt die Reihenfolge der Eingabefelder in der Suchmaske und deren interne (!) Bezeichner, die im CGI-Suchstring enthalten sind. Die Eingabefeld-Inhalte werden in der Reihenfolge der Eingabefelder in den CGI-Suchstring übernommen, der in der Betriebsphase an den Domänen-Experten übermittelt wird (siehe Abschnitt 4.4.2). Durch die Reihenfolge

Abbildung 4.5.2 zeigt eine Suchmaske mit folgenden Eingabefeldern:

- Authors
- Title
- Number of Pages
- Language
- Document Format
- URL
- Timestamp

Unter den Eingabefeldern befinden sich zwei Buttons: "Start Search" und "Clear Fields".

Abbildung 4.5.2: Suchmaske nach den Definitionen von Abbildung 4.5.1

der Eingabefeld-Inhalte im CGI-Suchstring ist dann die Zuordnung zu der Auswertungsdefinition in der internen Tabelle des Domänen-Experten (z.B. Abbildung 4.5.3) möglich. Abbildung 4.5.4 auf Seite 56 verdeutlicht diesen Sachverhalt. Die erwähnten internen Bezeichner der Eingabefelder, die im CGI-Suchstring erscheinen, dienen der Fehlerkontrolle. Steht ein interner Bezeichner nicht an der Stelle der Reihenfolge im CGI-Suchstring, an der er stehen müsste, ist ein Fehler aufgetreten, weswegen eine korrekte Bearbeitung der Suchanfrage nicht mehr gewährleistet werden kann. Die Bearbeitung der Suchanfrage wird mit einer Fehlermeldung abgebrochen.

4.6 Weitere Funktionalität

4.6.1 Indexsuche unabhängig von Merkmalen

Soll im Index eines Domänen-Experten nach Wörtern gesucht werden, und zwar unabhängig davon, ob sie Teil einer Ausprägung eines Merkmales sind, oder nicht, so ist dies realisierbar, indem man in den geschweiften Klammern nach dem Schlüsselwort `indexSearch` kein Merkmal angibt, also

```
... : indexSearch{}
```

definiert. In dem zugehörigen Eingabefeld eingegebene Ausdrücke werden dann in dieser Art ausgewertet.

indexSearch	author		true	Authors
indexSearch	title		false	Title
documentDataRecordSearch	numberOfPages		false	Number of Pages
documentDataRecordSearch	language	matching	false	Language
documentDataRecordSearch	docformat	liberal	false	Document Format
documentDataRecordSearch	url	matching	false	URL
documentDataRecordSearch	timestamp		false	Timestamp

↑
↑
↑
↑
↑

Teil des Datenbestandes, in dem
gesucht werden soll

Attribut oder Teil des
Dokumenten-Datensatzes

Vergleichsart

Relevanz der Eingaben
für Anfragen an die
konventionelle Such-
maschine

Beschriftung des
Eingabefeldes

Abbildung 4.5.3: Interne Tabelle mit Informationen zur Bearbeitung von Suchanfragen

4.6.2 Anfragen an die konventionelle Suchmaschine

Will ein Domänen-Experte im Laufe der Bearbeitung einer Suchanfrage eines Benutzers Hinweise auf zur Anfrage passende Elemente der Domäne von der konventionellen Suchmaschine einholen (siehe Abschnitt 2.2), so muß er eine geeignete Suchanfrage an sie formulieren. Dazu dient das Schlüsselwort `conventionalSearchEngineCriterion`. Erhält ein Eingabefeld bei der Definition dieses Schlüsselwort (wie in Abbildung 4.5.1 das Eingabefeld für Autoren), so werden die vom Benutzer eingetragenen Ausdrücke für die Formulierung der Anfrage an die konventionelle Suchmaschine verwendet.

Es muß kein Feld, können jedoch beliebig viele Felder mit dem Schlüsselwort `conventionalSearchEngineCriterion` definiert werden.

4.7 Die Anfragesprache

Befindet sich der Domänen-Experte in der Betriebsphase, so kann ein Benutzer über das World Wide Web Suchanfragen bezüglich Elementen der Domäne des Domänen-Experten an diesen stellen. Dazu verwendet er die beim Erzeugen des Domänen-Experten generierte Suchmaske. In deren Eingabefelder trägt er die Anforderungen ein, die er jeweils bezüglich der Merkmale stellen will.

Für die Formulierung der Anforderungen ist eine Anfragesprache vorgegeben, die für alle Domänen-Experten, also unabhängig von der Domäne, gültig ist. Sie wird im folgenden Abschnitt 4.7.1 beschrieben. Abschnitt 4.7.2 enthält die im wesentlichen formale Beschreibung der lexikalischen Elemente und der Syntax der Sprache, sowie informell die wichtigsten Angaben zur Semantik.

4.7.1 Beschreibung

Der Benutzer eines Domänen-Experten formuliert eine Suchanfrage, indem er in die Eingabefelder der Suchmaske *Anforderungen* einträgt. In einem Eingabefeld können keine, eine oder mehrere Anforderungen eingetragen werden. Es müssen nicht alle Eingabefelder ausgefüllt werden, es genügt bereits in ein Eingabefeld eine Anforderung einzutragen.

Eine *Anforderung* besteht aus einem optionalen *Präfix* gefolgt von einem Ausdruck. Bei dem Ausdruck kann es sich um einen *Phrasen-Ausdruck* oder um einen *Wort-Ausdruck* handeln. Ein Wort-Ausdruck kann mit einem Relationen-Operator beginnen. Abbildung 4.7.1 zeigt und beschreibt einige solche Anforderungen.

Mit dem Präfix + kann ein Benutzer festlegen, daß nur solche Elemente der Domäne im Anfrageergebnis enthalten sein dürfen, die den nachfolgenden Ausdruck erfüllen. Das Präfix - bewirkt, daß Elemente der Domäne, die den nachfolgenden Ausdruck erfüllen, nicht im Anfrageergebnis enthalten sein dürfen. Wird kein Präfix angegeben, so wird dadurch ausgedrückt, daß Elemente der Domäne, die den nachfolgenden Ausdruck erfüllen, dann im Anfrageergebnis enthalten sein sollen, wenn andere Anforderungen des Benutzers dies nicht ausschließen.

Ein Wort-Ausdruck besteht aus einer zusammenhängenden Folge von Zeichen, optional eingeleitet durch einen Relationen-Operator. Ein Relationen-Operator kann <, <=, >, >=, <> oder = sein.

Ein Phrasen-Ausdruck besteht aus mehreren zusammenhängenden Folgen von Zeichen, voneinander getrennt durch mindestens ein Leerzeichen. Der Phrasen-Ausdruck besitzt als erstes und letztes Zeichen ein ".

Phrasen-Ausdrücke dienen zur Durchführung von Phrasen-Suchen.

In Wort- und Phrasen-Ausdrücken erlaubte Zeichen sind die ASCII-Zeichen zwischen 33 und 125 je einschlieslich, mit Ausnahme der Zeichen +, -, ", >, <, = und ^ . Diese Zeichen können jedoch durch Voranstellen eines ^ ebenfalls an diesen Stellen verwendet werden.

Leerraum zwischen Anforderungen kann mit Leerzeichen oder horizontalen Tabulatoren eingefügt werden.

4.7.2 Übersicht

Lexikalische Elemente werden im folgenden informell oder durch reguläre Ausdrücke angegeben. Syntax ist in der Extended Backus-Naur-Form (EBNF) notiert.

Lexikalische Elemente

Space → Leerzeichen

`RecordSearch` angegeben werden. Erfolgt dies nicht, so wird eine default-Vergleichsart verwendet.

Welche Vergleichsarten und Relationen-Operatoren möglich sind und ob Phrasen-Suche möglich ist, hängt von dem Typ des Feldes des Dokumenten-Datensatzes ab, in dem die Ausprägung des Merkmals abgelegt ist. Tabelle 4.8.1 gibt eine Übersicht.

4.9 Bewertung von Elementen der Domäne

Suchanfragen bestehen aus Anforderungen, die sich auf Merkmale der Domäne beziehen.

Angenommen, ein Element a der Domäne kann eine Anforderung einer Suchanfrage n -mal erfüllen, ein Element b der Domäne die gleiche Anforderung der Suchanfrage m -mal. Gilt $n > m$, so kann man vermuten, daß das Element a mehr Relevanz bezüglich der Suchanfrage besitzt als Element b .

Beispiel: Domänen-Experte für wissenschaftliche Veröffentlichungen. Es werden Dokumente zum Thema (Software-)Agenten gesucht. Beauftragt man den Domänen-Experten damit, nach Dokumenten zu suchen, die das Wort „Agent“ enthalten, so ist anzunehmen, daß Dokumente, die dieses Wort öfter enthalten, sich eingehender mit diesem Thema befassen, als solche, die dieses Wort nur einmal oder zweimal enthalten. Erstere Dokumente werden damit für den Suchenden größere Relevanz hinsichtlich der Anfrage besitzen als letztere. \square

Term Frequency Invers Document Frequency (TF/IDF) [Grollmuss98] ist ein Verfahren, das versucht, solches in konkrete Werte zu fassen.

fre_j : Anzahl, wie oft Element j die Anforderung erfüllt

$$Bewertung_{Element\ k} = \frac{fre_k}{\sum_{Elemente\ der\ Domäne} fre_i}$$

Man beachte, daß im Nenner des Bruches über alle Elemente der Domäne summiert wird, d.h. auch über Element k .

Ein Element erzielt eine umso höhere Bewertung, je häufiger es selbst eine Anforderung erfüllt und je seltener andere Elemente die Anforderung erfüllen.

Ein anderer Sachverhalt ist auch geeignet, Elementen einer Domäne unterschiedliche Relevanz bezüglich einer Anfrage zuzuordnen: Angenommen ein Element x der Domäne entspricht r (nicht zwingenden) Anforderungen einer Suchanfrage und ein Element y der Domäne entspricht s (nicht zwingenden) Anforderungen derselben Suchanfrage. Gilt $r > s$, so kann man sagen, daß Element x den Spezifikationen der Suchanfrage mehr entspricht als Element y . Man könnte es aber auch hier so ausdrücken, daß Element x relevanter bezüglich der Suchanfrage ist als Element y .

Wird für jede Anforderung und für jedes Element, das dieser entspricht, ein Wert nach TF/IDF berechnet (s.o.), so lassen sich die gerade dargestellten Relevanzunterschiede einfach dadurch berücksichtigen, daß die TF/IDF-Werte je Element aufsummiert werden. Bei Element x werden dann die r TF/IDF-Werte aufsummiert, bei Element y eben die s TF/IDF-Werte.

Ein Domänen-Experte berechnet bei einer Suchanfrage für Elemente der Domäne Relevanzgrade bezüglich der Suchanfrage. Er verwendet dabei die geschilderten Verfahren.

Man beachte, daß die geschilderten Verfahren sowohl bei Suchvorgängen im Index, als auch bei Suchvorgängen in den Dokumenten-Datensätzen anwendbar sind.

Betrachtet man Bewertungen bei Suchvorgängen im Index und Suchvorgängen in Dokumenten-Datensätzen, so fällt auf: Bei Suchvorgängen in Dokumenten-Datensätzen ist zu erwarten, daß der Zähler der Berechnungsvorschrift für TF/IDF-Werte häufig 1 wird. Dadurch werden auch die Werte, die für den Nenner der Berechnungsvorschrift addiert werden, häufig nur 1 sein. Bei Suchvorgängen im Index sind für Zähler und Summanden im Nenner höhere Werte zu erwarten. Es kann darum nicht a priori davon ausgegangen werden, daß sich Bewertungen von Suchvorgängen im Index von denen von Suchvorgängen in Dokumenten-Datensätzen aus prinzipiellen Gründen signifikant unterscheiden werden. Eine einfache Aufsummierung von TF/IDF-Werten unabhängig davon, ob sie bei Suchvorgängen im Index oder bei Suchvorgängen in Dokumenten-Datensätzen ermittelt wurden, kann von daher als geeignet betrachtet werden.

Ein Domänen-Experte geht nun genau so vor, daß er TF/IDF-Werte eines Elementes bei einer Anfrage unabhängig davon, ob sie bei Suchvorgängen im Index oder bei Suchvorgängen in Dokumenten-Datensätzen erzeugt wurden, aufsummiert.

Eine genauere Untersuchung der Unterschiede von TF/IDF-Werten, die bei Suchvorgängen im Index ermittelt wurden, zu TF/IDF-Werten, die bei Suchvorgängen in Dokumenten-Datensätzen ermittelt wurden, blieb im Rahmen dieser Diplomarbeit außen vor.

4.10 Auswertung von Suchanfragen

4.10.1 Prinzipieller Algorithmus

Im Abschnitt 4.9 wurde die Bewertung von Elementen einer Domäne bezüglich einer Suchanfrage dargestellt. Diesem Bewertungsmechanismus übergeordnet existiert jedoch ein anderer Selektionsmechanismus, der noch stärker die Auswahl von Elementen einer Domäne für eine Suchanfrage steuert. Dieser Selektionsmechanismus ergibt sich durch die in Abschnitt 4.7.1 eingeführten semantischen Präfixe.

Durch die semantischen Präfixe wird gesteuert, *welche* Elemente der Domäne zur Suchanfrage passen. Die im Abschnitt 4.9 dargestellte Bewertung von Elementen einer Domäne hinsichtlich einer Suchanfragen erlaubt es, Werte zu ermitteln, die für

Elemente, die zur Suchanfrage passen, Aussagen darüber machen, wie gut sie zu der Suchanfrage passen. Insofern ist der Selektionsmechanismus, der sich durch die semantischen Präfixe ergibt, dem Verfahren zur Bewertung von Elementen bezüglich einer Suchanfrage übergeordnet, was die Rolle als Selektionsmechanismus betrifft.

Nachfolgend wird ein Algorithmus zur Auswertung von Suchanfragen angegeben, der die semantischen Präfixe berücksichtigt.

Algorithmus 4.10.1: Auswertung von Suchanfragen (prinzipiell)

```

(1.)   begin
(2.)      $M_+ := M_{I+} \cap M_{D+}$ 
(3.)      $M_0 := M_{I0} \cup M_{D0}$ 
(4.)      $M_{+0} := (M_+ - (M_+ \cap_n M_0)) \cup (M_+ \cap M_0)$ 
(5.)      $M_- := M_{I-} \cup M_{D-}$ 
(6.)      $M := M_{+0} - M_-$ 
(7.)   end

```

Dieser Algorithmus beschreibt, wie die hinsichtlich einer Suchanfrage relevanten Elemente einer Domäne ermittelt werden können. Die dabei verwendeten Abkürzungen M_{xy} sind in Tabelle 4.10.1 auf Seite 59 erklärt. Man kann sich die Elemente der Mengen M_{xy} bei diesem Algorithmus am besten als 2-Tupel bestehend aus einem Element der Domäne und dessen Bewertung nach TF/IDF vorstellen.

Bei dem im Algorithmus beschriebenen Vorgehen werden die Lösungsmengen (inklusive Bewertungen) nach Auswertungsort (Index oder Dokumenten-Datensätze) und semantischem Präfix getrennt ermittelt. Anschließend werden die Lösungsmengen durch Mengenoperationen zusammengeführt, wobei die durch die semantischen Präfixe gegebene Funktionalität realisiert wird und die bei verschiedenen Auswertungen ermittelten Bewertungen der Elemente je Element aufsummiert werden.

Die Mengenoperationen führen die entsprechenden allgemeinbekannten Mengenoperationen durch (\cap_n verhält sich bezüglich der Mengenoperation genau wie die Schnittmenge \cap). Darüber hinaus werden bei den Mengenoperatoren \cup und \cap verschiedene Bewertungen eines Elementes durch Aufsummieren zusammengefaßt. Bei den Mengenoperatoren \cap_n und $-$ werden auf den Bewertungen keine Operationen ausgeführt.

Erläuterungen zu den Anweisungen des Algorithmus im einzelnen:

1. (begin)
2. M_+ ergibt sich durch $M_{I+} \cap M_{D+}$, da nur die Elemente der Schnittmenge die Ausdrücke von A_{I+} und A_{D+} erfüllen. Elemente der Schnittmenge kommen in beiden Mengen vor; ihre Bewertungen werden bei der Schnittmengenbildung aufsummiert.

3. Für die Berechnung von M_0 kann die Mengenvereinigung verwendet werden, da - was Anforderungen angeht, die kein semantisches Präfix enthalten - ein Element zur Lösungsmenge gehört, wenn es mindestens einen Ausdruck solcher Anforderungen erfüllt. Bewertungen gleicher Elemente werden bei der Mengenvereinigung aufsummiert.
4. Soll M_{+0} als Lösungsmenge aus den Mengen M_+ und M_0 ermittelt werden, so können darin nur Elemente aus M_+ enthalten sein (sofern M_+ nicht leer ist); Elemente aus M_0 können lediglich die Bewertungen von Elementen aus M_+ verbessern. Jedes Element von M_+ , das auch in M_0 enthalten ist, muß also in M_{+0} gewissermaßen durch sich selbst ersetzt werden, allerdings mit der Summe der Bewertungen in den beiden Mengen.
5. M_{I-} enthält die Elemente, die aufgrund von Auswertungen im Index als für die Lösungsmenge nicht erlaubt identifiziert wurden. M_{D-} enthält Elemente für die das gleiche gilt, festgestellt allerdings aufgrund von Auswertungen in Dokumenten-Datensätzen. Ist ein Element aufgrund Zugehörigkeit zur einen dieser beiden Mengen für die Lösungsmenge nicht zugelassen, so ist es unwichtig, ob es auch in der anderen Menge enthalten ist. Zur Zusammenfassung der beiden Mengen kann darum die Mengenvereinigung verwendet werden. Sowohl für M_{I-} und M_{D-} , als auch für M_- müssen keine Bewertung der Elemente berechnet bzw. berücksichtigt werden.
6. Zur Berechnung der Lösungsmenge (M) für die Suchanfrage müssen aus M_{+0} noch die Elemente entfernt werden, die nicht in der Lösungsmenge enthalten sein dürfen (M_-). Dafür eignet sich die Mengendifferenz. Die Bewertungen der Elemente müssen bei dieser Operation nicht beachtet, d.h. verändert werden.
7. (end)

4.10.2 Konkretisierung des Auswertungsalgorithmus

Der in Abschnitt 4.10.1 besprochene prinzipielle Algorithmus berücksichtigt noch nicht alle Auswertungssituationen, die sich ergeben können. Wenn eine Suchanfrage keine Anforderung mit + als semantischem Präfix enthält, so arbeitet der Algorithmus nicht korrekt. In diesem Abschnitt wird dies dargestellt und ein Algorithmus angegeben, der diesen Mangel behebt und dabei den prinzipiellen Algorithmus konkretisiert.

Man muß sich vor Augen halten, daß sich je nach Suchanfrage völlig verschiedene Situationen ergeben können: Enthält die Suchanfrage *keine* Anforderung mit + als semantischem Präfix, so wird das Anfrageergebnis (sieht man von den Anforderungen mit - als semantischem Präfix ab, die ja lediglich Elemente „verbieten“) allein durch die Anforderungen ohne semantischem Präfix bestimmt (es wurden ja keine zwingenden Anforderungen angegeben). Enthält eine Suchanfrage jedoch Anforderungen *mit* + als semantischem Präfix, so wird das Anfrageergebnis (sieht man auch hier von den

Elemente ausschließenden Anforderungen mit - als semantischem Präfix ab) quantitativ von diesen Anforderungen bestimmt; die Auswertung der Anforderungen ohne semantischem Präfix ändert höchstens noch die Bewertungen der Elemente, die durch Auswertung der Anforderungen mit + als semantischem Präfix ermittelt wurden.

Enthält eine Suchanfrage bei dem prinzipiellen Algorithmus von Abschnitt 4.10.1 keine Anforderung mit + als semantischem Präfix, jedoch Anforderungen ohne semantischem Präfix, so ergibt sich folgende Situation: Die Menge M_+ ist in jedem Fall leer. Dadurch wird $M_{+0} = \{\}$, selbst wenn $M_0 \neq \{\}$. Solches wiederum bewirkt, daß auch die Ergebnismenge M in jedem Fall leer ist - was im Fall von $M_0 \neq \{\}$ offensichtlich falsch ist. Dies ist nun die Situation, in der der prinzipielle Algorithmus noch nicht korrekt arbeitet.

Der im folgenden dargestellte Algorithmus konkretisiert den prinzipiellen Algorithmus und beseitigt diesen Fehler. Vom Umgang mit den Bewertungen der Elemente wird bei dem Algorithmus abstrahiert.

Algorithmus 4.10.2: Auswertung von Suchanfragen (konkretisiert)

```

begin
  (*  $A_{I+}$ ,  $A_{I0}$ ,  $A_{I-}$ ,  $A_{D+}$ ,  $A_{D0}$  und  $A_{D-}$  können nicht Nil sein *)
  (* ----- Berechnen von  $M_+$  ----- *)
   $M_+ := E_{I+}(A_{I+});$ 
  (*  $M_+ = Nil \Leftrightarrow A_{I+} = \{\}$  *)
  (*  $M_+ = \{\} \Leftrightarrow$  Auswertung ergab kein passendes Element *)
   $M_+ := E_{D+}(A_{D+}, M_+);$ 
  (*  $M_+ = Nil \Leftrightarrow (A_{I+} = \{\} \text{ and } A_{D+} = \{\})$  *)
  (*  $M_+ = \{\} \Leftrightarrow$  Auswertungen ergaben kein passendes Element *)
  (* ----- Berechnen von  $M_0$  ----- *)
   $M_0 := E_{I0}(A_{I0});$ 
  (*  $M_0 = Nil \Leftrightarrow A_{I0} = \{\}$  *)
  (*  $M_0 = \{\} \Leftrightarrow$  Auswertung ergab kein passendes Element *)
   $M_0 := E_{D0}(A_{D0}, M_0);$ 
  (*  $M_0 = Nil \Leftrightarrow (A_{I0} = \{\} \text{ and } A_{D0} = \{\})$  *)
  (*  $M_0 = \{\} \Leftrightarrow$  Auswertungen ergaben kein passendes Element *)
  (* ----- Berechnen von  $M_{+0}$  ----- *)
  if  $M_+ = Nil$  then
     $M_{+0} := M_0$ 
  else
    if  $M_+ \langle \rangle \{\}$  and  $M_0 \langle \rangle Nil$  and  $M_0 \langle \rangle \{\}$  then
       $M_{+0} := FühreZusammen(M_+, M_0)$ 
    else
       $M_{+0} := M_+$ 
    endif
  endif;
  (*  $M_{+0} = Nil \Leftrightarrow (M_+ = Nil \text{ and } M_0 = Nil)$  *)
  (* ----- Berechnen von M (Teil 1) ----- *)

```

```

if  $M_{+0} = Nil$  or  $M_{+0} = \{\}$  then
     $M := M_{+0}$ 
else
    (* ----- Berechnen von  $M_-$  ----- *)
     $M_- := E_{I_-}(A_{I_-});$ 
    (*  $M_- = Nil \Leftrightarrow A_{I_-} = \{\}$  *)
    (*  $M_- = \{\}$   $\Leftrightarrow$  Auswertung ergab kein passendes Element *)
     $M_- := E_{D_-}(A_{D_-}, M_-);$ 
    (*  $M_- = Nil \Leftrightarrow (A_{I_-} = \{\}$  and  $A_{D_-} = \{\})$  *)
    (*  $M_- = \{\}$   $\Leftrightarrow$  Auswertungen ergaben kein passendes Element *)
    (* ----- Berechnen von M (Teil 2) ----- *)
    if  $M_- \langle \rangle Nil$  and  $M_- \langle \rangle \{\}$  then
         $M := Entferne(M_{+0}, M_-)$ 
    else
         $M := M_{+0}$ 
    endif
endif;
(*  $M = Nil \rightarrow$  Meldung an Benutzer: „Ohne ‚ ‚ oder ‚+‘-Suchbegriff
    kann kein Ergebnis ermittelt werden.“ *)
(*  $M = \{\}$   $\rightarrow$  Meldung an Benutzer: „Es wurden keine passenden
    Dokumente gefunden.“ *)
(*  $M = \{\dots\} \rightarrow$  Ergebnisse nach Bewertungen sortieren und Benutzer
    schicken *)
end

```

„*Nil*“ bedeutet soviel wie „ist nicht definiert“. Diese Situation der undefiniertheit einer Menge darf nicht mit der leeren Menge gleichgesetzt werden! Vielmehr ist die Einführung dieses Zustandes *Nil* für eine Menge der Schlüssel dafür, den genannten Fehler des prinzipiellen Algorithmus zu beheben.

Der Berechnung von M_{+0} und M , sowie der Kontrollflußsteuerung mit Hilfe von if-then-else-Statements liegen die Festlegungen der Tabellen 4.10.2 und 4.10.3 zugrunde. Die Funktionen E_{xy} werten die als Parameter übergebenen Ausdrücke aus. Die Funktion *FühreZusammen* realisiert die im prinzipiellen Algorithmus erwähnte Berechnungsvorschrift

$$M_{+0} = (M_+ - (M_+ \cap_n M_0)) \cup (M_+ \cap M_0),$$

die Funktion *Entferne* die ebenfalls vom prinzipiellen Algorithmus bekannte Berechnungsvorschrift

$$M := M_{+0} - M_-.$$

Im übrigen werden auch hier die Abkürzungen von Tabelle 4.10.1 verwendet.

4.11 Bearbeiten von Suchanfragen

In den letzten Abschnitten wurde die Anfragesprache beschrieben, mit der ein Benutzer Anforderungen formulieren kann. Außerdem wurde darauf eingegangen, wie ein Ausdruck, der Teil einer solchen Anforderung ist, mit entsprechenden Ausprägungen von Merkmalen bei Elementen der Domäne verglichen wird. Schließlich wurde auch ein Algorithmus zur Auswertung von Suchanfragen angegeben. In diesem Abschnitt wird nun beschrieben, wie Suchanfragen unter Verwendung all dieser Konzepte ausgewertet werden.

Erhält ein Domänen-Experte einen CGI-Suchstring, so zerlegt er diesen und entnimmt die darin enthaltenen Anforderungen des Benutzers. Zu jeder Anforderung merkt er sich, aus welchem Eingabefeld sie stammt - wodurch er zuordnen kann, in welcher Zeile der Tabelle mit den Auswertungsdefinitionen die Auswertungsdefinition dieser Anforderung steht.

Der Domänen-Experte legt sechs Tabellen an, je drei für Anforderungen, die anhand des Index des Domänen-Experten überprüft werden müssen (`indexSearch`), und für Anforderungen, die anhand der Dokumenten-Datensätze überprüft werden müssen (`documentDataRecordSearch`). Von den drei Tabellen ist je eine für Anforderungen mit dem semantischen Präfix +, je eine für Anforderungen mit dem semantischen Präfix -, je eine für Anforderungen ohne semantisches Präfix (Realisierung der Mengen A_{xy} von Tabelle 4.10.1 auf Seite 4.10.1).

Die Anforderungen werden auf die sechs Tabellen verteilt. Dabei werden die Anforderungen in den Tabellen ohne eventuelles semantisches Präfix abgelegt - d.h. in den Tabellen stehen nach der Terminologie der letzten Abschnitte nur noch die Ausdrücke. Zu jedem Ausdruck in einer Tabelle wird ebenfalls vermerkt, in welcher Zeile der Tabelle mit den Auswertungsdefinitionen die zugehörige Auswertungsdefinition zu finden ist.

Man beachte, daß jede Anforderung eindeutig einer der sechs Tabellen zugeordnet werden kann.

Die jetzt in Form der sechs Tabellen vorliegende Suchanfrage wird durch den Algorithmus von Abschnitt 4.10.2 bearbeitet. Dabei wird für jeden Ausdruck die entsprechende Auswertungsdefinition der Tabelle mit Auswertungsdefinitionen angewandt und die Elemente werden wie beschreiben bewertet.

Liefert der Algorithmus eine nicht leere Menge von Elementen, so werden diese nach der jeweiligen Bewertung in absteigender Reihenfolge sortiert. Diese Liste von Elementen wird in eine zu diesem Zweck zu generierende HTML-Seite geschrieben, welche dann an den Benutzer übermittelt wird. Liefert der Algorithmus jedoch eine leere Liste, so wurden keine zur Suchanfrage passenden Elemente der Domäne gefunden. In diesem Fall wird eine HTML-Seite mit dieser Information generiert und an den Benutzer gesandt. Wurde bei Abarbeitung des Algorithmus für die Ergebnismenge nicht einmal eine leere Menge erzeugt ($M = Nil$), so hat der Benutzer weder eine Anforderung mit

semantischem Präfix +, noch eine Anforderung ohne semantischem Präfix angegeben. Der Benutzer muß in einer zu generierenden HTML-Seite darauf hingewiesen werden, daß keine Elemente der Domäne gefunden werden können, wenn nicht mindestens eine Anforderung mit + als semantischem Präfix oder ohne semantischem Präfix angegeben wird.

4.12 Formatierung der Suchergebnisse

Wurden bei der Auswertung einer Suchanfrage passende Elemente der Domäne gefunden, so werden die Elemente gemäß ihrer Bewertung in absteigender Reihenfolge sortiert und in eine zu generierende HTML-Seite eingefügt. Diese HTML-Seite wird an den Benutzer geschickt.

Es ist nicht Sinn und Zweck eines Suchwerkzeuges, die vollständigen Elemente an den Benutzer zu schicken. Es muß darum festgelegt werden, *welche* Informationen zu den Elementen in eine HTML-Seite geschrieben und darin dem Benutzer geschickt werden sollen.

Betrachtet man das Anwendungsgebiet „wissenschaftliche Veröffentlichungen“, so sollte dem Benutzer zu jeder gefundenen wissenschaftlichen Veröffentlichung Autor, Titel, Erscheinungsjahr und der URL des Dokumentes geliefert werden, den URL am besten als Hyperlink. Es kann sinnvoll sein, noch weitere Angaben wie Größe des Dokumentes in Kilobyte anzugeben. Bei dem Anwendungsgebiet „abrufbare public domain Software“ sind hingegen die Informationen Titel, Versionsnummer, notwendiges Betriebssystem und der URL der Software sinnvoll, ggf. auch eine Erklärung, wozu die Software gedacht ist. Wie diese beiden Beispiele zeigen sind die Informationen, die dem Benutzer zu einem Element der Domäne zu übermitteln sinnvoll sind, von Domäne zu Domäne verschieden. Für das generische Framework zur Erzeugung von Domänen-Experten muß darum eine Möglichkeit vorgesehen werden, die es erlaubt festzulegen, welche Informationen zu einem Element dem Benutzer übermittelt werden sollen.

Das bei meinem generischen Framework hierzu verwendete Konzept wird im folgenden beschrieben.

4.13 Festlegung des Suchergebnis-Formats

Welche Informationen dem Benutzer zu einem Element der Domäne im Rahmen einer Suchergebnismeldung übermittelt werden sollen und wie diese Informationen formatiert sein sollen, wird beim Erzeugen des Domänen-Experten festgelegt. Abbildung 4.13.1 zeigt ein Beispiel für eine solche Definition anhand eines (fiktiven) zu erzeugenden Domänen-Experten für wissenschaftliche Veröffentlichungen. Man beachte, daß eine solche Definition die Ausgaben für *ein* Element der Domäne festlegt, d.h. diese Definition wird sozusagen auf alle Elemente der Suchergebnismenge iterativ angewendet.

Die im Framework vorgesehenen Mittel zur Definition einer Suchergebnis-Formatierung werden im folgenden anhand dieses Beispiels beschrieben.

4.13.1 Bezeichner

In Abbildung 4.13.1: `authors, url, title, date, institutions, numberOfPages, size, docformat, fileformat, language, topic`

Bei den verwendbaren *Bezeichnern* handelt es sich um die von der Definition von Dokumenten-Datensätzen schon bekannten. Mit anderen Worten: Ein Bezeichner, der hier verwendet werden soll, muß Bezeichner eines Teilfeldes des Dokumenten-Datensatzes sein (siehe Abschnitt 3.5.2).

Wirkung: Der Teil des Dokumenten-Datensatzes, für den der Bezeichner steht, wird ausgelesen und in die Suchergebnisseite eingefügt.

4.13.2 String

In Abbildung 4.13.1: `":", ",", "pages", "kByte", "language:"`

Strings sind zwischen Anführungszeichen eingeschlossene Zeichenketten.

Wirkung: Die zwischen den Anführungszeichen stehende Zeichenkette wird in die Suchergebnisseite eingefügt.

4.13.3 Zeilenumbruch

Ein Zeilenumbruch auf der Suchergebnisseite wird durch die Zeichenfolge `\` eingefügt.

4.13.4 Funktionsaufrufe

In Abbildung 4.13.1: `\wwwLink{url}{title}`

Funktionsaufrufe beginnen mit einem Backslash. Prinzipiell können nur im Domänen-Experten-Framework vordefinierte Funktionen verwendet werden. Es ist jedoch möglich weitere Funktionen für das Framework zu definieren.

Die Parameter für einen Funktionsaufruf werden je in ein paar geschwungene Klammern gesetzt. Die Anzahl der Parameter wird seitens der Funktionsdefinition im Domänen-Experten-Framework vorgegeben und muß beim Aufruf eingehalten werden.

Seitens des Frameworks ist festgelegt, daß für die aktuellen Parameter als Typen nur *Bezeichner* und *String* erlaubt sind. Overloading (gleicher Funktionsname, verschiedene Anzahlen von Parametern und/oder Parametertypen) ist durch entsprechende Implementierung der Funktion im Framework realisierbar.

Prinzipiell ist es möglich, für das generische Framework Funktionen mit beliebig vielen Parametern zu definieren; auch Funktionen ohne Parameter sind möglich.

Wirkung: Ein Funktionsaufruf erzeugt anhand der Parameter eine Ausgabe, die in die Ergebnisseite geschrieben wird.

In obigem Beispiel wird der HTML-Quellcode für einen Hyperlink erzeugt, der am Text von `title` festgemacht ist und auf das durch `url` spezifizierte Dokument des World Wide Web verweist.

4.14 Quellen

Die Abschnitte 4.2 bis 4.4 wurden unter Verwendung von [SELFHTML/Münz] und [Schmidt98] erstellt. HTML-Anweisungen dieses Kapitels stammen ebenfalls aus den Quellen [SELFHTML/Münz] und/oder [Schmidt98]. Ansonsten sind verwendete Literaturquellen an der jeweiligen Stelle durch Angabe des Verweises vermerkt.

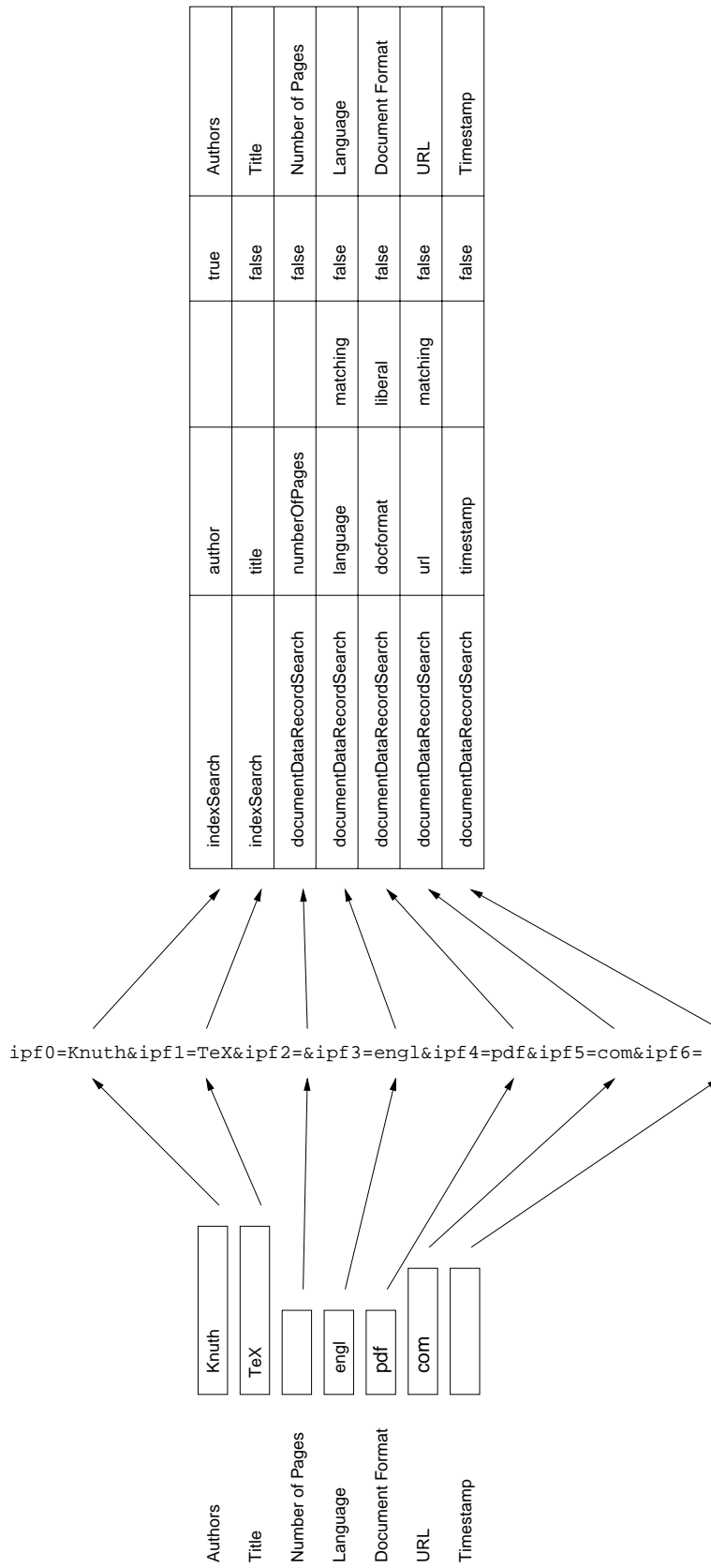


Abbildung 4.5.4: Zuordnung Eingabefeldinhalte zu interner Tabelle mit Auswertungsdefinitionen

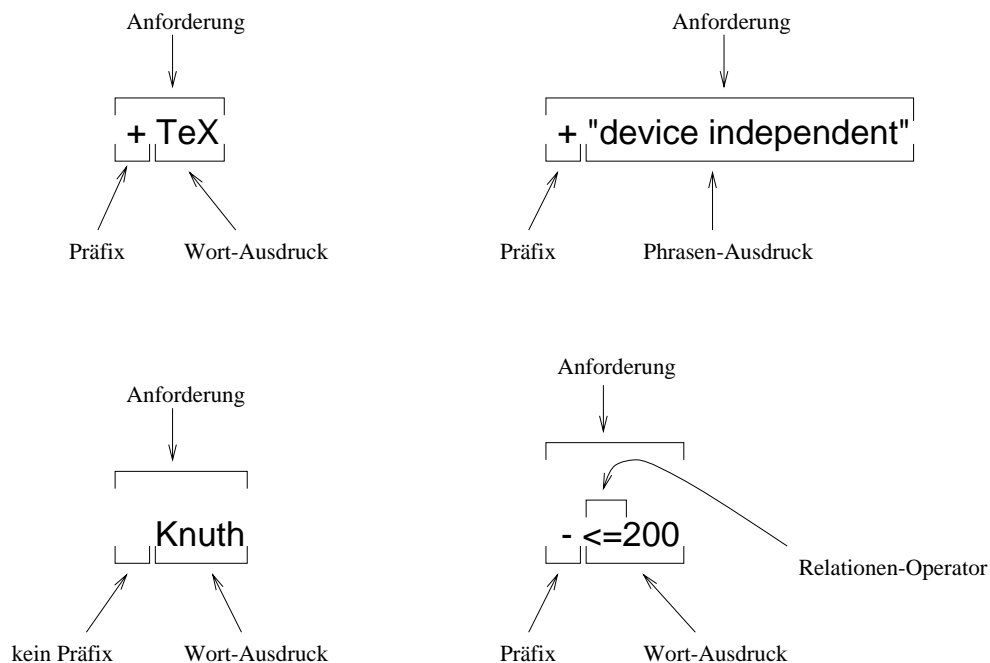


Abbildung 4.7.1: Beispiele für Anforderungen, die bei der Formulierung von Suchanfragen möglich sind

```

authors ":" \\
\wwwLink{url}{title} "," date \\
institutions \\
numberOfPages "pages" size "kByte" docformat fileformat \\
"language:" language \\
topic

```

Abbildung 4.13.1: Beispiel für die Definition eines Suchergebnis-Formats

Datentyp	Vergleichsart	Beschreibung	bei Phrasen-Suche zusätzlich	Relationen-Operator
FTString FTStringCollection	exact	normaler Stringvergleich	-	-
	liberal (default)	Groß-/Kleinschreibung ignoriert	nur eine Leerstelle zwischen Wörtern	
	matching	Patternmatching	nur eine Leerstelle zwischen Wörtern	
FTInt	-	-	(keine Phrasensuche möglich)	<, <=, >, >=, <>, =
FTIntCollection	-	-	(keine Phrasensuche möglich)	<, <=, >, >=, <>, =
FTDate	-	-	(keine Phrasensuche möglich)	<, <=, >, >=, <>, =
FTDateCollection	-	-	(keine Phrasensuche möglich)	<, <=, >, >=, <>, =
FTInterval	total (default)	Länge des Intervalls wird verwendet [◇]	(keine Phrasensuche möglich)	<, <=, >, >=, <>, =
FTIntervalCollection	-	-	(keine Phrasensuche möglich)	<, <=, >, >=, <>, =
FTTimestamp	-	-	(keine Phrasensuche möglich)	<, <=, >, >=, <>, =
FTTimestampCollection	-	-	(keine Phrasensuche möglich)	<, <=, >, >=, <>, =
FTURL FTURLCollection	exact	normaler Stringvergleich	-	-
	liberal (default)	Groß-/Kleinschreibung ignoriert	(keine Phrasensuche möglich)	
	matching	Patternmatching	(keine Phrasensuche möglich)	

◇ Dieser Datentyp speichert Anfangs- und Endwert des von ihm repräsentierten Intervalls. Bei der Vergleichsart `total` verhält er sich jedoch so, als wäre die Länge des Intervalls gespeichert. Wird ein Feld vom Typ `FTInterval` in das Ergebnis auf eine Suchanfrage ausgegeben (siehe Abschnitt 4.13), so erscheint die Angabe des Intervalls, nicht dessen Länge.

Tabelle 4.8.1: Feld-Typen und Vergleichsmöglichkeiten

Menge	Definition
A_{I0}	Menge von Ausdrücken, denen kein semantisches Präfix vorangeht und die im Index ausgewertet werden
A_{I+}	Menge von Ausdrücken, denen + als semantisches Präfix vorangeht und die im Index ausgewertet werden
A_{I-}	Menge von Ausdrücken, denen - als semantisches Präfix vorangeht und die im Index ausgewertet werden
A_{D0}	Menge von Ausdrücken, denen kein semantisches Präfix vorangeht und die an den Dokumenten-Datensätzen ausgewertet werden
A_{D+}	Menge von Ausdrücken, denen + als semantisches Präfix vorangeht und die an den Dokumenten-Datensätzen ausgewertet werden
A_{D-}	Menge von Ausdrücken, denen - als semantisches Präfix vorangeht und die an den Dokumenten-Datensätzen ausgewertet werden
M_{I0}	Menge der Elemente der Domäne, für die mindestens ein Ausdruck von A_{I0} zutrifft
M_{I+}	Menge der Elemente der Domäne, für die alle Ausdrücke von A_{I+} gleichzeitig zutreffen
M_{I-}	Menge der Elemente der Domäne, für die mindestens ein Ausdruck von A_{I-} zutrifft
M_{D0}	Menge der Elemente der Domäne, für die mindestens ein Ausdruck von A_{D0} zutrifft
M_{D+}	Menge der Elemente der Domäne, für die alle Ausdrücke von A_{D+} gleichzeitig zutreffen
M_{D-}	Menge der Elemente der Domäne, für die mindestens ein Ausdruck von A_{D-} zutrifft
M_+	Menge von Elementen der Domäne, die alle Ausdrücke mit semantischem Präfix + in der Anforderung erfüllen
M_0	Menge von Elementen der Domäne, die mindestens einen Ausdruck erfüllen, der in der Anforderung kein semantisches Präfix besitzt
M_{+0}	Menge von Elementen der Domäne, die alle Ausdrücke mit semantischem Präfix + erfüllen. Die Bewertung jedes Elementes dieser Menge wurde um die Bewertung erhöht, die das Element ggf. in der Menge M_0 hat
M_-	Menge von Elementen der Domäne, die mindestens einen Ausdruck erfüllen, der in der Anforderung das semantische Präfix - besitzt
M	Menge von Elementen der Domäne, die zur Suchanfrage passen

Anmerkung: Nach der Terminologie der letzten Abschnitte besteht eine Suchanfrage aus *Anforderungen* und eine Anforderung aus einem Ausdruck, dem ein *semantisches Präfix* vorangehen kann (aber nicht muß).

Tabelle 4.10.1: Mengendefinitionen

M_{+0} :

M_+	M_0	Nil	$\{\}$	$\{\dots\}$
Nil	Nil	$(= M_0)$	$\{\}$	$(= M_0)$
$\{\}$	$\{\}$	$(= M_+)$	$\{\}$	$(= M_+)$
$\{\dots\}$	M_+	M_+	$(M_+ - (M_+ \cap_n M_0)) \cup (M_+ \cap M_0)$	

Tabelle 4.10.2: Berechnung der Menge M_{+0}

M :

M_{+0}	M_-	Nil	$\{\}$	$\{\dots\}$
Nil	Nil	$(= M_{+0})$	Nil	$(= M_{+0})$
$\{\}$	$\{\}$	$(= M_{+0})$	$\{\}$	$(= M_{+0})$
$\{\dots\}$	M_{+0}	M_{+0}	$M_{+0} - M_-$	

Tabelle 4.10.3: Berechnung der Menge M

Kapitel 5

Generieren und Betriebsphase

Dieses Kapitel nennt zunächst weitere Generierungsparameter. Im Anschluß daran wird das Konfigurationsfile erklärt, anhand dessen die Erstellung eines Domänen-Experten erfolgt. Schließlich gehe ich auf die im Rahmen dieser Diplomarbeit für Domänen-Experten erstellte Betriebsphase ein.

Inhaltsangabe

5.1	Verschiedene Generierungsparameter	62
5.2	Details zur Erstellung eines Domänen-Experten - Übersicht	64
5.3	Das Konfigurationsfile	64
5.4	Konsistenzbedingungen	67
5.5	Lexikalische Struktur und Syntax	68
5.6	Weiteres zum Konfigurationsfile	70
5.7	Betriebsphase - Vorbemerkung	70
5.8	Übersicht über die Betriebsphase	70
5.9	Ablauf des Wissenserwerbs	71

5.1 Verschiedene Generierungsparameter

Das generische Framework bietet die Möglichkeit, bei der Erzeugung eines Domänen-Experten - neben der schon in früheren Kapiteln dargestellten Definition einer Suchmaske, von Auswertungsfunktionalität für Suchanfragen, von Dokumenten-Datensätzen und des Ergebnisausgabe-Formates - weitere Parameter anzugeben, die bestimmte Einstellungen vornehmen.

Der Benutzer des Frameworks, der den Domänen-Experten erzeugen will, gibt diese Parameter unter Verwendung folgender EBNF-Syntax an:

```
Bezeichner "=" (String | Zahl)
```

Die möglichen **Bezeichner** sind vom Framework vorgegeben. Der Parameter wird rechts des = angegeben und kann somit syntaktisch entweder ein String oder eine Zahl sein. Ob der Parameter im konkreten Fall jedoch ein String oder eine Zahl sein muß, hängt vom Bezeichner links des = ab.

Die vom Framework vorgegebenen und damit verwendbaren Bezeichner werden in den folgenden Unterabschnitten besprochen.

5.1.1 ConventionalSearchEngineSearchString

In Abschnitt 4.6.2 wurde erklärt, wie die Parameter für Anfragen eines Domänen-Experten an eine konventionelle Suchmaschine ermittelt werden.

Je nach Anwendungsgebiet des Domänen-Experten kann es sinnvoll sein, die so ermittelten dynamischen Parameter um weitere Parameter zu ergänzen, die aufgrund des Anwendungsgebietes des Domänen-Experten bei jeder Suchanfrage an die konventionelle Suchmaschine zu senden sinnvoll sind.

Beispiel: Bei einem Domänen-Experten für wissenschaftliche Veröffentlichungen kann es sinnvoll sein, die Anforderungen

```
+abstract +references
```

bei jeder Anfrage an die konventionelle Suchmaschine mit anzugeben. □

Das Framework bietet zur Festlegung solcher statischer Anfrageparameter den Bezeichner **ConventionalSearchEngineSearchString** an. Der Parameter muß dabei ein String sein. Es wird für die Formulierung des Parameters - abgesehen von den umschließenden " - die Syntax verwendet, die auch zur Formulierung von Suchanfragen an den Domänen-Experten während der Betriebsphase zum Einsatz kommt.

Beispieldefinition für einen zu erstellenden (fiktiven) Domänen-Experten für wissenschaftliche Veröffentlichungen:

```
ConventionalSearchEngineSearchString = "+abstract +references"
```

5.1.2 QueryEvaluatingProgram

Das Framework generiert bei der Erstellung eines Domänen-Experten gleichzeitig die zugehörige themenspezifische Suchmaske. Diese themenspezifische Suchmaske ist eine HTML-Datei, die ein Formular enthält. In diesem Formular muß vermerkt sein, an welches Programm oder auch Skript die Eingaben, die ein Benutzer in diesem Formular macht, gehen sollen. Dies hat zur Konsequenz, daß das Framework zur vollständigen Generierung der themenspezifischen Suchmaske auch diese Information besitzen muß.

Zur Angabe dieser Information stellt das Framework den Bezeichner `QueryEvaluatingProgram` zur Verfügung. Der Parameter muß ein String sein.

Beispieldefinition:

```
QueryEvaluatingProgram  
    = "http://www.informatik.uni-stuttgart.de/vsbin/search.cgi"
```

5.1.3 MaxNumberOfResults

Bei der Erstellung eines Domänen-Experten kann angegeben werden, wieviel Elemente der Domäne auf eine Suchanfrage hin maximal dem Benutzer genannt werden sollen. Dazu dient der Bezeichner `MaxNumberOfResults`. Der Parameter ist eine Zahl.

Beispieldefinition:

```
MaxNumberOfResults = 40
```

5.1.4 DocumentAcceptanceThreshold

Die Filter-Agenten beurteilen Elemente auf anderen Servern hinsichtlich ihrer Relevanz für das Anwendungsgebiet des Domänen-Experten. Mit Hilfe des Bezeichners `DocumentAcceptanceThreshold` kann der Ersteller eines Domänen-Experten angeben, welchen Relevanzgrad ein Element bezüglich des Anwendungsgebietes des Domänen-Experten mindestens haben muß, um als für das Anwendungsgebiet relevant eingestuft zu werden. Der Parameter dieses Bezeichners muß eine Zahl sein.

Beispieldefinition:

```
DocumentAcceptanceThreshold = 0.5
```

5.2 Details zur Erstellung eines Domänen-Experten - Übersicht

In Abschnitt 3.4 wurde bereits das Grundprinzip meines Konzeptes zur Erstellung eines Domänen-Experten dargestellt. Dieser Abschnitt konkretisiert dieses Grundprinzip. Dazu werden insbesondere auch die in den Abschnitten 3.5 bis 5.1 dargestellten Konzepte verwendet.

In Abschnitt 3.4 wurde dargelegt, daß der Domänen-Experte durch Konfigurieren des Frameworks mit Hilfe eines Konfigurationsfiles erstellt wird. Dieses Konfigurationsfile ist dadurch ein Kernstück der Erzeugung eines Domänen-Experten (das andere Kernstück ist der Filter-Agent, da dieser einen domänenspezifischen Teil zwingend benötigt (siehe Abschnitt 2.5.1)).

Für das Erstellen eines Konfigurationsfiles wurde eine formale Sprache definiert. Abschnitt 5.3 geht auf Konfigurationsfiles anhand eines Beispiels ein und Abschnitt 5.4 nennt die Konsistenzbedingungen innerhalb eines Konfigurationsfiles. Anschließend gibt Abschnitt 5.5 lexikalische Struktur und Syntax der zurundeliegenden formalen Sprache an und Abschnitt 5.6 gibt weitere Erklärungen zum Konzept für das Konfigurationsfile.

5.3 Das Konfigurationsfile

Dieser Abschnitt erklärt Konfigurationsfiles anhand des Beispiels eines zu erstellenden (fiktiven) Domänen-Experten für wissenschaftliche Veröffentlichungen.

Das Beispiel eines Konfigurationsfiles für die Erstellung eines solchen Domänen-Experten ist wie folgt:

```
DomainExpertConfiguration "Domain Expert for Scientific Publications";  
(% Steffen Arnold, 1999 %)
```

```
BeginMiscSettings  
  ConventionalSearchEngineSearchString = "+abstract +references";  
  QueryEvaluatingProgram =  
    "http://www.informatik.uni-stuttgart.de/vsbin/search.cgi";  
  MaxNumberOfResults = 40;  
  DocumentAcceptanceThreshold = 0.5  
EndMiscSettings
```

```
BeginIndexWordAttributes  
  author, title, institution, published, url, refAuth,  
  refTitle, refPublished, refURL, abstract, introduction,  
  relatedWork, conclusion
```

EndIndexWordAttributes

BeginDocumentDataRecord

```
authors : FTStringCollection;
title : FTString;
institutions : FTStringCollection;
referencePublished : FTDateCollection;
referencePages : FTIntervalCollection;
keywords : FTStringCollection;
numberOfPages : FTInt;
date : FTDate;
timestamp : FTTimestamp;
url : FTURL;
topic : FTString;
language : FTString;
size : FTInt;
docformat : FTString;
fileformat : FTString;
relatedWork : FTStringCollection
```

EndDocumentDataRecord

BeginGUIDefinition

```
"Authors", 40 : indexSearch{author};
"Title", 40 : indexSearch{title};
"Keywords", 30 : documentDataRecordSearch{keywords}{liberal}
                , conventionalSearchEngineCriterion;
"Institutes", 40 : indexSearch{institution};
"Number of Pages", 20 : documentDataRecordSearch{numberOfPages};
"Date", 20 : documentDataRecordSearch{date};
"Timestamp", 20 : documentDataRecordSearch{timestamp};
"URL", 40 : documentDataRecordSearch{url}{matching};
"Topic", 40 : documentDataRecordSearch{topic}{matching};
"Language", 20 : documentDataRecordSearch{language}{liberal};
"Size", 20 : documentDataRecordSearch{size};
"Document Format", 20 :
    documentDataRecordSearch{docformat}{liberal};
"File Format", 20 : documentDataRecordSearch{fileformat}{liberal};
"Related Work Keywords", 40 :
    documentDataRecordSearch{relatedWork}{liberal};
"Authors (References)", 40 : indexSearch{refAuth};
"Titles (References)", 40 : indexSearch{refTitle};
"Publishing Entity (References)", 40 : indexSearch{refPublished};
"Publishing Date (References)", 30 :
    documentDataRecordSearch{referencePublished};
```

```

"Number of Pages (References)", 20 :
    documentDataRecordSearch{referencePages}{total};
"URL (References)", 40 : indexSearch{refURL};
"Words in Entire Document", 40 : indexSearch{};
"Abstract", 40 : indexSearch{abstract};
"Introduction", 40 : indexSearch{introduction};
"Related Work", 40 : indexSearch{relatedWork};
"Conclusion", 40 : indexSearch{conclusion}
EndGUIDefinition

BeginResultDefinition
authors ":" \\
  \wwwLink{url}{title} "," date \\
  institutions \\
  numberOfPages "pages" size "kByte" docformat fileformat \\
  "language:" language \\
  topic
EndResultDefinition

EndDomainExpertConfiguration.

```

Erläuterungen:

1. Ein Konfigurationsfile beginnt mit dem Schlüsselwort `DomainExpertConfiguration`. Der darauf folgende String spezifiziert den Namen des Domänen-Experten. Er erscheint als Überschrift auf der HTML-Suchmaske und wird auch als deren Titel verwendet. Außerdem wird er zur Generierung des `description-meta-tags` der HTML-Suchmaske eingesetzt.
2. Kommentare beginnen mit der Zeichenfolge `(%` und enden mit der Zeichenfolge `%)`. Sie können an beliebigen Stellen bis vor dem die Konfigurationsbeschreibung abschließenden Punkt eingefügt werden.
3. Der erste Block ist der `MiscSettings`-Block. In ihm können bestimmte Konfigurationsparameter angegeben werden (siehe Abschnitt 5.1). Man beachte, daß die Definitionen durch einen Strichpunkt *getrennt, nicht abgeschlossen* werden.
4. Im zweiten Block, dem `IndexWordAttributes`-Block werden die im Index verwendeten Attribute deklariert. Dies dient der Vollständigkeit der Beschreibung eines Domänen-Experten und trägt dazu bei, daß ein Konfigurationsfile auch nach Erstellen eines Domänen-Experten als umfangreichere Beschreibung dessen herangezogen werden kann.
5. Der dritte Block ist der `DocumentDataRecord`-Block. In ihm werden die in Abschnitt 3.5.2 dargestellten Definitionen vorgenommen. Die Definitionen werden dabei durch Strichpunkte *getrennt*. Es ist zu beachten, daß das Framework die

Definition eines Feldes `url` mit Typ `FTURL` und eines Feldes `timestamp` mit Typ `FTTimestamp` zwingend vorschreibt. Diese Felder sind - wie Abschnitt 5.9 zeigen wird - unerlässlich für die Betriebsphase eines Domänen-Experten.

6. Es folgt der `GUIDefinition`-Block. Die darin enthaltenen Anweisungen wurden in Abschnitt 4.5 eingeführt. Man beachte auch hier, daß die Definitionen durch Strichpunkte *getrennt, nicht abgeschlossen* werden.
7. Der letzte Block ist der `ResultDefinition`-Block. Die darin enthaltene Funktionalität wurde bereits in Abschnitt 4.13 dargestellt.
8. Nach dem `ResultDefinition`-Block folgt das Schlüsselwort `EndDomainExpertConfiguration`.
9. Die Konfigurationsbeschreibung wird durch einen Punkt abgeschlossen. Nach diesem Punkt darf das Konfigurationsfile keine weiteren Zeichen enthalten.
10. Die Reihenfolge der Blöcke (`MiscSettings`-Block, `IndexWordAttributes`-Block, `DocumentDataRecord`-Block, ...) kann nicht geändert werden. Durch diese Festlegung, die keine Einschränkung des Konzeptes darstellt, ist es möglich, mit einem One-Parse-Compiler das Konfigurationsfile zu analysieren, die themenspezifische Suchmaske zu erzeugen und das Framework zu konfigurieren (d.h. den Domänen-Experten zu erstellen).

5.4 Konsistenzbedingungen

In diesem Abschnitt werden die in einem Konfigurationsfile geltenden Konsistenzbedingungen genannt. Das Framework stellt deren Einhaltung durch entsprechende Prüfungen sicher.

1. Im `MiscSettings`-Block dürfen nur solche Bezeichner verwendet werden, die vom Framework vorgegeben sind. Der jeweilige Bezeichner legt auch fest, ob bei einer Definition ein String oder eine Zahl angegeben werden muß.
2. Im `IndexWordAttributes`-Block sollte es keine zwei Attribute gleichen Namens geben. Ist dies doch der Fall, werden die Attribute als identisch betrachtet. Der Ersteller des Domänen-Experten erhält eine Warnung. Dies ist eine schwächere Konsistenzbedingung, auf deren Einhaltung vom Framework aus nicht bestanden wird.
3. Im `DocumentDataRecord`-Block darf es keine zwei Felder mit dem gleichen Bezeichner geben.
4. Im `DocumentDataRecord`-Block dürfen nur vom Framework vorgegebene Typen verwendet werden.

5. Im GUIDefinition-Block können nur Attribute des Index referenziert werden, die im IndexWordAttributes-Block deklariert wurden.
6. Im GUIDefinition-Block können nur Felder des Dokumenten-Datensatzes referenziert werden, die im DocumentDataRecord-Block definiert wurden.
7. Im GUIDefinition-Block dürfen nur bei Auswertungsdefinitionen für Dokumenten-Datensätze Vergleichsarten angegeben werden, und dann auch nur solche, die der Typ des entsprechenden Feldes des Dokumenten-Datensatzes zuläßt.
8. Es ist möglich ein Index-Attribut und ein Feld des Dokumenten-Datensatzes mit gleichem Bezeichner zu definieren.
9. Im ResultDefinition-Block dürfen als Bezeichner nur solche von Feldern des Dokumenten-Datensatzes verwendet werden.
10. Im ResultDefinition-Block dürfen nur solche Funktionen aufgerufen werden, die vom Framework vorgegeben sind. Dabei muß die jeweilige Funktion mit Anzahl und Typen der aktuellen Parameter des Funktionsaufrufes zurechtkommen können.

5.5 Lexikalische Struktur und Syntax

Im folgenden werden lexikalische Struktur und Syntax der dem Konfigurationsfile zugrundeliegenden formalen Sprache angegeben. Dies geschieht durch reguläre Ausdrücke (überwiegend) bzw. durch Produktionen in Extended-Backus-Naur-Form (EBNF).

5.5.1 Lexikalische Struktur

```

Char → ASCII-Zeichen 0 bis 127 je einschließlich
StringChar → ASCII-Zeichen 32 bis 125 je einschließlich,
             mit Ausnahme von ', ", \
Letter → a|b|...|z|A|B|...|Z
OktDigit → 0|1|...|7
DecDigit → OktDigit|8|9
HexDigit → DecDigit|a|A|b|B|...|f|F
Identifier → Letter(Letter|DecDigit)*
FunctionIdentifier → \ (Letter|DecDigit)(Letter|DecDigit)*
Delimiter → ;|,|:|{|}|\\|=|.
UnicodeChar → \ u HexDigit HexDigit HexDigit HexDigit
String → "(StringChar|UnicodeChar|\ ("|'|\\))*"
Number → DecDigit DecDigit* (ε|. DecDigit DecDigit*)
Comments → (% Char* %)
SyntacticSpace → Leerzeichen, horizontaler Tabulator, Zeilen-
                vorschub, Wagenrücklauf, Seitenvorschub

```

5.5.2 Syntax

```

ConfigFile = Head Body ".".
Head = "DomainExpertConfiguration" String ";".
Body = MiscSettings IndexWordAttributes DocumentDataRecord
      GUIDefinition ResultDefinition "EndDomainExpertConfiguration".

MiscSettings = "BeginMiscSettings" Settings "EndMiscSettings".
Settings = OneSetting {";" OneSetting}.
OneSetting = Identifier "=" (String|Number).

IndexWordAttributes = "BeginIndexWordAttributes"
                    [AttributesOfWords] "EndIndexWordAttributes".
AttributesOfWords = Identifier {"," Identifier}.

DocumentDataRecord = "BeginDocumentDataRecord" DefinitionsForRecord
                    "EndDocumentDataRecord".
DefinitionsForRecord = OneRecordDefinitionPart
                    {";" OneRecordDefinitionPart}.
OneRecordDefinitionPart = Identifier ":" Identifier.

GUIDefinition = "BeginGUIDefinition"
                DefinitionsForGUI "EndGUIDefinition".
DefinitionsForGUI = OneGUIDefinitionPart {";" OneGUIDefinitionPart}.
OneGUIDefinitionPart = SearchGUISettings ":"
                    ("indexSearch" IndexSearchSettings|
                    "documentDataRecordSearch"
                    DocumentDataRecordSearchSettings).
SearchGUISettings = String "," Number.
IndexSearchSettings = "{" [Identifier] "}"
                    ["," "conventionalSearchEngineCriterion"].
DocumentDataRecordSearchSettings = "{" Identifier "}"
                    [{" Identifier "}]
                    ["," "conventionalSearchEngineCriterion"].

ResultDefinition = "BeginResultDefinition" OneResultDefinition
                  {OneResultDefinition} "EndResultDefinition".
OneResultDefinition = (Identifier|String|"\"|FunctionCall).
FunctionCall = FunctionIdentifier {"{"(String|Identifier)}"}.

```

5.6 Weiteres zum Konfigurationsfile

Im folgenden werden noch Aussagen zu dem dem Konfigurationsfile zugrundeliegenden Konzept gemacht.

Die dem Konfigurationsfile zugrundeliegende formale Sprache ist so definiert, daß der Code von Konfigurationsfiles möglichst „sprechend“ wird. Dadurch soll das Verständnis für die Definitionen in einem Konfigurationsfile verbessert werden.

Der `IndexWordAttributes`-Block wäre prinzipiell für ein Konfigurationsfile nicht notwendig. Ich schreibe ihn jedoch vor, um das Konfigurationsfile nicht nur Mittel zum Zweck der Erstellung eines Domänen-Experten sein zu lassen, sondern eine *Beschreibung* des zu erstellenden Domänen-Experten, die einen gewissen Überblick über diesen gibt. Dazu trägt auch bei, daß die Felder mit den Bezeichnern `url` und `timestamp` vom Typ `FTURL` bzw. `FTTimestamp` für den Dokumenten-Datensatz explizit definiert werden müssen und nicht automatisch definiert sind.

5.7 Betriebsphase - Vorbemerkung

In den folgenden Abschnitten wird die Betriebsphase eines Domänen-Experten dargestellt. Dies beinhaltet insbesondere beim Verlauf der Bearbeitung einer Suchanfrage die Erweiterung und Aktualisierung des Datenbestandes.

Ein mit dem im Rahmen dieser Diplomarbeit entworfenen Framework erstellter Domänen-Experte muß nicht die komplette Betriebsfunktionalität eines Domänen-Experten mit lernender Arbeitsweise, autonomer Arbeitsweise und proaktiver Arbeitsweise besitzen (siehe Abschnitt 2.1.2), sondern nur zu einer einfachen Anfragebearbeitung mit Element-Suche und -Analyse, sowie Aktualisierung des Datenbestandes in der Lage sein.

5.8 Übersicht über die Betriebsphase

Konnte sich das Framework erfolgreich konfigurieren (*Konfigurationsphase*), so wechselt es in die *Betriebsphase* und ist damit ein Domänen-Experte (siehe Abschnitt 3.4).

Erster Schritt der Betriebsphase ist, die Empfangsmöglichkeit für Suchanfragen einzurichten. Anschließend wartet der Domänen-Experte auf Suchanfragen, die er bei Eintreffen bearbeitet, um danach auf die nächste Suchanfrage zu warten.

Das Einrichten der Empfangsmöglichkeit für Suchanfragen besteht darin, daß der Domänen-Experte den Zugriff mittels Remote Method Invokation (RMI) ermöglicht.

Das Bearbeiten einer Suchanfrage beinhaltet folgendes: Der Domänen-Experte holt Hinweise auf relevante Elemente von der konventionellen Suchmaschine ein. Anschließend geht er Hinweisen auf ihm unbekanntere Elemente/neuere Versionen bekannter

Elemente nach, indem er sie durch Filter-Agenten untersuchen läßt. Anhand der Ergebnisse der Untersuchung erweitert oder korrigiert der Domänen-Experte dann seinen Datenbestand. Aufgrund dieses Datenbestandes wird schließlich die Suchanfrage bearbeitet, worauf die so erhaltenen Suchergebnisse an den Anfragersteller übermittelt werden.

Eine Suchanfrage wird in Form eines CGI-Suchstrings mittels RMI dem Domänen-Experten übergeben. Die Suchergebnisse werden in eine HTML-Seite eingefügt, die dann als Rückgabeparameter der RMI zurückgegeben wird.

Das im vorletzten Absatz geschilderte Vorgehen mit Einholen von Hinweisen auf Elemente des Anwendungsgebietes, der Untersuchung von unbekanntem Elementen/neueren Versionen bekannter Elemente und erweitern/ändern des Datenbestandes ist noch recht abstrakt. Die folgenden Abschnitte gehen näher darauf ein.

5.9 Ablauf des Wissenserwerbs

In diesem Abschnitt wird darauf eingegangen, wie ein Domänen-Experte im Zuge einer Suchanfrage seinen Datenbestand pflegt. Dabei gilt das im Abschnitt 5.8 dargestellte allgemeine Vorgehen eines Domänen-Experten.

Der Ablauf des Wissenserwerbs wird in zwei Schritten erklärt. Zunächst wird der allgemeine Ablauf dargestellt (Abschnitt 5.9.1), anschließend wird auf die Zuordnung der Funktionalitäten zu den funktionellen Komponenten eines Domänen-Experten eingegangen (Abschnitt 5.9.2). Dabei wird auch jeweils kurz der Kontext der Bearbeitung einer Suchanfrage geschildert.

5.9.1 Allgemeiner Ablauf

Erhält ein Domänen-Experte einen CGI-Suchstring, so analysiert er diesen und filtert die Anforderungen der im CGI-Suchstring codierten Suchanfrage heraus. Die so erhaltenen Anforderungen zerlegt er jeweils wiederum in semantisches Präfix und Ausdruck, wobei er jeden Ausdruck in eine der sechs Anfragetabellen einordnet (siehe Abschnitt 4.11). Außerdem formuliert er mit Hilfe der entsprechenden Anforderungen die Anfrage an die konventionelle Suchmaschine (siehe Abschnitte 4.6.2 und 5.1.1).

Im nächsten Schritt schickt er die Suchanfrage für die konventionelle Suchmaschine an eben diese, welche dann eine Liste von Hinweisen auf für das Sachgebiet des Domänen-Experten möglicherweise relevante Elemente schickt. Ein Hinweis auf ein Element enthält bestimmte Angaben zu diesem Element, darunter auch dessen URL und einen Timestamp. Für jeden Hinweis erstellt der Domänen-Experte ein Tupel mit eben diesen zwei Angaben. Die so ermittelten Tupel werden in der Menge T abgelegt.

Die URL eines Elementes im World Wide Web spezifiziert das Element bis auf dessen Version eindeutig. Verschiedene Versionen können durch verschiedene Timestamps unterschieden werden. Man kann daher davon ausgehen, daß ein Tupel

(*URL, Timestamp*)

ein Element des World Wide Webs eindeutig spezifiziert. Damit gilt aber auch, daß die Menge T genau die Elemente, spezifiziert, die von der Hinweisliste der konventionellen Suchmaschine genannt wurden.

Der Domänen-Experte sucht aus der Menge T diejenigen Tupel heraus, deren URL mit keiner URL eines Elementes in seinem Datenbestand übereinstimmt, und merkt sie sich in der Menge T_u . Außerdem werden von ihm die Tupel aus T in T_u eingefügt, deren URL er zwar einem Element seines Datenbestandes zuordnen kann, dessen Timestamp jedoch einen späteren Zeitpunkt bezeichnet, als der Timestamp des Elementes im Datenbestand. Das bedeutet, es werden vom Domänen-Experten die Tupel aus T in T_u abgelegt, die - ihm unbekannte Elemente spezifizieren, oder - neuere Versionen ihm bekannter Elemente spezifizieren. Man beachte, daß der Domänen-Experte beim Ausschuchen der Tupel für T_u nicht nur den Datenbestand durchsucht, in dem er Informationen zu Elementen hält, die als zum Sachgebiet des Domänen-Experten zugehörig eingestuft worden sind. Der Domänen-Experte berücksichtigt auch seinen Datenbestand zu Elementen, die als nicht zu seinem Sachgebiet gehörig eingestuft worden sind.

T_u enthält die Hinweise auf Elemente, die vom Domänen-Experten auf Zugehörigkeit zum Sachgebiet des Domänen-Experten untersucht werden müssen. Der Domänen-Experte benutzt mobile Filter-Agenten um diese Untersuchung für jedes $t_u \in T_u$ durchzuführen.

Will ein Filter-Agent ein durch t_u spezifiziertes Element untersuchen, so ist es möglich, daß er unter dem URL von t_u ein Element mit einem einen späteren Zeitpunkt spezifizierenden Timestamp vorfindet. Der Grund hierfür liegt darin, daß der Datenbestand der konventionellen Suchmaschine nicht notwendigerweise Elemente in ihrer aktuellen Version enthält (Suchwerkzeuge, die sich heute weltweit im Einsatz befinden, basieren auf einem *zentralisierten Ansatz*. Vertreter dieses Ansatzes sind *Suchmaschine* und *Katalog*. Suchmaschinen pflegen ihren Datenbestand periodisch, Kataloge werden von Hand gepflegt (siehe Abschnitt 1.3.3)).

Das vom Filter-Agenten für t_u vorgefundene Element sei durch das Tupel t_u^* , das URL und Timestamp des vorgefundenes Elementes enthält, spezifiziert. (Man beachte: Findet ein Filter-Agent genau das von t_u spezifizierte Element vor, so ist $t_u^* = t_u$. Unabhängig vom Timestamp des vorgefundenes Elementes sind jedoch in jedem Fall die URLs von t_u^* und t_u identisch.)

Jedes vorgefundene Element t_u^* wird hinsichtlich der Zugehörigkeit zum Sachgebiet des Domänen-Experten bewertet. Je nach Ergebnis dieser Bewertung geht der Domänen-Experte unterschiedlich vor:

- Das durch t_u^* spezifizierte Element wird als für das Sachgebiet des Domänen-Experten *nicht relevant* eingestuft:
 1. Löschen aller Informationen im Datenbestand des Domänen-Experten, die dort ggf. zu einem Element mit der URL von t_u^* vorhanden sind. (Mit

- Datenbestand ist hier Index, Dokumenten-Datensätze und Informationen zu Elementen, die nicht zum Sachgebiet des Domänen-Experten gehören, gemeint.)
2. In den Teil des Datenbestandes des Domänen-Experten, in dem Informationen zu Elementen, die nicht zum Sachgebiet des Domänen-Experten gehören, gespeichert werden, die Informationen aus t_u^* ablegen.
 3. In T_u das Tupel t_u löschen, das dieselbe URL enthält wie t_u^* .
- Das durch t_u^* spezifizierte Element wird als für das Sachgebiet des Domänen-Experten *relevant* eingestuft:
 1. Löschen aller Informationen im Datenbestand des Domänen-Experten, die dort ggf. zu einem Element mit der URL von t_u^* vorhanden sind. (Mit Datenbestand ist hier Index, Dokumenten-Datensätze und Informationen zu Elementen, die nicht zum Sachgebiet des Domänen-Experten gehören, gemeint.)
 2. Das durch t_u^* spezifizierte Element und Informationen zu diesem Element in den Datenbestand des Domänen-Experten (Index und Dokumenten-Datensätze) aufnehmen.
 3. In T_u das Tupel t_u löschen, das dieselbe URL enthält wie t_u^* .

Ist die Menge T_u schließlich leer, so ist jedes von einem Tupel dieser Menge ursprünglich spezifizierte Element (oder eine neuere Version des Elementes) untersucht und der Datenbestand des Domänen-Experten aufgrund der Untersuchungsergebnisse geändert und/oder erweitert worden. Der Domänen-Experte kann nun anhand dieses modifizierten Datenbestandes die Suchanfrage des Benutzers an ihn auswerten.

Man beachte, daß nach Abschluß der Untersuchung der Elemente t_u^* und der Einarbeitung der Untersuchungsergebnisse in den Datenbestand des Domänen-Experten der Datenbestand des Domänen-Experten hinsichtlich der untersuchten Elemente aktueller sein kann als der der konventionellen Suchmaschine.

5.9.2 Aufteilung der Funktionalität

Am in Abschnitt 5.9.1 geschilderten allgemeinen Ablauf des Wissenserwerbs sind die verschiedenen funktionellen Komponenten des Domänen-Experten beteiligt. Dieser Abschnitt ordnet die Funktionalitäten des geschilderten allgemeinen Ablaufes den Komponenten des Domänen-Experten zu und konkretisiert den Ablauf dabei etwas weiter.

Der CGI-Suchstring wird vom Domänen-Experten empfangen und an den Anfragen-Prozessor weitergegeben. Der Anfragen-Prozessor zerlegt den CGI-Suchstring, baut die sechs Anfragetabellen auf und gibt die Anforderungen für die konventionelle Suchmaschine an die Hinweis-Sammlung, die diese Anforderungen sammelt. Im Anschluß

daran formuliert die Hinweis-Sammlung die Anfrage für die konventionelle Suchmaschine und übermittelt ihr diese. Anhand der von der konventionellen Suchmaschine darauf geschickten Hinweise erstellt die Hinweis-Sammlung die Menge T . Diese Menge wird an den Anfragen-Prozessor geschickt, der von dieser Menge T anhand der Dokument-Wissensbasis die Menge T_u ableitet. Der Anfragen-Prozessor schickt die Menge \bar{T}_u , die eine Kopie der Menge T_u ist, an den Agenten-Koordinator, der Filter-Agenten mit der Untersuchung der durch \bar{T}_u spezifizierten Elemente beauftragt. Die Menge T_u selbst bleibt beim Anfragen-Prozessor.

Index, Dokumenten-Datensätze und Informationen zu Elementen, die nicht zum Sachgebiet des Domänen-Experten gehören, befinden sich in der Dokument-Wissensbasis. Operationen auf diesen Datenstrukturen werden von Filter-Agenten oder der Dokumenten-Analyse daher in der Dokument-Wissensbasis vorgenommen.

Nach Abschluß der Untersuchungen zu einem Element wird das Tupel, das die URL des Elementes enthält, vom Filter-Agent oder der Dokumenten-Analyse aus der Menge T_u des Anfragen-Prozessors gelöscht.

Ist die Menge T_u leer, so wertet der Anfragen-Prozessor die Suchanfrage des Benutzers des Domänen-Experten aus und erstellt die HTML-Ergebnisseite.

Anmerkung: Die Realisierung der Funktionalität von Agenten-Koordinator, Filter-Agent und Dokumenten-Analyse ist nicht Teil dieser Arbeit. Aus diesem Grund wird die Funktionalität dieser Komponenten des Domänen-Experten nicht weiter konkretisiert.

Kapitel 6

Hinweise und Bewertung

In diesem sechsten Teil der Ausarbeitung gehe ich auf die Sachgebietsunabhängigkeit meines Konzeptes ein und gebe Hinweise zur Realisierung ausgesuchter Teile des Konzeptes. Den Abschluß bilden eine Bewertung und ein Ausblick.

Inhaltsangabe

6.1	Anwendungsunabhängigkeit des Ansatzes	76
6.2	Hinweise zur Realisierung	76
6.3	Bewertung und Ausblick	80
6.4	Quellen	83

6.1 Anwendungsunabhängigkeit des Ansatzes

Das zu erstellende generische Framework soll die Erzeugung von Domänen-Experten für die unterschiedlichsten Anwendungsgebiete (Domänen) erlauben. Um dies zu gewährleisten ist es nötig, das Framework mit den ihm zugrundeliegenden Konzepten anwendungsunabhängig zu gestalten.

Im Rahmen dieser Diplomarbeit bin ich dazu wie im folgenden dargestellt vorgegangen.

Für die Domäne „wissenschaftliche Veröffentlichungen“ (engl. „Scientific Publications“) habe ich Merkmale von Dokumenten dieses Anwendungsgebietes zusammengestellt. Einige Merkmale wissenschaftlicher Veröffentlichungen sind z.B. die Angabe eines oder mehrerer Autoren, ein Titel, Literaturliste und Umfang in Seiten. Die komplette Auflistung der Merkmale findet sich im Anhang in Abschnitt A.1.1 .

Aufgrund der zusammengestellten Merkmale der Domäne „wissenschaftliche Veröffentlichungen“ habe ich in Pseudocode die Spezifikation eines Domänen-Experten erstellt. Dies umfaßte neben der Festlegung der internen Datenstruktur und der Anfrage-Suchmaske auch die Definition der Ergebnispräsentation (siehe Abschnitt 5.3).

Die Anwendungsunabhängigkeit dieses Ansatzes wurde von mir dadurch überprüft, daß ich seine Anwendbarkeit auf zwei weitere, thematisch völlig verschiedene Domänen überprüfte. Bei diesen Domänen handelte es sich um „abrufbare public domain Software“ (engl. „Downloadable Public Domain Software“) und „Hotelinformationen“ (engl. „Informations about Hotels“). Auch für diese Anwendungsgebiete habe ich Merkmale ermittelt und einen Domänen-Experten in eben demselben Pseudocode spezifiziert (siehe Anhang Abschnitte A.2 und A.3). Unverträglichkeiten des Ansatzes mit den Gegebenheiten der beiden Domänen wurden nicht bemerkt.

Für den Rahmen dieser Diplomarbeit wurde die Anwendungsunabhängigkeit des Ansatzes mit dieser erfolgreichen Überprüfung als gegeben betrachtet.

6.2 Hinweise zur Realisierung

In diesem Abschnitt werden noch Hinweise zur Realisierung bestimmter Teile des in den letzten Kapiteln dargestellten Konzeptes gegeben. Insbesondere wird auf die Realisierung des Index eingegangen.

6.2.1 Index

Auf den Index wurde in Abschnitt 3.5.1 eingegangen. Für die Realisierung des dort dargestellten, habe ich die Struktur von Abbildung 6.2.1 gewählt. Sie realisiert eine platzeffiziente Speicherung der in Abschnitt 3.5.1 erwähnten Tupel

$$(\text{Wort}_i, \text{Merkmal}_r, \text{Dokument}_j, \text{Position}_{j_i})$$

Betrachtet man die Suchvorgänge im Index eines Domänen-Experten, so wird bei der Auswertung eines Wort-Ausdruckes ein Wort_{*i*} und ein Merkmal_{*r*} als Parameter verwendet um eine Menge von Dokumenten zu ermitteln. Bei der gewählten Datenstruktur liegt nach Ermitteln dieser Parameter in den Tabellen (1) und (2) die Menge der Dokumente fest. Bei der Auswertung von Phrasenausdrücken erfolgt die Suche nach dem ersten Wort wie soeben geschildert, nur daß die Wortpositionen ebenfalls ausgelesen werden müssen. Für die restlichen Worte der Phrase muß noch überprüft werden, ob in den schon ermittelten Dokumenten das nächste Wort der Phrase bei gleichem Merkmal an der nächsten Position zu finden ist.

Beispiel: Die Phrase sei durch „java virtual engine“ gegeben, das Merkmal sei „related work“. Im ersten Schritt wird mit dem Wort „java“ und dem Merkmal gesucht. Angenommen, es wird ein Dokument *x* und eine Position *r* ermittelt, so wird im nächsten Schritt überprüft, ob das Wort „virtual“ im Dokument *x* an der Stelle *r* + 1 mit dem Merkmal „related work“ vorhanden ist. Wenn ja, wird noch überprüft, ob das Wort „engine“ im Dokument *x* an Position *r* + 2 mit dem Merkmal „related work“ existiert. □

Diese Suchvorgänge sind mit *linearer Komplexität* in der für den Index gewählten Datenstruktur durchführbar. Bei Sortierung der Daten und Verwendung von binärer Suche in den Tabellen (1), (2) und (4) kann hier logarithmische Komplexität erreicht werden ([Appelrath91, Seiten 289-291]). (Daß auf eine Ordnung in Tabelle (3) verzichtet wird, liegt an den darin enthaltenen Einträgen für die Dokumente. Sie bestehen nämlich aus den jeweiligen Dokumenten-Datensätzen, genauer gesagt aus Verweisen auf diese.)

Der Zugriff auf die einzelnen Tabellen mittels Hashing könnte eine konstante Komplexität erreichen ([Appelrath91, Abschnitt 6.4.2]). Solches ist jedoch zumindest in der Anfangsphase eines Domänen-Experten bei hauptsächlich *lernender Arbeitsweise* des Domänen-Experten nicht praktikabel, da hier der Datenbestand ständig wächst und die Tabellen damit ständig größer werden.

6.2.2 Dokumenten-Datensatz

Da die Realisierung des generischen Frameworks in der objektorientierten Programmiersprache JAVA erfolgen soll, bietet es sich für den Dokumenten-Datensatz (siehe Abschnitt 3.5.2) an, die Feldtypen als Klassen zu realisieren. Eine Instanz einer solchen Klasse realisiert dann ein Feld eines Dokumenten-Datensatzes (inklusive der entsprechenden Vergleichsarten). Abbildung 6.2.2 veranschaulicht diesen Sachverhalt.

Erhalten die Klassen, die Feldtypen realisieren, eine gemeinsame abstrakte Oberklasse, so können die Instanzen der Klassen im Dokumenten-Datensatz einheitlich durch eine Methode (die jede Klasse anders implementiert) angesprochen werden (Polymorphismus).

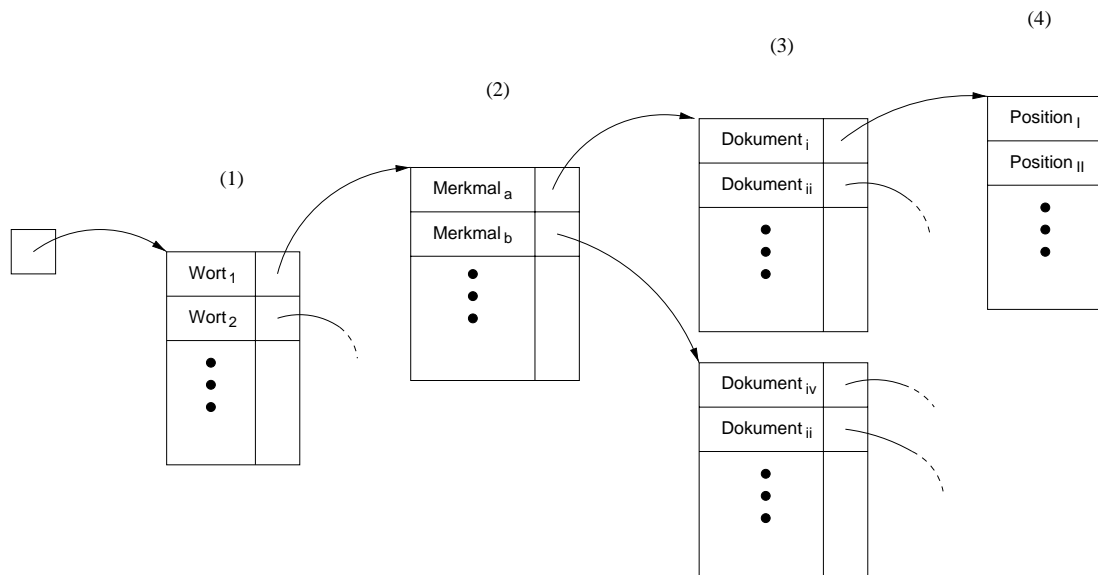


Abbildung 6.2.1: Realisierung des Index

6.2.3 Interne Darstellung des Suchergebnis-Formats

Bei der Erstellung eines Domänen-Experten muß vom Benutzer des generischen Frameworks angegeben werden, welche Informationen zu auf eine Suchanfrage hin ermittelten Elementen der Domäne an den Benutzer übermittelt werden sollen, und wie diese Informationen anzuordnen sind (siehe Abschnitt 4.13).

Diese Angaben speichert der Domänen-Experten in einer internen Darstellung ab, um während der Betriebsphase darauf zurückgreifen zu können.

Abbildung 6.2.3 veranschaulicht das interne Format anhand der ersten Anweisungen des Beispiels von Abbildung 4.13.1 (Seite 57).

Diese interne Darstellung wird vom Objektorientierten Programmierparadigma motiviert, das der für die Implementierung zu verwendenden Programmiersprache JAVA zugrundeliegt.

Im Einzelnen gilt folgendes: Für die Definition des Suchergebnis-Formats können die Kategorien „Bezeichner“, „String“ und „Funktionsaufruf“, sowie die Zeichenfolge `\\` für einen Zeilenumbruch zum Einsatz kommen (siehe Abschnitt 4.13).

Für die Kategorien „Bezeichner“ und „String“ werden Klassen erstellt, deren Instanzen für die interne Darstellung von Elementen der Kategorie verwendet werden. Eine Instanz der Klasse für die Kategorie „Bezeichner“ wird mit der Feldnummer des durch den Bezeichner festgelegten Feldes des Dokumenten-Datensatzes belegt, eine Instanz der Klasse „String“ mit dem String selbst.

Ein Zeilenumbruch wird mit Hilfe einer Instanz der Klasse für die Kategorie „String“ dargestellt, und zwar dadurch, daß die HTML-Anweisung für einen Zeilenumbruch in diese Instanz der Klasse geschrieben wird.

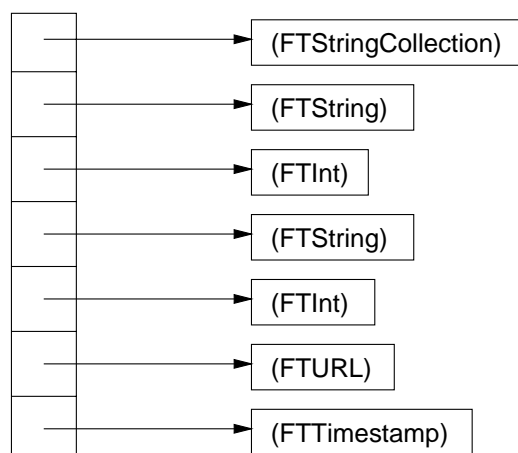


Abbildung 6.2.2: Beispiel für die Realisierung von Dokumenten-Datensätzen

Für die Kategorie „Funktionsaufruf“ wird etwas anders vorgegangen. Jede Funktion wie `\wwwLink` wird durch eine eigene Klasse realisiert. In der internen Darstellung eines Aufrufes der Funktion wird dann eine Instanz der Klasse verwendet, in die die aktuellen Parameter geschrieben werden - bei einem Bezeichner als aktuellem Parameter die Nummer des entsprechenden Feldes des Dokumenten-Datensatzes, bei einem String der String selbst. In der Instanz der Klasse wird auch vermerkt, zu welcher Kategorie („Bezeichner“ oder „String“) jeder aktuelle Parameter gehört; dadurch weiß die Instanz, wie sie die Parameter zu interpretieren hat. Letzteres ermöglicht es, Funktionen zu *überladen* (*Overloading*), d.h. bei einem Parameter verschiedene Kategorien aktueller Parameter akzeptieren zu können.

Auch im Fall der internen Darstellung des Suchergebnis-Formates wird mit Polymorphismus gearbeitet. Die Klassenhierarchie ist in Abbildung 6.2.4 dargestellt. Auf diese Weise können alle Instanzen von Klassen, die zur internen Darstellung des Suchergebnis-Formates dienen, durch einen einheitlichen Methodenaufruf angesprochen werden.

Konkret kann folgendermaßen verfahren werden: Allen Instanzen der internen Darstellung des Suchergebnis-Formates wird nacheinander der Dokumenten-Datensatz des auszugebenden Elementes der Domäne übergeben. Jede Instanz ermittelt darauf die von ihr zu erstellende Ausgabe und gibt diese Ausgabe als HTML-Codestück zurück. Diese Codestücke werden gesammelt, wodurch sich die HTML-Ausgabe für das Element der Domäne ergibt. Dieser Vorgang wird für alle durch die Suchanfrage ermittelten Elemente der Domäne durchgeführt. Dadurch ergibt sich die HTML-Antwortseite.

Man beachte, daß der Dokumenten-Datensatz des einheitlichen Methodenaufrufes wegen an alle Instanzen, die die interne Darstellung des Suchergebnis-Formates realisieren, geschickt wird. Instanzen der Klasse für die Kategorie „String“ benötigen diesen Dokumenten-Datensatz natürlich nicht und ignorieren ihn einfach.

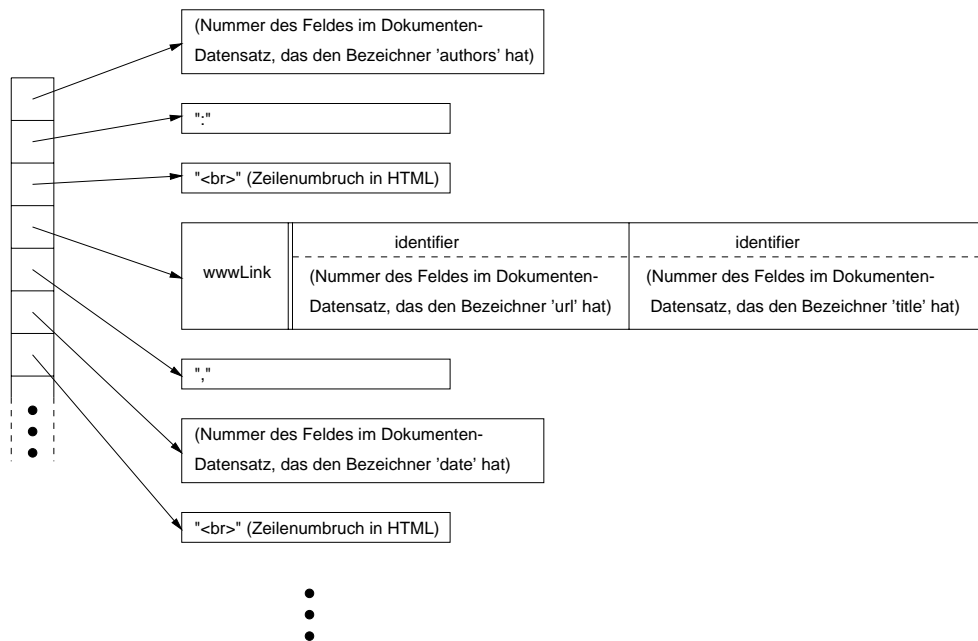


Abbildung 6.2.3: Interne Darstellung des Suchergebnis-Formats für die ersten Anwendungen von Abbildung 4.13.1 auf Seite 57

6.3 Bewertung und Ausblick

In Kapitel 2 wurde das Konzept der Domänen-Experten dargestellt, ab Kapitel 3 mein Konzept für ein generisches Framework, mit dessen Hilfe Domänen-Experten erstellt werden sollen. Die folgenden Unterabschnitte bewerten diese Konzepte. Abschnitt 6.3.4 gibt einen kurzen Ausblick auf weiterführende Aufgabenstellungen.

6.3.1 Bewertung des Konzeptes der Domänen-Experten

In Abschnitt 1.4.3 wurden drei für die Architektur eines effektiven und effizienten Suchwerkzeuges entscheidende Punkte genannt. Das Konzept der Domänen-Experten berücksichtigt jeden dieser drei Punkte und eignet sich daher zur Konstruktion solcher Suchwerkzeuge.

Im einzelnen gibt es drei Gründe dafür, daß das Konzept der Domänen-Experten die Konstruktion *effizienter* Suchwerkzeuge ermöglicht:

1. Durch Verwendung der existierenden Suchmaschine ist es nicht notwendig, daß jeder Domänen-Experte von neuem das komplette World Wide Web durchsucht.
2. Der Einsatz mobiler Filter-Agenten zur Bewertung der Dokumente ist effizienter als das Vorgehen, die Dokumente in den Domänen-Experten zu übertragen und sie dort lokal zu bewerten.

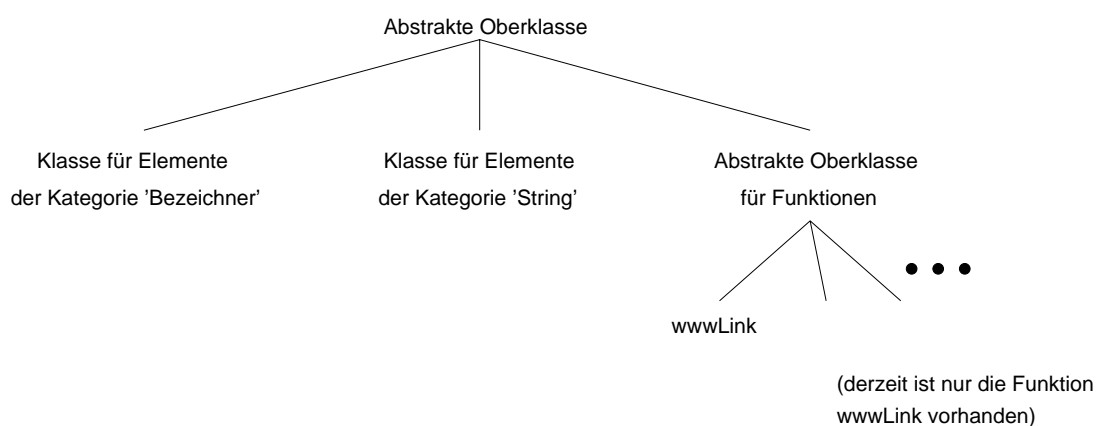


Abbildung 6.2.4: Klassenhierarchie der Klassen für die interne Darstellung des Suchergebnis-Formats

3. Die Beschränkung auf Dokumente einer Domäne begrenzt den Umfang der Wissensbasis. Dadurch ist es möglich, die Informationen alle lokal in einer Datenbasis zu halten und die Informationen auch zu pflegen, d.h. aktuell zu halten.

Die Aspekte des Konzeptes, die zur *Effektivität* von Domänen-Experten am wesentlichsten beitragen, sind:

- Ein Domänen-Experte hat durch die Beschränkung der Dokumente auf eine Domäne schon eine Vorauswahl der Dokumente des World Wide Web getroffen (Genauigkeits-Werte).
- Welche Facetten für ein Dokument berechnet werden, hängt von der Domäne ab. Bei jeder Domäne wird also eine andere Menge von Facetten für die Dokumente berechnet. Dadurch hat der Benutzer die Möglichkeit, präzisere Anfragen zu stellen und präzisere Ergebnisse zu erhalten als dies mit konventionellen Suchmaschinen möglich ist (Genauigkeits-Werte).
- Das Konzept ist flexibel. Dies ermöglicht bei der Konstruktion von Domänen-Experten Anpassungen an bisher noch nicht vorhergesehene Situationen.
- Ein Domänen-Experte sucht im ganzen World Wide Web nach Dokumenten seiner Domäne – sowohl durch eine konventionelle Suchmaschine, als auch proaktiv. Manuelle Eingriffe durch eine Person sind nicht notwendig. (Abruf-Werte)

Ein weiterer Vorteil des Konzeptes der Domänen-Experten ist, daß keine internationalen Standards definiert werden müssen, bevor Domänen-Experten in Betrieb gehen können. Auch können Domänen-Experten Schritt für Schritt eingeführt werden. Schon ein einziger Domänen-Experte ist in der Lage sehr gut zu arbeiten.

6.3.2 Bewertung meines Konzeptes für ein generisches Framework

Das dargestellte Konzept für ein generisches Framework zur Erstellung von Domänen-Experten ist in der Lage, die für den Rahmen meiner Arbeit relevanten Bereiche eines Domänen-Experten durch Generieren zu erzeugen. Manuelle Eingriffe in den Sourcecode der Software sind nicht notwendig, was die Fehlerträchtigkeit bei der Erstellung von Domänen-Experten reduziert und die Erstellung vereinfacht.

Das generische Framework ist hinsichtlich der Datentypen für Dokumenten-Datensätze und deren Vergleichsarten erweiterbar. Ebenso können weitere Funktionen zur Formatierung der Suchergebnisse definiert werden. Solche Funktionen können dabei beliebig viele Parameter besitzen und auch Overloading realisieren. Dies macht das generische Framework anpaßbar und damit flexibel.

Durch Realisierung des Benutzerdialogs über das World Wide Web unter Benutzung des Common Gateway Interfaces kommen in diesem Punkt gängige und benutzerfreundliche Technologien zum Einsatz. Domänen-Experten haben dadurch in diesem Punkt gegenüber heute gängigen Suchwerkzeugen keinen Nachteil. Außerdem ist so ein derzeit gängiger Web-Server die einzige infrastrukturelle Voraussetzung für den Betrieb eines Domänen-Experten im World Wide Web.

Das Konzept ist - soweit sich das aufgrund meiner Untersuchungen sagen läßt - unabhängig von konkreten Sach- oder Anwendungsgebieten für Domänen-Experten.

6.3.3 Gesamtbewertung

Offen ist noch, in wieweit der Domänen-Tester eines Domänen-Experten (der nicht zu meiner Diplomarbeit gehörte) generisch gestaltet und erzeugt werden kann. Sollte die Erstellung des Domänen-Testers nicht generisch möglich sein, so ist ein Weg zu finden, wie ein Benutzer des generischen Frameworks diese Komponente mit angemessenem Aufwand erstellen oder anderweitig erhalten kann.

Ist dieses Problem mit dem Domänen-Tester gelöst und mein Konzept tatsächlich so domänenunabhängig, wie dies aufgrund der eingeschränkten Überprüfung anhand von drei Domänen für den Rahmen dieser Diplomarbeit als gegeben betrachtet wurde, so läßt sich meiner Meinung nach sagen: Das Konzept der Domänen-Experten ist im Zusammenhang mit dem von mir erarbeiteten Konzept für ein generisches Framework ein vielversprechender Ansatz für spezialisierte Suchmaschinen, die Suchergebnisse höherer Qualität liefern als derzeit verfügbare universelle Suchmaschinen.

6.3.4 Ausblick

Mit dem von mir erarbeiteten generischen Framework erstellte Domänen-Experten verfügen nur über einen Teil der in [Theilmann98b] beschriebenen Funktionalität. Aufgabe fortführender Arbeiten wird es sein, das generische Framework diesbezüglich

zu ergänzen. Außerdem werden weiterführende Arbeiten sich damit zu beschäftigen haben, den Zugriff auf verschiedene Domänen-Experten zu erleichtern (siehe Abschnitt 2.7).

6.4 Quellen

Abschnitt 6.3.1 basiert auf [Theilmann98b]. Die HTML-Anweisung für den Zeilenumbruch in Abbildung 6.2.3 stammt aus [SELFHTML/Münz]. Darüber hinaus verwendete Literaturquellen sind an den entsprechenden Stellen angegeben.

Anhang A

Analyse dreier Sachgebiete

A.1 Wissenschaftliche Veröffentlichungen

A.1.1 Merkmale des Sachgebietes

- Autor
- Titel
- Schlüsselwörter
- Institution
- Anzahl Seiten
- Datum
- Zeitmarke
- URL
- Thema (worum es in der Veröffentlichung geht)
- Sprache
- Größe
- Dokumentenformat
- Dateiformat
- Schlüsselwörter zu anderen Arbeiten des Themengebietes (engl. „related work“)
- Literaturverweise

- Autor
- Titel
- Verlag
- Erscheinungsdatum
- Seitenangaben („Seiten x bis y“)
- URL
- Wörter oder Phrasen
 - im ganzen Dokument
 - in der Inhaltsangabe (engl. „abstract“)
 - in der Einführung
 - im Abschnitt, der auf andere Arbeiten des Themengebietes eingeht (engl. „related work“)
 - in der Zusammenfassung

A.1.2 Spezifikation eines Domänen-Experten

Die Spezifikation eines Domänen-Experten aufgrund der in Abschnitt A.1.1 genannten Merkmale ist in Abschnitt 5.3 angegeben.

A.2 Abrufbare public domain Software

A.2.1 Merkmale des Sachgebietes

- Titel der Software
- Version
- Release-Datum
- Zeitmarke
- unterstütztes Betriebssystem
- Dateigröße
- Dateiformat
- Anwendungsgebiet der Software
- Weitere Informationen zur Software

- Hersteller
- URL (Abrufen der Software)
- Schlüsselwörter zur Charakterisierung der Software
- Sprache die die Software zur Benutzerführung verwendet
- URL der WWW-Seite, die den Abruf der Software bereitstellt
- Benutzungsbestimmungen
- Programmiersprachen, die zur Implementierung der Software verwendet wurden

A.2.2 Spezifikation eines Domänen-Experten

```
DomainExpertConfiguration
```

```
    "Domain Expert for Downloadable Public Domain Software";
```

```
BeginMiscSettings
```

```
    ConventionalSearchEngineSearchString =  
        "+^"public domain^" +software";
```

```
    ...
```

```
EndMiscSettings
```

```
BeginIndexWordAttributes
```

```
    details, title
```

```
EndIndexWordAttributes
```

```
BeginDocumentDataRecord
```

```
    title : FTString;  
    version : FTString;  
    dateofrelease : FTDate;  
    timestamp : FTTimestamp;  
    platform : FTString;  
    size : FTInt;  
    fileformat : FTString;  
    topic : FTString;  
    details : FTString;  
    company : FTString;  
    url : FTURL;  
    keywords : FTStringCollection;  
    language : FTString;  
    urldownloadpage : FTURL;  
    purchaseconditions : FTString;  
    writtenin : FTStringCollection
```

EndDocumentDataRecord

BeginGUIDefinition

```
"Title", 40 : indexSearch{title}, conventionalSearchEngineCriterion;
"Version", 10 : documentDataRecordSearch{version}{matching};
"Date of Release", 20 : documentDataRecordSearch{dateofrelease};
"Timestamp", 20 : documentDataRecordSearch{timestamp};
"Platform", 30 : documentDataRecordSearch{platform}{liberal};
"Size (kByte)", 10 : documentDataRecordSearch{size};
"Fileformat", 20 : documentDataRecordSearch{fileformat}{liberal};
"Topic", 40 : documentDataRecordSearch{topic}{matching};
"Details", 40 : indexSearch{details};
"Company", 40 : documentDataRecordSearch{company}{matching};
"URL", 40 : documentDataRecordSearch{url}{matching};
"Keywords", 40 : documentDataRecordSearch{keywords}{liberal};
"Language", 20 : documentDataRecordSearch{language}{liberal};
"URL Downloadpage", 40 :
    documentDataRecordSearch{urldownloadpage}{matching};
"Purchase Conditions", 40 :
    documentDataRecordSearch{purchaseconditions}{matching};
"Written in", 40 : documentDataRecordSearch{writtenin}{liberal}
```

EndGUIDefinition

BeginResultDefinition

```
\wwwLink{url}{title} ", " version ", " dateofrelease \\
company \\
platform ", " size "kByte" \\
topic \\
details \\
\wwwLink{urldownloadpage}{"Download-Page"}
```

EndResultDefinition

EndDomainExpertConfiguration.

A.3 Hotelinformationen

A.3.1 Merkmale des Sachgebietes

- Name des Hotels
- Kategorie des Hotels
- Hotelgruppe, zu der das Hotel gehört

- Land
- Anschrift
- URL der Homepage des Hotels
- URL der Homepage der Hotelgruppe
- Name des Pächters
- Preis Übernachtung min
- Preis Übernachtung max
- Anbindung an öffentlichen Verkehr
- nächstes Zentrum einer Stadt
- Entfernung zu nächstem Zentrum
- Hoteleinrichtungen
- Freizeitangebote Umgebung
- Ausstattung der Zimmer (z.B. Naßzelle)
- Aktualität der Informationen zum Hotel (Zeitmarke)

A.3.2 Spezifikation eines Domänen-Experten

```
DomainExpertConfiguration
    "Domain Expert for Informations about Hotels";

BeginMiscSettings
    ConventionalSearchEngineSearchString = "+hotel";
    ...
EndMiscSettings

BeginIndexWordAttributes
    freetimeenvironsfacilities, hotelfacilities, roomfacilities
EndIndexWordAttributes

BeginDocumentDataRecord
    name : FTString;
    address : FTString;
    category : FTInt;
    hotelgroup : FTString;
    country : FTString;
```

```

url : FTURL;
urlgroup : FTURL;
pricemin : FTInt;
pricemax : FTInt;
pricecurrency : FTString;
traffic : FTStringCollection
distancetocenteroftown : FTInt;
hotelfacilities : FTString;
freetimeenvironsfacilities : FTString;
roomfacilities : FTString;
timestamp : FTTimestamp
EndDocumentDataRecord

BeginGUIDefinition
  "Name", 40 : documentDataRecordSearch{name}{matching}
                , conventionalSearchEngineCriterion;
  "Address", 40 : documentDataRecordSearch{address}{matching};
  "Category", 20 : documentDataRecordSearch{category};
  "Group", 40 : documentDataRecordSearch{hotelgroup}{matching};
                , conventionalSearchEngineCriterion;
  "Country", 20 : documentDataRecordSearch{country}{liberal};
  "Min. Price", 10 : documentDataRecordSearch{pricemin};
  "Max. Price", 10 : documentDataRecordSearch{pricemax};
  "Public Traffic", 40 : documentDataRecordSearch{traffic}{matching};
  "Distance to Center Next Town", 10 :
        documentDataRecordSearch{distancetocenteroftown};
  "Hotel Facilities", 40 : indexSearch{hotelfacilities};
  "Freetime Environs Facilities", 40 :
        indexSearch{freetimeenvironsfacilities};
  "Room Facilities", 40 : indexSearch{roomfacilities};
  "Timestamp of Page", 20 : documentDataRecordSearch{timestamp};
  "Hotel Web Page Contents", 40 : indexSearch{};
EndGUIDefinition

BeginResultDefinition
  \wwwLink{url}{title} ", " category "Star(s)" \\
  address \\
  \wwwLink{urlgroup}{hotelgroup} \\
  "One night:" pricemin "to" pricemax pricecurrency \\
  "Room Facilities:" roomfacilities \\
  "Hotel Facilities:" hotelfacilities \\
  "Freetime Facilities of the Environs:"
        freetimeenvironsfacilities \\
  "Public Traffic:" traffic

```

EndResultDefinition

EndDomainExpertConfiguration.

Anhang B

Zeitplan

Tabelle B.0.1 stellt den ersten Zeitplan für meine Diplomarbeit dar. Er basiert auf der Definition der Arbeitspakete, die für meine Diplomarbeit als inhaltlich abzuarbeiten vorgegeben waren. Der Zeitplan ist so angelegt, daß die notwendigen Arbeitspakete in der vorgegebenen Zeit erstellt werden. Es war jedoch geplant, noch ein optionales Arbeitspaket zusätzlich in diesem Zeitrahmen zu bearbeiten.

Bei der Arbeit zeigte sich, daß eine Gliederung des Vorgehens in oben angegebene Arbeitspakete für die Erstellung des Konzeptes nicht geeignet war. Anfrageauswertung und Dokumentenmodell, Generierung der themenspezifischen Suchmaske und Anfragemodell/Dokumentenmodell z.B. sind, da sie voneinander abhängen, sinnvoller erstellbar, wenn sie dabei gegenseitig berücksichtigt werden. Eine bewußte separate Entwicklung wäre sicherlich möglich, jedoch nicht nötig und würde höchstwahrscheinlich neben zusätzlichem Entwicklungsaufwand auch Effizienzverluste zur Laufzeit nach sich ziehen. Darum erstellte ich das Konzept bei möglichst gleichzeitiger Berücksichtigung aller Arbeitspakete und passte den Zeitplan daran an. Tabelle B.0.2 zeigt den an diese Vorgehensweise angepaßten Zeitplan. Jeder Teilbereich des Konzeptes wurde somit in Abhängigkeit von der Entwicklung der anderen Teilbereiche entworfen.

Zu Tabelle B.0.2 ist zu sagen, daß auf die Aufschlüsselung des letzten Blockes in feinere Arbeitspakete bewußt verzichtet wurde, da zum Zeitpunkt der Erstellung des Zeitplanes (20.5.99) die genaueren Aufgaben und Herausforderungen dieses Teiles noch nicht zu erkennen waren und somit noch nicht genügend Informationen für eine feinere Gliederung dieses Blockes vorlagen. Der Grobentwurf würde dem Abhilfe schaffen, weswegen die Definition feinerer Arbeitspakete auf die Zeit nach dem Grobentwurf verschoben wurde.

Tabelle B.0.3 zeigt den Zeitplan mit Definitionen von Arbeitspaketen für den letzten Block von Tabelle B.0.2.

Arbeitsinhalte	vorgesehener Zeitraum	tatsächlicher Zeitraum	Grund für Abweichung
<ul style="list-style-type: none"> • Einarbeitung in Thematik • Ausarbeitung: allgemeine Einführung in die Thematik und in Domänen-Experten 	1.3.99 - 11.4.99 (6 Wochen)		
Konzept, Implementierung und Ausarbeitung für <ul style="list-style-type: none"> • Erstellung eines generischen Dokumentenmodells (Index und Zusatzinformationen) • Erstellung eines generischen Anfragemodells (typisierte Stichwörter und Operatoren) 	12.4.99 - 5.5.99 (3,5 Wochen, davon Ausarbeitung 0,5 Wochen)		
Konzept, Implementierung und Ausarbeitung für automatische Generierung einer themenspezifischen Suchmaske (inkl. cgi-Anbindung)	6.5.99 - 26.5.99 (3 Wochen, davon Ausarbeitung 0,5 Wochen)		
Konzept, Implementierung und Ausarbeitung für Berechnung des Relevanzgrades eines Dokumentes (vorliegend in obigem Format) bezüglich einer Benutzeranfrage	27.5.99 - 27.6.99 (4,5 Wochen, davon Ausarbeitung 0,5 Wochen)		
Konzept, Implementierung und Ausarbeitung für Abwicklung einer Suche (inkl. Steuerung der Dokumentenuntersuchung unter Ausnutzung vorgegebener Code-Module) und Präsentation der Ergebnisse (Generierung von HTML-Antwortseiten)	28.6.99 - 8.8.99 (6 Wochen, davon Ausarbeitung 1 Woche)		
<ul style="list-style-type: none"> • Ausarbeitung vollends fertigstellen • Abgabe 	9.8.99 - 31.8.99 (3 Wochen)		

Tabelle B.0.1: Zeitplan Stand 21.4.99

Arbeitsinhalte	vorgesehener Zeitraum	tatsächlicher Zeitraum	Grund für Abweichung
<ul style="list-style-type: none"> • Einarbeitung in Thematik • Ausarbeitung: allgemeine Einführung in die Thematik und in Domänen-Experten 	1.3.99 - 11.4.99 (6 Wochen)		
<p>Erarbeitung eines Konzeptes für:</p> <ul style="list-style-type: none"> • Erstellung eines generischen Dokumentenmodells (Index und Zusatzinformationen) • Erstellung eines generischen Anfragemodells (typisierte Stichwörter und Operatoren) • Automatische Generierung einer themenspezifischen Suchmaske (inkl. cgi-Anbindung) • Berechnung des Relevanzgrades eines Dokumentes (vorliegend in obigem Format) bezüglich einer Benutzeraufgabe • Abwicklung einer Suche (inkl. Steuerung der Dokumentenuntersuchung unter Ausnutzung vorgegebener Code-Module) und Präsentation der Ergebnisse (Generierung von HTML-Antwortseiten) • Aufbau eines Indexes zur Speicherung von untersuchten Dokumenten (je inkl. Teile des Entwurfes) 	12.4.99 - 9.5.99 (4 Wochen)		
-Krankheit-	10.5.99 - 16.5.99 (1 Woche)		
Fortsetzung der Erarbeitung des Konzeptes (inkl. Teilen des Entwurfes)	17.5.99 - 30.5.99 (2 Wochen)	17.5.99 - 10.6.99 (3,5 Wochen)	(mehr Aufwand als geschätzt)
<ul style="list-style-type: none"> • Implementierung • Ausarbeitung <p>(Aufteilung dieses Blockes in feinere Arbeitspakete ist nach Grobentwurf noch vorzunehmen.)</p>	31.5.99 - 31.8.99 (13 Wochen)		

Tabelle B.0.2: Zeitplan Stand 20.5.99

Arbeitsinhalte	vorgesehener Zeitraum	tatsächlicher Zeitraum	Grund für Abweichung
Implementierung Konfigurationsfunktionalität bei eingeschränktem Testen	11.6.99 - 24.6.99 (2 Wochen)	(wie geplant)	
Implementierung Betriebsfunktionalität (Anfrageauswertung, Manipulation Datenbestand) bei eingeschränktem Testen	25.6.99 - 11.7.99 (2,5 Wochen)	(wie geplant)	
Ausarbeitung	12.7.99 - 8.8.99 (4 Wochen)	12.7.99 - 30.8.99	mehr Aufwand als geschätzt, Bewerbungsgespräche
<ul style="list-style-type: none"> ● Implementierung fertigstellen <ul style="list-style-type: none"> – Konfigurationsfunktionalität testen (0,5 Wochen) – Betriebsfunktionalität (Anfrageauswertung, Manipulation Datenbestand) fertigstellen und testen (1,5 Wochen + 2 Tage) ● Ausarbeitung fertigstellen (1 Woche) Zeiteinteilung noch offen, da Fertigstellung der Ausarbeitung erst möglich ist, wenn Betreuer die erste Fassung durchgesehen hat.	9.8.99 - 31.8.99 (3 Wochen + 2 Tage)	31.8.99 -	

Tabelle B.0.3: Definition von Arbeitspaketen für den letzten Block des Zeitplanes vom 20.5.99 (Tabelle B.0.2). (Stand 10.6.99)

Literaturverzeichnis

- [AltaVista] AltaVista: <http://www.altavista.com/>
- [Appelrath91] H.-J. APPELRATH, J. LUDEWIG: *Skriptum Informatik: eine konventionelle Einführung*, B. G. Teubner, Stuttgart und Verlag der Fachvereine an den schweizerischen Hochschulen und Techniken, Zürich, 1991
- [Bowman95] C. M. BOWMAN, P. B. DANZIG, D. R. HARDY, U. MANBER, M. F. SCHWARTZ, D. P. WESSELS: *Harvest: A Scalable, Customizable Discovery and Access System*, Technical Report CU-CS-732-94, Department of Computer Science, University of Colorado – Boulder, USA, March 1995
- [Dolin97] R. DOLIN, D. AGRAWAL, A. E. ABBADI, L. DILLON: *Pharos: A Scalable Distributed Architecture for Locating Heterogeneous Information Sources*, Proceedings of the 6th ACM International Conference on Information and Knowledge Management, Las Vegas, Nevada, USA, November 10–14, 1997, F. Golshani and K. Makki (Eds.)
- [Francis95] P. FRANCIS, T. KAMBAYASHI, S. SATO, S. SHIMIZU: *Ingrid: A Self-Configuring Navigation Infrastructure*, Proceedings of the 4th International World Wide Web Conference, Boston, MA, USA, December 11-14, 1995,
URL: http://www.w3.org/Conferences/WWW4/Program_Full.html
- [Grollmuss98] M. GROLLMUSS: *Entwurf einer Plattform für eine Meta-Suchmaschine und Implementierung einer Verwaltung von Suchaufträgen*, Diplomarbeit Nr. 1575, Institut für Parallele und Verteilte Höchstleistungsrechner, Fakultät Informatik, Universität Stuttgart, 1998, Seite 16, letzter Absatz
- [Gudivada97] V. N. GUDIVADA, V. V. RAGHAVAN, W. I. GROSKY, R. KASANAGOTU: *Information Retrieval on the World Wide Web*, IEEE Internet Computing, September/October 1997, pp. 58–68
- [Lycos] Lycos: <http://www-english.lycos.com/>, <http://www.de.lycos.de/>
- [MetaCrawler] MetaCrawler: <http://www.go2net.com/search.html>

- [MetaGer] MetaGer: <http://meta.rrzn.uni-hannover.de/>
- [SELFHTML/Münz] S. MÜNZ: *SELFHTML*, <http://www.netzwelt.com/selfhtml>
<http://www.netzwelt.com/selfhtml/tl.htm>
- [Nitsche–Ruhland95] D. NITSCHÉ–RUHLAND: Folien zur Vorlesung „Hypertext und Hypermedia“ WS 95/96, Abteilung Dialogsysteme, Institut für Informatik, Fakultät Informatik, Universität Stuttgart, WS 95/96, Folien 174, 176 und 177
- [Rothermel98] K. ROTHERMEL, M. SCHWEHM: *Mobile Agents*, In: A. Kent and J. G. Williams (Eds.): *Encyclopedia for Computer Science and Technology*, New York: M. Dekker Inc., 1998 (accepted for publication)
- [Sander–Beuermann98] W. SANDER–BEUERMANN: *Schatzsucher – Die Internet-Suchmaschinen der Zukunft*, c't Magazin für Computer Technik 1998, Heft 13, Heise–Verlag, Hannover, 1998, Seiten 178–184
- [Schmidt98] S. SCHMIDT, O. DIEDRICH: *Interaktiv im Web – CGI–Programmierung für den Hausgebrauch, Teil 1*, c't Magazin für Computer Technik 1998, Heft 24, Heise–Verlag, Hannover, Deutschland, 1998, Seite 226–235
- [Sheldon95] M. A. SHELDON, A. DUDA, R. WEISS, D. K. GIFFORD: *Discover: A Resource Discovery System based on Content Routing*, Proceedings of the 3rd International World Wide Web Conference, Darmstadt, Germany, April 10–14, 1995, Elsevier Science B.V., Journal Computer Networks and ISDN Systems 27:6
- [Theilmann98a] W. THEILMANN: *Informationssuche im World Wide Web*, 5. Berichtskolloquium des Graduiertenkollegs Parallele und Verteilte Systeme, 3. Juli 1998, P. J. Kühn (Ed.), Institut für Nachrichtenvermittlung und Datenverarbeitung (IND), Universität Stuttgart, Seiten 91–100
- [Theilmann98b] W. THEILMANN, K. ROTHERMEL: *Domain Experts for Information Retrieval in the World Wide Web*, CIA–98: Learning, Mobility and Electronic Commerce for Information Discovery on the Internet, Proceedings of the 2nd Int. Workshop on Cooperative Information Agents, Paris, July 4–7, 1998, M. Klusch, G. Weiß (Eds.) *Lecture Notes in Artificial Intelligence* 1435, Springer–Verlag, Berlin, Heidelberg, New York, 1998, pp. 216–227
- [Theilmann99] W. THEILMANN, K. ROTHERMEL: *Maintaining Specialized Search Engines through Mobile Filter Agents*, Proc. 3rd Int. Workshop on Cooperative Information Agents (CIA'99), Uppsala, Sweden, July 31 – August 2, 1999, M. Klusch, O. Shehory, G. Weiß (Eds.), *Lecture Notes in Artificial Intelligence* 1652, Springer, July 1999, pp. 197–208

- [Weider96] C. WEIDER, J. FULLTON, S. SPERO: *Architecture of the Whois++ Index Service*, Network Working Group, RFC 1913, February 96,
URL: <ftp://ds.internic.net/rfc/rfc1913.txt>
- [Yahoo!] Yahoo!: <http://www.yahoo.com/>

Erklärung

Hiermit versichere ich, diese Arbeit selbständig verfaßt und nur die angegebenen Quellen benutzt zu haben.

(Steffen Arnold)